

**DISEÑO E IMPLEMENTACION DE UN SISTEMA DE CONTROL QUE PERMITA
INTEGRARSE CON DIFERENTES TIPOS DE ROBOT MOVILES TERRESTRES**

JOSE LUIS PANIAGUA JARAMILLO

**UNIVERSIDAD AUTONOMA DE OCCIDENTE
FACULTAD DE INGENIERIA
DEPARTAMENTO DE AUTOMATICA Y ELECTRONICA
PROGRAMA DE INGENIERIA ELECTRONICA Y TELECOMUNICACIONES
SANTIAGO DE CALI
2014**

**DISEÑO E IMPLEMENTACION DE UN SISTEMA DE CONTROL QUE PERMITA
INTEGRARSE CON DIFERENTES TIPOS DE ROBOT MOVILES TERRESTRES**

JOSE LUIS PANIAGUA JARAMILLO

**Proyecto de Grado para optar al título de Ingeniero Electrónico y de
Telecomunicaciones**

**Director
PhD. JUAN CAMILO ACOSTA**

**UNIVERSIDAD AUTONOMA DE OCCIDENTE
FACULTAD DE INGENIERIA
DEPARTAMENTO DE AUTOMATICA Y ELECTRONICA
PROGRAMA DE INGENIERIA ELECTRONICA Y TELECOMUNICACIONES
SANTIAGO DE CALI
2014**

Nota de aceptación:

Aprobado por el Comité de Grado en cumplimiento de los requisitos exigidos por la Universidad Autónoma de Occidente para optar al título de Ingeniero Electrónico y de Telecomunicaciones.

ALVARO JOSE ROJAS
Jurado

JESUS LOPEZ
Jurado

Santiago de Cali, 20 de octubre de 2014

AGRADECIMIENTOS

Dr. Juan Camilo Acosta, Director de Proyecto de grado, por su confianza, constante apoyo y excelente orientación.

Ingeniero Jose Fernando Gil, por sus recomendaciones que fueron fundamentales para la realización de este proyecto.

CONTENIDO

| | Pág. |
|--|-----------|
| RESUMEN | 12 |
| INTRODUCCIÓN | 13 |
| 1 ANTECEDENTES | 16 |
| 2 PROBLEMA DE INVESTIGACION | 23 |
| 2.1 PLANTEAMIENTO DEL PROBLEMA | 23 |
| 3 JUSTIFICACIÓN | 24 |
| 4 OBJETIVOS | 25 |
| 4.1 OBJETIVO GENERAL | 25 |
| 4.2 OBJETIVOS ESPECÍFICOS | 25 |
| 5 MARCO TEORICO | 26 |
| 5.1 ANATOMÍA DE UN ROBOT MÓVIL | 26 |
| 5.1.1 Sensor. | 26 |
| 5.1.1.1 Sensores de ultrasonido. | 28 |
| 5.1.1.2 Lidar(Laser Imaging Detection and Ranging). | 29 |
| 5.1.1.3 Encoders. | 30 |
| 5.1.2 Actuador. | 33 |
| 5.1.2.1 Motor de CC. | 33 |
| 5.1.2.2 Servo motor. | 33 |
| 5.1.2.3 Motor paso a paso. | 37 |
| 5.1.3 Locomoción mediante ruedas. | 38 |
| 5.1.4 Cinemática del robot móvil. | 42 |

| | | |
|------------|--|-----------|
| 5.2 | UNIDAD DE PROCESAMIENTO | 52 |
| 5.2.1 | Hardware reconfigurable: FPGA'S. | 53 |
| 5.2.2 | Microprocesadores sin sistema operativo. | 54 |
| 5.2.2.1 | Arduino. | 55 |
| 5.2.2.2 | Psoc. | 56 |
| 5.2.3 | Microprocesadores con Sistema Operativo. ARM. | 56 |
| 5.2.3.1 | Raspberry pi. | 57 |
| 5.2.3.2 | Udoo. | 58 |
| 5.2.3.3 | Beaglebone Black. | 59 |
| 5.3 | LENGUAJES DE PROGRAMACIÓN PARA ROBOTS MÓVILES | 60 |
| 5.3.1 | Java. | 60 |
| 5.3.2 | C/C++. | 60 |
| 5.3.3 | Python. | 61 |
| 5.4 | ENTORNOS DE PROGRAMACIÓN PARA ROBOTS MOVILES. ROS | 61 |
| 5.4.1 | Nodo. | 62 |
| 5.4.2 | Topic. | 62 |
| 5.5 | SLAM | 63 |
| 6 | METODOLOGIA APLICADA | 66 |
| 7 | DISEÑO E IMPLEMENTACION | 67 |
| 7.1 | DISEÑO HARDWARE | 67 |
| 7.2 | DISEÑO SOFTWARE | 70 |
| 7.3 | IMPLEMENTACIÓN HARDWARE | 71 |
| 7.3.1 | Estructura móvil. | 71 |
| 7.3.3 | Control. | 74 |

| | |
|---|------------|
| 7.3.4 Comunicación. | 75 |
| 7.3.5 Alimentación. | 77 |
| 7.4 IMPLEMENTACIÓN SOFTWARE | 78 |
| 7.4.1 Software Tarjeta Arduino. | 78 |
| 7.4.2 Software BeagleBone Black. | 78 |
| 8 PRUEBAS Y RESULTADOS | 81 |
| 9 CONCLUSIONES | 101 |
| BIBLIOGRAFIA | 105 |
| ANEXOS | 109 |

LISTA DE CUADROS

| | Pág. |
|--|-----------|
| Cuadro 1. Sensores más utilizados en la robótica móvil | 27 |
| Cuadro 2. Configuraciones de dos y tres ruedas para robots móviles. | 40 |
| Cuadro 3. Configuraciones de cuatro ruedas para robots móviles. | 41 |
| Cuadro 4. Configuraciones de seis ruedas para robots móviles. | 42 |
| Cuadro 5. Etapas de desarrollo del proyecto. | 66 |
| Cuadro 6. Plataformas Hardware más Adecuadas para el proyecto | 68 |

LISTA DE FIGURAS

| | Pág. |
|---|------|
| Figura 1. a) Robot Sojourner. b) Robot Plustch c) Robot Pioneer | 13 |
| Figura 2. Robot desarrollado por Grey Walter en 1950 | 16 |
| Figura 3. Robot Shakey 1970 | 17 |
| Figura 4. Robot Stanford Cart 1979 | 17 |
| Figura 5. Robots para la exploración de Marte | 18 |
| Figura 6. Robot Prometeo | 19 |
| Figura 7. Robot Giraa_02 | 19 |
| Figura 8. Robot Arcadio | 20 |
| Figura 9. a) Robot móvil. b) Mapa obtenido por el Robot móvil | 21 |
| Figura 10. Prototipo de la Plataforma | 21 |
| Figura 11. Funcionamiento del Ultrasonido | 28 |
| Figura 12. Robot Neato XV-11 | 29 |
| Figura 13. Sensor Lidar | 30 |
| Figura 14. Encoder Incremental | 31 |
| Figura 15. Encoder Absoluto | 32 |
| Figura 16. Servo Motor de Rotación Continua (Interior) | 34 |
| Figura 17. Servo Motor de Rotación Continua (Exterior) | 34 |
| Figura 18. Servo Continuo en Posición Cero | 35 |
| Figura 19. Servo Continuo Girando en Sentido Horario | 36 |
| Figura 20. Servo Continuo Girando en Sentido Anti-Horario | 36 |
| Figura 21. Motor Paso a Paso | 38 |
| Figura 22. Diferentes Tipos de Ruedas | 39 |

| | |
|--|-----------|
| Figura 23. Posición de un Robot Diferencial | 43 |
| Figura 24. Posición del Robot perpendicular al marco de referencia global | 44 |
| Figura 25. Rueda Fija Estándar | 47 |
| Figura 26. Restricciones de una rueda fija Estándar | 48 |
| Figura 27. Rueda Orientable Estándar | 50 |
| Figura 28. a) Estructura de una FPGA. b) Kit UP 2. c) Kit DE-270. | 54 |
| Figura 29. a) Arduino DUE. b) Psoc 4 con Arduino shield | 55 |
| Figura 30. Raspberry Pi | 58 |
| Figura 31. Udoo Quad | 58 |
| Figura 32. BeagleBone Black | 59 |
| Figura 33. Funcionamiento Básico de ROS | 62 |
| Figura 34. Ejemplo del Funcionamiento de ROS | 63 |
| Figura 35. El Problema de Slam | 64 |
| Figura 36. Diagrama del Hardware del Sistema | 69 |
| Figura 37. Diseño Software | 70 |
| Figura 38. Estructura Móvil | 71 |
| Figura 39. Sensor XV11 LIDAR | 72 |
| Figura 40. Contenido de Cada paquete enviado por el sensor LIDAR. | 73 |
| Figura 41. Tarjeta Arduino Mega | 74 |
| Figura 42. Convertidor Lógico de Nivel de Voltaje | 75 |
| Figura 43. Nano router TP – Link | 76 |
| Figura 44. Modulo Bluetooth RN-41 | 77 |
| Figura 45. Regulador BEC (Izq) y Batería lipo(Der) | 77 |
| Figura 46. Formato del Mensaje LaserScan de los Datos del LIDAR | 79 |

| | |
|---|-----------|
| Figura 47. Formato del Mensaje Odometry de los Encoders | 80 |
| Figura 48. Simulador Stage | 83 |
| Figura 49. Mapa resultante con map_saver | 84 |
| Figura 50. Visualización del Proceso en rviz | 85 |
| Figura 51. Prototipo de robot móvil con el sistema de control acoplado | 86 |
| Figura 52. Mapa generado a partir del entorno de la figura 53 | 87 |
| Figura 53. Entorno real empleado en la prueba de la figura 52 | 88 |
| Figura 54. Entorno real empleado en la prueba de la figura 55 | 89 |
| Figura 55. Mapa resultante del entorno de la figura 54 | 89 |
| Figura 56. Visualizacion del Slam en rviz | 91 |
| Figura 57. Primer entorno utilizado en la prueba con hector_salm | 92 |
| Figura 58. Mapa generado con hector_slam | 92 |
| Figura 59. Entorno en forma de L. | 93 |
| Figura 60. Mapa obtenido del entorno en forma de L | 94 |
| Figura 61. Entorno Abierto con obstáculo | 95 |
| Figura 62. Construcción del mapa del entorno de la figura 61 | 95 |
| Figura 63. Mapa obtenido del Laboratorio de Robótica de la UAO | 97 |
| Figura 64. Robot en Configuración Ackerman Axial SCX10 ESC | 98 |
| Figura 65. SLAM con Robot Ackerman a velocidad Constante | 99 |
| Figura 66. SLAM con Robot Ackerman a gran velocidad | 99 |

RESUMEN

En el presente trabajo se presenta el proceso de diseño e implementación de un sistema electrónico basado en sistemas embebidos, capaz de integrarse con diferentes plataformas de robótica móvil y poder ejecutar aplicaciones de un alto nivel de complejidad como lo es SLAM.

Para la realización de este proyecto se empleó una metodología basada en la investigación descriptiva de cada uno de los módulos que conforman el sistema, seguida por una etapa de diseño e implementación, y terminando con una fase de pruebas las cuales ayudaron a medir el alcance de este trabajo.

El uso de nuevas tecnologías ha permitido que el desarrollo hardware aumente en complejidad y al mismo tiempo minimice el tiempo que se toma para lograr una aplicación terminada. Este proyecto es un claro ejemplo de lo anterior, pues gracias al uso de estas nuevas tecnologías y la integración con otras de mayor edad, se pudo obtener un sistema que cumpliera con el objetivo principal, pero que además pudiera ser modificado y quedara abierto para futuros desarrollos. Por otra parte, este proyecto aporta una herramienta de investigación y desarrollo muy útil para universidad, ya que no cuenta con plataformas de este tipo, abiertas en cuanto a hardware software, ya que todas utilizan sistemas operativos con licencias libres como lo son Linux o Android.

Palabras claves: slam, sistema embebido, robótica móvil, ARM, ROS

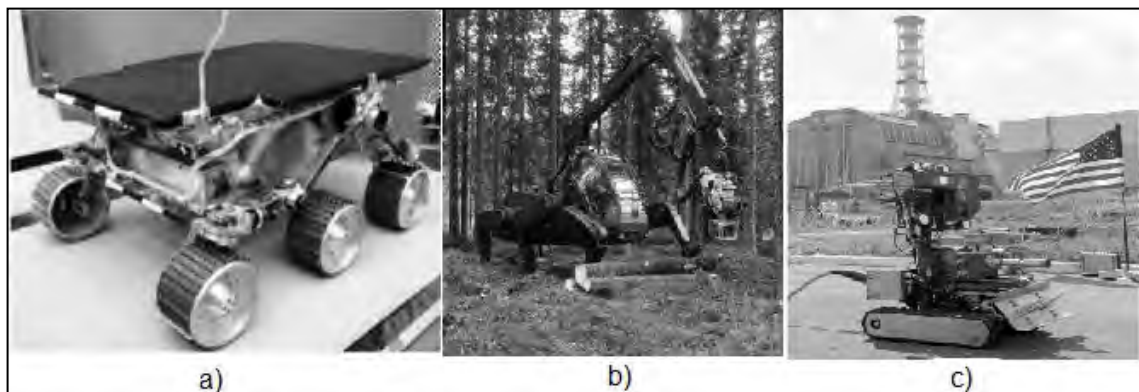
INTRODUCCIÓN

La robótica es sinónimo de progreso y desarrollo tecnológico. Los países y las empresas que cuentan con una fuerte presencia de robots no solamente consiguen altos niveles de competitividad, sino también transmiten una imagen de modernidad. Es por esto que las inversiones en investigación en robótica han aumentado en los últimos años en países como Estados Unidos y Japón¹.

Lo anterior ha permitido que la robótica móvil sea un campo que en los últimos años ha presentado un gran crecimiento en cuanto a desarrollo e investigación. Una de las mayores motivaciones para que este gran desarrollo se diera es su aplicación en diversas áreas de la vida del ser humano como lo son la industria, la academia, la industria aeroespacial, el campo doméstico, el área militar etc.

Estas aplicaciones han permitido la exploración y estudio de terrenos inhóspitos, hostiles y peligrosos para el ser humano como por ejemplo el espacio exterior. La robótica móvil ha hecho posible la creación de plataformas robóticas móviles teleoperadas desde la tierra, las cuales han sido enviadas a Marte como en el caso del robot Sojourner enviado en el verano de 1997 por la NASA. Figura 1.1 a); o las misiones como Opportunity y Spirit en 2004 o el Curiosity en 2009. Todos enviados por la NASA.

Figura 1. a) Robot Sojourner. b) Robot Plustch c) Robot Pioneer



Fuente: tomada de: SIEGWART Roland, Nourbakhsh Illah R. Introduction to Autonomous Mobile Robots. 1 ed. Cambridge, London: The MIT Press, 2004.

¹ COMITÉ Español de Automática. Libro Blanco de la Robotica: De la investigación al desarrollo tecnológico y aplicaciones futuras. [En línea]. Julio de 2008. [Consultado en Agosto de 2013]. Disponible en internet: http://www.ceautomatica.es/sites/default/files/upload/10/files/LIBRO%20BLANCO%20DE%20LA%20ROBOTICA%202_v1.pdf

En otros terrenos se encuentran aplicaciones como el robot desarrollado por Plustech. Figura 1.b) el cual es operado por una persona para la navegación mientras el brazo que se encuentra en la parte superior realiza de forma automática el trabajo de recolección de madera dentro de un bosque de cultivo².

De igual forma, explorar una planta nuclear es otra aplicación de gran impacto de la robótica móvil, un ejemplo de esto es el robot Pioneer el cual fue diseñado para explorar la planta nuclear de Chernobyl, reduciendo a cero el riesgo que implicaría que algún ser humano entrara en contacto directo con las condiciones de este ambiente³.

Un robot móvil es aquel artefacto o dispositivo que puede realizar tareas o procesos automáticos y que cuenta con la capacidad de desplazarse en un entorno determinado⁴. La anterior descripción hace referencia de forma general a lo que es un robot móvil, por lo cual existe una clasificación mucho más específica, la cual define el tipo de robot móvil de acuerdo al ambiente en el que este operara y su tipo de locomoción. De acuerdo a esta clasificación existen robots móviles aéreos, polares, acuáticos y por último los terrestres, los cuales serán objeto de estudio en el presente trabajo.

En cuanto a la clasificación por tipo de locomoción se encuentran robots bioinspirados como humanoides, serpentinos, de ruedas, siendo estos últimos los que se estudiarán en el presente trabajo, puesto que al tener una base móvil el entorno cambia constantemente.

La localización de un robot móvil en un entorno dado es una tarea que requiere de una alta capacidad de cómputo, ya que para estimar lo mejor acertado posible la posición, se debe percibir el entorno y los cambios propios del robot por medio de sensores, procesar esta información y entregar un resultado al mismo robot o un usuario externo.

Esta tarea es considerada actualmente un como un problema de investigación y se torna aún más complejo cuando no solo se requiere estimar la posición, sino que además el robot cumple con otras tareas como por ejemplo generar el mapa de un espacio al mismo tiempo que se estima su localización. A este problema se le conoce como SLAM (Simultaneous Localization And Mapping, o en español:

² SIEGWART Roland, Nourbakhsh Illah R. Introduction to Autonomous Mobile Robots. 1 ed. Cambridge, London: The MIT Press, 2004.

³ Ibid.

⁴ WIKIPEDIA. Robot móvil. [en línea]. [Consultado en agosto de 2013]. Disponible en internet: http://es.wikipedia.org/wiki/Robot_m%C3%B3vil

Localización Y Mapeado Simultáneos) y es uno de los temas más complejo pero atractivo para los investigadores en esta área actualmente en el mundo. Un ejemplo de esto es el trabajo realizado por Hugh Durrant Whyte y Tim Bailey⁵.

El presente trabajo estará conformado por un sistema central de procesamiento robusto capaz de soportar algoritmos complejos como lo es el SLAM. Un subsistema de comunicación inalámbrico con un computador para tener un monitoreo del funcionamiento del robot. Además de etapas de potencia y adquisición de datos que permita el control sobre los actuadores y una óptima percepción del ambiente en el que operara el robot. Todo lo anterior se situara en el robot, haciendo que se evite el uso de algún otro hardware externo, pues normalmente se recurre a esto a causa del costo computacional.

Además la plataforma quedará abierta y será modular para que en ella se puedan desarrollar diversas aplicaciones, cumpliendo con el objetivo de dotar a la Universidad Autónoma de Occidente con una herramienta de investigación acorde con la necesidad actual tanto de estudiantes como de temas de investigación en el área de la robótica móvil.

⁵ DURRANT Whyte, Hugh; Fellow; IEEE; Bailey, Tim. Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. [En Línea]. [Consultado en marzo de 2014]. Disponible en internet: http://www.cs.berkeley.edu/~pabbeel/cs287-fa09/readings/Durrant-Whyte_Bailey_SLAM-tutorial-I.pdf

1 ANTECEDENTES

A partir de los años 50 comenzaron a conocerse desarrollos en el área de la robótica móvil en las principales universidades de alrededor del mundo, siendo la universidad de Bristol y la universidad de Stanford las pioneras en la investigación y desarrollo de estas plataformas.

Figura 2. Robot desarrollado por Grey Walter en 1950



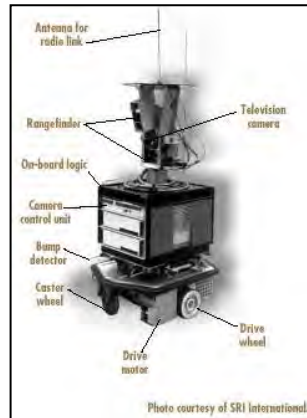
Fuente: tomada de: LA MAQUINA DE VON NEUMANN. Las tortugas de Grey Walter. [En línea]. [consultado en agosto de 2013]. Disponible en internet: <http://vonneumannmachine.wordpress.com/2011/05/01/las-tortugas-de-grey-walter/>

Esta plataforma tiene como principales características un sensor de luz, motor para de propulsión y motor de dirección, un sensor de contacto y un tubo de vacío de cómputo.

En los años 70 se presentó un gran avance en el desarrollo de robots móviles con la aparición del robot Shakey desarrollado por el SRI (Stanford Research Institute), el cual fue el primer robot móvil de propósito general capaz de razonar y tomar sus propias decisiones para realizar acciones. Entre sus principales características está el reconocimiento y la ubicación de objetos⁶.

⁶ ARTIFICIAL INTELLIGENCE CENTER. Shakey. [En línea]. [consultado en agosto de 2013]. Disponible en internet: <http://www.ai.sri.com/shakey/>

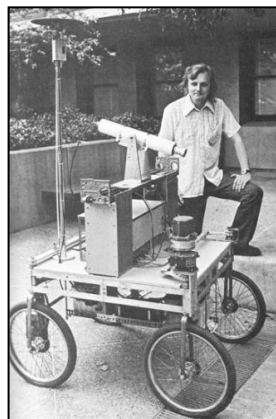
Figura 3. Robot Shakey 1970



Fuente: tomada de: ARTIFICIAL INTELLIGENCE CENTER. Shakey. [En línea]. [consultado en agosto de 2013]. Disponible en internet: <http://www.ai.sri.com/shakey/>

A finales de los años 70 y principios de los 80 siguieron apareciendo grandes avances con la invención de plataformas como el Stanford cart desarrollado en la universidad de Stanford, la cual revolucionó aspectos como el tamaño y tipo de locomoción, e introdujo el desarrollo de algoritmos para determinar caminos más cortos para llegar a un punto. Entre sus principales características esta una operación constante durante 5 horas y la navegación con éxito evitan obstáculos que se encuentre dentro del campo visual de su sensor.

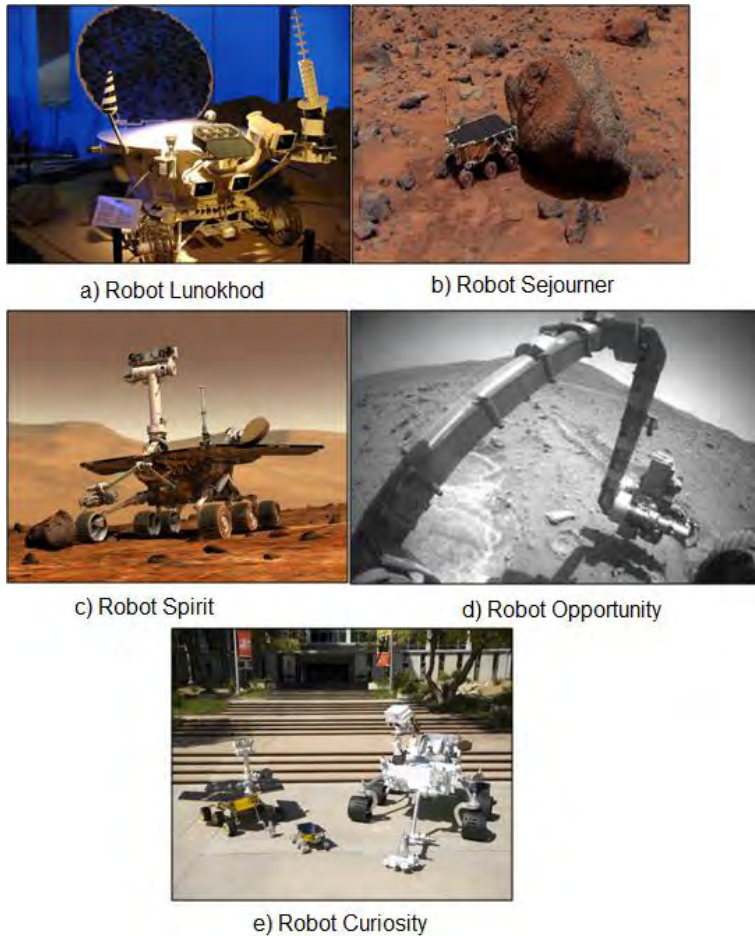
Figura 4. Robot Stanford Cart 1979



Fuente: tomada de: STANFORD UNIVERSITY. Stanford Cart. [En línea]. [Consultado en agosto de 2013]. Disponible en internet: <http://www.stanford.edu/~learnest/cart.htm>

De esta manera fueron surgiendo uno tras otro, nuevos robots móviles con diferentes aplicaciones y nuevas mejoras que iban de la mano con el desarrollo tecnológico y la evolución de los dispositivos electrónicos hasta llegar a la actualidad con los llamados ROVERS, los cuales fueron desarrollados por la NASA para una misión de exploración al planeta Marte, teniendo como predecesor al LUNOKHOD desarrollado en 1970 – 1973 por la Unión Soviética para una misión de exploración a la luna. Este conjunto de robots desarrollado por la NASA comenzó en 1997 con el robot SEJOURNER, seguido por el SPIRIT y el OPPORTUNITY en 2004 y terminando con el que hoy conocemos con el nombre de CURIOSITY⁷.

Figura 5. Robots para la exploración de Marte

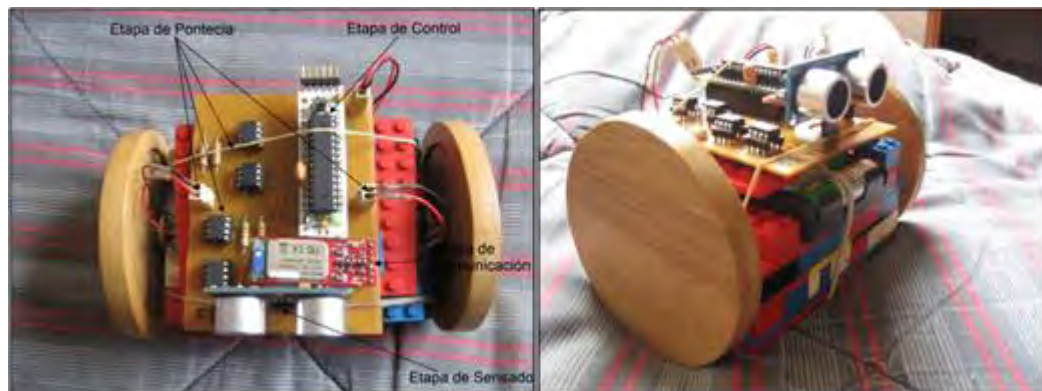


Fuente: tomada de: NASA. Nasa Education Robotics. [En línea]. [Consultado en agosto de 2013]. Disponilbe en internet: <http://www.nasa.gov/>

⁷ NASA. Nasa Education Robotics. [En línea]. [Consultado en agosto de 2013]. Disponilbe en internet: <http://www.nasa.gov/>

A nivel nacional se encuentran varios desarrollos realizados por importantes universidades del país. El robot Prometeo, el cual surge de un proyecto llamado "sistema de robot móvil para mapeo en 2 dimensiones" desarrollado por EDGAR MUNOZ, JESUS ARTURO QUINTERO Y JOSE LUIS PANIAGUA en la institución Universitaria Antonio José Camacho, fue una aplicación orientada hacia el mapeo en dos dimensiones de espacios controlados, la cual contaba con una comunicación por medio de bluetooth con un computador, en el cual el usuario puede monitorear la identificación realizada por el robot.

Figura 6. Robot Prometeo



Una aplicación de mayor complejidad fue el robot Giraa_02 realizado en la universidad de Antioquia, el cual se obtuvo gracias al desarrollo del proyecto llamado "Diseño y Construcción de un robot móvil orientado a la enseñanza e investigación". Esta plataforma fue desarrollada por el grupo de investigación en robótica y Mecatrónica, y como su nombre permite desarrollar conocimientos en el campo de la robótica móvil.

Figura 7. Robot Giraa_02



Fuente: tomado de: MUÑOZ David, Andrade Carlos, Londoño Nelson. Diseño y Construcción de un Robot Móvil Orientado a la Enseñanza e Investigación [en línea]. Abril de 2006. [Consultado en agosto de 2013]. Disponible en internet: <http://rcientificas.uninorte.edu.co/index.php/ingenieria/article/viewFile/2490/1632>

Por otra parte se encuentran plataformas orientas a otros campos como el militar. Un ejemplo de esto es el robot Arcadio desarrollado por el grupo de investigación de sistemas inteligentes, Robótica y Percepción (Sirp) de la universidad Javeriana, el cual tiene como función detectar minas antipersonales en un campo hostil para las fuerzas militares y la sociedad en general.

Figura 8. Robot Arcadio



Fuente: tomada de : MANUELA VALLEJO. Inventos Colombianos En El Campo De La Robótica. [En línea]. [Consultado en agosto de 2013]. Disponible en internet: <http://manuelavallejo.jimdo.com/la-robotica-en-colombia/>

A nivel local en la universidad existen proyectos afines con el área de la robótica móvil. El proyecto desarrollado por el ingeniero mecatronico JOSE FERNANDO GIL, el cual se titula “Un Acercamiento Entre Ros, Kinect Y Una Plataforma Móvil Propia Para La Ejecución De Técnicas SLAM” que consiste en un robot móvil que permite probar diferentes algoritmos de mapeo y localización simultáneos por medio de un programa Middleware llamado “Robotics Operative System (ROS)”. Como se observa en la figura 1.9a para lograr la ejecución del SLAM fue necesario utilizar un computador personal, por lo que el presente trabajo ofrece una opción alternativa mucho más económica y ajustada a los requerimientos del SLAM, pues un computador utilizado solo para realizar esta tarea, desde el punto de vista de la ingeniería está sobredimensionado.

Figura 9. a) Robot móvil. b) Mapa obtenido por el Robot móvil



Fuente: tomada de: GIL. Jose. Un Acercamiento Entre Ros, Kinect Y Una Plataforma Móvil Propia Para La Ejecución De Técnicas SLAM. Universidad Autónoma de Occidente – Cali 2012

Por último se tiene la plataforma desarrollada por LINDA SOLANYI PÉREZ OCHOA y MARIO FERNANDO ERASO en el proyecto que tiene como nombre “Diseño y construcción de un robot móvil terrestre que sirva de plataforma de investigación en el área de robótica móvil en ambientes abiertos y cerrados”, el cual se encuentra en desarrollo.

Figura 10. Prototipo de la Plataforma



Cabe resaltar que existen muchos más antecedentes a nivel nacional e internacional en el campo de la robótica móvil que aquí no se han referenciado, esto debido a que como se expuso al comienzo de este trabajo, existen robots móviles de diferentes tipos como los robots bioinspirados, de patas, serpentinos etc. Estas plataformas hacen parte de una clasificación diferente a las plataformas que este trabajo se enfoca, las cuales poseen locomoción por medio de ruedas.

Es por esta razón que los antecedentes presentados hacen referencia a plataformas de este tipo.

En cuanto a la parte electrónica, los antecedentes están ligados al desarrollo y evolución que los dispositivos electrónicos han tenido a lo largo de la historia. Los primeros robots fabricados eran completamente mecánicos, pero con la aparición de la tecnología electrónica nacieron una nueva generación de robots controlados electrónicamente. La aparición de los semiconductores y por ende la evolución de los microprocesadores produjo que la electrónica y la mecánica se fusionarán en una sola rama llamada robótica.

2 PROBLEMA DE INVESTIGACION

2.1 PLANTEAMIENTO DEL PROBLEMA

SLAM (Simultaneous Localization And Mapping) es una tarea que requiere de una alta capacidad de cómputo si se quieren tener buenos resultados [Simultaneous Localisation and Mapping (SLAM): Part II State of the Art, Tim Bailey and Hugh Durrant-Whyte].

Actualmente la Universidad Autónoma de Occidente cuenta con plataformas de robótica móvil como el LEGO NXT y VEX los cuales hacen limitado el desarrollo de aplicaciones de alta complejidad por su capacidad de cómputo. Estas plataformas permiten desarrollar gran variedad de aplicaciones de forma rápida y sencilla, pues son plataformas que cuentan con diversos tipos de sensores y actuadores que pueden ser fácilmente adaptados a un cerebro o unidad central de procesamiento. Pero a medida que el nivel de complejidad de los desarrollos aumenta, dichas plataformas ya no aplican de forma óptima o en algunos casos no sirven para desarrollar estos proyectos debido a sus características mecánicas o capacidad de cómputo de su electrónica.

Una tarea como SLAM es difícil de lograr con un kit de desarrollo como el LEGO NXT, esto debido a que el hardware o parte electrónica no cuenta con la capacidad de cómputo para llevar a cabo este tipo de tareas, o por su naturaleza no permite realizar implementaciones como ejecutar un sistema operativo.

Por tal motivo surge la siguiente pregunta: ¿Cómo diseñar e implementar un sistema electrónico que permita controlar diferentes tipos de robots móviles en la UAO y que además permita ejecutar tareas que requieran una alta capacidad de cómputo?

3 JUSTIFICACIÓN

Este proyecto suple la necesidad de una plataforma hardware de robótica móvil que permita el desarrollo y ejecución de aplicaciones de mayor complejidad de forma local (en el mismo robot) que contribuyan a la investigación en el campo de la robótica móvil en la universidad, pues solamente se cuenta con plataformas como LEGO NXT la cual debido a sus propias características técnicas limitan el desarrollo de sistemas de robótica móvil.

Por otro lado es de gran utilidad debido a que es una plataforma en la cual estudiantes de otras ingenierías como mecánica, mecatrónica, sistemas y multimedia pueden intervenir para futuros desarrollos o mejoramiento de la misma.

Además de la importancia que representa una plataforma de este tipo dentro del ámbito académico, está comprobado que actualmente se vienen presentando necesidades a nivel internacional como la exploración en otros planetas y el espacio exterior, y la búsqueda de fuentes de recursos naturales en lugares de alto riesgo e inaccesibles para el ser humano. A nivel nacional se tienen problemáticas como la desactivación de minas antipersonales y la exploración en zonas de desastre, lo cual abre una oportunidad para contribuir no solo en lo académico, para los estudiantes de la universidad interesados en desarrollar su conocimiento en el campo de la robótica móvil.

El solo hecho de que la robótica móvil sea aplicable a una gran cantidad de campos, hace que el incentivo por desarrollar esta plataforma aumente, pues es fundamental que desde la ingeniería en general y en especial la electrónica, se pueda contribuir con el desarrollo y bienestar de la sociedad.

4 OBJETIVOS

4.1 OBJETIVO GENERAL

- Diseñar e implementar un sistema electrónico que permita controlar diferentes tipos de robots móviles terrestres.

4.2 OBJETIVOS ESPECÍFICOS

- Investigar plataformas hardware que permitan manejar la odometría y controlar los actuadores de diferentes tipos de robots móviles, manteniéndola abierta a mas sensores y actuadores.
- Seleccionar una plataforma hardware que sea abierta a sensores y actuadores en cuanto a tipo y cantidad.
- Diseñar la plataforma hardware requerida para el procesamiento de la información entregada por los sensores y actuadores.
- Diseñar e implementar un sistema de comunicaciones inalámbrico para monitoreo externo.
- Validar el funcionamiento de la plataforma con dos robots móviles con morfologías diferentes de la UAO.
- Validar el funcionamiento por medio de un algoritmo de mapeo y localización.
- Generar la documentación necesaria para su posterior utilización.

5 MARCO TEORICO

5.1 ANATOMÍA DE UN ROBOT MÓVIL

Un robot móvil se caracteriza por poseer algún mecanismo que le permita desplazarse por un determinado ambiente. Estos mecanismos son determinantes a la hora de establecer el diseño que tendrá el robot teniendo en cuenta el ambiente en el cual este operara.

Actualmente se pueden encontrar robots equipados con ruedas, extremidades, orugas etc., los cuales permiten el movimiento de acuerdo a las órdenes que reciben desde la unidad de procesamiento de la plataforma.

Un robot móvil puede ser visto como un sistema conformado por múltiples etapas o subsistemas, los cuales cumplen con una función específica dentro de la tarea que el robot desarrolla. Dentro las principales etapas que conforman un plataforma de robótica móvil se encuentran sensores, actuadores y unidades de procesamiento. Además de la estructura mecánica, la cual soporta las etapas anteriormente mencionadas.

5.1.1 Sensor. es un dispositivo capaz de convertir una variable física o química en una variable eléctrica. Actualmente existe gran variedad de estos dispositivos, pero no todos aplican o son utilizados en el campo de la robótica móvil.

Los sensores utilizados en esta área se encuentran clasificados en dos importantes grupos: propioceptivos/exterioceptivos y activos/pasivos. Los sensores propioceptivos son aquellos que miden valores internos del sistema (robot móvil) como por ejemplo la velocidad del motor de una rueda, el voltaje de la batería, el grado de inclinación de un brazo etc.

Los sensores exterioceptivos en cambio, se encargan de tomar mediciones de variables externas (ambiente) al robot; por ejemplo la distancia a la cual el sistema se encuentra de un objeto, la intensidad de luz, la humedad etc⁸.

Por otro lado, los sensores pasivos, como su nombre lo indica, absorben energía del ambiente que rodea al robot, para de esta forma brindar información a la

⁸ SIEGWART Roland, Nourbakhsh Illah R. Introduction to Autonomous Mobile Robots. 1 ed. Cambridge, London: The MIT Press, 2004.

unidad de procesamiento. Dentro de esta clasificación se encuentran sensores como las sondas de temperatura, micrófonos y cámaras CCD o CMOS.

Los sensores activos por el contrario, emiten energía hacia el ambiente con el fin de obtener alguna reacción que les permita determinar alguna medición. Este tipo de sensor suele tener mejor rendimiento que uno pasivo, pero existen inconvenientes como la interferencia con otras señales propias o externas del robot. Un ejemplo de este tipo de sensor es el ultrasonido, el cual emite una onda sonora y espera que esta se refleje en alguna parte del ambiente para de esta forma obtener una medición de distancia.

En el Cuadro 1 se muestra un resumen de los sensores más utilizados en la robótica móvil.

Cuadro 1. Sensores más utilizados en la robótica móvil

| General classification (typical use) | Sensor Sensor System | PC or EC | A or P |
|---|------------------------------|-------------|--------|
| Tactile sensors (detection of physical contact or closeness; security switches) | Contact switches, bumpers | EC | P |
| | Optical barriers | EC | A |
| | Noncontact proximity sensors | EC | A |
| Wheel/motor sensors (wheel/motor speed and position) | Brush encoders | PC | P |
| | Potentiometers | PC | P |
| | Synchros, resolvers | PC | A |
| | Optical encoders | PC | A |
| | Magnetic encoders | PC | A |
| | Inductive encoders | PC | A |
| | Capacitive encoders | PC | A |
| Heading sensors (orientation of the robot in relation to a fixed reference frame) | Compass | EC | P |
| | Gyroscopes | PC | P |
| | Inclinometers | EC | A/P |
| Ground-based beacons (localization in a fixed reference frame) | GPS | EC | A |
| | Active optical or RF beacons | EC | A |
| | Active ultrasonic beacons | EC | A |
| | Reflective beacons | EC | A |
| Active ranging (reflectivity, time-of-flight, and geo- metric triangulation) | Reflectivity sensors | EC | A |
| | Ultrasonic sensor | EC | A |
| | Laser rangefinder | EC | A |
| | Optical triangulation (1D) | EC | A |
| | Structured light (2D) | EC | A |
| Motion/speed sensors (speed relative to fixed or moving objects) | Doppler radar | EC | A |
| | Doppler sound | EC | A |
| Vision-based sensors (visual ranging, whole-image analy- sis, segmentation, object recognition) | CCD/CMOS camera(s) | EC | P |
| | Visual ranging packages | | |
| | Object tracking packages | | |

A, active; P, passive; P/A, passive/active; PC, proprioceptive; EC, exteroceptive.

Fuente: tomada de SIEGWART Roland, Nourbakhsh Illah R. Introduction to Autonomous Mobile Robots. 1 ed. Cambridge, London: The MIT Press, 2004.

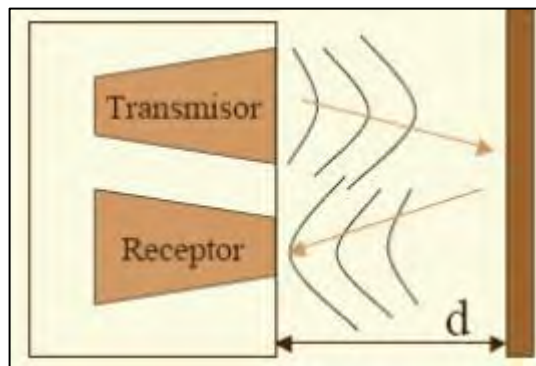
Para las todas las aplicaciones y tareas que se desarrollen en robótica móvil se hace necesario el uso de sensores y en especial para tareas como SLAM es importante disponer con sensores que permitan determinar la distancia del robot a un objeto.

5.1.1.1 Sensores de ultrasonido. En robótica móvil, los sensores de ultrasonido son muy utilizados como elemento principal de interacción del robot con su entorno. Entre las aplicaciones más comunes se encuentran los robots móviles que evaden obstáculos. Estos hacen uso de las funcionalidades que este sensor brinda para evitar que el robot se choque mientras sigue una trayectoria.

Los ultrasonidos son antes que nada sonido, exactamente igual que los que oímos normalmente, salvo que tienen una frecuencia mayor que la máxima audible por el oído humano. Ésta comienza desde unos 16 Hz y tiene un límite superior de aproximadamente 20 KHz, mientras que nosotros vamos a utilizar sonido con una frecuencia de 40 KHz. A este tipo de sonidos es a lo que llamamos Ultrasonidos⁹.

El funcionamiento básico de los ultrasonidos como medidores de distancia se muestra de una manera muy clara en el siguiente esquema, donde se tiene un receptor que emite un pulso de ultrasonido que rebota sobre un determinado objeto y la reflexión de ese pulso es detectada por un receptor de ultrasonidos:

Figura 11. Funcionamiento del Ultrasonido



Fuente: tomada de: PÉREZ, Diego. Sensores de Distancia por Ultrasonidos. [En línea]. [Consultado en enero de 2014]. Disponible en internet: <http://www.alcabot.com/alcabot/seminario2006/Trabajos/DiegoPerezDeDiego.pdf>

⁹ PÉREZ, Diego. Sensores de Distancia por Ultrasonidos. [En línea]. [Consultado en enero de 2014]. Disponible en internet: <http://www.alcabot.com/alcabot/seminario2006/Trabajos/DiegoPerezDeDiego.pdf>

La mayoría de los sensores de ultrasonido de bajo coste se basan en la emisión de un pulso de ultrasonido cuyo lóbulo, o campo de acción, es de forma cónica. Midiendo el tiempo que transcurre entre la emisión del sonido y la percepción del eco se puede establecer la distancia a la que se encuentra el obstáculo que ha producido la reflexión de la onda sonora, mediante la fórmula:

$$d = \frac{1}{2} V \cdot t$$

Ecuación 1. Fórmula para determinar distancia en un sensor ultrasónico

Donde V es la velocidad del sonido en el aire y t es el tiempo transcurrido entre la emisión y recepción del pulso¹⁰.

5.1.1.2 Lidar(Laser Imaging Detection and Ranging). Los sensores lidar o simplemente lidar, tienen su mayor aplicación en áreas como la geología, sismología y física de la atmosfera. Pero debido al aumento en la complejidad de las tareas que los robots móviles actualmente realizan, esta tecnología se ha convertido en una de las más usadas en este campo. Robots móviles domesticos como el NEATO utilizan como sensor de percepción de su entorno un sensor lidar.

Figura 12. Robot Neato XV-11

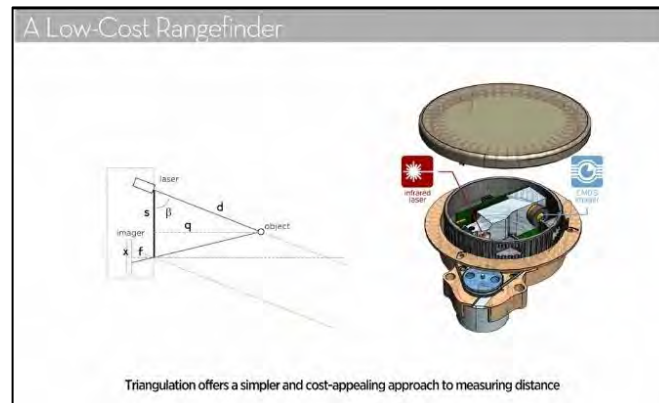


Fuente: Tomada de: NEATO HACKING. Neato. [En línea]. 2014. [Consultado en Abril de 2014]. Disponible en Internet: <http://xv11hacking.wikispaces.com/>

¹⁰ PÉREZ, Diego. Sensores de Distancia por Ultrasonidos. [En línea]. [Consultado en enero de 2014]. Disponible en internet: <http://www.alcabot.com/alcabot/seminario2006/Trabajos/DiegoPerezDeDiego.pdf>

Estos sensores han ido desplazando paulatinamente el uso ultrasonidos, dejándolos replegados para ser usados en aplicaciones no tan complejas. Esto debido a la necesidad de una mejor forma de adquirir información del entorno, en cuanto a cantidad de datos y precisión.

Figura 13. Sensor Lidar



Fuente: tomada de: ROBOT REVIEWS. Robot chat. [En línea]. Julio de 2011. [Consultado en abril de 2014]. Disponible en internet: <http://www.robotreviews.com/chat/viewtopic.php?f=20&t=15016>

El funcionamiento de este tipo de sensores se basa en el mismo principio en que se basan los radares, con la diferencia que el lidar utiliza la luz en forma de laser pulsado, en lugar de ondas de radio.

Para obtener la distancia el lidar emite un haz laser pulsado hacia la superficie u objeto deseado y mide el tiempo de retraso entre la emisión del pulso y la detección de la señal reflejada.

Aplicaciones como SLAM necesitan estimar la posición del robot. Para ello se hace necesario el uso de otro tipo de sensor como lo son los encoders.

5.1.1.3 Encoders. En robótica móvil es fundamental conocer la posición del robot con respecto a un punto de referencia inicial, pues este es un dato importante para desarrollar tareas como navegación autónoma, seguimiento de trayectorias y más aún para mapeo y localización.

Conocer la posición con exactitud es prácticamente imposible y lo máximo que se puede lograr es una estimación por medio de la cantidad de giros de las ruedas

del robot. Esto aplica para robots basados en ruedas, ya que para robots con otro tipo de locomoción existen otras formas de estimar su posición.

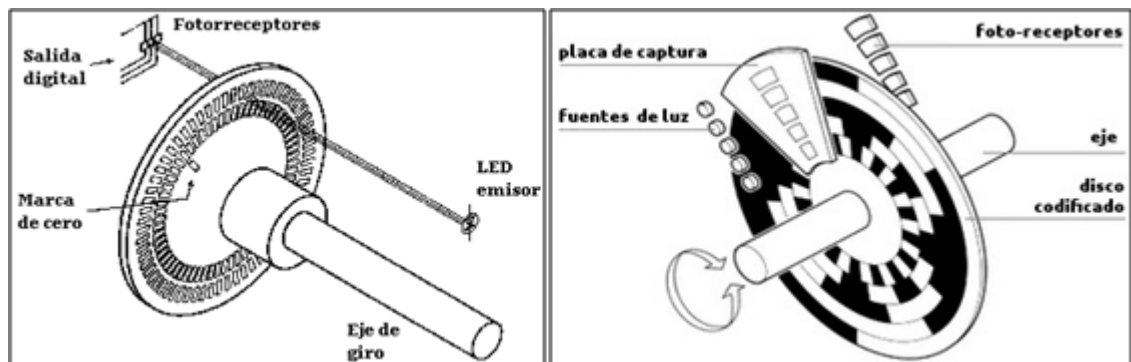
Para obtener la estimación de la posición se utilizan encoders y en especial encoders incrementales. Un encoder es un sensor que se utiliza en diversas áreas para contar las revoluciones de un eje. En robotica móvil se utiliza para contar las revoluciones en los ejes de cada una de las ruedas y se encuentra clasificado dentro del grupo de los sensores propioceptivos, pues brinda información acerca de un estado interno del robot.

Los encoder se acoplan a los ejes de las ruedas y dan cuenta de la cantidad de revoluciones o vueltas que estos han dado en un determinado tiempo. Estos dispositivos están contruidos con diferentes tecnologías y existen de varios tipos.

El funcionamiento de un ecoder consiste en que un disco gira, con zonas transparentes y opacas que interrumpen un haz de luz captado por foto-receptores, luego estos transforman los impulsos luminosos en impulsos eléctricos los cuales son tratados y transmitidos por la electrónica de salida. Como se dijo anteriormente existen varios tipos de encoders y en robotica móvil son muy utilizados 2 tipos, los incrementales y los absolutos.

El encoder incremental se caracteriza porque determina su posición contando el número de impulsos que se generan cuando un rayo de luz es atravesado por las marcas opacas de la superficie de un disco unido al eje.

Figura 14. Encoder Incremental



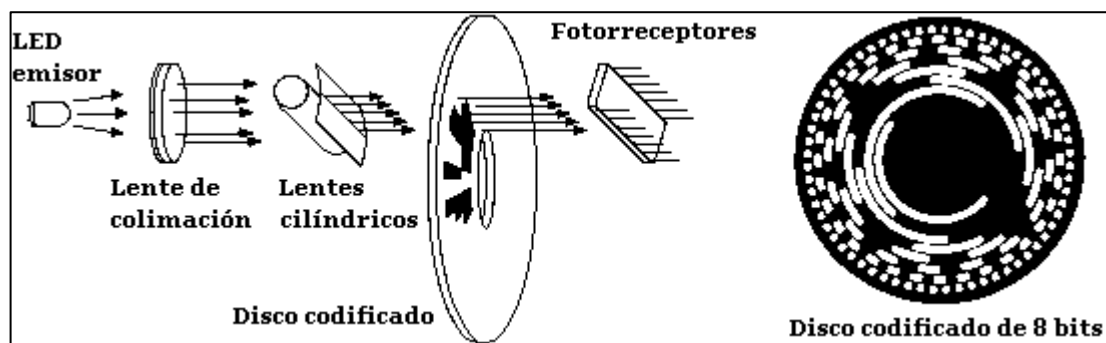
Fuente: tomada de: GARCÍA. Belmar. Diseño y Construcción de un Robot móvil controlado con tenicas de lógica difusa implementadas en un FPGA. [En línea]. Septiembre de 2012. [Consultado en enero de 2014]. Disponible en internet: <http://www.sepi.esimez.ipn.mx/electronica/archivos/992.pdf>

En el estator hay como mínimo dos pares de foto-receptores ópticos, escalados un número entero de pasos más $\frac{1}{4}$ de paso. Al girar el rotor genera una señal cuadrada, el escalado hace que las señales tengan desfase de $\frac{1}{4}$ de periodo si el rotor gira en un sentido y de $\frac{3}{4}$ si gira en el sentido contrario, lo que se utiliza para discriminar el sentido de giro.

Un simple sistema lógico permite determinar desplazamientos a partir de su origen, a base de contar impulsos de un canal y determinar el sentido de giro a partir del desfase entre los dos canales. Algunos encoders pueden disponer de un canal adicional que genere un pulso por vuelta y la lógica puede dar número de vueltas por fracción de vuelta. La resolución del encoder depende del número de impulsos por revolución¹¹.

Por otra parte, en el encoder absoluto, el disco contiene varias bandas dispuestas en forma de coronas circulares concéntricas, dispuestas de tal forma que en sentido radial el rotor queda dividido en sectores, con marcas opacas y transparentes codificadas en código Gray.

Figura 15. Encoder Absoluto



Fuente: tomada de: GARCÍA. Belmar. Diseño y Construcción de un Robot móvil controlado con técnicas de lógica difusa implementadas en un FPGA. [En línea]. Septiembre de 2012. [Consultado en enero de 2014]. Disponible en internet: <http://www.sepi.esimez.ipn.mx/electronica/archivos/992.pdf>

El estator tiene un foto-receptor por cada bit representado en el disco. El valor binario obtenido de los foto-receptores es único para cada posición del rotor y representa su posición absoluta. Se utiliza el código Gray en lugar de un binario

¹¹ GARCÍA. Belmar. Diseño y Construcción de un Robot móvil controlado con técnicas de lógica difusa implementadas en un FPGA. [En línea]. Septiembre de 2012. [Consultado en enero de 2014]. Disponible en internet: <http://www.sepi.esimez.ipn.mx/electronica/archivos/992.pdf>

clásico porque en cada cambio del sector solo cambia el estado de una de las bandas, evitando errores por falta de alineación de los captadores¹².

5.1.2 Actuador. En cuanto a los actuadores, se suelen usar motores eléctricos de CC, por su facilidad de control y porque existen diversos tipos como los servo motores, motores paso a paso etc; diseñados con características técnicas que se ajustan a las necesidades de la robótica en general.

5.1.2.1 Motor de CC. Un motor de corriente continua es una maquina eléctrica que convierte energía eléctrica en mecánica. Por lo regular es utilizado en las ruedas que permiten el movimiento de las plataformas de robótica móvil. A menudo suele estar adaptado a engranajes que reducen la velocidad de giro aumentando la potencia.

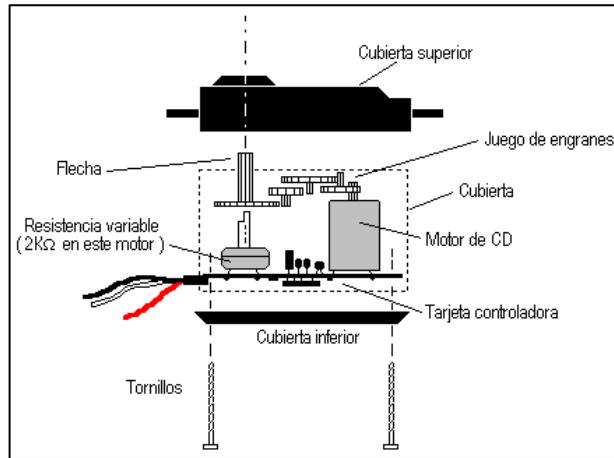
5.1.2.2 Servo motor. Del motor de CC se derivan otros actuadores como los servo motores y los motores paso a paso. Los servos traen una caja reductora que permite controlar de una mejor manera el giro del mismo.

En robótica móvil se utiliza un tipo especial de servo motor que es controlado de igual forma que el servo, pero tiene la capacidad de girar continuamente en ambos sentidos dependiendo de la señal de control. Este servo motor es un motor de corriente continua que incorpora un circuito electrónico que permite controlar de forma sencilla la dirección y la velocidad de giro de sus ejes mediante impulsos eléctricos (PWM)¹³.

¹² GARCÍA. Belmar. Diseño y Construcción de un Robot móvil controlado con técnicas de lógica difusa implementadas en un FPGA. [En línea]. Septiembre de 2012. [Consultado en enero de 2014]. Disponible en internet: <http://www.sepi.esimez.ipn.mx/electronica/archivos/992.pdf>

¹³ HIDALGO. Manuel; Martínez Antonio, Curso Provincial: Control y Robotica en Tecnologia. Robotica Educativa. EducaBot. Motores y Movilidad. [En línea]. Noviembre de 2009 [Consultado en febrero de 2014]. Disponible en internet: http://platea.pntic.mec.es/~mhidalgo/docEducaBot/04_EducaBot_ServomotoresMovilidad02.pdf

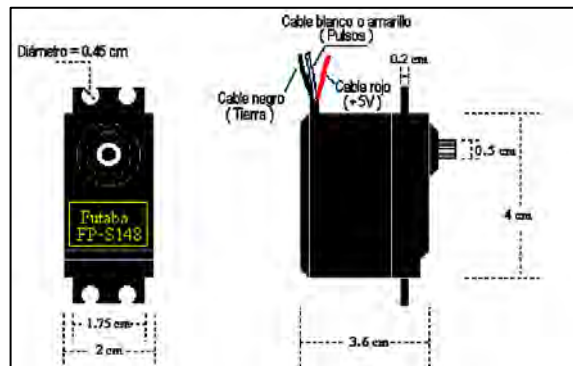
Figura 16. Servo Motor de Rotación Continua (Interior)



Fuente: tomado de: HIDALGO. Manuel; Martinez Antonio, Curso Provincial: Control y Robotica en Tecnologia. Robotica Educativa. EducaBot. Motores y Movilidad. [En línea]. Noviembre de 2009 [Consultado en febrero de 2014]. Disponible en internet: http://platea.pntic.mec.es/~mhidalgo/docEducaBot/04_EducaBot_ServomotoresMovilidad02.pdf

El servomotor de rotación continua tiene 3 cables. Dos de alimentación (rojo (+) y negro (-)), entre 5V y 7.5V, y uno de señal de control (amarillo o blanco).

Figura 17. Servo Motor de Rotación Continua (Exterior)

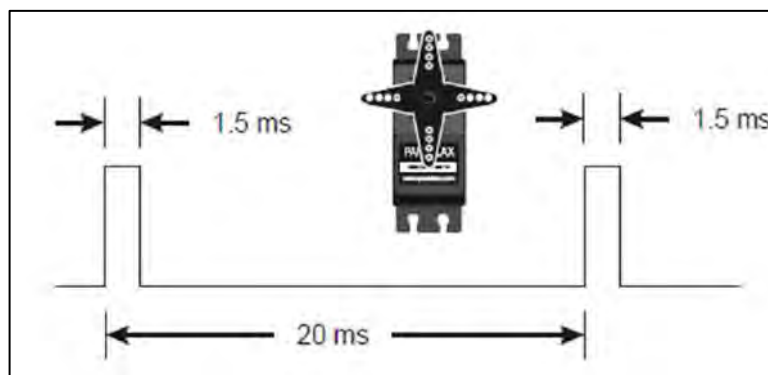


Fuente: tomado de: HIDALGO. Manuel; Martinez Antonio, Curso Provincial: Control y Robotica en Tecnologia. Robotica Educativa. EducaBot. Motores y Movilidad. [En línea]. Noviembre de 2009 [Consultado en febrero de 2014]. Disponible en internet: http://platea.pntic.mec.es/~mhidalgo/docEducaBot/04_EducaBot_ServomotoresMovilidad02.pdf

La señal de control de los servomotores de rotación continua es una señal de pulsos modulada en anchura PWM (Pulse Width Modulation). Este tipo de señal de control se utiliza en los servos estándar para realizar los giros desde 0° a 180°. En el caso de los servos continuos, va a permitir controlar el giro y la velocidad del eje del servomotor¹⁴.

Con el siguiente pulso de señal el servomotor de rotación continua tendría que estar parado. Para un servo estándar la posición del eje sería de 90°.

Figura 18. Servo Continuo en Posición Cero

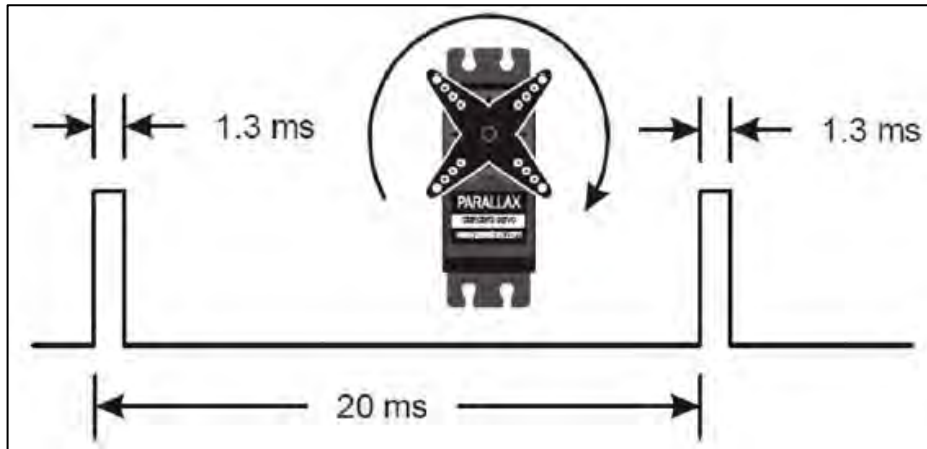


Fuente: tomado de: HIDALGO. Manuel; Martinez Antonio, Curso Provincial: Control y Robotica en Tecnologia. Robotica Educativa. EducaBot. Motores y Movilidad. [En línea]. Noviembre de 2009 [Consultado en febrero de 2014]. Disponible en internet: http://platea.pntic.mec.es/~mhidalgo/docEducaBot/04_EducaBot_ServomotoresMovilidad02.pdf

Para el giro a máxima velocidad en el sentido de las agujas del reloj el pulso de la señal sería:

¹⁴ HIDALGO. Manuel; Martinez Antonio, Curso Provincial: Control y Robotica en Tecnologia. Robotica Educativa. EducaBot. Motores y Movilidad. [En línea]. Noviembre de 2009 [Consultado en febrero de 2014]. Disponible en internet: http://platea.pntic.mec.es/~mhidalgo/docEducaBot/04_EducaBot_ServomotoresMovilidad02.pdf

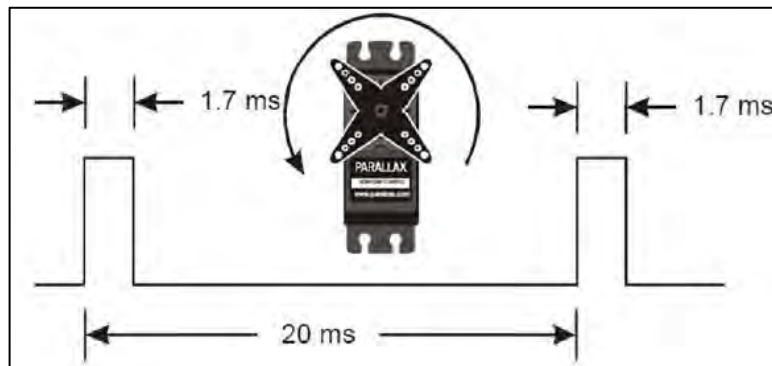
Figura 19. Servo Continuo Girando en Sentido Horario



Fuente: tomado de HIDALGO. Manuel; Martinez Antonio, Curso Provincial: Control y Robotica en Tecnologia. Robotica Educativa. EducaBot. Motores y Movilidad. [En línea]. Noviembre de 2009 [Consultado en febrero de 2014]. Disponible en internet: http://platea.pntic.mec.es/~mhidalgo/docEducaBot/04_EducaBot_ServomotoresMovilidad02.pdf

Para el giro a máxima velocidad en el sentido contrario a las agujas del reloj el pulso de la señal sería:

Figura 20. Servo Continuo Girando en Sentido Anti-Horario



Fuente: tomado de: HIDALGO. Manuel; Martinez Antonio, Curso Provincial: Control y Robotica en Tecnologia. Robotica Educativa. EducaBot. Motores y Movilidad. [En línea]. Noviembre de 2009 [Consultado en febrero de 2014]. Disponible en internet: http://platea.pntic.mec.es/~mhidalgo/docEducaBot/04_EducaBot_ServomotoresMovilidad02.pdf

Cabe resaltar que el periodo y la anchura del pulso de la señal de control varía dependiendo del fabricante del servo continuo.

5.1.2.3 Motor paso a paso. El motor paso a paso permite que el giro de su eje se comporte de manera discreta dependiendo de las señales de control aplicadas a cada una de sus bobinas, por lo cual el control de giro se logra de forma más precisa.

Un motor paso a paso es una maquina eléctrica en la que sus devanados se energizan uno después del otro, estas excitaciones provocan un giro discontinuo en un angulo que se determina por la posición que toma el eje. Este es capaz de transformar información digital, en movimientos mecánicos, el eje del motor gira un determinado angulo por cada impulso de entrada, el resultado final del movimiento es fijo y repetible, produce un posicionamiento preciso y fiable, el sentido de rotación del motor se define con el sentido de excitación de las bobinas que al ser excitadas con impulsos, actúan sobre un nucleo de hierro dulce o iman permanente y lo hacen girar un angulo determinado, la velocidad de rotación del eje del motor (N) en revoluciones por minuto es:

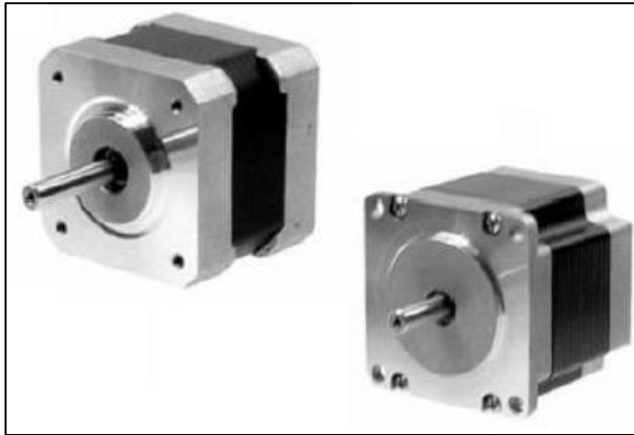
$$N = \frac{60f}{n}$$

Ecuación 2. Velocidad de rotación en un motor paso a paso

Donde f =frecuencia de los impulsos (Hz) y n =numero de espiras

El desplazamiento angular al pasar de una bobina a otra es $2/n$, lo que representa una conversión de señales digitales de excitación a una posición angular discontinua definida sobre el eje del motor. Al desplazamiento angular se le denomina paso angular y es precisamente lo que caracteriza a este tipo de motores, ya que funcionan en pasos con un ángulo determinado.

Figura 21. Motor Paso a Paso



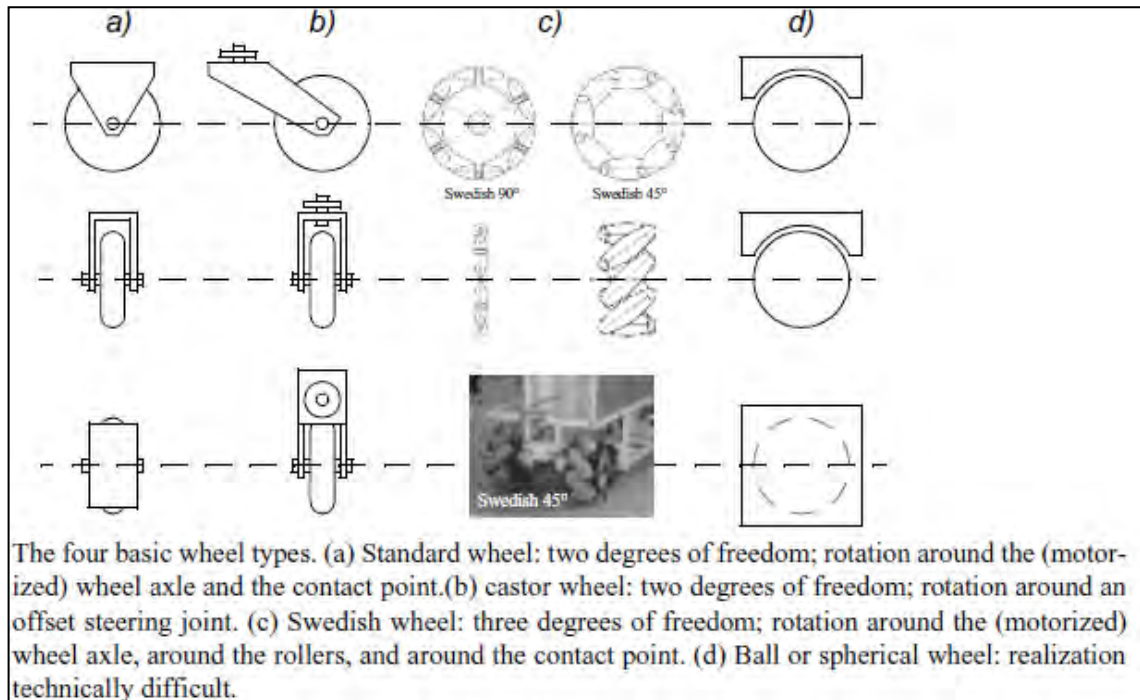
Fuente: tomada de: GARCÍA. Belmar. Diseño y Construcción de un Robot móvil controlado con técnicas de lógica difusa implementadas en un FPGA. [En línea]. Septiembre de 2012. [Consultado en enero de 2014]. Disponible en internet: <http://www.sepi.esimez.ipn.mx/electronica/archivos/992.pdf>

5.1.3 Locomoción mediante ruedas. En robótica móvil la locomoción es un aspecto fundamental, pues es lo que permite que un robot pueda moverse en un entorno determinado. Existen diversas formas en las que un robot puede adquirir movimiento; una de las más utilizadas y en la que se hará énfasis es la locomoción mediante ruedas.

El sistema de ruedas es el más común en las plataformas de robótica móvil, ya que el control se hace mucho más sencillo a comparación de otros sistemas de locomoción como patas o articulaciones. Las ruedas ofrecen estabilidad y se pueden obtener movimientos suaves y bien calculados. Además la mayoría de aplicaciones desarrolladas a lo largo de la historia operaban en ambientes controlados y superficies regulares lo que hacía que las ruedas fueran la mejor solución. Cabe resaltar que en aplicaciones en donde la superficie sea irregular debe contar con un diseño muy sofisticado de ruedas que garantice el movimiento fluido de la plataforma, pues de lo contrario, este tipo de locomoción tendrá desventaja frente a otros como las articulaciones.

En la figura 22 se presenta un resumen de las configuraciones de ruedas más comunes en plataformas de robótica móvil.

Figura 22. Diferentes Tipos de Ruedas



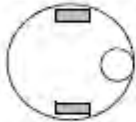
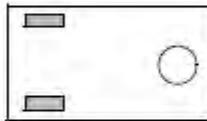
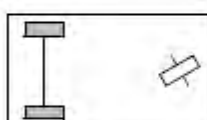
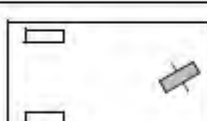

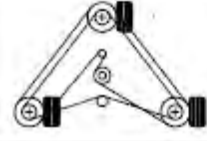


Fuente: tomada de: SIEGWART Roland, Nourbakhsh Illah R. Introduction to Autonomous Mobile Robots. 1 ed. Cambridge, London: The MIT Press, 2004.

De igual forma en la que existen diferentes tipos de ruedas, existen diversas formas de configuración dentro de una plataforma de robótica móvil. Estas configuraciones hacen que las plataformas tomen un comportamiento distinto unas de otras en cuanto a aspectos como la estabilidad y la maniobrabilidad.

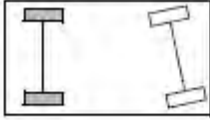
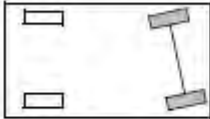
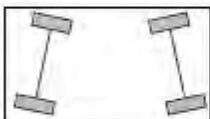
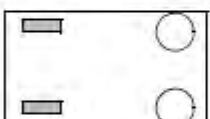
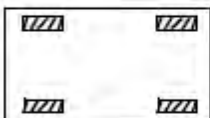

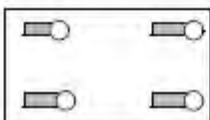
A continuación se presentan una serie de tablas en donde se resumen las diferentes configuraciones y sus posibles aplicaciones.

Cuadro 2. Configuraciones de dos y tres ruedas para robots móviles.

| # of wheels | Arrangement | Description | Typical examples |
|-------------|---|--|--|
| 2 |  | One steering wheel in the front, one traction wheel in the rear | Bicycle, motorcycle |
| |  | Two-wheel differential drive with the center of mass (COM) below the axle | Cye personal robot |
| 3 |  | Two-wheel centered differential drive with a third point of contact | Nomad Scout, smartRob EPFL |
| |  | Two independently driven wheels in the rear/front, 1 unpowered omnidirectional wheel in the front/rear | Many indoor robots, including the EPFL robots Pygmalion and Alice |
| |  | Two connected traction wheels (differential) in rear, 1 steered free wheel in front | Piaggio minitrucks |
| |  | Two free wheels in rear, 1 steered traction wheel in front | Neptune (Carnegie Mellon University), Hero-1 |
| |  | Three motorized Swedish or spherical wheels arranged in a triangle; omnidirectional movement is possible | Stanford wheel Tribolo EPFL, Palm Pilot Robot Kit (CMU) |
| |  | Three synchronously motorized and steered wheels; the orientation is not controllable | "Synchro drive" Denning MRV-2, Georgia Institute of Technology, I-Robot B24, Nomad 200 |

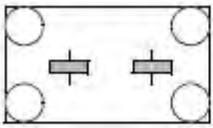
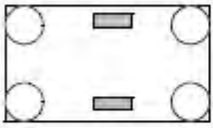







Fuente: tomada de: SIEGWART Roland, Nourbakhsh Illah R. Introduction to Autonomous Mobile Robots. 1 ed. Cambridge, London: The MIT Press, 2004.

Cuadro 3. Configuraciones de cuatro ruedas para robots móviles.

| # of wheels | Arrangement | Description | Typical examples |
|-------------|---|---|--|
| 4 |  | Two motorized wheels in the rear, 2 steered wheels in the front; steering has to be different for the 2 wheels to avoid slipping/skidding. | Car with rear-wheel drive |
| |  | Two motorized and steered wheels in the front, 2 free wheels in the rear; steering has to be different for the 2 wheels to avoid slipping/skidding. | Car with front-wheel drive |
| |  | Four steered and motorized wheels | Four-wheel drive, four-wheel steering Hyperion (CMU) |
| |  | Two traction wheels (differential) in rear/front, 2 omnidirectional wheels in the front/rear | Charlie (DMT-EPFL) |
| |  | Four omnidirectional wheels | Carnegie Mellon Uranus |
| |  | Two-wheel differential drive with 2 additional points of contact | EPFL Khepera, Hyperbot Chip |
| |  | Four motorized and steered castor wheels | Nomad XR4000 |

Fuente: tomada de: SIEGWART Roland, Nourbakhsh Illah R. Introduction to Autonomous Mobile Robots. 1 ed. Cambridge, London: The MIT Press, 2004.

Cuadro 4. Configuraciones de seis ruedas para robots móviles.

| # of wheels | Arrangement | Description | Typical examples |
|---|---|--|---|
| 6 |  | Two motorized and steered wheels aligned in center, 1 omnidirectional wheel at each corner | First |
| |  | Two traction wheels (differential) in center, 1 omnidirectional wheel at each corner | Terregator (Carnegie Mellon University) |
| Icons for the each wheel type are as follows: | | | |
|  | unpowered omnidirectional wheel (spherical, castor, Swedish); | | |
|  | motorized Swedish wheel (Stanford wheel); | | |
|  | unpowered standard wheel; | | |
|  | motorized standard wheel; | | |
|  | motorized and steered castor wheel; | | |
|  | steered standard wheel; | | |
|  | connected wheels. | | |

Fuente: tomada de: SIEGWART Roland, Nourbakhsh Illah R. Introduction to Autonomous Mobile Robots. 1 ed. Cambridge, London: The MIT Press, 2004.

5.1.4 Cinemática del robot móvil. La cinemática es el estudio del movimiento de un cuerpo o en este caso, de un sistema mecánico, sin importar las causas que lo originan. En robótica móvil, es indispensable conocer el comportamiento mecánico del robot, esto con el fin de poder diseñar un robot apropiado, mejorar uno ya existente o simplemente poder diseñar e implementar un software de control que permita desarrollar tareas lo más acertadas posibles.

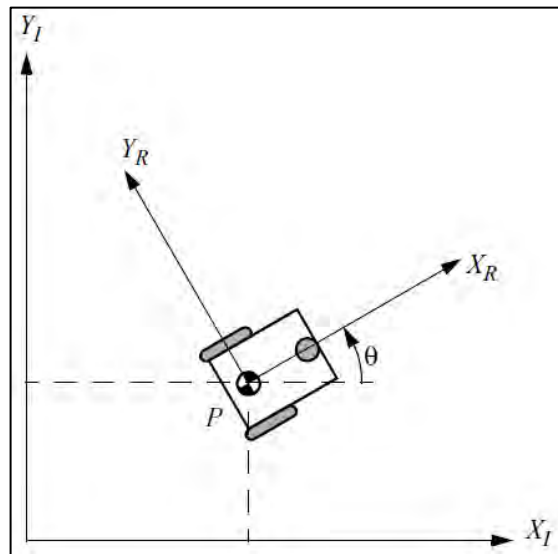
En robótica móvil la cinemática está fuertemente ligada a la estimación del movimiento del robot y con la contribución que aporta cada una de las ruedas al movimiento de todo el robot.

Así como las ruedas son las causantes del movimiento del robot, también imponen restricciones a ese movimiento. Estas restricciones dependen del tipo y configuración de rueda que utilice el robot.

Un robot móvil en configuración diferencial y con ruedas fijas es el caso de estudio más sencillo que existe para obtener un modelo cinemático en robótica móvil.

El proceso para obtener un modelo cinemático comienza por definir 2 marcos de referencia, uno local al robot y otro global. Teniendo definido lo anterior se continúa con el estudio cinemático de cada una de las ruedas dentro del chasis del robot, que tipo de ruedas son, como se encuentran geoméricamente acopladas y cuantas ruedas posee el chasis. Con todo lo anterior hecho y habiendo obtenido tanto la cinemática como las restricciones que impone cada una las ruedas al movimiento del robot, se finaliza con la obtención de un modelo en el cual el robot es visto como un todo. [Sieg04]

Figura 23. Posición de un Robot Diferencial



Fuente: tomada de: SIEGWART Roland, Nourbakhsh Illah R. Introduction to Autonomous Mobile Robots. 1 ed. Cambridge, London: The MIT Press, 2004.

Todos los robots móviles son modelados a partir de la suposición de que son cuerpos rígidos y en este caso un cuerpo rígido sobre ruedas, moviéndose u operando en un plano horizontal.

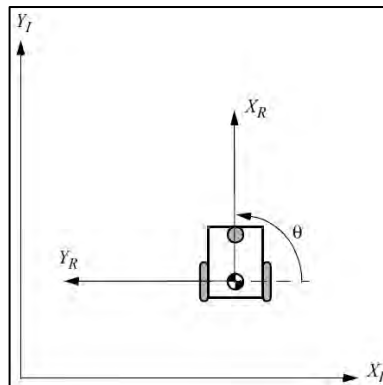
Para especificar la posición del robot en el plano horizontal, se elige un punto P en el chasis del robot como punto de referencia de la posición. La base $\{X_R\}$ define dos ejes relativos a P en el chasis del robot y es por lo tanto el marco de referencia local del robot. La posición de P en el marco de referencia global está especificado por las coordenadas x e y, y la diferencia angular entre los marcos de referencia global y local está dada por θ . Se puede describir entonces la posición del robot como un vector con estos tres elementos. Se debe tener en cuenta el uso del subíndice I para aclarar que esta postura se ha establecido con respecto al marco de referencia global¹⁵.

$$\xi_I = \begin{bmatrix} X \\ Y \\ \theta \end{bmatrix}$$

Ecuación 3. Representación de la Posición del Robot

Para describir el movimiento del robot en términos de sus componentes de movimiento, será necesario describir el movimiento a lo largo de los ejes del marco de referencia global para cada movimiento a lo largo de los ejes del marco de referencia local del robot. Cabe resaltar que el movimiento en el marco de referencia global es función de la posición actual del robot. Esta descripción se lleva a cabo utilizando la matriz de rotación ortogonal:

Figura 24. Posición del Robot perpendicular al marco de referencia global



Fuente: tomada de: SIEGWART Roland, Nourbakhsh Illah R. Introduction to Autonomous Mobile Robots. 1 ed. Cambridge, London: The MIT Press, 2004.

¹⁵ SIEGWART Roland, Nourbakhsh Illah R. Introduction to Autonomous Mobile Robots. 1 ed. Cambridge, London: The MIT Press, 2004.

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

Ecuación 4. Matriz de Rotación

Esta matriz se puede utilizar para describir el movimiento en el marco de referencia global en términos de movimiento en el marco de referencia local. Esta operación se denota mediante (θ) debido a que el cálculo de esta operación depende del valor de θ .

Por ejemplo, considerando el robot en la figura 24. Para este robot, debido a que $\theta = \pi/2$ se puede calcular fácilmente la matriz de rotación instantánea:

Por otra parte, dada alguna velocidad (X, Y, θ) en el marco de referencia global, se pueden calcular las componentes de movimiento a lo largo de los ejes del marco de referencia local mediante la ecuación:

$$\dot{\xi}_R = R\left(\frac{\pi}{2}\right) \dot{\xi}_I = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{Y} \\ -\dot{X} \\ \dot{\theta} \end{bmatrix}$$

Ecuación 5. Posición a lo largo del Marco de Referencia Local del Robot

La ecuación anterior es un modelo simple de la cinemática directa de un robot móvil, pero se han despreciado factores que son determinantes para el movimiento de un robot. Considerando un robot móvil diferencial como el de la figura 2.13, se tienen dos ruedas con sus respectivos diámetros r , se establece un punto P en el medio de las ruedas cuya distancia a cada una de ellas es l . con estos datos, además de la orientación θ y las velocidades de rotación de cada una de las ruedas ϕ_1 y ϕ_2 el modelo cinemático directo para predecir la velocidad en el marco de referencia global es una función que depende de estos parámetros:

$$\dot{\xi}_I = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{bmatrix} = f(l, r, \theta, \phi_1, \phi_2)$$

Ecuación 6. Posición a lo largo del Marco de Referencia Global del Robot

De la ecuación 3 se puede deducir que:

$$\dot{\xi}_l = R(\theta)^{-1} \dot{\xi}_R$$

Lo cual indica que es posible calcular el movimiento en el marco de referencia global a partir del movimiento en el marco de referencia local. [Sieg04]

Siguiendo con el ejemplo del robot diferencial, las contribuciones de las dos ruedas pueden ser simplemente sumadas para calcular la componente X_R en el marco de referencia ξ_R . Por lo cual la velocidad de rotación es $X_{r1} = r\dot{\phi}_1$ y $X_{r2} = r\dot{\phi}_2$ respectivamente. En cuanto a la componente Y_R es siempre cero ya que por la composición de este tipo de robot este nunca tendrá desplazamientos a lo largo de este eje. Esto se verá en mejor forma más adelante cuando se explique las restricciones en cada una de las ruedas y se obtenga un modelo cinemático general que aplica para cualquier robot móvil cuya locomoción se base en ruedas¹⁶.

Continuando con el ejemplo falta por obtener la componente rotacional θ_r del marco de referencia local ξ_R , la cual se obtiene nuevamente calculando la contribución de cada una de las ruedas y luego sumándolas. Considerando la rueda derecha como la rueda 1 la rotación hacia a delante de esta rueda resulta en contra de las manecillas del reloj en el punto P. si la rueda 1 rota sola, mientras la rueda 2 permanece quieta, la velocidad de rotación ω_1 puede ser calculada porque la rueda se estará moviendo instantáneamente a lo largo de un arco de círculo de radio $2l$ ¹⁷.

$$\omega_1 = \frac{r\dot{\phi}_1}{2l}$$

Y para la rueda 2

$$\omega_2 = \frac{-r\dot{\phi}_2}{2l}$$

¹⁶ SIEGWART Roland, Nourbakhsh Illah R. Introduction to Autonomous Mobile Robots. 1 ed. Cambridge, London: The MIT Press, 2004.

¹⁷ Ibid.

Como se puede observar en la anterior ecuación, el mismo cálculo aplica para la rueda 2, con la diferencia de que la rotación hacia adelante se da en el sentido de las manecillas del reloj¹⁸.

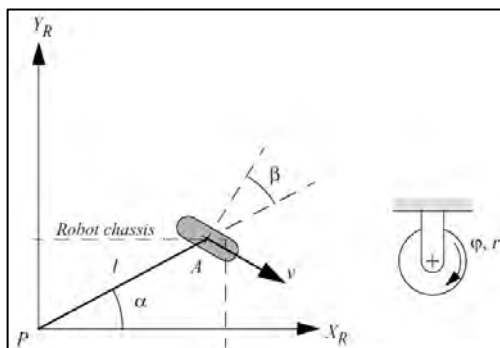
Combinando las dos fórmulas anteriores se obtiene un modelo cinemático para un determinado robot diferencial:

$$\dot{\xi}_I = R(\theta)^{-1} \begin{bmatrix} \frac{r\dot{\phi}_1}{2} + \frac{r\dot{\phi}_2}{2} \\ 0 \\ \frac{r\dot{\phi}_1}{2l} + \frac{-r\dot{\phi}_2}{2l} \end{bmatrix}$$

Todo lo anterior describe el modelo cinemático de un robot diferencial, pero es de gran importancia hallar un modelo general que sea aplicable a cualquier de configuración de robot móvil con ruedas. Para esto se empieza por describir las restricciones cinemáticas que cada tipo de rueda impone al movimiento total del robot.

En este caso se hará énfasis en la rueda estándar fija, ya que es la utilizada en la plataforma móvil para realizar las pruebas del sistema que se describe en este trabajo¹⁹.

Figura 25. Rueda Fija Estándar



Fuente: tomada de: SIEGWART Roland, Nourbakhsh Illah R. Introduction to Autonomous Mobile Robots. 1 ed. Cambridge, London: The MIT Press, 2004.

¹⁸ SIEGWART Roland, Nourbakhsh Illah R. Introduction to Autonomous Mobile Robots. 1 ed. Cambridge, London: The MIT Press, 2004.

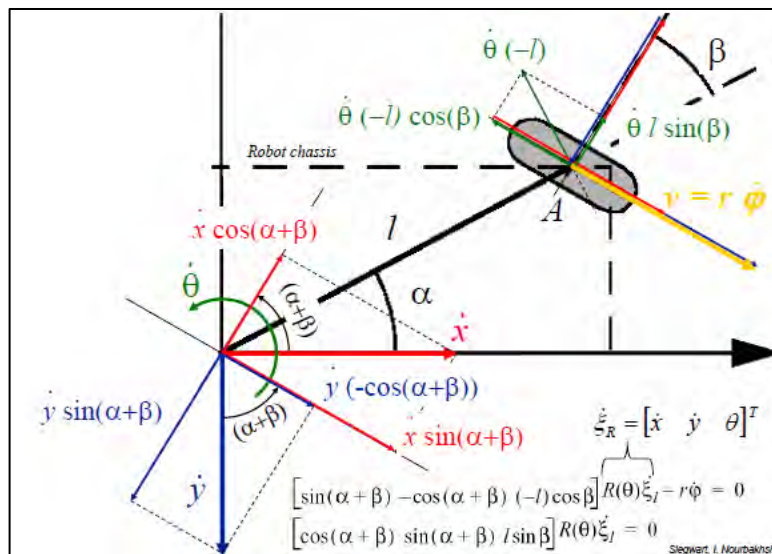
¹⁹ Ibid.

La rueda fija estándar no tiene un eje de dirección, su ángulo con respecto al chasis del robot es siempre fijo y su movimiento se limita sobre el plano de la rueda hacia adelante y atrás. En cuanto a su rotación está limitada alrededor del punto de contacto de la rueda con la superficie.

La figura 25 describe una rueda fija estándar A e indica su posición relativa con respecto al marco de referencia local al robot $\{XR, \}$. La posición de A esta expresada en coordenadas polares por la distancia l y el ángulo α . El ángulo del plano de la rueda con respecto al chasis esta denotado por β , el cual es siempre fijo debido a que se trata de una rueda fija la cual no tiene eje de dirección. La rueda posee su radio r y puede rotar a través del tiempo por lo que la posición de rotación con respecto al eje horizontal es un función del tiempo (t) .

Este tipo de rueda impone dos restricciones, una sobre el mismo plano de la rueda y la otra en el plano perpendicular a la misma. La primera restricción impone que todo movimiento a lo largo del plano de la rueda va acompañado de la cantidad apropiada de giro de la rueda de manera que exista rodadura pura en el punto de contacto. Es decir que exista un único punto de contacto y que en este punto la rueda no patine, garantizando un movimiento puro (rodadura pura).

Figura 26. Restricciones de una rueda fija Estándar



$$[\sin(\alpha + \beta) \quad -\cos(\alpha + \beta) \quad (-l)\cos\beta]R(\theta)\dot{\xi}_I - r\dot{\phi} = 0$$

Ecuación 7. Movimiento en el Plano de la Rueda

El primer término de la anterior ecuación representa el movimiento total a través del plano de la rueda. Los tres términos de este vector describen la contribución de cada una de las componentes X, θ al movimiento total del robot. El término (θ) se usa de igual forma que en las ecuaciones anteriores para pasar del marco de referencia global al local, esto debido a que todos los demás parámetros α, β, l están en términos del marco de referencia local. De acuerdo al enunciado de esta restricción todo el movimiento de la rueda sobre su plano debe ser igual al movimiento que se logra cuando la rueda gira, $r\dot{\varphi}$ ²⁰.

La segunda restricción que impone esta rueda establece que no existe movimiento a lo largo del plano ortogonal de la rueda por lo que:

$$[\cos(\alpha + \beta) \quad \sin(\alpha + \beta) \quad l \sin \beta] R(\theta) \dot{\xi}_I = 0$$

Ecuación 8. Movimiento en el Plano Ortogonal a la Rueda

Dado un robot con M numero ruedas con todo lo anterior ahora es posible establecer las restricciones cinemáticas del chasis. Como se observó anteriormente, cada rueda contribuye al movimiento total del chasis, por lo que combinando estas ecuaciones se puede llegar a un modelo cinemático general del robot²¹.

Existen tipos de ruedas como las castor y las omnidireccionales, las cuales no imponen ningún tipo de restricción al movimiento propio y por ende al del chasis del robot, pero por otra parte si existen otro tipo de ruedas como las fijas y las orientables, las cuales si imponen restricciones al movimiento. Por tal razón a continuación se mostrara como obtener un modelo general de un robot que contenga este tipo de ruedas²².

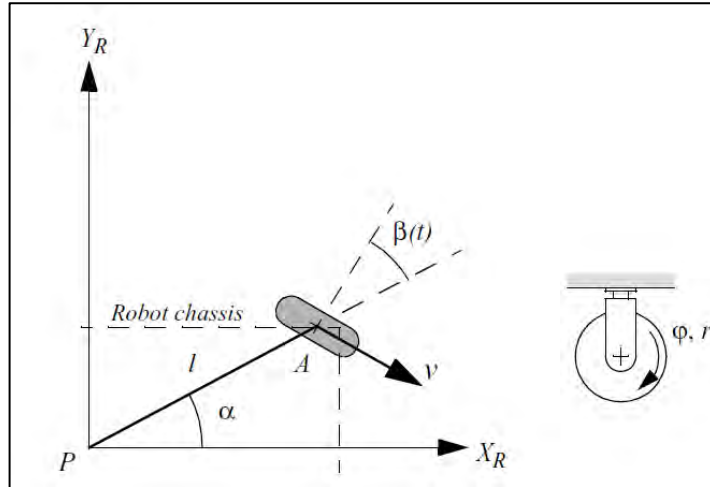
Antes de obtener el modelo cabe resaltar que así como la rueda fija impone restricciones al movimiento total del robot, la rueda orientable también. La restricción para este caso en el plano de la rueda es:

²⁰ SIEGWART Roland, Nourbakhsh Illah R. Introduction to Autonomous Mobile Robots. 1 ed. Cambridge, London: The MIT Press, 2004.

²¹ Ibid.

²² Ibid.

Figura 27. Rueda Orientable Estándar



Fuente: tomada de: SIEGWART Roland, Nourbakhsh Illah R. Introduction to Autonomous Mobile Robots. 1 ed. Cambridge, London: The MIT Press, 2004.

$$[\sin(\alpha + \beta) \quad -\cos(\alpha + \beta) \quad (-l) \cos \beta] R(\theta) \dot{\xi}_I - r \dot{\varphi} = 0$$

Ecuación 9. Movimiento en el Plano de la Rueda

En el plano ortogonal a la misma es:

$$[\cos(\alpha + \beta) \quad \sin(\alpha + \beta) \quad l \sin \beta] R(\theta) \dot{\xi}_I = 0$$

Ecuación 10. Movimiento en el Plano Ortogonal a la Rueda

Estas limitaciones son idénticas a las de la rueda estándar fija porque φ y β no tiene un impacto directo sobre las restricciones de movimiento instantáneo de un robot. Es sólo mediante la integración en el tiempo que los cambios en el ángulo de dirección pueden afectar a la movilidad de un vehículo. Esto puede parecer sutil, pero es muy importante la diferencia entre el cambio en la posición de orientación β , y el cambio en el giro de las ruedas φ ²³.

²³ SIEGWART Roland, Nourbakhsh Illah R. Introduction to Autonomous Mobile Robots. 1 ed. Cambridge, London: The MIT Press, 2004.

Dado un robot con N número de ruedas estándar (fijas y orientables) de las cuales N_f son ruedas fijas y N_s son ruedas orientables. El término (t) denota el ángulo con respecto al chasis del robot de las ruedas orientables, mientras que el término β_s hace referencia a este mismo ángulo, pero en el caso de las ruedas fijas. Como se expuso anteriormente ambas ruedas acopladas al chasis del robot rotaran alrededor de un eje horizontal, dicha rotación varia con respecto al tiempo por lo que se usan los términos $\varphi_s(t)$ y $\varphi_f(t)$ para representar esta rotación para las ruedas orientables y fijas respectivamente. Estos dos términos se combinan en una sola matriz (t) :

$$\varphi(t) = \begin{bmatrix} \varphi_s(t) \\ \varphi_f(t) \end{bmatrix}$$

De igual forma, las restricciones de rodamiento de las ruedas se pueden reescribir en una sola expresión:

$$J_1(\beta_s)R(\theta)\dot{\xi}_I - J_2\dot{\varphi} = 0$$

Donde J_2 es una matriz diagonal $N \times N$ cuyas entradas son los radios de cada una de las ruedas. $J_1(\beta_s)$ es una matriz que denota las proyecciones para todas las ruedas de sus movimientos para cada uno de sus planos individuales (ecuación 5 para cada una de las ruedas).

$$J_1(\beta_s) = \begin{bmatrix} J_{1f} \\ J_{1s}(\beta_s) \end{bmatrix}$$

J_{1f} Hace referencia a las ruedas mientras que $J_{1s}(\beta_s)$ hace referencia al tipo de ruedas orientable en la cual el ángulo β varia.

La anterior ecuación hace referencia a la primera restricción, la cual se impone sobre el mismo plano de la rueda. Esta ecuación es la misma que la ecuación 5, solo que en lugar de simples valores se escriben matrices.

Tanto $J_{1s}(\beta_s)$ como J_{1f} son matrices de $N_s \times 3$ y $N_f \times 3$ donde cada fila consiste de los tres términos en las tres matrices de la ecuación 5 para cada rueda estándar fijo. Así como la rueda fija tiene dos restricciones, la rueda orientable

también por lo que la matriz $J_S(\beta_s)$ en sus filas posee los tres términos en las tres matrices de la ecuación 7 para cada rueda estándar orientable²⁴.

De igual forma que se puede reescribir la restricción sobre el plano ortogonal de las ruedas de la siguiente manera:

$$C_1(\beta_s)R(\theta)\dot{\xi}_I = 0$$

$$C_1(\beta_s) = \begin{bmatrix} C_{1f} \\ C_{1s}(\beta_s) \end{bmatrix}$$

Donde C_{1f} y C_{1s} son matrices de $N_f \times 3$ y $N_s \times 3$ cuyas filas son los tres términos en las tres matrices de las ecuaciones 3.13 y 3.16 para todas las ruedas estándar fijas y orientables.

De esta manera se puede resumir el modelo general de un robot móvil que posea ruedas fijas y orientables:

$$\begin{bmatrix} J_1(\beta_s) \\ C_1(\beta_s) \end{bmatrix} R(\theta)\dot{\xi}_I = \begin{bmatrix} J_2\varphi \\ 0 \end{bmatrix}$$

Ecuación 11. Modelo cinemático general

En la anterior ecuación es una notación en forma matricial de las dos ecuaciones anteriores²⁵.

5.2 UNIDAD DE PROCESAMIENTO

Hasta ahora se ha presentado de forma general la anatomía a nivel estructural de los que es una plataforma de robótica móvil.

²⁴ SIEGWART Roland, Nourbakhsh Illah R. Introduction to Autonomous Mobile Robots. 1 ed. Cambridge, London: The MIT Press, 2004.

²⁵ Ibid.

Un sistema embebido es una combinación de hardware y software, además de algunas piezas mecánicas, diseñado para realizar una tarea específica dentro de un sistema mucho más grande.

Los sistemas embebidos no son de propósito general, por lo que una vez son diseñados no podrán ser reprogramados para que realicen una función distinta.

Actualmente los sistemas embebidos se encuentran en mucho de los dispositivos que nos rodean. Desde lavadoras, automóviles, máquinas industriales, cámaras digitales fotográficas, hasta plataformas de robótica móvil como la tratada en este trabajo.

Un sistema embebido no está definido por una tecnología en particular, ya que por lo regular están conformados por sensores, actuadores, procesadores y elementos de almacenamiento integrados, los cuales pueden ser de diferente tecnología, arquitectura y/o fabricante²⁶.

Todo sistema embebido posee un subsistema digital como elemento principal, el cual está conformado por procesadores programados para que ejecuten funciones sobre hardware y software. Por tal motivo el diseño de sistemas embebidos se encuentra ligado a una metodología de codiseño hardware/software también conocida con el nombre de particionado hardware/software. Esta metodología ayuda a que los diseñadores definan que funciones se deben implementar en hardware y cuales en software.

Actualmente se pueden encontrar sistemas embebidos basados en microcontroladores pic, atmel y socs en general.

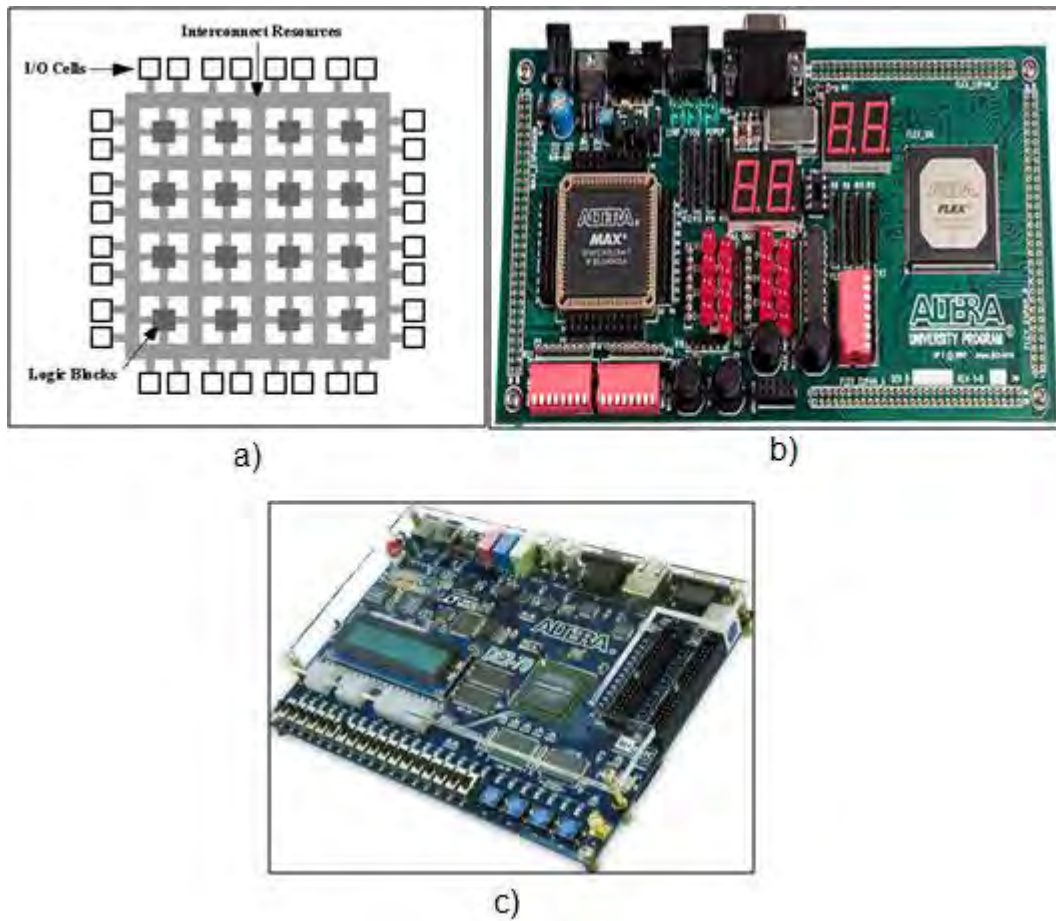
También se pueden encontrar sistemas embebidos basados en tarjetas como FPGAs y pequeños computadores como por ejemplo el Raspberry Pi o el Space Cube.

5.2.1 Hardware reconfigurable: FPGA'S. Las FPGAs son dispositivos semiconductores formados por bloques de celdas lógicas cuya interconexión logran efectuar operaciones lógicas sencillas como las funciones básicas AND, OR; hasta complejos algoritmos que controlan procesos reales como el nivel de un

²⁶ GALEANO, Gustavo. Programación de Sistemas Embebidos en C. Bogotá: Alfa omega, 2009.

tanque. Es común encontrar que las FPGAs vienen en conjunto con tarjetas de desarrollo las cuales poseen puertos de expansión de propósito general interfaces como VGA, usb, Ethernet etc. Un ejemplo de lo anterior son las tarjetas de desarrollo de altera como la UP2 o la DE-270.

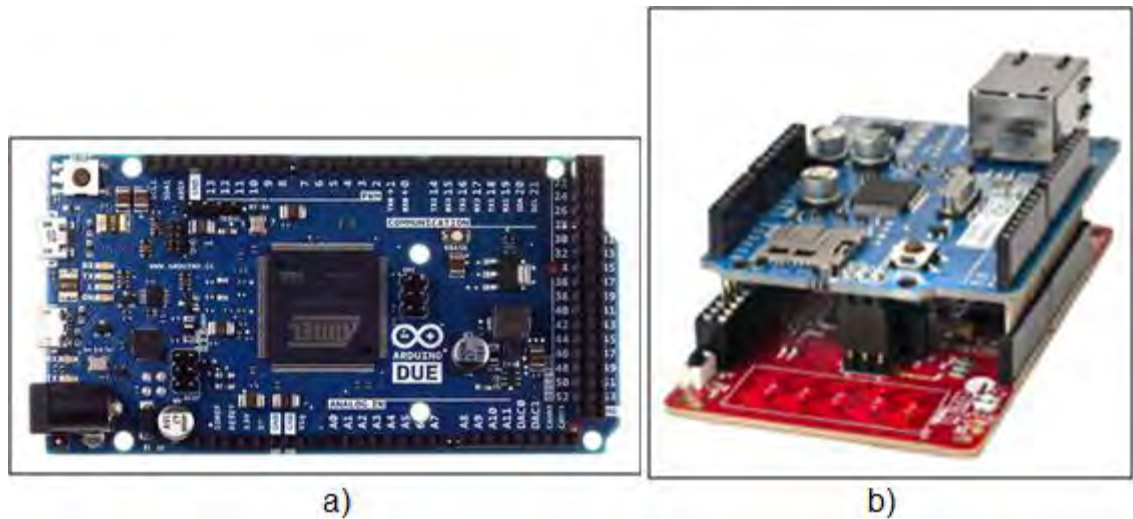
Figura 28. a) Estructura de una FPGA. b) Kit UP 2. c) Kit DE-270.



Fuente: tomada de: ALTERA.Kit development university de-270. [En línea]. [Consultado en agosto de 2013]. Disponible en internet: <http://www.altera.com/education/univ/materials/boards/de2-70/unv-de2-70-board.html>

5.2.2 Microprocesadores sin sistema operativo. Un ejemplo de este tipo de sistemas son los sistemas arduino con microcontroladores atmel y los psocs desarrollados por la empresa cypress.

Figura 29. a) Arduino DUE. b) Psoc 4 con Arduino shield



Fuente: a) tomada de: ARDUINO. Arduino DUE. [En línea]. [Consultado en agosto de 2013]. Disponible en internet: <http://www.arduino.cc/> b) tomada de: PSOC. Psoc 4. [En línea]. [Consultado en agosto de 2013]. Disponible en internet: <http://www.cypress.com/psoc/>

5.2.2.1 Arduino. Es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinarios.

El hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida.⁴ Los microcontroladores más usados son el Atmega168, Atmega328, Atmega1280, ATmega8 por su sencillez y bajo coste que permiten el desarrollo de múltiples diseños. Por otro lado el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque que es ejecutado en la placa.

Arduino puede tomar información del entorno a través de sus entradas analógicas y digitales, puede controlar luces, motores y otros actuadores. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un computador²⁷.

²⁷ ARDUINO. Arduino DUE. [En línea]. [Consultado en agosto de 2013]. Disponible en internet: <http://www.arduino.cc/>

5.2.2.2 Psoc. Es la abreviación de Program System On Chip el cual es un microprocesador muy versátil, el que se asemeja a un lego, totalmente dinámico ya que podemos disponer de sus componentes a nuestra voluntad, cuanta con innumerables dispositivos electrónicos, los cuales se pueden modificar pra crear de forma interna, filtros análogos y digitales, amplificadores, comparadores, conversores analógicos digitales de varios tipos y resolución, moduladores de ancho de pulso (PWM) de 8, 16, 32 Bits, contadores de 8, 16, 32 Bits entre muchos otros.

El PSOC consta de 2 tipos de bloques para desarrollo, análogos y digitales; la cantidad puede variar de acuerdo a la familia del microprocesador seleccionado, la más común es la CY8C27443 fabricado por cypress, la cual consta de 9 bloques análogos y 9 digitales, además posee una unidad multiplicadora MAC de 8X8 pudiendbo almacenar resultados de 32 bits.

Este puede funcionar con un clock interno que multiple configuración pudiendo operar con 48Mhz, 24Mhz, 12Mhz, 6Mhz, 3Mhz o hasta 32 Mhz. Si el usuario lo desea puede operar con un cristal externo.

También cuanta con una unidad de referencia de voltaje múltiple la cual permite variar el voltaje de referencia para trabajar con sensores y otros dispositivos. Es alimentado con 0.5V - 3.0V hasta 1.0V²⁸.

5.2.3 Microprocesadores con Sistema Operativo. ARM. Es una arquitectura RISC (Reduced Instruction Set Computer=Ordenador con Conjunto Reducido de Instrucciones) de 32 bits desarrollada por ARM Holdings. Se llamó Advanced RISC Machine, y anteriormente Acorn RISC Machine. La arquitectura ARM es el conjunto de instrucciones de 32 bits más ampliamente utilizado en unidades producidas. Concebida originalmente por Acorn Computers para su uso en ordenadores personales, los primeros productos basados en ARM eran los Acorn Archimedes, lanzados en 1987.

La relativa simplicidad de los procesadores ARM los hace ideales para aplicaciones de baja potencia. Como resultado, se han convertido en dominante en el mercado de la electrónica móvil e integrada, encarnados en microprocesadores y microcontroladores pequeños, de bajo consumo y relativamente bajo coste. En 2005, alrededor del 98% de los más de mil millones de teléfonos móviles vendidos utilizaban al menos un procesador ARM. Desde

²⁸ UNICROM. Microcontroladores Psoc. [En línea]. [Consultado en agosto de 2013]. Disponible en Internet: http://www.unicrom.com/Cmp_microcontroladores_PSOC.asp

2009, los procesadores ARM son aproximadamente el 90% de todos los procesadores RISC de 32 bits integrados, cabe hacer mención que no existe una tabla de equivalencias del performance entre tecnologías de procesadores y se utilizan ampliamente en la electrónica de consumo, incluyendo PDA, tabletas, Teléfono inteligente, teléfonos móviles, videoconsolas portátiles, calculadoras, reproductores digitales de música y medios (fotos, vídeos, etc.), y periféricos de ordenador como discos duros y routers²⁹.

5.2.3.1 Raspberry pi. Desarrollado por la organización Raspberry Pi del Reino Unido es un pequeño ordenador que contiene compones basicos que cualquier ordenador debe tener. Cuenta con un procesador de referencia ARM1176JZF-S a 700 MHz con posibilidad de aumentar su velocidad a 1GHz mediante overlocking. Memoria ram de 512 MB, y un slot para una tarjeta SD card, la cual hace las veces de disco duro para almacenamiento permanente.

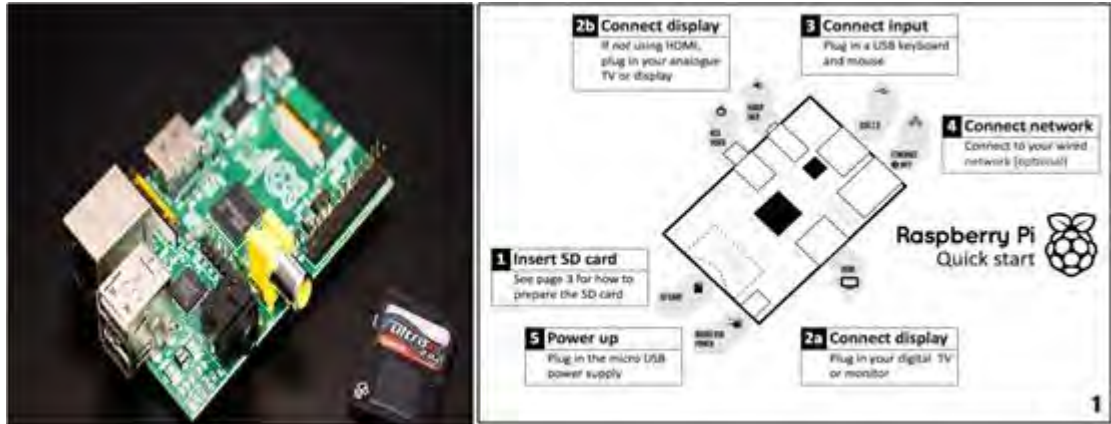
Ademas de esto cuenta con las interfaces de entrada y salida para monitor, conexión de red y alimentacion la cual es externa a este ordenador.

Actualmente se consiguen en el mercado dos modelos A y B, los cuales vienen con un sistema operativo llamado Raspbian, el cual es derivado de la distrubucion debian de linux³⁰.

²⁹ WIKIPEDIA. Arquitectura ARM. [en línea].[consultado en septiembre de 2013]. Disponible en internet: http://es.wikipedia.org/wiki/Arquitectura_ARM

³⁰ RASPBERRY. Raspberry Pi [en línea]. [consultado en agosto de 2013]. Disponible en internet: <http://www.raspberrypi.org/quick-start-guide>

Figura 30. Raspberry Pi



Fuente: tomada de RASPBERRY. Raspberry Pi [en línea]. [consultado en agosto de 2013]. Disponible en internet: <http://www.raspberrypi.org/quick-start-guide>

5.2.3.2 Udo. La UDOO es una plataforma de desarrollo basada en una computadora de una sola tarjeta que combina un ARM de doble núcleo o cuádruple Freescale Cortex-A9 i.MX.6 CPU y un Arduino incorporado con un ARM CPU Atmel SAM3X8E dedicado.

Udoo surgió del trabajo conjunto de SECO USA Inc. y Aidilab, en colaboración con un grupo de investigadores con experiencia en sistemas embebidos, redes de sensores y ciencia cognitiva.

Figura 31. Udo Quad

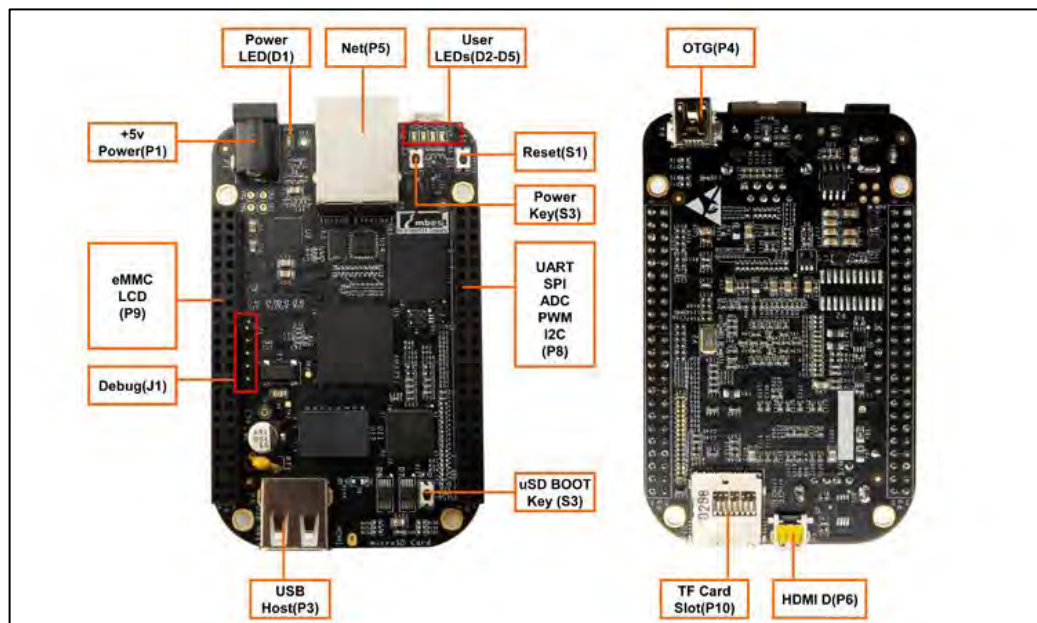


Fuente: tomada de: UDOO. Udo Quad, Getting Started. [En línea]. 2014. [Consultado en Diciembre de 2013]. Disponible en internet: <http://www.udoo.org/>

Esta tarjeta está conformada por un ARM cortex-A9 compatible con sistema operativo Android y Linux. Además posee un procesador Atmel SAM3X8E ARM Cortex-M3 igual al Arduino Due, memoria RAM DDR3 de 1GB, módulo wifi, conexión sata, salida HDMI y funciona a 12V. Actualmente esta plataforma es muy utilizada en aplicaciones de ingeniería multimedia por su alta capacidad de procesamiento y es uno de los mejores en su especie³¹.

5.2.3.3 Beaglebone Black. La Beaglebone black es la última versión de una serie de tarjetas llamadas beagleboard desarrolladas por algunos empleados de Texas Instruments y otros investigadores en estados unidos.

Figura 32. BeagleBone Black



Fuente: tomada de: BEAGLEBONE. BeagleBone Black, Getting Started. [En línea]. 2014. [Consultado 5 de Diciembre de 2013]. Disponible en Internet: <http://beagleboard.org/getting-started>

Esta tarjeta está conformada por un procesador ARM Cortex-A8 de 1 GHz, con 512 MB de memoria en la tarjeta (DDR3 RAM), con un almacenamiento de 2 GB con un sistema Linux preinstalado (no se requiere de una tarjeta SD) y más aún, con un conector MicroHDMI para entrada y salida de video y audio.

³¹ UDOO. UdoO Quad, Getting Started. [En línea]. 2014. [Consultado en Diciembre de 2013]. Disponible en internet: <http://www.udoo.org/>

Esta tarjeta está orientada al desarrollo de sistemas embebidos y es muy utilizada para desarrollo netamente hardware. Con 74 puertos de entrada/salida, 5 puertos seriales, 8 salidas PWM y 7 convertidores analógico/digital a 1.8v máximo; es la hace una plataforma en la cual se pueden desarrollar una infinidad de aplicaciones³².

5.3 LENGUAJES DE PROGRAMACIÓN PARA ROBOTS MÓVILES

Actualmente existe gran variedad de lenguajes empleados en la programación de robots móviles. Estos lenguajes son los mismos empleados en otras áreas como la informática, pero contienen librerías o toolbox diseñados específicamente para la programación de este tipo de plataformas. Sin embargo existen algunos modelos de robots, plataformas de desarrollo y fabricantes, que poseen su propio lenguaje de programación, los cuales conforman un grupo especial de robots. Pero en términos generales la mayoría de los lenguajes de programación actualmente conocidos, sirven para programar un robot móvil. Entre los más utilizados se encuentran: Java, C, C++ y Phyton.

5.3.1 Java. Es un lenguaje de programación de propósito general compilado e interpretado. Esto significa que todo programa hecho bajo este lenguaje es primero compilado y luego interpretado por una máquina virtual. Una de las principales características de java es que es un lenguaje orientado a objetos, lo cual es muy útil para la programación de robots móviles, pues se pueden tratar como objetos, lo que implica que es posible crear métodos y atributos propios de cada aplicación. Un ejemplo de esto se puede observar en la librería que java tiene para la plataforma LEGO NXT.

Cabe resaltar que aunque java se basa en C/C++ no es una mejora ni tiene alguna otra relación con estos otros lenguajes³³.

5.3.2 C/C++. C es un lenguaje de programación de propósito general de alto nivel, por lo que es una evolución de los lenguajes B y BCPL, en los cuales el programador no podía manejar tipos de datos. Aunque este lenguaje fue creado para la implementación de sistemas operativos, actualmente se emplea para la programación de dispositivos electrónicos como microcontroladores o tarjetas de procesamiento digital de señales (DSP).

³² BEAGLEBONE. BeagleBone Black, Getting Started. [En línea]. 2014. [Consultado 5 de Diciembre de 2013]. Disponible en Internet: <http://beagleboard.org/getting-started>

³³ BELMONTE Fernández. Oscar. Introducción al lenguaje de programación Java. Una guía básica. [En línea]. Junio de 2005 [consultado en agosto de 2013]. Disponible en internet: <http://www3.uji.es/~belfern/pdidoc/IX26/Documentos/introJava.pdf>

C++ es la evolución del lenguaje C pues introdujo el paradigma de programación orientado a objetos. De igual forma es un lenguaje de propósito general y entre sus más relevantes características está el poder definir y crear funciones que se comportan como si fueran originales.

5.3.3 Python. Es un lenguaje de programación orientado a objetos muy sencillo y dinámico, el cual se puede aprender en muy poco tiempo. Al igual que Java, es un lenguaje interpretado cuyo intérprete posee compatibilidad con el lenguaje C/C++. Una de las principales características de Python es su filosofía de código abierto, lo que lo hace óptimo para el desarrollo de nuevas aplicaciones en el campo de la investigación.

Actualmente es un lenguaje que de gran aplicación en la programación de plataformas de robótica móvil como el LEGO NXT³⁴.

5.4 ENTORNOS DE PROGRAMACIÓN PARA ROBOTS MÓVILES. ROS

ROS (Robot Operating System) es un sistema operativo de código abierto para robots. ROS proporciona los servicios que se esperan de un sistema operativo, incluyendo abstracción de hardware, control de bajo nivel de dispositivos, transferencia de mensajes entre procesos y gestión de paquetes. También proporciona herramientas y librerías para obtener, construir, escribir y ejecutar código a través de múltiples computadores. ROS hasta ahora funciona solo en sistemas Unix y es similar en algunos aspectos a “robot frameworks” tales como Player, Yarp, Orocos y Microsoft Robotics Studio.

Por estar diseñado para operar bajo sistemas Unix, ROS por recomendación de sus creadores, debe instalarse en el sistema operativo Linux Ubuntu.

ROS además permite integrar sistemas tanto software como hardware, lo cual hace que sea un sistema abierto de modo que se pueda integrar plataformas hardware como Arduino, con una tarjeta BeagleBone Black. En cuanto a software, ROS no se limita a un solo lenguaje, siendo posible tener códigos escritos en Python, C/C++ o Java, los cuales se integran sin problema.

ROS maneja una estructura de datos bastante elemental conformada por paquetes, los cuales son el elemento más atómico que se puede encontrar. Un

³⁴ VAN ROSSUM. Guido. El tutorial de Python. [En línea]. Septiembre de 2009. [consultado en agosto de 2013]. Disponible en internet: <http://docs.python.org.ar/tutorial/pdfs/TutorialPython2.pdf>

paquete no es más que un directorio o carpeta que contiene otras subcarpetas con un significado específico e igualmente importante para el correcto funcionamiento de ROS. Dentro de un paquete de ROS se encuentran la carpeta SRC que es donde se almacenas los nodos o programas escritos en C++; la carpeta scripts que es donde se almacenan los nodos escritos en lenguaje Python; un archivo de configuración llamado “CmakeList.txt” encargado de definir la compilación y ejecución de cada uno de los nodos contenidos en el paquete; y un archivo llamado “package.xml”, el cual contiene información importante como la versión, el autor, sus dependencias y alguna otra información exportada de otros paquetes³⁵.

5.4.1 Nodo. La forma en que ROS funciona está basada en la comunicación, es decir, envío y recepción de mensajes entre módulos llamados nodos. Un nodo no es más que un programa escrito en C++ o Python que publica un mensaje en un topic o se suscribe al mismo para recibir mensajes.

Los nodos en ROS deben contener las librerías propias de ROS, las cuales proporcionan los métodos y funciones para que un programa pueda ser considerado como un nodo de ROS y pueda funcionar bajo la estructura de este sistema operativo. Las librerías se llaman ROSPY y ROSCPP para Python y C++ respectivamente³⁶.

5.4.2 Topic. Es un nombre que se utiliza para identificar el contenido de un mensaje que un nodo publica o está preparado para recibir. Por ejemplo un nodo que está diseñado para recibir un mensaje de tipo String cuyo contenido es una frase como “Hola mundo” se suscribirá a un topic llamado Mensaje para de esta forma recibirlo. De igual forma un nodo que publica este mismo tipo de mensajes publicara la frase “Hola mundo” en el topic llamado Mensaje. Gráficamente podría ser de la siguiente manera:

Figura 33. Funcionamiento Básico de ROS

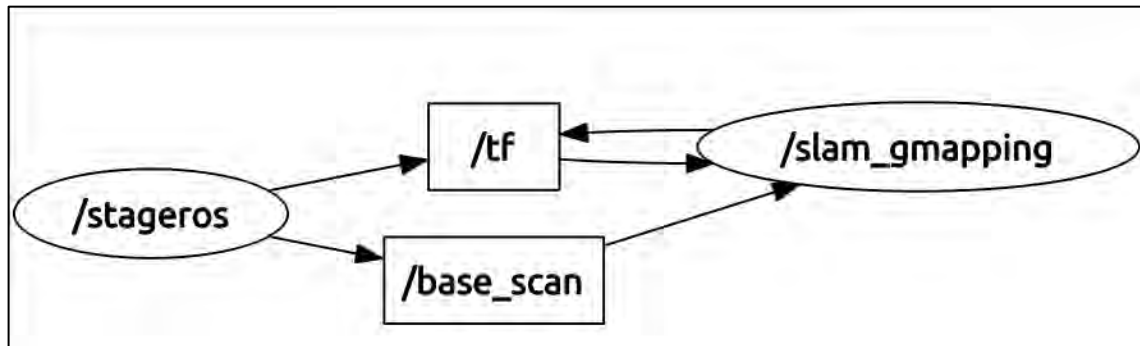


³⁵ ROS. Robotics Operative System. [En línea]. [Consultado en febrero de 2014]. Disponible en internet: <http://wiki.ros.org/>

³⁶ROS. Robotics Operative System. [En línea]. [Consultado en febrero de 2014]. Disponible en internet: <http://wiki.ros.org/>

Este es un claro ejemplo del funcionamiento de ROS. Cabe resaltar que es posible tener un nodo publicando en varios tópicos (topics o temas) a la vez o un nodo suscribiéndose a varios topics a la vez. Figura 34.

Figura 34. Ejemplo del Funcionamiento de ROS



Fuente: tomada de: ROS. Robotics Operative System. [En línea]. [Consultado en febrero de 2014]. Disponible en internet: <http://wiki.ros.org/>

Cabe resaltar que en las figuras 33 y 34 los círculos hacen referencia a los nodos, mientras que los cuadrados identifican los topics.

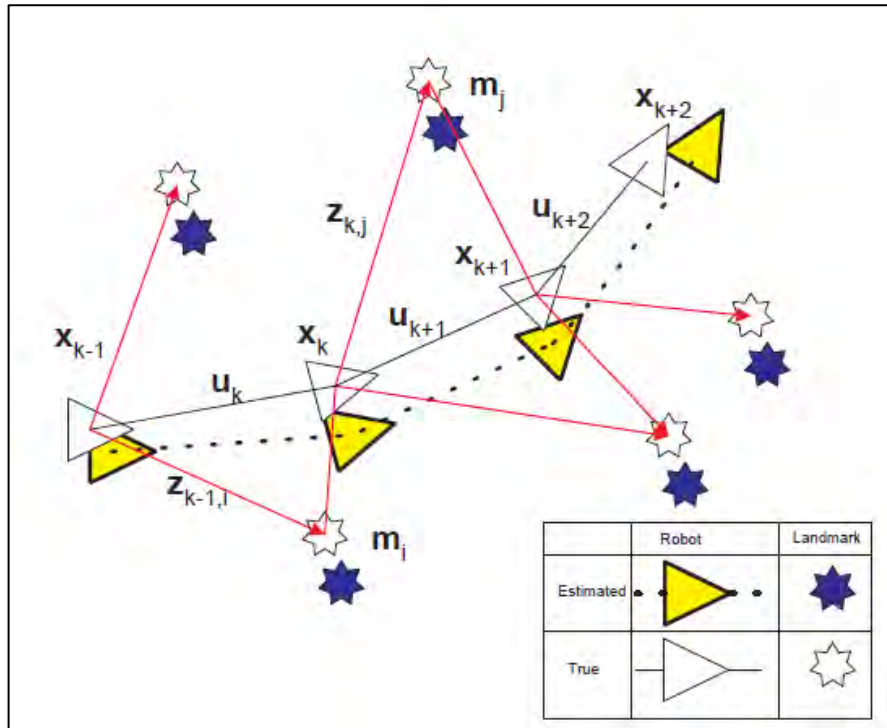
5.5 SLAM

SLAM (Simultaneous Localization And Mapping, o Localización Y Mapeado Simultáneos) es una técnica o método utilizado en robótica móvil para construir mapas en 2D o 3D de entornos físicos en los cuales un robot opere, al mismo tiempo que se conoce la posición del mismo dentro de dicho entorno. [Gil12]

La información necesaria para la construcción del mapa se obtiene de los sensores propioceptivos y exteroceptivos que el robot móvil posee. Por lo regular la información del entorno se obtiene con sensores como los ultrasonidos o LIDAR, mientras que la información de la posición, se obtiene de encoders conectados a las ruedas del robot (odometría).

Debido a que es casi imposible obtener información exacta del entorno y aún más difícil conocer con precisión la ubicación dentro del mismo, el SLAM es visto como un problema y las mejores soluciones se encuentran en la aplicación de técnicas probabilísticas.

Figura 35. El Problema de Slam



Fuente: tomada de DURRANT Whyte, Hugh; Fellow; IEEE; Bailey, Tim. Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. [En Línea]. [Consultado en marzo de 2014]. Disponible en internet: http://www.cs.berkeley.edu/~pabbeel/cs287-fa09/readings/Durrant-Whyte_Bailey_SLAM-tutorial-I.pdf

Existen varias técnicas de SLAM, algunas con ventajas sobre otras. El EKF-SLAM, fundamentado en los trabajos realizados por Randall Smith, Matthew Self y Peter Cheeseman es una de las técnicas más extendidas y con mejores resultados, la cual utiliza el FILTRO EXTENDIDO DE KALMAN, pero a costa de un alto consumo computacional³⁷.

Otra técnica muy conocida pero un poco menos efectiva es la llamada MAPAS DE OCUPACIÓN DE CELDILLAS (Occupancy Grid Mapping), la cual se basa en discretizar el entorno dividiéndolo en una especie de grilla, cuyas unidades son de tamaño predefinido y se clasifican como ocupadas o vacías con un determinado nivel de confianza o probabilidad. Esta técnica es la que utiliza el algoritmo de GMAPPING, desarrollado por Giorgio Grisetti; Cyrill Stachniss; Wolfram Burgard; y

³⁷ GIL. Jose. Un Acercamiento Entre Ros, Kinect Y Una Plataforma Móvil Propia Para La Ejecución De Técnicas SLAM. Universidad Autónoma de Occidente – Cali 2012

del cual existe un “wrapper” o versión en ROS y es la que se utilizó para validar uno de los objetivos de este trabajo³⁸.

El SLAM es toda una temática de estudio muy extensa y compleja propia de un arduo trabajo de investigación, por lo que para este caso se trabajó como una caja negra utilizada para validar el funcionamiento del sistema si entrar en detalle de su estructura interna.

³⁸ GIL. Jose. Un Acercamiento Entre Ros, Kinect Y Una Plataforma Móvil Propia Para La Ejecución De Técnicas SLAM. Universidad Autónoma de Occidente – Cali 2012

6 METODOLOGIA APLICADA

En este proyecto se empleó una investigación descriptiva sobre la robótica móvil, sistemas de sensado, actuadores, sistemas micro procesados y embebidos.

Para la realización de este proyecto, las técnicas de recolección de información fueron:

- La revisión documental: Se utilizaron recursos como libros, proyectos de grado, artículos de revistas o formatos científicos que proporcionaron información teórica y conceptual de los diferentes dispositivos electrónicos que conforman el proyecto en general.
- La experimentación: La experimentación se efectuara cuando inicie el proceso de diseño e implementación de las diferentes etapas que hacen parte del proyecto global.

Cuadro 5. Etapas de desarrollo del proyecto.

| Ítem | Actividad | Precedencia | Duración (Días) |
|------|--|-----------------|-----------------|
| 1 | Estudio de plataformas de robótica móvil disponibles en la universidad | | 4 |
| 1.1 | Verificar estado actual | | 8 |
| 1.2 | Realizar inventario | 1.1 | 3 |
| 1.3 | Estudio y selección de sensores | 1.2 | 8 |
| 1.4 | Estudio y selección de actuadores | 1.2 | 8 |
| 2 | Estudio de plataformas hardware disponibles | 1 | 8 |
| 2.1 | Selección de los dispositivos para la unidad central de procesamiento del robot móvil. | 1.1 1.2 1.3 1.4 | 8 |
| 3 | Diseño del sistema de control de la plataforma | 2 2.1 | 15 |
| 3.1 | Selección de la etapa de potencia | 1.2 | 8 |
| 3.2 | Selección de la etapa de adquisición de datos | 1.3 | 8 |
| 4 | Implementación | 3.1 3.2 2.1 | 30 |
| 4.1 | Probar odometría | 3.2 | 15 |
| 4.2 | Probar actuadores | 1.2 | 15 |
| 4.3 | Implementar un control de una trayectoria | 4.1 4.2 4.3 | 15 |
| 5 | Prueba de capacidad de procesamiento con un algoritmo de SLAM | | 15 |

7 DISEÑO E IMPLEMENTACION

7.1 DISEÑO HARDWARE

El diseño hardware comenzó por realizar un estudio de las plataformas hardware disponibles en el mercado actualmente. Con esto se obtuvo información relevante que ayudo a la escogencia de la plataforma a usar, teniendo en cuenta aspectos fundamentales como disponibilidad y facilidad de importación, características técnicas adecuadas a las necesidades, precio y tamaño.

En la tabla 5.1 se muestra las cuatro plataformas más reconocidas en el mercado actualmente y que son adecuadas para las necesidades del proyecto. Como se puede observar, la mejor de este grupo es la Udo, pues sus características técnicas están sobredimensionadas con respecto a los requerimientos del proyecto, pero uno de los inconvenientes que presenta es la disponibilidad para adquisición (compra) pues su fabricante las vende por encargo, es decir, las fabrican para cada cliente que realiza un pedido. Por otra parte su precio es el más elevado con respecto a las demás plataformas.

Debido a lo anterior se optó por adquirir en primera instancia la tarjeta beaglebone black, ya que es mucho más fácil de conseguir a nivel nacional en ciudades como Bogotá o Medellín. Además su precio es mucho más económico y durante el proceso de recolección de información e investigación de este proyecto se encontraron diversas aplicaciones y publicaciones de trabajos afines con la temática del presente trabajo. Por lo anterior se optó por iniciar el proceso de desarrollo con esta tarjeta. No obstante se realizó la adquisición de la Udo pero cuando finalmente arribó ya se tenía muy adelantado el desarrollo con la Beaglebone black por lo que se optó por terminar el trabajo bajo esta plataforma y como prueba final tratar de replicarlo en la Udo. Más adelante en el capítulo 6 se hará referencia a los resultados de la utilización de esta tarjeta, los cuales no resultaron como se esperaba debido a que esta tarjeta es muy nueva y todavía no existen drivers compatibles con sistemas operativos diferentes a los que provee y recomienda su fabricante.

Cuadro 6. Plataformas Hardware más Adecuadas para el proyecto

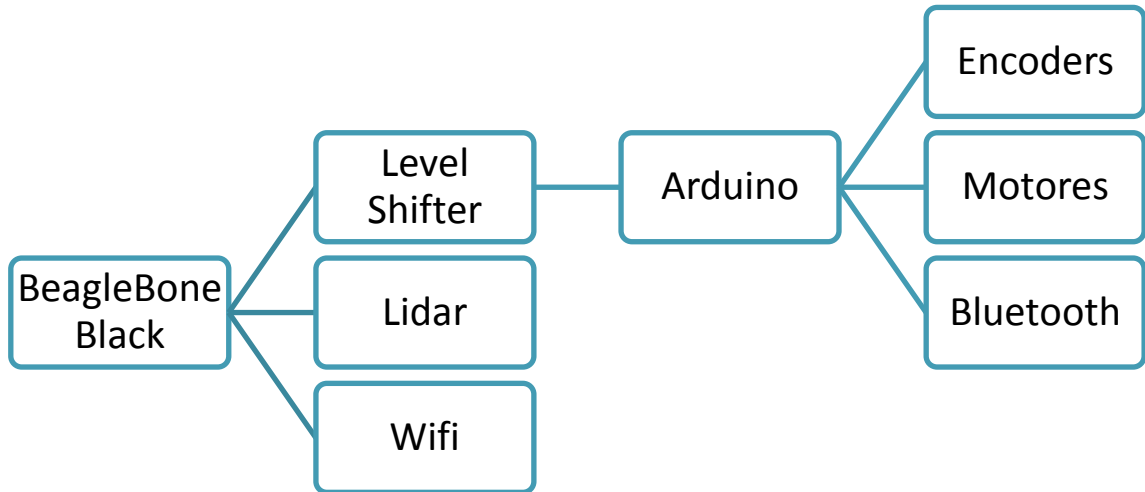
| Tarjeta | Modelo | Procesador | Alimentacion | Entradas Analogas | Salidas Analogas | Entradas Digitales | Salidas Digitales | GPIO | Precio | RAM | Almacenamiento | Conexiones de video | Resolucion Soportada | Audio | Sistemas Operativos | Consumo de energia | Perifeticos |
|--------------|-----------|-------------------------------------|--------------|-------------------|------------------|---------------------|-------------------|------|--------|-------------------------|-----------------------------|---------------------|--|---|---|--------------------|--|
| Arduino | Due | AT91SAM3X8E-85MHz | 7-12V | 12 | 2(DAC) | 54(12PWM) | 54(12PWM) | 54 | 39€ | 96KB | 512KB | X | X | X | X | 600mA | USB-SERIAL |
| Raspberry Pi | B | ARM1176JZF5-700MHz | 5V | X | X | 26 | 26 | 26 | 35USD | 512 MB SDRAM @ 400 MHz | 32GB via SD | 1HDMI, 1 Composite | Desde 640x350 hasta 1920x1200, incluye 1080p | Stereo a través de HDMI y jack de 35 mm | Raspbian (Recomendado), Ubuntu, Android, ArchLinux, FreeBSD, Fedora, RISC OS, others... | 150-350mA @ 5V | 2 USB Hosts, 1 Micro-USB Power, 110/100Mbps Ethernet, FPCamera connector |
| Beagle Board | BoneBlack | 1GHz TI Sitara AM3359 ARM Cortex A8 | 5V/2A | 7 | X | 65 | 65 | 65 | 40USD | 512 MB DDR3 L @ 400 MHz | 4 GB on-board eMMC, MicroSD | 1 Micro-HDMI | 1280x1024 (5:4), 1024x768 (4:3), 1280x720 (16:9), 1440x900 (16:10) all at 16 bit | Stereo a través de HDMI | Angstrom (Default), Ubuntu, Android, ArchLinux, Gentoo, Minix, RISC OS, others... | 210-460mA @ 5V | 1 USB Host, 1 Mini-USB Client, 110/100Mbps Ethernet |
| Udoo Quad | | 4 X ARM CORTEX A-9 1GHz | | 12 | 2 | 6214 ANALOG/DIGITAL | 62 | 62 | 135USD | 1GB DDR3 | | HDMI/LVDS | | | | | 3 USB 2.0 / SATA |

En esta etapa se realizó un diseño a alto nivel de todo el sistema basado en un conjunto de etapas o subsistemas dedicados a una tarea específica. El diseño de este sistema se basó en la interconexión de plataformas de desarrollo como arduino y la tarjeta beaglebone black de Texas instruments principalmente. Para lograr la interconexión de estas plataformas hardware se empleó una comunicación serial full-duplex con una interfaz hardware en medio, compuesta por un convertidor lógico de nivel de voltaje, pues la beaglebone black funciona con un nivel de tensión de 3.3 v mientras que el arduino lo hace a 5v.

De igual forma se diseñaron dos subsistemas diferentes de comunicación inalámbrica. El más importante de ellos es el que está basado en tecnología wifi, el cual permite inicializar y detener el sistema remotamente por medio del protocolo SSH (Secure Shell). Por otro lado se encuentra el subsistema basado en tecnología bluetooth, el cual emplea un protocolo de comunicación serial y tiene como función recibir comandos para que el sistema realice tareas muy específicas y de menor complejidad como enviar información acerca de sensores, nivel de la batería etc.

Adicionalmente se diseñó un subsistema de monitoreo dedicado a medir constantemente el nivel de la batería principal de todo el sistema.

Figura 36. Diagrama del Hardware del Sistema



En la figura 36 se muestra un esquema general de los componentes del sistema y como se encuentran interconectados. Cabe resaltar que el sistema de comunicación wifi que aparece en este diagrama es un dispositivo externo que se conectó directamente a la beaglebone black por medio de su puerto ethernet. Los encoders, motores y el sensor lidar hacen parte de un prototipo de robot móvil tipo diferencial implementado para realizar las pruebas respectivas del sistema de control y validar su funcionamiento.

Haciendo uso de la plataforma de robótica VEX se implementó una estructura móvil en configuración diferencial, la cual se encarga de brindar movilidad al sistema de control en conjunto con sensores y actuadores. Esto con el fin de contar con todo lo necesario para ejecutar un algoritmo de SLAM sobre el sistema diseñado y de esta manera cumplir con uno de los objetivos de este trabajo. Esta estructura móvil además, fue construida para realizar diversas pruebas sobre ella, por tal motivo cuenta con dos servomotores continuos en cada rueda y dos encoders acoplados de igual forma a cada una de ellas.

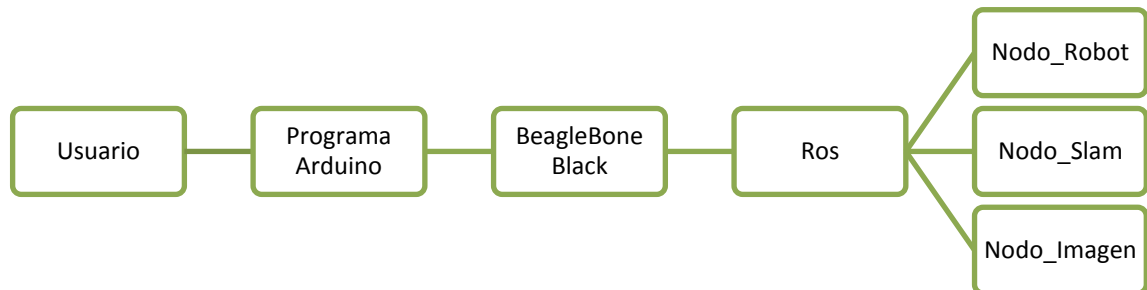
7.2 DISEÑO SOFTWARE

Esta etapa se compone básicamente de dos tipos de software, uno para el arduino y otro para la beaglebone black dentro de los lineamientos del sistema operativo ROS.

El software diseñado para el arduino se encarga de hacer la lectura de cada uno de los datos todos los sensores que necesite leer el sistema y enviarlos de forma serial a la beaglebone black cuando esta se lo solicite. De igual forma este software se diseñó para que el arduino ejecute alguna acción sobre los actuadores de acuerdo a órdenes enviadas desde la beaglebone black o desde un usuario externo.

Por otra parte el software diseñado para la beaglebone black se rige bajo las condiciones que el sistema operativo ROS impone para que un software pueda interactuar con el resto de componentes de ROS. En este caso el nodo diseñado tiene la función de recibir los datos provenientes del arduino, procesarlos en la forma que ROS lo exige y enviarlos dentro de este mismo entorno hacia otros nodos para completar una tarea. En otras palabras, envía la información de forma que el nodo que se encarga de hacer el SLAM los entienda.

Figura 37. Diseño Software

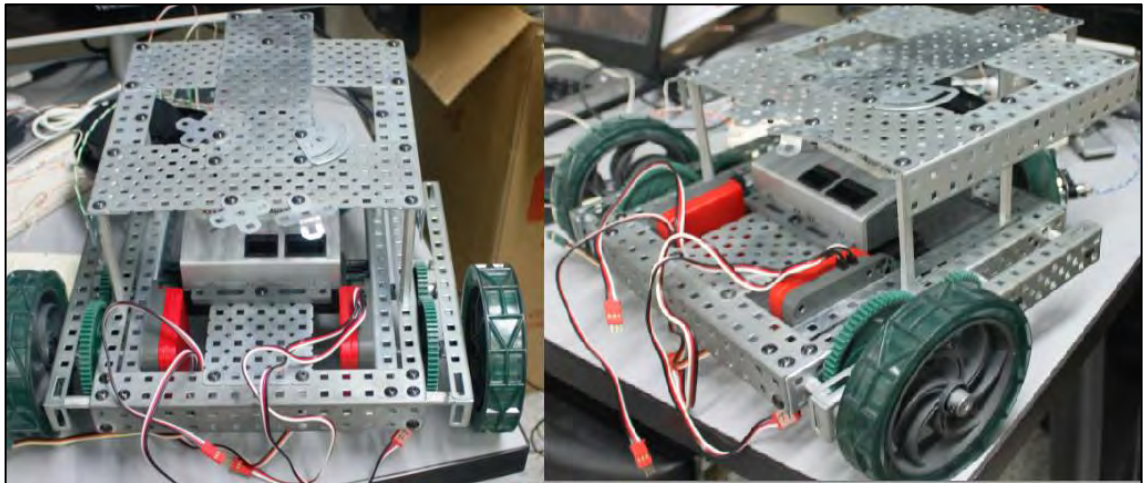


7.3 IMPLEMENTACIÓN HARDWARE

Para dar a entender la forma en cómo se llevó a cabo la implementación en esta etapa se hace la siguiente división con los principales componentes que conforman el hardware del sistema.

7.3.1 Estructura móvil. La parte hardware comienza con una estructura móvil en configuración diferencial, con una rueda loca en la parte de atrás para brindar estabilidad. Esta estructura se desarrolló con la plataforma de robótica VEX, la cual además de permitir construir estructuras mecánicas, ofrece gran variedad de sensores y actuadores para equipar a los robots móviles armados. En este caso la plataforma móvil se equipó con dos servomotores continuos controlados por pwm en cada una de las ruedas, y dos encoders incrementales igualmente acoplados al eje de cada una de ellas. Con el fin de obtener desplazamientos más cortos, las ruedas tienen acopladas piñones, dando una relación de $\frac{1}{2}$ de vuelta, es decir, por 1 vuelta del eje del motor, la rueda realiza media vuelta.

Figura 38. Estructura Móvil



7.3.2 Sensado. Esta etapa fue necesaria implementarla para realizar las pruebas y validar el funcionamiento del mismo. Como uno de los objetivos de este proyecto era poder ejecutar un algoritmo de SLAM, se utilizó un sensor laser tipo lidar, dispositivo fundamental y recomendado para aplicaciones de este tipo.

El sensor utilizado fue extraído de un robot muy comercial llamado neato XV11 fabricado por neato robotics. Este dispositivo viene en con diferentes versiones en

su firmware lo cual hace que la forma en como este entrega los datos cambie. Para este caso, la versión de firmware del sensor es la v2.4.

Figura 39. Sensor XV11 LIDAR



Fuente: tomada de: NEATO HACKING. Neato. [En línea]. 2014. [Consultado en Abril de 2014]. Disponible en Internet: <http://xv11hacking.wikispaces.com/>

Este sensor entrega los datos de distancia utilizando un protocolo de comunicación serial. De acuerdo con la versión de este dispositivo, los datos entregados se encuentran en un formato específico el cual contiene, los datos de distancia, unos datos de checksum o comprobación y los datos de la velocidad a la cual está girando el sensor. Lo último debido a que el sensor posee un motor dc cuyo eje se encuentra acoplado al laser y hace que este gire 360 grados constantemente, lo que permite que el sensor tome medidas de distancia en todas las direcciones. Adicionalmente el motor dc posee un econdor, cuyos datos también se envían en el mismo mensaje o trama en que este dispositivo envía los datos de distancia.

El formato del mensaje está compuesto por 90 paquetes por cada revolución (360 grados) del sensor. Cada uno de estos paquetes contiene 4 lecturas consecutivas de distancia, es decir, el paquete número 1 contiene las lecturas a 0, 1, 2 y 3 grados hasta el paquete 90 que contiene las últimas cuatro lecturas correspondientes a los grados 356, 357, 358 y 359. Cada paquete tiene una longitud de 22 bytes por lo cual la longitud total de bytes para las 360 lecturas es de 1980 bytes³⁹.

³⁹ NEATO HACKING. Neato. [En línea]. 2014. [Consultado en Abril de 2014]. Disponible en Internet: <http://xv11hacking.wikispaces.com/>

Cada paquete está organizado de la siguiente manera:

Figura 40. Contenido de Cada paquete enviado por el sensor LIDAR.

```
<start> <index> <speed_L> <speed_H> [Data 0] [Data 1] [Data 2] [Data 3] <checksum_L> <checksum_H>
```

Fuente: tomada de: NEATO HACKING. Neato. [En línea]. 2014. [Consultado en Abril de 2014]. Disponible en Internet: <http://xv11hacking.wikispaces.com/>

Dónde:

- start siempre tiene el valor de 0xFA.
- Index es el byte de índice para identificar los 90 paquetes, comenzando con el valor 0xA0 (paquete número 0, lecturas 0 a 3) hasta 0xF9 (paquete 89, lecturas 356 a 359).
- Speed son dos bytes que representan la velocidad en la 64^a de RPM (también conocido como valor en RPM representado en punto fijo, con 6 bits utilizados para la parte decimal).
- [Data 0] a [Data 3] son las 4 lecturas. Cada uno tiene 4 bytes de longitud, y está organizado de la siguiente manera:
 - byte 0 : <distancia 7:0>
 - byte 1 : <"dato invalido" flag> <"strength warning" flag> <distancia 13:8>
 - byte 2 : <signal strength 7:0>
 - byte 3 : <signal strength 15:8>

La información de distancia está en milímetros y codificada en 14 bits con un rango de medida entre 0.15m y 6m.

Cuando el bit 7 del byte 1 se pone en 1, esto indica que la distancia no puede ser calculada. Cuando se establece en 1 este bit, significa que el byte 0 contiene un código de error. Los ejemplos de código de error son 0x02, 0x03, 0x21, 0x25, 0x35 o 0x50.

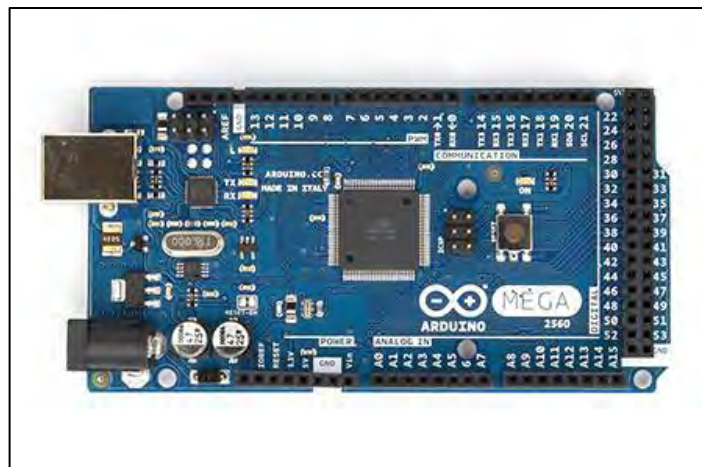
El bit 6 del byte 1 es una advertencia cuando la resistencia reportada es muy inferior a lo que se espera en esta distancia. Esto puede suceder cuando el material tiene una baja reflectancia (material negro), o cuando el punto no tiene el

tamaño esperado o forma (material poroso, tela transparente, rejilla, con el borde de un objeto), o tal vez cuando hay reflexiones parásitas (vaso)⁴⁰.

Otro dispositivo que hace parte de esta etapa es el encoder el cual es muy utilizado en la robótica móvil, ya que permiten obtener la odometría del robot, aspecto fundamental para la realización de SLAM.

7.3.3 Control. Está conformado por una plataforma de desarrollo arduino mega, programado con el software para el mismo y teniendo como función la lectura de los datos provenientes de los sensores, además de enviar de forma serial esta información hacia la tarjeta beaglebone black que es la otra parte que conforma esta etapa de control. Por otra parte el arduino recibe de forma serial e inalámbrica comandos para accionar los motores del robot.

Figura 41. Tarjeta Arduino Mega



Fuente: tomada de: ARDUINO. Arduino DUE. [En línea]. [Consultado en agosto de 2013]. Disponible en internet: <http://www.arduino.cc/>

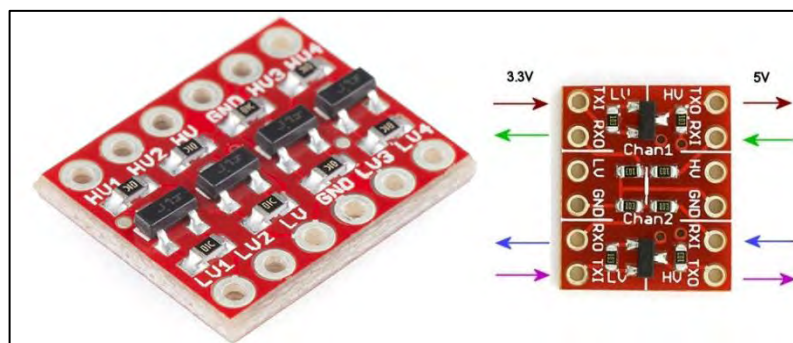
El otro dispositivo que conforma esta etapa y el más importante dentro de todo el sistema es la tarjeta beaglebone black. Con el sistema operativo Ubuntu armhf 13.04 instalado, es el cerebro del sistema. Se encarga procesar toda la información proveniente del arduino y de acuerdo a esto tomar las decisiones para ejecutar las acciones. En ella se encuentra instalado el sistema operativo para robots ROS y las librerías respectivas para el manejo de sus puertos de expansión

⁴⁰ NEATO HACKING. Neato. [En línea]. 2014. [Consultado en Abril de 2014]. Disponible en Internet: <http://xv11hacking.wikispaces.com/>

entre esos los de comunicación serial. Sobre esta tarjeta se ejecuta todos programas para controlar el robot móvil y el algoritmo de SLAM.

Como se expuso anteriormente, esta tarjeta y el arduino mega se comunican de forma serial, lo cual hizo necesario la implementación de un convertidor lógico de nivel de voltaje para evitar algún tipo de daño en cualquiera de los puertos de la BeagleBone black ya que esta opera con un nivel de voltaje de 3.3V mientras que el arduino lo hace a 5V.

Figura 42. Convertidor Lógico de Nivel de Voltaje



Fuente: tomada de: AGELECTRONICA. Tarjeta – Convertidor lógico de nivel. [En línea]. Julio de 2013. [Consultado en Abril de 2014]. Disponible en internet: <http://www.agspecinfo.com/pdfs/B/BOB08745.PDF>

El dispositivo observado en la figura 6.4 posee dos áreas, una para el voltaje alto (HV o 5V) y otra para el voltaje bajo (LV o 3.3V) con dos canales con TX (transmisor) y RX (receptor) respectivamente por cada área. Este dispositivo fue adquirido a un proveedor nacional y se puede considerar un clon del que fabrica

y comercializa la marca sparkfun Electronics. Con este dispositivo se logró la comunicación entre dos puertos seriales de la beaglebone black y del arduino. Uno de estos puertos se utilizó para enviar la información del sensor lidar y el otro para la información de los encoders, todo esto desde el arduino hacia la beaglebone black.

7.3.4 Comunicación. El sistema contiene dos subsistemas de comunicación. El primero y más importante está compuesto por un nanorouter inalámbrico conectado al puerto Ethernet de la tarjeta beaglebone black. Este router se implementó con el fin de poder tener un acceso remoto a la tarjeta y poder ejecutar todas las tareas desde un equipo base, en especial el sistema ROS junto

con todos los nodos para llevar a cabo el SLAM y poder visualizar la imagen del mapa generado. El protocolo de comunicación que se uso fue SSH a través de wifi.

Figura 43. Nano router TP – Link



Fuente: tomada de: TP-LINK. Nano Router Inalámbrico N de 150Mbps. [En línea]. [Consultado en Mayo de 2014]. Disponible en internet: <http://www.tp-link.com/co/>

El otro subsistema de comunicación está compuesto por un módulo bluetooth que convierte una comunicación serial física normal a inalámbrica utilizando tecnología bluetooth. Este módulo de fabricado por Sparkfun Electronics, se implementó con el fin de poder manipular los motores del robot móvil y si se desea, al mismo tiempo poder monitorear el entorno en el cual se encuentra, todo esto de manera remota. Este módulo esta acoplado uno de los puertos serial de la tarjeta arduino mega, la cual en su programa interno es capaz de identificar los comandos recibidos para de acuerdo a estos ejecutar acciones sobre los motores del robot.

Figura 44. Modulo Bluetooth RN-41



Fuente: tomada de: SPARKFUN. Bluetooth Modem - BlueSMiRF Gold. [En línea]. [Consultado en marzo de 2014]. Disponible en internet: <https://www.sparkfun.com/products/12582>

7.3.5 Alimentación. La alimentación de todo el sistema se basa en una batería lipo de 7.4 Volts acoplada a una conexión en Y y un regulador programable BEC programado a 5V. los 7.4V son los que alimentan directamente la tarjeta arduino mientras que los 5V regulados por el BEC son los encargados de suministrar la energía a la tarjeta beaglebone black. Como protección para la batería se implementó un sistema de monitoreo y alarma en el arduino que por medio de uno de sus conversores análogo-digital lee constantemente una de las celdas y avisa cuando el voltaje de la batería se encuentra por debajo de 3.5V.

Figura 45. Regulador BEC (Izq) y Batería lipo(Der)



Por último se realizó una PCB sencilla y modular para interconectar los sensores y motores tanto a la alimentación como a los pines respectivos en cada una de las

tarjetas. Esto con el fin de asegurar una mejor conexión, más segura y con una presentación de mayor calidad.

7.4 IMPLEMENTACIÓN SOFTWARE

En esta etapa fue necesario el diseño y la implementación de software para diferentes tecnologías por medio de diferentes lenguajes de programación, así como la utilización de software ya existente como por ejemplo el programa Putty que permite tener un cliente para conexiones por SSH o terminales para comunicación serial entre otros. Además de la utilización de diferentes distribuciones del sistema operativo Linux para procesadores ARM el cual posee la tarjeta beaglebone black.

7.4.1 Software Tarjeta Arduino. Este programa se encuentra desarrollado en el lenguaje de programación desarrollado por arduino, este se encarga de entregar el programa de funcionamiento al microcontrolador del arduino MEGA quien hace de interfaz entre los sensores, actuadores y el cerebro principal, en este caso la beaglebone black.

7.4.2 Software BeagleBone Black. En esta tarjeta fue en la que más implementación se realizó desde el punto de vista del software. En primer lugar se instaló el sistema operativo Ubuntu armhf 13.04 en una memoria micro sd de 8GB externa, esto debido a que esta tarjeta tiene una memoria interna de solo 2GB lo cual no era suficiente para todo el desarrollo que se haría en ella. En segunda instancia se procedió con la instalación del sistema operativo para robots ROS, en su versión compatible con Ubuntu arm y siguiendo el instructivo disponible en la página web de la organización ros.org.

Ya con el ROS instalado fue necesario la instalación de forma individual de los paquetes de ROS necesarios para llevar a cabo uno de los objetivos de este proyecto que era poder ejecutar un algoritmo de SLAM sobre el sistema diseñado. Dichos nodos (paquetes) fueron básicamente dos el slam_gmapping y el map_server. El primero el más importante, es un “wrapper” o envoltorio que permite utilizar el proyecto gmapping desarrollado por la organización de investigadores de openslam.org, como un nodo de ROS y con el cual se puede crear un mapa en 2D a partir de datos obtenidos de un sensor laser y de la posición acoplados a un robot móvil (SLAM). El segundo nodo permite crear una imagen con extensión .pgm a partir de la información generada por gmapping, es decir, este nodo permite la visualización del resultado final de gmapping

Por último se diseñó e implemento un nodo propio basado en lenguaje de programación Python y las respectivas librerías de este lenguaje para ROS llamada ROSPY. Este se encarga de una gran variedad de tareas como decodificar los datos provenientes del sensor LIDAR a través del arduino y de igual forma los datos provenientes de los encoders. Con el dato de los encoders, este nodo calcula y realiza la estimación de la odometria del robot móvil teniendo en cuenta los valores reales de las articulaciones del mismo, como lo son diámetro de las ruedas, relación de vueltas entre el encoder y la rueda, distancia entre ellas etc. Por otra parte, este mismo crea, organiza y da formato a un mensaje en el cual van adheridos los datos de distancia del sensor LIDAR. Finalmente este mensaje es publicado junto con la odometria calculada para que el gmapping los recoja y genere el mapa para que el map_server pueda generar la imagen del mismo.

Figura 46. Formato del Mensaje LaserScan de los Datos del LIDAR

```
# Single scan from a planar laser range-finder
#
# If you have another ranging device with different behavior (e.g. a sonar
# array), please find or create a different message, since applications
# will make fairly laser-specific assumptions about this data

Header header          # timestamp in the header is the acquisition time of
                        # the first ray in the scan.
                        #
                        # in frame frame_id, angles are measured around
                        # the positive Z axis (counterclockwise, if Z is up)
                        # with zero angle being forward along the x axis

float32 angle_min      # start angle of the scan [rad]
float32 angle_max      # end angle of the scan [rad]
float32 angle_increment # angular distance between measurements [rad]

float32 time_increment # time between measurements [seconds] - if your scanner
                        # is moving, this will be used in interpolating position
                        # of 3d points
float32 scan_time      # time between scans [seconds]

float32 range_min      # minimum range value [m]
float32 range_max      # maximum range value [m]

float32[] ranges       # range data [m] (Note: values < range_min or > range_max should be discarded)
float32[] intensities  # intensity data [device-specific units]. If your
                        # device does not provide intensities, please leave
                        # the array empty.
```

Fuente: tomada de : ROS. Robotics Operative System. [En línea]. [Consultado en febrero de 2014]. Disponible en internet: <http://wiki.ros.org/>

Figura 47. Formato del Mensaje Odometry de los Encoders

```
# This represents an estimate of a position and velocity in free space.
# The pose in this message should be specified in the coordinate frame given by header.frame_id.
# The twist in this message should be specified in the coordinate frame given by the child_frame_id
Header header
string child_frame_id
geometry_msgs/PoseWithCovariance pose
geometry_msgs/TwistWithCovariance twist
```

Fuente: tomada de: ROS. Robotics Operative System. [En linea]. [Consultado en febrero de 2014]. Disponible en internet: <http://wiki.ros.org/>

8 PRUEBAS Y RESULTADOS

Durante el desarrollo de este trabajo se realizaron pruebas, las cuales llevaron a implementar cambios e incluir nuevos dispositivos para la consecución de todos los objetivos planteados al inicio.

En cuanto al prototipo de robot móvil, este no presento cambios durante el proceso de desarrollo, ya que siempre se tuvo claro que iba a ser un robot móvil en configuración diferencial, pues es el tipo de robot móvil con ruedas más sencillo de modelar y por ende su integración con el sistema de control presentaría una complejidad menor.

Con respecto al sistema electrónico de control, se realizaron cambios determinantes y se hizo necesaria la inclusión de otra plataforma como Arduino. Inicialmente las pruebas realizadas se basaron en conocer cómo funcionaba la tarjeta beaglebone black, su sistema operativo y cómo manejar sus puertos, tales como los puertos seriales, entradas analógicas y las salidas/entradas digitales. Se realizaron pruebas conectando sensores de ultrasonido y actuadores como servos, obteniendo buenos resultados con los servos, pero encontrando los primeros inconvenientes en la lectura de sensores de ultrasonido.

En estas pruebas se usaron dos sensores de ultrasonido, el PING fabricado por parallax y el sonar EZ1 fabricado por maxbotix, además de la creación de programas en lenguaje Python y C++. Con el primero se presentaron los inconvenientes más drásticos, pues para leer la información que este entrega se requiere realizar una lectura en su terminal de señal en el orden de los microsegundos, lo cual generaba bloqueos y sobresaltos en la ejecución del script encargado de hacer esta lectura. Con el otro sensor se obtuvieron mejores resultados, pues su lectura se hizo por medio de uno de los conversores analógico digital de la tarjeta. Sin embargo se encontraron errores muy significativos en los valores de distancia obtenidos.

Por otra parte, el sensor PING funciona a 5V, por lo que fue necesario la implementación de una etapa de acondicionamiento de voltaje para evitar el daño del terminal de la beaglebone black ya que esta trabaja a 3.3V.

Todo lo anterior fue determinante para llevar a cabo la inclusión de la plataforma arduino, ya que está diseñada para trabajar a 5V, nivel de voltaje estándar en el cual operan casi todos los sensores y actuadores usados en robótica móvil. Con esta plataforma se logró tener una sola interfaz entre sensores y actuadores, y la

beaglebone black, evitando la implementación de un acondicionador de voltaje por cada sensor y actuador conectado. Además se le disminuyo trabajo computacional a la beaglebone black, pues solo estaba leyendo un puerto serial por donde llegaba la información ya decodificada de los sensores y actuadores.

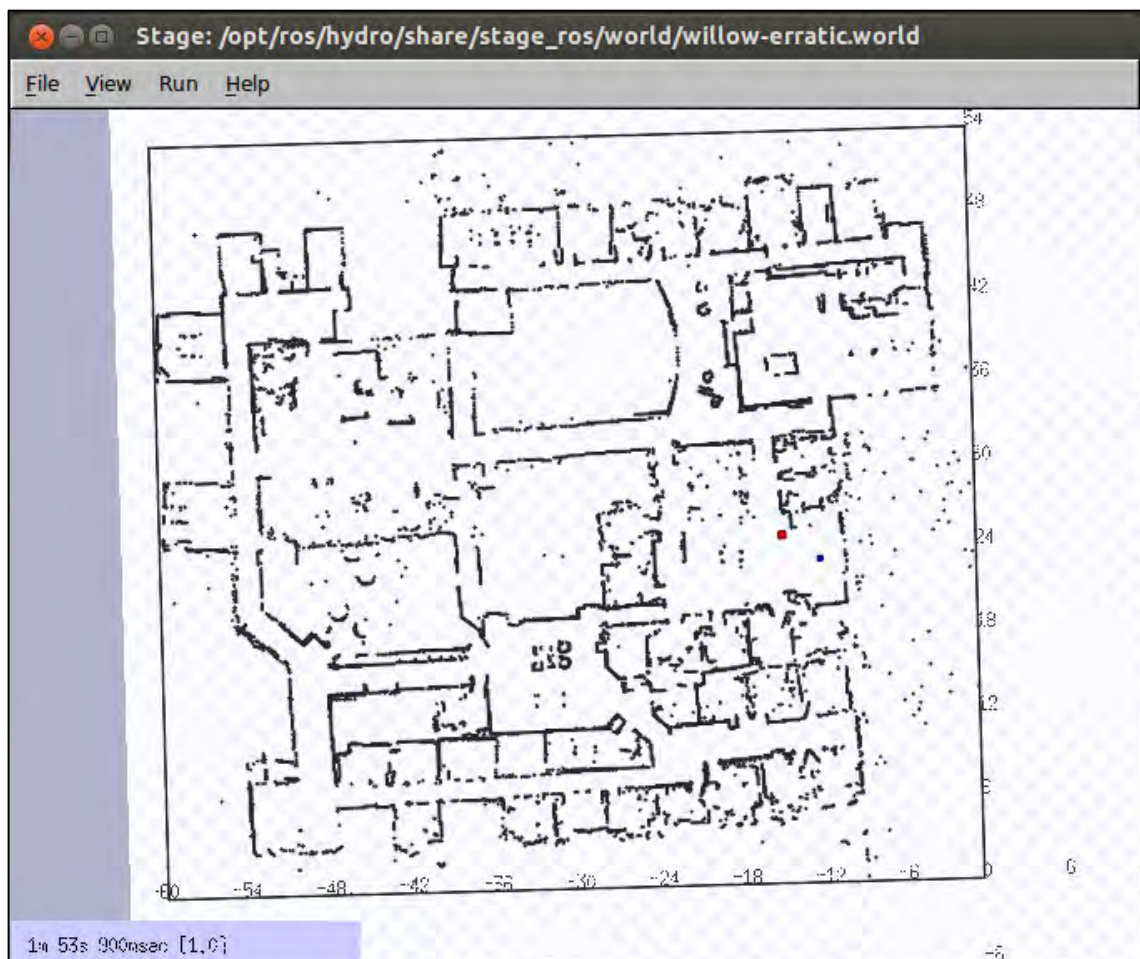
Siguiendo con el cumplimiento de los objetivos planteados, el paso a seguir era poder validar el funcionamiento del sistema de control por medio de la ejecución de un algoritmo de SLAM. Para esto se realizó un estudio de como instalar ROS en la tarjeta BeagleBone Black con sistema operativo Angstrom Linux. El instalar ROS no produjo mayores inconvenientes pues solo se trató de descargar los paquetes respectivos y configurar algunos parámetros. Ya a la hora de crear paquetes y programas propios de ros y compilarlos, surgieron obstáculos de gran importancia. Debido al sistema de archivos del sistema operativo angstrom no era posible compilar los scripts creados bajo el entorno de ros, por lo que se debía realizar un cross-compile (compilación cruzada) de los mismos en un computador externo con sistema operativo Ubuntu con una configuración especial. Cross-compile (compilación cruzada) o un cross-compiler es un compilador que permite crear un archivo ejecutable para una plataforma distinta a la que se encuentra instalado el compilador o en otras palabras crear un código ejecutable para una arquitectura de procesador ARM desde un compilador instalado en computador portátil con procesador INTEL. Luego de obtener el ejecutable se procedía a instalarlo y ejecutarlo como cualquier otro paquete en la tarjeta Beaglebone black, proceso por el cual era necesario atravesar cada vez que se necesitada modificar cualquier cosa por mínima que fuera en el código. Por lo anterior y teniendo en cuenta que el cross-compiler tomaba cerca de 10 minutos se optó por una solución que permitiera el manejo de los puertos de la tarjeta junto con el entorno ROS de una manera similar a como funciona en un computador convencional. Después de una ardua investigación se llegó a la conclusión que el sistema operativo Ubuntu armhf 13.04 era el indicado para suplir dichas necesidades. Este nuevo sistema operativo permitió compilar y ejecutar directamente paquetes de ros y manejar los puertos de la plataforma por medio de librerías desarrolladas por entes externos como los es la desarrollada por Adafruit Industries llamada io-python-library-on-beaglebone-black que como su nombre lo indica está escrita en lenguaje Python.

Con todo lo anterior establecido se procedió con el reconocimiento de ros y específicamente en la comprensión conceptual y del funcionamiento de gmapping y hector_slam, paquetes diseñados para realizar SLAM. Para esto se optó por realizar todo el estudio en un computador portátil con sistema operativo Ubuntu y el ROS instalado. Dentro del sistema ROS existe un paquete llamado stage_ros el cual permite simular un robot móvil equipado con diferentes sensores entre ellos un sensor laser similar al utilizado para este trabajo (XV-11 LIDAR) dentro de un mapa o entorno con diversos obstáculos. Este paquete permitió generar la

información necesaria para que gmapping o hector_slam funcionaran y pudieran generar un mapa ejecutando un algoritmo de SLAM.

Por otra parte el paquete map_server con su nodo map_saver es un paquete de ros que recoge la información generada por gmapping (o hector_slam) y genera una imagen con extensión .pgm con el mapa generado. Por ultimo rviz es una herramienta de ros que permite visualizar en 3D todo el proceso de construcción del mapa viendo en el los movimientos del robot y los resultados que arroja el algoritmo de SLAM.

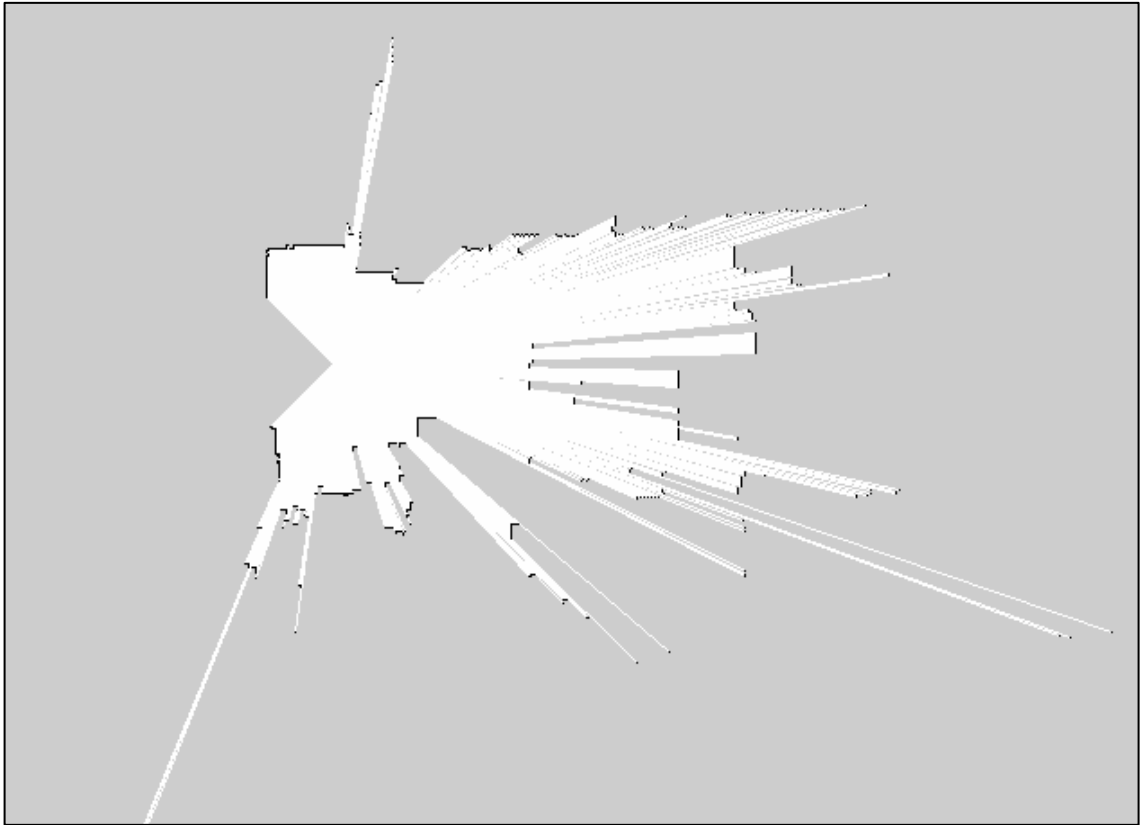
Figura 48. Simulador Stage



En la figura 48 se muestra el simulador stage con el robot (cuadrado de color rojo) dentro del entorno. Con la ayuda de este simulador se pudo establecer con exactitud la manera en que se debía entregar la información de distancia

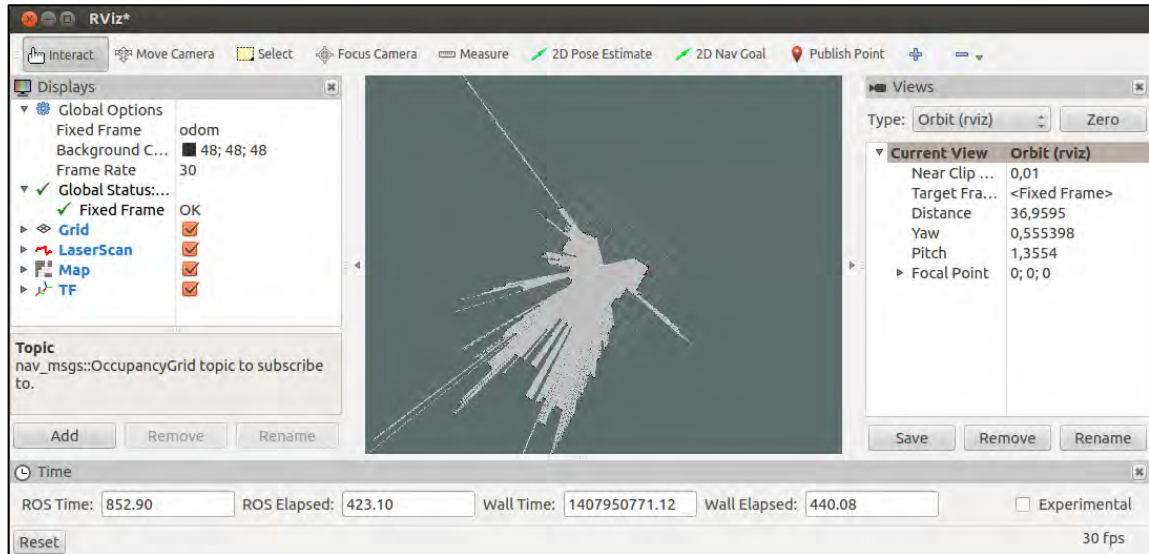
proveniente del láser, y la odometria, al algoritmo encargado de realizar el SLAM, en este caso gmapping.

Figura 49. Mapa resultante con map_saver



En la figura 49 se muestra el resultado obtenido después de ejecutar el nodo map_saver del paquete map_server. En este se puede observar el mapa que hasta el momento ha generado gmapping. Cabe resaltar que este nodo permite ir guardando mapas a medida que el proceso de SLAM se lleva a cabo por lo que lo que ayuda a corroborar si se está realizando o no el slam, ya que un mapa al inicio del proceso estará incompleto en comparación al mapa al final del mismo.

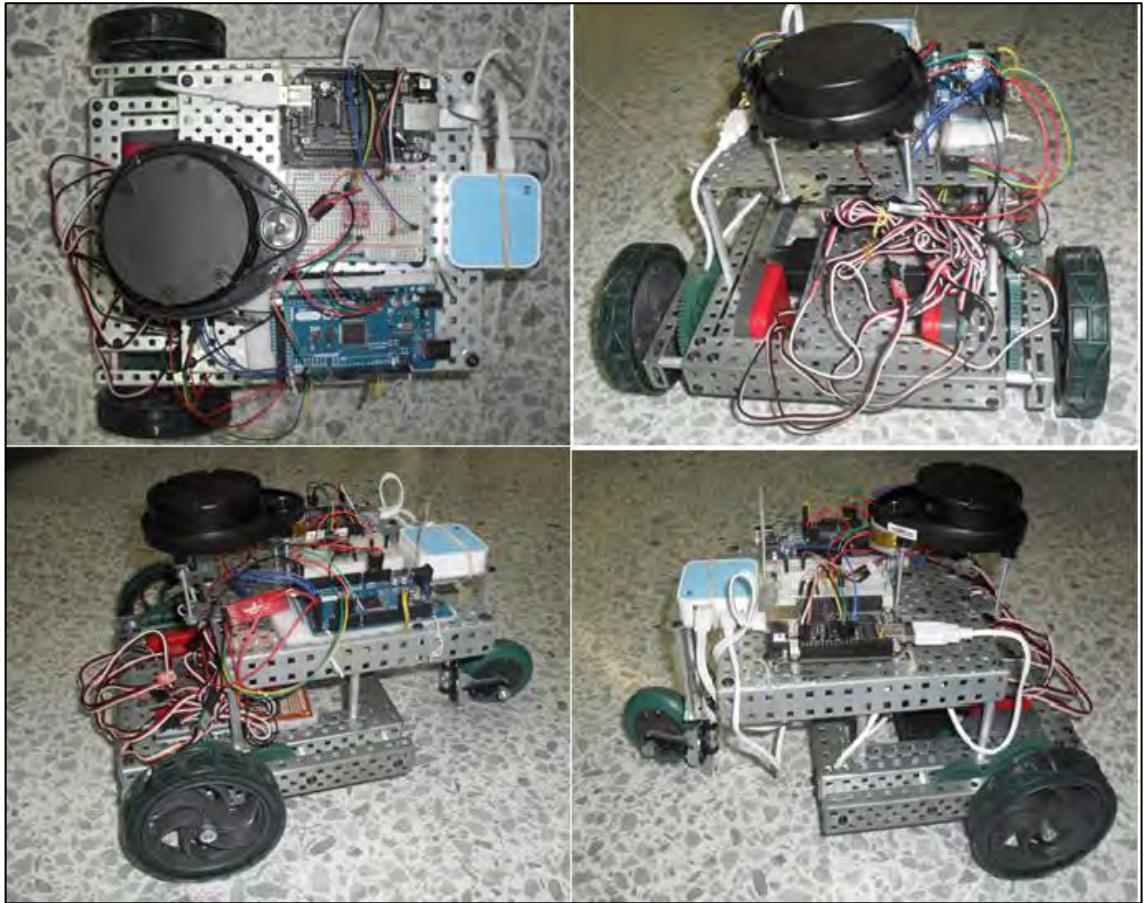
Figura 50. Visualización del Proceso en rviz



En la figura 50 se muestra el resultado del proceso de una forma dinámica y casi que en tiempo real de lo que el robot está percibiendo de su entorno y de lo que genera gmapping.

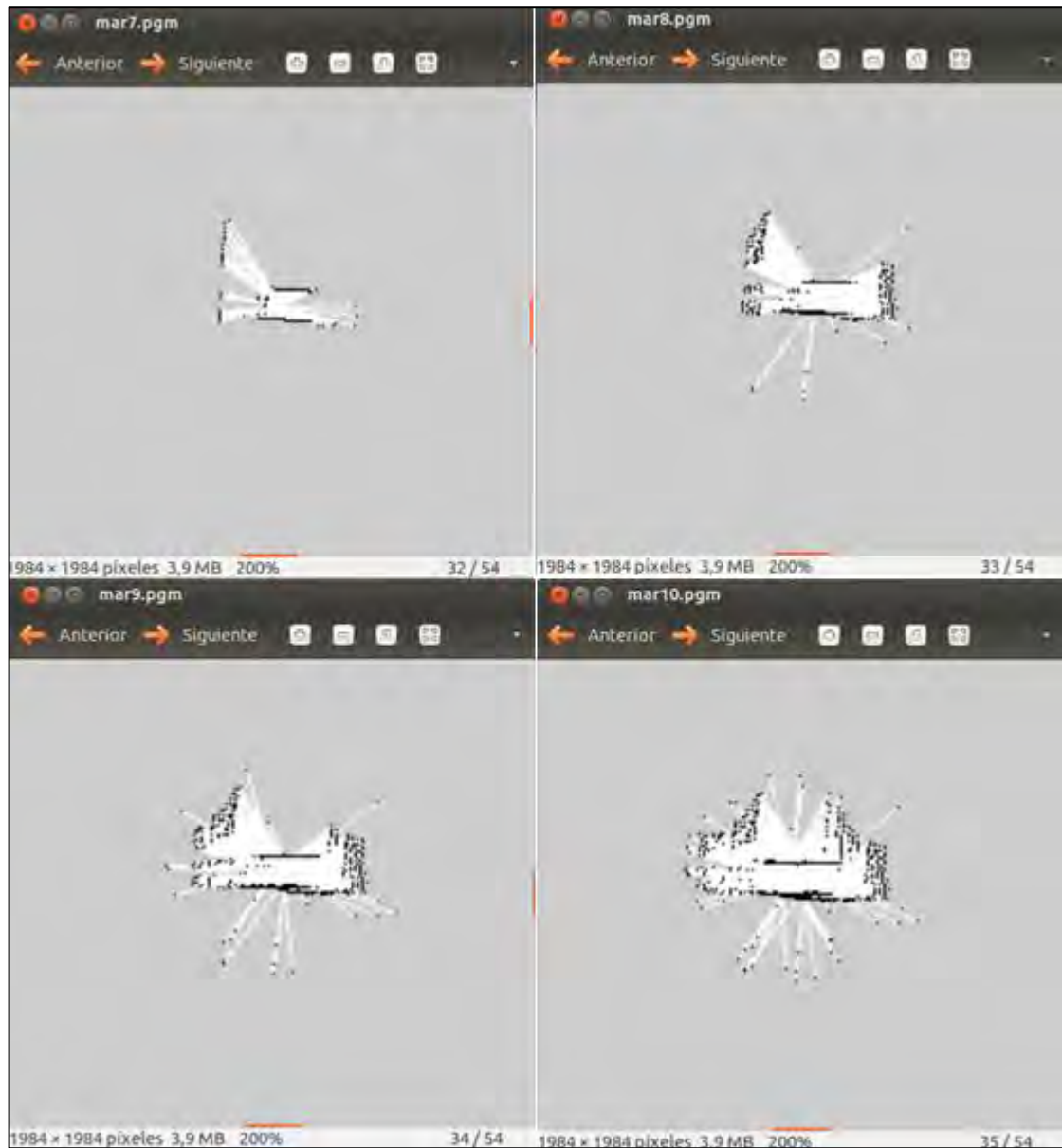
Ya con los resultados de estas pruebas, el siguiente paso consistió en lograr un funcionamiento similar con todo el sistema implementado y acoplado a la plataforma robótica móvil.

Figura 51. Prototipo de robot móvil con el sistema de control acoplado



Las primeras pruebas realizadas con el prototipo completo se basaron en ambientes controlados con formas sencillas como cuadrados, rectángulos y círculos. Los resultados arrojados en estas pruebas no fueron satisfactorios, pues cuando el robot comenzaba a moverse se presentaba una distorsión en el mapa.

Figura 52. Mapa generado a partir del entorno de la figura 53



En la figura 52 se puede observar el resultado obtenido con el robot moviéndose hacia adelante a través de un mapa en forma de L como el mostrado en la figura 53.

Figura 53. Entorno real empleado en la prueba de la figura 52



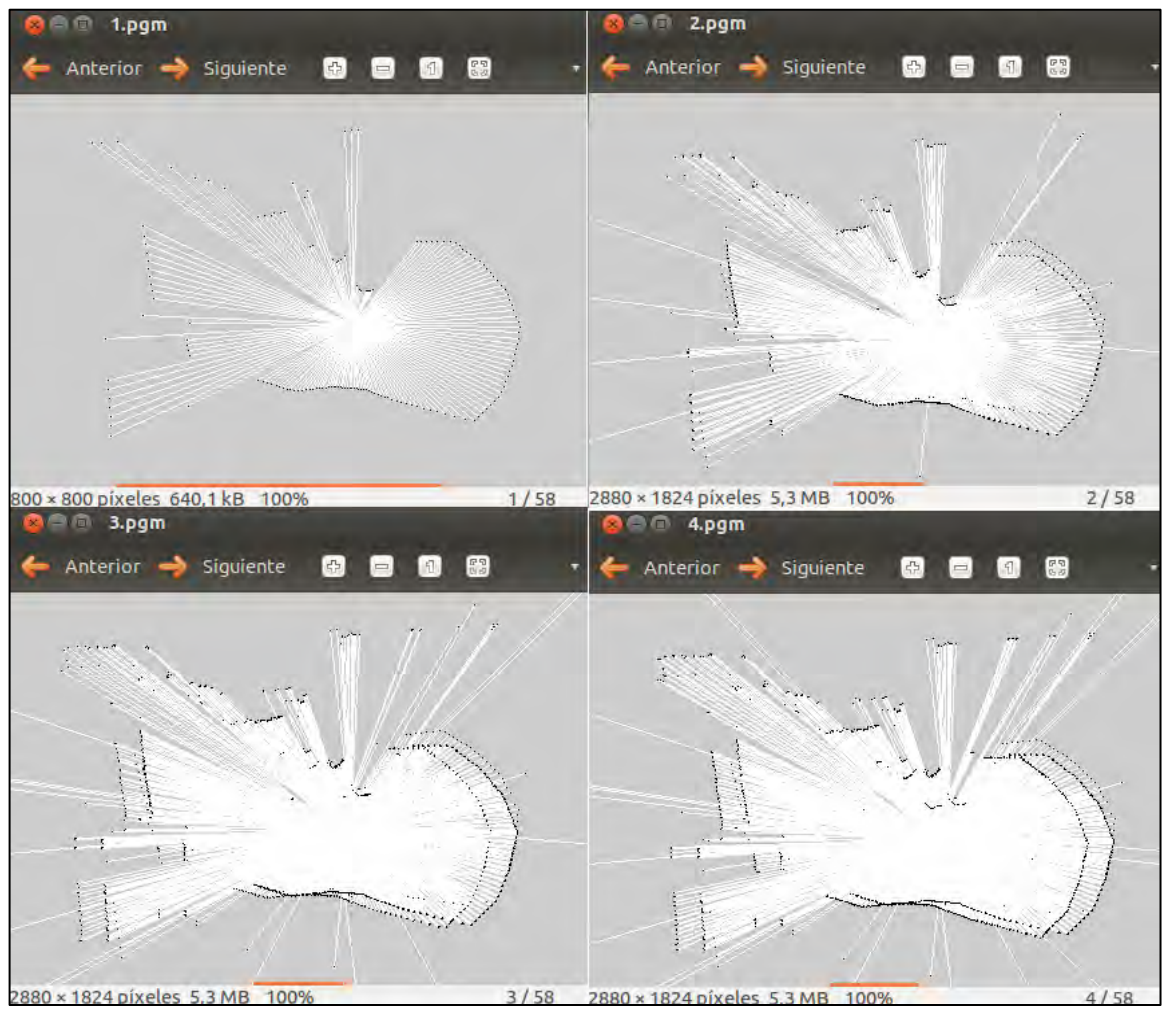
En este punto del desarrollo de este trabajo, se realizó una prueba de todos y cada uno de los componentes del sistema, calibración de encoders con respecto a la distancia recorrida por el robot, código del arduino, código de y configuración de ROS e incluso las baterías que alimentan el sistema. Por otra parte, se realizó una revisión de la configuración del robot con respecto a ROS, es decir, se verificó que el modelo del robot móvil junto con los marcos de referencia tanto del sensor LIDAR como el del robot estuvieran correctamente definidos con respecto al concepto que ROS maneja para su óptimo funcionamiento.

De igual forma se efectuó un estudio y se verificó la configuración de los parámetros de gmapping, encontrando los valores óptimos de dichos parámetros con lo cual se logró obtener mapas de mejor tamaño y resolución que los mostrados en la figura 52.

Figura 54. Entorno real empleado en la prueba de la figura 55



Figura 55. Mapa resultante del entorno de la figura 54



Además de los resultados obtenidos en la figura 55, se logró que la velocidad de ejecución y actualización del gmapping fuera más rápida y por ende reducir el tiempo a menos de la mitad del que se tomaba el sistema en construir el mapa, el cual al inicio de las pruebas era muy elevado, 20 minutos aproximadamente para un entorno de 4m² de área.

Sin embargo, como también se observa en la figura 55, el mapa generado pareciera tener dos mapas, uno más adelante que otro, esto se obtuvo cuando el robot se desplazaba hacia adelante en línea recta.

Debido a todo lo anterior se generó la hipótesis de que la tarjeta BeagleBone Black estaba presentando problemas de procesamiento en cuanto a la capacidad de procesar y ejecutar el gmapping. Por tal motivo se optó por realizar un SLAM "off line", es decir, no al mismo tiempo que se estaba ejecutando el nodo encargado de obtener la información de los sensores y publicarla para que gmapping opere, sino poder almacenar dicha información en un archivo binario y después poder ejecutar este junto con el gmapping solamente. Esto se logró con la ayuda de los archivos BAG de ROS.

Un archivo BAG es un formato de archivo en ROS que permite almacenar datos de mensajes. Como se sabe los nodos en ROS se comunican a través de mensajes como lo es en este caso el tipo de mensaje "sensor message/Laserscan_message" utilizado para hacer llegar la información del LIDAR a gmapping.

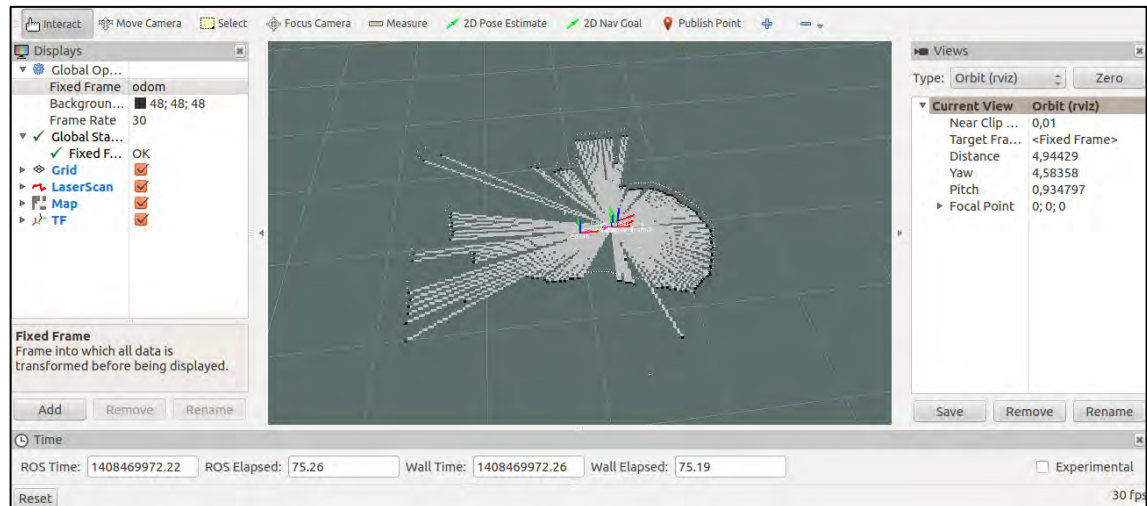
Los archivos BAG (por su extensión .bag) juegan un papel importante en ROS, ya que poseen una serie de herramientas que permiten almacenar, procesar, analizar y visualizar resultados o procesos en cualquier momento por fuera de los mismos.

En este caso el archivo bag almacenó toda la información recolectada y suministrada por el robot móvil con el sensor LIDAR y los encoders para después ejecutar dicho archivo y el gmapping con el fin de descartar que la tarjeta BeagleBone Black tuviera problemas en cuanto a capacidad de procesamiento.

Dicha prueba se realizó en un computador portátil y en la misma tarjeta BeagleBone Black, obteniendo los mismos resultados de la figura 6.8 por lo que se recurrió a realizar otra serie de pruebas con la ayuda de otra herramienta de ROS llamada rviz, el cual es un visualizador que permite visualizar los datos obtenidos por el LIDAR, la posición que va tomando el robot y el mapa que se está

generando con todos los respectivos marcos de referencia. Todo lo anterior se realizó en el computador portátil.

Figura 56. Visualización del Slam en rviz



Con todo lo anterior ahora toda la atención se concentraba en la configuración de gmapping, por lo cual se amplió aún más la investigación acerca de este algoritmo y sus parámetros de configuración. Consultando el foro de ROS <http://answers.ros.org> se encontró que el algoritmo de gmapping es muy riguroso con la odometría del robot móvil, por lo que si no se cuenta con encoders con una buena calibración que permitan una estimación de posición altamente acertada gmapping introducirá una corrección de error que aumenta a medida que aumenta este en la transformación de coordenadas del robot al marco de referencia global o del mapa generado. Dicha corrección es lo que se observa en la figura 6.8 entre un mapa y el siguiente.

Las pruebas siguieron y se realizó un nuevo ajuste al código en el arduino para la lectura de los encoders y el cálculo de la distancia, esto con el fin de mejorar la calibración entre la distancia recorrida por el robot y la distancia reportada por los encoders. Pero los resultados no mejoraron por lo que después de realizar una evaluación de todo el proyecto se llegó a la conclusión de probar con otro algoritmo de slam que posee ROS llamado Hector_Slam.

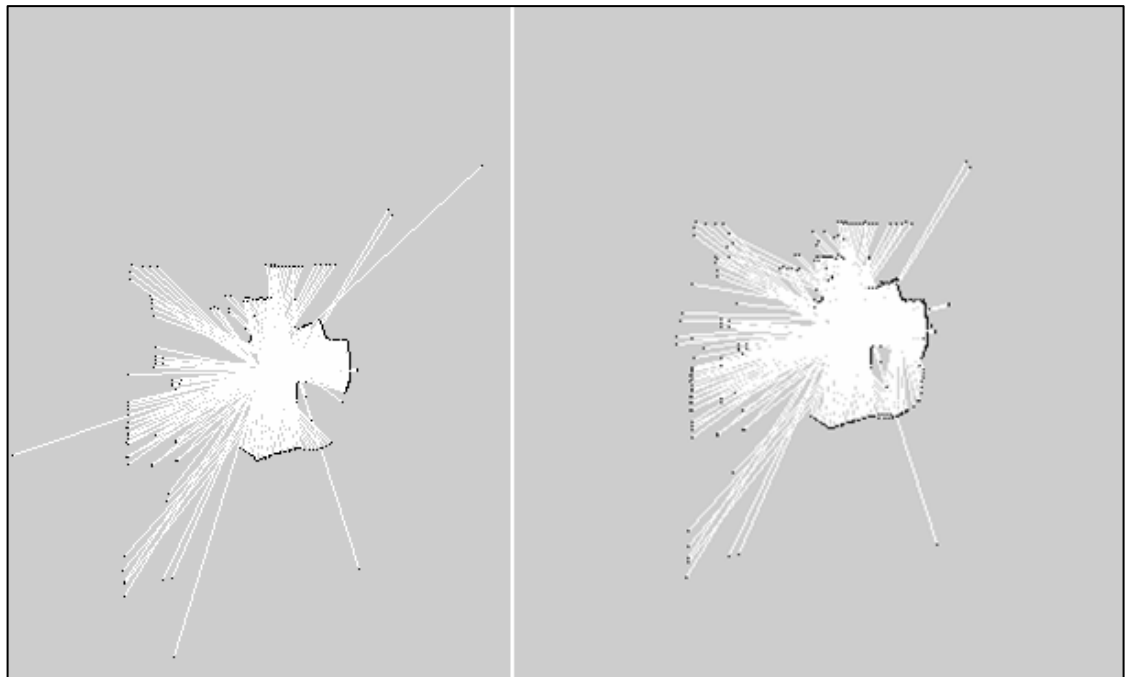
Hector_Slam es un paquete de ROS que contiene un nodo llamado Héctor_mapping el cual funciona con y sin odometria y permite realizar SLAM. De igual forma que gmapping, este necesita suscribirse a un topic llamado scan que transfiere mensajes provenientes desde un sensor LIDAR. Con este paquete

instalado en la tarjeta Beaglebone black se realizaon pruebas con y sin odometria obteniendo resultados satisfactorios. De igual forma se crearon archivos .bag para realizar un análisis en el computador portátil.

Figura 57. Primer entorno utilizado en la prueba con hector_salm



Figura 58. Mapa generado con hector_slam



Como se puede observar en las figuras 57 y 58 los resultados obtenidos son los esperados ya que finalmente se logró un mapa casi idéntico al entorno real y con

un nivel de complejidad más alto debido a que se insertó un obstáculo para validar de que efectivamente el SLAM estaba funcionando de buena forma.

En la parte izquierda de la figura 58 se observa el mapa generado al inicio de la ejecución de todo el sistema. Como es de esperarse el obstáculo impide que el sensor LIDAR pueda detectar lo que hay detrás de este y por eso se observa esa parte de color gris lo que indica que hasta ese momento el robot no sabe que existe detrás del obstáculo.

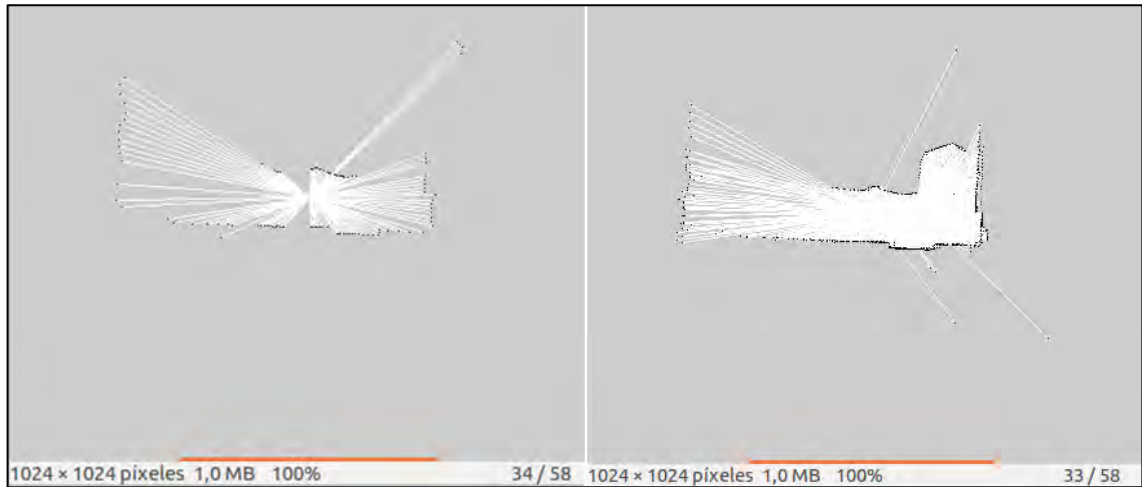
Después de mover el robot hacia adelante y haciendo que pase por el lado izquierdo del obstáculo el sensor LIDAR ya alcanza a detectar lo que hay detrás de este logrando así el mapa mostrado en la parte derecha de la figura 58. Al comparar esta última imagen con la imagen de la parte derecha de la figura 57, se puede apreciar la similitud del mapa generado con el entorno real.

A partir de este punto se realizaron otra serie de pruebas con entornos más grandes y abiertos, y no tan controlados, esto con el fin de medir el alcance del proyecto.

Figura 59. Entorno en forma de L.



Figura 60. Mapa obtenido del entorno en forma de L



En la figura 60 se puede observar el mapa resultante a partir del entorno en forma de L de la figura 59. Este mapa es poseen gran similitud con respecto al ambiente real, salvo por los píxeles alejados. Estos píxeles erróneos surgen debido a errores en la percepción del sensor LIDAR, ya que puede darse que en ocasiones el rayo láser en esa dirección no rebotó, por lo cual el sistema interpreta algún dato de distancia errónea.

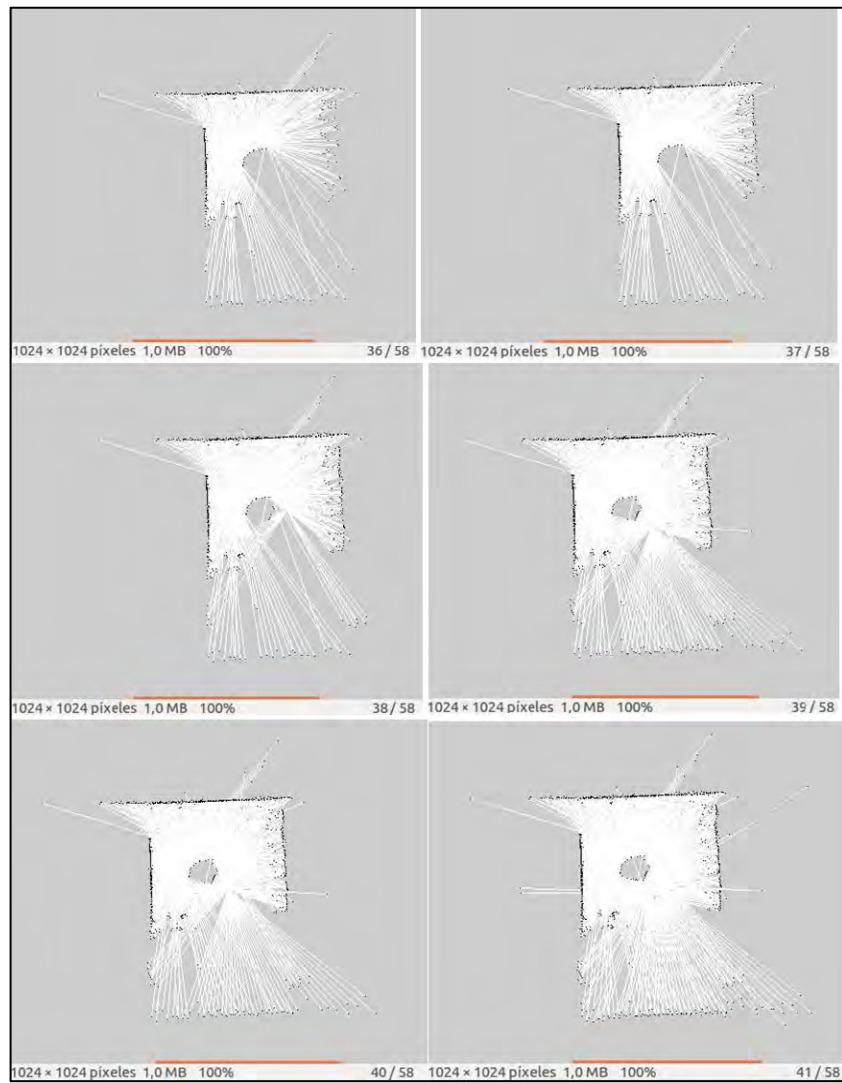
En la imagen de la parte derecha de la figura 60 se observa el mapa generado al inicio del mismo. En esta prueba el robot arranco en la parte recta más larga, de tal forma que no fuera posible detectar la esquina que forma la L en el primer escaneo del sensor. Por tal razón en dicha imagen no se parecía la otra parte que conforma la L.

En la imagen de la parte izquierda de la figura 60 se observa el mapa generado después de que el robot móvil se moviera en línea recta hasta la esquina donde comienza la L y rotar para quedar de frente hacia la parte donde termina el entorno. Un aspecto a resaltar de este mapa, es que en esta prueba se logró que en el mapa generado se pudieran diferenciar detalles como lo son las puertas de color café, el hueco al final de la L y la parte inicial del entorno el cual no se encontraba tapado por las láminas de cartón.

Figura 61. Entorno Abierto con obstáculo



Figura 62. Construcción del mapa del entorno de la figura 61



La prueba cuyos resultados se muestran en la figura 61 se realizó en un ambiente no controlado propio del área de laboratorio de la Universidad Autónoma de Occidente. En este caso se puede apreciar como el SLAM va construyendo el mapa del entorno; en este caso el robot no siguió una trayectoria en línea recta sino que se movió de forma circular alrededor del obstáculo (cuadro de cartón) ubicado casi al centro del entorno. Para evitar la interferencia de personas circulando por el pasillo y hacia la sala que se encuentra al lado del lugar donde se estaba efectuando la prueba, se colocaron sillas acostadas justo enfrente del obstáculo. Estas sillas aparecen en la parte derecha de todas las imágenes mostradas asemejándose a una pared. Los puntos negros que aparecen en el centro más hacia la izquierda son las patas de la silla junto con en la cual se ubicó el computador portátil para el monitoreo externo.

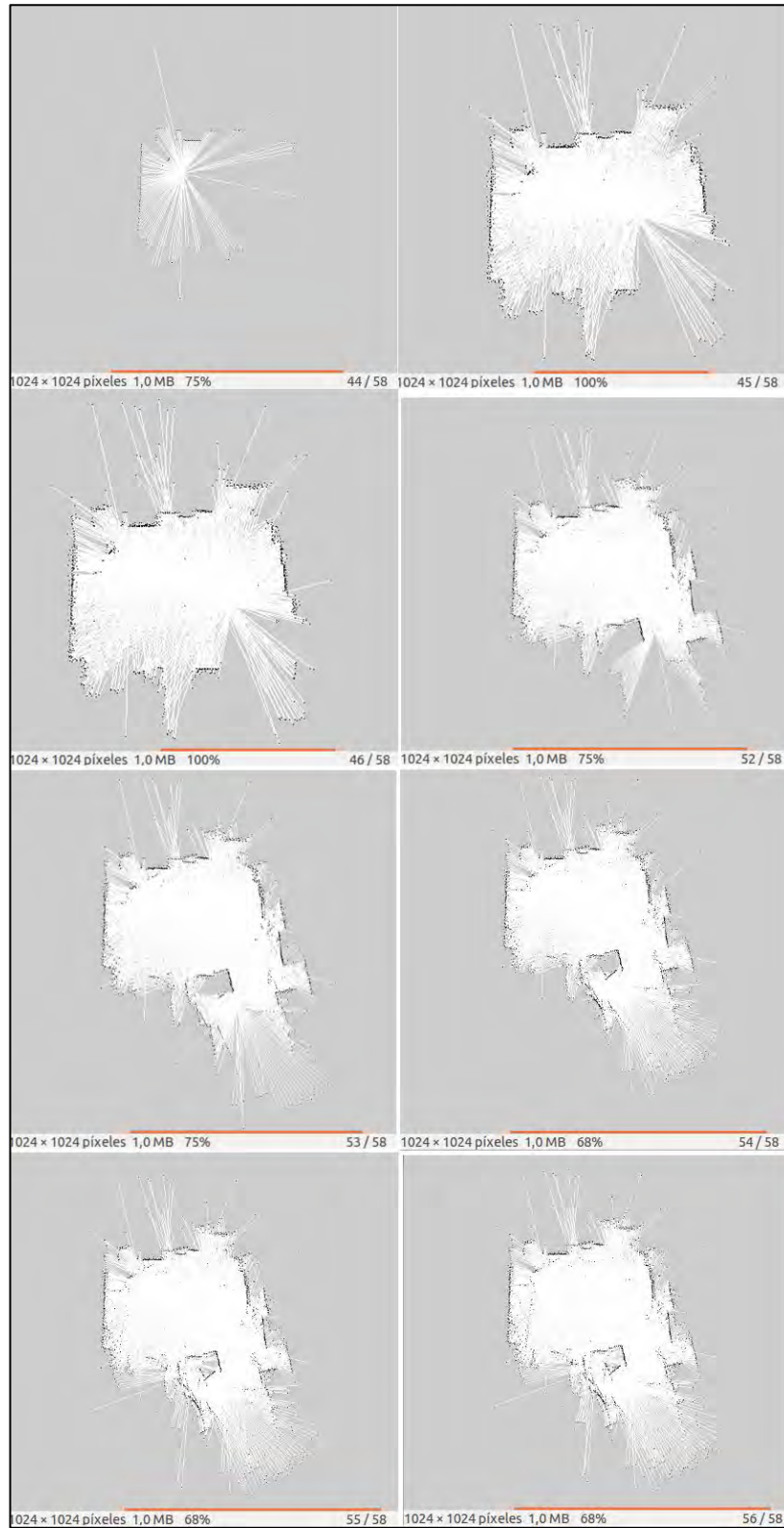
Por último se realizó una prueba en el laboratorio de robótica de la Universidad Autónoma de Occidente. El objetivo principal de esta prueba era realizar el mapa de un entorno que constara de dos espacios distintos pero conectados entre sí, además de perturbaciones naturales del sitio como sillas, mesas puertas y divisiones hechas en vidrio con lo que esto implica para el sensor LIDAR.

En esta prueba el prototipo de robot móvil siguió una trayectoria circular al rededor del laboratorio pasando por debajo de una mesa y regresando al punto inicial, cuya ubicación se encuentra en la esquina superior derecha de cada una de las imágenes mostradas en la figura 63.

En el mapa obtenido se puede observar la mesa, la cual se encuentra localizada en la parte inferior izquierda de la última imagen tomada del mapa generado y se asemeja a una forma cuadrada de color gris rodeada de puntos de color negro.

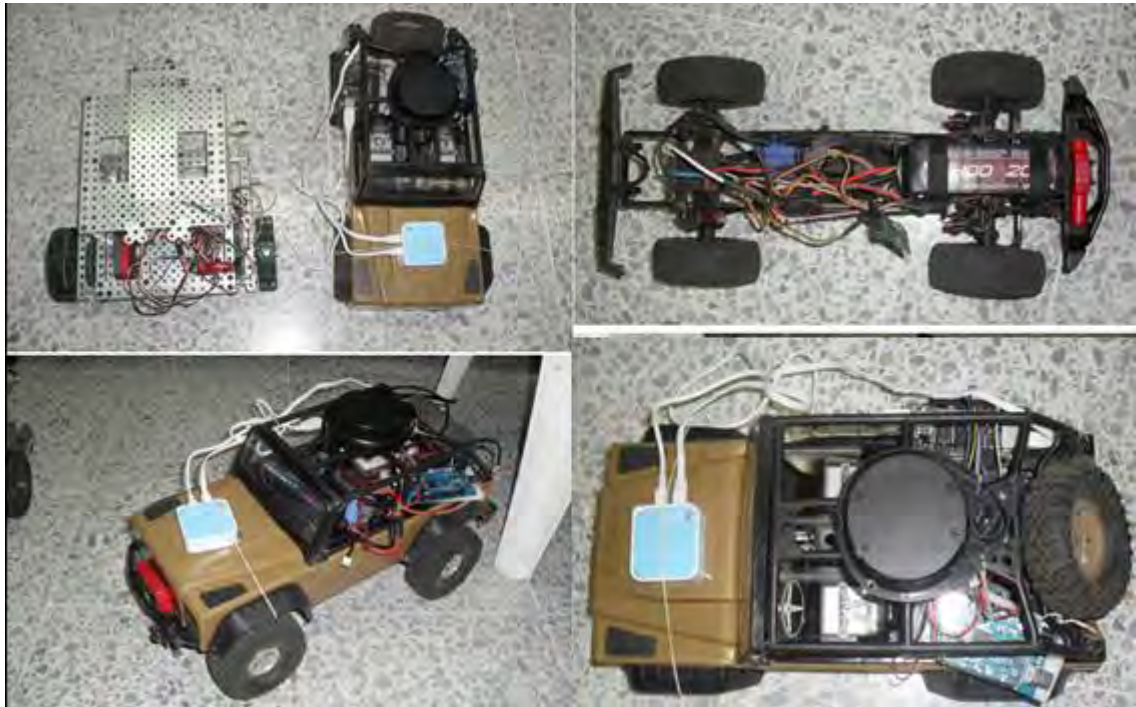
De esta manera se logró medir el alcance del sistema completo obteniendo resultados satisfactorios tanto en ambientes controlados como en los no controlados, además a diferencia de las dos primeras pruebas, en este caso se puso en operación todo el sistema más el prototipo de robot móvil con las baterías LIPO, las cuales brindaron excelente autonomía energética al sistema y al robot.

Figura 63. Mapa obtenido del Laboratorio de Robótica de la UAO



Con el fin de cumplir los objetivos planteados en el anteproyecto se desarmó el prototipo con el cual se habían realizado las pruebas hasta este momento y se acoplo el sistema electrónico a un carro Axial SCX10 ESC (Electronics Speed Controller) en configuración Ackerman, el cual posee un único motor que posee dos transmisiones para las ruedas traseras y delanteras respectivamente. Además posee un servo para la caja de velocidades y otro para la dirección. En cuanto a la parte electrónica posee un “speed controller” y un emisor-receptor de radio frecuencia para que funcione su control remoto.

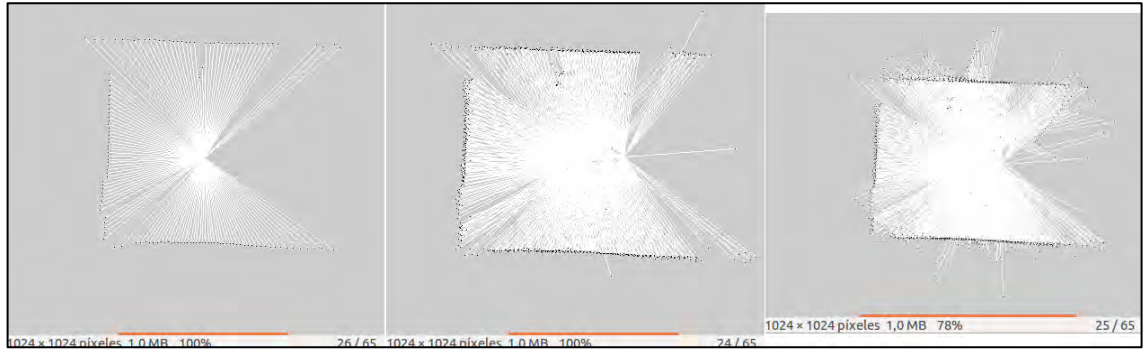
Figura 64. Robot en Configuración Ackerman Axial SCX10 ESC



El objetivo principal de esta prueba era poder validar que el sistema previamente diseñado era capaz de integrarse con otro prototipo de robot móvil distinto al creado para realizar la mayoría de las pruebas. Debido a que esta integración se dio de forma sencilla se optó por ir un poco más allá y se efectuó una prueba de SLAM con el nuevo robot y sistema integrado.

Básicamente se realizaron dos pruebas en este caso, la primera en se basó en desplazar el robot en línea recta a velocidad constante por un pasillo del área de laboratorios de la universidad. Esto con el fin de comprobar si moviéndose a velocidad constante el mapa resultante mejoraba, cosa que no sucedió, pues los resultados fueron similares a los obtenidos con el prototipo anterior.

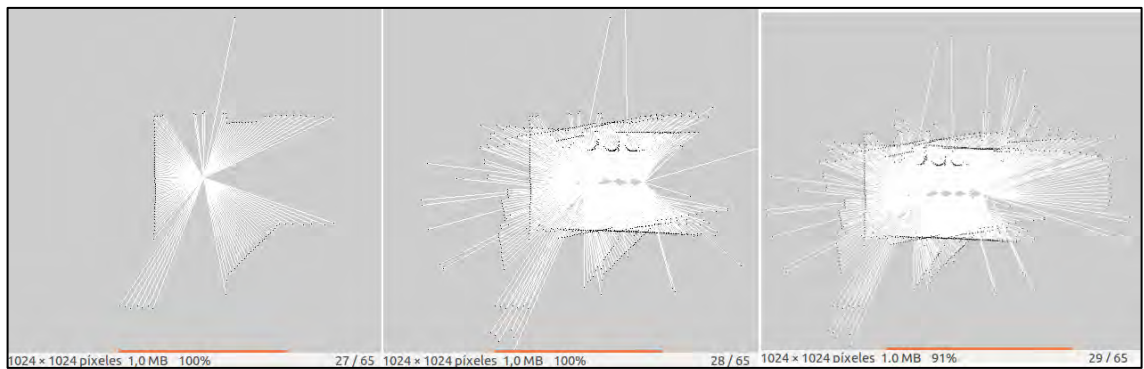
Figura 65. SLAM con Robot Ackerman a velocidad Constante



En la figura 65 se muestra el proceso del mapa obtenido al desplazar el robot a lo largo del pasillo, en este se aprecia como el SLAM va construyendo el entorno en a lado y lado del mapa, esto debido a que el entorno no presentaba cambios como secciones puertas u obstáculos por lo que el mapa resultante no poseen grandes cambios con respecto a su mapa previo.

La segunda prueba de esta última parte consistió en realizar un mapa del mismo pasillo pero moviendo el prototipo a gran velocidad igualmente en línea recta, esto con el fin de comprobar la hipótesis de que el SLAM es mucho mejor cuando los desplazamientos del robot móvil se hacen de forma lenta.

Figura 66. SLAM con Robot Ackerman a gran velocidad



Como se puede apreciar, a medida que el robot avanzaba hacia adelante el mapa generado presenta gran distorsión y se alcanzan a percibir escaneos anteriores al actual. Un análisis más profundo a este y otros problemas encontrados serán descritos de una mejor manera en el último capítulo el cual comprende las conclusiones y recomendaciones para trabajo a futuro.

Por último, como se expuso al inicio del capítulo 5, se trató de instalar el sistema ROS en la tarjeta Udo, obteniendo inconvenientes similares a los encontrados en la Beaglebone Black. Estos problemas están relacionados con el sistema operativo propio de Udo llamado udoobuntu el cual no es compatible con ROS por lo que se realizó el cambio de sistema operativo tal y como lo sugiere la página web oficial de ROS. Con esto se logró tener el ROS ejecutándose sin errores en la udo pero debido al nuevo sistema operativo (Ubuntu ARM) se perdieron los archivos de configuración o drivers para los periféricos de la tarjeta por lo que fue imposible conectar los sensores o realizar algún manejo de los puertos. Hasta la fecha no se ha encontrado alguna librería para el manejo de los puertos para la udo como si se encontró para el caso de la Beaglebone Black con la librería de Adafruit Industries llamada io Python library para dicha tarjeta. Por todo lo anterior se dejó como unidad central de procesamiento para el proyecto a la BeagleBone black y se descartó la Udo, pero se sienta un precedente para trabajo a futuro sobre esta plataforma.

9 CONCLUSIONES

Muchos desarrollos en robótica y en especial en robótica móvil se han visto limitados por que los dispositivos existentes dedicados al procesamiento no permitían la ejecución de algoritmos complejos o la instalación de sistemas operativos para desarrollo abierto.

Actualmente existen robots muy sofisticados desarrollados por marcas reconocidas como EPSON, BOSCH, HONDA, muy capaces pero que operan bajo un sistema operativo propio de su fabricante simplemente están diseñados para ejecutar instrucciones preestablecidas por los mismos. Factor que dificulta el desarrollo de nuevas aplicaciones con dichos robots en cuanto a hardware y software.

Por otra parte, existen los kits de desarrollo para robótica como LEGO o VEX los cuales en el ámbito educativo son de gran aplicación y sirven en gran medida para desarrollar aplicaciones propias del nivel de un pregrado y en muy poca medida en niveles más altos.

Por tal motivo es que dichas plataformas no pueden ser aplicadas para desarrollar soluciones de gran complejidad como lo es el SLAM, no solo por su capacidad de procesamiento sino porque impiden que sistemas como ROS, en la parte software y otras plataformas hardware puedan ser integradas, haciendo que se recurra a dar soluciones poco eficientes como utilizar un computador portátil para realizar tareas de este tipo.

Gracias a plataformas como arduino, desde hace varios años es posible la integración de dispositivos de otras plataformas como VEX con este, permitiendo hacer más abierto el desarrollo de aplicaciones. Un ejemplo de esto es el proyecto citado como antecedente de este trabajo desarrollado en la Universidad Autónoma de Occidente por el Ingeniero Jose Fernando Gil con la dirección del Profesor Juan Carlos Perafan, en donde se diseñó e implemento un robot móvil con las partes mecánicas de VEX, controlado por dos arduinos y capaz de transportar un computador portátil, todo esto con el fin de realizar SLAM.

Todo lo anterior marca el punto de partida del presente trabajo, pues aprovechando plataformas ya existentes como las mencionadas y el surgimiento y auge de nuevas plataformas o dispositivos denominados mini pc, nace la idea de realizar un diseño a alto nivel para concebir un sistema electrónico capaz de integrarse con diferentes tipos de robots móviles terrestres y que además fuera

capaz de ejecutar tareas de gran complejidad como el SLAM manteniéndose abierto tanto en hardware como en software. El resultado final fue un sistema electrónico compuesto por una tarjeta BeagleBone Black conectada a un Arduino por medio de un puerto serial, el cual se encarga de hacer interfaz entre la tarjeta y los sensores y actuadores del robot móvil. El prototipo de robot móvil se construyó con partes de la plataforma de desarrollo en robótica VEX y con una configuración diferencial. Además cuenta con encoders y dos servomotores continuos de la misma marca. Por otro lado se implementaron dos sistemas de comunicación inalámbrica acoplados al sistema electrónico, uno basado en tecnología bluetooth con un módulo conversor bluetooht a serial y el otro basado en un nanorouter inalámbrico. El módulo bluetooht se encarga de recibir los comandos para accionar los motores del robot móvil, mientras el nanorouter permite una conexión inalámbrica desde un pc al robot por medio de una sesión de escritorio remoto utilizando el protocolo SSH.

Después de realizadas todas las pruebas con el sistema completo y el prototipo se puede concluir que en términos generales los sistemas embebidos como la plataforma BeagleBone Black abren un puerta para el desarrollo tecnológico en diversos campos entre ellos el de la robótica en general, reduciendo costos y haciendo abierto tanto hardware como software.

Entrando más en detalle de lo que fue el desarrollo del sistema, se puede concluir que debido a que la tarjeta BeagleBone Black funciona a 3.3 v la inclusión del Arduino como interfaz entre dicha plataforma y los sensores y actuadores, hizo que la carga computacional de la misma se redujera pues la decodificación de encoders y el accionamiento de los motores se efectuó en su totalidad en el Arduino.

El cambio de sistema operativo en la tarjeta Beaglebone Black, produjo un deterioro en la rapidez de operación de la misma, pues según lo recomienda su fabricante el sistema operativo recomendado es la distribución Angstrom Linux, la cual viene de fábrica para las versiones antiguas de esta plataforma o Debian para la versión más actual. Este deterioro se debe básicamente a que además de lo anterior, el sistema operativo Ubuntu ARMhf se instaló y se ejecutó en su totalidad desde una memoria microsd externa, por tal motivo se recomienda usar una memoria de este tipo en su clase que presente mejores características en cuanto a velocidad de transferencia de datos, por ejemplo una microsd clase 10.

Una de las desventajas que implicó el cambio de sistema operativo fue que el sistema de archivos de Ubuntu ARMhf por no estar diseñado exclusivamente para la BeagleBone Black y su kernel no ser totalmente compatible, no contiene el directorio `/sys/class/gpio` para controlar entradas y salidas digitales por ejemplo,

por lo que se recurrió a la instalación de una librería escrita en Python creado por Adafruit Industries para la tarjeta Beaglebone Black, la cual puede ser instalada en sistema operativo Angstrom Linux o Ubuntu ARM y permite controlar, acceder y leer todos sus puertos, desde entradas digitales, pines exclusivos para comunicación SPI, I2C y el usado en este proyecto el puerto serial.

Por otra parte, la principal ventaja que presento el cambio de sistema operativo fue que gracias al sistema de archivos de Ubuntu ARMhf y a su arquitectura de software, permitió que ROS se pudiera ejecutar sin problema y permitiera crear y compilar paquetes propios de forma idéntica a como se haría en un computador lo que facilitó el desarrollo de la parte correspondiente a ROS y por ende la ejecución de SLAM.

ROS es un sistema muy robusto para el desarrollo de forma abierta en el campo de la robótica en general, pues contiene controladores para plataformas reboticas comerciales y permite realizar aplicaciones propias de una forma coherente y con una estructura propia del mismo que al comienzo puede parecer compleja pero que una vez se aprende a manejar potencializa todo trabajo desarrollado bajo sus lineamientos. Por tal motivo se recomienda tener conocimientos básicos en sistemas operativos Linux y lenguajes de programación como Python y C++, pues además de ser muy comunes actualmente en el campo de la robótica y electrónica en general, son los que permiten trabajar bajo licencia abierta.

Los resultados obtenidos de la validación del sistema por medio de un algoritmo de SLAM, confirman que esta problemática lleva desarrollándose desde hace varios años y que muchos de los problemas e inconvenientes que se presentaron en este trabajo están aún sin resolver como es el caso de gmapping, el cual lo que hace es encontrar puntos iguales en dos frames o marcos de referencia consecutivos a través de una transformación de 6 datos, posición y ángulos. Si del primer frame al segundo se presenta un error de transformación, ese será pasado al tercer frame y así que se va acumulando el error. Esto se ve reflejado en los resultados obtenidos en las primeras pruebas en donde se parecían mapas sobrepuestos. Una forma de corregir este error es obtener una buena calibración del elemento sensor, sin embargo siempre se obtendrán mapas no exactos. En otras palabras, gmapping es un algoritmo muy riguroso con la estimación de odometría y si esta presenta un error que aumenta con el tiempo, la corrección que gmapping inserta también aumenta, esto crea un “drift” o deriva en el mapa generado. En conclusión, uno de los factores fundamentales para que este algoritmo funcione es tener una estimación de la odometría de buena calidad, por esta razón las pruebas realizadas con hector_slam sin odometría arrojaron mejores resultados.

Otro factor que pudo influir en los resultados con gmapping es que este por ser tan robusta, depende en gran medida de la configuración de sus parámetros, esto se pudo deducir de las primeras pruebas, ya que la inicio los mapas obtenidos no eran coherentes en cuanto a escala y resolución, pero luego de modificar los valores de los parámetros de configuración, dichos resultados mejoraron notoriamente.

Como trabajo a futuro queda el mejorar o implementar nuevas técnicas más eficientes de estimación de la odometría, mejorar la calibración del sensor y seguir estudios de personalidades como Felix Endres con su trabajo sobre RGB-D SLAM y RGB-D MAPPING de Peter Henry y Dieter Fox, los cuales se encuentran trabajando arduamente en cómo mejorar los resultados en esta área. Por otra parte queda el cambiar la plataforma móvil por una con mejores características mecánicas y en cuanto a la parte hardware hacer una migración a plataformas como la Udo que ya vienen con todo el hardware incluido con lo cual se espera mejore los resultados aquí obtenidos.

BIBLIOGRAFIA

AGELECTRONICA. Tarjeta – Convertidor lógico de nivel. [En línea]. Julio de 2013. [Consultado en Abril de 2014]. Disponible en internet: <http://www.agspecinfo.com/pdfs/B/BOB08745.PDF>

ALTERA. FLEX 10k. [En línea]. [Consultado en agosto de 2013]. Disponible en internet: <http://www.altera.com/literature/ds/archives/dsf10k.pdf>

ALTERA. Kit development university up2. [En línea]. [Consultado en agosto de 2013]. Disponible en internet: <http://www.buyaltera.com/scripts/partsearch.dll?Detail&name=544-1293-ND>

ALTERA. Kit development university de-270. [en línea]. [Consultado en agosto de 2013]. Disponible en internet: <http://www.altera.com/education/univ/materials/boards/de2-70/unv-de2-70-board.html>

ARDUINO. Arduino DUE. [En línea]. [Consultado en agosto de 2013]. Disponible en internet: <http://www.arduino.cc/>

ARTIFICIAL INTELLIGENCE CENTER. Shakey. [En línea]. [consultado en agosto de 2013]. Disponible en internet: <http://www.ai.sri.com/shakey/>

BEAGLEBONE. BeagleBone Black, Getting Started. [En línea]. 2014. [Consultado 5 de Diciembre de 2013]. Disponible en Internet: <http://beagleboard.org/getting-started>

BELMONTE Fernández. Oscar. Introducción al lenguaje de programación Java. Una guía básica. [En línea]. Junio de 2005 [consultado en agosto de 2013]. Disponible en internet: <http://www3.uji.es/~belfern/pdidoc/IX26/Documentos/introJava.pdf>

BROWN, S; Vranesic Z. Fundamentos de Logica Digital con Diseno VHDL. Ed 2. Mexico DF: MacGraw-Hill, 2006.

COMITÉ Español de Automática. Libro Blanco de la Robotica: De la investigación al desarrollo tecnológico y aplicaciones futuras. [En línea]. Julio de 2008. [Consultado en Agosto de 2013]. Disponible en internet: http://www.ceautomatica.es/sites/default/files/upload/10/files/LIBRO%20BLANCO%20DE%20LA%20ROBOTICA%202_v1.pdf

CYPRESS. Psoc. [en línea]. [Consultado en agosto de 2013]. Disponible en internet: <http://www.cypress.com/psoc/>

DURRANT Whyte, Hugh; Fellow; IEEE; Bailey, Tim. Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. [En Línea]. [Consultado en marzo de 2014]. Disponible en internet: http://www.cs.berkeley.edu/~pabbeel/cs287-fa09/readings/Durrant-Whyte_Bailey_SLAM-tutorial-I.pdf

GALEANO, Gustavo. Programacion de Sistemas Embebidos en C. Bogotá: Alfa omega, 2009.

GARCÍA. Belmar. Diseño y Construcción de un Robot móvil controlado con técnicas de lógica difusa implementadas en un FPGA. [En línea]. Septiembre de 2012. [Consultado en enero de 2014]. Disponible en internet: <http://www.sepi.esimez.ipn.mx/electronica/archivos/992.pdf>

GIL. Jose. Un Acercamiento Entre Ros, Kinect Y Una Plataforma Móvil Propia Para La Ejecución De Técnicas SLAM. Universidad Autónoma de Occidente – Cali 2012

GONZÁLEZ, Víctor; López Antonio, Cabero Jose. Curso de Control y Robótica. [En línea]. [Consultado en agosto de 2013]. Disponible en internet: http://platea.pntic.mec.es/vgonzale/cyr_0708/index.htm

HIDALGO. Manuel; Martinez Antonio, Curso Provincial: Control y Robotica en Tecnologia. Robotica Educativa. EducaBot. Motores y Movilidad. [En línea]. Noviembre de 2009 [Consultado en febrero de 2014]. Disponible en internet: http://platea.pntic.mec.es/~mhidalgo/docEducaBot/04_EducaBot_ServomotoresMovilidad02.pdf

LA MAQUINA DE VON NEUMANN. Las tortugas de Grey Walter. [En línea]. [Consultado en agosto de 2013]. Disponible en internet: <http://vonneumannmachine.wordpress.com/2011/05/01/las-tortugas-de-grey-walter/>

MANUELA VALLEJO. Inventos Colombianos En El Campo De La Robótica. [En línea]. [Consultado en agosto de 2013]. Disponible en internet: <http://manuelavallejo.jimdo.com/la-robotica-en-colombia/>

MARQUEZ, G; Osorio, S; Olvera, N. Introduccion a la Programacion Estructurada en C. 1 ed. México DF: PEARSON EDUCATION, 2011.

MUÑOZ David, Andrade Carlos, Londoño Nelson. Diseño y Construcción de un Robot Móvil Orientado a la Enseñanza e Investigación [en línea]. Abril de 2006. [Consultado en agosto de 2013]. Disponible en internet: <http://rcientificas.uninorte.edu.co/index.php/ingenieria/article/viewFile/2490/1632>

NASA. Nasa Education Robotics. [En línea]. [Consultado en agosto de 2013]. Disponilbe en internet: <http://www.nasa.gov/>

NEATO HACKING. Neato. [En línea]. 2014. [Consultado en Abril de 2014]. Disponible en Internet: <http://xv11hacking.wikispaces.com>

PÉREZ, Diego. Sensores de Distancia por Ultrasonidos. [En línea]. [Consultado en enero de 2014]. Disponible en internet: <http://www.alcabot.com/alcabot/seminario2006/Trabajos/DiegoPerezDeDiego.pdf>

RASPBERRY. Raspberry Pi [en línea]. [consultado en agosto de 2013]. Disponible en internet: <http://www.raspberrypi.org/quick-start-guide>

ROBOT REVIEWS. Robot chat. [En línea]. Julio de 2011. [Consultado en abril de 2014]. Disponible en internet: <http://www.robotreviews.com/chat/viewtopic.php?f=20&t=15016>

ROS. Hector_slam. [En línea] 2014 [Consultado 15 de Julio de 2014]. Disponible en Internet: http://www.ros.org/wiki/hector_slam

ROS. Robotics Operative System. [En línea]. [Consultado en febrero de 2014]. Disponible en internet: <http://wiki.ros.org/>

SIEGWART Roland, Nourbakhsh Illah R. Introduction to Autonomous Mobile Robots. 1 ed. Cambridge, London: The MIT Press, 2004.

SPARKFUN. Bluetooth Modem - BlueSMiRF Gold. [En línea]. [Consultado en marzo de 2014]. Disponible en internet: <https://www.sparkfun.com/products/12582>

STANFORD UNIVERSITY. Stanford Cart. [En línea]. [Consultado en agosto de 2013]. Disponible en internet: <http://www.stanford.edu/~learnest/cart.htm>

TP-LINK. Nano Router Inalámbrico N de 150Mbps. [En línea]. [Consultado en Mayo de 2014]. Disponible en internet: <http://www.tp-link.com/co/>

UDOO. Udoos Quad, Getting Started. [En línea]. 2014. [Consultado en Diciembre de 2013]. Disponible en internet: <http://www.udoo.org/>

UNICROM. Microcontroladores Psoc. [En línea]. [Consultado en agosto de 2013]. Disponible en Internet: http://www.unicrom.com/Cmp_microcontroladores_PSOC.asp

VAN ROSSUM. Guido. El tutorial de Python. [En línea]. Septiembre de 2009. [consultado en agosto de 2013]. Disponible en internet: <http://docs.python.org.ar/tutorial/pdfs/TutorialPython2.pdf>

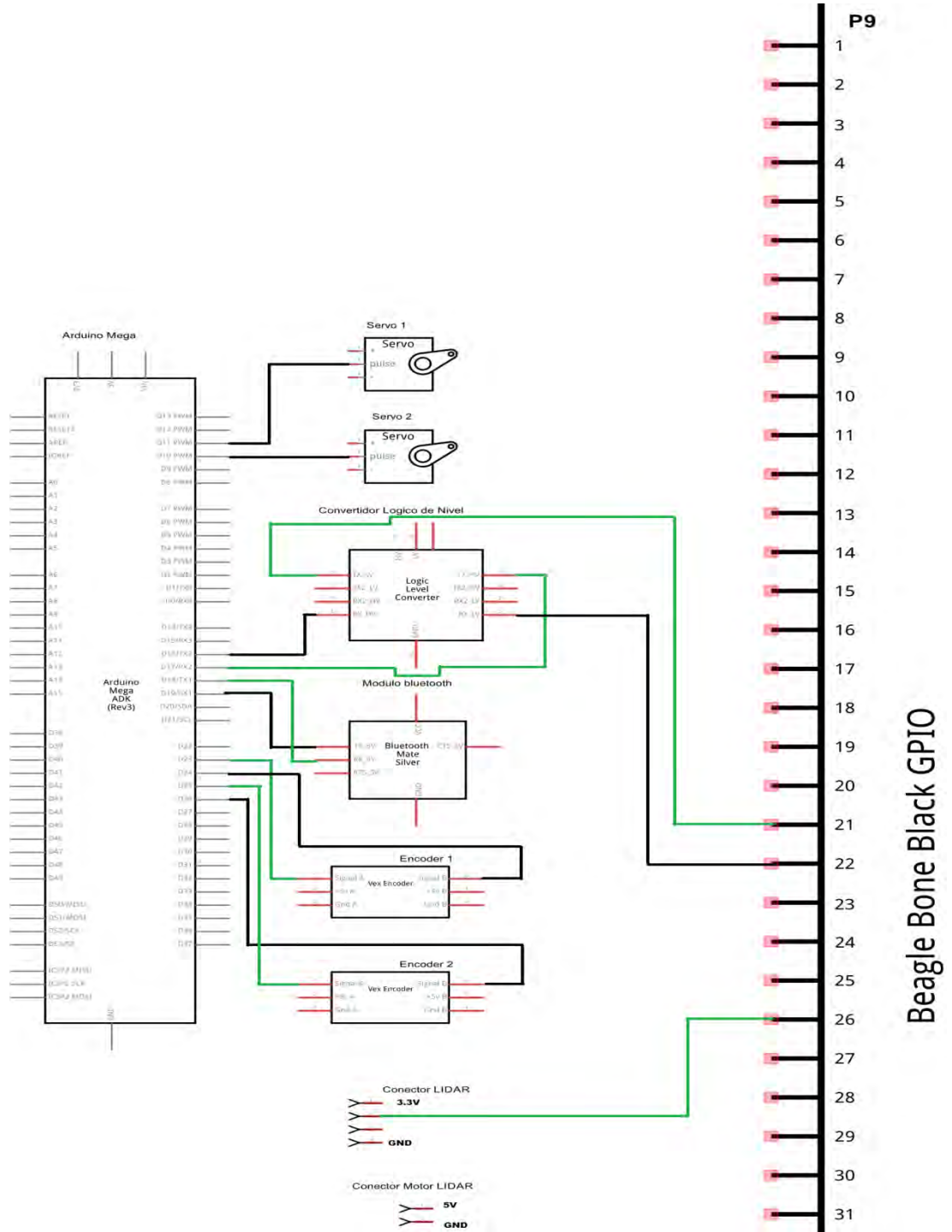
VEX. Vex Robotics. [en línea] 2014 [Consultado en Mayo de 2014]. Disponible en Internet: <http://www.vexrobotics.com/>

WIKIPEDIA. Arquitectura ARM. [en línea]. [consultado en septiembre de 2013]. Disponible en internet: http://es.wikipedia.org/wiki/Arquitectura_ARM

WIKIPEDIA. Robot móvil. [en línea]. [Consultado en agosto de 2013]. Disponible en internet: http://es.wikipedia.org/wiki/Robot_m%C3%B3vil

ANEXOS

Anexo A. Diagrama de conexión del hardware



En el diagrama anterior se muestra la conexión del hardware empleado en el sistema de control. Dispositivos como el router inalámbrico no aparecen en el este debido a que se considera como un componente externo al hardware. No obstante cabe resaltar que esta va conectado al puerto Ethernet de la tarjeta BeagleBone Black.

Nota: los encoders y motores se encuentran conectados a la alimentación de 5v proporcionada por el Arduino, el cual a su vez utiliza una alimentación que puede estar en el rango de 7v a 12V. El sensor LIDAR se encuentra conectado a la alimentación de 3.3v proporcionada por el Arduino. Por otra parte, la tarjeta beaglebone black utiliza una alimentación diferente a la del Arduino, la cual es de 5v. Las fuentes que alimentan tanto al Arduino como a la Beaglebone Black deben tener sus referencias a tierra unidas.