La Mecatrónica en México, Vol. 2, No. 3, páginas 99 - 108, Septiembre 2013.
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN en trámite,

Integración de Tecnologías para el Desarrollo de un Sistema de Robótica de Rescate

Galíndez-Olascoaga Laura-Isabel, Balderas-Hill Rafael, Delbouis-Fuentes Sergio, Herrera Katia, Sánchez-Hernández Carlos-Sebastián, Flores-García Erick, Aceves-López Alejandro y Carbajal-Fernández Cuauhtémoc

Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Estado de México
Carretera Lago de Guadalupe Km. 3.5, Atizapán de Zaragoza, Edo. De México.
A00967491@itesm.mx, A01166499@itesm.mx, A01165872@itesm.mx, aaceves@itesm.mx, carbajal@itesm.mx

Resumen

Un robot de rescate es un robot diseñado específicamente para coadyuvar a la búsqueda y auxilio en caso de un siniestro. El siguiente artículo describe las características del robot de rescate Kauil. Su principal atributo es su integración funcional mediante el uso del meta-sistema operativo ROS (Robot Operating System), utilizado para el control y programación de todas las interfaces y algoritmos. Se detalla el funcionamiento de cada uno de los módulos que lo componen, tanto mecánicos, como de hardware y software. El robot se caracteriza por contar con una excelente capacidad de movilidad en terrenos agrestes y por el uso eficiente de la energía necesaria para alimentar los elementos funcionales del sistema. A partir de todos los estudios y procesos implicados se consiguió un sistema completamente funcional para ambientes de desastre en su modalidad teleoperada.

1. Introducción

La construcción de un robot móvil enfocado a las tareas de rescate surge de la necesidad de auxiliar e identificar víctimas en una zona de desastre. La función del robot consiste en explorar la zona de desastre, identificar sobrevivientes y su estado, evaluar el peligro de la situación, realizar un mapa de la zona, ubicarse dentro del mismo y ser capaz de enviarle la información al operador para que se pueda tomar una acción más rápida y eficaz, disminuyendo así el número de pérdidas humanas. La estructura de este artículo comienza con la descripción de un sistema mecánico, seguido de la descripción del funcionamiento de la electrónica para su locomoción y la arquitectura de software implicada en el sistema, para terminar con la integración de todos los elementos.

2. Sistema mecánico

Para cubrir las necesidades de movilidad a través de una zona agreste y donde se requiere de versatilidad para la evasión de obstáculos, se diseñó una estructura mecánica a base de aluminio, con las especificaciones geométricas y de transmisión de potencia pertinentes para el escenario planteado. La principal característica es que el robot tiene una articulación pasiva (*flipper*), es decir, no motorizada ni controlable de forma remota, que funge como enlace de dos orugas por cada lado del robot para conseguir una mejor estabilidad y evitar el balanceo al subir escaleras y al moverse a través de superficies irregulares.

Se implementaron los *flippers* en cada de una de las tracciones de tal manera que fueran independientes y se pudiera tener una movilidad eficiente independientemente del lado en el que se atacaran los obstáculos, es decir, de frente, girando hacia el lado derecho o lado izquierdo.

Para el sistema de transmisión de potencia se diseñó un tren de engranes con una razón de transmisión de 8:1, la justificación fue que se planteó una pendiente de 45° como obstáculo de mayor

dificultad de movilidad, y mediante esta etapa de potencia se alcanzó el torque necesario para superarlo; los actuadores incorporados a este sistema mecánico son dos motores de corriente directa de 12 Volts a 10 Amperes acoplados a la entrada del tren de engranes definida mediante engranes de diente recto los cuales, como se muestra en la Fig. 1, están acoplados a un conjunto de dos engranes cónicos para cada una de las dos flechas de salida de transmisión [1].

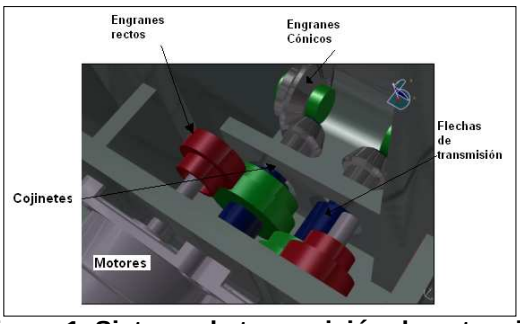

Figura 1. Sistema de transmisión de potencia

Se tienen dos catarinas impulsoras, acopladas a las flechas de transmisión, que proporcionan el avance a las cadenas. Éstas aseguran la tracción mediante perfiles de neopreno insertados a cada uno de sus eslabones, con la finalidad de subir y escalar pendientes aumentando la fricción, evitando así el deslizamiento como se muestra en la Figura 2 [2].

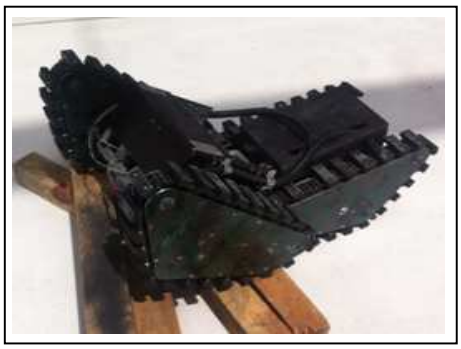

Figura 2. Tracción con perfiles de neopreno que permite aumentar la fricción al contacto con el suelo

3. Arquitectura de software

ROS [10] es un meta-sistema operativo que provee herramientas y servicios propios de un sistema operativo con el fin de facilitar el desarrollo de aplicaciones robóticas. De igual manera incluye tipos de datos específicos. La estructura de comunicación de ROS consiste en una red de procesos de tipo *peer-to-peer* en donde la comunicación se lleva a cabo mediante el transporte de mensajes por medio de tópicos.

4. Sistemas electrónicos

El control de los motores de corriente directa se logra con el uso de dos puentes H de alta potencia cuyo modelo es MD03. Para la transmisión de datos entre el procesador central y las tarjetas controladoras de motores se utiliza el bus I2C.

Los algoritmos de control de los motores se programan y ejecutan en el procesador central que en el caso presente es una computadora portátil. El procesador central se comunica inalámbricamente a una estación de teleoperación. En la Figura 3 se explica cómo se logra el

intercambio de información entre procesos, por lo cual se debe programar un publicador y un subscriptor. El subscriptor es un nodo que funciona de interfaz entre el procesador central y las tarjetas controladoras. Y el publicador es un nodo mediante el cual el usuario puede operar remotamente el movimiento del robot mediante el uso del teclado de su PC [3].

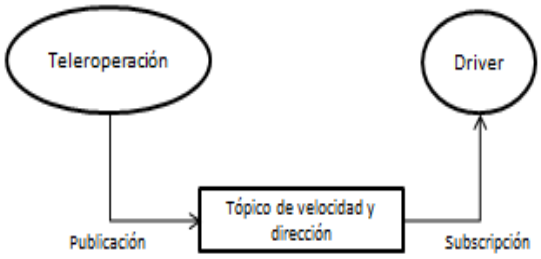

Figura 3. Interfaz de comunicación

El programa que funciona como driver envía cadena de hexadecimales que le indican a la interfaz I2C-USB y a las tarjetas controladoras qué dirección y velocidad se mandará a los motores. Enseguida se presentan la estructura de la cadena:

1. 0x57: Modo de operación I2C.
2. 0x01: Bit de inicio.
3. 0x32: Número de bytes de datos.
4. 0xB0/0xB2: Dirección del dispositivo.
5. 0x02/0x00: Registros de velocidad y dirección respectivamente.
6. Datos de velocidad (entre 0x00 y 0xFF) y datos de dirección (0x01 adelante y 0x02 atrás).
7. 0x03: Bit de paro.

En cuanto al suministro de energía de los motores de corriente directa de 12 Volts a 4 Amperes, se tienen dos baterías de polímero de litio de 3 celdas a 4000mAh conectadas en paralelo para cada uno de los actuadores. Es importante mencionar que se optó por utilizar este tipo de baterías debido a que permiten una mejor densidad de energía, logrando un funcionamiento más prolongado; la conexión en paralelo le brinda al sistema una mayor autonomía energética. Se implementó también un circuito con un diseño de filtros entre las baterías y los motores que sirviera como supresor de picos de corriente.

5. Odometría

El objetivo de la odometría es obtener la información sobre los cambios en la posición, velocidad y orientación del robot tomando en cuenta una referencia global. Cada par de orugas se encuentran acopladas a la misma flecha de transmisión correspondiente de cada tracción, por lo que para fines de análisis se aproxima el modelo cinemático del robot a un diferencial de dos ruedas como se muestra en la Figura 4.

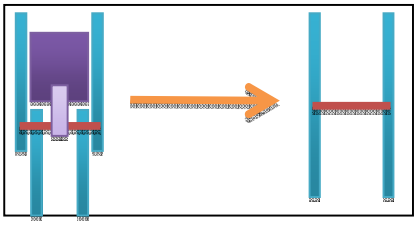

Figura 4. Aproximación a modelo cinemático diferencial.

Luego se define que se utiliza un marco de referencia global, en donde el robot conocerá siempre su posición en el plano *XY* con respecto al punto de inicio de todo su desplazamiento.

Para recolectar la información necesaria se utilizaron *encoders* de cuadratura acoplados a cada uno de los rotores del motor. Los datos entregados por los *encoders* son el número de pulsos de cada uno de ellos cada uno con un signo positivo o negativo dependiendo del sentido del desplazamiento. Posteriormente haciendo uso de la Ecuación 1 se obtiene la conversión de pulsos generados a metros desplazados por el móvil.[4]

$$s = \frac{2\pi r_p}{\eta C_e} P \tag{1}$$

Donde $s < m >$ es el desplazamiento lineal deseado, $r_p < m >$ es el radio de la catarina, η es la relación de transmisión de los trenes de engranes, $C_e < pulsos >$ es la resolución del encoder (cantidad de pulsos de una revolución) y $P < pulsos >$ es la cantidad de pulsos contados en un tiempo dado. En esta ecuación, $\frac{2\pi r_p}{\eta C_e}$ es conocido como $C_m < \frac{m}{pulsos} >$ (factor de conversión "pulso encoder /desplazamiento lineal"). Por tanto el cálculo se reduce a la ecuación 2.

$$s = C_m P = \gamma r \tag{2}$$

El cálculo de la ecuación 2 se realiza para ambas tracciones (S_L *y* S_R). A partir de las longitudes de arco se puede calcular la velocidad para cada una de las tracciones, descrito en la ecuación 3.

$$v = \frac{s}{dt} \tag{3}$$

El diferencial de tiempo *dt* se actualiza cada vez que el programa cicla usando la frecuencia del oscilador de la computadora.

Una vez planteados los dos parámetros iniciales se considera que el robot se encuentra principalmente bajo tres situaciones de movimiento:

1. Traslación lineal: Los dos motores están encendidos como se muestra en la Figura 5.

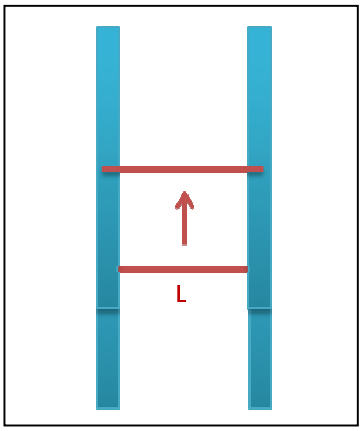

Figura 5. Traslación lineal del robot

En la ecuación 4 se describe cómo se obtiene la velocidad del robot en esta situación.

$$v = \frac{v_L + v_R}{2dt} \tag{4}$$

En ambas situaciones cuando el robot se encuentra haciendo un movimiento angular se utiliza la ecuación 5 para el cálculo de velocidad lineal

$$v = \frac{s_R - s_L}{dt} \tag{5}$$

2. Rotaciones en su propio centro: los dos motores se encuentran girando a la misma velocidad pero a direcciones contrarias como se muestra en la figura 6.

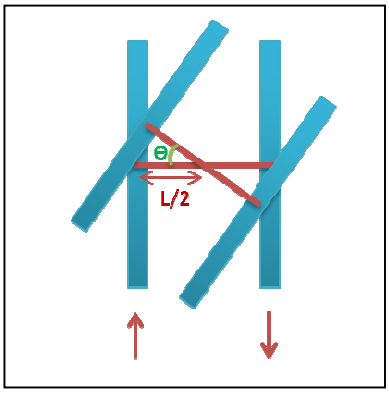

Figura 6. Rotación sobre su centro

Cuando el robot se encuentra rotando sobre su propio centro se utiliza la siguiente ecuación:

$$\theta = \frac{2(s_R - s_L)}{L} \tag{6}$$

Donde la longitud al eje sobre el cual gira el robot es igual a *L/2*

3. Rotaciones en un eje diferente al centro del robot: cuando sólo uno de los motores se encuentra encendido. Se muestra en la Figura 7.

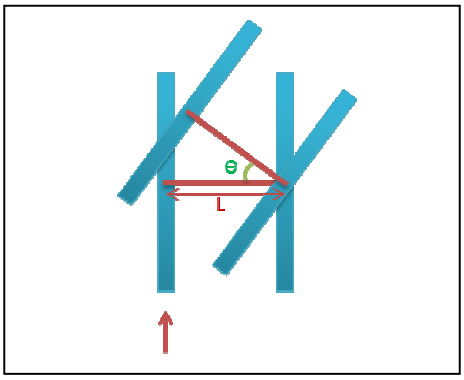

Figura 7. Rotación sobre un eje diferente

Para esta situación la orientación se calcula mediante la ecuación 7.

$$\theta = \frac{s_R - s_L}{L} \tag{7}$$

6. Transformaciones entre ejes coordenados

ROS incluye una serie de librerías que le permiten al usuario hacer las transformaciones de ejes coordenados.Las utilerías de ROS acomodan la organización de ejes coordenados en una estructura de árbol como se observa en la Figura 8.

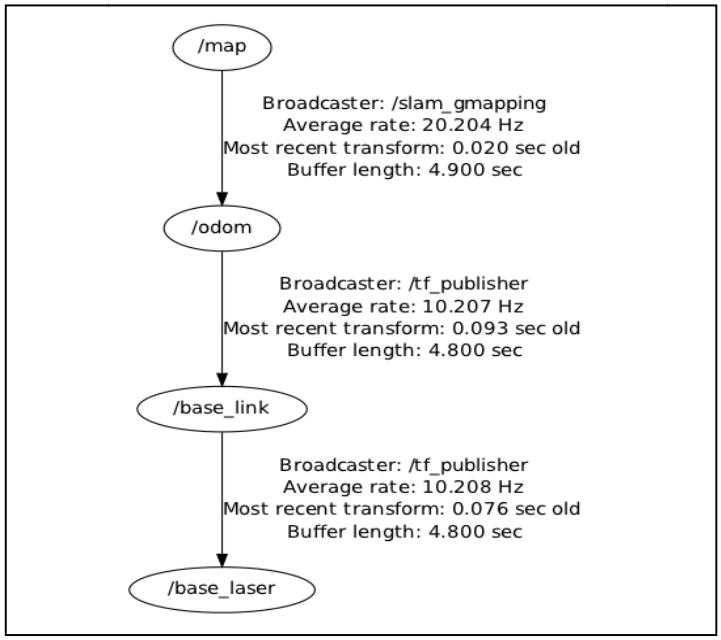

Figura 8. Árbol de transformadas del robot. Se observa que existen cuatro diferentes marcos de ejes coordenados en el sistema.

El primer marco denominado *base_laser* corresponde al punto en el cual se sitúa el láser. El segundo marco se ubica en el centro de masa del robot y se denomina *base_link*. La transformación que existe entre estos dos marcos consiste en una traslación como se muestra en la Figura 9.

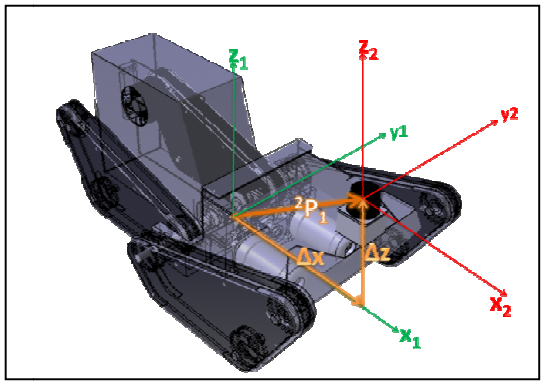

Figura 9. El diagrama muestra la traslación entre el punto donde se encuentra el centro de masa del robot y el punto donde se ubica el láser. El sistema de ejes coordenado (x_1,y_1,z_1) corresponde al marco *base_link* y el sistema (x_2,y_2,z_2) corresponde al marco *base_laser*.

La librería TF de ROS nos permite realizar rotaciones y traslaciones de puntos y vectores usando una sola función. Para la transformación de un punto *P* de */base_link* a */base_laser* se realiza la siguiente traslación detallada en la ecuación 8:

$$^{base\_laser}P = {}^{base\_link}P + {}^{2}P_1 \tag{8}$$

La siguiente transformación se realiza entre */base_link* y */odom* y tiene la función de indicarle al robot cuánto se ha movido de acuerdo a los cálculos de odometría. Cuando se realizan los cálculos de odometría, el resultado final son dos parámetros de traslación (*x,y*) y uno de rotación (*Θ),* más adelante se explicará la obtención de cada uno a detalle. La transformación se detalla en las ecuaciones 9, 10 y 11.

$$^{base\_link}P = {}^{base\_link}_{odom}R * {}^{odom}P + {}^{base\_link}P_{odom} \tag{9}$$

Donde

$$^{base\_link}_{odom}R = \begin{bmatrix} cos\theta & -sen\theta & 0 \\ sen\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

$$^{base\_link}P_{odom} = [x\ y\ 0] \tag{11}$$

Finalmente, la transformación de *map* a *odom,* detallada en el algoritmo incluido dentro del paquete slam_gmapping de ROS

7. Mapeo

Grisetti, Stachnis y Burgard [5] [6] proponen un filtro de partículas de Rao Blackwell modificado como solución al problema de SLAM(Simultaneous Localization and Mapping)[7][8][9].

La meta principal de un filtro de partículas aplicado al mapeo [10] es la de encontrar el *posterior* del mapa *m*, dada una trayectoria $x_{1:t}$, contando con las observaciones provenientes del láser Hokuyo $z_{1:t}$ y tomando en cuenta las mediciones de Odometría $u_{1:t-1}$ como muestra la Ecuación 11.

$$(x_{1:t},\ |z_{1:,1:t-1}) = (m\ |x_{1:,1:t}) \cdot (x_{1:t}|z_{1:,1:t-1}) \tag{11}$$

A continuación se detalla el algoritmo [9] completo:

1. Se obtiene la estimación de la pose inicial $x_t(i)$ a partir de la pose pasada usando el resultado de la Odometría u_{t-1}.
2. Se corre un algoritmo para determinar en qué áreas se encontrarán los máximos de las lecturas del láser.
3. Los puntos de muestreo son seleccionados dentro de esta área y usándolos se calculan las matrices de media y covarianza al evaluar las posiciones muestreadas como se observa en la Ecuación 12.

$$(z_t|m_{t-1(i)},x_j)(x_j|x_{t-1(i)},u_{t-1}) \tag{12}$$

4. A partir de lo obtenido en el punto 3, se realiza un nuevo muestreo usando la aproximación Gaussiana resultante $(\eta_{(i)},\Sigma_{t(i)})$.
5. Se calculan los pesos de las partículas y si se observa que existirá depleción de las mismas se vuelve a hacer un muestreo.
6. Por último se calcula el mapa *m* para cada partícula $(m\ |x_{1:\ ,1:t})$.

En el caso particular se debe realizar una integración del algoritmo descrito con los algoritmos de odometría y transformación de ejes coordenados creados para la plataforma mecánica disponible. Como se mencionó, el algoritmo del filtro de partículas requiere lecturas del láser y datos de odometría. Las lecturas del láser se mandan directamente al algoritmo de mapeo denominado GMapping [5] mientras que los de Odometría son enviados por medio del algoritmo de transformación de ejes coordenados descrito en la sección 6. La Figura 10 muestra como se realiza dicha integración a nivel software por medio del uso de ROS.

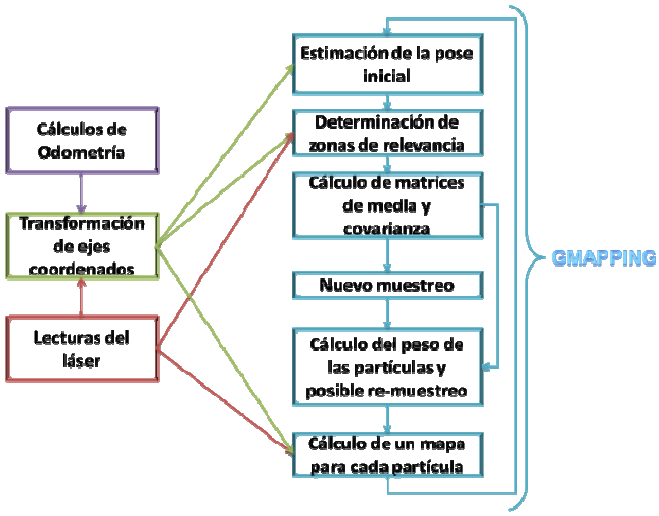

Figura 10. Integración de diferentes algoritmos con el objetivo final de generar mapas. ROS es la herramienta que permite dicha integración.

En la Figura 11 se muestra un mapa generado mediante la implementación del algoritmo detallado en las secciones anteriores

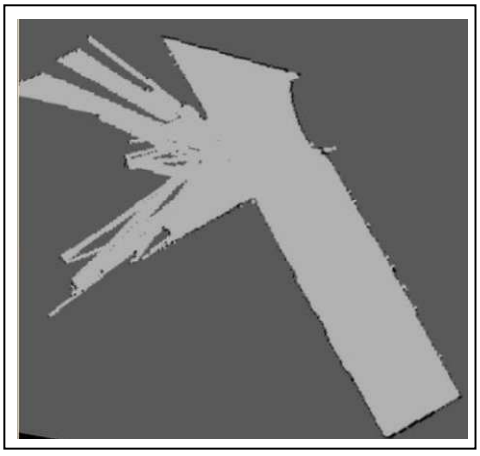

Figura 11. Mapa

8. Resultados de Integración

El resultado obtenido a partir de la integración de las tecnologías descritas es un robot capaz de mapear una zona de desastre previamente desconocida. Para dicho objetivo el procesamiento de tres tipos de datos es necesario: odometría, lecturas del láser y transformación de ejes coordenados. La odometría se debe ajustar a las especificaciones de la estructura mecánica con la que se cuenta, la cual consta de una doble tracción que permite tener una movilidad adecuada para terrenos agrestes y de difícil acceso. El procesamiento de las lecturas del láser al igual que las transformaciones de ejes coordenados es posible gracias a la arquitectura de software que ROS proporciona. El control del robot es teleoperado para lo cual se envían mensajes vía *WiFi* entre el procesador central y una estación remota. Es importante mencionar que ROS tiene una gran relevancia ya que es la plataforma que proporciona la capacidad de integración entre todos los elementos involucrados como muestra la Figura 12.

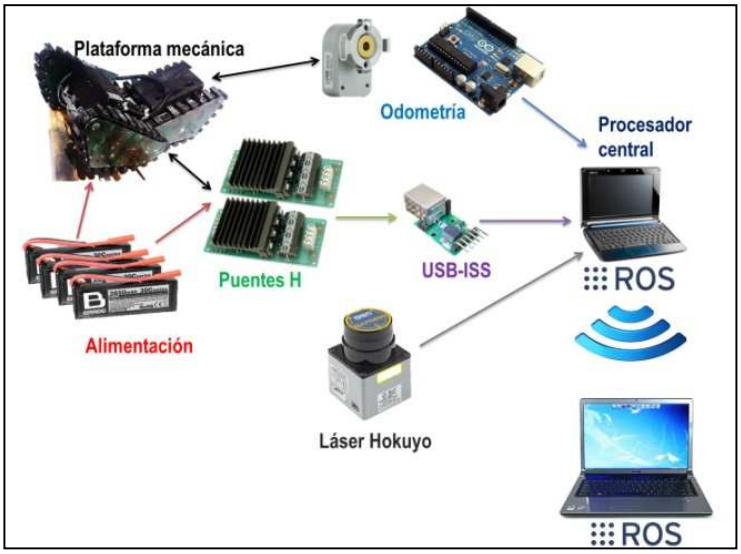

Figura 12. Integración de los elementos

9. Conclusión

El diseño de un robot es una labor compleja que consiste en la integración de diferentes tecnologías y conocimientos. Se concluye que el sistema cumple con los requerimientos de una aplicación de rescate, ya que se cuenta con una estructura mecánica robusta con la cual se puede acceder a terrenos complejos sin poner en riesgo al operador, además de que se tiene la capacidad de hacer mapas de zonas inicialmente desconocidas.

Agradecimientos

Agradecemos el apoyo recibido por el Conacyt a través de la Red Mexicana de Robótica así como del apoyo académico recibido por el ITESM.

Referencias

[1] Nieto-Granda C., Galindez-Olascoaga D., Chavez-Muñoz D., Camargo-Rosas C., Arriaga-Orta A. RoboCup Rescue 2008 . Robot League Team - Tec Rams – TMR Verlab. Para 2008 Robocup Rescue. Suzhou, China.

[2] Nieto-Granda C., Chavez-Muñoz D., Galindez-Olascoaga D., Camargo-Rosas C., Arriaga-Orta A., Pineda-Olivares A., Hayet J.B., Arachavaleta G., Rodríguez-*Salazar L. TMR VERLAB - Robot Kauil. Robot Challenge Workshop. International Conference on Robotics and Automation (ICRA 2008).* May 19-23, 2008. Pasadena, California. USA.

[3] Galindez-Olascoaga L., Aceves-Lopez A., Nieto-Granda C., Chávez-Muñoz D., Galíndez-Olascoaga D., RoboCup Rescue 2012 . Robot League Team – Kauil-TecMTY. Para 2012 Robocup Rescue. Mexico City, Mexico.

[4] Choset H., Lynch K., Hutchinson S., Kantor G., Burgard W., Kavraki L., Thrun S. *"Principles of Robot Motion. Theory, Algorithms and Implementations".* The MIT Press, 603, 2005.

[5] Grisetti G., Stachniss C., Burgard W.*" Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters, IEEE Transactions on Robotics"*, 2006

[6] K. Conley (2012). ROS Introduction. [En línea]. Disponible: http://www.ros.org/wiki/ROS/Introduction

[7] H. Durrant-White, T. Bailey. Simultaneous Localization and Mapping: part I. IEEE Robotics & Automation Magazine.Pages 99-110.2006.

[8] J.J. O'Reilly. SLAM II tutorial. Simulation Conference, 1990. Proceedings.Winter,1990.

[9] Choset H., Lynch K., Hutchinson S., Kantor G., Burgard W., Kavraki L., Thrun S. Principles of Robot Motion. Theory, Algorithms and Implementations.The MIT Press,Page 294, 2005.

[10] A. Doucet, N. de Freitas, and N. Gordan, editors. Sequential Monte- Carlo Methods in Practice. Springer Verlag, 2001.