Universidad de Almería

Escuela Superior de Ingeniería

Grado en Ingeniería Electrónica Industrial

Navegación autónoma de un robot móvil SUMMIT

Autor:
Enrique Rodríguez Miranda

Tutores:
José Luis Guzmán Sánchez
José Carlos Moreno Ubeda

Septiembre 1, 2015

Agradecimientos

A mis tutores José Luis Guzmán y José Carlos Moreno por iniciarme en el mundo de la robótica y el control automático y guiarme en el presente proyecto.

A José Luis Blanco por su ayuda incondicional en los momentos más difíciles de este trabajo. Sin él, no hubiera sido posible cumplir todos los objetivos.

Índice

Índice de figuras

Índice de tablas

Resumen

El propósito de este proyecto es utilizar un robot móvil terrestre con configuración Ackerman para desarrollar mapas de terreno e implementar distintos métodos de navegación autónoma. Para ello, en el trabajo se utiliza un sensor lidar para la creación de mapas 2D para la implementación de distintos métodos de navegación en ROS, con el fin de realizar una comparativa entre ellos.

Palabras clave: Robot móvil, ROS, SLAM y navegación autónoma.

Abstract

The purpose of this project is to use a mobile robot with Ackerman steering to develop environment grid maps and implement different methods of autonomous navigation. To this end, a lidar sensor is used to create 2D maps for implementing various methods of navigation in ROS, in order to make a comparison between them.

Key words: Mobile robot, ROS, SLAM and autonomous navigation.

Capítulo 1: Interés y objetivos

1.1 Interés

Con el progreso de la investigación en robótica móvil, están siendo explorados muchos aspectos de la navegación autónoma de robots.

La realización de este trabajo se lleva a cabo para probar las nuevas tecnologías referentes a los robots móviles, como son el Sistema Operativo de Robots (ROS) [22], técnicas de mapeado y métodos de navegación autónoma.

Gracias al robot móvil Summit [21], se van a llevar a cabo una serie de actividades de control remoto para poder aplicar las técnicas actuales de SLAM (*Simultaneous Localization And Mapping*) y desarrollar un mapa preciso de la zona de trabajo en el edificio CITE IV de la Universidad de Almería. Asimismo, una vez establecido un mapa de la zona se podrán realizar pruebas referentes a técnicas de navegación autónoma, con planificación de ruta, detección de obstáculos y control de velocidad.

Es claro que desde la creación de la robótica móvil autónoma hace menos de dos décadas, se han hecho avances significativos en un número importante de temas, especialmente comprendido en los aspectos de las aspiraciones biológicas, el uso de comunicación en grupos de multi-robots y el diseño de arquitecturas de control de multi-robots. También se ha progresado considerablemente en localización, mapeo, exploración, transporte cooperativo de objetivos y coordinación de movimientos.

1.2 Contexto

La definición más importante en cuanto al tema de este proyecto es el concepto de "Robótica". Si se recurre a la Wikipedia en busca de una definición para este término, se encuentra que su significado hace referencia a la ciencia y la tecnología de los robots. La cosa se complica ya que definir lo que es un robot, no es nada fácil. Se pueden encontrar multitud de definiciones, sin embargo casi ninguna de ellas logra ser perfecta.

Por ello, en este informe, un robot será aquella entidad electromecánica que tenga capacidad de percibir su entorno y actuar sobre él con un propósito definido.

Por buena que parezca, esta definición deja a un lado casi todos los robots industriales que se emplean en la actualidad en la producción en masa de productos, como por ejemplo la fabricación de coches.

Estos robots actúan sobre su entorno con un propósito definido, pero no perciben el entorno para ello. Están programados de tal forma que siempre ejecutan los mismos movimientos, independientemente de lo que suceda en su entorno. Es decir, si en vez de un coche se pone una bicicleta, el robot repetirá su ciclo de movimientos, generando un auténtico desastre.

Sin embargo, para el tipo de robótica que atañe a este proyecto, esta definición es ideal, ya que, los robots necesitarán percibir su entorno, interpretarlo, razonar y planificar sobre lo que sucede y decidir cuáles son las acciones adecuadas para cumplir con sus tareas. Necesitarán también interactuar con los humanos de diversas formas: reconociendo sus caras, entendiendo lo que dicen, interpretando sus movimientos y otro sinfín de formas que hasta ahora eran impensables. Este nuevo ámbito de funcionalidades se enmarca en la Robótica Autónoma.

La Robótica Autónoma [15] es el área de la Robótica que desarrolla robots capaces de desplazarse y actuar sin intervención humana. Para ello el robot debe percibir su entorno y actuar en consecuencia, además de llevar a cabo su tarea.

La construcción experimental de pequeños robots móviles en laboratorios universitarios y a nivel de aficionados (microbots) está haciendo surgir un tipo de investigación que aborda los aspectos de conexión senso-motora (el aspecto más fundamental de la Robótica) desde un punto de vista diferente a aproximaciones anteriores.

A la hora de integrar los robots móviles autónomos en nuestra sociedad, el desafío es conseguir que estén presentes de forma natural en el interior de los edificios, con el fin de que puedan realizar las tareas como guiar visitantes, repartir medicinas en hospitales, monitorizar la seguridad de un edificio, asistir a discapacitados o incluso actividades de mayor complejidad. Este tipo de actividades se encuentran no sólo dentro de la robótica móvil sino también en el campo de la inteligencia ambiental, cuyo fin último es el desarrollo de sistemas inteligentes integrados de manera transparente en el entorno y con la capacidad de asistir al ser humano.

Otra de las principales actividades realizadas por los robots autónomos es la exploración, que puede definirse como el proceso mediante el cual un robot móvil recorre el entorno, detectando lugares de referencia y transiciones entre ellos sin otro objetivo que construir de forma autónoma un modelo del entorno lo más completo posible.

Al contrario de la actividad anterior, los robots móviles también pueden hacer uso de mapas de terreno preestablecidos y llevar a cabo diversas tareas en puntos concretos del mismo, como pueden ser el seguimiento de trayectorias, ya sea con puntos de control o de rutas continuas, búsqueda de obstáculos, autoposicionamiento y analizar un terreno conocido parcialmente.

1.3 Objetivos

Considerando lo descrito anteriormente, el principal objetivo de este proyecto es la navegación autónoma del robot móvil Summit, así como el seguimiento de trayectorias establecidas.

Los objetivos se pueden establecer de la siguiente manera:

- Seguimiento de trayectorias establecidas.
- Detección y evasión de obstáculos.
- Desarrollo de mapas de terreno a partir de visión artificial.
- Navegación autónoma a partir de mapas de terreno preestablecidos.

1.4 Resumen del sistema desarrollado

El trabajo que se va a realizar se centra en la robótica móvil, empleando para su desarrollo la plataforma Summit (figura 1.1), un robot de mediano tamaño de alta movilidad y bajo coste, pensado tanto para exteriores como para interiores.

La mecánica es equivalente a la de un coche 4x4 y utiliza un chasis de alta calidad fabricado en aluminio. El robot puede navegar de forma autónoma o ser teleoperado ya que cuenta con una cámara PTZ para la transmisión de video en tiempo real y sensorización avanzada (sensor lidar, RTK-DGPS, unidad inercial, etc. . .).

La arquitectura de control es abierta y modular, basada en ROS [1], lo que permite la programación y simulación de algoritmos offline y lo hace extremadamente útil para su utilización en docencia e investigación.

Figura 1.1: Plataforma móvil Summit

El Sistema Operativo de Robots (ROS) [22] es un entorno de código abierto para el desarrollo de software para robots que provee la funcionalidad de un sistema operativo en un clúster heterogéneo.

Para este trabajo se desarrollará finalmente un ensayo comparativo entre dos métodos de navegación autónoma, navegación planificada frente a navegación reactiva, para comprobar las ventajas y desventajas que ofrece cada método en un robot móvil con configuración Ackerman.

1.5 Planificación temporal

Periodo de tiempo	Actividad realizada	Duración
Mayo (2014)	- Redacción del anteproyecto.	5 días
Junio - Julio (2014)	- Recogida de información sobre ROS.	10 días
Julio - Agosto (2014)	- Estudio de la información recogida sobre ROS. - Toma de contacto con el robot móvil Summit.	20 días
Agosto (2014)	- Estudio de la información relacionada sobre el robot. - Revisión de los componentes del robot y su funcionamiento.	10 días
	- Primeras pruebas con el robot teleoperado.	5 días
Septiembre (2014)	- Revisión del software del robot. - Solución de diversos problemas relacionados con ROS.	20 días
Septiembre - Octubre (2014)	- Primeras pruebas con los nodos del robot en simulación. - Cambio de software a ROS Hydro.	10 días
Octubre (2014)	- Adaptación de los componentes al nuevo software. - Solución de errores. - Nuevas pruebas de teleoperación en simulación.	20 días
Octubre - Noviembre (2014)	- Revisión de información referente a robot móviles.	5 días
Noviembre (2014)	- Redacción de la primera parte de la memoria. - Pruebas en simulación de navegación autónoma Pure Pursuit.	10 días
Noviembre - Diciembre (2014)	- Inclusión y prueba de un sensor lidar en el robot.	10 días
Enero - Febrero (215)	- Pruebas de mapeado de entornos interiores. - Pruebas con el método de navegación autónoma Pure Pursuit.	7 días
Marzo (2015)	- Primeros intentos de navegación autónoma planificada. - Solución de diversos errores relacionados a la navegación. - Estudio de los diversos tipos de navegación autónoma.	15 días
Abril (2015)	- Solución de diversos problemas relacionados con los métodos de navegación autónoma de ROS. - Pruebas de navegación autónoma planificada. - Mapeado de diversos entornos interiores. - Instalación de un nuevo método de navegación reactiva.	22 días
Mayo (2015)	- Pruebas con un método de navegación reactiva (MRPT). - Adaptación de componentes al software del robot. - Creación y mapeado de un circuito de obstáculos para realizar un ensayo de comparación de sistemas de navegación.	10 días
Junio (2015)	- Ensayo comparativo con diversos métodos de navegación. - Redacción de un artículo sobre navegación autónoma.	22 días
Agosto - Septiembre (2015)	- Revisión de la memoria. - Finalización de la memoria.	10 días

Tabla 1.1: Planificación temporal

En base a la duración de las actividades realizadas, se estima que el tiempo de realización del proyecto ha sido de 216 días, es decir, 7 meses aproximadamente. En cada día se han trabajado unas 4 horas y media, por lo que, la duración final del proyecto ha sido de 972 horas.

1.6 Estructura de la memoria del proyecto

El capítulo 1 inicia el proyecto mostrando una introducción sobre el tema que se va a tratar y los objetivos que se plantean cumplir.

En el capítulo 2 se recoge toda la información estudiada referente al tema del proyecto.

El capítulo 3 se presenta los materiales que se han utilizado para el desarrollo del proyecto y los métodos aplicados.

Los resultados que se han obtenido en las diferentes pruebas y ensayos realizados son expuestos en el capítulo 4.

El capítulo 5 expone las conclusiones referentes a los resultados obtenidos en el capítulo anterior y muestra posibles líneas de trabajos futuros relacionados con en el tema tratado.

Todos los problemas que han surgido en el desarrollo de este proyecto se muestran en el capítulo 6.

En el capítulo 7 se presenta la bibliografía de los documentos o páginas webs de donde se ha obtenido la información para realizar este proyecto.

Como capítulo final se presentan los anexos, que contienen información adicional o complementaria al proyecto, como la nomenclatura.

Capítulo 2: Revisión bibliográfica

2.1 Evolución de la robótica móvil

Cada vez más a menudo se habla de robots autónomos o de la importancia creciente que está adquiriendo la investigación sobre los robots móviles en los últimos años.

La Robótica ha evolucionado hacia los sistemas autónomos, que son aquellos que son capaces de desenvolverse por sí mismos en entornos desconocidos y parcialmente cambiantes sin necesidad de supervisión.

El primer robot móvil de la historia [17], pese a sus limitadas capacidades, fue ELSIE (Electro Light Sensitive Internal-External) (figura 2.1), construido en Inglaterra en 1953 por William Grey Walter, uno de los padres de la robótica moderna. ELSIE se limitaba a seguir una fuente de luz utilizando un sistema mecánico realimentado sin incorporar inteligencia adicional.

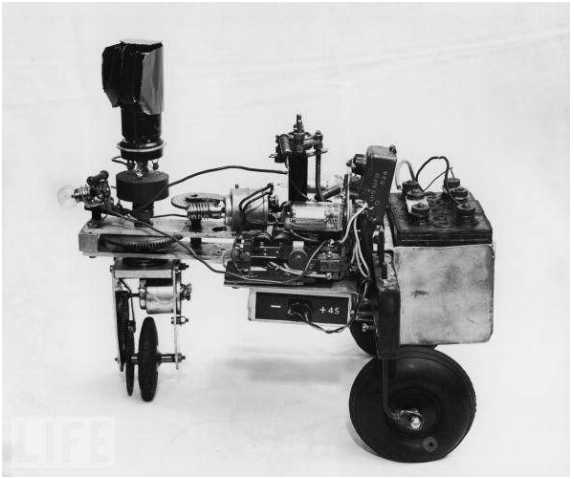

Figura 2.1: Robot móvil ELSIE

En 1968 apareció SHAKEY (figura 2.2) del SRI (Standford Research Institute), que estaba provisto de una diversidad de sensores así como una cámara de visión y sensores táctiles, además de poder desplazarse por el suelo. El proceso se llevaba en dos computadores conectados por radio, uno a bordo encargado de controlar los motores y otro remoto para el procesamiento de imágenes. Mientras que a otros robots debían enviarles las tareas paso por paso para realizar un objetivo, Shakey podía analizar por sí mismo el comando y dividirlo en tareas básicas.

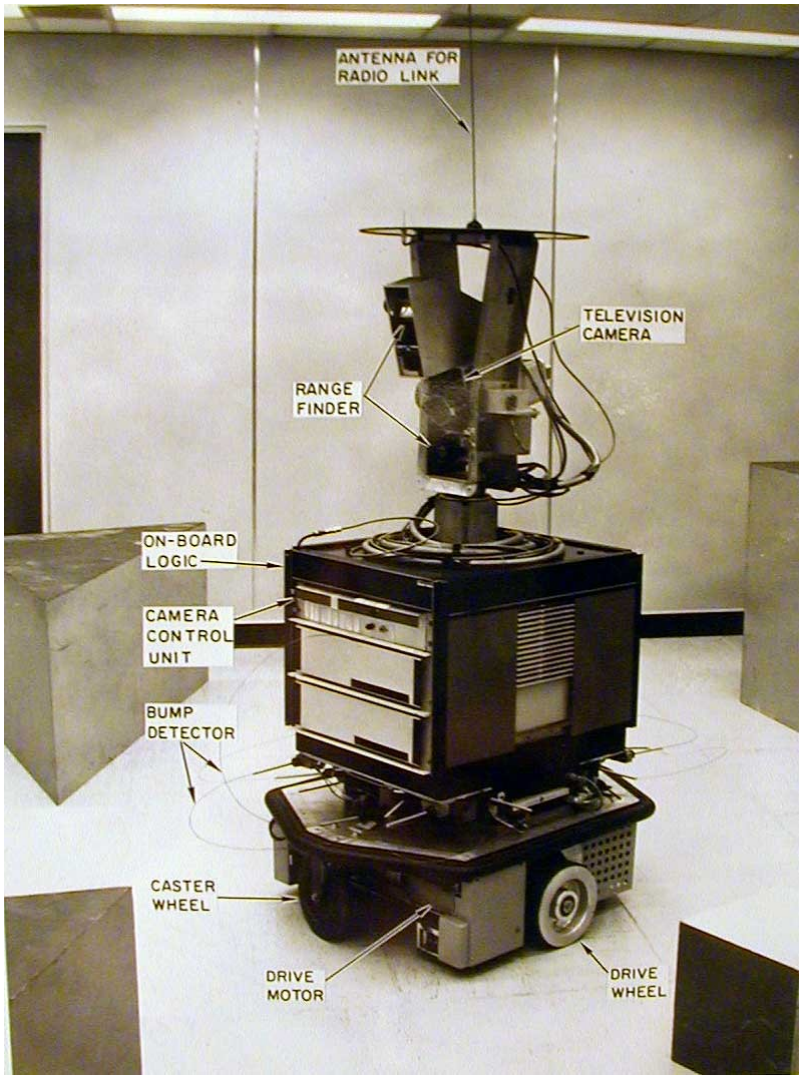

Figura 2.2: Robot móvil SHAKEY

En los setenta, la NASA inició un programa de cooperación con el Jet Propulsion Laboratory para desarrollar plataformas capaces de explorar terrenos hostiles.

El primer fruto de esta alianza sería el MARS-ROVER, que estaba equipado con un brazo mecánico tipo STANFORD, un dispositivo telemétrico láser, cámaras estéreo y sensores de proximidad.

Al comienzo del proyecto se enviaron los rovers SPIRIT y OPPORTUNITY, que realizaban exploraciones de la superficie de Marte y mediciones de las condiciones atmosféricas, las cuales, han servido para confirmar la teoría de la enorme cantidad de agua que existió en Marte y que existe en forma de hielo actualmente.

Desde entonces, la NASA ha seguido con el proyecto y en 2011 fue lanzada la Mars Science Laboratory, conocida como CURIOSITY (figura 2.3). Este vehículo astromóvil de exploración es tres veces más pesado y dos veces más grande que sus anteriores hermanos, equipado con instrumentos desarrollados en los últimos años.

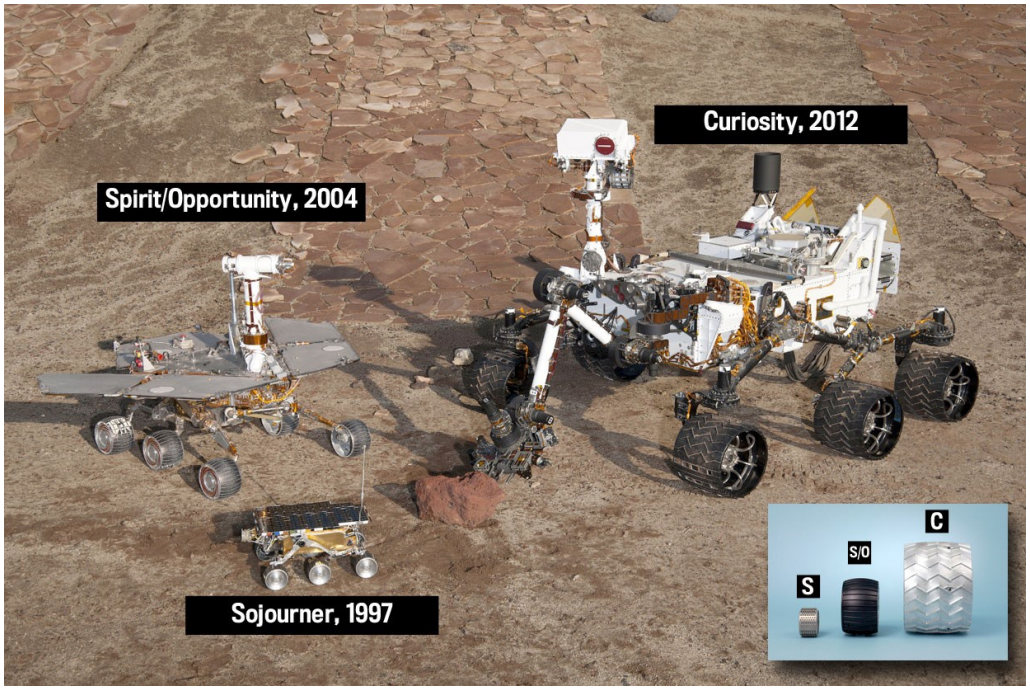

Figura 2.3: Evolución de los rovers de la NASA

En los ochenta aparece CART del SRI que trabaja con procesador de imagen estéreo, más una cámara adicional acoplada en su parte superior. También en la década de los ochenta, el CMU-ROVER (figura 2.4) de la Universidad Carnegie Mellon incorpora por primera vez una rueda timón, lo que permite cualquier posición de orientación del plano.

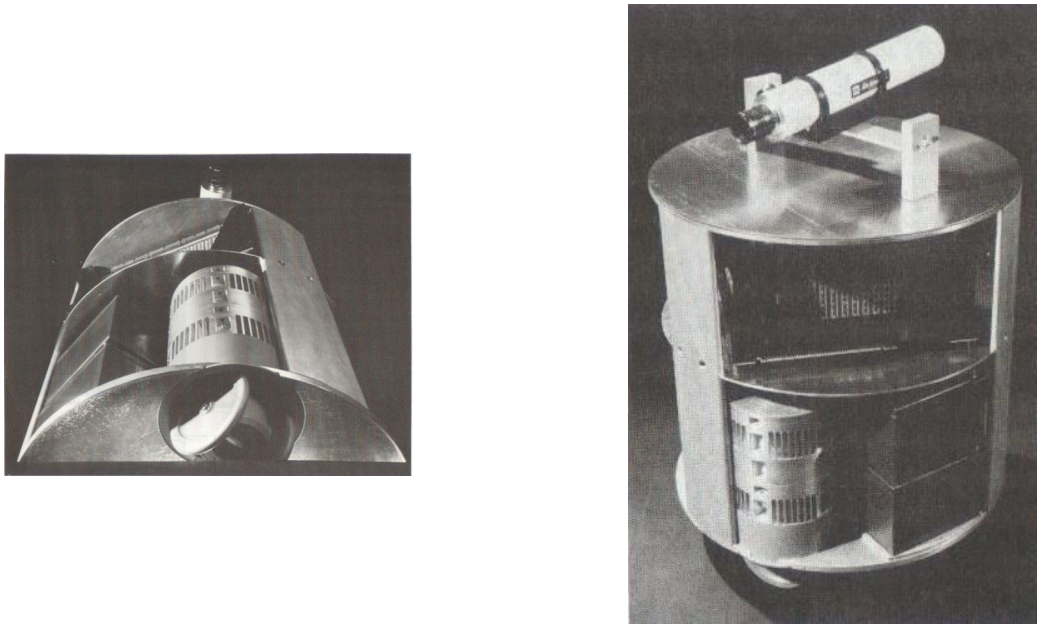

Figura 2.4: Robot móvil CMU-ROVER

En la actualidad, la robótica se debate entre modelos sumamente ambiciosos, como es el caso del IT, diseñado para expresar emociones, el COG, también conocido como el robot de cuatro sentidos, el LUNAR ROVER, vehículo de turismo con control remoto, y otros mucho más específicos como el CYPHER, un UAV de uso militar (figura 2.5).

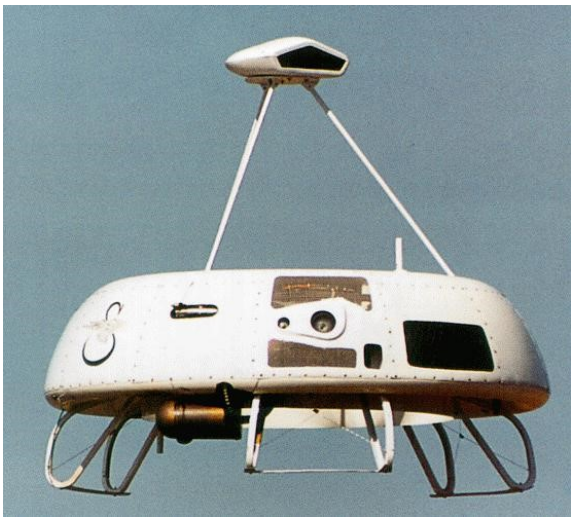

Figura 2.5: UAV CYPHER

En el campo de los robots antropomorfos se debe mencionar el P3 de Honda© (figura 2.6), que mide 1.60 m, pesa 130 Kg y es capaz de subir y bajar escaleras, abrir puertas, pulsar interruptores o empujar vehículos, entre otras muchas funciones.

Figura 2.6: Evolución de los robots de Honda©

Cabe destacar el último robot desarrollado por Honda©, ASIMO (figura 2.7) , el precursor del robot androide P3. Este modelo, a pesar de tener un tamaño reducido, cuenta con una movilidad superior al modelo anterior, así como un peso menor y una mayor capacidad de análisis de su entorno, pudiendo realizar tareas como jugar al fútbol, estrechar la mano, bailar o incluso dirigir una orquesta o tocar el violín.

Figura 2.7: Robot humanoide ASIMO

Sin embargo, uno de los pioneros para el entretenimiento basado en formas de animales, es el robot mascota de Sony© llamado Aibo (figura 2.8).

Dotado de una cierta capacidad de aprendizaje y comportamiento autónomo. Fue desarrollado por el laboratorio D-21 de Sony© y mostrado al público en junio de 1999, vendiendo 5000 ejemplares en internet en solo cuatro días.

Figura 2.8: Robot Aibo de Sony©

Otro de los hitos más recientes de la robótica de introducción masiva a los hogares de todo el mundo es la aspiradora autónoma Roomba (figura 2.9), la cual salió al mercado en el año 2002 y fue desarrollada por la empresa irobot©.

Figura 2.9: Robot aspiradora Roomba de irobot

La evolución de los robots industriales y de servicio ha sido vertiginosa. En poco más de 40 años las investigaciones y desarrollos sobre robótica han permitido que los robots tomen posiciones en casi todas las áreas productivas y tipos de industria, así como en los hogares. Los robots pueden sustituir al hombre en aquellas tareas repetitivas y hostiles, adaptándose inmediatamente a los cambios de producción solicitados por la demanda variable.

El futuro de la robótica apunta hacia el aumento de la movilidad, destreza y autonomía de los robots, así como a mantener una elevada interacción con los humanos.

El diseño y construcción de un robot móvil puede llevar diferentes soluciones, escogiendo la que mejor se adapte a las necesidades y aplicaciones para las cuales será utilizado el robot. En el mercado de robots, se encuentran desde los que se desplazan por medio de ruedas, hasta los que tienen una morfología muy parecida a los humanos; los cuales son conocidos como antropomorfos o humanoides.

En los últimos años, el desarrollo e investigación sobre robots móviles ha ido en aumento, debido a una nueva necesidad de concepción industrial que es conocida como planta de fabricación flexible, la cual requiere la reconfiguración de la secuencia de acciones necesarias para una producción variada, lo que a su vez exige una necesidad de desplazamiento por parte de los robots, ya que anteriormente el robot se encontraba en una célula de trabajo en la que a su alrededor se localizaban los distintos dispositivos con los que trabajaba.

Con el propósito de aumentar la movilidad del robot y ampliar su espacio de trabajo, se dispone de la robótica móvil [23].

El robot hace uso de un sistema locomotor para moverse libremente en cualquier dirección para tener mayor independencia y autonomía, al no verse obligado a que los dispositivos y herramientas que necesite se encuentren dentro de su zona de trabajo, la cual sería reducida.

En un entorno industrial tradicional, una solución viable, cuando ya se encuentra dispuesta toda la planta de producción, es que el robot móvil lleve los materiales o dispositivos que sean necesarios a los robots que no cuentan con la capacidad de poder desplazarse.

La robótica ofrece una gran gama de soluciones en cuanto a lo referente a robots móviles, encontrándose entre ellas los vehículos autoguiados denominados usualmente AGV (Autonomous Guided Vehicles), los cuales son dirigidos mediante sistemas externos preprogramados, tales como seguir un rail de cables eléctricos que generan un campo magnético, perseguir un camino definido o recrear un mapa mediante balizas posicionadas en el espacio donde se desea que el robot trabaje.

Conociendo todas las posibles aplicaciones de los robots móviles, que van desde la realización de tareas en la industria hasta llegar a los servicios proporcionados en la vida diaria a las personas, se vislumbra que éstos llegaron para quedarse, y

apenas comienza el ascenso para su introducción en muchas más actividades tanto en la industria como en la vida cotidiana del ser humano.

En general la historia de la Robótica se puede clasificar en cinco generaciones: las dos primeras, ya alcanzadas en los ochenta, incluían la gestión de tareas repetitivas con autonomía muy limitada. La tercera generación tuvo como extra la visión artificial, en lo que se avanzó mucho en los años ochenta y noventa y hoy día sigue siendo una tecnología con un gran auge en desarrollo e investigación. La cuarta generación incluye movilidad avanzada en exteriores e interiores y la quinta entraría en el dominio de la inteligencia artificial, tema que se está tratando en la actualidad.

2.2 Robótica móvil

2.2.1 Introducción

La robótica móvil [23] ha evolucionado conjuntamente con los avances tecnológicos recientes, lo cual ha dado lugar al desarrollo de muchas aplicaciones en áreas muy diversas. Si bien, dicha evolución ha alcanzado un nivel elevado en la actualidad, aún hay que esperar para la llegada de la revolución social de la robótica; donde los robots llegarían a ser un elemento más en el hogar, la empresa y la sociedad en general.

Los robots hoy en día son en su mayoría excesivamente caros y su función está muy especializada, por lo que, los estudios actuales se centran en conseguir que sean más generales y económicos, de tal manera que sean accesibles a todas las personas que quieran resolver algún problema en sus tareas cotidianas.

Debido a los avances tecnológicos actuales, construir un robot es posible y lo vienen realizando Universidades y actualmente en compañías de juguetes. Además, construir un pequeño robot se ha convertido en el pasatiempo de muchos jóvenes, motivados por los concursos que se realizan en muchos lugares alrededor del mundo.

La industria como tal no es la única que se ha beneficiado con la aparición de los robots, ya que existen áreas como: la medicina, aeronáutica y agricultura que también se han beneficiado. Una de las aplicaciones fuera de la industria, es el robot móvil todo terreno Soujorner, más conocido como Pathfinder, el cual recorrió la superficie de Marte el 5 de julio de 1997.

Los dispositivos y mecanismos que pueden agruparse bajo el concepto de Robots son muy diversos y, por lo tanto, es difícil establecer una clasificación coherente de los mismos que resista un análisis crítico y riguroso [16]

La Asociación de Robots Japonesa (JIRA) ha clasificado a los robots dentro de seis clases en base a su nivel de inteligencia:

1. Dispositivos de manejo manual, controlador por una persona.
2. Robots de secuencia arreglada.
3. Robots de secuencia variable, donde un operador puede modificar la secuencia fácilmente.
4. Robots regeneradores, donde el operario conduce el robot a través de una tarea.
5. Robots de control numérico, donde el operario programa el movimiento del robot en un computador o a través de aprendizaje manual.
6. Robots inteligentes, los cuales pueden entender e interactuar con el entorno.

Por otra parte, atendiendo al tipo de configuración general del robot se encuentra otra forma de clasificación basada en la arquitectura:

- **Poli-articulados:** Bajo este grupo están los robots de diversas formas cuya característica común es que son fijos. En este grupo se encuentran los manipuladores, los robots industriales o los robots cartesianos.
- **Móviles:** Son robots con gran capacidad de desplazamiento, dotados de un sistema locomotor.
- **Humanoides:** Son robots que intentan reproducir total o parcialmente la forma y el comportamiento cinemático del ser humano. Su principal característica es la locomoción bípeda.
- **Zoomórficos:** Este tipo de robots constituyen una clase caracterizada principalmente por sus sistemas de locomoción imitando el movimiento de diversos seres vivos.
- **Híbridos:** Estos robots corresponden a aquellos de difícil clasificación cuya estructura viene determinada por la combinación de las clases anteriores ya expuestas.

El desarrollo de este trabajo se centra en los robots móviles.

Los robots móviles pequeños o microbots por lo general están destinados a realizar tareas pequeñas con rapidez y precisión, similares a actividades de humanos y animales. Su principal filosofía es la movilidad, lo cual es muy ventajoso para realizar ciertas tareas y poder trabajar en equipo con otros robots o de manera aislada. Son útiles para realizar tareas en lugares peligrosos para los humanos, aburridos, sucios o difíciles. Tal es el caso de los robots móviles que se envían a Marte, o los que se utilizan para limpiar centrales nucleares.

Una de las campañas que ha hecho posible que los robots móviles sean conocidos son los concursos de robótica, que han despertado la curiosidad e imaginación de muchos, como '*DARPA Robotics Challenge*' o '*First Lego League*'

Si bien los robots móviles por lo general son de reducidas dimensiones con un bajo grado de control (el control existente es para regular la velocidad de las ruedas de tracción, el grado de control no es alto comparado con el nivel de control que se emplea en los robots articulados), adquieren un alto grado de complejidad cuando tienen que realizar tareas cooperativas. Esto ha dado como resultado que se realicen investigaciones sobre distintos temas, que luego pasan a formar parte de una gran lista de información existente en Internet. Información que puede ser utilizada para realizar nuevas investigaciones o para desarrollar nuevos prototipos de robots.

El comportamiento y la forma de actuar del microbot está determinado por el programa que se ejecuta en él. El software de los microbots se encuentra actualmente estructurado en tres niveles:

- Sistema operativo.
- Plataforma de desarrollo.
- Aplicaciones concretas.

Que un microbot realice automáticamente tareas de modo eficiente depende de la construcción mecánica y de la programación. La precisión en su desplazamiento se encuentra determinada por su sistema mecánico, mientras que la autonomía y la inteligencia se encuentran residentes en el programa que gobierna las acciones del microbot.

La influencia de los robots ha sido tan grande que ahora podemos hablar de un mercado de robots, el cual se encuentra en una fase de crecimiento. Cada vez son más los productos robóticos que salen al mercado, existiendo varios movimientos empresariales alrededor de la tecnología robótica. El mercado de robots móviles está aún inmaduro y todavía no se ha abierto al público. Esto se debe a que aún no se ha alcanzado el grado de autonomía y fiabilidad suficientes, que son necesarios

para conseguir la evolución futura en este mercado. El crecimiento en autonomía abre la puerta a nuevas aplicaciones comerciales con los robots, como son los robots de servicio y los de entretenimiento. Sin embargo, la autonomía es difícil, y salvo casos contados, el desarrollo de aplicaciones con robot sigue siendo un tema de investigación.

2.2.2 Estructura general de un robot móvil

Debido a que un robot móvil por lo general está destinado a simular el comportamiento de personas y animales con un nivel de eficiencia similar, la estructura del robot móvil es muy similar a la estructura del ser vivo.

En la figura 2.10 se puede observar de manera general la estructura, tanto de un robot móvil como de un ser vivo. Las diferentes estructuras que componen el esquema quedan definidas como:

- Estructura mecánica: estructuras con ruedas, patas y orugas.
- Actuadores: motores, luces, brazos, ruedas y, en definitiva, cualquier elemento que permita interactuar con el entorno.
- Sensores: sonar, láser, cámaras y cualquier elemento que proporcione información del entorno.
- Inteligencia: métodos, algoritmos, etc. Estos van a permitir, a partir de la información de los sensores, interactuar con el entorno.

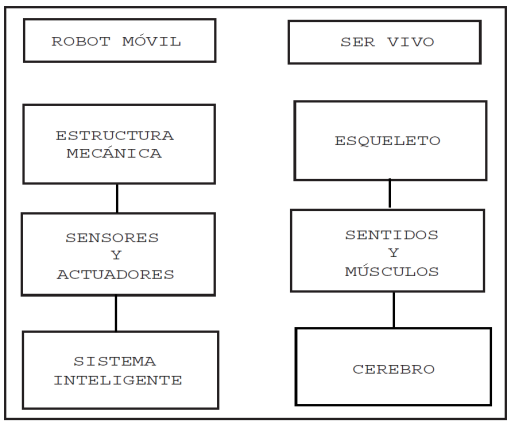

Figura 2.10: Similitudes entre un robot móvil y un ser vivo

2.2.3 Locomoción en robots móviles

Los robots son usados actualmente en distintas facetas profesionales como puede ser la fabricación industrial, la medicina, la seguridad y defensa o la investigación espacial, entre otras. Dependiendo de la tarea para la que se destine el robot, puede ser de gran importancia el tipo de movilidad que tenga, pudiendo depender el éxito de dicha tarea de la eficacia con que el robot se desenvuelva en el medio en el que trabaja. A continuación se presentan los tipos de locomoción más comunes [11] [4] [5].

Robots fijos

Los robots que no tienen movilidad suelen utilizarse en tareas industriales, especialmente en la producción en cadena. Poseen uno o varios brazos articulados programados para realizar una tarea monótona y periódica, o bien, manipulados mediante control remoto por el operario encargado. La desventaja de su inmovilidad en cuanto al desplazamiento es compensada por su precisión en el movimiento del brazo, pudiendo llegar a realizar trabajos delicados como la fabricación de circuitos eléctricos o incluso, manejado por un cirujano, operaciones médicas.

Robots con patas

Cuando la tarea a la que se destina el robot requiere de movilidad, los creadores de éstos han intentado imitar las distintas formas de desplazamiento que la naturaleza ha dotado a los animales, incluidos los humanos.

Al dotar de movimiento con patas a un robot, se debe tener en cuenta su posición y velocidad, pero también se debe asegurar que el robot permanezca en equilibrio y no se caiga, usando solamente el movimiento en las articulaciones mediante motores. En robots bípedos, el desplazamiento requiere necesariamente mantener el equilibrio en una de las patas mientras la otra realiza el movimiento, lo que conlleva una inestabilidad en cada paso.

Una posible solución para asegurar la estabilidad al desplazarse ha sido aumentar el número de patas. De esta forma, un robot de 6 patas puede sostenerse con gran estabilidad sobre 3 de sus patas mientras mueve las otras 3. Para el caso de 4 patas, el movimiento es más lento ya que debe sostenerse sobre 3 y mover 1 en cada paso.

Otra vía que se ha llevado a cabo para intentar mejor la estabilidad de los robots, así como mejorar su agilidad es construirlo de tal forma que su movimiento sea lo más parecido posible al de un humano, es decir, que sea bípedo. Para conseguir esto,

los tobillos deber ser móviles y, por tanto, estar dotados de motores que permitan al robot desplazarse y no perder el equilibrio. Los principales problemas que tiene este diseño son la poca velocidad que se puede proporcionar al robot y la gran cantidad de energía que necesita.

Una alternativa a la anterior consiste en eliminar los pies y sus articulaciones, dejando solamente las mínimas necesarias para las piernas (rodillas y pelvis). Mediante esta técnica se pretende que la única fuerza que genere un desplazamiento hacia delante sea la de la gravedad. Las articulaciones en las patas solamente van variando la posición del robot respecto al eje de inercia.

Robots con ruedas

La principal ventaja por la que se utilizan las ruedas como medio locomotor en un robot es que éstos son más fáciles de construir. Para que un robot que use ruedas pueda moverse simplemente hay que suministrar energía al eje de las ruedas motrices. Además, con este tipo de locomoción, el robot puede desplazar mayor peso que usando patas.

En cuanto a las desventajas, las ruedas no permiten salvar grandes obstáculos, en concreto, cualquier objeto que tenga más altura que el radio de la rueda, no podrá ser superado por esta.

Al diseñar el robot, se debe decidir cuál va a ser la disposición de las ruedas y cuáles van a ser las motrices, es decir, las que proporcionan el movimiento.

2.2.4 Robots móviles con ruedas

Ya que la plataforma móvil de estudio es un robot con ruedas, el trabajo se centrará en este tipo de locomoción, que se detalla en este apartado.

Los grados de libertad de un robot, así como los tipos de ruedas son aspectos que intervienen en el proceso de control y análisis de movimientos del robot .

Grado de libertad (GDL): Es el número de magnitudes que pueden variarse independientemente en un robot. En un manipulador robot cada articulación proporciona habitualmente un grado de libertad. Si se quiere posicionar el extremo del manipulador en cualquier punto del espacio en cualquier orientación se necesitarán seis grados de libertad, tres para la posición y tres para la orientación del extremo. En el caso de un vehículo que se desplaza en un plano serían tres grados de libertad, dos para la posición y uno para la orientación.

En un sistema no holonómico, las ecuaciones diferenciales no son integrables en la posición final del robot. El saber la distancia recorrida por cada rueda, no es suficiente para poder calcular la posición final del robot. Hay que conocer como fue ejecutado el movimiento, en función del tiempo.

Robot holonómico y no holonómico: Un robot es holonómico si es capaz de modificar su dirección instantáneamente, y sin necesidad de rotar previamente (tiene los mismos grados de libertad efectivos que controlables). Un robot no holonómico no podrá desplazarse lateralmente en un solo movimiento, tal y como se muestra en la figura 2.11.

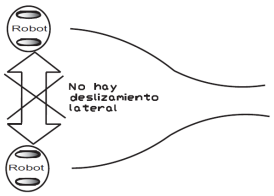

Figura 2.11: Sistema no holonómico

Tipos de ruedas

Las ruedas son el elemento que proporciona la capacidad de movilidad en un robot móvil. Se clasifican como [11]:

- ***Rueda fija.*** El movimiento se produce en la dirección de la rueda. Donde ω es la velocidad angular de la rueda, V la velocidad lineal y a_x el vector unitario de dirección. (Figura 2.12)

$$V = (r \cdot \omega) \cdot a_x \tag{2.2.1}$$

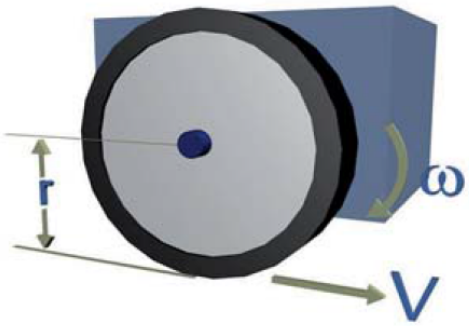

Figura 2.12: Rueda fija

- ***Rueda orientada centrada.*** Además del giro t de la rueda, existe rotación alrededor del eje vertical que está dirigido al centro de la rueda. Donde ω es la velocidad angular de la rueda, V la velocidad de avance lineal y a_x es un vector unitario de dirección. (Figura 2.13)

$$V = (r \cdot \omega) \cdot a_x \tag{2.2.2}$$

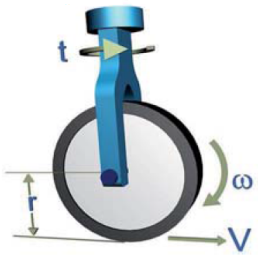

Figura 2.13: Rueda orientada centrada

- ***Rueda orientada descentrada o libre.*** Gira sobre el eje de la rueda y rota alrededor del eje vertical situado a una distancia d desde el centro de la rueda. Donde ω es la velocidad angular de la rueda, V la velocidad lineal y a_x es un vector unitario de dirección. (Figura 2.14)

$$V = (r \cdot \omega) \cdot a_x + (d \cdot t) \cdot a_y \tag{2.2.3}$$

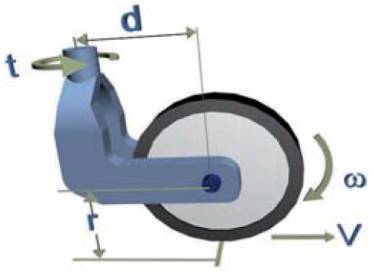

Figura 2.14: Rueda libre

- ***Rueda sueca.*** Además de moverse en la dirección de la rueda, se mueve en dirección perpendicular a la dirección de la rueda. Donde U es la velocidad de deslizamiento, ω es la velocidad angular de la rueda y V la velocidad lineal. (Figura 2.15)

$$V = (r \cdot \omega) \cdot a_x + U \cdot a_x \tag{2.2.4}$$

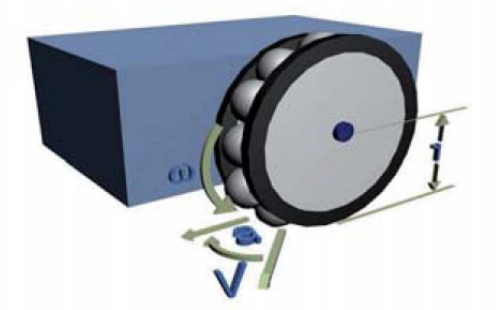

Figura 2.15: Rueda sueca

Centro instantáneo de rotación (ICR) o centro instantáneo de curvatura (ICC): Se define como el punto por el cual cruzan los ejes de todas las ruedas; es el punto alrededor del cual el robot gira en un instante determinado. En la figura 2.16 se muestra el ICC para la configuración diferencial, Ackerman y triciclo.

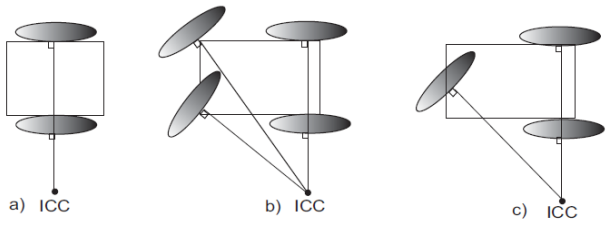

Figura 2.16: ICC en a) Configuración diferencial, b) Configuración Ackerman y c) Configuración en triciclo

2.2.5 Configuraciones típicas para robots móviles con ruedas

La elección del tipo de configuración es uno de los pasos necesarios para la construcción de un robot móvil. La configuración hace referencia a la forma en que se encuentran distribuidos los principales elementos que lo componen: plataformas, motores, ruedas, etc. Existen distintas configuraciones para robots móviles con ruedas: diferencial, triciclo, Ackerman, sincronizada, omnidireccional, con múltiples grados de libertad y desplazamiento mediante orugas [11].

Configuración diferencial

Es la más sencilla de todas. Consta de dos ruedas colocadas en el eje perpendicular a la dirección del robot. Cada rueda es controlada por un motor, de tal forma que el giro del robot queda determinado por la diferencia de velocidad de las ruedas. Así, para girar a la izquierda, hay que darle una velocidad mayor a la rueda derecha y para girar a la derecha, hay que darle una velocidad mayor a la rueda izquierda. En la figura 2.17, se muestra este tipo de configuración.

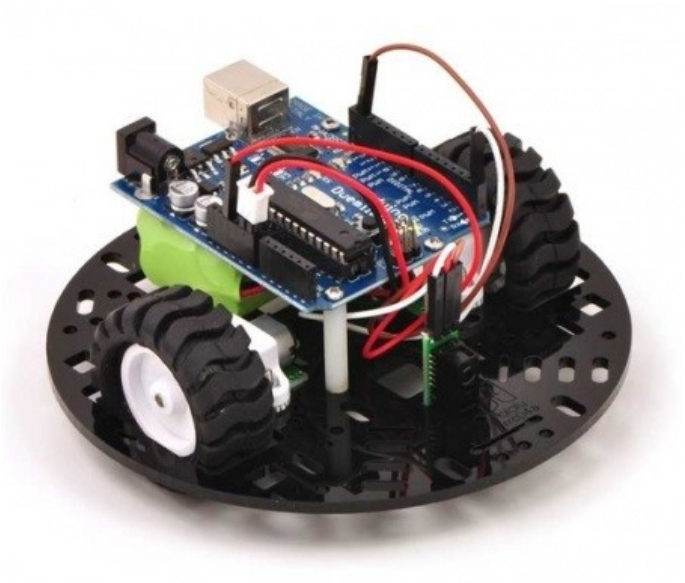

Figura 2.17: Robot diferencial

Uno de los problemas que tiene la configuración diferencial es mantener el equilibrio del robot, ya que consta de dos ruedas, por lo cual se le agregan ruedas de giro libre. Estas ruedas sirven para mantener horizontal al robot, por lo que giran libremente según el movimiento y orientándose hacia la dirección del robot. Dependiendo de las necesidades, se pueden agregar una o más ruedas de giro libre.

El problema que surge debido a más de tres apoyos en el robot es la pérdida de tracción y graves errores en los cálculos de odometría cuando el robot se encuentra en terrenos irregulares (recuérdese que la odometría es la técnica que calcula la distancia desplazada por el robot midiendo el giro de las ruedas). La falta de tracción es provocada cuando existe más de una rueda de giro libre, esto se muestra en la figura 2.18.

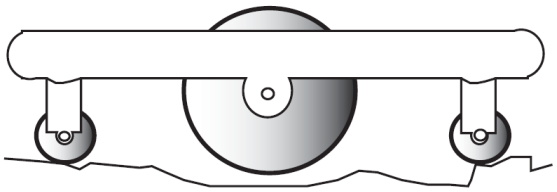

Figura 2.18: Falta de tracción debido a las ruedas de giro libre

Este tipo de robots no son aptos para terrenos irregulares, por lo que su campo de acción se encuentra limitado a superficies planas.

Debido a que la estructura diferencial es muy empleada, se calculará a continuación la posición (x, y) momentánea del robot relativo a un punto de comienzo mediante el uso de la odometría y posteriormente mediante la cinemática del robot.

Antes de realizar el cálculo mediante odometría es necesario saber cuántas revoluciones ha dado la rueda, para lo cual se hace uso de encoders incrementales, los mismos que van colocados sobre los motores de tracción, entonces, se emplea un factor que convierte los pulsos del encoder en una descomposición lineal de la rueda para poder calcular las velocidades de cada rueda. El factor de conversión es determinado por la siguiente ecuación.

$$C_m = \frac{\pi \cdot D_n}{n \cdot C_e} \tag{2.2.5}$$

Donde:

C_m: Factor de conversión que convierte los pulsos del encoder en una descomposición linal de la rueda.

D_n: Diámetro nominal de la rueda (mm).

C_e: Resolución del encoder en pulsos por revolución.

n: Relación de engranajes. Engranajes de reducción entre el motor y la rueda de tracción.

Ahora supóngase que a intervalos de muestreo i, los encoders de la rueda izquierda y derecha muestran un incremento de pulsos, N_L para el encoder de la rueda izquierda y N_R para el encoder de la rueda derecha.

Se puede calcular entonces el incremento de la distancia recorrida por la rueda izquierda $\Delta U_{L,i}$ y derecha $\Delta U_{R,i}$ tal y como muestra la ecuación (2.2.6).

$$\Delta U_{L/R,i} = C_m \cdot N_{L/R,i} \tag{2.2.6}$$

El incremento lineal de la distancia recorrida por el centro C del robot denotado como ΔU_i se calcula por medio de la ecuación (2.2.7).

$$\Delta U_i = \frac{\Delta U_R + \Delta U_L}{2} \tag{2.2.7}$$

Ahora, se obtiene el incremento en el cambio de orientación como se indica en la ecuación (2.2.8).

$$\Delta \theta_i = \frac{\Delta U_R - \Delta U_L}{b} \tag{2.2.8}$$

Donde b es la distancia ente los dos puntos de contacto de las ruedas con el suelo.

La nueva orientación relativa θ_i del robot puede ser calculada con la ecuación (2.2.9).

$$\theta_i = \theta_{i-1} + \Delta \theta_i \tag{2.2.9}$$

Entonces, la posición (x, y) relativa del punto central del robot queda expresada como las ecuaciones finales (2.2.10a) y (2.2.10b).

$$x_i = x_{i-1} + \Delta U_i \cdot \cos(\theta_i) \tag{2.2.10a}$$
$$y_i = y_{i-1} + \Delta U_i \cdot \text{sen}(\theta_i) \tag{2.2.10b}$$

Donde x_i e y_i son la posición relativa del punto central c del robot en un instante i.

A continuación se calcula la posición del robot, mediante el uso de la cinemática. La cinemática trata sobre los aspectos geométricos del movimiento, se caracteriza por especificar en un momento determinado, la posición, velocidad y aceleración de un objeto. Las ecuaciones a determinar mediante el uso de la cinemática son (2.2.11a) y (2.2.11b).

$$x_c = f_1(t) \tag{2.2.11a}$$
$$y_c = f_2(t) \tag{2.2.11b}$$
$$\theta_c = f_3(t) \tag{2.2.11c}$$

Cada rueda está controlada por un motor y existe un solo punto de contacto entre las ruedas y la superficie. Esto se muestra en la figura 2.19.

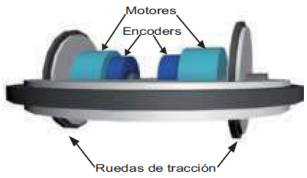

Figura 2.19: Robot con estructura diferencial

Se supone un movimiento sin deslizamiento respecto al suelo en la dirección ortogonal a la del movimiento.

Para poder obtener las ecuaciones cinemáticas es necesario establecer gráficamente todos los factores involucrados, entonces el modelo cinemático se ilustra en la figura 2.20, donde se denomina p al punto de contacto existente entre la rueda con la superficie, v la velocidad de traslación del centro del robot, ω la velocidad angular del robot con respecto a su centro de gravedad, cr es el centro instantáneo de rotación, r es el radio de las ruedas, L es el ancho del robot, V_R es la velocidad del motor derecho, V_L es la velocidad del motor izquierdo, V_C es la velocidad promedio de los dos motores, θ_C es el ángulo del robot respecto a las coordenadas absolutas (x, y), s el vector $[v, \omega]$ de velocidad del robot, P_s el vector $[x_c, y_c, \theta_C]$ que representa la posición y orientación del robot y (x_c, y_c) son las coordenadas del centro de masa del robot.

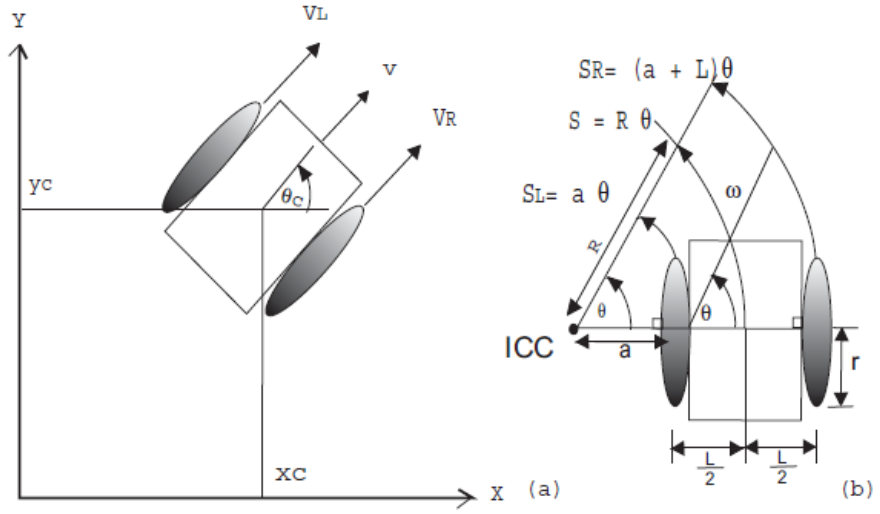

Figura 2.20: Modelo cinemático del robot diferencial

Todos los cálculos están referidos a la figura 2.20.

Se parte de la ecuación general para la velocidad de una rueda, la ecuación (2.2.12).

$$v = (r \cdot \omega) \tag{2.2.12}$$

A partir de esta ecuación se calcula V_R y V_L.

$$V_R = r \cdot \omega_R \tag{2.2.13a}$$
$$V_L = r \cdot \omega_L \tag{2.2.13b}$$

Se comienza por calcular S_L que es la distancia recorrida por la rueda izquierda, S_R que es la distancia recorrida por la rueda derecha y S que es la distancia recorrida por el punto central del robot, tal como muestran las siguientes ecuaciones.

$$S_L = a \cdot \theta \tag{2.2.14a}$$
$$S_R = (a + L) \cdot \theta \tag{2.2.14b}$$
$$S = \theta \cdot R \tag{2.2.14c}$$

En la ecuación (2.2.14c), R es el radio de curvatura instantáneo de la trayectoria del robot.

Al reemplazar la ecuación (2.2.14a) en la ecuación (2.2.14b), se obtiene:

$$S_R = S_L + L \cdot \theta \tag{2.2.15}$$

Para obtener ω se deriva la ecuación (2.2.15), dando como resultado la ecuación (2.2.16).

$$V_R = V_L + L \cdot \omega \tag{2.2.16}$$

Al depsejar ω de la ecuación (2.2.16) se obtiene:

$$\omega = \frac{V_R - V_L}{L} \tag{2.2.17}$$

Ahora, al sumar las ecuaciónes (2.2.14a) y (2.2.14b), resulta:

$$S_L + S_R = 2 \cdot a \cdot \theta + L \cdot \theta = 2\theta \cdot \left(a + \frac{L}{2}\right) \tag{2.2.18}$$

Donde:

$$R = a + \frac{L}{2} \tag{2.2.19}$$

Entonces, reemplazando la ecuación (2.2.14c) en la ecuación (2.2.18), se obtiene:

$$S = \frac{S_R + S_L}{2} \tag{2.2.20}$$

Al derivar la ecuación (2.2.20), se consigue la velocidad resultante. Esta velocidad queda expresada en la ecuación (2.2.21).

$$v = \frac{V_R + V_L}{2} \tag{2.2.21}$$

Cálculo de R:

Al derivar la ecuación (2.2.14c), se tiene.

$$v = R \cdot \omega \tag{2.2.22}$$

Por lo que, al reemplazar las ecuaciones (2.2.21) y (2.2.17) en la ecuación anterior, se obtiene:

$$R = \frac{L}{2} \cdot \left(\frac{V_R + V_L}{V_R - V_L} \right) \tag{2.2.23}$$

Al analizar la ecuación (2.2.23), $R = \infty$ cuando $V_R = V_L$, esto quiere decir que el ICC se encuentra en el infinito o que el robot se moverá en linea recta. Si $V_R = -V_L$ entonces $R = 0$, por lo que el ICC se encontraría dentro del robot y éste giraría alrededor de su centro.

Ahora, a partir de las ecuaciones (2.2.21) y (2.2.17) se obtiene s.

$$s = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{V_R + V_L}{2} \\ \\ \frac{V_R - V_L}{2} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \\ \frac{-1}{L} & \frac{1}{L} \end{bmatrix} \cdot \begin{bmatrix} V_L \\ V_R \end{bmatrix} \tag{2.2.24}$$

A continuación se obtiene $\dot{p}_s$ en la siguiente ecuación a partir del producto de la matriz jacobiana $J(\theta)$ con s.

$$\dot{p}_s = \begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\theta}_c \end{bmatrix} = \begin{bmatrix} \cos(\theta_c) & 0 \\ \text{sen}(\theta_c) & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v \\ \omega \end{bmatrix} = J(\theta_c) \cdot s \tag{2.2.25}$$

Para obtener la posición del robot integramos la ecuación (2.2.25) como se indica en las ecuaciones siguientes:

$$x_c = \int_0^t v(T) \cdot \cos(\theta_c) dT \tag{2.2.26a}$$

$$y_c = \int_0^t v(T) \cdot \text{sen}(\theta_c) dT \tag{2.2.26b}$$

$$\theta_c = \int_0^t \omega(T) dT \tag{2.2.26c}$$

El punto ICC estará localizado en:

$$ICC = (x_c - R \cdot \text{sen}(\theta), y_c - R \cdot \cos(\theta)) \tag{2.2.27}$$

Configuración en triciclo

Consta de tres ruedas. Dos ruedas traseras pasivas que no están acopladas a ningún motor, sirven únicamente de soporte. La dirección y tracción del robot son proporcionadas por la rueda delantera. En este tipo de configuración no se pueden realizar giros de $\pm 90^o$. El radio de curvatura instantáneo siempre se encuentra a lo largo del eje que cruza las dos ruedas traseras. Un problema asociado con la configuración en triciclo es que el centro de gravedad del vehículo tiende a moverse lejos de la rueda delantera en terrenos inclinados, cuando el robot se encuentra en pendiente ascendente, causando la pérdida de tracción.

A continuación se calculará la posición del robot relativo a un punto de comienzo, mediante el uso de la cinemática. Las ecuaciones a las que se desea llegar mediante este método son las (2.2.28a) y (2.2.28b).

$$x_c = f_1(t) \tag{2.2.28a}$$

$$y_c = f_2(t) \tag{2.2.28b}$$

Para obtener las ecuaciones cinemáticas es necesario establecer el modelo cinemático, el cual muestra la figura 2.21, donde la rueda de dirección está a un ángulo $\alpha(t)$ a partir de la dirección en línea recta del robot. El triciclo rotará con una velocidad angular $\omega(t)$ alrededor del punto ICC.

Se denota como r al radio, $V_S(t)$ a la velocidad lineal y $\omega_s(t)$ a la velocidad angular, todas estas variables referidas a la rueda de dirección. La distancia existente entre el centro de la rueda delantera y el punto central entre las dos ruedas traseras se expresa como d.

La velocidad lineal de la rueda delantera es obtenida mediante la ecuación (2.2.12), resultando la ecuación (2.2.29).

$$V_s(t) = \omega_s(t) \cdot r \tag{2.2.29}$$

El radio de curvatura instantáneo R se calcula como:

$$R = d \cdot \tan\left(\frac{pi}{2} - \alpha(t)\right) \tag{2.2.30}$$

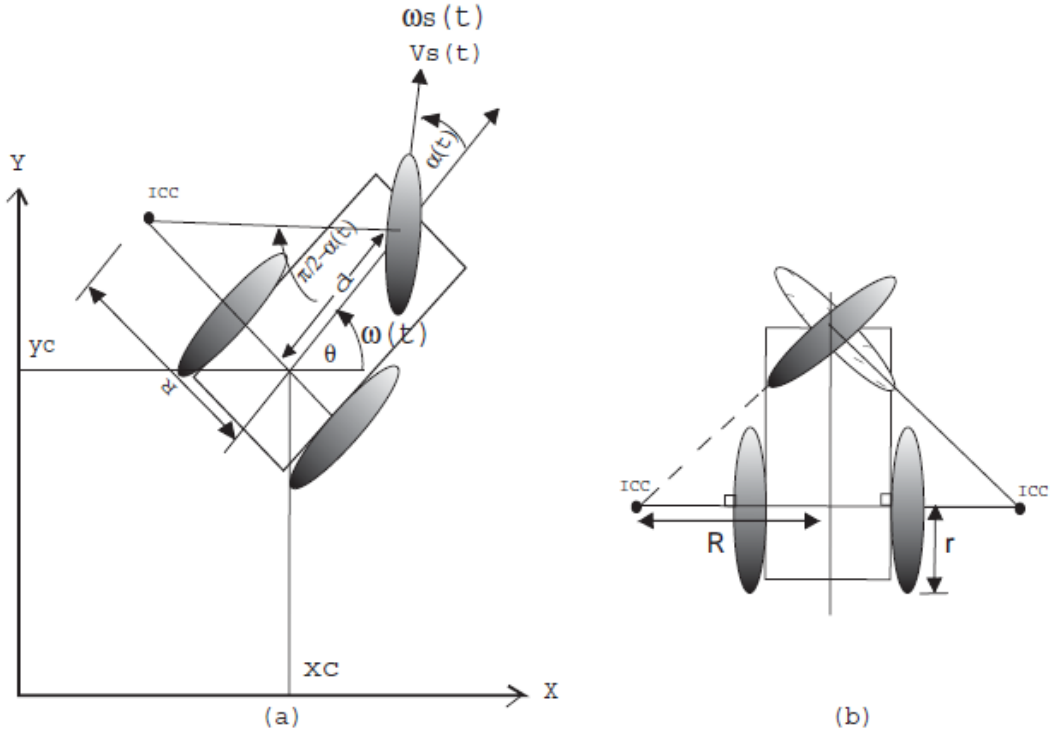

Figura 2.21: Sistema no holonómico

Ahora partimos de la ecuación siguiente para obtener $\omega(t)$.

$$S = d\theta \tag{2.2.31}$$

Donde S es la distancia de la curvatura alrededor de θ.
Aplicamos la derivada a la ecuación (2.2.31) y resulta:

$$V = d \cdot \omega(t) \tag{2.2.32}$$

V es una componente de la velocidad de la rueda delantera del robot, que se encuentra a un ángulo $\alpha(t)$ respecto al horizonte del robot. Dicha componente se encuentra localizada en el eje perpendicular al robot. Entonces, la velocidad V será: $V = r \cdot \omega_s(t) \cdot \text{sen}(\alpha(t))$. Este valor de V se reemplaza en la ecuación (2.2.32), dando como resultado:

$$V_S(t) \cdot \text{sen}\left(\alpha(t)\right) = d \cdot \omega(t) \tag{2.2.33}$$

Ahora, se despeja $\omega(t)$ de la ecuación (2.2.33):

$$\omega(t) = \frac{V_S(t) \cdot \text{sen}\left(\alpha(t)\right)}{d} = \dot{\theta} \tag{2.2.34}$$

La velocidad $v(t)$ del robot queda expresada en la ecuación (2.2.35).

$$v(t) = V \cdot s \cdot \cos\left(\alpha(t)\right) \tag{2.2.35}$$

Las ecuaciones para $\dot{x}_c$ e $\dot{y}_c$, que son las ecuaciones finales que indican la posición del robot, se escriben como en las ecuaciones (2.2.36a) y (2.2.36b).

$$\dot{x}_c = v(t) \cdot \cos\left(\theta(t)\right) = V_S(t) \cdot \cos\left(\alpha(t)\right) \cdot \cos\left(\alpha(t)\right) \tag{2.2.36a}$$

$$\dot{y}_c = v(t) \cdot \text{sen}\left(\theta(t)\right) = V_S(t) \cdot \cos\left(\alpha(t)\right) \cdot \text{sen}\left(\alpha(t)\right) \tag{2.2.36b}$$

La representación matricial de las ecuaciones (2.2.35), (2.2.36a) y (2.2.36b) es:

$$\begin{bmatrix} \dot{x}_c(t) \\ \dot{y}_c(t) \\ \dot{\theta}_c(t) \end{bmatrix} = \begin{bmatrix} \cos(\theta_c) & 0 \\ \text{sen}(\theta_c) & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \tag{2.2.37}$$

Entonces, para obtener la posición del robot integramos la ecuación (2.2.37) como se muestra en las siguientes ecuaciones.

$$x_c = \int_0^t v(T) \cdot \cos\left(\theta(T)\right) dT \tag{2.2.38a}$$

$$y_c = \int_0^t v(T) \cdot \text{sen}\left(\theta(T)\right) dT \tag{2.2.38b}$$

$$\theta_c = \int_0^t \omega(T) dT \tag{2.2.38c}$$

Para el caso de la "***configuración en bicicleta***", el proceso para la obtención de las ecuaciones es similar al empleado en la configuración en triciclo. Esta configuración se muestra en la figura 2.22.

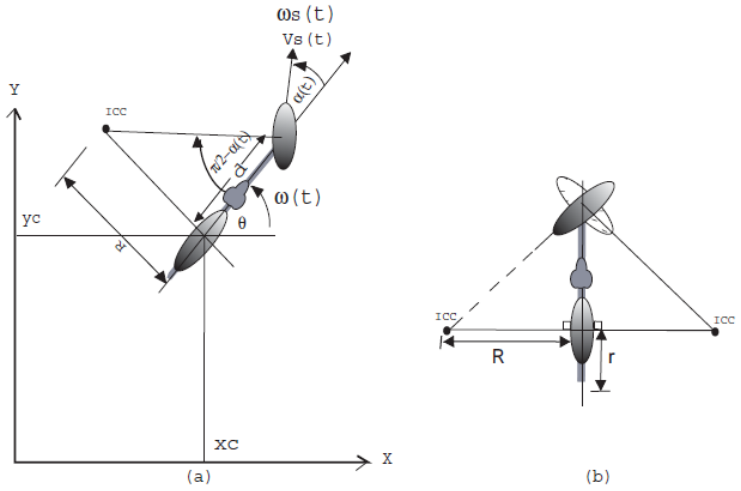

Figura 2.22: Configuración en bicicleta

Configuración Ackerman

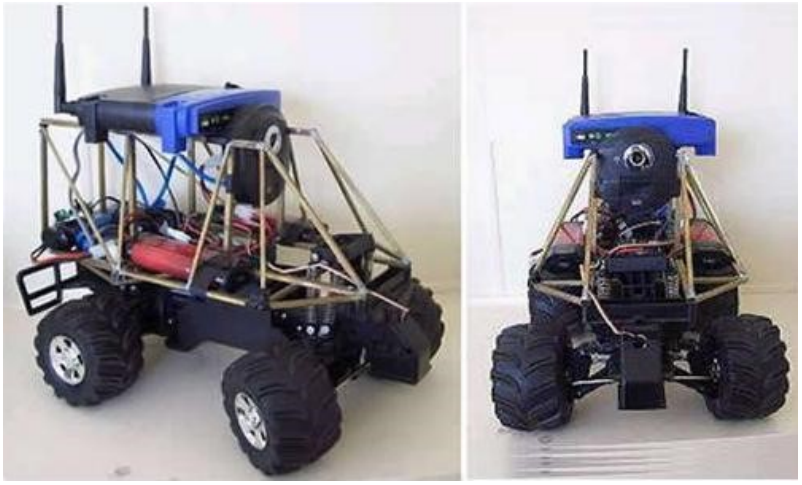

Figura 2.23: Robot con configuración Ackerman

Esta configuración es la que utiliza la industria del automóvil. Posee dos ruedas traseras de tracción y dos ruedas delanteras para la dirección. Con esta configuración se puede evitar el derrape de las ruedas, lo que se consigue haciendo que la rueda delantera interior posea un ángulo θ_i ligeramente mayor que el ángulo de la rueda exterior θ_o cuando el sistema se encuentra girando. El modelo cinemático puede verse en la figura 2.24.

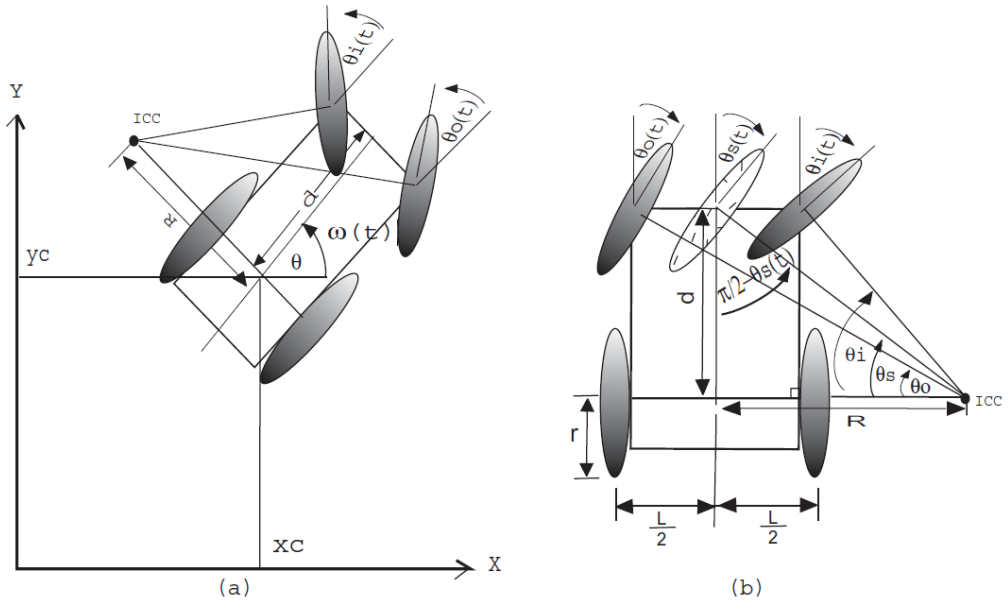

Figura 2.24: Configuración Ackerman

Donde:
L: Separación lateral entre las ruedas.
D: Separación longitudinal entre las ruedas.
ω: Velocidad angular del robot.
v: Velocidad lineal del robot.
θ: Ángulo de rotación del robot.

La configuración Ackerman constituye un buen sistema de tracción, que se puede aplicar incluso para terrenos inclinados.

A continuación se obtendrán las ecuaciones para la posición del robot en función del tiempo, las ecuaciones que se plantean conseguir mediante el uso de la cinemática son las (2.2.39a) y (2.2.39b).

$$x_c = f_1(t) \tag{2.2.39a}$$
$$y_c = f_2(t) \tag{2.2.39b}$$

Según la figura 2.24, que representa el modelo cinemático para la configuración Ackerman, se obtienen las ecuaciones (2.2.40a) y (2.2.40b), que relaciona la dirección de las dos ruedas delanteras.

$$\cot\left(\theta_i(t)\right) = \frac{\left(R - \frac{L}{2}\right)}{d} \tag{2.2.40a}$$

$$\cot\left(\theta_o(t)\right) = \frac{\left(R - \frac{L}{2}\right)}{d} \tag{2.2.40b}$$

Al igualar el Radio de Curvatura Instantáneo R en la ecuación (2.2.40a) y (2.2.40b), resulta la ecuación (2.2.41):

$$\cot\left(\theta_o(t)\right) - \cot\left(\theta_i(t)\right) = \frac{L}{d} \tag{2.2.41}$$

En la figura 2.24 (b), se ha colocado una rueda imaginaria adicional, entre las dos ruedas delanteras. Si se deja a un lado las dos ruedas reales, podemos apreciar una configuración en triciclo. Por ello, al calcular el ángulo de giro de la rueda adicional, igualando R de la ecuación (2.2.42a) con las ecuaciones (2.2.40a) y (2.2.40b), se obtienen las ecuaciones (2.2.42b) y (2.2.42c).

$$\cot\left(\theta_s(t)\right) = \frac{R}{d} \tag{2.2.42a}$$

$$\cot\left(\theta_s(t)\right) = \cot\left(\theta_i(t)\right) + \frac{L}{2 \cdot d} \tag{2.2.42b}$$

$$\cot\left(\theta_s(t)\right) = \cot\left(\theta_o(t)\right) + \frac{L}{2 \cdot d} \tag{2.2.42c}$$

$\theta_s(t)$ es equivalente al ángulo α_s en las ecuaciones calculadas para la configuración en triciclo.

La velocidad del robot v es la velocidad que poseen las ruedas de tracción, que en la configuración Ackerman corresponden a las ruedas traseras. La velocidad de la rueda imaginaria, V_S, es calculada como se indica en la ecuación (2.2.43).

$$V_S = \frac{v}{\cos\left(\theta_s(t)\right)} \tag{2.2.43}$$

Para obtener la posición del robot se emplea el mismo método y las mismas ecuaciones que para la configuración en triciclo.

Configuración sincronizada

Este tipo de configuración está conformada por tres o más ruedas, todas ellas acopladas mecánicamente y dotadas de tracción, de tal forma que todas rotan en la misma dirección y a la misma velocidad. La orientación de los ejes de rotación se puede controlar.

Figura 2.25: Robot sincronizado B21

El ICC del robot está siempre en el infinito, lo que simplifica el control. Cambiando la orientación de las ruedas se puede manipular la dirección del ICC.

Este sistema mejora la odometría, reduciendo el deslizamiento de las ruedas, ya que todas las ruedas generan fuerzas con vectores de igual módulo y paralelos en todo momento.

La sincronización de las ruedas se realiza de forma mecánica aunque hay casos en los que se realiza de forma electrónica, colocando un motor en cada rueda.

El sistema sincronizado es holonómico, posee dos motores, uno para el movimiento y otro para la dirección. El robot no tiene la necesidad de girar para cambiar de dirección, ya que solo giran las ruedas. Este tipo de configuración se representa en la figura 2.26.

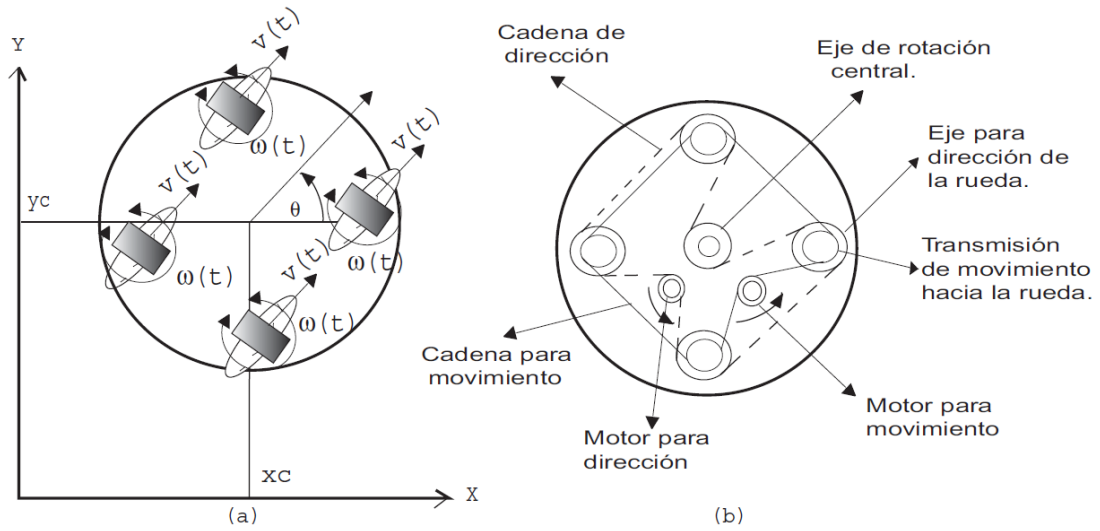

Figura 2.26: Configuración sincronizada

La posición del robot queda determinada por las ecuaciones (2.2.44a), (2.2.44b) y (2.2.44c).

$$x_c = \int_0^t v(T) \cdot \cos\left(\theta(T)\right) dT \tag{2.2.44a}$$

$$y_c = \int_0^t v(T) \cdot \text{sen}\left(\theta(T)\right) dT \tag{2.2.44b}$$

$$\theta_c = \int_0^t \omega(T) dT \tag{2.2.44c}$$

Configuración omnidireccional

Este tipo de configuración está provista de ruedas omnidireccionales, lo que hace que los cálculos de odometría sean más complejos, pero el robot podrá moverse en cualquier dirección.

La rueda omnidireccional se define como una rueda estándar a la cual se le ha dotado de una corona de rodillos. Los ejes de giro de los rodillos son perpendiculares a la dirección normal de avance. De este modo, al aplicarle una fuerza lateral, los rodillos giran sobre sí mismos y permite que la componente de la velocidad en el eje x no sea nula, y por tanto, se elimina la restricción de no holomicidad. Las ruedas omnidireccionales tienen por lo general un costo elevado.

Figura 2.27: Robot omnidireccional Rovio

El modelo cinemático de esta configuración se muestra en la figura 2.28. La configuración cinemática de este robot se define por una estructura triangular equilátera, en cuyos vértices se encuentran colocadas ruedas omnidireccionales. La distancia desde el centro del triángulo equilátero a cualquiera de las ruedas es L. Todas las ruedas son no direccionales. Se representa como ω_1, ω_2 y ω_3 a las velocidades angulares de las ruedas 1, 2 y 3.

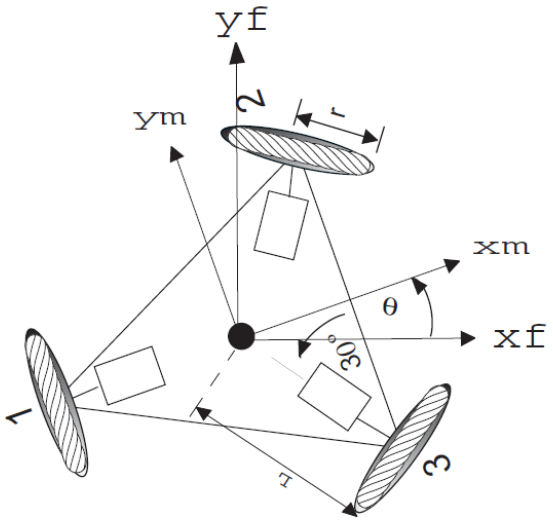

Figura 2.28: Configuración omnidireccional

Finalmente, el modelo cinemático en el marco del robot queda representado por medio de la ecuación (2.2.45), donde r es el radio de las ruedas.

$$\begin{bmatrix} V_x \\ V_y \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{\sqrt{3}} \cdot r & \frac{-1}{\sqrt{3}} \cdot r \\ \frac{2}{3} \cdot r & \frac{-1}{3} \cdot r & \frac{-1}{3} \cdot r \\ \frac{-r}{3 \cdot L} & \frac{-r}{3 \cdot L} & \frac{-r}{3 \cdot L} \end{bmatrix} \cdot \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{2.2.45}$$

Configuración con múltiples grados de libertad

Este tipo de configuración posee múltiples motores para la propulsión y dirección del vehículo, con lo cual, se mejora mucho la movilidad aunque se dificulta el control. El problema que surge con este tipo de configuración se debe al incremento del deslizamiento en las ruedas, por lo que se reduce la exactitud de la odometría. El problema principal radica en el control de la coordinación de todos los motores de forma simultánea. Un ejemplo de este tipo de configuración sería el robot Curiosity de la NASA (figura 2.29).

Figura 2.29: Robot Curiosity (NASA)

Tracción mediante orugas

Figura 2.30: Robot con orugas

En este tipo de configuración se sustituyen las ruedas por orugas (figura 2.30). Esta implementación es un caso especial de la configuración diferencial. En esta configuración, el deslizamiento en los giros es muy grande, por lo que se pierde exactitud en el cálculo de la odometría. Se emplea en lugares donde el suelo es muy irregular. Debido a que no es posible saber la posición exacta en la que se encuentra el robot, su uso general es para robot teleoperados, lo que constituye algo opuesto a las aplicaciones de los robots autónomos. Más concretamente, son utilizados en búsqueda y rescate, desactivación de bombas y en la industria nuclear.

Al poseer orugas, el vehículo tiene una gran área de contacto, lo que impide los hundimientos en tierra suave. Para la maniobrabilidad se usa direccionamiento mediante rodillos, lo que causa deslizamientos mayores y por lo tanto, dificulta la capacidad de determinar la posición en la que se encuentra el robot. La exactitud del direccionamiento viene a ser más difícil a altas velocidades, debido al deslizamiento y a la resistencia de las huellas desiguales, causadas por las fuerzas centrífugas.

2.2.6 Estimación de la posición de un robot

Los robots móviles se caracterizan por su capacidad de desplazarse de un lugar a otro en forma autónoma (capacidad de percibir, modelar, planificar y actuar para alcanzar unos objetivos sin la intervención, o con poca intervención de personas), ya sea en un lugar conocido parcialmente o desconocido en su totalidad.

Un robot móvil es un sistema en el cual se encuentran inmersos diversos subsistemas de locomoción, control de movimientos, percepción y planificación, que interaccionan entre sí. El subsistema de percepción hace que el robot sea capaz de interactuar en entornos cambiantes, así como poder reaccionar ante eventos inesperados, lo que exige la existencia de un sistema sensorial que suministre información sobre el entorno. Esta información requerida debe permitir al robot realizar tres tareas fundamentales: estimar su posición y orientación, mantener actualizado el mapa del entorno y detectar los posibles obstáculos [15].

El robot móvil rara vez va equipado con un único sensor para realizar todas estas tareas, sino que la práctica más habitual consiste en combinar dentro del sistema sensorial varios sensores, que en mayor o menor medida se complementan. Algunas veces se emplean varios sensores redundantes con el propósito de validar la información adquirida.

Para que un robot móvil pueda afrontar tareas como generar trayectorias, evitar obstáculos, etc; se requiere que éste sea capaz de determinar su localización (posición y orientación) con respecto a un sistema de referencia absoluto.

La estimación de la posición de un robot móvil, viene dado por el tipo de entorno por el cual ha de navegar, el conocimiento que se tenga sobre el entorno, tipos de sensores que dispone y de la tarea a realizar.

La mayoría de robots móviles disponen de encoders para detectar su movimiento, lo que permite estimar en cada instante su posición, empleándose modelos de locomoción. Sin embargo, esta estimación no resulta muy conveniente para la mayoría de aplicaciones, ya que no es lo suficientemente precisa para ello. El motivo es que los errores, por más pequeños que sean, se van acumulando durante la navegación, por lo cual se suelen usar sistemas de posicionamiento externo. Las principales técnicas utilizadas para la estimación de la posición y orientación de un robot autónomo pueden verse en la figura 2.31.

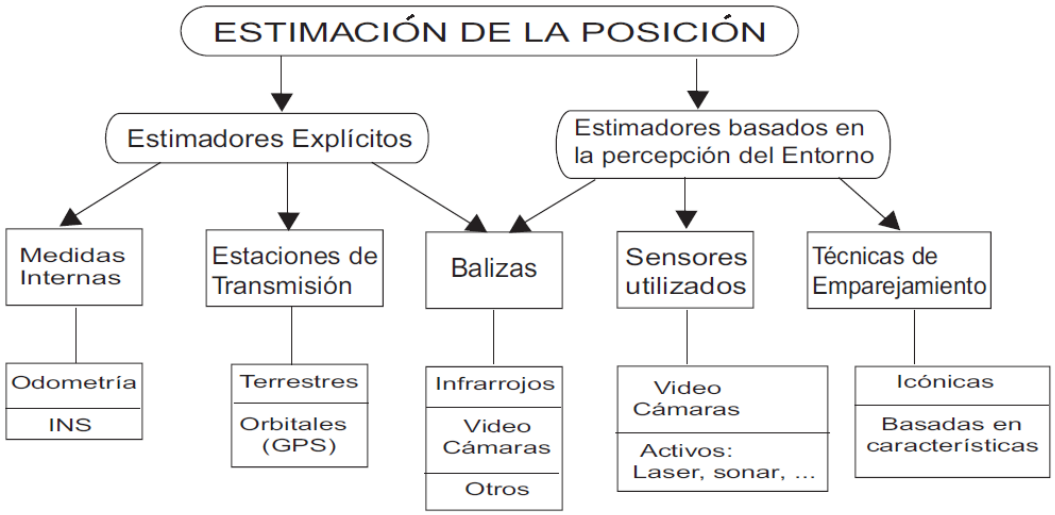

Figura 2.31: Sistemas para la estimación de la posición en robots [15]

Para la estimación de la posición y orientación de un robot móvil autónomo se distinguen los estimadores explícitos y los estimadores basados en la precepción del entorno. Los estimadores explícitos proporcionan directamente la posición y orientación del robot por medio de las medidas, sin tener que realizar procesamiento de información para la interpretación del entorno. Los estimadores basados en la percepción del entorno emplean sensores que suministran información del exterior, por medio de la cual se puede saber la localización del vehículo mediante comparación de esta información con otros datos o modelos conocidos (pueden ser mapas, marcas naturales, objetos, etc) [15].

El problema de la estimación reside en que las medidas se encuentran asociadas a una cierta cantidad de ruido. Tanto las propias medidas como las estimaciones realizadas a partir de estas, tendrán una naturaleza aleatoria. Si no existiese el ruido, la posición y orientación del robot móvil se obtendría simplemente al resolver el modelo matemático.

Estimación explícita de la posición

Se considera como estimación explícita a todos aquellos sistemas capaces de estimar la posición del robot móvil sin que se realice una interpretación del entorno.

En los sistemas de estimación explícita se distingue la estimación basada en medidas internas y la estimación basada en estaciones de transmisión.

La estimación basada en medidas internas trabaja solamente con los sensores internos del robot móvil. Los sensores empleados pueden ser: giróscopos, encoders, brújulas, acelerómetros, tacómetros, etc. La estimación basada en estaciones de transmisión se basa en el empleo de dos unidades, la unidad montada sobre el robot móvil y la unidad o unidades externas. La unidad montada sobre el robot móvil actúa como sensor receptor, mientras que las externas actúan como emisores o transmisores de señales de referencia.

- **Estimadores basados en medidas internas**

  La posición y orientación de un robot móvil puede obtenerse integrando la trayectoria recorrida por este a partir de una serie de medidas internas como pueden ser: las vueltas que dan las ruedas, la velocidad, aceleración, cambios de orientación, etc. Se pueden distinguir los sistemas odométricos y los sistemas de navegación inercial:

  - *Sistemas odométricos*

    La odometría tiene por objeto determinar la posición y orientación del robot móvil a partir del número de pulsos obtenidos cuando giran las ruedas. Se utilizan codificadores ópticos de elevada precisión en al menos dos ruedas. La simplicidad y el bajo costo es una gran ventaja que nos ofrece el sistema odométrico. Sin embargo, es necesario una calibración constante debido al desgaste y pérdida de presión de las ruedas, desajuste de los ejes, etc. Esta técnica es vulnerable a las imperfecciones en el suelo, al deslizamiento de las ruedas y a las variaciones en la carga transportada (aunque en este caso es posible diseñar un modelo para corregir las desviaciones introducidas). La idea fundamental de la odometría es la integración de los incrementos del movimiento en el tiempo, lo que produce inevitablemente una acumulación de errores. La acumulación de errores en la orientación causa grandes errores en la posición, los cuales se incrementan proporcionalmente con la distancia recorrida por el robot. A pesar de estas limitaciones, la odometría es una parte importante en los sistemas de navegación del robot. Las tareas de navegación serán simplificadas si se mejora la precisión de la odometría.

  - *Sistemas de navegación inercial (INS)*

    Este método obtiene la posición y orientación del robot móvil por medio de las medidas de aceleraciones y ángulos de orientación. Para obtener la posición, se integra la aceleración obteniéndose la velocidad, la cual se integra para calcular la posición. Como se ha dicho, este sistema emplea la aceleración para la obtención de la posición, para lo cual hace uso de los acelerómetros, que suelen estar basados en sistemas pendulares.

La precisión del acelerómetro resulta crítica, ya que, los errores en la aceleración, aunque sean pequeños, afectan en la obtención de la posición, debido a la doble integración que hay que realizar. Para medir la orientación se emplean giróscopos e inclinómetros. También es posible medir el ángulo de orientación mediante brújulas. Los sistemas de navegación inercial no son afectados por los problemas derivados de la interacción del vehículo con el suelo. En este sistema se pueden corregir los efectos de las ondulaciones e irregularidades en el terreno, lo que hace que en la práctica sean mucho más fiables y precisos que los sistemas basados en odometría. Estos sistemas son más frágiles y caros que los anteriores.

- **Estimadores basados en estaciones de transmisión**

Este tipo de sistema de posicionamiento es absoluto y su ventaja más destacada es proporcionar la posición absoluta del robot móvil en un área suficientemente grande, sin tener que estructurar el entorno, lo que hace que el robot móvil pueda moverse por escenarios muy diversos, recorriendo grandes distancias (carreteras, caminos forestales, parajes semidesérticos, etc.).

Estos sistemas constan de un receptor, que se encuentra colocado en el robot móvil, y de estaciones de transmisión de radiofrecuencia. Dentro de este sistema se encuentran los sistemas de posicionamiento mediante estaciones fijas y los sistemas de posicionamiento mediante estaciones móviles. Los sistemas de posicionamiento mediante estaciones fijas se han empleado en la navegación de barcos, submarinos y aviones. Utilizan señales de radio de media y alta frecuencia que son emitidas por estaciones terrestres fijas. Los sistemas de posicionamiento mediante estaciones móviles son operados mediante satélites. Su utilización era en aplicaciones militares (misiles, aviación, submarinos) y en la navegación civil (fundamentalmente aviación).

Hoy en día se emplea un sistema similar pero mucho más potente denominado GPS ("Global Positioning System") (figura 2.32) en el cual, al menos utilizando 3 satélites, el receptor calcula por triangulación la altitud, latitud y altura del vehículo de forma instantánea y continua. También se puede determinar la velocidad a partir del desplazamiento en frecuencias mediante el efecto Doppler.

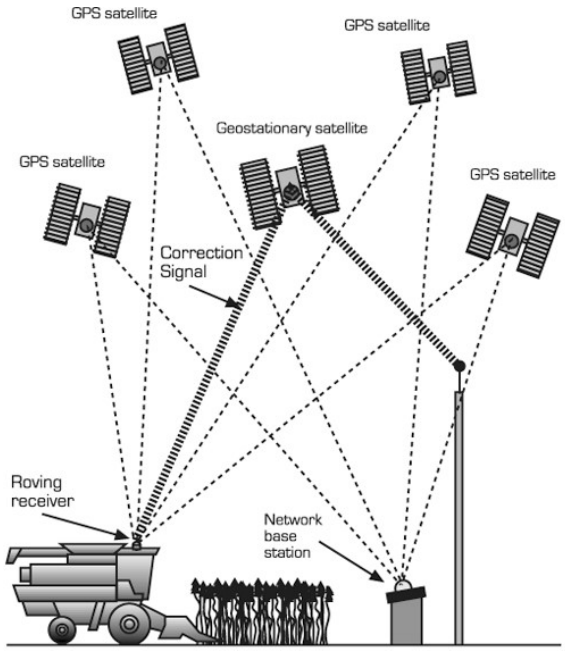

Figura 2.32: Sistemas GPS

- **Estimadores basados en balizas**

  Permite determinar la posición del robot móvil en un entorno delimitado por balizas que se encuentran en posiciones conocidas (figura 2.33). La posición del robot es determinada de una forma más o menos directa en base al principio de triangulación, a partir de medidas de distancias, ángulos o de la combinación de ambas.

  La precisión y fiabilidad que este tipo de estimación proporciona depende fundamentalmente del tipo de señal utilizada (infrarrojos, láser, radio, ultrasonidos, etc.), de las características del sensor y del número de balizas utilizadas en la triangulación. Si bien, este método está considerado como uno de los más precisos, las principales desventajas radican en la necesidad tanto de configurar apropiadamente el entorno de trabajo, como de garantizar que un número suficiente de estas señales quede en todo momento libre de obstrucciones y dentro del campo visual del sensor. También deben tenerse en cuenta los problemas que pueden originar las condiciones ambientales de iluminación y ruido (acústico, electromagnético, etc.). Todo ello impide la correcta utilización de esta técnica en entornos muy dinámicos o no estructurados.

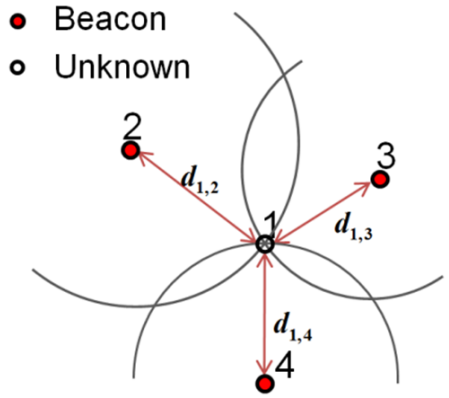

Figura 2.33: Balizas de posición

Estimación mediante percepción del entorno

Consiste en dotar al robot móvil de un sistema sensorial capaz de proporcionar suficiente información del entorno como para que este pueda de forma autónoma determinar su localización. El sistema sensorial puede operar en base a distintos tipos de sensores (cámaras CCD, sónares, telémetros láser, etc.) y seleccionando determinados tipos de datos u objetos a partir del conjunto de información adquirida (marcas naturales, puntos de interés, entorno completo percibido, etc.). En cualquier caso, la localización del robot móvil se determina a través del emparejamiento de los datos extraídos del entorno por el sistema sensorial, con datos previamente conocidos del entorno.

Los sensores utilizados para la navegación de un robot móvil pueden situarse en dos grupos: activos y pasivos. Los sensores activos son aquellos que emiten algún tipo de energía al medio (luz infrarroja, ultrasonidos, luz láser, ondas de radio, etc.). Los sensores denominados pasivos se limitan a captar la energía ya existente en el medio. De este tipo son las cámaras de vídeo CCD, las cuales perciben el entorno a través de la cantidad de luz que les llega procedente de fuentes luminosas o bien a través de reflexiones en los objetos del entorno.

2.2.7 Métodos de navegación

La navegación permite guiar a un robot móvil a través de un entorno con obstáculos. El objetivo es llevar al robot móvil a su destino de forma segura. Las tareas involucradas en la navegación son: la percepción del entorno a través de sus sensores, la planificación de una trayectoria libre de obstáculos y el guiado del vehículo a través de la referencia construida.

Por tanto, la navegación es el proceso de usar los datos de los sensores para producir una representación del entorno. La navegación contiene tres componentes principales: cartografía, planificación y acción. La cartografía utiliza los datos de los sensores para formar un modelo medioambiental (mapa). El mapa es usado en la etapa de planificación para calcular la dirección del destino evitando los obstáculos de la forma más segura. Cuando la dirección es encontrada, la acción entra en funcionamiento y el robot comienza a moverse hacia el lugar de destino [2].

El problema de la navegación se divide en las cuatro etapas siguientes [2]:

- *Percepción del mundo*: Mediante el uso de los sensores externos, se crea un mapa del entorno donde se desarrollará la tarea de navegación.
- *Planificación de la ruta*: Crea una secuencia ordenada de objetivos o submetas que deben ser alcanzadas por el robot móvil. La secuencia es calculada utilizando el modelo o mapa de entorno, la descripción de la tarea y algún tipo de procedimiento estratégico.
- *Generación del camino*: Se define una función continua que interpola la secuencia de objetivos y luego la discretiza a fin de tomar el camino.
- *Seguimiento del camino*: Según el camino generado, hace que el robot móvil empiece a desplazarse mediante el control adecuado de los actuadores.

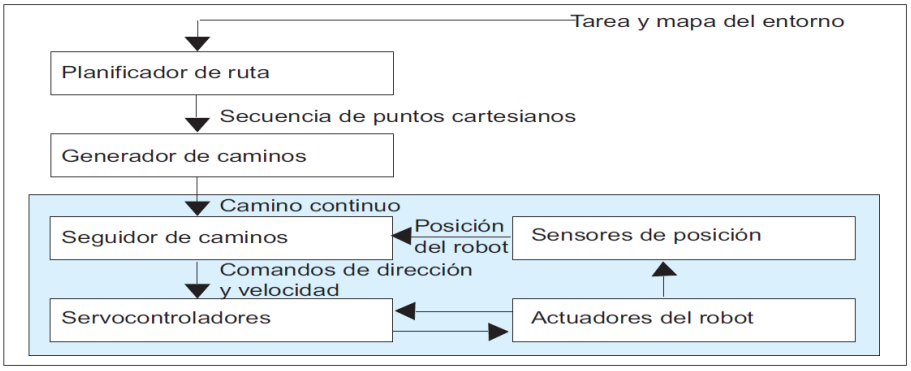

Figura 2.34: Estructura de navegación de un robot móvil

En la figura 2.34 se muestra la estructura para un control de navegación básico, la cual parte de un mapa de entorno y de las especificaciones de la tarea de navegación. Posteriormente se realiza la planificación de un conjunto de objetivos, los cuales se encuentran representados por puntos cartesianos dispersos que definen la ruta a seguir.

Mediante el uso del generador de caminos se construye la referencia que utilizará el seguidor de caminos para generar los comandos para la dirección y velocidad que actuarán sobre los servocontroladores del robot móvil. Finalmente, mediante el uso de los sensores internos del robot móvil (sensores de posición) en conjunto con las técnicas odométricas, se produce una estimación de la posición actual, la cual será realimentada al seguidor de caminos.

Cuando el esquema presentado en la figura anterior se torne ineficiente habrá que introducir en la estructura de control básica nuevos elementos que solucionen los imperfectos. Un esquema de navegación utilizado cuando la información acerca del entorno de trabajo varía desde un perfecto conocimiento del mismo hasta cuando se posee un cierto grado de incertidumbre es el mostrado en la figura 2.35. Dicho sistema corresponde al planteado en el robot móvil Blanche de los laboratorios AT/T.

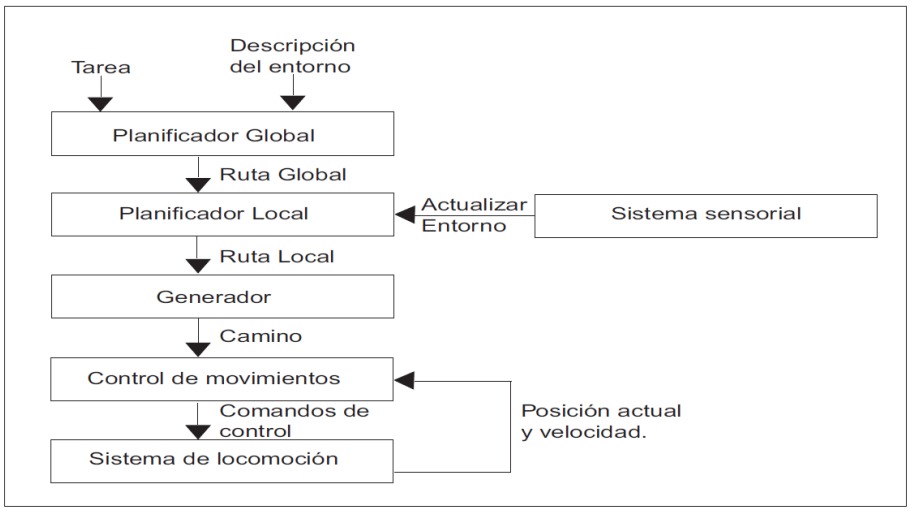

Figura 2.35: Sistema de navegación del robot móvil Blanche

El esquema presentado en la figura anterior se diferencia de la estructura de navegación básica debido a que la tarea de planificación se encuentra dividida en dos subtareas que son: planificación global y local. La primera de estas subtareas forma la ruta sobre la cual se define un camino libre de obstáculos según la información que se posee del entorno. La tarea de planificación local recibe la información del sistema sensorial sobre el entorno local del robot, según el alcance de los sensores externos. Mediante el análisis de estos datos se actualiza el modelo preliminar del entorno y se decide si es necesario replanificar la ruta local del robot.

A continuación definimos la planificación global y la planificación local:

- *Planificación global*: Planificación de la ruta que lleve al robot a cada una de las submetas determinadas, según las especificaciones del problema a resolver. Con esta planificación se obtiene una aproximación del camino final que seguirá el robot, no se consideran los detalles del entorno local del robot móvil.

- *Planificación local*: Con esta planificación se intenta solucionar las obstrucciones sobre la ruta global en el entorno local del robot. El modelo del entorno local se construye mediante la fusión de la información proporcionada por los sensores externos del robot móvil.

La ruta global puede realizarse antes de que el robot móvil comience a realizar la tarea, mientras que la planificación local se lleva a cabo en el tiempo de ejecución. En el caso de realizar una navegación sobre entornos totalmente conocidos es obvio que resulta innecesario proceder a una planificación local. Cuando no se conoce el entorno por el cual el robot móvil realizará su tarea, se emplea la planificación local.

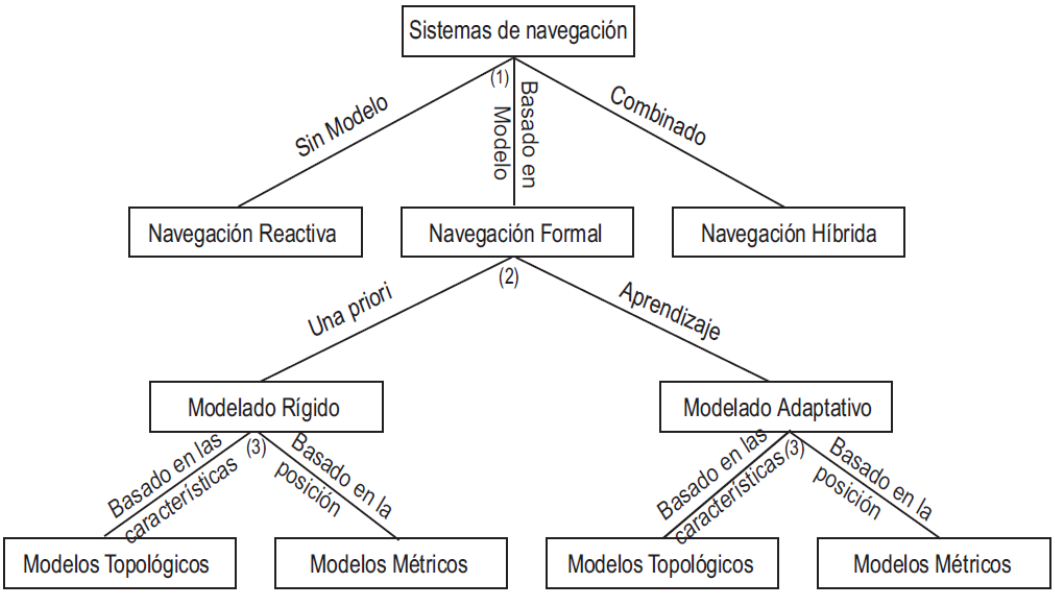

Figura 2.36: Árbol de Clasificación de los métodos de navegación para robots móviles

La figura 2.36 muestra la clasificación de los métodos de navegación. Cada rama en el árbol conduce a diferentes alternativas en las correspondientes características. Cada tipo de esquema de navegación formal (rígida y adaptativa) puede ser clasificado en topológico y métrico. En los modelos métricos, la posición y orientación numérica del robot es necesaria para propósitos de navegación mientras que

en los modelos topológicos la información de la navegación es más cualitativa que cuantitativa:

- *Modelos métricos*: En este modelo se especifica la posición y orientación de los objetos que se encuentran en el entorno en un sistema de referencia fijo y se dispone de abundante información métrica de todos los objetos.
- *Modelos topológicos*: En estos modelos la información del entorno se basa en una lista de lugares que pueden ser reconocidos por el robot y que está acompañada por una lista de conexiones entre los lugares. Puede existir algún tipo de información métrica pero sobre aspectos concretos.

Navegación reactiva vs planificación

En los métodos de navegación reactivos, los datos percibidos por los sensores van directamente a comandar el control de los actuadores, controlándose la velocidad y ángulo de dirección. Los comportamientos reactivos son jerárquicamente clasificados como emergentes y primitivos. Los comportamientos emergentes son similares a las tareas o metas para proyectar tareas. Comportamientos primitivos son sistemas de acciones ejecutadas simultáneamente o temporalmente para llevar a cabo el comportamiento emergente.

La navegación reactiva tiene la ventaja de tener un procesamiento muy rápido. Este tipo de navegación es buena para aplicaciones de bajo nivel, como puede ser el seguimiento de una pared o la evasión de obstáculos.

La navegación planificada confía en un mapa mucho más completo del entorno que lo rodea, por lo que es más intensiva computacionalmente y no permite la ejecución de la trayectoria global en tiempo real. Tiene la ventaja de memorizar el entorno global y local por medio de mapas. La cartografía permite planear sistemas para producir decisiones más inteligentes.

Sistemas de navegacion híbrida

El sistema híbrido adiciona a los métodos estándar habilidades reactivas, lo que ha demostrado ser excelente para tareas de navegación de nivel muy bajo en entornos complejos, desconocidos y variables. Por otro lado, la desventaja de un sistema de navegación puramente reactivo originado por la falta de planeamiento y capacidades de razonamiento es superada en un sistema híbrido debido a la incorporación de esquemas basados en modelos.

Por lo tanto un sistema híbrido combina las habilidades de navegación en bajo nivel de un sistema reactivo con habilidades de navegación en alto nivel de esquemas topológicos basados en modelos. El sistema híbrido está compuesto de dos subsistemas: un módulo de navegación reactiva y un modelo de navegación topológico.

El módulo de navegación reactiva está basado en la teoría de campos potenciales artificiales [9]. La teoría de campos potenciales considera al robot como una partícula dentro de un campo potencial, en la cual las variaciones locales de la función potencial representan la estructura del entorno. Los obstáculos son modelados como fuerzas repulsivas. Los movimientos del robot son reiterativamente calculados, obteniendo a cada paso la fuerza generada por el campo. Entonces, la trayectoria del robot es generada mediante la dirección de las fuerzas. El propósito es evidentemente guiar al robot móvil hacia la meta sin colisionar con los obstáculos.

Sistemas de navegacion basados en comportaminetos

En este sistema el proceso asociado con lo que se llama comportamientos de navegación del robot son ejecutados en un sentido cíclico, de modo que uno o más comportamientos son activados dependiendo de las condiciones del entorno. Una activación secuencial de comportamientos individuales produce lo que se denomina tareas de navegación.

Se definen tres grupos diferentes de comportamientos de navegación.

El primer grupo o grupos de comportamientos básicos permite al robot móvil poder realizar operaciones básicas de navegación tales como evadir obstáculos y dirigirse hacia adelante. Para este fin se emplean tres comportamientos que son: avance, evasión lateral y evasión frontal.

El segundo grupo de comportamientos o grupo de estrategias de navegación está formado por los comportamientos que realiza el robot móvil para activar los comportamientos básicos dependiendo de la estructura específica del entorno. Se definen cuatro estrategias de navegación: seguimiento de pared y evasión de obstáculos, desvío o rodear, girar y movimiento libre o distraído.

El tercero y último grupo de comportamientos operacionales comprende aquellos comportamientos necesarios para realizar tareas de navegación de nivel superior. Este grupo se encarga de seleccionar las tareas apropiadas para el robot, por ejemplo, activación de una misión de exploración para construir mapas de entorno o activación de una misión de excursión para alcanzar objetivos específicos y aquellos encargados de monitorizar el estado interno del robot.

Cartografía

Los mapas son muy importantes para el proceso de navegación, ya que son necesarios para la autonomía del robot móvil. El tipo de información que representa el entorno puede ser utilizada para clasificar los mapas como mapas de caminos, de espacio libre, orientados a objetos o compuestos.

Los mapas pueden ser de naturaleza local o global. Los mapas locales definen un área finita alrededor del robot móvil. La frontera de un mapa local pueden ser las paredes de un cuarto o un espacio alrededor del robot móvil. Los mapas globales son mucho más amplios. La extensión de un mapa global puede ser varios pisos en una edificación o una extensión entera en un camino con obstáculos. Los mapas globales son usados para colocar una dirección final de la meta. Los mapas locales incluyen información detallada, tal como obstáculos inesperados. La navegación reactiva suele contar únicamente con mapas locales. El planeamiento en la navegación requiere cartografía global, por lo menos a tal grado de memorizar mapas locales.

La incertidumbre y el ruido en los sensores es algo muy importante a tener en cuenta en todos los tipos de mapas. La posición detectada de un obstáculo caerá dentro de un rango de la posición actual debido a la resolución actual del sensor. Esto es especialmente significativo cuando se usan sensores de baja resolución. La cartografía a menudo confía en los datos de los sensores que son relativos a la localización del robot móvil. La localización del robot móvil es, a su vez, relativo a un sistema coordenado global. Otros mapas confían en los datos de múltiples sensores que son alineados según el movimiento relativo del robot móvil sobre un periodo de tiempo.

Se distinguen los siguientes tipos de mapas [3]:

- *Mapas de caminos*: Contienen listas de caminos o movimientos basados en programación humana u otros métodos. Los mapas son guardados internamente y usados para guiar al robot móvil con la ayuda de indicadores externos. Los mapas de caminos son usados extensivamente en aplicaciones industriales donde hay poca variación en la trayectoria del robot móvil. Estos mapas son típicamente de poco uso en robot móviles completamente autónomos.
- *Mapas de espacio libre*: Se interesa por los espacios entre obstáculos. Ya que un robot móvil prosigue a través de un entorno desconocido, los datos concernientes a la extensión del espacio libre son recolectados y la localización de las áreas ocupadas u obstáculos son grabados en un gráfico espacial.

- *Mapas orientados a objetos*: Se interesa por la localización de los obstáculos en el entorno. Los mapas de espacio libre están determinados por implicación (por ejemplo, áreas sin obstáculos). Los objetos son típicamente grabados como una colección de vértices relativos al marco de referencia global. Alternativamente, la posición y orientación de cada objeto puede estar especificada en una lista enlazada de (x,y) y θ. Estos mapas son muy efectivos en situaciones donde el entorno es bien conocido.

- *Mapas compuestos*: Ninguno de los mapas anteriores es capaz de representar completamente el entorno. Los mapas de espacio libre ignoran los objetos, mientras que los mapas orientados a objetos están solamente interesados en los objetos y en el espacio libre implicado. Los mapas compuestos abarcan los datos concernientes de los objetos y del espacio libre.

Planificación de caminos

Una vez que el mapa del entorno se encuentra generado, el planeamiento de caminos puede ocurrir en un nivel global o local. Los caminos globales dan direcciones generales para el destino, mientras que el planeamiento de caminos locales modifica el camino hacia la meta basado en obstáculos inesperados. El objetivo de ambos caminos proyectados es encontrar la trayectoria óptima para llega hacia la meta. Distancia, tiempo, requerimientos de energía, puede ser usados para definir un camino óptimo. Los caminos planeados pueden ser ampliamente clasificados como: exploración gráfica o campos de potencia.

2.2.8 Inteligencia artificial

La inteligencia artificial se encuentra definida de forma general por los autores Rich y Knight [19] como "*la capacidad que tienen las máquinas para realizar tareas que en el momento son realizadas por seres humanos*". La inteligencia artificial supone un serio esfuerzo por entender la complejidad de la experiencia humana en términos de procesamiento de información. No trata solamente sobre cómo representar y usar lógicamente una compleja e incompleta información, sino sobre cuestiones de cómo ver (visión), moverse (robótica), comunicarse (lenguaje natural, habla), aprender, etc.

Actualmente existen tres paradigmas en cuanto al desarrollo de la IA:

- *Redes neuronales.*
- *Algoritmos genéticos.*
- *Sistemas de lógica difusa.*

Una aplicación de la IA es el reconocimiento de ambientes en robótica móvil por medio de redes neuronales. Este estudio está centrado en la identificación global de ambientes ejecutada por un robot móvil con base en el entrenamiento de una red neuronal que recibe la información captada del medio ambiente por el sistema sensorial del robot. Se considera que el robot, a través de la red neuronal, tiene como única tarea maximizar el conocimiento del ambiente que se le presenta. De esta forma, éste modela y explora el ambiente eficientemente mientras se ejecutan algoritmos de evasión de obstáculos.

La IA es la ciencia que estudia el modo de mejorar las capacidades cognoscitivas de las máquinas, de modo que se acerquen lo más posible al razonamiento humano. Gracias a esta ciencia uno o más robots móviles pueden ser capaces de resolver tareas complejas a partir de razonamientos.

Robots móviles pequeños como agentes

Los agentes son un reflejo de los diferentes campos de la IA. Un agente realiza tareas a favor del dueño, adquiriendo un cierto grado de inteligencia, lo cual le permite interactuar con su entorno de un modo útil y autónomo.

El agente es considerado como autónomo si el tipo de acciones que realiza dependen de su experiencia y no del conocimiento previo introducido en el momento del diseño. Por lo tanto, para realizar bien su tarea requiere:

- *Metas*: Es lo que se quiere conseguir.
- *Sensores*: Permiten captar información del entorno.
- *Inteligencia*: Permite interpretar las percepciones del entorno.
- *Actuadores*: Empleados para llevar a cabo las acciones que permitan al robot alcanzar las metas.

Se pueden diferenciar tres tipos de agentes:

1. *Agentes humanos.*
2. *Agentes software.*
3. *Agentes hardware.*

Para elegir el tipo de agente que se va a diseñar es indispensable conocer el entorno y los objetivos que se requiere que cumpla. A partir de esto se determinarán las posibles percepciones y acciones que se necesitan para el cumplimiento de estas metas, con lo cual se diseña el programa de agente.

Comunicación entre agentes

La comunicación entre agentes es un punto muy importante, en especial cuando varios robots móviles tienen que trabajar en equipo o conjuntamente en un mismo entorno, por lo cual, no hay un método mejor que otro, sino más bien, cada método que se emplee dependerá de la aplicación que se desee realizar. Las posibles opciones para realizar la comunicación son las siguientes [7]:

- *Métodos de llamadas*

  Cada agente dispone de sus propios procedimientos. Estos procedimientos captan los datos, los procesan y obtienen otros datos. Si un agente necesita un dato que lo obtuvo otro, debe llamar al procedimiento correspondiente para recibir algún valor como respuesta. Este método no es adecuado para un intercambio constante de información.

- *Métodos de la pizarra*

  A la zona de trabajo común se le denomina pizarra, donde cada agente puede dejar y coger datos. La comunicación no se realiza entre agentes, sino más bien, entre agente y pizarra. Los agentes pueden tanto leer como escribir en la pizarra. En este método es necesario establecer algún protocolo de orden debido a los conflictos que se producen cuando varios agentes acceden a un mismo dato en un momento determinado. El método de la pizarra puede evolucionar hacia una solución más elaborada en la que entra en juego el elemento del moderador. Éste se encargaría de recoger los datos de cada agente y distribuirlos según su criterio, es decir, de estar al tanto de los caminos e informar solo a aquellos agentes a quienes pudiera interesar. Esta técnica es empleada en los campeonatos de fútbol robótico.

- *Sistemas basados en mensajes*

  En este sistema se permite implementar estrategias de coordinación complejas. Consta de un protocolo conocido por todos los agentes, pero el mensaje no está restringido a comandos y respuestas. Un agente al recibir un mensaje no devuelve automáticamente un dato, sino que analiza el mensaje para conocer su tipo y responder de la manera correcta. Un agente al pedir cierta información no está seguro de que el agente con el cual se comunica la tenga, cosa que sí ocurría en el sistema de llamadas, pero puede decirle que consiga dicha información comunicándose con otros agentes.

- *Sistemas de contratos*

  En este sistema la tarea objetivo es dividida en subtareas. Usando una metodología similar a la de un mercado, las subtareas salen a la venta y cualquier agente puede solicitar las que sean de su interés. Las subtareas se asignan al "mejor postor", es decir, a aquel agente que tenga mejores recursos para ejecutar dicha subtarea. El "vendedor" deberá por tanto tener un conocimiento de las características de cada agente. Se dice que cuando a un agente se le ha aceptado como ejecutor de una subtarea se ha realizado un contrato.

- *PGP: Partial Global Planning*

  En este método de comunicación, la información sobre el estado y los objetos obtenidos por todos los agentes está a disposición de cualquier agente. De tal forma que determinado agente puede cambiar de estrategia conforme emplea la información obtenida por otros agentes.

2.2.9 Cooperación en robótica móvil

La robótica cooperativa engloba un campo de investigación innovador con varias aplicaciones en áreas muy diversas, destacándose su gran influencia en el fútbol robótico, el cual es un problema muy desafiante, donde los robots tienen que cooperar no solo para empujar y/o golpear un objeto (pelota), sino también detectar y esquivar obstáculos estáticos (paredes, robots detenidos) y dinámicos (robots en movimiento) mientras se mueven con o siguiendo la pelota. Por lo tanto, los robots tienen que cooperar para derrotar al equipo contrario. Todas estas características son muy comunes para muchos otros problemas en robótica cooperativa.

Una población de robots cooperativos se comportan tal y como un robot distribuido para llevar a cabo tareas que resultan dificultosas para un simple robot.

Algunos problemas pueden ser resueltos de una manera más sencilla, más fiable, o más rápida por un grupo de robots en lugar de uno solo; pero puede ser algo complicado, por ejemplo, el planeamiento de caminos, que es mucho más sencillo para un solo robot que para un grupo. A parte de esto, un grupo de robots puede ser visto como un sistema multiagente.

Se puede notar también que un grupo de robots puede ser visto como un multi-robot con control centralizado. Mientras semejante punto de vista es de hecho posible, hay un número de argumentos en contra de esta perspectiva. Primero, este multi-robot tiene un número muy extenso de grados de libertad, de manera que resulta computacionalmente difícil de controlar.

Segundo, la comunicación con las distintas partes del robot puede ser imposible o puede ser que el ancho de banda de comunicación del robot sea muy bajo. Tercero, resulta complejo estimar el estado del sistema global porque se desconoce el estado de algunas partes del robot. Cuarto, el fallo de las partes de un multi-robot se traduce en fallos distribuidos.

Censado cooperativamente

Si hay un grupo de robots que pueden comunicarse entre sí, entonces estos deben compartir sus observaciones entre sí. De este modo, se pueden compensar las limitaciones de los sensores, por ejemplo, la restricción del rango en que un objeto puede ser sensado.

Todas las medidas de los sensores son inciertas. Hay siempre algo de ruido y un error sistemático que no se puede anticipar, por ejemplo, cuando se utiliza la odometría. Por estas razones se emplean otras medidas para solucionar el problema de la localización. Las medidas de otros sensores son usados para corregir las estimaciones derivadas de la odometría. La herramienta matemática que es empleada para solventar este problema es a menudo el filtro de *Kalman*. Este método combina todas las mediciones para obtener una estimación óptima.

2.2.10 Niveles de diseño en robótica móvil

Una herramienta útil para el desarrollo de un robot es la torre Bot, la cual divide en diferentes etapas o niveles la fabricación de robots. Los niveles proporcionados por la torre Bot se muestran en la tabla 2:

NIVEL DE COOPERACIÓN
NIVEL DE COMUNIDAD
NIVEL DE INTELIGENCIA
NIVEL DE DE CONTROL
NIVEL DE REACCIÓN
NIVEL FÍSICO

Tabla 2.1: Torre Bot

El nivel inferior (Nivel físico) es el primer paso que debemos dar para evaluar las necesidades y tener un cierto orden de prioridades. El nivel superior (Nivel de cooperación) es el último paso, aunque esto no quiere decir que se tenga que completar hasta esta etapa, ya que depende de la aplicación que se realice. El significado de cada nivel es el siguiente:

- *Nivel físico*: Está conformado por la estructura física, las unidades motoras y las etapas de potencia.

- *Nivel de reacción*: Está formado por los transductores, es decir, el conjunto de sensores y circuitos de polarización. Este tipo de unidades trabajan cumpliendo la premisa, acción o reacción. Los sensores son los propios controladores de las unidades motoras, sin ningún tipo de control intermedio.

- *Nivel de control*: Incluye los circuitos más básicos que relacionan las salidas de los sensores con las unidades restantes. Partiendo de una simple lógica digital hasta potentes microcontroladores, se busca dotar al robot de la capacidad para procesar la información obtenida por los sensores, así como actuar de una manera controlada sobre las unidades motoras.

- *Nivel de inteligencia*: Se trata de un planificador a largo plazo. En este nivel, se introducen los objetivos del robot que tienen relativa independencia de los sensores. Este es el más alto nivel de inteligencia que puede alcanzar un robot como una unidad individual.

- *Nivel de comunidad*: Consiste en conformar recintos que se denominan granjas, en donde se pone en funcionamiento a más de un robot dentro de un mismo entorno, de forma simultánea y sin que ninguno de ellos tenga conocimientos explícitos de la existencia de los otros que se encuentran en su mismo entorno.

- *Nivel de cooperación*: Comprende los sistemas donde a partir de un nivel de comunidad, se planifican o programan los robots para que tengan conocimiento de la existencia de otros, tal que posean la capacidad de cooperar para el buen desarrollo de una tarea.

2.2.11 Sensores y actuadores

Un sensor es un elemento que permite al robot saber ciertas características del entorno por el cual se desplaza. En otras palabras, los sensores son los dispositivos que permiten a un robot percibir su entorno.

Un sensor es un transductor que convierte algún fenómeno físico en señales eléctricas que el microprocesador del robot puede procesar. Los hay de muy variadas clases y se utilizan dependiendo de lo que se desea realizar.

Los actuadores son los elementos que acciona el robot para reaccionar a los estímulos de los sensores. Para los robots móviles los actuadores son fundamentalmente el medio de locomoción que tenga.

Tipos de sensores

A continuación se detallan algunos tipos de sensores que se emplean a menudo en robótica móvil [12].

- *Sensores para medir distancias basados en ultrasonidos*

  Son una tecnología de medida activa en donde se emite una señal ultrasónica en forma de pulso, para posteriormente recibir el reflejo de la misma o eco. Se pueden explorar diferentes aspectos de la señal reflejada: el tiempo de vuelo o la atenuación. Los sensores de ultrasonidos están formados por una cápsula emisora y otra receptora situada al lado de la emisora o bien por un transductor que actúa de emisor y receptor (figura 2.37). En los robots móviles se suelen montar en la periferia de forma que los sensores se encuentren separados a intervalos uniformes a lo largo del contorno del robot. Una estrategia alternativa es colocar un sensor montado en una plataforma rotatoria, obteniendo así omnidireccionalidad.

Figura 2.37: Sensores de ultrasonidos

- *Sensores para medir distancias basados en infrarrojos*

  A través de estos sensores se pueden estimar las distancias a las que se encuentran los objetos en el entorno. La precisión que se obtiene con estos sensores es muy elevada, debido a que son muy direccionales al ser muy pequeña su longitud de onda. La distancia máxima de medida depende de la potencia que se aplica al rayo de salida. Los sensores de infrarrojo se suelen montar de forma similar a los sensores de ultrasonidos.

  Estos sensores tienen un emisor y un receptor de luz (figura 2.38), recogen la luz que emiten al rebotar en los objetos y calculan la distancia a la que se encuentra. Es utilizado en la navegación de robots en laberintos.

Figura 2.38: Sensor de infrarrojos

- *Sensores CCD*

  Físicamente, un sensor CCD es una malla de electrodos de polisilicio colocados sobre la superficie de un chip. Al impactar los fotones sobre el silicio se generan electrones que pueden guardarse temporalmente. Periódicamente se lee el contenido de cada píxel haciendo que los electrones se desplacen físicamente desde la posición donde se originaron (en la superficie del chip) hacia el amplificador de la señal, con lo que se genera una corriente eléctrica que será proporcional al número de fotones que llegaron al píxel.

  Cámaras CCD (Charge Coupled Device): El captor CCD es un mosaico de diodos sensibles a la luz: cuanto más brillante es la luz recibida, mayor es la carga eléctrica. Este mosaico es sensible a la luz, pero insensible a los colores. Para recuperar los colores de una imagen, la información captada por los fotositos es pasada a través de diversos filtros. En la figura 2.39 se puede ver la apariencia de este tipo de sensor.

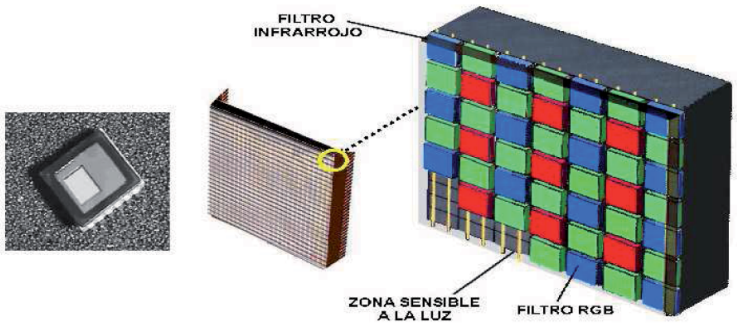

Figura 2.39: Sensor CCD

- *Sensores de proximidad*

Los sensores de proximidad suelen tener una salida binaria que indica la presencia de un objeto dentro de un intervalo de distancia especificado. En condiciones normales, los sensores de proximidad se utilizan en robótica para agarrar o evitar un objeto. Cualquier sensor que mida distancia se puede usar como sensor de proximidad.

Sensores inductivos: Son sensores basados en un cambio de inductancia debido a la presencia de un objeto metálico, están entre los sensores de proximidad industriales de uso más frecuente. La figura 2.40 muestra un sensor inductivo que consiste fundamentalmente en una bobina arrollada, situada junto a un imán permanente empaquetado en un receptáculo simple y robusto.

Figura 2.40: Sensor inductivo de proximidad

Sensores de efecto Hall: El efecto Hall relaciona la tensión entre dos puntos de un material conductor o semiconductor con un campo magnético a través del material. Cuando se utilizan por sí mismos, los sensores de efecto Hall sólo pueden detectar objetos magnetizados. Sin embargo, cuando se emplean en conjunción con un imán permanente en la configuración indicada en la figura 2.41, son capaces de detectar todos los materiales ferromagnéticos. Cuando se utilizan de dicha manera, un dispositivo de efecto Hall detecta un campo magnético intenso en ausencia de un material ferromagnético en el campo cercano.

Figura 2.41: Sensor magnético de proximidad

Sensores capacitivos: A diferencia de los sensores inductivos y de efecto Hall, que detectan solamente materiales ferromagnéticos, los sensores capacitivos son potencialmente capaces (con diversos grados de sensibilidad) de detectar todos los materiales sólidos y líquidos. Como su nombre indica, estos sensores se basan en la detección de un cambio en la capacidad inducida por una superficie que se encuentra cerca del elemento sensor (figura 2.42).

Figura 2.42: Sensor capacitivo de proximidad

- *Celda fotoconductora*

  La celda fotoconductora (figura 2.43) es un dispositivo semiconductor de dos terminales cuya resistencia entre terminales varía linealmente con la intensidad de la luz incidente. Se le llama con frecuencia dispositivo foto resistivo.

Figura 2.43: Sensor LDR o sensor foto resistivo

- *Sensores de contacto*

  Son elementos metálicos que nos indican si se ha producido un impacto. De forma general podemos decir que son interruptores (figura 2.44). Este tipo de sensores se emplean en robots reactivos.

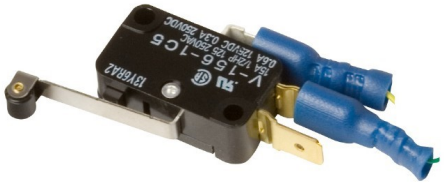

Figura 2.44: Sensor de contacto

- *Sensores para medir la orientación del robot*

  Dentro de este tipo de sensores se encuentran la brújula y el inclinómetro, ya que ambos suelen funcionar de forma conjunta para poder estimar la orientación del robot y son sensores de posicionamiento de medida absoluta. Un inclinómetro es un dispositivo muy simple y barato que mide la orientación del vector gravitacional. Los más comunes son de mercurio. Para poder medir la inclinación es necesario que se encuentren en una plataforma que no esté sometida a aceleración, para que la medida no sea errónea. Este sensor es muy sensible a las vibraciones. La brújula usa el campo magnético de la tierra para poder conocer la orientación del robot. La ventaja fundamental reside en que son sensores de medida absoluta, miden la orientación en cualquier lugar del mundo. Un pequeño error en la orientación supone cometer constantemente

errores en la posición según avanza el robot, por eso tener un sensor que ayude a corregir estos pequeños errores de forma inmediata permite al robot moverse de forma más precisa.

Las limitaciones que presentan estos sensores son:

- Son sensibles a los campos magnéticos externos. Si provienen del robot y son constantes se pueden corregir. Esto se denomina Hard Iron Distorsion. Para evitarlos basta con separarse de la fuente magnética una pequeña distancia, ya que el campo se atenúa con la distancia al cuadrado.
- Son sensibles a los elementos metálicos que están muy cerca del robot, ya que distorsionan el campo magnético de la tierra. Para que suceda esto el elemento metálico tiene que estar muy cerca del robot. Si la distorsión viene causada por el propio robot se puede corregir. Esto es lo que se denomina Soft Iron Distorsion.

Las precisiones que se pueden obtener oscilan desde cinco grados los más baratos, hasta décimas de grado los más caros. Existen brújulas de dos dimensiones, ya que miden el campo magnético en dos direcciones ortogonales, que solamente se pueden usar en entornos llanos. Si el entorno no es llano, es necesario usar una brújula de tres dimensiones y dos inclinómetros para poder saber la orientación del robot. La brújula medirá el campo en tres ejes ortogonales y los inclinómetros medirán la inclinación del robot en los dos ejes horizontales.

Giróscopo: Son brújulas de medida incremental, es decir, miden cambios en la orientación del robot. Se basan en medir la aceleración usando las leyes de Newton (figura 2.45).

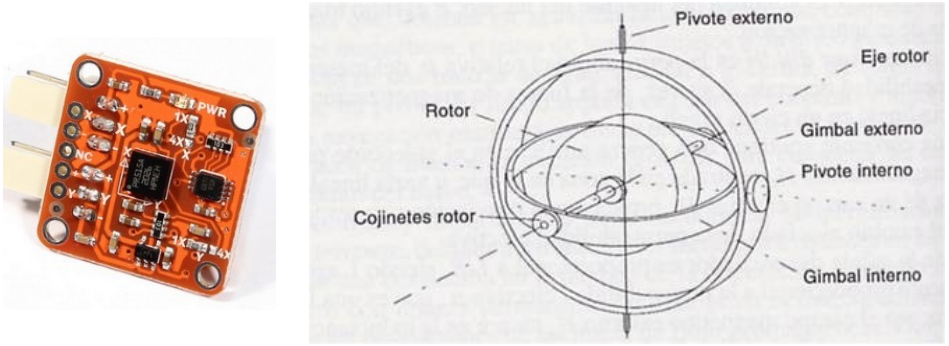

Figura 2.45: Giróscopo

Los giróscopos tienen varias limitaciones:

- Acumulan el error con el paso del tiempo.
- Si se requiere de precisiones aceptables (décimas de grado por segundo) son muy caros.
- Los giróscopos baratos tienen errores mayores de un grado por minuto.

Existen giróscopos basados en láser, que permiten medir con mucha precisión la orientación del robot (errores desde varios grados por hora hasta 0.0001 grado por hora).

- *Encoders o codificadores*

  Los codificadores ópticos o encoders incrementales (figura 2.46) constan en su forma más simple de un disco transparente con una serie de marcas opacas colocadas radialmente y equidistantes entre sí, de un sistema de iluminación en el que la luz es colimada de forma adecuada, y de un elemento fotorreceptor. El eje cuya posición se quiere medir va acoplado al disco transparente. Con esta disposición, a medida que el eje gire se irán generando pulsos en el receptor cada vez que la luz atraviese cada marca, y llevando una cuenta de estos pulsos es posible conocer la posición del eje y la velocidad de rotación.

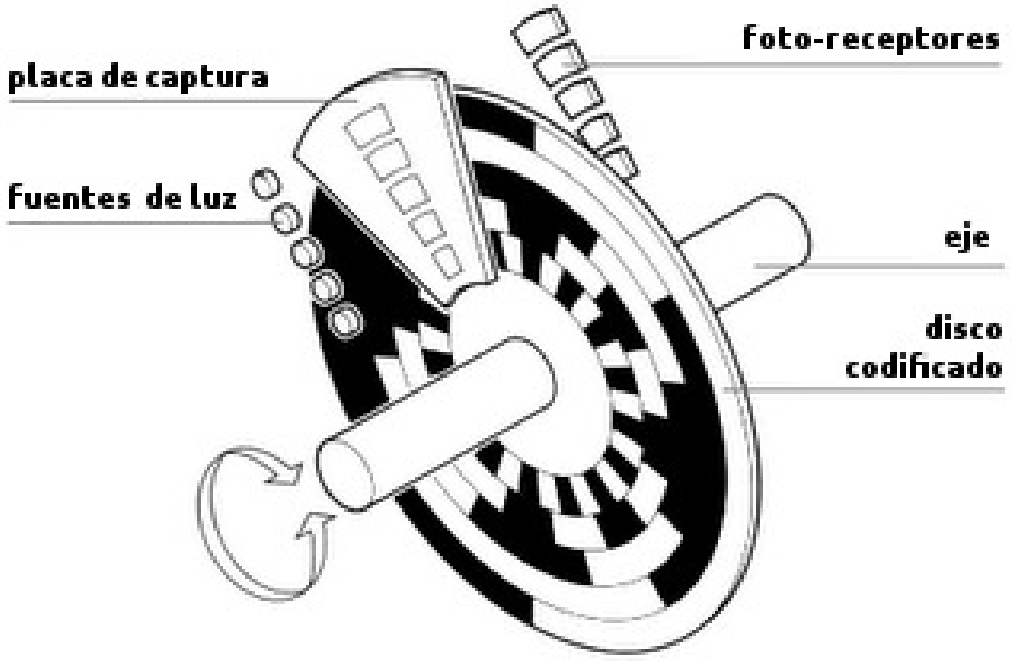

Figura 2.46: Codificador (encoder)

Por otro lado, las principales limitaciones con las que cuenta son:

- Siempre es necesario un circuito contador para obtener una salida digital compatible con el puerto de entrada/salida de un microcontrolador. Otra forma posible de realizarlo se basaría en software especial según sea la aplicación específica, como por ejemplo, alguna interrupción o programación de alta velocidad, en tiempo real, para obtener el tiempo de cambio entre un sector y otro.
- Los encoders pueden presentar problemas mecánicos debido a la gran precisión que se debe tener en su fabricación. La contaminación ambiental puede ser una fuente de interferencias en la transmisión óptica. Son dispositivos particularmente sensibles a golpes y vibraciones, estando su margen de temperatura de trabajo limitado por la presencia de componentes electrónicos.

Para incrementar la resolución del encoder y saber el sentido de giro se dispone de otra señal pero desplazada de la existente, de manera que el tren de pulsos B que se genere esté desplazado 90º eléctricos con respecto al generado por la primera franja, A. De esta manera, con un circuito relativamente sencillo, es posible obtener una señal adicional que indique cuál es el sentido de giro. Es necesario además disponer de una marca de referencia Z sobre el disco que indique que se ha dado la vuelta completa y que, por tanto, se ha de empezar de nuevo la cuenta.

Mientras que en los encoders incrementales la posición está determinada por el cómputo del número de impulsos con respecto a la marca de cero, en los encoders absolutos la posición queda determinada mediante la lectura del código de salida, el cual es único para cada una de las posiciones dentro de la vuelta.

La resolución de este tipo de sensores depende directamente del número de marcas que se pueden poner físicamente en el disco. El modelo de encoders más utilizado son los ópticos, que constan de diferentes sectores que pueden ser opacos o transparentes. Otro tipo también muy usado es el magnético, el cual está equipado con un sistema de detección magnética sin contacto. La variación de campo magnético provocada por los dientes de una rueda de medida produce una onda senoidal en una señal de onda cuadrada, triangular, etc. Las ventajas principales que presentan frente a los ópticos son la resistencia a los golpes, vibraciones o polvo y su funcionamiento en un rango de temperaturas más amplio. Los encoders magnéticos son los más precisos.

Tipos de actuadores

A continuación se describen algunos de los actuadores más comúnmente utilizados en robótica [12].

- *Alambres musculares*

  Los alambres musculares, representados en la figura 2.47 en apariencia son similares a un alambre común pero que se contraen al activarlos con corriente eléctrica.o.

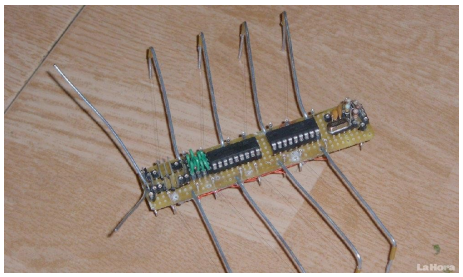

Figura 2.47: Robot con alambres musculares

  Estos alambres pueden ser empleados en la disposición adecuada para accionar o mover elementos mecánicos.

- *Motores paso a paso.*

  Son dispositivos electromagnéticos que convierten pulsos eléctricos en movimientos mecánicos (figura 2.48). El eje del motor paso a paso gira con incrementos discretos (pasos). Cuando los pulsos eléctricos son aplicados en la secuencia apropiada el eje del motor girará en el sentido horario o antihorario, dependiendo de la secuencia.

Figura 2.48: Motor paso a paso

- *Motores de corriente continua*

  Son los más comunes y económicos. Constan por lo general de dos imanes permanentes fijados en la carcasa y una serie de bobinados de cobre ubicados en el eje del motor (figura 2.49). Son eficientes para girar con poco par y gran velocidad. Son los más simples en cuanto a motores, fáciles de conseguir y baratos. Para poder controlarlos en ambos sentidos se utiliza un puente en H.

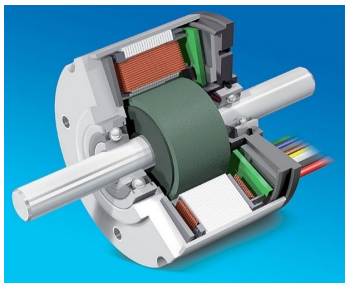

Figura 2.49: Motor sin escobillas (brushless)

  Mientras que los motores de corriente continua más comunes son con escobillas, se han desarrollado motores más eficientes sin escobillas (brushless) que no necesitan de estas para realizar el cambio de polaridad en el rotor.

- *Servomotores*

  Un servomotor es un motor que puede poner el eje en una determinada posición a través de una señal eléctrica de control (figura 2.50). De esta manera, modificando el valor de la señal, el servo se puede posicionar en cualquier ángulo en un rango de 0º a 180º. El servomotor se utiliza normalmente a la hora de posicionar elementos mecánicos.

Figura 2.50: Servomotor

Capítulo 3: Materiales y métodos

3.1 Materiales

A continuación se detallan los materiales que se han empleado en este proyecto.

3.1.1 Sistema Operativo de Robots (ROS)

Introducción

Robot Operating System (ROS) [22] [13] es una plataforma software, lo que se conoce como framework, para el desarrollo de software específico para robótica. Fue creado por el instituto de investigación Willow Garage en 2007 bajo licencia BSD y todos sus programas son de código abierto, permitiéndose así su uso gratuito tanto para la investigación como para fines comerciales.

Su filosofía es la de que todos los programas sean de uso libre, reutilizables, escalables según la aplicación deseada e integrables con todo el software de robótica existente y futuro. Por ello el sistema se creó teniendo en cuenta otras plataformas software abiertas ya existentes, como OpenCV, Player/Stage, Gazebo, Orocos/KDL y otros, existiendo paquetes para su uso en ROS.

ROS se compone de un conjunto de librerías de programación, aplicaciones, drivers y herramientas de visualización, monitorización, simulación y análisis, todas reutilizables para el desarrollo de nuevas aplicaciones para robots tanto simulados como reales.

Por otro lado, ROS nos proporciona también servicios estándares propios de un sistema operativo pero sin serlo, ya que se instala sobre otro, en general Linux y de manera recomendada Ubuntu, y por ello también recibe la denominación de Meta-Sistema Operativo. Posee su propio manejador de paquetes mediante comandos desde terminal para gestión, compilación y ejecución de archivos, así como abstracción de hardware.

Su estructura es modular, de forma que cada programa o aplicación, lo que en ROS se denomina como **nodo**, se ejecuta dentro de otro ejecutable (**Master**) que funciona de núcleo del sistema y hace las veces de plataforma por la que se comunican los diferentes nodos mediante tópicos y/o servicios, de los que se hablará más adelante.

Para comprobar que ROS representa un sistema completo de desarrollo de software para robots, basta con observar el trabajo realizado con el robot PR2, mostrado en la figura 3.1, cuyo sistema software ha sido desarrollado por Willow Garage únicamente usando ROS. Además, todas las capacidades del PR2 (visión, movimientos de la base y los brazos...) están disponibles vía ROS, por lo que podemos utilizar y programar el robot a través de ROS.

Figura 3.1: Robot PR2

Esto último hace que algunos paquetes de ROS sean en principio específicos para este robot, pero una gran parte de ellos pueden adaptarse y utilizarse a cualquier otro robot real o simulador, siempre que se cumplan con las interfaces necesarias.

El paquete *-ros-pkg es un repositorio común para el fácil desarrollo de librerías de alto nivel. Muchas de las capacidades asociadas a ROS, como la librería de navegación y el visor Rviz, están desarrolladas en este repositorio. Estas librerías brindan una serie de potentes herramientas para trabajar con ROS de forma sencilla, sabiendo que está ocurriendo en cada momento. Una de las herramientas más importantes para ello son las de visualización, simulación y depuración, tal como se muestra en la figura 3.2.

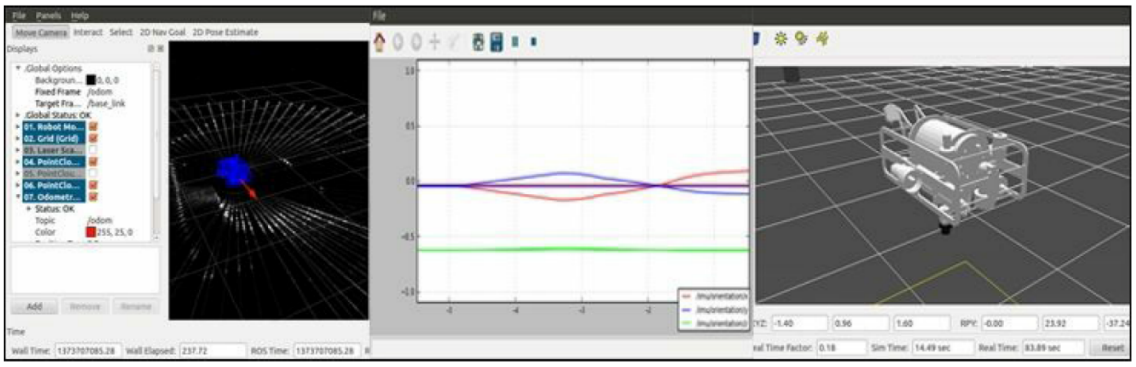

Figura 3.2: Entorno de simulación Gazebo y herramineta de visualización Rviz

La mayor ventaja de este sistema es que ROS promueve la reutilización de código para que los científicos y desarrolladores no tengan que crear los mismos programas ya hechos otra vez. Otra de las ventajas reside en poder obtener código del repositorio, mejorarlo y compartirlo otra vez.

Arquitectura

La arquitectura de ROS está diseñada y dividida en tres secciones o niveles:

- El nivel de sistema de archivos.

- El nivel de cómputo.

- El nivel de Comunidad.

El primer nivel es el sistema de archivos. En este nivel, una serie de conceptos explican cómo está formado internamente ROS, la estructura de carpetas y el mínimo de archivos necesarios para funcionar.

El segundo nivel es el gráfico de cómputo, donde se establece la comunicación entre procesos y sistemas.

El tercer nivel es la Comunidad, en la que se encuentran los códigos desarrollados por los usuarios de ROS. Este nivel es la forma en la que ROS puede crecer rápidamente.

- ***Nivel de sistema de archivos***

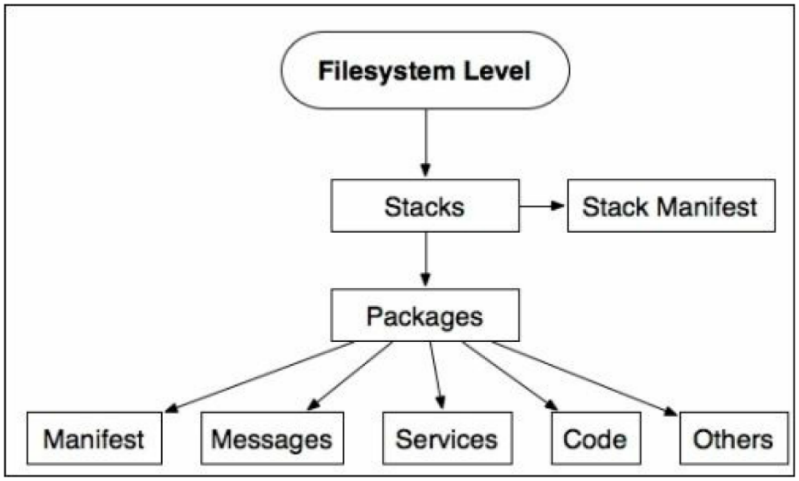

Figura 3.3: Nivel de sistema de archivos

Parecido a un sistema operativo (figura 3.3), un programa de ROS está dividido en carpetas, y estas carpetas contienen archivos que describen sus funciones:

- **Paquetes (*packages*)**: Son el nivel más bajo de organización del sistema de archivos de ROS. Pueden contener cualquier cosa: ejecutables (nodos), modelos, tipos de mensajes y servicios, herramientas o librerías (figura 3.4).

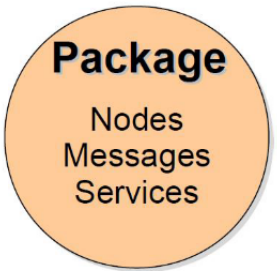

Figura 3.4: Paquetes

- **Manifiestos**: Los manifiestos proporcionan información sobre los paquetes, licencias, dependencias, interrupciones, etc. Los manifiestos se encuentran en el archivo *manifests.xml*.
- **Pilas (*stacks*)**: Son agrupaciones de paquetes que forman una librería de alto nivel. Cada pila agrupa paquetes que son complementarios entre ellos (figura 3.5).

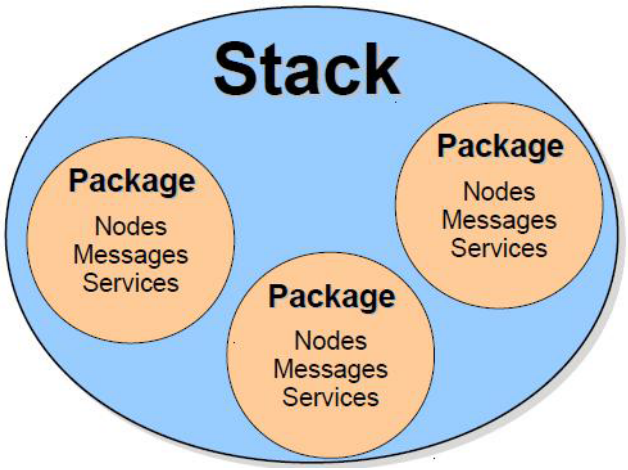

Figura 3.5: Pilas (stacks)

- **Manifiestos de Stacks**: Los manifiestos de Stacks (*stack.xml*) proporcionan información sobre los stacks, incluyendo sus licencias y las dependencias de otros Stacks.

- **Tipos de mensajes**: Un mensaje es la información que un proceso envía a otro. ROS tiene muchos tipos de mensajes estándar. Los mensajes se guardan en el archivo *my_package/msg/MyMessageType.msg*.

- **Tipos de servicios**: Las descripciones de los servicios definen las estructuras de las solicitudes y respuestas para los servicios en ROS. Se almacenan en *my_package/srv/MyServiceType.srv*.

Tanto los paquetes como las pilas poseen unos archivos de información especificando lo que contienen, su funcionalidad y los paquetes de los que depende. En las pilas éste archivo se denomina *stack.xml* y en los paquetes *manifiest.xml*..

A la hora de ser descargados e instalados, las pilas se agrupan en repositorios tal y como se muestra en la figura 3.6, que son equivalentes a los repositorios de Linux. De hecho, los repositorios de ROS pueden ser descargados equivalentemente a los de Linux, directamente por línea de comandos o mediante el administrador de descargas.

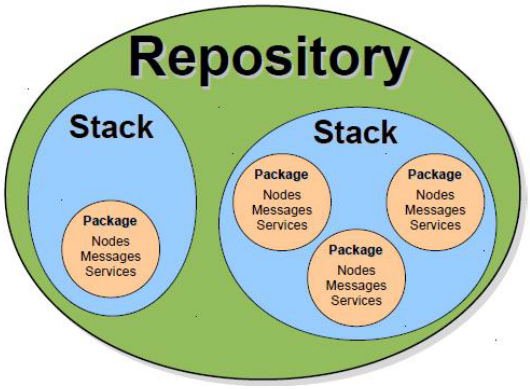

Figura 3.6: Repositorios

Por defecto, todo el sistema de ROS se instala en la dirección */opt/ros* estando restringidos los derechos de modificación de sus paquetes para evitar dañar el sistema. Por ello, en general y en el caso de este proyecto, se crea un entorno de trabajo (*hydro_workspace*) para crear nuestras aplicaciones en ella, que se añade al path de ROS para que éste tenga acceso.

Para entender cómo se organizan los paquetes de las aplicaciones, será conveniente explicar los tipos de carpetas y archivos que podemos encontrar normalmente, sin entrar en detalle en cada uno:

- Carpeta *bin*: contiene los ejecutables del paquete, los que se podrán lanzar como nodos.
- Carpeta *build*: contiene los archivos de compilación internos del paquete.
- Carpeta *src*: contiene los códigos de los programas (.cpp), que al compilarlos crearán ejecutables en *bin*.
- Carpeta *include*: contiene los archivos de cabeceras (.h) propios de los programas del paquete.
- Carpeta *lib*: contiene los archivos de librerías (.lib, .so).
- Carpeta *launch*: contiene los archivos de lanzamiento (.launch) de aplicaciones completas.
- Carpeta *yaml*: contiene los archivos de lista de parámetros (.yaml).
- Carpeta *msg*: contiene los tipos de mensajes (.msg) definidos en el paquete.
- Carpeta *msg_gen*: contiene los archivos de cabeceras (.h) auto-generadas al compilar los tipos de mensajes definidos en el paquete.

- Carpeta *srv*: contiene los tipos de mensajes de los servicios definidos.
- Carpeta *srv_gen*: contiene los archivos de cabeceras (.h) auto-generadas al compilar los tipos de servicios definidos en el paquete.
- Archivo *CMakeList.txt*: es una lista en la que se especifica al compilador qué debe compilar de nuestro paquete: nodos, mensajes, servicios, librerías, etc.
- Archivo *manifest.xml*: contiene una descripción del paquete (función, auto, licencia, etc.) y una lista con los paquetes de los que depende.

- ***Nivel gráfico de cómputo***

  ROS crea una red donde todos los procesos están conectados (figura 3.7). Cada nodo en el sistema puede acceder a esta red, interactuar con otros nodos, ver información que están enviando y transmitir datos a la red.

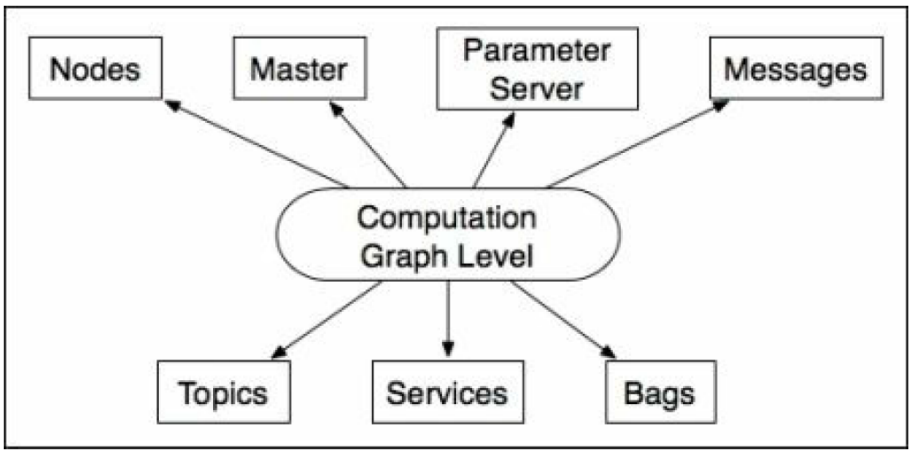

Figura 3.7: Nivel de gráfico de cómputo

Los conceptos básicos en este nivel son los nodos, el Maestro, el Servidor de Parámetros, los mensajes, los servicios, los tópicos y las bolsas, todos ellos proporcionan información al gráfico de diferentes maneras:

- **Nodos**: Los nodos son procesos donde se realizan las operaciones. Para que un proceso interactúe con otros nodos, es necesario crear un nodo con este proceso para conectarlos a la red de ROS. Normalmente, un sistema tendrá muchos nodos para controlar las diferentes funciones. Es mejor tener muchos nodos que proporcionen una única función a tener un nodo extenso que realice todas las tareas en el sistema. Los nodos se forman con una librería cliente de ROS, como por ejemplo *roscpp* o *rospy*.

- **Maestro**: El Maestro proporciona registros de nombres y vigilancia para el resto de los nodos. Si no existe un Maestro en el sistema, no habrá comunicación con nodos, servicios, mensajes y otros. Es posible tener un Maestro en un ordenador y trabajar con los nodos en un ordenador distinto.
- **Servidor de parámetros**: El Servidor de Parámetros brinda la posibilidad de almacenar datos utilizando llaves en una localización central. Mediante este parámetro es posible configurar los nodos mientras se ejecutan o cambiar el funcionamiento de los mismos.
- **Mensajes**: Los nodos se comunican entre sí a través de los mensajes. Un mensaje contiene datos que envían información a otros nodos. ROS posee muchos tipos de mensajes y se pueden desarrollar tipos de mensajes propios.
- **Tópicos**: Cada mensaje debe tener un nombre en la red de ROS para poder enlazarlo. Cuando un nodo envía datos, se dice que está publicando un tópico. Los nodos pueden recibir tópicos de otros nodos simplemente suscribiéndose al tópico. Un nodo se puede subscribir a un tópico sin ser necesario que el tópico exista. Esto permite desacoplar el problema de la producción con la demanda. Es importante que el nombre sea único para evitar problemas entre tópicos de igual nombre.
- **Servicios**: Cuando se publica un tópico, se envían datos pero si se quiere recibir una respuesta, no se puede hacer mediante tópicos. Los servicios brindan la posibilidad de interactuar con nodos. Además, los servicios deben tener un nombre único. Cuando un nodo tiene un servicio, todos los nodos pueden comunicarse con él, gracias a las librerías cliente de ROS.
- **Bolsas**: Las bolsas (bags) son un formato para guardar y reproducir la información de los mensajes de ROS. Son un mecanismo muy importante para almacenar datos, por ejemplo de los sensores.

- ***Nivel de Comunidad***

  El nivel de la Comunidad de ROS son los recursos que proporcionan los grupos para el intercambio de software y conocimientos. Estos recursos incluyen:

  - **Distribuciones**: Las distribuciones de ROS son una colección de stacks que se pueden instalar. Hacen que sea más fácil instalar software y proporcionan mantenimiento constante de las versiones.
  - **Repositorios**: ROS depende de una red de repositorios de código, donde diferentes instituciones pueden desarrollar y publicar sus propios componentes y software para robots.

- **La Wiki de ROS**: La Wiki de ROS es el foro principal para documentar la información sobre ROS. Cualquiera puede registrarse y contribuir con sus propios documentos, corrigiendo, actualizando, creando tutoriales, etc.
- **Lista de correo**: La lista de correo de ROS es el principal canal de comunicación para nuevas actualizaciones, así como un foro de preguntas sobre el software relacionado.

3.1.2 Robot móvil Summit

Introducción

En este capítulo se describen los componentes principales del robot móvil Summit, así como sus conexiones. Cada pieza incluye una pequeña descripción de los componentes mecánicos que la forman, enfatizando los elementos que necesitan un control periódico y mantenimiento [21]. Este robot móvil es producto de Robotnik© [20], una empresa compuesto por un equipo de ingenieros multidisciplinar dedicada al desarrollo de productos en el área de la Robótica de Servicio, cuya actividad se centra en el diseño, fabricación y comercialización de productos de robótica.

Elementos externos

Las figuras 3.8 y 3.9 muestran las partes principales del robot:

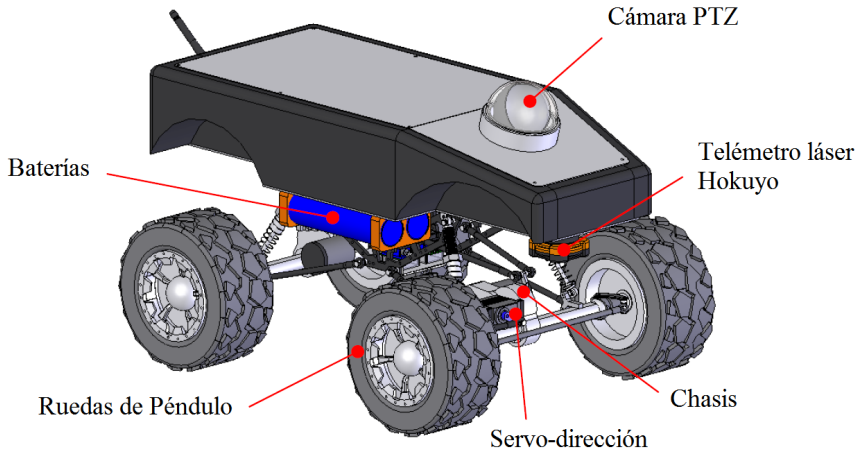

Figura 3.8: Partes principales del robot Summit

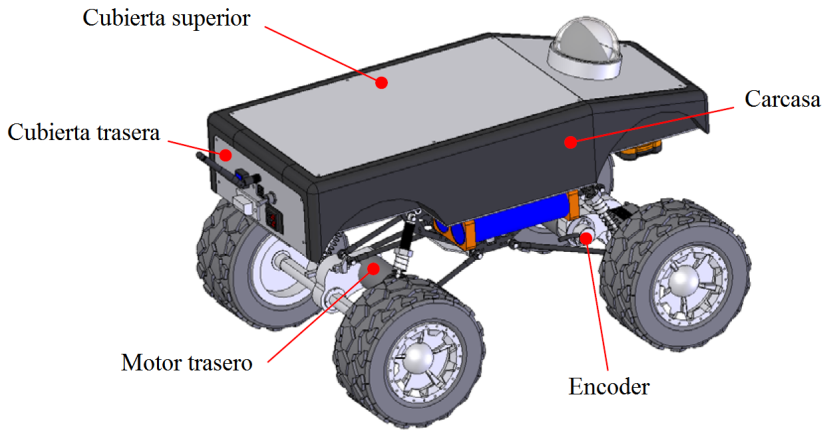

Figura 3.9: Partes principales del robot Summit, vista trasera

Las partes principales del robot son:

- *Carcasa*: Fabricada mediante CNC, sostiene las cubiertas superior y trasera. Los componentes eléctricos se colocan dentro. Solo los drivers de los motores están fuera.
- *Cubierta superior*: Puede ser extraída fácilmente para acceder al interior del robot, donde están situados algunos componentes de control, como el ordenador.
- *Cubierta trasera*: Es donde se encuentra el panel de control, los botones y las antenas.
- *Ruedas de péndulo*: Ruedas de aluminio con un péndulo de pesas extraíble.
- *Baterías*: 4 LiFePo4 3.2 V 16 Ah por celda.
- *Chasis*: Chasis de aluminio con fuertes amortiguadores.
- *Motores*: Dos motores bushless, uno en cada eje.
- *Servo-dirección*: Dos servos de 38 Kg/cm, uno en cada eje.
- *Cámara PTZ*: Cámara con micrófono y protegida por una cúpula extraíble.
- *Telémetro láser*: Se pueden montar distintos modelos. Se dispone de un telémetro RPlidar de RoboPeak© montado en la cubierta superior.

El robot requiere muy poco mantenimiento y está centrado en el área inferior (chasis y ruedas). Casi todos los tornillos poseen roscas de sellado para evitar su pérdida, pero deben ser revisados de vez en cuando, especialmente después de una conducción agresiva.

- ***Ruedas de péndulo, neumáticos y espumas***

  El bloque del neumático posee un péndulo de pesas extraíble (figura 3.10) que se apoya en unos rodamientos. Como el peso siempre está en el punto más bajo de la rueda, esto ayuda a mantener el robot estable cuando la inclinación es elevada. El peso puede ser fácilmente ajustado o eliminado de la rueda.

Figura 3.10: Péndulo y pesas extraíbles

  Los neumáticos se basan en una goma avanzada ultra suave con un compuesto súper pegajoso, que contiene espuma dura en su interior (figura 3.11). Se recomienda encarecidamente no dejar peso sobre las ruedas durante un largo periodo de tiempo. Esta situación puede conducir a que los neumáticos y la espuma se aplanen. El problema se soluciona normalmente haciendo rodar el robot durante un rato.

Figura 3.11: Espuma rígida extra-dura

- ***Chasis de aluminio***

Figura 3.12: Chasis y ruedas

El chasis (ver figura 3.12) cuenta con sellador o tuercas en todos sus tornillos para evitar su pérdida. Se recomienda encarecidamente revisarlos de forma regular. Se deben verificar después de largos periodos de conducción o en el exterior, especialmente en terrenos duros.

Todos los tornillos están medidos y la mayoría son de métrica M3.

- ***Servo-dirección***

Figura 3.13: Servomotor

Los servos que utiliza el robot son HITECH HS-7980TH con 38Kg/cm de par a 6V, como el mostrado en la figura 3.13. Es un servo para robots de alta calidad con los engranajes de titanio y rodamientos de bolas duales.

Por defecto, solo hay un servo montado en el eje frontal para controlar la dirección del robot, pero la tarjeta de control está preparada para manejar otro. El radio de giro se puede reducir a la mitad si se añade un servo adicional en el eje trasero. El robot Summit que se ha utilizado para el proyecto cuenta con dos servos, uno en cada eje, permitiendo el giro de ambos.

- ***Motores***

El robot Summit posee dos potentes motores brushless de 4 polos. Este tipo de motor tiene más tiempo de vida y mayor eficiencia que los motores con escobillas. Se pueden montar varias marcas con características similares, como Leopard LBP4074, Turnigy XK4074 o K4465- E (1050 kv). En las figuras siguientes (ver 3.14 y 3.15) y en la tabla 3.1 se muestran las características de los motores.

Figura 3.14: Motores

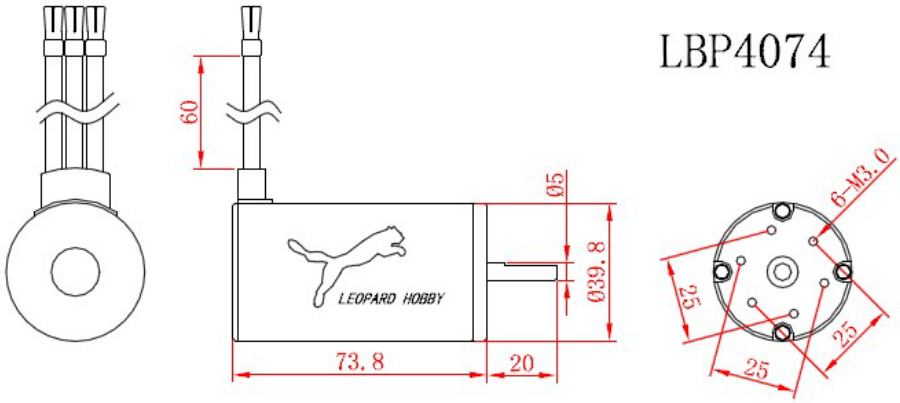

Figura 3.15: Dimensiones del motor

Modelo	4074/3Y
Intensidad (max)	80 A
Voltaje entrada (max)	22.2 V
Potencia (max)	1775 W
KV (Rpm/V)	1400 KV
Resistencia	0.0140
Corriente en vacío	1.2 A
Diámetro x longitud	F39.8 x 73.8 mm
Profundidad del agujero	7 mm
Longitud del eje	20 mm
Diámetro del eje	5.0 mm
Peso	358 g

Tabla 3.1: Características de los motores

- Drivers de los motores

Figura 3.16: Drivers de los motores y programador Castle Link

Los drivers de los motores son dos Castle Creations Mamba Monster ESCs (Electronic Speed Controller) (ver figura 3.16). Estos poderosos drivers pueden proporcionar una corriente continua de hasta 120 A y soportar una tensión de entrada de hasta 25 V.

Los drivers están programados y calibrados por Robotnik©. Se pueden recuperar las características de fábrica con el programador Castle Link que se proporciona y el archivo de configuración ESC_mod_setup.dat que contiene el CD.

Hay tres indicadores led en el driver:

- Verde: Adelante.
- Amarillo: Neutro.
- Rojo: Atrás.

Es necesaria una señal de aceleración neutral unos segundos al inicio. Se escucharán una serie de tonos y dos pitidos finales cuando los drivers estén armados. El motor no se moverá si su driver no está armado.

Si el led amarillo está parpadeando significa que el driver no está armado. Al inicio todas las luces parpadean juntas.

El driver necesita un estado neutro en los cambios de velocidad de atrás hacia adelante y viceversa. Esto se produce para no desgastar los motores y los engranajes. Si se cambia la referencia sin detener el robot, el motor dejará de moverse.

- ***Encoder magnético***

  El RM22 (figura 3.17) es un codificador magnético de rotación compacto y de alta velocidad (hasta 300 rpm), diseñado para usos en entornos difíciles. Las dos partes están diseñadas para no hacer contacto y evitar el uso de juntas o cojinetes.

Figura 3.17: Codificador magnético

El robot está equipado con un codificador que puede ser puesto en cualquiera de los ejes. Como opción, se puede colocar un codificador adicional en el otro eje. La placa de control AGVCTRL-V2 está preparada para manejar un segundo codificador, por lo que ambos motores se pueden controlar en lazo cerrado.

El codificador se encuentra en el extremo del eje del motor, separado de éste por una caja de engranajes reductora (ver figura 3.18). Gira 64.53 veces más rápido que la rueda.

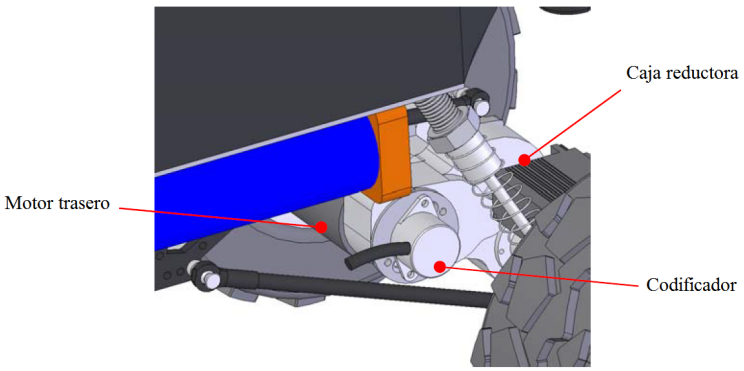

Figura 3.18: Posición del codificador

- ***Cámara PTZ***

Figura 3.19: Cámara esférica Logitech

El robot dispone de una cámara PTZ con movimiento vertical y horizontal (figura 3.19), colocada en la parte frontal.

Especificaciones técnicas:

- Movimiento motorizado (189º horizontal y 102º vertical).
- Óptica Carl Zeiss©.
- Sistema de enfoque automático.
- Resolución ultra alta de 2 megapíxeles con sensor RightLight$^{(TM)}$ 2 Technology.
- Profundidad de color: 24 bit de color verdadero.
- Captura de vídeo: Hasta 1600 x 1200 píxeles (calidad HD).
- Velocidad del obturador: Hasta 30 marcos por segundo.
- USB 2.0.

La cámara se encuentra en una carcasa al aire libre cubierta por una cúpula extraíble.

- ***Panel de control***

Figura 3.20: Panel de control

En la parte trasera del robot (ver figura 3.20) se muestran una serie de botones, indicadores y conexiones:

- Interruptor de la CPU (botón azul): Enciende y apaga el PC empotrado.
- Interruptor general ON/OFF: Corta la alimentación de todo el robot.
- 12 V DC: Preparado para alimentar dispositivos externos y protegido con un fusible de 2 A.
- Cargador: Para conectar el cargador de batería correspondiente.
- Dos puertos USB.
- Antena Wi-Fi.
- Indicador led de alimentación.

Elementos internos

La figura 3.21 muestra otros componentes no mecánicos a tener en consideración para las tareas de mantenimiento.

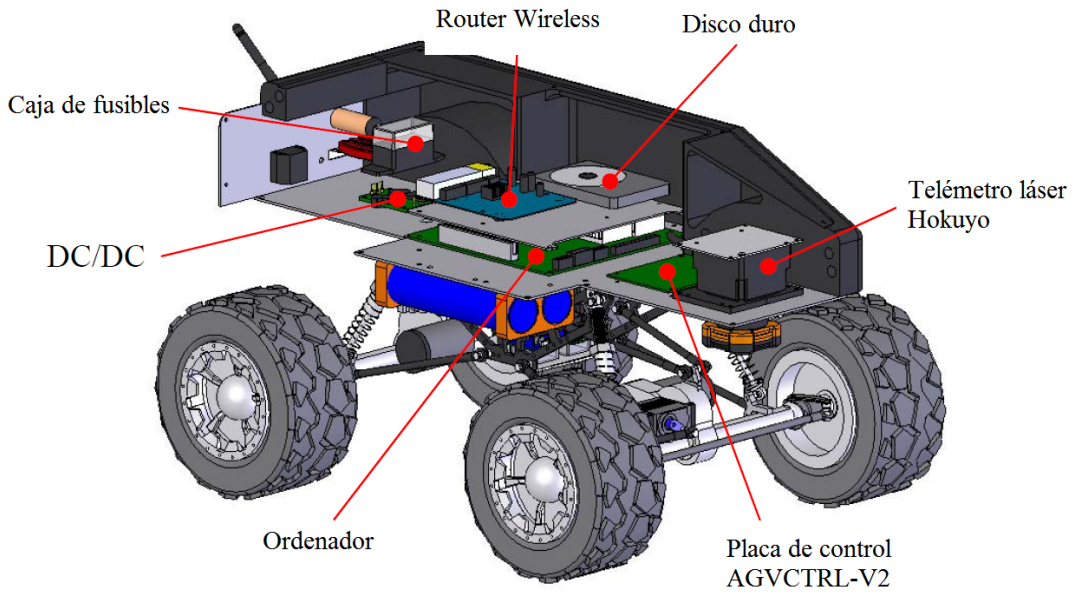

Figura 3.21: Vista interior del robot Summit

- ***PC empotrado***

  La placa principal es una Intel DN2800MT (figura 3.22) con un factor de forma mini ITX, por lo que puede ser reemplazado fácilmente por otra placa si fuera necesario. El equipo se completa con 4 GB de memoria RAM y un disco duro SATA de 2.5". Las características de este PC se muestran en la tabla 3.2.

Figura 3.22: PC empotrado

Factor de forma	Mini-ITX (170.18 mm x 170.18 mm)
Procesador	Refrigerado con ventilador, procesador soldado Dual-Core Intel©Atom^{TM} con gráficos y controlador de memoria integrados
Memoria	• Dos pines 204 DDR3 SDRAM SO-DIMM (Small Outline Dual Inline Memory Module) • Soporte para DDR3 800 MHz y DDR3 1066 MHz SO-DIMMs • Soporte para hasta 4 GB de memoria de sistema en una única SO-DIMM
Circuito integrado	Intel©NM10 Express Chipset

Gráficos	• Gráficos integrados:
	  ◦ Indicadores digitales (HDMI)
	  ◦ Indicadores analógicos (VGA)
	  ◦ Pantallas planas internas:
	  ◇ LVDS
	  ◇ eDP (Puerto indicador empotrado)
	• Gráficos externos:
	  ◦ Una PCI Express 1.0a x1 con trajeta conectora de gráficos

Audio	• 2 + 2 Intel©High Definition (Intel©HD) audio mediante el códec de sonido Realtek ALC888S
	  ◦ Salida estéreo analógica (conector Jack trasero)
	  ◦ Altavoces estéreo incluidos en el chasis (3 W/3 Ω)
	  ◦ Salida digital de audio S/PDIF (cabezal interno)
	  ◦ Entrada digital de micrófono DMIC (cabezal interno)
	  ◦ Conexión analógica (Jack en el panel trasero)
	  ◦ Panel frontal Intel©HD Audio/AC'97 con soporte de micro y cascos (cabezal interno)
	• Intel©High Definition Audio de 8 canales mediante interfaz HDMI

Interfaces periféricas	• Diez puertos USB:
	  ◦ Cuatro puertos en el panel frontal
	  ◦ Dos puertos están implementados con conectores en el panel de control (negro)
	  ◦ Dos puertos de alta corriente/carga rápida implementados a través del panel de control (amarillo)
	  ◦ Un puerto implementado en el PCI Express Half-Mini
	  ◦ Un puerto implementado en el PCI Express Full/Half-Mini
	• Dos puertos SATA:
	  ◦ Un puerto interno SATA (negro)
	  ◦ Un puerto interno SATA (multiplexado con mSATA, enrutado hacia PCI Express Full/Half-Mini) (gris)

Capacidad de expansión	• Un conector PCI Express 1.0a x1 • Un espacio para PCI Express Half-Mini • Un espacio para PCI Express Full/Half-Mini
BIOS	• La BIOS de Intel©reside en el dispositivo de interfaz periférica serial (SPI) • Soporte para Configuración Avanzada e Interfaz de Alimentación (acpi), Plug and Play y Sistema de Gestión de la BIOS (SMBIOS)
Soporte LAN	Subsistema LAN Gigabit (10/100/1000 Mbits/s) que utiliza el controlador Ethernet Intel©82574L Gigabit
Control E/S Legacy	Controlador para gestión de hardware Nuvoton W83627DHG, con soporte de puerto serie y paralelo
Subsistema de monitorización	• Monitorización del hardware a través del controlador Nuvoton • Detección de voltaje fuera de rango en la alimentación • Detección de temperatura fuera de límites • Sistema de ventiladores • Sensor de ventilación para monitorizar la actividad • Control de velocidad del ventilador

Tabla 3.2: Caracteristicas del PC DN2800MT

El PC empotrado tiene incorporado el sistema operativo Linux y se encuentra colocado en el medio del robot, debajo del router *wireless* y el disco duro. Su mantenimiento es el equivalente al de un ordenador estándar.

El mayor problema puede ser la acumulación de polvo en los componentes internos, ya que que actúa como aislante térmico. El calor generado por los componentes no puede disiparse bien por culpa del polvo acumulado.

Las partículas de aceite y grasa en el ambiente se mezclan con el polvo, creando así una capa de aislamiento que refleja el calor a otros componentes. Este efecto provoca que la vida útil del sistema disminuya. Por otro lado, el polvo contiene partículas conductoras que pueden generar cortocircuitos a través de las placas o en las conexiones.

La mejor manera de extender la vida de los equipos y no tener que repararlos durante muchos años es limpiar el polvo con frecuencia.

- ***Router Wifi***

Figura 3.23: Router SL-R7205 11N 300M (OpenWRT)

El robot Summit se comunica con un ordenador remoto a través de una señal WiFi generada por el propio robot a partir del router SL-R7205 11N 300M (figura 3.23).

Especificaciones técnicas:

- Interfaz LAN: 4 puertos 10/100 Mbps RJ-45.
- Radio de transferencia de datos Wireless 802 11n: 300 Mbps.
- Frecuencia de operación: 2.4-2.4835 GHz.
- Potencia de salida del transmisor: 20 dBm (EIRP).
- Antena: 5 dBi omnidireccional.
- Protocolo estándar: IEEE 802.11N Draft 2.

- ***Conversor DC/DC***

Para proporcionar una fuente de alimentación estable de 12 V DC para los elementos electrónicos (PC, Router WiFi, AGVCTRL-V2,...), el robot está equipado con un conversor DC/DC de 12 V y 8 A (figura 3.24).

La entrada tiene una protección extra con un diodo y un gran condensador para prevenir paradas cuando los drivers de los motores exigen altos picos de corriente de las baterías. Se localiza bajo la caja de fusibles.

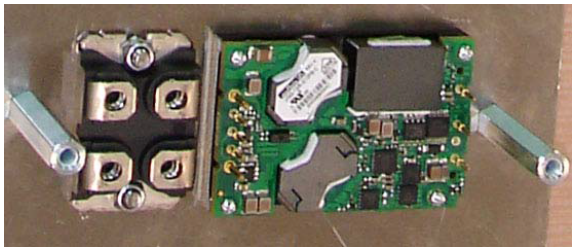

Figura 3.24: Conversor DC/DC y diodo

- ***Caja de fusibles***

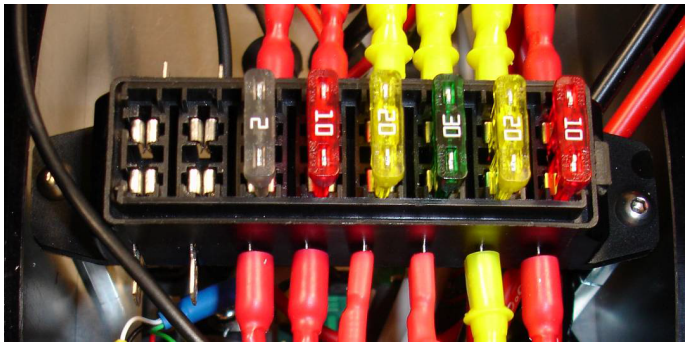

Figura 3.25: Caja de fusibles

La imagen superior (ver figura 3.25) muestra la caja de fusibles interna que contiene los siguientes fusibles:

- Fusible F1: 10 A (rojo). Conector de carga del panel trasero.
- Fusible F2: 20 A (amarillo). Driver frontal, motor y servo-dirección.
- Fusible F3: 30 A (verde). Fusible principal, conectado a la batería a traves del interruptor S1.
- Fusible F4: 20 A (amarillo). Driver trasero, motor y servo-dirección, si estuvieran presentes.
- Fusible F5: 10 A (rojo). Conversor DC/DC (+12 V).
- Fusible F6: 2 A (gris). Salida de alimentación de 12 V del panel trasero (desde el conversor DC/DC).

- ***Placa de control AGVCTRL-V2***

La placa de control está conectada al PC por un puerto serie y recibe comandos del mismo a 40 Hz. Si no se reciben comandos durante un segundo, la placa AGVCTRL-V2 (figura 3.26) detendrá el robot de forma inmediata.

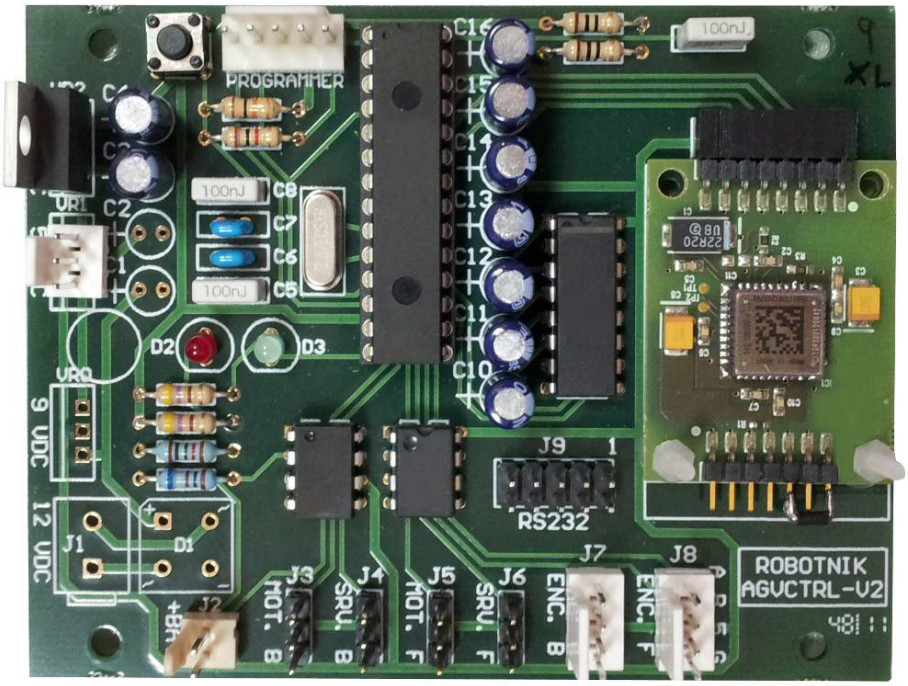

Figura 3.26: Placa AGVCTRL-V2

Conexiones:

- 5 VDC: Para alimentar la placa.
- BAT: Para medir la tensión de la batería.
- MOT. B: Driver del motor trasero.
- SRV. B: Servo-dirección trasera (incluida).
- MOT. F: Driver del motor delantero.
- SRV. F: Servo-dirección delantera.
- ENC. B: Codificador trasero (no se utiliza).
- ENC. F: Codificador delantero.
- RS232: Puerto serie conectado al PC.
- PROGRAMMER: Únicamente para uso de Robotnik©.

Se puede ver el botón de RESET en la esquina superior izquierda, dos leds indicadores y un sensor digital de velocidad angular. El led rojo se encontrará encendido cuando el conector de 5 VDC esté alimentado y el led verde deberá parpadear a 1 Hz cuando esté en funcionamiento normal.

Control manual (*Gamepad*)

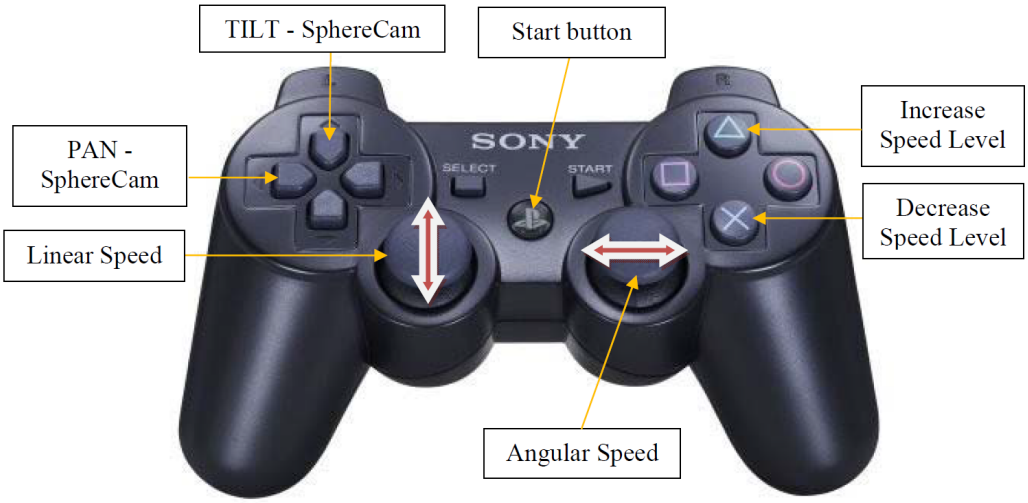

Figura 3.27: Controlador Dualshock

El *Gamepad* usado para el control manual del robot Summit es un mando *Dualshock* de Play Station 3 © con un receptor Bluetooth (figura 3.27). El receptor puede estar colocado dentro o fuera del robot, conectado a un puerto USB del PC.

Los dos Joystick se utilizan para controlar la dirección y la tracción, y los botones se utilizan para otros comandos importantes, como el control de velocidad, que puede establecerse en cinco categorías: muy lento, lento, medio, rápido y muy rápido.

Las instrucciones para operar el robot mediante el *Gamepad* son las siguientes:

- Encender el robot.
- Encender el ordenador mediante el botón azul.
- Esperar a que el ordenador se inicie y se encienda el led de la cámara (si estuviese instalada).
- Encender el *Gamepad* mediante el botón de inicio.

- Los cuatro leds rojos del mando parpadearán hasta que se establezca la conexión con el receptor Bluetooth. Cuando se realice la conexión, solo el primer led se mantendrá encendido. Si la conexión no se establece, comprobar el ordenador y el receptor Bluetooth y reiniciar el robot.

- Manteniendo pulsado el botón de Hombre Muerto es posible mover el robot y la cámara.

- Si el led 1 parpadea mientras los otros están apagados, se debe cargar el *Gamepad* mediante un puerto USB.

Figura 3.28: Botón de Hombre Muerto

Para poder ejecutar comandos en el robot a través del *Gamepad* es necesario mantener pulsado el botón de hombre muerto (figura 3.28).

Es sencillo cambiar la función de los botones dependiendo de las preferencias. El archivo *ps3.yaml* en la carpeta *Summit_pad/launch* contiene la asignación de los botones.

Se puede saber el número de cada botón ejecutando el siguiente comando:

jstest/dev/inputs/js0

Emparejamiento del controlador *Dualshock* con el receptor Bluetooth

El controlador *Dualshock* viene emparejado por defecto con el receptor Bluetooth pero dicho emparejamiento puede perderse si se consume la batería del mando o si el botón de inicio se pulsa de forma continua durante 10 segundos.

Si el emparejamiento se pierde, se puede reestablecer en el inicio del robot. Basta con apagar el ordenador, conectar el mando *Dualshock* mediante el cable USB y encender el ordenador. Esperar hasta que el robot inicie, desconectar el cable e iniciar el mando de nuevo.

Batería y cargador

El robot recibe alimentación de un pack de baterías LiFePO4. Con este set de baterías el robot es capaz de operar hasta 3 horas, dependiendo de los movimientos que realice.

Los circuitos del robot se alimentan cuando se conecta el interruptor principal S1. El conversor DC/DC, que proporciona alimentación a los distintos dispositivos de control, se alimenta a la misma vez, y también el conector de alimentación externo de 12V.

Las baterías están conectadas al robot a través de la caja de fusibles. Para cargar la batería hay un conector en la parte trasera del robot donde se puede conectar el cargador. Es una conexión directa, por lo que el interruptor general no afecta a la carga. Es posible cargar el robot y seguir trabajando al mismo tiempo sin ningún problema. Por seguridad, hay un fusible de 10 A (F1) entre el conector y las baterías.

El tiempo de carga total es de aproximadamente 45 minutos para el cargador proporcionado. No se debe usar otro tipo de cargador sin comprobar antes las especificaciones de la batería:

- ***Pack de baterías LiFePO***

  El pack de baterías (figura 3.29) está compuesto por cuatro celdas LiFePO (figura 3.30) y un módulo de protección (figura 3.31).

  Las baterías deben mantenerse limpias y secas para evitar escapes de corrientes. Debe comprobarse el recubrimiento de los cables de la batería para evitar cortocircuitos.

La batería se puede extraer fácilmente del robot abriendo la tira de velcro y desenchufando el conector de alimentación.

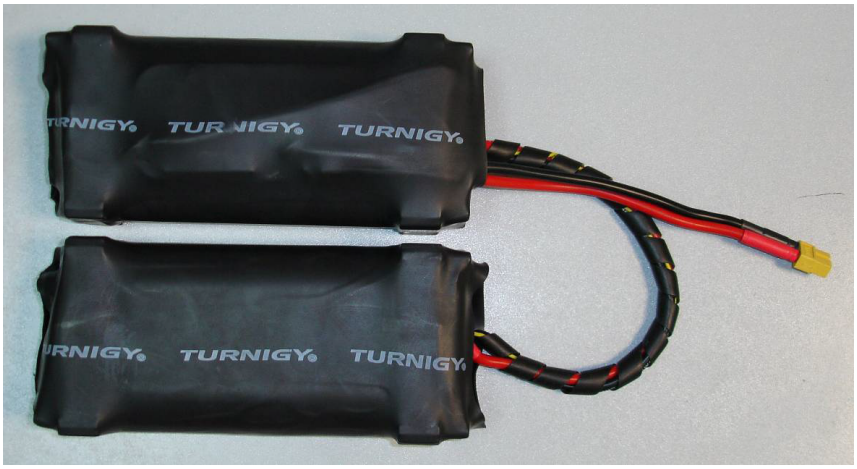

Figura 3.29: Pack de baterías

– *Célula LiFePO*

Figura 3.30: Célula LiFePO

Especificaciones:

- Capacidad normal de 15000 mAh.
- Tensión normal de 3.2 V.
- Impedancia interna < 8mOhms.
- Máxima corriente de descarga 10 C (150 A).
- Temperatura de carga: -10 a 45ºC.
- Temperatura de descarga: -20 a 60ºC.
- Rendimiento del ciclo > 2000 (80 % de su capacidad inicial a 1 C).
- Corriente estándar de carga: 1 C (15 A), Max 5 C (80 A).
- Peso: 500 g.

- ***Módulo de protección***

  En la figura 3.31 se muestra una imagen del módulo de protección asociado a las baterías, el cual, permite una descarga homogénea de las células LiFePO para extender la vida útil de la batería.

Figura 3.31: Módulo de protección

- ***Especificaciones de la batería***

  En la tabla 3.3 se muestran las características de la batería.

Elemento		**Criterio**
Tensión	Tensión de carga	CC/CV: Li-ion/Li-polymer (4.2 V/célula) LiFePO4 (3.6 V/célula)
	Balance de tensión para una célula	Li-ion: 4.20 V ± 0.025 V LiFePO4: 3.60 V ± 0.025 V
Corriente	Balance de corriente	0 ∼ 130 mA
	Balance de corriente para una célula	$< 25\ \mu$A
	Máxima corriente de carga y descarga	30 A
Protección contra sobrecarga	Tensión de detección de sobrecarga	3.8 ∼ 4.4 V (ajustable)
	Tiempo de retardo para detección	0.5 ∼ 2.5 s
	Tensión de liberación de sobrecarga	3.8 ∼ 4.4 V (ajustable)
Protección contra sobre-descarga	Tensión de detección de sobre-descarga	1.9 ∼ 3.1 V (ajustable)
	Tiempo de retardo para detección	50 ∼ 300 ms
	Tensión de liberación de sobre-descarga	1.9 ∼ 3.1 V (ajustable)
Protección contra sobre-corriente	Tensión de detección de sobre-corriente	0.06 ∼ 0.6 V
	Corriente de detección de sobre-corriente	50 ∼ 500 A
	Tiempo de retardo para detección	1 ∼ 50 ms
	Condición de liberación	Corte de corriente
Protección corta	Condición de detección	Cortocircuito exterior
	Tiempo de retardo para detección	200 ∼ 500 μs
	Condición de liberación	Corte de corriente
Resistencia	Circuito de protección	$\leq$ 50 mΩ
Temperatura	Rango de temperatura de operación	-40 ∼ 85 °C
	Rango de temperatura de almacenamiento	-40 ∼ 125°C
Tamaño		L60 x W60 x T5 mm

Tabla 3.3: Especificaciones de la batería

- ***Cargador***

Figura 3.32: Cargador

El cargador inteligente (figura 3.32) está diseñado para una carga rápida del pack de baterías LiFePO4. Es muy importante comprobar la potencia antes de conectarlos.

Especificaciones:

- ENTRADA
  - 115/230 VAC (ajustable).
  - Frecuencia de entrada 50-60 Hz.
- CARGA
  - Corriente de carga: puede variar entre 10, 12 o 15 A dependiendo del cargador.
  - Protecciones:
    - Protección contra cortocircuitos.
    - Protección contra sobretensiones.
    - Protección contra sobrecorrientes.
    - Protección contra polaridad inversa.
  - El cargador se para automáticamente cuando la batería está totalmente cargada.

- SALIDA
  - 14.6 VDC (4S x 3.65 V)
- Estado de los leds (ver figura 3.33)
  - LED1 ROJO: Alimentación AC activa.
  - LED2 ROJO: Cargando.
  - LED2 VERDE: Carga completa o batería no conectada.
- Temperatura
  - Temperatura de operación: -5 $\sim$ 40ºC.
  - Temperatura de almacenamiento: -10 $\sim$ 70ºC.

Precauciones:

- El cargador está diseñado exclusivamente para uso en interiores.
- El cargador debe colocarse de forma horizontal y bien ventilado, evitando la humedad y mantenerlo alejado de material inflamable o explosivo.
- La carcasa de aluminio es un disipador de calor, no cubrirlo.
- No desarmar el cargador debido al alto voltaje de su interior.

Figura 3.33: Vista frontal del cargador

Si se tienen problemas con la carga de la batería, comprobar las indicaciones mostradas en la tabla 3.4:

LED1 off	Comprobar el interruptor de potencia
	Comprobar el fusible de entrada
LED2 verde	Si la batería está aproximadamente a 14.6 V, carga completa
	Comprobar que el conector está correctamente conectado al robot
	Comprobar el fusible de salida
	Comprobar el fusible F1 del robot

Tabla 3.4: Problemas de carga

Diagrama de comunicaciones

La figura 3.34 muestra el diagrama de comunicaciones que existe dentro del robot.

La funcionalidad del sistema se puede extender aún más utilizando los puertos USB libres de los que disponga el robot.

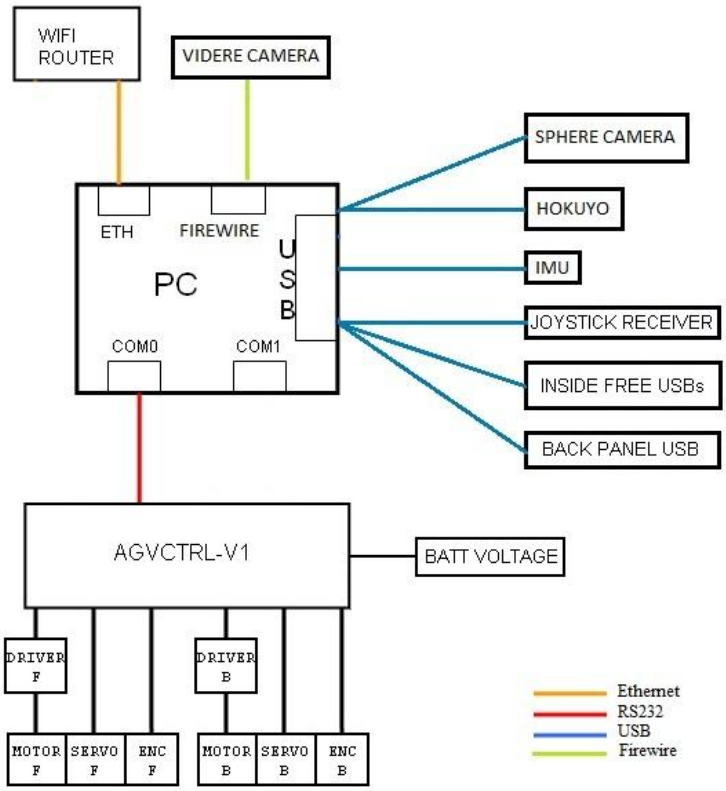

Figura 3.34: Diagrama de comunicación

Mantenimiento

En los apartados anteriores algunas partes principales del vehículo y componentes que necesitan mantenimiento han sido nombradas. La tabla 3.5 resume todos los elementos que necesitan mantenimiento, así como la frecuencia de revisión.

	De vez en cuando	**Cada 6 meses**	**Observaciones**
Tornillos	Comprobar que no se hayan perdido		
Drivers de los motores	Comprobar el correcto funcionamiento del ventilador		
Cables exteriores		Control de la tasa de desgaste	Si se desgastan, se recomienda cambiarlos
Ruedas		Control de la tasa de desgaste	Reemplazar cuando sea necesario
Rodamientos		Controlar el estado	Si se dañan, se recomienda cambiarlos por unos nuevos
PC		Limpieza interior	
Batería		Comprobar su autonomía	Intercambiando las células se puede extender la vida útil

Tabla 3.5: Resumen de mantenimiento

3.1.3 Escáner láser RPlidar de RoboPeak©

RPlidar [24] es un escáner láser 2D de bajo coste desarrollado por RoboPeak©. El sistema puede realizar un escaneo de 360º en un rango de 6 metros. La nube de puntos 2D que se genera se puede usar para realizar mapas, localización y modelado de objetos, y modelado de entornos.

La frecuencia de escaneo que alcanza el dispositivo es de 5.5 Hz cuando muestrea 360 puntos en cada vuelta, y se puede configurar hasta un máximo de 10 Hz para un número de puntos superior.

RPlidar es básicamente un sistema de medida por triangulación láser. Puede funcionar de forma excelente en cualquier tipo de entorno interior.

- ***Conexión***

  RPlidar (figura 3.35) contiene un escáner láser y un motor. Al conectarse la alimentación, el dispositivo comienza a rotar en sentido horario. El usuario puede obtener los datos del escáner a través de la interfaz de comunicación (Puerto serie/USB).

  El sistema viene con detección de velocidad y un sistema de adaptación. El sistema ajustará la frecuencia del láser automáticamente dependiendo de la velocidad del motor. Se puede obtener la velocidad real a través de la interfaz de comunicación.

Figura 3.35: RPlidar de RoboPeak©

- ***Mecanismo***

  RPlidar se basa en el principio de triangulación (ver figura 3.36) y usa un hardware de procesamiento y adquisición de alta velocidad, desarrollado por RoboPeak©. El sistema mide la distancia más de 2000 veces por segundo en alta resolución (<1 % de la distancia).

  El dispositivo emite una señal láser infrarroja modulada que es reflejada por el objeto que tiene que detectar. La señal reflejada es muestreada por el sistema de adquisición del RPlidar y el DSP embebido comienza a procesar la información y enviar la salida como un valor de distancia y el ángulo entre el objeto y el RPlidar.

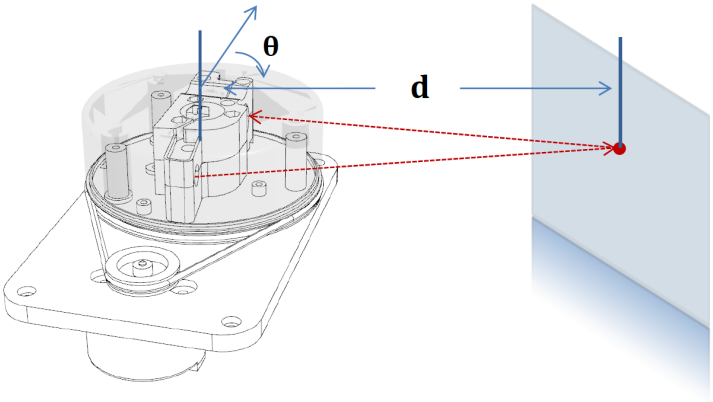

Figura 3.36: Sistema de triangulación

- ***Datos de salida***

Cuando el RPlidar está funcionando, los datos de salida se transfieren a través de la interfaz de comunicación. Cada muestreo contiene la información representada en la tabla 3.6. El dispositivo emite la información de forma continua pero se puede configurar para que se detenga mediante un comando de *stop*. El protocolo de muestreo del escáner se muestra en la figura 3.37.

Tipo de dato	Unidad	Descripción
Distancia	mm	Valor actual de medida de distancia
Cabecera	Grados(°)	Valor actual del ángulo de medida
Calidad	Nivel	Calidad de la medida
Bandera de incio	Booleano	Bandera para un nuevo escaneo

Tabla 3.6: Datos de salida de RPlidar

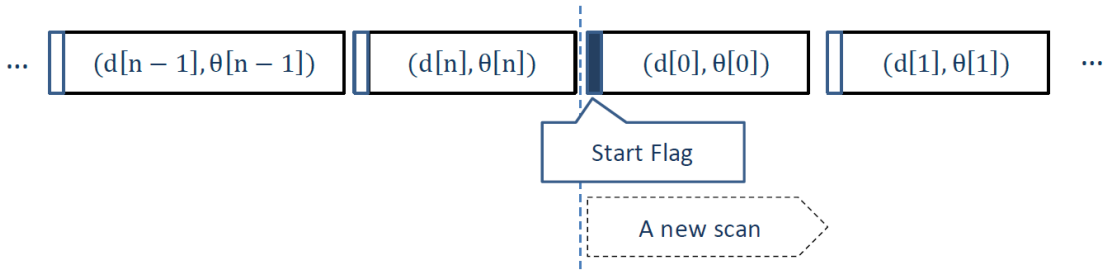

Figura 3.37: Protocolo de muestreo

- ***Aplicaciones***

  - Navegación y localización de robots.
  - Detección de obstáculos.
  - Modelado de entornos.
  - Localización simultánea y mapeado (SLAM).

3.2 Métodos

A continuación se exponen los métodos empleados para cumplir los objetivos propuestos en este proyecto.

3.2.1 Puesta en marcha del robot Summit

Para poder trabajar con el robot móvil Summit se tuvo que realizar una puesta en marcha previa y una evaluación tanto de los componentes incorporados en el robot como del software de los mismos. Se comenzó con la colocación de los diversos fusibles que aseguran la protección de los componentes internos y la sincronización del *Gamepad* bluetooth para poder realizar la teleoperación del robot.

Una vez completados los pasos anteriores, se realizó una prueba de campo teleoperando el robot, comprobando su funcionamiento antes de conectarlo a un ordenador. Evaluado el comportamiento del robot y sus distintos movimientos se procedió a comprobar qué dispositivos estaban incorporados. El robot, disponía de una cámara de visión, una unidad inercial o IMU y un encoder en el eje delantero. Para poder realizar mapas de entornos y navegación autónoma, el robot requería de un sensor láser, que se adquirió posteriormente para dichos fines.

Al conectar el robot al ordenador se observó que el software preinstalado era ROS Fuerte y además contenía las carpetas de paquetes pertenecientes a los componentes del robot. Ya que la versión ROS Fuerte estaba obsoleta se procedió a instalar ROS Hydro y a ponerse en contacto con la empresa Robotnik© para conseguir los paquetes del robot actualizados a dicha versión.

Actualizados los paquetes, se procedió a trabajar con el robot en los aspectos de construcción de mapas y navegación autónoma. Se realizaron varias pruebas en salas del edificio CITE IV de la Universidad de Almería para comprobar el funcionamiento del sensor RPlidar y posteriormente se realizó un ensayo de navegación autónoma en el pasillo del edificio colocando varios obstáculos para comprobar la efectividad de diversos métodos de navegación y compararlos entre sí.

3.2.2 Instalación y entorno de ROS Hydro

El Sistema Operativo de Robots (ROS) [22] dispone de una serie de versiones, siendo ROS Jade la última de ellas. Ya que el controlador del robot móvil Summit proporcionado por la empresa Robotnik© está adapatado a la versión ROS Hydro, se optó por instalar dicha versión y realizar todos los proyectos en base a la misma.

A continuación se detallan los pasos de instalación de ROS Hydro:

1. *Configurar los repositorios de Ubuntu.*

   Configurar los repositorios de Ubuntu para permitir 'restringidos'. 'universales' y 'multiversales'.

2. *Configurar la source.list.*

   Ejecutar el siguiente comando:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu precise main" >
/etc/apt/sources.list.d/ros-latest.list'
```

3. *Configurar las llaves.*

   Ejecutar el siguiente comando:

```
wget http://packages.ros.org/ros.key -O - | sudo apt-key add -
```

4. *Instalación.*

   Actualizar Ubuntu mediante:

```
sudo apt-get update
```

   Instalar la versión completa de ROS Hydro ejecutando el siguiente comando:

```
sudo apt-get install ros-hydro-desktop-full
```

   También se pueden instalar los paquetes de forma individual:

```
sudo apt-get install ros-hydro-<paquete>
```

5. *Iniciar ROSdep.*

   Antes de poder usar ROS, se debe inicializar *rosdep*. *Rosdep* permite instalar las dependencias para los archivos fuente que se quieran compilar y es requerido para ejecutar algunos componentes importantes de ROS.

```
sudo rosdep init
rosdep update
```

6. *Configurar el entorno.*

   Añadir la variable de entorno de ROS Hydro al archivo *bash*:

```
echo "source /opt/ros/hydro/setup.bash" >> ~/.bashrc . ~/.bashrc
```

   Para cambiar a la variable de estado de otra versión de ROS y poder trabajar con sus paquetes se debe ejecutar el siguiente comando en un terminal:

```
source /opt/ros/<versión de ROS>/setup.bash
```

7. *Instalar ROSinstall.*

   ROSinstall es una herramienta de línea de comandos que es usada frecuentemente. También permite descargar facilmente muchos árboles de código fuente con un solo comando.

```
sudo apt-get install python-rosinstall
```

8. *Crear un entorno de trabajo.*

```
$ mkdir -p  /catkin_ws/src
$ cd  /catkin_ws/src
$ catkin_init_workspace
```

   Los paquetes descargados se colocan en el directorio *src* dentro de la carpeta *catkin_ws*. Para compilar estos paquetes basta con ejecutar el siguiente comando:

```
$ cd  /catkin_ws/
$ catkin_make
```

3.2.3 Entorno de simulación Gazebo

Gazebo [1], que nace con el nombre de *Gazebo Project*, es un simulador 3D cinemático, dinámico y multi-robot que permite realizar simulaciones de robots articulados en entornos complejos, interiores o exteriores, realistas y tridimensionales. En la figura 3.38 se puede ver un ejemplo de la pantalla principal de Gazebo al cargar el modelo virtual de un robot con brazos manipuladores en un entorno urbano. Al ejecutar comandos de velocidad o iniciar una navegación autónoma, el modelo virtual se moverá por el espacio representando al robot real.

Figura 3.38: Entorno de simulación Gazebo

Entre sus características cabe destacar las siguientes [1]:

- Gazebo es un software libre, financiado en parte por *Willow Garage*, pudiendo ser reconfigurado, ampliado y modificado.

- Compatible con ROS y Player. Se puede ejecutar Gazebo desde ROS y utilizar las APIs de este último para controlar los robots en las simulaciones, es decir, enviar y recibir datos de éstas.

- Simulación realista de la física de los cuerpos rígidos. Los robots pueden interactuar con el mundo (pueden coger y empujar cosas, rodar y deslizarse por el suelo) y viceversa (les afecta la gravedad y pueden colisionar con obstáculos del mundo).

- Capacidad de desarrollar y simular modelos de robots propios (URDF) e incluso cargarlos en tiempo de ejecución.

- Posibilidad de crear escenarios (mundos) de simulación, variando las características de los contactos con el suelo, los obstáculos e incluso los valores de la gravedad en las tres dimensiones. También es posible variar las características de contacto en cada *link* individualmente.

- Contiene diversos plugins para añadir sensores al modelo del robot y simularlos, como sensores de odometría (GPS e IMU), de fuerza, de contacto, láseres y cámaras estéreo.

Internamente, Gazebo utiliza:

- *Open Dynamics Engine (ODE)*, un motor de física basada en la formulación de problemas sobre la complementariedad de restricción. Se compone de una serie de bibliotecas de alto rendimiento para la simulación de la cinemática de cuerpos rígidos.

- *OGRE*, motor de renderizado de escenas gráficas en 3D.

En cuanto a su integración en ROS, el simulador posee su propio stack denominado *simulator_gazebo*, que se divide en una serie de paquetes con diferentes funcionalidades:

Gazebo: Contiene la versión más reciente de *Gazebo Project* integrada en ROS, de forma que es ejecutable como un nodo más de ROS, poseyendo sus propios tópicos y servicios.

Gazebo_msgs: Contiene los tipos de servicios (svr) y mensajes (msg) para interactuar con Gazebo desde ROS. Por tanto, los tipos de mensajes y servicios a utilizar para enviar o recibir del simulador deben ser incluidos a partir de este paquete.

Gazebo_plugins: Contiene los plugins para utilizar los diferentes sensores.

Gazebo_tools: Contiene, entre otras herramientas, las que nos permiten enviar o eliminar modelos URDF en el simulador, mediante el nodo *gazebo_model* del mismo paquete.

Gazebo_worlds: Contiene los archivos de configuración estándares del entorno (*.world*) desarrollados por *Willow Garage*, así como diferentes modelos URDF de objetos (muros, mesas, sillas, ...) a modo de ejemplo.

Configuración y generalidades del entorno de simulación

Para ejecutar el simulador con un entorno ya configurado es necesario pasarle como argumento un archivo de configuración de extensión *.world*. Este tipo de archivo se escribe en XML y en él se especifican parámetros del motor físico ODE (como el tiempo de integración, el valor de la gravedad,...), del motor gráfico OGRE (renderizado, sombras, cielo, ambiente,...) y de los objetos que deben aparecer en el simulador automáticamente.

Modelos URDF (Universal Robotic Description Format)

El archivo URDF es un formato XML que representa el modelo del robot. Permite describir las partes mecánicas de un robot y sus especificaciones. Se puede utilizar para la mayoría de robots pero posee ciertas limitaciones, por ejemplo para los robots paralelos, cuyos bucles no se pueden representar. En la figura 3.39 se muestra la relación de los componentes de URDF.

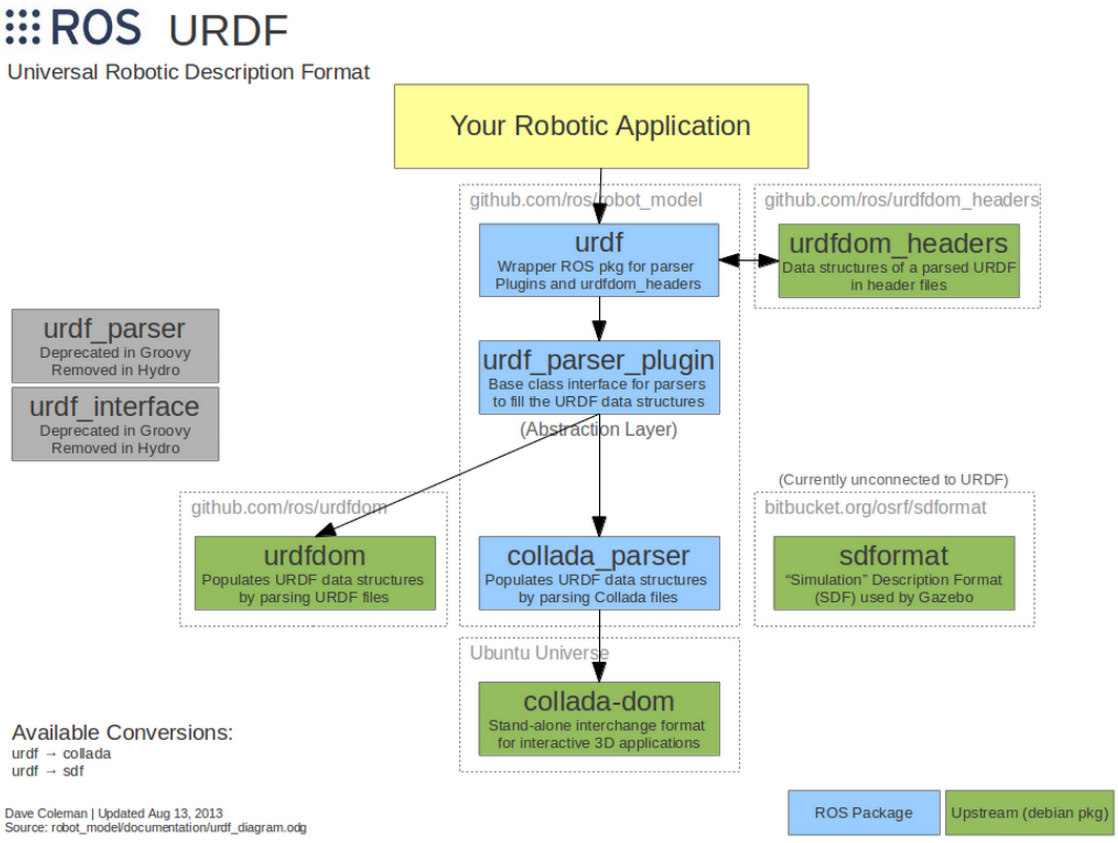

Figura 3.39: Relación de los componentes para URDF

Las especificaciones técnicas que cubre el archivo URDF son:

- Descripción cinemática y dinámica del robot.
- Representación visual.
- Modelo de colisiones.

Cada nodo y sub-nodo dentro del modelo al menos tiene dos propiedades: un nombre y una posición (posición y orientación).

Los robots se describen en URDF utilizando dos tipos de elementos: los enlaces (*links*) y las articulaciones (*joints*).

- *Enlaces*

  Los enlaces representan las partes físicas del cuerpo del robot. Pueden tener tres tipos de nodos hijos:

  - Los nodos inerciales contienen las propiedades físicas del sólido, por ejemplo, la masa en kg y su matriz inercial a lo largo de los ejes x, y y z.
  - El nodo de colisión contiene la geometría del cuerpo sólido. Generalmente se simplifica debido a la carga computacional que produce la colisión de dos cuerpos.
  - El nodo de visualización contiene también la geometría del robot, pero de forma precisa y detallada. Se utiliza para el renderizado del robot únicamente y no afecta a las propiedades físicas del robot. También contiene la información del material del elemento, su color y su textura.

  Los enlaces son generalmente partes físicas del robot que se encuentran entre dos motores o transmisiones o partes del robot que se dividen debido a la complejidad de su geometría.

- *Articulaciones*

  Las articulaciones son la parte que une dos enlaces. Representan motores y actuadores del robot real. Pueden ser de los siguientes tipos:

  - Revolución: Una articulación que gira en torno a un eje con una posición máxima y mínima.
  - Continuo: Una articulación de revolución que no tiene límite máximo y mínimo.
  - Prismático: Una articulación deslizante que se traslada sobre un eje. Tiene límites superior e inferior fijos.

- Fijo: Articulación que no posee grados de libertad. Se utiliza para unir dos enlaces que componen el mismo elemento pero que se han separado para simplificar la geometría.
- Flotante: Esta articulación permite 6 grados de libertad y se le pueden imponer distintos límites de posición y movimiento.
- Planar: Articulación con dos grados de libertad que permite el movimiento en un plano definido por su vector ortogonal.

Las articulaciones tienen dos nodos hijos llamados padre e hijo. Contienen el nombre de los enlaces que une. El sistema de coordenadas del hijo está determinado por la transformada dada en el origen del sistema de coordenadas del padre. Si no se da la transformada, hijo y padre tendrán el mismo sistema de coordenadas.

La mayoría de las articulaciones requieren un eje para describir la dirección del movimiento. Esto se hace mediante un nodo hijo con las coordenadas x, y y z en el sistema de coordenadas del padre.

Además, las articulaciones pueden tener otros nodos hijos opcionales que proporcionen información mecánica, como fricción, límites físicos o límites para el controlador.

3.2.4 Interfaz de visualización RVIZ

Rviz (*ROS visualization*) es una herramienta de visualización 3D para representar datos proporcionados por los sensores e información de estados proveniente de ROS. Usando esta herramienta se puede ver la configuración de un modelo virtual de nuestro robot, así como de modelos de robots ya creados para trabajar (como es el caso del robot Summit). A parte de poder visualizar una representación del robot, también se pueden representar los valores de los sensores del robot que proceden de los tópicos publicados en ROS, incluyendo datos de cámaras, sensores de distancia, sónares, lidars, etc.

En la figura 3.40 se observa el modelo del robot Summit para la interfaz Rviz. Este modelo se representa igual que en el entorno de simulación Gazebo, siendo este último el entorno donde se puede diseñar un mapa personalizado, mientras que en Rviz no se podrían ver los obstáculos sin un sensor que permita la detección de los mismos, es decir, en Rviz no se puede personalizar el entorno por el que se desplaza el robot.

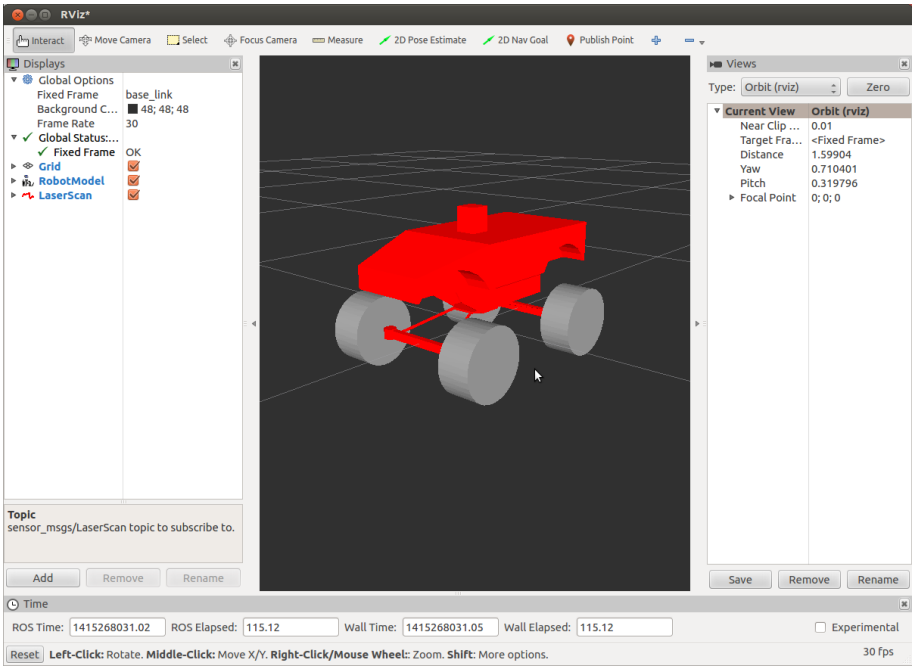

Figura 3.40: Modelo Summit en Rviz

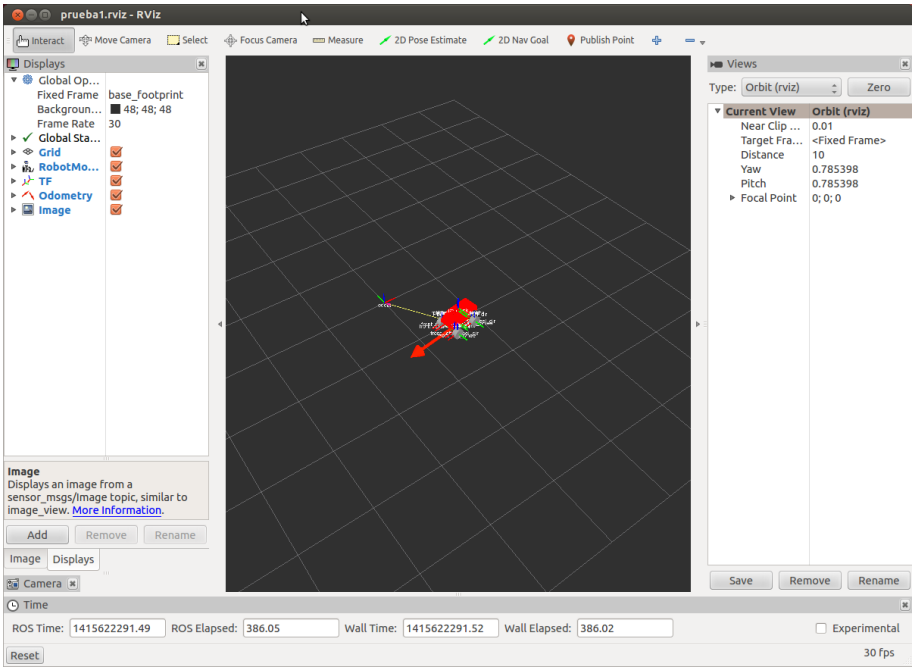

Figura 3.41: Interfaz Rviz

Observando la interfaz de Rviz, en la ventana izquierda (*Displays*) se añaden los diversos nodos con los que se esté trabajando. En el caso de la figura 3.41 se puede ver el modelo del robot, el árbol de transformadas de los enlaces del robot (TF), la odometría que publica el robot (flecha roja), un escenario 2D cuadriculado y el nodo de la cámara. Dentro de estas aplicaciones se selecciona el tópico correspondiente a cada dispositivo, es decir, la aplicación de Rviz se conecta a los datos que emite el dispositivo correspondiente del robot. Por ejemplo, el nodo de la cámara del robot esta emitiendo las imágenes en tiempo real, por lo que, para ver estas imágenes en Rviz se añadiría la opción *Image* y se conectaría al tópico en el cual el nodo de la cámara está emitiendo las imágenes.

3.2.5 *Stacks* del robot móvil Summit

El robot Summit viene incorporado con una serie de carpetas con paquetes de ROS en el PC empotrado. Estas carpetas contienen los paquetes necesarios para operar el robot tanto en simulación como de forma autónoma o teleoperado mediante el *Gamepad*.

Carpeta *Summit_sim*

Esta carpeta contiene los paquetes necesarios para realizar la simulación del robot y la navegación autónoma.

- ***Summit_2dnav***

  Este paquete contiene todos los archivos de configuración para ejecutar el algoritmo de navegación plnificada del paquete *navigation* junto con el localizador AMCL (*Adaptative Monte Carlo Localization*) o la creación de mapas mediante SLAM (*Simultaneous Localization And Mapping*).

- ***Summit_controller***

  El controlador del robot para el entorno de simulación Gazebo. Implementa el control de la cinemática Ackerman del robot Summit. Permite controlar al robot en modo "Ackerman individual", "Doble Ackerman" y "Modo Paralelo".

- ***Summit_description***

  Esta carpeta contiene los archivos *urdf*, modelos y otros elementos necesarios para la descripción del modelo del robot.

- ***Robotnik_msgs***

  Proporciona los mensajes y los servicios para los distintos componentes del robot, como los mensajes de entrada/salida o el pant/tilt para la cámara.

Estos mensajes han sido desarrollados para facilitar la interoperabilidad a través del sistema.

- ***Summit_joystick_teleop***

  Este paquete nos permite usar el *Gamepad* con *Summit_controller*, adaptando y enviando los mensajes recibidos a través de las entradas del *Gamepad* al tópico "Summit_controller/cmd_vel", siendo éste el tópico de velocidad y posición del robot.

- ***Summit_odometry***

  Nodo que proporciona la odometría ficticia del robot para el entorno de simulación Gazebo.

- ***Summit_sbpl_nav***

  Hace uso de los paquetes *sbpl* (*Search Based Planning*) y *sbpl_lattice_planner* para crear las rutas de navegación.

Carpeta *Summit_robot*

En esta carpeta se encuentran los paquetes para iniciar los componentes del robot real:

- ***Summit_controller_dspic***

  Esta carpeta contiene el controlador del robot real. Además proporciona la odometría del robot a partir del encoder instalado en el eje delantero.

- ***Robotnik©_arduimu***

  Contiene los programas para iniciar y procesar los datos del IMU que incorpora el robot.

- ***Imu_tools***

  Contiene los archivos de configuración del IMU, así como un filtro para mejorar la señal obtenida.

- ***Summit_pad***

  Este paquete nos permite usar el *Gamepad* con el robot real. Los mensajes del *Gamepad* se transmiten al tópico "Summit_controller_dspic/cmd_vel".

- ***Sphere_camera***

  Esta carpeta contiene los archivos para iniciar y controlar la cámara que incorpora el robot.

- ***Usb_cam***

  Contiene los archivos de configuración e inicio para la cámara.

- ***Webcam_tools***

  Contiene los archivos de configuración para el control del movimiento de la cámara.

- ***Summit_complete***

  En esta carpeta se encuentra el archivo *launch* que inicia todos los componentes del robot de forma conjunta.

Carpeta *Experimental*

Este paquete contiene los nodos necesarios para la navegación y el control del robot, además de paquetes para investigación en cuanto a navegación autónoma:

- ***Assisted_teleop***

  Contiene archivo relacionados con la teleoperación del robot.

- ***Pose_base_controller***

  Contiene un programa para determinar la posición del robot.

- ***Eband_local_planner***

  Contiene archivos relacionados con la navegación local del robot.

- ***Pose_follower***

  Contiene un programa para seguimiento de posiciones.

- ***Twist_recovery***

  Contiene los archivos para configurar el movimiento de recuperación del robot frente a un camino bloqueado en la trayectoria.

- ***Goal_passer***

  Contiene los archivos para establecer metas.

- ***Sbpl_lattice_planner***

  Contiene el programa para crear rutas de navegación.

- ***Sbpl_recovery***

  Contiene archivos de configuración para el paquete *sbpl*.

Nuevos paquetes que implementan nuevos métodos de navegación se incorporan a ROS de forma continua, por lo que, algunos de los paquetes preinstalados en el robot pueden quedar obsoletos.

3.2.6 *Stacks* para construcción de entornos

Una de las principales tareas que se debe resolver en la navegación autónoma de robots reside en la localización del robot en el entorno que le rodea, así como disponer de un mapa detallado que le permita moverse. Para ello se utilizan diversas técnicas basadas en SLAM (*Simultaneous Localization And Mapping*), que permiten la construcción de mapas de entornos a partir de los datos recogidos por sensores láseres de rango y los sensores de posición incoporados en el robot.

Slam Gmapping

Slam_gmapping contiene el paquete *gmapping* de ROS que permite el SLAM, además de crear un nodo llamado *slam_gmapping*. A partir de este nodo se pueden crear mapas de ocupación cuadriculados en 2D del entorno a partir del láser y la posición recogida por el robot móvil.

Hector Slam

Hecto_slam, similar a *gmapping*, es un paquete que incorpora diversos paquetes basados en SLAM, siendo el paquete principal *hector_mapping*, que permite realizar dicha técnica sin necesidad de tomar datos de la odometría del robot. Aprovecha las altas tasas de actualización de los LIDARs actuales para proporcionar una estimación de la posición en 2D. Los paquetes que incopora *hector_slam* permiten extender las capacidades de esta aplicación incluso a plataformas que permitan el movimiento de pitch y roll.

3.2.7 *Stacks* para navegación autónoma

En este apartado se presentan los distintos métodos de navegación implementados en ROS que se han utilizado con el robot móvil Summit.

Navigation

Es un *stack* de navegación en 2D que obtiene información de la odometría, los sensores y una meta para crear una trayectoria desde el robot móvil hasta el objetivo, proporcionando los comandos de velocidad adecuados para seguir esa ruta.

Este paquete implementa una navegación planificada, es decir, en base a un mapa preestablecido y la posición del robot en dicho mapa, es capaz de trazar la ruta óptima desde la posición del robot a un punto dado en el mapa, evadiendo los obstáculos que estén representados en el mapa, sin tener en cuenta obstáculos emergentes que puedan aparecer. La creación de la trayectoria no se realiza en tiempo real, sino previamente al inicio del movimiento del robot. Por ello, si el robot no puede realizar un movimiento de esa trayectoria, se detendrá y recalculará la trayectoria a seguir.

Este paquete está especialmente diseñado para robots con configuración diferencial o holonómica y un contorno cuadrado o circular, por lo que, se debe adaptar el código para robots que posean configuraciones distintas y tener especial atención para robots con gran longitud, ya que tendrán problemas en espacios estrechos.

Sus nodos más importantes son:

- ***Amcl:*** Sistema de localización probabilística para robots en entornos 2D a partir del algoritmo AMCL (*Adaptative Monte Carlo Localization*).
- ***Base_local_planner:*** Implementa los algoritmos de Ventana Dinámica (*Dynamic Windows*) y Trayectorias Curvas (*Trajectory Rollout*) para realizar la navegación local de robots en un plano.
- ***Carrot_planner:*** Planificador que busca lugares para colocar metas que el robot pueda seguir.
- ***Global_planner:*** Nodo de planificación de trayectorias.
- ***Map_server:*** Nodo que provee la información referente a un mapa establecido.
- ***Move_base:*** Paquete que implementa las acciones necesarias para desplazar una plataforma móvil hasta un punto establecido.
- ***Nav_core:*** Paquete que proporcional las interfaces necesarias para la navegación de robots.
- ***Robot_pose_ekf:*** Paquete de localización en 3D a partir de la medida de posición de distintos sensores.

De este paquete cabe destacar *move_base*, el paquete de navegación autónoma por referencia en ROS.

El nodo *move_base* proporciona a ROS una interfaz de configuración e interactuación para el *stack* de navegación en un robot. En la figura 3.42 se puede observar una esquema en alto nivel de este nodo y su interacción con otros componentes. Los cuadros azules representan información proporcionada por el robot, los cuadros grises son opcionales pero se pueden utilizar en todos los sistemas, y lo nodos blancos son estrictamente necesarios, proporcionados por en el *stack* de navegación.

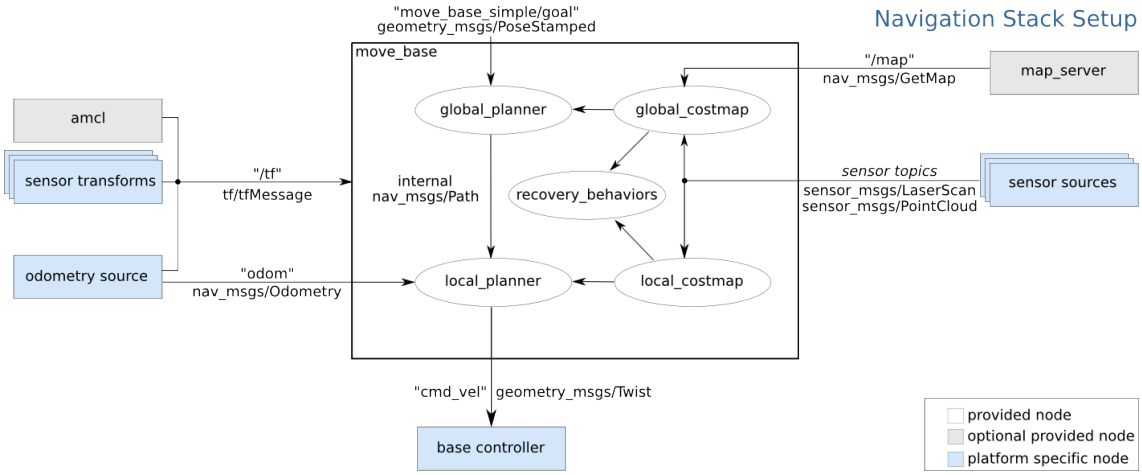

Figura 3.42: Estructura de move_base

Al ejecutar el nodo *move_base* correctamente configurado para un robot, proporcionará una navegación autónoma del mismo hasta una meta establecida con una tolerancia especificada por el usuario. En ausencia de obstáculos dinámicos, el robot alcanzará su objetivo o devolverá una señal de fallo hacia el usuario. El robot puede realizar comportamientos de recuperación si llega a un punto de atasco (figura 3.43). Estos movimientos son:

Primero, los obstáculos alrededor del una región serán eliminados del mapa del robot. Después, si es posible, el robot realizará una rotación sobre sí mismo para buscar espacio libre. Si hasta aquí todo falla, el robot tendrá un comportamiento más agresivo, eliminando los obstáculos de fuera de la región permitida donde puede girar sobre sí mismo. Seguidamente a este paso, se intentará realizar otra rotación sobre sí mismo. Si todos los procesos fallan, se notificará al usuario que la meta es inaccesible y se abortará la navegación. Todos estos comportamientos se pueden configurar, y por su definición, se comprueba que están diseñados para robot con configuración diferencial, ya que, un robot con configuración Ackerman no puede realizar un giro sobre sí mismo.

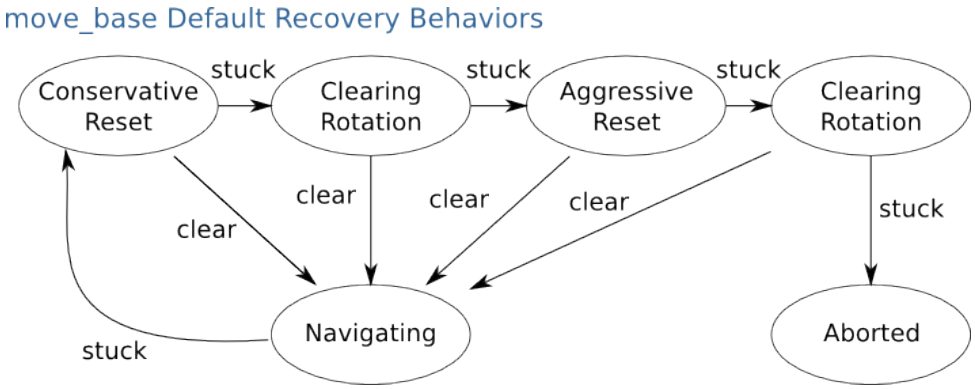

Figura 3.43: Comportamientos de recuperación

Robotnik© Pure Pursuit

El *stack purepursuit_planner*, creado por la empresa Robotnik©, implementa un planificador para seguir una lista de metas implementando el algoritmo *Pure Pursuit*, sin tener en cuenta el entorno que rodea al robot, es decir, sin detectar los objetos de su alrededor.

Pure Pursuit [10] es un algoritmo de búsqueda que funciona calculando la curva que movería un vehículo de su posición actual hacia una posición de destino. El objetivo es crear un punto en el camino a cierta distancia en frente del vehículo. El nombre de seguimiento puro proviene de la analogía que se utiliza para describir el método. El vehículo intenta perseguir un punto en movimiento en el camino, a cierta distancia de él. Esta analogía se utiliza a menudo para comparar este método con la forma en la que conducen los humanos. Se visualiza un punto a cierta distancia enfrente del coche. La distancia de este punto cambia a medida que el vehículo se desplaza o realizamos giros.

El seguimiento puro es un método geométrico para determinar la curvatura que trazaría un vehículo hacia una posición determinada, denominada meta. Esta meta es un punto en el camino que está a una distancia especifica frente a la posición del vehículo. Se construye un arco que une el punto actual con la meta.

Este paquete se centra en la navegación punto a punto. El robot se desplazará desde su posición actual hasta el punto designado sin tener en cuenta el entorno que lo rodea, por lo tanto, si se desea mover el robot de un punto a otro en una zona con obstáculos se deberá crear la ruta de forma manual mediante una sucesión de puntos de tal manera que entre dos puntos consecutivos no haya un obstáculo entre medias.

Mrpt Navigation

El paquete *mrpt_navigation* [14] procedente de MRPT (*Mobile Robot Programming Toolkit*) [8], surge como alternativa frente a *navigation*, implementando un sistema de navegación reactiva pura basado en el algortimo TP-Space y un localizador similar a AMCL (figura 3.44).

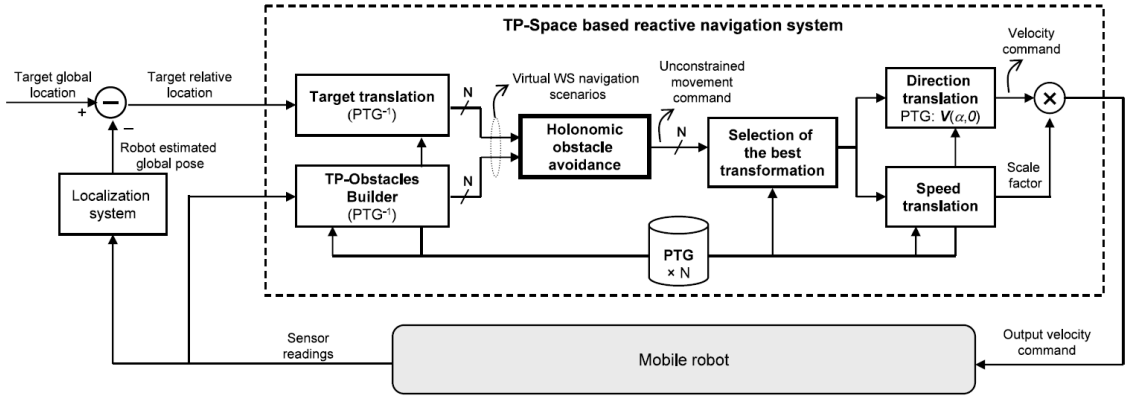

Figura 3.44: Sistema de navegación reactiva MRPT

Este método de navegación permite desplazar el robot desde su posición a una meta dada evitando la colisión con los obstáculos cercanos.

La navegación no se basa en un mapa, es decir, el propio sistema genera un mapa interno de obstáculos alrededor del robot y escoge el mejor camino posible en base al espacio libre, por lo que no traza la ruta hacia la meta de forma previa, como el paquete *navigation*, sino que traza la trayectoria en tiempo real. Esta es una de las principales ventajas de este método frente a la navegación planificada.

El paquete *mrpt_navigation* incluye los siguientes nodos:

- **Mrpt_bridge:** Funciones en C++ que permiten convertir los mensajes de ROS en clases para MRPT.
- **Mrpt_local_obstacles:** Construcción de un mapa de obstáculos (nube de puntos o mapa de ocupación) a partir de las lecturas recientes de los sensores.
- **Mrpt_localization:** Nodo para la localización de robots en 2D con un filtro de partículas y diferentes estilos de mapas. Representa una implementación alternativa para el algortimo AMCL.
- **Mrpt_map:** Nodo para publicar mapas (estáticos o preestablecidos).

- **Mrpt_msgs:** Mensajes comunes para los paquetes MRPT.
- **Mrpt_rawlog:** Nodo para almacenar datos, similar a *rosbag*.
- **Mrpt_reactivenav2d:** Navegación reactiva pura mediante el algortimo TP-Space.

Se pueden configurar fácilmente los parámetros de navegación, así como definir las dimensiones del robot, las velocidades máximas (lineal y angular), un filtro que proporcione un cambio gradual de velocidad o el método de detección de obstáculos (ND o VFF).

Los métodos de detección de obstáculos que implementa este sistema son los siguientes:

- *Vector Force Field (VFF)*

  El objetivo principal del *campo de fuerzas potenciales* (VFF) [9] es producir una fuerza que actúa en el robot de tal manera que le permite evadir obstáculos. Este método se realiza creando una fuerza repulsiva que aumenta conforme el robot se acerca al obstáculo y viceversa. En un espacio con múltiples obstáculos se generan múltiples fuerzas que conforman un espacio de fuerzas vectoriales. Este campo de fuerzas y la dirección del robot se combinan para crear un vector resultante que impulsa al robot en la dirección correcta.

- *Nearness Diagram (ND)*

  A partir de diagramas se calculan la distancia hasta los obstáculos cercanos y el espacio libre disponible [18], definiendo una serie de situaciones e implementando leyes de movimiento (acciones) para cada situación. En tiempo real, se utiliza la información de los sensores para identificar las situaciones en las que se encuentra el robot y ejecutar la acción asociada a la situación, generando el comando de velocidad correspondiente.

La ventaja que ofrece *mrpt_navigation* frente a otros métodos de navegación reactiva reside en la implementación de distintas trayectorias que puede realizar el robot. Comúnmente para la navegación reactiva tradicional se utilizan trayectorias circulares para definir los movimientos del robot. En cambio, este paquete permite implementar otros tipos de trayectorias (denominadas PTGs) que se tienen en cuenta a la hora de navegar para crear rutas más eficientes y optimizadas. Las PTGs implementadas en MRPT son:

- **C PTG:** *Trayectorias circulares*

  El modelo de trayectoria más simple y el más utilizado para la navegación reactiva tradicional. La velocidad permanece constante a lo largo de la trayectoria.

- **α-A PTG:** *Trayectorias con rumbo asintótico*

  Estas trayectorias se generan mediante comandos de velocidad linales y angulares, que son directa e inversamente proporcionales a la diferencia entre el rumbo del robot y el parámetro α.

- **α-SP PTG:** *Trayectorias en espiral*

  El objetivo de estas trayectorias es mostrar que la navegación reactiva no solo se compone de lineas rectas y trayectorias en arco.

- **C$\|C_{\pi/2}$S and CS PTG:** *Trayectorias óptimas*

  Estos modelos han sido diseñados para crear trayectorias óptimas en robots de tipo coche con un radio de giro mínimo R.

Capítulo 4: Resultados

A continuación se exponen las pruebas realizadas con el robot tanto en simulación como con el robot real.

4.1 Ensayos en simulación

Antes de llevar a cabo los ensayos de navegación autónoma sobre el robot móvil real, se realizaron pruebas en el entorno de simulación Gazebo utilizando los *stacks* de simulación del robot proporcionados por Robotnik©.

4.1.1 Comienzo

Al inicio de este proyecto, el robot disponía de ROS Fuerte en su fase inicial. Se intentó realizar una serie de pruebas con su contenido pero resultaron fallidas debido a la obsolescencia del sistema operativo del que disponía el robot. Como solución a dicho problema, la empresa Robotnik© proporcionó los *stacks* del controlador del robot Summit, el *Gamepad* y la carpeta de simulación del robot actualizados para ROS Hydro. Comprobado el funcionamiento de las mismas, se actualizó a la versión ROS Hydro tanto en el robot, como en el ordenador remoto y se procedió a realizar las pruebas pertinentes.

4.1.2 Teleoperación en simulación

La primera prueba que se realizó con el robot en simulación fue la teleoperación del mismo a través de la ventana de control de la interfaz Rviz. Los programas en ROS se denominan nodos y se pueden ejecutar en conjunto o de forma independiente. El robot Summit dispone de varias carpetas con diversos nodos que proporcionan diversas funcionalidades. Es necesario un nodo central (*ROS Core*) que gestione la información del resto de nodos. Este nodo central se ejecuta con el controlador del robot real o, en el caso de simulación, al ejecutar el nodo del robot simulado en Gazebo. Por lo tanto, el primer paso es ejecutar el nodo contenido en la carpeta *summit_gazebo* que contiene un *launch* o ejecutable que inicia el entorno Gazebo y coloca un modelo del robot Summit en el mundo, tal como muestra la figura 4.1.

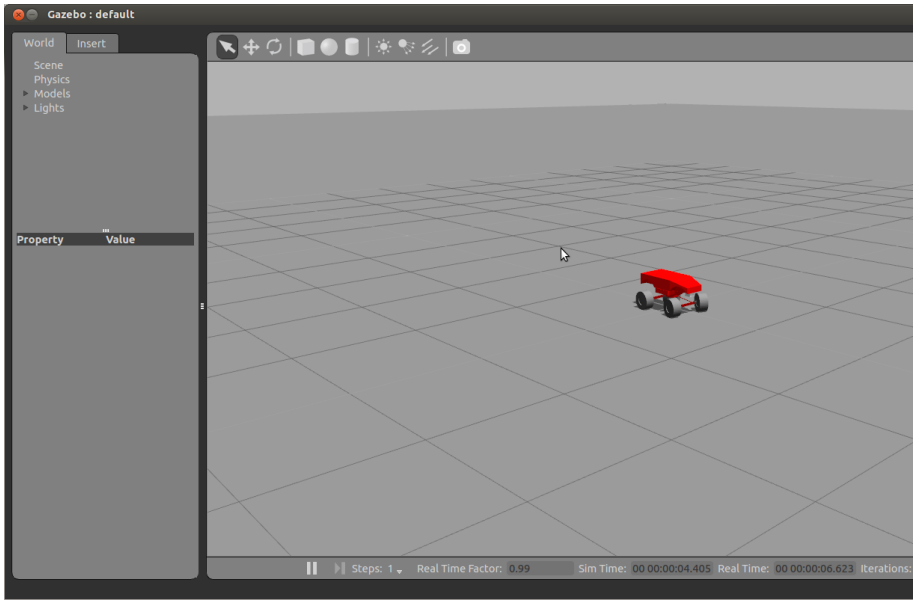

Figura 4.1: Entorno Gazebo con robot Summit

Una vez que se ha cargado el modelo en simulación, se debe ejecutar el controlador del robot, el cual proporciona los datos de la odometría virtual y permite mandar referencias de velocidad a los motores. Para ello, se ejecuta *summit_robot_control.launch* incluido en la carpeta *summit_robot_control*. Terminado este paso, el robot ya estaría listo para ser controlado.

La teleoperación, tanto del robot real como el modelo en simulación, puede realizarse por medio del *Gamepad* proporcionado junto al robot o mediante la ventana de teleoperación disponible en la herramienta Rviz. En este caso, se va a teleoperar el robot mediante la segunda opción, la herramienta Rviz. Por tanto, se ejecuta Rviz mediante el comando *rosrun rviz rviz*, que inicia el programa y permite añadir los nodos que se deseen. En la figura 4.2 se puede ver el modelo del robot en Rviz y los nodos añadidos. En este caso se han añadido todos los nodos necesarios para controlar el robot de forma autónoma, pero no se va a disponer de ellos aún. El modelo del robot en simulación se representa con un color gris, mientras que el modelo del robot real se representa en rojo. El cuadro inferior izquierdo muestra la ventana de teleoperación, que permite controlar el robot con el ratón del ordenador pulsando en distintas zonas de la cuadrícula y arrastrando. Esta ventana de teleoperación es genérica, es decir, sirve para cualquier robot, por lo que, en el apartado *Output Topic* se debe introducir el tópico al que se le envían las referencias de velocidad en el robot, que es */cmd_vel*.

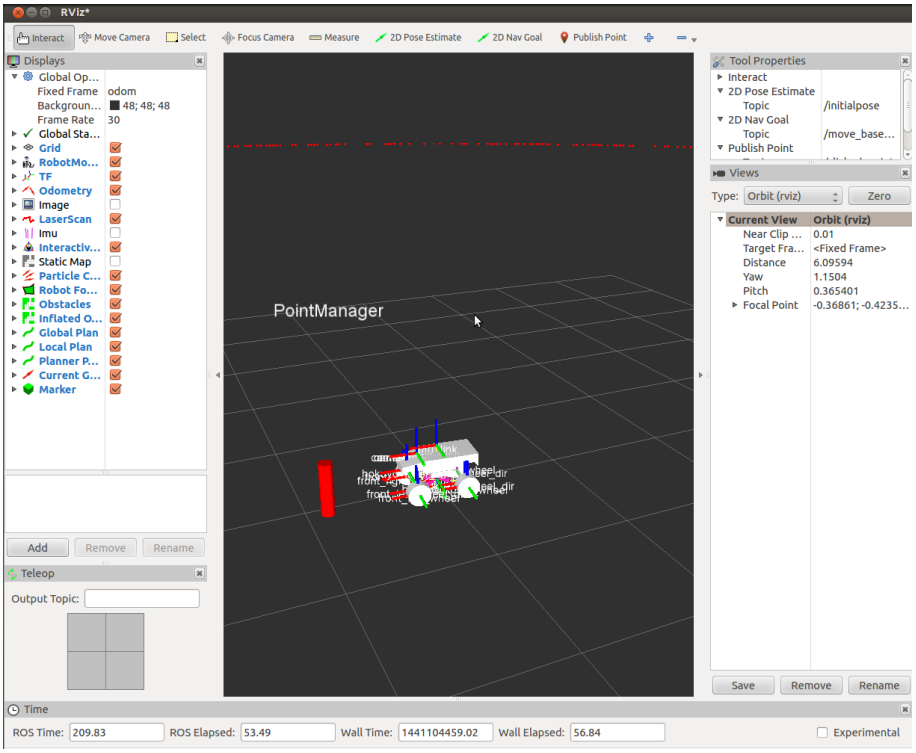

Figura 4.2: Interfaz RVIZ con robot Summit

Al emplear el cuadro de teleoperación en Rviz, el modelo del robot se desplaza por la interfaz tanto de Rviz como en Gazebo. La pantalla de Rviz muestra la información que proporcionan los nodos, como el punto de origen donde se inició el robot, la trayectoria que deja a su paso o la forma de obstáculos que tiene alrededor. Si se creara un mundo en Gazebo (por ejemplo, un entorno de oficina con sillas y mesas como obstáculos) en la pantalla de Rviz no se podrían ver estos objetos definidos, ya que sólo representa información de los nodos. Por lo tanto, se debe incluir un sensor de distancia que muestre los objetos del entorno que rodea al robot o incluir un mapa 2D detallado con la posición de todos los objetos fijos del entorno.

4.1.3 Navegación autónoma en simulación

Comprobado el funcionamiento de la teleoperación mediante Rviz se procedió a realizar una prueba de navegación autónoma. Para ello, la empresa Robotnik© proporcionó una carpeta llamada *robotnik_pp_planner* que implementa un navegador basado en el algoritmo Pure Pursuit, es decir, el seguimiento de metas establecidas punto a punto. Este método de navegación permite introducir un punto en la ventana de Rviz para que el robot se desplace desde su posición actual hasta dicho punto.

Si se introducen varios puntos, el robot iniciará una ruta desde su posición hasta el último punto colocado, pasando por el resto de ellos. Este método de seguimiento puro no tiene en cuenta los obstáculos del entorno, por lo que, el robot puede colisionar con ellos mientras se desplaza hacia un objetivo. Por ello, se debe realizar una ruta de puntos cercanos entre sí, evitando los obstáculos.

Para iniciar el método de navegación se deben ejecutar los dos archivos *launch* incluidos en la carpeta *robotnik_pp_planner*, cuyos nombres son *purepursuit.launch*, que inicia el nodo principal, y *purepursuit_marker.launch* que inicia el nodo de la herramienta visual para crear marcas en Rviz.[1]

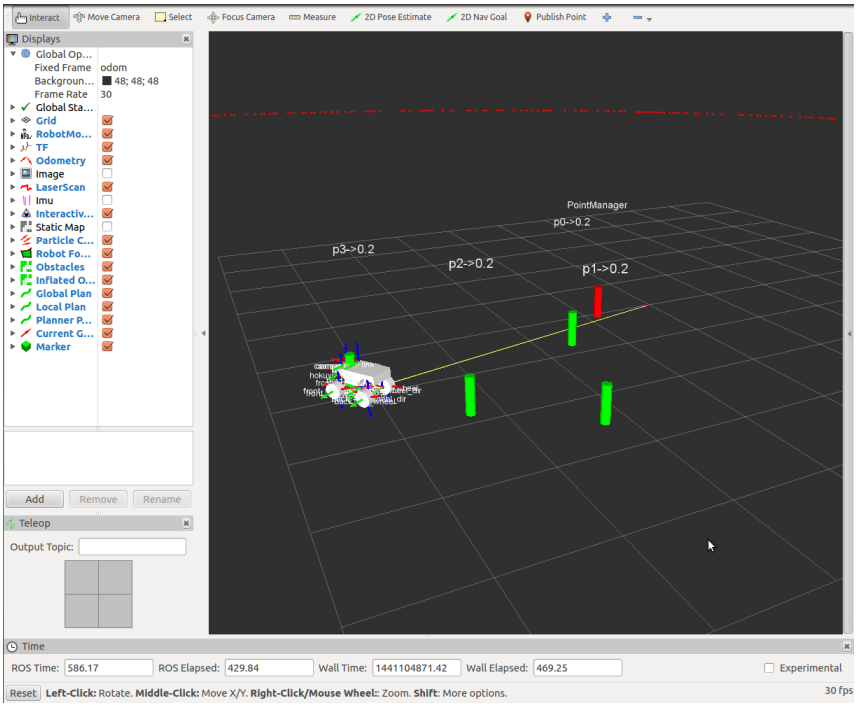

Figura 4.3: Navegación autónoma mediante Pure Pursuit en simulación

En la figura 4.3 se puede ver una serie de objetivos puestos en el entorno de Rviz. La herramienta que permite introducir los puntos se llama *PointManager* y para crear un nuevo punto basta con hacer click derecho en la herramienta y seleccionar la velocidad con la que el robot se desplazará hasta el objetivo. Se creará un cilindro verde representando la meta, que se podrá colocar en cualquier parte del entorno. Para ejecutar el inicio de la navegación se debe hacer click derecho sobre la última marca y seleccionar *Go*.

[1]Navegación Pure Pursuit: https://youtu.be/qsenjoseiqY

4.2 Ensayos con el robot real

Los resultados mostrados en este apartado representan pruebas y ensayos realizados sobre el robot móvil Summit, al que se le ha incluido un sensor lidar montado en la carcasa superior (figura 4.4). Se ha creado una carpeta interna en el espacio de trabajo ROS del robot denominada *launchs* donde se encuentran todos los archivos ejecutables creados para realizar los ensayos de SLAM y de navegación autónoma.

Figura 4.4: Robot móvil Summit

Como objetivo final de este proyecto se propuso realizar una comparación entre los métodos de navegación planificada y reactiva pura para comprobar las ventajas y desventajas que ofrecía cada método en un robot con forma rectangular y configuración Ackerman. El ensayo consiste en la navegación autónoma del robot Summit desde una posición inicial hasta una meta dada evitando los obstáculos que se encuentre en el camino. Se evaluará tanto la duración de la prueba como la trayectoria seguida para alcanzar la meta, así como, el comportamiento que muestre el robot en velocidad y dirección con cada método.[2]

[2]Navegación planificada: https://vimeo.com/133033742
Navegación reactiva: https://vimeo.com/133034961

4.2.1 Teleoperación

La primera prueba que se realizó con el robot real fue de teleoperación mediante el *Gamepad* proporcionado. Al encender el robot, se debe ejecutar *summit_complete.launch* contenido en la carpeta *summit_robot/summit_complete* que inicia varios nodos conjuntamente: el controlador del robot, la cámara, el sensor IMU y el control mediante *Gamepad*. El *Gamepad* se comunica con el robot mediante un receptor Bluetooth USB y permite enviarle diversos comandos de velocidad al robot o de control de la cámara (mostrados en el capítulo 3).

Se realizó una prueba de control en el descampado adyacente al edificio CITE IV, comprobando la resistencia del robot, su radio de giro y la autonomía de la batería. El robot poseía una configuración Ackerman de fábrica, pero tras una modificación posterior, se le instaló una servo-dirección adicional en el eje trasero (configuración doble Ackerman), permitiendo realizar giros más cerrados. Se adaptó el código del *Gamepad* para permitir cambiar la configuración del robot en tres modos distintos:

- ***Configuración Ackerman único:*** Únicamente tiene giro el eje delantero del robot.

- ***Configuración doble Ackerman orientado (Modo Paralelo):*** Ambos ejes del robot giran en el mismo sentido, imposibilitando el giro pero implementando un desplazamiento lateral simultáneo al de avance o retroceso del robot.

- ***Configuración doble Ackerman invertido:*** Los dos ejes del robot giran en sentidos opuestos, reduciendo el ángulo de giro.

La teleoperación del robot Summit es un aspecto muy importante a la hora de cumplir los objetivos propuestos, ya que, para poder realizar pruebas de SLAM es necesario controlar el robot de forma manual por la zona que se desea modelar. Por ello, las pruebas de teleoperación pudieron determinar las velocidades adecuadas con las que debe desplazarse el robot. A altas velocidades, se produce mayor error en la información de la odometría, debido a que se produce mayor deslizamiento entre las ruedas y la superficie del suelo, por lo que se optó por realizar los ensayos de SLAM y navegación autónoma a una velocidad inferior al punto medio de la escala de velocidad. En cuanto a la velocidad de giro, se limitó también para impedir que el robot realizara giros a altas velocidades, evitando que se dañaran los ejes de las ruedas. Una vez realizadas las pruebas de control manual e instalado el sensor lidar, se procedió con los ensayos de mapeado de entornos.

4.2.2 Ensayos de SLAM

Para los ensayos de SLAM se han empleado dos métodos. Se comenzó utilizando *gmapping*, un nodo de ROS que crea un mapa 2D a partir de la información de un sensor lidar y la odometría proporcionada por el robot. Este nodo estaba incluido en las carpetas preinstaladas del robot y ya se encontraba configurado para el robot Summit, pero tras una serie de pruebas se optó por emplear otro método debido al tiempo excesivo en la actualización de los datos del mapa y a los errores en la odometría. Los mapas que se obtenían no cerraban en el punto inicial de comienzo, por tanto, no representaban fielmente el entorno mapeado. Se realizaron varias pruebas de SLAM mediante le nodo *gmapping* en el interior de la oficina de trabajo (sala cuadrada con un obstáculo en el medio). El ciclo de mapeado consistía en dar la vuelta a la oficina y terminar en el punto de comienzo. Sucesivos ensayos demostraron que la representación 2D del entorno no terminaba en la posición inicial del robot, por lo que se buscó un método alternativo, siendo éste el paquete *hector_slam*.

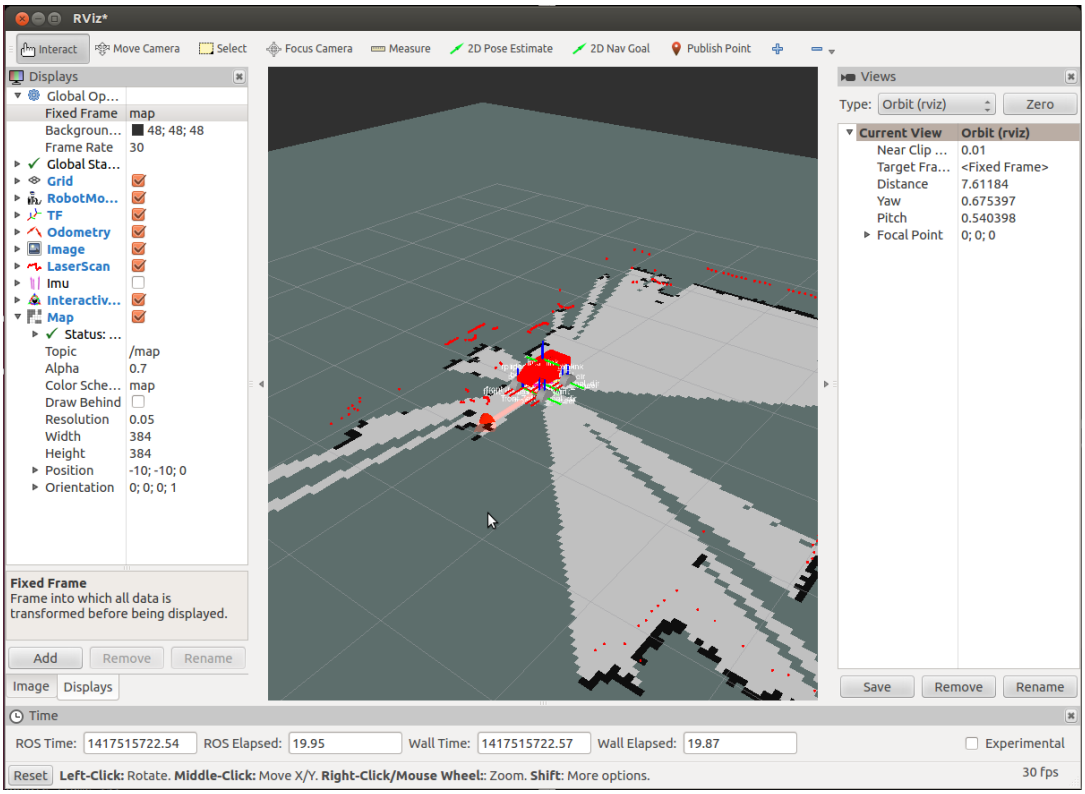

Figura 4.5: Proceso de SLAM

Como se ha mencionado anteriormente, el segundo método empleado para la construcción de mapas fue *Hector Slam*, un nodo similar a *gmapping* pero más sencillo y completo. La ventaja frente al anterior reside en la velocidad de actualización del mapa y en un número reducido de parámetros configurables. En la figura 4.5 se puede ver una demostración de la ventana de Rviz durante el proceso de mapeado. Las zonas libres se tornan de un color gris, mientras que los límites marcados por el sensor lidar se tornan de un color negro, formando paredes. Se realizaron mapas completos de la oficina de trabajo y el pasillo del edificio. Se intentó realizar un mapa del interior de un invernadero, resultando en un ensayo fallido debido al desnivel del terreno. El proceso de SLAM sólo es válido para entornos relativamente planos y sin deslizamiento, ya que los errores en la odometría aumentan con el deslizamiento del terreno. Además, el sensor láser varía la información del entorno mapeado si el terreno es excesivamente irregular porque el plano de mapeado del sensor dejaría de ser paralelo respecto al plano del terreno.

Una vez completado el mapa, se guarda mediante el comando:

```
rosrun map_server map_saver -f [nombre del mapa]
```

Para utilizar los mapas guardados en los métodos de navegación autónoma basta con seleccionar la ruta y el nombre del mapa dentro del parámetro de configuración correspondiente en el archivo *launch* de los nodos de navegación.

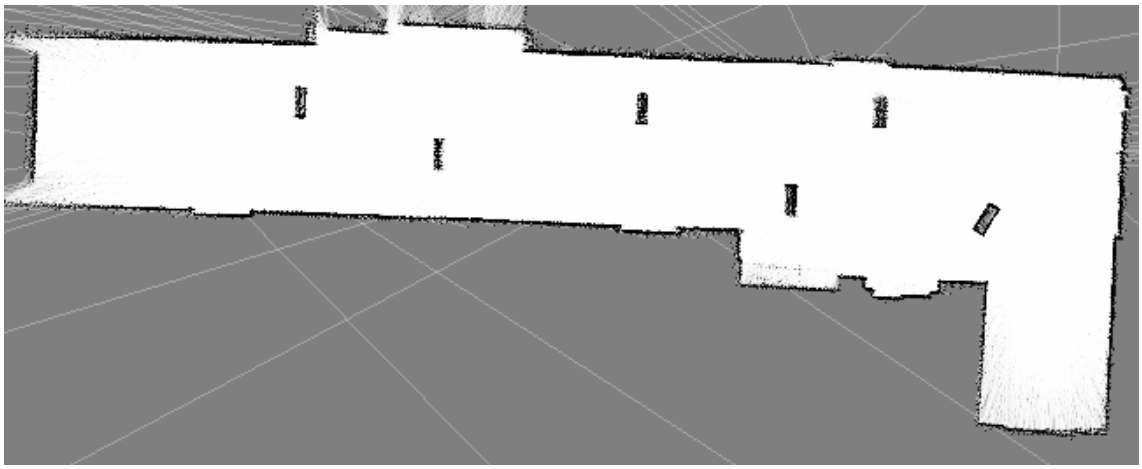

Figura 4.6: Mapa de ensayo

La figura 4.6 representa el mapa 2D del circuito de obstáculos creado para llevar a cabo el ensayo de navegación autónoma comparativa. El circuito de ensayo está creado colocando una serie de obstáculos rectangulares en el pasillo de la planta baja del edificio CITE IV.

4.2.3 Ensayos de navegación autónoma

Para realizar las pruebas de navegación autónoma y cumplir los objetivos de este proyecto, se disponía de tres métodos de navegación: un método basado en seguimiento de puntos (Pure Pursuit), un método de navegación planificada (*navigation*) y un método de navegación reactiva pura (*mrpt_navigation*).

Método Pure Pursuit

Debido a problemas iniciales de comunicación en el método de navegación planificada, se comenzó a trabajar con el método Pure Pursuit, proporcionado por Robotnik© junto al controlador del robot de la versión ROS Hydro. Este método permitía realizar una navegación por puntos sin necesidad de un mapa, opción que no era viable, ya que no se podía controlar la distancia a la que estaba el robot de los objetos de todo el entorno. Por ello, este método se empleó junto a diversos mapas de zonas de trabajo, en los que era necesario construir el camino que debía seguir el robot mediante puntos sucesivos entre sí. Este método terminó descartado, ya que la trayectoria seguida por robot estaba determinada por el usuario, mientras que en los otros métodos, la ruta es calculada por el propio robot. El comportamiento que muestra el robot en el seguimiento de los puntos es continuo y preciso, trazando arcos circulares hasta llegar al objetivo, implementando incluso un movimiento de retroceso.

Navegación planificada

El método de navegación planificada por defecto en ROS es el paquete *navigation*. El nodo de navegación se denomina *move_base* y se ejecuta conjuntamente con el localizador AMCL, que carga el mapa sobre el que se desea navegar. Para ejecutar el navegador se parte del nodo central del robot iniciado y se ejecuta el archivo *move_base* de la carpeta *launchs*, que se encuentra configurado para el robot Summit.

Las metas de navegación se establecen desde la ventana de Rviz, pulsando la opción superior *2D Nav Goal*, que permite colocar una flecha en el entorno, indicando la meta deseada y la posición final del robot (figura 4.7).

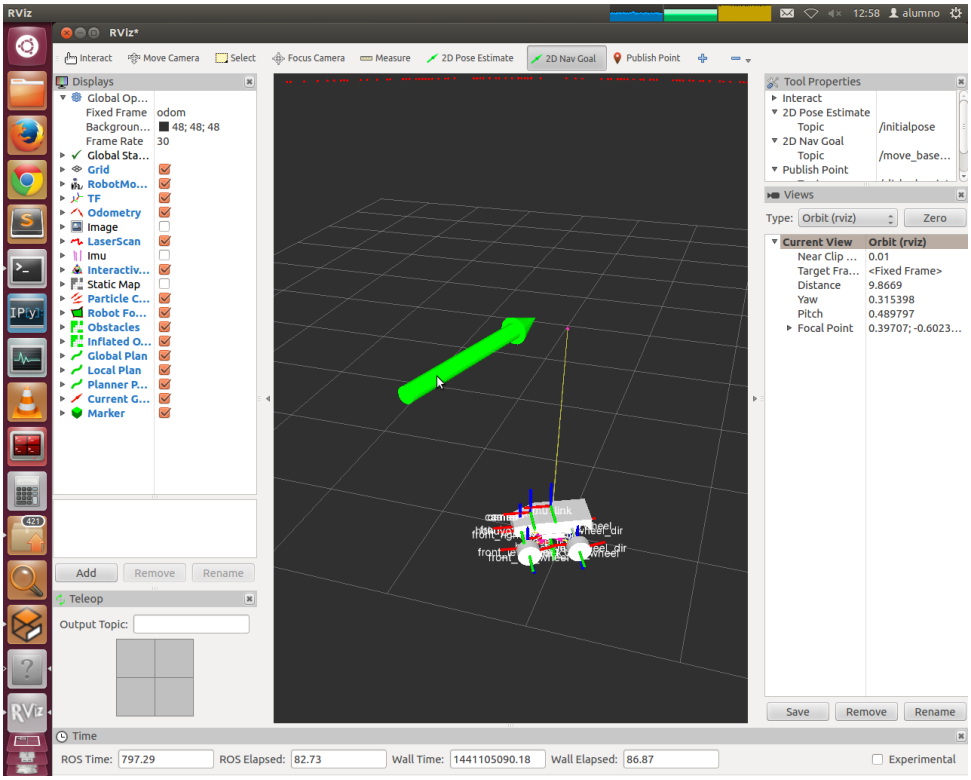

Figura 4.7: Selección de meta en Rviz

Para este ensayo se ha utilizado "*global_planner*" como el planificador global, el cual genera una trayectoria representada por una línea desde la posición del robot hasta la meta, y "*dwa_local_planner*" como planificador local, cuyo objetivo es descomponer la trayectoria global en submetas y controlar el robot mediante arcos circulares para seguir la línea representada por la trayectoria global.

En la figura 4.8 se puede observar la trayectoria global calculada por *global_planner* en rojo. El recuadro blanco que rodea al robot representa su zona local, controlada por el planificador local.

El resultado del ensayo determinó que el robot se mueve de forma intermitente, es decir, la velocidad lineal cambia constantemente e impide que el robot se mueva de forma continua. Además, al ser una navegación planificada con una ruta a seguir representada por una línea, la servo-dirección se mueve rápidamente en ambos sentidos tratando de seguir esa línea. Esto se debe a que el método de navegación esta diseñado como un seguidor de líneas para robots con configuración diferencial.

Figura 4.8: Trayectoria global

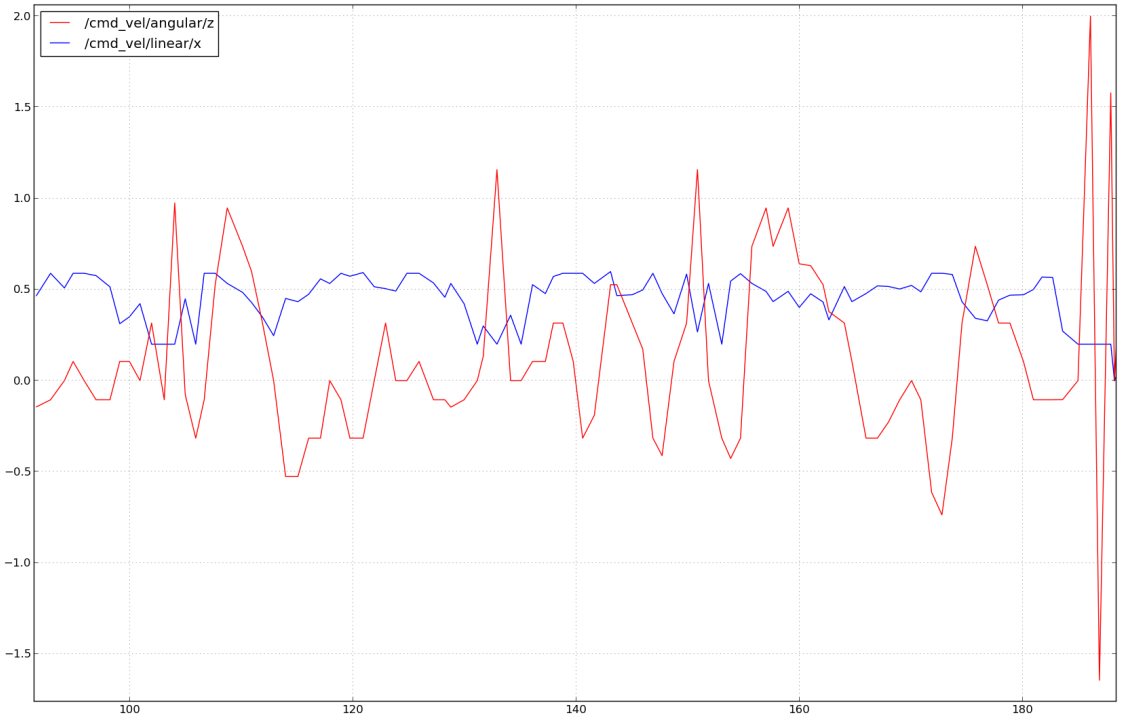

Figura 4.9: Representación de las velocidades (Navegación planificada)

La figura 4.9 representa la velocidad lineal y angular del robot durante el ensayo obtenidas gráficamente a partir de *rqt* un paquete de aplicaciones gráficas para manejo de datos en ROS. La velocidad angular se representa en azul mientras que la velocidad angular (giro) se representa en rojo.

Se ha limitado la velocidad lineal del robot a un máximo de 0.7 m/s y la velocidad angular a un máximo de 2 rad/s para evitar problemas en la navegación debidos a aceleraciones puntuales.

El robot completó el ensayo en un tiempo de 96.66 segundos, tiempo determinado a partir de la gráfica de velocidades.

Navegación reactiva pura

Para la navegación reactiva en este ensayo se ha utilizado el paquete *mrpt_navigation* con dos tipos de PTGs: trayectorias criculares (C) y trayectorias con rumbo asintótico (α-A) [6]. Como método de evasión de obstáculos se ha empleado el *campo de fuerzas potenciales* (VFF). Además, este método utiliza un filtro que permite cambiar la referencia de velocidad del robot de forma progresiva, manteniendo un movimiento constante sin aceleraciones repentinas.

Antes de ejecutar el método se debe iniciar previamente el nodo central del robot y después ejecutar el archivo *mrpt_localization* que carga el mapa del entorno y el localizador. Tras estos pasos, se inicia el navegador reactivo ejecutando el archivo *reactive_nav* contenido en la carpeta *launchs* y configurado correctamente para el robot Summit.

La figura 4.10 muestra la trayectoria seguida por el robot durante el ensayo. Se observa como el robot traza curvas suaves para evitar los obstáculos, manteniendo un movimiento continuo. Cuando el robot se aproxima a un obstáculo disminuye la velocidad, comportamiento característico del método de evasión de obstáculos VFF.

En la figura 4.11 están representadas las velocidades lineal y angular para el ensayo de navegación reactiva. En este caso, la velocidad lineal se mantiene constante en torno a 0.5 m/s la mayoría del tiempo y la velocidad angular solo cambia cuando el robot necesita evadir un obstáculo o modificar su orientación.

El robot completó el ensayo en 41.63 segundos, tiempo obtenido también a partir de la gráfica de velocidades.

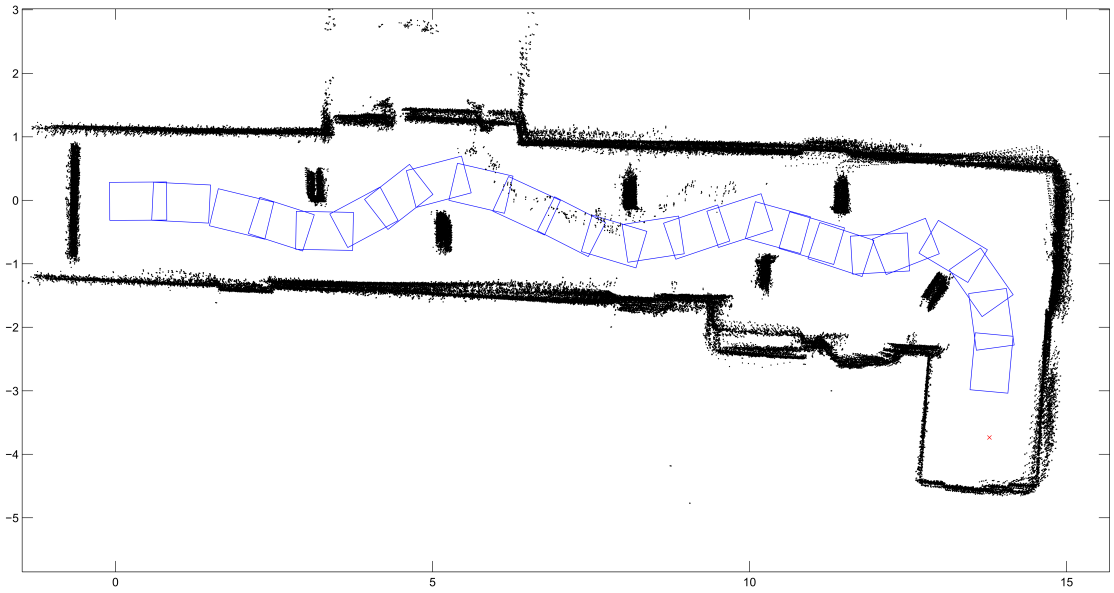

Figura 4.10: Trayectoria seguida mediante navegación reactiva

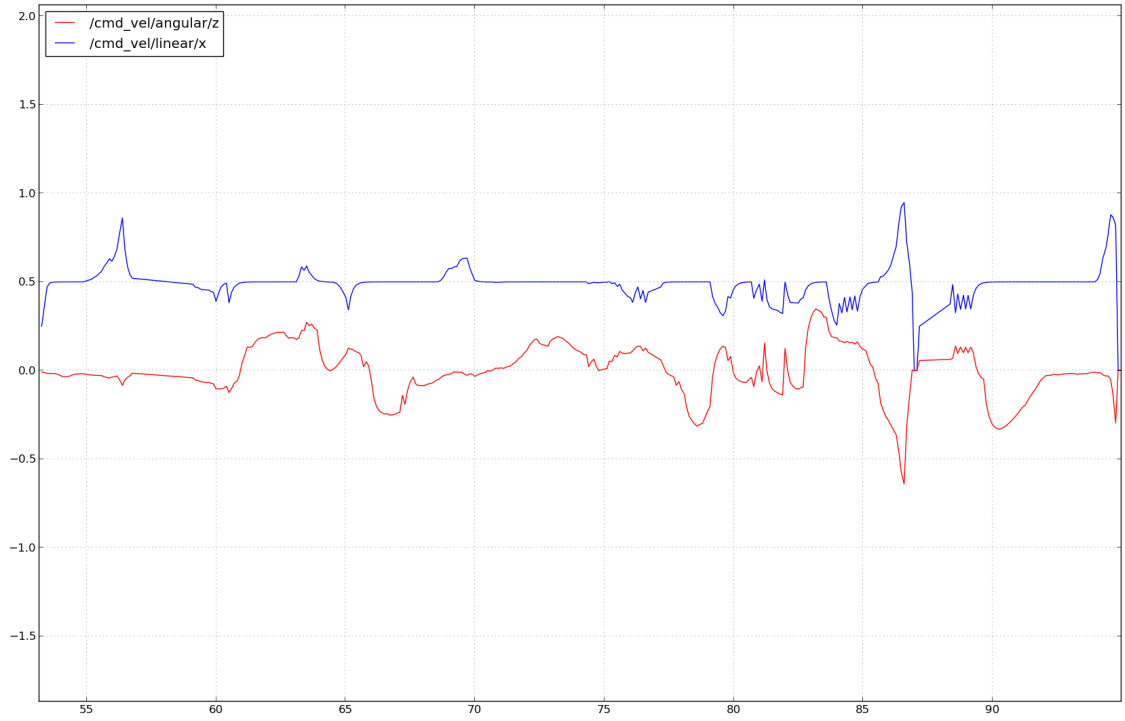

Figura 4.11: Representación de las velocidades (Navegación reactiva)

4.2.4 Localización del robot en el entorno

Ambos métodos de navegación mostrados en el ensayo anterior cuentan con un nodo de localización en el mapa. El robot cuenta con la odometría para calcular el desplazamiento que realiza sobre el suelo, pero este cálculo esta sujeto a errores que pueden desvirtuar en gran medida la posición real del robot. Por ello, existe un nodo de localización basado en el algoritmo AMCL que se utiliza para corregir los errores producidos en la odometría. Este método se basa en comparar las lecturas del sensor lidar con los límites representados en el mapa (comparar la esquina de una pared con las lecturas del sensor, por ejemplo), para determinar la posición probable del robot, mostrada con un haz de vectores rojos tal y como representa la figura 4.12. Cuanto más detallado sea el mapa, mejor será la localización realizada. El paquete *navigation* incorpora el localizador AMCL por defecto, mientras que el paquete *mrpt_navigation* implementa un localizador basado en el algorimto AMCL que soporta distintos filtros de particulas y mapas, llamado *mrpt_localization*.

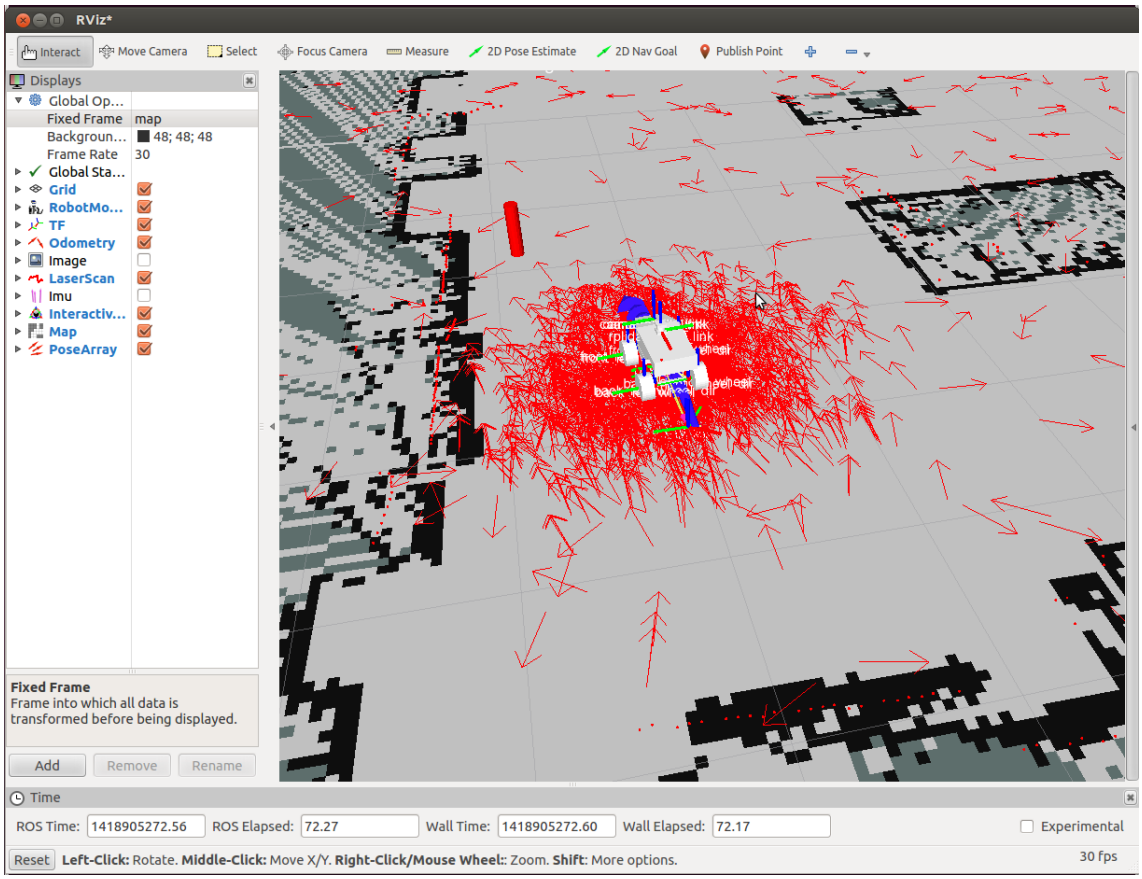

Figura 4.12: Localización mediante AMCL

Capítulo 5: Incidencias, conclusiones y líneas de trabajo futuras

Este proyecto se ha elaborado con el objetivo principal de llevar a cabo pruebas de navegación autónoma en un robot móvil. Para ello, se ha utilizado el robot móvil Summit, partiendo del aspecto más básico de control, la teleoperación. A continuación se muestran las incidencias ocurridas a lo largo del desarrollo del proyecto y las conclusiones de los ensayos realizados. Para finalizar se exponen una serie de objetivos futuros que se podrían realizar con la plataforma móvil.

5.1 Incidencias

En el proceso de desarrollo del proyecto han surgido diversos problemas cuya solución ha sido necesaria para progresar en el cumplimiento de los objetivos establecidos, suponiendo un incremento de tiempo considerable en la realización de este trabajo.

5.1.1 Versión de ROS obsoleta

La versión preinstalada dentro del robot Summit cuando se comenzó el desarrollo del proyecto correspondía a ROS Fuerte, una versión obsoleta que no estaba operativa, por lo que fue necesario reinstalar una versión posterior (ROS Hydro) y adaptar los nodos a este software. La empresa proporcionó el nodo del controlador y el *Gamepad*, así como la carpeta de simulación, pero el resto de componentes se tuvieron que adaptar para funcionar en la nueva versión. El nodo del *Gamepad* no permitía cambiar la configuración de las ruedas del robot, por lo que se tuvo que modificar para mejorar el control. Otros nodos que se tuvieron que adaptar a esta versión fueron el nodo del sensor IMU, el nodo del GPS y los nodos de navegación.

5.1.2 Documentación insuficiente sobre el robot

Durante la revisión de la documentación referente al robot Summit, se comprobó que la información proporcionada en la web referente al robot en ROS era nula y la documentación escrita proporcionada por la empresa contenía errores y

estaba creada para la versión obsoleta de ROS Fuerte, por lo que fue necesario invertir gran cantidad de tiempo en la revisión de los nodos que contenía el robot y su funcionamiento.

5.1.3 Problema en el modelo URDF del robot

Al realizar las pruebas de simulación con el robot o durante los ensayos de teleoperación del robot real, surgía un error con el modelo del robot en la herramienta Rviz (figura 5.1), que impedía representarlo correctamente, por lo que el modelo del robot no se desplazaba por el entorno y resultaba imposible determinar su movimiento. Fue necesario un estudio exhaustivo del código URDF del robot y varias pruebas para lograr corregir los errores, debidos a problemas de conexión en los enlaces de las ruedas.

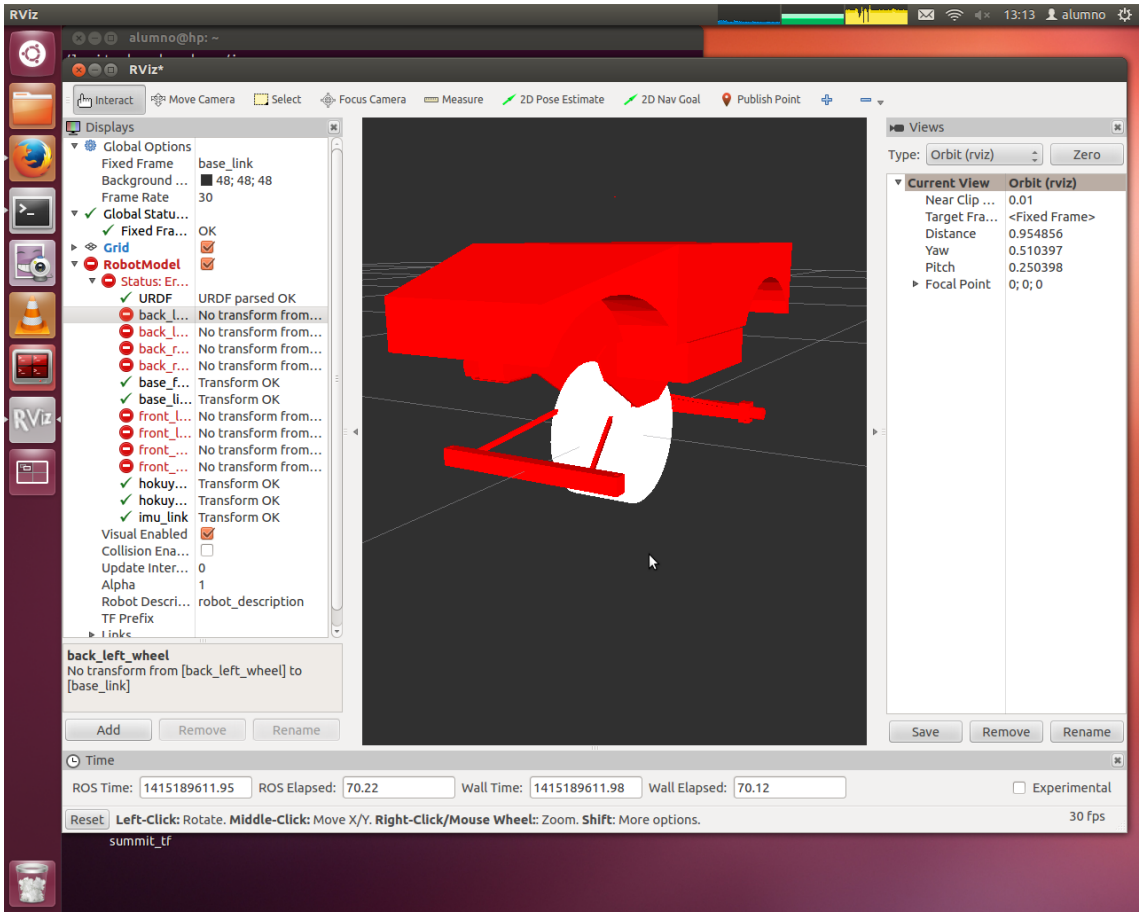

Figura 5.1: Problema en el modelo URDF del robot

5.1.4 Estructura delicada del sistema de giro de las ruedas

El robot se desplaza normalmente con estructura doble Ackerman invertida, lo que le permite girar las ruedas de ambos ejes en sentido contrario, realizando giros cerrados. Si estos giros se realizan a máxima velocidad, la unión de la rueda con el eje sufre demasiado esfuerzo y el pasante que une la rueda al sistema de giro se parte, desacoplando la rueda del giro del motor. Este problema reside en un mal diseño de la unión del eje de la rueda con el eje del motor. Esta unión permite que la rueda gire junto al eje del motor, permitiendo además un giro lateral guiado por la servo-dirección. Es posible que esta unión se bloquee cuando el robot realiza un giro cerrado a alta velocidad, produciéndose la rotura del pasante. Antes este problema ha sido necesario reparar todas las ruedas del robot y mantener especial atención en la velocidad del robot durante la navegación.

5.1.5 Problema de comunicación al establecer metas en las pruebas de navegación autónoma

Al realizar las primeras pruebas de navegación autónoma con el robot Summit, resultaba imposible establecer una meta de navegación. La información del robot se mostraba en la ventana de la herramienta Rviz y el nodo de navegación funcionaba correctamente, pero al seleccionar un punto en el mapa, el robot no se desplazaba. La ayuda proporcionada por la empresa no lograba solucionar este problema, por lo que fue necesario revisar información referente a otros robots móviles hasta identificar el problema como un fallo en la comunicación del ordenador remoto con el robot. El fallo se solucionó exportando la dirección IP del ordenador remoto en el documento *bashrc* de ROS perteneciente a este ordenador. Este problema permitía que el ordenador remoto pudiera recibir datos de los sensores del robot pero impedía enviar datos desde el ordenador hacia el robot, como las coordenadas de las metas de navegación.

5.2 Conclusiones

Una vez finalizados los ensayos se puede llegar a la conclusión de que ambos métodos cuentan con diferencias significativas, así como algunas semejanzas.

La trayectoria trazada por el robot en la navegación planificada, calculada mediante *global_planner*, es similar a la trayectoria realizada por el robot durante la navegación reactiva, manteniendo cierta distancia respecto a los obstáculos. A parte de esta semejanza, la trayectoria realizada realmente por el robot durante la navegación planificada no ha sido precisa.

La velocidad angular representada en el diagrama de la figura 4.9 muestra picos de desviación considerables, debido al seguimiento de la línea de la trayectoria. Los picos que se producen al final del ensayo en dicha velocidad corresponden al movimiento de las ruedas tratando de seguir la línea cuando la velocidad lineal es nula. El robot no detiene el giro hasta pasados unos segundos de alcanzar la meta, debido a la tolerancia impuesta.

Durante el ensayo de navegación reactiva, el robot ha mantenido un movimiento continuo, evadiendo todos los obstáculos y alcanzando la meta de forma precisa. Su diagrama de velocidades muestra una valor casi constante en la velocidad angular, excepto en el tramo final del mapa (último obstáculo) en el que el robot acelera para completar el giro cerrado que debe realizar. Este punto también coincide con el punto de máxima velocidad angular, debido al giro cerrado anteriormente mencionado. La evolución de la velocidad angular es suave, solo cambiando cuando el robot debe evadir un obstáculo.

Referente a las velocidades en ambos navegadores, se observa como la velocidad lineal se mantiene más constante en el ensayo de navegación reactiva que en la navegación planificada, así como el movimiento continuo del robot. Comparando las velocidades angulares también se comprueba que los cambios son más suaves en la navegación reactiva y los picos que se producen no afectan al movimiento del robot.

Finalmente, comparando el tiempo de ensayo, la navegación reactiva alcanza la meta en la mitad de tiempo respecto al ensayo de navegación planificada.

En resumen a las comparaciones anteriores, se demuestra que la navegación planificada, independientemente de encontrar la trayectoria óptima hasta la meta, depende fuertemente de un sistema de navegación local que permita seguir la trayectoria calculada de forma precisa. Por otra parte, la navegación reactiva proporciona un sistema de navegación continuo y preciso pero carece de una ruta óptima precalculada sobre un mapa, es decir, una navegación reactiva pura no asegura encontrar el camino mas corto o apropiado hacia una meta, ya que, podría llegar a un callejón sin salida al no tener en cuenta el mapa de la zona.

Teniendo en cuenta que ambos métodos de navegación presentan ventajas únicas, la solución óptima para mejorar la navegación autónoma de este robot móvil sería la implementación de un método híbrido que combinara las ventajas expuestas anteriormente, es decir, una navegación reactiva en tiempo real sobre un mapa del entorno, teniendo en cuenta posibles trayectorias que puede realizar el robot.

En cuanto a los ensayos realizados con el método Pure Pursuit, lo más destacable reside en el comportamiento preciso del robot. El movimiento que implementa el algoritmo Pure Pursuit se basa en arcos circulares para calcular la circunferencia que debe seguir el robot para alcanzar un objetivo. Por supuesto, si la meta se encuentra en linea recta respecto al robot, éste no realizará un movimiento circular, sino que abordará la meta de forma lineal.

Referente a la simulación del robot, la herramienta de simulación Gazebo no cuenta con la información necesaria para crear entornos de forma sencilla. Se intentaron crear mapas en entornos de oficina, resultando una tarea imposible por falta de información en cuanto a uso de la herramienta o librerías de objetos aplicables.

5.3 Líneas de trabajo futuras

En base al trabajo realizado, una vez completados los objetivos propuestos se ha llegado a la conclusión que existen diversos objetivos adicionales que se pueden llevar a cabo en el robot móvil Summit, mostrados a continuación.

5.3.1 Método de navegación híbrido

Como se ha comentado en las conclusiones, el mejor método de navegación que se podría aplicar en el robot sería un método híbrido, pero ROS no cuenta con un sistema estándar de este tipo, por lo que, como proyecto futuro se podría implementar un método de navegación basado en la conjunción de los sistemas mostrados en el ensayo.

5.3.2 Nuevo planificador local para el paquete *navigation*

El inconveniente de este método, como se ha mencionado anteriormente reside en la imposibilidad de detección de obstáculos por parte del método en sí. El movimiento que produce el planificador local en el paquete *navigation* de navegación reactiva deja mucho que desear en cuanto a seguimiento de trayectorias, por lo que se podría realizar una implementación del algoritmo Pure Pursuit adaptada como navegador local para el paquete *navigation*.

5.3.3 Mapeado de exteriores

Otra línea de trabajo futuro reside en la posibilidad de crear mapas de entornos exteriores. El sensor RPlidar del que dispone el robot esta diseñado para procesos de SLAM en interiores, pero la Universidad de Almería cuenta con sensores de distancia para exteriores, los cuales se podrían usar con en el robot. Para ello, bastaría con crear un nodo ROS que publicara la información del sensor en un tópico.

5.3.4 Navegación autónoma basada en GPS

Para finalizar, y aprovechando la línea de trabajo anterior. Si el robot dispusiera de mapas de entornos exteriores se podrían realizar pruebas de navegación autónoma con posicionamiento mediante GPS, siendo este posicionamiento absoluto, descartando la odometría y los errores ocasionados en su cálculo debido al terreno.

Capítulo 6: Bibliografía

Referencias

[1] Entorno de simulación ros/gazebo. `http://bibing.us.es/proyectos/abreproy/5139/fichero/PFC-+Por+cap%EDtulos%252F3-Entorno+de+simulaci%F3n.pdf`. [Online, consulta 5-12-2014].

[2] Navegación en robots móviles. `http://webpersonal.uma.es/~VFMM/PDF/cap2.pdf`. [Online, consulta 23-11-2014].

[3] Robots móviles. `http://platea.pntic.mec.es/vgonzale/cyr_0708/archivos/_15/Tema_5.5.htm`. [Online, consulta 1-11-2014].

[4] Robots móviles: diseño. `http://platea.pntic.mec.es/vgonzale/cyr_0204/cyr_01/robotica/movil.htm`. [Online, consulta 28-10-2014].

[5] Robots móviles: diseño. `http://www.xatakaciencia.com/robotica/robots-moviles-i`. [Online, consulta 28-10-2014].

[6] Anónimo. A ros reactive navigation system for ground vehicles based on tp-space transformations. *No Institute Given*, pages 1–9, 2015. [Consulta 27-7-2015].

[7] J.A. Bañares. Comunicación entre agentes. `http://webdiis.unizar.es/asignaturas/ISBC/lecciones/10.ConversacionesAgentes.pdf`. [Online, consulta 5-11-2014].

[8] J.L. Blanco et al. Mobile robot programming toolkit (mrpt). `http://www.mrpt.org/`, 2015. [Online, consulta 23-2-2014].

[9] Y. Koren, J. Borenstein. Potential field methods and their inherent limitation for mobile robot navigation. *IEEE International conference on robotics and automation.*, 1:1398–1405, 1991. [Consulta 20-2-2014].

[10] R. Craig. Implementation of the pure pursuit path tracking algorithm. *The Robotics Institute Carnegie Mellon University*, pages 1–15, 1992.

[11] Departamento de física aplicada III, Universidad de Sevilla. Sistemas de locomoción de robots móviles. `http://www.esi2.us.es/~vivas/ayr2iaei/LOC_MOV.pdf`. [Online, consulta 29-10-2014].

[12] J. Borenstein, H.R. Everett, L. Feng. Navigating mobile robots: Sensors and techniques. *A. K. Peters, Ltd., Wellesley, MA*, pages 1–282, 2009. [Consulta 11-5-2014].

[13] A. Martinez, E. Fernández. Learning ros for robotics programming. *PACKT Publishing*, pages 1–438, 2013. [Consulta 5-12-2014].

[14] J.L. Blanco, J. González, J.A. Fernández. Extending obstacle avoidance methods through multiple parameter-space transformations. *Autonomous Robots*, 24(1):29–48, 2008. [Consulta 17-2-2014].

[15] G. Dudek, M. Jenkin. Computational principles of mobile robotics. *Cambridge University Press*, pages 1–294, 2011. [Consulta 11-5-2014].

[16] Y. Maritza. Clasificación de los robots. `http://roboticapuno.blogspot.com.es/2013/01/clasificacion-de-los-robots.html`, 2013. [Online, consulta 2-11-2014].

[17] V. Perez, R. Mayta. Actualidad y perspectiva de la robótica. `http://sisbib.unmsm.edu.pe/bibvirtual/publicaciones/indata/v04_n1/actualidad.htm`, 2001. [Online, consulta 27-10-2014].

[18] J. Minguez, L. Montano. Nearness diagram navigation: Collision avoidance in troublesome scenarios. *IEEE Transactions on robotics and automation.*, 20(1):45–59, 2004. [Consulta 20-2-2014].

[19] E. Rich, K. Knight, S. B Nair. Artificial intelligence. *McGraw-Hill*, pages 1–640, 2009. [Consulta 5-11-2014].

[20] Robotnik. Robotnik automation s.l.l. `http://www.robotnik.es/`, 2002. [Consulta 25-6-2014].

[21] Robotnik. Documentación referente al robot summit, 2013. [Consulta 11-2-2014].

[22] ROS. Sistema operativo de robots. `http://www.ros.org/`. [Online, consulta 25-6-2014].

[23] R. Siegwart, I.R. Nourbakhsh, D. Scaramuzza. Introduction to autonomous mobile robots (2º edition). pages 1–335, 2011. [Consulta 11-5-2014].

[24] RoboPeak Team. Robopeak rplidar low cost 360 degree 2d laser scanner (lidar) system. introduction and datasheet. `http://www.robopeak.com/`, 2014. [Online, consulta 10-3-2015].

Nota: El derecho de las imágenes les corresponde a sus determinados autores.

Capítulo 7: Anexos

7.1 Siglas y acrónimos

En esta sección se muestran las abreviaturas utilizadas en este informe.

ROS: Robot Operating System (Sistema Operativo de Robots).

SLAM: Simultaneous Localization And Mapping (Lozalización y mapeado simultáneos).

ICC: Instantaneous Center of Curvature (Centro instantáneo de curvatura).

CCD: Charge Coupled Device (Dispositivo de acoplamiento de carga).

CPU: Central Processing Unit (Unidad Central de Procesamiento).

GPS: Global Positioning System (Sistema de Posicionamiento Global).

IMU: Inertial Measurement Unit (Unidad de Medida Inercial).

ODE: Open Dynamics Engine.

URDF: Universal Robotic Description Format (Formato de Descripción Universal de Robots).

AMCL: Adaptative Monte Carlo Localization (Localización Adaptativa de Monte Carlo).

MRPT: Mobile Robot Programming Toolkit (Herramientas de Programación de Robots Móviles).

VFF: Vector Force Field (Campo de Fuerzas Potenciales).

ND: Nearness Diagram (Diagrama de Cercanía).

7.2 Esquemas básicos

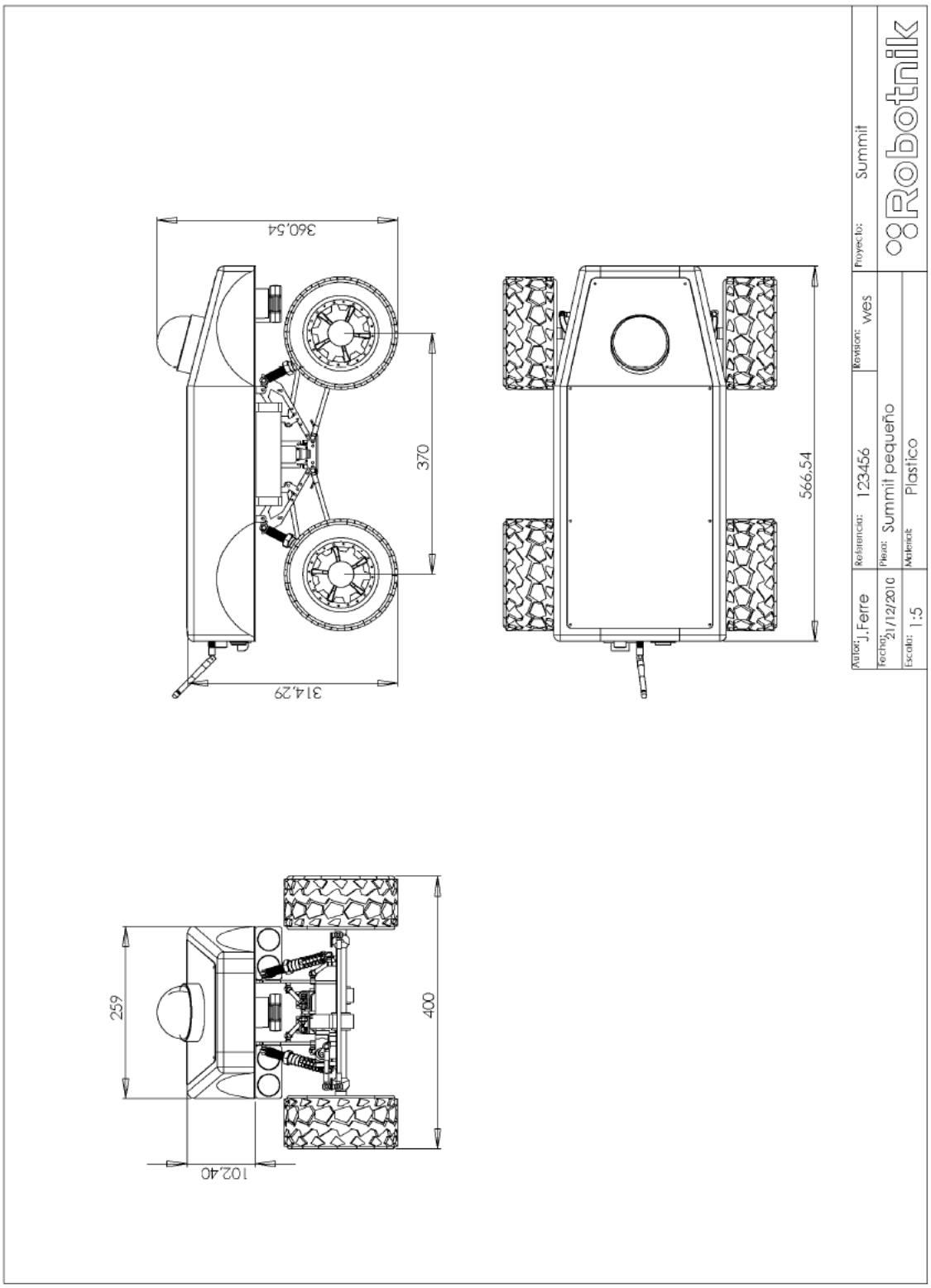

Figura 7.1: Esquemas básicos exteriores del robot