



UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y
DISEÑO INDUSTRIAL

Grado en Ingeniería Electrónica Industrial y Automática

TRABAJO FIN DE GRADO

GUIADO DE UN ROBOT MÓVIL BASADO EN ROS Y KINECT

Autor: Daniel Manzaneque Amo

Cotutor: Dr. Miguel Hernando

Gutiérrez

Departamento: Electricidad,
Electrónica, Automática y Física
aplicada.

Tutor: Dr. Alberto Brunete

González

Departamento: Electricidad,
Electrónica, Automática y Física
aplicada.

Madrid, Febrero 2016



POLITÉCNICA

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y
DISEÑO INDUSTRIAL

Grado en Ingeniería Electrónica y Automática Industrial

TRABAJO FIN DE GRADO

GUIADO DE UN ROBOT MÓVIL
BASADO EN ROS Y KINECT

Firma Autor

Firma Cotutor

Firma Tutor

Copyright ©2016. Daniel Manzaneque Amo

Esta obra está licenciada bajo la licencia Creative Commons

Atribución-NoComercial-SinDerivadas 3.0 Unported (CC BY-NC-ND 3.0). Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by-nc-nd/3.0/deed.es> o envíe una carta a Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, EE.UU. Todas las opiniones aquí expresadas son del autor, y no reflejan necesariamente las opiniones de la Universidad Politécnica de Madrid.

Título: Guiado de un robot móvil basado en ROS y kinect
Autor: Daniel Manzaneque Amo
Tutor: Dr. Alberto Brunete González
Cotutor: Dr. Miguel Hernando Gutiérrez

EL TRIBUNAL

Presidente:

Vocal:

Secretario:

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día de de ... en, en la Escuela Técnica Superior de Ingeniería y Diseño Industrial de la Universidad Politécnica de Madrid, acuerda otorgarle la CALIFICACIÓN de:

VOCAL

SECRETARIO

PRESIDENTE

"Es que jo... Se me escapa la
vida"

— Sofía Díaz, Memorias de un
estudiante en prácticas y con
TFG

Agradecimientos

Agradezco a

Resumen

Realizar la navegación y guiado de un robot móvil surge como herramienta para acceder a lugares donde el ser humano no puede o se encontraría en riesgo, para realizar tareas repetitivas o que conllevarasen algún tipo de desgaste.

Un robot autónomo es por tanto una pieza fundamental en tareas de rescate, salvamento, inspección, exploración de entornos peligrosos o inaccesibles, como la exploración en la superficie de otros planetas. Además, las tareas sociales cada vez están tomando más relevancia en nuestro día a día, como la asistencia a humanos en entornos públicos, la interacción con el entorno o una navegación más segura, como es el caso de los coches autónomos.

Este proyecto de fin de grado trata sobre el guiado y control de un robot móvil de cuatro ruedas, con un sistema motriz en configuración diferencial, equipado con una serie de sensores que permiten su orientación y posicionado en el entorno así como un sensor capaz de captar este en tres dimensiones y un sensor adicional que lo haría tan solo en dos dimensiones.

Los datos de los sensores sirven tanto para construir mapas en dos dimensiones del entorno del robot como para navegar por él evitando obstáculos de manera dinámica. El robot es capaz de generar mapas de celdas en los que situar tanto los objetos estáticos como los móviles, calcular una trayectoria adecuada y dirigirse hasta un punto indicado evitando obstáculos interpuestos en su camino.

Todo esta información, procesado de datos, cálculo de trayectorias y ejecución de movimientos se realiza en un ordenador de abordo integrado en el propio robot utilizando el software Robot Operating System (conocido en robótica por sus siglas ROS), que nos ofrece una interfaz común para interconectar nuestro robot con los sensores y con los algoritmos de navegación.

El proceso de navegación se realiza de dos formas conjuntamente. Por un lado el robot realiza un mapa global con obstáculos que permanecen inmóviles y calcula la trayectoria más adecuada, es lo que denominamos navegación global. Por otro lado, el robot genera un mapa dinámico a su alrededor e identifica la información de los sensores como obstáculos, a continuación, el robot calcula continuamente una trayectoria que se ajuste todo lo posible a la trayectoria global pero que evite los obstáculos cercanos, es lo que se denomina navegación reactiva o local.

Finalmente, un algoritmo de cálculo de movimientos realiza el control de los

motores para que el robot realice el movimiento adecuado en base a las trayectorias definidas anteriormente. De esta forma el robot puede avanzar, retroceder, darse la vuelta o realizar tareas de recuperación de trayectoria en caso de encontrarse bloqueado en algún punto.

A parte de la navegación autónoma, también dispone de un sistema de telecontrol del robot mediante otro ordenador externo y de un algoritmo de detección frontal de objetos en 3 dimensiones (nubes de puntos) que puedan servirle como guía. De esta forma, el robot es capaz de navegar siguiendo el movimiento de una persona o de un robot que le preceda.

El robot Pioneer 3 AT es el robot móvil que se ha empleado en este proyecto (Figura 1) y sobre el que se ha trabajado de manera específica para realizar las pruebas reales de este proyecto. A este robot se le incorporan un sensor láser de dos dimensiones (sensor Sick LMS100) y un sensor de tres dimensiones (sensor Kinect). El cómputo de la navegación se realiza en un ordenador compacto incorporado en el robot (Intel NUC NUC5i7RYH).

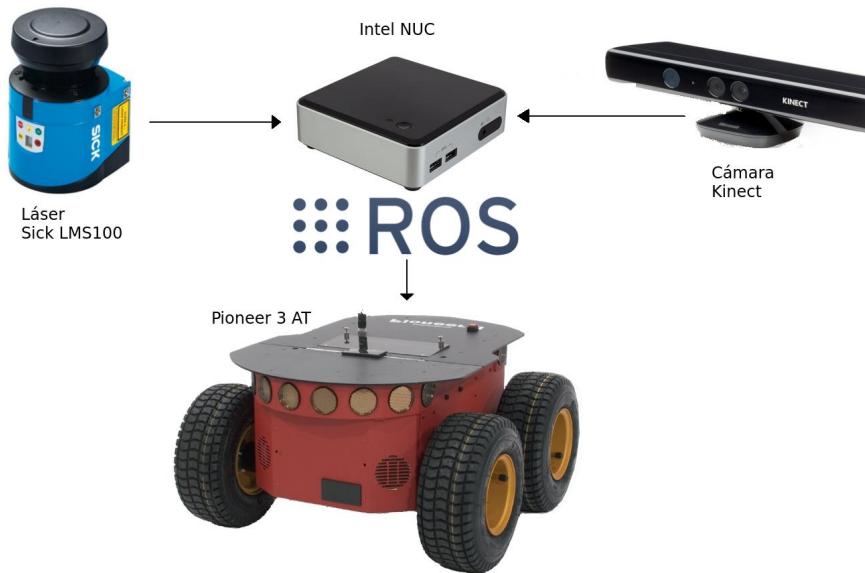


Figura 1: Esquema del sistema robótico utilizado en el proyecto

Las consignas de navegación se realizan mediante un ordenador externo cualquiera conectado a una red inalámbrica o mediante consignas de voz, en las que se indica al robot las tareas de navegación a realizar (avanzar, girar, seguir a una persona...) o un punto del entorno al que dirigirse.

Todas estas implementaciones están desarrolladas bajo el entorno ROS, lo cual permite añadir funcionalidades de manera más rápida y menos laboriosa, como es el caso del control mediante comandos de voz o la interacción mediante sonidos. Es el caso también del simulador de robótica Gazebo, que se integra como funcionalidad en ROS y que ha servido para testar el sistema y aportar las pruebas teóricas pertinentes para luego aplicarlas en el robot real.

Para concluir, podemos decir que este proyecto se encarga de integrar ROS como sistema en un ordenador de abordo incorporado en el robot que permita conectarse con los sensores y realizar la construcción de mapas y navegación autónoma mediante el cálculo de mapas y trayectorias globales y locales, realizar los movimientos del robot, así como reconocer consignas de voz o de teleoperación.

Palabras clave: robot móvil, ROS, navegación reactiva, cálculo de trayectorias.

Abstract

Achieving navigation and guidance of mobile robot comes up as a tool for rescue purposes in places where humans can't access or that involve a high risk for life. Many of those repetitive and fatigating tasks could be done with a robust and capable mobile robot.

An autonomous robot is, by the way, an essential part in rescue, inspection and exploration tasks developed in dangerous or non-reachable places, such as the surface of other planets. Moreover, social tasks are taking more and more interest in our nowadays, such as assistance for humans in public places, interaction with the environment or a safer navigation in the cities. Autonomous car navigation is a good example of this.

This final degree project is about guidance and control of a four-wheel mobile robot with a skid-steer configuration. It is equipped with a sort of sensors, allowing it to make positioning and orientation in the environment. There is also a main sensor capturing the environment in three dimensions and an additional one doing it in two dimensions.

Sensor data is used to build two dimensional maps of the exploration place as well as to take care of dynamical obstacles. The robot can build maps formed by cells where to incorporate or raytrace static and dynamic obstacles, calculate the proper trajectory plan and head for a destination point avoiding obstacles in its way.

All this information, data processing, trajectory calculation and movement execution is done in an onboard computer inside the robot. It uses the Robot Operating System software (known as ROS), which offers a common interface to communicate the robot with sensors and navigation algorithms.

The navigation process is divided in two parts. Firstly, a global obstacle map is done and static objects are added, then the most suitable trajectory is planned. This is called global navigation. Secondly, a dynamic map is done and sensor data incorporates obstacles near the robot. Immediately, a possible trajectory is planned following the original trajectory of the global navigation but avoiding the obstacles. This is called reactive or local navigation.

Finally, a movement algorithm does the control over the robot. It calculates movements to make the robot go forward, backward, turn around or make recovery tasks to recover if the robot has lost the trajectory path or it is stucked at any point.

Apart from autonomous navigation, the robot also has a telecontrol system from an outside computer and an algorithm to detect frontal objects in three dimensions (pointclouds) that can guide the robot. This is how it can navigate following a person when it is walking or another robot in front of it.

Pioneer 3 AT robot is the one used in this project (Figure 2). It is the specific platform and all real tests have been made with it. This robots is equipped with a two dimension laser scanner (Sick LMS100 sensor) and a three dimensional sensor (Kinect sensor). The navigation computation is done in an onboard compact computer (Intel NUC NUC5i7RYH computer).

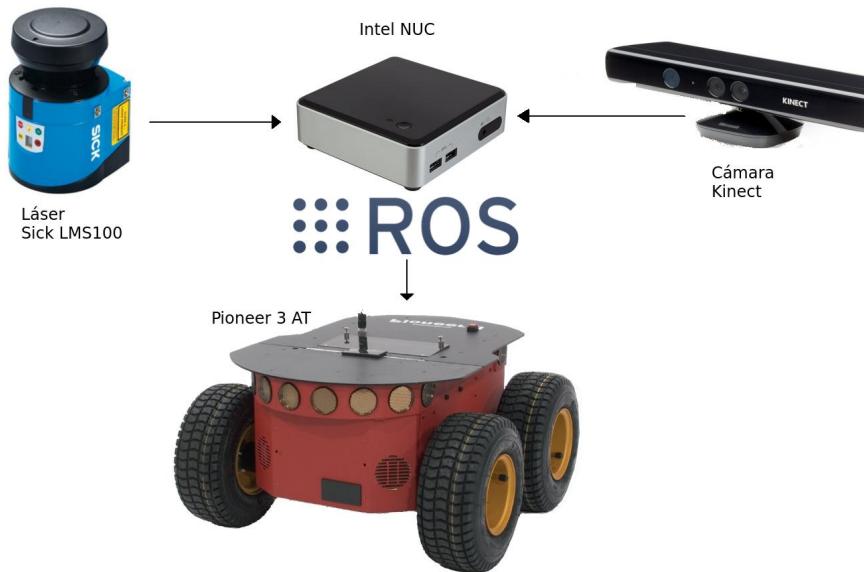


Figure 2: Diagram of the robotic system used for this project

The navigation commands are sent from an outside computer connected to the same wireless network or from voice navigatin commands speaking directly to the robot (go forward, backward, turn right...) or a point in the map to move forward.

All those implementations are developed under ROS framework. This is why additional features can be added in a faster and effortless way. Tha is the case of the robot simulator Gazebo, which integrates as an add-on in ROS. Gazebo has been used to perform tests in navigation and to check theorical concepts to lately incorporate them in the real robotic system.

To conclude, it can be said that this project integrates ROS as a robotic system in an onboard computer and connects to sensors to perform tasks such as building maps or navigation from one point to other. The system calculates local and global maps and trajectories, makes movements according to them, as well as recognises voice or teleoperation commands.

Keywords: mobile robot, ROS, reactive navigation, trajectory calculation.

Índice general

Agradecimientos	v
Resumen	vii
Abstract	xi
1 Introducción	1
1.1 Robótica	1
1.2 Robótica Móvil	2
1.3 Motivación del proyecto	3
2 Estado del arte	5
2.1 Hardware en robótica móvil	5
2.2 Sensores en robótica móvil	8
2.2.1 Sensores internos	9
2.2.2 Sensores externos	10
2.3 Control en la robótica móvil actual	12
2.3.1 Localización y orientación en un entorno	12
2.4 Aplicaciones actuales de la robótica móvil	12
3 Alcance y objetivos del proyecto	19
3.1 Propósito y alcance	19
3.2 Objetivos	20

4 Desarrollo del proyecto	21
4.1 Planteamiento	21
4.2 Planificación del proyecto	22
4.3 Tecnologías y herramientas empleadas en el proyecto	27
4.3.1 Robot Operating System	27
4.3.2 Lenguaje de programación C++	28
4.3.3 Lenguaje de programación Python	28
4.3.4 Controlador de versiones git y repositorios GitHub	28
4.3.5 Simulador de robótica Gazebo	29
4.3.6 RViz: Herramienta de visualización robótica	30
4.3.7 Impresión 3D	31
4.4 Hardware	31
4.4.1 Pioneer 3 AT	31
4.4.2 Sensor Kinect	33
4.4.3 Láser SICK LMS100	34
4.4.4 Intel NUC NUC5i7RYH	36
5 Arquitectura	39
5.1 Arquitectura general	39
5.2 Arquitectura del proyecto	40
6 Entorno ROS	43
6.0.1 Funcionamiento de ROS	43
6.1 Configuración de ROS	43
6.2 Configuración de los paquetes ROS	44
7 Control primario	45
7.1 Nodos hardware	45

ÍNDICE GENERAL	xv
7.1.1 Control del robot: Rosaria y p2os	45
7.1.2 Sensor Kinect	46
7.1.3 Sensor Láser Sick LMS100	47
7.2 Nodo de teleoperación	48
7.3 Nodo de navegación estimada	49
7.4 Nodo de guiado (follower)	49
8 Navegación	51
8.1 Navigation Stack	51
8.1.1 Requisitos para la navegación	51
8.1.2 SLAM mediante slam_gmapping	51
8.2 Costmaps	51
8.3 Planificador de trayectoria global	51
8.4 Planificador de trayectoria local	51
8.5 Navegacion global y local	51
9 Implementación del sistema	53
9.1 Nodo de navegacion por puntos	53
9.2 Nodo de comandos por voz	53
9.2.1 Reconocimiento de comandos de voz	53
9.2.2 Feedback mediante text-to-speech	53
9.3 Primera configuración	53
9.4 Nodo de ejecución automática de nodos	53
10 Pruebas del sistema	55
10.1 Simulación con MobileSim	55
10.2 Simulación con Gazebo	55
10.2.1 Segunda configuracion	55

10.2.2 Datos de navegacion	55
10.3 Visualización mediante RViz	55
10.4 Pruebas reales	55
10.4.1 SLAM	55
10.4.2 Test de resistencia	55
10.4.3 Aspectos de la navegación	55
11 Conclusion	57
11.1 Conclusión sobre la metodología	57
11.2 Conclusión sobre los resultados	57
11.3 Desarrollos futuros	57
12 Anexo I: Configuración del sistema	63
12.1 Configuración del espacio de trabajo	63
12.1.1 Instalación de las librerías	63
12.1.2 Gestión de las dependencias	63
12.2 Configuración del hardware	63
12.2.1 Calibración de los encoders	63
12.2.2 Ordenador de abordo	63
12.2.3 Sensor Kinect	63
12.2.4 Láser SICK LMS100	63

Índice de figuras

1	Esquema del sistema robótico utilizado en el proyecto	viii
2	Diagram of the robotic system used for this project	xii
2.1	Motor de corriente continua con encoder	6
2.2	Configuraciones hardware. Basado en [SOGSBS ⁺ 10]	6
2.3	Robot de configuración diferencial Pioneer 3 DX	7
2.4	Ejemplos de configuración skid-steer: Cargador frontal, Robotnik Guardian, Pioneer 3 AT	8
2.5	Configuración síncrona	8
2.6	Configuración síncrona y rueda omnidireccional	9
2.7	Robot Uranus con ruedas tipo Mecanum	9
2.8	Esquema de un encoder absoluto	9
2.9	Unidad de medida inercial, IMU	10
2.10	Cámara estereoscópica del robot PR2	11
2.11	Sensor de 3 dimensiones Kinect para Xbox 360	12
2.12	Imagen tomada a sí mismo por el robot Curiosity en la superficie marciana	13
2.13	Robots de desactivación de explosivos: iRobot 510 Packbot y TALON	14
2.14	Robot de exploración de desastres Quince	14
2.15	Robot para limpieza del hogar Roomba, de iRobot	15
2.16	Robot pulverizador de aplicación agrícola, AgriRobot	15

2.17 Robot de inspección de viñedos, VinBot	15
2.18 Robot social Maggie y robot de asistencia ROSA	16
2.19 Robot PR2 desarrollado por Willow Garage	16
2.20 Sensores del coche autónomo de Google	17
4.1 Logo de ROS	27
4.2 Entorno de simulación Gazebo	30
4.3 Entorno gráfico RViz	30
4.4 Robot Pioneer 3-AT	31
4.5 Panel de control del robot Pioneer 3-AT	32
4.6 Sensor Kinect	33
4.7 Proyección de infrarrojos y obtención de la nube de puntos	34
4.8 Sensor escaner láser Sick LMS100	35
4.9 Campo de visión del sensor láser Sick LMS100	35
5.1 Estructura del proyecto	40
5.2 Estructura de carpetas del paquete pioneer_utils	41

Índice de tablas

4.1	Especificaciones del robot Pioneer 3 AT	32
4.2	Características del sensor Kinect	34
4.3	Características del sensor láser Sick LMS100. Basado en [SIC09] . . .	36
4.4	Características del ordenador Intel NUC NUC5i7RYH	37

Índice de códigos

7.1	Launchfile para RosAria.	46
7.2	Launchfile para Kinect en el paquete freenect_launch.	47
7.3	Launchfile para el sensor Láser Sick LMS100.	47
7.4	Líneas del archivo <i>.bashrc</i> en el ordenador de abordo.	48
7.5	Ejemplo <i>.bashrc</i> en un ordenador externo para realizar comunicación con el máster.	48

Capítulo 1

Introducción

Esta primera sección será un apartado previo para poner en contexto al lector sobre la robótica móvil en general y a todo el desarrollo del proyecto ”Guiado de un robot móvil basado en ROS y Kinect” en particular.

En esta sección explicaremos qué consiste y cuáles son los principales problemas de la robótica móvil así como las motivaciones para el desarrollo del proyecto.

1.1 Robótica

a historia de la robótica tiene su precursor en la mecánica y los mecanismos desarrollados para imitar el movimiento y funciones de los seres vivos. Los antiguos egipcios ya empleaban mecanismos para mover los brazos de las estatuas y los griegos utilizaban sistemas hidráulicos para adornar sus templos.

Podemos decir que en torno al siglo XVII es cuando se inicia la historia de la robótica actual. En concreto, cuando Jacques de Vaucanson en 1745 inventó el primer telar automático de la historia. También construyó lo que se denominó como autómatas en la época, con obras como El flautista, El tamborilero o El pato, obra conocida como una de las pioneras en la historia de la robótica [Roj07].

Más tarde, Joseph Jacquard inventa en 1801 una máquina textil programable mediante tarjetas perforadas refinando la tecnología empleada por Vaucanson [MGB00]. Años más tarde, la Revolución Industrial impulsó el desarrollo de máquinas que automatizaban tareas que antes realizaban las personas dando paso a la historieta de la automatización industrial.

Antes ya se habían desarrollado algunos mecanismos automáticos como el León mecánico creado por Leonardo Da Vinci en el siglo XVI [DL12], que abría su pecho mostrando el escudo del rey Luis XII, o el Hombre de palo, un autómata de madera construido por Juanelo Turriano que andaba y movía la cabeza, ojos, boca y brazos.

La palabra “robot” comenzó a utilizarse en el mundo del teatro y de la ciencia

ficción. Una obra checoslovaca publicada en 1917 por Karel Čapek, denominada Rossum's Universal Robots, dio lugar al término robot. La palabra checa "Robota" significa servidumbre o trabajador forzado, y cuando se tradujo al inglés se convirtió en el término robot. Como ejemplo de ciencia ficción se podría mencionar al escritor Isaac Asimov por la publicación sobre las tres leyes de la robótica [AdN87].

Los primeros robots llamados como tal fueron creados en 1958 y eran de tipo teleoperados, con el objetivo de manipular elementos sin riesgo para el operario. Este tipo de robots consisten en un sistema maestro-esclavo en el que los movimientos realizados por el maestro son transmitidos mecánicamente a cierta distancia y reproducidos por el robot esclavo.

Años más tarde, se comenzó a utilizar la tecnología electrónica y el uso del servocontrol, sustituyendo la transmisión mecánica por otra eléctrica. Ejemplos de estos manipuladores fue el realizado por Ralph Mosher, Handy-Man, consistente en dos brazos mecánicos teleoperados mediante un maestro del tipo denominado exoesqueleto.

La sustitución del operador en este tipo de sistemas por un programa de ordenador que controlase los movimientos del manipulador dio paso al concepto de robot [BPBA07].

1.2 Robótica Móvil

Prácticamente cualquier robot consta de alguna parte móvil que le permite realizar algún tipo de tarea, sin embargo nos referimos a la «robótica móvil» como el área de la robótica que estudia los robots con capacidad para trasladarse en un ambiente dado.

Los robots móviles son aquellos que tienen la capacidad de desplazarse utilizando algún sistema locomoción como pueden ser ruedas, patas, girar sobre sí mismos... Estos robots se diferencian respecto a los robots fijos que permanecen anclados a una superficie, como un brazo robótico industrial. Tampoco debe confundirse con los robots destinados a desplazarse por otros medios como agua o aire, ya que estaríamos entrando en el área de la robótica acuática/submarina o robótica aérea respectivamente. Podemos decir por tanto que la robótica móvil se refiere a robots que se mueven en el entorno terrestre.

Las aplicaciones dentro de la robótica móvil pueden ser múltiples: exploración de entornos peligrosos, exploración espacial o minera, misiones e búsqueda y rescate de personas, telepresencia, automatización de procesos, transporte autónomo, vigilancia, inspección y reconocimiento del terreno o utilizados como plataformas móviles que incorporan otros sistemas robóticos como podrían ser un brazo manipulador.

La robótica móvil surgió como manera extender el campo de la robótica hacia robots que anclados a un punto fijo. La capacidad de estos robots para desenvolverse en entornos diferentes ofrecía la posibilidad de abrir nuevas líneas de investigación

y automatizar tareas que estaban asociadas con la navegación y localización.

Los primeros pasos dentro de la robótica móvil eran motivados por la idea de introducir la mayor autonomía posible a los robots, tanto en términos de suministro de energía como en computación para realizar las tareas de planificación, percepción y control.

Ampliar **REFERENCIAS A LIBROS**

1.3 Motivación del proyecto

Dotar a un robot de la capacidad de navegar autónomamente puede ser una alternativa imprescindible en el caso de que se necesite explorar un entorno que no sea fácilmente accesible para el ser humano o que conlleve cierto riesgo.

Cualquier proyecto que desarrolle la automatización de un proceso es ya una motivación, puesto que se va a diseñar una máquina que sea capaz de realizar una tarea que antes solo podía realizarse por un ser humano. Además, dichas tareas realizadas por un robot pueden realizarse, en principio, con una mayor precisión y con mayor repetibilidad debido a que se elimina el factor del cansancio.

Este proyecto también viene motivado por la integración de ROS dentro de una plataforma móvil. Con esta plataforma de desarrollo software podemos explorar un concepto diferente de programación en robótica, que ofrece características como:

- Abstraerse de la programación a bajo nivel.
- Reutilizar software ya desarrollado (nodos).
- Interfaz de comunicación común.
- Escalabilidad del sistema.
- Simulación mediante Gazebo.
- Visualización gráfica de la información aportada por sensores.
- Transformación entre los diferentes sistemas de coordenadas.

Utilizar sensor de bajo coste Kinect es otra de las motivaciones de este proyecto debido a que este sensor de bajo coste permite obtener información en tres dimensiones del entorno, pudiéndose realizar una navegación basada solamente en este sensor además de reconocer objetos por su forma y realizar el guiado del robot detectando objetos.

De las aplicaciones que más han servido como motivación para el desarrollo de este proyecto ha sido la automatización de las tareas de conducción de automóviles [Xat11], un sector que se encuentra en auge y que comienza a dar sus primeros pasos en el mundo real [Nev12].

También los robots de exploración espacial de la NASA, en especial a su último rover en Marte, Curiosity [NAS12], que permite explorar el entorno árido de la superficie marciana con un alto grado de autonomía en las labores de inspección y análisis de elementos.

Capítulo 2

Estado del arte

La robótica móvil vive actualmente un momento de gran desarrollo para multitud de aplicaciones en entornos diversos, desde espacios abiertos con orografía accidentada y condiciones climáticas adversas [Gui13] [PBR06], entornos controlados y espacios interiores conocidos como la automatización de tareas de almacenaje de productos [Reu12], hasta orientación y exploración de espacios interiores desconocidos con robots usados para la creacion de mapas de edificios.

De la misma forma, el interés en robots que sea capaces de reproducir las capacidades de un ser humano e incluso que pueda dar asistencia ya sea en entornos conocidos o no abre un área de posibilidades en las que los robots móviles cobran importancia.

Los avances tanto el las características hardware como software son notables aunque estas suelen variar dependiendo de la aplicación a la que un robot esté destinado. En este capítulo trata de hacer un resumen del estado actual de la robótica móvil.

2.1 Hardware en robótica móvil

Como hemos indicado previamente, la configuración hardware de un robot móvil varía dependiendo de la aplicación a la que vaya destinado. Es cierto que lo ideal para un robot sería disponer de una configuración hardware común que fuera polivalente en los diferentes terrenos y situaciones, sin embargo, debido a la variedad de aplicaciones y dado que un robot suele destinarse a tareas específicas, la elección del hardware que mejor se adapta es una tendencia común en robótica.

Para seleccionar el hardware debemos valorar el tipo de actuador que se requiere, entendiéndose por actuador al dispositivo que genera el movimiento de los elementos que hacen que el robot móvil se desplace. En robótica móvil suelen utilizarse los actuadores de tipo eléctrico, ya que ofrecen unas prestaciones de potencia, controlabilidad y coste adecuados. Además, ofrecen la posibilidad de que la alimentación

tación esté integrada en el robot, haciéndolo independiente de una fuente de energía accesoria.

Los actuadores eléctricos son, por tanto, los más utilizados. En concreto, los motores de corriente continua (Figura 2.1) ofrecen un fácil control y acoplamiento a un encoder. Los encoder son sensores de posición que permiten conocer el giro de un eje de rotación. Estos sensores son muy importantes en robótica móvil, ya que a partir de la información que arrojan el robot tiene conciencia de su posición relativa, en el caso de los encoders incrementales, o su posición absoluta, en el caso de los encoders absolutos (Más detalladamente en el apartado 2.2).



Figura 2.1: Motor de corriente continua con encoder

Existen otros tipos de actuadores eléctricos que se utilizan en robótica, como puede ser el caso de los motores paso a paso, sin embargo su baja velocidad de giro no los hacen adecuados para robots móviles.

La disposición de los actuadores determina la configuración del robot. Centrándonos en robots que se desplazan mediante ruedas y descartando a los robots con patas, podemos distinguir las siguientes configuraciones: Ackerman, triciclo clásico, tracción diferencial, skid-steer, síncrona y omnidireccional.

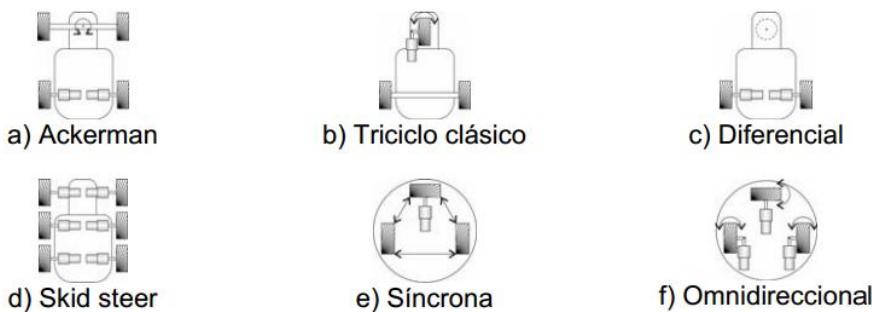


Figura 2.2: Configuraciones hardware. Basado en [SOGSBS⁺10]

- a. Configuración Ackerman

Consta de cuatro ruedas. Las ruedas motrices son las traseras o delanteras, y éstas últimas se encargan además de la dirección. Permite un desplazamiento a altas velocidades y la posibilidad de realizar giros con estabilidad. Esta configuración es la que se utiliza en la industria del automóvil.

b. Triciclo clásico

Consta de tres ruedas. Las ruedas motrices pueden ser las dos ruedas traseras o solo la delantera, que se encarga de la dirección. Este es el caso de los triciclos y de algunas bicicletas. Esta configuración ofrece alto grado de maniobrabilidad penalizando la estabilidad del conjunto y realizar giros den 90°.

c. Configuración diferencial

Consta de dos ruedas colocadas en el eje perpendicular a la dirección de desplazamiento del robot. Cada rueda es controlada por un motor, de tal forma que la diferencia de velocidad giro de una rueda respecto a otra determina el giro, avance o retroceso del robot. Los robots que presentan esta configuración suelen utilizar una tercera rueda que gira libremente que sirve como apoyo (rueda loca). Es la configuración típica de las sillas de ruedas y su característica principal es que permite realizar giros completos sobre sí mismo.



Figura 2.3: Robot de configuración diferencial Pioneer 3 DX

d. Skid steer

Consta de cuatro ruedas, todas ellas motrices, y su principio de funcionamiento es el mismo que el utilizado en la configuración diferencial. Esta configuración presenta las ventajas de la configuración diferencial, pudiendo realizar giros sobre el eje del robot, pero presenta la desventaja de que las ruedas deben deslizarse lateralmente, por tanto existe un rozamiento que varía en función de la inclinación el tipo de terreno que dificulta realizar un modelo cinemático.

Proporciona mucha tracción y estabilidad y suele encontrarse en aplicaciones relacionadas con la exploración, vehículos obra o vehículos todo terreno (Figura 2.4).

Este sistema es el que se utiliza también en los tanques de guerra, aunque en vez de neumáticos se utilizan orugas, denominado configuración por deslizamiento de cintas [dS07].



Figura 2.4: Ejemplos de configuración skid-steer: Cargador frontal, Robotnik Guardian, Pioneer 3 AT

e. Configuración síncrona

Conformado por tres o más ruedas acopladas mecánicamente y dotadas de tracción, este sistema permite que todas las ruedas rotén en la misma dirección y giren a la misma velocidad (Figura 2.5). Es utilizada ampliamente en robótica para robots móviles de interior, aunque está siendo desplazada por la configuración omnidireccional.

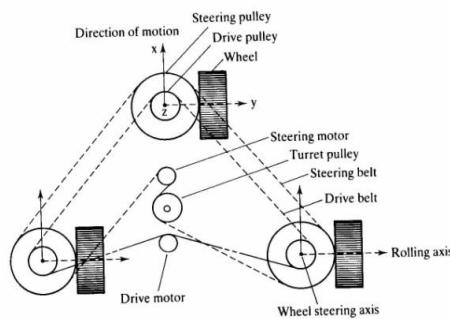


Figura 2.5: Configuración síncrona

f. Configuración omnidireccional

Consta de 3 ruedas cada una con un motor independiente, que permiten el desplazamiento en cualquier dirección (Figura 2.6). Las ruedas omnidireccionales constan de una serie de rodillos con el eje de rotación perpendicular a la dirección de avance.

Esta configuración diferencial empieza a utilizarse en sistemas de 4 ruedas con las denominadas "Mecanum Wheels" [DL91], que son ruedas similares a las omnidireccionales pero con los rodillos colocados en cierto ángulo (Figura 2.7). La combinación de los giros de cada una permiten al robot moverse en cualquier dirección.

2.2 Sensores en robótica móvil

Para que un robot pueda realizar tareas con una determinada precisión y velocidad debe conocer el entorno del sistema en el que se quiera actuar así como el estado del robot en ese sistema.

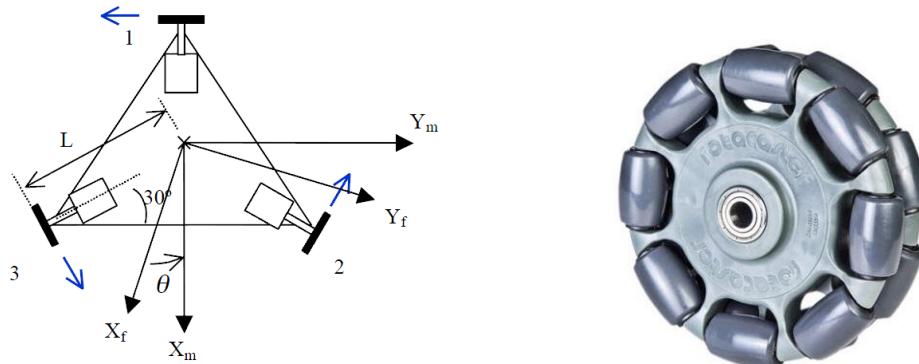


Figura 2.6: Configuración síncrona y rueda omnidireccional



Figura 2.7: Robot Uranus con ruedas tipo Mecanum

Existen dos tipos de sensores, los sensores internos, que aportan información sobre la posición orientación del robot, y los externos, que aportan información del entorno alrededor del robot.

2.2.1 Sensores internos

Dentro de los sensores internos, los sensores de posición primordiales son los encoders, tanto los de tipo incremental como los de tipo absoluto (Figura 2.8). Su funcionamiento se basa en un foto-emisor y un foto-receptor que detectan el paso o no de luz a través de un disco con ciertas marcas acoplado al eje de giro del actuador.

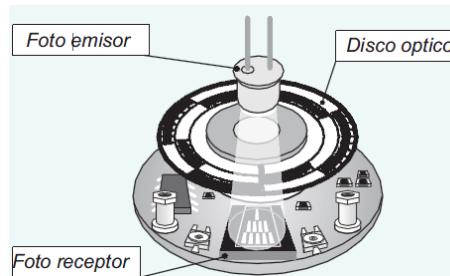


Figura 2.8: Esquema de un encoder absoluto

Los sensores de velocidad son similares a los encoders pero miden la velocidad de giro del eje del actuador. La tacogeneratriz proporciona una tensión proporcional a la velocidad de giro.

Los sensores acelerómetros o inclinómetros, permiten conocer la inclinación del robot en cada uno de sus ejes, así como las aceleraciones producidas por su propio desplazamiento.

Existen otros sensores más sofisticados como las Unidades de medida inercial (IMU) (Figura 2.9). Son dispositivos que combinan las medidas de un giróscopo y varios acelerómetros para determinar la posición relativa (x , y , z) y la orientación (roll, pitch, yaw), velocidad y aceleración respecto a un sistema de referencia.



Figura 2.9: Unidad de medida inercial, IMU

Debido a que la aceleración se ha de integrar dos veces para obtener la posición, el error crece de forma cuadrática. Luego para largos períodos de operación las unidades IMU se deben de resetear con otros sensores tipo GPS.

2.2.2 Sensores externos

Los sensores externos son aquellos que nos aportan información sobre el estado del robot respecto al entorno o que nos da información sobre lo que ocurre alrededor de este.

Los sensores de presencia, como son los sensores de tipo inductivo, capacitivo, óptico o mecánico. Sea cual sea la naturaleza del sensor, su función es la de detectar presencia. Un ejemplo de aplicación a un robot móvil sería una serie de sensores de presencia mecánicos, denominados "fin de carrera", colocados en la parte delantera, de modo que al tocar algún obstáculo se tuviera conciencia de la presencia de un obstáculo.

Sensores de posicionamiento global GPS (Global Positioning System) que permiten determinar la posición de un objeto en todo el mundo, normalmente con una precisión de metros. El GPS funciona con una red de satélites con trayectorias sincronizadas que cubren toda la superficie de la tierra. El GPS lanza señales a los satélites y calculando el tiempo que tarda éstos en responder, se obtiene la posición por triangulación.

Los sensores GPS se utilizan en robots móviles que operan en el exterior y suelen combinarse con otros sensores que ofrezcan una mayor precisión.

Los sensores de distancia son aquellos que nos dan una referencia de la longitud que existe a los objetos cercanos. Es el caso de los sensores e ultrasonidos, donde un emisor emite una onda ultrasónica y cuando es reflejada por un objeto se puede determinar la distancia a la que se encuentra midiendo el tiempo que tarda el sonido en ir y volver. Los sensores de distancia también pueden ser infrarrojos, funcionando de la misma manera.

Existen sensores de distancia que utilizan tecnología láser para determinar la longitud de un punto a otro, se denominan Scanners láser. Estos sensores emiten rayos láser en un plano de 2 dimensiones y en un rango determinado, y midiendo el tiempo de vuelo del haz láser son capaces de obtener una medida muy precisa de la distancia.

Existen otro tipo de sensores de distancia que permiten obtener distancias a puntos de manera tridimensional. Algunos utilizan un sistema de doble cámara conocidos como cámara estereoscópica (Figura 2.10). Estos sensores son capaces de obtener imágenes 3D con la información de dos imágenes tomadas a cierta distancia una de otra. Es el sensor más parecido a la visión humana.



Figura 2.10: Cámara estereoscópica del robot PR2

Otros sensores de distancia en 3 dimensiones, son los sensores de tipo infrarrojo, como es el sensor Kinect, que ha sido muy popular debido a su bajo coste y su buena respuesta.

Kinect es un dispositivo desarrollado por PrimeSense y distribuido por Microsoft para la videoconsola Xbox 360 (Figura 4.6).

Inicialmente permitía controlar e interactuar con la consola XBOX sin necesidad de tener contacto físico con un controlador. Este sensor permite reconocer gestos, comandos de voz, objetos e imágenes; esto hace que tenga mucho interés en el mundo de la robótica.

Para captar el entorno en 3 dimensiones, Kinect incluye una cámara de vídeo RGB, un emisor de haz infrarrojo y una cámara infrarroja.



Figura 2.11: Sensor de 3 dimensiones Kinect para Xbox 360

2.3 Control en la robótica móvil actual

El control del movimiento en los robots móviles con ruedas puede describirse, de manera general, en cuatro tareas fundamentales: localización y orientación, planificación de trayectoria, seguimiento de la misma y evasión de los obstáculos.

2.3.1 Localización y orientación en un entorno

Normalmente, uno de los mayores problemas que conciernen a la navegación de robots móviles consiste en la determinación de su localización respecto a un mapa en función de la información captada por los sensores.. No basta con situar una referencia global, si no que es imprescindible conocer la posición relativa respecto a los posibles obstáculos, tanto móviles como estáticos, de su entorno. Para esta tarea existen diferentes opciones, como utilizar mapas introducidos en el robot, o bien elaborar un mapa de manera simultánea al movimiento del robot por un lugar, como si se tratase de un robot de exploración. Es lo que se conoce como SLAM (Simultaneous Localization and Mapping).

Uno de las motivaciones para el uso de esta técnica es la construcción de mapas desde el punto de vista del robot, así como el ruido que se genera en los sensores de posición internos del robot que miden la odometría. Sin embargo, esta técnica en ocasiones puede producir efectos no deseados, como incorrecciones en el mapa debido a su alto coste computacional o variaciones debidas a objetos que se mueven en torno al robot.

Pueden distinguirse tres tipos de mapas: geométrico, topológico y semántico. El nivel geométrico es el más utilizado y consiste en un mapa métrico donde se representan los segmentos básicos de un entorno, o un mapa discretizado, donde se efectúa la descomposición de los elementos en celdillas. En el nivel topológico, se representan nodos y conexiones entre ellos, y el nivel semántico es cuando se elimina la información geométrica.

2.4 Aplicaciones actuales de la robótica móvil

Algunas de las aplicaciones del área de la robótica móvil ya han sido mencionadas con anterioridad en este documento, estas van enfocadas a sustituir la labor que

realiza el ser humano en situaciones de riesgo o en tareas repetitivas que aportan poco valor.

Una de las aplicaciones más famosas sobre robótica móvil son los Rovers de exploración espacial de la NASA "Spirit" y "Opportunity" dentro de la misión "Mars Exploration Rover" lanzada en 2003. Estos robots disponen de sistemas de navegación y exploración ideados para sus misiones en la superficie del planeta Marte, con el objetivo de analizar el entorno y los materiales de sus rocas y cráteres y enviar la información de vuelta a la Tierra [NAS03].

Un tercer robot no tripulado fue enviado con posterioridad a la superficie del planeta rojo. El robot "Curiosity" forma parte de la segunda generación de robots de exploración espacial y aterrizó en Marte en el año 2012. La misión "Mars Science Laboratory" ha permitido descubrir la existencia de antiguos lagos en la superficie del planeta [NAS12].



Figura 2.12: Imagen tomada a sí mismo por el robot Curiosity en la superficie marciana

Otras aplicaciones conocidas de la robótica móvil son los robots de búsqueda, reconocimiento y desactivación de explosivos. Robots como el iRobot 510 PackBot (Figura 2.13), desarrollado por iRobot Corporation, o TALON (Figura 2.13), desarrollado por QinetiQ North America, son ejemplos de cómo la robótica móvil es una herramienta muy valiosa en situaciones peligrosas o en sectores como seguridad y defensa.

También existen robots ideados para realizar tareas en entornos peligrosos o en situaciones de desastre. Como ejemplo podemos citar la catástrofe que sufrió Japón el 11 de Marzo de 2011, en la que un terremoto causó daños catastróficos en la central nuclear de Fukushima [Paí11]. Los niveles de radiación fueron tan altos que solo los robots eran capaces de entrar a valorar la situación de la central.

Estos robots, preparados para aguantar la radiación y realizar tareas de exploración y limpieza fueron Quince (Figura 2.14), desarrollado por Chiba Institute of Technology [NKO⁺11], un robot móvil capaz de subir y bajar escaleras y operar en las plantas superiores de la central nuclear.

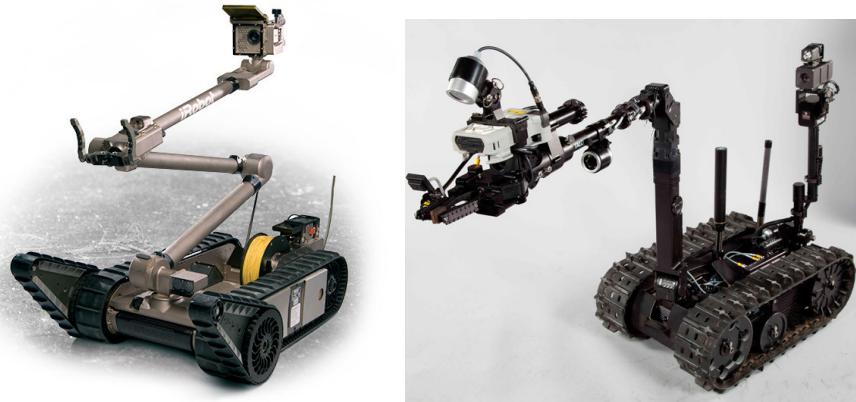


Figura 2.13: Robots de desactivación de explosivos: iRobot 510 Packbot y TALON

Y Raccoon (Figura 2.14), desarrollado por Tepco, equipado con dos cabezales móviles preparados para aspirar y limpiar, encargado de recuperar el polvo contaminado del edificio del reactor 2.

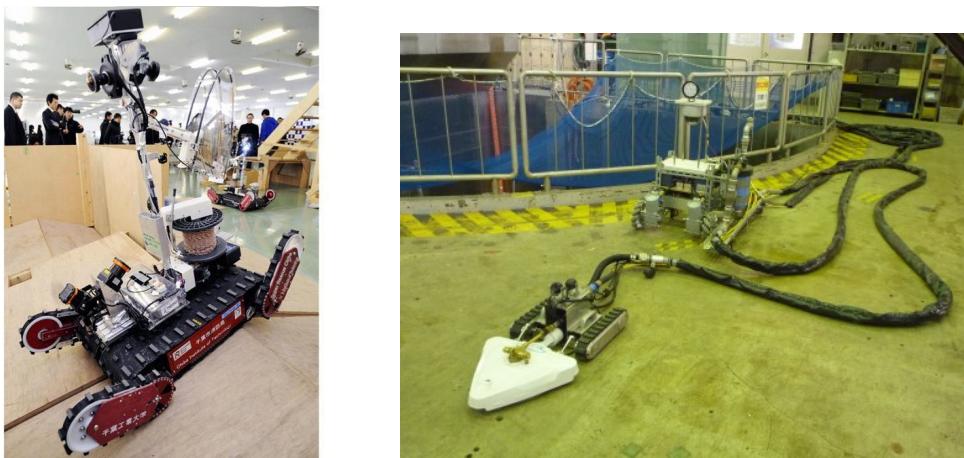


Figura 2.14: Robot de exploración de desastres Quince

Siguiendo con la aplicación de robots móviles en tareas de limpieza, podemos destacar la popularización de los robots domésticos de tipo aspiradora, que se encargan de las tareas repetitivas del entorno del hogar, como el iRobot Roomba (Figura 2.15).

También existen aplicaciones de robots móviles en el ámbito de la agricultura. La empresa Robotnik [Rob02] tiene en marcha diferentes programas para incorporar sus robots en tareas de recolección o inspección de la cosecha.

Su proyecto AgriRobot (Figura 2.16), investiga el aspecto de la interacción humano-robot (Human-Robot Interface, HRI, en inglés). Para ello se sirven del robot Summit X Lincorporado con 4 ruedas motoras de alta potencia. El robot contiene un pulverizador eléctrico de 10 litros, tiene un sistema de visión, navegación y localización, y utiliza el software ROS.

Al igual que el proyecto VinBot (Figura 2.17), de la misma empresa. Un robot



Figura 2.15: Robot para limpieza del hogar Roomba, de iRobot



Figura 2.16: Robot pulverizador de aplicación agrícola, AgriRobot

móvil autónomo todo terreno dotado con un conjunto de sensores capaces de capturar y analizar imágenes de viñedos y datos en 3D mediante el uso de aplicaciones de cloud computing. Su finalidad es determinar el rendimiento de los viñedos y compartir esta información con los viticultores.



Figura 2.17: Robot de inspección de viñedos, VinBot

En el apartado de la robótica social, podemos hablar de robots móviles con ruedas destinados a la asistencia de personas en lugares públicos como centros comerciales, aeropuertos u hospitales, o también como asistentes domésticos en hogares de personas con movilidad reducida. Es el caso del robot Maggie [AMRS11], desarrollado por la Universidad Carlos III de Madrid, o el robot de asistencia social ROSA,

desarrollado por la Universidad Politécnica de Madrid (Figura 2.18).



Figura 2.18: Robot social Maggie y robot de asistencia ROSA

Existen otro tipo de robots con altas capacidades que realizan la función de un robot móvil, como el robot de investigación PR2 (Figura 2.19), dotado con un sistema de desplazamiento omnidireccional.



Figura 2.19: Robot PR2 desarrollado por Willow Garage

Este robot, desarrollado por los investigadores de Willow Garage, fue creado para proporcionar una plataforma común junto con ROS (Robot Operating System), sobre la que realizar investigaciones que ayuden al desarrollo software de aplicaciones robóticas.

Finalmente, podemos destacar una de las aplicaciones que más han llamado la atención en la sociedad, los coches sin conductor. Potenciados por los desarrollos de

la empresa Google, cuyo proyecto consiste en combinar la información obtenida del servicio de mapas de la compañía con la inteligencia artificial, estos coches (Figura 2.20) suponen un cambio notable a nivel de seguridad en el transporte urbano de nuestra sociedad.

Los vehículos van equipados con cámaras de vídeo, un sensor láser de 360° colocado en la parte superior del vehículo, sensores radar, sensores de odometría en las ruedas y de posición con localización GPS [Gui11].

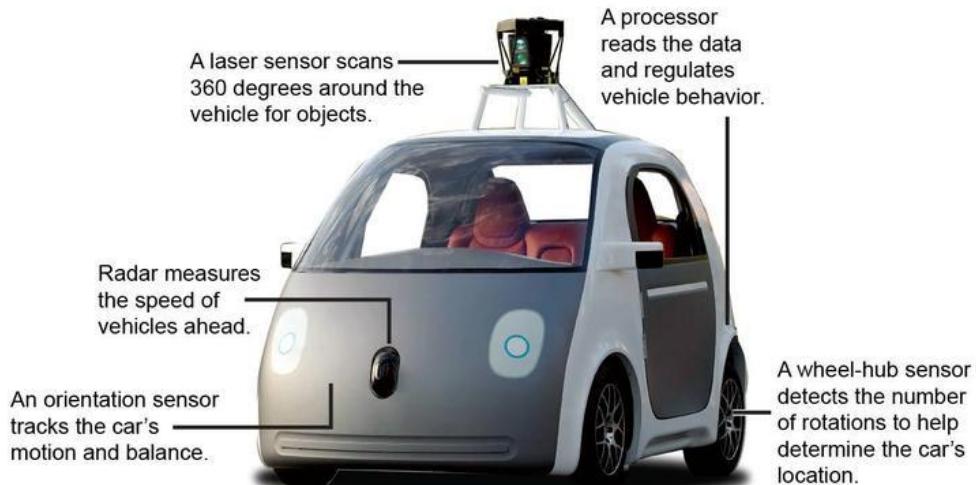


Figura 2.20: Sensores del coche autónomo de Google

Estos coches disponen actualmente de permiso para circular en algunos estados de Estados Unidos tras haber superado 1.800.000 millas desde que el programa (Google Self-Driving Car Project) comenzase en 2009 [Goo09].

En conclusión, podemos decir que los campos de aplicación de la robótica móvil son amplios, variados y con grandes perspectivas de futuro. Sin embargo, el mercado de la robótica civil a penas acaba de comenzar con la aparición de los primeros robots domésticos y los robots de carácter social. En otros campos, vemos que existe un auge de la robótica móvil y que a medida que los sistemas robóticos avanzan surgen nuevas posibilidades donde aplicarla.

Capítulo 3

Alcance y objetivos del proyecto

En este capítulo se define el alcance y los objetivos de este proyecto, es decir, lo que se pretende conseguir con este proyecto y hasta donde puede llegar.

3.1 Propósito y alcance

El propósito de este proyecto es el control automático de un robot móvil utilizando ROS. Lo que se pretende es implementar la navegación autónoma del robot basándose en un control reactivo a partir de la información obtenida a través del sensor Kinect.

El alcance del proyecto requiere múltiples elementos de trabajo:

En primer lugar, requiere un conocimiento previo del sistema hardware, como es el robot Pioneer 3 AT así como el sensor Kinect. Cómo integrar estos elementos y acceder a la información que aportan sus sensores y comandar al robot para que realice movimientos.

En segundo lugar, requiere un conocimiento del entorno de desarrollo ROS. Las herramientas software de las que dispone, el funcionamiento interno y la manera de programar e interaccionar con los diferentes elementos, el aprendizaje y comprensión.

En tercer lugar, incorporar los sensores pertinentes para obtener la información que permita al robot posicionarse en el entorno.

En cuarto lugar, implementar los ajustes necesarios para que el robot pueda operar utilizando el entorno de navegación ofrecido por ROS. Realizar una configuración óptima de los sensores y realizar las pruebas reales para el cálculo de trayectorias y el control reactivo del robot.

Por último, realizar la integración del sistema dentro de la plataforma robótica. Disponer de todo lo necesario para que el robot quede totalmente adaptado al sistema ROS e integrado con el sensor Kinect y los sensores pertinentes.

3.2 Objetivos

El objetivo de este proyecto es realizar el control de un robot móvil para que sea un robot autónomo, basándose en la información que da la odometría y la nube de puntos que proporciona un sensor que captura el entorno en 3 dimensiones como el sensor Kinect. Este control debe realizarse con la ayuda del software ROS, integrándolo como parte del sistema robótico para que sirva de soporte al desarrollo del proyecto.

El robot debe ser capaz de localizarse y situarse en el entorno, el sensor Kinect ofrecerá información sobre los objetos alrededor del robot, y el software desarrollado para ROS deberá ser capaz de hacer un control reactivo sobre los movimientos para permitir al robot moverse por interiores y guiarlo hacia un punto indicado.

Los objetivos principales de este proyecto serán los siguientes:

- a) El primer objetivo es familiarizarse con el control de los robots móviles y más concretamente en el control del robot Pioneer 3 AT. Familiarizarse con el sensor Kinect y con el software necesario para su uso. Y familiarizarse con el framework ROS y sus herramientas para el desarrollo de sistemas robóticos.
- b) Detección de obstáculos simples con el sensor Kinect para su inclusión posterior en el sistema de navegación del robot.
- c) Telecontrol del robot Pioneer 3 AT a partir del software ROS para realizar un controlador manual desde un ordenador externo.
- d) Realizar un sistema de navegación autónomo que sea capaz de dirigir el robot basándose en la información aportada por el sensor Kinect y otro tipo de sensores embebidos.

De los objetivos anteriores podemos desgranar algunos objetivos intermedios:

- a) Comprender el funcionamiento de ROS y la integración e interacción del software desarrollado bajo este entorno.
- b) Control del movimiento del robot Pioneer 3 AT a través de ROS, así como la obtención de la información de la odometría, estado de la batería, encendido de motores...
- c) Puesta en marcha el sistema de navegación para robots de ROS conocido como "Navigation Stack" y exploración de las capacidades del sistema.
- d) Valorar el uso de sensores adicionales y buscar una disposición óptima de los mismos para integrarlos en el sistema de navegación y en la arquitectura hardware del propio robot.

Capítulo 4

Desarrollo del proyecto

En esta capítulo se expone cuál ha sido el planteamiento del proyecto y los pasos que se han seguido para conseguir los objetivos y llegar a unos resultados óptimos.

4.1 Planteamiento

El robot sobre el que se pretende trabajar es el Pioneer 3 AT, de la empresa Adept Mobile Robots, cuyas características se detallarán más adelante. La configuración del sistema motriz es de tipo skid-steer y será determinante a la hora de realizar el control del desplazamiento.

Para realizar la teleoperación del robot, utilizaremos las herramientas de comunicación de ROS, que hacen que la ejecución de los diferentes nodos de forma distribuida entre equipos se realice de forma transparente para el usuario. Con esta característica podremos desarrollar con facilidad un sistema de telecontrol sin preocuparnos en exceso por la implementación de la comunicación entre equipos.

Para la navegación se pretende que el robot base sus movimientos en un sistema reactivo, es decir, que el robot base su navegación principalmente en la información captada por sus sensores y no en un mapa preestablecido. El control en navegación del robot se basará en la funcionalidad "Navigation Stack" de ROS, que también será explicada en detalle más delante.

El desarrollo principal para la navegación se basa en la infomación aportada por el sensor Kinect, sin embargo, el sensor láser proporciona una información muy potente para robots y también ha sido incluido en el desarrollo de este proyecto, utilizándolo en conjunto con el sensor Kinect.

Seguidamente, se han realizado los ajustes pertinentes en la navegación del robot, para la cual se ha seguido el concepto de mapas de coste y descomposición en celdas. También se ha valorado la disposición de ambos sensores para capturar el entorno, así como el tratamiento dispar de los datos capturados por cada uno de ellos. De esta

forma logramos que no se produzcan detecciones de objetos de manera duplicada y que no haya discrepancias entre los obstáculos que detecta un sensor respecto al otro¹.

Finalmente, se han incorporado características adicionales que aportan valor al desarrollo del proyecto, como el uso del simulador Gazebo o la interacción con el robot mediante comandos de voz y sintetizado de voz.

4.2 Planificación del proyecto

En este apartado se desarrollan las fase por las que ha pasado este proyecto y realizaremos un análisis de tiempos.

Fase inicial: Familiarización con el entorno ROS y elección de herramientas.

- i. Utilizaremos las herramientas proporcionadas por ROS para evaluar los datos que pueda manejar el robot: RViz, rqt_graph, map_server, rostopics...
- ii. Para el control del movimiento del robot utilizaremos el nodo ROSAria debido a sus amplias posibilidades.
- iii. Para acceder a la información de la Kinect utilizaremos los drivers libfreenect, por ser librerías de código libre y utilizadas ampliamente.

Segunda fase: Realización del nodo de teleoperación y comunicación entre equipos conectados a la misma red.

- i. Utilizamos la configuración de equipos en red para acceder a la información publicada por nodos que se ejecuten en varias máquinas **referencia**
- ii. Partiendo del nodo de teleoperación de "Turtlesim"**refencia*, realizamos un nodo similar para nuestro robot.

Tercera fase: Incorporación de los sensores al robot y acceso a los datos.

- i. Para el sensor Kinect, realizamos un adaptador para conectarlo a la alimentación del robot. Utilizando el nodo "freenect_stack"**REFERENCIA*, accedemos a la nube de puntos y la imagen.
- ii. Utilizamos el nodo LMS1xx **Referencia** para la puesta en marcha del láser y el acceso a los datos.
- iii. Incorporación de sistemas de referencia "base_link", "laser", "camera_link" y sus transformadas mediante el paquete "tf"**referencia**.

¹De especial interés en la incorporación de obstáculos al mapa mediante el uso de Costmaps en el sistema de navegación de ROS

iv. Visualización del conjunto de datos junto con los ejes de referencia en RViz.

Cuarta fase: Incorporación del sistema de navegación y ajuste de los parámetros

- i. Calibrado de los encoders de las ruedas del robot y ajuste de la odometría mediante RosAria.
- ii. Incorporación del sistema de navegación ROS de forma básica.
- iii. Navegación utilizando el sensor Kinect y el sensor Sick y generado de mapas mediante "slam_gmapping".
- iv. Ajuste de los planeadores de trayectoria del robot y parámetros de giro y control.

Quinta fase: Ajuste de la navegación y simulación mediante Gazebo

- i. Navegación en modo global (utilizando un mapa guardado) y en modo local (completamente reactivo).
- ii. Puesta en marcha del simulador Gazebo y configuración del robot en el entorno.
- iii. Disposición de los sensores de manera óptima y remodelado de la estructura física del robot.

Sexta fase: Nuevas funcionalidades y toma de datos.

- i. Incorporación de la funcionalidad "follower", adaptada a partir del robot Turtlebot **Referencia**, para el guiado del robot.
- ii. Interfaz de comandos por voz y sintetizador de texto a voz.
- iii. Pruebas físicas, recogida y análisis de los datos.

Análisis de tiempos:

Este proyecto fin de grado comenzó en Noviembre de 2014 y terminó en Febrero de 2015.

Durante el primer mes de Noviembre se estuvo recopilando información sobre ROS y su funcionamiento, los desarrollos existentes aplicados a robots reales y la filosofía del sistema.

En el mes de Diciembre se comenzó a trabajar con el robot, comprobando que todos los elementos se encontraban en correcto funcionamiento y se instaló el sistema operativo en su ordenador de abordo

Durante el mes de Enero se pudo avanzar menos debido a los exámenes y trabajos de las últimas asignaturas.

En el mes de Febrero se retomó el trabajo, empezando por una primera toma de contacto con la librería Aria y la ejecución de movimientos desde un ordenador externo conectado vía puerto serie.

Durante los meses de Marzo y Abril, el robot comenzó a funcionar con ROS, realizando los primeros movimientos con control por teclado. Seguidamente se realizó el nodo de telecontrol y un nodo para realizar movimientos basados tan solo en la odometría.

En el mes de mayo, se comenzaron a probar la compatibilidad con ROS de la cámara Kinect y el sensor Láser. Acto seguido, comenzaron las primeras pruebas de navegación autónoma.

En los meses de Junio y Julio, siguieron los ajustes en la navegación, tanto en el planificador de trayectoria como en los mapas de coste, así como en el sensor Kinect para la detección de obstáculos a diferente altura. Además se incorporó la funcionalidad de seguimiento.

En Julio también comenzaron las primeras pruebas de comandos de voz y la sintetización de voz.

Durante ese mes y el mes de Agosto, se comenzó a redactar gran parte del trabajo en esta memoria, donde se organizó la estructura del proyecto y la información a incluir.

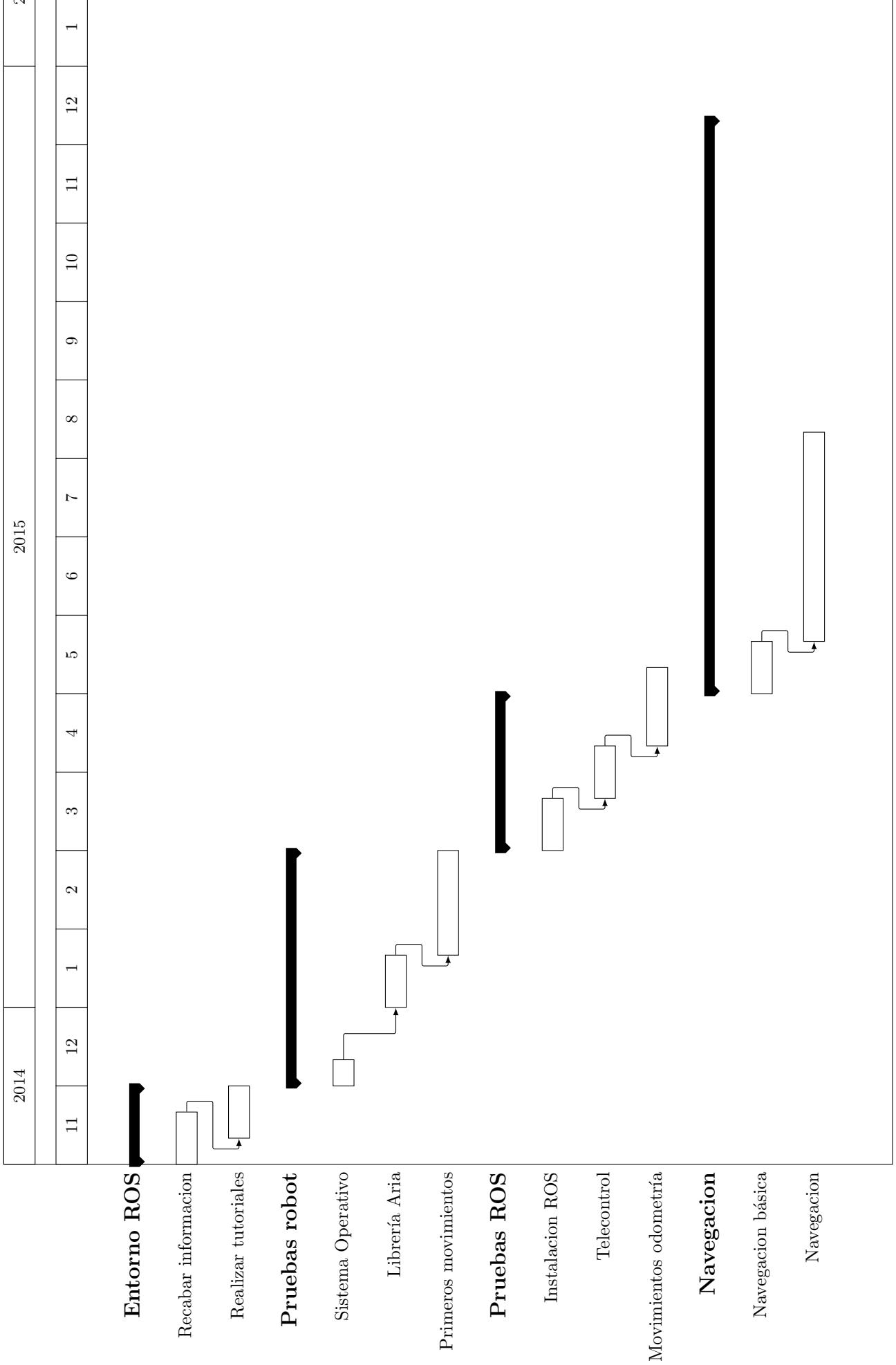
En el mes de Septiembre se decidió incorporar un ordenador más potente al robot y reestructurar su chasis para dejar el sistema desarrollado integrado de manera permanente. También se utilizó el array de micrófonos del sensor Kinect para los comandos de voz.

Durante el mes de Octubre se organizó la estructura del proyecto y se puso en marcha el simulador Gazebo. A continuación se realizó el ajuste de los sensores y la optimización del sistema de navegación. En paralelo se realizaron las modificaciones mecánicas y estructurales para la integración de los sensores y el ordenador en el robot.

Durante el mes de Noviembre se realizó un pequeño parón a nivel de software, se continuó con la parte mecánica y con la redacción de la memoria de este proyecto.

A continuación, comenzaron a realizarse las pruebas reales con la nueva configuración en el robot.

A continuación se muestra un diagrama de Gantt con el análisis de tiempos de las diferentes tareas.



4.3 Tecnologías y herramientas empleadas en el proyecto

En esta sección se describen tanto las tecnologías como las herramientas utilizadas en el desarrollo del proyecto.

4.3.1 Robot Operating System

El Sistema Operativo Robótico [ROSa] (conocido en inglés como Robot Operating System o ROS) es un framework para el desarrollo de software para robots que provee la funcionalidad de un sistema operativo [QCG+09]. ROS fue desarrollado originalmente en 2007 por el Laboratorio de Inteligencia Artificial de Stanford para dar soporte al proyecto del Robot con Inteligencia Artificial de Stanford [NQG+08]. Desde 2008, el desarrollo continua primordialmente en Willow Garage, un instituto de investigación robótico con más de veinte instituciones que colaboran conjuntamente.

ROS provee los servicios estándar de un sistema operativo como abstracción del hardware, control de dispositivos de bajo nivel, implementación de funcionalidad de uso común, paso de mensajes entre procesos y mantenimiento de paquetes. Está basado en una arquitectura de nodos interconectados que pueden mandar, recibir y multiplexar mensajes de sensores, control, estados, planificaciones y actuadores, entre otros. La librería está orientada para un sistema UNIX (Ubuntu (Linux)) aunque también se está adaptando a otros sistemas operativos como Fedora, Mac OS X, Arch, Gentoo, OpenSUSE, Slackware, Debian o Microsoft Windows, considerados como 'experimentales'.



Figura 4.1: Logo de ROS

ROS consta de dos partes básicas: la parte del sistema operativo, ros, como se ha descrito anteriormente y ros-pkg, una suite de paquetes aportados por la contribución de usuarios (organizados en conjuntos llamado en inglés "stacks") que implementan la funcionalidades tales como localización y mapeo simultáneo, planificación, percepción, simulación, etc.

ROS ofrece principalmente dos lenguajes de programación para acceder a su API (Application Programming Interface) completa. Esos lenguajes son C++ y Python [ROS13].

ROS es software libre bajo términos de licencia BSD. Esta licencia permite libertad para uso comercial e investigador. Las contribuciones de los paquetes en ros-pkg están bajo una gran variedad de licencias diferentes.

Actualmente ROS es mantenido y desarrollado de manera Open Source por

Open Source robotics Foundation [[OSF](#)], una organización independiente sin ánimo de lucro fundada por miembros de la comunidad robótica a nivel global.

4.3.2 Lenguaje de programación C++

El lenguaje C++ es un lenguaje orientado a objetos, y como tal, tiene como objetivo la reducción del tiempo de desarrollo aumentando la eficacia del proceso de generación de los programas.

Como consecuencia, los programas tienden a tener menos líneas de código y con más facilidad de introducir elementos nuevos escritos por otras personas.

Al tratarse de un lenguaje compilado, presenta una buena eficiencia en tiempo de ejecución frente a los lenguajes interpretados.

En sistemas operativos basados en Linux, el lenguaje C++ se compila bajo el compilador GCC (GNU Compiler Collection).

Dentro del desarrollo software en C++ para ROS (`roscpp` [[ROSb](#)]), existe una amplia interfaz para acceder a las diferentes funcionalidades y comunicarse con nodos desarrollados tanto en C++ como en Python.

4.3.3 Lenguaje de programación Python

Python es un lenguaje de programación interpretado cuya principal característica es que utiliza una sintaxis que favorece el código legible.

Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Gracias a sus características, su uso es totalmente flexible y permite un tiempo de desarrollo menor principalmente por su tipado dinámico y su sintaxis. Sin embargo, al tratarse de un lenguaje interpretado, el tiempo de ejecución es más alto lo cual no lo hace adecuado para tareas que requieran altos niveles de eficiencia.

Dentro del desarrollo en Python para ROS (`rospy` [[ROSc](#)]), existe una interfaz completa para comunicarse con los nodos y otras funcionalidades de ROS desarrolladas en Python o C++.

4.3.4 Controlador de versiones git y repositorios GitHub

Git es un software de control de versiones libre. Es decir, git gestiona los archivos y directorios y los cambios hechos en ellos a lo largo del tiempo. Esto te permite recuperar antiguas revisiones del proyecto o ver tu historial de cambios.

Git fue creado pensando en la eficiencia y la confiabilidad del mantenimiento e versiones cuando estas tienen un gran número de archivos de código fuente. Tiene la capacidad de poder trabajar varias personas con el mismo paquete siempre que no modifiquen el mismo archivo, en ese caso, sería posible ver las diferencias entre ambas versiones, y unirlas o crear unrama del proyecto principal si fuera necesario tener las dos versiones.

GitHub **Referencia pag web**es un sistema de almacenamiento público de código fuente (de cualquier tipo) o un servicio de repositorios. Su principal característica es la de ofrecer una plataforma de interacción social [DSTH12]en la que distintas personas pueden trabajar en conjunto. Esto permite que varios desarrolladores contribuyan a un proyecto y trabajen de manera coordinada.

Tanto para el desarrollo software de este proyecto como para la redacción de esta memoria se han utilizado estas herramientas, y el acceso a los repositorios se encuentran en las siguientes direcciones:

- Desarrollo software del proyecto https://github.com/danimtb/pioneer3at_ETSIDI
- Memoria del proyecto https://github.com/danimtb/TFG_pioneer3at

4.3.5 Simulador de robótica Gazebo

Gazebo [Ope] es un simulador de robótica en tres dimensiones que ofrece la simulación de complejos entornos de diversas características, así como robots de todo tipo, su interacción con el entorno y la representación visual de datos obtenidos por diversos sensores como cámaras, láseres, ultrasonidos...

Un buen simulador de robótica es esencial para cualquier tipo de desarrollo robótico, ya que podemos realizar las pruebas software o la viabilidad de un sistema antes de construirlo. Gazebo cuenta con un potente motor de física simulada, interacción con objetos y dinámica de los mismos [KH04].

Gazebo permite una integración completa con ROS, gestiona modelos físicos de robots utilizando el formato URDF (Unified Robot Description Format) [Gar11] y añade características específicas como el tipo de material, los momentos de inercia o el modelo de colisión. Además, incorpora plug-ins (funcionalidades añadidas) que permiten la simulación de robots de tipo diferencial, simulación de sensores y el cálculo de transformadas entre los distintos sistemas de referencia.

Gazebo es mantenido y desarrollado actualmente por la Open Source Robotics Foundation [OSF].

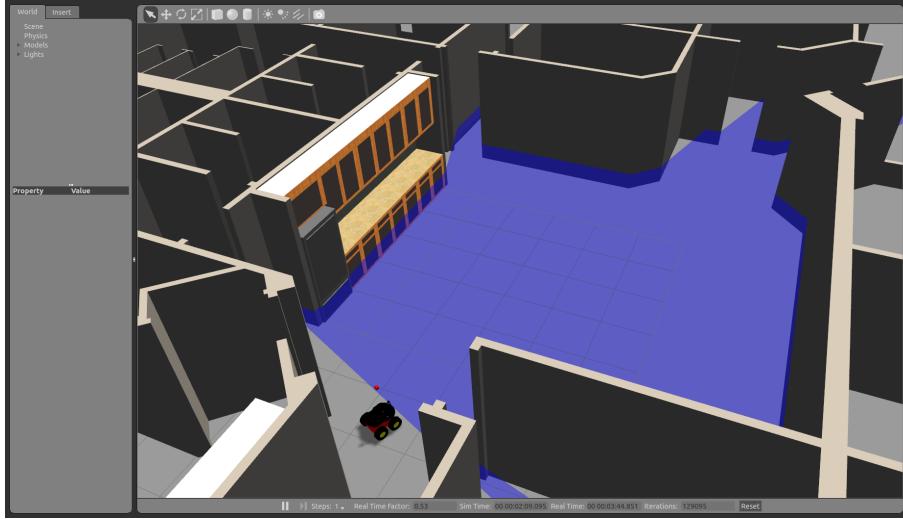


Figura 4.2: Entorno de simulación Gazebo

4.3.6 RViz: Herramienta de visualización robótica

RViz es una herramienta para la visualización de datos en 3 dimensiones de forma gráfica que trabaja dentro del entorno ROS (Figura 4.3). Esta aplicación nos permite ver lo que está ocurriendo en nuestra plataforma robótica a tiempo real.

RViz puede usarse para mostrar lecturas de sensores, datos devueltos por sensores de percepción en 3 dimensiones (nubes de puntos), visualizar mapas, visualizar un modelo de nuestro robot y su posición...

También puede utilizarse para interactuar con nuestro robot, utilizando marcadores interactivos o su interfaz de usuario.

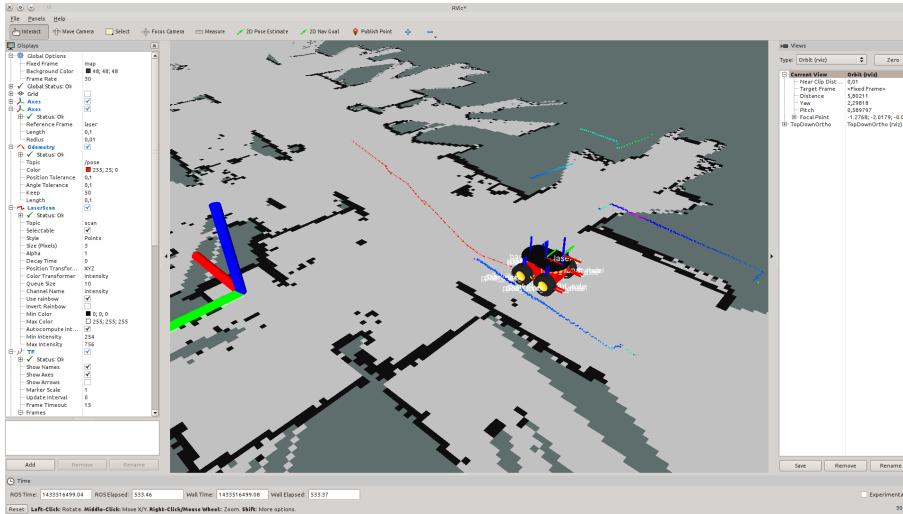


Figura 4.3: Entorno gráfico RViz

Al uso, RViz es un nodo más dentro de ROS que se suscribe o publica mensajes a otros nodos.

Su uso está muy extendido en ROS ya que nos permite entender lo que ocurre alrededor del robot y la información que manejan los nodos dentro del sistema.

4.3.7 Impresión 3D

4.4 Hardware

En esta parte se explica detalladamente el hardware empleado en el desarrollo del proyecto.

4.4.1 Pioneer 3 AT

El robot Pioneer 3 AT (Figura 4.4), perteneciente a la empresa Adept MobileRobots, es un robot de cuatro ruedas en configuración skid-steer y todo terreno (AT, All Terrain) de operación e investigación en laboratorio.



Figura 4.4: Robot Pioneer 3-AT

Su configuración en skid-steer permite un control relativamente simple utilizando el modo diferencial para poder realizar giros con gran maniobrabilidad, sin embargo, esta configuración depende mucho del tipo de suelo, con lo que se pierde precisión.

Este robot dispone de baterías, interruptor con parada de emergencia, dos motores de corriente continua para cada par de ruedas con transmisión mediante correa, encoders para leer la odometría y un microcontrolador con firmware ARCOS.

Ademas cuenta con un pequeño computador interno conectado al microcontrolador que puede utilizarse para realizar operaciones de manera autónoma.

El cuerpo del robot es de aluminio y su parte delantera así como superior es fácilmente desmontable para realizar las conexiones pertinentes y acceder al ordenador de a bordo y la placa microcontroladora. En la plataforma superior se sitúa el panel de control (Figura 4.5)para acceder al ordenador de abordo conectando un

monitor, teclado y ratón, puerto serial RS-232, botones de encendido y reset varios leds indicadores de estado y de envío y recepción de datos.

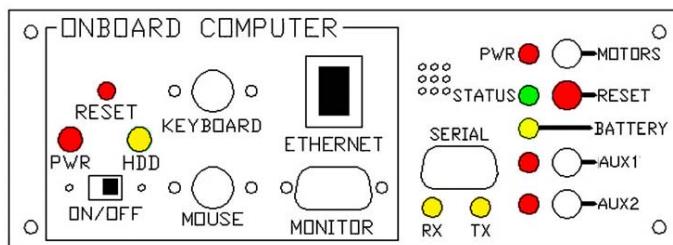


Figura 4.5: Panel de control del robot Pioneer 3-AT

En la siguiente tabla (Tabla 4.1) se describen las principales características del robot.

Especificaciones	Pioneer 3 AT
Largo	508 mm
Ancho	497 mm
Alto	277 mm
Distancia al suelo	80 mm
Peso	12 kg
Carga útil	32 kg
Cuerpo	Aluminio de 1.6 mm
Baterías	3 de 12 V Ah, estancas, plomo-ácido
Autonomía	4-8 horas
Sistema motriz	4 ruedas motrices
Ruedas	Neumáticos de Nylon
Diámetro de rueda	222 mm
Ancho de rueda	88 mm
Sistema de giro	Diferencial
radio máxima curvatura	40 cm
Radio de giro	0 cm
Máxima velocidad de avance	1.2 m/s
Máximo escalón	10 cm
Máximo hueco	15.2 cm
Terreno	Asfalto, Tierra, Césped, etc.
Encoders	500 pulsos
Procesador	Hitachi H8S

Tabla 4.1: Especificaciones del robot Pioneer 3 AT

4.4.2 Sensor Kinect

Kinect es un conjunto de sensores de bajo coste que lo convierte en una herramienta excepcional (Figura 4.6). Este dispositivo incluye una cámara de vídeo RGB, una cámara infrarroja de profundidad, un array de micrófonos y altavoces, un acelerómetro y un pequeño motor que le permite hacer movimientos de inclinación.



Figura 4.6: Sensor Kinect

Su función principal es la de percibir el entorno captando una serie de puntos que se ubican en las tres dimensiones. Su funcionamiento a grandes rasgos se basa en un emisor de infrarrojos a 830 nm que interactúa con los objetos y una cámara infrarroja que detecta la diferencia entre la proyección anterior y la actual, obteniendo la distancia a cada objeto.

En primer lugar, el laser infrarrojo es emitido por Kinect con un patrón determinado (Projected textures ****REFERENCIA****), el cual no es simétrico sino que tiene puntos aleatorios que se dispersa gracias a unas lentes de proyección. Estos puntos aleatorios se reflejan en los objetos, los cuales sería posible verlos con una cámara externa.

A continuación, al sensor de Kinect MT9M001C12STM, que no es más que el sensor CMOS de una cámara en la que se le trata para que observe solo el infrarrojo, obteniendo los puntos infrarrojos en el plano 2D. El motivo por el que podemos medir la profundidad de los objetos (su distancia) es porque sabemos el patrón de cómo emite el laser emisor [Kon10], por tanto sabremos que si un punto no está en el sitio que corresponde, se ha trasladado respecto al punto inicial y se le aplica la correspondiente transformación (Figura 4.7), obteniendo finalmente los puntos de toda la nube en coordenadas cartesianas XYZ.

La siguiente tabla (Tabla 4.2) muestra las especificaciones del sensor Kinect.

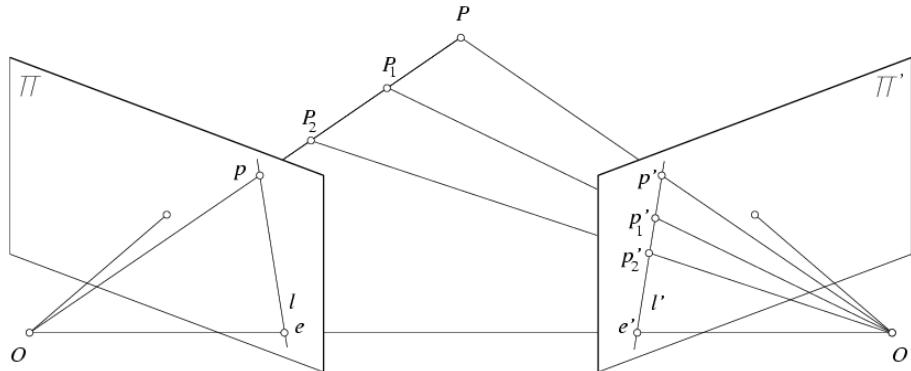


Figura 4.7: Proyección de infrarrojos y obtención de la nube de puntos

Especificaciones	Sensor Kinect
Dimensiones del conjunto	270mm x 50mm x 70mm
Fuente infrarroja	830nm
Potencia	60 mW
Cámara Infrarroja	MT9M001C12STM
Resolución cámara infrarroja	1200x960 pixeles
Frecuencia	30 Hz
Tamaño pixel	5.2um x 5.2um
Pixel activos	1280H x 1024V
Campo de visión	58° H, 45° V, 70° D
Resolución espacial	3mm (a 2 metros de distancia)
Resolución de profundidad	1cm (a 2 metros de distancia)
Distancia de operación	0.45m ? 6.5m
Cámara RGB	MT9M112
Resolución cámara RGB	640 x 480
Audio	TAS1020B (Controlador de Audio)
Formato	16kHz, 16-bit mono, modulación por codificación de pulso (PCM)
Entrada de audio	4 micrófonos con conversión analógico digital de 24bits
Acelerómetro	KXSD9-2050

Tabla 4.2: Características del sensor Kinect

4.4.3 Láser SICK LMS100

Aunque el planteamiento inicial del proyecto planteaba la navegación basada únicamente en el sensor kinect, debemos mencionar el uso del sensor láser Sick LMS100 (Figura 4.8).

Este es un sensor láser por infrarrojos de clase I (Inofensivo para el ojo humano), que obtiene la medida de distancias con gran precisión y rapidez en un solo plano y realizando un barrido de 270° (Figura 4.9).

Este sensor está colocado en la parte trasera del robot, enfocando hacia atrás para cubrir un mayor rango y conocer todo el entorno alrededor del robot.

En la siguiente tabla (Tabla 4.3) se recogen sus características principales.



Figura 4.8: Sensor escaner láser Sick LMS100

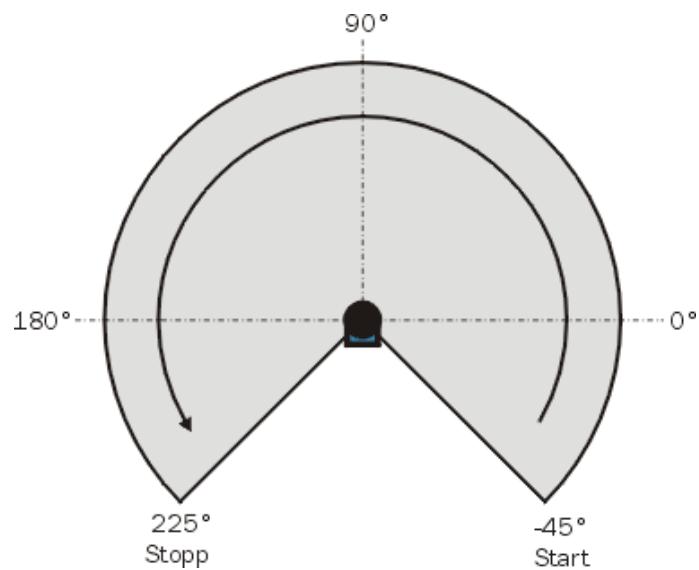


Figura 4.9: Campo de visión del sensor láser Sick LMS100

Especificaciones	Sick LMS100
Campo de aplicación	Interno
Fuente infrarroja	905 nm
Clase Láser	1 (IEC 60825-1)
Campo de visión	270°
Frecuencia de escaneo	25Hz/50Hz
Resolución angular	0.25°/0.5°
Distancia de operación	0.05 - 20 m
Tiempo de respuesta	20 ms
Error	30 mm
Interfaz de datos	Ethernet
Tensión de operación	10.8V - 20V DC
Consumo	20 W
Peso	1.1 Kg
Dimensiones	105mm x 102mm x 152mm

Tabla 4.3: Características del sensor láser Sick LMS100. Basado en [SIC09]

4.4.4 Intel NUC NUC5i7RYH

El ordenador Intel NUC NUC5i7RYH, es un ordenador de altas prestaciones y de tamaño compacto que ofrece unas buenas características para procesar datos y realizar la algoritmia adecuada para tareas de robótica.

Está equipado con un procesador Intel i7-5557U de quinta generación que ofrece una frecuencia de reloj de 3.1 GHz. Está incorporado con un disco duro de estado sólido que permite una alta velocidad de lectura y escritura en disco, así como una tarjeta RAM de tipo DDR3L de 8GB que permitirá el intercambio de información entre los nodos ROS de una manera fluida.

Su cometido será el de procesar la información de los sensores, generar los mapas incorporando los obstáculos, generar las trayectorias de navegación y comandar los motores del robot para realizar movimientos.

Dispone de tamaño compacto y un consumo bajo, juto con una alimentación a partir de los 12 voltios, lo que lo hace ideal para incorporarlo en robots móviles que requieran realizar tareas sin depender de una infraestructura.

En la tabla 4.4 pueden consultarse sus características principales.

Especificaciones	Intel NUC NUC5i7RYH
Procesador	Intel Core i7-5557U, dual-core
Frecuencia de reloj	3.1 GHz hasta 3.4 GHz
Memoria RAM	DDR3L1 **DATO**
Disco duro	M.2 SSD **DATO**
Gráficos	Iris Graphics 6100
Conectividad de periféricos	2 x USB 3.0 en el panel posterior 2 x USB 3.0 en el panel frontal 2 x USB 2.0 internos vía colector
Conectividad de red	Intel 10/100/1000 Mbps Intel® Wireless-AC 7265 M.2, antenas inalámbricas (IEEE 802.11ac)
Alimentación	12-19V DC
Consumo	65 W
Dimensiones	115mm x 111mm x 48.7mm

Tabla 4.4: Características del ordenador Intel NUC NUC5i7RYH

Capítulo 5

Arquitectura

Esta sección tiene como objetivo plantear la arquitectura general utilizada en el robot, las comunicaciones con el resto del hardware y con los nodos que proporcionan la información necesaria para que el robot sea totalmente autónomo.

5.1 Arquitectura general

A nivel de hardware utilizado en el proyecto, como es el propio robot, los sensores el ordenador de abordo el sistema se estructura de a siguiente manera:

- i) Robot Pioneer 3 AT: Este es el robot mencionado anteriormente, el cual debe ser configurado para acceder al puerto serie RS-232 de su placa controladora. Esto nos permite conectarnos con el firmware ARCOS **referencia** y comunicarnos a través de la librería ARIA. De esta forma controlamos los motores y podemos leer el valor de los encoders de la odometría.
- ii) Sensores: Tanto el sensor Kinect como el sensor láser irán alimentados a través de las baterías del robot y se comunicarán con el ordenador de abordo a través de puerto USB y ethernet respectivamente.
- iii) Ordenador Intel NUC: Será el ordenador de abordo encargado de ejecutar ROS y realizar todo el procesamiento necesario. Irá equipado con el sistema operativo Ubuntu 14.04 por ser la última versión estable disponible a fecha de la entrega del proyecto. Irá conectado al robot mediante un convertidor RS-232 a USB, el sensor láser se comunica vía ethernet y el sensor Kinect a través de puerto USB igualmente. También se conectarán el audio al altavoz integrado del robot.
- iv) Ordenador externo: Como se ha mencionado anteriormente, un ordenador externo opcional equipado con ROS podrá utilizarse para realizar tareas de supervisión inalámbrica a través de RViz y para realizar la teleoperación del robot vía TCP/IP integrado en ROS.

5.2 Arquitectura del proyecto

El proyecto está estructurado siguiendo la filosofía de "paquetes" desarrollados en ROS. Los paquetes se compilan dentro de un entorno de trabajo tipo Catkin **REFERENCIA** que se encarga de compilar correctamente a través de CMake **referencia** todos los ejecutables y de realizar el enlazado correctamente.

En el directorio raíz del proyecto por tanto, encontraremos los paquetes necesarios para que el sistema funcione:

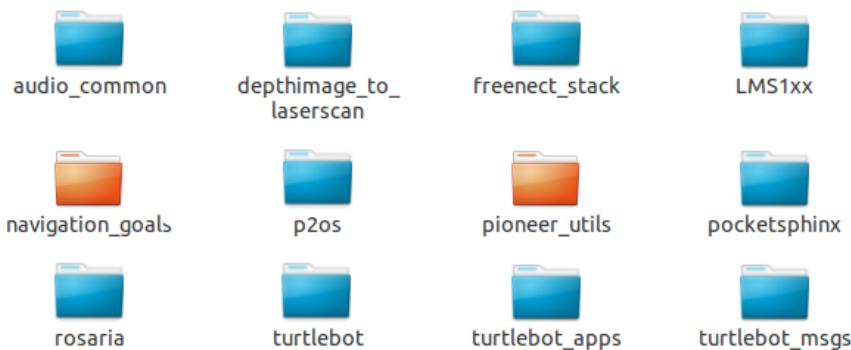


Figura 5.1: Estructura del proyecto

Las carpetas de color azul son paquetes desarrollados por terceros que no vienen integrados por defecto en ROS. El procedimiento para utilizarlos es clonar su repositorio en github e incluirlos como submodulos dentro de nuestro proyecto.

A continuación se realiza una breve descripción de cada uno:

- **audio_common:** Agrupa todas las funcionalidades para reproducir sonidos y voz sintetizada.
- **depthimage_to_laserscan:** Nodo que realiza la conversión del tipo de dato Pointcloud2 a laserscan mediante el análisis de imagen. Configurable mediante parámetros.
- **freenect_stack:** Agrupa los nodos controladores del sensor Kinect basados en libfreenect y tranfroman toda la información a la interfaz de ROS.
- **LMS1xx:** Nodo para la conexión con los sensores Láser Sick De la familia LMS100 a través de puerto ethernet.
- **p2os:** Agrupa utilidades y nodos para conectarse con los robots de la familia Pioneer, en especial Pioneer 3 AT y 3 DX. Ofrece modelos 3D de cada robot y algunos parámetros de configuración de los robots.
- **pocketsphinx:** Utilidad para el reconocimiento de voz mediante cualquier tipo de micrófono.
- **rosaria:** Interfaz de comunicación ROS con la librería Aria para el control de los motores del robot y la lectura de los encoders. Ofrece parámetros de calibración

de los encoders y acceso al array de ultrasonidos del robot (funcionalidad no incorporada en el robot utilizado para este proyecto).

- turtlebot, turtlebot_apps y turtlebot_msgs: Paquetes que agrupan funcionalidades para el robot Turtlebot, usadas en este caso en nuestro desarrollo.

Los paquetes ROS de desarrollo propio en este proyecto son los indicados en color naranja.

El paquete navigation_goals es un nodo que realiza navegación a través de puntos establecidos de modo que se puedan programar rutas a seguir por el robot.

El paquete pioneer_utils es donde se encuentra el desarrollo principal de este proyecto y se describirá en detalle a continuación.

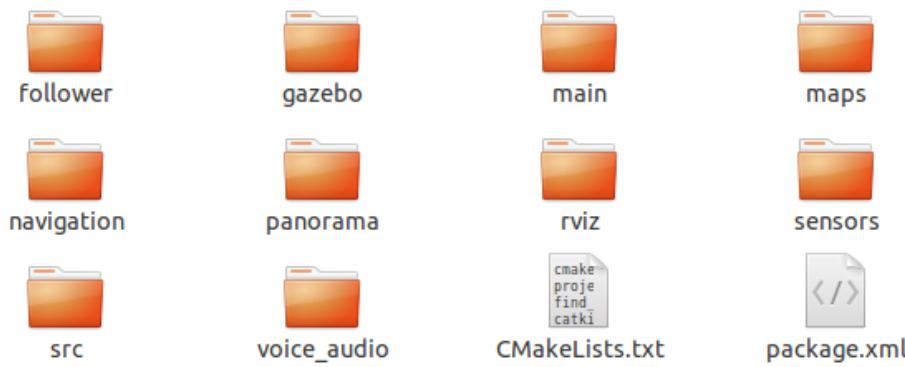


Figura 5.2: Estructura de carpetas del paquete pioneer_utils

Como cualquier paquete en ROS, disponemos de un archivo CMakeLists.txt y un package.xml, donde se indican los objetivos a compilar y las librerías adicionales para el enlazado así como las dependencias respectivamente.

En cada carpeta encontramos:

- follower: Archivos de configuración para utilizar la funcionalidad follower del robot Turtlebot en el nuestro.
- gazebo: Configuraciones necesarias para ejecutar Gazebo utilizando un modelo del robot Pioneer 3 AT y sus sensores del mismo tipo que el utilizado en la realidad. Se incluyen launchfiles ****OJO**** para realizar la navegación con diferentes mapas.
- main: Launchfiles de las aplicaciones principales que incorpora el robot ****COMPLETAR****.
- maps: Mapas guardados realizados con la funcionalidad de slam_gmapping.
- navigation: archivos y ajustes necesarios para realizar navegación del robot con y sin mapa (global navigation o local navigation).
- panorama: Archivos de configuración para utilizar la funcionalidad panorama del robot Turtlebot en el nuestro.

- rviz: Configuraciones preguardadas para rviz.
- sensors: Archivos de configuración para tener acceso a toda la información del hardware y los sensores que se utilizan.
- src: Carpeta donde se incluye el código fuente en C++ o Python de los nodos de teleoperación, test de navegación y navegación por estima (Dead reckoning).
- voice_audio: Archivos de configuración para el nodo de reconocimiento de voz y sonido, diccionarios de palabras y pronunciación.

Capítulo 6

Entorno ROS

La versión del software ROS utilizada para el desarrollo del proyecto ha tratado de ser siempre la más actual posible, ya que eso nos asegura mantener la compatibilidad en futuros trabajos y que el software esté actualizado.

La versión utilizada fue ROS Indigo Igloo desde el comienzo del proyecto, bajo el sistema operativo Ubuntu 14.04. A fecha de entrega del proyecto existe una versión más actualizada del software ROS, sin embargo se desestimó su uso debido a que aún no era una versión estable y algunos paquetes no se encontraban disponibles para tal versión.

Dentro de ROS existen diferentes conceptos como los nodos, los rostopics, ros-services...

- Nodos. - Servicios. - Rostopics.

6.0.1 Funcionamiento de ROS

6.1 Configuración de ROS

Para la instalación de ROS es necesario seguir ciertos pasos bien explicados en la wiki de su página.

Para el ordenador de abordo del robot utilizamos la versión completa del software.

Añadimos nuestra variable de entorno para buscar en los directorios de ros cuando ejecutemos los nodos.

Para el desarrollo dentro de ROS se utiliza el entorno de desarrollo Catkin, que facilita el enlazado y compilación de los paquetes. para ello es necesario tenerlo instalado:

instalacion catkin

Y a continuación es necesario incluir nuestro directorio de desarrollo para que sea reconocido:

source

A partir de este punto podríamos realizar nodos utilizando las funcionalidades de ROS.

Para el desarrollo de este proyecto es necesario instalar las siguientes funcionalidades adicionales de ROS:

- Navegacion - Rosaria - pocketsphinx - rosdeps - mirar cuaderno

6.2 Configuración de los paquetes ROS

Los paquetes ROS son funcionalidades desarrolladas por terceros que se integran en el sistema ROS y que son transferibles de un robot a otro.

Los paquetes ROS incorporan un archivo CMakeLists.txt para la compilación de los nodos que se hayan desarrollado así como sus mensajes. Y un archivo Package Manifest package.xml, donde se indican los requisitos del paquete, el autor, el contacto, y las dependencias del mismo.

Capítulo 7

Control primario

En este capítulo se realiza una primera aproximación al control del robot y a los nodos básicos que deben ejecutarse para controlarlo y acceder a la información de los sensores.

7.1 Nodos hardware

Los nodos necesarios para el control del robot requieren el acceso a los motores y a la lectura de la odometría. También es necesario disponer de controladores para ambos sensores utilizados y que su información se publique en tipos de datos reconocibles por ROS. Los nodos utilizados para estos dispositivos hardware se describen a continuación.

7.1.1 Control del robot: Rosaria y p2os

Como hemos indicado anteriormente, la librería que nos proporciona el acceso a la placa controladora de nuestro robot Pioneer es Aria. Esta librería es la que proporciona Adept Mobile Robots para realizar el control completo del robot y acceder a sus parámetros configurables.

Los paquetes disponibles en ROS para el control de los robots de la familia Pioneer son dos, por un lado tenemos Rosaria y por otro p2os.

p2os es un paquete que agrupa conjunto de utilidades y nodos desarrollados para controlar el robot. Su característica principal es que accede de manera nativa a la placa controladora del robot y no dependen de la librería Aria. Además incorpora funcionalidades configuradas como modelos 3D de robot, simulación con Gazebo o la configuración de la navegación.

Sin embargo, p2os no integra todas las funcionalidades a las que tiene acceso Aria como son la reconfiguración de los parámetros de la odometría.

Rosaria es un nodo de interfaz entre ROS y Aria, por tanto incluye todas prácticamente todas las funcionalidades de esta. Podemos acceder a la calibración de los encoders de la odometría así como conectar con el simulador MobileSim (ver sección 10.1)

A continuación se muestra el launchfile para ejecutar el nodo RosAria:

```

1 <launch>
2 <!-- Starting rosaria driver for motors and encoders -->
3   <node name="rosaria" pkg="rosaria" type="RosAria" args="_port:=/dev/ttyUSBO">
4     <rosparam>
5       TicksMM: 166
6       RevCount: 37350
7       DriftFactor: 0
8     </rosparam>
9     <remap from="~cmd_vel" to="cmd_vel"/>
10    </node>
11 </launch>
```

Fuente: *pioneer_utils/sensors/rosaria.launch*

Código 7.1: Launchfile para RosAria.

Como puede verse, podemos modificar los valores usados por Aria para realizar el cómputo de la odometría.

ROSTOPICS UTILIZADOS

7.1.2 Sensor Kinect

Para la puesta en marcha del sensor Kinect existen en ROS diferentes paquetes que utilizan una u otra librería de código en función de quién haya lo haya desarrollado.

Existen dos paquetes destinados al control del sensor Kinect:

Por un lado tenemos *openni_kinect* **referencia**, que utiliza los drivers originales desarrollados por la empresa PrimeSense encargada de fabricar este dispositivo. Este paquete y en concreto los drivers del dispositivo han sido utilizados ampliamente tanto en desarrollos realizados con ROS como fuera de este entorno. Sus características principales son la total funcionalidad, aprovechamiento de toda la tecnología de este sensor y capacidad para monitorear la posición del esqueleto de una persona. Sin embargo, desarrollo del paquete *openni_kinect* solo se mantuvo activo hasta a versión de ROS Fuerte **referencia** debido a la compra de PrimeSense por la conocida marca de informática Apple **referencia**.

Por otro lado, gracias al gran desarrollo software llevado a cabo por la comunidad OpenSource, disponemos de los divers *libfreenect* desarrollados por el proyecto OpenKinect http://openkinect.org/wiki/Main_Page que trata de ofrecer una vía alternativa para controlar el sensor de Microsoft. Estos drivers se encapsulan y adaptan su interfaz a ROS a través del paquete *freenect_stack* **referencia** el cual nos ofrece acceso tan solo a la imagen y la nube de puntos del sensor. Su integración no es completa, no dispone de características adicionales como el monitoreo de la posición de una persona, sin embargo su funcionamiento es correcto y está adaptado

a ROS en su versión Indigo y esto nos ofrece la posibilidad de integrarlo en nuestro sistema. Por estas razones ha sido el software utilizado para acceder al sensor Kinect en este proyecto.

freenect_stack, al ser un paquete de terceros, debe clonarse desde su repositorio de código fuente e incorporarlo a nuestro entorno catkin.

Su puesta en marcha es bastante inmediata y podemos hacer uso de los launch files que ofrece freenect_launch desde consola de la siguiente forma:

```
1 rosrun freenect_launch freenect.launch
```

Fuente: *freenect_stack/freenect_launch/launch/freenect.launch*

Código 7.2: Launchfile para Kinect en el paquete freenect_launch.

7.1.3 Sensor Láser Sick LMS100

Existe un amplio soporte para sensores láser de la marca Sick, entre ellos el más popular es la familia Sick LMS200 ya que se utiliza en muchos desarrollos relacionados con la robótica móvil **referencia**. Esa familia de sensores utiliza una interfaz de comunicación en serie a través de puerto RS-232, sin embargo, la familia de dispositivos Sick LMS100 utiliza interfaz ethernet y requiere un tratamiento de datos diferente.

El láser Sick LMS100 ha sido integrado en ROS y utilizado en este proyecto ya que se había dado uso en proyectos anteriores **referencia alejandro** y se consideró conveniente incorporarlo y utilizarlo para obtener una navegación más precisa del robot.

Para acceder al sensor Sick LMS100 utilizamos el paquete LMS1xx desarrollado por Clearpath Robotics **referencia** que se basa en el trabajo de otros dos desarrolladores de la comunidad ROS, y en concreto en los drivers desarrollados por **referencia** <https://github.com/konradb3/libLMS1xx>

El paquete LMS1xx consta de un solo nodo que se conecta a través de una IP indicada como parámetro. Su uso es sencillo mediante un archivo launchfile y tan solo debemos tener la precaución de configurar correctamente la IP manual del puerto ethernet de nuestro ordenador.

```
1 <launch>
2   <arg name="host" default="192.168.1.14" />
3   <node pkg="lms1xx" name="lms1xx" type="LMS1xx_node">
4     <param name="host" value="$(arg host)" />
5   </node>
6 </launch>
```

Fuente: *LMS1xx/launch/LMS1xx.launch*

Código 7.3: Launchfile para el sensor Láser Sick LMS100.

Pueden precisarse algunos ajustes previos con la herramienta que ofrece el fabri-

cante "SOPAS Engineering tool", los cuales pueden encontrarse en el anexo de este trabajo (Sección 12.2.4).

ROSTOPICS UTILIZADOS

7.2 Nodo de teleoperación

Uno de los primeros objetivos de este proyecto es realizar el control teleoperado del robot. Utilizando ROS y sus características para operar de manera distribuida en diferentes máquinas, esta tarea se vuelve inmediata para el usuario.

ROS trabaja en forma de procesos que se ejecutan de manera independiente y se comunican a través del nodo principal o Máster con el paso de mensajes. Ya que el máster dispone de una dirección IP en la máquina que lo ejecuta, basta con indicar en el entorno ROS de cada máquina la dirección de este para que los nodos abran una comunicación con esa dirección IP.

Los pasos para configurar las máquinas bajo la misma red se describen con detalle en la guía ROS NETWORKING y consisten básicamente en indicar en el script `.bashrc` el parámetro ROS_IP y ROS_MASTER_URI.

ROS_IP debe contener la IP que tenga nuestra máquina en la red que esté operando y ROS_MASTER_URI la dirección *http* correspondiente de la máquina donde se ejecute el nodo principal.

```
1 | export ROS_IP=10.42.0.1
2 | export ROS_MASTER_URI=http://10.42.0.1:11311
```

Fuente: `~/.bashrc`

Código 7.4: Líneas del archivo `.bashrc` en el ordenador de abordo.

```
1 | export ROS_IP=10.42.0.77
2 | export ROS_MASTER_URI=http://10.42.0.1:11311
```

Fuente: `~/.bashrc`

Código 7.5: Ejemplo `.bashrc` en un ordenador externo para realizar comunicación con el máster.

De esta manera podemos desarrollar un nodo ROS que se conecte al topic de RosAria que comanda los motores `cmd_vel` y publicar diferentes valores de velocidad en función de las teclas que se pulsen.

Fragmento de código teleoperación

ROSTOPICS UTILIZADOS

7.3 Nodo de navegación estimada

La navegación estimada, más conocida en inglés como Dead Reckoning **referencia**, es la capacidad para realizar navegación en un entorno basándose solamente en la información que aportan los sensores de la odometría.

Es un método estimado, ya que solo se basa en la información de los encoders y no tiene en cuenta el tipo de superficie, la inclinación, el rozamiento o incluso obstáculos que puedan frenar o modificar el desplazamiento del robot (a pesar de que sus ruedas giren).

Este nodo de navegación puede utilizarse para indicar al robot que avance cierta cantidad de metros y que realice giros a derecha o izquierda en un determinado ángulo.

Fragmento de código teleoperación

ROSTOPICS UTILIZADOS

7.4 Nodo de guiado (follower)

El nodo de guiado se basa en el procesamiento de la nube de puntos obtenida a través de los nodos del paquete freenect_stack.

El nodo está originalmente desarrollado para el robot Turtlebot pero es fácil adaptable a otros robots.

Requiere un rostopic de tipo pointcloud2 al que suscribirse para leer la nube de puntos y un rospotic de tipo **VELOCIDAD** al que publicar los movimientos de giro, avance y retroceso.

EL tratamiento de la nube de puntos se realiza con la librería PCL (PointCloud Library) **referencia** y su funcionamiento es el siguiente:

- 1.- Busca puntos que sobresalgan del plano principal.
- 2.- Calcula las dimensiones de esa zona.
- 3.- Calcula el centroide de la zona destacada.
- 4.- Mueve el robot de manera acorde y mantiene la distancia.

código del funcionamiento y paquetes necesarios

LAUNCH FILE

ROSTOPICS UTILIZADOS

Capítulo 8

Navegación

8.1 Navigation Stack

8.1.1 Requisitos para la navegación

8.1.2 SLAM mediante slam_gmapping

8.2 Costmaps

8.3 Planificador de trayectoria global

8.4 Planificador de trayectoria local

8.5 Navegacion global y local

Capítulo 9

Implementación del sistema

9.1 Nodo de navegacion por puntos

9.2 Nodo de comandos por voz

9.2.1 Reconocimiento de comandos de voz

9.2.2 Feedback mediante text-to-speech

9.3 Primera configuración

9.4 Nodo de ejecución automática de nodos

Capítulo 10

Pruebas del sistema

10.1 Simulación con MobileSim

10.2 Simulación con Gazebo

10.2.1 Segunda configuracion

10.2.2 Datos de navegacion

10.3 Visualización mediante RViz

10.4 Pruebas reales

10.4.1 SLAM

10.4.2 Test de resistencia

10.4.3 Aspectos de la navegación

Capítulo 11

Conclusion

Se presentan a continuación las conclusiones...

11.1 Conclusión sobre la metodología

11.2 Conclusión sobre los resultados

Una vez finalizado el proyecto...

11.3 Desarrollos futuros

Un posible desarrollo...

Bibliografía

- [AdN87] Isaac Asimov and Rosa S de Naveira. *Robots e imperio*. Círculo de lectores, 1987.
- [AMRS11] F Alonso-Martin, Arnaud A Ramey, and Miguel A Salichs. Maggie: el robot traductor. In *Proceedings of the 9th Workshop RoboCity2030-II, Madrid, Spain*, pages 57–73, 2011.
- [BPBA07] Antonio Barrientos, Luis Felipe Peñin, Carlos Balaguer, and Rafael Aracil. *Fundamentos de robótica*. McGraw-Hill, Interamericana de España, 2007.
- [DL91] Stephen L Dickerson and Brett D Lapin. Control of an omnidirectional robotic vehicle with mecanum wheels. In *Telesystems Conference, 1991. Proceedings. Vol. 1., NTC'91., National*, pages 323–328. IEEE, 1991.
- [DL12] E Duliep Lescaille. Leonardo da vinci: Arte y técnica//leonardo da vinci: art and technique. *Ingeniería Mecánica*, 3(2):91–94, 2012.
- [dS07] Universidad de Sevilla. Sistemas de locomoción de robots móviles. http://www.esi2.us.es/~vivas/ayr2iae/L0C_MOV.pdf, 2007. [Online, consultado 15 de Junio de 2015].
- [DSTH12] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. Social coding in github: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1277–1286. ACM, 2012.
- [Gar11] Willow Garage. Unified Robot Description Format (URDF). <http://wiki.ros.org/urdf>, 2011. [Online, consultado 28 de Marzo de 2015].
- [Goo09] Google. Google self-driving car project. <https://www.google.com/selfdrivingcar/>, 2009.
- [Gui11] Erico Guizzo. How google?s self-driving car works. *IEEE Spectrum Online, October*, 18, 2011.
- [Gui13] Erico Guizzo. DARPA's Rescue-Robot Showdown. <http://spectrum.ieee.org/robotics/humanoids/darpas-rescuerobot-showdown>, 2013. [Online, consultado 11 de Junio de 2015].

- [KH04] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154. IEEE, 2004.
- [Kon10] Kurt Konolige. Projected texture stereo. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 148–155. IEEE, 2010.
- [MGB00] R Martínez and A García-Beltrán. Breve historia de la informática. *España, Universidad Politécnica de Madrid*, 2000.
- [NAS03] NASA. Mars Exploration Rover Mission: Overview. <http://mars.nasa.gov/mer/overview/>, 2003. [Online, consultado 20 de Junio de 2015].
- [NAS12] NASA. Curiosity Rover, Mars Science Laboratory. <http://www.nasa.gov/feature/jpl/nasas-curiosity-rover-team-confirms-ancient-lakes-on-mars/>, 2012. [Online; consultado 20 de Agosto de 2013].
- [Nev12] Nevada Department of Motor Vehicles. Nevada dmv issues first autonomous vehicle testing license to google. <http://www.dmvnv.com/news/12005-autonomous-vehicle-licensed.htm>, 2012. [Online, consultado 10 de Junio de 2015].
- [NKO⁺11] Keiji Nagatani, Seiga Kiribayashi, Yoshito Okada, Satoshi Tadokoro, Takeshi Nishimura, Tomoaki Yoshida, Eiji Koyanagi, and Yasushi Hada. Redesign of rescue mobile robot quince. In *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, pages 13–18. IEEE, 2011.
- [NQG⁺08] Andrew Y Ng, Morgan Quigley, Stephen Gould, Ashutosh Saxena, and Eric Berger. STAIR: STanford Artificial Intelligence Robot. <http://stair.stanford.edu/>, 2008.
- [Ope] Open Source Robotics Foundation. Gazebo. <http://gazebosim.org/>.
- [OSF] OSFR. Open Source Robotics Foundation. <http://www.osrfoundation.org/>. [Online; consultado 12 de Noviembre de 2014].
- [Paí11] El País. Fukushima vive el peor accidente nuclear desde Chernóbil. http://internacional.elpais.com/internacional/2011/03/12/actualidad/1299884402_850215.html, 2011. [Online, consultado 21 de Junio de 2015].
- [PBR06] R Playter, M Buehler, and M Raibert. Bigdog. In *Defense and Security Symposium*, pages 62302O–62302O. International Society for Optics and Photonics, 2006.
- [QCG⁺09] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source

- robot operating system. In *ICRA workshop on open source software*, volume 3, page 5, 2009.
- [Reu12] Reuters. Amazon's Kiva Systems Acquisition To Put Robots To Work. http://www.huffingtonpost.com/2012/03/19/amazoncom-kiva-acquisition_n_1365512.html, 2012. [Online, consultado 11 de Junio de 2015].
- [Rob02] Robotnik. Servicios de ingeniería robótica. <http://www.robotnik.es/>, 2002.
- [Roj07] Sergio Gálvez Rojas. Historia de la informática (i). *Encuentros en la Biología*, 119:2, 2007.
- [ROSa] ROS. Robot Operating System. <http://www.ros.org/>.
- [ROSb] ROS.org. rosCPP C++ API. <http://wiki.ros.org/roscpp>. [Online, consultado 10 de Noviembre de 2014].
- [ROSc] ROS.org. rospy Python API. <http://wiki.ros.org/rospy>. [Online, consultado 10 de Noviembre de 2014].
- [ROS13] ROS.org. ROS APIs. <http://wiki.ros.org/APIs>, 2013. [Online; consultado 10 de Noviembre de 2014].
- [SIC09] SICK. Laser scanners, LMS100-10000. <https://www.mysick.com/ecat.aspx?go=FinderSearch&Cat=Row&At=Fa&Cult=English&FamilyID=344&Category=Produktfinder&Selections=34242>, 2009. [Online; consultado 22 de Agosto de 2013].
- [SOGSBS⁺10] Ramón Silva Ortigoza, Rafael García Sánchez, Ricardo Barrrientos Sotelo, María Aurora Molina Vilchis, Víctor Manuel Hernández Guzmán, and Gilberto Silva Ortigoza. Una panorámica de los robots móviles. *Télématique*, 6(3):1–14, 2010.
- [Xat11] Xataka. Conducción autónoma y futuro de los coches. <http://www.xataka.com/automovil/conduccion-autonoma-y-futuro-de-los-coches>, 2011. [Online, consultado 10 de Junio de 2015].

Capítulo 12

Anexo I: Configuración del sistema

12.1 Configuración del espacio de trabajo

12.1.1 Instalación de las librerías

12.1.2 Gestión de las dependencias

12.2 Configuración del hardware

12.2.1 Calibración de los encoders

12.2.2 Ordenador de abordo

12.2.3 Sensor Kinect

12.2.4 Láser SICK LMS100

