Universidad Carlos III de Madrid

Escuela Politécnica Superior

Ingeniería en Informática

Proyecto Fin de Carrera

Sistema de teleoperación mediante una interfaz natural de usuario

Autor: Santiago Alfaro Ballesteros

Tutor: Moisés Martínez Muñoz

Octubre, 2012

A mi familia, por su apoyo.

A mis amigos, por su compañía.

A mis compañeros, por su ayuda.

A Ana, por su cariño.

RESUMEN

En este proyecto, se desarrollará un sistema de teleoperación para el robot humanoide Nao, de forma que una persona utilice su cuerpo para teleoperar el robot. Para llevar a cabo este proceso, se utilizará un dispositivo llamado Kinect fabricado por Microsoft. Este dispositivo está orientado a crear lo que se conoce como una interfaz natural de usuario; es decir, una forma de comunicación entre el hombre y la máquina mucho más instintiva, en la que se no se necesitan los periféricos de entrada clásicos: teclado, ratón y joystick. Esto permite eliminar las limitaciones que tienen los dispositivos clásicos cuando intentan controlar robots complejos, ya que la información sobre el estado del robot la poseerá el operario en la propia posición de su cuerpo, que tendrá una correspondencia 1:1 con la posición del robot. El uso de la interfaz natural de usuario facilita enormemente la teleoperación del robot por parte de cualquier persona, sin necesidad de una extensa formación previa, lo que amplía el número de sectores donde se puede aplicar esta tecnología.

ÍNDICE

ÍNDICE **1**

ÍNDICE DE FIGURAS **3**

ÍNDICE DE TABLAS **7**

1. INTRODUCCIÓN **10**

1.1. OBJETIVO 13

1.2. ESTRUCTURA DEL DOCUMENTO 15

2. ESTADO DE LA CUESTIÓN **17**

2.1. HISTORIA DE LA ROBÓTICA 17

2.1.1. Robot NAO *24*

2.2. TELEOPERACIÓN 28

2.2.1. Historia *28*

2.2.2. Elementos de un sistema de teleoperación *30*

2.2.2.1. Interfaz 31

2.2.2.1.1. Dispositivos de control 31

2.2.2.1.2. Dispositivos de realimentación 34

2.2.2.1.3. Dispositivos bilaterales 35

2.2.3. Aplicaciones *35*

2.3. KINECT 42

3. DISEÑO E IMPLEMENTACIÓN DEL SISTEMA **46**

3.1. METODOLOGÍA 46

3.2. ANÁLISIS 47

3.2.1. Alcance del proyecto *47*

3.2.2. Especificación de Casos de Uso *49*

3.2.2.1. Características de los casos de uso 49

3.2.2.2. Lista de casos de uso. 50

3.2.3. Especificación Requisitos *56*

3.2.3.1. Características de los requisitos 56

3.2.3.2. Catálogo de requisitos 57

3.3. DISEÑO 66

3.3.1. Arquitectura *66*

3.3.2. Descripción General del Sistema *70*

3.3.3. Descripción de componentes *73*

3.3.3.1. teleop_msgs 73

3.3.3.2. Capa operador 74
3.3.3.2.1. openni_camera 74
3.3.3.2.2. skeletal_tracker 76
3.3.3.2.3. pcl 80
3.3.3.2.4. hand_interaction 81
3.3.3.2.5. body_capture 84
3.3.3.2.6. recognizer 96
3.3.3.3. Capa de control 98
3.3.3.3.1. teleop_control 98
3.3.3.4. Capa robot 104
3.3.3.4.1. teleop_nao 104
3.3.3.4.2. nao_vision 105

4. EVALUACIÓN 106

4.1. DESCRIPCIÓN DE LOS EXPERIMENTOS 107
4.1.1. Experimento 1 107
4.1.2. Experimento 2 108
4.1.3. Experimento 3 109
4.1.4. Experimento 4 110
4.2. CUESTIONARIO DE EVALUACIÓN 111
4.3. RESULTADOS DEL CUESTIONARIO 118

5. GESTIÓN DEL PROYECTO 121

5.1. PLANIFICACIÓN 121
5.2. PRESUPUESTO 125

6. CONCLUSIONES 127

6.1 CONCLUSIONES GENERALES 127
6.2 REALIZACIÓN DE LOS OBJETIVOS 128
6.3 PROBLEMAS ENCONTRADOS 130
6.4 LÍNEAS FUTURAS 133

A. ROS 135

B. MODELO DE COMUNICACIÓN 140

C. INSTALACIÓN 148

D. EJECUCIÓN 150

E. MANUAL DE USUARIO 153

F. GLOSARIO DE TÉRMINOS 154

BIBLIOGRAFÍA 156

ÍNDICE DE FIGURAS

Figura 2-1: Portada de la primera edición de Rossum's Universal Robots 18

Figura 2-2: (A) Gallo de Estrasburgo y (B) Papamoscas de Burgos 19

Figura 2-3: (A) Telar de Joseph Marie Jacquard y (B) tarjeta perforada 20

Figura 2-4: (A) ELSIE y (B) SHAKEY 21

Figura 2-5: (A) Robot PUMA y (B) Genghis 22

Figura 2-6; (A) Furby, (B) Aibo y (C) LEGO Mindstorm 22

Figura 2-7: (A) Robot Spirit y (B) robot Romeo 23

Figura 2-8: Dimensiones de NAO 24

Figura 2-9: (A) Articulaciones en detalle del NAO y (B) posición de las cámaras ... 25

Figura 2-10: Resumen de los componentes de NAO 26

Figura 2-11: Partido de la Robocup 27

Figura 2-12: Goertz trabajando con el M1 29

Figura 2-13: Goertz trabajando con el E1 29

Figura 2-14: Elementos de un sistema de teleoperación 30

Figura 2-15: Joystick 32

Figura 2-16: Teleoperación maestro-esclavo en cirugía. 32

Figura 2-17: Teleoperación mediante exoesqueleto 33

Figura 2-18: Maqueta del Lunojod 1 36

Figura 2-19: MQ-9 Reaper, un UAV con capacidad de ataque con misiles 37

Figura 2-20: Sistema Quirúrgico Da Vinci 38

Figura 2-21: Sistema de exploración submarina JASON 38

Figura 2-22: Telesar V 39

Figura 2-23: iControlNao 40

Figura 2-24: Nao imitando la posición del cuerpo del operador 41
Figura 2-25: Personas jugando a la Xbox 360 con Kinect 42
Figura 2-26: Ejemplo del funcionamiento del sensor Kinect 44
Figura 2-27: Partes de Kinect 45
Figura 3-1: Esquema del sistema 48
Figura 3-2: Diagrama de casos de uso 50
Figura 3-3: Arquitectura del sistema 67
Figura 3-4: Elementos del diagrama 68
Figura 3-5: Esquema de componentes 69
Figura 3-6: Esquema del nodo teleop_msgs 73
Figura 3-7: Esquema del nodo openni_camera 74
Figura 3-8: Contenido del mensaje /camera/rgb/points 75
Figura 3-9: Esquema del nodo skeletal_tracker 76
Figura 3-10: Funcionamiento skeletal_tracker 1 77
Figura 3-11: Funcionamiento skeletal_tracker 2 78
Figura 3-12: Funcionamiento skeletal_tracker 3 78
Figura 3-13: Funcionamiento skeletal_tracker 4 79
Figura 3-14: Esquema del nodo pcl 80
Figura 3-15: Esquema del nodo hand_interaction 81
Figura 3-16: Demostración del piano virtual del MIT 81
Figura 3-17: Composición de los mensajes recibidos por detectskelhands 82
Figura 3-18: Ejemplo de mensaje enviado por detectskelhands 83
Figura 3-19: Procesamiento de las manos por parte de detecfingers 83
Figura 3-20: Procesamiento de las manos por parte de detecfingers 2 83
Figura 3-21: Esquema del nodo body_capture 84

Figura 3-22: Cálculo de la apertura del hombro 86
Figura 3-23 : Cálculo de la rotación del hombro 87
Figura 3-24: Cálculo de la apertura del codo..... 88
Figura 3-25: Ángulo de rotación del codo con el brazo cerrado 89
Figura 3-26: Ángulo de rotación del codo con el brazo abierto 89
Figura 3-27: Ejemplo del cálculo del ángulo de rotación del codo 90
Figura 3-28: Posibles posiciones que puede tomar la mano..... 91
Figura 3-29: Procesamiento de la mano 1 y 2 92
Figura 3-30: Procesamiento de la mano 3 y 4 92
Figura 3-31: Procesamiento de la mano 5 y 6 93
Figura 3-32: Procesamiento de la mano 7 y 8 94
Figura 3-33: Ejemplos de ángulos de la muñeca 1 95
Figura 3-34: Ejemplos de ángulos de la muñeca 2 95
Figura 3-35: Ejemplos de ángulos de la muñeca 3 95
Figura 3-36: Esquema del nodo recognizer 96
Figura 3-37: Esquema de funcionamiento del nodo recognizer 97
Figura 3-38: Esquema del nodo teleop_control 98
Figura 3-39: Comparación entre ángulos del operador y ángulos del Nao 99
Figura 3-40: Cálculo de la rotación y la inclinación de la cabeza del robot 100
Figura 3-41: Ejemplo del cálculo del ángulo de rotación de la cabeza 101
Figura 3-42: Posibles direcciones para la orden de andar 103
Figura 3-43: Esquema del nodo teleop_nao 104
Figura 3-44: Esquema del nodo nao_vision 105
Figura 4-1: Esquema del experimento 1 107
Figura 4-2: Esquema del experimento 2 108

Figura 4-3: Esquema del experimento 3 109

Figura 4-4: Esquema del experimento 4 110

Figura 5-1: Diagramas de Gantt con el tiempo real y el planificado 124

Figura 6-1: Utilización simultánea de las cámaras del NAO 131

Figura A-1: Ejemplo de posible infraestructura de un sistema en ROS 135

Figura A-2: Elementos de ROS 137

Figura B-1: Mensaje /camera/rgb/points 140

Figura B-2: Mensaje /skeletons 141

Figura B-3: Contenido del mensaje /Skeletons 141

Figura B-4: Mensajes /hand1_fullcloud y /hand0_fullcloud 143

Figura B-5: Mensaje recognizer/output 144

Figura B-6: Mensaje /bodyAngle 144

Figura B-7: Mensaje /walkInfo 145

Figura B-8: Mensaje /LHandOpening y /RHandopenind 146

Figura B-9: Mensaje /naoSays 146

Figura B-10: Mensajes del tipo teleop_msgs/joint_anlge 147

Figura D-1: Fichero teleop_nao.launch 151

Figura D-2: Fichero word_cmd.dic 152

Figura D-3: Fichero word_cmd.lm 152

ÍNDICE DE TABLAS

Tabla 3-1: Formato de descripción de caso de uso 49

Tabla 3-2: Caso de uso Iniciar teleoperación 50

Tabla 3-3: Caso de uso Detener teleoperación 51

Tabla 3-4: Caso de uso Abrir mano 52

Tabla 3-5: Caso de uso Cerrar mano 52

Tabla 3-6: Caso de uso Transmitir ordenes por voz 53

Tabla 3-7: Caso de uso Andar 54

Tabla 3-8: Caso de uso Girar 55

Tabla 3-9: Caso de uso Mover brazos 55

Tabla 3-10: Formato de tabla de requisito 56

Tabla 3-11: Requisito funcional Capturar posición 57

Tabla 3-12: Requisito funcional Capturar voz 57

Tabla 3-13: Requisito funcional Iniciar teleoperación 57

Tabla 3-14: Requisito funcional Detener teleoperación 58

Tabla 3-15: Requisito funcional Mover brazos 58

Tabla 3-16: Requisito funcional Andar hacia delante 58

Tabla 3-17: Requisito funcional Andar hacia atrás 58

Tabla 3-18: Requisito funcional Andar hacia la derecha 59

Tabla 3-19: Requisito funcional Andar hacia la izquierda 59

Tabla 3-20: Requisito funcional Girar a la derecha 59

Tabla 3-21: Requisito funcional Girar a la izquierda 59

Tabla 3-22: Requisito funcional Parar 60

Tabla 3-23: Requisito funcional Mover muñecas 60

Tabla 3-24: Requisito funcional Cambiar estado mano derecha 60
Tabla 3-25: Requisito funcional Cambiar estado mano izquierda 61
Tabla 3-26: Requisito funcional Mostrar cámara 61
Tabla 3-27: Requisito funcional Seguir manos 61
Tabla 3-28: Requisito funcional Mostrar información textual 62
Tabla 3-29: Requisito funcional Mostrar imágenes de Kinect 62
Tabla 3-30: Requisito funcional Notificaciones del robot 62
Tabla 3-31: Requisito inverso Mover piernas 63
Tabla 3-32: Requisito inverso Mover cabeza 63
Tabla 3-33: Requisito no funcional Utilizar el robot NAO 63
Tabla 3-34: Requisito no funcional Utilizar Kinect 63
Tabla 3-35: Requisito no funcional Utilizar ROS 64
Tabla 3-36: Requisito no funcional Utilizar Ubuntu 64
Tabla 3-37: Requisito no funcional Utilizar micrófono 64
Tabla 3-38: Requisito no funcional Tiempo de retardo 64
Tabla 3-39: Requisito no funcional Diseño modular 65
Tabla 3-40: Requisito no funcional Rápido aprendizaje 65
Tabla 4-1: Resultados del cuestionario 1 118
Tabla 4-2: Resultados del cuestionario 2 119
Tabla 5-1: Tareas del proyecto 122
Tabla 5-2: Desglose presupuestario del personal 125
Tabla 5-3: Desglose presupuestario del material 126
Tabla 5-4: Desglose presupuestario de costes directos 126
Tabla 5-5: Resumen de costes 126

1. INTRODUCCIÓN

El avance de la tecnología a lo largo de la historia ha permitido a la humanidad mejorar su calidad de vida. Uno de los avances que más repercusión ha tenido en las últimas décadas ha sido el nacimiento de la informática y unida a ella, la evolución de la robótica. La relación de las personas con la informática ha cambiado mucho desde su nacimiento a la actualidad, pasando del ámbito militar y académico a ser parte indispensable en la vida diaria de la población. De esta forma se ha llegado al momento actual, en el que existe un ordenador (o varios) en cada casa y un *smartphone* en cada bolsillo, el siguiente paso que se puede prever es una gran expansión en el uso de robots en todos los ámbitos de la vida [1].

El objetivo que tienen la mayoría de los robots es la de ayudar o sustituir a un humano en la realización de ciertas actividades, que debido a sus características no son adecuadas para ser realizadas por personas, bien porque son demasiado complicadas o peligrosas, o bien porque un robot puede realizarlas mejor. En un principio, la robótica se utilizó principalmente en la industria, aplicándose con éxito a las cadenas de montaje, lo que favoreció su expansión hacia otros ámbitos. No ha sido hasta los últimos años cuando se viene haciendo mayor hincapié en los robots de servicio [2]. Estos intentan ampliar las funcionalidades de los descritos anteriormente, de manera que no se limiten a sustituir a las personas en las líneas de producción, si no que sean capaces de realizar tareas muy diversas y que puedan estar presentes en todos los sectores de la sociedad: agricultura, minería, sanidad, educación, etcétera. Dentro de este tipo de robots podemos encontrarnos a los robots humanoides, robots cuya fisionomía y forma de interactuar con el medio intenta imitar a la de las personas y que están teniendo un gran desarrollo en la actualidad. La mayor ventaja de los robots humanoides es que al estar construidos imitando nuestra fisionomía serán capaces de interactuar en los entornos humanos con mayor facilidad: podrán abrir puertas, subir escaleras e, incluso manejar herramientas. Esta ventaja los convierte en los más aptos para remplazarnos en la realización de determinadas labores.

En la actualidad la mayor parte de los robots existentes son industriales, fuera de este ámbito son los robots teleoperados frente a los autónomos los que son utilizados en un mayor número de actividades, debido a que la inteligencia artificial todavía no está lo suficientemente avanzada como para llevar a cabo operaciones complejas [3]. Los robots teleoperados son aquellos que son manejados por un usuario a distancia desde una estación remota. En los casos en que el trabajo que se debe realizar se encuentra en un entorno peligroso, los robots teleoperados permiten mantener a salvo a la persona que maneja el robot, ya que esta se puede encontrar muy alejada del lugar donde está el robot. Algunos ejemplos de este tipo de robot son los usados por los artificieros para desactivar bombas [4], los vehículos espaciales enviados a la Luna [5] o a Marte [6] o, más recientemente, los robots utilizados en la catástrofe de la central nuclear de Fukushima tras el terremoto de Japón [7].

Normalmente, las estaciones de teleoperación suelen estar compuestas por un ordenador al que se le conectan distintos dispositivos de entrada y salida. Los dispositivos de entrada más comunes son el teclado, el ratón y joysticks especialmente preparados para el manejo del robot. El dispositivo de salida por excelencia es el monitor, donde mediante una interfaz gráfica el operario recibe toda la información relevante para el manejo del robot. Esta información puede abarcar desde imágenes que el robot este tomando con una cámara, datos sobre la posición de las articulaciones del robot, indicaciones de los distintos sensores, etc. Este tipo de teleoperación es muy útil para robots móviles sencillos que dispongan de un solo brazo articulado para realizar operaciones [3] [8] [9]. Cuando el robot es más complejo, con distintas articulaciones que se mueven independientemente, con la posibilidad de realizar tareas y movimientos variados, este sistema de control presenta limitaciones a la hora de que el operario pueda apreciar de un modo ágil e intuitivo el estado en el que se encuentra el robot teleoperado.

En este proyecto, se desarrollará un sistema de teleoperación mediante una interfaz natural de usuario para manejar un robot humanoide NAO. Una interfaz natural de usuario suministra una manera de interactuar con un sistema o aplicación mucho más instintiva, en la que el usuario utiliza sus manos, su cuerpo y su voz para transmitir las órdenes, en lugar de periféricos de entrada como el ratón o el teclado. Gracias a estas características, se consigue que personas sin conocimientos informáticos sean capaces de manejar todo tipo de aplicaciones. Para crear dicha interfaz se utilizará el dispositivo Kinect desarrollado por Microsoft.

1.1. OBJETIVO

El objetivo que se persigue con la realización de este proyecto consiste en la creación de una interfaz natural de usuario para la teleoperación de un robot humanoide. Para el desarrollo de la interfaz natural se utilizará el dispositivo Kinect creado por Microsoft y el robot humanoide NAO construido por la empresa Aldebaran Robotics. Para llevar a cabo la lógica de control y la conexión entre los distintos componentes se utilizará un ordenador con Ubuntu donde se ejecutará ROS (*Robot Operating System*) un *framework* para la construcción de software relacionado con la robótica y que se explica con detalle en el anexo A. ROS.

Mediante la interfaz natural el robot imitará los movimientos que el operario realice con los brazos, andará o girará en función de la posición que tome la persona y obedecerá órdenes sencillas para iniciar la teleoperación, detenerla o abrir y cerrar las manos. Además el operario dispondrá de una pantalla donde podrá observar lo que ve el robot a través de una de sus cámaras y recibirá información sobre el estado de la teleoperación; tanto por voz, si es capaz de escuchar al robot desde el lugar de teleoperación, como mediante mensajes que aparecerán en pantalla. Se busca que cualquier persona con un periodo de aprendizaje mínimo sea capaz de controlar al robot con soltura, pudiendo mover al robot en un entorno desconocido y realizar tareas sencillas en él.

Para lograr alcanzar este objetivo general, se ha realizado una división en objetivos más específicos:

- Estudiar el *framework* ROS y familiarizarse con sus características para poder aprovechar sus capacidades y posibilitar la realización de un diseño que tenga en cuenta su manera de funcionar.
- Realizar una investigación entre los paquetes ya existentes y publicados para ROS, con el objetivo de reutilizar código ya existente. Principalmente aquellos que estén relacionados con el sensor Kinect.
- Estudiar el funcionamiento del robot Nao, como se conecta con un ordenador, cuales son sus características físicas, como se mueven sus

articulaciones y que comportamientos básicos tiene. Examinar los módulos existentes que permiten controlar al robot Nao.

- Diseñar a alto nivel un sistema de control que sea altamente modular y, en la que cada uno de los módulos sea reutilizable.
- Diseñar como será el procedimiento de teleoperación teniendo en cuenta que se precisa una alta usabilidad y una curva de aprendizaje muy corta y poco pronunciada.
- Llevar a cabo la implementación del sistema de teleoperación, realizando las pruebas necesarias hasta que el resultado sea satisfactorio. También el código deberá estar abundantemente comentado para facilitar futuras modificaciones.
- Realizar una serie de experimentos en los que participen distintos usuarios con el objetivo de evaluar el funcionamiento final del sistema. Analizar los resultados para obtener conclusiones e ideas sobre posibles mejoras del proyecto a realizar en un futuro.
- Plasmar en la documentación los aspectos más relevantes para entender el desarrollo y funcionamiento del proyecto.

1.2. ESTRUCTURA DEL DOCUMENTO

A continuación se detalla la estructura del documento, ofreciendo un resumen de cada una de las secciones de las que se compone:

1. **Introducción**

En este apartado, tras una breve introducción, se presentan las motivaciones que han llevado al desarrollo de este proyecto, enunciando porque se considera relevante y necesario. También se plantean los objetivos que debe cumplir el proyecto para ser realizado con éxito.

2. **Estado de la cuestión**

En el estado de la cuestión se busca situar al proyecto en el entorno tecnológico en el que se desarrolla. Se comienza con un resumen de la historia de la robótica, para luego centrarse en las características del robot NAO. A continuación, se realiza una descripción de los fundamentos de la teleoperación, su historia, sus elementos y los campos donde se aplica. Finalmente, se presenta el sensor Kinect, sus orígenes y sus características técnicas.

3. **Diseño e implementación del sistema**

En este apartado se describe el sistema de control. En primer lugar se explica la metodología utilizada y posteriormente, el análisis del sistema junto con el alcance del sistema, un catálogo de requisitos y casos de uso. La última sección le corresponde al diseño, donde se determina la arquitectura utilizada y se especifica el funcionamiento de los distintos componentes que forman el sistema.

4. **Evaluación**

Durante la evaluación se pondrá a prueba el sistema con diferentes entornos y usuarios para comprobar si funciona correctamente y hasta que punto se han cumplido con éxito algunos de los objetivos del proyecto. Por lo tanto, en esta apartado se detallará en que han consistido estas pruebas y cuales han sido sus resultados.

5. **Gestión del proyecto**

En este apartado se muestra la distribución temporal en las distintas fases de desarrollo del proyecto, tanto la planificación realizada al inicio, como el tiempo real que ha llevado su realización. También aparece el presupuesto, donde se representan los costes asociados al proyecto.

6. **Conclusiones**

En este último punto se presentan las conclusiones obtenidas tras la finalización del proyecto y las posibles mejoras que se podrían implementar en un futuro.

Anexos

En los anexos se encuentran todos aquellos contenidos que se consideran interesantes o necesarios para una completa comprensión del proyecto. Entre estos contenidos se pueden encontrar la preparación del entorno de ejecución, el manual de usuario, la descripción detallada de los mensajes utilizados, el glosario de términos, etc.

.

2. ESTADO DE LA CUESTIÓN

A continuación se presenta una aproximación al contexto en el que se realiza este proyecto, para ayudar a comprender mejor cual es su motivación y su situación con respecto al estado de la robótica y la teleoperación. También, se hablará sobre los antecedentes y características de Kinect, ya que es una parte fundamental del proyecto.

2.1. HISTORIA DE LA ROBÓTICA

Son muchos los relatos desde el principio de la historia en los que aparecen autómatas artificiales similares a humanos, los cuales obedecen órdenes y se comportan como nosotros. Este tipo de relatos han surgido en muy distintas culturas, lo que demuestra que este es uno de los mayores anhelos de la humanidad. En la cultura griega son muchos los ejemplos de este tipo de historias, como es el caso de Talos, un autómata gigante de bronce que protegía la isla de Creta. No obstante, no fue hasta 1921 cuando surgió el termino *robot* apareciendo por primera vez en la obra *Rossum's Universal Robots* del autor Karel Capek [10]. Atendiendo a su etimología, robot proviene de la palabra checa *robota*, que significa labor forzada o servidumbre. En esta obra, la palabra robot se utiliza para designar a máquinas orgánicas de forma humanoide que trabajan para los seres humanos. A partir de aquí, el uso del término robot ha evolucionado hasta designar a todas aquellas máquinas capaces de realizar trabajos autónomamente o mediante teleoperación.

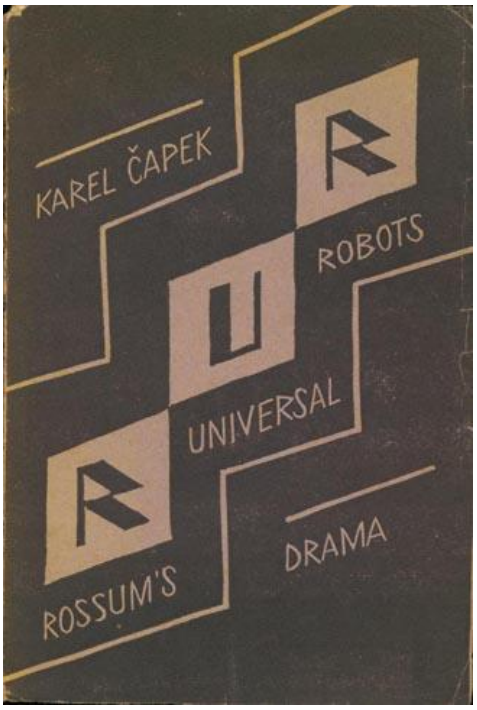


FIGURA 2-1: PORTADA DE LA PRIMERA EDICIÓN DE ROSSUM'S UNIVERSAL ROBOTS

Los primeros autómatas de los que se tiene constancia nacieron en la Grecia Clásica, como por ejemplo: "la paloma" de Arquitas de Tarento, un pájaro mecánico que funcionaba con vapor, o un reloj de agua o clepsidra inventado por Ctesibio que tenía figuras móviles. Más adelante, Herón de Alejandría escribió en el año 62 d.C un libro titulado "Autómata", donde muestra diseño de juguetes capaces de moverse por si solos de forma repetida y de imitar el comportamiento de animales, como las aves. Durante el Imperio Romano, también existía afición por los autómatas como se ve en "El banquete de Trimalco" donde se describe un frutero con un Príapo que arrojaba perfume al ejercer una cierta presión. En Oriente también se produjeron avances en la construcción de estos mecanismos: Huang Kun construyó figuras animales y humanas capaces de cantar y danzar, y Yang Wu-Lien fabricó un mono capaz de pedir limosna y guardar las monedas al llegar a un cierto peso. En el siglo XII Al-Jazari inventó un barco que contenía una banda de música formada por músicos mecánicos que incluían tamborileros, un arpista y un flautista, y que eran accionados con la fuerza del agua.

El autómata más antiguo que se conserva es el Gallo de Estrasburgo, que funcionó de 1352 hasta 1789, y que movía las alas y el pico cuando el reloj de la catedral donde se encuentra daba las horas. En España se conserva funcionando el Papamoscas del siglo XVI, un autómata que se encuentra en la catedral de Burgos y que consiste en un hombre mecánico cuyo movimiento también está ligado al reloj de la catedral. Otros autómatas de la Edad Media y del Renacimiento fueron: el hombre de hierro de *Alberto Magno*, la cabeza parlante de *Roger* Bacon, el león mecánico construido por Leonardo Da Vinci en 1515 en honor al rey de Francia o un pato capaz de agitar las alas e imitar el proceso digestivo que construyó Jacques de Vaucanson. Existen otros muchos ejemplos de autómatas de este tipo, la mayoría artilugios mecánicos que imitan a persona o animales, y que no tienen otra finalidad que la de entretener y a los que todavía no se les ha encontrado una actividad productiva.

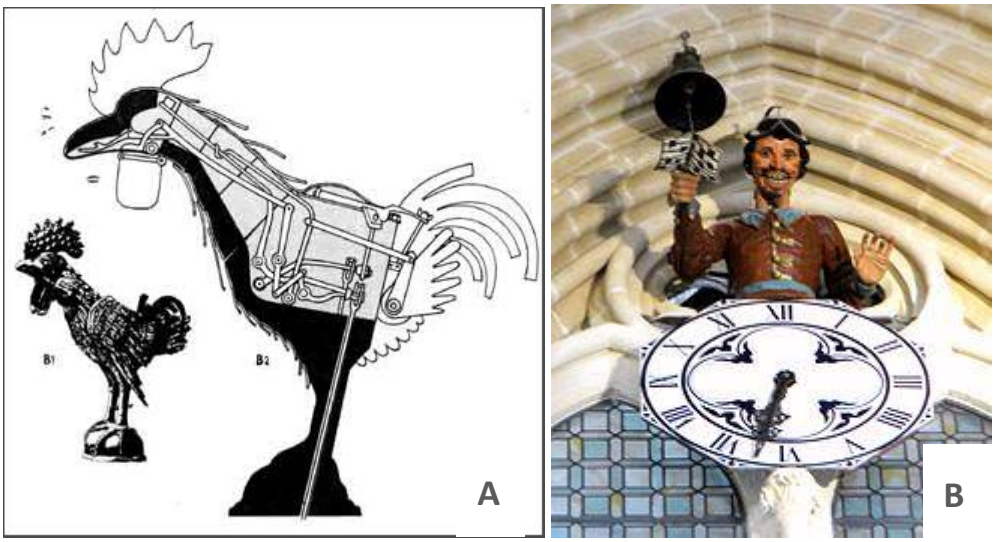


FIGURA 2-2: (A) GALLO DE ESTRASBURGO Y (B) PAPAMOSCAS DE BURGOS

No fue hasta el siglo XVIII, cuando un mayor avance del sector industrial propiciado por el nacimiento de la máquina de vapor, dio lugar al nacimiento de máquinas capaces de realizar tareas de forma automática para ayudar en los procesos de producción, como por ejemplo: la hiladora giratoria construida por Hargreaves en 1770 o el telar mecánico desarrollado por Cartwright en 1785. En 1801 Joseph Marie Jacquard construyó un sistema de telares que permitía programar su funcionamiento automático mediante tarjetas perforadas, idea que fue incorporada por IBM en sus primeras computadoras.

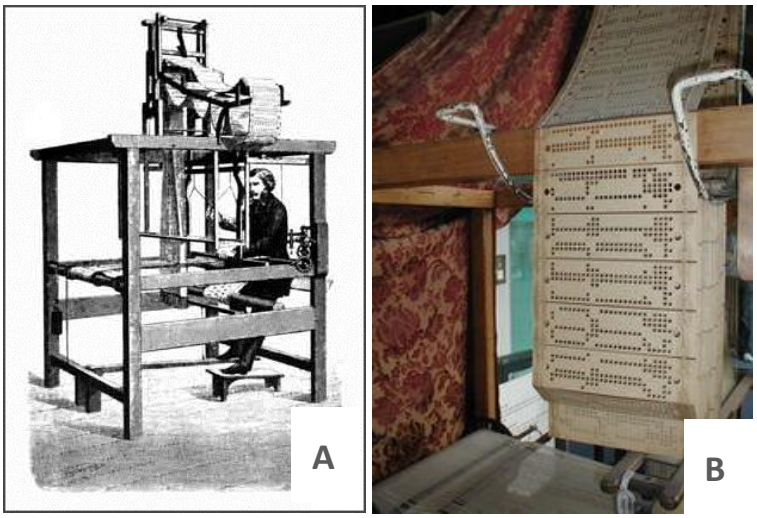


FIGURA 2-3: (A) TELAR DE JOSEPH MARIE JACQUARD Y (B) TARJETA PERFORADA

El término robótica fue utilizado por primera vez por Isaac Asimov en su obra "*Yo robot*" de 1942 y estableció las famosas tres leyes de la robótica en el relato "*Runaround*" [11], coincidiendo con el apogeo de la robótica moderna. En estos años empezaron a aparecer los primeros robots industriales. El auge de este tipo de robots, debido a que demostraron ser capaces de aumentar la productividad y ser altamente rentables, provocó un aumento de la investigación y del desarrollo en la robótica en general. Muchos investigadores empezaron a trabajar en la búsqueda de robots cada vez más flexibles, robustos, veloces y sobre todo, autónomos. Uno de estos robots fue ELSIE (*Electro-Light-Sensitive Internal-External*) el primer robot móvil autónomo de la historia, que fue creado en Inglaterra en 1953. A finales de la década de los 60, fue construido en el Standford Research Institute (SRI) el robot móvil Shakey, provisto de sensores táctiles y una cámara de video, y que era capaz de reconocer objetos y planificar acciones.

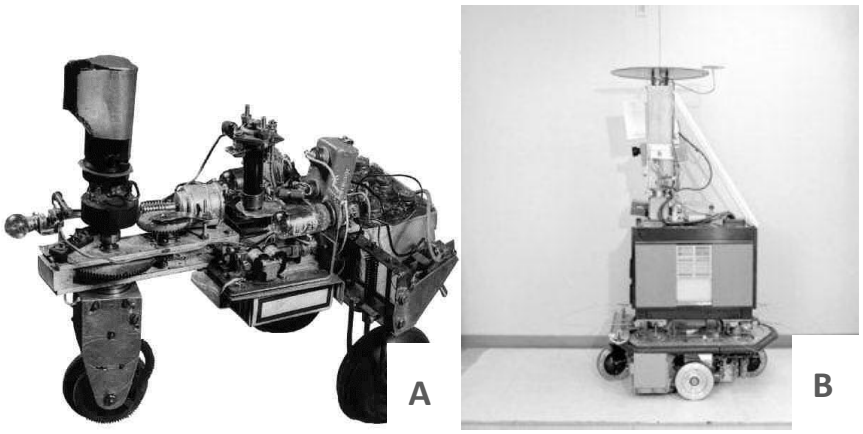


FIGURA 2-4: (A) ELSIE Y (B) SHAKEY

En la segunda mitad del siglo XX los avances en electrónica e informática dieron lugar a una verdadera explosión de la robótica. En este periodo de tiempo nacieron una gran cantidad de grupos de investigación y empresas que desarrollaron una ingente cantidad de robots, de una gran variedad en lo que se refiere a formas, tamaños, utilidades, costes, etcétera. Algunos de los desarrollos más importantes de esta época son:

- En 1970 el SRI construye el primer manipulador con motores eléctricos y la URSS consigue enviar a la luna un vehículo robotizado manejado por control remoto.
- En los setenta la NASA inicia un programa de cooperación con el *Jet Propulsión Laboratory* (JPL) para desarrollar plataformas con posibilidad de exploración de terrenos agrestes, produciendo el *Mars-Rover*.
- En 1975 se comienza a usar microprocesadores en robots permitiendo nuevos tamaños y reduciendo su coste.
- En 1978 el robot PUMA comienza a trabajar en la línea de montaje de General Motors.
- En 1986 Honda inicia un proyecto de I+A con la consigna de construir un robot capaz de coexistir y coopera con los seres humano para beneficiar a la sociedad.
- En 1989 R. Brooks y AM Flynn del MIT publican un artículo en el que apuestan por pasar de construir robots grandes, sofisticados y caros, a

crear robots pequeños, simples y a menor coste. Al mismo tiempo, desarrollan el robot Genghis un hexápodo similar a un insecto.

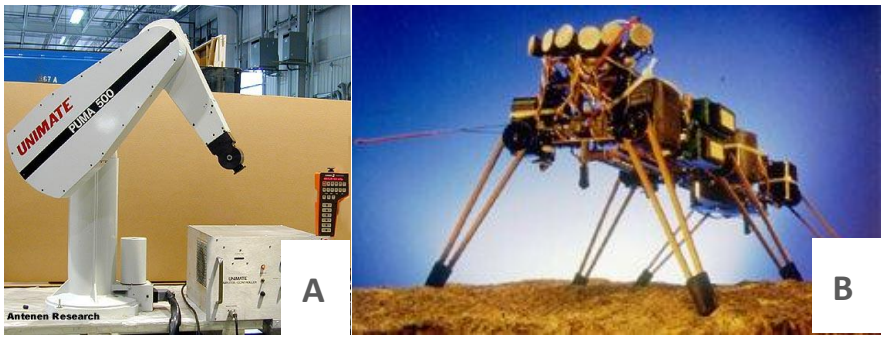


FIGURA 2-5: (A) ROBOT PUMA Y (B) GENGHIS

- En 1998 la robótica da varios pasos para ir entrando en los hogares y se crean el robot *Furby*, que imita a una mascota capaz de comunicarse; el robot *Aibo* de SONY, un robot similar a un perro; y LEGO lanza *Mindstorm*, una manera de acercar la robótica más avanzada a cualquier persona.

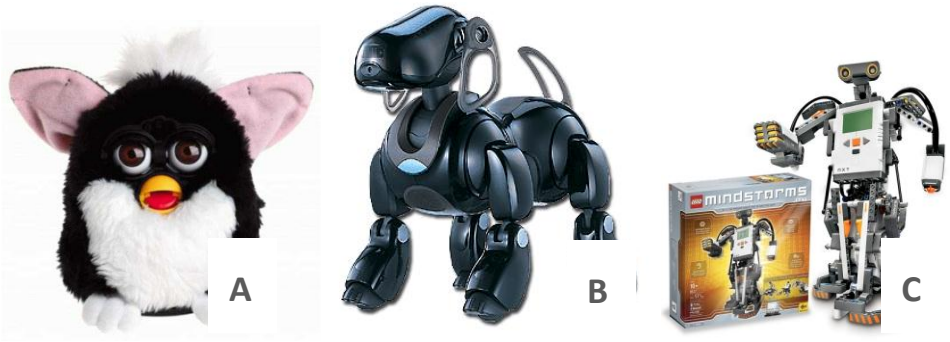


FIGURA 2-6; (A) FURBY, (B) AIBO Y (C) LEGO MINDSTORM

- En el año 2000, HONDA presenta ASIMO un robot humanoide capaz de caminar, correr, subir y bajar escaleras y de realizar otras muchas acciones que imitan a las que puede hacer un ser humano.
- En 2003, la NASA envía a Marte el robot *Spirit* y el *Opportunity*.
- En 2005, *Aldebaran Robotics* comenzó el proyecto NAO, que culminó con la creación del robot humanoide NAO unos años después.
- En 2010, *Aldebaran Robotics* presenta el proyecto Romeo, con la intención de crear un hermano mayor de NAO, es decir, un robot humanoide mucho

más avanzado que sea capaz de ayudar a personas con perdida de autonomía.

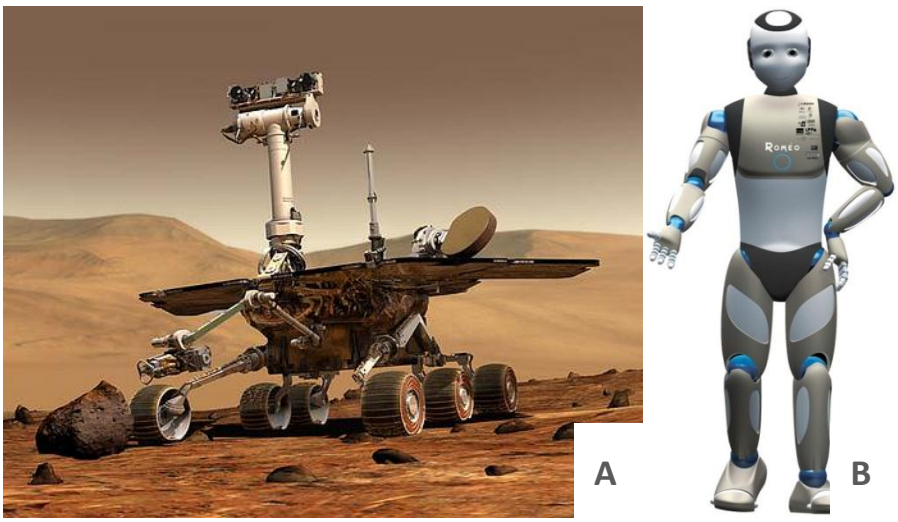


FIGURA 2-7: (A) ROBOT SPIRIT Y (B) ROBOT ROMEO

2.1.1. ROBOT NAO

El robot NAO es un robot humanoide creado por *Aldebaran Robotics*, una empresa francesa con sede en París que nació en el año 2005 [12]. *Aldebaran Robotics* fue fundada con la idea de crear robots humanoides que pudieran asistir a las personas. NAO fue creado con este objetivo,2 ofrecer una plataforma hardware y software que permitiera un avance en las investigaciones en este ámbito, a un precio razonable.

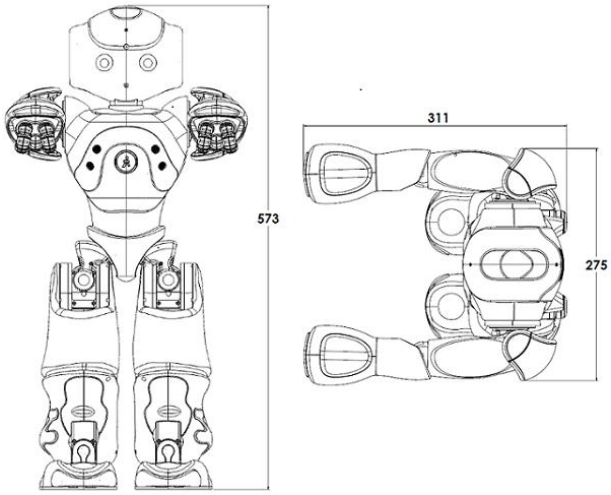


FIGURA 2-8: DIMENSIONES DE NAO

Nao mide aproximadamente 58 cm de altura y pesa unos 4,8 Kg. Dispone de una batería de ion de litio que le permite una autonomía de unos 90 minutos y funciona con un procesador "*x86 AMD GEODE 500MHz CPU*" con 256 MB de memoria SDRAM y 2 GB de memoria flash. Cuenta con 26 articulaciones, que se distribuyen de la siguiente manera:

- 2 grados de libertad en la cabeza.
- 5 grados de libertad en cada brazo.
- 2 grados de libertad en cada mano.
- 5 grados de libertad en cada pierna.
- 1 grados de libertad en la pelvis.

En la Figura 2-9 se indican las articulaciones del lado izquierdo del robot.

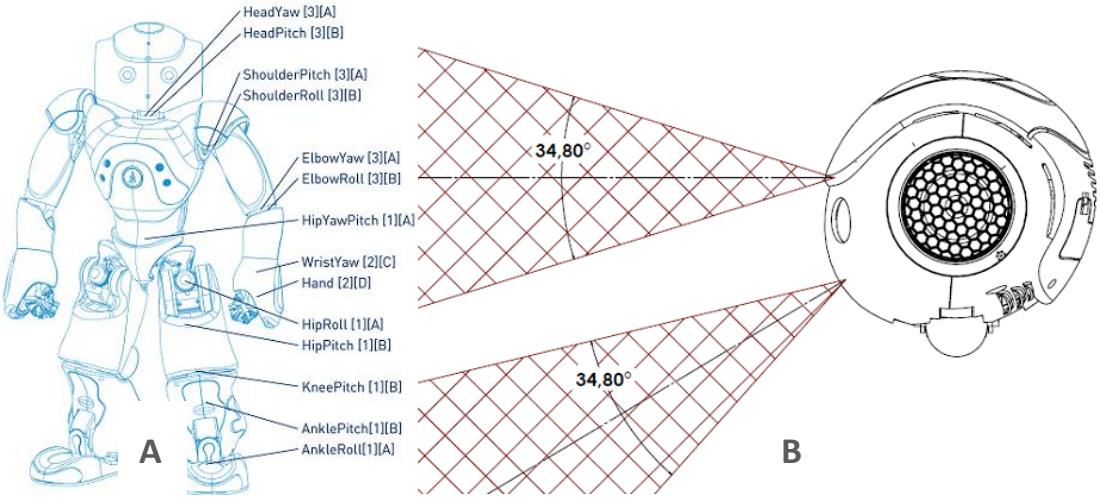


FIGURA 2-9: (A) ARTICULACIONES EN DETALLE DEL NAO Y (B) POSICIÓN DE LAS CÁMARAS

Cuenta con 4 micrófonos en la cabeza, uno a cada lado, uno en la parte de adelante y otro en la parte de atrás; y también dos altavoces uno a cada lado de la cabeza. Para la visión posee dos cámaras una que le permite mirar hacia el frente y otra para ver la parte del mundo que tiene más cercana y que esta inclinada hacia abajo, como se puede ver en la Figura 2-9. Presenta varios sensores de presión en todo el cuerpo: un botón en el pecho, 2 botones tipo bumper en cada pie, 3 sensores táctiles en la cabeza y otros 3 sensores táctiles en cada mano. Cuenta con 8 FSRs (*force-sensing resistor*), sensores que miden los cambios de resistencia debidos a la fuerza ejercida en un punto, que se encuentran en los pies y son utilizados para que el robot mantenga el equilibrio. Además tiene 2 girómetros de un eje, un acelerómetro de 3 ejes y 2 sónares.

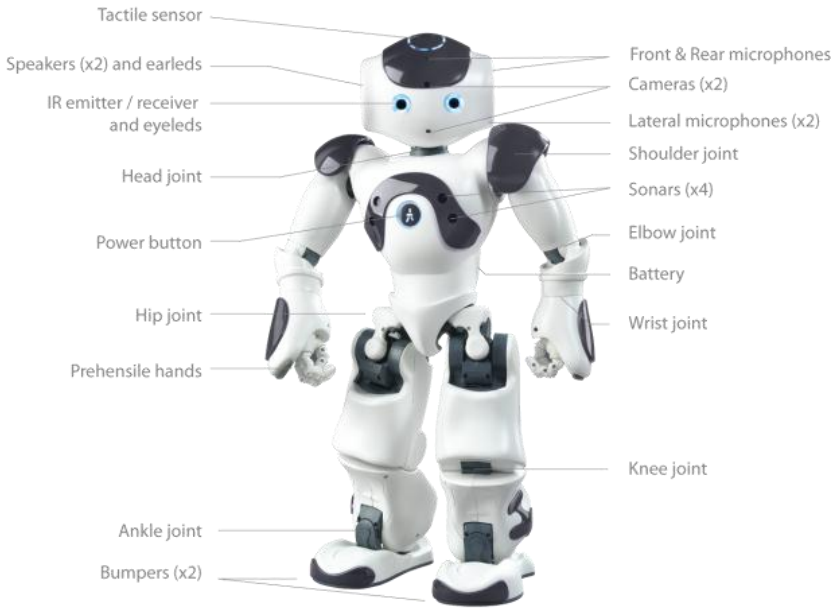


FIGURA 2-10: RESUMEN DE LOS COMPONENTES DE NAO

Con estas características el 15 de agosto de 2007, NAO sustituyó al perro *Aibot* de Sony como robot que competiría en la *RoboCup* (*Robot Soccer World Cup*), una competición internacional en la que los robots juegan al fútbol. Desde su nacimiento son muchas las aplicaciones que distintas entidades han desarrollado para el robot NAO: aparte de jugar al futbol, es capaz de reconocer objetos, caras o voces; colaborar con otros robots NAO para cargar objetos; obedecer ordenes; escribir; realizar coreografías en grupo; tocar un xilófono; ayudar en la cocina, y otras muchas cosas. Como se puede ver, a pesar de sus limitaciones, las posibilidades del robot NAO son muy grandes.

FIGURA 2-11: PARTIDO DE LA ROBOCUP

A finales del 2011, *Aldebaran Robotics* sacó al mercado una nueva generación de robots NAO con un procesador mejorado (*Intel Atom* a 1.6GHz) y cámaras de alta definición que permiten un reconocimiento de objetos y rostros más fiable. Además también incluía nuevos algoritmos para reconocimiento de voz y una nueva forma de andar y de actuar ante las colisiones.

2.2. TELEOPERACIÓN

Desde el principio de los tiempos los hombres han buscado maneras de poder aumentar su capacidad de actuación sobre el entorno. Al principio eran simples palos utilizados para llegar a lugares de otro modo inaccesible, a lo que siguieron otras herramientas como las pinzas de herrero para manejar materiales incandescentes o las tijeras para facilitar el corte de distintos elementos. Este desarrolló de herramientas que nos han permitido aumentar las acciones que podemos llevar a cabo, ha terminado desembocando en la teleoperación tal como hoy la conocemos. Está se puede definir como el manejo a distancia de un dispositivo que posibilita o facilita la realización de una actividad que de otro modo sería irrealizable o muy difícil para un ser humano [3] [9] [8].

2.2.1. HISTORIA

La teleoperación nació junto con la industria nuclear debido a la necesidad de manejar materiales altamente radioactivos, muy peligrosos para la salud simplemente con estar en su presencia.

En 1947 Raymond Goertz del *Argonne National Laboratory* en Estados Unidos, comenzó las investigaciones en este ámbito, con el objetivo de desarrollar un manipulador que un operador pudiera manejar a distancia. El primer fruto de esta investigación vino un año después y consistía en un sistema que hacía que un manipulador (esclavo) imitará los movimientos que un operador hacía con su brazo, que estaba unido al manipulador maestro. Este primer sistema de teleoperación se denominó M1 y era enteramente mecánico, de tal manera que los movimientos realizados en el maestro se transmitían eje a eje al manipulador esclavo.

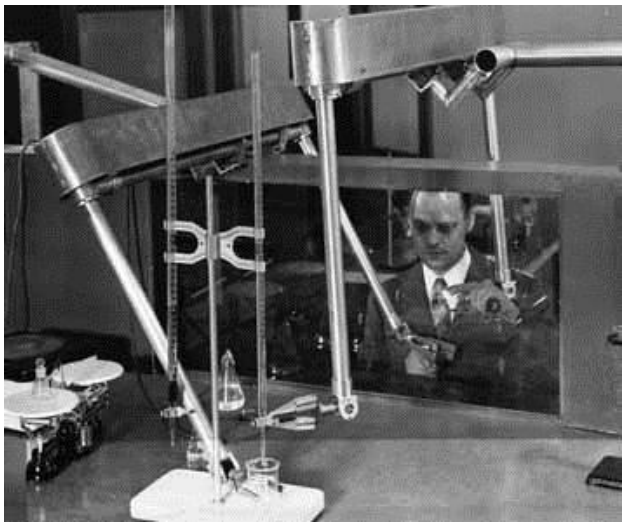


FIGURA 2-12: GOERTZ TRABAJANDO CON EL M1

En los cincuenta, Goertz siguió con sus investigaciones para pasar de un sistema mecánico a otro accionado por motores. En 1954 creó el E1, el primer sistema de teleoperación maestro-esclavo accionado por electricidad y con servo controles en ambos manipuladores.

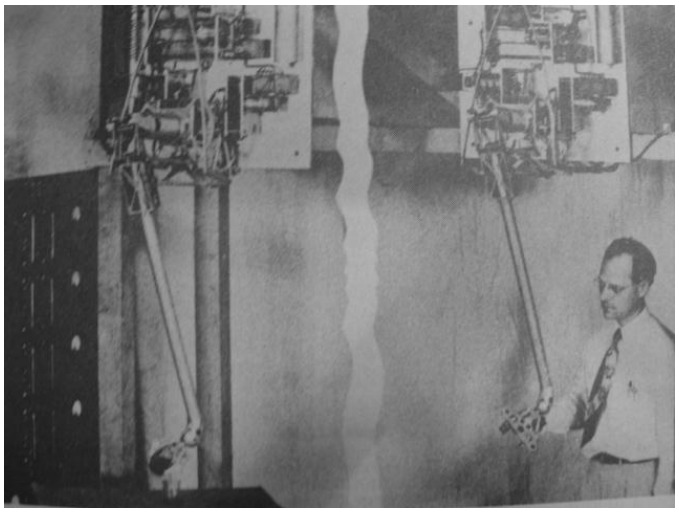


FIGURA 2-13: GOERTZ TRABAJANDO CON EL E1

En la década de los sesenta, se produjeron avances en la teleoperación de máquinas submarinas, con sistemas cada vez más avanzados debido a la inclusión de cámaras y otros dispositivos que mejoraban la sensación de telepresencia. En los años setenta, la teleoperación alcanzó su madurez con sus aplicaciones en las misiones espaciales, por ejemplo, los vehículos a control remoto *Lunojod* 1 y *Lunojod* II que fueron enviados a la Luna en 1970 y 1973, respectivamente.

En las últimas décadas, los avances en teleoperación han ido muy ligados a la evolución de la robótica y la informática. Gracias a estos avances, muchos sistemas de teleoperación actuales consisten en robots que tienen una gran autonomía y solo precisan ser teleoperados para determinadas acciones que, debido a las limitaciones de la robótica, no puede realizar por si solos. También se ha progresado en las interfaces hombre-máquina buscando una mayor sensación de control de la máquina y de telepresencia.

2.2.2. Elementos de un sistema de teleoperación

Los elementos comunes a todo sistema de teleoperación son los siguientes:

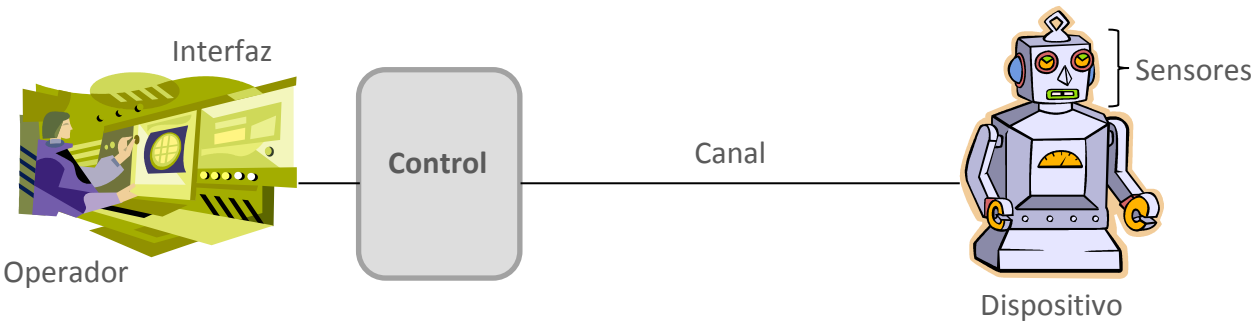


Figura 2-14: Elementos de un sistema de teleoperación

- Operador: es el ser humano encargado de llevar a cabo el control a distancia de la operación. Este control puede ser total, donde el operador maneja todas las posibles acciones del dispositivo teleoperado, o puede reducirse a marcar planes y objetivos que el dispositivo debe llevar a cabo.
- Dispositivo teleoperado: es el artefacto que actúa en el lugar remoto y que es controlado por el operador. El dispositivo se puede corresponder con un vehículo, un manipulador, un robot o cualquier otro artefacto similar. No obstante, a partir de ahora se utilizará la palabra robot o dispositivo teleoperador indistintamente.
- Interfaz: es el medio a través del cual el operador se comunica con el sistema de operación. Se refiere tanto a los instrumentos que permiten que el operador lleve a cabo el control, como a los componentes que ofrecen al operador información sobre el estado de la teleoperación, es

decir, se refiere tanto a un joystick como al monitor que presenta las imágenes que dispositivo teleoperado está tomando con una cámara.

- Canales de comunicación: es el medio a través del cual se transmite la información del operador al dispositivo teleoperado y viceversa. Se puede realizar mediante cables, inalámbricamente o mediante canales mixtos.
- Control: es el modulo que se encarga de procesar las señales que se envían entre el operador y el dispositivo para adaptarlas a sus necesidades. Por ejemplo, se encarga de transformar un movimiento del joystick en una acción efectiva que realice el dispositivo teleoperado.
- Sensores: son los elementos encargados de recoger información para ser utilizada por la interfaz y el control.

La interfaz es el elemento más importantes y característico dentro de un sistema de teleoperación y merece una explicación más detallada. Además, está íntimamente relacionada con el tipo de control, ya que la utilización de un tipo de interfaz implica un tipo de control, y viceversa.

2.2.2.1. INTERFAZ

La interfaz de un sistema de teleoperación esta formada por dispositivos de control y de realimentación. Los primeros son los que utiliza el operador para generar los distintos comandos, y los segundos tienen la finalidad de transmitir información al operador sobre el estado del dispositivo teleoperado y del entorno remoto. También existen dispositivos bilaterales, capaces de realizar las dos funciones al mismo tiempo.

2.2.2.1.1. DISPOSITIVOS DE CONTROL

Dentro de los dispositivos de control se pueden diferenciar dos tipos:

- Dispositivos que generan comandos de bajo nivel.

Los comandos de bajo nivel están relacionados con el movimiento de las articulaciones del robot. Este tipo de control genera las referencias para controlar

directamente los servos del robot. Algunos dispositivos que generan comandos de bajo nivel son:

- Joysticks: consiste en una palanca de control con dos o tres ejes y que es muy útil para la teleoperación de vehículos.

FIGURA 2-15: JOYSTICK

- Maestros: son dispositivos similares al dispositivo teleoperado o esclavo, como se le denomina en este caso. El operador actúa directamente sobre el maestro, guiando sus articulaciones y el esclavo imitará sus movimientos.

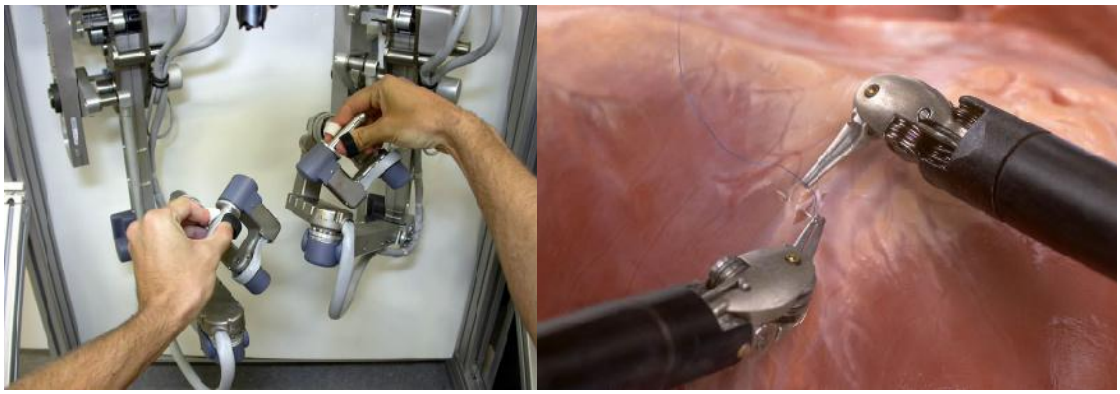


FIGURA 2-16: TELEOPERACIÓN MAESTRO-ESCLAVO EN CIRUGÍA.

- Interfaces corporales: se basan en detectar la posición del cuerpo del operador y utilizar dicha información para mover al robot. Normalmente, se utiliza para sistemas de teleoperación de robots humanoides, en los que el cuerpo del operador funcionaría como maestro y el robot, como esclavo. Para obtener la postura del teleoperador, se pueden utilizar exoesqueletos, trajes de captura de movimiento o técnicas de visión artificial.

FIGURA 2-17: TELEOPERACIÓN MEDIANTE EXOESQUELETO

- Dispositivos que generan comandos de alto nivel.

  Los comandos de alto nivel se refieren a tareas a realizar. En lugar de controlar en cada momento la posición de las articulaciones del robot, se le envían órdenes más complejas que es capaz de realizar autónomamente. Ejemplos de este tipo de orden serían “Ir hasta la posición X”, “Coger el objeto rojo”, “Sentarse”, “Bailar”, etc. Esta clase de control es muy usado en sistemas en los que el retardo es muy grande, como las misiones espaciales. Los dispositivos más comunes para generar comandos de alto comandos de alto nivel son:

  - Interfaces visuales: estas interfaces se pueden controlar con teclado y ratón desde un computador, o se pueden utilizar en móviles o tablets. Suelen consistir en paneles de botones y campos de texto en los que insertar parámetros.
  - Reconocimiento de voz: en este caso el operador contará con un micrófono y le comunicará las órdenes al robot mediante su voz.

2.2.2.1.2. DISPOSITIVOS DE REALIMENTACIÓN

Los dispositivos de realimentación se encargan de excitar los sentidos del teleoperador para que conozca el estado de la tarea teleoperada. Los sentidos que se utilizan habitualmente en teleoperación son:

- Vista:

  Las imágenes de video son el método habitual de realizar la realimentación visual del entorno remoto. Estás pueden venir tanto de cámaras que formen parte del robot, como de cámaras fijas situadas en el entorno. Se pueden usar cámaras estereoscópicas para transmitir al operador sensación de profundidad, y mejorar su percepción del entorno y la tarea que esta llevando a cabo.

- Oído:

  La realimentación auditiva no suele ser muy utilizada, y se limita a mensajes procedentes del ordenador o alarmas producidas durante la ejecución de la tarea. Tiene la ventaja de que no necesita la atención continuada del operador, sino que es un método pasivo.

- Tacto:

  También es conocida como realimentación háptica. Se basa en transmitir al operador las fuerzas que se están produciendo en el dispositivo teleoperado. De esta forma, podría conocer cuando una articulación ha chocado con algún elemento del entorno o, incluso, apreciar la forma y textura de un objeto que haya cogido el robot.

2.2.2.1.3. DISPOSITIVOS BILATERALES

Los dispositivos bilaterales incluyen en un único elemento tanto la generación de comandos como la realimentación. Este tipo de dispositivos permite realizar la teleoperación de forma muy efectiva, ya que transmiten al operador la sensación de encontrarse realizando la tarea en el entorno remoto mejor que cualquier otra técnica. La utilización de dispositivos bilaterales, da lugar a lo que en teleoperación se denomina control bilateral.

El ejemplo más representativo de estos dispositivos son los maestros con reflexión de fuerzas. Este tipo de maestros, además de generar las posiciones que debe tomar el esclavo, son capaces de transmitir los esfuerzos que se están desarrollando en el esclavo.

2.2.3. APLICACIONES

La teleoperación ha demostrado ser una herramienta muy útil para permitir al ser humano realizar tareas en zonas lejanas y/o peligrosas, lo que ha servido para ampliar nuestras fronteras y posibilidades. Algunas de las aplicaciones más comunes son:

- **Industria nuclear:**

Como se ha descrito anteriormente la teleoperación nació en la industria nuclear para evitar el contacto directo con sustancias radioactivas. Se utiliza para la experimentación y estudio de estas sustancias, para el mantenimiento de instalaciónes, para tareas de descontaminación e, incluso, en desastres nucleares.

- **Exploración del espacio:**

Debido a que el coste y el riesgo de enviar al espacio máquinas susceptibles de ser teleoperadas es mucho menor al de mandar personas, desde los comienzos de la carrera espacial estos sistemas se han utilizado mucho para explorar el espacio. Estos se enfrentan a grandes retardos en las comunicaciones, que normalmente se han solventado mediante la aplicación del método "mueve y espera".

El primer vehículo teleoperado en la Luna pertenecía a la Unión Soviética, se lanzó en 1970 y se llamaba *Lunojod* 1. Tuvo casi un año de actividad y en ese tiempo recorrió más de 10 Km de la superficie lunar, mandando miles de imágenes del satélite y realizando experimentos sobre el terreno. Unas tres décadas después Estados Unidos repetiría el éxito que tuvo *Lunojod* 1, pero esta vez en Marte, primero con el *rover* Sojourner y, posteriormente, con el *Spirit* y el *Opportunity*. El 6 de agosto de 2012 aterrizó en Marte el *Curiosity*, un *rover* que duplica en tamaño a los anteriormente mandados y cuyo objetivo es estudiar la viabilidad de la vida en Marte.

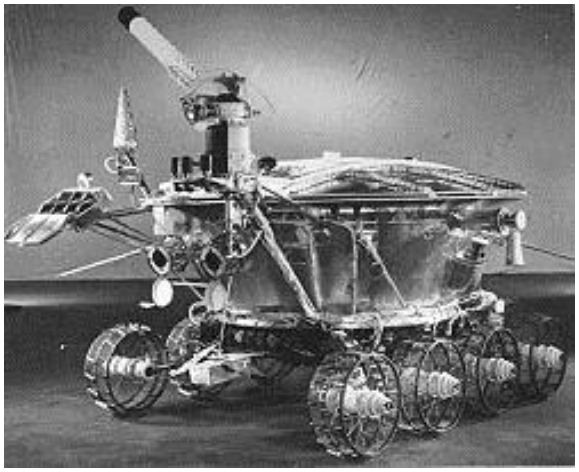


FIGURA 2-18: MAQUETA DEL LUNOJOD 1

- **Ejército:**

El área de la teleoperación tiene una gran atractivo para las aplicaciones militares, debido a que permite llevar a cabo operaciones con muy buenos resultados sin poner en peligro vidas humanas. Un ejemplo de teleoperación con uso militar son los UAV, vehículos aéreos no tripulados que se manejan o programan a distancia y que han demostrado su valía en diferentes escenarios como la Guerra de Bosnia o, más recientemente, la guerra de Afganistán o Irak. Estos sistemas se utilizan para vigilancia, adquisición de objetivos, reconocimiento de zonas peligrosas e, incluso, son capaces de llevar a cabo ataques. Los avances en el campo militar han permitido que se les de nuevos usos en aplicaciones civiles como en cartografía, lucha contra incendios, agricultura, etc.

FIGURA 2-19: MQ-9 REAPER, UN UAV CON CAPACIDAD DE ATAQUE CON MISILES

- **Medicina**

En los últimos años la teleoperación está teniendo un gran impacto en la medicina, sobre todo en el campo de la telecirugía. Esta permite realizar operaciones aunque existan kilómetros de distancia entre el médico y el paciente. Esto se consigue mediante robots cirujanos que reproducen las órdenes que el cirujano le está dando remotamente. Actualmente el robot cirujano más avanzado es el denominado Sistema Quirúrgico Da Vinci, que está capacitado para realizar cualquier operación que pueda ser realizada por laparoscopia.

FIGURA 2-20: SISTEMA QUIRÚRGICO DA VINCI

- **Exploración submarina**

Debido a la imposibilidad que tiene el ser humano para llegar a determinadas profundidades la teleoperación es una de las mejores soluciones para explorar los fondos marinos. En este caso se utilizan vehículos teleoperados conocidos como ROV (*Remote Operated Vehicle*) a los que se les suelen acoplar otra serie de manipuladores teleoperados para llevar a cabo distintas funciones en las profundidades. Este tipo de vehículos pueden llevar a cabo muchas operaciones como: construir y mantener instalaciones submarinas, minería submarina, mantenimiento de puertos y presas, exploración, reflotamientos, etc. Un ejemplo de este tipo de vehículos es el JASON que puede llegar a profundidades de hasta 500 Km.

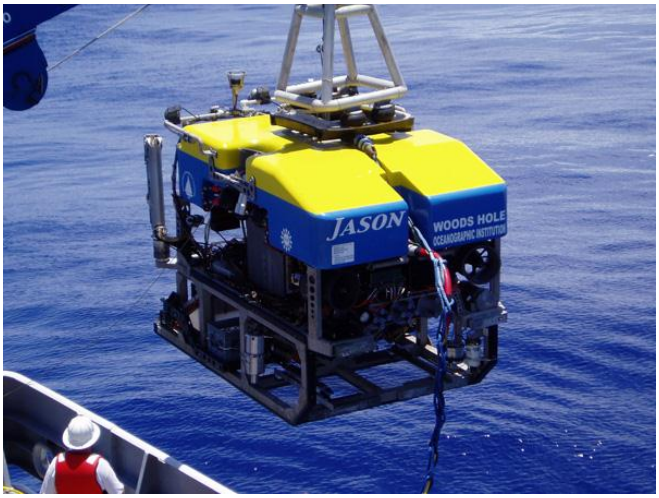


FIGURA 2-21: SISTEMA DE EXPLORACIÓN SUBMARINA JASON

- **Robots humanoides**

Son muchos los ejemplos en los que un se ha utilizado la teleoperación para el manejo de robots humanoides. A continuación, se presentan algunos ejemplos, muy interesantes por su relación con este proyecto:

  - **Telesar V.**

Telesar V (*TELexistence Surrogate Anthropomorphic Robot*) [13], es un robot humanoide capaz de hacer que la persona que lo está controlando sea capaz de sentir, ver y oír lo mismo que si se encontrará en el lugar del robot. Para ello, el operador se tiene que colocar un casco y una serie de sensores que captan el movimiento de su cuerpo. El casco dispone de auriculares y de dos pantallas frente a los ojos, que retransmiten lo que el robot está captando por sus dos cámaras, y permiten que el operador capte la profundidad del entorno. El aspecto donde más destaca *Telesar* V, es en los sensores táctiles de que dispone, que son capaces de transmitir al usuario la textura y temperatura del objeto que está manejando el robot.

FIGURA 2-22: TELESAR V

- **iControlNao.**

Es una aplicación para *iPhone* creada por *Angisoft* para controlar el robot Nao [14]. Está aplicación permite:

- Detectar y conectarse al robot.
- Comenzar y detener distintos comportamientos preprogramados.
- Hacer que el Nao se siente, se levante y ande.
- Insertar una oración para que sea pronunciada por el Nao.

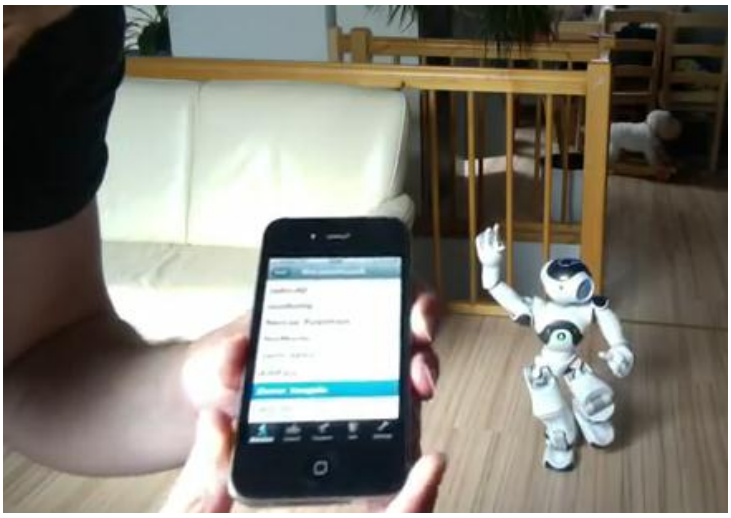


FIGURA 2-23: ICONTROLNAO

- **Teleoperación mediante un traje de captura de movimiento.**

Jonas Koenemann de la Universidad de Friburgo ha desarrollado un sistema de teleoperación, que consiste en un robot Nao que imita los movimientos de una persona ataviada con un traje de captura de movimiento denominado *Xsens MVN* [15]. La característica más importante de este sistema es la capacidad que tiene para mantener estable al robot en todo tipo de postura. Esto se consigue, asegurando que la proyección en el suelo del centro de gravedad del robot coincide con la posición del pie.

FIGURA 2-24: NAO IMITANDO LA POSICIÓN DEL CUERPO DEL OPERADOR

2.3. KINECT

El sensor Kinect, desarrollado por Microsoft, nació como un nuevo controlador para la videoconsola Xbox 360 [16]. Fue creado por Alex Kipman y su innovación reside en que permite manejar la videoconsola sin necesidad de un controlador tradicional o mando, puede controlarla solamente con movimientos de su cuerpo y ordenes de voz. Esto se conoce como interfaz natural de usuario o NUI por sus siglas en inglés, es decir, una manera de interactuar con una máquina sin necesidad de utilizar dispositivos artificiales cuyo funcionamiento debe ser aprendido. De esta manera se consigue que el usuario se comunique con la máquina de un modo mucho más cercano al que le dicta su instinto, favoreciendo una curva de aprendizaje mucho más pequeña y un paso de principiante a experto casi inmediato.

FIGURA 2-25: PERSONAS JUGANDO A LA XBOX 360 CON KINECT

Kinect fue anunciada en el E3 de 2009 con el nombre de *Project Natal*, y no fue hasta un año después, en el E3 de 2010, cuando se dio a conocer su nombre definitivo: Kinect. Finalmente, salió a la venta en Noviembre de 2010 y ha logrado más de 18 millones de unidades vendidas. El mismo mes de su lanzamiento Industrias *Adafruit* anunció una recompensa para la persona que creara un controlador libre y abierto para Kinect. Recompensa que obtuvo el español Hector Martín tras crear un controlador para Linux capaz de obtener información de Kinect y que es el antecesor

del controlador utilizado en este proyecto [17]. Tras esto, y viendo el interés que estaba suscitando el sensor Kinect fuera del ámbito de los videojuegos, Microsoft decidió publicar sus propios controladores para *Windows* en Febrero del 2012 [18].

La tecnología utilizada en Kinect se basa en software creado por *Rare*, una compañía perteneciente a Microsoft, y en una cámara capaz de medir distancias creada por *PrimeSense* [19], una compañía israelí especializada en la creación de interfaces naturales de usuario. Kinect esta formado por tres sensores distintos que trabajan juntos para crear la experiencia de una interfaz natural de usuario, estos son:

- **Cámara RGB**: la denominada por Microsoft como cámara RGB, se trata de una cámara VGA a color, con una resolución de 640x480 pixels, fabricada con un sensor CMOS y capaz de emitir 30 fps [Figura 2-26-1]. Esta cámara se encarga de la captura de video y colabora con el reconocimiento facial.
- **Sensor de profundidad**: el sensor de profundidad esta formado por un emisor de laser infrarrojos, un sensor CMOS monocromo y el módulo encargado de construir la nube de puntos tridimensional. El funcionamiento de este sensor es similar al del sonar de un barco, primero el emisor de infrarrojos proyecta una matriz de puntos de luz infrarroja invisible al ojo humano [Figura 2-26-2]. Cada uno de estos rayos de luz viaja por el aire hasta rebotar con algún objeto, para posteriormente ser capturado por el sensor CMOS. Los rayos de luz infrarroja van perdiendo intensidad cuanto más distancia han recorrido, por lo tanto, una vez que el sensor CMOS ha capturado toda la imagen, se puede calcular la distancia a la que cada rayo de luz ha rebotado a partir de su intensidad. [Figura 2-26-3]. Una de las ventajas de este sistema, es que al utilizar luz infrarroja funciona bien bajo prácticamente cualquier condición lumínica. [19] [20] [21]

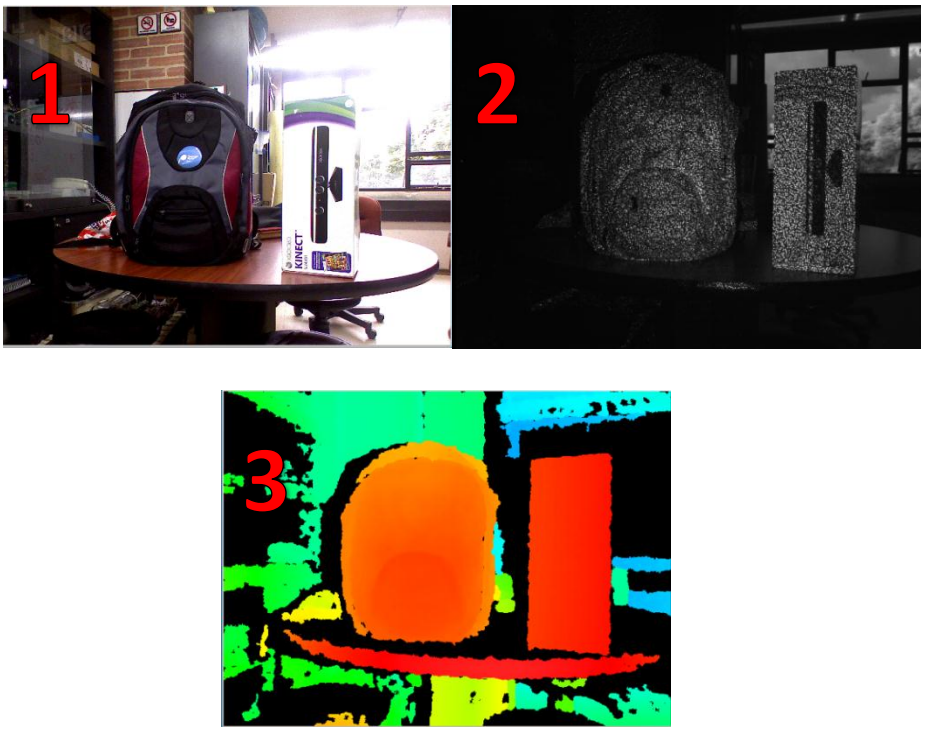


FIGURA 2-26: EJEMPLO DEL FUNCIONAMIENTO DEL SENSOR KINECT

- **Micrófono multiarray**: un micrófono multiarray consiste en realidad en varios micrófonos colocados en fila sobre una superficie y cada uno de ellos captura todo el sonido de su entorno de manera independiente. De este modo, el sonido le llegará a cada micrófono en instantes diferentes y gracias a ese desfase se puede calcular de donde viene el sonido. En Kinect, estos micrófonos se encuentran en la parte inferior, tres en el lado izquierdo y uno en el derecho. La parte crucial de este tipo de micrófonos es la manera en la que se procesan las distintas señales recibidas, en el caso de Kinect aplica filtros para quedarse solo con el sonido que tiene enfrente, que es donde estará el usuario, y también elimina todo lo que se encuentre fuera de las frecuencia de la voz humana (entre 80 y 1100 Hz) [22].

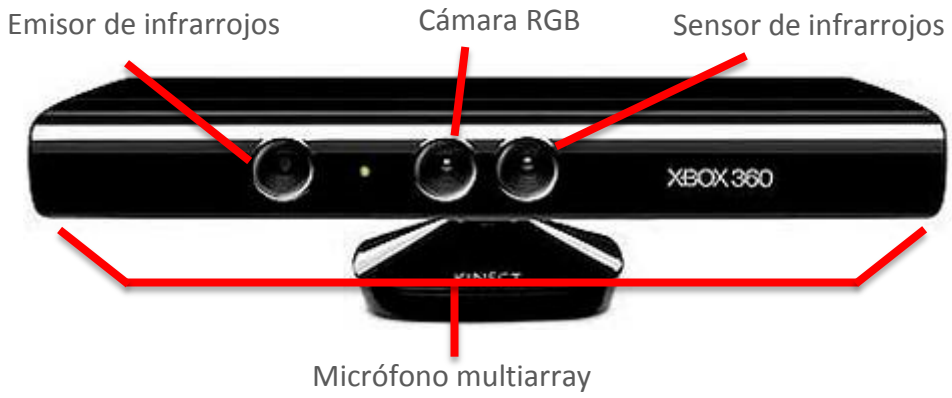


FIGURA 2-27: PARTES DE KINECT

Tanto la cámara RGB como el sensor de profundidad tienen una resolución de 640X480 *pixels*. Según Microsoft, la distancia necesaria para el correcto funcionamiento del sensor Kinect está entre 1'2 y 2'5 metros, no obstante con un rango entre 0'5 y 3'5 metros se siguen obteniendo resultados más que aceptables. Kinect tiene un ángulo de visión de 57° en horizontal y 43° en vertical, y además cuenta con una base motorizada que le permite inclinar el sensor 27° hacia arriba o hacia abajo desde la posición horizontal

A pesar de todas estas características técnicas, donde realmente destaca Kinect es en su software [16]. El software le permite detectar hasta seis personas al mismo tiempo a partir de la información recibida por el sensor de profundidad, siendo capaz de diferenciar con gran exactitud lo que es una persona del resto de mobiliario. Además de esto, puede averiguar la posición y movimientos que realiza una persona, devolviendo información de la posición de sus articulaciones 30 veces por segundo.

3. DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

En este apartado se realiza una descripción del funcionamiento de la aplicación y del proceso de desarrollo que se ha seguido para su implementación. Para ello, tras hablar brevemente de la metodología utilizada, se presentará el análisis y diseño.

3.1. METODOLOGÍA

Para la realización de este proyecto se necesitaba una metodología suficientemente flexible como para permitir cambios sin causar grandes dificultades. Además, debía proporcionar la manera de ir creando versiones cada vez más completas, que aprovecharan el conocimiento adquirido de las versiones anteriores. Esto se debe a que la persona que va a desarrollar el sistema no disponía de conocimientos previos en el manejo de estas tecnologías, y los estudios iniciales no proporcionan la suficiente preparación para llevar a cabo un análisis y un diseño que den lugar a una solución óptima o, al menos, cercana. Se consideró, que la experiencia adquirida a base del trabajo en las distintas versiones iniciales, daría lugar a un sistema realmente robusto y que aprovechará al máximo las capacidades de la tecnología. Aplicar una metodología con estas características ha sido posible gracias a que es un proyecto de pequeñas dimensiones, con una única persona encargada de su desarrollo.

De este modo se seleccionó una metodología incremental que está formada por dos etapas: etapa de inicialización y etapa de iteración. En la etapa de inicialización se lleva a cabo la creación de una primera versión que cumpla con la funcionalidad básica del sistema, definida en la primera versión de los requisitos. Durante la etapa de

iteración se analiza la versión anterior y se rediseña para poder aplicar las mejoras que se crean convenientes. Uno de los inconvenientes de esta metodología es la dificultad para decidir cuando el sistema está terminado, lo que puede provocar grandes retrasos.

A lo largo de este documento, no se reflejará el proceso llevado a cabo al aplicar esta metodología, sino que se presentará una "foto finish" del estado del proyecto.

3.2. ANÁLISIS

A continuación se definirá el proceso de análisis que se ha realizado para construir la aplicación. El objetivo del análisis es llevar a cabo una especificación profunda y detallada de que es lo que debe realizar el sistema y como lo debe hacer, para servir de base durante el posterior diseño de la aplicación.

3.2.1. ALCANCE DEL PROYECTO

En este apartado se definirá de manera completa y concisa lo que debe ser capaz de hacer el sistema a desarrollar y lo que se debe quedar fuera de su funcionalidad.

El objetivo del proyecto es crear un sistema informático que permita a una persona controlar a un robot NAO haciendo uso de una interfaz natural de usuario desde una estación remota de teleoperación. Para crear la interfaz natural de usuario se utilizará el sensor Kinect y un micrófono conectado a un ordenador que a su vez comunicará las órdenes al robot NAO por red.

FIGURA 3-1: ESQUEMA DEL SISTEMA

El usuario podrá usar ordenes de voz para iniciar o detener la teleoperación y para abrir y cerrar las manos. Para controlar el movimiento de los brazos del robot, bastará con que el usuario mueva sus propios brazos y el robot lo imitará. El usuario también debe ser capaz de hacer andar al robot en distintas direcciones, así como de hacerlo girar sobre si mismo, simplemente tomando distintas posiciones con el cuerpo. Será importante que el teleoperador pueda controlar la colocación exacta de las manos, es decir, que sea capaz de que el robot coloque su mano con la palma hacia arriba o en cualquier otra dirección de un modo sencillo.

El usuario deberá poder ver lo que esta viendo el robot NAO con su cámara y también lo que esta observando Kinect para saber si esta siendo correctamente detectado por el sensor. Además, se le debe presentar información textual sobre el estado del proceso de teleoperación. Por su parte, el robot NAO también transmitirá su estado al entorno en el que se encuentra haciendo uso de los altavoces de que dispone, lo que permitirá saber a las personas que estén con él si está siendo teleoperado o no.

El robot no deberá imitar los movimientos que el usuario haga con sus piernas, ya que esto provocaría una perdida de equilibrio y haría que el robot se cayera. Lo únicos movimientos que el robot hará con las piernas serán los necesarios para andar o girar cuando el teleoperador lo requiera. El usuario tampoco controlará directamente el movimiento de la cabeza con movimientos de su propia cabeza, porque al moverla perdería de vista la pantalla donde se le muestra la información, no obstante, el robot

mirara siempre al frente menos cuando se levante una mano para realizar alguna operación, momento en el cual el robot se centrará en seguir la mano.

3.2.2. Especificación de Casos de Uso

En este punto se mostrará el diagrama de casos de uso del sistema y una descripción detallada de cada uno.

3.2.2.1. Características de los casos de uso

Para realizar la definición detallada de cada caso de uso se utilizará el siguiente formato de tabla:

Nombre	
Descripción	
Precondiciones	
Postcondiciones	
Escenario	

Tabla 3-1: Formato de descripción de caso de uso

- **Nombre:** Nombre descriptivo del caso de uso.
- **Descripción:** Explicación clara y concisa del caso de uso que se está especificando.
- **Precondiciones:** Condiciones que se deben cumplir previamente para poder realizar una determinada operación.
- **Postcondiciones:** Estado que presenta el sistema tras la ejecución de una determinada operación.
- **Escenario:** Ejecución del caso de uso paso a paso con un orden determinado.

3.2.2.2. LISTA DE CASOS DE USO.

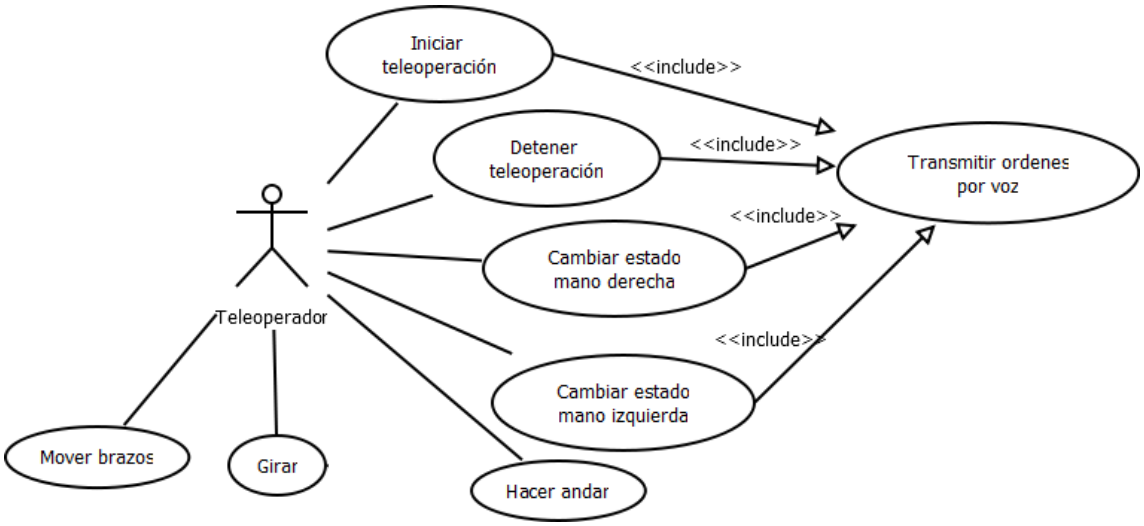


FIGURA 3-2: DIAGRAMA DE CASOS DE USO

Nombre	**Iniciar teleoperación**
Descripción	Se inicia el proceso de teleoperación por el cual el robot comenzará a moverse obedeciendo al usuario.
Precondiciones	• Usuario detectado por Kinect. • Teleoperación detenida.
Postcondiciones	• Teleoperación iniciada.
Escenario	1. Se comprueba si se ha recibido la orden "*start*". 2. Si es así, se cambia el estado de la teleoperación a "iniciada". 3. Se almacena la posición inicial del usuario. 4. Se muestra por pantalla el cambio de estado. 5. Se hace que el robot anuncié el cambio de estado por sus altavoces.

TABLA 3-2: CASO DE USO INICIAR TELEOPERACIÓN

Nombre	**Detener teleoperación**
Descripción	Se detiene el proceso de teleoperación, robot se parará y dejará de imitar los movimientos del usuario y de obedecer sus órdenes.
Precondiciones	• Teleoperación iniciada.
Postcondiciones	• Robot detenido. • Manos del robot cerradas. • Teleoperación detenida.
Escenario	1. Se comprueba si se ha recibido la orden "*finish*". 2. Si es así, se cambia el estado de la teleoperación a "detenida". 3. Se detiene el movimiento del robot. 4. Se cierran las manos del robot. 5. Se muestra por pantalla el cambio de estado. 6. Se hace que el robot anuncié el cambio de estado por sus altavoces.

TABLA 3-3: CASO DE USO DETENER TELEOPERACIÓN

Nombre	**Cambiar estado mano derecha**
Descripción	Mediante esta operación se cambia el estado de la mano derecha del robot, pasando esta de abierta a cerrada o viceversa. De esta manera el robot será capaz de coger y soltar objetos.
Precondiciones	• Teleoperación iniciada.
Postcondiciones	• Mano derecha del robot pasa de abierta a cerrada o viceversa.
Escenario	1. Se comprueba si se ha recibido la orden "*right*". 2. Se manda la orden al robot para que cambie el estado de su mano derecha. 3. Se hace que el robot anuncie el cambio de estado de su mano derecha por sus altavoces.

TABLA 3-4: CASO DE USO ABRIR MANO

Nombre	**Cambiar estado mano izquierda**
Descripción	Mediante esta operación se cambia el estado de la mano izquierda del robot, pasando esta de abierta a cerrada o viceversa. De esta manera el robot será capaz de coger y soltar objetos.
Precondiciones	• Teleoperación iniciada
Postcondiciones	• Mano izquierda del robot pasa de abierta a cerrada o de cerrada a abierta
Escenario	1. Se comprueba si se ha recibido la orden "*left*". 2. Se manda la orden al robot para que cambie el estado de su mano izquierda. 3. Se hace que el robot anuncie el cambio de estado de su mano izquierda por sus altavoces.

TABLA 3-5: CASO DE USO CERRAR MANO

Nombre	**Transmitir ordenes por voz**
Descripción	Se captura lo que el usuario está diciendo por el micrófono y se comprueba si se corresponde con alguna de las órdenes.
Precondiciones	• Teleoperación iniciada.
Postcondiciones	• Orden capturada.
Escenario	1. El usuario habla por el micrófono 2. El sistema detecta si lo que ha dicho corresponde con una orden. 3. Si es una orden, se manda un mensaje con el contenido de dicha orden.

TABLA 3-6: CASO DE USO TRANSMITIR ORDENES POR VOZ

Nombre	**Andar**
Descripción	Acción que hará que el robot comience a andar. Dependiendo de lo que haga el teleoperador, el robot se andará hacia adelante, hacia atrás o hacia los lados. Este caso de uso se podría separar en cuatro distintos, pero debido a que serían prácticamente iguales se ha preferido mantenerlo en uno para simplificar.
Precondiciones	• Teleoperación iniciada.
Postcondiciones	• El robot comenzará a andar en la dirección indicada.
Escenario	1. Se comprueba si el usuario ha variado su posición con respecto al inicio de la teleoperación. 2. Si no se ha movido: a. El robot se mantendrá detenido. 3. Si ha dado un paso al frente: a. El robot comenzará a andar hacia delante. 4. Si ha dado un paso atrás: a. El robot comenzará a andar hacia atrás. 5. Si ha dado un paso a la derecha: a. El robot andará lateralmente hacia la derecha. 6. Si ha dado un paso a la izquierda: a. El robot andará lateralmente hacia la izquierda.

TABLA 3-7: CASO DE USO ANDAR

Nombre	Girar
Descripción	Mediante este caso de uso el robot girará. Se puede combinar con el caso de uso Andar, para que el robot gire a la vez que está andando. Si no se combina con Andar, el robot girará en el sitio sin avanzar hacia ninguna dirección.
Precondiciones	• Teleoperación iniciada.
Postcondiciones	• El robot comenzará a girar en la dirección indicada.
Escenario	1. Se comprueba si el usuario tiene el cuerpo girado, con un hombro delante de otro con respecto a Kinect. 2. Si el usuario no está girado: a. El robot no girará. 3. Si el usuario está girado hacia la derecha: a. El robot girará hacia la derecha. 4. Si el usuario está girado hacia la izquierda: a. El robot girará hacia la izquierda.

TABLA 3-8: CASO DE USO GIRAR

Nombre	Mover brazos
Descripción	Este es el caso de uso mediante el cual el robot imitará los movimientos que el teleoperador haga con los brazos.
Precondiciones	• Teleoperación iniciada.
Postcondiciones	• El robot imitará el movimiento de los brazos del usuario.
Escenario	1. El usuario mueve sus brazos. 2. El sistema capta los ángulos que forma su cuerpo. 3. El sistema adapta los ángulos para que se correspondan con lo que necesita el robot. 4. El robot mueve sus articulaciones de un modo similar al usuario.

TABLA 3-9: CASO DE USO MOVER BRAZOS

3.2.3. ESPECIFICACIÓN REQUISITOS

A partir de la descripción del alcance del sistema, de los casos de uso y del estudio de las necesidades de este tipo de aplicaciones se ha generado el catálogo de requisitos de usuario presente en este apartado.

3.2.3.1. CARACTERÍSTICAS DE LOS REQUISITOS

Para la descripción de cada requisito se utilizará el siguiente formato de tabla:

Nombre			
Descripción			
Prioridad		**Necesidad**	
Estabilidad		**Verificabilidad**	

TABLA 3-10: FORMATO DE TABLA DE REQUISITO

- **Nombre:** Nombre descriptivo del requisito.
- **Descripción:** Explicación clara y concisa del requisito que se está especificando.
- **Prioridad:** Orden de cumplimiento de un requisito. Cuanta mayor prioridad tenga antes tendrá que ser cumplido.
- **Necesidad:** Importancia del requisito.
- **Estabilidad:** Probabilidad de cambio de un requisito.
- **Verificabilidad:** Facilidad para comprobar su funcionamiento.

3.2.3.2. CATÁLOGO DE REQUISITOS

El catálogo de requisitos se separará en requisitos de capacidad y requisitos de restricción. Los requisitos funcionales especifican que tiene que hacer el sistema, incluyendo también los requisitos inversos que describen lo que este no debe hacer. Los requisitos no funcionales especifican la forma en que el sistema debe alcanzar los objetivos o realizar las funcionalidades.

- **Requisitos funcionales**

Nombre	Capturar posición.		
Descripción	El sistema debe ser capaz de analizar la figura del usuario para crear un esqueleto virtual y poder establecer los ángulos y la posición de las articulaciones.		
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Si	**Verificabilidad**	Alta

TABLA 3-11: REQUISITO FUNCIONAL CAPTURAR POSICIÓN

Nombre	Capturar voz.		
Descripción	El sistema debe capturar lo que dice el usuario para permitir comandos de voz.		
Prioridad	Media	**Necesidad**	Esencial
Estabilidad	Si	**Verificabilidad**	Alta

TABLA 3-12: REQUISITO FUNCIONAL CAPTURAR VOZ

Nombre	Iniciar teleoperación.		
Descripción	El sistema permitirá al usuario dar comienzo al proceso de teleoperación mediante el comando de voz "*start*".		
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	No	**Verificabilidad**	Alta

TABLA 3-13: REQUISITO FUNCIONAL INICIAR TELEOPERACIÓN

Nombre	Detener teleoperación.		
Descripción	El sistema permitirá al usuario detener el proceso de teleoperación mediante el comando de voz "*finish*".		
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	No	**Verificabilidad**	Alta

TABLA 3-14: REQUISITO FUNCIONAL DETENER TELEOPERACIÓN

Nombre	Mover brazos.		
Descripción	A partir de la posición capturada del usuario, el sistema procesará los datos recibidos y se los comunicará al robot para que tome una posición similar.		
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Si	**Verificabilidad**	Alta

TABLA 3-15: REQUISITO FUNCIONAL MOVER BRAZOS

Nombre	Andar hacia delante.		
Descripción	Cuando el usuario dé un paso al frente, el sistema mandará la orden al robot para que ande hacia delante.		
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Si	**Verificabilidad**	Alta

TABLA 3-16: REQUISITO FUNCIONAL ANDAR HACIA DELANTE

Nombre	Andar hacia atrás.		
Descripción	Cuando el usuario dé un paso atrás, el sistema mandará la orden al robot para que ande hacia atrás.		
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Si	**Verificabilidad**	Alta

TABLA 3-17: REQUISITO FUNCIONAL ANDAR HACIA ATRÁS

Nombre	Andar hacia la derecha.		
Descripción	Cuando el usuario dé un paso a la derecha, el sistema mandará la orden al robot para que ande lateralmente hacia la derecha.		
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Si	**Verificabilidad**	Alta

TABLA 3-18: REQUISITO FUNCIONAL ANDAR HACIA LA DERECHA

Nombre	Andar hacia la izquierda.		
Descripción	Cuando el usuario dé un paso a la izquierda, el sistema mandará la orden al robot para que ande lateralmente hacia la izquierda.		
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Si	**Verificabilidad**	Alta

TABLA 3-19: REQUISITO FUNCIONAL ANDAR HACIA LA IZQUIERDA

Nombre	Girar a la derecha.		
Descripción	Cuando el usuario tenga el cuerpo girado hacia la derecha, el sistema mandará la orden al robot para que gire hacia la derecha.		
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Si	**Verificabilidad**	Alta

TABLA 3-20: REQUISITO FUNCIONAL GIRAR A LA DERECHA

Nombre	Girar a la izquierda.		
Descripción	Cuando el usuario tenga el cuerpo girado hacia la izquierda, el sistema mandará la orden al robot para que gire hacia la izquierda.		
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Si	**Verificabilidad**	Alta

TABLA 3-21: REQUISITO FUNCIONAL GIRAR A LA IZQUIERDA

Nombre	Parar.		
Descripción	Para que el robot deje de andar o girar bastará con que el usuario vuelva a su posición inicial o detenga la teleoperación.		
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Si	**Verificabilidad**	Alta

TABLA 3-22: REQUISITO FUNCIONAL PARAR

Nombre	Mover muñecas.		
Descripción	El sistema deberá ser capaz de interpretar la posición de las manos del usuario en cuanto a su orientación espacial para, posteriormente, mandar al robot la orden que ponga sus manos en la misma posición.		
Prioridad	Media	**Necesidad**	Esencial
Estabilidad	Si	**Verificabilidad**	Alta

TABLA 3-23: REQUISITO FUNCIONAL MOVER MUÑECAS

Nombre	Cambiar estado mano derecha.		
Descripción	El sistema mandará la orden al robot de que cambie el estado de su mano de derecha de abierta a cerrada o viceversa, cuando el usuario exprese el comando de voz "*right*".		
Prioridad	Media	**Necesidad**	Esencial
Estabilidad	No	**Verificabilidad**	Alta

TABLA 3-24: REQUISITO FUNCIONAL CAMBIAR ESTADO MANO DERECHA

Nombre	Cambiar estado mano izquierda.		
Descripción	El sistema mandará la orden al robot de que cambie el estado de su mano de izquierda de abierta a cerrada o viceversa, cuando el usuario exprese el comando de voz "*left*".		
Prioridad	Media	**Necesidad**	Esencial
Estabilidad	No	**Verificabilidad**	Alta

TABLA 3-25: REQUISITO FUNCIONAL CAMBIAR ESTADO MANO IZQUIERDA

Nombre	Mostrar cámara.		
Descripción	El sistema capturará lo que esta viendo el robot por su cámara y se lo mostrará al usuario por pantalla.		
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Si	**Verificabilidad**	Alta

TABLA 3-26: REQUISITO FUNCIONAL MOSTRAR CÁMARA

Nombre	Seguir manos.		
Descripción	El robot deberá mirar siempre al frente, pero cuando el sistema dé la orden de levantar alguna mano, el robot se centrará en ellas. Siempre que la mano derecha del robot esté levantada, la cámara la seguirá, cuando la mano derecha este baja y se levante la izquierda, el robot se fijará en la izquierda.		
Prioridad	Media	**Necesidad**	Deseable
Estabilidad	Si	**Verificabilidad**	Alta

TABLA 3-27: REQUISITO FUNCIONAL SEGUIR MANOS

Nombre	Mostrar información textual.		
Descripción	El sistema deberá ir mostrando por la consola texto que indique el estado de la teleoperación.		
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	No	**Verificabilidad**	Alta

TABLA 3-28: REQUISITO FUNCIONAL MOSTRAR INFORMACIÓN TEXTUAL

Nombre	Mostrar imágenes de Kinect.		
Descripción	El sistema mostrará al usuario lo que está capturando el sensor Kinect para que pueda saber si esta siendo correctamente detectado o existen errores de calibración.		
Prioridad	Alta	**Necesidad**	Deseable
Estabilidad	No	**Verificabilidad**	Alta

TABLA 3-29: REQUISITO FUNCIONAL MOSTRAR IMÁGENES DE KINECT

Nombre	Notificaciones del robot.		
Descripción	El robot notificará por sus altavoces cuando se produzca un cambio en el estado de la teleoperación o cuando comience a abrir o cerrar una mano.		
Prioridad	Baja	**Necesidad**	Deseable
Estabilidad	No	**Verificabilidad**	Alta

TABLA 3-30: REQUISITO FUNCIONAL NOTIFICACIONES DEL ROBOT

- **Requisitos inversos**

Nombre	Mover piernas.		
Descripción	El sistema no debe mover las piernas del robot, imitando el movimiento del usuario.		
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Si	**Verificabilidad**	Alta

TABLA 3-31: REQUISITO INVERSO MOVER PIERNAS

Nombre	Mover cabeza.		
Descripción	El sistema no debe mover la cabeza del robot, imitando el movimiento del usuario.		
Prioridad	Baja	**Necesidad**	Deseable
Estabilidad	No	**Verificabilidad**	Alta

TABLA 3-32: REQUISITO INVERSO MOVER CABEZA

- **Requisitos no funcionales**

Nombre	Utilizar el robot NAO.		
Descripción	El robot utilizado en el proyecto será el robot NAO.		
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Si	**Verificabilidad**	Alta

TABLA 3-33: REQUISITO NO FUNCIONAL UTILIZAR EL ROBOT NAO

Nombre	Utilizar Kinect.		
Descripción	La captura del teleoperador se realizará mediante Kinect.		
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Si	**Verificabilidad**	Alta

TABLA 3-34: REQUISITO NO FUNCIONAL UTILIZAR KINECT

Nombre	Utilizar ROS.		
Descripción	El sistema deberá utilizar el *framework* ROS.		
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Si	**Verificabilidad**	Alta

TABLA 3-35: REQUISITO NO FUNCIONAL UTILIZAR ROS

Nombre	Utilizar Ubuntu.		
Descripción	El sistema deberá utilizar el sistema operativo Ubuntu 11.04.		
Prioridad	Alta	**Necesidad**	Deseable
Estabilidad	No	**Verificabilidad**	Alta

TABLA 3-36: REQUISITO NO FUNCIONAL UTILIZAR UBUNTU

Nombre	Utilizar micrófono.		
Descripción	El sistema deberá usar un micrófono para capturar la voz del usuario. Este requisito fue añadido a posteriori al comprobar que los controladores de Kinect para Ubuntu no incluían la funcionalidad de manejar el micrófono multiarray.		
Prioridad	Media	**Necesidad**	Esencial
Estabilidad	Si	**Verificabilidad**	Alta

TABLA 3-37: REQUISITO NO FUNCIONAL UTILIZAR MICRÓFONO

Nombre	Tiempo de retardo		
Descripción	El retardo entre que el usuario se mueve y esto se ve reflejado en el robot debe ser mínimo, siempre inferior a un segundo.		
Prioridad	Media	**Necesidad**	Deseable
Estabilidad	No	**Verificabilidad**	Alta

TABLA 3-38: REQUISITO NO FUNCIONAL TIEMPO DE RETARDO

Nombre	Diseño modular.		
Descripción	El sistema desarrollado tendrá un diseño modular que permita de una manera sencilla ampliar funcionalidad, sustituir el robot a teleoperar o cambiar el método de teleoperación.		
Prioridad	Media	**Necesidad**	Deseable
Estabilidad	Si	**Verificabilidad**	Media

TABLA 3-39: REQUISITO NO FUNCIONAL DISEÑO MODULAR

Nombre	Rápido aprendizaje.		
Descripción	El usuario no necesitará de un costoso aprendizaje para realizar la teleoperación de manera correcta y alcanzar objetivos sencillos rápidamente.		
Prioridad	Alta	**Necesidad**	Deseable
Estabilidad	Si	**Verificabilidad**	Media

TABLA 3-40: REQUISITO NO FUNCIONAL RÁPIDO APRENDIZAJE

3.3. DISEÑO

El principal objetivo de este apartado es realizar la definición de la arquitectura del sistema y de todo el entorno tecnológico que le dará soporte, así como desarrollar la especificación detallada de los componentes del sistema.

En el análisis se ha ofrecido la respuesta a qué debe hacer el sistema. En este punto se responderá al como se ha desarrollado la aplicación para que cumpla con las necesidades identificadas en el análisis.

3.3.1. ARQUITECTURA

En este apartado se mostrará la arquitectura del sistema y su descomposición en distintos componentes. Para llevar a cabo la descomposición, el sistema se basara en una arquitectura por capas con tres niveles diferenciados. Estos niveles son:

- **Capa robot**: esta capa lleva a cabo la conexión con el robot y la transmisión de las órdenes llegadas de la capa de control.
- **Capa de control**: es la capa encargada de implementar la lógica de control del proceso de teleoperación y de analizar y adaptar la información que llega de la capa operador para transmitírsela al robot.
- **Capa operador**: en esta capa se encuentran todos los componentes encargados de captar correctamente la silueta y la voz del usuario.

Por encima de estas tres capas se podría encontrar otra capa que se encarga de definir el formato de los mensajes que se transmiten las distintas capas entre sí. A su vez, cada una de estas capas mostrará información por pantalla sin necesidad de que las demás se vean afectadas.

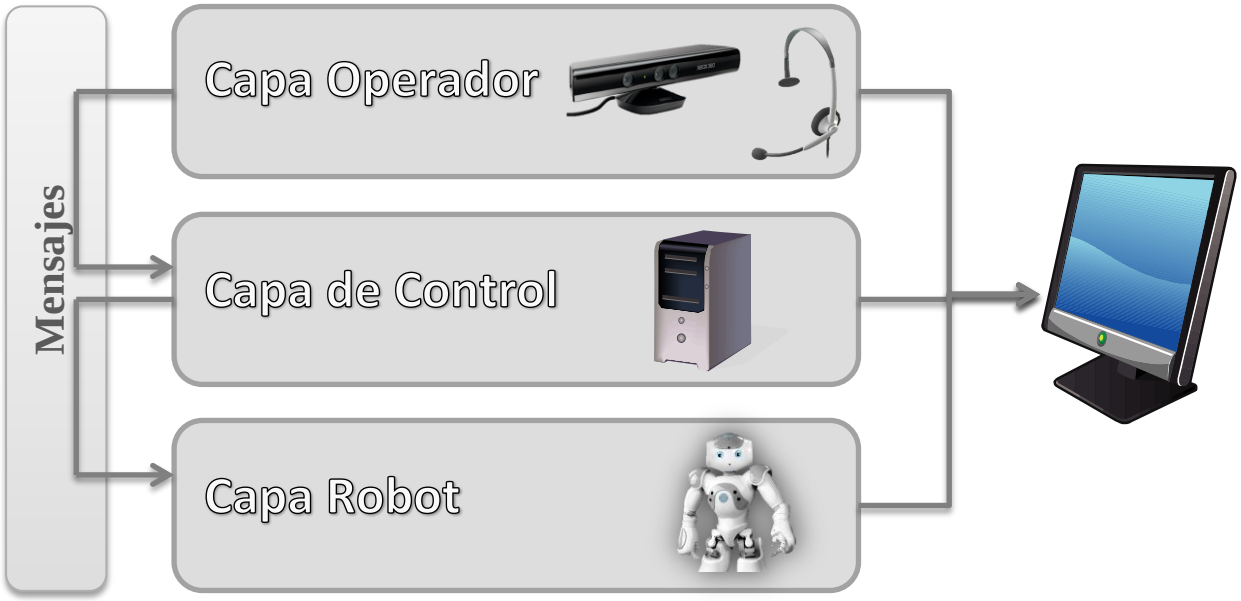


FIGURA 3-3: ARQUITECTURA DEL SISTEMA

La principal razón para elegir esta arquitectura y no otra, es que la disposición por capas da lugar a una gran modularidad. Cada capa es independiente de las otras y no es necesario que conozca nada de su funcionamiento interno, simplemente necesita conocer el modelo de comunicación y la estructura de los mensajes que se intercambiarán. En este caso particular, esta arquitectura hace al sistema muy flexible, ya que permite cambiar cualquiera de los módulos sin que esto suponga un gran conflicto en el resto de ellos. Por ejemplo, si pasado un tiempo aparece un sensor más avanzado que Kinect bastaría con intercambiar la capa operador actual por otra que utilice el nuevo sensor; y si se quisiera teleoperar un robot que no fuese el robot NAO, únicamente habría que modificar la capa robot.

En la página siguiente se muestra un esquema con la descomposición del sistema en distintos componentes y como se integran dentro de la arquitectura seleccionada. Este diagrama no se basa en ningún estándar, es una descripción básica de las relaciones que se establecen entre los distintos módulos, de forma que un usuario sin conocimientos avanzados pueda comprenderlo. En el diagrama se pueden observar los siguientes elementos:

- Flechas continuas: indican que un componente envía mensajes a otro.
- Flechas discontinuas: señalan que un componente depende de algún elemento de otro.
- Componentes de texto blanco sobre fondo oscuro: se refieren a los componentes que han sido implementados por completo como parte del desarrollo del proyecto.
- Componentes de texto oscuro sobre fondo blanco: son los que ya estaban previamente desarrollados por distintas personas e instituciones y que son accesibles gracias a la amplia comunidad de colaboradores que ROS tiene en internet.

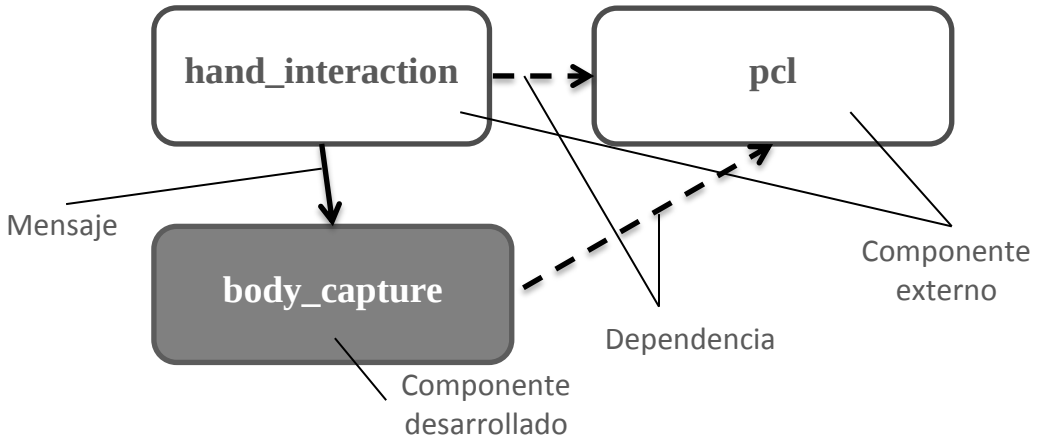


FIGURA 3-4: ELEMENTOS DEL DIAGRAMA

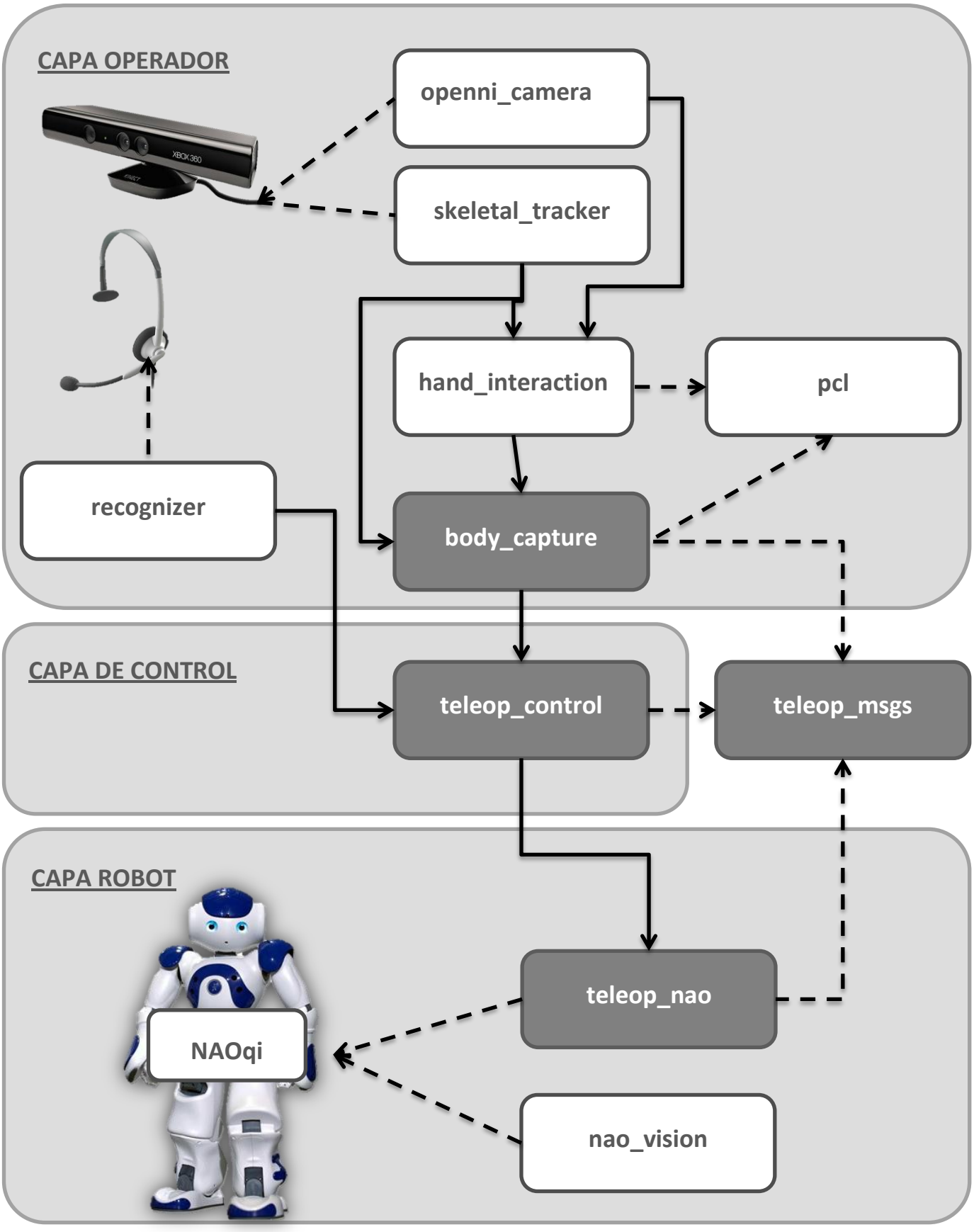


FIGURA 3-5: ESQUEMA DE COMPONENTES

3.3.2. DESCRIPCIÓN GENERAL DEL SISTEMA

En este apartado se expondrá una descripción general del funcionamiento del sistema, identificando que nodos actúan en cada fase. Simplificando, el sistema se encarga de capturar la posición del cuerpo del operador mediante Kinect, analizarla para calcular los ángulos que forman sus articulaciones, recoger las palabras que pronuncia, controlar el proceso de teleoperación, conectarse al robot Nao y formar las ordenes que se le enviarán. Las fases en las que se puede dividir el funcionamiento del sistema son:

1. **Capturar la postura del operador.**

   Los nodos que están directamente relacionados con esta fase son:

   - *openni_camera*: obtiene la nube de puntos capturada por Kinect y se la envía a hand_interaction.
   - *skeletal_tracker*: a partir de la información recibida de Kinect, comprueba si hay alguna persona en la escena y, tras una calibración inicial, comienza a capturar la posición de distintas partes del cuerpo del usuario. Está información se la enviará a *hand_interaction* y a *body_capture*.
   - *hand_interaction*: recibe la nube de puntos capturada y la postura del operador. Posteriormente, analiza esta información para obtener las dos secciones de la nube de puntos que corresponden con el lugar que ocupa cada mano. Las dos nubes de puntos resultantes se las enviará a *body_capture*. Para realizar las operaciones con las nubes de puntos utiliza *pcl*.

2. **Cálculo de los ángulos de las articulaciones.**

El nodo encargado de llevar a cabo este procesamiento es *body_capture*. Utiliza la información recibida de *skeletal_tracker*, con la posición del operador, para obtener los distintos ángulos que forman sus articulaciones. Además, examina las nubes de puntos que corresponden con las manos del operador, enviadas por *hand_interaction*, para determinar su colocación. Toda esta información se le enviará a *teleop_control*.

3. **Reconocimiento del habla.**

El nodo recognizer lleva a cabo esta función. Para ello, accede al audio que esta siendo recogido por el micrófono y cuando detecta que el operador ha pronunciado alguna palabra que debe ser capturada, le envía un mensaje a *teleop_control*.

4. **Control del proceso de teleoperación.**

El encargado de llevar a cabo el control es *teleop_control*. Sus funciones son:

- Adaptar los ángulos a la forma que deben tener para ser transmitidos al robot.
- Comunicar al robot las frases que debe pronunciar por sus altavoces.
- Recibir las palabras detectadas por recognizer, y llevar a cabo acción que corresponde con cada una. Estas acciones serán, iniciar y detener la teleoperación y ordenar la apertura y cierre de las manos de robot.
- Analizar la posición del operador para comprobar si se debe ordenar al robot que inicie algún tipo de desplazamiento.
- Formar los mensajes que se le enviarán a *teleop_nao*.

5. **Comunicación con el robot.**

Los nodos involucrados en esta fase son:

- *teleop_nao*: se conecta al robot Nao y comienza a transmitirle las órdenes con la información de los mensajes recibidos de *teleop_control*.
- *nao_vision*: se encarga de acceder a las imágenes que el robot Nao está capturando con su cámara y publicarla.

6. **Publicar información para el operador.**

Los nodos encargados de mostrar la información al operador para que pueda realizar la teleoperación, son:

- *skeletal_tracker*: muestra una ventana donde se puede observar lo que esta detectando Kinect.
- *teleop_control*: va escribiendo en consola información relevante para conocer el estado del proceso de teleoperación.
- *nao_vision*: publica en una ventana las imágenes que el robot NAO detecta a través de su cámara.

3.3.3. DESCRIPCIÓN DE COMPONENTES

En este apartado se va a proceder a dar una descripción detallada del funcionamiento de cada componente identificado en el punto 3.3.1 Arquitectura. Se agruparán según las tres capas de la arquitectura descrita en el apartado anterior, con un primer apartado para el módulo encargado de la definición de los mensajes.

3.3.3.1. TELEOP_MSGS

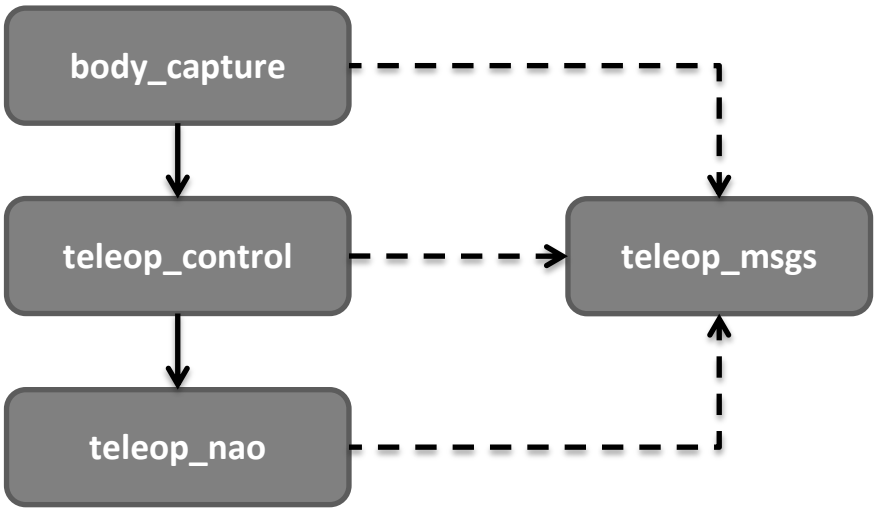


FIGURA 3-6: ESQUEMA DEL NODO TELEOP_MSGS

Este módulo se encarga de definir los mensajes que se intercambiaran el resto de nodos entre sí. De esta manera se evitan fuertes dependencias entre los nodos, consiguiendo que si se quiere sustituir alguno, valga con que el nuevo nodo utilice los mensajes de ***teleop_msgs***.

Los mensajes que se definen en el nodo son:

- *body_angle.*
- *joint_angle.*
- *walk_info.*

3.3.3.2. CAPA OPERADOR

En esta sección se detallarán las características de los nodos que están comprendidos dentro de la capa operador. Estos nodos son los que se encargarán de recoger directamente la posición del cuerpo del operador y sus órdenes de voz.

3.3.3.2.1. OPENNI_CAMERA

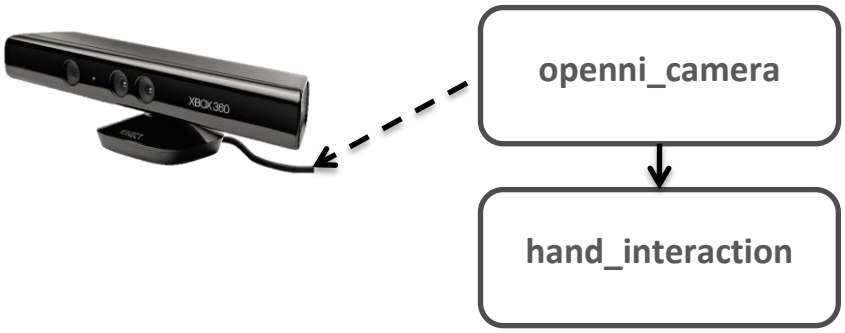


FIGURA 3-7: ESQUEMA DEL NODO OPENNI_CAMERA

OpenNI es una entidad sin ánimo de lucro encargada de certificar y mejorar la interoperabilidad entre los dispositivos que crean interfaces naturales de usuario, las aplicaciones que los utilizan y el *middleware* que facilita su uso. Uno de los principales colaboradores de *OpenNI* es la empresa *PrimeSense* encargada de la creación de Kinect. Como parte de su función, *OpenNI* creo un API (*application programming interface*) que suministra un método estándar de acceder a las funciones de este tipo de dispositivos y en el que se basa este nodo para trabajar con Kinect. ***openni_camera*** se utiliza como controlador del sensor Kinect, realizando la interconexión entre el dispositivo y ROS, es decir, conecta el sensor, accede a la información que este está recogiendo y realiza una traducción a la estructura de mensajes ofrecida por ROS.

Este nodo es capaz de transmitir muy variada información y, por lo tanto, varios tipos de mensajes sobre la configuración y la información que está capturando Kinect. No obstante, este sistema solo hace uso de uno de estos mensajes denominado */camera/rgb/points*. Este mensaje transporta la información sobre la nube de puntos recogida por Kinect; consiste en una larga lista en el que cada elemento consta de las tres coordenadas espaciales e información sobre el color detectado en esa posición. A

continuación se puede ver un ejemplo de lo que transporta este mensaje en un instante, en la imagen se ha eliminado la información del color para sustituirlo por una escala que representa la profundidad de cada punto y que pasa del blanco (cerca del sensor) al negro (lejos del sensor).

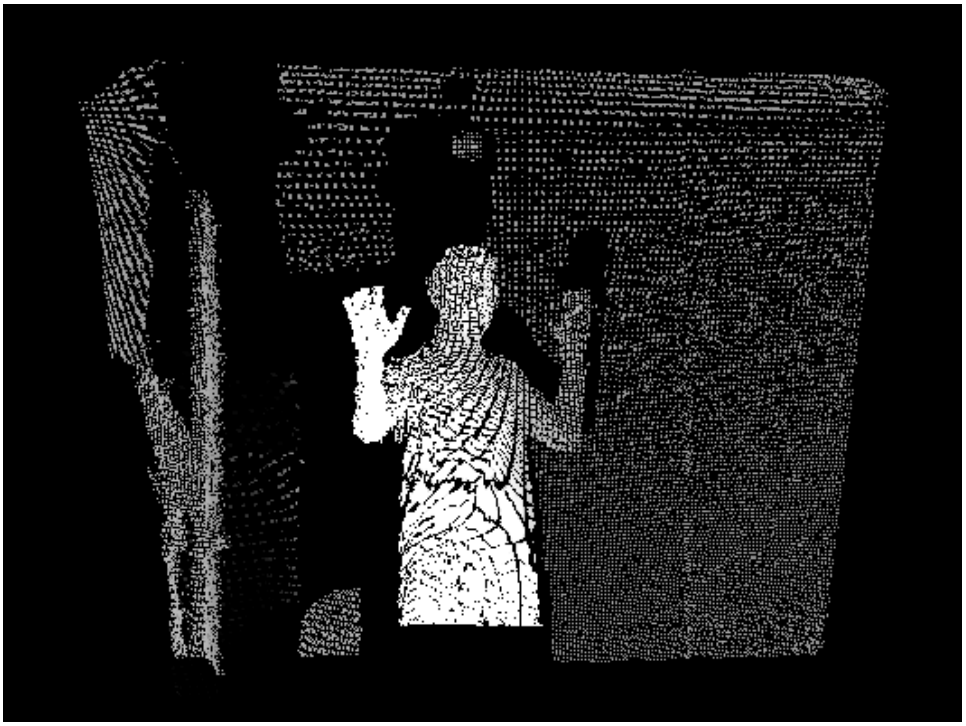


FIGURA 3-8: CONTENIDO DEL MENSAJE /CAMERA/RGB/POINTS

3.3.3.2.2. SKELETAL_TRACKER

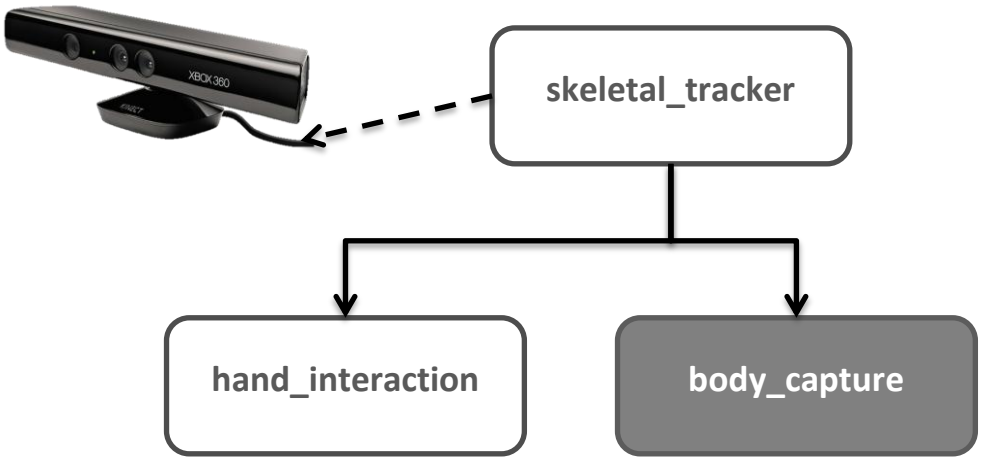


FIGURA 3-9: ESQUEMA DEL NODO SKELETAL_TRACKER

Este nodo ha sido creado por Garratt Gallagher, un ingeniero del MIT como parte de un conjunto de herramientas y curiosidades que el MIT ha desarrollado para Kinect. Al igual que el nodo anterior, utiliza el estándar creado por *OpenNI* que permite trabajar con Kinect y otra variedad de sensores similares. Haciendo uso de las funciones que le proporciona *OpenNI*, ***skeletal_tracker*** lleva a cabo las siguientes funciones:

- Diferenciar lo que es una persona dentro de la escena captada por Kinect.
- Mostrar por pantalla una ventana donde se muestra lo que Kinect está percibiendo.
- Realizar una calibración del usuario, para lo cual este debe mantenerse en una determinada posición, como se ve en la Figura 3-12.
- Determinar la posición de quince partes del cuerpo del usuario, las cuales se detallarán más adelante en la descripción de los mensajes.

En este proyecto se ha utilizado una versión de ***skeletal_tracker*** a la que se le ha realizado una pequeña modificación. El nodo necesita que cierto archivo de configuración se encuentre almacenado en un lugar determinado del sistema operativo y se ha decidido realizar los cambios oportunos para que la ubicación de dicho archivo se pueda pasar como parámetro en un fichero de configuración, cuyo contenido se detalla en el anexo D. Ejecución , en lugar de que tenga que estar en un

lugar fijo como ocurre en la versión normal. Es importante destacar, que para que este nodo pueda acceder a Kinect, el nodo ***openni_camera*** ha tenido que conectarse al sensor previamente.

Este nodo emite dos tipos de mensajes, pero solo nos centraremos en uno de ellos, el llamado */Skeletons* del paquete ***body_msgs***. El contenido de este mensaje se detalla en B.2 /skeletons. El paquete ***body_msgs*** solo se encarga de definir algunos mensajes para que sean utilizados por otros nodos.

El proceso para la construcción del mensaje consiste en los siguientes pasos:

1. Cuando se ejecuta el nodo ***openni_camera***, se conecta a Kinect y ***skeletal_tracker*** comienza a emitir lo que Kinect está detectando.

FIGURA 3-10: FUNCIONAMIENTO SKELETAL_TRACKER 1

2. Se detecta el lugar que ocupa la persona dentro de la escena capturada y en la imagen mostrada se colorea su silueta. El sistema se queda a la espera de que el usuario tome la posición inicial, para comenzar el proceso de calibración.

FIGURA 3-11: FUNCIONAMIENTO SKELETAL_TRACKER 2

3. El sistema detecta que el usuario ha adquirido la posición inicial y comienza el proceso de calibración que dura unos pocos segundos.

FIGURA 3-12: FUNCIONAMIENTO SKELETAL_TRACKER 3

4. Una vez que la calibración ha terminado, se empieza a mostrar un rudimentario esqueleto sobre la silueta del usuario y se comienzan a enviar los mensajes con la información del lugar que ocupan las articulaciones del cuerpo.

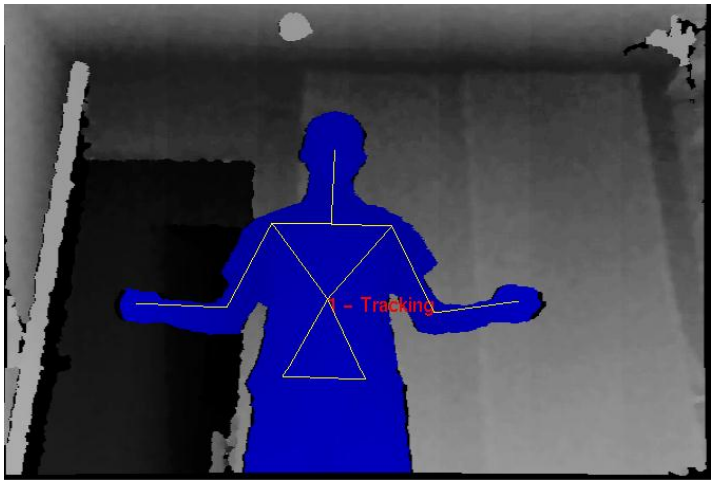


FIGURA 3-13: FUNCIONAMIENTO SKELETAL_TRACKER 4

3.3.3.2.3. PCL

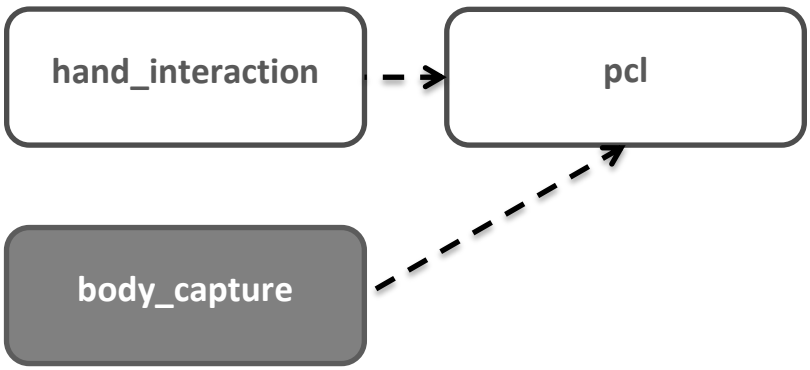


FIGURA 3-14: ESQUEMA DEL NODO PCL

Las siglas PCL se corresponden con *Point Cloud Library*, que es una librería de código libre, que permite trabajar con nubes de puntos tridimensionales [23]. Ofrece un amplio número de funciones que permiten realizar todo tipo de operaciones con nubes de puntos, como por ejemplo:

- Aplicar distintos filtros.
- Estimación de funciones.
- Reconstrucción de superficies.
- Segmentación.
- Ajuste de modelos tridimensionales.
- Como la construcción de mapas.
- Reconocimiento de objetos.

En el caso de este proyecto, se han utilizado distintos filtros y funciones para búsqueda de características, más adelante se detallará cual es la manera en la que se usan las capacidades de esta librería por los distintos nodos.

3.3.3.2.4. HAND_INTERACTION

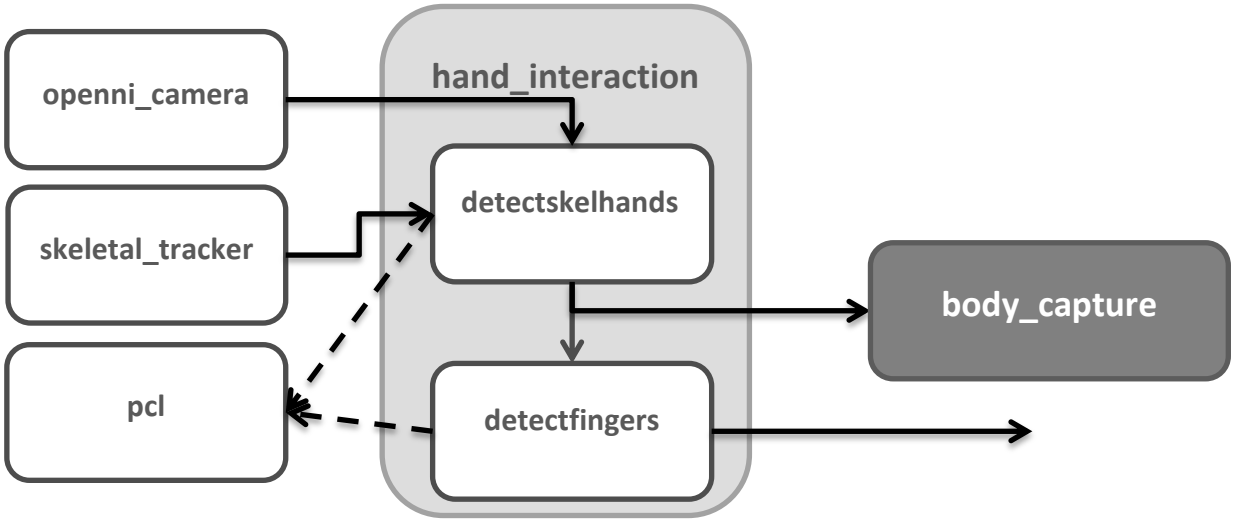


FIGURA 3-15: ESQUEMA DEL NODO HAND_INTERACTION

Este nodo, al igual que ***skeletal_tracker***, ha sido creado por Garrat Gallaguer para integrarse en el conjunto de nodos creados por el MIT para ROS y, especialmente, para Kinect. Fue creado con el objetivo de detectar los dedos de las manos para permitir al usuario tocar un piano virtual. Para ello, el usuario tiene que colocar las manos frente a Kinect con los dedos extendidos, y cuando haga el movimiento de pulsar una tecla con un dedo, sonará una nota dependiendo del dedo que haya articulado.

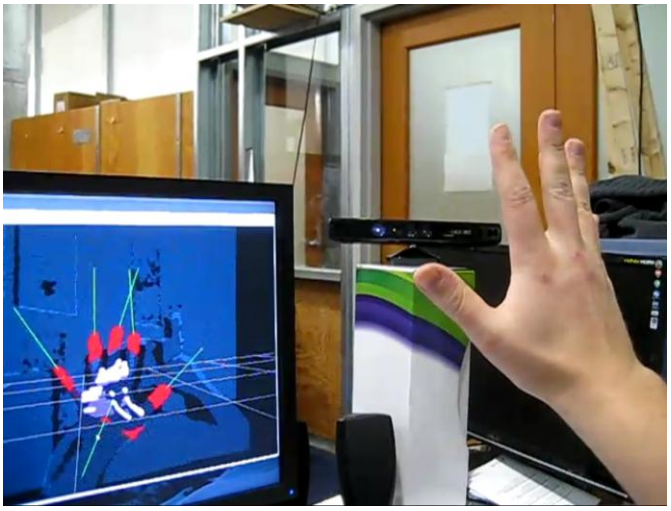


FIGURA 3-16: DEMOSTRACIÓN DEL PIANO VIRTUAL DEL MIT

Para la construcción de este piano virtual este paquete tiene dos nodos diferenciados: uno es el nodo ***detectskelhands*** y otro, el nodo ***detectfingers***. El nodo ***detectfingers*** es el encargado de averiguar la posición de los dedos a partir de la nube

de puntos capturada por ***detectskelhands*** que forma la mano. ***Detectfingers*** no se emplea en este sistema, ya que solo detecta los dedos en condiciones muy especificas, cuando la mano esta extendida delante de Kinect y a una distancia de unos 50 centímetros.

Este nodo utiliza ***pcl*** para llevar ha cabo su cometido. Su funcionamiento es el siguiente:

1. ***detectskelhands*** recibe dos mensajes: del nodo ***openni_camera***, la nube de puntos con lo detectado por Kinect, y del nodo ***skeletal_tracker***, la información con la posición de las partes del cuerpo del usuario.

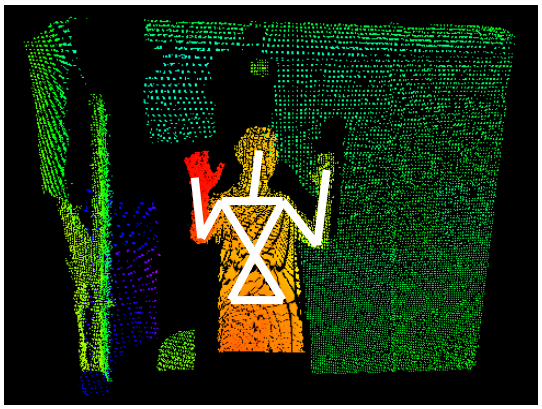


FIGURA 3-17: COMPOSICIÓN DE LOS MENSAJES RECIBIDOS POR DETECTSKELHANDS

2. Se aplica un método para seleccionar solo con los puntos cercanos a la posición que según el mensaje de ***skeletal_tracker*** ocupan las manos. A partir de los puntos resultantes, se obtiene su centro de gravedad y se eliminan los puntos alejados de este, para asegurarse de que los puntos obtenidos son los que forman la mano y reducir el ruido. Con este procedimiento, se obtienen dos nubes de puntos, una para la mano izquierda y otra para la derecha, que formarán los mensajes utilizados posteriormente por ***detectfingers*** y, lo que es más importante para este proyecto, por el nodo ***body_capture***. En este proceso se emplea ***pcl***.

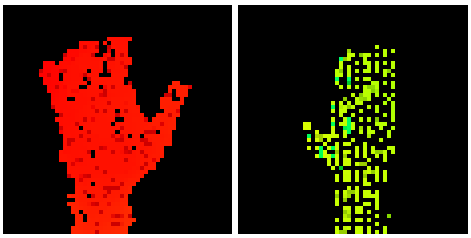


FIGURA 3-18: EJEMPLO DE MENSAJE ENVIADO POR DETECTSKELHANDS

3. Cuando ***detectfingers*** recibe el mensaje con una nube de puntos, lo primero que hace es eliminar los puntos que forman la palma de la mano. Esto se realiza aplicando un filtro que elimina los puntos que se encuentran a una distancia dada de uno central, que en este caso es el centro de la palma. En ***detectfingers*** esta distancia es fija, por lo que existen problemas para trabajar con manos de distintas formas y tamaños.

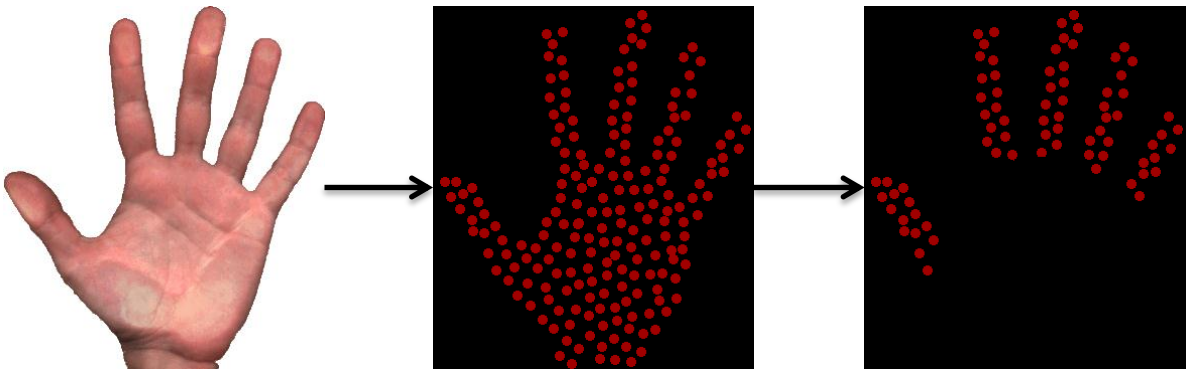


FIGURA 3-19: PROCESAMIENTO DE LAS MANOS POR PARTE DE DETECFINGERS

4. Posteriormente, aplica técnicas de *clustering* o agrupación, para averiguar exactamente que puntos forman cada dedo. Finalmente, busca el centroide de cada nube de puntos que forman los dedos, para saber su posición.

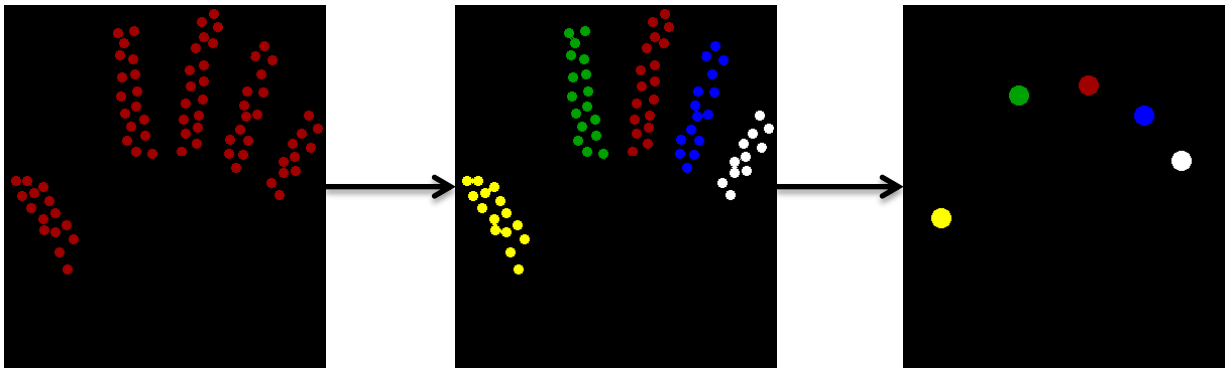


FIGURA 3-20: PROCESAMIENTO DE LAS MANOS POR PARTE DE DETECFINGERS 2

3.3.3.2.5. BODY_CAPTURE

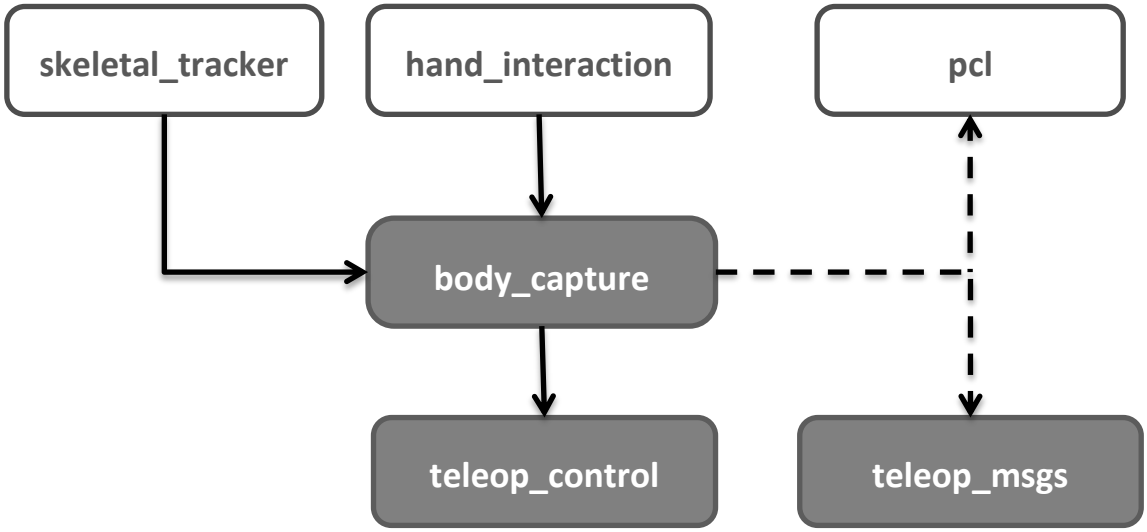
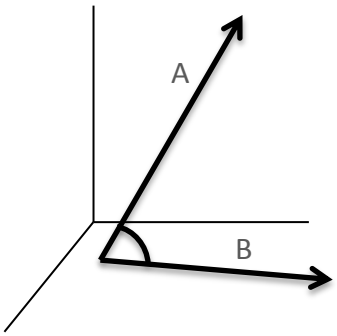


FIGURA 3-21: ESQUEMA DEL NODO BODY_CAPTURE

La funcionalidad de este nodo es analizar la postura del usuario y obtener a partir de ella el ángulo que forman las articulaciones de los brazos. Utilizando esta información crea un mensaje del tipo ***body_angle***, cuyo contenido exacto se detalla en *B.7 /bodyAngle*. Los ángulos que se determinarán son aquellos equivalentes a cada una de las articulaciones de las que dispone el robot NAO en sus brazos, lo que permite hacer una rápida traslación del ángulo medido en el operador a la postura que debe tomar el robot. A continuación, se procederá a describir el funcionamiento de este nodo:

- **Cálculo de los ángulos de las articulaciones**

En primer lugar, se recibe un mensaje del tipo */skeletons* formado por la posición de las distintas partes del cuerpo. Seguidamente, se utiliza la siguiente función trigonométrica para calcular los ángulos que forman las extremidades superiores:



$$\alpha = \arccos \frac{A \cdot B}{|A||B|}$$

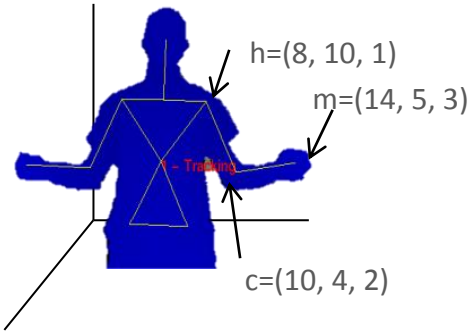
Siendo:

$$A \cdot B = A_x B_x + A_y B_y + A_z B_z$$

$$|A| = \sqrt{A_x^2 + A_y^2 + A_z^2}$$

$$|B| = \sqrt{B_x^2 + B_y^2 + B_z^2}$$

A continuación se muestra un ejemplo de como se calcularía el ángulo de apertura del codo mediante estas fórmulas, a partir de las posiciones de ciertas partes del cuerpo:



$$A = (c_x - h_x, c_y - h_y, c_z - h_z)$$

$$A = (10 - 8{,}4 - 8{,}2 - 1) = (2, -4, 1)$$

$$B = (m_x - c_x, m_y - c_y, m_z - c_z)$$

$$B = (14 - 10, 5 - 4{,}3 - 2) = (4{,}1{,}1)$$

$$A \cdot B = 2 \cdot 4 + (-4) \cdot 1 + 1 \cdot 1 = 5$$

$$|A| = \sqrt{2^2 + (-4)^2 + 1^2} = \sqrt{21} = 4{,}5$$

$$|B| = \sqrt{4^2 + 1^2 + 1^2} = \sqrt{18} = 4{,}2$$

$$\alpha = \arccos \frac{5}{4{,}5 \cdot 4{,}2} = 74{,}7°$$

Cabe destacar que los vectores de los que se quiere calcular el ángulo no tienen por qué compartir un vértice como se muestra en las figuras anteriores, si no que pueden estar totalmente separados, siendo independientes uno del otro.

Posteriormente, se va a detallar como se determina cada uno de los ángulos involucrados en el proceso de teleoperación y que son los que forman parte del mensaje */body_angles*.

- Ángulo de apertura del hombro

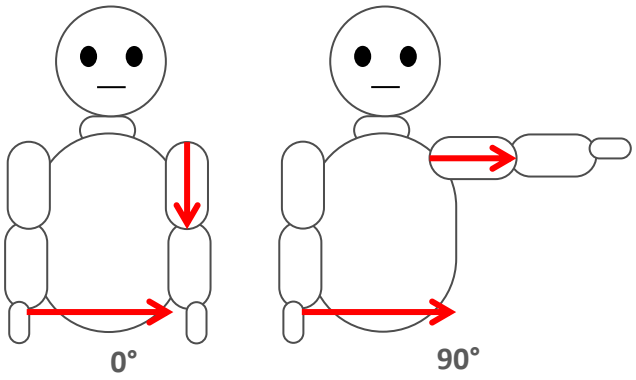


FIGURA 3-22: CÁLCULO DE LA APERTURA DEL HOMBRO

Para calcular el ángulo de apertura del hombro, los dos vectores que se utilizan son: el vector formado por la unión de las dos caderas, y el resultante de la unión del hombro y el codo del mismo brazo. Se ha seleccionado el vector de las caderas en lugar del que une el hombro con la cadera de su lado, aunque a priori parezca el más adecuado. De haber utilizado dicho vector, el ángulo que formaría con el brazo también variaría al moverlo de adelante a atrás, provocando que al mover el brazo hacia delante el robot lo extendiera hacia el lado. Como se observa en la Figura 3-22, para que el resultado sea más intuitivo, se le suman 90° al ángulo obtenido al aplicar la fórmula trigonométrica y, de este modo, cuando el brazo está pegado al cuerpo se considera que se forma un ángulo de 0° y cuando esta totalmente extendido el ángulo será de 90°. El motivo de no tener en cuenta un rango mayor de la apertura del brazo es la imposibilidad del robot NAO de realizar ese tipo de movimiento.

Dentro del mensaje */body_angles* el elemento que define la información de este ángulo se denomina *RShoulderOpeningAngle* para el brazo derecho y *LShoulderOpeningAngle* para el izquierdo.

- Ángulo de rotación del hombro

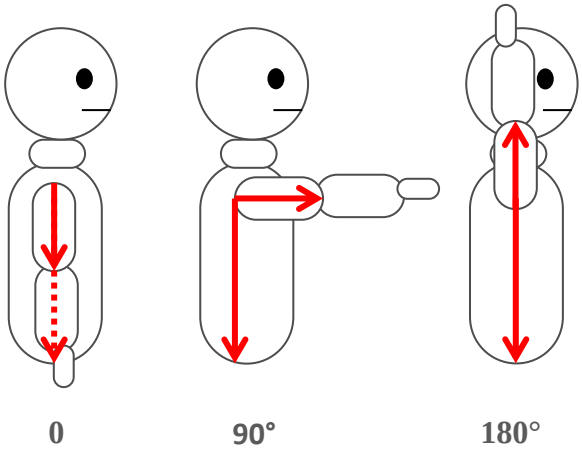


FIGURA 3-23 : CÁLCULO DE LA ROTACIÓN DEL HOMBRO

En este caso, los vectores utilizados para establecer el ángulo de rotación de la articulación del hombro son el vector que une el hombro con su cadera y el que lo une con el codo. Se han elegido estos vectores a pesar de que este ángulo también cambia cuando el brazo se extiende hacia el lado, por que esto no da lugar a un gran problema, ya que lo único que provoca es que el brazo gire sobre si mismo. De esta forma, el ángulo definido tendrá un valor de 0° cuando el brazo está pegado al cuerpo, 90° al estar extendido hacia el frente y de 180° si se coloca hacia arriba.

Dentro del mensaje */body_angles* el elemento que define la información de este ángulo se denomina *RShoulderRotationAngle* para el brazo derecho y *LShoulderRotationAngle* para el izquierdo.

- Ángulo de apertura del codo

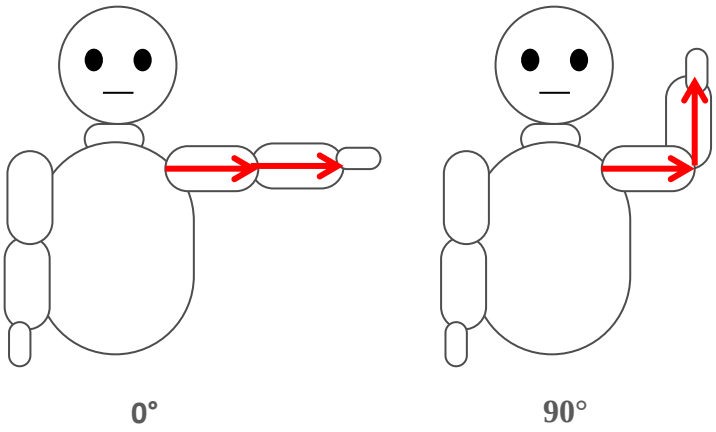


FIGURA 3-24: CÁLCULO DE LA APERTURA DEL CODO

Para establecer el valor del ángulo de apertura del codo se utiliza el vector que une el hombro con el codo y el que conecta el codo con la mano. Tras aplicar la fórmula trigonométrica se obtiene una apertura de 0° cuando el codo está extendido y de 90° cuando se dobla el codo, como está representado en la Figura 3-24.

Dentro del mensaje */body_angles* el elemento que define la información de este ángulo se denomina *RElbowOpeningAngle* para el brazo derecho y *LElbowOpeningAngle* para el izquierdo.

- Ángulo de rotación del codo

Obtener el ángulo de rotación del codo es más complejo que calcular el resto de los ángulos de los brazos. Esto es debido a que la posición de está articulación cambia cuando se gira el hombro, lo que provoca que los vectores elegidos para calcular el ángulo tengan que cambiar dependiendo de la posición del codo en cada momento. Debido a ello se ha seleccionado el vector que une los hombros para cuando el brazo está pegado al cuerpo, como se ve en la Figura 3-25

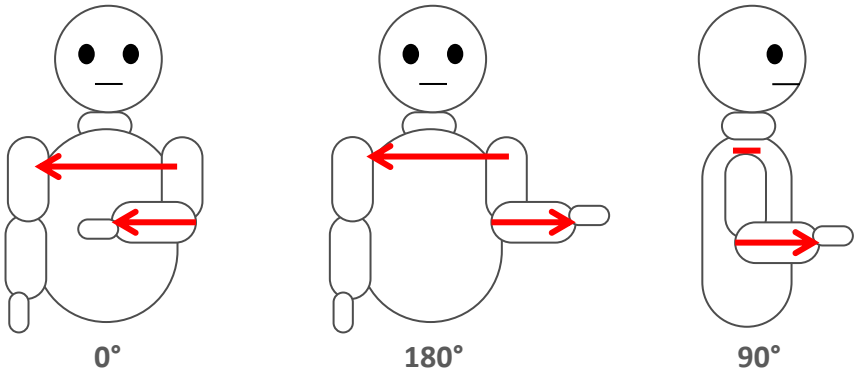


FIGURA 3-25: ÁNGULO DE ROTACIÓN DEL CODO CON EL BRAZO CERRADO

Utilizando estos dos vectores los ángulos resultantes cuando el brazo está pegado al cuerpo son 0° cuando el brazo se dobla hacia dentro, 90° si se tuerce hacia el frente y 180° si se abre hacia fuera del cuerpo. Observando las dos imágenes, se puede apreciar el porqué no es posible utilizar el mismo vector de referencia para determinar el ángulo cuando el brazo está cerrado o abierto, ya que si cuando el brazo esta abierto, se usara el vector que une los hombros, siempre se obtendría un ángulo de 90°. Por lo tanto, para averiguar el ángulo de rotación del codo con el brazo abierto se utilizará el vector que une el hombro con el lado de la cadera que tiene justo debajo.

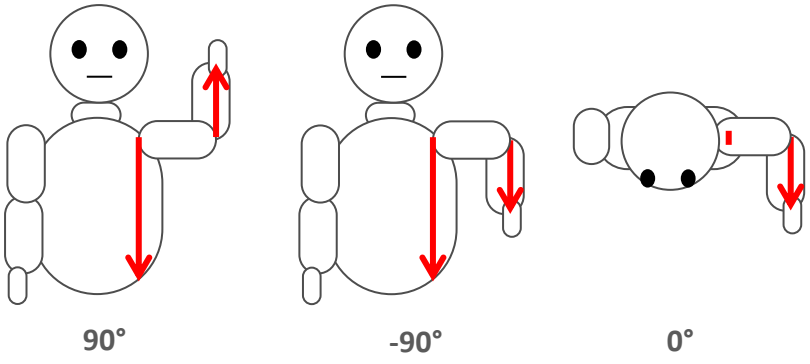


FIGURA 3-26: ÁNGULO DE ROTACIÓN DEL CODO CON EL BRAZO ABIERTO

A partir de estos vectores se obtienen los siguientes ángulos: 0° si se dobla el codo hacia delante, 90° si se dobla hacia arriba y -90° si se dobla hacia abajo.

Para que la transición de pasar de utilizar un vector a otro sea lo menos brusca posible se utiliza la siguiente fórmula:

$$RC = HC \times \frac{AH}{90°} + HH \times (1 - \frac{AH}{90°})$$

- RC ≡ Ángulo de rotación del codo.
- HC ≡ Ángulo de rotación del codo calculado mediante el vector que une el hombro a la cadera.
- HH ≡ Ángulo de rotación del codo calculado mediante el vector que une los hombros.
- AH ≡ Ángulo de apertura del hombro.

Aplicando esta fórmula se consigue que cuando el ángulo de apertura del hombro sea de 90°, se anule el segundo sumando, y por lo tanto solo se tenga en cuenta el ángulo calculado con el vector del hombro a la cadera. Y, por el contrario, también se logra que cuando el ángulo de apertura del hombro sea de 0° solo se utilice el ángulo obtenido mediante el vector que une los hombros. En el resto de casos, se consigue una combinación según la cual tendrá más peso un sumando que el otro según el brazo esté más o menos abierto. A continuación se presenta un ejemplo con un caso real.

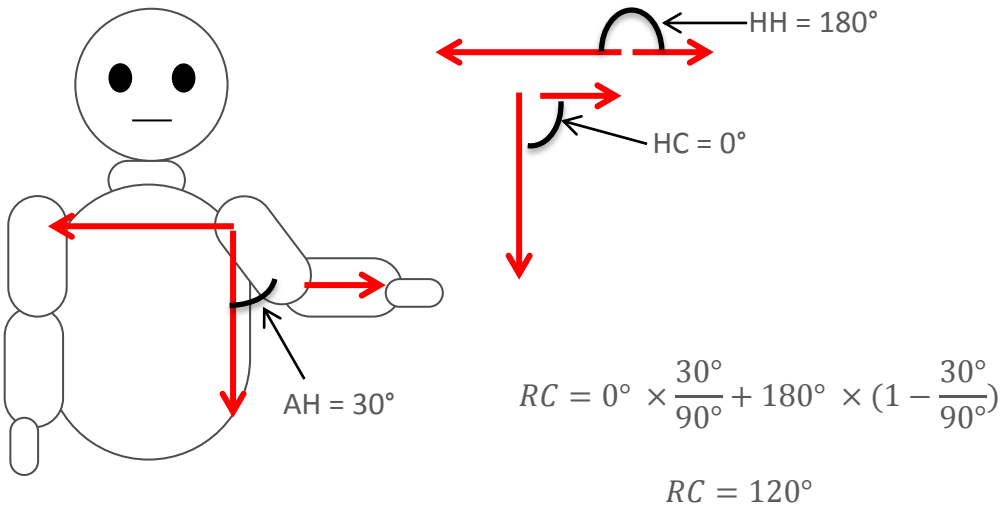


FIGURA 3-27: EJEMPLO DEL CÁLCULO DEL ÁNGULO DE ROTACIÓN DEL CODO

Dentro del mensaje */body_angles* el elemento que define la información de este ángulo se denomina *RElbowRotationAngle* para el brazo derecho y *LElbowRotationAngle* para el izquierdo.

- **Cálculo de la orientación de las manos**

Este proceso comienza cuando se recibe el mensaje de ***hand_interaction***, definido en *B.5 /hand1_fullcloud y /hand0_fullcloud*. Este mensaje contiene la nube de puntos que forma la mano del usuario. Realmente son dos mensajes, uno para cada mano, pero este apartado se centrará en el procesamiento de la mano derecha, que de todos modos, es equivalente al de la izquierda. Kinect no tiene la suficiente precisión para conocer con exactitud la posición de la mano, es capaz de detectar el lugar que ocupa en el espacio pero no su colocación ni su orientación. Más específicamente, determinar si una mano está abierta o cerrada, o si está mirando hacia arriba o hacia abajo no es en absoluto trivial. Existen algunas soluciones para este problema, como la que se ha presentado en el módulo ***hand_interaction***, que solo funciona bajo condiciones muy especiales, o la que se propone en [24], que tiene un gran coste computacional y es muy compleja de implementar. Para la realización de este proyecto, se ha optado por hacer que el operador tenga que mantener el puño cerrado con el pulgar extendido, lo que funciona como una flecha que señala la orientación que tiene la mano. Para la apertura y cierre de las manos se utilizarán órdenes de voz.

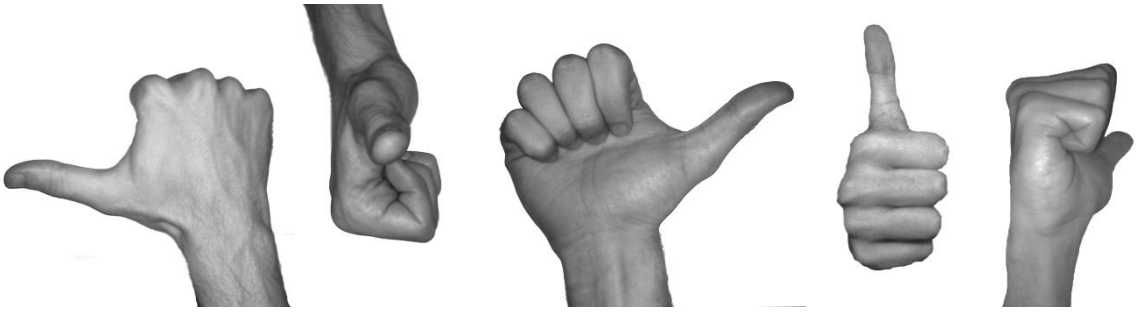


FIGURA 3-28: POSIBLES POSICIONES QUE PUEDE TOMAR LA MANO

Con el objetivo de mejorar la comprensión del modo en que se obtiene la posición de la mano se utilizará un ejemplo. En este ejemplo el usuario tendrá su mano en la posición que aparece en la imagen número 1, como se ha dicho anteriormente se debe mantener la mano cerrada con el pulgar extendido. Tanto el usuario como su entorno serán captados por Kinect, mediante el proceso de proyectar una malla de puntos infrarrojos que es captada por un sensor. Esta información le llegará al nodo ***hand_interaction*** que la analizará hasta obtener una nube de puntos para la mano izquierda y otra para la derecha. La nube de puntos de la mano derecha es la que aparece representada en la imagen número 2.

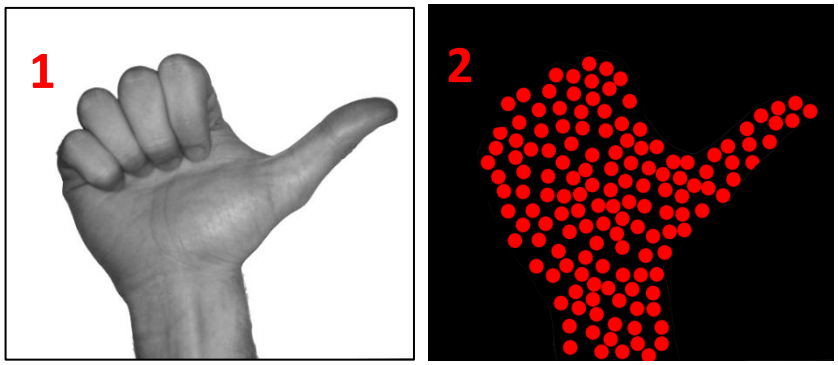


FIGURA 3-29: PROCESAMIENTO DE LA MANO 1 Y 2

Esta nube formará uno de los mensajes que ***hand_interaction*** envía a ***body_capture***. Cuando ***body_capture*** recibe la nube de puntos, lo primero que hace es aplicar un filtro para eliminar una gran cantidad de puntos que no aportan información relevante y ralentizan el procesamiento. El resultado sería algo similar a lo que se puede ver en la imagen número 3. Este filtro y los demás que se utilizarán para trabajar con la nube de puntos pertenecen al nodo ***pcl***, descrito anteriormente.

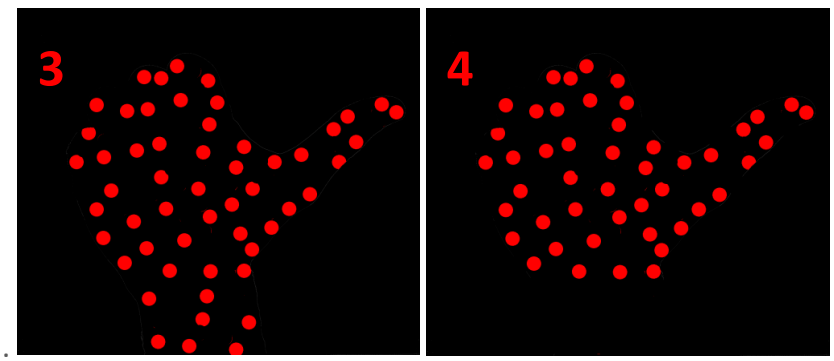


FIGURA 3-30: PROCESAMIENTO DE LA MANO 3 Y 4

A partir de la nube de puntos reducida, el siguiente paso que se llevará a cabo consiste en eliminar los puntos de la muñeca. Para realizar este proceso, se debe establecer cuales son los puntos de la muñeca mediante un análisis de la posición relativa de la mano con respecto a la del codo y esta información se consigue del mensaje recibido de ***skeletal_tracker***. Como aparece en la imagen número 8, la mano está encima del codo en este ejemplo, por lo tanto se puede determinar que los puntos de la muñeca serán los que están más abajo de la nube de puntos. Tras borrar estos puntos, se obtendría la imagen número 4.

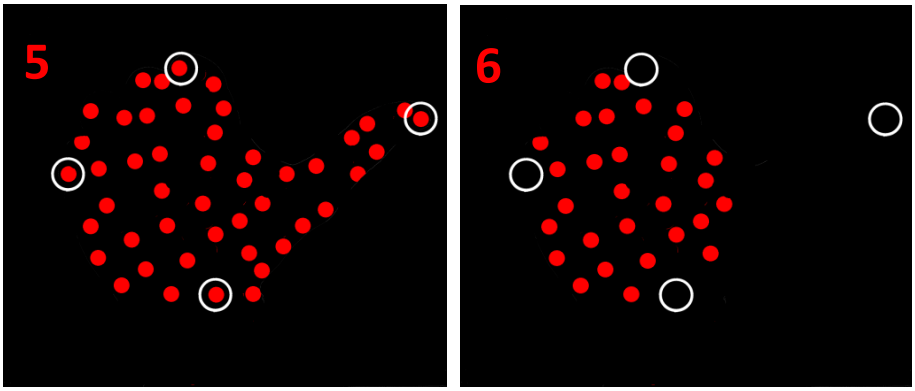


FIGURA 3-31: PROCESAMIENTO DE LA MANO 5 Y 6

Una vez obtenida la nube reducida tras la eliminación del ruido que suponían los puntos de la muñeca, comienza el proceso de detección de la posición de la mano. Para ello, se recorren todos los puntos en busca del más alto, el más bajo, el más a la izquierda y el más a la derecha, como se muestra en la imagen número 5. A continuación, se le aplicará a la nube un filtro que elimina aquellos puntos que tienen pocos vecinos, es decir, aquellos que no llegan a tener un mínimo de puntos a su alrededor. La consecuencia de aplicar este filtro es que se borrarán todos los puntos que forman el pulgar de la mano, manteniendo únicamente los que forman el puño, como se puede observar en la imagen número 6.

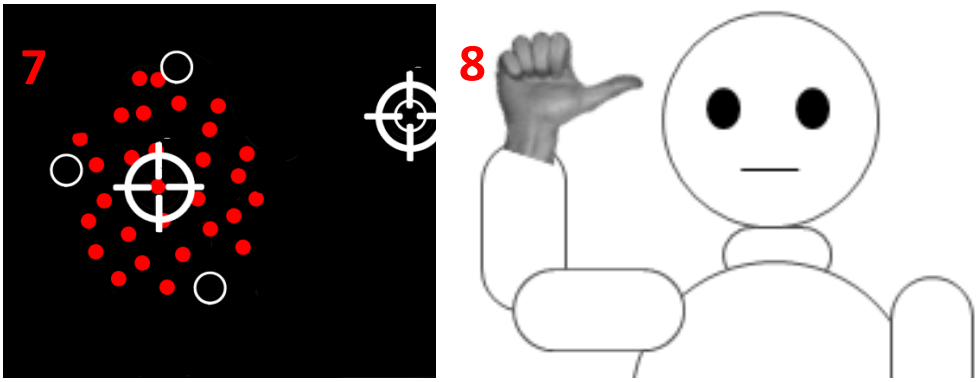


FIGURA 3-32: PROCESAMIENTO DE LA MANO 7 Y 8

Tras esto, se obtiene el centro de la nube de puntos resultante y se averigua cual es el punto más alejado entre los extremos encontrados antes, que coincidirá con el punto que representa el pulgar. En este caso, el extremo más alejado es el punto de la izquierda (izquierda según el usuario y derecha según el observador), como está representado en la imagen número 7. Una vez obtenido el centro de la mano y la situación del pulgar, se puede concluir la postura de la mano. A partir de esta, para calcular el ángulo de la muñeca se necesita, además, información de la situación del resto del brazo, en este ejemplo la posición del brazo se puede ver en la imagen número 8. El programa sigue un funcionamiento del tipo: si la mano está encima del codo, el codo no esta delante del hombro y el pulgar está orientado hacia la izquierda, entonces el ángulo de la muñeca será de 90°. A continuación se presentan otros ejemplos de este último paso:

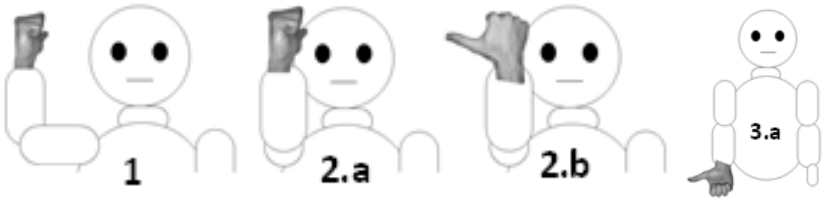


FIGURA 3-33: EJEMPLOS DE ÁNGULOS DE LA MUÑECA 1

1.- La mano está encima del codo, el codo no está delante del hombro y el dedo esta hacia atrás -> 0°

2.- La mano está encima del codo y el codo, delante del hombro:

a.- Dedo hacia atrás -> 90°

b.- Dedo hacia la derecha -> 0°

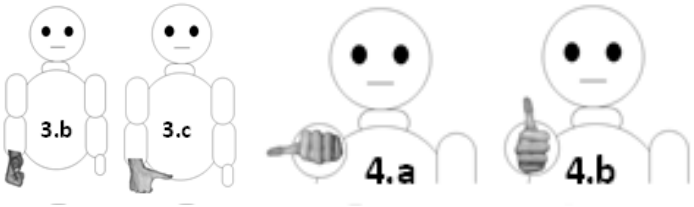


FIGURA 3-34: EJEMPLOS DE ÁNGULOS DE LA MUÑECA 2

3.- La mano debajo del codo y el codo no está delante del hombro:

a.- Dedo hacia la derecha -> 90°

b.- Dedo hacia delante-> 0°

c.- Dedo hacia la izquierda -> -90°

4.- La mano está delante del codo y el codo está delante del hombro:

a.- Dedo hacia la derecha -> 90°

b.- Dedo hacia arriba-> 0°

c.- Dedo hacia la izquierda -> -90°

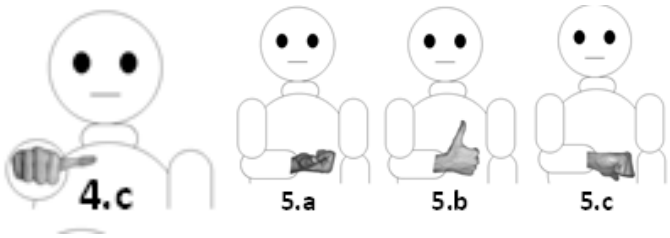


FIGURA 3-35: EJEMPLOS DE ÁNGULOS DE LA MUÑECA 3

5.- La mano está delante del codo y el codo está delante del hombro:

a.- Dedo hacia la delante -> 90°

b.- Dedo hacia arriba-> 0°

c.- Dedo hacia la detrás -> -90°

- **Otros elementos del mensaje body_angles**

A parte de los ángulos de los brazos, ***body_capture*** también envía la posición de las manos, los hombros y la cabeza, a través del mensaje */body_angle*. Para obtener está información no se lleva a cabo ningún procesamiento avanzado, se realiza una adaptación de los datos que se reciben de ***skeletal_tracker***. La posición de las manos será transportada por los elementos *RHandPosition* y *LHandPosition*; la de los hombros por *RShoulderPosition* y *LShoulderPosition*, y la de la cabeza por Head.

3.3.3.2.6. RECOGNIZER

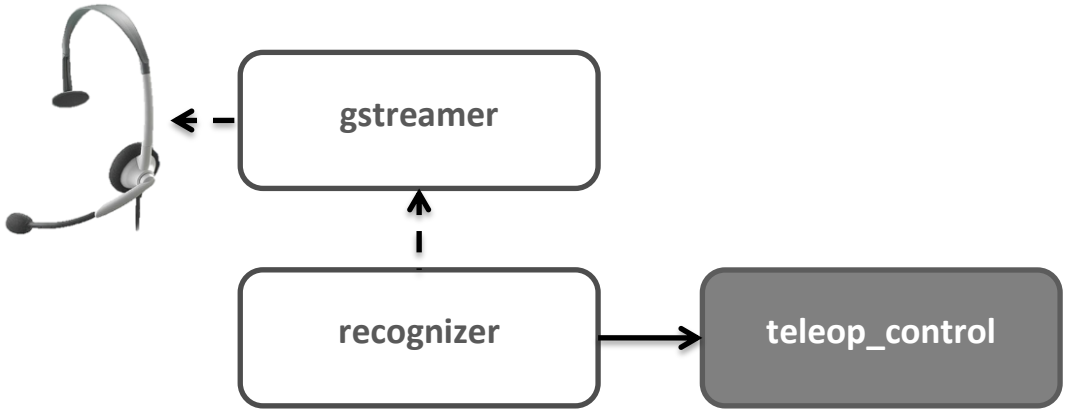


FIGURA 3-36: ESQUEMA DEL NODO RECOGNIZER

Este nodo ha sido creado en la Universidad de Albany, y se encarga de conectar *Pocket Sphinx* con ROS. *Pocket Sphinx* es un programa que se engloba dentro de CMU Sphinx, un grupo de sistemas de reconocimiento de voz desarrollados por la *Universidad de Carnegie Mellon*. *Pocket Sphinx* utiliza *GStreamer* para acceder a los datos recogidos por el micrófono. *GStreamer* es un *framework* para el trabajo con contenidos multimedia, libre y multiplataforma.

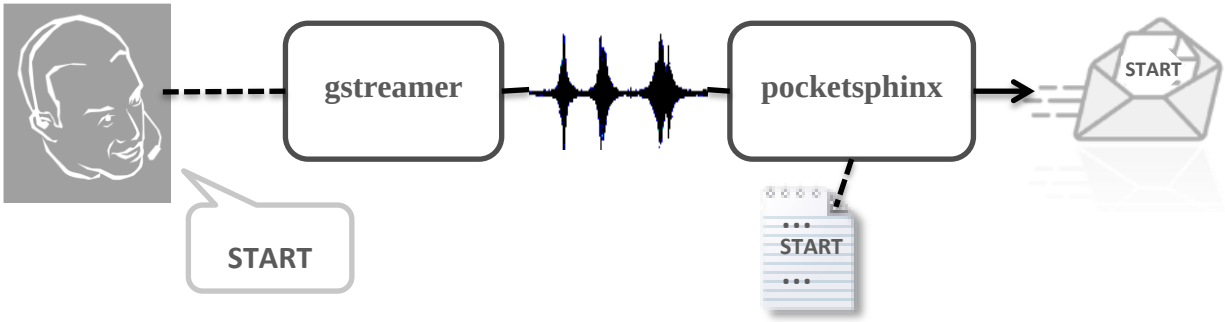


FIGURA 3-37: ESQUEMA DE FUNCIONAMIENTO DEL NODO RECOGNIZER

En resumen, este nodo permite definir un conjunto de palabras para que sean reconocidas. Estas palabras vienen especificadas en dos archivos cuya localización se debe pasar como parámetro del programa y que se describen en el anexo D. Ejecución. Para proceder al reconocimiento, ***recognizer*** recibe la señal audio del micrófono a través de *GStreamer* y la analiza para buscar si existen coincidencias con alguna de las palabras que se deben reconocer. . Una vez que se detecta una palabra, se crea el mensaje descrito en *B.6 recognizer/output*, que será recibido por el nodo ***teleop_control***.

En este sistema, las palabras que se deben capturar son:

- "*start*": para iniciar la teleoperación.
- "*finish*": para detener la teleoperación.
- "*left*": para abrir o cerrar la mano izquierda.
- "*right*": para abrir o cerrar la mano derecha.

3.3.3.3. CAPA DE CONTROL

En este apartado se describe el funcionamiento del nodo *teleop_control*, que hace de intermediario entre el operador y el robot.

3.3.3.3.1. TELEOP_CONTROL

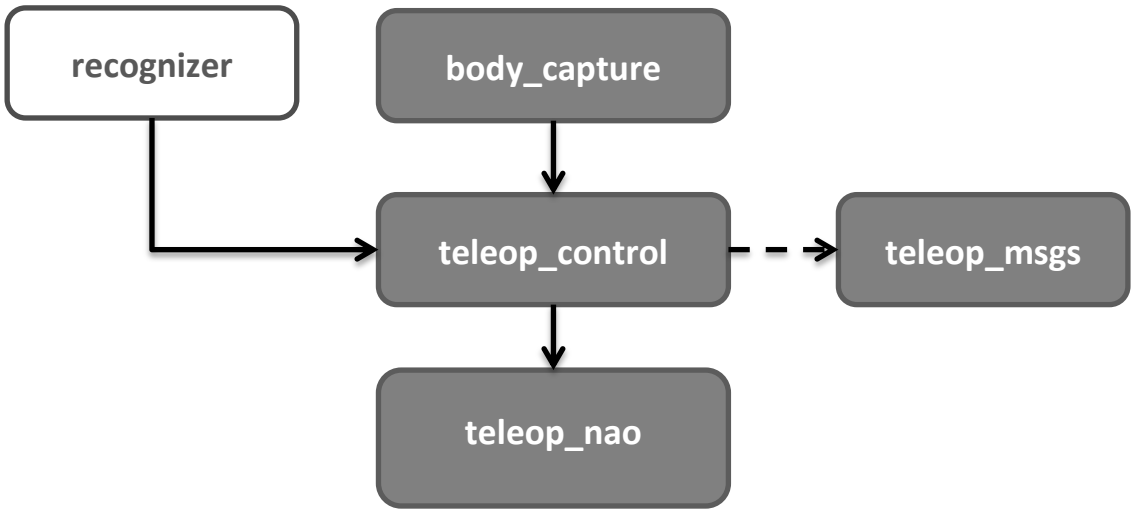


FIGURA 3-38: ESQUEMA DEL NODO TELEOP_CONTROL

Este nodo funciona de intermediario entre el usuario y el robot. Recibe la información de las acciones del teleoperador, las analiza, la interpretas y las adapta para enviarlas al robot. Más concretamente sus funciones son:

- **Adaptación de los ángulos del usuario**

Cuando ***teleop_control*** recibe el mensaje */body_angle* con los ángulos descritos en *B.7 /bodyAngle*, se realiza una pequeña modificación de estos ángulos. Está modificación consiste en escalar e invertir algunos ángulos. Por ejemplo: el ángulo de apertura del hombro se considera de 90° cuando el brazo está abierto, tanto si es el izquierdo como el derecho; sin embargo, para el robot NAO la apertura total del brazo izquierdo son 90° y del brazo derecho son -90°, por esto es necesario invertir el ángulo de apertura del hombro derecho. Otro ejemplo es el ángulo de rotación del hombro que calculado en el teleoperador es de 0° cuando el brazo está hacia abajo, de 90°

cuando está hacia el frente y 180° cuando está hacia arriba, mientras que en el NAO es de 90°, 0° y -90°, respectivamente.

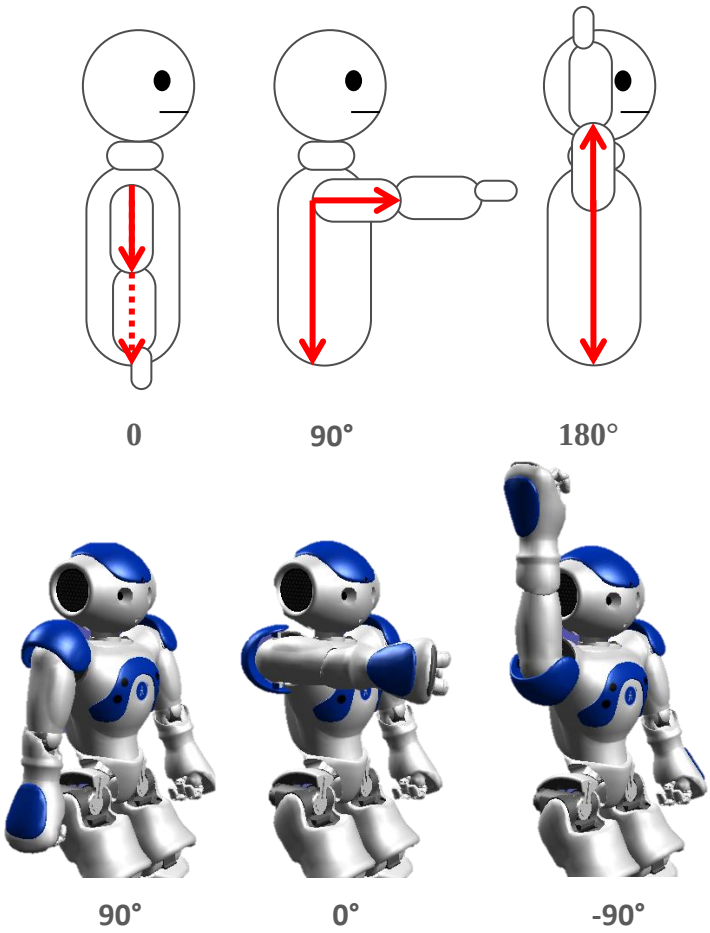
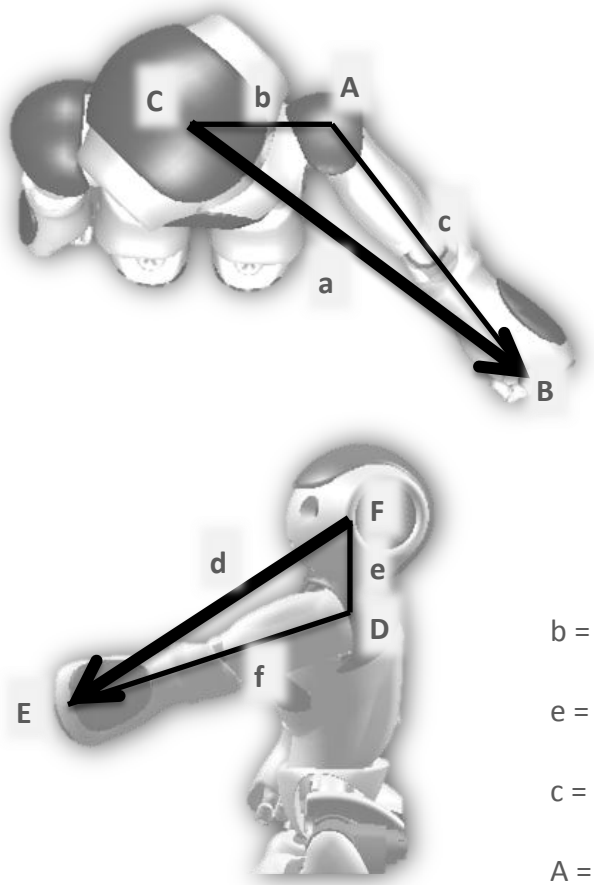


FIGURA 3-39: COMPARACIÓN ENTRE ÁNGULOS DEL OPERADOR Y ÁNGULOS DEL NAO

También se lleva a cabo la división del mensaje */body_angle*, que empaqueta todos los ángulos del cuerpo del usuario, en multitud de mensajes del tipo */joint_angle*, definidos en *B.11 Conjunto de mensajes del tipo teleop_msgs/joint_angle*. Esto se realiza con el objetivo de intentar que el robot pueda manejar estos mensajes de forma individual, ya que si recibiera todos los ángulos unidos se vería obligado a procesarlos todos de una vez, creando una ventana de tiempo en la que estaría ocupado, no pudiendo realizar otra función.

- **Seguimiento de las manos con la cámara**

Cuando el operador desea realizar alguna operación con las manos del robot, necesita recibir información del estado en que estas se encuentran. Para ello se ha optado por realizar un seguimiento de las manos con la cámara del robot, cuando estas se elevan para realizar alguna acción. Está función se encarga de calcular cuanto debe rotar y cuanto se debe inclinar la cabeza para que la cámara las siga. Para obtener dicho ángulo se utiliza el teorema del seno y el teorema del coseno, más concretamente, la fórmula clásica para calcular los elementos de un triangulo del que se conocen dos de sus lados y el ángulo del vértice que los une.



$$a^2 = b^2 + c^2 - 2 \cdot \mathrm{b} \cdot \mathrm{c} \cdot \cos A$$

$$\mathrm{C} = arcsin \frac{c \cdot \sin A}{a}$$

$$d^2 = e^2 + f^2 - 2 \cdot \mathrm{e} \cdot \mathrm{f} \cdot \cos D$$

$$\mathrm{F} = arcsin \frac{f \cdot \sin D}{d}$$

b = 11 cm

e = 7 cm

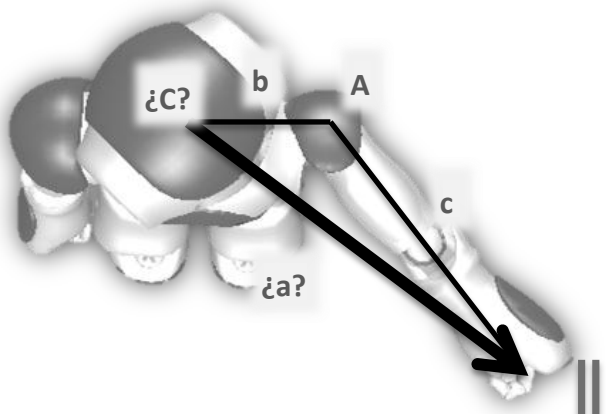
c = f = 20 cm

A = ángulo de apertura del hombro + 90°

D = 180° - ángulo de rotación del hombro

FIGURA 3-40: CÁLCULO DE LA ROTACIÓN Y LA INCLINACIÓN DE LA CABEZA DEL ROBOT

En este caso, el triángulo del que se necesita conocer las características es el formado por la cabeza, el hombro y la mano. La longitud del brazo (c y f) y la distancia de la cabeza al hombro (b y e) son conocidas y constantes. El ángulo que forman viene dado por el ángulo de apertura del hombro (A), si estamos calculando la rotación de la cabeza, o por el ángulo de rotación del hombro (B), si estamos calculando la inclinación. Estos ángulos necesitan un ligero ajuste, como está representado en la Figura 3-40: Cálculo de la rotación y la inclinación de la cabeza del robot. A partir de estos datos y aplicando las fórmulas, se consigue el ángulo de rotación de la cabeza (C) necesario para que la cámara apunte a la mano. A continuación se muestra un ejemplo para el cálculo de la rotación de la cabeza:



A = 45° + 90° = 135°

b = 11 cm

c = 20 cm

$$a = \sqrt{b^2 + c^2 - 2 \cdot \mathrm{b} \cdot \mathrm{c} \cdot \cos A}$$

$$a = \sqrt{11^2 + 20^2 - 2 \cdot 11 \cdot 20 \cdot \cos 135°}$$

$$a = 28{,}8\ cm$$

$$\mathrm{C} = arcsin \frac{c \cdot \sin A}{a}$$

$$\mathrm{C} = arcsin \frac{20 \cdot \sin 135°}{28{,}8}$$

$$\mathbf{C = 29{,}4°}$$

FIGURA 3-41: EJEMPLO DEL CÁLCULO DEL ÁNGULO DE ROTACIÓN DE LA CABEZA

Cabe mencionar, que la cabeza solo girará cuando las manos del robot se eleven para realizar alguna acción, el resto del tiempo la cabeza mirará al frente para que la cámara detecte lo que hay delante del robot. Además, siempre se dará prioridad al seguimiento de la mano derecha, siguiendo a la mano izquierda solo cuando la derecha no esté elevada.

- **Captura de comandos de voz**

Debido a los problemas que tiene Kinect en lo referente a la detección de las manos, ha sido necesario desarrollar una manera alternativa de realizar la apertura y cierre de las manos del robot. Para solucionar este problema, se decidió crear un sistema mixto que utilizará tanto la posición del operador, como su voz.

Cuando el usuario pronuncia alguna palabra que corresponde con uno de los comandos de voz, el nodo ***recognizer*** lo detecta y envía un mensaje con dicha palabra como contenido. Este será recibido por ***teleop_control***. Dependiendo de la orden de voz, se realizará alguna de las acciones siguientes:

- "*START*": cuando el usuario pronuncia la orden "*start*" se comienza la teleoperación, iniciando el envío de mensajes al robot para que este comience a moverse. Además se memoriza la posición inicial de la cabeza del usuario para que se pueda detectar cuando se ha desplazado.
- "*FINISH*": si la orden que invoca el teleoperador es "*finish*" se detiene el envío de mensajes al robot, provocando que se detenga el proceso de teleoperación.
- "*LEFT*": con el comando "*left*" la mano izquierda del robot cambia de estado, pasando de abierta a cerrada o viceversa.
- "*RIGHT*": con el comando "*right*" la mano derecha del robot cambia de estado, pasando de abierta a cerrada o viceversa.

En todos los casos se imprime por pantalla un mensaje que informa al usuario de que su orden ha sido recibida y se hace que el robot emita por sus altavoces un mensaje para que las personas que están a su alrededor conozcan el estado del proceso de teleoperación.

- **Transmisión de la orden de andar y girar**

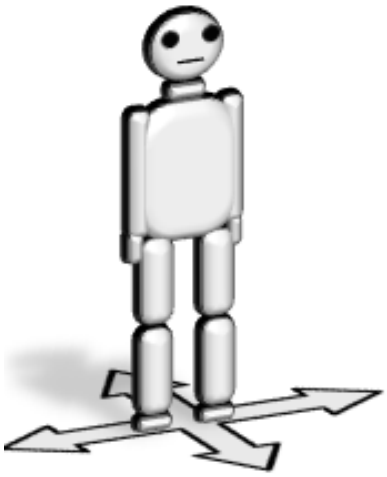


FIGURA 3-42: POSIBLES DIRECCIONES PARA LA ORDEN DE ANDAR

Como se ha descrito anteriormente, cuando el usuario pronuncia la orden "*start*" para dar comienzo a la teleoperación, se almacena la posición inicial de la cabeza del usuario. De esta manera, cuando se detecta que la posición actual de la cabeza difiere mucho de la inicial, se deduce que el usuario ha dado un paso. Si se ha dado un paso hacia al frente, hacia atrás, hacia la izquierda o hacia la derecha, el robot NAO comenzará a andar en la misma dirección del paso. Para calcular el giro se comprueba la diferencia entre la posición de los hombros: si ambos hombros están alineados frente a Kinect el robot no girará, si el izquierdo está más adelantado el robot girará hacia su derecha y si es el derecho el que se encuentra más cerca del sensor el robot girará hacia su izquierda.

3.3.3.4. CAPA ROBOT

En este punto se detallaran las características de los nodos encargados de llevar a cabo la comunicación directa con el robot.

3.3.3.4.1. TELEOP_NAO

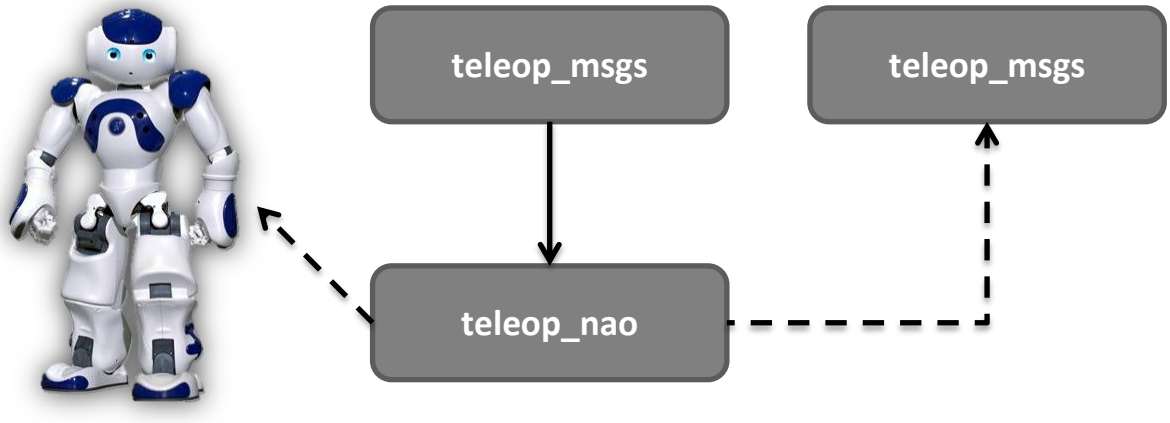


FIGURA 3-43: ESQUEMA DEL NODO TELEOP_NAO

Este nodo es el encargado de comunicarse directamente con el robot NAO para transmitirle las órdenes que recibe de ***teleop_control***. Para realizar está comunicación el nodo utiliza *NAOqi*, un software intermediario que se ejecuta dentro del propio robot y que provee de los métodos necesarios para acceder y manejar todos los elementos del robot. Por ejemplo, a través de *NAOqi* se puede controlar el movimiento del robot, accediendo a cada articulación por separado o mediante funciones de más alto nivel como la que hace al robot andar. También permite el manejo de los aspectos relacionados con el audio y el video, como: la detección de sonido, el reconocimiento de voz, el habla, el acceso a las cámaras, reconocimiento facial y de imágenes, etcétera. *NAOqi* está formado por distintos módulos que empaquetan un conjunto de funciones, en ***teleop_nao*** se utilizan los módulos *ALMotion*, para mover las articulaciones y hacer andar al robot, y *ALTextToSpeech*, para que el robot hable. Al iniciar ***teleop_nao***, se establece una conexión con estos dos módulos de *NAOqi* y se mantiene a la espera de recibir mensajes. Cuando llega un mensaje, por ejemplo con la información sobre el ángulo de apertura del codo, ***teleop_nao*** llamará a la función encargada de mover una articulación del módulo

ALMotion y se volverá a quedar a la espera del siguiente mensaje. En lugar del ángulo de una articulación, también puede recibir mensajes sobre como debe andar el robot o frases que tiene que pronunciar, en cuyo caso invocará la función adecuada dependiendo del caso. En resumen, este nodo no realiza ningún procesamiento de los datos, funcionando únicamente como conector entre el robot y el resto de sistema.

3.3.3.4.2. NAO_VISION

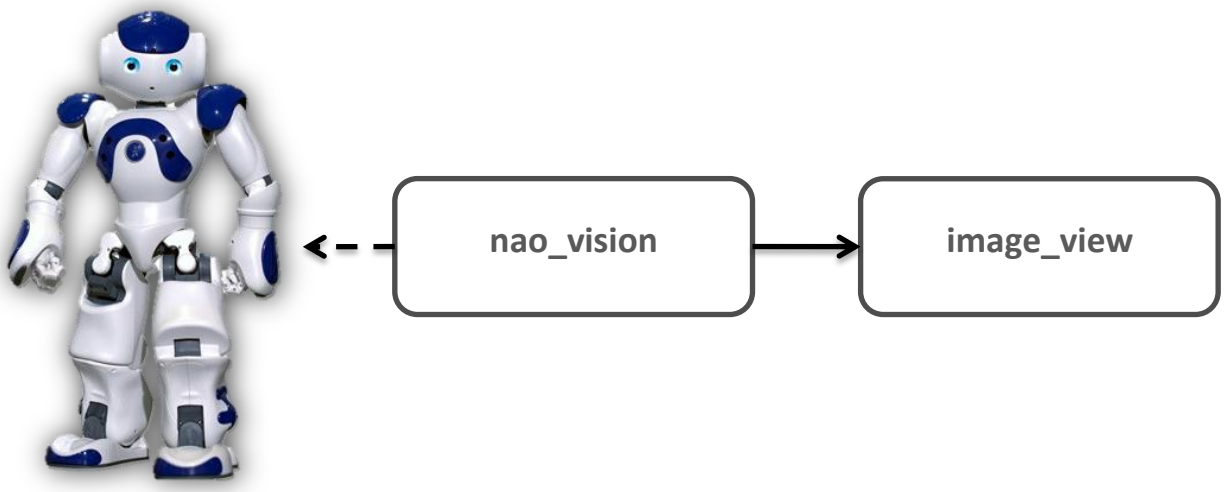


FIGURA 3-44: ESQUEMA DEL NODO NAO_VISION

La función de este nodo es conectarse al robot NAO, recibir las imágenes que este está tomando con su cámara inferior y adaptarlos a un mensaje de tipo */image* que se pueda enviar a través de ROS. Posteriormente, este mensaje lo recoge el nodo *image_view*, que se encarga de emitir por la pantalla el contenido del mensaje y, de esta manera, permitir al teleoperador observar lo que el robot está viendo a través de su cámara.

4. EVALUACIÓN

En este apartado se detallará como se llevó a cabo la evaluación del proyecto. Para realizarla, se contó con la participación de cinco usuarios que no tenían ningún tipo de relación con el proyecto y que tuvieron que efectuar varios experimentos mientras manejaban el robot. Estos experimentos consistían en una serie de tareas de dificultad variable que tenían como objetivo poner a prueba las capacidades del sistema. Tras su realización, los usuarios tuvieron que responder a un cuestionario en el que se les preguntaba sobre el resultado de los experimentos y su opinión acerca del funcionamiento de la aplicación. Los objetivos de la evaluación son:

- Comprobar la capacidad del sistema para ser utilizado en la realización tareas sencillas.
- Observar hasta que punto el sistema no necesita un aprendizaje previo para su manejo.
- Conocer las sensaciones y opiniones que tienen los usuarios durante la utilización de la aplicación.
- Recoger sugerencias de los usuarios para que puedan ser aplicadas en trabajos futuros.

4.1. DESCRIPCIÓN DE LOS EXPERIMENTOS

A continuación se describirán los cuatro experimentos que fueron llevados a cabo por los usuarios para evaluar el proyecto. Previamente a su realización, se les proporcionó una breve explicación del funcionamiento del sistema para que pudieran comenzar a realizar las tareas, no obstante, estas tienen una dificultad incremental que permita al usuario familiarizarse poco a poco con la aplicación. Todos los experimentos se llevaron a cabo de manera remota, es decir, el único contacto que el evaluador tenía con el entorno del robot venía dado por la información que le proporcionaba la cámara de este. La evaluación fue realizada de manera individual por cada usuario.

4.1.1. EXPERIMENTO 1

Este primer experimento consiste en trasladar al robot NAO a lo largo de un pasillo, hasta llegar a la puerta del laboratorio donde el usuario esta realizando la teleoperación. En su posición inicial el robot estará girado con respecto a la dirección que deberá tomar, como se observa en la siguiente figura.

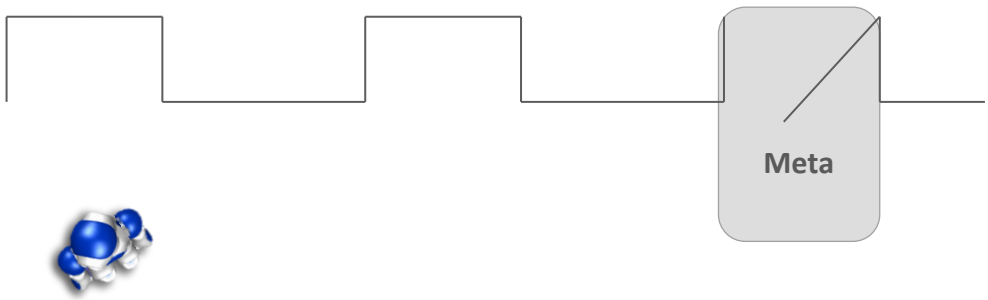


FIGURA 4-1: ESQUEMA DEL EXPERIMENTO 1

El objetivo de este experimento es comprobar la capacidad que tiene el sistema para hacer que el usuario conozca la situación del robot en el entorno remoto y permitirle moverlo hacia una posición de destino.

4.1.2. EXPERIMENTO 2

El experimento número 2 consiste en la resolución de un circuito formado por una serie de obstáculos. La forma que tendrá dicho circuito será la que aparece representada en la siguiente figura. Como se puede observar, el usuario tendrá que utilizar todos los desplazamientos del robot que el sistema pone a su disposición: caminar, desplazamientos laterales y giros. De esta manera irá sorteando los obstáculos hasta llegar a la zona de meta, contando únicamente con la información que reciba a través de la cámara del robot.

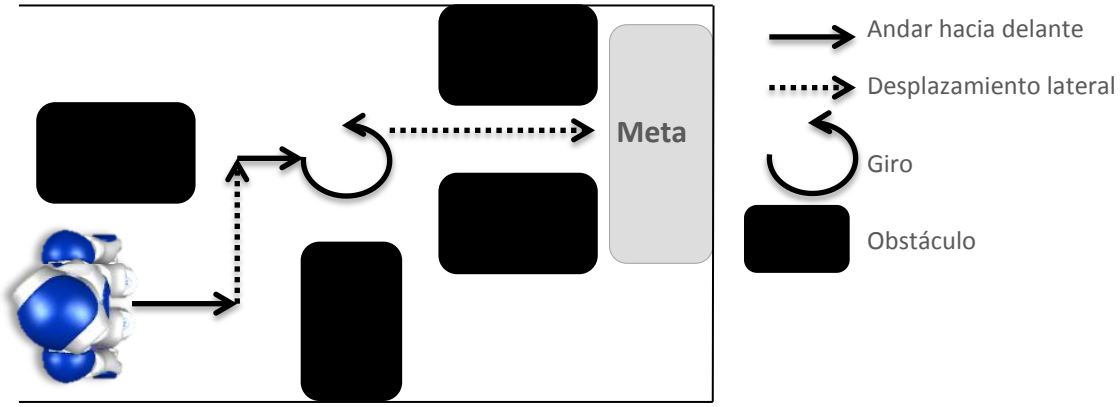


FIGURA 4-2: ESQUEMA DEL EXPERIMENTO 2

Con este experimento se pretende comprobar que el control se puede llevar a cabo con la precisión suficiente para mover al robot por este tipo de espacios cerrados.

En el siguiente enlace se puede ver una versión simplificada de este experimento.

http://www.youtube.com/watch?v=Nek8WqwfeKE&feature=plcp

4.1.3. EXPERIMENTO 3

En este tercer experimento el usuario evaluador deberá mover al robot hacia una persona que le hará entrega de un objeto. Para recogerlo, el robot tendrá que extender el brazo y abrir y cerrar la mano para poder sujetarlo. Posteriormente, el usuario dirigirá al robot hasta una caja cercana, depositando en ella el objeto. El objetivo del experimento es poner a prueba el movimiento de los brazos y manos del robot durante la realización de una tarea sencilla en colaboración con un humano.

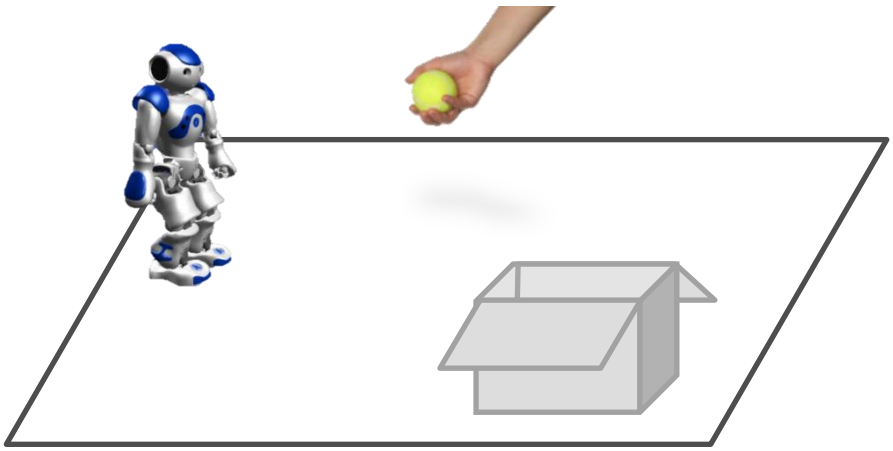


FIGURA 4-3: ESQUEMA DEL EXPERIMENTO 3

En el siguiente enlace se puede ver un video con la realización de este experimento.

http://www.youtube.com/watch?v=mAlOxQkeUbk&feature=plcp

4.1.4. EXPERIMENTO 4

El cuarto experimento consiste en que el usuario consiga hacer que el robot recoja un objeto situado cerca de su posición inicial. Debido a las características técnicas del robot NAO, más concretamente, a las características de su mano, se ha optado porque el objeto sea de tela y con arrugas, ya que, el movimiento de pinzas que hacen sus dedos al cerrar la mano necesita una precisión milimétrica para poder coger un objeto más sólido y pequeño. Este experimento sirve para comprobar la precisión del sistema a la hora de mover los brazos del robot y para observar como funciona el seguimiento de la mano llevado a cabo por la cámara de la cabeza del robot.

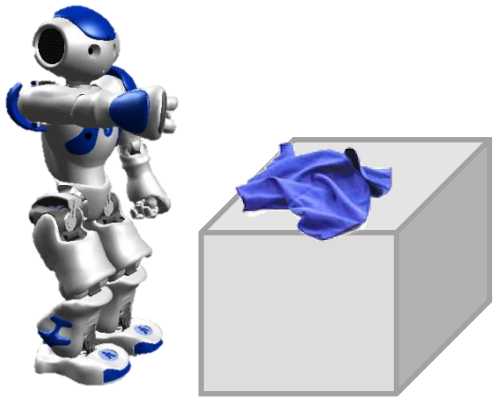


FIGURA 4-4: ESQUEMA DEL EXPERIMENTO 4

4.2. CUESTIONARIO DE EVALUACIÓN

A continuación se presenta el cuestionario que tuvieron que rellenar los participantes en los experimentos:

Evaluación nao_kinect

A continuación se presenta el cuestionario de evaluación del sistema de teleoperación nao_kinect. Después de que el usuario haya realizado una serie de experimentos orientados a poner a prueba los distintos aspectos del proyecto, deberá contestar estas preguntas para conocer como se han desarrollado y cuales han sido sus sensaciones durante la realización de las distintas pruebas..
*Obligatorio

Nombre *

Apellidos *

Correo electrónico *

Experimento 1: Desplazamiento por espacios abiertos.

Las siguientes preguntas se refieren al primer experimento, que consiste en desplazarse por un pasillo del Edificio Sabatini de la Universidad Carlos III de Madrid, hasta llegar a la sala donde se encuentra el operario realizando la teleoperación.

1.-¿Ha alcanzado el objetivo? *

- Si
- No

2.-Si ha respondido No en la pregunta anterior indique las razones:

3.-¿Se ha perdido? *

- Si
- No

4.-Si se ha perdido, indique cuantas veces:

5.-Si se ha perdido, indique las razones:

6.-¿Cómo de difícil le ha resultado el experimento? *

	1	2	3	4	5	
Muy fácil	◯	◯	◯	◯	◯	Muy difícil

Experimento 2: Desplazamiento en espacios cerrados.

La siguientes preguntas se refieren al segundo experimento, que consiste en resolver un laberinto , sin chocar con obstáculos. Se creará un pequeño circuito con obstáculos en el que el robot deberá realizar movimientos precisos para alcanzar la salida, alternando giros con pequeños desplazamientos frontales y laterales

7.-¿Cuántos intentos ha necesitado para terminar el circuito? *

Se considera un intento fallido cuando no se ha podido completar el circuito de principio a fin y sin chocar con algún obstáculo.

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ Más de 3
- ◯ No lo he terminado

8.-Si ha tenido intentos fallidos, indique las razones:

9.-¿Cómo de precisos le han resultado los desplazamientos para moverse por este tipo de espacios?

	1	2	3	4	5	
Nada precisos	◯	◯	◯	◯	◯	Muy precisos

10.-¿Cómo de difícil le ha resultado el experimento? *

	1	2	3	4	5	
Muy fácil	◯	◯	◯	◯	◯	Muy difícil

Experimento 3: Interacción con humanos para la manipulación de objetos pequeños.

Las siguientes preguntas se refieren al tercer experimento, que consiste en que el robot deberá desplazarse desde una posición inicial hasta situarse frente a una persona que le dará un objeto, que el robot deberá coger y llevar hasta una caja donde lo depositará.

11.-¿Cuántos intentos ha necesitado para realizar la tarea con exito? *

Se considera intento fallido cuando no se ha podido realizar el proceso completo de ir hacia la persona, recibir el objeto, ir a la caja y depositar el objeto sin interrupciones.

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ Más de 3
- ◯ No la he podido realizar

12.-¿Cuál de estas razones han provocado los intentos fallidos?

- ☐ El robot se ha perdido al acercarse a la persona o al ir hacia la caja.
- ☐ El objeto se ha caído en el momento del intercambio.
- ☐ El objeto se ha caído en el camino hacia la caja.
- ☐ No se ha acertado al depositar el objeto en la caja.
- ☐ El robot ha chocado con algún obstáculo.
- ☐ Otro: _____

13.-¿Cómo de difícil le ha resultado el experimento? *

	1	2	3	4	5	
Muy fácil	◯	◯	◯	◯	◯	Muy difícil

Experimento 4: Manipulación de objetos.

Las siguientes preguntas se refieren al cuarto experimento, que consiste en coger un objeto de tela con la mano del robot. El objeto estará situado en una caja cerca de la posición inicial del robot.

14.-¿Cuántos intentos ha necesitado para coger el objeto? *

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ Más de 3
- ◯ No lo he podido coger

15.-¿Cuál de estas razones han provocado los intentos fallidos?

- ☐ El robot se ha perdido al acercarse a la caja del objeto.
- ☐ El robot ha chocado con algún obstáculo.
- ☐ El objeto se ha caído al intentar cogerlo.
- ☐ El objeto se ha caído una vez que el robot lo habia agarrado.
- ☐ Otro: _____

16.-¿Cómo de difícil le ha resultado el experimento? *

	1	2	3	4	5	
Muy fácil	◯	◯	◯	◯	◯	Muy difícil

Experimentos 3 y 4: Preguntas comunes

En este apartado se realizarán preguntas que afectan tanto al experimento 3 como al 4, referentes al control de los brazos.

17.-¿Cúal de los dos objetos usados en los experimento le ha resultado más fácil de manipular? *

- El pequeño objeto amarillo del experimento 3.
- El objeto de tela del experimento 4.

18.-¿Cómo de complejo le ha resultado el control de los brazos? *

	1	2	3	4	5	
Muy sencillo						Muy complejo

19.-¿Cómo de preciso le ha resultado el control de los brazos? *

	1	2	3	4	5	
Nada preciso						Muy preciso

20.-¿Cómo de preciso le ha resultado el control de la posición de las manos, es decir, el giro de las muñecas? *

	1	2	3	4	5	
Nada preciso						Muy preciso

21.-¿Cómo de útil le ha resultado el seguimiento de las manos que realiza la cámara del robot? *

	1	2	3	4	5	
Nada útil						Muy útil

22.-¿Han funcionado correctamente los comandos de voz para abrir y cerrar las manos? *

- Sí
- No

23.-Si ha respondido "No", indique las razones: *

Preguntas generales.

24.-¿Cómo de cómodo le ha resultado el sistema de teleoperación? *
Conteste según las sensaciones que ha experimentado durante el desarrollo de los experimentos

	1	2	3	4	5	
Muy incómodo	◯	◯	◯	◯	◯	Muy cómodo

25.-¿Cómo de rápido se ha familiarizado con los controles? *
Conteste según las sensaciones que ha experimentado durante el desarrollo de los experimentos

	1	2	3	4	5	
Todavía no lo he conseguido	◯	◯	◯	◯	◯	Prácticamente instantáneo

26.-¿Cómo de integrado se ha sentido con el robot ? *
Conteste según las sensaciones que ha experimentado durante el desarrollo de los experimentos

	1	2	3	4	5	6	7	8	9	10	
No he sentido que el robot me obedeciera	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	La respuesta del robot ha sido perfecta

27.-¿Ha necesitado alguna indicación directa para completar los experimentos? *

- ◯ Si
- ◯ No

28,-Si ha respondido Si en la pregunta anterior, especifiqué que tipo de indicaciones:

29.-¿Qué aspectos modificaría o mejoraría de la aplicación?

30.-Otras observaciones:

4.3. RESULTADOS DEL CUESTIONARIO

En este apartado se muestra el resultado de los cuestionarios entregados a los participantes en la evaluación tras la realización de los experimento.

Pregunta	Usuario 1	Usuario 2	Usuario 3	Usuario 4	Usuario 5
Experimento 1					
1.-¿Ha alcanzado el objetivo?	Si	Si	Si	Si	Si
3.-¿Se ha perdido?	No	No	No	No	No
6.-¿Cómo de fácil le ha resultado el experimento?	1	1	1	1	1
Experimento 2					
7.-¿Cuántos intentos ha necesitado para terminar el circuito?	1	2	3	2	1
9.-¿Cómo de precisos le han resultado los desplazamientos para moverse por este tipo de espacios?	4	4	4	2	2
10.-¿Cómo de fácil le ha resultado el experimento?	2	2	2	2	3
Experimento 3					
11.-¿Cuántos intentos ha necesitado para realizar la tarea con exito?	1	2	No la he podido realizar	No la he podido realizar	1
12.-¿Cuál de estas razones han provocado los intentos fallidos?		El objeto se ha caído en el camino hacia la caja.	No se ha acertado al depositar el objeto en la caja.	No se ha acertado al depositar el objeto en la caja.	
13.-¿Cómo de fácil le ha resultado el experimento?	2	3	2	4	2
Experimento 4					
14.-¿Cuántos intentos ha necesitado para coger el objeto?	2	No lo he podido coger	No lo he podido coger	No lo he podido coger	No lo he podido coger
15.-¿Cuál de estas razones han provocado los intentos fallidos?	El objeto se ha caído al intentar cogerlo.	El objeto se ha caído al intentar cogerlo.	No detectaba bien el levantamiento lateral del brazo	El objeto se ha caído al intentar cogerlo.	El objeto se ha caído al intentar cogerlo.
16.-¿Cómo de fácil le ha resultado el experimento?	4	5	5	3	4

TABLA 4-1: RESULTADOS DEL CUESTIONARIO 1

Experimentos 3 y 4					
17.-¿Cúal de los dos objetos usados en los experimento le ha resultado más fácil de manipular?	El pequeño objeto amarillo del experimento 3.	El pequeño objeto amarillo del experimento 3.	El pequeño objeto amarillo del experimento 3.	El pequeño objeto amarillo del experimento 3.	El pequeño objeto amarillo del experimento 3.
18.-¿Cómo de complejo le ha resultado el control de los brazos?	3	4	2	2	1
19.-¿Cómo de preciso le ha resultado el control de los brazos?	2	2	3	3	2
20.-¿Cómo de preciso le ha resultado el control de la posición de las manos, es decir, el giro de las muñecas?	4	4	1	4	2
21.-¿Cómo de útil le ha resultado el seguimiento de las manos que realiza la cámara del robot?	4	4	5	4	5
22.-¿Han funcionado correctamente los comandos de voz para abrir y cerrar las manos?	No	No	Sí	No	Sí
Preguntas generales					
24.-¿Cómo de cómodo le ha resultado el sistema de teleoperación?	4	4	5	3	5
25.-¿Cómo de rápido se ha familiarizado con los controles?	5	5	4	4	5
26.-¿Cómo de integrado se ha sentido con el robot ?	7	6	8	7	7
27.-¿Ha necesitado alguna indicación directa para completar los experimentos?	Si	Si	No	Si	No

TABLA 4-2: RESULTADOS DEL CUESTIONARIO 2

Resumen de los comentarios y observaciones.

Una vez recogidos todos los comentarios se pueden separar en tres grupos distintos:

- Los usuarios han sido críticos con el sistema de detección de voz. La dificultad para pronunciar una palabra del modo que necesita el sistema o que en ocasiones tome el ruido de fondo como una orden dada por el usuario, han sido aspectos que han dificultado el manejo del robot, haciéndolo algo confuso e incomodo.
- La imagen recogida por la cámara del robot que se les mostraba a los usuarios también ha recibido comentarios. A pesar de que todos están de acuerdo en que les ha sido muy útil, señalan que les costaba apreciar las perspectivas, por ejemplo, a la hora de dejar el objeto en la caja durante el experimento 3. Además, consideran que la imagen no es lo suficientemente amplia a la hora de esquivar con seguridad los obstáculos que puedan estar demasiado cerca o ser demasiado pequeños.
- El movimiento de los brazos no les ha resultado suficientemente preciso a los usuarios, sobretodo a la hora de realizar el experimento 4, donde se necesita una gran minuciosidad para conseguir agarrar el objeto sin tirarlo ni chocar con la caja donde este se encuentra apoyado.

5. GESTIÓN DEL PROYECTO

En este apartado se muestra la planificación que se ha llevado a cabo para la realización del proyecto, así como los costes derivados de su proceso de desarrollo.

5.1. PLANIFICACIÓN

Para llevar a cabo la planificación se dividió el proceso de desarrollo en diferentes fases:

- **Análisis:** el objetivo de esta fase es detallar que debe hacer el sistema y de que manera. Como parte del desarrollo de esta fase se creará una lista de casos de uso y un catálogo de requisitos.
- **Diseño:** durante esta fase se determina la arquitectura del sistema, identificando los distintos módulos funcionales y las interconexiones entre ellos.
- **Implementación:** en esta fase se realiza el desarrollo del sistema diseñado en la fase anterior.
- **Pruebas:** tras obtener una primera versión funcional del sistema, se llevan a cabo una serie de pruebas en el entorno real para comprobar su correcto funcionamiento y que cumple con lo requisitos establecidos.
- **Evaluación:** para comprobar la usabilidad y facilidad de uso del sistema, se realiza una evaluación con usuarios externos.
- **Documentación**: esta fase consiste en la creación de este documento.

Cada una de estas fases se separó a su vez en tareas más pequeñas. En la siguiente tabla se muestran las fases y tareas que se han realizado para la consecución del proyecto. Para cada una de estas tareas se puede observar también su fecha de inicio y fin, tanto las fechas reales como las estimadas al comienzo del proyecto. Como se puede ver, existen tareas que no cuentan con fechas estimadas, esto se debe a que surgieron durante la realización del proyecto y no fueron previstas cuando se realizó la planificación. Cabe mencionar, que cada día de trabajo se corresponde con una jornada laboral de 4 horas. En la tabla también se mostrará el rol o perfil encargado de la realización de cada una de las tareas, haciendo uso de la siguiente nomenclatura: el

analista se representa con una A; el diseñador, con una D; el programador, con una P y el operador, con una O. Además, en la realización del proyecto también ha participado un jefe de proyecto, pero se considera un perfil trasversal que ha colabora en todas las fases, por lo que no se refleja en la siguiente tabla. Para conocer el número de horas que el jefe de proyecto ha dedicado, se recomienda consultar Tabla 5-2: Desglose presupuestario del personal.

	REAL			ESTIMADO			PERFIL
Nombre	**Fecha de inicio**	**Fecha de fin**	**Duración**	**Fecha de inicio**	**Fecha de fin**	**Duración**	
Inicio del proyecto	03/10/2011	08/10/2011	5	03/10/2011	08/10/2011	5	A
Análisis	10/10/2011	06/12/2011	39	10/10/2011	29/11/2011	34	A
Estudio de ROS	10/10/2011	19/10/2011	7	10/10/2011	19/10/2011	7	
Estudio de Kinect	19/10/2011	22/10/2011	3	19/10/2011	25/10/2011	4	
Estudio de NAO	24/10/2011	26/10/2011	2	25/10/2011	29/10/2011	4	
Investigación trabajos anteriores	26/10/2011	25/11/2011	20	02/11/2011	16/11/2011	10	
Descripción de requisitos de	25/11/2011	30/11/2011	3	16/11/2011	22/11/2011	4	
Descripción de requisitos del	30/11/2011	06/12/2011	4	22/11/2011	29/11/2011	5	
Diseño	12/12/2011	09/03/2012	53	29/11/2011	19/01/2012	22	D
Diseño de arquitectura	12/12/2011	16/12/2011	4	29/11/2011	13/12/2011	6	
Diseño de mensajes	16/12/2011	21/12/2011	3	13/12/2011	17/12/2011	4	
Diseño a bajo nivel	21/12/2011	21/01/2012	12	19/12/2011	19/01/2012	12	
Modificación de arquitectura	05/03/2012	09/03/2012	4	Tarea no planificada.			
Implementación	23/01/2012	28/04/2012	63	19/01/2012	15/03/2012	40	P
Nodo teleop_robot 1ª versión	23/01/2012	28/01/2012	5	19/01/2012	31/01/2012	8	
Nodo body_capture 1ª versión	30/01/2012	17/02/2012	14	31/01/2012	28/02/2012	20	
Nodo teleop_control 1ª versión	17/02/2012	23/02/2012	4	28/02/2012	15/03/2012	12	
Captura de posición de manos	09/03/2012	12/04/2012	17	Tarea no planificada.			
Nodo teleop_msgs	12/04/2012	13/04/2012	1	Tarea no planificada.			
Nodo body_capture versión final	13/04/2012	20/04/2012	5	Tarea no planificada.			
Nodo teleop_robot versión final	20/04/2012	24/04/2012	2	Tarea no planificada.			
Nodo teleop_control versión final	24/04/2012	28/04/2012	4	Tarea no planificada.			
Pruebas	23/02/2012	22/06/2012	76	15/03/2012	10/05/2012	30	P
Pruebas 1ª versión	23/02/2012	03/03/2012	7	Tarea no planificada.			
Pruebas versión final	03/05/2012	22/06/2012	36	Tarea no planificada.			
Evaluación	13/09/2012	27/09/2012	10	Tarea no planificada.			O
Documentación	22/06/2012	06/10/2012	53	10/05/2012	10/07/2012	43	A,D,P
Realización de la memoria	22/06/2012	03/10/2012	50	10/05/2012	05/07/2012	40	
Realización de la presentación	03/10/2012	06/10/2012	3	05/07/2012	10/07/2012	3	

TABLA 5-1: TAREAS DEL PROYECTO

En la página siguiente se exponen cuatro diagramas de Gantt, en el primero y el tercero está representado el tiempo real que se le ha dedicado a cada una de las tareas y, en el segundo y el cuarto se muestra la distribución de las tareas que se planificó en los inicios del proyecto. De este modo, se puede observar fácilmente los desfases producidos entre lo que se estimó en un principio y como se ha producido el desarrollo realmente. Dentro de estos desfases, aparte de las tareas que llevan más o menos días que los previamente planificados, destaca el rediseño que se llevó a cabo tras la primera fase de pruebas, culpable de la mayor parte del retraso del proyecto.

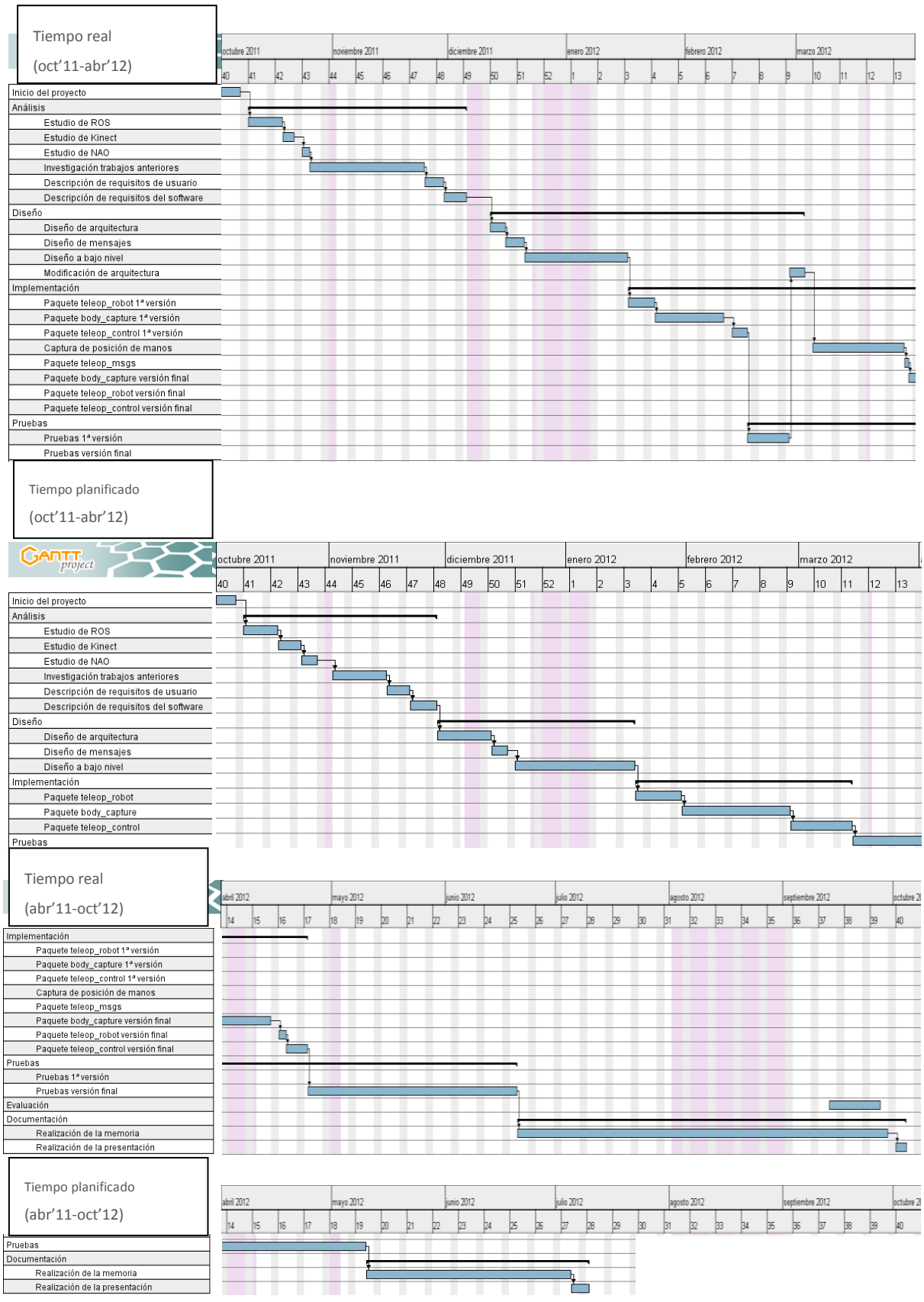


FIGURA 5-1: DIAGRAMAS DE GANTT CON EL TIEMPO REAL Y EL PLANIFICADO

5.2. PRESUPUESTO

En este apartado se llevará a cabo el desglose del presupuesto.

1.- Autor:

Santiago Alfaro Ballesteros

2.- Departamento:

Informática

3.- Descripción del proyecto:

-Título: Sistema de teleoperación mediante una interfaz natural de usuario

- Duración (meses): 12

-Tasa de costes Indirectos: 20%

4.- Presupuesto total del Proyecto (valores en Euros):

34.157,40 €

5.- Desglose presupuestario (costes directos)

PERSONAL

Perfil	Dedicación (hombre/mes)[1]	Coste hombre/mes	Coste final
Analista	1,8	3.806,25 €	6.851,25 €
Diseñador	1,3	3.806,25 €	4.948,13 €
Programador	3,2	3.281,25€	10.500,00 €
Operador	0,5	3.281,25€	1.640,63 €
Jefe de proyecto	0,8	4.593,75 €	3.675,00 €
		TOTAL	**27.615,00 €**

TABLA 5-2: DESGLOSE PRESUPUESTARIO DEL PERSONAL

[1] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas) Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

MATERIAL

Nombre	Coste	% Uso dedicado al proyecto	Dedicación	Periodo de depreciación	Coste imputable [2]
Alienware m15x	1.300 €	20 %	12 meses	60 meses	39 €
Microsoft Kinect	70 €	100 %	12 meses	60 meses	10,5 €
Robot NAO	12.000 €	30 %	12 meses	60 meses	720 €
				TOTAL	**769,5 €**

TABLA 5-3: DESGLOSE PRESUPUESTARIO DEL MATERIAL

OTROS COSTES DIRECTOS

Descripción	Coste
Material de oficina	50 €
Transporte	30 €
TOTAL	**80 €**

TABLA 5-4: DESGLOSE PRESUPUESTARIO DE COSTES DIRECTOS

6.- Resumen de costes

Concepto	Coste
Personal	27.615,00 €
Amortización	769,5 €
Subcontratación de tareas	0 €
Otros costes	80 €
Costes Indirectos	5.692,90 €
TOTAL	**34.157,40 €**

TABLA 5-5: RESUMEN DE COSTES

[2] Fórmula de cálculo de la Amortización:

$\frac{A}{B} \times C \times D$

A=nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

6. CONCLUSIONES

En este apartado se presentan las conclusiones obtenidas tras la finalización del proyecto y la realización de la evaluación. Además, se comentarán las posibles mejoras que se podrían realizar en el futuro.

6.1 CONCLUSIONES GENERALES

La principal conclusión obtenida tras la realización de este proyecto es que las oportunidades que ofrece la teleoperación de robots humanoides son muy grandes, desde la aplicación en todo tipo de trabajos de riesgo, a la posibilidad de viajar a cualquier lugar utilizando robots a modo de avatar. En este contexto, este proyecto es una primera aproximación hacia la telepresencia. Su diseño modular permitirá seguir mejorando el sistema según vaya evolucionando la tecnología. Esta evolución puede venir tanto en lo que se refiere a la robótica, como en las técnicas para que el usuario tenga la sensación de ser el robot.

El desarrollo del proyecto también ha servido para descubrir las posibilidades de sensores como Kinect para llevar a cabo todo tipo de funciones relacionadas con la visión artificial. Sus capacidades no solo ofrecen buenos resultados a la hora de capturar a un usuario, también es capaz de funcionar como sensor de visión en cualquier robot que necesite captar y analizar su entorno. Para apreciar todo lo que se puede conseguir con Kinect es recomendable visitar la página de PCL [23].

Otro de los aspectos a destacar de la realización del proyecto es la utilización de ROS. La existencia de *frameworks* como ROS, que consigan que una gran cantidad de desarrolladores e instituciones desarrollen software compatible y que este sea muy accesible, facilitan el avance de la tecnología. Este avance es posible gracias a que los desarrolladores no tienen que perder el tiempo repitiendo lo que ya ha sido realizado, si no que pueden reutilizarlo y centrarse en crear nuevos programas, que a su vez serán útiles a otros en el futuro.

6.2 REALIZACIÓN DE LOS OBJETIVOS

En este apartado se analizará el grado de realización de los objetivos señalados en el apartado 1.1 Objetivo, y las conclusiones que se pueden obtener tras su ejecución.

- Estudio del *framework* de ROS.

El estudio inicial de ROS fue muy útil y necesario para llevar a cabo la primera versión del diseño. No obstante, el mayor conocimiento se obtuvo tras comenzar con la implementación y empezar a entender mejor las capacidades y el funcionamiento de este *framework*.

- Investigación de los paquetes ya existentes.

El cumplimiento de este objetivo se ha llevado a cabo con éxito, ya que se encontraron numerosos paquetes que cumplían funciones que resultaban indispensables para el desarrollo de este proyecto. Sin estos paquetes la dificultad del proyecto hubiera aumentado mucho y no se hubiera llegado a un resultado tan satisfactorio.

- Análisis de las capacidades del robot NAO.

Se llegó a un conocimiento exhaustivo de las capacidades del NAO, que permitió diseñar un sistema que aprovechará sus características al máximo.

- Diseño del proceso de teleoperación.

En este punto, se pensó en que debía consistir la teleoperación del robot y se llegó al procedimiento que aparece en el anexo E. Manual de usuario. Se considera que este proceso cumple con los objetivos de ser lo suficientemente sencillo como para que cualquier persona puede llevarlo a cabo sin un entrenamiento previo, y sin perder la capacidad de realizar cualquier tipo de operación posible.

- Diseñó de la arquitectura.

El diseño de la arquitectura que se ha alcanzado cumple con el objetivo de ser modular y, por lo tanto, modificable y reutilizable. Se puede sustituir fácilmente cada

uno de los módulos sin tener que modificar los demás gracias a que se han eliminado las dependencias entre ellos.

- Implementación del sistema.

Esta ha sido la parte del proyecto más compleja debido al desconocimiento inicial de este tipo de tecnología y a la dificultad de saber cual era la mejor solución a los distintos problemas a lo que se ha debido hacer frente durante el desarrollo del proyecto. Finalmente, se ha alcanzado un sistema robusto y capaz de llevar a cabo la funcionalidad requerida.

- Evaluación del proyecto.

La evaluación me ha permitido poner a prueba el sistema siendo manejado por usuarios externos a este proyecto. Además, la creación del cuestionario de evaluación, que rellenaron estos usuarios, me ha aportado un *feedback* muy útil a la hora de obtener conclusiones y conocer hacia donde debían estar encaminadas las mejoras futuras.

- Desarrollo de la documentación.

La documentación me ha permitido crear una guía sobre como desarrollar un sistema como este, especificar el entorno en el que se desarrolla el proyecto, definir un manual de usuario, detallar el proceso de instalación y ejecución del sistema, concretar la gestión del proyecto; en definitiva, plasmar por escrito todos los aspectos vinculados con la realización del proyecto.

6.3 PROBLEMAS ENCONTRADOS

En este apartado se describirán los principales los problemas encontrados durante el desarrollo de este proyecto y las distintas soluciones aportadas a cada uno de ellos.

- Falta de precisión de Kinect para la captura de la colocación de las manos.

Kinect tiene muy buenos resultados en lo que se refiere a detectar la posición de las distintas partes del cuerpo de un usuario. No obstante, tiene muchas limitaciones a la hora de reconocer la postura exacta que tienen las manos. En este proyecto, era indispensable disponer de un método para controlar las manos del robot, tanto su posición, como su apertura. Tras encontrar este problema, el primer paso fue investigar que soluciones habían encontrado otros desarrolladores. La mayoría de proyectos solo se centran en analizar la colocación de la mano, por lo que ignoran el resto del cuerpo, y obligan a mantener la mano extendida hacia Kinect, como se puede ver en el apartado 3.3.3.2.4 hand_interaction, o en [25] y [26] . En nuestro sistema esta solución no es válida, ya que el operador debe mover los brazos para controlar al robot y no podría mantenerlos extendidos frente a Kinect. La solución más interesante fue la encontrada en [24], que se basa en la creación de un modelo tridimensional de una mano, para buscar cual es la posición de ese modelo que más se asemeja a la capturada por Kinect. Sin embargo, este sistema no está disponible para su utilización y el desarrollo de uno similar queda fuera del alcance del proyecto. Una vez comprobado que ningún desarrollo existente podía ser reutilizado, se fueron probando distintas soluciones hasta dar con la que finalmente se implementó y que se detalla en el apartado 3.3.3.2.5 body_capture. La principal solución desechada, que tuvo un desarrollo más avanzado, consistía en:

  - Alineación de nubes de puntos.

En una fase inicial de calibración, el operador debía ir colocando sus manos en distintas posiciones, mientras Kinect las detectaba y se almacenaban las nubes de puntos resultantes. Durante la teleoperación, el sistema iba capturando la posición de la mano del operador en cada momento y comprobando a cual de las almacenadas

durante la calibración se parecía más. Para desarrollar esta solución se utilizó el proceso descrito en [27]. Este método resultó ser demasiado lento y, a pesar de ello, fallaba más del 50% de las veces.

- Incompatibilidad del micrófono de Kinect con Ubuntu.

Debido a los problemas de Kinect para detectar las manos, que se describen en el punto anterior, se decidió utilizar la voz para ordenar al robot abrir y cerrar las manos. Además, se podrían utilizar al completo las capacidades de Kinect, mediante la utilización de su micrófono multiarray. No obstante, pronto se descubrió que ninguno de los controladores que estaban disponibles para Ubuntu es capaz de utilizarlo. Solo el controlador desarrollado por Microsoft para Windows puede aprovechar el micrófono multiarray de Kinect. Como solución, se optó por utilizar cualquier otro tipo de micrófono, preferiblemente inalámbrico, a la espera de que alguna nueva versión de los controladores consiga acceder al micrófono de Kinect.

- Imposibilidad de utilizar las dos cámaras de NAO simultáneamente.

Poder acceder a las dos cámaras de Nao al mismo tiempo hubiera sido muy útil para ampliar el campo visual que se le presenta al operador. De esta forma, el operador tendría información tanto de la parte más lejana del entorno, como de lo que el robot tiene inmediatamente delante. Sin embargo, debido a la infraestructura interna del NAO, es imposible. Se ha optado por hacer que el robot mire siempre hacia el frente y solo mueva la cámara para seguir las manos cuando se va a realizar alguna acción con ellas. Otra posible solución hubiera consistido en añadir un nuevo comando de voz para poder intercambiar el uso de la cámara superior e inferior.

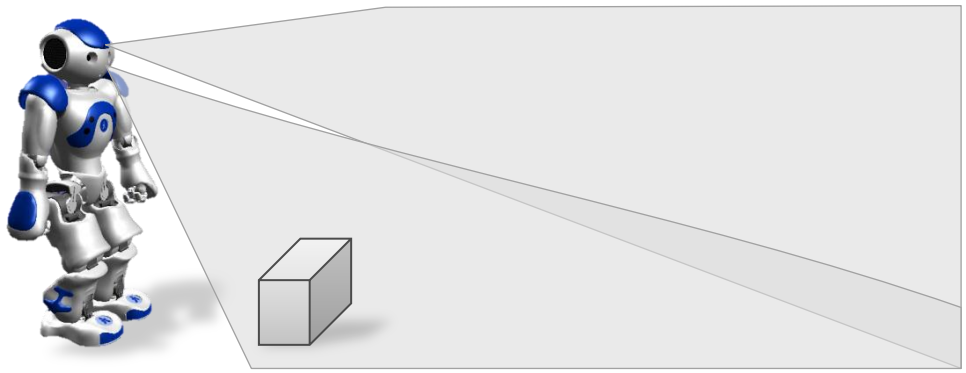


FIGURA 6-1: UTILIZACIÓN SIMULTÁNEA DE LAS CÁMARAS DEL NAO

- Existencia de retardos.

Existen retardos entre el momento en que el operador da la orden hasta que el robot la ejecuta. Estos no afectan demasiado en lo que se refiere a la teleoperación de los brazos del robot, pero sí se aprecian en el momento de hacer caminar al robot o hacerlo girar. Son especialmente peligrosos los retardos a la hora de detener el desplazamiento del robot, ya que si no se realiza la parada con suficiente tiempo, puede provocar choques y caídas. Para descartar que la mayor parte del tiempo de este retardo se estuviera consumiendo en los distintos procesamientos que lleva a cabo el sistema, se calculó la duración desde que se recibía la información de Kinect hasta que se le transmitía al robot. El resultado fue que el procesamiento apenas suponía unas milésimas de segundo, por lo que se concluyó que el retardo se debía a la utilización de una red *wireless* y al funcionamiento de *NAOqi*.

- Incompatibilidades entre la versión de *NAOqi* y la versión de *python*.

Es muy importante que la versión de *NAOqi* sea compatible con la versión de *python* que se esté utilizando, si no, se pueden producir errores que produzcan un mal funcionamiento. A los largo del desarrollo, este problema surgió varias veces al probar el sistema en distintos equipo .Fue muy complicado dar con la solución, debido a la falta de información respecto a este tipo de errores que *Aldebaran Robotics* proporciona. En la versión final del proyecto se utiliza *NAOqi* 1.12.3 y *Python* 2.6.5.

6.4 LÍNEAS FUTURAS

Las principales mejoras que se podrían aplicar al proyecto en un futuro son:

- Utilizar directamente la matriz de micrófonos de Kinect, en lugar de un micrófono externo. Esto no se ha podido realizar debido a que los controladores disponibles para Ubuntu todavía no soportan estas capacidades del sensor.
- Desarrollar una adaptación para Windows cuando exista una versión más estable de ROS para este sistema operativo. Utilizando Windows se podrían emplear los controladores oficiales de Kinect creados por Microsoft, cuya principal ventaja es poder disponer de su reconocimiento de voz.
- Realizar las modificaciones oportunas para que el robot reproduzca por sus altavoces lo que esté diciendo el teleoperador. Para ello sería necesario sustituir los comandos de voz por otro tipo de orden o crear una palabra clave para que a partir de ella, lo siguiente que se pronuncie, sea considerada un comando.
- Realizar una interfaz gráfica que permita iniciar la aplicación y ofrezca la información de un modo accesible y atractivo al usuario.
- Añadir nuevos robot para que puedan ser teleoperados haciendo uso de la interfaz natural de usuario creada con Kinect.
- Utilizar sensores cada vez más sensibles, como el anunciado Kinect 2, para que la captación del usuario se realice de una manera más precisa, sobre todo en lo que respecta a evitar que el usuario se tenga que mantener con el pulgar extendido y el puño cerrado. Si se llegara a una precisión realmente buena, se podrían sustituir las órdenes de voz para abrir o cerrar la mano, por la imitación directa de la posición de las manos del teleoperador.
- Aumentar el número de comandos de voz, para permitir invocar más acciones del robot. Se podrían crear comandos para ordenar al robot sentarse o realizar cualquier tipo de acción preprogramada. Todavía más interesante sería otorgar al operador la posibilidad de utilizar comandos del tipo “Anda 70 centímetros” o “Gira 20 grados”, lo que daría una precisión adicional al sistema actual.
- Aplicar todas las mejoras posibles para que aumente la sensación de telepresencia del usuario. Una de estas posibles mejoras sería reproducir por unos altavoces lo

que el robot esta capturando por su micrófono. También se podría utilizar un robot con visión estereoscópica y un HMD para que el usuario vea lo que está captando el robot y sea capaz de apreciar la profundidad. Un HMD es un dispositivo similar a un casco con una pantalla muy cercana a los ojos, que provoca que el usuario tenga una sensación de inmersión en la imagen que se le muestra. También se podría aplicar lo que se conoce como realimentación cinestésica, que es la que informa al teleoperador de las fuerzas que está sufriendo el robot. Por ejemplo, si existiera un obstáculo que impidiera al robot mover el brazo derecho, el usuario podría tener un traje especial que le dificultara realizar ese movimiento.

A. ROS

Las siglas de ROS se refieren a *Robot Operating System*, que se puede traducir como un sistema operativo de robots [28] [29]. No obstante, no funciona como lo que normalmente se entiende por un sistema operativo, es decir, no se encarga de la administración y manejo de procesos y recursos. Nació durante el desarrolló del STAIR Project en la Universidad de Stanford [30] [31] y del *Personal Robots Program* del *Willow Garage* [32] para dar solución a una serie retos que se les fueron planteando. De está manera, crearon ROS para que cumpliera los siguientes objetivos:

- ***Peer-to-peer***

ROS tiene una topología de peer-to-peer lo que le permite no tener que disponer de un servidor central por el que tengan que pasar todas las comunicaciones. Normalmente, los sistemas realizados en ROS cuentan con una gran cantidad de procesos que se comunican entre sí. Estos pueden estar ejecutándose en distintas máquinas conectadas mediante distintos tipos de red, cableada o inalámbrica. Por ejemplo, en el caso de la imagen se tienen dos grupos de máquinas conectadas internamente mediante *Ethernet*, y usando *WiFi* para conectar un grupo con otro, la utilización de un servidor central podría provocar un cuello de botella crítico, debido a la menor velocidad de la red *WiFi*.

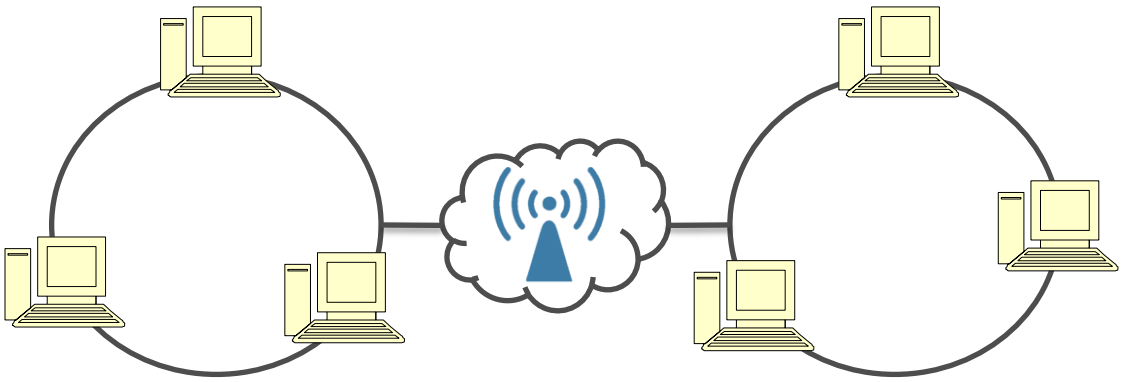


FIGURA A-1: EJEMPLO DE POSIBLE INFRAESTRUCTURA DE UN SISTEMA EN ROS

A pesar de que la tipología peer-to-peer no necesita que todas las comunicaciones pasen por un único sitio, sí es necesario un nodo maestro que haga posible que los procesos se encuentren en tiempo de ejecución.

- **Libre y de código abierto**

ROS se distribuye bajo una licencia BSD, que permite tanto proyectos no comerciales, como comerciarles. Todo el código de ROS es abierto y accesible, lo que facilita la depuración de todos los niveles del software desarrollado.

- **Multi-lenguaje**

Para facilitar el trabajo de varios equipos en un mismo proyecto se consideró esencial que ROS soportará distintos lenguajes de programación. Desde sus inicios, ROS ha sido compatible con C++, Python y Lisp, contando ya con versiones avanzadas para Java y Lua.

- **Sin gran núcleo central**

ROS funciona a partir de un gran número de pequeños módulos, cada uno encargado de llevar a cabo una pequeña parte de la funcionalidad total del sistema. A pesar de la perdida de eficiencia que supone este tipo de construcción frente a la existencia de un núcleo central, la mayor estabilidad y facilidad de administración conseguida se consideraron más importantes.

- **Ligero**

ROS está construido con la idea de que todo el código de controladores y algoritmos complejos se mantenga en librerías independientes, con el objetivo de que puedan ser reutilizadas fuera del proyecto para el que fueron construidas. Para que el resto de ROS pueda hacer uso de estas librerías, solo se precisa de unos sencillos y ligeros ejecutables que permitan el acceso a su funcionalidad. Siguiendo esta filosofía, ROS ya hace uso de gran cantidad de proyectos de código libre, como OpenCV que provee algoritmos para llevar a cabo visión artificial o PCL, que se utiliza en este proyecto.

A.1. Elementos de ROS

EL elemento principal de ROS son los **nodos**. Un nodo es un proceso encargado de realizar determinado procesamiento. Los sistemas basados en ROS suelen estar formados por varios de estos nodos trabajando juntos. Los nodos están dentro de **paquetes**, con el objetivo de que los nodos que tengan una funcionalidad similar permanezcan agrupados en el mismo paquete.

Los nodos se comunican unos con otros mediante el paso de **mensajes**. Los mensajes son estructuras de datos que pueden estar compuestos por tipos de datos básicos y por otros mensajes. Al igual que los nodos, los mensajes están definidos dentro de paquetes. Para que un nodo comience a mandar mensajes, tendrá que anunciar el tipo de mensaje que va a emitir y asignarle un nombre o ***topic***, como se le denomina en ROS al nombre de un mensaje. Otro nodo se subscribirá al *topic* y comenzará a recibir los mensajes.

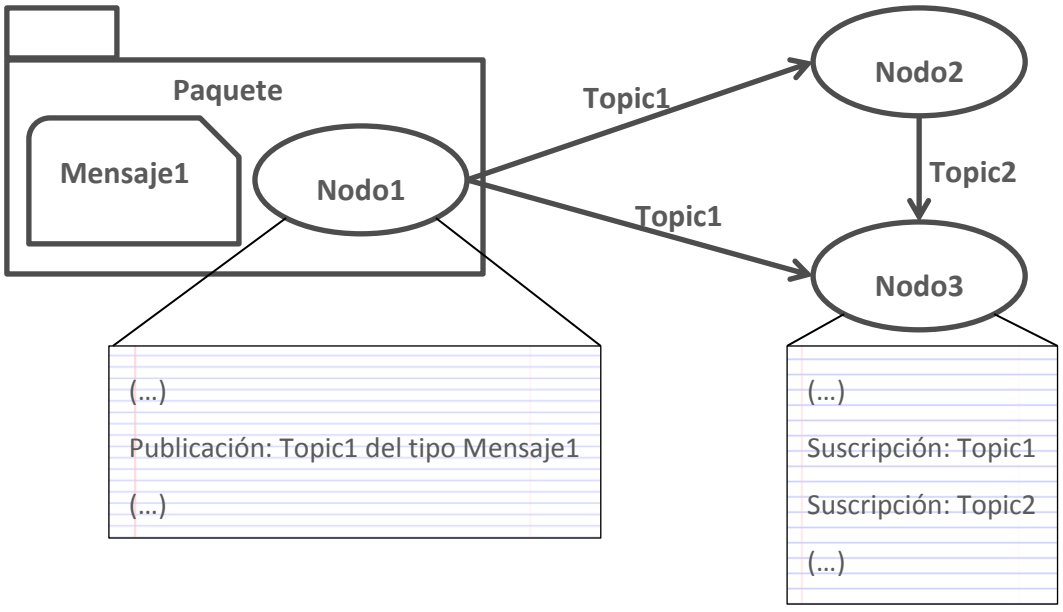


FIGURA A-2: ELEMENTOS DE ROS

Para ayudar a diferenciar mejor los que es un mensaje de lo que es un *topic*, se puede realizar un símil con la programación orientada a objetos, donde el mensaje sería una clase y el *topic*, un objeto. ROS posibilita que varios nodos emitan o reciban

el mismo *topic*, y que un nodo emita o reciba varios *topic*. Un sistema se puede representar como un grafo dirigido en el que los nodos se corresponden con los nodos de ROS y los arcos con los *topic* que salen del emisor y llegan al receptor.

Para las comunicaciones síncronas, ROS implementa **servicios**. Estos servicios se refieren a funciones que realizan determinados nodos y que pueden ser invocadas por otros. Se representan con un nombre, un tipo de mensaje de entrada y otro de salida, y se definen en un paquete. A diferencia de los *topics*, un servicio solo puede ser realizado por un nodo al mismo tiempo.

A.2. Componentes básicos de un paquete

Como se ha dicho anteriormente, un paquete es el lugar físico donde se agrupan los nodos, aunque es muy común encontrar paquetes con un único nodo. Un paquete se corresponde con un directorio que suele tener la siguiente estructura:

- **Subdirectorio "src":** En esta carpeta se almacena el código fuente necesario para la creación del nodo ejecutable, por ejemplo, los ficheros .cpp si se está trabajando con el lenguaje C++.
- **Subdirectorio "bin":** Una vez que se compila el paquete en este directorio se encontrarán los ficheros ejecutables
- **Subdirectorio "nodes":** Cuando se está trabajando en el lenguaje Python, los archivos .py se almacenan en esta carpeta ya que no necesitan ser compilados.
- **Subdirectorio "include":** Los ficheros de cabecera .h que se necesiten permanecen en esta carpeta.
- **Subdirectorio "msgs":** En esta carpeta se definen los mensajes que se crean en el paquete, si no se encuentra significa que el paquete no se encarga de la construcción de mensajes.
- **Subdirectorio "srv":** En está carpeta se definen los servicios que ofrece el paquete, si no se encuentra significa que el paquete no ofrece ninguno.

- **Subdirectorio “launch”:** En este directorio se encuentran los ficheros .launch, documentos utilizados para invocar la ejecución de varios nodos que normalmente tienen interdependencias.
- **Archivo “manifest.xml”:** En este archivo se describen las principales características del paquete, como el autor, su licencia, una breve descripción o su página web si existe. También se definen las dependencias con otros paquetes, siendo imposible compilar este paquete sin disponer de los que depende.
- **Archivo “CMakeLists.txt”:** En este archivo se especifican las opciones de compilación del paquete, como el nombre del nodo ejecutable o referencias a las librerías externas que son necesarias.

B. MODELO DE COMUNICACIÓN

En este apartado se describirá el contenido de cada uno de los mensajes implicados en el sistema. Se detallará que nodo emite el mensaje, que nodos lo reciben y cual es su contenido.

B.1. /camera/rgb/points

FIGURA B-1: MENSAJE /CAMERA/RGB/POINTS

- **Descripción**: Transporta la nube de puntos detectada por Kinect.
- **Emisor**: openni_camera
- **Receptores**: hand_interaction
- **Tipo**: sensor_msgs/PointCloud2
- **Contenido:**

```
Header header          //Encabezamiento con información temporal
uint32 height  //Altura de la nube de puntos, será 1 si no está ordenada
uint32 width   //Anchura de la nube de puntos
PointField[] fields       //Descripción de cada entrada de punto
bool    is_bigendian    //Es True si está en big-endian
uint32  point_step      //Tamaño de un punto en bytes
uint32  row_step        //Tamaño de una fila en bytes
uint8[] data     //Lista con los puntos que forman la nube
bool is_dense           //Es True si no tiene puntos inválidos
```

B.2. /skeletons

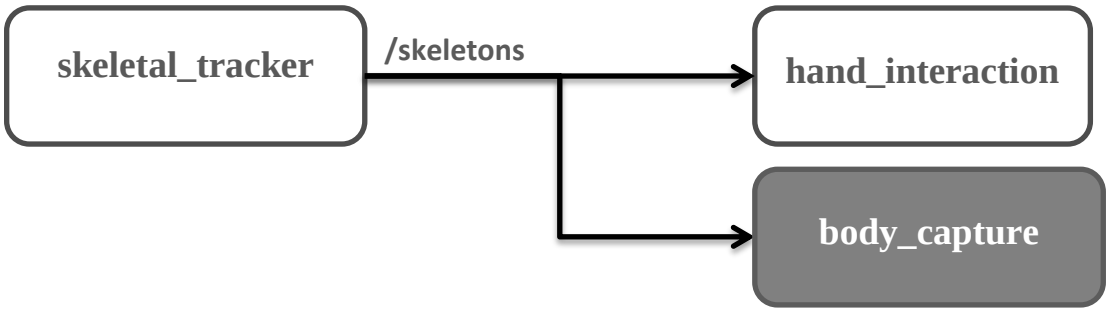


FIGURA B-2: MENSAJE /SKELETONS

- **Descripción:** Transporta la lista de usuarios detectados por Kinect. En este proyecto solo se atenderá al primer usuario detectado.
- **Emisor**: skeletal_tracker
- **Receptores**: hand_interaction, body_capture
- **Tipo**: body_msgs/Skeletons
- **Contenido**:

  Header header //Encabezamiento con información temporal

  Body_msgs/Skeleton[] skeletons //lista de mensajes de tipo skeletons

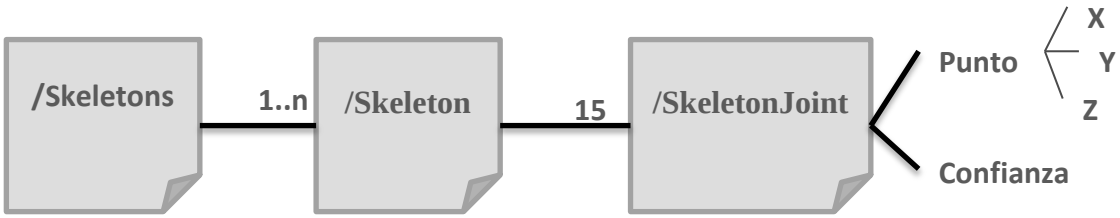


FIGURA B-3: CONTENIDO DEL MENSAJE /SKELETONS

El mensaje /*Skeletons* consiste en una lista de mensajes de tipo /*Skeleton,* uno por cada uno de los usuarios que estén siendo detectados por la Kinect. En este sistema sólo se realiza la detección de un único usuario, ignorando el resto. A su vez, /*Skeleton* esta formando por un conjunto de mensajes de tipo /*SkeletonJoint*, uno para cada parte del cuerpo necesaria para la teleoperación. A continuación, se detalla más a fondo el contenido de estos dos mensajes.

B.3. /Skeleton

- **Descripción:** Forma parte del mensaje /skeletons y transporta la información de las distintas partes del cuerpo de un usuario.
- **Tipo**: body_msgs/Skeleton
- **Contenido:**

  int32 playerid //Identificador del usuario al que pertenece este esqueleto, en este proyecto solo se usa el primer usuario

  body_msgs/SkeletonJoint head //Datos de la cabeza

  body_msgs/SkeletonJoint neck //Datos del cuello

  body_msgs/SkeletonJoint right_hand //Datos de la mano derecha

  body_msgs/SkeletonJoint left_hand //Datos de la mano izquierda

  body_msgs/SkeletonJoint right_shoulder //Datos del hombro derecho

  body_msgs/SkeletonJoint left_shoulder //Datos del hombro izquierdo

  body_msgs/SkeletonJoint right_elbow //Datos del codo derecho

  body_msgs/SkeletonJoint left_elbow //Datos del codo izquierdo

  body_msgs/SkeletonJoint torso //Datos del torso

  body_msgs/SkeletonJoint left_hip //Datos de la cadera izquierda

  body_msgs/SkeletonJoint right_hip //Datos de la cadera derecha

  body_msgs/SkeletonJoint left_knee //Datos de la rodilla izquierda

  body_msgs/SkeletonJoint right_knee //Datos de la rodilla derecha

  body_msgs/SkeletonJoint left_foot //Datos del pie izquierdo

  body_msgs/SkeletonJoint right_foot //Datos del pie derecho

B.4. /SkeletonJoint

- **Descripción:** Forma parte del mensaje /skeleton y transporta información sobre la posición de una parte del cuerpo.
- **Tipo**: body_msgs/Skeleton
- **Contenido:**

  geometry_msgs/Point position //Punto tridimensional que representa la posición de la parte del cuerpo

  float32 confidence //Grado de confianza con que se está capturando la parte del cuerpo. Tiene un valor comprendido entre 0, cuando no se detecta esa parte del cuerpo, y 1, si se captura correctamente.

B.5. /hand1_fullcloud y /hand0_fullcloud

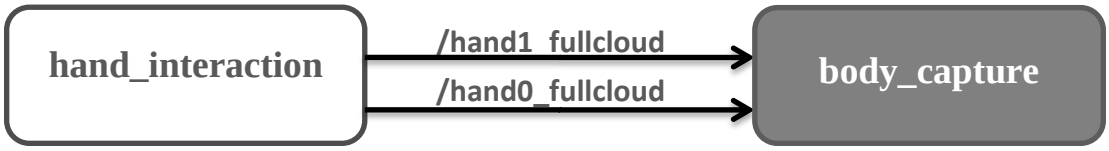


FIGURA B-4: MENSAJES /HAND1_FULLCLOUD Y /HAND0_FULLCLOUD

- **Descripción:** Transportan las nubes de puntos de la mano izquierda (/hand0_fullcloud) y de la derecha (/hand1_fullcloud)
- **Emisor**: handdetector
- **Receptores**: body_capture
- **Tipo**: sensor_msgs/PointCloud2
- **Contenido**: similar al mensaje /CAMERA/RGB/POINTS descrito anteriormente.

B.6. recognizer/output

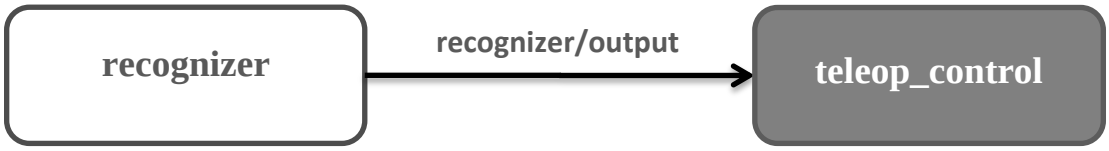


FIGURA B-5: MENSAJE RECOGNIZER/OUTPUT

- **Descripción:** Transporta la orden de voz detectada o está vacío si se ha escuchado algo pero no se ha identificado como un comando.
- **Emisor:** recognizer
- **Receptores:** teleop_control
- **Tipo:** std_msgs/String
- **Contenido:**

```
string data      //Cadena de texto con la orden detectada o vacía.
```

B.7. /bodyAngle

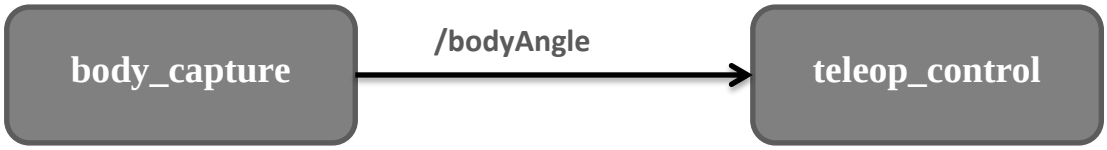


FIGURA B-6: MENSAJE /BODYANGLE

- **Descripción:** Transporta los ángulos de las articulaciones del operador y la posición de algunas partes del cuerpo.
- **Emisor:** body_capture
- **Receptores:** teleop_control
- **Tipo:** teleop_msgs/body_angle
- **Contenido:**

```
float64 RShoulderOpeningAngle      //Apertura del hombro derecho
float64 RShoulderRotationAngle     //Rotación del hombro derecho
float64 RElbowOpeningAngle         //Apertura del codo derecho
float64 RElbowRotationAngle        //Rotación del codo derecho
float64 RHandRotationAngle         //Rotación de la mano derecha
```

geometry_msgs/Point RShoulderPosition //Posición del hombro derecho

geometry_msgs/Point RHandPosition //Posición de la mano derecha

geometry_msgs/Point Head //Posición de la cabeza

float64 LShoulderOpeningAngle //Apertura del hombro izquierdo

float64 LShoulderRotationAngle //Rotación del hombro izquierdo

float64 LElbowOpeningAngle//Apertura del codo izquierdo

float64 LElbowRotationAngle //Rotación del codo izquierdo

float64 LHandRotationAngle //Rotación de la mano izquierda

geometry_msgs/Point LShoulderPosition //Posición del hombro izquierdo

geometry_msgs/Point LHandPosition //Posición de la mano izquierda

B.8. /walkInfo

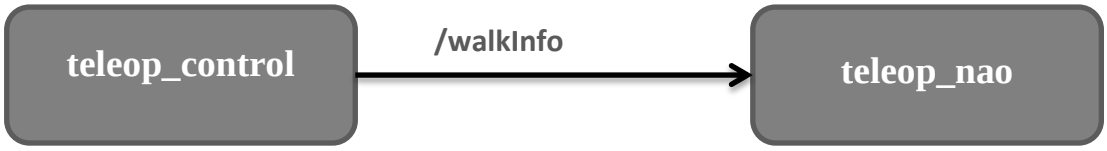


FIGURA B-7: MENSAJE /WALKINFO

- **Descripción:** Transporta la información sobre como debe andar el robot.
- **Emisor:** teleop_control
- **Receptores:** teleop_nao
- **Tipo:** teleop_msgs/walk_info
- **Contenido:**

  float64 angle //Ángulo de giro

  float64 x //Movimiento del robot NAO en su eje X

  float64 y //Movimiento del robot NAO en su eje Y

B.9. LHandOpening y /RHandOpening

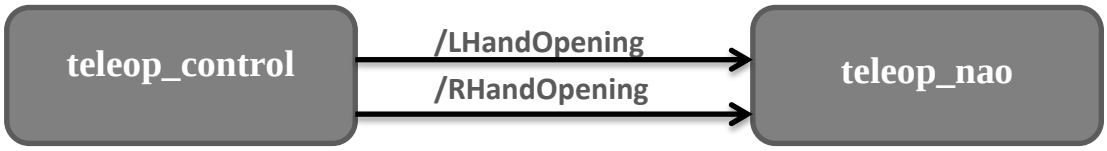


FIGURA B-8: MENSAJE /LHANDOPENING Y /RHANDOPENIND

- **Descripción:** Transporta la orden de cambiar la posición de las manos.
- **Emisor:** teleop_control
- **Receptores:** teleop_nao
- **Tipo:** std_msgs/Bool
- **Contenido**

  bool data //Será True para abrir la mano y False para cerrarla.

B.10. /NaoSays

FIGURA B-9: MENSAJE /NAOSAYS

- **Descripción:** Transporta una cadena de texto para que sea reproducida por el robot.
- **Emisor:** teleop_control
- **Receptores:** teleop_nao
- **Tipo:** std_msgs/String
- **Contenido**

  string data //Cadena de texto que el robot debe reproducir.

B.11. Conjunto de mensajes del tipo teleop_msgs/joint_angle

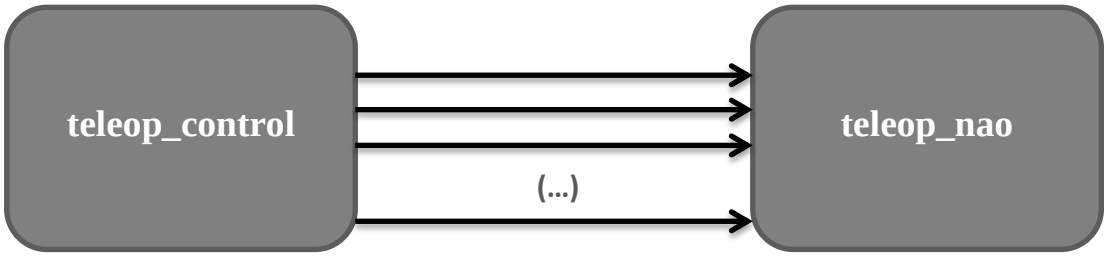


FIGURA B-10: MENSAJES DEL TIPO TELEOP_MSGS/JOINT_ANLGE

- **Descripción:** Estos mensajes son los encargados de transportar la información sobre los ángulos que deben adoptar las articulaciones del robot.
- **Emisor:** teleop_control
- **Receptores:** teleop_nao
- **Tipo:** teleop_msgs/joint_angle
- **Contenido:**

  float64 angle //Ángulo que debe tomar la articulación del robot

- **Conjunto de mensajes:**

  /HeadPitch //Inclinación de la cabeza
  /HeadRotation //Rotación de la cabeza
  /LElbowOpening //Apertura del codo izquierdo
  /LElbowRotation //Rotación del codo izquierdo
  /LHandRotation //Rotación de la mano izquierda
  /LShoulderOpening //Apertura del hombro izquierdo
  /LShoulderRotation //Rotación del hombro izquierdo
  /RElbowOpening //Apertura del codo derecho
  /RElbowRotation //Rotación del codo derecho
  /RHandRotation //Rotación de la mano derecha
  /RShoulderOpening //Apertura del hombro derecho
  /RShoulderRotation //Rotación del hombro derecho

C. INSTALACIÓN

En este manual de instalación se describirán los pasos a seguir para posibilitar la ejecución del sistema de teleoperación.

1. Instalar Ubuntu 10.04 o superior, hasta 11.04.
2. Instalar ROS Diamondback
    a. Configurar “source.list”
        i. Para Lucid 10.04

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

        ii. Para Natty 11.04

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

    b. Configurar claves

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

    c. Instalación

```
sudo apt-get update
sudo apt-get install ros-diamondback-desktop-full
```

    d. Configuración del entorno

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

3. Descargar el paquete nao_kinect desde el cd a la carpeta home
    a. Configuración del entorno

```
echo ". /opt/ros/diamondback/stacks/nao_kinect/setup.sh" >> ~/.bashrc
. ~/.bashrc
```

4. Instalar paquetes externos
    a. Dar permisos de escritura

```
cd /opt/ros/diamondback
sudo chmod -R 777 ./stacks
cd /opt/ros/diamondback/stacks
```

b. Descargar el paquete nao_kinect desarrollado en este proyecto en la carpeta /opt/ros/diamondback/stacks

c. Configurar entorno

```
echo ". /opt/ros/diamondback/stacks/nao_kinect/setup.sh" >> ~/.bashrc
. ~/.bashrc
```

d. Descarga de los paquetes del MIT

```
svn co https://svn.csail.mit.edu/mit-ros-pkg/trunk/kinect_utils/body_msgs
svn co https://svn.csail.mit.edu/mit-ros-pkg/trunk/pcl_tools
svn co https://svn.csail.mit.edu/mit-ros-pkg/trunk/nnn
svn co https://svn.csail.mit.edu/mit-ros-pkg/trunk/kinect_utils/hand_interaction
svn co https://svn.csail.mit.edu/mit-ros-pkg/trunk/parallel_tools
svn co https://svn.csail.mit.edu/mit-ros-pkg/trunk/kinect_utils/skeletal_tracker
```

e. Descarga del paquete nao_vision

```
svn co http://wpi-rail.googlecode.com/svn/trunk/distribution/wpi-rail-ros-pkg/nao_rail/nao_vision
```

f. Descarga e instalación de pocketsphinx

```
svn checkout http://albany-ros-pkg.googlecode.com/svn/trunk/rharmony rharmony
sudo apt-get install gstreamer0.10-pocketsphinx
```

g. Descarga e instalación de openni

```
sudo apt-get install ros-diamondback-openni-kinect
```

5. Compilación de paquetes

a. Compilación de paquetes externos

```
rosmake body_msgs
rosmake pcl_tools
rosmake hand_interaction
rosmake nao_vision
```

b. Compilación de paquetes propios

```
rosmake body_capture
rosmake nao_control
rosmake teleop_nao
rosmake skeletal_tracker_new
```

D. EJECUCIÓN

Para llevar a cabo la ejecución del sistema lo primero es enchufar Kinect al ordenador y asegurarse de que este tiene acceso al robot NAO a controlar. Finalmente, se lanza el fichero teleop_nao.launch escribiendo el siguiente comando en consola:

```
roslaunch teleop_robot teleop_nao.launch
```

En la página siguiente se muestra el contenido del fichero teleop_nao.launch. Mediante este fichero se van iniciando todos los nodos que forman el sistema en el siguiente orden:

1. openni_camera
2. recognizer
3. nao_vision
4. handdetector
5. skeletal_tracker_new
6. body_capture
7. teleop_control
8. teleop_nao

Es muy posible que sea necesario realizar ciertas modificaciones en el fichero teleop_nao.launch, concretamente en la dirección IP especificada en el momento de ejecutar ***nao_vision*** (línea 31) y ***teleop_nao*** (línea 60), que debe coincidir con la dirección del robot que se quiere controlar.

```
<launch>
    <!-- Import your PYTHONPATH -->
        <env name="PYTHONPATH" value="$(env PYTHONPATH)"/>

    <!--    launch kinect sensor -->
        <arg name="debug" default="false"/>
        <arg if="$(arg debug)" name="launch_prefix" value="xterm -rv -e gdb -ex run -args"/>
        <arg unless="$(arg debug)" name="launch_prefix" value=""/>
        <node pkg="openni_camera" type="openni_node" name="openni_node1" output="screen" launch-prefix="$(arg launch_prefix)">
            <param name="device_id" value="#1"/> <!-- this line uses first enumerated device -->
            <rosparam command="load" file="$(find openni_camera)/info/openni_params.yaml" />
            <param name="rgb_frame_id" value="/openni_rgb_optical_frame" />
            <param name="depth_frame_id" value="/openni_depth_optical_frame" />
            <param name="use_indices" value="false" />
            <param name="depth_registration" value="true" />
            <param name="image_mode" value="2" />
            <param name="depth_mode" value="2" />
            <param name="debayering" value="2" />
            <param name="depth_time_offset" value="0" />
            <param name="image_time_offset" value="0" />
        </node>

    <!-- launch speech recognition -->
        <node name="recognizer" pkg="pocketsphinx" type="recognizer.py"  >
                <param name="lm" value="$(find teleop_robot)/dic/word_cmd.lm"/>
                <param name="dict" value="$(find teleop_robot)/dic/word_cmd.dic"/>
        </node>

    <!-- launch nao_vision -->
        <!-- Edit the parameters here for your Nao -->
            <param name="/naoqi/host" type="string" value="192.168.1.50"/>
            <param name="/naoqi/port" value="9559"/>
        <!-- Optional Parameter if your PYTHONPATH is not setup correctly
            <param name="/naoqi/path" type="string" value="$(find teleop_robot)/lib"/> -->

        <!-- Optional Parameters for Camera Settings -->
            <param name="/nao_vision/resolution" value="0"/>
            <param name="/nao_vision/camera" value="1"/>

        <!-- Create an instance of each appropriate node -->
            <node name="nao_vision" pkg="nao_vision" type="nao_vision.py" respawn="false"/>


    <!-- run hand detection -->
        <node pkg="hand_interaction" type="detectskelhands" name="handdetector">

        <!-- This reduces the point cloud resolution to 320x240, for fast hand segmentation... -->
        <node pkg="dynamic_reconfigure" type="dynparam" name="ressetter" args="set /openni_camera point_cloud_resolution 1" />

    <!-- This node has to be launched after the openni_kinect nodes, so a sleep(10) was added to the code... -->
        <param name="config_file_path" value="$(find teleop_robot)/config/SamplesConfig.xml" />
        <node pkg="skeletal_tracker_new" type="tracker" name="skel_tracker" respawn="true" >

    <!-- launch image_view -->
        <node pkg="image_view" type="image_view" name="image_view" args="image:=/nao_camera" />

    <node pkg="body_capture" type="body_capture"  name="body_capture" >

    <node pkg="teleop_control" type="teleop_control"  name="teleop_control" output="screen" >

    <node pkg="teleop_robot" type="teleop_nao.py"   args="--pip 192.168.1.50 --pport 9559" name = "teleop_nao">

</launch>
```

FIGURA D-1: FICHERO TELEOP_NAO.LAUNCH

También, se pueden destacar las líneas 25 y 26 donde se especifican los dos archivos que le indican al nodo ***recognizer*** las palabras que debe ser capaz de detectar. Estos archivos se han generado automáticamente haciendo uso de la página web http://www.speech.cs.cmu.edu/tools/lmtool.html, que los crea a partir de un archivo cuyo único contenido son las palabras a reconocer, escritas cada una en una línea. El contenido de estos archivos se muestra a continuación.

```
1  LEFT    L EH F T
2  RIGHT   R AY T
3  START   S T AA R T
4  FINISH  F IH N IH SH
```

FIGURA D-2: FICHERO WORD_CMD.DIC

```
1  Language model created by QuickLM on Thu Mar 29 06:52:20 EDT 2012
2  Copyright (c) 1996-2000
3  Carnegie Mellon University and Alexander I. Rudnicky
4
5  This model based on a corpus of 4 sentences and 6 words
6  The (fixed) discount mass is 0.5
7
8  \data\
9  ngram 1=6
10 ngram 2=8
11 ngram 3=4
12
13 \1-grams:
14 -0.7782 </s> -0.3010
15 -0.7782 <s> -0.2218
16 -1.3802 LEFT -0.2218
17 -1.3802 RIGHT -0.2218
18 -1.3802 START -0.2218
19 -1.3802 FINISH -0.2218
20
21 \2-grams:
22 -0.9031 <s> LEFT 0.0000
23 -0.9031 <s> RIGHT 0.0000
24 -0.9031 <s> START 0.0000
25 -0.9031 <s> FINISH 0.0000
26 -0.3010 LEFT </s> -0.3010
27 -0.3010 RIGHT </s> -0.3010
28 -0.3010 START </s> -0.3010
29 -0.3010 FINISH </s> -0.3010
30
31 \3-grams:
32 -0.3010 <s> LEFT </s>
33 -0.3010 <s> RIGHT </s>
34 -0.3010 <s> START </s>
35 -0.3010 <s> FINISH </s>
36
37 \end\
```

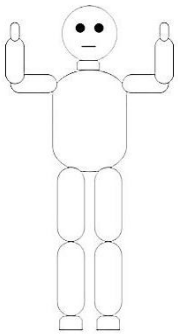
FIGURA D-3: FICHERO WORD_CMD.LM

E. MANUAL DE USUARIO

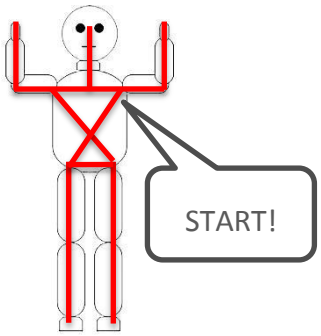
MANUAL DE USUARIO

INICIO

1. Adoptar posición inicial y esperar la calibración.
2. Decir "**START**" para comenzar a controlar el robot.



1 ADOPTANDO POSICIÓN INICIAL



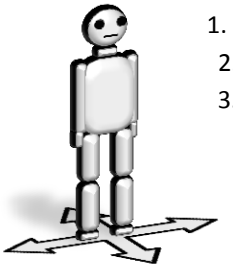
2 CALIBRACIÓN REALIZADA

3. ATENCION!!! Mantener siempre ambas manos con el pulgar extendido.

ABRIR/CERRAR MANOS

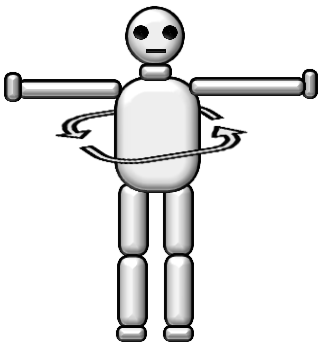
1. Decir "**LEFT**" para que la mano izquierda del robot pase de abierta a cerrada o viceversa.
2. Decir "**RIGHT**" para que la mano derecha del robot pase de abierta a cerrada o viceversa.

ANDAR Y GIRAR

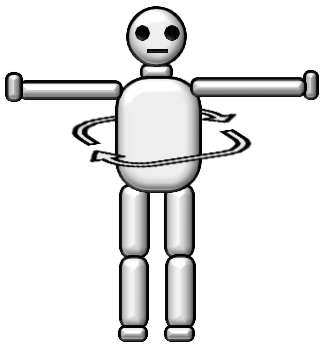


1. Para que el robot ande hacia delante, dar un paso al frente.
2. Para que el robot de marcha atrás, dar un paso atrás.
3. Para que el robot se mueva lateralmente, dar un paso a un lado.

4. Para que el robot gire, girar el cuerpo sin mover los pies.



EL ROBOT GIRARÁ HACIA LA DERECHA



EL ROBOT GIRARÁ HACIA LA IZQUIERDA

TERMINAR

1. Decir "**FINISH**" para dejar de controlar al robot.

F. GLOSARIO DE TÉRMINOS

- **Acelerómetro:** es un instrumento encargado de medir aceleraciones.
- **Aldebaran Robotics:** una empresa francesa con sede en París que nació en el año 2005, creadora del robot NAO.
- **API:** son las siglas de *Application Programming Interface* o Interfaz de Programación de Aplicaciones, en español. Se refiere al conjunto de funciones que ofrece una biblioteca para ser utilizada por otro software.
- **Autómata:** Instrumento o aparato que encierra dentro de sí el mecanismo que le imprime determinados movimientos.
- **Biblioteca:** es un módulo de software que contiene código y datos, y que proporciona servicios a otros programas.
- **Bumper:** es un sensor de presión utilizado comúnmente en robótica para detectar choques.
- **RGB:** son las siglas de Red Green Blue, y se refiere a los tres colores en los que se puede descomponer una imagen.
- **CMOS:** un sensor CMOS es aquel capaz de detectar la luz.
- **Dirección IP:** es una etiqueta numérica que identifica a un dispositivo dentro de una red que utilice el protocolo IP.
- **E3:** se refiere a la Electronic Entertainment Expo, la convención de videojuegos más importante, que se celebra anualmente en Los Ángeles, California.
- **Ethernet:** es un estándar de transmisión de datos para redes de área local.
- **Fps:** son las siglas de Fotogramas Por Segundo.
- **Framework:** es una estructura conceptual y tecnológica sobre el que se soportan otros proyectos software. Son diseñados con la intención de facilitar el desarrollo de software.
- **Diagrama de Gantt:** es un diagrama que muestra el tiempo adjudicado a distintas tareas.
- **General Motors:** es una de las empresas fabricante de automóviles más importante del mundo, con filiales como Chevrolet, Daewoo u Opel.
- **Girómetro:** instrumento encargado de detectar y medir giros.

- **Honda:** es una empresa de origen japonés que fabrica automóviles, propulsores para vehículos terrestres, acuáticos y aéreos, motocicletas y en general componentes para la industria automotriz**.**
- **IBM:** son las siglas de International Business Machines, una empresa multinacional de tecnología y consultoría.
- **Manipulador:** es un mecanismo formado generalmente por elementos en serie, articulados entre sí, destinado al agarre y desplazamiento de objetos. Es multifuncional y puede ser gobernado directamente por un operador humano o mediante un dispositivo lógico.
- **Nube de puntos:** se refiere a una estructura formada por un conjunto de puntos.
- **Peer-to-peer:** es una topología de red en la que los nodos se comunican entre si como iguales, sin que sea necesaria la existencia de un servidor central.
- **Pixel:** es la unidad menor en la que se puede dividir una imagen digital. Se caracteriza principalmente por su color y su posición.
- **Rover:** es un vehículo de exploración espacial diseñado para moverse a través de la superficie de un objeto astronómico.
- **Servo:** es un motor de corriente continua con la capacidad de ubicarse y mantenerse en una posición determinada.
- **Smartphone:** se refiere a teléfonos móviles cuyas características lo convierten en un pequeño ordenador.
- **Software:** Es la parte lógica del computador y corresponde a un conjunto de instrucciones que le dicen a la parte física qué debe hacer.
- **Sónar:** aparato que detecta la presencia de objetos mediante ondas acústicas.
- **Telepresencia:** situación que se da cuando el usuario de un sistema de teleoperación tiene la sensación de encontrarse en el lugar remoto, se consigue transmitiendo al usuario suficiente información sobre el entorno en el que se encuentra el robot.
- **VGA:** son las siglas de Vídeo Graphics Array y es un estándar para el despliegue de imágenes en un monitor o pantalla de manera analógica.
- **WiFi:** se refiere a la tecnología para conexión de ordenadores y otros aparatos electrónicos sin necesidad de cables.

BIBLIOGRAFÍA

1] K. Yokoi, «La robótica japonesa, presente y futuro,» 2008. [En línea]. Available: http://www.uc3m.es/portal/page/portal/actualidad_cientifica/noticias/conferencia_yokoi.

2] Rafael Aracil, Carlos Balaguer y Manuel Armada, «Robots de servicio,» 2008. [En línea]. Available: http://e-archivo.uc3m.es/bitstream/10016/9855/1/robots_balaguer_riai_2008.pdf.

3] R. Chellali, «Tele-operation and Human Robots Interactions,» 2010.

4] P. G.-R. y. J. Torrijos, «Robots de Seguridad y Defensa,» [En línea]. Available: http://www.disam.upm.es/~barrientos/Curso_Robots_Servicio/R_servicio/Defensa_files/Robots%20de%20Seguridad%20y%20defensa.pdf.

5] «Programa Lunojod,» [En línea]. Available: http://es.wikipedia.org/wiki/Programa_Lunojod.

6] «Misiones espaciales a Marte,» [En línea]. Available: http://astrociencia-universo.blogspot.com.es/2011/07/misiones-espaciales-marte-lista.html.

7] «Los robots de Fukushima,» *El Mundo,* 17 04 2011.

8] Emmanuel Nuño Ortega y Luis Basañez Villaluenga, «Teleoperación: técnicas, aplicaciones, entorno sensorial y teleoperación inteligente,» 2004. [En línea]. Available: http://upcommons.upc.edu/e-prints/bitstream/2117/570/1/IOC-DT-P-2004-05.pdf.

A. C. Correa, «Sistemas robóticos teleoperados,» 2005. [En línea]. Available:
9] http://redalyc.uaemex.mx/src/inicio/ArtPdfRed.jsp?iCve=91101505.

K. Capek, Rossum's Universal Robots, 1921.

1
0]

I. Asimov, Runaround, 1942.

1
1]

«Web oficial de Aldebaran Robotics,» [En línea]. Available:
1 http://www.aldebaran-robotics.com/en/.
2]

J. J. Velasco, «Telesar V, un robot al estilo Avatar desarrollado en Japón,» [En
1 línea]. Available: http://alt1040.com/2012/02/teselar-v-robot-avatar-japon.
3]

Angisoft, «iControlNao,» [En línea]. Available:
1 http://itunes.apple.com/us/app/icontrolnao/id534908200?mt=8.
4]

J. Koenemann, «Whole-Body Imitation of Human Motions,» [En línea].
1 Available: http://hrl.informatik.uni-freiburg.de/papers/koenemann12hrl.pdf.
5]

«Microsoft games exec details how Project Natal was born,» 02 06 2009. [En
1 línea]. Available: http://venturebeat.com/2009/06/02/microsoft-games-executive-
6] describes-origins-of-project-natal-game-controls/.

«WE HAVE A WINNER – Open Kinect driver(s) released,» Adafruit industries, 10
1 11 2010. [En línea]. Available: http://www.adafruit.com/blog/2010/11/10/we-have-

7] a-winner-open-kinect-drivers-released-winner-will-use-3k-for-more-hacking-plus-an-additional-2k-goes-to-the-eff/.

«Web oficial de Kinect para Windows,» [En línea]. Available:
1 http://www.microsoft.com/en-us/kinectforwindows/.
8]

«Kinect: The company behind the tech explains how it works,» 19 06 2010. [En
1 línea]. Available: http://www.joystiq.com/2010/06/19/kinect-how-it-works-from-
9] the-company-behind-the-tech/.

Carlo Dal Mutto, Pietro Zanuttigh y Guido M Cortelazzo, «Time-of-Flight
2 Cameras and Microsoft Kinect™,» 2012, pp. 33-47.
0]

Navab, Victor Castaneda y Nassir;, «Time-of-Flight and Kinect Imaging,» 01 06
2 2011. [En línea]. Available:
1] http://campar.in.tum.de/twiki/pub/Chair/TeachingSs11Kinect/2011-DSensors_LabCourse_Kinect.pdf.

«Kinect: Cómo funciona su micrófono multiarray,» 26 11 2010. [En línea].
2 Available:
2] http://www.pensamientoscomputables.com/entrada/Kinect/microfono/multiarray/como-funciona/xbox-360.

«Web oficial de PCL,» [En línea]. Available: http://pointclouds.org.
2
3]

A. Argyros, «Efficient model-based 3D tracking of hand,» [En línea]. Available:
2 http://www.ics.forth.gr/~argyros/research/kinecthandtracking.htm.
4]

J. Y. a. Z. Z. Zhou Ren, «Hand Gesture Recognition,» [En línea]. Available: http://www.ntu.edu.sg/home/renzhou/HandGesture.htm.
[25]

«Candescent NUI,» [En línea]. Available: http://candescentnui.codeplex.com/.
[26]

«Aligning object templates to a point cloud,» [En línea]. Available: http://www.pointclouds.org/documentation/tutorials/template_alignment.php.
[27]

«Web oficial de ROS,» [En línea]. Available: http://www.ros.org/wiki/.
[28]

Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berge, Rob Wheeler y Andrew Ng, «ROS: an open-source Robot Operating System,» 2009. [En línea]. Available: http://www.robotics.stanford.edu/~ang/papers/icraoss09-ROS.pdf.
[29]

Morgan Quigley, Eric Berger y Andrew Y. Ng, «STAIR: Hardware and Software Architecture,» 2007. [En línea]. Available: http://www.aaai.org/Papers/Workshops/2007/WS-07-15/WS07-15-008.pdf.
[30]

«Web del proyecto STAIR: STanford Artificial Intelligence Robot,» [En línea]. Available: http://stair.stanford.edu/index.php.
[31]

«Web del Personal Robots Program,» [En línea]. Available: http://personalrobotics.stanford.edu/.
[32]

Sánchez Martín FM, Millán Rodríguez, Salvador Bayarri, Palou Redorta J, Rodríguez Escovar F, Esquena Fernández S, Villavicencio Mavrich H., «Historia de la robótica: de Arquitas de Tarento al Robot da Vinci,» *ACTAS UROLÓGICAS ESPAÑOLAS,* 2007.

«Documentación de NAO,» [En línea]. Available: http://www.aldebaran-robotics.com/documentation/index.html.