Universidad Carlos III de Madrid

Integración de un sensor láser 3D en el manipulador móvil Manfred

Proyecto fin de carrera Ingeniería Técnica Industrial: Electrónica Industrial

Autor: Raúl Villajos Rayo

Tutor: Javier V. Gómez González

Título: Integración de un sensor láser 3D en el manipulador móvil Manfred
Autor: Raúl Villajos Rayo
Tutor: Javier V. Gómez González

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día ___ de _____ 20___ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Resumen

La robótica móvil es una de las ramas de la robótica de más investigación y desarrollo en la actualidad. Tiene la ventaja de poder emplearse en cualquier lugar que sea necesario, por esto, es necesarario modelar el entorno que rodea al robot para la navegación.

Existen diferentes lineas de investigación y desarrollo en el modelado del entorno. Las más antiguas se basan en el efecto SONAR con ultrasonidos y RADAR con microondas, y las más modernas se basan en la luz láser, como el LIDAR, y por último la Kinect.

Este proyecto se basa en la utilización de un sensor láser Hokuyo UTM-30lx y un motor Dynamixel EX-106+ para hacer girar el láser, y la creación del software necesario para hacer que estos cumplan las funciones necesarias, con el fin de obtener mapas tridimensionales para la posterior utilización del robot manipulador MANFRED.

Abstract

The mobile robotic is a robotic area of more research and development at the moment. It has the advantage of being used wherever necessary. For this it is necessary model the environment around the robot for navigation.

There are different lines of research and development in environmental modeling. The oldest are based on the SONAR effect with ultrasonic and RADAR with microwave and modern are based on laser light sush as LIDAR and finally the Kinect.

This project is based on the use of a laser sensor Hokuyo UTM-30lx and a motor Dynamixel EX-106+ for rotating laser and the creation of software to make these devices meet the required functions. This is in order to get three-dimensional maps for further use of the robot manipulator MANFRED.

Índice

Índice de figuras

Indice de tablas

1. Introducción

Cuando hablamos de robótica lo primero que se nos suele venir a la mente es la imagen de C3PO y R2D2 de la saga de Star Wars, y es que la la literatura y el cine ha contribuido a la imagen, el interés y la generación de ideas para el desarrollo de la robótica actual. Pero la robótica hoy en día dista mucho de la evolución mostrada en la ciencia ficción y abarca muchos más campos y aplicaciones de los que, a priori, podemos imaginar.

1.1. Breve historia de la robótica

El término robot proviene de la palabra polaca *robota*, que significa trabajos forzados, y se popularizó con la obra RUR (Rossum's Universal Robots) escrita por Karel Capek en 1920. El término robótica lo define Isaac Asimov, como la ciencia que estudia a los robots. Asimov también creó las tres leyes de la robótica. Y es que el hombre, en la ciencia ficción ha imaginado a las robots viajando por el universo, haciéndose con el poder del mundo, usándolos para las tareas del hogar y del trabajo, etc.

Pero la historia de la robótica empieza mucho antes, en el siglo I a.C, el inventor y científico Herón de Alejandría, describió más de cien máquinas y autómatas en Pneumatica y Autómata, construyendo algunos de ellos. Entre sus inventos se pueden destacar la primera máquina de vapor de la historia, una máquina operada mediante una moneda y un órgano de viento. Siglos después, al finalizar la edad media, Leonardo da Vinci diseña un robot humanoide (Caballero Mecánico). Jacques de Vaucanson en 1738 diseña un pato mecánico capaz de "comer", agitar sus alas y "excretar". Sobre el 1800 se inventaron juguetes mecánicos japoneses que sirven té, disparan flechas y pintan, llamados juguetes Karakuri. Durante el siglo XX, y tras las publicaciones de Capek y Asimov, se produjeron gran cantidad de escritos y adelantos en robótica. En 1956 George Devol comercializa el primer robot programable y funda la compañía Unimation, la primera empresa robótica de la historia. En la década de los 60 aparecen las primeras asociaciones de robótica, como destacables, JIRA (Japan Industrial Robot Association), la RIA (Robot Industries Association) y la BRA (British Robot Association).

Fue la aparición del microchip la que lanzó la industria al acabar con los problemas de computación y empezarse a tratar los temas como el cálculo de

trayectorias, sensorización, etc. En la década de los 60 y 70 se desarrollaron las primeras manos y brazos robóticos, enfocados para la producción industrial.

Desde la década de los 80 hasta hoy día, la robótica ha avanzado a pasos agigantados gracias a la tecnología (la velocidad y capacidad de procesadores y memorias, la informática, los nuevos sensores de todo tipo, etc.). La tendencia ha sido la investigación de los robots humanoides, que imitan la forma y conducta de los seres humanos, la imitación de algunos animales (zoomorfos), que pueden ser con patas tipo araña, cabra, etc, o sin patas como las serpientes, y últimamente los robots voladores. Estos tipos de robots han sido diseñados, sobre todo, por parte de compañías japonesas. Ejemplo de esto, son los robots humanoides P1, P2, P3 y ASIMO de HONDA (ver figura 1), y el robot perro Aibo y el robot humanoide Qrio, ambos de la empresa SONY.

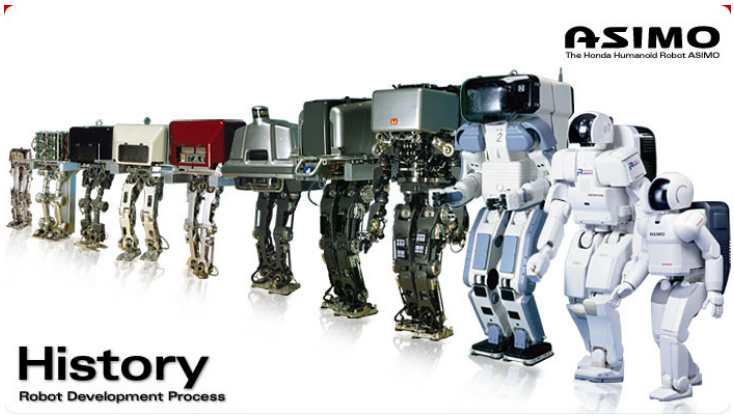


Figura 1: Familia robots Honda.

Por parte de EEUU las investigaciones se han centrado en la ingeniería espacial. Como ejemplo de los robots en ingeniería espacial están las misiones no tripuladas de la NASA como pueden ser los robots Viking y Curiosity, el robot humanoide Robonaut.

(a) Robot Curiosity (b) Robonaut

Figura 2: Robots NASA.

También, por lo general, se ha buscado diseñar robots que cumplan funciones que para las personas resultarían pesadas, peligrosas o dificultosas. Se pueden destacar los robots utilizados en catástrofes, energía nuclear, desactivación de explosivos, etc, y otros más sencillos y comerciales como los robots aspiradora o los limpia-fondos de piscinas.

Actualmente la robótica ha alcanzado su mayor grado de implantación en el ámbito industrial. Los brazos robóticos o manipuladores mueven una gran cantidad de empresas e industria. En esta caso, los robots, están anclados a un punto fijo, pues no tiene la necesidad de moverse. Pero otros tipos de robots como los robots humanoides o los exploradores de la NASA deben tener movimiento, por lo que nos introducimos en la robótica móvil.

1.2. Robótica móvil

La robótica móvil es un campo de investigación reciente y que, en la actualidad, ocupa numerosas líneas de investigación y trabajo en laboratorios de todo el mundo. Su desarrollo supone la integración de disciplinas como la automática, electrónica, informática, inteligencia artificial, estadística, etc.

Los robots móviles tienen la ventaja de poder emplear sus habilidades donde sea necesario. Ésta es la clave de su creciente demanda comercial. En los últimos años se han multiplicado sus aplicaciones en entornos industriales, militares, domésticos y de seguridad. Algunos ejemplos son los cortacésped,

los robots aspiradora, los Rovers de Marte, los robots guía, etc.

Empezó a finales de los años 60 con el proyecto Shakey en el SRI (Stanford Research Institute). El documento "A Mobile Automaton: An Application of Artificial Intelligence Techniques" escrito por N.J. Nilsson en 1969 ya mencionaba la percepción, el mapeado del entorno, la planificación del movimiento. El boom de proyectos de robótica móvil se produjo en la década de los 80 donde fueron apareciendo diferentes lineas de investigación y tres conceptos fundamentales para la robótica movil: SLAM, sistemas no holónomos y arquitecturas de control (Arquitectura Subsumption).

Un robot móvil es un sistema que integra percepción del entorno mediante sensores, toma de decisiones y acción. Para tener esta consideración el robot debe contar con un sistema de locomoción. El sistema más utilizado son las ruedas por su alta eficiencia energética, sencillez y facilidad de navegación. Tienen el problema que son poco eficaces en terrenos pedregosos y con desniveles. Las orugas pueden paliar este problema en algunos terrenos, pero no con demasiados desniveles. También existen otros sistemas como las patas que proporciona mayor movilidad en terrenos abruptos pero es poco eficiente energéticamente. Luego encontramos los robots que imitan el movimiento de los gusanos y las serpientes que permiten el movimiento por terrenos poco accesibles y estrechos. Y por último, los robots acuáticos, imitando el movimiento de los peces o con hélices, y aéreos, imitando el movimiento de las aves, insectos o la aviónica.

Los robots móviles deben ser capaces de conseguir información del entorno, por ello surge la necesidad de dotarlos de sensores que sean capaces de darles información para que, de forma autónoma, se muevan por el entorno evitando los obstáculos y elijan la trayectoria más adecuada. Existen numerosos tipos de sensores diseñados para percibir la información externa de una magnitud física y transformarla en un valor electrónico que sea posible introducir al circuito de control, de modo que el robot sea capaz de cuantificarla y reaccionar en consecuencia.

Suelen incorporar sensores propioceptivos que suministran medidas relativas a su estado: velocidad, incremento de posición y aceleraciones. También existen los sensores inerciales como acelerómetros, giróscopos y magnetómetros, que miden la aceleración lineal, velocidad angular y dan la dirección del norte magnético, respectivamente, que permiten tener un cierto control de la cinemática del robot.

Por otra parte, existen los sensores estereoceptivos para conseguir datos externos del robot. Los más simples de estos sensores son los de contacto. Al chocar el robot contra un obstáculo este sensor es "pulsado" como un interruptor. Pero este tipo no se puede utilizar para una navegación rápida y precisa, pues el robot no puede conocer su situación espacial, solo si se choca o no. Para anticiparse al choque y generar mapas del entorno existen otros sensores, los ópticos y los de ultrasónicos, que utilizan luz infrarroja y ondas sonoras para detectar los obstáculos o calcular la distancia a ellos.

Una vez que el robot cuenta con un sistema locomotor y un sistema sensorial, los problemas son la navegación y la localización. Para resolver estos problemas se suelen realizar cuatro preguntas, "¿a dónde voy?", "¿dónde estoy?", "¿cuál es el mejor camino para ir?" y "¿dónde he estado?". La primera pregunta normalmente es respondida por un humano o por algún subsistema inteligente. Esta tarea se suele considerar como una etapa previa a la navegación. La segunda pregunta se centra en la localización, como averiguar la situación del robot en cada momento mediante la percepción del entorno y la información previa. La tercera pregunta trata de dar solución a uno de los problemas de la navegación de más interes de los investigadores, planear un camino que permita alcanzar el destino especificado. Finalmente, la construcción de mapas es una posible misión de los robots. A parte de estos problemas que presenta la navegación, en la robótica móvil también tienen que centrar las investigaciones para dar solución a la inteligencia y la autonomía del robot, el cómo debe actuar el robot en cada momento.

El problema de localización se puede clasificar según su grado de dificultad: seguimiento de posición (Tracking), posicionamiento global. En el problema del seguimiento de la posición el robot conoce su posición inicial. El objetivo es mantener la localización del robot en el seguimiento de la ruta establecida.

En el problema del posicionamiento global, el robot no conoce su posición inicial y tendrá que autolocalizarse. Los métodos para este tipo de problema se denominan como técnicas de localización global. La dificultad es superior al problema de seguimiento ya que se debe buscar sobre todo el espacio de trabajo.

Para la solución de los problemas anteriores es necesario contar con un mapa del entorno. En algunos casos el mapa se dipone a priori, pero es más común que el robot tenga que construir mapas a partir de los sensores de los que dispone y también que tenga que hacerlo al mismo tiempo que

avanza. Este enfoque se conoce como SLAM (Simultaneous Localization and Mapping). Esto es más complejo, pues se requiere correlación entre la posición estimada del robot y la del mapa.

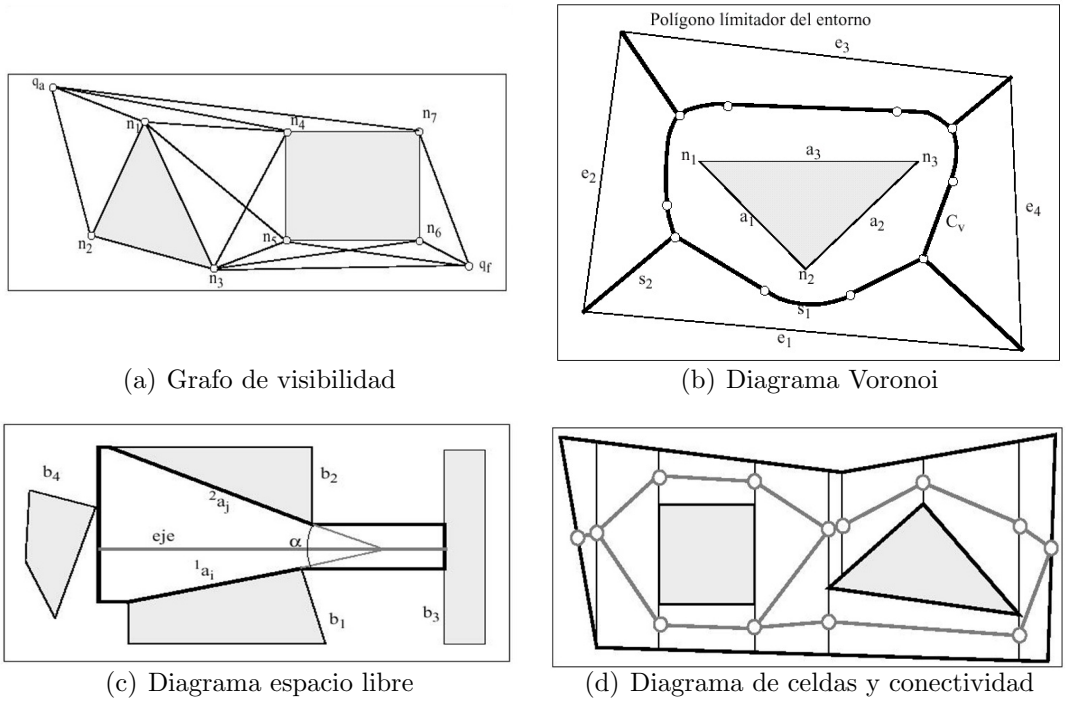
Existen diferentes técnicas para la localización absoluta del robot. La localización con hitos o landmark está basada en elementos del entorno que poseen características distintivas y especiales que el robot puede detectar fácilmente con el fin de localizarse. Otra técnica de localización es la basada en mapas, que se denomina correspondencia entre módelos (Model Matching). Se basa en encontrar correspondencias entre un mapa local que el robot construye mediante los sensores, y un mapa global del entorno que se conoce previamente, o que el mismo robot ha construido ya.

Los diferentes métodos de localización basados en mapas se diferencian en la forma de caracterizar el mapa del entorno. Hay el método de localización con mapas métricos, los cuales usan mapas que describen el entorno en términos métricos referenciados a un sistema de coordenadas global, como pueden ser los mapas de rejillas (grid), o los de características geométricas (features). Otro método es la localización con mapas topológicos, en los cuales se indica la conectividad de los puntos pero no sus relaciones métricas. Por último, el método de correspondencia o Scan Matching, en el que se comparan el mapa que ve el robot con otro mapa adquirido con anterioridad o ya dado y busca las distancias y rotaciones entre ellos.

Hasta aquí hemos visto los problemas y formas actuales de localización. Pero tras la localización es necesario planificar rutas para que el robot pueda moverse de un punto a otro de forma segura.

En primer lugar, existe un enfoque de planificación global y otro de planificación local que deben tenerse en cuenta en un robot. La planificación global se centra en dar una aproximación al camino total y final a seguir, y se realiza antes de que el robot comience la ruta. Y la planificación local funciona a la vez que el robot se desplaza hacia su destino y se centra en proporcionar la ruta exacta paso a paso, evitando posibles obstáculos inesperados.

Existen diferentes métodos de planificación de trayectorias según la forma de los datos que tienen los mapas. Algunos de estos son: grafos de visibilidad, diagramas de Voronoi, roadmap probabilístico (RPM), modelos de espacio libre, mapas descompuestos en celdas y campos potenciales.



(a) Grafo de visibilidad

(b) Diagrama Voronoi

(c) Diagrama espacio libre

(d) Diagrama de celdas y conectividad

Figura 3: Diagramas para planificación de trayectorias.

1.3. Modelado 3D

La navegación y la localización es un problema muy relacionado con la construcción de mapas. Como hemos visto, los robots suelen generar los mapas mientras lo están explorando. Uno de los métodos utilizados para la construción de mapas es la extracción de características, como por ejemplo, objetos estáticos, puertas, esquinas, paredes, etc. Las características suelen contener información semántica y métrica, es decir, el tipo de característica y las dimensiones, color, textura, etc. Las características que suelen ser más útiles son las de geometría simple, ya que son fáciles de detectar y son frecuentes en entornos artificiales. Los modelos más empleados para describir características son las líneas, segmentos, esquinas, bordes y curvas. Ésta es una buena forma de caracterizar puertas, el suelo, el techo, etc.

Para construir los mapas los sensores más utilizados son los de ultrasonidos y los láser. Ambos se basan en el mismo funcionamiento, obtener las

distancias de los puntos del entorno midiendo la distancia que les separa del robot. También se utilizan, junto con los anteriores, otros sensores como cámaras CMOS, obteniendo información del color y las texturas del entorno, y la reciente Kinect con la que podemos conseguir mapas 3D con color y con una alta densidad de puntos. La calidad de los mapas generados dependerá de los sensores utilizados, su cantidad y variedad. Cuanta más información se pueda conseguir con los sensores, más completos serán los mapas.

Hay varios tipos de mapas que se pueden construir y se pueden clasificar de la siguiente manera. En primer lugar, los mapas pueden ser globales, si representan toda el área de acción del robot, o locales, si solo representan el entorno cercano al robot.

Dependiendo de la información que contienen los mapas, se pueden clasificar en mapas geométricos o mapas topológicos y también podemos encontrar una mezcla de ellos, los mapas híbridos. Los mapas geométricos contienen información geométrica, como líneas, celdas, puntos o polígonos. Son el tipo de mapa con más detalle y ocupan más memoria. Dentro de este grupo, podemos encontrar los mapas de objetos que almacenan los obstáculos que el robot puede encontrarse, donde se puede encontrar su forma, su punto central, etc.

Los mapas topológicos representan el espacio como un conjunto de grafos con nodos y arcos, donde los arcos representan lugares de interés y los arcos la forma de ir de uno a otro. Éstos son más compactos pero eliminan mucha información. Dentro de este grupo podemos mecionar los mapas de landmark, mapas de ocupación o mapas de espacio libre. Los mapas de landmark contienen nodos que representan localizaciones particulares fácilmente reconocibles por el sistema sensorial del robot. Los mapas de ocupación, en el que el mapa está dividido en cuadrículas que contienen información sobre si está o no ocupado ese espacio, es decir, si el robot puede ocupar el espacio o no. Los mapas de espacio libre contienen nodos que representan puntos de parada donde el robot puede detenerse para explorar el entorno.

1.4. Objetivos

El objetivo final de este proyecto es proporcionar al robot manipulador MANFRED mapas tridimendionales mediante un sensor láser. Se han tenido en cuenta una serie de condiciones para la elección del hardware y el desarrollo software de este proyecto:

Figura 4: Robot MANFRED.

- El hardware utilizado debe ser comercial.
- Se debe diseñar y construir los elementos de subjección al robot.
- Se debe poder tomar puntos en todo el campo de visión del robot. Por esto se decide poner el láser en la parte superior del robot.
- Se tiene que poder utilizar en exteriores.
- Alimentación común y segura para los dispositivos empleados.
- El software debe estar integrado en ROS, que es el meta-sistema operativo usado por el robot MANFRED.
- Se creará un nodo ROS independiente para cada elemento hardware para usarse induvidualmente si fuera necesario.
- Se debe entregar la nube de puntos a cualquier nodo de ROS del robot que lo precise.

2. Tecnologías actuales

A continuación se pretende describir el estado actual de las tecnologías usadas en este proyecto y dar una visión global de las ventajas e inconvenientes de cada tipo de tecnología.

2.1. Sensor láser 3D

Los sensores láser 3D son dispositivos que analizan un objeto o una escena para reunir datos de su forma, distancias, etc. La información obtenida se puede emplear para producir modelos tridimensionales que se usan en diferentes aplicaciones. El propósito final de un láser 3D es generalmente el de crear una nube de puntos. Estas nubes de puntos describen la posición en el espacio tridimensional de cada punto obtenido por el láser.

Los sensores ópticos son sensibles a las propiedades ópticas de las superficies que están inspeccionando. Problemas con la contaminación del aire y la luz de fondo. Hay dos grupos, físicos y geométricos. Los físicos miden la amplitud de la luz reflejada. Los geométricos miden las distancias y la orientación.

A continuación se exponen los principales tipos de estos sensores que se pueden clasificar en función de la tecnología que usan.

2.1.1. Tipos de láser 3D

Láser basado en el tiempo de vuelo

El funcionamiento de estos sensores se basa en la técnica del "tiempo de vuelo". Este método consiste en determinar la distancia al punto del objeto o escena cronometrando el tiempo de viaje de ida y vuelta de un pulso de luz.

Para conseguir medir en todo el espacio tridimensional se mueve el dispositivo emisor de luz o espejos para desviar el haz de luz hacia donde se quiere medir.

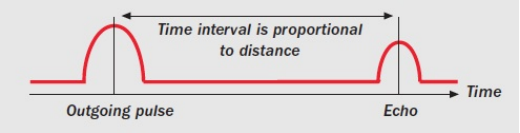


Figura 5: Tiempo de vuelo (Wikipedia).

Las características de estos sensores son: Rápido muestreo, suelen ser equipos de alta precisión, generación de una alta densidad de puntos.

Algunos ejemplos de escáneres basados en el tiempo de vuelo:

- Velodyne LIDAR
- Callidus CP3200
- Leica ScanStation2
- Leica C10
- Mensi GS100/200 (ahora Trimble GX)
- Optech ILRIS
- Riegl (Toda la gama)

Como ejemplo, destaca el láser Velodyne LIDAR que está embarcado en el nuevo coche autónomo que está creando google llamado "Google Self-Driving Car". Este láser 3D tiene 64 rayos láser y gira sobre sí mismo 360º de manera permanente hasta 900 vueltas por minuto para monitorizar todo el entorno del coche, con 1,3 millones de puntos por segundo. Gracias al LIDAR se contruye una imagen tridimensional alrededor del coche, con todo tipo de objetos posicionados (peatones, otros vehículos, farolas, árboles, etc). Tiene un alcance de 50 m para el pavimento y de 120 m para coches y árboles.

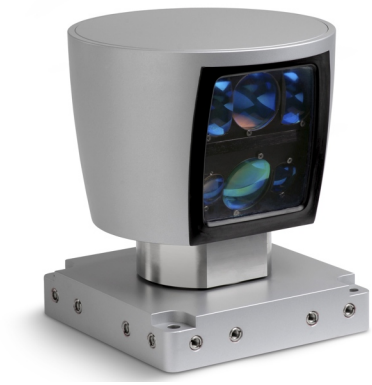


Figura 6: Velodyne Lidar.

Láser basado en la triangulación

Este tipo de láser usa la luz del láser para examinar el entorno. Un haz de luz láser incide en un objeto y se usa una cámara para buscar la ubicación del punto láser. Dependiendo de la distancia a la que el láser incida en una superficie, la reflexión del punto del láser aparece en lugares diferentes en el sensor de la cámara. Esta técnica se llama triangulación porque el punto del láser, la cámara y el emisor del láser forman un triángulo.

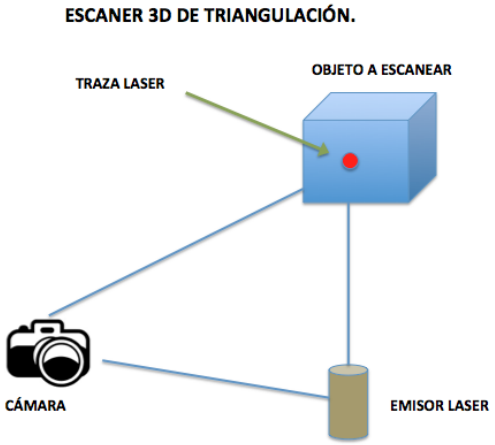


Figura 7: Principio de la triangulación (3dimpresora3D.com).

La precisión de este sistema de medida puede ser muy elevada, pero depende del ángulo del vértice opuesto al láser. Dado que este ángulo depende fuertemente de la distancia entre el emisor láser y la cámara, el aumentar el alcance supone incrementar mucho el tamaño del equipo de medida. En la práctica, el alcance máximo de estos láseres se limita a 20-30 cm.

Algunos ejemplos de escáneres 3D por triangulación:

- Minolta Vivid

Figura 8: Minolta Vidid.

Diferencia de fases

Este tercer tipo de escáner mide la diferencia de fase entre la luz emitida y la recibida, y utiliza dicha medida para estimar la distancia al objeto. El haz láser emitido por este tipo de escáner es continuo y de potencia modulada. El rango y la precisión de este tipo de escáner es intermedio, situándose como una solución entre el largo alcance de los dispositivos basados en el tiempo de vuelo y la alta precisión de los escáneres por triangulación. Su alcance ronda los 200 m en condiciones de poco ruido (baja iluminación ambiente), y su error característico ronda los 2mm por cada 25 m. En algunos modelos el alcance está limitado precisamente por su modo de funcionamiento, ya que al modular el haz con una frecuencia constante, existe ambigüedad en la medida de la distancia proporcional a la longitud de onda de la modulación utilizada. La precisión de la medida también depende de la frecuencia utilizada, pero

de manera inversa a cómo lo hace el alcance, por lo cual estos conceptos son complementarios, y se debe encontrar un punto de compromiso entre ambos, o bien utilizar dos frecuencias distintas (multi-frequency-ranging). La velocidad de adquisición es muy alta, consiguiendo los modelos actuales velocidades de escaneo que oscilan entre los 100.000 y 1 millón de puntos por segundo, en función de la precisión requerida.

Algunos ejemplos de escáneres basados en Diferencia de Fase:

- Faro Photon, Zoom
- Trimble CX (mixto, fase y tiempo de vuelo)
- Trimble FX
- Z+F Imager 5005, 5010

2.1.2. Tecnologías semejantes

También existen otras tecnologías que pueden cumplir el propósito de los sensores láser como mapas del entorno o de objetos, y también complementarse con ellos obteniendo otros datos como el color.

-Ultrasonidos:

Consiste en ondas sonoras de una frecuencia por encima del espectro auditivo del oído humano. El principio de funcionamiento es el tiempo de vuelo. Las ondas ultrasonicas son emitidas, rebotan en los objetos y después son recibidas. Midiendo el tiempo que tarda en volver la onda sonora y conocida la velocidad del sonido se obtiene la distancia. En la figura 9 se puede observar un módulo comercial de ultrasonidos. En el se puede apreciar de que partes consta, un emisor de ultrasonido (altavoz) y un receptor (micrófono). También consta del sistema de medición de tiempo entre la emisión y la recepción y la interfaz hardware.

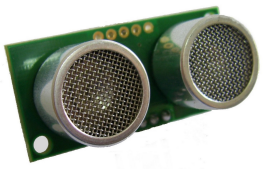


Figura 9: Sensor de ultrasonidos (superrobotica.com).

Pero los sensores de ultrasonidos sufren algunos problemas que lo hacen menos favorables para algunas aplicaciones, como la direccionalidad, la dependencia con la temperatura, la humedad y el aire polvoriento. No permiten antialiasing [1]. Solo son posibiles bajas tasas de adquisición. El rango de medidas es limitado según la orientación.

Los principales usos de esta tecnología son la medición de distancias, caracterización interna de materiales, la detección de posibles anomalías en ingeniería civil, en medicina, por ejemplo, en ecografías, etc. En los últimos años ha proliferado su uso en la industria del automóvil. Los sensores se incorporan en la parte trasera de los automóviles con el fin de controlar las

[1]No se consigue una gran resolución por lo que se producen escalones en la imagen tridimensional formada.

distancias en las maniobras de estacionamiento y prevenir los choques con los obstáculos cuando se conduce marcha atrás.

-Microondas:

Esta tecnología se basa en ondas electromagnéticas definidas en un rango de frecuencias determinado, generalmente entre 1 GHz y 300 GHz. El principio de funcionamiento es el mismo que en los ultrasonidos, el tiempo de vuelo, pero se usan ondas electromagnéticas en lugar de ondas ultrasónicas. Los sensores de microondas son también conocidos por sensores RF, radar o doppler. La tecnología de microondas es utilizada por los radares meteorológicos, de tráfico, etc, para detectar el rango, velocidad, información meteorológica y otras características de objetos remotos, o en el máser, un dispositivo semejante a un láser pero que trabaja con frecuencias de microondas.

Las cámaras de RF ejemplifican el gran cambio que recientemente ha surgido en este tipo de tecnologías. Desempeñan un papel importante en el ámbito de radar, detección de objetos y la extracción de identidad mediante el uso del principio de imágenes microondas de alta resolución, que consiste, esencialmente, en un transmisor de impulsos para iluminar la tarjeta, un auto-adaptador aleatorio de fase seguido por un receptor de microondas que produce un holograma a través del cual se lee la información de la fase e intensidad de la tarjeta de radiación.

Los sensores microondas tienen algunas desventajas. Son peligrosos para la salud y no deben ser usados cerca de las personas, por este motivo sus usos son puntuales. No pueden detectar objetos del rango de pocos milímetros, ya que las ondas de radio que usan tienen una longitud de onda mayor que esos objetos. También existen una multitud de fuentes que pueden crear interferencias, con lo que se produce un error en la medición de las distancias.

-Kinect:

Originalmente, Kinect es un controlador de juego libre desarrollado por Microsoft para la videoconsola Xbox 360. Este dispositivo es una barra horizontal de aproximadamente 23 cm (9 pulgadas) conectada a una pequeña base circular con un eje de articulación de rótula, y está diseñado para ser colocado longitudinalmente por encima o por debajo de la pantalla de vídeo. Cuenta con un sensor CMOS para RGB, un sensor CMOS para infrarrojo de Clase 1, un dispositivo láser que genera un patrón de puntos y segmentos, un micrófono de múltiples matrices y el chip PrimeSensor que se encarga de todos los controles del sistema (captura de movimiento de todo el cuerpo en 3D, reconocimiento facial y capacidades de reconocimiento de voz).

El funcionamiento de Kinect se basa en la información obtenida de las dos cámaras CMOS y el láser de IR. La cámara CMOS de IR detecta los reflejos del láser IR y genera mapas de la intensidad de cada punto o segmento, permitiendo ver la escena en 3D en cualquier condición de luz ambiental. Este proceso se realiza a una distancia de pocos metros y se obtiene una resolución de hasta 1 cm dentro de la dimensión de la profundidad (eje Z) y la resolución espacial (ejes X e Y) es del orden de milímetros. La cámara CMOS de RGB se utiliza para conseguir el color.

Figura 10: Kinect.

Kinect fue diseñado como interfaz de control para videojuegos, pero tras la aparación de controladores de código abierto, investigadores de todo el mundo han desarrollado sus propias aplicaciones. Han sido creadas numerosas interfaces de control por gestos de gran utilidad. Como ejemplo, interfaces de control para manejar el sistema operativo de un PC o interfaces de control para brazos robóticos. También se está utilizando e investigando para

usarlo en el sistema sensorial de robots. Kinect puede proporcionar imágenes tridimensionales con color que aportan gran información para la navegación, generación de mapas y reconocimiento de objetos y gestos en un robot. Un ejemplo de aplicación robótica de Kinect es el robot TurtleBot (ver figura 11), que es una plataforma móvil de bajo coste ideal para educación e investigación.

Figura 11: TurtleBot.

-Cámaras estereoscópicas:

Estas cámaras intentan imitar los ojos humanos utilizando dos cámaras separadas estratégicamente captando la fotografía en el mismo instante y como resultado obteniendo las imágenes 3D. Una de las aplicaciones más antiguas es la visualización y medición del relieve terrestre y el de otros planetas utilizando fotografías aéreas. También son usadas en robótica para dar una visión artificial y con profundidad, como se muestra en la figura 12. En Medicina, uno de los campos donde más se usa, se utiliza en endoscopias, microscopios y en técnicas para visualizar imágenes del interior del cuerpo hunmano.

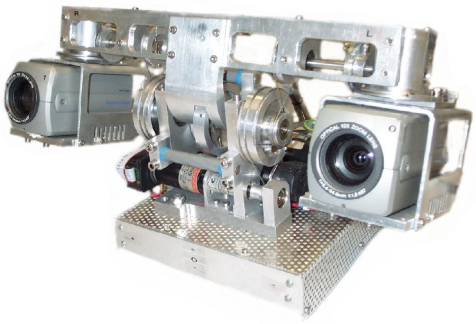


Figura 12: Cámara estereo robótica (Australian National University).

-Femto fotografía:

Recientemente en el MIT (Masachusets institute of tecnology), un grupo de científicos ha desarrollado una cámara que permite tomar 1 trillón de cuadros por segundo. Con esta velocidad de captura de imágenes es posible captar un haz de luz en movimiento. Esta tecnología consiste en iluminación láser de fentosegundos, detectores de precisión de picosegundos y técnicas de reconstrucción matemática. La fuente láser de zafiro titanio emite pulsos regulares en un intervalo de 13 nanosegundos y en diferentes direcciones utilizando un espejo giratorio. Estos pulsos iluminan la escena y la cámara captura la luz que retorna de la escena, incluso después de varios rebotes. Despúes con técnicas matematicas se utilizan los puntos obtenidos para construir la imagen tridimensional. Una de las cosas más impresionante de esta tecnología es que nos permite "ver a través de las paredes". Con la velocidad de la cámara se consiguen obtener los pulsos de luz rebotados detrás de rincones y así reconstruir la imagen que hay detrás de los obstáculos.

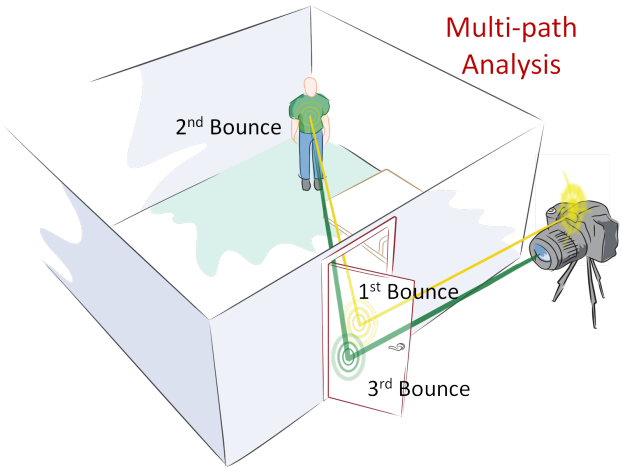


Figura 13: Femto fotografía: viendo detrás de las paredes (MIT).

Por lo tanto, esta nueva tecnología define a la imagen, abriéndole un nuevo horizonte para su desarrollo en diferentes campos como la imagenología médica y el procesamiento digital de imágenes. También permitirá desarrollar tecnologías de búsqueda de sobrevivientes en ambientes peligrosos como incendios o incluso evitar las colisiones de los automóviles.

2.1.3. Aplicaciones

Industria

El escáner 3D ha encontrado una aplicación insustituible en el control dimensional de fabricación de componentes que requieren tolerancias muy estrictas, como álabes de turbina, mecanizados de alta precisión, estampación y matricería ... Las piezas se escanean y la nube de puntos se compara con el modelo teórico, permitiendo un control muy minucioso sobre la producción. También se utiliza para escalar diseños a partir de modelos creados a mano.

Ingeniería inversa

La ingeniería inversa de un componente mecánico requiere un modelo digital preciso de los objetos para ser reproducido. Mediante el escaneo 3D se consigue el mapa del objeto para después reproducirlo con la maquinaria adecuada, como fresadoras y tornos CNC.

Documentación

Los escáneres 3D permiten obtener modelos precisos de la situación real de un edificio o instalación, de manera que se pueden realizar proyectos de documentación o mantenimiento basados en su situación real. Además, permiten comparar la evolución temporal de un objeto, permitiendo identificar deformaciones, movimientos, etc.

Entretenimiento

Escáneres 3D son usados por la industria del entretenimiento para crear los modelos 3D digitales para películas y videojuegos. En caso de que donde un equivalente de mundo verdadero de un modelo exista, es mucho más rápido escanear el objeto físico que crear manualmente el modelo 3D por medio de software de modelado. Frecuentemente, los artistas esculpen los modelos físicos de lo que ellos quieren y los escanean en forma digital antes de pasarlos directamente a modelos digitales en una computadora.

Patrimonio cultural

Ha habido muchos proyectos de investigación que emprendieron el escanear sitios y artefactos históricos. La técnica de escaneo láser contribuye a la documentación y mantenimiento de edificaciones, monumentos y otros elementos históricos. Además, puede ser una herramienta para la divulgación de turismo histórico a través de modelos virtuales.

Robótica

Con el escaneo 3D se consigue dar al robot un mapa tridimensional del entorno en el que se encuentra o de objetos, con el fin de facilitarle la navegación y el reconocimiento de objetos, generando mapas y modelos 3D. Un ejemplo de aplicación es el láser Velodyne Lidar usado en el coche de Google que le permite disponer de imágenes tridimensionales de su entorno, reconocer peatones, otros coches, señales, etc.

2.2. Nubes de puntos

Una nube de puntos es un conjunto de puntos en un sistema de coordenadas tridimensional. Estos puntos se identifican habitualmente como coordenadas X, Y, y Z y son representaciones de la superficie externa de un objeto o una escena.

Las nubes de puntos tienen múltiples aplicaciones, entre las que se incluyen la elaboración de modelos tridimensionales en CAD de piezas fabricadas, visualización, animación, texturización y aplicaciones de personalización masiva. Una aplicación en la que las nubes de puntos se utilizan directamente es la metrología y en la inspección industrial. La nube de una pieza fabricada se alinea a un modelo CAD (o a otra nube de puntos) y se comparan para verificar las diferencias. Éstas se visualizan como mapas de color que proporcionan un indicador visual de la desviación entre la pieza fabricada y el modelo CAD. Las dimensiones geométricas y tolerancias también se pueden extraer directamente de la nube de puntos.

La robótica es otra de las aplicaciones en donde las nubes de puntos son utilizadas ampliamente. Son un soporte vital a la hora de crear mapas y modelos, para la navegación de los robots y el modelado.

En las figuras 14 y 15 se presentan dos nubes de puntos como ejemplo. La primera tomada con Kinect y la segunda con Velodyne Lidar desde el coche del MIT (MIT DARPA Urban Grand Challenge), aunque esta nube ya está procesada para resaltar algunas características.

Figura 14: Nube de puntos tomada con Kinect (willowgarage.com).

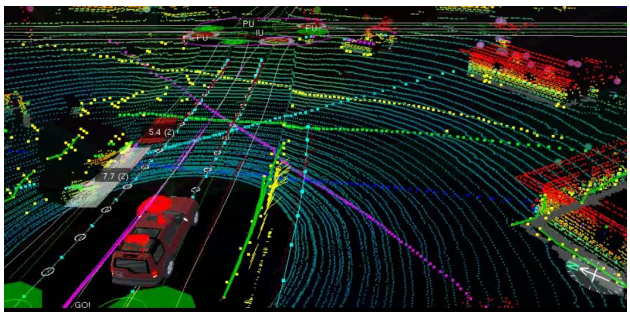


Figura 15: Nube de puntos tomada con Velodyne LIDAR (MIT DARPA Urban Grand Challenge).

Para el manejo de estas nubes de puntos la herramienta más usada es “Point Cloud Library “(PCL), que es una librería independiente, a gran escala, de código abierto y procesamiento de nubes de puntos. PCL está liberado bajo los términos de licencia BSD, y por lo tanto libre para uso comercial y de investigación. Están apoyados financieramente por Open Perception, Willow Garage, NVIDIA, Google, Toyota, Trimble, Urban Robotics, Honda Research Institute, Sandia, Dinast, Optronic, Ocular Robotics, Velodyne, Fotonic, Leica Geosystems, y MKE.

2.3. ROS

ROS (Robot Operating System) es un meta-sistema operativo para robots de código abierto desarrollado por la empresa Willow Garage. ROS provee librerías y herramientas para ayudar a desarrolladores software a crear aplicaciones robóticas. ROS proporciona abstracción de hardware, drivers de dispositivos, librerías de visualización, paso de mensajes, manejo de paquetes, etc, y tiene licencia de código abierto BSD. Se comporta como una red de procesos (nodos) peer-to-peer que se ejecutan individualmente y se acoplan entre sí utilizando la estructura interna de comunicaciones que proporciona ROS, pudiendo ser ésta de 3 maneras diferentes: comunicación síncrona en modo cliente-servidor a través de los servicios, comunicación asíncrona por envío continuo de datos a través de topic o almacenamiento de datos en servidores de parámetros.

Una de las cosas importantes que proporciona ROS es la sencilla creación de comunicaciones entre los diferentes nodos y aplicaciones dentro de un sistema robótico. Mediante varias líneas de código ROS crea las comunicaciones que de otra manera habría que programar tediosamente mediante pipes, memoria compartida y sockets. Está diseñado para ser lo más ligero posible, y para que el código que se escribe para ROS pueda ser utilizado con pocos cambios en otras plataformas. Está integrado por ejemplo con OpenRAVE(simulador), Orocos(librerías) y Player(interfaz hardware). Es independiente del lenguaje de programación utilizado, permitiendo el uso de la mayoría de los lenguajes más comunes. Hoy en día existen nodos implementados en Python, C++ o Lisp . Tiene una unidad de trabajo orientada a las pruebas llamada Rostest que facilita la corrección de errores. Es apropiado para sistemas grandes en los que se necesita desarrollar numerosos módulos y luego ejecutarlos a la vez. Actualmente Ros solo funciona con sistemas basados en Unix. Está bien probado en Ubuntu y Mac OS.

ROS puede agruparse en tren niveles: el sistema de archivos, el esquema de ejecución y el nivel de la comunidad.

Sistema de archivos: Son los recursos que se pueden encontrar en el disco duro. Por ejemplo:

- Packages: Es la unidad principal de organización de software en ROS. Puede contener procesos de ejecución, librerías o archivos de configuración.

- Manifest: Proporciona la información sobre un package.
- Stack: Es un conjunto de packages que proporciona una funcionalidad concreta.
- Messages: Archivo donde se define la estructura de datos de un mensaje.
- Services: Archivo donde se define la estructura de datos de un servicio.

Esquema de ejecución: Es una red peer-to-peer de procesos ejecutándose a la vez. Los conceptos básicos son:

- Nodes: Son la unidad básica del procesado.
- Master: Es el nodo principal de ROS. Guarda lo necesario para que los nodos se encuentren y se comuniquen.
- Parameter server: permite almacenar datos de forma centralizada, forma parte del Master.
- Messages: Intercambio de mensajes entre nodes.
- Topic: Es un nombre que se utiliza como identificador. Son usados por el Master para comunicar los mensajes de los nodos(publicador y subscriptor).
- Services: Modelo de comunicación cliente-servidor. Un nodo ofrece un servicio bajo un nombre y responde las peticiones de un cliente.

Hay muchos desarrolladores que utilizan ROS para manejar los robots como PR2 de la empresa Willow Garage, Robonaut 2 de la NASA, TurtleBot, y muchos más.

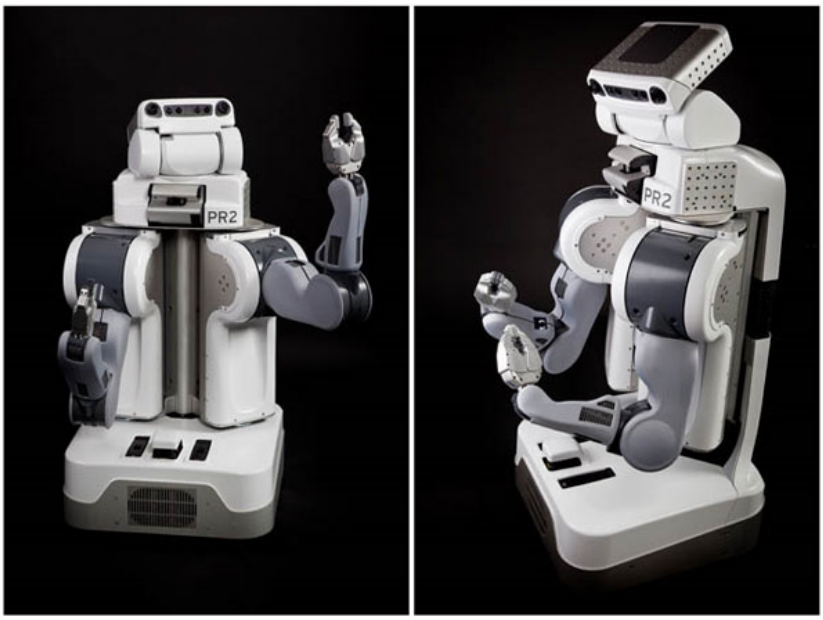


Figura 16: Robot PR2.

También existen otros sistemas de robótica similares a ROS. Estos son algunos: Orocos, orca, Moos, Mrtp, OpenMora, Carmen, Yarp, Player, Stage, Gazebo, Urbi.

3. Hardware empleado

El hardware empleado es el siguiente:

1. Láser Hokuyo UTM-30lx (de 2 dimensiones)
2. Motor Dynamixel EX-106+
3. Tarjeta de alimentación (por debajo en la imagen)
4. Acoplamiento del láser al motor
5. Placa soporte conjunto

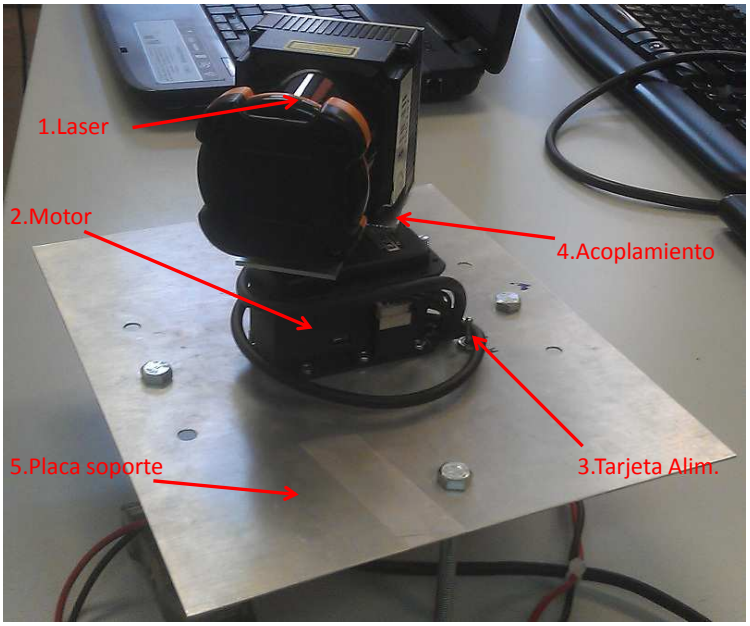


Figura 17: Hardware montado.

En las siguientes secciones se explicará el diseño del láser 3D y cada elemento por separado.

3.1. Robot manipulador móvil MANFRED

El manipulador móvil MANFRED se presenta en la Figura 18. Se trata de un manipulador con 8 g.d.l. Consta de una base móvil de tipo diferencial y 2 grados de libertad, junto con un brazo ligero de 6 grados de libertad y diseño antropomórfico. MANFRED está diseñado para ser capaz de desenvolverse de forma autónoma por entornos diseñados para personas como: pasillos, salas, despachos, etc., donde en ocasiones resulta necesario abrir y atravesar puertas, evitar obstáculos o coger y manipular objetos. Todo esto requiere que el robot integre los siguientes aspectos:

- Todas las capacidades básicas de un robot móvil para desplazarse de forma segura y autónoma por el entorno.
- La coordinación motora entre base y manipulador.
- La coordinación sensorial necesaria para poder manipular objetos.

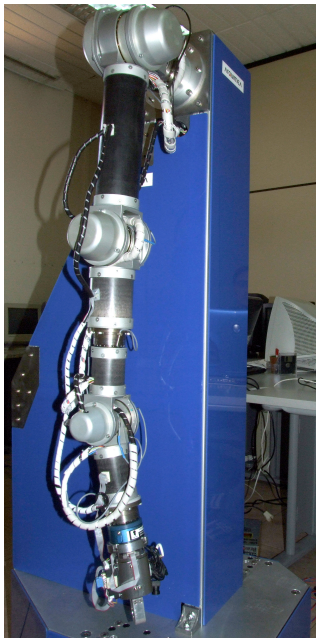


Figura 18: Robot MANFRED.

Los subsistemas que constituyen el robot MANFRED son:

1. **Estructura del robot:** Manfred está formado por una estructura metálica que permite integrar todos los componentes que necesita para su funcionamiento y que se puede dividir principalmente en tres partes:

   - **Una base móvil:** Está constituida por dos plataformas de acero con un diámetro de unos 61 cm y una altura en torno a 65 cm. Está equipada con ruedas que le permiten desplazarse.
   - **Un torso fijo:** Bastidor que aloja todo el cableado de potencia y de sensores, y soporta el brazo y los diferentes sensores.
   - **Un brazo manipulador ligero:** El brazo manipulador LWR-UC3M-1 constituye el elemento fundamental del robot manipulador Manfred. Está compuesto por elementos rígidos conectados por medio de articulaciones de revolución que le permiten alcanzar un total de 6 g.d.l.

2. **Sistema sensorial:** El sistema sensorial permite transformar las variables físicas que caracterizan el entorno en un conjunto de datos que serán procesados por otros módulos como por ejemplo los encargados de la localización, del sistema de seguridad, el planificador de movimiento, etc. Este sistema consta de:

   - **Telémetro láser bidimensional PLS de Sick:** 180º de apertura que permite la adquisición de datos en 2D. Para lograr la portabilidad a tres dimensiones está instalado sobre un chasis motorizado que permite su movimiento vertical dentro de un rango de 45º.

Figura 19: Sick 3000.

- **Cámaras color:** El sistema incorpora una cámara Sony EVI-D100 para reconocer los objetos y estimar su posición respecto al robot. Esta cámara se sitúa en la parte frontal del cuerpo del manipulador móvil. El robot también incorpora una minicámara Sony B/N XC-ES50CE situada sobre la muñeca del manipulador. Esta cámara se utiliza en las tareas de manipulación cuando el brazo se encuentra situado delante del objeto a manipular de forma que interfiere en el campo de visión de la cámara Sony.

Figura 20: Cámaras color.

- **Camara 3D:** El robot también incorpora una cámara que le permite obtener una imagen 3D que consiste en una matriz de distancias a los distintos objetos que se encuentran delante. La información de la cámara 3D y la cámara de color convencional se fusiona para poder realizar una segmentación de objetos más precisa, facilitando así la manipulación de éstos.

- **Sensor fuerta/par:** Para la interacción con el entorno en las tareas de manipulación el robot Manfred porta en el extremo del brazo un sensor fuerza/par JR3 modelo 67M25A-U560. El sensor se sitúa entre el extremo del brazo y la pinza o elemento terminal. Este dispositivo tiene como características relevantes una capacidad de carga de hasta 11Kg, un peso de 175g y proporciona medidas de fuerza y par en 3 ejes que se pueden utilizar en el lazo de control de fuerza del manipulador móvil.

Figura 21: Sensor fuerza/par.

- **Sensores cinemáticos:** La principal función de los sensores cinemáticos es aportar información sobre la posición, pudiendo obtener en el sistema de control las variables derivadas (velocidad y aceleración). Estos sensores se encuentran principalmente acoplados a los ejes de los motores proporcionando información del giro de éstos en forma de cuentas de encoder. En el caso de los motores del brazo, para complementar a los sensores cinemáticas se incorporan unos sensores de detección de home que permiten a través de una rutina de arranque inicial establecer las posiciones absolutas de cada articulación del brazo. Los sensores cinemáticos utilizados son encoders ópticos de alta resolución de la casa HP con referencia HEDS550. En cuanto a los sensores de home se han utilizado sensores inductivos de 3 milímetros de diámetro y una distancia de detección de 1 milímetro, alojados en cada articulación. El principio básico de estos sensores inductivos se basa en la detección de materiales ferromagnéticos mediante la variación de flujo que provoca su presencia cerca del área de detección del sensor.

3. **Sistema locomotor:** El sistema locomotor del manipulador móvil está constituido por la base móvil citada anteriormente, a la que se incorporan cinco ruedas: tres ruedas de apoyo que mejoran la estabilidad del robot y facilitan su movimiento, y dos ruedas motrices asociadas con motores sin escobillas y sus correspondientes servoamplificadores. La ruedas motrices dan lugar a un desplazamiento de tipo diferencial que permite al robot girar sobre si mismo.

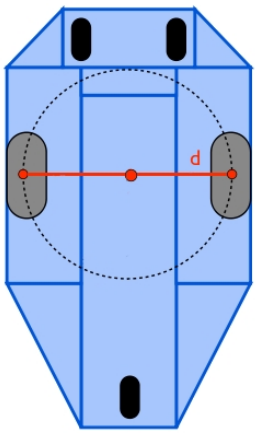


Figura 22: Base de MANFRED.

4. **Sistema de alimentación:** La base móvil alberga el sistema de alimentación compuesto por baterías que dotan de autonomía a todo el robot. Consiste una conexión serie de 4 baterías de 12 V para proporcionar una tensión de alimentación de 48V en continua. Las baterías seleccionadas son el modelo LC-X1242P de Panasonic, que proporcionan una tensión de salida de 12V y una capacidad de 42 A/h. Además, como sistema de seguridad, lleva instalado un sistema de monitorización de las baterías a través de un micro-controlador PIC16F818, que permite medir en todo instante la tensión proporcionada por las baterías y la corriente que circula por ellas. Este sistema permite comunicar continuamente el estado de la alimentación al ordenador de control, así como realizar una parada automática controlada de los motores del robot en caso de baja tensión de alimentación o de superar una corriente umbral.

5. **Brazo manipulador LWR-UC3M-1:** Con objeto de realizar las tareas de manipulación, Manfred está equipado con un brazo robótico ligero de seis grados de libertad, el cual ha sido íntegramente diseñado en la Universidad Carlos III de Madrid. Las características principales de este brazo manipulador son: redundancia cinemática similar al brazo humano, masa del conjunto de 18 kilogramos, capacidad de carga máxima de 4,5 kilogramos en el extremo del brazo, relación carga/peso entre 1:3 y 1:4 y alcance en torno a 955 milímetros.

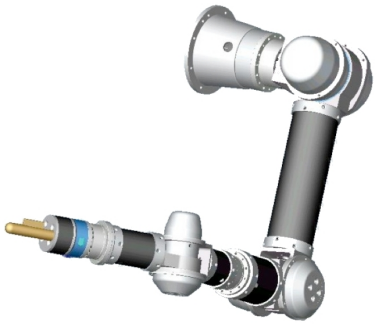


Figura 23: Brazo manipulador LWR-UC3M-1.

Dicho brazo se monta lateralmente de forma que el sistema de visión y la telemetría láser 3D puedan percibir lo que se desea manipular sin ser obstaculizados por el brazo. Las articulaciones del brazo están compuestas por motores sin escobillas brushless de corriente continua que permiten alimentar directamente en continua a los accionadores, una

etapa de reducción de velocidad y aumento de par de tipo Harmonic Drive, más concretamente de la compañía alemana Harmonic Drive AG en su categoría HFUC-2UH que incorporan rodamientos de salida integrados.

6. **Sistema de control:** Manfred tiene 8 motores que le permiten mover tanto su base (2 motores) como su brazo robótico (6 motores). Para que estos motores se muevan correctamente, es necesario que se realice un control continuo sobre ellos. En este caso el control lo lleva a cabo la tarjeta controladora PMAC2-PCI.

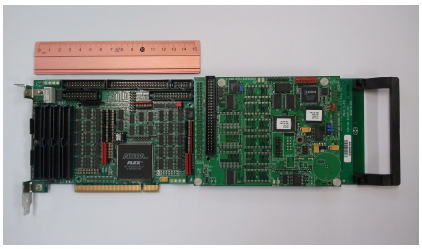


Figura 24: Tarjeta controladora PMAC2-PCI.

La tarjeta controladora de múltiples ejes PMAC2-PCI (Programmable Multi-Axis Controller) de la compañía Delta Tau Data Systems, Inc. es un dispositivo de alto rendimiento capaz de comandar hasta 8 ejes simultáneamente con un alto nivel de precisión. Gracias a sus más de 1000 variables de configuración y a la capacidad de cómputo de sus procesadores digitales de señales actuales, la tarjeta PMAC2-PCI puede ofrecer una relación precio-rendimiento muy satisfactoria. El DSP que incorpora la tarjeta PMAC2-PCI es el modelo DSP56002 de 24 bits y frecuencia de trabajo de 40 MHz de la firma Motorola.

7. **Sistema de procesamiento:** Un manipulador móvil es un sistema de enorme complejidad que debe ser capaz de procesar e interpretar una gran cantidad de información sensorial y cerrar varios lazos de control simultáneamente. Esta complejidad se traduce en crecimiento del software que se desarrolla. Una de las características principales sobre la que se ha estructurado la arquitectura software del sistema es la modularidad, de manera que se favorece el desarrollo y el mantenimiento del sistema software del robot. Para ello, se han desarrollado

módulos funcionales que puedan trabajar conjuntamente mediante el intercambio de datos pero que se ejecuten como procesos completamente independientes. Con esta estructura modular se facilita el desarrollo, comprobación y actualización de los diferentes algoritmos. Cada uno de los módulos funcionales se diseña con capacidad para el auto-diagnóstico, de esta forma se dispone de un primer control de fallos en cada uno de los módulos.

La figura muestra un esquema de la arquitectura de control diseñada para el manipulador robótico Manfred (Ansari, 2011). Al tratarse de módulos de ejecución independientes, cada tarea o cada fase de una tarea definirá el conjunto de módulos necesarios para su desarrollo. El computador asignado al sistema de control coordinado se encarga de las funciones básicas de control del sistema y en lleva incorporado una tarjeta controladora de ocho ejes con la que se comunica vía memoria de doble puerto. Dicha tarjeta permite cambiar dinámicamente los parámetros y el programa de control que se ejecuta para disponer de un sistema flexible que implemente técnicas avanzadas de control. Esta tarjeta realiza el control de la base (2 g.d.l.) y del brazo (6 g.d.l.).

3.2. Diseño láser 3D

La idea general de este diseño es la utilización del sensor láser Hokuyo de 2 dimensiones. Para conseguir puntos en las 3 dimensiones se hará girar el láser con el motor Dynamixel. El láser Hokuyo mide en un plano vertical, el cual se hace girar para obtener diferentes planos de medida.

Con el fin de conseguir puntos en todo el entorno del robot, el láser 3D irá situado en la parte superior del robot. De esta manera el cuerpo del robot nunca se interpone entre el láser y los puntos a medir.

Se puede ver en la figura 25 un esquema del diseño del láser 3D.

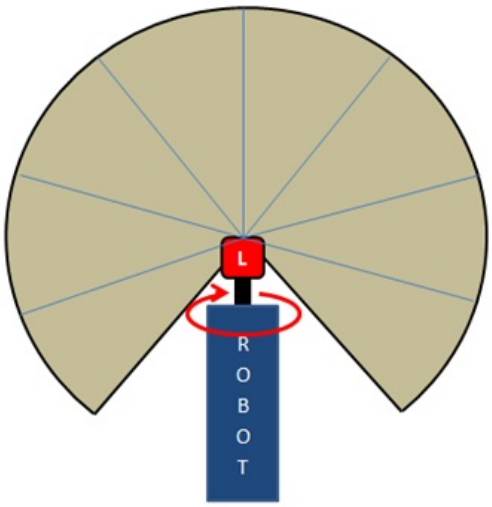


Figura 25: Diseño láser 3D.

A continuación se presentan las transformaciónes geometricas que serán requeridas para obtener los puntos 3D en este diseño.

3.3. Transformaciones geométricas

Para obtener una nube de puntos XYZ debemos transformar los datos obtenidos por el láser y el motor. Se puede observar en la figura 26 la representación geométrica de la captura del láser. El láser entrega 1080 medidas

las cuales se representan como distancias r y sabemos su ángulo θ porque la resolución del láser es de 0,25º. También conocemos el ángulo φ, que es el ángulo de giro del motor.

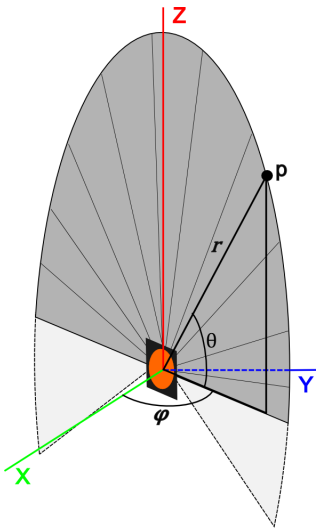


Figura 26: Transformaciones nube de puntos.

En la captura de una nube de puntos, miestras el motor gira se capturan medidas del láser constantemente. Cada vez que se toman las 1080 medidas del laser correspondientes a un plano, se guarda tambien el correspondiente ángulo del motor. Cuando el motor termina de girar tenemos todos los datos y podemos hacer las transformaciones.

Para convertir los datos a una nube de puntos XYZ se usan las siguentes ecuaciones:

$$x = r \cdot cos(\theta) \cdot cos(\varphi) \tag{1}$$

$$y = r \cdot cos(\theta) \cdot sen(\varphi) \tag{2}$$

$$z = r \cdot sen(\theta) \tag{3}$$

3.4. Láser Hokuyo UTM-30lx

Figura 27: Láser Hokuyo utm-30lx.

El láser Hokuyo UTM-30lx es el sensor de mayor gama que tiene la empresa Hokuyo, al ser el que tiene una mayor resolución angular y mayor velocidad de barrido. Es un sensor que tiene un rango de medida desde 0.1 hasta 30 metros, pudiendo llegar en óptimas condiciones hasta 60 metros. Tiene un rango angular de 270º, en el que se pueden tomar 1080 medidas gracias a su precisión angular de 0,25º . Estas medidas se toman en un tiempo de 25ms gracias a la alta velocidad de rotación del motor del láser, 2400 rpm. En la figura 28 se muestra una representación del rango de medida del láser.

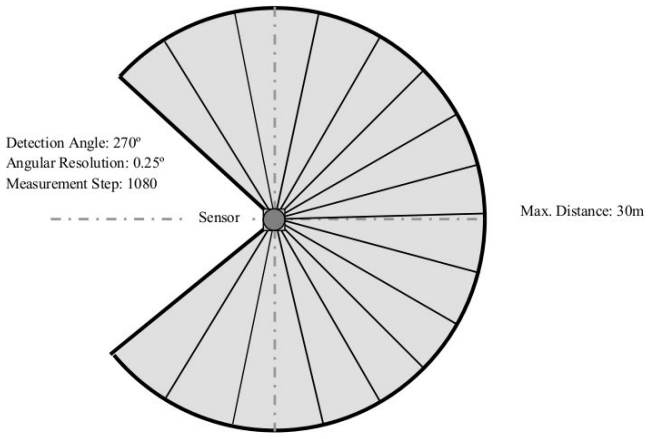


Figura 28: Rango de medida Hokuyo.

Este láser es idoneo para esta aplicación. Tiene una velocidad de adquisición de datos suficiente (25 ms) como para tomar todas las medidas necesarias en pocos segundos (4,5 s si se toma una medida por cada grado de rotación en 180º).

La precisión del láser para la aplicación es más que suficiente. Si transformamos la precisión angular en precisión lineal según la ecuación , como se puede apreciar en la figura 29 y los resultados en la tabla 1, vemos que para hacer mapas de, por ejemplo, una sala grande, la precisión lineal a 10 metros (sala de medidas considerables) es de unos 4,3 cm. Con esta precisión podremos mapear sobradamente paredes, techos, puertas, ventanas e incluso mesas, sillas, personas, etc.

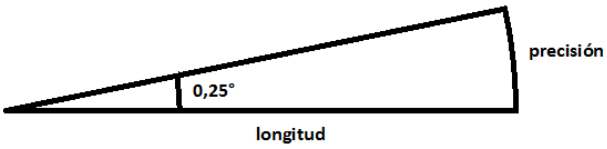


Figura 29: Calculo precisión lineal láser Hokuyo.

$$precision = radianes * radio = (2\pi * 0,25)/360 * radio \tag{4}$$

Alcance (m)	Precisión lineal (cm)
1	0,43
5	2,2
10	4,3
20	8,7
30	13

Tabla 1: Preciones lineales láser Hokuyo

Para exterirores, si cogemos el alcance máximo para cualquier condición de 30 metros, la precisión lineal es de unos 13 cm. Con esta precisión podremos mapear los objetos lejanos que nos encontráramos. Y para cortas distancias la precisión lineal es de unos 4 mm, suficiente para mapear objetos pequeños.

La cantidad de datos que toma este láser es alta, consiguiendo 1080 puntos cada vez que hace un barrido. En cuanto a la interfaz de comunicaciones con el PC, el láser utiliza USB 2.0. Además, el fabricante facilita las librerias de programación en lenguaje C++, lo que facilita la integración con el sistema operativo ROS, el cual usa el robot MANFRED.

En la tabla 2 se muestran las especificaciones del sensor de forma más detallada:

Tipo de fuente de luz	Semiconductor láser λ=905nm Clase 1
Voltaje de alimentación	12V $\pm$ 10 %
Consumo de corriente	0.7A nominal, 1A máximo
Potencia de consumo	Menos de 8W
Rango de detección	0.1m a 30m (hasta 60m óptimas condiciones)
Precisión con 3000lx	$\pm$30mm (0.1m a 10m)
Precisión con 10000lx	$\pm$50mm (0.1m a 10m)
Angulo de escaneo	270ºC
Resolución angular	0,25º (270/1080)
Velocidad escaneo	25ms (velocidad de motor: 2400rpm)
Interfaz comunicaciones	USB Ver2.0 Full Speed (12Mbps)
Salida	Salida síncrona 1-point
Condiciones ambientales	-10º a +50º / menos de 85 % humedad (sin niebla)
Efecto del entorno	Distancia medida será menor bajo lluvia, nieve y sol directo
Resistencia vibraciones	10Hz a 55Hz (doble amplitud 1.5mm en cada eje) durante 2h 15Hz a 200Hz (98m/s2, barrilo 2min cada eje) durante 2h
Resisntencia impacto	196m/s2 en cada eje, 10 veces.
Estructura de protección	Optica: IP64
Aislamiento	10MΩ 500VDC
Peso	210g (sin cable)
Material envoltura	Policarbonato
Dimensiones (WxDxH)	60mmx60mmx85mm

Tabla 2: Especificaciones láser Hokuyo

3.5. Motor Dynamixel EX-106+

Figura 30: Motor Dynamixel.

Con este motor se pretende rotar el sensor láser Hokuyo para conseguir medir en diferentes planos y así obtener puntos de todo el espacio tridimensional.

Se ha elejido este motor por su robusted, velocidad variable, interfaz sencilla de comunicaciones, funcionalidad y el amplio uso en la comunidad opensource y por ello gran variedad de librerias.

Los requerimientos mecánicos están más que cubiertos con su par de 107 Kg por cm y su velocidad de 91rpm, suficientes para mover el láser. La velocidad variable del motor permite tomar nubes de puntos más o menos densas en función de las necesidades, ahorrando tiempo de adquisición si fuera necesario. Su iterfaz USB 2.0 lo hacen manejable, fácil de conectar en cualquier PC y comunicación sencilla con las librerías. A parte de esto, el motor ofrece una gran amplitud de funcionalidades entre las que destacan:

- Obtener posición en escala 0 a 4095 o en escala gradual 0º a 250º.
- Obtener velocidad en escala 0 a 1023, con 0 como máxima velocidad.
- Mover a una posición con velocidad dada o mover incrementos.
- Obtener errores de comunicaciones y de hardware.

A continuación se muestran las especificaciones del sensor de forma más detallada:

Peso	154g
Dimensiones	40.1mm x 65.3mm x 50.1mm
Resolución	0.06º
Relación de reducción	184:1
Torque	107Kg x cm
Velocidad sin carga	91 rpm (a 18.5V)
Rango de movimiento	0º a 251º
Temperatura de funcinamiento	-5ºC a +80ºC
Voltaje de funcionamiento	12V a 18.5V (14.8V recomendado)
Protocolo de comunicación	RS485 Asíncrono (8bit, 1stop, No paridad)
Enlace físico	RS485 Multi Drop Bus
ID	254 ID (0 a 253)
Velocidad de comunicación	7343 bps a 1 Mbps
Feedback	Posición, temperatura, carga, voltaje de entrada, etc
Corriente de standby	55 mA

Tabla 3: Especificaciones motor Dynamixel

La velocidad del motor se controla mediante el uso interno de 10 bits, lo que hace que el motor se controle entre 0 y 1023 (velocidad relativa). La velocidad máxima del motor es 91rpm. En este proyecto, por motivos de seguridad para el hardware, usaremos velocidades de entre 10 y 100. A continuación se presenta una tabla con los valores de velocidad del motor para determinados valores de los 10 bits.

Velocidad relativa	Velocidad real	Velocidad relativa	Velocidad real
10	1,11	50	5,55
20	2,22	60	6,66
30	3,33	70	7,77
40	4,44	80	8,88

Tabla 4: Velocidades motor Dynamixel

3.6. Tarjeta de alimentación

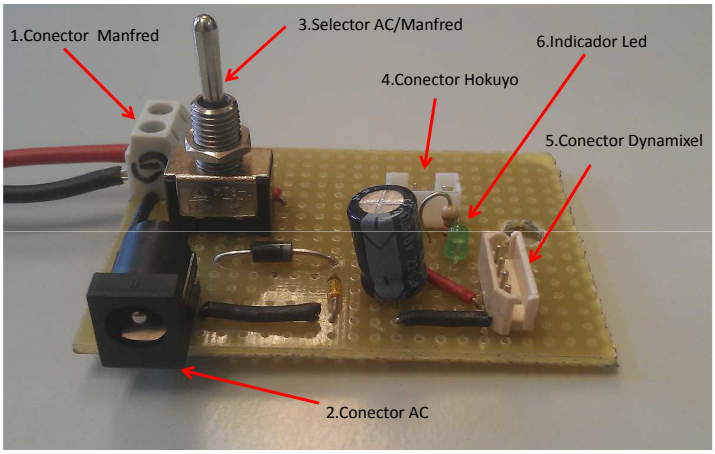


Figura 31: Tarjeta de alimentación.

La tarjeta electrónica se diseña y fabrica para unificar y facilitar las conexiones de alimentación del láser Hokuyo y el motor Dynamixel. Ambos dispositivos pueden alimentarse con el mismo voltaje por lo que las dos alimentaciones cuelgan del mismo punto. Está compuesta por conectores específicos para cada dispositivo por lo que no se pueden confundir las posiciones. También tiene funcionalidades de seguridad. Solo se permite una polaridad de alimentación mediante un diodo en serie y la alimentación está limitada mediante un diodo zener. Hay dos posibilidades de alimentación que se seleccionan mediante un switch. Una desde un adaptador de alimentación de red de 220V a 12V, y otra desde la alimentación que proporciona manfred de 12V.

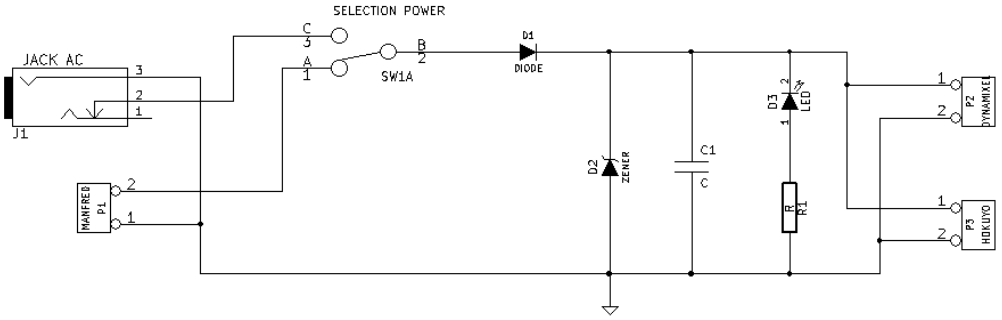


Figura 32: Esquema tarjeta de alimentación.

3.7. Acoplamiento del láser al motor

Figura 33: Placa acoplamiento del laser al motor.

Esta placa se diseña y se fabrica para el acoplamiento del láser Hokuyo al motor Dynamxel. Debe conseguir que el plano de medida del láser gire sobre su centro, de lo contrario se producirían datos erroneos en las medidas y se deberá tratar el error por software.

Para conseguir el acoplo de ambos dispositivos en la posición deseada la placa tiene forma de L. Y para que la placa tenga la suficiente resistencia se fabrica en chapa de aluminio de 2 mm de espesor. También consta de dos agujeros pasantes para los cables del láser.

Montaje Para el montaje del conjunto se siguen los siguientes pasos:

1. El motor se acopla por la parte exterior de la placa (suponemos el exterior de la L que forma) al taladro de 5 mm y se atornilla en los 4 taladros de 1.1 mm.
2. Se pasan los cables del láser por cada agujero desde el interior hasta el exterior.
3. El láser se acopla por la parte interior de la placa en la superficie cuadrada y se atornilla en los 4 taladros de 2 mm.

En la figura 34 se puede observar el esquema de la placa:

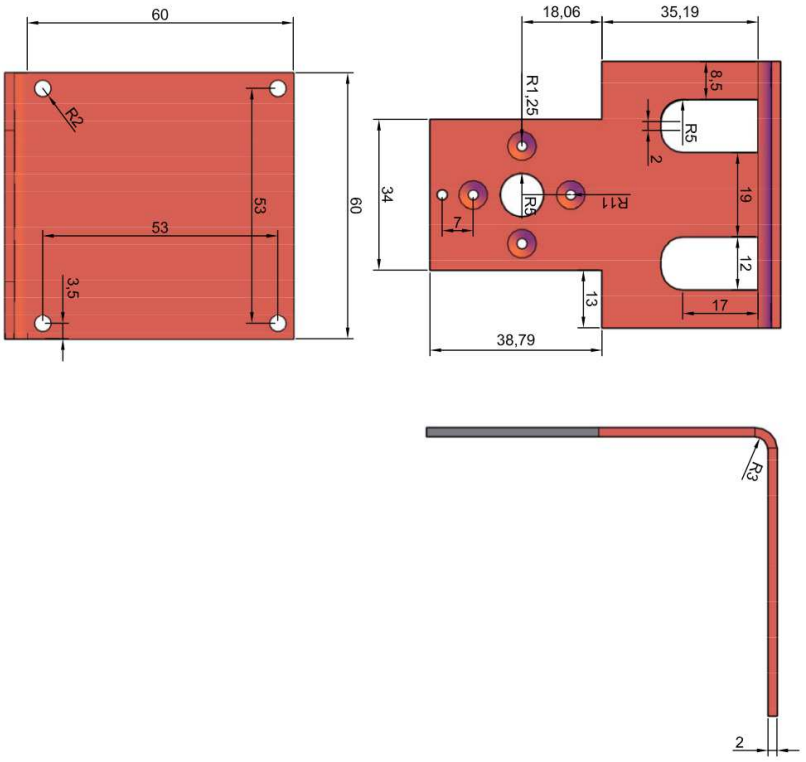


Figura 34: Esquema placa acoplamiento del láser al motor.

3.8. Placa soporte conjunto

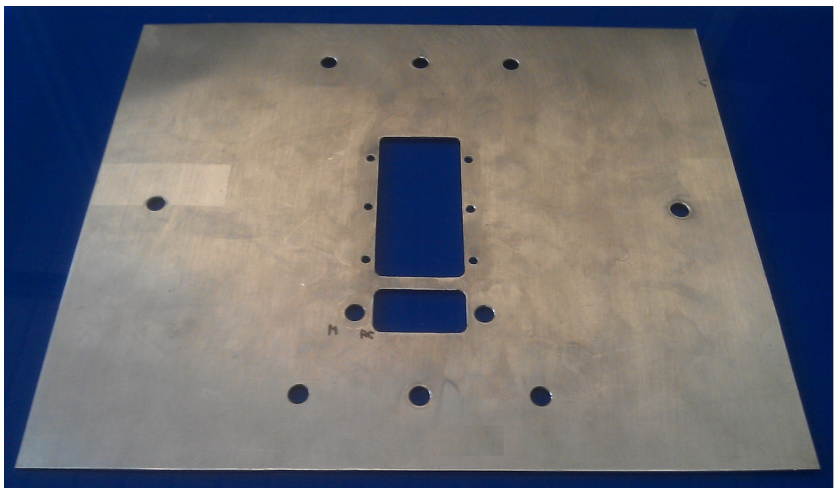


Figura 35: Placa soporte motor y laser.

Esta placa se diseña y fabrica para la sujeción del conjunto formado por el láser Hokuyo, el motor Dynamixel y la placa electrónica de alimentación y acoplarlo a la parte superior de MANFRED. Está fabricada en chapa de aluminio de 2 mm de espesor y tiene las dimensiones de la parte superior del robot.

Como se puede observar en la figuara 36, la placa esta compuesta de 8 taladros de 6 mm situados a 22 mm del exterior para la sujeción a la parte superiror del robot. Tiene un orificio central con la forma del motor para su acoplo junto a 6 taladros de 2,7 mm para atornillarlo. Y cuenta con un orificio para el paso de los cables del motor y el láser junto a dos taladros de 6 mm para la sujeción de la tarjeta de alimentación.

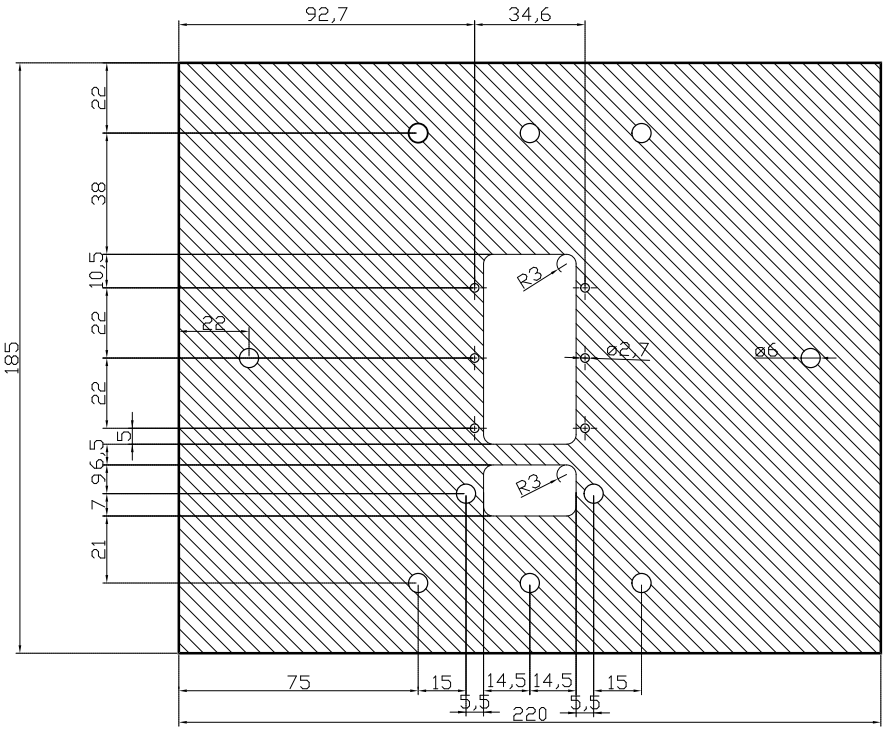


Figura 36: Plantilla placa soporte láser.

Montaje Con el conjunto formado por el motor, el láser y el acoplamiento entre los dos ya montado, y por otro lado la tarjeta de alimentación, se procederá a montarlo a la placa de soporte.

1. Se pasan los cables del motor y el láser por el agujero rectangular inferior.
2. Se enchufan los conectores del láser y el motor a la tajeta electrónica.
3. Se fija la tarjeta electrónica a la placa medieante los dos taladros de los lados del pasacables. Se utiliza para ello la base del interruptor y una tuerca a un lado y un tornillo más 3 tuercas para el otro lado.
4. Se fija el motor en el agujero central y se atornilla a la placa mediante los 6 taladros.
5. Se ajusta el largo de los cables del láser con cuidado para permitir todo el rango de movimiento.

4. Software desarrollado

El software se encarga de controlar el hardware (Dynamixel y Hokuyo), establecer las comunicaciones entre los diferentes bloques y finalmente controlar de forma secuencial el proceso de obtención de una nube de puntos y enviarla al nodo Cliente.

Este software está programado en C++ y está basado en la utilización de ROS, el cual se explicó la sección 2.3. Para esta aplicación se usan 3 nodos. Nodo Dynamixel y nodo Hokuyo, que se encargan del control del motor y el láser y el nodo láser3D que se encarga de dirigir y relacionar los datos de los otros dos nodos. Estos nodos se explicarán de forma más exhaustiva en los siguientes apartados.

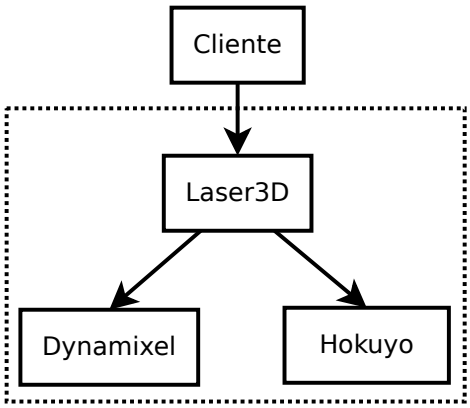


Figura 37: Esquema software.

4.1. Útiles necesarios y configuración

Para la utilización de este software en un PC es necesario lo siguiente:

- PC con Ubuntu como sistema operativo. Se puede descargar Ubuntu desde: http://www.ubuntu.com/download
- Instalación de ROS versión Groovy. Con versiones posteriores es posible que se tenga que reconstruir el package eliminando los archivos temporales de ROS y/o realizar modificaciones causadas por cambios en las librerías que proporciona ROS. Se puede obtener la última versión de ROS y información de instalación en:

  http://www.ros.org/wiki/groovy/Installation/Ubuntu

También es necesario instalar las siguientes librerías que son utilizadas por el compilador para el código fuente:

- PCL (Point cloud library): Es la librería para el uso de nubes de puntos. Se puede descargar aquí:

  http://pointclouds.org/downloads/

- SDL (Simple DirectMedia Layer): Es una librería que proporciona acceso a bajo nivel de hardware, en este caso al láser Hokuyo. Se puede descargar aquí:

  http://www.libsdl.org/download-1.2.php.

- BOOST: Son una serie de librerías generales que se utilizan en el software que controla al láser Hokuyo. Se puede descargar en:

  http://www.boost.org/users/download/

- URG: Es la librería de desarrollo software del láser Hokuyo. Se puede obtener en:

  http://www.hokuyo-aut.jp/02sensor/07scanner/download/urg_programs_en/

- DXL: Es una librería para el motor. Archivo libdxl.a.

Para la comunicación por los puertos usb del láser y del motor sera necesario dar permisos de acceso a los puertos en los que estén conectados:

- Para Hokuyo: Se debera dar permisos al puerto ACMx (donde x puede cambiar cada vez que se conecta). La manera de hacerlo es poner en el terminal el siguiente comando:

  "sudo chmod 777 /dev/ttyACM0"

- Para Dynamixel: Se debera dar permmisos al puerto USBx (donde x puede cambiar cada vez que se conecta). Se tiene que poner en el terminal el siguiente comando:

  "sudo chmod 777 /dev/ttyUSB0"

Para no tener que cambiar el numero de puerto si estos cambian se utilizan reglas udev de linux. Esto se consigue incluyendo un arhivo laser.rules en la carpeta /etc/udev/rules.d.

El archivo laser.rules contiene las dos lineas siguientes:

- KERNEL=="ttyACM*", SUBSYSTEMS=="usb", ATTRS{manufacturer}=="Hokuyo Data Flex for USB", SYMLINK+="laser"
- KERNEL=="ttyUSB*", SUBSYSTEMS=="usb", ATTRS{manufacturer}=="Linux 3.2.0-44-generic-pae uhci_hcd", SYMLINK+="motor"

Después de esto se deberá actualizar ejecutando en una consola la siguiente instrucción:

"sudo service udev reload".

4.2. Estructura de archivos

ROS utiliza un sistema de carpetas para crear los packages. A continuación se explica la utilidad de cada carpeta y los archivos necesarios.

- Carpeta bin: En esta carpeta se crearán los archivos ejecutables de los nodos de ROS tras la compilación, y también contiene los archivos que los ejecutables utilizaran. Los archivos de esta carpeta son los siguientes:
  - bin_cliente: Archivo ejecutable del nodo Cliente generado después de la compilación.
  - bin_nodo_dynamixel: Archivo ejecutable del nodo Dynamixel generado después de la compilación.
  - bin_nodo_hokuyo: Archivo ejecutable del nodo Hokuyo generado después de la compilación.
  - bin_nodo_laser3D: Archivo ejecutable del nodo Laser3D generado después de la compilación.
  - parameters.xml: Archivo xml para lectura del nodo Cliente que contiene las posiciones inicial y final, y las velocidades de medida y posicionamiento del motor. Usando este archivo se consigue no tener que compilar si queremos cambiar uno de estos parámetros.
- Carpeta launch: Esta carpeta contiene los archivos launch de ROS que sirven para lanzar un nodo desde el terminal.
  - laser3D.launch: Lanza los nodos Dynamixel, Hokuyo y Laser3D a la vez desde el terminal con el comando: "roslaunch laser3D laser3D.launch". Además contiene los parámetros de conexión de los puertos USB Y ACM del láser y el motor. [2]
- Carpeta msg: Contiene los archivos de texto que describen los mensajes de ROS y son usados para generar el código fuente.
  - srv_laser_scan.msg: Es la descripción del mensaje que utiliza el topic del nodo Hokuyo para publicar los datos obtenidos del sensor láser.

[2]Esta configuración impide lanzar los nodos individualmente sin usar roslaunch porque el archivo launch contiene los puertos de conexion por defecto, puediendose pasar por parametros de referencia también. Si se desea lanzar individualmente los nodos se debe fijar el puerto de conexión en el código de los nodos Dynamixel y Hokuyo.

- Carpeta src: Esta carpeta contiene el código fuente, archivos de cabecera y libreráas usadas por el compilador.
  - cliente.cpp: Código fuente del nodo Cliente
  - dynamixel.h: Archivo de cabecera que contiene parámetros del motor Dynamixel.
  - ex106p.h: Archivo de cabecera que contiene parámetros del motor Dynamixel.
  - libdxl.a: Libreria dxl usada en el nodo Dynamixel.
  - motordynamixel.cpp: Código fuente de las clases implementadas para el motor Dynamixel.
  - motordynamixel.h: Archivo de cabecera de las clases del motor Dynamixel.
  - nodo_dynamixel.cpp: Código fuente del nodo Dynamixel.
  - nodo_hokuyo.cpp: Código fuente del nodo Hokuyo.
  - nodo_laser3D.cpp: Código fuente del nodo Laser3D
- Carpeta srv: Contiene los archivos de texto que describen los servicios de ROS y son usados para generar el código fuente.
  - srv_dynamixel.srv: Descripción de los datos que utilizan los servicios del nodo Dynamixel.
  - srv_hokuyo.srv: Descripción de los datos que utilizan los servicios del nodo Hokuyo.
  - srv_laser.srv: Descripción de los datos que utilizan los servicios del nodo laser3D.
- CMakeLists.txt: Es el archivo de configuración para compilar el código.
- Findlibdxl.cmake: Sirve de apoyo al archivo cmake para encontrar la libreria dxl.
- mainpage.dox: Contiene información del package
- Makefile: Archivo que contiene información para llamar a CMake.
- manifest.xml: Contiene información del package y de las dependencias (packages que utiliza) de este paquete.

En la figura 38 se puede observar el contenido del directorio del package.

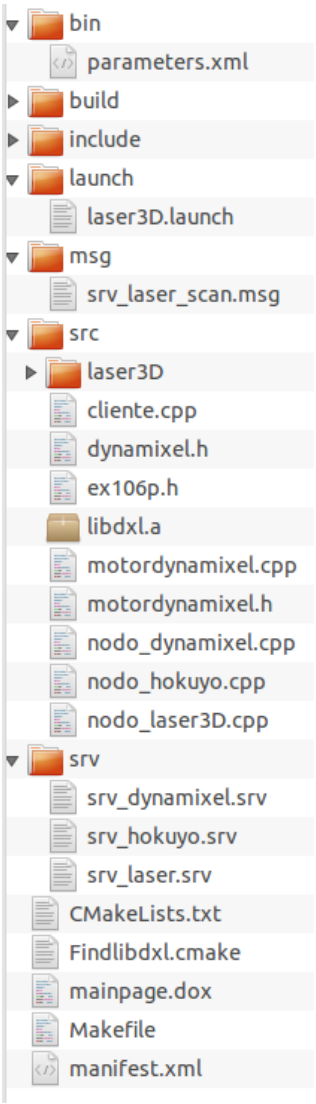


Figura 38: Estructura ROS.

Las carpetas build e include y sus contenidos no son necesarias. ROS las creará al compilar.

4.3. Estructura de comunicaciones ROS

En esta sección se explican todas los comunicaciones que existen entre los nodos de ROS, el tipo y los datos que se transmiten. Las comunicaciones en este proyecto se consideran de dos tipos: topics y servicios. Están programadas en los archivos de las carpetas "srv" y "msg", como se puede ver en la sección anterior y contienen los tipos de datos de los mensajes que utilizan los servicios y los topics. En la figura 39 se puede observar un diagrama general de las comunicaciones de este proyecto.

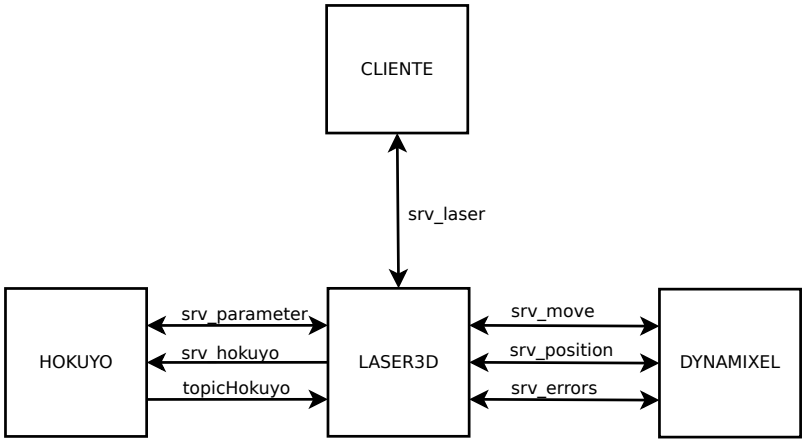


Figura 39: Esquema comunicaciones software.

A continuación se describen cada una de las comunicaciones del proyecto, los datos que utilizan cada una y entre paréntesis el tipo de dato:

- **topicHokuyo (sensor_msgs/LaserScan):** Es el topic que envía el nodo Hokuyo con los datos de las capturas del láser.

- **srv_parameter:** Es el servicio que tiene el nodo Hokuyo para configurar los parámetros del láser. Y contiene los siguientes datos:
  - **Angulo mínimo (int):** Es el ángulo inicial desde el que mide el láser.
  - **Angulo máximo (int):** Es el ángulo final hasta donde mide el láser.
- **srv_hokuyo:** Es el servicio que tiene el nodo Hokuyo para seleccionar si pide datos al láser y envía los topics o no.
  - **Opción (int):** Con esta variable se selecciona el envío o no, dependiendo si es 0 o 1.
- **srv_move:** Es el servicio del nodo Dynamixel que ordena mover el motor.
  - **Posición (float):** Esta variable se usa para dar la posición destino del movimento.
  - **Velocidad (int):** Se usa para dar la velocidad con la que el motor debe moverse.
- **srv_position:** Es el servicio del nodo Dynamixel que pide la posición al motor.
  - **Posición (float):** Esta variable es la respuesta del servicio con la posición del motor.
- **srv_errors:** Este servicio del nodo Dynamixel se usa para comprobar los posibles errores en el motor y enviarlos.
  - **Errores de funcionamiento:** Esta variable guarda los errores de funcionamiento del motor.
  - **Errores de comunicaciones:** Guarda los errores de comunicaciones.
- **srv_laser:** Este servicio del nodo Laser3D es el principal del funcionamiento del proyecto. Es llamado por un cliente que quiere una nube de puntos y gestiona la secuencia de configuración, inicialización, proceso de obtención de puntos, transformación y envío de la nube.
  - **Velocidad de medida (int):** Es la velocidad con la que se desplazará el motor mientras el láser toma puntos y la envía el nodo Cliente al nodo Laser3D.

- **Velocidad de desplazamiento (int):** Es la velocidad con la que se desplazará el motor mientras va a la posición inicial y la envía el nodo Cliente al nodo Laser3D.
- **Posición inicial (float):** Es la posición inicial de barrido del láser y la envía el nodo Cliente al nodo Laser3D.
- **Posición final (float):** Es la posición final de barrido del láser y la envía el nodo Cliente al nodo Laser3D.
- **Angulo mínimo (int):** Es el ángulo mínimo desde el cual el láser toma puntos y es enviado desde el nodo Cliente al nodo Laser3D.
- **Angulo máximo (int):** Es el ángulo máximo hasta el cual el láser toma puntos y es enviado desde el nodo Cliente al nodo Laser3D.
- **Rango mínimo (int):** Es la distancia mínima desde la que mide el láser los puntos y es enviado desde el nodo Cliente al nodo Laser3D.
- **Rango máximo (int):** Es la distancia máxima hasta la que mide el láser los puntos y es enviado desde el nodo Cliente al nodo Laser3D.
- **Nube de puntos (sensor_msgs/PointCloud2):** Es la nube de puntos 3D recopilada y es enviada desde el nodo Laser 3D al nodo Cliente.

4.4. Nodo Dynamixel

Este nodo se encarga de controlar el motor Dynamixel. Para comunicarse con otros nodos ROS este nodo se comporta como servidor usando el sistema cliente-servidor.

Para controlar el motor y comunicarse realiza tres tareas que corresponden a tres servicios y se puede ver representado en la figura 40:

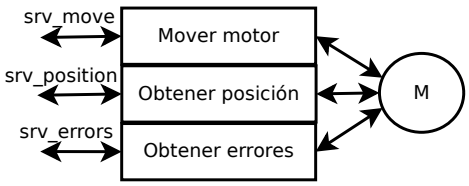


Figura 40: Esquema nodo Dynamixel.

- **Mover el motor a una posición y velocidad dadas:** El nodo recibe una posición entre 0º y 250º y una velocidad de 1 (mínimo) hasta 1023 (máximo) y ordena al motor el movimiento. El servicio finaliza tras mandar la orden al motor y este seguirá el moviemiento sin el servicio activo.
- **Pedir al motor la posición en la que se encuentra:** El nodo recibe una petición de envio de posición, este ordena al motor el envío de su posición y el nodo la reenvía.
- **Pedir al motor que le responda con los errores que tiene:** El nodo recibe una petición de errores en el motor, este ordena al motor que le envíe los errores y el nodo los reenvía.

A continuación se explica el funcionamiento del nodo y en la figura 41 se presenta su flujograma:

1. Se inicia el nodo y sus comunicaciones.
2. Permanece en estado de espera hasta que llega una petición de alguno de los servicios.
3. Realiza uno de los servicos (mover motor, pedir posición o pedir errores).
4. Vuelve al estado de espera.

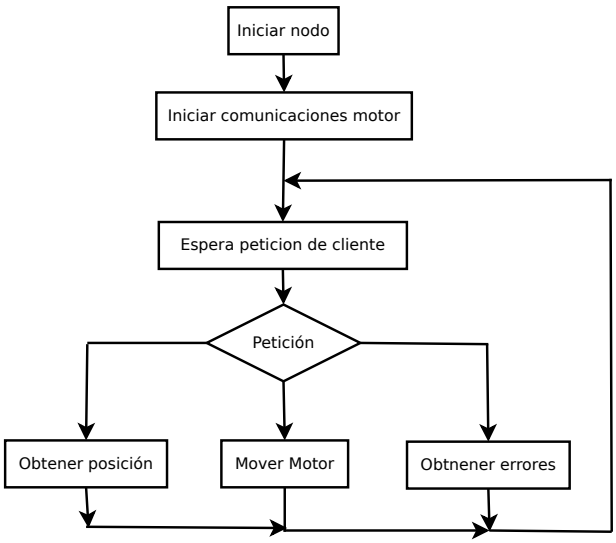


Figura 41: Flujograma nodo Dynamixel.

4.5. Nodo Hokuyo

Este nodo se encarga de controlar el láser Hokuyo y enviar los datos medidos. Para comunicarse se comporta por un lado como topic, para enviar los datos obtenidos del láser, y por otro lado como servidor para controlar el funcionamiento del láser. En la figura 42 se pude ver un esquema del funcionamiento del nodo y a continuación se describen sus partes.

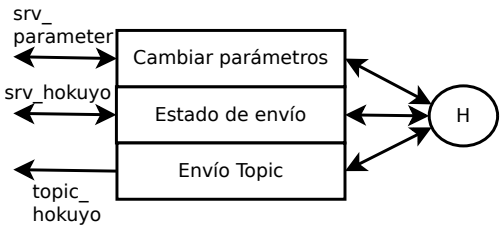


Figura 42: Esquema nodo Hokuyo.

- Cambiar parámetros del motor: A través de este servicio se puden configurar los parámetros de captura del láser como el ángulo mínimo y máximo de captura del láser.
- Estado de envio: Con este servicio se controla si se capturan datos del láser y si se envian los topics.
- Envio de topic: Esta parte del nodo se encarga de publicar el topic con los datos obtenidos del láser.

A continuación se explica el funcionamiento y se presenta el flujograma del nodo en la figura 43:

1. Se inicia el nodo y sus comunicaciones.
2. Se dan dos situaciones dependiendo si el envío de datos esta activo o no:
   - Envío de datos inactivo: Sigue el proceso sin pedir al láser que capture datos ni publique topics.
   - Envío de datos activo:
     a) Se pide al láser que haga un barrido para obtener datos.
     b) Se publica el topic con los datos.
3. Se atienden las peticiones de algún posible cliente a los servicios de cambio de parámetros o cambio de estado.
4. Después de atender las peticiones a los servicios, o si no hubiera peticiones, se vuelve al punto 2.

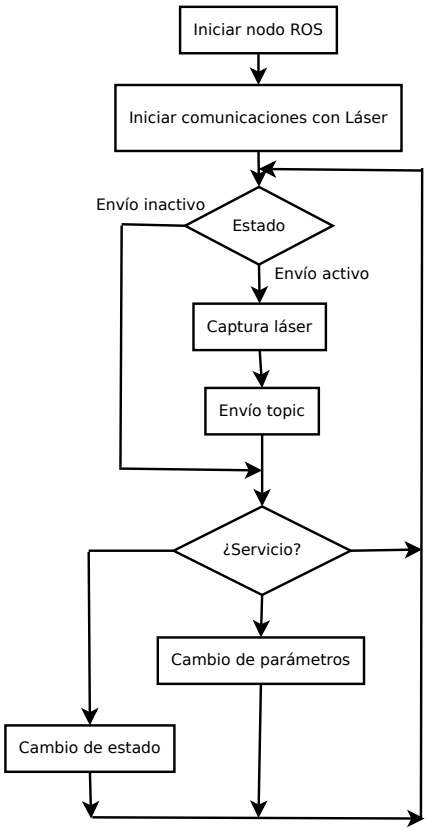


Figura 43: Flujograma nodo Hokuyo.

4.6. Nodo Laser3D

Este nodo se encarga de llevar a cabo la secuencia de la captura de una nube de puntos. Para ello el nodo se comunica con los nodos Hokuyo y Dynamixel para dirigirlos, y recibir y relacionar sus datos.

Las funciones de este nodo son:

- Comunicarse con los nodos Dynamixel y Hokuyo para ordenar sus acciones.
- Obtener los datos de ambos nodos y guardarlos en vectores.
- Transformar los datos obtenidos en una nube de puntos XYZ.
- Enviar la nube de puntos al nodo Cliente.

En la figura 44 se representan las comunicaciones del nodo. El nodo se comporta como cliente de los nodos Dynamixel y Hokuyo, como subscriptor del topic del nodo Hokuyo, que envía los puntos obetenidos por el láser, y como servidor del nodo que esté pidiendo la nube de puntos.

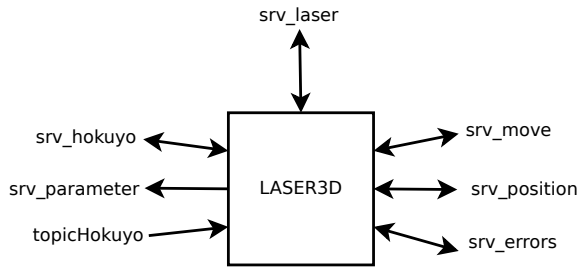


Figura 44: Diagrama comunicaciones nodo Laser3D.

A continuación se explica la secuencia de acciones de este nodo y se puede observar en la figura 45:

1. El nodo inicia las comunicaciones del servidor láser (solo comunicaciones entre esté y su cliente) y después permanece en espera hasta que un nodo cliente realiza una petición.

2. Se rellena dos vectores con los senos y cosenos de todos los ángulos que van a ser utilizados numerosas veces en las transformaciones con posterioridad.

3. Cuando se recibe la petición del envío de una nube de puntos por parte de un cliente, el nodo entra en el servicio y se inician las comunicaciones con los nodos Dynamixel y Hokuyo.

4. Se mueve el motor hasta la posición inicial, la cual se elige de entre las dos posiciones extremo del recorrido, dependiendo de donde se encuentre más cerca el motor.

5. Cuando el motor llega a la posición inicial, se ordena al nodo Hokuyo que envíe datos del láser y se ordena al motor que se mueva hacia la dirección final. En este punto el nodo entra en un bucle del que saldrá cuando el motor llegue a la posición final. Mientras se permanece en el bucle, cada vez que llegan datos del láser, se pide la posición del motor al nodo Dynamixel y se guardan en las posiciones de memoria de 2 vectores. Un vector que guarda los datos del láser y un vector que guarda las posiciones del motor.

6. Cuando el motor llega a su punto final el programa sale del bucle anterior y se para el envío de datos del láser del nodo Hokuyo.

7. Se transforman los datos obtenidos generando una nube de puntos XYZ que se explica en la sección 3.3.

8. Por último se envía la nube de puntos al cliente y se libera el espacio de memoria utilizado.

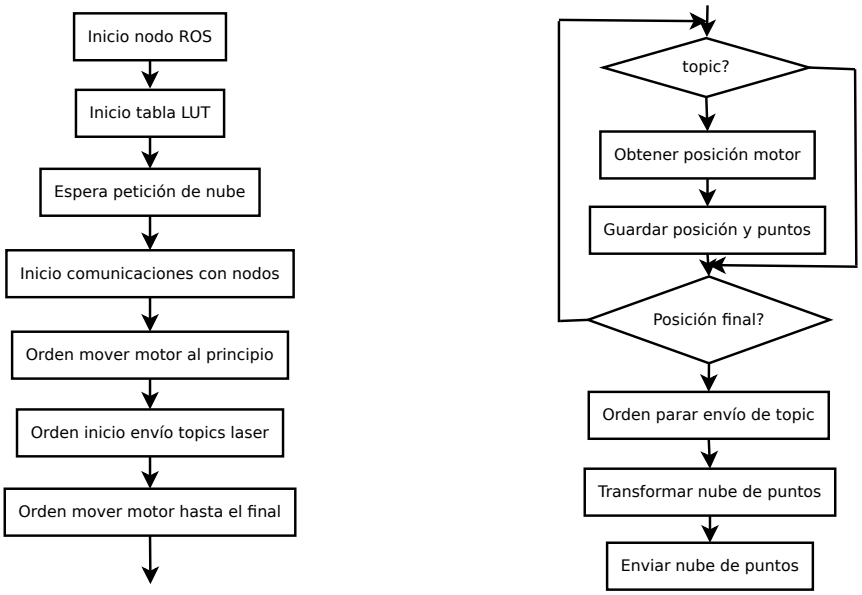


Figura 45: Flujograma nodo Laser3D.

4.7. Nodo Cliente

Este nodo es un ejemplo del nodo que pediría una nube de puntos al nodo Laser3D y se ha usado para simular las comunicaciones entre ambos nodos.

El funcionamiento es sencillo. Tras iniciarse el nodo y sus comunicaciones con ROS, lee los parámetros del motor y del láser que estan guardados en un archivo XML. Después el nodo hace una petición de nube de puntos. Cuando la nube de puntos es recibida se guarda en un archivo llamado nube.pcd. Y por último, se lanza el visualizador pcd viewer para ver la nube.

A continuación, en la figura 46, se puede observar el flujograma de este nodo.

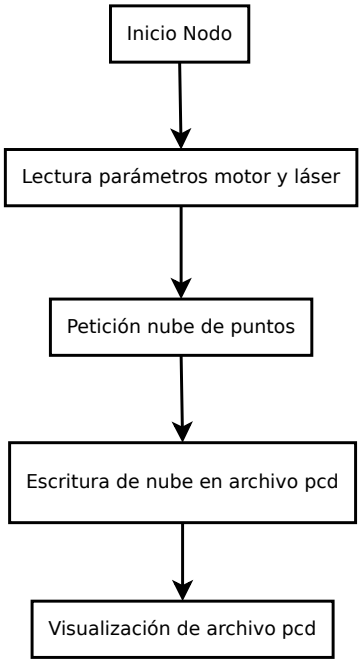


Figura 46: Flujograma nodo Cliente.

En el apéndice C se puede ver un ejemplo del código de un nodo cliente explicado.

4.8. Parámetros modificables

Para la obtención de una nube de puntos es posible modificar una serie de parámetros en el motor y en el láser por parte del nodo Cliente:

- **Posiciones inicial y final de moviemiento del motor:** Se puede modificar estos parámetros para que el motor mida todo el entorno o solo una parte que nos interese. Aunque el motor puede moverse desde 0º a 250º, para capturar una nube de puntos de todo el entorno basta con mover el motor 180º, pues el láser mide a ambos lados del motor.
- **Velocidad de posicionamiento del motor:** Este parámetro se puede modificar para que el motor se mueva más o menos rápido en el proceso de posicionamiento a la posición inicial. Aunque el motor puede moverse a velocidades relativas de 0 a 1023, por motivos de seguridad y demasiada lentitud, se puede establecer entre 10 y 80. Su velocidad recomendable es de 50. Consultar tabla 4 en la sección 3.5 para ver velocidades reales.
- **Velocidad de medida del motor:** Es posible modificar la velocidad con la que el motor se mueve en el proceso de captura con el fin de obtener una mayor o menor densidad de puntos. Aunque el motor puede moverse a velocidades relativas de 0 a 1023, por motivos de seguridad y demasiada lentitud, se puede establecer entre 10 y 80. Consultar tabla 4 en la sección 3.5 para ver velocidades reales.
- **Angulo máximo y mínimo del láser:** Estos parámetros son desde qué ángulo a qué ángulo toma puntos el láser. El láser puede medir 270º, desde -45º a 225º. Siempre se pondrá el ángulo mínimo el más pequeño y el máximo el mayor. Por defecto el láser tiene como ángulo mínimo -45º y como ángulo máximo 225º.
- **Rango de medida mínimo y máximo del láser:** Estos parámetros son la distancia mínima y máxima de medida del láser. Estos rangos pueden moverse desde 0 hasta 60000 mm. Por defecto el láser tiene como rango mínimo 23 mm y rango máximo 60000 mm.

5. Resultados

En esta sección se muestran los resultados obtenidos de forma visual. En primer lugar, hay que tener en cuenta que el objetivo del proyecto no es la visualización de la nubes de puntos, sino que es la creación de nubes de puntos para el uso del robot. Por esto la visualización no será demasiado buena, sobre todo desde algunas perspectivas en las que puntos de objetos tapan otros.

En los siguientes apartados se van a mostrar algunos resultados obtenidos en distintos lugares como los laboratorios de robótica y un pasillo, y también se presentará la captura de puntos aislada mostrando los puntos de una silla.

Las imágenes de las nubes de puntos han sido visualizadas con la herramienta "PCD VIEWER", perteneciente a las librerías PCL. Con esta herramienta se puden visualizar las nubes de puntos en 3 dimensiones, alejando, acercando y rotando la imagen en todos los ejes.

Estos son los comandos que se deben utilizar en un terminal para visualizar una nube de puntos:

- Con PCL: "pcd_viewer nube.pcd".
- Con PCL_ROS: "/usr/bin/pcd_viewer nube.pcd".

5.1. Prueba de simulación

En una primera fase de pruebas en la captura de una nube de puntos se optó por simular los datos del láser y del motor. Con esto se conseguiría determinar la coordinación del software y las comunicaciones de los diferentes nodos de ROS.

En la simulación las medidas del láser serían siempre las mismas para todos los ángulos (de 0º a 180º) y las medidas del motor se irían sumando en 1º cada ciclo de ejecución. Con esto, la nube de puntos resultante sería una semiesfera, que podemos ver en la figura 47. Y como se puede observar se consigue una semiesfera perfecta, demostrando la correcta sincronización de los datos de los diferentes nodos.

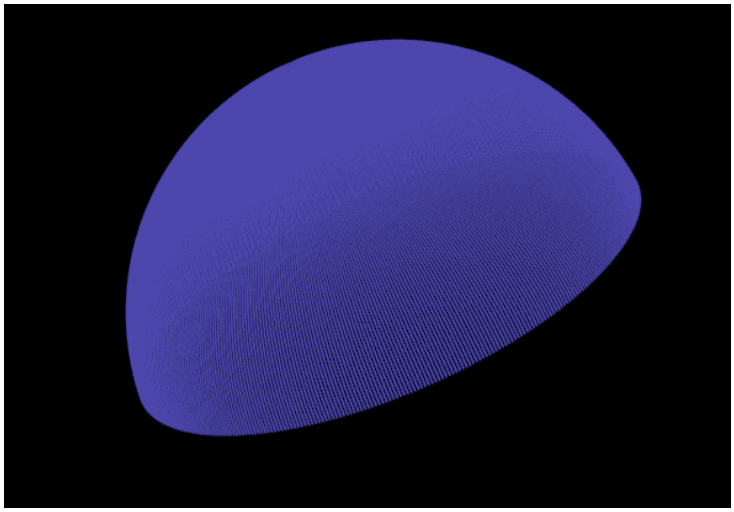


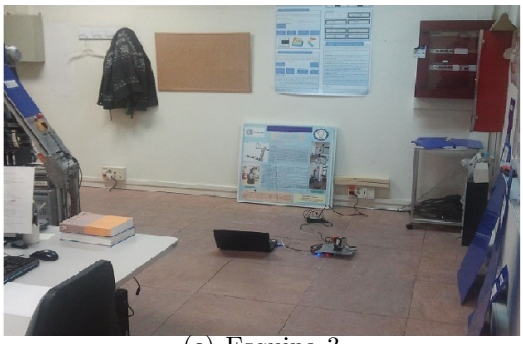
Figura 47: Nube de puntos semiesfera de prueba (Nº puntos: 129.600).

5.2. Laboratorio

En esta sección se presentan nubes de puntos obtenidas en un laboratorio con el fin de mostrar el funcionamiento de este proyecto en habitaciones de un tamaño medio y con diferentes objetos. En la figura 48 se observa el laboratorio de robótica de la UC3M donde se han tomado las capturas de nubes de puntos que a continuación se presentan.

(a) Esquina 1

(b) Esquina 2



(c) Esquina 3

(d) Esquina 4

Figura 48: Laboratorio 1: Imágenes reales.

En la figura 49 se representa una nube de puntos vista desde el exterior. La nube se ha capturado con el láser en la posición en la que se muestra en la figura 48 abajo izquierda y con todo el rango del láser y el motor.

Se puede ver el contorno de todo el laboratorio, mostrando con claridad las aristas y esquinas. También se pueden observar las luminarias del techo, la puerta de entrada en la esquina inferior derecha. Se observa también una serie de puntos, a la derecha de la imagen, que pertenecen al laboratorio de al lado que ha capturado el láser al estar la puerta abierta. Del interior del laboratorio no se puede apreciar correctamente las siluetas formadas por los puntos por estar delante los puntos pertenecientes a las paredes y el techo.

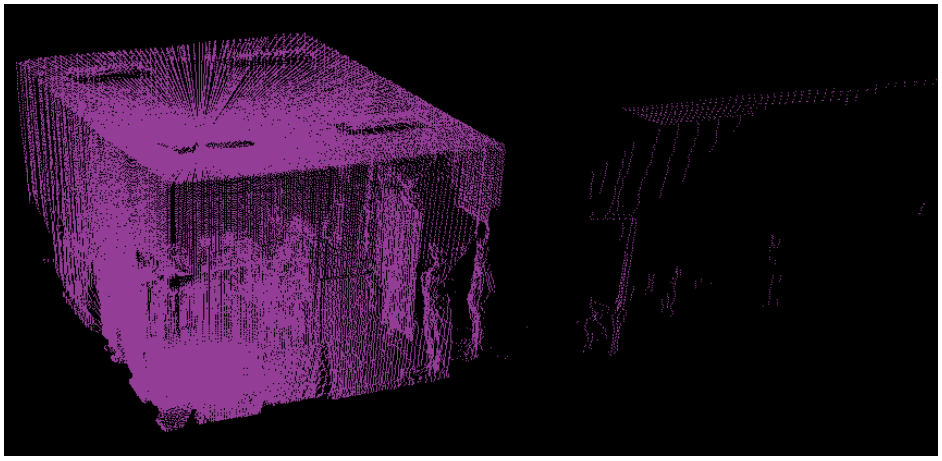


Figura 49: Laboratorio 1: Imagen nube de puntos 1 (Nº puntos: 328320).

En la figura 50, se puede ver una vista desde arriba de una nube de puntos del laboratorio. Se puede observar que se ha conseguido una muy buena precisión en las medidas del láser y el motor, pues las paredes que se ven a la izquierda y abajo son completamente rectas. No es así en la pared que se ve arriba porque no es recta en realidad. También se aprecia en la parte derecha una aparente anomalía, que es la parte superior del muro de las ventanas. En la esquina superior izquierda si se presenta una anomalía. Son puntos erroneos tomados por el láser. Esto se produce porque en esa esquina hay dos vitrinas de metal en las que la luz láser es reflejada.

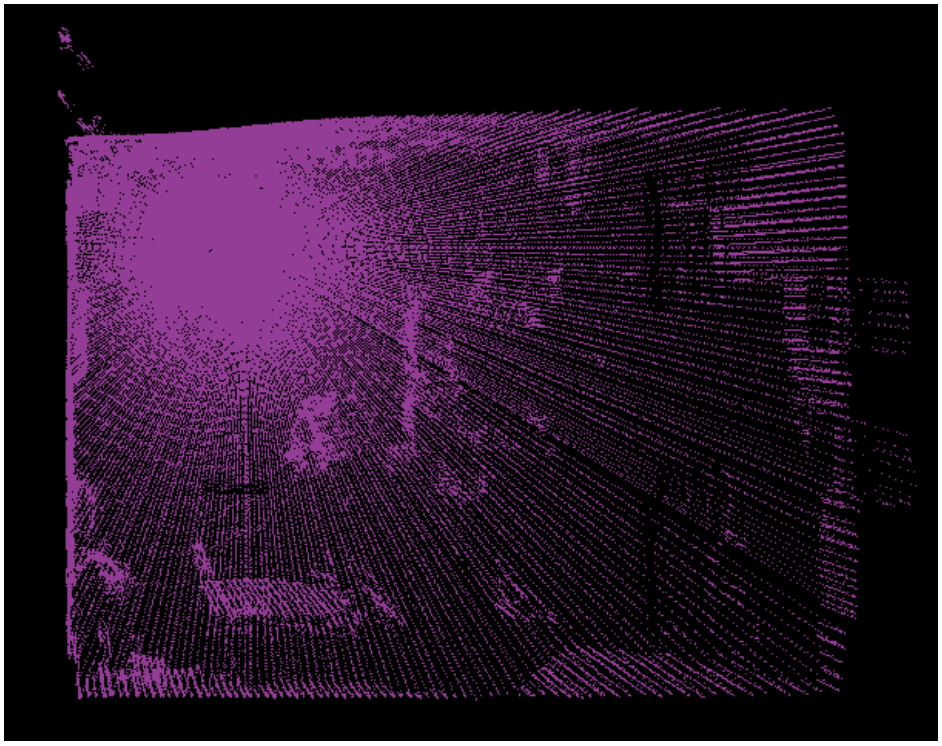


Figura 50: Laboratorio 1: Imagen nube de puntos 2 (Nº puntos: 328320).

En la figura 51 se muestra otra perspectiva del laboratorio en una nube de puntos tomada con menor velocidad (20, siendo lo normal 50) para obtener más puntos en la que se ve la esquina del laboratorio que se ve en la figura 48. Esta imagén tiene un alto grado de detalle gracias a las sombras formadas de los objetos. Se pude apreciar que el láser es capaz de mostrar hasta el contorno de un corcho colgado en la pared, una pizarra y otros objetos delgados. Tambien se aprecia el contorno de los abrigos que hay en la percha, las vitrinas de la esquina y los puntos superpuestos del robot MANFRED.

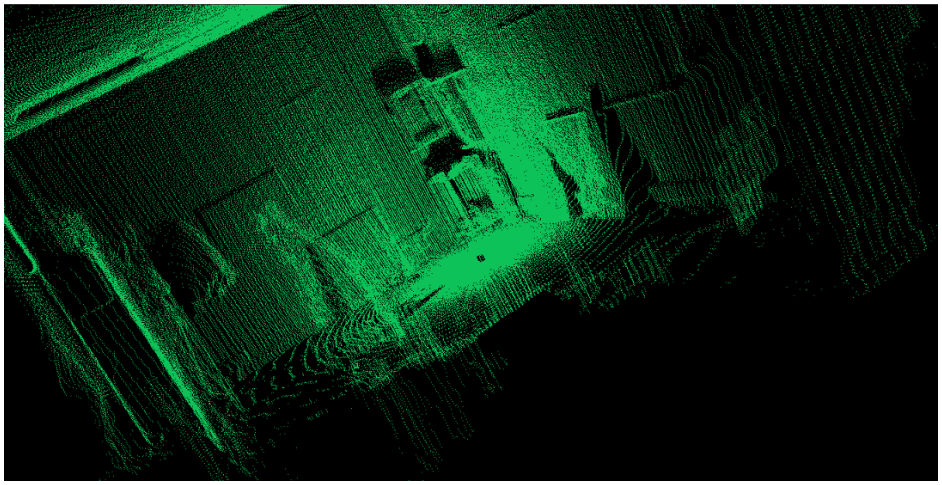


Figura 51: Laboratorio 1: Nube de puntos 3 (girada para mejor visualización)(Nº puntos: 668520).

En la figura 52 se presenta una nube de puntos tomada desde el emplazamiento final del láser, la parte superior de MANFRED. Se puden apreciar casi en el centro de la imagen puntos pertenecientes al soporte del láser y al robot, que podrán ser filtrados en el futuro si se desea. En la imagen se pueden ver con claridad las ventanas, la estructura del laboratorio, los armarios, la pizarra y el espacio de mesas sillas, ordenadores, etc.

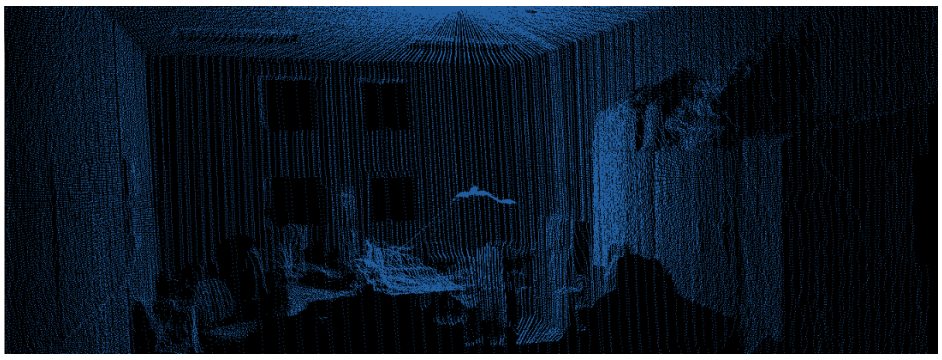


Figura 52: Laboratorio 1: Nube de puntos 4 (Nº puntos: 338040).

En la figura 53 se muestra como se pueden modificar los parámetros de captura de una nube de puntos para obtener puntos de una parte del entorno nada más, y también se modifica la velocidad a 10. En la imagen se ve una

de las esquinas del laboratorio , con las vitrinas de los automáticos eléctricos y del extintor, y la pizarra a la derecha.

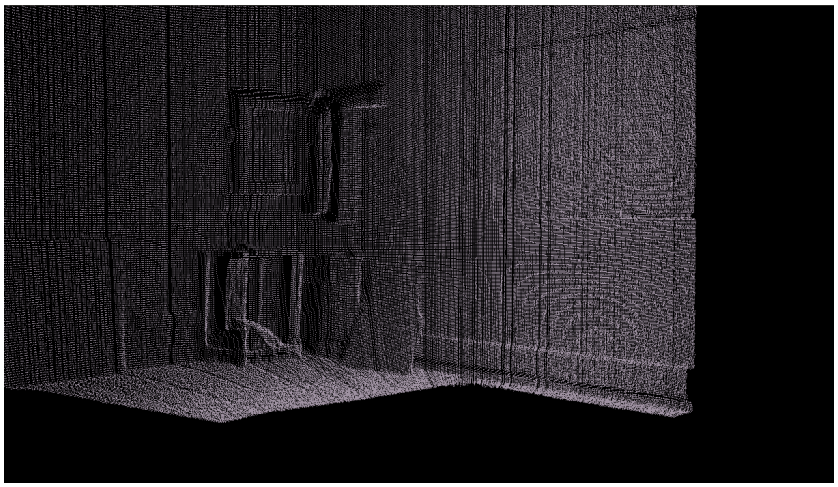


Figura 53: Laboratorio 1: Nube de puntos 5 (N° puntos: 429640).

También se han tomado capturas en otro laboratorio del cual se pueden ver las imagenes reales en la figura 54. En este laboratorio sera más complicada la visualización de los objetos al haber muchos más que en el anterior.

Figura 54: Laboratorio de robótica 2.

Las figuras 55 y 56 se pueden observar dos perspectivas diferentes de una nube de puntos tomada en la posición que se puede ver en la figura 54 abajo. Se puede ver el contorno del laboratorio en la primera capa de puntos y con algo menos de detalle, al haber muchos objetos de tamaños pequeños, podemos apreciar los objetos del interior del laboratorio como mesas, ordenadores, monitores, etc. También podemos ver la estructura metálica y su sombra. Y puntos fuera del laboratorio que pertenecen al laboratorio de al lado y han sido capturados al estar la puerta abierta.

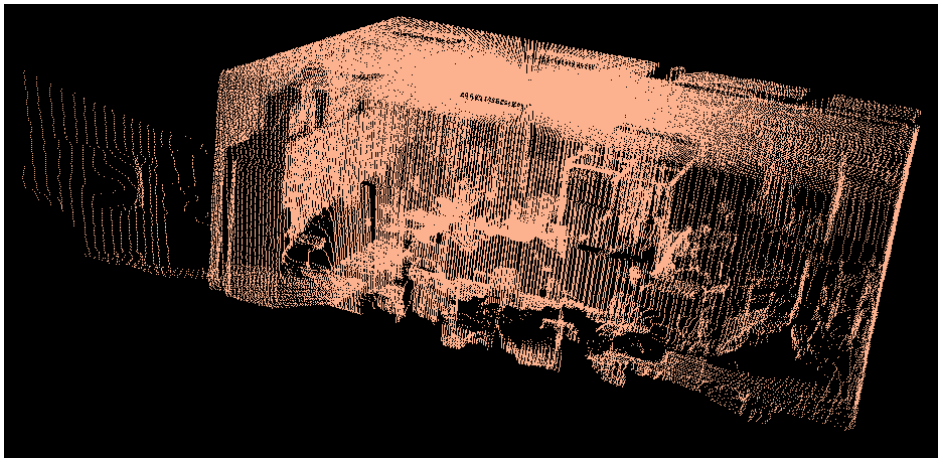


Figura 55: Laboratorio 2: Nube de puntos 1 (Nº puntos: 330480).

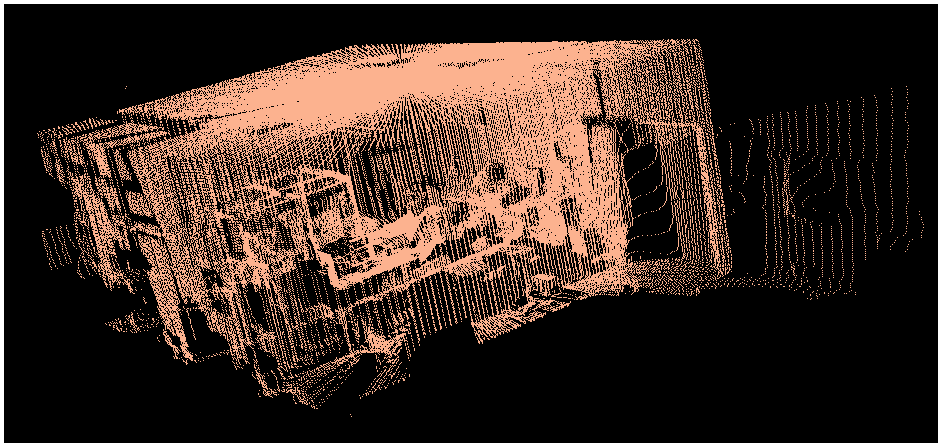


Figura 56: Laboratorio 2: Nube de puntos 2 (Nº puntos: 330480).

5.3. Pasillo

En esta sección se presentan diferentes perspectivas de una nube de puntos obtenida en un pasillo de la universidad con el fin de mostrar el funcionamiento del proyecto en lugares más grandes, mostrando la captura de puntos a larga distancia. En la figura 57 se puede ver una imagen real de uno de los lados del pasillo y el emplazamiento del láser, y en la figura 58 se puede ver el otro lado del pasillo.

Figura 57: Imagen real pasillo lado 1.

Figura 58: Imagen real pasillo lado 2.

En la figura 59 se observa la nube de puntos desde fuera en la que se puede ver el contorno del pasillo. Podemos ver que hay una gran densidad de puntos cerca del punto de medida y que los puntos de captura llegan hasta una distancia de entre 10 y 20 metros. Hay que tener en cuenta en este entorno que el suelo del pasillo es de un material que produce un efecto espejo, por lo que puede generar puntos erroneos. También podemos ver los marcos de las puertas de los despachos.

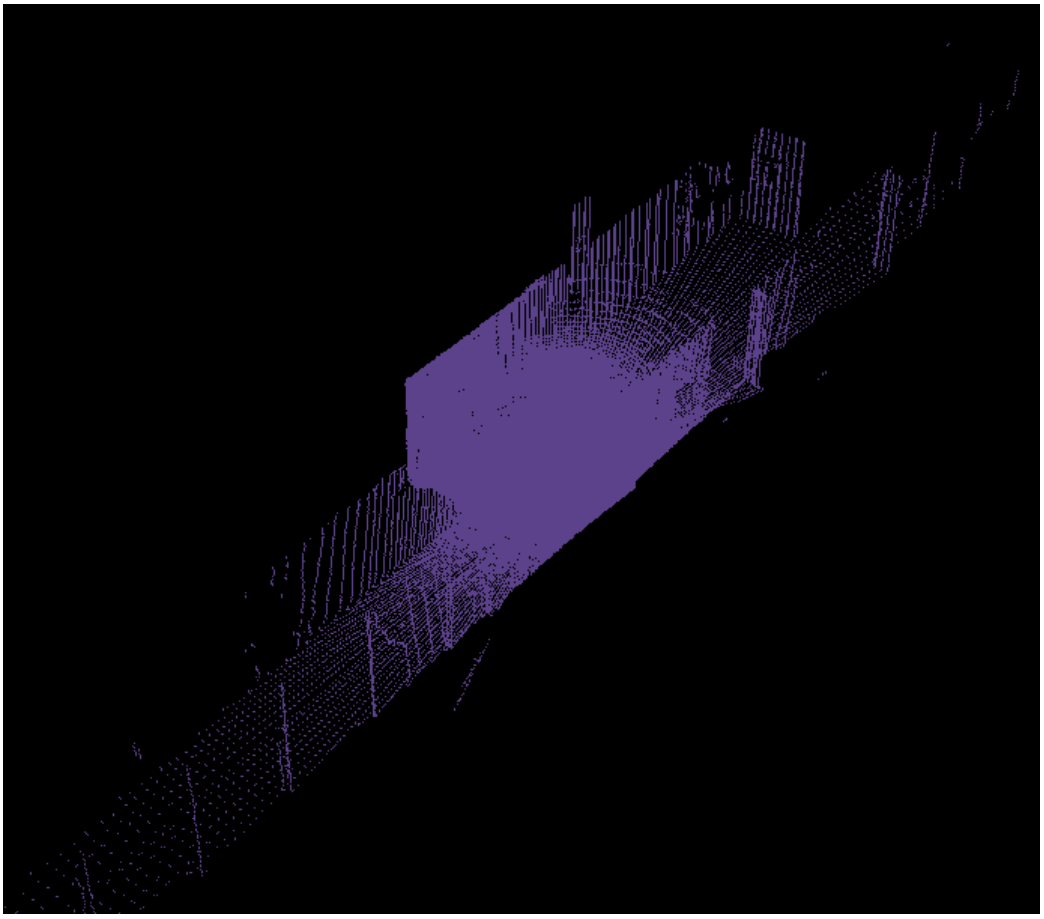


Figura 59: Nube pasillo 1 (Nº puntos: 335880).

En las figuras 60, 61 y 62 se muestran unas perspectivas cercanas al láser. Se puede ver con claridad todo el contorno del pasillo y algunos marcos de puertas cercanas.

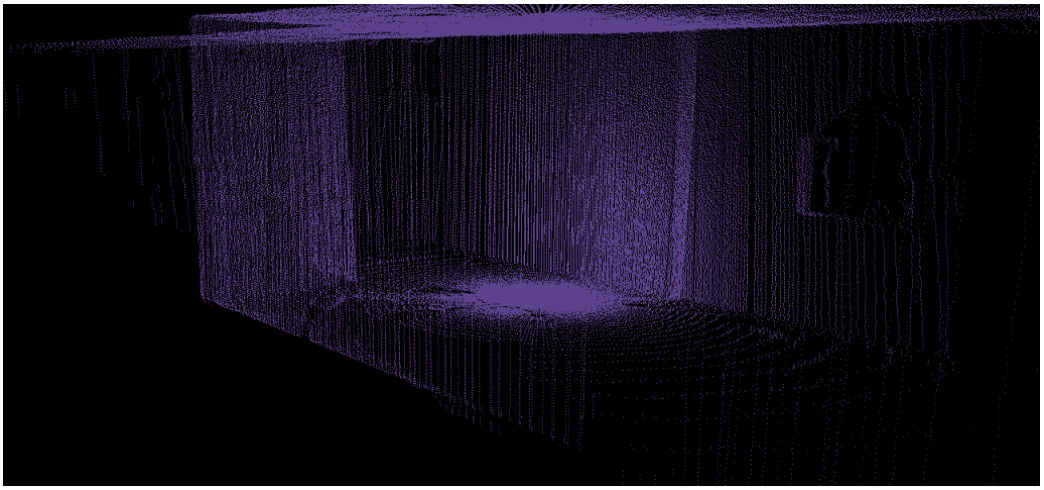


Figura 60: Nube pasillo 2 (Nº puntos: 335880).

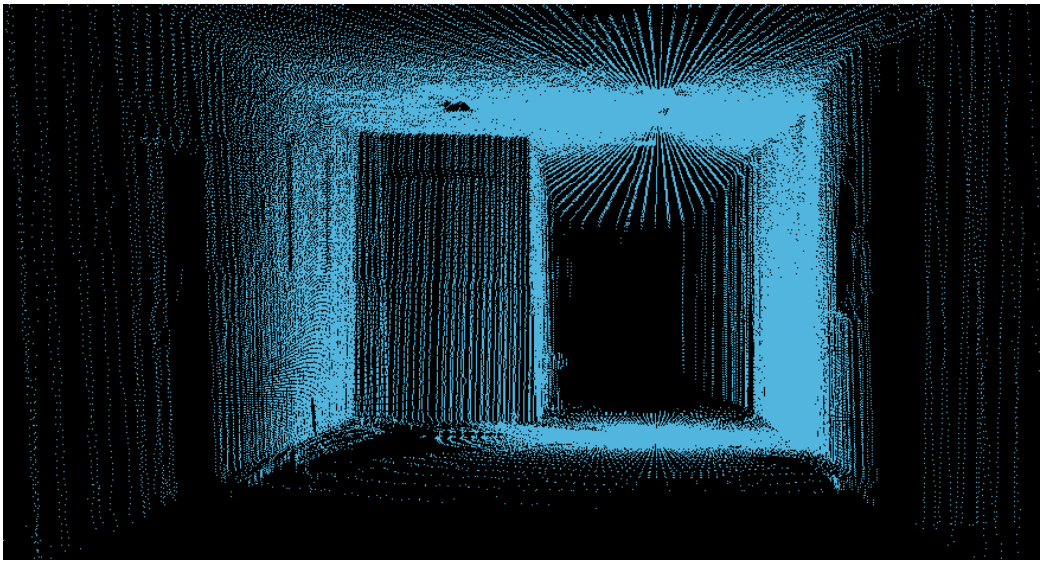


Figura 61: Nube pasillo 3 (Nº puntos: 335880).

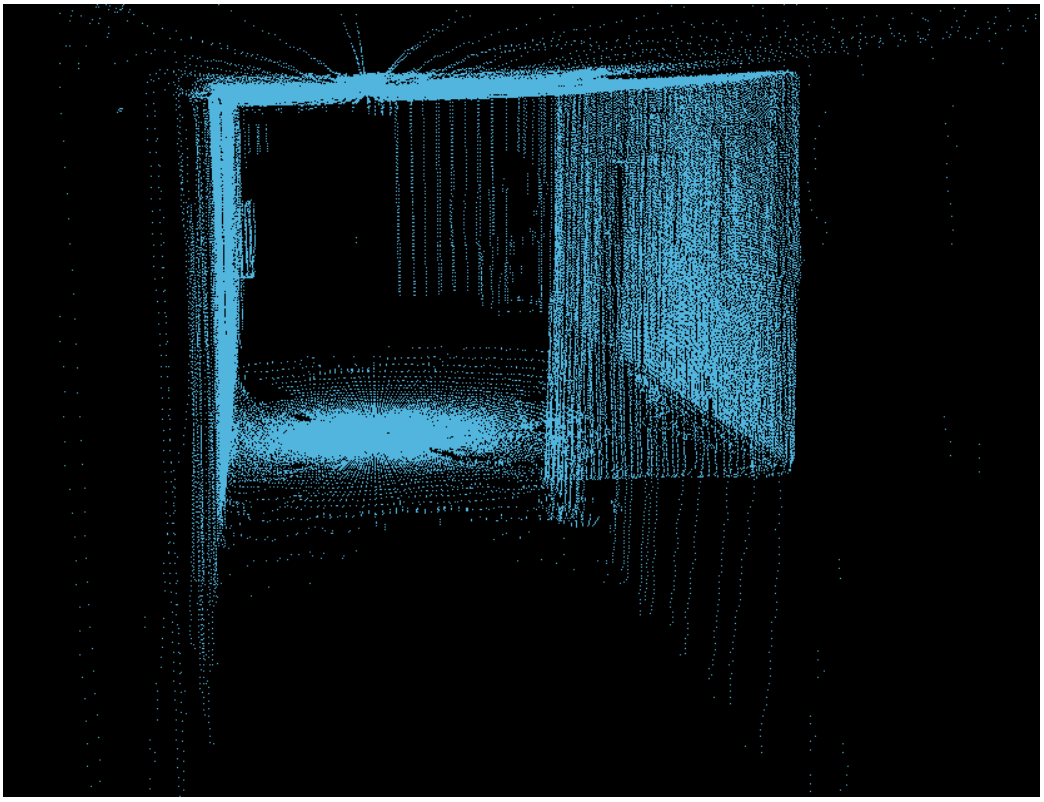


Figura 62: Nube pasillo 4 (Nº puntos: 335880).

5.4. Silla

En esta sección se muestran los resultados obtenidos en capturas de nubes de puntos con ángulos de captura pequeños, capturando puntos de algun objeto en particular o una parte de la escena. También se verán nubes tomadas a diferentes velocidades de motor para obtener mayor densidad de puntos. En particular, se han capturado nubes de puntos de una silla de escritorio y se puede ver la imagen real en la figura. 63.

Figura 63: Silla de laboratorio.

A continuación se presentan nubes de puntos tomadas de la silla con diferentes velocidades de barrido:

- La nube de puntos de la figura 64 se ha tomado usando unos 45º de barrido del motor, unos 90º de barrido del láser y una velocidad de motor de 20.

Figura 64: Silla de laboratorio: barrido velocidad 20 (Nº puntos: 107280).

- La nube de puntos de la figura 65 se ha tomado usando unos 45º de barrido del motor, unos 90º de barrido del láser y una velocidad de motor de 10.

Figura 65: Silla de laboratorio: barrido velocidad 10 (Nº puntos: 209880).

6. Conclusiones y aplicaciones futuras

Con este proyecto se ha conseguido resolver de forma satisfactoria los objetivos planteados y obtener unas muy buenas prestaciones.

- Se consigue tomar nubes de puntos desde cualquier nodo de ROS con solamente incluir unas pocas líneas de código.
- Se obtienen puntos de todo el entorno en un solo barrido.
- Se puede utilizar en todo tipo de entornos, incluido exteriores al contar con un rango máximo de 30m y hasta 60m en condiciones óptimas.
- Las nubes de puntos tomadas no necesitan de corrección de errores a causa de las partes mecánicas.
- Se ha unificado la alimentación eléctrica, pudiendo conectarse a la red eléctrica mediante un adaptador de tensiones universal o las baterías del robot MANFRED.

Además de los objetivos propuestos, se ha conseguido la posibilidad de configurar algunos parámetros para la obtención de nubes de puntos, como por ejemplo, los rangos, ángulos y velocidades de captura, lo que permite adaptar la captura a las necesidades.

Como aplicaciones futuras se pueden destacar las siguientes:

- Procesado de las nubes de puntos para el realce de las características.
- Extracción de características geométricas como rectas, planos, objetos, etc.
- Detección de planos (piso y paredes).
- Creacción de mapas 3D y 2D para navegación.

Referencias

[1] Tipos de escaner 3D, 2013. Disponible en: http://es.wikipedia.org/wiki/Escaner_3D

[2] Documentation ROS, 2013. Disponible en: http://www.ros.org/wiki/

[3] Exploration Rovers Mars, 2013, Disponible en: http://marsrover.nasa.gov/home/

[4] Robot Asimo Honda, 2013. Disponible en: http://world.honda.com/ASIMO/

[5] Velodyne LIDAR, 2013. Disponible en: http://velodynelidar.com/lidar/lidar.aspx

[6] Robotics research, Australian Robotics and Automation Association, 2013. Disponible en: http://www.araa.asn.au/research/perception.html

[7] Femto-Photography, MIT, 2013. Disponible en: http://web.media.mit.edu/ř̃askar/trillionfps/

[8] M. Fernández, D. Fernández D, C. Valmaseda, "Planificación de trayectorias para un Robot Móvil" Universidad Complutense, Madrid 2009

[9] A. Ollero, "Planificación de trayectorias para Robots Móviles", Universidad de Málaga, 1995

[10] K.S.Fu, R.C.González y Lee, "G.S.G. Robótica, Control, Detección, Visión e Inteligencia". Mc Graw-Hill, 1990

[11] R.Murphy "Introduction to AI Robotics. MIT Press", 2000

[12] A.Barrientos, L.F.Peñín, C.Balaguer, R.Aracil: "Fundamentos de robótica", McGraw-Hill, 1997

[13] P.J. Deitel, H.M. Deitel, "How to program C++", Deitel, 2007

[14] D. Álvarez, "Controlador cartesiano para el brazo LWR-UC3M-1 del robot manipulador MANFRED con detección de contacto", 2011

.

Apéndice

A. Como compilar el software

- En primer lugar tendremos que tener en una carpeta, que llamaremos laser3D para este ejemplo, todos los archivos necesarios. Podemos ver los archivos necesarios explicados en la sección 4.2.
- Despues deberemos incluir esta carpeta en el sistema de busqueda de packages de ROS. Para esto abriremos un terminal y pondremos lo siguiente:

  $ export ROS_PACKAGE_PATH=/home/user/ros/ros-pkg:/another/path
- Tras esto, deberemos entrar en un terminal y posicionarnos en la carpeta del packate:

  $ roscd laser3D
- Por ultimo, compilamos el package con el siguiente comando:

  $ rosmake

Despues de esto deberemos ver en el terminal algo parecido a lo que muestra la figura 66. Indicando que se ha terminado y se han cometido 0 fallos, en caso correcto.

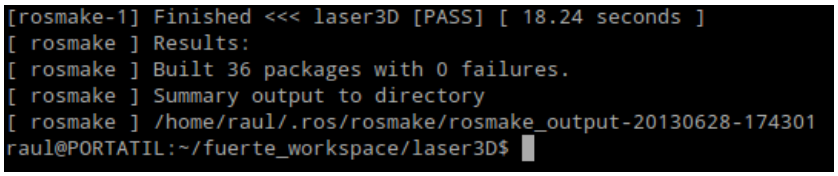


Figura 66: Resultado compilar ROS.

B. Como ejecutar el software

Lo primero que se debe hacer para que ROS funcione es lanzar el nucleo de ROS. En un terminal escribiremos lo siguiente:

$ roscore

Para ejecutar el software con ROS debemos poner en un nuevo terminal lo siguiente:

$ roslaunch laser3D laser3D.launch

De esta manera los nodos Hokuyo, Dynamixel y Laser3D serán lanzados. También se lanzara una ventana para el nodo Laser3D, que indicará el estado del proceso.

Después tenemos que lanzar el nodo Cliente, que esta programado para que pida una nube de puntos nada más lanzarlo. Lo haremos poniendo en un nuevo terminal lo siguiente:

$ rosrun laser3D bin_cliente

Si con este comando no funciona, es porque no consigue los parámetros si estos están en un archivo xml. Deberemos ir a la carpeta del archivo binario:

$ roscd laser3D $ cd bin

Y después ejecutar el node con este comando:

$ ./bin_cliente

Podemos ver, en la figura 67, la respuesta que obtendremos y veremos la nube de puntos al final del proceso.

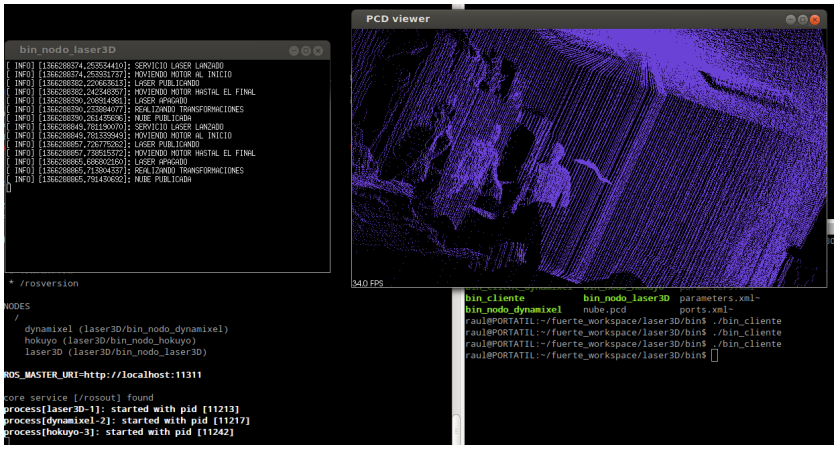


Figura 67: Resultado ejecución software.

C. Código Cliente ejemplo

```
#include "ros/ros.h"
#include "laser3D/srv_laser.h"

#include <sensor_msgs/PointCloud2.h>
#include <pcl/point_types.h>
#include <pcl_ros/point_cloud.h>
#include <pcl/ros/conversions.h>
#include <pcl/io/pcd_io.h>
#include <pcl/visualization/cloud_viewer.h>
#include <pcl/io/io.h>

 using namespace std;

 int main(int argc, char **argv)
 {

   //Inicialización nodo
   ros::init(argc, argv, "cliente");

   //Declaración variable para almacenar la nube de puntos
   pcl::PointCloud<pcl::PointXYZ> cloud;

   //Declaración variable para escritura de nube de puntos
   pcl::PCDWriter writer;

   //Punto de acceso con las comunicaciones de ROS
   ros::NodeHandle n;

   //Creación del cliente para el servicio "srv_laser"
   ros::ServiceClient client1 = n.serviceClient<laser3D::srv_laser>("srv_laser");

   //Se genera la clase para el uso del servicio
   laser3D::srv_laser srv1;


   //Parámetros motor. NECESARIO ESPECIFICAR
   srv1.request.initialPosition=0; //Posición inicial
   srv1.request.finalPosition=185; //Posición final
   srv1.request.measureSpeed=50;   //Velocidad de medida
   srv1.request.positionSpeed=50;  //Velocidad de posicionamiento

   //Parámetros laser. NO NECESARIO ESPECIFICAR, VALORES POR DEFECTO
   srv1.request.anguloMin=-45;     //Angulo inicio de medida
   srv1.request.anguloMax=225;     //Angulo final de medida
   srv1.request.rangoMin=23;       //Rango mínimo de medida
   srv1.request.rangoMax=60000;    //Rango máximo de medida

   //Llamada al servicio
   client1.call(srv1);

   //Declaración variable tipo sensor_msgs::PointCloud2
   sensor_msgs::PointCloud2 cloud2;

   //Copia de la nube a la variable cloud2
   cloud2=srv1.response.cloud;

   //Transformación entre variable sensor_msgs::PointCloud2 y
   //pcl::PointCloud<pcl::PointXYZ>
   pcl::fromROSMsg(cloud2, cloud);

   //Escritura de la nube en el archivo nube.pcd
   writer.write("nube.pcd",cloud);

   return 0;
 }
```