



Proyecto Final de Máster

Máster en Ingeniería de Sistemas y de Control

"Creación de mapas tridimensionales mediante robot teleoperado y dispositivo Kinect"

Autor : Lucas Rivière Visier

Directores: Dr. Dictino Chaos García
Dr. Luis De la Torre Cubillo

Curso académico 2015/2016 – Convocatoria de Septiembre





Proyecto Final de Máster

Máster en Ingeniería de Sistemas y de Control

Título:

"Creación de mapas tridimensionales mediante robot teleoperado y dispositivo Kinect"

Proyecto Tipo A

Autor : Lucas Rivière Visier

Directores: Dr. Dictino Chaos García
Dr. Luis De la Torre Cubillo

Autorización

Autorizamos a la Universidad Complutense y a la UNED a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firmado:



Lucas Rivière Visier

DECLARACIÓN JURADA DE AUTORÍA DEL TRABAJO CIENTÍFICO, PARA LA DEFENSA DEL TRABAJO FIN DE MASTER

Fecha: 01/09/2016

Quién se suscribe:

Autor: Lucas Rivière Visier
D.N.I.: 05289149-T

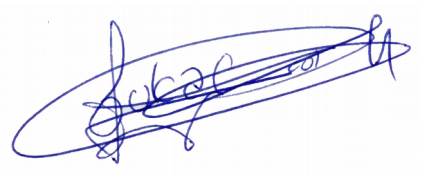
Hace constar que es el autor del trabajo:

"Creación de mapas tridimensionales mediante robot teleoperado y dispositivo Kinect"

En tal sentido, manifiesto la originalidad de la conceptualización del trabajo, la interpretación de datos y la elaboración de las conclusiones, dejando establecido que los aportes intelectuales de otros autores se han referenciado debidamente en el texto de dicho trabajo.

DECLARACIÓN:

- ✔ Garantizo que el trabajo que remito es un documento original y no ha sido publicado, total ni parcialmente, en otra revista.
- ✔ Certifico que he contribuido directamente al contenido intelectual de este manuscrito, a la génesis y análisis de sus datos, por lo cual estoy en condiciones de hacerme públicamente responsable de él.
- ✔ No he incurrido en fraude científico, plagio o vicios de autoría; en caso contrario, aceptaré las medidas disciplinarias sancionadoras que correspondan.



Fdo.

Resumen

La adquisición de datos del entorno utilizando robots móviles autónomos o controlados remotamente es un campo muy activo en la robótica y concierne a áreas tales como la inteligencia artificial o el modelado de entornos mediante el análisis de los datos adquiridos.

La correcta adquisición de los datos es esencial para la generación de una imagen fidedigna del entorno del robot. Existen diferentes algoritmos que permiten la construcción de mapas del entorno gracias a la adquisición de los datos mencionados. Muchos de estos algoritmos se basan en modelos probabilísticos. Otros, sin embargo, se basan en modelos diferenciales.

El objetivo del presente trabajo es generar una herramienta que permita realizar la adquisición de datos del entorno mediante múltiples canales. Gracias a estos datos, los usuario podrá realizar un estudio pormenorizado del entorno, combinando la información que guarda el robot por las distintas vías.

En resumen, el robot que hemos construido se trata de una plataforma con la que distintos usuarios podrán realizar experimentos remotamente. Los datos generados en cada sesión podrán posteriormente ser analizados; pudiendo aplicarse distintos modelos y combinación de los mismos que les permitan realizar prácticas en cuanto a la generación de mapas tridimensionales.

Palabras clave

Laboratorio remoto, Adquisición de datos, Kinect, javascript, Node.js, Octave

Índice general

INTRODUCCIÓN 1

1.- Introducción 2
1.1.- Marco del proyecto 2
1.2.- Motivación y objetivos 2
1.3.- Alcance del proyecto 3
1.4.- Organización del documento 4
2.- Estado del Arte 5
2.1.- Antecedentes de Investigación 5

DESARROLLO 10

3.- Conexiones 12
3.1.- Conexiones Hardware 15
3.2.- Conexiones Alimentación 16
4.- Componentes 17
4.1.- Funcionamiento de los Componentes 17
4.1.1.- Cámara Kinect I 17
4.1.2.- Cámaras Trust 20
4.1.3.- Encoders 21
4.1.4.- Sensores IR 23
4.1.5.- Raspberry Pi 2 24
4.1.6.- Arduino UNO R3 25
4.1.7.- Placa Motor Shield R3 25
5.- Software 26
5.1.- Software Libre 27
5.2.- Integración de Elementos con JavaScript 27
5.2.1.- Node.js 28
6.- Lazos de Ejecución 29
6.1.- Lazo Kinect 29
6.1.1.- Tratamiento de la información recibida por el dispositivo Kinect 31
6.2.- Lazos Webcams 34
6.2.1.- Lazo WebCam streaming 35
6.2.2.- Lazo Webcam techo 36
6.3.- Lazo Arduino 37
7.- Herramientas para la Construcción de Mapas 38
7.1.- Localización 40
7.1.1.- Modelo diferencial 40

7.1.2.- Localización mediante visión....45
8.- Control Remoto-Interfaz Gráfica....46
9.- Tratamiento de la Información Adquirida....50
9.1.- Almacenamiento de la Información....50
9.2.- Construcción del Mapa 3D....54
RESULTADOS 58
10.- Resultados de la Experiencia....59
CONCLUSIONES 74
11.- Conclusiones....75
12.- Lineas Futuras....75
Referencias-Bibliografía 77
Abreviaturas y acrónimos 81
ANEXOS 82
1.- Guía de instalación....83
2.- Puesta en Marcha....84
3.- Datasheets....89
4.- Linea temporal de los hitos del proyecto....89

Índice de Figuras

Figura 1: Stanford Cart (1961).....5
Figura 2: Stanford Cart (1963).....6
Figura 3: Stanford Cart (1980).....7
Figura 4: CMU 3-D Mapping Robot.....7
Figura 5: Groundhog Mapping Robot.....8
Figura 6: Flota Centibots 9
Figura 7: Pioneer P3DX con el módulo “quick laser mapping”.....9
Figura 8: Vistas Robot.....12
Figura 9: Componentes externos del Robot.....13
Figura 10: Medidas Robot.....13
Figura 11: Esquema interconexiones hardware.....15
Figura 12: Esquema conexiones eléctricas.....16
Figura 13: Dispositivo Kinect.....17
Figura 14: Dispositivo Kinect (componentes internos).....18
Figura 15: Geometría de adquisición del dispositivo Kinect.....19
Figura 16: Maya de puntos proyectados por el dispositivo Kinect.....20
Figura 17: Encoder Lynxmotion Quadrature Motor.....21
Figura 18: Sensor distancia Sharp.....23
Figura 19: Sensor distancia Sharp (captación señal).....24
Figura 20: Vistas placa Arduino Motor Shield.....26
Figura 21: Esquema software.....29
Figura 22: Esquema I Lazo Kinect.....30
Figura 23: Imagen consola Octave.....33
Figura 24: Esquema II Lazo Kinect.....34
Figura 25: Esquema I Lazo WebCam Streaming.....35
Figura 26: Esquema II Lazo WebCam Streaming.....36
Figura 27: Esquema Lazo Webcam techo.....36
Figura 28: Esquema I Lazo Arduino.....37
Figura 29: Esquema II Lazo Arduino.....38
Figura 30: Representación giro ruedas “Skid steer”.....42
Figura 31: Esquema I geometría del giro.....43
Figura 32: Esquema II geometría del giro.....44
Figura 33: Esquema III geometría del giro.....45
Figura 34: Interfaz gráfica 1.....48
Figura 35: Interfaz gráfica 2.....49
Figura 36: Interfaz gráfica 3.....50
Figura 37: Visualización del archivo matriz.m.....52
Figura 38: Visualización del archivo Sensor.m.....52
Figura 39: Flujo de adquisición.....54
Figura 40: Comparativa temporal.....55

Figura 41: Esquema giro en el plano.....57
Figura 42: Esquema giro en el espacio.....58
Figura 43: Fotografía del lado desde donde se inició la adquisición.....60
Figura 44: Fotografía del lado contrario de la estancia.....61
Figura 45: Fotografías estancia.....61
Figura 46: Esquema estancia.....62
Figura 47: Adquisición 1.....63
Figura 48: Adquisición 2.....64
Figura 49: Adquisición 3.....65
Figura 50: Adquisición 4.....65
Figura 51: Detalle Adquisición 4.....66
Figura 52: Adquisición 5.....66
Figura 53: Adquisición 6.....67
Figura 54: Adquisición 7.....67
Figura 55: Adquisición 8.....68
Figura 56: Detalle Adquisición 8.....68
Figura 57: Gráfica 1.....69
Figura 58: Gráfica 2.....70
Figura 59: Gráfica 3.....71
Figura 60: Gráfica 4.....72
Figura 61: Gráfica 5.....72
Figura 62: Gráfica 6.....73
Figura 63: Gráfica 7.....73
Figura 64: Linea temporal de los hitos del proyecto.....84

Índice de tablas

Tabla 1: Dimensiones....13
Tabla 2: Encoder I....21
Tabla 3: Encoder II....22
Tabla 4: Raspberry Pi II....24
Tabla 5: Arduino....25
Tabla 6: Sensores....39
Tabla 7: Archivos....52
Tabla 8: Elementos/Enseres....60
Tabla 9: Arranque....85
Tabla 10: Arranque manual....85
Tabla 11: Alternativa ServerCam12....88
Tabla 12: Datasheets....89

INTRODUCCIÓN

1.- Introducción

1.1.- Marco del proyecto

El presente proyecto se enmarca como trabajo de fin de Máster del "Máster universitario en Ingeniería de Sistemas y de Control", impartido conjuntamente por la UNED y la UCM, en el Departamento de Informática y Automática de la Escuela Técnica Superior de Ingeniería Informática (UNED).

Este proyecto está estrechamente relacionado con el trabajo de fin de Máster "Navegación de un robot autónomo utilizando balizas y luz estructurada"[1], desarrollado por Juan Ignacio Forcén Carvalho en el Departamento de Informática y Automática de la Escuela Técnica Superior de Ingeniería Informática (UNED), y presentado en la convocatoria de septiembre de 2015.

1.2.- Motivación y objetivos

El objetivo primordial del proyecto es generar una herramienta que permita el desarrollo de experiencias mediante control remoto de un robot móvil, que lleve a cabo la adquisición de datos para la caracterización del medio en el que se encuentre.

Para lograr este objetivo fue necesario dividir el proyecto en varias etapas que se realizaron con éxito:

- Investigación de la comunicación entre el dispositivo de adquisición de datos de profundidad (Kinect) y algún software que permitiese la correcta manipulación de los mismos (Octave).

- Investigación de los distintos software necesarios para el control del resto de elementos que conforman el robot.

- Integración de los diferentes dispositivos hardware en la plataforma anterior.

- Adquisición de todos los datos proporcionados por parte de los distintos dispositivos y su correcto almacenamiento.

- Establecimiento de la comunicación remota con el robot que permitiese recibir y enviar datos.

- Análisis de los datos proporcionados por el robot en situaciones reales para demostrar la correcta adquisición de los mismos.

1.3.- Alcance del proyecto

El proyecto consta de distintas etapas que han sido abordadas por separado. En primera instancia, nos dedicamos a realizar un profuso estudio sobre la manipulación de los datos aportados por el dispositivo Kinect por parte de software de análisis de datos, Matlab y Octave, en sistemas operativos de base GNU/Linux.

Una vez superada esta fase, nos centramos en la búsqueda del mejor candidato software para controlar el resto de elementos. Tras esto, nos dedicamos a implementar el código necesario para compatibilizar la adquisición de los diferentes datos aportados por los sensores y para poder realizar el control remoto sobre el dispositivo.

En cuanto a los conocimientos previos al proyecto, podemos decir que disponíamos de bastante experiencia con software como Octave y Matlab. También contábamos con amplios conocimientos de Arduino, gracias a la elaboración de distintos proyectos de diferente índole. Disponíamos, asimismo, de práctica previa con sistemas operativos en base a GNU/Linux. No contábamos con experiencia en JavaScript, pero la necesidad de su utilización propició el desarrollo de las capacidades con las que hemos podido desempeñar el proyecto.

Tras realizar con éxito la instalación de los distintos software, llevamos a cabo las

pruebas con las que pusimos en evidencia los distintos fallos o disfunciones, que finalmente pudimos depurar. En el Anexo 1 (Guía de instalación) encontraremos una sección en la que se describen los pasos a seguir para poder iniciar el proyecto desde cero. Es importante destacar que nos hemos encontrado con numerosos problemas en cuanto a la instalación de algún software, todo esto será comentado en el Anexo 1 y se describirá el procedimiento para solventar dichos problemas.

Finalmente, y tras el análisis de los datos arrojados por el robot, nos dispusimos a elaborar la presente Memoria, cuya organización se describe en el siguiente sección.

1.4.- Organización del documento

El documento está dividido en cuatro apartados bien diferenciados:

- INTRODUCCIÓN
- DESARROLLO
- RESULTADOS
- CONCLUSIONES
- ANEXOS

Cada una de estos apartados consta de distintas secciones, que se centran en aquellos aspectos que consideramos necesario desarrollar en profundidad para comprender completamente el funcionamiento de la herramienta diseñada.

En el primer apartado, “INTRODUCCIÓN”, describimos el marco, la motivación, los objetivos y el alcance del proyecto y mostramos el estado del arte.

En el segundo apartado, “DESARROLLO”, describimos todos los elementos que hemos utilizado y las interconexiones entre los mismos. Presentamos también el funcionamiento del software.

En el tercer apartado, “RESULTADOS”, exponemos los resultados fruto de la utilización de la herramienta desarrollada, mostrando la funcionalidad de la misma a través de la generación de un mapa tridimensional.

En el cuarto apartado, “CONCLUSIONES”, exponemos las conclusiones que se derivan del desarrollo del proyecto, incluyendo los objetivos logrados y posibles lineas futuras.

El quinto y último apartado, “ANEXOS”, contiene una guía de instalación para desarrollar el proyecto desde cero, una guía para la puesta en marcha del robot, una tabla con los enlaces a todas las hojas de especificaciones (*Datasheets*) de cada uno de los elementos utilizados y, por último, un esquema con la línea temporal de los hitos logrados durante el desarrollo de todo el trabajo.

2.- Estado del Arte

2.1.- Antecedentes de Investigación

La evolución que ha sufrido el campo de la robótica, y más concretamente los robots mapeadores, se desarrolla en los últimos 60 años. Entre 1960 y 1961 James L. Adams creó el Stanford Cart con conexión cableada. Este proyecto tenía como finalidad investigar los problemas del control remoto utilizando la información que aportaba una cámara de vídeo [2].

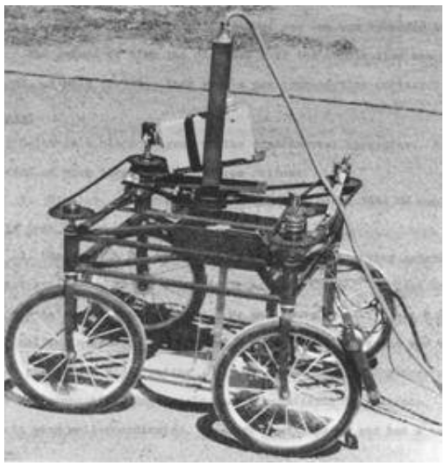


Figura 1: Stanford Cart con cable, 1961

Adams investigó la controlabilidad del vehículo y las posibilidades de evitar obstáculos generando diferentes retardos en la comunicación y trabajando a distintas velocidades.

Entre 1962 y 1963 Paul W. Braisted ideó una estrategia para mejorar la controlabilidad de este tipo de vehículos. Se le ocurrió añadir un ordenador analógico que predijera la posición del vehículo cuando una determinada orden fuese a tener efecto. Esto permitía reducir los efectos producidos por los retardos [3].

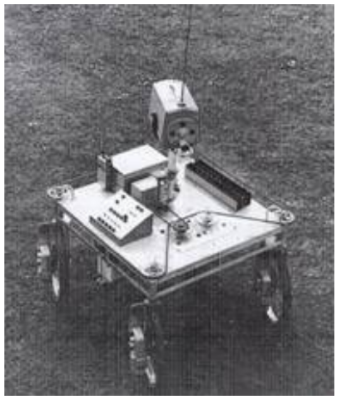


Figura 2: Stanford Cart con recepción por radio, 1963.

Sin embargo, todavía existía la limitación que suponía el retardo en la comunicación: si el retardo fuese mayor que el tiempo necesario para salvar la distancia a un determinado obstáculo, se produciría un choque con el mismo, ya que no daría tiempo a evitarlo. El retardo en las comunicaciones es un problema que siempre ha existido en los robots teleoperados. Una posible solución para evitar la colisión con los obstáculos es la incorporación de inteligencia artificial que permitan semiautomatizar el robot. Si el robot puede decidir por si mismo realizar una acción para evitar un "accidente", nos ahorraremos las consecuencias de los retardos. En [4] se expone una sección dedicada a los tiempos de retardo o latencias que presenta la teleoperación.

En el año 1966 el Laboratorio de Inteligencia Artificial Stanford (SAIL, en sus siglas en inglés) se puso en contacto con James Adams, el creador del Stanford Cart, para pedirle que les cediera el robot, con la finalidad de mejorarlo añadiéndole la tecnología necesaria para que su guiado se realizara mediante balizas visuales.



Figura 3: Stanford Cart (diseño autónomo)

Entre los años 1971 y 1980, Hans Moravec trabajó con el Stanford incorporándo nuevas funcionalidades que permitieron más versatilidad al Stanford Cart. Entre estas nuevas funcionalidades, destaca la incorporación de un eje para poder girar la cámara de visión sin necesidad de mover las ruedas [5].

Durante la décadas de los años 80’ y 90’ se han desarrollado ordenadores con mayores capacidades que han permitido llevar a cabo proyectos mucho más ambiciosos.

Entre estos, podemos citar los llevados a cabo por Carnegie Mellon University, que presenta dos proyectos de robots mapeadores con los que puede generar mapas de interiores y exteriores. El objetivo principal de este proyecto es el mapeo tridimensional de minas donde las condiciones, nocivas para el humano, no permiten el paso.

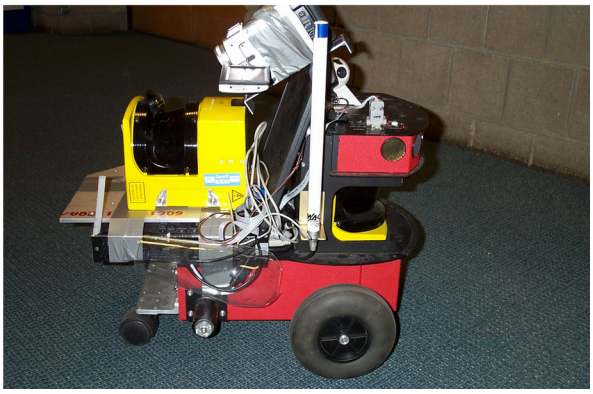


Figura 4: CMU 3-D Mapping Robot

El primero de estos dos robots tiene que ser empujado para realizar las trayectorias. Dispone de 4 telémetros láser 2-D que le permiten adquirir información de la sección transversal de la mina que hay delante del robot, y de la estructura del techo que hay por encima del mismo.

El segundo robot, cuyo nombre es “Groundhog”, fue construido por Sebastian Thrun y su equipo en el año 2003 [6]. Para su construcción utilizaron el chasis de un vehículo todoterreno, está dotado con 2 telémetros láser, dispuestos del mismo modo que en el robot anterior. Estos dispositivos le permiten generar un mapa 3-D preciso. El control del robot se realiza de manera remota. “*Groundhog*” no dispone de sensores inerciales ni de odómetros que permitan posicionar al robot. La estimación de la trayectoria se realiza mediante la aplicación de un algoritmo que realiza la interpolación de la misma en función de las adquisición de los datos recibidos por los sensores.



Figura 5: *Groundhog Mapping Robot.*

Centibots es un proyecto financiado por la Agencia de Proyectos de Investigación Avanzada de Defensa (DARPA, en sus siglas en Inglés) en EE.UU y desarrollado por SRI (Stanford Research Institute, en sus siglas en Inglés), en el año 2003 [7]. El proyecto tiene como objetivo el despliegue de 100 robots coordinados destinados a labores de vigilancia urbana o de edificios.



Figura 6: Flota Centibots.

La manera de trabajar de estos robots consiste en el envío de un primer equipo con telémetros láser que escanean el área en la que se va a realizar las labores de vigilancia. El mapa generado es compartido con todos los robots. Tras estos, un segundo equipo trabaja de manera coordinada para buscar regiones de interés en el área de trabajo (algún posible intruso). Los robots son autónomos y trabajan con independencia de cualquier infraestructura de red.

Mobile Robots desarrolló en el año 2011 el prototipo Pioneer P3DX [8]. Fue construido con el propósito de generar mapas 2-D. En primera instancia se diseñó para ser guiado por un humano, posteriormente se le añadió un módulo Wi-Fi para realizar las tareas de control y seguimiento. Este robot dispone de un telémetro láser, gracias al cual puede generar mapas 3D.

Según comenta la página web de Mobile Robots [8], Pioneer P3DX se trata de un robot robusto y manejable. Una de las aplicaciones que se mencionan es la utilización del P3DX como laboratorio de experimentación.

El movimiento de este robot se realiza por tracción diferencial y dispone de dos encoder de 500 pulsos que permiten realizar la estimación de la trayectoria. Dispone también de 8 sensores de ultrasonidos orientados hacia delante y opcionalmente 8 más orientados hacia atrás.

Una peculiaridad de este robot es que está diseñado especialmente para para poder añadir distintos módulos que amplían su funcionalidad. En la figura 7 puede apreciarse el Pioneer P3DX con un módulo “quick láser mapping” especialmente diseñado para la

realización de mapas en interiores gracias al telémetro láser que presenta este módulo.



Figura 7: *Pioneer P3DX* con el módulo *"quick laser mapping"*.

DESARROLLO

Respecto a la arquitectura del robot que presentamos, podemos separarla en dos grupos bien diferenciados. Por un lado, disponemos de los componentes hardware y sus correspondientes interconexiones. Por otro lado, tras éstos, una estructura de software cuya funcionalidad estriba en generar un diligente comportamiento de los elementos del grupo anterior.

Es evidente que una buena estructuración de uno de los grupos sin el otro, daría como resultado un mal funcionamiento del conjunto global. Por tanto, es necesario diseñar cada una de las arquitecturas teniendo siempre presente la otra.

El robot que hemos construido tiene el propósito de servir como plataforma para que los diferentes usuarios puedan realizar distintas experiencias, en las que se generarán una serie de datos. Estos datos deberán ser tratados *offline,* mediante la aplicación del modelo que se estime oportuno, para la generación de mapas en tres dimensiones.

El usuario se conectará al robot de forma remota y controlará los movimientos de éste a través de la interfaz gráfica. El robot, a lo largo del recorrido impuesto por el usuario, irá realizando una serie de adquisiciones de datos mediante los sensores de los que dispone. La descripción de dichos sensores tendrá lugar en la sección 4.1 (Funcionamiento de los Componentes). Una vez finalizada la experiencia, se podrá realizar una descarga del conjunto de los datos generados, para llevar un estudio pormenorizado de manera *offline* mediante algún software de análisis como Octave o Matlab.

En las secciones 3.1 (Conexiones Hardware) y 3.2 (Conexiones Alimentación.) exponemos una descripción de las conexiones de los dispositivos entre sí y las conexiones eléctricas que alimentarán a estos dispositivos. También detallaremos el funcionamiento de cada componente. A continuación, exponemos el funcionamiento del software y la relación de éste con el hardware descrito. También describimos el entorno gráfico del que dispone el usuario para la comunicación con el robot.

En la sección 9.2 (Construcción del Mapa 3D) exponemos la relación entre los datos adquiridos y la adaptación a distintos modelos relacionados con la generación de mapas. Por un lado, tendremos los datos que nos permitirán estimar la posición del robot en el entorno en el que se encuentre y, por otro lado, tendremos los datos que nos permitirán asociar los objetos u obstáculos en el mapa que se genere, en función de la posición que ocupe el robot en el momento de la adquisición.

Por último, describiremos el formato en el que se guardarán los datos y presentamos un diagrama donde se aprecia la manera en la que se gestionan los mismos:

3.- Conexiones

Nos centraremos en esta sección en describir las conexiones que presenta el robot. En primer lugar realizaremos una descripción de la geometría del robot y, seguidamente, llevaremos a cabo una descripción de las interconexiones tanto de hadware como de cableado. Por último, realizamos un descripción profusa de los componentes que utilizamos así como de su funcionamiento.

A continuación se muestran varias fotografías del robot que hemos diseñado:



Figura 8: Distintas fotografías del robot

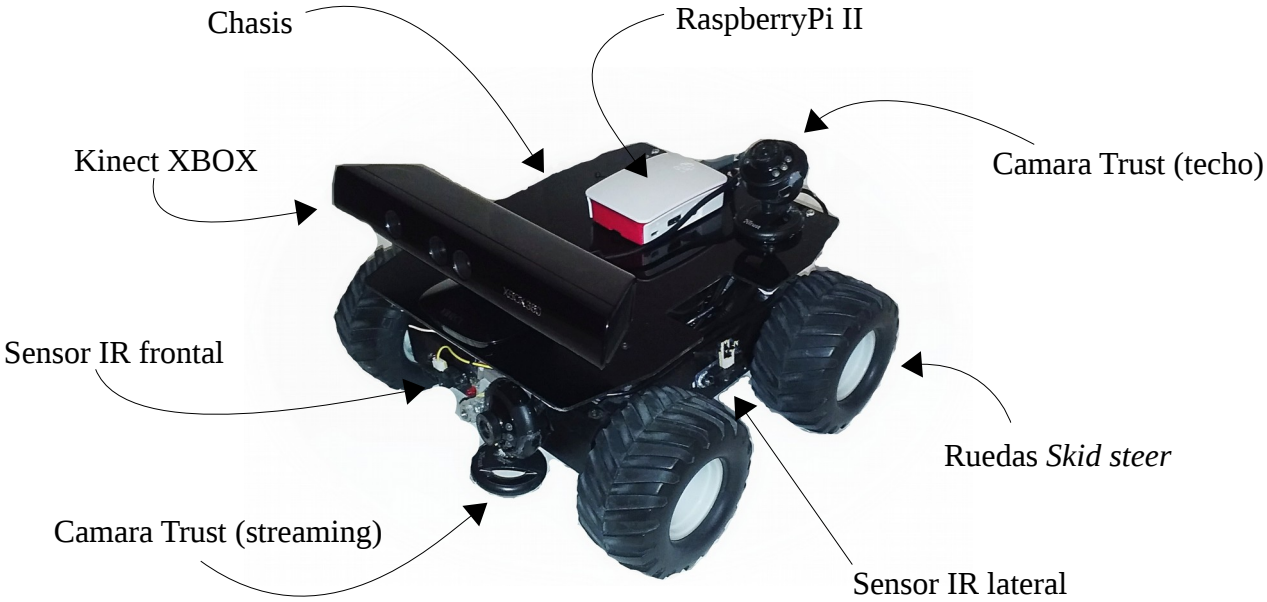


Figura 9: Fotografía del robot en la que mostramos distintos elementos externos del mismo.

El robot presenta las siguientes dimensiones:

Descripción	Valores	Detalle
Longitud	31 cm	
Anchura	34.5 cm	
Altura	21.5 cm	

Figura 10: Fotografía lateral del robot

Tabla 1: Dimensiones del robot.

El robot está dotado con cuatro ruedas “Skid steer” situadas en dos ejes paralelos. Con esta disposición su movimiento es no holónomo y los cambios en las trayectorias se logran mediante movimiento diferencial, es decir, distintas velocidades en los ruedas

izquierdas o derechas propiciarán cambios de orientación en la ejecución de las trayectorias. Realizamos una descripción más detallada de este movimiento en la sección 7.1.1 (Modelo diferencial).

Los componentes encargados de la adquisición de datos del robot son:

1) Una cámara Kinect (elemento principal para la adquisición de datos de profundidad).

2) Dos cámaras Trust.

3) Dos encoders Lynxmotion Quadrature.

4) Tres sensores de IR para captación de obstáculos.

Los elementos encargados del control y la gestión de la información percibida por el robot y de las órdenes enviadas al mismo por parte del usuario son los siguientes:

1) Una placa Arduino Uno R3.

2) Un módulo Motor Shield R3.

3) Una Raspberry Pi 2 Model B.

El robot, al ser completamente inalámbrico, recibe la energía de una batería de plomo de 12 voltios y 7 Ah. Como hemos comentado anteriormente, en el Anexo 1 presentamos una guía de instalación del software necesario para la reproducción del robot desde cero. En dicho Anexo explicamos todos los pasos a seguir y los principales problemas que surgen en la instalación del software en la Raspberry Pi, así como las soluciones que hemos encontrado para evitarlos. En el Anexo 3 encontraremos una tabla con los enlaces que nos permitirán acceder a todas las *Datasheets* de los distintos elementos hardware.

Toda esta información se presentará con el objetivo de facilitar la reproducibilidad de este robot en cualquier lugar del mundo, para su utilización como herramienta con fines educativos.

3.1.- Conexiones Hardware

En el siguiente esquema se puede apreciar las interconexiones presentes entre los distintos elementos que conforman el hadware:

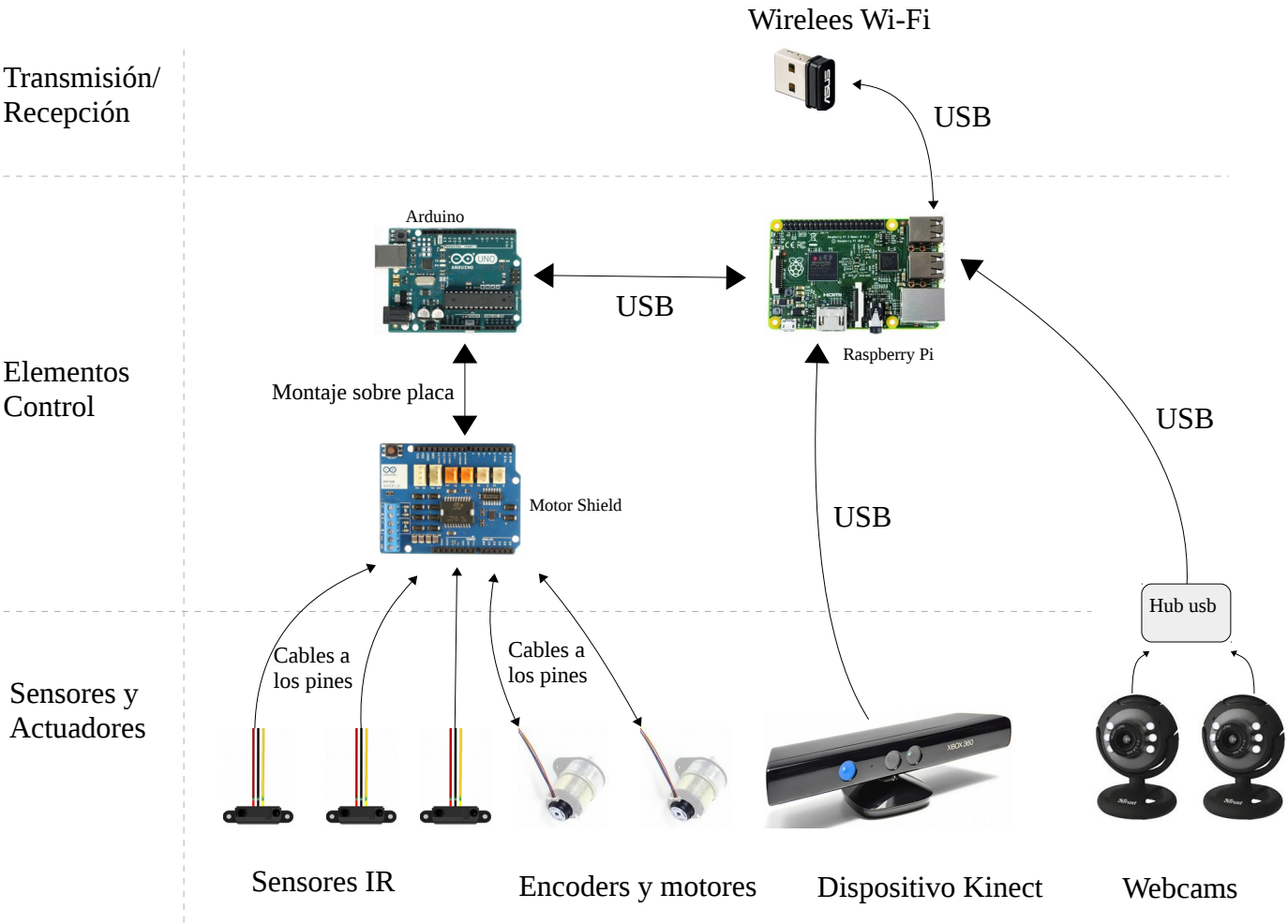


Figura 11: Esquema interconexiones hardware.

3.2.- Conexiones Alimentación.

En cuanto a las conexiones para la alimentación con la batería de plomo se decidió utilizar enchufes de mechero con salida USB para reducir el voltaje de 12V a 5V ya que estos cuentan con un estabilizador de corriente que evita dañar alguno de los dispositivos que utilizamos. Los dispositivos que funcionan con una tensión de corriente de 5 V son la Raspberry Pi, la placa Motor Shield y las dos Webcams. Los que funcionan con una tensión de corriente de 12 V son la cámara Kinect y la placa Arduino.

En el siguiente esquema podemos apreciar como son las conexiones de cada elemento en cuanto a la alimentación eléctrica:

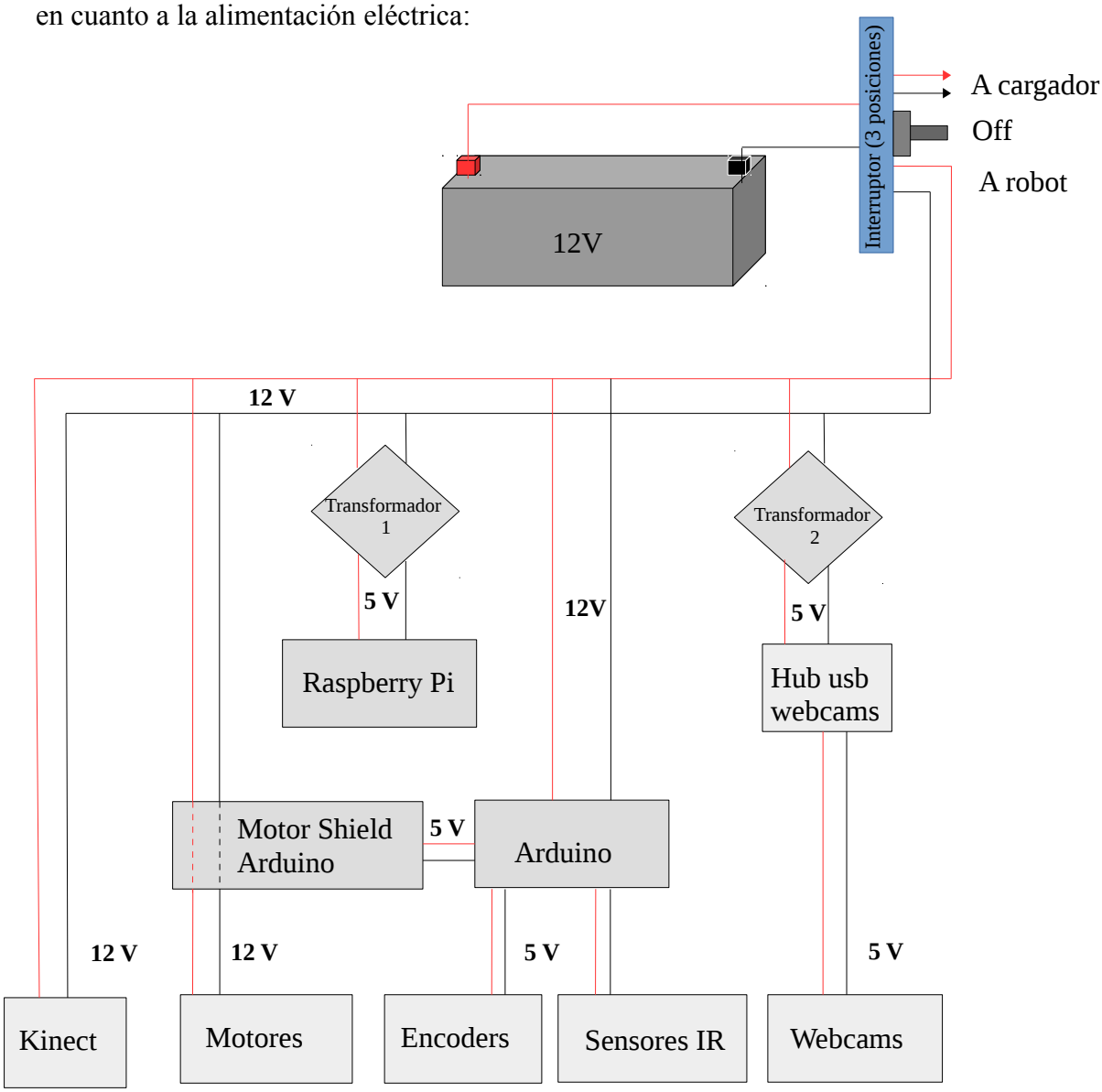


Figura 12: Esquema conexiones eléctricas.

4.- Componentes

4.1.- Funcionamiento de los Componentes

A continuación se llevará a cabo una descripción del funcionamiento de cada componente de los que dispone el robot

4.1.1 Cámara Kinect I

La cámara Kinect que utilizamos para este proyecto es un producto de Microsoft lanzado al mercado en noviembre de 2010 como un dispositivo controlador de juegos que permite a los usuarios de la XBOX interactuar con la consola sin necesidad de utilizar mandos o dispositivos en contacto directo con dichos usuarios.



Figura 13: Dispositivo Kinect

En la siguiente imagen, obtenida de la propia página de Microsoft, cuyo enlace presentamos en la Tabla 12 (Enlaces, *Datasheets*), podemos ver los distintos elementos de los que se compone este dispositivo:

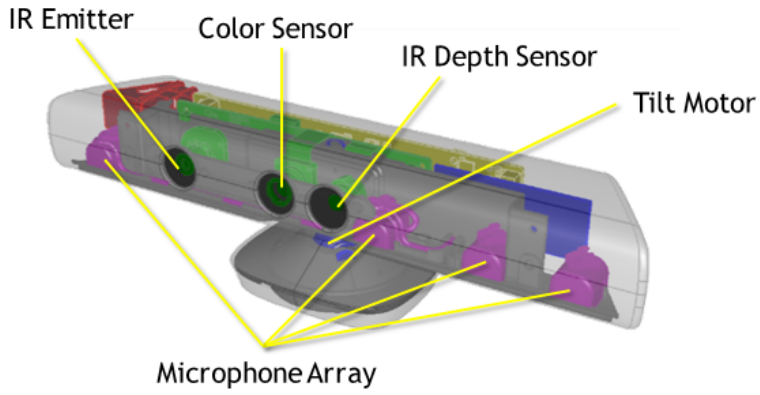


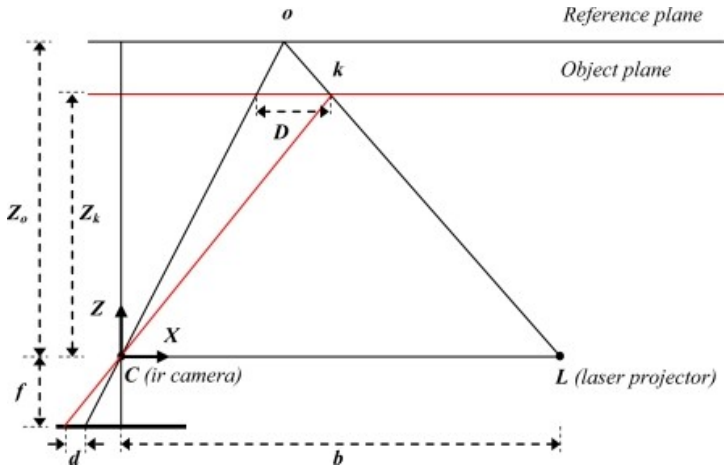
Figura 14: Dispositivo Kinect (componentes internos).

Como puede apreciarse, la cámara Kinect consta de un emisor de luz infrarroja (IR Emitter), una cámara receptora de luz infrarroja (IR Depth Sensor), cuatro micrófonos (Microphone Array), un motor de posicionamiento (Tilt Motor) y una cámara RGB (Color Sensor).

Para nuestro propósito solo utilizaremos el emisor y receptor de luz infrarroja, ya que nuestro objetivo se centrará en la adquisición de medidas de profundidad en tres dimensiones y por tanto, el resto de elementos no serán de utilidad. El rango de adquisición de datos de este dispositivo está entre 0.5 y 5m, de acuerdo con las especificaciones del producto. Se puede acceder a estas especificaciones mediante el enlace facilitado en la Tabla 12 (Enlaces / *Datasheets*) en el Anexo 3 (Datasheets).

Es importante realizar una detallada descripción del funcionamiento de este emisor y sensor de luz infrarroja para entender como se realiza la adquisición de los datos de profundidad. En [9] podemos encontrar un análisis geométrico más detallado de la manera de adquirir datos por parte de la Kinect.

El emisor de luz infrarroja genera una malla de rayos que conforman una matriz proyectada en el entorno. Esta maya tiene una frecuencia de refresco de 30 Hz y el cálculo de la distancia de cada punto se realiza por simple geometría del modo que describiremos a continuación. Para llevar a cabo la adquisición de datos, la Kinect realiza un cálculo por medio de triangulación. Estas distancias son calculadas mediante la siguiente razonamiento:



Fgura 15: Geometría de adquisición del dispositivo Kinect.

Sabiendo a que distancia se encuentra el plano de referencia Z_0, distancia que conocemos ya que es la utilizada para realizar la calibración del dispositivo, podemos tratar de averiguar a que distancia se sitúa cualquier otro objeto que pueda ser capturado. Para calcular esta distancia, tendremos que emitir un haz de luz infrarroja desde el emisor, situado en L en el esquema anterior, y comprobar que píxel excita en la pantalla de fotorrecptores de la cámara infrarroja [9]. Por semejanza de triángulos podemos apoyarnos en las siguientes expresiones:

$$\frac{D}{b}=\frac{Z_0-Z_k}{Z_0}$$

y

$$\frac{d}{f}=\frac{D}{Z_k}$$

donde Z_k es la profundidad del punto k en el espacio, b es la distancia entre el emisor de IR y el centro de la cámara IR, f es la focal de la cámara IR, d es la distancia en el eje x entre el fotorreceptor excitado del plano de referencia y el fotorreceptor excitado en una adquisición.

Uniendo ambas expresiones para deshacernos del parámetro D llegamos a lo siguiente:

$$Z_k=\frac{f\cdot b\cdot Z_0}{d\cdot Z_0+f\cdot b}$$

Lo que implica que las distancias Z_k son calculadas mediante la diferencia entre los píxeles excitados (plano de referencia-objeto puntual) y parámetros constantes que dependen de la geometría del dispositivo.

Sucesivamente se realiza la operación anterior con cada uno de los haces que conforman la maya proyectada sobre el habitáculo con una frecuencia de 30 Hz. Todos las distancias son almacenadas para ser gestionadas por el software pertinente.

En la siguiente imagen puede apreciarse la maya de puntos proyectada por la Kinect [10]:

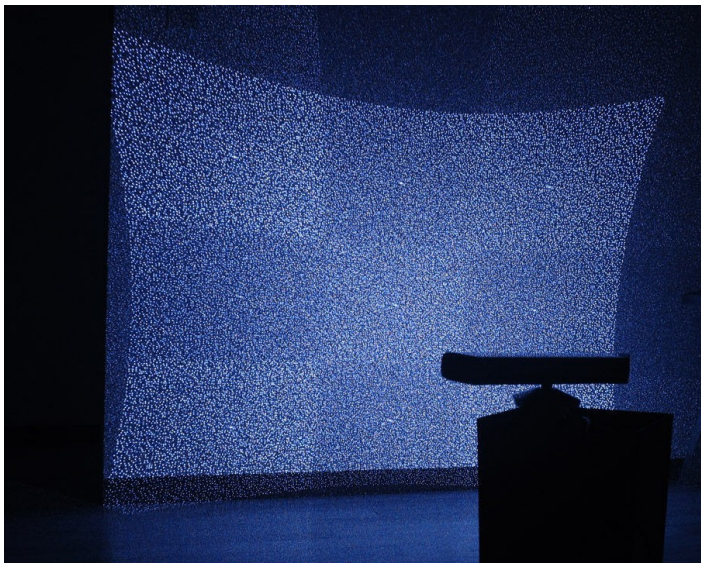


Figura 16: Maya de puntos proyectados por el dispositivo Kinect

4.1.2 Cámaras Trust

Las dos cámaras que utilizamos serán de la marca Trust y entre sus características podemos citar que presenta una resolución de 1,3 mega-píxeles (1.280 x 1.024). Una de ellas se utilizará para captar los códigos QR o códigos de color que posicionamos en el techo para realizar la localización mediante balizas visuales, técnica que utilizó Juan Ignacio Forcén Carvalho en el desarrollo de su trabajo de investigación [1]. La otra se utilizará para realizar un *streaming* de la visión del robot de tal forma que el usuario sepa en todo momento los obstáculos que hay frente al robot.

En [4] se describe el uso de Webcams en los robots *online* con el fin de la manipulación remota y la movilidad. Se puede encontrar en este libro también una sección dedicada a los tiempos de retardo o latencias que presenta la teleoperación.

4.1.3 Encoders

Los encoders que utilizamos son *"Lynxmotion Quadrature Motor Encoders"* y disponen de 6000 pulsos por vuelta. Presentan dos canales para dos señales diferenciadas que nos permitirán saber tanto el número de pulsos avanzados como la dirección en la que se han realizado.

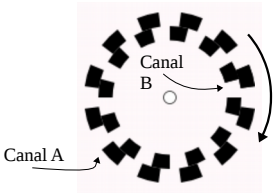


Figura 17: Encoder Lynxmotion Quadrature Motor

De los cuatro cables que salen del enconder dos estarán destinados a la alimentación del mismo (5V), estos serán el rojo (+) y el negro (-). Los otros dos están destinados a las señales que nos permitirán contabilizar el número de pasos. Cuando el encoder gira, permite o impide el paso de un haz de luz que estará asociado a las correspondientes señales de los cables rojo y verde (cada uno tendrá su respectivo haz de luz). En caso de que se permita el paso del haz el cable asociado mantendrá una señal que será directamente dirigida a los pines de Arduino, donde se almacenarán el número de cuentas. El nombre de *Quadrature* proviene de los cuatro posibles estados en los que puede encontrarse el encoder respecto a las señales de salida [11].

En la siguiente tabla podemos visualizar los cuatro estados posibles que pueden presentar las señales de salida y su correspondencia con el estado del encoder:

En el caso de que se produzca un giro horario:

Giro Horario			
Fase	Canal A	Canal B	Esquema
1	0	0	
2	1	0	
3	1	1	
4	0	1	

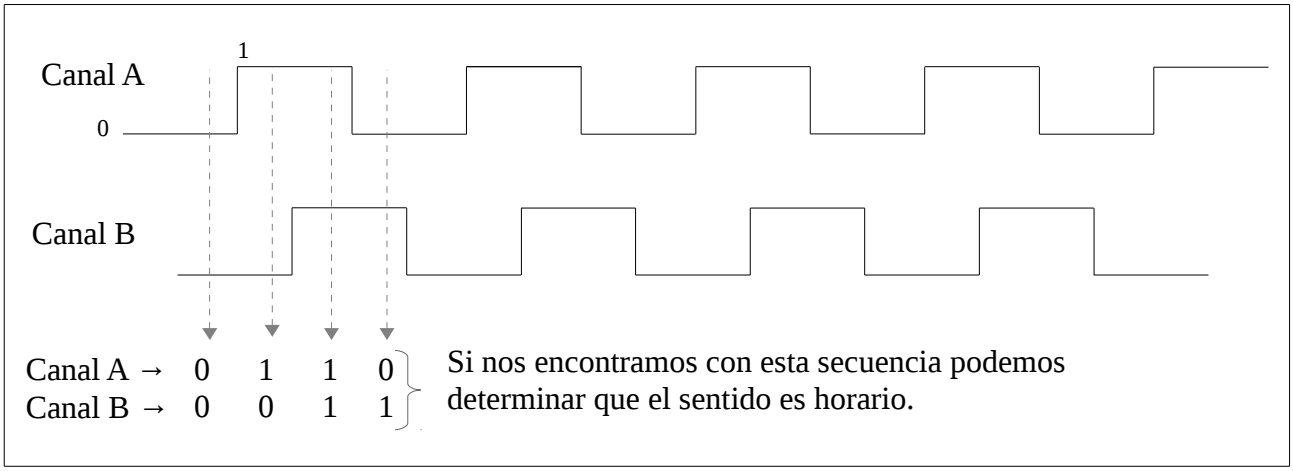
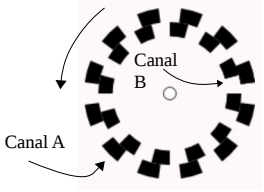


Tabla 2: Encoder I.

Los valores de las columnas Canal A y canal B, se corresponden con los siguientes estados de la señal de los cables verde y amarillo: 1→ Señal activa ; 0 → Señal inactiva.

En el caso de un giro antihorario:

Giro Horario			
Fase	Canal A	Canal B	Esquema
1	0	1	
2	1	1	
3	1	0	
4	0	0	

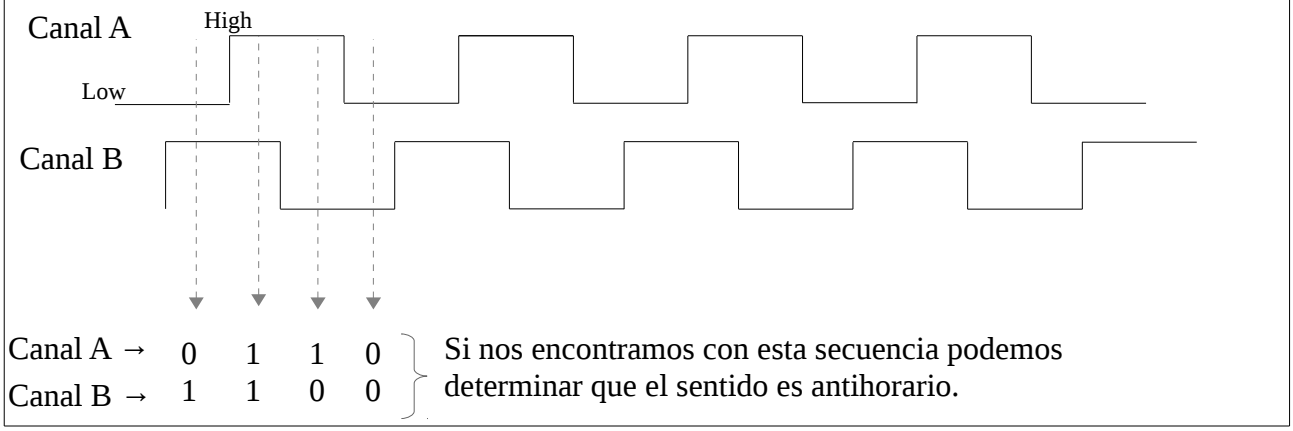


Tabla 3: Encoder II.

En [4] podemos encontrar un capítulo completo dedicado a la física de los encoders. En él se especifican las diferencias entre los distintos tipos de encoders.

4.1.4 Sensores IR

Los sensores infrarrojos que vamos a utilizar serán los Sharp GP2Y0A21 F55 IR. Presentan un rango de acción de 10 a 80 cm. El funcionamiento de este dispositivo consiste en el cálculo de la distancia mediante la geometría que forma el haz emitido con el haz recibido por un array de foto-transistores infrarrojos [12].

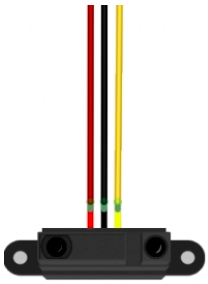


Figura 18: Sensor distancia Sharp

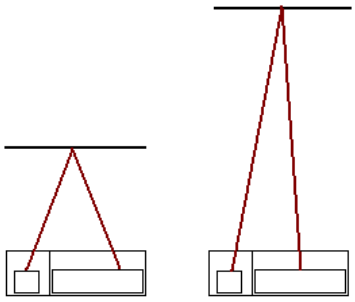


Figura 19: Sensor distancia Sharp (captación señal)

El haz infrarrojo es emitido por un diodo infrarrojo que, tras rebotar en el objeto u obstáculo, regresa al sensor y excita algún foto-transistor infrarrojo del array. La distancia que separa el sensor del obstáculo está relacionada con el ángulo que forma el haz incidente con el haz reflejado.

Siendo el haz lo suficientemente estrecho excitará un único foto-tarnsistor del array y cada foto-transmisor se corresponderá con una distancia concreta. El array de foto-transmisores se llama PSD (Position Sensitive Detector). Su funcionamiento es similar al de una cámara CCD pero en vez de presentar una disposición matricial presentará una disposición lineal.

4.1.5 Raspberry Pi 2

Para poder alojar el servidor y realizar las comunicaciones con todos los elementos hardware, se necesita un pequeño ordenador. Pensamos que la mejor opción es Raspberry Pi 2 modelo B ya que se trata de un ordenador de bajo coste (ronda los 40 euros) que se adapta perfectamente a nuestras necesidades.

Raspberry Pi 2 consta de una diminuta placa base de 85 x 56 milímetros en la que se aloja un chip Broadcom BCM2836 SoC con procesador Quad-core ARM Cortex-A7 de hasta 900 MHz, GPU VideoCore IV y 1GB de memoria RAM.

La alimentación se realiza a través de un micro USB socket 5V, 2A. En la siguiente tabla podemos encontrar las conexiones más destacadas que presenta Raspberry Pi 2 Model B:

Conexión	**Descripción**
Ethernet	10/100 BaseT Ethernet socket
Salida de vídeo	HDMI (rev 1.3 & 1.4)
Salida de audio	3,5mm jack, HDMI
USB	4 USB 2.0 Connector
Conexiones GPIO	27 GPIO pins (+3.3 → + 5 V) + GND
Conexión Cámara	15-pin MIPI Camera Serial Interface (CSI-2)
Ranura para memory Card	Micor SDIO

Tabla 4: Especificaciones Raspberry Pi II model B.

El sistema operativo que instalamos fue Raspbian. Este es el más aconsejado para poder utilizar Octave. Por supuesto, este sistema operativo es compatible con el resto de elementos. Durante el desarrollo del proyecto nos planteamos el uso de el software Matlab, pero finalmente nos decantamos por utilizar el software Octave, ya que, a día de hoy, Matlab no puede ser instalado en dispositivos que presenten procesadores con arquitectura ARM, información que podemos encontrar en las especificaciones de Matlab [13].

4.1.6 Arduino UNO R3

Arduino es una placa compuesta por un circuito impreso con un microcontrolador reprogramable (ATmega328). Es ideal para proyectos multidisciplinares en el ámbito de la robótica y electrónica. Arduino dispone de pines digitales y analógicos para el tratamiento de señales de entrada y salida. Gracias a dichos pines podemos adaptar distintas placas de extensión, en nuestro caso, la placa Shield R3 para el control de los motores, que aumenta la funcionalidad de Arduino.

También dispondremos de una entrada USB que nos permite programar la placa, alimentarla y realizar la transacción de datos. Al principio, utilizamos la entrada de USB para alimentar, programar y realizar la transacción de datos. No obstante, a medida que avanzamos en el proyecto, nos dimos cuenta de que cada elemento hadware requería de su propia alimentación. Por este motivo alimentamos la placa Arduino a través de un conector de 12 V y no a través del cable USB. Cuantos menos dispositivos se alimenten a través de Raspberry Pi, más estable será su funcionamiento.

En la siguiente tabla se pueden mostrar algunas de las características que presenta Arduino:

Elemento	**Características**
Microcontrolador	ATmega328
Voltaje operativo	5V
Pins Digitales (Entrada/Salida)	14
Pins Analógicos de Entrada	6
DC corriente para I/0 Pines	40 mA
DC corriente para 3.3V Pines	50 mA
Memoria Flash	32 KB

Tabla 5: Características Arduino UNO R3.

4.1.7 Placa Motor Shield R3

La placa Motor Shield R3 nos permite realizar el control de los motores de manera eficiente. El funcionamiento de la placa se basa en el microprocesador L298, se trata de un circuito integrado diseñado para manejar elementos inductivos tales como relés,

solenoides, motores de corriente continua (como es nuestro caso) y motores paso a paso [14].

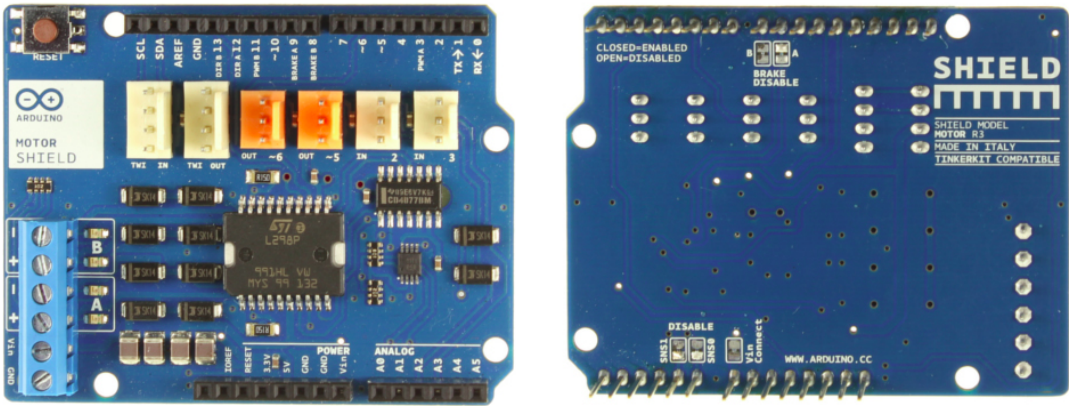


Figura 20: Vistas placa Arduino Motor Shield R3:
a)Vista frontal de Arduino Motor Shield R3.
b) Vista trasera de Arduino Motor Shield R3.

La placa se encaja encima de Arduino introduciendo las conexiones en los pines de la placa Arduino. Esta se encargará de activar o desactivar los 4 motores de las ruedas del robot en el sentido que le ordene Arduino.

Comentamos en este punto que tuvimos que realizar una modificación del hardware de ARDUINO. Los pines 2 y 3 de Arduino son necesarios para llevar el contador de los encoders. La placa Motor Shield R3 utiliza el pin 3 como entrada de la señal PWM, que determina la cantidad de corriente que se suministra a un determinado motor. Esta coincidencia de pines nos obligó puentear el pin 3 de Arduino R3 para dejar su salida en el pin 10 de la placa Motor Shield R3. Por lo tanto, en la programación de Arduino, para llevar a cabo las cuentas del encoder, programamos en función del pin 2 y 3, pero cableamos en los pines 2 y 10 para cada una de las señales de salida de los encoders.

5.- Software

En esta sección mostramos la estructura del software y explicamos los aspectos más relevantes del funcionamiento del mismo.

5.1.- Software Libre

Para el conjunto de tareas y rutinas que nos permiten gestionar la información que recibimos del robot y las órdenes que enviamos, hemos utilizado software libre. En un primer momento nos planteamos la instalación de Matlab para la gestión de los datos aportados por la Kinect. Sin embargo, no es factible la instalación de este software en la Raspberry Pi (como hemos comentado en la sección 4.1.5 (Raspberry Pi 2)), ya que, a fecha de inicio del proyecto, Matlab no es soportado por dispositivos con arquitecturas tipo ARM como la que presenta Raspberry Pi 2. Surgió entonces el planteamiento de instalar otro software similar, que nos permitiera gestionar los datos del dispositivo Kinect. Octave ha sido un estupendo candidato con excelentes resultados.

Levamos a cabo la instalación de Octave en Raspberry Pi y, tras una intensa búsqueda de información, nos decantamos por la utilización de las librerías Psychtoolbox-3 [31] que nos permiten la comunicación con el dispositivo Kinect. Gracias a las cuales hemos podido gestionar los datos de profundidad en forma matricial.

Todos los pasos que hay que seguir para llevar a cabo la instalación de este software y las librerías mencionadas son descritos en el Anexo 1 (Guía de instalación).

El conjunto de las placas Arduino R3 y Arduino Shield R3, encargadas del control de los motores y de la lectura de los sensores de proximidad y encoders, dispone del software propio de Arduino. La programación se realizó desde el propio entorno de desarrollo (IDE) de Arduino. Entorno gratuito que puede descargarse desde la página oficial de Arduino [15].

Como hemos comentado en la sección 4.1.2 (Cámaras Trust) utilizamos dos cámaras web; una para la adquisición de imágenes del techo y otra cámara que nos permitirá realizar el streaming para el control del robot por parte del usuario remoto. Utilizamos JavaScript y Node.js para la gestión de la información que obtenemos de ambas cámaras. Las imágenes que se obtienen del techo serán guardadas en el lado del servidor (Raspberry pi) para ser enviadas al cliente una vez finalizada la sesión, y las imágenes de la cámara frontal, que muestran la visión del robot, son enviadas al navegador, en el ordenador remoto, para visualizar lo que está viendo el robot.

5.2.- Integración de Elementos con JavaScript

Una vez instalados los software que nos permiten la comunicación con los sensores y actuadores, necesitamos algún lenguaje que unifique todos los elementos y nos

permitiera la gestión centralizada de la información que recibimos de los sensores y las órdenes que envía el cliente.

Este lenguaje es Javascript, se trata de un lenguaje de programación interpretado, orientado a objetos y que es una estupenda herramienta para dinamizar la interfaz del cliente del lado del navegador [16], desde donde el usuario podrá gestionar el control del robot. Por supuesto también nos es útil utilizarlo en el lado del servidor (Raspberry Pi) para gestionar toda la información que arrojan los distintos dispositivos.

5.2.1 Node.js

Node.js es un entorno javascript del lado del servidor, basado en eventos. Generalmente Javascript ha sido un lenguaje de programación del lado del cliente que se ejecuta en el navegador [16]. No obstante, gracias a Node.js, podemos implementar el código JavaScript para ser utilizado del lado del servidor, creando programas de red que nos permitan gestionar la información y el control de nuestro robot.

En el siguiente esquema UML podemos apreciar los enlaces que existen entre los distintos elementos descritos:

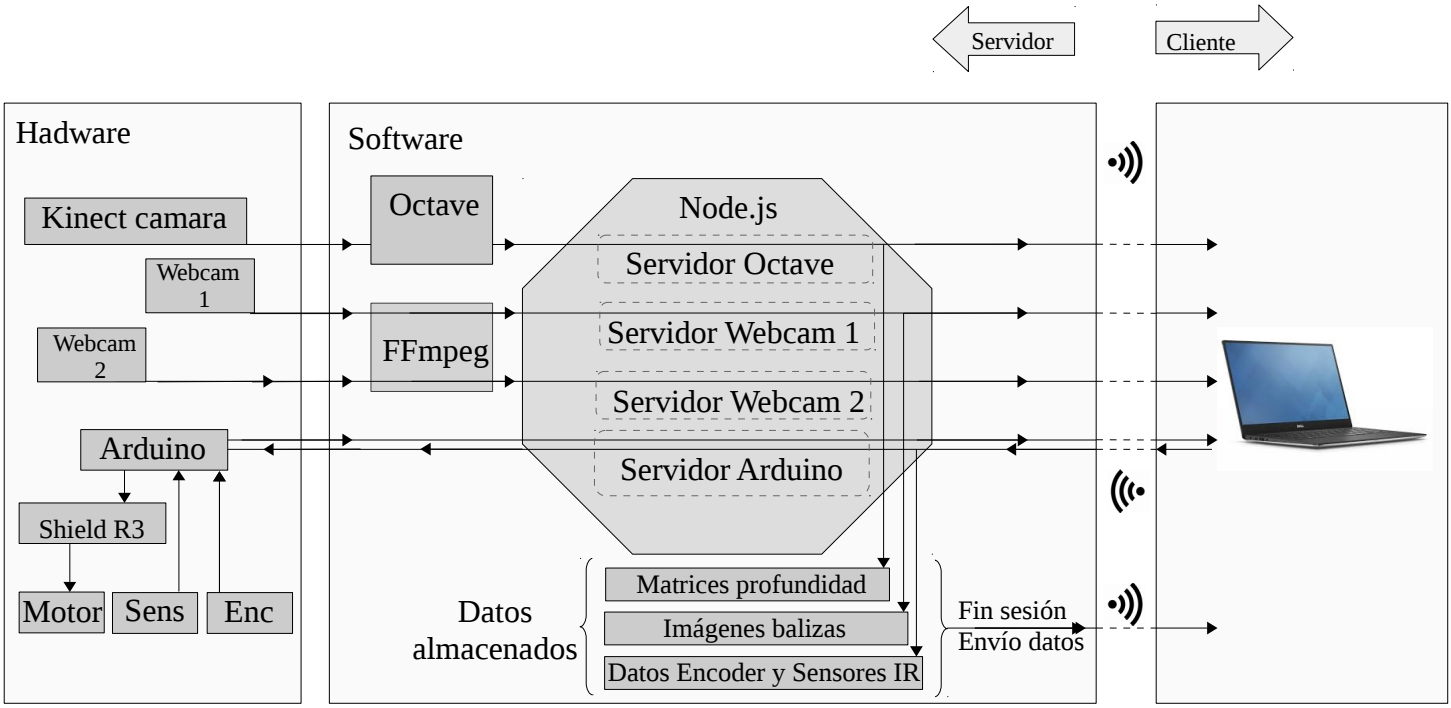


Figura 21: Esquema software.

En las siguientes secciones procederemos a explicar en detalle el funcionamiento de cada uno de los lazos que conforman el dispositivo global.

6.- Lazos de Ejecución

En esta sección presentamos los distintos lazos de ejecución que tienen lugar durante la experiencia con el robot. Cada lazo se encargará de tramitar la información correspondiente para poder llevar a cabo el proceso global de manera coordinada.

6.1.- Lazo Kinect

Este lazo es el encargado de recibir las matrices con los datos de profundidad que ha adquirido el dispositivo Kinect. Las matrices con los datos de profundidad son tratadas en primera instancia por Octave, desde donde se realizarán diferentes operaciones.

Octave recibe la matriz de profundidad y realiza un análisis de la misma para comprimir sus valores, conseguiremos con ello, reducir el espacio de memoria que ocupa la matriz con los datos de la adquisición, siendo la gestión posterior en relación a los envíos al cliente, más eficiente. ¿En qué consiste esta compresión de valores? ¿Cómo se lleva a cabo?. Son preguntas que debemos responder para entender el correcto tratamiento de estas matrices. En la siguiente sección, 6.1.1 (Tratamiento de la información recibida por el dispositivo Kinect), explicamos en qué consiste este tratamiento de compresión.

Presentamos a continuación un esquema de los distintos elementos involucrados en el lazo Kinect-Octave-Cliente:

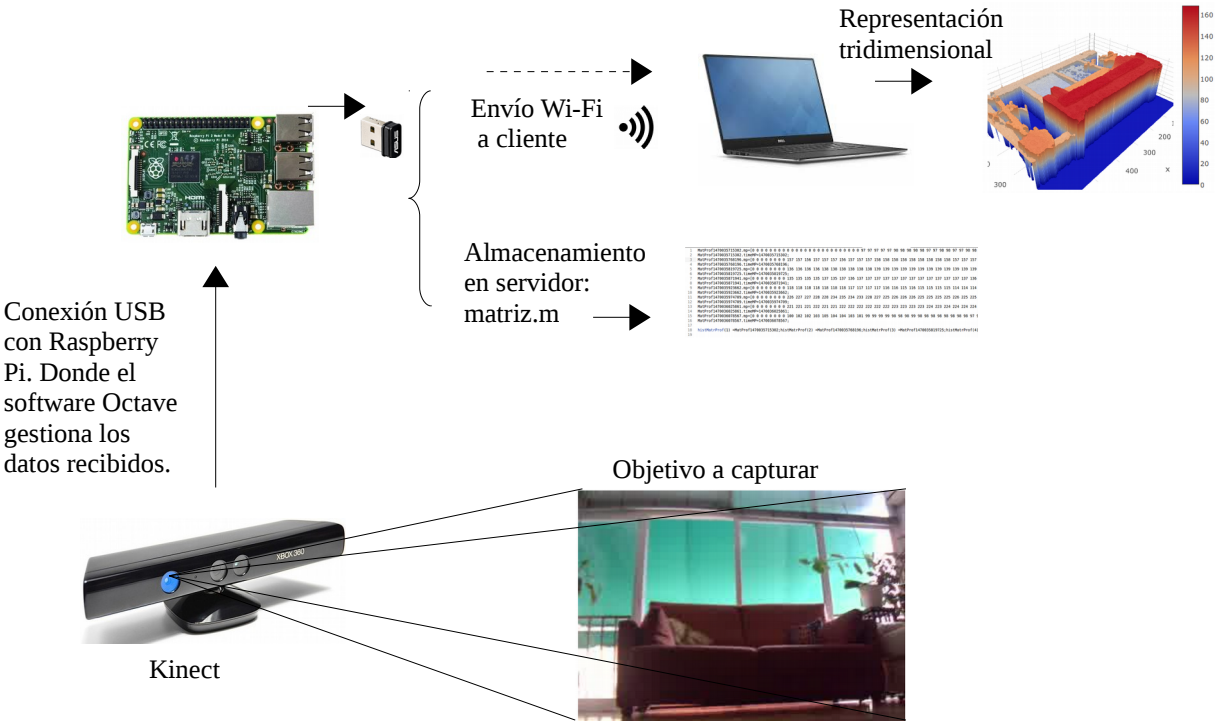


Figura 22: Esquema I Lazo Kinect.

El dispositivo Kinect está directamente conectado con la placa Raspberry Pi 2 mediante USB. En Raspberry Pi 2 se aloja el servidor encargado de recibir las matrices por parte de Octave, también instalado en Raspberrypi 2. Este servidor se denomina **ServerOctave.** En resumen, es Octave el software encargado de recibir los datos de profundidad como matrices, y éste realiza la compresión de los datos para disminuir el espacio que ocupan.

Hemos diseñado un programa en Octave que está en continua comunicación con el dispositivo Kinect, mediante un bucle en el que disponemos de un contador que, al llegar a un determinado límite, realiza un envío a **ServerOctave** . Este límite ha sido fijado en 70 iteraciones, produciéndose un envío de datos cada 30 segundos aproximadamente. No es aconsejable disminuir esta frecuencia de envío, ya que tiene que dar tiempo a manejar el robot remotamente para colocarlo en una posición nueva que nos permita la adquisición de nuevos datos. Si se aumenta la frecuencia de envío de matrices corremos el riesgo de generar archivos demasiado pesados y que alberguen información redundante.

Una vez recibida la información en **ServerOctave**, éste se encargará de guardarla en un archivo denominado matriz.m, del que hablaremos en la sección 9.1 (Almacenamiento de la Información), y también será enviada al cliente, en formato JSON, para su representación tridimensional en el interfaz del usuario, alojado en un ordenador remoto.

6.1.1 Tratamiento de la información recibida por el dispositivo Kinect

Las matrices que Octave recibe de la Kinect presentan una dimensión de 480x640, que se corresponde con el número de píxeles de la cámara receptora de infrarrojos. Por lo que por cada píxel tendremos un dato de profundidad en formato "double". El formato "double" utiliza 8 bytes por cada elemento, de tal forma que siendo las matrices de 480x640 pixeles, tendremos 307200 elementos y, como cada elemento es de formato double, ocuparán 2457600 Bytes: (8·307200 = 2457600 Bytes).

Como decíamos, en [9] podemos encontrar un análisis geométrico de la manera de adquirir datos por parte de la Kinect y en el cual nos basaremos para poder realizar la compresión de los datos de profundidad.

Realizándose la compresión de los datos podremos reducir el tamaño que ocupan las matrices para llevar a cabo operaciones de envío y guardado de manera más eficiente.

En la sección 4.1.1 (Cámara Kinect I) llevamos a cabo un razonamiento por el cual llegamos a la conclusión que las distancias Z_k se calculaban en función de la diferencia de píxeles excitados (plano de referencia-objeto puntual) y parámetros constantes que dependen de la geometría del dispositivo. Recordemos la expresión a la que llegamos:

$$Z_k = \frac{f \cdot b \cdot Z_0}{d \cdot Z_0 + f \cdot b}$$

Podemos realizar un proceso de compresión de cada medida. Para ello expresaremos las mismas en función del parámetro d (distancia en número de píxeles). Como d es un número entero, ya que es la diferencia entre un número de píxeles determinado, podemos construir una matriz con los valores enteros que se correspondan con estas medidas. Para recuperar la matriz con los valores de las distancias reales, simplemente haremos la operación inversa.

$$d_i = \frac{Z_{ki}^{-1} \cdot (f\ b\ Z_0) - f\ b}{Z_0}$$

$$d_i = \frac{Z_{ki}^{-1} - Z_0^{-1}}{(f \cdot b)^{-1}}$$

Tras un análisis de las primeras matrices adquiridas llegamos a la siguiente expresión:

$$d_i = \frac{Z_{ki}^{-1} - 0{,}0018753817}{0{,}0030711016}$$

El parámetro *d* es un número entero y, por tanto, podemos guardarlo en formato uint8, que solo ocupa un Byte por dato. De tal modo que conseguimos reducir 8 veces la matriz de los datos de profundidad gracias a la aplicación de este sencillo método. Una vez reducida la matriz, ésta será guardada en un archivo, en el cual se irán almacenando todos los valores de las matrices adquiridas y cuyo formato será explicada más adelante. También aprovecharemos que está reducida para ser enviada al cliente para su representación tridimensional en la interfaz del usuario.

Una vez finalizada la sesión y descargados los datos, se podrán volver a descomprimir siguiendo los pasos contrarios. Como hemos comentado antes, para descomprimir la matriz bastará con hacer la operación inversa:

$$Z_{ki} = (d_i \cdot 0{,}0030711016 + 0{,}0018753817)^{-1}$$

La siguiente imagen muestra una captura de la consola de Octave en la que aparece el tamaño de la matriz original y la matriz comprimida.

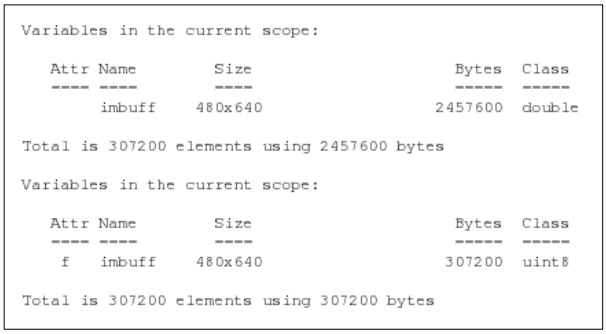


Figura 23: Imagen consola Octave.

$$\frac{Bytes_{original}}{Bytes_{comprimida}} = \frac{Bytes_{double}}{Bytes_{uinit8}} = 8$$

Como se puede apreciarse, la matriz comprimida ocupa ocho veces menos que la matriz original y sin pérdida de información. La recuperación de la información es íntegra ya que, por lo expuesto anteriormente, el cálculo de las distancias reales se efectúa utilizando las distancias entre los píxeles, que recordemos que son números enteros, por tanto, en realidad nos interesa quedarnos con la distancia entre los píxeles para más tarde calcular a qué distancias reales se corresponden.

Una vez comprimidas las matrices, Octave las enviará al servidor **ServerOctave** donde se realizarán dos operaciones. En primer lugar, éstas matrices son guardadas, con sus valores comprimidos, en una carpeta del servidor, en un único archivo de extensión "*.m*", para su tratamiento y posterior análisis por algún software como Octave o Matlab. En segundo lugar, éstas matrices son enviadas al navegador con el que el usuario está realizando la experiencia para su representación tridimensional.

Al iniciarse el servidor, éste permanece a la espera hasta que recibe alguna matriz por parte de Octave con los datos comprimidos. Una vez recibidos se realiza las gestiones mencionadas y se repite la operación tantas veces como matrices se reciban. A continuación podemos visualizar un esquema con un diagrama de flujo en el cual se puede apreciar los pasos que se llevan a cabo en el lazo Kinect:

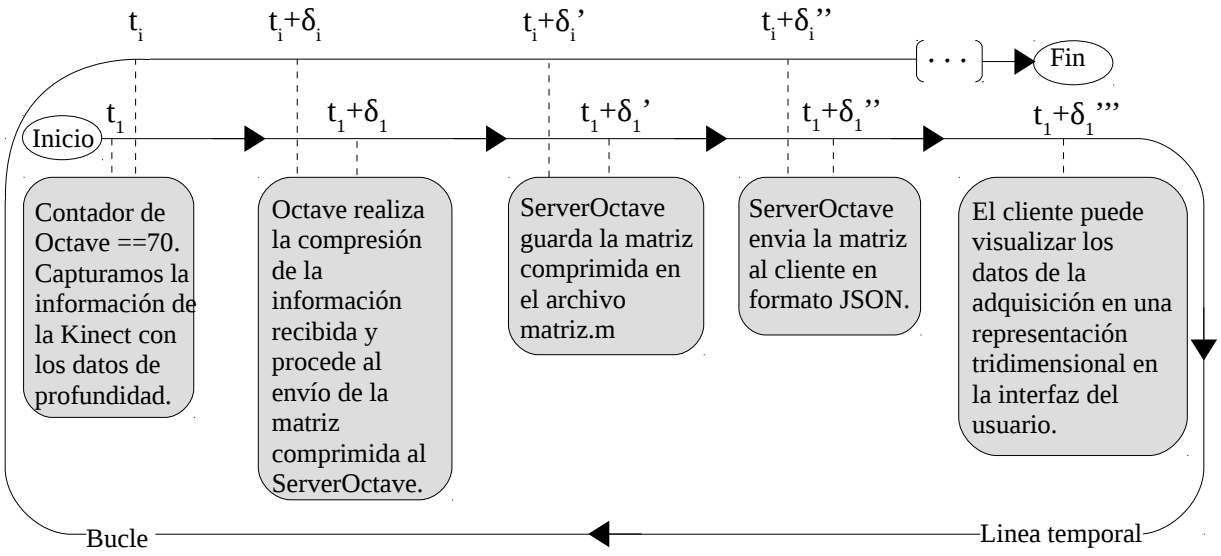


Figura 24: Esquema II Lazo Kinect.

El orden de magnitud del parámetro δ^i depende del proceso que se esté llevando a cabo. Mientras que δ_1, correspondiente al retardo que introduce Octave, puede situarse en 2 o 3 segundos (periodo en el que Octave realiza la compresión y procede al envío de los datos), δ_1' y δ_1'' se sitúa en el rango, como mucho, de una o dos décimas de segundo. Por su parte, δ_1''' depende, en mayor medida, del tiempo de retardo que provoca el envío via Wi-Fi que del retardo introducido por el propio código en el lado del cliente. En cualquier caso, hay que tener en cuenta que el servidor envía una matriz cada 30 segundos, aproximadamente, y por tanto, la acumulación de los retardos está muy lejos de provocar problemas en el flujo de datos.

6.2.- Lazos Webcams

Las dos webcam, como hemos explicado anteriormente, tienen funcionalidades distintas. La webcam frontal se encarga de realizar un streaming de la visión delantera del robot, mientras que la webcam que apunta al techo realiza las adquisiciones de fotografías en formato "*png*", que posteriormente podrían utilizarse para la detección de códigos de localización como se aplican en el trabajo [1].

6.2.1 Lazo WebCam streaming

Este lazo es muy sencillo, ya que sólo se encarga de generar un vídeo que será reproducido en el lado del cliente, para que el usuario tenga una vista subjetiva del entorno del robot desde la cámara frontal. Con esta información se podrá decidir cuales serán los próximos movimientos.

En el siguiente esquema podemos visualizar las interconexiones hardware presentes entre los distintos elementos pare este lazo.

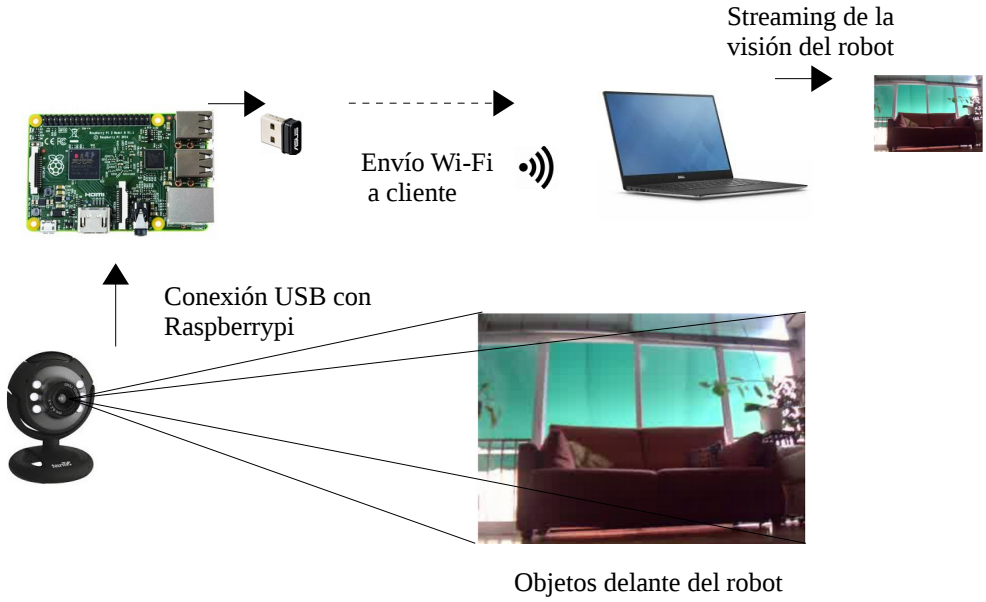


Figura 25: Esquema I Lazo WebCam Streaming

El software encargado de la comunicación con la webcam, a la que llamaremos webcam1, es FFmpeg, se trata de software libre. Es un *framework* multimedia capaz de codificar y decodificar imágenes y audio de diferentes dispositivos [17]. Para ello creamos el Servidor **ServerCam1** encargado de recibir las imágenes de la cámara webcam1. El servidor **ServerCam1** se encuentra a la escucha de las imágenes provenientes de FFmpeg, que a su vez está adquiriendo constantemente la información percibida por la Webcam1. Cada vez que el servidor ServerCam1 recibe una imagen de FFmpeg, será remitida al cliente vía Wi-Fi para su *streaming* en el ordenador remoto.

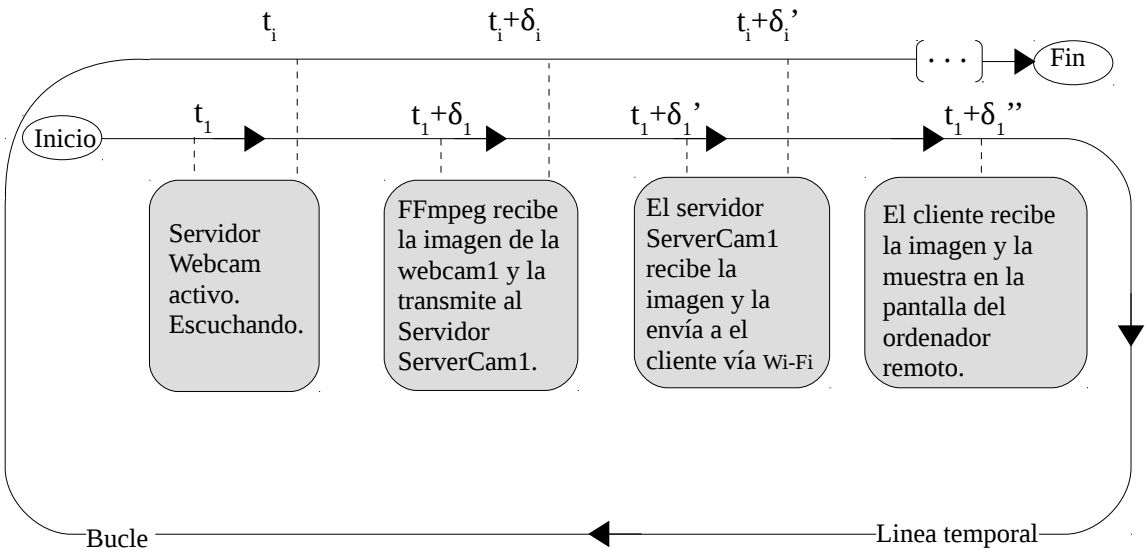


Figura 26: Esquema II Lazo WebCam Streaming.

6.2.2 Lazo Webcam techo

En este caso FFmpeg también se encarga de adquirir las imágenes de la webcam que está apuntando al techo. A esta cámara la llamaremos webcam2. Esta vez, las imágenes que recibimos de webcam2 no serán enviadas al cliente sino que el ServerCam2 las guarda en el disco cada 10 segundos.

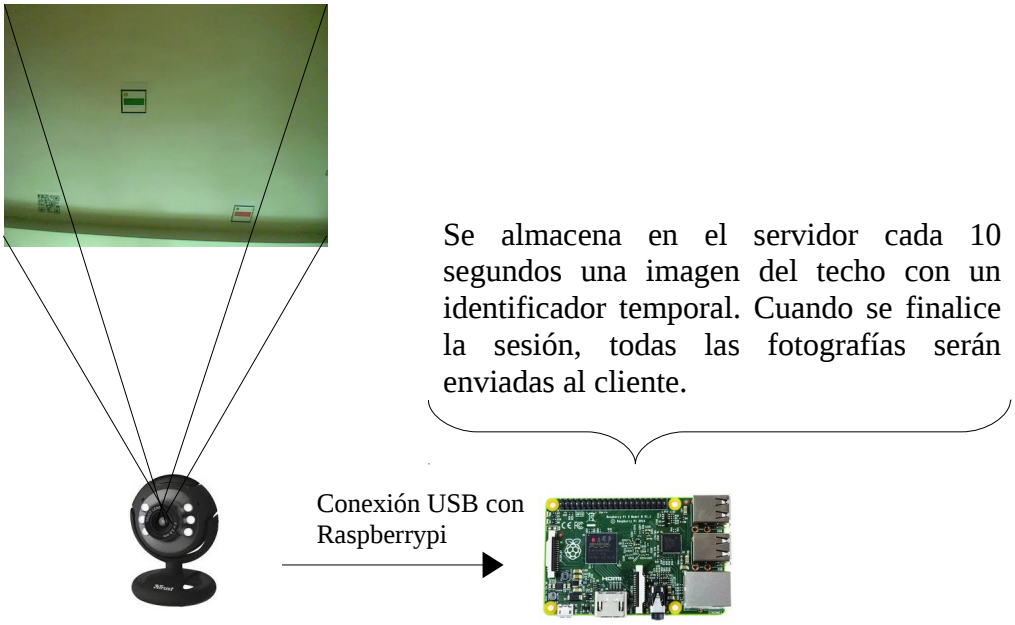


Figura 27: Esquema Lazo Webcam techo.

6.3.- Lazo Arduino

Arduino se encarga de gestionar, en primera instancia, los datos de los sensores IR (sensores de presencia) y los encoder. También es el encargado de enviar las señales que permiten el movimiento de los motores. Como necesitamos enviar y recibir mensajes, tiene que existir un canal bidireccional, que permita recaudar los datos y realizar las acciones que imponga el usuario.

Los datos de los sensores son enviados a Raspberry Pi, donde son guardados en el servidor. Las ordenes que se reciban del cliente en la Raspberry Pi son remitidas a Arduino para el control de los motores.

En el siguiente esquema podemos apreciar las interconexiones de los elementos involucrados en este lazo:

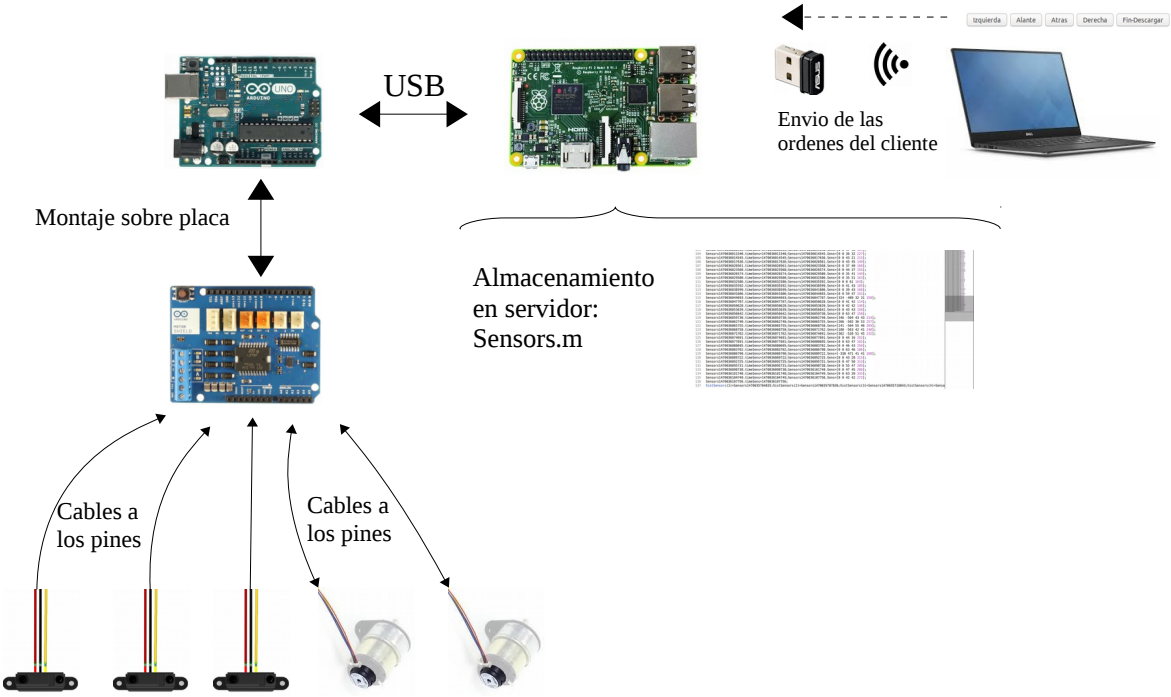


Figura 28: Esquema I Lazo Arduino

A continuación mostramos un esquema con el diagrama de flujo que muestra las distintas acciones que se llevan acabo en este lazo. Los datos de los sensores son guardados cada 2 segundos aproximadamente, sin embargo, el tiempo en recibir las órdenes por parte del cliente, depende de la frecuencia con la que éste, el cliente, presione los botones para gobernar el robot:

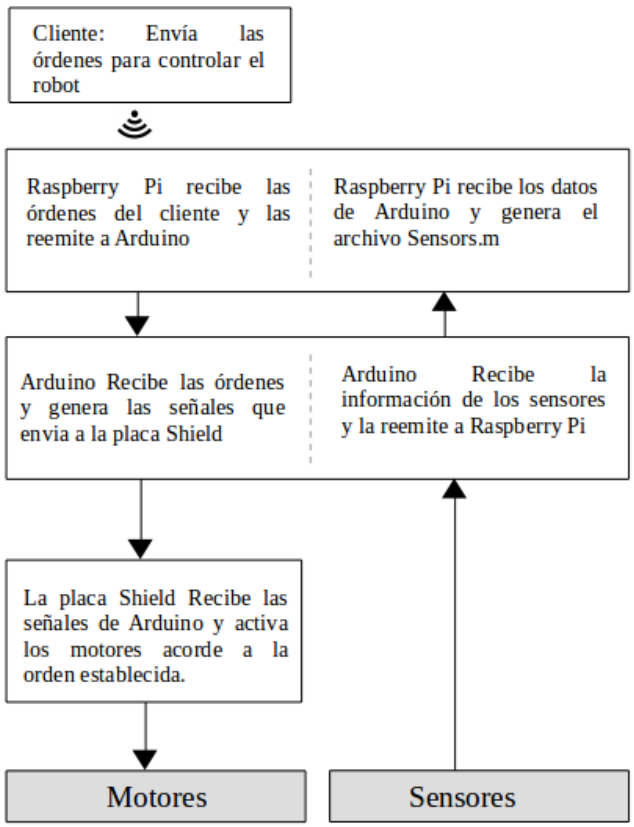


Figura 29: Esquema II Lazo Arduino

7.- Herramientas para la Construcción de Mapas

Sólo con el sensor principal es muy complicado generar mapas tridimensionales. Para solucionar este problema dotamos al robot de distintos dispositivos, descritos en la sección 4 (Componentes). Éstos dispositivos nos proporcionan la información necesaria para posicionar en el espacio los datos de profundidad de manera coherente. Una vez relacionada la posición y la orientación del robot con cada adquisición de los datos de profundidad, se puede empezar a trabajar con éstos para la generación de los mapas citados.

En [18] se propone un algoritmo para poder posicionar las adquisiciones del sensor principal correctamente en el espacio, en función de las diferencias que presentan tales adquisiciones con respecto a las contiguas, evitando los errores de deriva.

Como hemos mencionado en la introducción, la plataforma construida tiene como fin ser, en sí misma, una herramienta para la realización de diferentes prácticas, que pueden ser llevadas a cabo por distintos usuarios. Por ello es necesario dotar al robot con la mayor cantidad posible de sensores que caractericen el medio, de tal forma que posteriormente cada usuario pueda decidir cómo tratar los datos adquiridos y qué modelos aplicar para la generación de los mapas tridimensionales.

En la siguiente tabla podemos ver la clasificación de los sensores según estén destinados para la localización del robot, la adquisición de datos de profundidad o el control remoto:

Destinado a	**Elemento**
Localización	Cámara Trust apuntando al techo con las que se realizan capturas cada 10 segundos. Encoders incrementales en las ruedas traseras del robot.
Adquisición de datos de profundidad.	Sensor principal → Cámara Kinect Sensores secundarios → IR presencia
Control remoto del robot	Cámara Trust frontal con la que realizamos un vídeo en streaming para el control directo del robot por parte del usuario.

Tabla 6: Clasificación Sensores.

7.1.- Localización

Como hemos comentado antes, cada usuario puede decidir qué método implementar para localizar al robot con mayor o menor exactitud. El robot está dotado, como puede verse en la clasificación de la tabla 6 (Clasificación Sensores), de una cámara que apunta al techo, destinada a la percepción de marcadores de localización absolutos como pueden ser los códigos basados en colores o los códigos QR.

No obstante, también se pueden realizar localizaciones del robot mediante el modelo diferencial, gracias a la información que nos proporciona los encoders (localización relativa). Por supuesto, también se puede realizar fusión multisensorial. Para ello se utiliza el canal más preciso en cada momento o una combinación de los varios canales para obtener una posición fidedigna de nuestro robot.

En el artículo [19] se realiza una disertación muy interesante sobre la corrección de los errores derivados del uso de localización relativa. Localización que se obtiene mediante la evaluación de la información de los encoder, giroscopios, acelerómetros, etc. En nuestro caso solo disponemos de encoders, para realizar la localización relativa.

Precisamente en ese artículo se menciona el uso de la técnica del filtro de Kalman, como método para corregir los errores acumulativos de las técnicas de posicionamiento relativo. Método que utiliza Juan Ignacio Forcén Carvalho en el desarrollo de su trabajo de investigación “Navegación de un robot autónomo utilizando balizas y luz estructurada” [1], con excelentes resultados.

Siendo nuestro objetivo desarrollar una herramienta que nos permita aplicar diferentes métodos de localización y mapeo, no profundizaremos demasiado en los métodos de localización que no se utilicen para la elaboración de los resultados experimentales expuestos en el apartado RESULTADOS.

7.1.1 Modelo diferencial

Los robots terrestres cuyos ejes de giro permanezcan fijos presentan todas sus ruedas en planos paralelos unos respecto a otros. Los giros y cambios de trayectorias solo podrán efectuarse por movimientos a diferentes velocidades de unas ruedas respecto a las otras. Tengamos en cuenta que las ruedas de cada lado siempre giraran a la misma velocidad, es decir, las ruedas derechas presentan una velocidad angular sincronizada entre ellas, del mimo modo, las ruedas izquierdas se moverán coordinadamente. Sin embargo, los

deslizamientos sobre el terreno en los giros, son inevitables, ya que, al disponer de dos ejes paralelos, las ruedas no son tangentes al centro instantáneo de rotación. Es necesario realizar un ajuste experimental a la hora de trabajar con el modelo diferencial, en cuanto al cálculo de los desplazamientos según los datos de los encoders, ya que el deslizamiento depende de factores como el coeficiente de rozamiento del suelo con la rueda y éste, a su vez, dependerá de factores ambientales como la humedad y temperatura, factores que serán diferentes en cada experimentación.

La reconstrucción de la trayectoria se puede realizar por acumulación de los datos de los encoder. Una vez finalizada la experimentación, se descargarán todos los datos, pudiéndose reconstruir las trayectorias seguidas por el robot mediante la aplicación del método diferencial del movimiento.

Para poder reconstruir la trayectoria a partir de los datos almacenados de los encoders deberemos exponer algunos conceptos teóricos referidos al modelo diferencial.

En primer lugar, tenemos que saber cual es la relación entre el desplazamiento lineal de una rueda y el número de cuentas que nos proporciona el encoder. Como estamos trabajando con un ecoder de 6000 cuentas por revolución, y el diámetro de la rueda "Skid steer" es de 124 mm, esta relación será la siguiente:

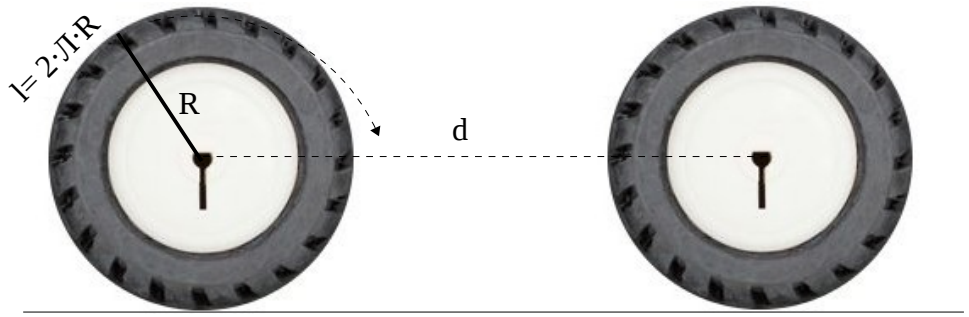


Figura 30: Representación giro ruedas *"Skid steer"*.

$$d_i = \frac{l \cdot N_i}{N_T} = \frac{2 \cdot \pi \cdot R \cdot N_i}{N_T},$$

donde d_i es el desplazamiento longitudinal del centro de la rueda, l es la longitud perimetral de la rueda, R es el radio de la rueda $(R = 62\,mm)$, N_i es el número de

cuentas del encoder (en la adquisición i) y N_T es el número de cuentas totales del encoder con el que estamos trabajando $(N_T = 6000)$.

La distancia d_i no se corresponde con el movimiento del centro de masas del robot sino con el desplazamiento de una de las ruedas. Para calcular el desplazamiento del centro de masas del robot, deberemos seguir el siguiente razonamiento, en el que realizamos la simplificación para un solo eje, ya que en primera aproximación los dos ejes pueden ser sustituidos por un solo eje que no desliza, que pasa por el centro geométrico y cuya distancia entre las ruedas se determina experimentalmente.

I → Rueda izquierda.

D → Rueda derecha.

CIR → Centro instantáneo de rotación.

L → Distancia entre ruedas, longitud del eje.

r → Distancia desde el centro de la rueda derecha hasta el CIR

β → ángulo girado, se corresponde con $-\theta$ de la referencia que escogeremos ($\beta = -\theta$).

Δd_I → Desplazamiento lineal centro rueda izquierda.

Δd_c → Desplazamiento lineal centro rueda derecha.

Δd_D → Desplazamiento lineal centro geométrico del sistema

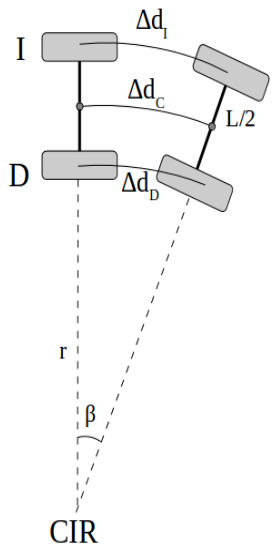


Figura 31: Esquema I geometría del giro.

El desplazamiento lineal que sufre el centro de la rueda izquierda (I) y derecha (D) serán los siguientes:

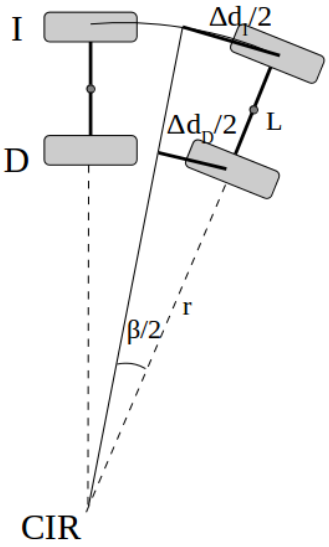


Figura 32: Esquema II geometría del giro.

$$\sin\left(\frac{\beta}{2}\right)=\frac{(\Delta d_I/2)}{(r+L)} \quad \rightarrow \quad (r+L)=\frac{(\Delta d_I/2)}{\sin\left(\frac{\beta}{2}\right)} \quad [e1]$$

$$\sin\left(\frac{\beta}{2}\right)=\frac{(\Delta d_D/2)}{(r)} \quad \rightarrow \quad (r)=\frac{(\Delta d_D/2)}{\sin\left(\frac{\beta}{2}\right)} \quad [e2]$$

Si a [e2] le sustraemos [e1] obtendremos lo siguiente:

$$L=\left(\frac{\Delta d_I}{2}-\frac{\Delta d_D}{2}\right)\cdot\frac{1}{\sin\left(\frac{\beta}{2}\right)} \quad \rightarrow \quad \sin\left(\frac{\beta}{2}\right)=\left(\frac{\Delta d_I}{2}-\frac{\Delta d_D}{2}\right)\cdot\frac{1}{L} \quad [e3]$$

Efectuamos el desarrollo en serie de Taylor de [e3], quedándonos:

$$\sin\left(\frac{\beta}{2}\right)\approx\frac{\beta}{2}-\frac{(\beta/2)^3}{3!}+\frac{(\beta/2)^5}{5!}+\ldots \quad ;$$

Trabajando con ángulos pequeños podemos despreciar los términos de orden superior a uno. El error que cometemos introduciendo esta simplificación es menor del 1% para ángulos menores de 14 grados sexagesimales. Según esto, nos quedará:

$$\sin\left(\frac{\beta}{2}\right) = \frac{\beta}{2} = \left(\frac{\Delta d_I}{2} - \frac{\Delta d_D}{2}\right)\cdot\frac{1}{L} \rightarrow \beta = \frac{\left(\Delta d_I - \Delta d_D\right)}{L}$$

Como hemos comentado antes $\beta = -\theta$. Esta decisión ha sido tomada para que los giros hacia la derecha nos proporcionen ángulos negativos y viceversa.

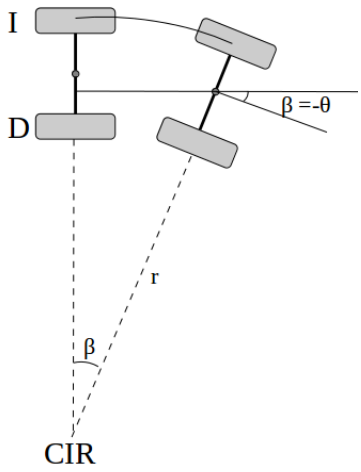


Figura 33: Esquema III geometría del giro.

$$\beta = \frac{\left(\Delta d_I - \Delta d_D\right)}{L} \quad [e4] \rightarrow \theta = \frac{\left(\Delta d_D - \Delta d_I\right)}{L} \quad [e5]$$

Una vez calculado el ángulo de la orientación que adopta nuestro sistema, en función de los desplazamientos lineales que sufre cada una de las ruedas, podemos deducir el valor que tendrá el desplazamiento del centro geométrico de nuestro sistema de tracción. Para ello, usaremos las expresiones [e1], [e2] (teniendo en cuenta la consideración $\sin x \approx x$) y [e4]. El desplazamiento lineal que sufre el centro geométrico de nuestro sistema que será deducido utilizando las mismas consideraciones geométricas que en el caso del desplazamiento de I y de D:

$$\Delta d_c = \beta\cdot\left(r + \frac{L}{2}\right) \quad [e6]$$

de [e1] despejamos L:

$$L = \left(\frac{\Delta d_I}{\beta} - r\right) \quad [e7]$$

introducimos [e2] en [e7]:

$$L=\left(\frac{\Delta d_I}{\beta}-\frac{\Delta d_I}{\beta}\right) \quad \text{[e8]}$$

Introduciendo [e8] en [e6], llegamos al desplazamiento Δd_c buscado:

$$\Delta d_c=\left(\frac{\Delta d_D+\Delta d_I}{2}\right) \quad \text{[e9]}$$

Podemos ahora, gracias a las deducciones anteriores, ofrecer las siguientes expresiones recursivas para calcular los valores de las posiciones x_i, y_i y θ_i, correspondientes la situación centro geométrico de nuestro sistema motor, y a su orientación θ.

$$x_i=x_{i-1}+\Delta d_{c,i}\cdot\cos(\theta_i)$$

$$y_i=y_{i-1}+\Delta d_{c,i}\cdot\sin(\theta_i)$$

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

En [20], [21] y [22] podemos diferentes razonamientos que nos conducen a las ecuaciones anteriores.

7.1.2 Localización mediante visión

Como hemos comentado anteriormente, en el trabajo de investigación de Juan Ignacio Forcén [1] se aplica fusión sensorial mediante la unión de localización relativa (considerando los datos de los encoders) y localización por visión (considerando las fotografías del techo). En el caso de la localización por visión el análisis de las

fotografías del techo se basa en visión artificial.

En el trabajo mencionado se utiliza la librería de marcadores ArUco. En [23] podemos encontrar una descripción detallada de las funcionalidades de estas librerías así como ejemplos para su implementación.

En nuestro caso hemos dotado al robot con una cámara que apunta al techo para la detección de los posibles marcadores. No nos extendernos mucho en este punto ya que no hemos utilizado dichas fotografías para la elaboración de los resultados. No obstante, sí que podemos decir que las fotografías que se van realizado gracias a esta cámara serán designadas con un identificador temporal (igual que los datos del encoder y las matrices de profundidad de la Kinect). Una vez descargados todos los datos se podrán realizar comparaciones temporales para asociar el momento de la adquisición con el resto de datos (los encoders y las matrices del dispositivo Kinect)

8.- Control Remoto-Interfaz Gráfica

En esta sección describiremos la interfaz gráfica a disposición del usuario, con la que se puede controlar los movimientos del robot de manera remota. Gracias a la interfaz gráfica se puede visualizar la parte frontal del robot y los datos de profundidad representados de manera tridimensional. Por tanto, se visualiza un streaming de la cámara web delantera (webcam1) y una representación tridimensional de los datos adquiridos por la Kinect.

También tendremos unos botones para poder controlar el movimiento del robot, y así posicionarlo en los lugares adecuados para realizar las capturas que nos interesen. Además dispondremos de un botón para finalizar la sesión y recaudar los todos los datos de la experiencia: el script con las matrices de profundidad, el script con los valores de los sensores conectados a Arduino, las fotografías del techo que se han ido realizando y el script con el histórico de las fotografías del techo.

La librería que utilizamos para poder efectuar la representación tridimensional es “plotly.js”. Pueden encontrarse información sobre su utilización y aplicaciones en [24]. Es necesario transformar en formato JSON los datos que enviamos al cliente para su correcta interpretación por las funciones de la citada librería.

El aspecto de la interfaz en el navegador es el siguiente:

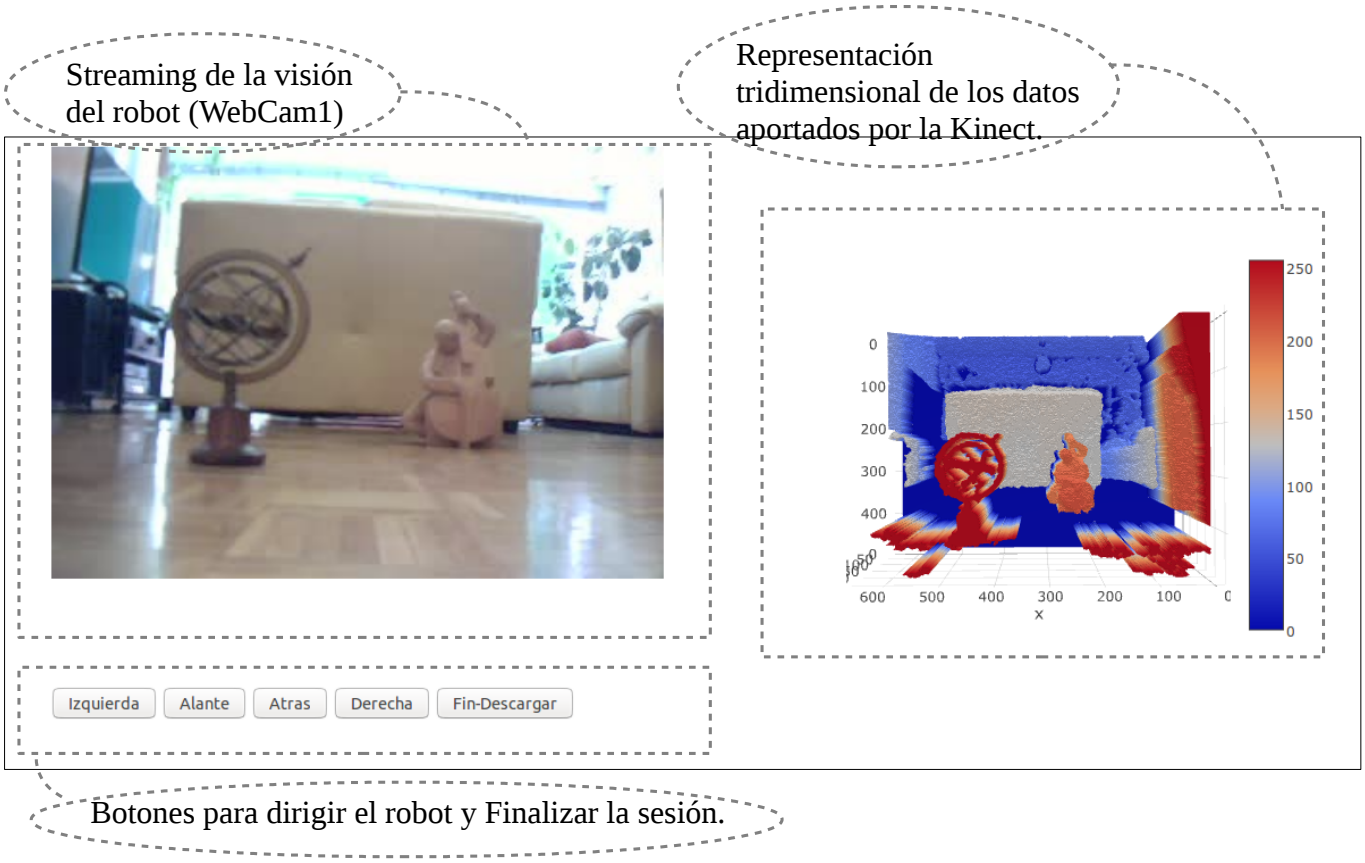


Figura 34: Interfaz gráfica 1.

Como puede apreciarse en la imagen anterior, en el navegador podemos visualizar tanto un streaming con las imágenes que recibimos de la webcam delantera, como una representación tridimensional de los datos de profundidad que se reciben del dispositivo Kinect. Los datos de profundidad son recibidos cada 30 segundos aproximadamente, ya que esta representación no es imprescindible para desarrollar la experiencia de adquisición, pero sí que nos permite comprobar que el dispositivo Kinect, efectivamente, está recibiendo los datos que deseamos. La webcam delantera sí que nos muestra una imagen en directo de la parte frontal del robot.

También podemos apreciar, como hemos comentado anteriormente, un panel de control en el cual disponemos de varios botones con los que podremos gobernar el movimiento del robot, así como un último botón con el que indicaremos al servidor que deseamos finalizar la adquisición, y gracias al cual recibiremos en el ordenador remoto los datos, guardados en el servidor durante la experiencia.

La representación de las imágenes tridimensionales se hace en base a la matriz

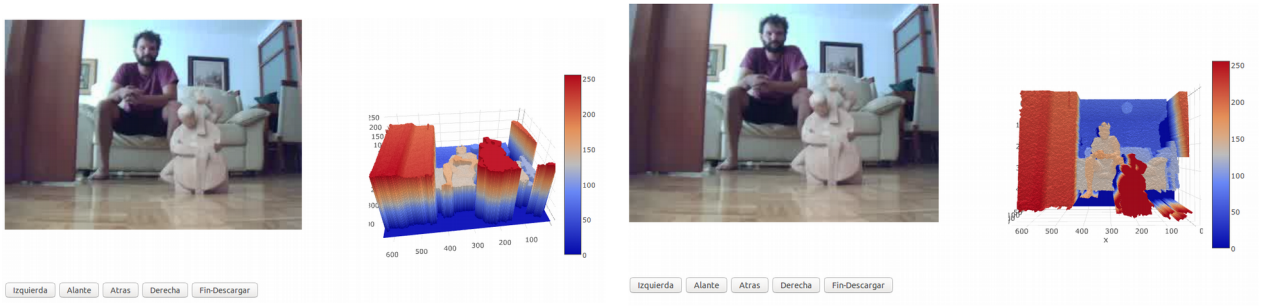
comprimida ya que las distancias relativas se mantienen, no correspondiéndose con las distancias reales hasta que no se realiza el proceso de descompresión. Evidentemente, debemos tratar de no saturar el canal de transmisión y por tanto el envío de estos datos se realizarán con la comprimida ya que ocupa 8 veces menos que la matriz de los datos reales, como hemos explicado en la sección 6.1.1 (Tratamiento de la información recibida por el dispositivo Kinect).

En la siguiente imagen podemos visualizar diferentes figuras que aparecen representadas tridimensionalmente en el lado del cliente:

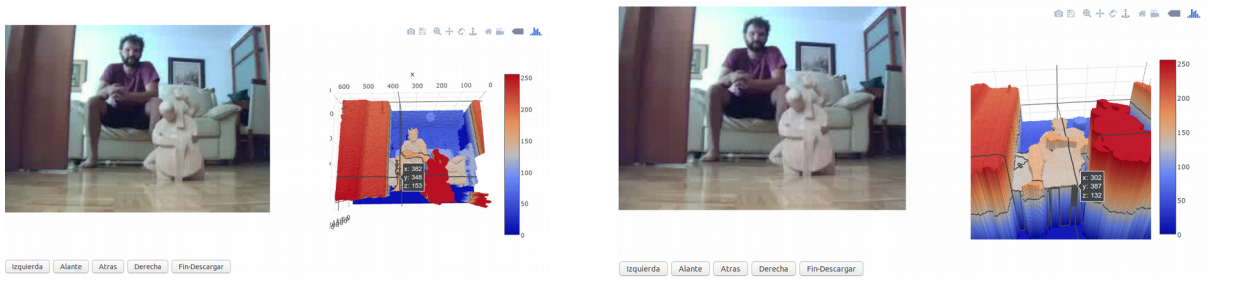


Figura 35: Interfaz gráfica 2.

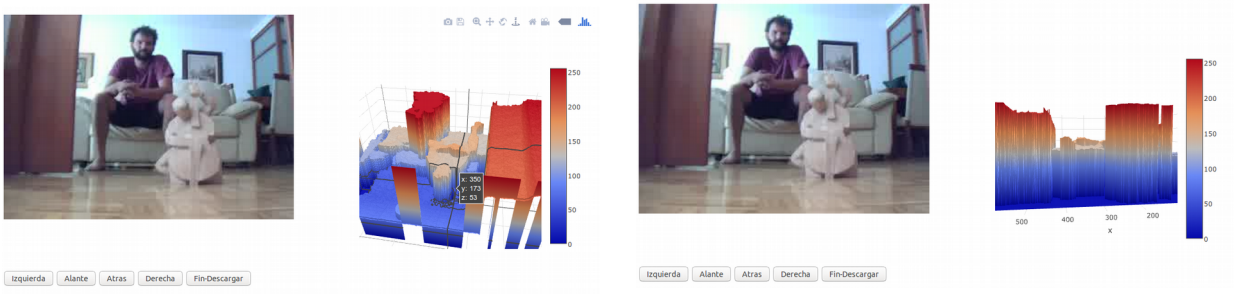
En la siguiente secuencia de imágenes podemos apreciar cómo se puede interactuar con la gráfica tridimensional para obtener diferentes perspectivas, así como acercarse u obtener los valores de distancia de diferentes puntos. Recuérdese que estos valores provienen de la matriz comprimida y, por tanto, no se corresponden con los valores reales.



a) b)



c) d)



e) f)

Figura 36: Interfaz gráfica 3: En a) se puede apreciar la primera representación con la que podemos actuar, en b) hemos rotado la imagen para poder verla desde frente, en c) situamos el puntero para obtener el valor de algún punto, en d) hacemos zoom para acercarnos, en e) rotamos la gráfica para visualizarla desde arriba, en f) seguimos rotando la gráfica para ver el perfil.

9.- Tratamiento de la Información Adquirida

La forma de almacenar la información adquirida por los distintos sensores de los que dispone el robot es de suma importancia, ya que, una vez finalizada la sesión, deben poder recogerse los datos de manera ordenada para su posterior análisis. Para tal efecto, nos decantamos por asignarle un identificador temporal a cada dato recibido. Gracias a estos identificadores temporales podemos realizar comparaciones entre los datos situarlos en el momento concreto de su adquisición.

Utilizando estos identificadores temporales podremos realizar una simulación del recorrido y generar la trayectoria que adopta el robot en el espacio en el momento de la adquisición de los datos de profundidad y de las fotografías de la cámara web superior con la que podremos realizar los posicionamientos en base a los códigos QR o códigos de color.

9.1.- Almacenamiento de la Información

Cada vez que Octave realiza una adquisición de datos de profundidad, éstos son enviados al servidor **ServerOctave** para ser guardados en la carpeta del servidor en un único archivo denominado matriz.m. Serán guardados en forma de *struct* con dos campos. El primer campo será la matriz con los datos de profundidad. Y el segundo campo albergará el tiempo en milisegundos que tiene su inicio a las 0:00 del 1 de enero de 1970. Con esto conseguiremos el identificador temporal del que hemos hablado. Este identificador es unívoco y nos permitirá trabajar los datos de cada experiencia sin peligro de ambigüedad. Este documento también dispondrá de una variable, denominada histMatrProf, que guardará el historial de todos los datos para su comparación con los datos del encoder o las fotografías del techo.

A continuación mostramos una captura de pantalla en la que se abrió el archivo matriz.m para su visualización:

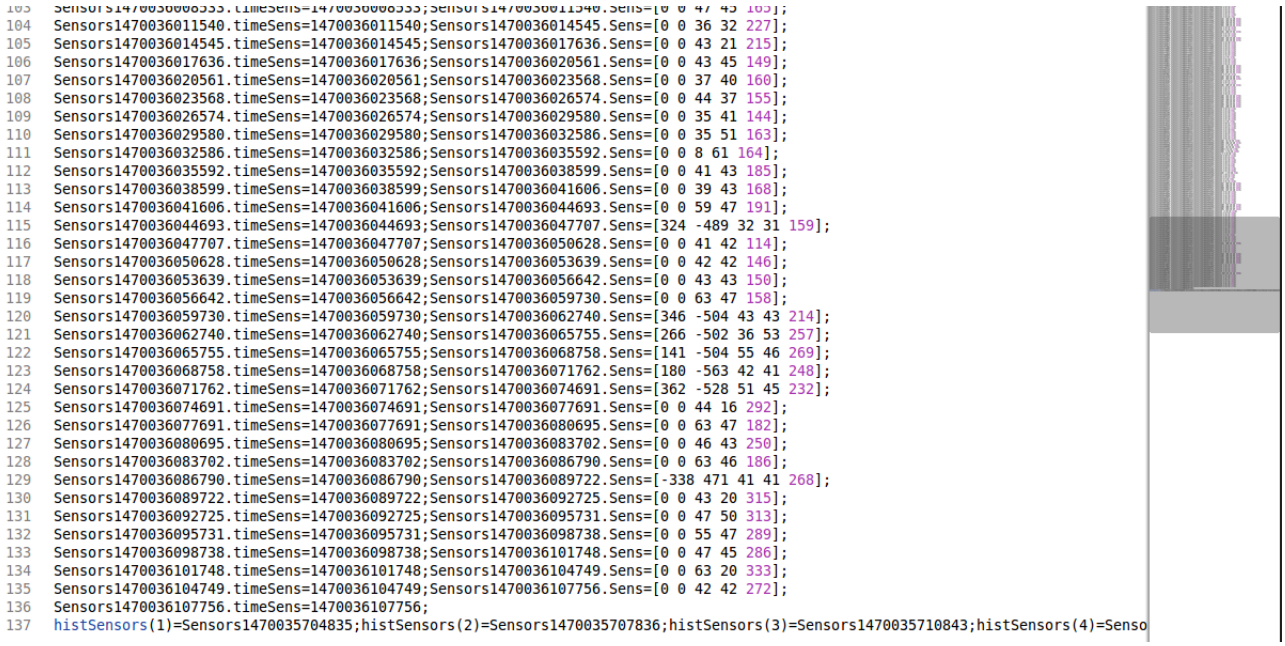
```
MatProf1470035715302.mp=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 97 97 97 97 97 98 98 98 98 97 97 98 98 97 97 98 98
MatProf1470035715302.timeMP=1470035715302;
MatProf1470035768196.mp=[0 0 0 0 0 0 0 0 157 157 156 157 157 157 156 157 157 157 158 158 158 158 158 158 158 158 158 157 157 157
MatProf1470035768196.timeMP=1470035768196;
MatProf1470035819725.mp=[0 0 0 0 0 0 0 0 136 136 136 136 138 138 138 138 138 138 139 139 139 139 139 139 139 139 139 139 139
MatProf1470035819725.timeMP=1470035819725;
MatProf1470035871941.mp=[0 0 0 0 0 0 0 0 135 135 135 135 137 135 137 136 137 137 137 137 137 137 137 137 137 137 137 137 136
MatProf1470035871941.timeMP=1470035871941;
MatProf1470035923662.mp=[0 0 0 0 0 0 0 0 118 118 118 118 118 118 118 117 117 117 117 116 116 115 116 115 115 115 115 114 114 114
MatProf1470035923662.timeMP=1470035923662;
MatProf1470035974709.mp=[0 0 0 0 0 0 0 0 226 227 227 228 228 234 235 234 233 228 227 225 226 226 226 225 225 225 225 226 225 225
MatProf1470035974709.timeMP=1470035974709;
MatProf1470036025861.mp=[0 0 0 0 0 0 0 0 221 221 221 222 221 221 222 222 222 222 222 222 223 223 223 223 224 223 224 224 224 224
MatProf1470036025861.timeMP=1470036025861;
MatProf1470036078567.mp=[0 0 0 0 0 0 0 0 100 102 102 103 105 104 104 103 101 99 99 99 99 98 98 98 99 98 98 98 98 98 98 98 97 9
MatProf1470036078567.timeMP=1470036078567;

histMatrProf(1) =MatProf1470035715302;histMatrProf(2) =MatProf1470035768196;histMatrProf(3) =MatProf1470035819725;histMatrProf(4)

```

Figura 37: Visualización del archivo matriz.m

Los datos de los encoder, sensores de distancia y fotografías del techo también recibirán un identificador temporal, que permitirá unificar todos ellos de manera coherente. Es decir, los datos que se reciben de Arduino en el servidor **ServerArduino**, también serán guardados en un archivo denominado Sensors.m y dispondrán la misma estructura que los datos del archivo matriz.m.



```
Sensors1470036011540.timeSens=1470036011540;Sensors1470036014545.Sens=[0 0 36 32 227];
Sensors1470036014545.timeSens=1470036014545;Sensors1470036017636.Sens=[0 0 43 21 215];
Sensors1470036017636.timeSens=1470036017636;Sensors1470036020561.Sens=[0 0 43 45 149];
Sensors1470036020561.timeSens=1470036020561;Sensors1470036023568.Sens=[0 0 37 40 160];
Sensors1470036023568.timeSens=1470036023568;Sensors1470036026574.Sens=[0 0 44 37 155];
Sensors1470036026574.timeSens=1470036026574;Sensors1470036029580.Sens=[0 0 35 41 144];
Sensors1470036029580.timeSens=1470036029580;Sensors1470036032586.Sens=[0 0 35 51 163];
Sensors1470036032586.timeSens=1470036032586;Sensors1470036035592.Sens=[0 0 8 61 164];
Sensors1470036035592.timeSens=1470036035592;Sensors1470036038599.Sens=[0 0 41 43 185];
Sensors1470036038599.timeSens=1470036038599;Sensors1470036041606.Sens=[0 0 39 43 168];
Sensors1470036041606.timeSens=1470036041606;Sensors1470036044693.Sens=[0 0 59 47 191];
Sensors1470036044693.timeSens=1470036044693;Sensors1470036047707.Sens=[324 -489 32 31 159];
Sensors1470036047707.timeSens=1470036047707;Sensors1470036050628.Sens=[0 0 41 42 114];
Sensors1470036050628.timeSens=1470036050628;Sensors1470036053639.Sens=[0 0 42 42 146];
Sensors1470036053639.timeSens=1470036053639;Sensors1470036056642.Sens=[0 0 43 43 150];
Sensors1470036056642.timeSens=1470036056642;Sensors1470036059730.Sens=[0 0 63 47 158];
Sensors1470036059730.timeSens=1470036059730;Sensors1470036062740.Sens=[346 -504 43 43 214];
Sensors1470036062740.timeSens=1470036062740;Sensors1470036065755.Sens=[266 -502 36 53 257];
Sensors1470036065755.timeSens=1470036065755;Sensors1470036068758.Sens=[141 -504 55 46 269];
Sensors1470036068758.timeSens=1470036068758;Sensors1470036071762.Sens=[180 -563 42 41 248];
Sensors1470036071762.timeSens=1470036071762;Sensors1470036074691.Sens=[362 -528 51 45 232];
Sensors1470036074691.timeSens=1470036074691;Sensors1470036077691.Sens=[0 0 44 16 292];
Sensors1470036077691.timeSens=1470036077691;Sensors1470036080695.Sens=[0 0 63 47 182];
Sensors1470036080695.timeSens=1470036080695;Sensors1470036083702.Sens=[0 0 46 43 250];
Sensors1470036083702.timeSens=1470036083702;Sensors1470036086790.Sens=[0 0 63 46 186];
Sensors1470036086790.timeSens=1470036086790;Sensors1470036089722.Sens=[-338 471 41 41 268];
Sensors1470036089722.timeSens=1470036089722;Sensors1470036092725.Sens=[0 0 43 20 315];
Sensors1470036092725.timeSens=1470036092725;Sensors1470036095731.Sens=[0 0 47 50 313];
Sensors1470036095731.timeSens=1470036095731;Sensors1470036098738.Sens=[0 0 55 47 289];
Sensors1470036098738.timeSens=1470036098738;Sensors1470036101748.Sens=[0 0 47 45 286];
Sensors1470036101748.timeSens=1470036101748;Sensors1470036104749.Sens=[0 0 63 20 333];
Sensors1470036104749.timeSens=1470036104749;Sensors1470036107756.Sens=[0 0 42 42 272];
Sensors1470036107756.timeSens=1470036107756;
histSensors(1)=Sensors1470035704835;histSensors(2)=Sensors1470035707836;histSensors(3)=Sensors1470035710843;histSensors(4)=Senso
```

Figura 38: Visualización del archivo Sensor.m

Durante la sesión generamos una variable *struct* con el histórico de los datos adquiridos, tanto de las matrices como de los sensores. Dichos datos son guardados al

final de cada uno de los archivos conteniendo los nombres de todos los datos almacenados. Esto nos permitirá gestionar de manera eficaz los datos guardados, y facilitará la comparativa entre unos y otros para ser puestos en común a la hora de realizar el análisis.

Una vez cargado este archivo en Octave o Matlab, podremos saber los valores del tiempo (en el cual ha sido adquirido el dato) o datos de la adquisición i-ésima mediante las siguientes sentencias:

Sentencia	Descripción
histMatrProf(1).mp →	Datos de profundidad.
histMatrProf(1).timeMP →	Valor temporal de la adquisición.
histSensors(i).Sens →	Datos de los sensores (Encoders y Sensores IR).
histSensors(i).timeSens →	Valor temporal de la adquisición.
histPhoto(i)	Nombre de la foto i-ésima

Tabla 7: Archivos para el análisis *offline*.

En el siguiente diagrama podemos apreciar la manera en la que se gestionan los datos adquiridos:

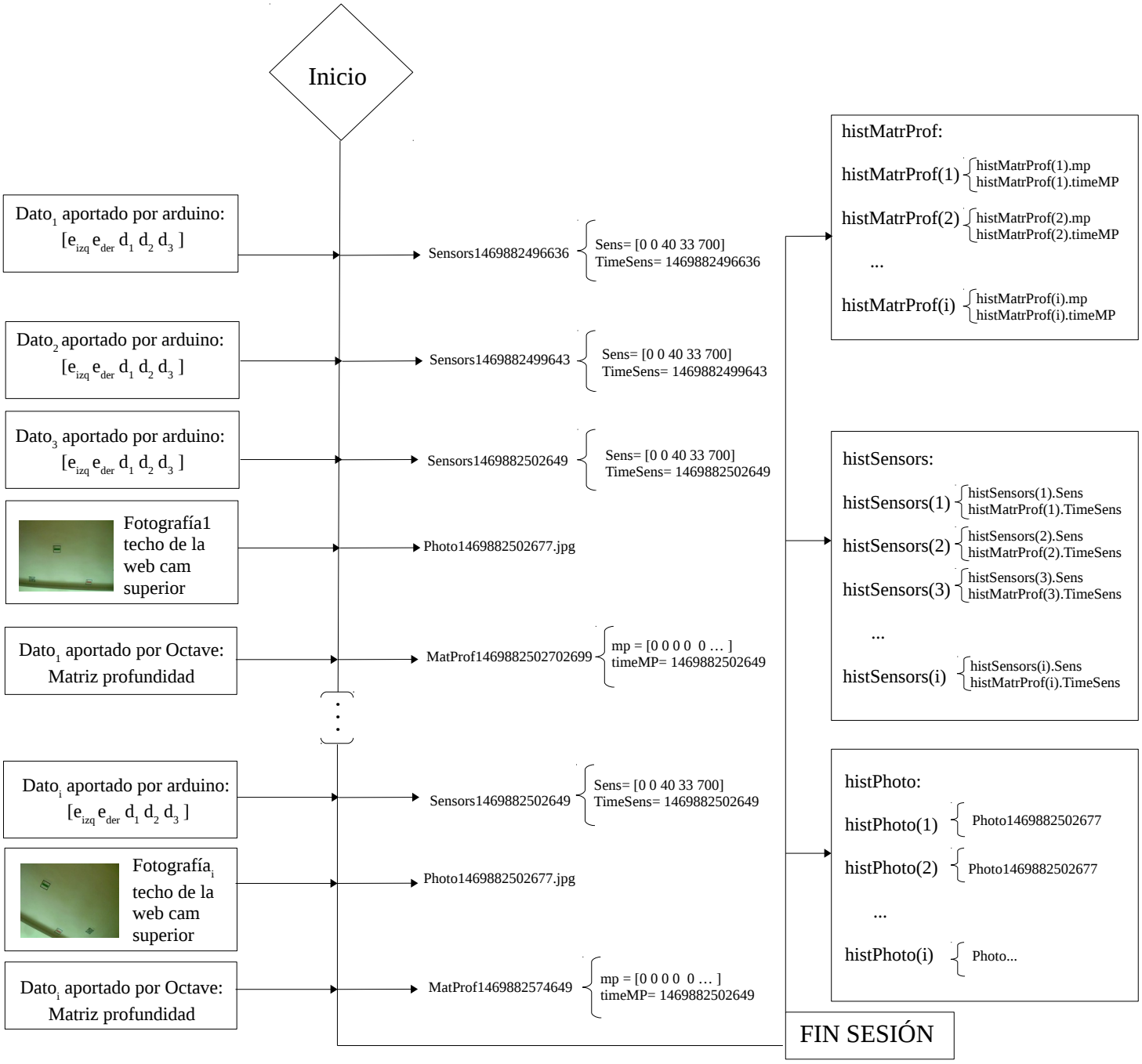


Figura 39: Flujo de adquisición.

9.2.- Construcción del Mapa 3D

Una vez almacenados todos los datos y finalizada la sesión. Procedemos a descargarnos los archivos que contienen los datos almacenados para poder realizar un estudio de los mismos. En nuestro caso utilizamos los datos del encoder para llevar a cabo un posicionamiento relativo y asociar los datos de profundidad con las posiciones que ocupaba el robot en el momento de la adquisición.

Realizamos una comparación de las asignaciones temporales entre los datos de las matrices de profundidad y los datos del encoder para calcular la posición y orientación que dispone el robot en el momento de cada una de las adquisiciones de datos de profundidad.

En el siguiente esquema podemos apreciar el modo a proceder en el análisis de datos llevados a cabo para la generación de los resultados que mostraremos en el apartado de resultados (RESULTADOS):

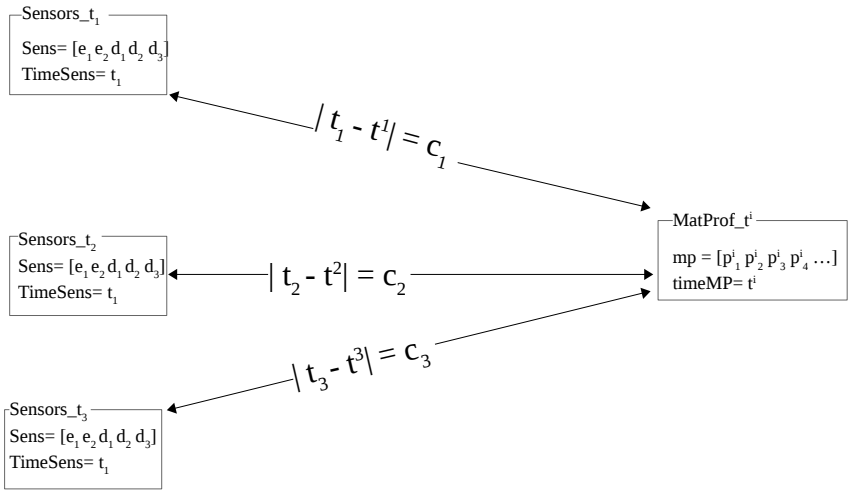


Figura 40: Comparativa temporal.

Supongamos que $c_3<c_2<c_1$, , por tanto, decidimos que la matriz de profundidad ha sido adquirida cuando el robot ha realizado los movimientos del encoder correspondientes a Sensor_t1.Sens(e1 y e2) +Sensor_t2.Sens(e1 y e2) +Sensor_t3.Sens(e1 y e2) . Una vez realizados estos movimientos el robot presenta una posición del centro geométrico respecto al punto inicial y una orientación determinada. Consideraremos que la orientación inicial es $\theta=0^o$ y que el punto inicial es $(x_i,y_i)=(250,250)$, $z_t=0\,\forall\, t$ .Teniendo en cuenta estas consideraciones y aplicando el modelo diferencial expuesto en la sección 7.1.1 (Modelo diferencial), llegamos a que la orientación y posición del robot en el momento de la adquisición es:

$$x_t = \sum_{i=1}^{n} \frac{2 \cdot \pi \cdot R \cdot N_i}{N_T} \cdot \cos(\theta_i)$$

$$y_t = \sum_{i=1}^{n} \frac{2 \cdot \pi \cdot R \cdot N_i}{N_T} \cdot \sin(\theta_i)$$

$$\theta_i = \frac{(\Delta d_{Di} - \Delta d_{Ii})}{L} + \theta_{i-1} = \frac{\left(\left(\frac{2 \cdot \pi \cdot R \cdot N_{Di}}{N_T}\right) - \left(\frac{2 \cdot \pi \cdot R \cdot N_{Ii}}{N_T}\right)\right)}{L} + \theta_{i-1}$$

Realizando esta operación iterativamente tendremos las posiciones y orientaciones en el momento de todas las adquisiciones. Información imprescindible que nos permitirá construir los mapas tridimensionales que exponemos en la sección 10 (Resultados de la Experiencia) dentro del apartado RESULTADOS.

Para situar los puntos de cada adquisición en el mapa tridimensional correctamente, debemos rotar los mismos en función de la posición y la orientación con la cual han sido adquiridos. Para ello debemos utilizar la matriz de rotación con la que colocaremos cada punto adquirido en su posición correspondiente en el espacio. En el siguiente esquema se ejemplifica la rotación de un punto $(x_{1,} y_1)$ hasta su verdadera posición situada en $(x_{2,} y_2)$ en función de un ángulo θ.

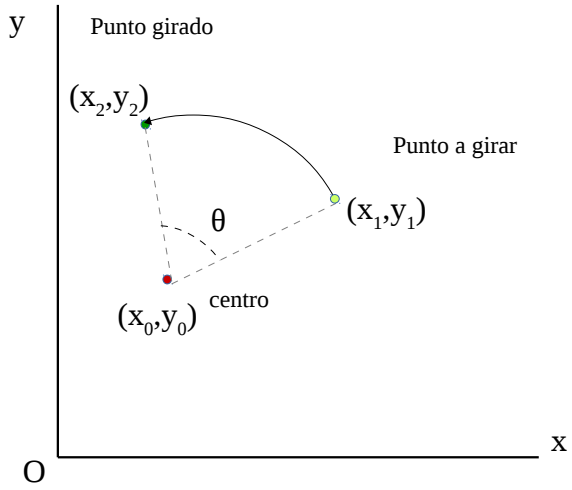


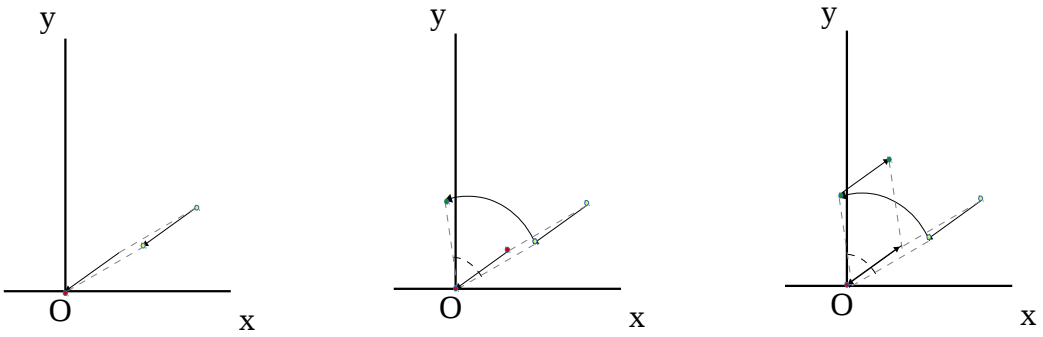
Figura 41: Esquema giro en el plano.

$$(x_2, y_2) = (x_1 - x_0, y_1 - y_0) \cdot \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} + (x_0, y_0)$$

Trasladamos el punto a rotar como si hubiese sido adquirido desde el origen.

Aplicamos la matriz de rotación al punto trasladado.

Volvemos a desplazar el punto para situarlo en su posición final.



a) Trasladamos el punto adquirido como si hubiese sido captado desde el origen.

b) Rotamos el punto trasladado con centro en el origen.

c) Deshacemos la traslación.

Figura 41: Esquema giro en el plano.

Una vez dispongamos de todas las posiciones y orientaciones tendremos que elaborar la representación tridimensional, teniendo en cuenta la posición y orientación con la que ha sido tomada cada una. En el siguiente esquema podemos visualizar cómo se aplica la matriz de rotación, según la diferencia entre la primera adquisición y la segunda adquisición:

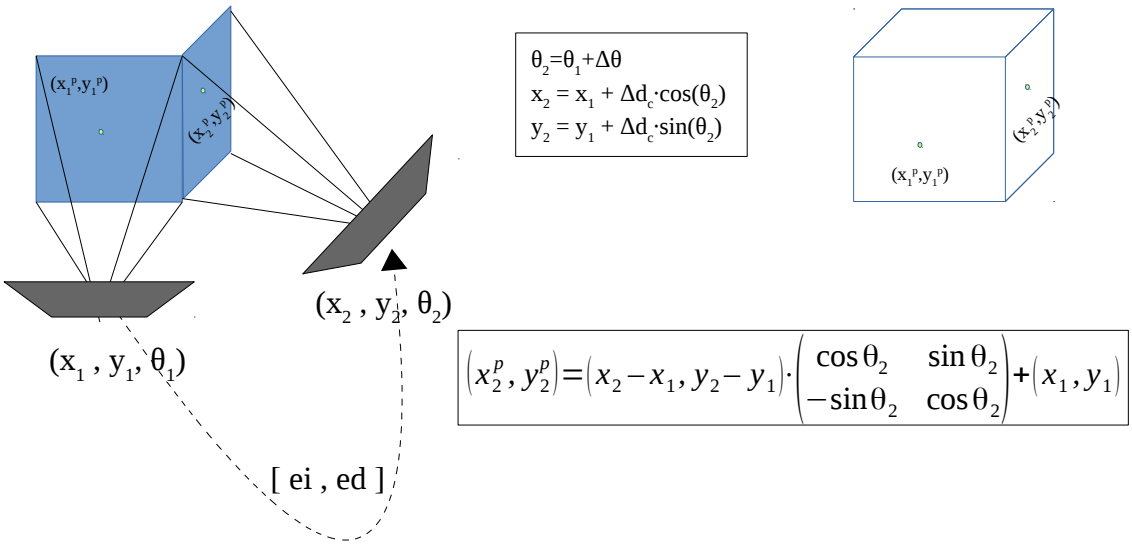


Figura 42: Esquema de giro en el espacio. [e_i , e_d] son los datos de los encoder izquierdo y derecho entre la posición (x_1 , y_1) y (x_2 , y_2).

Las modificaciones en el espacio tridimensional sólo se efectúan para las coordenadas x e y. La coordenada z es un invariante respecto a las modificaciones de giros en el espacio, ya que estas solo dependen de lo que el robot ve y no de como está orientado ni de en que punto se realizó la adquisición.

Cabe preguntarse cuál es el error en el posicionamiento a la hora de determinar la posición y la orientación en el robot. En [25] se realiza un estudio de la mejora del posicionamiento de un robot en base a la corrección de los errores sistemáticos.

Como comentamos en la sección 7.1.2 (Localización mediante visión), los datos de odometría pueden ser fusionados con medidas realizadas en base a posicionamientos absolutos para mejorar la fiabilidad de los resultados. En [26] encontramos un artículo en el que se propone un método para combinar las diferentes técnicas de navegación aplicadas a robots móviles.

RESULTADOS

10.- Resultados de la Experiencia

Mostramos en esta sección el funcionamiento de la herramienta que hemos diseñado y propondremos un ejemplo fruto de la experiencia realizada. Gracias a la adquisición de los datos y al posterior análisis de los mismos se pueden visualizar los mapas en tres dimensiones que se exponen en esta sección. La manera de almacenar la información y el tratamiento de la misma para la construcción de los mapas tridimensionales ha sido explicada en las secciones 9.1 (Almacenamiento de la Información) y 9.2 (Construcción del Mapa 3D). El modelo que utilizamos para obtener las posiciones y orientaciones del robot en el momento de cada adquisición es el modelo diferencial, explicado en la sección 7.1.1 (Modelo diferencial).

A continuación proponemos una experiencia para mostrar la funcionalidad del robot. Realizaremos una adquisición de datos en el interior de un salón. Colocamos al robot en el centro de la estancia y giramos entorno a su centro geométrico $\pi/4$ radianes tras cada adquisición. Se habrán realizado 8 adquisiciones. Mostramos a continuación varias fotografías del lugar donde se realizó la experiencia, para poder ser cotejadas con los resultados obtenidos tras el análisis. Las correspondencias de los elementos numerados se pueden encontrar en la tabla 8 (Elementos/Enseres) que situamos tras las fotografías.

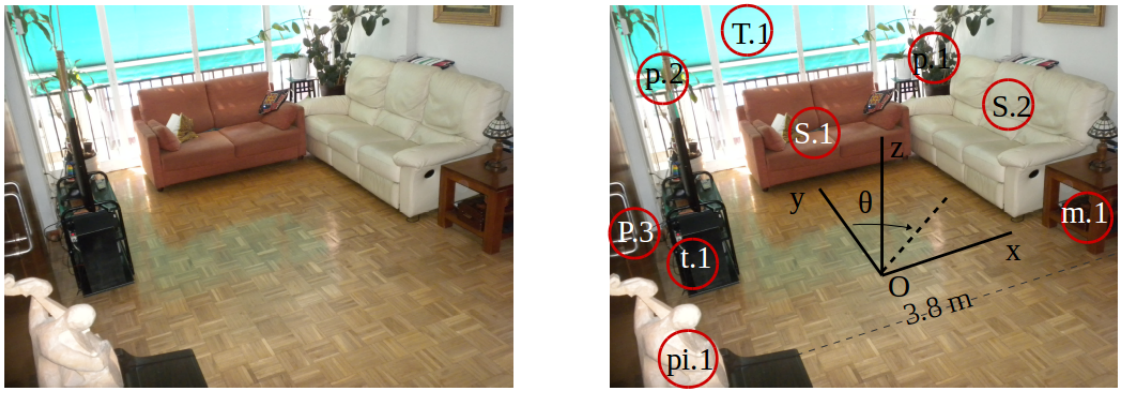


Figura 43: Fotografía del lado desde donde se inició la adquisición



Figura 44: Fotografía del lado contrario de la estancia.

En las imágenes de la derecha podemos apreciar dos fotografías más con los elementos de las fotografías anteriores pero tomados desde otra perspectiva, ya que en las fotografías principales no se aprecia con claridad la profundidad de algunos de los elementos. En concreto de los elementos t.1 y S.3.

La fotografía superior muestra S.2 y la fotografía inferior muestra t.1 con mayor claridad que en las fotografías anteriores.

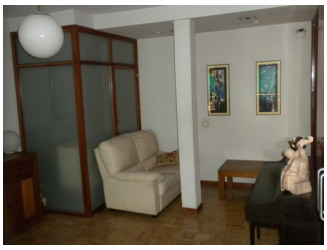




Figura 45: Fotografías estancia.

En la siguiente tabla se muestran las correspondencias con los elementos de las fotografías anteriores:

Elemento	Descripción
S.1	Sofá rojo dos plazas
p.1	Planta 1

S.2	Sofá blanco tres plazas
Elemento	**Descripción**
m.1	Mesilla
a.1	Aparador
P.1	Puerta uno
S.3	Sofá blanco dos plazas
c.1	Columna vertical
m.2	Mesa
P.2	Puerta dos
pi.1	Piano
P.3	Puerta 3
t.1	Televisor
p.2	Planta 2
T.1	Toldo exterior

Tabla 8: Elementos/Enseres.

A continuación podemos visualizar un esquema de la estancia con las medidas reales:

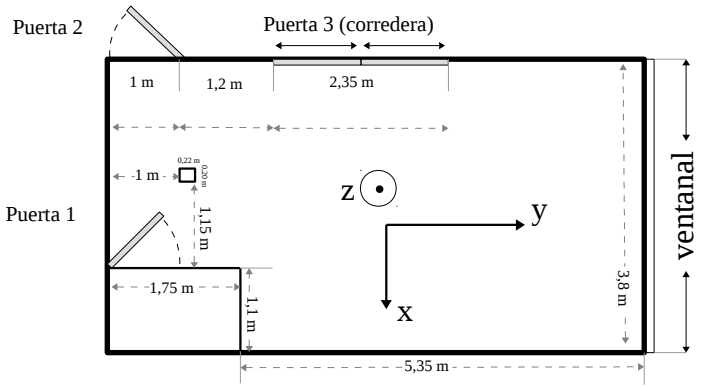


Figura 46: Esquema estancia.

Las siguientes imágenes muestran el entorno gráfico con el que el usuario controla el robot remotamente en el momento de la adquisición por parte del dispositivo Kinect:

- Primera adquisición:

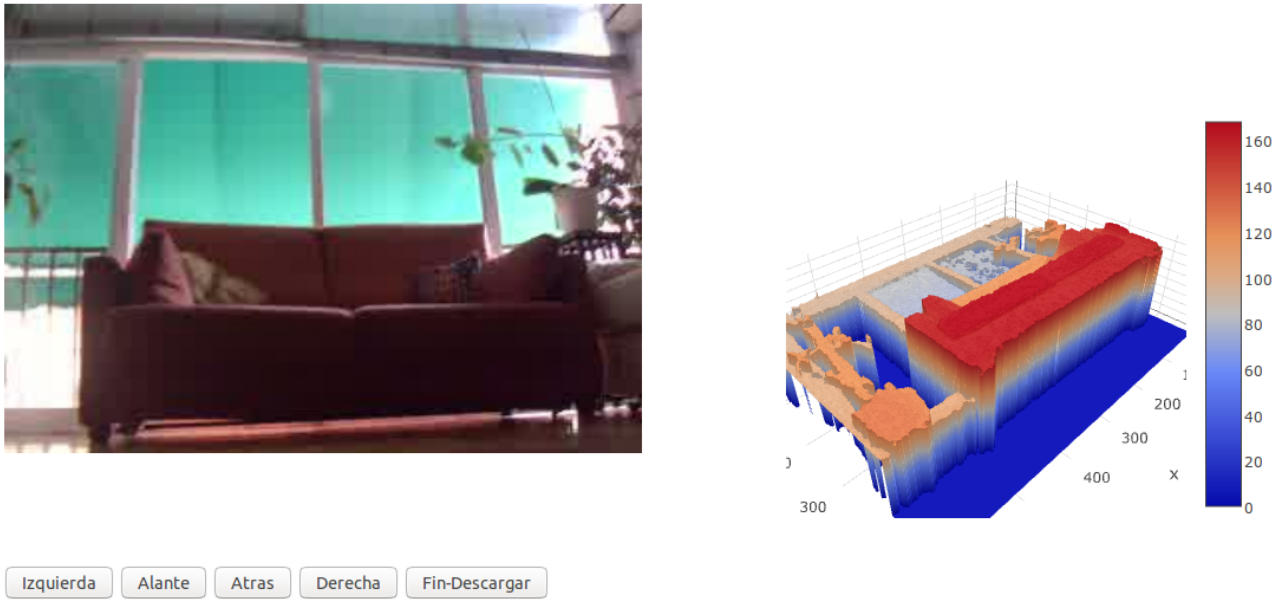


Figura 47: Adquisición 1.

La figura anterior se corresponde con la orientación inicial $\theta = 0\,rad$ y se aprecian los elementos p.2, S.1 y p.1. La representación tridimensional muestra cómo efectivamente la adquisición se ha realizado con éxito.

- Segunda adquisición:

Como hemos realizado ocho adquisiciones, la orientación con la que aproximadamente hemos efectuado esta segunda adquisición es $\theta = \left(\frac{2 \cdot \pi}{8}\right) rad = \left(\frac{\pi}{4}\right) rad$. Obsérvese que en este caso la parte derecha del elemento S.2 se encuentra demasiado cerca de la cámara Kinect y por tanto los datos no son del todo fidedignos. Esto traerá consecuencias en cuanto a la representación tridimensional. Recordemos que el rango de adquisición de datos del dispositivo Kinect se sitúa entre 0.5 m y 5 m, como hemos comentado en la sección 4.1.1 (Cámara Kinect I). Otra fuente de incertidumbre es el error acumulativo de los datos del encoder. A medida que se realizan adquisiciones después de un movimiento del robot, las posiciones y las orientaciones serán menos

exactas respecto a la posición original. Por este motivo dotamos al robot con otros elementos que nos ayuden a posicionarlo como es la cámara que apunta al techo para la localización de los códigos QR y códigos de color.



Figura 48: Adquisición 2.

- Tercera adquisición:

  La orientación de la tercera adquisición es $\theta_3 = 2 \cdot \left(\frac{\pi}{4}\right) rad$. En este caso se aprecian los elementos S.2, m.1 y a.1. Como puede observarse, la representación tridimensional ha sido rotada para mostrar su visión frontal.

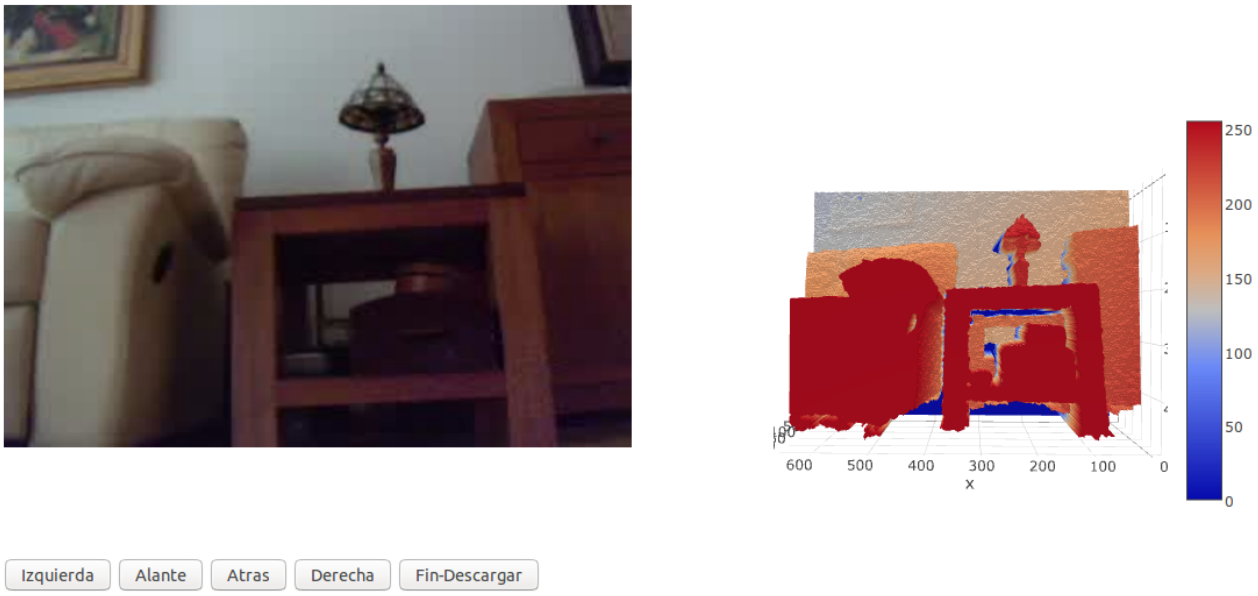


Figura 49: Adquisición 3.

- Cuarta adquisición:

  La orientación en el caso de la cuarta adquisición es $\theta_4 = 3 \cdot \left(\frac{\pi}{4}\right) rad$ En esta ocasión podemos ver los elementos a.1, P.1, S.3 y c.1. Mostramos también dos recortes de la representación tridimensional rotada para apreciar correctamente todos los detalles de la adquisición realizada por el dispositivo Kinect.

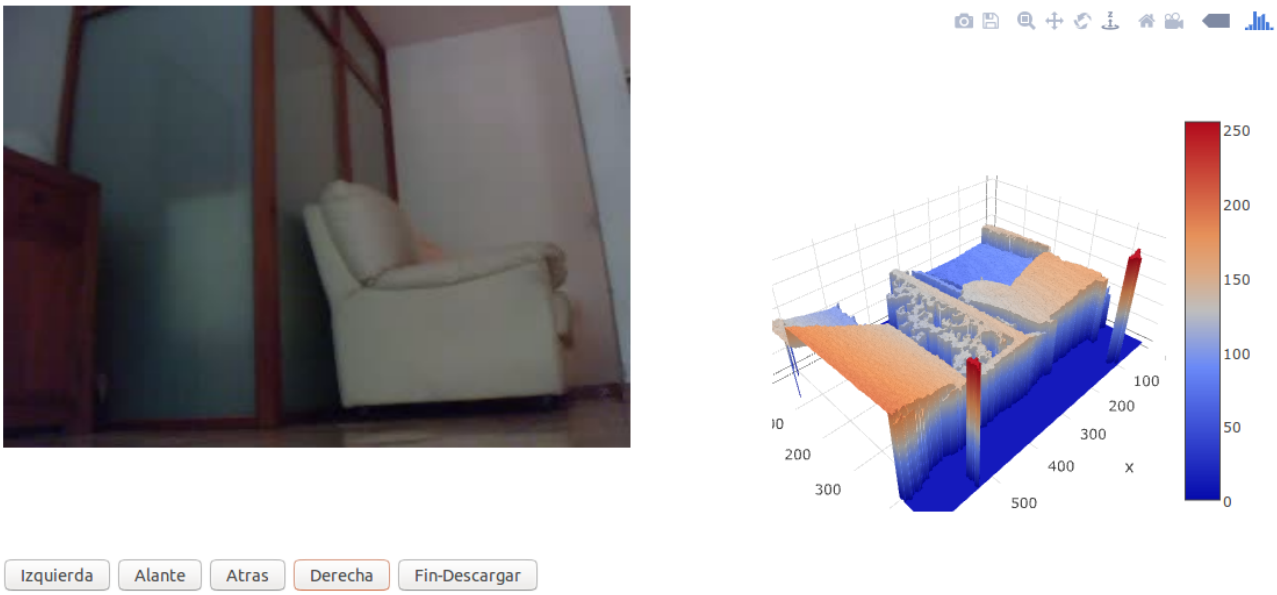


Figura 50: Adquisición 4.

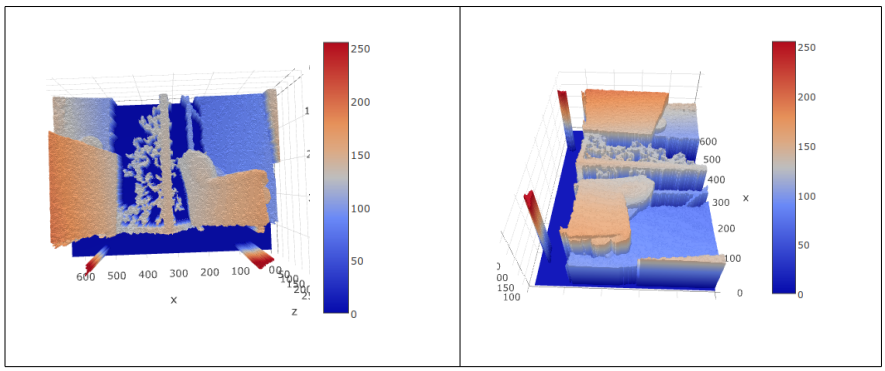


Figura 51: Detalle Adquisición 4.

- Quinta adquisición:

La orientación de la quinta adquisición es $\theta_5 = 4 \cdot \left(\frac{\pi}{4}\right) rad$ .En esta adquisición podemos ver los elementos S.3, c.1, m.2, P2 y pi.1.

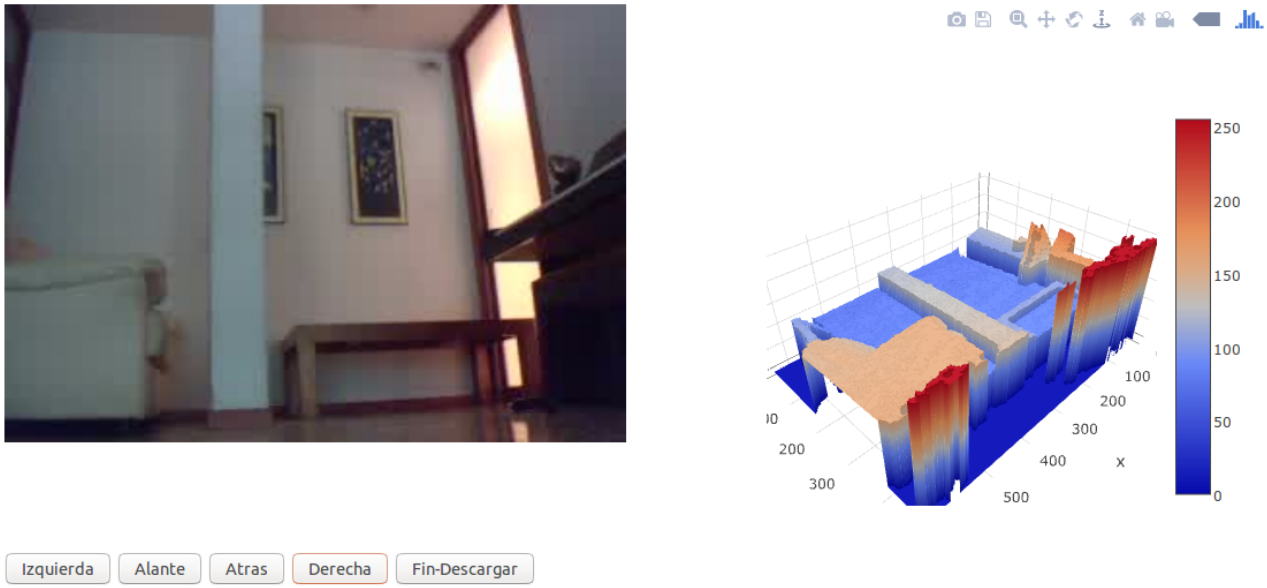


Figura 52: Adquisición 5.

- Sexta adquisición:

En la sexta adquisición el robot presentaba una orientación con $\theta_6 = 5 \cdot \left(\frac{\pi}{4}\right) rad$. En esta adquisición podemos apreciar los elementos pi.1 y P.3

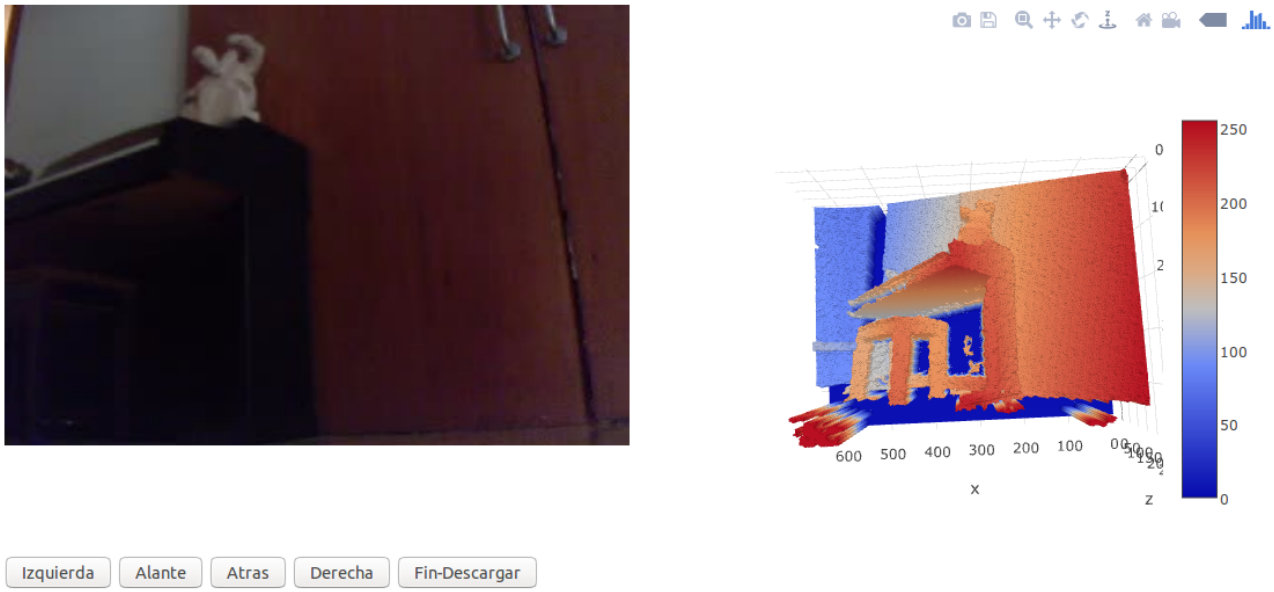


Figura 53: Adquisición 6.

Séptima adquisición:

En la séptima adquisición la orientación del robot es $\theta_7 = 6 \cdot \left(\frac{\pi}{4}\right) rad$ en esta adquisición solo se aprecia el elemento P.3

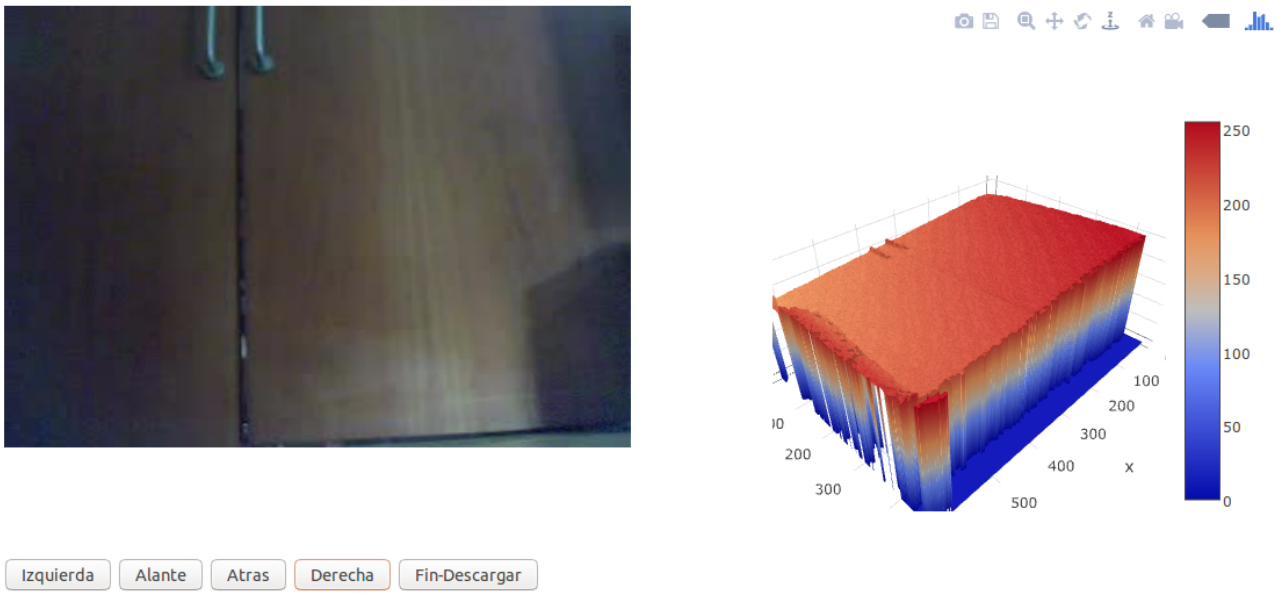


Figura 54: Adquisición 7.

Octava adquisición:

Esta es la última adquisición y se puede visualizar P.3, t.1 y p.2. La orientación que disponía el robot en esta adquisición es $\theta_8 = 7 \cdot \left(\frac{\pi}{4}\right) rad$.

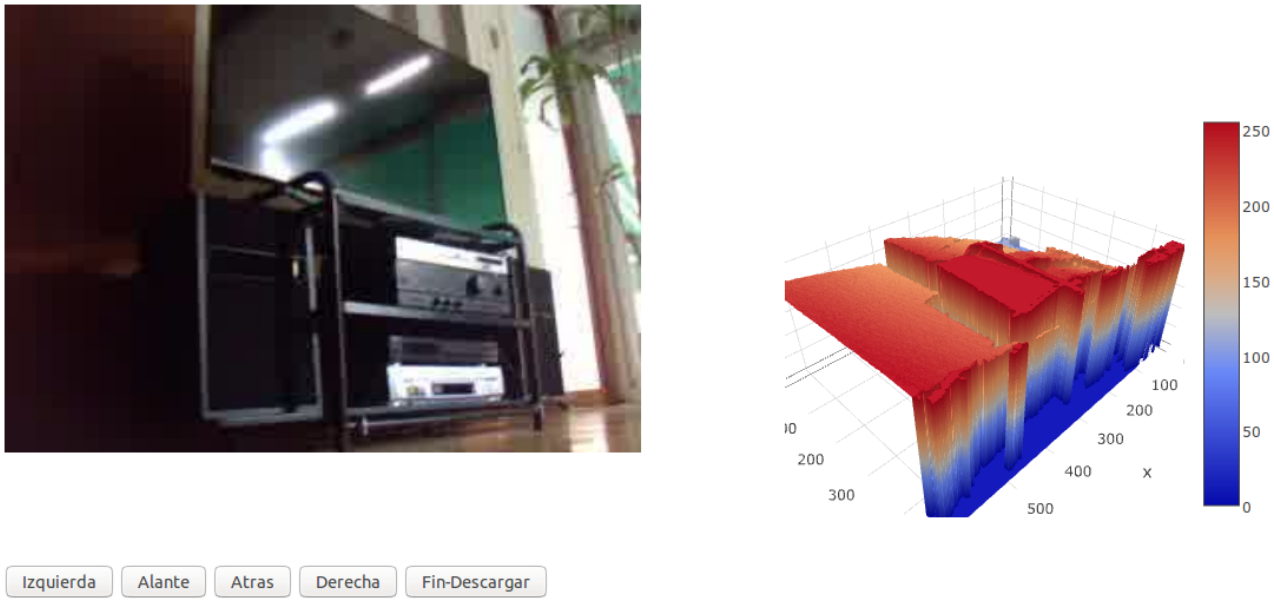


Figura 55: Adquisición 8.

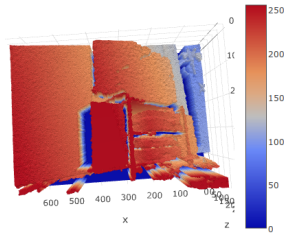


Figura 56: Detalle Adquisición 8.

Una vez realizada la experiencia anterior procedemos a finalizar la sesión y descargar los datos pulsando el botón Fin-descargar. La descarga se realizará por comunicación ssh, para trasladar los archivos desde el servidor al cliente. Los archivos que recibimos serán los siguientes:

matriz.m	Sensors.m	Photo.m	*.png

Gracias a los cuales podemos realizar el siguiente mapa tridimensional, mediante el análisis de los mismos utilizando Matlab y aplicando las consideraciones expuestas en las secciones 7.1.1 (Modelo diferencial) y 9.2 (Construcción del Mapa 3D).

En la primera gráfica que presentamos (Figura 57: Gráfica 1) podemos visualizar distintos enseres que se hallaban en la estancia. Se aprecian dos sofás, las ventanas, el toldo, unos trazos irregulares que se corresponden con las plantas, mesilla y un televisor; el resto de elementos no se aprecia con claridad, ya que la perspectiva nos impide verlos. No obstante, mostraremos una sucesión de imágenes en las cuales se rota la gráfica tridimensional, para poder visualizar los elementos citados con mayor claridad.

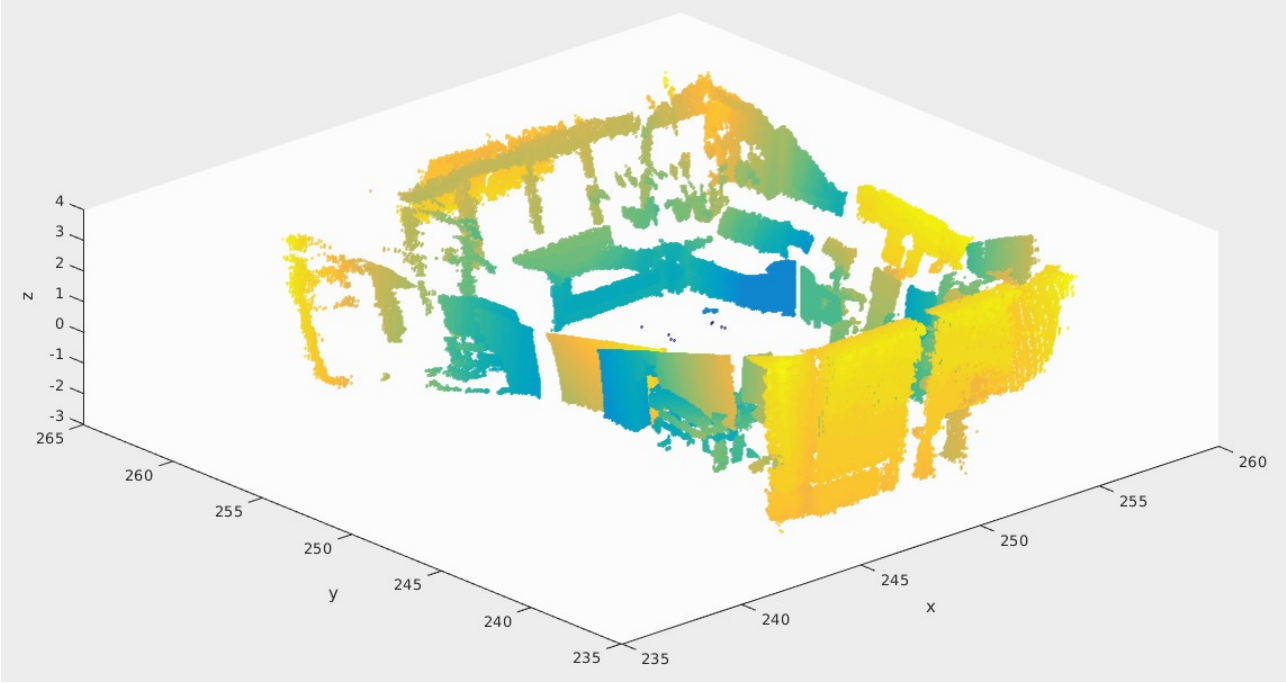


Figura 57: Gráfica 1.

El punto inicial O desde el cual el robot realiza la primera adquisición lo hemos impuesto en el $(x_i, y_i, z_i) = (250, 250, 0)$. Apréciese que la escala que hemos utilizado es $1 : \frac{7}{20} m$, es decir, una unidad en la gráfica equivalen a $0{,}35\, m$ reales. De tal modo que por inspección directa de la gráfica, se puede concluir que el largo total de la figura (en el eje y) ocupa desde 240 hasta 260, es decir, 20 unidades. Y el ancho (eje x) ocupa desde 242 hasta 257, es decir, 10,5 unidades. Y según la escala que hemos propuesto,

estas distancias se corresponderán con $20 \cdot \frac{7}{20} = 7\,m$ para el largo y $10 \cdot \frac{7}{20} = 3{,}68\,m$. Distancias que se corresponden con las mostradas en el esquema de la estancia (Figura 42).

En la siguiente secuencia de imágenes podemos apreciar el mapa tridimensional anterior desde distintas perspectivas. Aprovechamos para identificar los distintos elementos que se encuentran en la estancia y que han sido descritos en la tabla 8 (Elementos/Enseres).

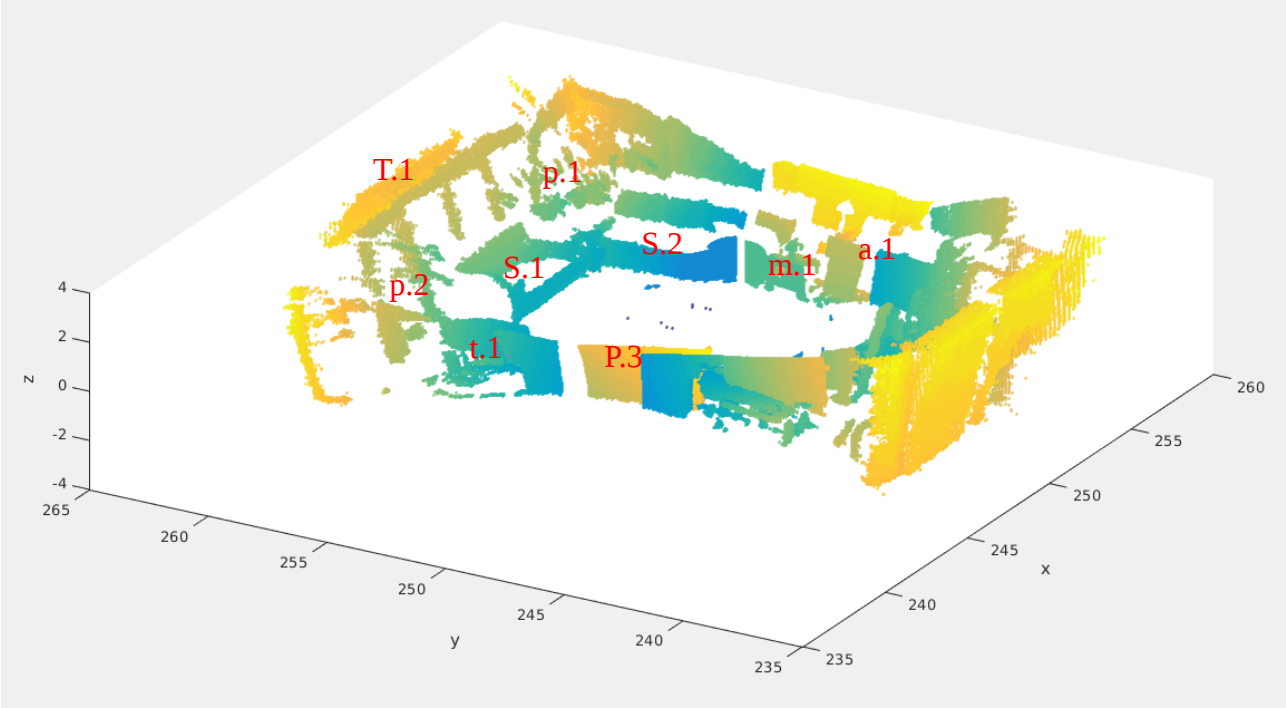


Figura 58: Gráfica 2.

En la siguiente imagen mostramos una gráfica similar a la anterior pero, vista desde un poco más arriba. En ella podemos apreciar claramente como han sido captados todos los elementos de la estancia. En concreto, en esta gráfica se aprecian claramente los elementos T.1, S.1, p.1,S.2, m.1, a.1, P.3, t.1 y p.2.

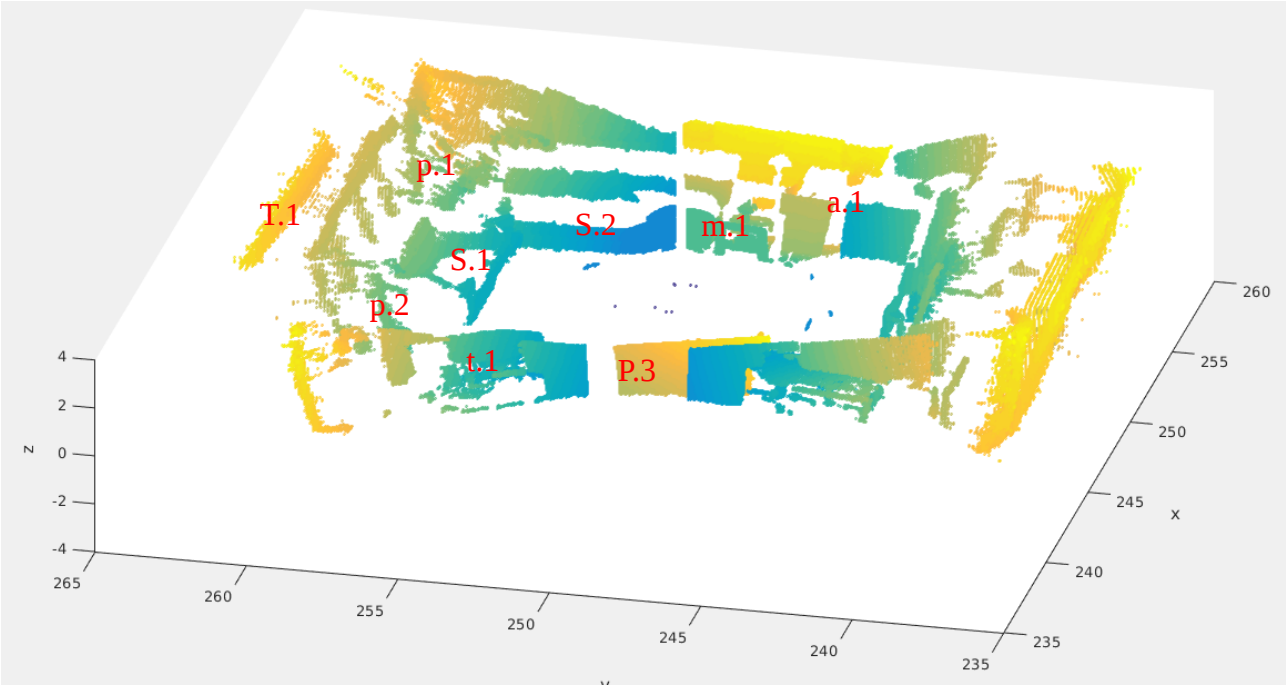


Figura 59: Gráfica 3.

Seguimos rotando la figura, a medida que giramos la gráfica se distorsionan unos elementos y se ven con mayor claridad otros.

En la siguiente imagen se empiezan a ver con mayor claridad los elementos situados en el lado contrario de la estancia. En concreto podemos ver con claridad los elementos S.3, c.1, P.1, m.1, a.1, P.3 y T.1. Obsérvese que el elemento T.1 (toldo) se ve con mucha mas claridad que con las perspectivas anteriores.

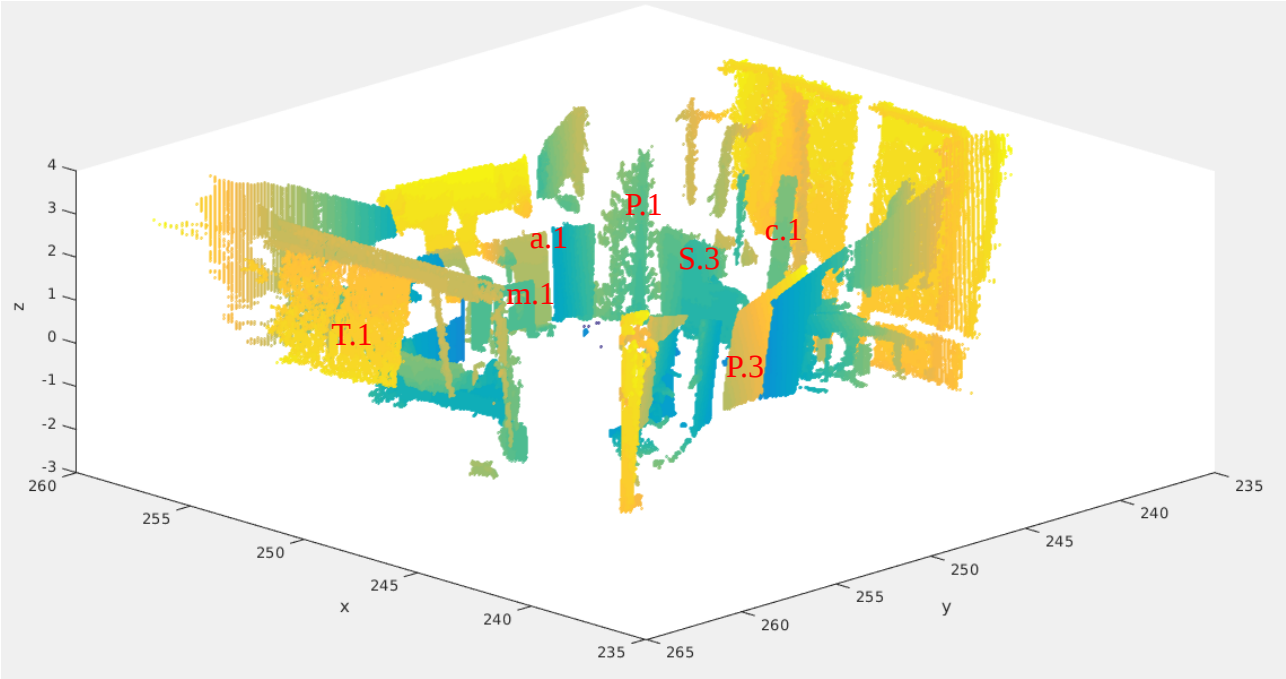


Figura 60: Gráfica 4.

La siguiente imagen muestra la misma perspectiva que en la imagen anterior pero un poco más arriba.

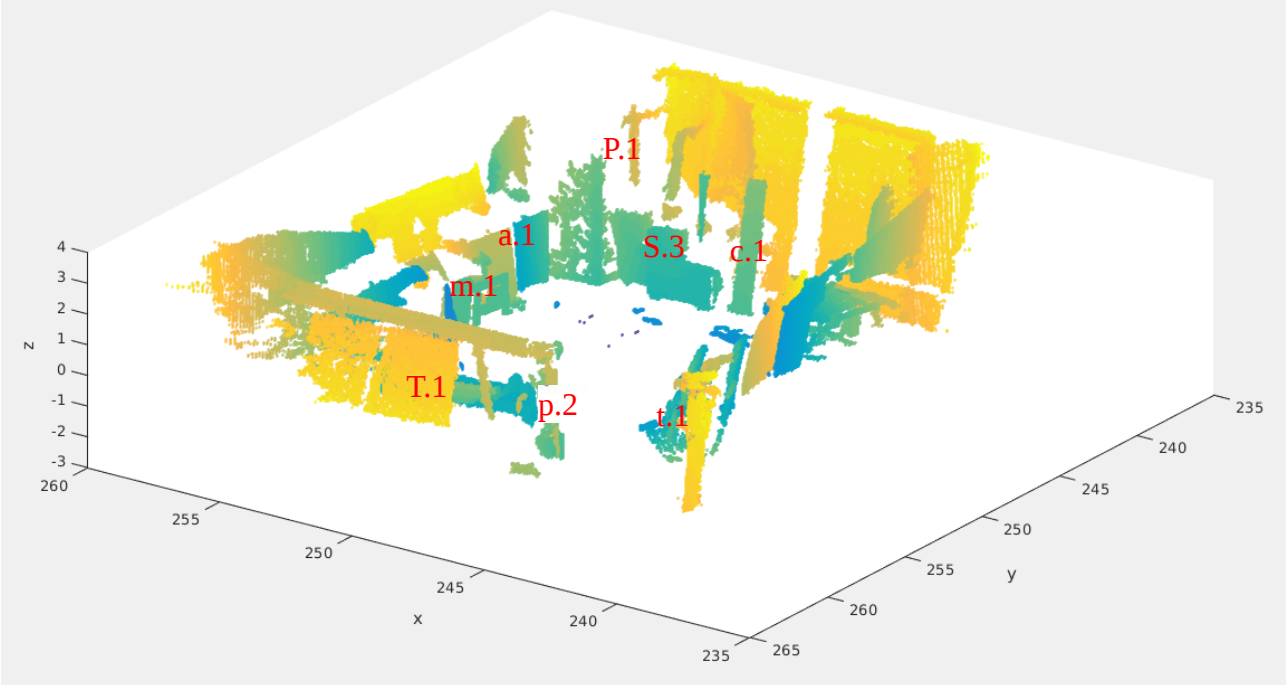


Figura 61: Gráfica 5.

Si seguimos girando para apreciar los elementos que antes no se veían. Es decir pi.1, P.3 y t.1.

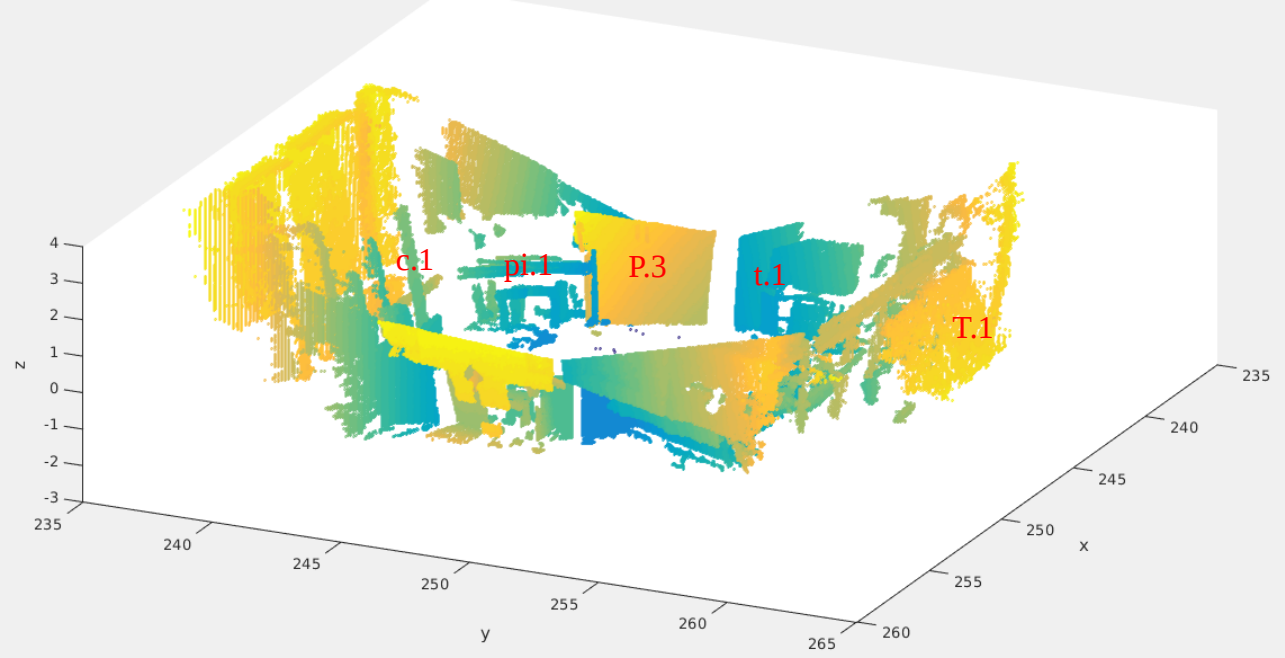


Figura 62: Gráfica 6.

En la siguiente imagen podemos visualizar el mapa tridimensional visto desde abajo. La linea discontinua representa el suelo.

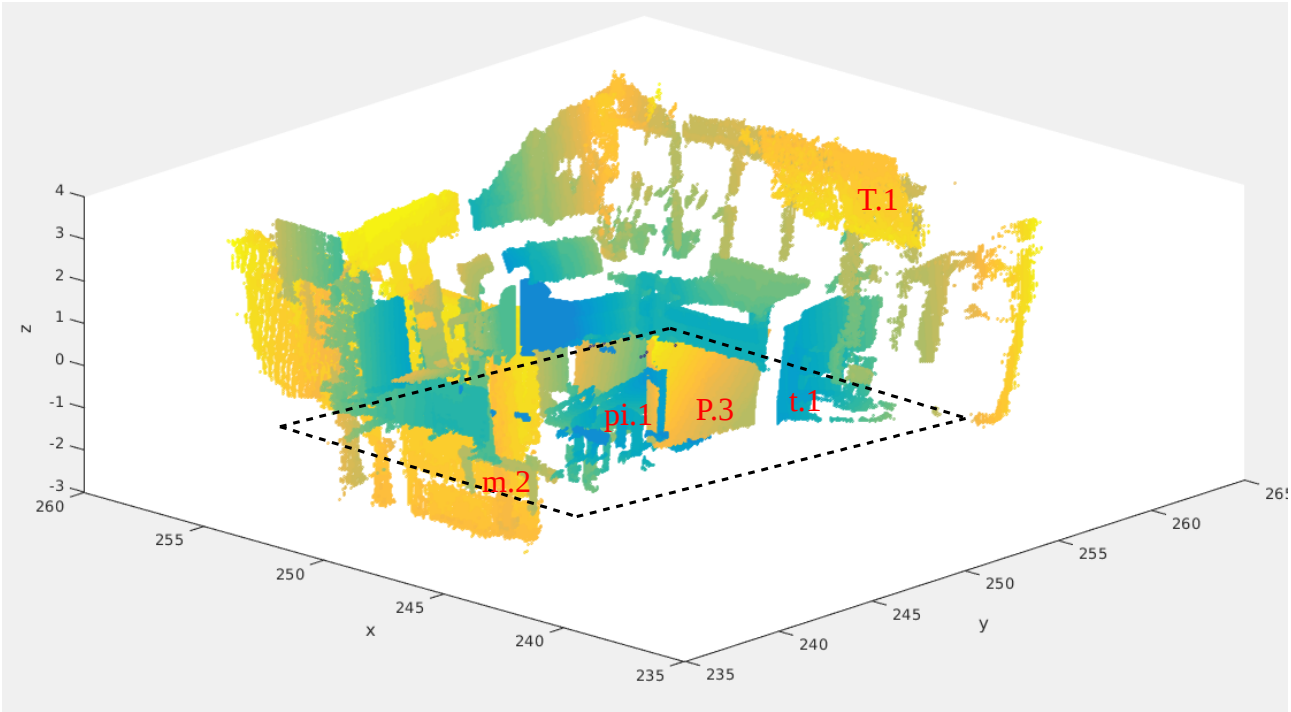


Figura 63: Gráfica 7.

El conjunto de imágenes anteriores muestra la construcción del mapa tridimensional en base a los datos recogidos por la Kinect. La colocación en el espacio tridimensional se

ha realizado en base a los datos del encoder y aplicando el modelo diferencial expuesto en la sección 9.2 (Construcción del Mapa 3D). Tal y como hemos comentado al inicio de esta sección. El código utilizado para la elaboración de los resultados se adjunta con la presente memoria, igual que el código desarrollado para la creación de la plataforma.

CONCLUSIONES

11.- Conclusiones

En el presente proyecto se ha creado una plataforma que permite el desarrollo de experiencias por parte de distintos usuarios. Gracias a ella se pueden llevar a cabo diversos experimentos que permiten la implementación de diferentes modelos para el desarrollo de mapas tridimensionales.

Tuvimos que llevar a cabo una amplia investigación sobre el estado del arte en el desarrollo de mapas tridimensionales y familiarizarnos con el dispositivo Kinect para el cual existen distintas librerías de software libre que nos permiten el control y adquisición de los datos para su posterior tratamiento.

Fue importante también familiarizarse con JavaScript que nos ha permitido controlar todo de manera unificada.

A lo largo del desarrollo del proyecto nos hemos encontrado con diferentes problemas que hemos tenido que solventar gracias a la dedicación continuada en las pruebas y modificaciones que debimos realizar. Enfrentarnos a estos problemas y el ahínco en la búsqueda de soluciones nos ha permitido encontrar un vasto abanico de posibilidades que nos a ayudado a alcanzar nuestros objetivos.

Finalmente conseguimos que el robot en su conjunto funcionara como se pone de manifiesto en los resultados experimentales.

12.- Lineas Futuras

El principal objetivo de este proyecto consiste en generar una herramienta que pudieran utilizar otros alumnos para poder llevar a cabo el análisis de los resultados obtenidos. Por tanto, sería interesante generar las adaptaciones necesarias para incorporarlo a UNILABS junto con otros dispositivos que permiten el desarrollo de experiencias similares.

En este proyecto, el control es directo por órdenes del usuario sin la posibilidad de generar un control autónomo que dependa de la información que esté recibiendo el robot. Sería interesante proporcionar un sistema que permita un control autónomo en función de la información adquirida por los distintos sensores. Así como ampliar el código para que el usuario pueda realizar la adquisición cuando desee.

Por otro lado, también sería interesante dotar al robot con otros canales para la adquisición, como pueden ser acelerómetros, magnetómetros, etc. Utilizando métodos como los propuestos en [27] se puede realizar fusión de la información de los encoders, sensores inerciales y magnéticos para la localización de robots móviles.

Referencias-Bibliografía

[1] "Navegación de un robot autónomo utilizando balizas y luz estructurada" Juan Ignacio Forcén Carvalho, Dictino Chaos García. September 2015.

[2] Adams, James Lowell, "Remote control with long transmission delays", PhD in Mechanical Engineering, 1961.

[3] Braisted, Paul Wilder, "Study of a predictor for remote control systems operating with signal transmission delays", PhD in Mechanical Engineering, 1963.

[4] Ken Goldber, Roland Siegwart. "Beyond Webcams: an introduction to online robots".ISBN 0-262-07225-4 (hc. : alk. Paper).

[5] Moravec, Hans Peter, "Obstacle avoidance and navigation in the real world by a seeing robot rover", PhD in Computer Science, 1980.

[6] Sebastian Thrun, Dirk Hähnel, David Ferguson, Michael Montemerlo, Rudolph Triebel, Wolfram Burgard, Christopher Baker, Zachary Omohundro, Scott Thayer, William Whittaker. "A System for Volumetric Robotic Mapping of Abandoned Mines". School of Computer Science Carnegie Mellon University Pittsburgh, PA, USA. ICRA-2003.

[7] http://www.ai.sri.com/centibots/ "(Accesed on 5 September 2016).".

[8] http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx "(Accesed on 5 September 2016).".

[9] Kourosh Khoshelham, Sander Oude Elberink."Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications", 1 February 2012.

[10] http://blog.livedoor.jp/helloworld_since2003/?p=3 "(Accesed on 4 August 2016).".

[11] J. Borenstein H. R. Everett , and L. Feng Contributing authors: S. W. Lee and R. H. Byrne Edited and compiled by J. Borenstein; "Where am I? Sensors and Methods for Mobile Robot Positioning" April 1996.

[12] http://electgpl.blogspot.com.es/2014/02/sensor-de-proximidad-sharp-gp2y0a21.html, “Sensor de proximidad Sharp GP2Y0A21 (Accesed on 5 August 2016).”.

[13] http://es.mathworks.com/products/availability/ “System Requirements & Platform Availability by Product for R2016ª (Accesed on 20 January 2016).”.

[14] https://www.arduino.cc/en/Main/ArduinoMotorShieldR3, “Arduino Motor Shield (Accesed on 9 August 2016).”.

[15] https://www.arduino.cc/en/Main/Software, “Download the Arduino Software. (Accesed on 10 August 2016).”.

[16] http://www.netconsulting.es/blog/nodejs/ “Node.js: ¿Qué es y para que sirve NodeJS? (Accesed on 10 August 2016).”.

[17] https://ffmpeg.org/about.html, “About FFmpeg (Accesed on 12 August 2016).”.

[18] http://news.mit.edu/2013/3d-mapping-in-real-time-without-the-drift-0828, “3-D mapping in real time, without the drift” (Accesed on 13 August 2016).”.

[19] Puneet Goel, Stergios I., S.Sukhatme; “Robot Localization Using Relative and Absolute Position Estimates” Department of Computer Science, Institute for Robotics and Intelligent Systems, University of Southern California, Los Angeles, CA 90089-0781.

[20] Jhonny A. Valencia, V.Alejandro Montoya O., Luis Hernando Rios, M. Sc. “Modelo cinemático de un robot móvil tipo diferencial y navegación a partir de la estimación odométrica” Universidad Tecnológica de Pereira. ISSN 0122-1701. Scientia et Technica Año XV, No 41, May 2009.

[21] Alberto Bañó Azcón, Albert Oller Pujol. “Análisis y Diseño del Control de Posición de un Robot Móvil con Tracción Diferencial” Enginyeria en Automática i Electrónica industrial. Universitat Rovira I Virgili Departament d’Enginyería Electrónica Elèctrica i Automática . June 2003.

[22] https://cuentos-cuanticos.com/2011/12/15/robotica-estimacion-de-posicion-por-odometria/, “Robótica: Estimación de posición por odometría” (Accesed on 13 August).”.

[23] http://www.uco.es/investiga/grupos/ava/node/26, “ArUco: a minimal library for

Augmented Reality applications based on OpenCV (Accesed 15 August 2016).”.

[24] https://plot.ly/javascript/, “The open source JavaScript graphing library that powers plotly (Accesed 17 August 2016).”.

[25] Danilo Navarro.Gines Benet T. Gilabert. Luis Hernando Rios. Maximiliano Bueno L. “Mejoras en la localización odométrica de un robot diferencial mediante la corrección de errores sistemáticos” 2007.

[26] Surachai Panich, Nitin Afzulpurkar, “Sensor Fusion Techniques in Navigation Application for Mobile Robot” Srinakharinwirot University, Thailand. June 13, 2011.

[27] Thomas Fischer , Matías A. Nitsche, Sol Pedre. “Fusión de encoders de cuadratura, sensores inerciales y magnéticos para la localización de robots móviles” Laboratorio de Robótica y Sistemas Embebidos, Departamento de Computación, FCEyN – UBA Grupo de Robótica CAREM - CAB – CNEA]

[28] http://wiki.octave.org/Rasperry_Pi “Raspberry Pi (Accesed 16 August 2016).”.

[29] https://www.gnu.org/software/octave/, “GNU Octave (Accesed 17 August 2016).”.

[30] https://openkinect.org/wiki/Getting_Started#Ubuntu_Manual_Install, “Getting Started (Accesed 18 August 2016).”.

[31] http://psychtoolbox.org/linux/ “Psychtoolbox-3 (Accesed 18 August 2016).”.

[32] http://www.jeffreythompson.org/blog/2014/11/13/installing-ffmpeg-for-raspberry-pi/, “Insstalling FFMPEG for Raspberry Pi (Accesed 21 August 2016).”.

[33] ttps://www.npmjs.com/package/raspberry,“Raspberry Toolkit (Accesed 24 August 2016).”.

[34] https://github.com/EmergingTechnologyAdvisors/node-serialport/issues/649,“Failed to install node-serialport on Raspberry Pi2 (Accesed 25 August 2016).”.

Abreviaturas y acrónimos

ARM: Advanced RISC Machine (Máquina RISC Avanzada).

ArUco: Augmented Reality Universidad de Códoba.

ETS: Escuela de Técnicos Superiores.

IDE: Integrated development environment (Entorno de Desarrollo Integrado).

Wi-Fi: Wireless Fidelity.

ANEXOS

1.- Guía de instalación

En la presente sección mostraremos los pasos necesarios para llevar a cabo el proyecto desde cero. En primer lugar señalaremos los pasos a seguir para para la instalación del software relacionado con el control de la cámara Kinect y Octave. Con los cuales podremos llevar a acabo la adquisición de los datos de profundidad

Instalamos Octave y qtOctave en Raspberry Pi para disponer del entorno gráfico necesario para llevar a cabo la programación del servidor ServerOctave. Para ello seguimos los pasos descritos en [28] y [29]. Debemos también instalar los drivers necesarios para el control de la cámara Kinect [30]. Y las librerías Psychtoolbox-3 [31] en Octave.

Para poder llevar a cabo el control de la cámaras web debemos instalar el software ffmpeg. Para ello seguimos los pasos descritos en [32].

La instalación de Node.js para utilizar los scrips de Javascrip que nos permiten el funcionamiento de cada lazo se realiza mediante los pasos descritos en [33].

Cuando hayamos instalado Node.js y utilicemos los scrips de Javascrip, es posible que nos solicite la instalación de alguna librería en concreto. Esto no supone ningún problema ya que mediante la sentencia (escrita en la consola de Raspberry pi) `-$: npm install "nombre_de_la_librería"` se instalarán correctamente. Surgieron numerosos problemas con una librería en concreto, se trata de "serialport" Fue necesario descargarse una versión precompilada de la misma y esto se consiguió mediante las siguientes sentencias [34], escritas en la terminal de Raspberry Pi:

```
wget http://node-arm.herokuapp.com/node_latest_armhf.deb
dpkg -i node_latest_armhf.deb
npm install -g node-gyp
sudo npm install --unsafe-perm serialport
```

2.- Puesta en Marcha

En esta sección indicamos los pasos a seguir para iniciar el Servidor y el Cliente. Las indicaciones son necesarias para la puesta en marcha del sistema y, gracias a ellas, se podrá iniciar la experiencia que nos permita recaudar los datos necesarios para la construcción de los mapas tridimensionales.

En primer lugar procederemos al encendido del robot. Como hemos comentado en la sección 3.2 (Conexiones Alimentación.), el robot dispone de un botón en la parte trasera con 3 posiciones, abajo [Posición de carga], en medio [Posición apagado] y arriba [Posición encendido]. Encendemos el robot colocando el interruptor trasero en la posición de encendido y esperamos, un tiempo prudente (3 minutos aproximadamente), a que Raspberry Pi se haya encendido. Una vez encendida Raspberry Pi nos comunicaremos con ella a través de SSH mediante la siguiente instrucción:

ssh pi@192.168.1.34

Cuando la conexión haya sido establecida, nos pedirá la contraseña (“raspberry”). Una vez dentro, podemos iniciar los servidores, pero antes, debemos proceder al borrado de los archivos generados en la experimentación anterior. Para borrar estos archivos solo tendremos que teclear “Borrar” en una terminal que esté comunicada mediante ssh con Raspbeerry Pi. El scrip “Borrar” está alojado en el bash y tiene las ordenes necesarias para realizar el borrado de los archivos Sensor.m, matriz.m, Photo.m, así como todas las fotografías del techo que se hayan ido guardando en el servidor; ahora podemos iniciar los servidores.

En nuestro caso, por mayor comodidad a la hora de desarrollar el código, decidimos establecer comunicación mediante un escritorio remoto gracias a VNC. Establecemos los permisos para que se pueda ejecutar el escritorio remoto mediante la siguiente orden (dentro de Raspberry Pi):

vncserver :1

Iniciamos VNC en el ordenador remoto y procedemos a arrancar todos los servidores. Para ello, en primer lugar, iniciamos qtoctave y, mientras arranca, vamos iniciando los servidores. Las instrucciones, en terminales independientes de la Raspberry Pi, serán las siguientes:

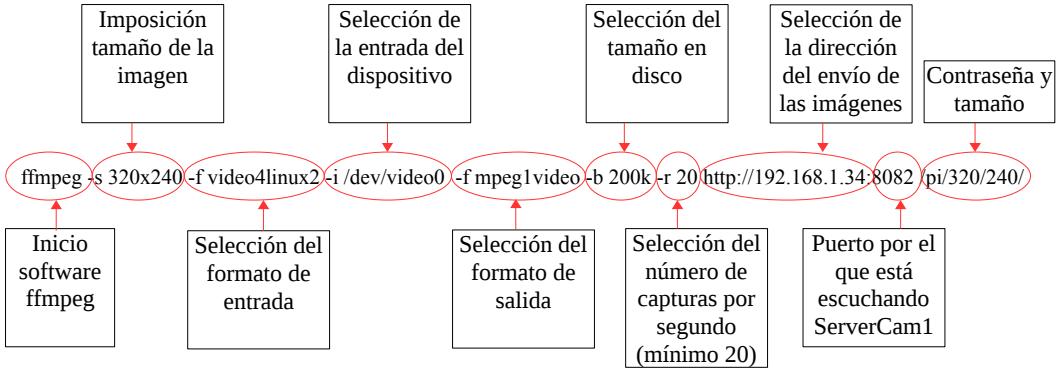
Instrucción + → En terminal ++ → En software específico	**Acción**
qtoctave+	Se inicia el software Octave.
ServerOctave+	Se inicia el Servidor ServerOctave, encargado de gestionar las matrices que recibe de Octave.
ServerCam1+	Se inicia el servidor ServerCam1, encargado del streaming.
ServerCam11+	Se inicia el software ffmpeg, que suministra las imágenes a ServerCam1.
ServerCam2+	Se inicia el servidor ServerCam2, encargado de las fotografías del techo.
ServerCam22+	Se inicia el software ffmpeg, que suministra las imágenes a ServerCam2.
run('Kinect_node')++	Nos situamos en qtoctave y arrancamos el script Kinect_node.m.Tengamos en cuenta que qtoctave siempre se sitúa en la carpeta donde se encuentra el archivo Kinect_node.m.
ServerArduino+	Se inicia el servidor ServerArduino, encargado de gestionar la comunicación bidireccional de Arduino.

Tabla 9: Instrucciones Arranque.

Las órdenes que escribimos en la terminal pertenecen a scrips alojados en el bash que, a su vez, llaman a otros scrips para iniciar los archivos js o los software que trabajan con js. En la siguiente tabla podemos encontrar los directorios específicos de cada archivo y su manera de arrancarlos. Esta información nos permite realizar modificaciones en los scripts y arrancarlos manualmente desde la carpeta en la que se encuentran alojados.

ServerOctave	
Carpeta:	/home/pi/Desktop/Servertutti/OCTAVEII/
Archivo:	server.js
Arranque manual:	node server.js

ServerCam1	
Carpeta:	/home/pi/Desktop/Servertutti/jsmpeg-stream
Archivo	Stream-server.js
Arranque manual:	Node Stream-server.js

ServerCam11
ServerCam11 no es en sí mismo un servidor, sino el software que coopera con el servidor ServerCam1. ServerCam11 se encarga de activar la web cam delantera y enviar las imágenes a la dirección http://192.168.1.34:8082/ , en el puerto 8082 estará escuchando ServerCam1 para gestionar las imágenes recibidas.
Arranque manual: 

ServerCam2	
Carpeta:	/home/pi/Desktop/Servertutti/jsmpeg-stream
Archivo	Stream-server.js
Arranque manual:	Node Stream-server.js

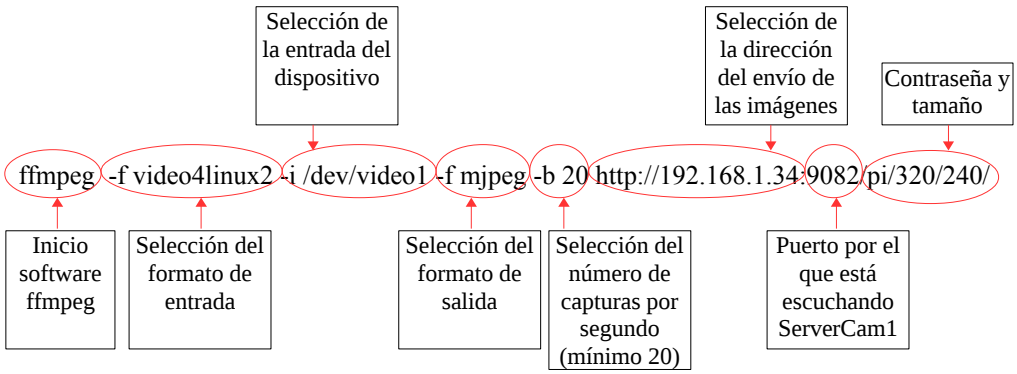
ServerCam22
Al igual que ServerCam11, ServerCam22 es el software que coopera con el servidor ServerCam2. ServerCam22 se encarga de activar la webcam que apunta al techo y enviar las imágenes a la dirección http://192.168.1.34:9082/ , en el puerto 9082 estará escuchando ServerCam2 para gestionar las imágenes recibidas.
Arranque manual: 

Tabla 10: Instrucciones arranque manual.

Durante las pruebas, nos dimos cuenta de que, a veces, tener los tres dispositivos activados, las dos webcam y la cámara Kinect, podía ocasionar interrupciones en las adquisiciones. No siempre ocurría, pero es motivo suficiente para buscar una alternativa que nos asegure realizar las adquisiciones con garantías. Una solución posible es generar un scrip en el bash que llamara alternativamente a una y otra webcam, de tal forma que nunca trabajen a la vez. Generalmente estará activa la webcam del streaming y cada cierto tiempo se interrumpe para dejar que la otra cámara realice una adquisición. En la siguiente tabla se puede apreciar la estructura de este script. Como puede observarse, en este caso, hemos añadido la instrucción -t 10 (que cortará la transmisión en 10 segundos) para dejar paso a la otra webcam (que apunta el techo) para realizar la fotografía, con la orden -t 2, solo estará activa 2 segundos. Si decidimos trabajar con ServerCam11 y ServerCam22, no iniciaremos ServerCam12 y viceversa.

ServerCam12
Alojado en: /usr/local/bin
Arranque: ServerCam12

```
#!/bin/bash

while true
do

ffmpeg -t 10 -s 320x240 -f video4linux2 -i /dev/video0 -vf vflip,hflip -f mpeg1video -b 200k -r
20 http://192.168.1.34:8082/pi/320/240/

ffmpeg -t 2 -f video4linux2 -i /dev/video1 -f mjpeg -b 20 http://192.168.1.34:9082/pi/320/240/

done
```

Esta orden es importante: se utiliza para rotar la imagen. vflip cambia las filas de orden, de arriba a abajo. Y hflip cambia las columnas de orden de derecha a izquierda. Actualmente esta orden es necesaria, ya que hemos dispuesto la cámara web en posición inversa.

Tabla 11: Alternativa ServerCam12

NOTA IMPORTANTE: Antes de iniciar el servidor ServerArduino, activaremos la página web del ordenador remoto, cuyo código se adjunta, como el resto de código, con la presente memoria. Una vez iniciada la página web "client.html", podemos activar el último servidor, ServerArduino, para iniciar la experiencia. Si no seguimos escrupulosamente el orden indicado los datos recibidos por Arduino serán almacenados de manera corrupta. No se debe tampoco refrescar la página web, ya que de nuevo se corromperán las adquisiciones que provengan de Arduino (datos de los encoders y datos de los sensores IR).

Por último, una vez finalizada la experiencia, pulsamos el botón Finalizar/Descargar. Este botón envía un mensaje al servidor indicando que se desea finalizar la experiencia. El navegador indicará al usuario cual es la instrucción para descargar los archivos con los datos generados durante la experiencia:

```
scp pi@192.168.1.34:~/Desktop/Servertutti/OCTAVEII/matriz.m
/destino/donde/queremos/alojar/los/datos

scp pi@192.168.1.34:~/Desktop/Servertutti/Sensors.m
/destino/donde/queremos/alojar/los/datos

scp pi@192.168.1.34:~/Desktop/Servertutti/jsmpeg-techo/Photo.m
/destino/donde/queremos/alojar/los/datos

scp pi@192.168.1.34:~/Desktop/Servertutti/jsmpeg-techo/*.png
/destino/donde/queremos/alojar/los/datos
```

3.- Datasheets

En los siguiente tabla exponemos los enlaces con los cuales podemos acceder a las hojas de especificaciones de cada uno de los elementos que forman parte de este proyecto. En ellas, se puede consultar todas las cualidades y especificaciones que requiere cada uno de los dispositivos:

Elemento	Enlace
Raspberry Pi 2 model B	https://www.adafruit.com/pdfs/**raspberrypi2**modelb.pdf
Arduino Rev 3	https://www.arduino.cc/en/Main/ArduinoBoardUno
Motor Shield	https://www.arduino.cc/en/Main/ArduinoMotorShieldR3
Kinect	https://msdn.microsoft.com/en-us/library/jj131033.aspx
Camara trust	http://www.trust.com/es/product/16428-spotlight-webcam-pro
Sensores IR	https://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwiH0ZSMiOnOAhXGLMAKHXiOAUEQFggmMAA&url=http%3A%2F%2Fwww.sharpsma.com%2Fwebfm_send%2F1208&usg=AFQjCNH5JjTZfEtejaZF0DlQEeQNM-zEzw&sig2=TWCZnk94IYOXfTb8s8HdpA&cad=rja
Wi-Fi Usb	https://www.asus.com/es/Networking/USBN10_NANO/HelpDesk_Manual/

Tabla 12: Enlaces *Datasheets*.

4.- Linea temporal de los hitos del proyecto



Figura 64: línea temporal de los hitos del proyecto.