Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería en Electrónica Industrial y Automática

Estrategia de movimientos sin colisiones para un robot en el ámbito de cirugía robotizada

Autor:

García Panadero, Sergio

Tutor:

Fraile Marinero, Juan Carlos

Departamento de sistemas y automática

Valladolid, abril 2017.

Agradecimientos:

Quiero agradecer a mi familia, y en especial a mis padres todo el apoyo y ánimo recibido durante todos estos años de grado. En verdad, tengo mucho más que agradecerles, empezando por los todos los valores que me han inculcado desde pequeño y todo el esfuerzo que han realizado para que pueda llegar hasta aquí.

A Sara Caletrío por ser el pilar de mi vida, por darme tanto cariño, amor. Y por estar siempre ahí al pie del cañón pase lo que pase.

A todas esas grandísimas personas que he conocido durante todo el transcurso del grado y tengo el privilegio de llamar amigos. A todos los trabajos y exámenes que hemos superado juntos, a todas esas tardes de biblioteca y aulario. Y como no, por todas las cervezas que hemos compartido juntos.

A mis compañeros de MAPAO por toda la confianza que habéis depositado en mí durante todo este tiempo.

Por último agradecerles también, a mis amigos del pueblo, por seguir ahí como los que más, pese a no vernos en mucho tiempo. Agradecerles todos esos buenos momentos y los que nos queden.

A todos vosotros, de corazón: Muchísimas gracias.

Resumen

En este trabajo fin de grado se presenta el desarrollo de una estrategia de control de la trayectoria del robot ABB IRB 120 para su integración en un sistema robótico colaborativo orientado a la cirugía laparoscópica tipo HALS (Hand Robotic Laparoscopic Sugery), utilizando el sistema operativo ROS (Robot Operating System). Las tareas de comunicación se encargan de tomar la información procedente de una cámara de visión 3D y de un guante de datos, y se transmiten al robot a través de un Socket TCP. Una vez en el robot, los datos se almacenan y se utilizan para implementar la estrategia de control de la trayectoria del robot, orientada al seguimiento sin colisión del dedo meñique de la mano del cirujano.

Palabras clave:

HALS, ROS, ABB, Nodo, Tarea, Framework, Socket

Índice

INDICE DE FIGURAS

INDICE DE TABLAS

1. Introducción

1.1. Sistemas robóticos colaborativos.

Los sistemas robóticos colaborativos son aquellos en los cuales el robot puede cooperar con el humano dentro de condiciones seguras para este último. Supone un enfoque totalmente distinto al clásico, donde los robots son situados tras barreras y la colaboración del humano y el robot es prácticamente nula.

Esta tecnología ha sido posible gracias a la inclusión de sistemas de sensorización en el sistema robótico, tales como: sistemas de visión, sensores de esfuerzos y/o sensores de presión sobre todo el cuerpo del robot La incorporación de sensores junto a la utilización de nuevos materiales y diseños que incluyen conceptos como la eliminación de aristas vivas para reducir los daños de un eventual choque entre el humano y el robot. Además se han incluido nuevas rutinas de control por las cuales si los sensores del robot detectan que el choque es inminente, o si el operario entra en contacto con el robot, el robot cancelará toda operación de movimiento hasta que el riesgo sea nulo.

Un modelo de brazo robótico colaborativo de la empresa Universal Robots se muestra en la siguiente imagen:

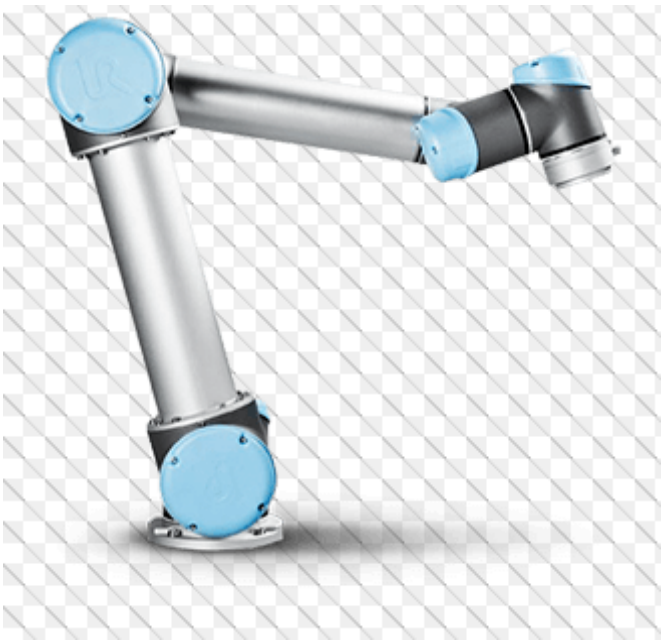

Figura 1 Robot colaborativo Modelo UR3 de la empresa Universal Robots

Esta nueva generación de robots posibilita el desarrollo de nuevos métodos de automatización como por ejemplo la cirugía laparoscópica robotizada o en procesos de montaje.

1.2. Cirugía robótica

Durante muchos años la cirugía laparoscópica ha sido el método quirúrgico más avanzado debido a la utilización de técnicas menos invasivas en el paciente que la cirugía convencional, lo que reduce el tiempo de convalecencia. Sin embargo, pese a ser un procedimiento quirúrgico muy eficaz la cirugía laparoscópica convencional presenta inconvenientes como la imposibilidad de palpar los órganos del paciente

Otros de los inconvenientes que tiene este tipo de cirugía radican en la mala ergonomía con la que debe trabajar el cirujano y el resto del equipo durante la intervención. Además, en este tipo de intervenciones todo el equipo médico se encuentra muy concentrado en torno a un área de operación muy pequeña provocando estrés e incomodidad en todo el equipo médico, limitación de movimientos.

Como medio de mejora para este tipo de intervenciones, se están desarrollando diferentes métodos para integrar robots en los procesos de cirugía. Así pues existen dos enfoques principales:

El primero de ellos consiste en utilizar un sistema robótico controlado por el cirujano que realizará toda la operación sin necesidad de disponer a un equipo de especialistas auxiliares. Este sistema mejora la ergonomía del cirujano pues pasa de estar delante del paciente a estar sentado al mando de los controles del sistema robótico. También mejora la precisión de las operaciones al incorporar técnicas de realidad aumentada.

El otro enfoque se fundamenta en sustituir parte del equipo médico por robots formando un sistema robótico colaborativo. La utilización de robots colaborativos permite aumentar la precisión de las operaciones a realizar.

Al igual que en el caso anterior, el hecho de implementar un sistema robótico colaborativo implica reducir el personal necesario para la cirugía y aportar mayor confort al cirujano.

En la siguiente imagen se expone una tabla comparativa de las ventajas que aporta la cirugía asistida por robots o cirugía robótica contra la cirugía laparoscópica convencional

Figura 2 Ventajas cirugía robótica

A continuación se presentan diferentes alternativas de sistemas de cirugía robótica

1.2.1 Robot Da Vinci:

El robot Da Vinci es un sistema robotizado de cirugía controlados por el cirujano utiliza una técnica de realidad aumentada. Este proyecto es pionero en su campo, siendo el primer sistema de cirugía robótica comercializado.

Figura 3 Proyecto Da Vinci

El robot incorpora una cámara de visión tridimensional de alta resolución cuya óptica es de 12 aumentos. El cirujano estará sentado en el puesto de mando donde visualizará las imágenes tomadas por la cámara en las pantallas. En el puesto de

mando el cirujano manipula los controles del robot, los movimientos detectados en los controles serán primero filtrados para eliminar a deriva producida por el temblor esencial (movimientos involuntario cuerpo). Una vez filtrados los movimientos serán reproducidos a escala, mejorando la precisión de la cirugía, minimizando la invasión y la seguridad del paciente.

Una limitación de estos robots son los grados de libertad que posee, por lo que el cirujano deberá tener en cuenta los movimientos que son permitidos durante la cirugía.

Las herramientas que pueden ser acopladas a los brazos del sistema son: una mano robótica, un endoscopio, una pinza y una herramienta para el corte de tejido. Estas herramientas son introducidas en el cuerpo del paciente a través de los trocares, unos cilindros huecos que permiten el cambio de herramienta sin necesidad de mover el brazo robótico de su posición.

Los resultados de utilizar el proyecto Da Vinci se resumen en:

- Reducción en un 40% del tiempo de convalecencia del paciente.
- Menor invasión que la cirugía laparoscópica.
- Menor sangrado

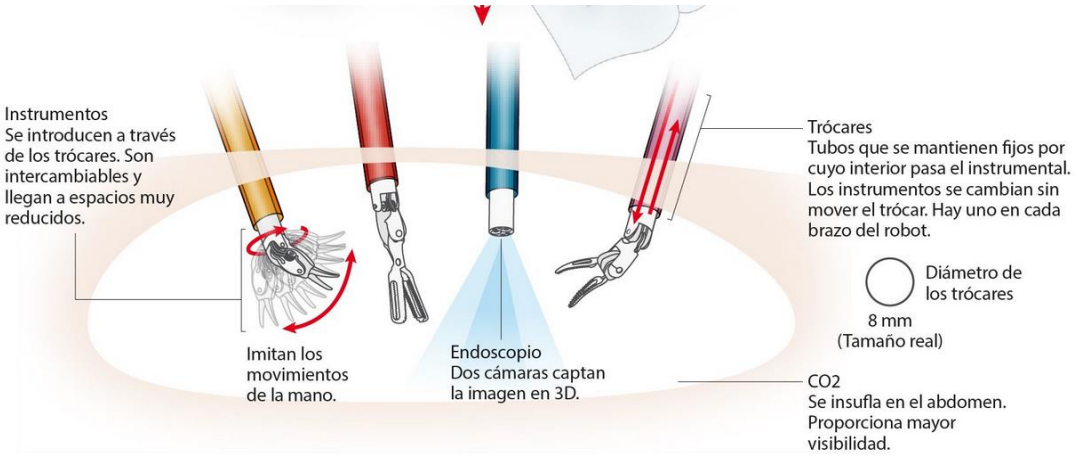

Figura 4 Herramientas robot Da Vinci

1. Aplicaciones del robot Da Vinci

Este sistema de cirugía es idóneo para aquellas cirugías más complejas y de difícil acceso. Las principales aplicaciones médicas del proyecto Da Vinci son las siguientes:

- Otorrinolaringología: Síndrome de apnea obstructiva del sueño, extracción de tiroides a través de la axila.
- Urología: prostatectomía radical, nefrectomía parcial reimplantación uteral pieloplastia
- Cirugía oncológica: Cirugía en colon y recto.
- Cirugía cardiotorácica: by-pass aorto-coronario, reparación de válvula tricúspide
- Ginecología: Histerectomía extirpación de tumores, y reconstrucciones rubricas
- Cirugía gástrica.

2. Desventajas de la cirugía robótica da Vinci

La principal desventaja se traduce en su elevado precio de compra, que está en torno a los 2 millones de euros que sumado al coste de las herramientas de material quirúrgico específicos y el mantenimiento supone un coste importante.

Otro de los principales inconvenientes de esta tecnología es la cantidad de tiempo requerido para que un cirujano aprenda y domine su uso. Se requiere de bastante práctica antes de que un cirujano pueda realizar cirugías con confianza.

Algunos de los otros inconvenientes están relacionados con su uso. Los cirujanos carecen sensación háptica (la capacidad de palpar los tejidos sometidos a la intervención) durante el uso de este dispositivo. Además, debido al gran espacio que ocupa el sistema robótico junto con el panel de mando, el sistema no es apto para todos los quirófanos.

1.2.2 El proyecto Broca:

El proyecto broca es una colaboración entre la Universidad de Córdoba y la Universidad de Málaga que da como resultado el desarrollo de un sistema de cirugía robótica de menor coste y más versátil que los sistemas comercializados actualmente.

Figura 5 Proyecto Broca

En concreto, el robot se compone de tres brazos que pueden operar tanto de forma coordinada como de forma individual y, por tanto, el cirujano podrá adaptarlo en función de las necesidades de cada intervención. Además, la estructura metálica que soporta a los brazos robóticos no es reducida, inalámbrica y fácil de mover, esencial en caso que la intervención se reconvierta a cirugía abierta.

Por otro lado, permitirá al cirujano operar sentado frente a una pantalla con visión 3D empleando simplemente unas gafas y no dentro de una consola. Los mandos de control del robot también estarán dotados de un sistema que emula sensaciones táctiles para ofrecer al cirujano información del estado de los tejidos operados. Esta característica aumenta la percepción sensorial del cirujano al operar.

Los sistemas de control también implementan sistemas que permiten al cirujano percibir la fuerza que se está aplicando al extremo del instrumental que está controlando remotamente pudiendo realizar la discriminación entre tejidos de diferente dureza.

Las ventaja principal que ofrece el este sistema es su menor tamaño, en comparación con el Da Vinci, por lo que no precisará de espacios adaptados ni de grandes habitáculos, y es más accesible a los quirófanos.

Otra ventaja que ofrece es su posibilidad de configuración usando uno o varios brazos con instrumental adecuado para cada tipo de especialidad quirúrgica respondiendo así a las necesidades particulares de cada hospital.

El sistema es compatible con un amplio abanico instrumental laparoscópico convencional, facilitando así el uso de instrumental esterilizarle y poder así reducir el coste por intervención.

Los brazos robóticos empleados son brazos robóticos colaborativos comerciales. En este caso los brazos empleados son de la marca "Universal Robots"

1.3. Desarrollo del proyecto de cirugía HALS.

En la Universidad de Valladolid, en colaboración con la Universidad Miguel Hernández de Elche y la Universidad de Málaga, se está desarrollando un sistema robótico colaborativo orientado a cirugía laparoscópica HALS (Hand Assisted Laparoscopic Surgery). En esta cirugía se practica una incisión en el paciente para que el cirujano pueda insertar la mano y palpar los tejidos y órganos. El robot que se utilizará portará un endoscopio que se insertará en el cuerpo del paciente, y cuya función será la iluminación de las herramientas utilizadas por el cirujano o de su propia mano.

La visión del interior de la cavidad abdominal se realizará con un sistema de visión 3D. Las imágenes tomadas por dicho sistema serán tratadas con algoritmos de visión artificial para obtener la posición del elemento a enfocar con el endoscopio y la posición actual del extremo del mismo.

Estos datos serán enviados al robot junto con el volumen que ocupa la mano del cirujano en la cavidad y la configuración actual de sus dedos. El controlador del robot determinará la trayectoria que deberá seguir el robot para no colisionar con la mano.

La obtención de la configuración de los ángulos de flexión de los dedos del cirujano se realizará con un guante de datos. La información obtenida del guante será utilizada para el tratamiento de gestos que el cirujano puede hacer para indicar al brazo robótico la acción a realizar.

Con el fin de poder simular el cuerpo del paciente, las pruebas se harán dentro de un pelvitrainer que simula la cavidad torácica del paciente y están específicamente diseñadas para el adiestramiento de los cirujanos.

1.4. Objetivos de este TFG

En el apartado anterior se han indicado las líneas generales del proyecto. En este trabajo fin de grado tratamos de realizar una primera aproximación de la comunicación, estableciendo un flujo de datos entre el guante de datos, el sistema de visión 3D y el robot ABB IRB 120. La información captada por el guante de datos y el sistema de visión será tratada para calcular una trayectoria del robot que permita realizar el seguimiento del dedo meñique de la mano del cirujano, sin colisionar con ella.

Para este TFG se han considerado tres partes:

- Sistema de visión 3D: Está compuesto por un juego de cámaras normales que al combinar sus imágenes se obtiene una imagen 3D. Este sistema se encarga de:
    - obtener los datos de posición de las puntas de los dedos (del cirujano) respecto del sistema de referencia del objetivo de una de las cámaras
    - identificar el tipo de volumen que adopta la mano.

  Estos datos del sistema de visión 3D son adquiridos por el ordenador.

- Guante de datos: En esta parte se controla la adquisición de datos del guante que llevará puesto en su mano el cirujano. El guante utilizado es el modelo 5DT Data Glove 14. Los datos son adquiridos por el ordenador, y servirán para el reconocimiento de gestos de la mano del cirujano. Entre estos gestos se incluye, órdenes específicas al robot así como servir al controlador como identificador de la fase de operación que se está realizando.

- Sistema robótico: El controlador del robot ABB IRB 120 se encarga de procesar la información que le llega, desde el ordenador, del sistema de visión 3D y del guante de datos, y calcula una trayectoria, sin colisiones, entre la posición en la que se encuentra el robot y la posición del dedo meñique. Esta trayectoria será ejecutada con el robot ABB.

De las tres partes anteriores el trabajo desarrollado en este TFG comprende la comunicación de los datos del sistema de visión 3D y de los datos del guante, almacenados en el ordenador, al robot, utilizando un socket TCP. El tratamiento de estos datos en el robot permite el cálculo online de la trayectoria, sin colisiones, que debe realizar el robot para seguir el dedo menique del cirujano (que lleva el guante de datos colocado en su mano).

Para no limitar el rango de movimientos posibles del robot en este TFG se ha decidido utilizar una pinza como herramienta de trabajo del robot, en lugar del endoscopio. También se ha decidido no imponer restricciones de movimientos, ni de puntos de fulcro.

2. Entorno de trabajo en el laboratorio.

2.1. Componentes del sistema

Una vez fijados los objetivos que debe cumplir este trabajo fin de grado. Se procede a explicar los componentes de la estación de trabajo. Se muestra en la imagen siguiente se muestra un esquema de los componentes del sistema, así como su conexión.

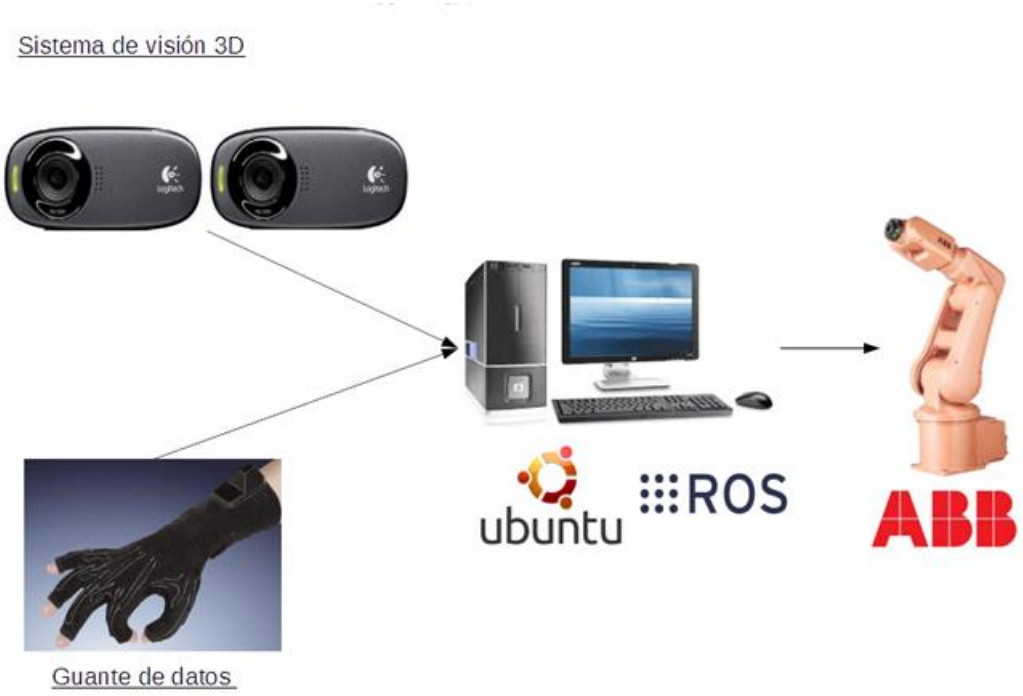

Figura 6 Esquema de componentes de la estación

La estación de trabajo está compuesta por los siguientes equipos:

1. Sistema de visión 3D. El sistema de visión 3D, formado por dos cámaras de visión que combinan sus imágenes para obtener una imagen en tres dimensiones. El modelo de cámaras utilizadas han sido unas Logitech modelo C310

2. **Guante de datos:** Con el guante de datos adquirimos los ángulos de flexión de los dedos de la mano del cirujano. El modelo utilizado es: "5DT Data Glove 14 Ultra Left" de la marca 5DT.

3. **Sistema robótico:** El robot ABB IRB 120 se comportará como el actuador del sistema. Será el encargado de recibir la información de la posición de la mano y su volumen característico e calculará las trayectorias correspondientes para evitar colisionar con la mano del cirujano.

4. **Ordenador:** El ordenador contendrá todas las rutinas software de adquisición de datos del sistema de visión 3D y del guante. Simultáneamente el ordenador clasificará y enviará al robot los datos adquiridos por medio de un socket TCP. El sistema operativo utilizado será Ubuntu 14.04.

   Para simplificar los procesos de comunicación de datos entre los distintos procesos se ha utilizado el Framework ROS y sus rutinas desarrolladas bajo el modelo publicador subscriptor.

Una vez definidos los componentes del sistema, mostramos la estación de trabajo así como la ubicación de los componentes.

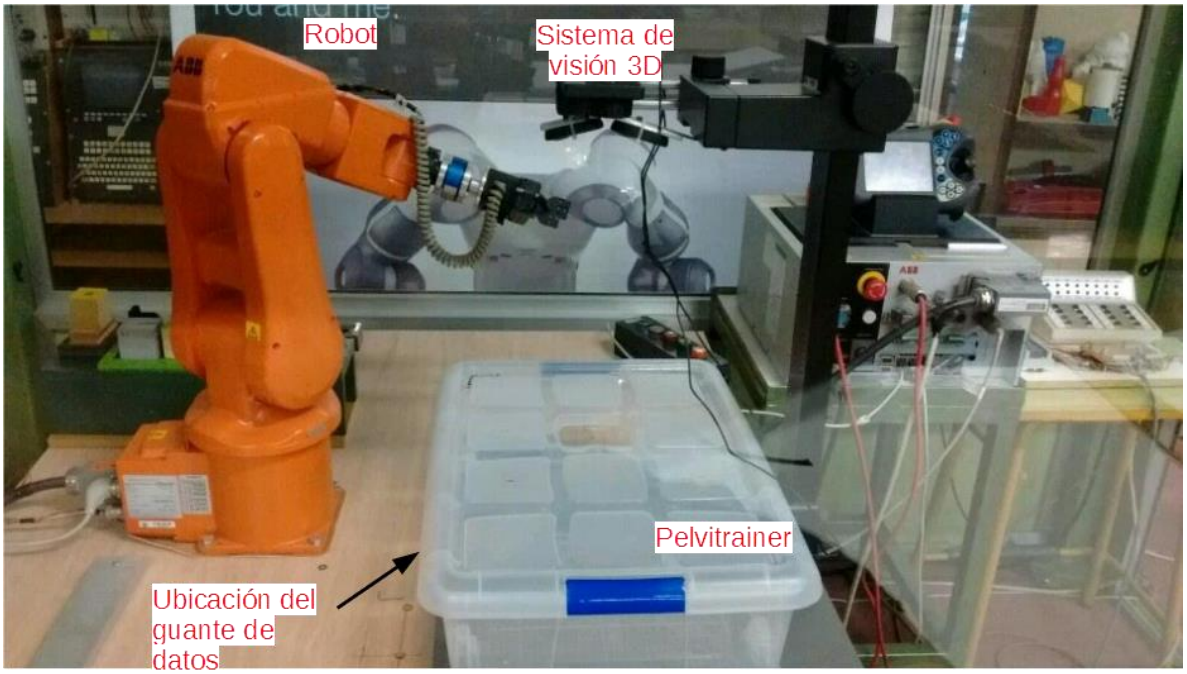

Figura 7 Estación de trabajo

2.2. Ubicaciones de los sistemas de referencia

Durante todo el desarrollo de este trabajo fin de grado existen varios sistemas de referencia respecto a los cuales estarán referidos los datos. Explicaremos a continuación su cometido, los datos que dependen de ellos y su ubicación respecto del sistema de coordenadas origen del robot.

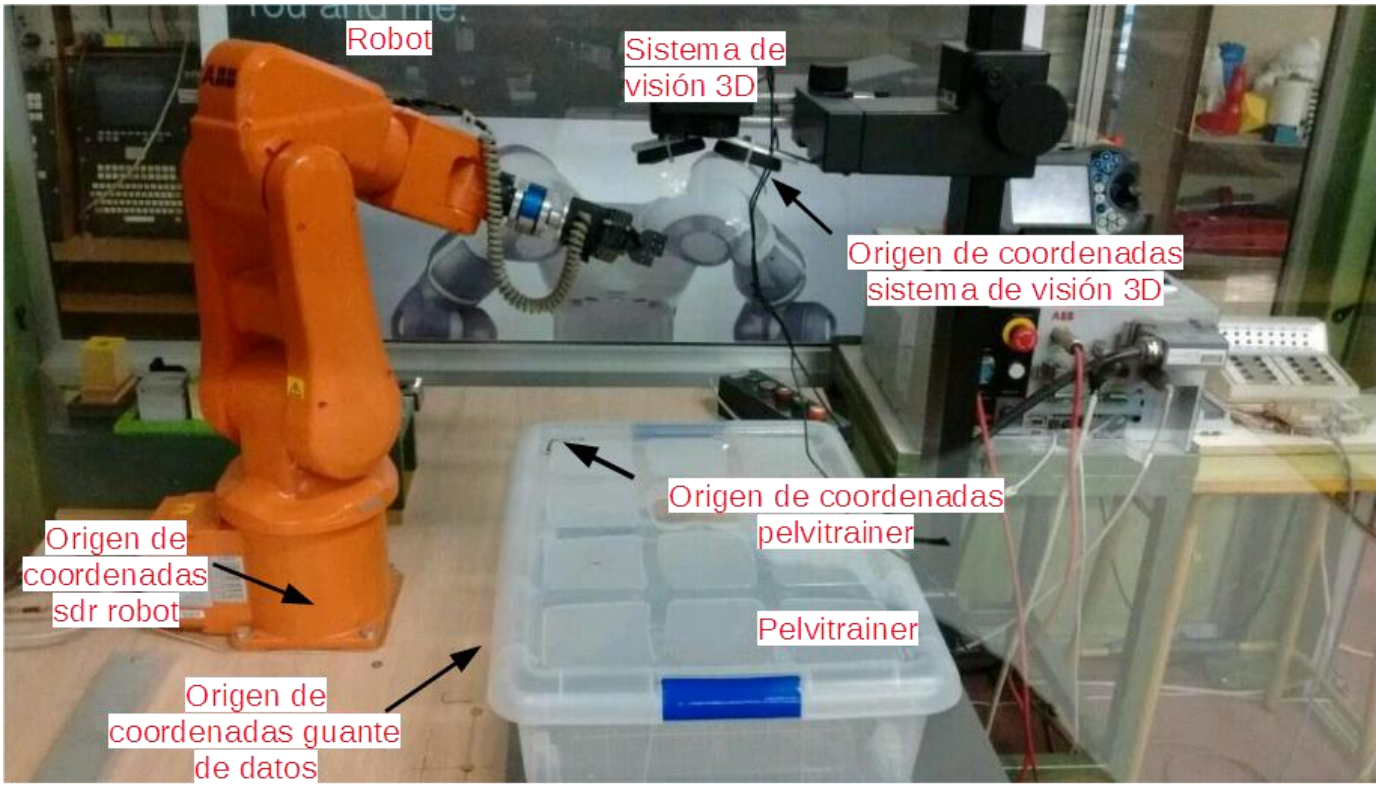

Figura 8 Ubicación de los sistemas de referencia en la estación real

Nota: La orientación de los sistemas de referencia en su conjunto se muestra en la Figura 13, donde se muestra una simulación 3D de la estación de trabajo real.

- Robot: El origen de coordenadas del sistema de referencia del robot se ubica en el centro de la base del robot. En la siguiente figura podemos apreciar una imagen del robot con su sistema de referencia.

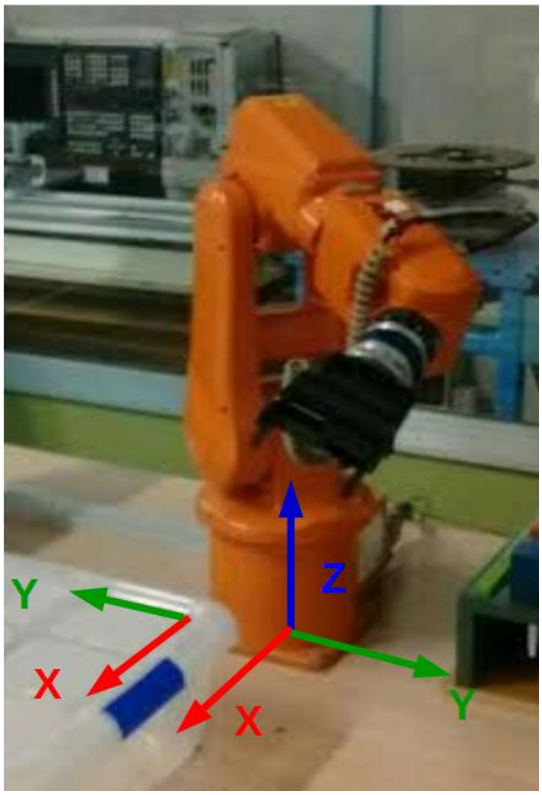

Figura 9 Orientación del sistema de referencia robot

- <u>Sistema de visión 3d:</u> El sistema de referencia establece la posición de las cámaras de visión respecto del sistema de referencia robot El origen del sistema de referencia cámara para este proyecto se ha considerado el punto 480,-213, 275 en coordenadas XYZ respectivamente. En la Figura 10 se muestra una imagen de la ubicación y orientación del origen de coordenadas del sistema de visión 3D.

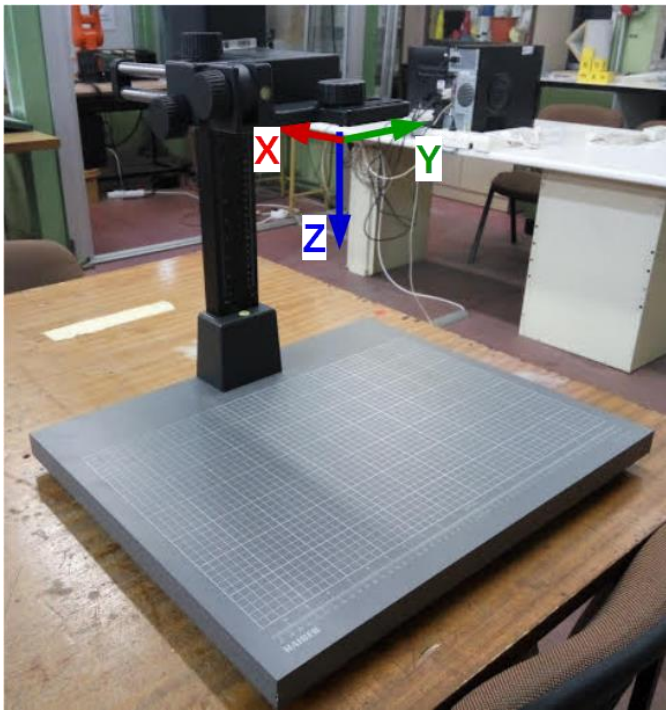

Figura 10 Orientación sistema de visión 3D

- **Pelvitrainer:** Todos los movimientos de desplazamiento del robot serán referenciados respecto a este sistema de referencia. Este sistema posee la misma orientación que el sistema de visión 3D. Sin embargo, su ubicación para este trabajo fin de grado estará en las coordenadas [260,27,175] XYZ respectivamente. En la Figura 9 se muestra el origen de coordenadas del pelvitrainer respecto del robot. Y en la figura Figura 11 se muestra el mismo origen de coordenadas junto al origen de coordenadas del guante de datos.

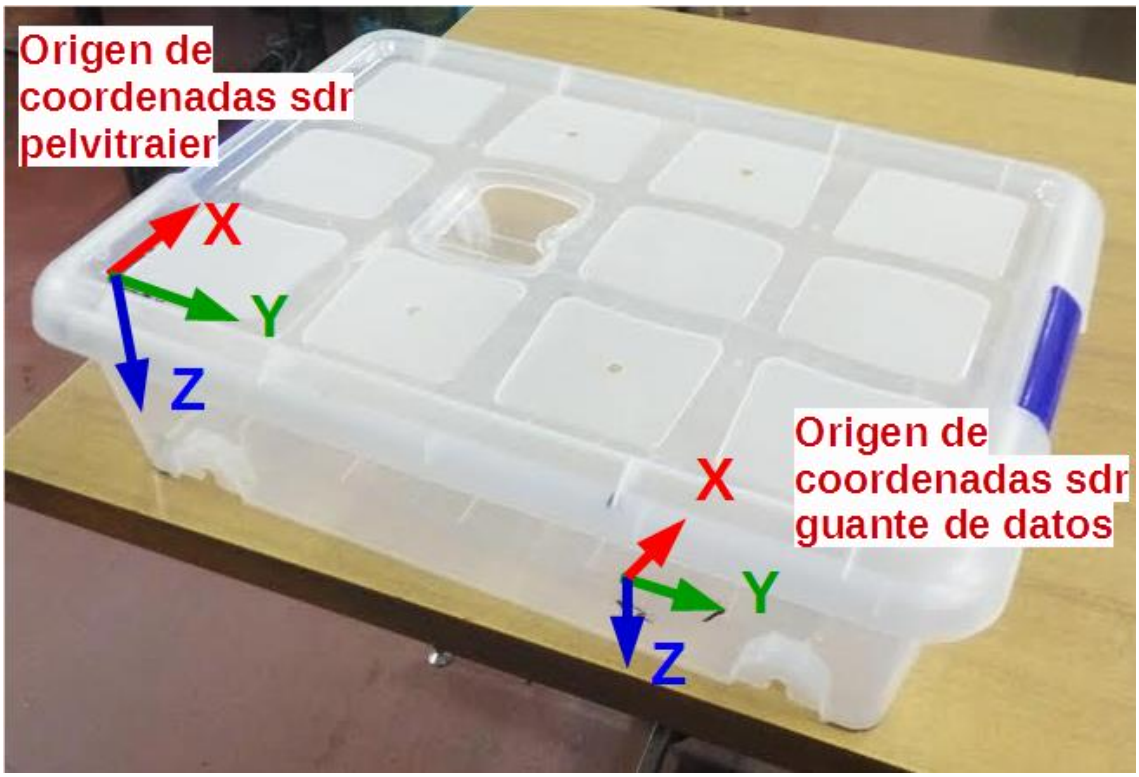

Figura 11 Orígenes de coordenadas de los sdr pelvitrainer y guante de datos

- <u>Guante de datos:</u> representa el punto de inserción de la mano del cirujano en el cuerpo del paciente. Este sistema de referencia se considera fijo en el estado de desarrollo actual del proyecto pero en un futuro podrá variar en función del cuerpo del paciente. El punto de origen de coordenadas se encuentra en el centro de la muñeca por su lado interior. En la Figura 11 se muestra el origen de coordenadas del sistema de referencia en el pelvitrainer.

  La importancia del sistema de referencia mano nos permite ubicar el volumen de zonas alcanzables por la mano del cirujano respecto del sistema de coordenadas del robot.

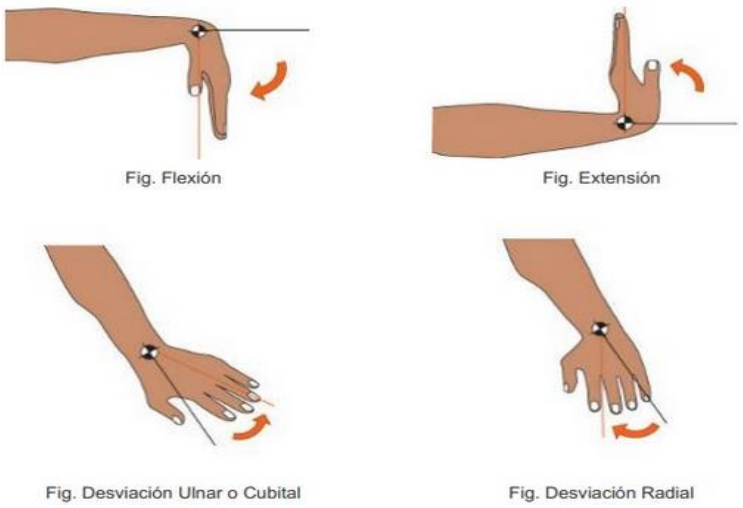

Figura 12 Movimientos de la muñeca

El sistema de referencia guante de datos también delimita el volumen de trabajo del cirujano. Dicho volumen se define idealmente por el barrido con la mano extendida de las desviaciones cubital y radial desde que la muñeca está en posición extendida hasta que la muñeca pasa a estar flexionada (ver aclaración en la Figura 12.

En la Tabla 1 se muestra la ubicación concreta de los sistemas de referencia y su posición respecto al sistema de referencia robot.

Sistema de referencia	Ubicación respecto del sistema de referencia robot	Nomenclatura utilizada en el código
Pelvitrainer	[260,27,175],[0,1,0,0]	Pelvitrainer
Sistema de visión 3D	[480,-213,275],[0,1,0,0]	Cámara
Guante de datos	[320,-500,87.5],[0,1,0,0]	Mano

Tabla 1 Ubicación concreta de los sistemas de referencia

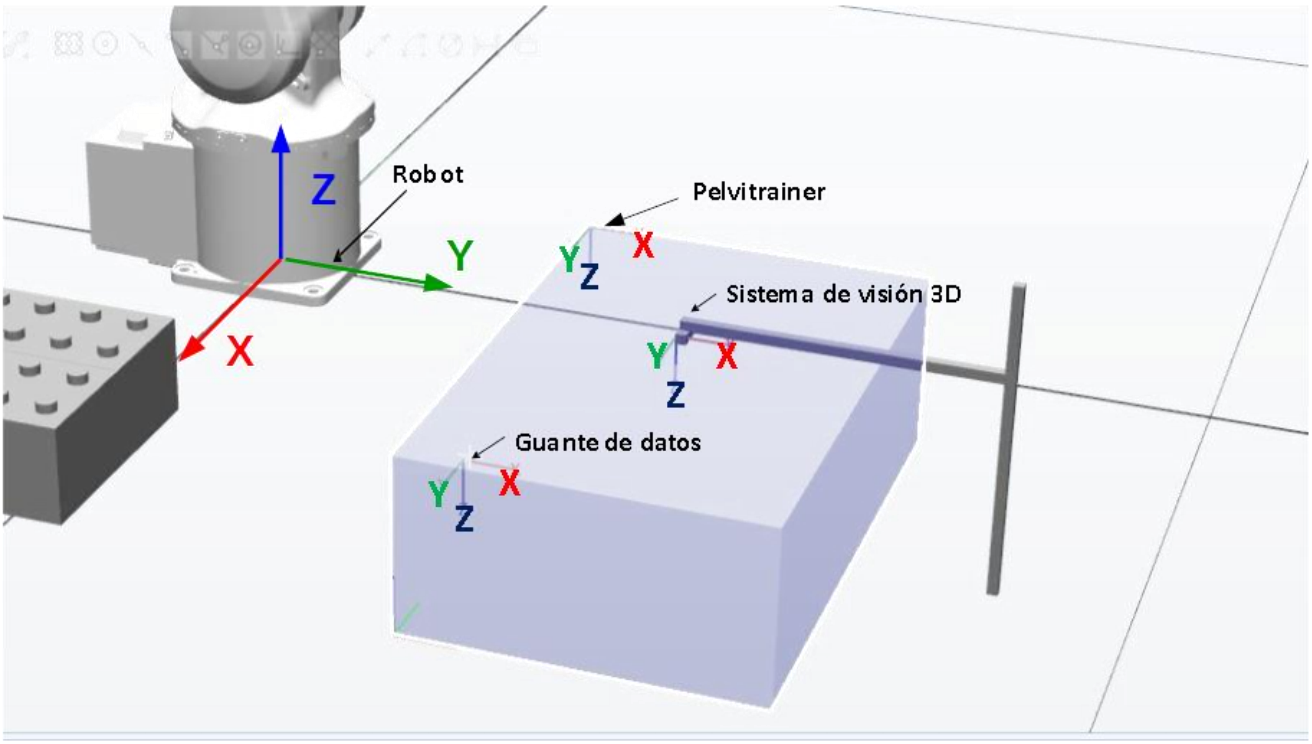

Figura 13 ubicación y orientación de los sistemas de referencia de la estación

2.3. Datos a enviar al robot

Una vez se han fijado los componentes de la estación de trabajo, el siguiente paso es conocer qué datos serán adquiridos por los sistemas de visión y del guante. Estos datos se clasificaran y encapsularan para su envío al robot.

Los datos que captan los sistemas de adquisición y deben ser mandados al robot aparecen contenidos en la siguiente tabla:

DEDOS			VOLUMEN		
Ángulos de flexión dedos	Ángulos de separación entre dedos	Posición de las Puntas	Tipo esfera	Tipo paralelepípedo	Dimensiones constantes de la mano
pulgar1	pulgarindice	pulgar_x	tipo	tipo	ancho_paral
pulgar2	indicecorazon	pulgar_y	x_centro	pax	largo_paral
indice1	corazonanular	pulgar_z	y_centro	pay	alto_paral
indice2	anularmenique	indice_x	z_centro	paz	
corazon1		indice_y	radio	pbx	
corazon2		indice_z		pby	
anular1		corazon_x		pbz	
anular2		corazon_y			
menique1		corazon_z			
menique2		anular_x			
		anular_y			
		anular_z			
		menique_x			
		menique_y			
		menique_z			

Tabla 2 Datos a enviar al robot

Una vez los datos llegan al robot, son almacenados en la memoria para su posterior uso. Se procede a explicar más en detalle los tipos de datos contenidos en esta tabla:

- <u>Ángulos de flexión de los dedos:</u>

  Los ángulos de flexión de los dedos se corresponden con los ángulos formados entre los metacarpianos y las falanges proximales y estas con las falanges medias. La variable que contendrá los datos del primer ángulo se denominará con el nombre del dedo al que nos referimos más el sufijo 1. El otro llevará el sufijo 2.

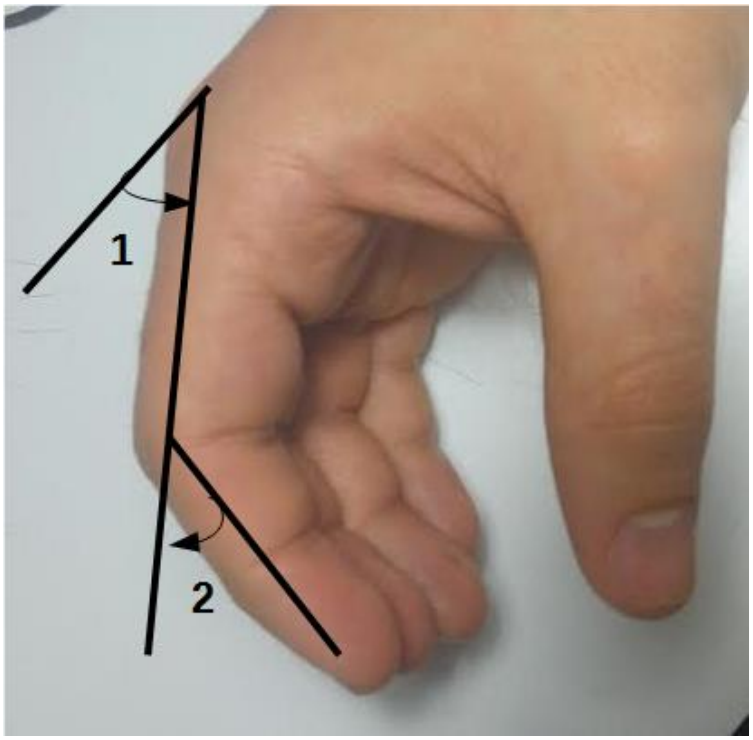

Figura 14 Representación de los ángulos 1 y 2 del dedo índice

Existe un tercer ángulo formado entre las falanges medias y las distales pero por simplicidad de modelado no se considera. Otro motivo por el cual no consideramos el ángulo anterior se debe a que mantiene una proporción directa con el ángulo formado por las falanges proximales y medias (ángulo 2 de la figura anterior).

- <u>Ángulos de separación entre dedos:</u>
  Estos ángulos se determinan por el ángulo existente entre las falanges proximales de dos dedos. El nombre de la variable que contendrá la información llevará los dos nombres de los dedos implicados. Así pues, en el caso de la siguiente figura el ángulo formado por el meñique y el anular estará almacenado en la variable anularmenique.

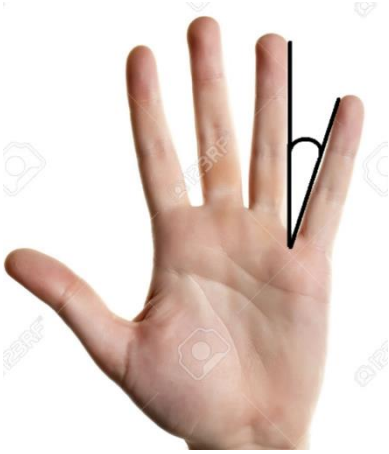

Figura 15 ángulo anular meñique

- <u>Posición puntas de los dedos:</u>
  La posición de las puntas de los dedos será transmitida al robot, mediante tres datos correspondientes a las coordenadas XYZ de la puntas de los dedos respecto al sistema de visión. Cada coordenada de cada dedo está representada por el dedo al que pertenece más el sufijo X, Y o Z. Todas ellas aparecen representadas en la "Tabla 2", en la columna "Posición de las puntas". Estos datos son de vital importancia para el control de trayectorias del robot.

- <u>Volúmenes adoptados por la mano:</u>

  Por otra parte, tenemos los datos correspondientes al volumen ocupado por la mano del cirujano y su posición en el espacio. Estos datos los trataremos para que el robot nunca entre en dicho volumen y pueda lastimar la mano del cirujano, o restringir sus movimientos.

  Dentro de los posibles volúmenes que puede tomar la mano del cirujano para nuestro caso, y por simplicidad de tratamiento consideramos dos casos posibles:

  - Un paralelepípedo en el caso que no haya ninguna articulación flexionada.
  - Una esfera en caso de existir flexión en las articulaciones.

Figura 16 Configuración de la mano en posición de paralelepípedo y en posición esfera

El tipo de volumen simplificado será contenido en la variable de tipo entera (int) cuyo contenido será 1 para el volumen tipo esfera o 2 para el tipo paralelepípedo.

Volumen tipo esfera: Es el volumen de tratamiento más sencillo, está compuesto a su vez por tres datos que contienen la posición de cada una de las coordenadas del centro de la esfera, que quedará determinado por la articulación formada por el metacarpiano y la falange proximal.

El radio de la esfera se determina en función de la mano del cirujano al calcular la distancia desde el centro de la esfera hasta la punta del dedo pulgar cuando la mano adopta la configuración mostrada en la siguiente imagen:

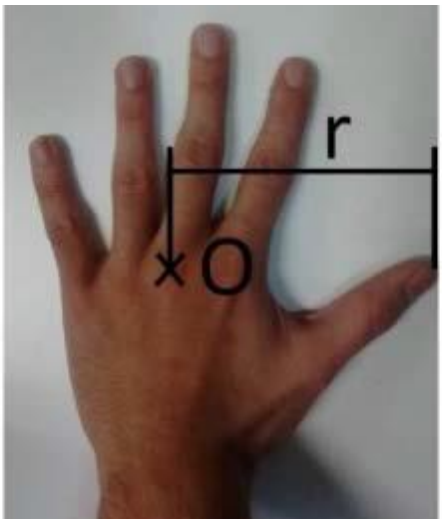

Figura 17 Ubicación origen esfera y radio

Volumen tipo paralelepípedo: Este tipo de volumen se compone por 2 posiciones que determinan la diagonal mayor del cuerpo y se denominan

"pa" y *"pb"*. A su vez, estas dos posiciones se corresponden con tres datos cada una que determinan la posición en coordenadas XYZ respecto del sistema de visión (ver Tabla 2 columna "tipo paralelepípedo").

Por último también se transmiten las dimensiones del paralelepípedo: largo, ancho y alto. Estas aparecen representadas en la columna "dimensiones constantes de la mano" de la Tabla 2.

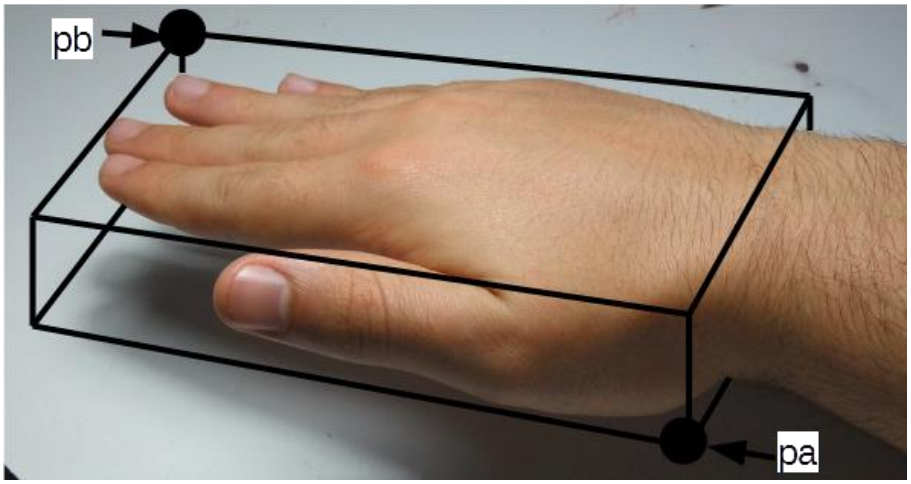

Figura 18 Representación de los puntos pa y pb del paralelepípedo

Las posiciones que definen los volúmenes simplificados de la mano están referidos al sistema de visión 3D.

2.4. Estructuración de los procesos.

En este apartado se trata de dar una visión general a los procesos que se realizan para llevar a cabo el envío de la información hacia el robot y conseguir el movimiento de este.

Los procesos de adquisición de datos del sistema de visión 3D y del guante de datos son ajenos a este TFG. Con el fin de esquematizarlo de forma simplificada agruparemos todas estas tareas en una tarea mayor llamada adquisición de datos. Estas tareas cumplen con la función de poner la información a disposición de terceras tareas. Se utilizará el framework ROS (para más información ver apartado 3.1) como medio de transmisión de la información.

Una vez que los datos están siendo adquiridos correctamente un segundo proceso se encargará de agrupar los datos en función a su contenido (Nodo *dealer_all*) para ser enviado a una tarea de envío u otra (ver Figura 19 Tareas entre "sistema visión-guante" y ordenador Figura 19). Para simplificar los procesos de transmisión de datos entre las tareas, la agrupación de los datos se ha realizado en el código en forma de dos mensajes *.msg*. Estos mensajes como se verá en el apartado 3.1 son

estructuras de datos de diverso tipo que son transmitidas a través de un único publicador. La principal razón de elegir concentrar todo los datos en un mensaje *msg* frente a realizar el envío individual de los datos es debida a la mejora en la legibilidad del código.

La agrupación se hará teniendo en cuenta dos categorías:

- Datos principales para el control de trayectoria: Los datos a enviar engloban al tipo de volumen que adopta la mano del cirujano, los datos para definir su ubicación espacial y las coordenadas XYZ de la punta de cada dedo. Todos los datos descritos se pueden encontrar en la Tabla 2 en las columnas "posición puntas dedos", "volumen tipo esfera" y "volumen tipo paralelepípedo"
  Los datos que definen el volumen de la mano, serán utilizados para definir zonas de exclusión donde el robot no podrá acceder. La nomenclatura dada en el código a esta agrupación de datos es *mensaje ez*. Siendo *ez* el acrónimo de "exclusión-zone".
- Datos secundarios: Los datos restantes se corresponden con las dimensiones constantes de la mano del cirujano cuando esta toma la forma de paralelepípedo y de los ángulos de flexión y separación de los dedos. Estos datos se corresponden con el resto de columnas de la Tabla 2.
  La nomenclatura dada en el código a este tipo de mensaje se denomina *mensaje hp*. Siendo *hp* el acrónimo de "hand parameters"

Tras esta agrupación, los datos son enviados a los procesos encargados de los envíos al robot.

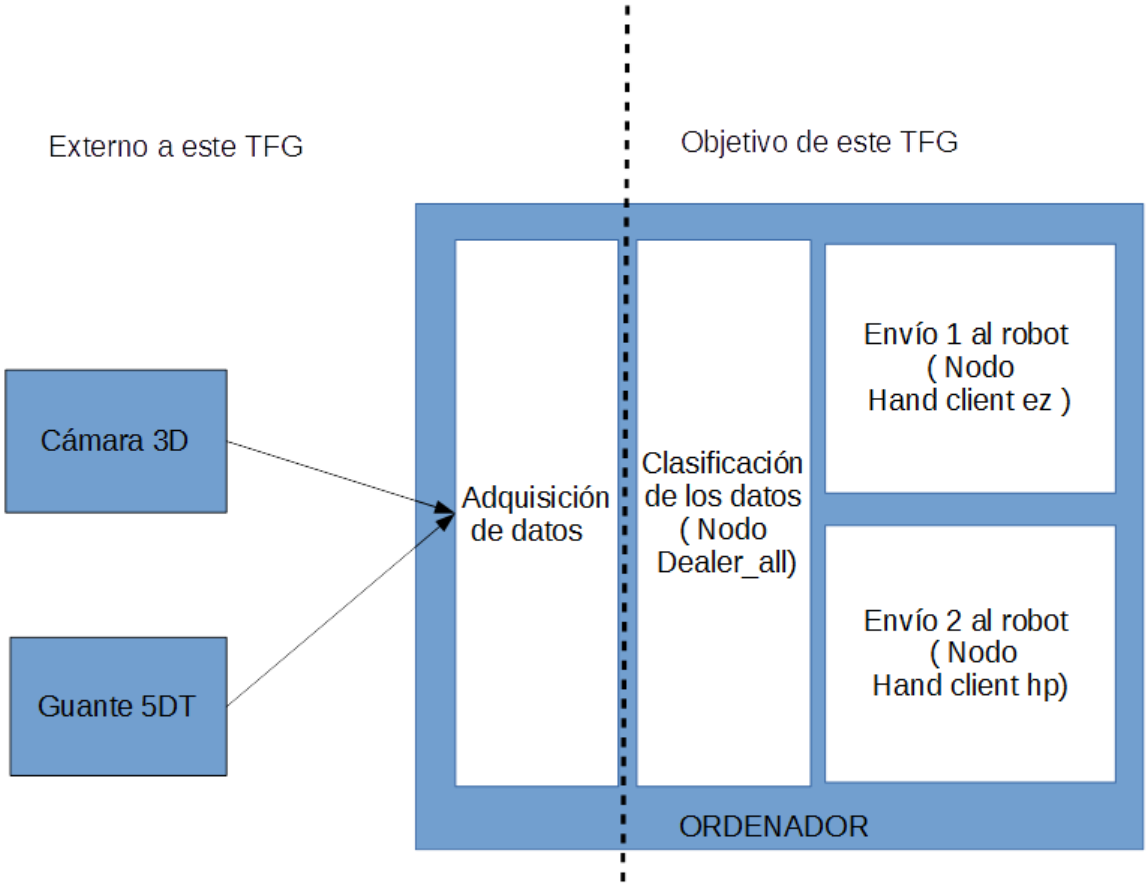

Figura 19 Tareas entre "sistema visión-guante" y ordenador

Las tareas de envío de información al robot se encargan de obtener la información a enviar, serializarla correctamente y transmitirla a través de un socket TCP a las tareas receptoras del Robot.

Nota 1: Utilizaremos "nodo" para referirnos a un proceso del ordenador que se ejecuta bajo el Framework ROS.

Nota2: Utilizaremos "tarea" para referimos a un proceso ejecutado en el robot.

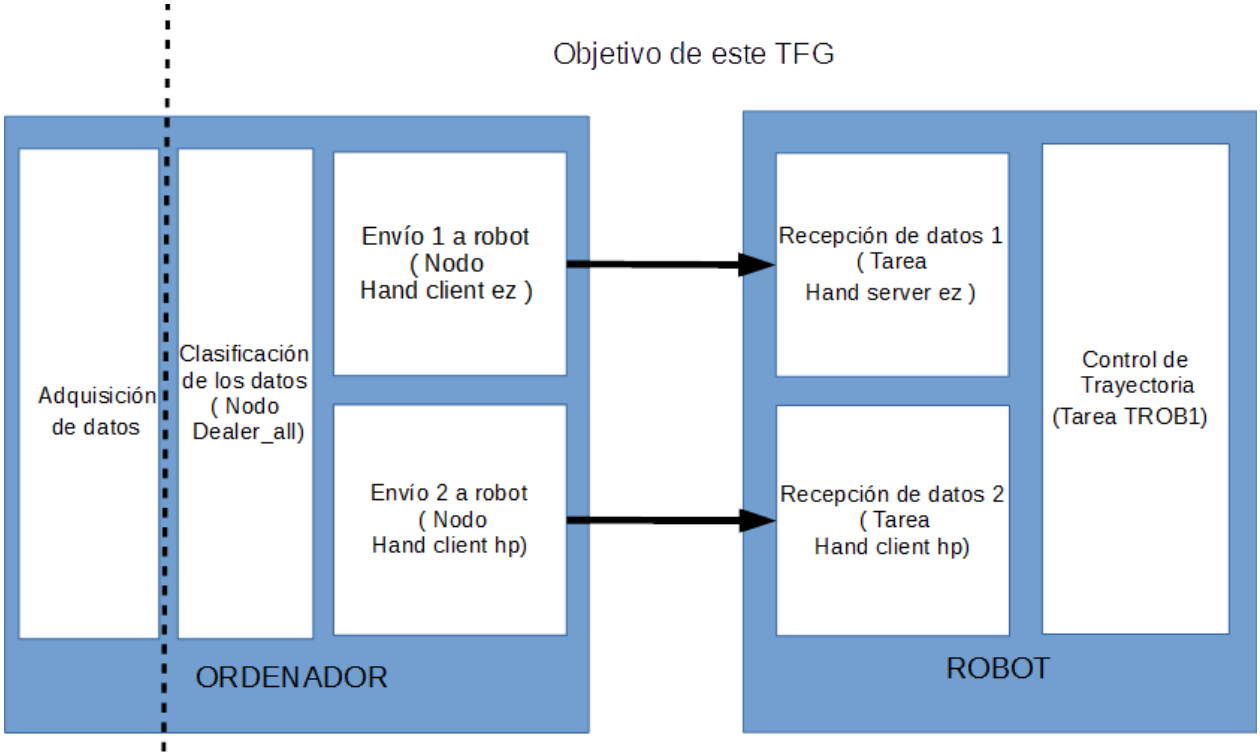

Figura 20 Tareas entre ordenador y robot.

Tras la llegada de la información a las tareas receptoras del robot, se produce el almacenamiento de la misma. No obstante, determinados datos (ver Tabla 3) son cambiados de sistema de referencia antes de ser almacenados para su posterior uso en el control de trayectorias

Datos	SDR origen	SDR destino
Coordenadas que definen el volumen y la ubicación de la mano	SDR visión 3D	SDR Robot
Punto objetivo	SDR visión 3D	SDR Pelvitrainer

Tabla 3 Datos que cambian su SDR

El punto objetivo será la posición de uno de los dedos. Por defecto, en este trabajo fin de grado se ha considerado la posición del dedo meñique. Para más información ver el apartado 2.1.2 del capítulo 3.

Nota: Para lograr una mejor comprensión de lo aquí descrito se recomienda consultar el “Diagrama de flujo de datos del sistema” Contenido en la segunda página del anexo 2. “Diagramas”.

3. Gestión de tareas en el ordenador

En este capítulo nos centraremos en cómo se realiza la comunicación entre tareas en el ordenador. Para ello, explicaremos como funciona una tarea en ROS y como se realiza la comunicación entre nodos, entendiendo por nodos tareas de ROS. Tras esto, profundizaremos en el funcionamiento de los nodos de clasificación de datos y envío al robot.

3.1. ROS

ROS es el acrónimo de *Robot Operative System*, un entorno de trabajo (framework) de código libre para desarrollar software para robots. Incluye entre sus herramientas, programas, conjuntos de librerías y estándares con el fin de simplificar el desarrollo de entornos robotizados sea cuales sean sus características.

La aparición de ROS supuso un importante avance en el desarrollo de la actual generación de robots colaborativos y de enjambre. Estos últimos basados en la colaboración de multitud de robots.

Para el correcto desarrollo de estas aplicaciones el ordenador deberá incorporar un sistema operativo basado en UNIX, en particular se ha elegido Ubuntu 14.04 y la versión *Indigo igloo* de ROS que es la única versión LTS (*Long Time support*) para este sistema operativo.

Utilizaremos la nomenclatura nodo referirnos a una tarea o proceso de ROS

3.1.1 Funcionamiento de un programa en ROS:

El funcionamiento de una rutina en ROS es similar al programa que puede incluir cualquier microcontrolador. Tendrá en el nodo principal un parte de definición de entradas, salidas, puertos y variables internas, y acto seguido un bucle infinito donde contendrá el cuerpo del programa, en el cual, podemos controlar los ciclos por segundo cambiando la instrucción “loop rate” al final del lazo de ejecución.

3.1.2 Desarrollo de la comunicación entre nodos en ROS:

El primer paso que debe cumplir los nodos de este TFG consiste en clasificar los datos a enviar. Estos datos son proporcionados por otros nodos de ROS y la forma de comunicación entre los nodos es mediante el uso de modelos publicador subscriptor.

El modelo publicador suscriptor incluido en las librerías de ROS permite establecer una cola de mensajes unidireccional (salvo en el caso de utilizar mensajes tipo SRV), donde el publicador será el único nodo con acceso de escritura a dicha cola. Mientras que el, o los suscriptores tienen derecho de lectura de la información. Los

tipos de datos que es posible publicar están recogidos en la librería de ROS *std_msgs* (/opt/ros/"ros_ditro"/share/) donde aparecen los tipos de datos principales permitidos por ROS.

Por el contrario, si el desarrollador desea publicar y recibir más de un tipo de dato por el mismo canal, deberá desarrollar un mensaje tipo MSG o SRV (este último, es utilizado cuando requerimos información al subscriptor). Ambos tipos se componen de tipos de datos principales formando un objeto que será el que se transmita a través de la cola de mensajes.

Los mensajes MSG y SRV se crean a partir de ficheros de texto editados por el desarrollador y que se utilizarán como código fuente para desarrollar nuevos tipos de datos (compuestos por campos de tipos de datos básicos) en diferentes lenguajes.

La diferencia que existe entre los mensajes msg y los srv radica en que estos últimos incluyen una sección de datos que deberá ser editada por el suscriptor y devuelta al publicador (permite el flujo de datos bidireccional).

El contenido de un fichero típico puede ser el siguiente:

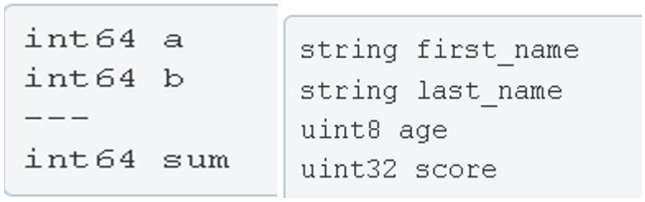

Figura 21 Ejemplo de mensajes

En la figura anterior, la parte de la izquierda se corresponde con la estructura típica de un mensaje tipo *msg*. En el lado derecho se muestra un mensaje tipo SRV donde la parte inferior de la separación con guiones (---) se corresponde con los campos del mensaje que debe editar el subscriptor.

Para que los ficheros de texto sean reconocidos como mensajes deben estar ubicados en la carpeta *msg*, dentro del directorio del paquete actual.

3.2. Clasificación de los datos.

Una vez conocidos los datos a enviar a través de nuestra arquitectura de comunicación (apartado 2.3), debemos conocer el flujo que siguen estos y como se distribuyen hasta que se envían a través del socket al robot.

Los datos son recibidos a través de suscriptores en el nodo "Dealer_all", cuyo fin, es agrupar los datos en objetos definidos por ficheros *.msg* en función a sus características. La clasificación tiene que ver, a su vez, con la criticidad de los datos para el sistema. Así pues, como se detalló en el apartado 2.4 los datos a enviar se agrupan en dos categorías: datos principales y secundarios.

Una vez agrupados, se enviarán a través de publicadores a dos nodos receptores distintos, que se encargarán de gestionar el envío al robot a través del socket.

El esquema principal del nodo "Dealer all" se encuentra en la Figura 22

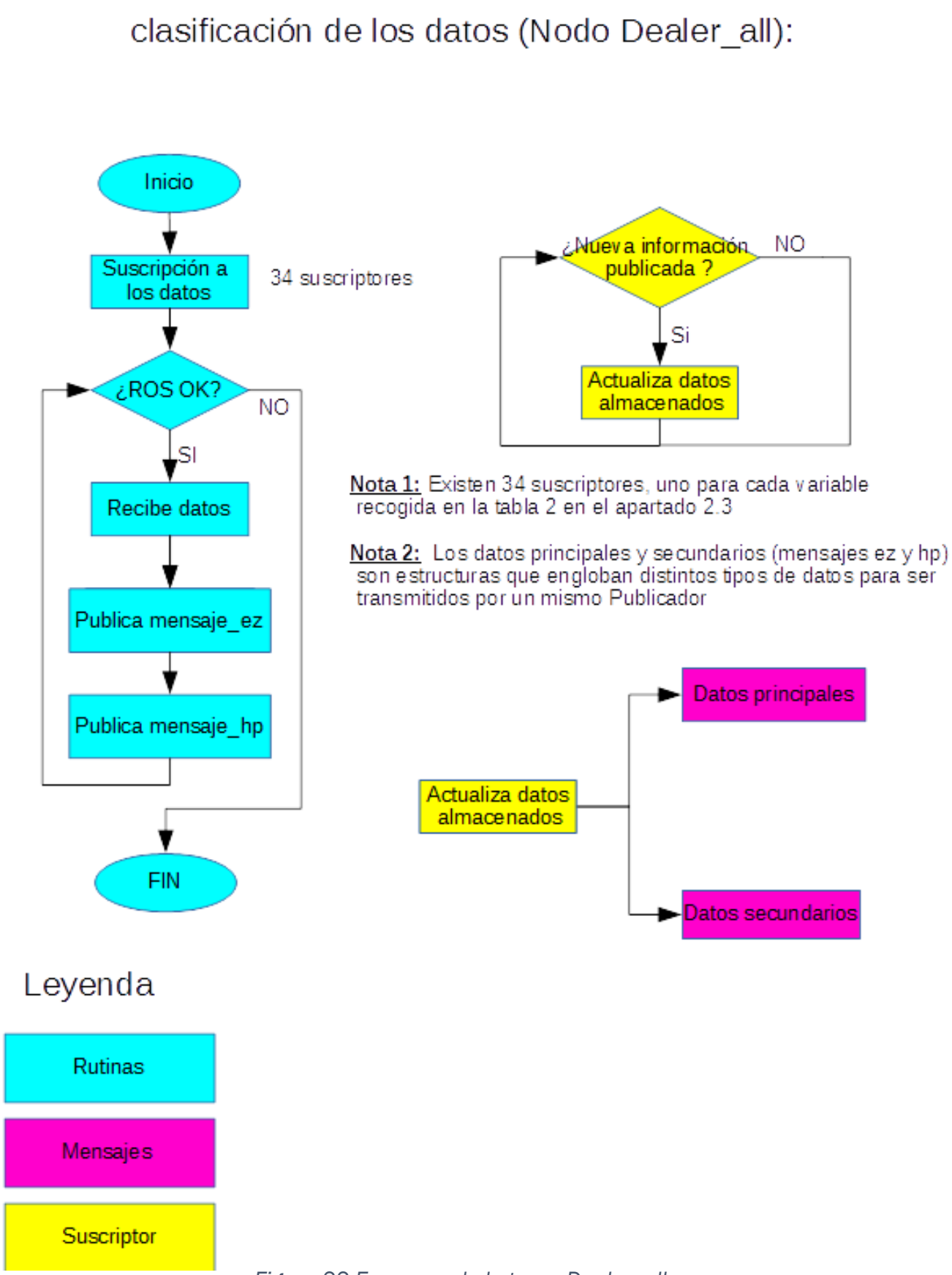

Figura 22 Esquema de la tarea Dealer_all

Una vez conocida la función de cada nodo, se procede a explicar más en detalle los nodos de envío de datos al robot.

3.3. Nodo Hand Client EZ

El nodo hand_client_EZ representado en la Figura 19 y la Figura 20 como envío 1 al robot es una de las dos tareas encargadas de transmitir la información desde el ordenador hasta el robot. Esta tarea es encargada de transmitir los datos principales definidos en el apartado 2.4

El esquema del funcionamiento del nodo aparare representado en la Figura 23

En primer lugar, la tarea crea un socket TCP con el que establecerá comunicación con la tarea receptora del robot. Para este caso, se ha considerado el robot como servidor, siendo su IP la 157.88.201.130 y el puerto utilizado para esta conexión el 80. Una vez la conexión socket es establecida correctamente, se continúa habilitado la comunicación con la tarea Dealer_all de la que recibirá los datos a enviar al robot. Esta comunicación se realiza suscribiéndose al publicador de datos principales, representado en el código como mensaje_ez.

3.3.1. Tratamiento de datos previo al envío:

Tras recibir los datos, estos se vuelven a clasificar en función a su cometido para generar dos tipos de mensajes distintos a enviar a través del mismo socket. Así pues, para este nodo se obtienen dos familias de datos con las que formaremos dos tipos de mensajes distintos:

- Los posición de la punta de los dedos: Representados en la columna “Posición de las puntas” de la Tabla 2 del apartado 2.4.
- Los referidos al volumen ocupado por la mano del cirujano: Representados en las columnas “Tipo esfera” y “tipo paralelepípedo” Tabla 2 del apartado 2.4.

Esta segunda clasificación se ha realizado con el fin de reducir el tamaño de los mensajes a enviar al robot. Este detalle se debe tener en cuenta ya que Rapid impone restricciones en cuanto a la longitud del mensaje a recibir por las tareas del robot.

Antes de continuar se debe tener en cuenta que los datos obtenidos de los sistemas de visión 3D y del guante de datos presentan cierta variabilidad. Esta variabilidad se debe a diversos factores como la resolución de las cámaras utilizadas, los factores ambientales del entorno y el hecho de cuantificar una medida digitalmente.

Teniendo en cuenta todo esto, y pensando en optimizar recursos tanto del ordenador como del controlador del robot se ha decidido imponer un cierto intervalo de tolerancia para reducir el número de envío de datos al robot. Si la diferencia de cualquiera de los datos recibidos del nodo dealer_all con los últimos datos enviados al robot son menores que el intervalo de tolerancia fijado los datos no se enviarán. En el caso que alguno de los datos recibidos cuya diferencia entre el actual y el último enviado sea mayor que el intervalo de tolerancia se enviarán todos los datos al robot.

En este nodo el intervalo de tolerancia escogido ha sido 1 mm.

La comprobación de la tolerancia se realizara dos veces por ciclo de nodo en ROS, una para cada agrupación de datos. Salvo en el caso que sea la primera vez que se ejecuta el lazo del nodo. En ese caso, ambos mensajes se enviarán al robot.

3.3.2. Composición de los mensajes:

Si el nodo finalmente decide enviar los datos al robot, pasará estos a una subrutina que compondrá el mensaje a enviar por el socket al robot.

Características generales:

El mensaje con todos los datos enviados al robot a través del socket será de tipo “String”. En él se incluirán un primer campo (cabecera 1) que indique el tipo datos enviado al robot. El resto de campos del mensaje serán completados en función del tipo de datos.

Las razones por las que se ha elegido enviar los datos contenidos en un mensaje tipo string se deben a:

- La facilidad de comunicación entre el robot y ROS que ofrece este tipo de datos.
- La ventaja de utilizar caracteres especiales como separadores de datos. En nuestro caso, se ha utilizado el carácter asterisco “*” como separador entre datos.

Por el contrario, el principal hándicap de este método es la longitud máxima del “String” a recibir por el Robot. En nuestro caso Rapid (lenguaje de programación de ABB) permite como máximo un “Sting” de 80 caracteres.

Durante la composición del mensaje se realiza una conversión de todas las variables de tipo numérico recibidas al tipo String. En este punto, se realiza la agrupación de las componentes de los vectores de posición recibidos individualmente del subscriptor para formar vectores.

Este proceso se realiza insertando los corchetes y comas respectivas para quedar correctamente definido en Rapid. Es decir, insertaremos un corchete inicial para determinar el inicio de un vector e insertaremos las componentes XYZ

respectivamente separadas por comas. Tras el último dato, se inserta el corchete de cierre. Los vectores resultantes clasificados en función a su tipo se muestran en la Tabla 4 y la Tabla 6.

Posición punta de los dedos:

En caso que el mensaje contenga información sobre las puntas de los dedos el primer campo contendrá la palabra "dedo". Los 5 campos siguientes se compondrán con los vectores de las posiciones de la puntas de los dedos en orden desde el pulgar hasta el meñique, tal y como se muestra en la Tabla 5.

Vector resultante	Componentes						
Pulgar	[	Pulgar_X	,	Pulgar_Y	,	Pulgar_Z	]
Índice	[	Indice_X	,	Indice_Y	,	Indice_Z	]
Corazón	[	Corazon_X	,	Corazon_Y	,	Corazon_Z	]
Anular	[	Anular_X	,	Anular_Y	,	Anular_Z	]
Meñique	[	Menique_X	,	Menique_Y	,	Menique_Z	]

Tabla 4 Vectores de posición de la punta de los dedos

Cabecera 1	Punto1	Punto2	Punto3	Punto4	Punto5	Objetivo
"dedo"	Pulgar	Índice	Corazón	Anular	Meñique	"men"

Tabla 5 Mensaje con la información de las puntas de los dedos

Por último, el campo objetivo contendrá las iniciales del dedo que deberá seguir el robot. Así pues, para el seguimiento del dedo meñique el contenido de este campo será la abreviatura "men".

Vector resultante	Componentes						
Centro	[	x_centro	,	y_centro	,	z_centro	]
Punto A	[	Pax	,	pay	,	Paz	]
Punto B	[	Pbx	,	pby	,	Pbz	]

Tabla 6 Vectores de posición del volumen de la mano

Volumen ocupado por la mano:

Cuando el mensaje que se envía contiene los datos del volumen ocupado por la mano del cirujano el campo cabecera 1 contiene en la palabra "vol". Para este caso, incluimos una segunda cabecera (cabecera 2) para diferenciar el tipo de volumen enviado:

- la palabra "paral" cuando el volumen a enviar es un paralelepípedo.
- Y la palabra "esfera" cuando el volumen es una esfera.

En ambos casos, los campos siguientes se compondrán por dos campos de tipo vector y uno tipo dato para simplificar asignación de los valores una vez lleguen al robot. En la Tabla 7 se muestran el mensaje que contiene los volúmenes posibles adoptados por la mano.

Los dos campos tipo posición serán utilizados cuando el volumen sea un paralelepípedo para transmitir los puntos que forman su diagonal mayor. En el caso de la esfera, sólo se necesitará un campo de tipo posición para enviar el centro de la esfera. Pero para mantener el mismo tipo de formato el segundo punto se rellenará con los mismos datos que el primero.

Cabecera 1	Cabacera2	Punto1	Punto2	Radio
Vol	Esfera	Centro	Centro	Radio
Vol	Paral	Punto A	Punto B	Radio

Tabla 7 Mensaje con la información del volumen adoptado por la mano

El último dato, es un dato individual y contendrá el radio de la esfera. Este dato pese a ser enviado en ambos casos, sólo cobra sentido en el caso de que el volumen de la mano sea una esfera.

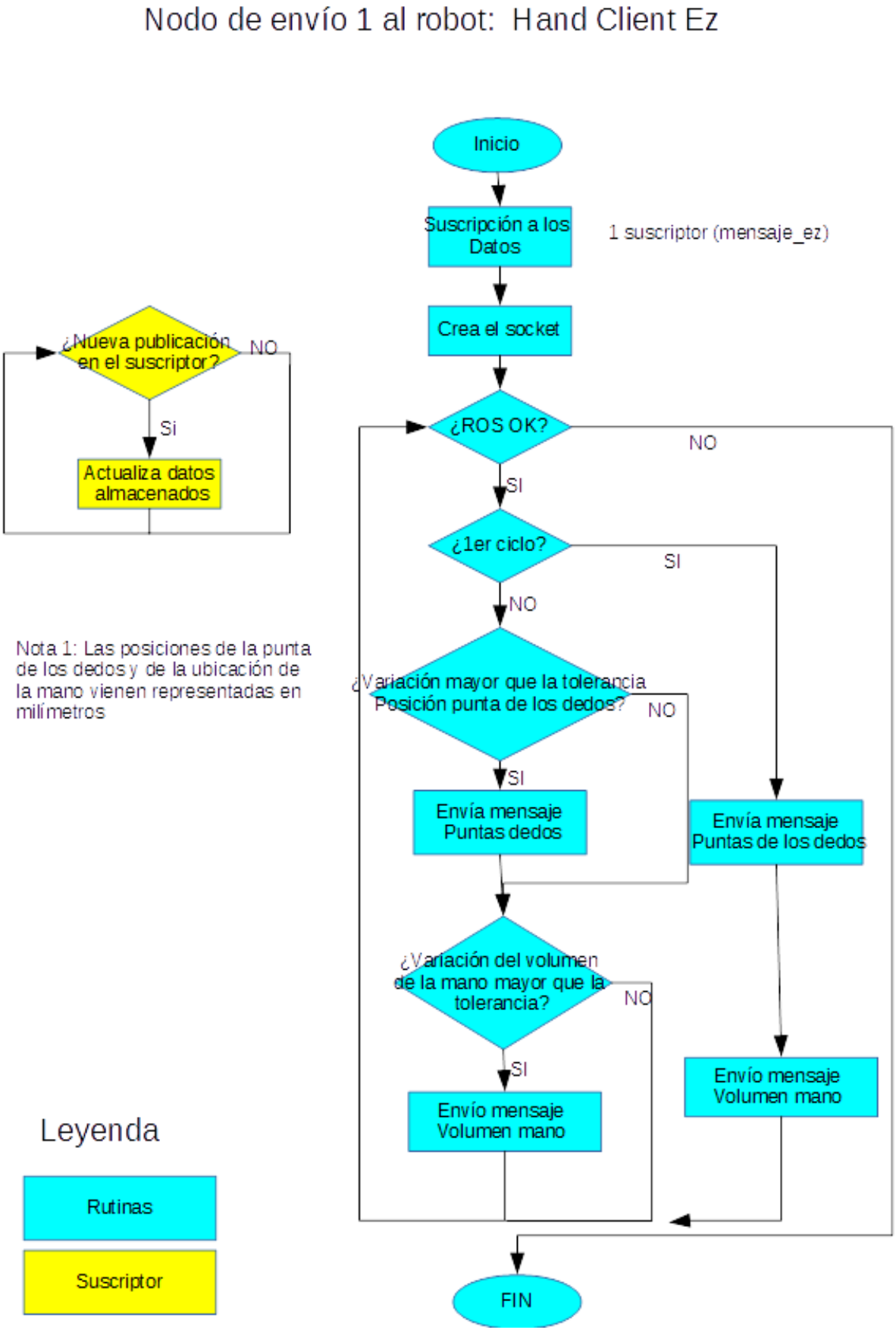

Figura 23 Esquema del nodo Hand_client_ez

3.4. Nodo Hand Client HP

El nodo hand_client_Hp representado en la Figura 19 y la Figura 20 como envío 2 al robot es una de las dos tareas encargadas de transmitir la información desde el ordenador hasta el robot. Esta tarea es la encargada de transmitir los datos secundarios definidos en el apartado 2.4.

El diagrama de este nodo se encuentra representado en la Figura 23.

El nodo al iniciarse crea un socket TCP que se conectará a la tarea homóloga en el robot. La IP del robot es la 157.88.201.130 y el puerto utilizado para la conexión será el 8080.

3.4.1. Tratamiento de datos previo al envío

Tras establecer el socket de comunicación con el robot, el nodo habilita el suscriptor por el que recibirá los datos secundarios (representado en el código como mensaje hp). Una vez los datos son obtenidos por el nodo se clasifican en dos nuevas categorías en función a su tipo:

- Dimensiones constantes del paralelepípedo: Como su nombre indica, los datos que contiene esta categoría se corresponde con la columna "Dimensiones constantes de la mano" de la Tabla 2.

- Ángulos de flexión de los dedos: Contiene los datos de los ángulos de flexión de los dedos, el número total de los datos que contiene esta categoría se corresponden con la columnas "Ángulos de flexión dedos" y "Ángulos de separación entre dedos" de la Tabla 2.

- El motivo de clasificar nuevamente los datos obtenidos en dos categorías se debe a que las dimensiones constantes del paralelepípedo sólo se envían una vez desde a través del socket hacia el robot. De esta forma, se crean dos tipos de mensaje independientes y podemos optimizar el flujo de datos a través del socket.

Al igual que en ocurre en el nodo anterior (apartado 3.3), el mensaje a enviar estará codificado en una cadena de caracteres, que utilizará un asterisco como separador de campos dentro del propio mensaje. También se aplicará el margen de tolerancia para reducir el número de datos enviados exclusivamente a los datos agrupados en la categoría "ángulos flexión dedos". En este caso el margen de tolerancia impuesto será 0.02 radianes. Los datos correspondientes a las dimensiones del paralelepípedo no se aplican este margen de tolerancia puesto que sólo son enviados una vez al robot.

3.4.2. Composición de los mensajes

Tras agrupar los datos y determinar que han de ser mandados al robot, son previamente adjuntados a una tarea que se encarga de componer el mensaje a enviar por el socket.

1. **Ángulos de flexión de los dedos**

La composición del mensaje que contendrá el valor de los ángulos de los dedos será compuesto por una cabecera que servirá para identificar el tipo de mensaje enviado. La composición del mensaje puede verse en la siguiente tabla:

Cabecera1	Var 1	Var 2	Var 3	Var 4	Var 5	Var 6	Var 7	Var 8
"p_m"	Pulgar 1	Pulgar 2	Indice 1	Indice 2	Corazón1	Corazón2	Anular1	Anular2

Var 9	Var 10	Var 11	Var 12	Var 13	Var 14
Menique1	Menique2	Pulgarindice	Indicecorazón	Corazónanular	Anularmenique

Tabla 8 Mensaje que contiene la información de los ángulos de la mano

2. **Constantes del paralelepípedo**

En este caso, la cabecera de identificación será compuesta por la palabra "valor_paral" conteniendo a continuación los valores de las dimensiones del paralelepípedo según el orden en que se muestran en la siguiente tabla:

Cabecera	Var 1	Var 2	Var3
"valor_paral"	Largo paralelepípedo	Ancho paralelepípedo	Alto paralelepípedo

Tabla 9 Mensaje que contiene las constantes del paralelepípedo

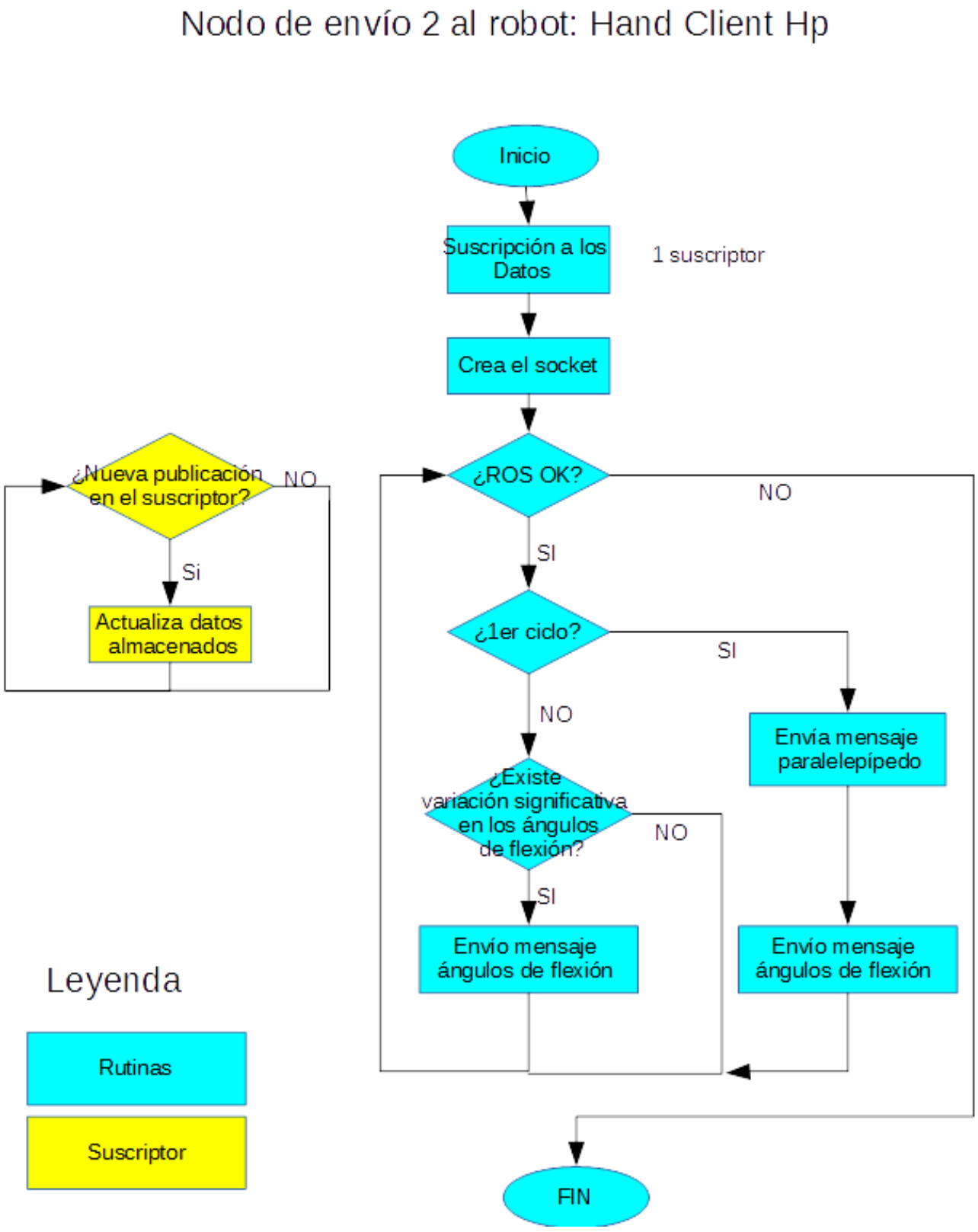

Figura 24 Esquema del nodo hand client hp

4. Gestión de tareas en el robot

En este capítulo se explica cómo se desarrolla la recepción de datos en el robot, la comunicación entre tareas. Y por último, se realiza el control de trayectorias.

En primer lugar, el robot debe ser configurado en multitarea para ser capaz de ejecutar varios programas en paralelo. De esta manera se podrá recibir datos desde los programas desarrollados en el framework ROS. Toda la información disponible de esto se puede encontrar en el manual del operador y en el anexo A1 donde muestra un tutorial para configurar el robot en modo multitarea. Se debe tener en cuenta que el controlador de Rapid no posee una gestión multi hilo de los procesos. Esto implica que para conseguir una gestión de las tareas en paralelo, el controlador ejecuta una instrucción de la tarea 1, almacena sus resultados en registros. Y recupera los datos de la siguiente tarea (variables, puntero de programa) y ejecuta la siguiente instrucción de la tarea. Una vez terminada la ejecución el resultado se almacena en los registros y el procesador cargará los datos de la siguiente tarea. Esto en la práctica significa que ante un sistema multitarea los tiempos de ejecución del programa serán mayores.

Por esta característica y en función de la cantidad de datos a recibir, debemos ser conscientes que el robot marcará la máxima velocidad de recepción de datos.

Otro hándicap que se debe considerar es la ausencia de punteros en Rapid, debiendo adjuntar las variables del programa por valor.

Al igual que en el extremo robot, la organización de las tareas en el robot serán 3: La tarea Ros_Hand_Server_ez, Ros_Hand_Server_hp y TROB1 que se corresponderán respectivamente con las tareas Recepción de datos 1, recepción de datos 2 y control de trayectoria de la Figura 20.

- La tarea Ros_Hand_Server_ez se encargará de recibir y decodificar los mensajes recibidos desde el ordenador, que se corresponden con el volumen ocupado por la mano del cirujano y por la posición de la punta de los dedos.

- La tarea Ros_Hand_Server_hp será la encargada de gestionar los datos correspondientes con los ángulos de flexión de las falanges medias y proximales, (el ángulo formado por estas últimas con los metacarpianos). Y el ángulo de separación entre dedos.

- La tarea TROB1 se encargará de implementar el control de la trayectoria del robot.

En la Figura 25 se muestra un esquema de las tareas que se ejecutan en paralelo en el robot.

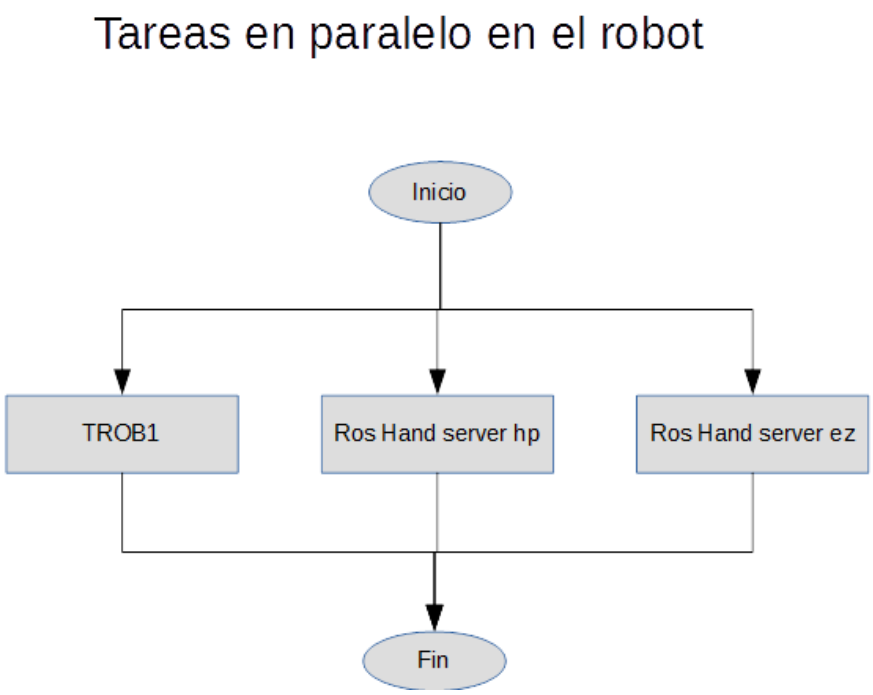

Figura 25 Tareas en paralelo en el controlador del robot

4.1. Intercambio de datos entre tareas:

El método que Rapid habilita para el intercambio de datos entre tareas consiste en declarar en todas las rutinas a compartir datos variables globales y persistentes con el mismo nombre y tipo. Por ejemplo, en este TFG uno de los datos principales que debe comunicarse desde las tareas de recepción a la tarea motora del robot es la posición de la punta del dedo meñique.

Como se mencionó anteriormente este TFG se ha desarrollado con vistas a que en un futuro el punto que deba seguir el robot pueda ser distinto que la posición de la punta del dedo meñique. Es por esto, que en el código del control de trayectoria el valor de la posición de la punta del dedo meñique es asignado a la variable de tipo POS y con el nombre “punto_a_seguir”.

4.1.1. Shared_variables

La forma en la que Rapid comparte datos entre tareas es declarándolas en cada tarea con el mismo tipo y nombre, este procedimiento puede generar errores de declaración al tener que definir todos los datos a compartir en las tres tareas, para

minimizar la repetición de declaraciones y simplificar el código, se ha optado por declararlas en el módulo "Shared_variables". Para el correcto funcionamiento del programa, el módulo se deberá cargar en todas las tareas que compartan variables

Este módulo, aparte de las variables a compartir entre tareas, incluye el resto de variables que se reciben a través de las tareas de recepción, la declaración de los sistemas de referencia, la definición de las herramientas y la definición de funciones y procedimientos con los que modificar las variables.

El flujo de datos entre tareas aparece representado en Figura 26.

Estos últimos procedimientos, pese a no ser necesarios para el acceso a las variables que deseamos compartir se utiliza con el fin de controlar el acceso y la modificación de los parámetros como si todo el módulo se tratara de una Clase con variables privadas, y las funciones que controlan su lectura y escrituras fueran el único método de acceso a dichas variables. Como consecuencia de imponer esta restricción durante el desarrollo, se redujeron los tiempos de corrección de errores ligados a la comunicación entre tareas de Rapid.

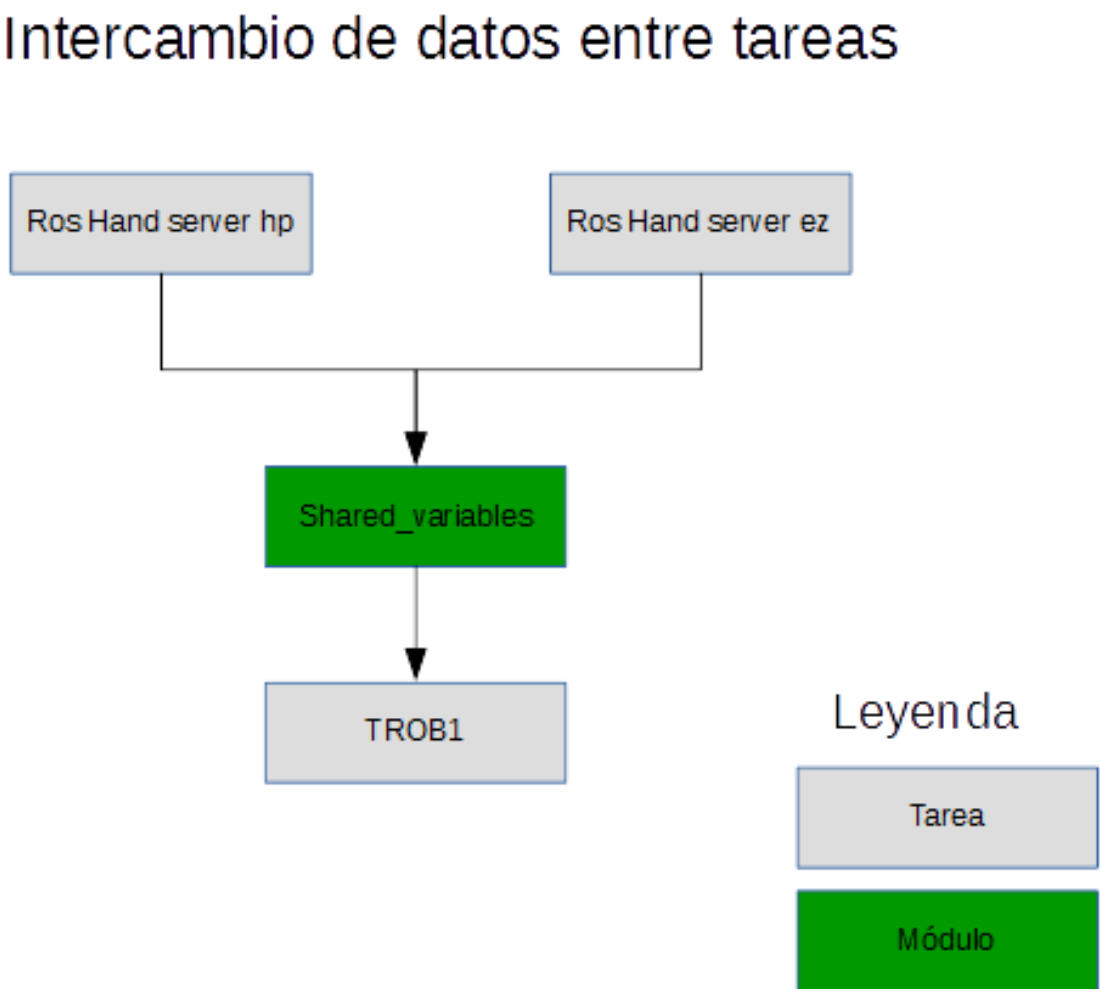

Figura 26 Intercambio de datos entre tareas en Rapid

4.2. Recepción de datos:

Las tareas de recepción de datos en el robot son dos: Ros_Hand_Server_ez, Ros_Hand_Server_hp. Estas tareas se encargan de recibir y almacenar todo el flujo de datos enviados respecto del robot. En los aparatados 3.3 y 3.4 detallamos los puertos que utilizarían cada nodo de ROS para comunicarse con la correspondiente tarea en el robot. No obstante, en la Tabla 10 mostramos esta información:

Tarea	Puerto
Ros_Hand_Server_ez	8080
Ros_Hand_Server_hp	80

Tabla 10 Puertos utilizados por cada tarea

El funcionamiento de ambas tareas es el mismo. Las tareas se inician llamando a la rutina ROS_init_socket que se encarga de crear el canal de comunicación Socket de tipo TCP, devolviendo un error en caso contrario. Seguidamente las tareas llaman a la función Ros_wait_for_client, la cual se encarga de esperar de forma indefinida la solicitud de parte de un equipo remoto para iniciar la comunicación. En ese momento acepta la conexión y habilita la transmisión de datos. Por último, ambas tareas entran en un bucle de recepción de datos indefinida mientras el Socket respectivo siga operativo.

Una vez explicadas la parte común a ambas tareas, pasamos a describir las diferencias entre ambas

4.2.1. Ros_Hand_Server_ez

EL esquema de la tarea se encuentra representado en la Figura 27, se explicará todo el funcionamiento descrito a continuación.

Las instrucciones de respecto a cómo realiza la recepción y el tratamiento de los datos se realiza en la función ROS_receive_DZmsg. En esta función se declara una variable de tipo String llamada buffer y que almacenará la lectura de los datos del socket.

Tras esto, y como se explicó en el capítulo anterior, la tarea el identifica las posiciones del mensaje donde se localizan los separadores, estos separadores fueron definidos en el apartado 3.2

Una vez localizados, se extrae el primer campo del mensaje que identifica el tipo de datos que contiene el mensaje. Si la cabecera, en este caso es la palabra “vol” se corresponderá con el volumen que ocupa la mano del cirujano. Si la cabecera contiene la palabra “dedo” contendrá las posiciones de las puntas de los dedos

El resto de campos serán igualmente extraídos, y aquellos cuyo contenido sea numérico serán convertidos del formato String al formato numérico en cuestión con

la instrucción “StrtoVal”. Es decir, el uso de esta instrucción posibilita la conversión a datos tipo de numérico individual (Num), como de agrupaciones de ellos (POS, Robtarget, Jointtarget,…..) siempre y cuando el dato convertido y el tipo de variable donde se asignará sean compatibles.

Para recordar la composición de los mensajes recibidos en el robot ver el apartado de composición de mensaje del apartado 2.3

En función del tipo de mensaje contenido dicho mensaje será des encapsulado en dos tipos de estructuras diferentes:

1. Posición de las puntas de los dedos.

   En este caso, la estructura que contendrá la información de la posición de los dedos definida en el apartado 3.3.2.

   Una vez terminado el des encapsulamiento toda la estructura es pasada al procedimiento set_posición_puntas_dedos contenido en el módulo shared_variables, en el cual se encarga de asignar las variables compartidas entre tareas correspondientes a la posición de las puntas de los dedos y del punto a seguir. Se debe tener en cuenta que todas las posiciones están referidos respecto del sistema de referencia de la cámara estereoscópica. Para poder utilizar el punto a seguir en el control de trayectorias más adelante es necesario hacer un cambio de base de este punto respecto al sistema de referencia pelvitrainer.

2. Volumen ocupado por la mano

   El contenido de este mensaje es el volumen que ocupa la mano del cirujano que definimos en el apartado 3.3.2. Al igual que pasara en el caso anterior, las coordenadas vienen referidas al objetivo del sistema de visión. El volumen definido se utilizará para definir una zona mundo. A la cual, el TCP del robot no pueda acceder para asegurar que el robot no daña la mano del cirujano.

   En este caso, los datos también son almacenados en una estructura que consta de los siguientes campos:

   - Tipo geometría: De tipo string, sirve para identificar el tipo de geometría enviada
   - posicion1: De tipo posición, contiene la posición de un punto significativo de la mano.
   - posicion2: De tipo posición, contiene la posición de un punto significativo de la mano.
   - radio: Contiene el valor del radio de la esfera al que se aproxima la mano. Cuando este no es el caso este valor no se utilizará.

Terminada esta fase de des encapsulamiento, para poder definir las zonas mundo correctamente se debe hacer un cambio de base de las coordenadas recibidas a

través del socket. Las coordenadas deberán ser definidas previamente respecto del sistema de coordenadas base del robot. Y más tarde, todas las variables serán asignadas a sus homólogas dentro del módulo Shared_variables para poderlas utilizar en la tarea motora del robot.

Para obtener una posición respecto de otro sistema de coordenadas se utiliza la instrucción de Rapid posevect.

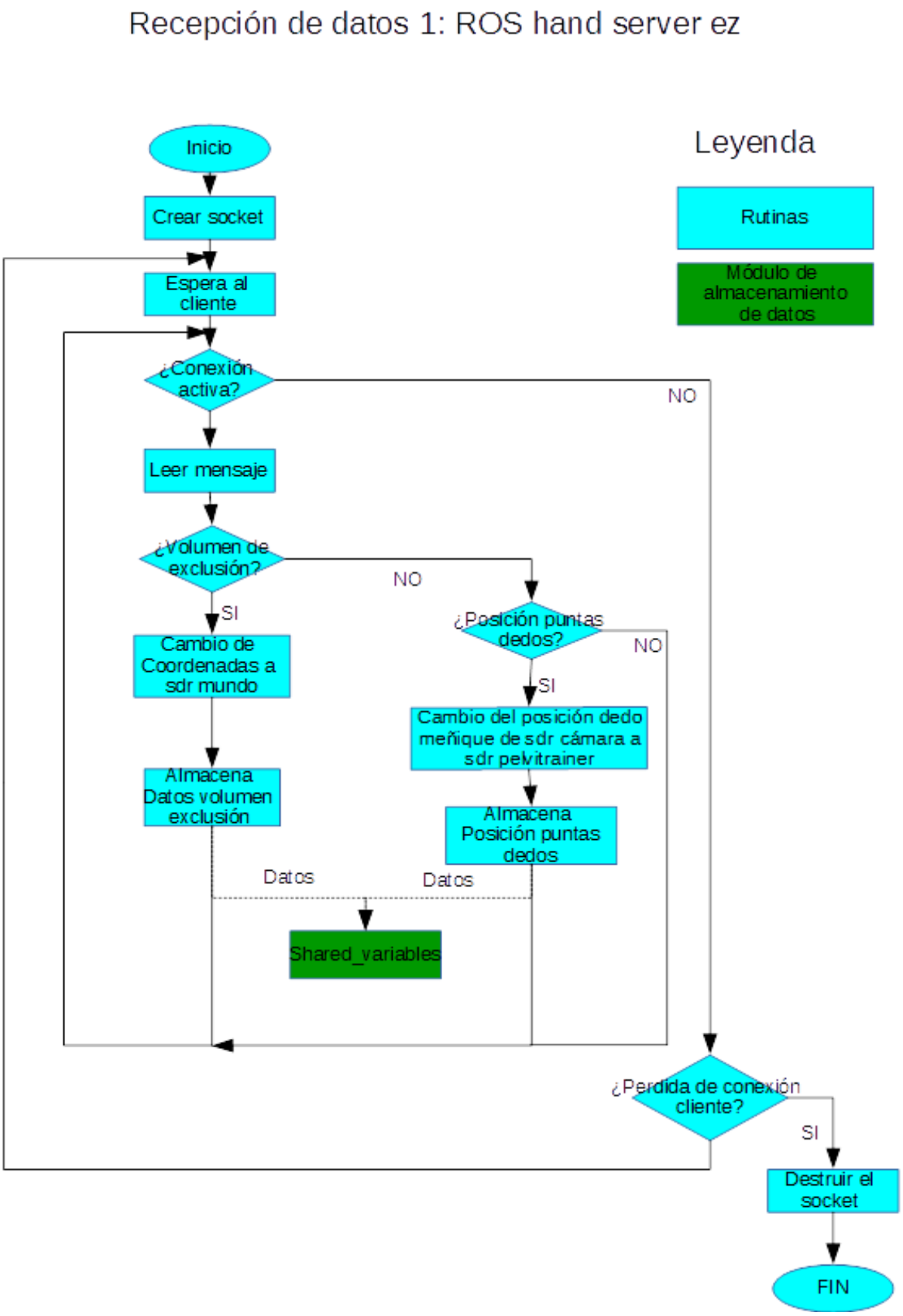

Figura 27 Tarea Ros hand server EZ

4.2.2. Ros_Hand_Server_hp

El esquema de la tarea Ros_hand_server_hp se muestra en la Figura 28.

Las instrucciones de respecto a cómo realiza la recepción y el tratamiento de los datos se realiza en la función ROS_receive_HP msg. En esta función, declara una variable de tipo String que se llamará buffer y donde se almacenará la lectura de los datos recibidos por el socket.

En esta tarea se utiliza el mismo modo de proceder que en la tarea ROS_Hand_Server_ez. Donde se buscarán los caracteres especiales que delimitan la longitud de los campos. Una vez encontrados se extraerá la cabecera que identificará el tipo de mensaje recibido y su tratamiento posterior.

Los tipos de mensajes que esta tarea recibe y trata son las constantes del paralelepípedo y los ángulos de flexión entre falanges medias y proximales, definidas en la Tabla 2.

1. Constantes del paralelepípedo

   El primero se corresponde con las dimensiones constantes del paralelepípedo (alto, largo y ancho) formado por la aproximación de la mano del cirujano cuando esta se encuentra totalmente extendida.

   Como se mencionó anteriormente, las dimensiones del paralelepípedo son constantes a cada cirujano y será recibido en el robot sólo una vez durante todo el proceso.

2. Ángulos de flexión y separación

   El contenido de este mensaje, está compuesto por el envío de los ángulos de flexión entre falanges medias y proximales. Y el formado de estas últimas con sus correspondientes metacarpianos. Además contiene los ángulos de separación entre dedos. Cuando se reciba este tipo de mensaje, la cabecera estará compuesta por la palabra *"p_m"*

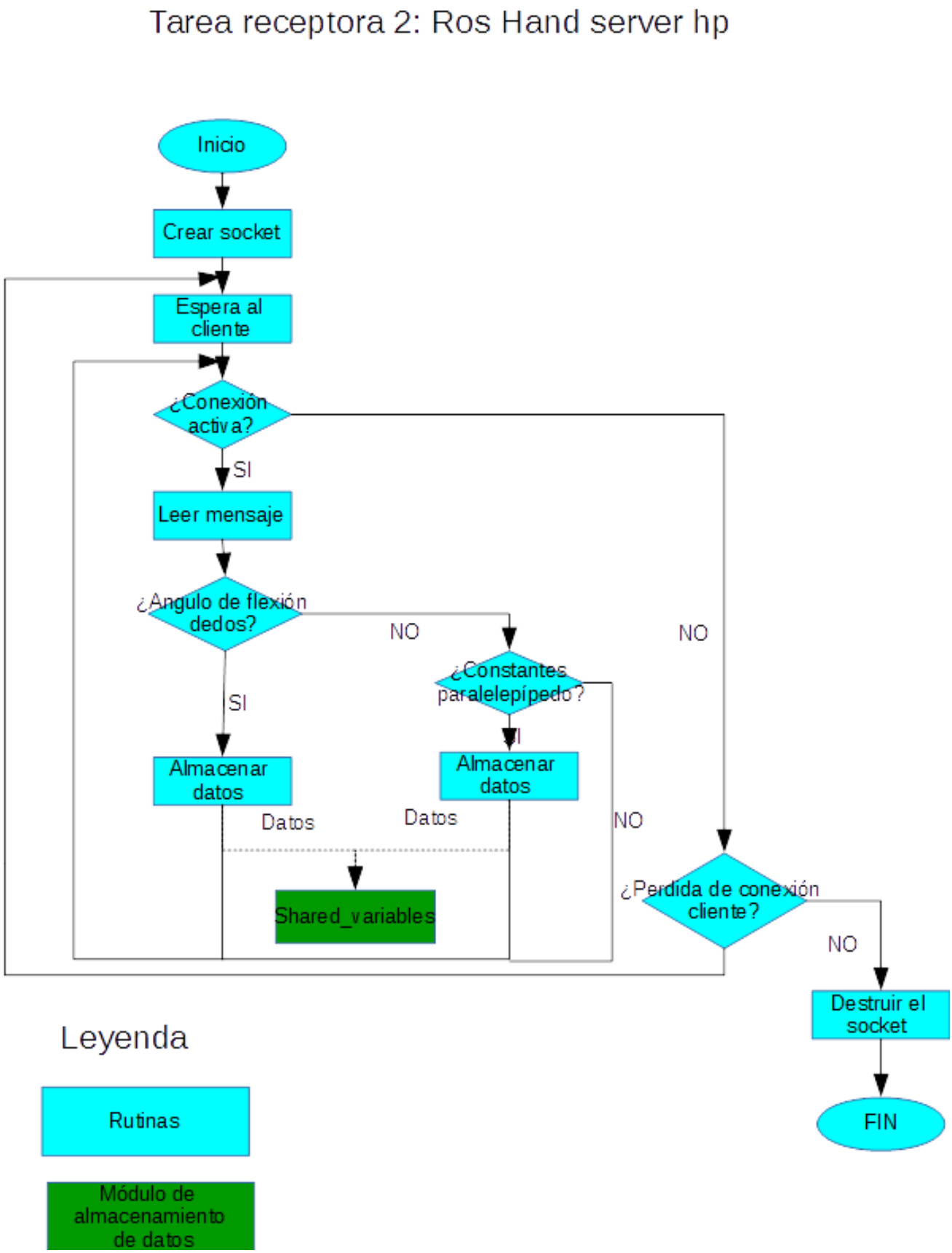

Figura 28 Tarea Ros hand server HP

4.3. Control de trayectoria

Una vez se ha establecido la comunicación con el robot, el siguiente paso es la implementación del control de trayectorias del robot para seguir el dedo meñique sin colisionar con él. Sin embargo, con vistas de futuro, cuando explicamos la composición de mensajes en el 3.3.2, se insertó un campo que incluiría las iniciales del dedo a seguir.

Los datos indispensables que el control de trayectorias debe conocer son:

- La posición del dedo meñique.
- La información que determina el volumen de la mano simplificado en los dos tipos especificados (esfera y paralelepípedo)

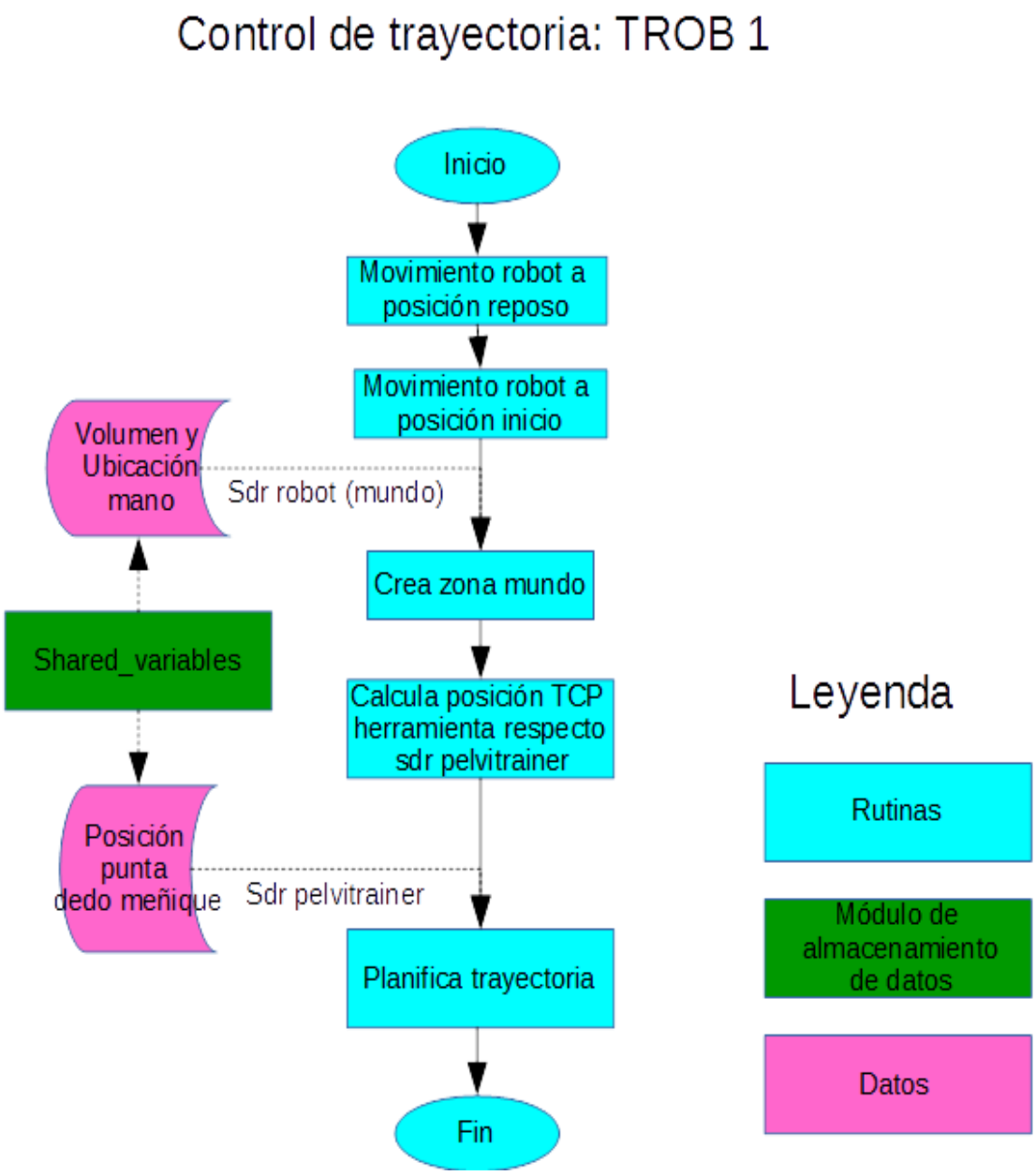

Figura 29 Esquema tarea TROB1

Ante todo, se debe tener en cuenta que la herramienta del robot jamás penetrará en el volumen ocupado por la mano. En este caso especial el robot se parará y detendrá el programa. La forma para implementar esta medida de seguridad en Rapid es utilizar zonas mundo.

4.3.1. Zonas mundo

Una de las soluciones encontradas para implementar en el sistema de movimiento sin colisión con obstáculos consiste en la utilización de zonas mundo. Las zonas mundo son volúmenes dentro del área de trabajo que pueden ser utilizados para evitar que el TCP de una herramienta o del propio robot acceda a dicha zona. O en caso contrario, no sea capaz de salir de un volumen predeterminado.

La utilización de zonas mundo implica la supervisión del TCP de la herramienta en cuestión. El controlador del robot habilitará rutinas de supervisión de del TCP que muestreará la posición con tal de que este no penetre en la zona mundo. Se debe tener en cuenta que mientras se muestrea y se comprueba si la posición del robot está dentro o fuera de los márgenes de las zonas mundo, transcurre un tiempo en el cual el robot sigue su movimiento. En función de la velocidad de movimiento la frecuencia de comprobación debe ser superior.

En el caso en que el robot permanece dentro de la zona un intervalo de tiempo demasiado breve, existe la posibilidad de que este atraviese una esquina de la zona sin ser detectado. Por tanto, el usuario debe definir la zona mundo de un tamaño mayor que el área de riesgo.

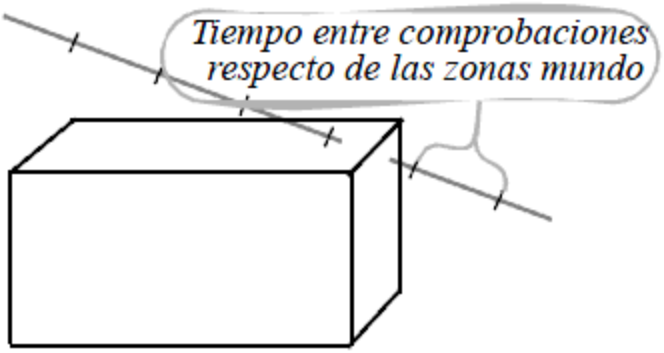

Figura 30 Posibilidad de no detección

Rapid por defecto sólo determina tres volúmenes posibles de zonas mundo: un cilindro cuyo eje coincide con el eje z del robot, una esfera y un paralelepípedo.

Para definir cualquier tipo de zona mundo es necesario que todas las coordenadas estén referidas al sistema mundo. Este sistema se corresponde con el sistema de coordenadas propio del robot con origen en su base.

Las acciones que habilita el controlador de Rapid cuando el TCP penetra en una zona mundo consisten en la activación de salidas digitales.

Esta acción activa una señal digital cuando el TCP se encuentra dentro o en las inmediaciones de una zona mundo. Tal y como se muestran en las Figura 31 y Figura 32.

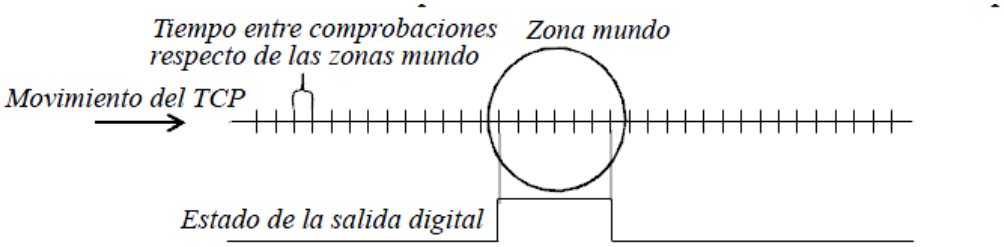

Figura 31 Activación de la señal digital dentro de la zona mundo

Esta aplicación de las zonas mundo resulta útil a la hora de indicar que el robot se detenga dentro un área específica, como es nuestro caso. Aun así deberemos prestar atención al tiempo transcurrido desde que se activa una señal digital hasta que el robot se detiene. Como se muestra en la Figura 32, pese a que la señal digital se activa antes de que el TCP del robot penetre en la zona mundo, el tiempo de parada es suficientemente grande como para que el TCP del robot una vez detenido pueda penetrar en la zona seleccionada. Los motivos de este tiempo de parada se deben a la conjunción de varios factores.

El primer factor, aunque no aparezca en la imagen, se asocia al retardo de tiempo desde que se detecta que el robot está en las inmediaciones de la zona mundo hasta, que se activa la salida digital. Tras este, el sistema de control del robot comprueba periódicamente entradas y salidas digitales transcurriendo cierto tiempo desde que la salida se activa hasta que el sistema de control la detecta. Por último, una vez detectada, se manda una señal de paro a los motores del robot y hasta que estos se detienen transcurre otro intervalo de tiempo.

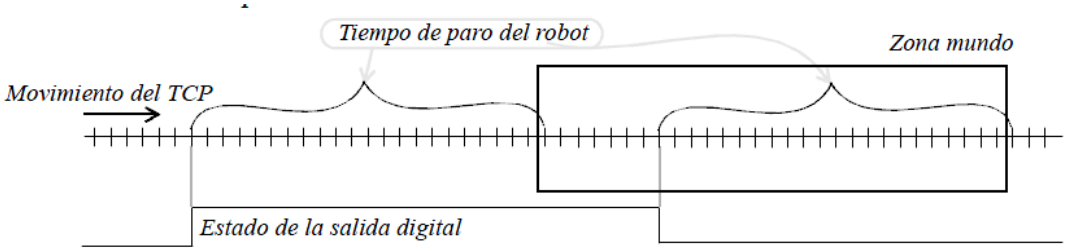

Figura 32 TCP penetrando en la zona mundo

Esta acción activa una salida digital antes de que el TCP alcance una zona mundo.

Pese a todos los retardos de tiempo, Rapid ofrece la posibilidad de que el TCP del robot no penetre en la zona mundo estimando internamente un tiempo de parada que satisface las necesidades del usuario.

La aplicación de las zonas mundo a nuestro trabajo consistirá en definir dos zonas mundo. La primera constituida por el volumen simplificado que ocupa la mano del cirujano. Como medida de seguridad se establecerá que si el TCP del robot se

encuentre cerca de la frontera de la zona mundo se activare una salida digital que. A su vez, genere una parada del robot utilizando una rutina de interrupción. Así pues, este sistema será constituido como un sistema de seguridad crítico.

La segunda zona mundo definida en este proyecto será del mismo tipo de volumen de la zona que ocupa la mano del cirujano y que englobará a esta. Será utilizada en el control de trayectoria para modificar la trayectoria seguida hasta el momento.

4.3.2. Método de los potenciales

El método de los potenciales se basa en aplicar al espacio de trabajo un campo de potencial, que en la aplicación computarizada consistiría en aplicar a cada punto del espacio un valor escalar determinado por un campo vectorial.

Este sistema nace de la modelación de procesos físicos para simulación en computador. Por ejemplo, con este tipo de algoritmos se pueden simular interacciones de elementos situados dentro de campos gravitatorios, magnéticos o eléctricos.

La aplicación de este método en el control de trayectorias consiste en situar al punto de fin en el interior de un campo muy atractivo, a los posibles obstáculos asignarles como generadores de un campo muy repulsivo y el punto inicio tendrá un ligero potencial repulsivo.

1. **El potencial atractivo:**

   El potencial atractivo debe estar compuesto por una fuerza de atracción que disminuye conforme el elemento a controlar (en este caso el robot) se aproxima a él.

   La fuerza atractiva resultante del campo de potencial se define como:

$$\overrightarrow{F^{atr}} = -\xi * D * \vec{D}$$

Ecuación 1 Fuerza atractiva del campo de potencial puntual

   Siendo:

- F^{atr}: El vector de fuerzas
- ξ: El coeficiente proporcional.
- D: La distancia euclidea entre el robot y el objetivo.
- $\vec{D}$: Distancia vectorial entre robot y objetivo

   En la Figura 33 se puede mostrar la representación del campo de potencial atractivo.

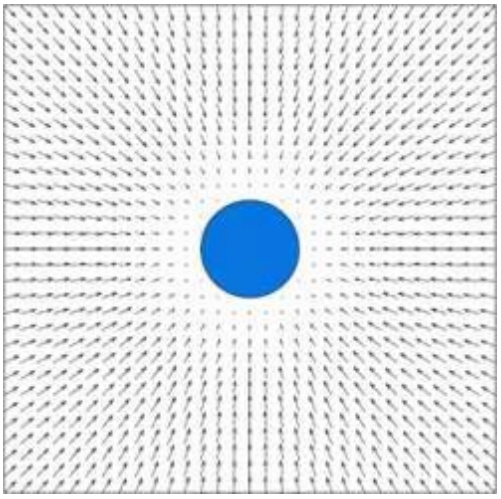

Figura 33 Representación del campo de potencial atractivo

2. El potencial repulsivo:

Al contrario de lo que ocurriera con el potencial atractivo, el potencial repulsivo debe aumentar su valor conforme el objeto a controlar se adentra en el campo. Este campo empezará ejercer repulsión cuando el objeto a controlar (robot) se encuentre dentro del volumen de influencia

$$\overrightarrow{F^{rep}} = \begin{cases} \eta\left(\frac{1}{D} - \frac{1}{do}\right)\frac{1}{D^2}\vec{D}; SiD \leq do \\ o \\ 0; SiD > do \end{cases}$$

Ecuación 2 Fuerza repulsiva ejercida por un campo de potencial puntual

Siendo:

- F^{rep}: El vector de fuerzas
- η: Constante proporcional
- D: Distancia euclídea entre el robot y el origen del campo
- do: Distancia de influencia del campo.

En la Figura 34 se puede mostrar la representación del campo de potencial repulsivo.

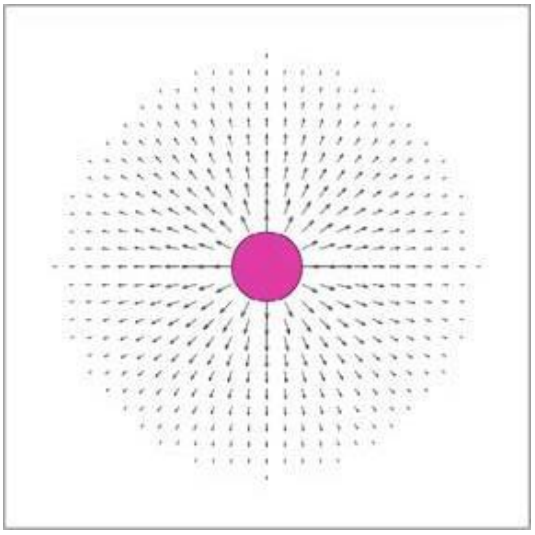

Figura 34 Representación del campo de potencial repulsivo

3. La superposición de potenciales:

La superposición de todos los campos de potenciales generados, tanto por obstáculos como por puntos de inicio y fin son superpuestos, determinando el campo resultante. Puede darse el caso que en determinadas áreas la confluencia de un campo positivo y uno negativo se contrarresten y den como resultado un campo neutro. Este caso debe ser tenido muy en cuenta cuando el punto fin y el punto final se encuentra muy cerca de un obstáculo. Pues la cancelación de fuerzas puede originar una trayectoria que penetre dentro del obstáculo.

Una vez explicado el funcionamiento pasamos a evaluar sus ventajas e inconvenientes.

Ventajas:

- Permite la generación de trayectorias en tiempo real, las cuales serán son construida iteración a iteración en tiempo real. Siendo especialmente utilizado en aplicaciones de gran velocidad de planificación.

- Las trayectorias generadas son suaves:
  Esta característica es debida al determinar un campo de costes sin discontinuidades ayudando así al movimiento progresivo del robot, evitando cambios de trayectorias bruscas.

Inconvenientes:

- Mínimos locales: El principal problema de esto método reside en la aparición de mínimos locales. Estos mínimos son lugares con menor potencial que todos los de su alrededor. El del control de trayectoria, al

detectar que todos los puntos de alrededor poseen mayor potencial que el punto actual, se quedará “atrapado” en esa ubicación sin salir de ella.
La solución a este problema radica en ejecutar rutinas de “backtraking” o ejecución marcha atrás. En las que deberemos almacenar una cantidad suficiente de puntos previos al actual para, en caso de que el robot quede “atrapado” y sea detectado por el control. Poder retroceder a los puntos anteriores hasta una distancia prudencial, Entonces se producirá un desplazamiento en cualquier dirección del robot. Y dejando al controlador que vuelva a calcular la trayectoria. Si la elección del desplazamiento es correcta, el robot no volverá a caer en el mismo mínimo local.

4.3.3. Implementación del método

La implementación del sistema de control de trayectorias se insertará en la tarea principal del robot (Tarea TROB1) por ser esta la que tiene la capacidad de indicar las instrucciones de movimiento del robot. Esta tarea aparece representada en la Figura 29

Para un correcto funcionamiento del sistema, el tipo de tarea asignado a las tareas receptoras será “Semistatic”. Esto en la práctica supone que las tareas empiezan a ejecutarse automáticamente con el arranque del controlador.

Habilitando la recepción de datos sin ser necesario el inicio de la rutina principal. Estas tareas permanecerán ejecutándose pese a un fallo del control de movimiento. El motivo de su recomendación se basa en el hecho de mantener una comunicación robusta y estable previa al inicio, la ejecución y el fin del programa. Sin embargo, al configurar este tipo de rutinas, se pueden generar fallos internos del controlador del robot. Para solucionarlos será necesario reiniciar tanto el controlador del robot como el programa en el ordenador desde donde se estaba configurando. Aun así, estos errores pueden seguir generándose.

El comienzo de la rutina comienza con la rutina de establecer las zonas mundo determinadas por el volumen de la mano del cirujano. Como se explicó en apartados anteriores, se establecerán dos zonas mundo. La principal, donde estará contenida la mano del cirujano. Esta zona mundo activará una señal de control digital en caso de una penetración inminente del TCP de la herramienta dentro del robot, que causará la parada inmediata del sistema. A este volumen se denomina en el código como volumen de exclusión. Puesto que será un volumen que el robot jamás podrá alcanzar.

El otro volumen será una extensión de la anterior, englobándola y constituyendo el volumen de influencia del campo repulsivo generado por el obstáculo, que en este caso supone la mano del cirujano. Este volumen será 1.3 veces el volumen definido por la mano del cirujano y el nombre que tomará esta zona en el código será llamado Capa1. Para la identificar si el TCP del robot se encuentra dentro del

volumen de exclusión se ha asociado una señal digital que se activará cuando el robot esté dentro de dicho volumen.

Para ilustrar mejor este concepto, en la Figura 35 se muestra como el volumen ocupado por la mano aparece modelado como una esfera de color rojo en el centro. De la misma forma, en la Figura 36 se representa de color marrón el paralelepípedo que contiene a la mano del cirujano, envuelto a su vez por el paralelepípedo de color verde que se corresponde con su volumen de exclusión.

Figura 35 Modelado 3D de las zonas mundo para el caso esfera

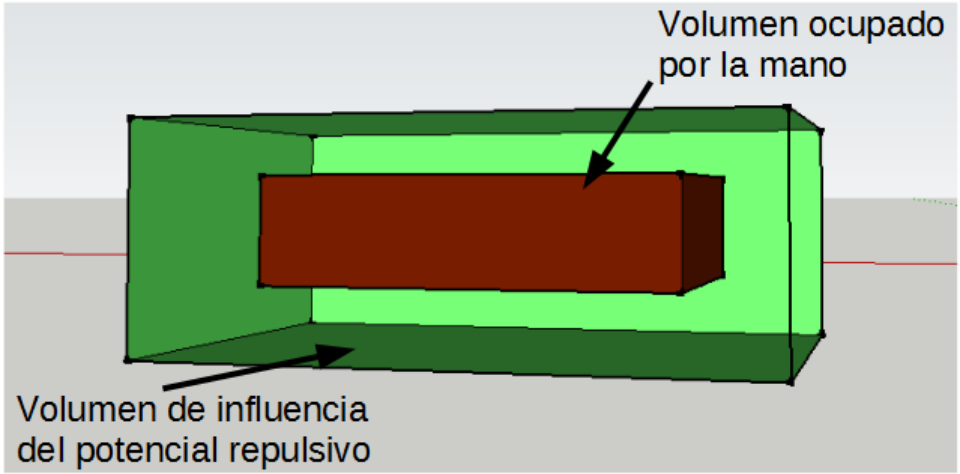

Figura 36 Modelado 3D de la zona mundo para el caso paralelepípedo

En el caso que el volumen de la mano adopte la forma de paralelepípedo el volumen de influencia del potencial repulsivo será también 1.3 veces el volumen del ocupado por la mano.

En la definición teórica del método de los potenciales las ecuaciones 1 y 2 son definidas para campos de potencial puntuales. Sin embargo, para el caso de la

mano en cualquiera de los volúmenes que adopta el campo de potencial pasa a ser generado por la superficie exterior del volumen de la mano. Por lo tanto, es necesario redefinir los términos de la ecuación 2 para adaptarla a este caso:

$$\overrightarrow{F^{rep}} = \begin{cases} \eta\left(\frac{1}{d} - \frac{1}{d1}\right)\frac{1}{D^2}\vec{D}; SiD \leq d1 \\ o \\ 0; SiD > d1 \end{cases}$$

Ecuación 3 Fuerza repulsiva del campo de potencial volumétrico

Siendo:

- F^{rep}: El vector de fuerzas
- η: Constante proporcional
- d: Mínima distancia entre el robot y la superficie del volumen ocupado por la mano
- $d1$: Mínima distancia de influencia del campo, definida por la distancia entre la superficie del volumen ocupado por la mano y la superficie del área de influencia

De acuerdo con la ecuación 3, en la

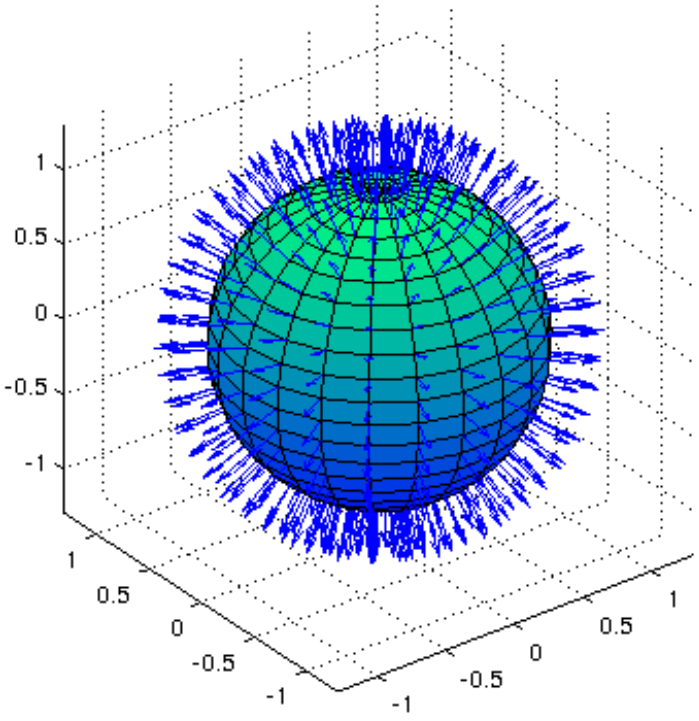

Figura 37 se puede apreciar una representación aproximada del el campo de potencial repulsivo que generará la mano cuando esta adopte la forma de una esfera y el robot se encuentre inmerso en su volumen de influencia

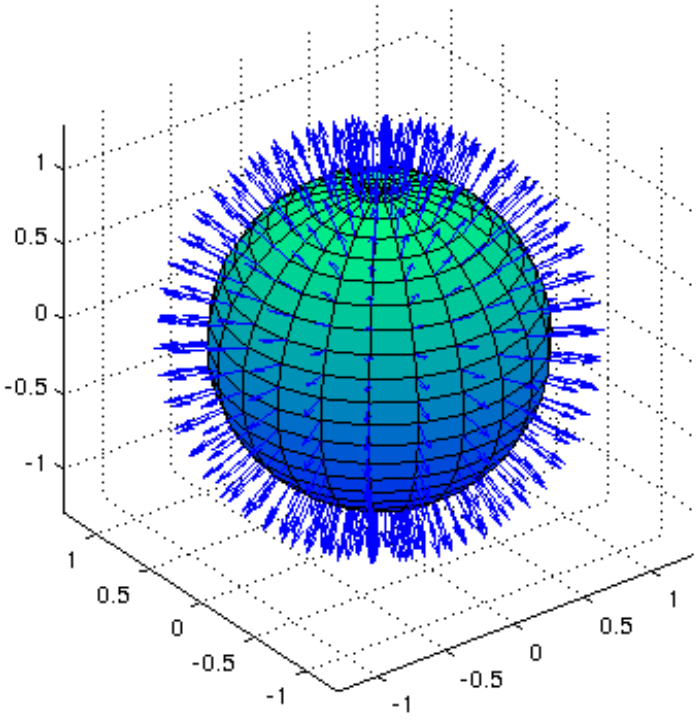

Figura 37 Campo de potencial generado por la mano en posición esfera

Teniendo en cuenta todo lo anterior y debido a que la posición del dedo meñique se encontrará en el perímetro del volumen ocupado por la mano. Al superponer los campos de potencial repulsivo del volumen ocupado por la mano contra el atractivo generado por la posición del dedo meñique se genera un volumen donde el potencial repulsivo es reducido o incluso nulo. Para que dicho volumen sea lo más reducido posible se caracterizará con un valor muy grande a la constante proporcional del campo repulsivo definido en la ecuación 3. El valor de todas las contantes de los campos de potencial utilizados en este TFG se aparecen en la Tabla 11.

En la Figura 38 se representa un esquema en 2D con la superposición de potenciales para el caso en el que la mano adopta la posición esfera. En él también se representa el volumen de potencial donde este es susceptible de anularse.

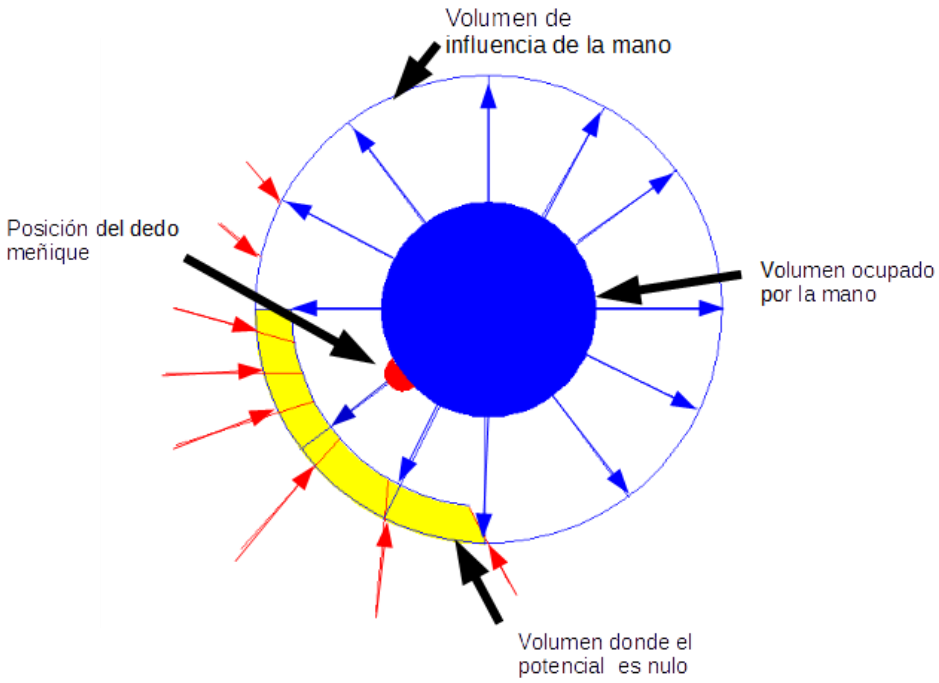

Figura 38 Superposición de potenciales para el caso esfera

Aclaración: En la Figura 38 se debe tener en cuenta que el campo atractivo reduce su potencial conforme la distancia al objetivo se reduce. Inversamente, el campo de potencial repulsivo aumenta su potencial conforme nos acercamos a la superficie del volumen ocupado por la mano.

4.3.4. Módulo planifica trayectoria

Una vez establecidas las zonas mundo iniciales estableceremos entramos como tal en el módulo del control de trayectorias. Este módulo se ejecutará cíclicamente determinando el siguiente punto donde se moverá el robot.

En primer lugar, se establecen y habilitan las interrupciones que determinan cuando existe un cambio del volumen de la mano y una nueva posición del dedo meñique que se almacenará en la variable "punto a seguir" del esquema de la figura 34. Y las interrupciones determinadas por la inserción inminente o reciente del TCP del robot sobre las zonas mundo.

Otra consideración que no se pasar inadvertida es la distancia entre el punto actual y el punto siguiente calculado. Esta variable dentro del sistema se denominará "paso". La importancia del paso de cálculo radica en la precisión con la que el desarrollador desea alcanzar el punto deseado. Si la precisión fuese poca, el esfuerzo de cálculo del robot sería muy inferior. Por el contrario, la precisión del robot se vería notablemente afectada, dándose el caso que el robot habiendo llegado al punto se quedase pivotando en torno a sus alrededores sin llegar a encontrar el punto exacto. Además, existe que los movimientos producidos por el control de trayectoria sean más irregulares dando la impresión que el robot se queda "trabado".

En este TFG se ha tratado de conciliar la ventaja que ofrece el paso de cálculo grande con la precisión que ofrece el paso de cálculo pequeño. Para ello, se ha considerado un paso de cálculo de 1 mm. Para garantizar que el robot alcanza el punto de fin con exactitud, y no se queda pivotando entorno a él se ha adaptado el algoritmo de la bisección al paso de cálculo. Se ha establecido que si la distancia euclídea entre el robot y el punto de destino es menor de paso y medio. Este será reducido a la mitad. El paso será restaurado al original cuando el punto a seguir sea modificado.

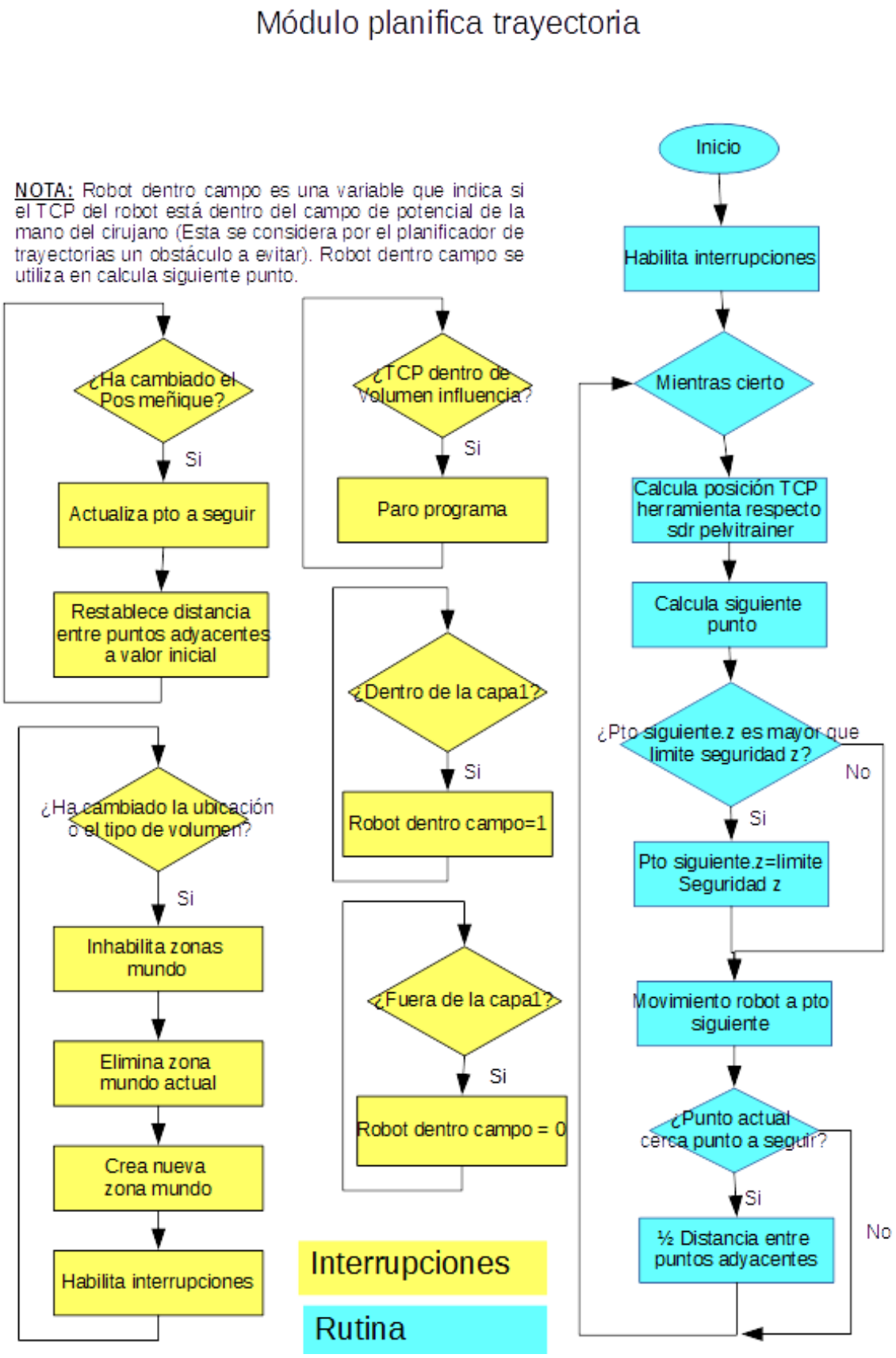

Figura 39 Módulo planifica trayectoria

Tras almacenar la posición del dedo meñique en la posición del punto a seguir, habilitar las interrupciones y determinar la posición actual del TCP del robot se ejecuta la rutina de cálculo del punto siguiente.

1. Calculo del punto siguiente:

La rutina de cálculo del punto siguiente se encarga de determinar el siguiente punto que deberá seguir el robot en la trayectoria.

El funcionamiento de la rutina está compuesto a su vez, por varias subrutinas que se encargan de calcular el potencial en tres dimensiones que ejercen cada uno de los campos sobre el punto actual del robot.

Los campos de potencial que se consideran para este trabajo fin de grado son:

- Campo repulsivo inicial: Corresponde con la posición de partida desde la que el robot comenzó a moverse. Este campo es de influencia muy pequeña y se usa para compensar los efectos de un posible mínimo local en la posición inicial del robot.

- Campo atractivo final: Corresponde con el punto objetivo del robot. Es un campo de gran atracción, puesto que como se explicó en la teoría del método, este depende cuadráticamente de la distancia entre el punto objetivo y actual.

- Campo repulsivo mano: Este campo, se ha configurado como un campo obstáculo. Por lo tanto, teniendo en cuanta la explicación teórica del método, el campo generará potencial repulsivo hasta una determinada distancia del obstáculo.
  Para conocer si el robot se encuentra dentro del volumen de influencia Esto supone, que la rutina de detección de intrusión en el volumen de influencia es simplificada y optimizada a una señal digital. Si esta salida no está activada los cálculos de esta rutina no se efectuarán.
  También se debe tener en cuenta que, al estar el punto a seguir dentro de la zona de obstáculo existe riesgo de cancelación de potencial y una aproximación crítica a la mano del cirujano. Para evitarlo se ha establecido el término proporcional del campo con un valor muy alto.
  Se debe tener en cuenta que los campos repulsivos generados varían entre una esfera y un paralelepípedo en función de la posición adoptada por el volumen de la mano del cirujano.

Representamos en la Tabla 11 un resumen con las características de cada campo de potencial:

Campo de potencial	Tipo	Cte proporcional	Tipo
Mano	Repulsivo	η = 9000	Volumétrico
Punto final	Atractivo	ξ = 90	Puntual
Punto inicial	Repulsivo	η = 0.9	Puntual

Tabla 11 Constantes utilizadas en los campos de repulsión

Una vez, se han obtenido las fuerzas ejercidas por todos los campos de potenciales sobre el extremo del robot estos se suman y se calcula el módulo del sumatorio.

Para obtener el siguiente punto con el paso deseado bastará con dividir el paso entre el sumatorio de fuerzas para obtener una constante.

El punto siguiente será la suma de las posición actual más el sumatorio de fuerzas multiplicado por la constante que acabamos de calcular. Esta constante, junto con el resto de fuerzas se calcula de nuevo en cada ciclo.

Una vez obtenido el punto siguiente se implementa una última capa de seguridad para seguridad para garantizar que el robot no colisione con la mesa se establece limitando el movimiento en el eje Z del robot cuando la distancia entre la cota Z del sdr robot y el tcp de la herramienta es menor del 15 % de la distancia en Z entre el sdr robot y el sdr pelvitrainer.

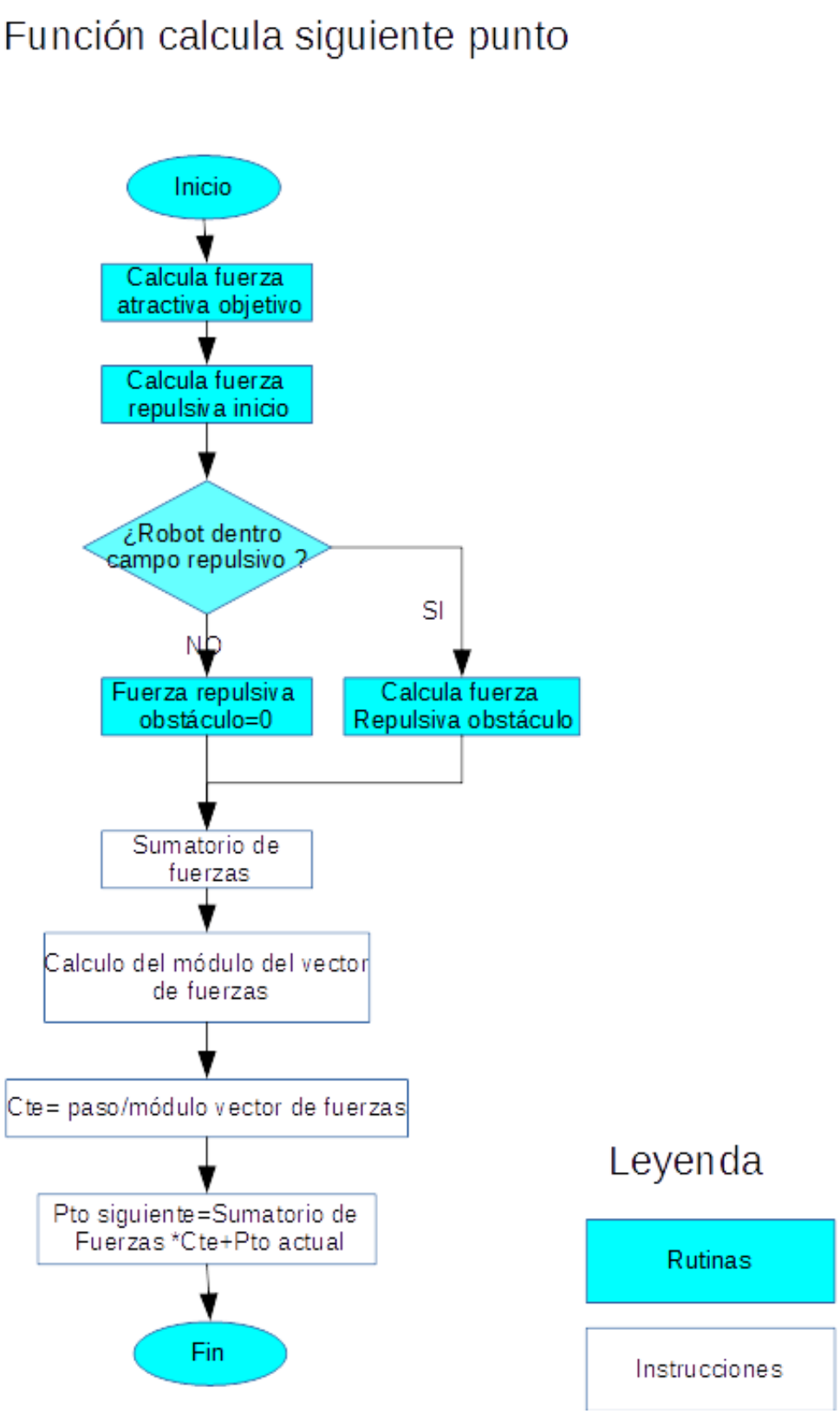

Figura 40 Función calcula siguiente punto

4.4.5. Limitaciones en la implementación

Como hemos visto, el método de los potenciales es un método de cálculo de trayectorias en tiempo real, no obstante se han detectado algunas limitaciones a tener en cuenta.

Primero de todo es la velocidad de cálculo con la que puede obtener los puntos. Debido a que el procesador del controlador es mono proceso y no habilita gestión de procesos multi hilo

El principal inconveniente de la aplicación de este método al robot consiste, en la dificultad para encontrar maneras eficientes en las que almacenar dinámicamente los últimos puntos definidos por el control de trayectorias. Estos últimos puntos podrían ser utilizados en rutinas de backtraking en caso que el robot quede atrapado en un mínimo local.

La forma más eficiente de almacenamiento de los puntos se en una cola de tipo LIFO de tamaño predeterminado. El funcionamiento de las colas en otros lenguajes de programación está basadas en listas enlazadas, donde los elementos a almacenar están diseminados por toda la memoria RAM y las relaciones de orden se determinan a través punteros a la dirección de memoria del dato. De esta manera, solo es necesario cambiar los punteros de inicio de la cola y del fin de esta. Sin recolocar ningún dato.

El único método factible en RAPID consiste en el almacenamiento de dichos puntos en un vector de tamaño determinado. Sin embargo el vector en la memoria se declara todos sus elementos uno tras otro. Y la forma en la que se accede El vector lleva declarada todas sus posiciones una tras otra en la memoria. En cuanto al sistema de organización de la hipotética cola desarrollada en el vector. Se necesitaría reasignar todos los datos iteración tras iteración en el array.

Por último, también es necesario comprender que en este TFG existe la posibilidad de que el punto a seguir se encuentre dentro del campo repulsivo generado por el obstáculo (la mano del cirujano). Para que no se produzca cancelación de potencial y el robot no invada el volumen ocupado por la mano, se ha dotado al campo de potencial repulsivo generado por el obstáculo una fuerza muy elevada.

5. Conclusiones

Las conclusiones que se obtienen de este trabajo fin de grado son las siguientes:

1. Se ha implementado un proceso de comunicación de datos entre nodos de ROS que ha permitido la comunicación entre los nodos de adquisición del sistema de visión 3D y del guante de datos, con los nodos de clasificación y envío al robot.

2. Se ha desarrollado una comunicación socket de tipo TCP entre ordenador y el robot ABB IRB 120 que ha posibilitado el envío de los datos adquiridos en el ordenador.

3. Se ha implementado un sistema multitarea de gestión de los datos recibidos a través del socket en el robot IRB 120. Todas las variables que almacenan los datos se han declarado en un módulo que es compartido por todas las tareas.

4. Por último se han utilizado los datos provenientes de la posición del dedo meñique, y los parámetros que definen la posición y volumen de la mano del cirujano, para implementar una estrategia de seguimiento del dedo meñique evitando colisiones.

5. Para garantizar que el TCP del robot jamás toca a la mano se ha definido una zona mundo que se adapta al volumen simplificado (esfera o paralelepípedo) que ocupa la mano en el espacio. La rutina de supervisión de la zona mundo activa una señal de parada del robot si su TCP va a penetrar dentro de dicha zona.

6. La estrategia que se ha utilizado para el seguimiento del dedo meñique ha sido el método de los potenciales, dotando a la punta del dedo meñique con un potencial atractivo. Se ha caracterizado el volumen ocupado por la mano con un campo de potencial repulsivo normal a su superficie. Para delimitar el volumen de influencia del potencial repulsivo ejercido por la mano se ha caracterizado otra zona mundo mayor, que engloba a la anterior y es del mismo tipo. Entre las superficies de ambas zonas será donde el potencial repulsivo será tenido en cuenta. El campo de potencial repulsivo ha sido caracterizado con valores muy altos para que el TCP del robot no colisione con la mano del cirujano.

Como aporte personal este TFG me ha permitido fijar conocimientos de comunicación de equipos utilizando sockets TCP, así como fijar una estructura de envío de datos.

También me ha permitido conocer y utilizar ROS, y profundizar mi conocimiento del robot ABB IRB 120. Estos conocimientos de programación de robots suponen una ayuda de cara a mi futura la inserción laboral.

6. Propuestas de mejora

En este capítulo se enumeran las opciones de mejoras posibles que se han encontrado durante el desarrollo de este TFG. Se abordan desde pequeñas mejoras, hasta la reubicación de algunas partes del software del controlador del robot en el ordenador.

- Reducir el número de datos enviados al robot:

Una primera propuesta consiste en reducir el número de datos enviados al robot. De esta manera el flujo de comunicación entre el ordenador y el robot se puede reducir y ganar rapidez en el sistema.

- Ubicar el control de trayectorias en el ordenador:

Otra propuesta consiste en ubicar el control de trayectorias fuera del robot, dejando a este como mero receptor y ejecutor de las posiciones que vienen del control de trayectorias. De esta forma, además de reducir el número de datos enviados podemos disponer de mayores recursos de procesamiento.

Para este caso, al disponer de mayores recursos y mejores medios de gestión de ellos, se pueden implementar métodos de backtracking en el control de trayectorias al existir la posibilidad de crear pilas de tipo LIFO que almacenen las N posiciones anteriores con un coste de recursos mínimo.

Otra de las ventajas de disponer de mayor número de recursos se traduce al poder reducir el paso entre dos puntos adyacentes calculados traduciéndose una trayectoria más suave.

- Utilización de un método de control de trayectorias alternativo.

La última propuesta descrita en este TFG consiste en implementar otro método de control de trayectorias distinto al método de los potenciales. El motivo radica en que al estar el objetivo tan cerca del obstáculo el existe una cancelación de potencial. Otra forma de solucionar este problema de distinta forma a como se ha desarrollado en este TFG consiste en emplear otro método de control de trayectorias.

Una de las opciones a tener en cuenta es el desarrollo de una arquitectura de control de dos capas, una reactiva capaz de responder rápidamente ante cambios del entorno en tiempo real. Y otra deliberativa que buscará la trayectoria entre el punto actual y el objetivo. De esta forma, la capa deliberativa permite encontrar una trayectoria óptima mientras que la capa reactiva responde ante cambios del entorno.

7. Bibliografía

7.1. Manuales de ABB

- Instrucciones, funciones y tipos de datos de Rapid Software de controlador IRC5. Manual de referencia técnica de ABB.
- Descripción general de Rapid. Software de controlador IRC5. Manual de referencia técnica de ABB.
- Herramientas de ingeniería. Robotware 5.0. Manual de aplicaciones de ABB.
- Robotstudio versión 5.13. Manual del operador de ABB.
- Curso de Programación de Robots ABB. Autores: Alberto Herreros y JC Fraile. Escuela de Ingenierías Industriales. Universidad de Valladolid. 2016.

7.2. Publicaciones

- Fu-guang, D., Peng, J., Xin-qian, B., & Hong-jian, W. (2005). AUV local path planning based on virtual potential field. In Mechatronics and Automation, 2005 IEEE International Conference (Vol. 4, pp. 1711-1716). IEEE.

- Barraquand, J., Langlois, B., & Latombe, J. C. (1992). Numerical potential field techniques for robot path planning. IEEE Transactions on Systems, Man, and Cybernetics, 22(2), 224-241.

- Safadi, H. Local Path Planning Using Virtual Potential Field, McGill University School of Computer Science, 2007. Available at: http.

- Susa, J., & Acosta, D. A. R. (2012). (ABEO)-algoritmo bioinspirado de evasión de obstáculos. Revista Tecnura, 13(25), 36-47.

- Santos, L., González, J. L., Turiel, J. P., Fraile, J. C., & de la Fuente, E. Guante de datos sensorizado para uso en cirugía laparoscópica asistida por la mano (HALS). Actas de las XXXVI Jornadas de Automática, 2 - 4 de septiembre de 2015. Bilbao, pp 779-785.

- Santos, L., González, J. L., Turiel, J. P., Fraile, J. C., & de la Fuente, E. Guante de datos sensorizado en sistema robótico colaborativo para cirugía laparoscópica asistida por la mano.

7.3. Libros

- Russell, S., & Norvig, P. (2004). Inteligencia Artificial Un Enfoque Moderno.

7.4. Webgrafía

http://www.lcc.uma.es/~eat/services/i_socket/i_socket.html
http://www.ros.org/about-ros/
http://wiki.ros.org/ROS/Tutorials
http://www.quirurgica.com/cirugia-robotica-2/
http://www.cun.es/.imaging/dmsChain/dms/cun/infograficos/Tecnologia/2012-cirugia-robotica-davinci-max.jpg
http://www.proyecto-broca.es/documentos/jornadas-salud-investiga.pdf
http://www.proyecto-broca.es/proyecto/
http://www.roboticamedica.uma.es/src-hals/?project=guante-inteligente
http://www.5dt.com/data-gloves/
https://www.logitech.com/es-es/product/hd-webcam-c310
https://www.universal-robots.com/es/productos/robot-ur3/

Anexo 1: Configuración del robot en multitarea

1. Resumen

En este trabajo fin de grado es condición indispensable que el robot pueda recibir datos a través del socket a la par que este se mueve. En este momento se hace necesario configurar al robot en modo multitarea.

El propósito de la configuración multitarea es lograr la ejecución de varias tareas al mismo tiempo. La forma en que el sistema ejecuta las tareas es en pseudoparalelo. Esto quiere decir que el sistema ejecutará una o varias sentencias de una tarea, desalojará el kernell almacenando el estado del programa en un registro de estados. Y pasará a ejecutar la sentencia o sentencias de las siguientes tareas volviendo a repetir este proceso.

Por razones de capacidad, Rapid termite crear veinte tareas como máximo.

2. Crear nuevas tareas en el robot

Para crear dos o más tareas en paralelo, debemos entrar en el apartado controlado situado en la barra superior de robot studio. Hecho esto, nos aparecerá en la parte de la izquierda el controlador de la estación y el árbol de archivos que cuelga de él. Se debe desplegar el apartado de configuración, y seleccionar el apartado controller. Una vez aquí, se desplegará en la parte central de la pantalla una menú en el que clicaremos con el botón derecho el apartado de Task y seleccionaremos nuevo task.

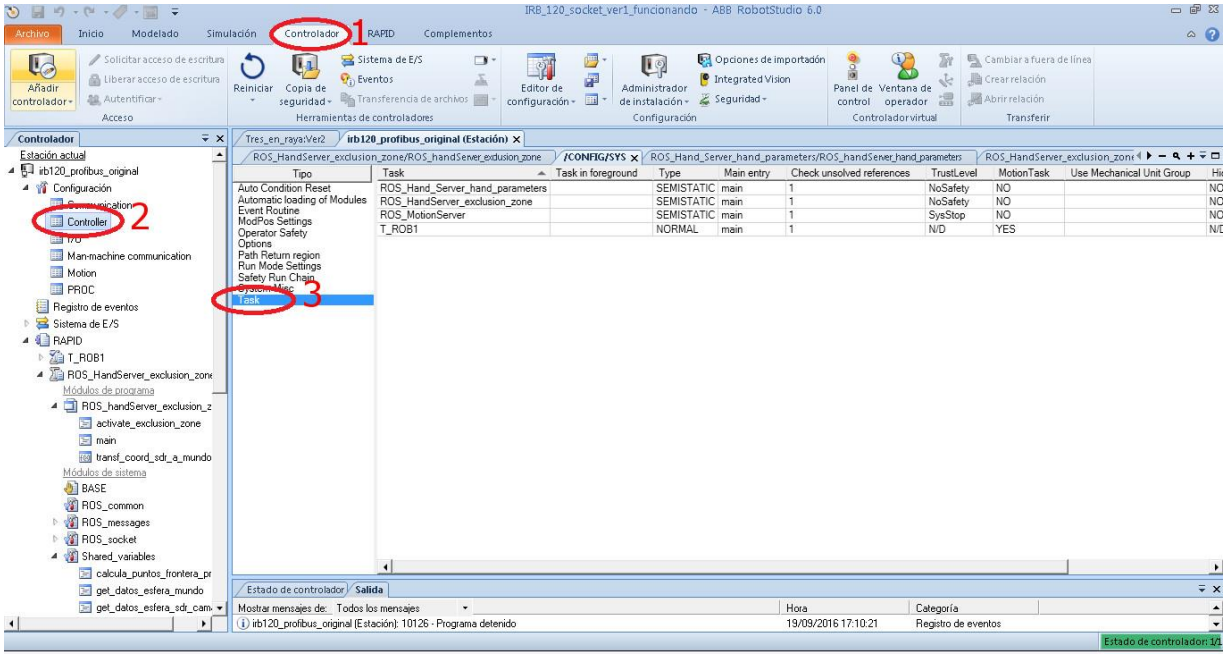

Figura 41 Crear una nueva tarea

3. Descripción de parámetros

Task: Nombre de la tarea, este debe ser único no pudiendo coincidir ni con variables, unidades mecánicas o con otra tarea.

Task in foregroung: Definimos si la tarea forma parte de otra tarea superior. Esta tarea se ejecutará siempre y cuando en la tarea principal no sea necesario realizar ninguna operación. Si este campo está vacío la tarea creada tendrá la máxima prioridad.

Type: define el comportamiento de inicio, parada y reinicio de la tarea. Existen tres tipos:

- Normal: El inicio y parada de la tarea se produce manualmente
- Static: La tarea no puede ser parada de forma manual ni por la flexpendant, ni por parada de emergencia. Tras un reinicio en caliente del controlador la tarea recupera su estado de ejecución anterior al reinicio.
- Semistatic: Al igual que en el caso anterior, la tarea no puede ser parada de forma manual. Sin embargo, esta se reiniciará desde el principio tras un arranque en caliente.

Main entry: Se puede definir el nombre de la rutina principal de la tarea. Por defecto se dejará main

Check unresolved references: Este parámetro determina si la tarea puede aceptar referencias sin resolver mientras se enlaza un módulo. En caso afirmativo, se establecerá un cero o un uno en caso negativo.

Trustlevel: define el comportamiento del sistema cuando una tarea STATIC o SEMISTATIC es detenida. Existen cuatro tipos:

- No safety: El programa de la tarea se detendrá.

- SysFail: Si el programa de esta tarea se detiene, todos los programas de las tareas de tipo normal se pararán, y el resto de tareas continuarán su ejecución si es posible siendo necesario un arranque en caliente para restablecer el estado de paro del sistema.
- SysHalt: Si el programa de esta tarea se detiene, todos los programas de las tareas de tipo normal se pararán, siendo necesario un arranque en caliente para restablecer el estado de paro del sistema.
- SysStop: Si el programa de esta tarea se detiene, todos los programas de las tareas de tipo normal se pararán pero son reiniciables

Motion Task: Indica cuales de las tareas puede controlar el movimiento del robot. Si la opción MultiMove no está habilitada, sólo una tarea podrá controlar el robot.

Una vez creada una nueva tarea, será necesario un arranque en caliente para poder empezar a utilizarla.

ESCUELA DE INGENIERÍAS INDUSTRIALES

Universidad de Valladolid

Anexo 2. Manual de usuario ROS_ABB_COM

Herramienta comunicación ROS ABB

EII – DISA

DESCRIPCIÓN:

> **Objetivos:** El objetivo de esta herramienta es facilitar la comunicación entre el controlador IRC5 de ABB, el guante de datos 5dt data glove y el sistema de visión 3D, y establecer así un control de trayectorias que siga el dedo meñique de la mano del cirujano

DATOS DE ENTRADA:

Los datos de entrada serán proporcionados por el guante 5dt data glove y por el sistema de visión 3D

CONTACTOS:

Para cualquier problema contactar con:

Sergio García

INFORMACIÓN COMPLEMENTARIA:

Documento desarrollado por: Sergio García.

Fecha de la versión actual: 19/03/207

Versión: V1_0

1 Presentación de la herramienta ROS_ABB_COM

La herramienta ROS_ABB_COM permite al usuario una comunicación efectiva entre el guante de datos 5dt data glove y el sistema de visión 3D. Esta herramienta forma parte de un conjunto de herramientas software cuyo fin es implementar una aproximación a un sistema robótico colaborativo orientado a cirugía laparoscópica.

La herramienta está compuesta a su vez por dos paquetes de código. El paquete Hand_c_sergio desarrollado en el framework ROS y estará alojado en el ordenador, y el paquete ABB_Server desarrollado en Rapid, y su contenido estará cargado en el controlador del robot

Objetivos de la herramienta

La herramienta ROS_ABB_COM permite:

- Establecer un canal de comunicación entre el guante de datos 5dt glove y el controlador IRC5 de ABB.
- Establecer un canal de comunicación entre los sistemas de visión 3d y el controlador IRC5 de ABB
- Establecer la trayectoria que seguirá el dedo índice con el efector del robot.

2 Herramienta ROS_ABB_COM: paquete hand_c_sergio

Este paquete establece comunicación con el guante de datos 5dt data glove y las cámaras de visión artificial, encapsulará la información y enviará los datos a través de dos rutinas.

2.1. Requisitos

El Framework ROS sólo puede instalado en sistemas operativos Ubuntu y Debian.

Se recomienda que el sistema operativo sea Ubuntu 14.04 y ROS índigo igloo.

2.2. Comprobaciones previas

1. Verificación de la ubicación del paquete

El usuario que desee iniciar el paquete Hand_c_sergio deberá comprobar primero que esté correctamente ubicado en el directorio de desarrollo de ROS. Para ello deberá dirigirse a la carpeta de desarrollo de paquetes de ROS catkin_ws.

El usuario deberá encontrar el paquete Hand_c_sergio descomprimido al acceder a la subcarpeta "src". En caso contrario, deberá descomprimirlo en esta ubicación.

2.3. Compilación del paquete

1. Compilación del paquete

Una vez el usuario ha iniciado el núcleo de ROS deberá compilar los paquetes. Para ello, el usuario deberá iniciar una nueva terminal y dirigirse a la carpeta de desarrollo de ROS introduciendo el comando "cd catkin_ws".

En este punto el usuario deberá teclear la instrucción "Catkin_make" para compilar el paquete.

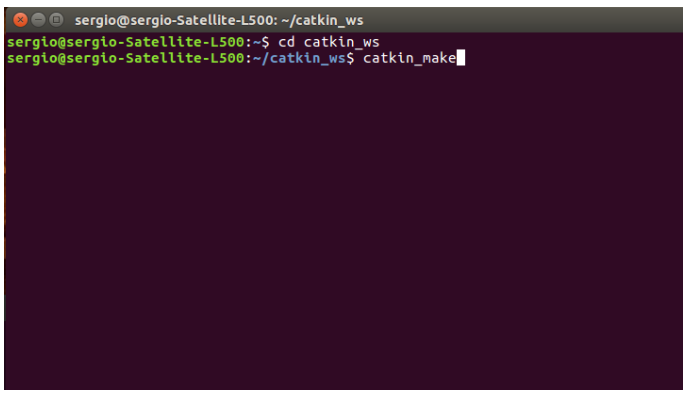

Figura 42 Instrucción de compilado

Tras introducir la orden la consola mostrará por pantalla el progreso de la compilación en tanto por ciento y los eventos generados.

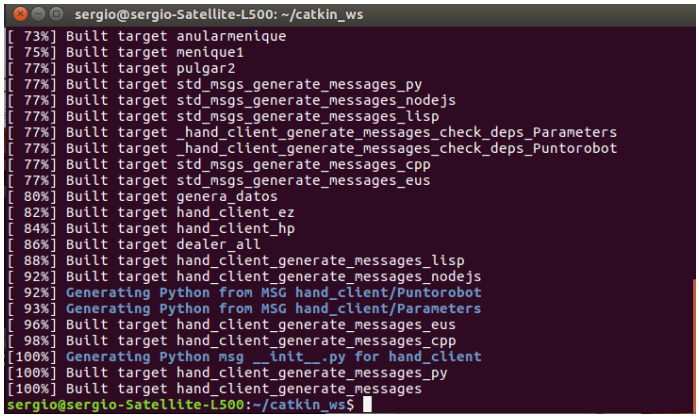

Figura 43 Compilado satisfactorio

Si la compilación ha sido satisfactoria en la consola la consola del usuario deberá aparecer como en la Figura 44

2. Errores de compilación

En el caso que algún error de compilación apareciera sin que el usuario haya modificado ninguna línea del código deberá realizar las siguientes acciones:

- Dirigirse a la carpeta "build" que está ubicada dentro de la carpeta "catkin_ws" y borrar la carpeta con el mismo nombre del paquete Hand_c_sergio.
- Dirigirse a la carpeta "devel" ubicada dentro de la carpeta "catkin_ws". Acceder a la subcarpeta "lib" y borrar la carpeta con el mismo nombre del paquete Hand_c_sergio.
- Retornar a la carpeta cartkin_ws y ejecutar catkin_make de nuevo.

Nota: La compilación podrá omitirse si el usuario ya compiló el código y no ha habido ningún cambio sobre este.

2.4. Lanzamiento de los programas

Para iniciar cualquier rutina el primer paso a realizar es el inicio de núcleo de ROS.

1. Inicio del núcleo de ROS

Tras comprobar que l paquete está correctamente ubicado. El usuario debe abrir una terminal de sistema (CTRL+ ALT+ T) e introducir el comando "Roscore"

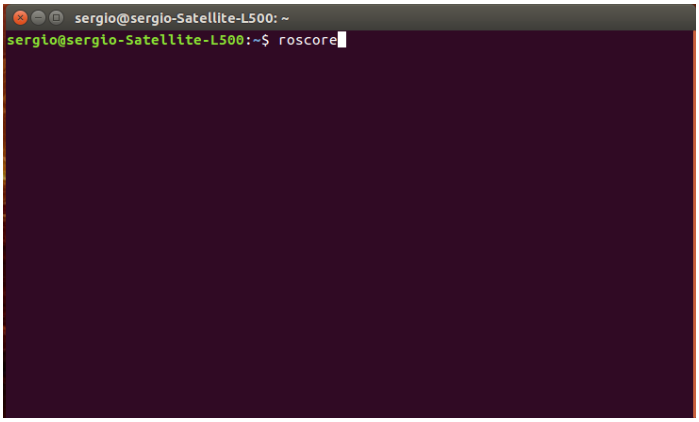

Figura 44 Inicio del núcleo de ros

Tras la ejecución del comando, el usuario comprobará que la ejecución ha sido satisfactoria si la consola aparece igual que en la Figura 44

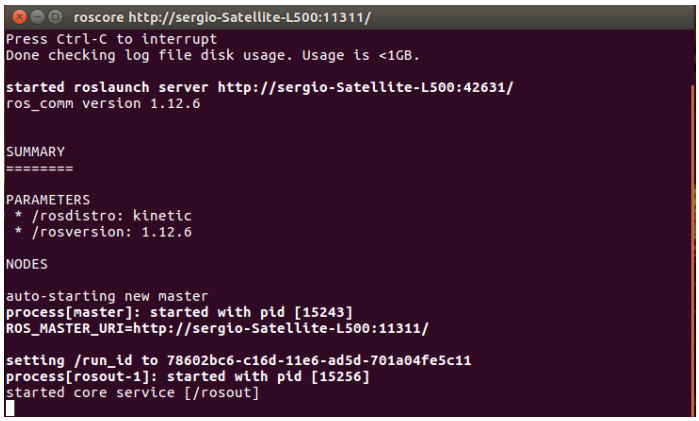

Figura 45 Núcleo operativo

Esta terminal NO debe cerrarse durante la ejecución del paquete

Una vez ha sido iniciado el núcleo los programas pueden iniciarse sin errores.

Para iniciar cualquier programa de cualquier paquete de ROS el usuario deberá abrir una nueva terminal y dirigirse a la carpeta de desarrollo de ROS (catkin_ws). Tras esto deberemos ejecutar la instrucción ". devel/setup.bash"

Por último, para ejecutar un programa dentro de un paquete se debe usar la instrucción rosrun:

Rosrun "nombre del paquete" "nombre del programa sin extensión"

El paquete Hand_c_sergio está compuesto por tres rutinas que explicamos a continuación:

4. Dealer_all: Este programa adquiere todos los datos procedentes tanto de la cámara como del guante de datos y los envía a las tareas Hand_client.

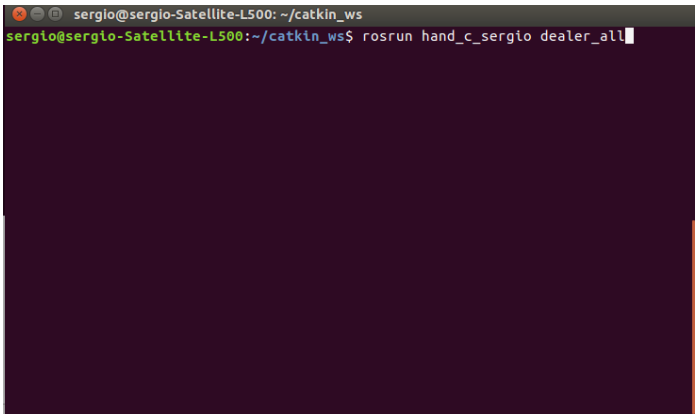

Figura 46 Inicio de la rutina dealer_all

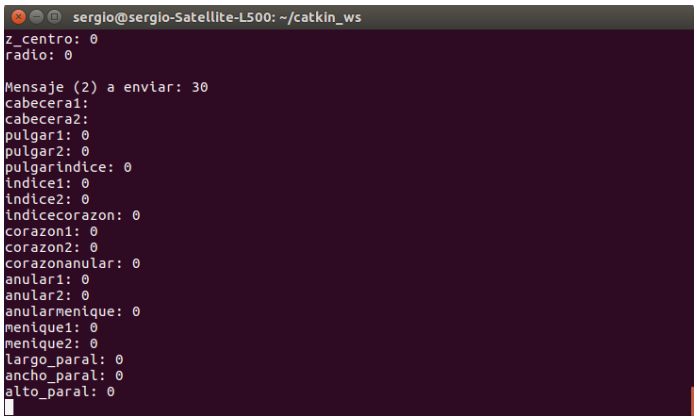

Figura 47 Instrucción dealer_all ejecutándose

5. Hand_client_ez: esta tarea adquiere los datos de la posición de las puntas de los dedos y los datos que definen el volumen ocupado por la mano del cirujano en el espacio. Los datos son enviados controlador del robot a través

de un socket. Esta tarea es crítica de cara al desempeño del control de trayectorias en el robot. Por lo que, no se recomienda iniciar la tarea motora del robot sin conseguir antes establecer la comunicación. Este tema se tratará más a fondo en la sección 3 de este manual.

6. Hand_client_hp: Esta tarea adquiere los datos de los ángulos formados por los metacarpianos y sus correspondientes falanges proximales. Y el de estas últimas con las falanges medias. Todos estos datos son enviados al controlador IRC5 a través de un socket.

El orden de lanzamiento debe ser el mismo que el aquí mostrado.

3 ROS_ABB_COM: ABB_Server

El paquete ABB_Server constituye las tareas que serán ejecutadas en el controlador IRC5 del robot ABB.

3.1. Requisitos

El requisito indispensable para ejecutar el programa del robot será:

- Robot ABB IRB 120
- Controlador IRC5
- Flexpendant
- PC con sistema operativo Windows y el programa de ABB robotstudio.

3.2. Configuración previa

Primero de todo el usuario deberá configurar el robot para permitir la ejecución de más de una tarea a la vez.

1. Añadir un nuevo controlador

El usuario deberá abrir archivo de estación robotizada.

Tras esto, el usuario deberá ir a la pestaña “controlador”, en la parte de la izquierda encontraremos la opción “Añadir controlador”

Figura 48 Pestaña controlador

Seleccionando sobre esta última, aparecerá un menú desplegable donde seleccionaremos la opción “Añadir un nuevo controlador” y nos aparecerá una lista de los controladores disponibles.

En esta lista, nosotros seleccionaremos el controlador llamado “Sistema profinet claves” y pulsaremos aceptar

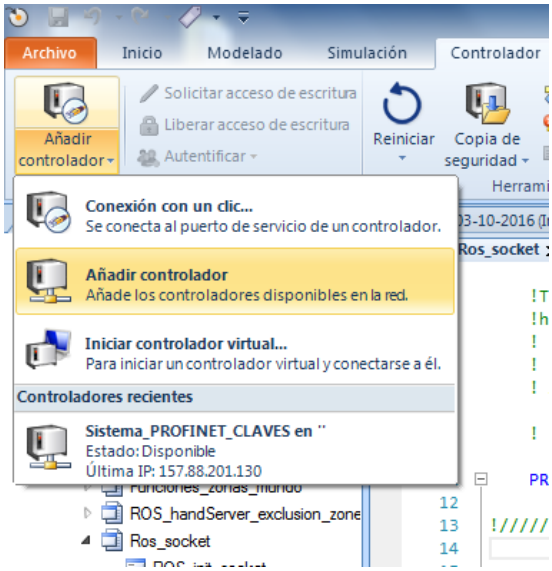

Figura 49 Pestaña controlador

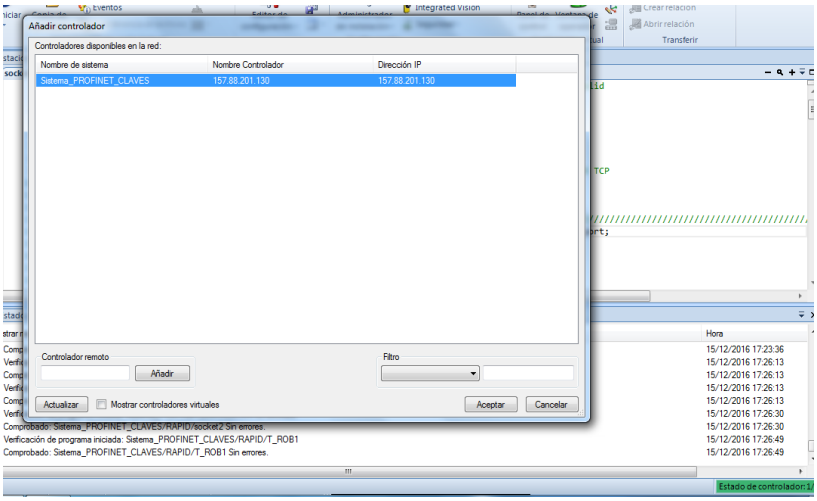

Figura 50 Controladores disponibles

Tras esto, en la barra (controlador) situada izquierda de la pantalla nos aparecerá la lista de los controladores disponibles donde aparecerá el controlador del robot.

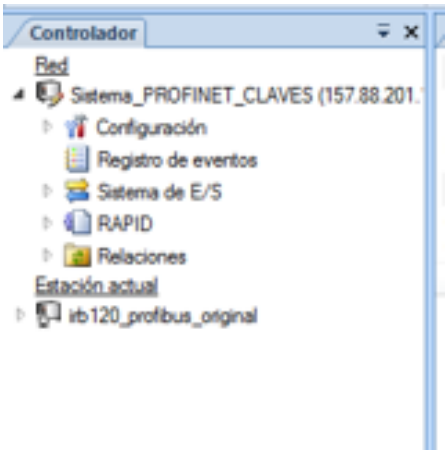

Figura 51 Controlador real y controlador simulado

2. Solicitar acceso de escritura

Una vez aquí el usuario deberá solicitar permiso de escritura para modificar el código o la configuración del robot.

Para ello deberá seleccionar el controlador del robot físico () y seleccionar la opción "Solicitar acceso de escritura"

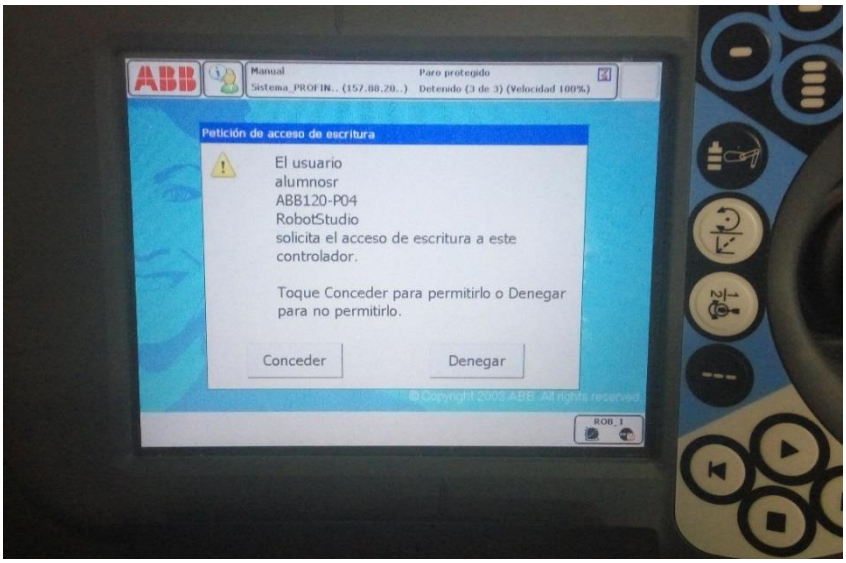

Figura 52 Solicitud de acceso de escritura

Para poder acceder a la configuración del robot, tras solicitar el acceso de escritura deberemos ratificar el acceso de escritura dirigiéndonos a la flexpendant y pulsando en la opción conceder. Acto seguido en la misma flexpendant el usuario verá un mensaje donde especifica cual es el usuario que tiene acceso a la configuración del controlador y habilita la opción de desautorizar dicho acceso.

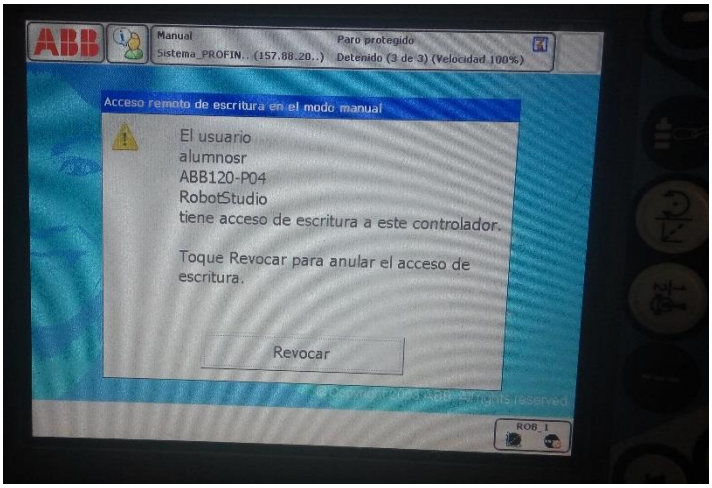

Figura 53 Usuario con acceso de escritura

3. Configurar el controlador en modo multitarea

Una vez el usuario ya tiene concedido el acceso de escritura deberá volver al programa robot estudio y crear dos tareas nuevas:

Para ello, el usuario deberá dirigirse a la pestaña "controlador" de la barra superior (1), desplegar la opción "configuración" del controlador a modificar y hacer doble clic sobre la opción controler (2).

Aparecerá una nueva ventana en el centro de la pantalla donde seleccionaremos la opción Task (3) que nos mostrará las tareas actualmente definidas en el robot.

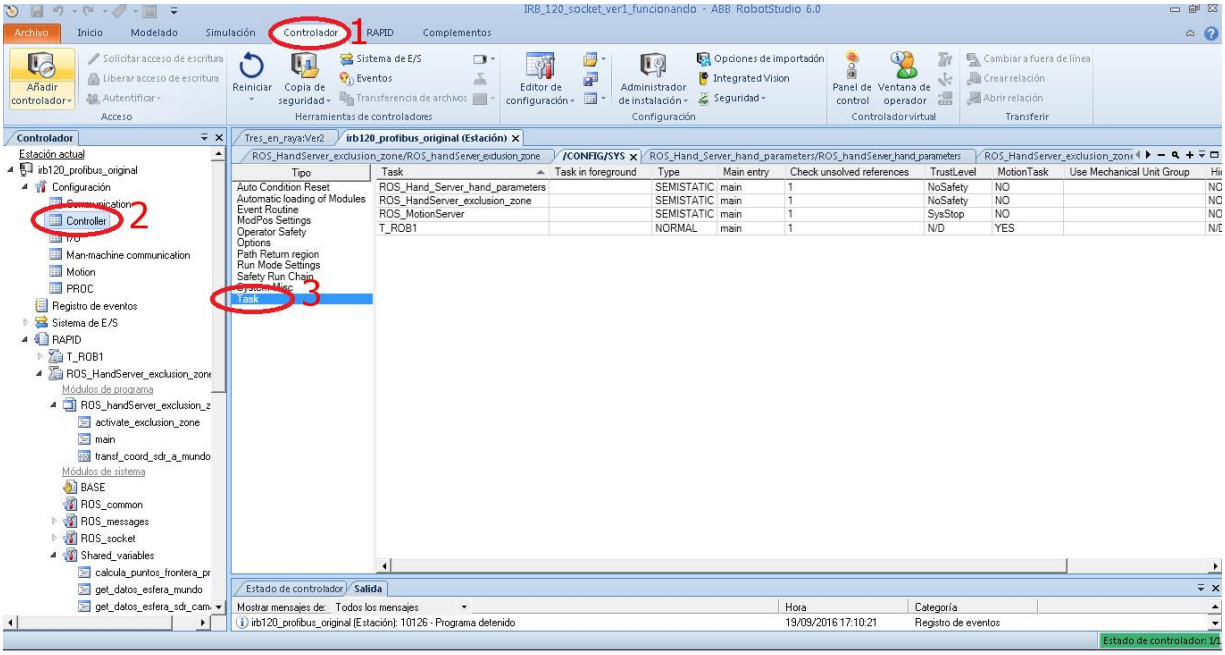

Figura 54 Crear una nueva tarea

Para crear una nueva tarea el usuario deberá hacer click derecho sobre la opción "Task" y seleccionar "Nuevo Task".

Las tareas a crear serán llamadas Hand server ez y Hand server hp. Ambas sean declaradas de tipo "Semistatic" y nivel de seguridad "sysfail".

Al terminar la configuración deberemos reiniciar el sistema con un "arranque en caliente" cuya opción se encuentra en la ventana "controlador" y pulsando el botón reiniciar en la parte superior. Terminar la configuración, el robot debería iniciarse de nuevo sin ningún problema.

4. Solución de errores

En ocasiones, tras el reinicio, el controlador IRC5 entra en modo de fallo de sistema. Si esto nos ocurriese deberíamos restaurar el sistema a la configuración original del controlador antes de iniciar la configuración.

Para reiniciar el controlador se deberá acceder al apartado "copia de seguridad y restauración" de la flexpendant y seleccionar una de las carpetas de backup existentes. Se recomienda realizar una copia del sistema previa a la configuración.

Una vez restaurado el sistema debemos volver al estado inicial de este apartado.

En caso que, el error persista, el tipo de tarea podrá ser configurado como NORMAL. Pero se hace saber al usuario que esta configuración NO se recomienda.

3.3. Carga de los programas en la tarea

La carga de los programas en las respectivas tareas del robot puede llevarse a cabo de varias formas. En este manual se explican las dos más sencillas para el usuario.

1. Crear una relación entre estaciones:

Para utilizar esta opción el usuario deberá haber descomprimido primero la estación que se proporciona con este paquete de comunicación. Si no lo hizo antes, deberá utilizar el segundo método. O cerrar la estación actual y abrir el archivo proporcionado. En este caso no hará falta volver a configurar el robot. Pero sí solicitar nuevamente el acceso de escritura.

La relación entre estaciones consiste en una vía rápida de traspaso de código desde una estación simulada hasta una estación real. O a la inversa.

Para ello el usuario debe dirigirse nuevamente a la pestaña "controlador" y buscar en la parte superior derecha la opción "crear relación" donde aparecerá el

controlador simulado como controlador principal. En el segundo controlador que estará inicialmente vacío, el usuario seleccionará controlador del robot.

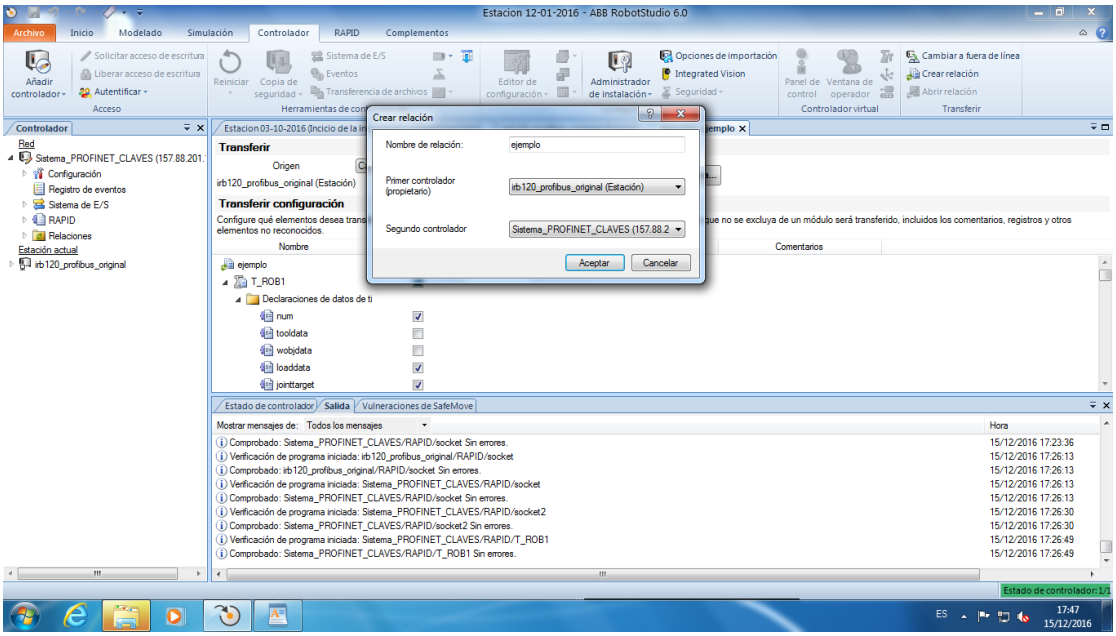

Figura 55 Crear una relación entre estaciones

Tras esto, se muestra una nueva pestaña donde indica los elementos a transferir entre los controladores. En la parte superior se indica el sentido de la transferencia, siendo posible cambiarlo. Y el botón “Transferir ahora que inicia la transferencia”

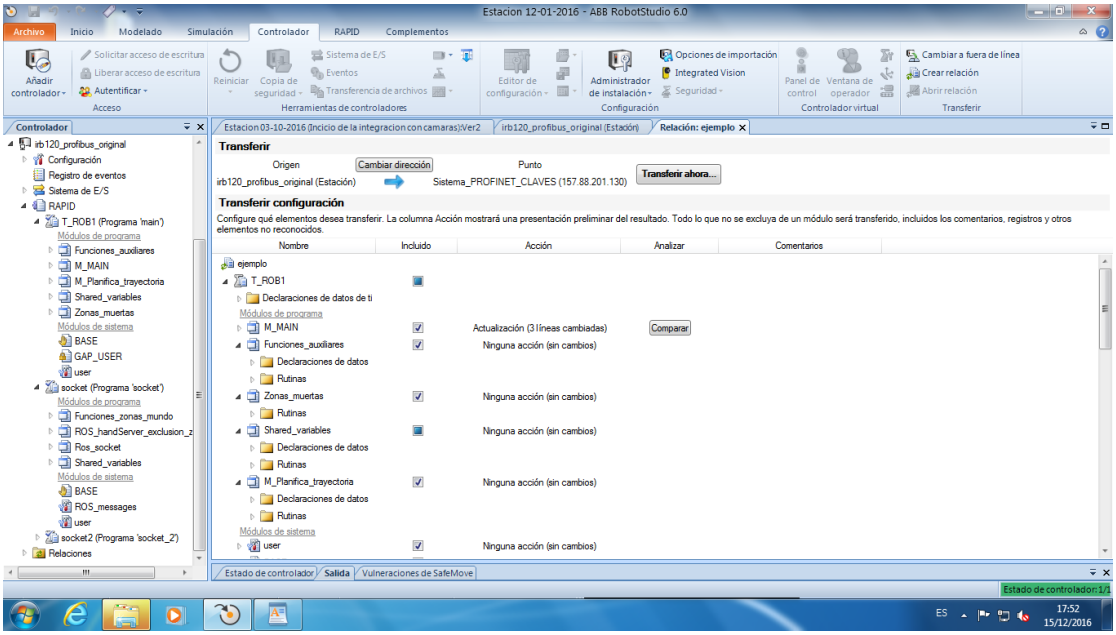

Figura 56 Establecer una relación entre estaciones

2. Cargar los programas directamente

El otro método de carga de los programas en las tareas consiste en cargar los programas uno a uno.

Para realizar este método el usuario debe desplegar la opción Rapid del controlador del robot situado en la barra izquierda, seleccionar una a una las tareas haciendo clic sobre ella y seleccionando la opción cargar programa. Una ventana se abrirá para que el usuario seleccione el programa que desea cargar en formato comprimido. Los programas a cargar serán proporcionados en un archivo comprimido con el nombre “Programas_ABB”

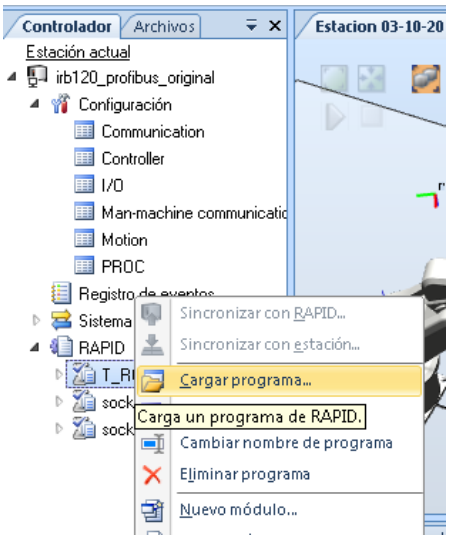

Figura 57 Carga de un programa en una tarea

3.4. Inicio del robot

Tras finalizar la configuración la tarea motriz del robot será iniciado al pulsar el botón de inicio de la flexpendant. Las tareas de recepción de datos serán automáticamente iniciadas y esperarán a establecer una comunicación.

Antes de iniciar el programa, el usuario lanzar los procesos de envío de datos desde ROS situados en el ordenador. Si la comunicación ha sido correctamente establecida, un mensaje de cada tarea de recepción aparecerá en la flexpendant indicando que está asociada a una IP. En este momento el usuario podrá iniciar la tarea motriz del robot.

Si el usuario hubiera configurado las tareas de recepción de datos como "Semistatic" estas no se inician de manera automática y se iniciarán cuando el usuario presione el botón de inicio de la flexpendant. En este caso, si la tarea Hand Server EZ no ha establecido una conexión antes que el robot haya terminado de posicionarse en el lugar de introducción de la herramienta, el sistema deberá pararse puesto que no puede poner en peligro tanto al cirujano como al paciente.

Anexo 3: Guía de desarrollo de comunicaciones socket en el robot ABB irb 120.

1. Objetivo

El objetivo de esta guía es mostrar los pasos necesarios para desarrollar aplicaciones de comunicación socket del robot ABB IRB 120 con otros equipos.

La comunicación mediante sockets posibilita:

- Envío de ordenes desde un equipo remoto al robot
- Envío desde el robot a un equipo remoto de la información (señales de error, posición actual …)

Un ejemplo claro de aplicación de los sockets en el robot puede ser el envío de datos tales como el muestreo del ángulo que adoptan cada uno de los ejes del robot en cada instante. Otros ejemplo

2. Base teórica

Un socket es una interfaz de entrada/salida de datos que permite la intercomunicación entre procesos. Los procesos pueden estar ejecutándose en el mismo o en distintos sistemas, unidos mediante una red (con o sin acceso a internet). La tecnología socket se basa en los mecanismos de entrada/salida de archivos en UNIX. Es decir, utiliza los descriptores de ficheros estándar en Unix. Este mecanismo de comunicación entre procesos, apareció por primera vez en la distribución 4.1.c de Berkeley Software Distribution.

La comunicación entre procesos remotos, es mucho más eficaz y fácil de implementar ya que el socket conceptualmente se trata como si fuera un fichero. Los sockets requieren para funcionar de n IP's y n puertos (una pareja IP/puerto en cada máquina a conectar) a través del cual se enviará la información. Los sockets, además, se pueden caracterizan por el dominio (local o internet), el tipo (con o sin conexión).

Desde el punto de vista de la arquitectura de red, un socket permite implementar u interfaz con el nivel de transporte, usando los servicios de protocolo TCP o UDP. Hay muchos tipos de sockets en Internet, pero solo nos detendremos en dos:

- **Sockets de Datagramas (SOCK_DGRAM):** Estos sockets se basan en el protocolo UDP ("User Datagram Protocol"), y no necesitan de una conexión

accesible como los sockets de Flujo. Se construye un paquete de datos con información sobre su destino y se enviará afuera, sin necesidad de una conexión. El protocolo UDP, por tanto, al ser un protocolo sin conexión, no garantiza que los paquetes de datos lleguen el orden correcto.

- **Sockets de Flujo (SOCK_STREAM):** Este protocolo nos asegura la llegada en orden de todos los objetos durante la transmisión de datos. Estos sockets usan TCP ("Transmissión Control Protocol"). Además establece mecanismos para reenvió para datos que no se han recibido correctamente.

Nota: El controlador IRC5 del robot ABB sólo permite el uso sockets bajo el protocolo TCP (Sockets de flujo).

3. Desarrollo

3.1 Concepción del sistema de comunicación:

Primero de todo el desarrollador debe tener en cuenta el sentido del flujo de datos en la comunicación. El desarrollador debe determinar, si el robot será receptor de información, emisor o ambos. Además se deberá establecer cuál de los equipos llevará el rol de servidor.

Se entiende que la comunicación debe ocurrir en paralelo a la ejecución de un programa en el propio robot. Por lo que se deberá configurar el robot en modo multitarea, asignando como tarea principal el programa de movimiento de robot. Mientras que las N tareas restantes serán asignadas con programas orientados a la comunicación.

Por simplicidad, se recomienda que en caso que el desarrollador desee implementar una comunicación bidireccional entre cualquier equipo y el robot utilice dos sockets distintos. Cada uno con un sentido determinado de envío de datos. De esta forma, pese a duplicar los canales de comunicación, se posibilita el envío en paralelo de la información. En el caso que el desarrollador desee implementar una comunicación bidireccional sobre un mismo socket deberá considerar un esquema de interacción entre las máquinas que garanticen la interpretación de la información.

Para desarrollar una rutina capaz de realizar una comunicación socket bajo cualquier lenguaje de programación debemos conocer primero si el equipo que contendrá la rutina se comportará como cliente o como servidor. Las ilustraciones 1 y 2 son esquemas de la estructura típica de un socket configurado como servidor y como cliente.

3.2 Instrucciones necesarias

Una vez escogido el modelo de comunicación que implementará el sistema a desarrollar procedemos a explicar las instrucciones principales que nos permiten establecer una rutina de comunicación por sockets:

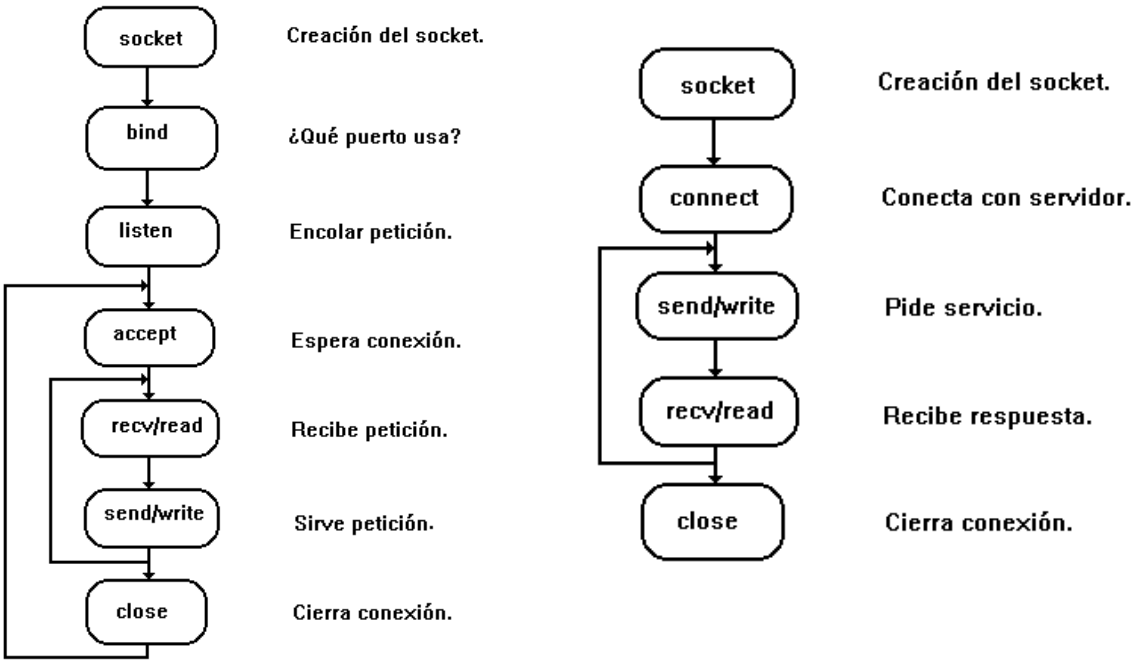

Figura 58 Esquema de un extremo servidor y otro extremo cliente

El controlador IRC5 del robot ABB sólo admite el envío de mensajes por sockets bajo protocolo TCP/IP. Lo que implica mecanismos internos al protocolo que garantizan el envío de mensajes y su recepción en el mismo orden en el que fueron enviados.

1. <u>Crear el socket.</u>
   El extremo del socket se crea con la instrucción SocketCreate encargándose internamente de habilitar los mecanismos necesarios para la comunicación.

```
LOCAL VAR socketdev server_socket;
SocketCreate server_socket;
```

   Como argumento de entrada se le debe de adjuntar una variable de tipo socketdev que actuará como identificador de proceso (handle) del extremo del socket en el servidor. Este handle será exclusivo para cada nuevo extremo de una comunicación socket implementado en el servidor.
   Nota: Para prevenir errores durante el desarrollo de la aplicación, se recomienda comprobar si existe un socket activo asociado al handle deseado

2. **Bind**

   Una vez el dispositivo de comunicación ha sido creado, es necesario configurarlo para que pueda llevarse a cabo la comunicación. La configuración se lleva a cabo utilizando la instrucción SocketBind.
   Como argumentos de la función se especificarán el handle del extremo del socket que deseamos configurar, la dirección IP de la propia máquina que alojará el servidor. Y el puerto a través del cual recibirá los datos.

```
SocketBind server_socket,"157.88.201.130",port;
```

   Nota: Si esta comunicación se va a realizar a través de la red WiFi de la escuela se debe recordar que la red Eduroam permite conexiones a través de los puertos 80 y 8080. Si se desean realizar conexiones por otros puertos distintos a los citados, la conexión deberá ser por vía Ethernet. Utilizando para ello las conexiones que tienen ocupadas los ordenadores del taller de sistemas robotizados.

3. **Listen.**

   Una vez creado el socket y configurado correctamente es necesario configurarlo en modo "Escucha". Esto significa que el socket permanecerá a la espera de una petición de conexión por parte del cliente.

```
SocketListen server_socket;
```

   Si durante la comunicación con un cliente llegan nuevas peticiones de establecimiento de la comunicación por parte de terceros clientes estos serán puestos en cola. Cuando finalice la conexión activa actual el servidor atenderá en orden estas peticiones de comunicación pasando a atender al primer usuario que realizó la petición.

4. **Acept.**

   La instrucción SocketAccept se ejecuta tras la recepción de una petición de conexión por parte de un cliente. Con ella, el servidor, acepta la conexión y habilita la comunicación de datos entre él y el cliente quedando ambos extremos conectados.

```
SocketAccept server_socket,client_socket,\ClientAddress:=client_ip,\Time:=WAIT_MAX;
```

   La instrucción debe contener como mínimo dos argumentos de tipo handle. El primero de ellos es el handle del socket del servidor (server_socket). El segundo servirá para almacenar los datos correspondientes al extremo cliente (client_socket)
   Como parámetro opcional puede almacenar la IP del cliente en una variable de tipo STRING. Para ello deberemos insertar el argumento opcional "ClientAddress" e igualarlo a la variable de tipo STRING donde se almacenará la IP del cliente.

Otro parámetro opcional que posee esta instrucción es el tiempo en segundos que esperará el programa a que se produzca una petición de conexión representado como “Time”. Si este parámetro no se introduce el tiempo por defecto será 60 segundos. Transcurrido este tiempo si no ocurre ninguna petición de conexión se generará un error de tipo “ERR_SOCK_TIMEOUT”. Si el desarrollador desea establecer un tiempo de espera infinito deberá asignar a time la constante predefinida en Rapid “WAIT_MAX”.

5. Rcv/Read

Una vez establecida la conexión socket entre los dos extremos se pueden comenzar a recibir datos desde el otro extremo del socket.
Para la recepción de datos o “lectura del socket” se utiliza la instrucción “SocketReceive”. El argumento principal de esta función es el handle del emisor de la información. Que en este caso el emisor de la información es el cliente.

```
SocketReceive client_socket,\Str:=buffer;
```

Tipo de los datos enviados/recibidos: Por defecto Rapid tolera él envió/recepción de mensajes formados como cadenas de caracteres (\Str), de datos sin formato (\Rawdata) y de datos tipo byte (\Data).

Nota: la máxima longitud que permite Rapid para un mensaje de tipo String es de 80 caracteres. Mientras que los datos sin formato y datos tipo byte se pueden tener una longitud máxima de 1024bytes.

En la imagen superior el contenido del socket es almacenado en la variable de tipo String “buffer” (previamente declarada).

6. Send/write

Cuando el robot desea enviar datos al otro extremo este debe realizarlo con la instrucción “SocketSend”.
Como parámetros de entrada se debe especificar el handle del receptor del mensaje. Y como parámetros opcionales especificar el formato de los datos enviados (String, Rawdata y Data).

```
SocketSend client_socket \Str:=buffer;
```

7. Close

Una vez se ha terminado el proceso de comunicación la instrucción Socketclose se encarga de cerrar la comunicación socket y liberar los recursos prestados para ello.

```
SocketClose server_socket;
```

El único argumento que necesita esta instrucción es el handle de la conexión del extremo ubicado en la propia máquina.

En la imagen anterior, con la instrucción server socket deshabilitamos la conexión socket contenida en el handle serversocket.

8. Connect

La instrucción “SocketConnect” se utiliza en el extremo cliente cuando este quiere realizar la petición de conexión a un servidor.

Los parámetros de entrada de esta aplicación deben ser un handle de dispositivo socket creado pero no enlazado con otro. La dirección IP del servidor en formato String y el puerto de conexión en formato numérico

```
SocketConnect socket1, "192.168.0.1", 1025;
```

9. Conocer obtener el status de un socket.

Para conocer el status que posee un socket, se utiliza la función socketGetStaus cuyo parámetro de entrada es el handle del dispositivo socket.

```
IF (SocketGetStatus(server_socket)=SOCKET_CLOSED)
```

Esta devolverá un valor asociado a una constante de RAPID en función de ese valor conoceremos el estado del socket.

Constante de Rapid	Valor	Estado del Socket
SOCKET_CREATED	1	Creado
SOCKET_CONNECTED	2	Cliente conectado a un host remoto
SOCKET_BOUND	3	Servidor enlazado a una dirección y un puerto locales
SOCKET_LISTENING	4	Servidor a la escucha de conexiones entrantes
SOCKET_CLOSED	5	Cerrado

Tabla 12 Estado del socket

4. Ejemplos de aplicación en Rapid

Mostramos a continuación dos pequeñas rutinas desarrolladas en Rapid que implementan una comunicación socket. El esquema de comunicación es similar a los modelos de cliente y servidor que se han utilizado para explicar la comunicación. La única diferencia entre el código y el esquema radica en el código del servidor que sólo atenderá la primara petición de conexión. En caso de pérdida de conexión cerrará la comunicación y terminará el ciclo de programa.

Para probar su ejecución es necesario con cargar dos estaciones distintas de Robotstudio en el mismo pc. Modificar la dirección IP con la IP del propio ordenador y ejecutar la simulación de ambas estaciones.

```
MODULE Client_socket
    CONST num port:=8080;
    CONST string IP:="157.88.201.130";
    VAR socketdev client_socket;
    VAR string buffer;
    PROC cliente()
        IF (SocketGetStatus(client_socket)=SOCKET_CLOSED) SocketCreate client_socket;
        socketconnect client_socket,IP,port
        WHILE SocketGetStatus(client_socket) <> SOCKET_CLOSED DO
            SocketSend client_socket,\Str:=buffer;
            buffer:="Envío a servidor";
            SocketReceive client_socket,\Str:=buffer;
            TPWrite "buffer:"+buffer;
        ENDWHILE
        SocketClose client_socket;
    ENDPROC
ENDMODULE
```

```
MODULE Servidor_socket
    CONST num port:=8080;
    CONST string IP:="157.88.201.130";
    VAR socketdev server_socket;
    VAR socketdev client_socket;
    VAR string client_ip;
    VAR string buffer;
PROC servidor()
        IF (SocketGetStatus(server_socket)=SOCKET_CLOSED) SocketCreate server_socket;
        IF (SocketGetStatus(server_socket)=SOCKET_CREATED) SocketBind server_socket,IP,port;
        IF (SocketGetStatus(server_socket)=SOCKET_BOUND) SocketListen server_socket;
        SocketAccept server_socket,client_socket,\ClientAddress:=client_ip,\Time:=WAIT_MAX;
        TPWrite "Client at "+client_ip+" connected.";
        WHILE SocketGetStatus(client_socket) <> SOCKET_CLOSED DO
            SocketReceive client_socket,\Str:=buffer;
            TPWrite "buffer:"+buffer;
            buffer:="Envio a cliente";
            SocketSend client_socket,\Str:=buffer;
        ENDWHILE
        SocketClose server_socket;
        SocketClose client_socket;
ENDPROC
ENDMODULE
```

5. Desarrollo de una comunicación en otro SO bajo otro lenguaje

En esta guía se ha explicado cómo implementar comunicación socket en Rapid. Puede darse el caso en que el equipo con el que el robot intercambie datos lleve otro sistema operativo distinto al controlador IRC5. En ese caso, la rutina de comunicación de ese extremo será implementada en otro lenguaje de programación.

El único contra existente para el desarrollador al implementar esta vía consistirá en los mismos tipos de datos que Rapid tiene permitidos (Sting, rawbytes y bytes). La forma en la que se declaran sockets es similar en todos los lenguajes de programación. Existiendo numerosa bibliografía en red con ejemplos para su creación.

Nota: El tipo String de Rapid se corresponde en C/C++ con un vector de datos tipo Char.