UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA MECÁNICA

TESIS

Diseño e implementación de un módulo robótico y visión artificial para la automatización del procedimiento de toracocentesis

Para obtener el Título Profesional de Ingeniero Mecatrónico.

Elaborado por

Diego Enrique Velásquez Vergara

0009-0001-4022-3394

Asesor

Dr. Ing. Ricardo Raúl Rodríguez Bustinza

0000-0002-6411-7123

LIMA – PERÚ

2025

Citar/How to cite	Velásquez [1]
Referencia/Reference Estilo/Style: IEEE (2020)	[1] D. Velásquez, "*Diseño e implementación de un módulo robótico y visión artificial para la automatización del procedimiento de toracocentesis*" [Tesis]. Lima (Perú): Universidad Nacional de Ingeniería, 2025.

Citar/How to cite	(Velásquez, 2025)
Referencia/Reference Estilo/Style: APA (7ma ed.)	Velásquez, D. (2025). *Diseño e implementación de un módulo robótico y visión artificial para la automatización del procedimiento de toracocentesis*. [Tesis, Universidad Nacional de Ingeniería]. Repositorio institucional Cybertesis UNI.

DEDICATORIA

Dedico esta tesis a mi mamá, Juana Vergara, por su constante amor, esfuerzo y apoyo guiándome siempre con sabiduría y enseñándome que la disciplina y la perseverancia son esenciales para alcanzar grandes metas.

A mi familia, especialmente a mis padrinos Deifilia Vergara y César Vergara, quienes siempre estuvieron pendientes de mí, inspirándome a ser mejor cada día. Hoy, desde el cielo, pueden estar tranquilos, ya que las metas trazadas se han ido cumpliendo.

A todos mis profesores y catedráticos por compartir su conocimiento, amistad y experiencias, que han fortalecido mi desarrollo personal, académico y profesional.

Por último, y no menos importante, a aquellos que se esfuerzan por aplicar su conocimiento y experiencia al servicio de las necesidades de nuestra sociedad.

AGRADECIMIENTO

En primer lugar, agradezco al Dr. Ing. Ricardo Rodríguez Bustinza, por su guía y asesoramiento para la culminación de esta tesis. Asimismo, destaco su constante soporte académico a lo largo de mi etapa universitaria, impulsando mi interés por el estudio y desarrollo de tópicos emergentes como la Inteligencia Artificial y Automatización aplicada.

Agradezco a mi alma máter por brindarme años de formación y experiencias significativas que han contribuido a mi desarrollo integral, permitiéndome hoy aportar con conocimientos al servicio de la sociedad.

A mis colegas, amigos y todos aquellos que siempre me alentaron a continuar y concluir este proyecto con dedicación y entusiasmo.

Extiendo, de manera especial, este agradecimiento a otra etapa académica como la maestría, ya que esta experiencia me permitió reinsertarme en el mundo de la investigación. En particular, agradezco a la Dra. Maria Koskinopoulou por su inspiración y soporte durante ese periodo.

RESUMEN

En la última década, los proyectos de investigación en el campo de la robótica aplicada a la industria de salud están en contante desarrollo especialmente en las áreas de diagnóstico, administración de medicina para pacientes y la asistencia en procedimientos quirúrgicos. Estos avances en robótica quirúrgica destacan por su precisión y la confiabilidad de su performance lo que ha permitido mejorar la calidad de servicio y seguridad de muchos procedimientos médicos. Adicional a ello, dichos proyectos han sido fundamentales para el entrenamiento de los doctores al gestionar procesos médicos de alta complejidad antes de interactuar en un entorno real con pacientes siendo vital durante el proceso operacional. Basado en este enfoque, la presente tesis busca aplicar los conocimientos adquiridos en la especialidad de ingeniería mecatrónica para abordar uno de los procedimientos médicos más complejos en el tratamiento de Infecciones Respiratorias Agudas (IRA) que se encuentran dentro de las principales causas de muertes en el nuestro territorio nacional. Particularmente, este enfoque está dirigido a la toracocentesis, el cual es un procedimiento mínimamente invasivo (MIS) encargado del drenaje de la acumulación del líquido pleural (derrame pleural) el cual será analizado desde una perspectiva técnica para diseñar y codificar esta intervención en el paciente. El objetivo de este proyecto es optimizar el procedimiento mencionado mediante la integración de tecnologías robóticas avanzadas, análisis de procesamiento digital de imágenes (PDI) y control electrónico de componentes. Esto tiene como finalidad desarrollar un producto para mejorar la eficiencia, precisión del procedimiento y reducción de la mortalidad asociada a esta enfermedad. De acuerdo a las pruebas experimentales, se obtuvieron 10 ciclos completados de manera exitosa lo que permite afirmar la eficiencia del sistema autónomo y un prometedor avance en la industria de la salud.

Palabras clave: Toracocentesis, derrame pleural, robótica quirúrgica, análisis de imágenes.

ABSTRACT

In the last decade, research projects in field of robotics applied to healthcare industry have been consistently developed, with a particular focus on diagnosis, drug administration and assisting surgical procedures. Achievements in surgical robotics have demonstrated their precision and reliability, significantly enhancing service quality during operations and improving the safety of several medical procedures. Furthermore, such projects have played a crucial role in training medical learners by managing highly complex health procedures in a controlled environment before interacting with real patients. This has proven vital during the training process, reducing risks and improving learning outcomes. Based on this approach, the present thesis aims to apply the knowledge acquired during the specialization in mechatronic engineering to address one of the most complex medical procedures related to acute respiratory infections (ARIs), which rank among the leading causes on mortality nationwide. Specifically, the focus is on thoracentesis which is a minimally invasive surgery (MIS) used to drain the accumulation of water around the lung, also known as pleural effusion. This thesis approaches thoracentesis from a technical perspective designing and coding all the steps involved in such procedure. The objective of this thesis is to optimize the thoracentesis performance through the integration of advanced robotic technologies, analysis of digital image processing and control electronic system. The goal is to develop a product that enhances procedural efficiency and precision while contributing to a reduction in the mortality rate associated with pleural effusion. According to the experimental tests, 10 cycles were successfully completed, demonstrating the efficiency of the autonomous system and representing a promising advancement in the healthcare industry.

Key words: **Thoracentesis, pleural effusion, surgical robotics, image processing.**

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCIÓN

La presente tesis propone un sistema autónomo para llevar a cabo la toracocentesis que es el tratamiento para aliviar la enfermedad de derrame pleural, el cuál es un exceso de líquido formado alrededor de los pulmones. Para ello, este sistema se soporta esencialmente en la aplicación de tópicos relacionados a robótica y visión artificial. El contenido se ha distribuido en 4 capítulos.

En el **CAPITULO I**, se aborda la problemática relacionada a esta enfermedad, antecedentes de investigación relacionados.

En el **CAPITULO II**, se presenta todos los conceptos necesarios desde el perfil clínico hasta el perfil de aplicación de ingeniería relacionado a robótica e inteligencia artificial (en adelante, IA), los cuáles serán fundamentales para entender el enfoque de esta tesis.

En el **CAPÍTULO III**, se detalla cómo se ha enfocado el proyecto y su proceso de materialización para lograr el objetivo de tener un sistema autónomo basado en precisión, eficiencia y seguridad.

En el **CAPÍTULO IV**, se expone los resultados en las pruebas unitarias y del sistema integrado, puesto que es importante destacar el paso a paso para lograr las metas establecidas.

Por último, se destacan las conclusiones del proyecto, así como también las recomendaciones para abordar un proceso clínico, así como también durante el proceso experimental de este proyecto para reducir riesgos y eliminar errores.

CAPITULO I

GENERALIDADES

1.1 Antecedentes de la Investigación

(Silvera-Tawil, 2024), destacó el estado del arte del impacto de la robótica en la industria de la medicina moderna con un enfoque exhaustivo en el potencial de estos en un escenario real. El objetivo de este estudio destaca la aplicación robótica en este sector primario el cual se encuentra en constante desarrollo ya que constantemente las innovaciones tecnológicas van aperturando mecanismos capaces de abordar los procedimientos clínicos de manera altamente eficiente y segura. Al inicio de este estudio, el autor presentó las aplicaciones de la robótica en medicina categorizadas en 5 grandes grupos como: robótica de servicio, de asistencia, de asistencia social, tele operacional y de intervención clínica. Este último grupo representa los avances tecnológicos, en su totalidad sistemas robóticos con cierto grado de autonomía orientados a los MIS tales como laparoscópica, intervención vascular o a la columna vertebral. Asimismo se destaca que debido a las limitaciones técnicas durante el proceso manual debido a la falta de precisión e inexperiencia, un sistema automatizado contribuye no solo a la asistencia durante el procedimiento quirúrgico sino también al entrenamiento de los médicos con la finalidad de entrenarlos en el manipuleo de los instrumentos médicos para controlar la trayectoria y en consecuencia gestionar un procedimiento seguro y de alta confiabilidad especialmente cuando la zona anatómica es de alto riesgo como ejemplo para la aplicación de biopsia en el cerebro.

Como se aprecia en la **Figura 1**, algunos robots comerciales ya participan en procedimientos complejos de intervención quirúrgica tales como “da vinci” para intervención de laparoscópica, luego “Mako” y “Navio” para abordar casos de ortopedia; sin embargo, aún existe un amplio margen de desarrollo en este tipo de sistemas para abordar otras aplicaciones quirúrgicas de alto grado de complejidad.

Figura 1

Robots comerciales en intervenciones quirúrgicas

Nota. Estos robots destacan por su intervención en procedimientos quirúrgicos mínimamente invasiva. Tomado de (Silvera-Tawil, 2024).

De acuerdo con (Iftikhar, 2024), su estudio puntualiza el avance de los sistemas robóticos quirúrgicos los cuales han sido beneficiados por el notable aporte de la IA ya que mejoran substancialmente la consistencia de las tareas automáticas y reducen la carga de trabajo operativo que conlleva a reducir los errores provocados por factor humano. La IA incorpora funcionalidades sobresalientes en los robots como por ejemplo reconocimiento de imágenes para detección de las regiones de interés, control de movimiento para optimizar el correcto manipuleo de los instrumentos médicos, retroalimentación háptica para simular sensaciones y resistencias de ciertos tejidos, principalmente esto último casos de tele operación permitiendo que el cirujano obtenga consiguiendo alta eficiencia y confiabilidad en los resultados de dichas tareas automáticas.

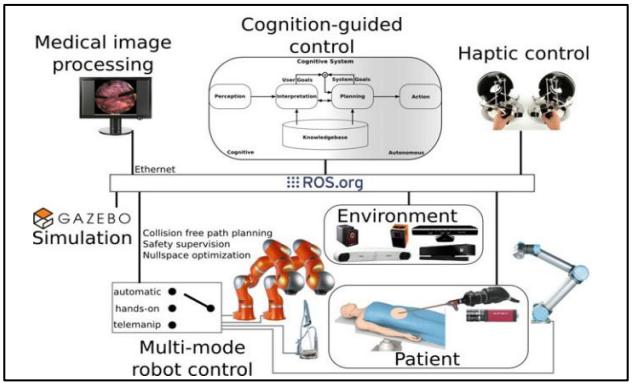
Adicionalmente, el estudio menciona dos técnicas que están influyendo fuertemente en la obtención de resultados destacables en la integración de la IA y la robótica los cuales son el Deep Learning (Aprendizaje profundo) y el Reinforcement Learning (Aprendizaje por refuerzo). La primera de estas técnicas basa su enfoque en el uso de imágenes para extraer las características más relevantes identificando estructuras anatómicas específicas de acuerdo con el procedimiento médico. En cuanto a la segunda técnica, su arquitectura es usada para acumular experiencias a través de pruebas y errores lo que permite incrementar el funcionamiento del algoritmo para tareas cada vez más

complejas. A medida que los algoritmos basados en IA se acondicionan para interactuar con los mecanismos que contienen los instrumentos quirúrgicos se determinan categorías dentro de los niveles de autonomía: bajo, medio y alto. Mientras que en las categorías bajo y medio se presenta intervención humana; en la categoría alta el sistema genera su propia retroalimentación lo que significa que el sistema es absolutamente autónomo haciéndolo independiente del factor humano.

Dado que este campo de la robótica aplicado a intervenciones quirúrgicas soportadas por algoritmos basados en IA se encuentra en permanente evolución, el estudio realizado por (Bihlmaier, 2016) destaca por su perspectiva de este campo afianzado en la configuración de ROS (Robot Operative System) donde se busca profundizar en el modo en que se combinan los componentes de percepción, el robot, los instrumentos médicos como se aprecia en la **Figura 2**, y como ROS gobierna de manera interna a través de comandos para que el sistema opere independientemente logrando desarrollar el perfil cognitivo del robot que le permitirá tomar decisiones en la marcha del procedimiento médico designado. Esta flexibilidad es vital en los sistemas autónomos para lidiar con agentes externos que provoquen alteración del performance del robot en la MIS.

Figura 2

Vista detallada de una intervención quirúrgica autónoma



Nota. Este esquema detalla la interacción de todos los componentes centralizado en entorno ROS y la aplicación en el paciente. Tomado de (Bihlmaier, 2016).

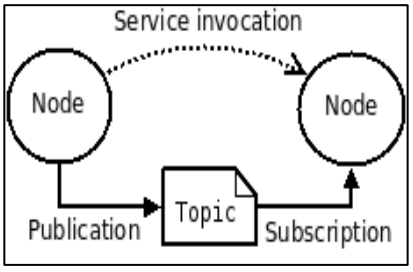
(Robotics, 2018). ROS jugará un rol importante en el proyecto tanto en el análisis, desarrollo de software e implementación dentro del entorno de simulación por lo que es necesario profundizar en los aspectos computacionales. ROS es una plataforma que

contiene una extensa variedad de librerías y herramientas las cuáles están diseñadas para crear aplicaciones para robótica, así como también herramientas de visualización a través de mensajes, servicios y/o acciones administradas por paquetes. Esta plataforma tiene la facilidad de ser implementada en lenguajes ampliamente usados tales como Python o C++ integrándose fácilmente de acuerdo con el sistema operativo (OS) de la estación principal de trabajo. ROS es compatible con el OS Linux; por lo que en casos de estaciones de trabajo con OS Windows se debe tratar de instalar una máquina virtual; sin embargo, estos no son plataformas suficientemente estables debido a la compatibilidad para desplegar apropiadamente el entorno de simulación del control de robots (Open Robotics, n.d.). Seguidamente a la instalación, se debe abordar cómo establecer el tipo de comunicación en ROS y conceptos básicos para entender el nivel de comunicación. Es decir, se describirá en detalle la arquitectura de esta plataforma como se visualiza en la **Figura 3**, que permite el intercambio de mensajes o servicios, entre ellos tenemos los siguientes términos por explorar.

- Mensaje: Qué es un mensaje y Tipo de mensajes.
- Tópico: Medio de transmisión del mensaje.
- Publisher / Subscriber: Elementos que publican y recepcionan el mensaje.
- Servicios y Acciones: Estructura para encapsular tipos de mensajes.
- Nodos: Unidad básica de la arquitectura que alberga los algoritmos de programación.

Figura 3

Arquitectura del sistema de comunicación en ROS



Nota. Representación de la estructura interna de comunicación en el marco ROS para publicar y subscribir mensajes los cuales transmiten la información de nodo a nodo. Tomado de ROS/Concepts (Open Robotics, n.d.)

Luego de conocer estos conceptos fundamentales y su funcionamiento dentro de la arquitectura, será posible una combinación de estos para lograr una unidad compacta que contenga todas las especificaciones de software bajos las siguientes pautas:

- Crear paquete principal de trabajo.
- Crear una carpeta “src” para almacenar archivos fuentes y una carpeta scripts para almacenar scripts en Python.
- Cada vez que se crea un script que contenga un algoritmo debe modificarse el perfil del script a ejecutable usando el comando “chmod”.
- Compilar el paquete principal con el comando “catkin_make” y habilitar el potencial de los archivos para ejecuciones ya sea en el entorno de simulación RVIZ o en las aplicaciones reales.
- Por último, “sourciar” con el comando “source devel/setup.bash” para reconocer y habilitar los comandos ROS dentro del paquete principal.

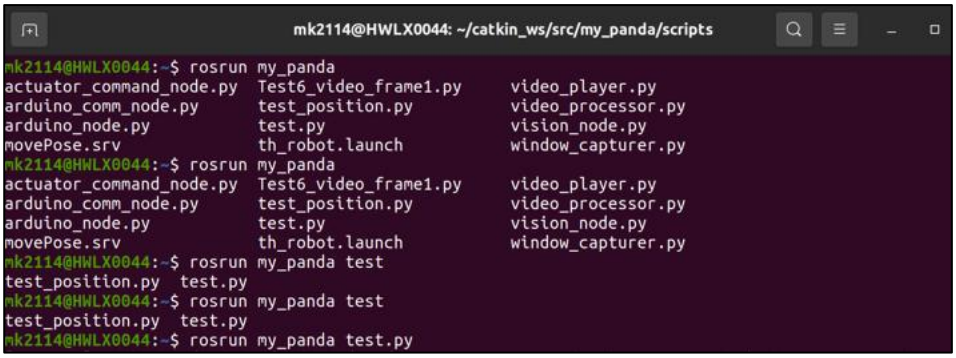
Esta secuencia debe ser ejecutada cada vez que se registra un cambio o modificación en los scripts o registros fuentes para que se ejecuten las tareas asignadas.

Asimismo, se detallará las principales librerías en ROS dentro del lenguaje de programación que contribuirán con la ejecución de estos scripts mencionados. Finalmente, luego que los archivos fuentes y ejecutables se encuentren listos, se gestionará el inicio de todo el software bajo los comandos rosrun o roslaunch como se visualiza en la **Figura 4**. Estos son similares ya que lanzan la ejecución de los scripts; sin embargo, tienen características que admiten lanzar un archivo ejecutable o un grupo de ellos a la vez demostrando un perfil colaborativo e integrador dentro de la plataforma ROS.

Para complementar la información aplicativa de ciertos comandos ROS como se visualiza en la **Figura 4**, que solo contribuyen al proceso operacional de ejecución y aplicación de ciertos comandos ros tales como:

Figura 4

Comandos ROS en Ubuntu



Nota. Entorno Ubuntu ejecutando comandos ROS para manipular, crear, modificar y visualizar archivos dentro de los paquetes de trabajo y registros fuente. *Fuente:* Elaboración propia.

- catkin_create_pkg: para la creación de paquetes,
- rostopic list, rosnode list: para mostrar la lista de tópicos o nodos en ejecución.
- rosmsg show: para mostrar el tipo de mensaje y la variable de almacenamiento.
- roscd: para búsqueda de paquetes.
- ls: para visualización de archivos dentro del paquete o folder actual.
- code.: para abrir el entorno de codificación. Particularmente se trabajará en el presente proyecto con la multiplataforma Visual Studio.

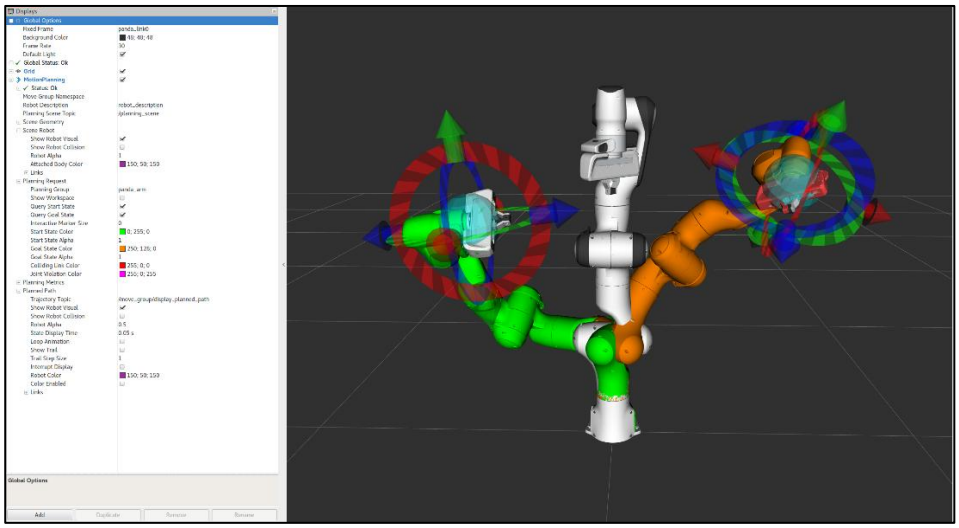
Cabe indicar que estos comandos necesitan atributos adicionales para asignar la tarea a un paquete, nodo, tópico o archivo ejecutable definido; de esta manera se podrá distinguir a que elemento de ros estamos apuntando las operaciones internas. De acuerdo con la **Figura 4**, el interfaz a usar será el software Ubuntu 20.04 y la versión de Noetic para ROS.

(Rviz, n.d.) Seguidamente, se describirá brevemente la cinemática del robot, en particular el modelado de la cinemática inversa ya que en una aplicación real para lograr

trasladar el efector final desde una posición inicial del robot (color verde) y una posición objetivo (color naranja) como se observa en la **Figura 5** según, se requiere variar los parámetros que controlan las articulaciones del robot a partir de la información proporcionada de las posiciones mencionadas. Este modelado y representación se llevará a cabo en RVIZ el cual es una herramienta de ROS que permite visualizar en 3D el movimiento y parámetros que permite controlar el sistema robótico en tiempo real. Sin embargo, hay muchas otras particularidades que serán abordadas cuando se despliegue el marco de trabajo para el control de trayectoria del robot ya que también para otorgar precisión en el movimiento se requiere un planeamiento de trayectoria cartesiana del efector final. En este sentido, la lógica será soportada por algoritmos en lenguaje de programación C++ o Python, los cuales estarán a cargo de gestionar la trayectoria del robot.

Figura 5

Control de robot Franka Emika Panda en ROS.



Nota. Para el control de movimiento del robot Franka Emika Panda se usa la plataforma basada en algoritmos programados en C++ o Python. *Fuente:* Tomada de (Rviz, n.d.))

Esta planificación del control cartesiano habilita al robot a que el efector final siempre se mantenga en una posición invariante de rotación en cualquier momento de su traslación. Como se observa en la **Figura 5**, la posición del efector final la pose inicial y la pose final difieren en su orientación de rotación con respecto a la referencia principal que

se encuentra en la base; lo que significa que el planeador no ha sido configurado en secuencia cartesiana. Esta condición es muy importante ya que, para el presente proyecto será vital que el módulo robótico adjunto al efector final no tenga variación con respecto a sus ejes rotacionales.

Luego de ahondar en el análisis mecánico, software de control y control de trayectoria para el robot, se analizará las siguientes técnicas no invasivas para obtener imágenes médicas.

Dentro de las distintas modalidades para obtener imágenes médicas tenemos el uso de los rayos x (RX), la tomografía computarizada (TC), resonancia magnética (RM) y por ultrasonido (US) que auscultan al paciente para desempeñar tareas de clasificación y segmentación al analizar patrones de estudio clínico. Estos patrones permiten generar diagnósticos y tomar decisiones acertadas según el tipo de intervención se tenga. Sin embargo, teniendo tantos modos surgen las siguientes interrogantes: ¿Cuál es la modalidad más apropiada para diagnosticar el derrame pleural en los pacientes? Para resolver ello es necesario evaluar las ventajas de cada una de las técnicas.

Tabla 1

Análisis de sensibilidad y especificidad para técnica de US y RX

TÉCNICA	CASOS ANALIZADOS	SENSIBILIDAD	ESPECIFICIDAD
US	12	94.54%	97.88%
RX	15	67.68%	85.30%

Nota. Estos valores resumen el análisis estadístico de los 12 casos con derrame pleural.

(Zaki, 2024), ha elaborado una exhaustiva revisión acerca de la eficiencia de las técnicas empleadas como por ejemplo de los RX versus el ultrasonido basado en criterios de sensibilidad y especificidad. Para determinar los valores de la **Tabla 1** se analizaron 12 casos confirmados bajo el análisis con US y 15 casos basado en RX.

Según (Yang, 2024) , manifiesta la efectividad de la técnica US con respecto de TC el cual también se basa en los criterios descritos anteriormente. Para la **Tabla 2** fueron considerados 285 pacientes con derrame pleural.

Tabla 2

Análisis de sensibilidad y especificidad para técnica de US y TC.

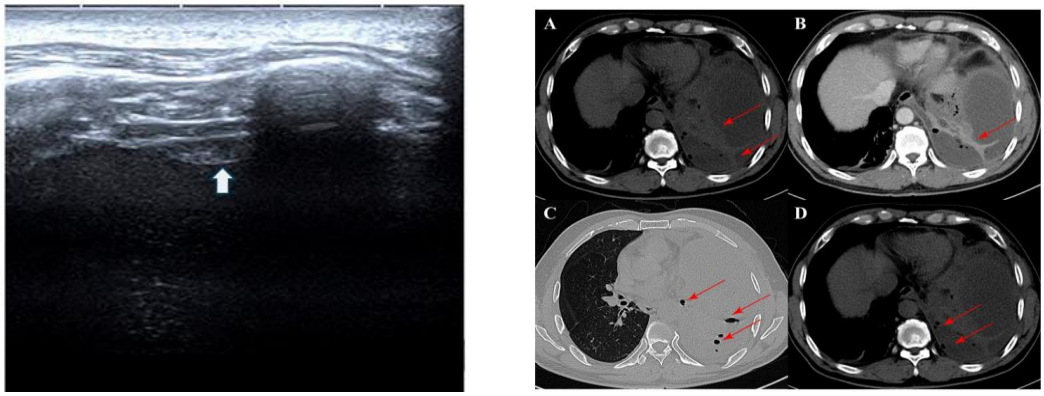
TÉCNICA	SENSIBILIDAD	ESPECIFICIDAD
US	82.6%	100%
TC	59.8%	87%

Nota. Valores resumen el análisis estadístico de los 15 casos con derrame pleural.

Estos criterios de sensibilidad y especificidad están relacionados con el reconocimiento de casos verdaderamente positivos y de falsos negativos lo que permite validar una correcta identificación de los casos que corresponden a un derrame pleural. Por tanto, basado en los datos estadísticos de estas dos tablas, se resalta la alta efectividad del US por encima de las demás técnicas.

Figura 6

Imagen de ultrasonido (izq) y tomografía computarizada (der)



Nota. Estas imágenes representan una descripción gráfica del derrame pleural en un formato de niveles de grises. Fuente: Tomada de (Yang, 2024).

Esta información es realmente valiosa al presentarse en el formato de mapa de grises como se aprecia en la **Figura 6** ya que con técnicas de procesamiento digital de

estas imágenes se podrá segmentar las regiones de interés y hasta incluso generar una plantilla para que evalúe cualquier otra imagen médica y brinde soporte automático en el diagnóstico de derrame pleural. Indudablemente, una gran ventaja para elaborar filtros que destaquen las regiones de interés con respecto de áreas que no contienen información relevante.

Aunque ninguna de las técnicas mencionadas para la examinación del paciente requiere algún corte o inserción de instrumento médico, si hay algunos factores que también deben tenerse en cuenta como por ejemplo la exposición a la radiación el cual no lo tiene la técnica de US. En cuanto al aspecto de manipulación es más accesible el US en lugar de CT y RX ya que estos últimos implican que sus máquinas correspondientes estén asignadas a un solo espacio debido al volumen de las máquinas; mientras que el US es perfectamente moldeable a cualquier posición del paciente y accesible.

Por último, con respecto al costo de adquisición, una probeta de ultrasonido es más viable de conseguir; sobre todo para lugares de escasos recursos donde es esencial tener equipos médicos necesarios y efectivos para la atención a la comunidad.

Por lo tanto, el procesamiento de imágenes será dirigido para US-imágenes ya que, debido a su alta precisión, tendremos imágenes de calidad; así como también es accesible y manipulable para cualquier caso médico.

1.2 Identificación y Descripción del Problema

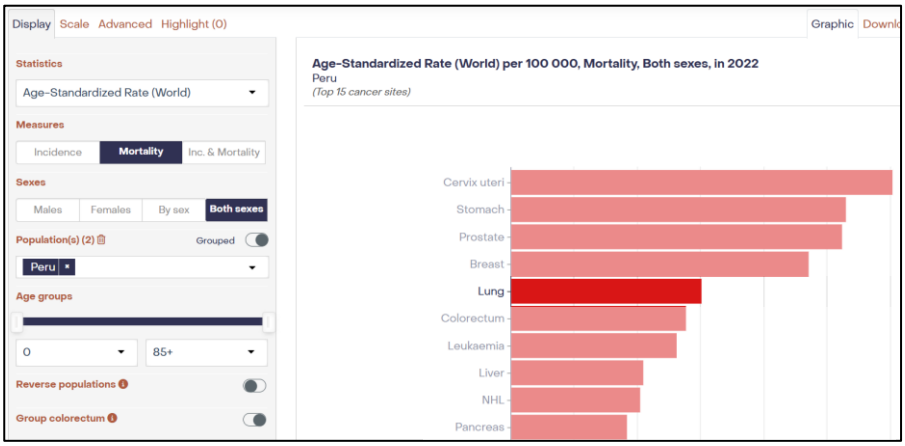
Según nota de prensa publicada por el MINSA en noviembre 2024, destaca que el cáncer de pulmón es la principal causa de mortalidad en el mundo (Gobierno del Perú, 2024). Según la Organización Mundial de la Salud (OMS), esta enfermedad representa la quinta causa de decesos con una incidencia del 6.8 por cada 100,000 personas (World Health Organization, 2022) cómo se visualiza en la **Figura 7**. Dada estas cifras alarmantes, el Instituto Peruano de Economía (IPN) profundizó en el análisis las incidencias de esta enfermedad y detectó un aumento del 26% en la mortalidad provocada por dicha enfermedad en los últimos 5 años siendo la variación de la ratio de mortalidad 14.5 a 18.3

muertos por cada 100,000 personas alrededor del mundo (Instituto Peruano de Economía, 2024).

El Instituto Nacional de Enfermedades Neoplásicas (INEN) afirma que cada año se reciben nuevos pacientes en un rango de 500 a 600; sin embargo, lo más alarmante de esta situación es que el 80% de ellos se encuentra en una etapa avanzada de la enfermedad, lo que causa una alta probabilidad de deceso en corto plazo.

Figura 7

Causas de mortalidad



Nota. Gráfica de Información de las causas de mortalidad en Perú. Fuente: Tomada de la Agencia Internacional para investigación de cáncer.

Ante esta situación, el Perú debe enfrentar desafíos para mejorar la eficiencia de diagnóstico dentro del sector salud tales como la gestión pública, entrenamiento de especialistas, equipamiento tecnológico, entre otros. Estos factores están altamente asociados al índice de mortalidad y se agudizan más en los sectores rurales de nuestro país donde el acceso a personal médico y equipos avanzados muchas veces están condicionados a la disponibilidad y/o destreza del médico.

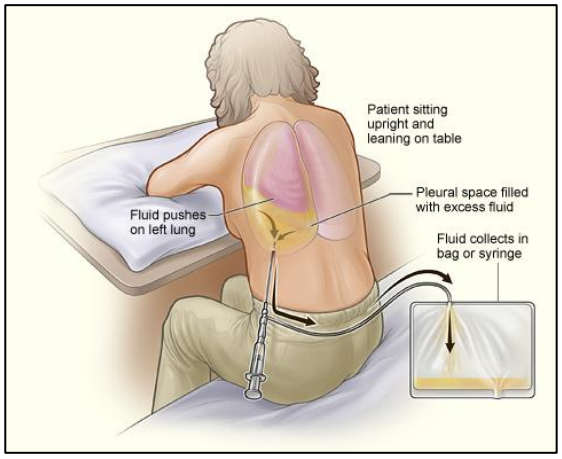
Uno de los procedimientos mínimamente invasivo (MI) como la toracocentesis, el cual es un método fundamental que contribuirá al diagnóstico y tratamiento del cáncer de pulmón, consiste en la extracción del exceso del líquido pleural a través de un catéter que es insertado hasta la zona pleural como se describe en la **Figura 8**. Este derrame es una

de las causas no solo de cáncer de pulmón sino también de tuberculosis y otras infecciones respiratorias.

Actualmente en nuestro país, dicho procedimiento es llevado a cabo manualmente; es decir, el especialista realiza una inspección ya sea asistida por rayos x, resonancia magnética o a través de un escaneo de ultrasonido aplicado en la zona torácica, para luego maniobrar la inserción del catéter hasta la cavidad pleural con la finalidad de aspirar dicha acumulación; sin embargo, este procedimiento podría generar algunos contratiempos técnicos ya que la intervención manual conlleva a ciertas limitaciones o podría provocar complicaciones futuras respecto del estado del paciente.

Figura 8

Procedimiento de Toracocentesis



Nota. El procedimiento describe la inserción del catéter hasta la zona pleural (derrame) y luego se extrae la acumulación del líquido alrededor del pulmón (amarillo). *Fuente:* Med Care Tips (Health & Medical Care)

Entre ellas, se encuentra la fatiga ocular para detectar la zona de interés en la imagen médica o determinación de la posición de órganos, tejidos o nervios; por otro lado, también influye la falta de entrenamiento entre los intervencionistas no teniendo la destreza adecuada para proceder con esta operación quirúrgica. Por tanto, estos factores comprometen notoriamente el procedimiento médico al carecer de una alta efectividad y de la adecuada protección para la condición médica de los pacientes.

1.3 Formulación del Problema

De la problemática, se puede evidenciar que para llevar a cabo el proceso de la toracocentesis con alta eficiencia existen limitaciones principalmente las relacionadas a las tecnológicas por lo que surgen algunas preguntas clave que serán distribuidas en problema principal y específicos.

1.3.1 Problema Principal

- ¿Es posible automatizar del procedimiento quirúrgico toracocentesis mediante un manipulador robótico guiado por visión artificial?

1.3.2 Problemas Específicos

- ¿La implementación de un modelo robótico guiado por visión artificial contribuye a la eficiencia del sistema para abordar el proceso de toracocentesis?
- ¿Es posible que el uso de este sistema en el proceso de toracocentesis reduzca el error durante la intervención clínica, de tal manera que sea un procedimiento preciso y seguro?

1.4 Justificación e Importancia

Se debe tener en cuenta algunos aspectos y criterios que fundamenten y destaquen el potencial de esta tesis, para lo cual se describirán los siguientes argumentos:

- ***Argumento Técnico:*** Esta tesis apunta a un enfoque integrado que combina tecnologías relativamente emergentes en nuestro país para abordar problemas críticos relacionados con patologías pulmonares a través un procedimiento quirúrgico esencial para su diagnóstico oportuno. El enfoque está dirigido al procesamiento digital de imágenes y control basado en ROS que al combinarlos generarán un notable impacto en el ámbito de la salud y de la ingeniería en Perú.
- ***Argumento Social:*** El acceso a los servicios médicos especializados es un desafío que afecta a la población peruana. La implementación una tecnología

accesible y segura, como un sistema automatizado tiene el potencial de reducir la dependencia de especialistas en ubicaciones lejanas como las zonas rurales. En consecuencia, se estará promoviendo el acceso a un servicio médico de calidad, así como también la mejora de diagnósticos y tratamientos oportunos los cuáles fortalecerán los indicadores de salud pública.

- ***Argumento Operativo:*** Desde el punto de vista médico, la precisión es un factor altamente valorado cuando se trata de intervenciones complejas. En ese sentido, la aplicación de procesamiento digital de imágenes permitirá reducir los riesgos asociados a la intervención como sangrado, infecciones o neumotórax. Por otro lado, este proyecto no solo será de gran utilidad para los diagnósticos tempranos, sino también que podrá adaptarse a entornos de simulación anatómicos para el entrenamiento de los intervencionistas lo que les permitirá desarrollar la destreza necesaria para su aplicación. Por último, el conocimiento de ingeniería en cuanto a la implementación es transferible a otros tipos de intervenciones complejas como diagnósticos en el sistema cardiovascular, biopsias, médula espinal, etc.

1.5 Objetivos del Estudio

1.5.1 Objetivo General

- Automatizar el procedimiento de la toracocentesis mediante el empleo de un manipulador robótico guiado por visión artificial.

1.5.2 Objetivos Específicos

- Determinar si la integración de un sistema basado en un manipulador robótico y visión artificial contribuye a la eficiencia del procedimiento de toracocentesis.

- Determinar si el desempeño del sistema autónomo propuesto contribuye a la reducción del margen de error clínico durante la ejecución de la intervención clínica.

1.6 Hipótesis

1.6.1 Hipótesis General

- El desarrollo de un sistema que integre un manipulador robótico guiado por visión artificial automatizará el procedimiento de toracocentesis optimizando la eficiencia de la intervención quirúrgica.

1.6.2 Hipótesis Específicas

- El desarrollo del sistema autónomo propuesto contribuirá a maximizar la eficiencia del procedimiento clínico.

- El desarrollo del sistema autónomo propuesto generará una reducción del margen de error clínico aplicado al procedimiento clínico.

1.7 Variables y Operacionalización de Variables

A continuación, se presenta la operacionalización de variables, las cuáles definen de manera precisa las variables involucradas, así como también los métodos a emplearse para determinar su medición dentro del entorno experimental.

VARIABLES	DEFINICIÓN CONCEPTUAL	DEFINICIÓN OPERACIONAL	DIMENSIONES	INDICADORES	ESCALA DE MEDICIÓN
VI: Sistema robótico guiado por visión artificial	Sistema autónomo para replicar el procedimiento quirúrgico de toracocentesis	Sistema autónomo controlado por ROS, visión artificial para detectar estructuras anatómicas que permitan identificar zona de punción	Control robótico Procesamiento de imágenes	Precisión del movimiento Distinción de estructura anatómica	Ordinal Nominal
VD: Eficiencia del sistema autónomo durante el procedimiento	Determinación del proceso para completar un ciclo de trabajo	Se refiere al tiempo requerido para la ejecución del procedimiento clínico y la evaluación de rendimiento durante tres ejecuciones sobre el simulador anatómico	Tiempo de ejecución. Repetibilidad	Duración de la ejecución Ciclos completados	Razón Razón
VD: Reducción del error clínico	Disminución de fallos durante el procedimiento	Se mide determinando la cantidad de fallos durante la ejecución del sistema dentro del entorno simulado	Precisión de zona de punción Seguridad en la intervención	Exactitud de la inserción en la zona objetivo Volumen aspirado	Nominal Razón

1.8 Metodología de la Investigación

1.8.1 Unidad de Análisis

La unidad de análisis consistirá en una maqueta anatómica que representará parte del torso principalmente la sección que contiene el pulmón, diafragma, el conjunto de costillas y la cobertura final que representará el tejido epitelial, siendo este último fundamental para que resista las punciones realizadas por la aguja.

1.8.2 Tipo, Enfoque y Nivel de Investigación

El perfil de este proyecto se ajusta a la aplicación netamente de los conceptos y técnicas de ingeniería para resolver un problema existente en el sector salud. Este proyecto apunta a ser netamente experimental validado en entornos de simulación proveyendo alta sensibilidad entre los conocimientos técnicos y la aplicación real de esta.

El enforque del proyecto es determinístico mixto puesto será configurado parámetros que definen condiciones iniciales y a partir de ellos, se iniciará una secuencia definida con la finalidad de validar si el proyecto logra operar de manera autónoma para lograr el procedimiento de la toracocentesis; asimismo, se estimará la cantidad del líquido pleural al extraerlo y se regulará de independiente para repetir la cantidad necesaria de procesos de aspirado.

El nivel o alcance del presente proyecto apunta a ser de carácter exploratorio y descriptivo. En cuanto al primero, el contexto y entorno de aplicación implica que se debe hacer una exploración para profundizar las variables independientes (inputs) para la correcta codificación de cada subetapa. Este punto de vista pretende transformar los inputs para que los resultados intermedios sigan procesándose. Posterior a ello, el proyecto cambiará un perfil descriptivo ya que los resultados permitirán describir si el proceso fue realizado eficazmente.

1.8.3 Diseño de la Investigación

El diseño de este proyecto es de carácter experimental ya que las variables independientes son transformadas para que los nuevos resultados se integren entre sí hasta lograr los objetivos establecidos. Este tipo de investigación contribuye a gestionar un análisis detallado de cada subetapa de la toracocentesis, lo que significa que deberá ser abordado aspectos de diseño, procesamiento digital de imágenes y control, de los equipos electrónicos bajo un framework capaz de gestionar comunicación interna entre los distintos equipos. Este espacio de trabajo es llamado ROS y será este quien controle todas las interacciones a nivel de software, logrando resultados tangibles que determinen la efectividad y cumplimiento de cada etapa del proceso.

1.8.4 Fuentes de Información.

La información obtenida ha sido variada dependiendo del tipo de conocimiento, a continuación, se brinda de manera organizada las fuentes según tipo de conocimiento.

- Médico: NHS, Páginas web.
- Científico: Google Scholar, IEEE, Scopus, Elsevier.
- Estadístico e informativo: INEN, IPN, MINSA, OMS.
- Técnico: Tópicos relacionados a robótica e IA, de libros detallados en la sección de referencias.

1.8.5 Técnicas e Instrumentos para procesar la información.

Las técnicas e instrumentos se organizan de la siguiente manera.

- Control del Robot: ROS, algoritmos en Python.
- Procesamiento de imágenes: Algoritmos en Python y técnicas SIFT.
- Control Electrónico: ROS, Arduino UNO.
- Instrumentos médicos: Jeringas, Agujas, Probeta de Ultrasonido.
- Módulo robótico personalizado: Impresión en 3D.

En el siguiente capítulo se profundizará acerca de la teoría correspondiente a la robótica y su análisis cinemático basado en la teoría de Denavit-Hartenberg, asimismo se introducirá conceptos que permitan comprender el procesamiento digital de las imágenes capturadas por el escáner de ultrasonido el mostrará la estructura interna del tórax para identificar los órganos, nervios, tejidos y vertebras.

CAPITULO II

MARCO TEÓRICO Y MARCO CONCEPTUAL

Este capítulo brindará una amplia descripción los conceptos clínicos asociados al procedimiento médico para el tratamiento de la efusión pleural. Estos conceptos son básicos para comprender el procedimiento quirúrgico de la toracocentesis y su integración con los componentes mecánicos y electrónicos propuestos para esta tesis. El objetivo es implementar un sistema compacto, capaz de lograr replicar la toracocentesis de manera autónoma. Por ello, en los siguientes subcapítulos se abordará la teoría asociada relacionado a los aspectos médicos como los fundamentos de ingeniería en el dominio de campos como la robótica, control y visión artificial.

2.1 Procedimiento clínico

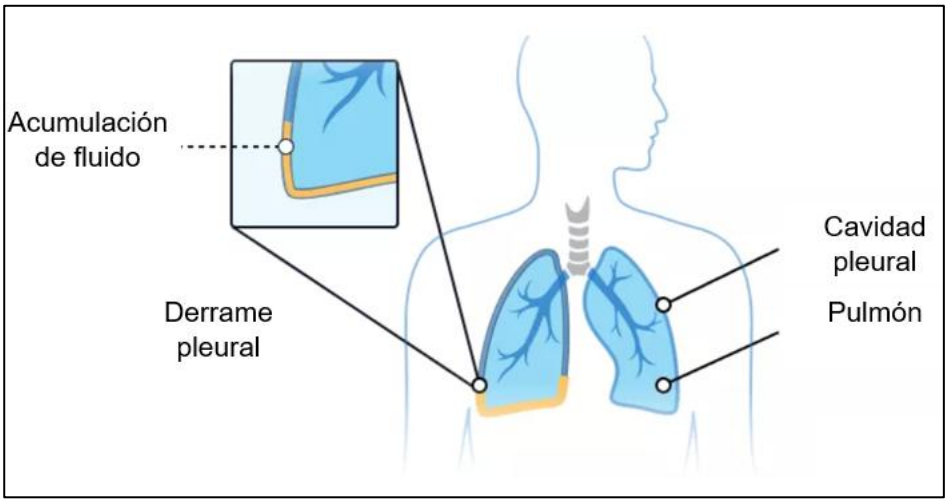
En esta subsección se presentará los conceptos clínicos que involucran a la condición médica del derrame pleural, así como del tratamiento que será la toracocentesis. Siendo esta última, la intervención quirúrgica que se replicará de manera autónoma en el desarrollo de esta tesis.

2.1.1 Efusión Pleural

La efusión pleural, conocida como derrame pleural o también llamada como agua en el pecho o agua en los pulmones, consiste en la acumulación excesiva de líquido en la zona pleural, la cual está constituida por 2 membranas delgadas ubicadas entre la cavidad torácica y el pulmón. La función de dichas membranas es cubrir los pulmones durante el proceso de inhalación de aire durante la respiración. Esto evita que los pulmones tengan contacto directo con las costillas, a su vez existe un mínimo de líquido entre las dos membranas cumpliendo la función de lubricación de estas delgadas capas (Cleveland Clinic, 2023; American Accreditation HealthCare Commission, 2022).

Figura 9

Efusión Pleural



Nota. Adaptado de (Selby, K., 2025)

Tal como se aprecia en la **Figura 9**, el saco alrededor del pulmón (pleura) contiene fluido (líquido) el cual está cubierto por las dos membranas del saco. La membrana interior, que cubre el pulmón se llama pleura visceral, mientras que la membrana del lado exterior es la pleura parietal que cubre las costillas, el diafragma y la región delimitada por el cuello.

2.1.2 Tratamiento médico

El tratamiento médico consiste en la extracción de la acumulación del líquido de la zona en cuestión. Para aliviar la condición médica de efusión pleural existen varios tratamientos médicos que dependerán de las condiciones de enfermedad del paciente. Según (Zhao, y otros), destaca en su trabajo de investigación los siguientes tratamientos:

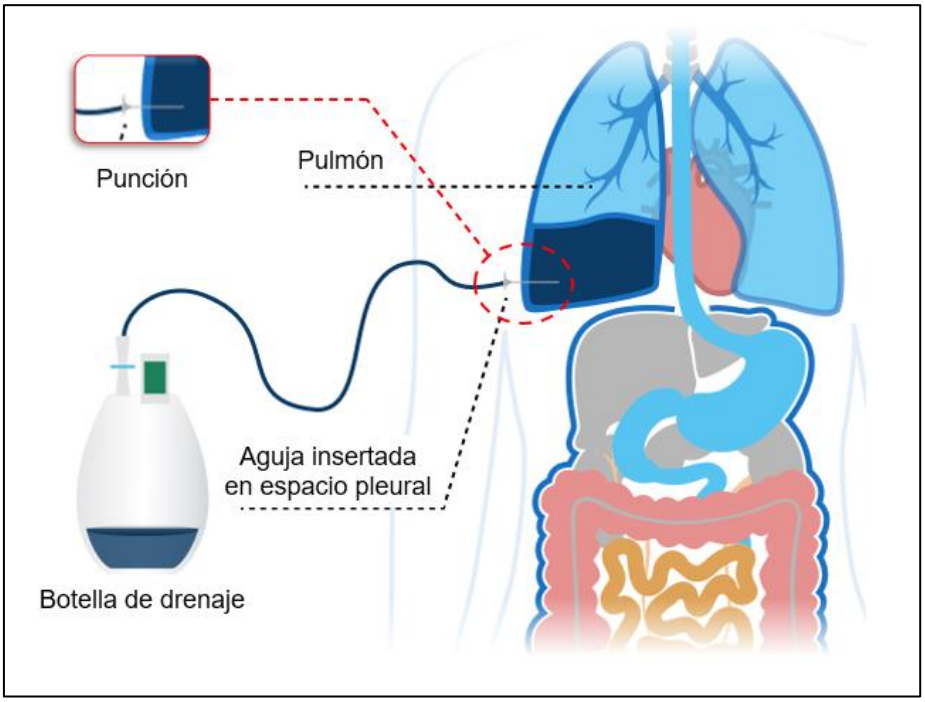
- **Toracoscopia médica o quirúrgica**, Insertar una cámara para visualización directa, además que se requiere también realizar punciones percutáneas, de acuerdo con la información brindada por (American Cancer Society, n.d.)
- **Drenaje Pleural**, tubo torácico insertado hasta la zona pleural de manera prolongada por días o semana, logrando controlar casos crónicos.

- **Toracocentesis**, aguja o catéter insertado hasta la zona pleural de manera temporal para extraer el líquido pleural.

De estos tratamientos mencionados, se elige la toracocentesis ya que es un procedimiento mínimamente invasivo tanto por la cantidad de punciones (solo una para el catéter) así como por el tiempo de tratamiento. La aplicación de este tratamiento se visualiza en la **Figura 10** en la que se percibe una aguja insertada hasta el espacio pleural la cual está conectada a una botella de drenaje mediante un tubo polímero biomédico. Esta conexión garantiza las condiciones de esterilidad requeridas para minimizar las opciones de agentes externos.

Figura 10

Tratamiento médico de la toracocentesis

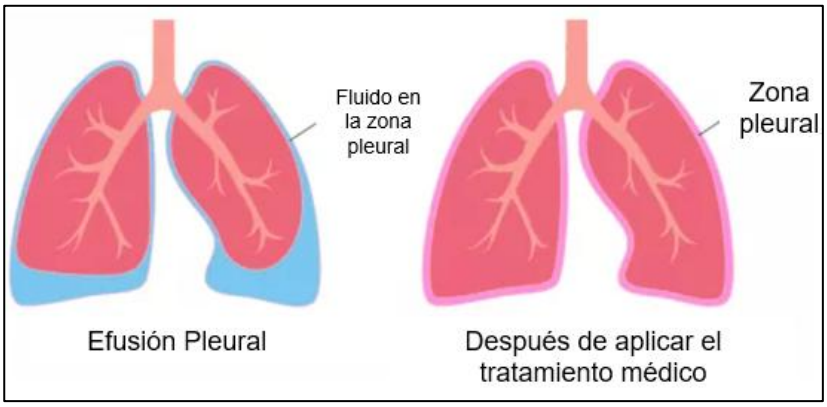


Nota. Adaptado de (Selby, K., 2025)

Luego de haber aplicado el tratamiento para aliviar la condición médica de derrame pleural, el sistema pulmonar debería recuperar su condición de normalidad según se observa en la **Figura 11**.

Figura 11

Comparación de pulmones antes y después de aplicar el tratamiento



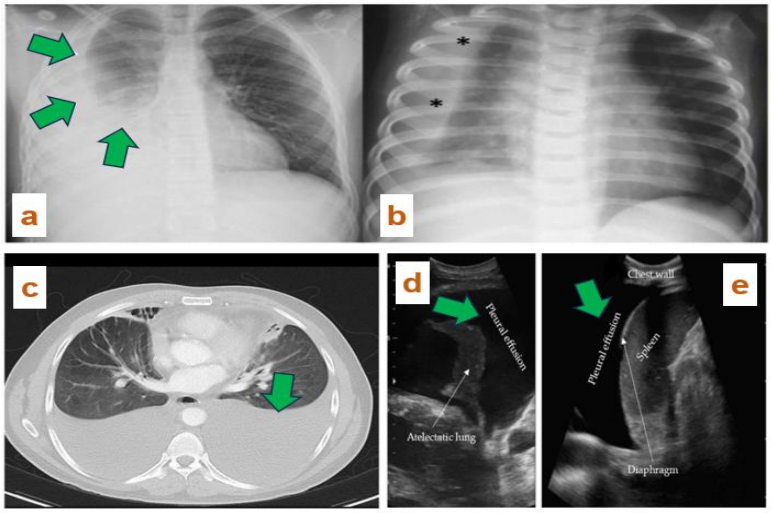
Nota. Adaptado de (Selby, K, 2025)

2.1.3 Métodos de Diagnóstico

El diagnóstico de la efusión pleural es de vital importancia, por lo que el paciente debe ser examinado en la zona superior torácica con el soporte de instrumentos médicos avanzados tales como radiografías (RX), tomografía computarizada (CT), resonancia magnética (MRI) o ecografía. (Gonnelli, y otros, 2024). La representación gráfica de estos diagnósticos se presenta en la **Figura 12**.

Figura 12

Modalidad de imágenes para diagnóstico



Nota. Resultado visual de diagnóstico según modalidades de imagen. Rayos X (a, b), Tomografía computarizada (c), Ultrasonido (d, e). Adaptación de (Nicholson, Manley, & Ahmad, 2023; Alexopoulou, y otros, 2024)

Todos los gráficos de la **Figura 12**, corresponden a casos de efusión pleural; sin embargo, se describe cómo es que se percibe la efusión pleural (señalado por flechas verdes) según método de diagnóstico:

- Diagnóstico por RX: Se observa manchas blanquecinas alrededor del pulmón.
- Diagnóstico por CT: Visualmente se observa una mancha gris dentro de la imagen presentada.
- Diagnóstico por US: Mancha negra en el interior de la cavidad torácica.

Todos estos métodos de diagnóstico a través de imágenes proporcionan información de la ubicación del derrame pleural. Adicionalmente, para determinar cuál sería el instrumento médico más adecuado para este proyecto se presenta la **Tabla 3**. Esta tabla muestra una comparativa entre ciertos aspectos de cada tecnología usada para el diagnóstico basado en las categorías de: Sin radiación ionizante, Resolución, Accesibilidad y Bajo Costo.

Tabla 3

Comparación de Ventajas y Desventajas de métodos de diagnóstico

TIPO DE TECNOLOGÍA	SIN RADIACIÓN IONIZANTE	RESOLUCIÓN	ACCESIBILIDAD	BAJO COSTO	TOTAL
Radiografía	NO	NO	**OK**	NO	1
Tomografía	NO	**OK**	NO	NO	1
Resonancia Magnética	**OK**	**OK**	NO	NO	2
Ecografía	**OK**	**OK**	**OK**	**OK**	***4***

Nota. Elaboración propia

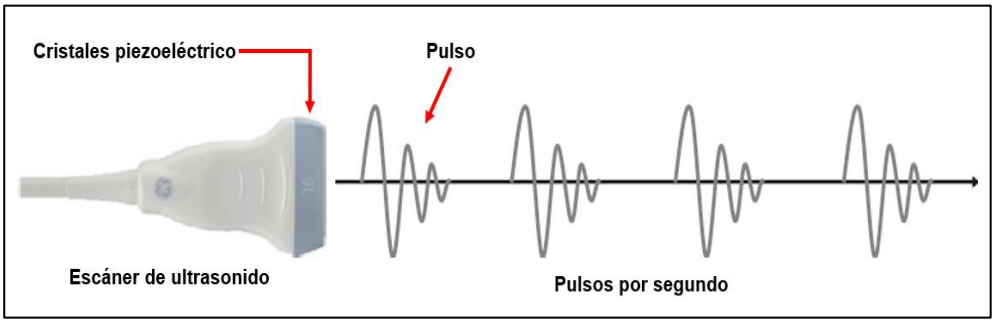
Como resultado del análisis, se destaca que las lecturas de ultrasonido (ecografía) es la tecnología que satisfice las categorías mencionadas; además que es de fácil manipulación como para integrarlo en un sistema autónomo.

2.1.4 Funcionamiento del escáner de ultrasonido

El escáner de ultrasonido es un instrumento médico fundamental en procedimientos clínicos de diagnóstico y de tratamiento, ya que provee imágenes médicas en tiempo real detallando la estructura anatómica interna del área en análisis (Sehmbi & Perlas, 2022). El proceso de obtención de imágenes de ultrasonido consiste en establecer contacto superficial de dicha herramienta con la primera capa de piel mas no involucra inserciones; por ello se considera como un procedimiento no invasivo. En la **Figura 13**, se observa que esta herramienta médica contiene cristales piezoeléctricos en el transductor de ultrasonido, los cuales, al ser comprimidos, producen diferencia de potencial eléctrico. Dicha condición de convertir energía mecánica (ondas) en energía eléctrica, es conocida como efecto piezoeléctrico. La energía eléctrica produce deformaciones en los cristales causando que la energía mecánica se convierta en vibración y emisión de ondas sonoras.

Figura 13

Composición y Funcionamiento de escáner de ultrasonido



Nota. Adaptado de (Sehmbi & Perlas, 2022)

Asimismo, este transductor es responsable tanto de la emisión de ondas sonoras como la de ser el receptor de las ondas reflejadas desde la capa de piel. Se emiten cierta cantidad de repeticiones de pulso por cada segundo. En este caso, según la **Figura 13**, la cantidad de pulsos sería 4 por segundo; y estas son emitidas en el rango de 1 a 10 kHz. Por último, un factor esencial en la representación de la imagen de ultrasonido es la impedancia acústica, que varía su grado de resistencia dependiendo del tipo de tejido. Así como se observa en la **Tabla 4**.

Tabla 4

Impedancia Acústica de algunas zonas del cuerpo humano

MATERIA	IMPEDANCIA ACÚSTICA (MRAYL)
Aire	0.0004
Agua	1.48
Grasa	1.34 – 1.38
Hígado	1.65
Sangre	1.61 – 1.65
Músculo	1.62 – 1,71
Hueso	6.0 – 7.8

Nota. Tomado de (Tran & Gulati, 2022)

Claramente, estos valores están alineados con la representación del nivel de gris. Como se puede visualizar, el agua tiene un bajo valor de impedancia por lo que corresponde el de color negro, mientras que un hueso tiene mayor impedancia lo que significa que será representado por el (color blanco); y así sucesivamente para los otros valores que se encuentran dentro del rango se representen con una sección de escala de grises (Tran & Gulati, 2022). El siguiente paso es identificar la mejor posición relativa entre la aguja y el escáner de ultrasonido.

2.1.5 Interacción del escáner de ultrasonido con los tejidos

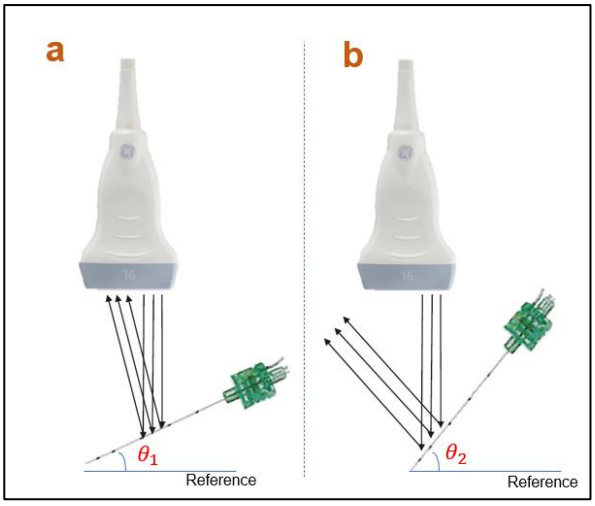
Según (Sehmbi & Perlas, 2022), el escáner de ultrasonido emite ondas que se propagan a través de los tejidos. Al posicionar un escáner sobre una específica del cuerpo humano, este envía pulsos de onda que inciden sobre las capas de tejido. Esta interacción, parte de la energía de la onda es reflejada hacia al transductor, generando una imagen que grafique la estructura anatómica en análisis.

En la **Figura 14** se observa la reflexión de la onda emitida por el escáner de ultrasonido el cual se ubica perpendicular a la línea de referencia. Asimismo, la imagen representa dos posiciones diferentes de inclinación de la aguja respecto del eje de referencia. La onda reflejada está definida por la orientación del eje de la aguja con respecto a una referencia alineada al eje x. Esta inclinación es medida mediante el

parámetro θ lo cual determina la proporción que retorna al lector del escáner de la onda incidente.

Figura 14

Reflexión de la aguja insertada para las ondas de ultrasonido



Nota. Efecto de reflexión de ondas según el ángulo de inclinación de la posición de la aguja. Adaptado de (Sehmbi & Perlas, 2022)

De la **Figura 14**, en la imagen izquierda el eje de la aguja determina un ángulo θ_1 respecto de la referencia, logrando que algunos rayos incidentes retornen al lector del escáner, mientras que, en la imagen derecha, se visualiza un escenario donde el ángulo de inclinación θ_2 es mayor que θ_1, dando como resultado que la onda reflejada no sea capturada por el transductor. Bajo este escenario la imagen de ultrasonido carecerá de información acerca de las capas de tejido de la parte anatómica en análisis. De ahí la relevancia en determinar el rango para ángulo óptimo para la posición de la aguja y obtener una representación precisa.

2.1.6 Interpretación de imágenes de ultrasonido

Una vez obtenida la imagen, es necesario reconocer ciertos patrones que representarán la estructura anatómica; por ello, se introducirán algunos términos clínicos para la compresión de la imagen de ultrasonido. Debido a la complejidad de la imagen de ultrasonido para ser interpretada; (Sehmbi & Perlas, 2022) describe 3 términos

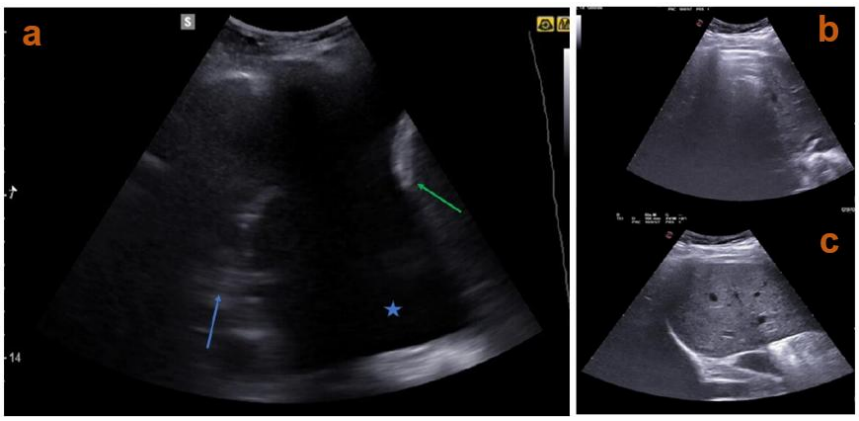
fundamentales basado en el nivel de brillo de las zonas estructurales que presenta la imagen para una mejor comprensión de estas.

- **Anecoico:** Este concepto hace énfasis a las áreas que no retornan el eco (**color negro**) describiendo zonas de fluidos, venas, arterias, quistes.
- **Hipoecoico:** Está área se ilustra en baja reflectividad (color en **escala de grises**) representando tumores, nudos linfáticos, órganos.
- **Hiperecoico:** Áreas brillantes (**color blanco**), el cual representa los huesos, estructuras calcificadas, tejido graso.

Por ejemplo, en la **Figura 15**, la imagen ubicada a la izquierda describe el pulmón con presencia de efusión pleural señalado por la estrella azul (área anecoica), el diafragma está representado por el área hiperecoica señalado por la flecha verde y, por último, el pulmón afectado representado por el área hipoecoica señalado por la flecha azul. Mientras que en las imágenes de la derecha se observa un pulmón sin alteración; es decir, sin áreas anecoicas.

Figura 15

Comparación de pulmón dañado y en condición saludable



Nota. Representación de efusión pleural (a), imágenes de pulmón en condición saludable (b y c). La parte superior de cada imagen representa la pared torácica. Adaptado de (Boccatonda, y otros, 2024)

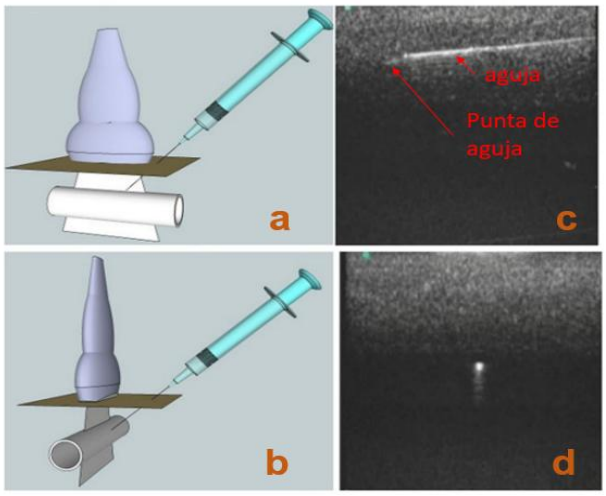
Luego de abordar la interpretación de las áreas de una imagen de ultrasonido y sus términos clínicos, en el rango de niveles de grises desde negro a blanco, es necesario identificar como es que debe interactuar este equipo médico con la inserción de la aguja.

2.1.7 Técnica de inserción de aguja

Una óptima visualización y localización precisa durante el procedimiento de inserción de aguja dependerá en gran medida del ángulo de inserción. Siendo fundamental que, a medida que la aguja penetre la estructura anatómica, se obtenga una representación visual clara que facilite la vista longitudinal y la punta de la aguja (Kimbowa, y otros, 2024). La **Figura 16** ilustra dos técnicas de inserción de aguja. La primera técnica, como se muestra en la imagen superior izquierda, consiste en ubicar la aguja alineada de manera lateral al escáner de ultrasonido. La segunda técnica, representada en la imagen inferior izquierda, la aguja se posiciona delante del escáner. Como resultado, se identifica que la primera configuración es la óptima, ya que la imagen superior derecha, grafica claramente la posición y vista axial de la aguja, así como su punta.

Figura 16

Inserción de aguja respecto del escáner de ultrasonido



Nota. Técnicas de inserción: alineación lateral (a), alineación frontal (b). Resultados: Visualización de la primera técnica (c), visualización de la segunda técnica (d). Adaptada de (Kimbowa, y otros, 2024)

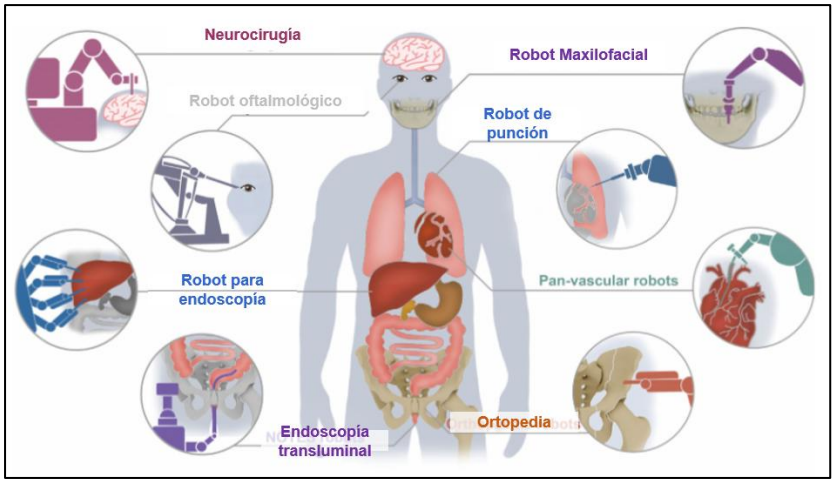
2.2 Sistema Robótico

Según (Song & Qinglei, 2024), la robótica es una ciencia en constante desarrollo tanto en el entorno académico como en la industria logrando avances sin precedentes en la última década debido a su integración con tópicos de IA, ciencia de los materiales, biología entre otros campos. Gracias a estos avances, ha sido posible abordar desafíos

globales de la sociedad en áreas como la educación, la reducción de la pobreza, energía, agricultura y la salud. Para enfrentar estos retos globales, (Murphy, 2019) destaca un enfoque particular en su libro, mencionando que la robótica permite la interacción humana a distancia en entornos de pequeña escala que requieran alta precisión como por ejemplo en cirugías. En el ámbito médico, la integración de la robótica y tecnologías innovadoras ha logrado resultados sobresalientes al promover modernización en términos de atención médica, tales como automatización de la administración de medicamentos, la mejora de la logística hospitalaria y la optimización de procesos complejos, como las intervenciones quirúrgicas que requieren alta eficiencia y precisión (Kangasniemi, Kari, Colley, & Voutilainen, 2019; Fragapane, Hvolby, Sgarbossa, & Strandhagen, 2020; Knudsen, Ghaffar, Ma, & Hung, 2024), según como se ilustra en la Figura 17. Estas contribuciones son significativas para abordar las carencias del sistema de salud; sin embargo, existen oportunidades de mejora donde la robótica interviene en procedimientos de cirugías complejas con el objetivo de garantizar la seguridad de los pacientes y lograr una eficiencia operativa en los sistemas de atención médica.

Figura 17

Aplicaciones de la robótica para intervenciones mínimamente invasivas



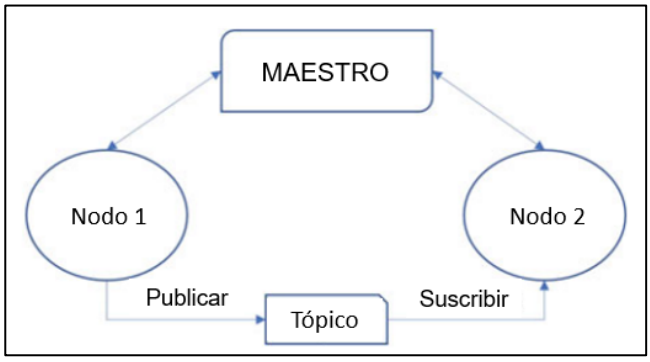
Nota. Adaptado de (Liu, y otros, 2024).

2.2.1 Robot Operating System (ROS)

El sistema operativo de robot (ROS) es un software libre que agrupa librerías, interfaces y herramientas para el desarrollo de proyectos de robótica avanzada. Esta plataforma, ROS, proporciona una fácil conexión entre sensores, actuadores y sistemas de control al utilizar un sistema de comunicación basado en “tópicos” y “mensajes”. Sin duda, esto representa una gran ventaja lo que simplifica la interacción entre ellos recibiendo y transmitiendo información de manera eficiente entre los diferentes componentes. Es importante recalcar que, aunque su nombre se describa como un sistema operativo, técnicamente no lo es, sino un middleware que actúa entre el sistema operativo y las aplicaciones robóticas. En esta tesis, el sistema operativo que gobernará la interacción de componentes será Linux, ofreciendo estabilidad, compatibilidad y flexibilidad con las herramientas de ROS. A continuación, se describirá en detalle algunos conceptos útiles que soportarán el desarrollo de este proyecto robótico. Según (Koubaa, 2023), los conceptos básicos de ROS están constituidos por nodos, maestro, mensajes y tópicos. Estos conceptos son esenciales para gestionar la comunicación entre procesos que involucren, procesamiento de datos registrados por los sensores, características de control y mensajes en general. En la **Figura 18** se observa la interacción de los conceptos mencionados aportando una eficiente comunicación.

Figura 18

Arquitectura de comunicación en ROS



Nota. Adaptado de Robot Operative System (ROS). (Koubaa, 2023)

A continuación, se profundizará en los conceptos descritos en la arquitectura que se visualiza en la **Figura 18**.

- ***Maestro:*** Trabaja como un nodo principal el cual habilita a todos los otros nodos, y activa la comunicación entre ellos para el intercambio de mensajes.
- ***Nodos:*** Los nodos contienen información, la cual se transmite al nodo central (Maestro). Asimismo, son los encargados de intercambiar mensajes con otros nodos y, a su vez, contienen archivos ejecutables escritos en C++ o Python.
- ***Mensajes:*** Los mensajes transportan la información que se transfiere entre nodos. Estos mensajes pueden ser del tipo: entero, punto flotante, booleano, matrices, etc.
- ***Tópicos:*** El medio por el cual se transportan los mensajes de nodo a nodo a través del método publicar / suscribir. Un nodo que emite un mensaje genera la acción de publicar a través de un nodo asignado; paralelamente, un nodo que requiere saber un tipo de dato se suscribe a un tópico específico.
- ***Servicios:*** Los servicios son estructuras más complejas destinadas para proveer rutinas definidas, en las que se establecen dos perfiles: El servidor y el cliente. En este caso, el servidor contiene la funcionalidad de la rutina mientras el cliente llama o solicita la funcionalidad del servicio. (The Construct, n.d.)
- ***Acciones:*** Las acciones tienen una funcionalidad similar a los servicios. Sin embargo, la principal diferencia es que cuando se requiere la ejecución de un servicio, el robot debe esperar la finalización del servicio; a diferencia de las acciones, que pueden ser ejecutadas de manera asíncrona; es decir, mientras otras ejecuciones se realizan a la vez. (The Construct, n.d.)

Habiendo definido los conceptos principales de la arquitectura ROS para establecer la comunicación, se puede observar en la **Figura 18** que el nodo principal, Maestro, se ubica en el centro del esquema, habilitando dos programas representados por los nodos 1 y 2. La conexión de los nodos con el maestro permite intercambiar información entre sí. Además, se visualiza que, el Nodo 1 **publica** (emisión de mensaje) información a través del tópico y a su vez, el nodo 2 se **suscribe** (recepción del mensaje) al tópico para recibir la información.

El framework de ROS se encuentra en constante desarrollo fortaleciendo las capacidades técnicas para optimizar el desempeño de las aplicaciones robóticas. A pesar de que ROS2 se encuentra disponible y está orientado a resolver limitaciones estructurales de ROS1, algunos paquetes aún se encuentran en proceso de transición; por ello, este proyecto se implementa utilizando ROS1, específicamente la versión Noetic. Esta selección se apoya en la disponibilidad y consolidación de la documentación del paquete MoveIt en dicha versión, así como el soporte estable para franka_ros y libfranka. Por tanto, basado en los criterios de estabilidad y confiabilidad, se opta por el framework ROS1 el cual permite un entorno robusto para el desarrollo del sistema robótico propuesto.

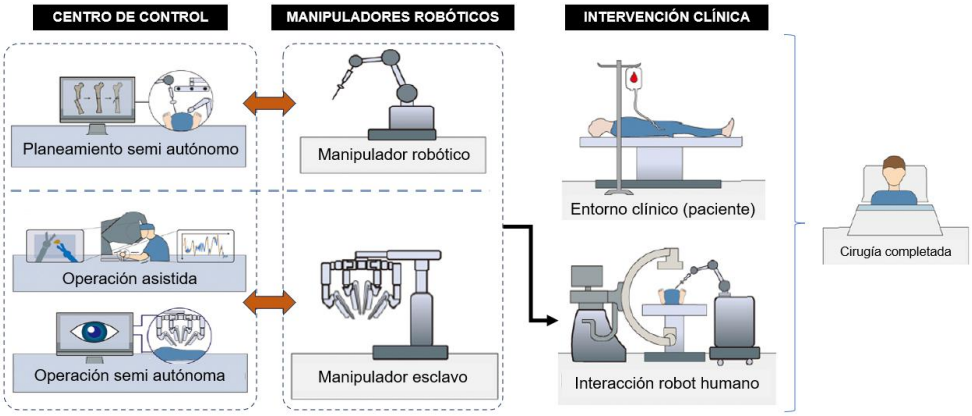
2.2.2 Manipuladores robóticos en entornos clínicos

En el ámbito de la salud, las cirugías mínimamente invasivas (MIS) han experimentado un avance significativo debido al impulso de la tecnología robótica. Según (Kenanoglu, Le Mesle, Luarasi, Sadeghian, & Haddadin, 2024), uno de los enfoques para abordar este procedimiento consiste en emplear ya sea uno o dos manipuladores robóticos ubicados alrededor del paciente; mientras que el cirujano monitorea el despliegue del sistema a través de una consola. Por otro lado, (Liu, y otros, 2024) mencionan que a estos manipuladores se les incorpora técnicas de IA, lo que permite simular y extender las capacidades de la inteligencia humana al manipulador, de tal manera que el manipulador sea capaz de reconocer el entorno del escenario operacional para llevar a cabo la tarea programada. En la **Figura 19**, se observa una arquitectura que abarca ambos enfoques, lo

que permite identificar la interacción del equipo de manipuladores en una intervención clínica al paciente.

Figura 19

Arquitectura de una intervención clínica asistida por IA

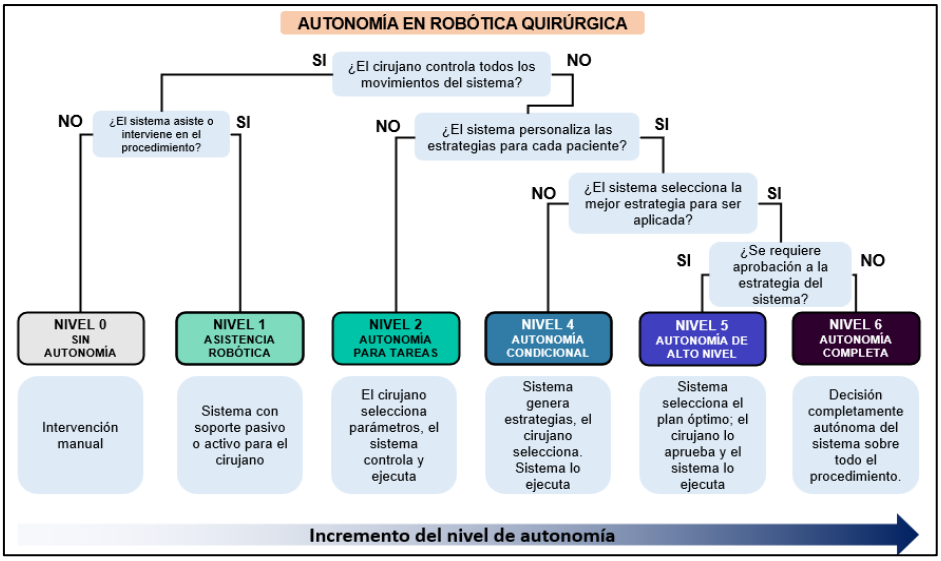


Nota. Esquema del flujo de trabajo en procedimiento quirúrgico asistido por cirugía robótica. Adaptado de (Liu, y otros, 2024).

Sin embargo, de acuerdo con (Lee, Baker, Bederson, & Rapoport, 2024), a medida que se incorporan subsistemas como visión artificial y/o instrumentos médicos, el grado de complejidad operativa aumenta, lo que implica que el sistema deba disponer de cierto grado de autonomía en la aplicación de la robótica en entornos quirúrgicos, como se describe en la **Figura 20**.

Figura 20

Niveles de autonomía en robótica quirúrgica



Nota. Características de cada nivel de autonomía de un manipulador robótico. Adaptado de (Lee, Baker, Bederson, & Rapoport, 2024)

De acuerdo con (Abbyasov, Zagirov, Gamberov, Li, & Magid, 2024; Robotics Knowledgebase, 2023), los manipuladores robóticos ejercen un rol crucial en aspectos de eficiencia y seguridad durante la intervención al paciente, ya que está diseñados para replicar los movimientos de la mano del cirujano con alta precisión y estabilidad. Para garantizar estos aspectos, el manipulador robótico debe contar con las siguientes características técnicas:

- ***Base fija:*** Proporciona estabilidad a la estructura del brazo robótico mientras se ejecutan los movimientos con suavidad y alta precisión.
- ***Diseño reconfigurable de peso ligero:*** En entornos quirúrgicos donde el espacio es limitado, se requieren un sistema ligero que reduzca la inercia del sistema para una interacción más suave.
- ***Actuadores de alta precisión:*** Controlar el movimiento preciso de las articulaciones y de sus torques para estabilidad en intervenciones quirúrgicas.

- **Alto desempeño dinámico:** La interacción dinámica debe ser estable, precisa y de alta repetibilidad, capaz de otorgar respuestas rápidas.
- **7 grados de libertad:** Esta característica es esencial en un entorno donde la precisión es un aspecto crítico, siendo que la redundancia del manipulador otorga la flexibilidad que se requiere para superar las singularidades y garantizar una trayectoria libre de obstáculos del brazo completo.

De acuerdo al análisis realizado por (Trabalza Marinucci, y otros, 2025), para una intervención clínica en la zona torácica, se demuestra que un robot manipulador tiene ventajas sobre las aplicaciones manuales e incluso por aquellas que son de video asistencia. Según (Raji, Gopaul, Babineau, Popovic, & Marquez-Chin, 2025; Bai, Zhao, & Ren, 2025), respecto a la variedad de los manipuladores robóticos más usados en tareas de alta precisión; mencionan que los robot Kuka, ABB, UR y Franka Emika Panda son altamente empleados para este tipo de tareas colaborativas donde el robot de Franka destaca por su sensibilidad y repetibilidad; así como también la compatibilidad con entornos de desarrollo, particularmente con ROS y MoveIt, permitiendo una interacción en tiempo real. Dicha compatibilidad también facilita la integración de un módulo personalizado adjunto al efector final del brazo robótico.

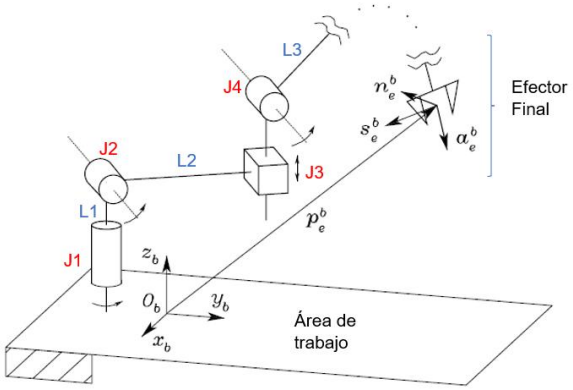
2.3 Control del robot y planificación de movimiento

El control del robot constituye una etapa fundamental de este proyecto, ya que permitirá ejecutar los movimientos con precisión. Antes de profundizar en los aspectos dinámicos del robot, se introducirán los elementos que forman parte de un manipulador robótico representados gráficamente en la **Figura 21**:

- **Links:** Elementos rígidos que conectan las articulaciones.
- **Articulaciones (Joints):** Elementos que proporcionan el efecto de rotación para el siguiente link conectado.
- **Efector Final:** Mecanismo que permite la manipulación de objetos

Figura 21

Elementos estructurales de un robot



Nota. En la imagen se visualizan 3 links (L1, L2, L3), 4 articulaciones (J1, J2, J3, J4) y un efector final para manipular o sostener elementos del entorno de trabajo. Adaptado de (Siciliano, Sciavicco, Villani, & Oriolo, 2008)

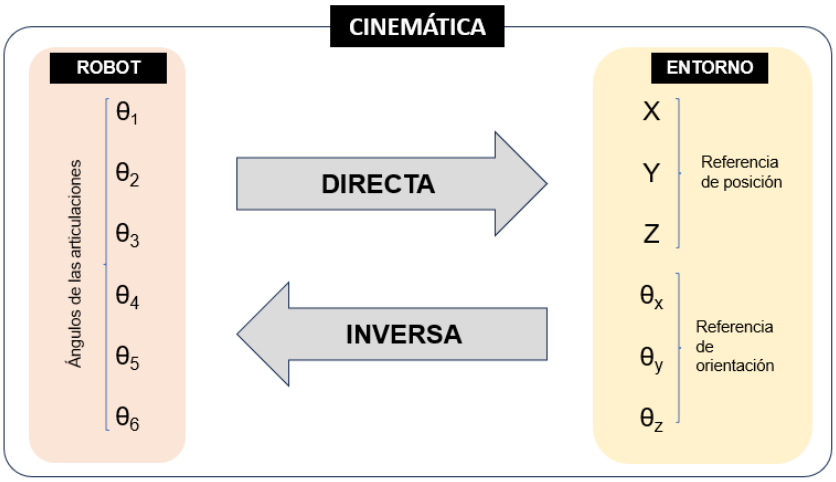
2.3.1 Análisis Cinemático.

El análisis cinemático de un robot consiste en el estudio del movimiento de sus elementos. Considerando que un robot está compuesto de partes rígidas (elementos) y articulaciones, este apartado se centrará en describir la posición de los elementos principalmente del efector final basado en los ángulos articulares. (Siciliano, Sciavicco, Villani, & Oriolo, 2008)

Para abordar las definiciones que comprenden el análisis cinemático (Talli & Giriyapur, 2021) se considerará un robot de 6GDL el cual tiene una gran variedad de aplicaciones industriales relacionados a procesos de soldadura, pintado, en procesos productivos, entre otros. En un entorno real, los tres primeros grados de libertad corresponden a la posición del efector final en un marco de referencia cartesiano, XYZ, mientras que los otros tres grados de libertad determina la orientación del efector final.

Figura 22

Síntesis de análisis cinemático.

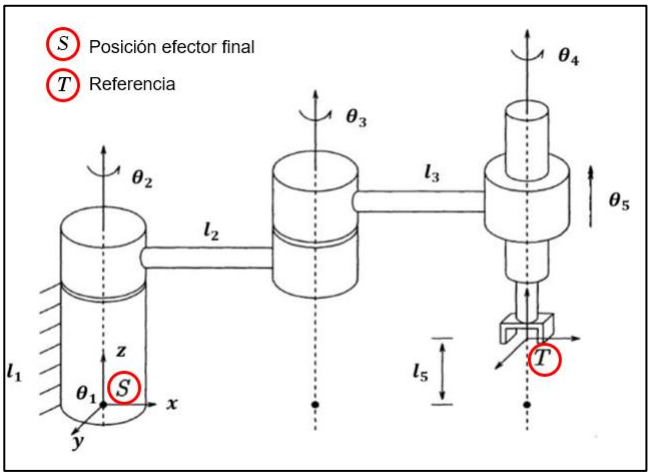


Nota. Adaptado de (Elhadidy, Abdalla, Abdelrahman, Elnaggar, & Elhosseini, 2024; Talli & Giriyapur, 2021)

Como se puede observar en la **Figura 22**, se representan dos enfoques de análisis cinemático: cinemática directa y cinemática inversa. Respecto a la cinemática directa, se determina la posición y orientación del efector final a partir de los ángulos articulares; mientras que la cinemática inversa resuelve el problema opuesto, es decir, se determina los ángulos articulares a partir de los parámetros de posición y orientación en el entorno. Con respecto a la cinemática directa, el método de cálculo empleado es el algoritmo de Denavit Hartenberg, para lo cual será analizado un robot manipulador de 4 GDL, como se visualiza en la **Figura 23**, que detallará los ángulos de las articulaciones y medidas de los links para análisis de la matriz de transformación total, de tal manera que defina la posición y orientación del efector final (T) con respecto de la posición de referencia (S). (Yasmini, Artha, & Gunadi, 2025).

Figura 23

Descripción de un robot de 4GL

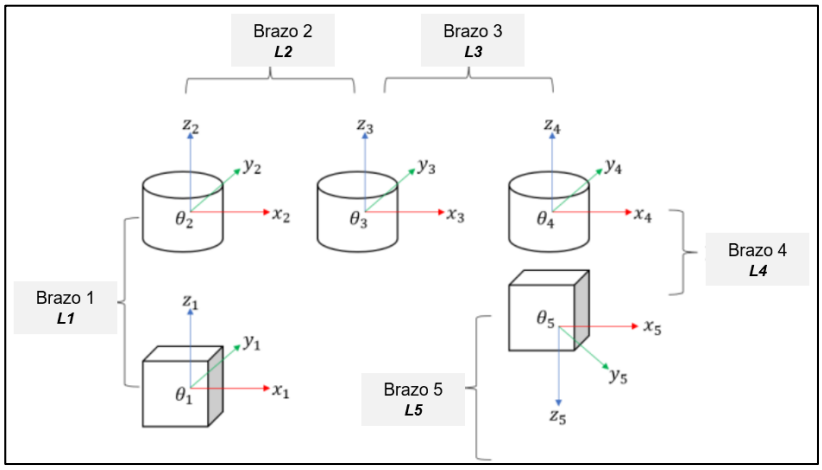


Nota. Adaptado de (Yasmini, Artha, & Gunadi, 2025)

Para aplicar el algoritmo de Denavit Hartenberg, se debe ubicar los sistemas de coordenadas XYZ de cada articulación, tal como se representa en la **Figura 24**, para determinar los parámetros que permiten el cálculo de la matriz de transformación

Figura 24

Coordenadas de cada link del robot



Nota. X_5, Y_5, Z_5, referencias del efector final. Adaptado de (Yasmini, Artha, & Gunadi, 2025)

Luego de ubicar el sistema de coordenadas en cada articulación, se establecen los parámetros de cada link según la **Tabla 5**, los cuáles se usarán para determinar la matriz de transformación.

Tabla 5

Parámetros para el algoritmo Denavit Hartenberg

LINK	θ	d	a	α	Información
1	*0*	*L1*	*0*	*0*	*Referencia*
2	θ_2	*0*	*L2*	*0*	*Brazo 1*
3	θ_3	*0*	*L3*	*0*	*Brazo 2*
4	θ_4	*0*	*0*	*180°*	*Brazo 3*
5	*0*	*L5*	*0*	*0*	*Efector Final*

Nota. Adaptado de (Yasmini, Artha, & Gunadi, 2025)

Es necesario tener en cuenta que la matriz, representada por la notación A_i, involucra los parámetros descritos en la **Tabla 5** mediante la siguiente fórmula.

$$A_i = R_{\theta_i} * T_{d_i} * T_{a_i} * R_{\alpha_i} \quad (1)$$

De la ecuación *(1)*, tenemos las siguientes notaciones que hacen referencia a los parámetros de la **Tabla 5**.

A_i: Resultado de la matriz de transformación para el link i

R_{θ_i}: Rotación del link i respecto del eje Z.

T_{d_i}: Traslación del link i respecto del eje Z.

T_{a_i}: Traslación del link i respecto del eje X.

R_{α_i}: Rotación del link i respecto del eje X.

Seguidamente, se describe la misma ecuación *(1)* con el contenido de las matrices de rotación y traslación basado los parámetros correspondientes para obtener la matriz de transformación generalizada como se visualiza en la ecuación *(2).*

$$A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & -\sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

$$A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & -\sin\theta_i \sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

La ecuación *(3)* representa el resultado de la matriz de transformación homogénea (A_i) basado en los parámetros para cada elemento i. De acuerdo con (Kong, y otros, 2024), se puede emplear la siguiente notación, como se representa en la ecuación *(4)*, para reducir la cantidad de términos y comprender de una manera simplificada lo que representa esta matriz.

$$A_i = \begin{bmatrix} r_{3*3} & p_{3*1} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

En la matriz simplificada se observan los términos r_{3*3} y p_{3*1}, los cuales representan los parámetros de orientación y posición, respectivamente, del elemento i . Entonces, al aplicar dicha matriz de transformación al robot manipulador de 4GL, considerando los 5 links de la **Tabla 5**, se obtiene la expresión de la ecuación *(5)*:

$$A = A_5 * A_4 * A_3 * A_2 * A_1 \quad (5)$$

De la ecuación (5) se obtiene la matriz de transformación homogénea, A, para el sistema mediante la aplicación del algoritmo Denavit Hartenberg; es decir que este resultado facilitará la posición del efector final “T” respecto de la posición de referencia “S”.

$$T = A * S \quad (6)$$

Finalmente, la ecuación *(6*) resume el resultado de la aplicación del algoritmo de Denavit Hartenberg en el proceso de cinemática directa para detectar la posición del efector final (T) respecto de la referencia principal del sistema (S) el cual también se toma como la referencia global del sistema de coordenadas.

2.3.2 Planificación de Trayectoria

La planificación de trayectoria de un robot consiste en definir la secuencia de movimientos del efector final hacia una posición objetivo del robot a partir de una posición inicial dada, considerando las limitaciones dentro de su área de trabajo y de los límites dimensionales del robot.

De acuerdo con (MoveIt Tutorials, n.d.), los algoritmos de planeación permiten elaborar la secuencia de movimientos dentro del espacio de configuración, los cuales se pueden establecer en dos enfoques: los algoritmos basados en la técnica de muestro tales como: Árbol aleatorio de expansión rápida (RRT) o Mapa probabilístico de cambios (PRM) que permiten gestionar la planificación a partir de la construcción de trayectorias libres de colisión. Mientras que el segundo enfoque destaca los algoritmos para planeación de movimiento basado en la optimización, a través de un enfoque de gradientes (CHOMP) o a través de técnicas estocásticas (STOMP) para gestionar trayectorias más suaves.

En aplicaciones quirúrgicas donde la precisión es un aspecto crítico, se prioriza una planificación directa dentro del espacio cartesiano. Basado en este enfoque y alineando el planeamiento de la trayectoria del efector final con el entorno ROS, se considerará un planificación directa en el espacio cartesiano, a través del método *compute_cartesian_path()* del paquete MoveIt, tal como se describe en el **ANEXO D**. MoveIt se encarga de ejecutar la cinemática cartesiana para determinar las poses intermedias dentro de la secuencia de movimiento; y, además, este método asegura que el efector final mantenga la misma orientación (vertical) a lo largo de la trayectoria.

Como un ejemplo complementario, (Dias, Souza, Rocha, Figueiredo, & Silva, 2024), describen una forma eficiente del planeamiento de trayectoria para un manipulador robótico resaltando estados de referencia del robot para iniciar la secuencia de movimientos y validación de la trayectoria, siendo esta última un aspecto relevante para identificar que dichas referencias se encuentren dentro del área de trabajo del robot, así como evitar la colisión de sus partes durante el movimiento.

En la **Figura 25** se representa gráficamente los estados de referencia principales del robot. siendo la

Figura 25

Estados de referencia del robot

Nota. Tomado de (Dias, Souza, Rocha, Figueiredo, & Silva, 2024) **Posición Inicial** (Home Position) mostrado en la imagen de la izquierda (a). Seguidamente, la **posición A** representada en la imagen central (b) y la **posición B** en la imagen derecha (c), describen estados de referencia intermedio y final respectivamente.

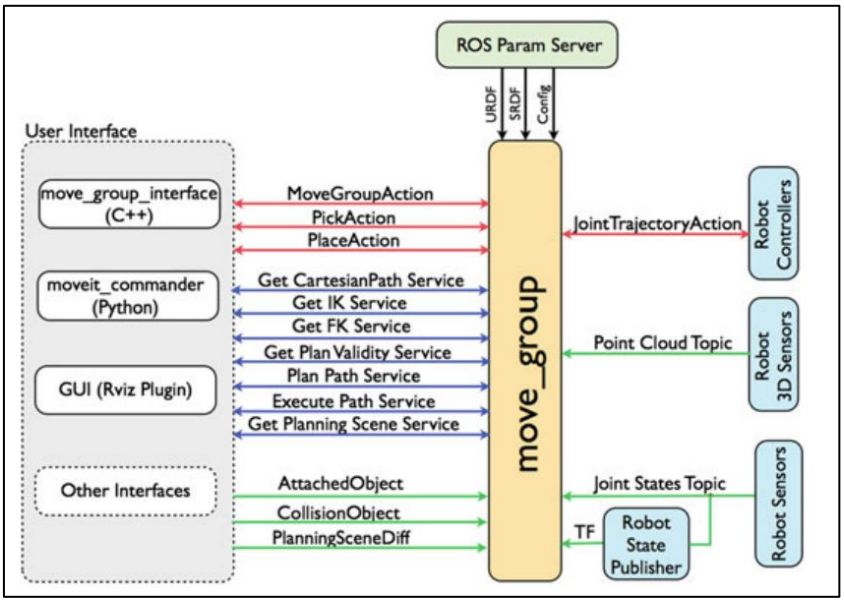
2.3.3 Control autónomo mediante ROS

En este apartado se presentará la interacción de ROS desde un enfoque respecto al movimiento del robot, control de trayectoria e información esencial relacionado a ROS para dicho control. De acuerdo con (Chitta, 2016), MoveIt es un superpaquete que contiene otros paquetes para aplicaciones robóticas avanzadas que abarca tareas de planificación de trayectoria, visualizador interactivo 3D, cinemática, control y navegación en más de 65 robots diferentes como por ejemplo Franka Emika Panda, KuKa, UR, entre otros.

La **Figura 26** muestra los principales elementos de la arquitectura del superpaquete MoveIt. Basado en dicha arquitectura, “move_group” es el nodo principal para desplegar servicios y acciones que gestionen la cinemática del robot.

Figura 26

Arquitectura de MoveIt!



Nota. Tomado de (Chitta, 2016)

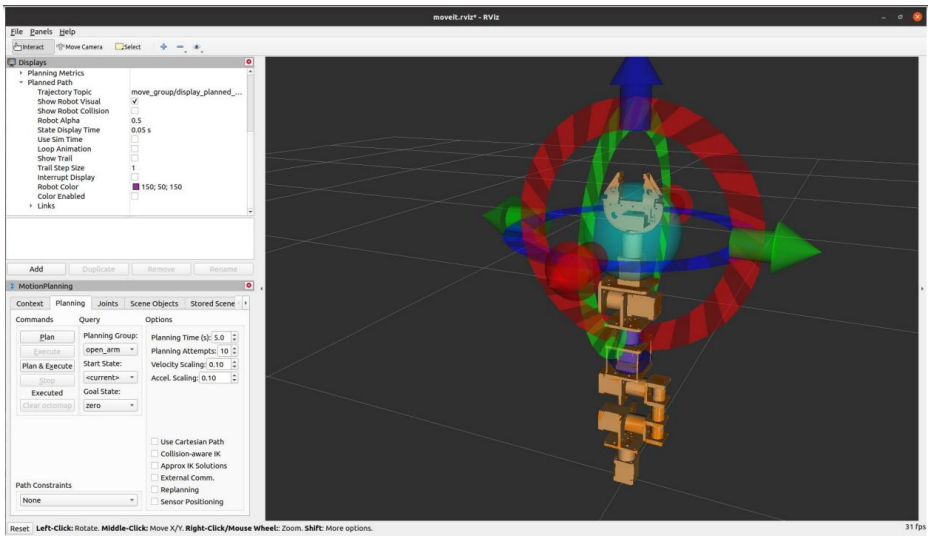
Particularmente, el paquete “moveit_commander” trabaja como una interfaz de programación de aplicaciones (API) enfocado en scripts de Python. De manera similar, el paquete “move_group_interface” con el lenguaje C++.

Respecto al paquete GUI, este proporciona un interfaz para visualización de interacción de robots mediante MoveIt, además de demostraciones visuales de las rutinas de los algoritmos descritos como se describe en la **Figura 27** al describir la configuración de un brazo robótico personalizado de 6 grados de libertad (GDL) elaborado en CAD SolidWorks mostrando el potencial de MoveIt siendo un análisis relevante para una posterior implementación en físico (Eaton & Tanveer, 2024).

Como se observa, el metapaquete “MoveIt” está estructurado para construir aplicaciones a través de rutinas en Python y/o C++, además de tener una representación gráfica de los movimientos de los componentes de un robot.

Figura 27

Representación gráfica en el entorno Rviz



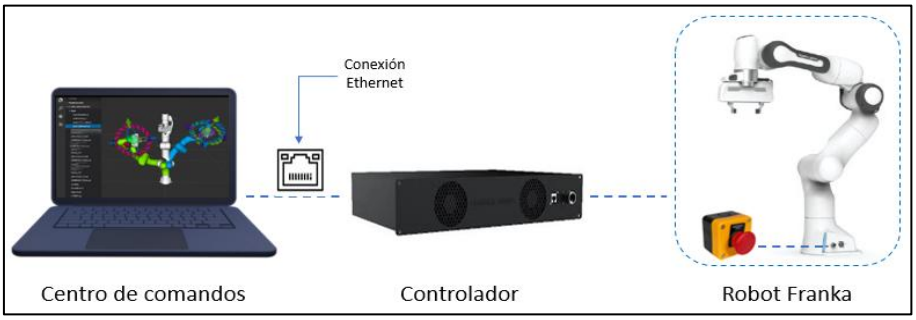
Nota. Tomado de (Eaton & Tanveer, 2024)

2.3.4 Conexión del Robot

En la **Figura 28** se representa la conexión del robot con centro de comandos. Esta gráfica es relevante porque resalta que no solo se gestiona la comunicación del robot a través de una conexión directa, sino que se lleva a cabo a través de un dispositivo que es el controlador, permitiendo una comunicación entre el robot y la programación desarrollada en el entorno ROS; con la finalidad de determinar la posición del efector final según el procedimiento establecido.

Figura 28

Conexión del Robot



Nota. Elaboración propia.

2.4 Sistema de Visión Artificial

Las cirugías mínimamente invasivas asistidas por manipuladores robóticos suelen incorporar un sistema de visión artificial, para facilitar el campo visual que el mecanismo requiere para replicar los movimientos del brazo del cirujano. Bajo este enfoque, la visión artificial es fundamental para guiar al manipulador durante la intervención. En esta sección, se describirán algunos conceptos y fundamentos del sistema de visión artificial, incluyendo la captura, preprocesamiento, segmentación y extracción de características relevantes.

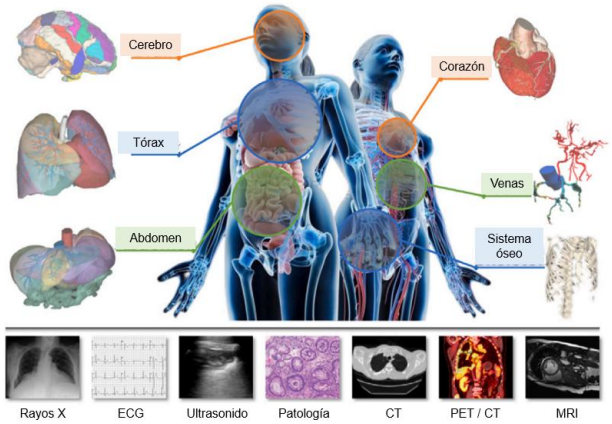
2.4.1 Captura de la imagen a procesar

La adquisición de la imagen representa la primera etapa del procesamiento visual. Para esta fase se requiere el uso de un dispositivo físico que integre un mecanismo sensible (sensor) que opere dentro del espectro electromagnético según la aplicación. Por ejemplo, las cámaras digitales operan dentro del espectro visible, mientras que las cámaras térmicas dentro de la banda infrarroja. En frecuencias bajas, el rango de las ondas de radio podría usarse el escáner de ultrasonido como alternativa.

En la **Figura 29**, se ilustran modalidades de imágenes médicas asociadas a regiones anatómicas. En la parte superior, se destacan los órganos y sistemas del cuerpo humano usualmente examinados, mientras que en la zona inferior se destacan imágenes médicas como radiografías, electrocardiogramas, ultrasonido, tomografía computarizada (CT), resonancia magnética (MRI), entre otros. Estás modalidades presentan características visuales particulares, por lo que se requiere diseño de algoritmos apropiados para procesar, segmentar y extraer los patrones relevantes según el tipo de aplicación.

Figura 29

Variedad de modalidades de imagen médica



Nota. En la zona superior de ejemplifican representaciones visuales de zonas anatómicas, mientras que en la zona inferior de la imagen se distingue la variedad de formaciones de imágenes médicas. Adaptado de (Zhang & Metaxas, 2024)

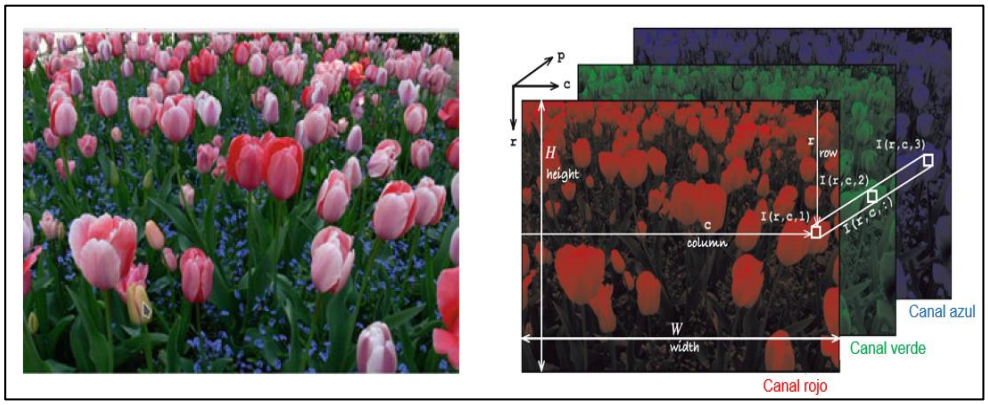
2.4.2 Pre - procesamiento de la imagen

Esta etapa del preprocesamiento se inicia la transformación de la imagen lo cual consiste en trabajar los formatos de color adecuados y aplicación de filtros para extraer la información de una manera eficiente.

La representación de la gran mayoría de imágenes está basada en los colores primarios rojo (R), verde (G) y azul (A); de ahí se desprende su nombre RGB, los cuales encajan en cada plano (canal) como se muestra en la **Figura 30**.

Figura 30

Imagen digital en formato RGB



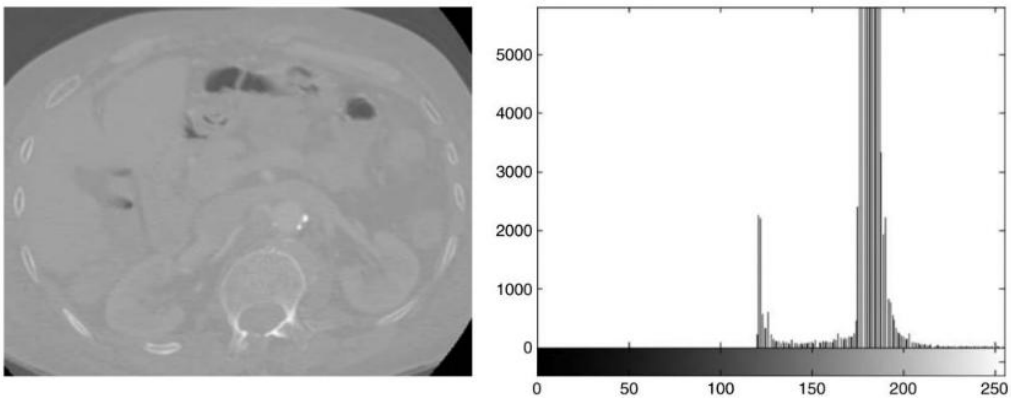
Nota. En la imagen de la izquierda se observa en formato RGB y en la derecha se observa la misma imagen representado en sus canales por separado. Adaptado de (Corke, Jachimczyk, & Pillat, 2023)

Además, de acuerdo con la **Figura 30**, la intensidad de un pixel ubicado en la fila "r" y columna "c", está compuesto por la combinación de sus respectivos valores de cada canal.

Respecto a la imagen digital en escala de grises, consiste en un solo canal representando la intensidad del brillo en cada pixel en el rango de 0 a $2^K - 1$, en la que usualmente el valor de "k" es igual a 8 Entonces, el valor de 0 representa al pixel más oscuro (negro) y el valor de 255 al pixel más brillante (blanco) (Burger & Burge, 2022). Por ejemplo, en la **Figura 31** se muestra una imagen obtenida de una tomografía computarizada de vista axial (transversal), donde cada pixel está representado por un valor de los 255 niveles en la escala de grises. A la derecha, se muestra un histograma el cual detalla la distribución de la cantidad total de pixeles (eje y) por cada nivel de la intensidad de escala de grises (eje x).

Figura 31

Imagen digital en formato de escala de grises



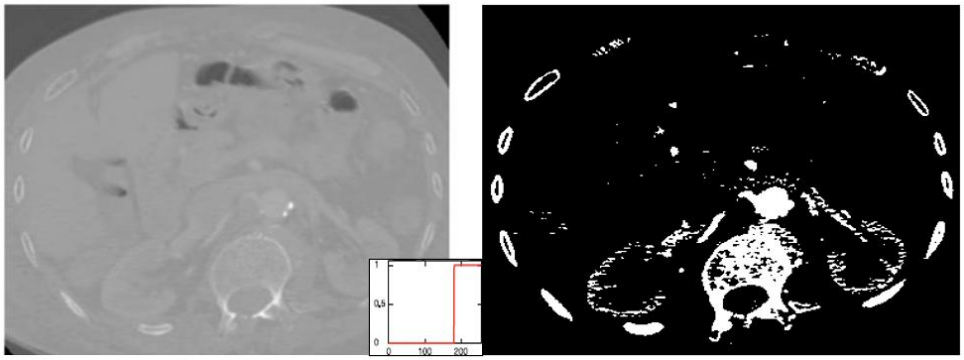
Nota. Imagen axial de tomografía computarizada (izquierda), histograma de intensidades en escala de grises (derecha). Tomado de (Bankman, 2008)

Parte de la estrategia de mejora de la imagen, existe el formato binario, el cuál solo contiene 2 tipos de pixeles negro y blanco. En este caso, se elige un valor apropiado entre 0 y 255, el cual es llamado umbral, de tal manera que los pixeles que tengan una intensidad menor que el umbral, se convertirán en negros, y aquellos que superen el umbral su intensidad será modificada a blanco. En la **Figura 32**, la imagen de la izquierda representa

la imagen mostrada en la figura anterior obtenida de una tomografía computarizada axial, mientras que a la derecha se representa la misma imagen en formato binario considerando un umbral de 185.

Figura 32

Imagen digital en formato binario



Nota. La imagen de la izquierda se encuentra en un formato de escala de grises, mientras que la derecha representa la misma imagen en una escala binaria, umbral seleccionado: 185 como se observa en el histograma reducido ubicado en la parte central. Adaptado de (Bankman, 2008)

Desde un enfoque matemático, la ecuación (7) describe la representación binaria de la imagen en escala de grises.

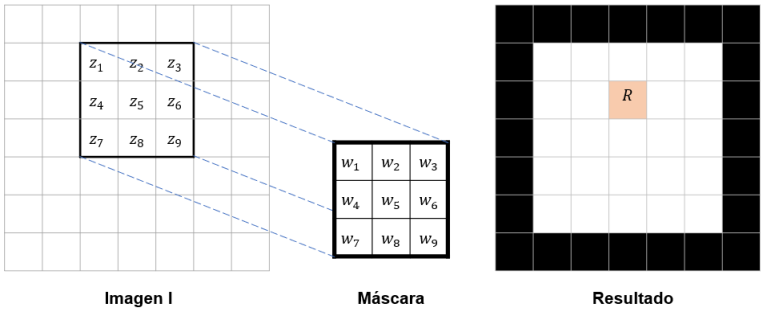
$$\begin{matrix} I_b = 0\,; & I_y < 185 \\ I_b = 1\,; & I_y > 185 \end{matrix} \qquad (7)$$

Siendo I_b la intensidad de la imagen binaria e I_y la intensidad de la imagen de la escala de grises.

Otra transformación que es ampliamente útil en las imágenes digitales es la aplicación de filtros, también conocidos como máscaras, kernels o sub-imágenes. Como resultado se obtiene una versión mejorada al reducir el ruido (suavizado), realce de los bordes, entre otros. La **Figura 33** representa la aplicación de la máscara sobre Imagen I, siendo esta máscara de un tamaño de 3x3.

Figura 33

Filtro de imagen usando máscara



Nota. Elaboración propia.

Siendo su interpretación matemática tal como se describe mediante la ecuación (8), a través de una suma ponderada entre los valores de la intensidad de los pixeles (representados por z) y los coeficientes de la máscara (representados por w).

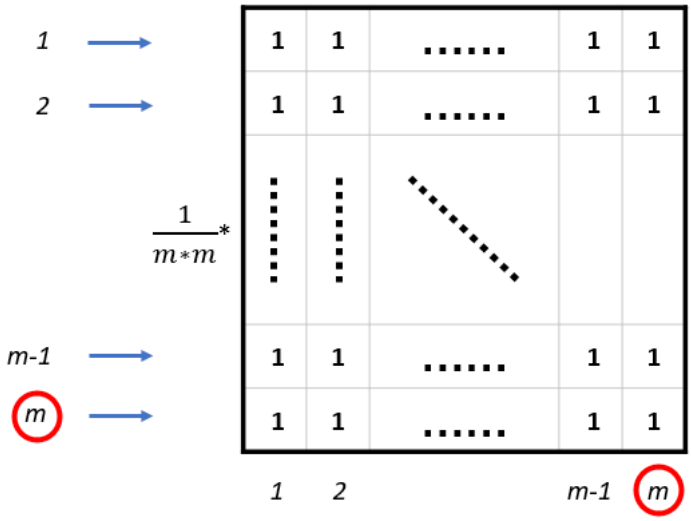
$$R = w_1 * z_1 + w_2 * z_2 + \cdots + w_9 * z_9 = \sum_{i=1}^{3*3} z_i * w_i \qquad (8)$$

Para el término de la sumatoria, el límite inferior considera al primer elemento de la máscara, mientras que el límite superior esta dado por la cantidad total de elementos de la máscara.

Para lograr el suavizado de una imagen, se aplica una máscara donde todos sus coeficientes son igual a 1 dividido por el resultado del cuadrado del tamaño del filtro (filtro de caja), como se visualiza en la **Figura 34** . En este caso, si el tamaño del filtro es m, entonces el factor para dividir será $m * m$.

Figura 34

Filtro para suavizar imágenes

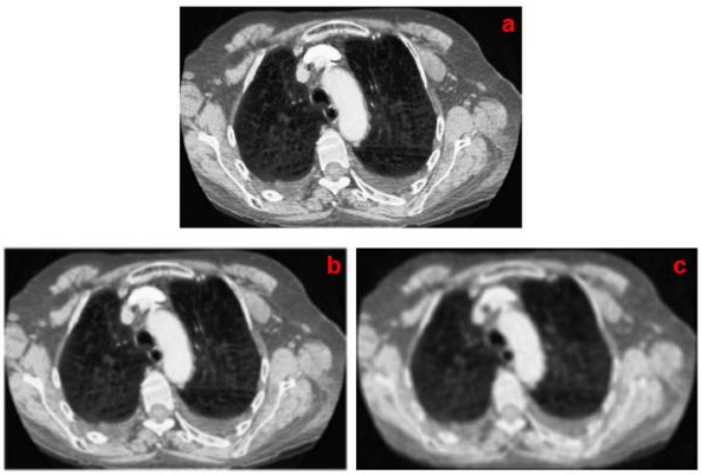


Nota. Filtro de caja de tamaño $m * m$. Elaboración propia.

Por ejemplo, en la **Figura 35** se ilustra en la zona superior la imagen original, mientras que, en la zona inferior, se visualiza los resultados de aplicar distintos tamaños de filtros de suavizado: 3*3 y 9*9 respecto de la imagen original.

Figura 35

Aplicación de filtro de suavizado



Nota. Imagen original (a), filtrado con máscara 3*3 (b), filtrado con máscara 9*9 (c). Adaptado de Adaptado de (Bankman, 2008)

Mientras el tamaño del filtro se incrementa, el resultado de la imagen se va haciendo más difuso, es decir, se va suavizando lo que implica pérdida de detalles como los bordes.

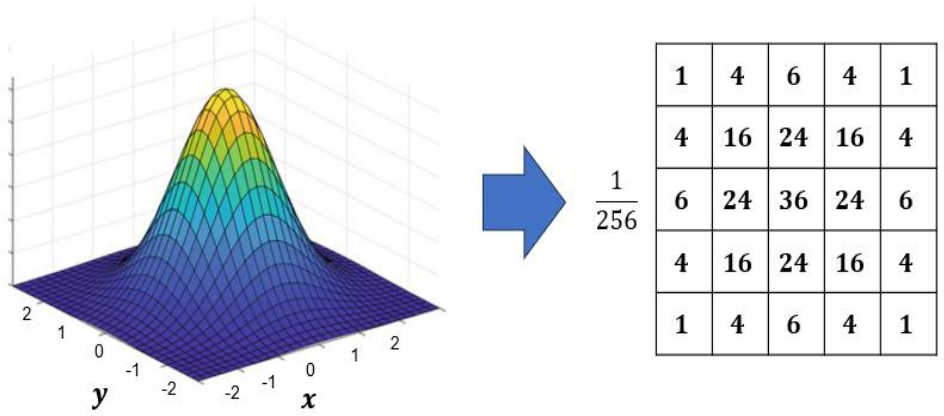
Otro filtro para suavizar una imagen es el filtro gaussiano, cuyos coeficientes son calculados a partir de la distribución de gauss de acuerdo con (Paulsen & Moeslund, 2020). Su función se describe con la ecuación (9) dependiendo de la desviación estándar σ.

$$G(x,y) = \frac{1}{2\pi\sigma^2} * e^{-\frac{x^2+y^2}{2*\sigma^2}} \quad (9)$$

La **Figura 36** ilustra dicha función en forma de campana correspondiente a una máscara $5 * 5$ y $\sigma = 1$.

Figura 36

Filtro Gaussiano



Nota. Representación gráfica de la ecuación (9) a la izquierda y el filtro de tamaño 5*5. Adaptado de (Corke, Jachimczyk, & Pillat, 2023).

El efecto de suavizado a partir de un filtro gaussiano se visualiza más natural en comparación al filtro de caja, evitando cambios bruscos de intensidad según (Nixon & Aguado, 2012).

Hasta aquí, se ha explorado los fundamentos de la imagen digital, su representación en los formatos de color que serán usados en esta tesis; así como también la transformación de la imagen antes de iniciar el procesamiento de esta.

2.4.3 Morfología

En esta etapa, las imágenes se transforman para uniformizar regiones que faciliten su análisis posterior. Para ello, se utilizan imágenes binarias para lograr cambios estructurales. Para profundizar en las técnicas de morfología, se considerará A y B conjuntos de Z^2, siendo A la imagen binaria y B el elemento estructural. En este apartado se detallará las operaciones de dilatación, erosión y cierre.

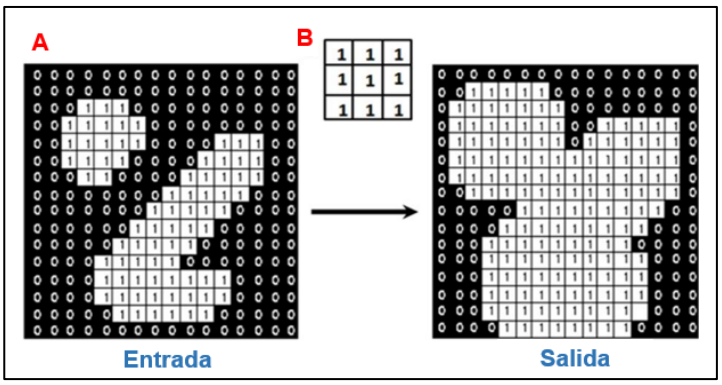
Dilatación: Esta operación extiende las regiones que son blancas bajo cierta estructura, logrando unificar regiones ligeramente fragmentadas según la ecuación (10) la cual representa la dilatación de A por B.

$$A \oplus B = \{ x \mid \left[\widehat{B_x} \cap A \right] \subseteq A \} \tag{10}$$

En la **Figura 37**, se aprecia el efecto de aplicar la operación dando como resultado una imagen nueva donde los objetos han sido expandidos.

Figura 37

Dilatación de una imagen digital



Nota. Adaptado de (Bhutada, Yashwanth, Dheeraj, & Shekar, 2022)

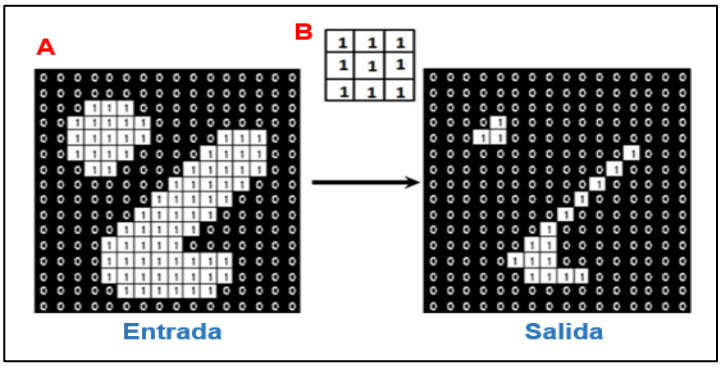
Erosión: Esta operación reduce aquellas regiones en blanco, al eliminar píxeles que no se encuentran dentro del patrón de la estructura (B). La ecuación (11) define la operación matemática de erosión de A por B.

$$A \ominus B = \{ x \mid B_x \subseteq A \} \tag{11}$$

La **Figura 38**, representa la contracción de los objetos tras aplicar la erosión.

Figura 38

Erosión de una imagen digital



Nota. Adaptado de (Bhutada, Yashwanth, Dheeraj, & Shekar, 2022)

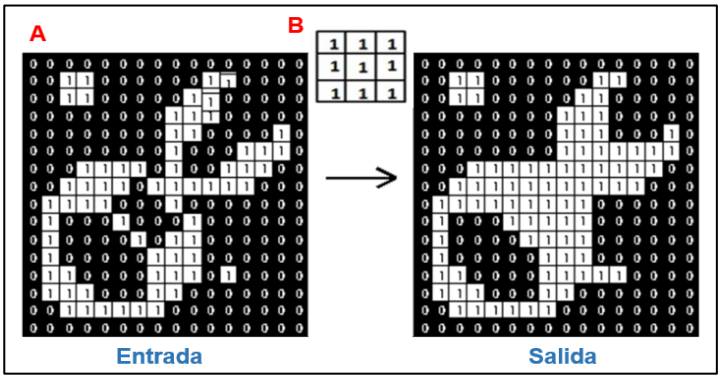
Cierre: Este concepto combina la acción de una dilatación seguida de una erosión, eliminando huecos pequeños o discontinuidades en estructuras definidas, sin afectar drásticamente su forma. La ecuación (12) describe su representación matemática.

$$A \circ B = (A \oplus B) \ominus B \tag{12}$$

La **Figura 39** muestra el resultado del cierre al suavizar el borde y rellenar espacios vacíos sin variar significativamente su forma.

Figura 39

Operación de cierre de imagen digital



Nota. Adaptado de (Bhutada, Yashwanth, Dheeraj, & Shekar, 2022)

2.4.4 Segmentación

Esta etapa de segmentación es en separar o aislar aquellos pixeles que representan un objeto específico con respecto de otros en la escena. Para llevar a cabo la

segmentación de regiones se aplican varios enfoques como la umbralización, técnicas de detección de líneas y bordes; y la transformada de Hough.

Según (Corke, Jachimczyk, & Pillat, 2023), uno de los métodos aplicados para clasificar regiones mediante las características de los pixeles es la umbralización, tal como se detalló al obtener la imagen binaria, la que consiste en establecer un valor fijo (umbral) como referencia. Expresado matemáticamente según la ecuación (13).

$$I_{m,n} = \begin{cases} 0; si\ G_{(m,\ n)} \leq t \\ 1;\ si\ G_{(m,\ n)} > t \end{cases} \quad (13)$$

donde:

- $G_{(m,\ n)}$ representa el valor de la intensidad de la imagen en escala de grises en la coordenada (m, n).
- t representa el valor del umbral.
- $I_{(m,\ n)}$ imagen binaria luego de aplicar el umbral.

Otro enfoque de la segmentación es la detección de líneas y bordes. Según (Nixon & Aguado, 2012), dicho enfoque se basa en el análisis de la diferencia de los niveles de intensidades entre pixeles adyacentes. Por ejemplo, en una imagen $\boldsymbol{I}$, un cambio de intensidad entre puntos horizontalmente conectados permite detectar un borde vertical, el cual es representado mediante el operador $\boldsymbol{Ex}$, como se describe en la ecuación (14).

$$\boldsymbol{Ex}_{x,y} = \left|\boldsymbol{I}_{x,y} - \boldsymbol{I}_{x+1,y}\right| \quad \forall\, x \in [1, N-1]; y \in [1, N] \quad (14)$$

Análogamente, para detectar un borde horizontal representado por el operador ($\boldsymbol{Ey}$) como se muestra en la ecuación (15).

$$\boldsymbol{Ey}_{x,y} = \left|\boldsymbol{I}_{x,y} - \boldsymbol{I}_{x,y+1}\right| \quad \forall\, x \in [1, N]; y \in [1, N-1] \quad (15)$$

Asimismo, (Nixon & Aguado, 2012) también resalta la serie de Taylor como una alternativa para el análisis de la diferenciación entre pixeles adyacentes con la finalidad de detectar los bordes. Esta serie está representada matemáticamente como se detalla en la ecuación

(16), siendo $\triangle x$ la diferencia entre los lugares de un pixel y f el valor de intensidad en un pixel específico de la imagen I.

$$f(x+\triangle x) = f(x) + \triangle x * f'(x) + \frac{\triangle x^2}{2!} * f''(x) + O(\triangle x^3) \quad (16)$$

Reordenando la ecuación de Taylor (16) se obtiene la ecuación (17):

$$Gx = f'(x) = \lim_{\triangle x \to 0} \frac{f(x+\triangle x) - f(x)}{\triangle x} \quad (17)$$

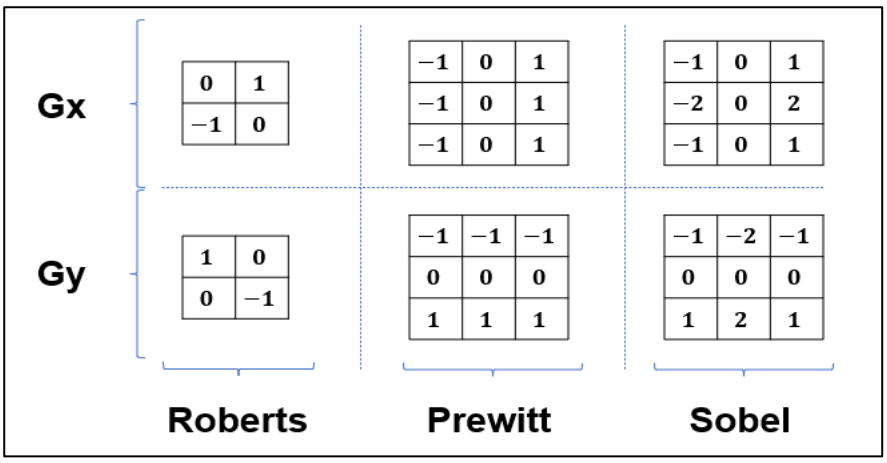
Bajo la misma lógica, se extiende la representación de la derivada a la diferencia $\triangle y$, el cual también se relaciona con el gradiente respecto del eje y, mediante la expresión matemática (18):

$$Gy = f'(y) = \lim_{\triangle y \to 0} \frac{f(y+\triangle y) - f(y)}{\triangle y} \quad (18)$$

Estas relaciones entre las intensidades de pixeles contiguos contribuyen a definir coeficientes para filtros de tamaño 2x2 o 3x3, los cuáles serán aplicados mediante la operación de convolución sobre una imagen para resaltar sus bordes. La **Figura 40**, muestra los filtros mencionados ordenados según la dirección del gradiente, es decir el gradiente Gx respecto del eje x o el gradiente Gy respecto del eje y.

Figura 40

Operadores de Roberts, Prewitt, Sobel



Nota. Adaptado de (Gonzalez & Woods, 2021)

Estas representaciones aisladas se integran para procesar el gradiente total como se observa en la ecuación (19):

$$\nabla \mathbf{f} = \begin{bmatrix} Gx \\ Gy \end{bmatrix} = \begin{bmatrix} f'(x) \\ f'(y) \end{bmatrix} \tag{19}$$

El vector gradiente, definido por $\nabla \mathbf{f}$, se descompone en dos parámetros importantes como lo es la magnitud y su dirección. La ecuación (20) manifiesta el cálculo de la magnitud del gradiente.

$$\nabla f = mag(\nabla \mathbf{f}) = [Gx^2 + Gy^2]^{\frac{1}{2}} \approx |Gx| + |Gy| \tag{20}$$

La **Figura 41**, muestra el resultado de la aplicación de las ecuaciones (17), (18) y (20) sobre una imagen en escala de grises dando como resultado las imágenes b, c y d, respectivamente.

Figura 41

Aplicación de operador gradiente

Nota. Imagen original (a), componente Gy (b), componente Gx (c), resultado de utilizar la ecuación (20). Adaptado de (Gonzalez & Woods, 2021).

El segundo parámetro es el ángulo de dirección del vector gradiente, $\nabla \mathbf{f}$, respecto del eje y.

$$\alpha(x, y) = \tan^{-1}\left(\frac{Gy}{Gx}\right) \quad (21)$$

Estos parámetros son esenciales para el algoritmo de la detección de borde mediante el método de Canny, conocido como el mejor de todos, dado que optimiza la línea detectada basado en los siguientes tres criterios clave, tal como lo describe (Solomon & Breckon, Image Segmentation, 2011).

- **Reducción de error**, minimizar la ausencia de pixeles en la detección de bordes y evitar detectar ruido como parte del borde.
- **Buena localización**, siendo que los pixeles detectados deben estar cercanos a los pixeles del borde real.
- **Única respuesta**, es decir, la detección debe detectar un solo punto respecto del borde real, para evitar múltiples respuestas.

El proceso de detección de bordes mediante el método de Canny, se describe en la **Figura 42**, a través del seudocódigo detallando el proceso de aplicación sobre una imagen.

Figura 42

Seudocódigo para método de detección de Canny

Detector de Bordes Canny $(I, \sigma, \theta_h, \theta_l)$

% 1. Aplicar el filtro gaussiano para suavizar la imagen
INPUT : Imagen original (I)
Igauss ← Filtro_Gaussiano (I, σ)

% 2. Computar gradientes
$[Gx, Gy]$: Gradiente $(Igauss)$
Magnitud_grad ← $\text{sqrt}(Gx^2 + Gy^2)$
Direccion_grad ← $atan2(Gy, Gx)$

% 3. Aplicar supresión no máxima
NMS ← NonMaximumSuppresion(Magnitud_grad, Direccion_grad)

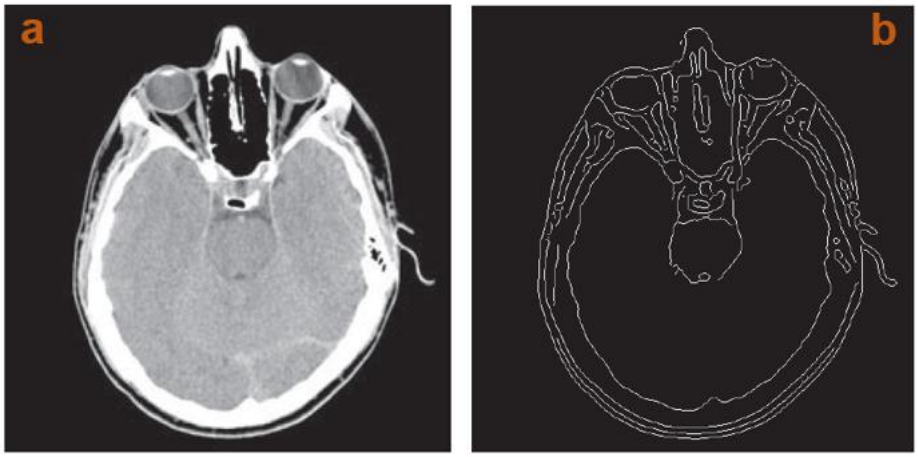
% 4. Establecer borde final
Bordes ← AplicarDobleUmbral $(NMS, \theta_h, \theta_l)$
Borde_Final ← Histéresis(Bordes)

Nota. El parámetro σ debe seleccionarse con precaución, ya que la idea de aplicar el filtro gaussiano es para eliminar el ruido. Un valor muy alto de dicho parámetro difuminaría detalles del borde. Adaptado de (Burger & Burge, 2022)

En la **Figura 43**, se aprecia la aplicación del método de detección de Canny (b) respecto de la imagen original (a) considerando los siguientes parámetros $\sigma = 2$, un kernel de 13 por 13 con umbrales $\theta_l = 0.05$ y $\theta_h = 0.15$.

Figura 43

Aplicación del detector de borde según el método Canny



Nota. La imagen original se ubica a la izquierda (a) y el resultado de aplicar el método de Canny (b). Adaptado de (Gonzalez & Woods, 2021).

El último método de segmentación de esta subsección es la Transformada de Hough, la cual es eficaz para detectar líneas, círculos y elipses siendo sus contornos los que definen espacios estructurales. Según (Gonzalez & Woods, 2021), describe matemáticamente la transformada de Hough mediante la siguiente lógica, la misma que se ilustra en la **Figura 44**:

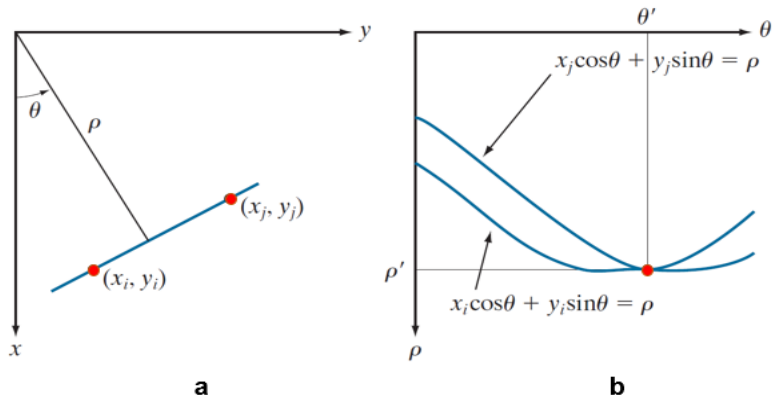
- Se considera dos puntos aleatorios del espacio cartesiano (x_i, y_i) y (x_j, y_j)
- Dado que dos puntos definen una recta, se identifica la ecuación de dicha recta.
- A partir del centro del sistema de coordenadas, se establecen dos parámetros adicionales: (ρ) como la distancia del centro a la recta y (θ) como el ángulo formado por el eje x y la perpendicular a la recta.

Por lo que cualquier punto de dicha recta cumplirá la ecuación (22), siendo ρ, θ constantes:

$$x \cos\theta + y \sin\theta = \rho \qquad (22)$$

Figura 44

Interpretación geométrica de la transformada de Hough

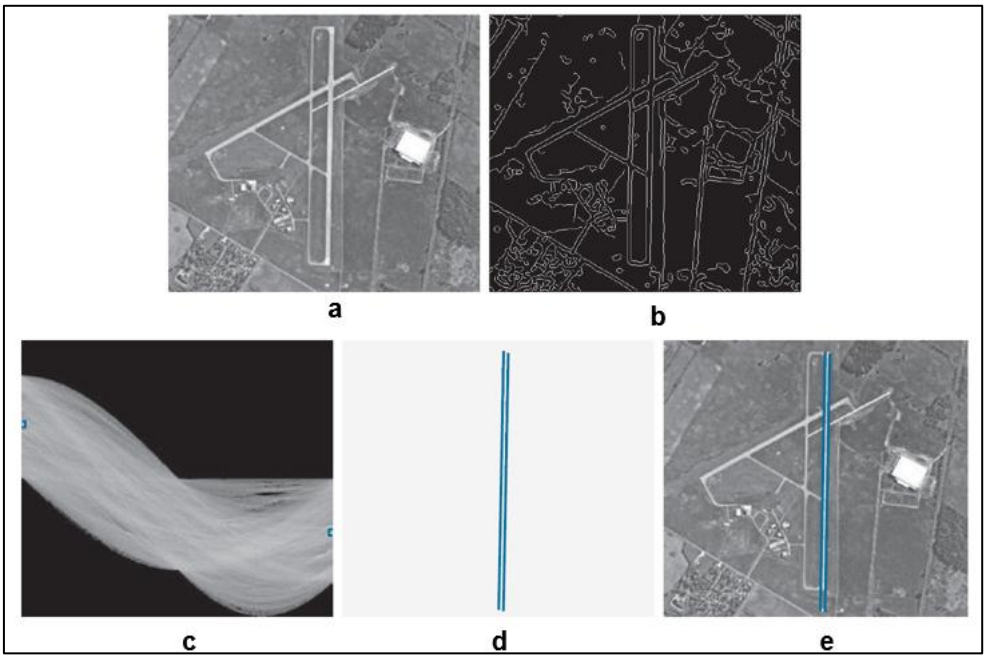


Nota. a) Parametrización de una línea en el plano XY, b) Curva sinusoide en el plano $P\Theta$, siendo la intersección ρ'θ' correspondiente con la línea que pasa a través de los puntos (x_i, y_i) y (x_j, y_j) del plano XY. Adaptado de (Gonzalez & Woods, 2021).

La **Figura 45** ilustra el procedimiento para la aplicación de la transformada de Hough. El proceso inicia considerando una imagen original en escala de grises del aeropuerto (a), seguidamente esta imagen es filtrada para obtener sus bordes (b), luego en el espacio del plano $P\Theta$ se resalta dos puntos de 0° y 360° (c), los cuáles se representan en el espacio cartesiano (d). Por último, las líneas detectadas se superponen en la imagen original dando como resultado una segmentación estructural (e); es decir, se logra resaltar en la imagen original permitiendo identificar o aislar la región de interés.

Figura 45

Detección de líneas con la transformada de Hough



Nota. a) Vista Horizontal de un aeropuerto, b) Detección de Líneas (Filtro de Canny), c) Espacio de parámetros para transformada de Hough, d) Líneas correspondientes a los puntos asociados en la imagen c, e) línea recta impuesta sobre la imagen original. Adaptado de (Gonzalez & Woods, 2021)

Este método de detección de bordes para aislar zonas de interés es fundamental para la extracción de características dado que facilitará la segmentación de las regiones estructurales.

2.4.5 Extracción de características

La última etapa del sistema de visión artificial planteado para esta tesis consiste en la extracción de características la cual es fundamental para identificar en tiempo real diferencias relevantes entre los objetos o clases de objetos. La técnica para aplicar consiste en la detección de patrones que serán los puntos clave (Key-Points) para identificar características distintivas en la imagen. Entre los algoritmos más usuales para la detección de puntos clave tenemos:

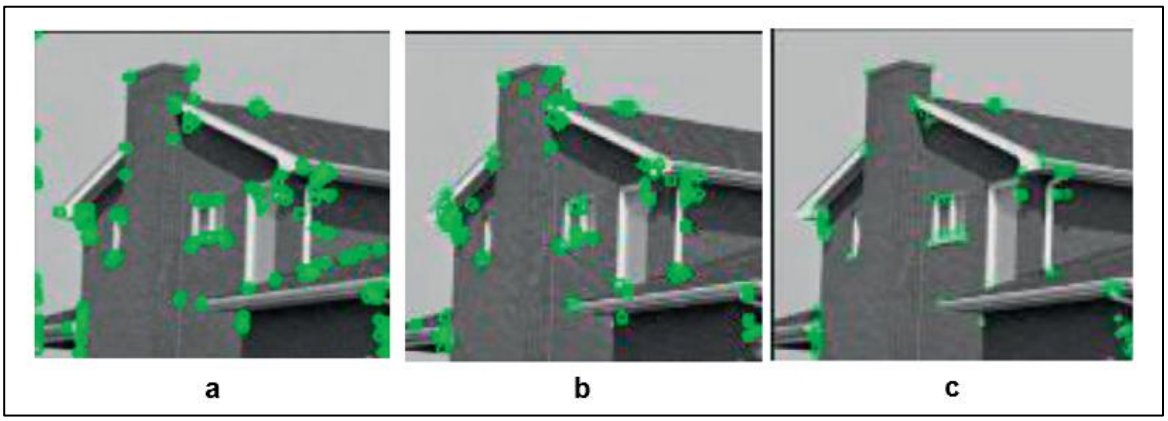
- Detector de esquinas Harris-Stephens. (Harris Corner Detector).
- Transformación de características invariantes a escala (SIFT).

- Detector ORB (Oriented Fast and Rotated Brief).

La aplicación de estos algoritmos de detección de puntos clave (Key-points), se observa en la **Figura 46**, en el mismo orden descrito anteriormente, detector de esquinas Harris, SIFT y ORB.

Figura 46

Detección de características



Nota. Adaptado de (Wang, Li, Wang, Lv, & Yang, 2022)

Según la investigación de (Adel, Elmogy, & Elbakry, 2015), hay una notable diferencia al comparar el performance de los métodos mencionados. En la **Tabla 6**, se muestra el resultado de aplicar los tres diferentes algoritmos mencionados para la detección de características sobre una misma imagen.

Tabla 6

Performance de los métodos de extracción de características

Detector de características	Cantidad de características detectadas	Tiempo en detección (s)	Error medio de posición (pixeles)
Esquinas (Harris)	1115	0.52	2.45
SIFT	989	1.2	2.72
ORB	175	0.06	1.93

Nota. Adaptado de (Adel, Elmogy, & Elbakry, 2015) y (Wang, Li, Wang, Lv, & Yang, 2022)

Luego de evaluar estos parámetros, el algoritmo ORB para detectar características se posiciona como el más rápido y ofrece mayor precisión en la notación de estos puntos clave.

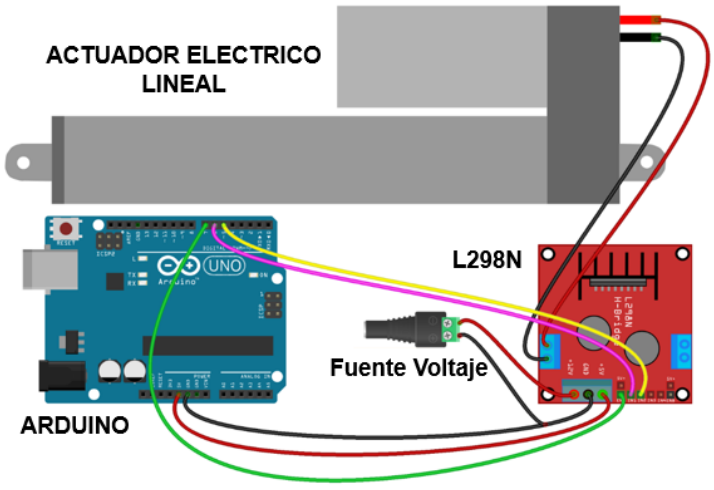
2.5 Control Electrónico

Los dispositivos de esta sección cumplen un rol determinante en el control de actuadores, los cuales están dirigido para el control de la inserción de la aguja y la aspiración del líquido pleural del modelo anatómico. La activación de los actuadores es gestionada por una placa Arduino que actuará como interface entre ROS y los actuadores físicos, sincronizando su funcionamiento con las etapas clínicas de la toracocentesis.

La integración de dichos componentes se describe en la **Figura 47**, donde la placa Arduino coordina la activación cada actuador, con el soporte de un módulo de control tipo Puente H (L298N) para el control el movimiento del vástago del actuador (extensión o retracción).

Figura 47

Esquema eléctrico para control del actuador eléctrico lineal



Nota. Adaptado de (Arduino, n.d.)

En adición, la misma placa Arduino controla la activación del servomotor, el cual controla la apertura y cierre de la válvula de tres vías, lo que impacta directamente en la

acción de succión y drenaje del líquido pleural. Esta integración se revisará con mayor detalle en la sección 1.12.6.4.

2.6 Marco Conceptual

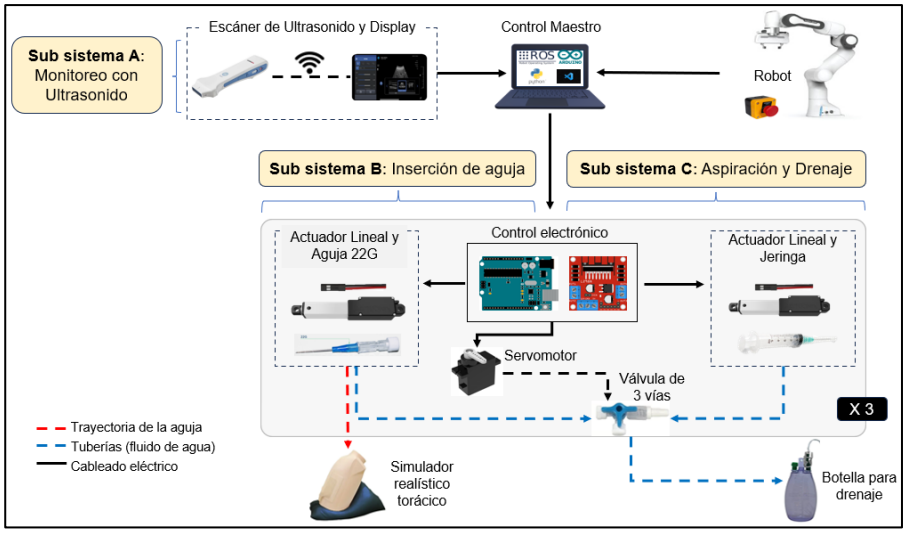
Habiendo definido todos los fundamentos clínicos para la toracocentesis y aspectos técnicos para llevar a cabo esta tesis, la siguiente etapa consiste en integrar los conceptos expuestos a lo largo de este capítulo dentro de un marco conceptual. Para ello, se presentarán diagramas funcionales de la propuesta de sistema; asimismo, para un mejor entendimiento el diagrama general será dividido en subsistemas.

2.6.1 Diagrama completo del sistema autónomo

Este diagrama tiene como objetivo presentar de manera clara y sencilla las interrelaciones de los instrumentos clínicos, tecnológicos y de diseño, consolidando las bases para lograr el sistema autónomo propuesto. La **Figura 48** lo representa gráficamente:

Figura 48

Marco conceptual del sistema autónomo



Nota. Elaboración propia.

Este diagrama del sistema autónomo compuesto por 3 subsistemas:

- A: Monitoreo con escáner de ultrasonido.
- B: Inserción y remoción de aguja.
- C: Aspiración y Drenaje

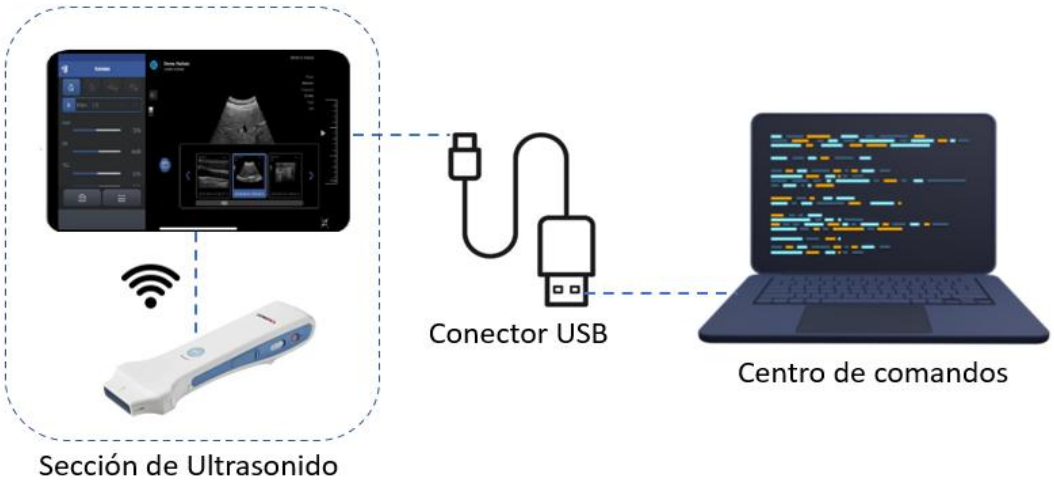
Estos subsistemas están compuestos por componentes electrónicos, instrumentos médicos y un robot, los cuales están conectados mediante conexiones que permiten el intercambio de señales basados en tópicos de visión artificial y robótica para su control dinámico durante el procedimiento. A continuación, se detallará cada subsistema.

2.6.2 Subsistema A: Obtención de lecturas ecográficas

Este subsistema relacionado a la lectura de imágenes de ultrasonido es fundamental, puesto que depende de su operatividad obtener los datos de entrada en tiempo real, que sería la representación estructural interna del simulador anatómico. La representación de su conectividad se observa en la **Figura 49**, el cual está compuesto por equipo médico y como se transmite al centro de comandos.

Figura 49

Subsistema A: Esquema para transmisión de lecturas US



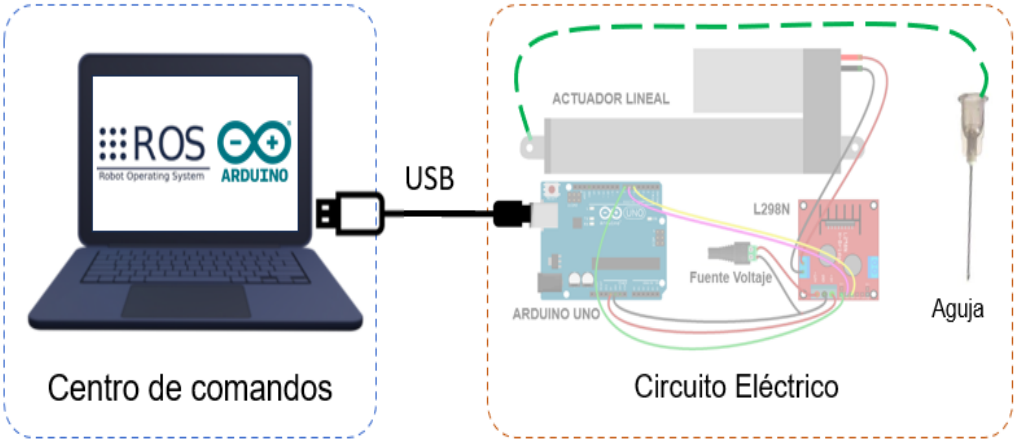
Nota. Elaboración propia

2.6.3 Subsistema B: Inserción de Aguja

Este subsistema está compuesto por un circuito eléctrico que tiene un actuador como su elemento principal. En ese caso el vástago irá conectado a una jeringa mediante un acople, tal como se visualiza en la **Figura 50**, para que replique la acción de punción (extensión del vástago) y extracción (retracción del vástago).

Figura 50

Subsistema B: Esquema de conexión para inserción de aguja



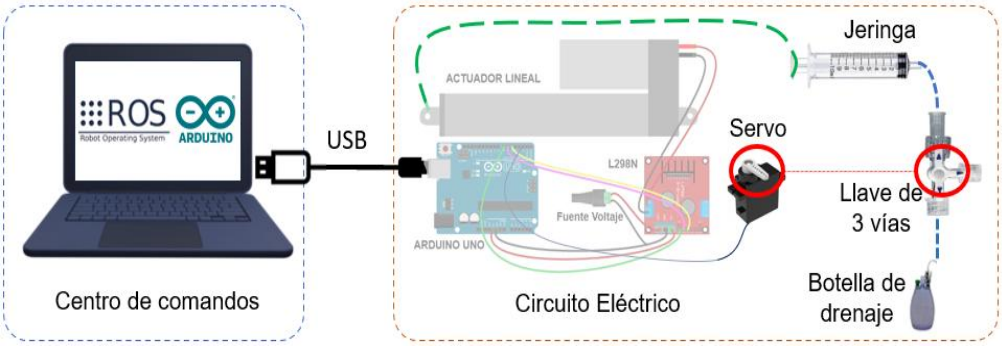
Nota. Adaptado de Circuits-DIY (Fuller, 2023)

2.6.4 Subsistema C: Aspiración y Drenaje

Este último subsistema ejecuta actividad al igual que el anterior; sin embargo, en este caso es más complejo puesto que el vástago de un segundo actuador esta unido a una jeringa mediante un acople para replicar la acción de succionar (retracción del vástago) y expulsar el agua (expansión del vástago). En tanto, mientras que se llevan a cabo dichas acciones, estas son sincronizadas con la acción de un servo para que al aperturar o cerrar la llave de tres vías canalice correctamente el agua extraída y expulsada. Esta condición de funcionamiento se aprecia en la **Figura 51**

Figura 51

Subsistema C: Esquema para drenaje del líquido pleural



Nota. Adaptado de Circuits-DIY (Fuller, 2023)

CAPÍTULO III

DESARROLLO DEL TRABAJO DE INVESTIGACIÓN

En este capítulo se desplegará el diseño que agrupará a todos los componentes de los capítulos previos, siguiendo la lógica descrita en el marco conceptual. Una vez establecido este diseño, se acoplará el software correspondiente (algoritmo en el control maestro) para controlar la activación de los subsistemas “b” y “c”. Esta lógica comprende todas las etapas de la toracocentesis; es decir desde el monitoreo con la probeta de ultrasonido, la inserción de la aguja hasta finalizar con el aspirado y retiro de la aguja. En este capítulo se abordará la estrategia técnica, el desarrollo de hardware y software, así como el preprocesamiento de las imágenes recibidas y su posterior adaptación para una óptima visualización y sea comprendido por la lógica.

3.1 Metodología para la tesis

Para lograr el objetivo de esta tesis, el cual es implementar un sistema autónomo capaz de replicar las operaciones descritas anteriormente en el mapa conceptual, se presenta la siguiente estrategia basado en dos implementaciones, de hardware y software.

Respecto al **desarrollo de hardware**, se profundizará en la elaboración del módulo robótico personalizado, así como la integración de los instrumentos médicos y los componentes electrónicos. Se apunta a lograr una integración compacta; es decir, agrupados en un solo mecanismo que sea funcional. Este enfoque permite obtener las siguientes ventajas operativas apuntando a la eficiencia y precisión:

- Interacción sincronizada entre los subsistemas para contribuir a la precisión durante la punción.
- Optimización significativamente del espacio requerido al crear un módulo robótico personalizado, siendo una característica vital en aplicaciones médicas donde el entorno de aplicación no es muy amplio.

- Minimización de la complejidad del sistema al integrar efectivamente los componentes.

La secuencia de desarrollo del hardware se representa a través de la **Figura 52**, en la que se destaca el paso a paso desde la creación hasta la integración al efector final del robot.

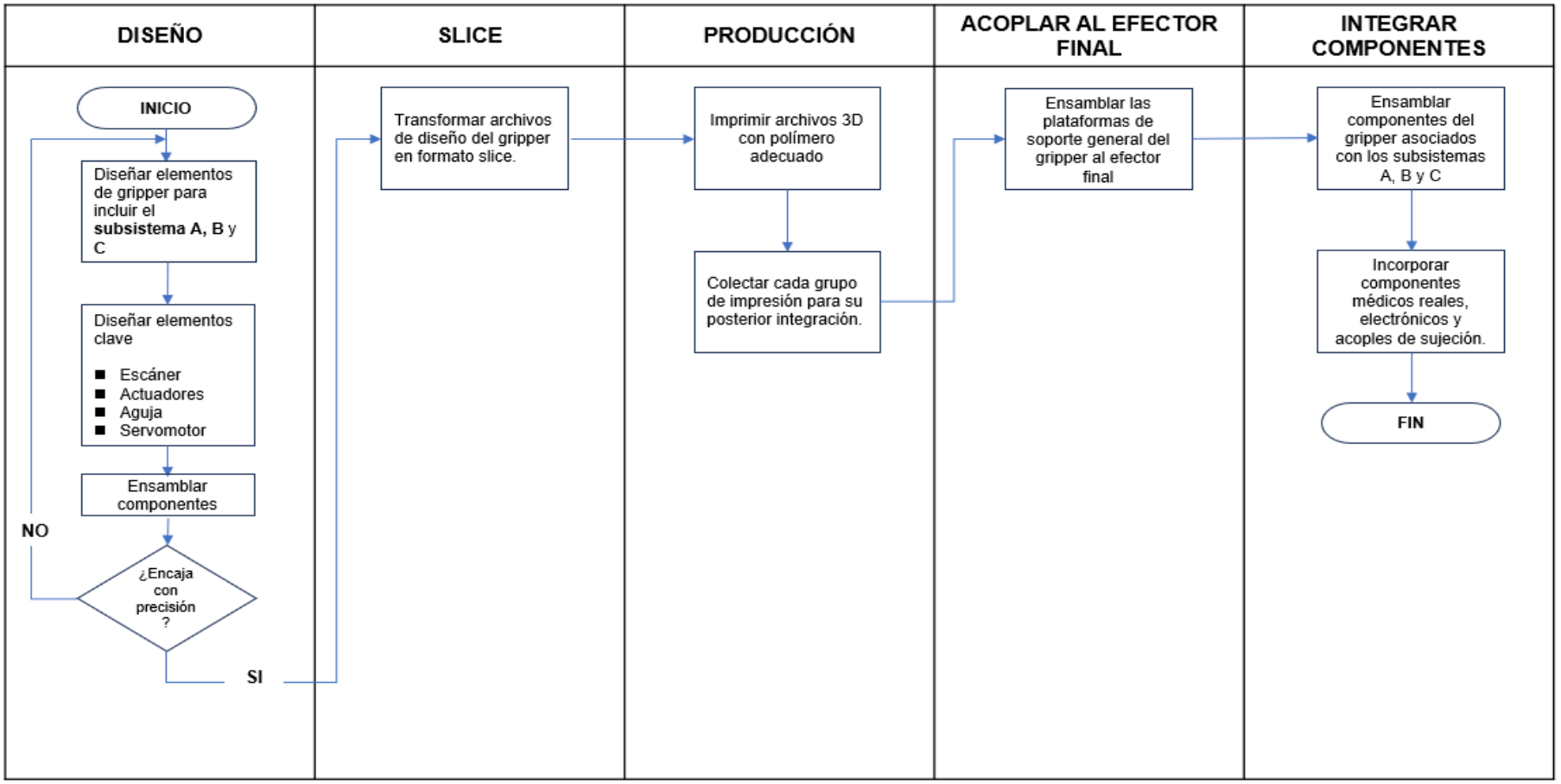
En tanto, la **Figura 53** detalla el flujo de trabajo paso a paso la adaptación del sistema robótico al proceso quirúrgico de la toracocentesis abordando la primera etapa del control del robot, pasando por etapas de lecturas de imágenes US para su posterior aplicación de los actuadores lineales para lograr punción, aspiración y extracción del líquido.

Finalmente, la **Figura 54** manifiesta el contexto técnico y estructural del sistema para identificar la organización de los entornos de desarrollo y su interacción computacional bajo el enfoque ROS destacando despliegue de nodos, tópicos y metapaquete MoveIt. En la parte superior se describe bloques funcionales del sistema desde que se activa mediante ROS, haciendo uso de MoveIt para planificar y ejecutar trayectorias, seguido de la ejecución de un programa para detección de la aguja por medio de imágenes de ultrasonido mediante algoritmo de python.

La sección media constituye los entornos virtuales y lenguajes para ejecución y despliegue de algoritmos para cada bloque funcional. Finalmente, la sección inferior presenta la arquitectura en ROS, donde se hace la activación computacional con un archivo launch, el cuál gobierna la ejecución del control del robot, así como los nodos principales que albergan los archivos para la lógica de visión artificial enfocado en el procesamiento de las imágenes de ultrasonido, así como para el control de los actuadores lineales.

Figura 52

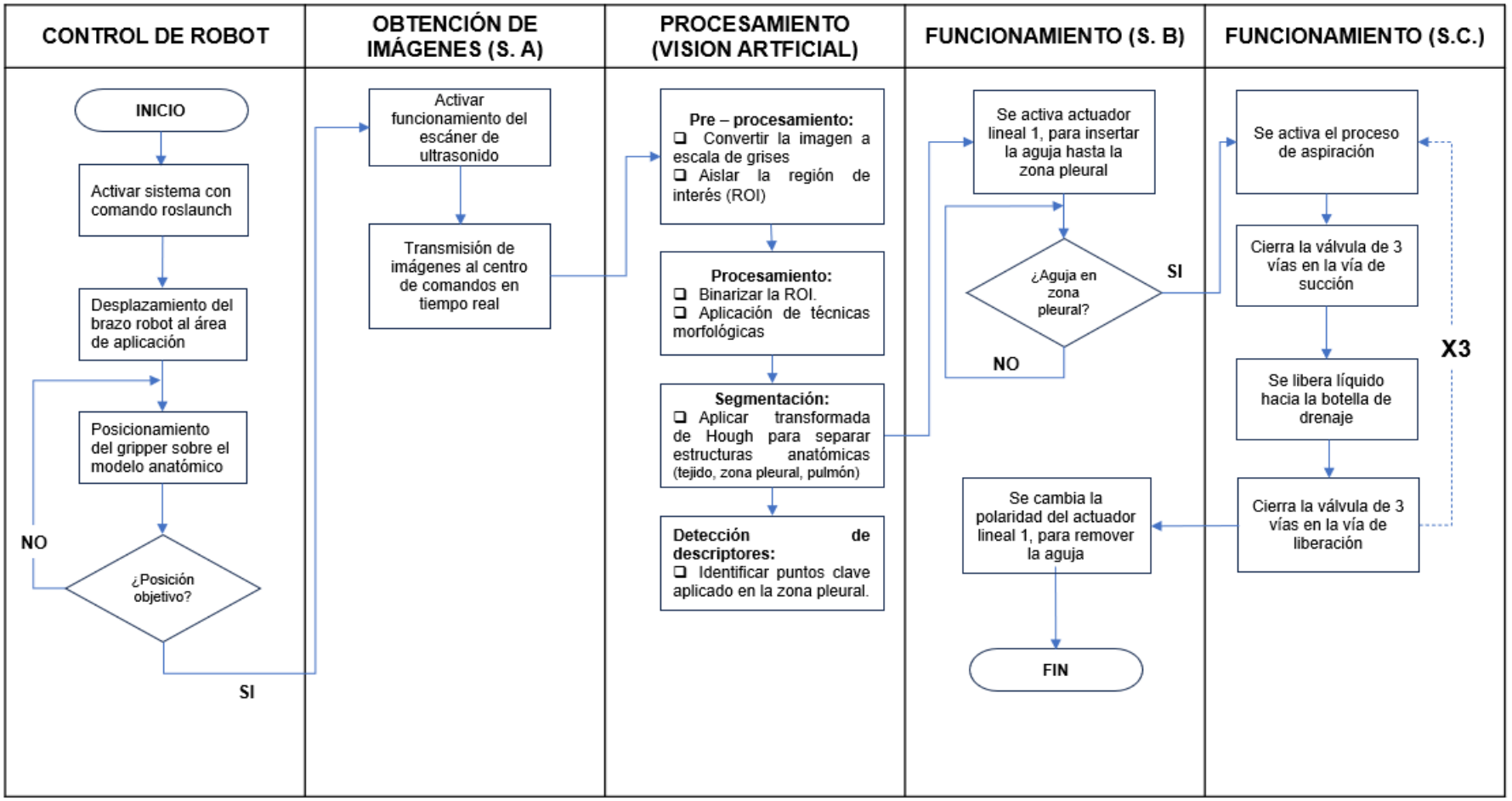
Desarrollo e Implementación de Hardware



Nota. Elaboración propia

Figura 53

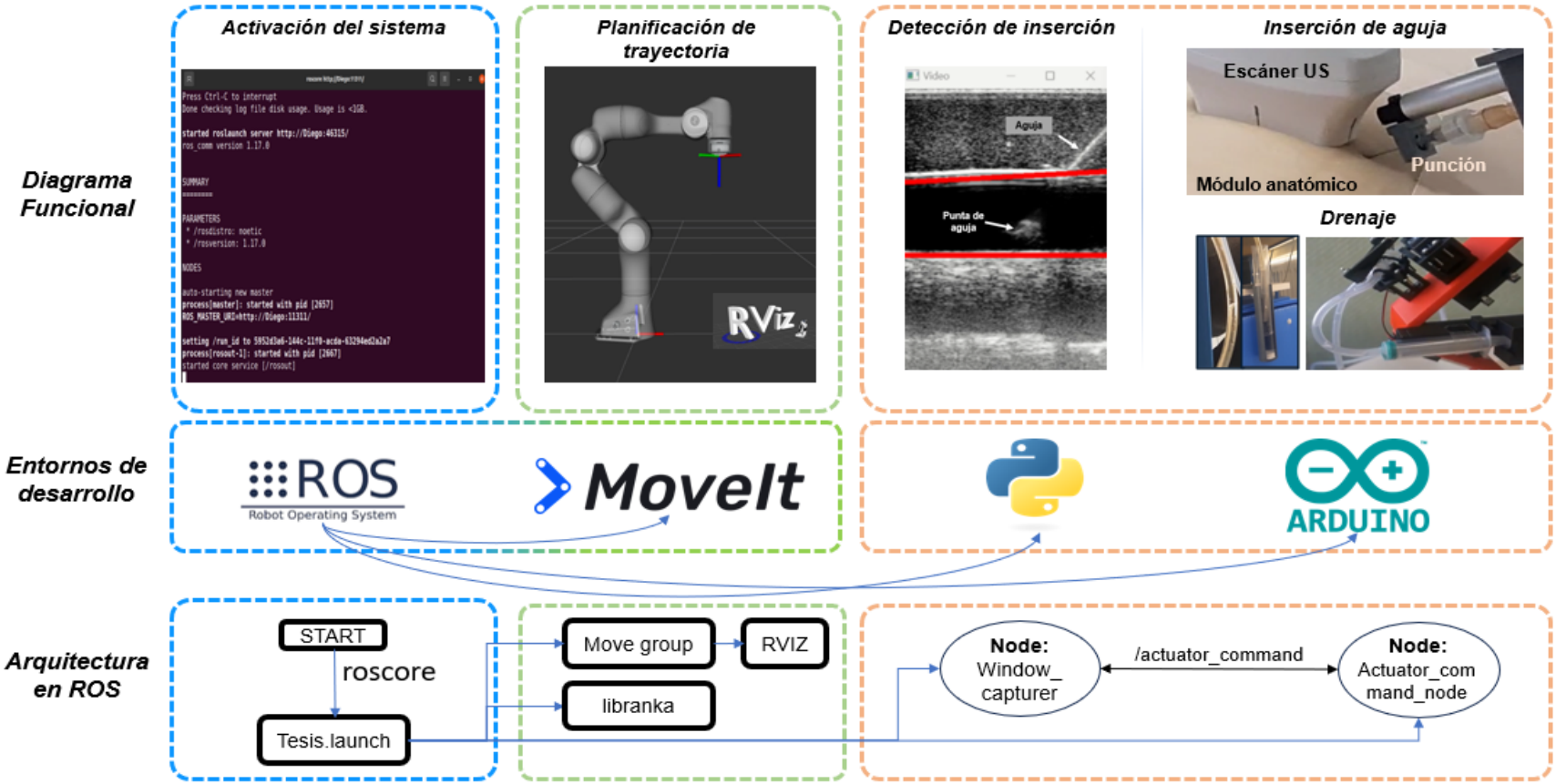
Flujo de trabajo del sistema robótico para toracocentesis



Nota. **S.A:** Subsistema A, **S.B:** Subsistema B, **S.C:** Subsistema C. Elaboración propia

Figura 54

Arquitectura funcional y de software del sistema autónomo



Nota. Elaboración propia

3.2 Desarrollo e Implementación de Hardware

Esta sección dedicada al desarrollo y montaje del hardware esta subdividida en tres apartados: Presentación del equipamiento médico , componentes electrónicos. una vez identificados todos los elementos que intervendrán en el proceso de la toracocentesis, se procederá a detallar en el tercer apartado, el acoplamiento del módulo robótico personalizado al efector final del robot. Este último apartado se explicará en detalle de acuerdo con la secuencia descrita en la **Figura 52**.

3.2.1 Equipamiento médico

Los instrumentos médicos son fundamentales para replicar el proceso clínico de la toracocentesis; por ello, se requiere el uso de un escáner de ultrasonido, modelo anatómico para simular la cavidad torácica de manera realista y elementos involucrados durante la punción y para el drenaje del líquido.

- **Escáner de ultrasonido**

El escáner de ultrasonido empleado en esta tesis es el dispositivo portable Sonon 300L de transductor lineal de peso ligero el cuál es ideal para el proceso de la toracocentesis. A continuación, se describe las características del escáner seleccionado.

Peso: 370g

Frecuencias disponibles: 5 MHz / 7.5 MHz / 10 MHz

Modo de imagen: Modo de Brillo.

Dimensiones: 78 x 229 x 38 mm

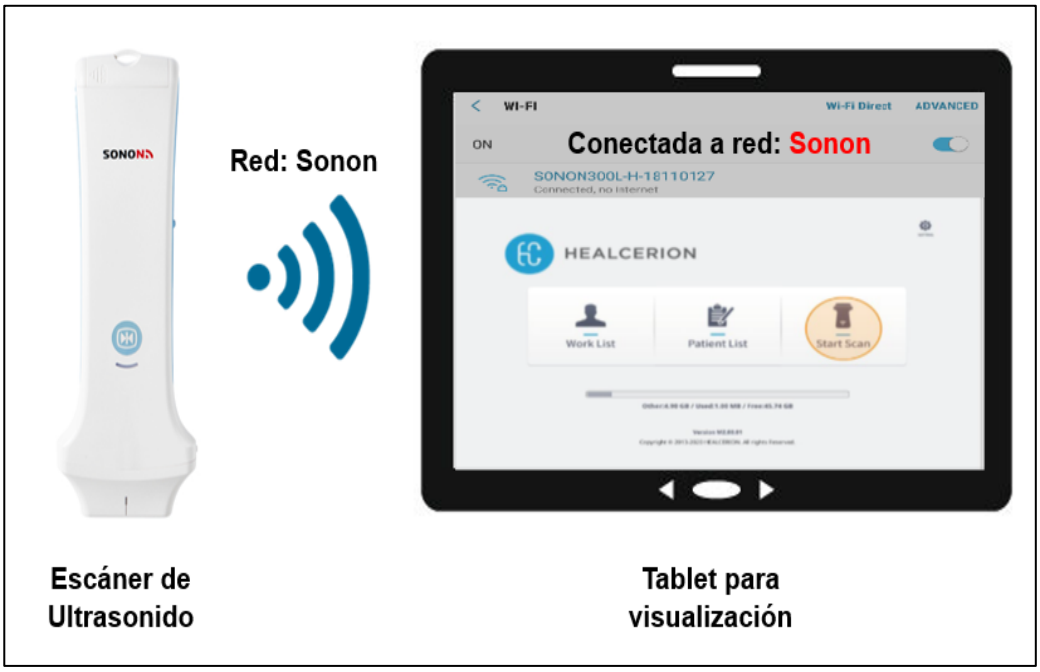
Aplicación móvil: Healcerion o Sonon app

Compatibilidad de aplicación: Android & iOS

Conectividad: Wi-Fi (2.4 y 5 GHz)

Figura 55

Configuración de implementación del equipo de ultrasonido



Nota. Adaptado de (Healcerion, s.f.)

Este escáner es considerado muy eficiente debido a las ventajas que presenta, por ejemplo:

- Reproduce las imágenes de ultrasonido mediante conectividad Wi-Fi a través de una tablet que tenga instalada la aplicación Healcerion, como se observa en la **Figura 55** otorgando flexibilidad al intervencionista para su manipulación durante el proceso clínico.
- Su transductor lineal, a diferencia de los otros transductores convexos, definen imágenes con visualización recta; es decir, sin alteración siendo beneficioso para el tratamiento computacional.
- Por último, se puede manipular parámetros que ajusten suavidad y brillo para mejorar visualización y calidad de la imagen, como se representa en la **Figura 56**.

Figura 56

Parámetros para optimizar resolución de imagen

Nota. En la sección ubicada en el lado izquierdo se enmarca en rojo los parámetros de ajuste; mientras que, en el resto de la pantalla de la tablet, se visualiza la imagen de ultrasonido. Elaboración Propia.

Para complementar el uso del escáner de ultrasonido, se utiliza una tablet de marca Lenovo y modelo TB-8505F, misma que se visualiza en la **Figura 56**.

- **Modelo de simulador anatómico**

El modelo anatómico para simular la aplicación del proceso de la toracocentesis el cual ofrece una sensación y resistencia realista a la inserción de una aguja. Este maniquí es un conjunto completo que comprende dos módulos para realizar la intervención quirúrgica, como se visualiza en la **Figura 57**.

Figura 57

Maniquí torácico para intervención quirúrgica con módulo en región axilar

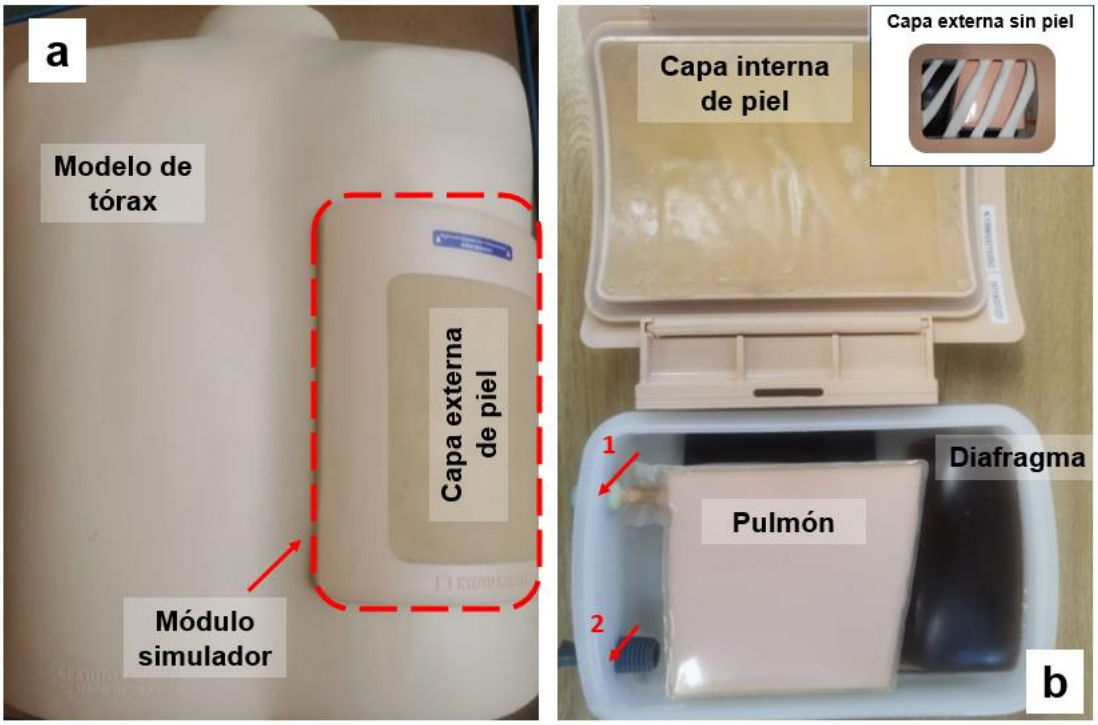
Nota. Tomado de (Kyoto Kagaku, s.f.)

Este maniquí cuenta con dos módulos para realizar la toracocentesis. El primero se ubica en la región axilar (vista lateral), como se detalla en la **Figura 57**, mientras que el segundo, se ubica en la región escapular posterior (espalda), **Figura 58**. Ambos módulos son absolutamente idénticos, compuestos por una cubierta exterior y su interior del módulo como se visualiza en la **Figura 58**.

Esta **Figura 58** es de alta importancia puesto que, no solo se presenta el modelo anatómico, sino también se distingue el acceso para el llenado de líquido, que en este caso será completado por agua para la representación de la efusión pleural. De acuerdo con el manual de fabricante (Healcerion, s.f.), se indica que este módulo puede almacenar líquido hasta 200 mL con el pulmón completamente lleno de aire y en un rango de 370 a 380 mL cuando el pulmón se encuentra completamente desinflado. Adicional a ello, la capa externa e interna de piel replican una textura y sensación realista de las capas de tejido que tenemos en el cuerpo humano.

Figura 58

Descripción de módulo para aplicación de toracocentesis



Nota. Maniquí completo de tórax (a), Despiece del módulo para la toracocentesis (b). En la imagen (b) se resalta dos orificios; orificio de acceso de aire para el pulmón (1) y tapón de acceso de líquido (2). Elaboración propia.

- **<u>Elementos sanitarios para la inserción percutánea.</u>**

  Los elementos que participan en la inserción son:

Aguja de medida 22G o 23G, medidas compatibles según el fabricante del módulo de toracocentesis; de tal manera que no deforme la capa epitelial durante la inserción o remoción de la aguja. En la

- **Figura 59**, se observa la aguja 22G de 0.7 mm de diámetro por 40 mm de longitud. Asimismo, también se considera una jeringa de capacidad de 10 mL, la cual creará el efecto de succión para la aspiración y también el efecto expulsión para la liberación del líquido, siendo dicho efecto producido por su pistón.

Figura 59

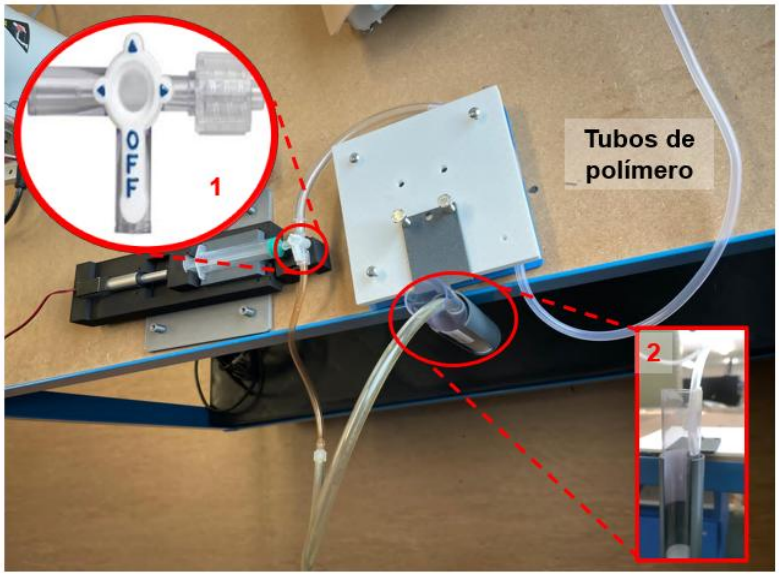
Aguja hipodérmica

Nota. Elaboración propia

La llave de triple vía, dispositivo que controla la dirección de canalización del líquido a través de tuberías de polímero biomédico; es decir, aspiración del líquido desde el módulo y el drenaje a un recipiente colector; tal como se representa en la **Figura 60**.

Figura 60

Llave de 3 vías y tubería de polímero biomédico



Nota. La llave de triple vía (1) que gestiona la aspiración y drenaje del líquido. El receptor de líquido que se muestra en la imagen (2). Elaboración propia

Los componentes electrónicos de control son fundamentales ya que son los encargados de enviar señales a los actuadores lineales controlando el movimiento de

avance y reversa de su extensión. Estos componentes han sido ubicados en la segunda plataforma de la zona superior del módulo robótico.

3.2.2 Componentes electrónicos

Este aparatado aborda los principales componentes electrónicos para el control automatizado del proceso de la toracocentesis los cuáles fueron definidos en el esquema descrito en **Figura 47**.

Los componentes electrónicos que se requieren para el desarrollo de esta tesis se describen en la **Tabla 7**, siendo fundamental justificar la función que desempeñarán cada uno.

Tabla 7

Lista de componentes electrónicos

CANTIDAD	COMPONENTE	FUNCION
1	Tarjeta Arduino UNO R3	Gestionar la acción de los actuadores
1	Puente H (Módulo L298N)	Gestión del sentido de corriente eléctrico para el movimiento lineal del vástago del actuador lineal
1	Servomotor	Control de posición de la válvula de la llave de triple vía
1	Protoboard	Soporte para conexiones de cablería
2	Actuadores Lineales	El primero, para replicar la punción y remoción de la aguja. El segundo para succionar y liberar el líquido del módulo.

Nota. Elaboración propia

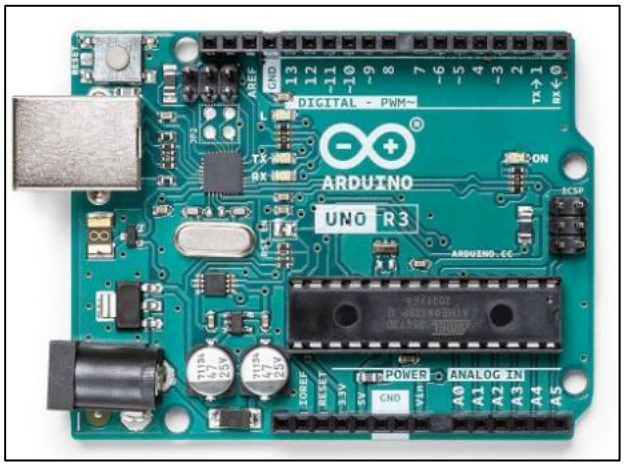
Por lo que a continuación se describe cada componente involucrado en el sistema autónomo.

- **Arduino UNO R3**

En esta tesis se usará la tarjeta electrónica Arduino UNO R3 el cual viene equipado con un procesador ATmega328P (Arduino, 2025). Este procesador puede llegar alcanzar 20MHz durante el procesamiento siendo altamente efectivo para controlar una baja latencia entre la transmisión de señales.

Figura 61

Tarjeta electrónica Arduino UNO

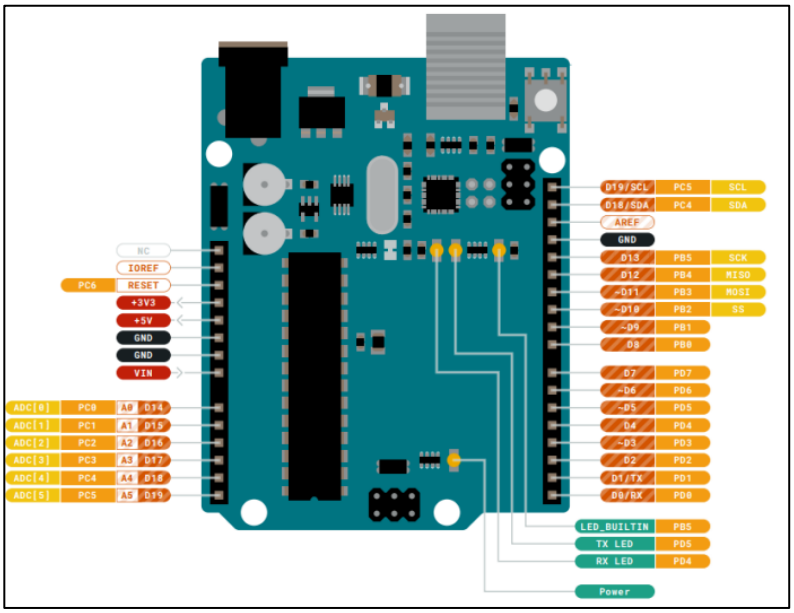


Nota. Tomado de (Arduino, n.d.)

Esta tarjeta electrónica viene equipada con el microcontrolador ATMega328P y elementos adicionales para gestionar las señales eléctricas enfocadas en el funcionamiento de los actuadores lineales. Esta plataforma dispone de pines de entrada y salida, como se ilustra en la **Figura 62**, que posteriormente serán configurados como entrada o salida de señales.

Figura 62

Diagrama de conectores de Arduino UNO



Nota. Tomado de (Arduino, 2025)

A continuación, se presenta sus características.

Procesador: ATMega328P

Voltaje de entrada: 5V (USB)

Voltaje de Operación: 5V

Voltaje de regulación: 3.3V

Corriente máxima: 50mA

Pines Digitales: 14 (Entradas/Salidas) | 6 con salida PWM

Pines Analógicos: 6

Peso: 25 g

- **Módulo de Control de Motor**

Este módulo viene equipado con el controlador dual L298N que facilita el cambio de polaridad en la alimentación para los actuadores lineales, de tal manera que su vástago se extienda y comprima. A continuación, se presentan sus características y su representación gráfica en la **Figura 63**.

Marca: 4tronix

Voltaje de entrada máxima: 46V

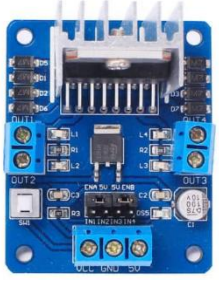
Voltaje de alimentación: 5V

Corriente continua máxima: 2A por canal

Salidas de carga: 2 unidades

Figura 63

Módulo de control de motor



Nota. Elaboración propia

- **Servomotor**

Este componente facilita la conmutación de la dirección del fluido aspirado al controlar la válvula de tres vías con precisión. Se aprovecha el actuador rotativo para alinearlo con un mecanismo con la válvula de tal manera que se aperture o cierre las vías para la extracción y liberación del líquido. A continuación, se destacan algunas características:

Voltaje de operación: 4.8 ~ 7.2V

Torque de bloqueo : 13.5Kg*cm

Rango para ángulo de operación: 0° – 270°

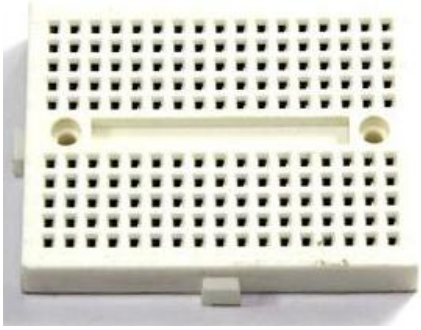
Señal de control : PWM

- **Protoboard**

Este placa permite combinar múltiples conexiones, siendo esencial para asegurar la integración eficiente y segura de la conectividad eléctrica entre los componentes.

Figura 64

Protoboard o plataforma de conexiones



Nota. Elaboración propia

- **Actuadores Lineales**

Los actuadores contienen un vástago del cual se aprovechará su compresión y extensión para los siguientes fines, mismo que se visualizan en la

Figura 65:

- Adaptarse a la dinámica de la punción de la aguja (inserción y remoción).

- Succión y expulsión de líquido una vez la punta de la aguja se encuentre en la zona pleural.

Para complementar la información de la utilidad de los actuadores lineales se tiene la **Tabla 8**, que muestra las etapas del tubo de extensión del actuador correspondiente y su correspondencia con las etapas de la toracocentesis.

Tabla 8

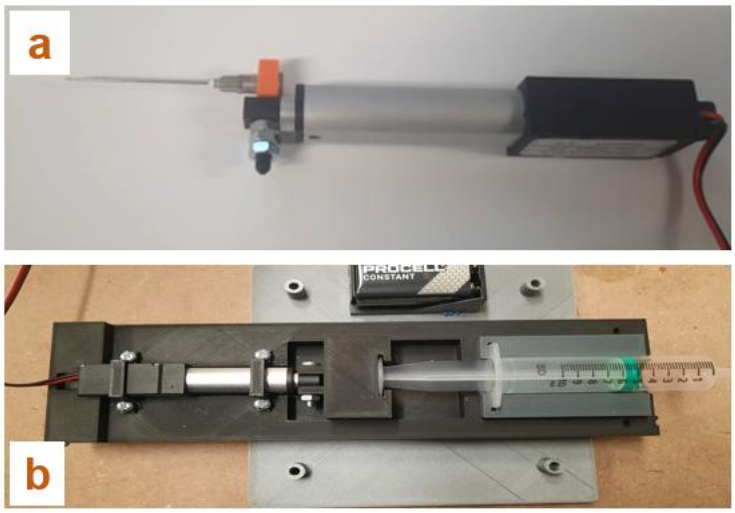
Actuadores en etapas de Toracocentesis

POSICIÓN	ACTUADOR LINEAL 1	ACTUADOR LINEAL 2
Avance	Inserción de aguja	Aspiración de Líquido
Reversa	Remoción de aguja	Liberación hacia el colector

Nota. Elaboración propia

Figura 65

Funciones de los actuadores lineales



Nota. Actuador lineal para punción de aguja (a), Actuador lineal para desplazamiento del pistón de la jeringa. Elaboración propia.

3.2.3 Robot Franka Emika Panda

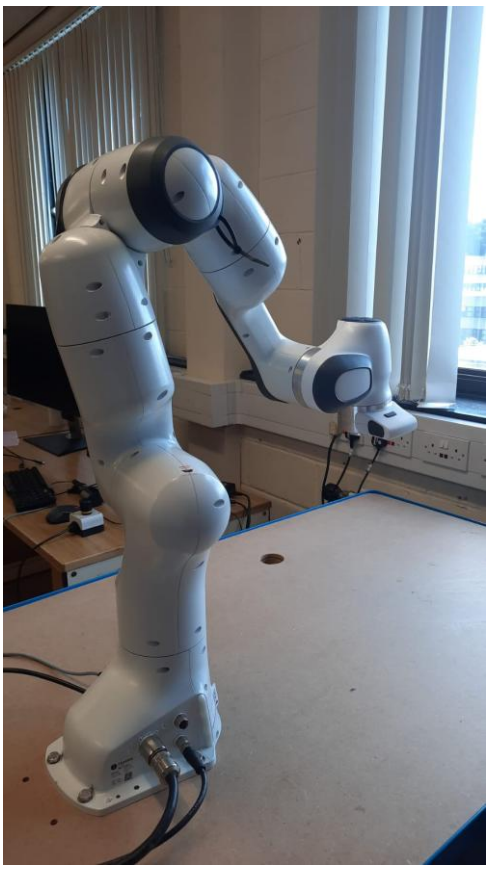
Este brazo robótico de la marca Franka Emika es uno de los más utilizados en el contexto de procedimientos quirúrgicos, el cual está equipado con 7 articulaciones (grados

de libertad), lo que brinda una versatilidad adicional respecto de los brazos robóticos tradicionales. Es decir, que el robot es capaz de planear trayectorias complejas para optimizar manipulaciones precisas. (Haddadin, 2024). Para esta tesis, este robot representa una estructura compacta y sólida, el cual se constituye como el núcleo del sistema tal como se visualiza en la **Figura 66**.

Adicionalmente, para llevar a cabo un control cinemático y dinámico de este brazo se hace uso de algoritmos sofisticados que gestionen apropiadamente el movimiento para asegurar la estabilidad y exactitud de este. Debido a la alta complejidad en los cálculos que infiere este brazo robótico, el control es dirigido mediante un algoritmo, esencialmente para planear y ejecutar su trayectoria. (Feng, Su, & Qian, 2024).

Figura 66

Robot Franka Emika Panda



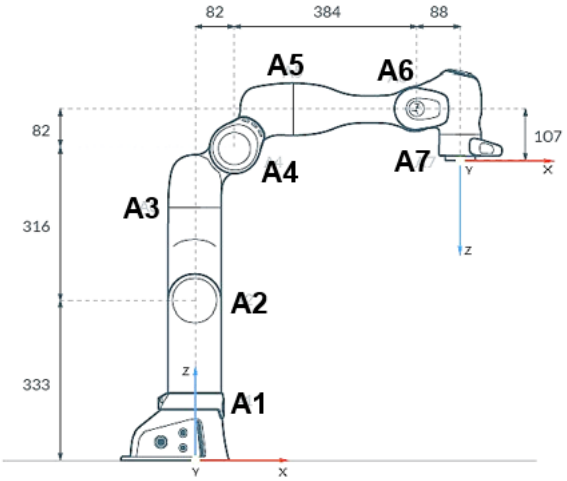
Nota. Elaboración propia

En la **Figura 67**, se muestra en detalle las articulaciones del robot Panda sus medidas en mm, así como también una tabla informativa de sus respectivos parámetros basados en el enfoque de Denavit-Hartenberg (D-H).

Por último, este robot basa su funcionamiento de acuerdo con los principios establecidos en las normas ISO 10218 e ISO 13849, los cuáles establecen lineamientos de seguridad referidos al diseño, operación de los robots industriales y sistemas de control relacionados a la seguridad. De esta manera, estas directrices apuntan a lograr un entorno de trabajo seguro (Franka Emika, n.d.).

Figura 67

Composición mecánica del robot



i	a_i	α_i	d_i	θ_i
1	0	0	0.333	θ_1
2	0	$-\pi/2$	0	θ_2
3	0	$\pi/2$	0.316	θ_3
4	0.82	$\pi/2$	0	θ_4
5	0.82	$-\pi/2$	0.384	θ_5
6	0	$\pi/2$	0	θ_6
7	0.88	$\pi/2$	0	θ_7
8	0	0	0.107	0

Nota. En la imagen izquierda se observa los links y articulaciones del robot con sus respectivas dimensiones en mm, mientras que en la derecha se visualiza la tabla de parámetros del robot con unidades de medida en metros y radianes. Adaptado de (Franka Emika GmbH, 2021; Gaz, Cognetti, Oliva, Robuffo Giordano, & De Luca, 2019)

Finalmente, habiendo detallado las características técnicas y operativas para el robot Franka Emika Panda, resulta adecuado para abordar el proceso de la toracocentesis.

3.2.4 Elaboración e Integración física de los componentes al robot

En esta sección se abordará el desarrollo del módulo robótico que irá acoplado al efector final de robot Franka Emika Panda. Este módulo ha sido elaborado con un enfoque personalizado, de tal manera que integre los componentes de manera eficiente y replique

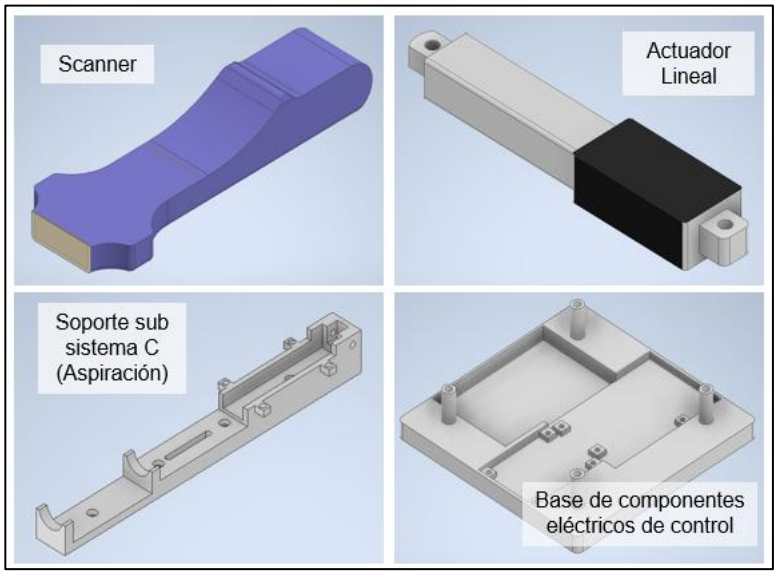
con precisión el proceso de la toracocentesis teniendo en cuenta la sincronización que debe existir entre el equipo médico y los actuadores. La fabricación de los componentes del módulo robótico tiene un rol clave dado que la ubicación de estos hará que este sistema sea preciso y eficiente. Esta subsección se abordará según las etapas descritas para desarrollo e implementación de Hardware, **Figura 52**. A continuación, se abordará cada fase de la creación en detalle de los componentes hasta lograr una integración completa de las piezas fabricadas y los componentes electrónicos e instrumentos médicos.

- **DISEÑO**

El módulo robótico es diseñado en Autodesk Inventor para compactar a los instrumentos médicos y a los componentes electrónicos tales como los actuadores lineales y servomotor. En la **Figura 68**, se muestran algunos componentes entre los que se encuentra los replicados como el escáner, el actuador lineal y algunas piezas para la creación del mecanismo.

Figura 68

Diseño de componentes en Autodesk Inventor



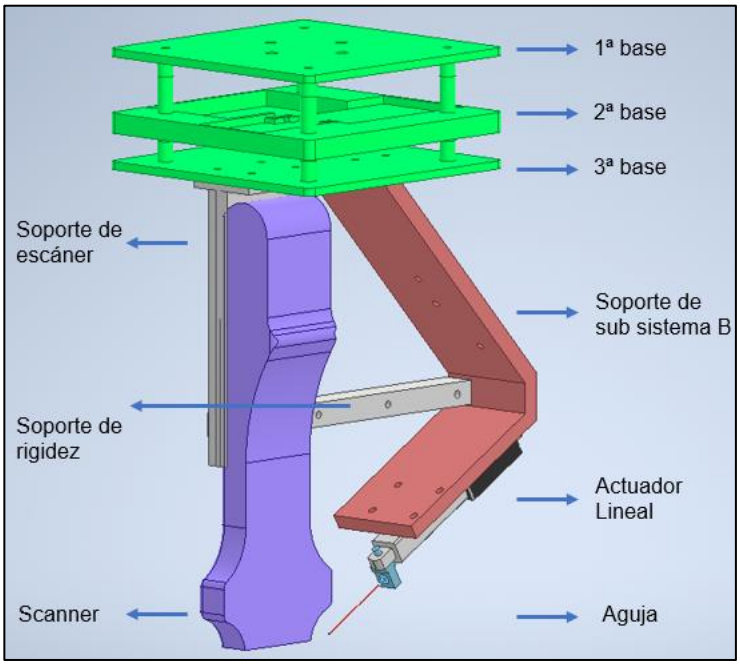
Nota. Elaboración propia

Una vez todos los componentes hayan sido diseñados, según las especificaciones detalladas en el **ANEXO A**, se procederá al ensamble tal como se visualiza en la **Figura 69**

y **Figura 70** desplegando el módulo robótico en su vista frontal y posterior, con todos sus componentes ensamblados. El propósito de integrar todos los elementos es verificar que las piezas creadas encajen adecuadamente con las dimensiones reales de los elementos electrónicos e instrumentos médicos, asegurando que no haya piezas sobredimensionadas del módulo robótico. Esta etapa es vital para los componentes operen de manera síncrona garantizando la precisión durante la intervención clínica

Figura 69

Módulo robótico - Vista Frontal



Nota. Elaboración propia

En la **Figura 69**, Vista Frontal del módulo robótico, se aprecia el mecanismo que sostiene al escáner (color gris) y al subsistema B, mecanismo para inserción de la aguja. En la sección superior del módulo mencionado, se visualiza 3 plataformas de color verde los cuáles tienen la siguiente utilidad:

- La primera para sostener el módulo al efector final del robot usando tornillos de tamaño M6 X 12.
- La segunda plataforma será para ubicar los componentes electrónicos de control tales como el puente H (L298N) y la tarjeta de Arduino UNO.

- La última plataforma conecta las plataformas mencionadas con los soportes de los subsistemas A, B y C.

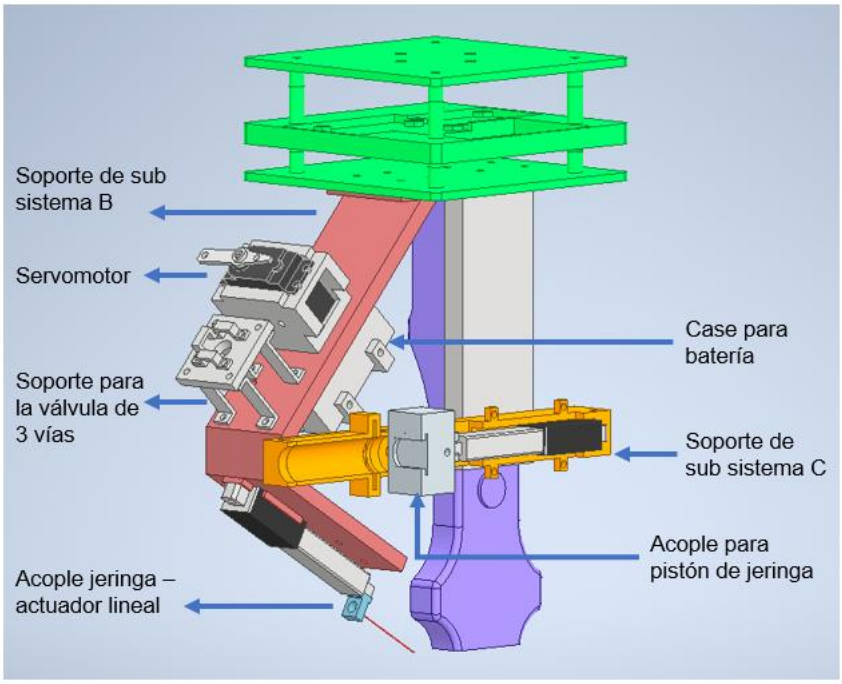
También se ha incorporado un elemento llamado “Soporte de rigidez” (color gris), el cual cumple dos funciones esenciales:

- Evitar una deflexión entre la posición del soporte del escáner y del soporte del subsistema B.
- Ser la base sólida en la que el soporte del subsistema C se ensamblará. Esta función será visible en la **Figura 70**.

Mientras que en la **Figura 70**, se presenta el lado posterior del mecanismo detallando los componentes que lo conforman.

Figura 70

Módulo robótico - Vista Posterior



Nota. Elaboración propia

En la **Figura 70** se observa el mecanismo que soporte al subsistema C (color naranja), mecanismo para extracción y liberación del líquido. En este subsistema, el vástago del actuador lineal está conectado con un acople que tiene un orificio para recibir al pistón

de la jeringa. También se observa el soporte del servomotor montado sobre el soporte del subsistema B.

- **SLICE (Formación de los componentes por capas**

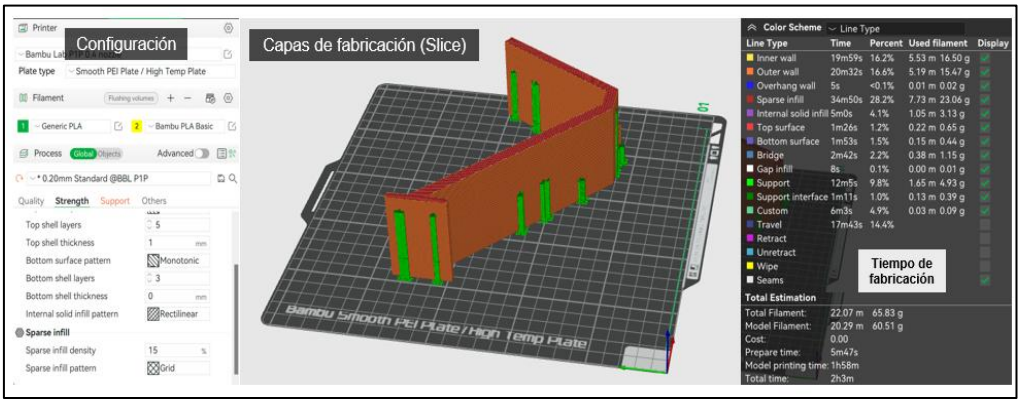
Seguidamente, una vez que el ensamble demuestra que todas las piezas encajan y no existen medidas sobredimensionadas, se procede a preparar cada elemento para ser fabricado. Esta preparación implica que el archivo de diseño debe ser transformado a un formato que contenga instrucciones para la fabricación capa por capa, entre los principales aspectos tenemos: conducción del cabezal de la impresora, temperatura del extrusor, velocidad de impresión, densidad de aplicación del material, entre otros detalles.

La **Figura 71** ofrece detalles del proceso de conversión del modelo 3D a un diseño laminado mediante el uso de tres secciones:

- Configuración: Permite establecer características del laminado
- Espacio Visual: Muestra en detalle el laminado de las capas establecidas
- Esquema de capas: Estimación de cantidad de material y tiempo para la fabricación.

Figura 71

Diseño de pieza en capas (Slice)



Nota. Elaboración propia.

- **PRODUCCIÓN - Fabricación de componentes**

Una vez la pieza ha sido sometida a una configuración por capas (Slice) se procede a guardar cada pieza en un archivo de extensión “gcode” que contiene toda la especificidad que necesita la impresora Bambu Lab P1P.

La elaboración física de los componentes del módulo robótico ha sido materializada a través una impresora 3D de las siguientes características:

- Marca: Bambu Lab
- Modelo: P1P
- Filamento usado: PLA
- Software: Bambu Studio

Finalmente, la **Figura 72** muestra la etapa final de la creación de los objetos.

Figura 72

Impresión de soporte para subsistema “B” (Inserción de aguja)

Nota. Elaboración propia

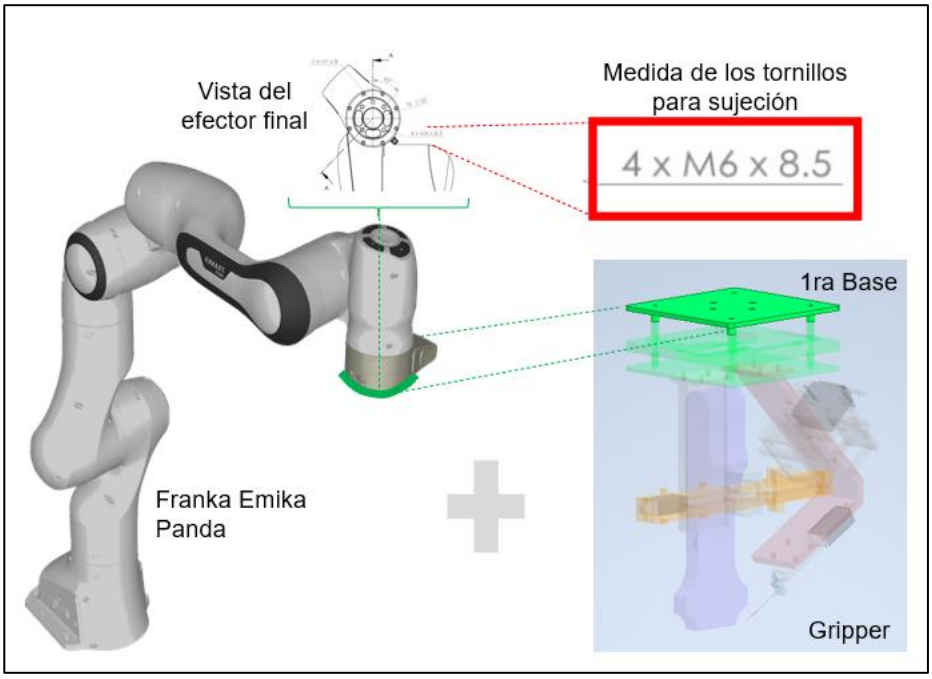
Asimismo, es posible incorporar más elementos dentro de la misma placa magnética para que se laminarán a la misma vez, en la que se configurará la cantidad de material y tiempo pertinente para la elaboración de los componentes.

■ INTEGRACIÓN DEL MÓDULO ROBÓTICO AL EFECTOR FINAL.

Una vez se haya impreso todas las piezas creadas, se integrarán con tornillos para así formar la estructura compacta requerida donde se ubicarán los componentes de control, transductor y actuadores, así como los elementos médicos para ejecución del procedimiento. Finalmente, la **Figura 73** ilustra la forma en que se conectará el módulo robótico y los demás componentes al efector final, dando como resultado a un sistema autónomo para abordar el proceso quirúrgico de la toracocentesis.

Figura 73

Módulo añadido al efector final del robot



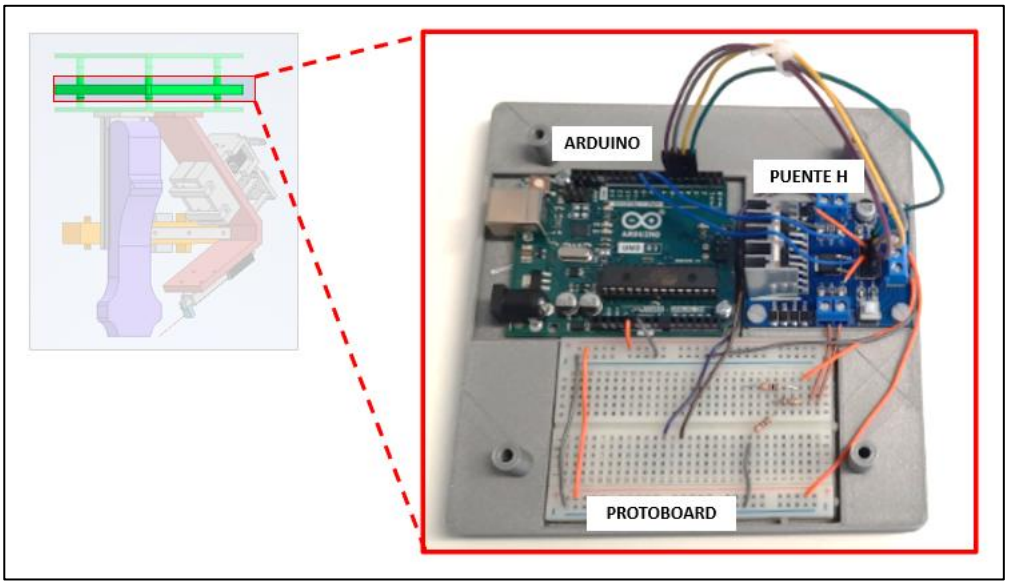
Nota. Elaboración propia

■ INTEGRAR COMPONENTES

Luego en la **Figura 74** muestra la segunda plataforma del módulo robótico con detalle del posicionamiento de la tarjeta electrónica Arduino UNO, puente H (L298N) y el protoboard, conectados según el diagrama presentado en la **Figura 47**, para controlar los movimientos de avance y reversa de los actuadores lineales. La función del protoboard el cual ejerce de soporte para las múltiples conexiones que comparten más de 2 puntos en común.

Figura 74

Ubicación de Componentes electrónicos de control



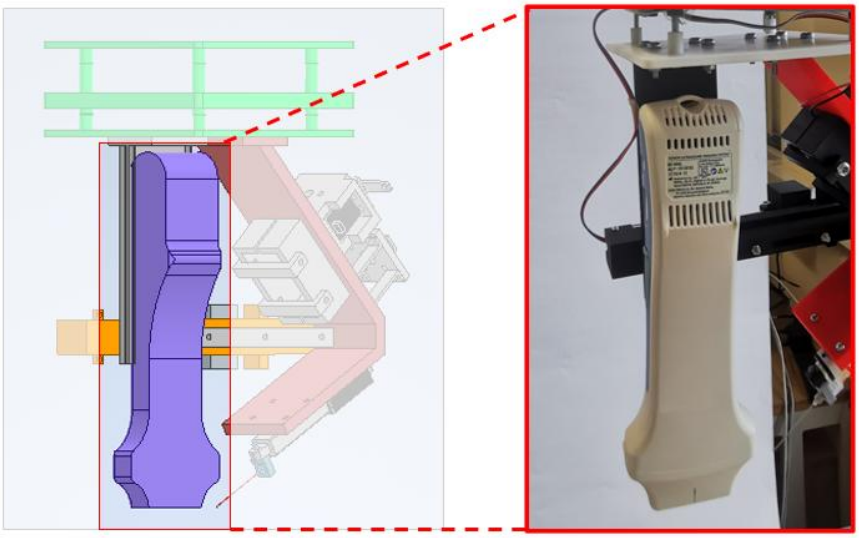
Nota. Elaboración propia

Seguidamente, se presentará la ubicación de los componentes relacionados a los subsistemas A, B y C, tal como se detalló en el esquema presentado en la **Figura 48**.

En la **Figura 75** se presenta la ubicación del escáner de ultrasonido en el módulo, la cual forma parte del subsistema A.

Figura 75

Montaje del Escáner de Ultrasonido

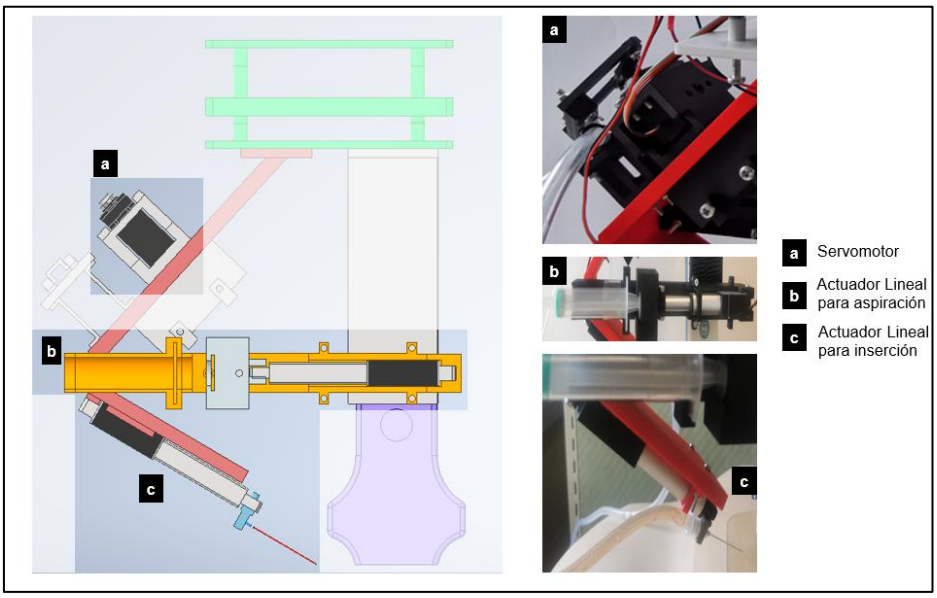


Nota. Elaboración propia

Luego, se incorporan los actuadores lineales eléctricos y servomotor que contribuirán a la ejecución del procedimiento de la toracocentesis, tal como se observa en la **Figura 76**, donde en la imagen izquierda se visualiza el diseño establecido, mientras que la imagen derecha representa el montaje de los componentes físicamente.

Figura 76

Actuadores Eléctricos



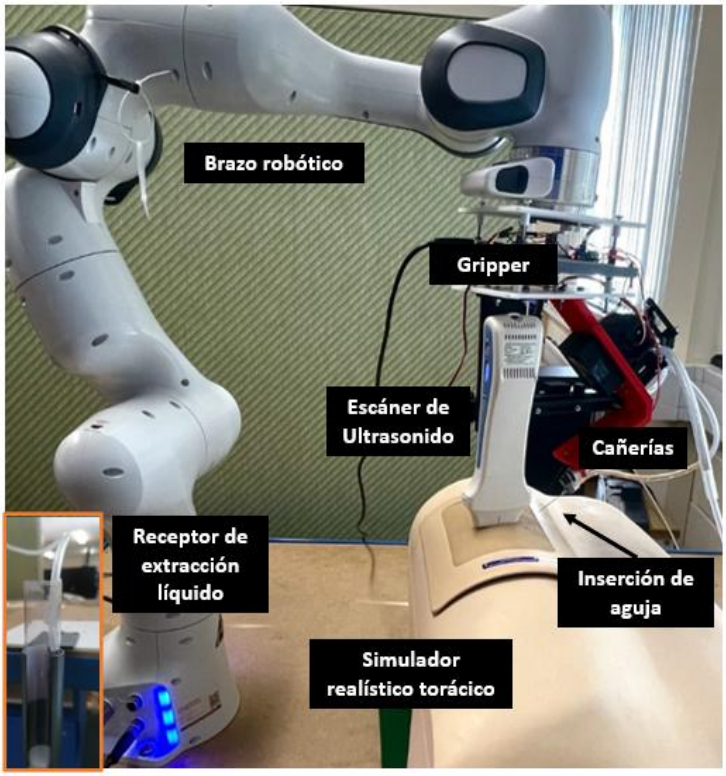
Nota. Elaboración propia

De la **Figura 76**, la sub-imagen (a) representa el servomotor, la sub-imagen “b” representa el actuador lineal para aspiración – drenaje, el cual se encuentra conectado a una jeringa mediante un acople para sincronizar el movimiento del vástago del actuador al pistón de la jeringa; por último, la sub-imagen “c” muestra el actuador lineal para inserción el cual está conectada a una aguja 22G.

Finalmente, una vez el módulo robótico se encuentre fabricado y los componentes debidamente incorporados, el conjunto total es acoplado al efector final. En la **Figura 77** se muestra la implementación de las piezas creadas, acopladas al efector final del robot, y los componentes que contribuyen a la ejecución de la toracocentesis.

Figura 77

Implementación del módulo añadido al Robot



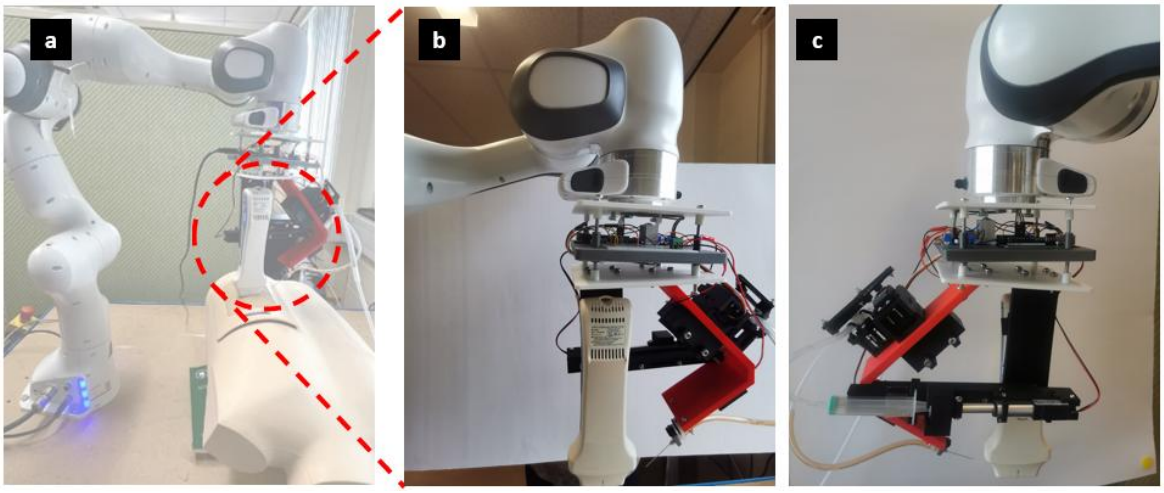
Nota. Elaboración propia

En la **Figura 78**, se observa la integración del módulo robótico, en la que se visualiza la ubicación de los componentes electrónicos como el servomotor, los actuadores y el escáner de ultrasonido. Esta imagen comprende la vista frontal y posterior de dicho módulo.

Por último, el diseño del módulo robótico resulta esencial, ya que para lograr la precisión que requiere el sistema, el eje axial de la aguja se encuentra perfectamente alineado con el plano de emisión del transductor de ultrasonido, como se observa en la **Figura 79**. Esta configuración es crucial para que la visión asistida durante el procedimiento quirúrgico sea precisa.

Figura 78

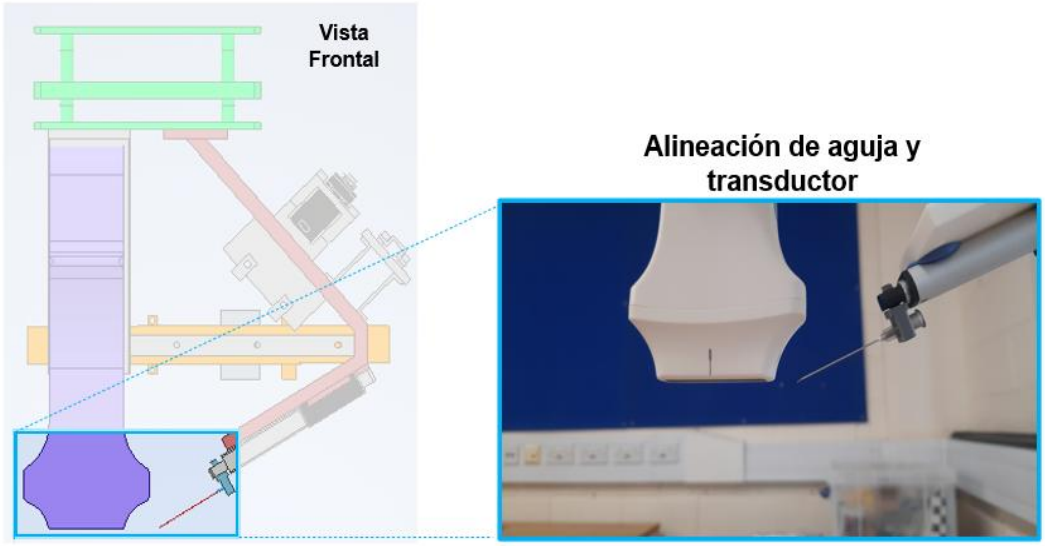
Vista frontal y posterior del sistema ensamblado



Nota. a) Implementación del módulo robótico como la **Figura 77**, b) vista frontal de la implementación, c) vista posterior de la implementación. Elaboración propia

Figura 79

Alineamiento correcta entre aguja y transductor para guiar al sistema



Nota. Elaboración propia

Luego de integrar físicamente el módulo robótico, componentes y, ser acoplado al efector final del robot, se debe abordar el aspecto de control. Esto implica incluir los programas en los componentes de control que permitirán la ejecución organizada de las

etapas de la toracocentesis para lograr la autonomía completa del sistema. En la siguiente subsección se profundizará acerca de cada algoritmo.

3.3 Bloques Funcionales del Sistema Autónomo

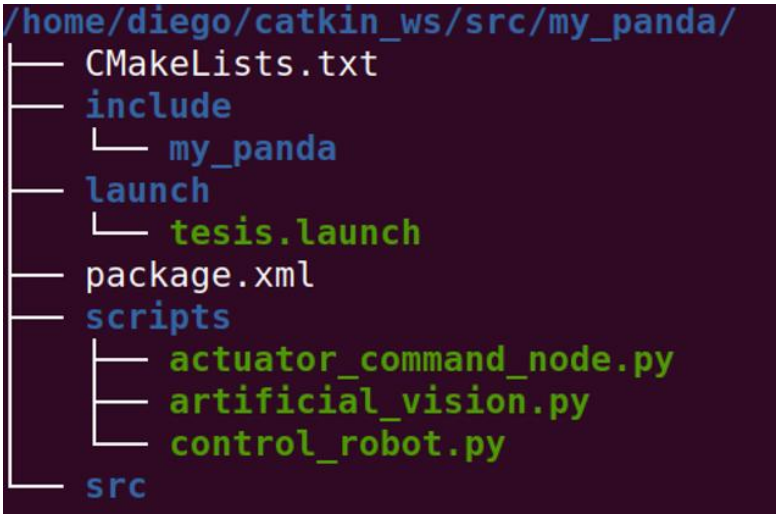
El sistema autónomo se ha subdividido en bloques funcionales que permiten la comprensión lógica del proceso médico de la toracocentesis para gestionar en tiempo real una ejecución asistida por las imágenes de ultrasonido. Por ello, este apartado será abordada según la secuencia de bloques funcionales ilustrados en la sección superior de la **Figura 54**.

3.3.1 Activación del sistema

La inicialización del sistema se ejecuta dentro del entorno ROS, para lograr comunicación interna entre los algoritmos. Sin embargo, se debe configurar el entorno de trabajo en jerarquías internas como se observa en la **Figura 80**.

Figura 80

Estructura organizacional del ambiente de trabajo



Nota. Elaboración propia

Esta configuración se realiza bajo la versión ROS Noetic. El espacio de trabajo se nombra "*catkin_ws*", seguido de un paquete llamado my_panda. Luego, dentro de este paquete, se crea un folder que contenga los archivos ejecutables. La lista de instrucciones

contiene comandos usados en ROS, los cuáles se explican en detalle en el **ANEXO B,** siendo estos comandos fundamentales para la creación básica de la estructura.

Una vez establecida la estructura organizacional a nivel de paquetes y programas, los códigos son almacenados dentro de su archivo correspondiente; en este caso dentro del archivo tesis.launch y de los archivos de extensión python. La ejecución de este archivo tipo launch pondrá en marcha la operatividad del sistema autónomo a través de la siguiente línea de código en el terminal.

$$roslaunch \quad my\_panda \quad tesis.launch \quad robot\_ip\text{:}\,192.168.0.1$$

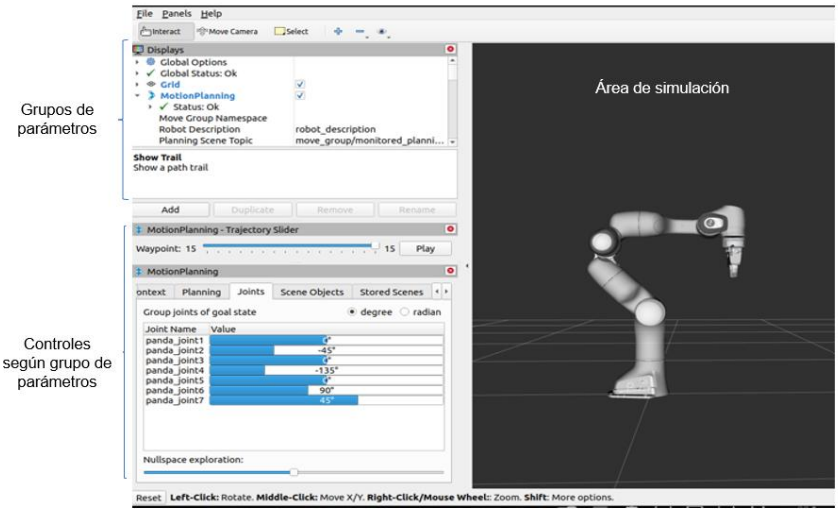
El código del archivo “tesis.launch” se describe en detalle dentro del **ANEXO C.**

3.3.2 Planificación de Trayectoria

Este apartado resulta esencial para definir y controlar el movimiento del robot Panda el cual será dirigido por un algoritmo basado en python. Para llevar a cabo la ejecución del movimiento, se hace uso del metapaquete MoveIt para la planificación y ejecución de la trayectoria, junto con el entorno RVIZ para simular planeamiento y ejecución de la trayectoria. La **Figura 81** representa dicho entorno para visualizar planificación de trayectoria.

Figura 81

RVIZ: Entorno para simulación de control de robot

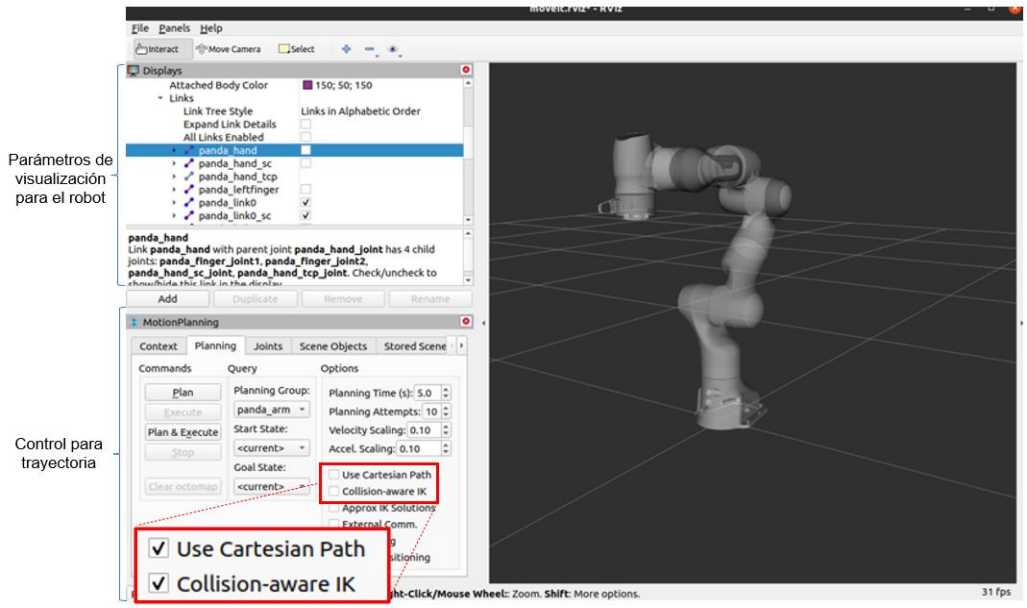


Nota. Elaboración propia

El entorno RVIZ permite establecer una configuración segura enfocado a la dinámica del robot. La **Figura 82** comprende dos opciones clave para garantizar la seguridad del sistema durante la planificación de trayectorias: la primera, relacionada al cálculo de la cinemática inversa libre de colisiones, y la segunda, el desplazamiento lineal del módulo robótico en el eje cartesiano; es decir, el movimiento no involucrará variaciones en su orientación, sino simplemente desplazamiento a lo largo de los ejes x, y, z.

Figura 82

Controles RVIZ para el proceso de planeamiento



Nota. Elaboración propia

El algoritmo control de robot comprende tres etapas, las cuáles definen la lógica para el dinamismo del robot mediante **ANEXO D.**

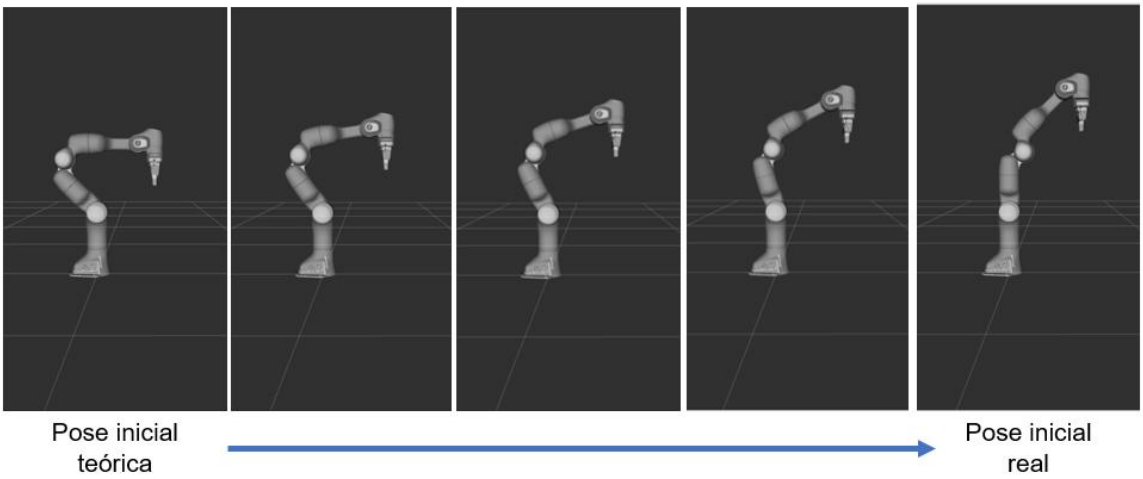
- **<u>Establecer posición real del robot</u>**

  El entorno de simulación RVIZ inicia presentando al robot en una posición real teórica; sin embargo, no necesariamente coincide con la posición real ya definida. Esta posición real del robot es vital, puesto que al incorporar el módulo robótico como parte del efector final se establece en una altura mayor para una ejecución segura.

La **Figura 83** describe el proceso visual de la transición de la posición real teórica hacia la posición inicial real al ejecutar la primera sección del algoritmo del control de robot. Esta alineación inicial permite sincronizar el robot del entorno de simulación con la posición del robot real para evitar colisiones.

Figura 83

Transición hacia la pose inicial real en sistema de simulación



Nota. Elaboración propia

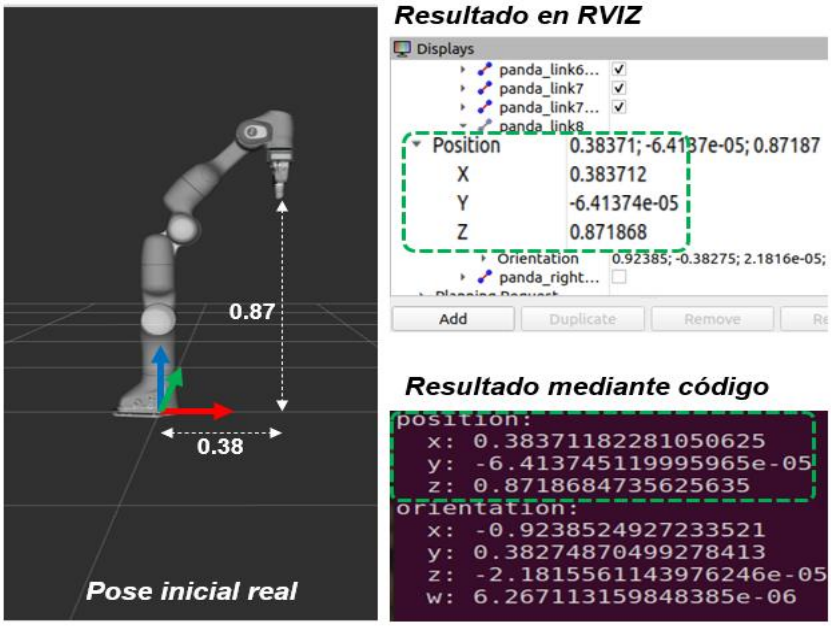
Simultáneamente, al ejecutar el algoritmo, el resultado de la posición inicial real se muestra tanto en el entorno de RVIZ como en la salida del terminal. Estos resultados, representados en la **Figura 84**, validan la sincronización del sistema entre el entorno simulado y real.

- **Planear la trayectoria del movimiento real**

  Esta fase de la planificación consiste en determinar los puntos cartesianos del desplazamiento del robot, desde la pose inicial real hacia una posición objetivo. La **Figura 85** presenta una primera imagen referido a la pose inicial del robot, la imagen intermedia representa la secuencia establecida para desplazamiento previsto siguiendo una trayectoria libre de colisiones, mientras que la última imagen muestra la pose final alcanzada.

Figura 84

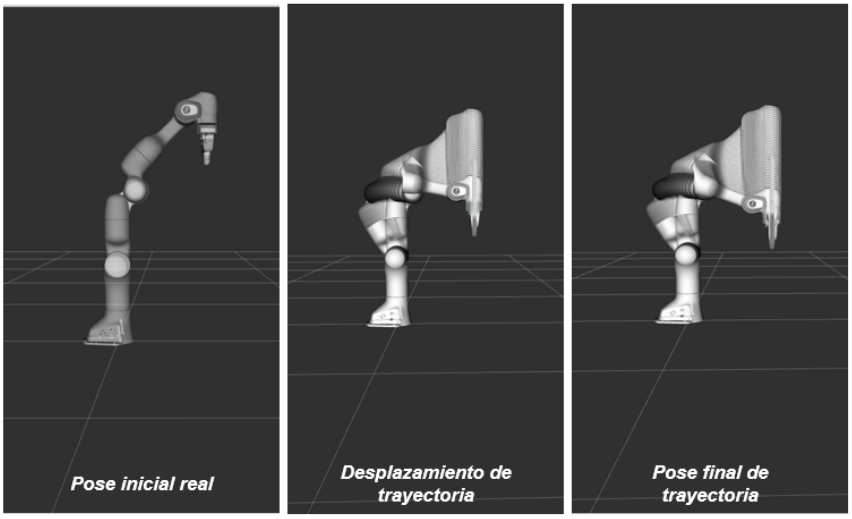
Posición inicial real del robot



Nota. Elaboración propia

Figura 85

Planificación de trayectoria en RVIZ



Nota. Elaboración propia

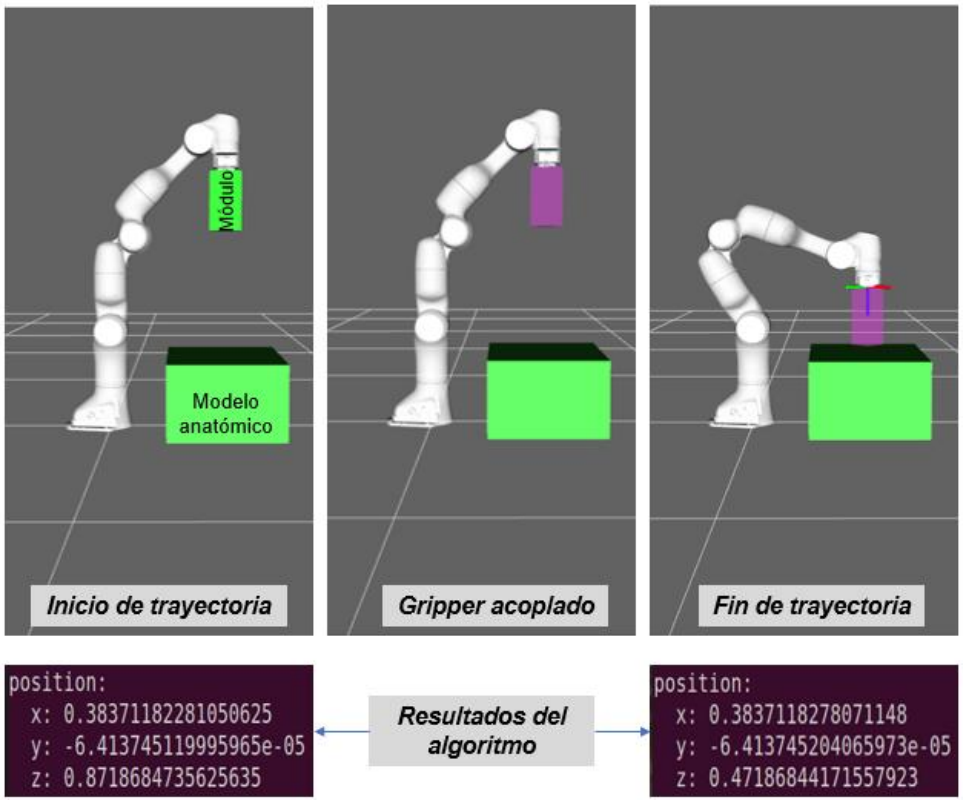
Esta fase es esencial para garantizar y validar un desplazamiento cartesiano del efector final libre de colisiones, antes de ejecutar el movimiento en un brazo físico.

- **Ejecución de trayectoria**

  Esta última fase representa la ejecución del movimiento del robot a través de la trayectoria definida en la fase anterior. Para darle un énfasis más realista a esta ejecución, se presenta la **Figura 86** donde no solo se percibe al robot sino también se ha considerado dos objetos que representen al simulador anatómico y al módulo robótico, basado en las dimensiones reales de cada uno.

Figura 86

Ejecución de trayectoria con el módulo acoplado al efector final



Nota. Elaboración propia

En la imagen izquierda de la **Figura 86** muestra el inicio de trayectoria con el modelo anatómico representado por el objeto inferior de color verde, mientras que el módulo robótico es presentado justo debajo del efecto final. La imagen intermedia, representa el módulo robótico acoplado al brazo robótico, en color morado, mostrando que se ha logrado una simulación de acoplamiento virtual. En tanto, la imagen derecha muestra el movimiento del

robot con el módulo robótico a lo largo del eje z, hasta llegar a una posición final, la cual está localizada sobre el modelo anatómico.

Los bloques inferiores representan el resultado numérico del algoritmo de planificación; siendo notoria la variación vertical en el eje z, mostrando un cambio descendente de 0.4 cm (De: 0.8718 a 0.4718). Mientras que la posición del módulo robótico a lo largo del eje x e y se mantiene constante.

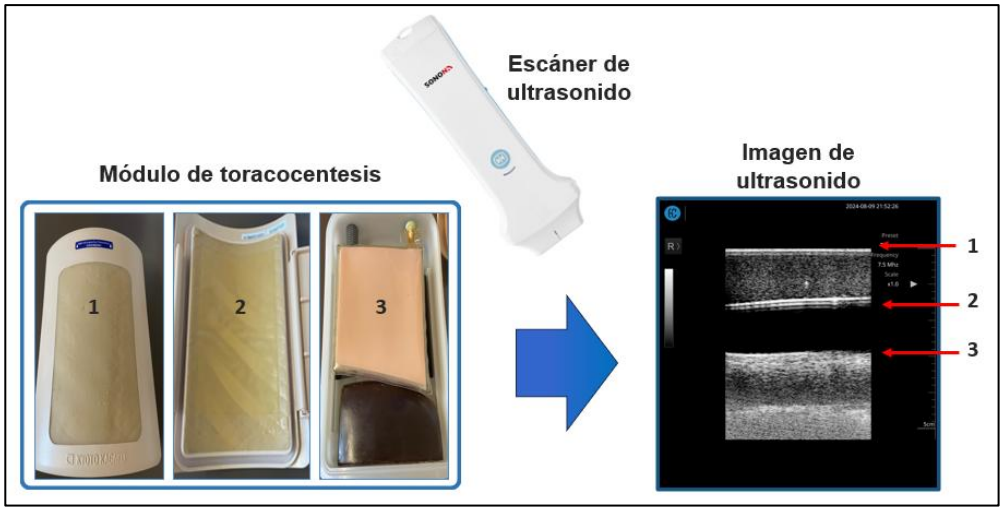
3.3.3 Sistema de Visión Artificial

- **Obtención de las imágenes de ultrasonido**

Para obtener la imagen de análisis, se utiliza el escáner de ultrasonido Sonon 300L con lo que se obtendrá una imagen de visibilidad rectangular debido al tipo de transductor lineal. Como se observa en la **Figura 87**, la imagen de ultrasonido es el resultado de los ecos emitidos por el escáner, describiendo la estructura del tejido epitelial y del pulmón.

Figura 87

Obtención de las imágenes de ultrasonido



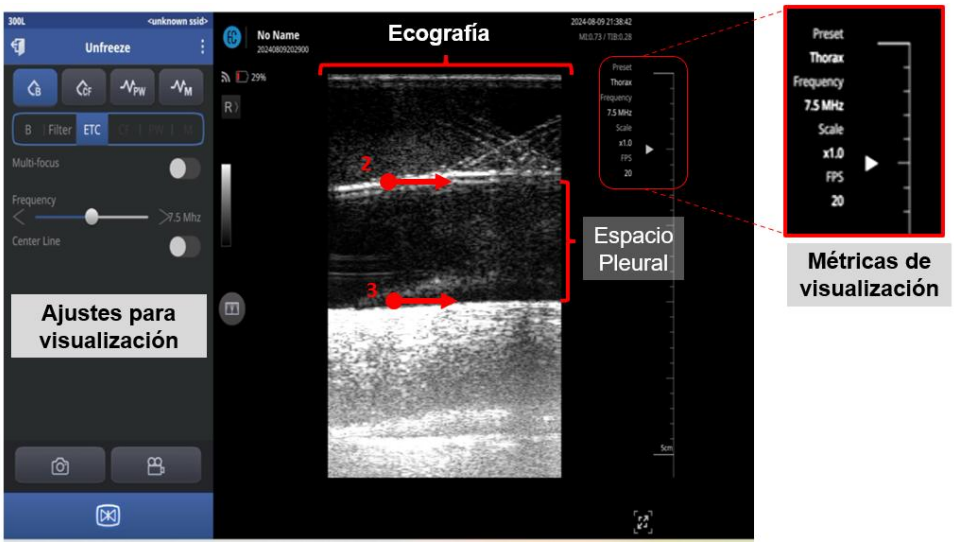
Nota. Superficie expuesta de tejido epitelial (1), superficie interna de tejido epitelial (2), y superficie de la representación de pulmón (3). Elaboración propia.

En la **Figura 88**, se aísla la imagen de ultrasonido, siendo el espacio pleural definido por la referencia (2) y la referencia (3). El espacio pleural que contiene líquido se representa por la zona anecoica que se muestra en color negro. Asimismo, también se ilustran las

métricas de la visualización al describir la frecuencia definida por los ecos (7.5 MHz), la escala real de la proyección respecto del entorno real usando el módulo (x1.0) y la transmisión de 20 frames por segunda (FPS).

Figura 88

Secciones de la imagen de ultrasonido



Nota. Detalles de la imagen de ultrasonido representando la ecografía hecha al módulo de toracocentesis. Elaboración propia

Esta imagen, describe una ecografía real del módulo de toracocentesis visualizado mediante la tablet; sin embargo, aún se percibe ciertas imperfecciones en la definición de la imagen, por lo que en la siguiente subsección se abordará el tratamiento de esta imagen para mejorar su visualización y posterior uso.

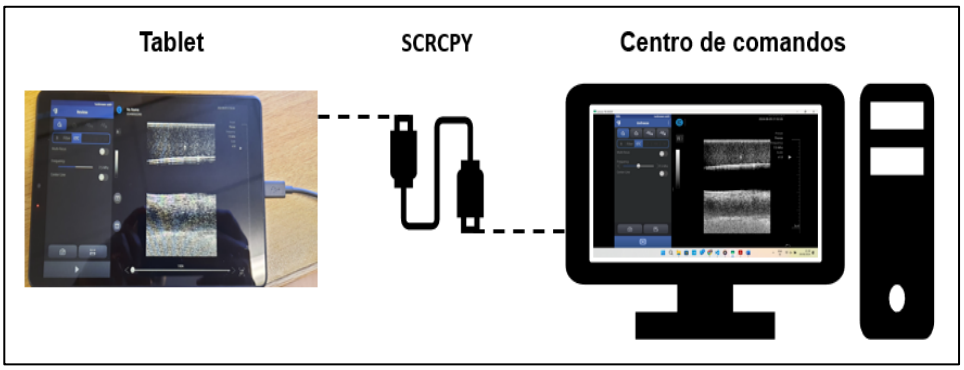
- **Visualización de lecturas de ultrasonido**

Las lecturas ecográficas son transmitidas a la tablet que, facilita la visualización de las estructuras anatómicas del módulo de toracocentesis. Sin embargo, es necesario que este contenido se proyecte en el centro de comandos con la finalidad de procesar dichas lecturas para extraer datos relevantes que se conviertan en señales de entrada para guiar al sistema de manera autónoma en el proceso de la toracocentesis. La **Figura 89** describe

visualmente la conexión real que se requiere para lograr la visualización del contenido en el centro de comandos.

Figura 89

Transferencia de información

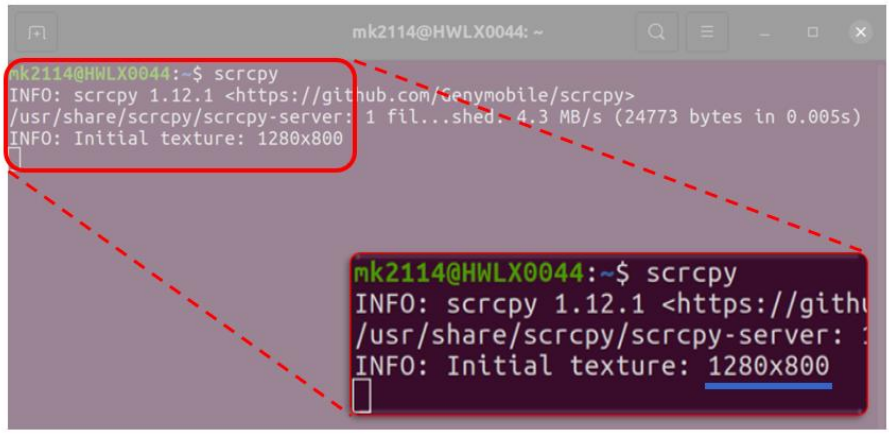


Nota. Se usa un cable USB para transmitir el contenido desde la pantalla de la tablet. Elaboración propia.

En la **Figura 89** se observa en el centro el cable que conecta la tablet con el centro de comandos. No obstante, para que se produzca una transmisión de datos vía USB se hace uso de una aplicación llamada SCRCPY que actúa como un duplicador de pantalla. Esta aplicación es fundamental puesto que reduce el costo al prescindir de un dispositivo streaming. En este caso, las imágenes son transmitidas desde una tablet con sistema operativo Android al centro de comandos equipado con sistema operativo Ubuntu, al ejecutar la aplicación como se observa en la **Figura 90**.

Figura 90

Aplicación SCRCPY para transmisión



Nota. Ventana de ejecución de comando de la aplicación en Ubuntu. Elaboración propia.

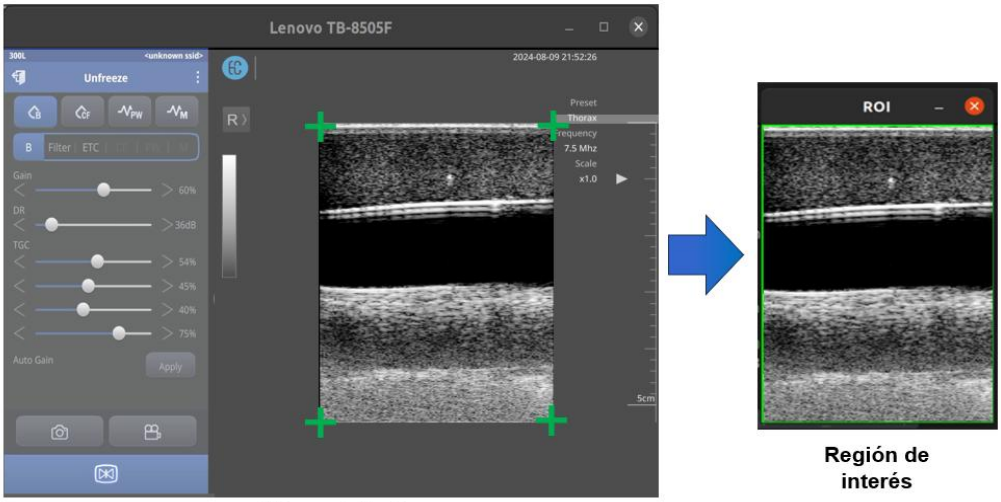
Un dato relevante de este despliegue es que su ejecución nos brinda la dimensión de la resolución de la ventana duplicada, es decir en el centro de comandos, en este caso se representa con 1280x800. Estas medidas permitirán identificar la zona de interés que debe ser aislada (segmentada) y su posición según la ubicación de los pixeles, para ejecutar el procesamiento digital de la imagen y se defina computacionalmente el área de análisis. El siguiente proceso relacionado al tratamiento de la imagen está basado en el algoritmo descrito en el **ANEXO E**.

- **Procesamiento de las imágenes de ultrasonido**

La imagen original recibida en el centro de comandos es tal cual como se visualiza **Figura 88**; sin embargo, a pesar de que la ecografía se visualiza en escala de grises, realmente no se encuentra en dicho formato sino en RGB. Esta afirmación se basa en la presencia de áreas de color azul en la sección “Ajustes de Visualización”. En este caso, tal como se observa la **Figura 91**, la imagen original es recibida en el centro de comandos y, dado que la ecografía se encuentra en una posición específica respecto de la imagen completa, se procede a segmentar la región delimitada por las marcas verdes.

Figura 91

Segmentación de la región de interés



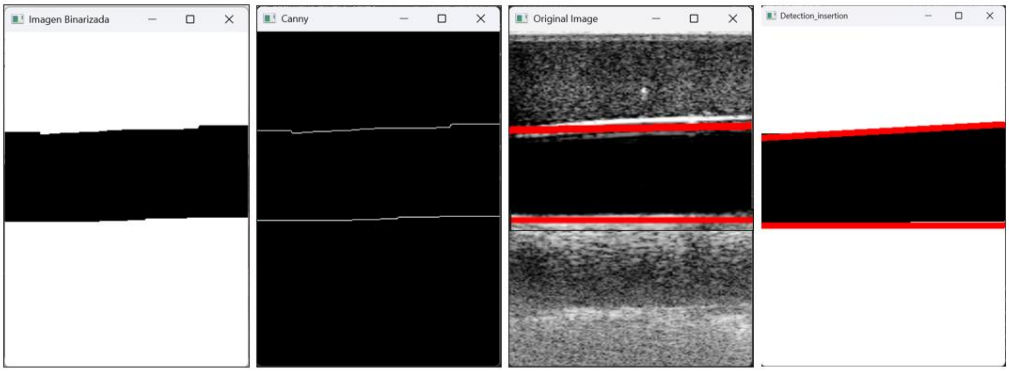
Nota. Elaboración propia.

El resultado de la segmentación destaca solo el área de la ecografía, según la **Figura 91**, la imagen ubicada en la derecha se exhibe a través de una ventana nueva la cual tiene por título la palabra, ROI (Región de Interés).

Seguidamente, una vez el ROI ya se encuentra en una sub-ventana, se realiza una serie de operaciones relacionada a procesar la imagen para obtener entornos equivalentes para el análisis computacional. La **Figura 92** manifiesta las etapas del procesamiento de la imagen ROI.

Figura 92

Etapas de desarrollo computacional de la región de interés



Nota. Elaboración propia

- Binarización de ROI, la imagen de la ecografía se transforma para resaltar estructuras anatómicas.
- Detección de Líneas, esta parte del procesamiento permite extraer los bordes relevantes de la región de interés. Asimismo, se aplica la transformada de Hough, para detectar línea recta.
- Vista original, es la representación de la ecografía, destacando en color rojo las líneas rectas (Hough).
- Vista computacional, es la representación gráfica mediante una máscara tomando como base la vista original incluyendo las líneas de Hough para un seguimiento de la punta de la aguja.

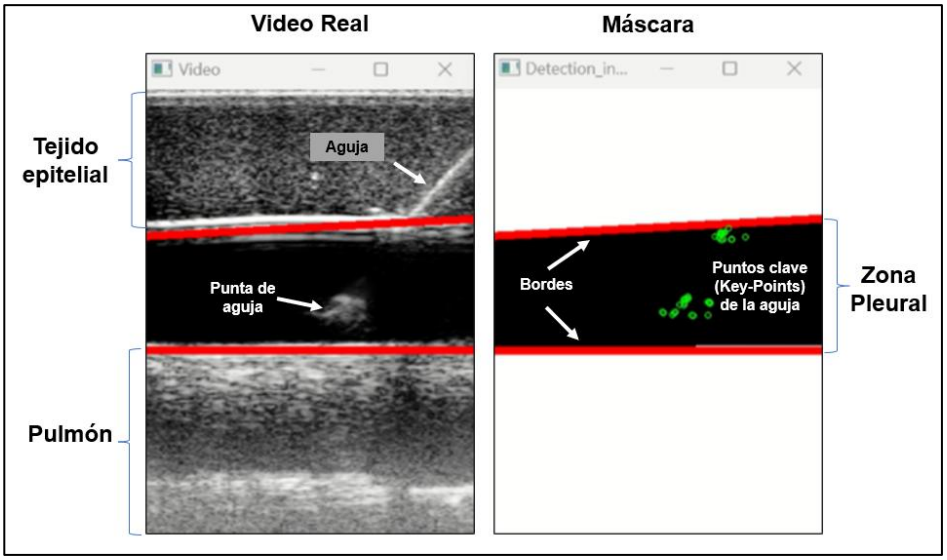
Estas dos últimas imágenes serán fundamentales para gestionar el seguimiento real de la punción de la aguja y el seguimiento computacional; siendo este último una representación artificial de tal manera que identifique en que momentos se generan las señales adicionales para los demás componentes del sistema.

- **Detección de Puntos Clave (Key-Points)**

Finalmente, luego de haber establecido las vistas base para gestionar el seguimiento de la punción percutánea tanto en el entorno ecográfico, así como el computacional, se logra una representación precisa de la trayectoria de la punta de la aguja en la zona pleural mediante la representación de los puntos clave (key-points) tal como se ilustra en la **Figura 93**.

Figura 93

Representación computacional de la ecografía



Nota. Representación del video real de la ecografía y computacional (máscara). Elaboración propia.

Esta mejora de la visualización será un aspecto clave en la asistencia al sistema autónomo, al optimizar el desarrollo del proceso clínico de la toracocentesis teniendo en cuenta aspectos de precisión y seguridad.

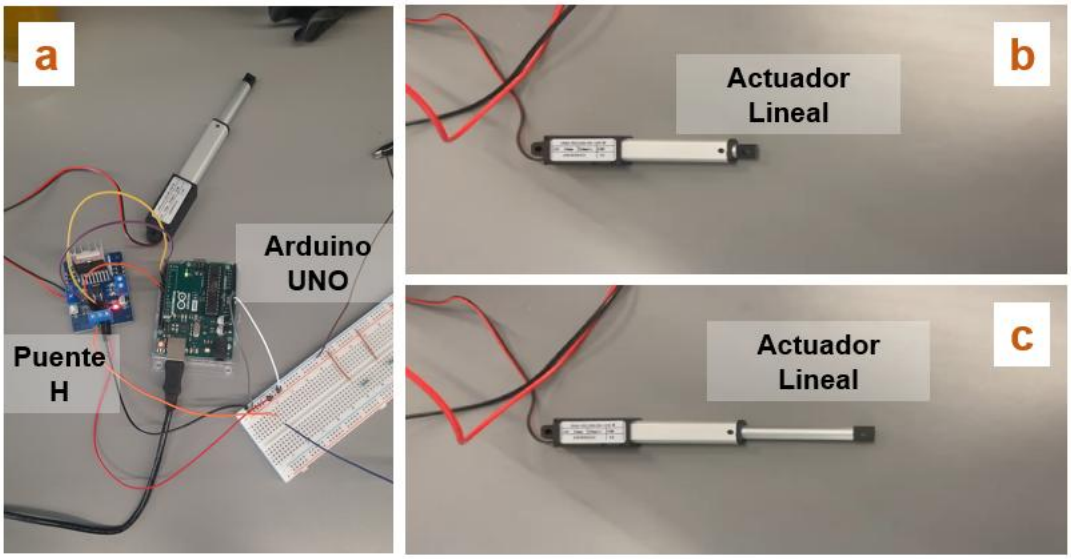
3.3.4 Inserción de Aguja

El funcionamiento del actuador del subsistema B es controlado Arduino UNO. El código se desarrolló en Arduino IDE basado en programación C++. Este algoritmo, descrito en el **ANEXO F**, es descargado vía USB a la placa Arduino para que finalmente, las rutinas implementadas de extensión y retracción sean ejecutadas a través de la comunicación mediante ROS.

Para esta tesis, se ha implementado un control electrónico como se muestra en la **Figura 94** el cual define la extensión y retracción del vástago, mediante conexiones de cables de calibre 24 AWG.

Figura 94

Componentes electrónicos para control



Nota. Configuración de la conexión entre componentes electrónicos (a), actuador lineal con vástago comprimido (b), actuador lineal con vástago extendido (c). Elaboración propia.

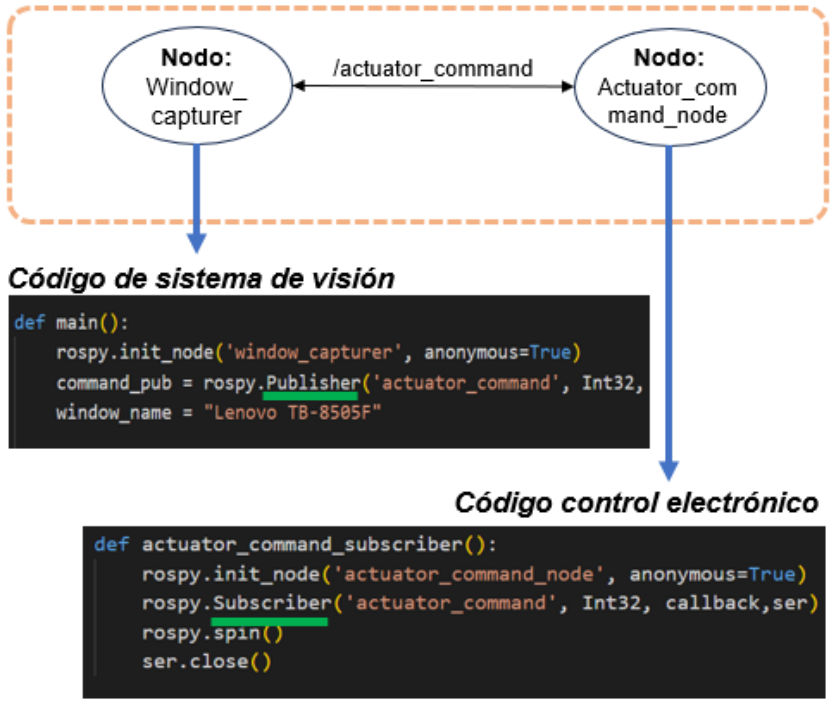
La configuración electrónica establecida en la **Figura 94 – a**, facilita el control de movimiento del vástago el cual será útil para replicar el efecto de inserción y remoción de la punción de la aguja en el módulo de la toracocentesis. En tanto, la sub-imagen b y c, representan el actuador con el vástago en posición retraída y extendida, respectivamente.

Otro aspecto importante de la activación del actuador para el subsistema B, es el momento en que el sistema determina que se debe efectuar la activación.

La **Figura 95** muestra la comunicación de dos nodos correspondientes al código del sistema de visión artificial y al control electrónico. El nodo “window_capturer” que contiene al código del sistema de visión, publica mensajes en el tópico “actuator_command”. Por su parte, el nodo “Actuator_command_node” que contiene al código de control electrónico, se subscribe al tópico mencionado para leer la información publicada. En este caso, se valida que el sistema de visión es el encargado de guiar y determinar en qué momentos debe activarse el actuador.

Figura 95

Comunicación ROS entre nodos para control electrónico



Nota. Elaboración propia.

En tanto, se puede visualizar que el mensaje asociado al tópico es de tipo “Int32”; por lo que el contenido del mensaje transmitido será numérico. Los mensajes establecidos en el algoritmo se muestran en la **Tabla 9**.

Tabla 9

Mensajes comunicados desde ROS a Arduino

MENSAJE	DESCRIPCIÓN	ACCIÓN FÍSICA DEL SISTEMA
1	Inserción de aguja	Subsistema B: Extensión del vástago
2	Remoción de la aguja	Subsistema B: Retracción del vástago
3	Expulsión de líquido	Subsistema C: Extensión del vástago
4	Succión de líquido	Subsistema C: Retracción del vástago
5	Aperturar válvula de 3 vías	Giro horario del servomotor
6	Cerrar válvula de 3 vías	Giro antihorario del servomotor.

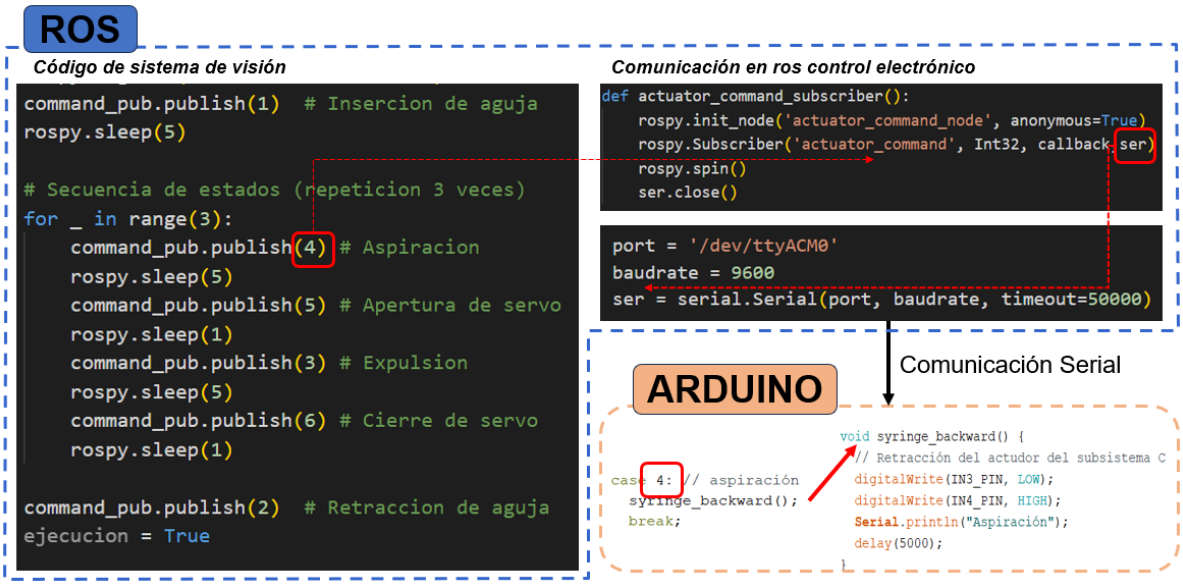
Nota. Elaboración propia.

Para mayor detalle del proceso de comunicación, se presenta la **Figura 96**, donde se describen cuatro puntos clave que intervienen en el proceso de envío de mensajes desarrollados en ROS y en Arduino para el proceso de aspiración (mensaje *4 según* **Tabla 9***)*, la cual físicamente corresponde con la retracción del actuador del subsistema C.

- **ROS |** El valor entero (4) se publica desde el código del sistema de visión.
- **ROS |** se suscribe recibiendo dicho valor entero almacenando esta información en la variable ‘ser’.
- **ROS |** El contenido de la variable se encamina para transferirlo a través del puerto serial.
- **Arduino** | Este valor (4) es recibido en la tarjeta Arduino UNO, relacionando dicho valor entero con el caso correspondiente descrito en el código implementado en la tarjeta Arduino. De acuerdo con la lógica establecida, este valor (4) es interpretado para retraer el actuador del subsistema C.

Figura 96

Secuencia de envío de mensajes de ROS a Arduino



Nota. Elaboración propia.

Esta imagen, describe claramente la transferencia de información desde ROS hacia la placa Arduino vía comunicación serial.

3.3.5 Drenaje del líquido

La última etapa que realiza el sistema está enfocada en la recuperación del líquido en un recipiente. Para ello, el líquido fluye a través de tubos transparentes que son de material de polímero biomédico para garantizar la esterilidad del procedimiento clínico.

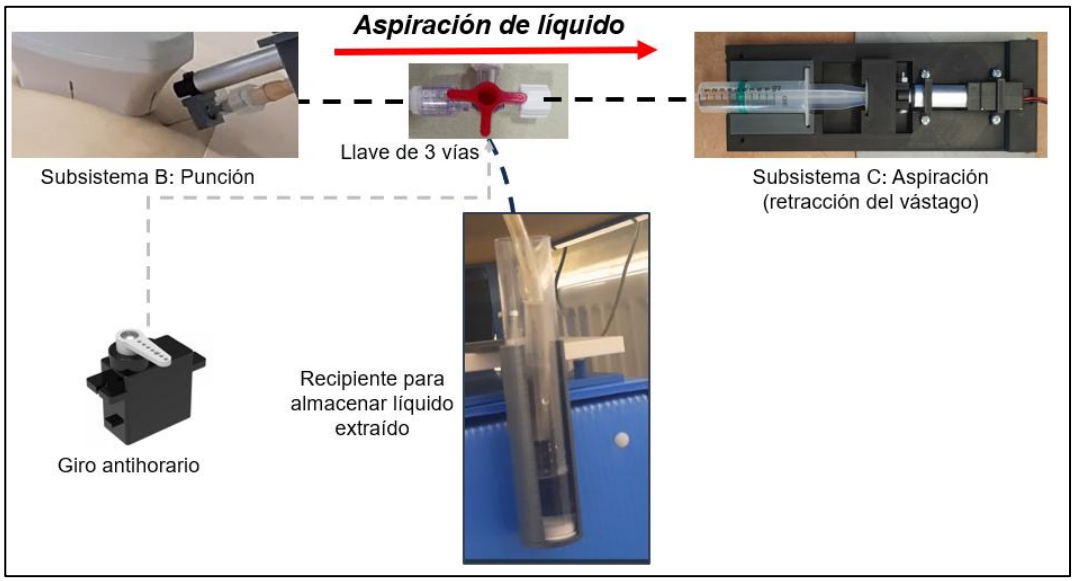
Basado en la **Tabla 9** y en la secuencia establecida **Figura 96**, los mensajes que intervienen directamente en el drenaje del líquido son ***5*** y ***6.***

El drenaje del líquido se divide en dos etapas, la primera corresponde a la aspiración del líquido donde la válvula de tres vías enlaza al subsistema B y al subsistema C mediante tubos de polímero como se observa en la **Figura 97**; mientras que la segunda etapa corresponde a la expulsión del líquido; donde el flujo cambia de curso de acuerdo al cierre de la válvula de tres vías como se observa en la **Figura 98**. El movimiento de la válvula se

acciona a partir del funcionamiento del servomotor, girando en sentido horario o antihorario lo que contribuye a la redirección del flujo del líquido.

Figura 97

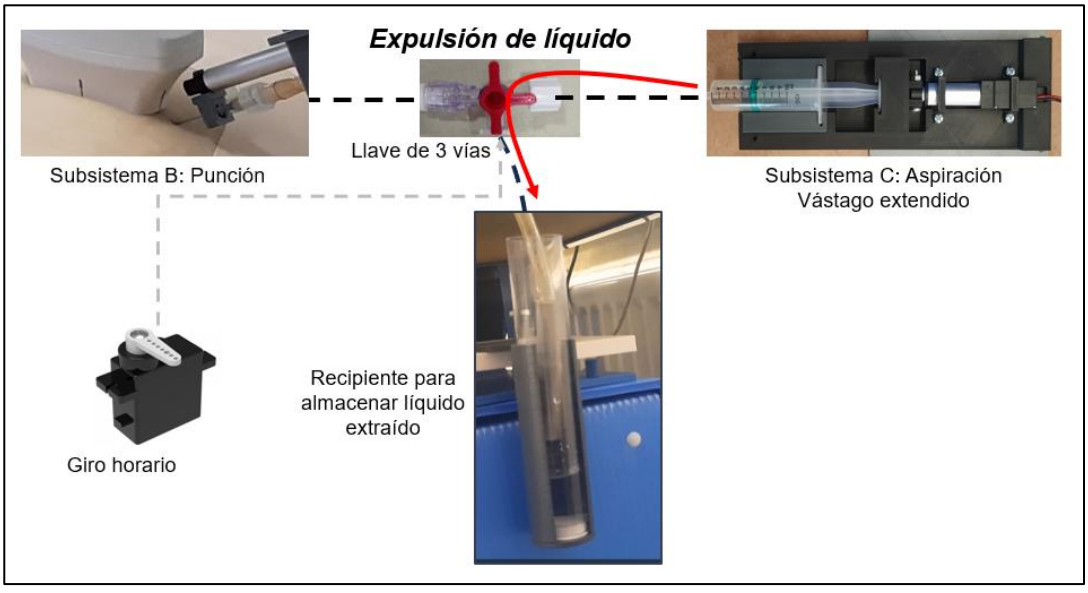
Flujo de aspiración del líquido



Nota. Elaboración propia.

Figura 98

Flujo de expulsión del líquido



Nota. Elaboración propia.

CAPÍTULO IV

RESULTADOS, CONTRASTE DE HIPÓTESIS Y DISCUSIÓN DE RESULTADOS

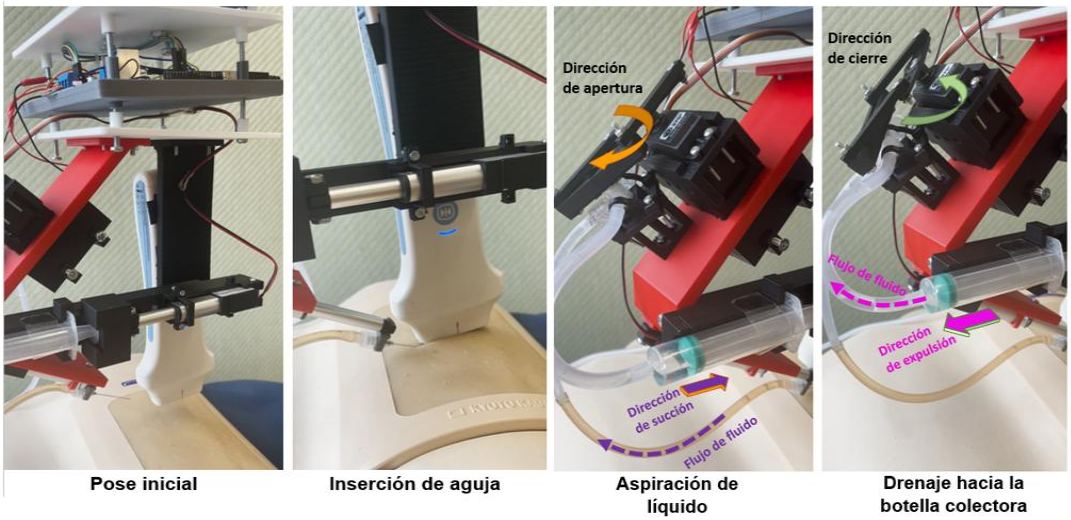
4.1 Resultados

El sistema robótico planteado en esta tesis logró replicar el proceso quirúrgico de la toracocentesis involucrando componentes electrónicos e instrumentos médicos basado en un enfoque tecnológico. En la **Figura 99** se observa la secuencia de un sistema autónomo determinado en 4 imágenes:

- Pose inicial: Brazo robótico ubicado en la zona superior del área de trabajo con el módulo robótico y sus componentes listos para dar inicio al proceso.
- Inserción de aguja: El actuador lineal realiza la punción una vez que se ha comenzado la lectura de las imágenes de ultrasonido.
- Aspiración del líquido: El actuador lineal retrae el pistón para generar el efecto de succión de líquido
- Drenaje del líquido: El servomotor cambia la posición de la válvula para expulsar el líquido hacia el recipiente

Figura 99

Representación Visual de las etapas de la toracocentesis

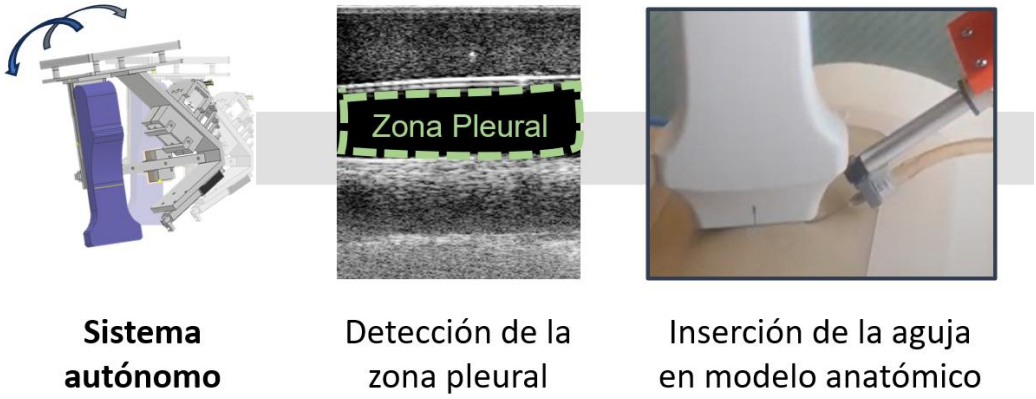


Nota. Elaboración propia.

A continuación, se presenta los resultados acompañados de una visualización interna generadas por el software, para lo cual, el resultado de un ciclo entero se ha dividido en tres resultados. El primer resultado, ***Resultado A***, tal como se ilustra en la **Figura 100**, aborda la acción del control robótico desplazándose hacia el modelo de simulación físico, luego se reconoce el área pleural para finalmente realizar la inserción de la aguja

Figura 100

Resultado A: Control robótico y punción

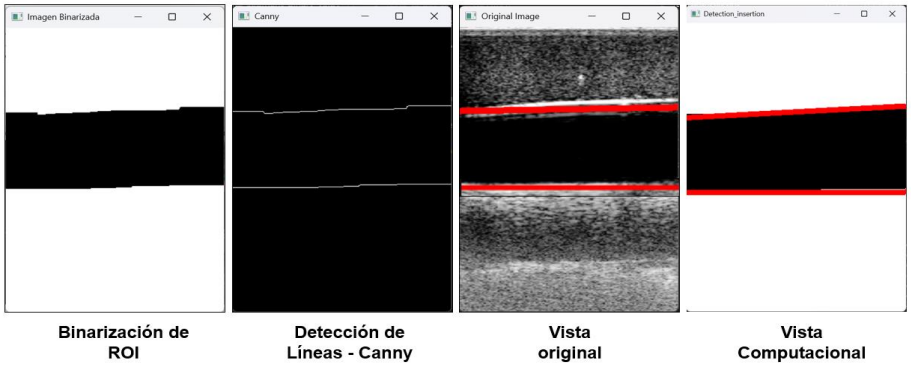


Nota. Elaboración propia.

El segundo resultado, ***Resultado B,*** tal como se ilustra en la **Figura 101**, detalla la secuencia de la transformación de las imágenes de ultrasonido aplicando técnicas de binarización, detección de líneas y generar una comparativa entre la vista original y computacional y realizar el monitoreo de la punción de la aguja mediante los puntos clave.

Figura 101

Resultado B: Sistema de visión artificial



Nota. Elaboración propia.

Una vez, insertada la aguja, se visualizará puntos descriptores como se aprecia en la primera imagen de la **Figura 102**, la cual es indispensable para determinar que la aguja ya se insertó y se puede proceder con la activación del subsistema C; es decir, para replicar la aspiración y expulsión del líquido.

Figura 102

Resultado C: Aspiración y Drenaje

Nota. Elaboración propia.

El sistema autónomo para ejecutar el tratamiento clínico de la toracocentesis fue evaluado en 11 ensayos experimentales. A continuación, en la **Tabla 10**, se describen los resultados cuantitativos de dichos ensayos.

Tabla 10

Resultados cuantitativos de la toracentesis

MÉTRICA	MEDIA	RANGO
Ejecuciones completadas con éxito (%)	90%	90 – 100
Guía mediante ultrasonido (s)	60	57 – 64
Precisión de la detección (%)	90%	90 – 100
Tiempo de inserción de la aguja (s)	6.5	6 – 7
Volumen de fluido aspirado (mL)	13	11 - 14
Tiempo de retracción de la aguja (s)	4.7	4 – 5

Nota: Elaboración propia.

La **Tabla 10** describe los resultados promedio obtenidos durante la ejecución del sistema en un entorno de simulación controlada. El éxito de este proyecto es de 90%, debido a que 10 de las 11 ejecuciones fueron exitosas durante todo el procedimiento. En promedio,

el proceso de guía de ultrasonido tomó 60 segundos, con un rango de variación entre 57 y 64 segundos. La precisión del área objetivo, detectada por el sistema de visión, está asociado con el desempeño del sistema, por lo que también se considera 90%. La ejecución que no progreso adecuadamente se debió a una segmentación inadecuada de la zona objetivo, siendo que las líneas fronteras que determinan dicha zona presentaban alteraciones.

Respecto a la ejecución de punción, el tiempo medio de inserción de la aguja fue de 6.5 segundos, mientras que el tiempo promedio de retracción fue de 4.7 segundos, lo que determina la estabilidad del control de movimiento. Finalmente, el volumen aspirado promedio es de 13 mL, dentro de un rango de 11 a 14 mL, lo que demuestra el cumplimiento del objetivo del sistema.

4.2 Contraste de Hipótesis

Basado en los resultados obtenidos, se procede a realizar un contraste con las hipótesis formuladas. En la **Tabla 11**, se muestra de manera organizada el resultado de esta comparación.

Tabla 11

Contraste de hipótesis

Tipo de Hipótesis	Hipótesis	Resultado	CONTRASTE
General	El desarrollo de un sistema que integre un manipulador robótico guiado por visión artificial automatizará el procedimiento de toracocentesis optimizando la eficiencia de la intervención quirúrgica.	Integración exitosa del reconocimiento de la zona objetivo en tiempo real y control de movimiento para lograr el proceso automatizado de la toracocentesis	Hipótesis confirmada
Específica	El desarrollo del sistema autónomo propuesto contribuirá a maximizar la eficiencia del procedimiento clínico.	Ejecución completa desde la detección hasta la retracción de la aguja con un promedio de **60 segundos**. Además, se confirma consistencia operativa en los tiempos de inserción **(6 – 7s)** así como la retracción **(4 – 5 s)**. Por último, la repetibilidad de las acciones	Hipótesis confirmada

		de aspiración (3 veces) en cada ejecución garantiza la estabilidad del sistema.	
	El desarrollo del sistema autónomo propuesto generará una reducción del margen de error clínico aplicado al procedimiento clínico.	El proceso automatizado del tratamiento de la efusión pleural reduce el riesgo asociado con el proceso manual logrando un **90% de éxito** en las ejecuciones de ciclo completo	Hipótesis confirmada

Nota. Elaboración propia.

4.3 Discusión de Resultados

- Esta fase de desarrollo del proyecto se optó por el uso del módulo L298N, el cual cumple con los requerimientos técnicos del sistema para el control del proceso de inserción / extracción de la aguja, así como para el proceso de aspiración / liberación del líquido. Por ello, no se diseñó un PCB debido a que se priorizó el enfoque funcional del sistema total. Debido a la constante evolución del módulo robótico, se requirió de módulos que permitan una integración rápida y segura dentro de la estructura robótica incorporada en el efector final.
- En el sistema de visión artificial fue trascendente incluir una lógica basado en la variación de puntos clave (KP) ya que, desde el punto de vista computacional para determinar la posición de la punta de la aguja en la zona pleural, habrá un cambio significativo en la cantidad de dichos puntos. Entonces, la transición de la inserción y la remoción de la aguja está relacionado a la variación. Un incremento de la cantidad de KP, representa la inserción, mientras que el descenso representa la remoción de la aguja.
- El resultado compacto del sistema desarrollado refleja un enfoque prometedor para automatizar intervenciones quirúrgicas en la que aspectos como la precisión, eficiencia y seguridad sean indispensables. Este desarrollo de un sistema guiado de manera autónoma representa un avance con alto potencial en el campo de la salud.

- Dado que los resultados fueron sobresalientes, es posible avanzar a una siguiente etapa de validación de resultados; involucrando la participación de profesionales de la salud con gran experiencia. Ello permitiría gestionar validaciones más realistas al interactuar con modelos biológicos como cadáveres con el objetivo de evaluar el desempeño en condiciones clínicas más reales hasta que eventualmente, sea probado en seres vivos.

A partir de estos resultados, se proyecta un enfoque tecnológico transferible a otras intervenciones quirúrgicas que requieran precisión y eficiencia, así como también procesos de optimización que contribuyan a trabajos futuros.

4.4 Trabajos futuros

Esta tesis ha demostrado una integración exitosa de los conceptos de robótica, IA adaptado a procesos quirúrgicos; sin embargo, existe amplio margen para optimizar el desempeño del sistema, así como también el alcance de aplicación.

- Actualmente el presente proyecto está dirigido al procedimiento quirúrgico de la toracocentesis; sin embargo, incorporar algoritmos de aprendizaje profundo, como las redes neuronales convolucionales, permitirá optimizar notablemente el procesamiento computacional de las imágenes de ultrasonido otorgando más flexibilidad ante aspectos altamente complejos. Por ejemplo, con el soporte de dicha arquitectura computacional, será posible realizar el monitoreo de la trayectoria de la aguja optimizando en enfoque de la segmentación por instancia. Este enfoque brindará mayor flexibilidad ante los argumentos de entrada mejorando la eficacia y seguridad del procedimiento.
- En etapas futuras de desarrollo de la sección electrónica del sistema, se compactaría las conexiones de soporte del protoboard en un PCB, de tal manera que se logre optimizar el aspecto de confiabilidad, mientras que

componentes como el DRV8833 contribuirían a modernizar el sistema apuntando a tener mejor eficiencia y optimización de energía del sistema.

- Una mejora significativa en este sistema sería integrarlo con sensores hápticos con la finalidad de transmitir la sensación de textura, interacciones táctiles hacia una estación remota. De esta manera, al ser un sistema controlado a distancia, los intervencionistas se familiarizarán con partes anatómicas como órganos, tejidos, etc. Durante su etapa de aprendizaje. En consecuencia, esta mejora podría gestionar sesiones de entrenamiento para intervencionistas en etapas de formación antes de ejecutar procedimientos quirúrgicos en pacientes.
- El control de la cantidad de fluido aspirado es sumamente importante. Por ello, en una siguiente fase de desarrollo del proyecto, la estrategia de este control estaría basada en sensores de presión y/o flujo ubicado en el subsistema de aspiración; determinando con precisión el volumen de líquido extraído. Asimismo, la cantidad a aspirar dependerá principalmente del volumen anatómico del paciente; lo cual implicaría que los ciclos de trabajo puedan adaptarse automáticamente mediante la incorporación de dichos sensores.
- Habiendo logrado resultados prometedores con este sistema autónomo sobre un simulador anatómico que representa de forma realista la estructura humana, y cumpliendo satisfactoriamente criterios de confiabilidad y eficiencia, se determina que la siguiente etapa estará enfocada en aplicaciones sobre organismos vivos; luego de validaciones médicas por expertos senior, el sistema podría aplicarse sobre cadáveres y, como fase final, en pacientes.

CONCLUSIONES

Esta tesis apunta al desarrollo de un sistema autónomo para replicar el procedimiento quirúrgico de la toracocentesis logrando resultados prometedores al integrar conocimientos específicos del campo de la robótica e IA en las etapas de dicha intervención clínica.

- Se logró diseñar e implementar un módulo robótico personalizado que sostiene de manera efectiva el escáner de ultrasonido, los componentes de control electrónico (Arduino y servomotores), así como las jeringas para la inserción y succión. Este diseño logró construir una estructura mecánica rígida para mantener la estabilidad de esta sección acoplada al efector final del robot durante sus movimientos. En consecuencia, debido a su rigidez se garantiza la posición relativa entre el lector del escáner y la posición de la punta de la aguja antes de la inserción, perfeccionando la precisión y confiabilidad del sistema.

- Las subetapas del proceso de toracocentesis se lograron codificar dentro del algoritmo de visión artificial, lo que permite al algoritmo capturar la imagen original del escaneo del transductor y procesarlo en tiempo real. Este enfoque precisa la sincronización de la inserción de la aguja y su avance hasta la zona pleural (región de interés) a través de una representación visual proporcionada por el software. El algoritmo contabiliza la cantidad de puntos clave (key-points) de la punta de la aguja para monitorear el desplazamiento de esta dentro del espacio pleural hasta la remoción de esta. Por ende, se garantiza que el algoritmo envié las señales oportunas a los demás componentes, habilitando una ejecución fluida del subproceso de aspiración.

- Fue exitosa la interacción del sistema autónomo compuesto por hardware y software al simulador torácico, logrando el principal objetivo de replicar el proceso de toracocentesis de manera independiente; es decir, sin intervención externa desde la

ejecución inicial (comando launch). Este logro se refleja en la cantidad de veces llevado a cabo que corresponde a 10 ejecuciones exitosas que van desde el movimiento inicial del robot, inserción de la aguja hasta la zona pleural, extracción del líquido acumulado y retiro de la aguja del espacio pleural.

- El desarrollo exitoso de este proyecto refleja el potencial que tiene la aplicación de robótica e IA al abordar procedimientos complejos como el de la toracocentesis en el que se debe tener en cuenta varios aspectos clínicos que mantengan un entorno seguro para el paciente y confiable para el procedimiento. Por consiguiente, este proyecto innovador remarca el compromiso de la ingeniería para ponerlo en práctica al servicio de la comunidad, sobre todo en sectores donde existe un amplio margen de mejora como el sector salud; confirmando que un sistema autónomo de alto performance, confiable y seguro promoverá el acceso oportuno diagnósticos médicos para su posterior mejoría.

RECOMENDACIONES

- Para la elaboración de la implementación de esta tesis se ha considerado una aguja de calibre 22 ya que de acuerdo con el manual del fabricante del modelo anatómico indica 22G o 23G. Sin embargo, el calibre dependerá de la masa anatómica de cada paciente. Por ejemplo, si el paciente tiene mayor masa anatómica que una medida estándar, entonces se requerirá una aguja más gruesa como la 18G o 19G.
- Se recomienda no aspirar más de 1.5L de líquido pleural, ya que la reexpansión del pulmón podría generar causas adversas como neumotórax (acumulación de aire) durante el procedimiento de la toracocentesis. Esta condición podría incluirse en el algoritmo que gobierna el subsistema de aspiración para monitorear el volumen extraído o bajo supervisión de un especialista.
- Durante el proceso de escaneo, se recomienda configurar apropiadamente los parámetros de visualización del transductor, para mejorar la imagen y reducir las técnicas de preprocesamiento. Asimismo, para mejorar la calidad de la imagen y se realce las características como bordes y tejidos, se sugiere aplicar gel sobre la superficie; esto facilitará que la representación visual despliegue un mejor contraste de la estructura interna del simulador.
- Dado que hay varias etapas del procedimiento quirúrgico que deben ser encodificadas, se sugiere que se realicen pruebas unitarias como la inserción de la aguja, la visualización de la punta de la aguja, el control de robot y luego, gestionar la integración de cada subetapa para lograr la operatividad del sistema autónomo en términos de precisión.
- Parte de las pruebas unitarias, es la creación de algoritmos y archivos ejecutables para el análisis de trayectoria del robot dentro del entorno ROS; por ello, se sugiere que dichos test se apliquen dentro del sistema operativo Linux y no una máquina virtual. Si bien esta última ofrece una adaptación del entorno Linux ejecutado dentro de Windows, presenta ciertas limitaciones de visualización debido a la ausencia

completa de la gráfica del robot además que la ejecución lo que podría comprometer el desempeño en tiempo real.

- La latencia es un factor altamente crucial en este tipo de intervenciones quirúrgicas ya que la respuesta en tiempo real del sistema define su confiabilidad; por lo que se recomienda que supervisar las etapas de implementación de software para evitar demoras en el análisis de las imágenes, así como en el accionar de los subsistemas controlados por Arduino.
- Para lograr una autonomía completa del sistema, se sugiere que todos los algoritmos sean iniciados bajo una sola ejecución; esto implica que un archivo tipo “launch” agrupe todos los ejecutables del tipo Python y se le brinde todos los argumentos de entrada requeridos para un libre acceso a los paquetes correspondientes de las dependencias de control.
- Desde el punto de vista clínico, la muestra recuperada durante la absorción del líquido debe ser almacenada para su posterior análisis; para identificar si la acumulación de líquido pleural es debido a una producción excesiva del mismo o liberación oportuna del espacio pleural.

REFERENCIAS BIBLIOGRAFICAS

A methodology for comparative analysis of collaborative robots for industry 4.0. (2019). En F. Ferraguti, A. Pertosa, C. Secchi, C. Fantuzzi, & M. Bonfè, *2019 design, automation & Test in europe conference & exhibition* (págs. 1070--1075). IEEE.

Abbyasov, B., Zagirov, A., Gamberov, T., Li, H., & Magid, E. (2024). Vision-based autonomous navigation for medical examination using a UR3e manipulator. En A. R. Ltd., *Proceedings of International Conference on Artificial Life and Robotics* (Vol. 29, págs. 308--311).

Adel, E., Elmogy, M., & Elbakry, H. (2015). Image stitching system based on ORB feature-based technique and compensation blending. *International Journal of Advanced Computer Science and Applications, 6*.

Alexopoulou, E., Prountzos, S., Raissaki, M., Mazioti, A., Caro-Dominguez, P., Hirsch, F. W., . . . Ciet, P. (2024). Imaging of acute complications of community-acquired pneumonia in the paediatric population—from chest radiography to MRI. *children, 11*, 122.

American Accreditation HealthCare Commission. (2022). *Pleural Effusion*. Obtenido de Penn Medicine: https://www.pennmedicine.org/for-patients-and-visitors/patient-information/conditions-treated-a-to-z/pleural-effusion

American Cancer Society. (n.d.). *Toracoscopia*. Obtenido de American Cancer Society: https://www.cancer.org/es/cancer/diagnostico-y-etapa-del-cancer/pruebas/endoscopia/toracoscopia.html#:~:text=de%20la%20toracoscopia-,%C2%BFQu%C3%A9%20es%20una%20toracoscopia?,cirug%C3%ADa%20tor%C3%A1cica%20asistida%20por%20video).

Arduino. (2025). *Documentation*. Obtenido de Arduino: https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf

Arduino. (n.d.). *Arduino Documentation*. Obtenido de Arduino Docs: https://docs.arduino.cc/programming/

Arduino. (n.d.). *Documentation*. Obtenido de Using the Arduino Software (IDE): https://docs.arduino.cc/learn/starting-guide/the-arduino-software-ide/

Arduino. (n.d.). *What is Arduino?* Obtenido de Arduino Documentation : https://docs.arduino.cc/learn/starting-guide/whats-arduino/

Bai, L., Zhao, L., & Ren, H. (2025). Ultrasound guidance and robotic procedures: Actual and future intelligence. En *Handbook of Robotic Surgery* (págs. 103--114). Elservier.

Bankman, I. (2008). *Handbook of medical image processing and analysis.* Elsevier.

Bhutada, S., Yashwanth, N., Dheeraj, P., & Shekar, K. (2022). Opening and closing in morphological image processing. *World Journal of Advanced Research and Reviews, 14*, 687--695.

Bihlmaier, A. B. (2016). ROS-based Cognitive Surgical Robots. *Robot Operating System (ROS) The Complete Reference (Volume 1)*, 317--342.

Boccatonda, A., Baldini, C., Rampoldi, D., Romani, G., Corvino, A., Cocco, G., . . . Serra, C. (2024). Ultrasound-Assisted and Ultrasound-Guided Thoracentesis: An Educational Review. *Diagnostics, 14*, 1124.

Burger, W., & Burge, M. J. (2022). *Digital image processing: An algorithmic introduction.* Springer Nature.

Chitta, S. (2016). MoveIt!: an introduction. In *Robot Operating System (ROS) The Complete Reference (Volume 1)* (pp. 3--27).

Cleveland Clinic. (2023). *Pleural Effusion: Causes, Symptoms, Diagnosis & Treatment*. Obtenido de Cleveland Clinic: https://my.clevelandclinic.org/health/diseases/17373-pleural-effusion

Control Automático Educación. (n.d.). *Cómo controlar un servomotor con PIC usando dos estrategias*. Obtenido de Control Automático Educación: https://controlautomaticoeducacion.com/sistemas-embebidos/microcontroladores-pic/servomotor-con-pic/

Corke, P., Jachimczyk, W., & Pillat, R. (2023). *Robotics, Vision and Control* (3rd ed. ed., Vol. 147). Springer.

DFRobot. (s.f.). *DSS-M15S 270° 15KG DF Metal Servo with Analog Feedback (SKU: SER0044)*. Obtenido de DFRobot: https://wiki.dfrobot.com/DSS-M15S_270%C2%B0_15KG_DF_Metal_Servo_with_Analog_Feedback_SKU__SER0044

Dias, P. A., Souza, J. C., Rocha, L. F., Figueiredo, D., & Silva, M. F. (2024). A ROS-Based Modular Action Server for Efficient Motion Planning in Robotic Manipulators. In *2024 7th Iberian Robotics Conference (ROBOT)* (pp. 1--7). IEEE.

Eaton, M., & Tanveer, M. H. (2024). The Development and Implementation of a Cost-Effective Educational Robotic Arm Using ROS-MoveIT. En *2024 IEEE Integrated STEM Education Conference (ISEC)* (págs. 1--4). IEEE.

El Octavo Bit. (2020). *Funcionamiento del módulo controlador de motores L298N*. Obtenido de El Octavo Bit: https://eloctavobit.com/proyectos-tutoriales-arduino/funcionamiento-del-modulo-controlador-de-motores-l298n

Eletricity - Magnetism. (2024). *What is an H-bridge and how does it work?* Obtenido de Eletricity - Magnetism: https://www.electricity-magnetism.org/what-is-an-h-bridge-and-how-does-it-work/

Elhadidy, M. S., Abdalla, W. S., Abdelrahman, A. A., Elnaggar, S., & Elhosseini, M. (2024). Assessing the accuracy and efficiency of kinematic analysis tools for six-DOF industrial manipulators: The KUKA robot case study. *AIMS Mathematics, 9*, 13944--13979.

Feng, Z., Su, Z., & Qian, Y. (2024). Dual-Criteria Robot Control: Optimization with End-Effector Orientation Constraints. En *Advances in Neural Networks* (págs. 492--501). Springer.

Firgelli Automations. (n.d.). *Actuators: What are they, how they work and different types*. Obtenido de Firgelli Automations: https://www.firgelliauto.com/pages/actuators

Fragapane, G., Hvolby, H.-H., Sgarbossa, F., & Strandhagen, J. (2020). Autonomous mobile robots in hospital logistics. En *IFIP International Conference on Advances in*

Production Management Systems (págs. 672--679). Springer International Publishing.

Franka Emika GmbH. (2021). *Franka Documentation Portal*. Obtenido de Franka Emika: https://download.franka.de/documents/100010_Product%20Manual%20Franka%20Emika%20Robot_10.21_EN.pdf

Franka Emika. (n.d.). *Franka Research 3*. Obtenido de Franka Emika: https://franka.de/products/franka-research-3

Fuller, J. (25 de 04 de 2023). *Linear Actuator – Arduino Tutorial*. Obtenido de Circuits-DIY: https://www.circuits-diy.com/linear-actuator-arduino-tutorial/

Gaz, C., Cognetti, M., Oliva, A., Robuffo Giordano, P., & De Luca, A. (2019). Dynamic Identification of the Franka Emika Panda Robot With Retrieval of Feasible Parameters Using Penalty-Based Optimization. *IEEE Robotics and Automation Letters, 4*.

Gerkey, B., Quigley, M., & Smart, W. (2015). *Programming Robots with ROS: A Practical Introduction to the Robot Operating System* (1 ed.). Sebastopol: O'Reilly Media, Incorporated.

Gobierno del Perú. (2024). *INEN: Solo el 20% de los casos de cáncer de pulmón se diagnostican a tiempo*. Obtenido de Gobierno del Perú: https://www.gob.pe/institucion/minsa/noticias/1059534-inen-solo-el-20-de-los-casos-de-cancer-de-pulmon-se-diagnostican-a-tiempo

Gonnelli, F., Hassan, W., Bonifazi, M., Pinelli, V., Bedawi, E., Porcel, J., . . . Mei, F. (2024). Malignant pleural effusion: current understanding and therapeutic approach. *Respiratory Research, 25*, 47.

Gonzalez, R. C., & Woods, R. E. (2021). *Digital image processing.* Pearson.

Haddadin, S. (2024). The Franka Emika Robot: A Standard Platform in Robotics Research. *IEEE Robotics \& Automation Magazine*.

Hall, M. M., Allen, G. M., Allison, S., Craig, J., DeAngelis, J. P., Delzell, P. B., . . . Tagliafico, A. (2022). Recommended musculoskeletal and sports ultrasound terminology: a

Delphi-based consensus statement. *Journal of Ultrasound in Medicine, 41*, 2395--2412.

Healcerion. (s.f.). *Downloads (Manuals and Guides)*. Recuperado el 2025, de Healcerion USA: https://healcerionusa.com/download/

Iftikhar, M. a. (2024). Artificial intelligence: revolutionizing robotic surgery: review. *Annals of Medicine and Surgery*, 5401--5409.

Instituto Peruano de Economía. (2024). *Mortalidad por cáncer en adultos se incrementó 26% en cinco años*. Obtenido de Instituto Peruano de Economía: https://www.ipe.org.pe/portal/mortalidad-por-cancer-en-adultos-se-incremento-26-en-cinco-anos/#:~:text=Mortalidad%20por%20c%C3%A1ncer%20en%20adultos%20se%20increment%C3%B3%2026%25%20en%20cinco%20a%C3%B1os,-1.&text=En%20el%20primer%20semestre%20del,crecien

Jehangiri, Z. M., Shahzad, M., & Khan, U. (2024). *Digital Image Processing: Advanced Technologies and Applications.* MDPI - Multidisciplinary Digital Publishing Institute.

Joseph, L. (2018). *Robot Operative System (ROS) for Absolute Beginners* (1st ed.).

Kangasniemi, M., Karki, S., Colley, N., & Voutilainen, A. (2019). The use of robots and other automated devices in nurses' work: An integrative review. *International journal of nursing practice, 25*, e12739.

Kenanoglu, C., Le Mesle, V., Luarasi, G., Sadeghian, H., & Haddadin, S. (2024). Design and Evaluation of a Surgical Tool Drive Unit for Sustainable Training in Robot-Assisted Minimally Invasive Surgery. En *2024 10th IEEE RAS/EMBS International Conference for Biomedical Robotics and Biomechatronics (BioRob)* (págs. 1555--1560).

Kimbowa, A., Pieters, A., Tadayon, P., Arora, I., Gulam, S., Pinos, A., . . . Hacihaliloglu, I. (2024). Advancements in needle visualization enhancement and localization methods in ultrasound: a literature review. *Artificial Intelligence Surgery, 4*, 149--169.

Knudsen, J. E., Ghaffar, U., Ma, R., & Hung, A. J. (2024). Clinical applications of artificial intelligence in robotic surgery. *Journal of Robotic Surgery, 18*.

Kong, Y., Yang, L., Chen, C., Zhu, X., Li, D., Guan, Q., & Du, G. (2024). Online kinematic calibration of robot manipulator based on neural network. *Measurement, 238*.

Koubaa, A. (2023). *Robot Operating System (ROS)* (Vol. 7). Springer Cham. doi:https://doi.org/10.1007/978-3-031-09062-2

Kyoto Kagaku. (s.f.). *MW4 Thoracentesis Model*. Recuperado el 2025, de Kyoto Kagaku Co., Ltd.: https://www.kyotokagaku.com/en/products_data/mw4/

Lee, A., Baker, T., Bederson, J., & Rapoport, B. (2024). Levels of autonomy in FDA-cleared surgical robots: a systematic review. *NPJ Digital Medicine, 7*, 103.

Liu, Y., Wu, X., Sang, Y., Zhao, C., Wang, Y., Shi, B., & Fan, Y. (2024). Evolution of Surgical Robot Systems Enhanced by Artificial Intelligence: A Review. *Advanced Intelligent Systems, 6*, 2300268.

LME Editorial Staff. (n.d). *Interface L298N DC Motor Driver Module with Arduino*. Obtenido de LastMinuteEngineers: https://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/#google_vignette

MoveIt Tutorials. (n.d.). *MoveIt tutorials*. Obtenido de MoveIt: https://moveit.github.io/moveit_tutorials/

Murphy, R. R. (2019). *Introduction to AI robotics.* MIT press.

Nicholson, M. J., Manley, C., & Ahmad, D. (2023). Thoracentesis for the diagnosis and management of pleural effusions: the current state of a centuries-old procedure. *Journal of Respiration, 3*, 208--222.

Niku, S. (2020). *Introduction to robotics: analysis, control, applications.* John Wiley \& Sons.

Nixon, M., & Aguado, A. (2012). *Feature extraction & image processing for computer vision* (3rd ed.). Oxford Academic.

Open Robotics. (n.d.). *ROS Concepts*. Obtenido de ROS Wiki: https://wiki.ros.org/ROS/Concepts

Open Robotics. (n.d.). *Writing a Simple Publisher and Subscriber (Python)*. Obtenido de Wiki ROS: https://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29

Paulsen, R. R., & Moeslund, T. B. (2020). *Introduction to Medical Image Analysis* (1st ed.). Springer Nature.

Peng, G., Hu, C., Yao, Y., Liu, J., Yang, F., & Lam, T. (2023). *Introduction to Intelligent Robot System Design: Application Development with ROS* (1 ed.). Springer. doi:10.1007/978-981-99-1814-0

Raji, A., Gopaul, U., Babineau, J., Popovic, M. R., & Marquez-Chin, C. (2025). Industrial-grade collaborative robots for motor rehabilitation after stroke and spinal cord injury: a systematic narrative review. *BioMedical Engineering OnLine, 24*, 50.

rgb2gray. (n.d.). Obtenido de Mathworks: https://www.mathworks.com/help/matlab/ref/rgb2gray.html

Robotics Knowledgebase. (2023). *Robotics Knowledgebase*. Obtenido de Robotics Project Guide – Choose a Robot: https://roboticsknowledgebase.com/wiki/robotics-project-guide/choose-a-robot/

Robotics, O. (2018). *ROS/Introduction*. Obtenido de ROS Wiki: https://wiki.ros.org/ROS/Introduction

Rviz, M. i. (s.f.). *Moveit*. Obtenido de https://moveit.github.io/moveit_tutorials/

Sachan, S., & Swarnkar, P. (2024). Intelligent fractional-order sliding mode optimised control of surgical manipulator for healthcare system. *Electrical Engineering, 106*, 2131--2142.

Seenivasan, L., & Ren, H. (2025). Artificial intelligence to understand robotic surgery scenes. En *Handbook of Robotic Surgery* (págs. 179--188). Elsevier.

Sehmbi, H., & Perlas, A. (2022). Basics of ultrasound imaging. En *Regional Nerve Blocks in Anesthesia and Pain Therapy: Imaging-guided and Traditional Techniques* (págs. 33--52). Springer.

Selby, K. (2025). *Pleurodesis for Mesothelioma: How a Pleurodesis Can Help You*. Obtenido de Asbestos.com: https://www.asbestos.com/treatment/surgery/pleurodesis/

Selby, K. (2025). *Pleural effusion and mesothelioma: Symptoms, causes & treatments*. Obtenido de The Mesothelioma Center at Asbestos.com: https://www.asbestos.com/mesothelioma/pleural-effusion/

Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2008). *Robotics: Modelling, Planning and Control.* Springer London.

Silvera-Tawil, D. (2024). Robotics in Healthcare: A Survey. *SN Computer Science*.

Singh, A. P. (2019). *Med Care Tips*. Obtenido de Health & Medical Care: https://medcaretips.com/thoracentesis/

Solomon, C., & Breckon, T. (2011). Image Segmentation. En *Fundamentals of Digital Image Processing: A practical approach with examples in Matlab* (2nd ed.). Wiley-Blackwell.

Solomon, C., & Breckon, T. (2011). Representation. En *Fundamentals of Digital Image Processing* (2nd ed.). Chichester: Wiley-Blackwell.

Song, Q., & Qinglei, Z. (2024). *Recent Advances in Robotics and Intelligent Robots Applications.* MDPI - Multidisciplinary Digital Publishing Institute.

Subasi, A., Qaisar, S. M., & Nisar, H. (2025). Artificial intelligence techniques for human-machine interaction. En *Artificial Intelligence and Multimodal Signal Processing in Human-Machine Interaction* (págs. 19-42). London: Elsevier.

Szeliski, R. (2022). *Computer vision : algorithms and applications* (Second edition. ed.). Springer.

Talli, A., & Giriyapur, A. C. (2021). Kinematic analysis and simulation of industrial robot based on RoboAnalyzer. En *Recent Advances in Mechanical Infrastructure: Proceedings of ICRAM 2020* (págs. 473--483). Springer.

The Construct. (n.d.). *ROS basics in 5 days (Python)*. Obtenido de https://www.theconstructsim.com/

Trabalza Marinucci, B., Tiracorrendo, M., Vanni, C., Messa, F., Piccioni, G., Siciliani, A., . . . D'Andrilli, A. (2025). Robotic Versus Sternotomy, Thoracotomy and Video-Thoracoscopy Approaches for Thymoma Resection: A Comparative Analysis of Short-Term Results. *Journal of Personalized Medicine, 15*, 34.

Tran, A., & Gulati, A. (2022). Basics of Ultrasound. En *Regenerative Medicine: A Complete Guide for Musculoskeletal and Spine Disorders* (págs. 103--113). Springer.

Varghese, C., Harrison, E. M., O'Grady, G., & Topol, E. J. (2024). Artificial intelligence in surgery. *Nature Medicine*, 1--12.

Wang, Y., Li, Y., Wang, J., Lv, H., & Yang, Z. (2022). A Target Corner Detection Algorithm Based on the Fusion of FAST and Harris. *Mathematical Problems in Engineering*, 4611508.

World Health Organization. (2022). *International Agency for Research on Cancer*. Obtenido de World Health Organization: https://gco.iarc.fr/today/en/dataviz/bars-compare-populations?mode=cancer&populations=604&group_populations=1&sort_by=value0&types=0

Yang, L. W. (2024). Chest ultrasound is better than CT in identifying septated efusion of patients with pleural disease. (N. P. London, Ed.) *Scientific Reports, 14*(1), 11964.

Yasmini, L. B., Artha, I., & Gunadi, I. (2025). Rigid Body Motion: The End Effector of 4-DoF Robot Motion Analysed by Denavit Hartenberg Method. En *Proceeding of International Joint Conference on UNESA.*

Zaki, H. A. (2024). Advancement in pleura efusion diagnosis a systematic review and meta-analysis of point-of-care ultrasound versus radiographic thoracic imaging. (Springer, Ed.) *The Ultrasound Journal, 16*, 3. doi:https://doi.org/10.1186/s13089-023-00356-z

Zhang, S., & Metaxas, D. (2024). On the challenges and perspectives of foundation models for medical image analysis. *Medical image analysis, 91*, 102996.

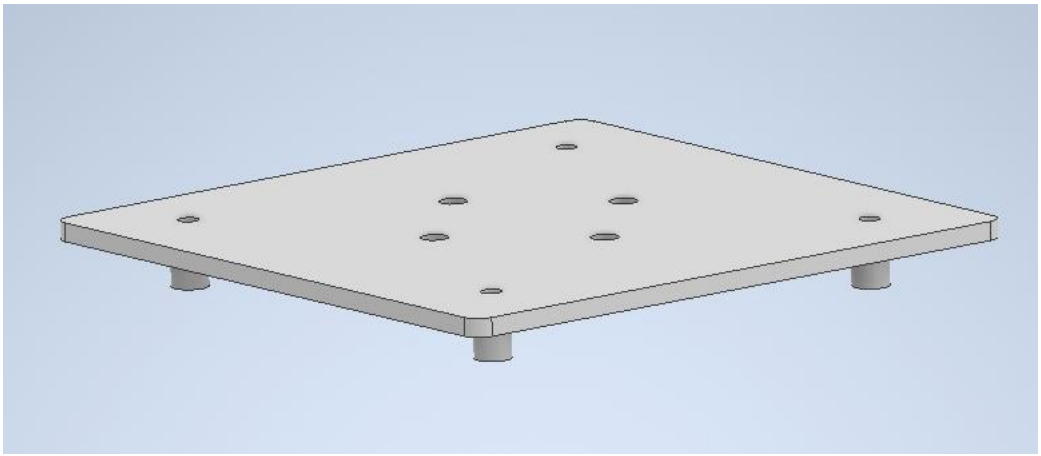
Zhao, Y., Yu, L., Wang, L., Wu, Y., Chen, H., Wang, Q., & Wu, Y. (s.f.). Current status of and progress in the treatment of malignant pleural effusion of lung cancer. *Frontiers in oncology, 12*, 2023.

ANEXOS

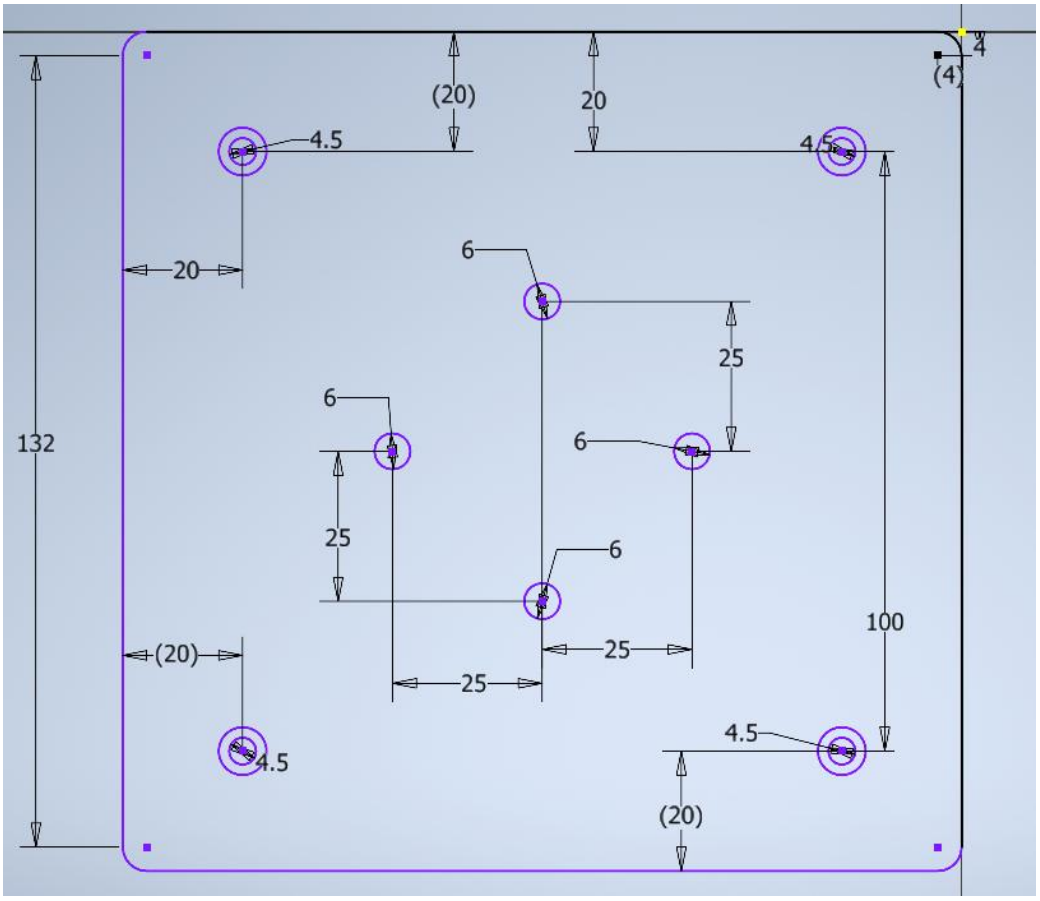
ANEXO A: ELEMENTOS DEL DISEÑO DEL MÓDULO ROBÓTICO

1ra plataforma de soporte

Vista Isométrica

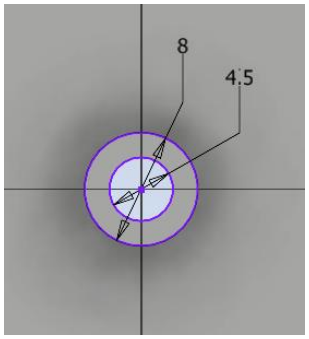


Vista Horizontal



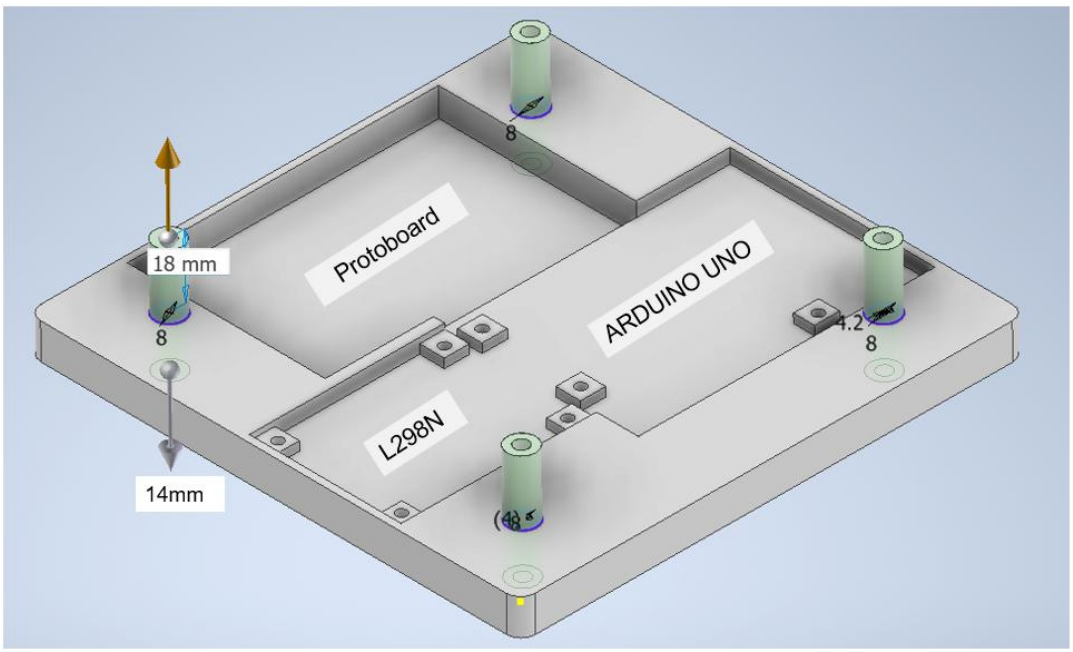
Vista Lateral

*Vista del plano **B-B***

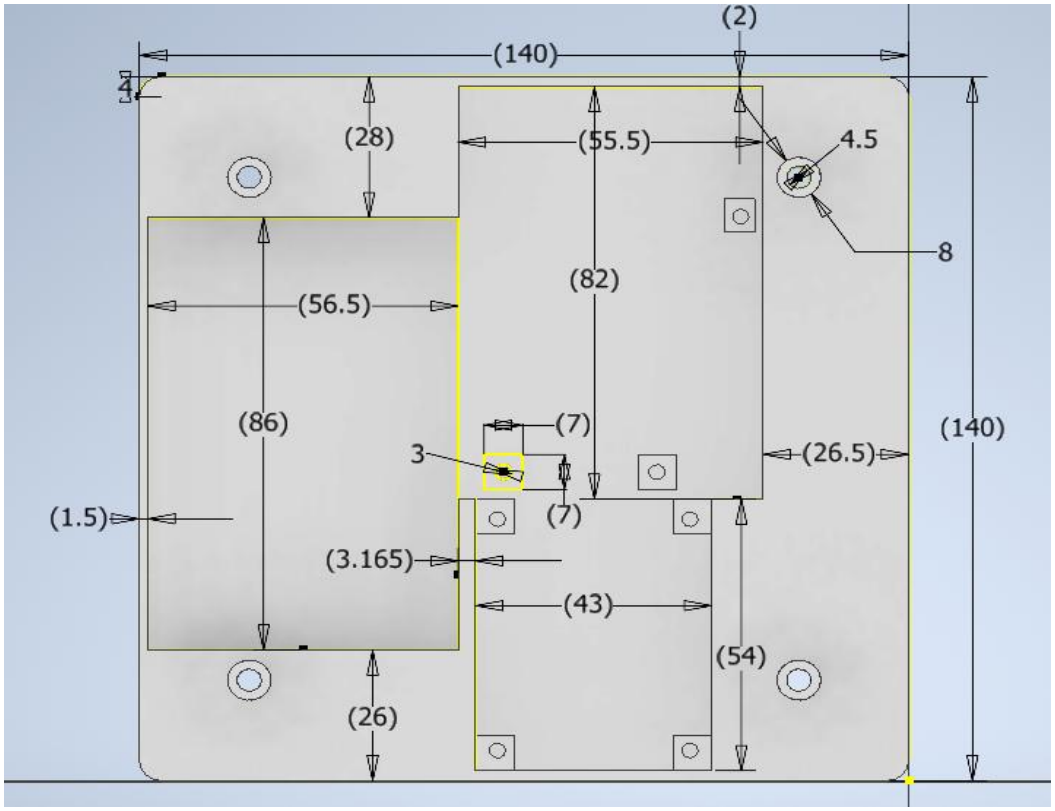


2da plataforma de soporte

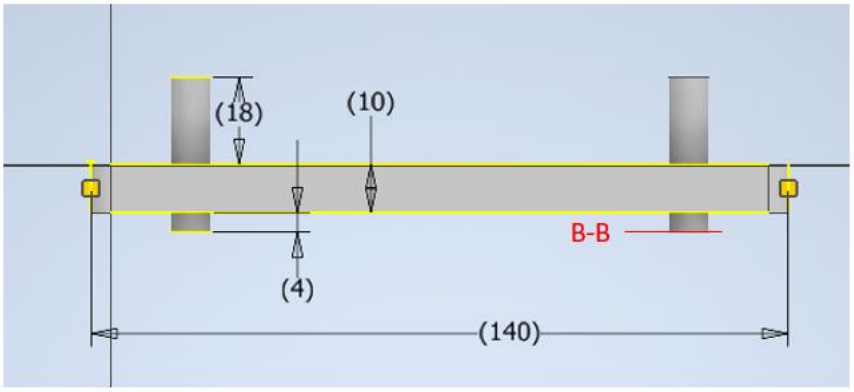
Vista Isométrica.- Los espacios etiquetados como ARDUINO UNO, L298N son calculados según su documentación para obtener sus medidas reales.



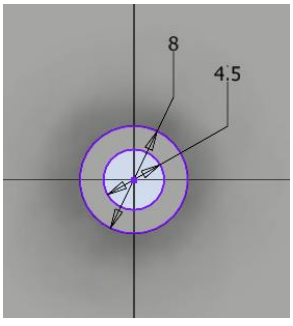
Vista Horizontal



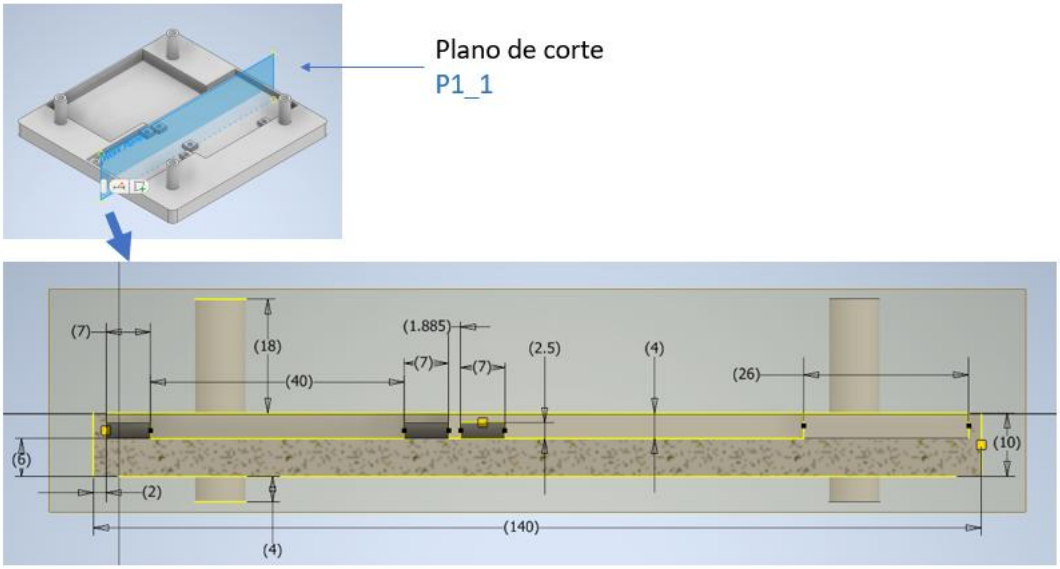
Vista Lateral



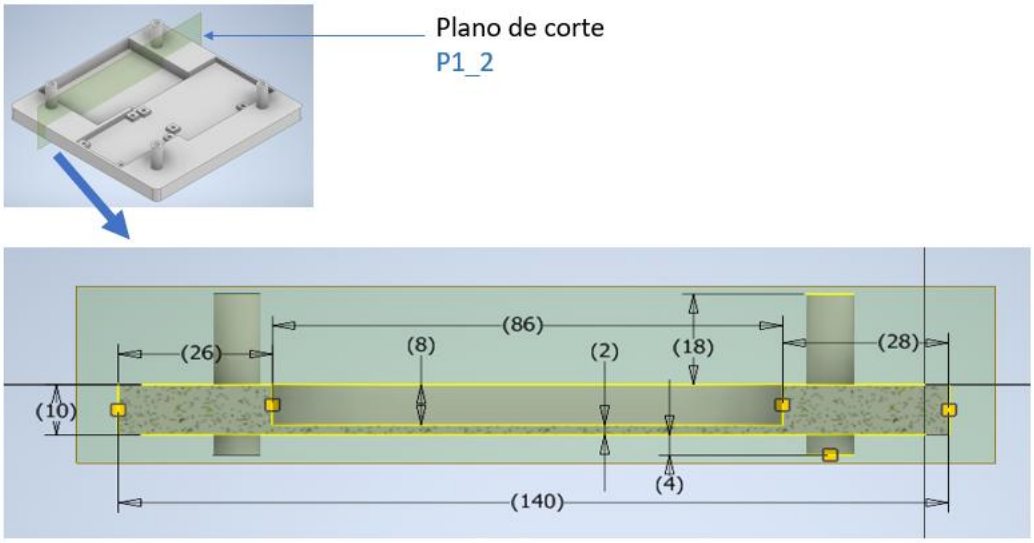
*Vista del plano **B-B***



Vista de corte P1_1

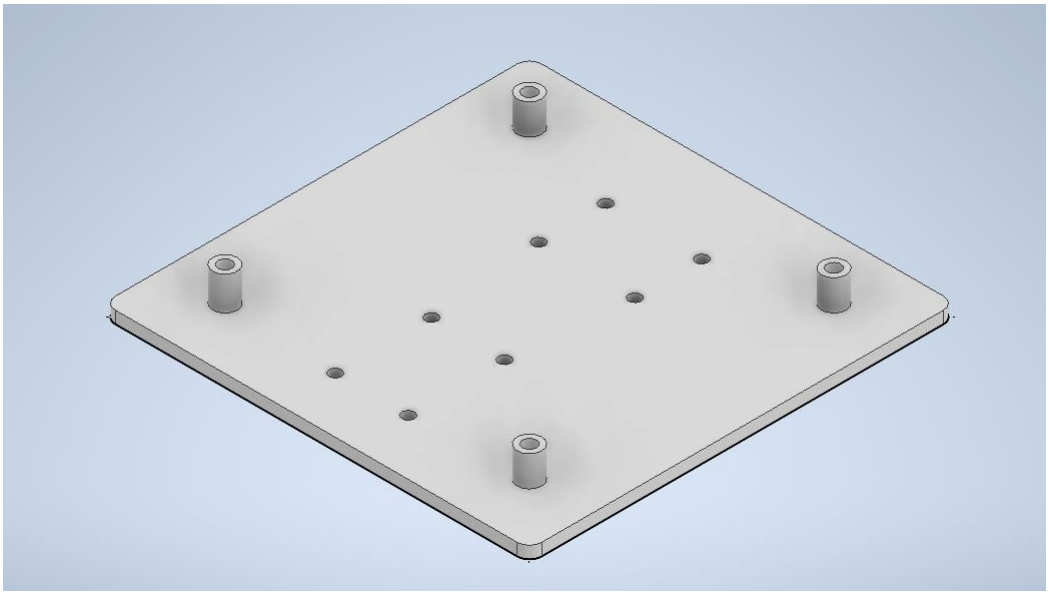


Vista de corte P1_2

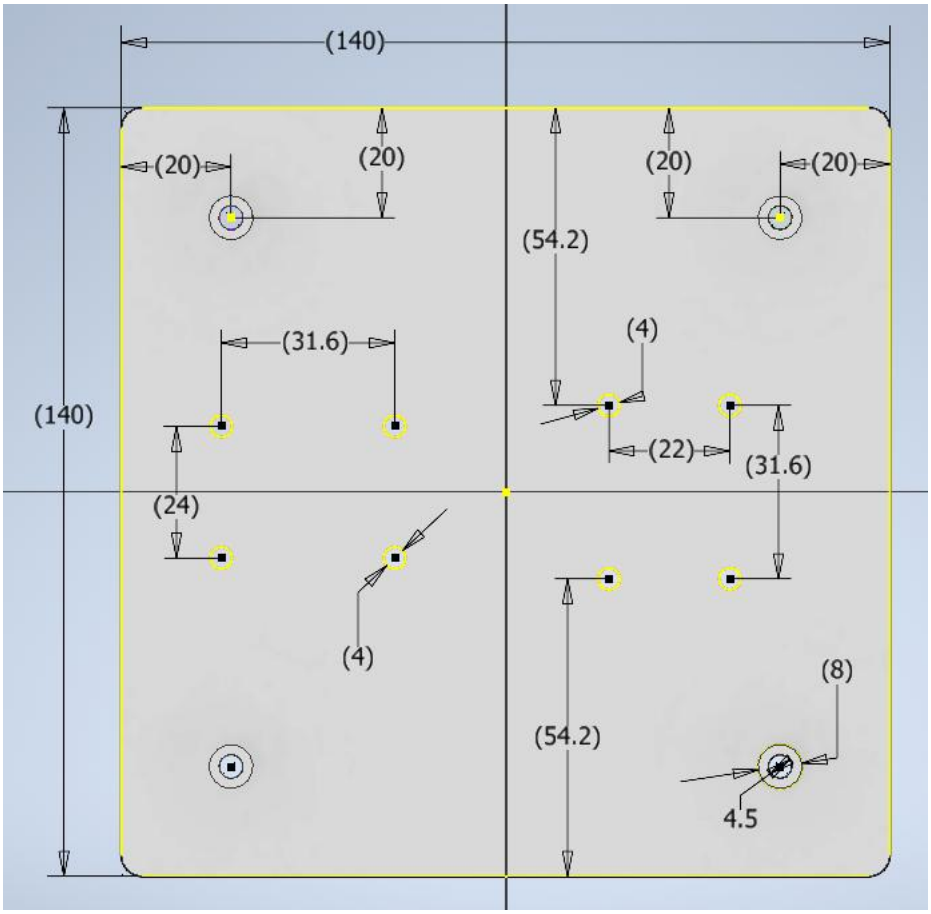


3ra plataforma de soporte

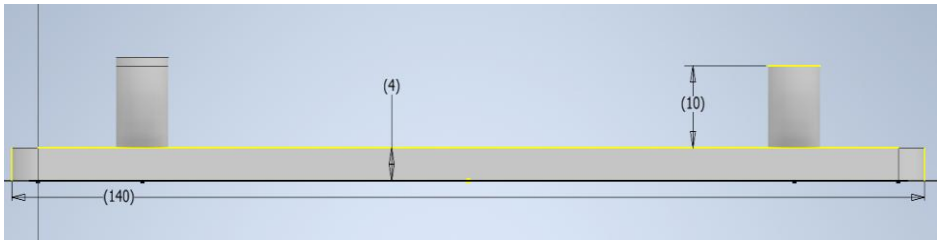
Vista Isométrica



Vista Horizontal

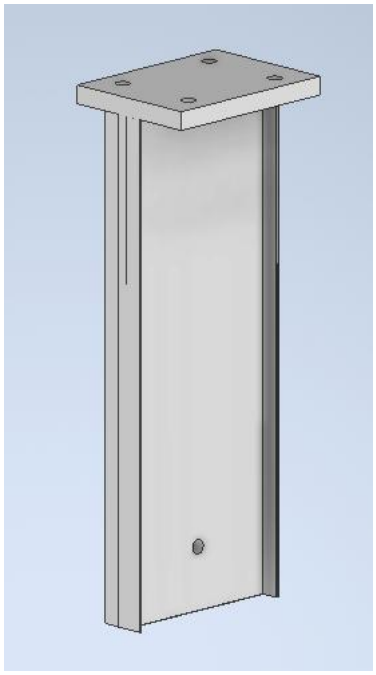


Vista Lateral

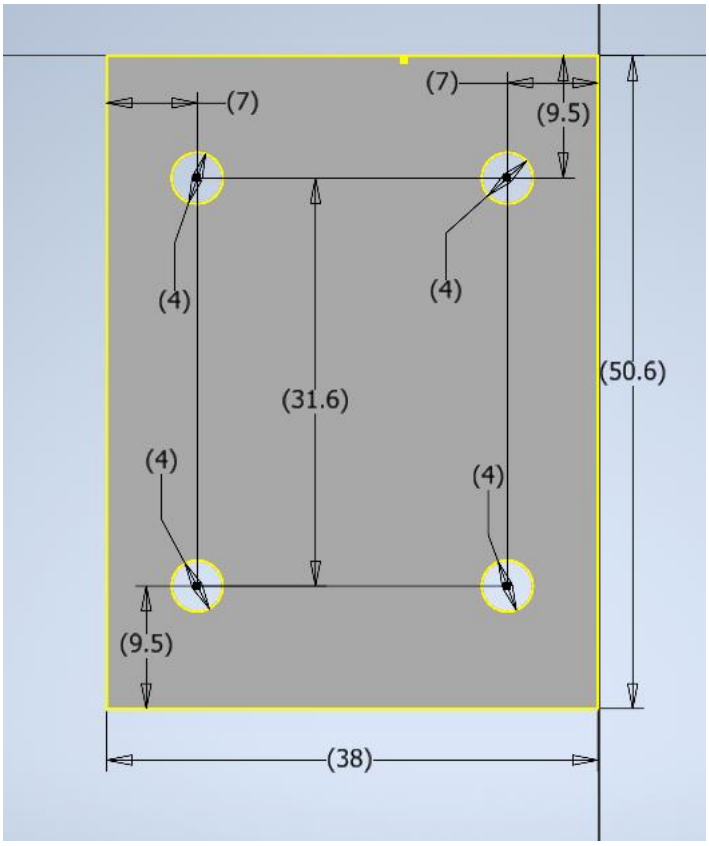


Soporte para el escáner de ultrasonido (Subsistema A)

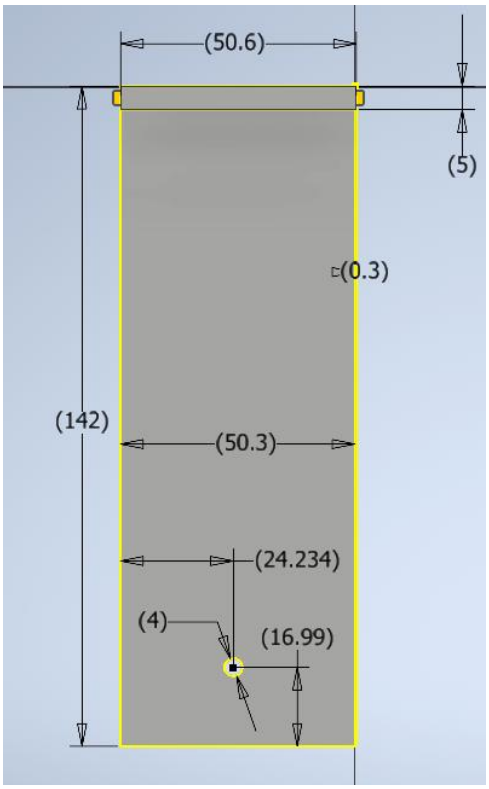
Vista Isométrica



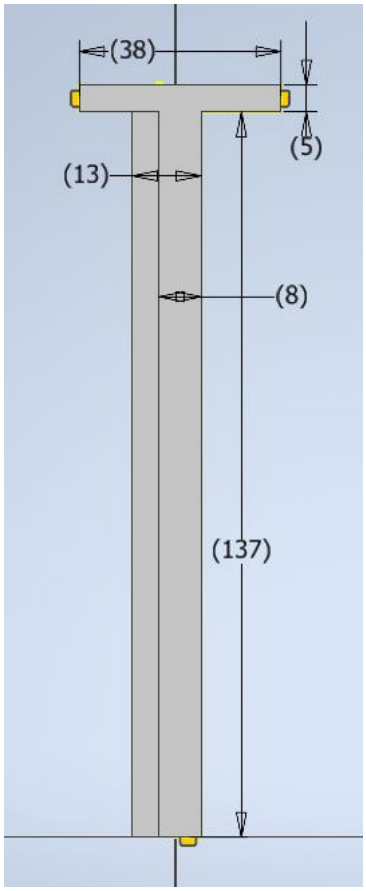
Vista Horizontal



Vista Frontal

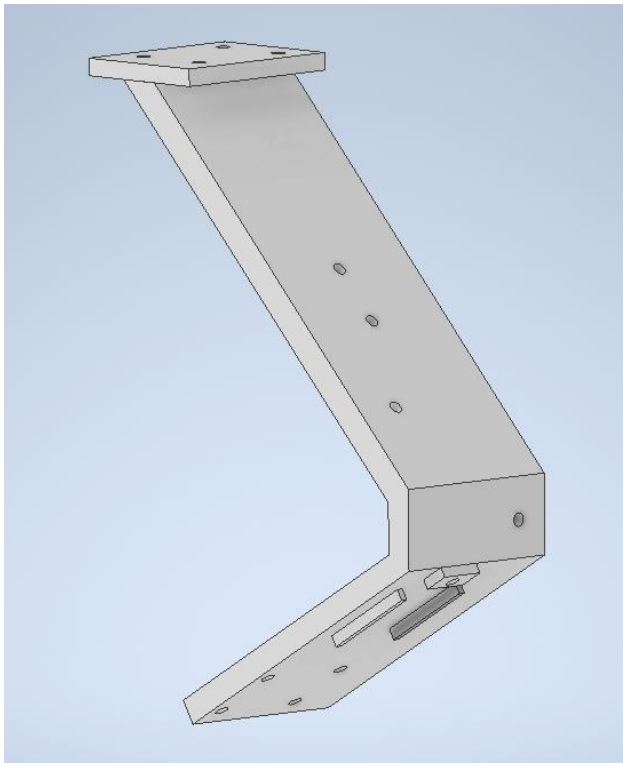


Vista Lateral

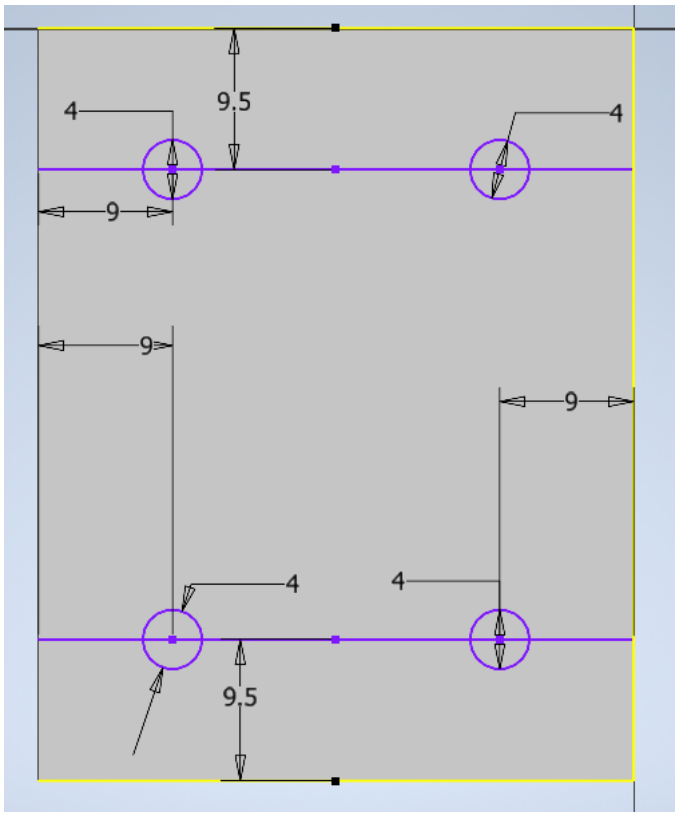


Soporte para el subsistema B

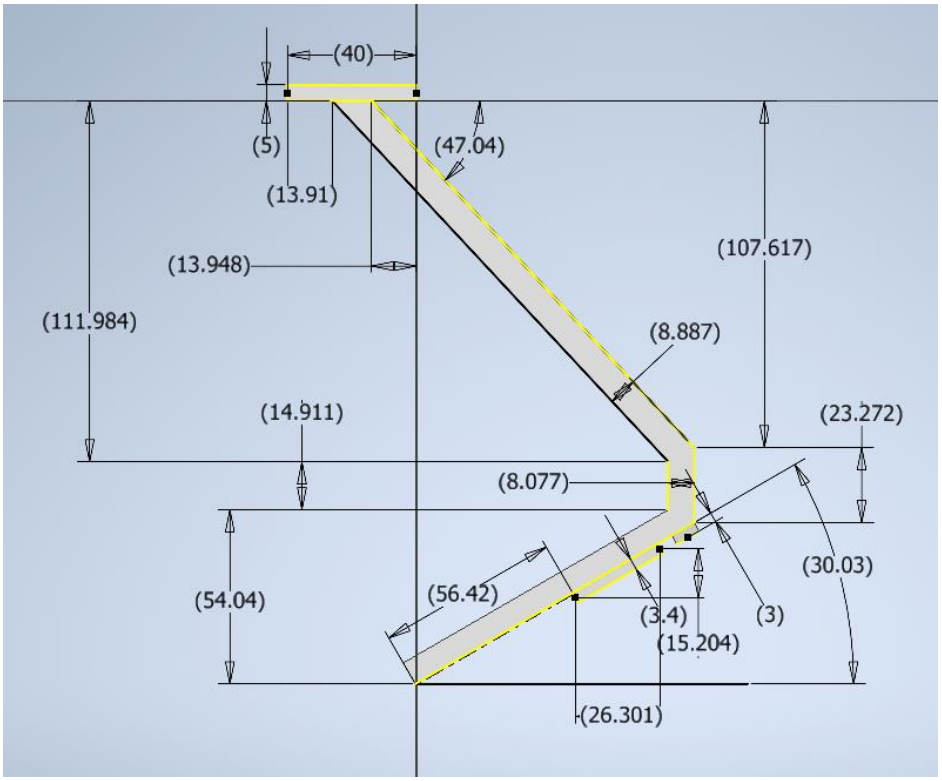
Vista Isométrica



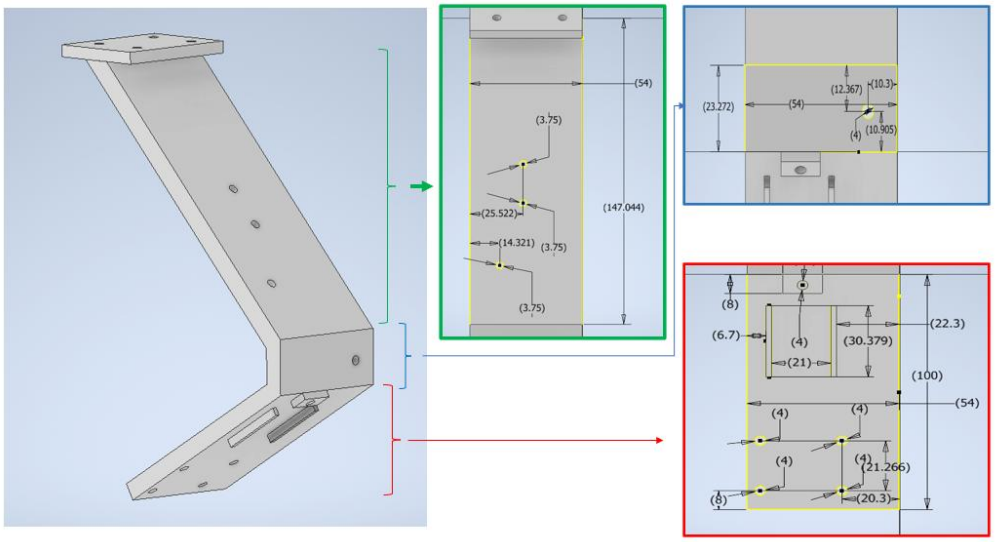
Vista Horizontal



Vista Frontal

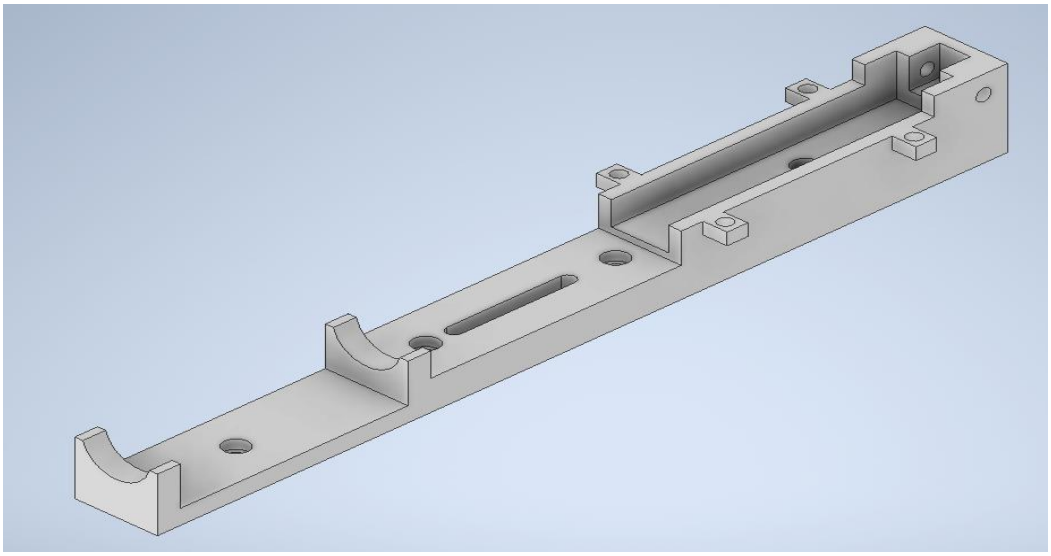


Vista Isométrica

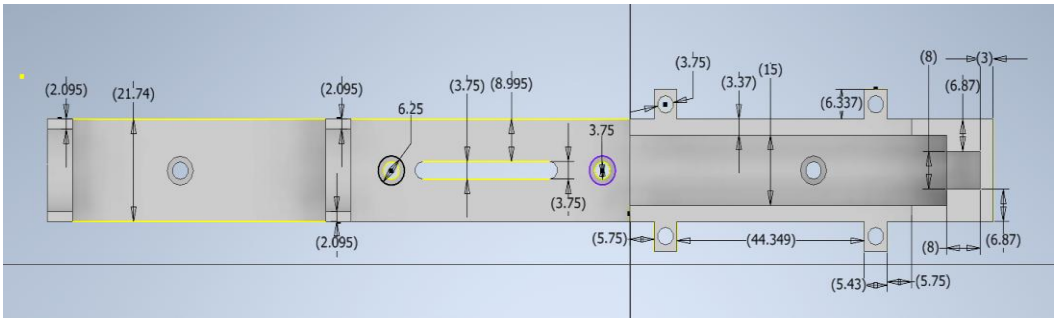


Soporte para el subsistema C

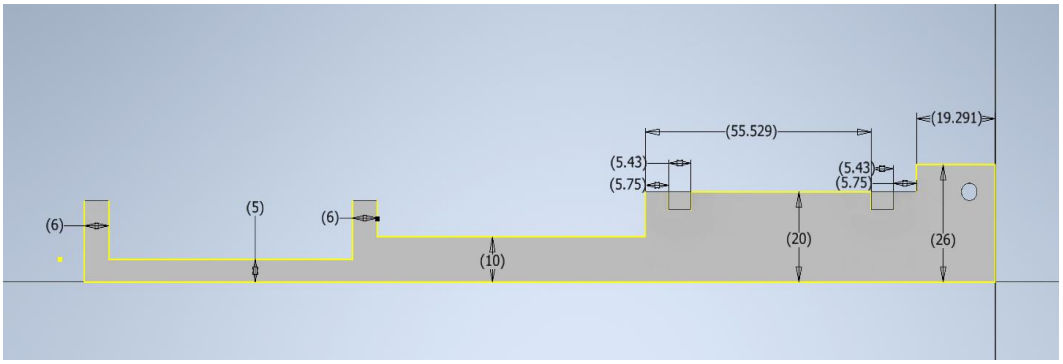
Vista Isométrica



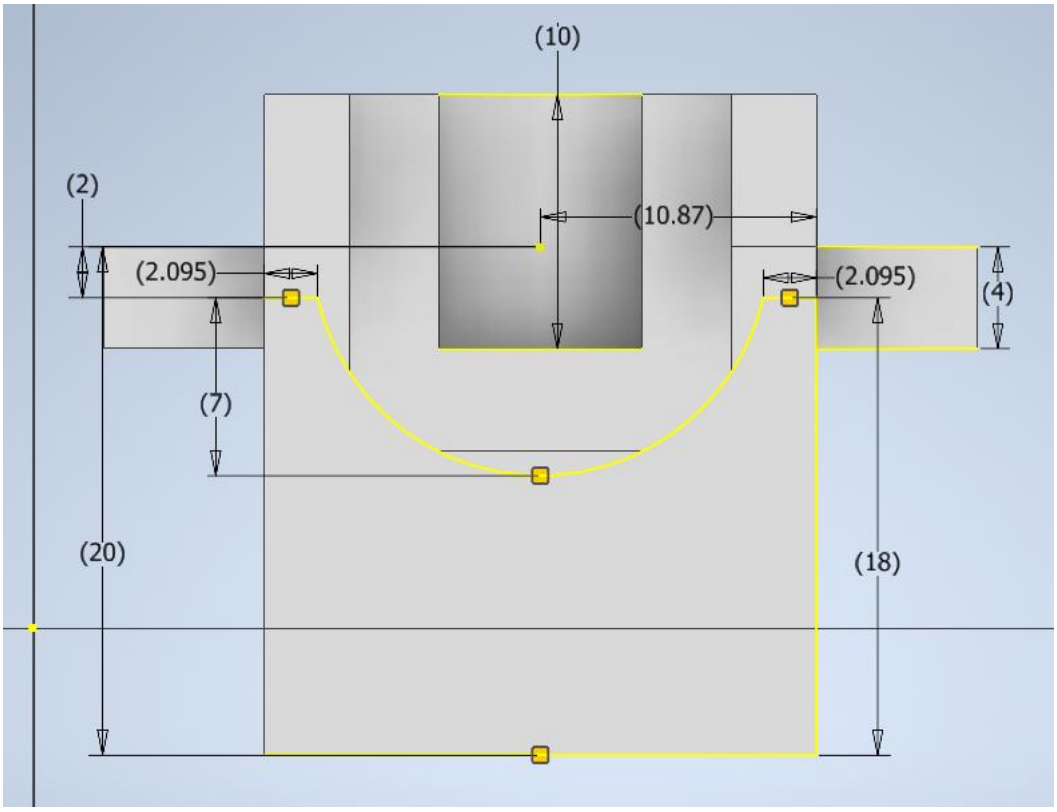
Vista Horizontal



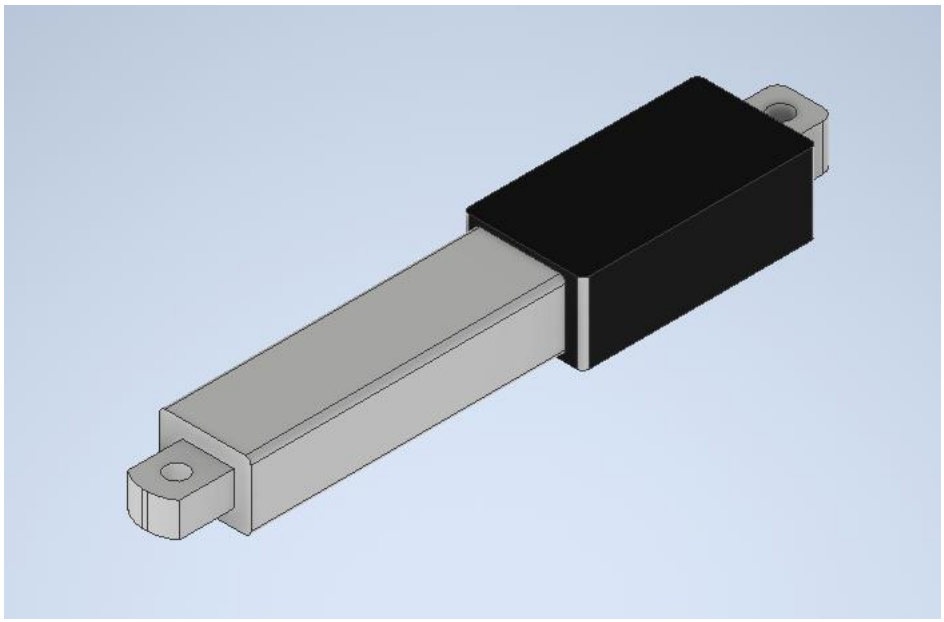
Vista Lateral



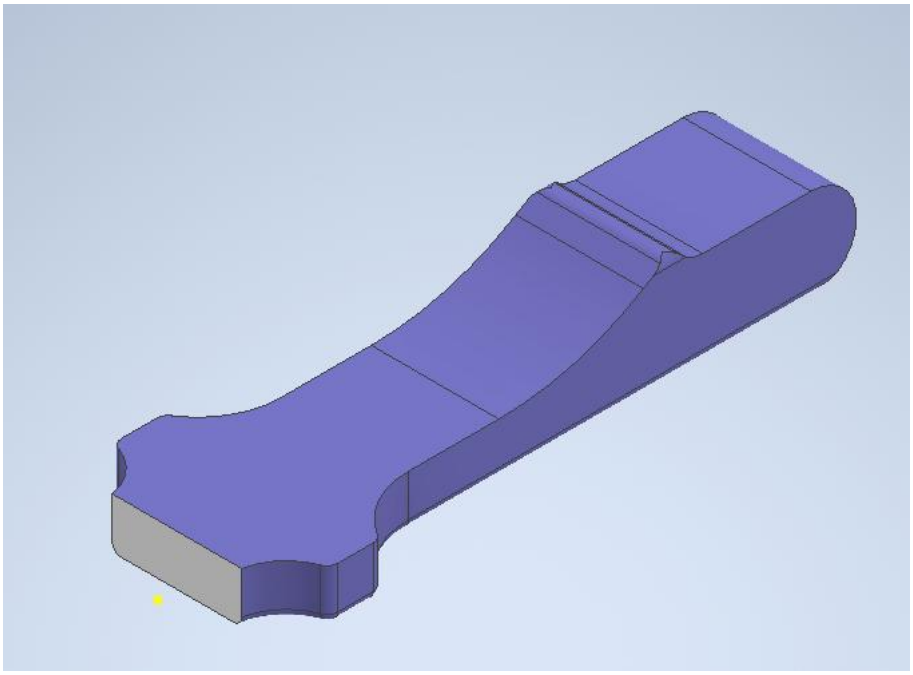
Vista Frontal



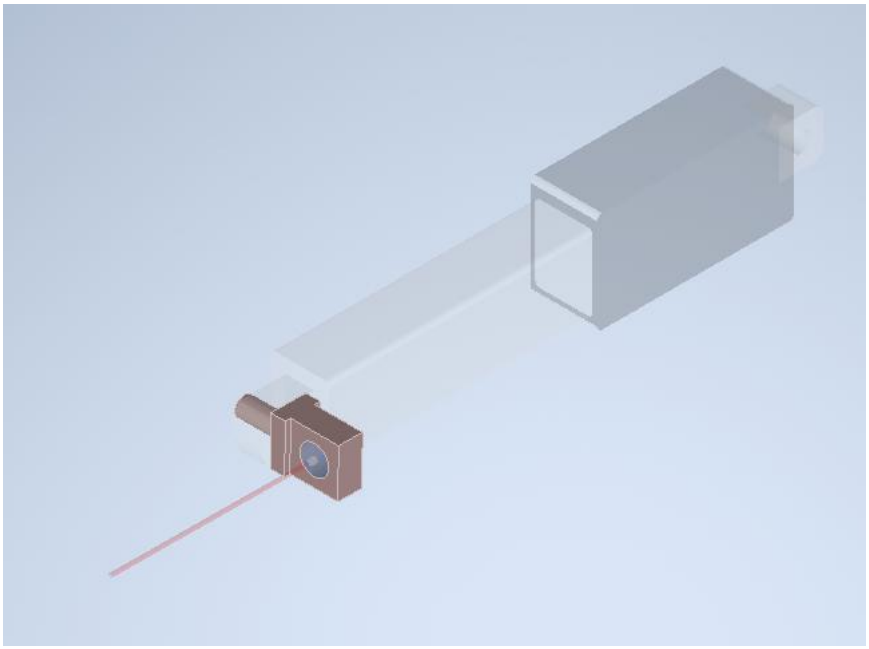
Actuador Lineal



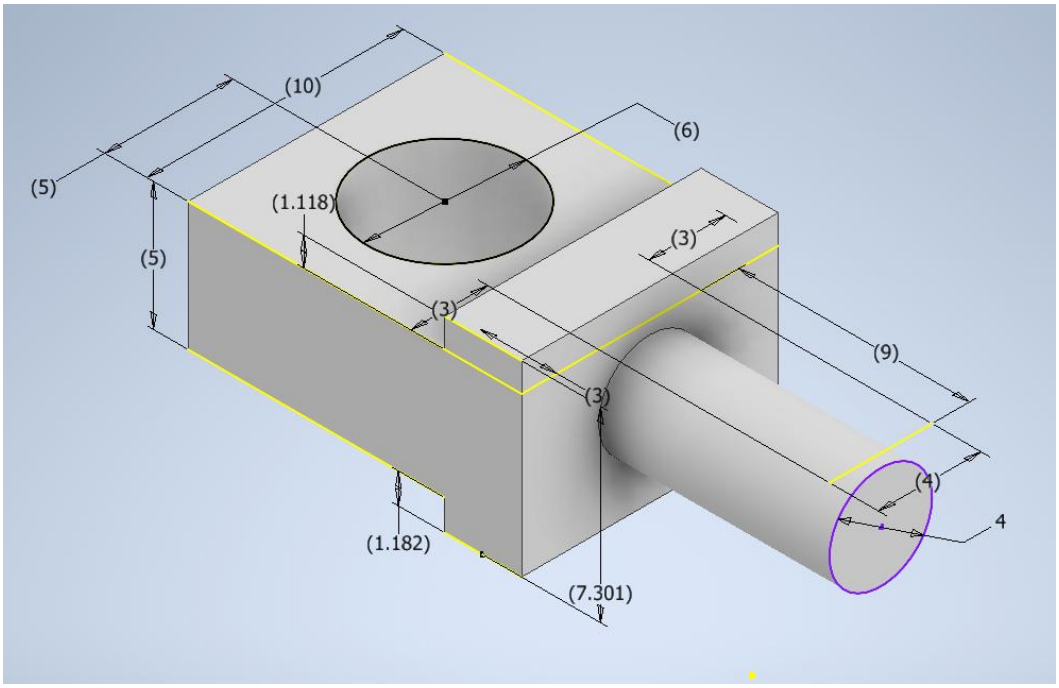
Escáner Ultrasonido



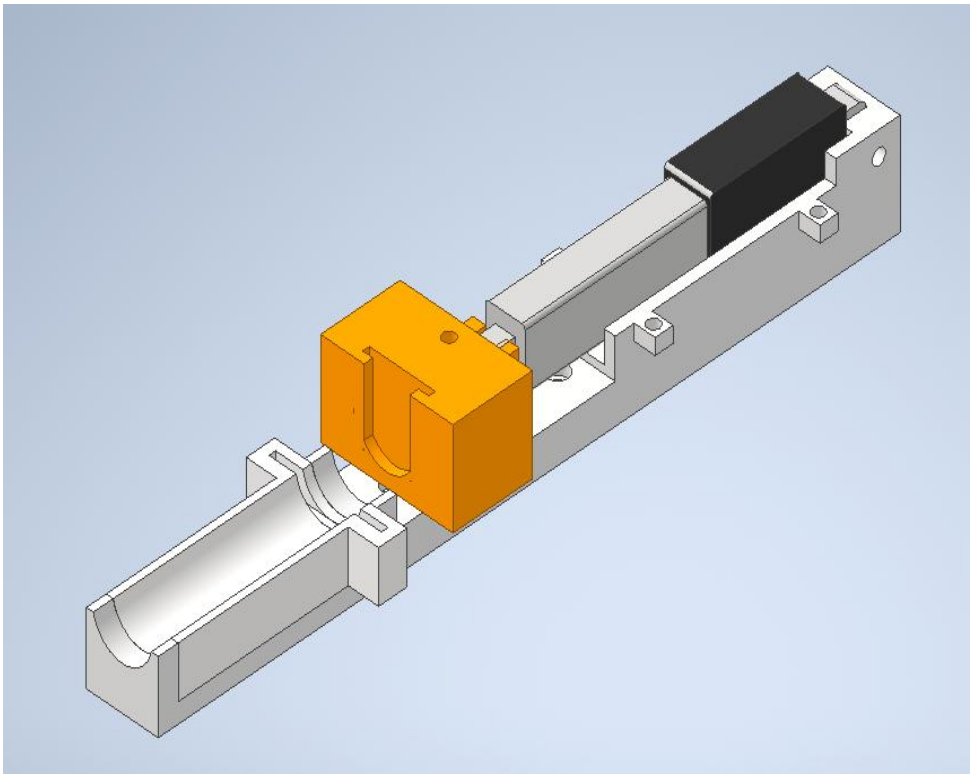
Acople para sujetar aguja a actuador lineal.



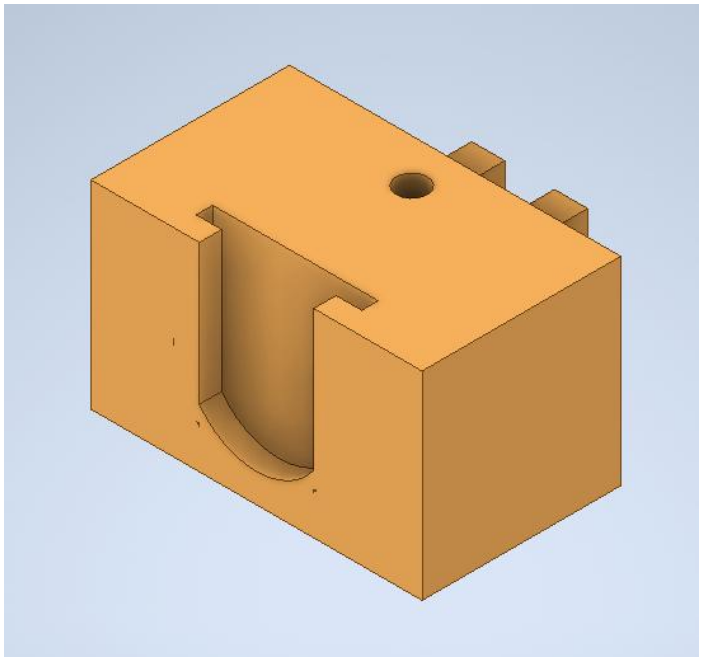
Vista Isométrica con medidas



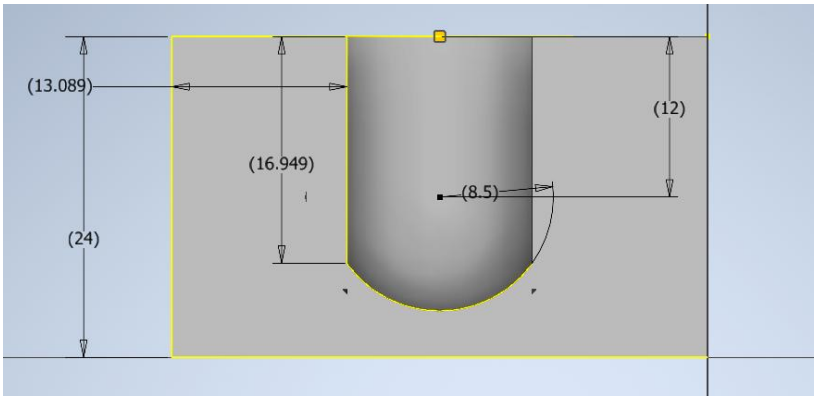
Acople para sujetar pistón de jeringa a actuador lineal (Subsistema C).



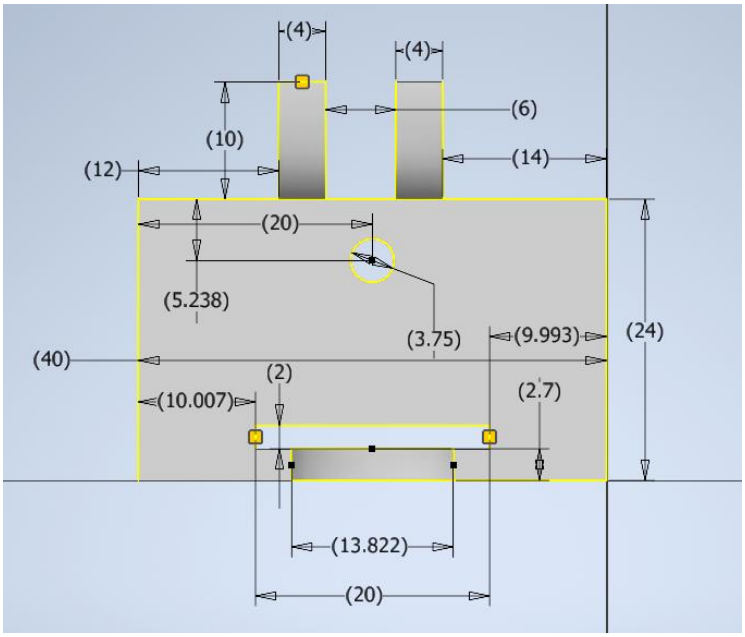
Vista Isométrica



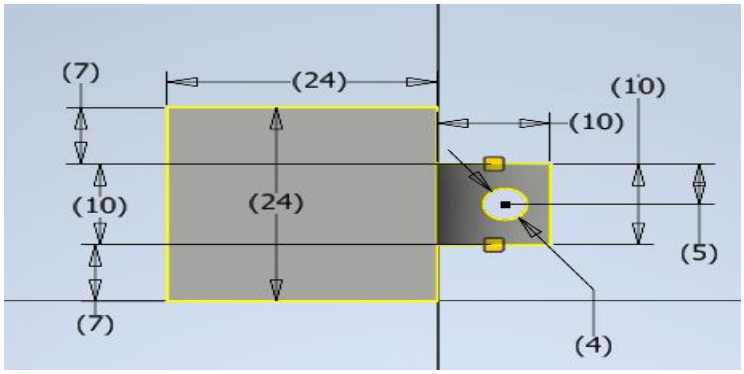
Vista Frontal



Vista Horizontal



Vista Perfil



ANEXO B: INSTALACIÓN DE PAQUETES DE CONTROL DE ROBOT

Principales comandos usados en ROS

roscore : activa el marco de trabajo al gestionar comunicación de nodos.

roslaunch : ejecutar archivos launch para ejecución simultanea de archivos.

rosrun : ejecución del código de un archivo python o c++

ls : mostrar archivos

cd : cambiar de directorio

Catkin_create_pkg: crear el espacio de trabajo.

catkin_make: compila el espacio de trabajo.

mkdir : crear folders.

touch : crear archivos de código (python, C++, launch).

chmod +x : convertir en ejecutable a los archivos de código

code . : abrir entorno de desarrollo (visual studio)

scrcpy : reproducir función espejo mediante cable USB (Android -> Windows)

tree : visualización desplegada de paquetes, carpetas y archivos.

ANEXO C: SECUENCIA DE ACTIVACIÓN DE LOS ALGORITMOS

```
<launch>
        <arg name="robot_ip" default="192.168.0.1" />
        <node name="actuator_command_node"
                pkg="my_panda"
                type="actuator_command_node.py"
                output="screen"/>
        <node name="window_capturer"
                pkg="my_panda"
                type="artificial_vision.py"
                output="screen"
                launch-prefix="bash -c 'sleep 10; exec "'/>

        <include                                                        file="$(find
        panda_moveit_config)/launch/panda_control_moveit_rviz.launch">
          <arg name="robot_ip" value="$(arg robot_ip)" />
        </include>

</launch>
```

ANEXO D: CÓDIGO DE CONTROL DE ROBOT PANDA EN ROS

```
#!/usr/bin/env python3
# Python 2/3 compatibility imports
from __future__ import print_function
from six.moves import input

import sys
import copy
import rospy
import moveit_commander
import moveit_msgs.msg
import geometry_msgs.msg

try:
    from math import pi, tau, dist, fabs, cos
except:  # For Python 2 compatibility
    from math import pi, fabs, cos, sqrt

    tau = 2.0 * pi

    def dist(p, q):
        return sqrt(sum((p_i - q_i) ** 2.0 for p_i, q_i in zip(p, q)))

from std_msgs.msg import String
from moveit_commander.conversions import pose_to_list



# Método para testear si el valor actual está dentro de la tolerancia al valor objetivo
```

```
def all_close(goal, actual, tolerance):
    if type(goal) is list:
        for index in range(len(goal)):
            if abs(actual[index] - goal[index]) > tolerance:
                return False

    elif type(goal) is geometry_msgs.msg.PoseStamped:
        return all_close(goal.pose, actual.pose, tolerance)

    elif type(goal) is geometry_msgs.msg.Pose:
        x0, y0, z0, qx0, qy0, qz0, qw0 = pose_to_list(actual)
        x1, y1, z1, qx1, qy1, qz1, qw1 = pose_to_list(goal)
        # Distancia Euclidiana
        d = dist((x1, y1, z1), (x0, y0, z0))
        # phi = ángulo entre orientaciones
        cos_phi_half = fabs(qx0 * qx1 + qy0 * qy1 + qz0 * qz1 + qw0 * qw1)
        return d <= tolerance and cos_phi_half >= cos(tolerance / 2.0)

    return True

class MoveGroupPythonInterfaceTutorial(object):
    def __init__(self):
        super(MoveGroupPythonInterfaceTutorial, self).__init__()
        moveit_commander.roscpp_initialize(sys.argv)
        rospy.init_node("move_group_python_interface_tutorial", anonymous=True)
        robot = moveit_commander.RobotCommander()
        scene = moveit_commander.PlanningSceneInterface()
        group_name = "panda_arm"
```

```
move_group = moveit_commander.MoveGroupCommander(group_name)
display_trajectory_publisher = rospy.Publisher(
    "/move_group/display_planned_path",
    moveit_msgs.msg.DisplayTrajectory,
    queue_size=20,
)

# Nombre del elemento de referencia:
planning_frame = move_group.get_planning_frame()
print("****Link de referencia: %s" % planning_frame)

# Nombre del efector final :
eef_link = move_group.get_end_effector_link()
print("*** Link del efector final: %s" % eef_link)

# Lista de los grupos del robot:
group_names = robot.get_group_names()
print("** Grupos disponibles del robot:", robot.get_group_names())

# Descripción del estado integral del robot
print("* Estado completo del robot")
print(robot.get_current_state())
print("")

# Variables
self.box_name = ""
self.robot = robot
self.scene = scene
```

```
        self.move_group = move_group
        self.display_trajectory_publisher = display_trajectory_publisher
        self.planning_frame = planning_frame
        self.eef_link = eef_link
        self.group_names = group_names

    def pose_state(self):
        move_group = self.move_group
        current_pose = self.move_group.get_current_pose().pose
        print(current_pose)
        print("")
        return all_close(current_pose, current_pose, 0.01)

    def go_to_joint_state(self):
        move_group = self.move_group

        joint_goal = move_group.get_current_joint_values()
        joint_goal[0] = 0
        joint_goal[1] = 0
        joint_goal[2] = 0
        joint_goal[3] = -tau / 8
        joint_goal[4] = 0
        joint_goal[5] = tau / 8  # 1/8 de vuelta
        joint_goal[6] = tau / 8
        move_group.go(joint_goal, wait=True)

        # Stop asegura que no se reproduzca movimiento residual
        move_group.stop()
```

```
    # Testeo:
    current_joints = move_group.get_current_joint_values()
    return all_close(joint_goal, current_joints, 0.01)

def go_to_pose_goal(self):
   move_group = self.move_group
    pose_goal = geometry_msgs.msg.Pose()
    pose_goal.orientation.w = 1.0
    pose_goal.position.x = 0.4
    pose_goal.position.y = 0.1
    pose_goal.position.z = 0.4

    move_group.set_pose_target(pose_goal)
    success = move_group.go(wait=True)
    move_group.stop()
    move_group.clear_pose_targets()
    current_pose = self.move_group.get_current_pose().pose
    return all_close(pose_goal, current_pose, 0.01)

def plan_cartesian_path(self, scale=1):
    move_group = self.move_group
    waypoints = []
    wpose = move_group.get_current_pose().pose
    wpose.position.z -=  0.4 #First move up (z)
    waypoints.append(copy.deepcopy(wpose))

    (plan, fraction) = move_group.compute_cartesian_path(
        waypoints, 0.01  # waypoints to follow  # eef_step)
```

```
        return plan, fraction

    def display_trajectory(self, plan):
        robot = self.robot
        display_trajectory_publisher = self.display_trajectory_publisher

        display_trajectory = moveit_msgs.msg.DisplayTrajectory()
        display_trajectory.trajectory_start = robot.get_current_state()
        display_trajectory.trajectory.append(plan)
        # Publicar
        display_trajectory_publisher.publish(display_trajectory)

    def execute_plan(self, plan):
        move_group = self.move_group
        move_group.execute(plan, wait=True)

    def wait_for_state_update(
        self, box_is_known=False, box_is_attached=False, timeout=4):
        box_name = self.box_name
        scene = self.scene

        start = rospy.get_time()
        seconds = rospy.get_time()
        while (seconds - start < timeout) and not rospy.is_shutdown():
            # Testeo si la caja esta acoplada
            attached_objects = scene.get_attached_objects([box_name])
            is_attached = len(attached_objects.keys()) > 0
```

```
        # Testeo si la caja se encuentra en area de trabajo.
        is_known = box_name in scene.get_known_object_names()

        # Testeo de reconocimiento de estado actual
        if (box_is_attached == is_attached) and (box_is_known == is_known):
            return True

        # Sleep para esperar finalizacion de subprocesos
        rospy.sleep(0.1)
        seconds = rospy.get_time()

    return False

def add_box(self, timeout=4):
    box_name = self.box_name
    scene = self.scene
    box_pose = geometry_msgs.msg.PoseStamped()
    box_pose.header.frame_id = "panda_link8"
    box_pose.pose.orientation.w = 1.0
    box_pose.pose.position.z = 0.10 # sobre el efector final
    box_name = "modulo"
    scene.add_box(box_name, box_pose, size=(0.075, 0.075, 0.2))

    self.box_name = box_name
    return self.wait_for_state_update(box_is_known=True, timeout=timeout)

def add_box2(self, timeout=4):
    box_name = self.box_name
```

```
        scene = self.scene

        box_pose = geometry_msgs.msg.PoseStamped()
        box_pose.header.frame_id = "panda_link0"
        box_pose.pose.orientation.w = 1.0
        box_pose.pose.position.z = 0.12
        box_pose.pose.position.x = 0.38
        box_name2 = "modelo anatomico"
        # Dimensiones del modelo anatomico segun fabricante
        scene.add_box(box_name2, box_pose, size=(0.38, 0.48, 0.25))
        self.box_name = box_name
        return self.wait_for_state_update(box_is_known=True, timeout=timeout)

    def attach_box(self, timeout=4):
        box_name = self.box_name
        robot = self.robot
        scene = self.scene
        eef_link = self.eef_link
        group_names = self.group_names

        grasping_group = "panda_arm" #anexado al grupo del brazo robotico
        touch_links = robot.get_link_names(group=grasping_group)
        scene.attach_box(eef_link, box_name, touch_links=touch_links)

        return self.wait_for_state_update(
            box_is_attached=True, box_is_known=False, timeout=timeout
        )
def main():
```

```
        try:
            print("Presionar control + D para salir")
            control_th = MoveGroupPythonInterfaceTutorial()
            #input("Presionar ‘Enter’ para mostrar posicion actual")
            #control_th.pose_state()

            #input("Presionar `Enter` para ejecutar movimiento basado en angulos para articulaciones")
            control_th.go_to_joint_state()

            #input("Presionar 'Enter' para mostrar posicion actual")
            control_th.pose_state()

            #input("Presionar `Enter` para planear y mostrar una trayectoria cartesiana ...")
            cartesian_plan, fraction = control_th.plan_cartesian_path()
            control_th.pose_state()  #####

            #input("Presionar `Enter` para mostrar la trayectoria guardada")
            control_th.display_trajectory(cartesian_plan)
            control_th.pose_state()  #####

            #input("Presionar `Enter` para agregar el modelo toracico al area de trabajo ...")
            control_th.add_box2()
            control_th.pose_state()  #####

            #input("Presionar `Enter` para agregar el módulo robótico al area de trabajo ...")
            control_th.add_box()
            control_th.pose_state()  #####
```

```
            #input("Presionar `Enter` para acoplar el módulo robótico al robot Panda ...")
            control_th.attach_box()
            control_th.pose_state()  #####

            #input("Presionar `Enter` para ejecutar la trayectoria guardad ...")
            control_th.execute_plan(cartesian_plan)
            control_th.pose_state()  #####

            #input("Presionar `Enter` para planear y ejecutar la trayectoria con un elemento acoplado ...")
            cartesian_plan, fraction = control_th.plan_cartesian_path(scale=-1)
            control_th.execute_plan(cartesian_plan)
            control_th.pose_state()  #####

        except rospy.ROSInterruptException:
            return
        except KeyboardInterrupt:
            return

    if __name__ == "__main__":
        main()
```

ANEXO E: CÓDIGO EN ROS PARA EL SISTEMA DE VISION ARTIFICIAL

```
#!/usr/bin/env python3

import rospy
import cv2
import numpy as np
import pyautogui
from Xlib import display, X
from std_msgs.msg import Int32
import math

# Parámetro para transformada de Hough
angle_threshold = 1

def get_window_geometry(window_name):
    # Conectando a la pantalla principal
    dsp = display.Display()
    root = dsp.screen().root

    # Lista de ventanas
    window_ids = root.get_full_property(dsp.intern_atom('_NET_CLIENT_LIST'),
    X.AnyPropertyType).value

    for win_id in window_ids:
        win_obj = dsp.create_resource_object('window', win_id)
        win_name = win_obj.get_wm_name()

        if win_name == window_name:
```

```
            # Geometría de la ventana
            geom = win_obj.get_geometry()
            parent = win_obj.query_tree().parent
            x, y = geom.x, geom.y

            while parent != root:
                geom_parent = parent.get_geometry()
                x += geom_parent.x
                y += geom_parent.y
                parent = parent.query_tree().parent

            width = geom.width
            height = geom.height

            return (x, y, width, height)

    return None

def is_horizontal(line, angle_threshold):
    x1, y1, x2, y2 = line
    dx = x2 - x1
    dy = y2 - y1
    angle = np.arctan2(dy, dx) * 180 / np.pi  # To convert degrees
    return abs(angle) < angle_threshold or abs(angle - 180) < angle_threshold

# Distancia del punto (x,y) hacia la línea definida por (x1,y1) y (x2,y2)
def distance_to_line(x, y, x1, y1, x2, y2):
```

```
    return abs((y2 - y1) * x - (x2 - x1) * y + x2 * y1 - y2 * x1) / math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)

def main():
    rospy.init_node('window_capturer', anonymous=True)
    command_pub = rospy.Publisher('actuator_command', Int32, queue_size=10)
    window_name = "Lenovo TB-8505F"

    orb = cv2.ORB_create()

    # Umbral de distancia para excluir puntos clave cerca de líneas rojas
    distance_threshold = 0  # 20 pixels

    frame1 = None

    transitions_count = 0  # Conteo para transiciones
    transitions_count_removal = 0
    previous_keypoints_count = 0  # Número de puntos clave en el frame previo
    keypoint_history = [ ]  # Lista para almacenar conteo de puntos clave.

    # Coordenadas de la región de interés (ROI) en la ventana de tiempo real
    roi_coords = [(618, 94), (1062, 94), (618, 708), (1062, 708)]

    first_roi_frame_captured = False
    first_roi_frame_bin = None

    f = False # Evitando repeticiones por KP residuales
    while not rospy.is_shutdown():
```

```
        # Obtener la geometría de la ventana actualizada en cada iteración
        window_geometry = get_window_geometry(window_name)

        if window_geometry:
            x, y, width, height = window_geometry

           # Si se encuentra la ventana, se procede a capturer la region correspondiente
            if width > 0 and height > 0:
                # Capture the specific window
                screenshot = pyautogui.screenshot(region=(x, y, width, height))
                frame = np.array(screenshot)
                frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)

                # Cambiar tamaño del marco (ej: reducir a 50% de su tamaño original)
                scale_percent = 50 # Cambiar valor para ajustar el tamaño
                width_resized = int(frame.shape[1] * scale_percent / 100)
                height_resized = int(frame.shape[0] * scale_percent / 100)
                        frame_resized = cv2.resize(frame, (width_resized, height_resized),
interpolation=cv2.INTER_AREA)

                # Desplegar la region de interés (ROI)
                # Convertir coordenadas de ROI a la escala del tamaño cambiado
                roi_x1 = int(roi_coords[0][0] * width_resized / width)
                roi_y1 = int(roi_coords[0][1] * height_resized / height)
                roi_x2 = int(roi_coords[1][0] * width_resized / width)
                roi_y2 = int(roi_coords[2][1] * height_resized / height)
                cv2.rectangle(frame_resized, (roi_x1, roi_y1), (roi_x2, roi_y2), (0, 255, 0), 2)
```

```
# Extraer region de interés del tamaño cambiado
roi_frame = frame_resized[roi_y1:roi_y2, roi_x1:roi_x2]

# Capturar y binarizar el primer “roi_frame”
if not first_roi_frame_captured:
    gray = cv2.cvtColor(roi_frame, cv2.COLOR_BGR2GRAY)

    # Imagen Binaria
  _, first_roi_frame_bin = cv2.threshold(gray, 40, 255, cv2.THRESH_BINARY)
    first_roi_frame_captured = True
    img_bin = first_roi_frame_bin
    # Erosion y Dilatación para resaltar bordes
    kernel = np.ones((7, 7), np.uint8)
    fill_image = cv2.morphologyEx(img_bin, cv2.MORPH_CLOSE, kernel)
    img_dilation = cv2.dilate(fill_image, kernel, iterations=10)
    img_erosion = cv2.erode(img_dilation, kernel, iterations=13)
    edges = cv2.Canny(img_erosion, 100, 150, apertureSize=3)
    kernel_size = 12
    kernel2 = np.ones((kernel_size, kernel_size), np.uint8)
    # dilated_edges = cv2.dilate(edges, kernel2, iterations=1)

    # Convertir a imagen de color
    img_dil_color = cv2.cvtColor(img_erosion, cv2.COLOR_GRAY2BGR)


    # Transformada de Hough para detectar lineas
    lines = cv2.HoughLines(edges, 1, np.pi / 180, 120)
```

```
upper_line = None
lower_line = None
height, width = frame.shape[:2]

if lines is not None:
    for line in lines:
        rho, theta = line[0]
        # Filtrar líneas para separar secciones
        if rho < height / 2:
            if upper_line is None or rho < upper_line[0]:
                upper_line = (rho, theta)
        elif rho > height / 2:
            if lower_line is None or rho > lower_line[0]:
                lower_line = (rho, theta)

    if lower_line is None and len(lines) > 1:
        for line in lines[1:]:
            rho, theta = line[0]
            if rho > height / 2:
                lower_line = (rho, theta)
                break

    # Trazar linea superior de la zona objetivo
    if upper_line is not None:
        rho, theta = upper_line
        a = np.cos(theta)
        b = np.sin(theta)
        x0 = a * rho
```

```
        y0 = b * rho
        x1 = int(x0 + 1000 * (-b))
        y1 = int(y0 + 1000 * (a))
        x2 = int(x0 - 1000 * (-b))
        y2 = int(y0 - 1000 * (a))
        #cv2.line(frame, (x1, y1), (x2, y2), (0, 0, 255), 8)
        cv2.line(roi_frame, (x1, y1), (x2, y2), (0, 0, 255), 8)
        cv2.line(img_dil_color, (x1, y1), (x2, y2), (0, 0, 255), 8)


    # Trazar línea en la zona inferior
    if lower_line is not None:
        rho, theta = lower_line
        a = np.cos(theta)
        b = np.sin(theta)
        x0 = a * rho
        y0 = b * rho
        x1 = int(x0 + 1000 * (-b))
        y1 = int(y0 + 1000 * (a))
        x2 = int(x0 - 1000 * (-b))
        y2 = int(y0 - 1000 * (a))
        #cv2.line(frame, (x1, y1), (x2, y2), (0, 0, 255), 8)
        cv2.line(roi_frame, (x1, y1), (x2, y2), (0, 0, 255), 8)
        cv2.line(img_dil_color, (x1, y1), (x2, y2), (0, 0, 255), 8)

cv2.imshow("Video",roi_frame)
f_gray = cv2.cvtColor(roi_frame, cv2.COLOR_BGR2GRAY)
equalized_frame = cv2.equalizeHist(f_gray)
```

```
# Detección de puntos clave (KP)
keypoints1 = orb.detect(equalized_frame, None)

keypoints_filtered = []
for kp in keypoints1:
    x = int(kp.pt[0])
    y = int(kp.pt[1])

    # Verificar puntos clave en la región objetivo
    pixel_value2 = img_dil_color[y, x]
    if pixel_value2[0] == 0 and pixel_value2[1] == 0 and pixel_value2[2] == 0:
        # Check distance to upper and lower red lines
        too_close_to_red_line = False

        # Calcular distancia a la línea superior roja
        if upper_line is not None:
            distance_to_upper = distance_to_line(x, y, x1, y1, x2, y2)
            if distance_to_upper < distance_threshold:
                too_close_to_red_line = True

        # Calcular distancia a la linea inferior roja
        if lower_line is not None:
            distance_to_lower = distance_to_line(x, y, x1, y1, x2, y2)
            if distance_to_lower < distance_threshold:
                too_close_to_red_line = True

        # Agregar puntos clave si no está cerca a la linea roja
        if not too_close_to_red_line:
```

```
            keypoints_filtered.append(kp)

current_keypoints_count = len(keypoints_filtered)
keypoint_history.append(current_keypoints_count)  # Almacenar conteo de KP

if not f and previous_keypoints_count <= 25 and current_keypoints_count > 30:
    transitions_count += 1  # Increment the transitions counter
    rospy.loginfo("Needle was inserted")
    command_pub.publish(1)  # Inserción de aguja
    rospy.sleep(5)

    # Ejecutar secuencia de casos 4, 5, 3, 4, 6, 3 (3 veces)
    for _ in range(3):
       command_pub.publish(4)  # Aspiración de líquido
       rospy.sleep(5)
       command_pub.publish(5)  # Apertura de valvula de tres vías (servo)
       rospy.sleep(1)
        command_pub.publish(3)  # Expulsión de líquido
        rospy.sleep(5)
       command_pub.publish(6)  # Cierre de válvula de tres vías (servo)
        rospy.sleep(1)

    command_pub.publish(2)  # Retracción de aguja
    f  = True; # Se evita repetición
elif previous_keypoints_count > 40 and current_keypoints_count <= 25:
    transitions_count_removal += 1
    rospy.loginfo("Needle was removed")
```

```
                # Graficar puntos clave (KP) en la ventana “Detection_insertion”
                frame_with_keypoints = cv2.drawKeypoints(img_dil_color, keypoints_filtered,
                None, color=(0, 255, 0))
                cv2.imshow('Detection_insertion', frame_with_keypoints)

                # Actualizar conteo de puntos clave anteriores para comparación con el sgte frame
                previous_keypoints_count = current_keypoints_count

                if cv2.waitKey(1) & 0xFF == ord('q'):
                    rospy.loginfo("Video playback interrupted by user.")
                    break

            # Presionar “q” para abortar ejecución
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
        else:
            rospy.loginfo(f"No se encontró ninguna ventana con el nombre '{window_name}'")

    cv2.destroyAllWindows()

if __name__ == '__main__':
    try:
        main()
    except rospy.ROSInterruptException:
                pass
```

ANEXO F: CÓDIGO DE CONTROL DE ACTUADORES Y SERVOMOTOR

```
#include <Servo.h>
// Actuador lineal _ aguja (Sub-Sistema B)
const int ENA_PIN = 9; // Pin 9 conectado a EN1 (pin de L298N)
const int IN1_PIN = 6; // Pin 6 conectado a IN1 (pin de L298N)
const int IN2_PIN = 5; // Pin 5 conectado a IN2 (pin de L298N)

// Actuador lineal _ jeringa (Subsistema C)
const int ENB_PIN = 8; // Pin 8 conectado a EN2 (pin de L298N)
const int IN3_PIN = 4; // Pin 4 conectado a IN1 (pin de L298N)
const int IN4_PIN = 3; // Pin 3 conectado a IN2 (pin de L298N)

Servo myServo;

// Función para inicialización
void setup() {
  // Inicialización de Serial
  Serial.begin(9600);

  // Inicializando la configuración de los pines digitales como salida
  pinMode(ENA_PIN, OUTPUT);
  pinMode(IN1_PIN, OUTPUT);
  pinMode(IN2_PIN, OUTPUT);
  pinMode(ENB_PIN, OUTPUT);
  pinMode(IN3_PIN, OUTPUT);
  pinMode(IN4_PIN, OUTPUT);
  digitalWrite(ENA_PIN, HIGH);
  myServo.attach(10);
```

```
  myServo.write(0);
  delay(1000);
}

// Funcion loop para ejecución de rutinas
void loop() {
  if (Serial.available() > 0) {
    int input = Serial.parseInt();

    switch (input) {
      case 1: // mover actuador lineal _ aguja hacia adelante
        needle_forward();
        break;

      case 2: // mover actuador lineal _ aguja en reversa
        needle_backward();
        break;

      case 3: // mover actuador lineal _ jeringa hacia adelante
        syringe_forward();
        break;

      case 4: // mover actuador lineal _ jeringa en reversa
        syringe_backward();
        break;

      case 5: // mover servo en sentido horario
        moveClockwise();
```

```
        break;

      case 6: // mover servo en sentido antihorario
        moveAnticlockwise();
        break;
    }
  }
}

void needle_forward() {
  // extensión de actuador lineal _ aguja
  digitalWrite(IN1_PIN, HIGH);
  digitalWrite(IN2_PIN, LOW);
  Serial.println("Move forward 1");
  delay(5000);
}

void needle_backward() {
  // retracción de actuador lineal _ aguja
  digitalWrite(IN1_PIN, LOW);
  digitalWrite(IN2_PIN, HIGH);
  Serial.println("Move backward 1");
  delay(5000);
}

void syringe_forward() {
  // extensión de actuador lineal _ jeringa
  digitalWrite(IN3_PIN, HIGH);
```

```
  digitalWrite(IN4_PIN, LOW);
  Serial.println("Move forward 2");
  delay(5000);
}

void syringe_backward() {
  // retracción de actuador lineal _ jeringa
  digitalWrite(IN3_PIN, LOW);
  digitalWrite(IN4_PIN, HIGH);
  Serial.println("Move backward 2");
  delay(5000);
}

void moveClockwise() {
  myServo.write(60);  // Mover servo 90 grados (sentido horario)
  Serial.println("Moved Clockwise to 90 degrees");
}

void moveAnticlockwise() {
  myServo.write(0); // Mover servo de retorno a 0 grados (sentido antihorario)
  Serial.println("Moved Anticlockwise to 0 degrees");
```

ANEXO G: CÓMUNICACIÓN ROS CON TARJETA ARDUINO (USB)

```
#!/usr/bin/env python3

import rospy
import serial
from std_msgs.msg import Int32

port = '/dev/ttyACM1'
baudrate = 9600
ser = serial.Serial(port, baudrate, timeout=1)

def send_command_to_arduino(command,ser):
    command_str = f"{command}\n"
    ser.write(command_str.encode())
    read_arduino()

def callback(data,ser):
    rospy.loginfo(f"Sending command {data.data} to Arduino")
    send_command_to_arduino(data.data, ser)

def read_arduino():
    response_from_arduino = ser.readline().decode('utf-8')
    print(response_from_arduino)
    rospy.loginfo(response_from_arduino)

def actuator_command_publisher():
    rospy.init_node('actuator_command_node', anonymous=True)
```

```
    # port = '/dev/ttyACM0'
    # baudrate = 9600
    # ser = serial.Serial(port, baudrate, timeout=1)

    rospy.Subscriber('actuator_command', Int32, callback,ser)
    rospy.spin()
    ser.close()

if __name__ == '__main__':
    try:
        actuator_command_publisher()
        # read_arduino()
    except rospy.ROSInterruptException:
        pass
```