ROBÓTICA DIDÁCTICA IMPLEMENTANDO SOFTWARES LIBRES

MARCO ANTONIO ARREDONDO OSORIO

JOEL ENRIQUE RODELO SEVERICHE

INGENIERÍA MECATRÓNICA

DEPARTAMENTO DE INGENIERÍA MECÁNICA, MECATRÓNICA & INDUSTRIAL

FACULTAD DE INGENIERÍAS Y ARQUITECTURA

UNIVERSIDAD DE PAMPLONA

PAMPLONA – NORTE DE SANTANDER

2022

ROBÓTICA DIDÁCTICA IMPLEMENTANDO SOFTWARES LIBRES

Autores

MARCO ANTONIO ARREDONDO OSORIO

JOEL ENRIQUE RODELO SEVERICHE

TRABAJO DE GRADO PRESENTADO COMO REQUISITO PARA OPTAR AL TÍTULO DE INGENIEROS EN MECATRÓNICA

DIRECTOR

OSWAL ALBEIRO VERA MOGOLLÓN
Ingeniero Mecatrónico

CO-DIRECTOR

YARA ANGELINE OVIEDO DURANGO
Ingeniera Mecatrónica
Magister (c) en controles Industriales

INGENIERÍA MECATRÓNICA

DEPARTAMENTO DE INGENIERÍA MECÁNICA, MECATRÓNICA & INDUSTRIAL

FACULTAD DE INGENIERÍAS Y ARQUITECTURA

UNIVERSIDAD DE PAMPLONA

PAMPLONA – NORTE DE SANTANDER

2022

Dedicatorias

Este proyecto lo dedicamos primeramente a Dios, por habernos dado la sabiduría y el entendimiento para culminar con éxito nuestro proyecto de grado, a todas las personas y amigos que siempre estuvieron acompañando en los momentos más difíciles durante todo el transcurso de este proceso y los cuales a través de sus consejos siempre nos impulsaron a seguir adelante, a la Universidad de Pamplona y a los docentes por habernos impartidos todos los conocimientos necesarios para poderle aportar algo a la sociedad con ayuda de este proyecto.

Agradecimientos

Marco Antonio Arredondo Osorio

Primeramente, a Dios por haberme dado la vida, sabiduría, salud y por guiarme en el camino correcto, ayudándome siempre a obtener logros tan importantes como lo es mi carrera profesional.

A mis padres, Josefa Osorio y Marco Arredondo por su esfuerzo en ayudarme de todas las formas posibles, por su apoyo incondicional que me brindan para dar cada uno de los pasos en mi vida, a ellos debo gran parte de este logro, porque ellos me enseñaron a luchar siempre por mis sueños y no desfallecer ante las adversidades, por los valores que me han inculcado que son los que hoy me hacen ser un buen ser humano. A mi hermana Mayra Arredondo por su apoyo incondicional, por estar siempre dispuesta a apoyarme en los buenos y malos momentos. A mis hijos, Kayler Arredondo y Juan David Arredondo por ser uno de los motivos que me impulsan en salir adelante y lograr todo lo que me propongo.

Al docente Nicolás Hernández Díaz por compartir sus conocimientos y guiarnos en el proceso de aprendizaje siempre haciéndonos entender que por más difícil que parezca con esfuerzo y dedicación todo es posible.

A mis compañeros francisco Gonzales Ortiz y Joel Rodelo por explicarme y apoyarme durante este proceso, compartiendo sus conocimientos y tiempo para que me fuera posible realizar con éxito este proyecto.

Joel Enrique Rodelo Severiche

Primero, agradecer a Dios por haberme dado la vida, la salud y la sabiduría ya que sin esto no hubiera sido posible cumplir el sueño de estudiar una carrera profesional.

A mis padres Carmen Severiche Arreola y Elmis Rodelo Bohórquez por el esfuerzo tan grande que realizaron al ayudarme de todas las formas posibles pese a las dificultades que en ocasiones se presentaban, gracias a su apoyo incondicional y a los valores que me inculcaron este logro también es parte de ellos porque me enseñaron que siempre es posible lograr los sueños cuando nacen del corazón.

A mis hermanos por siempre motivarme a seguir luchando pese a las dificultades y obstáculos presentados durante todo el transcurso de la carrera.

Al docente Oswal Albeiro Vera Mogollón por toda la ayuda brindada durante todo el proceso de aprendizaje, ya que gracias a sus conocimientos aportados fue posible de cierta manera la culminación de este ciclo.

Resumen

La principal finalidad de este proyecto es incentivar el uso de software libre para el aprendizaje de la robótica, por lo tanto, se construyó un robot antropomórfico de cuatro grados de libertad y un robot móvil omnidireccional de tres ruedas con fines didácticos, la aplicación para el robot antropomórfico estuvo enfocada en la escritura en lenguaje braille a través de unas rutinas preestablecidas, para este robot las articulaciones fueron de tipo rotacional, tiene un efector final fijo establecido que es el encargado de la escritura, por su parte, la aplicación del robot móvil es la de un robot mesero, este es el encargado de llevar a los usuarios los pedidos que estos realicen por medio de un menú de opciones prestablecido a través de una pantalla dispuesta en el robot. Los robots parten desde la etapa de diseño hasta su construcción, es por ello que con la ayuda de software CAD se planteó el diseño mecánico de piezas, así mismo se usó una impresora 3D para imprimir las piezas que se consideraron convenientes. Cabe resaltar que ambos robots contaron con un análisis para la selección de materiales dependiendo su viabilidad, esto comprendió motores, controladores, componentes electrónicos y diseño eléctrico, por consiguiente, se tuvieron en cuenta sus respectivos modelos cinemáticos. Finalmente, en cuanto a la construcción de algoritmos de control para la programación se usó Python el cual corresponde también a un lenguaje de programación gratuito y de alto nivel, de igual manera se usaron librerías de este lenguaje que permiten hacer estos cálculos. Debe señalarse que el control del robot móvil omnidireccional se realizó por medio de Bluetooth a través del PC de forma inalámbrica, mientras que el robot antropomórfico se hizo por comunicación serial por cable, ambos mediante el desarrollo de una interfaz gráfica que permitió la interacción de forma más práctica con los robots. Con el fin de proporcionar información de ayuda para futuros proyectos se crearon guías instructivas necesarias para replicar el proyecto de investigación e incentivar la

realización de mejoras en futuras investigaciones y su posible continuación. A manera de conclusión es importante mencionar la gran eficiencia que se obtuvo en cuanto al desempeño de los softwares libres en el área de la robótica, demostrando resultados iguales o mejores que las versiones pagas, así mismo se encontró una considerable ausencia de información en relación al tema de este proyecto, debido a la poca inversión e investigación usando softwares libres.

Palabras claves: Software libre, Python, robot antropomórfico, robot móvil omnidireccional.

Abstract

The main purpose of this project is to encourage the use of free software for learning robotics, therefore, an anthropomorphic robot with four degrees of freedom and a three-wheeled omnidirectional mobile robot were built for educational purposes, the application for the Anthropomorphic robot was focused on writing in braille language through pre-established routines, for this robot the articulations were of the rotational type, it has an established fixed end effector that is in charge of writing, for its part, the mobile robot application It is that of a robot waiter, this is in charge of bringing users the orders that they make through a menu of pre-established options through a screen arranged in the robot. The robots start from the design stage until its construction, which is why with the help of CAD software the mechanical design of parts was proposed, likewise a 3D printer was used to print the parts that were considered convenient. It should be noted that both robots had an analysis for the selection of materials depending on their viability, this included motors, controllers, electronic components and electrical design, therefore, their respective kinematic models were taken into account. Finally, regarding the construction of control algorithms for programming, Python was used, which also corresponds to a free and high-level programming language, in the same way libraries of this language were used that allow these calculations to be made. It should be noted that the control of the omnidirectional mobile robot was carried out by means of Bluetooth through the PC wirelessly, while the anthropomorphic robot was done by serial communication by cable, both through the development of a graphical interface that allowed interaction in a simple way. more practice with robots. In order to provide helpful information for future projects, the necessary instructional guides were created to replicate the research project and encourage improvements in future research and its possible continuation. By way of conclusion, it is important to mention

the great efficiency that was obtained in terms of the performance of free software in the area of robotics, demonstrating equal or better results than the paid versions, likewise a considerable absence of information was found in relation to the subject of this project, due to the low investment and research using free software.

Keywords: Free software, Python, anthropomorphic robot, omnidirectional mobile robot.

Tabla de contenido

Índice De Figuras

Índice De Tablas

Introducción

Actualmente el uso del software libre es una de las mejores opciones con los enormes cambios que impactaron la sociedad en los tiempos de pandemia, hecho que sin duda alguna revolucionó todo lo que está relacionado con la programación y que permite realizar proyectos de alto impacto, permitiendo que los investigadores hagan uso de las herramientas y funciones que dichos softwares disponen con el fin de que sean un avance para la ciencia. De tal manera se contribuye a la investigación y a la vez generan soluciones a distintas problemáticas que se presentan en la sociedad. Según (González-Barahona, 2011) hace ya tiempo que el software libre dejó de ser algo marginal, para convertirse en una realidad muy habitual para muchos usuarios. Incluso aunque no sepan que están usándolo, cada vez más usuarios realizan gran parte de sus tareas informáticas gracias a él.

Así mismo en el presente documento se plantea una serie de información con respecto a la aplicación del software libre más usado en los últimos años y en este caso se habla de dos modelos de robot un robot antropomórfico de cuatro grados de libertad y un robot móvil omnidireccional de tres ruedas con fines didácticos, la aplicación de estos robot se enfoca en brindar soluciones a problemáticas cotidianas de la sociedad, es por ello que se establece una función específica para el robot antropomórfico la escritura braille y para el robot móvil será de mesero. Cabe mencionar que la intención principal de este proyecto es aportar información base para futuras investigaciones las cuales puedan apoyarse en este trabajo de grado.

Objetivos

Objetivo General

Construir un robot didáctico móvil omnidireccional y un antropomórfico que permita el aprendizaje de la robótica, a través de la generación de guías instructivas implementando software libre.

Objetivos Específicos

- Diseñar el robot antropomórfico y móvil omnidireccional en un software 3D libre.
- Construir modelos de robots didácticos antropomórfico y móvil omnidireccional a partir de modelos generados en software libre.
- Implementar algoritmos cinemáticos para el control de los robots didácticos.
- Validar el funcionamiento de los algoritmos implementados.
- Generar guías instructivas para la implementación de proyectos didácticos.

Planteamiento del problema

La problemática en el uso de software con aplicaciones en el área de la robótica radica en los altos costos de adquisición de las licencias, teniendo en cuenta que incluso a nivel académico su costo sigue siendo bastante alto, en relación a la problemática expuesta, se hace necesario el uso de software gratuitos que tengan las mismas funcionalidades que los pagos, con el fin de no limitar el desarrollo de este proyecto de investigación a nivel académico.

Cabe mencionar que la desinformación en esta área es probablemente una de las causas del desinterés hacia este tema ya que son pocos los estudios encontrados que ayuden a evidenciar lo que aquí se propone, con esto se da a entender la importancia de incluir los softwares libres en el desarrollo de la ciencia y la tecnología. Otra de las problemáticas es que a pesar del gran potencial que poseen ciertos softwares libres como *Python* aun sea tan desconocido, generando el atraso y poco interés de los demás investigadores. Teniendo esto en cuenta nace la necesidad de mostrar mediante este proyecto el gran aporte que estos softwares pueden dar desde la ingeniería y otras ramas de la ciencia siempre en función de generar soluciones a las necesidades de esta sociedad cambiante.

Justificación

Se justifica el uso de software libre por razones de bajo presupuesto, además de recibir las mismas funciones y herramientas que sus equivalentes en versiones pagas, teniendo en cuenta la posibilidad de reducción de costos en el programa de ingeniería mecatrónica para proyectos académicos, cabe mencionar que al incentivar el uso de estos softwares se da un significativo avance a la ciencia y tecnología relacionada con la robótica. Teniendo en cuenta la enorme popularidad que caracteriza este lenguaje de programación a nivel mundial como lo menciona (Vallejo, 2021) "*Python* se proclama como líder en los lenguajes de programación más populares de la actualidad" lo cual fue reconocido por el índice de la comunidad de programación *TIOBE* (en inglés: *TIOBE programming community index*), hecho que demuestra la importancia de incrementar la enseñanza teórico-práctica de este lenguaje en la Universidad de Pamplona, con el fin de garantizar un aprendizaje acorde a los cambios y nuevas necesidades que surgen en la sociedad.

Otro aspecto no menos importante es la casi ausencia de equipos o herramientas didácticas en la Universidad específicamente para ingeniería mecatrónica que puedan ser usadas por los estudiantes para fortalecer su aprendizaje en el desarrollo académico.

Así mismo es importante recalcar las enormes ventajas que tiene usar el lenguaje de programación de *Python* debido a su influencia a nivel mundial ya que su uso abarca diversas áreas como inteligencia artificial, desarrollo web, *Data Science* y otros, además de su sencilla sintaxis y los foros de información que tiene a nivel global, esto lo hace mejorar en tiempo récord (Soloaga, 2018).

En cuanto a la impresión 3D es necesario recalcar las ventajas en cuanto a facilidad con respecto a producción de piezas para los robots que se construirán, además de la implementación

de una tecnología que ha tomado bastante fuerza en los últimos años y que también sus aplicaciones se han extendido en diversos campos como construcción, medicina, robótica, arte y otras, en otras palabras la industria tecnológica tiene altas expectativas en la impresión 3D teniendo características que hacen muy interesante como “la posibilidad de poder crear, de manera fácil, un objeto a la medida o porque simplemente cualquier diseño puede ser transformado en una realidad” (Camargo, 2020).

Así mismo la elección de los robots antropomórfico y móvil omnidireccional se basa principalmente en su gran diferencia entre sí, por ende, para cada uno se podrán desarrollar tareas muy distintas permitiendo el control por diferentes medios.

Por tanto, se hace pertinente el desarrollo de esta propuesta de investigación ya que actualmente *Python* se ha convertido en el lenguaje de programación más utilizado en el mundo, así mismo, con el desarrollo del presente trabajo se pretende contribuir a que en la Universidad de Pamplona profundice más en el uso de este tipo de herramientas las cuales se han popularizado significativamente en los últimos años con el drástico cambio que genero la pandemia del COVID – 19 en la sociedad.

Marco Teórico Y Estado Del Arte

Es necesario tener en cuenta que al paso de los años algunas tareas que representan peligro, cansancio o disminución de eficiencia en calidad y cantidad de producción para el ser humano, dieron paso a la aparición de los robots, uno de los inventos tecnológicos más representativos de la historia, cuyos elementos tecnológicos no solamente tienen la capacidad de apoyar actividades en el sector industrial, sino que tienen una infinidad de aplicaciones en los distintos sectores como económico, ambiental, social, militar y muchos otros más. Muchos sectores han apostado a la investigación de dichos robots con el fin de lograr una relación costo-beneficio, y de tal manera garantizar su inclusión en la modernización tecnológica.

Para comprender un poco más sobre estas sorprendentes maquinas inteligentes (Arnáez, 2015) menciona que "Robot, es una palabra que como tal fue empleada por primera vez por el escritor checo *Karel Čapek* en su obra *Opitek* en 1917, la cual etimológicamente proviene de la palabra 'robota', que significa 'servidumbre', 'esclavitud' o 'trabajo obligado'. Dichos robots se encuentran dentro de la ciencia de la robótica que según (Arnáez, 2015) "es una nueva disciplina que se encarga del estudio y del diseño de los robots y del movimiento de objetos en el espacio. Las causas que impulsan a la robótica están dadas por procesos industriales peligrosos como altas temperaturas o ambientes contaminados, por el alto costo de la fuerza de trabajo y por la efectividad económica al optimizar la relación costo-beneficio".

Los primeros robots con los que se comenzó a trabajar fueron llamados teleoperados, y fue *R.C Goertz* quien desarrolló el primer prototipo en el año 1948, este prototipo tenía el objetivo de manipular elementos radioactivos sin riesgo para el operador. Funcionaba con un dispositivo mecánico maestro-esclavo. (Sanchez et al., 2010).

Los robots en general poseen diversas clasificaciones dependiendo de sus características, dos tipos de robots que han tenido un acogimiento sorprendente han sido los robots móviles y los antropomórficos ya que son de los que se pueden adaptar con distintas tecnologías y sin duda alguna generan un excelente rendimiento.

Robots Móviles

Según (Robótica, 2021) menciona quepo por definición "un robot móvil es una combinación de sistemas mecánicos y eléctricos que se desplazan de manera autónoma controlados por un software de gestión de flota". Estos robots son muy utilizados en tareas que requieren desplazar algo de un lugar a otro, por ejemplo, en bodegas, almacenes o fábricas, donde constantemente se hacen movimientos de elementos de distintos pesos, y deben ser trasladados por distintos motivos. También menciona que "gracias a los sensores que integran, los robots móviles pueden percibir su entorno y desplazarse de manera eficiente por él. En el caso de los robots móviles autónomos (Robots AMR) pueden detectar un objeto o persona en su camino, siendo capaces de elegir una ruta alternativa para sortear el obstáculo." Lo anterior corrobora la enorme capacidad y desempeño que un robot móvil puede tener, y más aún cuando se le incorporan más tecnologías como la inteligencia artificial u otros elementos que le permitan tener más funcionalidades, entre otras características.

Generalmente hay dos tipos de robots móviles, los AMR "Robot Móvil Autónomo" y los AGV "Vehículo de Guiado Automático", cada uno tiene características positivas dependiendo para lo que se requiere.

AMR “Robot Móvil Autónomo”

Para (Robótica, 2021) los robots móviles autónomos “son los que se utilizan en las fábricas y almacenes para transportar y distribuir materiales por unas rutas flexibles que han sido previamente mapeadas.” Para poder desplazarse en el entorno donde son usados se les incorpora sensores además de un sistema LIDAR, esto con el fin de que puedan detectar tanto objetos como personas en su camino, sus rutas de desplazamiento se realizan basadas en la orden de un software de gestión de flota. Una de sus innumerables cualidades es su capacidad de adaptación a diversas necesidades de producción en tiempo real.

AGV “Vehículo de Guiado Automático”

Por otro lado, los robots AGV requieren de su propio hardware siendo diseñados con el fin de realizar trayectos preestablecidos, así mismo se pueden usar para transportar objetos pequeños o incluso cargas pesadas y aun así ofrecen una alta precisión en la entrega. Estos AGV utilizan tecnologías de navegación por medio de cintas magnéticas, imanes incrustados en el piso o sistemas láser. Tienen la ventaja de ser aplicaciones muy seguras ya que al detectar un obstáculo en su camino se detienen hasta que es retirado, no sin antes haber activado una alarma informando de su pausa en el recorrido (Robótica, 2021).

Robot Omnidireccional

Este tipo de robot puede moverse en cualquier dirección en el plano, sin importar la orientación en el mismo. También asegura que precisamente esta es una de sus mayores ventajas ya que le permite desplazarse sin limitaciones en su entorno y de esta manera garantizar su eficiencia en el desarrollo de las tareas asignadas (Davila, 2007).

Según (Martínez & Sisto, 2009) “Una de las principales características de los robots omnidireccionales son sus ruedas. Éstas, omnidireccionales, son las componentes que permiten

que el robot se desplace en cualquier dirección sin tener primero que rotar. El uso de estas ruedas también permite realizar trayectorias complejas que estén compuestas por un desplazamiento y rotación del robot simultáneamente para así poder alcanzar el destino con un ángulo deseado".

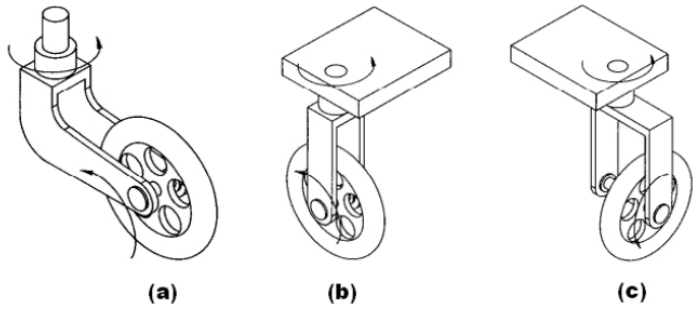
TIPOS DE RUEDAS OMNIDIRECCIONALES

Ruedas convencionales

Las ruedas convencionales posibilitan obtener desplazamientos omnidireccionales de cumplirse una configuración de posiciones apropiada sobre el robot. Estas ruedas son clasificadas de acuerdo a la posición del eje de rotación respecto de la rueda (Martínez & Sisto, 2009).

Figura 1. Tipos de ruedas convencionales.

(a) Tipo *forward offset steered,* (b) Tipo *convencional simple,* (c) Tipo lateral *offset steered*



Fuente: Citado por (Martínez & Sisto, 2009)

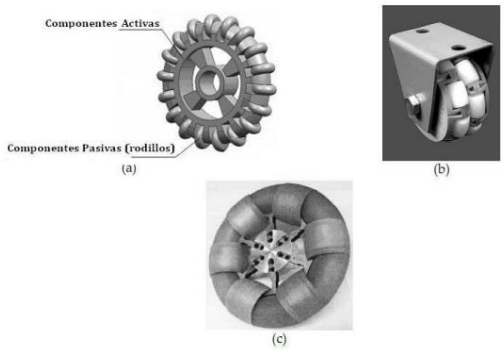
Ruedas especiales

Las ruedas especiales se basan en la idea de poseer una componente activa que provee tracción en una dirección y una componente pasiva en otra dirección. Entre las ruedas estudiadas, se incluyen las ruedas Universales, las ruedas *Mecanum* y por último las ruedas Esféricas. Las ruedas Universales poseen rodillos pasivos ubicados en la periferia de la rueda principal que

brindan una componente pasiva adicional a la activa que brinda la rueda en sí (Martínez & Sisto, 2009)

Figura 2. Ruedas universales.

(a) Rueda Simple, (b) Rueda doble, (c) Rueda alternada

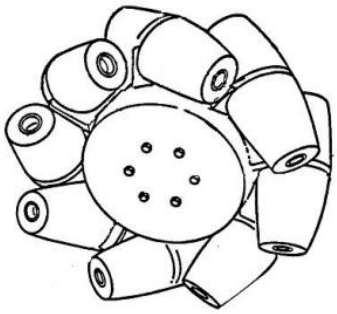


Fuente: Citado por (Martínez & Sisto, 2009)

Ruedas especiales *Mecanum*

Dentro de las ruedas especiales se encuentra la rueda de tipo *Mecanum*, los rodillos poseen una rotación de cierto ángulo, por lo general de 45° respecto de la circunferencia exterior de la rueda.

Figura 3. Rueda omnidireccional de tipo *Mecanum*



Fuente: Citado por (Martínez. S & Sisto. R, 2009)

ROBOT ANTROPOMÓRFICO

Los robots antropomorfos también han adquirido una amplia aplicabilidad ya que permiten desempeñar varias funciones y se usan en distintos sectores, según Tecnical “Los robots antropomórficos poseen una configuración especialmente indicada para acceder a zonas con obstáculos, haciendo uso de su configuración adaptable. Variando la posición y orientación, gracias a su elevado número de grados de libertad (usualmente 5 o 6), son idóneos en un amplio y variado abanico de aplicaciones industriales.”(Tecnical.cat, n.d.)

Según (Aguilar, 2013) los robots antropomorfos son llamados también manipuladores de codo, una de las principales características de este tipo de robots es que se conforman por tres partes; cuerpo, brazo, muñeca y un efector final (pinza o *gripper*). La aplicación de este robot es muy amplia, sin embargo, se puede decir que la más conocida es la industria, principalmente por las tareas de alto riesgo que allí se deben realizar y que estos robots realizan con alta eficiencia y sin exponer vidas humanas.

BRAZOS ROBÓTICOS

Los robots conocidos como brazos robóticos se han diseñado con la intención de asemejar las características de un brazo humano, incluyendo las funciones que este mismo puede desempeñar como manipulación de objetos, la velocidad o fuerza con que lo realizan, entre otras propiedades que resultan ser las mejores ventajas para esta adaptación tecnológica (Jabonero, 2010).

Así mismo se conoce que este tipo de robots se conforman por un grupo de elementos o eslabones que se unen por articulaciones y estas a su vez permiten movimientos. Teniendo en cuenta las sus características antropomórficas es por ello que muchas de sus partes se les atribuyen nombres relacionados con partes humanas, como cuerpo, brazo, muñeca o codo. Existe

también una característica especial que define la libertad que posee el robot en cuanto a movimientos independientes que este mismo pueda realizar y a ello se le llama grados de libertad (GDL), de la misma forma estos grados de libertad tratarse de desplazamiento o giro e incluso se pueden combinar los dos al mismo tiempo, igualmente dependiendo de las combinaciones que se realicen se pueden mencionar diferentes articulaciones como: Esférica o Rótula (3 GDL), Planar (2 GDL), Tornillo (1 GDL), Prismática (1 GDL), Rotación (1 GDL), Cilíndrica (2 GDL), por otra parte se conoce que la mayoría de veces se emplea las prismáticas y las de rotación cuando se habla de práctica (Jabonero, 2010).

El tema de los motores es crucial también para el proyecto, por esta razón se tienen en cuenta algunos conceptos dados por autores que han explorado esta área de la ingeniería (Garatejo & Raudales, 2021):

Motores de corriente continua (DC). Estos motores en la actualidad son los más implementados debido a que su control no resulta ser complejo.

• **Motores de corriente alterna (AC):** En comparación a los primeros, esta clase de motores no son muy implementados debido a que su manejo es mucho más complejo.

• **Motores paso a paso:** En el caso de los motores de este tipo, debido a que industrialmente los pares con los que trabajan son bastante pequeños, además de que no cuentan con suficiente precisión.

• **Servomotores:** Un servomotor es un actuador rotativo o motor que permite un control preciso en términos de posición angular, aceleración y velocidad, capacidades que un motor normal no tiene.

IMPRESIÓN 3D

Este fue sin duda alguna otro de los grandes aportes de la tecnología, y teniendo en cuenta un poco de su historia podemos ver que la primera máquina de impresión 3d se hizo en 1992, y fue impresión 3d de tipo SLA lo que se define como Estereolotigráfico, lo que fue realizado por la compañía 3d *Systems* (Adeva, 2022).

Para tener una definición más clara de lo que es y representa la impresión 3d se puede decir que es una serie de tecnologías de fabricación por adición que es capaz de crear objetos tridimensionales por medio de la superposición de una serie de capas de un material determinado, es decir se crea un objeto en tres dimensiones por medio de modelos digitales (Adeva, 2022).

Cabe mencionar que la impresión 3D requiere de dos elementos esenciales para su uso, y estos son un software diseñado para visualizar las figuras de lo que se pretende imprimir, y por supuesto, el hardware que representa la maquina o en si la impresora, además del material usado para la elaboración de las piezas en la impresión (Adeva, 2022).

Métodos de impresión 3D: Para este tipo de impresión hay distintas tecnologías que principalmente se distinguen por la forma en que son usadas las diversas capas para la creación de las piezas. Algunos de estos métodos son el SLS o FDM, depositar materiales líquidos para luego ser solidificados, teniendo en cuenta lo anterior es importante mencionar algunos de ellos (Adeva, 2022):

- **Impresión por inyección:** La impresora crea el modelo de capa esparciendo una capa de la sección de la pieza.
- **Modelado por deposición fundida (FDM):** depositando un material fundido sobre una estructura capa a capa que posteriormente es sintetizado por un láser para su solidificación.

- **Estereolitografía (SLA):** utiliza resinas líquidas foto poliméricas que se solidifican con el uso de una luz emitida por un láser ultravioleta.
- **Fotopolimerización por luz ultravioleta:** La fotopolimerización por luz ultravioleta o SGC, utiliza un recipiente de polímero líquido que es expuesto a la luz de un proyector bajo determinadas condiciones.
- **Fotopolimerización por absorción de fotones (SLS):** El objeto 3D es creado a partir del uso de un bloque de gel y mediante la utilización de un láser.

Materiales usados para la impresión 3D: En este caso los materiales deben ser previamente seleccionados teniendo en cuenta aspectos como el tipo de impresora en que se va a usar y el objeto que se va a crear, entre otros aspectos, los materiales que son más usados en la actualidad son: Ácido poliláctico (PLA), *Laywoo-D3*, Acrilonitrilo butadieno estireno (ABS, Poliestireno de alto impacto (HIPS), Tereftalato de polietileno (PET), Elastómero termoplástico (TPE), *Filaflex, Laybrick, Nylon* y Metales amorfos (BGM), (Adeva, 2022).

SOFTWARES LIBRES

Los Software CAD (Diseño Asistido por Computadora), son esenciales para el desarrollo profesional y académico de muchas áreas del conocimiento, pero los altos costos de algunos de estos programas restringen el alcance de increíbles investigaciones y proyectos, es por esto que se hace fundamental conocer la existencia de los diversos softwares gratuitos que tienen una enorme similitud en sus funciones y herramientas con respecto a sus equivalentes pagos. A continuación, se encuentra una tabla con la información más relevante de algunos de los softwares gratuitos para niveles de principiante, intermedio y avanzado (Shawn, 2021).

Tabla 1. *Softwares CAD gratuitos.*

PROGRAMA CAD	NIVEL	SISTEMA OPERATIVO (OS)	FORMATO
3D Builder	Principiante	Windows, Windows mobile, Xbox one y windows Hololense	3mf, obj, ply, stl
3D Slash	principiante	Windows, macOS, Linux, Raspberry PI o navegador	3dslash, obj, stl
BlocksCAD	Principiante	Navegador	Dxf, off, stl, scad
Leopoly	Principiante	Navegador	Stl, obj
LibreCAD	Principiante	Windows, MacOS y Linux	Dxf, pdf, svg
Wing 3D	Principiante	Windows, MacOS y Linux	3ds, fbx, obj, stl, dae, iwo, wrl, rwx, x, xml
Art of Illusion	Intermedio	Windows, MacOS y Linux	Obj, pov, wrl
FreeCAD	Intermedio	Windows, MacOS y Linux	Stl, step, iges, obj, dxf, svg, scad, iv, ifc, FCstd
Meshmixer	Intermedio	Windows, MacOS y Linux	Stl, amf, mix, obj, off
QCAD	Intermedio	Windows, MacOS y Linux	Dwg, dxf, dwf, pdf, svg
Blender	Avanzado	Windows, MacOS y Linux	Stl, 3ds, dae, fbx, dxf, obj, x, iwo, svg, ply, vrml, blend
BRL - CAD	Avanzado	Windows, MacOS, Linux, Solaris y BSD	Dxf, iges, stl, vrml, x3d

Fuente: (Shawn, 2021)

PYTHON (Lenguaje de programación gratuito)

Su historia indica que fue desarrollado a principios de los años 90 como un *hobby* por un ingeniero holandés conocido como Guido Van Rossum, el cual eligió el nombre de *Python* inspirado en el grupo cómico británico Monty Python del que Guido era una gran fan. (Robledano, 2019).

Python es un lenguaje de programación que se puede encontrar disponible para muchas aplicaciones y sistemas operativos como *iOS, Android, Linux*, *Windows o Mac*. También se conoce que es un lenguaje versátil multiplataforma y multiparadigma destacado por tener un

código legible y limpio, además cuenta con una licencia de código abierto, y es usado por grandes compañías como *Google, Facebook, YouTube* y otras. Cabe mencionar que Python es ideal para trabajar con grandes volúmenes de datos permitiendo su extracción y procesamiento siendo uno de los lenguajes favoritos para las empresas de *Big Data*, también tiene una gran influencia a nivel científico, e inclusive funciona para la creación de videojuegos. (Robledano, 2019).

Así mismo se destaca su amplio rango de aplicaciones en el mundo de las tecnologías, siendo usado desde desarrollo web, *Bigdata*, hasta Inteligencia artificial y robótica.

A continuación, se mencionan algunas de las ventajas que tiene usar este increíble lenguaje de programación, (Robledano, 2019).

- **Simplificado y rápido:** Este lenguaje simplifica mucho la programación, es un gran lenguaje para *scripting*.
- **Elegante y flexible:** El lenguaje ofrece muchas facilidades al programador al ser fácilmente legible e interpretable.
- **Programación sana y productiva:** Es sencillo de aprender, con una curva de aprendizaje moderada. Es muy fácil comenzar a programar y fomenta la productividad.
- **Ordenado y limpio:** es muy legible y sus módulos están bien organizados.
- **Portable:** Es un lenguaje muy portable. Podemos usarlo en prácticamente cualquier sistema de la actualidad.
- **Comunidad:** Cuenta con un gran número de usuarios. Su comunidad participa activamente en el desarrollo del lenguaje.

Otros de los aspectos importantes a mencionar sobre *Python* es que se desarrolla bajo una licencia de código abierto aprobada por OSI, lo que lo hace de libre uso y distribución, incluso

para uso comercial. La misión de *Python Software Foundation* es promover, proteger y hacer avanzar el lenguaje de programación *Python*, y apoyar y facilitar el crecimiento de una comunidad diversa e internacional de programadores de *Python.* (*About Python™ | Python.Org*, n.d.)

FUSION 360

"*Fusion 360* es una plataforma de *software CAD, CAM, CAE* y de circuitos impresos de modelado 3D basada en la nube para el diseño y la manufactura de productos".(*Fusion 360 | Software CAD, CAM, CAE y de Circuitos Impresos 3D Basado En La Nube | Autodesk*, n.d.)

EAGLE

"*EAGLE* es un software de automatización de diseño electrónico (EDA) que permite a los diseñadores de placas de circuito impreso (PCB) conectar sin problemas diagramas esquemáticos, ubicación de componentes, enrutamiento de PCB y contenido de biblioteca integral". (*EAGLE | PCB Design And Electrical Schematic Software | Autodesk*, n.d.)

FRITZING

"*Fritzing* es una iniciativa de hardware de código abierto que hace que la electrónica sea accesible como material creativo para cualquier persona. Ofrece una herramienta de software, un sitio web comunitario y servicios con el espíritu de *Processing* y Arduino, fomentando un ecosistema creativo que permite a los usuarios documentar sus prototipos, compartirlos con otros, enseñar electrónica en un salón de clases y diseñar y fabricar PCB profesionales".(*Fritzing*, n.d.)

VISUAL STUDIO CODE

"*Visual Studio Code* es un editor de código fuente ligero pero potente que se ejecuta en el escritorio y está disponible para *Windows, macOS y Linux*. Viene con soporte integrado para

JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes y tiempos de ejecución (*como C++, C#, Java, Python, PHP, Go, .NET*)".(*Documentación Para El Código de Visual Studio*, n.d.)

QTDESIGNER

"*Qt Designer* es una herramienta para diseñar y crear interfaces gráficas de usuario (GUI) con *Qt Widgets*. Pueden componerse y personalizarse las ventanas o cuadros de diálogo de manera *WYSIWYG* (lo que ve es lo que obtiene) y probarlos con diferentes estilos y resoluciones".(*Manual Del Diseñador Qt*, n.d.)

Diseño De Los Robots A Elaborar

Para la elaboración de los dos robots didácticos se busca que ambos cumplan con los requisitos que faciliten el aprendizaje de la robótica a través de la implementación de software gratuitos, y para ello se tendrán en cuenta sus respectivas funcionalidades.

Acotaciones

Al tratarse de una aplicación didáctica tanto el robot antropomórfico como el robot móvil omnidireccional van a estar condicionados por ciertos lineamientos en cuanto a sus respectivas funcionalidades. Por lo tanto, se busca garantizar al usuario la correcta manipulación de los mismos.

Por último, es conveniente acotar lo siguiente:

El robot antropomórfico al implementarse para una aplicación de escritura en lenguaje braille estará limitado solamente a una serie de frases predeterminadas, por consiguiente, el usuario no puede ingresar más palabras y solo puede seleccionar las que aparezcan dentro del menú de opciones.

Finalmente, para el robot móvil omnidireccional al ser implementado como robot mesero su principal aplicación será el de llevar y traer, este estará equipado con una pantalla dentro del cual estará un menú prestablecido para que el usuario pueda seleccionar la opción que desee.

CAPÍTULO I

Diseño mecánico

Para el diseño mecánico de las piezas de los robots se hizo necesario el uso del *software Fusión 360* versión 2.0.14337 con una licencia educativa otorgada por *Autodesk* de forma gratuita por un año, este facilita la generación, modificación y optimización de los diseños si es conveniente, al mismo tiempo genera una extensión compatible con el *software* de la impresora 3D que permite la impresión de las distintas piezas necesarias para la fabricación de los robots.

Además, durante la elaboración de los diferentes componentes mecánicos que integran a cada uno de los robots, se tendrán en cuenta las dimensiones de los motores, las tarjetas electrónicas y otros componentes requeridos dentro de los diseños para una fácil integración dentro de los modelos impresos.

A continuación, se muestran los distintos modelos *CAD* que van a integrar cada uno de los robots y se hace una proyección general de cada uno de los prototipos.

1. Diseño *CAD* de los Robots

El diseño *CAD* permite la facilidad de obtener un bosquejo más cercano a la realidad de lo que será el robot, de igual forma se implementan ciertas simulaciones que garanticen un mejor resultado a la hora de llevar a cabo la construcción del modelo real, igualmente, se cuenta con un diseño con menos incertidumbre y con la certeza de que se tendrán buenos resultados en cuanto a funcionalidad y desempeño permitiendo así ahorrar costos y material.

1.1. Diseño *CAD* Robot Móvil omnidireccional de tres ruedas

Para la construcción de las piezas mecánicas se tuvo en cuenta la aplicación a la cual está enfocado el robot, dicho esto, y partiendo que su función es la de un prototipo de robot mesero

con fines didácticos y que no se requiere de sistemas avanzados se procede con la elaboración de cada una de las piezas teniendo en cuenta que se trata de un diseño didáctico que busca favorecer su elaboración sin procedimientos tediosos y para el cual se tienen en cuenta aspectos como el tamaño, la funcionalidad y el desarrollo de una aplicación móvil.

El diseño del robot móvil omnidireccional consta de tres plataformas previamente dispuestas una encima de la otra por unos separadores, la primera plataforma constituye la parte donde se instalan los tres motores que controlaran el movimiento del robot y el soporte de la batería, por su parte, la segunda plataforma corresponde a la parte del mecanismo que soporta la electrónica utilizada, de modo similar, la tercera plataforma está dispuesta para cargar cada uno de los pedidos que se realicen por parte de los usuarios, esta a su vez dispone de un mecanismo que permite llevar el teléfono desde el cual está disponible una aplicación para la realización de los pedidos.

1.1.2. Parámetros de diseño robot móvil

A continuación, se definen algunos parámetros básicos que se tendrán en cuenta para el diseño del robot antropomórfico.

Cantidad de motores: 3

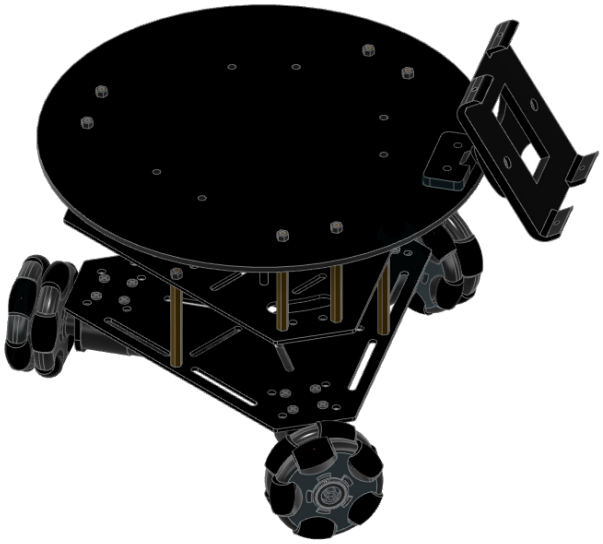
Cantidad de plataformas dispuestas: 3

Tamaño de la plataforma superior: 22cm de diámetro

Del mismo modo, se muestran los diseños *CAD* de cada una de las piezas y sus respectivas funcionalidades dentro del ensamble general del prototipo.

Ensamble: De acuerdo a la figura 4, esta representa una visión general de cómo quedará el robot móvil omnidireccional estructurado al llevarlo al ámbito físico para su posterior implementación.

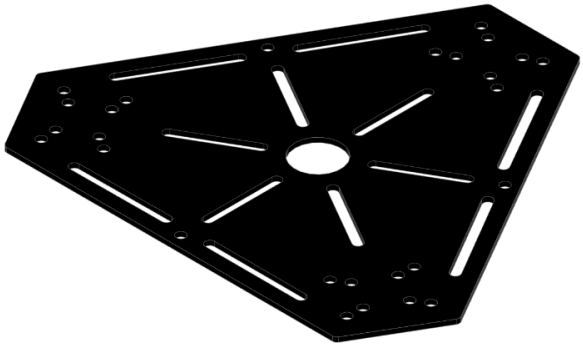
Figura 4. Diseño 3D del robot móvil Omnidireccional de 3 ruedas



Fuente: Elaboración propia

Plataforma I: Se constituye como una de las partes más importantes ya que está soporta sobre su estructura el resto de los componentes necesarios para el funcionamiento del robot y está representada en la figura 5, en esta estructura se integra la parte de actuadores y batería que son necesarios para la puesta en marcha del prototipo.

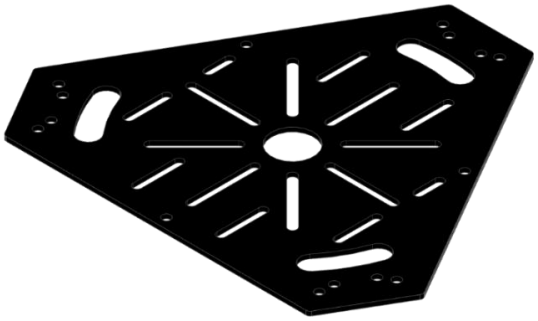
Figura 5. Plataforma I del robot móvil omnidireccional.



Fuente: Elaboración propia.

Plataforma II: Es la segunda plataforma del robot, esta va instalada sobre la plataforma I y se muestra en la figura 6, su función es la de soportar componentes relacionados con la parte electrónica necesarios para el control del mismo, esto incluye la PCB donde estarán integrados componentes como el Arduino nano, los *Driver DRV8833* y el módulo Bluetooth.

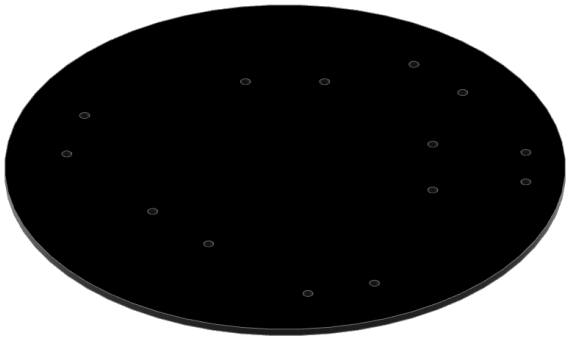
Figura 6. Plataforma II del robot móvil omnidireccional.



Fuente: Elaboración propia

Plataforma III: En la figura 7 se representa la pieza, es la parte del robot sobre la cual irán todos los elementos a llevar y sobre la cual está dispuesto un mecanismo que permite llevar un teléfono con una aplicación integrada como menú, para su diseño se tiene en cuenta las medidas y ubicación de cada uno de los soportes integrados en la plataforma II para que al momento de llevar a cabo el ensamble se eviten errores y las piezas encajen correctamente.

Figura 7. Plataforma III del robot móvil omnidireccional.



Fuente: Elaboración propia

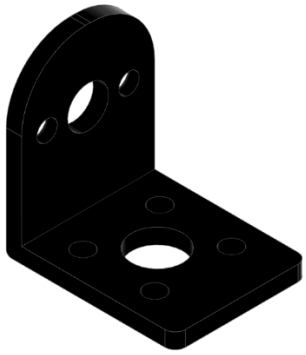
Soporte: Es importante porque permite sostener y separar las diferentes plataformas que van a constituir el robot, esta pieza se puede observar en la figura 8.

Figura 8. Soporte de plataformas del robot móvil omnidireccional.

Fuente: Elaboración propia

Soporte para motores: Constituye la parte del robot donde se aseguran los motores parte esencial para su funcionamiento, esta pieza se representa en la figura 9, para su diseño se tiene en cuenta la ubicación de los motores y el tamaño de sus agujeros.

Figura 9. Soporte para los motores del robot móvil omnidireccional.



Fuente: Elaboración propia

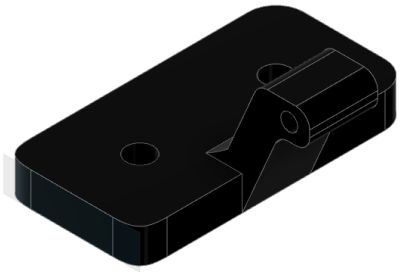
Ruedas: Se considera la parte principal y esencial del robot como se observa en la figura 10 ya que estas al ser de tipo omnidireccional permiten el movimiento en cualquier dirección debido a su configuración, esta cualidad representa ciertas ventajas en comparación con las ruedas tradicionales.

Figura 10. Ruedas del robot móvil omnidireccional.

Fuente: Elaboración propia.

Soporte para teléfono: Esta pieza como se muestra en la figura 11 es la encargada de soportar toda la estructura dispuesta para la ubicación del teléfono móvil con la aplicación de menú, en su construcción se tienen en cuenta especificaciones como el peso y tamaño del elemento a llevar.

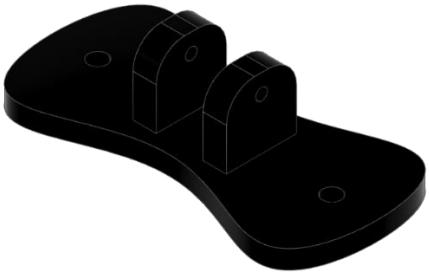
Figura 11. Soporte para teléfono.



Fuente: Elaboración propia.

Base para teléfono: Constituye la segunda pieza necesaria para articular toda la estructura que soportara el teléfono móvil, se representa en la figura 12 y es la encargada de soportar la última pieza donde se encuentra el elemento final, para su diseño se tienen en cuenta datos como el diseño de la pieza que lleva el teléfono, así como el tamaño de la misma.

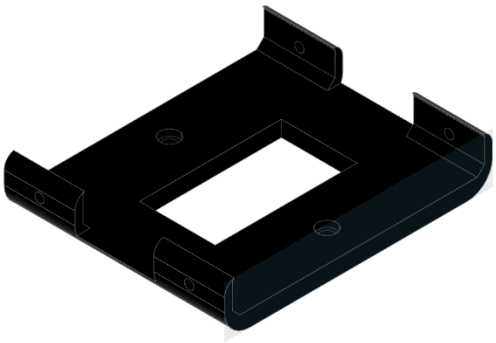
Figura 12. Base para teléfono.



Fuente: Elaboración propia.

Base de Encaje para teléfono: Como se observa en la figura 13, esta pieza es de gran importancia ya que es la encargada de llevar el teléfono y en el cual estará dispuesto el menú de opciones por medio de una aplicación móvil, para obtener el diseño se tienen en cuenta aspectos como el tamaño de teléfono y su forma.

Figura 13. Encaje para teléfono.



Fuente: Elaboración propia.

1.2. Diseño *CAD* Robot antropomórfico 4GDL

Para el diseño mecánico del robot antropomórfico se tiene en cuenta la aplicación a la cual está enfocado, en esta medida, y partiendo que su función es la de un prototipo de robot para escritura Braille con fines didácticos es necesario tener en cuenta que este prototipo consta de cuatro grados de libertad de tipo rotacional, estos a su vez definen la posición final del efector final.

Medidas:

Longitud de base - hombro: 123mm

Longitud eslabón antebrazo: 190mm

Longitud eslabón brazo: 120mm

Longitud eslabón muñeca – efector final: 120mm

1.2.2. Parámetros de diseño

A continuación, se definen algunos parámetros básicos que se tendrán en cuenta para el diseño del robot antropomórfico. Teniendo en cuenta que el área de trabajo es de 280mm x 220mm el robot antropomórfico puede realizar recorridos de 180 grados sobre su eje en la base.

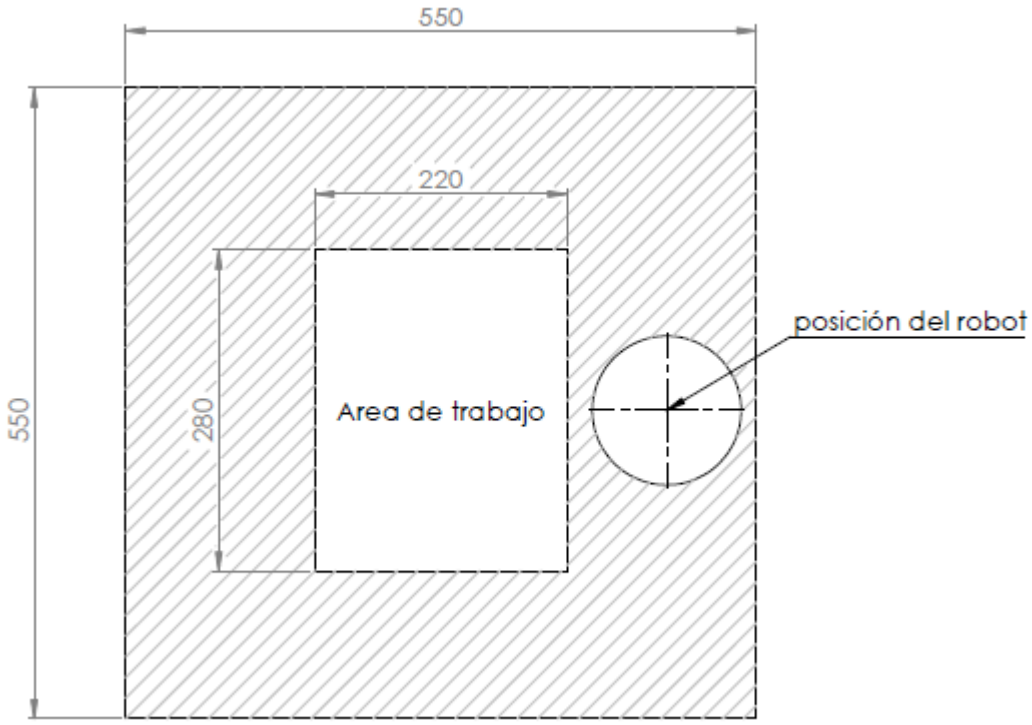
Dimensión base del robot: 550mm x 550mm

Área de trabajo: 280mm x 220mm

Grados de libertad: 4

En la figura 14 se muestra la zona de trabajo en donde se tienen en cuenta los límites de movimiento que puede implementar el robot.

Figura 14. Zona de trabajo.



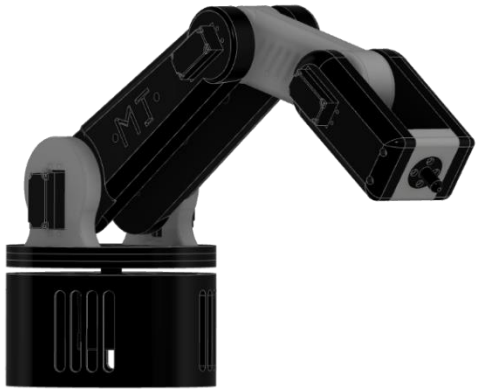
Fuente: Elaboración propia.

En la medida del espacio de trabajo se tiene en consideración la distancia total del robot incluido su efector final la cual es de 430mm.

Dicho lo anterior, seguidamente se muestran los diseños CAD de las respectivas piezas del robot antropomórfico.

Ensamble: Es la representación visual en la forma como quedará el robot a nivel estructural de forma física y está representado en la figura 15.

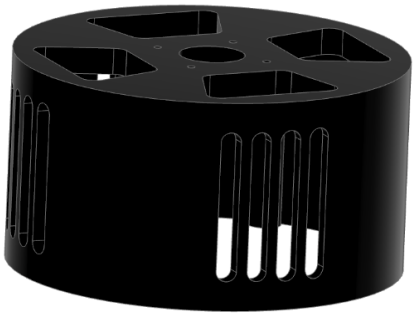
Figura 15. Diseño 3D del robot antropomórfico de 4GDL.



Fuente: Elaboración propia.

Base: Constituye la parte principal que se conoce como el soporte y es definido como el eslabón fijo, en este se sostiene toda la estructura del robot y sobre la cual se mueven las demás articulaciones, para el diseño se tiene en cuenta las medidas del motor a utilizar, cabe destacar que este elemento posee la primera articulación como se muestra en la figura 16.

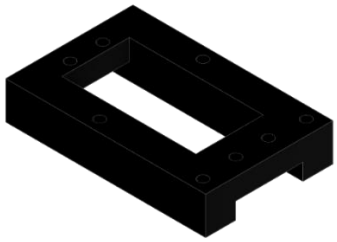
Figura 16. Base.



Fuente: Elaboración propia.

Soporte servomotor base: Esta parte se ubica en la base y es la encargada de contener el servomotor para el movimiento de la primera articulación, para su construcción se parte de saber las medidas proporcionadas por el fabricante del servomotor como se observa en la figura 17.

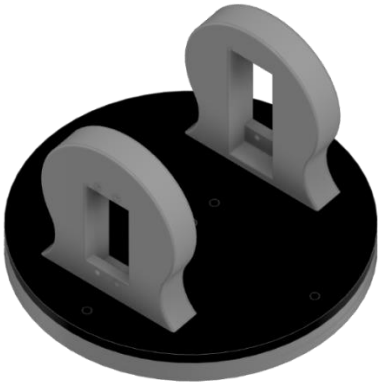
Figura 17. Soporte servomotor.



Fuente: Elaboración propia.

Hombro: Como se muestra en la figura 18, constituye el primer eslabón del robot sobre los cuales se integran dos servomotores, es una articulación de tipo rotacional, por consiguiente, debido a su ubicación es el elemento que más trabajo va a ejercer en comparación con el resto de los eslabones del sistema ya que sobre cada uno de sus servos se ejercen fuerzas las cuales son producto del resto de sus articulaciones, para su diseño al igual que las anteriores, se tienen en cuenta el tamaño de los servo y el resto de las articulaciones que constituyen el robot.

Figura 18. Hombro.



Fuente: Elaboración propia.

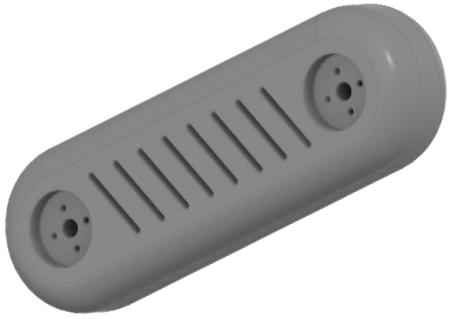
Antebrazo: Constituyen el segundo eslabón del robot y se pueden apreciar en la figura 19, en estas dos articulaciones se integran un servomotor y una balinera, para su construcción se tienen en cuenta parámetros como el tamaño de los componentes y la separación requerida entre ambas articulaciones para soportar el brazo del robot, es una articulación de tipo rotacional.

Figura 19. Antebrazo.

Fuente: Elaboración propia.

Brazo: Constituye el tercer eslabón del robot, es una articulación de tipo rotacional como se muestra en la figura 20, esta articulación va unida al antebrazo por medio del servomotor, para su elaboración se tienen en cuenta el espacio dejado entre las dos articulaciones del antebrazo y el tamaño de los sujetados de los servomotores.

Figura 20. Brazo.



Fuente: Elaboración propia.

Muñeca: Representado en la figura 21, constituye el cuarto eslabón del robot, al igual que el resto de articulaciones es de tipo rotacional y va enlazada al brazo, para el diseño se tiene en cuenta que lleva instalado un servomotor, una balinera y el efector final.

Figura 21. Muñeca.

Fuente: Elaboración propia.

Efector final: Representado en la figura 22, constituye el último eslabón de la cadena cinemática del robot, es uno de los más importantes ya que su propósito especifico es lograr la escritura en lenguaje braille para la cual va a ser la aplicación del robot, para su diseño se tuvieron en cuenta aspectos como la herramienta de trabajo.

Figura 22. Efector final.

Fuente: Elaboración propia.

CAPÍTULO II

1. Elección de Componentes Electrónicos

Es importante tener en cuenta ciertos parámetros para llevar a cabo una adecuada elección de los componentes electrónicos en cada uno de los robots, para esto es fundamental la realización de los cálculos previamente, estos datos permiten tener una visión más clara a la hora de seleccionar ciertos componentes con el propósito de obtener los mejores resultados en cuanto a funcionamiento y estabilidad. Para esto se deben tener en cuenta aspectos y características como la aplicación, es decir, basado en esto se pueden definir cuáles son los mejores elementos que garanticen un excelente desempeño físico de cada uno de los robots. Por ello, se debe tener especial cuidado a la hora de realizar estas operaciones ya que esto permite reducir perdida de materiales no aptos para el proyecto permitiendo ahorrar costos.

1.1. Componentes Electrónicos Robot móvil Omnidireccional

Para la elección de los componentes electrónicos se parte de requerimientos como el tipo de robot y la aplicación asignada al modelo didáctico, dicho lo anterior y teniendo en cuenta que se deben utilizar motores de corriente continua DC con *encoders* cuádruples para un mejor control de la posición y la orientación y que su comunicación debe ser establecida de forma inalámbrica se proceden a elegir los componentes que mejor de adapten a estos requisitos incluidos el tipo controlador que permita cumplir con las especificaciones de funcionamiento. Para ello, después de consultar todos los elementos disponibles en cuanto a motores, comunicación inalámbrica, controlador y drivers para el control de los actuadores se decidió emplear los que a continuación se describen por su fácil accesibilidad tanto económica como práctica, de igual forma se pretende facilitar su construcción para futuras investigaciones que

ayuden a mejorar el aprendizaje de la robótica de manera más didáctica y con bajo costo económico.

Actuadores

Los motores *DC* utilizados cuentan con un *encoder* de tipo incremental, una caja reductora de engranes metálicos y dos sensores de efecto hall los cuales generan dos señales de pulsos digitales desfasadas 90°, esto gracias a un disco giratorio magnético, los cuales están montados en el eje trasero del motor, debido a esto permite realizar diversidad de tareas que son fundamentales para el robot móvil ya que estos al ser de cuadratura tienen la capacidad de indicar tanto la velocidad como la posición y el sentido de giro del eje del motor y se pueden observar en la figura 23.

El tipo de *encoder* tiene varios modos de operación, los cuales son:

1. Control de posición.
2. Control de velocidad.
3. Control en la dirección del movimiento en el eje del motor.

Figura 23. Motor *DC* con *encoder*.

Fuente: (Amazon.com, n.d.)

A continuación, se presentan algunos parámetros del motor de escobillas otorgadas por el fabricante los cuales se describen en la tabla 2.

Tabla 2. *Parámetros del motor de escobillas otorgadas por el fabricante.*

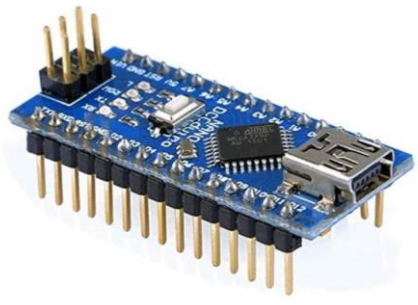
Voltaje de operación	9 – 12 [v]
Tamaño de la extensión del eje	14.5 [mm]
Voltaje de operación del sensor	3 -5 [v]
Par de rotor bloqueado	9.5 [Kg]
Velocidad de carga	100±10% [rpm]
Par de carga	3000 [g]
Corriente de carga	200 [mA]
Velocidad de salida	150±10% [rpm]
Corriente de rotor bloqueado	4500 [mA] (max)
Parámetros del disco de código	2 pulsos/ciclo
Diámetro del motor	21.4
Diámetro del eje	4 [mm]
Tamaño del motor	55.5 [mm]

Fuente: (Amazon.com, n.d.)

Tarjeta de Control

Se emplea una tarjeta Arduino nano de tipo *open source* como se muestra en la figura 24, este es un hardware bajo la licencia *Creative Commons Attribution – ShareAlike 3.0*, y es una placa de desarrollo de tamaño compacto basada en el microcontrolador ATmega328P. Esta permite realizar las configuraciones necesarias para el control del robot. Todo esto se realiza por medio de comunicación serial.

Figura 24. Arduino nano.



Fuente: (MercadoLibre, n.d.-b)

En la tabla 3 se presentan algunas de las características más importantes de la tarjeta dadas por el fabricante.

Tabla 3. *Características de la tarjeta dadas por el fabricante.*

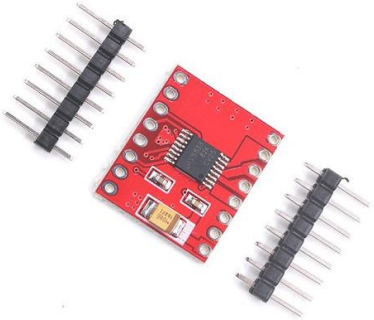
Dimensiones	45x18 [mm]
Peso	7 [g]
Microcontrolador	ATmega328
Tensión de funcionamiento	5 [v]
Memoria Flash	32 [kB]
Sram	2 [kB]
EEprom	1 [kB]
Oscilador de cristal	16 [MHz]
Arquitectura	AVR
Pines de entrada analógica	8
Corriente DC por pines E/S	40 [mA]
Tensión de entrada	7 – 12 [v]
Pines de E/S digital	22 (de los cuales 15 pueden proporcionar salidas PWM)
Pines PWM	6
Consumo de corriente	19 [mA]

Fuente: (MercadoLibre, n.d.-b)

Controladores

Como se muestra en la figura 25, para poder utilizar los motores de corriente directa se hace necesario utilizar los *drivers* DRV8833, estos proporcionan el control de corriente de puente H de doble solución, pueden conducir dos motores de corriente directa (DC), cuenta con dos controladores de puente H, estos permiten la alimentación conjuntamente con la tarjeta Arduino nano, así mismo permiten controlar el sentido de giro de manera independiente por medio de señales PWM.

Figura 25. Driver DRV8833.



Fuente: (Ferretrónica, n.d.)

Por otra parte, en la tabla 4 se presentan las siguientes características técnicas del *driver* que son consideradas como algunas de las más importantes y que por lo mismo se deben tener en cuenta al momento de darles un uso correcto.

Tabla 4. *Características técnicas del driver*

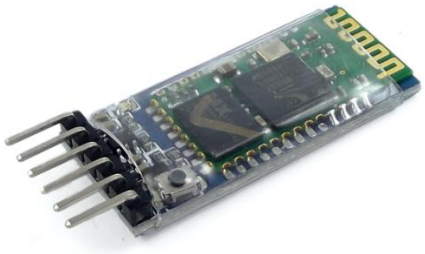
Alimentación	3 – 10 [VDC]
Corriente nominal	1.5 [A]
Pico máximo de corriente	2 [A] (Durante 2 segundos)
Canales	2 (Puede controlar 2 motores a la vez)
Dimensiones	18.1x21.1 [mm]

Fuente: Texas instruments, DRV8833 Dual H-Bridge Motor Driver

Comunicación

Para realizar la respectiva comunicación inalámbrica entre la interfaz gráfica realizada en Python y el robot móvil, se hizo necesario el uso del módulo bluetooth HC – 05 mostrado en la figura 26, este módulo permite comunicar de manera remota con Arduino por medio de nuestro PC con la facilidad de operación de un puerto serial.

Figura 26. Modulo bluetooth HC - 05.



Fuente: (Naylampmechatronics, n.d.)

Algunos de los datos característicos del módulo bluetooth HC - 05 se presentan en la tabla 5.

Tabla 5. *Datos característicos del módulo bluetooth HC - 05*

Voltaje de operación	3.6 – 6 [v]
Consumo de corriente	50 [mA]
Bluetooth	V2.0+EDR
Frecuencia	Banda ISM 2.4GHz
Modulación	GFSK (Gaussian Frequency Shift Keying)
Potencia de transmisión	4dBm, Class 2
Sensibilidad	-84dBm a 0.1% BER
Alcance	10 metros
Interfaz de comunicación	Serial TTL
Velocidad de transmisión	1200 bps hasta 1.3Mbps
Baudrate por defecto	38400,8,1,n
Temperatura de trabajo	-20°C a 75°C
Dimensiones	37x16 [mm]
Peso	3.6 [g]

Fuente: https://naylampmechatronics.com/inalambrico/43-modulo-bluetooth-hc05.html

Alimentación del robot

Batería lipo

Para la alimentación de los motores y el resto de componentes electrónicos se emplea una batería *lipo 2s* como la mostrada en la figura 27 la cual está constituida por dos celdas, su voltaje es de 7.4 V y posee una corriente de 1000 mAh con una duración de carga aproximada de 1 hora.

Figura 27. Batería *Lipo Turnigy* 7.4 V 2s 1000mah 20c.

Fuente: (MercadoLibre, n.d.-c)

Esta batería (*Lipo Turnigy 7.4 V 2s 1000mah 20c*) satisface toda la alimentación requerida en la electrónica del robot, ayudando al correcto funcionamiento y desempeño de todos los componentes electrónicos de una forma aceptable y demostrado de forma analítica.

Por esta razón, se debe tener en cuenta que todo el circuito va ser alimentado a una tensión de 7.4V, no obstante, se parte de saber que los motores tienen un consumo de corriente con carga de 1.2A, sin embargo, esta va a depender en gran medida de la corriente proporcionada por los *drivers DRV8833* a su salida, de igual forma se sabe que el módulo bluetooth tiene un consumo de corriente de 50mA, los *drivers DRV8833* consumen 3mA y el Arduino nano consume 19mA. Con ayuda de estos datos se determina el consumo total de corriente de todo el sistema. Así mismo, se determina el consumo total de los tres motores:

$$consumo\ de\ corriente\ maximo = N°\ Motor * consumo\ de\ corriente \quad (1)$$

La hacer uso de la ecuación (1) se obtiene la corriente máxima requerida por cada uno de los motores empleados.

$$consumo\ de\ corriente\ maximo = 3 * 0.2A$$

$$consumo\ de\ corriente\ maximo = 0.6A$$

Al tomar como referencia la corriente que consumen los motores, se justifica la elección de la batería *Lipo* mostrada en la figura 26.

1.2.Componentes Electrónicos Robot antropomórfico

Es importante mencionar que la elección de los materiales es una de las etapas más importantes ya que de esto depende el desempeño y la resistencia del robot, cada material se debe elegir cuidadosamente teniendo en cuenta múltiples factores, desde su objetivo principal hasta detalles como su duración, disponibilidad en el mercado y precio, claro está que todos ellos deben ser los idóneos con el fin de garantizar la mejor calidad que permita cumplir a cabalidad las funciones ya definidas para el robot. En base a lo anterior se hace elección del material PLA ya que cuenta con un amplio uso en la industria además de otras características como buena calidad, fácil acceso en el mercado y precios moderadamente asequibles. Con respecto a los actuadores eléctricos en el mercado se encuentra una gran variedad de motores y servomotores como lo son de corriente continua, corriente alterna y paso a paso, donde se ha decidido elegir el servomotor ya que permite un control preciso en términos de posición angular, aceleración y velocidad, por otro lado, para la parte del control y después de analizar la disponibilidad en el mercado se decidió optar por la opción del Arduino uno ya que es de fácil uso y de código abierto lo cual es primordial para el desarrollo del proyecto ya que su objetivo principal es el uso de softwares libres y por ende este dispositivo encaja perfectamente entre los componentes elegios. Esto permitirá darle una mejor calidad y desempeño al robot en sus movimientos lo que facilitará al robot llevar a cabo las funcionalidades para las cuales ha sido diseñado.

A continuación, se describen los elementos seleccionados teniendo en cuenta lo anteriormente mencionado.

Tarjeta de Control

Se emplea una tarjeta Arduino uno de tipo *open source* hardware bajo la licencia *Creative Commons Attribution – ShareAlike 3.0,* esta permite realizar las configuraciones necesarias para el control del robot. Todo esto por medio de comunicación serial y cuya representación se muestra en la figura 28.

Figura 28. Arduino Uno.

Fuente: (MercadoLibre, n.d.-a)

En la tabla 6 se presentan algunas de las características más relevantes de esta tarjeta.

Tabla 6. *Características del Arduino Uno*

Dimensiones	68.6x53.4 [mm]
Peso	25 [g]
Microcontrolador	ATmega328
Voltaje de operación	5 [v]
Voltaje de entrada	6 – 20 [v]
Pines de E/S digitales	14 (de los cuales 6 proporcionan salidas PWM)
Pines de entrada analógica	6
Corriente DC por pin de E/S	40 [mA]
Corriente DC para 3.3V pin	50 [mA]
Memoria flash	32 [Kb] de los cuales 0.5Kb utilizados por el bootloader
SRam	2 [kb]
EEPROM	1 [Kb]
Velocidad de reloj	16 [MHz]

Fuente: (MercadoLibre, n.d.-a)

Servomotores SG995

El Servomotor *TowerPro* MG995 mostrado en la figura 29, es un servo económico, sin embargo, posee un alto Torque de hasta 11Kg-cm. Cuenta con un diseño robusto, de alto rendimiento, tamaño estándar y cuenta con engranes de metal que lo hace más resistente a comparación de otros servos que trabajan con engranes de plástico. “Permite un control preciso de la posición, velocidad y aceleración angular. Este servomotor puede rotar de 0° hasta 180°.”(*Servomotor MG995 55g Digital Engranes de Metal - UNIT Electronics*, n.d.)

Figura 29. Servomotor SG995.

Fuente: (MercadoLibre, n.d.-d)

A continuación, en la tabla 7 se muestran algunas especificaciones técnicas proporcionadas por el fabricante.

Tabla 7. *Características del Servomotor SG995.*

Dimensiones	40.6x19.8x42.9 [mm]
Peso	55 [g]
Torque a 4.8V	8.9oz/in(10.00kg/cm)
Voltaje de operación	4 – 7.2 [V]
Velocidad de giro a 4.8V	0.2 seg/60°
Velocidad de operación (6V sin carga)	0.16 seg/60°
Tipo de interfaz	Analógica
Rango de temperatura	-30 a 60°C

Fuente: (MercadoLibre, n.d.-d)

CAPÍTULO III

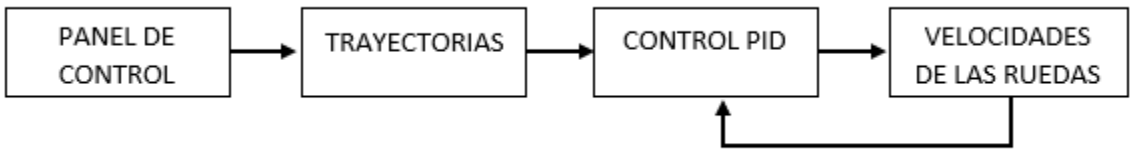
2. Modelados Matemático

2.1. Modelo cinemático Robot móvil omnidireccional

Debido a que el control cinemático propuesto es por seguimiento de trayectorias, en la figura 30 se plantea la ley de control que rige al sistema, en este se obtienen las velocidades necesarias de cada rueda para las trayectorias que el robot debe seguir.

El bloque que muestra el panel de control representa la interfaz de usuario desde el cual se indica la posición final a la que debe llegar el robot. Así mismo, en el bloque de trayectorias se encuentran los caminos propuestos que debe seguir el robot dependiendo la ubicación donde se encuentre la mesa a la cual se quiere llegar. Por su parte, el bloque control PID es la ley de control que rige el sistema para el seguimiento de las trayectorias generadas, finalmente, en el bloque de velocidades de las ruedas se obtienen las velocidades para lograr un seguimiento adecuado de las trayectorias por medio del control PID.

Figura 30. Esquema de operación del robot móvil.

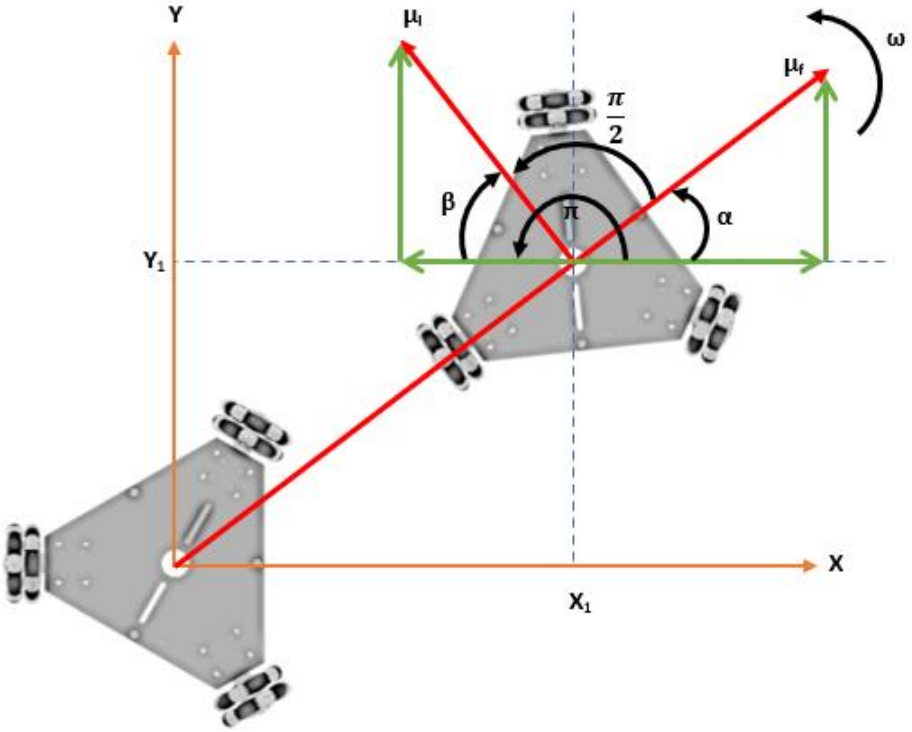


Fuente: Elaboración propia

2.2.Cinemática del Robot Móvil omnidireccional de tres ruedas

A continuación, se revisa el modelo cinemático cuyo objetivo es relacionar las velocidades del punto de interés con las velocidades que gobiernan el movimiento del robot y cuya representación se muestra en la figura 31. Para ello, se desplaza el robot un punto en los ejes x e y a un ángulo mayor que cero y menor a 90 grados observado desde la vista superior.

Figura 31. Componentes de la velocidad lineal y angular.



Fuente: Elaboración propia

$\mu_f = Velocidad\ frontal\ respecto\ a\ x$

$\mu_l = Velocidad\ lateral\ respecto\ a\ y$

$\omega = Velocidad\ angular$

$\alpha = Angulo\ de\ orientación$

Para obtener las ecuaciones de posición del robot necesarias para el cálculo de la cinemática se parte de la figura 31, teniendo en cuenta su ubicación respecto al eje x e y, quedando representadas como se observan en las expresiones (2) y (3).

$$h_x = x_1 \tag{2}$$

$$h_y = y_1 \tag{3}$$

Para determinar el valor del ángulo β mostrado en la figura 31 se tienen en cuenta todos los ángulos presentes en el diagrama como se muestra en la expresión (4).

$$\beta = \pi - \frac{\pi}{2} - \alpha \tag{4}$$

Al simplificar la ecuación (4) su solución se observa en la expresión (5).

$$\beta = \frac{\pi}{2} - \alpha \tag{5}$$

Para establecer la velocidad en el eje x por teorema de Pitágoras se obtiene la expresión mostrada en la ecuación (6).

$$\dot{h_x} = \mu_f * Cos(\alpha) - \mu_l * Cos(\beta) \tag{6}$$

Así mismo, buscando que la ecuación quede representada respecto al ángulo de orientación se remplaza (5) en (6), obteniendo la expresión (7).

$$\dot{h_x} = \mu_f * Cos(\alpha) - \mu_l * Cos\left(\frac{\pi}{2} - \alpha\right) \tag{7}$$

Por identidad trigonométrica aplicada a la ecuación (7) se tiene la expresión (8) necesaria para obtener el sistema de la velocidad respecto al eje x.

$$\dot{h_x} = \mu_f * Cos(\alpha) - \mu_l * (Cos\left(\frac{\pi}{2}\right) * Cos(\alpha) + Sin\left(\frac{\pi}{2}\right) * Sin(\alpha)) \tag{8}$$

Al simplificar los valores de la expresión (8) esta queda representada como se muestra en la ecuación (9), siendo la ecuación de velocidad requerida respecto al eje x.

$$\dot{h_x} = \mu_f * Cos(\alpha) - \mu_l * Sin(\alpha)) \tag{9}$$

Para establecer la velocidad en el eje y, se aplica el teorema de Pitágoras al triangulo rectángulo formado en la figura 31 obteniendo la expresión (10).

$$\dot{h_y} = \mu_f * Sin(\alpha) + \mu_l * Sin(\beta) \quad (10)$$

Al buscar nuevamente que el ángulo β de la expresión (10) quede en función del ángulo de orientación se remplaza nuevamente (5) en (10), quedando la ecuación (11).

$$\dot{h_y} = \mu_f * Sin(\alpha) - \mu_l * Sin\left(\frac{\pi}{2} - \alpha\right) \quad (11)$$

Por identidad trigonométrica aplicada a la ecuación (11), se obtiene la expresión (12), cuya representación es necesaria para el sistema de velocidad respecto al eje y.

$$\dot{h_y} = \mu_f * Sin(\alpha) - \mu_l * (Sin\left(\frac{\pi}{2}\right) * Cos(\alpha) - Cos\left(\frac{\pi}{2}\right) * Sin(\alpha)) \quad (12)$$

Simplificando la ecuación (12) se tiene que la expresión para la velocidad en el eje y queda representada como se muestra en (13).

$$\dot{h_y} = \mu_f * Sin(\alpha) - \mu_l * Cos(\alpha)) \quad (13)$$

La componente de la velocidad angular está dada por la expresión (14)

$$\dot{\alpha} = \omega \quad (14)$$

Recordando las expresiones (9), (13) y (14)

$$\dot{h_x} = \mu_f * Cos(\alpha) - \mu_l * Sin(\alpha)) \quad (15)$$

$$\dot{h_y} = \mu_f * Sin(\alpha) - \mu_l * Cos(\alpha)) \quad (16)$$

$$\dot{\alpha} = \omega \quad (17)$$

Se hace necesario reagrupar las ecuaciones (15), (16) y (17) en la expresión (18) para obtener la representación en forma matricial de la cinemática.

$$\begin{bmatrix} \dot{h_x} \\ \dot{h_y} \\ \dot{\alpha} \end{bmatrix} = [A] \begin{bmatrix} \mu_f \\ \mu_l \\ \omega \end{bmatrix} \quad (18)$$

Donde la matriz A mostrada en la ecuación (18) representa el jacobiano y está dada por los valores contenidos en la expresión (19).

$$[A] = \begin{bmatrix} Cos(\alpha) & -Sin(\alpha) & 0 \\ Sin(\alpha) & Cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (19)$$

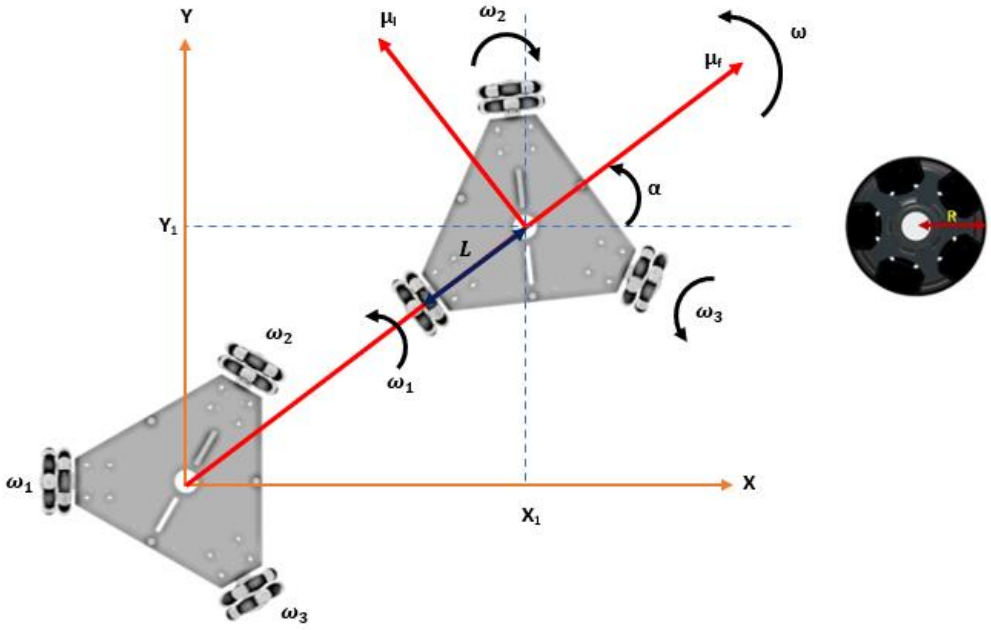
Al remplazar (19) en (18) se determina la cinemática del robot necesaria para la implementación de los algoritmos y cuya representación se muestra en la expresión (20).

$$\begin{bmatrix} \dot{h_x} \\ \dot{h_y} \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} Cos(\alpha) & -Sin(\alpha) & 0 \\ Sin(\alpha) & Cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mu_f \\ \mu_l \\ \omega \end{bmatrix} \quad (20)$$

Para determinar las velocidades de los motores y las velocidades globales a la que se debe mover cada rueda se basó en el modelo del *Paper* “Modelado cinemático y dinámico de un robot móvil omnidireccional”.(Muñoz & García, 2015)

En este documento se realizan los cálculos suficientes para la obtención de las velocidades de cada una de las tres ruedas y los cuales se pueden obtener a partir de la figura 30 haciendo uso de las ecuaciones (32) y (33) de dicho *Paper*.

Figura 32. Componentes de la velocidad global.



Fuente: Elaboración propia

La ecuación (32) tomada de este documento representa el jacobiano del robot móvil omnidireccional a analizar del cual se obtienen las velocidades globales dadas las velocidades de los motores, es decir, mediante los *encoders* se miden las velocidades a las que se mueven cada uno de los motores como se observa en la figura 32. Esta ecuación se representa en la expresión (21).

$$\begin{bmatrix} \mu_f \\ \mu_l \\ \omega \end{bmatrix} = \begin{bmatrix} 0 & \frac{R}{\sqrt{3}} & -\frac{R}{\sqrt{3}} \\ \frac{2R}{3} & -\frac{R}{3} & -\frac{R}{3} \\ -\frac{R}{3L} & -\frac{R}{3L} & -\frac{R}{3L} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{21}$$

Al resolver la matriz en la ecuación (21) se obtiene la solución mostrada en (22).

$$\begin{bmatrix}\mu_f\\ \mu_l\\ \omega\end{bmatrix}=\begin{bmatrix}\frac{R}{\sqrt{3}}(\omega_2)-\frac{R}{\sqrt{3}}(\omega_3)\\ \frac{2R}{3}(\omega_1)-\frac{R}{3}(\omega_2)-\frac{R}{3}(\omega_3)\\ -\frac{R}{3L}(\omega_1)-\frac{R}{3L}(\omega_2)-\frac{R}{3L}(\omega_3)\end{bmatrix} \tag{22}$$

Al simplificar los valores de la ecuación (22) se obtienen las expresiones (23), (24) y (25) las cuales determinan las velocidades globales a la cual el robot se va a mover. Siendo estas velocidades medidas las que se van aplicar al modelo cinemático.

$$\mu_f=\frac{R\sqrt{3}}{3}(\omega_2-\omega_3) \tag{23}$$

$$\mu_l=-\frac{R}{3}(\omega_2-2\omega_1+\omega_3) \tag{24}$$

$$\omega=-\frac{R}{3L}(\omega_1+\omega_2+\omega_3) \tag{25}$$

La ecuación (33) tomada del *Paper* y mostrada en la expresión (26) representa la inversa de la ecuación (32) para el cálculo de las velocidades globales de cada uno de los motores y cuyos valores dependen de paramentos como el radio de la rueda R y la longitud del centro de robot a la rueda L mostrados en la figura 32.

$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix}=\begin{bmatrix}0 & \frac{1}{R} & -\frac{1}{R}\\ \frac{\sqrt{3}}{2R} & -\frac{1}{2R} & -\frac{L}{R}\\ -\frac{\sqrt{3}}{2R} & -\frac{1}{2R} & -\frac{L}{R}\end{bmatrix}\begin{bmatrix}\mu_f\\ \mu_l\\ \omega\end{bmatrix} \tag{26}$$

$\omega_1,\omega_2,\omega_3 = Velocidades\ angulares$

$\mu_f,\mu_l = Velocidades\ angulares$

$\omega = Velocidad\ angular\ de\ rotacion\ sobre\ su\ propio\ eje$

$L = Radio\ de\ la\ rueda\ en\ metros$

Al operar la ecuación (26) se obtiene la representación matricial mostrada en (27).

$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix}=\begin{bmatrix}-\frac{L}{R}(\omega)+\frac{1}{R}(\mu_l)\\ -\frac{1}{2R}(\mu_l)-\frac{L}{R}(\omega)+\frac{\sqrt{3}}{2R}(\mu_f)\\ -\frac{1}{2R}(\mu_l)-\frac{L}{R}(\omega)-\frac{\sqrt{3}}{2R}(\mu_f)\end{bmatrix} \quad (27)$$

En base a las medidas de L y R, simplificando los datos en la ecuación (27) se obtienen las velocidades angulares en radianes por segundo de cada uno de los motores en función de las velocidades globales dada μ_f, μ_l y ω en la expresión (26) cuyo resultado se muestra en las expresiones (28), (29) y (30) y que son las velocidades de referencia que se deben aplicar a cada uno de los motores.

$$\omega_1=\frac{1}{R}(\mu_l-L\omega) \quad (28)$$

$$\omega_2=-\frac{1}{2R}(\mu_l+2L\omega-\sqrt{3}\mu_f) \quad (29)$$

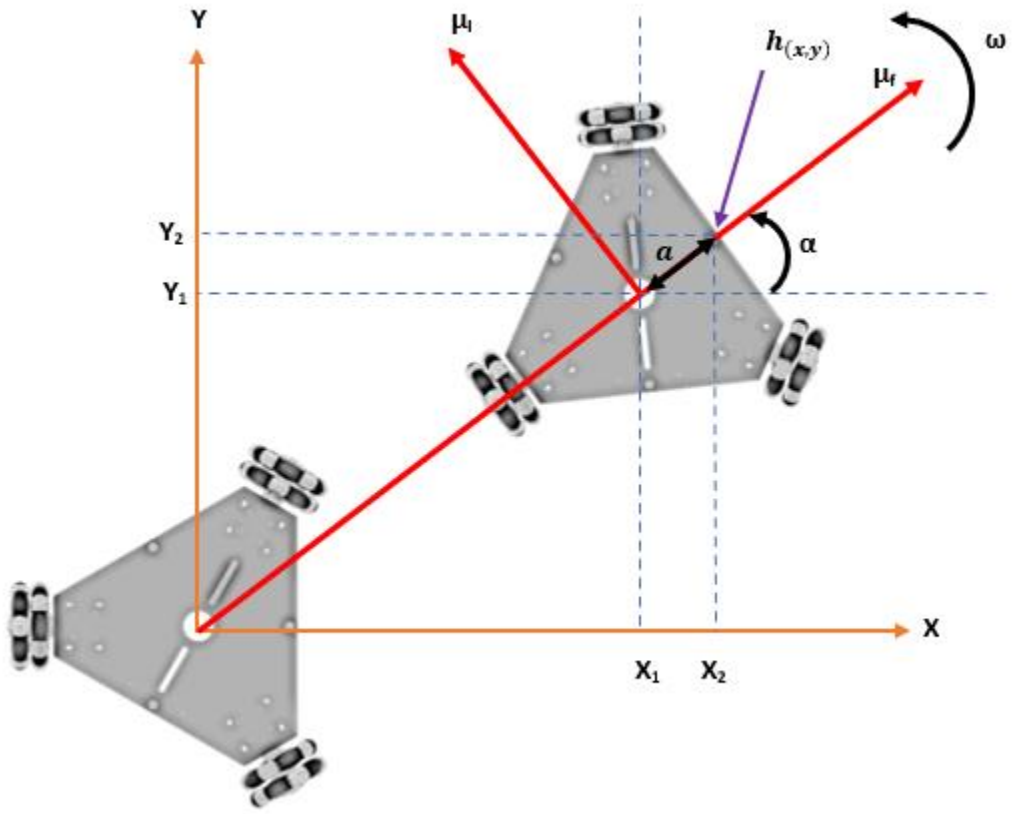
$$\omega_3=-\frac{1}{2R}(\mu_l+2L\omega+\sqrt{3}\mu_f) \quad (30)$$

Es de mencionar que si el valor de ω es positivo los motores giraran en sentido antihorario y viceversa.

2.2.1. Modelo Cinemático Con Punto Desplazado

Para determinar el punto de interés donde estará ubicada la herramienta, en este caso un teléfono inteligente con una aplicación móvil como menú, se desplaza un punto en el robot como se muestra en la figura 31.

Figura 33. Cinemática con punto desplazado.

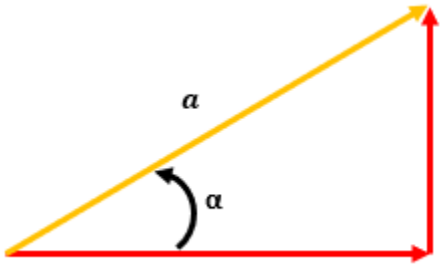


Fuente: Elaboración propia

Al analizar la figura 33 se observa que (a) representa el punto desplazado, esta forma un triángulo rectángulo que permite calcular los valores necesarios para la implementación de la cinemática del robot con la herramienta de trabajo dispuesta en el lugar donde se necesite.

Al extraer el triángulo rectángulo formado en la figura 33 este queda representado como se muestra en la figura 34.

Figura 34. Punto desplazado.



Fuente: Elaboración propia

Partiendo de los desplazamientos de x_1 y y_1 obtenidos en las ecuaciones (2) y (3), y teniendo en cuenta los datos de la figura 33, se debe agregar la porción a la cual se desplaza el punto de interés, estos se pueden ver representados en x_2 y y_2 como se muestran en las ecuaciones (31) y (32)

$$h_x = x_1 + x_2 \quad (31)$$

$$h_y = y_1 + y_2 \quad (32)$$

Al hacer uso de las relaciones por trigonometría en la figura 34, al remplazar los valores de x_2 y y_2 de las expresiones (31) y (32) para obtener la cinemática directa con su respectivo punto desplazado se determinan las ecuaciones (33) y (34) que representan las posiciones y están respecto al tiempo.

$$h_x = x_1(t) + a * \cos(\alpha)(t) \quad (33)$$

$$h_y = y_1(t) + a * \sin(\alpha)(t) \quad (34)$$

A continuación, para obtener el modelo cinemático diferencial del robot se realiza la derivada haciendo uso de las expresiones (35) y (36).

$$\frac{dh_x}{dt} = \frac{\partial h_x}{\partial x_1} * \frac{dy_1}{dt} + \frac{\partial h_x}{\partial \alpha} * \frac{d\alpha}{dt} \quad (35)$$

$$\frac{dh_y}{dt} = \frac{\partial y}{\partial y_1} * \frac{dx_1}{dt} + \frac{\partial h_y}{\partial \alpha} * \frac{d\alpha}{dt} \quad (36)$$

Al aplicar la derivada a las ecuaciones (33) y (34) por regla de la cadena a cada una de las variables que varían en el tiempo se obtienen las expresiones (37), (38) y (39).

$$\dot{h_x} = a * \omega * Sin(\alpha) \tag{37}$$

$$\dot{h_y} = a * \omega * Cos(\alpha) \tag{38}$$

$$\dot{\alpha} = \omega \tag{39}$$

Al agregar las expresiones (37) y (38) a las ecuaciones (15) y (16) se obtienen como resultado los sistemas (40) y (41), donde la componente de velocidad angular de la ecuación (17) queda representada de la misma forma como se muestra en (42).

$$\dot{h_x} = \mu_f * Cos(\alpha) - \mu_l * Sin(\alpha) - a * \omega * Sin(\alpha) \tag{40}$$

$$\dot{h_y} = \mu_f * Sin(\alpha) - \mu_l * Cos(\alpha) + a * \omega * Cos(\alpha) \tag{41}$$

$$\dot{\alpha} = \omega \tag{42}$$

Al organizar el sistema de ecuaciones (40), (41) y (42) se obtiene la representación matricial de A la cual queda como se muestra en la expresión (43).

$$[A] = \begin{bmatrix} Cos(\alpha) & -Sin(\alpha) & -a\omega Sin(\alpha) \\ Sin(\alpha) & Cos(\alpha) & a\omega Cos(\alpha) \\ 0 & 0 & 1 \end{bmatrix} \tag{43}$$

Se debe remplazar el valor de la ecuación (43) en la expresión (44).

$$\begin{bmatrix} \dot{h_x} \\ \dot{h_y} \\ \dot{\alpha} \end{bmatrix} = [A] \begin{bmatrix} \mu_f \\ \mu_l \\ \omega \end{bmatrix} \tag{44}$$

Luego de remplazar los datos en la expresión (44), se obtiene una ecuación matricial donde $[A]$ representa el jacobiano, esta queda representada como se muestra en (45).

$$\begin{bmatrix} \dot{h_x} \\ \dot{h_y} \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} Cos(\alpha) & -Sin(\alpha) & -a\omega Sin(\alpha) \\ Sin(\alpha) & Cos(\alpha) & a\omega Cos(\alpha) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mu_f \\ \mu_l \\ \omega \end{bmatrix} \tag{45}$$

2.2.2. Modelo Matemático Del Motor

Para la sintonización de los parámetros de control del algoritmo PID implementados, se toma como referencia del modelo dinámico del motor dispuesto en el *Paper* "Identificación de modelos de orden reducido a partir de la curva de reacción del proceso".(Tecnología & 2006, 2006)

Específicamente en este documento se detallan las ecuaciones matemáticas necesarias para la realización del control y el uso de los modelos de identificación para obtener los tiempos óptimos para los parámetros del modelo de identificación. Del cual se tomaron como referencia las ecuaciones (1) y (5) que definen el modelo de sintonización en lazo abierto para el control PID con un escalón unitario a la entrada.

Ya que existen diferentes maneras en las que se puede aproximar el comportamiento de un motor, se procedió aproximar este a un sistema de primer orden con un tiempo muerto.

La ecuación (1) tomada del *Paper* que representa este comportamiento de primer orden está dada como se muestra en la expresión (46), y cuya representación matemática está en el dominio de la frecuencia.

$$G(s) = \frac{k_p e^{-Ls}}{\tau s+1} \tag{46}$$

Donde

$K_p = Ganancia\ del\ proceso\ en\ estado\ estacionario$

$L = Tiempo\ de\ retardo$

$\tau = Constante\ de\ tiempo\ en\ lazo\ abierto\ (tao)$

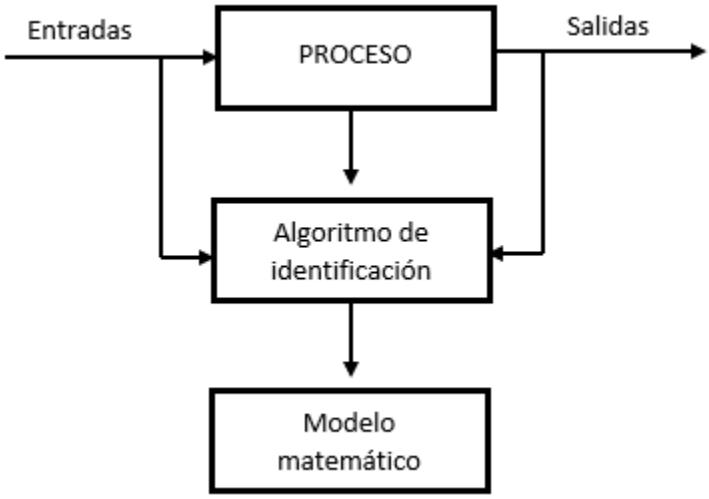
Donde se tiene que K_p, L, y τ son los datos que determinan el comportamiento del motor y por tal razón estos son los tres parámetros a identificar en el modelo.

La ecuación (5) referenciada en el *Paper* representa el modelo matemático del motor en el dominio del tiempo con un retardo y está dada como se muestra en la expresión (47).

$$p_v(t) = \begin{cases} 0 \\ p_v(1 - e^{-(t-L)/\tau})\Delta C_V \end{cases} \quad \begin{matrix} 0 \leq t < L \\ t \geq L \end{matrix} \tag{47}$$

Para la identificación de los parámetros de control basado en pruebas experimentales se hace necesario el uso del esquema mostrado en la figura 35.

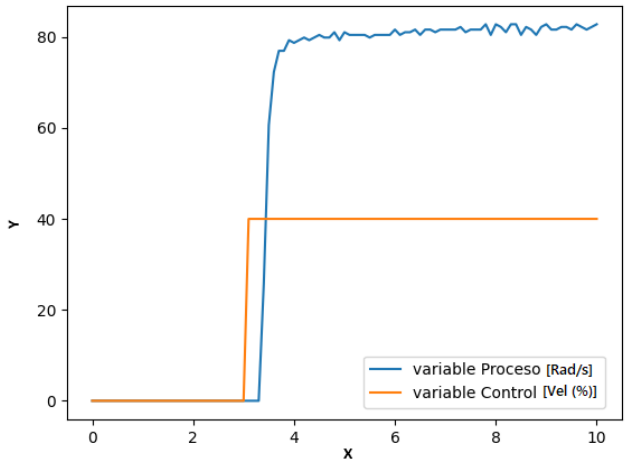
Figura 35. Esquema de parámetros de identificación.



Fuente: Elaboración propia

Al aplicar como entrada un escalón normalizado de cero a cien por ciento al sistema se obtiene como respuesta a la salida los radianes por segundo a la que gira el eje del motor y los cuales se muestran en la figura 36 validando el comportamiento obtenido por el sistema.

Figura 36. Respuesta al escalón.



Fuente: Elaboración propia

En la figura 36, se observa que al introducir un escalón al sistema como el representado en color naranja y al analizar su respuesta en color azul se puede ver que estos son los radianes por segundo a los cuales gira el eje del motor y está dada en el dominio del tiempo.

Seguidamente, se ejecuta el algoritmo de identificación llamado PSO cuya función va a ser la de proporcionar de manera aleatoria los valores de k_p, L, y τ del modelo matemático de la ecuación (46), estos valores se irán ajustando dependiendo el error entre la salida que se ha medido y la salida que proporcione el algoritmo de identificación.

Los parámetros proporcionados por el algoritmo de identificación son los siguientes, donde:

$K_p = 2.0286$

$L = 0.2578$

$\tau = 0.1089$

Una vez conseguidos los valores de los parámetros K_p, L y τ de la ecuación (46), se ingresan a la función Lambdatunning realizada en Arduino, con el objetivo de obtener los

parámetros de la ganancia proporcional, tiempo integral y el tiempo derivativo necesarios para el diseño del control PID del motor.

Las ganancias obtenidas por el método lambda están dadas por

$K_c = 0.18$

$T_i = 0.12$

$T_d = 0.01$

Donde:

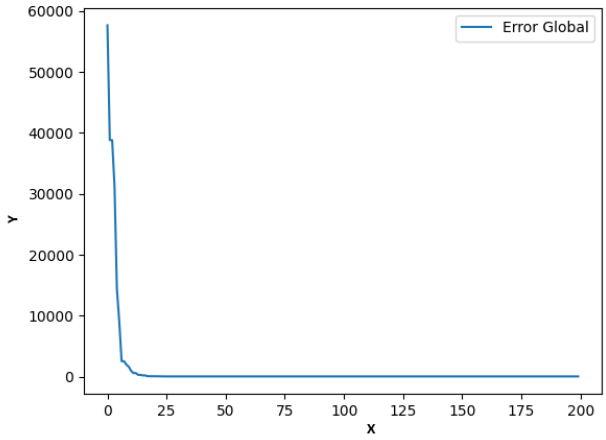
$K_c = Ganancia\ proporcional$

$T_i = Tiempo\ integral$

$T_d = Tiempo\ derivativo$

Al ejecutarse los parámetros obtenidos por el método lambda se puede observar el comportamiento del control PID realizado al motor.

Figura 37. Error global.

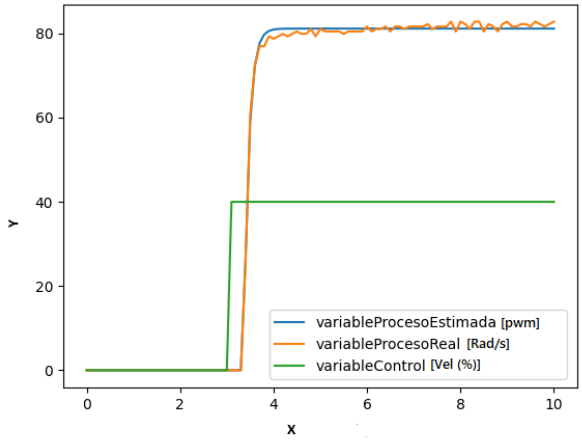


Fuente: Elaboración propia

Una vez que el error sea mínimo como se muestra en la figura 37 se aproxima la respuesta de la curva como se observa en la figura 38 gracias a que el método lamba ajusta de

manera automática los parámetros, la gráfica de color azul es la respuesta con el algoritmo de identificación y la de color naranja es la aproximación, en esta se puede analizar que su comportamiento es bastante bueno en comparación a la aproximación real en color azul.

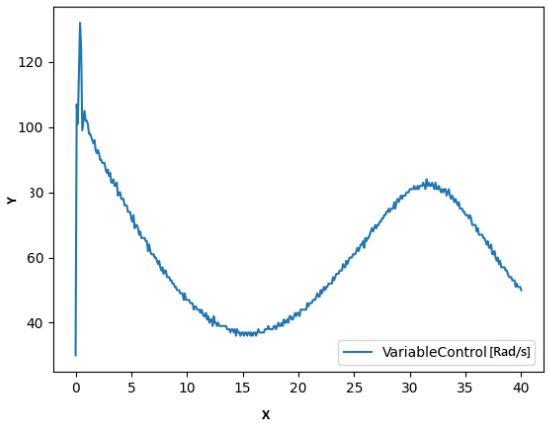
Figura 38. Modelo Identificado.



Fuente: Elaboración propia

Al aplicar una entrada tipo coseno, en la figura 39, se puede observar el comportamiento obtenido por el algoritmo de identificación en el control PID en lazo abierto.

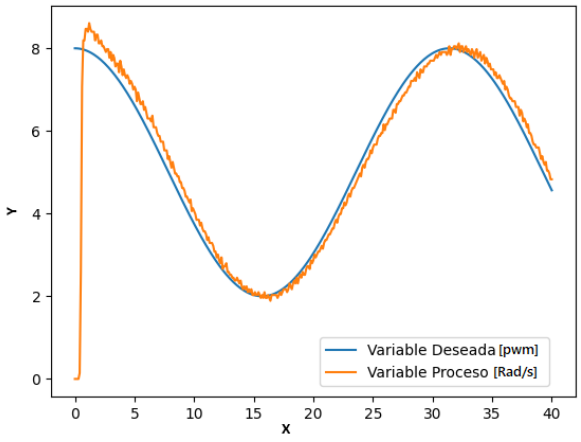
Figura 39. Señal coseno obtenida por algoritmo de identificación.



Fuente: Elaboración propia

En la figura 40, se observa la respuesta obtenida por el algoritmo de identificación donde se puede verificar que para valores fijos como los mostrados en la figura 38 relativamente presenta un buen resultado, mientras que para una trayectoria los resultados no son muy buenos ya que la variable de proceso se encuentra retardada en comparación a la variable deseada, por lo tanto, se debe solucionar el problema aplicando una sintonización fina.

Figura 40. Respuesta del algoritmo de identificación.



Fuente: Elaboración propia

2.2.3. Control PID Y Sintonía Fina Lambda

La sintonía fina consiste en tomar los valores de una sintonía ya realizada, para este caso el método lambda y a partir de estos valores ir variando los parámetros en base a las siguientes reglas.

1. Aumentando la ganancia proporcional disminuye la estabilidad.
2. El error decae más rápidamente si se disminuye el tiempo de integración.
3. Disminuyendo el tiempo de integración disminuye la estabilidad.
4. Aumentando el tiempo derivativo mejora la estabilidad.

Con base en los resultados obtenidos lo que se busca es que el error decaiga más rápido a cero por lo tanto se hace disminuyendo el tiempo de integración como lo menciona la regla número dos.

Las ganancias obtenidas por el método lambda están dadas por

$K_c = 0.18$

$T_i = 0.12$

$T_d = 0.01$

Al disminuir el tiempo integral expuesto en la regla dos a la mitad se tiene

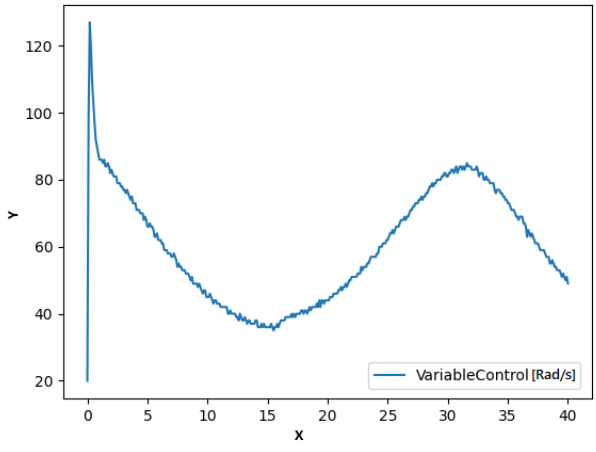
$K_c = 0.18$

$T_i = 0.06$

$T_d = 0.01$

En la figura 41 se muestran las acciones de control establecidas con los parámetros de ganancia proporcional, tiempo integral y tiempo derivativo.

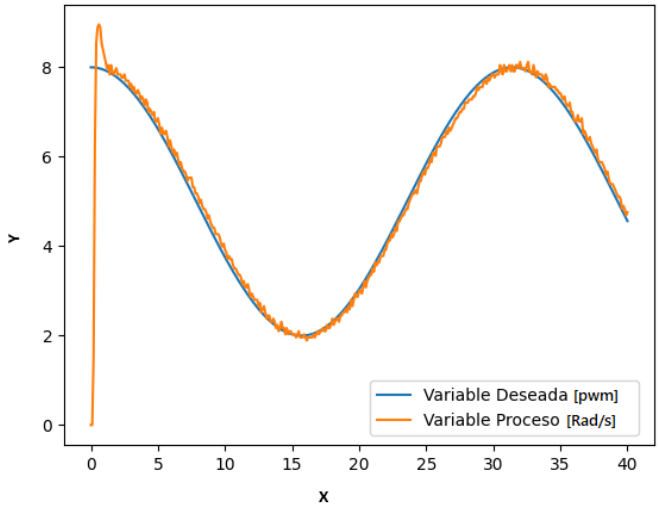
Figura 41. Señal coseno obtenida algoritmo de identificación sintonización fina.



Fuente: Elaboración propia

Al realizar la sintonización fina se puede observar la variable de control y la variable de proceso mostrado en la figura 42, se puede contemplar que el resultado del control PID es mucho mejor en comparación al obtenido en la figura 40.

Figura 42. Sintonización fina.



Fuente: Elaboración propia

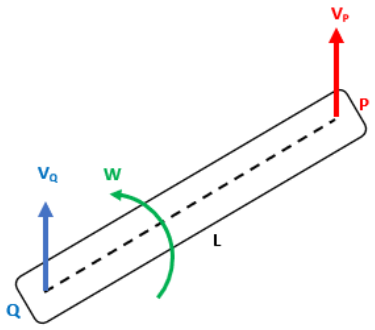
Al analizar los resultados obtenidos y constatando que son bastante buenos solo queda utilizar estos parámetros en cada uno de los tres motores dando que todos poseen las mismas características.

2.2.4. Análisis Odometrico Del Robot Móvil

La odometría permite estimar la posición del robot en un plano cartesiano, para ello se recurre a información que incluye datos como la rotación de cada rueda y algunos parámetros propios del robot móvil.

Por tal razón, es conveniente recurrir a la ecuación de movimiento para cuerpos rígidos la cual está representada en la figura 43.

Figura 43. Diagrama de velocidades de un cuerpo rígido.

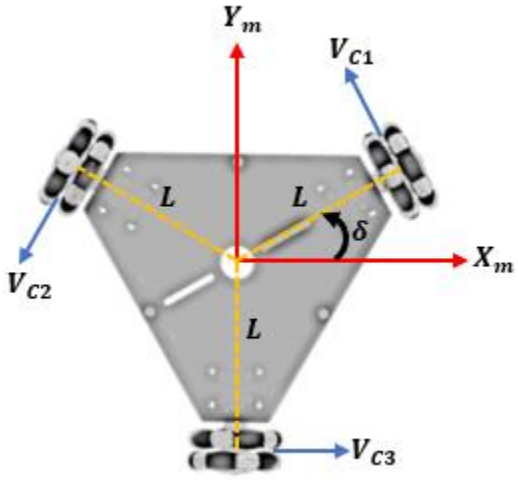


De la figura 43 tomando en cuenta la ubicación del punto P y la ubicación del punto Q y considerando que el elemento de longitud L rota alrededor del punto Q el cual presenta una velocidad lineal se obtiene que la velocidad en el punto P se encuentra dada por la expresión (48).

$$V_P = V_Q + W * L \tag{48}$$

Para calcular las velocidades de cada una de las ruedas se parte del sistema de coordenadas mostradas en la figura 44.

Figura 44. Componentes de velocidad de las ruedas.



Fuente: Elaboración propia

Teniendo en cuenta cada una de las velocidades angulares de las ruedas se obtienen las expresiones (49), (50) y (51).

$$v_{c1} = v_{01} + w_1 * r_{c1} \quad (49)$$

$$v_{c2} = v_{02} + w_2 * r_{c2} \quad (50)$$

$$v_{c3} = v_{03} + w_3 * r_{c3} \quad (51)$$

Como se observa en la figura 44, el robot móvil tiene un sistema de referencia el cual es propio y cuyas componentes son x_m y y_m.

Teniendo en cuenta que entre las ruedas y el piso no existe deslizamiento alguno, es decir $v_{01} = v_{02} = v_{03} = 0$ y además que el radio de las ruedas es el mismo $r_{c1} = r_{c2} = r_{c3} = r$ las expresiones (49), (50) y (51) se pueden reescribir tal como se muestran en las ecuaciones (52), (53) y (54).

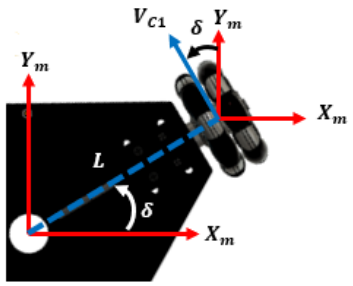
$$v_{c1} = w_1 * r \quad (52)$$

$$v_{c2} = w_2 * r \quad (53)$$

$$v_{c3} = w_3 * r \quad (54)$$

Al poner el sistema de coordenadas del robot sobre las ruedas 1 y 2 se obtiene la representación mostrada en la figura 45.

Figura 45. Componentes de velocidad rueda 1.



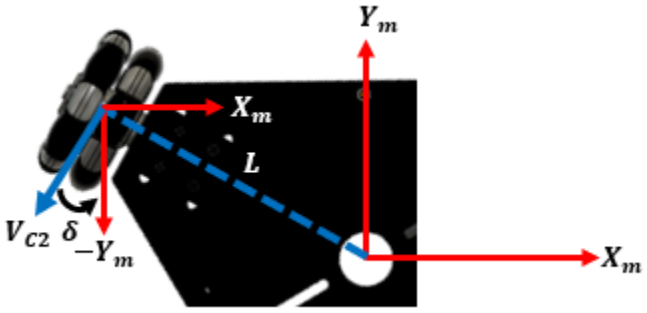
Fuente: Elaboración propia

Al observar la figura 45 y realizar las correspondientes consideraciones obtenemos las ecuaciones (55) y (56).

$$\dot{X_m} = -V_{C1}\sin(\delta) \quad (55)$$

$$\dot{Y_m} = V_{C1}\cos(\delta) \quad (56)$$

Figura 46. Componentes de velocidad rueda 2.



Fuente: Elaboración propia

Seguidamente, analizando la figura 46 y al realizar las correspondientes consideraciones en la rueda se determinan las expresiones (57) y (58):

$$\dot{X_m} = -V_{C2}\sin(\delta) \quad (57)$$

$$\dot{Y_m} = -V_{C2}\cos(\delta) \quad (58)$$

De la figura 44 se puede observar que V_{c3} está alineado completamente con la componente de velocidad X_m de tal manera que al tomar en cuenta la velocidad angular del propio robot se obtienen las soluciones mostradas en las ecuaciones (59), (60) y (61).

$$V_{C1} = L * \dot{\theta} + V_1 \quad (59)$$

$$V_{C2} = L * \dot{\theta} + V_2 \quad (60)$$

$$V_{C3} = L * \dot{\theta} + \dot{X_m} \quad (61)$$

Cabe resaltar que V_1 y V_2 son iguales a las componentes U_1 y U_2. Ahora bien, es necesario que todo quede expresado en términos de $\dot{X_m}$ y $\dot{Y_m}$.

Efectuando las ecuaciones (55) y (56) y al desarrollarse la solución en las expresiones (62) y (63) la componente de velocidad V_1 queda expresada como se muestra en la ecuación (64).

$$V_1 = V_1 sin^2(\delta) + V_1 cos^2(\delta) \quad (62)$$

$$V_1 = (V_1 sin(\delta)) * sin(\delta) + (V_1 cos(\delta)) * cos(\delta) \quad (63)$$

$$V_1 = \dot{X_m} * sin(\delta) + \dot{Y_m} * cos(\delta) \quad (64)$$

Efectuando las ecuaciones (57) y (58), y al aplicar su respectiva solución como se muestran en las expresiones (65) y (66) la componente de velocidad V_2 queda expresada en la solución de la ecuación (67).

$$V_2 = V_2 sin^2(\delta) + V_2 cos^2(\delta) \quad (65)$$

$$V_2 = (V_2 sin(\delta)) * sin(\delta) + (V_2 cos(\delta)) * cos(\delta) \quad (66)$$

$$V_2 = -\dot{X}_m * sin(\delta) - \dot{Y_m} * cos(\delta) \quad (67)$$

Al reescribir las ecuaciones (59), (60) y (61) se obtienen las expresiones (68), (69) y (70).

$$V_{C1} = L * \dot{\theta} + \dot{X_m} * sin(\delta) + \dot{Y_m} * cos(\delta) \quad (68)$$

$$V_{C2} = L * \dot{\theta} - \dot{X}_m * sin(\delta) - \dot{Y_m} * cos(\delta) \quad (69)$$

$$V_{C3} = L * \dot{\theta} + \dot{X_m} \quad (70)$$

Seguidamente se reemplazan los valores de V_{C1}, V_{C2} y V_{C3} de las expresiones (52), (53) y (54) en las expresiones (68), (69) y (70) dando como resultados las ecuaciones (71), (72) y (73).

$$w_1 * r = L * \dot{\theta} + \dot{X_m} * sin(\delta) + \dot{Y_m} * cos(\delta) \quad (71)$$

$$w_2 * r = L * \dot{\theta} - \dot{X}_m * sin(\delta) - \dot{Y_m} * cos(\delta) \quad (72)$$

$$w_3 * r = L * \dot{\theta} + \dot{X_m} \quad (73)$$

Al expresar las ecuaciones (71), (72) y (73) de forma matricial obtenemos la expresión (74).

$$r\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} -sin(\delta) & cos(\delta) & L \\ -cos(\delta) & -sin(\delta) & L \\ 1 & 0 & L \end{bmatrix}\begin{bmatrix} \dot{X_m} \\ \dot{Y_m} \\ \dot{\theta} \end{bmatrix} \quad (74)$$

De la ecuación (74) es posible obtener las velocidades angulares de cada rueda quedando representado como se muestra en la expresión (75).

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \frac{1}{r}\begin{bmatrix} -sin(\delta) & cos(\delta) & L \\ -cos(\delta) & -sin(\delta) & L \\ 1 & 0 & L \end{bmatrix}\begin{bmatrix} \dot{X_m} \\ \dot{Y_m} \\ \dot{\theta} \end{bmatrix} \quad (75)$$

2.2.5. Diseño De Sistema De Control

A partir del modelo matemático obtenido con sus respectivas restricciones al conseguir el comportamiento deseado se debe obtener una ley de control de tal manera que el sistema realimentando logre un comportamiento deseado.

Para ello se deber resolver los problemas relacionados al control de movimiento como lo son.

Posición: La idea fundamental en este caso es ubicar al robot en un punto dado de referencia y una orientación deseada.

Seguimiento De Trayectoria: Se debe lograr que el robot siga una determinada trayectoria, es decir el robot debe alcanzar y seguir con un error de cero en lo más posible una línea parametrizada en el tiempo.

Seguimiento De Camino: El robot debe seguir una ruta o trayectoria sin ninguna especificación en el tiempo.

2.2.6. Diseños De Sistemas De Control No Lineal Por Método De Lyapunov

Para el diseño de sistemas de control se basó en el *Paper* "Teoría cuantitativa de las ecuaciones diferenciales".(Catsigeras, 1998)

En este artículo se detallan las ecuaciones necesarias para el diseño de un control asintóticamente estable. Es importante realizar un sistema de control no lineal con el objetivo de tener en cuenta las no linealidades en el diseño de control y no como los típicos controladores lineales que trabajan en un cierto rango de trabajo como el clásico PID.

- La teoría de estabilidad de Lyapunov es una herramienta estándar y una de las más importantes en el análisis de sistemas no lineales y sistemas que varían en el tiempo.

-La teoría de Lyapunov permite analizar la estabilidad de un sistema de control y también puede ser aplicada como una estrategia para el diseño de controladores.

-El objetivo de este método es encontrar una función candidata de Lyapunov de tal modo que cumplan con ciertas condiciones asegurando así la estabilidad de sus estados de equilibrio.

3.2.7. Estados De Equilibrio

-Los estados de equilibrio son llamados también puntos estacionarios, puntos constantes o puntos de reposo, ya que si el sistema se ubica inicialmente en $x = x_e$, luego permanecerá en $x(t) = x_e$ para todo tiempo $t > 0$.

-En el caso de diseño de controladores los estados son los errores y el punto de equilibrio es cero.

3.2.8. Función Candidata De Lyapunov

La función candidata de Lyapunov $V(x)$ es una función de los estados x(errores) y cumple las siguientes condiciones.

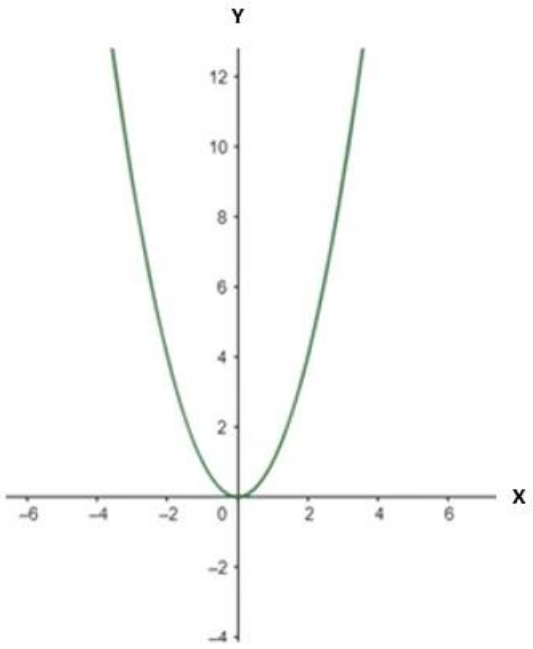
1. V(x) Es continua y tiene derivas continuas.

2. V(x) = 0 para $x = 0$
3. V(x) > 0 para $x \neq 0$

Se puede proponer cualquier función siempre y cuando esta cumpla las condiciones dadas anteriormente.

La función candidata propuesta es la mostrada en la figura 47.

Figura 47. Función candidata.



Con base a la función candidata se deber determinar si el sistema es estable.

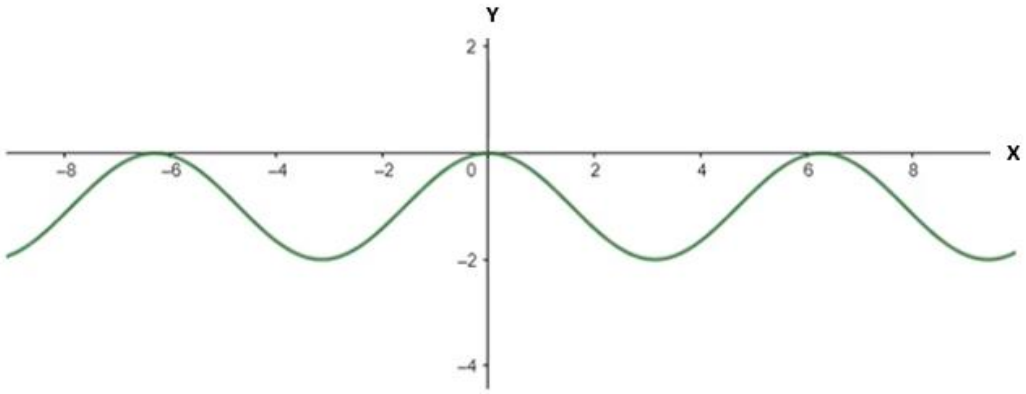
Un sistema se denomina estable alrededor de un punto de equilibrio en el origen si satisface las siguientes condiciones.

1. $\frac{dV(x)}{dt} = 0 \; para \; x = 0$
2. $\frac{dV(x)}{dt} \leq 0 \; para \; x \neq 0$

A continuación, en la figura 48, se muestra su representación gráfica para que el sistema pueda ser considerado estable teniendo en cuenta las condiciones previamente dadas.

Esta función representa a una función seno indefinida negativa a la cual al momento de aplicar las condiciones 1 y 2 se puede comprobar que estas se cumplen.

Figura 48. Función seno negativa indefinida.



Este tipo de análisis de estabilidad puede ser suficiente para un controlador dependiendo de las características y necesidades que se necesiten. Este sistema solo va a permitir que los errores estén cerca a cero mas no asegura que el error tienda a cero por eso se hace necesario el análisis para un sistema asintóticamente estable.

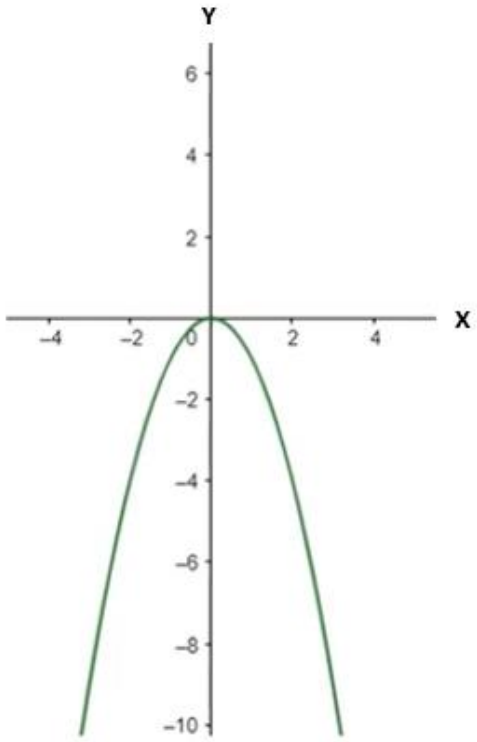
Un sistema se denomina asintóticamente estable alrededor de su punto de equilibrio en el origen si satisface las siguientes condiciones.

1. $\frac{dV(x)}{dt} = 0 \; para \; x = 0$
2. $\frac{dV(x)}{dt} < 0 \; para \; x \neq 0$

Si $x \to 0$ cuando $t \to \infty$, se dice que el sistema es asintóticamente estable.

La representación gráfica donde se cumplen las condiciones anteriores está dada en la figura 49 donde se representa una parábola invertida.

Figura 49. Parábola invertida.



Al observar la figura 49 se puede analizar que solamente para $x = 0$, es decir, para errores iguales a cero solamente en ese punto la derivada de la función candidata de Lyapunov es igual a cero por lo tanto si los errores tienden a cero cuando t tiende al infinito se dice que el sistema es asintóticamente estable.

Por lo tanto, si se cumplen las condiciones previamente dadas, es decir, lo que se desea en un controlador, por ende, este tipo de análisis es el que se utiliza para realizar el diseño de los controladores.

Parámetros de Denavit – Hartenberg

Para el cálculo de la cinemática directa se utilizan los parámetros de Denavit – Hartenberg los cuales establecen una serie de procedimientos que permiten obtener de manera más sencilla los sistemas de coordenadas en cada uno de los eslabones de la cadena articulada del robot necesarios para la construcción de la matriz de transformación.

Algoritmo Denavit – Hartenberg:

Este algoritmo es tomado como referencia del libro "fundamentos de robótica 2 edición pág. (126)".(Boyer et al., 1982)

En este se detallan los pasos necesarios para el análisis de las articulaciones presentes en el robot. Estos pasos se muestran a continuación.

D-H 1.- Numerar los eslabones comenzando con 1 (primer eslabón móvil de la cadena) y acabando con n (último eslabón móvil). Se numerará como eslabón 0 a la base fija del robot.

D-H 2.- Numerar cada articulación comenzando por 1 (la correspondiente al primer grado de libertad) y acabando en n

D-H 3.- Localizar el eje de cada articulación. Si ésta es rotativa, el eje será su propio eje de giro. Si es prismática, será el eje a lo largo del cual se produce el desplazamiento.

D-H 4.- Para i de 0 a n-1 situar el eje z_i sobre el eje de la articulación i+1.

D-H 5.- Situar el origen del sistema de la base {S0} en cualquier punto del eje z_0. Los ejes x_0 e y_0 se situarán de modo que formen un sistema dextrógiro con z0

D-H 6.- Para i de 1 a n-1, situar el sistema {Si} (solidario al eslabón i) en la intersección del eje z_i con la línea normal común a $z_i - 1$ y z_i. Si ambos ejes se cortasen se situaría {Si} en el punto de corte. Si fuesen paralelos {Si} se situaría en la articulación i+1

D-H 7.- Situar x_i en la línea normal común a $z_i - 1$ y z_i

D-H 8.- Situar y_i de modo que forme un sistema dextrógiro con x_i y z_i.

D-H 9.- Situar el sistema {Sn} en el extremo del robot de modo que z_n coincida con la dirección de $z_n - 1$ y x_n sea normal a $z_n - 1$ y z_n.

D-H 10.- Obtener θ_i como el ángulo que hay que girar en torno a $z_i - 1$ para que $x_i - 1$ y xi queden paralelos.

D-H 11.- Obtener di como la distancia, medida a lo largo de $z_n - 1$, que habría que desplazar {Si-1} para que x_i y $x_i - 1$ quedasen alineados.

DH 12.- Obtener a_i como la distancia medida a lo largo de xi (que ahora coincidiría con xi1) que habría que desplazar el nuevo {Si-1} para que su origen coincidiese con {Si}.

DH 13.- Obtener α i como el ángulo que habría que girar en torno a xi (que ahora coincidiría con xi-1), para que el nuevo {Si-1} coincidiese totalmente con {Si}.

DH 14.- Obtener las matrices de transformación i-1Ai

DH 15.- Obtener la matriz de transformación entre la base y el extremo del robot T = 0A1 1A2 ... n-1An.

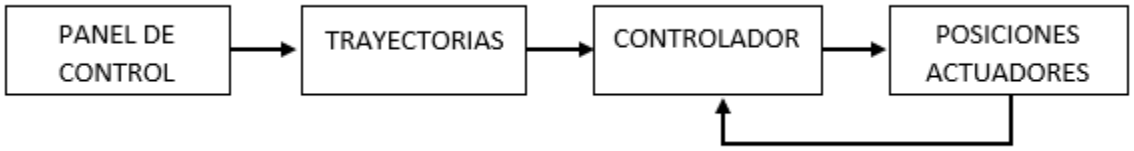
DH 16.- La matriz T define la orientación (submatriz de rotación) y posición (submatriz de traslación) del extremo referido a la base en función de las n coordenadas articulares.

2.3.Modelo cinemático Robot antropomórfico

A continuación, en la figura 50 se plantea el esquema de control que rige al sistema, en este se obtienen las posiciones angulares de los actuadores necesarias para el movimiento de cada articulación.

El bloque panel de control es donde se ejecutan las diferentes opciones por medio de la interfaz de usuario para que el robot se ponga en marcha. Así mismo, en el bloque de trayectorias, se encentran los puntos propuestos para que el robot haga el recorrido necesario para llegar al punto asignado. Por su parte, en el bloque controlador es donde se analiza la información enviada desde el panel de control obteniendo las coordenadas articulares para cada articulación, finalmente, en el bloque de posiciones de actuadores se ejecutan cada una de las posiciones angulares encontradas para generar el movimiento del robot y llegar al punto especifico.

Figura 50. Esquema de operación del robot antropomórfico.



Fuente: Elaboración propia

2.3.1. Cinemática Directa Robot Antropomórfico

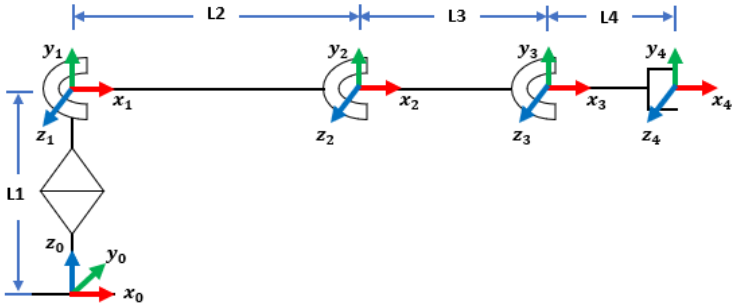
La cinemática directa permite mostrar información respecto a la posición y orientación de los ángulos del robot una vez sus variables encargadas de otorgar la posición u orientación a las

articulaciones se les asigna cierto valor, en otras palabras, permite al usuario conocer información relevante entorno a la ubicación de los puntos extremos del robot.

Existen dos métodos asociados a la cinemática directa los cuales corresponden a métodos geométricos y matrices de transformación homogénea, que en este caso fue usada esta última.

Para calcular la cinemática directa se parte de la representación esquemática del robot donde al aplicar el algoritmo de Denavit – Hartenberg se obtienen los sistemas de referencia de cada uno de los eslabones descritos en la figura 51.

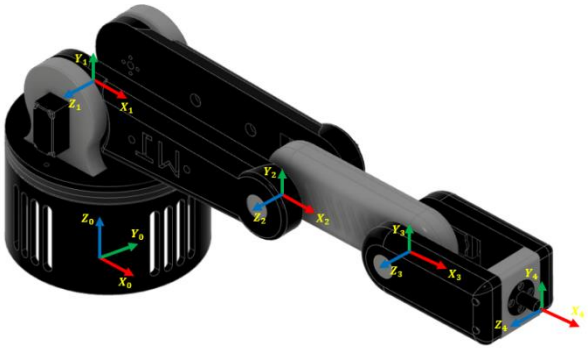
Figura 51. Sistemas elegidos.



Fuente: Elaboración propia

En la figura 49 se representan los sistemas coordenados presentes en cada articulación del robot en su diseño 3D y los cuales se hallaron por medio de los parámetros de D-H.

Figura 52. Robot antropomórfico 4GDL.



Fuente: Elaboración propia.

A continuación, en la tabla 10 se organizan los parámetros de cada una de las articulaciones obtenidos para la cinemática directa del robot y los cuales son necesarios para el análisis de cada una de las matrices de transformación.

Tabla 10. *Parámetros de Denavit - Hartenberg.*

ARTICULACIONES	$\boldsymbol{\theta}$	**D**	**A**	$\boldsymbol{\alpha}$
1	q1	L1	0	π
2	q2	0	L2	0
3	q3	0	L3	0
4	q4	0	L4	0

Por su parte, en la expresión (76) se representa la matriz de D-H la cual determina las matrices en cada uno de los sistemas obtenidos ya que permite relacionar la posición y traslación considerando la perspectiva nula y el escalón unitario de los diferentes eslabones que componen el robot.

$$DH = \begin{bmatrix} cos\theta_i & -cos\alpha_i sen\theta_i & sen\alpha_i sen\theta_i & a_i cos\theta_i \\ sen\theta_i & cos\alpha_i cos\theta_i & -sen\alpha_i cos\theta_i & a_i sen\theta_i \\ 0 & sen\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{76}$$

Partiendo de los parámetros obtenidos en la tabla 10 y de aplicar la matriz de D-H de la ecuación (76) con ayuda de Python al simplificar los datos se tiene la expresión (77) del sistema solidario al primer eslabón respecto al sistema de referencia solidario a la base.

$${}^{0}\mathrm{A}_{1} = \begin{bmatrix} \cos(q1) & -\cos(\pi/2) sen(q1) & sen(\pi/2) sen(q1) & 0 \\ sen(q1) & \cos(\pi/2) cos(q1) & -sen(\pi/2)\cos(q1) & 0 \\ 0 & sen(\pi/2) & \cos(\pi/2) & L1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{77}$$

De igual manera se obtiene la matriz del sistema de referencia solidario al segundo eslabón representada en la expresión (78).

$$ {}^{1}\mathrm{A}_{2} = \begin{bmatrix} \cos(q2) & -sen(q2) & 0 & L2(\cos(q2)) \\ sen(q2) & cos(q2) & 0 & L2(\mathrm{sen}(q2)) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{78} $$

Seguidamente se obtiene la matriz mostrada en (79) y cuyo sistema es solidario al tercer eslabón.

$$ {}^{2}\mathrm{A}_{3} = \begin{bmatrix} \cos(q3) & -sen(q3) & 0 & L3(\cos(q3)) \\ sen(q3) & cos(q3) & 0 & L3(\mathrm{sen}(q3)) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{79} $$

Por último, se obtiene la matriz mostrada en la expresión (80), este sistema es solidario al cuarto eslabón.

$$ {}^{3}\mathrm{A}_{4} = \begin{bmatrix} \cos(q4) & -sen(q4) & 0 & L4(\cos(q4)) \\ sen(q4) & cos(q4) & 0 & L4(\mathrm{sen}(q4)) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{80} $$

Finalmente, para obtener la matriz ${}^{0}T_{4}$ se emplea la ecuación mostrada en la expresión (81) que indica el sistema final con respecto a la base.

$$ {}^{0}T_{4} = \begin{bmatrix} {}^{0}\mathrm{A}_{1}\ {}^{1}\mathrm{A}_{2}\ {}^{2}\mathrm{A}_{3}\ {}^{3}\mathrm{A}_{4} \end{bmatrix} \tag{81} $$

Al operar la ecuación (81) con ayuda de Python y simplificando sus valores se obtiene la expresión (82).

$$ {}^{0}T_{4} = \begin{bmatrix} \cos(q2+q3+q4)*\cos(q1) & -sen(q2+q3+q4)*\cos(q1) & \sin(q1) & cos(q1)*(L3*\cos(q2+q3)+L2*\cos(q2)+L4+\cos(q2+q3+q4)) \\ \cos(q2+q3+q4)*\cos(q1) & -sen(q2+q3+q4)*\cos(q1) & -\cos(q1) & sin(q1)*(L3*\cos(q2+q3)+L2*\cos(q2)+L4+\cos(q2+q3+q4)) \\ \sin(q2+q3+q4) & \cos(q2+q3+q4) & 0 & L1+L3*\cos(q2+q3)+L2*\cos(q2)+L4+\cos(q2+q3+q4) \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

De la matriz ${}^{0}T_{4}$ se extrae la posición en los puntos $X, Y\ y\ Z$ los cuales se muestran en las ecuaciones (83), (84) y (85).

$$ P_x = cos\,(q1)*(L3*\cos(q2+q3)+L2*\cos(q2)+L4+\cos(q2+q3+q4)) \tag{83} $$

$$ P_y = sin\,(q1)*(L3*\cos(q2+q3)+L2*\cos(q2)+L4+\cos(q2+q3+q4)) \tag{84} $$

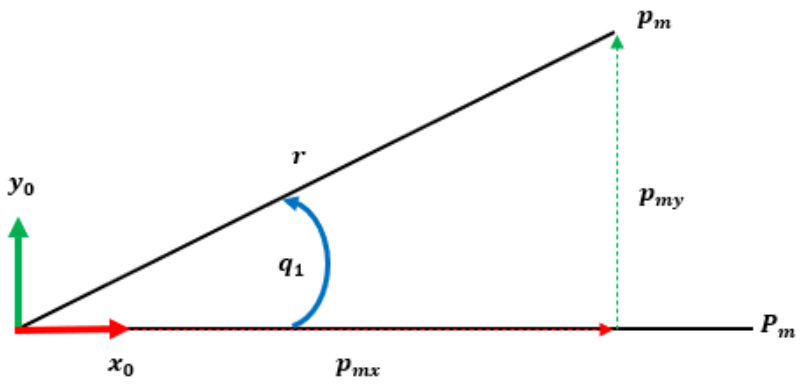
$$P_z = L1 + L3 * \cos(q2 + q3) + L2 * \cos(q2) + L4 + \cos(q2 + q3 + q4) \quad (85)$$

2.3.2. Cinemática Inversa Robot Antropomórfico

La finalidad de la cinemática inversa es ayudar a encontrar aquellos valores de las coordenadas que las articulaciones del robot definidas como $[q]$ deben tomar, todo con el objetivo de posicionar u orientar sus extremos teniendo en cuenta cierta localización espacial.

Para calcular la cinemática inversa se extiende el brazo y se gira una posición mayor a cero y menor a 90 grados vista desde la parte superior quedando representada como se muestra en la figura 53.

Figura 53. Robot antropomórfico 4GDL.



Fuente: Elaboración propia.

Para el cálculo de q_1 se parte del triángulo rectángulo de la figura 53 y luego de aplicar relaciones trigonométricas como la función arco tangente se obtiene la expresión mostrada en (86).

$$q1 = atan2\ (Pmy, Pmx) \quad (86)$$

Por teorema de Pitágoras aplicado al triangulo rectángulo mostrado en la figura 53 se obtiene la expresión mostrada en (87), en donde, para despejar r y luego de una serie de cálculos se obtiene la ecuación (88).

$$r^2 = p_{mx}^2 + p_{my}^2 \quad (87)$$

$$r = \sqrt{p_{mx}^2 + p_{my}^2} \quad (88)$$

Utilizando el teorema del coseno se pueden hallar los valores de Px', Py', Pz' respecto al efector final quedando representados como se muestran en las ecuaciones (89), (90) y (91).

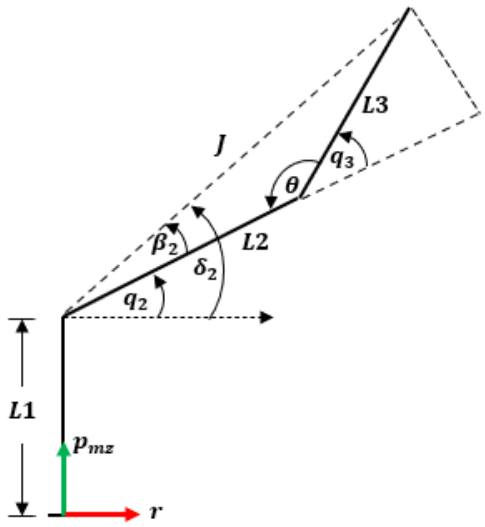
$$Px' = Px - L4\ \cos(\alpha)\cos(q1) \quad (89)$$

$$Py' = Py - L4\cos(\alpha)\sin(q1) \quad (90)$$

$$Pz' = Pz - L4\ sen\ (\alpha) \quad (91)$$

Seguidamente, para calcular el valor de q_3 y q_2 se ubica al observador frente a la parte lateral, pudiendo mirar al robot como se muestra en la figura 54.

Figura 54. Robot antropomórfico 4GDL.



Fuente: Elaboración propia.

Al aplicar relaciones trigonométricas para determinar J en la figura 54, se establece la ecuación mostrada en la expresión (92), esta a su vez después de varios cálculos queda representada como se muestra en la fórmula (93).

$$J^2 = r^2 + (Pmz - L1)^2 \quad (92)$$

$$J = \sqrt{r^2 + (Pmz - L1)^2} \quad (93)$$

Por otro lado, también se hace uso de la ley de cosenos para hallar la expresión (94).

$$J^2 = L2^2 + L3^2 - 2 * L2 * L3 * cos(\theta) \quad (94)$$

Para hallar el valor de θ mostrado en la figura 54, después de una serie de análisis por medio de relaciones trigonométricas se obtiene la expresión (95) de la cual se despeja el ángulo y se obtiene la ecuación (96).

$$\theta + q3 = 180 \quad (95)$$

$$\theta = 180 - q3 \quad (96)$$

Al remplazar la ecuación (96) en el ángulo θ de la expresión (94) se obtiene la formula mostrada en (97).

$$\cos(\theta) = \cos(180 - q3) \quad (97)$$

Por identidad trigonométrica se obtiene la representación mostrada en la ecuación (98).

$$\cos(\theta) = \cos(180) * \cos(q3) + sen(180) * sen(q3) \quad (98)$$

Luego de simplificar los valores de la expresión (98) se tiene como resultado la ecuación (99).

$$\cos(\theta) = -\cos(q3) \quad (99)$$

Al remplazar la ecuación (94) en (99) y se obtiene la expresión (100).

$$J^2 = L2^2 + L3^2 + 2 * L2 * L3 * cos(q3) \quad (100)$$

Al despejar $cos(q3)$ en (100) se tiene como solución la ecuación (101).

$$\cos(q3) = \frac{J^2 - L2^2 - L3^2}{2(L2 * L3)} \quad (101)$$

Al remplazar (92) en (101) esta queda representada como se muestra en la expresión (102).

$$\cos(q3) = \frac{(r^2 + (p_{mz} - L1)^2 - L2^2 - L3^2)}{2(L2 * L3)} \quad (102)$$

Por identidad trigonométrica se obtiene la ecuación (103).

$$\text{sen}(q3) = \pm\sqrt{1 - Cos^2(q_3)} \quad (103)$$

De igual forma, se tiene en cuenta la configuración codo arriba y codo abajo como se muestra en la ecuación (104) y cuya raíz genera dos soluciones.

$$\text{sen}(q3) = codo\sqrt{1 - Cos^2(q_3)} \quad (104)$$

Para calcular $q3$ se tubo en cuenta la expresión arco tangente quedando representada como se muestra en la ecuación (105), esta permite una solución más rápida a la hora de desarrollar los códigos

$$q3 = atan2(sen(q3), \cos(q3)) \quad (105)$$

Para el cálculo de q_2 teniendo en cuenta la figura 54, luego de un análisis trigonométrico y con ayuda de la función arco tangente se obtienen las expresiones (106) y (107) siendo estas determinantes para la obtención de la ecuación (109).

$$\beta_2 = atan2(L3 * sen(q3), \text{L3} * \cos(q3) + L2) \quad (106)$$

$$\delta_2 = atan2\left(p_{mz} - L1, \sqrt{p_{mx}^2 + p_{my}^2}\right) \quad (107)$$

$$q2 = \alpha - \beta \quad (108)$$

Para q_4 se calcula la diferencia de los ángulos del efector final con respecto a $q2$ y $q3$ teniendo como resultado la expresión (109).

$$q4 = \alpha - q2 - q3 \quad (109)$$

2.3.3. Jacobiano

Según (Barrientos y otros, 2007) la matriz Jacobiana de un robot, relaciona el vector de velocidades articulares (q_1, q_2, q_n) con otro vector de velocidades expresado en un espacio distinto. Es importante mencionar que a través de esta matriz se obtiene La relación entre ambas velocidades (articulares y linear-angular del extremo) cabe mencionar que en ambos casos, la matriz Jacobiana directa permite conocer una expresión de las velocidades del extremo del robot a partir de los valores de las velocidades de cada articulación, mientras que la matriz Jacobiana inversa solo permitirá conocer las velocidades articulares necesarias para obtener un vector concreto de velocidades del extremo.

$$\begin{pmatrix} Px \\ Py \\ Pz \end{pmatrix} = \begin{pmatrix} \frac{dPx}{dq1} & \frac{dPx}{dq2} & \frac{dPx}{dq3} & \frac{dPx}{dq4} \\ \frac{dPy}{dq1} & \frac{dPy}{dq2} & \frac{dPy}{dq3} & \frac{dPy}{dq4} \\ \frac{dPz}{dq1} & \frac{dPz}{dq2} & \frac{dPz}{dq3} & \frac{dPz}{dq4} \end{pmatrix} \begin{pmatrix} q1 \\ q2 \\ q3 \\ q4 \end{pmatrix} \quad (110)$$

De la ecuación (89) se deriva p_x, respecto a q_1, q_2, q_3 y q_4 obteniendo como resultado las ecuaciones (111), (112), (113) y (114).

$$\frac{dPx}{dq1} = -sin(q1) * (L3 * cos(q2 + q3) + L2 * cos(q2) + L4 * cos(q2 + q3 + q4)) \quad (111)$$

$$\frac{dPx}{dq2} = -cos(q1) * (L3 * sin(q2 + q3) + L2 * sin(q2) + L4 * sin(q2 + q3 + q4)) \quad (112)$$

$$\frac{dPx}{dq3} = -\cos(q1) * (L3 * \sin(q2 + q3) + L4 * \sin(q2 + q3 + q4)) \quad (113)$$

$$\frac{dPx}{dq4} = -L4 * sin(q2 + q3 + q4) * cos(q1) \quad (114)$$

De igual forma, al derivar p_y en la ecuación (90) respecto a q_1, q_2, q_3 y q_4 se tienen las expresiones (116), (116), (117) y (118).

$$\frac{dPy}{dq1} = cos(q1) * (L3 * cos(q2 + q3) + L2 * cos(q2) + L4 * cos(q2 + q3 + q4)) \quad (115)$$

$$\frac{dPy}{dq2} = -sin(q1) * (L3 * sin(q2 + q3) + L2 * sin(q2) + L4 * sin(q2 + q3 + q4)) \quad (116)$$

$$\frac{dPy}{dq3} = -sin(q1) * (L3 * sin(q2 + q3) + L4 * sin(q2 + q3 + q4)) \quad (117)$$

$$\frac{dPy}{dq4} = -L4 * \sin(q2 + q3 + q4) * \sin(q1) \quad (118)$$

Finalmente, al derivar p_z de la expresión (91) respecto a q_1, q_2, q_3 y q_4 se tiene como resultados las ecuaciones (119), (120), (121) y (122).

$$\frac{dPz}{dq1} = 0 \quad (119)$$

$$\frac{dPz}{dq2} = L3 * cos(q2 + q3) + L2 * cos(q2) + L4 * cos(q2 + q3 + q4) \quad (120)$$

$$\frac{dPz}{dq3} = L3 * cos(q2 + q3) + L4 * cos(q2 + q3 + q4) \quad (121)$$

$$\frac{dPz}{dq4} = L4 * cos(q2 + q3 + q4) \quad (122)$$

CAPÍTULO IV

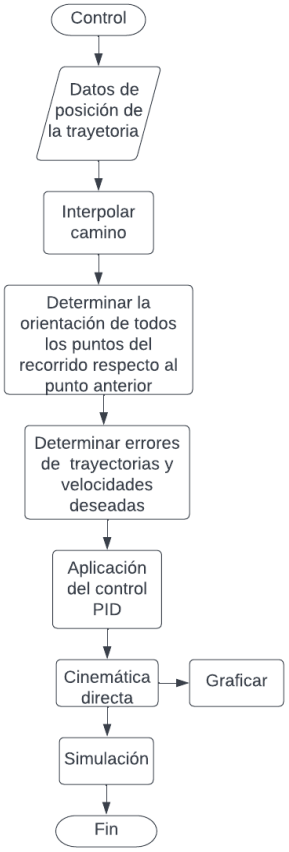
4. Validación de resultados

4.1.Validación de resultados robot móvil omnidireccional

4.1.1. Diagrama de flujo para el control de rutinas

Para la prueba de simulación de los códigos de programación realizados en Python haciendo uso del modelo cinemático calculado, en el diagrama de flujo mostrado en la figura 55, se describe el comportamiento de todo el proceso desde la interpolación de los caminos establecidos, el cálculo de los errores en las trayectorias, el diseño del control y el entorno para la lograr la simulación.

Figura 55. Diagrama de flujo para el control de rutinas.



Fuente: Elaboración propia.

4.1.2. Prueba simulada para verificar el rendimiento del controlador

Para ello se utiliza un tiempo de simulación de 50 segundos

Condiciones iniciales

$x_i = 0$

$y_i = 0$

$phi_i = 0$

Donde:

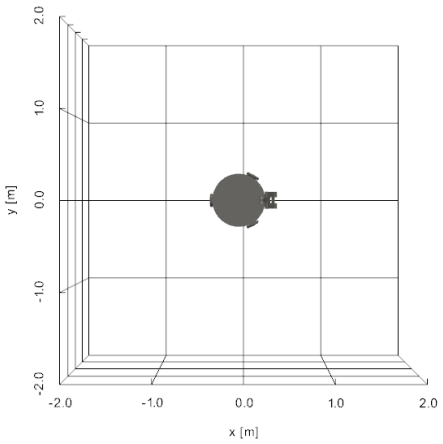
$x_i = Posición\ inicial\ en\ el\ eje\ x\ en\ metros$

$y_i = Posición\ inicial\ en\ el\ eje\ y\ en\ metros$

$alfa_i = Ángulo\ de\ orientacion\ inicial\ en\ radianes$

En la figura 56 se muestra el robot en su posición inicial como está indicado anteriormente en las condiciones iniciales.

Figura 56. Posición inicial robot



Fuente: Elaboración propia.

Posición final

$x_f = 1m$

$y_f = -1m$

$alfa_f = 45°$

Donde:

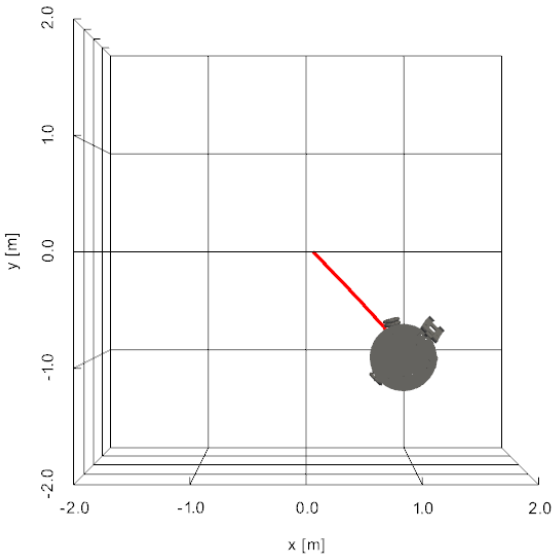
$x_f = Posición\ final\ deseada\ en\ el\ eje\ x\ en\ metros$

$y_f = Posición\ final\ deseada\ en\ el\ eje\ y\ en\ metros$

$alfa_f = Ángulo\ de\ orientacion\ final\ deseada\ en\ radianes$

Al enviar el robot móvil a las coordenadas propuestas en la posición final se observa que el robot alcanza la ubicación de $1mt$ en X y $-1mt$ en Y al igual que un ángulo de 45 grados como se muestra en la figura 57.

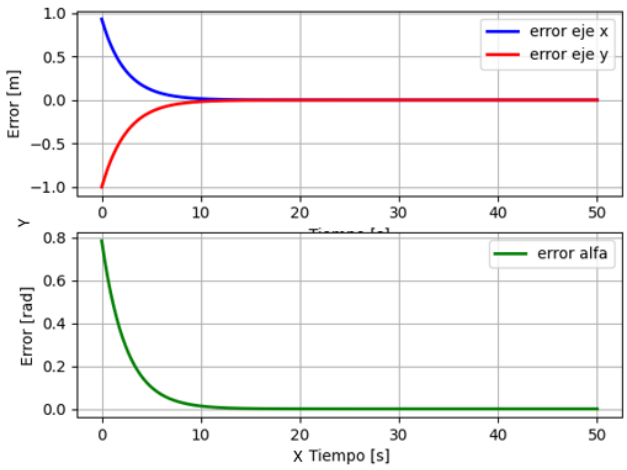
Figura 57. Prueba simulada de posición



Fuente: Elaboración propia

En la gráfica 58 se observa que los errores tanto en el eje x como en el eje y convergen a cero al igual que el error en el ángulo alfa en aproximadamente 10 segundos.

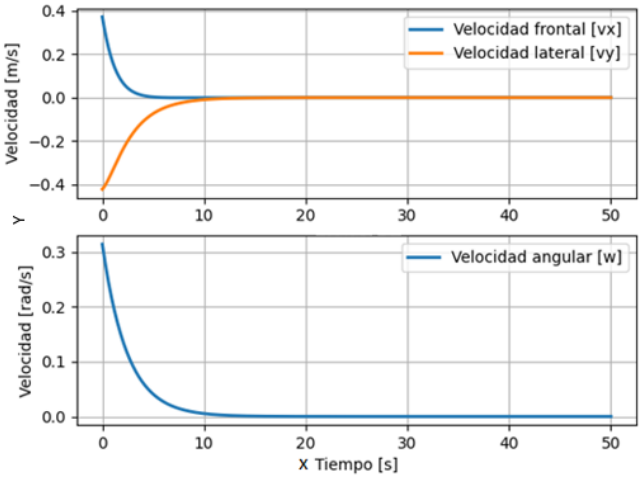
Figura 58. Errores en los ejes



Fuente: Elaboración propia

En la figura 59 se observan las acciones de control donde la máxima es de 0.4 aproximadamente y la mínima es de -0.4 aproximadamente esto para la velocidad frontal en x y la velocidad frontal en y, además se observa que la velocidad angular alcanza un máximo de 0.3 radianes por segundo aproximadamente.

Figura 59. Velocidades



Fuente: Elaboración propia

4.1.3. Control de posición prueba simulada y prueba experimental

A continuación, se validan el control de posición implementando para observar el comportamiento a través de pruebas experimentales con ayuda de pruebas simuladas.

Condiciones iniciales

$x_i = 0$

$y_i = 0$

$phi_i = 0$

Donde:

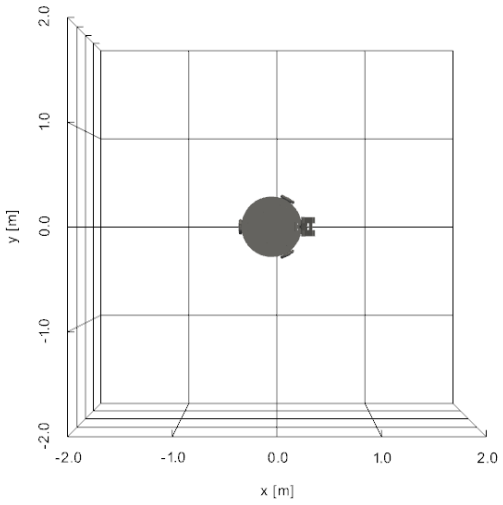
$x_i = Posición\ inicial\ en\ el\ eje\ x\ en\ metros$

$y_i = Posición\ inicial\ en\ el\ eje\ y\ en\ metros$

$alfa_i = Ángulo\ de\ orientacion\ inicial\ en\ radianes$

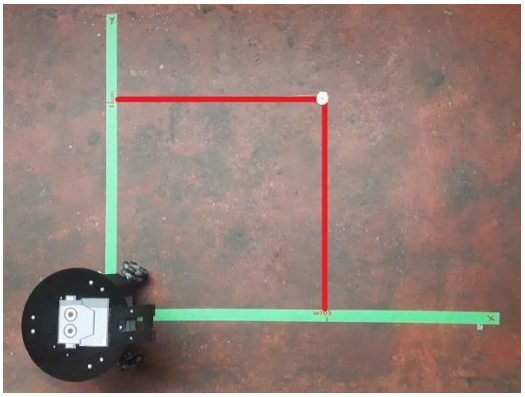
Respectivamente, en la figura 60 se representa al robot móvil en un entorno simulando, mientras que en la figura 61 se muestra en un entorno físico con las condiciones previamente dadas en un punto inicial en el origen del plano cartesiano.

Figura 60. Entorno simulado



Fuente: Elaboración propia

Figura 61. Entorno físico



Fuente: Elaboración propia

Posición final

$x_f = 0.5m$

$y_f = 0.5m$

$alfa_f = 90°$

Donde:

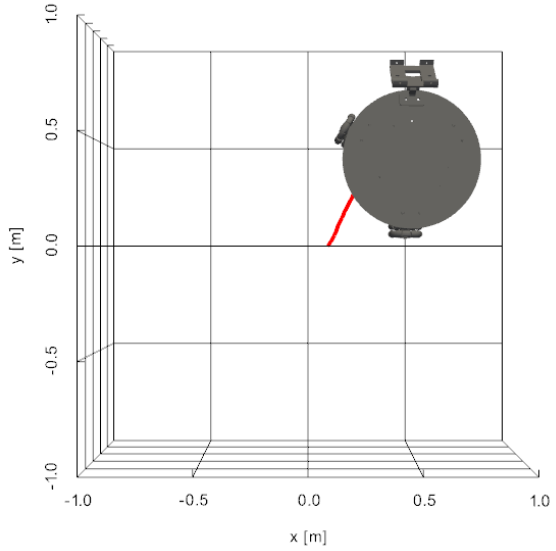
$x_f = Posición\ final\ deseada\ en\ el\ eje\ x\ en\ metros$

$y_f = Posición\ final\ deseada\ en\ el\ eje\ y\ en\ metros$

$alfa_f = Ángulo\ de\ orientacion\ final\ deseada\ en\ radianes$

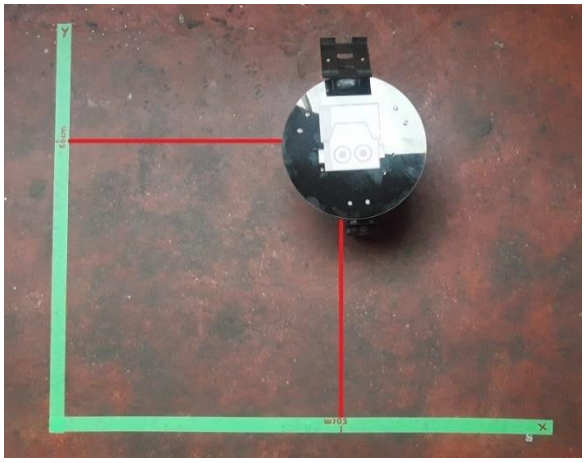
Seguidamente, en la figura 62 se muestra el desplazamiento del robot en la posición deseada tanto en el eje x como en el eje con un ángulo de rotación alfa, estos valores están establecidos en las condiciones finales previamente dadas, mientras que en la figura 63 se representa su posterior ubicación en el entorno experimental.

Figura 62. Ubicación deseada en entorno simulado



Fuente: Elaboración propia

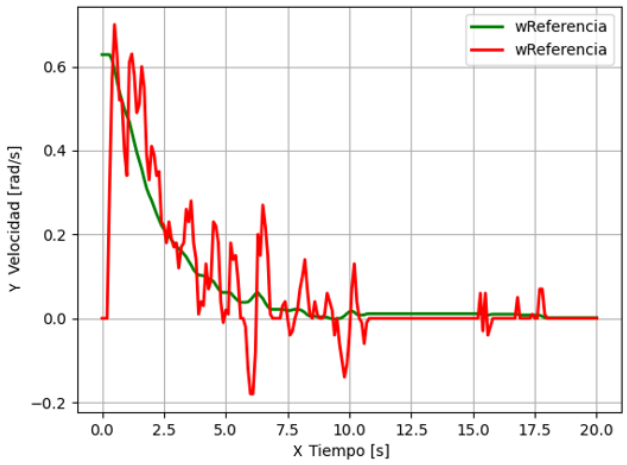
Figura 63. Ubicación deseada en entorno experimental



Fuente: Elaboración propia

Conviene subrayar que en la figura 64 se muestran las gráficas de la velocidad angular y se puede observar las oscilaciones presentes en el control al momento de seguir la trayectoria.

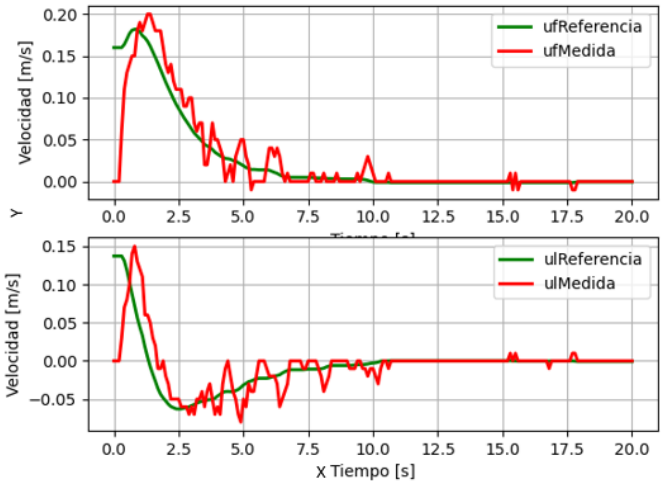
Figura 64. Velocidad angular



Fuente: Elaboración propia

Al mismo tiempo, la figura 65 se representan las gráficas de las velocidades lineales y las posteriores oscilaciones propias de las acciones de control.

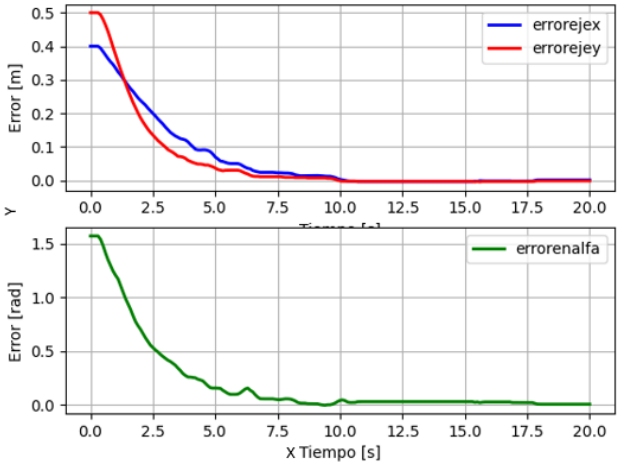
Figura 65. Velocidades lineales



Fuente: Elaboración propia

De igual modo, en la figura 66 están presentes los errores obtenidos por los parámetros de control implementados y cuyos valores convergen a cero.

Figura 66. Errores



Fuente: Elaboración propia

4.1.4. Control de seguimiento de trayectoria experimental

Para la verificación del control de seguimiento de trayectorias se parte de las siguientes condiciones.

Condiciones iniciales

$x_i = 0$

$y_i = 0$

$phi_i = 0$

Donde:

$x_i = Posición\ inicial\ en\ el\ eje\ x\ en\ metros$

$y_i = Posición\ inicial\ en\ el\ eje\ y\ en\ metros$

$alfa_i = Ángulo\ de\ orientacion\ inicial\ en\ radianes$

Seguidamente, se tiene que

Posición final

x_f *=[0,0.6,2,2]*

y_f *=[0,0.6,0.6,0]*

$alfa_f = 90°$

$x_f = Vector\ para\ generar\ cordenadas\ para\ el\ camino\ en\ posición\ x\ en\ metros$

$y_f = Vector\ para\ generar\ cordenadas\ para\ el\ camino\ en\ la\ posición\ y\ en\ metros$

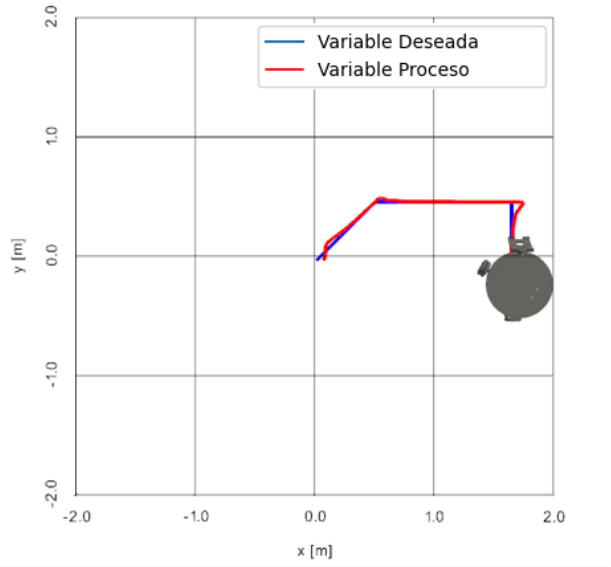
$alfa_f = Ángulo\ de\ orientacion\ final\ deseada\ en\ radianes$

Cabe resaltar que al unir los puntos del vector x_f con y_f estos permiten generar el camino a través del cual se va a desplazar el robot haciendo uso del control de seguimiento de trayectorias, así mismo, $alfa_f$ representa el ángulo que tendrá el robot el llegar a la posición

deseada al final del camino generado con ayuda de los vectores anteriormente dados en las condiciones finales.

De acuerdo con los parámetros establecidos, en la figura 67 se puede analizar el comportamiento presentado por el robot al seguir una trayectoria. En tal sentido la línea azul representa el camino a través de cual se debe desplazar, mientras que la línea en color rojo representa las acciones de control realizadas por el sistema para tratar de seguir el camino generado y donde se puede observar que a pesar de tener algunas oscilaciones su comportamiento es relativamente bueno.

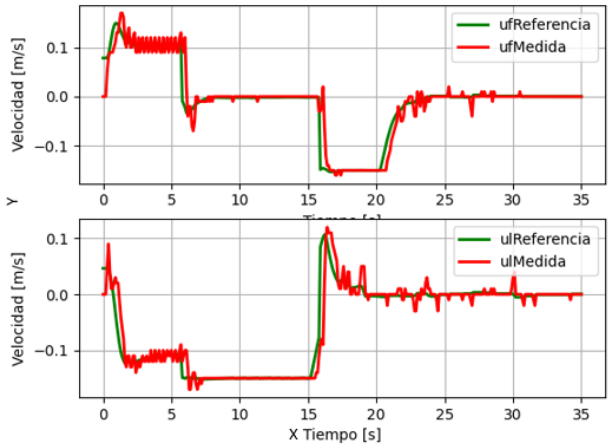
Figura 67. Seguimiento de trayectoria



Fuente: Elaboración propia

En la gráfica 68 se puede observar las respectivas oscilaciones presentadas por las velocidades medidas al momento de hacer el respectivo seguimiento a las velocidades de referencia.

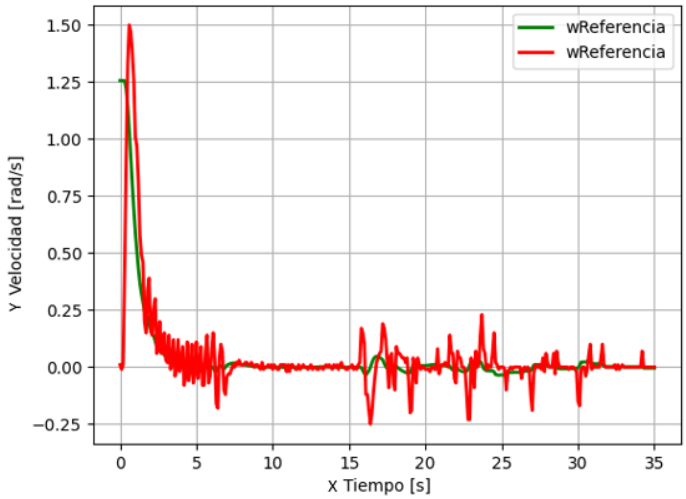
Figura 68. Velocidades de referencia vs velocidades medidas



Fuente: Elaboración propia

Al mismo tiempo, en la figura 69 se muestra el comportamiento de la velocidad angular y cuyo valor converge a cero presentando oscilaciones propias del control generado.

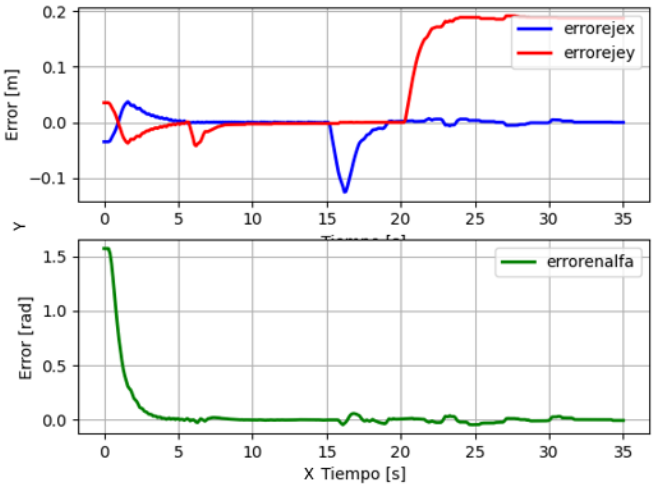
Figura 69. Velocidad angular



Fuente: Elaboración propia

Por otra parte, la figura 70 valida los errores presentes en el sistema donde se puede observar que a pesar de presentar oscilaciones propias del control estas convergen a cero en el tiempo.

Figura 70. Errores velocidades lineales y angulares

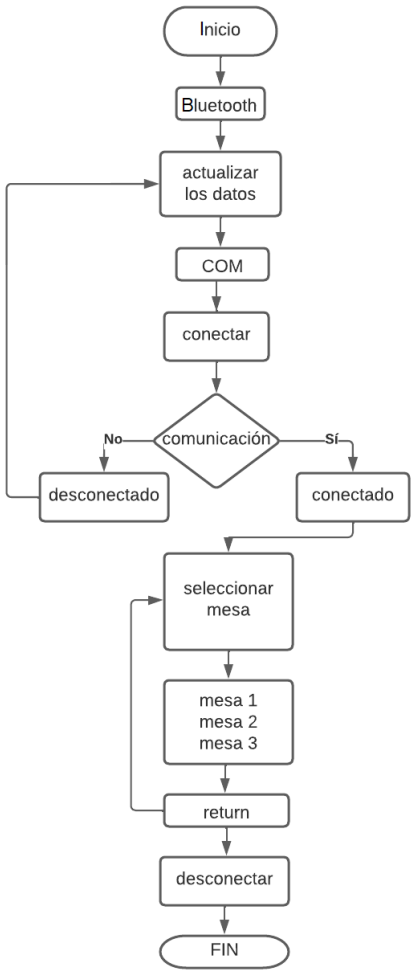


Fuente: Elaboración propia.

4.1.5. Seudocódigos para el control del robot móvil.

El la figura 71, se observa de forma visual el esquema para el control del robot por medio de la interfaz de usuario donde se muestran cada uno de los procesos llevados a cabo durante el transcurso de la comunicación desde él envió hasta la recepción de datos entre el módulo Bluetooth implementado en el prototipo y el PC.

Figura 71. Seudocódigo de control.



Fuente: Elaboración propia.

4.1.6. Interfaz de usuario y aplicación móvil de teléfono robot móvil.

En la figura 72, se muestra el respectivo diseño final de la interfaz de usuario la cual permite una interacción más practica y sencilla entre el estudiante y el robot.

Figura 72. Interfaz de usuario robot móvil.

Fuente: Elaboración propia.

Seguidamente, en la figura 73 se observa la aplicación móvil donde el cliente puede escoger dentro del menú las opciones disponibles partiendo primeramente de seleccionar la mesa en la que se encuentre, inmediatamente selecciona la mesa, se le despliega una ventana como la mostrada en la figura 74 donde podrá elegir entre las tres opciones disponibles la de su preferencia.

Figura 73. Menú principal.

Fuente: Elaboración propia.

Figura 74. Menú de opciones a elegir.

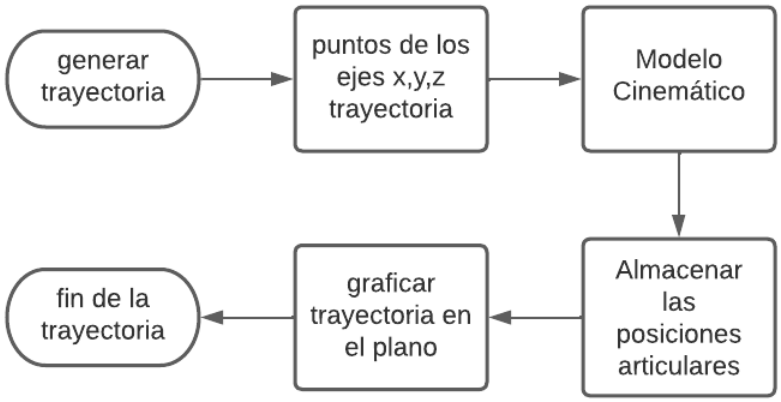
Fuente: Elaboración propia.

4.2.Validación de resultado robot antropomórfico

4.2.1. Diagrama de flujo de control rutinas robot antropomórfico.

Para la ejecución de la simulación de los algoritmos de programación realizados en Python se emplean los modelos cinemáticos calculados, estos se muestran en el diagrama de flujo de la figura 75 en donde se representa de manera visual el comportamiento interno para la ejecución de todos los procesos llevados a cabo en la realización de las rutinas los cuales van desde la generación de las trayectorias, el cálculo de los modelos cinemáticos, el control y el almacenamiento de las posiciones articulares necesarias para el correcto funcionamiento del robot.

Figura 75. Control de rutinas.

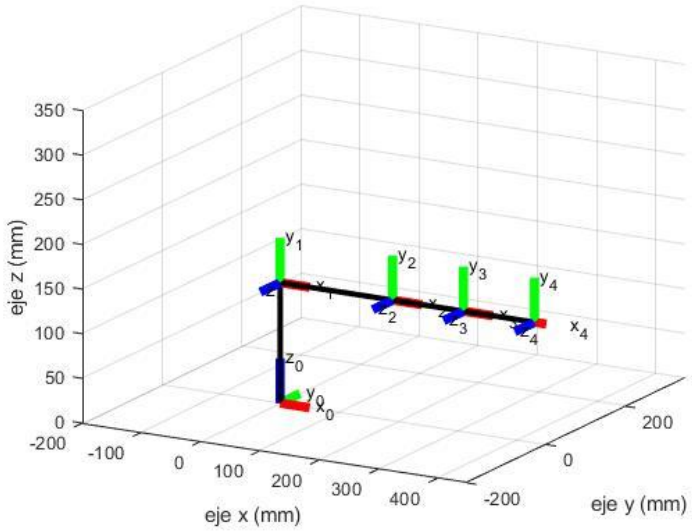


Fuente: Elaboración propia

4.2.2. Validación de la cinemática directa

En la figura 76 se analizan los ejes coordenados de cada articulación observando que sean iguales a los obtenidos en los cálculos de Denavit Hartenberg de la cinemática directa dispuesta en la figura 51 y la cual representa su posición inicial.

Figura 76. ejes coordenados



Fuente: Elaboración propia

En la figura 77 se comprueban las longitudes de cada una de las articulaciones presentes en el robot, estas medidas se presentan a continuación.

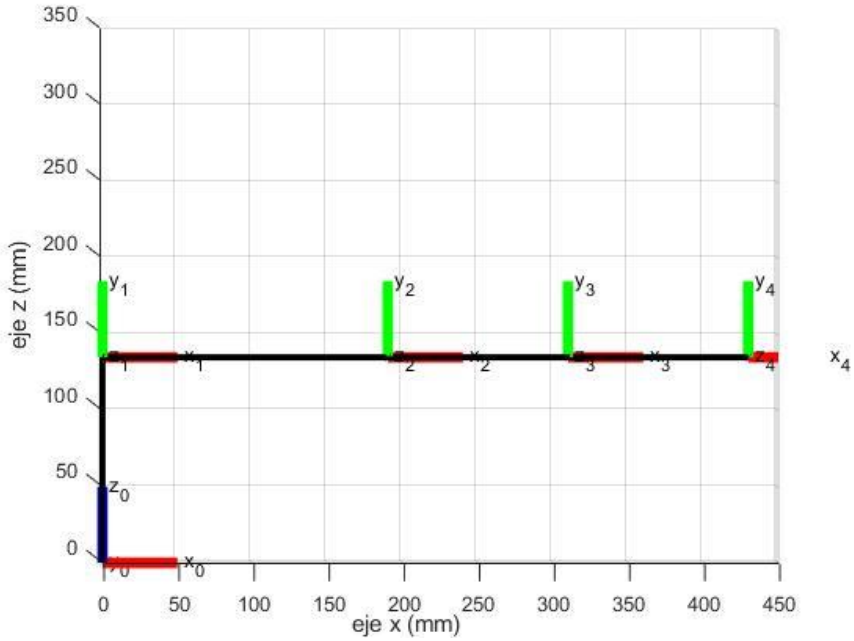
L1=135mm

L2=190mm

L3=120mm

L4=120mm

Figura 77. ejes coordenados



Fuente: Elaboración propia

Para verificar la posición en las articulaciones del robot se parte de los siguientes datos:

Condiciones iniciales

$Q_1 = 0°$

$Q_2 = 0°$

$Q_3 = 0°$

$Q_4 = 0°$

Condiciones finales

$Q_1 = 90°$

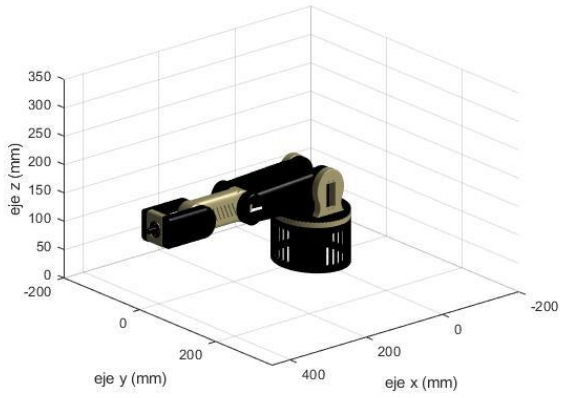
$Q_2 = 90°$

$Q_3 = 90°$

$Q_4 = 90°$

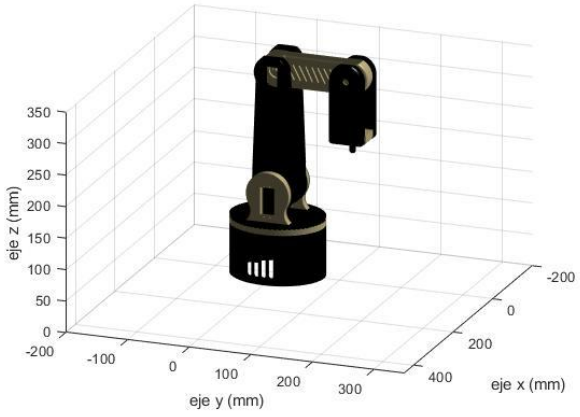
La figura 78 muestra el robot antropomórfico con las condiciones iniciales dadas y extendido completamente. Seguidamente, en la figura 79 se puede observar que el robot logra alcanzar las posiciones finales de manera correcta en el entorno de simulación mientras que en la figura 80 se verifica que efectivamente este alcanza las posiciones establecidas en pruebas experimentales

Figura 78. posición inicial



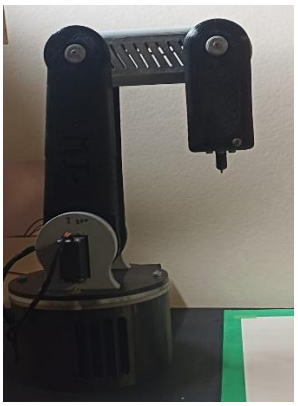
Fuente: Elaboración propia

Figura 79. posición final



Fuente: Elaboración propia

Figura 80. posición final en pruebas experimentales



Fuente: Elaboración propia

4.2.3. Validación de la cinemática inversa

Partiendo de las ecuaciones 86 hasta la 109 y remplazando los valores dados a continuación se procede a calcular los valores de cada articulación.

Puntos finales del efector final

$pm_x = 100$

$pm_y = 150$

$pm_z = 10$

Longitudes de las articulaciones

$L_1 = 135\text{mm}$

$L_2 = 190\text{mm}$

$L_3 = 120\text{mm}$

$L_4 = 120\text{mm}$

Al remplazar los valores en la ecuación 86 se tiene la expresión (123).

$$q1 = atan2(150{,}100) \tag{123}$$

Donde en la respuesta de $\boldsymbol{q1}$ queda mostrada en la expresión (124).

$$\boldsymbol{q1} = 0.98279 \tag{124}$$

Al tomar las ecuaciones 102 a 105 y remplazar sus valores se obtiene la expresión (125)

$$\cos(q3) = \frac{(32500+(130-135)^2-190^2-120^2)}{2(190*120)} \tag{125}$$

al operar los valores de la ecuación (125) se obtiene la respuesta mostrada en la ecuación (126).

$$\cos(q3) = -0.39418 \tag{126}$$

Seguidamente al remplazar los valores en la expresión (127) se tiene lo siguiente

$$\text{sen}(q3) = -1 - \sqrt{1-(-0.39418)} \tag{127}$$

luego de operar la expresión (127) se tiene como resultado lo mostrado en (128).

$$\text{sen}(q3) = -0.919067 \quad (128)$$

remplazando los valores de los resultados obtenidos en (126) y (128) se obtiene la expresión (129).

$$q3 = atan2(-0.919067, -0.39418) \quad (129)$$

$$\boldsymbol{q3} = -1.97596$$

Para el cálculo de la articulación $q2$ se toma la ecuación 106 y la cual al remplazar sus valores queda representada como se muestra en la expresión (130).

$$\beta = atan2(120 * sen(-1.97596), 120 * \cos(-1.97596) + 190) \quad (130)$$

$$\beta = -0.6580$$

$$\alpha = atan2\left(130 - 135, \sqrt{100^2 + 150^2}\right)$$

$$\alpha = -0.0277$$

Para hallar el valor de $q2$ se remplaza el valor de β y α en la ecuación (108) quedando representada como se muestra en la ecuación (131).

$$q2 = (-0.0277\ ) - (-0.6580) \quad (131)$$

$$\boldsymbol{q2} = 0.6303$$

Finalmente, para obtener $q4$ se remplazan los valores en la ecuación (109) los cuales quedan expresados como se muestra en la ecuación (132).

$$q4 = -{}^{\pi}/_{2} - 0.6303 - (-1.97596) \quad (132)$$

$$\boldsymbol{q4} = -0.2251$$

A continuación, se muestran los valores obtenidos para cada articulación en Python.

$q1 = 0.982793723247329$

$q2 = 0.6302370476766703$

$q3 = -1.975981118880563$

$q4 = -0.22505225559100395$

Por último, se calculan los errores presentes en cada articulación con los valores teóricos y los valores prácticos partiendo de la expresión (133), estos quedan establecidos de la siguiente manera.

Error de la articulación $q1$

$$error = \frac{Vteo - Vpract}{Vteo} * 100\% \quad (133)$$

$$error\ q1 = \frac{0.9827 - 0.98279}{0.9827} * 100\%$$

$$error\ q1 = 0.009158\%$$

Error de la articulación $q2$

$$error\ q2 = \frac{0.6303 - 0.63023}{0.6303} * 100\%$$

$$error\ q2 = 0.0111\%$$

Error de la articulación $q3$

$$error\ q3 = \frac{(-1.9759) - (-1.97598)}{-1.9759} * 100\%$$

$$error\ q3 = 0.004048\%$$

Error de la articulación $q4$

$$error\ q4 = \frac{(-0.2251) - (-0.22505)}{-0.2251} * 100\%$$

$$error\ q4 = 0.02221\%$$

Al verificar los datos arrojados en la simulación y los obtenidos por medio de cálculos se puede inferir que la variación entre los errores de cada articulación es mínima.

Otra forma de validar la cinemática inversa es agregando los valores de las articulaciones a la cinemática directa y verificar la matriz homogénea del efector final para comprobar que la posición de los puntos corresponda a los establecidos en los puntos finales del efector final con los obtenidos en la figura 81.

Figura 81. Posición efectora final

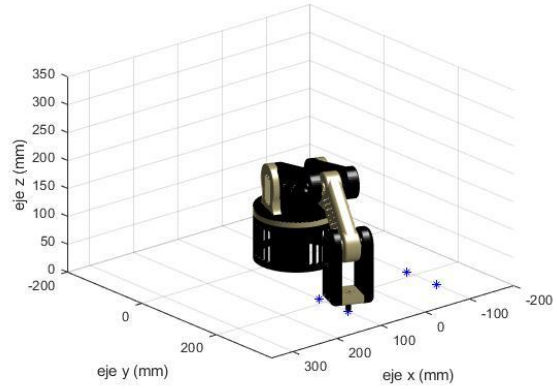
```
Matriz [[6.93889390390723e-17 0.554700196225229 0.832050294337844
  100.000000000000]
 [2.77555756156289e-17 0.832050294337844 -0.554700196225229
  150.000000000000]
 [-1.00000000000000 2.77555756156289e-17 6.12323399573677e-17
  10.0000000000000]
 [0 0 0 1]]
```

Fuente: Elaboración propia

4.2.4. Validación de modelo simulado y físico

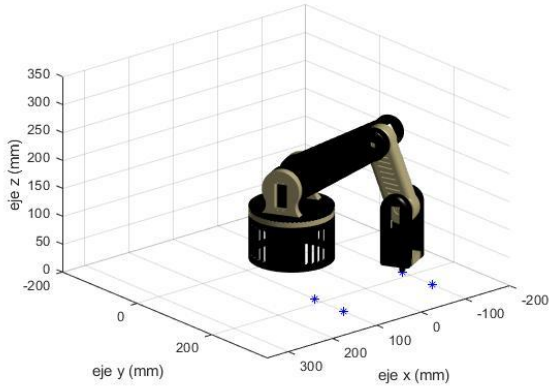
Para esta verificación se compara la simulación y el comportamiento físico de brazo dando cuatro puntos, en la figura 82 y 83 se puede ver el comportamiento generando en la simulación y chequear que el robot llega a cada punto.

Figura 82. posición inicial



Fuente: Elaboración propia

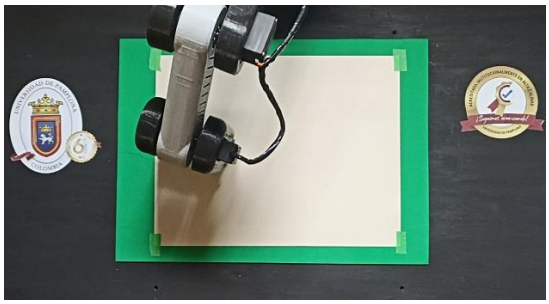
Figura 83. posición final



Fuente: Elaboración propia

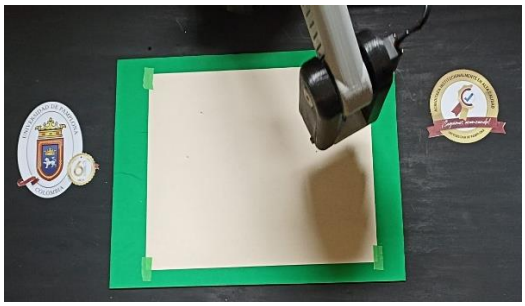
En la figura 84 y 85 se presenta el correspondiente funcionamiento del robot en el entorno de validación física, se puede examinar que efectivamente está llegando a los puntos propuestos en la figura 82 y 84, también se puede observar que existe un desfase y el robot no alcanza completamente los puntos debido a la precisión de los servomotores empleados en el prototipo.

Figura 84. posición inicial físico



Fuente: Elaboración propia

Figura 85. posición final en físico



Fuente: Elaboración propia

La figura 86 se presenta el resultado obtenido de los puntos hechos por el robot en el entorno experimental.

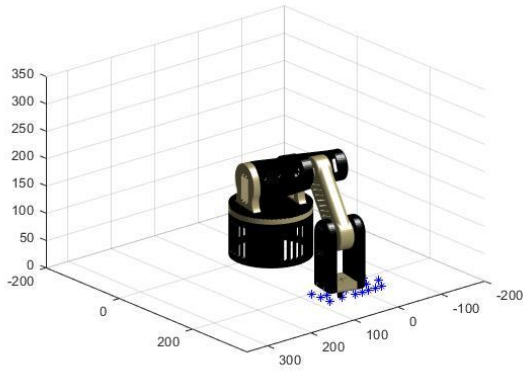
Figura 86. validación de puntos

Fuente: Elaboración propia

4.2.5. Validación de la palabra regalo en lenguaje braille simulado y físico

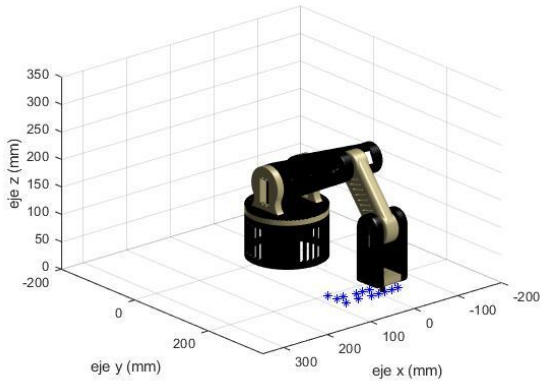
De igual forma, se procede a validar el modelo por medio de una palabra escrita en lenguaje braille de forma simulada y física donde la figura 87 y 88 representa la parte simulada y la figura 89 y 90 representa la parte experimental.

Figura 87. Palabra inicial



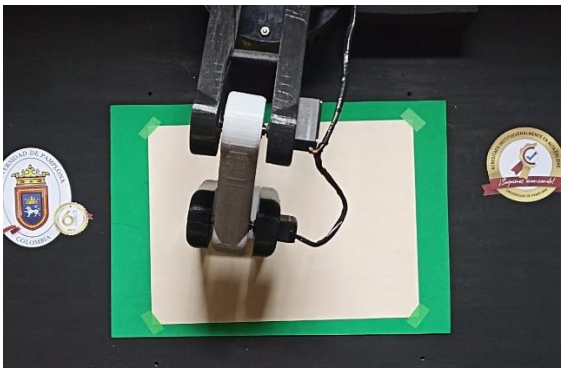
Fuente: Elaboración

Figura 88. Palabra final



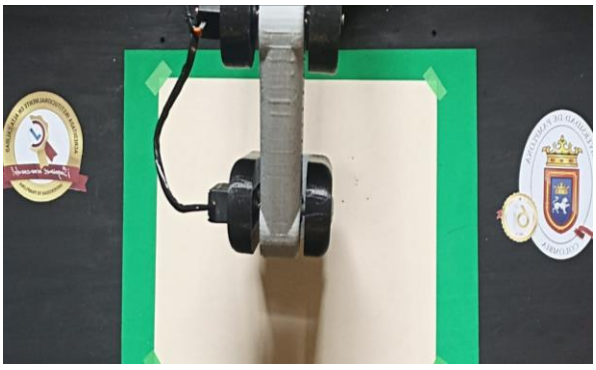
Fuente: Elaboración propia

Figura 89. Palabra física inicial



Fuente: Elaboración propia

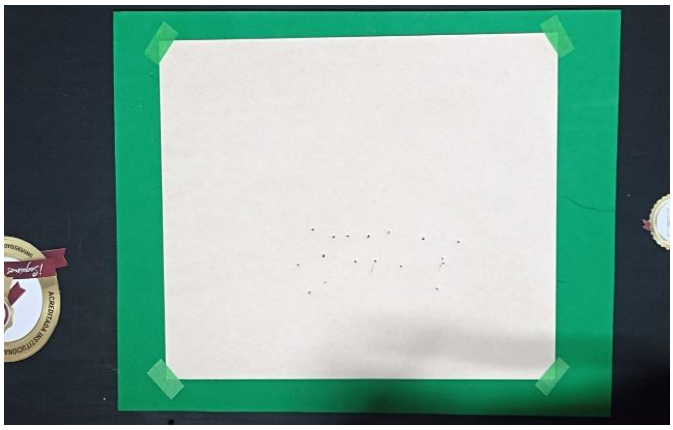
Figura 90. Palabra física terminando



Fuente: Elaboración propia

En la figura 91 se comprueba la palabra escrita obtenida por el robot en la parte física, al igual que en la figura 86 se observan desfaces en algunos puntos debido a la inestabilidad presentada en los servomotores ya que estos generan desfaces en las posiciones finales propuestas, lo cual se puede mejorar con la utilización de motores más precisos.

Figura 91. palabra finalizada

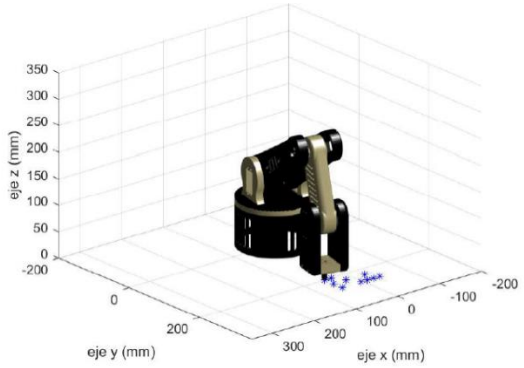


Fuente: Elaboración propia

4.2.6. Validación de la palabra vida en lenguaje braille simulado y físico

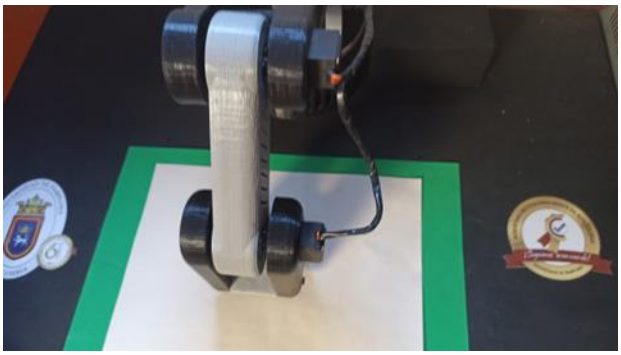
De manera similar, se procede a validar el modelo por medio de la palabra vida de forma simulada y física, en la figura 92 se observa su respectiva simulación y en la figura 93 se examina su representación experimental.

Figura 92. Palabra inicial



Fuente: Elaboración propia

Figura 93. Palabra inicial



Fuente: Elaboración propia

En los resultados obtenidos en la figura 94 se pude observar que debido a la precisión de los servomotores empleados la palabra se desfasa uno grados entre puntos, esto se puede mejorar con la utilización de actuadores de más precisión.

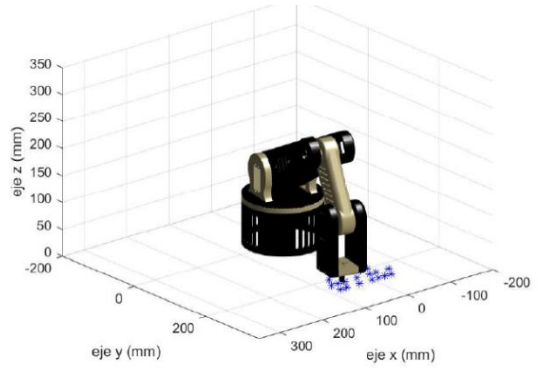
Figura 94. Resultado

Fuente: Elaboración propia

4.2.7. Validación de la palabra braille simulado y físico

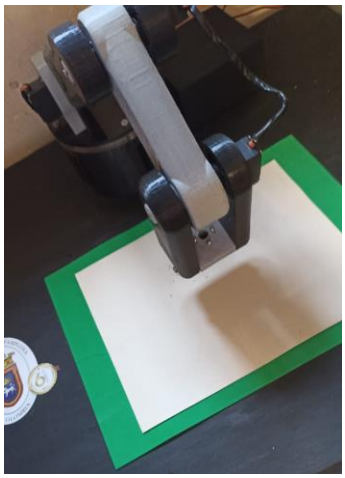
Finalmente, al validar el modelo con la palabra braille de forma simulada y física, en la figura 95 se puede observar su respectiva simulación y en la figura 96 se examina su representación experimental.

Figura 95. Modelo inicial



Fuente: Elaboración propia

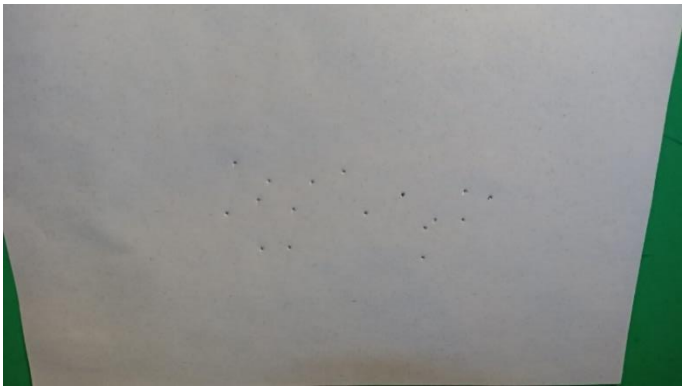
Figura 96. Modelo físico



Fuente: Elaboración propia

En la figura 97 se pude observar el resultado obtenido.

Figura 97. Resultado obtenido

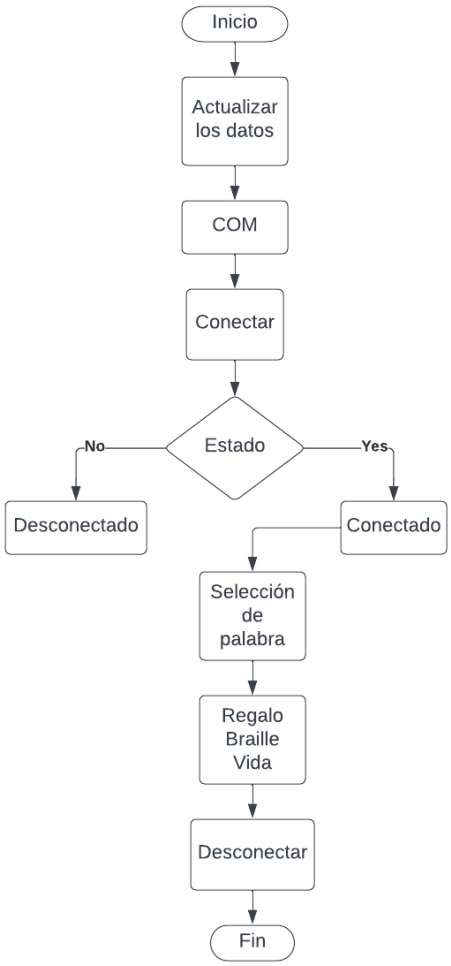


Fuente: Elaboración propia

4.2.8. Seudocódigos para el control del robot antropomórfico.

El la figura 98, se observa el esquema para el control del robot desde la interfaz de usuario donde se muestran cada uno de los procesos realizados durante el transcurso de la comunicación incluidos él envió de datos hasta la posterior recepción por parte del controlador desde el PC.

Figura 98. Seudocódigo de control para robot antropomórfico.



Fuente: Elaboración propia

4.2.9. Interfaz de usuario robot antropomórfico.

En la figura 99, se muestra el respectivo diseño final de la interfaz de usuario la cual permite una interacción más practica y sencilla entre el estudiante y el robot.

Figura 99. Interfaz de usuario robot antropomórfico.

Fuente: Elaboración propia

CONCLUSIONES

La presente investigación relacionada a la didáctica de la robótica e implementación de softwares libres permitió llegar a las siguientes conclusiones:

El diseño del robot antropomórfico y robot móvil omnidireccional permitió demostrar que, a través de la implementación de softwares libres como alternativa, se pueden obtener buenos resultados e incluso similares a los obtenidos en versiones pagas. De tal forma, que los softwares gratuitos como *Python, fusión 360, Fritzing, Visual studio code, QtDesigner*, tienen las mismas características de funcionamiento al momento de implementar algoritmos y diseños asistidos por computación.

Igualmente, con la construcción del robot móvil omnidireccional y robot antropomórfico se obtuvo la representación física, de manera que permitiera validar el funcionamiento de softwares libres como herramienta didáctica al momento de introducir a estudiantes en el campo de la robótica de manera conceptual y práctica. Así mismo, se impulsa en los estudiantes el interés de utilizar este tipo de herramientas como medios alternativos para el aprendizaje en este campo o en cualquier otro.

De manera similar, a través del cálculo de las cinemáticas se generaron los diferentes modelos matemáticos, de modo, que los resultados obtenidos son necesarios en la implementación de los algoritmos que permiten el control en los robots didácticos. De manera similar, se desarrollaron algoritmos para la creación de las interfaces de usuarios necesarias en ambos prototipos para facilitar la interacción entre persona y máquina de manera más practica y sencilla, así mismo, se creó una aplicación móvil la cual incorpora un menú de opciones y que es necesaria en la aplicación práctica a la cual se le asigno al robot móvil omnidireccional.

Del mismo modo, se validó el funcionamiento de los algoritmos implementados a través de pruebas experimentales y simuladas. Esto es muy útil ya que permite analizar el comportamiento en las posiciones generadas en cada uno de los prototipos permitiendo verificar que el comportamiento es el deseado.

Finalmente, se desarrollaron guías didácticas que proporcionan a estudiantes las ayudas y herramientas necesarias para el proceso de diseño y construcción de los robots didácticos, al mismo tiempo, en estas se especifican cada uno de los procesos que se deben seguir para lograr alcanzar con éxito la elaboración de ambos prototipos de robots.

Se evidencia que existen muy pocos estudios e información relacionada con este proyecto por lo que uno de los fines de esta tesis es también mostrar cuan necesario es incrementar la investigación hacia esta área, ya que podemos generar soluciones a distintas problemáticas que se presentan en la sociedad y aun así no tener altos gastos económicos en su ejecución ya que para esto se hace relevancia al uso de softwares libres.

Trabajos futuros

Teniendo en cuenta que al tratarse de modelos didácticos y que las necesidades de la sociedad van cambiando se hace necesario realizar ajustes o mejoras a este proyecto con el fin de cumplir con las expectativas que se plantean y se espera que en un futuro estos puedan mejorar algunas funcionalidades y características en las aplicaciones para las cuales fueron construidos, así mismo los cambios se realizan no solamente en la parte del diseño físico sino también en la programación ya que Python tiene un sinfín de aplicaciones y librerías que permitirían agregar otras funciones que pueden llevar a cabo tareas adicionales para lo cual se recomienda un cambio de motores en el caso del robot antropomórfico ya que los servos tienen desfaces y no permiten

una exactitud en los puntos de llegada del robot, también se podría agregar el efector final para un mejor acople.

Con respecto a la interfaz gráfica que es la parte visual a la que tienen acceso los usuarios en primera instancia se puede dar un diseño más dinámico y opcional en la que los usuarios tengan una mejor experiencia a la hora de interactuar con los robots, permitiendo que la finalidad sea cumplida y se cumplan todas las expectativas.

Así mismo, se pueden agregar sensores de voz para que el usuario interactúe con el robot antropomórfico, de tal forma que pueda pedir que le escriba cualquier palabra, ya que este prototipo solo trabaja con palabras preestablecidas, esto con el fin de que el usuario tenga más opciones de elección permitiendo así una mejor calidad en el servicio que ofrece dicho robot.

También se pueden hacer ajustes con sensores de proximidad para aquellos casos en que la persona es discapacitada y así evitar accidentes de cualquier índole.

Cabe mencionar que es pertinente una revisión del robot de forma periódica para comprobar el estado de las articulaciones y los eslabones de tal manera que dependiendo de las condiciones en que este se encuentre se puedan generar mantenimientos.

En cuanto a la estructura de robot móvil se puede realizar un diseño más robusto que cumpla con las expectativas de un robot mesero ya que al tratarse de un prototipo este presenta limitantes en cuanto a la altura.

Por otro lado, se pueden hacer mejoras en la parte del control implementado ya que al ser un PID este no cumple con los requerimientos necesarios para ser ejecutado en un restaurante y por ende presenta algunos inconvenientes en las trayectorias previamente incorporadas.

Por último, entorno a la aplicación dispuesta como menú de opciones en el robot móvil se pueden hacer mejoras que permitan una interacción más agradable entre el usuario y la máquina.

GUÍAS

Apéndice A

Instructivo para el diseño y construcción de un robot móvil omnidireccional y un robot antropomórfico.

GUÍA PARA EL DISEÑO Y CONSTRUCCIÓN DE UN ROBOT MÓVIL OMNIDIRECCIONAL Y UN ROBOT ANTROPOMÓRFICO

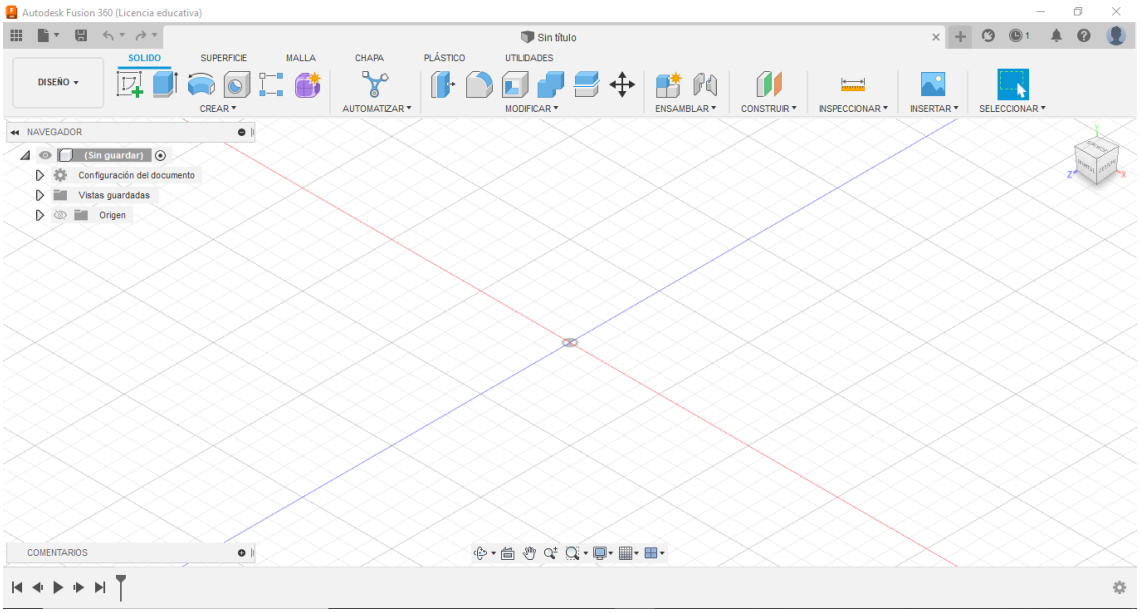
La presente guía tiene como objetivo guiar al estudiante en la construcción de un robot móvil omnidireccional y un robot antropomórfico a través de la implementación de softwares gratuitos. Para ello, se dan a conocer todas las herramientas utilizadas las cuales son necesarias en la realización de ambos proyectos enfocándolos en un entorno didáctico que permita una fabricación de forma sencilla y práctica. Es de mencionar que la guía didáctica contará con todas las ayudas requeridas para cumplir en buen término la realización de los prototipos de ambos robots.

DISEÑO 3D DE LAS PIEZAS DEL ROBOT MÓVIL OMNIDIRECCIONAL Y EL ROBOT ANTROPOMÓRFICO

Para el diseño 3D de las piezas de ambos prototipos se implementó fusión 360 en su versión 2.0.14337, este es un software CAD, CAM, CAE basado en la nube para el diseño y manufactura de productos. Dado que el objetivo es el uso de softwares gratuitos se implementó la versión educativa ofrecida por AUTODESK y la cual otorga una licencia gratuita de un año.

Seguidamente, se detallan algunas herramientas dispuestas dentro del entorno de es este software necesarias para la creación de piezas y los cuales se observan en la figura 100.

Figura 100. entorno fusión 360



Fuente: (*Fusion 360 | Software CAD, CAM, CAE y de Circuitos Impresos 3D Basado En La Nube | Autodesk*, n.d.)

DESCRIPCIÓN DE ALGUNOS ELEMENTOS DENTRO DEL ENTORNO GRÁFICO DE FUSION 360

Barra de herramientas:

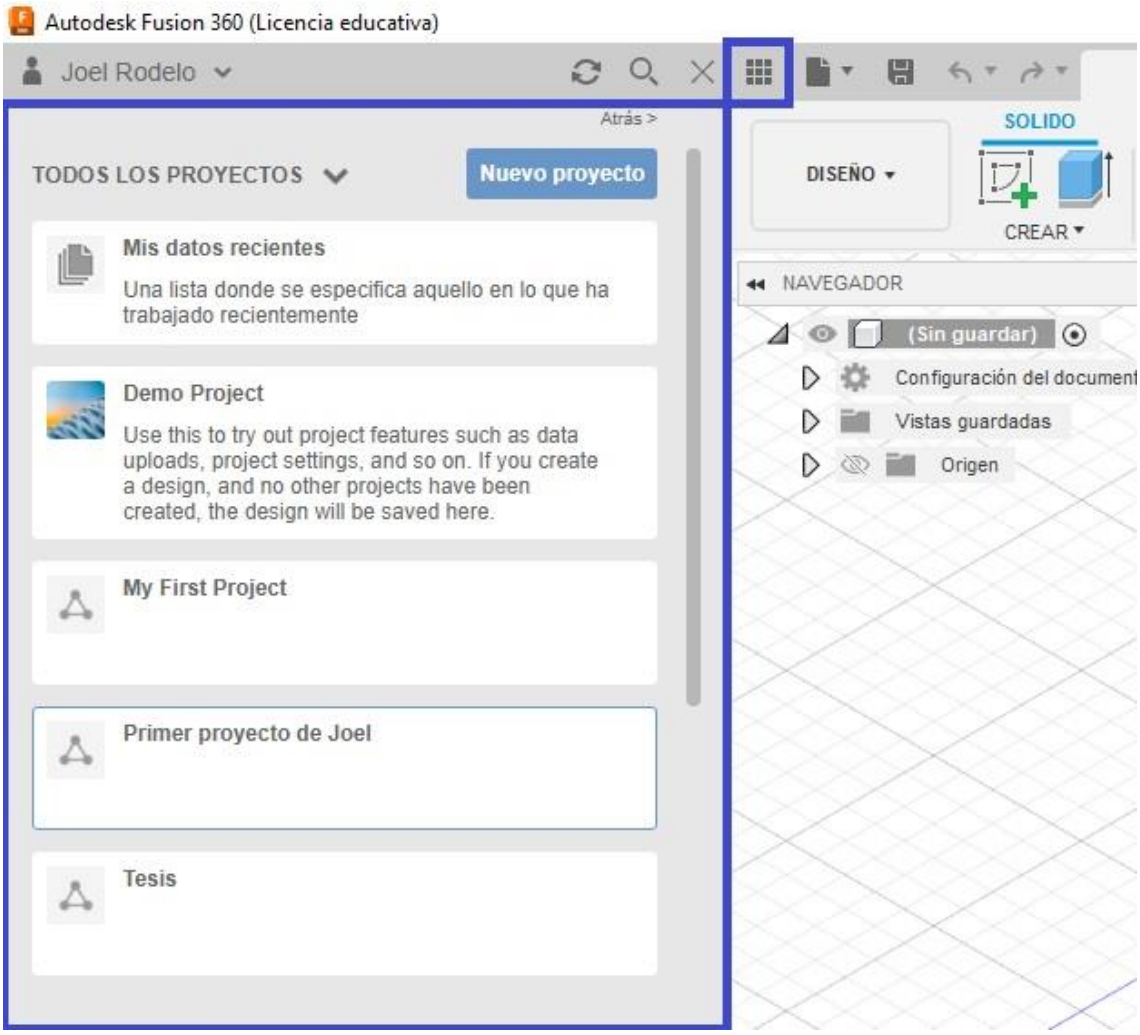
Como se observa en la figura 101, en esta barra se encuentran todos los elementos y características que están disponibles en el entono de trabajo. Con la ayuda de estas herramientas se pueden crear o modificar las piezas 3D.

Figura 101. herramientas

Barra de aplicaciones:

Esta herramienta permite a los usuarios acceder a la información de proyectos ya existentes, crear nuevos o guardarlos de forma manual, también permite la creación de espacios de trabajo los cuales se pueden organizar por carpetas, por otro lado, cuenta con muestras y tutoriales de ejemplo para una rápida compresión del entorno del software como se muestra en la figura 102.

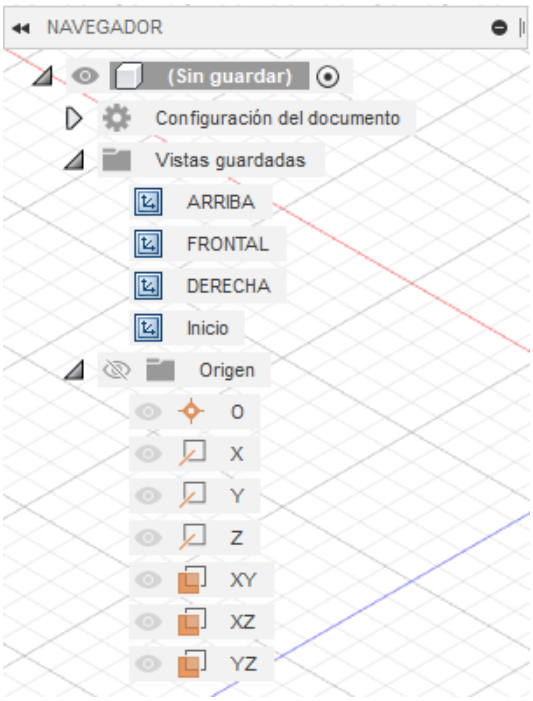
Figura 102. aplicaciones



Navegador:

Enumera los objetos del diseño tales como planos, bocetos, piezas, ensambles, geometrías de construcción etc., y permite controlar la visibilidad de los objetos tal como en la figura 103.

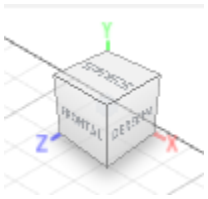
Figura 103. Navegador



Cubo de Vistas:

Permite orbitar a través del diseño o visualizar diferentes perspectivas de este como se puede observar en la figura 104.

Figura 104. Vista



Navegación:

En esta barra se encuentran opciones para girar, cambiar el modo de visualización del entorno de trabajo, quitar rejillas, así como la opción para ajustar el zoom entre otras y están representadas en la figura 105.

Figura 105. Navegación

Cronología:

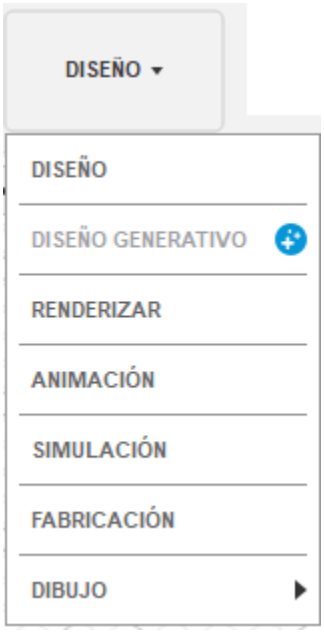
En esta herramienta se muestra en orden cronológico el historial de forma ordenada de todas las operaciones realizadas al momento de diseñar una pieza tal como se visualiza en la figura 106.

Figura 106. Cronología.

Espacio de Trabajo

Contiene los diferentes espacios de trabajo que muestran varias herramientas las cuales pueden adaptarse a las necesidades en cada etapa del proceso de diseño incluido diseño generativo, renderizado, animación, simulación, fabricación y dibujo. Estos son visibles en la figura 107.

Figura 107. Espacio trabajo.



GENERACIÓN DE PROYECTOS 3D

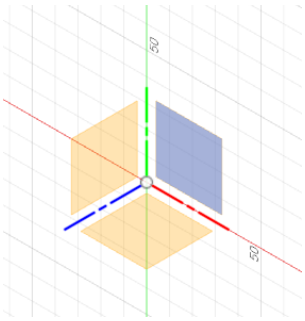
Para la creación de un nuevo proyecto en 3D primero se debe generar un boceto en 2D mediante la barra de herramientas.

1) Dentro de la barra de herramienta se selecciona la opción crear boceto mostrada en la figura 108 que permite crear perfiles geométricos que definen la base del diseño.

Figura 108. Boceto.

2) Luego de seleccionada la opción de crear boceto en la figura 109 se selecciona el plano de trabajo sobre el cual se va a crear el diseño, estos pueden ser.
   - Planos de origen XY, YZ o ZX
   - Caras
   - Planos de construcción

Figura 109. Planos.



3) Seguidamente se crea el boceto 2D con ayuda de la barra de herramientas mostrada en la figura 110 para crear las diferentes formas requeridas para el diseño.

Figura 110. Barra de herramientas para planos.

HERRAMIENTAS PARA CREAR UN DISEÑO

Estas herramientas son de gran ayuda porque permiten la creación de una pieza 3D en el entorno de fusión 360 y se pueden observar en la figura 111.

Figura 111. Herramientas.

A continuación, se describen algunas de estas herramientas que son imprescindibles a la hora de realizar un diseño 3D.

Línea: Crea una serie de líneas.

Rectángulo:

Rectángulo por dos puntos: Crea un rectángulo a partir de dos puntos seleccionados para definir las esquinas opuestas del rectángulo.

Rectángulo por tres puntos: Crea un rectángulo a partir de la definición de tres puntos específicos. Los puntos seleccionados definen la posición y el tamaño de las cuatro líneas.

Rectángulo por centro: Crea un rectángulo seleccionando el punto central y luego una esquina. El primer punto define la posición del rectángulo. El segundo punto define la longitud y el ancho.

Círculos:

Circulo por diámetro central: Crea un círculo especificando el centro del mismo y el diámetro. Se recomienda el uso de este método cuando se conoce el tamaño y la ubicación del círculo.

Circulo por dos puntos: Crea un círculo a partir de la definición de dos puntos específicos. Los puntos definen la posición y el tamaño del círculo.

Circulo por tres puntos: Crea un círculo a partir de la definición de tres puntos específicos. Los puntos definen la posición y el tamaño del círculo. Los tres puntos seleccionados se encuentran en la circunferencia del círculo.

Puntos de ajustes Spline: Crea una elipse a partir de la definición de varios puntos específicos. Los puntos definen la posición y el tamaño de la elipse

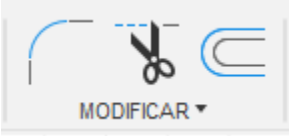
Simetría Genera un reflejo de la geometría del boceto en una línea de simetría.

Cota de boceto: se utilizan las cotas para utilizar el tamaño o la posición del objeto del boceto

HERRAMIENTAS PARA MODIFICAR UN DISEÑO

En la figura 112 su ubica esta herramienta que permite modificar ciertos paramentos sobre el diseño creado.

Figura 112. Modificadores.



Dentro de esta opción se encuentran ayudas como:

Empalme: Crea un arco de radio específico en una esquina o una intersección de dos líneas.

Recortar: Recorta las líneas seleccionadas del boceto a la siguiente intersección de una geometría

Desface: Copia las curvas de boceto seleccionadas aúna distancias específicas de las curvas originales

HERRAMIENTAS DE RESTRICCIONES

Las restricciones permiten la relación de entidades de un boceto, por defecto el programa deduce automáticamente las restricciones, pero también se pueden acceder a algunas de ellas y están representadas en la figura 113.

Figura 113. Restricciones.

Entre estas herramientas se encuentran:

Horizontal y vertical: Restringe una línea o dos puntos para que se sitúen en el eje vertical o horizontal.

Coincidente: Limita la posición de dos puntos o un punto y una línea o curva juntas.

Tangente: Restringe una curve y otro objeto para que se toquen en un solo punto, pero nunca se crucen.

Igual: Restringe objetos similares de forma que sus tamaños sean idénticos cuando se cambie el tamaño del objeto los demás también cambiaran.

Paralelo: Restringe dos líneas para que se extiendan en la misma dirección y nunca se intercepten.

Perpendicular: Restringe dos objetos para que queden perpendiculares (en un ángulo de 90 grados) entre sí.

Fijar/anular fijación: Bloquea el tamaño y la ubicación de un punto u objeto.

Punto medio: Restringe un punto u objeto al punto medio de otro objeto.

Concéntrica: Restringe dos o más arcos, círculos o elipses al mismo punto central.

4) Al finalizar el diseño del boceto dando clic en la herramienta terminar boceto como se muestra en la figura 114. En la figura 115 se puede observar el croquis terminado

Figura 114. Terminar boceto.

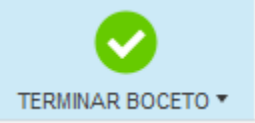
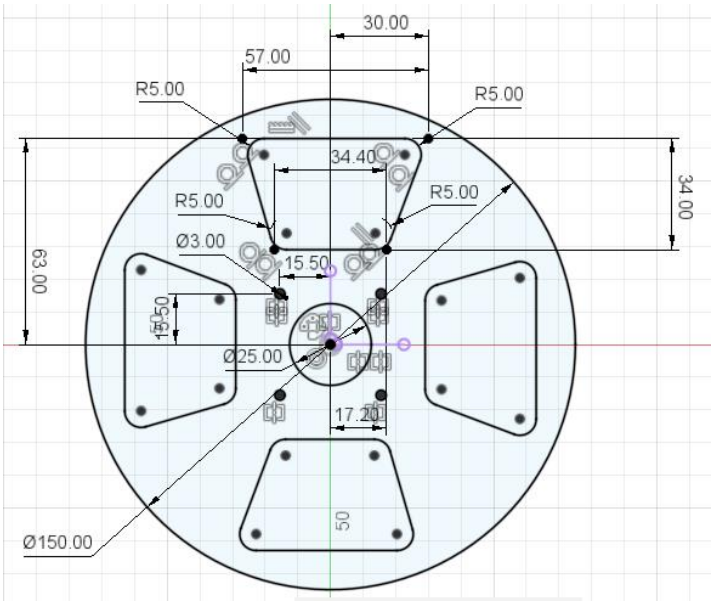


Figura 115. boceto terminado en 2d.



5) Para crear un diseño 3d a partir del boceto 2D se hace clic en la pestaña que dice sólido, esta permitirá utilizar herramientas como extruir, empalmes, vaciados, entre otros. Estas herramientas se pueden ver en la figura 116.

Figura 116. Herramienta de diseño 3d.

Herramientas de modelo 3D

Esta herramienta es de gran importancia porque con ella se pueden realizar las respectivas operaciones que permiten el proceso de ir construyendo una pieza en 3D y cuya distribución se presentan en la figura 117.

Figura 117. Boceto terminado en 2d.

A continuación, se describen algunas de ellas

Extruir: Agrega profundidad a un perfil de boceto cerrado o una cara plana.

Revolución: Gira un perfil de boceto o una cara plana alrededor de un eje seleccionad

Agujero: Crea un agujero a partir de valores y selecciones especificados por el usuario

Barrido: Barre un perfil de boceto o una cara plana a lo largo de un patrón seleccionado.

Nervio: Crea un perfil plano (paralelo) al plano de boceto.

Red: Crea un perfil normal al plano de boceto

Solevación: Crea una forma de transición entre dos o más perfiles de boceto o caras planas.

En la figura 118 se muestran otras herramientas que son de gran importancia al momento de diseñar piezas 3D.

Figura 118. Modificar.

De manera similar, se describen algunas de estas opciones a continuación.

Pulsar o tirar: Modifica la geometría seleccionada mediante comandos de desfase, extrusión o empalme.

Empalme: Redondea las aristas de un cuerpo solido mediante la adición de material alas aristas interiores y la eliminación de material alas aristas exteriores.

Vaciado: Elimina material del interior de una pieza, creando una cavidad hueca con paredes de un grosor determinado

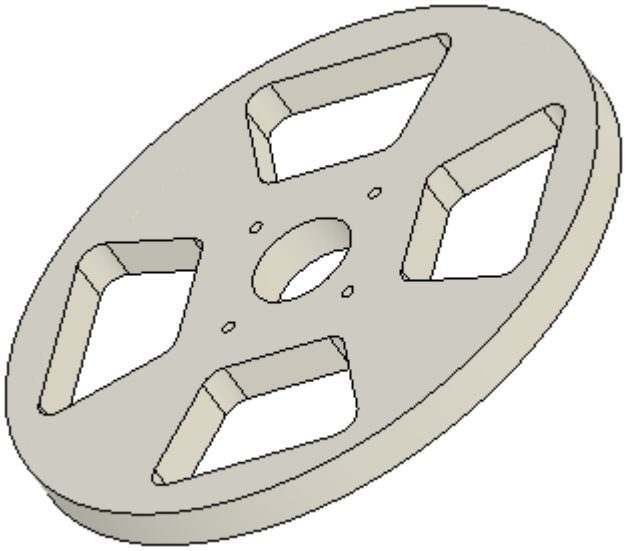
Combinar: Une, corta o intercepta dos o más cuerpos solidos para crear un único cuerpo sólido.

Dividir cuerpo: Divide los cuerpos seleccionados mediante un perfil, una cara o plano

Mover: Desplaza el cuerpo o el boceto a una distancia o ángulo determinado

6) Se seleccionan los objetos a los cuales se les va a generar la operación 3D, definiendo los valores deseados a través del cuadro de dialogo correspondiente como se ve en la figura 119.

Figura 119. Pieza 3D.

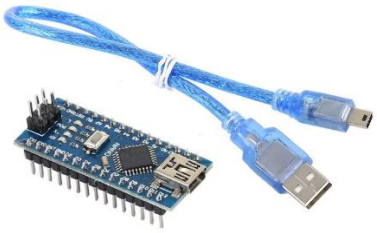


7) Por último, se debe repetir el paso 6 para los demás objetos con diferentes herramientas si desea hasta tener el diseño terminado

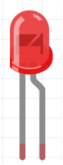
LISTA DE COMPONENTES ROBOT MÓVIL OMNIDIRECCIONAL

Para la construcción del robot móvil omnidireccional se requieren los siguientes componentes.

Tabla 8. *Componentes para construcción del robot móvil omnidireccional.*

COMPONENTE	CANTIDAD	REFERENCIA	DESCRIPCIÓN
Motor con encoder de cuadratura	3		BEMONOC 25GA370 DC Codificador de metal Motor de engranajes de 12 V de alta velocidad 150 RPM con codificador de efecto pasillo de dos canales para piezas de bricolaje.
Arduino nano con cable	1		Mini Nano V3.0 ATmega328P placa microcontrolador con cable USB para Arduino.
Driver DRV8833	2		Driver Dual Para Motores DC de color Rojo.
Modulo bluetooth HC-06	1		Necesario para la comunicación inalámbrica entre Python y Arduino.

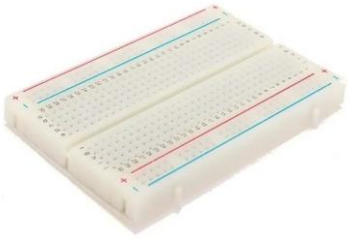
Batería lipo 2s 1000mAh	1		Batería Lipo Turnigy 1000mah 7.4v 30c
Cargador para batería lipo 2s y 3s	1		Cargador Balanceador Lipo 2s 7.4v 3s 11.1v Rc B3 Pro
Resistencia 5.1kΩ	1		Necesaria para el monitoreo de carga en la batería.
Resistencia 4.7kΩ	1		Necesaria para el monitoreo de carga en la batería.
Switch	1		Interruptor mediano de color negro para controlar el encendido y apagado del robot.
Ruedas omnidireccionales	3		Ruedas Omnidireccionales de plástico de 58 mm para plataformas robóticas ofrecen 360 grados de movimiento, rotación y movilidad

			lateral. movimiento y rotación en todas las direcciones, control de dirección fácil, giro y seguimiento rápidos
Led	1		Indicador de voltaje de la batería cuando esta requiera ser cargada nuevamente.
Borneras tres pines	6		Borneras 3 Pines Azules Conector Terminal Block T-block
Bornera dos pines	1		El conector del terminal de tornillo es fácil de instalar en la placa PCB
Jumper macho - macho	10		Permiten la conexión entre el circuito y la batería.
Pin header regleta hembra	4		Necesarias para la instalación del módulo bluetooth, driver DRV8833 y Arduino nano en la PCB.

LISTA DE COMPONENTES ROBOT ANTROPOMÓRFICO

Para la construcción del robot antropomórfico se requieren los siguientes componentes.

Tabla 9. *Componentes para construcción del robot antropomórfico.*

COMPONENTE	CANTIDAD	REFERENCIA	DESCRIPCIÓN
Servomotor Mg995	5		Modelo MG995, peso 55g, par de trabajo 15kg/cm, Velocidad de reacción 53-62R/M con engranes metálicos.
Arduino uno	1		Placa microcontrolador ATmega328 con cable USB para Arduino.
Mini Protoboard	1		Tablilla de pruebas de 400 Puntos - 8.5cm X 5.5cm
Cables Jumper Macho - Hembra	10		Permiten la conexión entre el circuito y la batería.

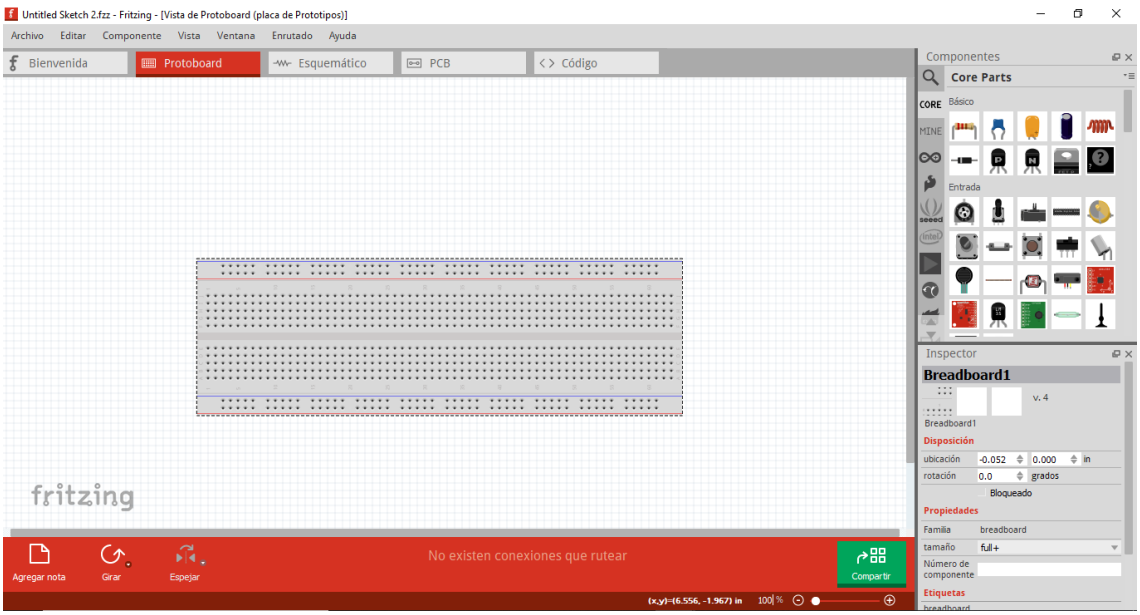
Rodamientos	2		Rodamiento rígido de bolas dimensiones 8mm x22mm x 7mm
Tabla	1		Tabla de MDF para el soporte del robot
Fuente de poder	1		Marca sompom S-100-5. Salida de 5V a 20ª
Tornillos	25		Tornillo milimétrico M4 para ensemble del robot

ESQUEMA ELÉCTRICO

Para el diseño del esquema eléctrico realizado en ambos prototipos de robots se hizo uso del *software* de *Fritzing* versión 0.9.3 portable cuya interfaz se puede observar en la figura 120, este es un hardware de código abierto que permite hacer de la electrónica algo más accesible a las personas.

Cuenta con gran variedad de componentes electrónicos visuales en 2D lo que permite una fácil creación de esquemas eléctricos. También permite crear esquemas de circuitos para una posterior fabricación. Además, cuenta con una página web que sirve de ayuda para que el usuario pueda mejorar su experiencia en el uso de esta herramienta.

Figura 120. Entorno gráfico de Fritzing



Fuente: (*Fritzing*, n.d.)

Para el diseño del esquema electrónico realizado en *Fritzing* se deber importar las librerías para el *driver DRV8833* y los motores de cuadratura correspondientemente, esto se hace porque el software por defecto no la tiene integradas.

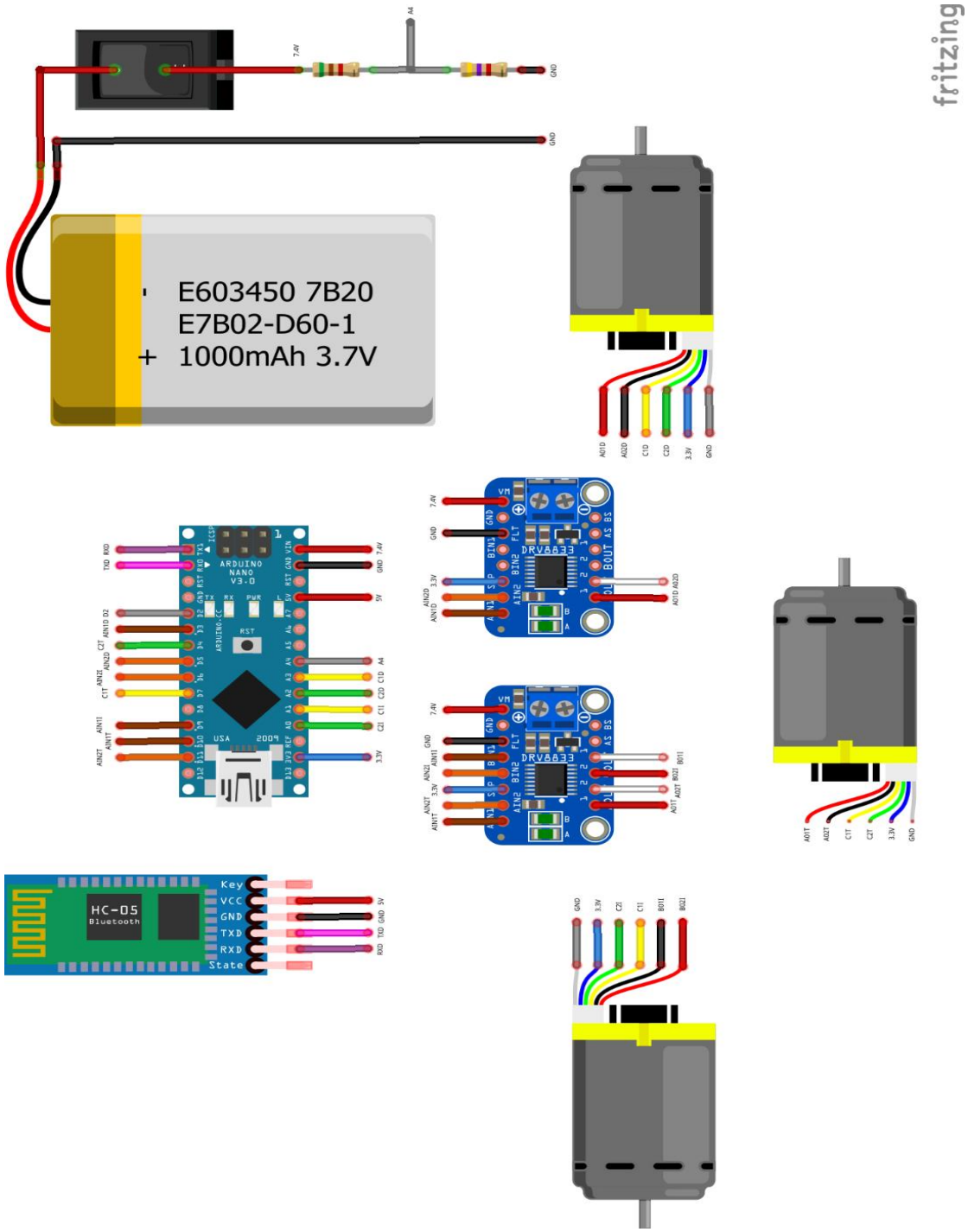
La librería para el *driver DRV8833* se puede encontrar en internet como *Adafruit DRV8833 DC-Stepper Motor Driver*, mientras que la librería para los motores de cuadratura se puede ubicar como *DC Motor with two-phase hall-encoder*. Al descargar ambas librerías lo siguiente es pegarlas en la carpeta donde tengamos ubicados el programa.

Con las dos librerías debidamente instaladas en el programa se procede a diseñar el esquema eléctrico del robot móvil el cual queda como se muestra a continuación.

ESQUEMA DEL CIRCUITO DEL ROBOT MOVIL OMNIDIRECIONAL

En la figura 121 se muestra el esquema eléctrico realizado para el robot móvil omnidireccional.

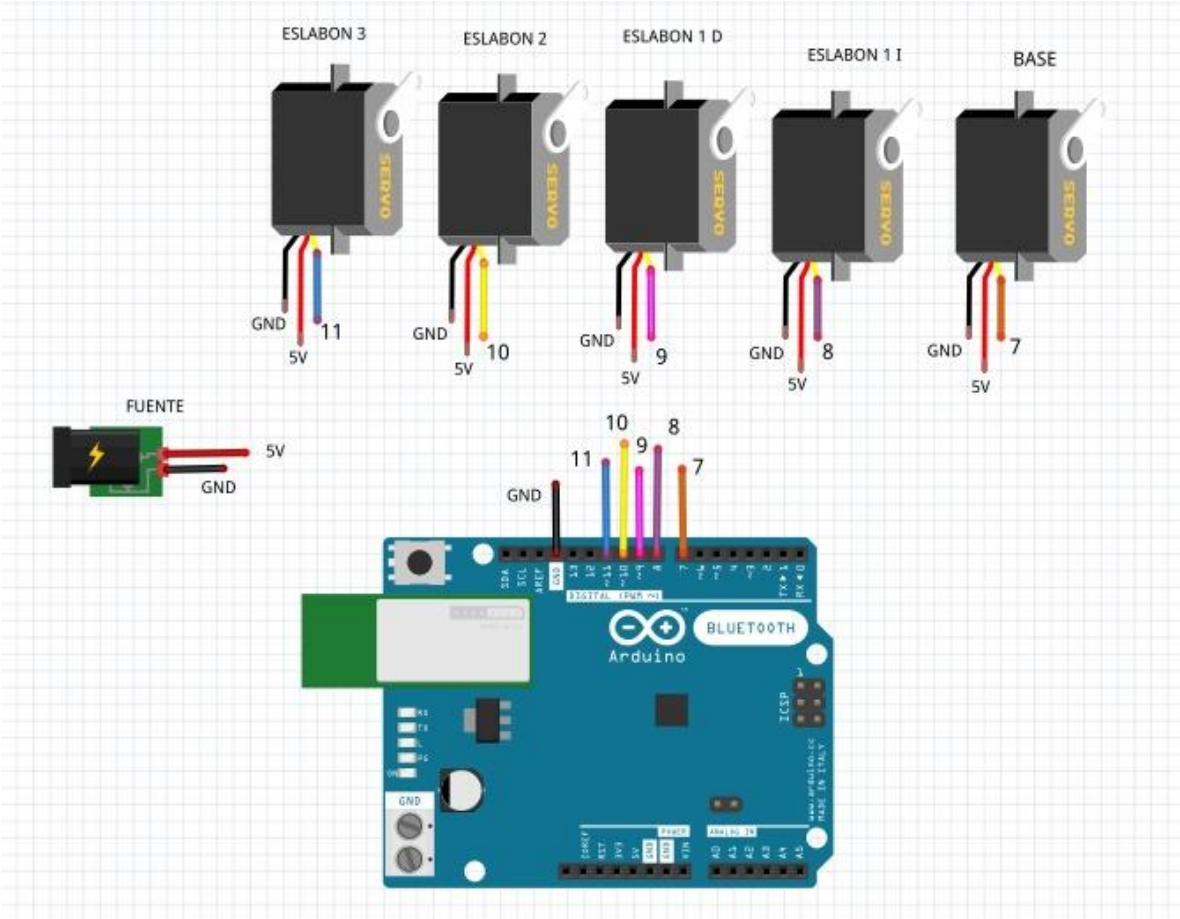
Figura 121. Esquema eléctrico robot móvil omnidireccional *Fritzing*.



ESQUEMA DEL CIRCUITO DEL ROBOT ANTROPOMÓRFICO

Representación del esquema electrónico del robot móvil omnidireccional desarrollado en *Fritzing* se puede observar en la figura 122.

Figura 122. Esquema eléctrico robot móvil omnidireccional *Fritzing*.



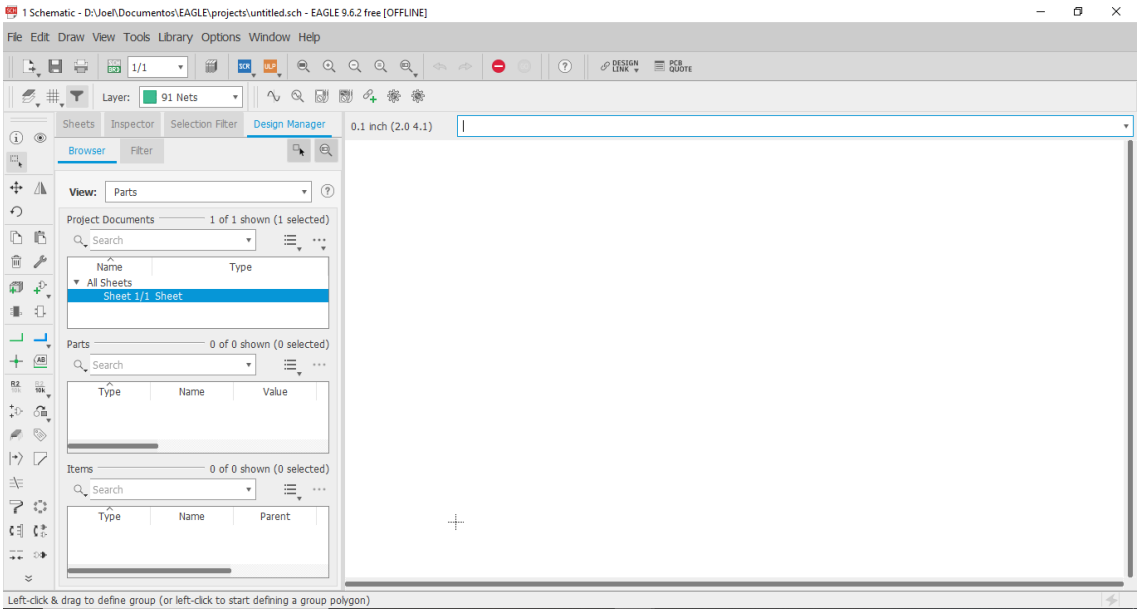
Fuente: Elaboración propia.

DISEÑO ELECTRÓNICO

Para el diseño electrónico de los circuitos dispuestos en los robots, se implementa el software de Eagle en su versión 9.6.2 free ya que es una herramienta que permite el diseño de diagramas y PCBS de manera sencilla permitiendo así agilizar los tiempos de fabricación, además es de gran ayuda porque permite que los circuitos tengan aspectos más profesionales necesarios al momento de implementar proyectos electrónicos.

Por otra parte, es una herramienta que cuenta con una gran variedad de componentes electrónicos incorporados que ayudan en la creación de circuitos electrónicos. También cuenta con auto rúter que es una opción muy interesante y tal vez una de las más importantes porque ayuda a generar de manera automática las pistas en el circuito agilizando los tiempos de trabajo. En la figura 123 se puede observar el entorno grafico del software.

Figura 123. Entorno de Eagle

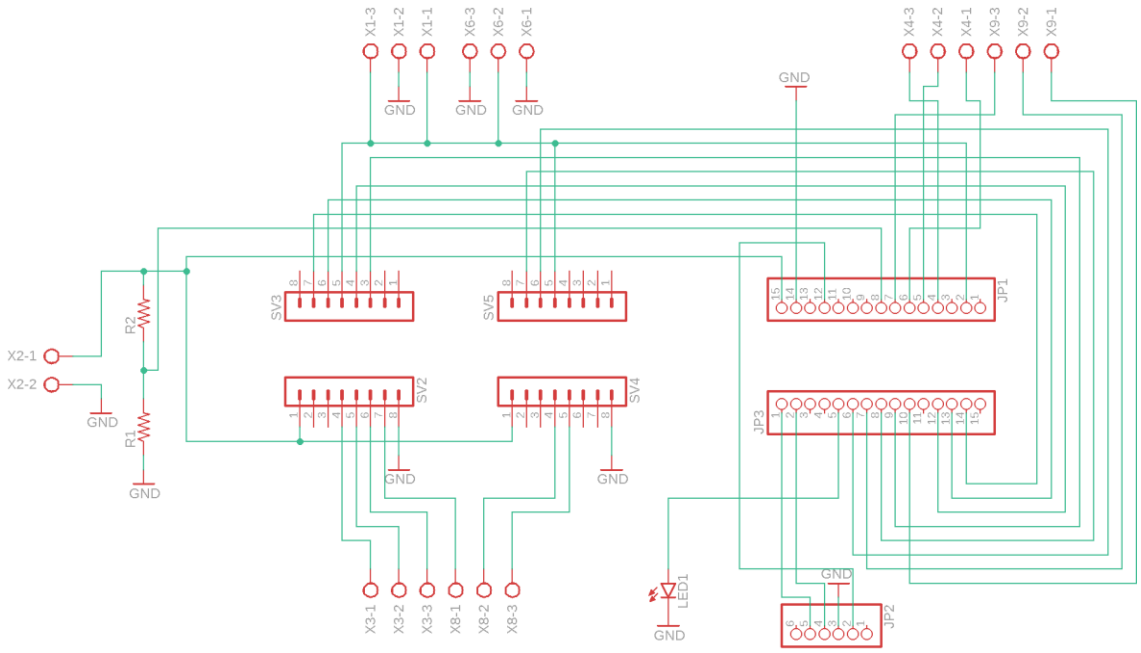


Fuente: (*EAGLE | PCB Design And Electrical Schematic Software | Autodesk*, n.d.)

Para realizar el archivo esquemático del circuito electrónico del robot móvil omnidireccional dentro del entorno de *Eagle* mostrado en la figura 123 se utilizan herramientas

como resistencias, borneras, *Pinhead* entre otros, quedando representado como se muestra en la siguiente figura 124.

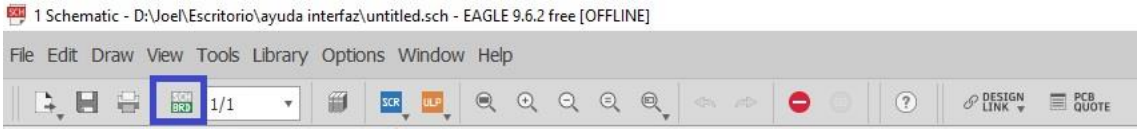
Figura 124. Esquema de conexiones robot móvil.



Fuente: Elaboración propia.

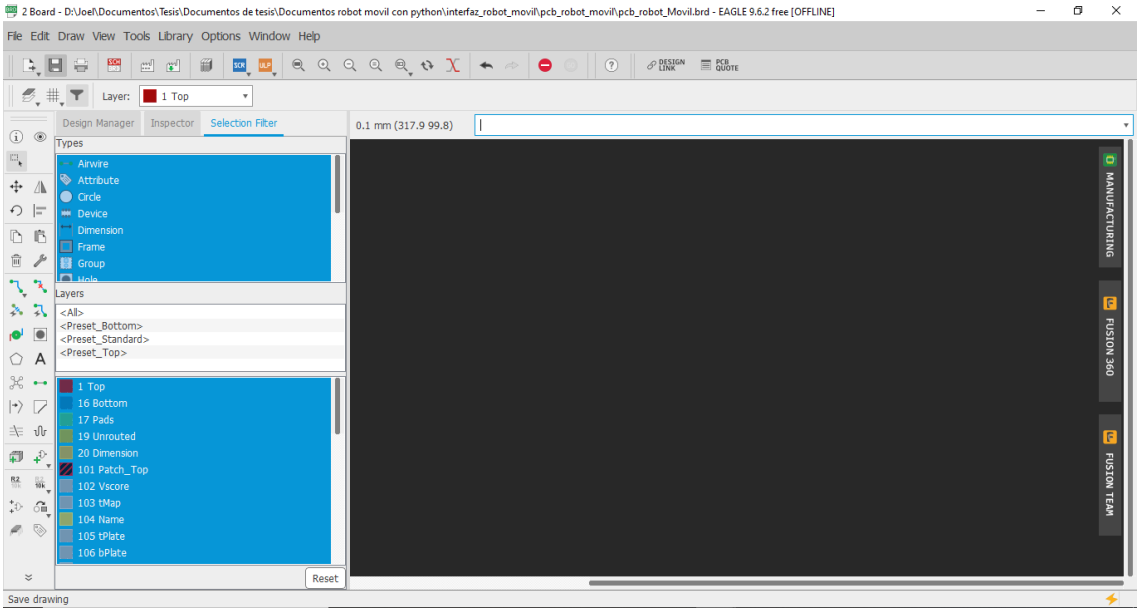
Una vez realizadas las conexiones necesarias en el archivo esquemático se procede hacer clic en la opción *Generate/switch to board* mostrada en la siguiente figura 125 y cual se resalta en color azul.

Figura 125. *Board*



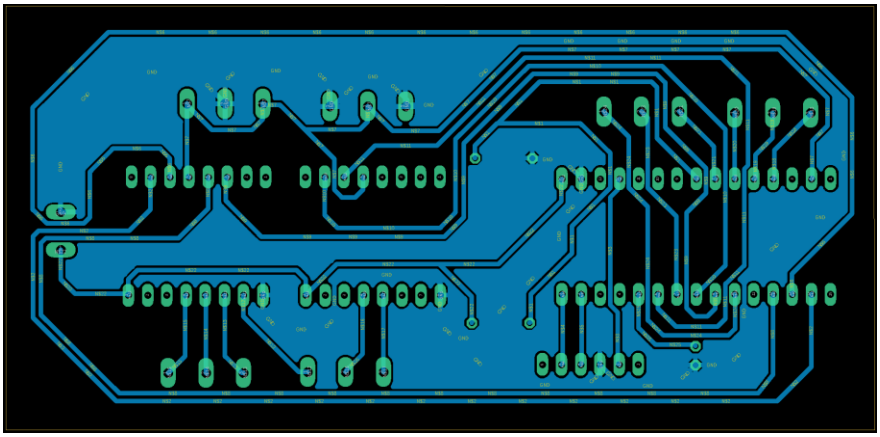
Inmediatamente se abre un nuevo entorno como el que se muestra en la figura 126 donde se estructuran todos los componentes electrónicos y conexiones realizadas.

Figura 126. Interfaz para *Board*



Luego de configurar todos los parámetros como tamaños de pistas, grosor, capas, entre otros en la parte esquemática de la figura 126 se obtiene como resultado el diseño de la PCB mostrada en la figura 127 la cual queda lista para su posterior fabricación.

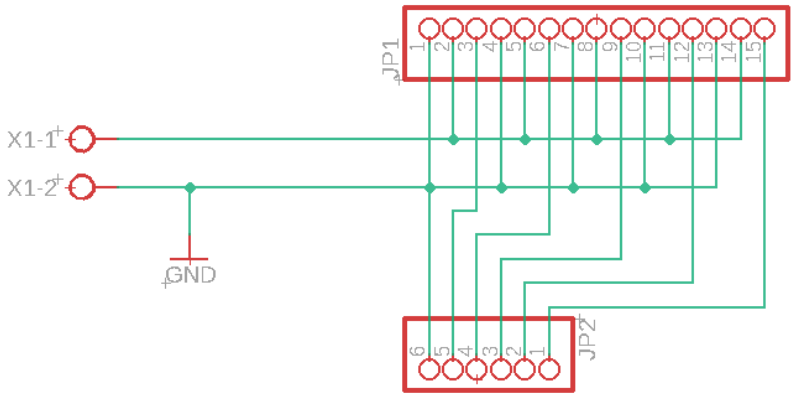
Figura 127. PCB robot móvil



Fuente: Elaboración propia.

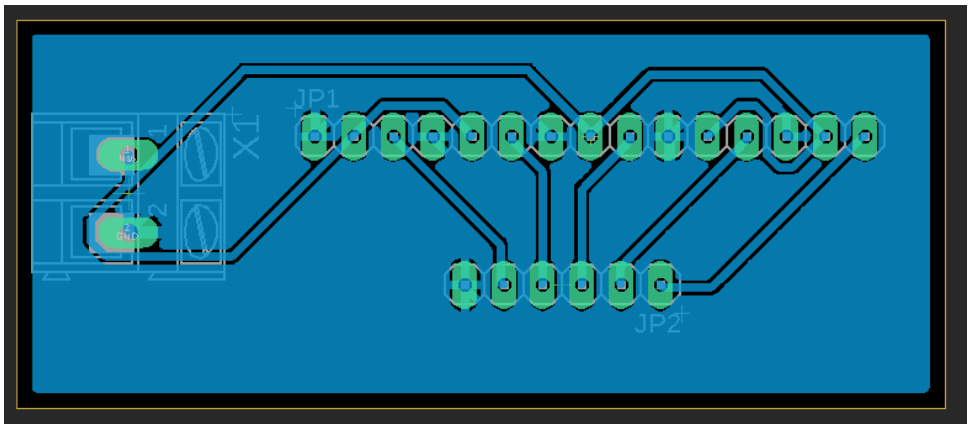
De igual manera, en la figura 128 se puede observar el esquema eléctrico del robot antropomórfico

Figura 128. Esquema de conexiones robot antropomórfico



Al igual que la PCB de la figura 128, después de configurar todos los parámetros de las pistas y conexiones se obtiene el diseño eléctrico final del robot antropomórfico para su posterior fabricación el cual se muestra en la figura 129.

Figura 129. PCB robot antropomórfico

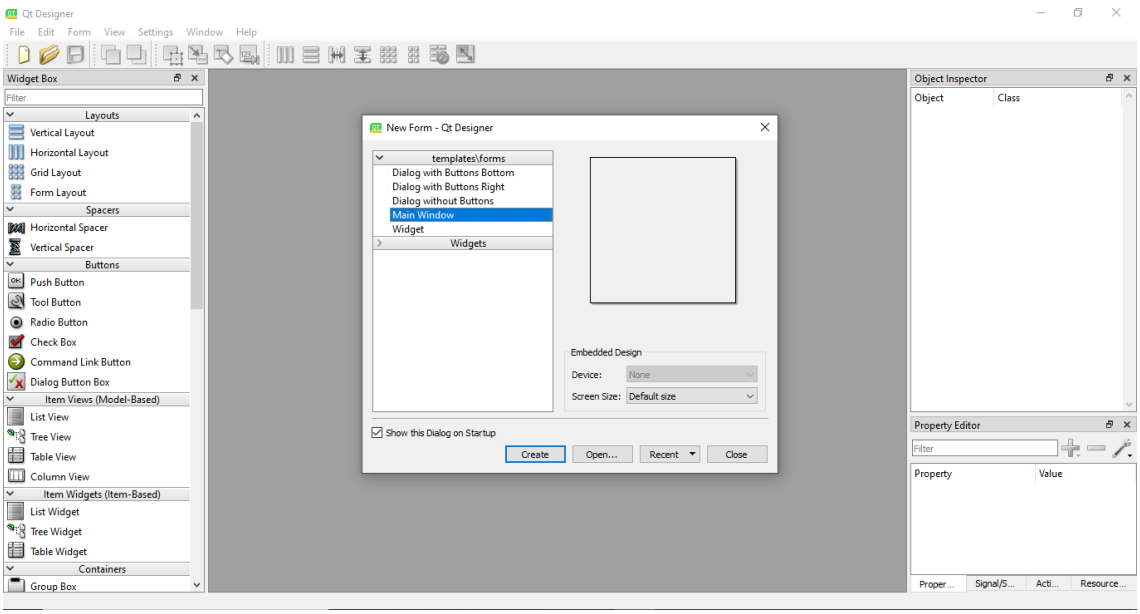


Fuente: Elaboración propia

INTERFAZ DE USUARIO

Para el diseño de la interfaz se utilizó *QTDESIGNER* en su versión 5.11.1, este es un *software* de código abierto que permite la creación de interfaces de usuario de manera intuitiva, sencilla y práctica permitiendo incluso personalizar las ventanas y los cuadrados de diálogos. Para ellos es necesario tener instalado el *software* en nuestro PC y cuyo entorno se muestra en la figura 130.

Figura 130. Entorno QTDESIGNER



Fuente: (*Manual Del Diseñador Qt*, n.d.)

A continuación, después de tener presente el tipo de interfaz que se quiere desarrollar y luego de haber implementados varias herramientas como *Vertical Layout, Push Button, Label, Horizontal Spacer, Frame* entre otros, la interfaz queda desarrollada como se muestra en la figura 131.

Figura 131. Interfaz de usuario.

USO DE PYTHON

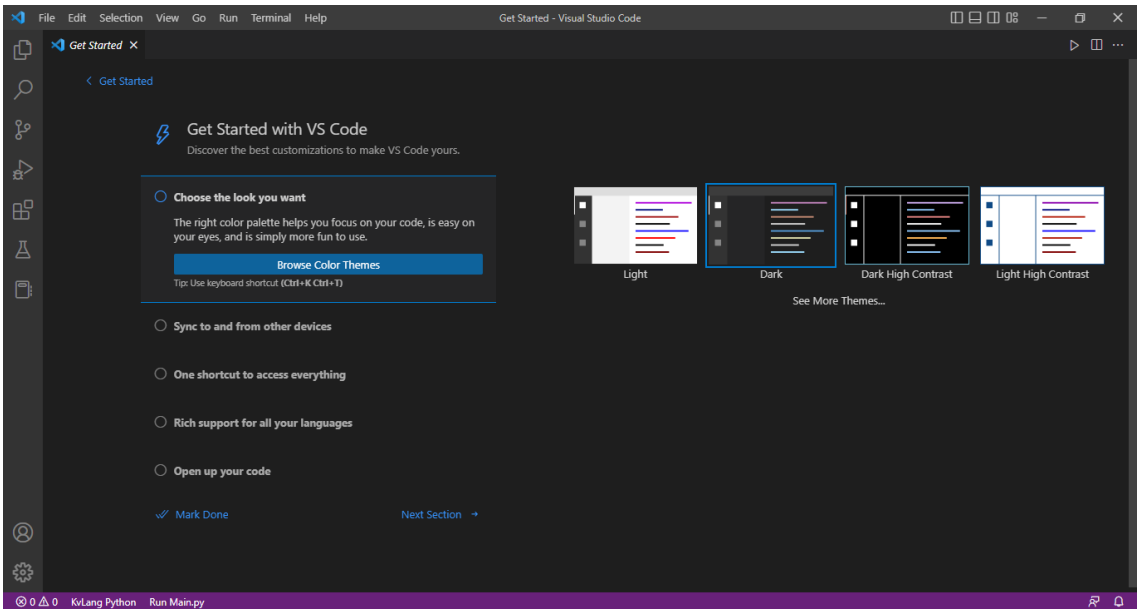
Para el desarrollo de la programación de los robots se requiere la instalación de Python en su versión 3.8.6, esto es necesario ya que algunas librerías requeridas solo funcionan de manera correcta con la implementación de esta versión en específico, esto contribuye a una mayor estabilidad y evita que se presenten algunos errores inesperados al momento de ejecutar los códigos.

EDITOR DE CÓDIGO UTILIZADO

Como editor de código implementado en *Python* se utilizó *Visual Studio Code* en su versión 1.72.2, este es un software gratuito y de código abierto multiplataforma compatible con varios lenguajes de programación lo que es útil al momento de querer realizar trabajos de forma más ágil, también cuenta con una gran cantidad de extensiones que brindan la posibilidad de

escribir y ejecutar los códigos ayuda de autocompletado y cuyo entorno se muestra en la figura 132.

Figura 132. Visual Studio Code.



Fuente: (*Documentación Para El Código de Visual Studio*, n.d.)

Apéndice B

Guías didácticas de ayuda para la conexión del robot móvil omnidireccional y el robot antropomórfico.

A continuación, se muestra la guía de ayuda didáctica para para una correcta conexión inalámbrica entre el robot móvil y la interfaz de usuario.

GUÍA DE AYUDA PARA USO CORRECTO DE LA INTERFAZ DE USUARIO ROBOT MÓVIL OMNIDIRECCIONAL

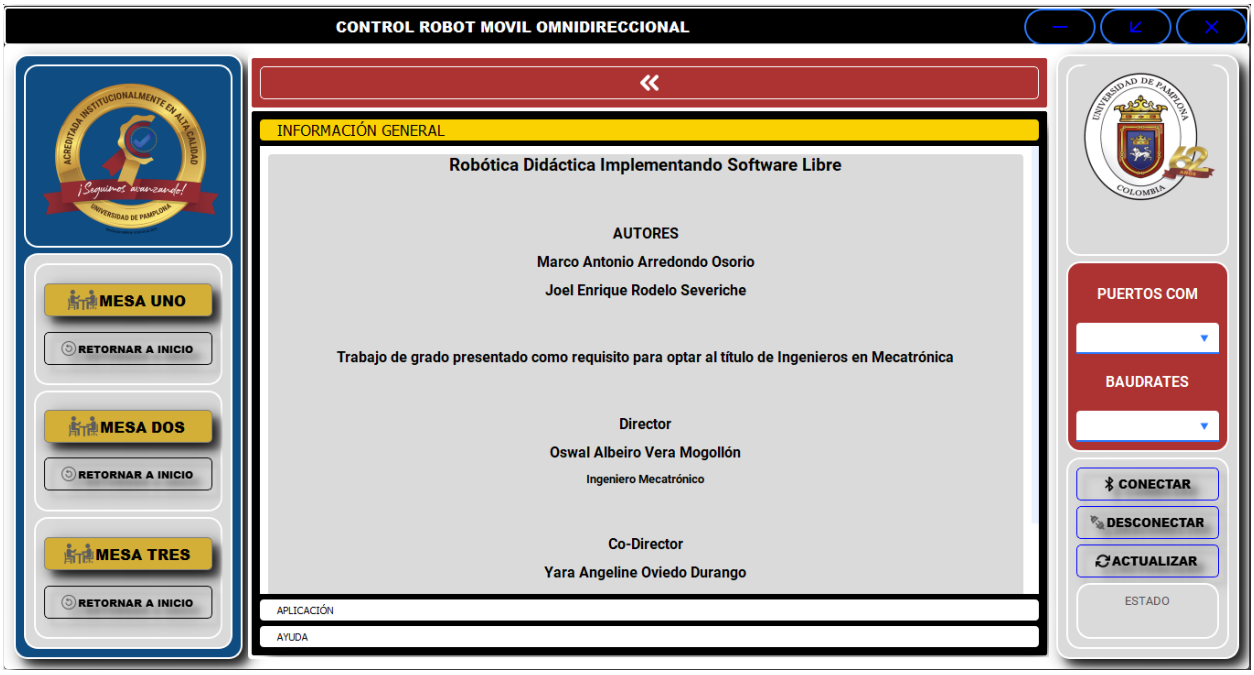
El siguiente instructivo tiene como finalidad guiar al estudiante en el uso correcto de la interfaz de usuario para el manejo del robot móvil omnidireccional partiendo desde su configuración inicial al vincular el módulo bluetooth de forma inalámbrica con el PC hasta la posterior puesta en marcha del robot.

Inicialmente, se detallan algunas ventanas dispuestas en la interfaz de usuario las cuales se muestran en la siguiente figura.

INTERFAZ DE USUARIO

El objetivo principal de la interfaz de usuario es brindar al estudiante una interacción sencilla y practica con el robot móvil omnidireccional facilitando tener una experiencia de usuario satisfactoria.

Figura 133. Panel de usuario robot móvil omnidireccional



Fuente: Elaboración propia.

PANEL DE COMUNICACIÓN

En la figura 133 se encuentran disponibles los elementos necesarios para lograr la comunicación serial inalámbrica entre la interfaz de usuario y el robot móvil omnidireccional.

Descripción De Los Elementos Del Panel De Comunicación

Puertos COM: Permite la selección del puerto disponible por medio del cual se va a establecer la comunicación serial entre la interfaz de usuario y el módulo bluetooth dispuesto en el robot móvil.

Baudrates: Es la velocidad de comunicación a la cual se van a enviar los datos y cuyo valor por defecto está establecido en 9600.

Conectar: Este botón permite llevar a cabo la conexión inalámbrica entre la interfaz de usuario y el módulo bluetooth dispuesto en el robot una vez se seleccione el puerto COM y los baudrates.

Desconectar: Como su nombre lo indica, este botón permite enviar datos desde la interfaz de usuario al módulo bluetooth de robot, es decir que al presionar este botón la comunicación se va a interrumpir.

Actualizar: Este botón es necesario ya que permite mostrar los puertos COM y los baudrates en la interfaz de usuario disponibles al momento de establecer la conexión ya que por defecto estos campos aparecen en blanco.

Estado: Es el encargado de mostrar al usuario el estado en el que se encuentra la comunicación, sea conectado o desconectado.

PANEL DE CONTROL

Este panel permite controlar el robot móvil omnidireccional, en él se encuentran incorporados los botones mesa uno, mesa dos, mesa tres y los botones de retornar a inicio. Cada uno de estos botones incorpora dentro trayectorias propias que dependen de la ubicación de cada mesa dispuesta en el restaurante.

Descripción de los botones

Básicamente lo que hacen los botones de la mesa uno, mesa dos y mesa tres es enviar el robot móvil omnidireccional a la posición donde se encuentre ubicada cada mesa, en estos botones se encuentra integrada las trayectorias que permiten guiar al robot para posicionarlo en la ubicación correspondiente.

Retornar A Inicio: Permiten que el robot regrese a su posición inicial una vez el usuario confirme su pedido. Al igual que los botones de mesa uno, mesa dos y mesa tres, estos cuentan con una trayectoria previamente integrada para poder regresar a la posición inicial de partida.

PANEL DE INFORMACIÓN

En este panel se presentan datos relevantes tales como información general, aplicación y ayuda, todos estos necesarios para conocer detalles del diseño de la interfaz de usuario.

Descripción de los datos dispuestos en panel de información general

Información General: En esta pestaña se encuentran detalles relacionados con el desarrollo del trabajo de investigación, incluidos los nombres de los usuarios que participaron en la realización de este proyecto.

Aplicación: En esta pestaña se describen algunos aspectos importantes relacionados al funcionamiento del robot móvil omnidireccional y su aplicación.

Ayuda: Detalla algunos aspectos importantes que se deben tener en cuenta para el uso adecuado tanto de la interfaz de usuario como la del robot móvil omnidireccional.

CONEXIÓN ENTRE LA INTERFAZ DE USUARIO Y EL ROBOT MÓVIL OMNIDIRECCIONAL

A continuación, se detallan los pasos a seguir para que el estudiante logre una correcta conexión y posterior comunicación entre la interfaz de usuario y el robot móvil omnidireccional.

Para lograr una comunicación entre la interfaz de usuario y el robot móvil omnidireccional primero debe asegurarse de tener encendido el bluetooth en su computadora portátil, esto es necesario ya que se va a establecer una comunicación serial de forma inalámbrica entre los dos dispositivos.

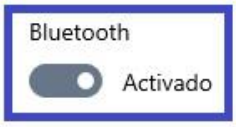
Esto se hace escribiendo “configuración de bluetooth y otros dispositivos” en el panel de inicio de nuestra computadora y al dar ENTER se muestra una ventana como la siguiente.

Figura 134. Configuración

Fuente: (*Disfruta Del Poder Del SO, Computadoras y Apps Windows 11 | Microsoft*, n.d.)

En la figura 134 se ubica la opción que dice Bluetooth y se activa como se muestra en la figura 135.

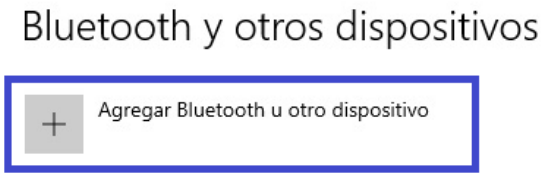
Figura 135. Bluetooth



Una vez se encuentre activado el bluetooth de la PC lo siguiente es vincularlo con el módulo HC-06 dispuesto en el robot móvil por medio del cual se va a establecer la comunicación inalámbrica.

Para esto se va a la opción que dice Agregar Bluetooth y otros dispositivos como se muestran en la figura 136 y se da clic.

Figura 136. Bluetooth y otros dispositivos

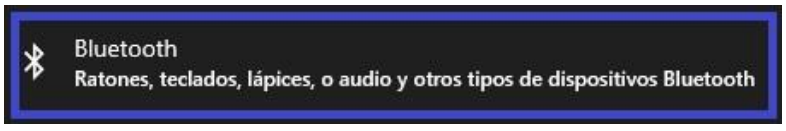


A continuación, se abre la ventana mostrada en la figura 137 dentro de la cual se va a establecer la comunicación entre el PC y el módulo HC-06 del robot móvil al poder vincular ambos dispositivos.

Figura 137. Agregar dispositivos.

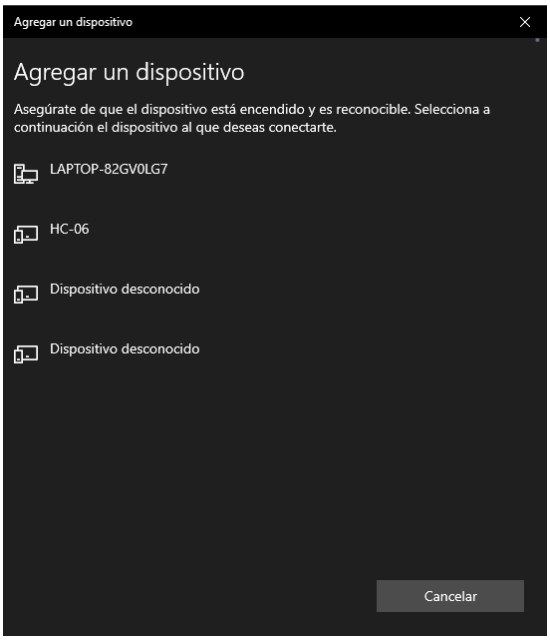
Para vincular los dos dispositivos se hace clic en la opción mostrada en la figura 138.

Figura 138. Buscar dispositivos.



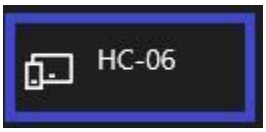
Una vez seleccionada la opción en la figura 138 se despliega una ventana que muestra todos los dispositivos disponibles en ese momento como indica la figura 139.

Figura 139. Dispositivos disponibles.



Dentro de las opciones disponibles se busca el módulo HC-06 y se selecciona tal y como aparece en la figura 140.

Figura 140. Modulo HC-06.

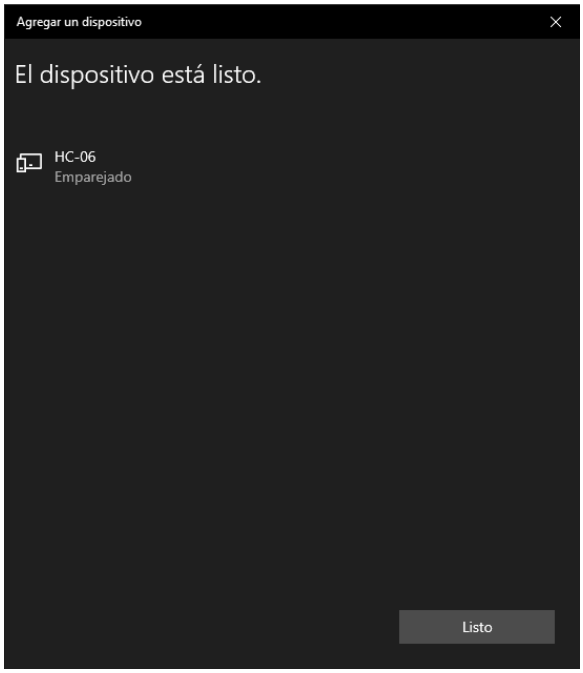


Al hacer clic sobre el módulo HC-06 se despliega una opción en la que se debe introducir un código que permite vincular el PC con el dispositivo y el cual se muestra en la figura 141. El código por defecto del módulo es 1234.

Figura 141. PIN módulo HC-06.

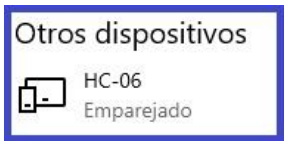
Una vez introducido el código de cuatro dígitos y presionado la opción de conectar se espera que se establezca la conexión. Cuando el dispositivo esté listo para usar se muestra una ventana como la mostrada en la figura 142 indicando que la conexión ha sido establecida con éxito. Se presiona la opción de listo para cerrar la ventana.

Figura 142. Conexión módulo HC-06.



Para garantizar que el dispositivo quede realmente vinculado al PC se regresa a la ventana inicialmente abierta en la figura 134 observando la opción que dice Otros dispositivos y se asegura que diga emparejado como se muestra en la figura 143.

Figura 143. Emparejado.



A continuación, lo que queda es establecer el puerto COM del módulo HC-06 al cual se va a conectar la interfaz de usuario. Este por defecto al realizar la actualización de los puertos genera dos opciones entre los cuales se encuentra un puerto tipo saliente y otro tipo entrante.

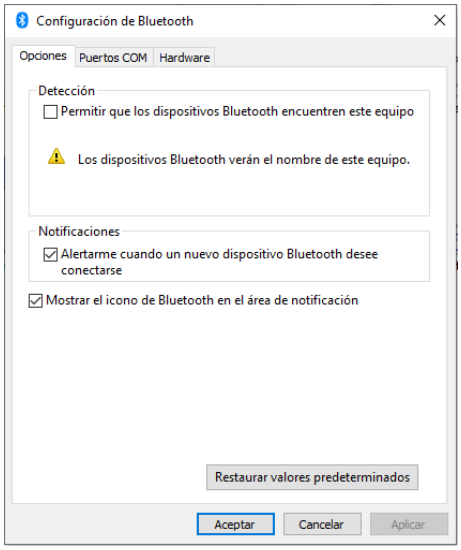
El puerto sobre el cual se debe poner principal atención es el de tipo saliente porque este en particular va a permitir que se puedan enviar los datos desde el PC para que el módulo HC-06 los pueda recibir y procesar. Para ubicar estos dos puertos se debe hacer lo siguiente

Se posiciona de nuevo en la ventana que se abrió inicialmente en la figura 134 para activar el bluetooth del PC y se busca la opción “Más opciones de Bluetooth” descrita en la figura 144.

Figura 144. Opciones.

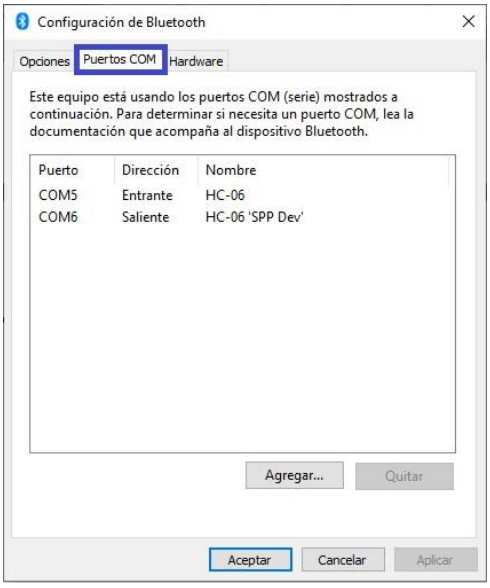
Al hacer clic en esta opción se muestra una nueva ventana como indica la figura 145.

Figura 145. Configuración.



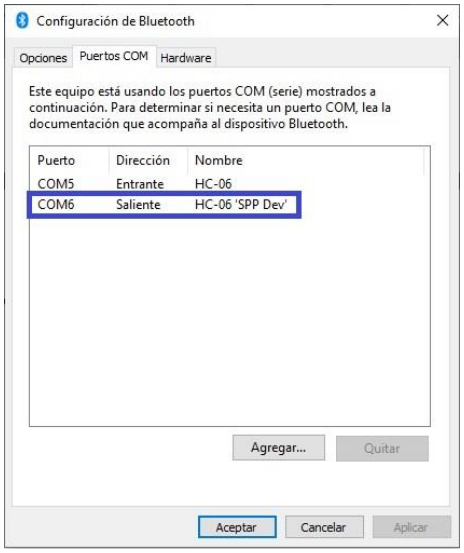
Se ubica la pestaña que dice “puertos COM” indicada en la figura 146 y al hacer clic sobre ella se muestra información de los dos puertos disponibles en el módulo HC-06.

Figura 146. Puertos COM.



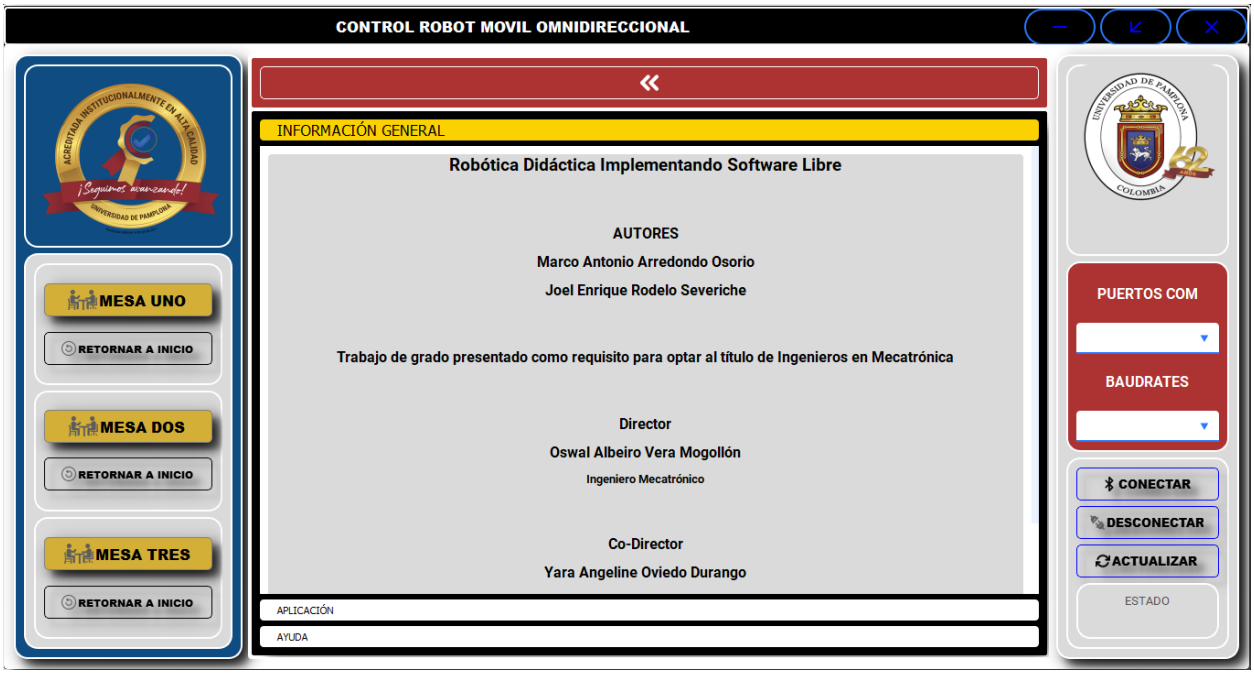
De igual forma, se observa el puerto donde se ubique la opción Saliente de la figura 147 que para este caso es el COM6 y el cual se debe tener en cuanta al momento de seleccionarlo para poder establecer la conexión entre la interfaz de usuario y el robot.

Figura 147. Saliente.



CONTROL DEL ROBOT MOVIL OMNIDIRECCIONAL Y COMUNICACIÓN

Figura 148. Conexión entre interfaz de usuario y robot móvil.



Fuente: Elaboración propia.

Para acceder a los puertos dispuestos en el PC, inicialmente se debe presionar el botón actualizar en la figura 148 para visualizar los puertos disponibles. Seguidamente, se selecciona el puerto que previamente se ubicó como saliente, para este caso el COM6, los *baudrates* por defecto quedan establecidos en 9600. Se procede a presionar el botón conectar y se esperan unos minutos que la opción de estado diga conectado indicando que se ha establecido la comunicación entre la interfaz de usuario y el robot móvil.

Finalmente, una vez establecida la comunicación con las indicaciones expuestas anteriormente, se puede acceder al menú de control donde se encuentran los botones de las tres mesas y dentro de los cuales se especifican las trayectorias a las que se envía el robot dependiendo la posición donde se encuentren ubicadas.

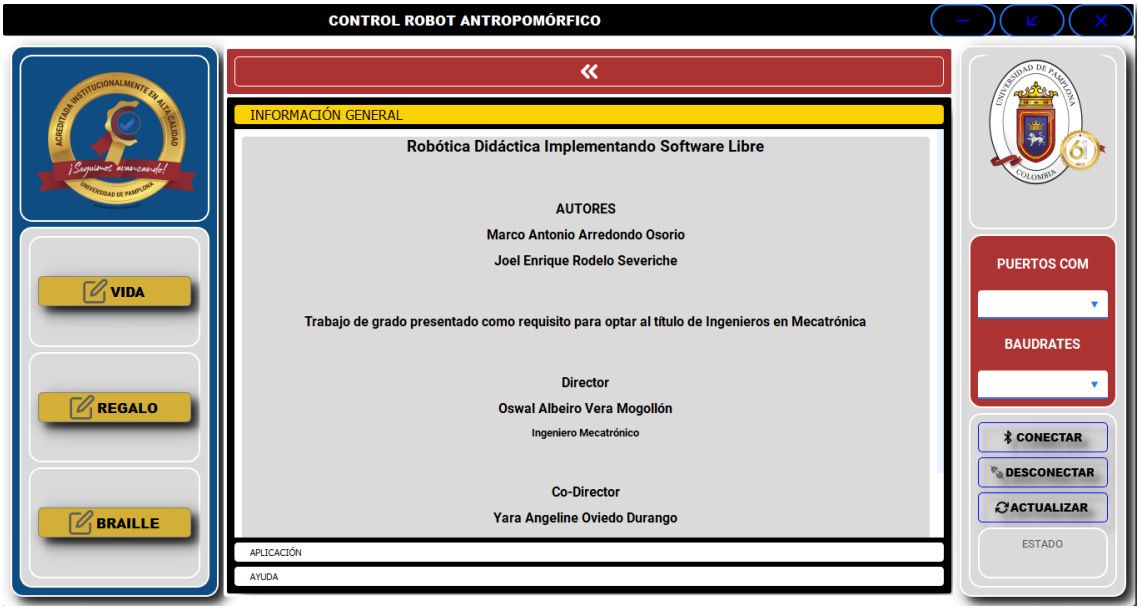
GUÍA DE AYUDA PARA USO CORRECTO DE LA INTERFAZ DE USUARIO ROBOT ANTROPOMÓRFICO

El siguiente instructivo tiene como finalidad guiar al estudiante en el uso correcto de la interfaz de usuario para el manejo del robot partiendo desde su configuración inicial hasta la puesta en marcha del robot.

INTERFAZ DE USUARIO

La interfaz de usuario mostrada en la figura 149 tiene como objetivo brindar al estudiante una interacción más sencilla y practica entre el robot antropomórfico y el PC facilitando así el uso correcto entre ambas herramientas.

Figura 149. Panel de usuario robot antropomórfico



Fuente: Elaboración propia.

PANEL DE COMUNICACIÓN

En este panel se encuentran disponibles los elementos necesarios para lograr la comunicación serial entre la interfaz de usuario y el robot y se puede observar en la figura 150.

Figura 150. Panel de comunicación

Fuente: Elaboración Propia.

Puertos COM: Permite la selección del puerto disponible por medio del cual se va a establecer la comunicación serial

Baudrates: Es la velocidad de comunicación a la cual se van a enviar los datos el cual por defecto está establecido en 9600.

Conectar: Este botón permite llevar a cabo la conexión entre la interfaz de usuario y el robot antropomórfico una vez se seleccione el puerto COM y los baudrates.

Desconectar: permite dejar de enviar datos es decir al presionar este botón la comunicación se va a interrumpir.

Actualizar: Este botón es necesario porque nos permite actualizar el puerto COM y los baudrates ya que por defecto estos aparecen en blanco.

Estado: Es el encargado de mostrar al usuario el estado en que se encuentra la comunicación, sea conectado o desconectado.

PANEL DE CONTROL

Este panel permite controlar el robot antropomórfico, en él se encuentran presentes los botones donde se visualizan las palabras vida, regalo y braille, estos tienen incorporados los puntos necesarios para lograr la escritura de cada palabra en lenguaje braille y se puede mirar en la figura 151.

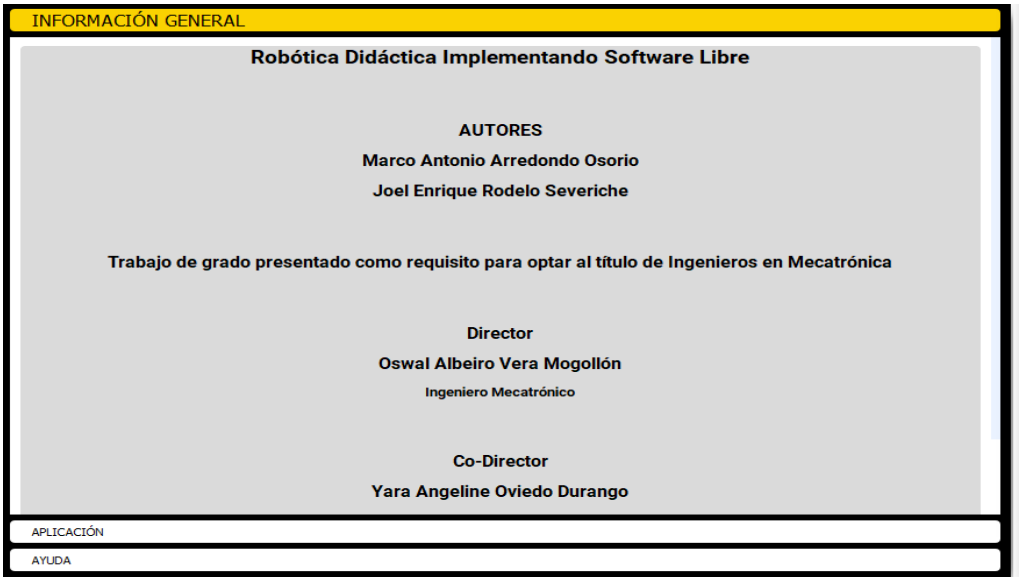
Figura 151. Panel de control

Fuente: Elaboración propia.

PANEL DE INFORMACIÓN

En este panel se presentan datos relevantes tales como información general, aplicación y ayuda, su interfaz se presenta en la figura 152, estos son necesarios para conocer detalles del diseño de la interfaz de usuario.

Figura 152. Panel de información



Fuente: Elaboración propia.

Descripción de los datos dispuestos en panel de información general

Información General: En esta pestaña se encuentran detalles relacionados con el desarrollo del trabajo de investigación, incluidos los nombres de los usuarios que participaron en la realización de este proyecto.

Aplicación: Dentro de esta se describen algunos aspectos importantes relacionados al funcionamiento del robot antropomórfico

Ayuda: Detalla algunos aspectos importantes que se deben tener en cuenta para el uso adecuado tanto de la interfaz de usuario como la del robot antropomórfico.

CONTROL DEL ROBOT ANTROPOMÓRFICO Y COMUNICACIÓN

Figura 153. Interfaz de usuario robot antropomórfico

Fuente: Elaboración propia.

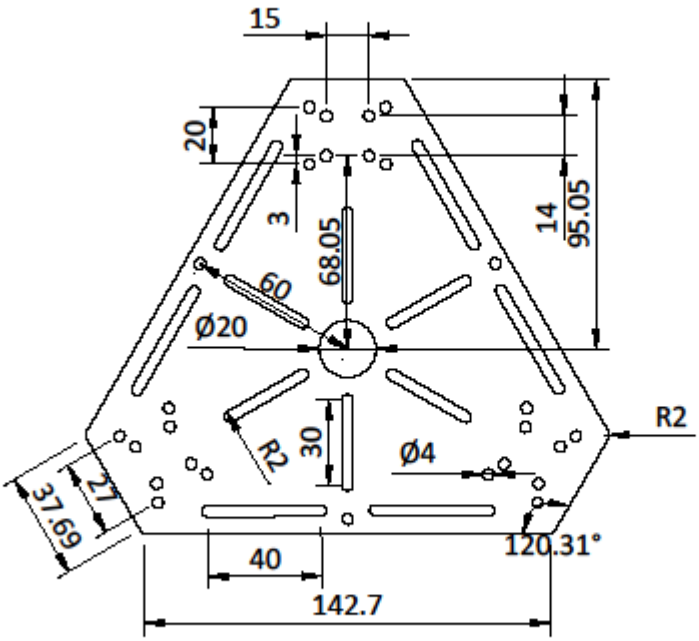
Para acceder a los puertos dispuestos en el PC, inicialmente se debe presionar el botón actualizar para visualizar los puertos disponibles mostrados en la figura 153. Posteriormente, se selecciona el puerto disponible para establecer la comunicación, para este caso el COM4, los *baudrates* por defecto quedan establecidos en 9600. Se procede a presionar el botón conectar y se esperan unos minutos que la opción de estado diga conectado indicando que se ha establecido la comunicación entre la interfaz de usuario y el robot móvil.

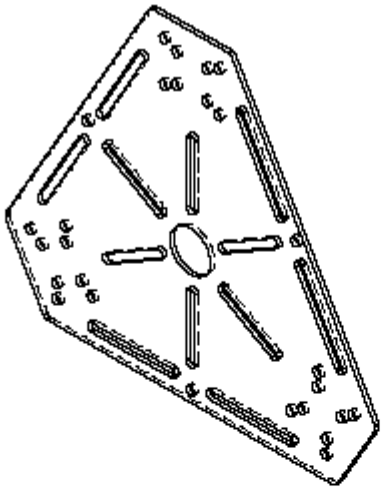
Finalmente, una vez establecida la comunicación con las indicaciones expuestas anteriormente, se puede acceder al menú de control donde se encuentran los botones de las tres mesas y dentro de los cuales se especifican las trayectorias a las que se envía el robot dependiendo la posición donde se encuentren ubicadas.

ANEXOS

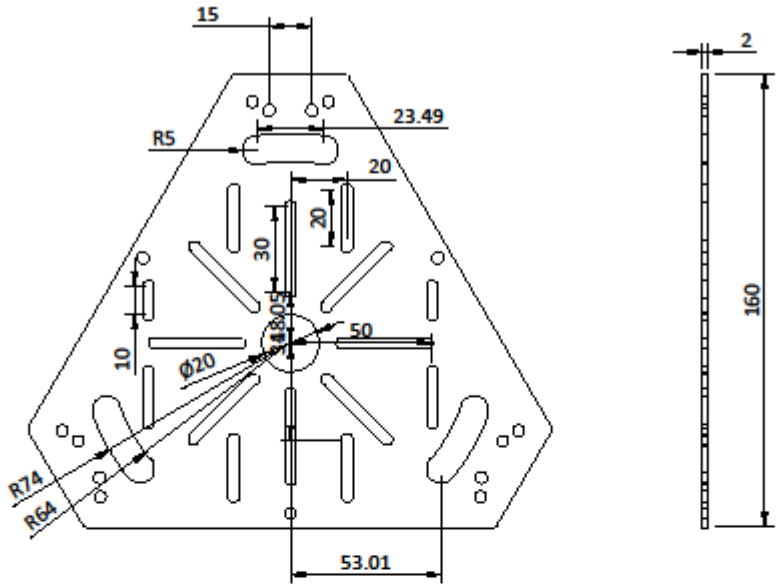
PLANOS ROBOT MÓVIL

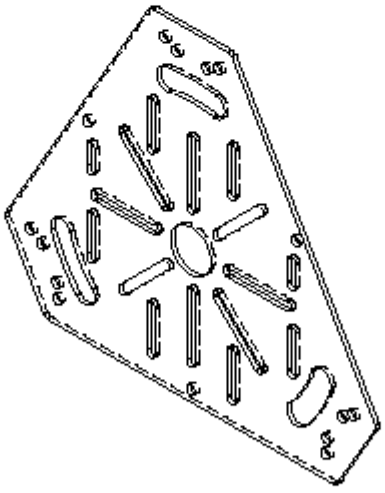
A continuación, se muestran los planos necesarios para la construcción del robot móvil omnidireccional con sus respectivas cotas y vistas.



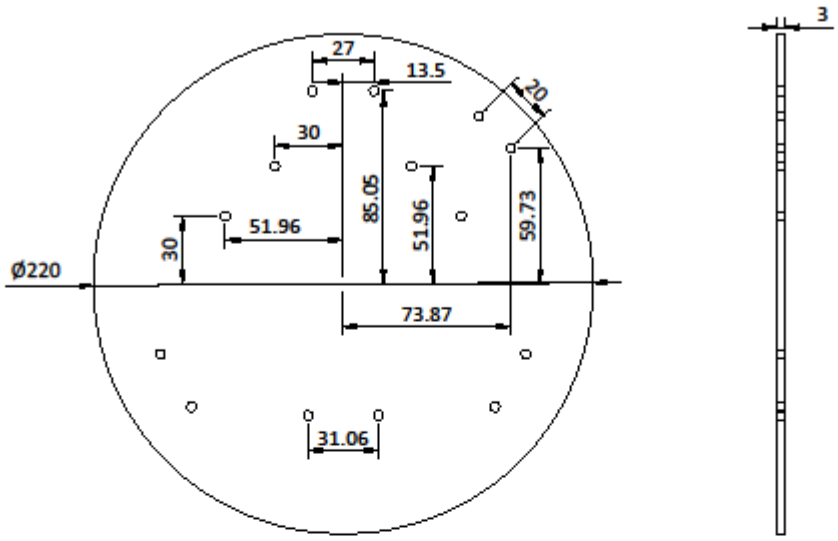


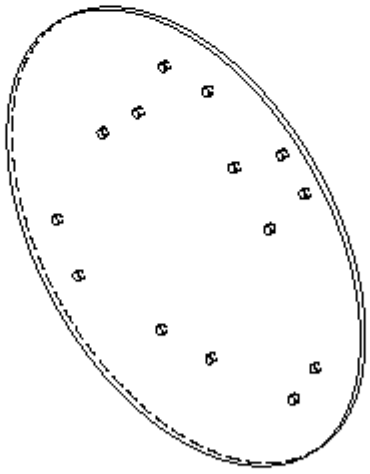
Dept.	Technical reference	Created by	Approved by
		Joel Rodelo 29/10/2022	
		Document type	Document status
		Title: Base_I_Robot_Movil v5	DWG No.
			Rev. / Date of issue / Sheet: 1/1



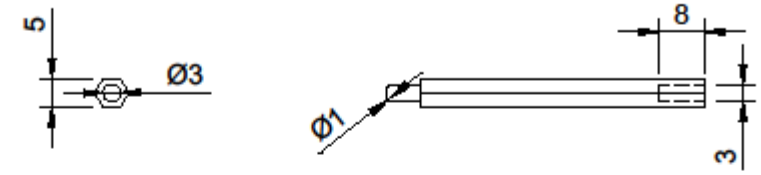


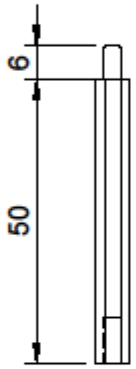
Dept.	Technical reference	Created by	Approved by
		Joel Rodelo 29/10/2022	
		Document type	Document status
		Title: Base_II_Robot_Movil	DWG No.
			Rev. / Date of issue / Sheet: 1/1

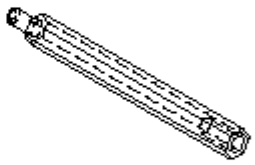




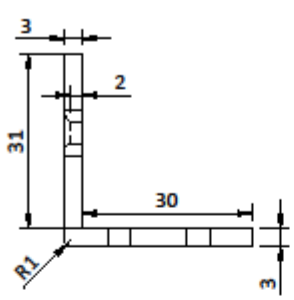
Dept.	Technical reference	Created by	Approved by
		Joel Rodelo 29/10/2022	
		Document type	Document status
		Title: Base_III_Robot_Movil	DWG No.
			Rev. / Date of issue / Sheet: 1/1

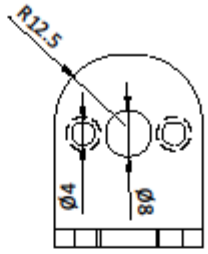


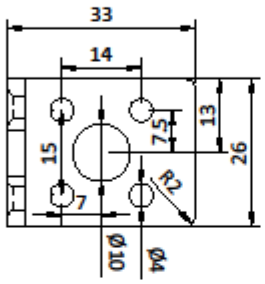


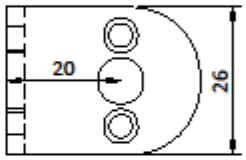


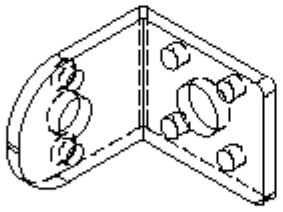
Dept.	Technical reference	Created by	Approved by
		Joel Rodelo 29/10/2022	
		Document type	Document status
		Title: Soporte_Base_II	DWG No.
			Rev. / Date of issue / Sheet: 1/1

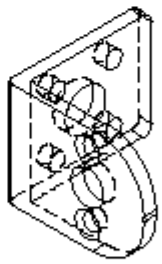




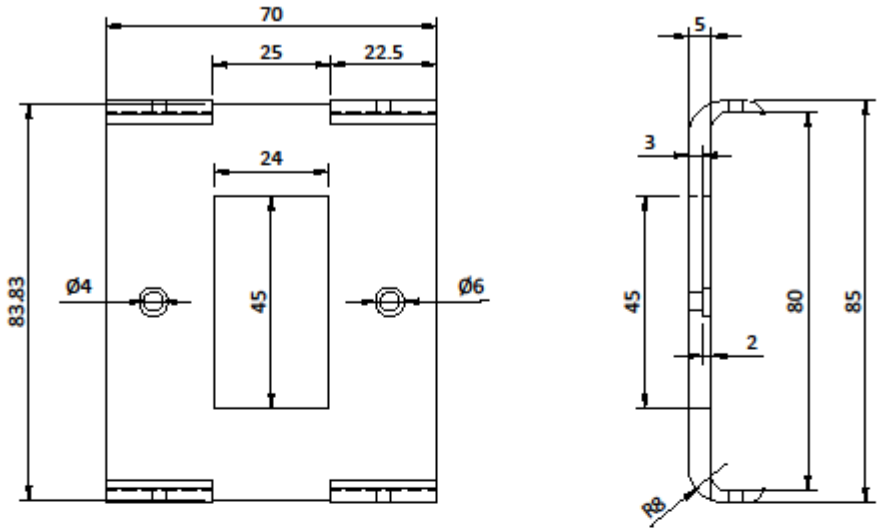


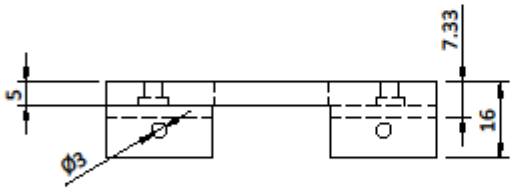


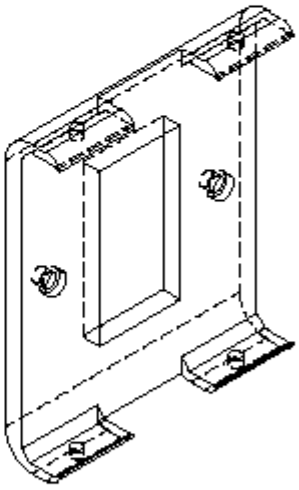




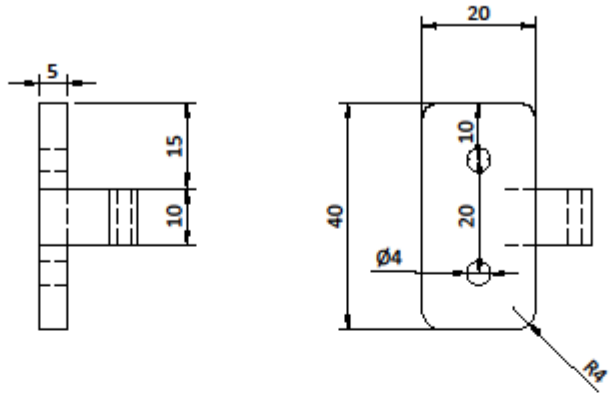
Dept.	Technical reference	Created by	Approved by
		Joel Rodelo 29/10/2022	
		Document type	Document status
		Title: Base_Motores	DWG No.
			Rev. / Date of issue / Sheet: 1/1

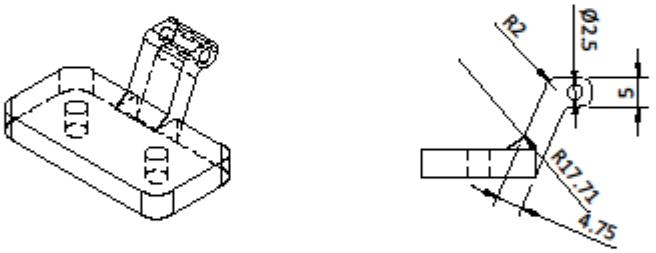




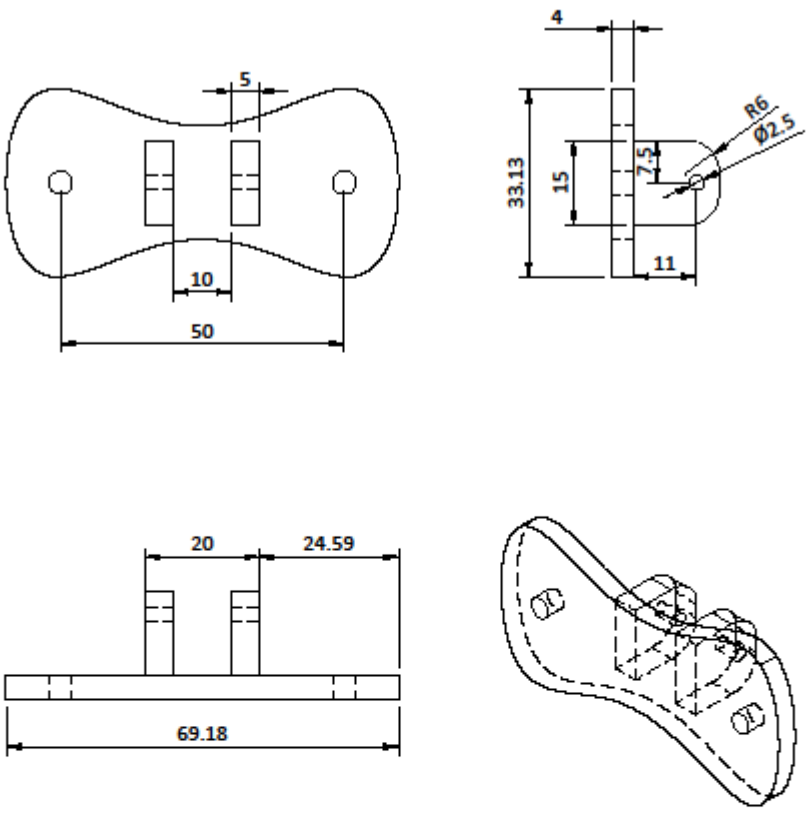


Dept.	Technical reference	Created by	Approved by
		Joel Rodelo 29/10/2022	
		Document type	Document status
		Title: Base_Telefono	DWG No.
			Rev. / Date of issue / Sheet: 1/1

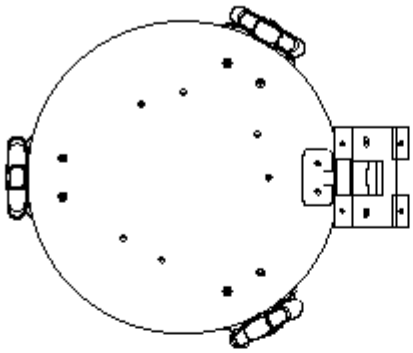


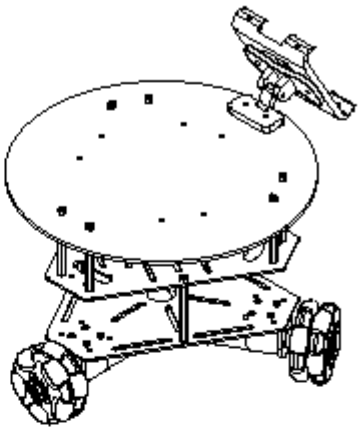


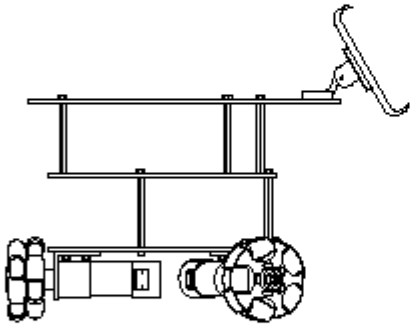
Dept.	Technical reference	Created by	Approved by
		Joel Rodelo 29/10/2022	
		Document type	Document status
		Title: Soporte_Telefono	DWG No.
			Rev. / Date of issue / Sheet: 1/1

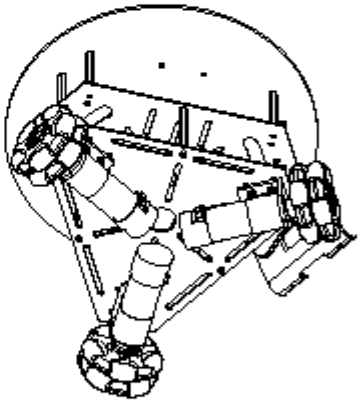


Dept.	Technical reference	Created by	Approved by
		Joel Rodelo 29/10/2022	
		Document type	Document status
		Title: soporte_II	DWG No.
			Rev. / Date of issue / Sheet: 1/1



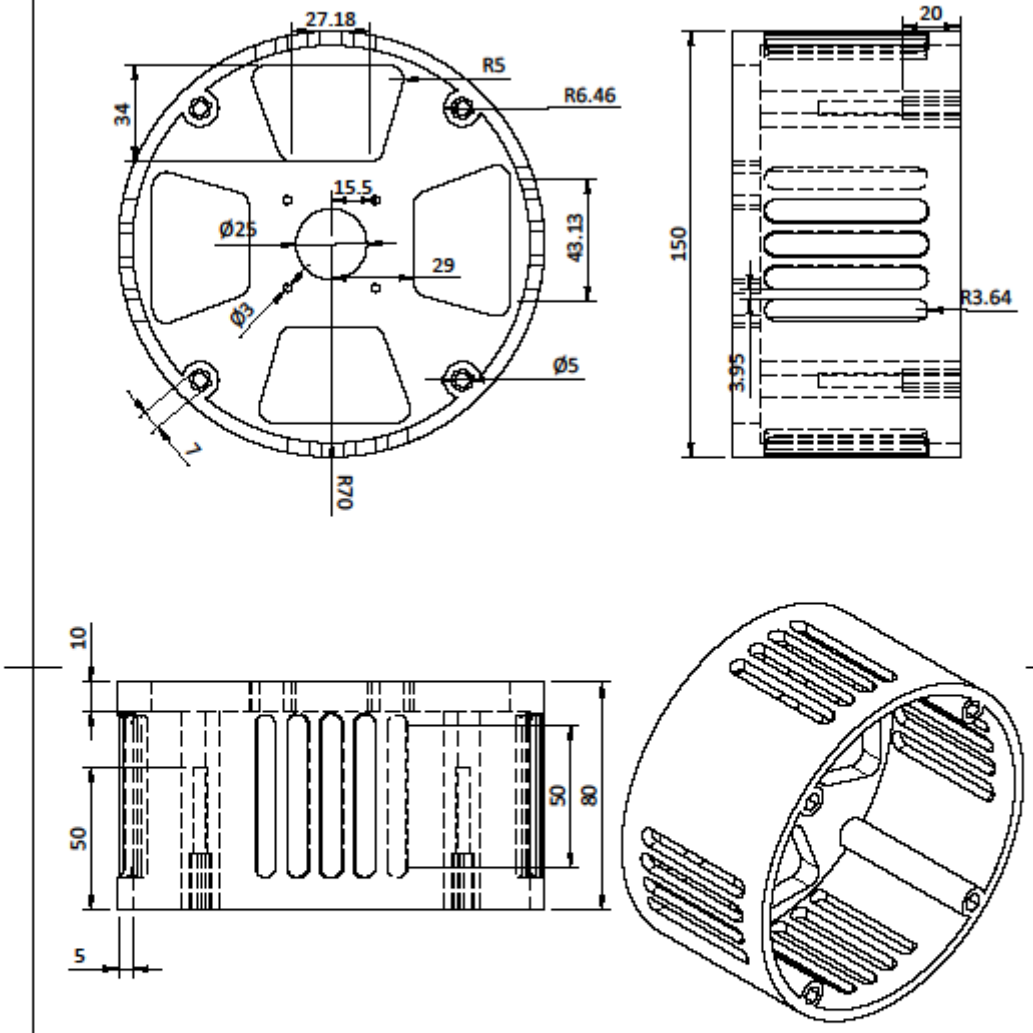




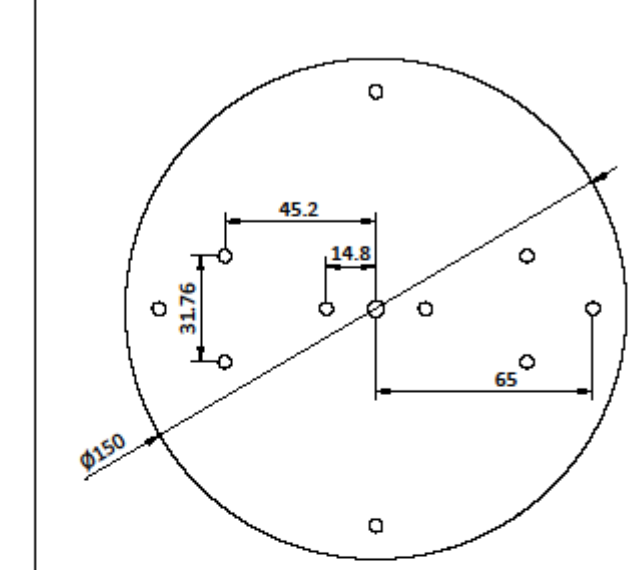


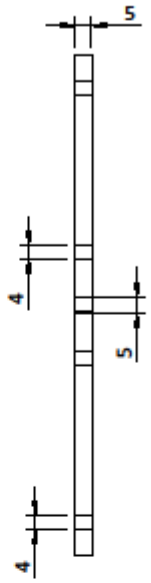
Dept.	Technical reference	Created by Joel Rodelo 2/11/2022	Approved by		
		Document type	Document status		
		Title Ensamble	DWG No.		
			Rev.	Date of issue	Sheet 1/1

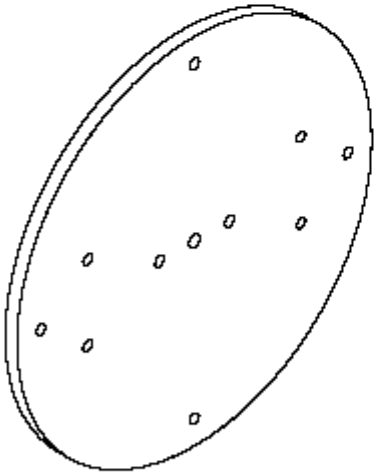
PLANOS ROBOT ANTROPOMÓRFICO



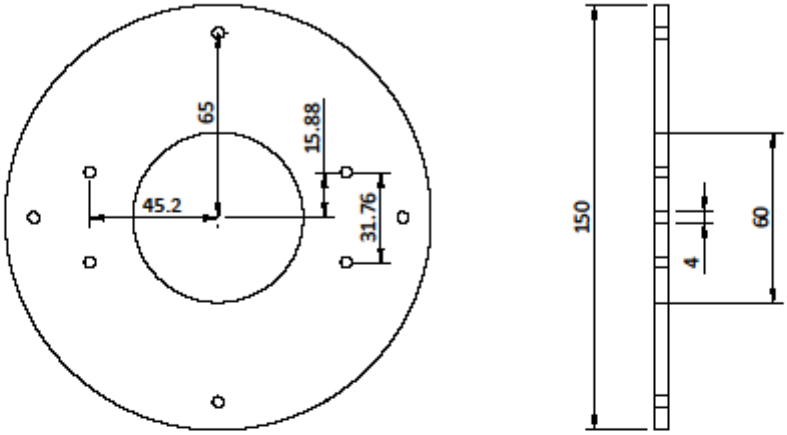
Dept.	Technical reference	Created by	Approved by
		marco arredondo 5/06/2022	
		Document type	Document status
		Title: base v7	DWG No.
			Rev. / Date of issue / Sheet: 1/1

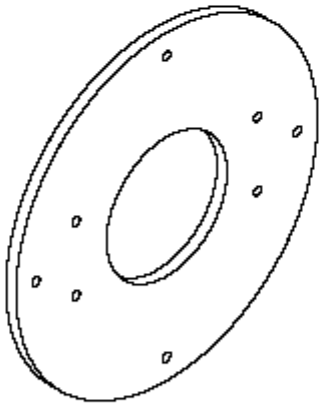




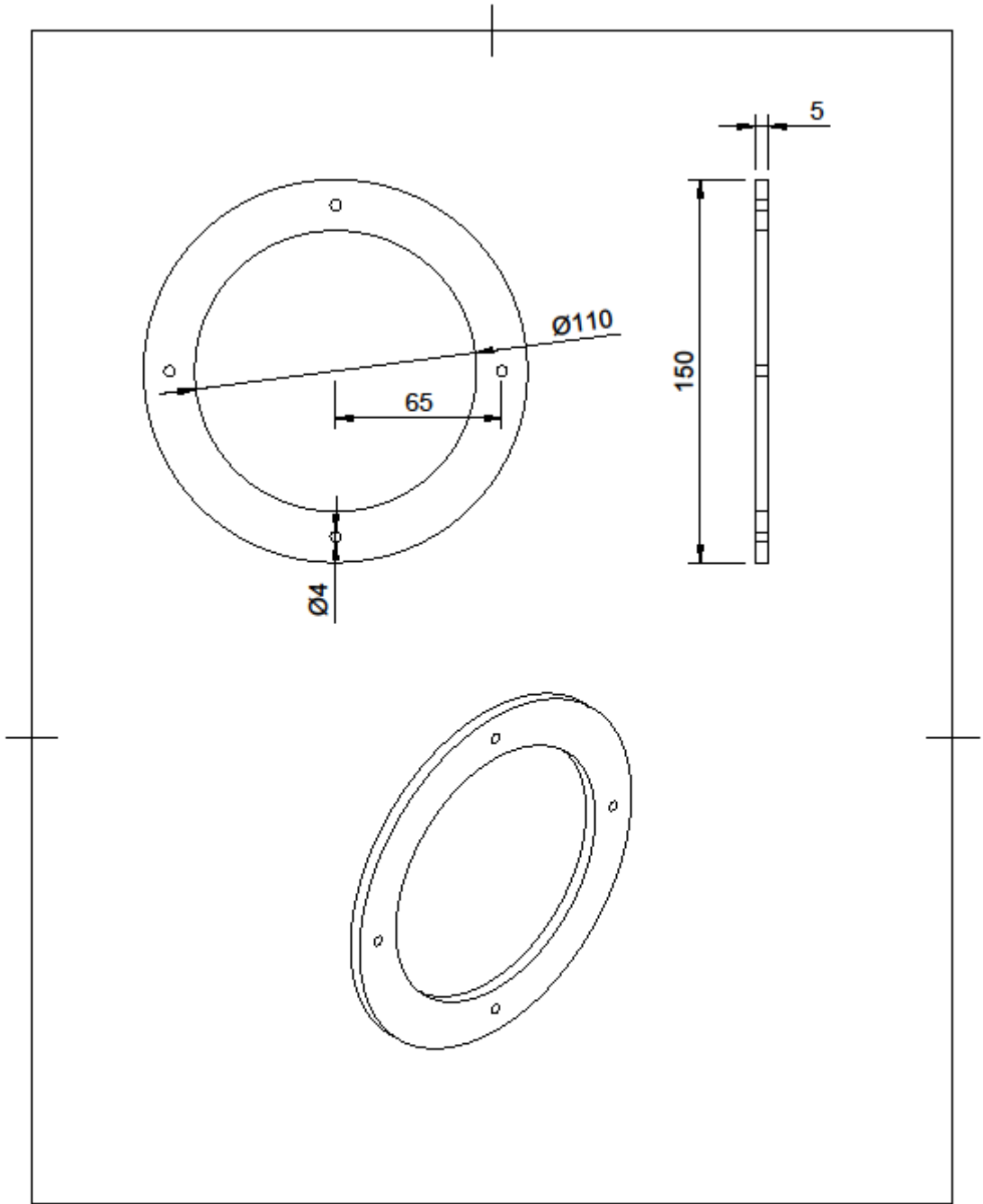


Dept.	Technical reference	Created by	Approved by
		marco arredondo 6/06/2022	
		Document type	Document status
		Title planta_bas v1	DWG No.
			Rev. / Date of issue / Sheet: 1/1

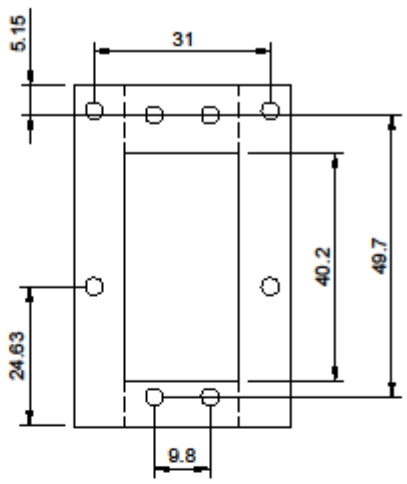


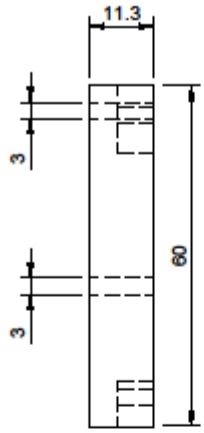


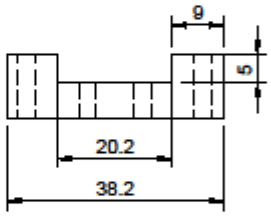
Dept.	Technical reference	Created by	Approved by
		marco arredondo 6/06/2022	
		Document type	Document status
		Title planta_bas_3 v1	DWG No.
			Rev. / Date of issue / Sheet: 1/1

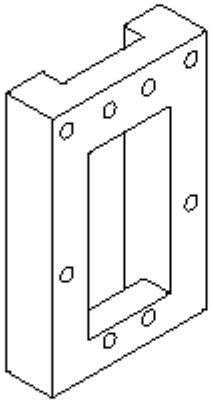


Dept.	Technical reference	Created by	Approved by
		marco arredondo 29/10/2022	
		Document type	Document status
		Title: planta_bas_2 v1	DWG No.
			Rev. / Date of issue / Sheet: 1/1

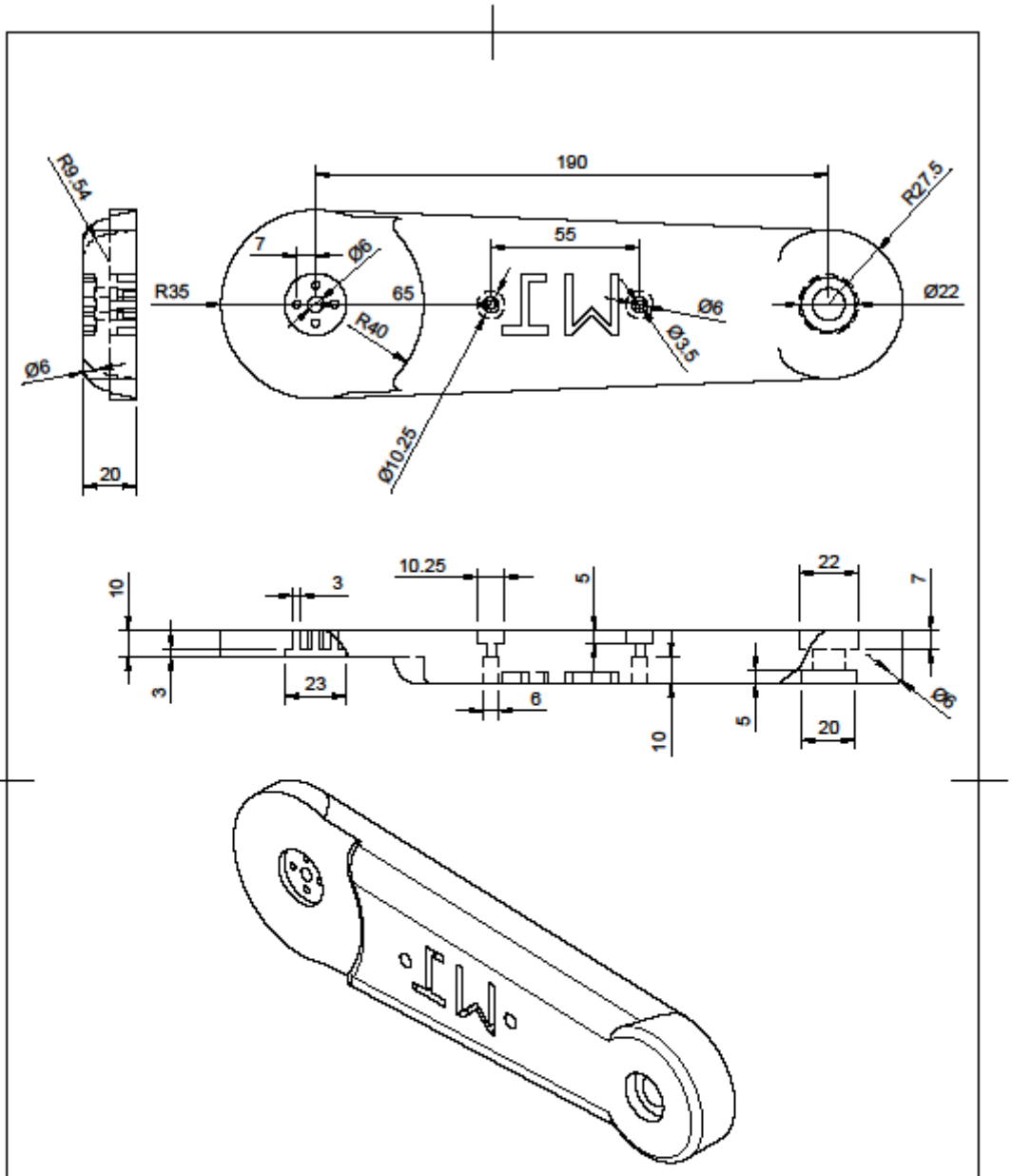




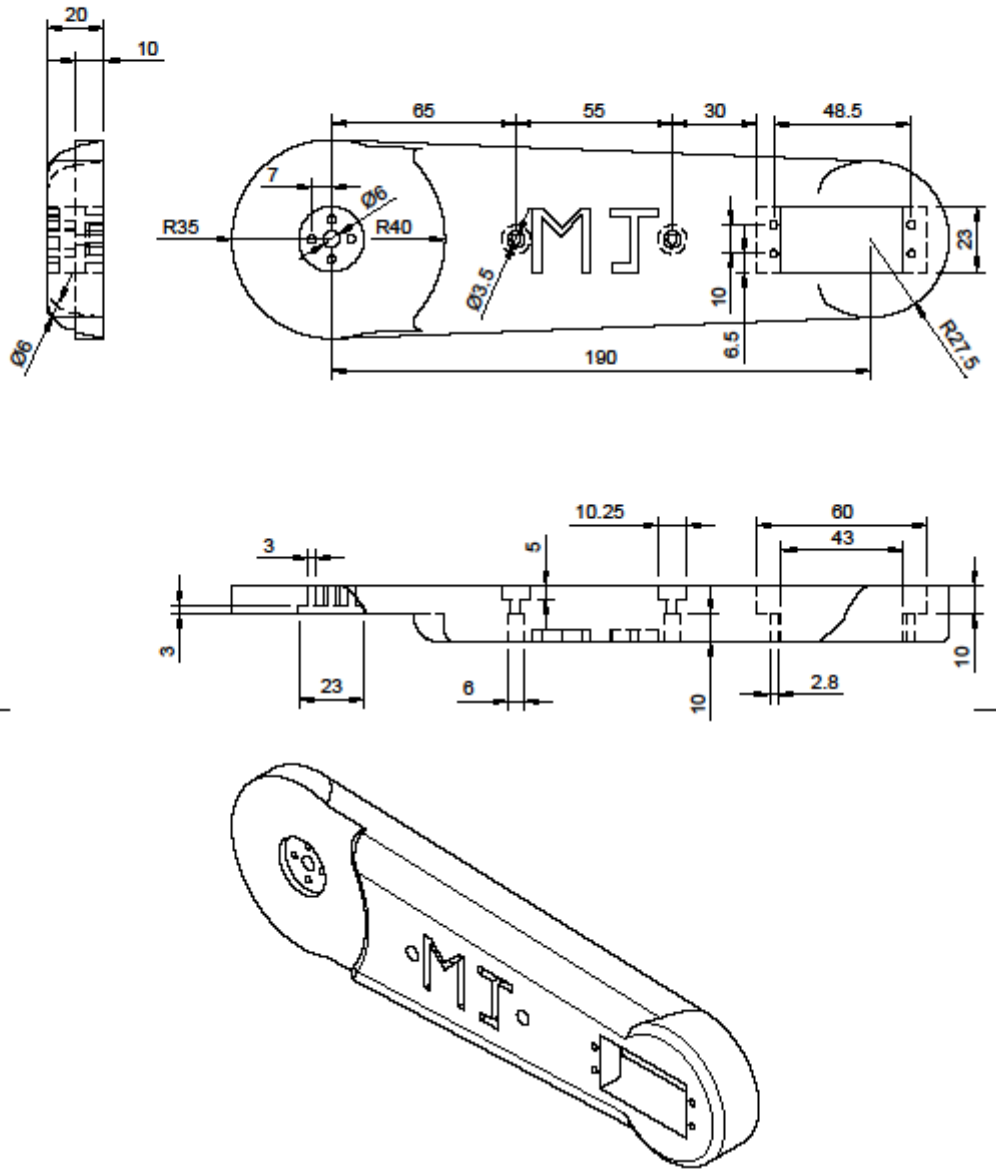




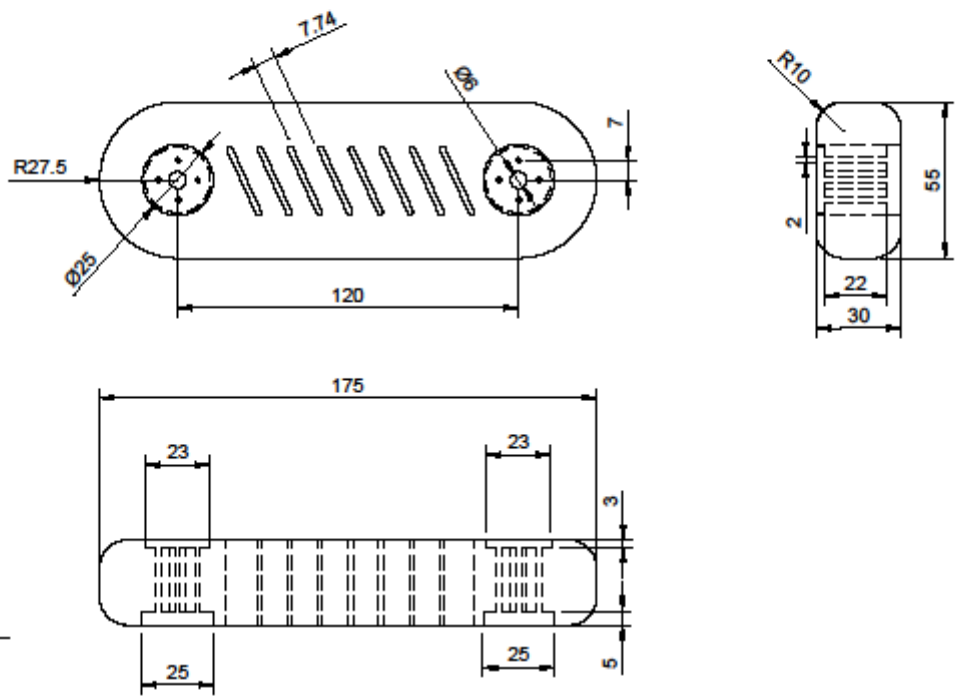
Dept.	Technical reference	Created by	Approved by
		marco arredondo 30/10/2022	
		Document type	Document status
		Title: base_servo	DWG No.
			Rev. / Date of issue / Sheet: 1/1

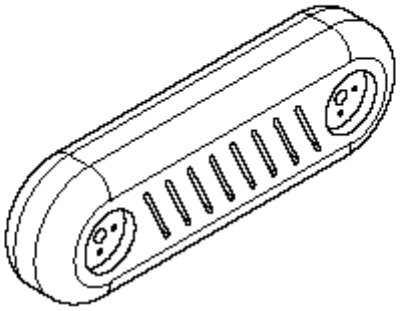


Dept.	Technical reference	Created by	Approved by
		marco arredondo 6/06/2022	
		Document type	Document status
		Title	DWG No.
		eslabon_1_2 v2	
		Rev. / Date of issue	Sheet
			1/1

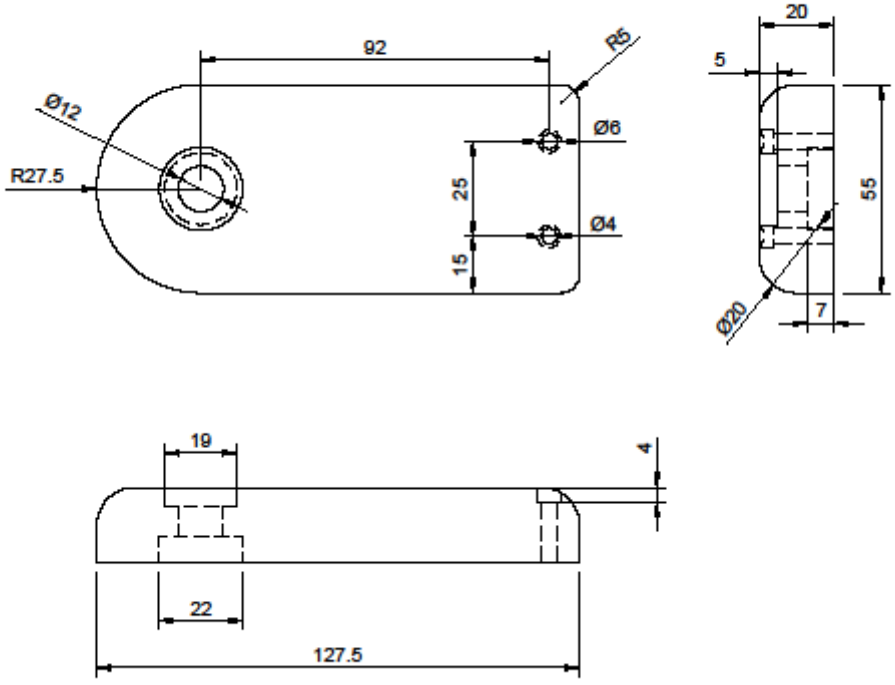


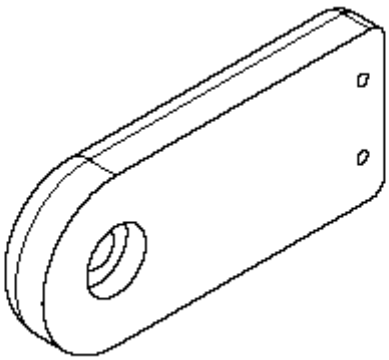
Dept.	Technical reference	Created by	Approved by
		marco arredondo 6/06/2022	
		Document type	Document status
		Title: eslabon_1 v1	DWG No.
			Rev. / Date of issue / Sheet: 1/1



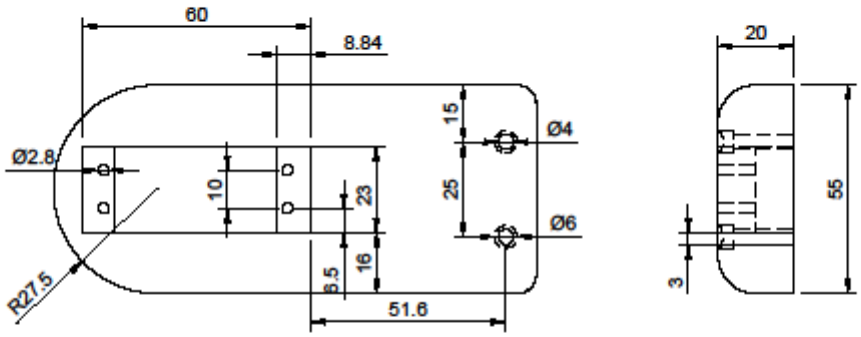


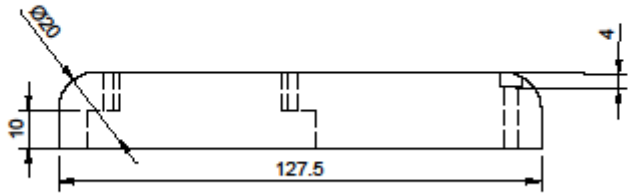
Dept.	Technical reference	Created by	Approved by
		marco arredondo 5/06/2022	
		Document type	Document status
		Title: eslabon_2 v1	DWG No.
			Rev. / Date of issue / Sheet: 1/1

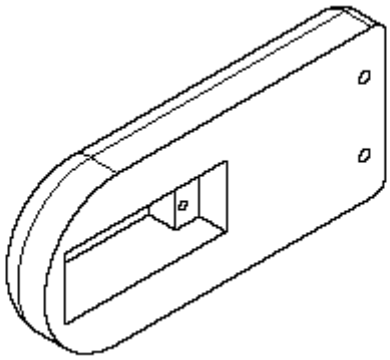




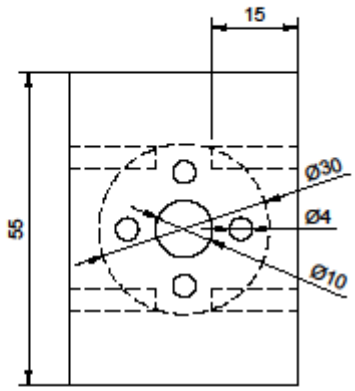
Dept.	Technical reference	Created by	Approved by
		marco arredondo 5/06/2022	
		Document type	Document status
		Title	DWG No.
		eslabon_3_2 v1	
		Rev. / Date of issue	Sheet
			1/1

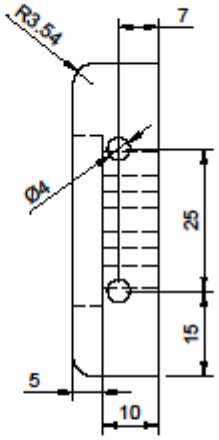


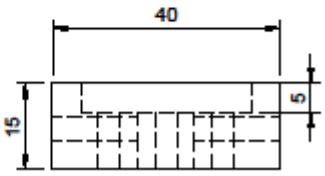


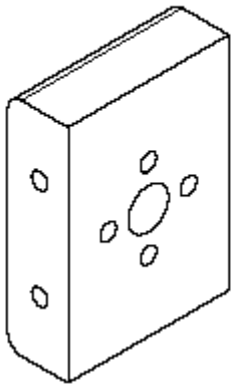


Dept.	Technical reference	Created by	Approved by
		marco arredondo 5/06/2022	
		Document type	Document status
		Title: eslabon_3 v4	DWG No.
			Rev. / Date of issue / Sheet: 1/1

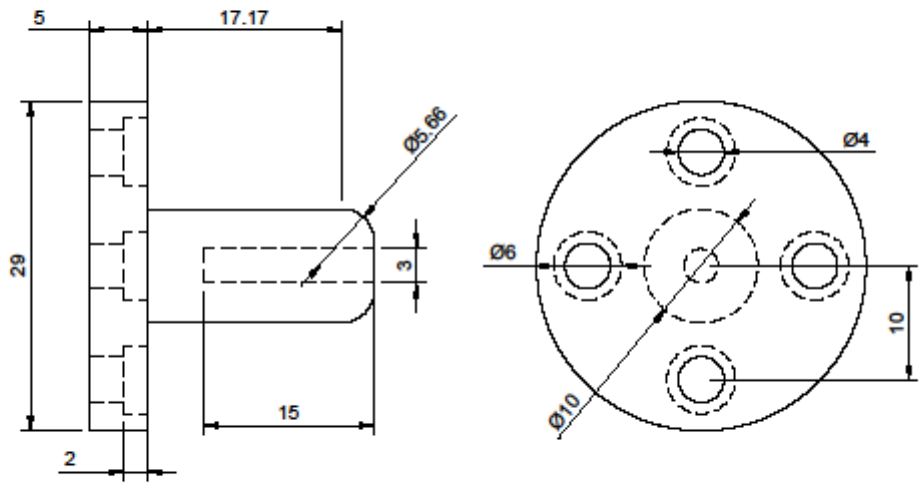


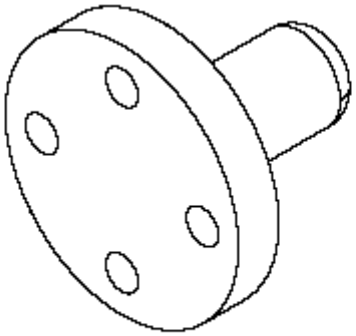




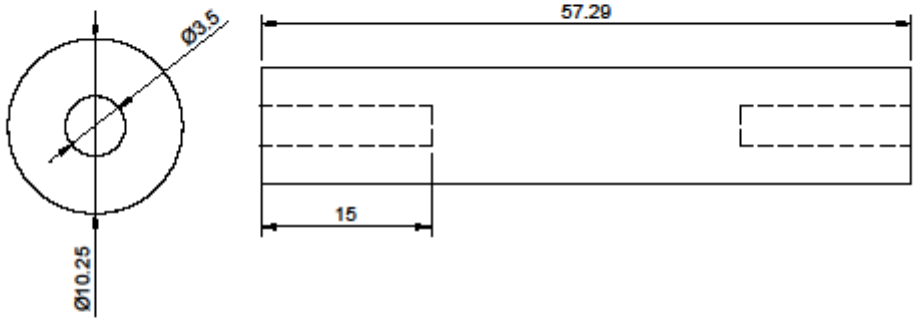


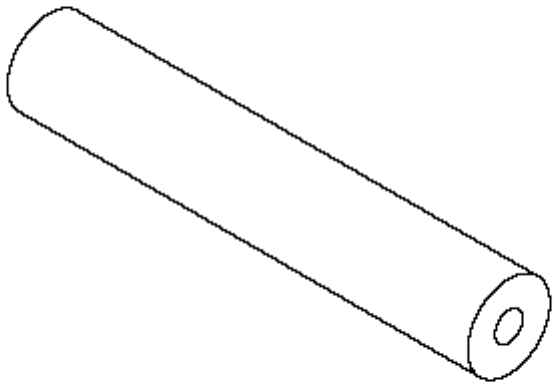
Dept.	Technical reference	Created by	Approved by
		marco arredondo 5/06/2022	
		Document type	Document status
		Title: base_efec_fin v2	DWG No.
			Rev. / Date of issue / Sheet: 1/1



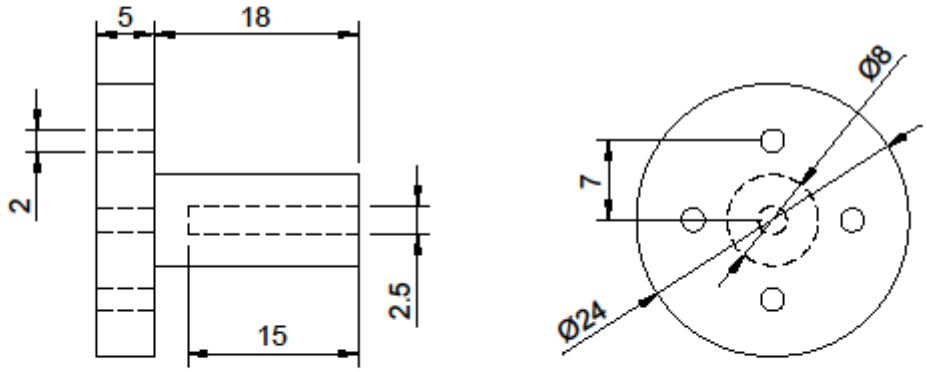


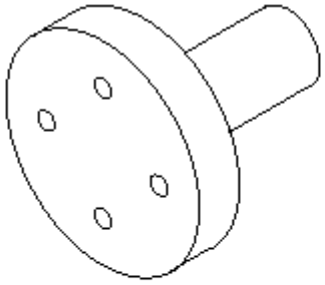
Dept.	Technical reference	Created by	Approved by
		marco arredondo 5/06/2022	
		Document type	Document status
		Title	DWG No.
		efector_final v1	
		Rev. / Date of issue	Sheet
			1/1



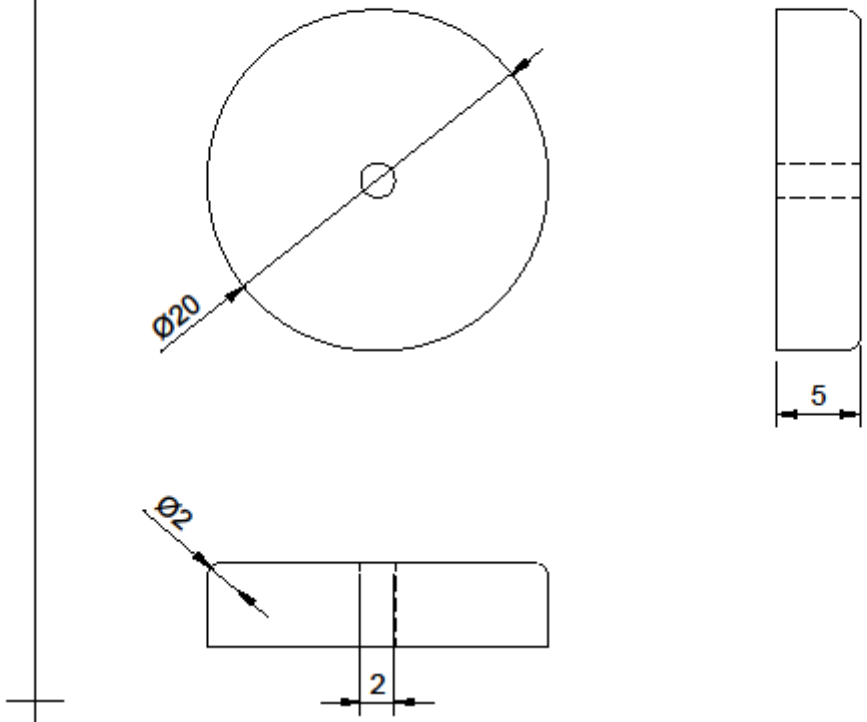


Dept.	Technical reference	Created by	Approved by
		marco arredondo 6/06/2022	
		Document type	Document status
		Title	DWG No.
		Pasador_articulacion_I	
			Rev. / Date of issue / Sheet: 1/1

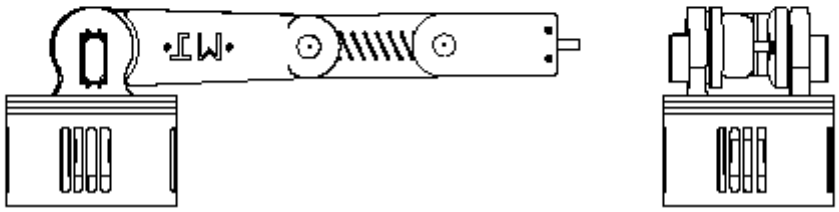


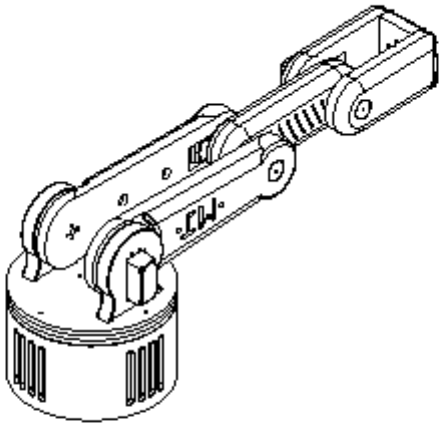


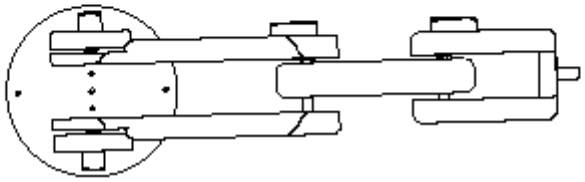
Dept.	Technical reference	Created by	Approved by
		marco arredondo 29/10/2022	
		Document type	Document status
		Title	DWG No.
		eje v1	
		Rev. / Date of issue	Sheet
			1/1



Dept.	Technical reference	Created by	Approved by
		marco arredondo 29/10/2022	
		Document type	Document status
		Title: tapa	DWG No.
			Rev. / Date of issue / Sheet: 1/1







Dept.	Technical reference	Created by marco arredondo 2/11/2022	Approved by	
		Document type	Document status	
		Title Ensamble	DWG No.	
			Rev.	Date of issue / Sheet 1/1

CÓDIGOS ROBOT MÓVIL OMNIDIRECCIONAL

Algoritmo de identificación Python

```
from pyArduino import*
import matplotlib.pyplot as plt
import time

tiempoMuestreo = 0.1                                    # Tiempo de muestreo
tiempoSimulacion  = 10                                  # Tiempo de simulacion
tiempo = np.arange(0,tiempoSimulacion+tiempoMuestreo,tiempoMuestreo)   # Array de tiempo
numeroMuestras = len(tiempo)                                           # Numero de muestras


# Comunicacion Serial
port = 'COM3'                                           # COM Arduino
baudRate = 9600                                         # Baudios

arduino = serialArduino(port,baudRate,1)                # Objeto serial

arduino.readSerialStart()                               # Inicia lectura de datos


# Señales
variableProceso = np.zeros(numeroMuestras)              # Variable de proceso
variableControl = np.zeros(numeroMuestras)              # Variable de control

# Loop
for k in range(numeroMuestras):

     tiempoInicio = time.time()                         # Tiempo actual

     # Escalon

     if k*tiempoMuestreo > 3:                  # Escalon a los 3 segundos
         variableControl[k] = 40               # Valor escalon del 0 al 100% (40%)
     else:
         variableControl[k] = 0

     arduino.sendData([variableControl[k]])  # Enviar VC (debe ser una lista)

     variableProceso[k] = arduino.rawData[0]   # Recibir variable proceso


     tiempoTranscurrido = time.time() - tiempoInicio   # Tiempo transcurrido

     time.sleep(tiempoMuestreo-tiempoTranscurrido)

arduino.sendData([0])                                   # Detener motor
arduino.close()                                         # Cerrar puerto serial

# Guardar señales

with open('firstResponse.npy','wb') as j:
    np.save(j,variableControl)
    np.save(j,variableProceso)
    np.save(j,tiempo)
    np.save(j,tiempoMuestreo)
```

```
# Mostrar figuras
plt.plot(tiempo,variableProceso,label='variable Proceso')
plt.plot(tiempo,variableControl,label='variable Control')
plt.legend(loc='upper left')
plt.show()
```

Función Cinemática robot móvil omnidireccional

```
from pyRobotics import *
from pyArduino import *
import matplotlib.pyplot as plt


def cinematica_robot_movil(PuntoDeseadoX, PuntoDeseadoY, numeroMuestras, alfaDeseado,
alfaPuntoDeseado,tiempoMuestreo, tiempo, tiempoSimulacion, arduino):

#  PARAMETROS DEL ROBOT

    a = 0.07                # Distancia hacia el punto de control en metros [m]

# CONDICIONES INICIALES
# Asignar memoria
    posInicialx1 = np.zeros(numeroMuestras+1)  # Posicion del robot
    posInicialy1 = np.zeros(numeroMuestras+1)
    alfa = np.zeros(numeroMuestras+1)

    posInicialx1[0] = 0        # Posicion inicial en el eje x en metros [m]
    posInicialy1[0] = 0        # Posicion inicial en el eje y en metros [m]
    alfa[0] = 0*(np.pi/180)    # Orientacion inicial en radianes [rad]


# PUNTO DE CONTROL


# Asignar memoria
    hx = np.zeros(numeroMuestras+1)
    hy = np.zeros(numeroMuestras+1)

    hx[0] = posInicialx1[0]+a*np.cos(alfa[0])
    hy[0] = posInicialy1[0]+a*np.sin(alfa[0])


# CAMINO DESEADO
    VelMaxDeseada = 0.15       # Velocidad maxima deseada en metros/segundo [m/s]

    cantidadPuntos = len(PuntoDeseadoX)   # Calcular la cantidad puntos en camino

    beta = np.zeros(cantidadPuntos)

    for n in range(cantidadPuntos):  # Calculo del angulo beta para todos los puntos
        if n == 0:
            beta[n] = np.arctan2(PuntoDeseadoY[n+1]-PuntoDeseadoY[n],PuntoDeseadoX[n+1]-
PuntoDeseadoX[n])
        else:
            beta[n] = np.arctan2(PuntoDeseadoY[n]-PuntoDeseadoY[n-1],PuntoDeseadoX[n]-
PuntoDeseadoX[n-1])


# VELOCIDADES DE REFERENCIA
    ufReferencia = np.zeros(numeroMuestras)  # Velocidad lineal en [m/s] eje x
    ulReferencia = np.zeros(numeroMuestras)  # Velocidad lineal en [m/s] eje y
    wReferencia = np.zeros(numeroMuestras)   # Velocidad angular en [rad/s]
```

```
# VELOCIDADES MEDIDAS
    ufMedida = np.zeros(numeroMuestras)      # Velocidad lineal en [m/s] eje x
    ulMedida = np.zeros(numeroMuestras)      # Velocidad lineal en [m/s] eje y
    wMedida = np.zeros(numeroMuestras)       # Velocidad angular en [rad/s]

# ERRORES
    errorejex = np.zeros(numeroMuestras)     # Error en el eje x en metros [m]
    errorejey = np.zeros(numeroMuestras)     # Error en el eje y en metros [m]
    errorenalfa = np.zeros(numeroMuestras)   # Error de orientación en radianes [rad]

# BUCLE
    for i in range(numeroMuestras):

        start_time = time.time()             # Tiempo actual
    # CONTROLADOR

    # Punto mas cercano
        minimo = 1000
        for n in range(cantidadPuntos):
            distancia = np.sqrt((PuntoDeseadoX[n]-hx[i])**2+(PuntoDeseadoY[n]-hy[i])**2)

            if distancia<minimo:
                minimo = distancia
                pos = n
    # Error

        errorejex[i] = PuntoDeseadoX[pos]-hx[i]
        errorejey[i] = PuntoDeseadoY[pos]-hy[i]
        errorenalfa[i] = alfaDeseado[i] - alfa[i]

        error = np.array([[errorejex[i]],[errorejey[i]],[errorenalfa[i]]])

    # Matriz Jacobiana
        J = np.array([[ np.cos(alfa[i]), -np.sin(alfa[i]), -a*np.sin(alfa[i])],
                      [ np.sin(alfa[i]),  np.cos(alfa[i]),  a*np.cos(alfa[i])],
                      [               0,                0,                1]])


    # Parametros de control
        k = 0.8*np.array([[1, 0, 0],
                          [0, 1, 0],
                          [0,  0, 1]])

    # Velocidad deseada
        pxdp = VelMaxDeseada*np.cos(beta[pos])
        pydp = VelMaxDeseada*np.sin(beta[pos])

        pdp = np.array([[pxdp],[pydp],[alfaPuntoDeseado[i]]])

    # Ley de control
        controlPuntoRef = np.linalg.pinv(J)@(pdp+k@error)
```

```
    # APLICAR ACCIONES DE CONTROL

        ufReferencia[i] = controlPuntoRef[0][0]
        ulReferencia[i] = controlPuntoRef[1][0]
        wReferencia[i] = controlPuntoRef[2][0]

        arduino.sendData([ufReferencia[i],ulReferencia[i],wReferencia[i]])

        ufMedida[i] = arduino.rawData[0]
        ulMedida[i] = arduino.rawData[1]
        wMedida[i] = arduino.rawData[2]

    # Integral numerica
        alfa[i+1] = alfa[i]+tiempoMuestreo*wMedida[i]

    # Modelo cinemático robot movil
        Puntox1 = ufMedida[i]*np.cos(alfa[i+1])-ulMedida[i]*np.sin(alfa[i+1])
        Puntoy1 = ufMedida[i]*np.sin(alfa[i+1])+ulMedida[i]*np.cos(alfa[i+1])

    # Integral númerica por método de euler
        posInicialx1[i+1] = posInicialx1[i] + tiempoMuestreo*Puntox1
        posInicialy1[i+1] = posInicialy1[i] + tiempoMuestreo*Puntoy1

        hx[i+1] = posInicialx1[i+1]+a*np.cos(alfa[i+1])
        hy[i+1] = posInicialy1[i+1]+a*np.sin(alfa[i+1])

        elapsed_time = time.time() - start_time   # Tiempo transcurrido

        time.sleep(tiempoMuestreo-elapsed_time)   # Esperar completar tiempo muestreo

# COMUNICACION SERIAL

    arduino.sendData([0,0,0])                     # Detener robot

# SIMULACION VIRTUAL
# Cargar componentes del robot
    path = "stl"
    color = ['yellow','white']
    omni3 = robotics(path,color)

# Configurar escena
    xmin = -1
    xmax = 1
    ymin = -1
    ymax = 1
    zmin = 0
    zmax = 1
    bounds = [xmin, xmax, ymin, ymax, zmin, zmax]
    omni3.configureScene(bounds)
```

```
# Mostrar graficas
    omni3.plotDesiredTrajectory(PuntoDeseadoX,PuntoDeseadoY)
    omni3.initTrajectory(hx,hy)

    escala = 0.003
    omni3.initRobot(posInicialx1,posInicialy1,alfa,escala)

    omni3.startSimulation(1,tiempoMuestreo)


# Graficas
    # Errores
    fig = plt.figure()

    plt.subplot(211)
    plt.plot(tiempo,errorejex,'b',linewidth = 2, label='errorejex')
    plt.plot(tiempo,errorejey,'r',linewidth = 2,  label='errorejey')
    plt.legend(loc='upper right')
    plt.xlabel('Tiempo [s]')
    plt.ylabel('Error [m]')
    plt.grid()

    plt.subplot(212)
    plt.plot(tiempo,errorenalfa,'g',linewidth = 2, label='errorenalfa')
    plt.legend(loc='upper right')
    plt.xlabel('Tiempo [s]')
    plt.ylabel('Error [rad]')
    plt.grid()

# Acciones de control
    fig = plt.figure()
    plt.subplot(211)
    plt.plot(tiempo,ufReferencia,'g',linewidth = 2, label='ufReferencia')
    plt.plot(tiempo,ufMedida,'r',linewidth = 2, label='ufMedida')
    plt.legend(loc='upper right')
    plt.xlabel('Tiempo [s]')
    plt.ylabel('Velocidad [m/s]')
    plt.grid()

    plt.subplot(212)
    plt.plot(tiempo,ulReferencia,'g',linewidth = 2, label='ulReferencia')
    plt.plot(tiempo,ulMedida,'r',linewidth = 2, label='ulMedida')
    plt.legend(loc='upper right')
    plt.xlabel('Tiempo [s]')
    plt.ylabel('Velocidad [m/s]')
    plt.grid()

    fig = plt.figure()
    plt.plot(tiempo,wReferencia,'g',linewidth = 2, label='wReferencia')
    plt.plot(tiempo,wMedida,'r',linewidth = 2, label='wReferencia')
    plt.legend(loc='upper right')
    plt.xlabel('Tiempo [s]')
    plt.ylabel('Velocidad [rad/s]')

    plt.grid()
    plt.show()
```

CÓDIGO DE INTERFAZ DE USUARIO

```
import sys
from turtle import width
from typing_extensions import Self
from PyQt5.QtWidgets import QApplication, QMainWindow, QGraphicsDropShadowEffect
from PyQt5.QtCore import QPropertyAnimation, QEasingCurve
from PyQt5.QtGui import QColor
from PyQt5 import QtCore, QtWidgets
from PyQt5.uic import loadUi
from PyQt5.QtSerialPort import QSerialPortInfo
from qt_material import apply_stylesheet
from mesa_I import *
from mesa_II import *
from mesa_III import *
from return_mesa_I import *
import matplotlib.pyplot as plt


class VentanaPrincipal(QMainWindow):
    def __init__(self):
        super(VentanaPrincipal,self).__init__()
        loadUi('interfaz_moderna.ui',self)  # Cargar interfaz hecha en qtdesigner

        self.Boton_Menu_Uno.clicked.connect(self.mover_menu)
        self.Boton_Menu_Dos.clicked.connect(self.mover_menu)

        # Ocultar los botones
        self.Boton_Restaurar.hide()
        self.Boton_Menu_Dos.hide()

        # Sombre de los widgets
        self.sombra_frame(self.frame_3)
        self.sombra_frame(self.comunicacion)
        self.sombra_frame(self.menu_lateral)
        self.sombra_frame(self.frame_superior)
        self.sombra_frame(self.toolBox)
        self.sombra_frame(self.Boton_Uno)
        self.sombra_frame(self.Boton_Dos)
        self.sombra_frame(self.Boton_Tres)
        self.sombra_frame(self.boton_conectar)
        self.sombra_frame(self.boton_desconectar)
        self.sombra_frame(self.boton_actualizar)
        #self.sombra_frame(self.boton_return_robot)
        self.sombra_frame(self.return_uno)
        self.sombra_frame(self.return_dos)
        self.sombra_frame(self.return_tres)

        # Control barra de titulos
        self.Boton_Minimizar.clicked.connect(self.control_boton_minimizar)
        self.Boton_Restaurar.clicked.connect(self.control_boton_normal)
        self.Boton_Maximizar.clicked.connect(self.control_boton_maximizar)
        self.Boton_Cerrar.clicked.connect(lambda: self.close())

        # Elimimar barra y e titulo - opacidad
        self.setWindowFlag(QtCore.Qt.FramelessWindowHint)
        self.setWindowOpacity(1)
```

```
        # SizeGrip
        self.gripSize = 10
        self.grip = QtWidgets.QSizeGrip(self)
        self.grip.resize(self.gripSize, self.gripSize)

        # Mover ventana
        self.frame_superior.mouseMoveEvent = self.mover_ventana

        # Estado de los botones al iniciar
        self.Boton_Uno.setEnabled(False)
        self.Boton_Dos.setEnabled(False)
        self.Boton_Tres.setEnabled(False)
        self.boton_desconectar.setEnabled(False)
        #self.boton_return_robot.setEnabled(False)
        self.return_uno.setEnabled(False)
        self.return_dos.setEnabled(False)
        self.return_tres.setEnabled(False)

        # Conexion arduino
        self.boton_conectar.clicked.connect(self.conectar)
        self.boton_actualizar.clicked.connect(self.puertos_disponibles)
        self.boton_desconectar.clicked.connect(self.desconectar)
        self.Boton_Uno.clicked.connect(self.mesa_uno)
        self.Boton_Dos.clicked.connect(self.mesa_dos)
        self.Boton_Tres.clicked.connect(self.mesa_tres)
        self.return_uno.clicked.connect(self.return_mesa_uno)

    def control_boton_minimizar(self):
        self.showMinimized()

    def control_boton_normal(self):
        self.showNormal()
        self.Boton_Restaurar.hide()
        self.Boton_Maximizar.show()

    def control_boton_maximizar(self):
        self.showMaximized()
        self.Boton_Maximizar.hide()
        self.Boton_Restaurar.show()

    # Creamos el metodo sombra
    def sombra_frame(self, frame):
        sombra = QGraphicsDropShadowEffect(self)
        sombra.setBlurRadius(30)
        sombra.setXOffset(8)
        sombra.setYOffset(8)
        sombra.setColor(QColor(0,0,0,255))
        frame.setGraphicsEffect(sombra)

    # SizeGrip
    def resizeEvent(self, event):
        rect = self.rect()
        self.grip.move(rect.right() - self.gripSize, rect.bottom() - self.gripSize)
```

```
# Mover Ventana
def mousePressEvent(self, event):
    self.clickPosition = event.globalPos()

def mover_ventana(self, event):
    if self.isMaximized() == False:
        if event.buttons() == QtCore.Qt.LeftButton:
            self.move(self.pos() + event.globalPos() - self.clickPosition)
            self.clickPosition = event.globalPos()
            event.accept()
    if event.globalPos().y() <= 10:
        self.showMaximized()
        self.Boton_Maximizar.hide()
        self.Boton_Restaurar.show()
    else:
        self.showNormal()
        self.Boton_Restaurar.hide()
        self.Boton_Maximizar.show()

# Método para mover el menu lateral izquierdo
def mover_menu(self):
    if True:
        width = self.menu_lateral.width()
        normal = 0
        if width == 0:
            extender = 300
            self.Boton_Menu_Dos.hide()
            self.Boton_Menu_Uno.show()

        else:
            self.Boton_Menu_Dos.show()
            self.Boton_Menu_Uno.hide()
            extender = normal
        self.animacion = QPropertyAnimation(self.menu_lateral, b"maximumWidth")
        self.animacion.setStartValue(width)
        self.animacion.setEndValue(extender)
        self.animacion.setDuration(500)
        self.animacion.setEasingCurve(QEasingCurve.OutInBack)
        self.animacion.start()

# Método para verificar lo puertos disponibles
def puertos_disponibles(self):
    self.baudrates = ['9600']

    portList =[]
    ports = QSerialPortInfo().availablePorts()

    for i in ports:
        portList.append(i.portName())

    self.puertos_com.clear()
    self.baudios.clear()
    self.puertos_com.addItems(portList)
    self.baudios.addItems(self.baudrates)
    self.baudios.setCurrentText("9600")
```

```
    # Método para iniciar la comunicación serial
    def conectar(self):
        port = self.puertos_com.currentText()
        baudRate = self.baudios.currentText()
        self.arduino = serialArduino(port,baudRate,3)
        self.arduino.readSerialStart()
        self.conectado.setText("<html><head/><body><p><span style=\"
color:#55aa00;\">Conectado</span></p></body></html>")
        self.Boton_Uno.setEnabled(True)
        self.Boton_Dos.setEnabled(True)
        self.Boton_Tres.setEnabled(True)
        self.boton_desconectar.setEnabled(True)
        #self.boton_return_robot.setEnabled(True)
        self.return_uno.setEnabled(True)
        self.return_dos.setEnabled(True)
        self.return_tres.setEnabled(True)


    # Método para terminar la comunicación serial
    def desconectar(self):
        self.arduino.close() # Cerrar puerto serial
        self.conectado.setText("<html><head/><body><p><span style=\"
color:#FF0000;\">Desconectado</span></p></body></html>")
        self.Boton_Uno.setEnabled(False)
        self.Boton_Dos.setEnabled(False)
        self.Boton_Tres.setEnabled(False)
        self.boton_desconectar.setEnabled(False)
        #self.boton_return_robot.setEnabled(False)
        self.return_uno.setEnabled(False)
        self.return_dos.setEnabled(False)
        self.return_tres.setEnabled(False)

    # Función para enviar robot a mesa uno
    def mesa_uno(self):
        PuntosX = [0,0.3,2,2]
        PuntosY = [0,0,0,-2]
        mesa_I(PuntosX, PuntosY, self.arduino)

    # Función para retornar robot a mesa uno
    def return_mesa_uno(self):
        PuntosX = [0,2,2,2]
        PuntosY = [0,0,-0.3,-2]
        return_mesa_I(PuntosX, PuntosY, self.arduino)

    # Función para enviar robot a mesa uno
    def mesa_dos(self):
        PuntosX = [0,0.6,4,4]
        PuntosY = [0,0.6,0.6,0]
        mesa_II(PuntosX, PuntosY,self.arduino)

    # Función para enviar robot a mesa uno
    def mesa_tres(self):
        PuntosX = [0,0.6,4,4]
        PuntosY = [0,0.6,0.6,0]
        mesa_III(PuntosX, PuntosY, self.arduino)

if __name__=='__main__':
    app = QApplication(sys.argv)
    apply_stylesheet(app, theme='light_blue.xml')
    mi_app = VentanaPrincipal()
    mi_app.show()
    sys.exit(app.exec_())
```

FUNCIÓN PARA ENVIAR ROBOT A MESA UNO

```
from pyRobotics import *
from pyArduino import *
import matplotlib.pyplot as plt
from cinematica_robot_movil import cinematica_robot_movil


def mesa_I(PuntosX, PuntosY, arduino):
    # TIEMPO

    tiempoSimulacion = 30                          # tiempo de simulacion
    tiempoMuestreo = 0.1                           #  tiempo de muestreo
    tiempo = np.arange(0,tiempoSimulacion+tiempoMuestreo,tiempoMuestreo)  # vector tiempo
    numeroMuestras = len(tiempo)

    # Camino de 3 linea

    Divisor = 800

    px = []
    py = []

    px.append(np.linspace(PuntosX[0],PuntosX[1],Divisor))
    py.append(np.linspace(PuntosY[0],PuntosY[1],Divisor))

    px.append(np.linspace(PuntosX[1],PuntosX[2],Divisor))
    py.append(np.linspace(PuntosY[1],PuntosY[2],Divisor))

    px.append(np.linspace(PuntosX[2],PuntosX[3],Divisor))
    py.append(np.linspace(PuntosY[2],PuntosY[3],Divisor))

    PuntoDeseadoX = np.hstack(px)
    PuntoDeseadoY = np.hstack(py)

    # VELOCIDAD ANGULAR DESEADA DEL ROBOT
    alfaDeseado = 0*(np.pi/180)*np.ones(numeroMuestras)
    alfaPuntoDeseado = np.zeros(numeroMuestras)              # Derivada de alfaDeseado

    cinematica_robot_movil(PuntoDeseadoX, PuntoDeseadoY, numeroMuestras, alfaDeseado,
alfaPuntoDeseado, tiempoMuestreo, tiempo, tiempoSimulacion, arduino)
```

FUNCIÓN PARA ENVIAR ROBOT A MESA DOS

```
from pyRobotics import *
from pyArduino import *
import matplotlib.pyplot as plt
from cinematica_robot_movil import cinematica_robot_movil


def mesa_II(PuntosX, PuntosY , arduino):
    # TIEMPO

    tiempoSimulacion = 35                                  # tiempo de simulacion
    tiempoMuestreo = 0.1                                   #  tiempo de muestreo
    tiempo = np.arange(0,tiempoSimulacion+tiempoMuestreo,tiempoMuestreo) # vector tiempo
    numeroMuestras = len(tiempo)

    # Camino de 3 linea
    Divisor = 800

    px = []
    py = []

    px.append(np.linspace(PuntosX[0],PuntosX[1],Divisor))
    py.append(np.linspace(PuntosY[0],PuntosY[1],Divisor))

    px.append(np.linspace(PuntosX[1],PuntosX[2],Divisor))
    py.append(np.linspace(PuntosY[1],PuntosY[2],Divisor))

    px.append(np.linspace(PuntosX[2],PuntosX[3],Divisor))
    py.append(np.linspace(PuntosY[2],PuntosY[3],Divisor))

    PuntoDeseadoX = np.hstack(px)
    PuntoDeseadoY = np.hstack(py)

    # VELOCIDAD ANGULAR DESEADA DEL ROBOT
    alfaDeseado = 120*(np.pi/180)*np.ones(numeroMuestras)
    alfaPuntoDeseado = np.zeros(numeroMuestras)                # Derivada de alfaDeseado

    cinematica_robot_movil(PuntoDeseadoX, PuntoDeseadoY, numeroMuestras, alfaDeseado,
alfaPuntoDeseado, tiempoMuestreo, tiempo, tiempoSimulacion, arduino)
```

FUNCIÓN PARA ENVIAR ROBOT A MESA TRES

```
from pyRobotics import *
from pyArduino import *
import matplotlib.pyplot as plt
from cinematica_robot_movil import cinematica_robot_movil


def mesa_III(PuntosX, PuntosY, arduino):
    # TIEMPO

    tiempoSimulacion = 35                                       # tiempo de simulacion
    tiempoMuestreo = 0.1                                        #  tiempo de muestreo
    tiempo = np.arange(0,tiempoSimulacion+tiempoMuestreo,tiempoMuestreo) # vector tiempo
    numeroMuestras = len(tiempo)

    # Camino de 3 linea
    Divisor = 800

    px = []
    py = []

    px.append(np.linspace(PuntosX[0],PuntosX[1],Divisor))
    py.append(np.linspace(PuntosY[0],PuntosY[1],Divisor))

    px.append(np.linspace(PuntosX[1],PuntosX[2],Divisor))
    py.append(np.linspace(PuntosY[1],PuntosY[2],Divisor))

    px.append(np.linspace(PuntosX[2],PuntosX[3],Divisor))
    py.append(np.linspace(PuntosY[2],PuntosY[3],Divisor))

    PuntoDeseadoX = np.hstack(px)
    PuntoDeseadoY = np.hstack(py)

    # VELOCIDAD ANGULAR DESEADA DEL ROBOT
    alfaDeseado = 120*(np.pi/180)*np.ones(numeroMuestras)
    alfaPuntoDeseado = np.zeros(numeroMuestras)             # Derivada de alfaDeseado

    cinematica_robot_movil(PuntoDeseadoX, PuntoDeseadoY, numeroMuestras, alfaDeseado,
alfaPuntoDeseado, tiempoMuestreo, tiempo, tiempoSimulacion, arduino)
```

FUNCIÓN PARA ESTABLECER COMUNCACIÓN ENTRE PYTHON Y ARDUINO

```
from threading import Thread
import serial
import time
import sys
import numpy as np


class serialArduino:
    def __init__(self,port,baud=9600,sizeData=1):
        self.port = port
        self.baud = baud
        self.sizeData = sizeData

        self.isReceiving = False
        self.isRun     = True
        self.thread    = None

        self.rawData = [None]*self.sizeData
        print('Intentando conectarse a: ' + str(self.port))
        try:
            self.serialConnection = serial.Serial(self.port,self.baud)
            print('Conectado a ' + str(self.port))
        except:
            sys.exit("No se pudo conectar a " + str(self.port))

    def readSerialStart(self):
        if self.thread == None:
            self.thread = Thread(target=self.backgroundThread)
            self.thread.start()
            while self.isReceiving != True:
                print("Comenzando a recibir datos ")
                time.sleep(0.1)
            print("Recibiendo datos")

    def backgroundThread(self):
        time.sleep(1.0)
        self.serialConnection.flushInput()
        while (self.isRun):
            for k in range(self.sizeData):
                try:
                    self.rawData[k] = float(self.serialConnection.readline().strip())
                except:
                    sys.exit("Error data receive")
            self.isReceiving = True

    def sendData(self,dataToSend,separator=','):
        stringData = ""
        sizeSendData = len(dataToSend)
        #dataToSend = lista[]
        for k in range(sizeSendData):
            if k < sizeSendData-1:
                stringData = stringData+str(dataToSend[k])+','
            else:
                stringData = stringData+str(dataToSend[k])
        self.serialConnection.write((stringData+'\n').encode())
```

```
def close(self):
     self.isRun = False
     if self.thread == None:
          pass
     else:
          self.thread.join()
     self.serialConnection.close()
     print('Desconectado...')
```

FUNCIÓN PARA ARDUINO NANO

```
#include "PinChangeInterrupt.h"
#include "motorControl.h"

unsigned long tiempoAnterior, tiempoMuestreo = 100;

motorControl motor_1(tiempoMuestreo);
motorControl motor_2(tiempoMuestreo);
motorControl motor_3(tiempoMuestreo);

//////////////////////////// COMUNICACION SERIAL ///////////////////
String datosDeEntrada = "";     // Variable para almacenar cadena desde el puerto serial
bool datosCompletos = false;    // Variable para identificar si datos llegaron correctos
const char delimitador = ',';   // Tipo de separador que se va a utilizar
const int cantidadDatos = 3;    // Datos recibidos desde el puerto serial
double datos[cantidadDatos];    // Velocidad referencia global del robot


//////////////////////////  MOTOR 1 TRASERO ///////////////////
int AIN1T = 10;                                 // Entradas a controlar
int AIN2T = 11;                                 // Entradas a controlar
const int CanalTrasero1 = 7;                    // Conexión Encoder motor Trasero canal 1
const int CanalTrasero2 = 4;                    // Conexión Encoder motor Trasero canal 2
volatile int estadoAnteriorMotorTrasero = 0;    // Variable estado anterior motor trasero
volatile int estadoActualMotorTrasero = 0;      // Variable estado actual motor trasero
int valorSalidaMotorTrasero = 0;                // Valores de salida del motor Trasero
volatile int cmt = 0;                           // Contador motor trasero
double wT = 0;
double wRefMotorT = 0;                          // Velocidad de referencia motor trasero

//////////////////////////  MOTOR 2 IZQUIERDO ///////////////////
int BIN1I = 9;
int BIN2I = 6;
const int CanalIquierdo1 = A1;                  // Conexión Encoder motor izquierdo canal 1
const int CanalIquierdo2 = A0;                  // Conexión Encoder motor izquierdo canal 2
volatile int estadoAnteriorMotorIquierdo = 0;   // Variable estado anterior motor izquierdo
volatile int estadoActualMotorIzquierdo = 0;    // Variable estado actual motor izquierdo
int valorSalidaMotorIzquierdo = 0;              // Valores de salida del motor Izquierdo
volatile int cmi = 0;                           // Contador motor izquierdo
double wI = 0;
double wRefMotorI = 0;                          // Velocidad de referencia motor izquierdo

//////////////////////////  MOTOR 3 DERECHO ///////////////////

int AIND1 = 3;
int AIND2 = 5;
const int CanalDerecho1 = A3;                   // Conexión Encoder motor derecho canal 1
const int CanalDerecho2 = A2;                   // Conexión Encoder motor derecho canal 2
volatile int estadoAnteriorMotorDerecho = 0;    // Variable estado anterior motor derecho
volatile int estadoActualMotorDerecho = 0;      // Variable estado actual  motor derecho
int valorSalidaMotorDerecho = 0;                // Valores de salida del motor derecho
volatile int cmd = 0;                           // Contador motor derecho
double wD = 0;
double wRefMotorD = 0;                          // Velocidad de referencia motor Derecho
```

```
//////// VARIABLES PARA CALCULAR VELOCIDADES ANGULARES /////////

double valorConstante = 6.9968;  // (1000*2*pi)/R ---> R = 898 Resolucion encoder cuadruple

////////////////////////// PAMRAMETROS DEL ROBOT ////////////////////////
// velocidades medidas
double ufMedidaRobot = 0;                // Velocidad frontal
double ulMedidaRobot = 0;                // Velocidad lateral
double wMedidaRobot = 0;                 // Velocidad angular
double alfa = 0;                         // Orientacion del robot
double R = 0.029;                        // Radio de la llanta
double L = 0.11;       // Distancia entre el centro del robot y el eje central de la rueda

//////////////////////////// BATERIA ////////////////////////
const int pinBateria = A4;
const int indicadorDeCarga = 2;

void setup() {

  Serial.begin(9600);

 ////////////////// SINTONIA FINA PID //////////////////

  motor_1.setGains(0.19, 0.095, 0.00);                // (Kc,Ti,Td) //0.22, 0.17, 0.03

  ////////////////////// Normalizar los datos ///////////////
  ////////////////// Limites de señales ////////////////////
  motor_1.setCvLimits(255,40);                    // Limites maximos acciones de control
  motor_1.setPvLimits(12,0);      // Limites máximos variable de procesos es decir rad/s


  motor_2.setGains(0.23, 0.095, 0.04);           // (Kc,Ti,Td) //0.23, 0.19, 0.04


  ////////////////// Limites de señales ////////////////////
  motor_2.setCvLimits(255,40);                    // Limites maximos acciones de control
  motor_2.setPvLimits(12,0);      // Limites máximos variable de procesos es decir rad/s


  motor_3.setGains(0.18, 0.06, 0.01);            // (Kc,Ti,Td) //0.18, 0.12, 0.01


  ////////////////// Limites de señales ////////////////////
  motor_3.setCvLimits(255,40);                      // Limites maximos acciones de control
  motor_3.setPvLimits(12,0);      // Limites máximos variable de procesos es decir rad/s

//////////////////////  MOTOR 1 ATRAS /////////////////
  pinMode(AIN2T, OUTPUT);
  pinMode(AIN1T, OUTPUT);
  pinMode(CanalTrasero1, INPUT);
  pinMode(CanalTrasero2, INPUT);

  digitalWrite(AIN2T, LOW);
  digitalWrite(AIN1T, LOW);
```

```
  attachPinChangeInterrupt(digitalPinToPinChangeInterrupt(CanalTrasero1),
encoderMotorTrasero, CHANGE);
  attachPinChangeInterrupt(digitalPinToPinChangeInterrupt(CanalTrasero2),
encoderMotorTrasero, CHANGE);

//////////////////////  MOTOR 2 IZQUIERDA //////////////////
  pinMode(BIN2I, OUTPUT);
  pinMode(BIN1I, OUTPUT);

  pinMode(CanalIquierdo1, INPUT);
  pinMode(CanalIquierdo2, INPUT);

  digitalWrite(BIN2I, LOW);
  digitalWrite(BIN1I, LOW);

  attachPinChangeInterrupt(digitalPinToPinChangeInterrupt(CanalIquierdo1),
encoderMotorIzquierdo, CHANGE);
  attachPinChangeInterrupt(digitalPinToPinChangeInterrupt(CanalIquierdo2),
encoderMotorIzquierdo, CHANGE);


//////////////////////  MOTOR 3 DERECHA //////////////////
  pinMode(AIND2, OUTPUT);
  pinMode(AIND1, OUTPUT);
  pinMode(CanalDerecho1, INPUT);
  pinMode(CanalDerecho2, INPUT);

  digitalWrite(AIND2, LOW);
  digitalWrite(AIND1, LOW);

  attachPinChangeInterrupt(digitalPinToPinChangeInterrupt(CanalDerecho1),
encoderMotorDerecho, CHANGE);
  attachPinChangeInterrupt(digitalPinToPinChangeInterrupt(CanalDerecho2),
encoderMotorDerecho, CHANGE);

  ////////////////////////// BATERIA ////////////////////////
  pinMode(indicadorDeCarga,OUTPUT);
  digitalWrite(indicadorDeCarga,LOW);

  tiempoAnterior = millis();

}

void loop()
{
  ////////// SI RECIBE DATOS /////////////
  if (datosCompletos)
  {
    for (int i = 0; i < cantidadDatos ; i++)
    {
      int index = datosDeEntrada.indexOf(delimitador);
      datos[i] = datosDeEntrada.substring(0, index).toFloat();
      datosDeEntrada = datosDeEntrada.substring(index + 1);
     }
```

```
    velocidadMotores(datos[0],datos[1],datos[2]);

    datosDeEntrada = "";
    datosCompletos = false;
  }

  ////////////////// CONTROLADOR PID CALCULO DE VELOCIDADES ////////////////
  if(millis()-tiempoAnterior >= tiempoMuestreo)
  {
    wT = valorConstante*cmt/(millis()-tiempoAnterior);
    wI = valorConstante*cmi/(millis()-tiempoAnterior);
    wD = valorConstante*cmd/(millis()-tiempoAnterior);

    tiempoAnterior = millis();

    cmt = 0;
    cmi = 0;
    cmd = 0;

    velocidadDelRobot(wT,wI,wD);

    alfa = alfa+wMedidaRobot*0.1;

    //Serial.println(phi,2);
    Serial.println(ufMedidaRobot,2);
    Serial.println(ulMedidaRobot,2);
    Serial.println(wMedidaRobot,2);

    /////////// CALCULO DE ECUACIONES DE CONTROL ////////////////////////

    valorSalidaMotorTrasero = motor_1.compute(wRefMotorT,wT);
    valorSalidaMotorIzquierdo = motor_2.compute(wRefMotorI,wI);
    valorSalidaMotorDerecho = motor_3.compute(wRefMotorD,wD);

    if (valorSalidaMotorTrasero > 0)
sentidoAntihorario(AIN2T,AIN1T,valorSalidaMotorTrasero);
    else sentidoHorario(AIN2T,AIN1T,abs(valorSalidaMotorTrasero));

    if (valorSalidaMotorIzquierdo > 0)
sentidoAntihorario(BIN1I,BIN2I,valorSalidaMotorIzquierdo);
    else sentidoHorario(BIN1I,BIN2I,abs(valorSalidaMotorIzquierdo));

    if (valorSalidaMotorDerecho > 0)
sentidoAntihorario(AIND2,AIND1,valorSalidaMotorDerecho);
    else sentidoHorario(AIND2,AIND1,abs(valorSalidaMotorDerecho));
    bateria();
  }


}
////////////////////////////////////// RECIBIR DATOS ////////////////////////

///////Interrupción que se activa cuando hay datos en el puerto serial ////////
void serialEvent() {
  while (Serial.available()) {
```

```
    char inChar = (char)Serial.read();
    datosDeEntrada += inChar;
    if (inChar == '\n') {
      datosCompletos = true;
    }
  }
}

////////////////// Monitorear estados del motor Trasero ////////////////////

void encoderMotorTrasero(void)
{
    estadoAnteriorMotorTrasero = estadoActualMotorTrasero;
    estadoActualMotorTrasero = PIND & 144;

    if(estadoAnteriorMotorTrasero == 0   && estadoActualMotorTrasero == 16)   cmt++;
    if(estadoAnteriorMotorTrasero == 16  && estadoActualMotorTrasero == 144)  cmt++;
    if(estadoAnteriorMotorTrasero == 128 && estadoActualMotorTrasero == 0)    cmt++;
    if(estadoAnteriorMotorTrasero == 144 && estadoActualMotorTrasero == 128)  cmt++;

    if(estadoAnteriorMotorTrasero == 0 && estadoActualMotorTrasero == 128)   cmt--;
    if(estadoAnteriorMotorTrasero == 16 && estadoActualMotorTrasero == 0)    cmt--;
    if(estadoAnteriorMotorTrasero == 128 && estadoActualMotorTrasero == 144) cmt--;
    if(estadoAnteriorMotorTrasero == 144 && estadoActualMotorTrasero == 16)  cmt--;
}

///////////////// Monitorear estados del motor Izquierdo ////////////////////

void encoderMotorIzquierdo(void)
{
    estadoAnteriorMotorIquierdo = estadoActualMotorIzquierdo;
    estadoActualMotorIzquierdo = PINC & 3;

    if(estadoAnteriorMotorIquierdo == 0 && estadoActualMotorIzquierdo == 1)  cmi++;
    if(estadoAnteriorMotorIquierdo == 1 && estadoActualMotorIzquierdo == 3)  cmi++;
    if(estadoAnteriorMotorIquierdo == 2 && estadoActualMotorIzquierdo == 0)  cmi++;
    if(estadoAnteriorMotorIquierdo == 3 && estadoActualMotorIzquierdo == 2)  cmi++;

    if(estadoAnteriorMotorIquierdo == 0 && estadoActualMotorIzquierdo == 2)  cmi--;
    if(estadoAnteriorMotorIquierdo == 1 && estadoActualMotorIzquierdo == 0)  cmi--;
    if(estadoAnteriorMotorIquierdo == 2 && estadoActualMotorIzquierdo == 3)  cmi--;
    if(estadoAnteriorMotorIquierdo == 3 && estadoActualMotorIzquierdo == 1)  cmi--;
}

////////////////// Monitorear estados del motor Derecho ////////////////////
void encoderMotorDerecho(void)
{
    estadoAnteriorMotorDerecho = estadoActualMotorDerecho;
    estadoActualMotorDerecho =PINC & 12;

    if(estadoAnteriorMotorDerecho == 0  && estadoActualMotorDerecho == 4)   cmd++;
    if(estadoAnteriorMotorDerecho == 4  && estadoActualMotorDerecho == 12)  cmd++;
    if(estadoAnteriorMotorDerecho == 8  && estadoActualMotorDerecho == 0)   cmd++;
    if(estadoAnteriorMotorDerecho == 12 && estadoActualMotorDerecho == 8)   cmd++;
```

```
    if(estadoAnteriorMotorDerecho == 0 && estadoActualMotorDerecho == 8)   cmd--;
    if(estadoAnteriorMotorDerecho == 4 && estadoActualMotorDerecho == 0)   cmd--;
    if(estadoAnteriorMotorDerecho == 8 && estadoActualMotorDerecho == 12)  cmd--;
    if(estadoAnteriorMotorDerecho == 12 && estadoActualMotorDerecho == 4)  cmd--;
}

////////////////// Sentido de giro de los motores horario ///////////////////

void sentidoHorario(int pin1, int pin2, int pwm)
{
  digitalWrite(pin1, LOW);
  analogWrite(pin2,pwm);
}


///////////////// Sentido de giro de los motores antihorario /////////////////

void sentidoAntihorario(int pin1, int pin2, int pwm)
{
  digitalWrite(pin2, LOW);
  analogWrite(pin1,pwm);
}

void velocidadDelRobot(double w1, double w2, double w3)
{
  ufMedidaRobot = (0.5774*R)*(w2-w3);                    //Calculo de velocidad frontal
  ulMedidaRobot = -(R/3)*(w2-(2*w1)+w3);                 //Calculo velocidad lateral
  wMedidaRobot = -(R/(3*L))*(w1+w2+w3);                  //Calculo velocidad angular
}
void velocidadMotores(double uf, double ul, double w)
{
  wRefMotorT =  (ul-(L*w))/R;                       // Velocidad referencia motor trasero
  wRefMotorI = -(ul+(2*L*w)-1.7321*uf)/(2*R);       // Velocidad referencia motor izquierdo
  wRefMotorD = -(ul+(2*L*w)+1.7321*uf)/(2*R);       // Velocidad referencia motor derecho
}


////////////////////////// Monitorear estado de la bateria ////////////////////////
void bateria()
{
  double voltaje = 0.0;
  voltaje  = analogRead(pinBateria)*(5.0/1023.0);
  if (voltaje<3.3) digitalWrite(indicadorDeCarga,HIGH); else
digitalWrite(indicadorDeCarga,LOW);
}
```

FUNCIÓN PARA SIMULACIÓN EN PYTHON

```
from threading import Thread
import time
import pyvista as pv
import numpy as np
import glob

class robotics:

     def __init__(self,path="",color = None):

          self.color = color
          self.path = path
          filenames = glob.glob(self.path+"/*.stl")
          self.robot = []
          self.robotCopy = []
          self.isTrajectory = False

          for filename in filenames:
               self.robot.append(pv.PolyData(filename))
               self.robotCopy.append(pv.PolyData(filename))

     def configureScene(self,bounds, window_size =[1024, 768], title="Python Robotics"):
          self.bounds = bounds
          self.plotter = pv.BackgroundPlotter(window_size=window_size,title=title)
          self.plotter.set_background(color='white')

     def initRobot(self,x1,y1,phi,escala):

          self.x1 = x1
          self.y1= y1
          self.phi = phi

          self.escala = escala

          for i in range(len(self.robot)):
               self.robot[i].points *= self.escala
               self.robotCopy[i].points *= self.escala
               if self.color == None:
                    self.plotter.add_mesh(self.robotCopy[i],'black')
               else:
                    self.plotter.add_mesh(self.robotCopy[i],self.color[i])

     def initTrajectory(self,hx,hy):

          self.isTrajectory = True
          self.hx = hx
          self.hy = hy
          self.sizeh = len(self.hx)
          points =
np.column_stack((np.zeros(self.sizeh),np.zeros(self.sizeh),np.zeros(self.sizeh)))
          self.spline = pv.Spline(points,self.sizeh)
          self.plotter.add_mesh(self.spline,color='red',line_width = 4)
```

```
    def plotDesiredTrajectory(self,hxd,hyd):
        sizehd = len(hxd)
        points = np.column_stack((hxd,hyd,np.zeros(sizehd)))
        self.spline1 = pv.Spline(points,sizehd)
        self.plotter.add_mesh(self.spline1,color='blue',line_width = 4)

    def startSimulation(self,step = 1,ts=0.1):
        cpos = [(-8, -8, 8), # zoom x y z
        (0.5, 0.5, 0.5), # Movimiento x y z
        (0.28, 0.28, 0.28)]
        self.plotter.show_bounds(grid='True',location = 'outer',color = '#000000',bounds =
self.bounds, xlabel = 'x [m]', ylabel = 'y [m]', zlabel = 'z [m]')
        #self.plotter.camera_position = cpos
        self.plotter.view_isometric()
        self.step = step
        self.ts = ts
        self.thread = Thread(target=self.simulation)
        self.thread.start()

    def simulation(self):
        for k in range(0,len(self.x1),self.step):
            if self.isTrajectory:
                self.plotTrajectory(self.hx[k],self.hy[k],k)
            self.robotUniciclo(self.x1[k],self.y1[k],self.phi[k],k)
            time.sleep(self.ts)

    def robotUniciclo(self,x1,y1,phi,k):
        Rz = np.array([[np.cos(phi),-np.sin(phi),0],
                       [np.sin(phi), np.cos(phi),0],
                       [    0         ,     0         ,1]])


        for i in range(len(self.robotCopy)):
            self.robotCopy[i].points = (Rz@self.robot[i].points.transpose()).transpose()
            self.robotCopy[i].translate([x1,y1,0])

    def plotTrajectory(self,hx,hy,k):
        self.spline.points[k:self.sizeh,0] = hx
        self.spline.points[k:self.sizeh,1] = hy
```

DISEÑO DE APLICACIÓN MÓVIL

Para el diseño de la aplicación móvil se implementó Kivy en su versión 2.1.0, esta es una plataforma de código abierto escrita en Python destinada al desarrollo rápido de aplicaciones. Para más información puede consultar "*kivi Documentation*". (Developers, 2022)

¿Como instalar Kivy?

Para instalar Kivy en la computadora se debe hacer lo siguiente.

- En la barra de tareas en la opción de buscar escribir "comando del sistema" y ejecutarlo como administrador.
- Una vez abierta la ventana de comando del sistema para instalar la librería de Kivy debe escribir el comando pip install kivy y presionar ENTER, con esto empezara la descarga y se instalaran todos los paquetes requeridos para que la librería funcione correctamente.
- Realizado los pasos anteriores el sistema queda listo para empezar a desarrollar la aplicación con ayuda de Python.

CÓDIGO PARA DESARROLLO DE LA APLICACIÓN MÓVIL

Lo primero que se debe hacer es crear dos scripts los cuales deben llevar por nombre main.py y main.kv. El primero hace referencia a una extensión de Python y el segundo a una extensión de kivy, estos dos archivos deben ser guardados en una misma carpeta ya que de ello depende que al momento de ejecutar el archivo con la extensión main.py la aplicación que se esté desarrollando funcione correctamente.

A continuación, se muestra los dos archivos implementados en la creación de la aplicación móvil.

El archivo main.py es el siguiente

```
from kivy.app import App
from kivy.uix.screenmanager import ScreenManager, Screen
from kivy.properties import BooleanProperty

class MainWid(ScreenManager):
    pass

class UnaScreen(Screen):
    count1_enabled1 = BooleanProperty(False)

    def on_toggle_button1_state(self, widget):
        print('Toggle state' + widget.state)
        if widget.state== 'normal':
            self.count1_enabled1 = False
        else:
            self.count1_enabled1 = True

class MainApp(App):
    title = "Screen Manager"
    def build(self):
        return MainWid()

if __name__=="__main__":
    MainApp().run()
```

Archivo main.kv

```
<UnaScreen>:
    name: 'Screen_DOS'
    BoxLayout:
        orientation: 'vertical'
        padding:dp(10)
        spacing:dp(10)
        Image:
            source:"rapida.jpg"
            allow_stretch: True
            keep_ratio: False
    BoxLayout:
        orientation: 'vertical'
        padding:dp(10)
        spacing:dp(10)
        Button:
            text: 'ELIGA SU OPCIÓN PREFERIDA'
            background_color: 0,0,0,0.5
            font_size: 28

        ToggleButton:
            id: 'boton_uno'
            text:'PIZZA'
            background_color: 0,0,1,0.7
            font_size: 28
            on_state: root.on_toggle_button1_state(self)

        ToggleButton:
            id: 'boton_dos'
            text:'HAMBURGESA'
            background_color: 0,0,1,0.7
            font_size: 28
            on_state: root.on_toggle_button1_state(self)

        ToggleButton:
            id: 'boton_tres'
            text:'PERRO CALIENTE'
            background_color: 0,0,1,0.7
            font_size: 28
            on_state: root.on_toggle_button1_state(self)

        BoxLayout:
            orientation: 'horizontal'
            padding:dp(15)
            spacing:dp(15)
            Button:
                text:'INICIO'
                background_color: 1,1,1,0.8
                on_press: root.manager.current = "SCREEN_UNO"

<MainWid>:
    Screen:
        name: 'SCREEN_UNO'
        BoxLayout:
            orientation: 'vertical'
            padding:dp(10)
```

```
            spacing:dp(10)
            Image:
                source:"hamburguesas.jpg"
                allow_stretch: True
                keep_ratio: False
        BoxLayout:
            orientation: 'vertical'
            padding:dp(15)
            spacing:dp(15)
            Button:
                text: 'RESTAURANTE'
                background_color: 0,0,0,0.8
                font_size: 30
            Button:
                text:'POR FAVOR SELECCIONE SU MESA'
                background_color: 0,0,0,0.7
                size_hint: 1, 0.4
                font_size: 25
            Button:
                text:'MESA UNO'
                background_color: 1,0,0,0.7
                font_size: 24
                on_press: root.current = "Screen_DOS"
            Button:
                text:'MESA DOS'
                background_color: 1,0,0,0.7
                font_size: 24
                on_press: root.current = "Screen_DOS"
            Button:
                text:'MESA TRES'
                background_color: 1,0,0,0.7
                font_size: 24
                on_press: root.current = "Screen_DOS"
UnaScreen:
```

Al ejecutar el archivo main.py se abre la aplicación desarrollada con ayuda de los dos scripts la cual se muestra en la figura 148.

Figura 148. Aplicación móvil.

Fuente: Elaboración propia

CÓDIGOS ROBOT ANTROPOMÓRFICO

FUNCIÓN DENAVIT-HARTENBERG

```
import numpy as np
import sympy as sp
import math

# FUNCION DENAVIT-HARTENBERG

def dh(theta, d, a, alpha):
    m = sp.Matrix([[sp.cos(theta),  -sp.cos(alpha)*sp.sin(theta),
                sp.sin(alpha)*sp.sin(theta),  a*sp.cos(theta)],
                [sp.sin(theta),   sp.cos(alpha)*sp.cos(theta), -
                sp.sin(alpha)*sp.cos(theta),  a*sp.sin(theta)],
                [0,                  sp.sin(alpha),
                sp.cos(alpha),               d],
                [0,0,0,1]])
    return m
```

FUNCIÓN DE DIBUJAR LINEA

```
#%Dibujar una linea a partir de dos puntos

from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
fig = plt.figure(1)
ax = fig.gca(projection='3d')

def dibujar_linea(pi, pf, color):#

    # punto inicial y final de cada cordenada
    # vectores que contienen las componentes de los puntos a dibujar
    x = [pi[0], pf[0]]
    y = [pi[1], pf[1]]
    z = [pi[2], pf[2]]

    #dibuja la linea con el color
    figure = ax.plot(x, y, z, c=color)

    return
```

FUNCIÓN DIBUJAR UN PUNTO

```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
fig = plt.figure(1)
ax = fig.gca(projection='3d')

#funcion dibujar punto
def dibujar_punto(px,py,pz):

    #cordenadas del punto
    x = px
    y = py
    z = pz

    #dibuja la linea con el color
    figure = ax.plot(x, y, z, c='r',marker='o')

    return
```

FUNCIÓN DIBUJAR UN SISTEMA DE REFERENCIA

```
import numpy as np
import sympy as sp
import math
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt

def dibujar_sistema_referencia_MTH(MTH, L,Subindicice):

    fig = plt.figure(1)
    ax = fig.gca(projection='3d')

    # Puntos finales de los ejes x, y, z
    pfx = np.dot(MTH,([L, 0, 0, 1]))
    pfy = np.dot(MTH,([0, L, 0, 1]))
    pfz = np.dot(MTH,([0, 0, L, 1]))

# Dibuja los ejes
    ax.plot([MTH[0, 3], pfx[0]], [MTH[1, 3], pfx[1]],  [MTH[2, 3], pfx[2]],
color='black', linewidth=2)
    ax.plot([MTH[0, 3], pfy[0]], [MTH[1, 3], pfy[1]],  [MTH[2, 3], pfy[2]], color='y',
linewidth=2)
    ax.plot([MTH[0, 3], pfz[0]], [MTH[1, 3], pfz[1]],  [MTH[2, 3], pfz[2]], color='g',
linewidth=2)
    #plt.show()

    # Dibuja los titulos de los ejes coordenados
    ax.text(pfx[0], pfx[1], pfx[2], ('x',Subindicice), color='black')
    ax.text(pfy[0], pfy[1], pfy[2], ('y',Subindicice), color='black')
    ax.text(pfz[0], pfz[1], pfz[2], ('z',Subindicice), color='black')

    return
```

FUNCIÓN CINEMÁTICA DIRECTA ROBOT ANTROPOMÓRFICO

```
import numpy as np
import sympy as sp
import math
from dh import dh
from dibujar_linea import dibujar_linea
from dibujar_sistema_de_referencia_MTH import dibujar_sistema_referencia_MTH
import matplotlib.pyplot as plt

#FUNCION CINEMATICA DIRECTA

def cinematica_directa(q):

    L1 = 123
    L2 = 190
    L3 = 120
    L4 = 120
    pi = math.pi

#Parámetros Denavit-Hartenberg del robot antropomórfico
    theta_dh = sp.Matrix([[q[0], q[1], q[2], q[3]]])
    d_dh = sp.Matrix([[L1, 0, 0, 0]])
    a_dh = sp.Matrix([[0, L2, L3, L4]])
    alpha_dh = sp.Matrix([[pi/2, 0, 0, 0]])


# MATRICES DE TRANFORMACIÓN
#Generación de matrices homogéneas de los sistemas
    A00 = np.eye(4)
    A01 = np.array(sp.simplify(dh(theta_dh[0], d_dh[0], a_dh[0], alpha_dh[0])))
    A12 = np.array(sp.simplify(dh(theta_dh[1], d_dh[1], a_dh[1], alpha_dh[1])))
    A23 = np.array(sp.simplify(dh(theta_dh[2], d_dh[2], a_dh[2], alpha_dh[2])))
    A34 = np.array(sp.simplify(dh(theta_dh[3], d_dh[3], a_dh[3], alpha_dh[3])))

    A02 = np.dot(A01, A12)
    A03 = np.dot(A02, A23)
    A04 = np.dot(A03, A34)
    A04 = np.array(sp.simplify(A04))
    print(A04)
   # EXTRACCION DE LAS POSICIONES DE CADA MATRIZ
    p00 = np.array(A00[:, 3])
    p01 = np.array(A01[:, 3])
    p02 = np.array(A02[:, 3])
    p03 = np.array(A03[:, 3])
    p04 = np.array(A04[:, 3])
#dibujamos las lineas decada eslabon
    dibujar_linea(p00, p01, 'r')
    dibujar_linea(p01, p02, 'b')
    dibujar_linea(p02, p03, 'r')
    dibujar_linea(p03, p04, 'b')
#dibujamos los ejes de cordenada de cada eslabon
    dibujar_sistema_referencia_MTH(A00, 40, 0)
    dibujar_sistema_referencia_MTH(A01, 40, 1)
    dibujar_sistema_referencia_MTH(A02, 40, 2)
    dibujar_sistema_referencia_MTH(A03, 40, 3)
    dibujar_sistema_referencia_MTH(A04, 40, 4)

    return
```

FUNCIÓN CINEMÁTICA INVERSA

```
import numpy as np
import sympy as sp
import math



def cinematica_inversa(t_x4, t_y4, t_z4):
    L1 = 135
    L2 = 190
    L3 = 120
    L4 = 120
    pi = math.pi

# CALCULOS q1
    q1 = math.atan2(t_y4, t_x4)
    q11=int((q1*180)/pi)
    t_x = t_x4-L4*sp.cos(-pi/2)*sp.sin(math.atan2(t_y4, t_x4))
    t_y = t_y4-L4*sp.cos(-pi/2)*sp.cos(math.atan2(t_y4, t_x4))
    t_z = t_z4-L4*sp.sin(-pi/2)

# CALCULOS q3

    cosenq3 = (t_x**2 + t_y**2 + (t_z-L1)**2 - (L2)**2 - (L3)**2)/(2*L2*L3)
    senoq3 = -1*math.sqrt(1-cosenq3**2)
    q3 = math.atan2(senoq3, cosenq3)
    q33=int((q3*180)/pi)

# CALCULOS q2

    alpha = math.atan2((t_z-L1), math.sqrt(t_x**2+t_y**2))
    betha = math.atan2(L3*sp.sin(q3), (L2+L3*sp.cos(q3)))
    q2 = alpha-betha
    q22=int((q2*180)/pi)

# CALCULOS q4

    q4 = -pi/2 - q2 - q3
    q44=int((q4*180)/pi)
    q=[q1, q2, q3, q4]
    qt=[q11, q22, q33, q44]

    return(qt)
```

JACOBIANO

```
import numpy as np
import sympy as sp
import math
from dh import dh
def jacobiano(q):
    L1 = 123
    L2 = 190
    L3 = 120
    L4 = 100
    pi = math.pi

    # MATRICES DE TRANFORMACIÓN
    theta_dh = sp.Matrix([[q[0], q[1], q[2], q[3]]])
    d_dh = sp.Matrix([[L1, 0, 0, 0]])
    a_dh = sp.Matrix([[0, L2, L3, L4]])
    alpha_dh = sp.Matrix([[pi/2, 0, 0, 0]])

    A00 = np.eye(4)
    A01 = np.array(sp.simplify(dh(theta_dh[0], d_dh[0], a_dh[0], alpha_dh[0])))
    A12 = np.array(sp.simplify(dh(theta_dh[1], d_dh[1], a_dh[1], alpha_dh[1])))
    A23 = np.array(sp.simplify(dh(theta_dh[2], d_dh[2], a_dh[2], alpha_dh[2])))
    A34 = np.array(sp.simplify(dh(theta_dh[3], d_dh[3], a_dh[3], alpha_dh[3])))

    A02 = np.dot(A01, A12)
    A03 = np.dot(A02, A23)
    A04 = np.dot(A03, A34)
    A04 = np.array(sp.simplify(A04))

    # calculo del primer jacobiano

    z00 = np.array([0, 0, 1])
    P04 = np.array(A04[0:3, 3])
    J1 = np.cross(z00, P04)

    ##calculo el segundo jacobiano
    z01 = np.array(A01[0:3, 2])
    P14 = np.array(A04[0:3, 3])-np.array(A01[0:3, 3])
    J2 = np.cross(z01, P14)

    ##calculo el tercer jacobiano
    z02 = np.array(A02[0:3, 2])
    P15 = np.array(A04[0:3, 3])-np.array(A02[0:3, 3])
    J3 = np.cross(z01, P15)

    ##calculo el cuarto jacobiano
    z03 = np.array(A03[0:3, 2])
    P16 = np.array(A04[0:3, 3])-np.array(A03[0:3, 3])
    J4 = np.cross(z01, P16)

#jacobiano total
    JT = np.array([J1,J2,J3,J4])
    print(JT)
```

CÓDIGO DE INTERFAZ GRÁFICA

```
import sys
from turtle import width
from typing_extensions import Self
from PyQt5.QtWidgets import QApplication, QMainWindow, QGraphicsDropShadowEffect
from PyQt5.QtCore import QPropertyAnimation, QEasingCurve
from PyQt5.QtGui import QColor
from PyQt5 import QtCore, QtWidgets
from PyQt5.uic import loadUi
from PyQt5.QtSerialPort import QSerialPortInfo
from qt_material import apply_stylesheet
from pyArduino import serialArduino

from todo  import completa

class VentanaPrincipal(QMainWindow):
    def __init__(self):
        super(VentanaPrincipal,self).__init__()
        loadUi('interfaz_moderna.ui',self)

        self.Boton_Menu_Uno.clicked.connect(self.mover_menu)
        self.Boton_Menu_Dos.clicked.connect(self.mover_menu)

        # Ocultamos los botones
        self.Boton_Restaurar.hide()
        self.Boton_Menu_Dos.hide()

        # Sombre de los widgets
        self.sombra_frame(self.frame_3)
        self.sombra_frame(self.comunicacion)
        self.sombra_frame(self.menu_lateral)
        self.sombra_frame(self.frame_superior)
        self.sombra_frame(self.toolBox)
        self.sombra_frame(self.Boton_Uno)
        self.sombra_frame(self.Boton_Dos)
        self.sombra_frame(self.Boton_Tres)
        self.sombra_frame(self.boton_conectar)
        self.sombra_frame(self.boton_desconectar)
        self.sombra_frame(self.boton_actualizar)



        # Control barra de titulos
        self.Boton_Minimizar.clicked.connect(self.control_boton_minimizar)
        self.Boton_Restaurar.clicked.connect(self.control_boton_normal)
        self.Boton_Maximizar.clicked.connect(self.control_boton_maximizar)
        self.Boton_Cerrar.clicked.connect(lambda: self.close())

        # Elimimar barra y e titulo - opacidad
        self.setWindowFlag(QtCore.Qt.FramelessWindowHint)
        self.setWindowOpacity(1)

        # SizeGrip
        self.gripSize = 10
        self.grip = QtWidgets.QSizeGrip(self)
        self.grip.resize(self.gripSize, self.gripSize)

        # Mover ventana
        self.frame_superior.mouseMoveEvent = self.mover_ventana
```

```
# Estado de los botones al iniciar
        self.Boton_Uno.setEnabled(False)
        self.Boton_Dos.setEnabled(False)
        self.Boton_Tres.setEnabled(False)
        self.boton_desconectar.setEnabled(False)


        # Conexion arduino
        self.boton_conectar.clicked.connect(self.conectar)
        self.boton_actualizar.clicked.connect(self.puertos_disponibles)
        self.boton_desconectar.clicked.connect(self.desconectar)
        self.Boton_Uno.clicked.connect(self.palabra_1)
        self.Boton_Dos.clicked.connect(self.palabra_2)
        self.Boton_Tres.clicked.connect(self.palabra_3)


    def control_boton_minimizar(self):
        self.showMinimized()

    def control_boton_normal(self):
        self.showNormal()
        self.Boton_Restaurar.hide()
        self.Boton_Maximizar.show()

    def control_boton_maximizar(self):
        self.showMaximized()
        self.Boton_Maximizar.hide()
        self.Boton_Restaurar.show()

    # Creamos el metodo sombra
    def sombra_frame(self, frame):
        sombra = QGraphicsDropShadowEffect(self)
        sombra.setBlurRadius(30)
        sombra.setXOffset(8)
        sombra.setYOffset(8)
        sombra.setColor(QColor(0,0,0,255))
        frame.setGraphicsEffect(sombra)

# SizeGrip
    def resizeEvent(self, event):
        rect = self.rect()
        self.grip.move(rect.right() - self.gripSize, rect.bottom() - self.gripSize)

    # Mover Ventana
    def mousePressEvent(self, event):
        self.clickPosition = event.globalPos()

    def mover_ventana(self, event):
        if self.isMaximized() == False:
            if event.buttons() == QtCore.Qt.LeftButton:
                self.move(self.pos() + event.globalPos() - self.clickPosition)
                self.clickPosition = event.globalPos()
                event.accept()
        if event.globalPos().y() <= 10:
            self.showMaximized()
            self.Boton_Maximizar.hide()
```

```
            self.Boton_Restaurar.show()
        else:
            self.showNormal()
            self.Boton_Restaurar.hide()
            self.Boton_Maximizar.show()

# Método para mover el menu lateral izquierdo
    def mover_menu(self):
        if True:
            width = self.menu_lateral.width()
            normal = 0
            if width == 0:
                extender = 300
                self.Boton_Menu_Dos.hide()
                self.Boton_Menu_Uno.show()

            else:
                self.Boton_Menu_Dos.show()
                self.Boton_Menu_Uno.hide()
                extender = normal
            self.animacion = QPropertyAnimation(self.menu_lateral, b"maximumWidth")
            self.animacion.setStartValue(width)
            self.animacion.setEndValue(extender)
            self.animacion.setDuration(500)
            self.animacion.setEasingCurve(QEasingCurve.OutInBack)
            self.animacion.start()

# Método para verificar lo puertos disponibles
    def puertos_disponibles(self):
        self.baudrates = ['9600']

        portList =[]
        ports = QSerialPortInfo().availablePorts()

        for i in ports:
            portList.append(i.portName())

        self.puertos_com.clear()
        self.baudios.clear()
        self.puertos_com.addItems(portList)
        self.baudios.addItems(self.baudrates)
        self.baudios.setCurrentText("9600")
         # Método para iniciar la comunicación serial
    def conectar(self):
        port = self.puertos_com.currentText()
        baudRate = self.baudios.currentText()
        self.arduino = serialArduino(port,baudRate,3)
        self.conectado.setText("<html><head/><body><p><span style=\"
color:#55aa00;\">Conectado</span></p></body></html>")
        self.Boton_Uno.setEnabled(True)
        self.Boton_Dos.setEnabled(True)
        self.Boton_Tres.setEnabled(True)
        self.boton_desconectar.setEnabled(True)
```

```
# Método para terminar la comunicación serial
    def desconectar(self):
        self.arduino.close() # Cerrar puerto serial
        self.conectado.setText("<html><head/><body><p><span style=\"
color:#FF0000;\">Desconectado</span></p></body></html>")
        self.Boton_Uno.setEnabled(False)
        self.Boton_Dos.setEnabled(False)
        self.Boton_Tres.setEnabled(False)
        self.boton_desconectar.setEnabled(False)


#funciones que llevan los puntos decada palabra
    def palabra_1(self):
        x1=[-20,-20,-20,-20,0,0,-20,-20,-20,-20,0,0,20,20,0,0,20,20,0]
        y1=[160,160,175,175,175,175,190,190,205,205,220,220,235,235,250,250,250,250,235]
        z1=[80,0,60,0,60,0,60,0,60,0,60,0,60,0,60,0,60,0,130]
        completa(x1, y1, z1, self.arduino)


    def palabra_2(self):
        x2=
[75,75,75,75,75,75,60,60,45,45,30,30,15,15,15,15,0,0,0,0,-15,-15,-30,-30,-30,-30,-30,-30,
-45,-45,-45,-45,-60,-60,0]
        y2=
[250,250,225,225,200,200,225,225,250,250,225,225,250,250,225,225,250,250,225,225,250,250,
250,250,225,225,200,200,250,250,200,200,225,225,235]
        z2=
[60,0,60,0,60,0,60,0,60,0,60,0,60,0,60,0,60,0,60,0,60,0,60,0,60,0,60,0,60,0,60,0,60,0,60,0,130
]
        completa(x2, y2, z2, self.arduino)


    def palabra_3(self):
        x3=
[75,75,75,75,60,60,60,60,60,60,45,45,30,30,15,15,0,0,-15,-15,-15,-15,-15,-15,-30,-30,-30,
-30,-30,-30,-45,-45,-60,-60,0]
        y3=
[250,250,225,225,250,250,225,225,200,200,225,225,250,250,225,225,250,250,250,250,225,225,
200,200,250,250,225,225,200,200,250,250,225,225,235]
        z3=
[80,0,80,0,80,0,80,0,80,0,80,0,80,0,80,0,80,0,80,0,80,0,80,0,80,0,80,0,80,0,80,0,80,0,80,0,130
]
        completa(x3, y3, z3, self.arduino)


if __name__=='__main__':
    app = QApplication(sys.argv)
    apply_stylesheet(app, theme='light_blue.xml')
    mi_app = VentanaPrincipal()
    mi_app.show()
    sys.exit(app.exec_())
```

FUNCIÓN PARA COMUNICACIÓN ENTRE PYTHON Y ARDUINO

```
from threading import Thread
import serial
import time
import sys
import numpy as np

class serialArduino:
     #FUNCION NOS PERMITE COMUNICACION
     def __init__(self,port,baud=9600,sizeData=1):
          self.port = port
          self.baud = baud
          self.sizeData = sizeData
          self.isReceiving = False
          self.isRun     = True
          self.thread    = None

          self.rawData = [None]*self.sizeData
          print('Intentando conectarse a: ' + str(self.port))
          try:
               self.serialConnection = serial.Serial(self.port,self.baud)
               print('Conectado a ' + str(self.port))
          except:
               sys.exit("No se pudo conectar a " + str(self.port))

     #FUNCION MANDA LOS DATOS ATRAVES DE LA COMUNICACION

     def sendData(self,ang):

          self.serialConnection.write(bytes(str(ang[0]) + ',' + str(ang[1]) + ',' +
str(ang[2]) + ',' + str(ang[3])+',' + str(ang[4]), 'utf-8'))

     #FUNCION NOS PERMITE DESCONECTAR LA COMUNICACION
     def close(self):
          self.isRun = False
          if self.thread == None:
               pass
          else:
               self.thread.join()
          self.serialConnection.close()
          print('Desconectado...')
```

FUNCIÓN PARA GENERAR TRAYECTORIAS

```
import numpy as np
from time import sleep
from cinematica_inversa import cinematica_inversa

def completa(x,y,z, configArduino):
    q11 = []
    q22 = []
    q33 = []
    q44 = []
    q=[0]
    # ingreso los valores a la funcion  cinematica inversa
    for i in range(len(x)):
        t_x4 = x[i]
        t_y4 = y[i]
        t_z4 = z[i]
        qt = cinematica_inversa(t_x4, t_y4, t_z4)

# guarda los valores que provienen de la cinematica inversa
        q11.append(abs(qt[0]))
        q22.append(abs(qt[1]))
        q33.append(abs(qt[2]))
        q44.append(abs(qt[3]))

    q1=q11+q
    q2=q22+q
    q3=q33+q
    q4=q44+q
    q22 = 0
    q33 = 0
#recoremos los valores para enviarselos al arduino
    for i in range(len(q1)):

        q22 = int((q2[i] - 0) * (0 - 180) / (180 - 0) + 180)
        q33 =int((q3[i] - 0) * (0 - 180) / (180 - 0) + 180)
        q = [q1[i], q2[i], q22, q33, q4[i]]

#enviamos los datos a la comunicacion
        configArduino.sendData(q)
        sleep(3)
```

CÓDIGO DE ARDUINO

```
#include <Separador.h>
#include <VarSpeedServo.h>

int tiempo = 15;
Separador se;
VarSpeedServo servo1;
VarSpeedServo servo2;
VarSpeedServo servo3;
VarSpeedServo servo4;
VarSpeedServo servo5;
void setup() {

//punto maxy min del servo

  servo1.attach(7, 450,2450);
  servo2.attach(8, 500,2470);
  servo3.attach(9, 480,2450);
  servo4.attach(10, 410,2360);
  servo5.attach(11, 830,2340);

// posicion inicial de los servos
  servo1.write(0);
  servo2.write(0);
  servo3.write(180);
  servo4.write(180);
  servo5.write(0);

// Iniciar la comunicacion serie
  Serial.begin(9600);
}
void loop() {

  while(Serial.available())
    {
       String dat = Serial.readString();
  //se recibe los datos de python
       String elemento1 = se.separa(dat, ',', 0);
       String elemento2 = se.separa(dat, ',', 1);
       String elemento3 = se.separa(dat, ',', 2);
       String elemento4 = se.separa(dat, ',', 3);
       String elemento5 = se.separa(dat, ',', 4);

       int pos1 = atoi(elemento1.c_str());
       int pos2 = atoi(elemento2.c_str());
       int pos3 = atoi(elemento3.c_str());
       int pos4 = atoi(elemento4.c_str());
       int pos5 = atoi(elemento5.c_str());

  //ingresa los angulos acada servo

       servo1.write(pos1, tiempo);
       servo2.write(pos2, tiempo);
       servo3.write(pos3, tiempo);
       servo4.write(pos4, tiempo);
       servo5.write(pos5, tiempo);
    }
}
```

CÓDIGO PARA SIMULACIÓN

```
 clear, close all, clc

x=linspace(50,50,1);      x11=linspace(50,50,1);
x22=linspace(50,50,1);    x33=linspace(50,50,1);
x44=linspace(50,50,1);    x55=linspace(50,35,1);
x66=linspace(35,35,1);    x77=linspace(35,20,1);
x88=linspace(20,20,1);    x99=linspace(20,5,1);
x10=linspace(5,5,1);      x1_1=linspace(5,-10,1);
x12=linspace(-10,-10,1);  x13=linspace(-10,-25,1);
x14=linspace(-25,-25,1);  x15=linspace(-25,-25,1);
x16=linspace(-25,-25,1);  x17=linspace(-25,-25,1);
x18=linspace(-25,-40,1);  x19=linspace(-40,-40,1);
x20=linspace(-40,0,1);
y=linspace(250,250,1);    y11=linspace(250,225,1);
y22=linspace(225,225,1);  y33=linspace(225,200,1);
y44=linspace(200,200,1);  y55=linspace(200,200,1);
y66=linspace(200,200,1);  y77=linspace(200,200,1);
y88=linspace(200,225,1);  y99=linspace(225,225,1);
y10=linspace(225,250,1);  y1_1=linspace(250,250,1);
y12=linspace(250,250,1);  y13=linspace(250,250,1);
y14=linspace(250,250,1);  y15=linspace(250,250,1);
y16=linspace(250,225,1);  y17=linspace(225,225,1);
y18=linspace(225,250,1);  y19=linspace(250,250,1);
y20=linspace(250,235,1);
z=linspace(60,0,1);    z11=linspace(0,60,1);
z22=linspace(60,0,1);  z33=linspace(0,60,1);
z44=linspace(60,0,1);  z55=linspace(0,60,1);
z66=linspace(60,0,1);  z77=linspace(0,60,1);
z88=linspace(60,0,1);  z99=linspace(0,60,1);
z10=linspace(60,0,1);  z1_1=linspace(0,60,1);
z12=linspace(60,0,1);  z13=linspace(0,60,1);
z14=linspace(60,0,1);  z15=linspace(0,60,1);
z16=linspace(60,0,1);  z17=linspace(0,60,1);
z18=linspace(60,0,1);  z19=linspace(0,60,1);
z20=linspace(60,130,1);
trax(:,1)=
[x,x11,x22,x33,x44,x55,x66,x77,x88,x99,x10,x1_1,x12,x13,x14,x15,x16,x17,x18,x19,x20];
tray(:,2)=
[y,y11,y22,y33,y44,y55,y66,y77,y88,y99,y10,y1_1,y12,y13,y14,y15,y16,y17,y18,y19,y20];
traz(:,3)=
[z,z11,z22,z33,z44,z55,z66,z77,z88,z99,z10,z1_1,z12,z13,z14,z15,z16,z17,z18,z19,z20];

% palabra vida
x2=[50,50,50,35,20,5,-10,-25,-25,-40];
y2=[250,225,200,200,225,250,250,250,225,250];
z2=[0,0,0,0,0,0,0,0,0,0];

%Obtenemos los angulos de cada articulacion
  for i = 1 : length(trax)
     t_x4=trax(i,1);, t_y4=tray(i,2);, t_z4=traz(i,3);
     %se ingresa posicion del efector final
     q = cinematica_inversa_MIO(t_x4,t_y4,t_z4);
    %guarda los angulos de cada articulacion
    q1(i)=q(1);
    q2(i)=q(2);
    q3(i)=q(3)
    q4(i)=q(4);
  end
%tama□o de todo los vectores
```

```
  for t = 1 : length(q1)
    cla;
    hold on;
     plot3(x2,y2,z2, '-*', 'color', 'b', 'LineStyle', 'none');
    DIBUJAR_ROBOT_MIO([q1(t);q2(t);q3(t);q4(t)]);
     axis([-200 350 -200 350 0 350]), view(140,30);
     pause(1);
end
```

REFERENCIAS BIBLIOGRÁFICAS

About Python™ | Python.org. (n.d.). Retrieved June 5, 2022, from https://www.python.org/about/

Adeva, R. (2022). *Todo lo que debes saber sobre la impresión 3D y sus utilidades*. https://www.adslzone.net/reportajes/tecnologia/impresion-3d/

Aguilar, J. (2013). *Robots móviles y antropomórficos*. https://es.slideshare.net/LuisAguilarCruz/robots-mviles-y-antropomrficos

Amazon.com. (n.d.). *BEMONOC 25GA370 DC Encoder Metal Gearmotor 12V Alta Velocidad 150RPM Motor de engranaje con dos canales Hall Efecto codificador para piezas DIY : Industrial y Científico*. Retrieved June 4, 2022, from https://www.amazon.com/-/es/BEMONOC-Gearmotor-Velocidad-engranaje-codificador/dp/B07GNFYGYQ/ref=sr_1_8?__mk_es_US=ÅMÅŽÕÑ&crid=1QPTS70HRIG7J&keywords=encoder+metal+gear+motor+150+rpm&qid=1654288055&sprefix=encoder+metal+gearmotor+150rpm%2Caps%2C226&sr=8-8

Arnáez, E. (2015). Enfoque práctico de la teoría de robots. Con aplicaciones en Matlab. *Enfoque Práctico de La Teoría de Robots. Con Aplicaciones En Matlab*. https://doi.org/10.19083/978-612-318-010-2

Boyer, J., Chouvet, B., Gebuhrer, L., Betuel, H., Descos, L., & Thivolet, J. (1982). Maladie coeliaque et lupus érythémateux disséminé. Association familiale et relation avec le système HLA-DR. In *La Nouvelle presse medicale* (Vol. 11, Issue 7, pp. 525–526).

Camargo, C. (2020). *10 usos prácticos que no sabías sobre impresión 3D.* https://somosmaker.com/10-usos-practicos-no-sabias-impresion-3d/

Catsigeras, E. (1998). *Definiciones*. 1–17.

Davila, A. (2007). *MODELO CINEMATICO Y CINETICO DE UN ROBOT OMNIDIRECCIONAL*. https://repositorio.uniandes.edu.co/bitstream/handle/1992/9609/u295443.pdf?sequence=1

Developers, T. K. (2022). *Kivy Documentation*.

Disfruta del poder del SO, computadoras y apps Windows 11 | Microsoft. (n.d.). Retrieved November 6, 2022, from https://www.microsoft.com/es-xl/windows?r=1

Documentación para el código de Visual Studio. (n.d.). Retrieved October 31, 2022, from https://code.visualstudio.com/docs

EAGLE | PCB Design And Electrical Schematic Software | Autodesk. (n.d.). Retrieved October 30, 2022, from https://www.autodesk.com/products/eagle/overview?term=1-YEAR&tab=subscription

Ferretrónica. (n.d.). *Modulo Puente H DRV8833 - Reemplaza TB6612FNG*. Retrieved June 4, 2022, from https://ferretronica.com/products/modulo-puente-h-drv8833-reemplaza-tb6612fng?variant=34471627882657&currency=COP&utm_medium=product_sync&utm_source=google&utm_content=sag_organic&utm_campaign=sag_organic&utm_campaign=gs-2021-10-19&utm_source=google&utm_medium=smart_campaign&gclid=Cj0KCQjw4uaUBhC8ARIsANUuDjXmXKC1_RMLRKRrwzb9naCnqcst_eAuVWjj_wT4ca1HGkrhI5H-i6QaAgxmEALw_wcB

Fritzing. (n.d.). Retrieved October 31, 2022, from https://fritzing.org/

Fusion 360 | Software CAD, CAM, CAE y de circuitos impresos 3D basado en la nube | Autodesk. (n.d.). Retrieved October 31, 2022, from https://latinoamerica.autodesk.com/products/fusion-360/overview?term=1-YEAR&tab=subscription

Garatejo, J., & Raudales, V. (2021). *DISEÑO DE UN MÓDULO PARA LA VALIDACIÓN DEL ALGORITMO DE DENAVIT-HARTENBERG DE UN BRAZO ANTROPOMÓRFICO APLICADO AL ÁREA DE AUTOMATIZACIÓN*. [Fundación Universitaria de América]. https://repository.uamerica.edu.co/bitstream/20.500.11839/8606/1/4161112-2021-2-IM.pdf

González-Barahona, J. M. (2011). EL CONCEPTO DE SOFTWARE LIBRE. *Revista Tradumàtica*. http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/viewArticle/1470/1385

Jabonero, J. (2010). *MODELADO Y ANALISIS DE UN BRAZO MECANICO*. https://e-archivo.uc3m.es/bitstream/handle/10016/10766/pfc_jesus_jabonero%28final%29.pdf?sequence=1&isAllowed=y

Manual del diseñador Qt. (n.d.). Retrieved November 5, 2022, from https://doc.qt.io/qt-6/qtdesigner-manual.html

Martínez, S., & Sisto, R. (2009). *Control y Comportamiento de Robots Omnidireccionales* [Universidad de la Republica Montevideo - Uruguay]. https://www.fing.edu.uy/inco/grupos/mina/pGrado/easyrobots/doc/SOA.pdf

MercadoLibre. (n.d.-a). *Arduino Atmega328p Ch340g Uno R3 Usb Arduino | MercadoLibre*. Retrieved June 4, 2022, from https://articulo.mercadolibre.com.co/MCO-451710166-arduino-atmega328p-ch340g-uno-r3-usb-arduino-

_JM?matt_tool=42035816&matt_word=&matt_source=google&matt_campaign_id=14634237770&matt_ad_group_id=122266243170&matt_match_type=&matt_network=g&matt_device=c&m

MercadoLibre. (n.d.-b). *Arduino Nano V3 Ch340 + Cable Usb | MercadoLibre*. Retrieved June 4, 2022, from https://articulo.mercadolibre.com.co/MCO-454327213-arduino-nano-v3-ch340-cable-usb-_JM?matt_tool=42035816&matt_word=&matt_source=google&matt_campaign_id=14634237770&matt_ad_group_id=122266243170&matt_match_type=&matt_network=g&matt_device=c&matt_creative=

MercadoLibre. (n.d.-c). *Batería Lipo Turnigy 7.4 V 2s 1000mah 20c Nueva*. Retrieved June 4, 2022, from https://articulo.mercadolibre.com.co/MCO-526963748-bateria-lipo-turnigy-74-v-2s-1000mah-20c-nueva-_JM

MercadoLibre. (n.d.-d). *Servomotor Mg995 Tower Pro Arduino Engranajes Metalicos | MercadoLibre*. Retrieved June 4, 2022, from https://articulo.mercadolibre.com.co/MCO-455154613-servomotor-mg995-tower-pro-arduino-engranajes-metalicos-_JM#position=1&search_layout=stack&type=item&tracking_id=175552d2-59cd-46ed-92aa-231e34f43476

Muñoz, V. ;, & García, A. ; (2015). *Modelado Cinematico Y Dinamico De Un Robot Móvil Omni-Direccional*. *April 2015*, 5. https://goo.gl/HbWdNB

Naylampmechatronics. (n.d.). *Módulo Bluetooth HC05*. Retrieved June 4, 2022, from https://naylampmechatronics.com/inalambrico/43-modulo-bluetooth-hc05.html

Robledano, A. (2019). *Qué es Python: Características, evolución y futuro | OpenWebinars*.

https://openwebinars.net/blog/que-es-python/

Robótica. (2021). *Robots móviles para almacén con ejemplos y fabricantes de robots móviles*. https://revistaderobots.com/robots-y-robotica/robots-moviles-para-almacen/

Sanchez, A., Angulo, C., & Garcia, M. (2010). *Robot móvil con aplicaciones metodologicas para el estudio de la robótica mediante lego NXT*. https://repositorio.cuc.edu.co/handle/11323/1086?show=full

Servomotor MG995 55g Digital Engranes de Metal - UNIT Electronics. (n.d.). Retrieved December 8, 2022, from https://uelectronics.com/producto/servomotor-mg995-55g-digital-engranes-de-metal/

Shawn, F. (2021). *Los mejores programas CAD gratis de 2021 | All3DP*. https://all3dp.com/es/1/mejores-programas-cad-gratuito-programas-diseno-cad-2d-3d/

Soloaga, A. (2018). *Python, los 5 usos más importantes de este lenguaje de programación*. https://www.akademus.es/blog/programacion/principales-usos-python/

Tecnical.cat. (n.d.). *Tecnical Robots ANTROPOMÓRFICOS Yakawa | Automatismos industriales. Soluciones en la automatización industrial. Manresa (Bages) Igualada (Anoia)*. Retrieved June 5, 2022, from https://www.tecnical.cat/es-robots-antropomorficos-tecnical-automatismos-industriales-ingenieria-manresa-bages-igualada-anoia.html

Tecnología, V. A.-R. de C. y, & 2006, undefined. (2006). Identificación de modelos de orden reducido a partir de la curvatura de reacción del proceso. *Revistas.Ucr.Ac.Cr*, *24*(2), 197–216. https://revistas.ucr.ac.cr/index.php/cienciaytecnologia/article/view/2647

Vallejo, A. (2021). *Python se convierte en el lenguaje de programación más popular según el*

índice TIOBE: adiós al largo reinado de C. https://www.genbeta.com/actualidad/python-se-convierte-lista-tiobe-lenguaje-popular-red-superando-incluso-a-c