Universidad de La Laguna

Trabajo de fin de grado

“ROBOT PARALELO CON SERVOS”

ESCUELA SUPERIOR DE INGENIERÍA Y TECNOLOGÍA

Titulación: Ingeniería Electrónica Industrial y Automática

Autor:

Efidencio César Pérez Gutiérrez

Tutor:

Jonay Tomás Toledo Carrillo

Julio de 2024

RESUMEN

En este trabajo final de grado se ha diseñado e implementado un robot paralelo, concretamente una plataforma de Stewart. Este robot formado por 6 actuadores aporta 6 grados de libertad, pudiendo cambiar su posición (x, y, z) y su orientación (R, P, Y) de forma independiente. Se ha realizado desde cero utilizando impresora 3D y servomotores de aeromodelismo.

En principio, la plataforma de Stewart utiliza actuadores lineales. Sin embargo, en este proyecto se han empleado servomotores de rotación, ya que son más rápidos, económicos y accesibles. Los pasos seguidos durante el proyecto han sido el diseño de cada una de las piezas del prototipo, impresión 3D de las mismas, montaje del prototipo y la programación y ejecución de los movimientos deseados mediante Arduino. La plataforma se ha diseñado e implementado eficazmente, y ha conseguido realizar las acciones deseadas cumpliendo con los objetivos planteados en el proyecto.

ABSTRACT

In this final degree project, a parallel robot has been designed and implemented, specifically a Stewart platform. This robot made up of 6 actuators provides 6 degrees of freedom, being able to change its position (x, y, z) and its orientation (R, P, Y) independently. It has been made from scratch using a 3D printer and aeromodelling model servos.

Initially, Stewart's platform uses linear actuators. However, in this project servomotors have been used that fit more within the budget and allow the proposed objectives to be achieved. The steps followed during the project have been the design of each of the prototype parts, their manufacture by 3D printing, assembly of the prototype and the programming and execution of the desired movements using Arduino boards. The platform has been designed and implemented effectively, and has managed to carry out the desired actions, meeting the objectives set in the project.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

En el presente trabajo de fin de grado correspondiente al título de Grado en Ingeniería Electrónica Industrial y Automática, se lleva a cabo la puesta a punto de un prototipo de robot paralelo con servos, desde su diseño hasta su implementación a través del microcontrolador Arduino. El robot diseñado es una plataforma de Stewart, pero en lugar de emplear actuadores prismáticos, se han utilizado servomotores rotacionales como actuadores, los cuales son más económicos.

1.2 Robots paralelos

Los robots paralelos son dispositivos compuestos por un soporte fijo unido a una plataforma móvil mediante una configuración de cadenas cinemáticas cerradas, es decir, el movimiento de un componente del mecanismo afecta a los demás. Además, el efector final está conectado directamente a la plataforma móvil [1]. En la bibliografía, los robots se clasifican atendiendo a si tienen un uso industrial, no industrial o especial, y es en este último grupo donde se enmarcan los robots paralelos. También se clasifican según su estilo de movimiento y grados de libertad en traslacionales (solo con movimiento de traslación), esféricos (no hay movimiento traslacional, solo de orientación) y mixtos (combinan rotación y traslación) [2].

1.3 Plataforma de Stewart

La plataforma Stewart es un tipo específico de robot paralelo con seis grados de libertad, por lo que también se le llama hexápodo. Este tipo de mecanismo se vuelve popular a raíz del trabajo publicado por D. Stewart [3] en 1965, el cual presentaba una plataforma móvil para llevar a cabo simulaciones de vuelo. Esta contaba con seis grados de movimiento, tres en direcciones lineales y tres angulares, tanto de forma individual como combinada. Como ilustra la figura 1, el planteamiento consistía en un plano triangular cuyas esquinas se conectaban a cada pata por una articulación de tres ejes. Además, la longitud de las patas era ajustable y estas se unían al suelo mediante articulaciones de dos ejes [3].

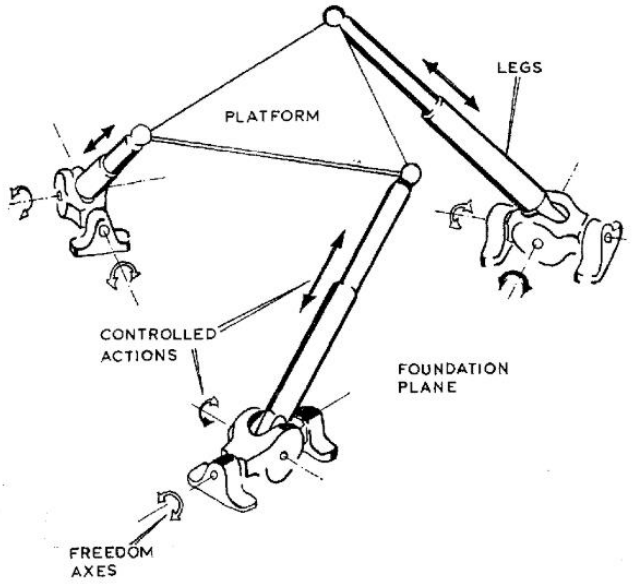


Figura 1. Plataforma de Stewart [3].

Las plataformas de Stewart actuales mantienen los principios básicos de la propuesta original, aunque han evolucionado en términos de diseño, materiales, tecnología de control y diversidad de aplicaciones. El mecanismo sigue constando de dos plataformas: una fija y otra móvil, que se conectan por las patas (variables en tamaño) a través de distintas combinaciones. Existen diferentes opciones para las patas, siendo algunas de las más modernas los cilindros hidráulicos o actuadores lineales eléctricos. La elección de la longitud de las patas y el ángulo de las articulaciones influye en el grado de movimiento. No obstante, el diseño siempre contempla moverse en seis direcciones, tres rotaciones y tres traslaciones [4].

1.4 Ventajas y desventajas

El interés científico acerca de los robots paralelos viene dado en parte por sus ventajas frente a los manipuladores en serie. En primer lugar, poseen un alto cociente carga/peso, lo cual los hace más eficientes desde el punto de vista energético. Además, ofrecen una mayor precisión gracias a su elevada rigidez y bajo peso. Y también son capaces de operar más rápido que los robots seriales [5].

No obstante, hay que tener en cuenta que los manipuladores paralelos pueden resultar menos ventajosos para algunas aplicaciones debido a que el área para

trabajar suele ser menor, a la mayor complejidad de su cinemática y la falta de un modelo dinámico general para su control [5].

1.5 Aplicaciones

En la industria se recurre frecuentemente a la plataforma Stewart cuando se requiere de un elevado control de movimiento y posiciones precisas. A continuación, se describen algunas de las aplicaciones más frecuentes:

- Seguimiento de objetivos [6]:

Puede ocurrir que los objetivos estén fijos mientras se mueve la plataforma como es el caso de los satélites geoestacionarios, o que sea el objetivo el que está en circulación y la plataforma sea fija como por ejemplo el radiotelescopio AMiBA (figura 2), un instrumento de seguimiento de galaxias y cuerpos celestes.

Figura 2. Radiotelescopio AMiBA [6].

- *Pick and place* [7]:

Aunque los robots *pick and place* son robots araña que van colgando, y no plataformas Stewart, ejemplifican la utilidad de los robots paralelos en la industria del embalaje, gestión de instrumentación médica, mecanizado de piezas de

madera o manejo de células fotovoltaicas. El robot Quattro (figura 3) es bastante popular por ser de los más rápidos del mercado. Consta de cuatro patas y cuatro grados de libertad, donde cada pata forma un sistema de paralelogramo que equilibra el centro de masas, aportando más estabilidad y precisión.

Figura 3. Robot Quattro [7].

- Medicina (rehabilitación y cirugía robótica) [7]:

Los robots paralelos suponen un apoyo al trabajo de los fisioterapeutas en terapias debido a la buena coordinación de sus movimientos y la elevada precisión de estos. Son especialmente útiles en terapias de rehabilitación de tobillo. Los robots planteados generalmente consideran una plataforma fijada al suelo y otra unida al pie. No obstante, también existen prototipos de tipo exoesqueleto en los que la base móvil va unida al pie mientras que la fija a la pierna.

Asimismo, la altísima precisión que ofrecen estos robots abre las puertas a grandes avances médicos en cuanto a cirugía mínimamente invasiva. De hecho, en el caso de los robots seriales ya han sido capaces de operar pacientes a través de orificios de tan solo 10 mm de diámetro. Los de tipo paralelo aún no están tan bien implantados en el mercado, pero existen varios prototipos propuestos en la bibliografía como, por ejemplo, el robot paralelo *P-arm* (figura 4), diseñado específicamente para asistir en cirugía laparoscópica.

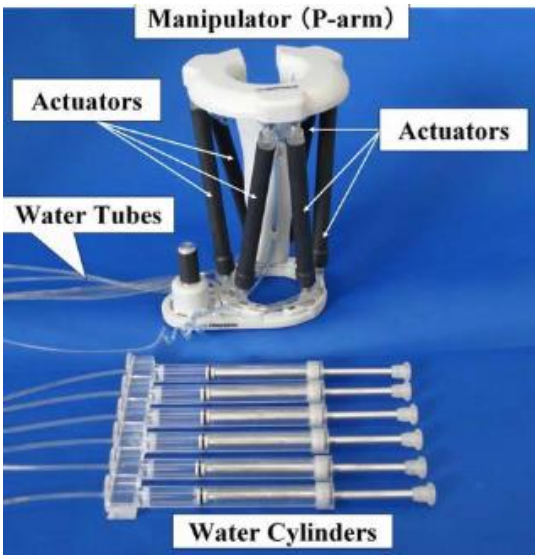


Figura 4. Prototipo del robot P-arm para laparocopia [7].

- Simulador de vuelo [6,8]:

Los robots paralelos, y en específico la plataforma Stewart, pueden imitar los movimientos y sensaciones que se dan al pilotar una aeronave. Esto permite instruir de manera segura a los pilotos acerca de protocolos de emergencia y maniobras de vuelo. En la figura 5 se observa el simulador de vuelo que utiliza Lufthansa, el cual puede recrear los controles de un avión real provocando las consecuentes vibraciones, aceleraciones o cambios de orientación en el espacio.

Figura 5. Simulador de vuelo de Lufthansa [6].

2. OBJETIVOS

2.1 Objetivo general

El objetivo general de este trabajo consiste en implementar un prototipo de robot paralelo con servos, basado en la plataforma Stewart que utiliza servomotores en lugar de actuadores prismáticos.

2.2 Objetivos específicos

Para lograr el objetivo general, se plantean los siguientes objetivos específicos:

- Diseñar y modelar las piezas que componen el mecanismo mediante la herramienta FreeCAD, para su posterior impresión en 3D.
- Hallar las ecuaciones de la cinemática inversa para su implementación en Arduino, de manera que la plataforma se encuentre en la posición más conveniente.
- Programar y ejecutar los movimientos de la plataforma a través del microcontrolador Arduino.
- Aplicar los conocimientos adquiridos durante el Grado en Ingeniería Electrónica Industrial y Automática.

3. DISEÑO

En la parte del diseño de la plataforma de Stewart se ha utilizado el software FreeCad. Es una aplicación de modelado paramétrico 3D, que permite la posterior impresión en una impresora 3D dada por la Universidad de La Laguna. Las piezas se han diseñado una a una, para su posterior ensamblaje.

3.1 Piezas diseñadas en FreeCAD

Para este proyecto se han realizado 6 piezas para su posterior impresión en 3D. A continuación, se explican las piezas que finalmente constituyen el prototipo, también las que no funcionaron, ya que todas han sido necesarias para completar la estructura de la plataforma de Stewart.

3.1.1 Base

- Primer diseño de la base:

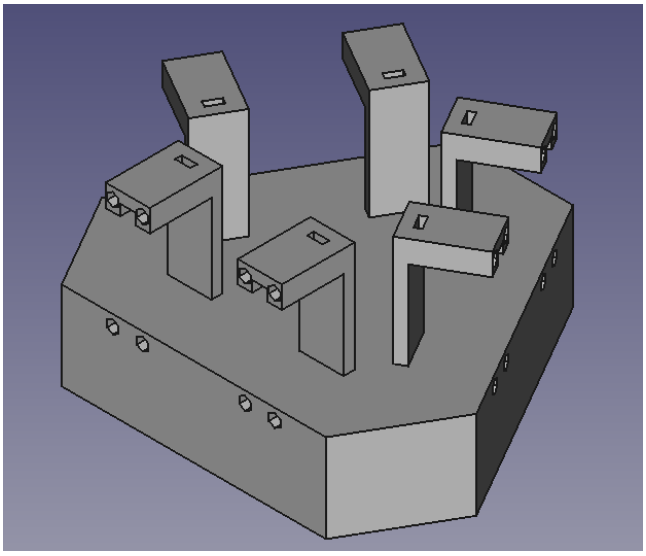


Figura 6. Base diseño inicial

En este primer diseño de la base se planteó la forma de un hexágono con tal que se dispusieran los servomotores dos a dos en los lados largos. Hay que tener en cuenta que había que hacer una sujeción para los servos ya que, al ponerlos en funcionamiento, estos tienden a moverse debido a la fuerza que ejercen. Por estos motivos, se planteó la disposición que se observa en la figura 6.

Pero este planteamiento se descartó ya que la impresora 3D no puede realizar esta configuración, debido a que su modo de operar provoca que mientras la

estructura se está imprimiendo se caigan los filamentos que no tienen apoyo. Además, esta configuración no es capaz de mantener una buena sujeción de los servos ya que pueden desviarse al accionarse.

Diseño final de la base:

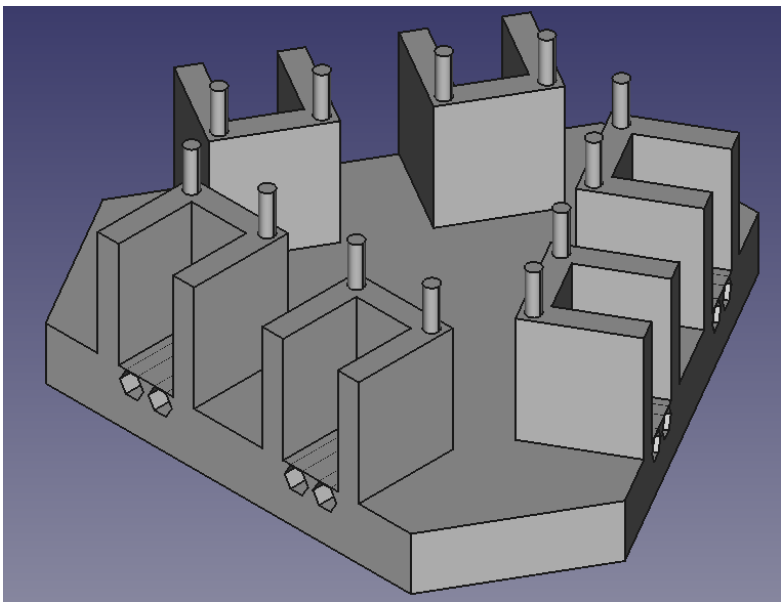


Figura 7. Base de la estructura

En el diseño final se han realizado varios cambios respecto al primero. El más significativo es el cambio del apoyo del servomotor. Se han tenido que introducir paredes laterales, por lo que, en la nueva disposición rectangular, los servomotores encajan y se apoyan correctamente.

También se han realizado huecos hexagonales (forma de tuercas) para atornillar los servomotores. Cabe mencionar que esto en el primer diseño estaba hecho, pero el tamaño de la tuerca no era el apropiado ya que era más pequeño. En cambio, en el nuevo diseño, el hueco está más pegado a la superficie y con un mayor tamaño para que pueda entrar la tuerca.

Hay que señalar que la impresora 3D no puede imprimir la forma de una rosca. Por eso se tiene que introducir primero una tuerca de seguridad y otra tuerca normal, las cuales llevan pegamento alrededor para pegarse a la pared del hueco. Después de eso, es posible enroscar el tornillo.

Por último, se añadieron unos pivotes en las esquinas de los apoyos de los servos esto se realizó para que posteriormente encajara con la tapa de la base.

3.1.2 Tapa-base

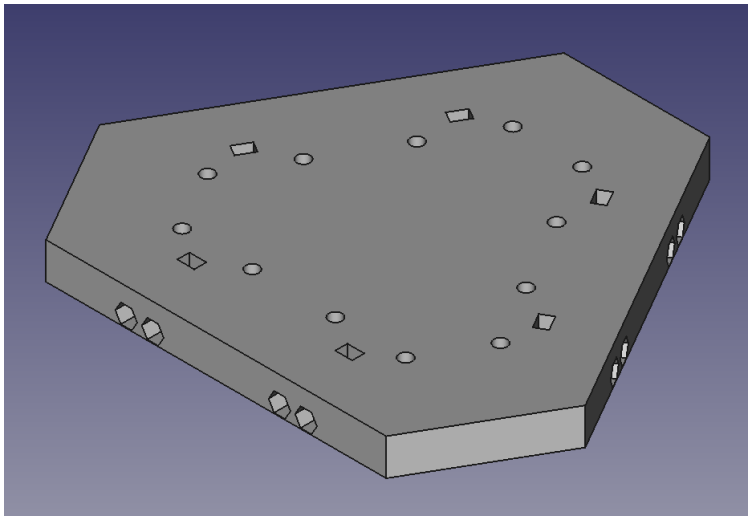


Figura 8. Tapa de la base

Esta pieza denominada tapa se planteó para conseguir una mejor sujeción de los servomotores e impedir que se desplacen hacia arriba. Se tuvieron en cuenta los agujeros para el cableado de los servos y los pivotes de la base. También se añadió el hueco para la sujeción de las tuercas.

Con esto se asegura que todo quede bien anclado y se puedan hacer los movimientos sin preocuparse de que se muevan los servomotores.

3.1.3 Plataforma móvil

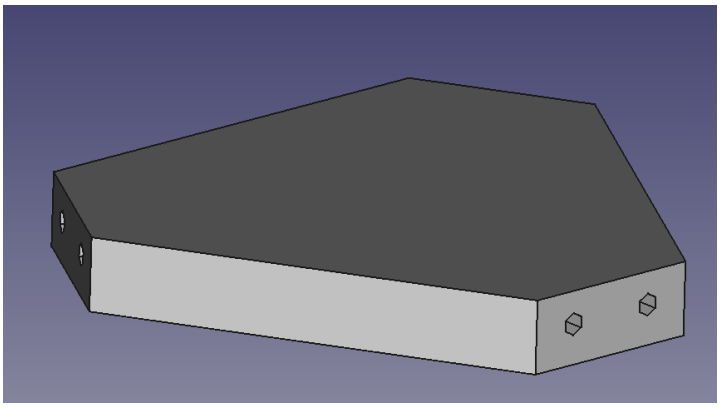


Figura 9. Plataforma móvil diseño inicial

- Primer diseño de la plataforma móvil:

La plataforma móvil se pensó para que la sujeción entre el brazo del servo y la plataforma móvil fuera a través del saliente de un tornillo con cabeza hexagonal.

Sin embargo, esta idea no era viable y se descartó ya que las varillas que se iban no permitían desarrollar este diseño de manera práctica. Por tanto, se rehace el diseño como se explica a continuación.

- Diseño final de la plataforma móvil:

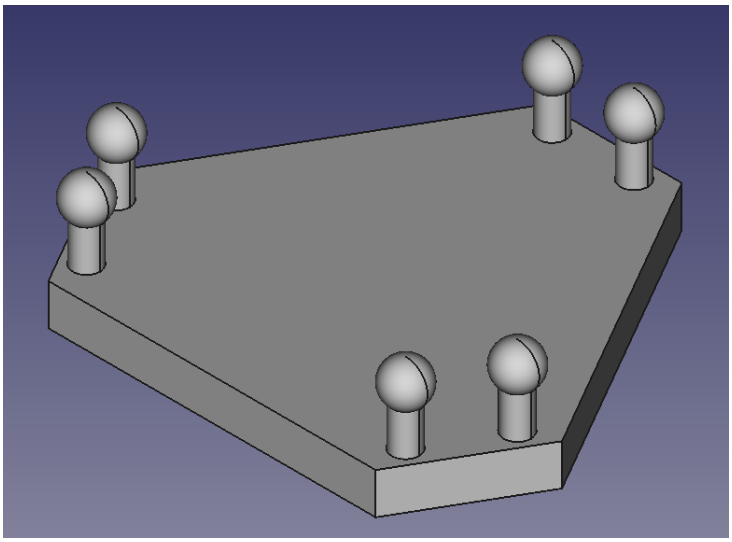


Figura 10. Plataforma móvil

Esta pieza es la plataforma móvil donde se conectan las varillas del brazo del servo. Es la misma geometría que la base, pero los apoyos para las varillas están en los lados cortos del hexágono, esta plataforma se coloca de forma que su lado corto está enfrentado con los lados largos del hexágono de la base. Se limita la distancia en la que se encuentran los servos de la plataforma para su mejor ensamble y mayor movimiento.

3.1.4 Apoyo inferior-varilla

- Diseño inicial del apoyo inferior-varilla:

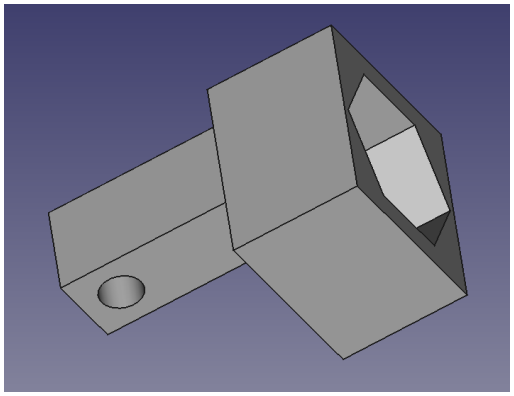


Figura 11. Diseño inicial apoyo inferior-varilla

Esta pieza se pensó para apoyar la parte inferior de la varilla (pieza normalizada) con el extremo del brazo del servo. La parte superior es más ancha para ajustarse al tamaño de la tuerca que soporta la varilla. Mientras que la parte inferior es más estrecha para que la pieza sea menos tosa.

Sin embargo, esto presentó el problema de que al accionar el robot e inclinarse la plataforma, se rompió la parte inferior al ser demasiado delgada. Por eso se tuvo que rediseñar esta pieza tal y como se observa en la figura 12.

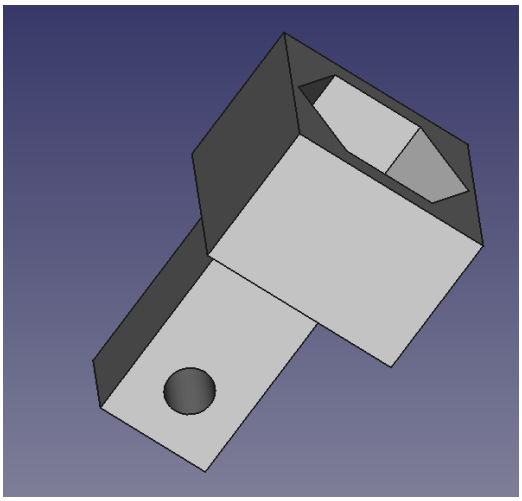


Figura 12.Diseño final apoyo inferior varilla

Para suplir este fallo, en el diseño se reubicó la posición del agujero, además de hacer un poco más larga la figura para que hubiera más grosor. El aspecto no cambia mucho con respecto a la anterior figura 11, pero sí ejerce su función de reforzar la parte donde va conectado el tornillo.

3.1.5 Apoyo superior-varilla

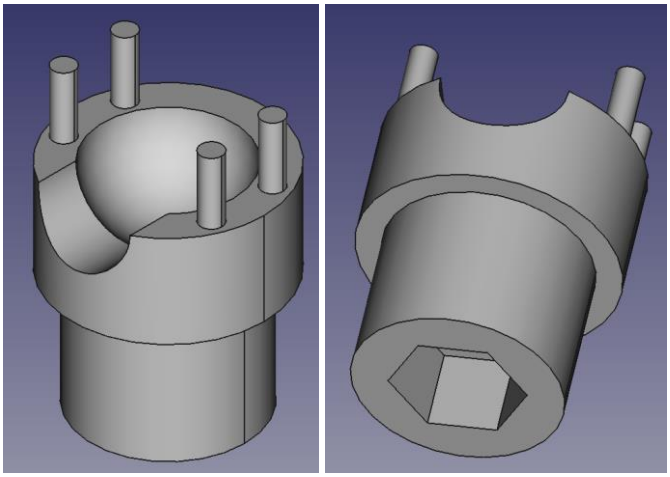


Figura 13. Apoyo parte superior varilla

El apoyo superior-varilla se ideó para servir de apoyo entre la plataforma móvil y la parte superior de la varilla.

La parte superior de la pieza tiene realizado un hueco semiesférico cuyo diámetro se ajusta al tamaño de la esfera que tiene la plataforma móvil, aunque no coincide exactamente, sino que es 55 mm más holgado porque hay que tener en cuenta el grosor del filamento de la impresora 3D. En la parte inferior se encuentra otro hueco hexagonal para la sujeción de tuerca donde encaja la varilla.

3.1.6 Tapa 855

- Diseño inicial de la tapa 855:

El desarrollo de esta pieza fue el más problemático del proyecto ya que había que realizar correctamente el hueco semiesférico, a la vez que se dejaba movilidad a la plataforma móvil.

El primer diseño de la tapa (figura 13) no cumplía los requisitos de movilidad necesarios, y los huecos para los pivotes resultaron demasiado pequeños ya que no se tuvo en cuenta la holgura de los filamentos de la impresora 3D.

Además, fueron necesarios varios intentos de diseño en los que se hicieron distintas aproximaciones del hueco semiesférico hasta hallar el correcto.

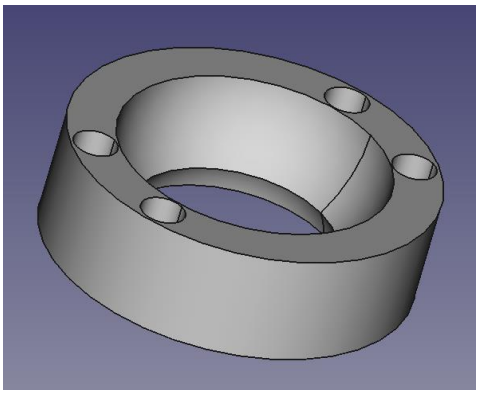


Figura 14.Diseño inicial Tapa 855

- Diseño final de la tapa 855:

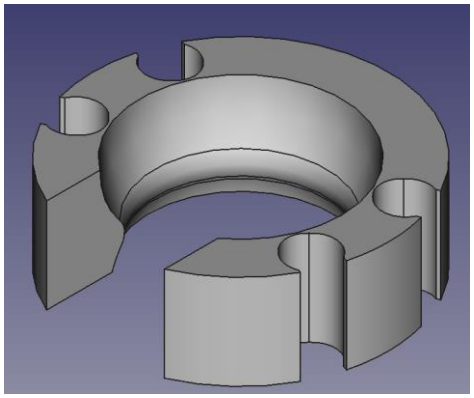


Figura 15. Diseño final de la tapa 855

En el planteamiento final de la tapa (figura 14) se introdujeron varias mejoras. En primer lugar, el medio del hueco semiesférico no acaba en punta, sino que se ha redondeado para darle mayor movilidad a los apoyos de la plataforma móvil. Asimismo, se realizó una abertura en los huecos de los pivotes para que estos encajasen mejor y tuvieran margen de movimiento al moverse la plataforma, favoreciendo el funcionamiento de toda la estructura.

3.2 Piezas normalizadas

Para el montaje del prototipo se han utilizado distintas piezas normalizadas como serian:

- Tornillo M2 cabeza hexagonal, 6 piezas
- Tuerca M2 hexagonal, 6 piezas
- Tuerca autoblocante acero inoxidable A2 DIN-985 M-3, 24 piezas
- Tuerca hexagonal DIN934-M3 acero inox(d), 24 piezas
- Tornillo de M3 Phillips, 24 piezas

- Tornillo M4 Phillips, 6 piezas
- Arandela Grove M4, 6 piezas
- Tuerca M6 hexagonal, 24 piezas
- Varillas roscadas, 6 piezas

Figura 16. Tornillos y tuercas M3 Phillips [9]

3.3 Componentes electrónicos

En este proyecto se han utilizado los siguientes componentes electrónicos:

- El MG996R es un servomotor de corriente continua (DC) con engranajes metálicos, que ofrece un rango de rotación de aproximadamente 120 grados. A 6.0V, su velocidad es de 0.15 segundos/60 grados y proporciona un par de torsión de 11 kg/cm. Sus dimensiones son 40.7 x 19.7 x 42.9 mm y su peso es de aproximadamente 55 gramos. Ideal para aplicaciones de robótica y modelismo, destaca por su durabilidad y precisión.

Figura 17. Servomotor MG996R [10]

- La placa Arduino Uno es una plataforma de desarrollo basada en el microcontrolador ATmega328P, ideal para proyectos electrónicos interactivos. Tiene 14 pines digitales (6 PWM) y 6 pines analógicos,

permitiendo la conexión de diversos componentes. Su memoria incluye 32 KB de Flash, 2 KB de SRAM y 1 KB de EEPROM. La placa se programa a través del software Arduino IDE utilizando un lenguaje basado en C/C++. Alimentada vía USB o conector externo (7-12V), opera a 16 MHz. Es ampliamente utilizada en prototipos electrónicos, robótica, automatización del hogar y educación por su simplicidad y versatilidad.

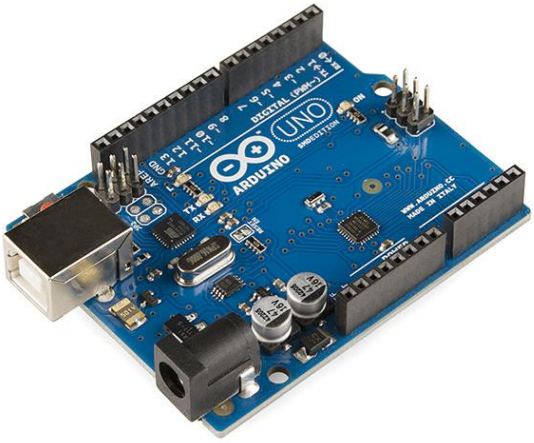


Figura 18. Arduino Uno [11]

- Una fuente de alimentación regulable es un dispositivo que proporciona energía eléctrica ajustable en voltaje y corriente, esencial para laboratorios de electrónica y talleres de reparación. Cuenta con un transformador, rectificador, filtro y regulador de voltaje para convertir y estabilizar la salida de corriente continua (CC). Permite ajustes precisos mediante potenciómetros o botones y ofrece protección contra sobrecargas y cortocircuitos. Su versatilidad la hace ideal para pruebas, desarrollo de prototipos y educación en electrónica, proporcionando una alimentación controlada y segura para diversos dispositivos y circuitos.

Figura 19. Fuente de alimentación regulable [12]

- La Arduino Sensor Shield V5.0 es una extensión para placas Arduino que facilita la conexión de sensores y módulos mediante conectores estándar de 3 pines. Compatible con varias placas Arduino, proporciona acceso a pines digitales, analógicos, PWM e interfaces I2C y serial. Permite una conexión rápida y ordenada, ideal para prototipado rápido, proyectos de robótica y automatización, y aplicaciones educativas. Su diseño modular y versátil elimina la necesidad de cableado complejo, permitiendo a los usuarios enfocarse en el desarrollo y pruebas de sus proyectos.

Figura 20. Arduino Shield V5.0 [13]

4. FUNDAMENTOS TEÓRICOS

En este apartado se observan las diferentes configuraciones de la plataforma de Stewart, además del cálculo de la cinemática inversa, que para robots paralelos

como estos es más sencillo que la cinemática directa. Esto sirve para implementarlo más adelante en un microcontrolador Arduino.

4.1 Arquitectura de la estructura

Como se ha indicado previamente, el diseño inicial de Stewart consistía en una plataforma triangular conectada por articulaciones esféricas a tres actuadores lineales de longitud adaptable, los cuales a su vez se unían a una base fija mediante articulaciones universales [1]. Posteriormente se popularizó una plataforma móvil triangular con actuadores prismáticos coincidiendo de dos en dos en los vértices, conectándose a seis puntos distintos en una base hexagonal [14]. Estas dos configuraciones son conocidas como 3-3 y 6-3 plataformas de Stewart, la 6-3 se mantuvo en auge un tiempo, pero presentaba el problema de que las articulaciones esféricas limitaban la movilidad del manipulador. Por último, la configuración 6-6 (elegida para utilizar en este proyecto) ganó fama, ya que utiliza triángulos equiláteros con sus vértices achatados tanto en la base como en la plataforma móvil. Esta configuración suele ser la preferida ya que su construcción es más sencilla al solo utilizar rótulas simples en cada una de las articulaciones no como la triangular pura que necesita rótulas dobles.

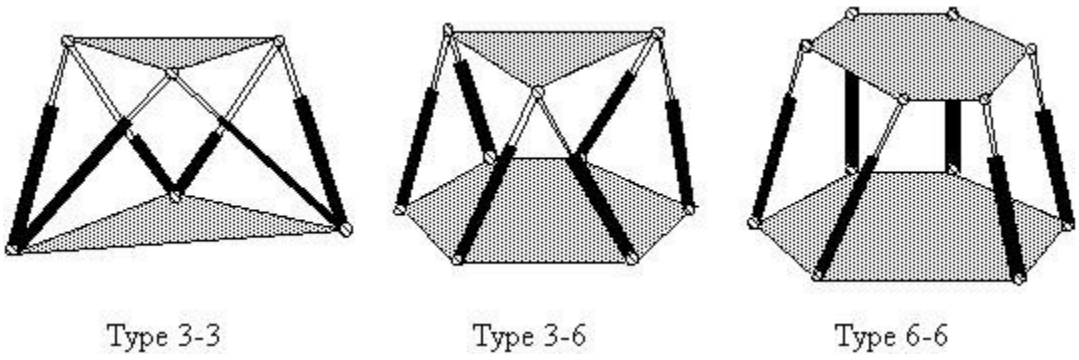


Figura 21. Tipo de arquitectura [14]

4.2 Cálculo de la cinemática inversa

La cinemática de un robot se centra en el estudio de sus movimientos sin considerar las fuerzas que los provocan. En este contexto, se abordan dos problemas principales:

- Cinemática Directa: Determina la posición y orientación del efector final del robot con respecto a un sistema de referencia, basándose en las medidas de las articulaciones y las características geométricas del robot.

- Cinemática Inversa: Establece la configuración necesaria del robot para alcanzar una posición y orientación específicas del efector final.

En el caso de la plataforma de Stewart, el enfoque principal es la cinemática inversa, ya que es la que permite que el robot se mueva como se desea. Para resolver este problema, se ha recurrido al trabajo de Nieves, D. de 2020, donde se detalla el desarrollo completo para abordar la cinemática inversa [15].

4.3 Explicación del código Matlab

En este apartado se explica el código de MATLAB que genera la cinemática inversa, desglosándolo parte por parte para facilitar su comprensión y destacar los aspectos más importantes de su funcionamiento. En cada sección, se analizarán las funciones y procedimientos utilizados, así como su propósito dentro del código completo. Cabe mencionar que el código correspondiente a Arduino se encuentra detallado en los anexos.

4.3.1 Medidas iniciales e iniciación de la función Stewart

En primer lugar, en FreeCAD se calculan los ángulos de los puntos de conexión entre la base y la plataforma (tanto en la plataforma móvil como en la base) con respecto al eje X. Este eje (señalado en rojo en las figuras 21, 22, 23 y24) pasa por el medio de la plataforma móvil o de la base y se toma como eje de referencia.

Hay seis servomotores, por lo que también hay seis medidas de ángulos diferentes. En la plataforma móvil las rótulas se numeran empezando por el de 78,87º (ver figuras 21 y 22) y continúa en sentido de las agujas del reloj. En la base, se sigue el mismo orden empezando por el ángulo de 71,63º (ver figuras 23 y 24).

Además, se obtienen las medidas del radio de la plataforma móvil (figura 21) y de la base (figura 23), para después calcular los puntos Bi y Pi que se indicarán más adelante.

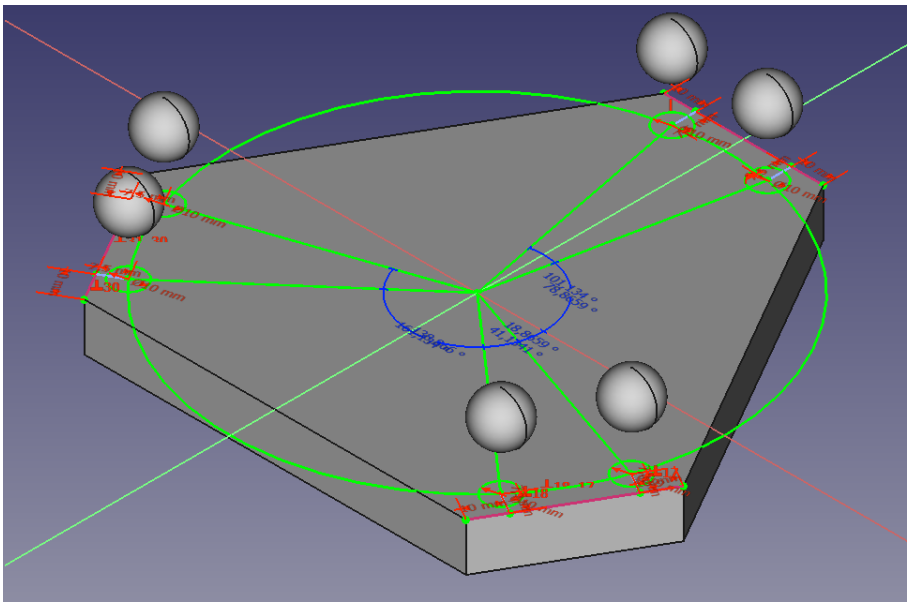


Figura 22. Plano para los ángulos y diámetro de la plataforma

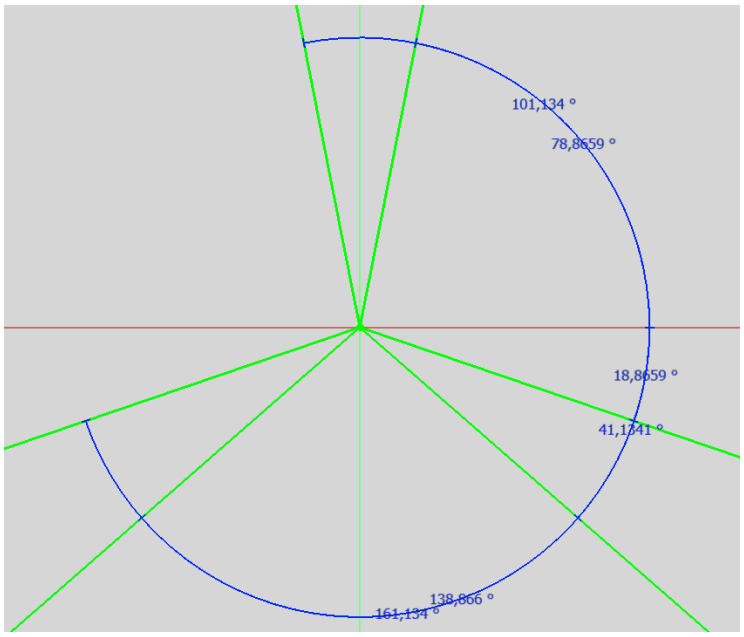


Figura 23. Medida de los ángulos de la plataforma

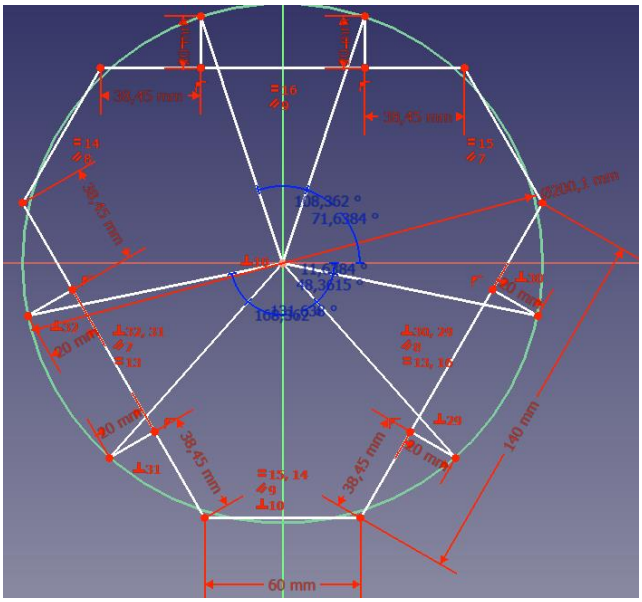


Figura 24. Plano de las posiciones de la base de los servomotores

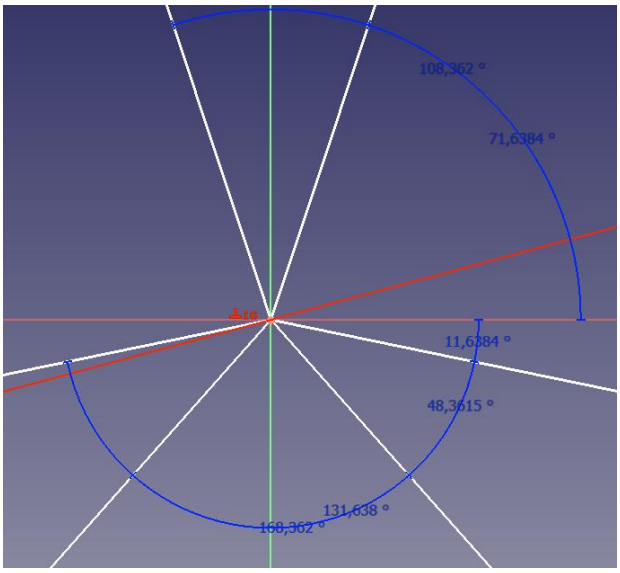


Figura 25. Medida de los ángulos de la base

En la figura 25 se observan las medidas y posiciones iniciales del conjunto de la plataforma que se han hallado en FreeCAD (ANG_BASE, ANG_PLAT, R_BASE y R_PLAT), lo cual hay que tener en cuenta para resolver la cinemática inversa.

La altura inicial (H0) es la distancia tomada con la posición del brazo del servo en 90º hasta punto del centro de la esfera de la plataforma. La longitud de las varillas (L_VAR) y la longitud de los brazos (L_BRZ) fueron tomadas directamente de la medición de esas piezas.

Para explicar la rotación de los ángulos de cada servo vamos a ayudarnos de un esquema donde se refleja la diferencia de 180º de cada uno:

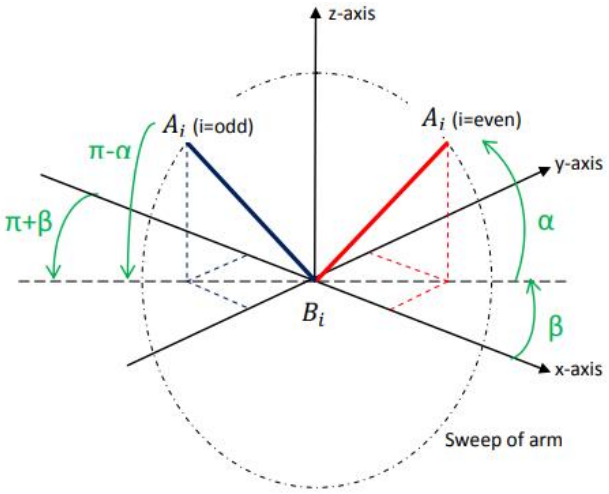


Figura 26. Diferencia de los ángulos con respecto al eje x

En la figura 26 se observa que la línea dibujada en color rojo comprende la medida del brazo del servo que tiene un ángulo con respecto al eje x de valor β, ocurre lo mismo con la línea dibujada en color azul pero el ángulo que forma con el eje x hay que sumarle el valor de π que en grados es una diferencia de 180º, de aquí viene que cada pareja de servo motores tenga una diferencia entre ellos de 180º.

```
%%% SIMULACIÓN PLATAFORMA STEWART

function [ang] = f_stewart(x,y,z,alphag,thetag,psig)
%% GEOMETRÍA INICIAL

% Rotación de los puntos de unión de la base
ANG_BASE=(pi/180)*[71.63 348.37 311.64 228.36 191.64 108.36];

% Rotación de los puntos de unión de la plataforma
ANG_PLAT=(pi/180)*[78.87 341.13 318.87 221.13 198.87 101.13];

% Distancia del centro de la base a los puntos de unión de la misma en mm
R_BASE=100.05;

% Distancia del centro de la plataforma a los puntos de unión de la misma en mm
R_PLAT=77.68;

% Altura inicial de la plataforma en mm
H_0=130;

% Longitud de las varillas en mm
L_VAR=134;

% Longitud de los brazos de los servos en mm
L_BRZ=25;

% Rotación de los brazos de cada servo
beta = (pi/180)*[180 240 60 120 300 0];
```

Figura 27. Geometría inicial de la plataforma de Stewart

A continuación, se indican los sistemas de coordenadas (figura 26) utilizados.

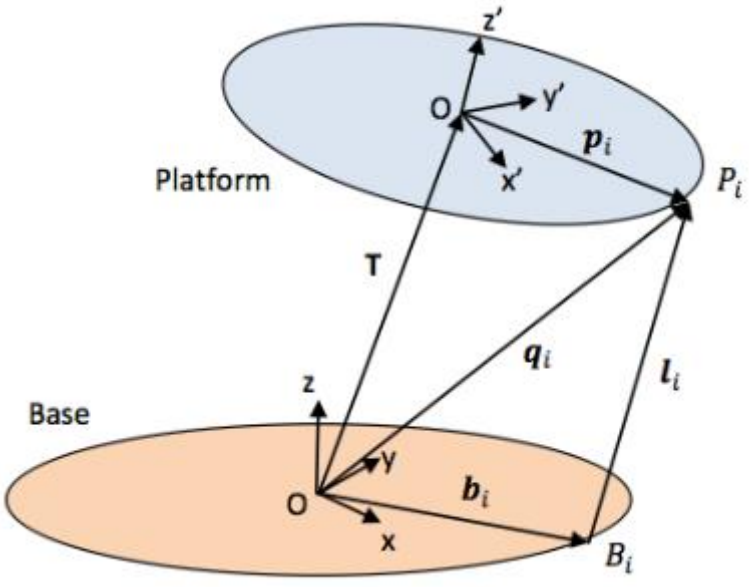


Figura 28. Sistema de coordenadas

La cadena cinemática del mecanismo se explica del siguiente modo: la letra T corresponde con el vector de traslación, que une el origen de la base (O) con el de la plataforma (O), el vector bi define la distancia entre el punto final (Bi) y el inicial (O). Del mismo modo, el vector pi define el punto Pi en el sistema de coordenadas de la plataforma móvil. El vector qi es la distancia entre el punto Pi y el origen de la base O. Por último, el vector li responde a la primera parte de la solución de la cinemática inversa, y su significado detallado se explica a continuación.

```
% Vectores que almacenan las posiciones x, y, z de los puntos de unión de la base
Bx_b = [0 0 0 0 0 0];
By_b = [0 0 0 0 0 0];
Bz_b = [0 0 0 0 0 0];

% Vectores que almacenan las posiciones x, y, z de los puntos de unión de la plataforma
Px_p = [0 0 0 0 0 0];
Py_p = [0 0 0 0 0 0];
Pz_p = [0 0 0 0 0 0];

for i=1:6
  Bx_b(i) = R_BASE*cos(ANG_BASE(i));
  By_b(i) = R_BASE*sin(ANG_BASE(i));
  Px_p(i) = R_PLAT*cos(ANG_PLAT(i));
  Py_p(i) = R_PLAT*sin(ANG_PLAT(i));
end
```

Figura 29. Posiciones de base y plataforma

A continuación (figura 27), se inicializan a cero los vectores que almacenan las posiciones de la base y de la plataforma. En este prototipo, solo se calculan x e y porque los puntos se encuentran en el plano xy. Luego se ejecuta un bucle for que calcula las coordenadas x e y de los puntos Bi y Pi teniendo en cuenta los ángulos donde estan posicionados los motores y los links. Hay que indicar que hay un punto Bi y Pi para las posiciones de cada servo (Bi1, Bi2, Bi3, … Pi6).

```
%% CÁLCULO LONGITUD ENTRE P Y B
% Vector de traslación de la plataforma
T=[x;y;(z)+H_0]
alpha=alphag*pi/180;
theta=thetag*pi/180;
psi=psig*pi/180;

% Matriz de rotación de la plataforma
R=[cos(alpha)*cos(theta), -sin(alpha)*cos(psi)+cos(alpha)*sin(theta)*sin(psi), sin(alpha)*sin(psi)+cos(alpha)*sin(theta)*cos(psi);
sin(alpha)*cos(theta), cos(alpha)*cos(psi)+sin(alpha)*sin(theta)*sin(psi) -cos(alpha)*sin(psi)+sin(alpha)*sin(theta)*cos(psi);
-sin(theta), cos(theta)*sin(psi), cos(theta)*cos(psi)]
% Matriz que almacena las posiciones x,y,z de los puntos de union de la plataforma en el sistema de coordenadas de la base
Q=zeros(3,6);
for j=1:6
  Q(1,j)=T(1)+R(1,1)*Px_p(j)+R(1,2)*Py_p(j)+R(1,3)*Pz_p(j);
  Q(2,j)=T(2)+R(2,1)*Px_p(j)+R(2,2)*Py_p(j)+R(2,3)*Pz_p(j);
  Q(3,j)=T(3)+R(3,1)*Px_p(j)+R(3,2)*Py_p(j)+R(3,3)*Pz_p(j);
end
% Matriz que almacena las posiciones x,y,z de los puntos de union de los brazos de los servos
L=zeros(3,6);
for k=1:6
  L(1,k)=Q(1,k)-Bx_b(k);
  L(2,k)=Q(2,k)-By_b(k);
  L(3,k)=Q(3,k)-Bz_b(k);
end
% Vector que almacena la distancia relativa de cada punto de unión de la base a su respectivo punto de la plataforma
longitud=[0 0 0 0 0 0];

for q=1:6
  longitud(q)=sqrt(L(1,q)^2+L(2,q)^2+L(3,q)^2);
end

longitud
```

Figura 30. Cálculo de la longitud L

Dado que ya conocemos la ubicación de todos los elementos en el sistema de coordenadas con origen en el centro de la base, en el código (figura 28) se refleja el cálculo de la longitud final (li) necesaria entre cada punto de conexión entre la base y su correspondiente punto de sujeción en la plataforma móvil. Esta distancia se calcula desde la posición donde está acoplado el motor en la base hasta su punto de sujeción en la plataforma. Esto se puede expresar con la siguiente fórmula:

$$li = T + {}^{P}R_b \cdot Pi - Bi$$

La fórmula se puede ver desarrollada en el artículo ya mencionado anteriormente [15], en este primero se calcula el vector de traslación y la matriz de rotación.

Debido a que las posiciones de los puntos de acoplamiento y sujeción son diferentes para cada actuador, usamos bucles “for” en el código para iterar a través de cada uno de estos puntos. Esto nos permite calcular la longitud final de cada servomotor de manera individual.

Cada iteración del bucle calcula la longitud correspondiente y la guarda en un vector llamado “longitud”, que contiene todas las longitudes finales necesarias para los servomotores. Este vector (li) es fundamental para el control preciso de

los actuadores, ya que proporciona las posiciones exactas que cada motor debe alcanzar para mover la plataforma a la posición y orientación deseadas.

```
%% CÁLCULO ÁNGULO SERVOS
a=[0 0 0 0 0 0];
b=[0 0 0 0 0 0];
c=[0 0 0 0 0 0];
d=[0 0 0 0 0 0];
ang=[0 0 0 0 0 0];
for w=1:6
  a(w)=2*L_BRZ*(L(3,w));
  b(w)=2*L_BRZ*(sin(beta(w))*L(2,w)+cos(beta(w))*L(1,w));
  c(w)=longitud(w)^2-L_VAR^2+L_BRZ^2;
  d(w)=c(w)/sqrt(a(w)^2+b(w)^2);
if d(w)<-1
  d(w)=-1;
elseif d(w)>1
  d(w)=1;
end

ang(w)=asin(d(w))-atan(b(w)/a(w));

%% Saturacion si sale un angulo mayor
if ang(w)>45*(pi/180)
  ang(w)=45*(pi/180);
elseif ang(w)<-45*(pi/180)
  ang(w)=-45*(pi/180);
end
end

ang*180/pi
```

Figura 31. Cálculo del ángulo Alpha

Por último, en la figura 31 se calcula el ángulo alpha que tiene que girar el servomotor para responder a la primera parte de la cinemática inversa.

Para calcular el ángulo Alpha se utiliza el triangulo del motor, tal como se muestra en la figura 32, donde A es la longitud del brazo del motor, S la longitud del eje que une las plataformas, l la longitud deseada calculada por la cinemática inversa, P el punto de anclaje a la plataforma y B el punto de anclaje al eje del motor:

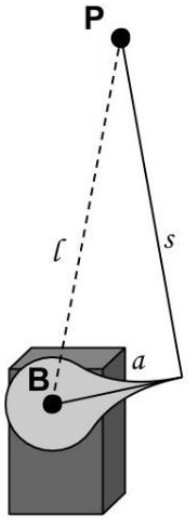


Figura 32. Representación de los elementos usados para la cinemática inversa

Habiendo realizado esto la posición (x,y,z) que recorre el extremo de la biela del servo (punto a) viene marcada por las ecuaciones:

$$x\_a = x\_b + A\cos(\alpha)\cos(\beta)$$

$$y\_a = y\_b + A\cos(\alpha)\,\mathrm{sen}(\beta)$$

$$z\_a = z\_b + A\,\mathrm{sen}(\alpha)$$

combinando las ecuaciones de la distancia entre los puntos, donde

$$A^2 = (x\_a - x\_b)^2 + (y\_a - y\_b)^2 + (z\_a - z\_b)^2$$

$$S^2 = (x\_p - x\_a)^2 + (y\_p - y\_a)^2 + (z\_p - z\_a)^2$$

Conocemos (x_p,y_p,z_p), (x_b,y_b,z_b), el ángulo beta, con lo que queda un sistema de 5 ecuaciones con 5 incógnitas que permite resolver el ángulo deseado alfa.

Resolviendo el sistema de ecuaciones se obtiene:

$$\alpha = sen^{-1}\left(\frac{c}{\sqrt{a^2+b^2}}\right) - tan^{-1}\left(\frac{b}{a}\right)$$

Siendo:

$$a = 2 * l\_brazo * (z\_p - z\_b)$$

$$b = 2 * l\_brazo\ [\ cos(\beta) * (x\_p - x\_b) + sen(\beta) * (y\_p - y\_b)]$$

$$c = l^2 - (s^2 - a^2)$$

Después simplemente se limita el ángulo mínimo y máximo que comprende entre unos -85º y 85º para por último pasarlo a grados.

5. PROGRAMACIÓN EN ARDUINO

En este apartado se explica la comunicación serial de Arduino y como se ha aplicado en nuestro código.

La comunicación serial de Arduino es una forma de comunicación entre dispositivos (en este caso ordenador y placa de Arduino), utilizando una línea de datos para enviar y recibir información bit a bit [17]. En Arduino, esta comunicación se gestiona a través del puerto serie hardware y/o software, y la biblioteca Serial proporciona una interfaz fácil de usar para interactuar con este puerto.

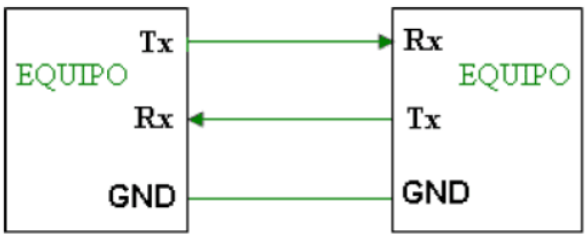


Figura 33. Líneas de comunicación serial

En la figura 31 se observan las 3 únicas líneas que se utilizan para la comunicación, siendo Tx la que envía, que se asocia con el pin 1. Y Rx la que recibe, que se asocia con el pin 0.

Para inicializar la comunicación se utiliza el comando serial.begin(). Este comando establece la velocidad de los baudios y establece que la placa de Arduino está preparada para recibir y enviar información.

```
//definir caracteres usados para el control de la comunicación serial [2-3-8]

#define SETPOSITIONS 50 // 2
#define PRINTPOS 51 // 3
#define GEPOSITION 56 //8


void setup() {
  Serial.begin(9600); // Inicialización del puerto serie para depuración del código
  Serial.print("inicializando");
```

Figura 34. Inicio del código

En las tres primeras líneas de código se definen los números con los que se van a hacer esas funciones. Están en numeración ASCI siendo el 52 el valor 2.

Las tres siguientes inicializamos la comunicación serial y además imprimimos por pantalla la inicialización dentro de la parte del void setup.

```
void loop()
{


  if(Serial.available()>0){
    int input=Serial.read();
    Serial.print("Seleccionado: ");
    Serial.println(input-48);
    delay(10);

    Serial.read();

    switch(input){

      //action to change position of platform, obtain 6 values representing desired position
      case GEPOSITION:
        retPos();
        break;

     case 51:
        Serial.println("Pos servos");
        for (int i=0; i < 6; i++) {
          Serial.print(ang[i]);
          Serial.print(" ");
        }
        Serial.println();


        break;
```

Figura 35. Bucle loop y switch

En la primera parte del código comprobamos si hay datos en el puerto de serie utilizando el comando “Serial.avaible() > 0”, en la línea siguiente lee un byte de datos del puerto serie.

En las dos líneas que se utilizan los serial imprime el valor leído menos 48, que es para convertir el carácter ASCII del número a su valor entero. (Por ejemplo, '0' en ASCII es 48).

En el GETPOSITION, llama a la función retPos para obtener la posición de la plataforma.

En cuanto al case 51, esta muestra las posiciones actuales de los servos almacenados en el array “ang”.

```
case SETPOSITIONS:
    Serial.println("Introduce x, y, z, yaw, pitch, roll, en mm y grados 3 cifras");
    char Variables[6] = {'X','Y','Z','Y','P','R'};
    for(int i=0;i<6;i++){
      int signo = 1;
      long kk;
      while(Serial.available()<1);

      int temp;
      temp = Serial.read();
      if (temp == 45) {
       while(Serial.available()<1);

       temp = Serial.read();
       signo = -1;

      }
      temp = temp-48;
      kk=(long)(temp)*100;
      while(Serial.available()<1);

      temp = Serial.read();
      temp = temp-48;
      kk=kk+(temp)*10;
      while(Serial.available()<1);

      temp = Serial.read();
      temp = temp-48;
      kk=kk+(temp);

      kk = kk*signo;
      while(Serial.available()<1);

      Serial.read();
      if(i<3){
        arr[i]=kk;
      }else{
        arr[i]=(kk);
      }
      Serial.print(Variables[i]);
      Serial.print(": ");
      Serial.println(kk);
    }
    postPlat(arr[0], arr[1], arr[2], arr[3], arr[4], arr[5]);
```

Figura 36. Colocación manual de números de las posiciones deseadas

El caso SETPOSITIONS del código permite al usuario escribir manualmente las posiciones de la plataforma, ingresando las coordenadas x, y, z en milímetros y los ángulos yaw, pitch, roll en grados. Cada valor debe ser una cifra de tres dígitos, y el programa se encarga de leer estos valores del puerto serie, convertirlos a números enteros, y luego aplicar estos valores a la plataforma.

```
void retPos(){
  for(int i=0;i<6;i++){
    long val;
    if(i<3){
      val=(long)(arr[i]);
    }else{
      val=(long)(arr[i]*(180/pi));
    }
    if ( i < 3) {
      Serial.print(arr[i]);
    } else
      Serial.print((180/pi)*arr[i]);
    Serial.print(" ");

  }
  Serial.println();
}
```

Figura 37. Función posición de la plataforma

La última función del código imprime la posición actual de la plataforma, convirtiendo los tres primeros números y mostrándolos a pantalla, los últimos tres valores los pasa de radianes a grados y también los muestra por pantalla.

6. PRESUPUESTO

En el presente trabajo, se ha llevado a cabo el diseño y la implementación de una plataforma de Stewart utilizando herramientas y software accesibles y gratuitos. En particular, se ha empleado FreeCAD para el modelado y diseño de las piezas, y el microcontrolador Arduino para el control y automatización del sistema.

La utilización de estos programas no solo ha permitido mantener los costos bajos, sino que también ha demostrado la viabilidad de desarrollar proyectos complejos sin necesidad de inversiones significativas en software propietario. Este enfoque ha facilitado la realización del proyecto de manera eficiente y económica, aprovechando al máximo los recursos disponibles.

Componente	Precio unitario (€/unidad)	Unidades	Total
Servomotor MG 996R	11,99	6	71,94
Placa Arduino Uno	30,99	1	30,99
Arduino Shield V5.0	7,19	1	7,19
Fuente de alimentación regulable	147,65	1	147,65
Conjunto de tuercas y tornillos	7,56	1	7,56
Cianocrilato	6,20	1	6,20
Fijador de tuercas	6,29	1	6,29
Varilla roscada M6 x 100	0,15	6	0,90
Mano de obra	25 €/h	300	7500
Total			7778,72€

7. CONCLUSIONES

Al finalizar este trabajo y analizar los resultados se han extraído las siguientes conclusiones:

- Se ha completado el diseño e impresión 3D de las diferentes piezas necesarias, utilizando el software FreeCAD. Así como su ensamblaje.
- Se han resuelto las ecuaciones de cinemática inversa para conocer cómo se mueve el motor.
- Se ha logrado ejecutar y controlar correctamente los movimientos del robot mediante su previa programación en Arduino.
- Se ha profundizado y se han puesto en práctica los conocimientos aprendidos durante el grado.
- En definitiva, en este trabajo de fin de grado se ha diseñado e implementado eficazmente un prototipo de plataforma Stewart, un tipo específico de robot paralelo.

At the end of this work and analyzing the results, the following conclusions have been drawn:

- The design and 3D printing of the different necessary parts has been completed, using FreeCAD software. As well as its assembly.
- The inverse kinematics equations have been solved to know how the engine moves.
- The movements of the robot have been correctly executed and controlled through prior programming in Arduino.
- The knowledge learned during the degree has been worked on and these have been put into practice.

All in all, in this final degree project, a prototype of the Stewart platform, a specific type of parallel robot, has been designed and effectively implemented.

8. LIMITACIONES E IDEAS DE MEJORA

El principal obstáculo que presenta la ejecución de este proyecto es el factor económico. La necesidad de emplear servomotores como actuadores en lugar de los prismáticos conlleva que el prototipo sea menos preciso. También el rango de movimiento de la plataforma es menor, por lo que tiene menos utilidades que si tuviera actuadores prismáticos. Por ejemplo, en una de las pruebas de movimiento, al indicar en la placa de Arduino que la plataforma se moviera 30º sobre el eje y, se rompieron las varillas y no se pudo llegar a la posición deseada.

Por tanto, en el caso de ser posible económicamente sería bueno utilizar actuadores prismáticos. Otra opción de mejora es el perfeccionamiento del diseño propuesto para la plataforma, posiblemente utilizando un software de diseño paramétrico más preciso como Solidworks, aunque estos suelen ser de pago.

BIBLIOGRAFÍA

[1] Khalil, W. y Dombre, E. (2002). Chapter 8 - Introduction to geometric and kinematic modeling of parallel robots. En W. Khalil (editor), *Modeling, Identification and Control of Robots* (pp. 171-190). Butterworth-Heinemann.

[2] Duarte, K. y Borrás, C. (2016). Generalidades de robots paralelos. *Visión electrónica, 10(1)*, 1-11. https://www.academia.edu/83503710/Generalidades_de_robots_paralelos?uc-sb-sw=81998371

[3] Stewart, D. (1965). A Platform with Six Degrees of Freedom. *Proceedings of the Institution of Mechanical Engineers, 180(1)*, 371–386. https://journals.sagepub.com/doi/10.1243/PIME_PROC_1965_180_029_02

[4] Caner, C. (s. f). *Basics of the Stewart Platform.* ACROME. https://acrome.net/post/the-basics-of-the-stewart-platform

[5] Aracil, R., Saltarén, R. J., Sabater, J. M., Reinoso, O. (enero, 2006). Robots paralelos: máquinas con un pasado para una robótica del futuro. *Revista Iberoamericana de Automática e Informática Industrial, 3(1),* 16-28. https://polipapers.upv.es/index.php/RIAI/article/view/8105

[6] Caner, C. (s. f). *The use of the Stewart Platform (aka Hexapod) in real life and industry.* ACROME. https://acrome.net/post/the-use-of-the-stewart-platform-in-real-life

[7] Díaz-Rodríguez, M., Quintero-Riaza, H. F., Mejía-Calderon, L. A., Holguin, G., Herrera-López, M., Mesa, C., Daraviña-Peña, G. (2018). Aplicaciones de los robots paralelos. *Manipuladores Paralelos: Síntesis, Análisis y Aplicaciones*. https://hal.science/hal-01907282

[8] Such, D. (agosto, 2023). *Arduino Library for the Stewart Flight Simulator Platform - Part 1.* Medium. https://reefwing.medium.com/arduino-library-for-the-stewart-flight-simulator-platform-part-1-9d1a0e24b5c

[9] Shenzhen Xiny Hei Trading Co., Ltd. (s. f). *Tornillo de cabeza redonda M3 M4 Phillips.* AliExpress. https://es.aliexpress.com/item/32844919854.html

[10] *Servomotor digital MG996R.* (s. f). BricoGeek. https://tienda.bricogeek.com/servomotores/1684-servomotor-digital-mg996r.html

[11] SparkFun Electronics (21 enero 2013). *Arduino Uno – R3.* Wikipedia. https://es.m.wikipedia.org/wiki/Archivo:Arduino_Uno_-_R3.jpg

[12] *HY3005E FUENTE DE ALIMENTACIÓN DIGITAL REGULABLE DE 0-30V/0-5A DE PROSKIT.* (s. f). Eleco-G. https://www.eleco-g.com/fuentes-para-laboratorio/80004423-hy3005e-fuente-de-alimentacion-digital-regulable-de-0-30v0-5a-de-proskit-8436300737926.html

[13] *ARD SHD EXPAN V5.* (S. f). Reichelt elektronik. https://www.reichelt.com/es/es/arduino-shield-placa-de-ampliaci-oacute-n-v5-uno-r3-ard-shd-expan-v5-p282608.html?CCOUNTRY=452&LANGUAGE=es&&r=1

[14] Rueda, J. D. (2008). *Metodología para el diseño de un robot paralelo industrial tipo delta* [Proyecto de grado, Universidad Pontificia Bolivariana]. Repository.upb. https://repository.upb.edu.co/bitstream/handle/20.500.11912/198/digital_16405.pdf

[15] Nieves, D. (2020). *Diseño e implementación de una plataforma Stewart* [Trabajo final de máster, Universidad Politécnica de Valencia]. RiuNet. https://m.riunet.upv.es/bitstream/handle/10251/158625/Nieves%20-%20Dise%C3%B1o%20e%20implementaci%C3%B3n%20de%20una%20Plataforma%20Stewart.pdf?sequence=1&isAllowed=y

[16] Ajna, R. y Hersan, T. (S. f). *Stewart Platform Math.* Memememe. https://memememememememe.me/post/stewart-platform-math/

[17] Castaño, S. A. (S.f). *Comunicación serial con Arduino.* Control Automático Educación. https://controlautomaticoeducacion.com/arduino/comunicacion-serial-con-arduino/#Que_es_la_comunicacion_serial_en_Arduino_y_como_se_utiliza

9. Anexos

8.1 Anexo I: Código Matlab

```
%%% SIMULACIÓN PLATAFORMA STEWART

function [ang] = f_stewart(x,y,z,alphag,thetag,psig)
%% GEOMETRÍA INICIAL

% Rotación de los puntos de unión de la base
ANG_BASE=(pi/180) *[71.63 348.37 311.64 228.36 191.64 108.36];

% Rotación de los puntos de unión de la plataforma
ANG_PLAT=(pi/180) *[78.87 341.13 318.87 221.13 198.87 101.13];

% Distancia del centro de la base a los puntos de unión de la misma en mm
R_BASE=100.05;

% Distancia del centro de la plataforma a los puntos de unión de la misma en
mm
R_PLAT=77.68;

% Altura inicial de la plataforma en mm
H_0=130;

% Longitud de las varillas en mm
L_VAR=134;

% Longitud de los brazos de los servos en mm
L_BRZ=25;

% Rotación de los brazos de cada servo
beta = (pi/180)*[180 240 60 120 300 0];

% Vectores que almacenan las posiciones x, y, z de los puntos de unión de la
base
Bx_b = [0 0 0 0 0 0];
By_b = [0 0 0 0 0 0];
Bz_b = [0 0 0 0 0 0];

% Vectores que almacenan las posiciones x, y, z de los puntos de unión de la
plataforma
Px_p = [0 0 0 0 0 0];
Py_p = [0 0 0 0 0 0];
Pz_p = [0 0 0 0 0 0];

for i=1:6
  Bx_b(i) = R_BASE*cos(ANG_BASE(i));
  By_b(i) = R_BASE*sin(ANG_BASE(i));
  Px_p(i) = R_PLAT*cos(ANG_PLAT(i));
  Py_p(i) = R_PLAT*sin(ANG_PLAT(i));
end

%% CÁLCULO LONGITUD ENTRE P Y B
% Vector de traslación de la plataforma
T=[x;y;(z)+H_0]
alpha=alphag*pi/180;
theta=thetag*pi/180;
psi=psig*pi/180;
```

```
% Matriz de rotación de la plataforma
R=[cos(alpha)*cos(theta), -
sin(alpha)*cos(psi)+cos(alpha)*sin(theta)*sin(psi),
sin(alpha)*sin(psi)+cos(alpha)*sin(theta)*cos(psi);
sin(alpha)*cos(theta), cos(alpha)*cos(psi)+sin(alpha)*sin(theta)*sin(psi) -
cos(alpha)*sin(psi)+sin(alpha)*sin(theta)*cos(psi);
-sin(theta), cos(theta)*sin(psi), cos(theta)*cos(psi)]
% Matriz que almacena las posiciones x,y,z de los puntos de union de la pla-
taforma en el sistema de coordenadas de la base
Q=zeros(3,6);
for j=1:6
  Q(1,j)=T(1)+R(1,1)*Px_p(j)+R(1,2)*Py_p(j)+R(1,3)*Pz_p(j);
  Q(2,j)=T(2)+R(2,1)*Px_p(j)+R(2,2)*Py_p(j)+R(2,3)*Pz_p(j);
  Q(3,j)=T(3)+R(3,1)*Px_p(j)+R(3,2)*Py_p(j)+R(3,3)*Pz_p(j);
end
% Matriz que almacena las posiciones x,y,z de los puntos de union de los bra-
zos de los servos
L=zeros(3,6);
for k=1:6
  L(1,k)=Q(1,k)-Bx_b(k);
  L(2,k)=Q(2,k)-By_b(k);
  L(3,k)=Q(3,k)-Bz_b(k);
end
% Vector que almacena la distancia relativa de cada punto de unión de la base
a su respectivo punto de la plataforma
longitud=[0 0 0 0 0 0];

for q=1:6
  longitud(q)=sqrt(L(1,q)^2+L(2,q)^2+L(3,q)^2);
end

longitud

%% CÁLCULO ÁNGULO SERVOS
a=[0 0 0 0 0 0];
b=[0 0 0 0 0 0];
c=[0 0 0 0 0 0];
d=[0 0 0 0 0 0];
ang=[0 0 0 0 0 0];
for w=1:6
  a(w)=2*L_BRZ*(L(3,w));
  b(w)=2*L_BRZ*(sin(beta(w))*L(2,w)+cos(beta(w))*L(1,w));
  c(w)=longitud(w)^2-L_VAR^2+L_BRZ^2;
  d(w)=c(w)/sqrt(a(w)^2+b(w)^2);
if d(w)<-1
  d(w)=-1;
elseif d(w)>1
  d(w)=1;
end

ang(w)=asin(d(w))-atan(b(w)/a(w));

%% Saturacion si sale un angulo mayor
if ang(w)>45*(pi/180)
  ang(w)=45*(pi/180);
elseif ang(w)<-45*(pi/180)
  ang(w)=-45*(pi/180);
end
end
```

```
ang*180/pi
```

8.2 Anexo II: Código Arduino

```
#include <Servo.h>

#define pi 3.14159

// Pines de los servos
#define SERVO1_PIN 3
#define SERVO2_PIN 5
#define SERVO3_PIN 6
#define SERVO4_PIN 9
#define SERVO5_PIN 10
#define SERVO6_PIN 11

Servo servo1;
Servo servo2;
Servo servo3;
Servo servo4;
Servo servo5;
Servo servo6;

// Declaración de las variables y datos conocidos
static float ang1, ang2, ang3, ang4, ang5, ang6; // Ángulo de giro de los
servos en grados
const float angBase[6] = {71.63, 348.37, 311.64, 228.36, 191.64, 108.36};
const float angPlat[6] = {78.87, 341.13, 318.87, 221.13, 198.87, 101.13};
const float beta[6] = {180.0, 240.0, 60.0, 120.0, 300.0, 0.0};

const float r_b = 100.05, r_p = 77.68, z_0 = 130, l_varilla = 134,
l_brazo = 25;
```

```
const float ANG_MIN = -45; // Define el ángulo mínimo para el servo
const float ANG_MAX = 45;  // Define el ángulo máximo para el servo
const int td = 100;        // Ejemplo de retardo
const int td1 = 100;       // Ejemplo de retardo

static float arr[6]={0,0.0,0, radians(0),radians(0),radians(0)};

float Xb[6], Yb[6], Zb[6];
float Xp[6], Yp[6], Zp[6];

static float T[3], R[3][3], Q[3][6], L[3][6], longitud[6], a[6], b[6],
c[6], d[6], ang[6];

//definir caracteres usados para el control de la comunicación serial [2-
3-8]

#define SETPOSITIONS 50 // 2
#define PRINTPOS 51 // 3
#define GEPOSITION 56 //8


void setup() {
  Serial.begin(9600); // Inicialización del puerto serie para depuración
del código
  Serial.print("inicializando");

  servo1.attach(SERVO1_PIN);
  servo2.attach(SERVO2_PIN);
  servo3.attach(SERVO3_PIN);
  servo4.attach(SERVO4_PIN);
  servo5.attach(SERVO5_PIN);
  servo6.attach(SERVO6_PIN);

  // Cálculo de las posiciones x e y de la posiciones de la plataforma y
la base
  for (int j = 0; j < 6; j++) {
    Xb[j] = r_b * cos(angBase[j] * pi / 180);
    Yb[j] = r_b * sin(angBase[j] * pi / 180);
    Xp[j] = r_p * cos(angPlat[j] * pi / 180);
    Yp[j] = r_p * sin(angPlat[j] * pi / 180);
  }

   Serial.println("Escribe Comando");
   Serial.println(" 2 fijar una nueva posicion");
   Serial.println(" 3 Posicion de los servos");
   Serial.println(" 8 consultar posicion");
}
```

```
void postPlat(float x, float y, float z, float alpha_grados, float
theta_grados, float psi_grados) {


   Serial.print("Colocando plataforma en: ");
  Serial.print(x);
  Serial.print(" ");
  Serial.print(y);
  Serial.print(" ");
  Serial.print(z);
  Serial.print(" ");
  Serial.print(alpha_grados);
  Serial.print(" ");
  Serial.print(theta_grados);
  Serial.print(" ");
  Serial.print(psi_grados);
  Serial.println();




  // Matriz de traslación
  T[0] = x;
  T[1] = y;
  T[2] = z + z_0;

  // Conversión de grados a radianes
  float alpha = alpha_grados * pi / 180;
  float theta = theta_grados * pi / 180;
  float psi = psi_grados * pi / 180;

  // Cálculo de la matriz de rotación de la plataforma
  R[0][0] = cos(alpha) * cos(theta);
  R[0][1] = -sin(alpha) * cos(psi) + cos(alpha) * sin(theta) * sin(psi);
  R[0][2] = sin(alpha) * sin(psi) + cos(alpha) * sin(theta) * cos(psi);
  R[1][0] = sin(alpha) * cos(theta);
  R[1][1] = cos(alpha) * cos(psi) + sin(alpha) * sin(theta) * sin(psi);
  R[1][2] = -cos(alpha) * sin(psi) + sin(alpha) * sin(theta) * cos(psi);
  R[2][0] = -sin(theta);
  R[2][1] = cos(theta) * sin(psi);
  R[2][2] = cos(theta) * cos(psi);

  // Cálculo de la matriz Q
  for (int j = 0; j < 6; j++) {
    Q[0][j] = T[0] + R[0][0] * Xp[j] + R[0][1] * Yp[j] + R[0][2] * Zp[j];
    Q[1][j] = T[1] + R[1][0] * Xp[j] + R[1][1] * Yp[j] + R[1][2] * Zp[j];
    Q[2][j] = T[2] + R[2][0] * Xp[j] + R[2][1] * Yp[j] + R[2][2] * Zp[j];
  }
```

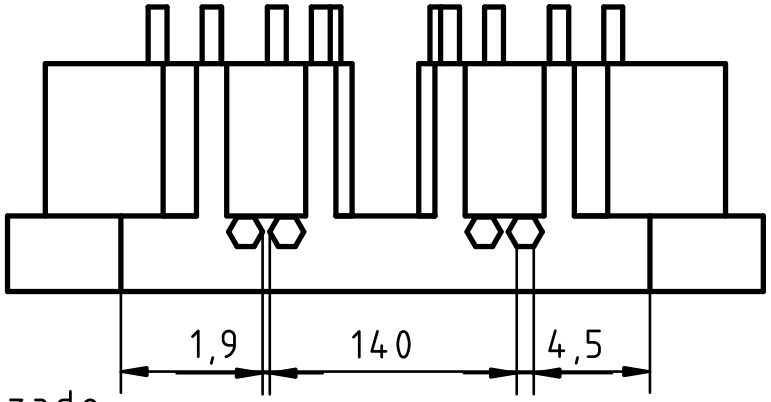
```
  // Cálculo de la matriz L
  for (int j = 0; j < 6; j++) {
    L[0][j] = Q[0][j] - Xb[j];
    L[1][j] = Q[1][j] - Yb[j];
    L[2][j] = Q[2][j] - Zb[j];
  }

  // Cálculo del vector longitud
  for (int j = 0; j < 6; j++) {
    longitud[j] = sqrt(pow(L[0][j], 2) + pow(L[1][j], 2) + pow(L[2][j],
2));
  }

  // Obtención del ángulo de giro de cada servo en radianes
  for (int j = 0; j < 6; j++) {
    a[j] = 2 * l_brazo * L[2][j];
    b[j] = 2 * l_brazo * (sin(beta[j] * pi / 180) * L[1][j] + cos(beta[j]
* pi / 180) * L[0][j]);
    c[j] = pow(longitud[j], 2) - pow(l_varilla, 2) + pow(l_brazo, 2);
    d[j] = constrain(c[j] / sqrt(pow(a[j], 2) + pow(b[j], 2)), -1, 1);

    ang[j] = constrain(asin(d[j]) - atan(b[j] / a[j]), ANG_MIN * pi /
180, ANG_MAX * pi / 180);
  }

  // Conversión del ángulo de giro a grados
  ang1 = ang[0] * 180 / pi;
  ang2 = ang[1] * 180 / pi;
  ang3 = ang[2] * 180 / pi;
  ang4 = ang[3] * 180 / pi;
  ang5 = ang[4] * 180 / pi;
  ang6 = ang[5] * 180 / pi;

  // Envía a los servos el ángulo deseado
  setServo(servo1, 90 - ang1);
  setServo(servo2, 90 + ang2);
  setServo(servo3, 85 - ang3);
  setServo(servo4, 95 + ang4);
  setServo(servo5, 90 - ang5);
  setServo(servo6, 95 + ang6);
}

void setServo(Servo& servo, float angle) {
  servo.write(angle);
}

void loop()
{
```

```
   if(Serial.available()>0){
      int input=Serial.read();
      Serial.print("Seleccionado: ");
      Serial.println(input-48);
      delay(10);

      Serial.read();

      switch(input){

         //action to change position of platform, obtain 6 values repre-
senting desired position
         case GEPOSITION:
            retPos();
            break;

        case 51:
            Serial.println("Pos servos");
            for (int i=0; i < 6; i++) {
              Serial.print(ang[i]);
              Serial.print(" ");
            }
            Serial.println();


            break;

       case SETPOSITIONS:
            Serial.println("Introduce x, y, z, yaw, pitch, roll, en mm y
grados 3 cifras");
            char Variables[6] = {'X','Y','Z','Y','P','R'};
            for(int i=0;i<6;i++){
               int signo = 1;
               long kk;
               while(Serial.available()<1);

               int temp;
               temp = Serial.read();
               if (temp == 45) {
                while(Serial.available()<1);

                temp = Serial.read();
                signo = -1;

               }
               temp = temp-48;
               kk=(long)(temp)*100;
               while(Serial.available()<1);
```

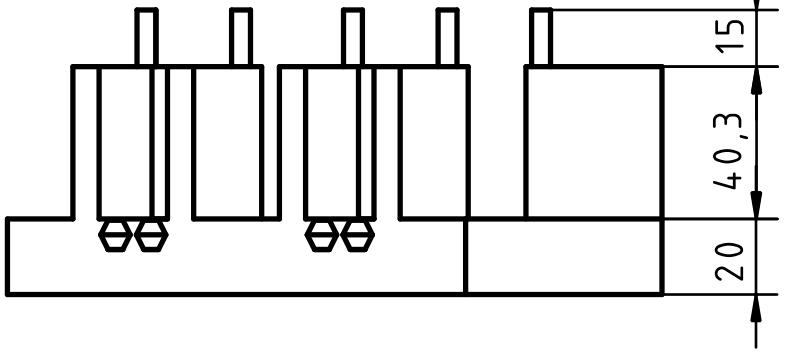
```
                temp = Serial.read();
                temp = temp-48;
                kk=kk+(temp)*10;
                while(Serial.available()<1);

                temp = Serial.read();
                temp = temp-48;
                kk=kk+(temp);

                kk = kk*signo;
                while(Serial.available()<1);

                Serial.read();
                if(i<3){
                   arr[i]=kk;
                }else{
                   arr[i]=(kk);
                }
                Serial.print(Variables[i]);
                Serial.print(": ");
                Serial.println(kk);
             }
             postPlat(arr[0], arr[1], arr[2], arr[3], arr[4], arr[5]);

             break;

          default:
             Serial.print("Default, entrada: ");
             Serial.println(input-48);
             break;
       }
       Serial.println("Escribe Comando");
       Serial.println(" 2 fijar una nueva posicion");
       Serial.println(" 3 Posicion de los servos");
       Serial.println(" 8 consultar posicion");
    }
}
void retPos(){
   for(int i=0;i<6;i++){
       long val;
       if(i<3){
           val=(long)(arr[i]);
       }else{
           val=(long)(arr[i]*(180/pi));
       }
       if ( i < 3) {
          Serial.print(arr[i]);
       } else
```

```
            Serial.print(arr[i]);
        Serial.print(" ");

    }
    Serial.println();
}
```

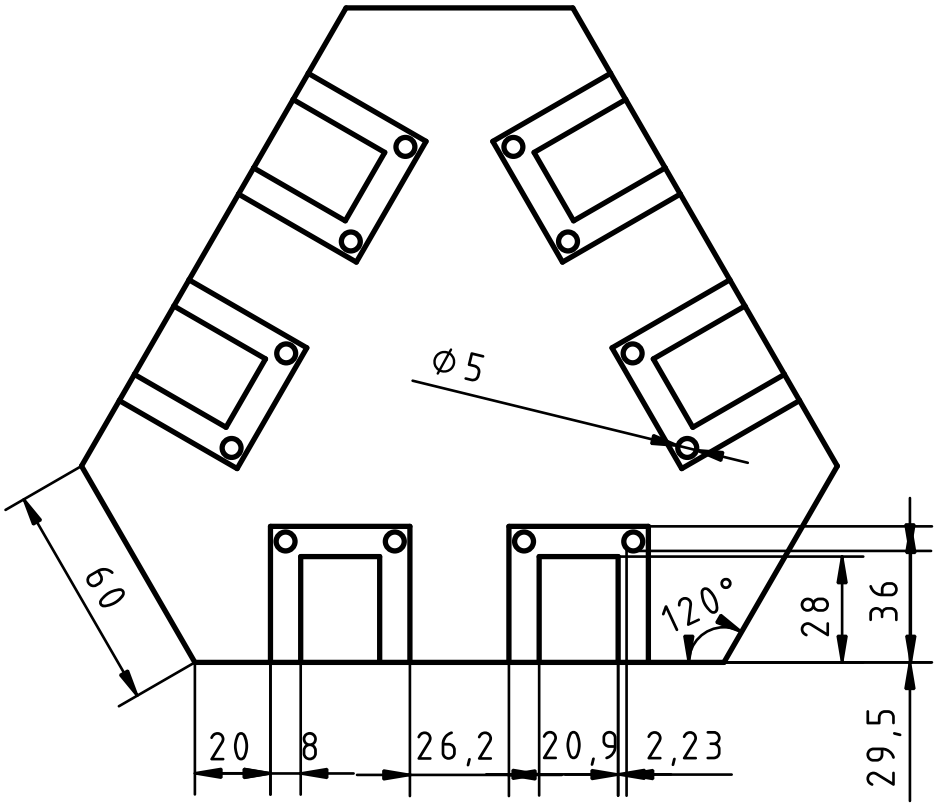
8.3 ANEXO III: Planos



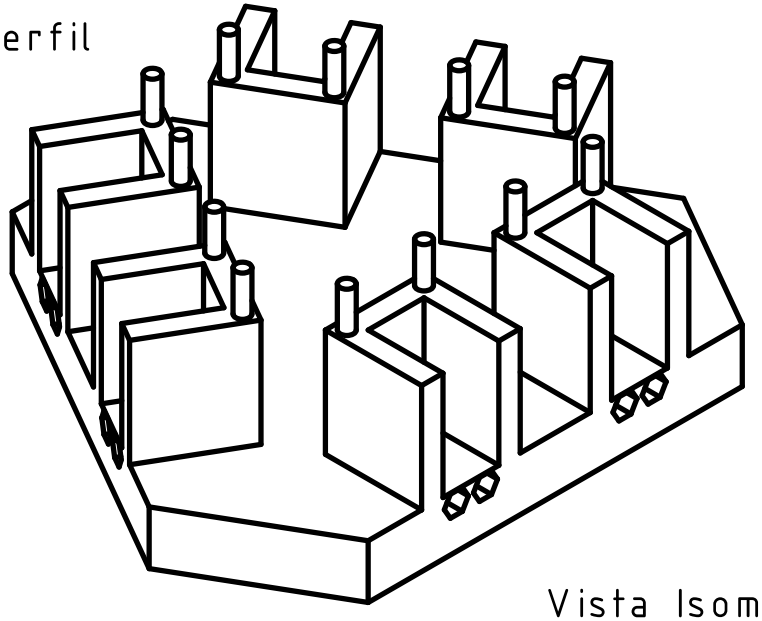
Alzado



Perfil

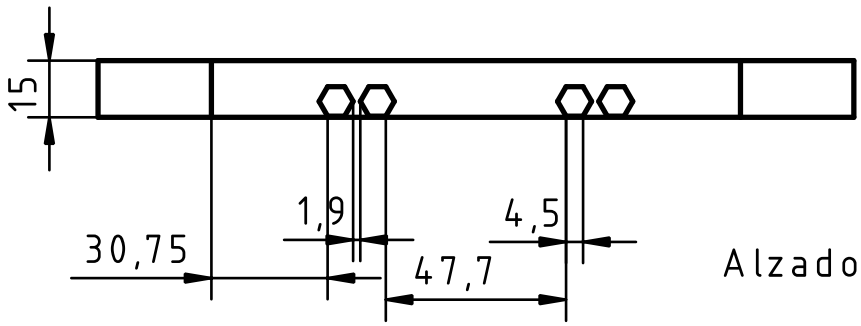


Planta

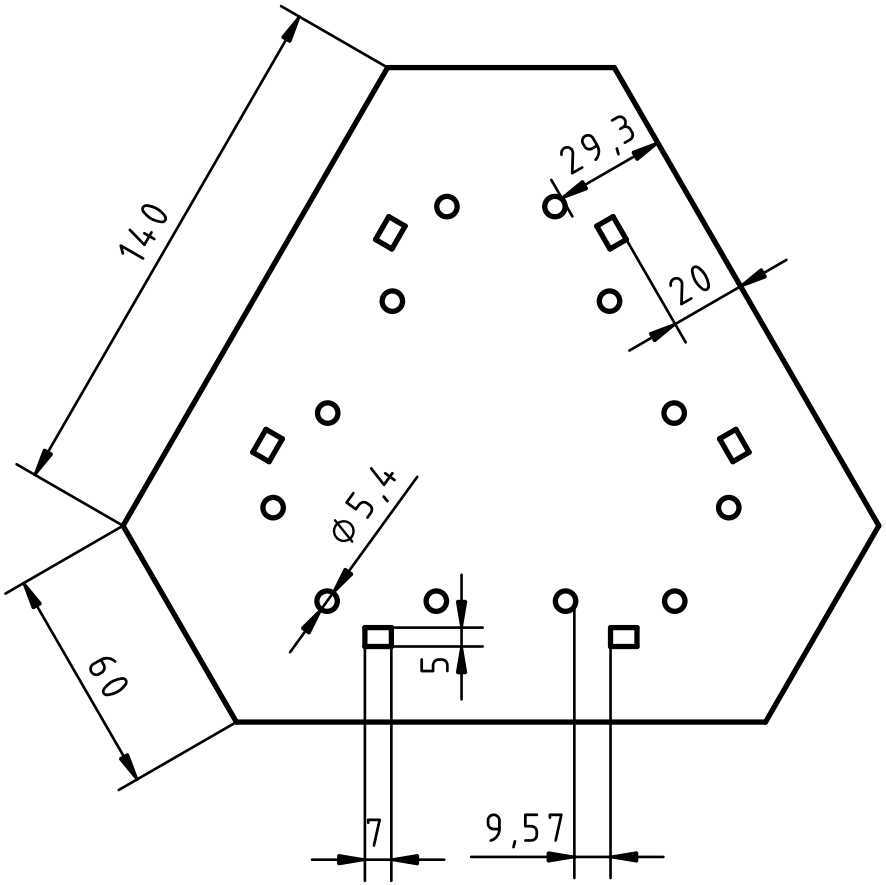


Vista Isométrica

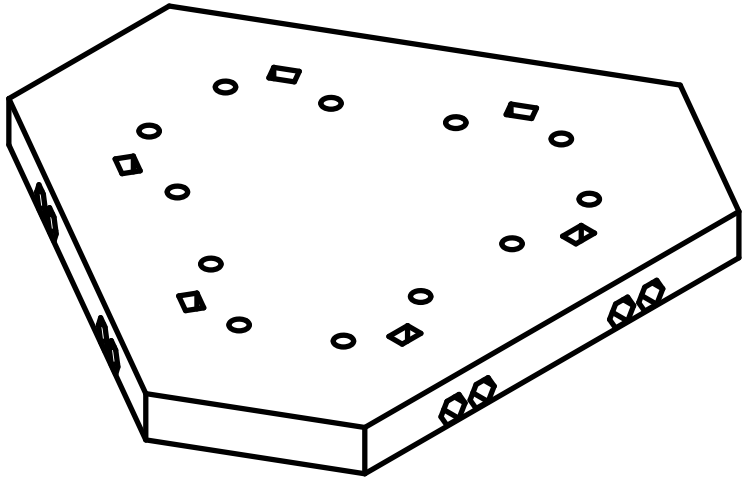
DESIGNED BY: César Pérez DATE: 08/07/2024		Base		
SIZE A4				
SCALE 1:2	WEIGHT (kg)	DRAWING NUMBER Primer Plano	SHEET 1	



Alzado

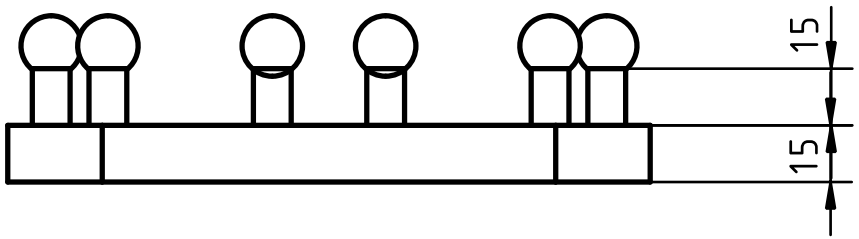


Planta

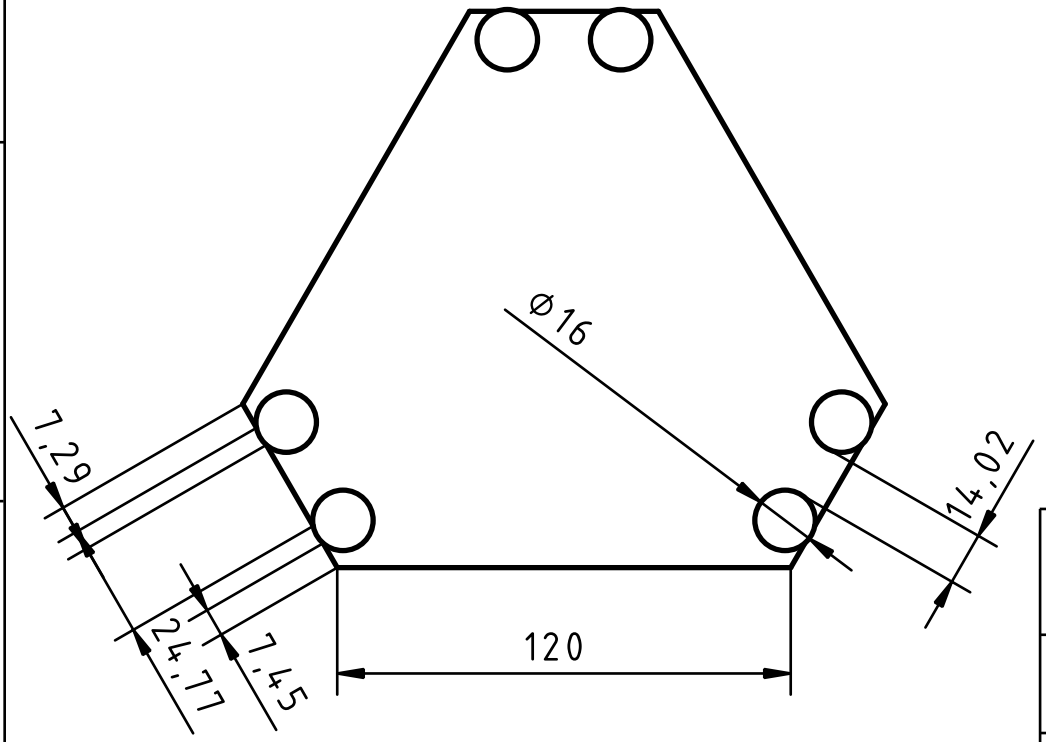


Vista Isométrica

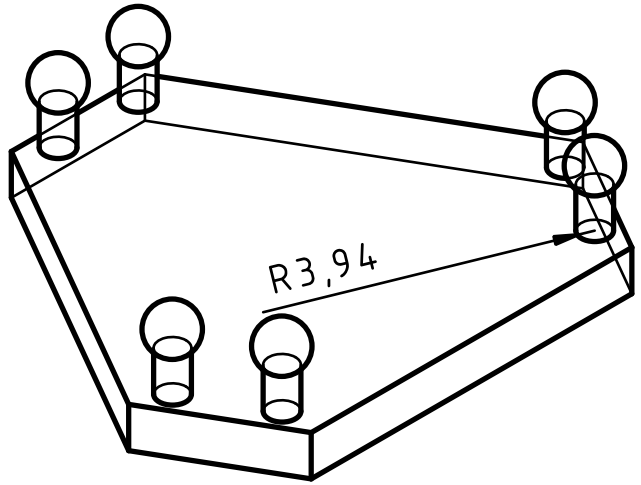
DESIGNED BY: César Pérez DATE: 08/07/2024		Tapa Base		G _
SIZE A4				F _
				E _
				D _
SCALE 1:2	WEIGHT (kg)	DRAWING NUMBER Tercer Plano	SHEET 3	C _
				B _
				A _



Alzado



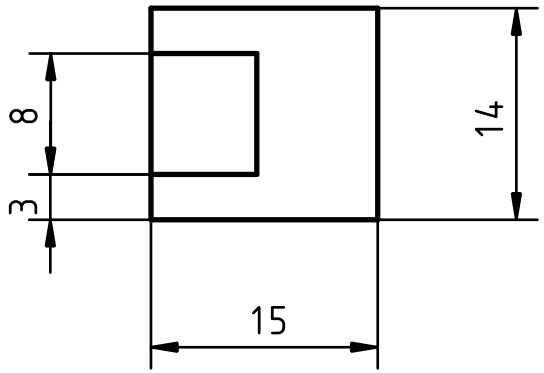
Planta



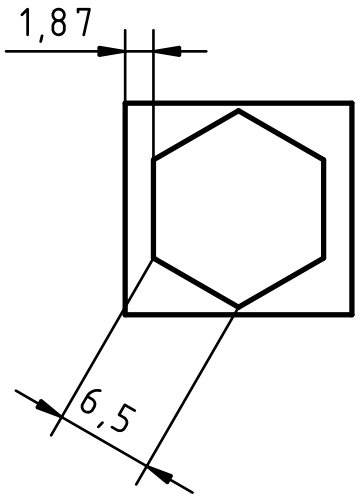
Vista Isométrica

DESIGNED BY: César Pérez DATE: 08/07/2024		Plataforma Móvil	
SIZE A4			
SCALE 1:2	WEIGHT (kg)	DRAWING NUMBER Segundo Plano	SHEET 2

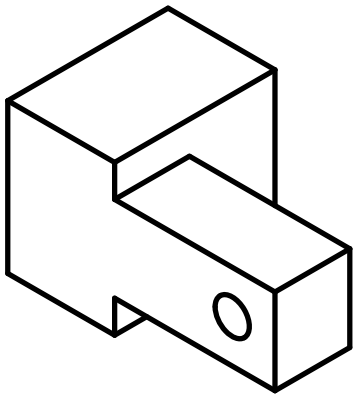
This drawing is our property; it can't be reproduced or communicated without our written consent.



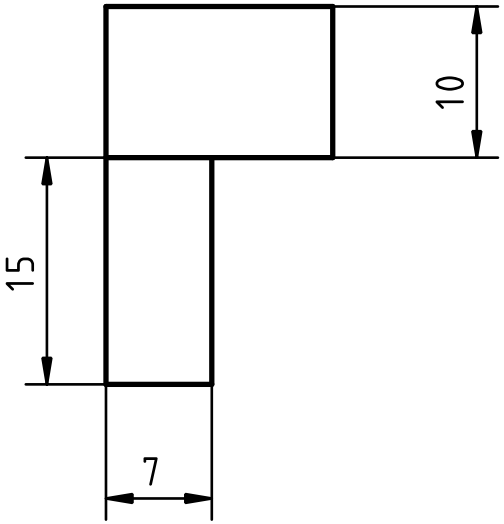
Alzado



Vista Posterior



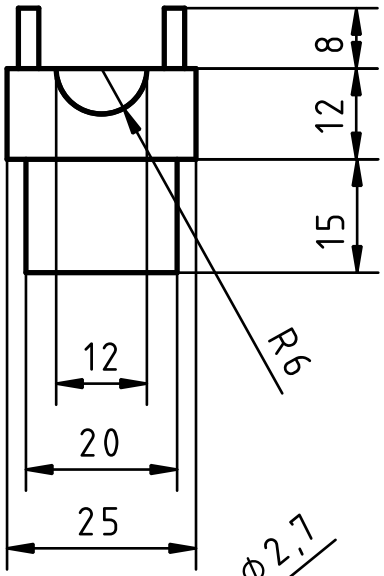
Vista Isométrica

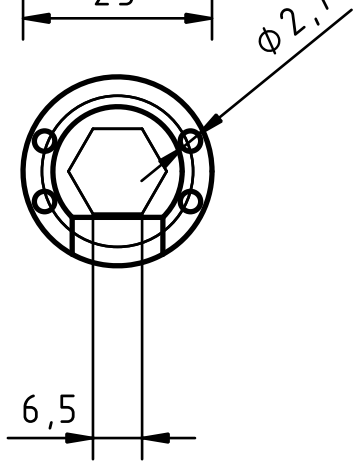


Planta

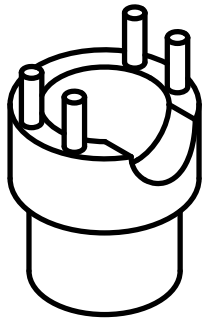
DESIGNED BY: César Pérez DATE: 08/07/2024		Apoyo Inferior Varilla		G _
SIZE A4				F _
				E _
				D _
SCALE 2:1	WEIGHT (kg)	DRAWING NUMBER Quinto Plano	SHEET 5	C _
				B _
This drawing is our property; it can't be reproduced or communicated without our written consent.				A _

Alzado



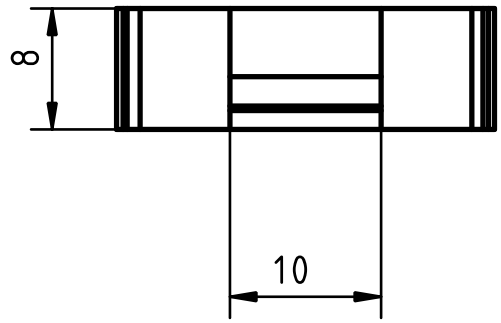


Planta

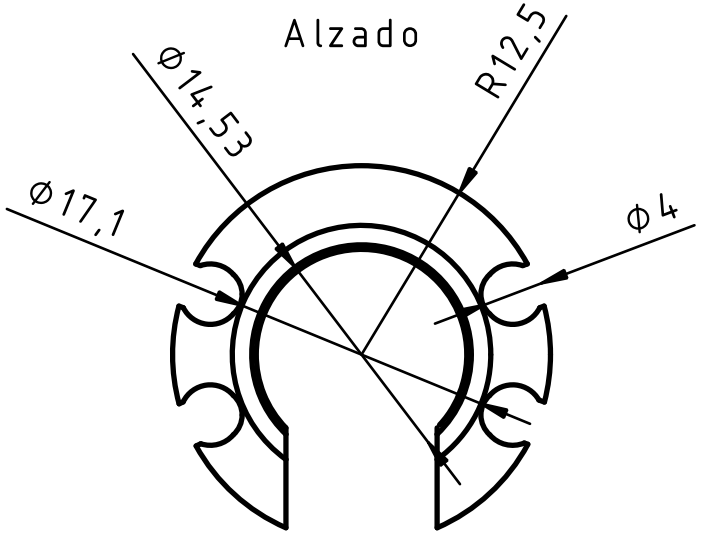


Vista Isométrica

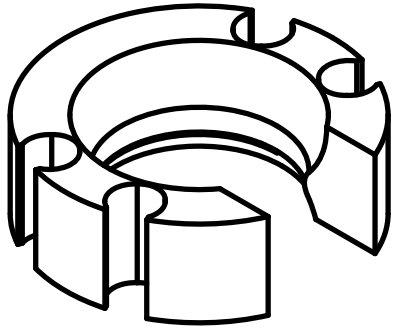
DESIGNED BY: César Pérez DATE: 08/07/2024		Apoyo Superior Varilla	
SIZE A4			
SCALE 1:1	WEIGHT (kg)	DRAWING NUMBER Cuarto Plano	SHEET 4



Alzado



Planta



Vista Isométrica

DESIGNED BY: César Pérez DATE: 08/07/2024		Tapa 855		G _ F _ E _
SIZE A4				D _
SCALE 2:1	WEIGHT (kg)	DRAWING NUMBER Sexto Plano	SHEET 6	C _ B _
This drawing is our property; it can't be reproduced or communicated without our written consent.				A _