UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA STEWART

TRABAJO FINAL DEL
Máster Universitario en Ingeniería Mecatrónica

REALIZADO POR
Diego Nieves Belmar

TUTORIZADO POR
Vicente Fermín Casanova Calvo

CURSO ACADÉMICO: 2019/2020

Resumen

Una Plataforma Stewart es un tipo de robot paralelo ampliamente utilizado en la industria con aplicaciones en todo tipo de operaciones, desde el empaquetado y ensamblaje de componentes hasta el mecanizado de piezas. Más allá del ámbito industrial, resulta común encontrar este tipo de manipuladores siendo utilizados para la docencia y la investigación en centros universitarios y laboratorios médicos. Estos robots paralelos han ido acrecentando con el paso de los años el interés de la comunidad científica debido a su elevada rigidez, su alta precisión de posicionamiento, y su alta capacidad de carga, que los posicionan como una alternativa a los robots serie tradicionales.

En este proyecto, se ha llevado a cabo la implementación de un prototipo de Plataforma Stewart con actuadores giratorios de bajo coste, capaz de realizar movimientos en 6 grados de libertad, y siendo controlada de manera remota a través de un teléfono móvil. A lo largo del proyecto se han realizado paso a paso cada una de las actividades propias del desarrollo de una máquina, desde el diseño de los componentes, la creación de un modelo simplificado y la simulación del movimiento de la máquina utilizando dicho modelo, hasta la fabricación de las piezas diseñadas y su montaje para crear un prototipo real. El prototipo fabricado finalmente ha sido capaz de realizar las acciones deseadas cumpliendo con los objetivos planteados en el proyecto.

Abstract

A Stewart Platform is a kind of parallel robot widely used in the industry with applications in several types of operations, from packaging and components assembly to parts machining. Beyond the industrial world, it is common to find this type of manipulators being used for teaching and research in universities and medical laboratories. These parallel robots have been increasing the interest of the scientific community over the years due to their high rigidity, their high positioning precision, and their high loading capacity, which place them as an alternative to traditional serial robots.

In this project, it has been carried out the implementation of a Stewart Platform prototype with low-cost rotary actuators, capable of performing movements in 6 degrees of freedom, and being remotely controlled through a mobile phone. Throughout the project, the different activities involved in a machine development have been accomplished step by step, from the design of the components, the creation of a simplified model and the simulation of the machine movement using that model, to the manufacture of the designed parts and their assembly to create a real prototype. The manufactured prototype has finally been able to execute the desired actions, fulfilling the objectives set in the project.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

La plataforma Gough-Stewart es un tipo de manipulador paralelo con seis grados de libertad en el cual una plataforma se acopla a una base mediante seis actuadores prismáticos, cada uno de ellos con dos articulaciones universales, esféricas o flexibles en sus extremos. El movimiento de estos actuadores permite a la plataforma, y por lo tanto a un dispositivo que en ella se encuentre, desplazarse y girar en cualquier eje de manera simultánea.

A continuación, se relatan brevemente los antecedentes de este tipo de mecanismo, así como los objetivos y la justificación de este trabajo.

1.1. Antecedentes

En 1965, D. Stewart publica un estudio en el que describe "*un mecanismo que tiene seis grados de libertad, controlado por cualquier combinación de seis motores fijados al suelo*", con el objetivo de ser utilizado en un simulador de vuelo para el entrenamiento de pilotos de helicóptero (Figura 1) [1].

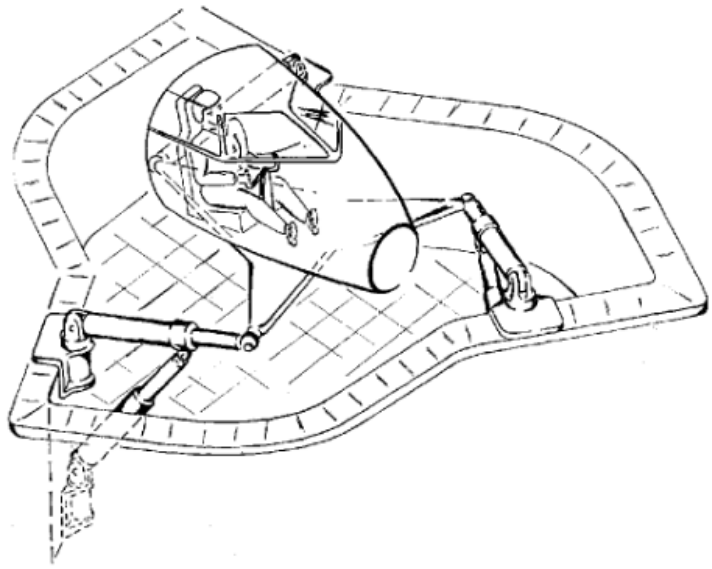


Figura 1. Diseño original del mecanismo por D. Stewart [1].

Dicho mecanismo será conocido a raíz de este artículo como plataforma Stewart, aunque algunos autores recientes se refieren a ella como plataforma Gough-Stewart dando crédito a un diseño anterior de V.E. Gough, el cual sugiere el uso de seis actuadores lineales en paralelo, da acuerdo al diseño que había utilizado en su máquina de testeo de neumáticos (Figura 2) [2].

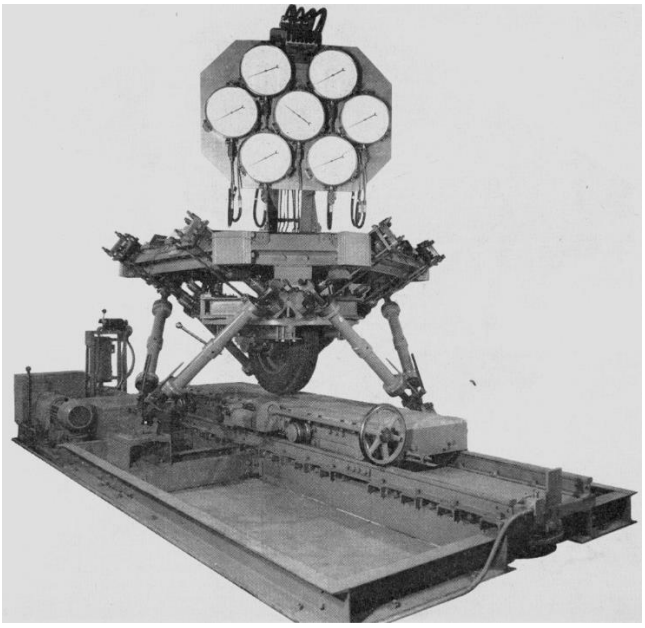


Figura 2. Fotografía de la máquina de testeo de neumáticos de V.E. Gough [1].

Es, no obstante, el trabajo de Stewart el que despierta el interés de la comunidad científica [2], que comienza a ver los manipuladores paralelos en general, y a la plataforma Stewart en concreto, como una alternativa a los robots serie clásicos debido a su elevada rigidez, su alta precisión de posicionamiento y su alta capacidad de carga [4]. Este interés se hace notable sobre todo a partir de los años 80, momento en el que el número de publicaciones relacionadas con este tipo de manipuladores paralelos comienza a aumentar de manera exponencial, acompañando a los continuos avances en el campo de la robótica. Inicialmente, la mayoría de estudios se centran en cuestiones teóricas como la cinemática [5], dinámica [6], la estimación de espacios de trabajo [7] y la generación de trayectorias [8], dejando atrás las amplias posibilidades en cuanto a aplicaciones debido a la falta de herramientas de síntesis racional que permitiesen llevar a cabo diseños prácticos. Sin embargo, el rápido desarrollo en los años siguientes en capacidad computacional y herramientas de diseño asistido por ordenador (CAD) favorecen el aumento de publicaciones relacionadas con las posibles aplicaciones de este tipo de mecanismos [3].

1.1.1. Aplicaciones

Las ya mencionadas ventajas de los manipuladores paralelos con respecto a los robots serie han ido propiciando a lo largo de los años su aparición en diferentes industrias. Hoy en día, es posible encontrar este tipo de mecanismos en un gran número de aplicaciones, tanto teóricas como prácticas, siendo la plataforma Stewart una de las configuraciones más extendidas [9]. Entre estas aplicaciones se encuentran:

- "Pick and place": Es una de las aplicaciones más extendida y desarrollada en la industria actualmente. Muchas de las grandes empresas de robots industriales incorporan en sus catálogos varios robots paralelos de entre 3 y 6 grados de libertad (Figura 3) para su uso en industrias relacionadas con el embalaje, empaquetado, almacenaje, ensamblaje, etc.

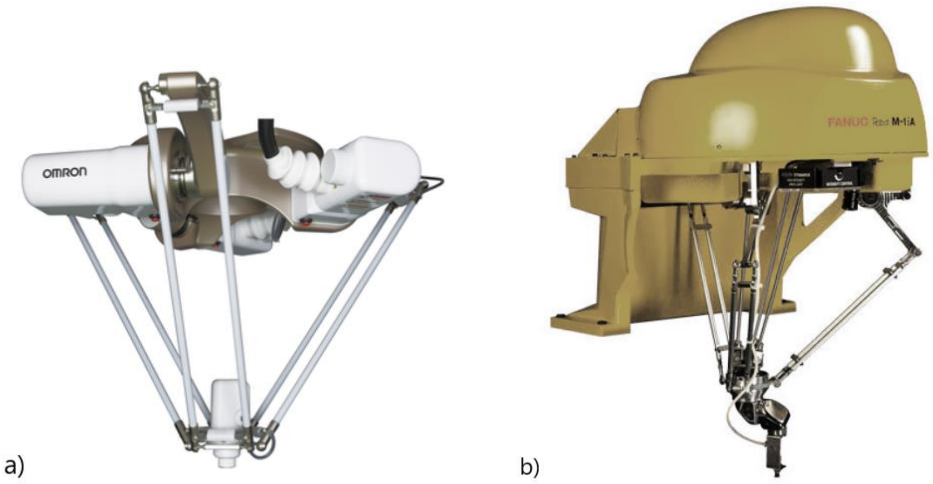


Figura 3. (a) Robot Sysmac Delta de Omron [10] y (b) robot M-1iA/0.5A de Fanuc [11].

- Mecanizado: Otra de las aplicaciones más extendidas de estos robots es en centros de mecanizado y CNC de alta velocidad, donde, para el mecanizado de materiales tradicionales de elevada dureza, los robots paralelos pueden llegar a mejorar las prestaciones de los robots serie.

Figura 4. Centro de mecanizado vertical UA2090Ti de Toyoda [12].

- Industria médica: Aunque en aplicaciones médicas el desarrollo de los manipuladores paralelos no ha llegado prácticamente al nivel comercial, existe un amplio catálogo de prototipos utilizados en laboratorios e institutos de

investigación como robots de asistencia en varios tipos de cirugías, además de un gran número de estudios relativos al desarrollo de robots de cirugía mínimamente invasiva. Más allá de la cirugía, es posible encontrar este tipo de robots para aplicaciones de rehabilitación, principalmente del tobillo, donde sí se puede ver algún prototipo comercial.

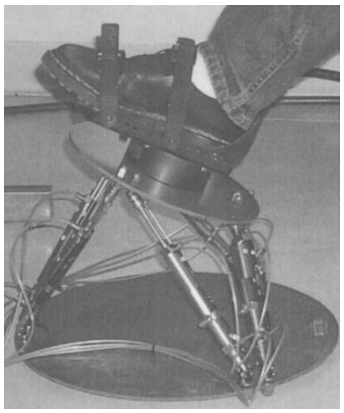
a)  b) 

Figura 5. (a) Balance System SD de Biodex [13] y (b) prototipo para rehabilitación del tobillo basado en la plataforma Stewart [14].

- Otras: Además de las mencionadas y de las aplicaciones "originales" (simulación de movimiento y testeo), es posible encontrar ejemplos de aplicación de la plataforma Stewart en otros campos como el del aislamiento de vibraciones [15] y la sensórica de fuerza [16][17], siempre teniendo en cuenta que, a pesar de tener ciertas ventajas con respecto a los robots serie, también presenta ciertos límites, como son el limitado espacio de trabajo y la complejidad en el cálculo de su cinemática.

1.2. Objeto del trabajo

El objetivo principal de este trabajo es el de diseñar, modelar e implementar un prototipo de Plataforma Stewart de bajo coste mediante la utilización de servomotores en lugar de actuadores prismáticos. Una vez construido el prototipo, se pretende controlar todos los movimientos del mismo utilizando para ello un microcontrolador *Arduino*.

Con este fin, se ha dividido el proyecto en 3 fases:

- Diseño de las diferentes piezas que componen el mecanismo (a excepción de los servomotores) mediante el software de diseño CAD 3D *SolidWorks*. Estos diseños se utilizarán posteriormente para el modelado del mecanismo, además de servir algunos de ellos para la impresión en 3D de ciertas piezas.

- Modelado del mecanismo mediante la herramienta de software matemático Matlab. El modelo cinemático creado permitirá predecir la respuesta de la máquina y la precisión de sus movimientos en condiciones ideales de funcionamiento.

- Implementación de la máquina, abarcando esta fase desde la obtención de los componentes hasta el ensamblaje de estos y la programación del microcontrolador. Se pretende también que la plataforma, una vez construida, pueda ser controlada de forma remota mediante la creación de una aplicación para el teléfono móvil.

1.3. Justificación del trabajo

La elaboración de este trabajo obedece en primer lugar a la necesidad de realizar un proyecto final para la obtención del título del Máster en Ingeniería Mecatrónica, siendo el mismo tutorizado por el profesor D. Vicente Fermín Casanova, perteneciente al Departamento de Ingeniería de Sistemas y Automática y que imparte docencia en la Escuela Técnica Superior de Ingeniería del Diseño perteneciente a la Universidad Politécnica de Valencia.

Con la realización de este trabajo se pretende reunir y aplicar los conocimientos adquiridos durante el curso del Máster en Ingeniería Mecatrónica, ya que guarda relación con diversas asignaturas de su plan de estudios tales como *Dinámica de Sistemas Mecánicos, Robótica, Control Aplicado de Sistemas Mecatrónicos y Sistemas de Medición y Actuación*.

Resulta interesante a su vez pasar por las diferentes fases del desarrollo de una máquina de una manera que se asemeja al desarrollo de una máquina industrial, utilizando herramientas relevantes en el ámbito de la ingeniería como son *SolidWorks, Matlab y Arduino*.

2. FUNDAMENTOS TEÓRICOS

En este apartado se exponen algunos de los conceptos teóricos más relevantes a la hora de llevar a cabo el diseño y el cálculo del mecanismo. Resulta especialmente importante el cálculo del modelo cinemático del robot, puesto que este modelo es utilizado más adelante tanto en la simulación como en la implementación de la plataforma.

2.1. Arquitectura de la plataforma

El mecanismo descrito por Stewart [1] consiste en una plataforma triangular soportada mediante articulaciones esféricas sobre tres patas de longitud ajustable conectadas al suelo mediante articulaciones de dos ejes. Más adelante, se define el concepto general de la Plataforma Stewart como un manipulador paralelo de seis grados de libertad consistente en dos cuerpos rígidos conectados mediante seis restricciones lineales o angulares entre 6 pares de puntos, líneas y/o planos de la base y la plataforma. Esta definición da lugar a más de tres mil arquitecturas o formas posibles para el mecanismo [3], aunque la mayoría de los diseños se centran en una serie de configuraciones en función del número de conexiones entre la base y la plataforma, su situación, el tipo de conexión y el tipo de actuador empleado.

En la Figura 6 se muestra una descripción esquemática de las arquitecturas de Plataforma Stewart más extendidas.

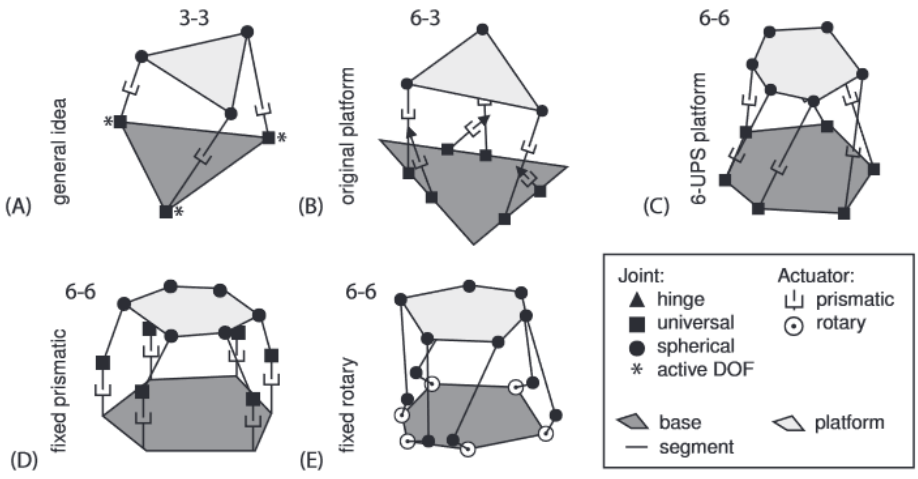


Figura 6. Diferentes arquitecturas de Plataforma Stewart [3]

Para la realización de este trabajo, la configuración seleccionada es la plataforma 6-6 con articulaciones esféricas y actuadores giratorios, ya que ésta puede ser realizada mediante servomotores simples, lo que se traduce en un prototipo más económico y sencillo de controlar.

2.2. Cálculo de la cinemática inversa

Una vez definida la configuración, es necesario determinar la relación existente entre el ángulo de giro de los servomotores y el movimiento de la plataforma. El objetivo es conocer en todo momento la posición y la orientación de la plataforma en función del ángulo de los servos, pues este ángulo es la variable más sencilla de controlar. Esta relación se determina mediante un modelo cinemático.

La cinemática de un robot estudia el movimiento del mismo con respecto a un sistema de referencia fijo, sin considerar las fuerzas y momentos que lo originan. Existen dos problemas fundamentales a resolver en la cinemática del robot:

- El problema cinemático directo, consistente en determinar la posición y orientación del efector final del robot, con respecto a un sistema de coordenadas tomado como referencia, conocidos los valores de las articulaciones y los parámetros geométricos del robot.
- El problema cinemático inverso, consistente en determinar la configuración que debe adoptar el robot para una posición y orientación del efector final conocidas.

Es el problema cinemático inverso el que se resuelve en este trabajo, ya que resulta mas sencillo que el directo dadas las características del mecanismo. En el supuesto caso de una Plataforma Stewart con articulaciones prismáticas, la solución de la cinemática inversa radica en encontrar la longitud de las barras que unen la base con la plataforma para la posición requerida de esta última. Esta longitud está definida por el vector l_k ($k \in \{1 \ldots 6\}$) que se muestra a continuación en la Figura 7.

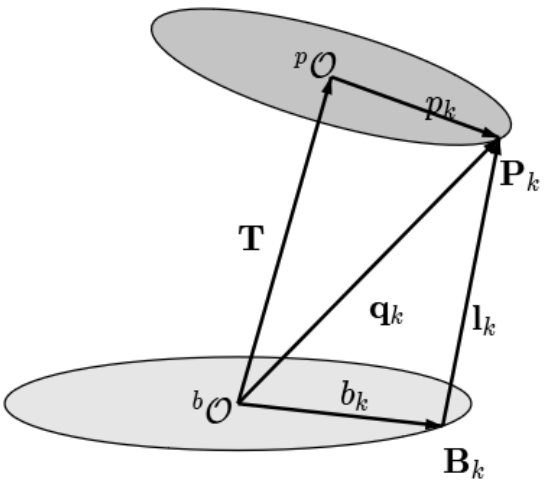


Figura 7. Representación vectorial de la Plataforma Stewart [18]

Además de la longitud, en la figura se muestran los elementos que forman la cadena cinemática aplicable a todas las barras del mecanismo, siendo T el vector de traslación del origen de coordenadas de la plataforma (^pO) con respecto del

origen de coordenadas de la base (${}^{b}O$); B_k y P_k los puntos correspondientes a las articulaciones que unen la barra con la base y la plataforma; b_k el vector que define el punto B_k en el sistema de coordenadas de la base; p_k el vector que define el punto P_k en el sistema de coordenadas de la plataforma; y q_k el vector que define el punto P_k en el sistema de coordenadas de la base.

Así, de la Figura 7 se obtiene que

$$q_k = T + {}^{p}R_b \cdot p_k \tag{1}$$

$$\boldsymbol{l_k} = q_k - b_k = \boldsymbol{T} + {}^{\boldsymbol{p}}\boldsymbol{R_b} \cdot \boldsymbol{p_k} - \boldsymbol{b_k} \tag{2}$$

Siendo el vector de traslación

$$T = (t_x\ t_y\ t_z)^T \tag{3}$$

Y ${}^{p}R_b$ la matriz de rotación del sistema de coordenadas de la plataforma con respecto al sistema de coordenadas de la base definida como

$${}^{p}R_b = R_z(\mu)R_y(\theta)R_x(\psi) \tag{4}$$

$${}^{p}R_b = \begin{pmatrix} cos\mu\ cos\theta & -\text{sen}\mu\ cos\psi + cos\mu\ sen\theta\ sen\psi & \text{sen}\mu\ sen\psi + cos\mu\ sen\theta\ cos\psi \\ \text{sen}\mu\ cos\theta & \cos\mu\ cos\psi + sen\mu\ sen\theta\ sen\psi & -\cos\mu\ sen\psi + sen\mu\ sen\theta\ cos\psi \\ -sen\theta & cos\theta\ sen\psi & cos\theta\ cos\psi \end{pmatrix} \tag{5}$$

Tanto los valores de T como los de ${}^{p}R_b$ son conocidos, ya que estos equivalen respectivamente a la posición final del efector en los ejes X (t_x), Y (t_y), y Z(t_z); y a su orientación en el sistema de coordenadas de referencia (el de la base), siendo μ el ángulo de guiñada (eje Z), θ el ángulo de cabeceo (eje Y), y ψ el ángulo de alabeo (eje X). Nótese que el efector final es en este caso el centro geométrico de la plataforma.

También es posible obtener los valores de los vectores b_k y p_k conociendo la geometría de la base y la plataforma, de forma que

$$b_k = (x_k\ y_k\ z_k)^T = (R_b \cdot \cos(\delta_k)\ \ R_b \cdot \cos(\delta_k)\ \ 0)^T \tag{6}$$

$$p_k = ({}^{p}x_k\ {}^{p}y_k\ {}^{p}z_k)^T = (R_p \cdot \cos({}^{p}\delta_k)\ \ R_p \cdot \cos({}^{p}\delta_k)\ \ 0)^T \tag{7}$$

Siendo R_b y R_p los radios de los círculos en los que se encuentran los puntos B_k y P_k respectivamente, y δ_k y ${}^{p}\delta_k$ las coordenadas angulares en las que se encuentran dichos puntos en la base y la plataforma respectivamente (el prefijo p en una

variable indica que ésta se define respecto al sistema de coordenadas de la plataforma). Estos valores se muestran más adelante en el Apartado 3 de diseño. Se conocen por tanto todas las variables necesarias para obtener el valor de la longitud de la barra, definida como la norma Euclídea del vector l_k. No obstante, al no tratarse en este caso de una plataforma hidráulica, es necesario añadir una fase más al cálculo. En lugar de disponer de una barra de longitud variable $|l_k|$, se dispone de una barra fija de longitud $|d|$ que une la plataforma con el brazo del servomotor, de longitud $|h|$. A continuación, se muestra en la Figura 8 una representación de estos elementos.

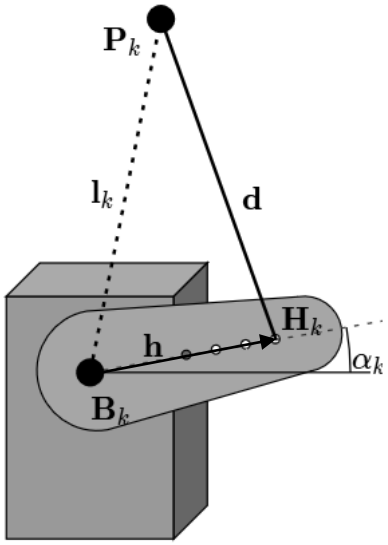


Figura 8. Representación de elementos utilizados en el cálculo de la cinemática inversa del servomotor. [18]

En esta figura se encuentra representado el vector l_k calculado en la fase anterior, así como el punto de unión del eje del servo con el brazo (B_k); la articulación que une el brazo del servo con la barra (H_k); la articulación que une la barra con la plataforma (P_k); el vector d que define la barra fija; el vector h que define el brazo del servo; y el ángulo de giro del servo α_k.

Este último ángulo es la variable que se busca definir y que irá en función del vector l_k calculado anteriormente. Al girar el servo, este ángulo cambiará su valor modificando la posición de la articulación H_k, que a su vez moverá la articulación P_k, modificándose así el valor de l_k.

Si se observa una vista en planta de la base, se puede ver también que cada brazo de servo se encuentra rotado con un ángulo β_k respecto del sistema de coordenadas de la base, siendo estos brazos ortogonales al eje de rotación de cada servo, los cuales residen en el plano XY de la base. A continuación, en la Figura 9, se muestra un ejemplo de diseño de la base en el que se encuentran definidos estos ángulos.

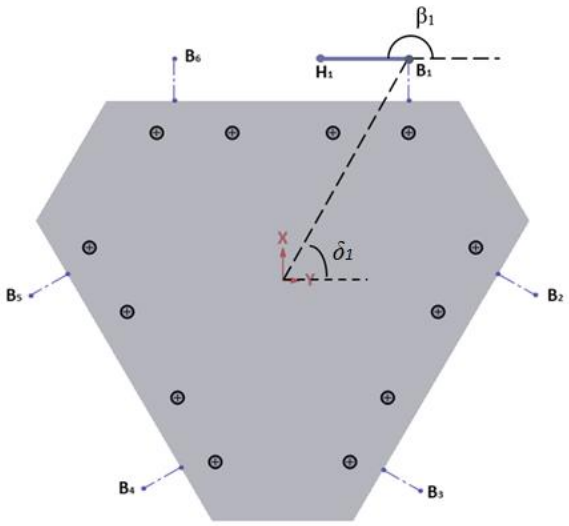


Figura 9. Ángulos característicos de diseño de la base.

Así, para los servos impares se tiene que la posición de H_k en el sistema de coordenadas de la base será

$$x_h = |h| \cdot \cos(\alpha) \cdot \cos(\beta) + x_b \quad (8)$$
$$y_h = |h| \cdot \cos(\alpha) \cdot \operatorname{sen}(\beta) + y_b \quad (9)$$
$$z_h = |h| \cdot \operatorname{sen}(\alpha) + z_b \quad (10)$$

Mientras que, para los servos pares, situados de forma que son una reflexión simétrica de los impares:

$$x_h = |h| \cdot \cos(\pi - \alpha) \cdot \cos(\pi + \beta) + x_b \quad (11)$$
$$y_h = |h| \cdot \cos(\pi - \alpha) \cdot \operatorname{sen}(\pi + \beta) + y_b \quad (12)$$
$$z_h = |h| \cdot \operatorname{sen}(\pi - \alpha) + z_b \quad (13)$$

Pero

$$\operatorname{sen}(\pi - \alpha) = sen(\alpha), y\ \cos(\pi - \alpha) = -\cos(\alpha) \quad (14)$$
$$\operatorname{sen}(\pi + \beta) = -sen(\beta), y\ \cos(\pi + \beta) = -\cos(\beta) \quad (15)$$

Y al sustituir estos valores en las ecuaciones obtenidas para los servos pares, se obtienen exactamente las mismas ecuaciones que para los servos impares.

Mediante el Teorema de Pitágoras se obtienen las siguientes ecuaciones:

$$|h|^2 = (x_h - x_b)^2 + (y_h - y_b)^2 + (z_h - z_b)^2 = \left(x_h^2 + y_h^2 + z_h^2\right) + \left(x_b^2 + y_b^2 + z_b^2\right) - 2(x_h x_b + y_h y_b + z_h z_b) \quad (16)$$

$$|l|^2 = (x_p - x_b)^2 + (y_p - y_b)^2 + (z_p - z_b)^2 = \left(x_p^2 + y_p^2 + z_p^2\right) + \left(x_b^2 + y_b^2 + z_b^2\right) - 2(x_p x_b + y_p y_b + z_p z_b) \quad (17)$$

$$|d|^2 = (x_p - x_h)^2 + (y_p - y_h)^2 + (z_p - z_h)^2 = \left(x_p^2 + y_p^2 + z_p^2\right) + \left(x_h^2 + y_h^2 + z_h^2\right) - 2(x_p x_h + y_p y_h + z_p z_h \quad (18)$$

Sustituyendo las ecuaciones 16 y 17 en la ecuación 18 y reorganizando

$$|l|^2 - (|d|^2 - |h|^2) = 2\left(x_b^2 + y_b^2 + z_b^2\right) + 2x_h(x_p - x_b) + 2y_h(y_p - y_b) + 2z_h(z_p - z_b) - 2(x_p x_b + y_p y_b + z_p z_b) \quad (20)$$

Y sustituyendo los valores de x_h , y_h , z_h de las ecuaciones 8, 9 y 10

$$\begin{aligned} |l|^2 - (|d|^2 - |h|^2) &= 2\left(x_b^2 + y_b^2 + z_b^2\right) + 2(|h|\cos(\alpha)\cos(\beta) + x_b)(x_p - x_b) \quad + 2(|h|\cos(\alpha)\,\mathrm{sen}(\beta) + y_b)(y_p - y_b) + 2(|h|\,\mathrm{sen}(\alpha) + z_b)(z_p - z_b) - 2(x_p x_b + y_p y_b + z_p z_b) \\ &= 2\left(x_b^2 + y_b^2 + z_b^2\right) + 2|h|\cos(\alpha)\cos(\beta)(x_p - x_b) + 2x_p x_b - 2x_b^2 + 2|h|\cos(\alpha)\,\mathrm{sen}(\beta)(y_p - y_b) + 2y_p y_b - 2y_b^2 + 2|h|\,\mathrm{sen}(\alpha)(z_p - z_b) + 2z_p z_b - 2z_b^2 - (x_p x_b + y_p y_b + z_p z_b) \end{aligned} \quad (21)$$

Que se reduce a

$$\begin{aligned} |l|^2 - (|d|^2 - |h|^2) &= 2|h|sen(\alpha)(z_p - z_b) + 2|h|\cos(\alpha)\cos(\beta)\,(x_p - x_b) + 2|h|\cos(\alpha)\,\mathrm{sen}(\beta)\,(y_p - y_b) \\ &= 2|h|(z_p - z_b)sen(\alpha) + 2|h|[\cos(\beta)\,(x_p - x_b) + \mathrm{sen}(\beta)\,(y_p - y_b)]\cos(\alpha) \end{aligned} \quad (22)$$

Que es una combinación lineal de ondas senoidales con una ecuación de la forma

$$C = A\ sen(\alpha) + B\ cos(\alpha) \quad (23)$$

Sabiendo que cualquier combinación lineal de ondas senoidales con el mismo período, pero desfasadas, es también una onda senoidal del mismo período, pero con un desplazamiento de fase diferente tal que

$$x\ sen(\alpha) + y\ cos(\alpha) = z\ sen(\alpha + \delta) \quad (24)$$

Donde $z = \sqrt{x^2 + y^2}$ y $\delta = \arctan\left(\frac{y}{x}\right) + \begin{cases} 0\,, & x \geq 0 \\ \pi\,, & x < 0 \end{cases}$

Es posible sustituir en nuestra ecuación anterior de forma que si

$$A = x \quad \text{,y} \quad B = y$$

Entonces, asumiendo que el valor de A es positivo

$$C = \sqrt{A^2 + B^2}\ sen(\alpha + \delta) \tag{25}$$

$$sen(\alpha + \delta) = \frac{C}{\sqrt{A^2 + B^2}} \tag{26}$$

$$\boldsymbol{\alpha = sen^{-1}\left(\frac{C}{\sqrt{A^2 + B^2}}\right) - \arctan\left(\frac{B}{A}\right)} \tag{27}$$

Siendo

$$\boldsymbol{A} = 2|h|(z_p - z_b) \tag{28}$$

$$\boldsymbol{B} = 2|h|[\cos(\beta)\,(x_p - x_b) + \text{sen}(\beta)\,(y_p - y_b)] \tag{29}$$

$$\boldsymbol{C} = |l|^2 - (|d|^2 - |h|^2) \tag{30}$$

Se ha llegado de esta forma a una ecuación que permite obtener el valor del ángulo de cada servomotor conociendo el valor del vector l_k, es decir, conociendo la posición y orientación final de la plataforma.

3. DISEÑO

El diseño de las diferentes piezas del mecanismo se ha realizado con el software CAD para modelado mecánico en 2D y 3D *SolidWorks*. Este software permite crear piezas de forma individual, proporcionando la información necesaria para su fabricación, y exportarlas. Una vez generadas todas las piezas de un mecanismo, estas pueden ensamblarse en el mismo software o en cualquier otro programa de cálculo con el objetivo de realizar simulaciones de movimiento y comportamiento mecánico.

Los elementos que componen el mecanismo desarrollado en este proyecto se pueden dividir en tres categorías: piezas mecánicas fabricadas mediante impresión 3D o corte láser, piezas normalizadas, y componentes electrónicos. Es necesario recalcar que, una vez realizado el diseño del modelo de la máquina, la mayoría de las piezas se han rediseñado de manera simplificada con vistas a reducir la complejidad del modelo utilizado en las simulaciones.

3.1. Piezas para fabricación en taller

Se ha realizado el diseño de 3 piezas con el objetivo de ser fabricadas mediante impresión en 3D o corte láser. Estas piezas son:

- Base del mecanismo, sobre la que descansan la mayoría de los componentes electrónicos, así como los soportes de los servomotores (Figura 10a).
- Soportes para los servomotores, unidos mediante dos pernos a la base y mediante cuatro pernos al servomotor, sirven de fijación para los mismos y permiten el total movimiento de sus brazos (Figura 10b).
- Plataforma, unida mediante seis pernos a las varillas, es la pieza que realiza la función de efector final, realizando los diferentes movimientos deseados. (Figura 10c).

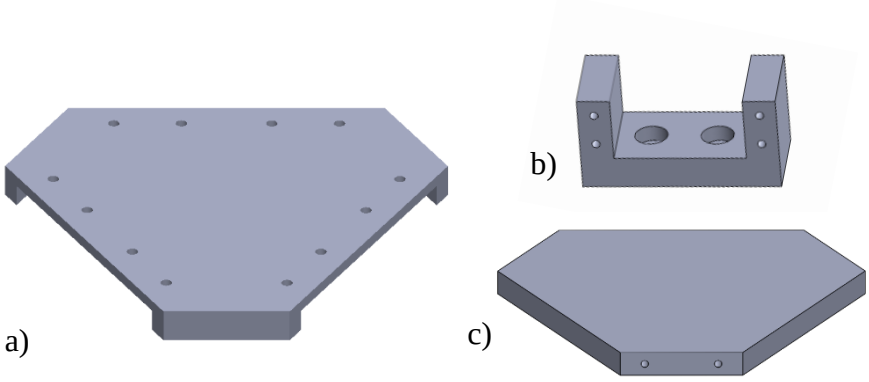


Figura 10. Piezas diseñadas para fabricación.

Los seis soportes para los servomotores se han realizado mediante impresión 3D en una impresora Zortrax M200 Plus con filamento de HIPS (filamento antichoque) y con los siguientes parámetros:

- Altura de capa: 0.14 milímetros
- Diámetro de la boquilla: 0.4 milímetros
- Relleno: 30%

En total se han usado para la fabricación 36.64 metros de filamento de 1.75 milímetros (91 gramos de material), tomando la impresión de todas las piezas un tiempo aproximado de 23 horas.

Tanto la base como la plataforma se han realizado en metacrilato mediante corte láser, realizando aparte el taladrado de los agujeros laterales de la plataforma. En total, el tiempo total requerido para la fabricación de ambas piezas ha sido de 42 minutos.

De los diseños de la base y la plataforma es posible obtener varios de los parámetros geométricos necesarios para la resolución de la cinemática inversa de la plataforma (ver Apartado 2.2). Los valores de estos parámetros son los siguientes:

- R_b = 104.89 milímetros (radio del círculo en el que se encuentran los puntos B_k)
- R_p = 90.54 milímetros (radio del círculo en el que se encuentran los puntos P_k)
- Ángulos δ_k y ${}^p\delta_k$ (coordenadas angulares de cada uno de los puntos B_k y P_k respecto de los sistemas de coordenadas de la base y la plataforma respectivamente):

i	1	2	3	4	5	6
δ_k	64.6º	355.4º	304.6º	235.4º	184.6º	115.4º
${}^p\delta_k$	40.6º	19.4º	280.6º	259.4º	160.6º	139.4º

Los planos de estas piezas se encuentran adjuntos en el Documento 3 “PLANOS” de este proyecto.

3.2. Piezas normalizadas

Para el montaje del mecanismo se han utilizado una serie de elementos normalizados tales como:

- Varillas roscadas, utilizadas para transmitir el movimiento de los servomotores a la plataforma final.
- Rótulas y rodamientos, utilizados en las uniones de las varillas con los brazos de los servos y con la plataforma, de forma que permitan el giro en los tres ejes.
- Pernos y tuercas, utilizados para fijar las distintas piezas entre sí.

Figura 11. Piezas normalizadas

3.3. Componentes electrónicos

Los componentes electrónicos necesarios para el funcionamiento de la plataforma son los siguientes:

- Arduino Due, placa basada en un microcontrolador ARM de 32 bits. Su hardware consiste, entre otras cosas, en 54 pines entrada/salida digitales (12 de los cuales pueden ser usados como señales PWM), 12 entradas y 2 salidas analógicas y 4 puertos serie. La placa tiene un voltaje de operación de 3.3 V, y un voltaje de alimentación recomendado de entre 7 y 12 V.

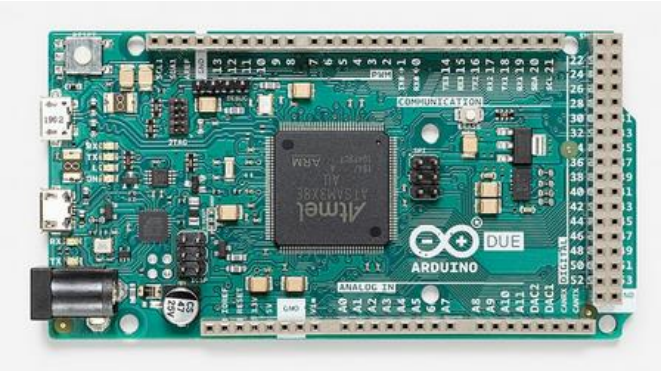


Figura 12. Microcontrolador Arduino Due [20]

- Módulo bluetooth HC-06, compatible con la placa Arduino y con un voltaje de entrada de entre 3.3 y 6 V. Este componente se utiliza para realizar la comunicación vía Bluetooth entre el microcontrolador y la aplicación móvil mediante la que se controlará el movimiento de la plataforma.

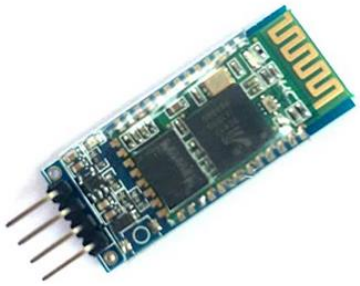


Figura 13. Módulo bluetooth HC-06 [21]

- Servomotores MG996R de la marca AZ-Delivery. Estos servomotores disponen de engranajes metálicos y un par máximo de 11 kg/cm^2. Tienen un voltaje de entrada de entre 4.8 y 7.2 V, y consumen aproximadamente 500mA. Pueden rotar aproximadamente 120 grados y tienen una velocidad de operación de 0.14s/60º cuando están alimentados a 6 V. Estos servomotores son controlados mediante señales PWM, cuyo ancho de pulso determina su ángulo de giro.

Figura 14. Servomotor MG996R [22]

- Módulo controlador de servos PCA9685 compatible con Arduino. Este módulo permite controlar un total de 16 señales PWM con 12 bits de resolución para el control de los servomotores y se comunica con el microcontrolador Arduino mediante un bus I2C. Este driver se alimenta con un voltaje de entre 3 y 5 V y dispone de bornes para conectar una alimentación externa para los servomotores de hasta 6 V.

Figura 15. Driver PCA9685 [21]

- Además de estos componentes también será necesaria para el conexionado y alimentación de la plataforma la utilización de cables y baterías externas.

3.4. Diseño del modelo

Una vez realizado el diseño de la plataforma final, se procede a rediseñar algunas de las piezas con el objetivo de simplificar el modelo que será utilizado en la simulación. Estos nuevos diseños respetan completamente las medidas de los diseños anteriores, pero eliminan objetos y uniones que puedan resultar complejas para el cálculo del movimiento del mecanismo en el software de la simulación.

En este sentido, se ha realizado un diseño de la base enfocado a la simulación, a la cual se han añadido unos bloques que representan a los servomotores, así como un nuevo diseño de los brazos de estos servomotores que se acoplen de manera sencilla a la nueva base. Por último, se han realizado unas ligeras modificaciones a la plataforma de forma que el acople de esta con las varillas también se realice de manera sencilla. Estos diseños se muestran a continuación en la Figura 16.

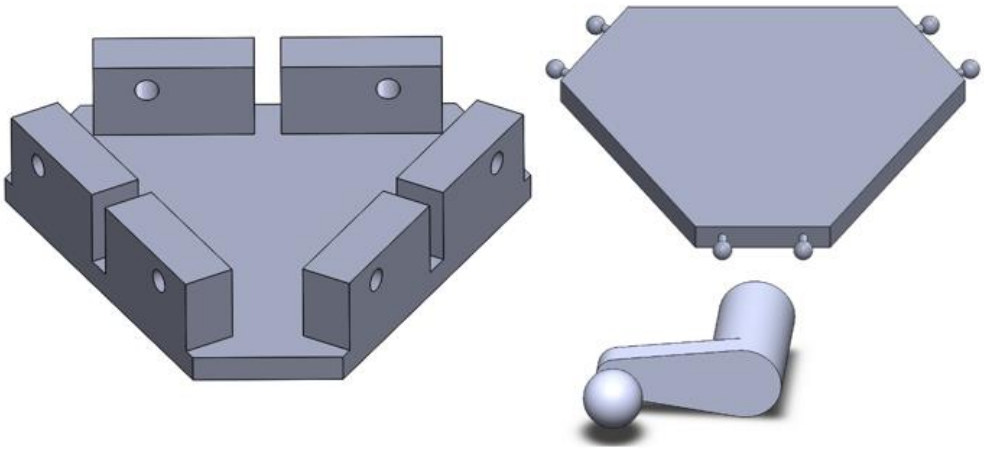


Figura 16. Piezas rediseñadas para la simulación

4. SIMULACIÓN

Una vez diseñado el modelo de la plataforma, se procede a simular el movimiento del mismo mediante *Simulink*. *Simulink* es un entorno de programación visual que funciona sobre el entorno de programación *Matlab*, en el cual es posible el modelado y la simulación de sistemas dinámicos.

Dentro del entorno de *Simulink* se ha utilizado la herramienta *Simscape Multibody*, la cual proporciona un entorno de simulación multicuerpo para sistemas mecánicos 3D. Esta herramienta permite generar el modelo de un mecanismo físico mediante la introducción de bloques que representan cuerpos, articulaciones, restricciones, elementos de fuerza y sensores. Estos bloques pueden ser generados en el propio entorno o importados de un montaje CAD externo. Una vez generado el modelo, *Simscape Multibody* formula y resuelve las ecuaciones de movimiento de todo el sistema mecánico, generando una animación 3D que permite visualizar la dinámica del mismo.

4.1. Generación del modelo

Anteriormente se ha mencionado la posibilidad de importar en *Simscape* diferentes cuerpos creados mediante un software CAD a través de los bloques de tipo “sólido”.

De esta forma, las piezas diseñadas en el Apartado 3.4 se han introducido utilizando estos bloques, los cuales permiten configurar una serie de parámetros tales como la densidad del objeto, que el programa utilizará para calcular su inercia; las propiedades gráficas para la visualización del objeto como su color y la opacidad; y por último es posible la determinación de una serie de marcos de referencia, que servirán como puntos de unión con el resto de objetos a través de los distintos tipos de articulaciones.

En la Figura 17 se muestra como ejemplo la visualización de la base con sus respectivos marcos de referencia: los marcos nombrados como “Brz 1…6” son los puntos de unión con cada uno de los brazos de los servos, mientras que el marco “R”, situado en el centro geométrico del suelo de la base, se ha utilizado como punto de referencia global para toda la plataforma.

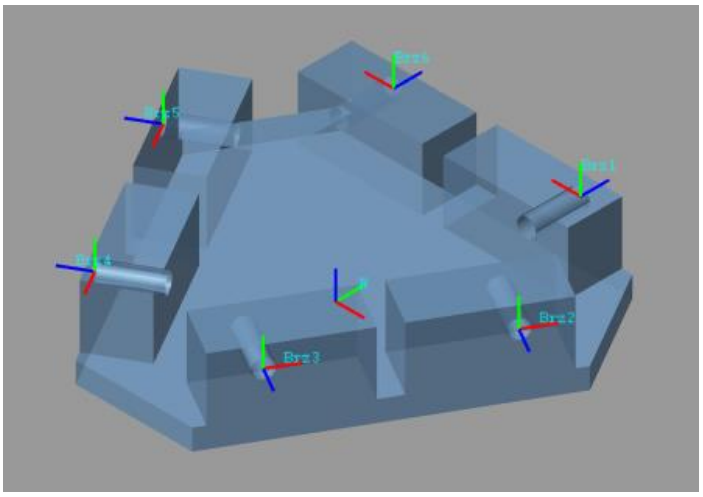


Figura 17. Visualización de la base con marcos de referencia en Simscape.

Cada uno de los brazos se ha unido a la base a través de una articulación de revolución de un grado de libertad. Estas articulaciones deben representar el movimiento de los servomotores, por lo que a cada una de ellas se le introduce una señal que indica el ángulo de giro calculado para cada servomotor. Esta señal se calculará en tiempo real en otro bloque que se especifica más adelante.

Así, cada brazo girará de manera independiente, transmitiendo el giro a su respectiva varilla mediante una articulación esférica de tres grados de libertad. Esta configuración se muestra ejemplificada a continuación en la Figura 18.

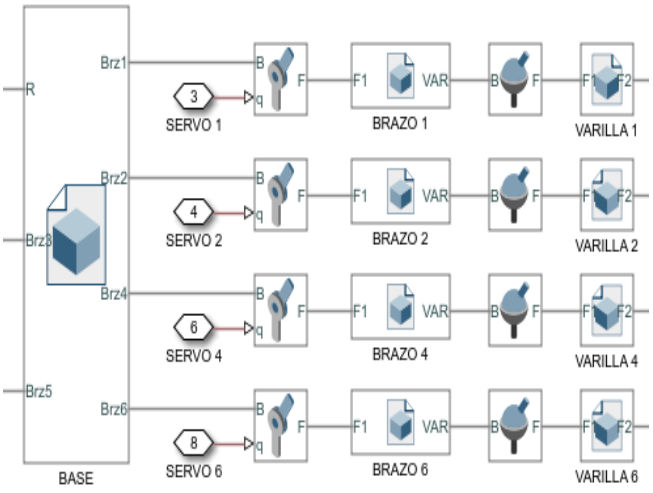


Figura 18. Detalle de conexiones entre la base, los brazos y las varillas.

A continuación, las varillas se unen a la plataforma móvil también mediante articulaciones esféricas de tres grados de libertad, quedando el modelo completo de la siguiente forma (Figura 19):

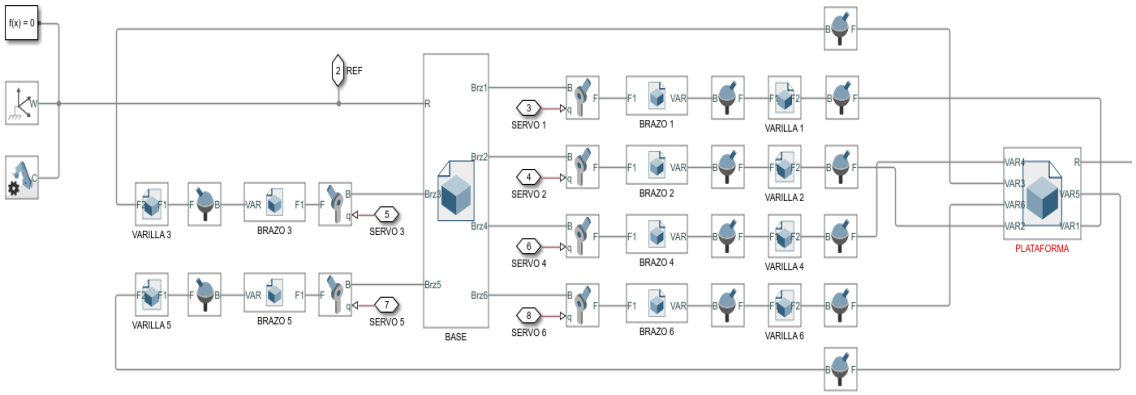


Figura 19. Modelo completo de la Plataforma Stewart en Simscape.

Por último, es necesario conectar la plataforma a tres bloques proporcionados por el entorno de *Simscape*, en estos bloques se encuentran los parámetros relativos a la configuración de la simulación y del mecanismo (Figura 20).

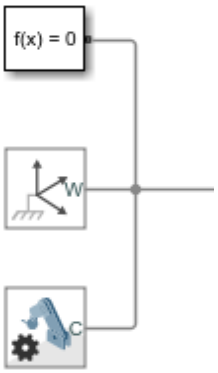


Figura 20. Detalle de bloques de configuración de Simscape.

La descripción de los bloques de la figura, de arriba abajo, es la siguiente:

- Configuración de la simulación: En este bloque se pueden modificar una serie de parámetros relativos al método de resolución utilizado por el software a la hora de realizar la simulación, como el número de iteraciones, la aplicación de filtros, etc. En este caso se ha utilizado la configuración por defecto.

- Marco de referencia del "mundo": Este bloque se utiliza como origen de coordenadas global en todo el entorno. Se trata de un origen de coordenadas inmóvil y ortogonal, regido por la regla de la mano derecha. Como se observa en la Figura 19, este origen se ha hecho coincidir con el marco de referencia "R" de la base de la plataforma, ya que todas las mediciones de los sensores se realizarán con respecto a este punto.

- Configuración del mecanismo: En este bloque se establecen los parámetros mecánicos y de la simulación que afectan directamente a la máquina. Se ha eliminado la acción de la gravedad para poder mantener los brazos de los servos en una posición inicial horizontal de la misma forma que en el montaje real.

De esta forma queda definido el modelo y ya es posible generar una animación en 3D del mismo, con la que será posible visualizar en tiempo real el movimiento de la plataforma. La animación generada se muestra a continuación en la Figura 21.

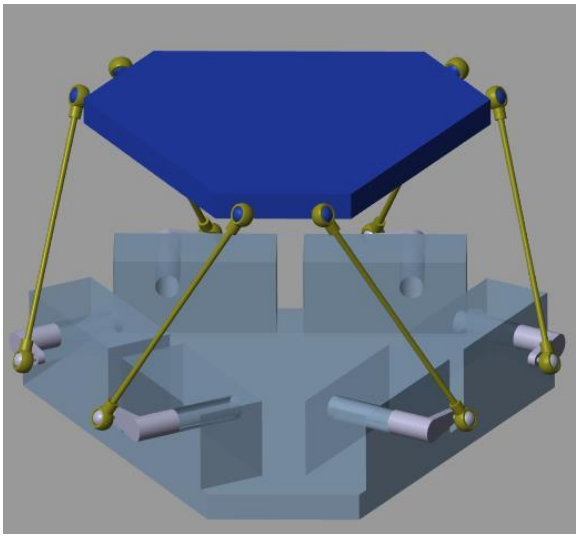


Figura 21. Animación 3D de la Plataforma Stewart generada en Simscape.

4.2. Generación de las referencias

El siguiente paso consiste en generar las señales que se envían a los servomotores y provocan el movimiento de la plataforma. Esta señal debe contener la información del ángulo de giro para cada uno de los seis servomotores de manera independiente, en radianes, de forma que la plataforma llegue a una posición y orientación determinadas.

Para ello se crean seis bloques, cada uno de los cuales genera una señal distinta que se corresponde con un parámetro de la posición y ángulo final (posición en los ejes X, Y, Z y ángulo en los ejes X, Y, Z). Estas señales pasan por un bloque de función que contiene una función llamada “f_stewart”, la cual recibe como argumentos los seis parámetros de la posición y orientación finales y devuelve los ángulos de giro de cada uno de los seis servomotores. Esta función se ha creado utilizando para el cálculo de los ángulos las relaciones trigonométricas halladas en el apartado 2.2 de este documento (*Cálculo de la cinemática inversa*). El código generado para esta función se muestra en el anexo A de este documento.

Por último, se añade un bloque "num/den" a cada referencia angular generada. Este bloque contiene una función de transferencia que incluye en la simulación una representación de la respuesta dinámica de los servos, de forma que esta se acerque a la respuesta real de los mismos. Además, se multiplica por -1 la referencia para los servos pares ya que estos se encuentran montados de forma inversa.

El subsistema dedicado a la generación de las señales queda por tanto definido en *Simscape* de la siguiente manera (Figura 22):

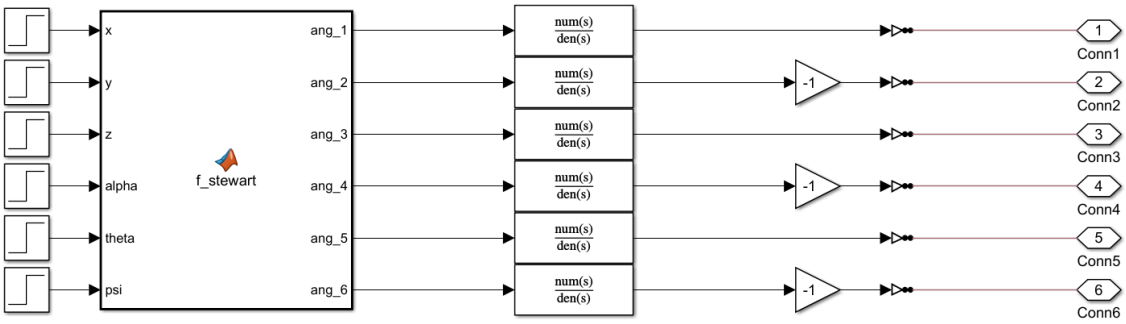


Figura 22. Subsistema de generación de señales en Simscape.

4.3. Toma de medidas

Con el objetivo de comprobar el correcto funcionamiento de la simulación, se ha generado un pequeño subsistema dedicado a registrar todos los movimientos que realiza la plataforma, tanto los desplazamientos como los giros en cada uno de los tres ejes. Para ello se ha utilizado un bloque de *Simscape* llamado "Sensor de transformación" (Figura 23), el cual es capaz de medir transformaciones tridimensionales dependientes del tiempo y sus derivadas entre dos orígenes de coordenadas.

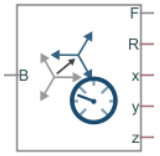


Figura 23. Bloque de "Sensor de transformación" en Simscape.

A este bloque se le introduce como marco de referencia fijo el origen de coordenadas "R" después de aplicarle una transformación en el eje Z equivalente a la altura inicial de la plataforma, de forma que, en su posición inicial, coincida con el origen de coordenadas del centro de la plataforma. Este último origen de coordenadas es el que se introduce al bloque como marco de referencia móvil. En

el momento en que la plataforma realice algún movimiento, el bloque registrará la transformación en el tiempo entre el origen de la plataforma y el marco de referencia fijo, arrojando los valores de los desplazamientos en los ejes X, Y, Z, así como la matriz de rotación.

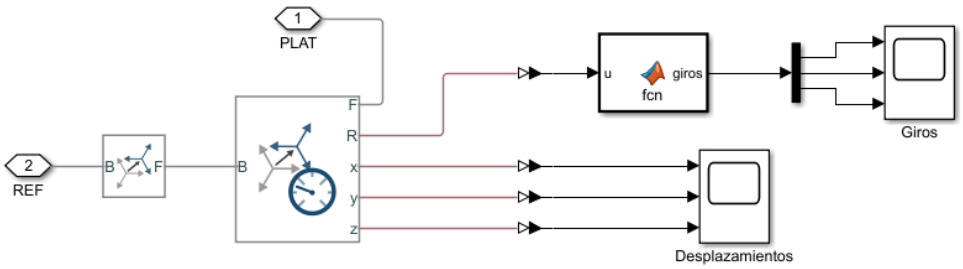


Figura 24. Subsistema de toma de medidas en Simscape.

Como se observa en la Figura 24, los valores de los desplazamientos pueden ser observados directamente en forma de gráficas, mientras que, para obtener las gráficas de los giros, es necesario pasar la matriz de rotación por una función "rotm2eul" que devuelve el valor de la rotación en radianes en cada uno de los ejes.

4.4. Resultados

De esta forma, queda definido por completo en *Simulink* el sistema que realiza la simulación. Como se ha visto anteriormente, este se ha dividido en tres subsistemas diferentes; uno dedicado a la generación de señales de referencia para los servomotores, otro que contiene el modelo en 3D de la Plataforma Stewart, y un último con los sensores necesarios para medir el movimiento de la misma (ver Figura 25).

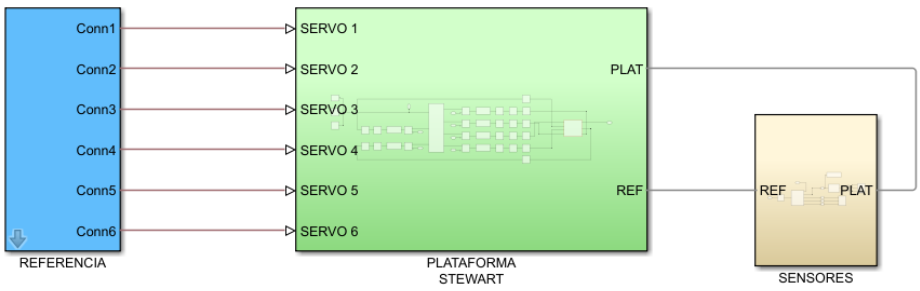


Figura 25. Sistema completo simplificado en Simulink para la simulación del movimiento.

En este sistema se han realizado dos tipos de simulación diferentes. En primer lugar, se ha comprobado que la plataforma puede realizar desplazamientos y giros,

tanto positivos como negativos, en cada uno de los tres ejes, y que es capaz de llegar a una serie de posiciones establecidas como referencia. En segundo lugar, se le han introducido una serie de trayectorias y se ha comprobado que la plataforma es capaz de seguirlas de manera uniforme en el tiempo.

4.4.1. Seguimiento de referencia

- Variación de una coordenada. En un período de 15 segundos, partiendo de la posición inicial en la que todas las coordenadas son igual a cero, se ha realizado una secuencia de escalones para cada una de las seis coordenadas que se pueden controlar en el sistema, variando una coordenada cada medio segundo de forma que nunca se modifiquen dos al mismo tiempo. La secuencia que se ha seguido es la siguiente:

Coordenada	X (mm)	Y (mm)	Z (mm)	ψ (º)	θ (º)	α (º)
Punto 1	2	2	3	3	3	2
Punto 2	0	0	0	0	0	0
Punto 3	-2	-2	-3	-3	-3	-2
Punto 4	2	2	3	3	3	2
Punto 5	0	0	0	0	0	0

Los resultados obtenidos se muestran a continuación:

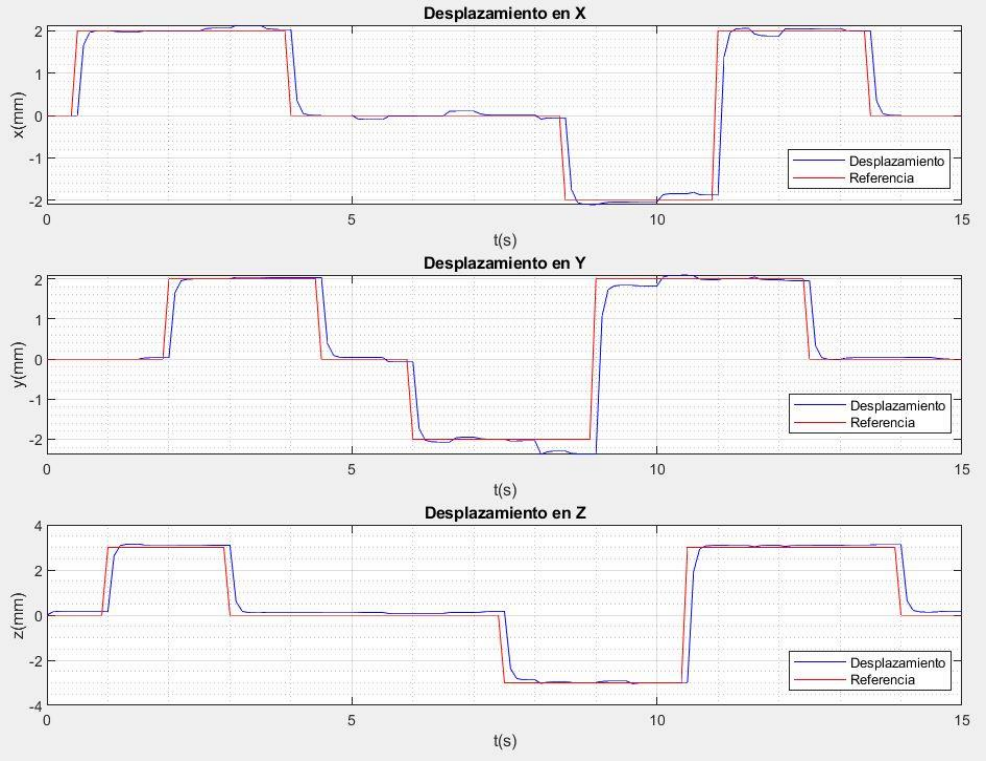


Figura 26. Desplazamientos de la plataforma en los 3 ejes vs tiempo

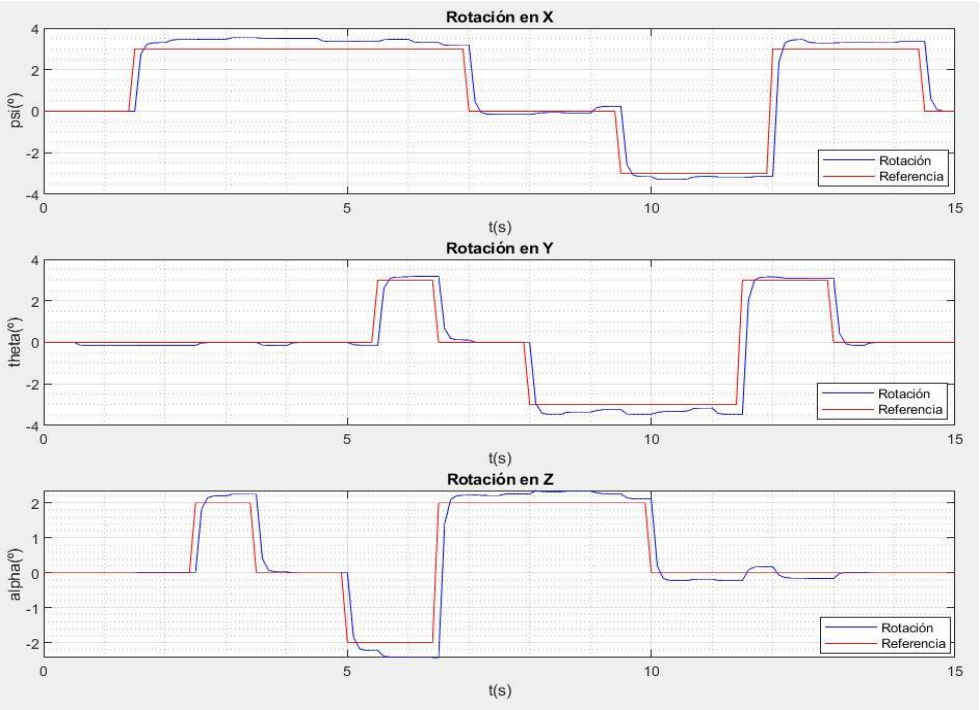


Figura 27. Rotación de la plataforma en los 3 ejes vs tiempo

En las figuras anteriores se puede comprobar como la plataforma alcanza de manera bastante precisa las posiciones en cada uno de los tres ejes, siendo esta precisión algo más baja en los giros. También se observan ciertas perturbaciones producidas por movimientos bruscos de la plataforma.

- Variación de todas las coordenadas. En un período de 10 segundos, partiendo de la posición inicial, se han modificado todas las coordenadas al mismo tiempo cada 2 segundos. La secuencia que se ha seguido es la siguiente:

Coordenada	X (mm)	Y (mm)	Z (mm)	ψ (º)	θ (º)	α (º)
Punto 1	1	-3	3	2	-3	1
Punto 2	-3	1	-4	-2	0	-2
Punto 3	4	-4	0	4	-4	0
Punto 4	2	2	4	-2	2	3

Los resultados obtenidos se muestran a continuación:

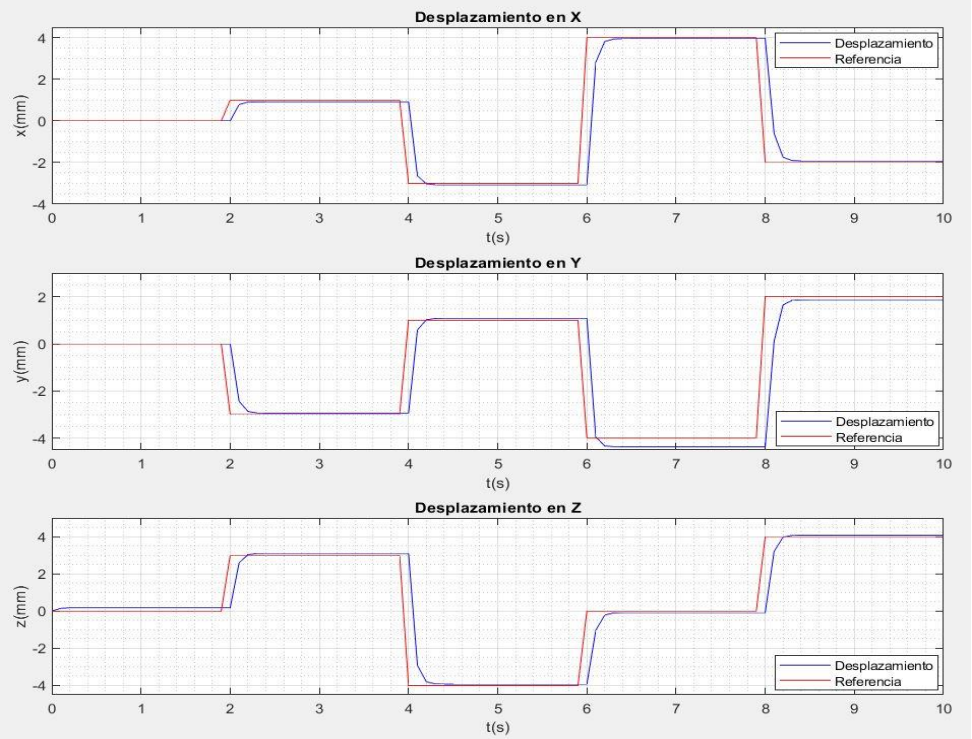


Figura 28. Desplazamiento de la plataforma en los 3 ejes vs tiempo

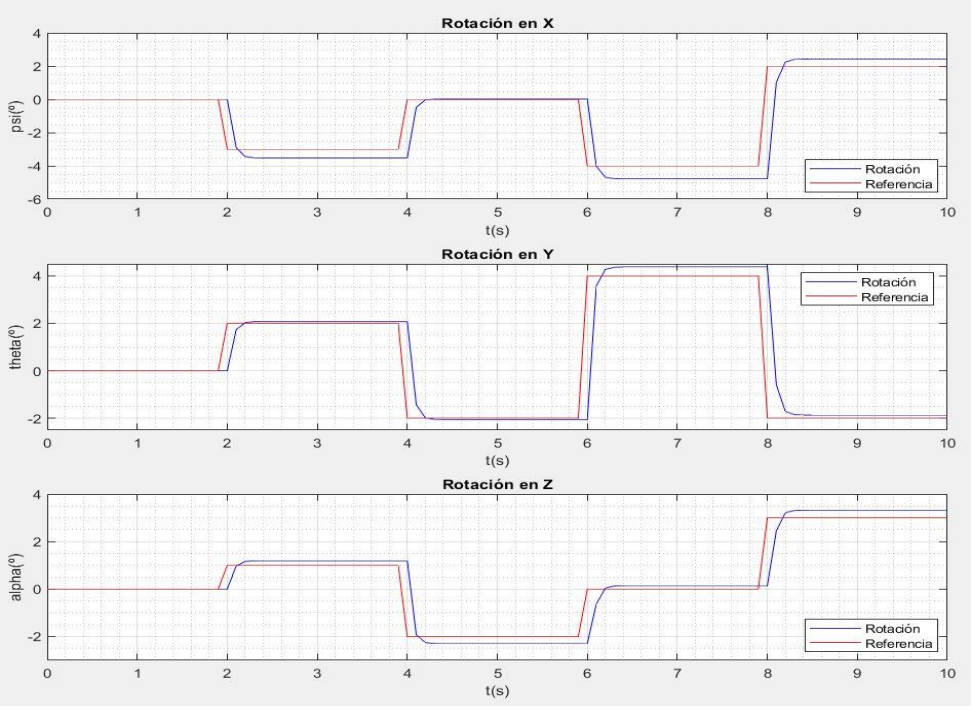


Figura 29. Rotación de la plataforma en los 3 ejes vs tiempo

De estas figuras se puede deducir que la plataforma simulada responde bien a los cambios de coordenadas, aunque en ocasiones no llega a seguir la referencia de manera exacta, sobre todo en lo que respecta a los giros. Cabe

destacar que este tipo de plataformas tiene una serie de limitaciones de movimiento por razones sobre todo geométricas. Estas limitaciones pueden ser la causa de que en ocasiones la plataforma no llegue a la referencia establecida. Es por esto que a la hora de introducir las coordenadas de referencia se ha tratado de no llevar a la plataforma a posiciones demasiado lejanas o complejas, quedando la mayoría de desplazamientos y giros en un rango de 4 milímetros y 4 grados.

4.4.2. Trayectorias

- Figuras de Lissajous. Las figuras de Lissajous son las curvas que recorre un punto sometido a un doble movimiento armónico simple en dos direcciones perpendiculares. Si denominamos a cada una de estas direcciones *x(t)* e *y(t)* podemos describir sus trayectorias individuales como:

$$x(t) = Asen(\omega_1 t)$$
$$y(t) = Bsen(\omega_2 t + \delta)$$

Donde *A* y *B* son las amplitudes de cada trayectoria, ω_1 y ω_2 sus respectivas frecuencias angulares, y δ la diferencia de fase entre ambos movimientos. La forma que adquiere la curva de Lissajous depende exclusivamente de la relación entre ambas frecuencias ω_1:ω_2 y del desfase. A continuación, en la Figura 30, se muestran las curvas correspondientes a relaciones de frecuencias sencillas para ciertos desfases.

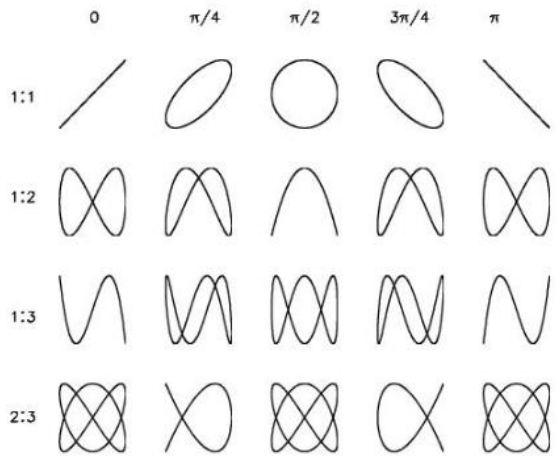


Figura 30. Curvas simples de Lissajous [19].

Este tipo de curvas se ha utilizado para comprobar la capacidad de la plataforma para seguir una trayectoria de manera uniforme en el tiempo. Para ello se les han introducido referencias senoidales a las entradas de las coordenadas X e Y.

En primer lugar, las ondas senoidales introducidas poseen las siguientes características:

- Coordenada X → A = 5 mm, $\omega_1 = 2\pi 0.2$ rad/s
- Coordenada Y → B = 5 mm, $\omega_2 = 2\pi 0.3$ rad/s, $\delta = \pi/2$ rad

Es decir, que entre las dos trayectorias existe una relación de frecuencias de 2:3 y un desfase de $\pi/2$ radianes. Los resultados obtenidos son los siguientes:

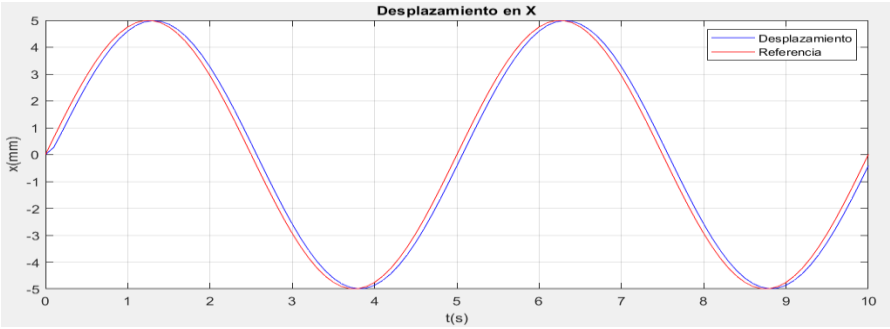


Figura 31. Desplazamiento de la plataforma en el eje X vs tiempo

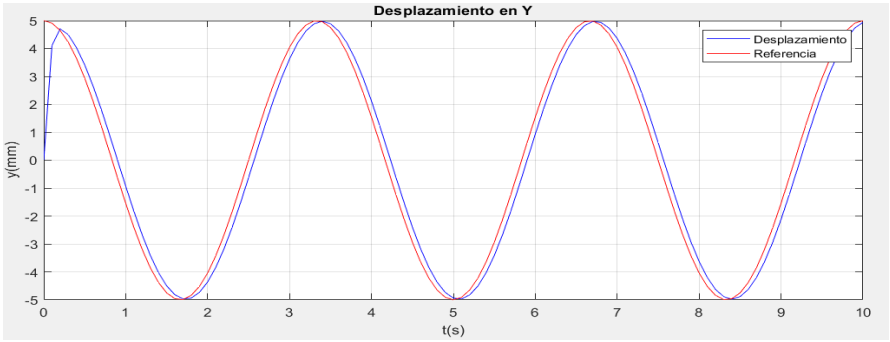


Figura 32. Desplazamiento de la plataforma en el eje Y vs tiempo

Como se muestra en las figuras anteriores, el movimiento de la plataforma en ambos ejes sigue con bastante exactitud la señal de referencia. Para observar que la curva resultante coincide con la correspondiente curva de Lissajous para esta relación de frecuencias y desfase (Figura 33) se muestran ambos desplazamientos enfrentados en una gráfica. El resultado es el siguiente:

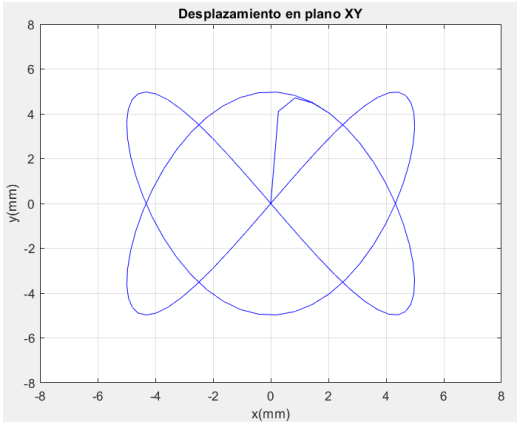


Figura 33. Curva de Lissajous recorrida por la plataforma en los ejes X e Y

A continuación, se ha tratado de que la plataforma realice una curva de Lissajous en tres dimensiones añadiendo una tercera referencia senoidal a la coordenada del eje Z. Las características de las ondas introducidas son las siguientes:

- Coordenada X → A = 3 mm, $\omega_1 = 2\pi 0.2$ rad/s
- Coordenada Y → B = 3 mm, $\omega_2 = 2\pi 0.2$ rad/s, $\delta = \pi/2$ rad
- Coordenada Z → C = 4 mm, $\omega_3 = 2\pi 0.25$ rad/s

Los resultados de los desplazamientos obtenidos se muestran a continuación:

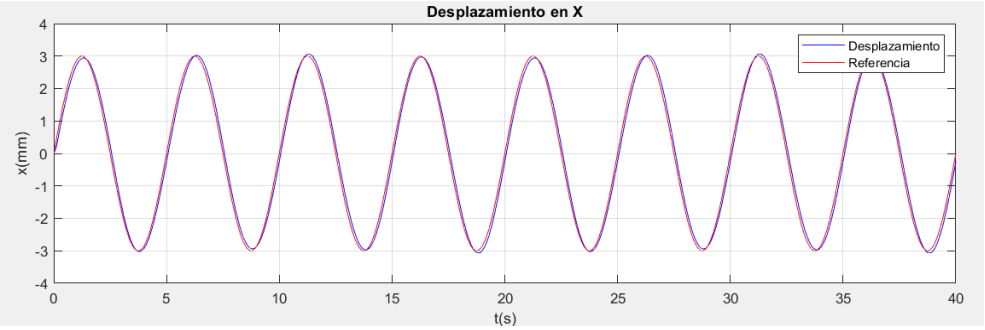


Figura 34. Desplazamiento de la plataforma en el eje X vs tiempo

Figura 35. Desplazamiento de la plataforma en el eje Y vs tiempo

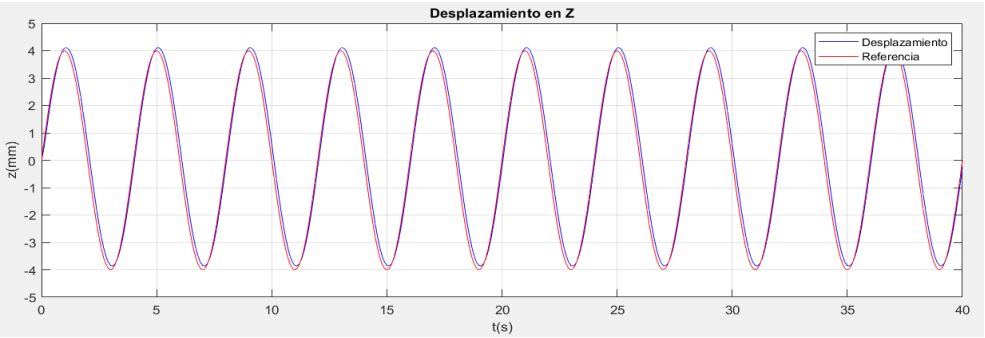


Figura 36. Desplazamiento de la plataforma en el eje Z vs tiempo

Como se observa, la plataforma continúa respondiendo bien y el desplazamiento de esta en todos los ejes se corresponde con la señal de referencia en cada uno de ellos. La curva de Lissajous obtenida es la siguiente:

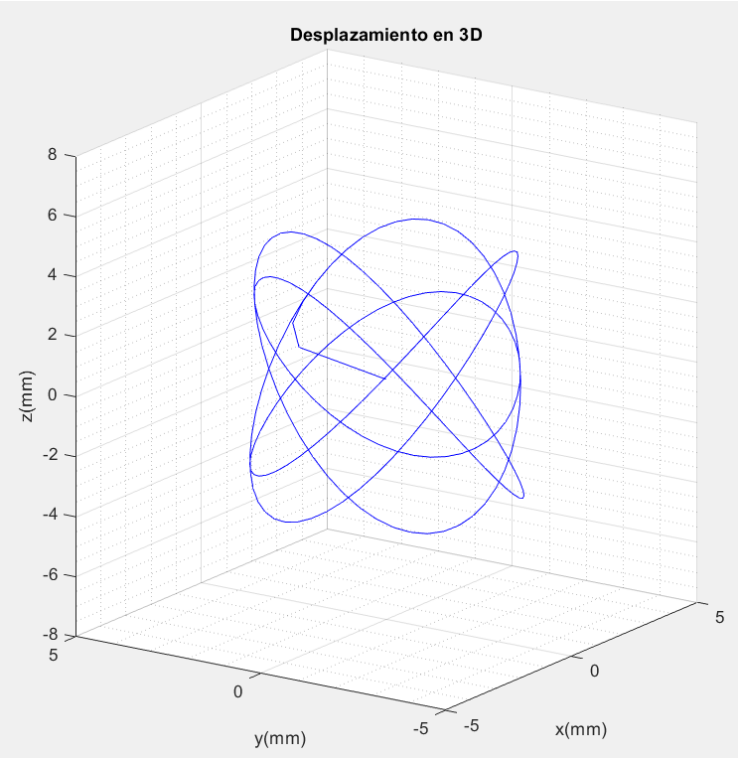


Figura 37. Curva de Lissajous recorrida por la plataforma en 3 dimensiones

Por último, se ha realizado la misma operación, pero introduciendo las tres señales de referencia senoidales a las coordenadas de los giros en cada uno de los ejes. Las características de las ondas introducidas son las siguientes:

- Coordenada ψ → A = 3º, $\omega_1 = 2\pi 0.2$ rad/s
- Coordenada θ → B = 3º, $\omega_2 = 2\pi 0.4$ rad/s
- Coordenada α → C = 2º, $\omega_3 = 2\pi 0.6$ rad/s

Los resultados de los giros obtenidos se muestran a continuación:

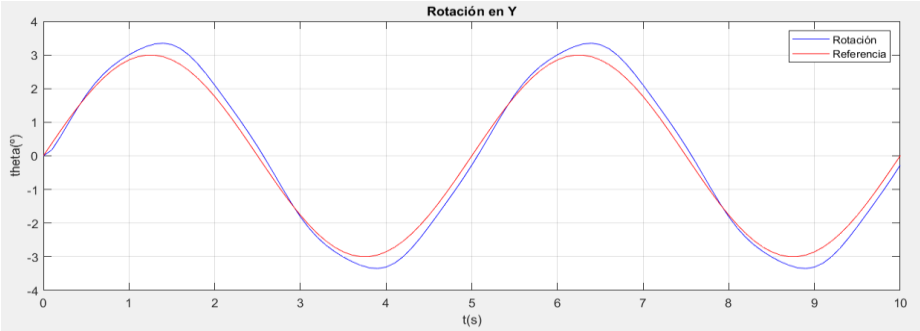


Figura 38. Rotación de la plataforma en el eje X vs tiempo

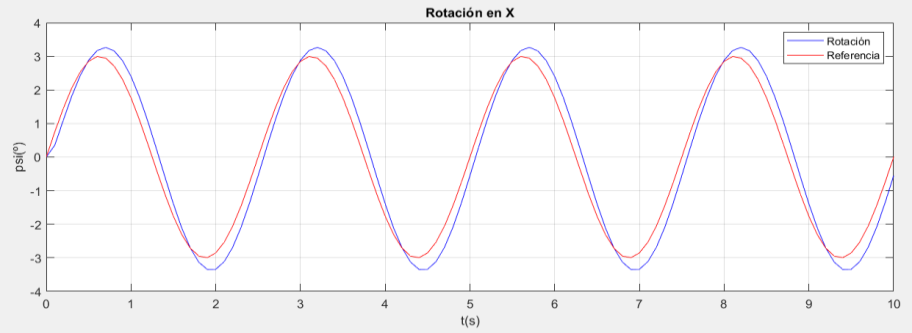


Figura 39. Rotación de la plataforma en el eje Y vs tiempo

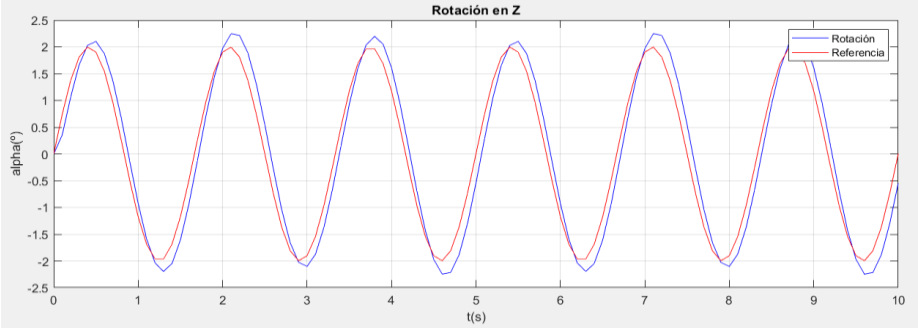


Figura 40. Rotación de la plataforma en el eje Z vs tiempo

Como se observa en las Figuras 38, 39 y 40, al igual que en el ensayo anterior, el seguimiento de la referencia de los giros resulta menos preciso que el de los desplazamientos. No obstante, la plataforma simulada parece comportarse de forma correcta ante el seguimiento de todo tipo de referencias, mostrando pequeños errores que podrían ser subsanados en el futuro con la implementación, por ejemplo, de un sistema de control mediante reguladores.

Los resultados obtenidos en la simulación permiten deducir que la Plataforma Stewart resulta viable para su fabricación, puesto que esta es capaz de realizar todo tipo de movimientos y de seguir una trayectoria deseada.

5. IMPLEMENTACIÓN

Una vez comprobado que el diseño de la plataforma resulta efectivo, se procede a su montaje e implementación real. Para ello, se han utilizado las piezas fabricadas a partir de los diseños del apartado 3, así como el resto de las piezas normalizadas y componentes electrónicos.

En este apartado se detallan las instrucciones a seguir para el montaje de la plataforma y la conexión de esta con el microcontrolador Arduino y el resto de componentes. Se explica de manera resumida el programa desarrollado para el Arduino, así como la aplicación creada para controlar el movimiento de la plataforma de manera remota a través de un teléfono móvil.

5.1. Instrucciones de montaje

A continuación se muestra, paso a paso, el proceso a seguir para llevar a cabo el montaje de las piezas de la plataforma.

En primer lugar, obtener uno de los soportes para servos y fijarlo a la base, introduciendo dos pernos para asegurar la unión tal y como se muestra en la figura 41.

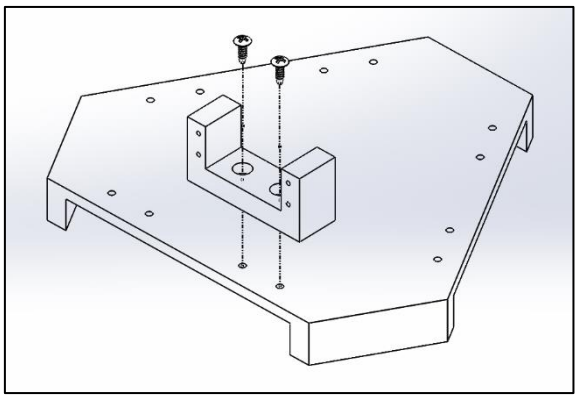


Figura 41. Paso 1 del montaje del prototipo

En segundo lugar, obtener un servomotor e introducirlo en el soporte anterior de la forma que se muestra en la figura 42, añadiendo cuatro pernos para fijar el motor al soporte. Introducir también uno de los brazos en el eje del servo ayudándose de las muescas que se encuentran en ambas piezas.

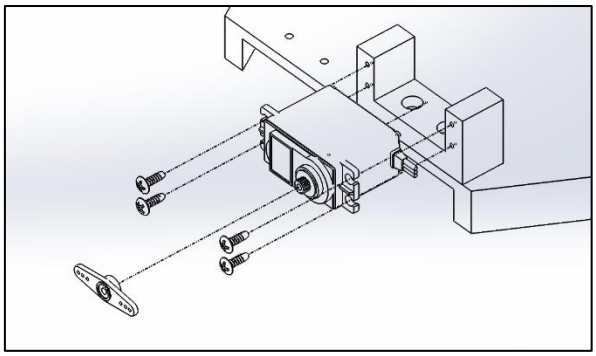


Figura 42. Paso 2 del montaje del prototipo

En tercer lugar, obtener una de las varillas y, introduciendo un perno a través de las rótulas, fijarla al brazo del servomotor por la parte inferior y a la plataforma por la parte superior. Este paso se detalla en la figura 43.

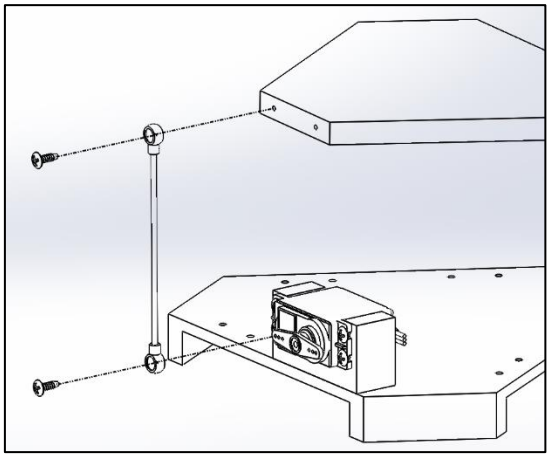


Figura 43. Paso 3 del montaje del prototipo

Por último, repetir estos tres pasos con el resto de las piezas hasta completar el montaje de la plataforma, que debe quedar terminada de la manera que se muestra a continuación en la figura 44.

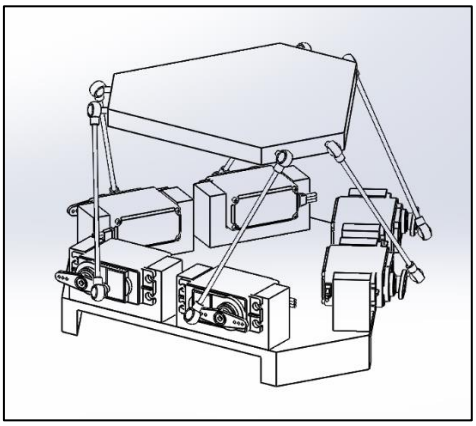


Figura 44. Visualización del prototipo completamente montado

5.2. Conexionado del sistema

Una vez montado el prototipo, se procede al conexionado del mismo con los componentes electrónicos y el microcontrolador Arduino.

Cada uno de los servomotores dispone de tres cables distintos: un cable de color marrón (GND) y otro de color rojo (V+) para la alimentación y un cable de color amarillo para la recepción de señales PWM. Todos los servomotores del montaje deben conectarse al driver PCA9685, puesto que no sería posible alimentarlos directamente desde el microcontrolador. Con este propósito se conecta al driver una batería externa compuesta por cuatro pilas AA de 1.5 V.

El PCA9685 se conecta al microcontrolador Arduino a través de cuatro cables: dos de ellos para la alimentación (la del propio driver, no la de los servomotores), y otros dos para la comunicación con el Arduino mediante el protocolo I2C.

Además del driver, a la placa Arduino se debe conectar también el módulo Bluetooth HC-06 mediante cuatro cables: dos de ellos para la alimentación del módulo y otros dos para establecer un bus de comunicación serie con el microcontrolador.

A su vez, la placa del microcontrolador Arduino dispone de dos métodos de alimentación: uno a través del puerto USB y otro a través del puerto de alimentación de corriente continua. El puerto USB se ha utilizado como método de alimentación al mismo tiempo que se ha realizado la programación y depuración del código del microcontrolador. Una vez finalizada esta fase, se procede a alimentar al Arduino mediante una batería de polímero de iones de litio de 7.4 V y 6000 mAh.

Las conexiones que se acaban de detallar se muestran de manera esquemática a continuación en la figura 45.

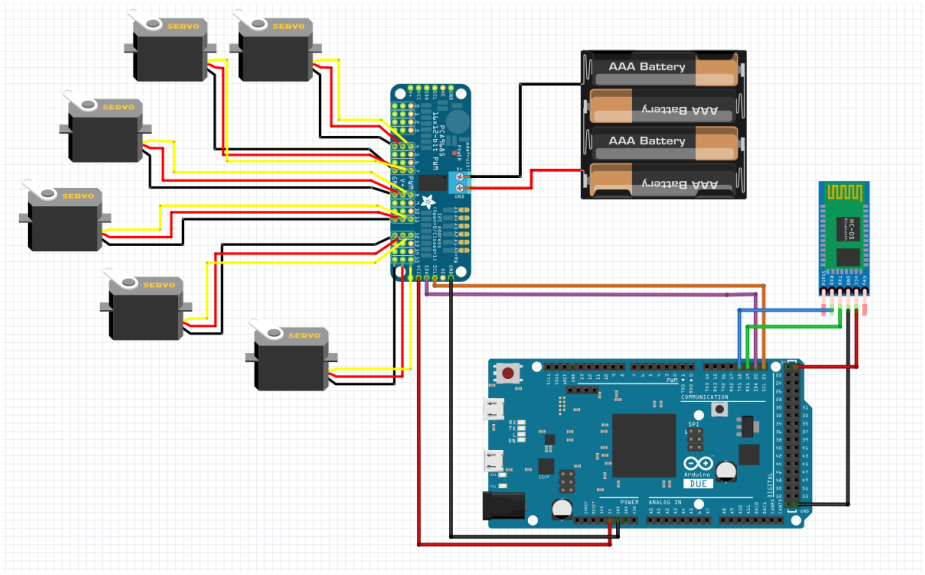


Figura 45. Esquema de conexiones del sistema

En la siguiente tabla se muestra un resumen de los pines de la placa Arduino Due utilizados para las distintas conexiones:

Pin de Arduino	Conexión	Componente
5 V	Vcc	HC-06
GND	GND	
18 (TX1)	RXD	
19 (RX1)	TXD	
5 V	Vcc	PCA9685
GND	GND	
20 (SDA)	SDA	
21 (SCL)	SCL	
USB programación	Puerto USB portátil	IDE Arduino
Alimentación CC	-	Batería externa

En el driver PCA9685, los servomotores se han conectado en los pines 4, 7, 8, 11, 12, y 15.

5.3. Programación del microcontrolador

La programación del microcontrolador se ha realizado en el entorno de desarrollo integrado de Arduino, aplicación que permite escribir y cargar programas en placas compatibles con Arduino. A continuación, se pasa a explicar de manera

resumida, y mediante diagramas de flujo, el código que compone el programa desarrollado. El código al completo, con comentarios aclaratorios, se muestra en el Anexo B de este documento.

- Cuerpo principal del programa (figura 46): Comienza con la inicialización de las librerías y variables necesarias para la ejecución del mismo. A continuación, pasa por una función que realiza la configuración inicial, antes de entrar de manera definitiva en el bucle sin fin.

- Configuración inicial (figura 46): En esta función se inicializa el objeto creado para el control de los servos y la comunicación serie, tanto para la depuración del código si fuera necesaria como para la comunicación con el módulo Bluetooth. Además, se realiza el cálculo de la geometría inicial de la plataforma y se envían todos los servos a la posición inicial (se ha tomado por posición inicial un giro de 90º, ya que todos los servos se encuentran tumbados), utilizando la función "set_servo" que se explica más adelante.

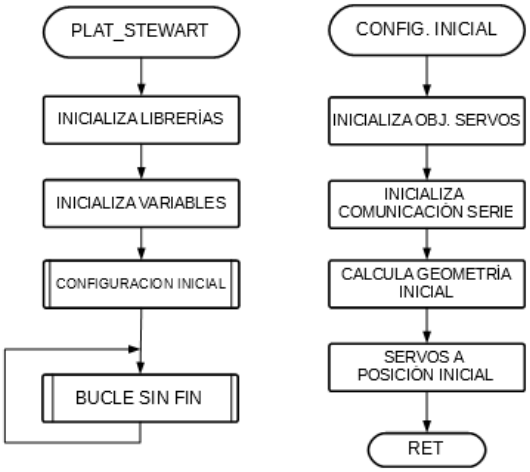


Figura 46. Diagramas de flujo I y II

- Bucle sin fin (figura 47): Este bucle se repite una y otra vez a la espera de recibir una orden del módulo Bluetooth. Estas órdenes se reciben en forma de trama, cadenas de caracteres del estilo:

"AAB,X.X,X.X,X.X,X.X,X.X,X.X,X.X"

Donde los caracteres "AAB" identifican el modo de funcionamiento de la plataforma y los caracteres "X.X" son los parámetros relevantes en dichos modos. Al recibir una trama, se identifica el modo seleccionado y se realiza una acción distinta en función de este:

- Modo de **selección de posición** (caracteres iniciales “MV”): Se leen la posición y giro deseados y se envía la plataforma a dichas coordenadas mediante la función “giro_servos”.
- Modo **curvas** (caracteres iniciales “RT1” o “RT2”): En este modo el programa genera tres curvas senoidales de forma que la plataforma pueda realizar una figura de Lissajous en tres dimensiones. Las dos primeras curvas generan una trayectoria circular de ecuaciones:

$$X = A * sen\left(\frac{2\pi}{T} * t\right)$$
$$Y = A * \cos\left(\frac{2\pi}{T} * t\right)$$

  La ecuación de la tercera curva es equivalente a la primera ecuación senoidal, pero añadiendo un posible desfase. Los parámetros recibidos en la trama se corresponden con la amplitud y el período de las dos primeras curvas (que son iguales) y la amplitud, período y desfase de la tercera. Estas curvas se envían, en el modo “RT1” a las referencias de posición de la plataforma, y en el modo “RT2” a las de giro, a través de la misma función “giro_servos”, y se mantienen durante un tiempo determinado mientras no se reciba una nueva orden.
- Modo **demo** (caracteres iniciales “DM”): En este sencillo modo se envía a cada una de las coordenadas, por orden, una referencia positiva, otra negativa, y se vuelve a la posición inicial.

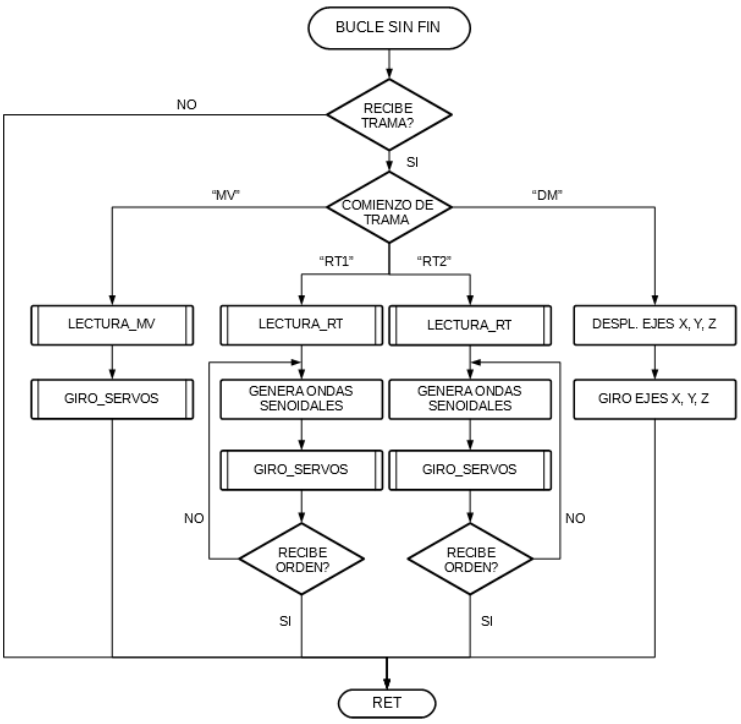


Figura 47. Diagrama de flujo III

- Lectura de trama (figura 48): La lectura de la trama para los modos de selección de posición y curvas se realiza de la misma manera, pero se ha separado en dos funciones por comodidad. En estas funciones se van extrayendo los valores contenidos en la trama utilizando las comas para distinguir entre cada uno de ellos, y se almacenan en sus variables correspondientes.

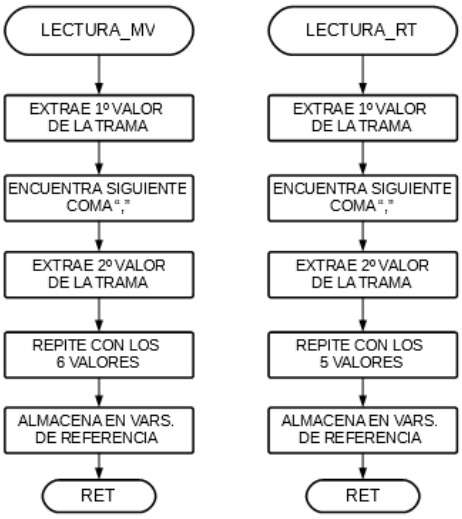


Figura 48. Diagramas de flujo IV y V

- Cálculo y envío de señales a los servos (figura 49): En la función "giro_servos" se introducen las ecuaciones obtenidas en el Apartado 2.2 de este documento, de forma que, al introducir las coordenadas de posición y giro deseadas, se obtiene el valor del ángulo de giro necesario de cada servomotor. Finalmente se envía dicho ángulo a cada uno de los servos mediante la función "set_servo", que transforma dicho ángulo en una señal PWM con un ancho de pulso determinado y la envía al servo correspondiente.

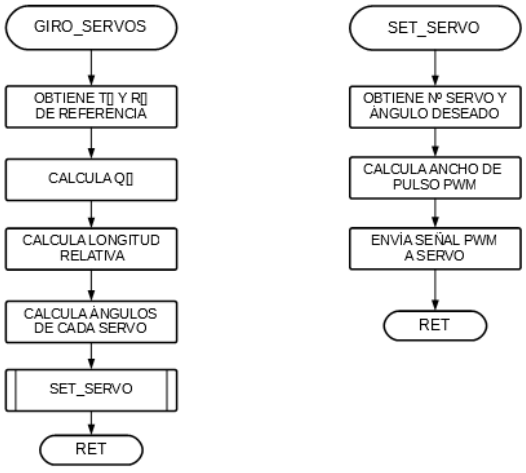


Figura 49. Diagramas de flujo VI y VII

5.4. Control remoto de la plataforma

Para controlar el movimiento de la plataforma de manera sencilla mediante un teléfono móvil, se ha creado una aplicación con la ayuda del entorno de desarrollo *App Inventor*, una plataforma gratuita creada por *Google Labs* que permite desarrollar aplicaciones sencillas para Android a través de un entorno gráfico y un sistema de programación por bloques.

Al abrir la aplicación creada, se muestra el menú de selección del modo de funcionamiento de la plataforma (figura 50). Pulsando en el botón "DEMO", la plataforma realiza la serie de movimientos definida en el punto 5.3 de este apartado, mientras que al pulsar en los botones "SEL POS" o "CURVAS", la pantalla cambia para mostrar los menús correspondientes a cada modo. En todo momento, en la parte superior de la pantalla se muestra un pequeño menú que gestiona la comunicación vía Bluetooth. Pulsando en el botón "BUSCAR", se abre una lista con los dispositivos Bluetooth guardados en el teléfono. Si el módulo HC-06 se encuentra emparejado y activo, al seleccionarlo el dispositivo móvil realiza la conexión y muestra el mensaje "Conectado" en color verde. Una vez conectada, la plataforma realiza cualquier orden que se le envíe desde la aplicación.

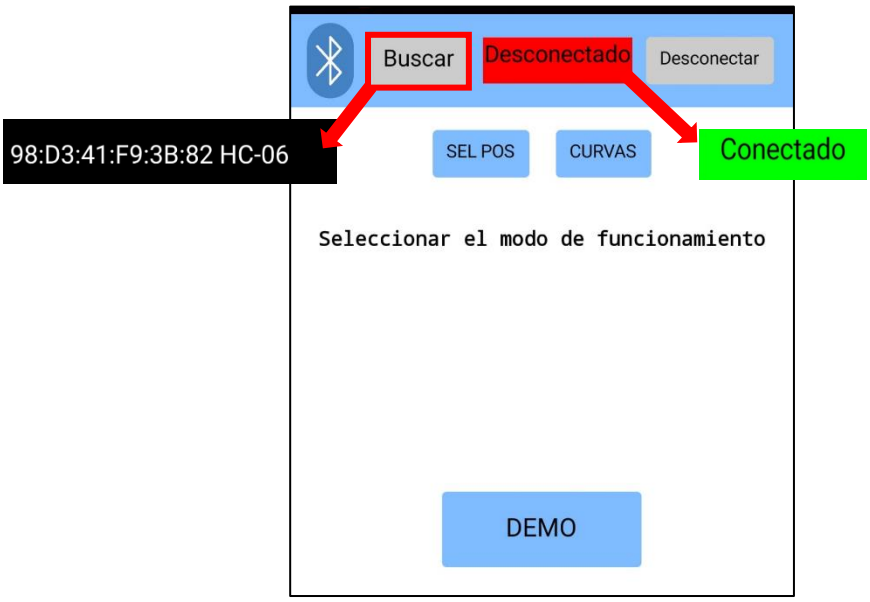


Figura 50. Menú principal de la aplicación

Al pulsar en el botón "SEL POS", se abre el menú de selección de posición (figura 51), apareciendo un total de seis deslizadores, cada uno de los cuales representa una coordenada de la plataforma. Moviendo la posición de estos deslizadores se puede seleccionar el valor deseado para cada coordenada (el valor se muestra a la derecha de cada deslizador y va cambiando a la vez que se realiza el

desplazamiento) y, una vez elegidas las coordenadas, se envían a la plataforma a través del botón "ENVIAR". Pulsando en el botón "RESET", la plataforma vuelve a la posición inicial y todos los deslizadores vuelven a situarse en la coordenada cero.

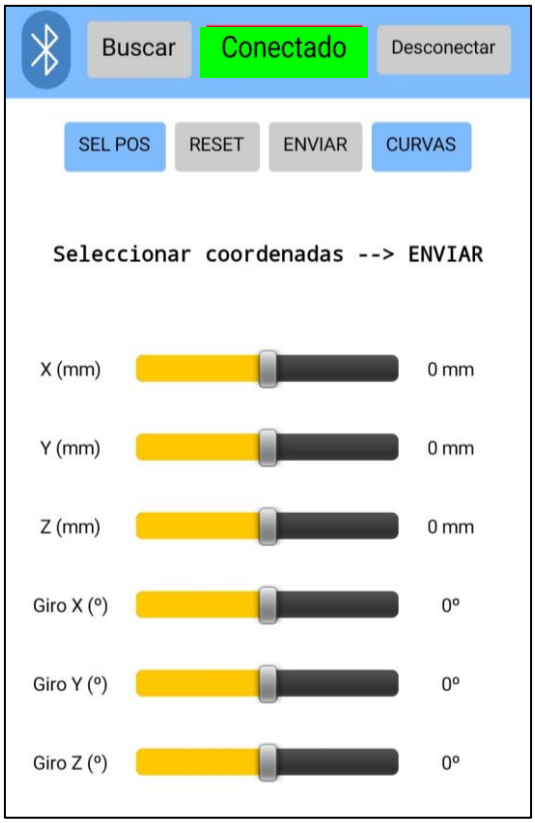


Figura 51. Menú de selección de posición de la aplicación

Al pulsar en el menú inicial el botón "CURVAS", se abre el menú de parametrización de las curvas (figura 52). En este menú, tal y como se ha detallado en el Apartado 5.3 de este documento, se pueden seleccionar los parámetros de amplitud y período de las curvas, así como el desfase entre ellas, a través de los deslizadores correspondientes. Bajo los deslizadores se encuentra un selector que permite elegir entre realizar estas curvas mediante desplazamientos o mediante giros de la plataforma. Al igual que en el menú anterior, pulsando en el botón "ENVIAR" se transmiten los parámetros a la plataforma que comenzará a realizar el movimiento durante un tiempo determinado de 100 segundos. Si se desea detener este movimiento antes de que termine, pulsar en el botón "PARAR" para que la plataforma vuelva a la posición inicial.

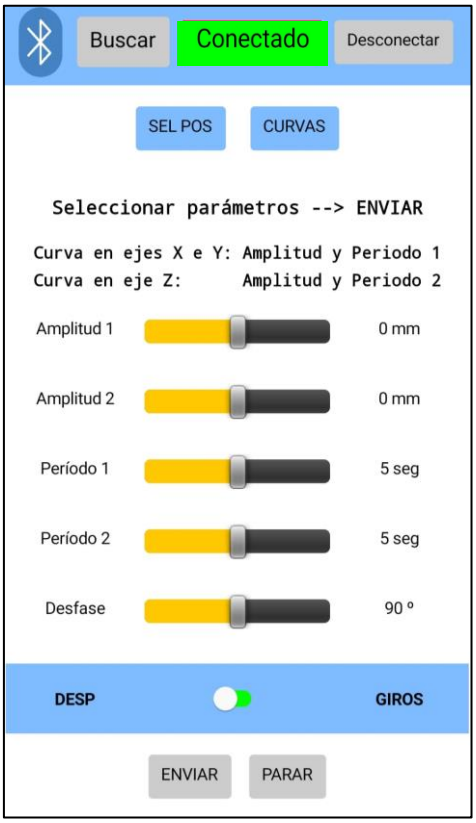


Figura 52. Menú de parametrización de curvas de la aplicación

En todo momento es posible cambiar entre el menú de selección de posición y el de curvas. Por último, pulsando en el botón "Desconectar" del menú superior el dispositivo cancela la comunicación por Bluetooth y vuelve al menú inicial.

Los bloques generados para la programación de esta aplicación se muestran por completo en el Anexo C de este documento.

5.5. Resultado de la implementación

Una vez montado y conectado, resulta complicado comprobar de una manera exacta que el prototipo funciona y realiza los movimientos deseados, ya que este no incorpora ningún sensor que permita obtener información real de los movimientos de la plataforma. No obstante, de forma visual, es posible afirmar que la plataforma tiene un funcionamiento correcto.

A continuación, se muestra la plataforma completamente montada y conectada (figura 53), así como una breve demostración de su funcionamiento, en el cual se envían una serie de referencias y se comprueba de manera visual que la plataforma es capaz de seguirlas. Se ha añadido una representación de los ejes X e Y globales a la plataforma para facilitar esta visualización.

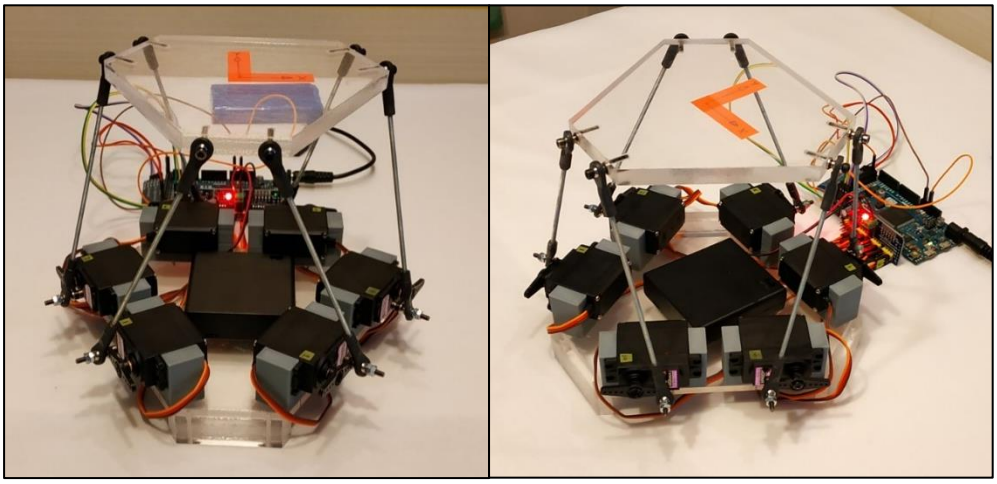


Figura 53. Plataforma Stewart implementada

En la siguiente figura, se muestra el paso de la plataforma desde la posición inicial Z = 0 mm (figura 54.a) a una posición de Z = 10mm (figura 54.b), resultado de enviar esta coordenada a través de la aplicación móvil.

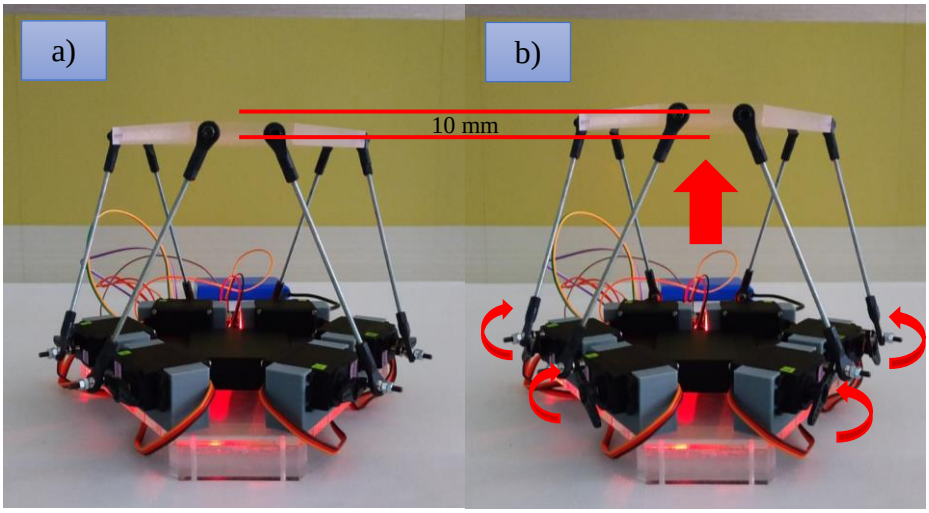


Figura 54. Desplazamiento de la plataforma en el eje Z

En la siguiente figura, se muestra el paso de la plataforma desde un giro en el eje Z inicial de α = 0º (figura 55.a) a un giro en el eje Z de α = 5º (figura 55.b), resultado de enviar esta coordenada a través de la aplicación móvil.

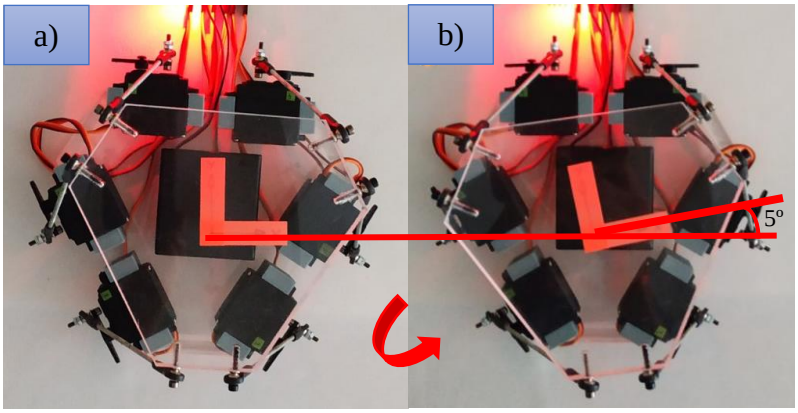


Figura 55. Rotación de la plataforma en el eje Z

En la siguiente figura, se muestra el paso de la plataforma desde una posición inicial de X = 0 mm (figura 56.a) a una posición de X = 8 mm (figura 56.b), resultado de enviar esta coordenada a través de la aplicación móvil.

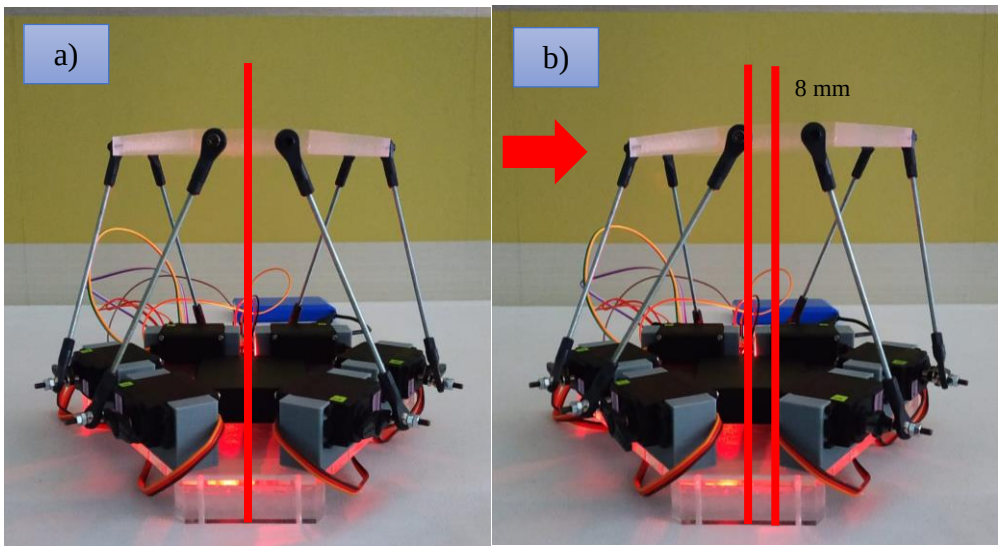


Figura 56. Desplazamiento de la plataforma en el eje X

6. CONCLUSIONES

En este Trabajo de Fin de Master se ha desarrollado con éxito el diseño y la implementación real de una Plataforma Stewart. Se ha demostrado que, tanto en el modelo simulado como en el prototipo final construido, la plataforma robótica obtenida es capaz de seguir una referencia de posición y orientación determinadas, tanto para alcanzar una posición fija como para seguir dicha referencia a lo largo del tiempo. Se considera, por lo tanto, que se han cumplido todos los objetivos definidos al comienzo de este proyecto ya que la resolución de todas sus fases ha sido satisfactoria: Se ha creado un diseño funcional a través del cual ha sido posible generar un modelo para la simulación y, a su vez, llevar a cabo la fabricación de las piezas clave para el montaje de la máquina; se ha realizado una simulación en la que se observa el comportamiento ideal de la plataforma y que ha permitido validar el diseño realizado anteriormente; por último, se ha construido un prototipo de plataforma que puede seguir de una manera adecuada una referencia, y que, pese a tener ciertas limitaciones geométricas, es capaz de realizar desplazamientos y giros en todas las direcciones gracias a sus seis grados de libertad.

Pese a considerar cumplidos los objetivos de este proyecto, resulta conveniente observar que la máquina desarrollada tiene una serie de limitaciones y carencias que pueden ser resueltas en el futuro. Con este fin, se detallan a continuación algunas recomendaciones y **propuestas de mejora** que permitirían aumentar el rendimiento y prestaciones de la plataforma desarrollada:

- Aunque, de manera visual, es posible comprobar que la plataforma construida realiza los movimientos que se le proponen, sería conveniente desarrollar un **bucle de control** mediante la adición de sensores que permitan conocer de manera precisa los valores de los desplazamientos y giros realizados por la plataforma. Esta información podría ser enviada al microcontrolador a través de una realimentación para generar un sistema de control mediante reguladores PID, asegurando de esta forma que la plataforma alcanza la referencia deseada y mejorando así su rendimiento.

- El diseño actual de la máquina permite su correcto montaje y funcionamiento. No obstante, no se ha previsto de un **correcto encapsulamiento** para la mayoría de los componentes electrónicos, que, a excepción de los servomotores, no se encuentran acoplados a la plataforma. Resultaría adecuado diseñar una pieza que, montada debajo de la base, permita encapsular y proteger el resto de componentes facilitando así el transporte de la plataforma.

7. BIBLIOGRAFÍA

[1] Stewart, D. (1965). *A Platform with Six Degrees of Freedom. Proceedings of the Institution of Mechanical Engineers, 180(1), 371–386.*
[2] Dasgupta, B., & Mruthyunjaya, T. S. (2000). *The Stewart platform manipulator: a review. Mechanism and Machine Theory, 35(1), 15–40.*
[3] Szufnarowski, F. (2013). *Stewart platform with fixed rotary actuators: a low cost design study*. *Advances in Medical Robotics, Chapter 4, 1st Ed*. (Rzeszow, Uniwersytet Rzeszowski).
[4] Furqan, M., Suhaib, M., & Ahmad, N. (2017). *Studies on Stewart platform manipulator: A review. Journal of Mechanical Science and Technology, 31(9), 4459–4470.*
[5] Nanua, P., & Waldron, K. J. (1989). *Direct kinematic solution of a Stewart platform. Proceedings, 1989 International Conference on Robotics and Automation.*
[6] Lebret, G., Liu, K., & Lewis, F. L. (1993). *Dynamic analysis and control of a stewart platform manipulator. Journal of Robotic Systems, 10(5), 629–655.*
[7] Merlet, J.-P., Gosselin, C. M., & Mouly, N. (1998). *Workspaces of planar parallel manipulators. Mechanism and Machine Theory, 33(1-2), 7–20.*
[8] Dasgupta, B., & Mruthyunjaya, T. S. (1998). *Singularity-free path planning for the Stewart platform manipulator. Mechanism and Machine Theory, 33(6), 711–725.*
[9] M. Díaz-Rodríguez and et al, *Aplicación de los robots paralelos* (2018), En *Manipuladores paralelos: Síntesis, análisis y aplicaciones (Pereira: Universidad Tecnológica de Pereira).*
[10] Robot Sysmac Delta de Omron de su web https://automation.omron.com/en/us/products/family/Sysmac%20Delta
[11] Robot M1iA/0.5A de Fanuc de su web https://www.fanuc.eu/es/es/robots/p%c3%a1gina-filtro-robots
[12] Centro de mecanizado vertical UA2090Ti de Toyoda de su web https://www.toyoda.com/machines/vertical-machining-centers/five-axis/ua2090ti
[13] Balance System SD de Biodex de su web https://www.biodex.com/physical-medicine/products/balance/balance-system-sd
[14] Girone, M., Burdea, G., Bouzit, M. *et al.* (2001). *A Stewart Platform-Based System for Ankle Telerehabilitation. Autonomous Robots 10, 203–212.*

[15] Preumont, A., Horodinca, M., … Abu Hanieh, A. (2007). *A six-axis single-stage active vibration isolator based on Stewart platform. Journal of Sound and Vibration, 300(3-5), 644–661.*
[16] Kang, C.-G. (2001). *Closed-form force sensing of a 6-axis force transducer based on the Stewart platform. Sensors and Actuators A: Physical, 90(1-2), 31–37.*
[17] F. Gao, Y. Zhang, X.C. Zhao, W.Z. Guo. (2007). *The design and applications of F/T sensor based on Stewart platform, in: Proceedings of the 12th IFToMM WorldCongress, Besancon,France.*
[18] https://www.xarg.org/paper/inverse-kinematics-of-a-stewart-platform/
[19] https://www.ucm.es/data/cont/docs/76-2013-07-11-05_Lissajous_figures.pdf
[20] https://store.arduino.cc/arduino-due
[21] https://www.amazon.es/

8. ANEXOS

A. Código de la función f_stewart en Matlab
B. Código de la programación del Arduino
C. Bloques de programación de la aplicación

A. Código de la función f_stewart en Matlab

```
%%% SIMULACIÓN PLATAFORMA STEWART

function [ang_1,ang_2,ang_3,ang_4,ang_5,ang_6] =
f_stewart(x,y,z,alphag,thetag,psig)


%% GEOMETRÍA INICIAL

% Rotación de los puntos de unión de la base
ANG_BASE=(pi/180)*[64.6 355.4 304.6 235.4 184.6 115.4];
% Rotación de los puntos de unión de la plataforma
ANG_PLAT=(pi/180)*[40.6 19.4 280.6 259.4 160.6 139.4];
% Distancia del centro de la base a los puntos de unión de la
misma en metros
R_BASE=104.89/1000;
% Distancia del centro de la plataforma a los puntos de unión de
la misma en metros
R_PLAT=90.545/1000;
% Altura inicial de la plataforma en metros
H_0=85.112/1000;
% Longitud de las varillas en metros
L_VAR=100/1000;
% Longitud de los brazos de los servos en metros
L_BRZ=15/1000;
% 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/1000;y/1000;(z/1000)+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
```

```
%% 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));

    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_1=ang(1)*180/pi;
ang_2=ang(2)*180/pi;
ang_3=ang(3)*180/pi;
ang_4=ang(4)*180/pi;
ang_5=ang(5)*180/pi;
ang_6=ang(6)*180/pi;
```

B. Código de programación del Arduino

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>

#define pi  3.14159

// Ángulos de giro límites para los servos
#define ANG_MAX pi/3
#define ANG_MIN -pi/4

// Servos en conexiones: 4, 8, 12 (sentido inverso de giro)
//                       7, 11, 15 (sentido normal de giro)
#define SERVO1 4
#define SERVO2 7
#define SERVO3 8
#define SERVO4 11
#define SERVO5 12
#define SERVO6 15


Adafruit_PWMServoDriver servos = Adafruit_PWMServoDriver();
// Declaración del objeto servos para el driver PCA9685

const uint16_t pos0 = 95;      // Ancho de pulsos PWM para la posición
de 0°
const uint16_t pos180 = 520;   // Ancho de pulsos PWM para la posición
de 180°
const uint16_t td = 10;        // Delay 1
const uint16_t td1 = 50;       // Delay 2

static float ang1,ang2,ang3,ang4,ang5,ang6;  // Ángulo de giro de cada
uno de los servos en grados

const float angBase[6] = {64.6,355.4,304.6,235.4,184.6,115.4};
// Rotación de los puntos de unión de la base
const float angPlat[6] = {40.6,19.4,280.6,259.4,160.6,139.4};
// Rotación de los puntos de unión de la plataforma
const float beta[6] = {180.0,240.0,60.0,120.0,300.0,0.0};
// Rotación de los brazos de cada servo respecto del eje X

// r_b distancia del centro de la base a los puntos de unión de la
misma en m
// r_p distancia del centro de la plataforma a los puntos de unión de
la misma en m
// h_0 altura inicial de la plataforma en m
// l_var longitud de las varillas en m
// l_brz longitud de los brazos de los servos en m
const float r_b = 104.89/1000, r_p = 90.545/1000, h_0 = 85.112/1000,
l_var = 100.0/1000, l_brz = 15.0/1000;

float Bx_b[6], By_b[6], Bz_b[6];      // Vectores que almacenan las
posiciones x,y,z de los puntos de unión de la base
float Px_p[6], Py_p[6], Pz_p[6];      // Vectores que almacenan las
posiciones x,y,z de los puntos de unión de la plataforma

// T[3] vector de traslación de la plataforma con respecto al origen
de coordenadas de la base
// R[3][3] matriz de rotación de la plataforma con respecto al origen
de coordenadas de la base
// Q[3][6] matriz que almacena las posiciones x,y,z de los puntos de
union de la plataforma en el sistema de coordenadas de la base
```

```
// L[3][6] matriz que almacena las posiciones x,y,z de los puntos de
union de los brazos de los servos
// longitud[6] vector que almacena la distancia relativa de cada punto
de unión de la base a su respectivo punto de la plataforma
// a[6],b[6],c[6],d[6] vectores que almacenan valores utilizados en el
cálculo del ángulo de giro de cada servo
// ang[6] vector que almacena el ángulo de giro de cada uno de los
servos en radianes
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];

// Variables para el tratamiento de las tramas recibidas por Bluetoth
String lectura,lectura1,lectura2,lectura3,lectura4,lectura5,lectura6;
int coma,coma_prev;

// Variables para almacenar datos de las curvas
float amp_1,amp_2,freq_1,freq_2,desfase;

const int ref_desp[3]={8,-8,0}; // Referencias de desplazamiento para
el modo DEMO
const int ref_rot[3]={5,-5,0};  // Referencias de rotación para el
modo DEMO

// Variables para almacenar la referencia de posición y giro deseada
para la plataforma
float posx,posy,posz,girox,giroy,giroz;


void setup() {
  Wire.begin();
  servos.begin();              // Inicialización del objeto servos
  servos.setPWMFreq(50);       // Frecuencia PWM: 50 Hz
  Serial.begin(9600);          // Inicialización del puerto serie para
depuración del código
  Serial1.begin(57600);        // Inicialización del puerto serie para
comunicación Bluetooth

  // Cálculo de posiciones x,y de las uniones de base y plataforma
  for (int j=0; j<=5; j++){
    Bx_b[j]=r_b*cos(angBase[j]*pi/180);
    By_b[j]=r_b*sin(angBase[j]*pi/180);
    Px_p[j]=r_p*cos(angPlat[j]*pi/180);
    Py_p[j]=r_p*sin(angPlat[j]*pi/180);
  }

  // Servos a posición inicial (Se considera una rotación de 90 grados
como posición inicial de todos los servos)
  setServo(SERVO1,90);
  delay(td1);
  setServo(SERVO2,90);
  delay(td1);
  setServo(SERVO3,90);
  delay(td1);
  setServo(SERVO4,90);
  delay(td1);
  setServo(SERVO5,90);
  delay(td1);
  setServo(SERVO6,90);
  delay(1000);

}
```

```
// FUNCIÓN DE CÁLCULO DE LA CINEMÁTICA INVERSA
void giroServos(float x,float y,float z,float alpha_grad,float
theta_grad,float psi_grad){

  // Esta función recibe los valores de las posiciones (en mm) y giros
(en grados) deseados para la plataforma en los ejes X,Y,Z y, mediante
las ecuaciones de cinemática inversa, obtiene los ángulos de giro
necesarios para cada servo y los envía

  // Valores de x,y,z al vector de traslación de la plataforma
  T[0] = x/1000;
  T[1] = y/1000;
  T[2] = z/1000 + h_0;

  // Conversión de ángulos de referencia recibidos a radianes
  float alpha = alpha_grad*pi/180;
  float theta = theta_grad*pi/180;
  float psi = psi_grad*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<=5; j++){
    Q[0][j] = T[0] + R[0][0]*Px_p[j] + R[0][1]*Py_p[j] +
R[0][2]*Pz_p[j];
    Q[1][j] = T[1] + R[1][0]*Px_p[j] + R[1][1]*Py_p[j] +
R[1][2]*Pz_p[j];
    Q[2][j] = T[2] + R[2][0]*Px_p[j] + R[2][1]*Py_p[j] +
R[2][2]*Pz_p[j];
  }

  // Cálculo de la matriz L
  for (int j=0; j<=5; j++){
    L[0][j] = Q[0][j] - Bx_b[j];
    L[1][j] = Q[1][j] - By_b[j];
    L[2][j] = Q[2][j] - Bz_b[j];
  }

  // Cálculo del vector longitud
  for (int j=0; j<=5; j++){
    longitud[j] = sqrt(pow(L[0][j],2) + pow(L[1][j],2) +
pow(L[2][j],2));
  }
```

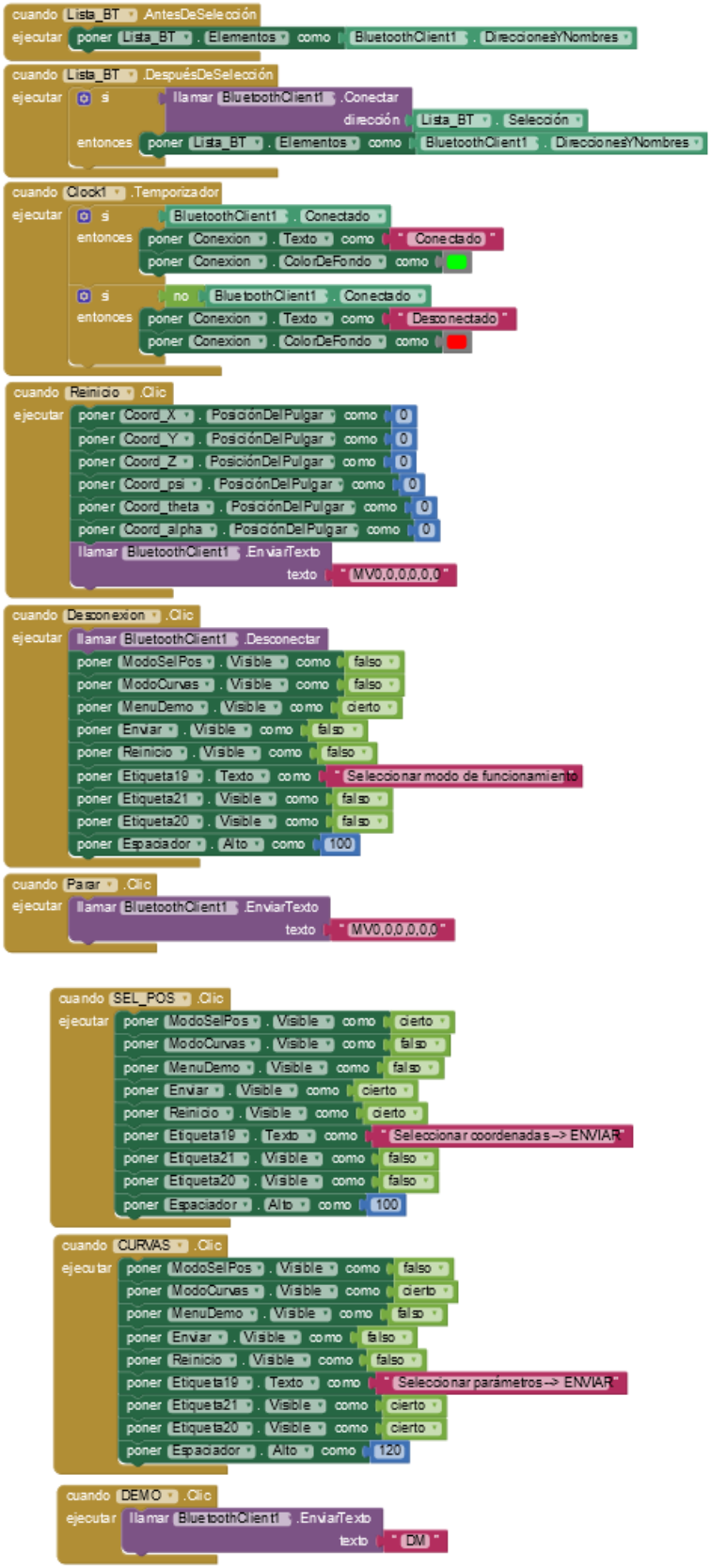
```
  // Obtención del ángulo de giro de cada servo en radianes
  for (int j=0; j<=5; j++){
    a[j] = 2 * l_brz * L[2][j];
    b[j] = 2 * l_brz * (sin(beta[j]*pi/180)*L[1][j] +
cos(beta[j]*pi/180)*L[0][j]);
    c[j] = pow(longitud[j],2) - pow(l_var,2) + pow(l_brz,2);
    d[j] = constrain(c[j]/sqrt(pow(a[j],2)+pow(b[j],2)), -1, 1);


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

  }

  // 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);
  delay(td);
  setServo(SERVO2,90+ang2);
  delay(td);
  setServo(SERVO3,90-ang3);
  delay(td);
  setServo(SERVO4,90+ang4);
  delay(td);
  setServo(SERVO5,90-ang5);
  delay(td);
  setServo(SERVO6,90+ang6);
  delay(td1);
}

// FUNCIÓN DE ENVÍO DE SEÑAL PWM A UN SERVO
void setServo(uint8_t n_servo, int angulo){

  // Recibe el valor del ángulo y el servo al que se desea enviar
dicho ángulo

  int pulsos;                                    // Ancho de pulso que
se envía al servo
  pulsos = map(angulo, 0, 180, pos0, pos180);    // Cálculo del ancho
de pulso en función del ángulo
  servos.setPWM(n_servo, 0, pulsos);             // Envía señal PWM al
servo deseado
}

// FUNCIÓN DE LECTURA DE TRAMA PARA MODO SELECCIÓN DE POSICIÓN
void lecturaMV(){

  lectura1 = lectura.substring(2,lectura.indexOf(","));
// Almacena el primer valor de la trama
  coma = lectura.indexOf(",")+1;
// Encuentra la siguiente "," en la trama
  lectura2 = lectura.substring(coma,lectura.indexOf(",",coma));
// Almacena el segundo valor de la trama
  coma_prev = coma;
```

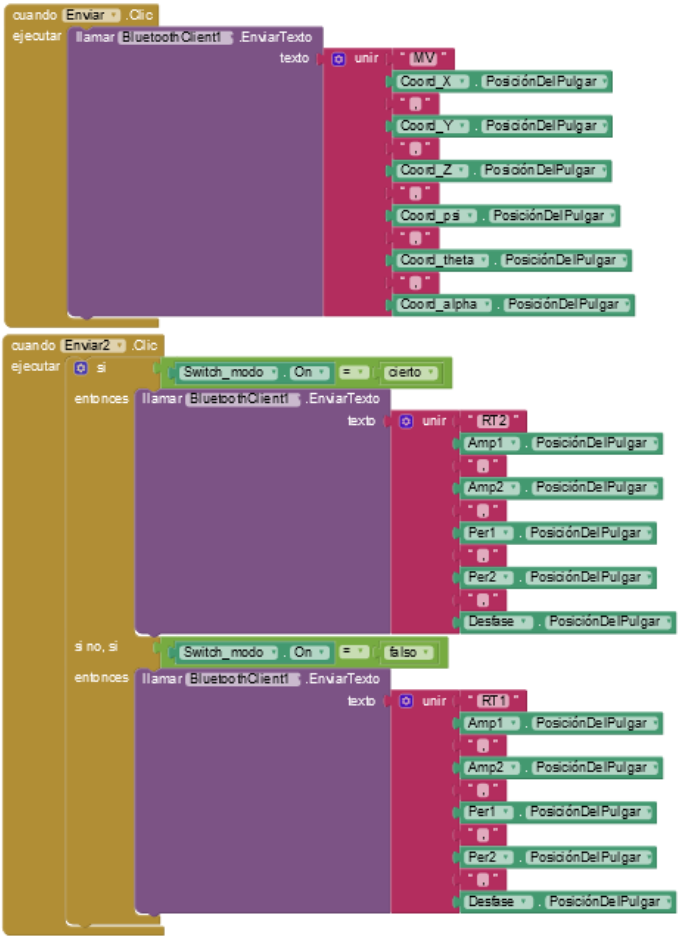
```
// ....
  coma = lectura.indexOf(",",coma_prev)+1;
  lectura3 = lectura.substring(coma,lectura.indexOf(",",coma));
  coma_prev = coma;
  coma = lectura.indexOf(",",coma_prev)+1;
  lectura4 = lectura.substring(coma,lectura.indexOf(",",coma));
  coma_prev = coma;
  coma = lectura.indexOf(",",coma_prev)+1;
  lectura5 = lectura.substring(coma,lectura.indexOf(",",coma));
  coma_prev = coma;
  coma = lectura.indexOf(",",coma_prev)+1;
  lectura6 = lectura.substring(coma,lectura.indexOf(",",coma));

  // Conversión de los valores leidos a número entero y almacenamiento
en sus respectivas variables
  posx = lectura1.toInt();
  posy = lectura2.toInt();
  posz = lectura3.toInt();
  girox = lectura4.toInt();
  giroy = lectura5.toInt();
  giroz = lectura6.toInt();

}

// FUNCIÓN DE LECTURA DE TRAMA PARA MODO CURVAS
void lecturaRT(){

  lectura1 = lectura.substring(3,lectura.indexOf(","));
// Almacena el primer valor de la trama
  coma = lectura.indexOf(",")+1;
// Encuentra la siguiente "," en la trama
  lectura2 = lectura.substring(coma,lectura.indexOf(",",coma));
// Almacena el segundo valor de la trama
  coma_prev = coma;
// ....
  coma = lectura.indexOf(",",coma_prev)+1;
  lectura3 = lectura.substring(coma,lectura.indexOf(",",coma));
  coma_prev = coma;
  coma = lectura.indexOf(",",coma_prev)+1;
  lectura4 = lectura.substring(coma,lectura.indexOf(",",coma));
  coma_prev = coma;
  coma = lectura.indexOf(",",coma_prev)+1;
  lectura5 = lectura.substring(coma,lectura.indexOf(",",coma));

  // Conversión de los valores leídos a número entero y almacenamiento
en sus respectivas variables
  amp_1 = lectura1.toInt();
  amp_2 = lectura2.toInt();
  freq_1 = 1.0/lectura3.toInt();
  freq_2 = 1.0/lectura4.toInt();
  desfase = (pi/180)*lectura5.toInt();

}
```

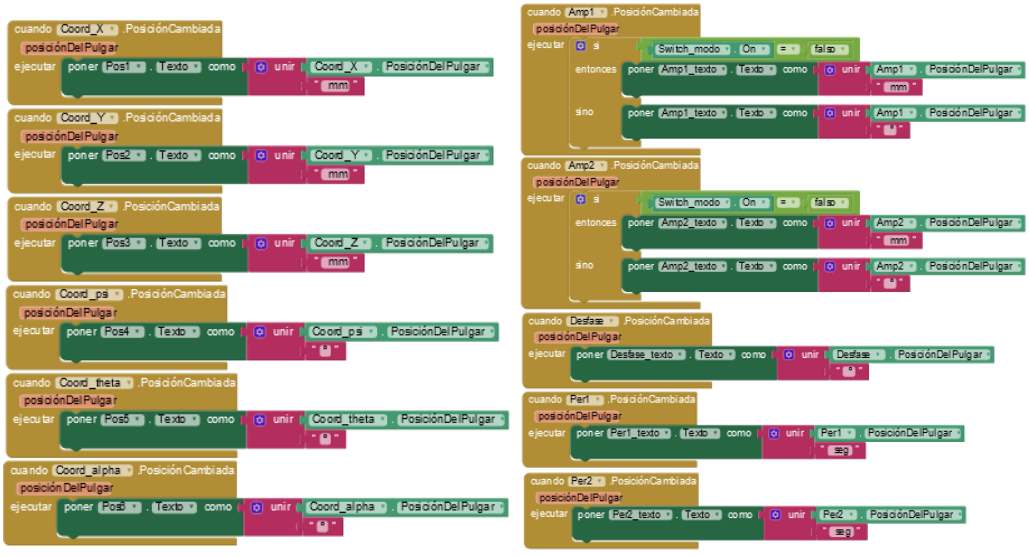
```
// BUCLE INFINITO
void loop() {

  if (Serial1.available()>0){                // Si se recibe una trama
por Bluetooth
    lectura=Serial1.readString();            // Almecena la trama en la
variable lectura

    if (lectura.startsWith("MV")){           // Si la trama comienza
con "MV" entra en modo de selección de posición
      lecturaMV();
      giroServos(posx,posy,posz,giroz,giroy,girox);

    }

    if (lectura.startsWith("RT1")){          // Si la trama comienza
con "RT1" entra en modo curvas de desplazamientos
      lecturaRT();
      for (float i=0; i<=100; i+=0.1){

        // Genera 3 ondas senoidales y las almacena en las variables
de las coordenadas de posición que se envían a la función giroServos()
        posx = amp_1*sin(2*pi*freq_1*i);
        posy = amp_1*cos(2*pi*freq_1*i);
        posz = amp_2*sin(2*pi*freq_2*i+desfase);

        giroServos(posx,posy,posz,0,0,0);
        if (Serial1.available()>0){break;}   // Si se recibe una orden
por la entrada serie del Bluetooth sale del bucle
      }

    }

    if (lectura.startsWith("RT2")){           // Si la trama comienza
con "RT2" entra en modo curvas de desplazamientos
      lecturaRT();
      for (float i=0; i<=100; i+=0.1){

        // Genera 3 ondas senoidales y las almacena en las variables
de las coordenadas de giro que se envían a la función giroServos()
        girox = amp_1*sin(2*pi*freq_1*i);
        giroy = amp_1*cos(2*pi*freq_1*i);
        giroz = amp_2*sin(2*pi*freq_2*i+desfase);

        giroServos(0,0,0,giroz,giroy,girox);
        if (Serial1.available()>0){break;}    // Si se recibe una
orden por la entrada serie del Bluetooth sale del bucle
      }

    }
```

```
    if (lectura.startsWith("DM")){            // Si la trama comienza
con "DM" entra en modo DEMO

      // Realiza un desplazamiento positivo, otro negativo y vuelve a
la posición inicial en cada uno de los ejes
      for(int i=0; i<=2; i++){
giroServos(ref_desp[i],0,0,0,0,0);delay(1000);}
      for(int i=0; i<=2; i++){
giroServos(0,ref_desp[i],0,0,0,0);delay(1000);}
      for(int i=0; i<=2; i++){
giroServos(0,0,ref_desp[i],0,0,0);delay(1000);}
      // Realiza un giro positivo, otro negativo y vuelve a la
posición inicial en cada uno de los ejes
      for(int i=0; i<=2; i++){
giroServos(0,0,0,ref_rot[i],0,0);delay(1000);}
      for(int i=0; i<=2; i++){
giroServos(0,0,0,0,ref_rot[i],0);delay(1000);}
      for(int i=0; i<=2; i++){
giroServos(0,0,0,0,0,ref_rot[i]);delay(1000);}

    }

  }


}
```

C. Bloques de programación de la aplicación


cuando Lista_BT .AntesDeSelección
ejecutar poner Lista_BT . Elementos como BluetoothClient1 . DireccionesYNombres
cuando Lista_BT .DespuésDeSelección
ejecutar si llamar BluetoothClient1 .Conectar
dirección Lista_BT . Selección
entonces poner Lista_BT . Elementos como BluetoothClient1 . DireccionesYNombres
cuando Clock1 .Temporizador
ejecutar si BluetoothClient1 . Conectado
entonces poner Conexion . Texto como " Conectado "
poner Conexion . ColorDeFondo como
si no BluetoothClient1 . Conectado
entonces poner Conexion . Texto como " Desconectado "
poner Conexion . ColorDeFondo como
cuando Reinicio .Clic
ejecutar poner Coord_X . PosiciónDelPulgar como 0
poner Coord_Y . PosiciónDelPulgar como 0
poner Coord_Z . PosiciónDelPulgar como 0
poner Coord_psi . PosiciónDelPulgar como 0
poner Coord_theta . PosiciónDelPulgar como 0
poner Coord_alpha . PosiciónDelPulgar como 0
llamar BluetoothClient1 .EnviarTexto
texto " MV0,0,0,0,0,0 "
cuando Desconexion .Clic
ejecutar llamar BluetoothClient1 .Desconectar
poner ModoSelPos . Visible como falso
poner ModoCurvas . Visible como falso
poner MenuDemo . Visible como cierto
poner Enviar . Visible como falso
poner Reinicio . Visible como falso
poner Etiqueta19 . Texto como " Seleccionar modo de funcionamiento "
poner Etiqueta21 . Visible como falso
poner Etiqueta20 . Visible como falso
poner Espaciador . Alto como 100
cuando Parar .Clic
ejecutar llamar BluetoothClient1 .EnviarTexto
texto " MV0,0,0,0,0,0 "
cuando SEL_POS .Clic
ejecutar poner ModoSelPos . Visible como cierto
poner ModoCurvas . Visible como falso
poner MenuDemo . Visible como falso
poner Enviar . Visible como cierto
poner Reinicio . Visible como cierto
poner Etiqueta19 . Texto como " Seleccionar coordenadas --> ENVIAR "
poner Etiqueta21 . Visible como falso
poner Etiqueta20 . Visible como falso
poner Espaciador . Alto como 100
cuando CURVAS .Clic
ejecutar poner ModoSelPos . Visible como falso
poner ModoCurvas . Visible como cierto
poner MenuDemo . Visible como falso
poner Enviar . Visible como falso
poner Reinicio . Visible como falso
poner Etiqueta19 . Texto como " Seleccionar parámetros --> ENVIAR "
poner Etiqueta21 . Visible como cierto
poner Etiqueta20 . Visible como cierto
poner Espaciador . Alto como 120
cuando DEMO .Clic
ejecutar llamar BluetoothClient1 .EnviarTexto
texto " DM "


cuando Enviar .Clic
ejecutar llamar BluetoothClient1 .EnviarTexto
texto unir " MV "
Coord_X . PosiciónDelPulgar
" , "
Coord_Y . PosiciónDelPulgar
" , "
Coord_Z . PosiciónDelPulgar
" , "
Coord_psi . PosiciónDelPulgar
" , "
Coord_theta . PosiciónDelPulgar
" , "
Coord_alpha . PosiciónDelPulgar
cuando Enviar2 .Clic
ejecutar si Switch_modo . On = cierto
entonces llamar BluetoothClient1 .EnviarTexto
texto unir " RT2 "
Amp1 . PosiciónDelPulgar
" , "
Amp2 . PosiciónDelPulgar
" , "
Per1 . PosiciónDelPulgar
" , "
Per2 . PosiciónDelPulgar
" , "
Desfase . PosiciónDelPulgar
si no, si Switch_modo . On = falso
entonces llamar BluetoothClient1 .EnviarTexto
texto unir " RT1 "
Amp1 . PosiciónDelPulgar
" , "
Amp2 . PosiciónDelPulgar
" , "
Per1 . PosiciónDelPulgar
" , "
Per2 . PosiciónDelPulgar
" , "
Desfase . PosiciónDelPulgar


cuando Coord_X .PosiciónCambiada
posiciónDelPulgar
ejecutar poner Pos1 . Texto como unir Coord_X . PosiciónDelPulgar " mm "
cuando Coord_Y .PosiciónCambiada
posiciónDelPulgar
ejecutar poner Pos2 . Texto como unir Coord_Y . PosiciónDelPulgar " mm "
cuando Coord_Z .PosiciónCambiada
posiciónDelPulgar
ejecutar poner Pos3 . Texto como unir Coord_Z . PosiciónDelPulgar " mm "
cuando Coord_psi .PosiciónCambiada
posiciónDelPulgar
ejecutar poner Pos4 . Texto como unir Coord_psi . PosiciónDelPulgar " º "
cuando Coord_theta .PosiciónCambiada
posiciónDelPulgar
ejecutar poner Pos5 . Texto como unir Coord_theta . PosiciónDelPulgar " º "
cuando Coord_alpha .PosiciónCambiada
posiciónDelPulgar
ejecutar poner Pos6 . Texto como unir Coord_alpha . PosiciónDelPulgar " º "
cuando Amp1 .PosiciónCambiada
posiciónDelPulgar
ejecutar si Switch_modo . On = falso
entonces poner Amp1_texto . Texto como unir Amp1 . PosiciónDelPulgar " mm "
sino poner Amp1_texto . Texto como unir Amp1 . PosiciónDelPulgar " º "
cuando Amp2 .PosiciónCambiada
posiciónDelPulgar
ejecutar si Switch_modo . On = falso
entonces poner Amp2_texto . Texto como unir Amp2 . PosiciónDelPulgar " mm "
sino poner Amp2_texto . Texto como unir Amp2 . PosiciónDelPulgar " º "
cuando Desfase .PosiciónCambiada
posiciónDelPulgar
ejecutar poner Desfase_texto . Texto como unir Desfase . PosiciónDelPulgar " º "
cuando Per1 .PosiciónCambiada
posiciónDelPulgar
ejecutar poner Per1_texto . Texto como unir Per1 . PosiciónDelPulgar " seg "
cuando Per2 .PosiciónCambiada
posiciónDelPulgar
ejecutar poner Per2_texto . Texto como unir Per2 . PosiciónDelPulgar " seg "

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

mim máster en ingeniería mecatrónica

PLANOS

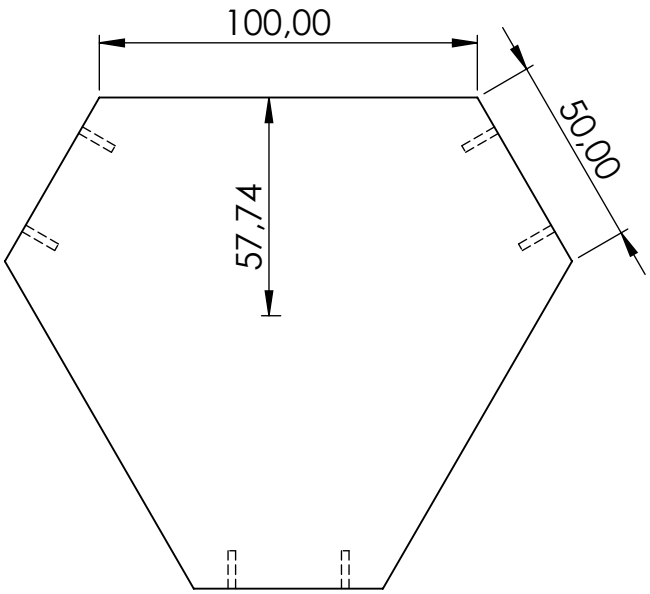
DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA STEWART

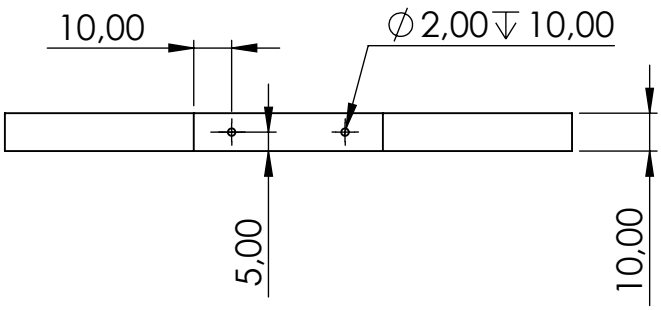
TRABAJO FINAL DEL
Máster Universitario en Ingeniería Mecatrónica

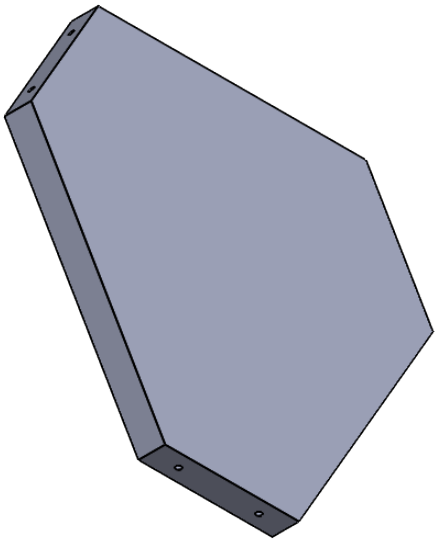
REALIZADO POR
Diego Nieves Belmar

TUTORIZADO POR
Vicente Fermín Casanova Calvo

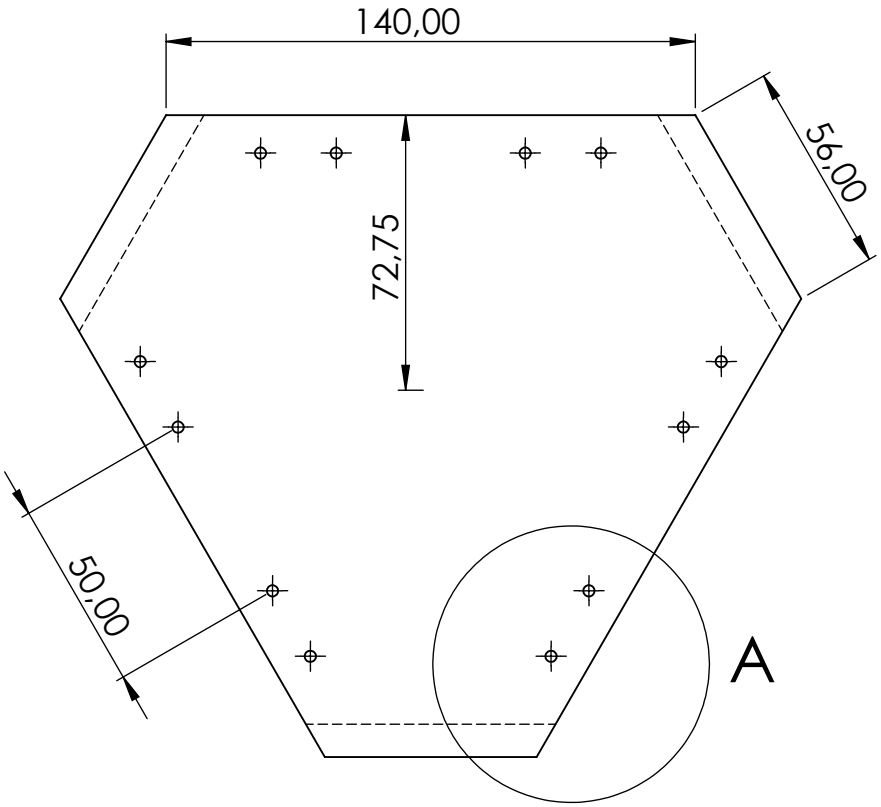
CURSO ACADÉMICO: 2019/2020

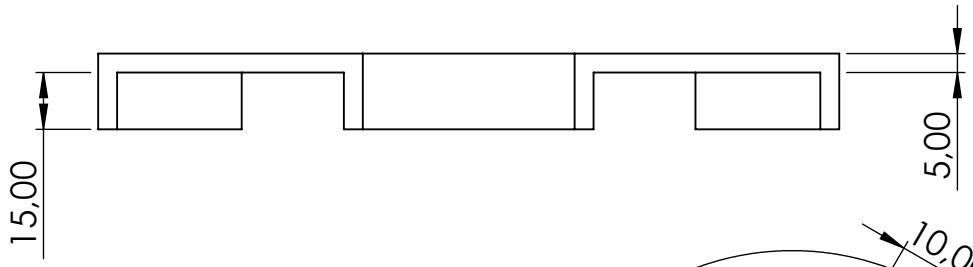


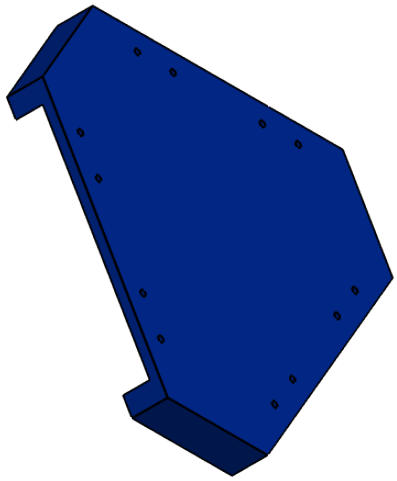


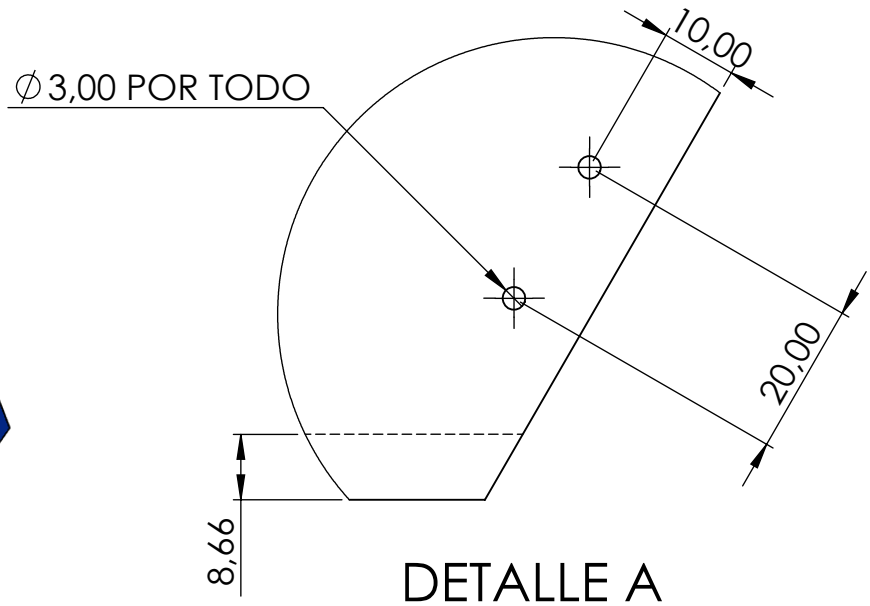


FECHA : 08/11/2020	PROYECTO : DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA STEWART		
PLANO : 1 de 4			
MATERIAL : METACRILATO			
DIBUJADO POR : Diego Nieves Belmar	DIBUJO : PLATAFORMA		A4
	ESCALA: 1:2	HOJA 1 DE 1	





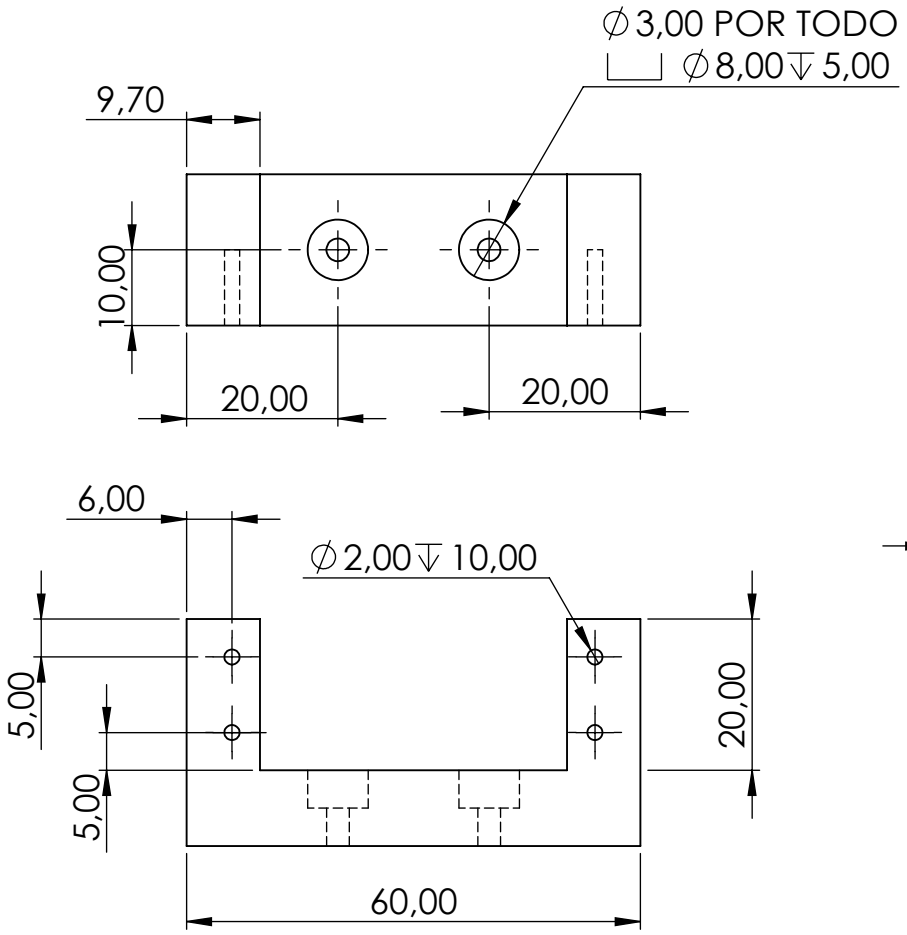


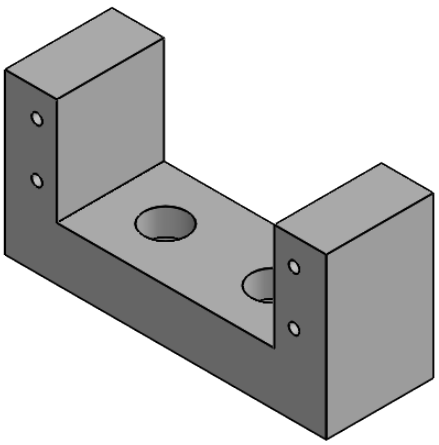


DETALLE A

ESCALA 1 : 1

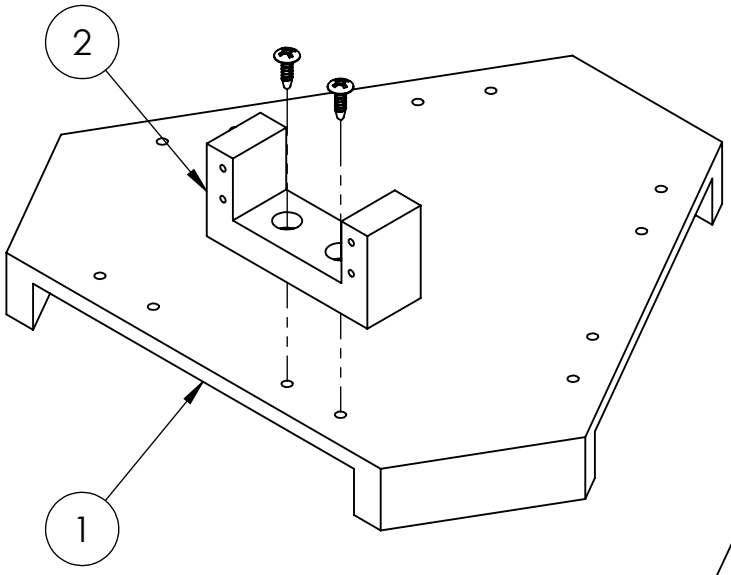
FECHA : 08/11/2020	PROYECTO : DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA STEWART		
PLANO : 2 de 4			
MATERIAL : METACRILATO			
DIBUJADO POR : Diego Nieves Belmar	DIBUJO : BASE		A4
	ESCALA: 1:2	HOJA 1 DE 1	



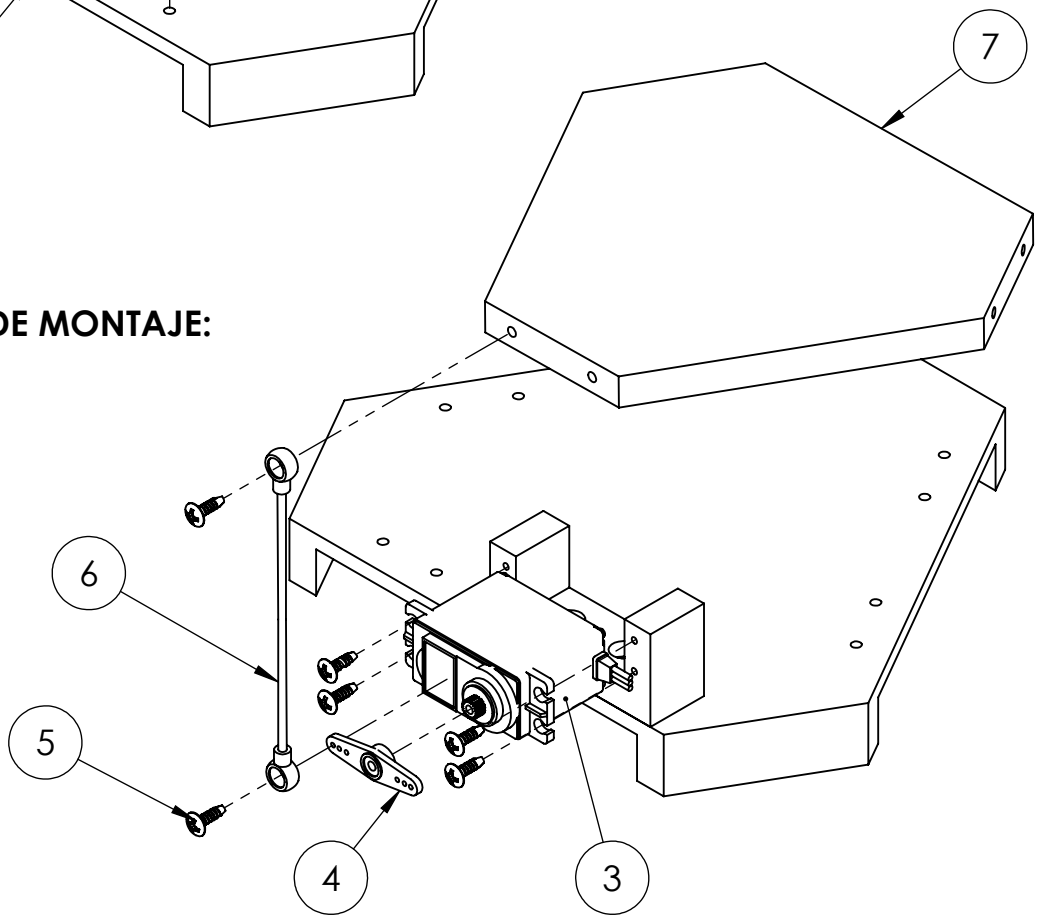


FECHA : 08/11/2020	PROYECTO : DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA STEWART		
PLANO : 3 de 4			
MATERIAL : Z-HIPS			
DIBUJADO POR : Diego Nieves Belmar	DIBUJO : SOPORTE PARA SERVO		A4
	ESCALA: 1 : 1	HOJA 1 DE 1	

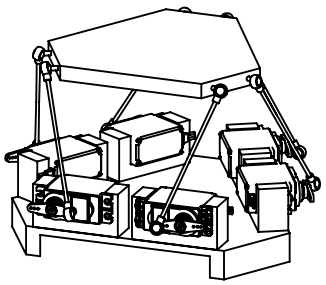
PASO 1 DE MONTAJE:



PASO 2 DE MONTAJE:



MONTAJE COMPLETO:



N.º ELEM	PIEZA	CANTIDAD
1	BASE	1
2	SOPORTE SERVO	6
3	SERVO MG996R	6
4	BRAZO SERVO	6
5	PERNO M3X10	48
6	VARILLA	6
7	PLATAFORMA	1

FECHA : 08/11/2020	PROYECTO : DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA STEWART	
PLANO : 4 de 4		
MATERIAL : VARIOS		
DIBUJADO POR : Diego Nieves Belmar	DIBUJO : PLANO DE MONTAJE	A4
	ESCALA: 1 : 2	HOJA 1 DE 1

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

PRESUPUESTO

DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA STEWART

TRABAJO FINAL DEL
Máster Universitario en Ingeniería Mecatrónica

REALIZADO POR
Diego Nieves Belmar

TUTORIZADO POR
Vicente Fermín Casanova Calvo

CURSO ACADÉMICO: 2019/2020

ÍNDICE GENERAL

1. Materiales

El presupuesto para los materiales utilizados en este proyecto se divide en:

- Materiales comprados a un proveedor externo
- Materiales utilizados en la fabricación de piezas
- Coste de fabricación de piezas

1.1. Compra de materiales

Los precios de los componentes que se muestran a continuación pueden variar en función del proveedor elegido.

Descripción	Coste unitario (€/ud)	Cantidad (unidades)	Coste total (€)
Servomotor MG996R	5,20	6	31,19
Placa Arduino Due	35,00	1	35,00
PCA9685 PWM Servo driver	7,00	1	7,00
Módulo Bluetooth HC-06	7,48	1	7,48
Varilla roscada M3 x 100	0,15	6	0,90
Rótula 2,5 x 27 x M3	1,41	12	16,92
Perno M3 x 10	0,05	48	2,40
Cable de conexión	0,10	8	0,80
Carcasa porta pilas AA x 4	2,50	1	2,50
Pila AA 1,5 V	0,75	4	3,00
Batería LiPo externa 7,4 V	11,99	1	11,99
TOTAL			**119,17**

1.2. Materiales para fabricación

El coste de los materiales utilizados para la fabricación de piezas se ha calculado en función del peso. El precio del material utilizado puede variar en función del proveedor elegido.

Descripción	Coste unitario (€/g)	Cantidad (g)	Coste total (€)
Plataforma	0,01	250	2,44
Base	0,01	300	2,92
Soporte para el servomotor	0,04	91	3,67
TOTAL			9,02

1.3.Coste de fabricación

El coste de fabricación de las piezas se ha calculado en función del tiempo de utilización de las máquinas. Se ha estimado un precio de 3.5 €/hora para la cortadora láser y de 5 €/hora para la impresora 3D.

Descripción	Coste unitario (€/h)	Cantidad (h)	Coste total (€)
Plataforma	3,50	0,5	1,75
Base	3,50	0,2	0,70
Soporte para el servomotor	5,00	23	115,00
TOTAL			117,45

2. Mano de obra

El coste de la mano de obra se ha calculado en función del tiempo aproximado invertido en las diferentes tareas del proyecto, estimando un salario de aproximadamente 20 €/hora.

Descripción	Coste unitario (€/h)	Cantidad (h)	Coste total (€)
Diseño	50,00	20	1000,00
Simulación	20,00	20	400,00
Programación	20,00	20	400,00
Montaje	3,00	20	60,00
Pruebas	15,00	20	300,00
TOTAL			760,00

3. Presupuesto total

El presupuesto final del proyecto asciende a una cantidad de **1005.65 €** (no se han tenido en cuenta precios de licencias de software ya que se han utilizado siempre licencias educacionales gratuitas)

Descripción		Coste total (€)
Materiales	Comprados	119,17
	Para fabricación	9,02
	Máquinas	117,45
Mano de obra		760,00
TOTAL		**1005,65**

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

mim máster en ingeniería mecatrónica

PLIEGO DE CONDICIONES

DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA STEWART

TRABAJO FINAL DEL
Máster Universitario en Ingeniería Mecatrónica

REALIZADO POR
Diego Nieves Belmar

TUTORIZADO POR
Vicente Fermín Casanova Calvo

CURSO ACADÉMICO: 2019/2020

ÍNDICE GENERAL

1. PLIEGO DE CONDICIONES GENERALES

1.1. Objeto

El objeto de este documento es recoger las especificaciones técnicas y legales que se han de seguir a la hora de llevar a cabo la ejecución del proyecto "Diseño e implementación de una Plataforma Stewart".

El cumplimiento de lo recogido en este documento es obligatorio para todos los agentes involucrados en la fabricación y montaje de la plataforma. Si en dicho proceso se realiza cualquier cambio no autorizado no será posible garantizar un correcto funcionamiento de la máquina.

1.2. Documentos del proyecto

El presente proyecto está constituido por los siguientes documentos:

- Documento 1: Memoria
- Documento 2: Anexos
- Documento 3: Planos
- Documento 4: Presupuesto
- Documento 5: Pliego de condiciones

Todos los documentos que integran el proyecto son compatibles entre sí, además de complementarse los unos a los otros. En caso de existir contradicciones, se establece un orden de prioridad entre los documentos, siendo los Planos el documento principal, seguido del Pliego de condiciones, la Memoria, el Presupuesto y los Anexos.

2. PLIEGO DE CONDICIONES TÉCNICAS

A continuación, se detallan las condiciones técnicas que se deben seguir para llevar a cabo la ejecución del presente proyecto, así como las condiciones que deben cumplir los materiales y herramientas utilizados. Todas las tareas realizadas y todos los materiales utilizados deben cumplir con la normativa vigente.

2.1. Especificaciones generales

Los documentos que integran este proyecto reúnen los datos necesarios para llevar a cabo las tareas de las que se compone. Estas tareas son: Diseño del prototipo, Simulación, Selección y compra de componentes, Fabricación de piezas, Montaje del prototipo, Programación del microcontrolador, Depuración y pruebas.

Cualquier cambio o modificación en los cálculos o en los procedimientos descritos sin autorización significará la pérdida de garantía de un correcto funcionamiento de la plataforma.

2.2. Descripción de los materiales

Todos los materiales utilizados deben cumplir con los requerimientos técnicos especificados en el proyecto. En el caso de ser necesaria la sustitución de algún material esta debe ser previamente autorizada, siendo las características técnicas del material sustitutivo idénticas o en su defecto superiores.

2.2.1. Filamento de impresión Z-Hips

El Z-Hips es un polímero termoplástico con una gran resistencia al impacto que se adapta fácilmente a los retoques y puede ser lijado, pintado y encolado.

Las especificaciones técnicas para este filamento son las siguientes:

- Fuerza de tensión: 16.9 MPa
- Rotura por estrés: 13.02 MPa
- Elongación en tensión máxima: 1.87 %
- Elongación de rotura: 7.75 %
- Dureza Shore (D): 73.2

Los parámetros de impresión son los siguientes:

- Altura de capa: 0.14 mm
- Diámetro de boquilla: 0.4 mm
- Relleno: 30 %

2.2.2. Metacrilato

El metacrilato es un material plástico transparente, de alta resistencia al impacto, resistente a la intemperie, aislante térmico y acústico, duro, ligero, y de fácil moldeo y mecanizado.

Las especificaciones técnicas para este material son las siguientes:

- Densidad: 1.18 g/cm^3
- Punto de fusión: 160 ºC
- Grosor de las placas: 5 y 10 mm

2.2.3. Arduino Due

El Arduino Due es una placa basada en un microcontrolador ARM de 32 bits. Su hardware consiste, entre otras cosas, en 54 pines entrada/salida digitales (12 de los cuales pueden ser usados como señales PWM), 12 entradas y 2 salidas analógicas y 4 puertos serie.

Las especificaciones técnicas para este componente electrónico son las siguientes:

- Tensión de alimentación: 7V – 12V
- Voltaje de operación: 3.3 V
- Corriente de salida en líneas I/O: 130 mA
- Corriente de salida para el pin 3.3 V: 800 mA
- Corriente de salida para el pin 5 V: 800 mA

2.2.4. Módulo HC-06

Módulo para la comunicación Bluetooth con la placa Arduino. El fabricante de este componente no se especifica en la memoria y por tanto podrá utilizarse

cualquiera que sea compatible con el microcontrolador y que cumpla con las especificaciones técnicas.

Las especificaciones técnicas para este componente electrónico son las siguientes:

- Tensión de alimentación: 3.3V – 6V

2.2.5. Servo motor MG996R

Un servomotor es un actuador rotativo que permite un control preciso mediante señales PWM en términos de posición angular, aceleración y velocidad. Los servomotores utilizados son del fabricante AZDelivery y disponen de engranajes metálicos.

Las especificaciones técnicas para este componente electrónico son las siguientes:

- Tensión de alimentación: 4.8V – 7.2V
- Consumo de corriente: 500 mA
- Rotación aproximada: 120º
- Velocidad de operación: 0.14 s/60º alimentados a 6 V

2.2.6. Controlador PCA9685

Driver compatible con el microcontrolador Arduino utilizado para el control simultáneo de varios servomotores.

Las especificaciones técnicas para este componente electrónico son las siguientes:

- Tensión de alimentación: 3V – 5V
- Nº de salidas PWM: 16
- Resolución de señales PWM: 12 bits
- Tensión máxima de alimentación externa: 6 V

2.3. Descripción del prototipo

El prototipo final debe constar de todas las piezas y componentes necesarios para el funcionamiento de la plataforma. Debe comprobarse que están incluidos todos los componentes que se detallan a continuación:

- 1 plataforma
- 1 base
- 6 varillas roscadas M3 x 100
- 12 rótulas 2.5 x 27 x M3
- 6 servomotores MG996R
- 48 pernos M3 x 10
- 1 placa Arduino Due
- 1 controlador PCA9685
- 1 módulo HC-06
- 8 cables de conexión
- 1 carcasa porta pilas para 4 pilas AA
- 4 pilas AA de 1.5 V
- 1 batería externa LiPo de 7.4 V y 6000 mAh con cargador incluido

2.4. Condiciones de uso

AVISO: Leer completamente este pliego de condiciones y seguir las recomendaciones dadas para una correcta utilización del prototipo. Todo daño causado a la plataforma o a cualquiera de sus componentes derivado de un uso inadecuado será responsabilidad del usuario.

2.4.1. Recomendaciones generales

1- Esta plataforma puede ser utilizada por menores de edad o personas con capacidades físicas o sensoriales reducidas o con falta de experiencia o conocimiento siempre y cuando se encuentren bajo la supervisión de una persona responsable y siempre que hayan recibido instrucciones adecuadas para el uso seguro de la misma.

2- Esta plataforma se ha concebido únicamente para uso doméstico y/o docente, y queda por tanto fuera de las condiciones de este proyecto su utilización en cualquier ámbito comercial o profesional.

2.4.2. Recomendaciones antes del uso

1- Antes de conectar la plataforma, asegurarse de que se encuentra en un entorno bien iluminado, limpio y seco, y de que ninguno de sus componentes se encuentra mojado.
2- Asegurarse de tener las manos bien secas para evitar mojar algún componente al manipularlo.
3- Comprobar que todos los componentes electrónicos, pines y cables de conexión no se encuentran deteriorados. En caso de observar algún componente averiado o en mal estado se debe avisar a la persona responsable y consultar con un técnico.
4- Comprobar que la batería y las pilas se encuentran en buen estado y cargadas. De no ser así, cargar la batería o sustituir las pilas por otras nuevas de las mismas características.

2.4.3. Recomendaciones durante el uso

1- No utilizar nunca cerca de chimeneas, estufas o cualquier otra fuente de calor. La temperatura de uso recomendada se sitúa entre 5ºC y 35ºC. En condiciones de temperatura extremadamente altas o bajas los componentes electrónicos podrían no funcionar adecuadamente o deteriorarse.
2- Colocar la plataforma en una superficie plana. Si esta se utiliza en una superficie inadecuada podría caerse causando un posible daño físico a la persona que la esté utilizando o una avería en la propia máquina.
3- No utilizar cerca de objetos o sustancias que puedan caer sobre la plataforma o sus componentes causando daños o averías.
4- No situar encima de la plataforma objetos o sustancias que puedan caer sobre ella o sus componentes causando daños o averías
5- No transportar la plataforma de una superficie a otra mientras se encuentre conectada y funcionando.

6- No tocar los cables ni los componentes electrónicos mientras la plataforma se encuentre en funcionamiento o conectada a una fuente de tensión.
7- Si durante la utilización de la plataforma esta sufre alguna avería o deja de funcionar adecuadamente, desconectar la batería y las pilas, avisar a la persona responsable y consultar con un técnico.
8- Cuando se haya terminado de utilizar la plataforma, desconectar la batería y las pilas para evitar que estas se consuman de forma innecesaria.

2.4.4. Desconexión

Desconectar la plataforma siempre que se haya terminado su utilización, o si durante el funcionamiento esta muestra un comportamiento inestable o deja de funcionar, o si se observa algún desperfecto o avería en la plataforma o en cualquiera de los componentes.

Para desconectar la plataforma completamente:

1- Desconectar las pilas. Para ello, colocar en posición “OFF” el interruptor que se encuentra sobre la superficie de la carcasa porta pilas.
2- Desenchufar la batería externa del Arduino. Para ello, tirar suavemente de la clavija, no del cable, hasta que esta quede totalmente desconectada.

2.4.5. Carga de la batería

La temperatura ambiental recomendada para la carga de la batería se sitúa entre los 5ºC y los 35ºC. En condiciones de temperatura extremadamente bajas o altas la batería podría no cargarse, o no hacerlo adecuadamente.

Enchufar la batería al cargador a través del cable de alimentación, y este a una toma de corriente doméstica para iniciar el proceso de carga.

El cargador dispone de un led que muestra el estado de la batería. La luz del led será de color rojo cuando la batería se esté cargando. La luz será de color verde cuando la batería esté completamente cargada.

No desconectar la batería de la toma de corriente hasta que esta se encuentre completamente cargada.

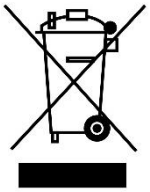
2.4.6. Cambio de las pilas

Las pilas incluidas no son recargables, por lo que, en caso de descarga, estas deben ser sustituidas por otras nuevas de las mismas características.

Para realizar la sustitución de las pilas no es necesario desconectar los cables de la carcasa porta pilas de los bornes del controlador. No obstante, manipular con cuidado de no dañar estos cables y asegurarse de que el interruptor de la carcasa se encuentra en posición “OFF” antes de realizar el cambio.

Para realizar la sustitución de las pilas, es necesario extraer el tornillo que cierra la tapa de la carcasa porta pilas con la ayuda de un destornillador de estrella. Una vez extraído el tornillo, quitar la tapa y sacar las pilas, colocando en su lugar las pilas nuevas en la posición adecuada. Esta posición se especifica en el propio hueco disponible para las pilas. Introducidas las nuevas pilas, cerrar la tapa y volver a enroscar el tornillo en su sitio.

2.4.7. Eliminación de desechos



(Residuos de aparatos eléctricos y electrónicos)
(Aplicable en la Unión Europea y en países europeos con sistemas recogida selectiva de residuos)

No tirar este producto ni ninguno de sus accesorios electrónicos (cargador, cables, pilas, etc.) al final de su vida útil a la basura o junto con otros residuos domésticos. Para evitar los posibles daños al medio ambiente o a la salud humana, separar estos desechos de otro tipo de artículos y reciclarlos correctamente en los puntos habilitados para ello.