DISEÑO DE SIMULADOR DE VUELO DINÁMICO PARA UN AERONAVE DE ENTRENAMIENTO PRIMARIO

EIBAR FABIÁN TAPIA GARCÉS

UNIVERSIDAD DEL CAUCA
FACULTAD DE CIENCIAS NATURALES, EXACTAS Y DE LA EDUCACIÓN
DEPARTAMENTO DE FÍSICA
POPAYÁN
2019

DISEÑO DE SIMULADOR DE VUELO DINÁMICO PARA UN AERONAVE DE ENTRENAMIENTO PRIMARIO

TRABAJO DE GRADO PARA OPTAR AL TÍTULO DE INGENIERO FÍSICO

EIBAR FABIÁN TAPIA GARCÉS

DIRECTOR
ING. MARIO ANDRÉS CÓRDOBA GONZALES

UNIVERSIDAD DEL CAUCA
FACULTAD DE CIENCIAS NATURALES, EXACTAS Y DE LA EDUCACIÓN
DEPARTAMENTO DE FÍSICA
POPAYÁN
2019

Director:

Ing. Mario Andrés Córdoba

Jurado:

Ing. Lina Marcela Jaller

Jurado:

Mg. Jorge Whashington Coronel

Nota de aceptación

Popayán, marzo 27 de 2019

Sé feliz haciendo lo que llene tu alma.

A mi heredero: Santiago Tapia.

Agradecido de manera infinita con el esfuerzo y la paciencia de mis padres Berta Garcés y Pionono Tapia; quienes a través de los años han apoyado cada proyecto que he iniciado.

Contenido

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE GRÁFICAS

RESUMEN

En este trabajo se diseña un simulador de vuelo dinámico para una aeronave de entrenamiento primario basado en el manipulador paralelo Plataforma Stewart capaz de generar movimientos en 6 grados de libertad lo cual permite emular el movimiento producido por una aeronave en vuelo. El software de simulación utilizado es X Plane 10 debido a que permite obtener datos de posición y orientación de la aeronave simulada y además es un software certificado por la FAA para ser utilizado en sistemas de entrenamiento para pilotos.

El diseño de dicha plataforma se lleva a cabo en un software de diseño mecánico de la compañía Dassault Systèmes conocido como SolidWorks. Para codificar el modelo cinemático del Robot fue apropiado conocer conceptos de dinámica directa y dinámica inversa; siendo la última la que resultó útil para dicho trabajo.

Con la ayuda del Software especializado se determinan los valores de los ángulos de orientación y los valores de posición en cada grado de libertad del robot.

Se utilizan actuadores rotacionales para generar los movimientos del robot mencionado, los cuales cumplen con parámetros de torque y potencia determinados en un estudio de movimiento llevado a cabo en Solidworks

Palabras clave: Simulador de vuelo, Inmersividad, Plataforma Stewart, XPlane 10, Solidworks, Cessna 172.

INTRODUCCIÓN

Definición del problema

El Grupo de Investigación en Ingeniería Aeronáutica y Aviación de la Universidad del Cauca no cuenta con un sistema de simulación de vuelo que permita realizar prácticas donde se apliquen conceptos de dinámica de vuelo, aeronavegación, manejo de instrumentos de vuelo, y de otros parámetros propios de la disciplina, parámetros que deben ser tenidos en cuenta en el proceso de diseño de un aeronave y cuando se está pilotando la misma, es por eso que se hace necesario el diseño de un simulador de vuelo y su posterior construcción para poder llevar a cabo prácticas donde se evidencien los conceptos antes mencionados.

Entre las categorías para Simuladores de vuelo descritas por la Federal Aviation Adminstration (FAA) y para el caso de Colombia, descritas en la norma RAC 24 de la Unidad Administrativa Especial de Aeronáutica Civil, es la categoría D, en la que se fija la atención, puesto que es la categoría más alta en este tipo de sistemas de entrenamiento para pilotos, este tipo de simuladores se conforma de un sistema generador de movimientos (para los simuladores categoría D, debe generar movimientos en 6 grados de libertad (Pitch, Roll, Yaw, Heave, Surge y Sway)), un software para simulación de vuelo, instrumentos de navegación aérea y sistemas visuales (pantallas, proyectores, sistemas de realidad aumentada, etc.).

Entre los sistemas para generar movimientos en los 6 grados de libertad mencionados, existe un robot paralelo conocido como plataforma Stewart, el cual es ampliamente usado en simuladores de vuelo y de movimiento en general. Este robot puede estar conformado por actuadores lineales (generalmente hidráulicos) o por actuadores rotacionales eléctricos, siendo éstos últimos fácilmente controlables y con características de precisión (en cuanto a control de posición), mejores a las de los actuadores lineales. Para diseñar un robot de este tipo, se debe determinar su modelo cinemático, el cual es útil para conocer el rango de movilidad en cada grado de libertad; dichos rangos se establecen según recomendaciones de expertos en el diseño y fabricación de simuladores de vuelo, descritos en Allerton David. (2009).

Para llevar a cabo pruebas que permitan determinar los rangos de movilidad mencionados anteriormente, se hace necesario construir un prototipo del robot ‘plataforma Stewart’. Para el presente proyecto se decide por construir un prototipo a escala del robot, con el fin de reducir costos de construcción.

Es necesario también definir el tipo de aeronave para la cual se diseña el simulador de vuelo, puesto que esto permite fijar parámetros de diseño tales

como, dimensiones de la cabina de entrenamiento, carga que debe soportar el sistema generador de movimientos, tamaño del sistema generador de movimientos, entre otros. Al ser éste el primer acercamiento al diseño de este tipo de sistemas, se decide diseñar el simulador para un aeronave de entrenamiento primario, específicamente para la aeronave Cessna 172 usada comúnmente en las escuelas de aviación de todo el planeta, debido a que es una aeronave con características de estabilidad, velocidad, maniobrabilidad y rendimiento, adecuadas para los vuelos iniciales de práctica de los futuros pilotos.

La primera versión de dicho simulador se diseña para un único tripulante, lo que reduce la carga que debe soportar el sistema generador de movimiento, comparada con un simulador para dos tripulantes.

Cuando se construye un simulador de vuelo para entrenamiento de pilotos, se busca que éste pueda ser certificado por la FAA o la autoridad encargada del control de aeronavegación de cada país, para su uso, esto se logra cumpliendo una serie de requisitos, según la categoría del simulador, entre estos requisitos está el uso de software para simulación de vuelo ya certificado para tal fin, uno de los softwares calificados es X Plane. Otro de los requisitos tiene que ver con el hardware; en específico, los controladores y procesadores también deben estar certificados para el uso en sistemas que se usen para la aeronavegación, aunque en este primer diseño no se tiene en cuenta dicho requisito; por lo tanto, se utiliza una plataforma de desarrollo conocida como Arduino, la cual tiene puertos de comunicación compatibles con el software de simulación elegido.

Impacto

El Grupo de Investigación en Ingeniería Aeronáutica y Aviación de la Universidad del Cauca tiene como objetivo a mediano plazo la creación del programa de pregrado en Ingeniería Aeronáutica, es por eso que está interesado en el diseño y la construcción de un sistema de simulación de vuelo que permita hacer prácticas de vuelo donde se apliquen conceptos básicos de aeronavegación, manejo de instrumentos, control de superficies de sustentación en un aeronave y muchas otras experiencias propias de la disciplina; además, el grupo de investigación en mención tiene como pilar de su desarrollo el proyecto "DISEÑO Y CONSTRUCCIÓN DE UN AERONAVE TIPO LSA PARA ENTRENAMIENTO PRIMARIO"; esto quiere decir que el Simulador de Vuelo que se planea diseñar y posteriormente construir, será de gran ayuda para analizar el comportamiento de esta aeronave cuando se tenga el modelo matemático del mismo, puesto que el software para simulación de vuelo utilizado, permite incluir nuevas aeronaves a su entorno para probar su desempeño en vuelo.

La elección del Simulador virtual de vuelo X-Plane 10 se hace porque este simulador está certificado por la Federal Aviation Administration (FAA) como simulador de entrenamiento para pilotos, lo que le permite al simulador de vuelo ser utilizado también por estudiantes de escuelas de aviación para hacer las horas de simulador de vuelo que se requieren en su entrenamiento.
El Software elegido permite incluir nuevas aeronaves a su entorno para probar el desempeño en el aire de la nueva aeronave; esto le da al simulador de vuelo un papel fundamental en el desarrollo de las investigaciones dentro del grupo de Investigación en Ingeniería Aeronáutica y Aviación de la Universidad del Cauca.

OBJETIVOS

Objetivo general

- Diseñar un simulador de vuelo dinámico para un aeronave de entrenamiento primario con interfaz al simulador virtual de vuelo X-Plane 10

Objetivos específicos

- Obtener datos de posición y orientación desde el simulador virtual de vuelo X-Plane 10.
- Diseñar una plataforma Stewart para usarla como base del simulador de vuelo
- Construir una plataforma Stewart a escala de la plataforma diseñada en el paso anterior
- Diseñar un sistema de control para la plataforma Stewart e implementarlo en la plataforma a escala
- Diseñar la estructura donde estarán ubicados los elementos de la cabina de un Cessna 172
- Elaborar un presupuesto para la construcción del simulador de vuelo en tamaño real

PARTE 1: MARCO TEÓRICO

1. SIMULADORES DE VUELO

En este capítulo se hace una introducción al mundo de la simulación en general respondiendo a las preguntas: ¿Qué es simular?, ¿Para qué se simula?, ¿Cómo se simula?, ¿Hasta dónde simular?; se define el concepto de inmersividad el cual es muy importante en el mundo de los simuladores de vuelo; se hace una definición para simuladores de vuelo; en seguida, se hace un repaso histórico del desarrollo de estos sistemas, se hace una clasificación y se describe los subsistemas que componen un simulador de vuelo dinámico.

1.1 Simulación.

El proceso de simulación se describe como el intento de replicar un universo físico de interés, para poder entenderlo y ejercer control sobre él, sin causar efectos a terceros. En la ingeniería, se desarrollan sistemas de simulación por medio de modelos matemáticos que permiten tener en cuenta o no parámetros que afectan directamente al universo de interés; eso hace referencia a que siempre se debe simular hasta el punto estrictamente necesario de interés para evitar caer en gastos de tiempo, económicos y de materiales en exceso [1].

La *Figura 1.* Muestra el proceso que se lleva a cabo cuando se hace una simulación de cualquier sistema [2]. El primer paso es elegir qué se desea simular; a partir del análisis de comportamiento del sistema se obtiene un modelo matemático teniendo en cuenta los parámetros del entorno que lo afectan, en el paso siguiente se procede a obtener una predicción teórica del comportamiento del sistema en estudio el cual se puede hacer preferiblemente con la ayuda de un código por computadora; esto, para evitar errores de cálculo al obtener los resultados, dependiendo de la complejidad del sistema que se está simulando.

Una vez obtenido un resultado teórico se sigue a un paso con gran importancia; este paso es la validación y verificación de la simulación. La validación hace referencia al proceso de comparar los resultados obtenidos teóricamente con datos obtenidos de manera experimental; la cercanía de los resultados teóricos a los experimentales dependen de si se tuvieron en cuenta los parámetros necesarios para construir el modelo matemático adecuado; es en esta parte, donde entra a desempeñar un importante roll el proceso de la verificación; el cual evalúa si las ecuaciones del modelo obtenido realmente describen al sistema, o si se han omitido algunos parámetros que afectan el comportamientos del mismo. Una vez obtenidos los resultados adecuados se establece que la simulación desarrollada para el sistema tiene validez o no y qué tan cercanos son sus resultados a los del sistema en el mundo real.

Figura 1.1: Proceso de Simulación de un universo físico de interés.

1.2 Inmersividad

El término inmersividad hace referencia al nivel de realismo que puede tener un universo virtual; lo que implica que se le hace creer a un usuario que se está dentro de ese universo simulado, esto se logra al estimular la mayor cantidad de sentidos posibles.

En el caso de los sistemas dedicados a las simulaciones de vuelos dinámicos se deben estimular sentidos que estén relacionados con la percepción del movimiento y sus efectos. Dichos sentidos son: *Sentido Vestibular* que es el encargado de percibir el movimiento y mantener el equilibrio, *Sentido de Kinestesia* que se encarga de informarle al sistema nervioso cómo se está moviendo el cuerpo; finalmente, se encuentra el *Sentido de Cenestesia* quien percibe sensaciones al producirse un tipo de movimiento; estas sensaciones pueden ser mareo, sueño, cansancio, entre otras.

El grado de inmersividad de un sistema simulado depende de cuantos sentidos involucrados directamente en el comportamiento el sistema es capaz de estimular; pero, se debe tener en cuenta que la estimulación de los sentidos debe ser la adecuada para no producir efectos no deseados e innecesarios en el usuario del simulador, reafirmando con esto que siempre se debe simular estrictamente hasta el nivel requerido y no excederse para evitar resultados que no interesan y que posiblemente entorpezcan los resultados esperados.

1.3 Sistemas de simulación de vuelo

Un Simulador de vuelo es un sistema que trata de replicar, la experiencia de volar un aeronave de forma precisa, realista y segura. Estos sistemas de simulación han sido utilizados a lo largo de la historia de la aviación y han estado en constante evolución, tanto como la tecnología lo ha permitido. En los primeros días de la aviación se construyeron simuladores de vuelos tan básicos o rudimentarios como una plataforma montada sobre dos barriles para simular la elevación y el giro de un biplano; actualmente, un simulador de vuelo es capaz de reproducir cualquier

escenario de un determinado tipo de aeronave y de todos sus sistemas de vuelo. Los simuladores de vuelo actuales reproducen fielmente los mandos y controles de las aeronaves, las situaciones por las que puede pasar un aparato en vuelo, las maniobras de despegue y aterrizaje en diversos aeropuertos y multitud de condiciones meteorológicas.

1.3.1 Revisión histórica.

Después del exitoso vuelo de 12 segundos del biplano de los Hermanos Wright en 1903, se ve la necesidad de entrenar a los pilotos en cómo maniobrar una aeronave durante el despegue, en vuelo y en el aterrizaje; para ello se desarrollaron sistemas que permitieran simular los movimientos de la aeronave en vuelo.

El primer simulador de vuelo que se construyó fue el "Entrenador de Vuelo Antoinette" desarrollado en Francia en 1909, el cual consistía en dos medios barriles uno montado sobre el otro para simular la elevación y el giro de un biplano; los movimientos del simulador eran logrados de forma manual.

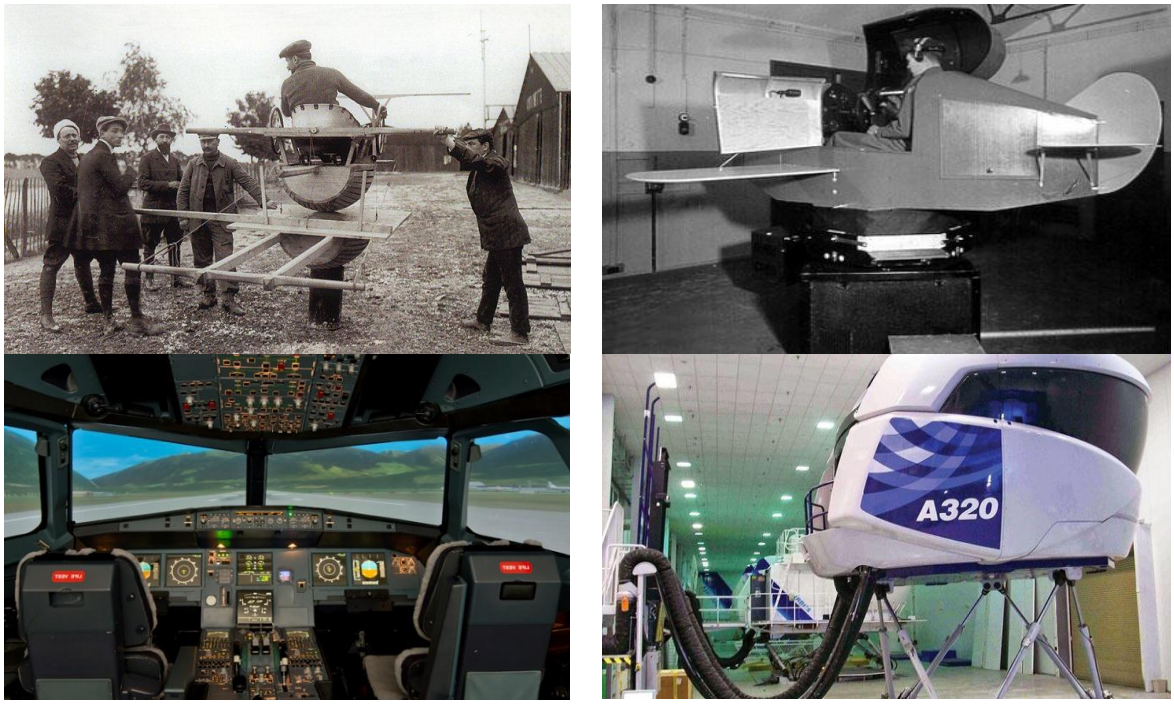


*Figura 1.2***:** *Evolución de los Simuladores de vuelo. Izquierda arriba: Entrenador de vuelo Antoinette 1909. Derecha arriba. Simulador de vuelo Link Trainer 1929. Izquierda abajo. Cabina de Simulador de vuelo de Airbus A320. Derecha Abajo. Simulador de vuelo de Airbus A320 vista externa. La actualidad.*

Durante la primera guerra mundial se desarrollaron varios simuladores de vuelo para entrenar a los pilotos; los que se destacaron fueron los construidos por

Lander y Heidelberg en Francia en 1917, el Orientador Ruggles de los Estados Unidos de Norteamérica también en 1917.

En 1929 Ed Link construye el simulador de vuelo Link Trainer el cual simulaba movimientos mecánicos e instrumentos de control; durante la segunda guerra mundial fue utilizado por varios países para entrenar a sus pilotos. [3,4].

En la actualidad, se han desarrollado Sistemas de simulación de vuelo tan complejos como lo permite la tecnología existente, estos sistemas constan de réplicas exactas de la cabina del aeronave a simular, un robot capaz de replicar los movimientos producidos por el aeronave en vuelo, un software que contiene el modelo matemático del aeronave y además es posible "volar" bajo las condiciones climáticas y mecánicas requeridas para entrenar a los pilotos de forma tal que cuando en vuelo se presente una situación de riesgo, éste esté preparado para solucionar la emergencia, salvando así su vida y la de las demás personas que vuelen con él.

1.3.2 Tipos de simuladores de vuelo

Existen básicamente dos tipos de simuladores de vuelo; los simuladores estáticos y los dinámicos. Los simuladores dinámicos, a diferencia de los estáticos, son capaces de estimular los sentidos relacionados con la percepción del movimiento debido a que cuentan con un mecanismo capaz de reproducir el movimiento de la aeronave que se está simulando, aumentando el grado de realismo percibido por el usuario. Para lograr un simulador de vuelo dinámico, hace falta conectar de manera adecuada los elementos mostrados en la ***figura 1.3***.

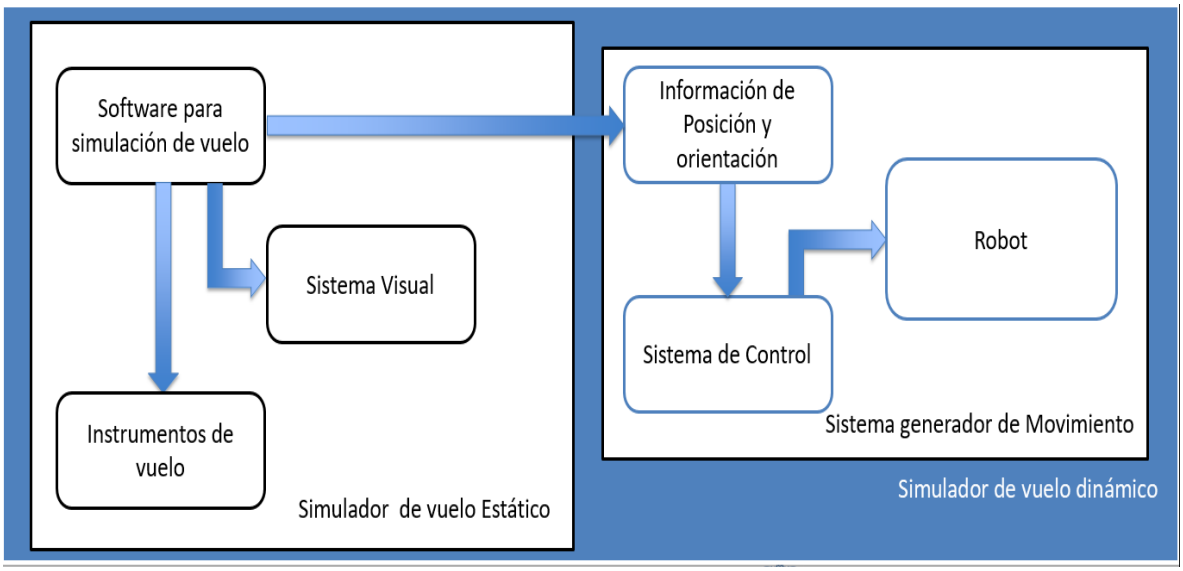


Figura 1.3: Elementos que conforman un simulador de vuelo dinámico

1.3.3 Clasificación de simuladores de vuelo

Los simuladores son evaluados por instituciones gubernamentales tales como la Administración Federal de Aviación de estados unidos (FAA) y Direcciones de Aeronáutica Civil de diferentes países, las cuales clasifican, regulan y certifican estos dispositivos según su categoría en niveles A, B, C y D. La principal exigencia para la certificación de estos equipos consiste en demostrar que sus características de vuelo coinciden exactamente con las de la aeronave para la cual fue fabricado el simulador. Esta clase de requerimientos de prueba para los simuladores están detallados en guías denominadas ATG (Guías de Pruebas de Aprobación) o QTG (Guías de Pruebas de Calificación), que no son otra cosa que documentación donde se especifica cada una de las características técnicas del simulador y cómo se prueba y comprueba su correcto funcionamiento. [5]

A continuación, se muestra una tabla con algunos de los requerimientos para Simuladores de vuelo según su categoría (A, B, C o D).

No	**Requisitos Generales**	**Niveles de Simulador**				Notas
		A	**B**	**C**	**D**	
1	**Configuración General de la Cabina**					
1.a	El simulador debe tener una cabina de vuelo que sea réplica del avión simulado, con controles, equipos, indicadores visuales, circuit breakers y mamparos colocados apropiadamente, funcionando correctamente y semejando al avión. La dirección del movimiento de los controles e interruptores debe ser idéntica a la del avión. Los asientos de los pilotos deben permitir al ocupante alcanzar el diseño "campo visual", establecido para el avión que está siendo simulado	**X**	**X**	**X**	**X**	Para propósitos del simulador, la cabina de vuelo consta de todo el espacio delante de una sección transversal del fuselaje en el punto posterior más extremo establecido en los asientos de los pilotos, incluidas las adicionales, estaciones requeridas para los miembros de la tripulación y aquellos mamparos requeridos detrás de los asientos de los pilotos
2	**Programación**					
2.a	Un modelo aerodinámico que contenga las diversas combinaciones de fuerzas de resistencia al avance y empuje normalmente encontrados en condiciones de vuelo, incluyendo el efecto por cambio en la actitud del avión, empuje, resistencia al avance, altitud, temperatura, peso total, momentos de inercia, posición del centro de gravedad y configuración.	**X**	**X**	**X**	**X**	
3	**Operación de Equipos**					
3.c	Los sistemas del simulador deben operar como los sistemas del avión en condiciones normales, anormales y en operaciones de emergencia en tierra y en vuelo.	**X**	**X**	**X**	**X**	

4	**Facilidades del Instructor o Evaluador**					
4.a	El simulador debe contar con al menos dos sillas adecuadas para el instructor/chequeador de rutas y el inspector de la UAEAC en adición de las de los tripulantes. Estas sillas deben proporcionar un campo visual adecuado del panel del piloto y de las ventanas delanteras. Todas las sillas diferentes a las de los tripulantes de cabina no necesariamente deben representar las del avión, pero deben estar debidamente aseguradas al piso y equipadas con cinturones de seguridad o arneses similares a los de la tripulación.	**X**	**X**	**X**	**X**	La Secretaría de Seguridad Aérea tomará en consideración alternativas a esta norma para sillas adicionales para cabinas con configuraciones de tipo único.
5	**Sistema de Movimiento**					
5.a	El simulador debe tener un sistema de movimiento (fuerza), señales perceptibles al piloto que sean representativas del movimiento del avión.	**X**	**X**	**X**	**X**	
5.b	El simulador debe tener un sistema de movimiento (señales de fuerza) con un mínimo de tres grados de libre rotación (al menos cabeceo, alabeo y ascenso).	**X**	**X**			
5.c	El simulador debe tener un sistema de movimiento (señales de fuerza) que produzca señales de al menos el equivalente a los seis grados de libre rotación (ej. cabeceo, alabeo, guiñada, ascenso, balanceo y ondulación).			**X**	**X**	
5.f	El simulador debe proporcionar las vibraciones que son características del movimiento que sean resultado de la operación del avión sí la vibración marca un evento o un estado del avión que pueda ser detectado en la cabina de vuelo.				**X**	El simulador debe estar programado e instrumentado de tal manera que los módulos de sacudidas características puedan ser medidos y comparados con los datos del avión.
6	**Sistema Visual**					
6.a	El simulador debe tener un sistema visual fuera de la cabina de los pilotos.	**X**	**X**	**X**	**X**	
7	**Sistema de Sonido**					
7.d	El simulador debe proporcionar amplitud y frecuencia realista de los sonidos y ruidos de la cabina de vuelo.				**X**	

Tabla 1.1 Requisitos mínimos de un simulador de vuelo según su clasificación. Tomado de "Dispositivos Simuladores para Entrenamiento de Vuelo" Reglamentos Aeronáuticos de Colombia, RAC 24. Aeronáutica Civil Colombiana [6].

2. ROBOTS MANIPULADORES

Este capítulo se enfoca en los conceptos básicos de la robótica para poder entender a lo que se refiere un manipulador paralelo, en específico el manipulador paralelo conocido como “Plataforma Stewart” del cual se hace una descripción general de sus características y el porqué de su uso en el mundo de la simulación de vuelo; finalmente, se explica de forma breve cómo modelar matemáticamente un manipulador de este tipo.

2.1 Definiciones

2.1.1 Máquina. Sistema concebido para realizar una tarea determinada que comporta la presencia de fuerzas, movimientos y en principio, la realización de trabajo.

2.1.2 Mecanismo. Conjunto de elementos mecánicos que hacen una función determinada en una máquina. El conjunto de la funciones de los mecanismos de una máquina ha de ser el necesario para que ésta realice la tarea recomendada.

2.1.3 Grupo o unidad. Conjunto diferenciado de elementos de una máquina. Así, el conjunto de elementos implicados en la tracción de un automóvil es el grupo tractor.

2.1.4 Elemento. Toda entidad constitutiva de una máquina o mecanismo que se considera una unidad. Son ejemplos de elementos un pistón, una biela, un rodamiento, una rótula, etc.

2.1.5 Miembro. Elemento material de una máquina o mecanismo que puede ser sólido rígido, sólido flexible o fluido. En la contabilización de los miembros de un mecanismo se debe tener en cuenta, si existe, el miembro fijo a la referencia de estudio, que recibe diferentes nombres según el contexto: Base, soporte, bancada, bastidor, etc.

2.1.6 Cadena cinemática. Conjunto o subconjunto de miembros de un mecanismo enlazados entre sí. Por ejemplo, la cadena de transmisión de un vehículo, el mecanismo pistón-biela-manivela, etc. Los miembros de una cadena cinemática se denominan *eslabones.*

- *Cadena cerrada o anillo.* Cadena cinemática tal que cada uno de sus miembros está enlazado nada más con dos miembros de la misma cadena.
- *Cadena abierta.* Cadena cinemática que no tiene ningún anillo.

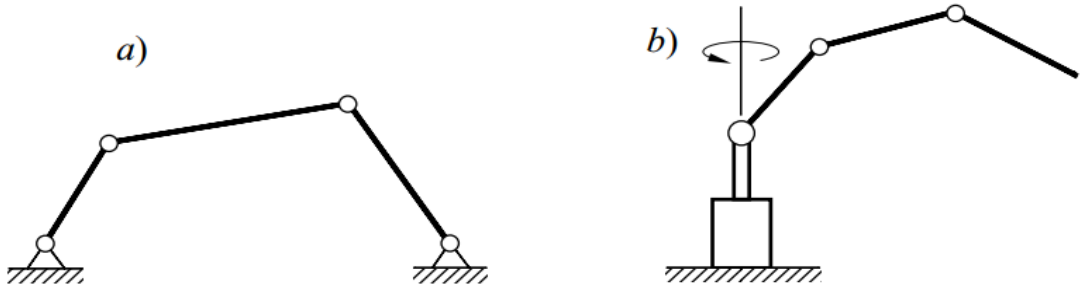


Figura 2.1: Cadenas cinemáticas: a) Cerrada, b) abierta.

2.1.7 Inversión de una cadena cinemática. Transformación de un mecanismo en otro por medio de la elección de diferentes miembros de la cadena como elemento fijo a la referencia. En todos los mecanismos obtenidos por inversión de una misma cadena cinemática los movimientos relativos son evidentemente los mismos, hecho que facilita el estudio.

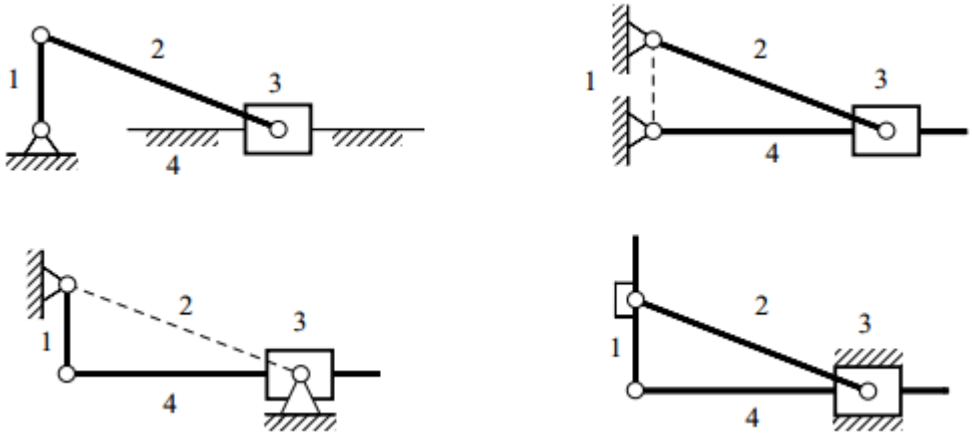


Figura 2.2. Las cuatro inversiones del mecanismo pistón-biela-manivela

2.1.8 Restricción o enlace. Una condición impuesta a la configuración se conoce como *condición de enlace geométrica* y una condición al movimiento del mecanismo se conoce como *condición de enlace cinemática.* En estas condiciones puede aparecer el tiempo explícitamente o no.

2.1.9 Par cinemático. Enlace entre dos miembros de un mecanismo causado por el contacto directo entre ellos y que puede ser puntual, según una recta o una superficie. En la materialización del enlace pueden participar sólidos auxiliares de enlace (SAE).

2.1.10 Junta. Ligadura entre dos miembros de un mecanismo que se realiza mediante elementos intermedios, como puede ser una junta elástica, universal, etc.

(Cardona, S. Clos, D. (2001) Pg 13-15)

Existen varios tipos de articulaciones que sirven para darle movilidad a los mecanismos [8]. La **articulación de rotación** suministra un grado de libertad consistente en una rotación alrededor del eje de articulación. Esta articulación es la más utilizada. En la **articulación prismática** el grado de libertad consiste en una traslación a lo largo del eje de la articulación.
En la **articulación cilíndrica** existen dos grados de libertad: una rotación y una traslación.
La **articulación planar** está caracterizada por el movimiento de desplazamiento en un plano, existiendo, por tanto dos grados de libertad.
Por último la **articulación esférica** combina tres giros en tres direcciones perpendiculares en el espacio.

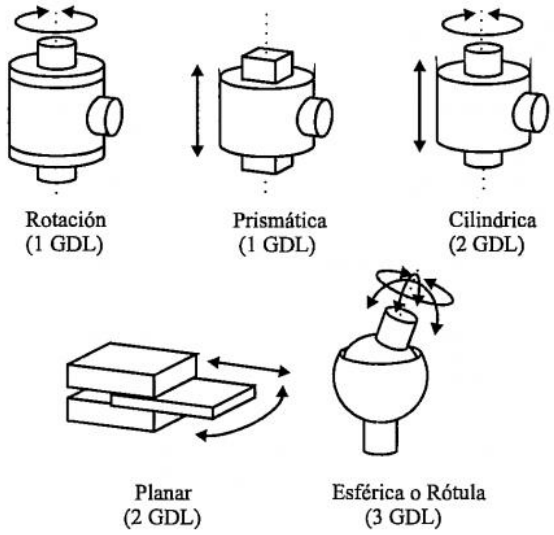


Figura 2.3. Tipos de articulaciones más utilizadas,

2.1.11 Grados de libertad de un mecanismo. Los grados de libertad son el número de parámetros independientes que fijan la situación del órgano terminal. El número de grados de libertad suele coincidir con el número de eslabones de la cadena cinemática. Así en la *figura 2.4*.a, se ilustra una estructura con dos eslabones, dos articulaciones prismáticas y dos grados de libertad.

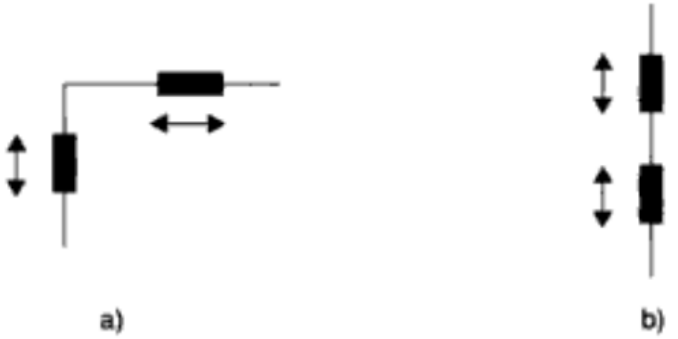


Figura 2.4 Pérdida de grados de libertad en estructura con dos eslabones: a) dos grados de libertad y b) un grado de libertad.

Sin embargo, pueden existir casos degenerados, tal como se ilustra en la *figura 2.4.b* en la cual se aprecia que, aunque existan dos eslabones y dos articulaciones prismáticas, tan solo se tiene un grado de libertad. Por consiguiente, en general, el número de grados de libertad es menor o igual que el número de eslabones de la cadena cinemática. Ollero, A. (2001).

2.1.12. Robot. Los robots son máquinas en las que se integran componentes mecánicos, eléctricos, electrónicos y de comunicaciones, dotados de un sistema informático para su control en tiempo real, percepción del entorno y programación. Ollero, A. (2001).

2.2 Robots paralelos.

2.2.1 Manipuladores paralelos generalizados. Es un mecanismo de una cadena cinemática cerrada cuyo efector final está unido a la base por varias cadenas cinemáticas independientes. Merlet. J. P. (2006).

2.2.2 Manipuladores paralelos. La definición generalizada de manipuladores paralelos incluye mecanismos redundantes (con más actuadores que número de grados de libertad controlados del efector final). Así como también manipuladores que trabajan en cooperación.

Merlet da las principales características de este tipo de manipuladores, con los que normalmente se trabaja

- Al menos dos cadenas cinemáticas sostienen al efector final, cada una de éstas contiene al menos un actuador simple.
- El número de actuadores es el mismo del número de grados de libertad del efector final.
- La movilidad del efector final es cero cuando los actuadores están bloqueados.

También se menciona las bondades de trabajar con estos mecanismos

- Un mínimo de dos cadenas cinemáticas permite disminuir la carga sobre las cadenas.
- El número de actuadores es mínimo
- El número de sensores necesarios para el control del mecanismo de lazo cerrado es mínimo
- Cuando los actuadores están bloqueados, el manipulador permanece en su posición. Este es un aspecto de seguridad importante para ciertas aplicaciones, como la robótica médica.

2.3 Plataforma Stewart

Es un manipulador paralelo consistente de dos cuerpos rígidos: una plataforma móvil, o simplemente una plataforma y una base. La posición y orientación de la base son fijas. La plataforma y la base están conectadas por seis piernas extensibles [9]. Éste mecanismo cuenta con 6 grados de libertad (DOF); esto es delante/atrás, arriba/abajo, izquierda/derecha (traslación en tres ejes perpendiculares), combinados con la rotación sobre tres ejes perpendiculares (Guiñada, Cabeceo, Alabeo) lo que lo hace idóneo para generar movimientos como los que se producen en un aeronave en vuelo; de hecho en sus inicios, ésta fue creada para tal fin. [10].

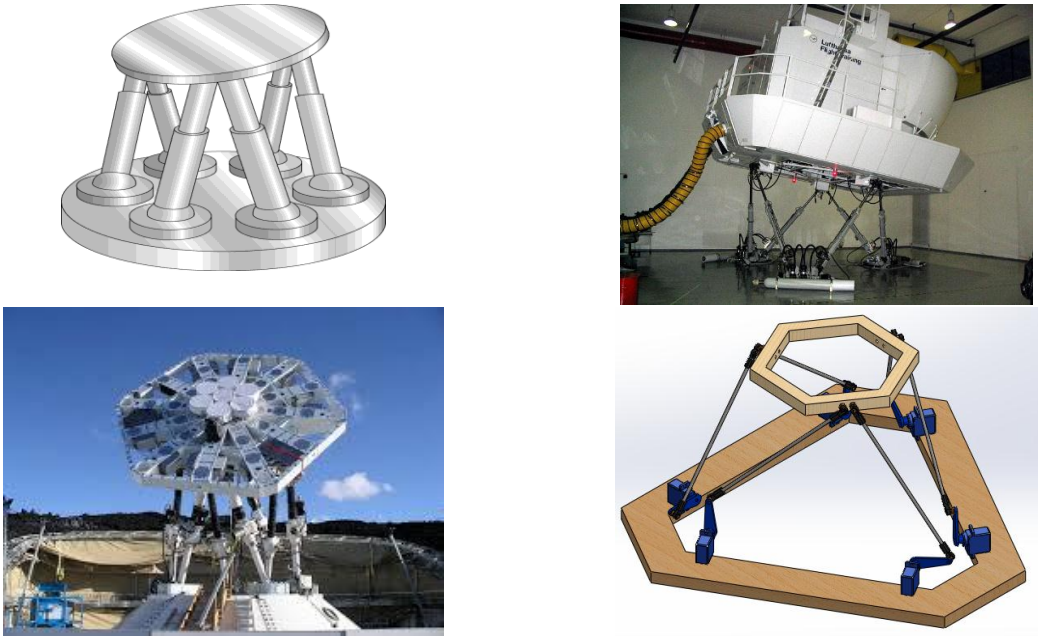


Figura 2.5 Plataforma Stewart. Arriba. Con Actuadores lineales. Abajo: Actuadores Rotacionales.

Al permitir movimientos en los grados de libertad necesarios para describir la posición y orientación de un cuerpo en el espacio se convierte en un mecanismo de suma importancia en los sistemas en los cuales se requiere simular y/o controlar movimientos. Este tipo de manipulador tiene aplicaciones en diferentes campos, Tanto en la industria, en la medicina, en la astronomía y en la aviación entre otros.

2.4 Diseño de Plataforma Stewart

Un robot paralelo tiene múltiples parámetros de diseño, los cuales en principio deben ser considerados y controlados, pero; ocurre que en el caso de una plataforma tipo Stewart, el número de parámetros que en teoría se deben conocer son muchos; por eso es necesario empezar haciendo ciertas suposiciones para reducir el cálculo de cada parámetro; por ejemplo suponemos que todas los links

del robot tienen la misma longitud; lo cual es cierto cuando el robot está en una posición inicial, sin rotar en ninguno de los ejes y sin desplazarse a lo largo de ellos.

En la ***Figura 2.6*** Se muestran las arquitecturas más utilizadas de plataformas Stewart con diferentes tipos de juntas; La figura E (Plataforma Stewart con actuadores rotacionales), es en la que nos enfocaremos, pues es la que se desarrollará en este trabajo; puesto que el tipo de actuadores que se han elegido son Servomotores.
El tipo de robot que se decide diseñar es un Robot 6-6 RSS (Robot de 6 grados de libertad con estructura de cadena cinemática Rotacional Esférica Esférica)

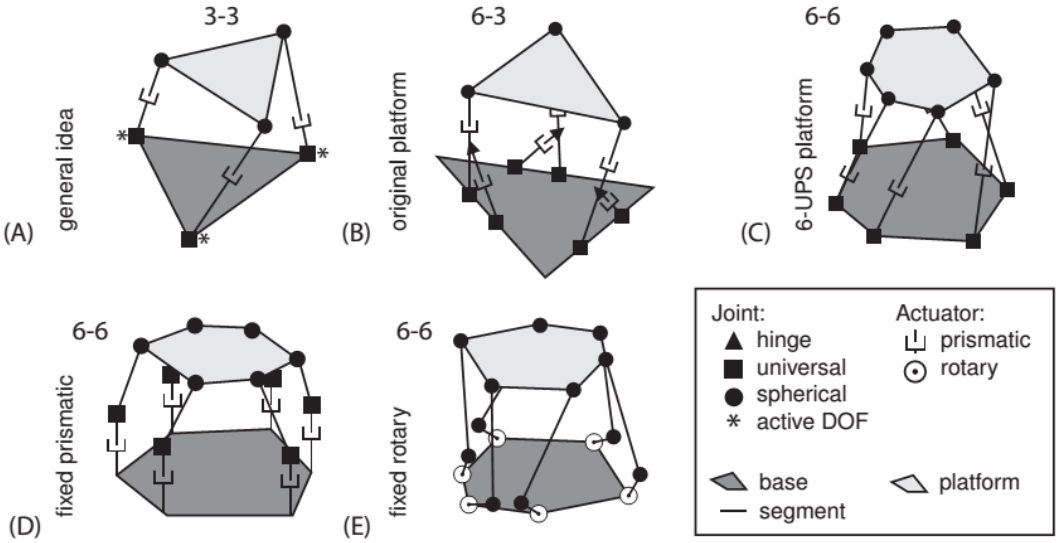


Figura 2.6 ***Representaciones simplificadas de varias arquitecturas de la GSP. (A) La idea original de Stewart y (B) su realización real con 6 juntas prismáticas (Hidráulicos). (C) la realización más típica de una plataforma de 6 DOF comúnmente conocida como "la plataforma de Stewart" o un robot hexápodo. (D) y (E) muestran ejemplos de plataformas Stewart con actuadores prismáticos y rotacionales respectivamente, fijados en sus bases.[11].***

Éste manipulador está formado por una base definida por un hexágono irregular y una plataforma similar que se unen a sus vértices por medio de 6 brazos que a su vez están formados por dos eslabones uno de entrada y otro de acoplamiento. El eslabón de entrada es el que está unido a la base y el de acoplamiento a la plataforma. La unión entre la base y la plataforma se hace por medio de una articulación rotacional entre la base y el eslabón de entrada, una articulación esférica entre el eslabón de entrada y el eslabón de salida y una articulación esférica entre el eslabón de salida y la plataforma; conformando así la configuración RSS mencionada anteriormente; debido a que se tienen 6 brazos se denomina 6-RSS.

Los parámetros de diseño que se toman para éste trabajo son la longitud de los vértices de los hexágonos de la base y la plataforma, la longitud del eslabón de entrada y del eslabón de salida; suponiendo además, que éstos dos últimos son iguales para cada brazo.

Los hexágonos tanto de la base como de la plataforma se definfen completamente mediante dos magnitudes una para tres aristas mayores y otra para las tres aristas menores restantes.

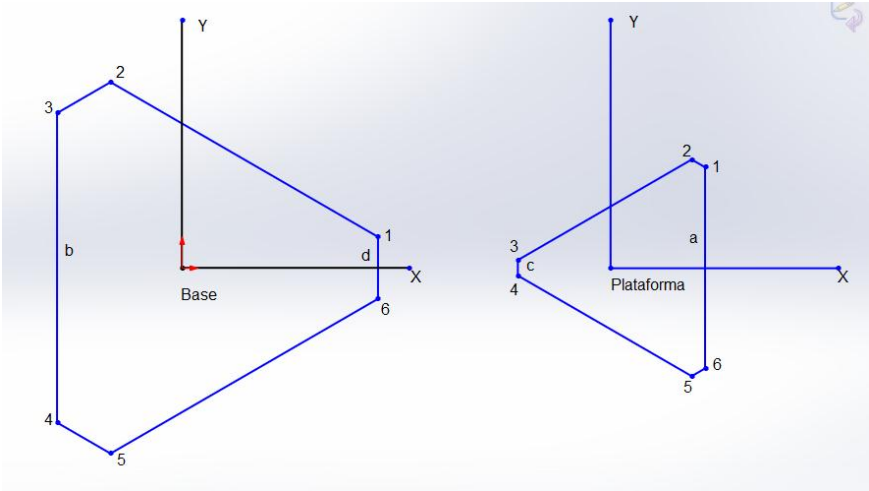


Figura 2.7 Base y plataforma con sus respectivos vértices

Las posiciones de cada uno de los vértices se encuentra usando geometría básica, teniendo en cuenta que cada hexágono está centrado en su sistema de coordenadas; Se llama a partir de ahora ***Bi*** a la coordenada del vértice de la base y ***ti*** a la coordenada de la plataforma. En ***la Tabla 1*** se muestran los valores de cada coordenada en términos de las magnitudes b y d para la base y de a y c para la plataforma.

Coordenadas vértices de la base	***Coordenadas vértices de la plataforma***
$B_1 = \left[\frac{\sqrt{3}}{6}(2b+d) \quad \frac{1}{2}d \quad 0\right]^T$	$t_1 = \left[\frac{\sqrt{3}}{6}(a+2c) \quad \frac{1}{2}c \quad 0\right]^T$
$B_2 = \left[-\frac{\sqrt{3}}{6}(b-d) \quad \frac{1}{2}(b+d) \quad 0\right]^T$	$t_2 = \left[\frac{\sqrt{3}}{6}(a-c) \quad \frac{1}{2}(a+c) \quad 0\right]^T$
$B_3 = \left[-\frac{\sqrt{3}}{6}(b+2d) \quad \frac{1}{2}b \quad 0\right]^T$	$t_3 = \left[-\frac{\sqrt{3}}{6}(2a+c) \quad \frac{1}{2}c \quad 0\right]^T$
$B_4 = \left[-\frac{\sqrt{3}}{6}(b+2d) \quad -\frac{1}{2}b \quad 0\right]^T$	$t_4 = \left[-\frac{\sqrt{3}}{6}(2a+c) \quad -\frac{1}{2}c \quad 0\right]^T$
$B_5 = \left[-\frac{\sqrt{3}}{6}(b-d) \quad -\frac{1}{2}(b+d) \quad 0\right]^T$	$t_5 = \left[\frac{\sqrt{3}}{6}(a-c) \quad -\frac{1}{2}(a+c) \quad 0\right]^T$

$B_6 = \left[\frac{\sqrt{3}}{6}(2b+d) \quad -\frac{1}{2}d \quad 0\right]^T$	$t_6 = \left[\frac{\sqrt{3}}{6}(a+2c) \quad -\frac{1}{2}c \quad 0\right]^T$

Tabla 2.1: Coordenadas de los vértices de la base y la plataforma del manipulador paralelo tipo Stewart.

2.4.1 Orientación de la plataforma

Para describir la orientación de la plataforma en el espacio se recurre al uso de tres ángulos. (α, β, γ). El ángulo ***α*** representa el ángulo que la plataforma ha rotado sobre el eje *X* fijo mientras que el ángulo ***β*** lo hace pero sobre el eje *Y* fijo. El ángulo ***γ*** representa el ángulo que la plataforma, ya orientada con ***α*** y ***β***, gira en torno a su vector normal, tal como se muestra en la ***figura 2.8.***

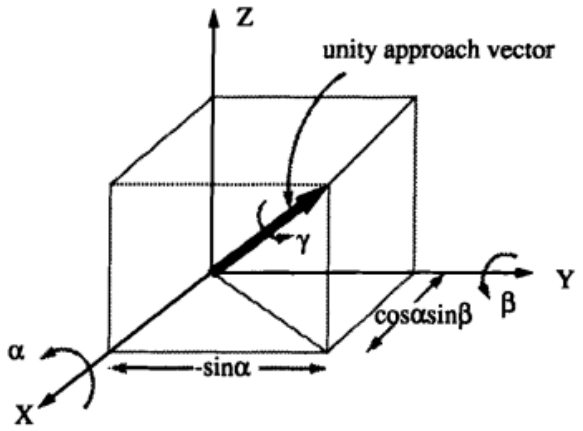


Figura 2.8 Representación de los ángulos de rotación de la plataforma

La forma que se utiliza para obtener la orientación de la plataforma es por medio de matrices de rotación consecutivas de la siguiente manera: Primero, se rota el sistema de coordenadas asociado a la plataforma sobre el ee X fijo y luego sobre el eje Y tambien fijo; finalmente, y considerando que el eje z está orientado según el vector unitario de las rotaciones anteriores, se hace una última rotación respecto al eje z. Este proceso se describe matematicamente como:

$$R_{\alpha,\beta,\gamma} = \left(R_{Y,\beta}.R_{X,\alpha}\right).R_{Z,\gamma} \quad [1]$$

$$R_{\alpha,\beta,\gamma} = \begin{bmatrix} cos\beta & 0 & sin\beta \\ 0 & 1 & 0 \\ -sin\beta & 0 & cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & -sin\alpha \\ 0 & sin\alpha & cos\alpha \end{bmatrix} \begin{bmatrix} cos\gamma & -sin\gamma & 0 \\ sin\gamma & cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad [2]$$

Se debe tener en cuenta que éste producto de multiplicación no es conmutativo; por lo Tanto debe realizarce estrictamente en el orden mencionado anteriormente.

Finalmente, se obtiene la matriz de rotación que representa la orientación de la plataforma.

$$R_{\alpha,\beta,\gamma} = \begin{bmatrix} cos\beta cos\gamma + sin\alpha sin\beta sin\gamma & -cos\beta sin\gamma + sin\alpha sin\beta cos\gamma & cos\alpha cos\beta \\ cos\alpha sin\gamma & cos\alpha cos\gamma & -sin\alpha \\ -sin\beta cos\gamma + sin\alpha cos\beta sin\gamma & sin\beta sin\gamma + sin\alpha cos\beta cos\gamma & cos\alpha cos\beta \end{bmatrix} \quad (3)$$

2.4.2 Localización de la plataforma

Para describir la localización de la plataforma, se utiliza una herramienta matemática conocida como matriz de transformación la cual está compuesta por una matriz de rotación R , un vector de posicion P , una matriz de perspectiva f y un factor de escalamiento w; en robótica f es nula y w tiene un valor unitario [12]

$$T = \begin{bmatrix} R_{3x3} & P_{3x1} \\ f_{1x3} & w \end{bmatrix} \quad (4)$$

Donde R es la matriz de rotación obtenida en (3) y P es el vector de posición de la plataforma descrito en la ***tabla 2.1*** conocido como $\boldsymbol{t_i}$**.**

$${}^{U}_{Top}T_r = \begin{bmatrix} cos\beta cos\gamma + sin\alpha sin\beta sin\gamma & -cos\beta sin\gamma + sin\alpha sin\beta cos\gamma & cos\alpha cos\beta & P_x \\ cos\alpha sin\gamma & cos\alpha cos\gamma & -sin\alpha & P_y \\ -sin\beta cos\gamma + sin\alpha cos\beta sin\gamma & sin\beta sin\gamma + sin\alpha cos\beta cos\gamma & cos\alpha cos\beta & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Con esta transformación se encuentran las coordenadas de los vértices de la plataforma $\boldsymbol{t_i}$ respecto al sistema de coordenadas de referencia $\boldsymbol{T_i}$

$$T_i = {}^{U}t_i = {}_{Top}{}^{U}T_r({}^{Top}t_i) \quad (5)$$

En el Apéndice 1 se muestra el código en MATLAB para encontrar cada una de las coordenadas mencionadas anteriormente.

2.5 Modelo matemático de plataforma Stewart

Es necesario conocer la posición del elemento terminal del robot para poder tener control sobre él. Éste análisis se puede hacer desde dos enfoques, problema cinemática inversa y cinemática directa.

- El enfoque de la cinemática inversa es determinar los valores de sus variables articulares activas, dadas una posición y una orientación determinadas.
- En el caso de la cinemática directa, se busca determinar la posición y orientación del elemento terminal del robot, con respecto a un sistema de

coordenadas de referencia, cuando se conocen las variables articulares activas y la geometría del robot.[13]

2.5.1 Estrategias de resolución

Los enfoques de cinemática inversa se pueden resolver mediante varias estrategias; las más usadas la representación Denavit-Hartenber (D-H) y la solución geométrica.

El enfoque (D-H) es una serie pasos para la colocación de sistemas de coordenadas en cada eslabón del robot para que la matriz de transformación homogénea entre ellos dependa de cuatro parámetros ($a_i, \alpha_i, d_i, \theta_i$) o sea de cuatro transformaciones básicas.

Los cuatro parámetros se definen de la siguiente manera:

- El parámetro a_i corresponde a la distancia de traslación a lo largo del eje X_i desde el eje Z_{i-1} hasta el origen del sistema $i-\acute{e}simo$. Representa la longitud del eslabón.
- El parámetro $\alpha_i$corresponde al ángulo de rotación sobre el eje X_i; es decir, el ángulo existente entre los ejes Z_{i-1} y Z_i. Representa la torsión del eslabón.
- d_i corresponde a la distancia de traslación a lo largo del eje Z_{i-1} desde el origen del sistema de coordenadas $(i-1)-\acute{e}simo$ hasta el eje X_i. Se trata de la variable articular para la articulación i si ésta es prismática.
- θ_i es el ángulo de rotación sobre el eje Z_{i-1}; es decir, el ángulo existente entre los ejes X_{i-1} y X_i. Es la variable articular para la articulación i si esta es rotacional.[14]

El enfoque geométrico se basa en la descomposición de la geometría espacial del robot en varios problemas geométricos planos, para lo que se usan herramientas geométricas y trigonométricas con el fin de encontrar el valor de las variables articulares o la posición y orientación del elemento terminal.

En lo que respecta a los robots seriales, el problema cinemático directo comúnmente se resuelve de manera sistemática mediante la representación D-H; sin embargo, dada la existencia de múltiples soluciones al resolver el problema cinemático inverso, un enfoque geométrico es más sencillo de aplicar en dicho caso.

Para los robots paralelos la existencia de varios lazos cerrados dificultaría la aplicación de la representación D-H. Además, la existencia de articulaciones con más de un DOF (las articulaciones esféricas como caso típico) imposibilita usar esta representación tal y como está definida, pues más de cuatro parámetros por eslabón son necesarios para definir completamente una determinada configuración del robot. Es por ello que para este tipo de robots es más conveniente el enfoque geométrico, definiéndose ecuaciones vectoriales para

cada brazo en donde las variables articulares pasivas puedan ser eliminadas del análisis [15].

2.5.2 Solución del problema de cinemática inversa

Como lo que se requiere es encontrar la variable articular de cada uno de los brazos del robot, entonces se empieza por descomponer el sistema en mecanismos en 2D (planares) tal y como se muestra en la ***figura 2.9*** Siendo P_i la longitud del brazo del actuador, L_i la longitud del eslabón de salida, θ_i la variable articular que se desea conocer, T_i el punto de unión entre el eslabón L_i y un vértice de la plataforma, O_i el origen ubicado en cada vértice, W_i la distancia entre el punto, T_i y O_i, $\varphi_i$el ángulo formado entre P_i y L_i ; finalmente en la figura se define a $Q_i$como el punto de unión entre los eslabones P_i y L_i.

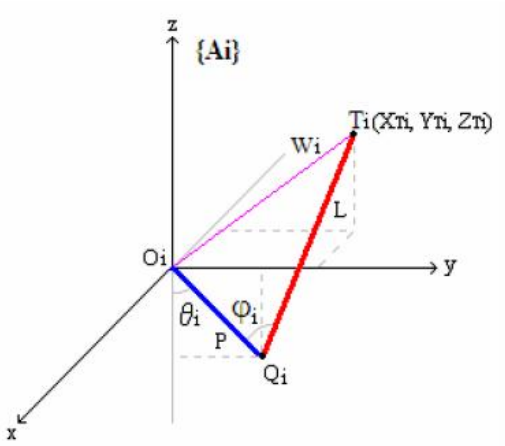


Figura 2.9 Descripción geométrica de un brazo de la plataforma con sus respectivos sistemas de coordenadas

Al unir las líneas formadas por P_i y L_i y W_i se forma un triángulo OQT no rectángulo lo que permite como se mencionó anteriormente tratar el problema en 2D. Usando la ley de los cosenos y sabiendo que P_i y L_i son fijos (en adelante se nombrarán como $P\ y\ L$) y conocidos se determina W_i con la expresión:

$$W_i = \sqrt{P^2 + L^2 - 2PL\,cos\varphi_i} \quad (6)$$

Donde φ_i se obtiene del producto punto entre los vectores $\boldsymbol{P_i}$ y $\boldsymbol{L_i}$ los cuales se definen partiendo desde el punto Q_i que define también al vector $\boldsymbol{Q_i}$ que parte desde el origen O_i

$$\boldsymbol{Q_i} = (Psin\theta_i)\hat{\boldsymbol{j}} - (Pcos\theta_i)\hat{\boldsymbol{k}}, \qquad \boldsymbol{P_i} = \boldsymbol{O_i} - \boldsymbol{Q_i}, \qquad \boldsymbol{L_i} = \boldsymbol{W_i} - \boldsymbol{Q_i} \quad \textbf{(7)}$$

Obteniendo entonces φ_i como:

$$\varphi_i = cos^{-1}\left(\frac{\boldsymbol{P_i}.\boldsymbol{L_i}}{PL}\right) \quad (8)$$

De donde se obtiene

$$W_i = \sqrt{P^2 + L^2 - 2(\boldsymbol{P_i}.\boldsymbol{L_i})} \quad (9)$$

Lo siguiente que se hace es referir cada sistema de coordenadas $\{\boldsymbol{A_i}\}$ a cada uno de los vértices de la base, con la posición y orientación descrita en la ***figura 2.10*** con el fin de obtener la expresión respecto al sistema de coordenadas de referencia.

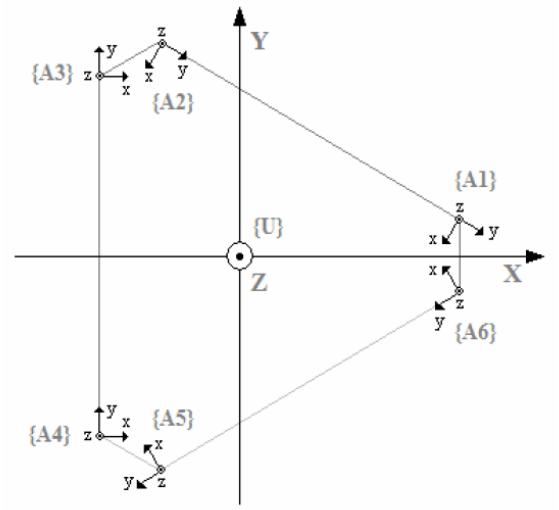


Figura 2.10 Sistemas de coordenadas de cada vértice de la base

Se usa una matriz de transformación homogénea para dicho fin, se hace una rotación respecto al eje z e un ángulo σ_i y una traslación hasta los puntos r_{xi} , r_{yi} la cual se muestra a continuación.

$$T_{Ai}\begin{bmatrix} cos\sigma_i & -sin\sigma_i & 0 & r_{xi} \\ sin\sigma_i & cos\sigma_i & 0 & r_{yi} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

De la figura 2.10 se pueden determinar los ángulos σ_i que están rotados cada sistema de coordenadas respecto al eje z; la *Tabla 2,2* resume los parámetros de la matriz de transformación T_{Ai}.[16]

i	1	2	3	4	5	6
σ_i	-120°	-120°	0	0	120°	120°
r_{xi}	$\frac{\sqrt{3}}{6}(2b+d)$	$-\frac{\sqrt{3}}{6}(b-d)$	$-\frac{\sqrt{3}}{6}(b+2d)$	$-\frac{\sqrt{3}}{6}(b+2d)$	$-\frac{\sqrt{3}}{6}(b-d)$	$\frac{\sqrt{3}}{6}(2b+d)$
r_{yi}	$\frac{1}{2}d$	$\frac{1}{2}(b+d)$	$\frac{1}{2}b$	$-\frac{1}{2}b$	$-\frac{1}{2}(b+d)$	$-\frac{1}{2}d$

Tabla 2.2: Parámetros de de T_{Ai} para $\{A_i\}$

Ahora; como ya se ha trasformado cada sistema de coordenadas con respecto al origen $\{\boldsymbol{O}\}$, necesario que los vectores $\boldsymbol{O_i}, \boldsymbol{Q_i}, \boldsymbol{P_i}\ y\ \boldsymbol{l_i}$ también se transformen quedando como:

$$\boldsymbol{O_i} = (r_{xi})\hat{\boldsymbol{\imath}} + (r_{yi})\hat{\boldsymbol{\jmath}}$$
$$\boldsymbol{Q_i} = (-P\theta_i sin\sigma_i + r_{xi})\hat{\boldsymbol{\imath}} + (-P\theta_i cos\sigma_i + r_{yi})\hat{\boldsymbol{\jmath}} + (-Pcos\theta_i)\widehat{\boldsymbol{k}}$$
$$\boldsymbol{P_i} = (P\theta_i sin\sigma_i)\hat{\boldsymbol{\imath}} - (P\theta_i cos\sigma_i)\hat{\boldsymbol{\jmath}} + (Pcos\theta_i)\widehat{\boldsymbol{k}}$$
$$\boldsymbol{L_i} = (X_{Ti} + P\theta_i sin\sigma_i - r_{xi})\hat{\boldsymbol{\imath}} + (Y_{Ti} - P\theta_i cos\sigma_i - r_{yi})\hat{\boldsymbol{\jmath}} + (Z_{Ti} + Pcos\theta_i)\widehat{\boldsymbol{k}}$$

Ahora; al encontrar el producto punto entre $\boldsymbol{P_i}$ y $\boldsymbol{L_i}$ y remplazando el resultado en la expresión (6) se obtiene

$$W_i = \sqrt{L^2 - P^2 - 2Psin\theta_i\left(sin\delta_i(X_{Ti} - r_{xi}) - cos\delta_i(Y_{Ti} - r_{yi})\right) - 2Pcos\theta_i(Z_{Ti})} \quad (10)$$

También es posible encontrar la magnitud de W_i haciendo uso del teorema de Pitágoras de la forma:

$$W_i = \sqrt{(X_{Ti} - r_{xi})^2 + (Y_{Ti} - r_{yi})^2 + (Z_{Ti})^2} \quad (11)$$

Es evidente que se pueden igualar las dos expresiones y teniendo en cuenta que $sin\delta_i\ y\ cos\delta_i$ son valores fijos y conocidos; por lo tanto se puede obtener una expresión en términos de la variable articular θ_i. Al solucionar la igualdad propuesta resulta que se pueden determinar las siguientes "constantes" para facilitar los cálculos posteriores:

$$A_i = 2P\left(sin\delta_i(X_{Ti} - r_{xi}) - cos\delta_i(Y_{Ti} - r_{yi})\right) \quad (12)$$
$$B_i = 2PZ_{Ti}$$
$$C_i = L^2 - P^2 - (X_{Ti} - r_{xi})^2 - (Y_{Ti} - r_{yi})^2 - (Z_{Ti})^2$$

Al escribir la ecuación completa se obtiene:

$$Asin\theta_i + Bcos\theta_i = C_i \quad (13)$$

Ahora; se hace uso de la identidad trigonométrica (14)

$$Asin\theta + Bcos\theta = \sqrt{(A^2 + B^2)}\cos\left(\theta - tan^{-1}\left(\frac{A}{B}\right)\right) \quad (14)$$

Al igualar los (13) con (14) se obtiene finalmente de manera clara y simple la expresión en términos de la variable articular θ_i que es lo que se está buscando

$$\theta_i = \pm cos^{-1}\left(\frac{C_i}{\sqrt{{A_i}^2 + {B_i}^2}}\right) + tan^{-1}\left(\frac{{A_i}^2}{{B_i}^2}\right)$$

Los valores de θ_i que son reales hacen parte de del espacio de trabajo del robot diseñado.

3. AERONAVE DE ENTRENAMIENTO PRIMARIO

Una aeronave de entrenamiento primario es usado para desarrollar las habilidades de pilotaje, de navegación o de combate de los tripulantes de aeronaves, para el entrenamiento de pilotos de la academia de vuelo de la Fuerza Aérea y para pilotos civiles, que luego tendrán un entrenamiento más avanzado y prácticas de vuelo en aeronaves superiores, más grandes y pesadas, de combate supersónicos, bombarderos y de uso civil.

Los pilotos normalmente son entrenados en una aeronave ligera, con dos o más plazas para alumno e instructor, que están sentados juntos como piloto y copiloto. La aeronave puede modificarse para resistir las condiciones de vuelo impuestas por los vuelos de entrenamiento. Este tipo de aeronaves deben poseer cualidades particulares, como por ejemplo, una excelente visibilidad para el alumno y para el instructor, buena maniobrabilidad a media y baja altitud y buenas características acrobáticas.

La aeronave de entrenamiento más popular alrededor del planeta es la fabricada por la compañía Cessna, conocida como Cessna 172. Es un monoplano de ala alta, lo que quiere decir, que las alas se sitúan por encima de la cabina. Algo que resulta especialmente útil para los estudiantes de aviación, pues disfrutan de una mejor visión del terreno y les facilita el aterrizaje, además de proporcionarle al aeronave mayor estabilidad y maniobrabilidad. Cuenta con capacidad para cuatro personas y peso alrededor de 800 kg sin combustible y sin pasajeros. Posee un único motor que alcanza una velocidad máxima de 140 mph (226 km/h). Y, aunque podría llegar hasta los 185 mph (297 km/h), no está recomendado por la marca [17].

Figura 3.1 Aeronave de entrenamiento primario Cessna 172. Tomada de https://cessna.txtav.com/en/piston/cessna-skyhawk [18]

4. SOFTWARE PARA SIMULACIÓN DE VUELO

En este capítulo se describe de manera breve qué es un software para simulación de vuelo, qué software se decide emplear en el presente trabajo y el porqué de su elección.

4.1 Definición

Un Software para simulación de vuelo es un programa dentro del cual están incluidas múltiples aeronaves con sus respectivos modelos matemáticos o tablas de verificación que permiten replicar el comportamiento de una aeronave en vuelo.

Uno de los principales Softwares para simulación de vuelo es X-Plane producido por Laminar Research creado por Austin Meyer.

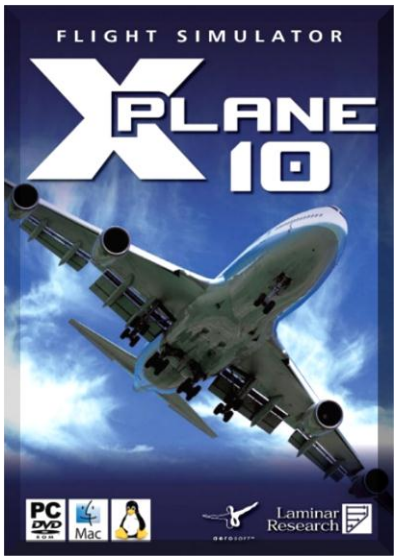


Figura 4.1 Portada del Software de simulación de vuelo X-Plane en su versión 10.

4.2 Información General

X-Plane es el simulador de vuelo más completo y potente del mundo para computadoras personales; ofrece los modelos de vuelo más realistas disponibles.
No es un video juego, sino, una herramienta de ingeniería que se puede usar para predecir las cualidades de vuelo de aeronaves de ala fija y rotativa con una precisión increíble [19].

Debido a que X Plane predice el rendimiento y el manejo de casi cualquier aeronave, es una gran herramienta para que los pilotos mantengan su atención en el simulador que vuela como una aeronave real, para los ingenieros a quienes les permite predecir cómo volará una nueva aeronave y para los entusiastas de la aviación que les posibilita explorar el mundo de las aeronaves.

Este software incluye dinámica de vuelo subsónica y supersónica, lo que permite a los usuarios predecir las características de aeronaves de las más lentas a las más veloces. Incluye también más de 30 aeronaves en su instalación por defecto, que abarca la industria de la aviación y parte de su historia; las aeronaves incluidas van desde el Bell 206 JetRanger y el Cessna 172 hasta el Transbordador espacial y el Bombardero B-2. Además, se pueden descargar más de 2000 modelos desde la web (X-Plane.org, X-Plane.com y Google), muchos de los cuales son gratuitos; la característica más importante para los ingenieros es que se puede diseñar sus propias aeronaves (con la ayuda de su aplicación Plane Maker) e incluirlas al simulador para hacer pruebas de vuelo [20].

Figura 4.2 instalación de escenarios disponibles alrededor del planeta en X Plane

El paquete completo de paisajes de X-Plane cubre la Tierra desde 74° latitud norte hasta 60° latitud sur. Los usuarios pueden aterrizar en cualquiera de los más de 33.000 aeropuertos disponibles alrededor del planeta o sobre plataformas petroleras, fragatas y helipuertos sobre edificios. Se pueden modelar también de manera realista aeronaves a control remoto, hacer el lanzamiento del transbordador espacial y su reingreso a la atmósfera.

Figura 4.3. Cessna 172SP en el aeropuerto Guillermo León Valencia simulado en X Plane.

El clima se puede modificar desde cielos despejados y alta visibilidad hasta tormentas eléctricas con viento, turbulencia y microrráfagas controlables. Se puede afectar los instrumentos debido al clima; en caso de requerirse se puede descargar desde la web el clima en tiempo real de determinada ubicación.

Se incluyen modelos detallados de fallas en sistemas que se pueden hacer fallar de forma manual a la orden de un instructor, o al azar. Las fallas que se pueden simular son fallas en motores, controles de vuelo, cables de control, antenas, trenes de aterrizaje o cualquiera de los sistemas que hagan parte de la aeronave en vuelo.

4.3 Quienes usan X Plane

X-Plane es utilizado por contratistas de defensa, fuerzas aéreas, fabricantes de aeronaves e incluso agencias espaciales líderes en el mundo para aplicaciones que van desde entrenamiento de vuelo hasta diseño de conceptos y pruebas de vuelo; se ha utilizado también en investigaciones de choques para representar la vista que experimentaron los pilotos momentos antes de una colisión en el aire, o para presentar gráficamente a los jurados y jueces las fuerzas, de impacto de una aeronave en vuelo.

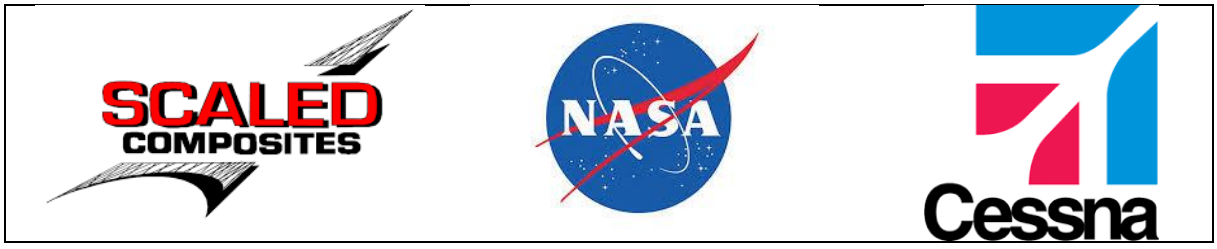


Figura 4.4. Compañías que usan X Plane como software para simulación de vuelo

La compañía Aeroespacial Estadounidense **Scaled Composites** usó X Plane para monitorear los vuelos del Space Ship One (El primer vehículo espacial tripulado de capital privado) al borde de la atmósfera en su simulador de vuelo. La aerolínea de carga **Kalitta Air LLC** usa a X Plane para entrenar a sus pilotos de los 747 de carga en vuelos nocturnos. **Northwest y Japan Airlines** lo usan para la revisión y entrenamiento de vuelos. **Cessna** hace uso de el para capacitar a nuevos clientes en las nuevas funciones del piloto automático Garmin G1000. La **NASA** también lo ha usado para probar el reingreso de planeadores a la atmósfera marciana. Estos clientes son quizá el respaldo más significativo de las capacidades de este software para simulación de vuelo.
X Plane tiene la certificación de la FAA para su uso en el registro de horas de vuelo, para experiencia y calificaciones. Esta experiencia puede dar crédito para una licencia de piloto privado, re entrenamiento, horas de entrenamiento con instrumentos; incluso horas para un certificado de una aerolínea de transporte de carga.

4.4 Salida de Datos

X Plane permite la salida de datos por medio de un puerto serial o a través de un puerto Ethernet, la elección de cuál de los dos tipos de comunicación con el exterior depende de la aplicación que se desarrolle con la información suministrada por el Software y del tipo de dispositivos externos con los que se establezca la comunicación.

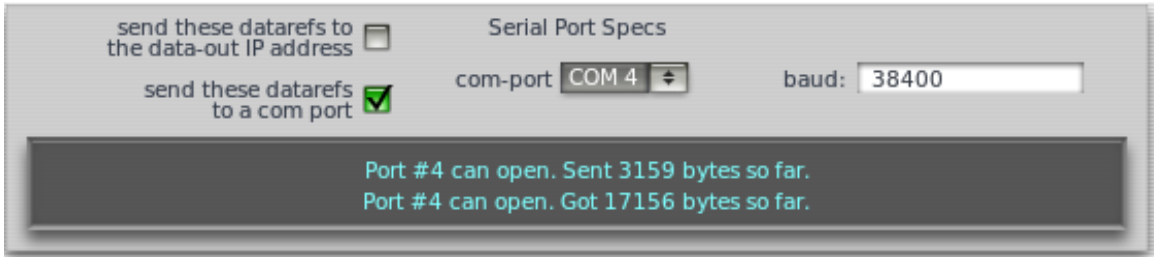


Figura 4.5 Selección de puerto de salida de datos en X Plane

4.4.1 Datarefs

La información suministrada por X Plane se denomina DataRefs . Un DataRef es una referencia de datos que el simulador usa para definir acciones o estados [21]. Una acción tienen la particularidad de evolucionar constantemente mientras que los estados solo tienen dos modos (Activo, inactivo), por ejemplo: Una acción mueve un alerón y un estado determina si las luces están encendidas o apagadas.

Existe una lista disponible de DataRefs que crece día a día, con la cual se puede acceder a datos específicos de una aeronave y del entorno en el que se encuentra [22].

5. ARDUINO MEGA 2560

En este capítulo se hace una descripción de la placa de desarrollo Arduino Mega 2560 sobre la que se implementa el modelo cinemático de la plataforma Stewart en diseño.

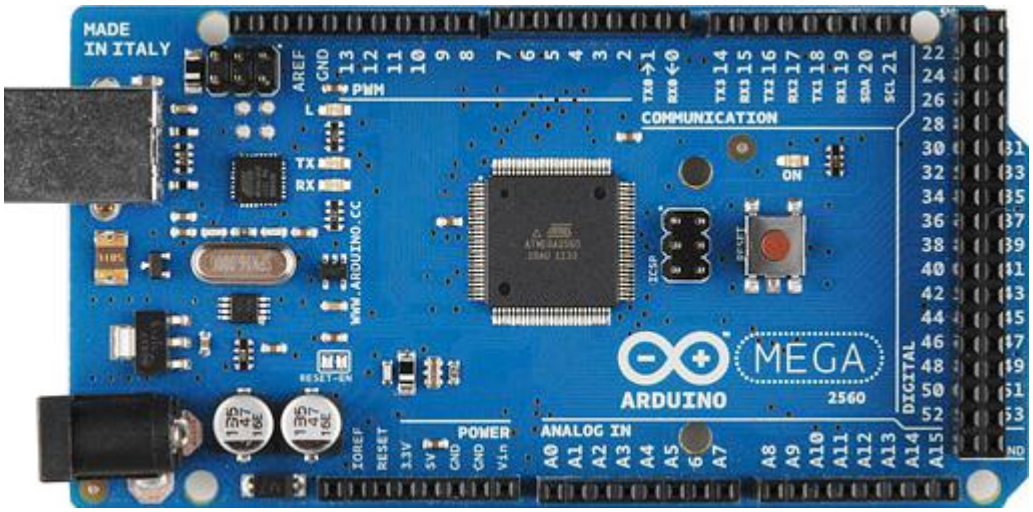


Figura 5.1 Placa de desarrollo Arduino Mega 2560. Tomado de: https://store.arduino.cc/usa/arduino-mega-2560-rev3

5.1 Descripción general: Arduino Mega es una placa de desarrollo de código abierto basada en el microcontrolador ATMEGA 2560 16U2 de Microchip [20] que cuenta con múltiples puertos digitales de entrada y salida (54 en total) de los cuales 15 pueden ser utilizados como salidas PWM, está dotada también con 16 puertos de entrada analógica, 4 UART's (puertos serie en Hardware), un cristal oscilador de 16MHz, una conexión USB, un conector de alimentación, un conector ICSP, y un botón de reset. Debido a sus características es altamente recomendado y usado como placa de impresoras 3D y proyectos de robótica.

Programación: Esta placa se programa con el software Arduino (IDE) que está codificado en C++, está pre programada con un cargador de arranque (bootloader) que permite cargar nuevo código en ella sin el uso de un programador de hardware externo. Se comunica utilizando el protocolo original STK500 (referencia, archivos de cabecera C). También se puede programar el microcontrolador a través del conector ICSP (programación serial en circuito) usando Arduino ISP.

Alimentación eléctrica: Este dispositivo se puede alimentar a través de la conexión USB o de una fuente de alimentación externa. Cuando se alimenta con una fuente externa se recomienda que el voltaje utilizado sea de 7 a 12 V, esto se debe a que si se alimenta con menos de 7 V, el pin 5 V puede suministrar menos

de 5 V a la placa y esta puede volverse inestable, en caso de usar más de 12V, el regulador de voltaje se puede sobrecalentar y acabar por dañar la placa.

Memoria: El microcontrolador Atmega2560 tiene una memoria flash de 256 kB para almacenar código (de la que 8 kB se usan para el bootloader), 8kB de SRAM y 4kB de EEPROM (Esta memoria puede ser leída y escrita con la biblioteca EEPROM)

Puertos de Entrada y Salida: Cada uno de los 54 pines digitales del Mega se pueden usar como entrada o salida, haciendo uso de las funciones pinMode(), digitalWrite() y digitalRead(). Todos los puertos operan a 5 V, cada pin puede proporcionar o recibir 20mA como condición de funcionamiento recomendada.
Pines de puertos Serie: Puerto Serie 0: 0 (RX) y 1(TX); Puerto Serie 1: 19 (RX) y 18 (TX); Puerto Serie 2: 17 (RX) y 16 (TX); Puerto Serie 3: 15 (RX) y 14 (TX). Los pines 0 y 1 también están conectados a los pines Serie del chip ATmega16U2 USB a TTL.
PWM: Los pines 2 a 13 y 44 a 46 proporcionan una salida PWM de 8 bits con la función digitalWrite().
SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). Estos pines soportan la comunicación SPI; los pines SPI también se repiten en el conector ICSP.
I2C: Los pines 20 (SDA) y 21 (SCL) soportan la comunicación I2C.

5.2 Especificaciones Técnicas.

Microcontrolador	ATmega2560
Tensión de trabajo	5V
Tensión de entrada	7-12V
Tensión de entrada (límite)	6-20V
Pines Digitales I/O	54 (de los cuales 15 proporcionan salida PWM)
Pines entrada Analógicas	16
DC Corriente por Pin I/O	20 mA
DC Corriente por Pin 3.3V	50 mA
Memoria Flash	256 KB de los cuales 8 KB se usan por el bootloader
SRAM	8 KB
EEPROM	4 KB
Velocidad del reloj	16 MHz
Largo	101.52 mm
Ancho	53.3 mm
Peso	37 g

Tabla 5.1: Especificaciones técnicas de Arduino Mega 2560.[23]

6. SOLIDWORKS

En el este apartado se habla del software para diseño mecánico SOLID WORKS® producido por la empresa Dassault Systèmes, en el cual, se hace el diseño del manipulador paralelo “Plataforma Stewart”, y desde el cual, se hace el análisis de esfuerzos con lo que se logra determinar las características de los actuadores necesarios para implementar el manipulador.

Solidworks es un software CAD para modelado mecánico en 2D y 3D programado en Visual Basic. Este software cuenta con herramientas de análisis de cuerpos sólidos que permiten establecer parámetros importantes de diseño, tales como resistencia mecánica, desgaste por rozamiento, entre otros [24].

6.1 Diseño de piezas mecánicas

El diseño de piezas en Solidworks es muy simple, pues sólo basta hacer una serie de croquis de la pieza que se quiere crear y el software por medio de operaciones de creación de sólidos genera la pieza que se desea.

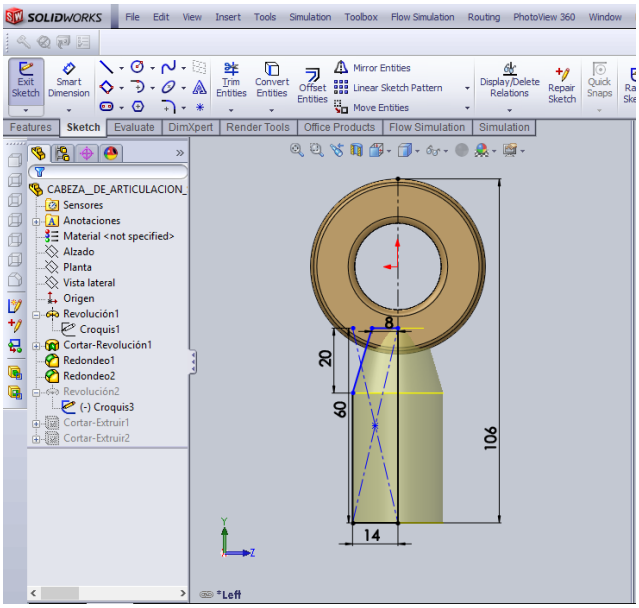


Figura 6.1 Creación de una pieza en Solidworks

Al modelo 3D de la pieza en creación, es posible asignarle un material ya sea desde el banco de materiales incluido en la herramienta o uno creado con parámetros específicos que aún no se encuentren incluidos en el software. En la figura 5.2 se muestra el banco de materiales disponibles y las propiedades específicas de un material seleccionado.

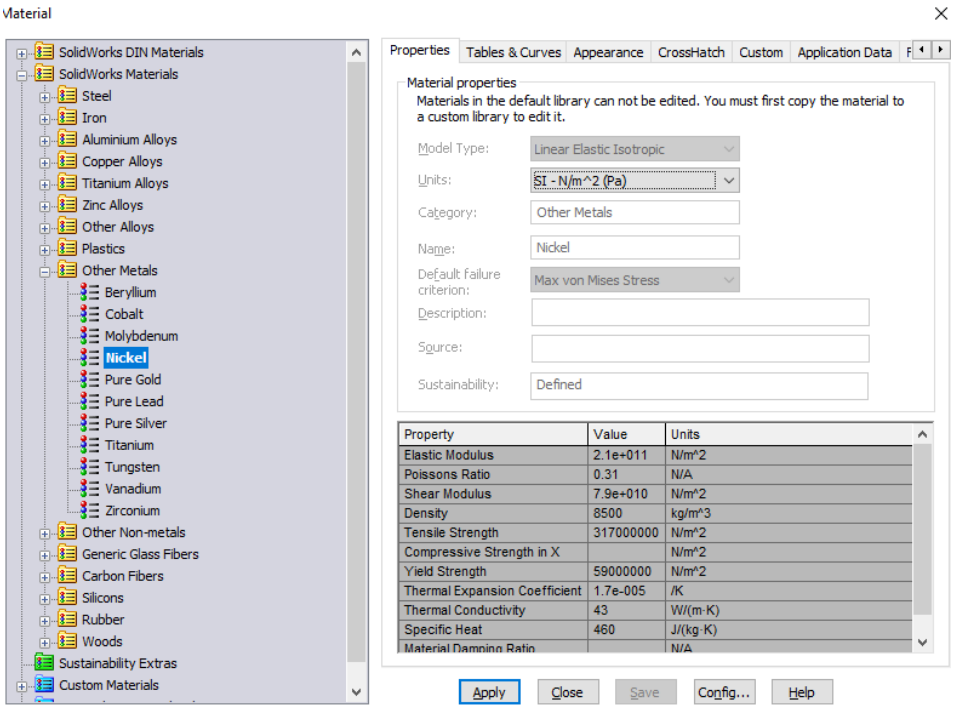


Figura 6.2 Banco de Materiales disponibles en Solidworks

Con el material asignado al modelo 3D se puede acceder fácilmente a las propiedades de masa del modelo, información importante que permite conocer la cantidad de material necesario para construir dicha pieza, conocer propiedades físicas como su momento de inercia, volumen, área superficial entre otras, tal como se muestra en la figura 5.3

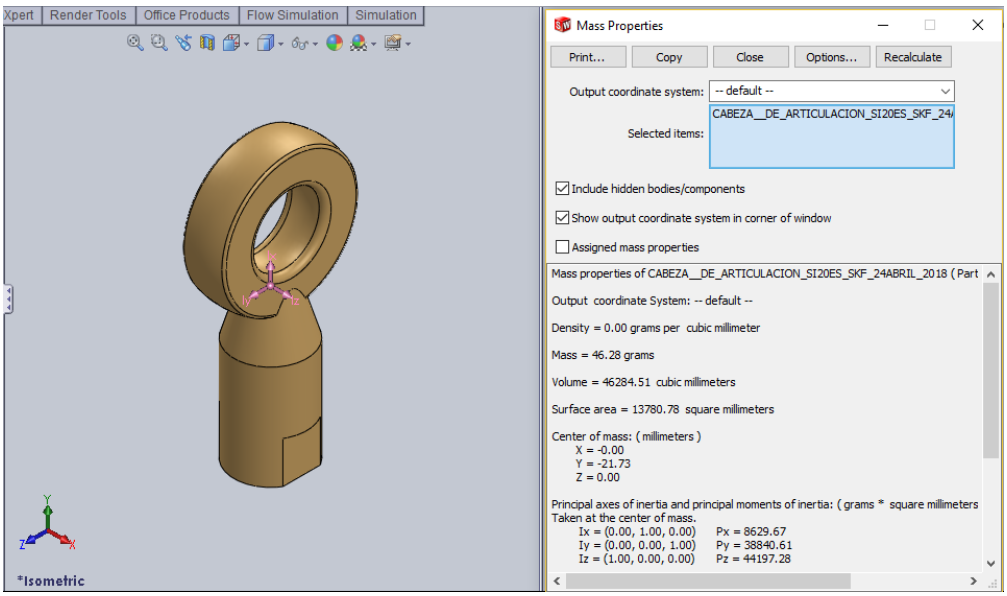


Figura 6.3 Porpiedades de masa de una pieza creada en Solidworks

6.2 Ensamblajes

Un ensamblaje es un grupo de piezas conectadas para crear un mecanismo y/o máquina, para crearlo es necesario importar todos los componentes para establecer relaciones de posición que restringen el movimiento de piezas en un plano o alrededor de un eje y dejar fijas las que sean necesarias.

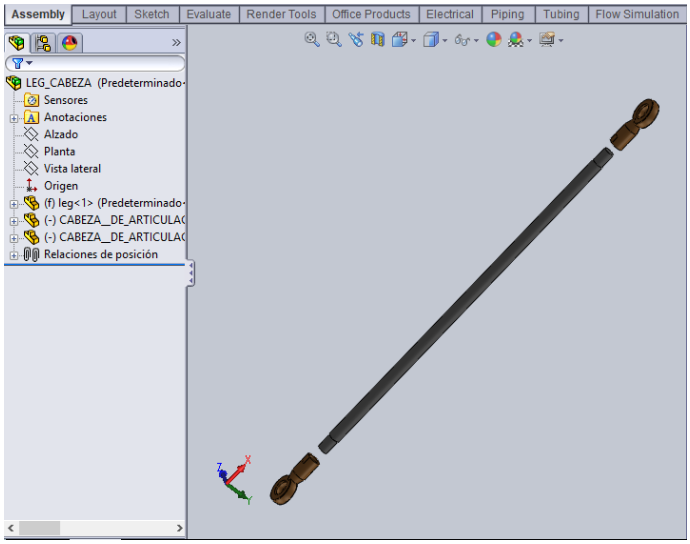


Figura 6.4 Ensamblaje de un link de un manipulador paralelo.

6.3 Análisis de movimiento

Solidworks cuenta con un complemento llamado Solidworks Motion el cual permite hacer análisis cinemático y dinámico de cuerpos rígidos. Para efectos del presente trabajo lo que se busca es establecer el torque y la potencia requerida por los actuadores por lo que el uso de esta herramienta es de suma importancia, debido a que esta permite incluir motores rotativos en los ensamblajes.

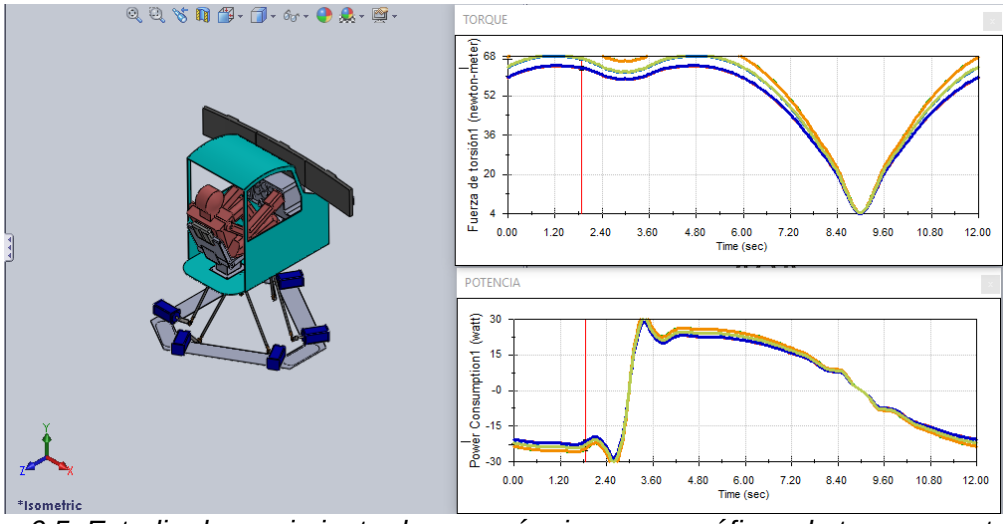


Figura 6.5: Estudio de movimiento de una máquina y sus gráficos de torque y potencia.

PARTE 2: METODOLOGÍA

7. PROCESO DE DISEÑO

7.1 Requerimientos de diseño

Para empezar a desarrollar este proyecto se parte de los requerimientos de diseño del manipulador paralelo, para lo cual se recurre a requerimientos recomendados para simuladores tipo D descritas en Allerton 2009 [25].

Eje	Desplazamiento	Velocidad	Aceleración
Vertical (z)	±34 in	30 in/s	1G
Lateral (y)	±60 in	35 in/s	1G
Longitudinal (x)	±60 in	39 in/s	1G
Pitch	±35°	22°/s	$400°/s^2$
Roll	±33°	24°/s	$400°/s^2$
Yaw	±37°	28°/s	$400°/s^2$

Tabla 7.1: Requerimientos de movimiento de la Plataforma tipo Stewart.

Los valores listados en la *Tabla 7.1* son para el mejor de los casos, esto se debe a que en situaciones donde se presentan movimientos en dos ejes o más de manera simultánea (un aeronave en vuelo siempre produce movimientos en los 6 grados de libertad) se presenta una reducción significativa en el alcance o recorrido de los ejes individuales en términos de desplazamiento.

Para efectos de éste proyecto, sólo se tienen en cuenta los requerimientos de orientación (Pitch, Roll y Yaw) del manipulador; esto debido a que el desplazamiento a lo largo de los ejes x,y,z producidos por el aeronave para el que se diseña el simulador, se pueden simular con desplazamientos menores a los listados en la *Tabla 7.1*.

Otro requerimiento de diseño muy importante es la capacidad de carga del manipulador, para determinar la carga se tienen en cuenta el peso del piloto, todos los instrumentos y accesorios que esté dentro de la cabina del simulador, tal como se lista en la *Tabla 7.2* obteniendo una carga aproximada de 240 kg.

Con los requerimientos mínimos de diseño establecidos es correcto avanzar hasta el siguiente paso del desarrollo del proyecto en donde se determinan parámetros específicos del manipulador paralelo con los que se logra satisfacer los requerimientos establecidos anteriormente.

Elemento	Peso aproximado (kg)
Piloto	80
Silla	20
Panel de Instrumentos	20
Visuales	20
Cabina	50
Otros	50
Total	240

Tabla 7.2: Estimación preliminar de la carga que debe mover el robot paralelo.

7.2. Modelado del manipulador paralelo

Con lo expuesto en el apartado **2.5** se obtiene el modelo cinemático general del manipulador y a partir de las razones expuestas en la sección **2.4**, se establecen los valores de longitud de la arista mayor y menor de la base de la plataforma, la longitud de la palanca de cada actuador y la longitud de los links que unen la palanca de los actuadores con la plataforma móvil, finalmente, se establece la longitud de la arista menor del hexágono que forma la plataforma móvil; siendo entonces la longitud de la arista mayor de la plataforma móvil, el único parámetro faltante por definir.

Al codificar el modelo cinemático en la herramienta Matlab que permite implementar funciones donde se establecen todos los puntos en los cuales es posible posicionar y orientar el manipulador en diseño, se puede determinar de manera acertada el valor de longitud de la arista mayor bajo el cual se satisfacen los requerimientos de diseño establecidos en la *tabla 7.1.*

7.3 Prototipado

Una vez obtenidos los valores de longitud de cada una de las piezas que forman el manipulador paralelo, bajo las cuales se cumplen los requerimientos de diseño respecto al desplazamiento en cada DOF, se procede a hacer un escalado (E 1:5) de las dimensiones de todas las piezas con el fin de construir un prototipo a dicha escala para posteriormente llevar a cabo las pruebas que verifiquen que el modelo matemático obtenido es el adecuado.

Antes de empezar con la construcción física del prototipo, se construye el manipulador y su prototipo en Solid Works para obtener los planos de cada pieza con el fin de facilitar la fabricación de algunas de las piezas del manipulador y la ubicación espacial de las mismas. El siguiente paso es adquirir las piezas para construir el prototipo del manipulador disponibles en el mercado y fabricar las que no se encuentren, como es el caso de la base y la plataforma móvil; la construcción de la base y la plataforma se facilita al tener disponibles los planos

correspondientes a las piezas diseñadas en Solid Works. El material que se elija debe ser uno que sea fácil de manipular y que además permita fijar de forma segura los actuadores en la base y las articulaciones esféricas en la plataforma móvil.

7.4 Calibración de Servomotores

Un paso importante en el proceso de construcción del prototipo a escala es calibrar los servo actuadores para que su recorrido sea el correcto (0° a 180°) lo cual permite reducir posibles errores de posición en la plataforma, para ello se debe crear un código que lleve a cabo dicha tarea y que garantice que cada servo actuador hace su trabajo de manera correcta.

7.5 Construcción de Prototipo.

Finalmente, con todas las piezas a disposición y calibrados los servomotores en el paso anterior, se ubican todos elementos en las posiciones establecidas en los planos obtenidos en SolidWorks.

7.6 Pruebas iniciales del prototipo desde Matlab

Con el prototipo construido correctamente y con el modelo cinemático codificado en Matlab, se procede a construir en la herramienta Simulink, un modelo virtual del manipulador con el fin de conectar el modelo cinemático con el microprocesador para poder controlar el prototipo; todo esto se lleva a cabo haciendo uso del toolbox de Simulink para dispositivos Arduino. Lo anterior permite evaluar el desempeño del prototipo en cada uno de los desplazamientos obtenidos en la evaluación del modelo para cada DOF y poder probar de forma experimental que el modelo desarrollado lleva al manipulador a cumplir con los requerimientos de diseño establecidos.

7.7 Adquisición de datos de X Plane

Después de hacer las pruebas iniciales del modelo, queda por establecer la comunicación entre el microcontrolador y el software de simulación X Plane 10 para obtener los datos del modelo del aeronave (para el que se diseña el simulador) que permiten establecer la orientación y posición del mismo. Hacer esto requiere conocer la forma en la cual X Plane 10 permite obtener información de la aeronave simulada y el tipo de comunicación que se puede establecer. Según el manual de usuario del software se puede establecer dos tipos de comunicación: Serial y Ethernet, teniendo en cuenta las prestaciones del microcontrolador utilizado para tal fin en este proyecto, se debe establecer una conexión tipo Serial.

Ya con la conexión establecida entre X Plane y Arduino se procede a implementar el modelo cinemático del manipulador paralelo en el microcontrolador para finalmente obtener un prototipo funcional del Simulador de Vuelo en diseño.

7.8 Diseño de cabina para simulador de vuelo dinámico.

Una parte muy importante en un simulador dinámico de vuelo, aparte del software de simulación, el sistema robótico (que permita generar movimientos similares al de una aeronave en vuelo) y un conjunto de instrumentos para el control de la aeronave, es sin duda, el entorno donde se lleva a cabo dicha simulación, en este caso la cabina del aeronave Cessna 172. Por lo tanto, es necesario diseñar una cabina con características espaciales similares a las de la aeronave para la cual se diseña.

7.9 Análisis de movimiento

Con una cabina diseñada y ubicada sobre el robot paralelo, es necesario hacer un análisis de esfuerzos para definir el torque necesario que debe producir cada actuador cuando se generan los movimientos requeridos por el manipulador y poder elegir en el mercado los servomoteres adecuados, dicho análisis se hace en el software Solid Works con ayuda de la herramienta Solid Works Motion. Es necesario resaltar que dicho análisis se aplica sólo al modelo 3D del manipulador en tamaño real.

PARTE 3: RESULTADOS Y DISCUSIÓN

8. PLATAFORMA STEWART.

8.1 Modelo cinemático del manipulador paralelo en Matlab

Para determinar la longitud de la arista mayor de la plataforma, se codifica el modelo cinemático del manipulador paralelo en Matlab, fijando los valores de longitud establecidos previamente y variando de forma manual el posible valor del parámetro en búsqueda, empezando con el valor establecido para la arista mayor de la base y reduciendo progresivamente su valor hasta encontrar el valor para el cual se cumple con los requerimientos de diseño establecidos en la tabla 7.1, el primer valor con el que se logra cumplir con dichos requerimientos es 500 mm, pero no se elige este valor porque aunque es un valor que sirve para el propósito, debido a las restricciones mecánicas de las articulaciones, se predice que al construir el manipulador no se alcanzarán las orientaciones requeridas, por lo que se sigue reduciendo el valor de longitud de la arista mayor de la plataforma hasta encontrar uno que teóricamente produzca valores de orientación por encima de los establecidos como requerimientos.
A continuación, se muestran los valores de orientación obtenidos con diferentes longitudes de la arista mayor de la plataforma.

DOF / Valor	Pitch(°)	Roll (°)	Yaw (°)	Surge(mm)	Sway (mm)	Heave (mm)
Mínimo	-20	-19	-18	-230	-180	465
Máximo	20	19	18	175	180	860

***Tabla 8.1**. Valores de orientación y posición en cada DOF con longitud de la arista mayor de la plataforma = 1000 mm.*

DOF / Valor	Pitch(°)	Roll (°)	Yaw (°)	Surge(mm)	Sway (mm)	Heave (mm)
Mínimo	-33	-31	-32	-215	-200	495
Máximo	33	34	-32	230	200	885

***Tabla 8.2**. Valores de orientación y posición en cada DOF con longitud de la arista mayor de la plataforma = 600 mm.*

DOF / Valor	Pitch(°)	Roll (°)	Yaw (°)	Surge(mm)	Sway (mm)	Heave (mm)
Mínimo	-39	-43	-37	-200	-195	490
Máximo	40	39	37	235	195	880

***Tabla 8.3**. Valores de orientación y posición en cada DOF con longitud de la arista mayor de la plataforma = 500 mm.*

DOF / Valor	Pitch(°)	Roll (°)	Yaw (°)	Surge(mm)	Sway (mm)	Heave (mm)
Mínimo	-41	-61	-40	-195	-190	480
Máximo	42	42	40	235	190	875

***Tabla 8.4**. Valores de orientación y posición en cada DOF con longitud de la arista mayor de la plataforma = 450 mm.*

DOF / Valor	Pitch(°)	Roll (°)	Yaw (°)	Surge(mm)	Sway (mm)	Heave (mm)
Mínimo	-43	-62	-41	-190	-190	480
Máximo	44	43	41	235	190	475

***Tabla 8.5**. Valores de orientación y posición en cada DOF con longitud de la arista mayor de la plataforma = 425 mm.*

Se decide entonces tomar el valor de longitud para a la arista mayor de la plataforma como 425 mm, esto debido a que ya se está por encima de los valores exigidos en los requerimientos, y además de eso, para evitar que hayan una mayor reducción en la movilidad a lo largo de los ejes X, Y y Z.

Lo que se mostró en la figura 2.7 indica la distancia de separación entre los puntos donde se unen las articulaciones rotacionales en la base y las articulaciones esféricas en la plataforma móvil; es esa la distancia que se determina. En la siguiente tabla se listan todos los valores de los parámetros que determinan por completo el manipulador paralelo diseñado en escala 1:1 y 1:5 respetivamente.

Pieza	Longitud (mm)
Arista mayor plataforma (a)	425
Arista menor plataforma (c)	50
Arista mayor base (b)	1000
Arista menor base (d)	200
Palanca actuador (p)	165
Leg palanca plataforma (l)	850
Arista mayor plataforma prototipo	85
Arista menor plataforma prototipo	10
Arista mayor base prototipo	200
Arista menor base prototipo	40
Palanca actuador prototipo	33
Leg palanca plataforma prototipo	170

***Tabla 8.6**. Longitud de parámetros de diseño para Manipulador paralelo y su prototipo a escala 1:5.*

Es preciso ahora mostrar el modelo codificado en la herramienta Matlab con la que se logra establecer los parámetros necesarios para definir completamente el manipulador paralelo. Suposición

```
%  DEFINICIÓN DE CONSTANTES
% Valores medidos en mm.
a = 425;   % Arista mayor plataforma
b = 1000;  % Arista mayor base
c = 50;    % Arista menor Plataforma
d = 200;   % Arista menor base
p = 165;   % Longitud de la palanca del actuador
l = 850;   % Longitud de la unión entre P y cada vértice de la plataforma
S = [ -120 -120 0 0 120 120 ]; %Ángulos que debe ser rotado cada vértice
en torno al eje Z

% Vectores de ángulos de orientación A, B, C. Vectores de posición
Px,Py,Pz.
A = (-90:1:90);
B = (-90:1:90);
G = (-90:1:90);
Px = (-1000:5:1000);
Py = (-1000:5:1000);
Pz = (0:5:2000);

% Cálculo de matrices de posición, orientación y ángulos en los
actuadores

M1 = Mateval(0,0,0,0,0,Pz); %Matriz de avance a lo largo del eje z
M_Heave = MValServo(a,b,c,d,p,l,S,M1); % Matriz de posición plataforma y
ángulos de los Servos en eje z

% El puntol en z en el que se alcanzó un mayor desplazamiento en los ejes
% x,y fue Pz = 675
M2 = Mateval(0,0,0,0,Py,675); %Matriz de avance a lo largo del eje y
M_Sway = MValServo(a,b,c,d,p,l,S,M2); % Matriz de posición plataforma y
ángulos de los Servos en eje y

M3 = Mateval(0,0,0,Px,0,675); %Matriz de avance a lo largo del eje x
M_Surge = MValServo(a,b,c,d,p,l,S,M3); % Matriz de posición plataforma y
ángulos de los Servos en eje x

M4 = Mateval(0,0,G,0,0,675); %Matriz de avance a rededor del eje z
M_Yaw = MValServo(a,b,c,d,p,l,S,M4); % Matriz de posición plataforma y
ángulos de los Servos al rededor de z

M5 = Mateval(0,B,0,0,0,675); %Matriz de avance a rededor del eje x
M_Roll = MValServo(a,b,c,d,p,l,S,M5); % Matriz de posición plataforma y
ángulos de los Servos alrededor de x

M6 = Mateval(A,0,0,0,0,675); %Matriz de avance a rededor del eje y
M_Pitch = MValServo(a,b,c,d,p,l,S,M6); % Matriz de posición plataforma y
ángulos de los Servos alrededor de y
```

La descripción de las funciones *Mateval* y *MValServo* se da en el Anexo 1.

8.2 Diseño 3D del manipulador paralelo.

Con los datos que se muestran en la *Tabla 8.6* los cuales confirman que se han alcanzado los requerimientos de diseño, se procede entonces a construir el manipulador en 3D con el fin de desarrollar los análisis correspondientes.

El primer paso que se da es diseñar las Base y la Plataforma móvil del robot. Para llevar a cabo esta misión se opta (por facilidad en su fabricación) por construir para la plataforma móvil un hexágono como el que se muestra en la *figura 8.1a,* y para la base un hexágono como el que se muestra en la *figura 8.1b* en Solid Works que permite ubicar de manera óptima los puntos donde se conectan las articulaciones rotacionales, esféricas, servomotores y todos los elementos que conforman el manipulador paralelo.

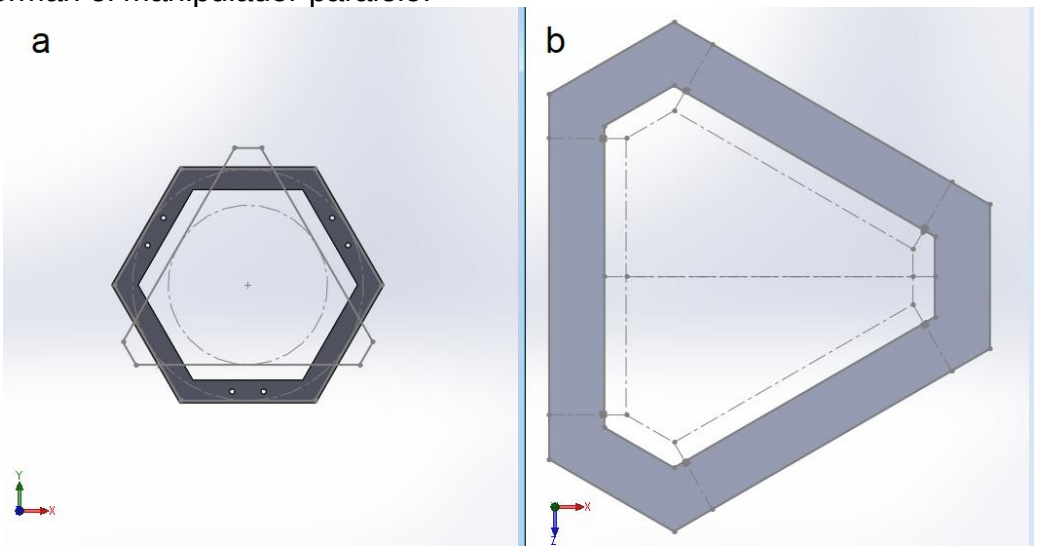


Figura 8.1. Piezas diseñadas en Solidworks para: a. Plataforma móvil, b. Base plataforma.

Para determinar las dimensiones de las piezas que se deben construir se sobreponen los hexágonos irregulares de la plataforma y de la base con las dimensiones listadas en la tabla 8.6 respectivamente tal como se muestra en la figura 8.1a y 8.1b y teniendo en cuenta las dimensiones de las articulaciones y sus uniones se determinan los parámetros de las piezas y los puntos donde deben ubicarse las articulaciones y/o sus uniones. El mismo proceso se hace para las piezas de la plataforma en dimensiones a escala.

El paso siguiente es modelar las articulaciones esféricas y los links entre articulaciones. El tipo de articulaciones que se eligen son las SI20ES de la empresa SKF, su modelo se crea a partir de los planos proporcionados por el fabricante.

De la misma manera se diseñan y se modelan los legs que unen las articulaciones esféricas a los brazos de los Servomotores y a la plataforma móvil.

Los sólidos de los servomotores se modelan sin detalles, esto teniendo en cuenta que se debe hacer un análisis de movimiento para determinar el torque y la potencia necesaria para llevar a cabo los movimientos que la plataforma requiere, pero antes de ése análisis, se deben conocer las propiedades físicas del simulador terminado. Dicho esto, para el modelado de los servos sólo se hace un esquema que lo represente, lo mismo pasa con los brazos conectados a estos. La plataforma diseñada se muestra en la figura 8.3.

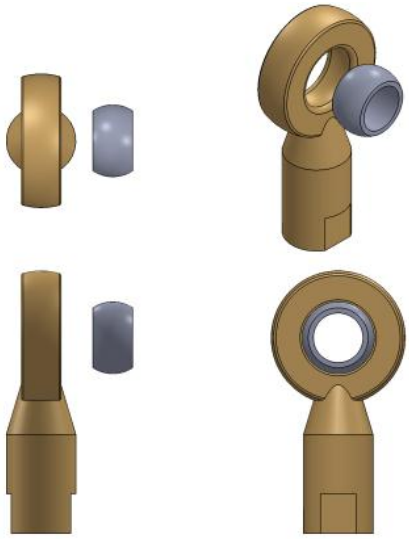


Figura 8.2.Articulación SI20ES SKF modelada en Solidworks

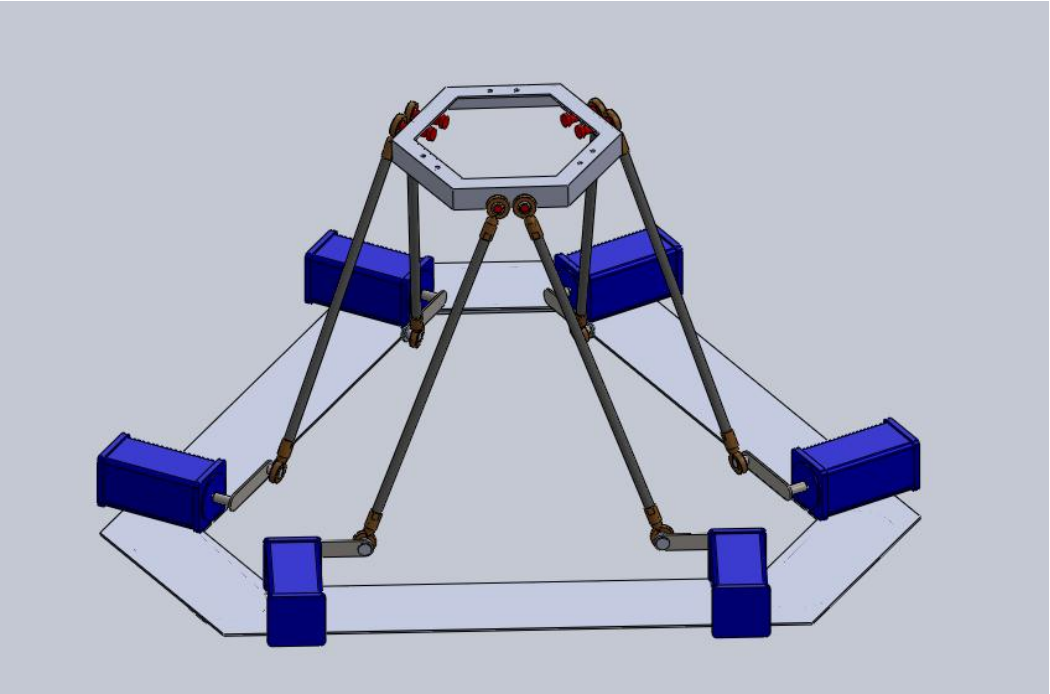


Figura 8.3. Plataforma Stewart para Simulador de vuelo.

8.3 Construcción de prototipo de Manipulador Paralelo

Lo que sigue entonces es construir el prototipo a escala de la plataforma Stewart. La escala que se elige es 1:5 debido a los materiales disponibles en el mercado nacional y a su costo moderado. Para llevar a cabo esta misión y teniendo la información de las dimensiones de cada parte de la plataforma se buscan en el mercado de la robótica piezas que cumplan con esos requerimientos; se encuentran entonces articulaciones esféricas en miniatura, servomotores para robótica, brazos para servomotores de las dimensiones adecuadas, y barras metálicas de longitud fácilmente ajustable para el propósito del presente proyecto.
Cada uno de los elementos que hacen parte del prototipo también se modela en SolidWorks, esto con el fin de lograr una construcción fiel al modelo y obtener los resultados esperados.

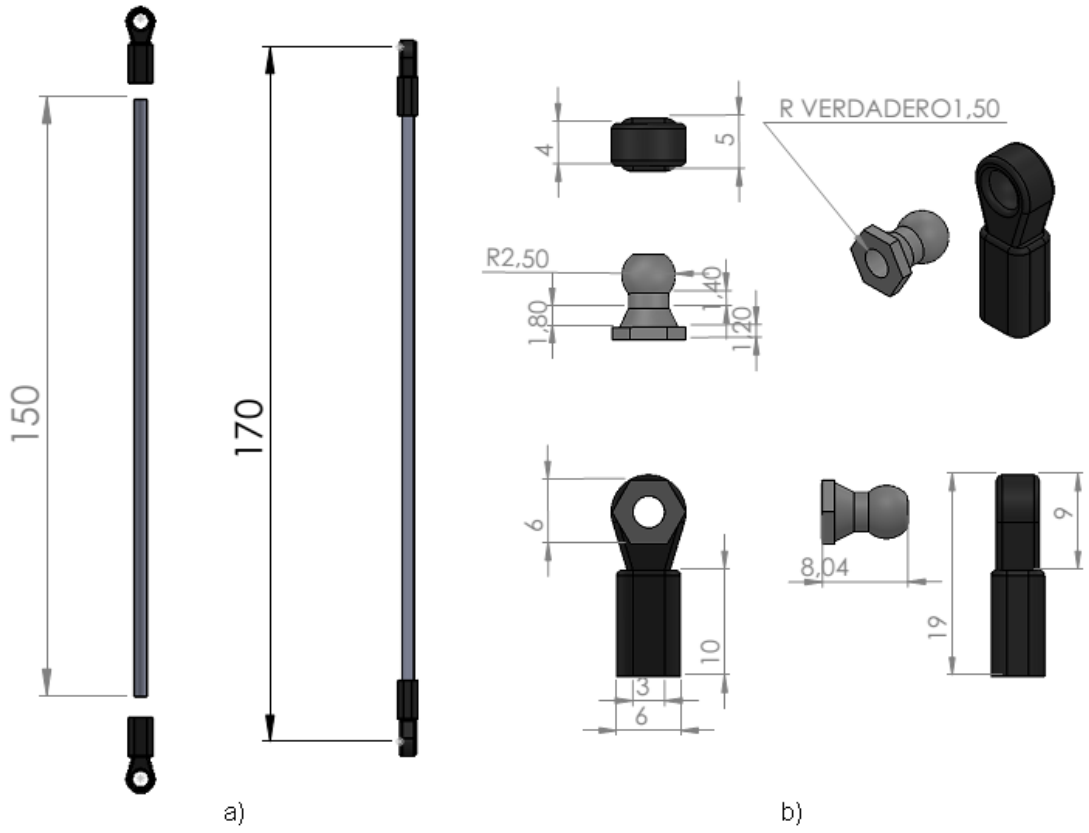


Figura.8.4 *Eslabón y articulación para plataforma prototipo. Todas las medidas se dan en mm*

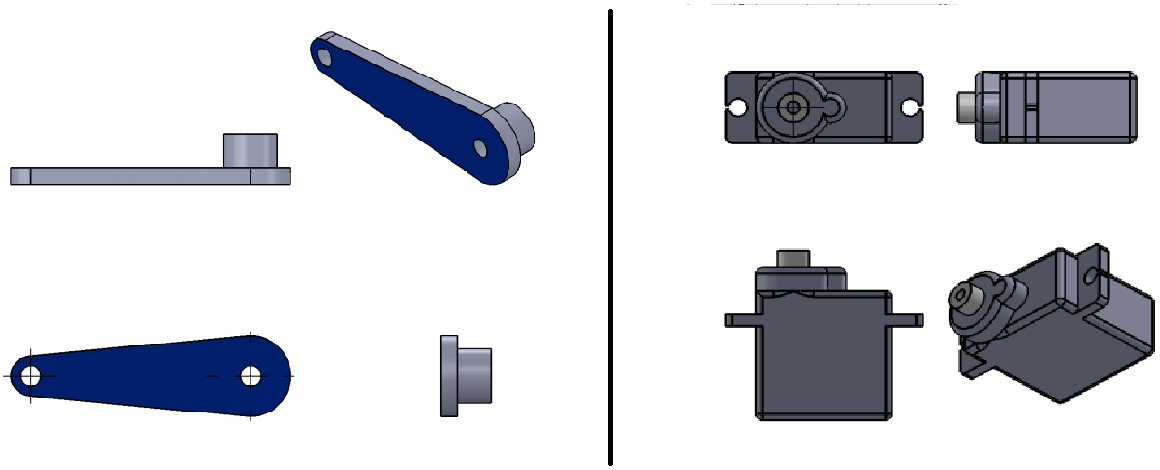


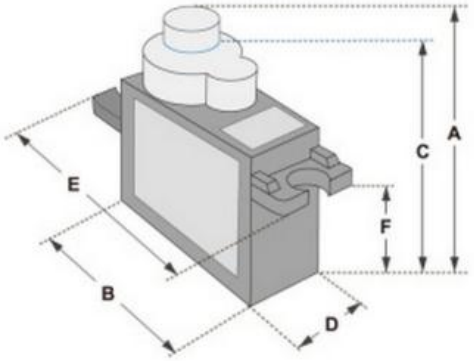
Figura.8.5. Brazo extendido para Servo SG90 y Servo SG90 para plataforma prototipo

Con las piezas encontradas en el mercado, quedan faltando para construir el prototipo del robot la base donde se ubican los actuadores y la plataforma móvil.

Teniendo en cuenta que son piezas de dimensiones específicas, es necesario diseñarlas y construirlas.

Siguiendo los pasos de diseño de la base y la plataforma móvil del robot de ascala 1:1 se obtienen los diseños de estas piezas para el robot en su versión a escala 1:5. Para la construcción en físico de dichas piezas se utiliza MDF de 12 mm; esto debido a la facilidad de corte y a la resistencia del material.
Se elige el MDF del espesor indicado para poder taladrar y fijar de forma segura piezas como las uniones entre las articulaciones y la plataforma móvil.

Se hace un análisis de movimiento del prototipo con el fin de corroborar que los actuadores empleados pueden soportar de forma cómoda la carga con la de deben lidiar.



Dimensiones y especificaciones
A(mm) : 32
B(mm) : 23
C(mm) : 28.5
D(mm) : 12
E(mm) : 32
F(mm) : 19.5
Torque(N-mm): 250
Peso(g) : 14.7
Voltaje : 4.8 – 6

Figura 8.6. Características de Servomotor Tower pro SG90.
Para poder hacer el estudio de movimiento es necesario conocer las propiedades de masa del prototipo, las cuales se muestran en la figura 8.7.

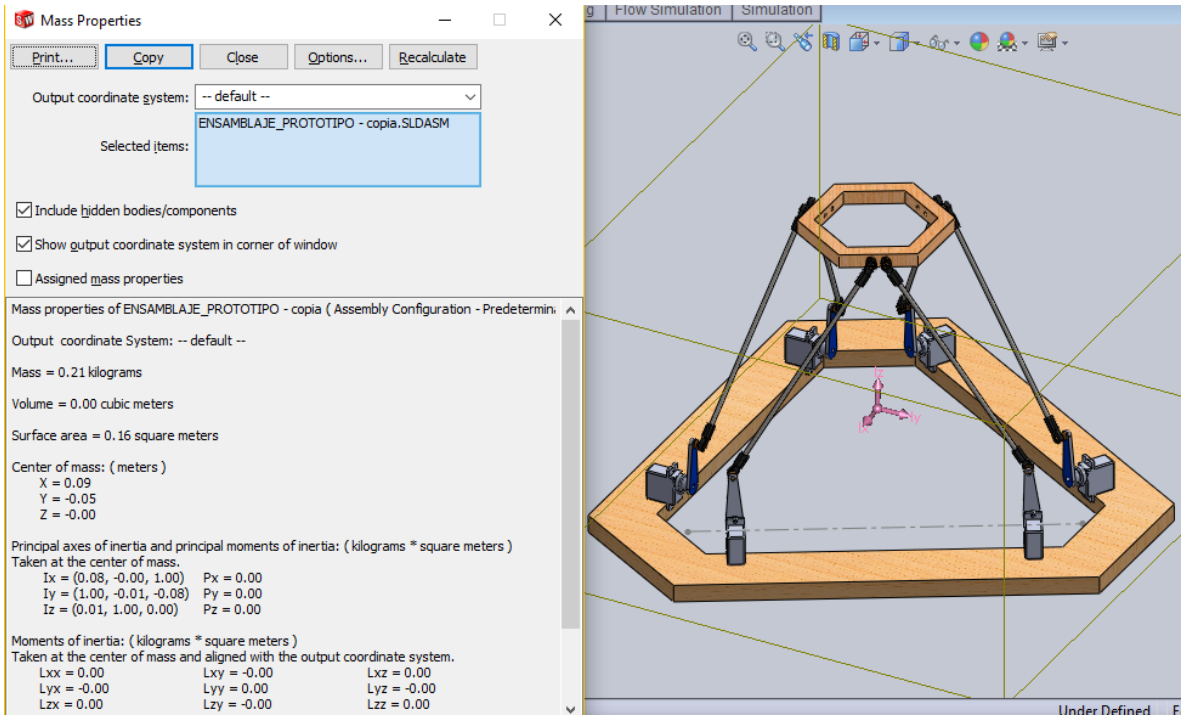
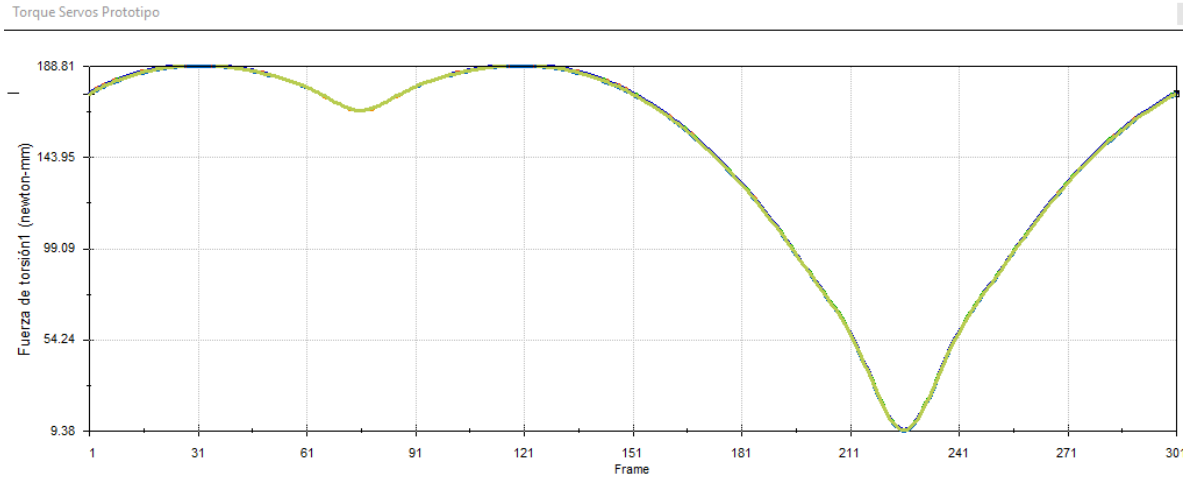


Figura 8.7. Propiedades físicas de plataforma Stewart prototipo a escala

Los resultados de torque necesario para mover la plataforma se muestran en la gráfica 8.1, donde se observa que los servos seleccionados como actuadores, tienen propiedades de torque superiores a los valores requeridos.



Gráfica 8.1. Torque necesario para accionar la plataforma Stewart prototipo.

Antes de proceder a ensamblar el prototipo, se hace una calibración previa de la posición de cada servomotor con la ayuda de un código incluido en el paquete de

ejemplos de Arduino y de una guía de ángulos fabricada para tal fin mostrada en la figura 8.8.

```
#include <Servo.h>

Servo myservo;
// create servo object to control a servo
int pos = 0;
// variable to store the servo position

void setup() {
  myservo.attach(9);
// attaches the servo on pin 9 to the servo object
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) {
// goes from 0 degrees to 180 degrees in steps of 1 degree
    myservo.write(pos);
// tell servo to go to position in variable 'pos'
    delay(15);
// waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) {
// goes from 180 degrees to 0 degrees
    myservo.write(pos);
// tell servo to go to position in variable 'pos'
    delay(15);
// waits 15ms for the servo to reach the position
  }
}
```

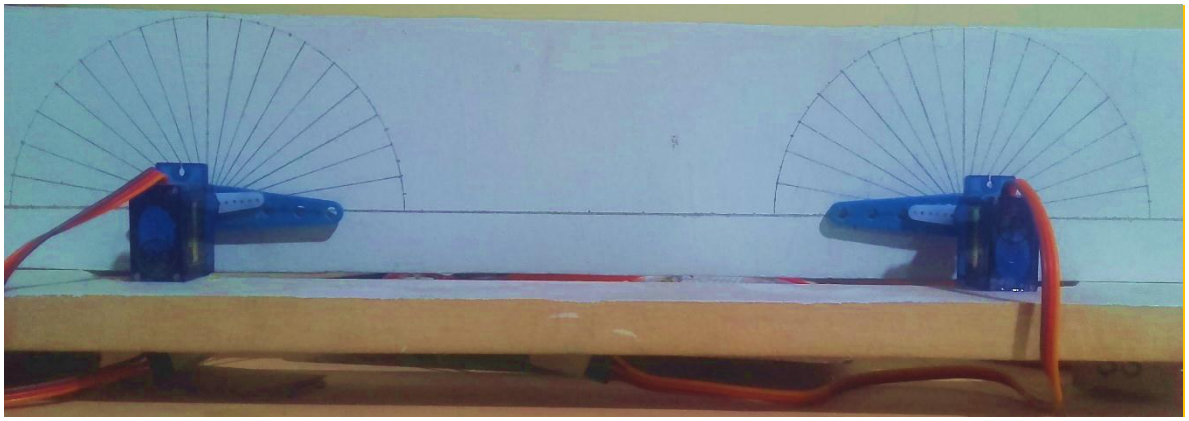


Figura 8.8.Guía para Calibración de posición de Servomotores

Con los resultados del análisis que confirman que los actuadores elegidos tienen el torque necesario y con todas las piezas diseñadas y construidas se procede a

ensamblar el modelo 3D y a construir en físico el prototipo. En la figura 8.9. se muestran los prototipos en físico y en Solidworks.

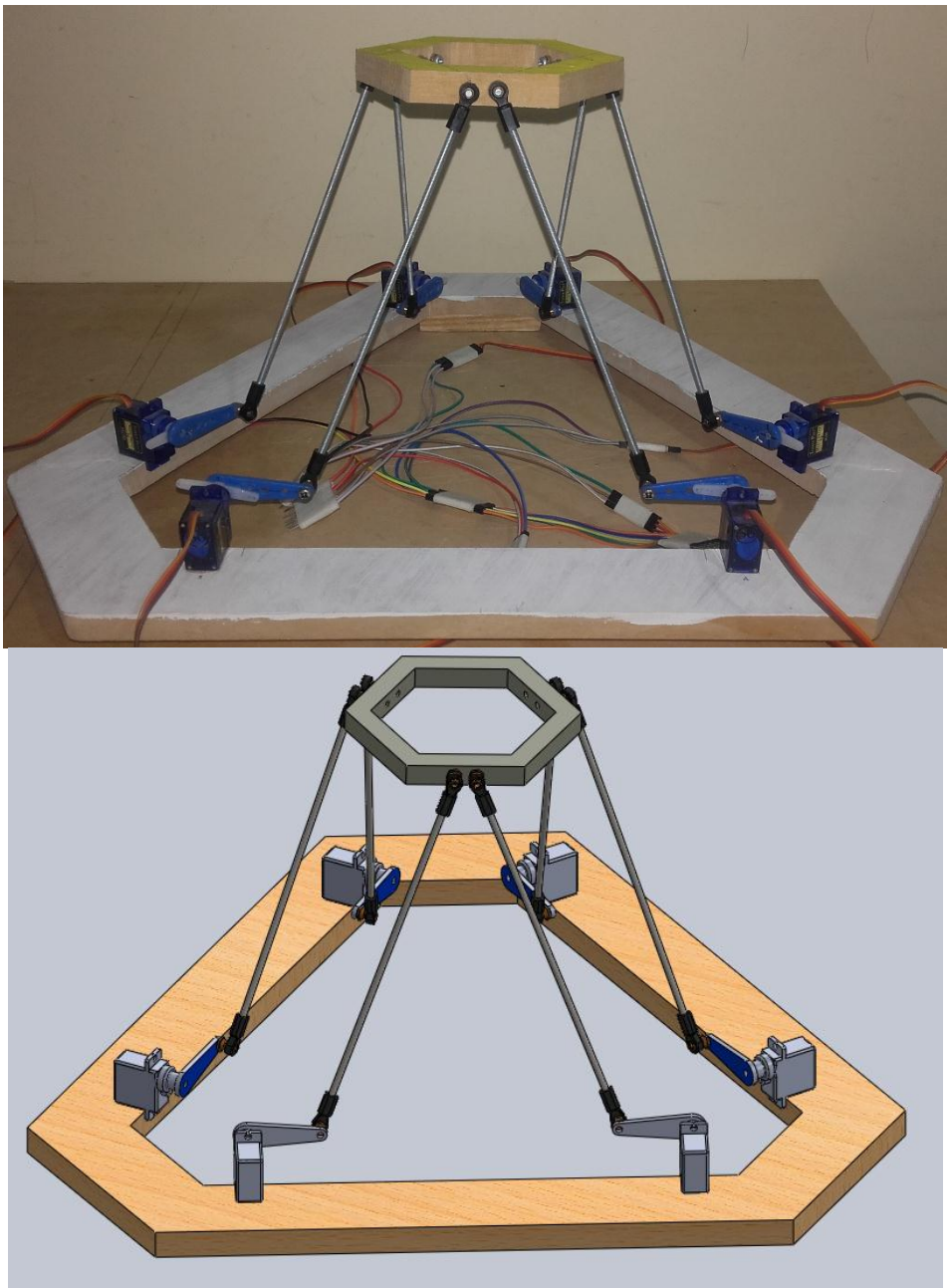


Figura 8.9. Prototipo Plataforma Stewart. Arriba: Prototipo en físico. Abajo: Prototipo en Solidworks.

Los materiales y piezas utilizadas en la construcción del prototipo se listan en la siguiente tabla.

PIEZA	DESCRIPCIÓN	CANTIDAD
Base de Plataforma	Fabricada en MDF de 13mm	1
Plataforma Móvil	Fabricada en MDF de 13mm	1
Micro Servo	Tower Pro SG90	6
Palanca Servo	Conector largo en ABS	6
Articulación esférica 5mm	Incluido en Tamiya 3mm Threaded Shaft set	12
Cabeza para articulación 5mm	Incluido en Tamiya 3mm Threaded Shaft set	12
Tornillo 3mmx10mm	Incluido en Tamiya 3mm Threaded Shaft set	6
Tornillo 3mmx25mm		6
Tuerca 3mm		18

Tabla 8.7. Componentes necesarios para construir el prototipo a escala de la plataforma Stewart.

8.4 Pruebas iniciales del prototipo en Matlab.

Como se explicó en el apartado 7.6, se crea en la herramienta Simulink de Matlab el modelo del manipulador paralelo, con el fin de enlazarlo directamente con el prototipo y así llevar a cabo pruebas de movilidad de este.

A continuación se muestra el esquema del modelo en Simulink

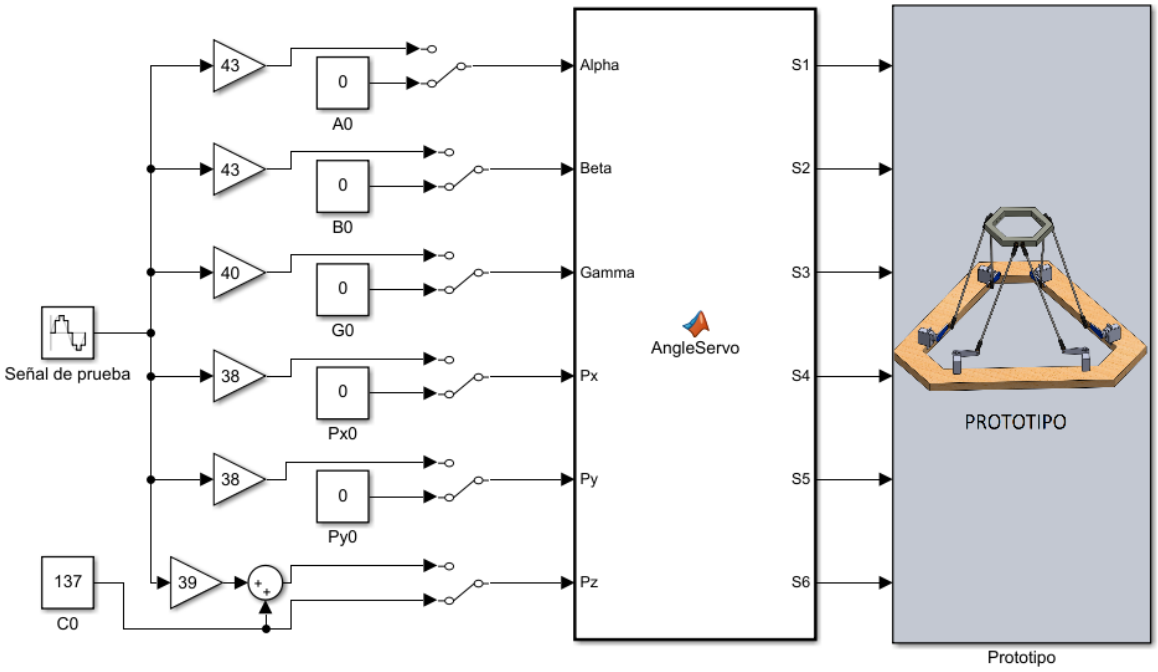


Figura 8.10. Esquema del modelo del prototipo en Simulink

En la parte de la izquierda afuera de los bloques se encuentra una señal sinusoidal de amplitud unitaria que se usa como señal de prueba que está

conectada a ganancias correspondientes a desplazamiento a cada uno de los grados de libertad de la plataforma prototipo, los cuales pueden ser probados de forma independiente, al activar o desactivar el correspondiente su correspondiente interruptor.

Dentro del bloque *AngleServo* se encuentra codificado el modelo cinemático de la plataforma con valores correspondientes al prototipo, en el bloque etiquetado como “Prototipo” están las salidas del modelo conectadas a salidas digitales de Arduino, en el caso de las señales correspondientes a los servos 2, 4 y 6 se aplica la señal directamente; en el caso de los servos 1, 3 y 5 la señal debe invertirse tal como se muestra en la *figura 8.11.*

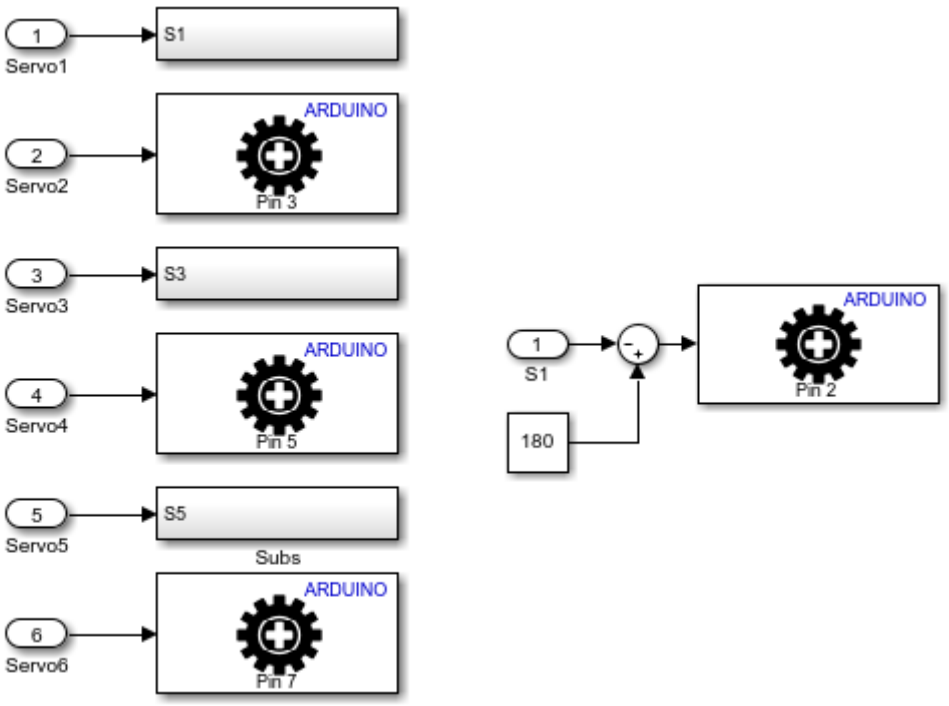


Figura 8.11. Interior del bloque “Prototipo”. Izquierda: Señales aplicadas a las salidas digitales de Arduino. Derecha: Señal que debe ser Invertida para Servos 1, 3 y 5.

Lo siguiente por hacer antes de poder llevar a cabo las pruebas de movilidad es crear una rutina en Arduino donde se lee la orientación de la plataforma en cada DOF, para ello se utiliza el sensor UM7LT de CH Robotics el cual permite llevar a cabo esta tarea de forma adecuada.

A continuación, se muestra el código implementado en el IDE Arduino para hacer la lectura de los ángulos de inclinación de cada DOF de orientación de la plataforma del manipulador paralelo.

```
#include <UM7.h>

UM7 imu;
float Alpha_imu = 0;
float Beta_imu = 0;
float Gamma_imu = 0;

void setup()
{
 Serial.begin(38400);
 Serial1.begin(38400);
}

void loop() {
    if (Serial1.available() > 0)
        {
            if (imu.encode(Serial1.read()))
                {  // Reads byte from buffer.  Valid packet returns true.
                    Alpha_imu = (imu.pitch)/91.02222;
                    Beta_imu  = (imu.roll)/91.02222;
                    Gamma_imu = (imu.yaw)/91.02222;
// El fabricante de la UM7LT pide dividir el valor  delos registros imu.x
// (x: pitch, roll, yaw) entre 91.02222 para obtener el valor en grados.

                    Serial.print(Alpha_imu);
                    Serial.print(" ");
                    Serial.print(Beta_imu);
                    Serial.print(" ");
                    Serial.print(Gamma_imu);
                    Serial.println(" ");
                }
         }
}
```

La ejecución de este código se lleva a cabo en una segunda placa Arduino dedicada solo a esta tarea, pudiendo de esta manera visualizar de forma sencilla los datos que se están buscando. En la tabla 8.8, se muestran los resultados de orientación preliminares obtenidos y se evidencia que aunque no se alcanzan los valores obtenidos de forma teórica listados en la tabla 8.5 debido a restricciones mecánicas de las articulaciones, aun así se obtienen valores de movilidad que cumplen con los requerimientos de diseño propuestos en la tabla 7.1. En la sección 8.5.3 se describe el procedimiento para llegar a estos valores.

Pitch max	Pitch min	Roll max	Roll min	Yaw max	Yaw min
38°	-35°	35°	-35°	39°	-37°

Tabla 8.8 Resultados experimentales de movilidad.

8.5 Implementación del modelo cinemático del manipulador paralelo en Arduino

Con el modelo cinemático codificado y probado en Matlab y con la construcción del prototipo del manipulador paralelo a escala ya se tienen las herramientas para proceder a implementar dicho modelo en la tarjeta de desarrollo Arduino Mega.

Los pasos a seguir para dicho propósito son descritos a continuación.

8.5.1 Adquisición de datos de orientación y posición desde el software de simulación de vuelo X Plane 10.

El software de simulación X Plane 10 permite tener acceso a una gran cantidad de datos durante una simulación de vuelo; para el propósito de este proyecto los datos que se requieren son los de orientación y posición de la aeronave simulada.

La salida de los datos desde el software se hace por medio de DataRefs que se envían a través de un puerto serial o a una dirección IP; en éste proyecto en específico se decide usar el puerto serial como medio de comunicación debido a que la tarjeta de desarrollo en la que se implementa el modelo cuenta con múltiples puertos serie. Para el caso de comunicación con X Plane se necesita uno y para enviar datos de control hasta los actuadores se necesita un puerto adicional.

El procedimiento para acceder a la información de interés es el que se describe a continuación: Desde el menú principal de X Plane acceder de forma directa a las opciones en **Ajustes/Entrada y salida de datos/Dataref-out,** en la opción **Select Datarefs** se escriben las direcciones de acceso a los datos necesarios, en éste caso se necesita conocer los datos de orientación y posición como se mencionó anteriormente.
En la tabla siguiente se listan los identificadores de las direcciones de dicha información.

Datos	Dirección Dataref
Pitch	sim/cockpit/gyros/the_ele_ind_deg
Roll	sim/cockpit/gyros/phi_ele_ind_deg
Yaw	sim/flightmodel/position/Rrad
Surge	sim/flightmodel/position/local_ax
Sway	sim/flightmodel/position/local_ay
Heave	sim/flightmodel/position/local_az

Tabla 8.9. Identificador de datos de orientación en X Plane

Paso siguiente se elige la opción **WRITE** para que el software sepa que estos datos serán transmitidos a una dirección IP o a un puerto serie. Como ya se mencionó, en este caso la comunicación se hace vía puerto serie, por lo tanto se selecciona la opción '**send these datarefs to a com port**'. El manual de X Plane recomienda que la transmisión de datos debe hacerse a una velocidad de 38400 baudios. Los datos enviados por el puerto seleccionado están en formato *String* separado por una coma.

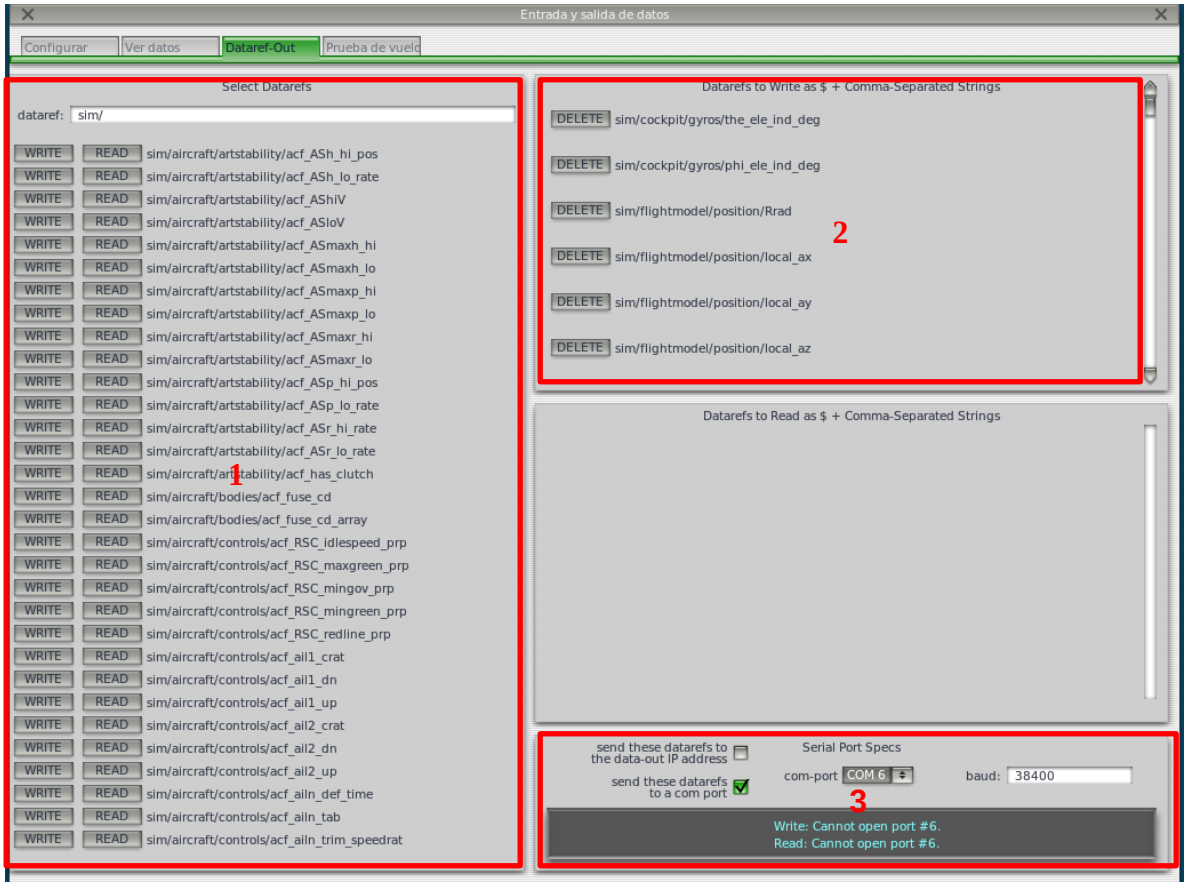


Figura 8.12. Interfaz de usuario de X Plane 10 para exportar datos por medio de Datarefs. 1: Direcciones de acceso a datos. 2: Datos seleccionados. 3: Configuración de envió de datos (puerto y velocidad de envío).

Es necesario entonces decodificar los datos enviados desde X Plane 10 por el puerto serial. Esto se hace por medio del código mostrado a continuación, en este se describe paso a paso la captura de la cadena de datos encontrada en el puerto, la separación de la cadena en grupos separados por una coma (,) y su posterior transformación de datos tipo cadena a datos tipo flotante, que son el tipo de dato con el que se lleva a cabo la implementación del modelo cinemático. Como se mencionó al inicio de esta sección ahora se está trabajando en Arduino por lo cual todo lo que sigue está codificado en el IDE Arduino.

```
//ADQUISICIÓN DE DATOS DESDE X PLANE 10

while (Serial2.read() != '$');
Data[Serial2.readBytesUntil(',', Data, sizeof(Data) - 1)] = 0;
float  Alpha = atof(Data);
Data[Serial2.readBytesUntil(',', Data, sizeof(Data) - 1)] = 0;
float Beta = atof(Data);
Data[Serial2.readBytesUntil(',', Data, sizeof(Data) - 1)] = 0;
float Gamma = atof(Data);
Data[Serial2.readBytesUntil(',', Data, sizeof(Data) - 1)] = 0;
float Px = atof(Data);
Data[Serial2.readBytesUntil(',', Data, sizeof(Data) - 1)] = 0;
float Py = atof(Data);
Data[Serial2.readBytesUntil(',', Data, sizeof(Data) - 1)] = 0;
float Pz_aux = atof(Data);
float Pz = Pz_aux + Pz_medio;
float A = DTR * Alpha;
float B = DTR * Beta;
float G = -Gamma;
```

8.5.2. Control de servomotores en Arduino.

Para el control de los servomotores se utiliza la librería *servo.h* que es capaz de transformar valores numéricos en una señal PWM con la que se accionan dichos actuadores.

Figura 8.13. Orientación de Servomotores en el robot paralelo.

Es necesario aclarar que debido a la ubicación de los servos en la plataforma se debe hacer una transformación del valor que deben girar los servos impares. Dicha transformación se hace restando 180° a la posición de cada actuador mencionado, tal como se muestra en el siguiente fragmento de código.

```
// Se escriben los valores de los ángulos a cada servo
Servo1.write(180 - Theta_i(0));
Servo2.write(Theta_i(1));
Servo3.write(180 - Theta_i(2));
Servo4.write(Theta_i(3));
Servo5.write(180 - Theta_i(4));
Servo6.write(Theta_i(5));
```

En este punto ya se tiene todo lo necesario para implementar el modelo del manipulador en la placa de desarrollo Arduino Mega 2560 desde el IDE Arduino. Tal como se muestra en la *figura 8.14 y figura 8.15.* El código completo se muestra en el Anexo 2.

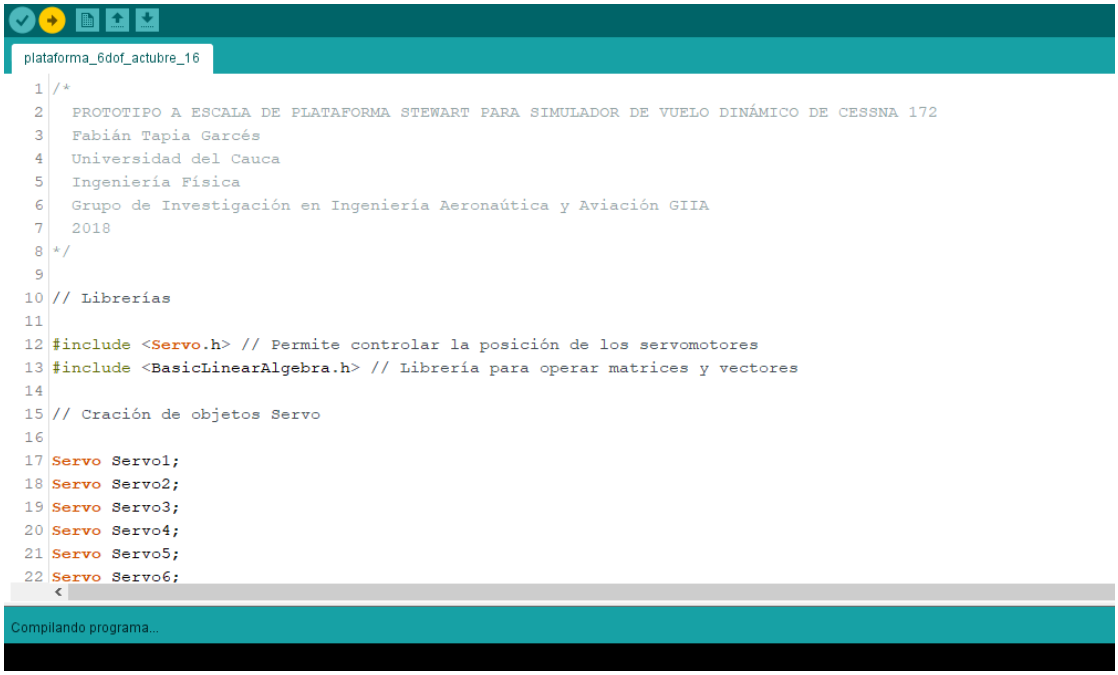


Figura 8.14: Compilado de programa en IDE Arduino

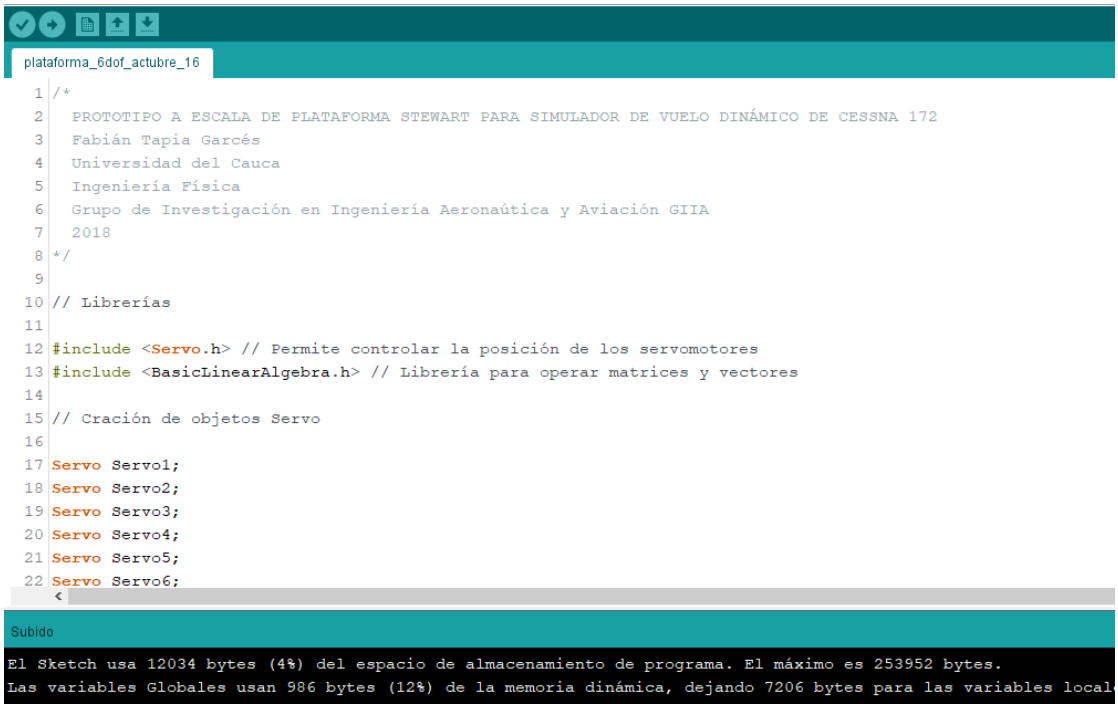
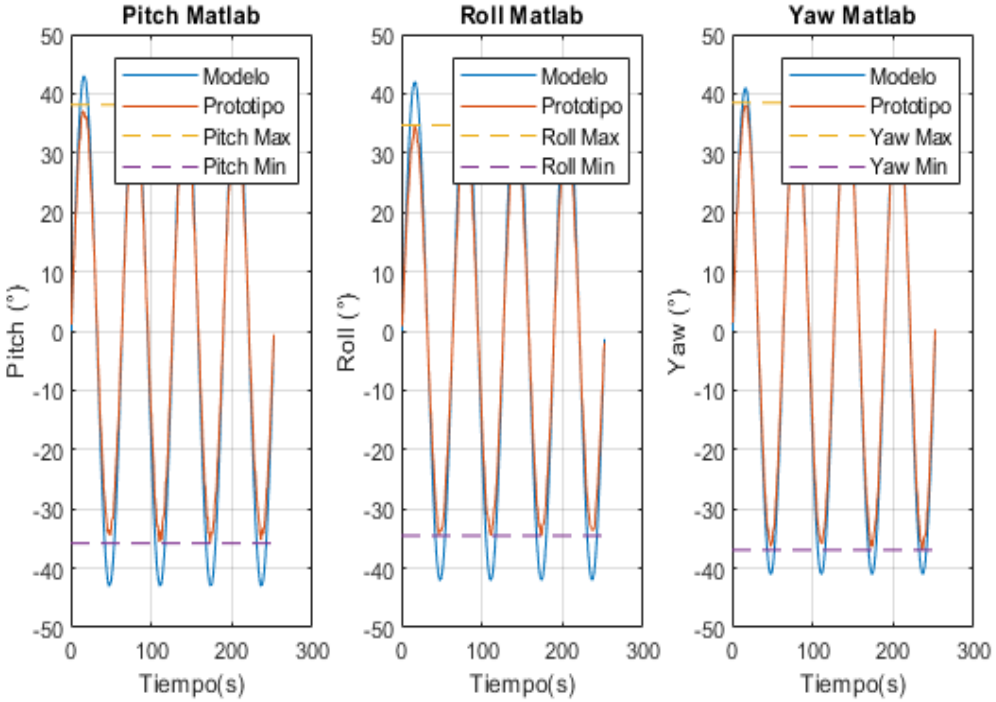


Figura 8.15: Programa compilado y subido a la tarjeta de desarrollo Arduino Mega 2560

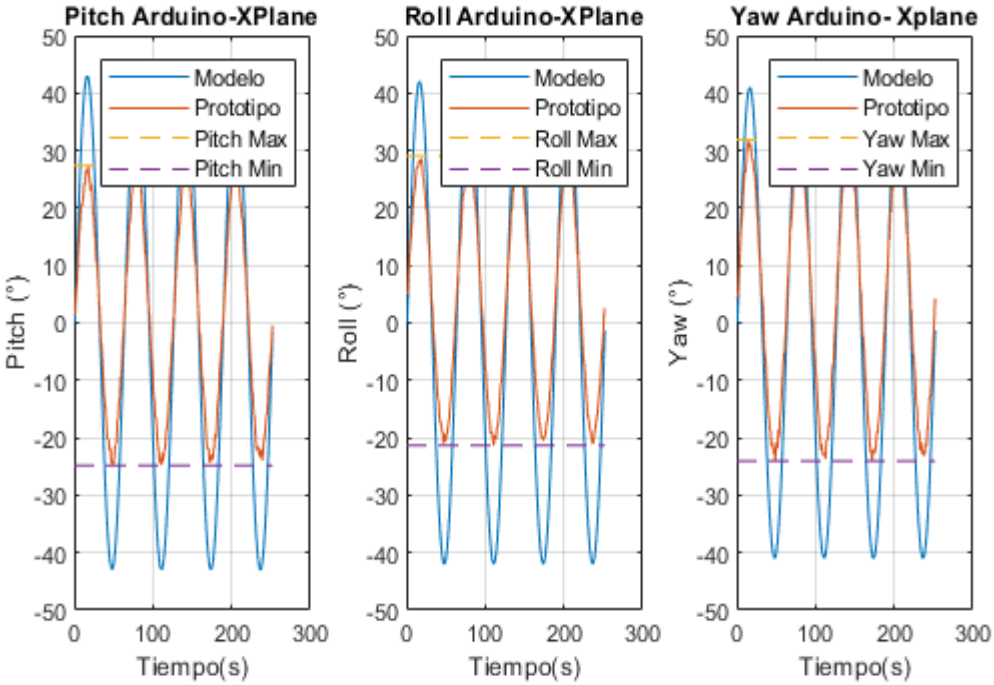
8.5.3 Caracterización en movilidad del prototipo a escala de la plataforma Stewart.

Se hace la caracterización del prototipo a escala de la plataforma Stewart, para conocer sus límites de movilidad en los ejes de orientación (pitch, roll y yaw); para esto se hace necesario hacer mediciones de los desplazamientos alrededor de los ejes X, Y y Z.
Dichas mediciones se hacen con la ayuda del sensor UM7LT, llevadas a cabo en dos etapas. En la primera etapa se hacen mediciones con el modelo cinemático codificado en Simulink, el cual permite hacer desplazamientos a lo largo de un solo DOF; en la segunda etapa se hacen mediciones con el modelo codificado en Arduino y controlado por los datos de orientación proporcionados por X Plane.

Los resultados de dichas mediciones se muestran en las gráficas 8.2 y 83. Lo que evidencian es que en las mediciones hechas con el modelo en Simulink se logran mayores rangos de desplazamiento a lo largo de cada DOF evaluado que los alcanzados en el modelo Arduino-X Plane, esto se debe principalmente a que mientras que en el modelo en Simulink se hacen los desplazamientos en un solo DOF, manteniendo valores fijos en los DOF restantes, en el modelo Arduino-X Plane no es posible aislar completamente cada DOF, debido a que todos los movimientos del aeronave tienen componentes en todos los grados de libertad, dando como resultado la disminución en el rango de movilidad de la plataforma.

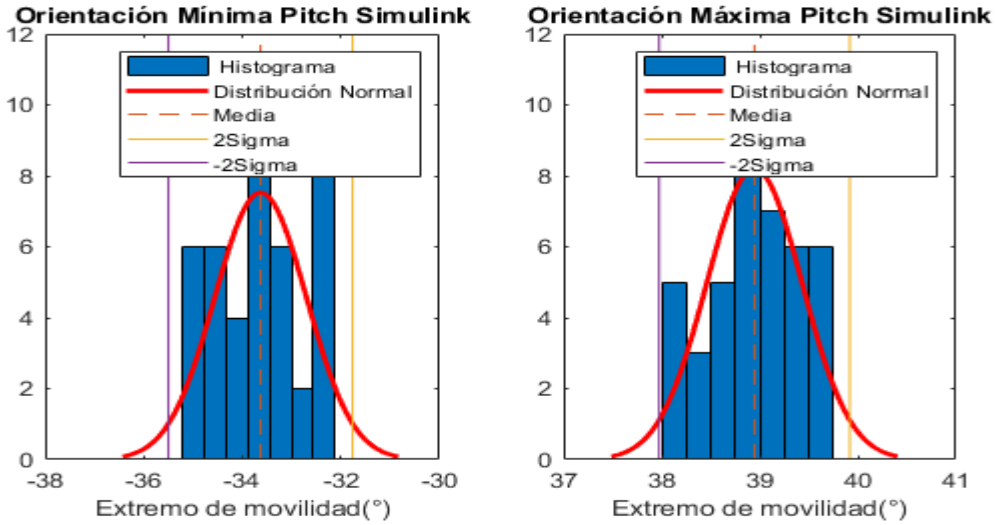


Gráfica 8.2 Diferencia de movilidad entre modelo teórico y modelo implementado en Simulink

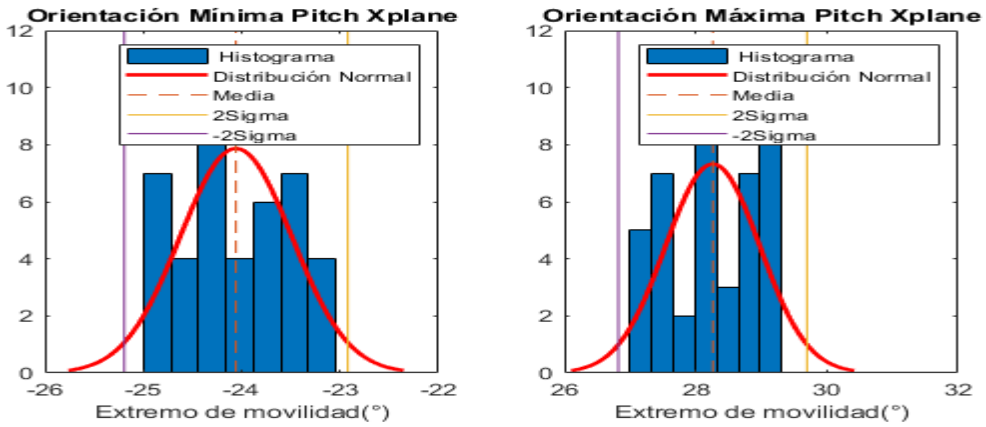


Gráfica 8.3 Diferencia de movilidad entre modelo teórico y modelo Arduino-X Plane.

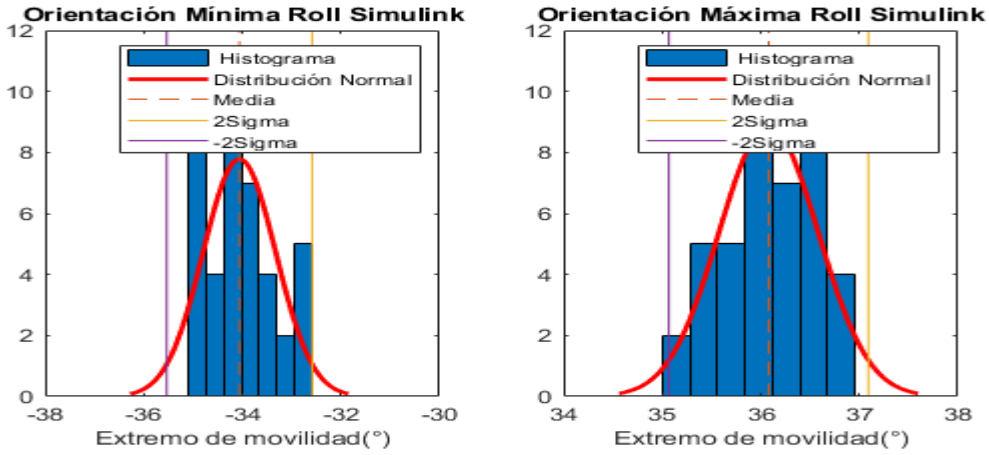
Lo siguiente por hacer es conocer los valores de los extremos de los rangos de movilidad en cada grado de libertad evaluado, para ello se toman 40 medidas suponiendo el valor teórico máximo y mínimo de cada rango.



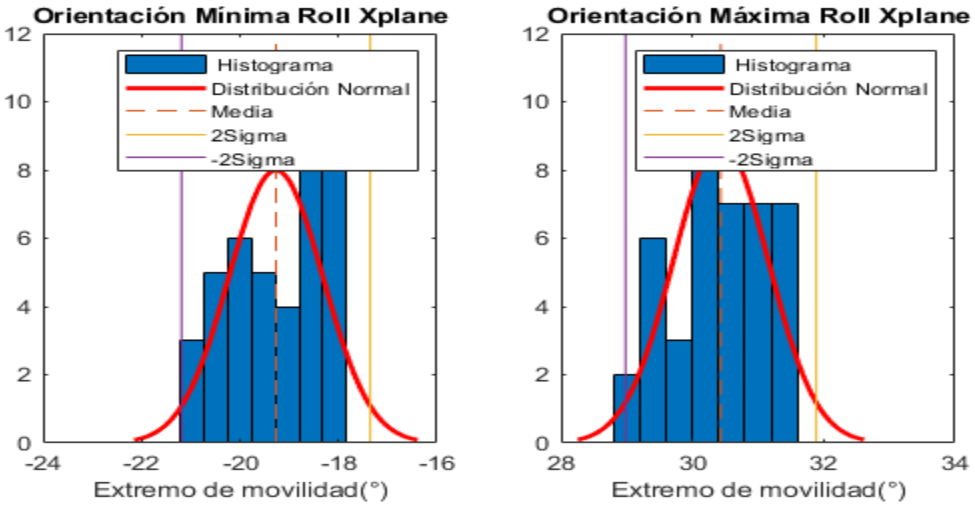
Gráfica 8.3. Pitch máximo y mínimo obtenido con modelo en Simulink.



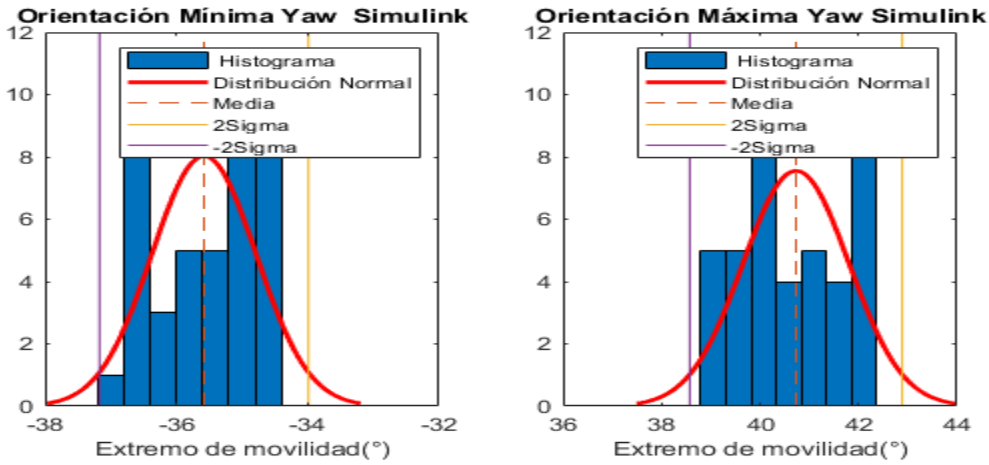
Gráfica 8.4. Pitch máximo y mínimo obtenido con modelo en X Plane 10.



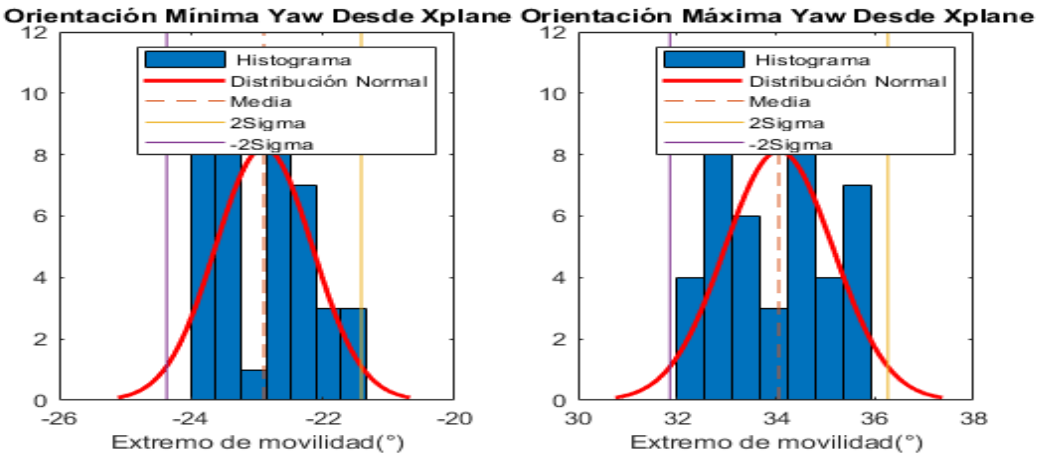
Gráfica 8.5. Roll máximo y mínimo obtenido con modelo en Simulink.



Gráfica 8.6. Roll máximo y mínimo obtenido con modelo en X Plane 10.



Gráfica 8.7. Yawmáximo y mínimo obtenido con modelo en Simulink.



Gráfica 8.8. Yaw máximo y mínimo obtenido con modelo en X Plane 10.

En las gráficas 8.3 a 8.8 se muestran los resultados de la evaluación de cada uno de los extremos de movilidad a lo largo de los ejes de orientación, obteniendo finalmente, los rangos de movilidad que se muestran en la tabla 8.10.

DOF /Rango	Teórico	Simulink		X Plane 10	
		Max	Min	Max	Min
Pitch	±43°	38.95° ± 0.48°	-33.63° ± 0.93°	28.26° ± 0.72°	-24.10° ± 0.56°
Roll	±42°	36.08° ± 0.50°	-34.06° ± 0.74°	30.43°± 0.73°	-19.29°± 0.96°
Yaw	±41°	40.73° ± 1.07°	-35.58° ± 0.79°	34.06° ± 1.10°	-22.89° ± 0.74°

Tabla 8.10 Valores máximos y mínimos en cada grado de libertad asociado a la orientación de la plataforma Stewart.

En Pitch se da una reducción del rango de movilidad en un 16.27% en el caso del modelo en Simulink y de un 39.01% en el modelo Arduino X Plane; en Roll la reducción que se encuentra es del 19.04% para el modelo en Simulink, mientras que en el modelo Arduino X Plane es del 40.75%, finalmente, se encuentra que en Yaw las reducciones en movilidad son del 9.7% y del 31.3% para el modelo en Simulink y Arduino X Plane, respectivamente.

8.6 Diseño de Cabina para simulador de vuelo dinámico.

Partiendo de lo dicho en la sección 7.8, se modelan en 3D también los elementos que deben estar presentes en la cabina del simulador, elementos como el panel de instrumentos, el timón, los pedales, la silla del piloto y los elementos de visualización.
Aunque la cabina de un Cessna 172 tiene espacio para piloto y copiloto, en este proyecto se decide diseñar una cabina con espacio únicamente para el piloto, debido a que al ser un simulador de entrenamiento primario, se puede prescindir de tener a un copiloto; además, esto implica una menor carga para los actuadores.

Para el sistema de visualización se decide utilizar tres pantallas curvas de la industria LG con referencia *fhd lps 29um68,* al usar pantallas, se hace necesario fijarlas con un soporte que debe estar sujeto a la cabina sobre la plataforma.

Para el posterior análisis de movimiento se crea también un maniquí con propiedades de masa similares a un humano, esto con el fin de tener sobre la plataforma el peso que se tendría dentro de la cabina de una aeronave Cessna 172 que es para la cual se construye.
Los modelos 3D de los elementos mencionados se encuentran en el Anexo 3.

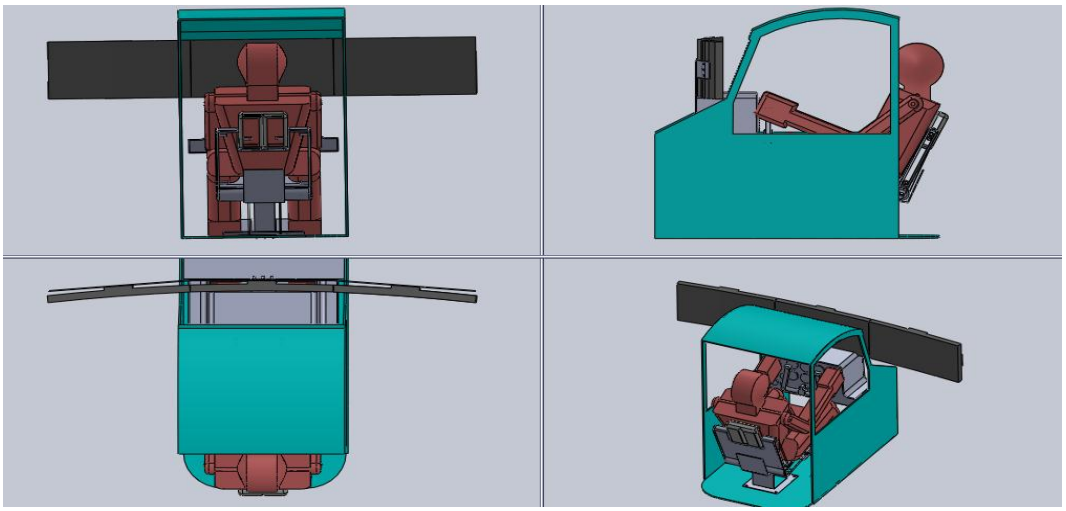


Figura 8.16.Cabina y componentes para simulador de vuelo de Cessna 172

Con todos los elementos presentes en la cabina (incluido el piloto) se encuentran las propiedades de masa del conjunto, tal como se muestra en la figura 8.17

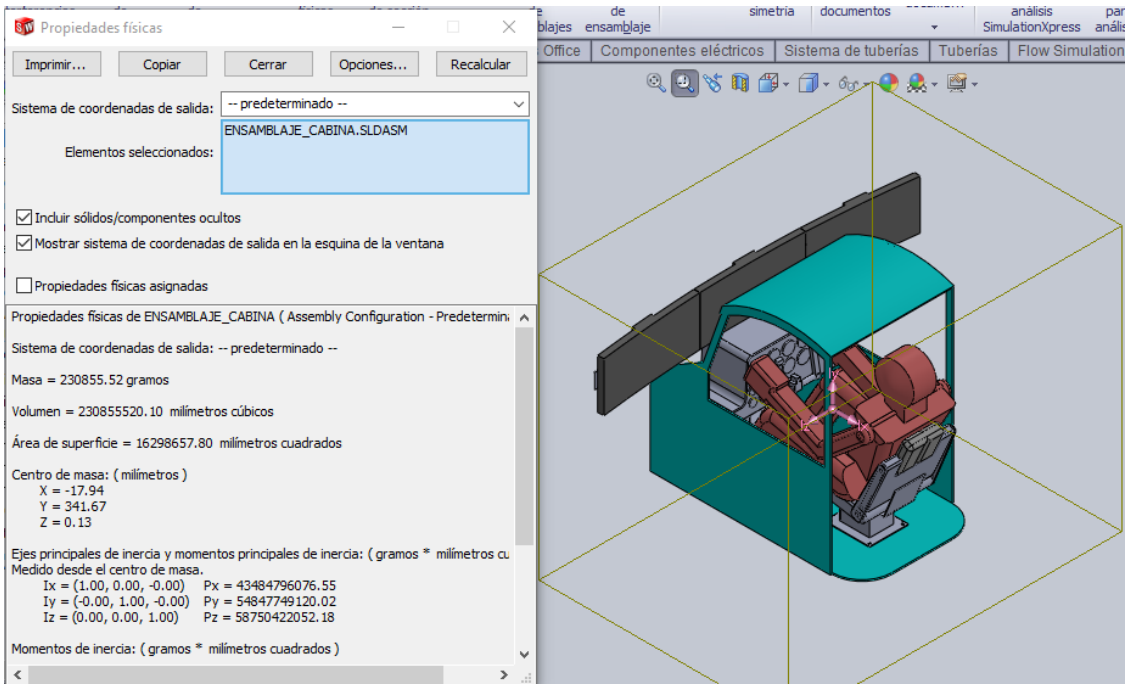


Figura 8.17. Cabina para simulador de vuelo de Cessna 172 y sus propiedades de masa.

Finalmente se unen el robot paralelo y la cabina con el piloto en una sola pieza para proceder a hacer los análisis de movimiento necesarios.

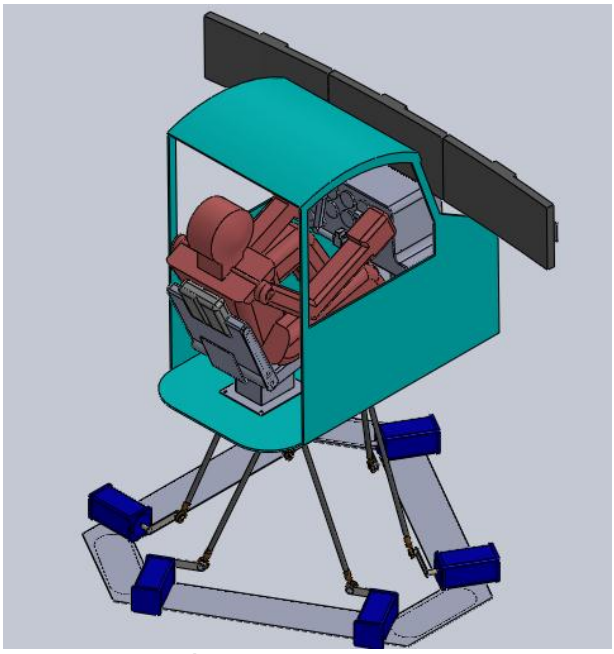


Figura 8.18. Simulador de vuelo dinámico para una aeronave de entrenamiento primario.

8.7 Análisis de movimiento de manipulador paralelo para determinar torque y potencia de los actuadores y su correcta elección.

Con el fin de determinar el torque y la potencia requerida por los actuadores para llevar la plataforma a una posición y orientación determinada se hace un análisis de movimiento con la ayuda de la herramienta Solidworks Motion, que permite crear actuadores lineales y rotativos para una pieza o una máquina. En el presente trabajo se utilizan actuadores rotativo de tipo servomotor tal y como se explicó en el capítulo 2.

Dicho estudio de movimiento se hace teniendo en cuenta el tiempo de desplazamiento de los brazos de los servomotores entre un punto y otro. Se llevan a cabo estudios a lo largo de Heave y para una combinación entre Yaw y Heave.

El primer paso para hacer el análisis mencionado es ubicar de manera correcta los actuadores rotacionales en el ensamblaje de la plataforma Stewart, en este caso, en los ejes de los Servomotores que están conectados a los brazos o palancas que unen a los servomotores con las articulaciones esféricas, tal como se muestra en la figura 8.19.

Lo siguiente por hacer es establecer la posición con respecto al tiempo de cada actuador en el robot, para poder calcular de forma óptima el torque y la potencia empleada por cada uno de ellos para llevar el robot a una posición determinada, tal procedimiento se hace en el constructor de funciones para motores incluido en Soliworks Motion mostrado en la figura 8.20 y que permite además de establecer la posición con respecto al tiempo de un actuador, conocer la velocidad y aceleración a la que se somete, en este caso, el eje de giro del actuador en estudio.

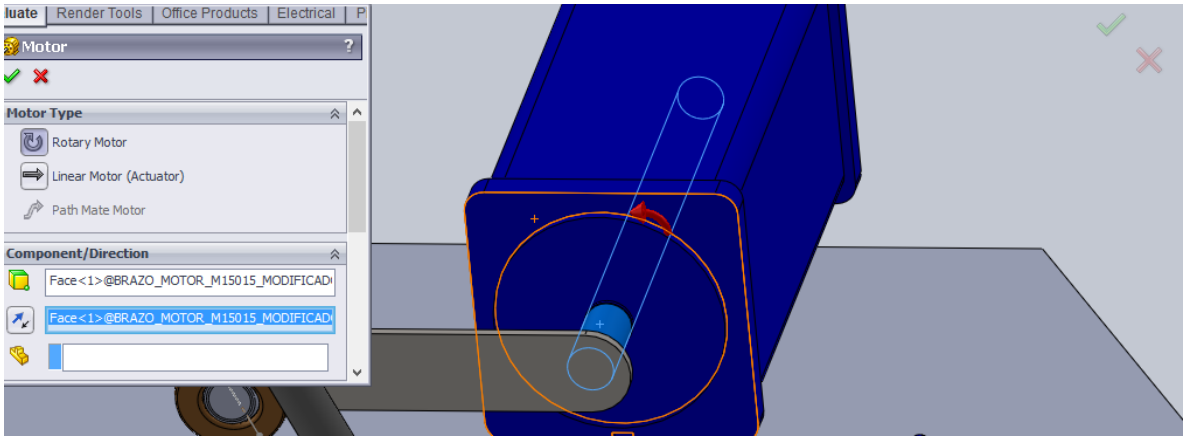


Figura 8.19. Ubicación de Actuador rotacional en modelo 3D de plataforma Stewart.

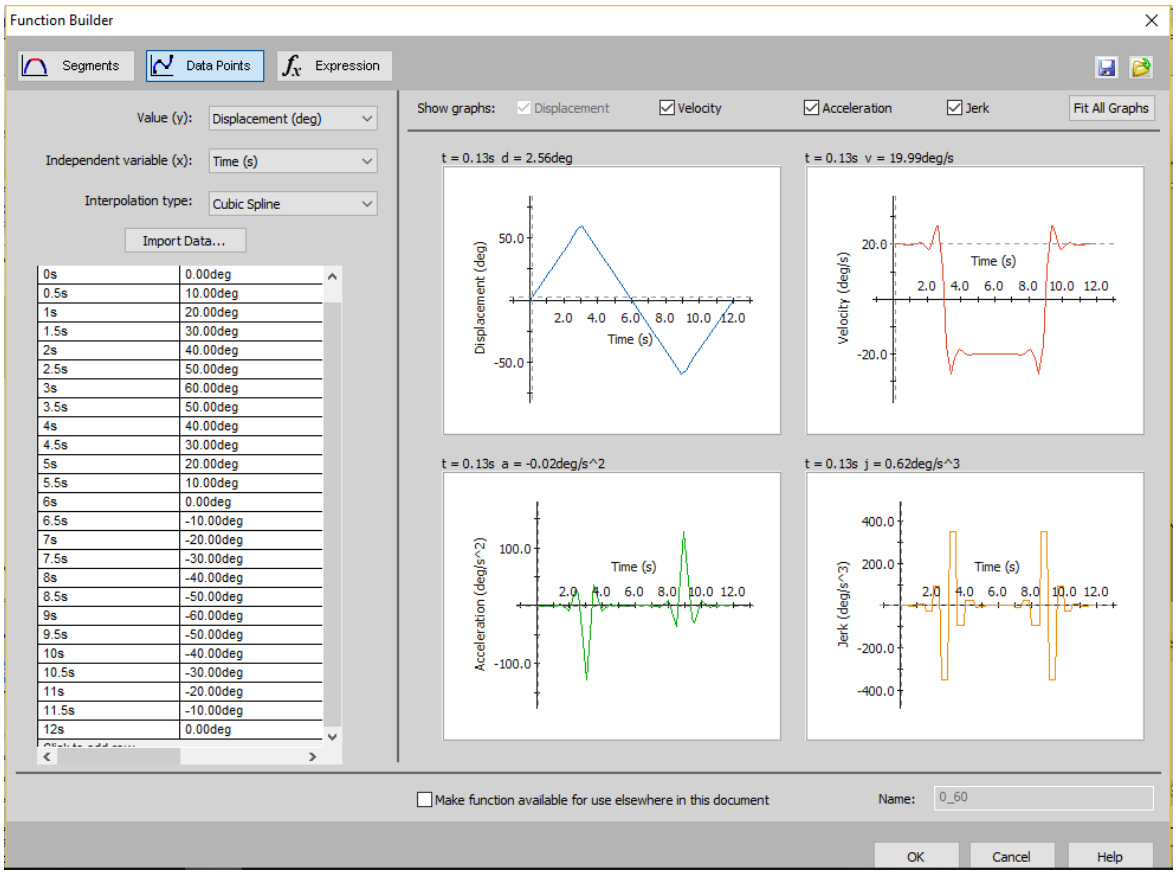


Figura 8.20 Interfaz de constructor de funciones para análisis de actuadores en Solidworks.

Los puntos con los que se hace el análisis se toman de las matrices de ángulos obtenidas de la evaluación del modelo cinemático descrito en la sección 8.1.

Tiempo(s)	Posición (°)	Tiempo(s)	Posición (°)	Tiempo(s)	Posición (°)
0,0	55,8	2,6	103,7	5,2	132,3
0,2	65,3	2,8	105,9	5,4	134,7
0,4	71,2	3,0	108,1	5,6	137,2
0,6	75,8	3,2	110,3	5,8	139,7
0,8	79,7	3,4	112,4	6,0	142,3
1,0	83,1	3,6	114,6	6,2	145,1
1,2	86,2	3,8	116,7	6,4	147,9
1,4	89,1	4,0	118,9	6,6	151,0
1,6	91,8	4,2	121,0	6,8	154,2
1,8	94,4	4,4	123,2	7,0	157,7
2,0	96,8	4,6	125,4	7,2	161,5
2,2	99,2	4,8	127,7	7,4	165,7

Tabla 8.11 Valores de desplazamiento de actuadores para cálculo de torque y potencia.

En las siguientes gráficas se muestran los resultados de los análisis de movimiento mencionados anteriormente.

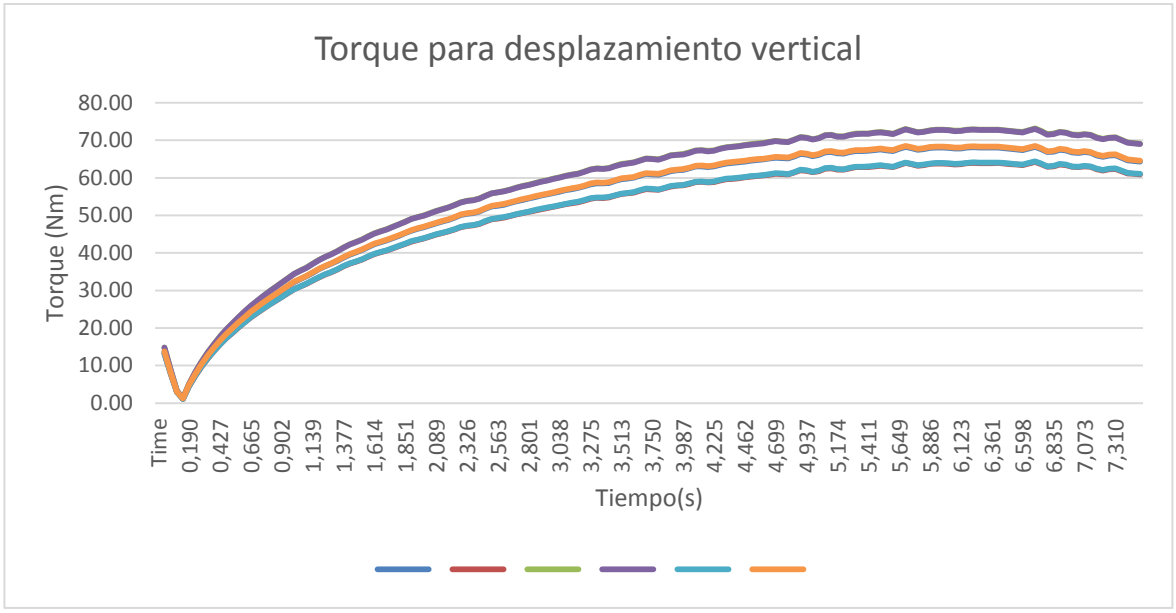


Grafico 8.9 Torque en los actuadores para generar el movimiento a lo largo del eje z

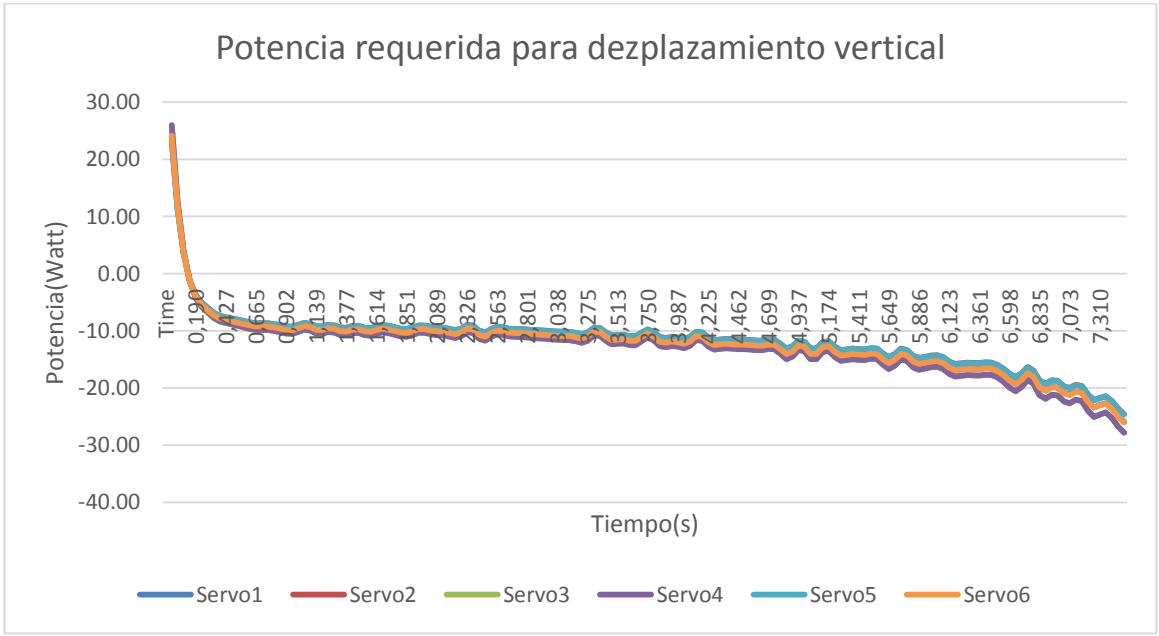


Grafico 8.10 Potencia requerida en los actuadores para generar el movimiento a lo largo del eje z.

Los gráficos 8.9 y 8.10 como se menciona, muestran de forma clara el torque y la potencia requerida por los actuadores para generar el movimiento a lo largo del

eje z, el cual se conoce como “Heave”. Se deduce fácilmente que se necesitan Servomotores de alto torque; la potencia necesaria para generar el movimiento es baja.

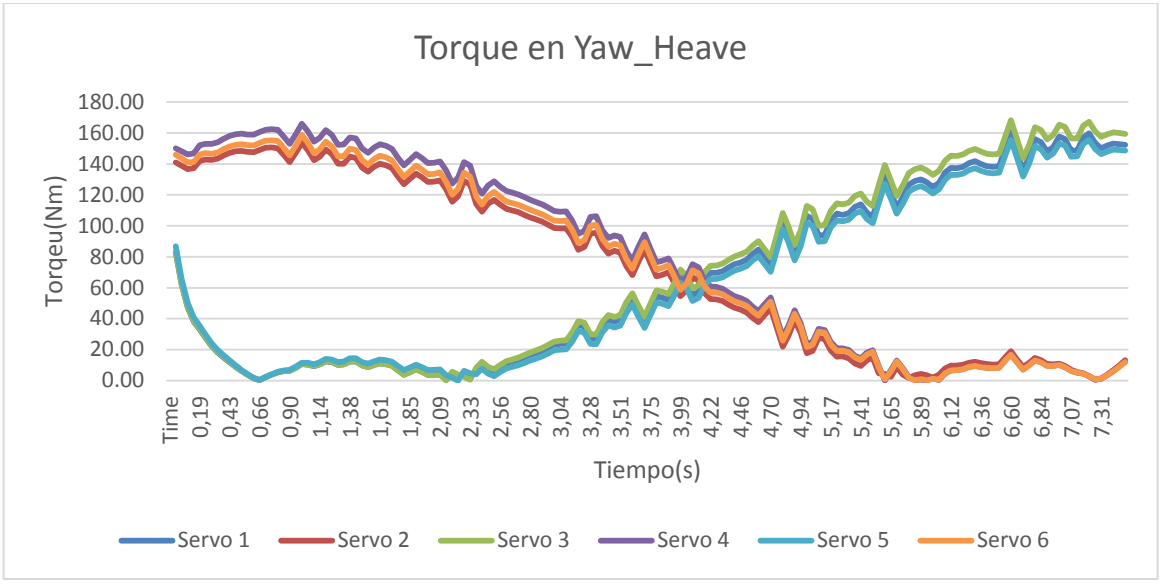


Grafico 8.11 Torque en los actuadores para generar el movimiento compuesto a lo largo de z y alrededor de ese eje.

El Grafico 8.11 muestra el torque que se necesita para generar el movimiento compuesto descrito anteriormente. Aunque se muestran valores de torque mucho más elevados que los obtenidos en el análisis del movimiento a lo largo del eje z, sólo se tiene en cuenta el torque encerrado en el recuadro rojo de la figura 8.12.

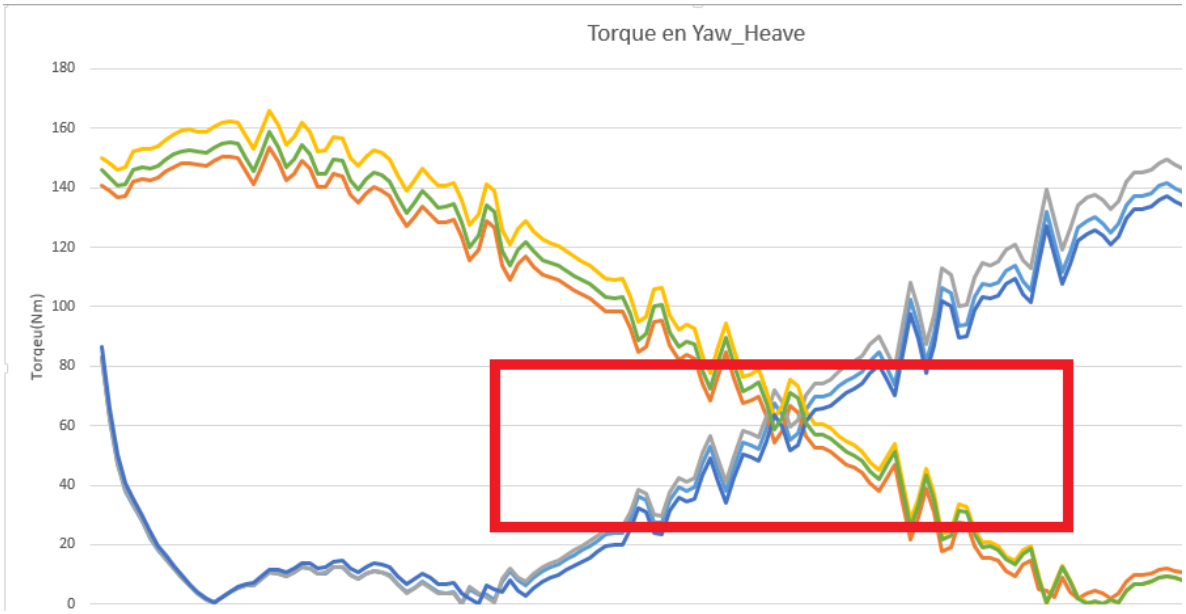


Figura 8.12 Torque necesario para generar movimiento compuesto Yaw – Heave

Lo anterior se asegura debido a que el movimiento de la plataforma en Yaw no se presenta en todo el rango de valores obtenido en el modelo, si no que se presenta en pequeñas oscilaciones alrededor del eje z.

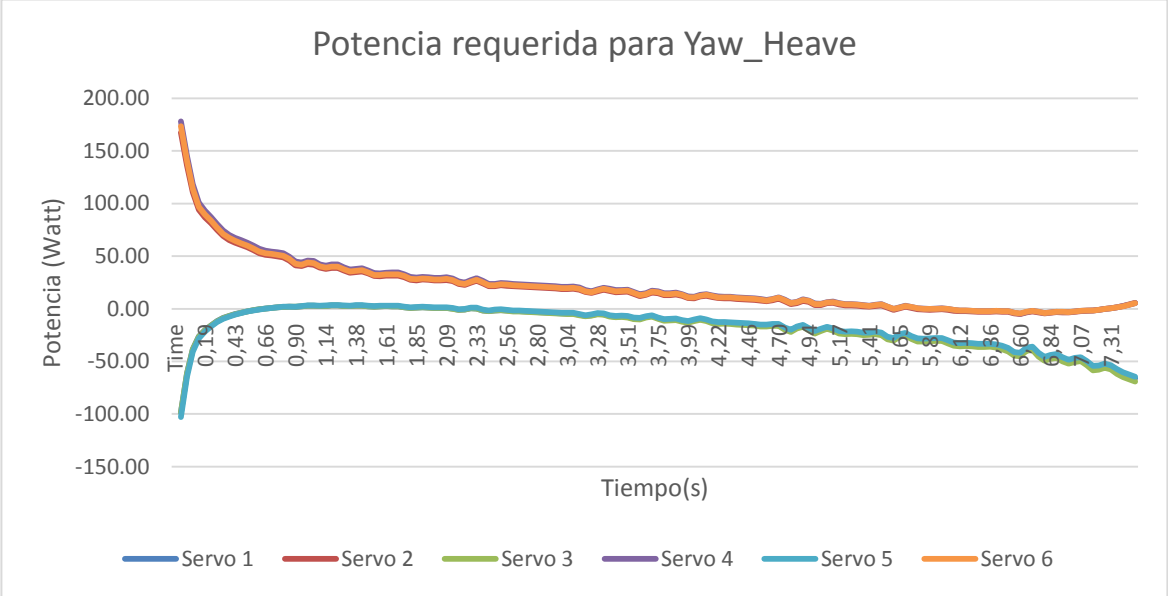


Grafico 8.12 Potencia requerida en los actuadores para generar el movimiento compuesto a lo largo de z y alrededor de ese eje.

El gráfico 8,10 al igual que el gráfico 8.12 muestran un bajo consumo de potencia por parte de los actuadores; por lo tanto el parámetro dominante para la elección de los servomotores que se deben usar para generar los movimientos de la plataforma de forma eficiente es en este caso el torque que sea capaz de producir y mantener cada actuador.

A partir de los datos obtenidos en los análisis se obtiene que el valor máximo de torque ejercido por cada actuador está alrededor de de 80 N-m cuando se producen un movimiento compuesto. Conociendo esta información de torque se determina el tipo de actuador que es necesario utilizar, uno de los actuadores que cumple con ésta especificación de torque es el ESM18E-552-154 de la compañía ABB

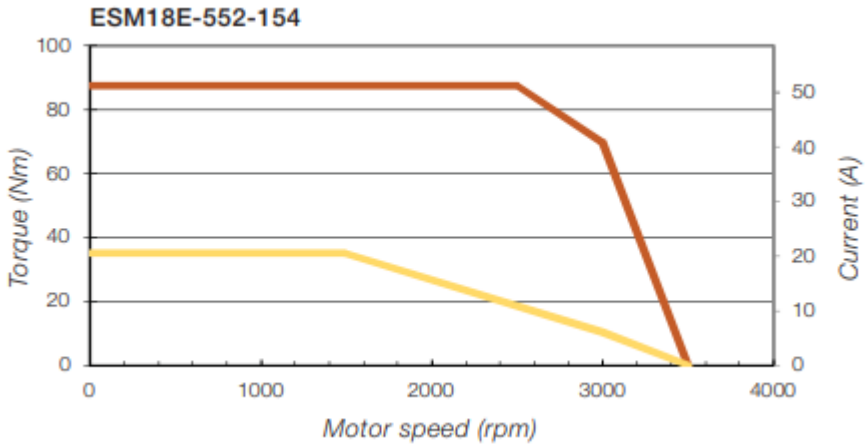


Grafico 8.13 Características del actuador ESM18E-552-154.

8.8 Presupuesto

En esta parte del documento se presenta el presupuesto utilizado en el diseño y construcción del prototipo del manipulador y los elementos necesarios para poder hacer pruebas en el Software de simulación, Luego se hace el presupuesto necesario para construir el simulador de vuelo dinámico a escala 1:1.

8.8.1 Costos prototipo.

A continuación se presenta una tabla de costos necesarios construir el prototipo de pruebas del simulador de vuelo dinámico, es necesario aclarar que en esta tabla no se incluyen costos de desarrollo, sólo se tienen en cuenta costos de diseño 3D, elaboración de piezas y el ensamblaje del prototipo.

Elemento	Cantidad	Valor Unitario($)	Valor Total ($)
X Plane 10 Demo	1	160000	160000
Micro Servo SG90 Tower Pro	6	7990	47940
Kit Ejes roscados	3	20500	61500
Conector largo con eje E2	6	1000	6000
Tornillos M3x30mm	18	200	3600
Pack Jumpers M m	1	6700	6700
Pack Jumpers H h	1	6700	6700
Pack Jumpers H m	1	6700	6700
módulo MDF 13mm	1	8000	8000
Arduino Mega 2560 R3	2	120080	240160

Sensor UM7 LT	1	436084	436084
Joystick Genius Maxfighter F17	1	70000	70000
Fuente de poder 12V, 20A, 240W	1	97000	97000
Protoboard Wish WB 104	1	60000	60000
Total Materiales			1210384
Mano de obra en elaboración de piezas y ensamblado (horas)	30	25000	750000
TOTAL			1960384

Tabla 8.12. Costos de fabricación de simulador de vuelo en fase prototipo.

8.8.2 Presupuesto Simulador de vuelo Dinámico para Cessna 172

EL presupuesto que se muestra a continuación es para construir el simulador de vuelo dinámico, teniendo en cuenta igual que para el prototipo solo el costo de materiales y mano de obra utilizada para la fabricación. Todos los valores están dados en pesos.

Elemento	Cantidad	Valor Unitario	Valor Total
X Plane 10 Professional Use	1	2337000	2337000
Servo SME 18E 552-154 + Driver	6	3403990	20423940
Saitek Pro Flight Yoke System	1	1182000	1182000
Saitek PRO Flight Switch Panel (PZ55)	1	285000	285000
Saitek Pro Flight Rudder Pedals	1	716000	716000
Silla Piloto Wise 8 Wd015	1	865000	865000
Pc Gamer I7 8700 Gtx 1050 Ti 4 Gb Hdd 2 Tb Ram 8 Gb	1	3150000	3150000
Tarjeta gráfica Triplehead2go T2gdpmif Dp Edition	1	1446900	1446900
Base para 3 Monitores de 15" a 27"	1	246000	246000
Arduino Mega 2560 R3	2	120080	240160
Cabina en fibre de vidrio	1	2600000	2600000
Articulaciones SI 20ES SFK	12	133760	1605120
Barras de acero para links de articulaciones	6	84900	509400
Placas de acero 4" x 1m para palancas de servo	3	60000	180000
Ángulo 6 metros 1/8 x 1-1/2 pulgada g - 50 para Plataforma Móvil	1	30900	30900
Mano de obra	50 horas	50000	2500000
Total			38317420

Tabla 8.13. Costos de fabricación de simulador de vuelo en escala 1:1.

CONCLUSIONES

Se obtienen de forma eficiente los datos de orientación y posición del aeronave para entrenamiento primario Cessna 172 proporcionados por el software de simulación X plane 10 haciendo uso de Datarefs los cuales se envían a través de un puerto serial.

Se diseña un robot paralelo tipo Plataforma Stewart para ser usado como sistema generador de movimiento del Simulador de vuelo propuesto.

Se diseña y se construye de forma virtual el simulador de vuelo dinámico propuesto, con el fin de llevar a cabo análisis de movimiento que permiten determinar el torque requerido por los actuadores para poder generar los movimientos exigidos por el simulador de vuelo.

Se construye un prototipo a escala de la plataforma Stewart con el fin de encontrar los rangos de movilidad del robot obteniendo los rangos de (-33.63° ± 0.93°, 38.95° ± 0.48°), (-34.06° ± 0.74°, 36.08° ± 0.50°), (-35.58° ± 0.79°, 40.73° ± 1.07°) en los ejes de orientación Pitch, Roll y Yaw respectivamente al ser probado con el modelo cinemático implementado en Simulink

Se implementa el modelo cinemático del robot codificado en Arduino y controlado con los datos de orientación y posición obtenidos de X Plane para determinar la movilidad en los ejes de orientación, obteniendo como resultados los rangos de movilidad (-24.10° ± 0.56°, 28.26° ± 0.72°), (-19.29°± 0.96°, 30.43°± 0.73°), (-22.89° ± 0.74°, 34.06° ± 1.10°) en los ejes de orientación Pitch, Roll y Yaw respectivamente.

RECOMENDACIONES

Para trabajos próximos orientados al desarrollo de simuladores de vuelo dinámicos se recomienda hacer un análisis de variación de parámetros del robot paralelo para poder crear modelos que permitan un mayor rango de desplazamiento a lo largo de los ejes x, y y z para lograr los requerimientos de diseño propuestos por Allerton.

Debido al alto torque que requeridos por los actuadores se recomienda partiendo del análisis propuesto anteriormente obtener parámetros correspondientes al manipulador paralelo que permitan reducir el torque de los actuadores.

Para poder llegar a una certificación del simulador por parte de la FAA se hace necesario utilizar un microprocesador que esté aceptado por dicha entidad, se recomienda trabajar con el microcontrolador de Texas Instruments Delfino TMS320F28377D, Hércules TMS570 o FPGAs tipo SoC (Intel, Silinx)

BIBLIOGAFÍA

1. Pr. Maroto J. Conferencia: “La experiencia con simuladores en el caso de inmersión total” UPM. Responsable del área de visual del CITEF. Centro de investigación en Tecnologías, 2013.
2. Strachan A. Seminario - Uso de simuladores en procesos de enseñanza-aprendizaje. EAFIT 2014
3. https://www.proflight.com/en/full-flight-simulatoren/historie.php
4. http://www.hispaviacion.es simulación-de-vuelo-un-poco-de-historia
5. http://jasaaviation.blogspot.com/2014/03/simuladores-de-vuelo.html
6. RAC 24. *Dispositivos Simuladores para Entrenadores de Vuelo*. Reglamentos Aeronáuticos de Colombia. Unidad Administrativa Especial de Aeronáutica Civil (2012).
7. Cardona, S. Clos, D. (2001), *Teoría de máquinas.* Barcelona, España, Edicions UPC.
8. Ollero, A. (2001). *ROBÓTICA, Manipuladores y robots móviles.* Barcelona, España. Marcombo.
9. X. Gao, D. Lei, Q. Liao, and G. Zhang, (2005). “*Generalized Stewart-Gough platforms and their direct kinematics*”, IEEE Transactions on Robotics, vol. 21, no. 2, pp. 141–51.
10. D. Stewart. (1965). *A platform with six degrees of freedom” Proc. Inst. Mech*. Eng., vol. 180, no. 1, pp. 371–386.
11. Szufnarowski, F. (2013). *Germany Stewart platform with fixed rotary actuators: a low cost design study*. Faculty of Technology, Bielefeld University, Tomadode: https://www.techfak.uni-bielefeld.de/~fszufnar/publications/Szufnarowski2013.pdf
12. Barrientos, A. Peñín, L. (1997). *Fundamentos de robótica.* Universidad Politécnica de Madrid, Madrid, McGraw-Hill.
13. Craig John. (1986). *Introduction to Robotics:* Mechanics & Control. Boston, Addison. Wesley Publishing Company.
14. Spong Mark W. & Vidyanagar M. *Robot and Control.* New York, John. Wiley & Sons, 1989.
15. Lung-Wen Tsai. *Robot Analysis: The Mechanics of Serial and Parallel Manipulators.* Maryland, John Wiley & Sons, Inc, 1999.
16. Cisneros R. *Modelo matemático de un robot paralelo de seis grados de libertad*. Universidad de Las Américas, Puebla, 2006
17. https://www.emavi.edu.co/aviones-de-entrenamiento-1
18. http://avia-es.com/blog/cessna-172-skyhawk-tehnicheskie-harakteristiki-foto
19. https://x-plane.com/manuals/desktop/10/index.html
20. https://developer.x-plane.com/manuals/planemaker/

21. http://xplanewiki.fr/386
22. http://www.xsquawkbox.net/xpsdk/docs/DataRefs.html
23. http://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf
24. https://www.solidworks.com/es/
25. Allerton David. (2009). *Principles of fight simulation.* John Wiley & Sons, Ltd
26. Roskam, J. (2002). *Airplane Desing. Part III: Layout Desing of Cockpit, Fuselage, Wing and Empennage: Cutaways and Inboard Profiles.* Kansas, USA, DARcorporation.

ANEXOS

ANEXO 1: Funciones creadas para evaluación del modelo cinemático del manipulador paralelo en Matlab.

FUNCIÓN *Mateval:*
La función *Mateval* es la encargada de generar una matriz de posibles puntos en los que se evalúa el modelo cinemático de la plataforma Stewart.

Se generan seis matrices con la ayuda de esta función, cada matriz moviéndose en un grado de libertad independiente, esto con el fin de encontrar los valores máximos y mínimos de orientación y/o desplazamiento a lo largo y/o alrededor de cada uno de ellos.
A continuación se muestra la implementación de la función.

```
function Mpoint = Mateval(A,B,G,Px,Py,Pz)

% Genera una matriz de puntos para evaluar el modelo cinemático y
encontrar el valor de los ángulos de los actuadores.

 AL = length(A);
 BL = length(B);
 GL = length(G);
 PxL = length(Px);
 PyL = length(Py);
 PzL = length(Pz);

Mpoint = [];

for k=1:AL
   for l=1:BL
       for m=1:GL
           for n=1:PxL
               for o=1:PyL
                   for r=1:PzL
                     Ma = [A(k) B(l) G(m) Px(n) Py(o) Pz(r)];
                     Mpoint = [Mpoint;Ma];
                   end
               end
           end
       end
    end
 end

end % fin de la función
```

FUNCIÓN *MValServo:*

La función *MValServo* está creada con base en el modelo cinemático de la plataforma, esta recibe los puntos de la matriz generada por *Mateval* para encontrar todos los puntos en los que los valores de giro de los servomotores son válidos; que como se explicó en el capítulo 2, sólo son válidos valores reales de ángulos y además; por la naturaleza de los actuadores, estos valores deben estar en el rango de 0° a 180°.

```
function M_AngleServo = MValServo(a,b,c,d,p,l,S,Mpoint)


%  Constantes de Plataforma  transformada
 Acte= (a/sqrt(3));
 Ccte= (c/sqrt(3));

% % POSICIONES DE LOS VÉRTICES DEL HEXÁGONO DE LA BASE
% % Todos estos vectores deben estar transpuestos;
 crobot = sqrt(3)/6;
 Base_1 = [ crobot*(2*b + d) 0.5*d 0];
 Base_2 = [ -crobot*(b - d) 0.5*(b + d) 0];
 Base_3 = [ -crobot*(b + 2*d) 0.5*b 0];
 Base_4 = [ -crobot*(b + 2*d) -0.5*b 0];
 Base_5 = [ -crobot*(b - d) -0.5*(b + d) 0];
 Base_6 = [ crobot*(2*b + d) -0.5*d 0];

 Base = [ Base_1; Base_2; Base_3; Base_4; Base_5; Base_6 ];
% Base Traspuesta.

 Org = Base(1:6,[1,2]); % Matriz que define el origen de la plataforma.

Maux = [];
Mfin = [];
M_AngleServo = [];
r = length(Mpoint);

 for t=1:r
     for u=1:6
     T1_X = Acte*( cosd(Mpoint(t,2))*cosd(Mpoint(t,3) + 60) +
sind(Mpoint(t,1))*sind(Mpoint(t,2))*sind(Mpoint(t,3) +60) ) + Ccte*(
cosd(Mpoint(t,2))*cosd(Mpoint(t,3)) +
sind(Mpoint(t,1))*sind(Mpoint(t,2))*sind(Mpoint(t,3)) ) + Mpoint(t,4);
     T1_Y = Acte*cosd(Mpoint(t,1))*sind(Mpoint(t,3) +60) +
Ccte*cosd(Mpoint(t,1))*sind(Mpoint(t,3)) + Mpoint(t,5);
     T1_Z = Acte*( sind(Mpoint(t,1))*cosd(Mpoint(t,2))*sind(Mpoint(t,3) +
60) - sind(Mpoint(t,2))*cosd(Mpoint(t,3) + 60) ) + Ccte*(
sind(Mpoint(t,1))*cosd(Mpoint(t,2))*sind(Mpoint(t,3)) -
sind(Mpoint(t,2))*cosd(Mpoint(t,3)) ) + Mpoint(t,6);

     T2_X = Acte*( cosd(Mpoint(t,2))*cosd(Mpoint(t,3) +60) +
sind(Mpoint(t,1))*sind(Mpoint(t,2))*sind(Mpoint(t,3) + 60 ) ) + Ccte*( -
```

```
cosd(Mpoint(t,2))*sind(Mpoint(t,3) +30) +
sind(Mpoint(t,1))*sind(Mpoint(t,2))*cosd(Mpoint(t,3) +30) ) +
Mpoint(t,4);
     T2_Y = Acte*cosd(Mpoint(t,1))*sind(Mpoint(t,3) + 60) +
Ccte*cosd(Mpoint(t,3) + 30) + Mpoint(t,5);
     T2_Z = Acte*( -sind(Mpoint(t,2))*cosd(Mpoint(t,3) + 60) +
sind(Mpoint(t,1))*cosd(Mpoint(t,2))*sind(Mpoint(t,3) + 60) ) + Ccte*(
sind(Mpoint(t,2))*sind(Mpoint(t,3) +30) +
sind(Mpoint(t,1))*cosd(Mpoint(t,2))*cosd(Mpoint(t,3) + 30) ) +
Mpoint(t,6);

     T3_X = -Acte*( cosd(Mpoint(t,2))*cosd(Mpoint(t,3)) +
sind(Mpoint(t,1))*sind(Mpoint(t,2))*sind(Mpoint(t,3)) ) + Ccte*(
sind(Mpoint(t,1))*sind(Mpoint(t,2))*cosd(Mpoint(t,3) + 30) -
cosd(Mpoint(t,2))*sind(Mpoint(t,3) + 30) ) + Mpoint(t,4);
     T3_Y = -Acte*cosd(Mpoint(t,1))*sind(Mpoint(t,3)) +
Ccte*cosd(Mpoint(t,1))*cosd(Mpoint(t,3) + 30) + Mpoint(t,5);
     T3_Z = -Acte*( sind(Mpoint(t,1))*cosd(Mpoint(t,2))*sind(Mpoint(t,3))
- sind(Mpoint(t,2))*cosd(Mpoint(t,3)) ) + Ccte*(
sind(Mpoint(t,2))*sind(Mpoint(t,3) + 30) +
sind(Mpoint(t,1))*cosd(Mpoint(t,2))*cosd(Mpoint(t,3) + 30) ) +
Mpoint(t,6);

     T4_X = -Acte*( cosd(Mpoint(t,2))*cosd(Mpoint(t,3)) +
sind(Mpoint(t,1))*sind(Mpoint(t,2))*sind(Mpoint(t,3)) ) + Ccte*( -
cosd(Mpoint(t,2))*cosd(Mpoint(t,3) + 60) -
sind(Mpoint(t,1))*sind(Mpoint(t,2))*sind(Mpoint(t,3) + 60) ) +
Mpoint(t,4);
     T4_Y = -Acte*cosd(Mpoint(t,1))*sind(Mpoint(t,3)) + Ccte*(-
cosd(Mpoint(t,1))*sind(Mpoint(t,3) + 60) )+ Mpoint(t,5);
     T4_Z = -Acte*(-sind(Mpoint(t,2))*cosd(Mpoint(t,3)) +
sind(Mpoint(t,1))*cosd(Mpoint(t,2))*sind(Mpoint(t,3))) + Ccte*(
sind(Mpoint(t,2))*cosd(Mpoint(t,3) + 60) -
sind(Mpoint(t,1))*cosd(Mpoint(t,2))*sind(Mpoint(t,3) + 60) ) +
Mpoint(t,6);

     T5_X = Acte*( cosd(Mpoint(t,2))*sind(Mpoint(t,3) + 30) -
sind(Mpoint(t,1))*sind(Mpoint(t,2))*cosd(Mpoint(t,3) + 30) ) + Ccte*(-
cosd(Mpoint(t,2))*cosd(Mpoint(t,3) + 60) -
sind(Mpoint(t,1))*sind(Mpoint(t,2))*sind(Mpoint(t,3) + 60) ) +
Mpoint(t,4);
     T5_Y = Acte*(-cosd(Mpoint(t,1))*cosd(Mpoint(t,3) + 30)) + Ccte*(-
cosd(Mpoint(t,1))*sind(Mpoint(t,3) + 60)) + Mpoint(t,5);
     T5_Z = Acte*(-sind(Mpoint(t,2))*sind(Mpoint(t,3) + 30) -
sind(Mpoint(t,1))*cosd(Mpoint(t,2))*cosd(Mpoint(t,3) + 30) ) + Ccte*(
sind(Mpoint(t,2))*cosd(Mpoint(t,3) + 60) -
sind(Mpoint(t,1))*cosd(Mpoint(t,2))*sind(Mpoint(t,3) +60) ) +
Mpoint(t,6);

     T6_X = Acte*( cosd(Mpoint(t,2))*sind(Mpoint(t,3) + 30) -
sind(Mpoint(t,1))*sind(Mpoint(t,2))*cosd(Mpoint(t,3) +30) ) + Ccte*(
cosd(Mpoint(t,2))*cosd(Mpoint(t,3)) +
sind(Mpoint(t,1))*sind(Mpoint(t,2))*sind(Mpoint(t,3)) ) + Mpoint(t,4);
```

```
     T6_Y = Acte*( -cosd(Mpoint(t,1))*cosd(Mpoint(t,3) + 30)) +
Ccte*cosd(Mpoint(t,1))*sind(Mpoint(t,3)) + Mpoint(t,5);
     T6_Z = Acte*( -sind(Mpoint(t,2))*sind(Mpoint(t,3) + 30) -
sind(Mpoint(t,1))*cosd(Mpoint(t,2))*cosd(Mpoint(t,3) + 30) ) +Ccte*( -
sind(Mpoint(t,2))*cosd(Mpoint(t,3)) +
sind(Mpoint(t,1))*cosd(Mpoint(t,2))*sind(Mpoint(t,3)) ) + Mpoint(t,6);

     T = [T1_X T1_Y T1_Z; T2_X T2_Y T2_Z; T3_X T3_Y T3_Z; T4_X T4_Y T4_Z;
T5_X T5_Y T5_Z; T6_X T6_Y T6_Z ];

         for k = 1:6
             A_i(k) = 2*p*(sind( S(k) )*( T(k,1) - Org(k,1) ) - cosd(
S(k))*( T(k,2) - Org(k,2) ));
             B_i(k) = 2*p*( T(k,3) );
             C_i(k) = l^2 - p^2 - (T(k,1) - Org(k,1))^2 - (T(k,2) -
Org(k,2))^2 - (T(k,3))^2;
         end
         CTES = [A_i; B_i; C_i];
     end
             for n=1:6
              TETHA_i(n)= acosd(C_i(n)/(sqrt( (A_i(n))^2 + (B_i(n))^2)) )
+ atand( (A_i(n))^2 / (B_i(n))^2 );
             end
             TETHA_i;
             Maux= [Maux;TETHA_i];
 end

 M_fin = [Mpoint Maux];

 for k =1:r
          if (imag(M_fin(k,:)) == 0)
               if ( M_fin(k,7)<= 180 && M_fin(k,8)<= 180 && M_fin(k,9)<=
180 && M_fin(k,10)<= 180 && M_fin(k,11)<= 180 && M_fin(k,12)<= 180)
                M_AngleServo = [M_AngleServo; M_fin(k,:)];
               end
          end
 end


 end % fin de la función
```

ANEXO 2: Modelo cinemático del manipulador paralelo codificado en el IDE Arduino.

```
/* PROTOTIPO A ESCALA DE PLATAFORMA STEWART PARA SIMULADOR DE VUELO
DINÁMICO DE CESSNA 172

  Fabián Tapia Garcés
  Universidad del Cauca
  Ingeniería Física
  Grupo de Investigación en Ingeniería Aeronaútica y Aviación GIIA
  2018
*/

// Librerías

#include <Servo.h> // Permite controlar la posición de los servomotores
#include <BasicLinearAlgebra.h> // Librería para operar matrices y
vectores

// Cración de objetos Servo

Servo Servo1;
Servo Servo2;
Servo Servo3;
Servo Servo4;
Servo Servo5;
Servo Servo6;

// COONSTANTES:

// Parámetros de Plataforma Stewart medidos en mm

int a = 85;  // Arista mayor plataforma
int b = 200; // Arista mayor base
int d = 40;  // Arista menor base
int c = 10;  // Arista menor plataforma

// Posición de vértices del hexágono base

using namespace BLA; // Permite usar la biblioteca BasicLinearAlgebra

float crobot = (sqrt(3)) / 6; // Constante obtenida en el modelo
cinemático.

// Se crea la Matriz on las posiciones de los vértices del hexágono base
BLA::Matrix<6, 3> V_Base = {crobot*((2 * b) + d), 0.5 * d, 0,
                            -crobot*(b - d), 0.5 * (b + d), 0,
                            -crobot*(b + (2 * d)), 0.5 * b, 0,
                            -crobot*(b + (2 * d)), -0.5 * b, 0,
                            -crobot*(b - d), -0.5 * (b + d), 0,
                            crobot*((2 * b) + d), -0.5 * d, 0
                           };

// Matriz de posiciones de vértices de plataforma móvil.
BLA::Matrix<6, 3> V_Plat = {crobot*(a + (2 * c)), 0.5 * a, 0,
```

```
                                crobot*(a - c), 0.5 * (a + b), 0,
                                -crobot*((2 * a) + c), 0.5 * c, 0,
                                -crobot*((2 * a) + c), -0.5 * c, 0,
                                crobot*(a - c), -0.5 * (a + b), 0,
                                crobot*(a + (2 * c)), -0.5 * a, 0
                               };

// Se deben pasar los grados a radianes para podeer usar la funciones
trigonométricas
float DTR = PI / 180;

// Punto fijo Z.
float Pz_medio = 135;

// Se trabaja en Coordenadas homogeneas.
float Acte = a / sqrt(3);
float Ccte = c / sqrt(3);

// Constantes para la Matriz T
float TR = DTR * 30;
float LX = DTR * 60;
BLA::Matrix<6, 3> T;

//elementos de la Matriz T
float T1_X = 0; float T1_Y = 0; float T1_Z = 0;
float T2_X = 0; float T2_Y = 0; float T2_Z = 0;
float T3_X = 0; float T3_Y = 0; float T3_Z = 0;
float T4_X = 0; float T4_Y = 0; float T4_Z = 0;
float T5_X = 0; float T5_Y = 0; float T5_Z = 0;
float T6_X = 0; float T6_Y = 0; float T6_Z = 0;

//PROBLEMA DE CINEMÁTICA INVERSA

//Parámetros de legs de la plataforma
int P = 33;  // Longitud (mm) de la palanca del actuador
int L = 170; // Longitud (mm) de la unión entre P y cada vértice de la
plataforma

// Ángulos que debe ser rotado cada vértice en torno al eje Z
float S = DTR * 120;
BLA::Matrix<6> Sigma{ -S, -S, 0, 0, S, S};

// Se definen los vectores origen de cada vertice de la base
// EL problema se puede tratar en dos dimensiones por eso se toman las
// componentes x y  de cada vertice de la base
// Org = Base(0:5,[0,1]);
// No es necesario definir la matriz, sólo se accede a los elementos
necesarios de la matriz V_Base

// identificadores para ciclos for de cinemática inversa
int i = 0; // Para ciclo cálculo de constantes A_i, B_i, C_i
int j = 0; // Para ciclo cálculo de ángulos Theta_i

BLA::Matrix<6> A_i;
```

```
BLA::Matrix<6> B_i;
BLA::Matrix<6> C_i;

// Tetha_i es el vector donde se guardan los valores de los ángulos
obtenidos a partir de la cinemática inversa.
BLA::Matrix<6> Theta_i;


// Posición Inicial de la pPlataforma
int pos = 90;

// Vector para leer Datos desde X_Plane
char Data[36]; // Por revisar

void setup()
{
  Serial.begin(38400);  // Se inicializa el puerto 0 de Arduino para
poder visualizar los datos requridos
  Serial2.begin(38400); // Se inicializa el puerto de comunicación entre
Arduino y X_Plane

  // Pin digital de Salida a Servos
  Servo1.attach(2);
  Servo2.attach(3);
  Servo3.attach(4);
  Servo4.attach(5);
  Servo5.attach(6);
  Servo6.attach(7);

  // Posición Inicial de la Plataforma
  Servo1.write(180 - pos);
  Servo2.write(pos);
  Servo3.write(180 - pos);
  Servo4.write(pos);
  Servo5.write(180 - pos);
  Servo6.write(pos);
  delay(10000);

}

void loop()
{
  if (Serial2.available())
  {
    /*
       ADQUISICIÓN DE DATOS DESDE X PLANE 10
    */
    while (Serial2.read() != '$');
    Data[Serial2.readBytesUntil(',', Data, sizeof(Data) - 1)] = 0;
    float  Alpha = atof(Data);
    Data[Serial2.readBytesUntil(',', Data, sizeof(Data) - 1)] = 0;
    float Beta = atof(Data);
    Data[Serial2.readBytesUntil(',', Data, sizeof(Data) - 1)] = 0;
    float Gamma = atof(Data);
```

```
    Data[Serial2.readBytesUntil(',', Data, sizeof(Data) - 1)] = 0;
    float Px = atof(Data);
    Data[Serial2.readBytesUntil(',', Data, sizeof(Data) - 1)] = 0;
    float Py = atof(Data);
    Data[Serial2.readBytesUntil(',', Data, sizeof(Data) - 1)] = 0;
    float Pz_aux = atof(Data);
    float Pz = Pz_aux + Pz_medio;
    float A = DTR * Alpha;
    float B = DTR * Beta;
    float G = -Gamma;


    // Se calcula la posición de los vértices de la plataforma orientada

    T1_X = Acte * ( cos(B) * cos(G + LX) + sin(A) * sin(B) * sin(G + LX)
) + Ccte * ( cos(B) * cos(G) + sin(A) * sin(B) * sin(G) ) + Px;
    T1_Y = Acte * cos(A) * sin(G + LX) + Ccte * cos(A) * sin(G) + Py;
    T1_Z = Acte * ( sin(A) * cos(B) * sin(G + LX) - sin(B) * cos(G + LX)
) + Ccte * ( sin(A) * cos(B) * sin(G) - sin(B) * cos(G) ) + Pz;

    T2_X = Acte * ( cos(B) * cos(G + LX ) + sin(A) * sin(B) * sin(G + LX
) ) + Ccte * ( -cos(B) * sin(G + TR) + sin(A) * sin(B) * cos(G + TR) ) +
Px;
    T2_Y = Acte * cos(A) * sin(G + LX) + Ccte * cos(G + TR) + Py;
    T2_Z = Acte * ( -sin(B) * cos(G + LX) + sin(A) * cos(B) * sin(G + LX)
) + Ccte * ( sin(B) * sin(G + TR) + sin(A) * cos(B) * cos(G + TR) ) + Pz;

    T3_X = -Acte * ( cos(B) * cos(G) + sin(A) * sin(B) * sin(G) ) + Ccte
* ( sin(A) * sin(B) * cos(G + TR) - cos(B) * sin(G + TR) ) + Px;
    T3_Y = -Acte * cos(A) * sin(G) + Ccte * cos(A) * cos(G + TR) + Py;
    T3_Z = -Acte * ( sin(A) * cos(B) * sin(G) - sin(B) * cos(G) ) + Ccte
* ( sin(B) * sin(G + TR) + sin(A) * cos(B) * cos(G + TR) ) + Pz;

    T4_X = -Acte * ( cos(B) * cos(G) + sin(A) * sin(B) * sin(G) ) + Ccte
* ( -cos(B) * cos(G + LX) - sin(A) * sin(B) * sin(G + LX) ) + Px;
    T4_Y = -Acte * cos(A) * sin(G) + Ccte * (-cos(A) * sin(G + LX) ) +
Py;
    T4_Z = -Acte * (-sin(B) * cos(G) + sin(A) * cos(B) * sin(G)) + Ccte *
( sin(B) * cos(G + LX) - sin(A) * cos(B) * sin(G + LX) ) + Pz;

    T5_X = Acte * ( cos(B) * sin(G + TR) - sin(A) * sin(B) * cos(G + TR)
) + Ccte * (-cos(B) * cos(G + LX) - sin(A) * sin(B) * sin(G + LX) ) + Px;
    T5_Y = Acte * (-cos(A) * cos(G + TR)) + Ccte * (-cos(A) * sin(G +
LX)) + Py;
    T5_Z = Acte * (-sin(B) * sin(G + TR) - sin(A) * cos(B) * cos(G + TR)
) + Ccte * ( sin(B) * cos(G + LX) - sin(A) * cos(B) * sin(G + LX) ) + Pz;

    T6_X = Acte * ( cos(B) * sin(G + TR) - sin(A) * sin(B) * cos(G + TR)
) + Ccte * ( cos(B) * cos(G) + sin(A) * sin(B) * sin(G) ) + Px;
    T6_Y = Acte * ( -cos(A) * cos(G + TR)) + Ccte * cos(A) * sin(G) + Py;
    T6_Z = Acte * ( -sin(B) * sin(G + TR) - sin(A) * cos(B) * cos(G + TR)
) + Ccte * ( -sin(B) * cos(G) + sin(A) * cos(B) * sin(G) ) + Pz;

    // Se llena la matriz T a partir de los valores obtenidos.
```

```
    T = { T1_X, T1_Y, T1_Z,
          T2_X, T2_Y, T2_Z,
          T3_X, T3_Y, T3_Z,
          T4_X, T4_Y, T4_Z,
          T5_X, T5_Y, T5_Z,
          T6_X, T6_Y, T6_Z
        };


    //Constantes para el calculo final de la cinempatica inversa
    //Es necesario escribir una vector de constantes A_i

    for (i = 0; i <= 5; i ++ )
    {
      A_i(i) =  (2 * P * (   sin( Sigma(i)) * (T(i, 0) - V_Base(i, 0) )
- cos( Sigma(i) ) * ( T(i, 1) - V_Base(i, 1) ))) ;
      B_i(i) = ( 2 * P * T(i, 2) );
      C_i(i) = ( pow(L, 2) - pow(P, 2) - pow( ( T(i, 0) - V_Base(i, 0) )
, 2) - pow( (T(i, 1) - V_Base(i, 1) ) , 2 ) - pow(T(i, 2), 2)  );
    }

    // Ecuación obtenida a partir de Cinemática Inversa poara obtener los
ángulos de Servo_i
    for (j = 0; j <= 5; j ++ )
    {
      Theta_i(j) = (1 / DTR) * ( acos( (C_i(j) / sqrt( pow(A_i(j), 2) +
pow(B_i(j), 2) ) )   ) + atan( ( pow(A_i(j), 2) / pow(B_i(j), 2))  )  );
    }

    // Se escriben los valores de los ángulos a cada servo
    Servo1.write(180 - Theta_i(0));
    Servo2.write(Theta_i(1));
    Servo3.write(180 - Theta_i(2));
    Servo4.write(Theta_i(3));
    Servo5.write(180 - Theta_i(4));
    Servo6.write(Theta_i(5));

  }

}
```

ANEXO 3: Modelos 3D de elementos de la cabina del Simulador de vuelo dinámico

A continuación se presentan los modelos 3D de los elementos de la cabina del simulador de vuelo diseñados en SolidWorks.

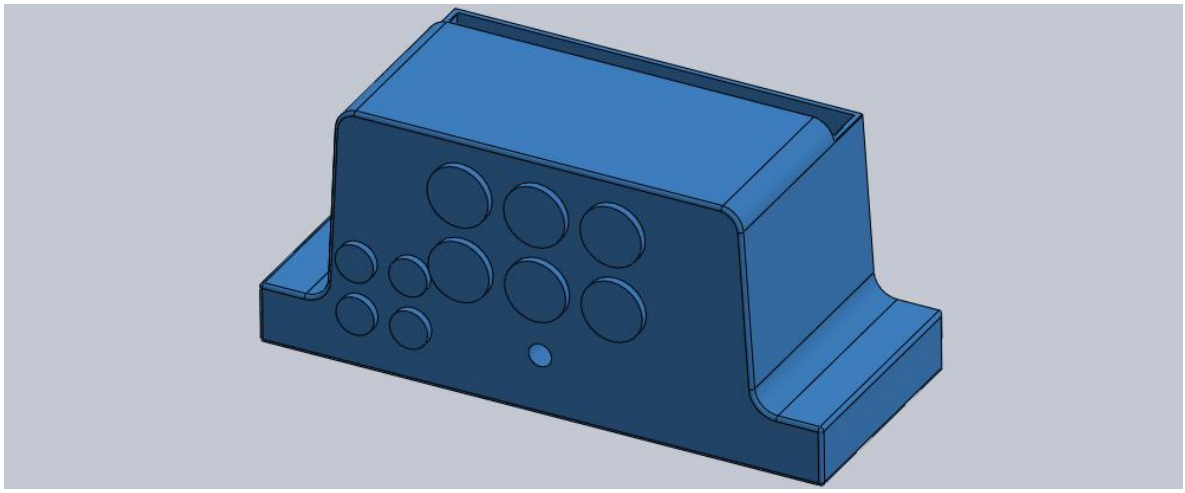


Figura A1. Panel de Instrumentos Cessna 172 3D

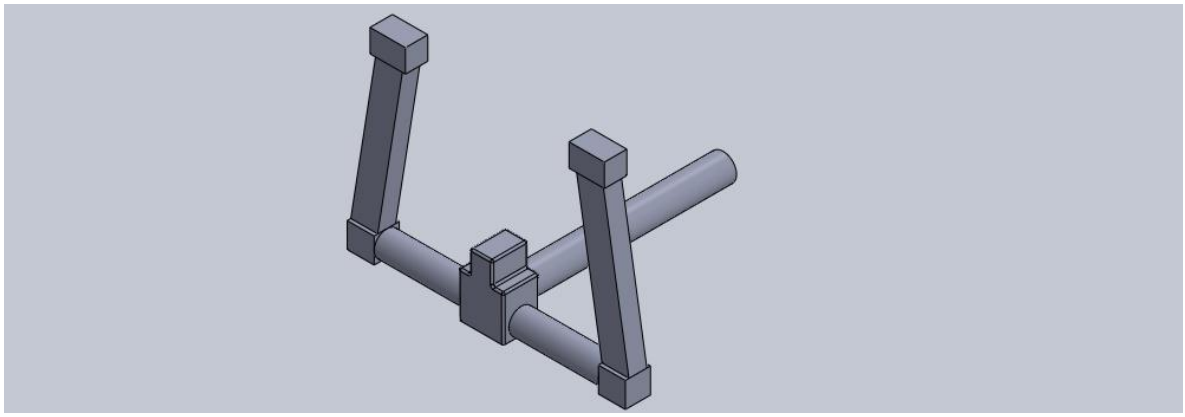


Figura A2. Modelo Timón Cessna 172

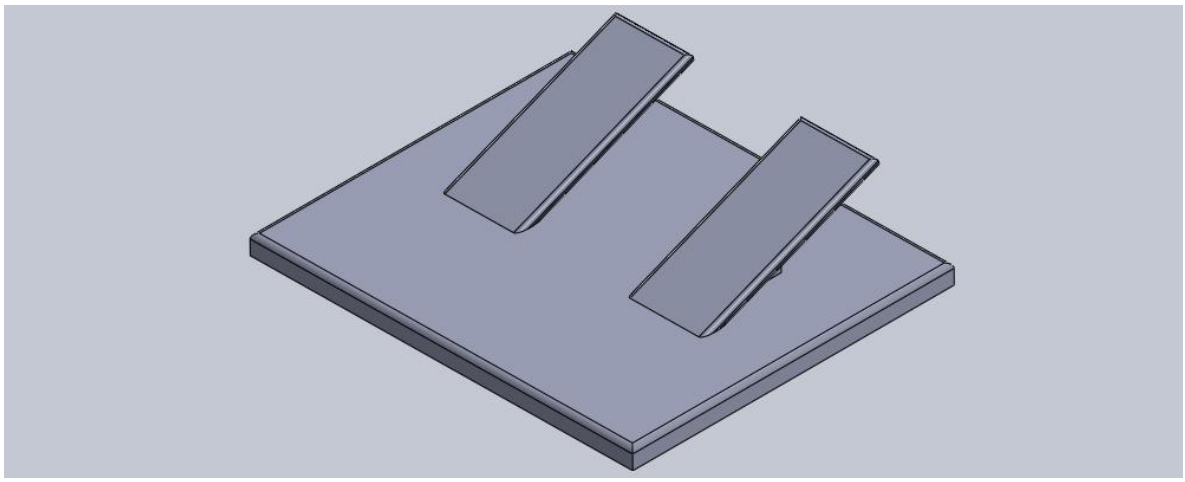


Figura A3. Esquema pedales

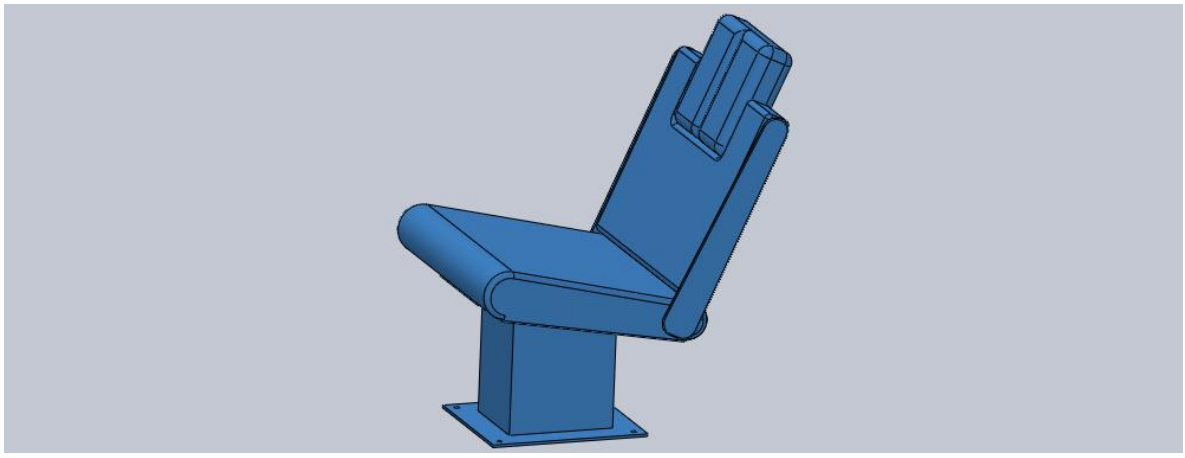


Figura A4. Silla piloto Cessna 172.

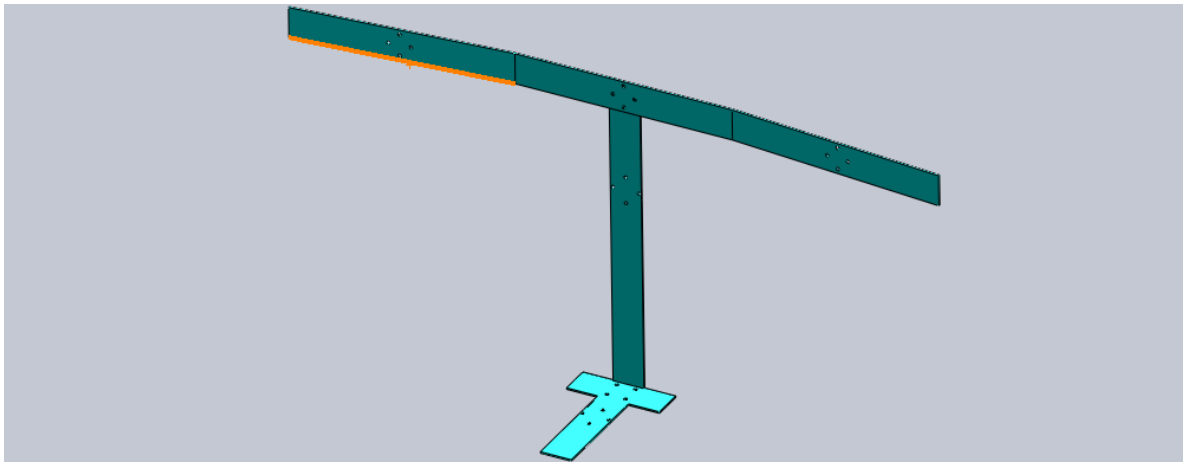


Figura A5. Soporte para monitores y panel de instrumentos

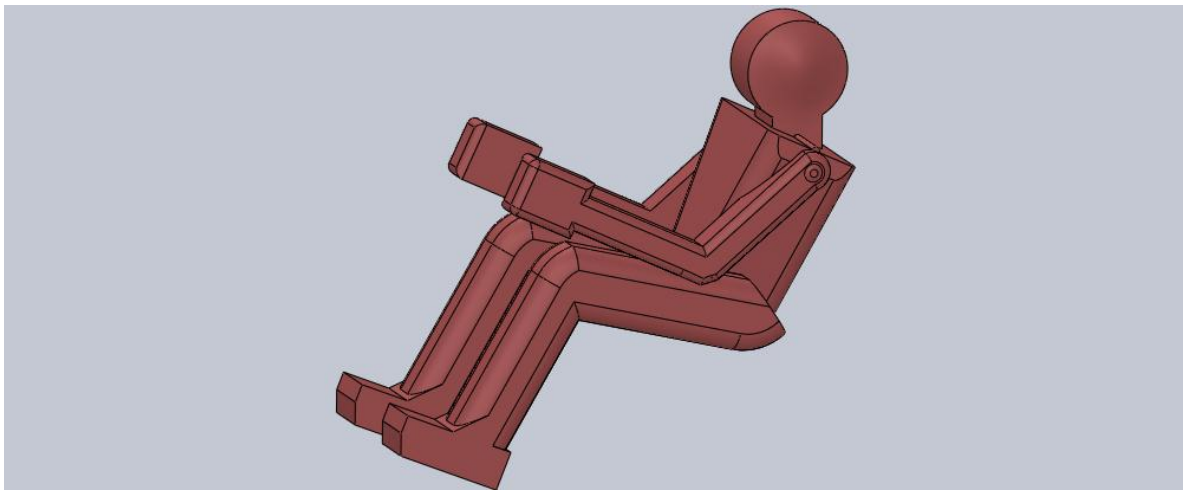


Figura A6. Maniquí de 80 kg.

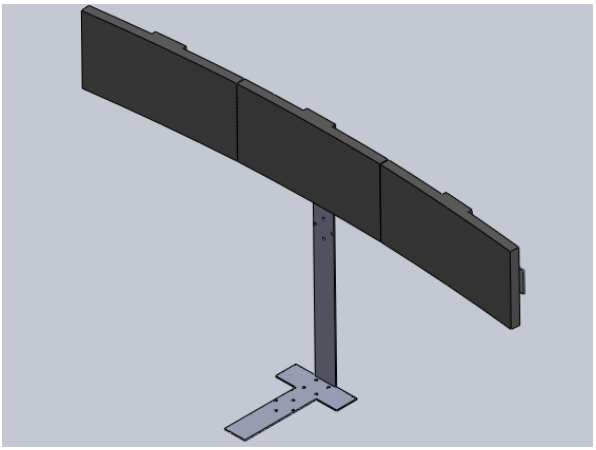


Figura A7. Monitores LG fhd lps 29um68 y soporte utilizado en simulador de vuelo