Universidad de Valladolid

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Mecánica

Modelado Físico y Simulación usando Matlab-Simscape de Planta de Laboratorio

Autor:

Pulido Antón, Sergio

Tutor:

Mazaeda Echevarría, Rogelio

Ingeniería de Sistemas y Automática

Valladolid, Julio de 2021

Resumen

Este proyecto de fin de grado tiene como objetivo desarrollar un modelo físico que permita reproducir el comportamiento del Balancín con Motor-Hélice del Departamento de Automática, además de poner en conocimiento las ventajas que ofrece Simscape.

En este documento se describirá todo el software utilizado, además de definir los componentes y las características de la planta de laboratorio.

Se desarrollará un modelo en Simulink que permita obtener los datos de la respuesta de la planta, y se diseñará el modelo físico en Simscape-Matlab. El cual será calibrado y validado con los datos del modelo real mediante las herramientas que ofrece Matlab.

Palabras Clave

Modelado Físico, Simscape, Simulación, Matlab, Arduino

Abstract

This end-of-degree project has as objective to develop a physical model that allows to reproduce the behavior of the Engine-Propeller Seesaw from the Automation Department, in addition to making known the advantages offered by Simscape.

This document will describe all the software used, in addition to defining the components and characteristics of the laboratory plant.

A Simulink model will be developed to obtain the plant response data, and the physical model will be designed in Simscape-Matlab. Which will be calibrated and validated with the data of the real model using the tools offered by Matlab.

Keywords

Physical modeling, Simscape, Simulation, Matlab, Arduino.

Índice de Contenido

Índice de Figuras

Índice de Tablas

1. Introducción.

1.1. Antecedentes.

Desde los 90 existe una gran demanda de software de modelado que permitieran trabajar en un entorno fácil de utilizar y con gran adaptación sin tener que programar desde cero, es aquí donde resulta realmente útil el modelado orientado a objetos como el implementado en Modelica o EcosimPro e implementado más recientemente en Simscape.

Estas herramientas le ofrecen al usuario poder reproducir un sistema real sin tener que conocer en detalle las ecuaciones que describen la dinámica del sistema, solo conociendo los componentes físicos que lo forman [1].

El modelado y la simulación presentan una capacidad imprescindible para comprender sistemas físicos desde la perspectiva de las diferentes disciplinas de la ingeniería frente a la manipulación, diseño y control del sistema [2].

El modelo a desarrollar en este documento parte de la base del funcionamiento de un helicóptero, en el que bajo el Teorema de Bernoulli se consigue generar la sustentación o empuje al hacer pasar el aire desde la parte superior a la inferior de su rotor principal, gracias al giro del motor.

1.2. Objetivos.

- Familiarizarse con el software Simscape y sus herramientas.
- Presentar las ventajas que ofrece el Modelado Orientado a Objetos frente al Modelado Orientado a Señales.
- Elaborar un modelo físico en simscape capaz de representar adecuadamente el modelo de balancín con motor y hélice.
- Estimar los parámetros del modelo a partir de simulaciones realizadas con el modelo del laboratorio.

1.3. Organización del Documento.

Este trabajo se estructura principalmente en los siguientes 6 capítulos más los anexos:
En primer lugar, el capítulo actual propone los objetivos del proyecto y como está estructurado.
En el segundo capítulo se describe el software utilizado durante todo el proyecto.
En el tercero se concretará la planta real, sus componentes y movimiento, además se desarrolla el modelo matemático del modelo.

A continuación, en el capítulo cuarto se desarrolla el modelo de Simulink, con el que se obtendrán los datos del modelo real.
Seguidamente en el capítulo quinto se expondrán las diferencias entre modelado físico y orientado a señales, y se describirá el proceso seguido para realizar el modelo físico en simscape.
En el siguiente capítulo se estiman los parámetros del sistema utilizando los datos experimentales del balancín, obtenidos con el modelo del capítulo 4.
Y en el capítulo séptimo se realiza la validación del modelo mediante diferentes señales de entrada. Por último, se expondrán unas conclusiones e ideas de trabajos futuros.

2. Software utilizado.

2.1. Matlab 2020b.

Matlab es un software de cálculo numérico diseñado para trabajar con matrices y vectores, utiliza un lenguaje que integra cálculo, programación y visualización grafica basado en una notación matemática sencilla de entender [3].

Algunas de las aplicaciones que Matlab ofrece son: fórmulas matemáticas, Algoritmos, Gráficas, Modelación, Simulación, Adquisición de datos, Análisis de datos, Exploración, Visualización y Aplicaciones con interfaz gráfica.

Matlab se está convirtiendo en una herramienta estándar para problemas de ingeniería y matemática tanto en la formación como a nivel profesional, con una gran aplicación en sistemas de control y para procesar señales [4].

Además, cuenta con un entorno de programación visual iterativo basado en bloques, que permite modelar sistemas dinámicos mediante métodos numéricos, denominado Simulink.

2.1.1. Simscape.

Este módulo amplía la capacidad de modelado de Simulink mediante la implementación de herramientas para modelar, simular y representar 3D sistemas físicos, como si fuese una red física, a diferencia de Simulink que utiliza una red de señales y operaciones matemáticas [5].

Permite integrar el modelo físico directamente en el entorno de Simulink, aprovechando a si las capacidades del mismo en el diseño de sistemas de control [6] y su validación antes de su producción real.

El módulo incluye bibliotecas de bloques que permiten la representación directa de los diferentes componentes de un sistema físico real, y sus relaciones geométricas y cinemáticas, además permite la creación de nuevos componentes utilizando el lenguaje básico de Simscape/Matlab. La biblioteca incluye 6 submódulos principales, como puede observarse en la figura 1.

Figura 1. Biblioteca de Simscape.

Simscape es un entorno más intuitivo, que requiere de un menor esfuerzo en derivación de ecuaciones de movimiento para un sistema particular, lo que permite resolver sistemas complejos en menor tiempo [7].

Pueden encontrarse algunas aplicaciones de este software en [8],[9],[10],[6],[11], además de trabajos enfocados a ser una guía educativa en [12].

2.1.2. Simscape Multibody Link.

Se trata de un complemento de Matlab que permite exportar modelos CAD y convertirlos en modelos de Simscape Multibody. Es compatible con SolidWorks, Autodesk Inventor y PTC Creo [13].

El procedimiento para su utilización será el siguiente, siempre que se necesite exportar modelos CAD será necesario activar Matlab como servidor de automatización, para ello es necesario iniciar Matlab como Administrador y ejecutar el comando *regmatlabserver*, posteriormente ya tendremos disponible el menú en Autodesk Inventor para exportar el ensamblaje generando un archivo XML y los archivos de geometría STEP.
Por último, se realiza la conversión de los archivos anteriores en modelos de Simscape desde Matlab mediante la función *smimport ('nombre.xml').* Sera necesario que todos los archivos se encuentren en la misma ruta en Matlab.

Algunos ejemplos de utilización de este complemento lo encontramos en la creación de un robot industrial KUKA en [14], el modelado de manipuladores de robots en [15].

2.1.3. Arduino Software.

Se necesita el Paquete de Simulink para Hardware Arduino de MathWorks, que es un complemento que permite crear y ejecutar modelos de Simulink en placas Arduino. Además de añadir una biblioteca de bloques propios para comunicarse con sensores y actuadores mediante Arduino [16].

2.2. Autodesk Inventor 2020.

Es un software de simulación computacional y diseño mecánico 3D, que permite definir la geometría, las dimensiones y el material de tal forma que con solo modificar las dimensiones toda la geometría se actualiza.

El diseño se realiza mediante la creación de piezas 3D, a partir de bocetos de dos dimensiones. Y permite el ensamblaje entre diferentes piezas estableciendo las restricciones entre bordes, superficies, planos, ejes y puntos [17].

Permite realizar una exploración visual del objeto, además de análisis dinámicos y de tensiones mediante elementos finitos antes de su fabricación, reduciendo los costes y el tiempo de desarrollo.

3. Modelo real.

En este capítulo se describen todos los componentes que forman el modelo real perteneciente al Departamento de Ingeniería de Sistemas y Automática.
Su funcionamiento básico se basa una hélice movida por un motor, que genera una fuerza de empuje, haciendo que el balancín se mueva respecto a la barra vertical con un movimiento circular. Podemos encontrar más información sobre el modelo matemático del sistema y sobre el sistema de control de posición en los trabajos anteriores [18],[19].
Podemos observar el modelo en cuestión y la caja que contiene la placa Arduino y la fuente de alimentación en la figura 2.

Figura 2. Modelo Balancín con Motor y Hélice.

3.1. Componentes Modelo Balancín con Motor y Hélice.

3.1.1. Base.

Formada por un tubo de acero de longitud L_b= 310 mm, diámetro d_1=25 mm y espesor e= 2 mm, colocada en vertical y soldada a una placa, sobre la cual se coloca la caja donde se encuentra la fuente de alimentación y el sistema de control de la planta.

En su extremo superior tiene un eje de unión con el balancín, además de un potenciómetro que permitirá conocer la posición del balancín, mediante la variación de la tensión que circula por él.

3.1.2. Balancín.

Se trata de una barra maciza de acero inoxidable de L= 560 mm y diámetro d_2= 12 mm, unida a la base mediante un grado de libertad, el giro. Su volumen se puede calcular mediante la ecuación 3.1.

$$V = \pi * L * (d_2/2)^2 = 6{,}33 \cdot 10^{-5} m^3 \quad (3.1)$$

Por lo tanto, siendo la densidad del acero inoxidable de ρ= 7980 Kg/m³, la masa de la barra es M_b= 0,505 Kg.
La barra se encuentra situada en reposo a 44º respecto a la vertical, y unida a una distancia de 220 mm del extremo A y 340 mm del extremo B según la figura 3. La barra puede girar hasta los 135º donde se encuentra limitada físicamente.

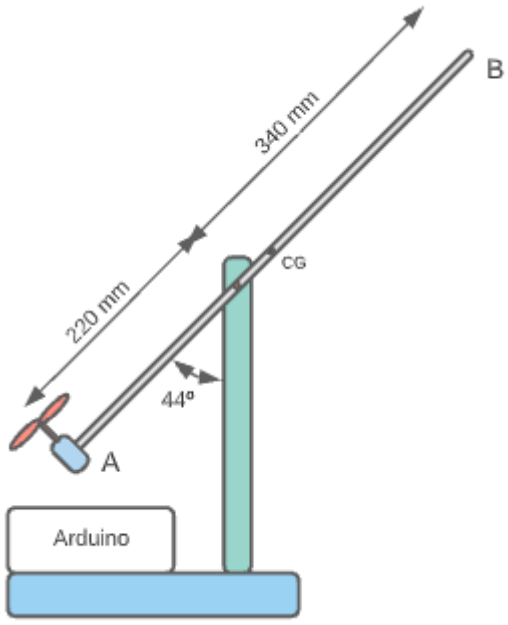

Figura 3. Modelo Balancín en Posición de Reposo.

3.1.3. Hélice.

Se trata de una hélice de nylon con diámetro exterior D=135 mm y tres palas, para motores con eje de 2 mm. Su masa es de aproximadamente M_h = 8 g.
El modelo tiene además instalado una protección alrededor de la hélice.

3.1.4. Motor.

Se utiliza un motor de corriente continua del fabricante Mabuchi [20] de 24V con una velocidad de giro de 14275 rpm y un par de 88 gcm, ambos en situación de máximo rendimiento, y una masa M_m = 70g, el resto de especificaciones técnicas pueden verse en el Anexo I. Será el encargado de suministrar el giro de la hélice, que a su vez genera la fuerza de empuje.

3.1.5. Arduino Mega 2560.

Nuestro modelo monta la placa Arduino Mega 2560 [21] que es alimentada mediante la toma USB. Se utilizará un puerto analógico para recibir la lectura del potenciómetro y un puerto digital para accionar el motor, este puerto solo puede suministrar 5 V lo que hace necesario utilizar una fuente externa de alimentación. Pueden revisarse las especificaciones de la placa en el Anexo II.

3.1.6. Fuente de alimentación.

El modelo cuenta con una fuente de alimentación externa para amplificar la señal de Modulación por Ancho de Pulso (PWM = Pulse Width Modulation) generada por la placa Arduino y que controla el motor de la hélice. En nuestro modelo se utiliza la fuente del fabricante FULLWAT [22] de 24 V y 1 A con una potencia suministrada máxima de 25 W. En el Anexo III pueden consultarse más especificaciones de la fuente.

3.1.7. Transistor NPN.

Se utiliza un transistor NPN modelo BD 139 para poder variar la tensión enviada al motor, ya que la fuente de alimentación suministra una tensión constante de 24 V. Para ello se utiliza un circuito de potencia como el de la figura 4, donde el transistor bipolar se controla mediante la señal PWM recibida de la placa Arduino.

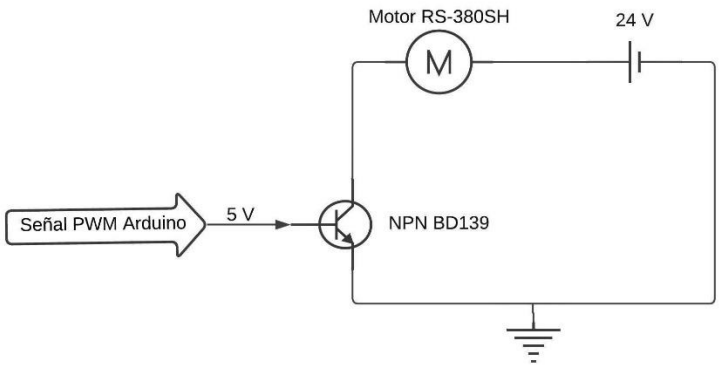

Figura 4. Circuito de Potencia.

3.2. Modelo Matemático.

El movimiento del balancín se trata de una rotación de un sólido rígido alrededor de un eje, que puede describirse mediante la ecuación del movimiento de un sólido rígido, análoga a la Segunda Ley de Newton. Su fórmula seria según la ecuación 3.2:

$$\begin{cases} \sum \tau = I\alpha \\ \tau = r \; x \; F \end{cases} \tag{3.2}$$

Donde:

- **r:** Es el vector de la posición de la fuerza aplicada.
- **F:** Es el vector de la fuerza aplicada.
- **I:** Es el tensor del momento de inercia respecto al eje de giro.
- **α:** Es la aceleración angular del sólido.
- **τ:** Es el torque producido por las fuerzas respecto al eje de giro.

En la figura 5 podemos ver el resumen de las fuerzas aplicadas sobre la planta.

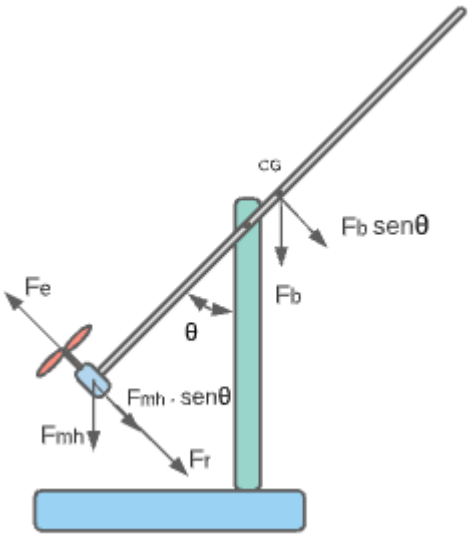

Figura 5. Fuerzas Aplicadas.

Donde:

- **F_b:** Es la fuerza resultante de la masa del balancín, se representa en el lado derecho por ser la barra más larga.
- **F_e:** Es la fuerza de empuje generada por la hélice al girar.
- **F_{mh}:** Es la fuerza resultante de la masa del motor y la hélice.
- **Fr:** Es la fuerza del rozamiento que se opone al movimiento.

3.2.1. Momentos de Inercia.

El momento de Inercia es la propiedad de un cuerpo de oponerse a cualquier cambio en su movimiento. Se trata de una magnitud escalar que únicamente depende de la distribución de la masa de un cuerpo y de la posición del eje de giro, siendo mayor el momento de inercia cuanta más distancia exista entre la masa y el centro de rotación.

Conocido un sistema de partículas y un eje arbitrario, se puede calcular el momento de inercia como la suma de los productos de las masas por el

cuadrado de la distancia de cada partícula al eje, se expresa según la ecuación 3.3:

$$I = \sum m_i \cdot r_i^2 \tag{3.3}$$

Particularizada para un sistema continuo se escribe según la ecuación 3.4:

$$I = \int r^2 dm \tag{3.4}$$

Procedemos a calcular los momentos de inercia del balancín y del conjunto motor-hélice.

3.2.1.1. Momento de Inercia del Balancín.

El balancín tiene una masa M_b= 0,505 Kg y una longitud L= 0,56 m. En la figura 6 mostramos el esquema que utilizaremos.
La masa dm del elemento diferencial de longitud de la barra comprendido entre x y x+dx se expresa según la ecuación 3.5.

$$dm = \frac{M_b}{L} dx \tag{3.5}$$

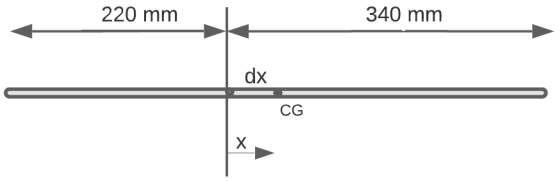

Figura 6. Esquema del Balancín para Calculo de Momentos de Inercia.

Sustituyendo el dm en la ecuación 3.4, obtenemos la ecuación 3.6 particularizada para nuestra barra y el momento de inercia.

$$I_b = \int \frac{M_b}{L} x^2 dx = \int_{-0.22}^{0.34} \frac{0.505}{0.56} \cdot x^2 dx = 1{,}502 \cdot 10^{-2}\ Kgm^2 \tag{3.6}$$

3.2.1.2. Momento de Inercia del Conjunto Motor-Hélice.

Consideramos el motor y la hélice como una masa concentrada en un punto, situado a una distancia r= 0,24385 m, siendo su masa total:

$$M = M_h + M_m = 0{,}008\ Kg + 0{,}070\ Kg = 0{,}078\ Kg \tag{3.7}$$

Aplicando la ecuación 3.3 el momento de inercia del conjunto se obtiene según la ecuación 3.8.

$$I_{mh} = 0{,}078\ Kg \cdot (0{,}24385\ m)^2 = 4{,}638 \cdot 10^{-3}\ Kgm^2 \tag{3.8}$$

Siendo entonces la inercia total sobre el modelo:

$$I = 1.9658 \cdot 10^{-2}\ Kgm^2 \tag{3.9}$$

3.2.2. Torque.

En nuestro modelo el centro de gravedad del balancín se encuentra desplazado 6 cm hacia la derecha, siendo el torque el calculado en la ecuación 3.10.

$$\tau_b = \int_{-0.22}^{0.34} g\frac{M}{L}xdx = \int_{-0.22}^{0.34} 9.8\frac{0.505}{0.56}xsen\theta(t)dx = 0.2969sen\theta(t) \tag{3.10}$$

Para el torque del conjunto motor-hélice considerando todo como una masa puntual se obtiene mediante la ecuación 3.11.

$$\tau_{mh} = r\ x\ F_{mh} = -0{,}24385 \cdot 9.8 \cdot 0.078 \cdot sen\theta(t) = -0.1864\ sen\theta(t) \tag{3.11}$$

Para el par de empuje, la fuerza permanece formando un ángulo de 90º con el vector posición obteniéndose como sigue en la ecuación 3.12.

$$\tau_e = r\ x\ F_e = 0.24385 \cdot F_e \cdot sen90º = 0.24385\ F_e \tag{3.12}$$

Particularizando la ecuación 3.2 para nuestro modelo obtenemos la ecuación 3.13, que podemos comprobar que se trata de un sistema de segundo orden.

$$0.1105sen\theta + 0.24385 \cdot F_e - \beta\dot{\theta} = 0.019658\ddot{\theta} \tag{3.13}$$

Del modelo obtenido anteriormente hay que destacar que es una ecuación no lineal, esto puede observarse en que la variable dependiente θ se encuentra afectada por una función trigonométrica, función no lineal. El carácter no-lineal dificulta la resolución del problema matemático, siendo necesaria la utilización de algoritmos iterativos que requieren de mucho tiempo y capacidad, como los implementados en las herramientas de modelado.

4. Modelo Simulink.

Para poder calibrar el modelo que queremos realizar en simscape, necesitamos realizar un modelo en Simulink que nos permita obtener datos de la respuesta del sistema real.

Para ello se desarrolla el modelo en dos partes:

4.1. Control del Motor.

Se compone de los siguientes bloques de Simulink:

- **Signal Editor**: Permite visualizar, crear y editar señales de entrada mediante una interfaz que permite modificar las señales gráficamente. Además de guardar las señales en un archivo *.mat,* y asignarlas a diferentes escenarios.

- **Saturation**: Permite limitar la señal mediante un límite inferior y uno superior, en nuestro caso se limita entre 0 y 255, que corresponde con la escala de 8 bit admitida por Arduino como salida digital.

- **Arduino PWM**: Pertenece a la librería Arduino y permite enviar la señal de entrada a la placa Arduino y generar una señal de pulsos cuadrados a través del pin asignado, en nuestro modelo se fija el pin 5 y una frecuencia de salida de 492 Hz.

- **Scope**: Nos permite visualizar en una gráfica la señal recibida durante la simulación en función del tiempo, en este caso la señal de entrada.

El esquema resultante quedaría como puede visualizarse en la figura 7.

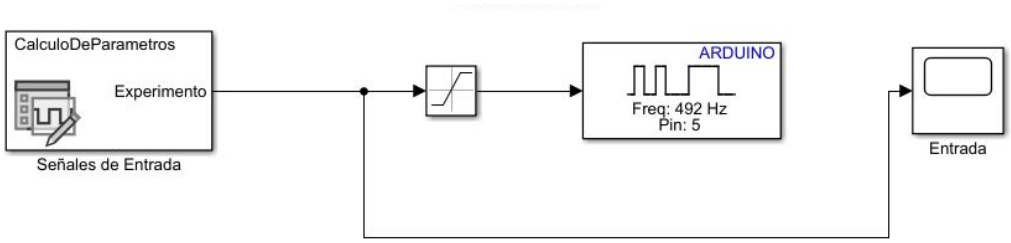

Figura 7. Control del Motor.

4.2. Lectura del Potenciómetro en Ángulos.

La segunda parte utilizada es el esquema de la figura 8, donde los bloques que se utilizarán para recibir la lectura del potenciómetro son:

- **Analog Input**: Permite recibir en el ordenador la señal analógica que Arduino obtiene del potenciómetro como señal digital de 10 bits, es decir en escala desde 0 a 1024. Se configura para trabajar con el pin 4.

- **Data Type Convert:** Permite convertir la señal recibida en el tipo de dato especificado, en nuestro caso en tipo double (doble precisión).

- **To Workspace:** Permite enviar los datos a la Workspace de Matlab para posteriormente trabajar con ellos, en el modelo corresponde con la posición en grados del balancín.

- Dos **scope** uno para visualizar la posición en ángulos, y otro para comparar la señal original leída y la filtrada.

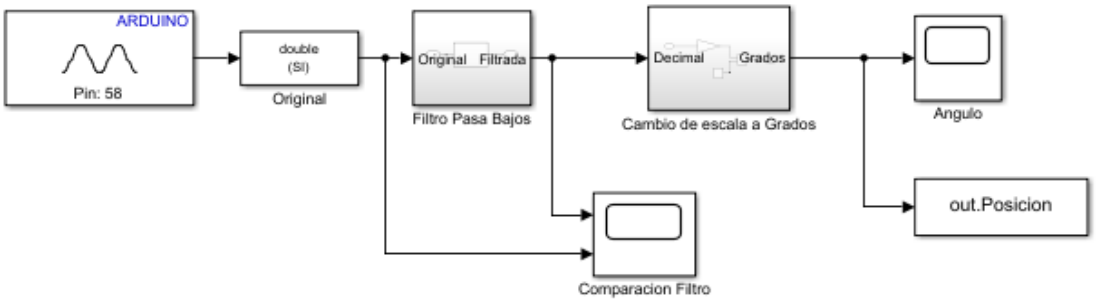

Figura 8. Lectura del Potenciómetro en Ángulos.

A demás se crean los dos subsistemas siguientes:

4.2.1. Cambio de Escala a Grados.

Arduino mide la señal analógica del potenciómetro mediante un voltaje entre 0 y 5 voltios, y nos muestra la lectura mediante una señal digital de 10 bits, es decir mostrando en pantalla un numero desde cero a 1023. Para facilitar la lectura de la posición es necesario transformar esta medida a grados, medida que nos es posible medir mediante el transportador de ángulos colocado en el modelo real.

Para conseguirlo comenzamos colocando manualmente el balancín en diferentes ángulos que mediremos con el transportador de ángulos y anotaremos en cada caso la medida obtenida en Simulink con el bloque de Arduino, obteniendo la tabla 1.

Tabla 1. Lecturas de Posición

Posición en Decimal	Posición en Grados
525	45
565	55
606	65
645	75
683	85
720	95
756	105
798	115
835	125
868	135

Representamos los datos anteriores en la figura 9 y obtendremos la ecuación que relaciona la posición en ambas escalas, siendo en nuestro caso una ecuación lineal como la ecuación 4.1.

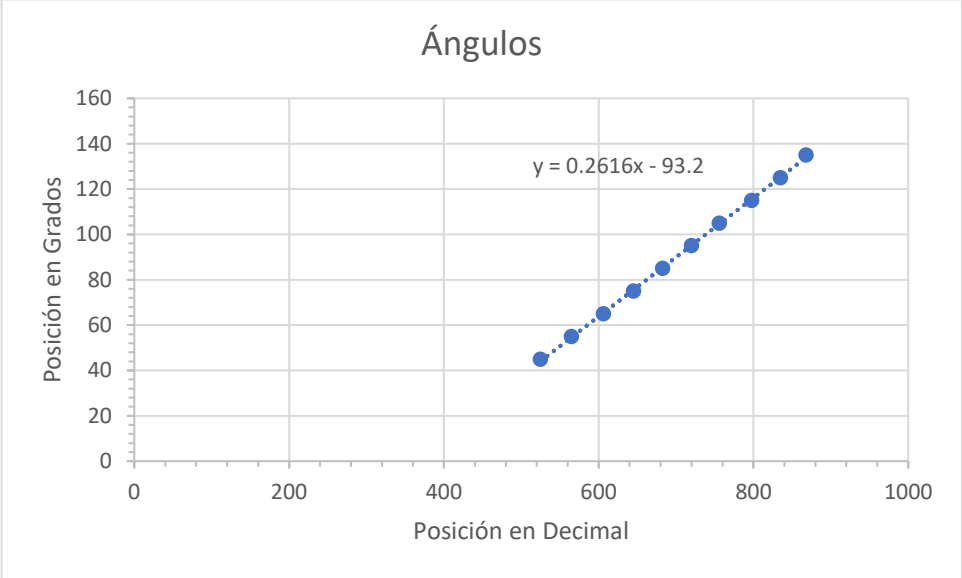

Figura 9. Representación de Lecturas de Posición.

Donde "y" corresponde a la posición en ángulos, mientras que "x" es la posición en escala decimal, "a" es la pendiente de la recta y "b" la coordenada en el origen.

$$y = a * x + b \tag{4.1}$$

La cual se particulariza para nuestro modelo como sigue en la ecuación 4.2.

$$y = 0.2616 * x - 93.2 \tag{4.2}$$

Y se implementa en Simulink mediante el esquema de la figura 10, formado por los siguientes bloques:

- **Constant**: un bloque que proporciona un valor constante.
- **Add**: Permite realizar la suma de las dos señales que recibe.
- **Gain**: Permite el producto de la señal de entrada por un valor fijado.
- **Input y Output:** Utilizados para recibir y enviar respectivamente las señales desde fuera del subsistema.

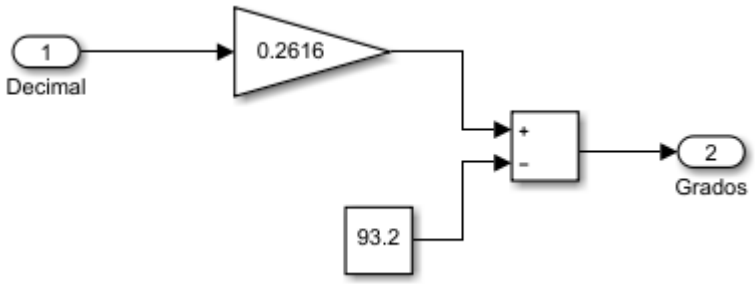

Figura 10. Conversor a Grados.

4.2.2. Filtro Pasa Bajos.

El subsistema permite la atenuación de los valores atípicos y el ruido de la señal de entrada, para ello implementa la ecuación 4.3.

$$S(t) = \begin{cases} Y(t) & t = 0 \\ \alpha * Y(t) + (1 - \alpha) * S(t-1) & t > 0 \end{cases} \quad (4.3)$$

Donde

- **α:** Es el factor de suavizado con valor entre 0 y 1, en este modelo se elige α=0.04.
- **S(t):** Es el valor de la señal de salida ya filtrada.
- **S(t-1):** Es la señal de salida en tiempo t-1.
- **Y(t):** Es el valor de la señal de entrada en periodo de tiempo t.

Esto se implementa con el esquema de la figura 11, donde se utilizan los bloques ya mencionados en el subsistema anterior además de los siguientes bloques:

- **Product:** Permite realizar el producto de los valores de las señales de entrada.
- **Delay:** Retiene el valor de la señal de entrada durante una iteración, devolviendo el valor de la señal en tiempo t-1.

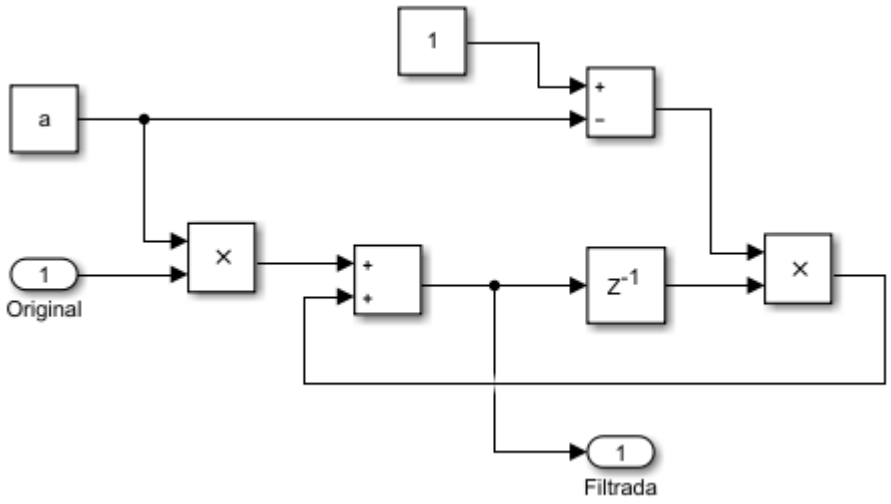

Figura 11. Filtro Pasa Bajos.

Podemos visualizar el efecto del filtro pasa bajos en la figura 12, para una entrada escalón desde 0 a 142, que produce un cambio de posición en decimal desde 523 a 580.

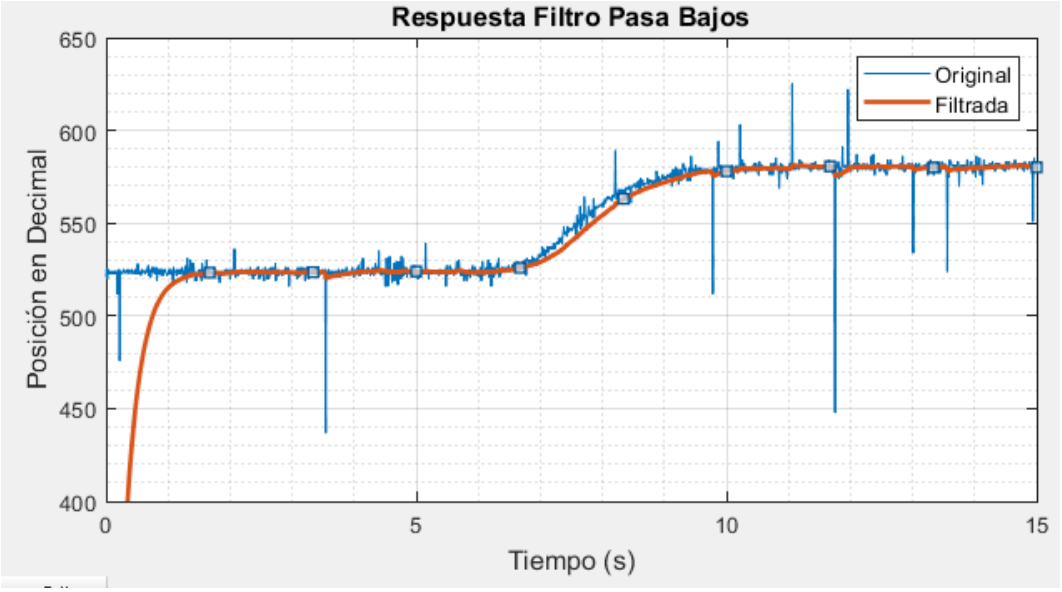

Figura 12. Respuesta Filtro Pasa Bajos.

Como se observa el filtro suaviza adecuadamente el ruido y los valores atípicos recibido en la señal original del potenciómetro, presentando únicamente una pequeña variación durante el cambio. También se observa que durante los primeros segundos de la medición el filtro genera ángulos no reales, menores a la posición de reposo del modelo, por ello de aquí en adelante solo se tendrán en cuenta los resultados a partir de los 3 s.

5. Modelado Orientado a Objetos.

5.1. Conceptos generales.

El modelado orientado a objetos [23] permite generar modelos con los siguientes principios:

- Intuitivos: donde el esquema coincide con la estructura del sistema real, las relaciones entre bloques se basan en intercambio de energía.
- Flexibles: Es posible realizar modificaciones del sistema sin tener que volver a implementar todo el modelo desde el principio, el software deriva las ecuaciones automáticamente y las parametriza individualmente en cada elemento.
- La calibración o ajuste de los parámetros libres del modelo final resultante, así como la validación final del mismo, debe hacerse, apelando como siempre a datos reales obtenidos de experimentos.
- La red física no presenta una dirección de flujo determinada. En herramientas de modelado y simulación como Simulink, el modelo y el orden de cálculo de las ecuaciones están definidos ambos simultáneamente, en el mismo diagrama gráfico que describe el modelo. En Simscape el modelo no especifica ningún orden específico que deba seguirse en la integración del modelo. La causalidad computacional está separada de la descripción del modelo. Esta característica es la que permite conectar componentes que representen elementos físicos reales en diferentes topologías. Es la herramienta la que debe decidir, a partir de la conexión final de los componentes individuales, el orden del modelo concreto que debe ser utilizado. Para que esto sea posible, los modelos matemáticos de los componentes físicos deben ser definidos de manera no causal: como relaciones matemáticas, no como asignaciones computacionales.
- Es posible trabajar con modelos complejos con diferentes constantes de tiempo (elementos discretos). Esto puede ser una dificultad del modelado físico. Es el resultado de la forma en que el modelo global es creado a partir de la conexión de los componentes que lo constituyen. Y así se pueden mezclar dinámicas muy rápidas con otras más lentas. El modelo matemático resultante suele ser del tipo ("*stiff*"), más difícil de integrar numéricamente, y que requiere de algoritmos de integración especialmente adaptados.

Ofrece una representación 3D que permite estudiar el problema dinámico y localizar posibles errores de diseño. Todo lo anterior además de poder contar con bibliotecas permite mejorar la productividad del modelista.

Frente a un modelo orientado a señales como Simulink que [23]:

- No presenta semejanza con el modelo real, el esquema resultante equivale a representar las ecuaciones diferenciales.
- El mínimo cambio en el modelo en Simulink puede implicar una modificación radical del esquema que define el modelo.
- El usuario debe implementar directamente las ecuaciones del modelo en el diagrama gráfico de procesamiento de señales que utiliza Simulink.
- El modelo se encuentra ligado al experimento.
- La relación entre bloques es direccional, este es el propio paradigma de procesamiento de señales.

Existen diferentes softwares en el mercado que implementan el modelado orientado a objetos, pero en este documento nos centraremos en Simscape.

Un claro ejemplo de las diferencias expuestas anteriormente seria por ejemplo para el modelado de un sistema masa-resorte-amortiguador como el de la figura 13. En Simscape (Modelado Orientado a Objetos) el modelo sería como la figura 14, donde puede verse la gran semejanza con el sistema real, mientras que el esquema en Simulink (Modelado Orientado a Señales) sería como la figura 15, que no se parece en nada al modelo real.

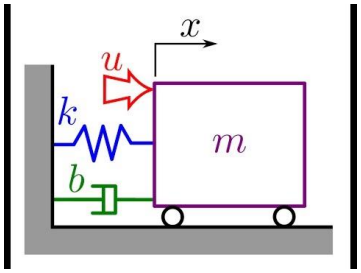

Figura 13. Sistema Masa Resorte Amortiguador Real.

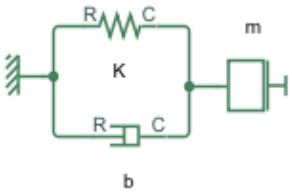

Figura 14. Modelo en Simscape.

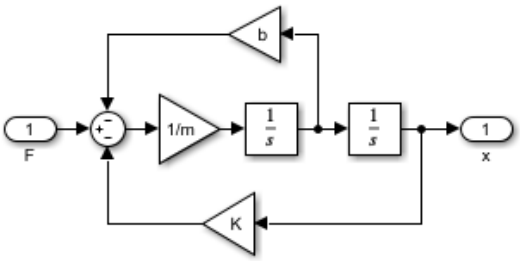

Figura 15. Modelo en Simulink.

5.2. Modelado con Simscape.

5.2.1. Tipos de Variables:

Simscape al igual que otros softwares de modelado físico como [24], EcosimPro [25], Proosis, utiliza dos tipos de variables [26].

- **Across:** Son todas las variables que pueden medirse en paralelo a un componente. Como son la velocidad angular, velocidad de translación, voltaje, etc.
- **Through:** Son todas las variables que se puedan medir en serie con el componente. Como son la fuerza y la corriente.

Las líneas de conexión en la red física aplican principalmente dos reglas en los nodos, las variables Across (como la velocidad angular) mantienen el mismo valor en todas las bifurcaciones de un mismo nodo, mientras que las variables Through (como el caudal) se dividen entre todas las conexiones que salen del nodo como lo haría en un modelo real [27].

Las variables se definen por su magnitud y signo, este definirá la dirección del flujo de energía [27]. El modelo matemático de cada bloque se basa por tanto en las dos variables anteriores, donde su producto determina la potencia transmitida (Flujo de Energía).

Las principales variables utilizadas en nuestro modelo para cada dominio serán según la tabla 2.

Tabla 2. Principales Variable Utilizadas.

Dominio Físico	Variable Across	Variable Through
Rotación Mecánica	Velocidad Angular	Par
Traslación Mecánica	Velocidad de translación	Fuerza
Eléctrico	Voltaje	Corriente

La definición anterior permitirá determinar el tipo de elemento según el signo de la energía.

5.2.2. Tipos de elementos:

Simscape utiliza dos tipos de componentes según si consumen o proporcionan energía [26].

- **Componentes Pasivos:** Aquellos elementos que emplean energía sin importar su orientación, como motores, etc.
- **Componentes Activos:** Aquellos elementos que suministran energía, y es necesario tener en cuenta su orientación, como fuentes de tensión, etc.

Simscape además permite integrar sus modelos con Simulink mediante los bloques Ps-Simulink y Simulink-Ps, que se encargan de cuantificar la energía recibida de la red física devolviendo una señal numérica o viceversa, hay que recordar que las señales de Simulink solo son un valor numérico sin unidades.

Para ello es necesario definir las unidades de la magnitud a trasmitir. Son necesarios para visualizar en pantalla datos o representarlos en gráficos, además de para introducir señales de entrada al sistema. Y de utilidad si queremos implementar una ecuación matemática como haremos con el cálculo de la fuerza de empuje en nuestro modelo.

Un ejemplo de esta interfaz se puede observar en la figura 16, donde la interfaz son los primeros bloques que encontramos a la entrada y salida del área sombreada, el área sombreada corresponde a la red física.

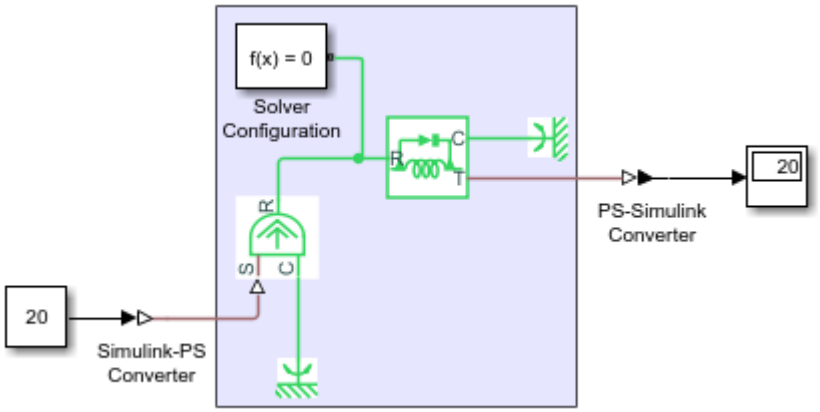

Figura 16. Ejemplo Interfaz Simulink y Simscape.

5.3. Implementación del modelo en Simscape.

Para desarrollar el modelo físico en Simscape comenzaremos con los tres bloques esenciales:

- **Solver Configuration:** Es necesario para configurar el solucionador utilizado en la simulación.

- **World Frame:** Permite establecer el sistema de ejes principales e inerciales de referencia del modelo. Nuestro modelo se encuentra apoyado sobre el plano XY con el eje Z positivo hacia arriba.

- **Mechanism Configuration:**
  Nos permite configurar la acción de la gravedad sobre el modelo como un vector, en este caso como (0 0 -9.8066), siendo negativa debido a los ejes principales escogidos.

Para la representación de los diferentes sólidos y su posicionamiento entre sí, se utilizarán principalmente dos tipos de bloque:

5.3.1. Body Elements:

Estos bloques nos permiten simular los diferentes cuerpos, sus masas y tensores de inercia, su posición y orientación en el espacio. El centro de gravedad se calcula respecto a los ejes locales y fijos al mismo sólido, al igual que el tensor de inercia.
En este proyecto se utilizan los siguientes cuatro bloques:

- **Brick Solid:** Crea un sólido prismático, que utilizaremos para elaborar la base y la representación de la caja que contiene

- **Cylindrical Solid:** Crea solidos cilíndricos como el elemento balancín y el eje de unión de mismo con la base.

- **Revolved Solid:** Crea un sólido de revolución a partir de una representación 2D, se utiliza para crear la barra vertical de la base.

- **File Solid:** Permite importar geometrías más complejas a partir de archivos STEP y STL, generando el sólido. Se utilizará para la creación del motor y la hélice, que hemos elaborado en Autodesk Inventor y exportado a un archivo STEP con el software mencionado en el capítulo 2.

Todos los cuerpos se generan con las dimensiones y masas establecidas en el Modelo Real.

5.3.2. Rigid Transform:

Este bloque aplica una transformación invariable en el tiempo de giro y translación entre los dos bloques conectados a sus puertos de entrada, permitiendo el movimiento como solido rígido.

5.3.3. Revolute Joint:

Representa una articulación con un grado de libertad rotacional, además permiten actuar como sensores. Se configuran dos como sigue:

- **Libertad de giro del Balancín:** En este bloque configuramos la posición de reposo del balancín en 44º, los límites físicos del modelo en 44º y 135º, el coeficiente de amortiguación que denominamos “De” y activaremos la medición de la posición.

- **Giro de la Hélice:** En este bloque solo es necesario activar la medición de la velocidad de giro y el coeficiente de amortiguación denominado como “Rh”.

Para facilitar el trabajo con el modelo se crean subsistemas que engloben los bloques del sólido y de su posición para cada pieza del sistema. Además de un subsistema para la representación del eje del balancín y su grado de libertad. A su vez los subsistemas del motor y la hélice se engloban junto a otros dos subsistemas en un único bloque denominado Conjunto Motor-Hélice. Obteniendo así un esquema del modelo completo como el de la figura 17.

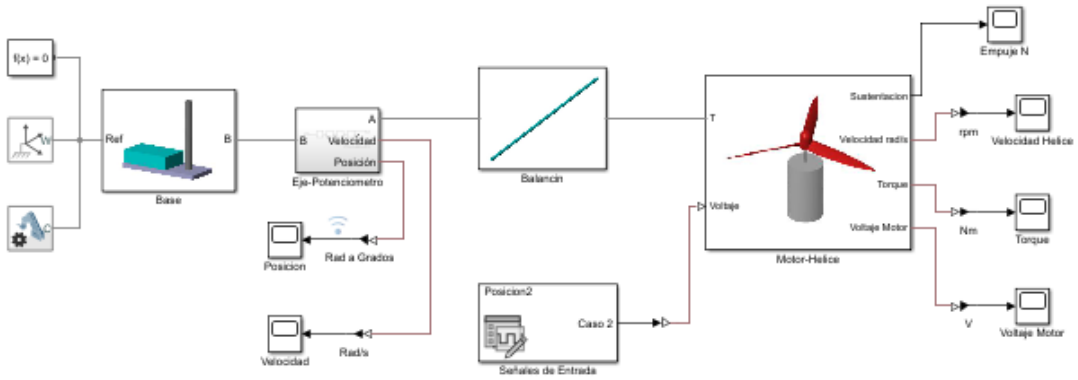

Figura 17. Modelo Completo Simscape.

En este modelo se utilizarán una interfaz Simulink-Simscape para introducir las señales de entrada de funcionamiento del motor. Y cinco interfaces Simscape-Simulink para realizar representaciones graficas de la posición, velocidad del balancín. Velocidad de la hélice, par motor y voltaje del motor.

5.3.4. Conjunto Motor- Hélice.

Además de estar formado por los bloques de solido motor y hélice ya mencionados, contara con los siguientes subsistemas quedando como en la figura 18.

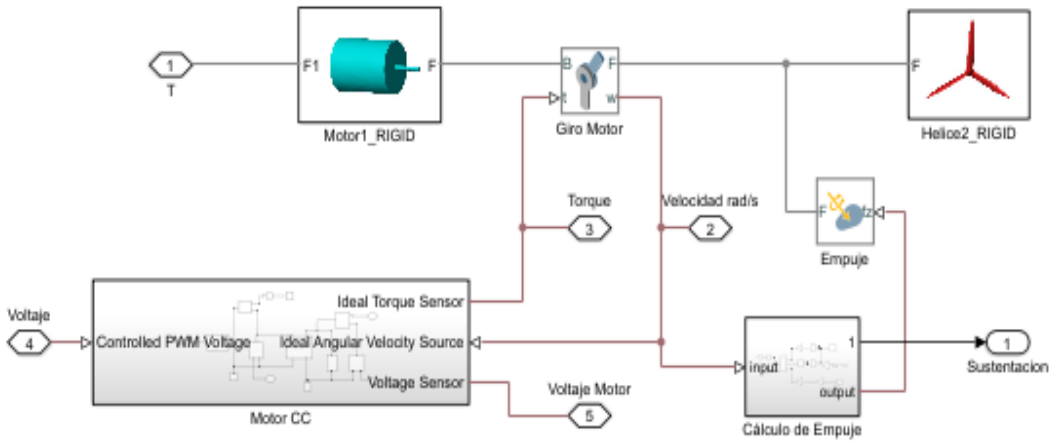

Figura 18. Conjunto Motor-Hélice.

5.3.4.1. Cálculo de Fuerza de Empuje.

Se crea un subsistema que recibe como entrada o input la velocidad de giro de la hélice y calcula la fuerza de empuje o sustentación generada mediante la ecuación 5.

$$F_e = \frac{1}{2}\rho v^2 S_{ref} C_L \qquad (5.1)$$

Donde:

- **ρ:** Es la densidad del aire en Kg/m^3.
- **v:** Es la velocidad de giro de la hélice en m/s.
- **S_{ref}:** Es la superficie generada por la hélice al girar, en m^2.
- **C_L:** Es el coeficiente aerodinámico de la hélice, adimensional.

En este bloque al tratarse de una representación mas matemática me decido por implementar la ecuación mediante bloques Simulink como los mencionados en el capítulo 5, obteniendo el esquema de la figura 19. Podría haberse modelado igualmente mediante la red física de simscape obteniendo resultados igualmente válidos.

Serán necesarios dos bloques conversores para cambiar de la interfaz de Simscape a la de Simulink y viceversa, uno para poder utilizar en el modelo de señales la velocidad medida de la hélice y otro para convertir el resultado del cálculo de empuje en una señal física (Fuerza en N).

La fuerza resultante se aplicará sobre el balancín mediante el bloque External Force and Torque configurado como una fuerza en la dirección del eje z.

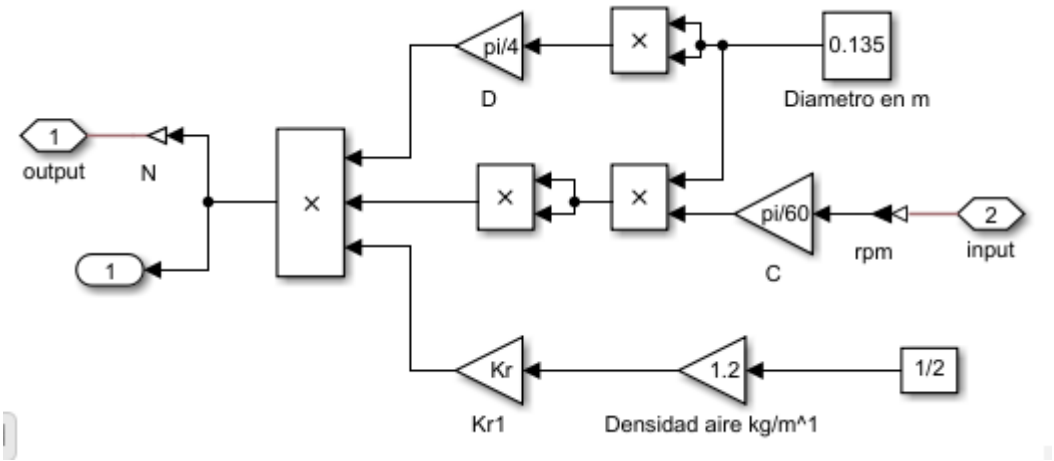

Figura 19. Cálculo de la Fuerza de Empuje.

5.3.4.2. Motor CC.

Este subsistema representa el funcionamiento del motor utilizado para girar la hélice. Está formado por los siguientes bloques:

Parte Eléctrica:

- **Controlled PWM Voltage:** Representa una fuente de voltaje modulada por ancho de pulso (PWM), permite simular la misma señal que genera Arduino en el modelo de Simulink, pero ya ampliada. Se configura la escala de la señal de entrada entre 0 y 255, con una frecuencia de 492 Hz, y la escala de voltaje de salida entre 0 y 24 V.

- **DC Motor:** Representa las características eléctricas y de par de un motor de CC. Se parametriza con las características del motor mencionadas en el capítulo 3, quedando solo por definir la inductancia del circuito eléctrico equivalente que llamamos "I" y la inercia del rotor del motor que llamaremos "Ir".

- **Voltage Sensor:** Nos permite medir la tensión que alimenta el motor.

- **Electrical Reference:** Establece la referencia a tierra en el circuito eléctrico.

Parte Mecánica:

- **Mechanical Rotational Reference:** Establece la referencia inercial de la red mecánica.

- **Ideal Torque Sensor:** Nos permite medir el par generado por el motor, y que se aplica para hacer girar la hélice.

- **Rotational Friction:** Representa la fricción entre cuerpos giratorios en contacto en función de la velocidad relativa, como la suma de fricción de Coulomb y componentes viscosos. Se configura con el coeficiente de amortiguación viscosa como "Rh" y la fricción de Coulomb como "Fc".

- **Ideal Angular Velocity Source:** Se trata de una fuente ideal que genera un diferencial de velocidad angular entre sus terminales en función de la señal de entrada. Nos permite conseguir que el motor alcance una velocidad de giro constante.

El resultado de este modelo puede observarse en la figura 20.

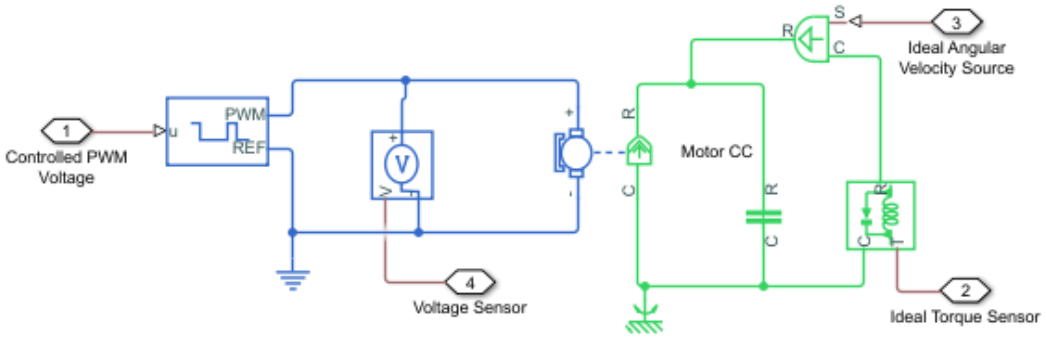

Figura 20. Esquema Motor CC.

6. Calibración.

La calibración consiste en la comparación de los resultados de un modelo con los datos de referencia, es decir con datos reales. Con el objetivo de buscar el ajuste óptimo de los parámetros que afectan al sistema [28].

Para realizar la estimación de los parámetros del modelo se utilizará la app Parameter Estimator de Matlab, que formula un problema de optimización a partir de:

- **p:** Variables de diseño, es el conjunto de parámetros a estimar y sus valores iniciales.
- **J:** Función objetivo, Función de Coste o Error de Estimación. En nuestro modelo utilizamos la función de errores cuadráticos de la ecuación 6.1, consiste en cuantificar la diferencia al cuadrado entre los datos experimentales y los simulados a partir de los valores estimados de nuestros parámetros, a lo largo del horizonte del problema dinámico. Matlab dispone también de otras funciones de coste como el error absoluto o el error residual.

$$J = \frac{1}{N}\sum_{i=1}^{N} e(t_i)^2 = \frac{1}{N}\sum_{i=1}^{N}[y(u,p,t_i) - y_p(t_i)]^2 \qquad (6.1)$$

- **Límites:** Se establece un límite inferior y uno superior para cada uno de los parámetros, como $\underline{p} \leq p \leq \overline{p}$.

A partir de esto aplica un algoritmo de optimización matemática con el objetivo de minimizar la función de coste, es el método que elegimos para el modelo, ecuación. Otras opciones que ofrece son la minimización de los gradientes de la función de coste, el algoritmo de Nelder-Mead o una búsqueda simple de patrones.
El problema matemático a resolver para nuestro modelo queda entonces como en la ecuación 6.2.

$$\begin{cases} min_p\ J = min_p\ \sum_{i=1}^{N}[y(V,p,t_i) - y_p(t_i)]^2 \\ p = (I, Ir, Rh, Re, De, Kr, Fc) \end{cases} \qquad (6.2)$$

Donde:

- **V:** Señal de voltaje en decimal de la entrada.
- **y:** Posición del balancín para el modelo simulado.
- **y_p:** Posición medida en el modelo real en tiempo t_i.

- **N:** Es el número de muestras tomadas en cada simulación, a lo largo del tiempo.
- **p:** Son los parámetros a estimar y sus valores iniciales aparecen en la tabla 3. Se considera valores bastante bajos para los rozamientos y de mayor magnitud para la inductancia y el coeficiente aerodinámico.

Tabla 3. Valores Iniciales de los Parámetros del Modelo.

Descripción	Abreviatura	Valor	Unidades
Inductancia de la Armadura del Motor	l	1	H
Inercia del Rotor	Ir	0.001	kg*m^2
Rozamiento del Eje del Balancín	Re	0.01	N*m/rad
Coeficiente de Amortiguamiento Eje	De	0.01	N*m/(rad/s)
Coeficiente de Amortiguamiento Motor	Rh	0.01	N*m/(rad/s)
Fricción de Coulomb en el motor.	Fc	0.0001	N*m
Coeficiente aerodinámico de la hélice	Kr	2	Adimensional

Comenzaremos creando un experimento donde es necesario indicar la señal de respuesta del sistema, y los datos de la respuesta real obtenidos con Simulink.

Para obtener estos datos utilizaremos en Simulink y Simscape la señal de entrada de la figura 21, se trata de una señal escalón desde 60 a 160, por lo que se parte de que el motor ya se encuentra en funcionamiento y el balancín en posición de reposo.

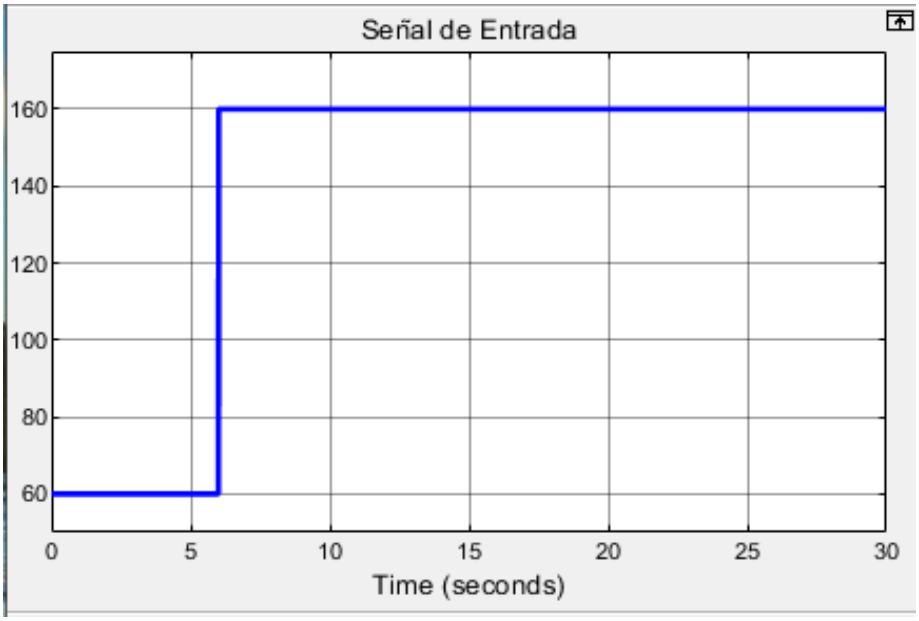

Figura 21. Señal de Entrada utilizada para calibración.

Siendo la estimación obtenida de las variables de decisión la mostrada en la tabla 4.

Tabla 4. Valores Estimados de los Parámetros del Modelo.

Descripción	Abreviatura	Valor	Unidades
Inductancia de la Armadura del Motor	l	0.99824	H
Inercia del Rotor	Ir	$9.9226*10^{-5}$	$kg*m^2$
Rozamiento del Eje del Balancín	Re	$2.2574*10^{-6}$	N*m/rad
Coeficiente de Amortiguamiento Eje	De	0.040936	N*m/(rad/s)
Coeficiente de Amortiguamiento Motor	Rh	$7.6596*10^{-4}$	N*m/(rad/s)
Fricción de Coulomb en el motor.	Fc	$3.2092*10^{-8}$	N*m
Coeficiente aerodinámico de la hélice	Kr	5.7668	Adimensional

Siendo el resultado de la respuesta del sistema el representado en la figura 22, donde podemos ver que con la señal de entrada anterior el modelo pasa de la posición de reposo a una posición de 92º, siendo la respuesta del modelo aproximadamente la misma que la obtenida con la planta de laboratorio.

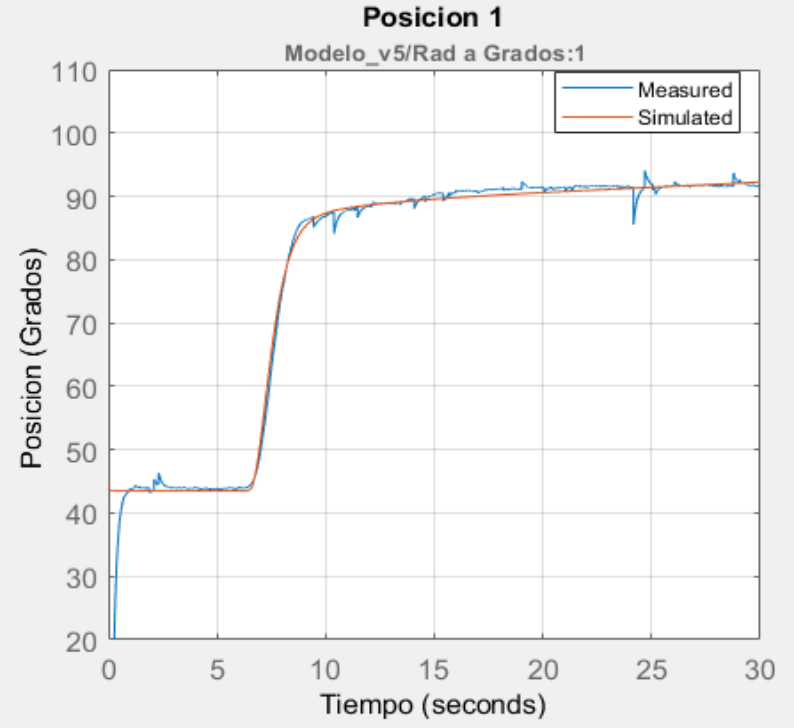

Figura 22. Respuesta del Modelo para la Señal de Calibración.

Procedemos a comprobar otras variables del sistema como la velocidad de la hélice en la figura 23, que se comprueba que ofrece una respuesta adecuada y con valores muy por debajo de la velocidad máxima de 14275rpm, lo cual es de esperar para el modelo real.

También visualizamos el comportamiento del par motor en la figura 24, donde se puede ver que el primer pico se corresponde al arranque del motor y el segundo al cambio de posición y aceleración de la hélice.

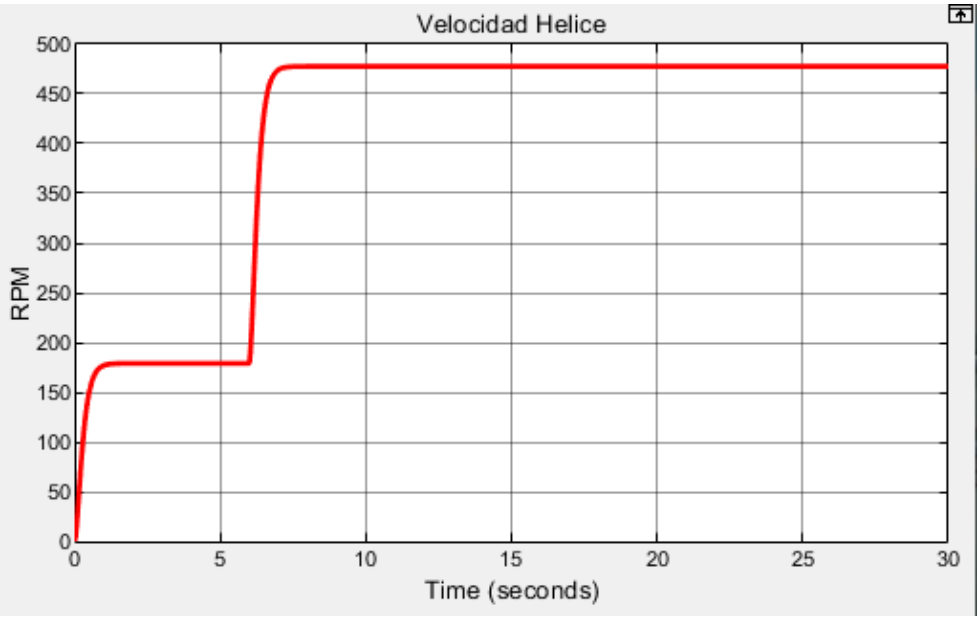

Figura 23. Respuesta de la Velocidad de la Hélice.

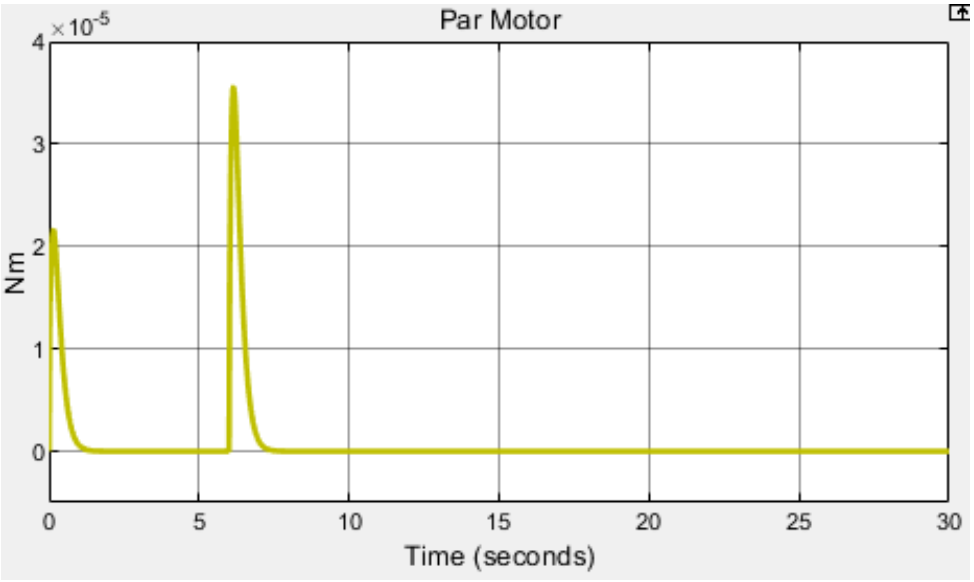

Figura 24. Respuesta del Par Motor.

Y por último representamos la fuerza de empuje en la figura 25, que produce un aumento al arrancar el motor y otro para producir el cambio de posición, siendo contante el resto del tiempo.

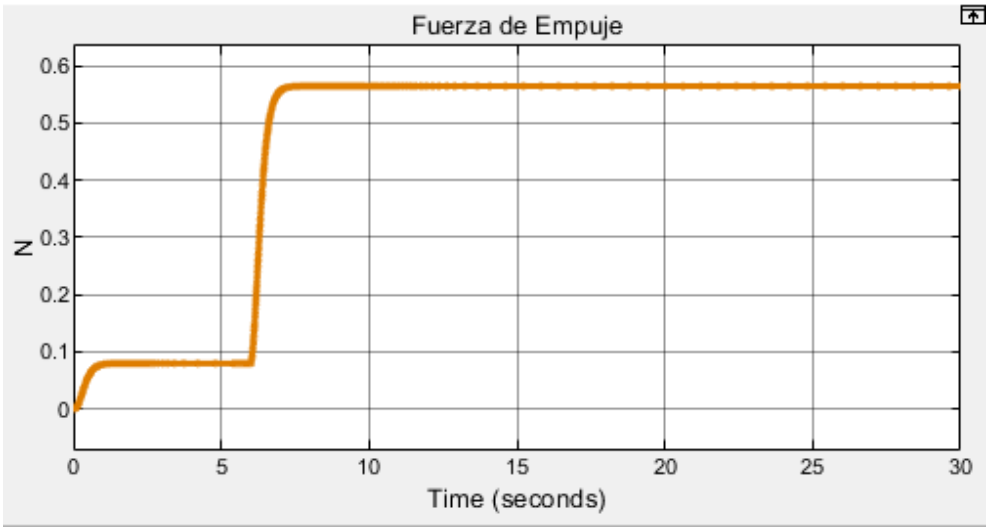

Figura 25. Fuerza de Empuje para la entrada de Calibración.

Se comprueba entonces que las respuestas de las diferentes variables del sistema responden como eran de esperar.

7. Validación del Modelo.

La validación es la comparación de los resultados de un modelo calibrado con datos reales, con el objetivo de generar la confianza suficiente en el modelo dentro de un rango de funcionamiento. Nunca es posible asegurar totalmente la validez de un modelo. Este proceso se puede realizar mediante diferentes técnicas como la inspección visual de la evolución real y la del modelo, estudio de los errores de ajuste de datos, pruebas estadísticas o estudio de sensibilidad a cambios en los parámetros [28].

Se observa en el modelo real que debido a que se está utilizando un modelo sin sistema de control no es posible conseguir una posición estable superado los 95º, debido que a partir de ahí se produce una menor resistencia al movimiento. Por ello, la validación solo se podrá realizar para ángulos entre 43º y 95º, esto corresponde a una señal de entrada en valor decimal de entre 135 y 160.
Para realizar la validación del modelo se utilizarán las siguientes señales de entrada, donde se considera un ángulo grande aquel cercano a 90º y pequeño el cercano a la posición de reposo de 43º.

7.1. Señal escalón para un ángulo intermedio.

El objetivo de esta señal será conseguir alcanzar un ángulo medio desde parado en posición de reposo, se obtendrá aproximadamente unos 65º. Esto se consigue con una señal de entrada en escalón desde 0 hasta 151 como en la figura 26.

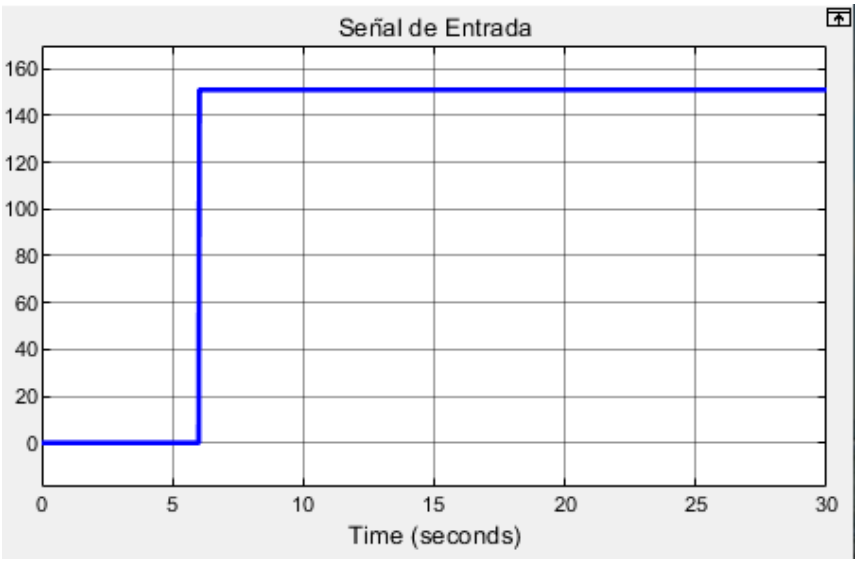

Figura 26. Señal de Entrada para la Situación 2.

Obteniendo la respuesta de la figura 27. Se observa que el sistema logra alcanzar una posición estable próxima a la obtenida con el modelo real, a

excepción del periodo transitorio entre la posición de reposo y la final donde se produce una respuesta típica de un sistema de segundo orden subamortiguado, con un tiempo de pico de 8,88 seg y un sobrepico máximo del 2.77%, esto recordando que el escalón se inicia a los 6 seg. Además de observar que el cambio de posición se produce de forma más rápida que en ángulos grandes como el utilizado en calibración.

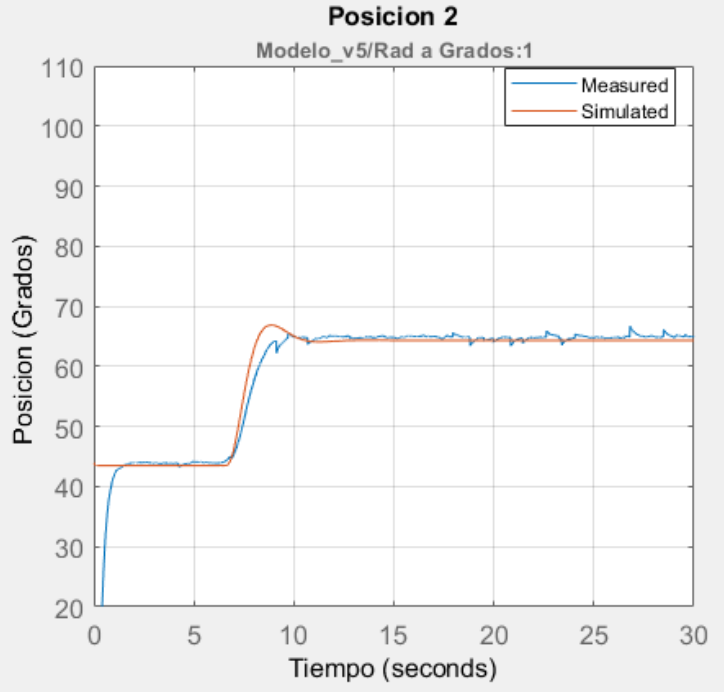

Figura 27. Respuesta Obtenida para la Situación 2.

7.2. Señal escalón para ángulo pequeño.

Para conseguir un ángulo pequeño se utiliza una señal escalón desde 60, donde el motor ya se encuentra funcionando y el balancín está en posición de reposo hasta 140, como en la figura 28, que provoca que el sistema alcance alrededor de 54º.

En la figura 29 se observa que sucede una situación similar al caso anterior con respuesta subamortiguada y estable para el resto del tiempo de simulación, con un tiempo de pico de 8.6 seg y un sobrepico del 2.33%, partiendo de que el escalón se inicia a los 6 seg. Además de lograr aproximar la posición alcanzada por la planta.

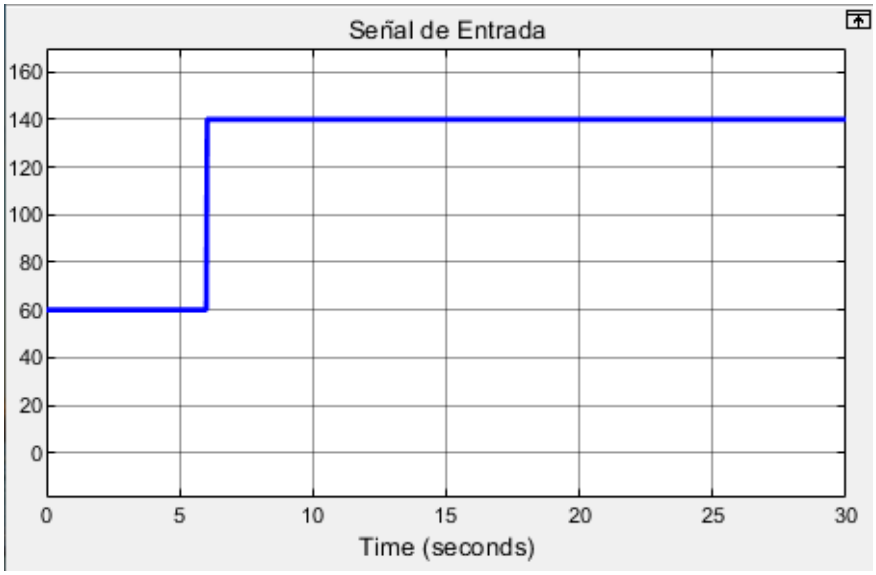

Figura 28. Señal de Entrada para la Situación 3.

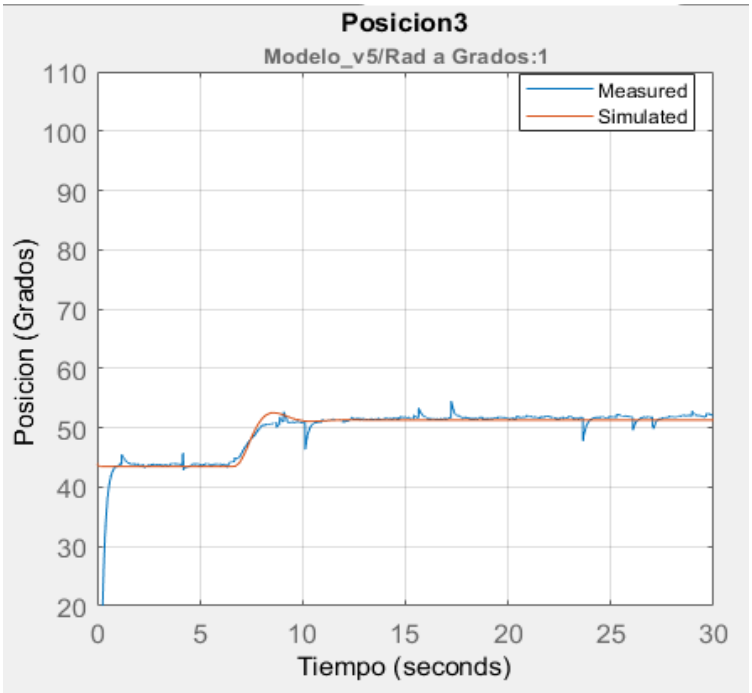

Figura 29. Respuesta Obtenida para la Situación 3.

7.3. Señal con doble escalón consecutivo.

Se utiliza una señal de entrada en escalón desde una posición estable a otra más elevada, para ello iniciamos en reposo con una señal de 100 y subimos a 145 (56.6º) donde se establece el primer escalón y posteriormente subimos a 159 (82.6º), como en la figura 30.

En la figura 31, se puede observar que se alcanza la posición deseada de forma estable, obteniéndose para el primer escalón un tiempo de pico de 7.46 seg y

un sobrepico de 3.18%, y para el segundo escalón un tiempo de pico de 39.9 seg y un sobrepico de 0.121%. Esto permite comprobar que para ángulos grandes el efecto del sistema de segundo orden subamortiguado se atenúa.

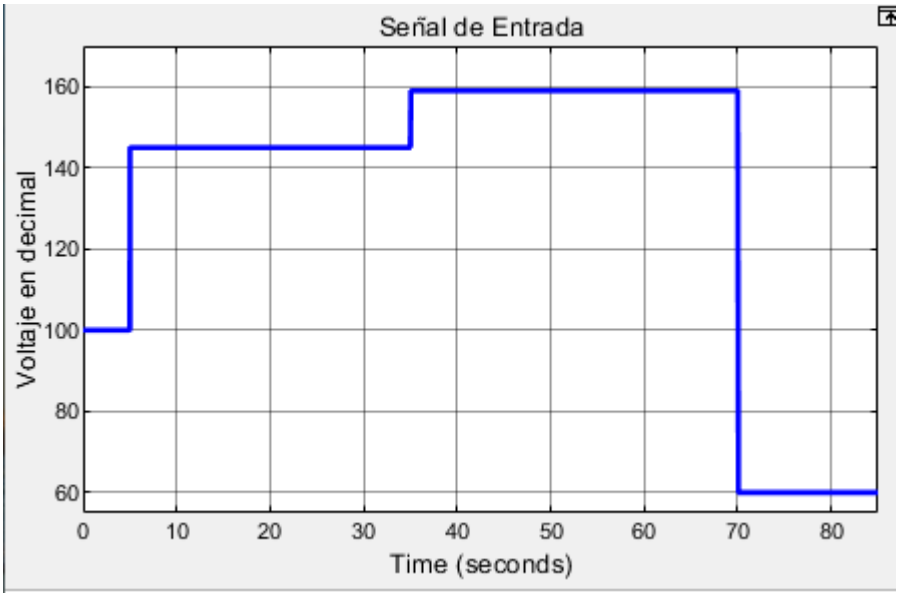

Figura 30. Señal de Entrada para la Situación 4.

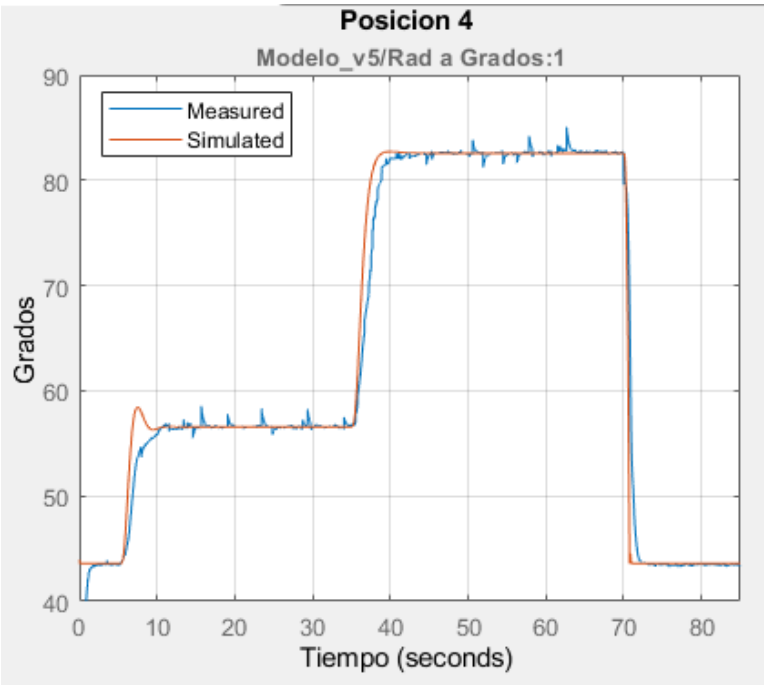

Figura 31. Respuesta Obtenida para la Situación 4.

En la figura anterior además se puede observar que en las zonas de transición es donde el modelo produce una discrepancia frente al modelo real, a pesar de aproximar bastante el comportamiento del mismo.

8. Conclusiones.

8.1. Conclusiones.

Este proyecto me ha permitido ampliar mis conocimientos sobre software de modelado, como Simulink ya conocido de la asignatura de Automática, pero sobre todo Simscape destinado a modelado físico, siendo este último nuevo para mí, lo que ha hecho necesario primero realizar un proceso de formación.

En este documento se comprueba que Simscape permite generar modelos de forma más fácil, facilita la implementación de modelos reales y permite el manejo de modelos más complejos, gracias a la flexibilidad de ser modificados de forma sencilla. Además, facilita el ensamblaje de elementos de diferentes dominios físicos en una única red. Todo esto junto a poder contar con una biblioteca de elementos físicos consigue aumentar la productividad del modelista, aunque no se debe olvidar que el modelado nunca es una tarea sencilla.

Tras elaborar este proyecto se ha completado el diseño y calibración de un modelo que represente de forma adecuada la planta del laboratorio, además de permitirme comprobar las capacidades de las que dispone Simscape-Matlab para los ingenieros.

A la vista de lo anterior se consideran cumplido los objetivos marcados al principio de este proyecto.

8.2. Trabajos Futuros.

Partiendo de los resultados obtenidos se puede plantear la posibilidad de realizar un cálculo de sensibilidades para buscar una mejor elección de los parámetros del sistema, logrando posiblemente un mejor ajuste posterior del modelo.

También se propone continuar con el diseño e implementación de un sistema de control, a partir del modelo ya validado. Esto podría permitir estudiar el modelo ante perturbaciones externas al sistema, como fuerzas puntuales aplicadas sobre el mismo. También sería de utilidad comparar las prestaciones sobre la planta real de un controlador PID (Proporcional, Integral y Derivativo) convencional, obtenido a partir de versiones linealizadas de la planta, con las que se pudieran obtener utilizando controladores más complejos basados en el modelo no-lineal desarrollado.

9. Bibliografía.

[1] J. Serrano *et al.*, "SIMULADORES BASADOS EN HERRAMIENTAS DE MODELADO FISICO PARA EL APOYO A LA ENSENANZA DE CONTROL AUTOMATICO (I): ROBOT MOVIL CON BRAZO FLEXIBLE," 2015.

[2] H. Albaladejo, "MODELADO ORIENTADO A OBJETOS Y SIMULACIÓN DE SISTEMAS SOLARES TÉRMICOS EN EDIFICIOS."

[3] H. Moore, *Matlab para ingenieros*. PEARSON Prentice Hall, 2007.

[4] A. Regidor, "Análisis y diseño en Matlab de un controlador de velocidad para un motor de inducción," p. 117, 2020, [Online]. Available: http://uvadoc.uva.es/handle/10324/41137.

[5] "Simscape - MATLAB & Simulink." https://es.mathworks.com/products/simscape.html (accessed Jun. 18, 2021).

[6] Z. W. Li, G. L. Zhang, W. P. Zhang, and B. Jin, "A simulation platform design of humanoid robot based on SimMechanics and VRML.," *Procedia Eng. ELSEVIER*, vol. 15, pp. 215–219, 2011, doi: 10.1016/j.proeng.2011.08.043.

[7] P. Frankovský, D. Hroncová, I. Delyová, and I. Virgala, "Dynamic Systems in Simulation Environment MATLAB/Simulink-SimMechanics," *Am. J. Mech. Eng.*, vol. 1, no. 7, pp. 282–288, 2013, doi: 10.12691/ajme-1-7-26.

[8] D. Copaci, J. C. Garc, A. Flores-caballero, and D. Blanco, "Simulación De La Mano Humana Mediante Matlab / Simmechanics," *Actas las XXXV Jornadas Automática.*, no. September, pp. 3–5, 2014.

[9] T. D. Le, H.-J. Kang, L. T. Dung, and Y.-S. Ro, "Robot manipulator modeling in Matlab-SimMechanics with PD control and online gravity compensation.," *Int. Forum Strateg. Technol.*, pp. 446–449, 2010, doi: 10.1109/IFOST.2010.5668085.

[10] M. Schlotter, "Multibody System Simulation with SimMechanics," no. May, pp. 1–23, 2003.

[11] I. P. Girsang, J. S. Dhupia, E. Muljadi, M. Singh, and L. Y. Pao, "Gearbox and drivetrain models to study dynamic effects of modern wind turbines," *IEEE Energy Convers. Congr. Expo. ECCE 2013*, no. September, pp. 874–881, 2013, doi: 10.1109/ECCE.2013.6646795.

[12] N. Mather, "Mathematical Modelling Using Predator-Prey," 2006.

[13] MathWorks, “Simscape Multibody Link Documentation - MathWorks España,” 2021. https://es.mathworks.com/help/physmod/smlink/index.html?s_tid=CRUX_topnav (accessed Jun. 19, 2021).

[14] A. D. Udai, C. G. Rajeevlochana, and S. K. Saha, “Dynamic Simulation of a KUKA KR5 Industrial Robot using MATLAB SimMechanics 1 Introduction 2 CAD Modeling and Exporting,” *15th Natl. Conf. Mach. Mech.*, vol. 96, pp. 1–8, 2011, [Online]. Available: http://www.rajeevlochana.com/wp-content/uploads/2015/01/nacomm2011_arun.pdf.

[15] P. Jamali and K. H. Shirazi, “Robot manipulators: Modeling, simulation and optimal multi-variable control,” *Appl. Mech. Mater.*, vol. 232, pp. 383–387, 2012, doi: 10.4028/www.scientific.net/AMM.232.383.

[16] MathWorks Simulink Team, “Simulink Support Package for Arduino Hardware - File Exchange - MATLAB Central,” 2021. https://es.mathworks.com/matlabcentral/fileexchange/40312-simulink-support-package-for-arduino-hardware?s_tid=srchtitle (accessed Jun. 19, 2021).

[17] W. Younis, *INVENTOR® Y SU SIMULACIÓN CON EJERCICIOS PRÁCTICOS - Marcombo, S.A. (ediciones técnicas)*. 2012.

[18] V. Viltres La Rosa, “Control de posición de un balancín con motor y hélice,” p. 59, 2012.

[19] Á. Martín Ballesteros and M. Del Río Carbajo, “Control de Posición de un Balancín con Arduino,” p. 86, 2013.

[20] “Motor Corriente DC, Voltaje 24.00V, R.P.M. 16500rpm - RS-380 SH 20150, ref. 003985-24 | MOOTIO Components.” http://www.mootio-components.com/motor-corriente-dc-voltaje-2400v-rpm-16500rpm-rs-380-sh-20150_ref_003985-24.html#.YM0qlWgzaM9 (accessed Jun. 19, 2021).

[21] “Arduino Mega 2560 Rev3 | Tienda oficial Arduino.” https://store.arduino.cc/arduino-mega-2560-rev3 (accessed Jun. 19, 2021).

[22] “Fuente de alimentación FULLWAT de carril DIN conmutada estándar serie FUS. 25W - 24VDC/1A. - FUS-25D-24 | Fullwat.” http://fullwat.com/FUS-25D-24-fuente-alimentacion-fullwat-carril-din-conmutada-estandar-serie-fus-25w-24vdc1a/ (accessed Jun. 19, 2021).

[23] “EcosimPro/PROOSIS as an Object Oriented (OO) modeling and simulation tool,” .

[24] "Modelica ®-A Unified Object-Oriented Language for Systems Modeling Language Specification Modelica Association," 2021. Accessed: Jun. 18, 2021. [Online]. Available: https://www.modelica.org.

[25] "EcosimPro | Software de modelado y simulación PROOSIS." https://www.ecosimpro.com/ (accessed Jun. 18, 2021).

[26] "Simulación de modelos físicos usando Simscape. - aprendiendo ingeniería." http://aprendiendoingenieria.es/simulacion-modelos-fisicos-usando-simscape/ (accessed Jun. 18, 2021).

[27] "Basic Principles of Modeling Physical Networks - MATLAB & Simulink - MathWorks España." https://es.mathworks.com/help/physmod/simscape/ug/basic-principles-of-modeling-physical-networks.html (accessed Jun. 27, 2021).

[28] C. De Prada and R. Mazaeda, "Model calibration and validation," *Dpt. Syst. Eng. Autom. Control . Univ. Valladolid.*, pp. 85–114, 2019, doi: 10.1007/978-3-030-00569-6_10.

[29] "Arduino." https://www.arduino.cc/ (accessed Jun. 19, 2021).

Anexos.

Anexo I: Motor Mabuchi RS 380 SH 20150.

Figura 32. Motor RS 380 SH 20150 24V.[20]

Tabla 5. Especificaciones técnicas del motor Mabuchi RS-380SH 20150.[20]

Voltaje Nominal	24 V
Velocidad sin carga	16500 rpm
Corriente sin carga	0,16 A
Velocidad Max. Rend.	14275 rpm
Corriente Max. Rend.	1 A
Par Max. Rend.	88 gcm
Corriente de bloqueo	6,40 A
Par bloqueo	650 gcm
Potencia	12.9 W
Peso (M_m)	70 g
Diámetro	27,7 mm
Longitud	37,8 mm
Dimensión Eje	2,3 x 16,4 mm

Anexo II: Arduino Mega 2560.

Arduino es una plataforma diseñada para facilitar la implementación de la electrónica en proyectos multidisciplinares [29]. Su hardware está formado por una placa electrónica que utiliza un microprocesador y puertos de entrada y salida tanto analógicos como digitales para la conexión de sensores y actuadores, además puede incluir conectividad Bluetooth o Wifi (IEEE 802.11) [29].

Sus principales ventajas son la facilidad de programarlas, en comparación con otras placas electrónicas del mercado, además de tratarse de software y hardware de uso libre que puede usarse en cualquier proyecto [19].

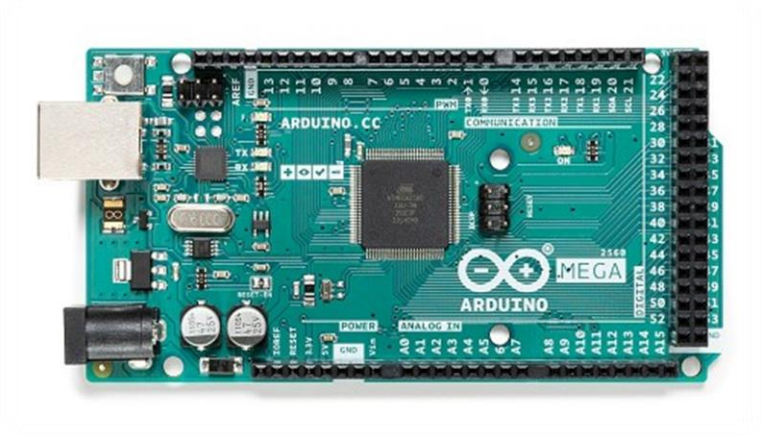

Figura 33. Placa Arduino Mega 2560.[21]

Tabla 6. Especificaciones de Arduino Mega 2560.[21]

Microcontrolador	ATmega2560
Voltaje entrada recomendado	7-12 V
Voltaje de entrada limites inferior/superior	6-20 V
Tensión de funcionamiento	5 V
Pines digitales E/S	54
Pines digitales E/S PWM	15
Pines analógicos entrada	16
Corriente pines de E/S	20 mA
Corriente para pin de 3.3V	50 mA
Frecuencia reloj	16 MHz
Memoria Flash	256KB
SRAM	8 KB
Dimensiones	101,52 x 53,3 mm
Peso	37 g

Anexo III: Fuente de Alimentación FULLWAT FUS-25D-24.

Figura 34. Fuente de Alimentación FULLWAT FUS-25D-24. [22]

Tabla 7. Especificaciones de la fuente de alimentación FULLWAT FUS-25D-24. [22]

Potencia	25 W
Voltaje de entrada AC	100 – 240 VAC
Salida DC	24 VDC con 1 A
Rizado y Ruido	240 mV
Rendimiento	88%
Temperatura de funcionamiento	-15 – 50 ºC
Dimensiones	45 x 35 x 102 mm