UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERÍA

DISEÑO, CONSTRUCCIÓN Y CONTROL DE UN DISPOSITIVO VTOL DE 2 GDL

TESIS

Que para obtener el título de

Ingeniero Mecatrónico

P R E S E N T A N

Víctor Octavio Romero Rivas

Saúl González Duardo

DIRECTOR DE TESIS

Dr. Edmundo Gabriel Rocha Cózatl

Ciudad Universitaria, Cd. Mx., 2023

Agradecimientos

VÍCTOR

Quiero aprovechar este espacio para expresar mi profundo agradecimiento a todas esas personas que hicieron posible este trabajo y me brindaron su apoyo durante todos estos años.

A mis padres Mónica y Víctor, cuyo tiempo, trabajo y cariño fue un gran sustento y motivación durante toda mi carrera. Que sepan que siempre serán una gran parte de la razón de este y muchos otros logros.

A mi hermano Alejandro, que a pesar todo, sé que en el podré encontrar un amigo y consejero; hoy y siempre. A mis abuelos, tíos y primos; en quienes siempre encuentro aprecio y sabiduría. Que me impulsan a ser feliz y constantemente mejorar sin perder de vista los valores que componen a un hombre de bien.

A mis amigos, que me acompañaron durante este camino y me brindaron su amistad, su experiencia y momentos inolvidables. Aquellos que conocí durante la carrera, con los que aprendí, sufrí, reí y conviví todos estos años; sé que se convertirán en excelentes profesionistas y espero poder llamarlos amigos durante el resto de mi vida. Y también a aquellos que desde mi niñez me han acompañado y ocupan un lugar especial en corazón.

A todos mis profesores que se esfuerzan y apasionan por enseñar y que, junto con la universidad, me brindaron una formación amplia e integral, además de profundizar mi gusto y pasión por la carrera y brindarme los recursos, medios y apoyo necesarios para hacer posibles mis estudios.

SAÚL

La presentación de este trabajo tiene un gran significado dentro de mi vida profesional y personal, por lo que en estos párrafos quiero manifestar mi gratitud y gran aprecio a quienes me ayudaron durante su elaboración y fueron importantes a lo largo de mi recorrido académico.

Principalmente a mis padres Joel y Mayela, con los que siempre pude contar y tener el apoyo en cada una de mis decisiones. Nunca terminaré de agradecerles todo lo que han hecho por mí y por mi hermana, espero algún día devolverles una parte de este favor.

A mi hermana Karen, que ha estado presente en los momentos importantes y siempre formará parte de mis motivaciones para continuar mejorando.

A Víctor y Guillermo, mis compañeros de la facultad con quienes me divertí mucho en el transcurso de la carrera y que demostraron ser grandes personas a las cuales admiro. Especialmente quiero agradecer a Víctor con quien realice este trabajo ya que sin su ayuda y capacidades no habría sido posible obtener este resultado.

A aquellos amigos y por supuesto a mi novia, que me dieron ánimo y consejo durante los momentos en que les compartí parte mis frustraciones y desmotivaciones, me da mucho gusto contar con ustedes, gracias a sus palabras e inclusive regaños logré alcanzar esta meta.

A mis sinodales y profesores que fueron parte de mi formación, les estoy muy agradecido por el gran empeño que demuestran en su profesión sobre todo a aquellos que lo hacen con entusiasmo porque nos transmiten ese gusto por el conocimiento.

Muchas gracias.

Índice

Índice de figuras

Índice de tablas

"La experiencia sin teoría es ciega, pero la teoría sin experiencia es un juego meramente intelectual"
Immanuel Kant

"Aquel que no está dispuesto a cambiar su forma de pensar, no cambiará nada"
Winston Churchill

"*Una vez que te percatas de tu ignorancia, ya no puedes ignorarla."*
Ana Luisa Selene Álvarez

Parte I

Introducción

1. Objetivo general

El objetivo de este proyecto es el diseño y construcción de un dispositivo VTOL, cuyos grados de libertad corresponden a su inclinación (Roll) y su posición vertical respecto al suelo, así como el desarrollo y la implementación de un control de lazo cerrado para regular cada una de estas variables. Este sistema servirá con fines demostrativos y experimentales para su uso en la investigación y docencia.

2. Planteamiento

En la ingeniería, la comprensión de nuevos conceptos se consolida mediante su implementación en problemas y aplicaciones prácticas, además de proveer una nueva perspectiva de sus posibles alcances. Es común encontrar limitaciones y discrepancias entre el conocimiento teórico y práctico debido a las herramientas y tecnologías, suposiciones, simplificaciones y otros factores externos no contemplados o sumamente complejos. El ingeniero debe ser capaz de abordar estos inconvenientes para lograr realizar diseños, proyectos y alternativas de solución funcionales. Las áreas de la mecatrónica y el control automático no son la excepción y es benéfico para los alumnos, investigadores y profesionistas enfrentarse a ellos y adquirir experiencia.

Las plantas de laboratorio son dispositivos que permiten verificar experimentalmente conocimientos prácticos y teóricos. En diversas ocasiones están basadas en problemáticas comunes o aplicaciones populares. La intención de este trabajo es aportar una nueva herramienta de soporte para la investigación y educación en el laboratorio de control de la Facultad de Ingeniería, referente a este problema.

Se planteó abordar el trabajo siguiendo cada etapa del desarrollo de un proyecto de ingeniería hasta concluir con la obtención de un prototipo funcional. Los objetivos particulares de cada etapa se especifican en cada una.

Primeramente, durante el diseño conceptual se definen los requerimientos, objetivos principales y secundarios, se establece un diagrama para ayudar en la identificación de las funciones lo que permite tener un concepto general más claro del sistema y definir los elementos necesarios de cada subsistema. Posteriormente, se describen las características y el funcionamiento de cada componente seleccionado, así como un diagrama donde se especifica la distribución física de los componentes. Se asignan los materiales para la fabricación de las estructuras y se muestra el ensamble de las piezas modeladas en CAD y un análisis CAE hecho con este mismo ensamble. Se concluye la sección de diseño con la construcción del dispositivo donde se describen los procesos utilizados para su manufactura y los pasos generales de cómo se realizó el ensamble.

En la segunda sección, se describe el proceso de diseño e implementación del controlador donde se muestran las diferentes configuraciones de controladores implementadas entre las que se encuentran el control por retroalimentación de salidas y el control PID. En la sección se incluye el modelado matemático del dispositivo, el diseño del controlador, los diagramas de bloques, la simulación mediante Matlab y Simulink, los diferentes métodos de sintonización, así como los detalles de la implementación. Finalmente se muestran los resultados y conclusiones de dichos procedimientos.

3. Estado del arte

3.1. Historia del control automático

La ingeniería busca aplicar la ciencia para beneficio de la humanidad, lo cual implica satisfacer una necesidad o cumplir un fin común mediante recursos técnicos y físicos. Naturalmente, durante este proceso, se desarrollan y construyen grupos de elementos interconectados entre sí con un propósito llamados sistemas; cuando estos manipulan variables, ya sea físicas o de otra índole, para cambiar el comportamiento original se les conoce como *sistemas de control*. Este tipo de técnicas han sido fundamentales para para el progreso de varias ramas de la ingeniería, tales como la robótica, mecatrónica, computación, electrónica, comunicaciones, entre otras.

La existencia de estos sistemas antecede a la humanidad y muchos de ellos ocurren de forma natural, por ejemplo, los procesos biológicos que son capaces de regular la temperatura corporal, la concentración de sustancias, el crecimiento y comportamiento a través del tipo y cantidad de hormonas.

Según Bissell (2009)[1] y Nise (2011)[2], la historia de los sistemas de control creados por el hombre comienza alrededor del año 300 A.C. en Grecia cuando Ktesibios desarrolló un reloj que marcaba el tiempo mediante el goteo de agua en un contenedor, para eso el goteo debía ser constante y por ende, el nivel del agua debía mantenerse. La solución fue un sistema de válvula con flotador similar al que tienen los inodoros hoy en día. Mas tarde este mismo sistema se difundió y utilizo para otras aplicaciones como control del nivel de aceite en lámparas y dispensadores de vino.

En el siglo XVII surgieron sistemas para controlar la presión y temperatura. En 1681, Denis Papin creó una válvula de seguridad similar a la de las ollas de presión modernas, pero que usaba un peso para mantenerse cerrada; al variar el peso se podía controlar la presión dentro del contenedor. Mientras tanto, en Holanda, Cornelis Drebbel trabajaba en un sistema mecánico para controlar la temperatura en un contenedor mediante la expansión del mercurio y alcohol; esta accionaba un apagador que regulaba la flama. Al descender la temperatura este se abría y al exceder cierto limite se cerraba.

Más tarde, en el siglo XVIII, surgió el control de velocidad aplicado a molinos y máquinas de vapor. En 1745, Edmund Lee implementó la idea de modificar el ángulo de las aspas de los molinos para controlar el efecto del viento sobre su velocidad y regularla. Un principio similar se empleó para que las aspas siempre estuvieran en dirección del viento. A finales de siglo, James Watt implementó el regulador centrifugo (Flyball governor) en sus máquinas y motores de vapor. A medida que la velocidad de rotación aumentaba, las masas a los extremos se elevaban por la fuerza centrífuga y limitaban el paso de vapor para reducir su velocidad. Cuando disminuía demasiado este permitía un mayor paso de vapor aumentando la velocidad. De esta forma se logró mantener la velocidad oscilando alrededor de un valor fijo dependiente de la masa y dimensiones del gobernador.

Durante el siglo XIX se incrementaron los esfuerzos y la investigación sobre la dinámica de los sistemas y su estabilidad, dando pie a la formación y formalización de la primera teoría de control. En 1868, James Clerk Maxwell publica su criterio de estabilidad para sistemas de 3er orden. Seis años más tarde Edward John Routh logró extender el criterio a sistemas de 5to orden y en 1877 publica *The criterion of Dynamic Stability* donde se estableció lo que hoy en día se conoce como el criterio de estabilidad de Routh-Hurwitz. Posteriormente, en Rusia, Alexandr Michailovich Lyapunov extendió este método a sistemas no lineales para su tesis doctoral.

En el siglo posterior esta rama de la ingeniería sufrió un gran impulso debido a los conflictos de la época. La demanda de más y mejor tecnología armamentista provocó un impulso en sistemas de seguimiento y apuntado automático, lo que llevó a la creación de servomotores más potentes y eficientes. También surgieron sistemas de navegación y estabilización para aeronaves y navíos. En 1922, la teoría desarrollada por Nicholas Minosky para mejorar el desempeño de sistemas de control automático derivó en la creación y aplicación de los primeros controles Proporcional-Integral-Derivativo (PID) utilizados para mejorar el direccionamiento de barcos, apuntado de armas y estabilización de numerosos sistemas. A su vez, a finales de los 20s e inicio de los 30s, H.W Bode y H. Nyquist desarrollaron el análisis de amplificadores retroalimentados para mejorar

las comunicaciones y ampliar su alcance. Estas contribuciones evolucionaron hacia métodos de análisis de frecuencia y diseño de sistemas de retroalimentación. En 1948, Walter R. Evans desarrolló un método gráfico para representar los polos y raíces de las ecuaciones características de los sistemas retroalimentados conocido como *root-locus*. Todas estas técnicas fundaron el análisis y diseño de controladores para sistemas lineales. Más tarde, la navegación de aeronaves y misiles demandó controladores que consideraran varias entradas y salidas en los sistemas dando pie a la representación de estados de los sistemas y sus técnicas de análisis.

Hoy en día estas teorías han evolucionado de muchas formas, por ejemplo: el control adaptativo cambia los parámetros de control para mantener un desempeño óptimo cuando cambian las condiciones, el control óptimo modifica las ganancias para priorizar el desempeño de una variable especifica y el control difuso (Fuzzy logic) utiliza técnicas empíricas para desarrollar controladores fáciles de implementar incluso cuando no es posible modelar el problema de forma sencilla o precisa. También han surgido nuevos métodos que involucran inteligencia artificial (IA) y redes neuronales artificiales para sistemas sumamente complejos.

3.2. Control en UAVs y VTOL

Una de las áreas beneficiadas del desarrollo de distintos sistemas y técnicas de control a lo largo de la historia es la aeronáutica a través de la modernización de sus sistemas, controles de vuelo, métodos de despegue e incluso la creación de aeronaves no tripuladas.

Muchos consideran que la aviación moderna inició a partir de 1903 con el vuelo del *Flyer* de los hermanos Orville y Wilbur Wright que era un pequeño plano de madera y tela impulsado por un motor de fabricación propia. Desde entonces el diseño, desarrollo y construcción de aeronaves ha ido mejorando y creciendo junto con los avances de la física e ingeniería. Pero, los impulsos tecnológicos más grandes que se suscitaron fueron debido a los conflictos armados en el siglo XX.

Con el avance de los sistemas de comunicación durante la primera guerra mundial surgieron las primeras propuestas de UAVs (Unmanned aerial vehicles). Los UAVs son vehículos aéreos no tripulados que resultaron a partir del peligro que corrían los pilotos en misiones de alto riesgo. [3]El primer prototipo de aeronave no tripulada fue desarrollado en Inglaterra en 1917; su propósito era ser utilizado como blanco de práctica para los pilotos. Controlado por ondas de radio, era posible pilotar la nave sin necesidad de un tripulante dentro. Mientras que, en Estados Unidos, el torpedo aéreo conocido como *Kettering Bug* alzó el vuelo por primera vez en 1918. Ambas aeronaves fueron prometedoras en los vuelos de prueba, pero no se usaron durante este conflicto.

No fue sino hasta el periodo entre guerras en 1935 donde los ingleses comenzaron a utilizar los blancos de práctica a control remoto. Se cree que el nombre de uno de estos modelos llamado *DH.82B Queen Bee* o abeja reina en español es el responsable de que se acuñara el termino *Drone* o zángano para los vehículos aéreos no tripulados. Durante esta guerra, tanto los aliados como la Alemania Nazi desarrollaron bombas y torpedos operados por ondas de radio y televisión; sin embargo, todos requerían de un operador. No pasaron de ser aviones controlados remotamente sino hasta la Guerra de Vietnam[4], donde proyectos de vehículos aéreos autónomos estadounidenses categorizados como “Clasificado” fueron enviados a misiones de ataque y reconocimiento. En 1973 el gobierno estadounidense aceptó que los utilizó durante este conflicto.

Conforme los métodos de manufactura mejoraron, muchos de los elementos mecánicos, eléctricos y electrónicos que conforman a los UAVs se lograron miniaturizar permitiendo así vehículos más pequeños, complejos e incluso menos costosos. Esto, aunado al desarrollo de nuevas teorías de control automático resultó en los drones, UAVs, y aeronaves RC que conocemos actualmente.

Estos sistemas de control remoto y autónomos tienen aplicaciones como:

- Reconocimiento y monitoreo de zonas remotas, peligrosas o de difícil acceso.
- Labores de vigilancia.
- Captura de imágenes y videos de cualquier índole que requiera una toma aérea.
- Transporte local de mercancía de poco tamaño y volumen.

- Aeromodelismo.
- Recreación y ocio.
- Investigación y adquisición de datos.
- Educación sobre aeronáutica, control, sistemas mecatrónicos y otras áreas de la ingeniería.
- Aplicaciones militares armamentistas y de reconocimiento.

Como se mencionó, otro avance logrado en este ámbito, mediante la aplicación de estas teorías de control, es en relación con la diversificación de los sistemas de despegue de las aeronaves. Tal como muestran Intwala y Parikh [5], actualmente se pueden encontrar distintos tipos de propulsión, siendo algunas más adecuadas para distintas situaciones.

La mayoría de las aeronaves comerciales aceleran en una pista hasta que generan la suficiente fuerza de sustentación para poder despegar. Normalmente esto requiere grandes velocidades y por ende una gran distancia en la pista de despegue para alcanzar dicha velocidad, sin embargo, existen aeronaves que no necesitan una pista *per se*, sino simplemente una plataforma. Cuando alcanzan la fuerza de sustentación a velocidades bajas se les conoce como aeronaves de despegue corto o STOL (Short take-off and landing). Otros modelos son capaces de generar esta fuerza sin acelerar en una pista, a estos se les conoce como VTOL (Vertical take-off and landing).

Los dispositivos VTOL son aeronaves con la capacidad única de despegar, flotar y aterrizar de forma vertical. Además de eso pueden volar a muy bajas velocidades y aterrizar o despegar de lugares pequeños a diferencia de las convencionales. Actualmente esta tecnología se implementa de dos diferentes formas[6]:

- Hélices: la fuerza de sustentación se logra por medio de la rotación de hélices alrededor de una masa central, generando un gran cambio de presión que impulsa la aeronave hacia arriba. Los helicópteros son los ejemplos más viejos, comerciales y conocidos con este tipo de tecnología de despegue vertical. Se utilizan como transporte de personal, vehículos de rescate, de carga, de combate, etc. En esta categoría también recaen los cuadricópteros, tricópteros, hexacópteros y demás multicópetros. Estas configuraciones son populares en drones eléctricos y pequeños. Sus ventajas son mayor maniobrabilidad, capacidad de carga y menores costos de fabricación.
- Propulsión: Las alas de estas aeronaves son fijas. Normalmente estas naves redirigen la fuerza generada por las turbinas o propulsores para despegar y aterrizar de forma vertical. Más adelante cambian a una configuración de vuelo tradicional para evitar las desventajas que conllevan las hélices; de esta forma logran una mayor altura, velocidad y menor gasto de energía durante el vuelo, pero son más costosos y difíciles de fabricar. El primer avión de esta configuración fue el caza Harrier o *Harrier jump jet*[7], el cual era un diseño británico que surgió a partir de los años 60 y que utilizaba propulsión vectorial para conseguir sus capacidades VTOL.

Un aspecto clave para lograr este comportamiento en las aeronaves es el control de cada uno de los actuadores y servomecanismos. Estos deben de trabajar en conjunto y de forma precisa para mantener la altura, modificarla, avanzar y girar en el aire a una velocidad segura. Las técnicas de control son las responsables de que todo esto suceda de forma exitosa.

3.3. Sistemas de control de laboratorio

El avance de los modelos, teorías, elementos y sistemas de control depende ampliamente del interés que el tema genera y las aplicaciones que tiene, pero esto solo es el primer paso. Para el desarrollo de nuevas tecnologías, formular nuevas teorías o lograr un descubrimiento se deben comprender los fundamentos, es decir, las características y fenómenos sobre los que se soportan temas de mayor complejidad o aplicabilidad.

Este conocimiento se forma a través de estudios, experimentación y experiencia. En consecuencia, contar con herramientas que faciliten el proceso de aprendizaje y que mejoran la comprensión del tema son de gran importancia.

Los conceptos de aprendizaje son comúnmente ejemplificados por medio de fenómenos idealizados diseñados para remarcar el tema de estudio sin inconvenientes. Pero, al querer emplear este conocimiento en aplicaciones reales, muchas veces hallamos obstáculos y problemas que no fueron considerados pertenecientes a un área distinta de conocimiento o habilidad. Los modelos caseros, estudiantiles y plantas de laboratorio son ideales para afrontar esta tarea.

Los proyectos académicos normalmente se realizan con poco presupuesto y tiempo limitado, por consiguiente, el funcionamiento en la mayoría de los casos no es similar al esperado por aspectos mecánicos, eléctricos, electrónicos o de programación. Estos defectos repercuten en los resultados de los controladores que se basan en un modelo simplificado, por lo que a muchos estudiantes se les dificulta probar de forma práctica los conceptos de control.

Existen empresas que se dedican a fabricar bancos de prueba y proporcionar equipo de laboratorio de control para mejorar las oportunidades de experimentación. Tanto el hardware, en especial los sensores y actuadores, como el software especifico constituyen equipos robustos aptos para realizar estas pruebas.

Una de ellas es *Quanser*[8], una empresa contemporánea de origen canadiense que investiga, diseña, manufactura, y suministra equipos de laboratorio a nivel global. Sus dispositivos, como los que se muestran en la Figura [3.1], son usados en el área de la robótica, informática, electrónica, aeronáutica, etc. Algunos otros ejemplos de empresas que actualmente proporcionan equipo afín a las universidades, instituciones y laboratorios, son *Tecquipment*[9], *Feedback*[10] y *Amira*[11]. Muchas veces ofrecen seguimiento, cursos de preparación, artículos y clases sobre los principios de funcionamiento de sus productos. En su mayoría ofrecen artículos referentes a pruebas "clásicas" de control como el *Ball & beam*, péndulo invertido, drones, servomotores o manipuladores.

Figura 3.1: *Plantas de laboratorio de Quanser* [1]

El objetivo de esta tesis es ofrecer un dispositivo que facilite el aprendizaje y la investigación de sistemas de control orientado a cuestiones innovadoras y de interés actual, cuyas características son diferentes al resto de los dispositivos desarrollados por las empresas antes mencionadas y orientado hacia las aeronaves.

[1] Los modelos de Quanser mostrados son: *Aero 2*, *2 DOF Ball Balancer*, *QUBE - Servo 2* y *Linear Servo Base Unit with Inverted Pendulum* - https://www.quanser.com/products/ (22/mayo/2022)

"A excepción del hombre, ningún ser se maravilla de su propia existencia."
Arthur Schopenhauer

"Es imposible para un hombre aprender lo que cree que ya sabe"
Epicteto

"Lo peor que puede pasarle a un hombre es llegar a pensar mal de sí mismo."
Johann Wolfgang Von Goethe

Parte II

Diseño y construcción del dispositivo

El diseño, en la ingeniería, es un proceso que define y desarrolla soluciones pertinentes a problemas no resueltos o nuevas alternativas a problemas con soluciones previamente establecidas, cuyo resultado es generalmente un prototipo, producto o proceso. En cualquier caso, se requiere de un entendimiento claro de la función y el desempeño esperado del producto final, [12].

En esta sección se busca la aplicación de una metodología de diseño con el fin de obtener un modelo físico de la planta de laboratorio junto con su interfaz. Sus objetivos particulares son: identificar las necesidades para poder plantear las metas del diseño, esclarecer las funcionalidades y características particulares del sistema que permitan proponer un concepto de solución, detallar los componentes empleados y finalmente, fabricar y ensamblar el prototipo.

4. Metodología de diseño

El proceso de diseño abarca toda una serie de transformaciones que le ocurren a un objeto determinado a partir de un estado inicial hasta que se alcanza uno nuevo en que él ha cambiado. Dicho proceso finaliza cuando el producto resultante adquiere las características deseadas de tal forma que satisface los requerimientos planteados por quien diseña [13].

Cada diseñador puede generar su propio proceso de diseño, además, el seguimiento de este cambia y se modifica para cada tipo de proyecto; no existe una secuencia universal perfectamente definida para hacerlo. No obstante, se pueden demarcar algunas fases generalmente invariantes que lo caracterizan y que a menudo se efectúan de forma iterativa. A continuación se mencionan algunas de estas etapas representativas:

- Definir el problema e identificar la necesidad: Es el primer paso en el diseño conceptual donde se reconoce la existencia de una *necesidad* y se explora el problema para mejorar su comprensión. Las necesidades son útiles para crear un sentido claro de los problemas que son de interés.
- Definir requisitos y especificaciones: Se traducen las necesidades del cliente en sentencias útiles llamadas *requerimientos*, que son parte necesaria para el establecimiento de los criterios de diseño. Las *especificaciones* que son desglosadas de los requerimientos ayudan al equipo de diseño a cuantificar el cumplimiento de las necesidades, a establecer los objetivos y a delimitar restricciones.
- Generar múltiples soluciones: Una vez identificados los elementos que componen al problema se generan distintas propuestas; esta etapa está relacionada con el proceso creativo que involucra la creación de ideas llamadas *conceptos* y es probablemente el punto clave durante el proceso. Para realizarlo se puede recurrir a técnicas como la llamada “lluvia de ideas”.
- Elegir la mejor solución: Se evalúan los conceptos con respecto a las necesidades del cliente y limitaciones propias, se comparan los puntos relativamente fuertes y débiles de cada uno y luego se selecciona uno o más para su investigación, prueba o desarrollo. Es un proceso convergente e iterativo donde se escoge un conjunto grande de conceptos hasta llegar a uno más pequeño, pero estos pueden combinarse y mejorar. Finalmente se escoge uno dominante.
- Realizar el trabajo de desarrollo o detalle acerca del diseño preferido: Tras la evaluación de conceptos y la elección del ganador, se analiza su desarrollo y se detallan aspectos de la solución como las dimensiones, geometrías, procesos de fabricación de las piezas, características de los elementos incluidos y análisis de su funcionamiento siempre teniendo en cuenta las especificaciones.
- Construcción de prototipo: El objetivo es obtener un producto preliminar al aplicar los procesos de manufactura y ensamble pertinentes. Sirve para identificar errores de diseño, conflictos en la producción o problemas de desempeño una vez construido el objeto. En algunas ocasiones el proceso de

diseño puede concluir con el *prototipo* o continuar hasta finalizar la vida útil del producto.

Simultáneamente, es posible replantear el esquema de diseño como un modelo prescriptivo para motivar al diseñador a adoptar formas sistemáticas de trabajar[14]. La estructura básica se entiende como:

- Análisis: Comprende la elaboración de la lista de requisitos y de especificaciones. Se busca el entendimiento del problema para resolver la forma de abordarlo y cuales podrían ser los puntos más relevantes.
- Síntesis: Involucra la identificación de los elementos de diseño que formarán parte del producto, su descomposición en partes y la combinación de la solución de cada elemento en un sistema completo funcional.
- Evaluación: Examinar la eficacia con la cual los diseños alternativos satisfacen los requisitos de las especificaciones técnicas para la operación, manufactura y ventas, antes de seleccionar el diseño final.

5. Diseño Conceptual

El planteamiento de la necesidad y definición del problema, necesarios para comenzar este proceso de diseño, son aquellos expuestos en la sección I donde se define el objetivo general de esta tesis, y en la Sección 3.3, donde se presenta la necesidad junto con algunas soluciones propuestas por distintas empresas.

5.1. Requerimientos y especificaciones

Los requerimientos del cliente pueden tener diferente valor para cada persona. En el desarrollo es importante identificar los requerimientos que son clave y satisfacer las necesidades planteadas para lograr que el producto sea un éxito. Es posible que sea una difícil distinción de realizar para algunos miembros del equipo de diseño ya que desde el punto de vista ingenieril se busca obtener el mejor desempeño posible en todos los aspectos del producto[12].

Los requerimientos se pueden clasificar de la siguiente manera:

- Básicos: Son características del producto que se consideran obligatorias. No aumentan la satisfacción, pero causan una gran insatisfacción si no se aportan.
- De desempeño: Estas características del producto aumentan proporcionalmente la satisfacción. Cuantas más se añaden o más funcionalidades ofrecen, mejor.
- De deleite: Son características no esperadas y que causan una gran satisfacción. Como no son esperadas, no provocan insatisfacción si no se añaden[15].

Para el sistema a desarrollar se realizó una lista de requerimientos que se agruparon y evaluaron en la Tabla [5.1].

Jerarquía de requerimientos		
Requerimiento		Importancia
Tipo	Descripción	10-1
Desempeño	Una respuesta exacta, que el sistema responda lo más cercano al valor esperado.	10
	Una respuesta precisa, que el sistema proporcione respuestas consistentes ante los valores de entrada.	10
	Alta capacidad de procesamiento.	9
	Velocidad de la respuesta.	9
	Uso durante un tiempo prolongado.	8
Mantenimiento	Elementos fácilmente remplazables.	8
	Con escaso o nulo mantenimiento.	5
	Fácil desmontaje para remplazo, ajustes y limpieza.	8
	Simplificación del diseño con el menor número de componentes.	4
Seguridad	Protecciones eléctricas contra sobre cargas y cortos circuitos.	6
	Indicadores de correcto funcionamiento: leds, alarmas etc.	5
	Interruptor de apagado de emergencia.	6
	Resistente a vibraciones y golpes.	8
	Disipación del calor de componentes electrónicos.	5
	Operación a distancia.	8
Estético	Silencioso.	2
	Compacto con elementos adecuadamente integrados.	3
	Dimensiones reducidas.	3
	Peso reducido (dispositivo general).	4
Operación	Interfaz intuitiva.	7
	Sistema claro de despliegue de información.	8
	Sistema de recopilación de información para análisis.	9

Tabla 5.1: *Ponderación y agrupación de requerimientos*

Por otro lado, Ulrich (2013)[16] menciona que las especificaciones surgen de la adaptación de las necesidades de los clientes, las cuales suelen ser un tanto ambiguas y subjetivas. Por esta razón, los grupos de desarrollo por lo general establecen un conjunto de especificaciones que explican, con detalles precisos y medibles, lo que el producto tiene que ser, hacer y tener. Estas no indican cómo manejar las necesidades del cliente, pero representan una base sobre lo que se deberá hacer para satisfacerlas.

Para poder obtener las especificaciones se necesitan traducir los requerimientos previamente listados. En la Tabla [5.2] se observan las especificaciones inferidas.

Especificaciones de diseño		
Especificación	**Descripción**	**Unidades (Referencia)**
Objetivos de control	El dispositivo logra los objetivos de estabilización y regulación mediante los controles implementados. (1 punto por objetivo)	2/2 puntos
Velocidad de procesamiento	Frecuencia de funcionamiento del procesador	16 [Mhz]
Tiempo de respuesta a solicitud (Latencia)	El tiempo que transcurre entre que el usuario envía el comando desde una interfaz hasta que se genera una respuesta.	100 [ms]
Tiempo de uso continuo	Tiempo que el dispositivo puede permanecer encendido realizando una prueba sin interrupción.	45 [min]
Dimensiones máximas del dispositivo	Medidas generales del dispositivo propuestas para su portabilidad.	60x60x60 [cm]
Masa total del dispositivo	Masa total del dispositivo que posibilite a una persona trasladarlo y manipularlo sin dificultad.	5 [Kg]
Masa de la estructura móvil	Carga móvil que levantan los motores asociada a la fuerza de empuje mínima requerida por los actuadores.	0.7 [Kg]
Tiempo de armado	Tiempo promedio de ensamble de todos los componentes y ajuste del dispositivo.	5 [hr]
Componentes intercambiables	Razón entre el número de componentes fácilmente remplazables y el número de componentes totales. (escasos, costosos o que se encuentren soldados a otros componentes no son de fácil remplazo)	0.7/1.0
Respuesta a perturbaciones	El dispositivo regresa a la posición deseada tras sufrir perturbaciones externas: cambios de referencia, variaciones de carga y ruido. (1 punto por cada una)	3 puntos
Puntaje de operación	De acuerdo al artículo *Evaluación de interfaces graficas de usuario* [114] se dará un puntaje a la interfaz tomando en cuenta los criterios que allí se mencionan.	7/10 puntos
Distancia de operación	Funciona inalámbricamente a una distancia segura para el usuario.	1-5 [m]
Protecciones eléctricas (Las más comunes)[137]	Protección contra altas temperaturas (OTP). La alimentación se interrumpe tras superar un límite.	70 [°C]
	Protección de sobrecarga (OPP). Interrupción de la energía al detectar un porcentaje de potencia elevado.	130-150%
	Protección de cortocircuito (SCP). Corte de energía cuando la corriente se eleva repentinamente.	si
	Protección de sobrecorriente (OCP). Desconexión ante una demanda de corriente mayor a la indicada.	130-150%
	Protección de alto voltaje (OVP). Corte de energía tras superar el límite de voltaje nominal en la salida de la alimentación.	110-130%
Nivel de ruido máximo	El ruido generado por los motores debe permanecer en valores inferiores limite según la NOM-081-ECOL-1994.	<68 [dB]

Tabla 5.2: *Definición de especificaciones de diseño*

5.2. Árbol de objetivos

El *árbol de objetivo* es una herramienta que forma parte de la descomposición del sistema y ayuda a definir sus características. Un *objetivo* describe el motivo de un sistema o alguna de sus partes para lograr una meta. El éxito de estos depende de las funciones que se plantea que debe realizar el sistema, es por ello por lo que es importante definirlos primero, [17].

En un diagrama de árbol, los objetivos se agrupan en orden jerárquico, de mayor a menor nivel. Se puede interpretar de tal forma que al leerlo en orden descendente se indican los medios para conseguir el fin superior, y en orden ascendente se leen los motivos. Dicho de otra manera, si el lector se desplaza hacia abajo encontraría el cómo lograrlo y hacia arriba el para qué se necesita.

Estos objetivos generalmente se recaban de la lista de requerimientos planteada previamente y se discuten entre el equipo de diseño. Se añaden de acuerdo a cuáles son imprescindibles para el funcionamiento y cuáles pueden ser propicios para mejorar el resultado. Estos objetivos pueden variar y ser modificados durante el proceso y mejorar el entendimiento del problema [15].

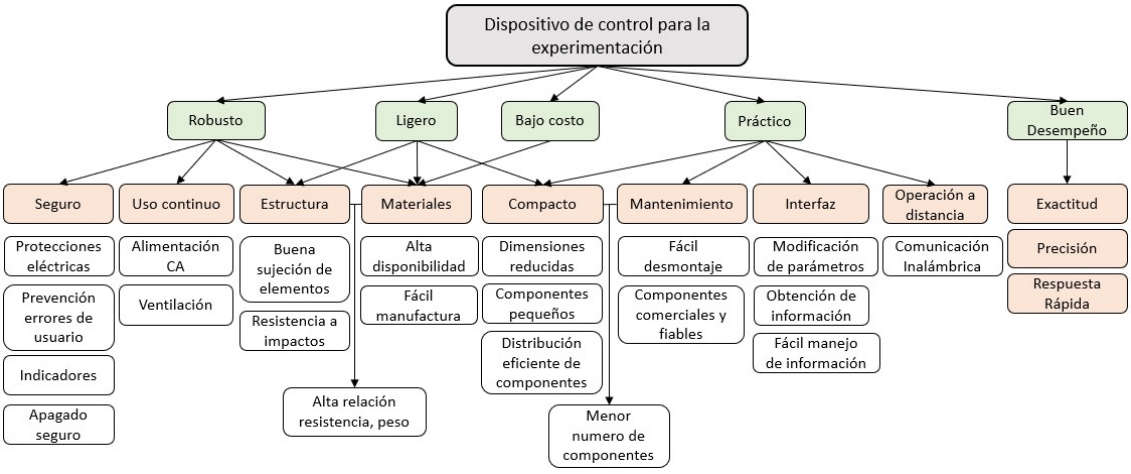


Figura 5.1: *Árbol de objetivos generales del sistema*

Para este trabajo los objetivos se clasificaron por jerarquías en la Figura [5.1]: los principales se encuentran resaltados en color verde, lo secundarios en color naranja y el resto en blanco como medios para alcanzar los primeros.

5.3. Descomposición funcional

Durante el proceso de diseño se utilizan distintas técnicas auxiliares en el desarrollo del concepto. Del mismo modo que en el desglose de objetivos expuesto anteriormente, podemos describir el sistema por medio de entradas, salidas y subprocesos involucrados, además de señalar sus relaciones.

5.3.1. Caja negra

Para comenzar con esta técnica de descomposición, es preciso definir el *modelo de caja negra* como una representación simplificada del sistema real que ayuda al análisis de la relación causa-efecto de las variables de entrada y salida. Se utiliza cuando no se tiene suficiente información del dispositivo ya que se plantea sin necesidad de conocer las relaciones entre los diferentes componentes internos del sistema ni su funcionamiento, es decir, si alguna parte de éste tiene cambios significativos no es posible determinar con precisión cuales fueron esas variaciones [18].

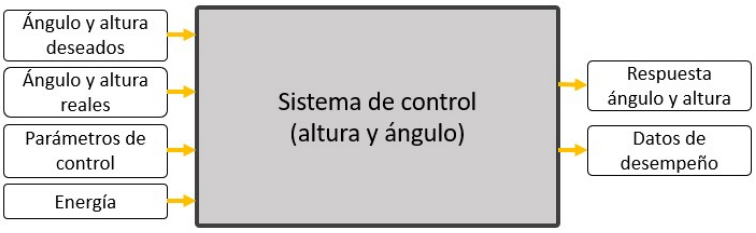


Figura 5.2: Modelo de caja negra dispositivo de control VTOL

En el modelo [5.2] se toman como entradas las posiciones, tanto las esperadas o deseadas como las actuales, que son recuperadas por medio de algún subsistema que hasta el momento es desconocido. Adicionalmente, se ingresan algunos parámetros que establece el usuario para ajustar el comportamiento del controlador u otro tipo de instrucciones. Finalmente se toma en consideración la energía consumida cuando se encuentra en funcionamiento.

Las salidas corresponden a las alteraciones de la posición angular y vertical como consecuencia del control. Para la experimentación y análisis se necesita entregar la información de las pruebas hechas, preferiblemente un conjunto de datos desplegados de forma visual para interpretarlos directamente.

5.3.2. Análisis de funciones

Una vez que se ha definido el sistema a grandes rasgos, se continúa con su especificación a un menor nivel. Un diagrama de funciones, también conocido como *modelo de caja transparente*, da una representación más detallada. Este presenta cuáles son las funciones más relevantes y sus interconexiones, independientemente de los componentes físicos que se utilicen. La función global es descompuesta en subfunciones enfocadas en cumplir los objetivos previamente planteados y estas, a su vez, se pueden separar en otras todavía más específicas. Cada una de ellas debe estar dentro de los límites del sistema global y debe tener conectadas todas sus entradas y salidas, con excepción de aquellas que provienen o se dirigen fuera de este[15]. Con el propósito de ordenar a estas subfunciones, se pueden agrupar en diferentes conjuntos de acuerdo con ubicación física, campo semántico, objetivo que cumplen, etc.

El fin de llevar a cabo este proceso, es encontrar los componentes apropiados que satisfagan mejor cada función implicada. La Figura [5.3], muestra la descomposición del sistema en cuestión en funciones que a su vez contienen a otras mas especificas.

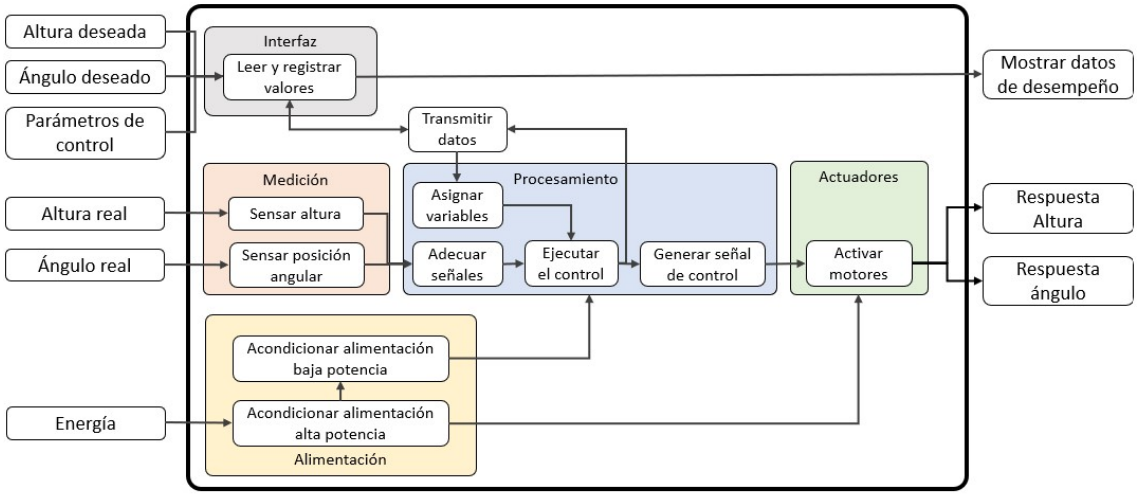


Figura 5.3: *Diagrama de funciones del dispositivo de prueba VTOL*

La primera función, en color naranja, involucra el registro de las posiciones actuales de forma periódica que servirán en otra función para la implementación de un control de lazo cerrado. Por las características del proyecto se necesitan dos dispositivos diferentes, uno para registrar la altura y otro para medir la inclinación respecto a la horizontal. Es imprescindible que los sensores tomen medidas con una alta frecuencia de muestreo, que los valores recabados sean precisos, exactos y en medida de lo posible, que su funcionamiento no sea muy susceptible a las vibraciones y otras perturbaciones.

Para que el usuario pueda tener control directo sobre las acciones del dispositivo hace falta un medio que permita modificar los valores de las constantes preestablecidas en el controlador, así como las posiciones deseadas y otras instrucciones básicas como variar la velocidad de los motores o pedir información de los sensores. Esta función color gris, en primera instancia, deberá realizar la lectura de instrucciones y posteriormente enviarlas a donde se esté ejecutando el procesamiento para su asignación. De igual manera a como los datos son ingresados, la interfaz recibirá la señal de control y los estados de la planta para que el usuario pueda visualizarlos y posea el registro del comportamiento.

La comunicación entre la interfaz y el dispositivo encargado de realizar el control es muy relevante. La transferencia de información se debe mantener en ambos sentidos para que el usuario de una interpretación en tiempo real de la respuesta. Esta función hace referencia al hardware y a la programación de los protocolos de comunicación entre emisor y receptor. El intercambio de información en tiempo real tiene un alto consumo de recursos como la memoria y el procesamiento, se desea que sea rápido y con la menor pérdida de información posible. La velocidad depende del tipo y cantidad de datos que sean enviados, así como las capacidades de cada protocolo.

Tras la recepción e interpretación de la información proveniente desde la interfaz, en la función de procesamiento, se asignarán los valores a las variables correspondientes para modificar los preestablecidos. Por otra parte, las medidas serán sometidas a filtros, que son algoritmos implementados con el fin de adecuar una señal al reducir el ruido y otro tipo de perturbaciones que se encuentren fuera de rango y así proporcionar datos de manera uniforme hacia el siguiente proceso que se ejecutará cíclicamente admitiendo nuevos datos en cada iteración y evaluando las funciones de control para generar una respuesta apropiada tomando en cuenta los valores deseados, la cual será enviada a los actuadores. Además, los valores relevantes de cada iteración se enviarán de vuelta hacia la interfaz a través de la comunicación.

Los actuadores se encargarán de alterar los estados de la planta de forma física, debido a que son capaces de ejecutar la respuesta del control previamente procesada. De este modo se completa el ciclo de funcionamiento, ya que serán estos mismos cambios los recabados en el proceso de medición.

La energía que consumen los componentes del dispositivo se contempla en una función de alimentación, la cual tiene como propósito acondicionar la entrada de energía y dividir el suministro de corriente a cada componente según su nivel de consumo. Dentro de la función se considera una línea de baja potencia para satisfacer la demanda de los circuitos electrónicos, incluyendo a los sensores y el procesamiento. También es necesaria otra línea de alta potencia que permita energizar los actuadores y otros elementos de mayor consumo.

5.4. Configuración solución

A fin de encaminar la selección de un concepto final, es importante realizar la asignación de los elementos funcionales y de construcción al aparato. Esta sección tiene como propósito definir los componentes físicos en términos de lo que hacen y sus conexiones con el resto del dispositivo.

De acuerdo con Ulrich y Steven(2013)[16], un producto puede considerarse en términos funcionales y físicos. Los elementos funcionales son las operaciones y transformaciones individuales que contribuyen al rendimiento general; regularmente se describen en forma esquemática antes de ser reducidos a tecnologías específicas, componentes o principios de trabajo. Un elemento físico hace referencia a las partes y componentes que ponen en práctica las funciones del producto y se definen mejor a medida que avanza el desarrollo. Algunos de ellos son dictados por el concepto del producto y otros se definen durante la fase del diseño de detalles. Un elemento físico puede desempeñar una o varias funciones, aunque también puede ocurrir que

varias de ellas se lleven a cabo en uno solo.

Los componentes de un sistema están organizados de manera específica en varios grupos de construcción más grandes llamados trozos, los cuales definen la relación de los elementos funcionales dentro de un esquema al que llamamos arquitectura de producto.

Durante el desarrollo del trabajo se contemplaron varias opciones de configuración que fueron evaluadas siguiendo los criterios de diseño, que a su vez contemplan los requerimientos, posibilidades y alcances del equipo de desarrollo. Con base en ello se llegó al concepto de la Figura [5.4]. Este esquema representa la interacción y distribución de los componentes en el dispositivo.

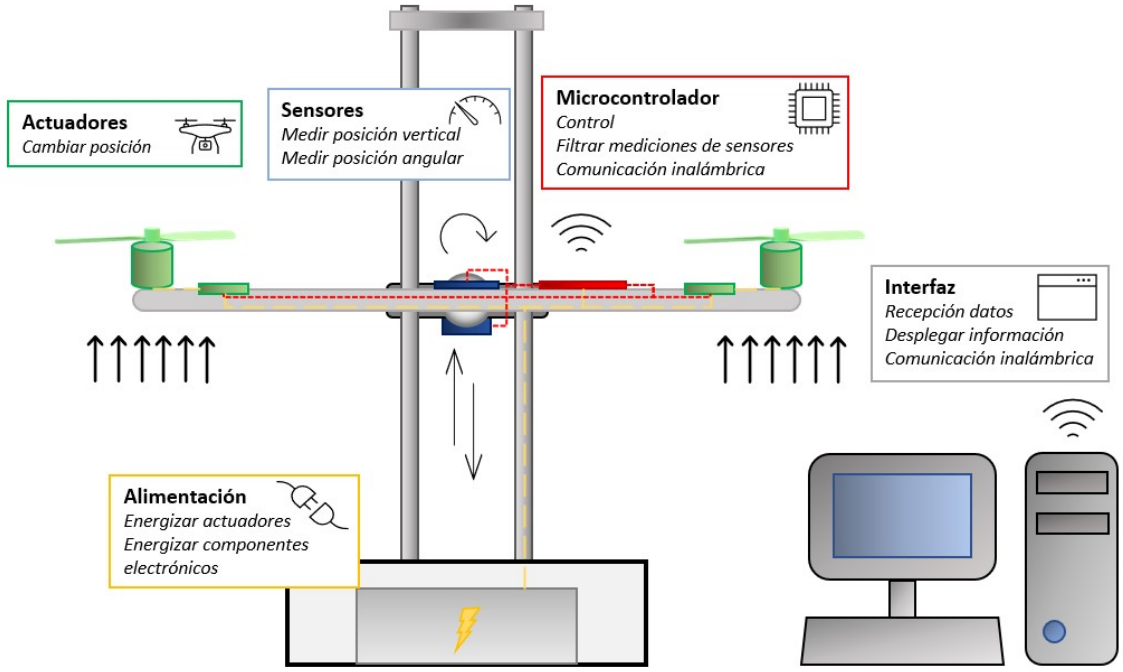


Figura 5.4: *Arquitectura seleccionada de dispositivo VTOL*

La presente solución considera algunas características, por ejemplo: la comunicación inalámbrica como Bluetooth y Wifi, la posición adecuada de elementos como los sensores mejora las condiciones de medición, la posición de los actuadores y sus controladores ayuda a distribuir la masa en los extremos del balancín, una estructura que consta de cuatro postes proporciona una mayor estabilidad y las dimensiones reducidas se logran al integrar los componentes en un menor espacio. Con el motivo de reducir el número de procesos del microcontrolador y ocupar menos memoria flash para el almacenamiento del programa se plantea desarrollar la interfaz en una computadora, además esto permite añadir mayores funciones para el almacenamiento de datos y la graficación.

6. Delimitación y composición de la solución.

Al finalizar la fase conceptual se entra progresivamente en el detallado explícito de cada subsistema del concepto seleccionado en la configuración; se consideran sus componentes, conexiones, funcionamiento, características y condiciones particulares. Cuando se alcanza esta etapa de diseño, es necesario tener un entendimiento sobre temas específicos como materiales, procesos productivos, técnicas de análisis, nuevas tecnologías del sector, entorno del componente, estética, etc. La mayor parte de las actividades que se realizan durante el diseño corresponden a esta fase y en ella juegan un papel importante factores como disponibilidad, costo, facilidad de fabricación, tiempo, etc [19].

6.1. Descripción y selección de componentes

Cabe aclarar que la arquitectura del prototipo seleccionada en la Figura [5.4], considera la implementación de un control tipo PID y otro en variables de estado, así que las capacidades del hardware seleccionado deberán ajustarse a las exigencias que pueden demandar de este tipo de controles, por ejemplo el periodo de muestreo necesario que se define en la sección 11.5.

Dado que, en este punto no se han diseñado los controladores, las decisiones que se tomaron acerca de ciertas características de los componentes, como la frecuencia de trabajo del microcontrolador, se basan en conocimiento previo acerca de la implementación de un controlador PID sobre un sistema similar. A causa de esto algunos componentes pueden tener prestaciones un tanto mayores a las requeridas posterior a la implementación de los controladores.

6.1.1. Actuadores: Motor brushless

En las áreas de aeromodelismo, construcción de UAVs y la robótica se está volviendo cada vez más común el uso de motores Brushless o BLDC. Estos son motores síncronos de corriente directa cuya principal diferencia con los de DC convencionales es que carecen de escobillas.

Los motores brushless están formados por dos componentes principales: El *estator*, que es formalmente definido como el conjunto de piezas dentro de la coraza que no desarrollan movimiento, aunque muchas veces se hace referencia solo al conjunto de embobinados fijos, y el *rotor*, compuesto por todas las piezas que durante el funcionamiento desarrollan un movimiento rotacional. Los BLDC cuentan con una coraza o armadura como protección y medio de disipación de calor, un eje y rodamientos para minimizar la fricción, además, el estator tiene un arreglo de imanes permanentes.

En este tipo de motores es común encontrar dos configuraciones. Los primeros son los *inrunners*, que son aquellos donde el rotor se encuentra en el centro de la misma forma que los motores más comunes, esta configuración se caracteriza por alcanzar mayores velocidades. El segundo tipo son los *outrunners*, donde el rotor se encuentra en la parte exterior y abarca una mayor área mientras que el estator está en el centro, en este caso toda la armadura del exterior gira junto con el eje. Estos motores son más robustos ante variaciones en la carga ya que su armadura funciona como un volante de inercia, sin embargo, requieren más energía para arrancar y desarrollan velocidades menores.

Las características de voltaje y par de torsión que entrega un Brushless son similares a las de un DC y por ello en muchas ocasiones, con propósitos de análisis y diseño, se considera como un motor DC más eficiente. Sin embargo, su funcionamiento no es exactamente igual. La mayoría de estos motores cuentan con tres embobinados diferentes en el estator; cada uno de estos puede estar compuesto de una o más bobinas en serie o paralelo (dependiendo el modelo y número de polos) y son colocados en ranuras unas frente a otras desfasadas 120° entre ellas. Si a uno de ellos se le aplica un voltaje, la corriente circulará a través de las espiras y se creará un campo magnético el cual interactuará con el generado por los imanes permanentes que se encuentran en el rotor atrayéndolos o repeliéndolos según la polaridad generada; esto produce un par

[2]Imagen editada a partir de:
`https://www.rotor-magazin.com/wp-content/uploads/2018/08/basis_motoren.jpg`
`https://1.bp.blogspot.com/-_av9Q9foEKg/W3t_hr5PF_I/AAAAAAAAKAA/1-c-X6L2Kec90MKWLf_X00Arz6mcRsWBQCLcBGAs/s400/mak.jpg` (27/abril/2021)

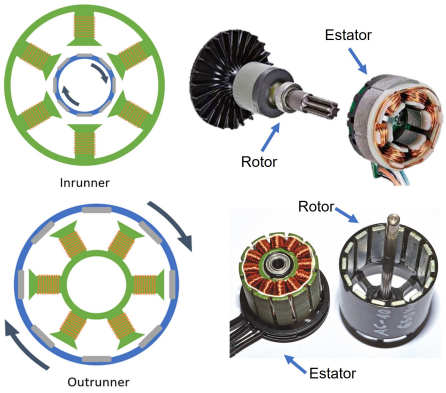


Figura 6.1: *Configuraciones de los motores BLDC* [2]

de fuerzas que permite que el motor comience a girar hasta alinearse con la bobina energizada. Dicho par se incrementa si se genera una polaridad contraria en la bobina anterior y se suman ambos efectos[20]. La calidad y fuerza de los imanes influye directamente en la potencia que puede desarrollar.

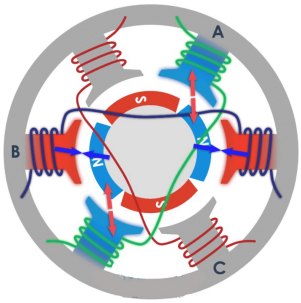


Figura 6.2: *Fuerzas que interactúan en un motor BLDC* [3]

Para que existan estas polaridades opuestas sin tener que energizar las bobinas independientemente, se interconectan los extremos de las bobinas. Esta conexión puede ser de dos tipos: el primero es Delta (Δ), donde las bobinas se conectan en serie y la tensión es aplicada en los nodos resultantes produciéndose una mayor velocidad; al segundo se le conoce como estrella (Y) donde la conexión se realiza en paralelo con un nodo central y la tensión se aplica en cada una de las terminales libres; esta favorece el torque bajo las mismas condiciones de voltaje y corriente. Lo más común es utilizar estrella para reducir la corriente necesaria y generar el par deseado además de reducir las corrientes parásitas ya que, al no contar con lazos cerrados, no propicia que se generen[21].

Si al tener una tensión en cualquiera de las terminales se puede generar un campo magnético que produce una fuerza, entonces podemos aplicar los pulsos de forma secuencial en las terminales para variar la dirección del campo magnético. Si la secuencia es la correcta se formará un campo magnético rotacional el cuál, junto con los imanes, es responsable de mantener el movimiento del motor de forma continua y síncrona [22].

Una de las principales características que se utiliza para clasificar este tipo de motores es su factor (Kv). Este es propio de cada motor y depende principalmente del número de embobinados, tamaño y numero de espiras. De forma simple este valor representa la velocidad (en rpm) que alcanza el rotor sin carga cuando se aplica 1[V]. Por ejemplo, si un motor indica 1200[Kv] significa que con una entrada de 2[V] su velocidad

[3] `https://howtomechatronics.com/how-it-works/how-brushless-motor-and-esc-work/` (27/Abril/2021)

[4] `https://www.digikey.com/-/media/Images/Article%20Library/TechZone%20Articles/2016/December/How%20to%20Power%20and%20Control%20Brushless%20DC%20Motors/article-2016december-how-to-power-and-fig3.jpg?ts=c302f01a-d22e-4cc1-b2ae-09acfa462ecb&la=en-US` (27/Abril/2021)

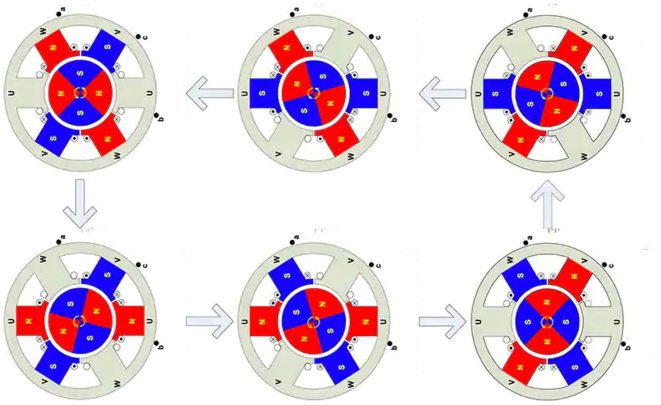


Figura 6.3: *Secuencia de funcionamiento de los motores BLDC* [4]

máxima será de 2400[rpm]. Entre más alto sea este valor, el motor será mucho más rápido y por lo tanto perfecto para aplicaciones como drones de carreras. Si el valor es bajo significa que el motor no es capaz de alcanzar grandes velocidades, pero produce un mayor par, ideal para aplicaciones donde se piensa mover una propela de gran tamaño o cargas considerables.

En comparación con otros motores DC los BLDC proporcionan la misma potencia pero con un diseño más compacto. Al no tener escobillas la energía que se pierde por fricción se reduce significativamente, además, como las corrientes son suministradas directamente al estator y no hay otras inducidas en el rotor, se reducen las pérdidas por calor debido al efecto Joule; estos aspectos lo convierten en un motor con mayor eficiencia de hasta 90 %, también, reducen la necesidad de mantenimiento continuo y tienen un mayor tiempo de vida útil.

Su capacidad de disipar calor y reducir pérdidas les permite trabajar con corrientes más altas y desempeñar labores que requieran de un mayor par. No requieren de un voltaje muy grande para llegar a grandes velocidades. Algunos modelos pueden alcanzar sin problema las 100,000 [rpm] con una relación par-velocidad relativamente lineal en un rango muy amplio.

Al no tener conmutación mecánica estos motores no producen chispas, por ello son ideales para aplicaciones donde esto pueda significar un riesgo. Hasta la fecha, los Brushless se utilizan en situaciones que requieren menos de 5[KW] ya que para potencias mayores los costos de los imanes se elevan progresivamente hasta volverse demasiado caros, es por ello que no es común verlos en maquinaria pesada. No son recomendables para aplicaciones donde estén expuestos a altas temperaturas ya que se pueden desmagnetizar los imanes (temperaturas por arriba de la de Curie). Siempre hay que tener en cuenta que la conmutación electrónica es más compleja y puede aumentar el costo [23].

Motor RacerStar BR2212 1400Kv

Para el desarrollo de este proyecto se utilizaron los motores brushless RacerStar BR2221 1400Kv[24], los cuales cuentan con las siguientes especificaciones:

- Factor Kv: 1400 [rpm/V]
- Voltaje de operación: 7.4 - 14.8 [V]
- Potencia: 210 [W]
- Dimensiones: (diámetro x altura) Ø27.7 x 40 [mm]
- Masa: 52[g]

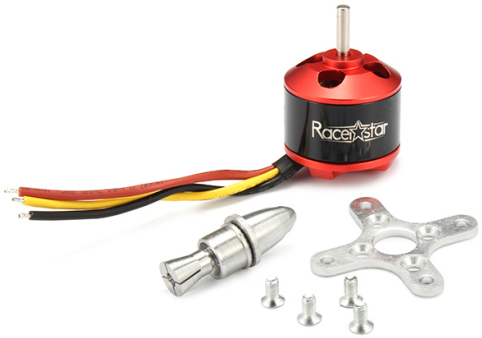


Figura 6.4: *Motor seleccionado: RacerStar BR2212 1400Kv* [5]

Este modelo es de tipo *Outrunner*, compatible con baterías LiPo de 2,3 y 4 celdas. El valor relativamente bajo de KV dice que es un motor más enfocado al par que proporciona que a la velocidad, ideal para nuestra aplicación donde solo se utilizan dos motores para elevar toda la estructura. Su adaptador de bala con cuerda inglesa permite acoplar numerosas propelas comerciales. Sin embargo, el fabricante recomienda utilizar hélices de entre 6 a 10 pulgadas como límite.

6.1.2. Variador de velocidad ESC

Para que el funcionamiento de los BLDC sea el correcto se deben aplicar tensiones en los embobinados en una secuencia específica y momento preciso. Algunos motores utilizan un conmutador mecánico para lograr esta tarea, pero este tipo utiliza uno electrónico, por esta razón también se les conoce como *motores conmutados electrónicamente*. Esta conmutación se logra mediante los *ESC* (Electronic speed control), que son los encargados de regular la velocidad y dirección del motor. En términos simples, los ESC son elementos que regulan al motor a partir de una señal PWM o PPM de control, esta normalmente es de 50[Hz] con un ancho de pulso que oscila entre 1[ms] y 2[ms]. El ESC varía la velocidad del motor de una forma aproximadamente lineal, es decir que en 1[ms] el motor está detenido y en 2[ms] alcanza su máxima velocidad.

Un ESC, como se puede observar en la Figura [6.5], cuenta con un inversor de seis pasos para generar las tres fases discretas necesarias para obtener el campo magnético rotacional. Dicho de otra forma, utiliza un arreglo de seis switches electrónicos (comúnmente MOSFET) para conseguir dichas fases estos dejan pasar la corriente de una fuente DC que proporciona la potencia, de tal manera que dos de los tres embobinados del motor están energizados en todo momento, esto significa que solo dos de los seis switches conducen en cualquier instante[20].

Para establecer el momento en el que se debe dar esta conmutación electrónica el ESC debe conocer la posición del motor y así asegurarse que los switches se activan en el momento justo. Algunas configuraciones utilizan sensores de efecto hall o encoders para realizar esta función y se les conoce como *sensored BLDC*. También se puede conocer la posición si se monitorea la corriente inducida en la bobina que no es energizada, estos se llaman *sensorless BLDC*[23].

La velocidad de rotación del campo magnético es directamente proporcional a la frecuencia de los pulsos en la señal de entrada. Al cambiar de una frecuencia fija a otra, el comportamiento es similar a un sistema de lazo abierto y la velocidad se ajusta acordemente.

[5] `https://www.racerstar.com/racerstar-br2212-1400kv-2-4s-brushless-motor-for-rc-models-p-54.html` (24/Marzo/2022)

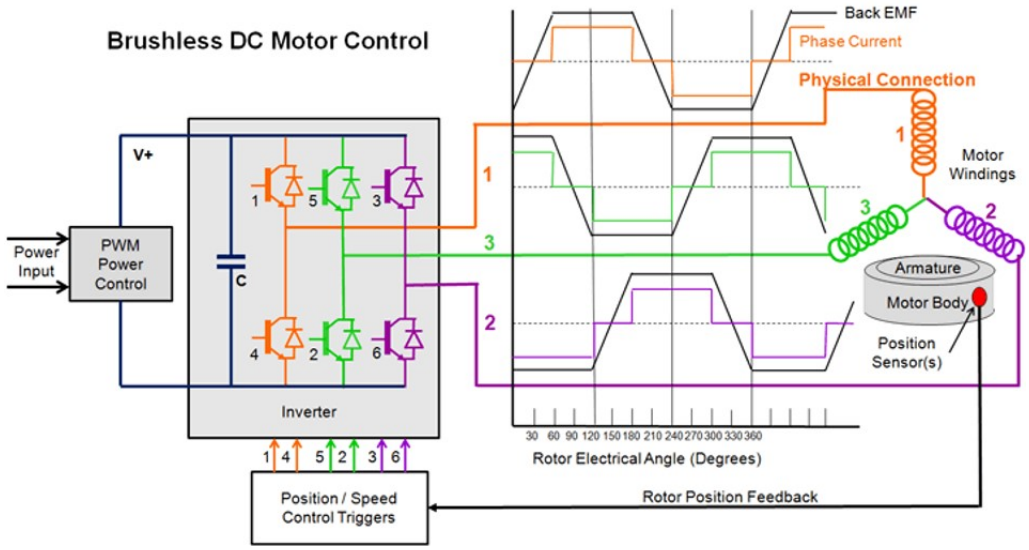


Figura 6.5: *Diagrama de funcionamiento del ESC* [6]

La característica más importante para seleccionar un ESC es la capacidad que tiene de suministrar corriente, lo cual depende de los motores que se van a utilizar y la tarea que estos van a realizar. Algunas cuentan además con un circuito *BEC* (Battery Eliminator Circuit) para alimentar periféricos sin necesidad de baterías extra.

Señal PWM y PPM

Como se mencionó anteriormente, la señal de entrada necesaria para controlar la velocidad del motor BLDC mediante un ESC debe ser una señal, ya sea PWM o PPM, con características especiales.

El PWM (Pulse Width Modulation) es una señal de voltaje ampliamente utilizada en circuitos digitales para emular una analógica. La modulación de ancho de pulso está formada por una onda cuadrada que no siempre tiene la misma relación entre el tiempo que está en un estado alto y bajo, es decir, que la señal en alto cambia el ancho relativo respecto al período de esta. A esta relación se le conoce como *ciclo de trabajo* (duty) y se representa en porcentajes. En esta técnica de modulación permanecen constantes la amplitud de los pulsos y la frecuencia de la señal.

$$D = \frac{t}{T} \tag{6.1}$$

$D =$ ciclo de trabajo
$t =$ tiempo en alto de la señal
$T =$ periodo

Para lograr emular una señal analógica se altera el ciclo de trabajo de tal manera que el valor promedio se aproxime al voltaje que se desea obtener de la original, siendo el rango de tensión que se puede aplicar entre 0[V] y la amplitud de pulso que pueda soportar el dispositivo PWM utilizado[25].

El PPM (Pulse Position Modulation), al igual que el PWM, es una señal de tensión de onda cuadrada que emula a una señal análoga y que también puede servir para controlar los ESC. Consiste en la generación

[6] `https://www.mpoweruk.com/motorsbrushless.htm` (27/Abril/2021)

de pulsos que varían en su posición dentro de un periodo definido respecto a una referencia en función del valor de una señal analógica, de tal manera que el mínimo desplazamiento indicará el valor mínimo de la señal analógica y viceversa. Por lo tanto, el cambio de posición es proporcional a la señal moduladora. Este método implica que la frecuencia, amplitud y ancho del pulso permanecen constantes. El ancho del pulso se determina en consideración de ser lo suficiente amplio para ser detectado por el dispositivo y lo suficiente corto para que pueda tener un mayor desplazamiento dentro del periodo.

La señal modulada por posición de pulso también puede utilizarse para transmitir información de varios canales; en este caso se compone de tramas de pulsos que se envían periódicamente, de tal manera que cada pulso contiene información según su duración y su posición en la trama. Cada canal se identifica como la posición del pulso dentro de la trama y siempre corresponde al mismo dato. El emisor y el receptor deben estar sincronizados para que el dato de un canal sea recibido en el mismo orden cuando llega la señal; en este caso también es importante contar con un tiempo suficiente para que se pueda diferenciar una trama de otra lo que permitirá sincronizar los dispositivos. Su duración será la diferencia entre la duración de la trama y el periodo de la señal[26].

En la figura [6.6] se observa la comparativa entre la señal moduladora en rojo, el PPM en verde y el PWM en azul.

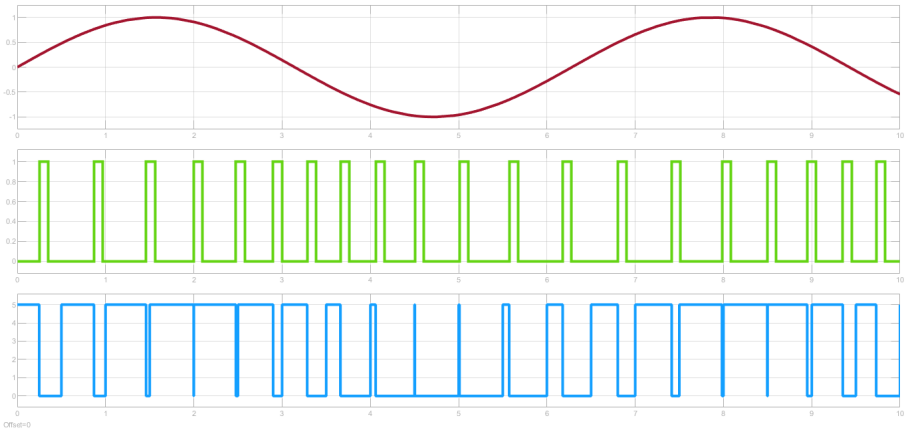


Figura 6.6: *Comparación entre la señal analógica, PWM y PPM.* [7]

ESC BLDC 30A

Para la etapa de potencia de los motores BLDC seleccionamos los variadores de velocidad ESC BLDC 30A[27]. Este modelo es común por su amplia gama de usos y disponibilidad. Cuenta con las siguientes características:

- Corriente: 30[A] (max 40[A] por 10 segundos)
- Voltaje de entrada: 7.4 - 14.8[V] (2s-4s LiPo)
- BEC de 5[V], 3[A] para alimentar periféricos
- Masa: 32[g]
- Tamaño: 55 x 26 x 13[mm]
- Frecuencia PWM: 50 - 60 [Hz]

[7] `https://medium.com/modulaciones-de-pulsos-muestreo-pam-ppm<-pcm-y/modulaciones-de-pulsos-ppm-558c91689e7` (30/mayo/2022)

Figura 6.7: *Variador seleccionado: ESC BLDC 30A* [8]

Esta ESC utiliza MOSFETs de canal N para llevar a cabo la conmutación y cuenta con un circuito regulador de voltaje independiente para alimentar periféricos tales como el microcontrolador, lo que resulta conveniente para nuestra aplicación. La señal de entrada debe ser PWM con una amplitud de 3.3[V] o 5[V] y un pulso cada 20[ms] que modifica su ancho desde 1[ms] a 2[ms] para mínima y máxima velocidad respectivamente. Además, es posible programar ciertas características de comportamiento mediante variaciones de la señal de entrada y los tonos que emite al estar conectada con el motor, por ejemplo, es posible modificar el tipo de arranque, el frenado activo, los límites de protección de bajo voltaje y alta temperatura entre otros. Para reducir el consumo de corriente, se apagó el frenado activo y se utilizó el modo de arranque suave.

6.1.3. Propelas

Para que cualquier aeronave se mantenga en el aire debe de tener una fuerza que la impulse hacia arriba y que soporte como mínimo su propio peso; a esta se le conoce como *fuerza de sustentación*. Un modo de producirla es mediante el uso de hélices, también conocidas como propelas. Este elemento mecánico está formado por dos o más *perfiles aerodinámicos giratorios*, que son objetos planos colocados con un poco de inclinación y rotan sobre un eje céntrico.

Cuando las hélices giran dentro de un fluido estas lo desplazan y generan una diferencia de presión entre la superficie superior e inferior; esto produce un empuje hidrostático perpendicular a la superficie de las propelas, la cual se puede usar como fuerza de sustentación.

Tal como mencionan Fernández, et al. (2016)[21] y algunos artículos de referencia para compra de drones[28], la magnitud de la fuerza producida depende de la velocidad de giro de las propelas, del par del motor que las impulsa y sus características geométricas. La primera característica es el diámetro que se mide de punta a punta de las palas (también conocidos como alabes) y generalmente se da en pulgadas. Cuando el área es mayor es capaz de desplazar un más volumen del fluido, en este caso aire, lo cual lleva a incrementar la fuerza de sustentación con una menor velocidad de rotación, sin embargo, esto requiere mayor potencia. En el caso de propelas con un diámetro pequeño se requiere más velocidad para lograr una sustentación suficiente pero la potencia requerida es menor. Normalmente, se utilizan las hélices grandes en aplicaciones de carga y vuelos lentos, mientras que las pequeñas en aeronaves veloces y de menor tamaño.

[8] `https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjKjpWo6t_2AhWImmoFHTczA14QFnoECAgQAQ&url=https%3A%2F%2Fwww.optimusdigital.ro%2Findex.php%3Fcontroller%3Dattachment%26id_attachment%3D451&usg=AOvVaw0K8g8cGZgvGU7jrp-ABJkw` (25/Marzo/2022)

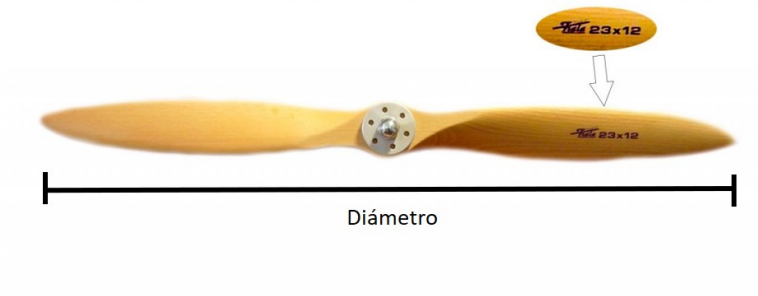


Figura 6.8: *Medidas estándar de una hélice* [9]

Otra característica geométrica importante de las hélices es el *paso*, Figura [6.9], que indica la inclinación de los perfiles respecto a su plano de giro; la mejor forma de verlo es como la distancia que avanzaría la hélice en un giro si se desplazara en un medio sólido, similar a un tornillo. Esta descripción es la que utilizan los fabricantes y por ello proporcionan el dato en pulgadas. El paso corto es recomendable para vuelos lentos y estables con bajo consumo de energía, mientras que los pasos largos se utilizan en vuelos rápidos.

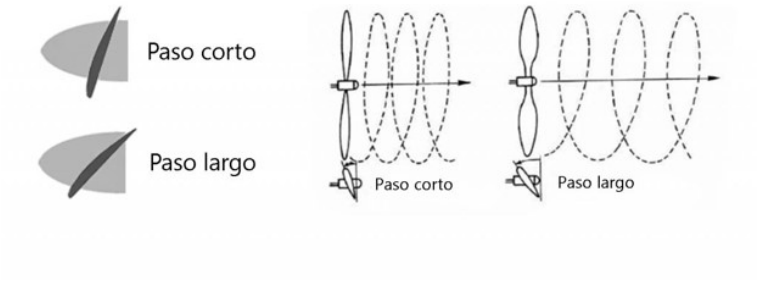


Figura 6.9: *Paso de una hélice* [10]

El acuerdo para designar las diferentes hélices dependiendo de estas dos características anteriores es mencionar primero el diámetro y después el paso en la misma hélice como se muestra en la Figura [6.8], siempre en pulgadas.

Aumentar el número de palas disminuye la eficiencia ya que cada una de ellas debe moverse en un medio turbulento. Dos álabes es el mínimo requerido para mantener el centro de masa alineado con el eje y disminuir el efecto indeseado de la turbulencia. Una regla práctica que se puede aplicar si desean aumentar el número de álabes es reducir el diámetro 1[in] y aumentar el paso 1[in] respecto a una hélice de dos álabes para la misma aplicación.

Además de la geometría, el material de la hélice es un factor que se debe tomar en cuenta. Pueden ser fabricadas con plásticos (comúnmente ABS), fibra de carbono, metales, madera y materiales compuestos. El material depende de la aplicación que se le quiera dar, se busca que sean ligeros, duros para soportar las fuerzas constantes a las que se encuentran sometidos además de minimizar las vibraciones, tenaces para resistir golpes sin llegar a fracturarse, resistentes a la fatiga ocasionada por las vibraciones y evitar superficies

[9] `https://www.comprardrones.online/wp-content/uploads/2017/08/dimensiones-de-una-helice-1024x478.jpg` (27/Abril/2021)

[10] `https://www.comprardrones.online/como-funcionan-las-helices-de-un-drone/` (27/Abril/2021)

porosas para minimizar las perdidas por fricción. El material elegido debe tener cierto equilibrio entre estas propiedades y el costo según la necesidad [29].

En el diseño de aeronaves, drones y UAVs que utilicen hélices es necesario tomar en cuenta el factor del par de motor o *Factor P*[28], este es un fenómeno que ocurre debido a la tercera ley de Newton. Para lograr la sustentación, la hélice debe generar una fuerza perpendicular al plano sobre el que esta está rotando, la inclinación de los álabes sirven para este propósito, pero esta misma inclinación produce fuerzas horizontales indeseadas, mostradas en [6.10], que generan un par y pueden provocar que el dron rote sobre sí mismo.

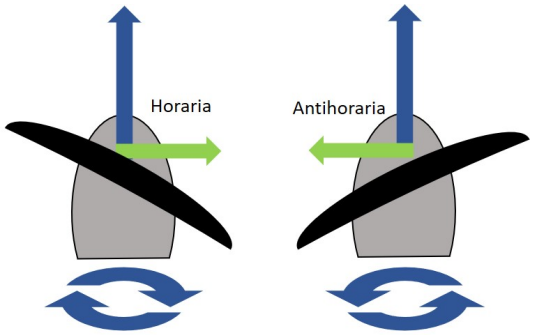


Figura 6.10: *Fuerzas horizontales provocadas por el factor P*

Algunos vehículos, como los helicópteros de una sola hélice, utilizan una propela secundaria más pequeña en la cola, perpendicular a la principal, para contrarrestar este efecto. También, se puede contrarrestar si se utilizan dos hélices con giro contrario, de esta manera generan un par opuesto y entre ellos se anulan mientras que las fuerzas verticales siempre van en la misma dirección. Por ello, también las hélices se clasifican como *Derechas (Horarias, dextrógiras o pushers)* e *izquierdas (antihorarias, levógiras o pullers)*.

Cálculo de la masa soportada

Uno de los puntos clave para el diseño del dispositivo es garantizar que el conjunto motor-hélice seleccionado sea suficiente para sostener y elevar el balancín, para ello es necesario calcular la masa máxima que soportan según sus características. El siguiente cálulo se basó y desarrolló a partir del trabajo de Fernández, et. al (2016)[21] y del artículo titulado *Impact of Jets*[30].

Primero se considera el diagrama de la Figura [6.11], que representa las fuerzas involucradas dentro de un volumen de control que rodea al motor y la hélice, donde P_i son presiones, v_i son velocidades del fluido, F_s es la fuerza de sustentación, D_{helice} es el diámetro de la hélice y $m_{max} * g$ es el peso máximo soportado expresado en términos de masa máxima y aceleración gravitacional.

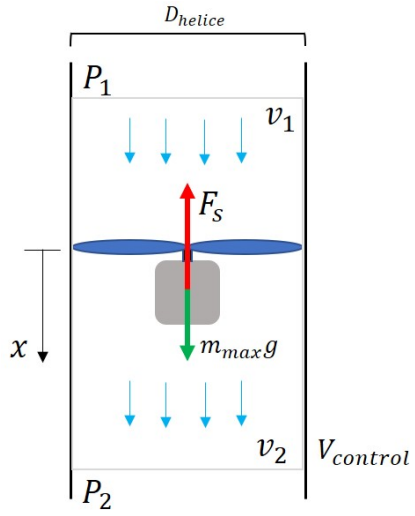


Figura 6.11: *Diagrama del volumen de control con motor y propelas*

Para simplificar la tarea de análisis, se considerará al sistema como abierto y de régimen estacionario, y al fluido de trabajo (aire) como incompresible cuya temperatura antes y después de la propela es igual.

Para conocer la masa máxima que puede sostener, en el caso estático, se aplica de forma directa la segunda ley de newton, con la que se obtiene la ecuación (6.3).

$$\sum F_x = m_{max}g - F_s = 0 \tag{6.2}$$

$$m_{max} = \frac{F_s}{g} \tag{6.3}$$

Para definir la fuerza de sustentación F_s, podemos aplicar el concepto de impulso y cantidad de movimiento. En este caso, como la fuerza solo tiene efecto en dirección del eje x la ecuación adquiere un carácter escalar y se puede representar por la ecuación (6.4), donde m es la masa del fluido (aire) dentro del volumen de control.

$$mv_1 + \int F_s \delta t = mv_2 \tag{6.4}$$

La velocidad inicial del fluido es $v_1 = 0$, si se sustituye en la ecuación (6.4) y se reescribe considerando la definición de gasto másico se obtiene la ecuación (6.6):

$$F_s = \frac{mv_2}{t} \tag{6.5}$$

$$F_s = \dot{m}v_2 \tag{6.6}$$

Para obtener la velocidad del fluido v_2 se debe conocer la energía que le transfiere la propela a éste. Se utiliza el balance de energías para sistemas abiertos de acuerdo con la primera ley de la termodinámica, representada en la ecuación (6.7), donde Q es calor, W trabajo, E_c energía cinética, E_p energía potencial y H entalpía.

$$Q + W = \Delta E_c + \Delta E_p + \Delta H \tag{6.7}$$

Al utilizar la definición de entalpía en términos de la energía interna U, presión P y volumen V representada en (6.8), junto con la de la la energía cinética (6.9) y potencial (6.10) se puede construir la ecuación (6.11); esta se puede expresar en función de potencia calorífica, potencia mecánica y gasto másico si tomamos en cuenta el tiempo, como se muestra en la ecuación (6.12).

$$H = U + PV \tag{6.8}$$

$$E_c = \frac{1}{2}mv^2 \tag{6.9}$$

$$E_p = mgz \tag{6.10}$$

$$Q + W = \left[\frac{1}{2}m(v_2^2 - v_1^2) + gm(z_2 - z_1) + \{(U_2 + P_2V_2) - (U_1 + P_1V_1)\}\right] \tag{6.11}$$

$$\dot{Q} + \dot{W} = \dot{m}\left[\frac{1}{2}(v_2^2 - v_1^2) + g(z_2 - z_1) + \{(u_2 + P_2V_{e2}) - (u_1 + P_1V_{e1})\}\right] \tag{6.12}$$

Como se propuso que la temperatura del aire no cambia, entonces el calor es $Q = 0$ y las energías internas específicas son iguales ($u_1 = u_2$). Al considerar al aire como incompresible los volúmenes específicos son iguales ($V_{e1} = V_{e2}$), además, las presiones al inicio y al final son atmosféricas y por lo tanto, son iguales ($P_1 = P_2$). También, se desprecia la diferencia de altura que existe debido a las dimensiones de la hélice ya que esta es muy pequeña ($z_1 = z_2$). Por último, tal como en la ecuación (6.6), la velocidad inicial es cero ($v_1 = 0$). Con estas consideraciones se puede expresar la energía del sistema de la siguiente forma:

$$\dot{W} = \dot{m}\frac{1}{2}(v_2)^2 \tag{6.13}$$

Si se define el gasto másico $\dot{m}$ en función de la velocidad del fluido, su densidad y el área transversal (6.14), que a su vez se expresa como se muestra en (6.15) así se pueden sustituir en (6.13) para obtener lo siguiente:

$$\dot{m} = Av_2\rho \tag{6.14}$$

$$A = \pi\left(\frac{d}{2}\right)^2 \tag{6.15}$$

$$v_2 = \sqrt[3]{\frac{8\dot{W}}{\pi d^2 \rho}} \tag{6.16}$$

Al sustituir las ecuaciones (6.6), (6.14) y (6.16) en (6.3) se obtiene:

$$m_{max} = \frac{1}{g}\left(\dot{W}d\sqrt{\rho\pi}\right)^{\frac{2}{3}} \tag{6.17}$$

Finalmente, hay que recordar que el motor no tiene una eficiencia del 100 % para transferir la energía, mucha se pierde en calor, vibración, etc. por lo que se decidió que la potencia transferida por el motor tenga un factor μ que representa estas pérdidas tal que $\dot{W} = \mu\dot{W}_m$. Como resultado, se puede calcular la masa que soporta el sistema con la ecuación (6.18).

$$m_{max} = \frac{1}{g}\left(\mu\dot{W}_m d\sqrt{\rho\pi}\right)^{\frac{2}{3}} \tag{6.18}$$

En la Tabla [6.1] se muestra el cálculo de la masa máxima soportada por el conjunto según las características específicas del motor y un valor comercial aceptable propuesto para las hélices, todo basado en la ecuación (6.18).

Constantes	
Diámetro hélice [in]	8.00
Diámetro hélice [m]	0.20
Corriente motor [A]	18.00
Voltaje motor [V]	12.00
Potencia máxima [W]	216.00
Densidad del aire [Kg/m^3]	1.23
Gravedad [m/s^2]	9.78

Eficiencia	Fuerza de sustentación por motor [N]	Masa soportada por motor [Kg]	Masa soportada por el sistema[Kg]
1.00	19.53	2.00	3.99
0.90	18.20	1.86	3.72
0.80	16.83	1.72	3.44
0.70	15.39	1.57	3.15
0.60	13.89	1.42	2.84
0.50	12.30	1.26	2.52
0.40	10.60	1.08	2.17
0.30	8.75	0.89	1.79
0.20	6.68	0.68	1.37
0.10	4.21	0.43	0.86

Tabla 6.1: *Masa máxima soportada por los motores*

Hélices 8045

Para este proyecto se seleccionaron hélices estándar 8045 de dos álabes fabricadas en plástico ABS Figura [6.12], con base en la información obtenida en la tabla [6.1]. Se utiliza el par complementario (1CW-1CCW) para contrarrestar el factor P. Sus medidas son de 8 x 4.5[in] (208x114[mm]) de acuerdo con la convención mencionada anteriormente. Cada una cuenta con los sujetadores de goma superior e inferior necesarios para ensamblar correctamente con el adaptador de bala del eje del motor.

Estas medidas de propelas pertenecen al rango recomendado por el fabricante del motor para funcionar con las condiciones correctas.

Figura 6.12: *Propelas seleccionadas: Hélices 8045 ABS* [11]

[11] https://www.google.com/url?sa=i&url=https%3A%2F%2Felectronica.mercadolibre.com.mx%2Fdrones-accesorios-repuestos%2Festado-de-mexico%2F&psig=AOvVaw3o_jljaWbJvBCkekTv8DCz&ust=1648254857890000&source=images&cd=vfe&ved=0CA0Q3YkBahcKEwiA9-78geD2AhUAAAAAHQAAAAAQBQ (24/Marzo/2022)

6.1.4. Sensores

Unidad de medición inercial (IMU)

Los *sistemas de navegación inercial* (INS) son capaces de calcular la posición, la orientación y la velocidad, ya sea en relación con algún sistema, punto de referencia o coordenadas geográficas para ayudar a la navegación y el control de vehículos. El concepto básico detrás de estos es la medición del movimiento relativo para proyectar una posición cambiante en algún marco de referencia inercial a lo largo del tiempo. El elemento central de un INS es su *unidad de medida inercial* (IMU), que es un dispositivo que mide ángulos, aceleraciones lineales e intensidades de campo magnético, mediante la combinación de diferentes sensores embebidos en su interior [31].

Comúnmente las IMU poseen 9 grados de libertad, como consecuencia de integrar un acelerómetro, giroscopio y magnetómetro de tres ejes cada uno, es decir, un eje por cada dirección movimiento. Se pueden clasificar según la tecnología en la que se basan. La tecnología FOG (Fiber Optic Gyroscope) funciona gracias a una bobina de fibra óptica que utiliza la interferencia de luz para realizar las mediciones; éstas generalmente tienen el mejor rendimiento, duración y precisión, aunque su precio es elevado. La RLG IMU (Ring Laser Gyroscope) es la opción más equilibrada entre rendimiento y precio. Y también existen las IMU basadas en MEMS (Micro Electro-Mechanical Systems), que son las más populares; estas realizan el sensado mediante chips de silicio con diapasones, por lo que son más económicos de producir, confiables, de poco mantenimiento y aplicables para proyectos que requieran soluciones miniaturizadas (UAS)[32]. Aunque no son particularmente precisas, se destacan por su bajo consumo de energía.

La referencia para los sensores que integran la IMU se define mediante un sistema dextrógiro como en la figura [6.13]. La rotación del eje X corresponde al *roll*, sobre Y al *pitch* y un giro en el eje Z es el *yaw*. Cada sensor que posee la IMU tiene ventajas y desventajas en el método para calcular dichos ángulos[33].

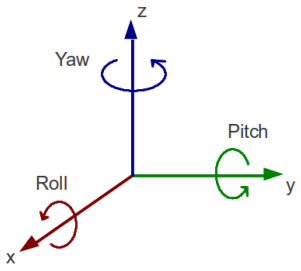


Figura 6.13: *Sistema de referencia dextrógiro de la IMU* [12]

Acelerómetro

Mide las aceleraciones aplicadas a un objeto; estos instrumentos típicamente consisten en: una *masa de prueba* o *masa sísmica* móvil utilizada para generar una fuerza al acelerar, una suspensión formada por uno o varios soportes rígidos que impiden el desplazamiento de la masa en los ejes fijos, resortes o soportes flexibles que regresan la masa a su posición original en el eje que se está midiendo una vez que la aceleración desaparece, un amortiguador que puede ser el medio en el que se encuentra la masa dentro del encapsulado y un mecanismo mediante el cual se registra el desplazamiento de la misma.

Existen diferentes tipos de acelerómetros basados en MEMS como: los piezoresistivos, piezoeléctricos y capacitivos. Los primeros utilizan un material conductor en los soportes de la masa que, al ser deformados, modifican su resistencia eléctrica. Los piezoeléctricos poseen un material que se presiona contra la masa, se deforma y genera una diferencia de potencial. Por último, los de tipo capacitivo, que son los más comunes y cuyo diagrama se encuentra en la Figura [6.14], consisten en dos placas de material conductor, uno fijo y otro

[12] `http://doc.aldebaran.com/2-4/family/robots/joints_robot.html` (14/Marzo/2022)

acoplado a la masa; a medida de que esta se desplaza, modifica la capacitancia entre ambas placas. Con cada una de estas variaciones se puede calcular la aceleración, primero al obtener indirectamente la deformación o desplazamiento de la masa y luego, según las propiedades de los soportes, la fuerza que genera dicho desplazamiento[34].

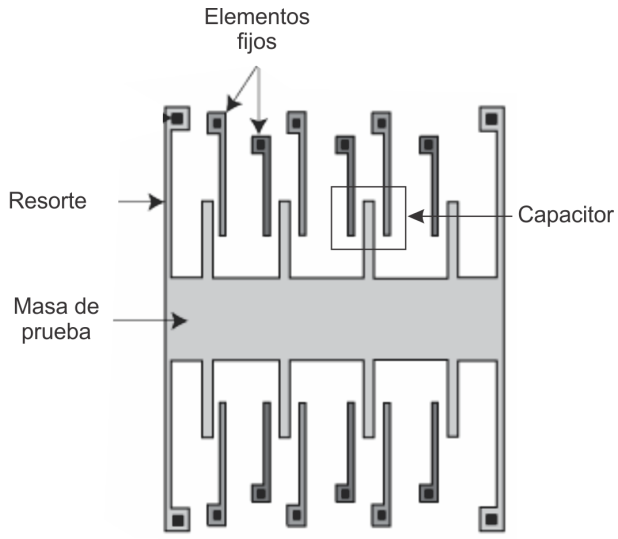


Figura 6.14: *Acelerómetro tipo MEMS capacitivo* [13]

A partir los componentes en cada eje de la aceleración, se determina la posición angular del *roll* y *pitch* respectivamente, mediante las siguientes ecuaciones obtenidas de las matrices de rotación:

$$\tan\theta_{accel} = \frac{a_y}{a_z} \tag{6.19}$$

$$\tan\phi_{accel} = -\frac{a_x}{\sqrt{(a_y^2 + a_z^2)}} \tag{6.20}$$

Al utilizar únicamente el acelerómetro no es posible determinar el ángulo *yaw*, porque sobre el eje vertical Z las aceleraciones en X y Y no varían.

El acelerómetro es muy preciso a largo plazo, sin embargo, cuando el sensor se está trasladando, la aceleración del movimiento afectará el cálculo de la rotación; si aplicamos un filtro a la salida del acelerómetro, la mayor parte del ruido se eliminará y el valor filtrado se usará para combinarse con otro sensor[33].

Magnetómetro

Es un instrumento que permite medir la intensidad del campo magnético en determinada posición mediante el efecto Hall. Cuando este sensor se encuentra sobre una superficie plana sin ningún campo magnético interfiriendo, se puede orientar respecto a los polos de la tierra, en cuyo caso, el ángulo *yaw* se calcula de la siguiente manera:

$$\tan\psi_{mag} = \frac{M_y}{M_x} \tag{6.21}$$

Esta forma de obtenerlo presenta un error cuando el dispositivo se encuentra inclinado, por lo que se podría recurrir a un algoritmo de fusión para compensar el error[35].

[13] `https://docplayer.es/10327528-Facultad-de-ciencias-de-la-electronica.html` (14/Marzo/2022)

Giroscopio

Es un sensor capaz de detectar velocidades angulares. Aquellos basados en MEMS funcionan gracias al efecto Coriolis, Figura [6.15]. Dicho efecto se presenta al hacer oscilar una masa de forma continua y el sistema empieza a rotar, la masa sufre una fuerza inercial ortogonal al eje de rotación del sistema y a la velocidad del objeto, provocando que esta sufra una desviación en su trayectoria. Esta desviación es captada como un cambio de capacitancia de igual modo que en el acelerómetro[36].

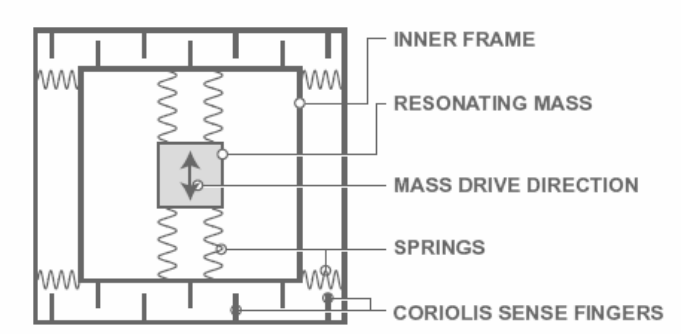


Figura 6.15: *Giroscopio MEMS de efecto Coriolis* [14]

Para transformar la velocidad angular obtenida un ángulo de rotación se utilizan las ecuaciones (6.22):

$$\theta_{gyro} = \omega_\theta \Delta t \qquad \phi_{gyro} = \omega_\phi \Delta t \qquad \psi_{gyro} = \omega_\psi \Delta t \tag{6.22}$$

La integración supone un problema ya que comienza a acumular errores de medición y ruido, lo que provoca una desviación de mediano a largo plazo. Sin embargo, son muy precisos a corto plazo, por lo que si se añade un filtro paso alto se puede mejorar en este aspecto[37].

Fusión de datos

Hace referencia a la unión de diferentes tipos de datos en uno solo para proporcionar una representación más fiable de las mediciones. El primer paso es combinar las mediciones del magnetómetro (m_x, m_y y m_z) con las del acelerómetro para compensar el error del ángulo *yaw* en la medición del magnetómetro cuando se encuentra inclinado y recalcular el ángulo mediante la ecuación (6.21).

$$M_x = m_x \cos(\phi_{accel}) + m_z \operatorname{sen}(\phi_{accel}) \tag{6.23}$$
$$M_y = m_x \operatorname{sen}(\theta_{accel}) \operatorname{sen}(\phi_{accel}) + m_y \cos(\theta_{accel}) - m_z \operatorname{sen}(\theta_{accel}) \cos(\phi_{accel}) \tag{6.24}$$

El filtro complementario es iterativo y se emplea para combinar las mediciones de los tres ángulos obtenidos del acelerómetro y magnetómetro con los del giroscopio, para tomar ventaja de sus características y compensar sus deficiencias. Utiliza dos constantes como pesos, uno para el filtro paso alto y otro paso bajo. Además, se ocupa el ángulo de una iteración anterior para sumarlo a la medición actual del giroscopio.

$$\theta = W_{HP}(\theta_{[t-1]} + \omega_{\theta_{gyro}} \Delta t) + W_{LP}\theta_{accel} \tag{6.25}$$

Al emplear esta expresión en los demás ángulos se reduce en gran medida la desviación de la medida real a largo plazo y también el efecto de las perturbaciones en corto plazo[33].

[14] `https://www.analog.com/ru/technical-articles/mems-gyroscope-provides-precision-inertial-sensing.html` (14/Marzo/2022)

El uso de este tipo de sensores junto a algoritmos de filtrado para determinar la información de orientación da como resultado un sistema conocido como AHRS (Attitude and Heading Reference System), Figura [6.16]. Los valores entregados corresponden al *roll*, *pitch* y *yaw* de cualquier aeronave. La mayoría de AHRS comerciales también cuentan con GPS y barómetros para obtener información de la ubicación y altitud. En este trabajo se implementará un sistema similar con la diferencia de que no es relevante la información correspondiente a la ubicación, altitud y dos de los ángulos de orientación[38].

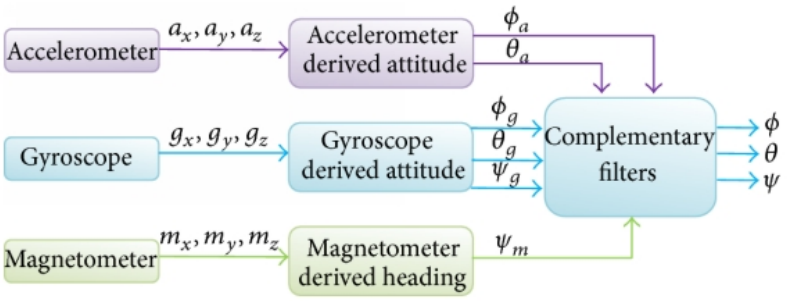


Figura 6.16: *Esquema general de un AHRS* [15]

En el caso particular del prototipo resultado de este trabajo, solo se requiere conocer la orientación sobre un eje (roll); esto significa que algunas características como el magnetómetro puedan exceder las solicitudes y no se utilicen. Sin embargo, la decisión de instalar un sensor IMU fue basada en características como la frecuencia de muestreo y resolución que generalmente ofrecen, la posibilidad de obtener más información de ser necesario, la practicidad debido al tamaño y peso de solo unos gramos, su costo asequible, la disponibilidad y variedad de este tipo de sensores en el mercado, la amplia difusión de su uso en sistemas tipo dron y VTOL los cuales comparten similitud con el prototipo, además de experiencia previa en su implementación.

IMU seleccionada

La unidad de medición inercial escogida para recabar la posición angular fue el modelo BMI085 fabricado por Bosch[39], Figura [6.17]. Tiene la capacidad de detectar movimientos y rotaciones en 6 grados de libertad (6-DoF) por medio de la tecnología MEMS; cuenta con un giroscopio y acelerómetro capacitivo de tres ejes y 16 bits de resolución.

- Resolución acelerómetro: 0.1 [mg] - 16 bits
- Resolución giroscopio: 0.004 [°/s] - 16 bits
- Voltaje de alimentación: 2.4-3.6 [V]
- Consumo de corriente en operación: 5.15 [mA]
- Dimensiones (integrado): 3 x 4.5 x 0.95 [mm]
- Dimensiones placa: 43 x 20 x 10 [mm]
- Protocolos de salidas y entradas digitales SPI e I2C
- Frecuencia de muestreo 12.5 [Hz] - 2 [kHz]

[15] `https://www.hindawi.com/journals/tswj/2014/597180/fig1/` (02/Junio/2022)

Figura 6.17: *IMU seleccionada: IMU BOSCH BMI085 6-DoF* [16]

Este modelo presenta algunos beneficios sobre otras alternativas que tienen mayor disponibilidad en el mercado. Se puede destacar la velocidad de lectura dada por la frecuencia de muestreo y su alta fiabilidad con resoluciones muy elevadas. El fabricante indica que cada sensor se encuentra previamente calibrado, por lo que agiliza la instalación para comenzar a hacer pruebas. La posibilidad de elegir cualquiera de sus protocolos, I2C o SPI según sea conveniente, lo hace compatible con cualquier otro dispositivo. El tamaño compacto de la PCB sobre la que está montada permite poder adaptarlo en casi cualquier parte de la estructura.

Sensor infrarrojo

Existen varios tipos de sensores para medir distancia basados en principios físicos distintos como el ultrasónico, resistivo, LVDT, fotoeléctrico, etc. Sus características les brindan rangos, sensibilidades y resoluciones distintas. En este trabajo se utilizará un sensor fotoeléctrico infrarrojo.

Tal como se menciona en el manual de explicación técnica de sensores fotoeléctricos de la empresa OMRON[40], un sensor *fotoeléctrico* puede detectar objetos, condiciones de superficie y otro tipo de propiedades. Normalmente, están compuestos por un *emisor* y un *receptor* de luz. El emisor se encarga de proporcionarla (ya sea infrarroja, ultravioleta o visible) y el receptor de detectar los cambios en esta. Siempre está calibrado para detectar el mismo tipo de luz que emite su contraparte y emitir una señal eléctrica que indica el cambio detectado. Se pueden dividir en tres los tipos de sensores de luz: sensor de barrera, retro-reflectivo y difuso reflectivo. Se profundizará en este último debido a que es el que se utilizará.

Los sensores *difuso-reflectivos* son capaces de medir desde unos centímetros hasta varios metros y basan su funcionamiento en la reflexión difusa de la luz. Cuando se tiene un rayo de luz y se dirige hacia una superficie plana como un cristal o un espejo, este rebota. El ángulo de salida es igual al de incidencia, a este fenómeno se le llama reflexión regular o de espejo. No obstante, cuando la superficie con la que choca es de un material opaco, la luz de igual forma rebota, pero de forma dispersa y con una menor intensidad. A este tipo de reflexión se le conoce como reflexión difusa. En la Figura [6.18], se observan ambos efectos.

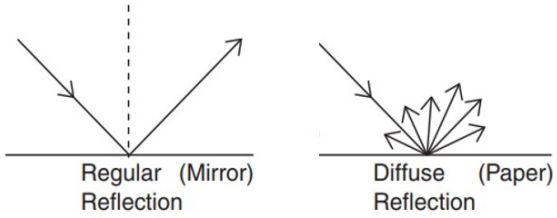


Figura 6.18: *Tipos de reflexión de la luz* [17]

[16] `https://media.digikey.com/Photos/Bosch%20Sensortec/SHUTTLE-BOARD-BMI085.jpg` (30/Marzo/2022)

Cuando el emisor envía luz, esta se refleja en el objeto cuya distancia se quiere medir, después, el receptor recibe el rayo producto de la reflexión difusa y amplifica su intensidad para facilitar la tarea de identificarla como pasa en [6.19]. Una vez amplificada, esta incide en un material piezoeléctrico (Fotodiodo, fototransistor o CCD -charged coupled device-) que emite una señal que ayuda a identificar el ángulo de dicho rayo. La linealidad del sensor depende ampliamente del piezoeléctrico que se utilice en el receptor.

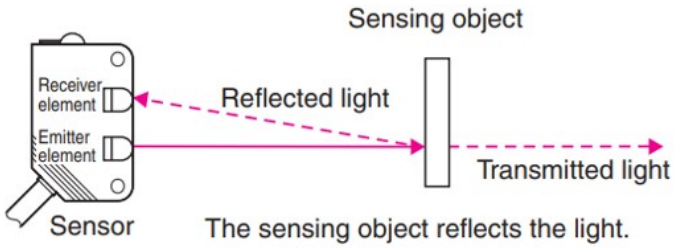


Figura 6.19: *Emisión y recepción de la luz en el sensor* [17]

Para obtener la medición se utiliza una técnica denominada *triangulación*. El ángulo de incidencia del rayo reflejado dependerá de la distancia del objeto en donde se reflejó, a mayor distancia menor la inclinación y viceversa, tal cual el diagrama [6.20]. Normalmente, los sensores difuso-reflectivos tienen el emisor y el receptor en la misma carcasa, lo cual permite que la separación entre ellos sea constante. Si se conoce el ángulo de incidencia y este espacio, se puede estimar la distancia del objeto.

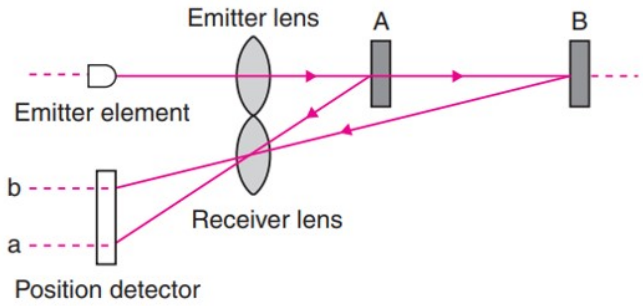


Figura 6.20: *Triangulación en el sensor* [17]

Finalmente, la distancia resultante se traduce como un voltaje, corriente o el dato transmitido por un protocolo de comunicación (Serial, i2c, etc), depende del diseño y marca del sensor.

Una de las ventajas de utilizar este tipo de sensores es que existen pocas restricciones de los objetos que se sensan, otra es la rapidez de sensado debido a la velocidad a la que viaja la luz y a que solo cuenta con componentes electrónicos. Además, pueden tener una resolución elevada dependiendo del tipo de tecnología que se utilice y no es un sensor invasivo. Hay que tener en cuenta que la intensidad de la luz reflejada y la distancia medida pueden variar según las condiciones del objeto como color, acabado y geometría de la superficie. De igual forma, fuentes de luz externas similares a la que emite el sensor pueden causar interferencias.

[17] `https://www.ia.omron.com/data_pdf/guide/43/photoelectric_tg_e_8_4.pdf` (24/Abril/2021)

Sensor infrarrojo seleccionado

Para medir de la distancia vertical del móvil se utiliza el sensor infrarrojo GP2Y0E03 de SHARP[41], mostrado en la Figura [6.21]; está compuesto por una combinación integrada de sensor de imagen CMOS e IR-LED. Este componente utiliza el método de triangulación y tiene las siguientes características:

- Rango de medición: 4 a 50 [cm]
- Voltaje mínimo de operación: 2.7 [V]
- Dimensiones: 16.7 x 11 x 5.2 [mm]
- Tipo de salida analógica/ digital (I2C)

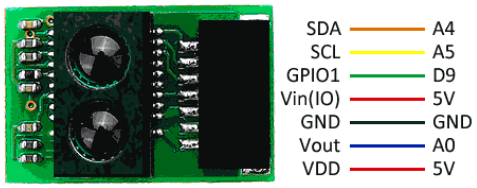


Figura 6.21: *Sensor infrarrojo seleccionado: GP2Y0E03* [18]

El sensor fue seleccionado por su rango de medición, que cubre completamente la distancia de operación, su comportamiento lineal en relación con la distancia medida y la posibilidad de utilizar una salida analógica y una digital I2C que lo hace disponer de flexibilidad de uso. Debido a la tecnología utilizada, el tipo de reflectividad del objeto medido, la temperatura ambiente y el tiempo de operación tienen poca repercusión en el valor de sus medidas. En los fines aplicativos, el sensor destaca por sus reducidas dimensiones, su peso y precio.

6.1.5. Microcontrolador

Un microcontrolador es un circuito integrado con una escala de integración muy grande (VLSI, very large scale integration) que internamente contiene una *Unidad Central de Procesamiento* (CPU, Central Processing Unit), memoria para código, memoria para datos, temporizadores, fuentes de interrupción y otros recursos necesarios para el desarrollo de aplicaciones con un propósito específico.

Si bien, un MCU incluye prácticamente todos los elementos necesarios para ser considerado como una computadora, frecuentemente no es tratado como tal debido a su capacidad de procesamiento limitada. Las aplicaciones que demanden un alto rendimiento y requieran manejar un conjunto masivo de datos, tales como el procesamiento de imágenes o video, están fuera de su alcance. No obstante, los recursos incluidos en un MCU son suficientes para aplicaciones de propósito específico. Su uso típico consiste en el desempeño de funciones de “control” que implican interacción con el “mundo real”, monitorear condiciones (a través de sensores) y encender o apagar dispositivos (por medio de actuadores)[42].

En el mercado existe una amplia gama de fabricantes de microcontroladores los cuales han desarrollado diversas familias de modelos, cada uno con capacidades y características especiales. Para identificar que opción es mejor en términos generales, se necesita recabar la mayor cantidad de información respecto a la aplicación en la que será implementado, para conocer que especificaciones técnicas y características son claves para el desempeño. En algunas ocasiones la aplicación no exige una alta demanda de recursos por lo que permite que cualquier placa de desarrollo sea útil.

El prototipo en construcción permite flexibilidad en cuanto a las capacidades del MCU, sin embargo, para mejorar el rendimiento se tomaron en cuenta las conexiones necesarias para los sensores y sus protocolos, las señales PWM para los actuadores, el Wifi para la comunicación con el ordenador, la capacidad de procesamiento para satisfacer los tiempos de muestreo y ejecutar los ciclos de control, la memoria flash para alojar

[18] `https://www.luisllamas.es/medir-distancia-con-arduino-y-el-sensor-gp2y0e03/` (30/Marzo/2022)

el código y otras características como las dimensiones, el peso y su precio. A continuación, se describen a mayor detalle en qué consisten estas especificaciones y como repercuten en el momento de seleccionar un MCU.

Primero se requieren listar las interfaces de hardware, que son aquellas que establecen la comunicación y conexiones correspondientes entre el MCU con el exterior, estas se encuentran embebidas y su propósito es controlar las funciones de los distintos puertos. Se pueden dividir en las interfaces de comunicación que incluyen los puertos como USB, I2C, SPI, UART, etc. y otras que engloban las entradas y salidas digitales y analógicas como PWM, PPM, ADC, DAC, etc[43].

El primer grupo de interfaces se relaciona con los protocolos de comunicación que permiten el uso de dispositivos periféricos para ampliar las capacidades del dispositivo. Se entiende como periféricos a sensores, actuadores e incluso otros microcontroladores.

Existen dos tipos de protocolos de comunicación: serial y paralelo. La comunicación en paralelo consiste en transmitir de manera simultánea una cierta cantidad de bits, para ello, es necesario de un gran número de líneas más la de referencia o tierra, lo que lo hace poco práctico, además de que sufre a largas distancias por el aumento de interferencia.

Por otra parte, la comunicación serial envía de forma secuencial un bit por un canal o bus, aunque es más lento que la comunicación en paralelo, permite enviar datos a una distancia mayor y usando menos líneas, por lo que se utiliza mayormente. Esta se puede dividir a su vez en síncrona y asíncrona. La síncrona, además de las líneas de transmisión de datos, utiliza una línea de reloj para sincronizar transferencia con cada pulso. La asíncrona no posee esta señal de reloj por lo que se necesitan enviar bits para reconocer el inicio y final del mensaje y establecer una velocidad de transmisión común en ambos dispositivos conocida como baud rate[44].

El protocolo de comunicación I2C(Inter-Integrated Circuit) fue diseñado por Philps Semiconductors en la década de 1980. Es un protocolo síncrono y usa dos líneas, una de datos y otra de reloj, como se ejemplifica en la Figura [6.22].

- SCL (System Clock) es la línea de los pulsos de reloj que sincronizan el sistema.
- SDA (System Data) es la línea por la que se mueven los datos entre los dispositivos.
- GND (Masa) común de la interconexión entre todos los dispositivos enganchados al bus.

Las líneas SDA y SCL son del tipo *drenaje abierto* y se deben polarizar en un estado alto por medio de resistores *pull-up*. El I2C permite el intercambio de información entre muchos dispositivos a una velocidad aceptable. En este tipo de comunicación uno de los dispositivos es maestro y el resto esclavos, solo el maestro puede iniciar la comunicación. La condición inicial (de bus libre) es cuando ambas señales tienen un estado lógico alto, de esta forma cualquier dispositivo maestro puede ocuparlo si establece una condición de inicio. Después, se transmite el primer byte que corresponde a la dirección del dispositivo con el que se quiere establecer la comunicación y la operación a realizar (lectura o escritura). El dispositivo IMU que se utiliza para reconocer la inclinación en el prototipo y el sensor infrarrojo se pueden comunicar de esta forma. Una consideración es que este protocolo trabaja en modo half dúplex, es decir que la comunicación va en una sola dirección en cada momento[45].

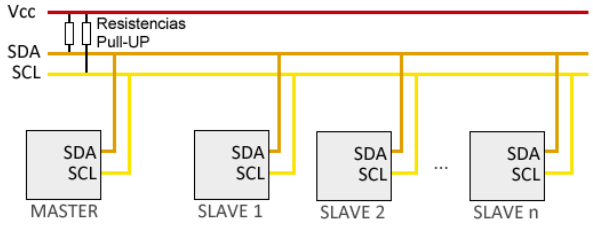


Figura 6.22: *Funcionamiento del protocolo I2C* [19]

Con el fin de calcular el número de puertos requeridos, se deben definir otro tipo de interfaces. El ADC se trata de un convertidor de señales analógicas a digitales que permite generar una representación binaria para la magnitud de una señal de tensión o corriente mediante un cuantificador de código binario. Por su parte, el DAC es lo opuesto y es muy utilizado cuando se busca una manera de generar señales de audio; una tarjeta de audio no es más que un DAC de muy buena calidad y muy rápido. También, otros puertos se tendrán que ocupar de producir señales de tipo PWM o PPM, cuyo funcionamiento se definió con anterioridad cuando se describieron los variadores de velocidad[46].

Con relación al objetivo de comunicación que se planteó, se debe contemplar que en las especificaciones del MCU se indique que puede comunicarse de manera inalámbrica.

El *Wifi* (Wireless Fidelity) es una tecnología que permite la conexión inalámbrica de dispositivos dentro de una red por medio de lo que se conoce como punto de acceso inalámbrico (WAP por sus siglas en inglés). La conexión se realiza mediante señales de radio y su rango de operación se encuentra entre 5 y 150 metros.

Existen distintos tipos de Wifi dependiendo los estándares que usen; estos se diferencian entre sí por la velocidad que se mide en Mbps y la frecuencia de red que corresponde al ancho de banda, ya sea de 2.4[GHz] o 5[GHz], la cual tiene como ventaja menos interferencias a costa de un menor alcance.

Los dispositivos que se encuentran en una red Wifi se dividen en dispositivos terminales como ordenadores, móviles, etc. y dispositivos de distribución o de red como enrutadores que incluyen un punto de acceso Wifi para dar acceso a la red a más equipos, puntos de acceso dedicados a emitir señales Wifi para dar este servicio, aumentar el rango y la cobertura de la señal, y repetidores que del mismo modo se encargan de crear una señal más fuerte y aumentar la cobertura[47].

Los Soft Access Point (Punto de Acceso habilitado por Software) son dispositivos de hardware que originalmente no fueron creados para ser una access point pero que se habilitan como tal por software y permiten crear una Red de Área Local Inalámbrica WLAN. Algunos microcontroladores que poseen conexión inalámbrica se pueden configurar de esta manera y así permitir que este mismo genere una WLAN para conectar otros dispositivos sin necesidad de conectarse a un enrutador[48].

El *Bluetooth* es un protocolo de comunicación inalámbrica para conectar dispositivos de manera remota desarrollado por Ericsson en 1994. Su implementación es diferente a la del Wifi a pesar de que ambos emiten ondas de radio. El Wifi tiene como principal objetivo el conectar dispositivos a la red mientras que el Bluetooth solo los enlaza entre sí[49]. Trabaja a una frecuencia de 2.4 [Ghz] por lo que puede llegar a ocasionar interferencias con las redes Wifi que utilicen esta misma banda; su alcance se limita generalmente a los 10 metros dependiendo de su clase y versión, además, el número máximo de dispositivos que pueden ser conectados al mismo tiempo es limitado. Sin embargo, la manera de emparejar los dispositivos con esta tecnología es mucho más rápida y sencilla porque no se tienen que configurar previamente los dispositivos y la conexión[50].

Determinados todos los medios que el microcontrolador necesitará para intercambiar información con otros dispositivos, se pueden describir otras características cruciales referentes a las capacidades de este como la frecuencia de trabajo del procesador, la capacidad de memoria, la posibilidad de programarlo en distintos entornos de desarrollo, etc.

La frecuencia de operación se debe de tener en mente dado que hay una variedad de dispositivos que pueden operar desde KHz a GHz. Si un microcontrolador trabaja en un rango amplio de frecuencias, es conveniente operarlo en la más baja que le permita un desempeño correcto de la aplicación, porque a menor velocidad de procesamiento, el consumo de energía también es menor.

Un MCU utiliza distintos tipos de memoria y se debe garantizar que exista el espacio suficiente para albergar el programa y sus variables, por lo que siempre es mejor escoger un microcontrolador con un excedente de estas prestaciones. La memoria para el programa (Flash) guarda cada instrucción del lenguaje ensamblador, C o uno de alto nivel que es convertida a instrucciones máquina con un tamaño particular de bits. La que almacena los datos o variables que agregamos al programa se llama memoria (RAM) que dispone de una

[19] `https://www.luisllamas.es/arduino-i2c/` (14/Marzo/2022)

velocidad relativamente alta ya que estas variables son reutilizadas para ejecutar procesos y cálculos. Por último se encuentran los registros, que son las memorias más rápidas y que trabajan a una velocidad cercana a la del procesador[46].

Los *entornos de desarrollo Integrado* o IDE, son software enfocados al diseño de aplicaciones que combinan distintas herramientas en una interfaz gráfica de usuario. Para escogerlas se debe considerar su complejidad, prestaciones y precio. Actualmente muchos fabricantes de microcontroladores dejan disponible de manera gratuita alguna suite de desarrollo.

El siguiente criterio por considerar es el costo cuyo aspecto es esencial una vez que se ha comprobado que el dispositivo cumple con las prestaciones requeridas, es decir, después de un análisis del rendimiento del hardware y software, considerando el uso medio o el peor de los casos.

Bajo la experiencia del desarrollador, muchas veces se prefiere acondicionar un microcontrolador conocido para incluir un recurso externo antes de aprender a manejar un nuevo dispositivo que ya tiene al recurso.

La compatibilidad de un MCU implica que se requiera de pocos ajustes en hardware y software para obtener una versión mejorada de un producto, por ello se recomienda usar un microcontrolador con mayores prestaciones. Este factor es importante si se toma en cuenta que la vida media de los productos es cada vez más corta; actualmente se llega a considerar como obsoleto a un sistema después que ha trabajado un par de años. Esto resalta la conveniencia de utilizar microcontroladores que pertenecen a familias con una gama amplia de dispositivos[42].

En cuanto a la selección del microcontrolador para este proyecto, se tomaron en cuenta como factores principales la velocidad de procesamiento (para afectar lo menos posible el periodo de muestreo), la compatibilidad con protocolos de comunicación I2C, la disposición de salidas PWM o PPM así como de un convertidor analógico digital (ADC) y la capacidad de comunicarse de forma inalámbrica (Wifi o bluetooth); todo por un precio accesible al presupuesto de un proyecto escolar.

Microcontrolador seleccionado

El SoC (System on Chip) ESP32 es diseñado por la compañía Espressif y fabricado por TSMC[51]. Integra en un único chip un procesador Tensilica Xtensa de doble núcleo, con conectividad Wifi y Bluetooth, Figura [6.23]. Actualmente es uno de los dispositivos más utilizados en IoT gracias a la posibilidad de implementar interfaces inalámbricas y sus grandes prestaciones; es considerada la mejor opción entre los módulos de esta categoría teniendo en cuenta su precio[52].

- Xtensa Dual-Core 32-bit LX62 (40nm)
- Frecuencia de operación: 2 núcleos 240[MHz]-(160[MHz] default)
- SRAM 520KB
- ROM 448KB
- 34x GPIOs programables,16x PWM,4x SPI,2x I2C,2x I2S,3x UART
- 18x ADC 12bits
- 2x DAC 8bits
- Wifi 2.4 GHz (150Mbps)QSPI
- Bluetooth 4.2 BLE
- Características adicionales: Sensor de tacto, sensor de efecto hall, sensor de temperatura, QSPI soporte para múltiples memorias flash.
- Entornos de desarrollo: (IDE) de Espressif, (IDE) de Arduino, (IDE) Thonny, uPyCraft (MicroPython)

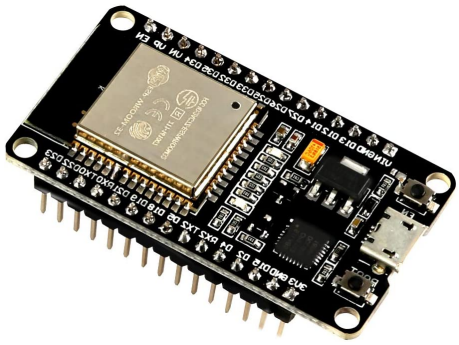


Figura 6.23: *Microcontrolador seleccionado: Placa de desarrollo ESP32 con Wifi y Bluethoot* [20]

Como punto de comparación, el ESP32 es muy superior en capacidades a un Arduino UNO y pude procesar operaciones alrededor de 15 veces más rápido. También, supera en todos los aspectos a su predecesor, que es el ESP8266, por el número de entradas, procesamiento, memoria y mayor compatibilidad.

Para el uso en este proyecto se requiere de un gran número de conexiones y el uso de distintos tipos de comunicación. Esta placa de desarrollo incluye una amplia variedad de interfaces periféricas como: I2C, SPI, CAN, PWM etc. a través de muchos pines por lo que aún quedaran muchos puertos disponibles para otros usos.

La velocidad juega un papel muy importante para la eficacia del controlador, al reducir el tiempo por ciclo se puede alcanzar una mayor fluidez. Con un procesamiento a 240[MHz] se pueden añadir más funciones sin que limite la frecuencia de muestreo.

Cuenta con otras características deseables como el bajo consumo de energía (gracias a un coprocesador que gestiona un modo de ahorro), el soporte de hasta 16[Mb] de memoria flash externa en caso de ser requerida, además es pequeño y ligero respecto a otras alternativas.

Para programarlo se utilizó la IDE de Thonny, que permite subir cada archivo por separado y llamarlos a través de un programa principal *main*. También es capaz de graficar directamente en la terminal los datos impresos. Esta IDE utiliza el lenguaje micropython, que es una versión optimizada para microcontroladores del lenguaje de alto nivel Pyhton; esto permitió utilizar herramientas como objetos, herencia y otras bibliotecas que simplificaron la programación, comunicación y organización del código.

6.1.6. Fuente de Alimentación

El prototipo requiere un suministro de energía mediante corriente eléctrica, pero a diferencia de un dron, el sistema es fijo, por lo tanto, las consideraciones para elegir una fuente de alimentación son diferentes. Los objetivos mencionan el uso continuo del dispositivo, por lo que utilizar una batería carece de sentido si se tiene en cuenta que son costosas y que su ciclo de vida depende de un número de recargas. En este proyecto no existen grandes limitaciones acerca del peso o las dimensiones. La autonomía no es un factor que se tenga en consideración, por lo que se puede remplazar el uso de una batería por una fuente de poder fija que mejorará la disponibilidad para trabajar y disminuirá el costo de mantenimiento.

La fuente de alimentación es uno de los elementos más importantes y costosos de un dispositivo. Si es de menor capacidad que la requerida se dañará y podrá ocasionar algún accidente, por el contrario una de mayores prestaciones probablemente será mucho más costosa, de forma que se debe seleccionar cautelosamente. Las fuentes de alimentación AC/DC se pueden clasificar en lineales y conmutadas. Las primeras fuentes en usarse fueron las lineales, que poseen menos componentes y son más fáciles de reparar. Proporcionan una o varias tensiones continuas estables y constantes. El primer elemento para su funcionamiento es un transformador que reduce la tensión de la corriente alterna. Después, la corriente pasa por un circuito rectificador que la trasforma en corriente directa pulsante. Luego, para que permanezca constante y se eliminen

[20] `https://www.sigmaelectronica.net/producto/esp-32/` (30/Marzo/2022)

las pulsaciones, un filtro es añadido mediante capacitores. Por último, el regulador se encarga de estabilizar el voltaje de salida en uno fijo independientemente de los cambios que pueda tener la entrada[53].

Una fuente conmutada tiene un menor tamaño y una mayor eficiencia gracias a que la regulación es efectuada por transistores de conmutación. Primero, se realiza la rectificación y el filtrado en alto voltaje para lograr un consumo de corriente menor y sea fácil disminuir el zumbido de la señal. Posteriormente, pasa por un circuito conmutador que convierte la señal continua en una pulsante de alta frecuencia que alimenta a un transformador para reducir el voltaje, el cual posee un núcleo de ferrita de menor tamaño. Nuevamente, el resultado pasa por un circuito rectificador y filtrado secundario que devuelven una señal continua. El uso de una fuente de este tipo tiene un mayor rendimiento, un menor costo y tamaño[54].

Una vez que se consideró instalar una fuente conmutada AC/DC se deben determinar sus características. Como consecuencia de la baja resistencia de los embobinados, los motores BLDC se caracterizan por un alto consumo de corriente, es por ello por lo que regularmente se utilizan baterías tipo LiPo con una gran capacidad de descarga. Los motores escogidos tienen un consumo de corriente aproximado de 19[A] cada uno, al considerar que el voltaje de una batería de 3 celdas (11.1[V]),que es lo recomendado en esa configuración según el fabricante, se determinó que la capacidad mínima de una fuente comercial para el funcionamiento de los motores y el resto de los componentes es de 600[W] y 12[V].

Fuente seleccionada

La fuente seleccionada es el modelo SE-600-12 del fabricante Mean Well[55], con una potencia de 600[W] (12[V] y 50[A]), Figura [6.24]. Cumple con los valores requeridos y ofrece algunas protecciones adicionales, sus características son:

- Entrada: 90-132 [VAC] y 180-264 [VAC] (por switch)
- Salida: 12[V] (rango ajustable 10 13[V]) y 50[A] DC
- Eficiencia: 83 %
- Ruido: 150 [mV_{pp}]
- Dimensiones: 247 x 127 x 63.543 [mm^3]
- Peso: 2.1 [Kg]
- Refrigeración: por ventilador
- Protección por sobrecarga OPP 105 125 % de la potencia de salida.
- Protección por sobre voltaje OVP de 13.8 16.2[V].
- Protección por sobrecalentamiento OTP después de 85[°C].

Figura 6.24: *Fuente seleccionada: SE-600-12 Mean Well* [21]

Su eficiencia se encuentra en un valor promedio para este tipo de fuentes. No tiene problemas con la disipación de calor gracias a su ventilador que tiene un buen flujo de aire, aunque es un poco ruidosa.

Las protecciones se encargan de apagar la fuente en caso de que el voltaje generado sobrepase cierto umbral del voltaje de salida (OVP). La protección (OPP) identifica si la potencia entregada es más del valor indicado por el fabricante, en este caso de 5 a 25 %. Por último, la protección (OTP) identifica mediante un termistor cuando la temperatura sobrepasa el valor de 85[°C] a causa de una sobrecarga o cuando el ventilador está bloqueado[56].

6.2. Diagrama eléctrico

La conexión de los componentes se realizó de acuerdo con el diagrama de la Figura [6.25]. Se colocó una salida de 5[V] libre para cualquier periférico externo y se redujo el voltaje de 12[V] a la salida de la fuente en dos etapas, mediante los reguladores LM7805 y LM7809. En esta misma salida se encuentra un LED indicador para cerciorase de que la fuente encendió correctamente.

Para que haya la menor cantidad de cables posibles desde la base hasta la parte móvil del prototipo, se utilizaron las salidas BEC de 5[V] de los variadores de velocidad para alimentar el microcontrolador y los sensores. Para el caso del módulo infrarrojo de distancia se usó su salida analógica conectada al Pin ADC D34 del MCU y así obtener su medición en 12 bits. En la IMU se colocaron dos resistores PUll-UP en las terminales de comunicación y se declararon dichos pines como SCL y SDA para el protocolo I2C. Debido a que estos dos sensores funcionan con un voltaje de entrada de 3.3[v], se conectaron a la salida regulada del microcontrolador con este mismo voltaje. Por último, se les asignó a los pines D18 y D19 las salidas de las señales PWM para regular la velocidad de los motores mediante los ESC.

[21] `https://pdf1.alldatasheet.com/datasheet-pdf/view/259427/MEANWELL/SE-600-12.html` (30/Marzo/2022)

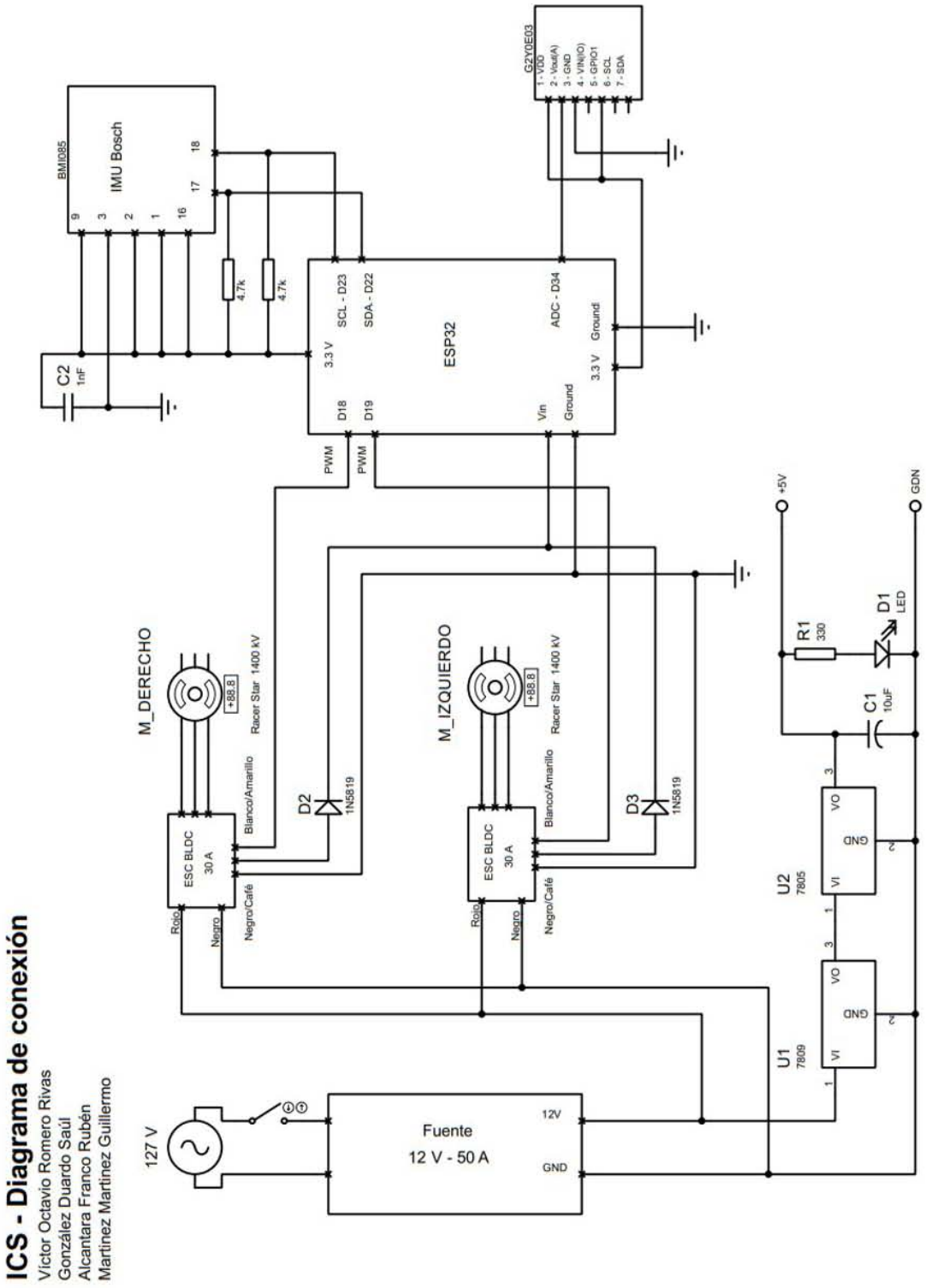


Figura 6.25: *Diagrama de conexión* [22]

[22] Realizado con Proteus 8 Professional

6.3. Modelado CAD

En un amplio sentido, se entiende al *Diseño Asistido por Computadora* (CAD) como la aplicación de la informática al proceso de diseño. Usa técnicas de generación de gráficos en computadora respaldados por software dedicado para automatizar el proceso de diseño de algún elemento. Tiene la capacidad para generar modelos virtuales bidimensionales y tridimensionales de cualquier objeto físico.

La representación computacional del modelo permite realizar automáticamente el dibujo de detalle y la documentación del diseño, además, posibilita la utilización de métodos numéricos para realizar simulaciones sobre el modelo como una alternativa y apoyo a la construcción de prototipos. Una vez que el diseño conceptual se materializa, el modelo geométrico puede comenzar a realizarse.

La popularidad de la utilización de sistemas CAD radica en la reducción de tiempo invertido en los ciclos de exploración, fundamentalmente por el uso de sistemas gráficos interactivos, que permiten realizar las modificaciones en el modelo y observar inmediatamente los cambios producidos en el diseño, sin tener que gastar recursos en modelos de prueba[57].

Durante el desarrollo del diseño se empleó la herramienta de software *Autodesk Inventor 2020*[58], en donde se modelaron las piezas y se generaron los ensambles de la estructura. También se importaron algunos componentes como los electrónicos y la tornillería para la visualización más detallada del sistema.

El ensamble principal se muestra en la Figura [6.26] y corresponde a la estructura del balancín o *frame*, que es la parte móvil donde se sujetan la mayor parte de los componentes eléctricos, incluyendo motores, variadores de velocidad, microcontrolador y sensores. En consecuencia, se requiere que dicha estructura sea liviana pero suficientemente rígida para proteger los componentes. Mediante algunas iteraciones, se generó un conjunto de piezas con perforaciones para formar una malla que disminuyera el peso del cuerpo, similar al diseño de una grúa. Se decidió que fuera una estructura tridimensional, que emula un perfil, para otorgar mayor rigidez al tener en mente que una placa no soportaría la flexión ejercida por las cargas. El ensamble también contempla elementos de plástico impresos que sujetan el cuerpo principal a los postes de la estructura y otras piezas para fijar cada componente mediante tornillería y adhesivos.

Figura 6.26: *Ensamble CAD del balancín y componentes*

El segundo ensamble concierne a la caja base [6.27], cuyo fin es albergar al sistema de alimentación y otorgar otras cualidades como estructura y estabilidad. Para su diseño se tomaron en cuenta las dimensiones de la fuente y la necesidad de mantener un flujo de aire suficiente para evitar el sobrecalentamiento. Esta caja contempla poder realizar mantenimientos al dispositivo como el remplazo de fuente o el desmontaje de los postes. Adicionalmente, posee dos compartimientos en los laterales, en el izquierdo se encuentra el interruptor y parte del cableado eléctrico, mientras que el derecho es para propósitos generales.

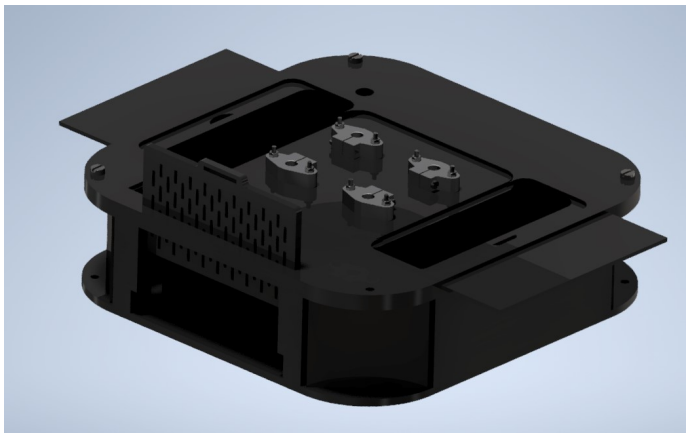


Figura 6.27: *Ensamble CAD de la base del dispositivo*

La vista previa del modelo completo puede observarse en la Figura [6.28]. La base se encuentra en la parte inferior y sobre ella se colocan cuatro postes que sostienen el balancín al centro de la estructura. Los límites del movimiento se restringen verticalmente mediante collares de retención sujetos a los postes y resortes para amortiguar las caídas; el movimiento angular se limita por tornillos colocados en los elementos de PLA y que hacen contacto con el *frame*. El presente modelo se sometió a un análisis de esfuerzos para identificar los posibles cambios necesarios en la geometría de las piezas que mejoren las propiedades mecánicas. Si se requiere más información acerca de alguna pieza en particular, puede ser consultada en los planos respectivos ubicados en el apéndice B.

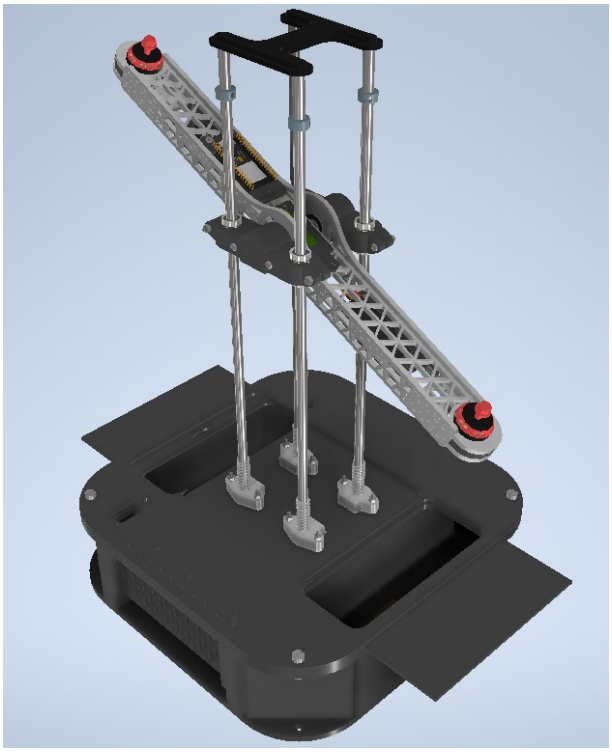


Figura 6.28: *Modelado CAD del ensamble completo*

6.4. Materiales

Al ser un dispositivo destinado para la práctica, se debe hacer una selección adecuada de los materiales para cada elemento de la estructura, considerando el *frame* (que constituirá el ensamble móvil), la base y el resto de los componentes que servirán como soporte al modelo. A continuación, se enlistan los materiales principales utilizados en la construcción del modelo, así como sus propiedades relevantes.

ACP (Aluminium Composite Panel)

Para la fabricación del *frame* en drones comerciales generalmente se emplean materiales con una resistencia considerable pero una baja densidad como la fibra de carbono (en drones de alta calidad), madera o algunos polímeros como PLA y ABS usados en impresión 3D.

El ACP es un material compuesto por tres capas, usualmente un núcleo de polietileno o poliuretano rodeado por dos capas de un material más rígido, en este caso aluminio. Su aplicación recae en situaciones donde la resistencia y el peso son muy relevantes. Es utilizado en el transporte y la industria de la construcción como un recubrimiento exterior, por lo cual también provee de una buena durabilidad. La distribución del material en capas condiciona su grosor, lo cual mejora las cualidades de la estructura.

El costo de fabricación de este material es menor en comparación al de una impresión 3D, además de que este último método restringe de forma importante las dimensiones de las piezas que se pueden producir. La madera tiene el inconveniente de que sus propiedades no son uniformes a lo largo del material y se ve altamente afectada por las condiciones ambientales. Por otra parte, el alto costo y baja disponibilidad de la fibra de carbono descarta su utilización. Por las características descritas, se considera al ACP un material idóneo para nuestra aplicación[59].

Acero rectificado

Debido a que el diseño contempla una estructura fija que restringe el movimiento de la parte móvil y que funciona como soporte de la misma, se necesita un material que principalmente sea resistente y que sufra de mínimas deformaciones para evitar que los golpes o vibraciones desplacen o dañen la estructura.

El acero es una aleación de hierro y carbono en distintas proporciones, lo que confiere propiedades diferentes según su composición, pero en general es un material con excelente resistencia mecánica y muy bueno para estructuras rígidas. Posee una alta densidad por lo que tendrá un mayor peso, sin embargo, para esta parte de la estructura esto no es un factor relevante. Como consecuencia de su alta comercialización y su compatibilidad con otros elementos necesarios para la construcción, se contempló como una buena alternativa para este problema[60].

PLA (Ácido Poliláctico)

Durante la generación de propuestas se requirió diseñar algunas piezas para sujetar el balancín al resto de la estructura y que de la misma forma permitieran su desplazamiento. Las piezas generadas poseen orificios en diferentes sentidos lo que complica su fabricación e igualmente necesitan ser ligeras, ya que los motores tendrán que soportarlas.

Se determinó que estas piezas pueden ser fabricadas con PLA, un termoplástico empleado en la impresión 3D. El PLA es un polímero extensamente utilizado por ser una de las mejores alternativas a aquellos generados a partir de petroquímicos, es biodegradable, reciclable, económico, ligero y a pesar de no ser altamente resistente tiene buena rigidez[61].

La impresión de este material, además de ser sencilla y barata, satisface en gran manera las características deseadas en la pieza, ligera y rígida. También soluciona algunos problemas como el tamaño reducido y una geometría moderadamente compleja. Es por ello por lo que resulta conveniente su uso.

MDF (Medium Density Fibreboard)

Como último elemento se considera la base, que consiste en una carcasa que soporta la estructura y donde se puede colocar el resto de los componentes que no se encuentren integrados en el balancín. Para su elaboración se seleccionó como material al MDF, ampliamente utilizado en la elaboración de muebles. Se caracteriza físicamente por ser un material suave con uniformidad en toda su superficie y por su facilidad de manipulación para realizar cortes y mecanizados. Está compuesto por el prensado de fibras de madera, adhesivos y aditivos[62]. Al no precisar de alguna propiedad especifica fue seleccionado por su bajo costo y comercialización.

6.5. Análisis CAE

El *CAE* (Computer Aided Engineering) es un conjunto de instrumentos que permite integrar al modelo las propiedades del material, condiciones a las que está sometido, etc. De esta forma, se calcula y simula cómo se comportará la pieza o la estructura en aspectos como deformaciones, resistencia, características térmicas, vibraciones, entre otros. Esta herramienta se apoya en el *MEF* (Método de Elementos Finitos), que es un método numérico para la aproximación de ecuaciones y modelos sumamente complejos. Consiste en representar la pieza como un modelo conformado por elementos de geometría sencilla que forman una malla, cuyo comportamiento se especifica mediante un número finito de parámetros asociados a ciertos puntos característicos denominados *nodos*. El cálculo es realizado y propagado sobre cada uno de ellos para obtener valores asociados a su comportamiento, que describen tanto la pieza como el ensamble[63]. El resultado obtenido se puede entender como una aproximación al sistema real que, dependiendo de los parámetros del análisis y de la malla, será más o menos exacto.

Realizar un análisis estructural ayuda a determinar si la elección de materiales y la geometría de los elementos modelados cumplen con las características necesarias para soportar los esfuerzos en condiciones de trabajo durante el tiempo que se considere su vida útil sin repercutir en el funcionamiento, la integridad de la pieza o poner en riesgo algún componente.

El presente estudio se hizo mediante la herramienta *Análisis de Estrés* de *Autodesk Inventor 2020*, donde las pruebas se llevaron a cabo desde una perspectiva estática. Para el análisis se utilizó el esfuerzo de Von Mises para materiales dúctiles (también llamado esfuerzo efectivo), que considera todas las solicitaciones que se puedan presentar sobre la estructura en cualquier dirección mediante un esfuerzo equivalente, ya sea tracción o compresión. También se recurrió al estudio del desplazamiento o deformación máxima, que tiene relación con la gráfica de esfuerzo-deformación de los materiales que conforman el dispositivo. Por último, se tomó en cuenta el *factor de seguridad* para garantizar la integridad de las piezas durante las pruebas, el cual depende del tipo de material (si es dúctil o frágil), el tipo de carga (estática o dinámica) y la incertidumbre acerca de otros factores.

Con base en los valores propuestos por Robert L. Mott en su libro Diseño de Elementos de Maquinas [64], el factor de seguridad de 4 se recomienda para el diseño de estructuras estáticas o elementos de máquinas bajo cargas dinámicas, con incertidumbre en cuanto a la combinación de cargas, propiedades del material, análisis de esfuerzos o el ambiente de operación. Es adecuado dadas algunas diferencias encontradas en las distintas fuentes de las propiedades del material ACP y la confianza en el análisis por elementos no contemplados. Para calcularlo se divide el esfuerzo de cedencia del material entre el esfuerzo de las condiciones del diseño[65]. Las propiedades de cada material necesarias para hacer las pruebas se encuentran en el apéndice C.

Las condiciones particulares para realizar los análisis en el prototipo contemplan que las fuerzas aplicadas en el ensamble son consecuencia del empuje producido por los motores y su propio peso. El valor de la carga de cada actuador en estas pruebas es de 26[N] y corresponde a la fuerza máxima de empuje del motor, medida experimentalmente como ≈ 6.5[N] (la obtención de este parámetro se muestra a detalle en la sección 13.1), multiplicada por el factor de seguridad seleccionado.

En cuanto a la configuración de la malla, se recomienda tener al menos dos elementos en las secciones más estrechas de la figura, por lo que se creó una malla local de 1[mm] por elemento en las caras con aristas

más delgadas del *frame* para asegurarse de cumplir con esta característica. El resto de los parámetros se establecieron tal que el tamaño medio del elemento es de 0.05-0.1 con relación al tamaño del modelo, el tamaño mínimo del elemento es 0.2 en función del tamaño medio, un factor de modificación por defecto de 1.5 y un ángulo máximo de giro de 60[°], cada uno de estos valores se definen en el manual de *Autodesk Inventor 2020*[66].

La primera simulación, expuesta en la Figura [6.29], representa una situación que es probable que ocurra durante las pruebas, muestra los esfuerzos que sufre el ensamble del *frame* de ACP cuando ambos motores generan el empuje máximo en la misma dirección y con la misma magnitud, además de que este se encuentra sujeto desde un punto medio ubicado en el centro del rodamiento. La prueba fue elaborada para determinar si la estructura mallada propuesta soporta la flexión por dichas cargas y si la deformación resultante se encuentra dentro de los límites del propio material.

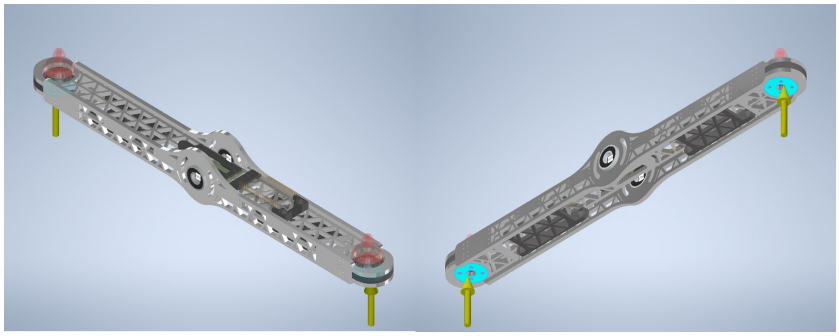
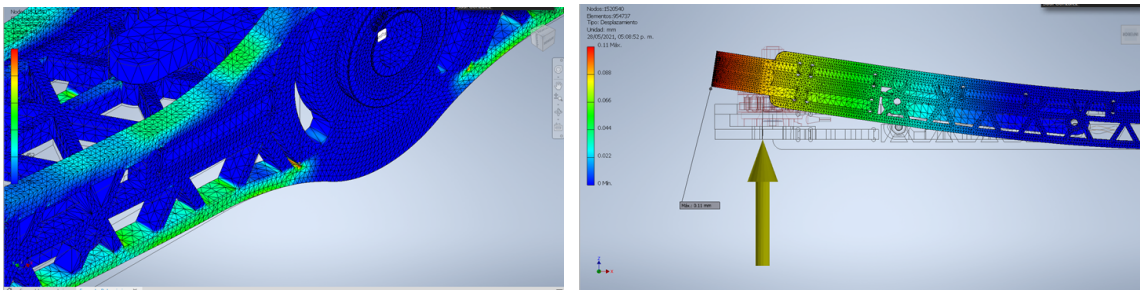


Figura 6.29: *Primera simulación de dos cargas simétricas aplicadas al frame*

De los estudios mostrados en la Figura [6.30], se obtuvo que el esfuerzo máximo de Von Mises es 7.26[MPa] en una de las ranuras más angostas de las piezas laterales del ensamble, donde existe un concentrador de esfuerzos a causa de la malla. El esfuerzo de cedencia del material es de 44[MPa], entonces se infiere que no está cerca de sufrir alguna deformación plástica. El desplazamiento mayor fue de 0.11[mm] a una distancia de 20[cm] del punto de sujeción, la cual se considera una deformación pequeña dentro de la zona elástica. Para finalizar, el coeficiente mínimo de seguridad obtenido en la zona de mayor esfuerzo es de 6.6, pero se debe tomar en cuenta que aunque tuviera el valor de 1 sería suficiente para cumplir con el factor, puesto a que se encuentra contemplado en las cargas como se mencionó anteriormente. Por lo tanto, se dice que la estructura cumple con los requisitos para soportar estas solicitaciones.



(a) Esfuerzo de Von Mises (b) Desplazamiento

Figura 6.30: *Estudios de la primera simulación*

La segunda prueba, Figura [6.31], se hizo para analizar el comportamiento de la estructura (incluyendo la base y postes) mientras que un motor proporciona el máximo empuje y el otro se encuentra apagado, es decir, en su mayor grado de inclinación y donde está en contacto con los tornillos usados como seguros para

limitar el giro. En esta circunstancia, el balancín experimenta una fuerza con una componente horizontal y por tanto ejerce una flexión sobre estructura, también, tanto los seguros como el eje de giro se encuentran sometidos a un esfuerzo cortante producido por el par que genera el empuje del motor.

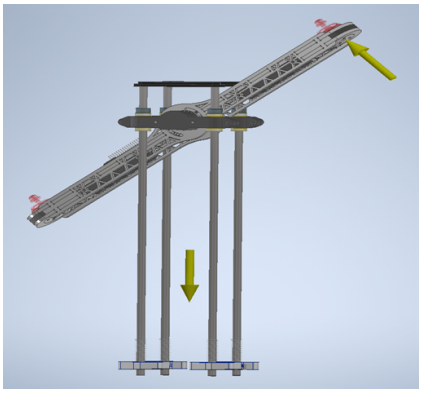
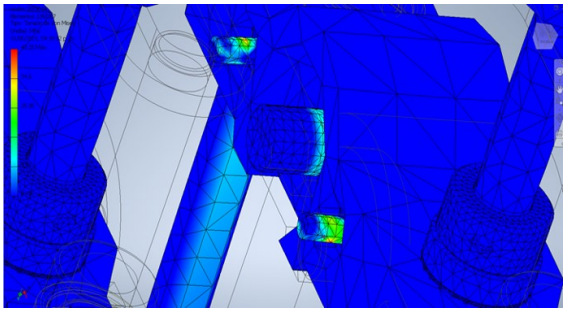
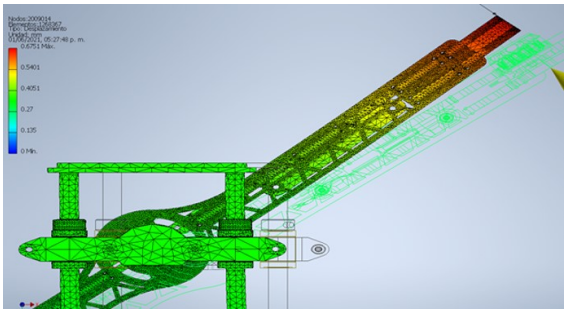


Figura 6.31: *Segunda simulación con empuje generado por el motor derecho*

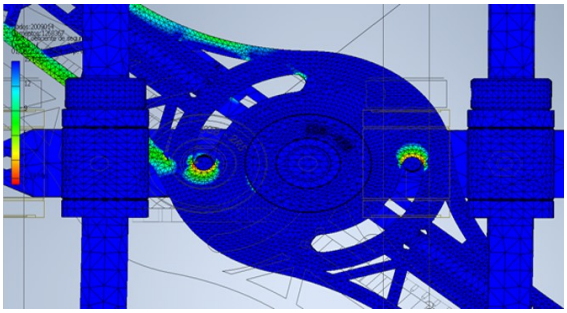
Los resultados de esta simulación en la Figura [6.32], indican que el esfuerzo máximo es de 59.6[MPa] y se encuentra en el punto de contacto entre los tornillos de acero (seguros) y el ACP en un área pequeña por donde se transmite la fuerza del balancín al resto del modelo. Al comparar los 200[MPa] del esfuerzo de cedencia de los tornillos, resulta que existe un gran margen de seguridad, pero no ocurre lo mismo en el ACP puesto que su esfuerzo es de 44[MPa]. Esto explica que el factor mínimo de seguridad en la simulación sea de 0.74 en dicho punto, lo que quiere decir que no se alcanza el factor de 4 sino que uno cercano a 3. El máximo desplazamiento fue de 0.65[mm], resultante de la suma de todas las deformaciones a lo largo de la estructura, a diferencia de la simulación anterior en donde solo se contempló el *frame*, se admite como una deformación aceptable y no pone el riesgo la estructura o a algún otro componente.



(a) Esfuerzo de Von Mises



(b) Desplazamiento



(c) Factor de seguridad

Figura 6.32: *Estudios de la segunda simulación*

Con base en los datos arrojados, se concluye que en la primera prueba la estructura del *frame* cumple con la solicitación de las cargas impuestas con un margen amplio, no existe riesgo de falla y la deformación es imperceptible. En el caso del segundo ejercicio, se debe contemplar que las pruebas indican un factor menor que el establecido; sin embargo, se considera aceptable si se toma en cuenta que no se encuentra alejado del deseado, además del resto de las ventajas que el material ofrece, como la baja densidad, el tiempo de fabricación y su disponibilidad. Como medida preventiva, se interpondrá un elemento semejante a un buje hecho de polímero tal como caucho o PVC en el área de mayor esfuerzo para amortiguar los posibles impactos sobre la superficie en contacto y de esta manera compensar esa diferencia. Finalmente, con estos argumentos, es posible determinar que el diseño cumple con los requisitos estructurales.

6.6. Interfaz gráfica de usuario

Una interfaz de usuario es el medio o herramienta con el que el usuario puede comunicarse con una máquina, esto quiere decir que el operador sea capaz de enviar instrucciones de forma sencilla y a su vez recibir actualizaciones y avisos del progreso de la instrucción, estado de la máquina o errores. Una interfaz puede consistir desde simples líneas de comando basado en texto como el utilizado en algunas distribuciones de Linux, paneles de control (comunes en máquinas herramienta) e incluso con diseños más complejos, como aquellas utilizadas para acceder a una app desde un smartphone o realizar compras por internet. Aunque muchas veces ocupen gran parte del tiempo, memoria y computo de los sistemas, están íntimamente relacionadas con la facilidad de uso del software o dispositivo, la experiencia de usuario y la seguridad del mismo[67].

Concorde con los requerimientos y especificaciones de este sistema de control de laboratorio, se desarrolló una *interfaz gráfica de usuario* (o GUI por sus siglas en inglés) para facilitar su uso, mejorar la obtención y manejo de datos experimentales y permitir el cambio de controladores y sus constantes de forma rápida y sencilla. De esta manera los profesores, estudiantes y demás usuarios serán capaces de realizar los experimentos sin necesidad de conocer profundamente el código del microcontrolador e interaccionar de forma directa con él.

La GUI se desarrolló utilizando el lenguaje de programación de *Python*[68]. La construcción de la interfaz comienza con una instancia o ventana base conocida como *Root* o raíz y todos los demás elementos que la conforman se encuentran dentro de esta instancia. A estos se les conoce como *Widgets*, por ejemplo, los *frames* o marcos que funcionan como contenedores secundarios de forma similar a la raíz, los *Botones* para disparar eventos y realizar funciones específicas, las *etiquetas* para señalizar e ilustrar mediante texto, las *entradas de texto* para definir comandos, matrices y constantes directamente con el teclado y los indicadores que proporcionan información visual extra. Para la creación y disposición de estos elementos se usó la herramienta de *Tkinter*[69], la cual es un *binding* de la biblioteca Tcl/Tk utilizada en sistemas como Windows, Linux, MacOS e incluso PIC para crear interfaces gráficas. Tkinter se puede utilizar mediante Python. Además, se empleó *Matplotlib* para añadir gráficas de forma sencilla y monitorear el desempeño del controlador, incluyendo los cuatro estados definidos durante el modelado, las dos entradas de los motores y los errores entre el valor y la referencia de las variables a controlar.

La ubicación de los componentes en el móvil y el riesgo que implica un cable en esa zona fueron parte de los motivos por los que se decidió hacer la comunicación entre el microcontrolador y la interfaz de manera inalámbrica mediante Wifi, así se permite la manipulación del dispositivo de forma remota y se reduce el riesgo de accidentes al manipularlo. La interfaz funge como el cliente, mientras que el dispositivo toma el papel de servidor dentro del protocolo de comunicación TCP/IP, por lo que es posible conectarlo con distintas computadoras de forma sencilla. Sin embargo, sólo se permite una conexión a la vez para evitar conflictos entre instrucciones de distintos clientes.

El manejo y análisis de datos es importante para este tipo de experimentos, por esta razón, se añadió la posibilidad de exportar los datos y gráficas como archivos de excel, mediante las librerías *xlrd*[70] y *xlwt*[71] y así poder importarlos en distintos programas para realizar análisis más profundos como Matlab o R. También es posible abrir pruebas pasadas guardadas de esta forma en la misma interfaz.

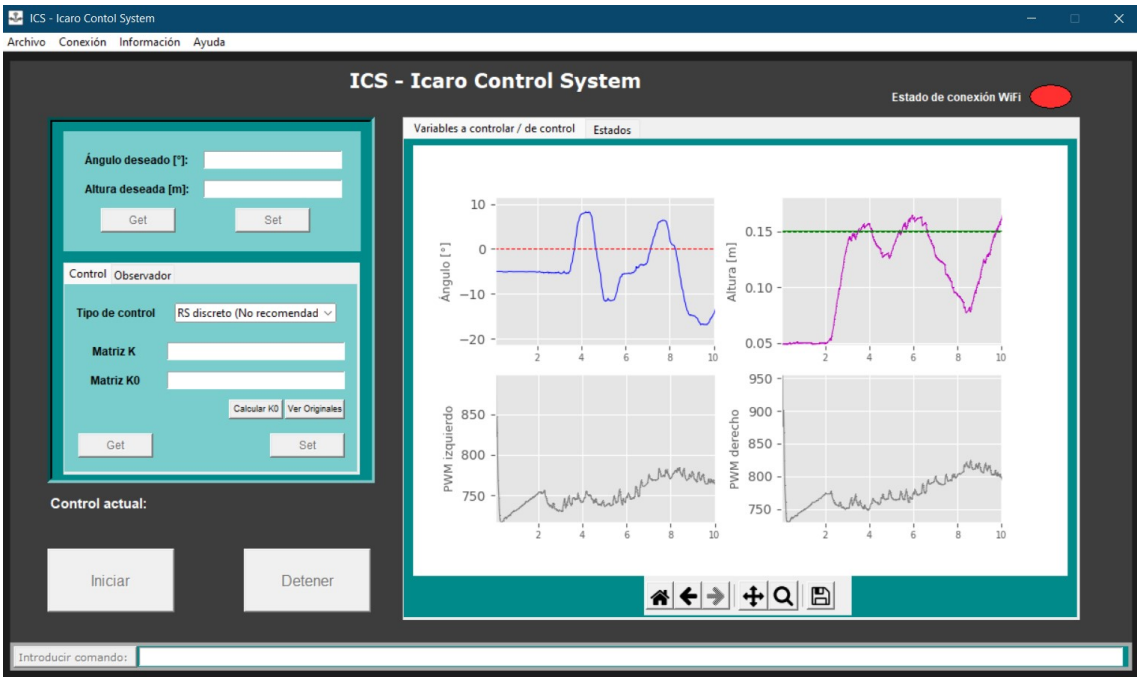


Figura 6.33: *Disposición general de la interfaz*

La interfaz de la Figura [6.33] está compuesta por varias secciones, cada una con un propósito diferente. La primera es la barra de menú en la parte superior, en ella se encuentran pestañas desplegables con los comandos para guardar, abrir y exportar los datos experimentales mencionados anteriormente. Los botones de conexión y desconexión entre el servidor (el microcontrolador) y el cliente (la computadora con la interfaz) también están en esta barra bajo la pestaña de conexión. Por último, es posible consultar información referente al funcionamiento del prototipo, tales como instrucciones de uso, troubleshooting (resolución de problemas), diagramas de bloques, diagramas eléctricos, etc.

La sección de control está dividida en dos marcos principales y se distingue por sus entradas y botones relacionados con el controlador activo en el dispositivo, además de dos grandes botones para iniciar y detener la prueba. El primer marco está compuesto por dos entradas de texto y dos botones con los cuales es posible modificar y conocer los valores deseados o de referencia de las variables a controlar (ángulo y altura). En el segundo marco se encuentran dos pestañas con estos mismos Widgets, que junto con un *combobox*, permiten conocer y seleccionar una estructura de control especifica de entre varias opciones para realizar el experimento. Las ganancias del controlador y observador pueden ser establecidas en esta misma sección o utilizar aquellas obtenidas en la sección de control de este trabajo.

La sección de mayor peso visual es la de monitoreo de variables, en ella se utilizan dos pestañas con cuatro figuras cada una donde se grafican las variables a controlar, sus valores deseados, las entradas de control y los estados del modelo. Esto ocurre siempre que el experimento esté en proceso después de presionar el botón de *Iniciar* de la sección anterior. También cuenta con una barra de herramientas para navegar de forma sencilla las gráficas, poder ajustar la visualización e incluso guardar una captura con lo que se muestra actualmente. Los datos graficados son los mismos que se guardan y exportan en la barra de menú en excel. Por último, en la parte inferior se encuentra la sección de comandos, cuya principal característica es una entrada de texto y un botón para introducirlos explícitamente que se envían de forma directa al microcontrolador.

Finalmente se utilizó la librería *PyInstaller*[72] para generar un ejecutable fácil de utilizar y distribuir. En los anexos G se encuentra un diagrama de flujo que ejemplifica de forma sencilla el funcionamiento general de la interfaz y la comunicación, de igual forma, el código original se puede encontrar y descargar con el enlace proporcionado en esta misma sección.

7. Fabricación de prototipo

7.1. Procesos de manufactura

En esta sección se describirán los procesos de manufactura usados para la construcción del dispositivo. Estos métodos de fabricación se definieron respecto a las propiedades de los materiales empleados, la geometría de las piezas, su función en el dispositivo y su costo, cada uno esta listado en la Tabla [7.1].

Para la fabricación del *frame*, se necesitó hacer el corte de elementos pequeños con un buen acabado para facilitar el ensamble, por lo que se utilizó el maquinado de *Router CNC*, una máquina que de acuerdo a un programa, es capaz de moverse y cortar simultáneamente en tres o más ejes mediante servomotores. Parámetros como la posición, revoluciones y velocidad de avance del cortador, se controlan por un conjunto de instrucciones de código generado por un post-procesador[73].

Otro método empleado fue la impresión 3D, que es un conjunto de tecnologías de *fabricación por adición*. La principal diferencia entre los métodos de impresión radica en la forma en la que se genera cada capa de material. La tecnología de extrusión ha sido el método más común en los últimos años, el *FDM* (modelado por deposición fundida) fabrica elementos mediante un filamento plástico almacenado en bobinas que se extruye a través de una boquilla con una temperatura superior a la de fusión del material y se mueve, por un control electrónico, con un patrón prescrito capa por capa. Después de que el material semi fundido es depositado en la mesa de trabajo, comienza a enfriarse y se une al resto del plástico de forma difusa[74]. Resulta conveniente producir de esta forma ciertas piezas de tamaño reducido que requieren ser rígidas y tener un bajo peso. Para lograrlo se aprovechó de una característica del software que remplazó el relleno solido en una estructura con panal de abeja para disminuir la densidad sin comprometer la rigidez.

Las piezas de la base se fabricaron mediante *corte láser*. Se eligió esta técnica principalmente por practicidad, aprovechamiento del material y su compatibilidad con este proceso. Consiste en enfocar un haz de luz en un punto del material que se desea tratar hasta que este alcance su temperatura de fusión, se derrita y evapore para lograr un corte. El efecto se genera al excitar las moléculas del medio activo, que son sustancias que emiten radiación láser (como el CO_2), mediante la aplicación de una señal de alta frecuencia, descargas eléctricas o impulsos de luz. El medio activo se encuentra entre dos espejos de diferentes características que conforman la cavidad resonante y provocan que la luz rebote y se amplifique[75]. Luego, el rayo es transportado por espejos desde el resonador hasta el cabezal de corte y enfocado por lentes en un área que abarca entre 0,1 a 0,3[mm] de diámetro[76].

Parte del dispositivo	Proceso	Características
Piezas del Frame	Maquinado Router CNC	Alta precisión del corte.
		Buen acabado de las piezas.
		Poco tiempo de fabricación.
		Permite maquinar una gran variedad de materiales y espesores.
Soportes de los componentes y correderas	Impresión 3D FDM (Modelado por deposición fundida)	Generación de geometrías complejas.
		Bajo costo de producción.
		Sin desperdicio de material.
		Propiedades de la pieza controladas.
Piezas de la base y tapa	Corte láser	Alta precisión del corte.
		Buen acabado de las piezas.
		Poco tiempo de fabricación.
		No requiere sujeción.
		No es necesario un tratamiento posterior.
		Buen aprovechamiento de material.

Tabla 7.1: *Procesos de manufactura empleados*

7.2. Ensamble

Una vez fabricadas las piezas, las partes de la base se pintaron con aerosol acrílico negro para protegerlas de la humedad y tener un mejor acabado. Para unir la base se usó cianoacrilato y tornillería, mientras que las piezas de ACP se pegaron con resina epoxi. Estos son adhesivos estructurales o de ingeniería; particularmente, la resina epóxica posee resistencia al calor, impacto y corrosión; puede trabajar en un rango de temperaturas entre -50 y 150[°C] y ofrece buena adherencia sobre multitud de materiales.

Las barras de acero se cortaron y fijaron sobre la base por medio de soportes de aluminio para varilla lisa y se procuró conservar su alineación para mejorar el recorrido de los baleros lineales. Posteriormente, se adhirieron rodamientos radiales con eje de 8[mm] a cada costado del *frame*. En las correderas de PLA se ajustaron los rodamientos lineales y se incrustó un eje del mismo diámetro interno que los rodamientos del *frame*.

En lo que respecta al montaje del resto de los componentes, la fuente de alimentación se montó y conectó en el interior de la base junto con un interruptor y un diodo LED indicador. Los motores, ESC, microcontrolador e IMU se atornillaron a la estructura del *frame* con ayuda de los soportes modelados e impresos específicamente para cada componente. Su cableado se llevó a cabo sobre la misma estructura mediante terminales eléctricas tipo bala, headers macho/hembra, abrazaderas y una placa fenólica con pines que sirven como nodos para las distintas terminales. Por otra parte, el sensor infrarrojo se atornilló en la parte inferior de una de las correderas de forma que sólo se mueve verticalmente.

Para terminar, las correderas de PLA se unieron al resto del *frame* con un eje y luego, este ensamble se montó sobre los cuatro postes (barras) encima de la base. Se añadieron resortes a cada barra para amortiguar la caída, collares de retención para restringir la altura máxima de desplazamiento y una tapa para mantener las barras alineadas en su posición.

La construcción requirió algunos ajustes debido a que se realizaron actividades de única ocasión, tales como: el pintado, limado de asperezas y rebabas, pegado, soldado de los componentes eléctricos, ponchado de terminales eléctricas y corrección de imperfecciones consecuencia de los procesos de fabricación. Los ensambles posteriores fueron posibles entre dos personas normalmente en un periodo de 4 a 6 horas, es por ello que se puede decir que el dispositivo cumple la especificación de diseño de tiempo de armado definida anteriormente.

8. Prototipo resultante

Después algunas iteraciones y ajustes al diseño, se obtuvo el prototipo que se muestra en la Figura [8.1]. Cuenta con dimensiones generales de 48x58x30[cm] y un peso alrededor de 4.7[Kg], mientras que las dimensiones del *frame* (balancín) son de 58x9x5[cm]con un peso de 0.54[Kg].

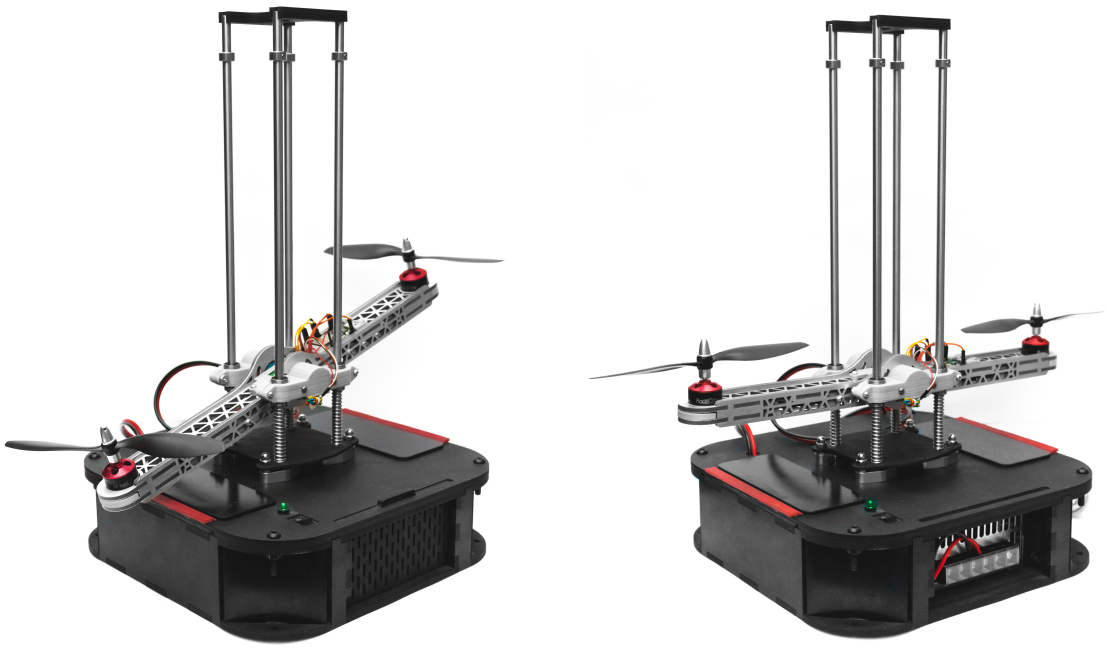


Figura 8.1: *Fotografía del prototipo resultante* [23]

El presupuesto de fabricación tomó en cuenta todos los componentes mecánicos, eléctricos-electrónicos y los costos de los procesos de manufactura. Algunos elementos como la fuente de alimentación, los actuadores y los variadores de velocidad, tuvieron un gran peso en este sentido. Debido a algunas modificaciones necesarias durante el desarrollo, el precio se incrementó ligeramente al planteado originalmente.

Durante las pruebas se observó que el flujo de aire en la fuente es suficiente y puede mantenerse encendida el tiempo propuesto en las especificaciones. Los motores mostraron un calentamiento notable cuando se requiere una potencia elevada durante un periodo prolongado; sin embargo, la lámina de aluminio del ACP ayuda a disipar la temperatura en toda la superficie.

El movimiento del modelo se percibe continuo y con poca fricción en el desplazamiento y la rotación. La estructura no presenta indicios de deformación permanente al recibir golpes y los componentes están firmemente sujetos, de manera que a pesar de sus notables vibraciones no se afecta la solidez del ensamble, aún por lapsos extensos.

Se logró la fabricación de un dispositivo compacto y robusto gracias a la elección de componentes con dimensiones pequeñas, a las propiedades de los materiales y a la geometría de cada pieza que permitió reducir el peso y mantener la rigidez estructural. Por otra parte, el peso del prototipo, permite que una persona pueda manipularlo y transportarlo, y a su vez es suficiente para otorgar mayor estabilidad al cuerpo al evitar que se desplace o se despegue de la superficie a causa del empuje de los motores cuando se encuentra en funcionamiento.

[23] Fotografías tomadas y editadas por David Alejandro Romero Rivas (2/Julio/2022)

"Atreverse es perder momentáneamente el camino, no atreverse es perderse a uno mismo"
Soren Kierkegaard

"Las mejores cosas en la vida se ocultan detrás del miedo"
Will smith, This is our planet

"Cuando miras hacia atrás en tu vida parece como si fuera una trama, pero cuando estás en ella es un desastre; sólo hay una sorpresa tras otra. Más tarde, ves que era perfecto"
Arthur Schopenhauer

Parte III

Diseño e implementación del control

Con el prototipo construido ahora se puede obtener e implementar técnicas de control automático para que el funcionamiento propuesto sea posible. Los objetivos particulares para esta sección son: Obtener un modelo matemático que aproxime el funcionamiento del sistema, diseñar una propuesta de control clásico para cumplir con la estabilización y regulación de la planta así como diseñar otra propuesta utilizando técnicas de control moderno que también cumplan con la estabilización y la regulación. Ambas propuestas deben ser simuladas para verificar su validez e implementadas en el prototipo.

9. Sistema de control

Un sistema de control consiste en subsistemas y procesos organizados e interconectados de tal manera que, a partir de una *entrada*, se obtenga una *salida* deseada y la mayoría de las veces, con un *desempeño* específico. Esta salida comúnmente es una variable física, tal como la temperatura, posición, voltaje, corriente, luminosidad, humedad, entre muchas otras. Se puede considerar como ejemplo un sistema de aire acondicionado, al que se llamará *planta*. Al seleccionar un nivel deseado se define una *entrada*, y al cambio de temperatura resultante en la habitación se le considera como una *salida*. La variación de la temperatura no debe ser muy drástica porque incomodaría a la persona, pero tampoco se busca que sea demasiado lenta, a este desarrollo se le conoce como *respuesta transitoria*. Por otra parte, a la temperatura final que alcanza la habitación se le nombra como *respuesta en estado estable*. Ambas respuestas representan el *desempeño* de nuestro sistema de control.

Según Nise (2011)[2], las dos configuraciones más conocidas de los controladores son los *Controladores de lazo abierto* y los *Controladores de lazo cerrado*, cuyos diagramas de bloques se muestran en la Figura [9.1]. Los primeros comúnmente empiezan con un transductor que, como su nombre indica, convierte la señal de entrada en el mismo tipo de señal utilizada por el controlador (puede ser eléctrica, mecánica, térmica, luz, etc.). Después este controlador emite una *señal de control* basado en la información de entrada, esta señal modifica o conduce el comportamiento de la planta para obtener una salida. En el ejemplo anterior se puede considerar el transductor de entrada como el control o perilla donde se selecciona el nivel deseado ya que este convierte la información en una señal eléctrica. Después el controlador modifica la magnitud de la señal al compresor y/o resistencias del aparato para que así la temperatura del aire cambie y poco a poco la del cuarto se modifique.

Entre cada uno de estos pasos pueden ocurrir perturbaciones o ruidos. Estos son señales indeseadas que no se pueden controlar y modifican el resultado de cada etapa provocando errores que se propagan por el sistema hasta que finalmente existe un error considerable en la salida. Los sistemas de lazo abierto son conocidos por ser baratos y fáciles de implementar; sin embargo, no son capaces de compensar estos errores, por ello son recomendados para sistemas que requieran un control muy simple.

Para compensar estos fallos surgen los sistemas de *control de lazo cerrado*. A estos sistemas se les suma un transductor en la salida, también llamado sensor, que mide el resultado obtenido. Este resultado se retroalimenta al controlador y se obtiene la diferencia entre el resultado esperado y el obtenido, conocida como *señal de error*, que a su vez genera una nueva señal de control para corregir el resultado. De esta forma se pueden compensar los ruidos y las perturbaciones. Además, ajustando los *parámetros de control* es posible modificar el desempeño de forma más amplia y flexible. Estos controladores son más complejos de implementar y normalmente requieren mayor equipo y planeación.

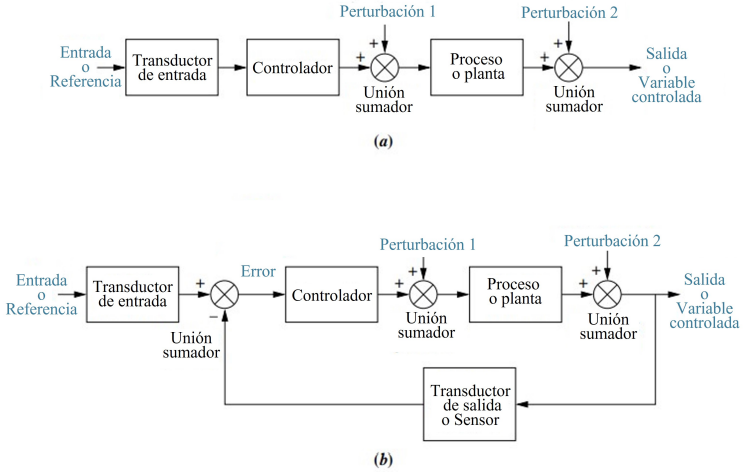


Figura 9.1: *Control de lazo abierto y de lazo cerrado* [24]

Para poder diseñar un sistema exitoso se deben tener claros los *objetivos de control*. Cuando no se cumple alguno de estos se le pueden agregar elementos que lo ayuden a cumplir dichas características requeridas.

El primer objetivo es la *estabilización*. Hablar de desempeño y errores en estado permanente es inútil si el sistema no es estable. Un sistema cumple con el objetivo de estabilización si su respuesta natural, es decir la forma en la que adquiere y disipa energía sin importar las estradas, decae a cero. Uno inestable tiende a acrecentar la respuesta natural hasta que esta supera la que se le forzó mediante el control, por ende, ya no está controlado e incluso puede dañarse o al usuario. Es posible inestabilizar sistemas naturalmente estables, sin embargo, en la mayoría de los casos la estabilización es un objetivo de control esencial.

De acuerdo con el mismo Nise (2011)[2], otra forma de definir a un sistema estable es si toda entrada limitada produce una salida limitada (BIBO - Bounded input bounded output); pero que la respuesta sea de este tipo no implica que su valor tiende a un valor deseado propuesto. Cuando se busca que la planta siga una referencia se habla de *regulación*. Además, cuando este valor deseado varia en el tiempo y se tiene que alcanzar en un lapso definido se le conoce como el objetivo de *seguimiento*. Este implica que la respuesta transitoria se comporte de una forma específica requerida. El diseño de este desempeño debe siempre ser acorde con las necesidades y capacidad del hardware y software utilizado.

Por último, es necesario tener en cuenta otros objetivos como la *robustez*, que significa diseñar el control de tal forma que sea capaz de funcionar a pesar de cambios inesperados en los parámetros previamente considerados como constantes sin que afecte demasiado el resultado. Además de otros factores como facilidad de implementación y costos.

10. Definición de planta y modelo matemático

10.1. Plantas y sistemas

Para poder realizar el análisis y diseñar el controlador primero se debe saber que es un sistema y considerar las características particulares del prototipo. La definición puede ser muy variada dependiendo del contexto en donde se trate de definir. Una de ella es considerarlo como un conjunto de elementos interrelacionados entre sí para lograr un objetivo deseado. Este sistema puede estar compuesto de subconjuntos llamados *subsistemas*, que a su vez están formados por *componentes* que son los elementos mínimos que lo conforman. En este proyecto, dichos componentes se pueden reducir a elementos mecánicos, eléctricos y electrónicos

[24] Nise, Norman. *Control Systems Engineering*. pág 8 - Traducido [2]

que se utilizaron en el proceso de diseño como lo son los motores, resistencias, baleros, ESC, soportes, pivotes, sensores, etc.

Para propósitos de análisis y diseño existe otra definición que dice que un sistema es un ente formado por un conjunto de entradas, un conjunto de salidas y una *relación* definida entre ambos conjuntos. En el caso de los sistemas físicos sus entradas y salidas son variables físicas como voltaje, posición, temperatura, etc.

Este concepto permite analizar el sistema de una forma más simple y definir algunos muy grandes, como un automóvil o un proceso de producción, como pequeños; por ejemplo, una bocina se representa con sus entradas como las señales eléctricas que recibe, sus salidas como las ondas mecánicas o sonido que produce y su relación está dada por la forma en que interactúan sus componentes tales como el imán, la membrana, el cono y la bobina.

Los sistemas físicos se pueden clasificar según su número de entradas y salidas, tanto si son múltiples como únicas (MIMO, SISO). Se pueden clasificar como *causales o no-causales* si cumplen el principio de causalidad que dice que todo efecto tiene una causa que lo provoca. Todos los sistemas físicos son causales. También se pueden clasificar como *estocásticos* si sus salidas son aleatorias o dependen de la probabilidad y *determinísticos* si la relación siempre se cumple de la misma forma. Otra clasificación es de *estáticos* si su salida actual depende solamente de su entrada actual o *dinámicos* si su salida actual depende de su entrada actual y sus entradas anteriores. También se pueden dividir en *variantes en el tiempo* o *invariantes en el tiempo* dependiendo de que sus propiedades inherentes como la masa, capacitancia, resistencia, volumen, entre otras, se mantengan constantes a lo largo del tiempo. Se pueden dividir de igual forma en sistemas *lineales* o *no lineales* dependiendo de su relación y en *continuos* o *discretos* dependiendo si su dinámica se da en el tiempo continuo o con un periodo definidos[77].

La representación del sistema propuesto se muestra en la Figura [10.1]; este se considera como uno de múltiples entradas y múltiples salidas (MIMO)[77] debido a las fuerzas de sustentación de cada uno de los motores, que actúan como entradas y los dos grados de libertad que tiene y se desean controlar (la altura y el ángulo de inclinación), que actúan como salidas. Es un sistema causal, ya que es físico, además de ser dinámico y determinístico. Debido a la compleja relación que tienen los componentes y otras consideraciones vistas más adelante en la Sección 10.2.1, se le considera también como no lineal e invariante en el tiempo.

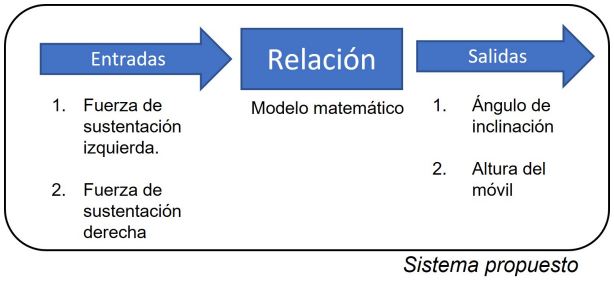


Figura 10.1: *Representación del sistema propuesto*

10.2. Modelado matemático

Ya que se establecieron las salidas y las entradas del sistema se necesita definir su relación para tener una representación completa. Esto se realiza mediante un modelo.

Definir un modelo, al igual que un sistema, depende del contexto en el que se esté trabajando, pero existen dos definiciones que pueden auxiliar:

> *"Para el ingeniero, modelo es un mecanismo mediante el cual se pueden aplicar técnicas analíticas en la solución de un problema práctico" (P.D. Smallwood)*

"[Models] Are simplified abstracted relations used to predict the behavior of a physical system, or certain aspects of the system's response to inputs" (D.C. Karnopp)
- [Los modelos] Son relaciones abstractas simplificadas utilizadas para predecir el comportamiento de un sistema físico, o ciertos aspectos de la respuesta del sistema a entradas -

A fin de cuentas, los modelos son versiones simplificadas del comportamiento real de un sistema, que se aproximan a describirlo de forma simple y suficiente como para poder aplicar técnicas analíticas sobre ellos. La suficiencia de esta descripción depende de la aplicación en específico. Los modelos pueden ser de muchos tipos como escala, virtual, gráfico, entre otros. Sin embargo, para este tipo de aplicaciones se utiliza el *modelo matemático*. Consiste en una o varias ecuaciones, normalmente diferenciales, que están basadas en los parámetros y leyes físicas que rigen al sistema. Este tipo de modelo puede ayudar al *análisis*, que se considera como el estudio del comportamiento para poder *predecirlo*, también a la *identificación* la cual ayuda a definir la relación entrada - salida y para la *síntesis o diseño*, que es donde se utiliza lo anterior para obtener las salidas deseadas.

Es importante mencionar que un sistema no sólo tiene un modelo válido, sino que puede tener varios que describan su comportamiento. El modelo obtenido puede variar según la metodología que se utilice, o la simplicidad que se considere darle. Puede ser tan complicado como uno sea capaz de hacerlo o tan simple como se desee, sin embargo, puede que uno demasiado simple no represente fielmente el comportamiento real del sistema. Por otra parte, puede ser que uno sumamente complejo sea difícil de analizar e identificar y por ello no sea viable para una etapa de diseño. Es importante buscar un modelo equilibrado para poder dar cierta eficacia al diseño además de siempre estar conscientes que ninguno puede representar completamente a un sistema físico. La superioridad de uno sobre otro en este aspecto solo puede saberse midiendo experimentalmente sus resultados.

Un procedimiento general para la obtención del modelo matemático puede ser descrito de la siguiente forma:

1. Dibujar un esquemático, como por ejemplo, un diagrama de cuerpo libre y definir las variables involucradas.
2. Escribir las ecuaciones que describen a cada componente y combinarlas de acuerdo con el esquemático anteriormente realizado. Está combinación debe estar basada siempre en leyes físicas.
3. Se puede verificar la validez del modelo con experimentos físicos o simulaciones computarizadas.

En este proyecto se utilizó el método de Euler-Lagrange[78] para obtener el modelo matemático. Este método, comúnmente utilizado para sistemas mecánicos pero no exclusivo de ellos, utiliza la energía mecánica para obtener las ecuaciones de movimiento de sistemas compuestos por múltiples cuerpos rígidos. Estas ecuaciones están basadas en el principio de D'Alembert y las leyes de Newton.

10.2.1. Desarrollo matemático

Antes de poder diseñar y aplicar un método de control a la planta es necesario definir un modelo que nos permita *analizar* y *predecir* el comportamiento que este tiene.

Siguiendo los pasos generales de la sección anterior, primero se redujo el sistema a un diagrama de cuerpo libre como se muestra en la Figura [10.2]. El significado de cada una de las variables está definido en la Tabla [10.1]. En este diagrama se muestran las fuerzas que actúan sobre cada una de las masas principales y las variables de interés que son el ángulo de inclinación y la altura. Es importante notar que el tipo y dirección de movimiento que puede desarrollar está restringido por elementos considerados como rígidos. Por ejemplo la masa m_3 solo puede desarrollar un movimiento traslacional en dirección del eje vertical, mientras que m_1 y m_2 pueden moverse en ambos ejes siempre y cuando se respete la restricción de la barra central rotatoria.

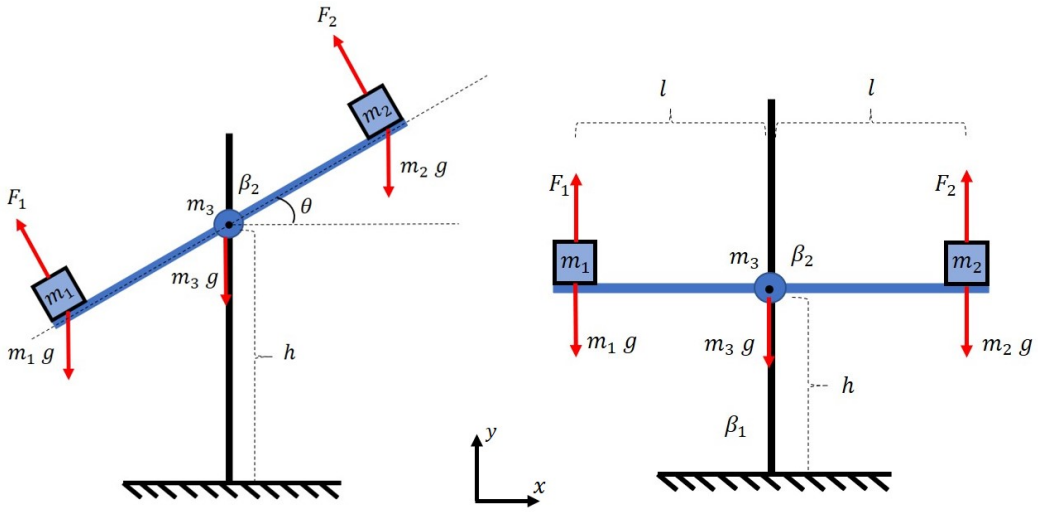


Figura 10.2: *Diagrama de cuerpo libre del prototipo*

F_1	var $[N]$	Fuerza de sustentación generada por el motor izquierdo
F_2	var $[N]$	Fuerza de sustentación generada por el motor derecho
m_1	0.0837 $[Kg]$	Masa equivalente del conjunto del motor izquierdo
m_2	0.0932 $[Kg]$	Masa equivalente del conjunto del motor derecho
m_3	0.360 $[Kg]$	Masa del balancín y sus componentes
g	9.78 $[m/s^2]$	Aceleración producida por la gravedad
β_1	aprox 0.0	Fricción lineal del móvil
β_2	aprox 0.0	Fricción rotacional del balancín
h	var $[m]$	Altura del móvil
θ	var $[rad]$	ángulo del balancín

Tabla 10.1: Constantes y variables del modelo matemático

Como se mencionó antes, el método seleccionado para obtener el modelo matemático es el de Euler-Lagrange. En este el comportamiento del sistema está descrito por (10.1) llamada *Ecuación Euler-Lagrange*. En esta expresión L es el Lagrangiano, descrito como la diferencia de la energía cinética E_c y la potencial E_p tal como se muestra en la ecuación (10.2), mientras que P_d es la potencia disipada por el sistema, E_i son las entradas que afectan directamente al sistema y q_i son las coordenadas generalizadas.

$$\frac{\partial}{\partial t}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} + \frac{\partial P_d}{\partial \dot{q}_i} = E_i \tag{10.1}$$

$$L = E_C - E_P \tag{10.2}$$

Para poder utilizar estas expresiones se deben definir las coordenadas generalizadas q_i. El número de coordenadas depende de la cantidad de grados de libertad que tiene el sistema, es decir, variables independientes entre si. En este caso son dos, la altura h y el ángulo θ lo cual nos dice que nuestro sistema tiene dos GDL. Las coordenadas generalizadas pueden ser entonces estas mismas variables. Es importante mencionar que todas las expresiones desarrolladas para construir la ecuación Euler-Lagrange deben estar en función de dichas coordenadas y de sus entradas.

$$q_1 = h, \quad q_2 = \theta$$

Para el caso de las energías cinética (10.3) y potencial (10.4), estas se presentan en cada uno de los elementos con masa de nuestro diagrama, entonces la energía total será la suma de cada una de las energías en cada elemento como se muestra en las ecuaciones (10.5) y (10.6)

$$E_{ci} = \frac{1}{2}mv^2 \tag{10.3}$$

$$E_{pi} = mgh \tag{10.4}$$

$$E_c = E_{c1} + E_{c2} + E_{c3} \tag{10.5}$$

$$E_p = E_{p1} + E_{p2} + E_{p3} \tag{10.6}$$

Para calcular la velocidad de la masa m_1 es necesario sumar las velocidades de su movimiento de traslación y de rotación como se muestra en la Figura [10.3]. Después de obtener dicha velocidad, esta se puede sustituir en (10.3) y expresarse con (10.7).

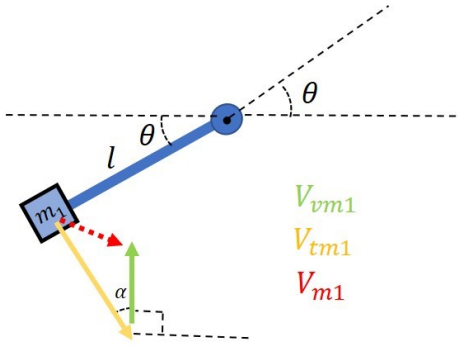


Figura 10.3: *Descomposición de los componentes vectoriales de la velocidad para* m_1

$$\begin{aligned}
&V_{vm1} = \dot{q_1} \\
&V_{tm1} = l\dot{q_2} \\
&V_{m1}^2 = V_{tm1}^2 + V_{vm1}^2 - 2V_{tm1}V_{vm1}\cos(\alpha) \\
&\alpha = \theta \\
&E_{c1} = \frac{1}{2}m_1\left[\dot{q_1}^2 + l^2\dot{q_2}^2 - 2l\dot{q_1}\dot{q_2}\cos(q_2)\right]
\end{aligned} \tag{10.7}$$

El caso de la velocidad de m_2 es muy similar la de m_1, así que se sigue el mismo procedimiento utilizando el diagrama de la Figura [10.4]. De esta forma se obtiene la energía cinética E_{c2} expresada en (10.8).

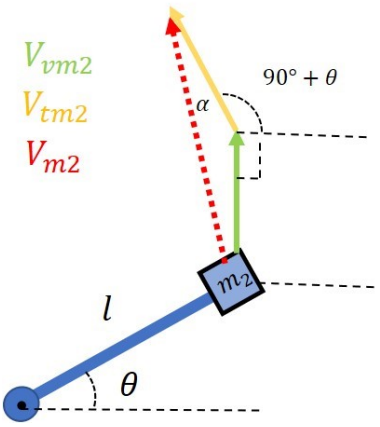


Figura 10.4: *Descomposición de los componentes vectoriales de la velocidad para* m_2

$$
\begin{aligned}
V_{vm2} &= \dot{q_1} \\
V_{tm2} &= l\dot{q_2} \\
V_{m2}^2 &= V_{tm2}^2 + V_{vm2}^2 - 2V_{tm2}V_{vm2}\cos(\alpha) \\
\alpha &= 180^\circ - \theta \\
E_{c2} &= \frac{1}{2}m_2\left[\dot{q_1}^2 + l^2\dot{q_2}^2 + 2l\dot{q_1}\dot{q_2}\cos(q_2)\right]
\end{aligned} \tag{10.8}
$$

La masa m_3 cuenta únicamente con movimiento vertical por lo que solo se deben sustituir los valores en (10.3) para obtener su energía cinética, expresada mediante (10.9).

$$E_{c3} = \frac{1}{2}m_3\dot{q_1}^2 \tag{10.9}$$

Se sustituyen las ecuaciones (10.7), (10.8) y (10.9) en (10.5) para obtener la ecuación que representa la energía cinética del sistema.

$$E_c = \frac{1}{2}\left[\dot{q_1}^2 + l^2\dot{q_2}^2 - 2l\dot{q_1}\dot{q_2}\cos(q_2)\right] + \frac{1}{2}m_2\left[\dot{q_1}^2 + l^2\dot{q_2}^2 + 2l\dot{q_1}\dot{q_2}\cos(q_2)\right] + \frac{1}{2}m_3\dot{q_1}^2 \tag{10.10}$$

Para obtener la energía potencial E_p del sistema solo se deben de tomar en cuenta las alturas que varían para cada una de las masas y ponerlas en la ecuación (10.4). Estas ecuaciones se deducen de la Figura [10.2] y una vez obtenidas se sustituyen en (10.6) para obtener la energía potencial.

$$E_{p1} = m_1 g(q_1 - l\sin(q_2)) \tag{10.11}$$

$$E_{p2} = m_2 g(q_1 + l\sin(q_2)) \tag{10.12}$$

$$E_{p3} = m_3 g q_1 \tag{10.13}$$

$$E_p = m_1 g(q_1 - l\sin(q_2)) + m_2 g(q_1 + l\sin(q_2)) + m_3 g q_1 \tag{10.14}$$

El siguiente elemento por obtener es la potencia disipada P_d. Esta representa todas las pérdidas de energía que sufre el modelo. En plantas más complejas estas pérdidas pueden ser térmicas, eléctricas o mecánicas como vibraciones o fricciones. En este caso se considera que las fricciones lineal y rotacional son las que disipan energía. Las fricciones pueden ser modeladas como amortiguamientos y la energía que desarrollan se representa mediante la función de disipación de Rayleigh descrita como F_{dp} en (10.15). Al sumar el efecto de ambas fricciones se puede describir la potencia disipada con (10.16).

$$F_{dp} = \sum_{i=1}^{n} \left(\frac{1}{2} k_i \hat{v}_i^{\,2} \right) \tag{10.15}$$

$$P_d = \frac{1}{2}\beta_1 \dot{q_1}^{\,2} + \frac{1}{2}\beta_2 \dot{q_2}^{\,2} \tag{10.16}$$

Después se obtienen las entradas externas E_i que, en este caso, son fuerzas o pares externos que afectan directamente a las coordenadas generalizadas. Los componentes verticales de las fuerzas producidas por los motores afectan directamente a la altura y el par que se produce entre los motores y el pivote central afecta al ángulo de inclinación. Estos dos fenómenos se representan por las ecuaciones (10.17) y (10.18).

$$E_1 = F_1 \cos(q_2) + F_2 \cos(q_2) \tag{10.17}$$

$$E_2 = F_2 l - F_1 l \tag{10.18}$$

Ya que se tienen E_c en (10.10), E_p en (10.14), P_d en (10.16) y las entradas E_i en (10.17) y (10.18), se pueden construir las ecuaciones que describen el modelo. Para ello se sustituyen estas expresiones en (10.1) y (10.2) y se realizan las derivadas parciales involucradas. Por último, para simplificar su manejo más adelante, se despejan las ecuaciones obtenidas respecto a sus derivadas de mayor orden y se sustituyen las coordenadas generalizadas por sus variables originales.

$$\begin{aligned}\ddot{\theta} = \Bigg[l \Big(& -F_2(3m_1 + m_2 + 2m_3) + F_1(m_1 + 3m_2 + 2m_3) + (m_1 - m_2)\Big[2\dot{h}\beta_1 \cos(\theta) \\ & - (F_1 + F_2)\cos(2\theta) + l(m_1 - m_2)\dot{\theta}^2 \sin(2\theta)\Big]\Big)\Bigg] \div \Bigg[-l^2\Big(m_1^2 + 6m_1 m_2 + m_2^2 \\ & + 2(m_1 + m_2)m_3\Big) - 2(m_1 + m_2 + m_3)\beta_2 + l^2(m_1 - m_2)^2 \cos(2\theta)\Bigg]\end{aligned} \tag{10.19}$$

$$\begin{aligned}\ddot{h} = \Bigg[& \sec(\theta)^2 \Big(-gl^2(m_1^2 + 6m_1 m_2 + m_2^2 + 2(m_1 + m_2)m_3) - 2g(m_1 + m_2 + m_3)\beta_2 \\ & - 2\dot{h}\beta_1(l^2(m_1 + m_2) + \beta_2) + 2(2l^2(F_2 m_1 + F_1 m_2) + (F_1 + F_2)\beta_2)\cos(\theta) + \\ & l(m_1 - m_2)\Big[gl(m_1 - m_2)\cos(2\theta) - 2\big(l^2(m_1 + m_2) + \beta_2\big)\dot{\theta}^2 \sin(\theta)\Big]\Big)\Bigg] \\ & \div \Bigg[-2l^2(m_1 - m_2)^2 + 2(m_1 + m_2 + m_3)\big(l^2(m_1 + m_2) + \beta_2\big)\sec(\theta)^2\Bigg]\end{aligned} \tag{10.20}$$

El modelo matemático obtenido es uno de cuarto orden debido a que está compuesto por dos ecuaciones diferenciales de segundo orden. No lineal ya que existen coeficientes y funciones trascendentes que involucran a las variables independientes como argumento, además de que existen derivadas con exponentes diferentes a uno.

Para verificar que el modelado se hizo de forma correcta, se realizó un diagrama de bloques en Simulink, mostrado en la Figura [10.5] donde se simuló su respuesta ante distintas entradas y partiendo de distintas condiciones iniciales. Los bloques de función contienen a (10.19) y (10.20), mientras que los integradores se utilizan para obtener las derivadas de menor grado de dichas variables. Este mismo diagrama servirá para simular la respuesta de los controladores sobre el sistema.

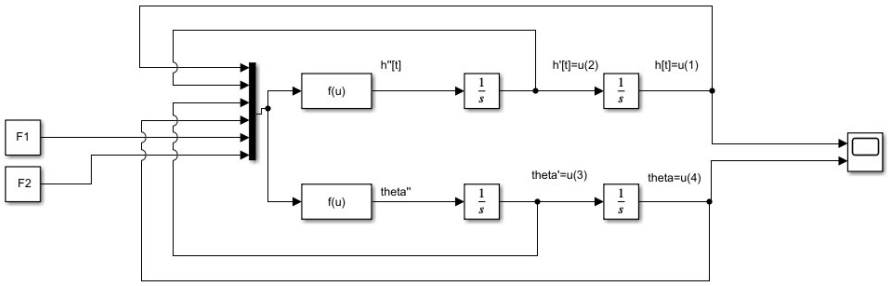


Figura 10.5: *Diagrama de bloques (Simulink): Simulación del modelo matemático*

11. Control clásico (dominio de la frecuencia)

Para desarrollar el controlador, se recurrió a las primeras técnicas de control realimentado que surgieron durante el siglo pasado cuando se comenzaron a aplicar los conocimientos adquiridos en problemas de comunicación y amplificación de señales a los problemas de control industrial[1]. Estas técnicas son pertenecientes al control clásico y se basan fuertemente en principios como la transformada de Laplace, la transformada de Fourier y la descripción externa de los sistemas. De esta misma corriente surge uno de los controladores más utilizados hasta el momento y primera propuesta de controlador para el prototipo: el PID.

11.1. Controlador Proporcional Integral Derivativo

El controlador PID (Proporcional - Integral - Derivativo) es uno de los controladores clásicos más conocidos e implementados en la industria, proyectos y prototipos debido a su robustez y facilidad de comprensión. Se utiliza comúnmente para sistemas SISO y, de preferencia, lineales ya que comúnmente se utilizan funciones de transferencia [2].

Es un tipo de controlador de lazo cerrado que realimenta la salida y la compara con la entrada para obtener un valor de error que pasa por tres etapas. La primera multiplica el error por una constante, la siguiente integra el error en el tiempo y la última deriva el error en el tiempo. La suma de estas tres etapas resulta en la ley de control que regirá al sistema. Ésta se puede representar en el dominio del tiempo y en el dominio transformado de Laplace como se muestra en las ecuaciones (11.1) y (11.2), correspondientemente.

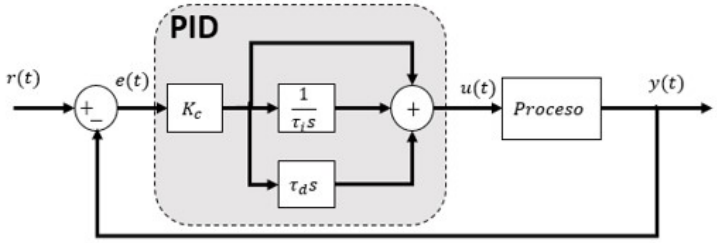


Figura 11.1: *Diagrama de bloques del controlador PID ideal* [25]

$$u(t) = k_c e(t) + \frac{k_c}{\tau_i}\int_0^t e(t)dt + k_c\tau_d\frac{de(t)}{dt} \tag{11.1}$$

$$G(s) = k_c\left(1 + \frac{1}{\tau_i s} + \tau_d s\right) \tag{11.2}$$

[25] `https://controlautomaticoeducacion.com/control-realimentado/controladores-pid-discreto/` (14/Febrero/2022)

La representación de la Figura [11.1] se conoce como representación *ideal*, pero existe otra forma común de representar el PID conocida como *paralela*. En ésta se le asigna una constante a cada etapa, como se muestra en (11.3). Ambas representaciones son equivalentes y se pueden obtener las constantes correspondientes para cada una, con base en la otra, utilizando las ecuaciones de equivalencia (11.4).

$$G_p(s) = k_p + \frac{k_i}{s} + k_d s \tag{11.3}$$

$$\begin{aligned} k_p &= k_c \\ k_i &= \frac{k_c}{\tau_i} \\ k_d &= k_c \tau_d \end{aligned} \tag{11.4}$$

Sin importar la representación que se emplee, estas constantes afectan de manera diferente al controlador. Dependiendo el valor que se le dé a cada una puede variar la velocidad de asentamiento, modificar el porcentaje de sobrepaso, cambiar el comportamiento del sistema completo a subamortiguado, sobreamortiguado o incluso inestable.

La constante proporcional, como su nombre lo dice, aplica una señal de control proporcional al error; cuando este es grande, la señal de control es mayor y lo contrario sucede cuando el error es pequeño. Suponiendo que el sistema tiene límites fijos y la respuesta que presenta tiene un límite tanto inferior como superior, la relación entre el error y la señal de control proporcionada por un control meramente proporcional se puede representar mediante la Figura [11.2].

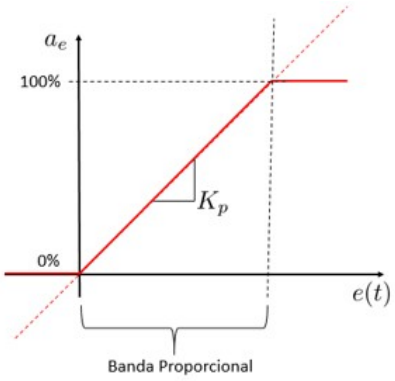


Figura 11.2: *Banda proporcional donde existe control.*

La constante k_p (o k_c) indica la pendiente de esta recta proporcional[79]. Entre mayor sea, más inclinada será la rampa, es decir, que ante errores pequeños la respuesta de control será más grande. Un valor alto en esta constante resulta en un control más rápido, agresivo, con mayor sobrepaso y más oscilaciones. Por el contrario, un valor bajo promueve un control lento, con pocas oscilaciones, poco sobrepaso y un gran error en estado permanente; por ello se debe regular adecuadamente para no comprometer el sistema. Es importante notar que un valor mayor implica que el error necesario para llegar a los límites físicos del sistema y saturarlo, es más pequeño; en muchos casos esto puede resultar en el daño de dichos sistemas. Además, una vez saturado, no se considera que exista control realimentado sobre el sistema debido a que la señal de control no cambia, aunque el error si lo haga.

Un controlador únicamente proporcional logra acercar el sistema al punto de referencia debido a esta acción inmediata correctiva, sin embargo, el error en estado estacionario no es eliminado. Se puede reducir proponiendo un valor muy alto en k_p, pero esto puede acortar la banda proporcional de forma impráctica.

La parte integral del controlador sirve como una "memoria"de los errores del sistema[80]. Estos de acumulan en cada ciclo con los pasados, lo cual permite generar una acción correctiva cuando el estado estacionario alcanzado por el control proporcional tiene un error constante respecto a la referencia. El valor de la constante k_i ($\frac{1}{\tau_i}$) determina que tan rápido se espera que el sistema alcance el error nulo. Una constante alta implica una corrección más veloz pero también ocasiona un mayor sobrepaso y oscilaciones.

Por último, la etapa derivativa es la encargada de anticipar del comportamiento futuro que va a tener la señal de error del sistema y tomar una acción correctiva preventiva[81]. Esta predicción se hace mediante la derivada del error en el instante de tiempo actual, que se puede interpretar como la tangente en ese punto que traza una proyección hacia el futuro posible, es decir, que aproxima la función de error mediante una recta en ese instante. Un valor demasiado grande en la constante k_d (τ_d) se puede interpretar como una estimación muy a futuro que, por lo general, puede ser bastante errónea, mientras que un valor muy pequeño no se anticipa lo suficiente y por ello no produce un resultado significativo en la señal del sistema. En general aumentar la acción derivativa de forma prudente reduce las oscilaciones y el sobrepaso. La representación gráfica de las tres etapas del controlador PID se encuentra en la Figura [11.3].

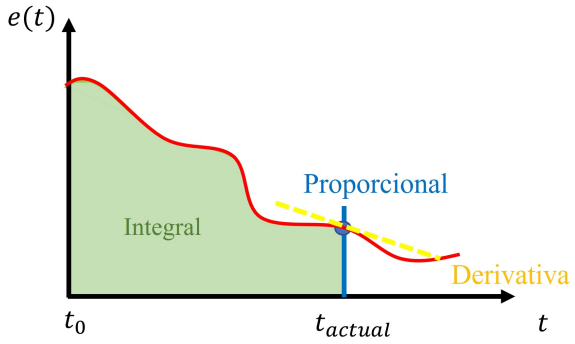


Figura 11.3: *Representación de la acción de cada constante del PID*

Antes de implementar este tipo de controladores es importante tener en cuenta algunos factores muy importantes que pueden hacer fracasar el control. El primero es el ruido de las mediciones de la salida del controlador. Un sensado ruidoso significa que la señal de error tendrá altibajos aleatorios, lo que puede provocar un control inconsistente, especialmente si el valor de la constante derivativa es alto. Cuando la señal de salida es demasiado ruidosa las predicciones de la etapa derivativa serán imprecisas e inexactas; lo cual provoca que, amplificadas por la constante derivativa, se produzcan señales de control bruscas y aleatorias en el sistema que, si son muy prominentes, pueden desestabilizarlo o dañarlo.

El segundo factor es el efecto *Wind-up*[82] provocado por la acción integradora. Se le conoce como Wind-up a la acumulación excesiva del error en el tiempo, entre mayor sea la constante integral más notorio será el efecto. Para explicarlo mejor, supongamos que tenemos un sistema real controlado por PID. Es bien sabido que los sistemas reales tienen límites físicos, por ejemplo, un motor tiene una velocidad limitada, una válvula no puede abrir más del %100 o un pistón no se puede extender más allá de sus dimensiones. Dependiendo del desempeño propuesto es probable que se alcancen estos límites y el sistema se sature. En este momento deja de existir un control de lazo cerrado debido a que el sistema se habrá salido de la banda proporcional y no podrá aplicar una acción mayor o menor de control. Este efecto se mantiene hasta que el sistema se acerque a la referencia lo suficiente y vuelva a entrar en la banda proporcional. Sin embargo, durante esta saturación, el error acumulado y la señal de control pueden seguir creciendo fuera de los límites físicos. Esto puede provocar que el sistema se quede saturado más de lo esperado y que al regresar cerca de la referencia la saturación de una señal de control reactiva excesiva debido a que se debe de descargar todo el error acumulado. Al implementar un controlador con acción integral se puede considerar diseñar algún efecto *Anti wind-up* que impida que se acumule el error cuando la señal de control está fuera de los límites físicos o que descargue la acción integral al llegar a la saturación del sistema.

Otra posible respuesta indeseada en este tipo de controladores es la llamada *Derivative Kick*[83] o patada

derivativa, la cual se produce principalmente cuando ocurre un cambio de referencia. Mientras que esta se mantenga constante, la función del error y su derivada se mantienen consistentes, pero en el momento en el que esta cambia, existe una diferencia importante en un periodo muy corto de tiempo. En teoría, en un sistema continuo, la derivada de un cambio de este estilo aproxima al infinito durante un instante. En la práctica, se obtiene un pico con un valor muy grande sobre la acción de control durante un periodo corto de tiempo debido a la acción derivativa. Esto es especialmente riesgoso en sistemas que involucren componentes mecánicos ya que puede desgastarlos más rápido o incluso comprometerlos. Para contrarrestar esto es común ejercer la acción derivativa directamente sobre la variable sensada ya que está no sufre cambios bruscos al cambiar la referencia. Otra forma de evitar este problema es evitar hacer los cambios de referencia de forma instantánea al modificar paulatinamente el valor hasta llegar a la referencia deseada.

11.2. Modelado de sistemas desacoplados

Antes de comenzar a diseñar un controlador PID para nuestro sistema, es importante tener en cuenta que solamente considera una entrada y una salida, es decir, que sólo se puede implementar en sistemas SISO, además de que es una técnica utilizada para sistemas lineales. Estas son dos características que no cumple el modelo matemático del sistema de estudio, anteriormente obtenido en la Sección 10.2.1. Pero existe una alternativa del sistema de estudio que permite utilizar este controlador; se puede descomponer este complejo modelo en subsistemas más simples que igualmente describan el comportamiento del sistema real. Al tener la libertad de cómo se lleva a cabo esta descomposición se pueden hacer tan simples como se desee, tratando de no caer en una sobre simplificación.

Al descomponerlo en dos sistemas SISO, donde cada uno tiene como salida una de las variables a controlar, es posible resolver el control de estas si se realiza un controlador separado para cada subsistema. Ambos funcionarán simultanea e independientemente. De esta manera, el diseño se vuelve más sencillo y amigable con sintonizaciones metodológicas o experimentales.

11.2.1. Sistema traslacional

El primer sistema, representado en la Figura [11.4] será traslacional y tendrá como salida la variable de la altura *h*. Cada una de las variables del diagrama se encuentra especificada en la Tabla [11.1].

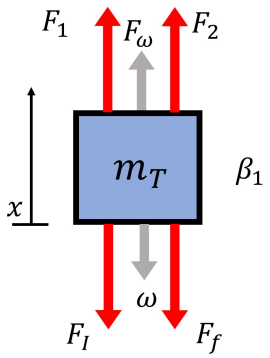


Figura 11.4: *Diagrama de cuerpo libre del sistema traslacional*

m_T	0.5369 $[Kg]$	Masa total del dispositivo móvil
F_1	var $[N]$	Fuerza de sustentación del motor izquierdo
F_2	var $[N]$	Fuerza de sustentación del motor derecho
x	var $[m]$	Distancia recorrida
F_ω	5.2506 $[N]$	Fuerza de equilibrio que contrarresta el peso
F_I	var $[N]$	Fuerza generada por inercia
ω	-5.2506 $[N]$	Peso del dispositivo
F_f	var $[N]$	Fuerza generada por fricción
β_1	aprox 0.0	Fricción lineal del móvil

Tabla 11.1: Variables del modelo traslacional

Como se quiere simplificar el dispositivo a sistemas de una sola entrada y salida se deben reducir las dos fuerzas originales de los motores a una sola entrada. Para ello se podría considerar la suma de las componentes verticales de cada una de las fuerzas como entrada, sin embargo, en este modelo simplificado propuesto no se considera ninguna inclinación, así que solamente se toma en cuenta la entrada como la suma de las fuerzas de sustentación que proporciona cada motor y se le llama F, como se muestra en la ecuación (11.5). Debido a que las entradas F_1 y F_2 pueden variar y se quiere que este subsistema se encuentre en equilibrio, por ahora se considerará que ambas fuerzas de los motores son iguales. De esta manera se pueden definir las ecuaciones constitutivas del modelo matemático como se muestra a continuación:

$$2F = F_1 + F_2 \tag{11.5}$$

$$m_T = m_1 + m_2 + m_3 = cte. \tag{11.6}$$

$$F_f = \beta_1 \dot{x} \tag{11.7}$$

$$F_I = m_T \ddot{x} \tag{11.8}$$

Para obtener las ecuaciones de equilibrio, se utiliza la tercera ley de Newton: A toda acción corresponde una reacción de igual magnitud y en sentido opuesto, y el principio d'Alembert: Las fuerzas aplicadas a un elemento junto con las fuerzas de inercia forman un sistema en equilibrio. Como el sistema que se quiere modelar debe estar en equilibrio, se asume que F_ω y ω tienen la misma magnitud y se construye la ecuación de equilibrio (11.9).

$$F_f + F_I = 2F \tag{11.9}$$

Al sustituir las ecuaciones constitutivas (11.7) y (11.8) se obtiene un modelo matemático lineal de segundo orden del subsistema desacoplado traslacional, que se muestra en la ecuación (11.10).

$$F = \beta_1 \dot{x} + m_T \ddot{x} \tag{11.10}$$

Este modelo se puede representar, si se utiliza la transformada de Laplace, como una función de transferencia como se muestra en (11.11). Con el propósito de tener mayor libertad para asignar sus condiciones iniciales en las simulaciones, también se decidió representarlo en variables de estado considerando $x_1 = x$, $x_2 = \dot{x}$, $u = F$ tal como se muestra en (11.12). Más adelante, en la propuesta de control moderno de la Sección 12, se profundiza sobre este tipo de representaciones.

$$G_1(S) = \frac{\frac{2}{m_T}}{S^2 + \frac{\beta_1}{m_T} S} \tag{11.11}$$

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{\beta_1}{m_T} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{2}{m_T} \end{bmatrix} u \tag{11.12}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [0]u$$

11.2.2. Sistema rotacional

El segundo sistema, representado en la Figura [11.5], es rotacional y tiene como variable de salida el ángulo de inclinación. Sus variables se especifican en la Tabla [11.2].

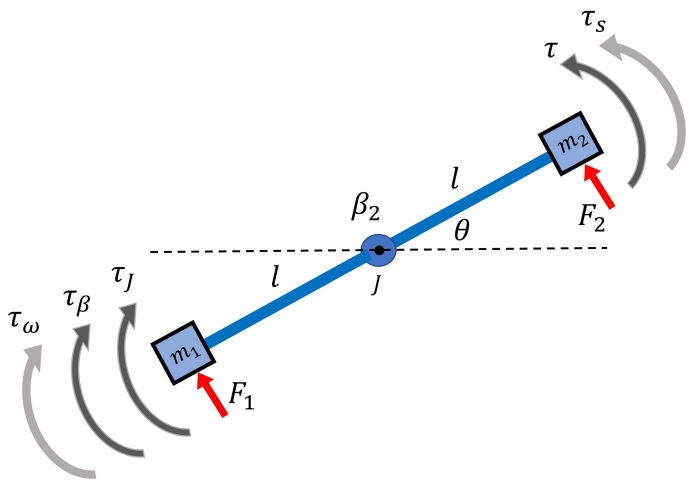


Figura 11.5: *Diagrama de cuerpo libre del sistema rotacional*

m_1	0.08 $[Kg]$	Masa del motor 1
m_2	0.13 $[Kg]$	Masa del motor 2
l	0.185 $[m]$	Largo del brazo
J	0.007187 $[Kgm^2]$	Inercia rotacional
θ	var $[rad]$	Ángulo de inclinación
τ_ω	cte $[Nm]$	Par ocasionado por la diferencia en las masas
τ_β	cte $[Nm]$	Par ocasionado por la fricción
τ_J	var $[Nm]$	Par ocasionado por la inercia rotacional
τ_s	var $[Nm]$	Par que mantiene el sistema en equilibrio
τ	var $[Nm]$	Par entrada
β_1	aprox 0.0 $[Nm]$	Fricción rotacional
F_1	var $[N]$	Fuerza de sustentación del motor izquierdo
F_2	var $[N]$	Fuerza de sustentación del motor derecho

Tabla 11.2: Variables del modelo rotacional

De igual forma que en el subsistema anterior, se pueden simplificar las dos fuerzas de los motores a una sola entrada, sólo que esta vez es se considera como la diferencia entre ellas que produce un par, llamada ΔF_τ como se muestra en (11.14). Las ecuaciones constitutivas de este modelo son las siguientes:

$$\Delta F_\tau = F_2 - F_1 \tag{11.13}$$
$$\tau = (\Delta F_\tau) l \tag{11.14}$$
$$\tau_\beta = \beta_2 \dot{\theta} \tag{11.15}$$
$$\tau_J = J\ddot{\theta} \tag{11.16}$$
$$\tag{11.17}$$

Para poder modelar al subsistema como un sistema en equilibrio se asume que el los pares τ_ω y τ_s tienen la misma magnitud. Se pueden utilizar los mismos principios de Newton y d'Alembert de equilibrio mecánico que se encuentran en el sistema traslacional, para así construir la ecuación de equilibrio (11.18).

$$\tau = \tau_J + \tau_\beta \tag{11.18}$$

Al sustituir las ecuaciones constitutivas (11.15), (11.16) y (11.14) en la ecuación de equilibrio (11.18) se obtiene un modelo matemático lineal de segundo orden para subsistema desacoplado rotacional.

$$\Delta F_\tau = \frac{J}{l}\ddot{\theta} + \frac{\beta_2}{l} \tag{11.19}$$

De igual manera, se puede representar como función de transferencia, presentada en (11.20) y en variables de estado considerando $x_1 = \theta$, $x_2 = \dot{\theta}$ y $u = \Delta F_\tau$ como se muestra en (11.21).

$$G_2(S) = \frac{\frac{l}{J}}{S^2 + \frac{\beta_2}{J}S} \tag{11.20}$$

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{\beta_2}{J} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{l}{J} \end{bmatrix} u \tag{11.21}$$
$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [0]u$$

11.3. Sobre el punto de equilibrio

Durante las secciones de control siguientes se utiliza el término *Punto de equilibrio* de forma frecuente, para facilitar la comprensión de las gráficas y diagramas más adelante es necesario comentar un poco sobre él y como se utilizó.

En el caso de las ecuaciones diferenciales, un punto de equilibrio se logra cuando la solución de la ecuación diferencial es constante, es decir, cuando todas sus derivadas son cero[2]. Si se toma como ejemplo un sistema mecánico significaría que su velocidad, aceleración, jerk y derivadas de mayor orden son nulas, por lo que su posición no varía con el tiempo respecto a la posición inicial siempre que no ocurran perturbaciones.

El equilibrio puede ser *estable*, *indiferente* e *inestable*; la diferencia entre ellos se puede entender si suponemos un sistema cuyas condiciones iniciales se dan en una situación tal como se muestra en la Figura [11.6]. Mientras no existan perturbaciones, la posición no cambiará debido a que es un punto de equilibrio, pero en el momento en que se da la perturbación, el comportamiento del sistema define que tipo de equilibrio es. Si tiende a regresar a la posición inicial el es estable, si se detiene en cualquier otra posición se considera indiferente y si cada vez se aleja más de la posición inicial el equilibro es inestable, normalmente al alejarse de un punto inestable el sistema busca un punto diferente estable[78].

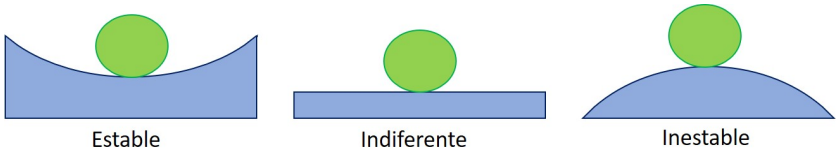


Figura 11.6: *Tipos de equilibrio*

En el caso de nuestro sistema, los valores constantes de ángulo (θ), altura (h), fuerza de sustentación del motor izquierdo (F_1) y la del motor derecho (F_2) que mantienen al sistema en equilibrio son:

$$\theta_e = 0[rad] \quad h_e = 0.15[m] \quad F_{1e} = 2.5791[N] \quad F_{2e} = 2.6715[N]$$

Estos valores se pueden calcular de las ecuaciones (10.19) y (10.20) al hacer cero todas sus derivadas y despejar dichas variables, o también experimentalmente. El ángulo resultó ser un equilibrio inestable mientras que la altura un equilibrio indiferente. Se comenta más sobre estos valores y su utilización en la Sección [12.1.1] de linealización.

Muchas veces por practicidad se utilizará en las gráficas la notación Δ (por ejemplo $\Delta Angulo$ o $\Delta Altura$), esto significa que el valor está referido al punto de equilibrio de la siguiente forma: $\Delta Altura = h - h_e$. Por ejemplo, si se muestra en la gráfica un valor donde $\Delta altura = -0.05[m]$ significa que la altura real del modelo es $h = 0.1[m]$ y está a $5[cm]$ del punto de equilibrio.

11.4. Controladores independientes tipo PID

Debido a la forma en la que se realizó el modelado anterior, cada uno de los subsistemas desacoplados son lineales y cuentan con una sola entrada y salida. El sistema traslacional tiene como entrada una diferencia de fuerza menos la necesaria para mantener en equilibrio al sistema y como salida la altura del móvil. El sistema rotacional cuenta con un par como entrada, causado por una diferencia de fuerzas entre los motores y el ángulo del móvil como salida. Tanto el modelo planteado de altura como el de ángulo pueden ser controlados independientemente por un controlador tipo PID, como se muestra en la Figura [11.7]. Los valores de referencia para ambos subsistemas se toman respecto al punto de equilibrio ya que su modelado parte de este punto (se aborda más sobre el sistema y sus puntos de equilibrio en la Sección 12.1.1).

A pesar de que los cálculos de los parámetros de control son posibles debido a que se parte de los subsistemas desacoplados, el modelo real no se comporta como dos sistemas separados. Se debe de asegurar que los

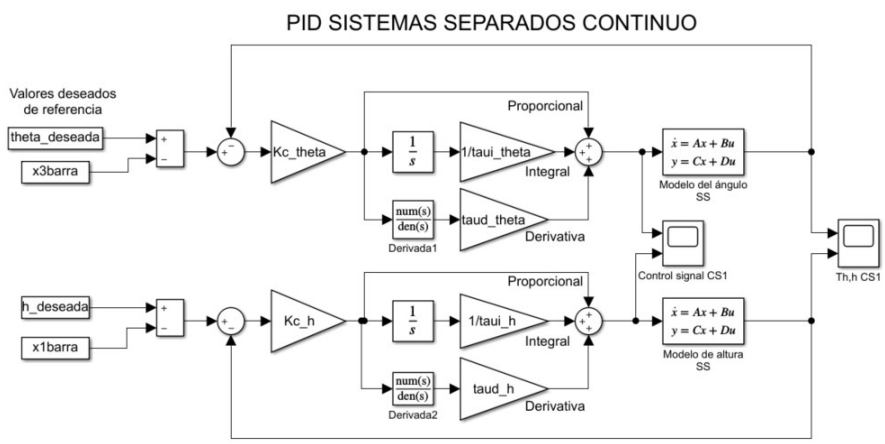


Figura 11.7: *Diagrama de bloques (Simulink): PIDs independientes para los sistemas desacoplados*

controladores pueden estabilizar y regular ambas variables en el prototipo. Hasta ahora la mejor aproximación del comportamiento de la planta es la obtenida en la Sección 10.2.1 mediante el modelado matemático por Euler-Langrange. Este modelo no lineal de cuarto orden se puede utilizar para simular y verificar si el control se realiza correctamente en el sistema completo. Para ello se utiliza el diagrama de bloques mostrado en la Figura [11.8], donde el modelo no lineal se encuentra remarcado en rojo.

Se presentan dos inconvenientes al sustituir los sistemas desacoplados por el modelo no lineal como la planta. El primero es que el modelo matemático considera la fuerza de cada motor, el ángulo y la altura como variables absolutas. Estas variables son las de entrada y salida de la planta. Debido que los modelos desacoplados consideran estos valores como diferencias del punto de equilibrio, es necesario adaptarlas como absolutas. El segundo inconveniente es que las entradas de los sistemas desacoplados no son directamente las fuerzas de los motores, sino valores provenientes de ellas. El PID referido al ángulo tiene como señal de control la diferencia de empuje entre los dos motores, mientras que la señal del controlador referido a la altura es la diferencia del empuje total de los motores respecto al necesario para mantener al sistema en equilibrio.

La forma en la que resolvió esta discrepancia fue mediante el bloque llamado “combinar efectos” (remarcado en azul); este es el encargado de adaptar las señales de control y convertirlas en las entradas de F_1 y F_2 necesarias para el modelo matemático original. Se puede ver el desglose de dicho bloque en la Figura [11.9].

Los bloques que se encuentran antes del modelo (marcados en verde) sirven solamente para visualizar la señal de control resultante de forma más intuitiva ya que presentan dicha señal en valores del ancho de pulso (*duty*) del PWM y considera además el redondeo de la señal de control a números enteros, ya que el microcontrolador no puede enviar valores flotantes como ancho de pulso en el prototipo. Las funciones de conversión entre PWM y Fuerza $[N]$ se tratan más adelante en la Sección 13 de implementación de los controladores. Así mismo, en la Sección D se presenta el desglose completo de todos los diagramas de bloques utilizados en este trabajo.

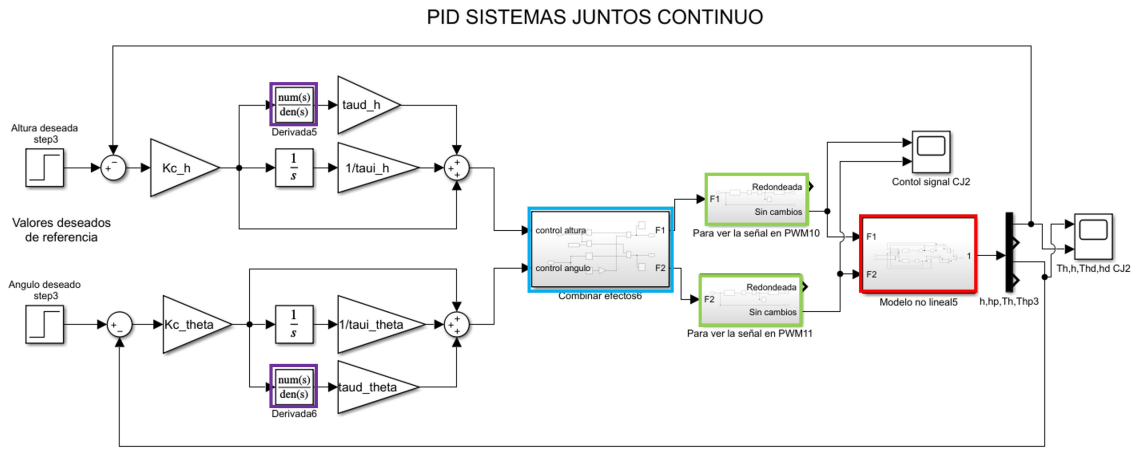


Figura 11.8: *Diagrama de bloques (Simulink): PIDs adaptados para el modelo no lineal*

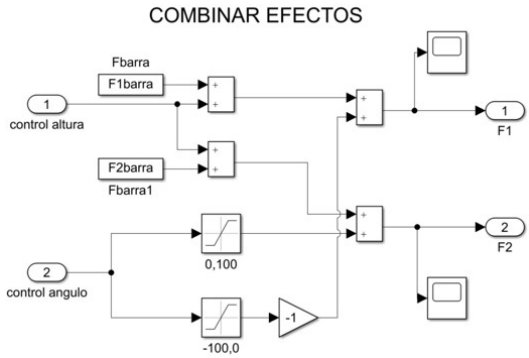


Figura 11.9: *Diagrama de bloques (Simulink): Función encargada de adaptar las señales de control*

De esta forma se puede simular el desempeño de los controladores tanto en los modelos desacoplados como en el modelo no lineal, el cual depende de la sintonización que se realice para cada uno de los PIDs.

11.4.1. Sintonización por ubicación de polos

El asignar valores a las ganancias de un controlador, en este caso a k_c, τ_i y τ_d , dependiendo del desempeño que se requiere, se le conoce como sintonización. Existen muchos procedimientos teóricos y prácticos para sintonizar un PID, una de las más populares es la sintonización por asignación de polos. Para utilizar este método es necesario conocer el modelo de la planta, es decir que utilizaremos los modelos desacoplados obtenidos en la Sección 11.2. Ambos modelos se calcularon de tal forma que pueden ser representados por una función de transferencia, como la que se muestra en (11.22), donde los valores A y B son constantes específicas de cada sistema.

$$G(s) = \frac{A}{s(s+B)} \tag{11.22}$$

Si se toma el diagrama de bloques del PID convencional de la Figura [11.1] el bloque de proceso corresponde a la función de transferencia de cada sistema. Se utiliza álgebra de bloques para obtener (11.23) y así reducir todo el diagrama a una sola función de transferencia que representa el comportamiento de lazo cerrado de un sistema junto con su controlador.

$$T(s) = \frac{Ak_c(s + \frac{1}{\tau_i}\tau_d s^2)}{s^3 + (B + Ak_c\tau_d)s^2 + Ak_c s + A\frac{k_c}{\tau_i}} \tag{11.23}$$

Se observa que el PID añadió dos ceros y un polo a la función de transferencia original del proceso, debido a que el numerador ahora es un polinomio de segundo grado y el denominador de tercero. Sin embargo, el denominador, también conocido como *polinomio característico*, es el que en su mayoría rige el desempeño. En este polinomio las constantes del control nos dan la libertad de asignar sus raíces (polos) libremente, por lo que sólo se debe definir un desempeño y asignar las constantes tal que el polinomio resultante cumpla las especificaciones.

Para definir el desempeño deseado se toman como base las características de los sistemas de segundo orden, como el *sobrepaso* ($\%SP$) y el *tiempo de asentamiento* (t_s). Estos valores se pueden usar para situar los polos de un sistema de segundo orden de tal forma que su respuesta sea *subamortiguada*, las ecuaciones (11.24) y (11.25) permiten obtener el coeficiente de amortiguamiento (ζ) y la frecuencia natural (ω_n), los cuales definen la posición de los dos polos como se muestra en (11.26). Es importante recordar que la parte real de estos polos debe ser negativa si se desea que el sistema de lazo cerrado resultante sea estable[78].

$$t_s = \frac{5}{\omega_n \zeta} \tag{11.24}$$

$$\%SP = 100e^{\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}} \tag{11.25}$$

$$P_{1,2} = -\zeta\omega_n \pm \omega_n\sqrt{1-\zeta^2} \tag{11.26}$$

Debido a que el polinomio característico del sistema es de tercer orden, es necesario definir la posición de un tercer polo. Esta raíz debe de tener poca influencia en el desempeño general del sistema de lazo cerrado, es decir que debe ser un polo *no dominante*. Se sabe que los polos más lentos son los que rigen la dinámica, entonces basta con hacerlo más rápido que los que serán elegidos por medio del desempeño y la ecuaciones (11.24) y (11.26) para que esto se cumpla. Un valor comúnmente aceptado es hacer el polo no dominante mínimo 5 veces más rápido que los dominantes.

Con las tres raíces se obtiene un *polinomio característico deseado*, el cual simplemente se debe igualar con el polinomio característico de lazo cerrado de cada uno de los subsistemas de control y resolver los sistemas de ecuaciones resultantes para obtener las ganancias de los controladores PID. El desempeño obtenido por este método, sobretodo en la simulación, será suficientemente similar al deseado pero no igual debido al tercer polo y los ceros que tiene la función de transferencia que lo distinguen de un sistema de segundo orden.

La asignación de polos para cada sistema desacoplado de la planta resultó como se muestra en la Tabla [11.3]:

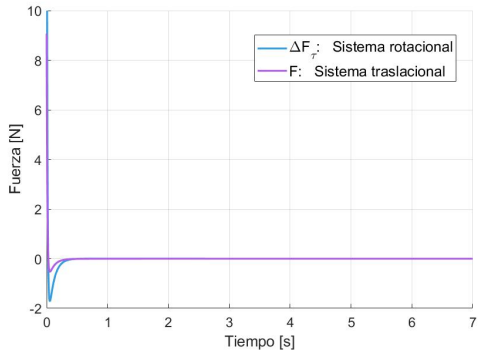
Al utilizar estas ganancias obtenidas y los diagramas de bloques de Simulink propuestos de las figuras [11.8] y [11.7], se comprobó que la propuesta de dos controladores desacoplados para un sistema complejo si es válida. La simulación obtenida del desempeño del controlador, tanto en los sistemas desacoplados como en el sistema no lineal, se encuentra en las gráficas de la Figura [11.10]. En el caso de las variables a controlar las gráficas muestran en violeta los datos, escala y unidades referentes a la altura; mientras que en azul aquello que tiene que ver con el ángulo. Es importante recordar que, debido a su construcción, los sistemas desacoplados alcanzan el valor deseado con una señal de control y variables de salida de cero ya que se plateó a partir del punto de equilibrio como se explica en la Sección [11.3], mientras que el sistema no lineal representa las variables físicas absolutas.

Se puede observar que esta propuesta logra llevar a la planta exitosamente al punto de equilibrio desde diferentes condiciones iniciales. El desempeño entre los distintos modelos varía. El sistema no lineal se aleja

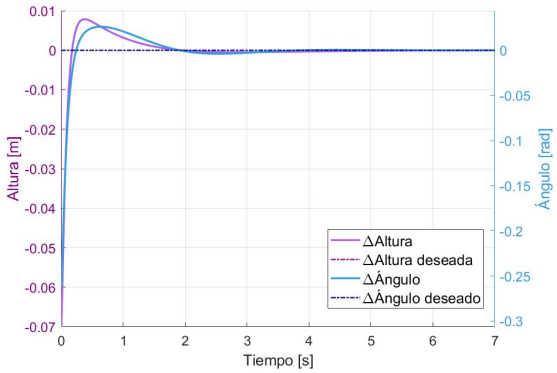
Constantes del subsistema	Desempeño deseado	Polos deseados	Polinomio característico deseado	Ganancias PID
Sistema traslacional				
$A = 3.725$ $B = 0.01863$	$t_s = 5[s]$ $\%SP = 0.5\%$	$p1,2 = -1 + 0.5929i$ $p3 = -10$	$s^3 + 12s^2 + 21.3516s + 13.5158$	$K_c = 5.7315$ $\tau_i = 1.5797$ $\tau_d = 0.5620$
Sistema rotacional				
$A = 30.56$ $B = 1.652$	$t_s = 5[s]$ $\%SP = 15\%$	$p1,2 = -1 + 1.6560i$ $p3 = -10$	$s^3 + 12s^2 + 23.7423s + 37.4227$	$K_c = 0.7769$ $\tau_i = 0.6344$ $\tau_d = 0.4358$

Tabla 11.3: Cálculo de las ganancias de los controladores PID por asignación de polos

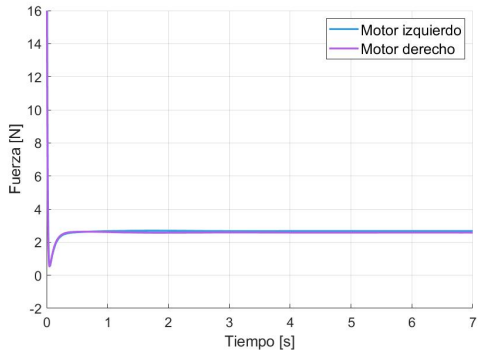
más del desempeño deseado ya que cuenta con un tiempo de asentamiento y sobrepaso mayor en ambas salidas, aun así cumple con su propósito. Ambos modelos cuentan con una señal agresiva de control al inicio. Una buena sintonización debe de tomar en cuenta los límites físicos del dispositivo y que la señal de control sea suave sin comprometer el desempeño.



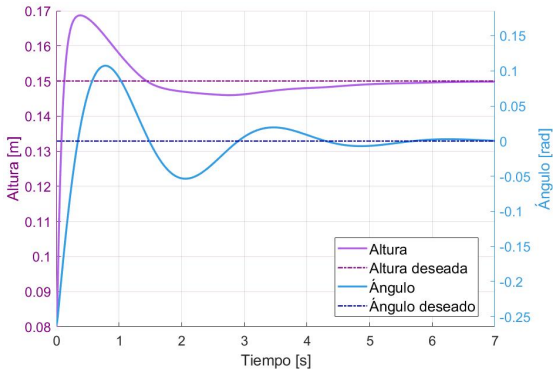
(a) Sistemas desacoplados: Señal de control



(b) Sistemas desacoplados: Variables a controlar



(c) Modelo no lineal: Señal de control

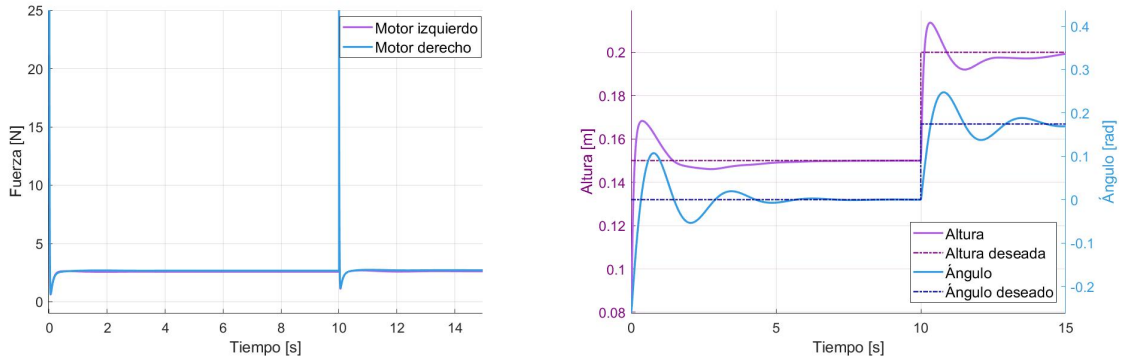


(d) Modelo no lineal: Variables a controlar

Figura 11.10: *Simulación del control PID por ubicación de polos para sistemas desacoplados y modelo no lineal.*

11.4.2. Patada derivativa

Las acciones proporcional e integral del controlador le permiten lidiar con cualquier cambio de referencia que se le pueda presentar, aun así, es necesario comprobar que esto se mantiene verdadero en la configuración del sistema no lineal de la Figura [11.8]. Al realizar un cambio en los valores deseados se obtienen los desempeños mostrados en la Figura [11.11], en ellos se presenta fuertemente el fenómeno que anteriormente se mencionó conocido como *Derivative Kick* justo en el cambio de referencia. Para contrarrestarlo se decidió derivar directamente la entrada en lugar del error para evitar estos valores pico[83].



(a) PID en modelo no lineal: Señal de control

(b) PID en modelo no lineal: Variables a controlar

Figura 11.11: *Simulación del control PID por ubicación de polos con cambio de referencia para el modelo no lineal*

Este cambio implica modificar el término derivativo de la ecuación (11.1), donde en lugar del error se deriva directamente la salida de la planta. Se puede notar que, debido a que la derivada de una constante es cero, siempre que la referencia se mantenga constante, la derivada del error será igual al negativo de la derivada de la salida.

$$\frac{de(t)}{dt} = \frac{d(\cancelto{0}{r(t)} - y(t))}{dt} = -\frac{dy(t)}{dt} \tag{11.27}$$

Esto significa que no es necesario un cambio en la sintonización del controlador, simplemente basta con multiplicar esta nueva derivada por $-K_c$ y mantener las demás constantes iguales. Con esta nueva configuración, mostrada en la Figura [11.12], se logró evitar el fenómeno de pico en la señal de control con un efecto mínimo en el desempeño de las variables a controlar, tal como se aprecia en la gráfica (a) de la simulación resultante mostrada en la Figura [11.13].

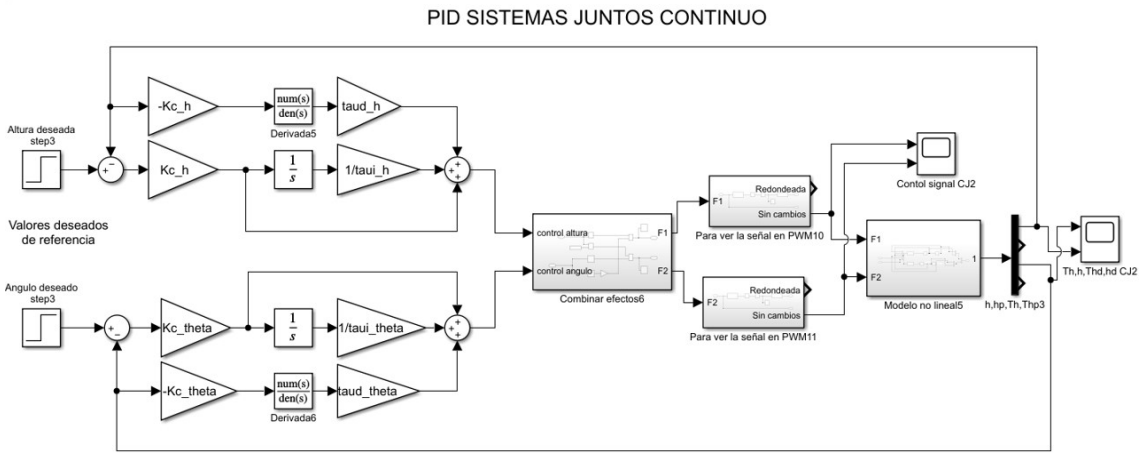
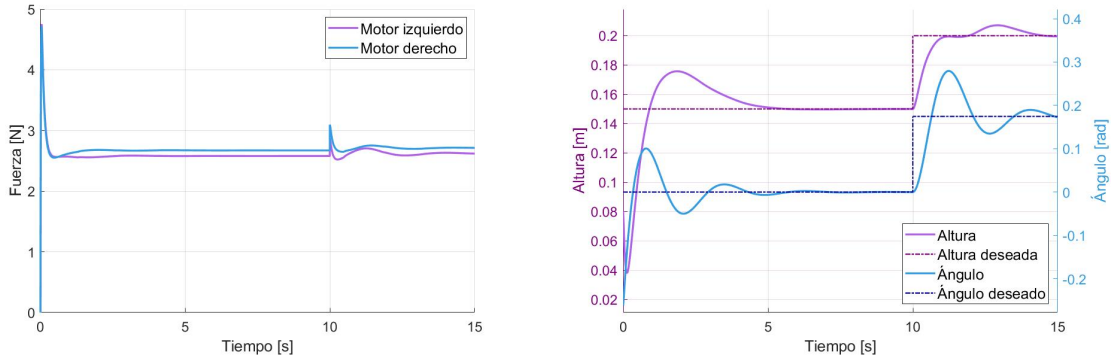


Figura 11.12: *Diagrama de bloques (Simulink): controladores PID para el modelo no lineal contrarrestando el efecto "Derivative Kick" (con derivada en la salida)*



(a) PID en modelo no lineal: Señal de control

(b) PID en modelo no lineal: Variables a controlar

Figura 11.13: *Simulación del control PID por ubicación de polos y derivada en la salida con cambio de referencia para el modelo no lineal*

11.4.3. Sintonización por Ziegler-Nichols

Para utilizar el método por asignación de polos se requiere conocer previamente el modelo de la planta, es por ello que, cuando este no aproxima de forma correcta el comportamiento real, esta forma de sintonizar puede dar resultados inesperados o incluso fallar. Existen otros tipos de métodos de sintonización basados en resultados empíricos tales como CHR[84], Cohen-Coon[85] y Ziegler-Nichols[86]. Este último es el de los más conocidos por su facilidad de uso y los buenos resultados que brinda.

Esta estrategia de sintonía fue creada en 1942 por los ingenieros John Ziegler y Nataniel Nichols con el propósito de obtener las ganancias de control sin la necesidad de conocer la planta. Ellos realizaron una serie de experimentos en plantas de distintas dinámicas buscando un desempeño con sobrepaso máximo de 25 % ante una entrada tipo escalón y un fuerte rechazo a las perturbaciones[86]; este método permite determinar los parámetros del controlador PID con base en la respuesta transitoria del sistema. Existen dos variaciones del mismo método: de lazo abierto y de lazo cerrado.

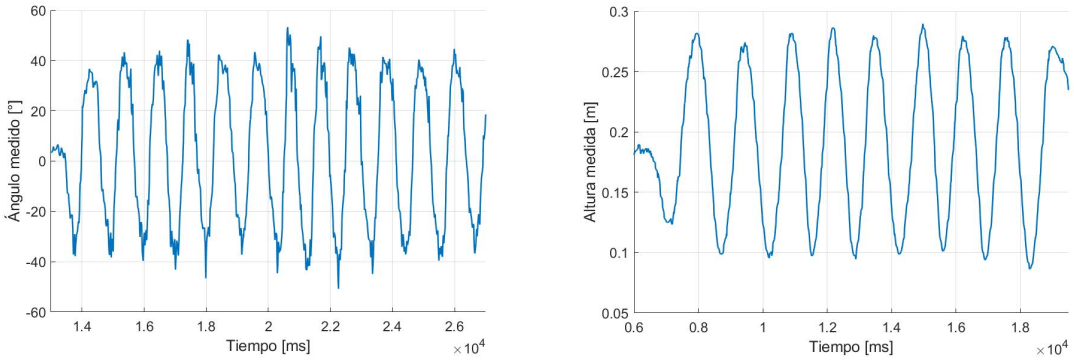
El método de lazo abierto sirve para sistemas estables de primer orden o superior cuya respuesta sea sigmoidal. Consiste en proporcionar al sistema una entrada de escalón unitario y registrar la respuesta en la

salida. Basado en algunos parámetros como el retardo, valor final y constante de tiempo obtenidos es posible calcular valores para las ganancias de control.

El segundo método debe realizarse con el sistema en lazo cerrado únicamente utilizando la ganancia proporcional. Se debe variar experimentalmente esta ganancia hasta obtener una respuesta oscilatoria con amplitud constante. Estas oscilaciones son posibles solo cuando el lazo cerrado de control es de orden superior. Básicamente, lo que se busca es llevar al sistema al punto de estabilidad crítica; esto solo se consigue si el sistema de lazo cerrado resultante es de tercer orden o mayor debido al lugar geométrico de sus raíces. Una vez que la respuesta del sistema es críticamente estable es posible obtener el periodo de oscilación llamado *periodo crítico*, este valor, junto con la *ganancia crítica* con la que se consiguió dicha estabilidad, son los parámetros base para obtener las ganancias del controlador. Es importante recalcar que llevar un sistema cerca de la inestabilidad es peligroso para el operario y el sistema en sí, por lo que que este tipo de sintonización no es muy común en sistemas industriales. Sin embargo, debido a la escala, componentes utilizados, y a la finalidad del dispositivo para su uso como banco de pruebas, este método no implica gran desventaja.

En el caso de los subsistemas sobre los que queremos implementar los controladores se tiene un elemento integral en cada uno que los vuelve naturalmente inestables, por lo tanto, no es posible aplicar el primer método, pero el sistema de lazo cerrado tiene el orden suficiente como para implementar el segundo.

Para lograrlo primero se bloqueó el movimiento de rotación para que pudiera solamente moverse verticalmente y se realizó el experimento utilizando el control de la Figura [11.8], en el que todas las ganancias son cero con excepción de la ganancia proporcional del control de altura, ya que ésta es la que se va a sintonizar. Para el sistema rotacional fue el mismo procedimiento, pero esta vez se bloqueó el movimiento vertical y varió la ganancia proporcional del control angular. Los resultados obtenidos se muestran en la Figura [11.14].



(a) *Ángulo:* $K_{c\theta} = 1.4$ (b) *Altura:* $K_{ch} = 8.4$

Figura 11.14: *Oscilaciones cerca de la estabilidad crítica conseguidas experimentalmente*

De las gráficas obtenidas se deben rescatar la ganancia crítica k_u, que es con la que se logró que oscilara el sistema (donde la de ángulo es $K_{c\theta} = 1.4$ y la de altura es $K_{ch} = 8.4$), y el periodo crítico P_u que simplemente es el periodo con el que oscila (el de ángulo es $P_{u\theta} \approx 1.16[s]$ y el de altura es $P_{uh} \approx 1.61[s]$). Con estos valores se puede utilizar la tabla del método Ziegler-Nichols de lazo cerrado representada en la Figura [11.15] para obtener las ganancias de los controladores. En ocasiones, si la respuesta obtenida es muy agresiva, se recomienda dividir a la mitad el valor de la constante proporcional obtenida.

Después de realizar los cálculos se obtienen los valores de la ecuación (11.28) para el controlador del ángulo y (11.28) para el de la altura.

$$k_c = 0.84 \quad \tau_i = 0.58 \quad \tau_d = 0.145 \tag{11.28}$$

$$k_c = 5.04 \quad \tau_i = 0.805 \quad \tau_d = 0.201 \tag{11.29}$$

Controlador	K_p	τ_i	τ_d
P	$0.5K_u$	∞	0
PI	$0.45K_u$	$\frac{1}{1.2}P_u$	0
PID	$0.6K_u$	$0.5P_u$	$0.125P_u$

Figura 11.15: *Tabla de sintonización de Ziegler-Nichols de lazo cerrado*

Es importante recordar el método no es universal y que estas tablas solamente son una recomendación inicial de valores; A veces es necesario ajustar paulatina y empíricamente los valores hasta obtener un desempeño satisfactorio. En este caso se ajustaron los valores por prueba y error hasta llegar a (11.30) para las ganancias del ángulo y (11.31) para las de altura.

$$k_c = 0.84 \quad \tau_i = 1.0 \quad \tau_d = 0.145 \tag{11.30}$$

$$k_c = 5.73 \quad \tau_i = 1.57 \quad \tau_d = 0.56 \tag{11.31}$$

Comparado con los valores de la Tabla [11.3] obtenidos por asignación de polos, se pueden ver que las ganancias de ambos sistemas son muy similares a las obtenidas por este método empírico, es por ello que el ajuste para obtener los valores finales se hizo basado en estos dos análisis y el desempeño del sistema real.

Está sintonización se realizó inspirada en un problema experimental, ya que los valores obtenidos por ubicación de polos resultaban en un desempeño deficiente en el prototipo. Las ganancias de (11.30) y (11.31) son las que demuestran un mejor resultado (el cual se puede ver más adelante en la Sección 14 de resultados). Por esto y la similitud respecto a aquellas obtenidas por ubicación de polos se decidió no centrarse en los resultados simulados de este método.

11.5. Control PID discreto

El análisis y cálculos que se hacen utilizando funciones de transferencia, como los realizados en las secciones pasadas, resultan en controladores de tiempo continuo ideales para sistemas análogos. Antes del avance digital de la tecnología estos diseños eran el estándar, tal como comenta Piedrafita (1999)[87]; por ejemplo, el controlador mecánico de temperatura *Foxboro Stabilog* desarrollado en los años 30 o los controladores electrónicos analógicos creados utilizando amplificadores operacionales. Sin embargo, desde la invención del transistor, alrededor de 1945, que posteriormente dio paso a los primeros microcontroladores alrededor de 1970, la forma en la que construyen e implementan los controladores ha cambiado, en muchos casos, a ser digital debido a la facilidad de estas tecnologías.

Cuando se trabaja y procesa la información de forma discreta, el tiempo de procesamiento y la conversión de la información mediante convertidores analógico - digital (ADC) y digital - analógico (DAC), se puede presentar que los resultados no sean consistentes con los esperados en un controlador continuo. Para mejorar esto, existen herramientas para aproximar ecuaciones y elementos de tiempo continuo a tiempo discreto. En este proyecto, buscando una mayor certidumbre en los resultados, se decidió discretizar el controlador.

La discretización de la ley de control debe partir de la continua expresada en (11.1) y se debe obtener una ecuación en diferencias que aproxime la expresión continua[88]. El primer elemento, que corresponde a la acción proporcional, se puede aproximar de la siguiente manera:

$$k_c e(t) \approx k_c e[k] \tag{11.32}$$

Para el elemento integral se decidió utilizar la *aproximación trapezoidal*. Una forma de ver a la integral es como el área bajo la curva de una función, como se muestra en color verde en la Figura [11.3]. La regla del trapecio aproxima el valor mediante dos puntos de la curva creando un trapecio entre dichos puntos y el eje horizontal como se muestra en la Figura [11.16], en este caso los puntos se pueden tomar como las muestras tomadas separadas por el periodo de muestreo. La suma total de las áreas de cada trapecio generado, cuando el tiempo entre los puntos seleccionados es suficientemente pequeño, se aproxima al valor original del área bajo la curva. Entonces, se puede aproximar la acción integral como se muestra en (11.33).

$$\frac{k_c}{\tau_i}\int_0^t e(t)dt \approx \frac{k_c}{\tau_i}\sum_{k=0}^{k}\left(\frac{e[k]+e[k-1]}{2}T_s\right) \tag{11.33}$$

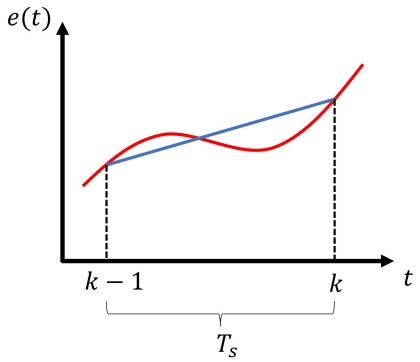


Figura 11.16: *Regla del trapecio construida con dos mediciones*

En lo que respecta al último elemento, la derivada continua representa la tasa de cambio instantánea de una variable, es decir, el cambio que tiene una variable cuando la diferencia de tiempo en la que ocurrió tiende a cero. Para aproximar este valor en el dominio discreto basta con obtener la diferencia entre dos mediciones y dividirla entre el tiempo que esto tomó, como se muestra en (11.34).

$$\tau_d\frac{de(t)}{dt} \approx \frac{\tau_d}{T_s}\left(e[k]-e[k-1]\right) \tag{11.34}$$

Al sustituir las ecuaciones (11.32), (11.33) y (11.34) en la ley de control continua (11.1) se obtiene la ecuación (11.35). Una de las ventajas de utilizar ecuaciones en diferencias es que se puede recorrer la muestra hacia delante y hacia atrás en el tiempo libremente sin afectar el resultado, de esta forma podemos representar la señal de control pasada, tal como se muestra en (11.36).

$$u[k] = k_c\left(e[k]+\frac{T_s}{2\tau_i}\sum_{k=0}^{k}\left(e[k]+e[k-1]\right)+\frac{\tau_d}{T_s}\left(e[k]-e[k-1]\right)\right) \tag{11.35}$$

$$u[k-1] = k_c\left(e[k-1]+\frac{T_s}{2\tau_i}\sum_{k=0}^{k}\left(e[k-1]+e[k-2]\right)+\frac{\tau_d}{T_s}\left(e[k-1]-e[k-2]\right)\right) \tag{11.36}$$

Con el objetivo de eliminar la sumatoria de (11.35) y obtener una ley de control que no requiera de todos los errores pasados se restan (11.35) y (11.36). De esta forma se obtiene la ley de control discreta del controlador PID en función del error actual, dos errores pasados y la señal de control pasada como se muestra en la ecuación (11.37), donde q_0, q_1 y q_2 son constantes que dependen de las ganancias del controlador y el periodo de muestreo. Esta representación es más fácil de implementar en un microcontrolador.

$$u[k]-u[k-1]=k_c\left(e[k]-e[k-1]+\frac{T_s}{2\tau_i}(e[k]+e[k-1])+\frac{\tau_d}{T_s}(e[k]-2e[k-1]+e[k-2])\right)$$

$$\begin{aligned}
u[k] &= u[k-1]+q_0e[k]+q_1e[k-1]+q_2e[k-2] \\
q_0 &= k_c\left(1+\frac{T_s}{2\tau_i}+\frac{\tau_d}{T_s}\right) \\
q_1 &= -k_c\left(1-\frac{T_s}{2\tau_i}+\frac{2\tau_d}{T_s}\right) \\
q_2 &= k_c\left(\frac{\tau_d}{T_s}\right)
\end{aligned} \tag{11.37}$$

Es posible simular la dinámica resultante en Simulink al utilizar los bloques de *Delay* o directamente con la transformada Z como se muestra en (11.38). El diagrama de bloques de la Figura [11.17] sirve para simular la respuesta de esta nueva propuesta discreta sobre el sistema utilizando dichos bloques de retraso de señal.

$$G(z)=\frac{u(z)}{e(z)}=\frac{q_0+q_1z^{-1}+q_2z^{-2}}{1-z^{-1}} \tag{11.38}$$

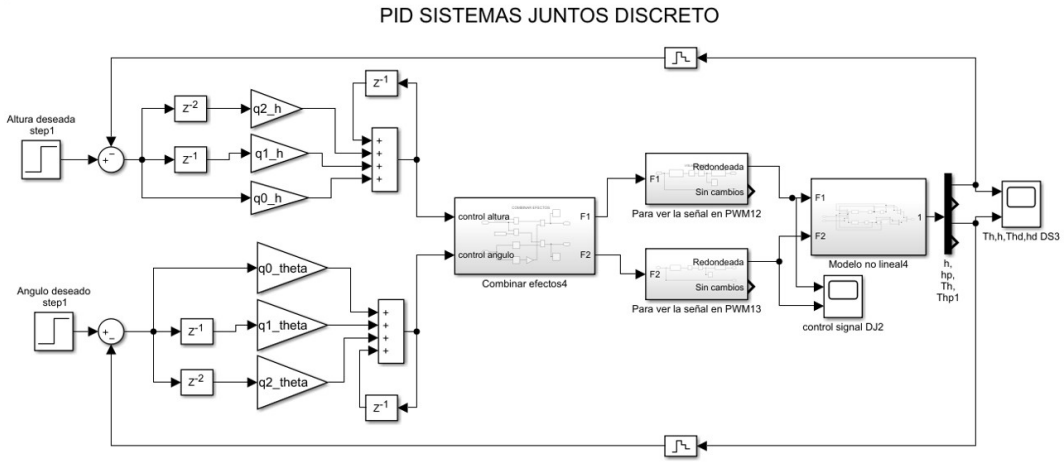


Figura 11.17: *Diagrama de bloques (Simulink): Sistema no lineal con PIDs discretos*

En caso de querer discretizar e implementar el sistema donde se deriva directamente la salida de la planta, el procedimiento es el mismo, sólo que se cambia la aproximación discreta de la derivada mostrada en (11.34) por el descrito en (11.39).

$$\tau_d\frac{de(t)}{dt}\approx-\frac{\tau_d}{T_s}\left(y[k]-y[k-1]\right) \tag{11.39}$$

De esta forma, se obtiene una nueva ley de control compuesta por (11.32), (11.33) y (11.39); que al restarse con la pasada (recorrida hacia atrás en el tiempo) da como resultado la ecuación en diferencias (11.40), que describe una nueva la ley de control PID discreta que evita el efecto de patada derivativa. La forma de implementar esta opción como diagrama de bloques se muestra en la Figura [11.18].

$$u[k] = u[k-1] + h_0 e[k] + h_1 e[k-1] + h_2 (y[k] - 2y[k-1] + y[k-2]) \tag{11.40}$$
$$h_0 = k_c \left(1 + \frac{T_s}{2\tau_i}\right)$$
$$h_1 = -k_c \left(1 - \frac{T_s}{2\tau_i}\right)$$
$$h_2 = -k_c \left(\frac{\tau_d}{T_s}\right)$$

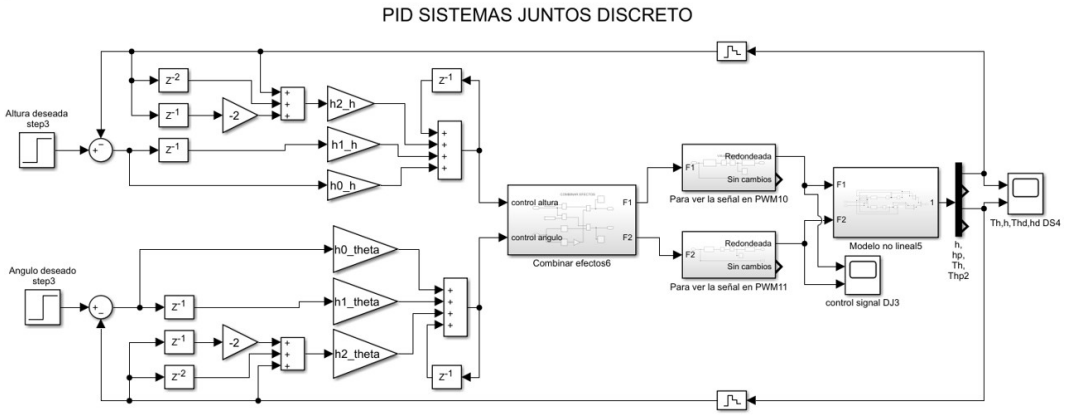
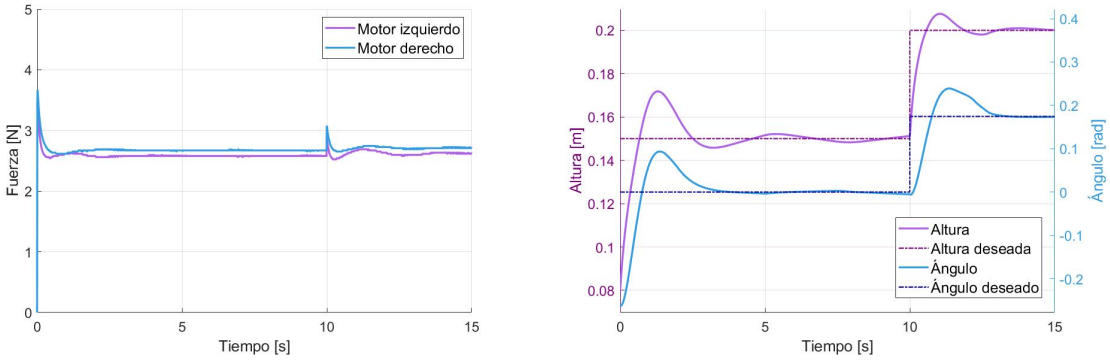


Figura 11.18: *Diagrama de bloques (Simulink): Sistema no lineal con PIDs discretos y derivada en la salida*

Según García Jaimes (2009)[89], la selección del periodo de muestreo (T_s) es muy importante en la implementación de cualquier controlador discreto y depende de varios factores. El primero de ellos es la velocidad de cálculo del procesador, ya que este definirá el mínimo posible. Además, (T_s) debe ser lo suficientemente pequeño comparado con la dinámica del proceso como para representar fielmente la señal sin perder información e identificar rápidamente las perturbaciones, pero tampoco demasiado bajo porque se presentan problemas de precisión y muestreo. Un periodo de muestreo incorrecto puede provocar perdida de información e incluso provocar la inestabilidad.

Existen distintas formas de calcular el periodo de muestreo. El mismo Garcia Jaimes propone utilizar una técnica conocida como *Teorema del Muestreo de Shannon*, la cuál se basa en el análisis del propio sistema. Este teorema dice que si se desea reconstruir una señal continua en un tiempo discreto, la frecuencia de muestreo debe ser suficientemente alta comparada con la componente de mayor frecuencia del sistema que se incluye en la señal de tiempo continuo para que las características de amplitud de la señal de tiempo continuo se puedan preservar en la envolvente de la señal muestreada. Una forma de conseguir esto es que la frecuencia de muestreo sea entre 8 y 12 veces más alta que el ancho de banda del sistema de lazo cerrado. En este proyecto se utilizó el diagrama de Bode para conocer el ancho de banda de la planta y aplicar dicha regla; el resultado mostró que T_s se puede definir entre $66[ms]$ y $43[ms]$. Otra regla práctica, de acuerdo con Carlos A. Smith (1991)[90], consiste en definir al tiempo de muestreo de un controlador entre un décimo y un vigésimo de la constante de tiempo efectiva del elemento más rápido del proceso; esta nueva propuesta colocaría nuestro valor de T_s entre $60[ms]$ y $30[ms]$. Aunque $30[ms]$ arrojó resultados estables experimentalmente, se observó que las mejores dinámicas resultaban de bajar un poco más este periodo, por lo que el periodo de muestreo se redujo hasta donde nos permitió el Hardware y se obtuvo al medir el tiempo que demora el prototipo en ejecutar un ciclo completo de control con el microcontrolador a máxima capacidad. El periodo de muestreo utilizado finalmente es de $T_s = 16[ms]$, que es el tiempo mínimo que se puede obtener con esta configuración.

Los resultados mostrados en de la Figura [11.19] corresponden a la simulación del PID discreto con derivada en la salida sintonizado por ubicación de polos. Se pueden comparar las diferencias de desempeño entre los controladores continuos y los discretos aplicados al sistema no lineal con las respuestas obtenidas en la Figura [11.10]. Ambos sistemas son estables; sin embargo, los resultados de la simulación muestran que el sistema discreto tiene una respuesta mas suave y lenta, este efecto es particularmente evidente en el desempeño del ángulo. La señal de control es ahora escalonada.



(a) PID discreto: Señal de control

(b) PID discreto: Variables a controlar

Figura 11.19: *Simulación del control PID discreto por ubicación de polos con derivada en la salida para el modelo no lineal*

12. Control moderno (variables de estado)

De acuerdo con el texto de Piedrafita (1999)[87], con el comienzo de la exploración espacial, alrededor de 1955, surgieron nuevos retos en el área aeroespacial frente a los que los métodos de control clásico no eran suficientes, así que se impulsó la investigación y el desarrollo de nuevas técnicas. Se creó una nueva forma de modelar, analizar y diseñar una gran variedad de sistemas en el dominio del tiempo utilizando *variables de estado*, conocido a partir de entonces como teoría de control moderna.

Esta representación, aunque menos intuitiva, permitió trabajar sistemas de mucho mayor complejidad además de brindar una nueva perspectiva de diseño a aquellos comúnmente representados mediante técnicas de control clásico. También, permitió representar sistemas de forma digital en una computadora y simular su respuesta ante cambios en sus parámetros, lo cual es una herramienta de diseño de controladores muy importante hoy en día. Más adelante, dentro de esta misma rama surgieron nuevas técnicas como el control óptimo y control adaptable que son muy utilizadas a la fecha.

12.1. Espacio de estados

En el área de control automático es común representar un sistema mediante una forma sencilla que permita describir la relación entre la entrada y la salida. Aquella adoptada por el control moderno permite representar sistemas complejos en el dominio del tiempo, de múltiples entradas y salidas e incluso considera características como saturación, zona muerta y condiciones iniciales diferentes de cero. Por ejemplo, es posible describir el comportamiento de un misil en diferentes alturas, velocidades y con cantidad de combustible variable. Esta representación consiste en variables del sistema llamadas *estados*, de ahí el nombre de espacio de estados, los cuales se pueden definir de la siguiente manera:

"Se define estado de un sistema como la mínima cantidad de información necesaria en un instante para que, conociendo la entrada a partir de ese instante, se pueda determinar cualquier variable del sistema en cualquier instante posterior. " (S. Dominguez, et. al. (2006), pág 4)[91].

Un sistema puede ser descrito por un número suficiente de ecuaciones diferenciales de primer orden simultaneas, escritas en función de los estados y las entradas y, si se conocen las condiciones iniciales de los estados, así como las entradas, se pueden resolver en cualquier instante. Normalmente el número de ecuaciones es igual al orden del sistema[77]. Esta representación se basa en que la combinación algebraica de los estados produce la salida del sistema. Para escribirlo se utilizan dos ecuaciones: la de estados y la de salidas, donde x es el vector de estados, u el vector de entradas y y el vector de salidas.

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{12.1}$$

Debido a que la elección de estados de un sistema no es única, pueden existir varias representaciones de un mismo sistema con diferentes valores para las matrices A, B, C y D y todas ser correctas. Si se toman en cuenta las cualidades de nuestro sistema, principalmente sus múltiples entradas y salidas, lo mejor es representarlo en variables de estado. Esto puede ser de forma directa y resultar en una representación no lineal más apegada al modelo original pero más difícil de trabajar o puede ser mediante una linealización, la cual puede aproximar el comportamiento en un punto de equilibrio. Muchas veces, también se recurre a la aproximación del sistema y sus matrices mediante datos experimentales en lugar de obtenerse de un modelo matemático. El diagrama de bloques para representar un sistema en variables de estado se muestra en la Figura [12.1].

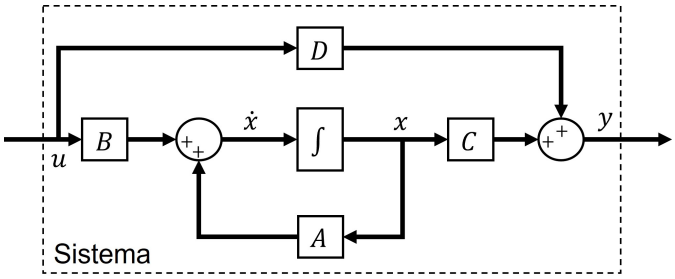


Figura 12.1: *Diagrama de bloques: Sistema en variables de estado*

12.1.1. Linealización

Los métodos de control que se utilizan en la presente tesis son herramientas empleadas en sistemas cuyos modelos son lineales, es decir, son técnicas de control lineal. Las ecuaciones (10.19) y (10.20), que describen el comportamiento del sistema de estudio, son ecuaciones diferenciales no lineales; por lo que se decidió linealizar el modelo.

La *linealización* de un modelo es una forma de simplificarlo alrededor de un punto de interés. El resultado debe aproximar suficientemente los estados a aquellos del modelo no lineal, esto se cumple siempre que los valores no se alejen demasiado del punto sobre el cual se linealizó, mientras más se alejen peor será la aproximación. Normalmente, el valor de interés alrededor del que se linealiza es un *punto de equilibrio*, el cuál se mencionó en la Sección [11.3].

Desarrollo mediante serie de Taylor

Con la linealización se busca conseguir un modelo en variables de estado que aproxime lo suficiente los valores del modelo matemático original alrededor de un punto y la *serie de Taylor* es una excelente herramienta para ello[2][78]. Al utilizar los dos primeros términos de esta serie sobre un modelo matemático $f(x,u)$ se obtiene la expresión mostrada en (12.2). Los vectores x_e y u_e son los valores de los estados y las entradas en el punto de equilibrio sobre el que se linealiza.

$$\dot{x} = f(x,u) = f(x_e,u_e) + \frac{\partial f}{\partial x}\Big|_{(x_e,u_e)}(x-x_e) + \frac{\partial f}{\partial u}\Big|_{(x_e,u_e)}(u-u_e) \tag{12.2}$$

Lo primero es expresar el modelo matemático original, obtenido en la Sección 10.2.1, en función de cuatro estados x (debido a que el sistema es de cuarto orden) y sus entradas u para obtener $f(x,u)$. Para los estados se decidió por la altura, el ángulo y sus derivadas, mientras que las entradas son las fuerzas que produce cada motor, tal como se muestra en (12.3).

$$x = \begin{bmatrix} x_1 = h \\ x_2 = \dot{h} \\ x_3 = \theta \\ x_4 = \dot{\theta} \end{bmatrix} \qquad u = \begin{bmatrix} u_1 = F_1 \\ u_2 = F_2 \end{bmatrix} \tag{12.3}$$

Como en el vector de estados derivados las derivadas también se deben expresar en función de los estados y las entradas, se escribe como se muestra en (12.4), donde $\ddot{h}$ y $\ddot{\theta}$ ya están definidas en el modelado con las ecuaciones (10.19) y (10.20) y cuyas variables pueden ser fácilmente sustituidas por las definiciones de (12.3).

$$\dot{x} = f(x,u) = \begin{bmatrix} \dot{x}_1 = x_2 \\ \dot{x}_2 = \ddot{h} \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = \ddot{\theta} \end{bmatrix} \tag{12.4}$$

Para obtener $f(x_e, u_e)$ primero se decide el punto de equilibrio sobre el que se quiere aproximar el resultado. Se recuerda que las derivadas, que son las velocidades y aceleraciones tanto angulares como lineales, deben ser nulas, por lo que:

$$f(x_e, u_e) = \begin{bmatrix} \dot{x}_{1e} = 0 \\ \dot{x}_{2e} = 0 \\ \dot{x}_{3e} = 0 \\ \dot{x}_{4e} = 0 \end{bmatrix} \tag{12.5}$$

Los valores de x_e y u_e se obtienen al despejarlos de la ecuación anterior. Estos deben asegurar que las componentes verticales de las fuerzas son suficientes para mantener la altura del modelo y los pares generados por las mismas sean iguales en ambos lados para mantener constante el ángulo. Para nuestro sistema, el punto de equilibrio seleccionado se encuentra en las condiciones expresadas en (12.6). Este es un punto de equilibrio inestable donde la altura le es indiferente.

$$x_e = \begin{bmatrix} x_{1e} = 0.15[m] \\ x_{2e} = 0[m/s] \\ x_{3e} = 0[rad] \\ x_{4e} = 0[rad/s] \end{bmatrix} \quad u_e = \begin{bmatrix} u_{1e} = 2.5791[N] \\ u_{2e} = 2.6715[N] \end{bmatrix} \tag{12.6}$$

Las expresiones $\frac{\partial f}{\partial x}$ y $\frac{\partial f}{\partial u}$ en la aproximación de Taylor son matrices jacobianas de los estados y las entradas, se construyen de la siguiente forma:

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial \dot{x}_1}{\partial x_1} & \frac{\partial \dot{x}_1}{\partial x_2} & \frac{\partial \dot{x}_1}{\partial x_3} & \frac{\partial \dot{x}_1}{\partial x_4} \\ \frac{\partial \dot{x}_2}{\partial x_1} & \frac{\partial \dot{x}_2}{\partial x_2} & \frac{\partial \dot{x}_2}{\partial x_3} & \frac{\partial \dot{x}_2}{\partial x_4} \\ \frac{\partial \dot{x}_3}{\partial x_1} & \frac{\partial \dot{x}_3}{\partial x_2} & \frac{\partial \dot{x}_3}{\partial x_3} & \frac{\partial \dot{x}_3}{\partial x_4} \\ \frac{\partial \dot{x}_4}{\partial x_1} & \frac{\partial \dot{x}_4}{\partial x_2} & \frac{\partial \dot{x}_4}{\partial x_3} & \frac{\partial \dot{x}_4}{\partial x_4} \end{bmatrix} \tag{12.7}$$

$$\frac{\partial f}{\partial u} = \begin{bmatrix} \frac{\partial \dot{x}_1}{\partial u_1} & \frac{\partial \dot{x}_1}{\partial u_2} \\ \frac{\partial \dot{x}_2}{\partial u_1} & \frac{\partial \dot{x}_2}{\partial u_2} \\ \frac{\partial \dot{x}_3}{\partial u_1} & \frac{\partial \dot{x}_3}{\partial u_2} \\ \frac{\partial \dot{x}_4}{\partial u_1} & \frac{\partial \dot{x}_4}{\partial u_2} \end{bmatrix} \tag{12.8}$$

Si se define un nuevo vector de estados y un nuevo vector de salidas que representen la diferencia entre el estado original y en equilibrio, tal que $x = (x - x_e)$ y $u = (u - u_e)$, entonces, se pueden comparar directamente la serie de Taylor (12.2) con la representación de un sistema en variables de estado (12.1); las matrices de este sistema en variables de estado se pueden definir ahora tal que:

$$A = \frac{\partial f}{\partial x}\Big|_{(x_e, u_e)} \quad B = \frac{\partial f}{\partial u}\Big|_{(x_e, u_e)} \tag{12.9}$$

Después de realizar todas las derivadas parciales necesarias para construir estas matrices, sustituir los estados en equilibrio y los valores constantes por aquellos señalados en la Tabla [10.1], se obtiene el modelo en variables de estado linealizado alrededor del punto de equilibrio expresado en (12.10). Las matrices C y D dependen únicamente de los estados de interés que consideramos como salidas, en este caso la altura ($x_1 = h$) y el ángulo ($x_3 = \theta$).

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 & 0 \\ 1.964 & 1.764 \\ 0 & 0 \\ -31.13 & 30.04 \end{bmatrix} u \tag{12.10}$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} u$$

Este nuevo sistema linealizado se puede representar en Simulink mediante un diagrama de bloques si se toman en cuenta los nuevos vectores de estados como se muestra en la Figura [12.2]. Es importante recalcar que el vector de estados ahora está referido respecto al punto de equilibrio sobre el que se linealizó; esté será un factor a considerar cuando se quiera implementar en el modelo original. Por ahora la notación Δ, propuesta en la sección 11.3, servirá para referirnos a estos nuevos estados.

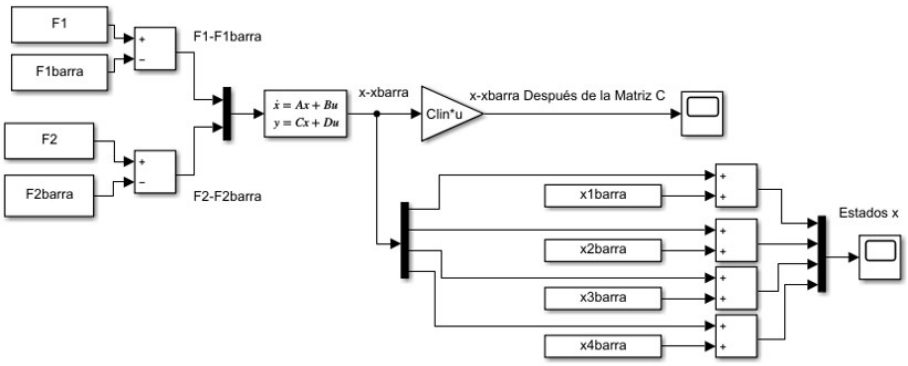


Figura 12.2: *Diagrama de bloques del modelo lineal*

Al comparar los resultados del diagrama de bloques del modelo no lineal de la Figura [10.5] con este nuevo modelo lineal, los resultados fueron iguales cuando se inicia en el punto de equilibrio. Sin embargo, con condiciones iniciales fuera de este, el modelo no lineal se desestabiliza mientras que el lineal no lo hace. Esta discrepancia quiere decir que al linealizarlo es posible que el modelo se haya sobre simplificado, es por ello que la matriz A de la ecuación (12.10) tiene tan pocos términos. Esto quiere decir que la aproximación no es muy buena respecto al modelo no lineal, aun así, los resultados demostraron que la linealización obtenida es suficiente para desarrollar un controlador que funcione con ambas representaciones.

12.1.2. Identificación de sistema

Otra forma de obtener un modelo de espacio de estados lineal es mediante algoritmos de *estimación* o *identificación*. Este método consiste en tratar a la planta como una caja negra y obtener un modelo que se aproxime al interno a partir únicamente de las entradas y las salidas.

Existen muchos métodos y algoritmos de identificación de sistemas de diferentes niveles de complejidad; en este trabajo no profundizaremos en ellos o su funcionamiento, solamente en el procedimiento de uno de ellos. El software de Matlab cuenta con herramienta útiles para esto tales como su *System Identification Toolbox*[92]. Parte de este conjunto de herramientas permite realizar la identificación de sistemas lineales tales como funciones de transferencia, variables de estados, modelos polinomiales y procesos. El código de MATLAB utilizado para obtener un modelo en variables de estado diferente del obtenido mediante la linealización previa puede solicitarse a los autores tal como se advierte en la parte V.

Para realizar esta identificación se necesitan poder enviar entradas arbitrarias al sistema y conocer sus salidas resultantes. Gracias a la interfaz creada es posible conocer, guardar y analizar esta información. Sin embargo, existen algunas consideraciones que se deben procurar para que la identificación sea lo más exitosa posible.

La primera es sobre las entradas enviadas al sistema; estas deben ser señales que oscilen alrededor de una referencia (en este caso el punto de equilibrio) con amplitudes y frecuencia lo más variadas posibles, con el objetivo de resaltar características importantes como el ancho de banda, es decir, que las entradas deben contener frecuencias donde el sistema no sea capaz de responder y otras que lo afecten evidentemente. Para

este propósito es común utilizar señales que aumenten o disminuyan la amplitud y frecuencia con el tiempo conocidas como señal *Chirp* o también *sweep signal*.

Una vez que la señal de entrada es rica en frecuencia y amplitud viene la segunda consideración: los límites físicos del sistema. Es importante comprender que este proceso es de lazo abierto, lo que significa que la señal de entrada puede ocasionar que los actuadores se saturen o el modelo alcance sus límites. En el caso de nuestro prototipo esto puede significar que los motores giren a máxima potencia, que estén completamente apagados, que el móvil llegue a su inclinación máxima o a los extremos máximos y mínimos de altura. Si esto sucede la relación entre las entradas y salidas ya no va a estar correctamente representada en los datos experimentales y, por ende, la identificación será errónea. Se realizaron varios experimentos modificando las señales de entrada hasta que se logró obtener datos experimentales donde el dispositivo osciló en su rango de trabajo.

Con estas consideraciones en mente, se hicieron varias pruebas con distintas señales de entrada y las mediciones de cada uno de los estados propuestos: el ángulo, la altura y sus respectivas velocidades. Cada experimento se guardó en un conjunto de datos específico de Matlab especial para la identificación de sistemas, conocido como *iddata*. En la Figura [12.3] se muestra un ejemplo de las mediciones recolectadas.

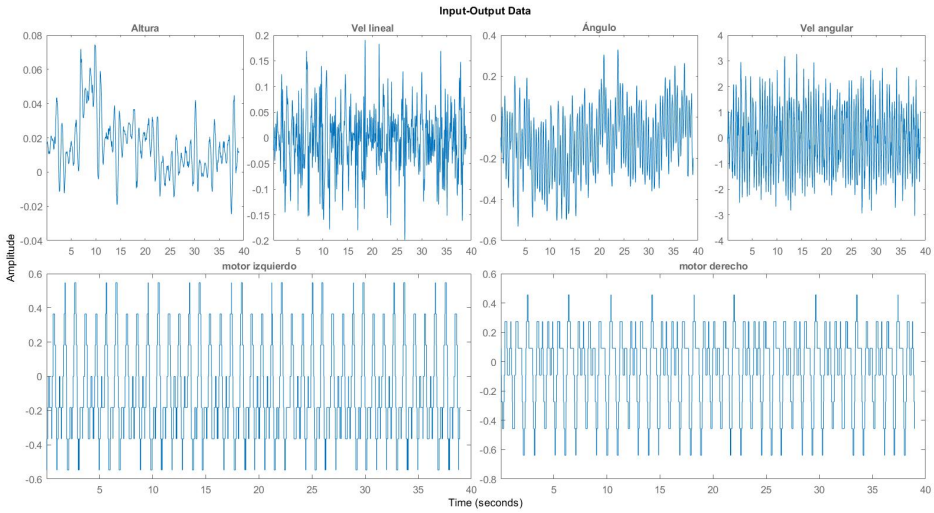


Figura 12.3: *Conjunto iddata utilizado para la identificación del sistema*

Cada una de estas pruebas puede ser utilizada para obtener una estimación. Matlab permite modificar opciones de estimación como definir las condiciones iniciales, especificar el orden del sistema e incluso definir un modelo inicial como base. Se definió la estimación como una de cuarto orden y se aprovechó esta última característica para que en cada iteración el resultado pasado fungiera como base para el siguiente resultado. Después de varias iteraciones con diferentes conjuntos de datos se logró un modelo que aproxima la respuesta de los datos experimentales con las entradas proporcionadas. En la Figura [12.4] se muestran en gris los datos reales de cada estado de uno de los experimentos y en azul la estimación del modelo obtenido basado en las mismas entradas. El software proporciona la raíz del error cuadrático medio como medida de comparación o ajuste entre los datos reales y los estimados. Normalmente se busca un porcentaje de relación alto, pero bajo estas condiciones se consideró suficiente este resultado. La representación en variables de estado del modelo obtenido se muestra en la ecuación (12.11).

$$\dot{x} = \begin{bmatrix} -0.7842 & 2.159 & 0.3371 & 0.1932 \\ 10.75 & -5.706 & 0.3085 & 0.1408 \\ -16.42 & -0.02705 & -4.372 & -0.0776 \\ -1242 & 152.3 & -138 & -9.974 \end{bmatrix} x + \begin{bmatrix} -0.6404 & 0.5671 \\ 0.5374 & 1.111 \\ 4.609 & -5.282 \\ 8.783 & -63.82 \end{bmatrix} u \tag{12.11}$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} u$$

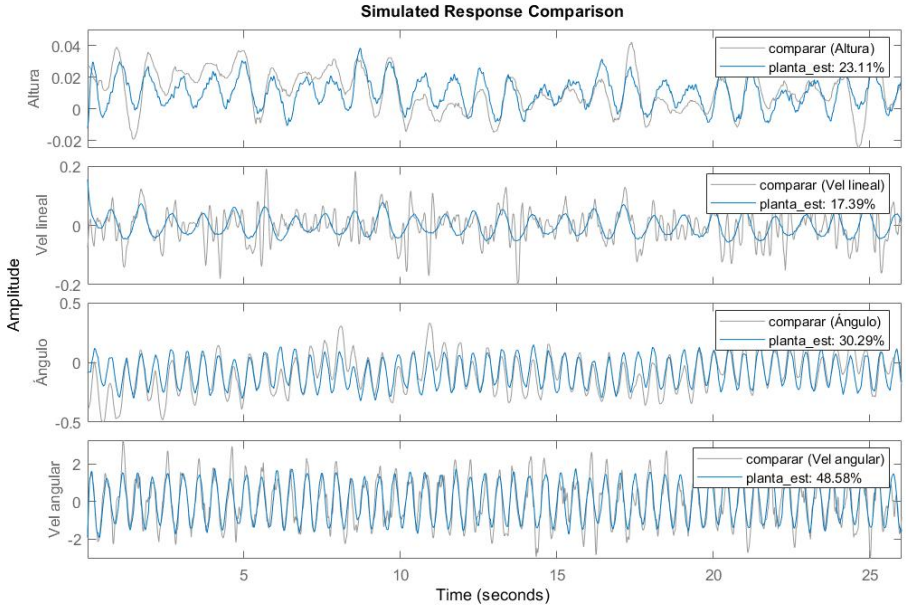


Figura 12.4: *Comparación entre datos experimentales y estimados del modelo obtenido*

Estimar una planta muchas veces proporciona mejores resultados en aplicaciones con plantas complejas o con muchas idealizaciones. Comparado con aquel obtenido en la linealización, este es meramente experimental, representa bien la frecuencia de la respuesta y considera un mayor número de elementos (algunos de los cuales se perdieron en la linealización del modelo matemático completo), lo cual lo vuelve más complejo.

A pesar de que la estimación es una buena herramienta, en este trabajo se decidió continuar con el modelo linealizado, ya que en general los controladores basados en el, arrojaron ganancias más bajas y dinámicas más lentas que permiten trabajar de forma más directa y con menos riesgo al manipular el dispositivo; por lo que, a partir de aquí, se calculan todos los controladores modernos con base en el modelo (12.10) de la Sección 12.1.1.

12.2. Controlabilidad y Observabilidad

En un sistema de control donde se realimentan los estados y se modifica la ubicación de los polos, como el que se planea proponer, se dice de forma implícita que, mediante la señal de control, se puede modificar cada estado. Si alguno de los estados de la planta no se puede influenciar mediante la entrada para alcanzar un valor deseado arbitrario, entonces no se pueden ubicar a discreción todos los polos. A esta característica se le conoce como *Controlabilidad* y se debe verificar antes de comenzar con el diseño del controlador.

"Si se puede encontrar una entrada para un sistema que lleve cada variable de estado de un valor inicial deseado a un valor final deseado, se dice que el sistema es controlable; de otra manera el sistema es no controlable"(N. Nise, pág 672)[2]

Si, en un sistema que no es completamente controlable, uno de estos polos que no podemos ubicar arbitrariamente provoca la inestabilidad en el sistema, no será posible implementar un controlador que lo estabilice.

Por ello es importante que antes de ello se verifique si el sistema es controlable. Esto se puede lograr por mera inspección en sistemas más sencillos; sin embargo, lo más común es utilizar la *matriz de controlabilidad* (12.12). Se dice que el sistema es completamente controlable si todas las columnas y renglones son linealmente independientes, es decir, si el rango de la matriz es igual al orden del sistema. Una forma de conocer el rango es mediante el método de Gauss o, si la matriz es cuadrada, se puede verificar si el determinante es diferente de cero, esto significa que la matriz es no singular, ergo, de rango completo.

$$C_M = \begin{bmatrix} A & AB & A^2B & \dots & A^{n-1}B \end{bmatrix} \tag{12.12}$$

Ambos sistemas obtenidos, tanto el linealizado a partir del modelado en la Sección 12.1.1 como el obtenido mediante la identificación de sistemas en la Sección 12.1.2, son controlables, por lo que es posible implementar un controlador que modifique los polos del sistema de lazo cerrado. La matriz de controlabilidad del sistema linealizado es (12.13).

$$CM_{lin} = \begin{bmatrix} 0 & 0 & 1.9644 & 1.7642 & 0 & 0 & 0 & 0 \\ 1.9644 & 1.7642 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -31.1265 & 30.0442 & 0 & 0 & 0 & 0 \\ -31.1265 & 30.0442 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{12.13}$$

$$Rank(CM_{lin}) = 4$$

Otro concepto importante aplica para las variables de estado que se quieren estimar mediante un observador, como se necesitará más adelante en la Sección 12.4. Se puede calcular un estado mediante la salida del sistema solamente si este tiene efecto directo sobre ella; si cualquiera de las variables de estado no tiene efecto alguno sobre la salida esta no se puede estimar. A este concepto se le conoce como *Observabilidad*.

"Si el vector inicial de estados, $x(t_0)$, puede ser encontrado de las mediciones $u(t)$ y $y(t)$ en un intervalo finito de tiempo a partir de t_0, se dice que el sistema es observable; de lo contrario el sistema es no observable" (N. Nise, pág 689)[2]

La observabilidad puede ser evaluada mediante el rango de la *matriz de observabilidad*, representada en la ecuación (12.14).

$$O_M = \begin{bmatrix} C \\ CA \\ CA^2 \\ \dots \\ CA^{n-1} \end{bmatrix} \tag{12.14}$$

Como no hay sensores suficientes en nuestro prototipo para medir directamente todos los estados, es necesario utilizar un observador para poder implementar el control en variables de estado y por ello es necesario evaluar primero la observabilidad del sistema. Se puede ver en la ecuación (12.15) que la matriz de observabilidad del sistema linealizado es de rango completo, entonces, el sistema es observable.

$$OM_{lin} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{12.15}$$

$$Rank(OM_{lin}) = 4$$

12.3. Control de realimentación de estados (RE)

Una de las desventajas de las técnicas de control en el dominio de la frecuencia, como la utilizada en la Sección 11, es que después de plantear la ubicación de los dos polos dominantes simplemente se espera que los demás polos no afecten demasiado la aproximación calculada de segundo orden ya que no existen suficientes parámetros ajustables para establecer la posición de todos los polos[2]. En el control por *retroalimentación de estados* se introducen suficientes constantes de control (y las técnicas para calcularlas) como para decidir la ubicación de cada uno de los polos. Sin embargo, esto no define la posición de los ceros, los cuales también pueden afectar el desempeño. Además, este tipo de controladores pueden llegar a ser muy sensibles a variaciones de los parámetros de la planta y el control.

La estrategia de control consiste en realimentar los estados de la planta mediante una matriz de ganancias K, esto con el objetivo de proporcionar suficientes valores como para ubicar todos los polos y estabilizar al sistema. Es común añadir también una matriz de ganancias K_0 a la referencia para minimizar el error en estado permanente del sistema, a esto se le conoce como *pre-compensador estático*[93]. El diagrama de bloques de esta propuesta está en la Figura [12.5], mientras que la ley de control resultante se muestra en (12.16).

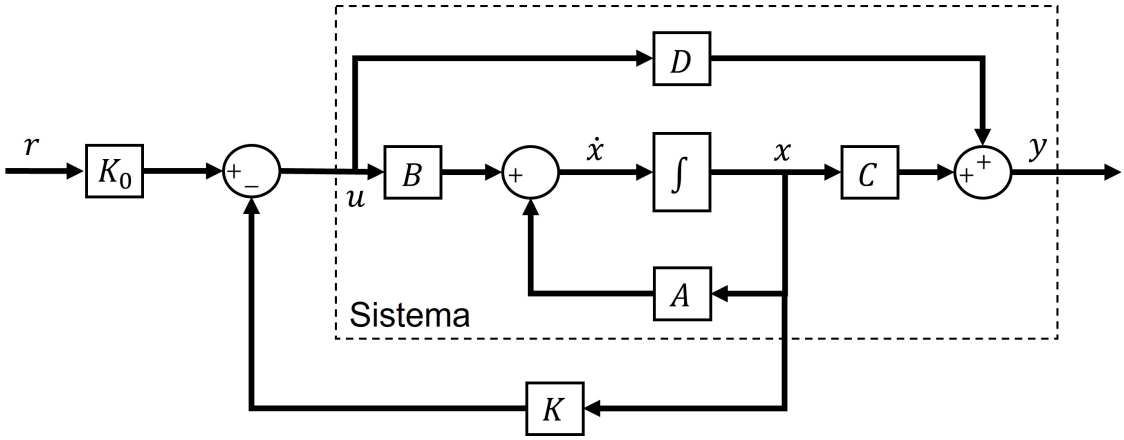


Figura 12.5: *Diagrama de bloques: Control por realimentación de estados y pre-compensador*

$$u = K_0 r - Kx \tag{12.16}$$

La representación del sistema de lazo cerrado, que se puede obtener al sustituir la ley de control en la entrada del sistema, se muestra en (12.17). Con esta nueva configuración es posible obtener la ecuación característica del lazo cerrado en función de los parámetros K, mediante la expresión (12.18). Este polinomio contiene los polos que definen la respuesta dinámica del sistema junto con el controlador. Como ya se hizo anteriormente, se puede igualar esta ecuación con un polinomio deseado, construido a partir de polos previamente definidos y así obtener los valores de K que permitan que el sistema tenga la dinámica esperada.

$$\begin{aligned} \dot{x} &= (A - BK)x + BK_0 r \\ y &= (C - DK)x + DK_0 r \end{aligned} \tag{12.17}$$

$$|\lambda I - (A - BK)| = 0 \tag{12.18}$$

El pre-compensador estático debe minimizar el error en estado permanente, es decir que la salida en estado estacionario y la referencia deben ser iguales. Una forma de lograrlo es utilizar la expresión (12.17) en forma de función de transferencia y forzar a la salida a ser igual que la entrada una vez terminado el transitorio del sistema, como se muestra en el desarrollo de (12.19). Como resultado se obtiene la ecuación (12.20) que permite calcular el valor de K_0 en función de las matrices del sistema y las ganancias de la realimentación de estados.

$$\begin{aligned} G(s) &= C(sI - A + BK)^{-1}BK_0 \\ y_\infty &= G(0)r_\infty \Rightarrow G(0) = I \\ G(0) &= C(-A + BK)^{-1}BK_0 = I \end{aligned} \tag{12.19}$$

$$K_0 = \left[C(-A + BK)^{-1}B\right]^{-1} \tag{12.20}$$

Para poder implementar lo anterior en nuestro proyecto primero es necesario definir la ubicación de los polos deseados para calcular los vectores de K y K_0. La ventaja mencionada anteriormente sobre la capacidad de definir la ubicación de todos los polos en estos sistemas de control solo es ventaja si se conoce la ubicación correcta, lo cual no siempre sucede. Muchas veces definir estos valores depende de la experiencia del ingeniero o del conocimiento del comportamiento de la planta en sí. Sin embargo, una primera aproximación, es definir un desempeño deseado para un sistema de segundo orden y alejar los polos restantes lo suficiente como para que no afecten de forma significativa la dinámica del sistema. Después de varias iteraciones y pruebas se definió el siguiente tiempo de asentamiento t_s y sobrepaso $\%SP$ como desempeño deseado.

$$\begin{aligned} t_s &= 4[s] \\ \%SP &= 0.5\,\% \end{aligned} \tag{12.21}$$

Con estos valores se construye un polinomio deseado de segundo orden que nos proporciona dos polos deseados. Para que éste corresponda con el polinomio de cuarto orden que describe la dinámica de lazo cerrado del controlador, obtenido de la ecuación (12.18), es necesario situar dos polos no dominantes, es decir, que estos dos polos deben de tener una dinámica mínimo cinco veces más rápida que los polos deseados. Este nuevo conjunto de cuatro polos corresponde a las raíces del polinomio deseado, el cual se puede igualar con el resultado de la ecuación (12.18) y se resuelve el sistema de ecuaciones para obtener los valores de la matriz K. En Matlab este proceso de ubicación de polos se facilita mediante la función *K = place(A,B,polos)*.

Es importante mencionar que en un sistema de múltiples entradas y múltiples salidas (MIMO), como es el caso, la matriz K no es única, o sea que es posible determinar varios valores distintos de K que satisfagan la dinámica deseada. En general, deben de seleccionarse aquellos valores que den un mejor desempeño experimental en el sistema con una señal de control suave. El comando *place* de Matlab en el caso MIMO optimiza el proceso para entregar la solución más robusta[94].

Una vez obtenidos los valores de K se puede obtener el vector K_0 mediante la ecuación (12.20) y así reducir el error en estado permanente. Este proceso se realizó en el sistema linealizado previamente y sus resultados se muestran en la Tabla [12.1].

Sistema	Polos	Matriz K	Matriz K_0
Linealizado	$P_1 = -1.0 + 0.5929i$ $P_2 = -1.0 - 0.5929i$ $P_3 = -5$ $P_4 = -5.1$	$K = \begin{bmatrix} -0.0839 & 1.3282 & -1.4824 & -0.3686 \\ 0.1359 & 1.4202 & -1.2049 & -0.1494 \end{bmatrix}$	$K_0 = \begin{bmatrix} -0.0839 & -1.4824 \\ 0.1359 & -1.2049 \end{bmatrix}$

Tabla 12.1: Constantes y polos resultantes del control RE por ubicación de polos

Se utilizó el diagrama de bloques de la Figura [12.6] en Simulink para verificar que el controlador obtenido funciona con el sistema linealizado. Se simuló con condiciones iniciales fuera del punto de equilibrio definido anteriormente en la linealización y con valores deseados en este mismo punto con el objetivo de que las salidas del sistema se estabilicen, es decir que tiendan a cero, tal y como se observa que ocurre en las gráficas (a) y (b) de la Figura [12.8]; recordemos que en estas gráficas se sigue la convención anteriormente propuesta donde los valores delta (Δ) son respecto al punto de equilibrio y donde los datos, escalas y unidades referentes a la altura están marcadas en violeta, mientras que aquellos del ángulo son azules. En general los resultados de las simulaciones son suaves y siguen el desempeño definido.

El objetivo principal es obtener un controlador capaz de controlar el sistema no lineal obtenido en la Sección 10.2.1; en la Figura [12.7] se encuentra el diagrama de bloques con el que se simuló el comportamiento de este controlador con dicho modelo. Es importante recalcar que, para el cálculo de las ganancias del controlador, se utiliza el modelo lineal el cual aproxima el comportamiento alrededor del punto de equilibro del sistema. Esto implica que entre más se alejen los valores de este punto de equilibrio en el sistema real es posible que la respuesta del controlador empeore. Además, tanto las variables deseadas (altura y ángulo) como los estados deben manejarse y referenciarse con respecto a este punto de equilibrio. El bloque llamado "Saturación" (cuyo desglose se encuentra en el Apéndice D) cuenta con lo necesario para manejar esta discrepancia con el sistema no lineal, el cual no está referido al punto de equilibrio. También, en el prototipo real los motores, que son los encargados de llevar a cabo la acción de control, funcionan a través de una señal PWM, que está compuesta de puros valores enteros limitados a los extremos de saturación del motor y está sujeta a la caracterización de los motores que se llevó a cabo en la Sección 13.1. Esto también lo toma en cuenta el bloque de saturación que forma parte de los diagramas de control en la simulación. Los resultados de esta simulación, mostrados en las gráficas (c) y (d) de la Figura [12.8], muestran una gran similitud con aquellos obtenidos con el sistema linealizado.

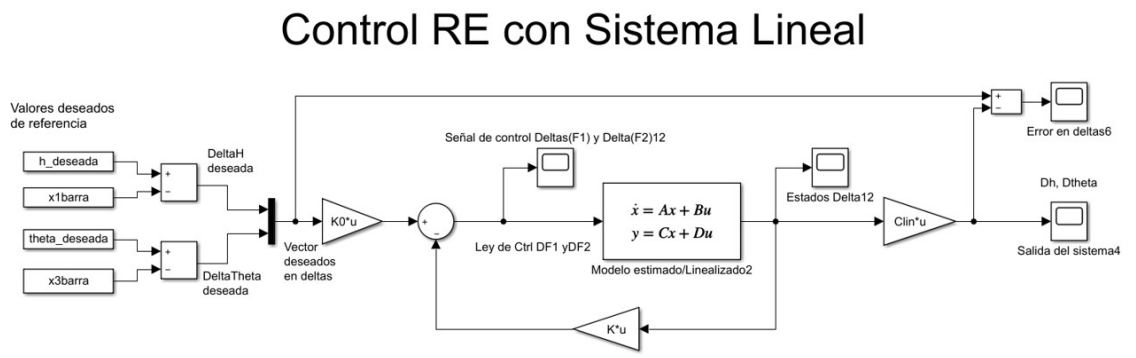


Figura 12.6: *Diagrama de bloques (Simulink): Realimentación de estados y pre-compensador*

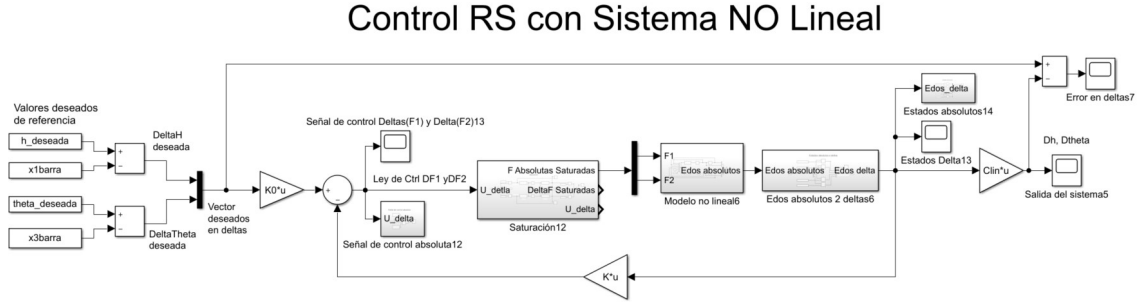
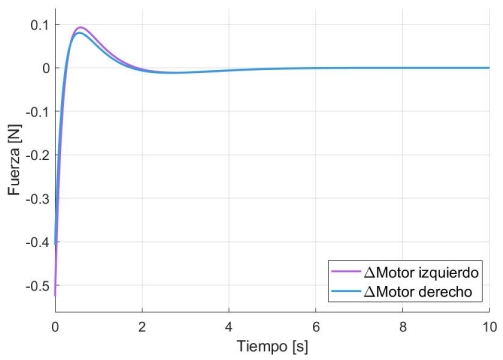
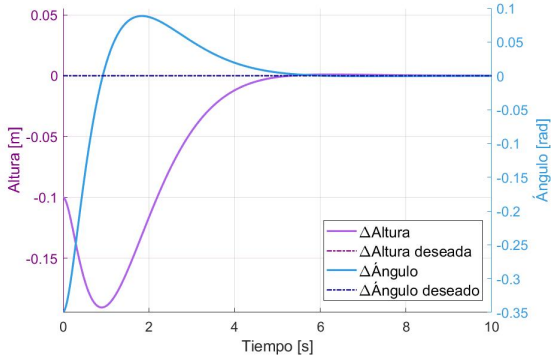


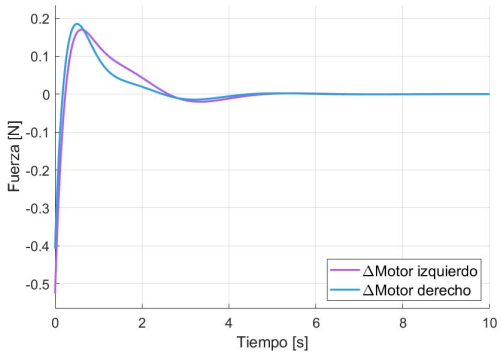
Figura 12.7: *Diagrama de bloques (Simulink): Realimentación de estados y pre-compensador con sistema no lineal*



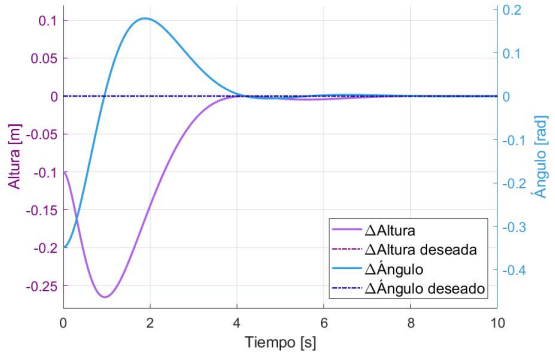
(a) Modelo lineal: Señal de control



(b) Modelo lineal: Variables a controlar



(c) Modelo no lineal: Señal de control



(d) Modelo no lineal: Variables a controlar

Figura 12.8: *Simulación del control RE con modelo lineal y no lineal*

12.3.1. Control óptimo LQR

Como se mencionó en la sección pasada, la facilidad que nos brinda el control RE de posicionar todos los polos puede no ser una ventaja cuando no se conoce dónde posicionarlos, ya que un lugar incorrecto puede ser perjudicial para el sistema[2]. El acercamiento anterior, donde se deciden dos polos dominantes tal que el desempeño sea una aproximación de segundo orden, no siempre está garantizada. Muchas veces el proceso de obtener los valores correctos es iterativo de prueba y error, dependiente de la experiencia o familiaridad del diseñador con la planta.

Según Anderson y Moore (1989)[95], existe un área muy amplia del control que busca facilitar el diseño de sistemas más complejos llamada *Control óptimo*. Con estas técnicas se quiere que el sistema obtenido no sea simplemente estable o cumpla ciertas características asociadas con el control clásico, sino que sea el mejor resultado posible dadas ciertas condiciones y restricciones; de ahí el nombre *Óptimo*. Una de las técnicas más conocidas es cuando el controlador utiliza medidas cuantitativas del desempeño de un sistema elegidos de manera que se dé énfasis a las especificaciones importantes del sistema; conocidos como índices de desempeño[96] (por ejemplo el índice de desempeño cuadrático), con lo cual se puede consigue un control lineal óptimo en el sentido de las especificaciones importantes elegidas. A diferencia de otros métodos, estos resultan fáciles de implementar, computar, simular y calcular. En general son suficientes para una gran variedad de sistemas.

Para el caso particular del prototipo construido, al principio no se conocía un valor adecuado para los polos y se propusieron muchos desempeños distintos mediante las ecuaciones de segundo orden (11.24) y (11.25) de forma iterativa durante el diseño del controlador RE. Sin embargo, no se puede estar seguro si el sistema es capaz de lograr estos desempeños de forma correcta, incluso después de simular los resultados, hasta comprobarlos en el real. Muchas veces los valores propuestos resultaban demasiado rápidos para el sistema,

otras veces demasiado lentos e incluso con acciones de control muy grandes las cuales alcanzaron y superaron los límites físicos del sistema. Para mejorar las propuestas de desempeño y aliviar un poco el proceso iterativo se propone obtener una posición de los polos mediante un regulador lineal cuadrático (LQR) y a partir de esta propuesta, probar opciones similares buscando un mejor desempeño[97].

El método de control LQR (Linear Quadratic Regulator) es una propuesta de control similar al RE donde se realimentan los estados mediante una matriz K con la ley de control $u = r - Kx$. Sin embargo, los valores de las matrices de ganancias se calculan tal que se minimice una función de costo. En este caso la ecuación (12.22) representa esta función, donde x es el vector de estados y u el vector de entradas.

$$J = \int_0^\infty \left(x^T Q x + u^T R u\right) dt \tag{12.22}$$

Los elementos Q y R son matrices de pesos que se proponen según la importancia que se le quiere dar a una variable durante la optimización. Como el objetivo es minimizar la función de costo, aquellas variables con mayor peso serán las que se minimicen más. Ambas matrices deben ser cuadradas simétricas y con valores mayores a cero, cuyo tamaño debe corresponder con el número de variables de estado y entradas correspondientemente.

La intención de esta sección no es hacer el desarrollo matemático para la obtención de K, sino demostrar el el algoritmo realizado mediante esta estrategia de control óptimo.El procedimiento, de forma simplificada, para obtener las ganancias consiste en los siguientes pasos:

1. Proponer los valores de peso de las matrices Q y R dependiendo de las variables que se quieran priorizar tal que:
$$Q = Q^T > 0, \quad R = R^T > 0 \tag{12.23}$$

2. Obtener la solución para la matriz P de la siguiente ecuación conocida como la *Ecuación algebráica de Riccati*
$$PA + A^T P - PBR^{-1}B^T P = -Q \tag{12.24}$$

3. Obtener los parámetros para el control de retroalimentación de estados tal que:
$$K = R^{-1}B^T P \tag{12.25}$$

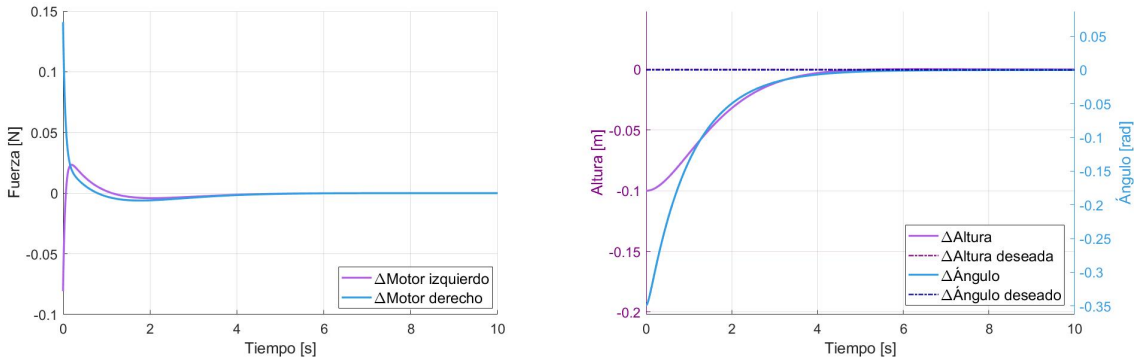
4. Revisar el desempeño obtenido e iterar desde el paso 1 variando los pesos de las matrices hasta obtener un resultado aceptable.

El software de Matlab cuanta con la función *[K,S,e] = lqr(sys,Q,R)*[98] que facilita este proceso, donde K es la matriz de ganancias calculadas mediante la ecuación de Riccati y e son los polos de lazo cerrado obtenidos del sistema. La salida S de lqr es la solución de la ecuación de Riccati para el modelo de espacio de estados explícito equivalente. En el caso de nuestra planta, se sugieren unas matrices de pesos donde aquellos relacionados a las entradas son mayores para que la acción de control sea suave; los resultados para los polos y las matrices de ganancias obtenidas se muestran en la Tabla [12.2].

Sistema	Matrices de pesos	Matriz K_{lqr}	Polos
Linealizado	$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ $R = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}$	$K = \begin{bmatrix} 0.3119 & 0.5115 & -0.3205 & -0.3359 \\ 0.3205 & 0.5272 & 0.3119 & 0.3285 \end{bmatrix}$ $K_0 = \begin{bmatrix} 0.3119 & -0.3205 \\ 0.3205 & 0.3119 \end{bmatrix}$	$P_{1,2} = -0.9673 \pm 0.4920i$ $P_3 = -1.0013$ $P_4 = -19.3212$

Tabla 12.2: Constantes y polos resultantes del control LQR

La respuesta del sistema del controlador RE calculado mediante LQR se puede ver en las gráficas de la Figura [12.9]; éstas muestran la respuesta del controlador y el sistema lineal del diagrama de bloques de la Figura [12.6] utilizando las matrices K y K_0 calculadas por control óptimo. Se puede apreciar una respuesta más suave, con menos sobrepaso y una señal de control menos agresiva que aquella obtenida mediante la aproximación de un sistema de segundo orden. Debido a la similitud de resultados entre el controlador con el modelo no lineal y el linealizado, como se aprecia en la sección anterior, se decidió solamente mostrar la simulación de este último.



(a) Señal de control

(b) Variables a controlar

Figura 12.9: *Simulación del control LQR con el modelo lineal*

Como se explicó anteriormente, este proceso es una manera de conocer la ubicación óptima (donde la optimización se basa en en los pesos asignados) de los polos de lazo cerrado para una configuración donde se realimentan los estados, pero esto no significa que los resultados sean definitivos; el diseñador puede tomarlos como punto de partida para buscar polos que brinden un desempeño más adecuado a las especificaciones de diseño. El desempeño deseado propuesto en la sección anterior se obtuvo por prueba y error tomando los resultados del LQR como base, por ello los polos dominantes son tan similares.

12.4. Control de realimentación de salidas RS (Observador)

El control RE se basa en la retroalimentación de *todos* los estados mediante ganancias ajustables; cada uno de ellos se debe medir mediante distintos tipos de sensores, hardware y software. Algunas variables físicas como voltaje, temperatura o distancia son sencillas de medir, pero otras como la velocidad, aceleración, radiactividad, deformación u otros estados de sistemas más complejos requieren equipo especializado, por ejemplo, acelerómetros, unidades de medición inerciales, contador de Geiger o probetas de medición sumamente precisas. Sin embargo, en algunas aplicaciones, medir estas variables puede resultar una tarea muy difícil, costosa o poco viable; es por ello por lo que si se quiere implementar un controlador de este estilo se debe tener cuidado con los estados que se escogen. Una solución alternativa es estimar los estados que no se pueden medir y enviar estos *estados estimados* al controlador en lugar de una medición directa[78]. El *observador* (también llamado estimador) es el encargado de calcular estas variables de la planta que no se pueden medir.

Como lo que se busca es calcular los estados que no se pueden medir y ya se tiene un modelo en variables de estado de la planta, es normal proponer que su entrada se aplique en paralelo de igual forma al modelo en variables de estado y de este se obtengan los estados faltantes. A esta configuración se le conoce como observador de lazo abierto, sin embargo, tiene múltiples limitantes. La primera es que si la planta (y por ende el modelo) no es estable por si sola el observador tampoco lo será y la diferencia entre los estados estimados y los reales puede crecer indefinidamente. Otra limitante es que la dinámica del observador será igual de lenta que la de la planta original y en caso de que las condiciones iniciales y/o los valores de las matrices no sean iguales, los estados estimados con los que se alimenta al controlador no serán buenas aproximaciones, resultando en un controlador ineficaz.

De acuerdo con el autor Nise (2011)[2], esto se puede corregir si se utiliza una disposición de lazo cerrado como se muestra en la Figura [12.10]. En esta configuración los estados estimados se obtienen a partir de las entradas y las salidas del sistema, es decir que, además de la señal de control (u), el observador también tiene la salida del sistema (y) como entrada. Esto le permite realimentar el error entre la salida estimada y la real con la finalidad de reducirlo. La matriz de ganancias L que realimenta esta diferencia le permite al observador tener una dinámica más rápida que la del sistema; esto deriva en que los estados que se envían al control sean aproximaciones del valor instantáneo de los estados reales. Como se mencionó anteriormente en la Sección 12.2, antes de comenzar a diseñar un observador de este tipo se debe verificar si el sistema es observable.

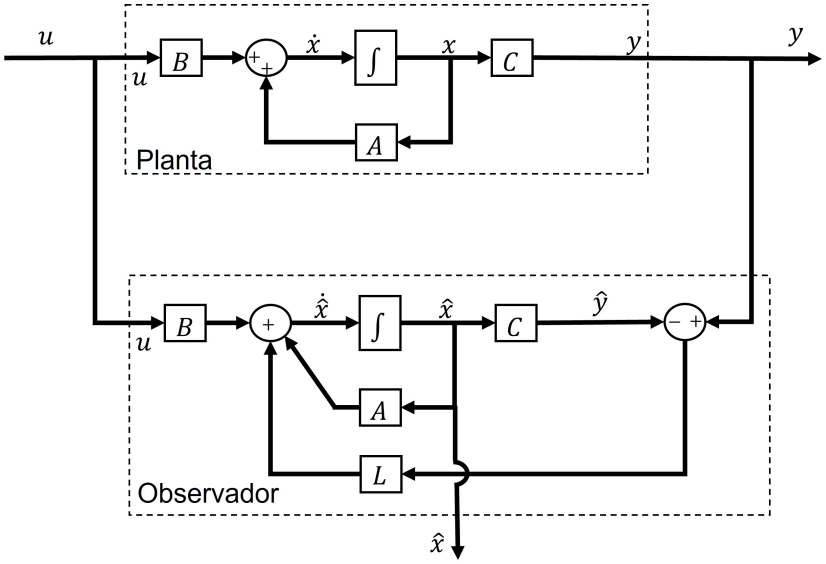


Figura 12.10: *Diagrama de bloques: Observador de lazo cerrado*

El error de estimación se define como $e_x = x - \hat{x}$ para poder representar la dinámica del observador mediante un modelo de variables de estado, donde x son los estados originales y $\hat{x}$ son los estimados. En esta representación el vector de estados corresponde a los errores de estimación de cada uno de ellos, como se muestra en la ecuación (12.26). El segmento $(A - LC)$ contiene la dinámica del error de estimación y al modificar L se puede estabilizar y definir la rapidez con la que esto sucede.

$$\begin{aligned} \dot{e_x} &= (A - LC)e_x \\ y - \hat{y} &= Ce_x \end{aligned} \tag{12.26}$$

También es posible representar en variables de estado la dinámica de lazo cerrado, cuya salida son los estados estimados y la entrada se define como las entradas y salidas de la planta, tal como se muestra en la ecuación (12.27). Esta representación facilita la implementación y simulación de los observadores.

$$\begin{aligned} \dot{\hat{x}} &= (A - LC)\hat{x} + \begin{bmatrix} L & B \end{bmatrix} \begin{bmatrix} y \\ u \end{bmatrix} \\ \hat{x} &= I\hat{x} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} y \\ u \end{bmatrix} \end{aligned} \tag{12.27}$$

El cálculo del vector L es similar al de las ganancias K del control RE, donde se busca que las ganancias produzcan una dinámica de lazo cerrado deseada para el observador. Para facilitar esta tarea, se puede utilizar el concepto de *dualidad de sistemas*[99]. Un sistema se dice dual si cumple con lo establecido en la ecuación (12.28), donde $\dot{\zeta}$ es dual con $\dot{x}$. Por medio de la consideración de un sistema de este tipo se pueden analizar

ciertas propiedades del sistema original, por ejemplo, si el sistema dual es observable, entonces el sistema original es controlable y viceversa. También, si el sistema dual es estable, se puede decir que el sistema original es igualmente estable.

$$\begin{aligned} \dot{x}(t) = Ax(t) + Bu(t) \quad &\Leftrightarrow \quad \dot{\zeta}(t) = A^T\zeta(t) + C^T u(t) \\ y(t) = Cx(t) + Du(t) \quad &\Leftrightarrow \quad y(t) = B^T\zeta(t) + D^T u(t) \end{aligned} \tag{12.28}$$

Lo cual se resume en el siguiente teorema:

"Sea el par (A,C). Todos los valores propios de $(A-LC)$ pueden ser arbitrariamente asignados a un vector apropiado L si y solo si el par (A,C) es observable." [99]

Es decir que para diseñar un observador para el sistema original solamente es necesario diseñar un controlador RE para el sistema dual, siempre y cuando el sistema original sea observable. Para ello, solo es necesario definir la dinámica que se desea del observador.

Debido a que se desea que la dinámica dominante sea la del controlador, se recomienda que la del observador sea significativamente más rápida; esto puede ser al menos 5 veces más rápida[78], 10 veces más rápida[2] o entre 2 y 6 veces más rápida[100] dependiendo del autor. También, lo mejor es que su respuesta transitoria sea sobreamortiguada o subamortiguada con un sobrepaso mínimo para evitar las oscilaciones[78]. Un controlador con los estados realimentadas por un observador tiene un atractivo especial: puede ser considerado como compuesto por dos partes, una la realimentación de estados y la otra el observador. La ganancia de realimentación K puede ser calculada como si todos los estados pudiesen ser medidos, y solo depende de A y B, mientras que la ganancia del observador L depende solo de A y C. La propiedad de que la ubicación de los polos de cada parte puede ser independiente se conoce como *principio de separación*.[101][100]

Para el caso de este proyecto se decidió que la dinámica del observador sea 10 veces más rápida que la del lazo cerrado del control y sobreamortiguada. El polinomio deseado para el observador que se obtiene de estas raíces se iguala a su ecuación característica (12.29).

$$|sI - (A - LC)| = 0 \tag{12.29}$$

Finalmente, el sistema de ecuaciones resultante se resuelve para obtener los valores de L; para esto también se utilizó el comando *place*[94] de Matlab. Las ganancias obtenidas, junto con los polos, se encuentran en la tabla [12.3] mientras que el modelo del observador en variables de estado se muestra en la ecuación (12.30).

Sistema	Polos del observador	Vector L
Linealizado	$P_1 = -10 \quad P_2 = -10$ $P_3 = -50 \quad P_4 = -51$	$L = \begin{bmatrix} 60.0004 & -0.0196 \\ 500.0038 & -0.1960 \\ -0.0196 & 60.9996 \\ -0.1960 & 509.9962 \end{bmatrix}$

Tabla 12.3: Constantes resultantes para el observador

$$\dot{\hat{x}} = \begin{bmatrix} -60.0004 & 1 & 0.0196 & 0 \\ -500.0038 & 0 & 0.1960 & 0 \\ 0.0196 & 0 & -60.9996 & 1 \\ 0.1960 & 0 & -509.9962 & 0 \end{bmatrix} \hat{x} + \begin{bmatrix} 60.0004 & -0.0196 & 0 & 0 \\ 500.0038 & -0.1960 & 1.9644 & 1.7642 \\ -0.0196 & 60.9996 & 0 & 0 \\ -0.1960 & 509.9962 & -31.1265 & 30.0442 \end{bmatrix} \begin{bmatrix} y \\ u \end{bmatrix} \tag{12.30}$$

$$\hat{x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \hat{x} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y \\ u \end{bmatrix}$$

Cuando se realimentan los estados estimados a un controlador RE se le conoce como *Control de retroalimentación de salidas (RS)*. Se utilizó Simulink para verificar la respuesta de esta nueva propuesta, tanto en el sistema lineal como en el sistema no lineal. Ambos diagramas de bloques se muestran en las figuras [12.11] y [12.12], donde se utiliza la representación del observador de lazo cerrado en variables de estado. Para facilitar la comparación, ambas gráficas se muestran como variables referidas al punto de equilibrio (Δ). Los valores deseados corresponden a este mismo punto y las condiciones iniciales del sistema están fuera de él.

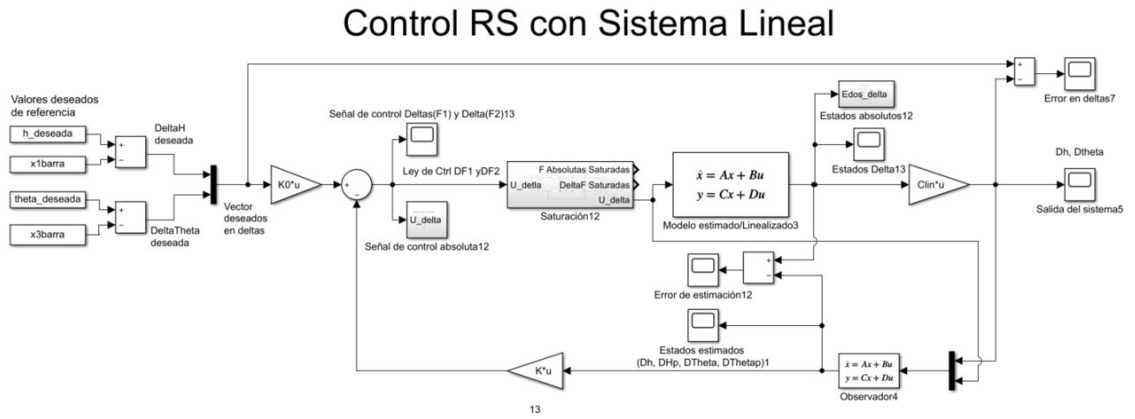


Figura 12.11: *Diagrama de bloques (Simulink): Control RS con sistema linealizado*

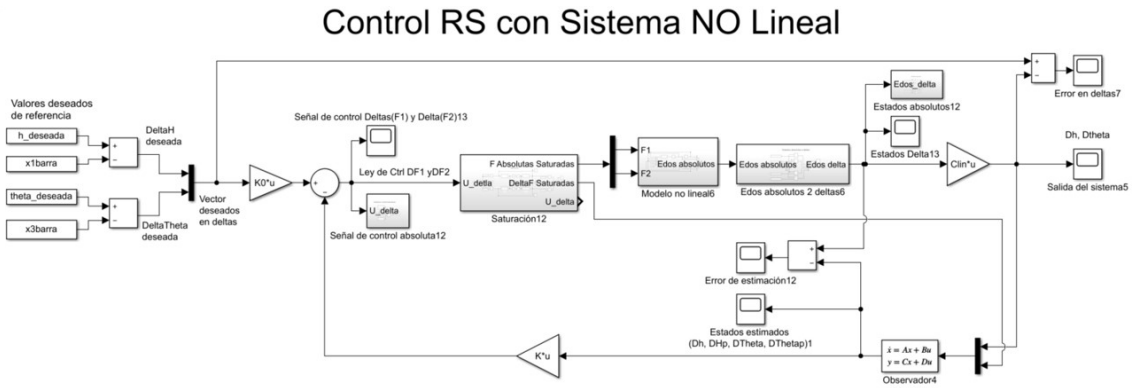
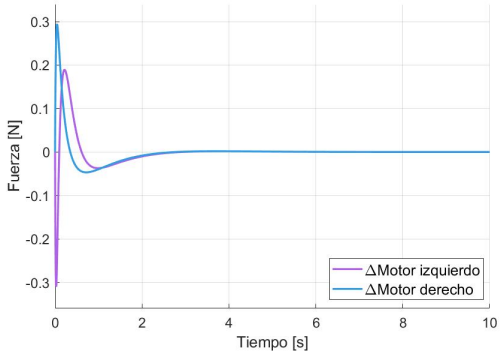


Figura 12.12: *Diagrama de bloques (Simulink): Control RS con sistema no lineal*

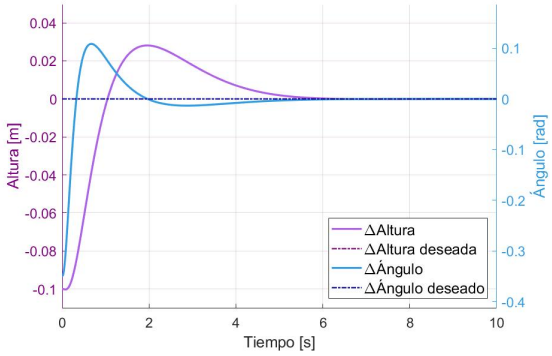
Las gráficas de la Figura [12.13] muestran los resultados de estas simulaciones. En ambos casos el observador logra su propósito de estimar los estados de la planta. Esto se puede ver en las gráficas (e) y (f) donde se restan los estados estimados por el observador a los estados originales del sistema para obtener el error de estimación, el cual tiende a ser cero en los 4 estados después de un tiempo. La discrepancia inicial se debe a que las condiciones iniciales del observador y del sistema son diferentes; sin embargo, la retroalimentación propuesta logra corregir esto.

Debido a que el controlador se calculó para el sistema lineal su respuesta tiene el desempeño esperado en las gráficas (a) y (b), mientras que el modelo no lineal representado en (c) y (d) se comporta de manera diferente; esto debido a que la linealización sólo es una mera aproximación y no una representación exacta. Aunque el desempeño discrepe, el objetivo de estabilizar a la planta no lineal utilizando un controlador de realimentación de estados y un observador resultó posible. Se puede ver que la regulación no se cumple del todo ya que las variables a controlar muestran un error en estado permanente, principalmente en el modelo no lineal. El pre-compensador estático no es eficaz en solucionar esta tarea debido a que este sólo es efectivo

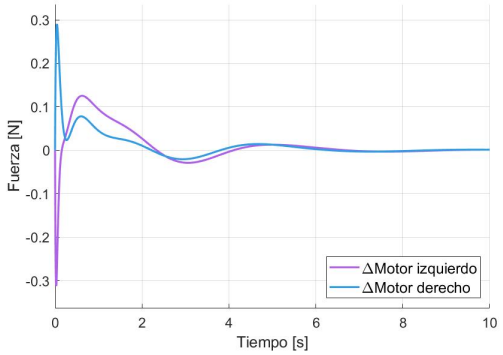
cuando la referencia no cambia[93]; entre más lejos del punto de equilibrio sobre el que se diseñó se encuentre la referencia, mayor será el error.



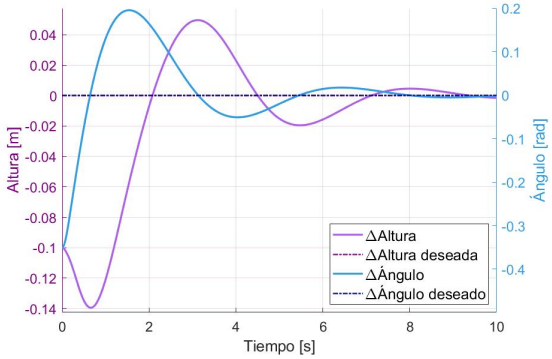
(a) Modelo lineal: Señal de control



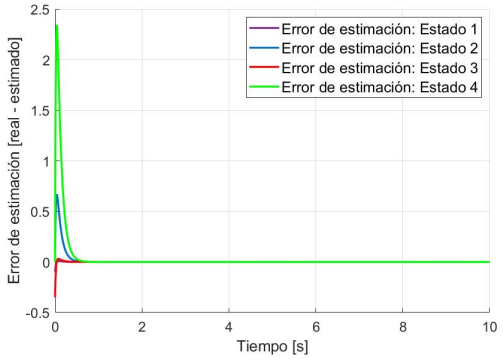
(b) Modelo lineal: Variables a controlar



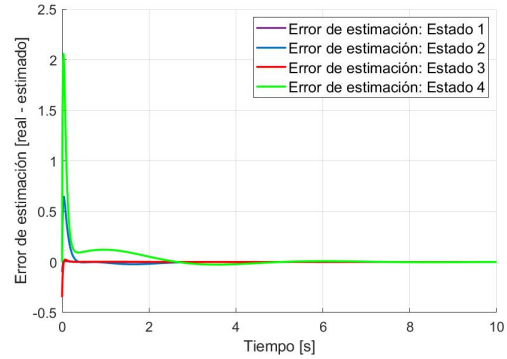
(c) Modelo no lineal: Señal de control



(d) Modelo no lineal: Variables a controlar



(e) Modelo lineal: Error de estimación



(f) Modelo no lineal: Error de estimación

Figura 12.13: *Simulación del control RS con el modelo lineal y no lineal*

12.5. Control RS con acción integral

Una de las desventajas de utilizar el pre-compensador estático como forma de llevar a cero el error en estado permanente es que este es muy poco robusto ante las variaciones de la referencia y discrepancias e imprecisiones en el modelado de la planta[93], debido a esto en la sección anterior no cumplió con el objetivo de regulación.

Para conseguir un sistema capaz de seguir una referencia sujeta a cambios y perturbaciones con un error en estado permanente nulo y que sea robusto ante incertezas del modelo, se requiere un pre-compensador distinto. Una nueva propuesta para ello es añadir un integrador del error en el lazo directo del sistema de control para lograr una configuración similar a un PI con todas las ventajas que implica un control en variables de estado. A esta nueva configuración se le conoce como *Retroalimentación de salidas con acción integral* o *Controlador RS tipo Servo*[93].

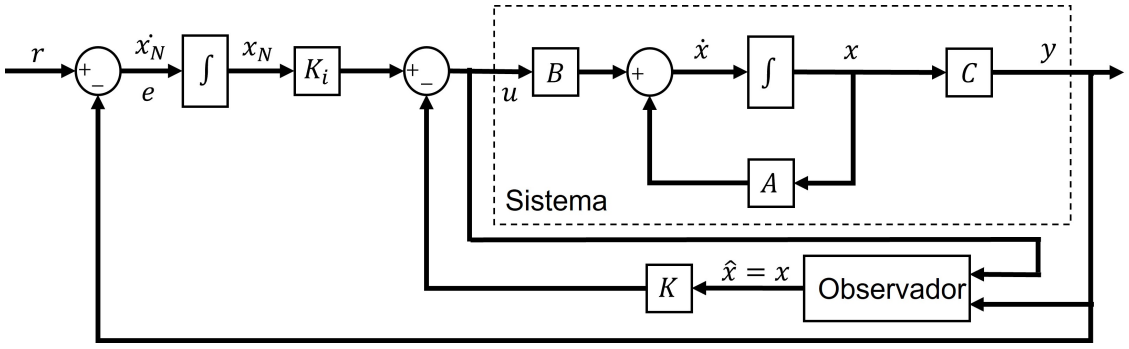


Figura 12.14: *Diagrama de bloques: Controlador RS tipo servo*

La adición de una acción integral puede cambiar drásticamente la dinámica de lazo cerrado del sistema al incrementar el tiempo de asentamiento, aumentar la amplitud y numero de oscilaciones, incluso inestabilizar al sistema; por ello es necesario modificar la forma de calcular las ganancias considerando esta nueva característica.

Para esto es necesario definir un nuevo estado (o conjunto de estados para sistemas MIMO) x_N que represente la integral del error entre la salida y la referencia. Con esta nueva variable es posible definir al sistema completo como un modelo en variables de estado al definir un conjunto de matrices A, B, C y D aumentadas que consideren a los nuevos estados agregados[2]. Este sistema aumentado está representado en la ecuación (12.31), mientras que la nueva acción de control que se lleva a cabo sobre la planta se muestra en la ecuación (12.32).

$$\begin{aligned} \dot{x_a} &= A_a x_a + B_a u + E_a r \\ y &= C_a x_a \end{aligned} \tag{12.31}$$

$$u = -Kx + K_i x_N = -K_a x_a \tag{12.32}$$

$$x_a = \begin{bmatrix} x \\ x_N \end{bmatrix} \quad A_a = \begin{bmatrix} A & \mathbf{0} \\ -C & \mathbf{0} \end{bmatrix} \quad B_a = \begin{bmatrix} B \\ \mathbf{0} \end{bmatrix} \quad E_a = \begin{bmatrix} \mathbf{0} \\ I \end{bmatrix} \quad K_a = \begin{bmatrix} K & -K_i \end{bmatrix}$$

La dinámica del sistema completo de lazo cerrado también se puede representar en variables de estado al sustituir la ley de control en el sistema aumentado tal que:

$$\begin{bmatrix} \dot{x} \\ \dot{x}_N \end{bmatrix} = \begin{bmatrix} (A-BK) & BK_i \\ -C & 0 \end{bmatrix} \begin{bmatrix} x \\ x_N \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} r \tag{12.33}$$
$$y = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x \\ x_N \end{bmatrix}$$

"Si el par (A,B) es controlable y $G(s) = C(sI-A)^{-1}B$ no tiene ceros en el origen, entonces los autovalores de la matriz aumentada A_a pueden ser arbitrariamente definidos por la realimentación de estados K_a"[93]

Lo anterior nos indica que es posible utilizar el mismo procedimiento realizado en el control RE (ya sea por ubicación de polos o LQR) para calcular los valores de las matrices K y K_i. Pero esta vez se debe emplear el sistema aumentado (12.31). Nótese que este sistema y su ecuación característica aumentaron de orden respecto al original al agregar la acción integral. Se deben añadir los polos necesarios no dominantes para que el polinomio deseado y el característico puedan ser igualados. En el caso de nuestra planta experimental, debido a que son dos entradas y dos salidas, la acción integral añade dos estados nuevos, lo cual significa dos polos extra. Estos polos surgen de la misma propuesta de desempeño establecida anteriormente.

La ecuación (12.34) muestra la matriz aumentada de ganancias obtenida para el modelo linealizado aumentado con el comando *place*[94] de Matlab. La Tabla [12.4] contiene los polos deseados y las ganancias obtenidas, derivadas de K_a, para los parámetros de control. Es importante tener en cuenta que los valores de K y K_i obtenidos con el procedimiento, corresponden a la función de la ecuación (12.32), esto quiere decir que los valores de K_i deben de tomar en cuenta el signo negativo cuando se implementen. La tabla muestra los valores listos para implementar en un controlador correspondiente con el de la Figura [12.14].

$$K_a = \begin{bmatrix} 17.3148 & 4.1826 & -1.0535 & 0.2843 & -14.4317 & 14.3810 \\ 15.9576 & 3.9962 & 1.7087 & 0.8548 & -13.6079 & 10.7271 \end{bmatrix} \tag{12.34}$$

Sistema	Polos	Matriz K	Matriz K_i
Linealizado	$P_1 = -1+0.5929i$ $P_2 = -1-0.5929i$ $P_3 = -5$ $P_4 = -5.1$ $P_5 = -10+5.9294i$ $P_6 = -10-5.9294i$	$K = \begin{bmatrix} 17.3148 & 4.1826 & -1.0535 & 0.2843 \\ 15.9576 & 3.9962 & 1.7087 & 0.8548 \end{bmatrix}$	$K_i = \begin{bmatrix} 14.4317 & -14.3810 \\ 13.6079 & -10.7271 \end{bmatrix}$

Tabla 12.4: Constantes resultantes para el controlador RE/RS con acción integral

De igual forma que con el control de retroalimentación de estados, es necesario implementar un observador para estimar las variables de estado. Como la dinámica de lazo cerrado deseada del observador es la misma que aquella en el control RS, se utiliza el mismo calculado en dicha sección. Los diagramas de bloques de Simulink para este controlador se muestran en las figuras [12.15] y [12.16]. El primero es para simular la respuesta con el sistema linealizado y el segundo con el no lineal.

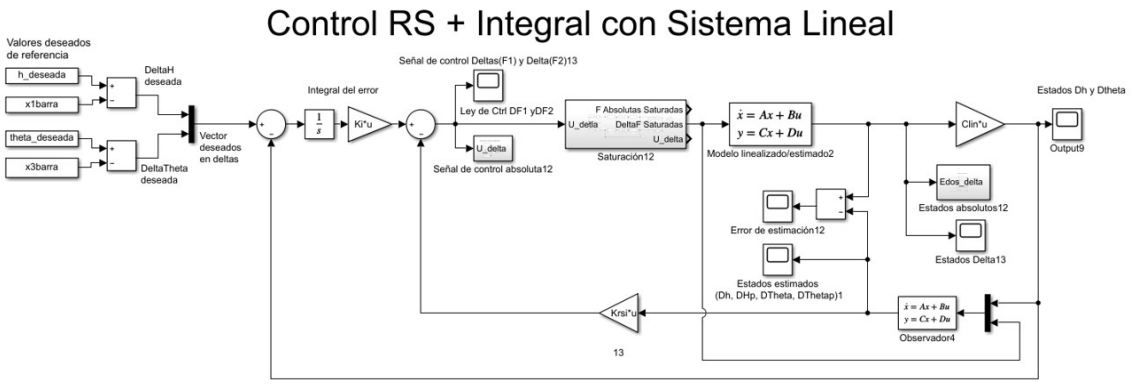


Figura 12.15: *Diagrama de bloques (Simulink): Controlador RS servo con sistema linealizado*

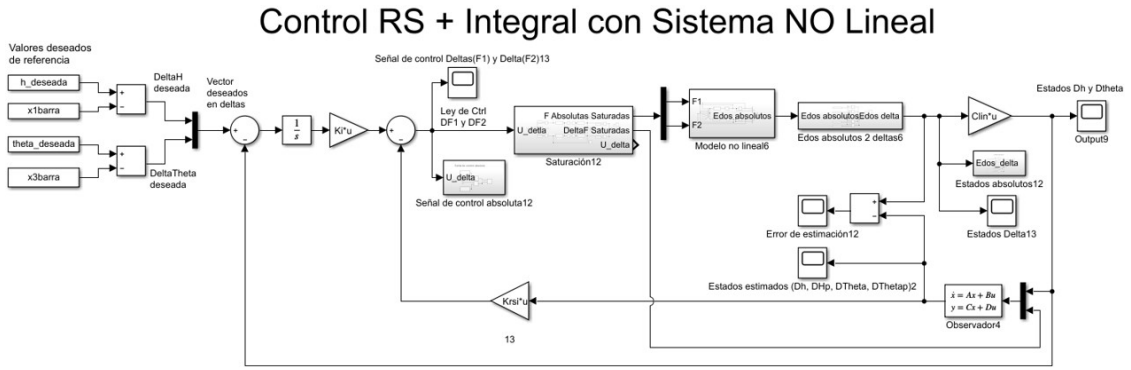
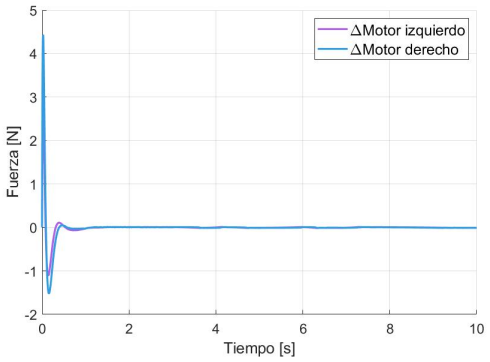
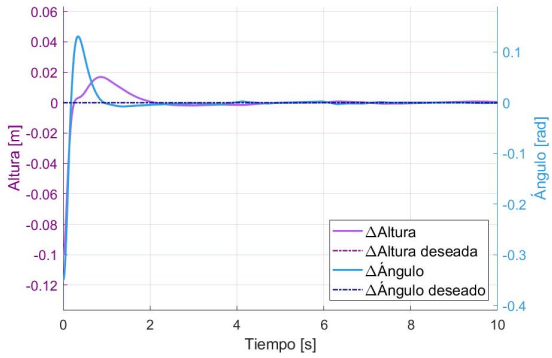


Figura 12.16: *Diagrama de bloques (Simulink): Controlador RS servo con sistema no lineal*

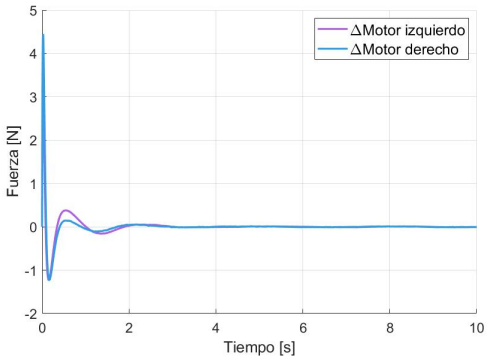
Se puede apreciar que con esta nueva adición ambos modelos consiguen llevar las variables a controlar al punto de equilibrio sin error en estado permanente. Además, la dinámica del sistema es muy similar a la esperada, sus señales de control son suaves y están dentro del rango de saturación. Ambas simulaciones indican un controlador eficaz y suficiente.



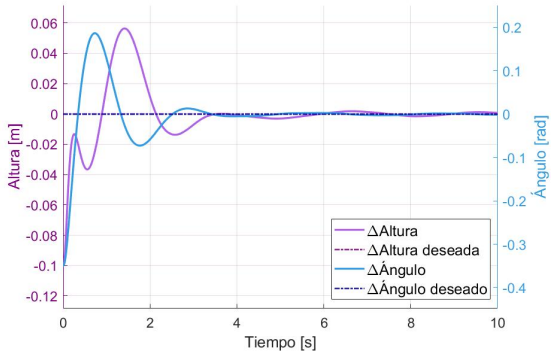
(a) Modelo lineal: Señal de control



(b) Modelo lineal: Variables a controlar



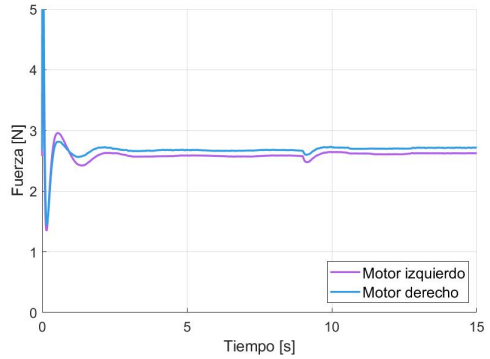
(c) Modelo no lineal: Señal de control



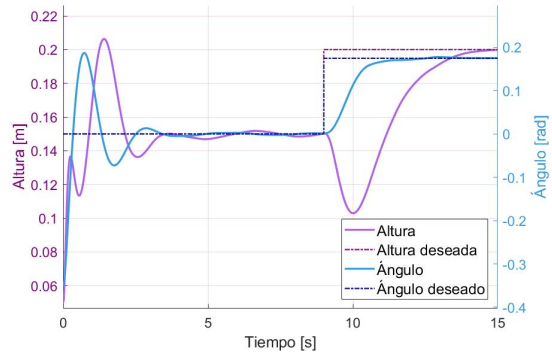
(d) Modelo no lineal: Variables a controlar

Figura 12.17: *Simulación del control RS+integral con el modelo lineal y no lineal*

La adición del término integral y la simulación con un error en estado permanente nulo indican que el sistema cumple con el objetivo de regulación, es decir que, ante un cambio de referencia, el sistema es capaz de modificar las variables hasta cumplir con este nuevo valor deseado. Cuando se tiene este comportamiento se les conoce como *sistemas servo*. En la Figura [12.18] se muestra la dinámica del controlador junto con el sistema no lineal ante tal situación con las variables representadas de forma absoluta, tal cual como son en el sistema real.



(a) Modelo no lineal: Señal de control



(b) Modelo no lineal: Variables a controlar

Figura 12.18: *Simulación del control RS servo con el sistema no lineal y cambio de referencia*

12.6. Control RS con acción integral discreto

Una vez que se tienen resultados satisfactorios en la simulación del control continuo de retroalimentación de salidas con acción integral, se debe de implementar y probar en el modelo real. Debido a que el microcontrolador encargado de llevar a cabo la ejecución de este proceso funciona de manera discreta y llevará a cabo los cálculos de manera cíclica donde la señal de control se actualiza únicamente al fin de ciclo, es necesario discretizar esta última propuesta de control para poder compensar el tiempo de ejecución donde la señal no se actualiza. Para ello, se propone el control discreto de la Figura [12.19] donde las dimensiones de los vectores de entrada, salida, integrador y señal de control se mantienen igual que en la configuración de tiempo continuo. Las ecuaciones en diferencias (12.35), (12.36) y (12.37) describen la planta discreta, el vector integrador y el vector de control respectivamente.

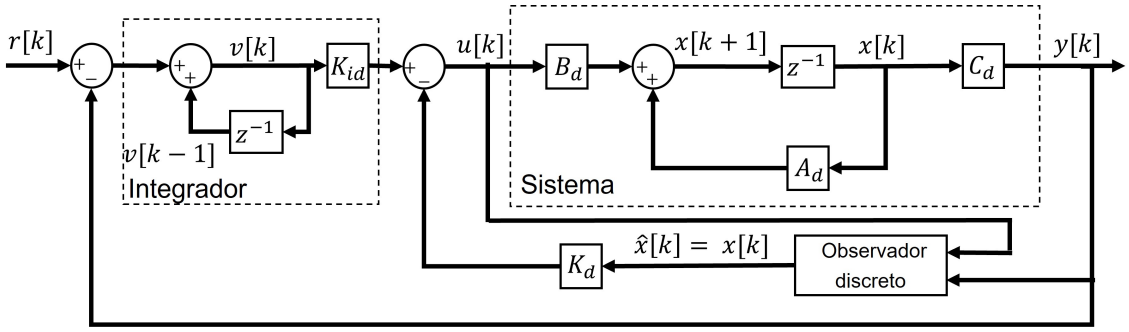


Figura 12.19: *Diagrama de bloques: Controlador RS tipo servo discreto*

$$\begin{aligned} x[k+1] &= A_d x[k] + B_d u[k] \\ y[k] &= C_d x[k] \end{aligned} \tag{12.35}$$

$$v[k] = v[k-1] + r[k] - y[k] \tag{12.36}$$

$$u[k] = -K_d x[k] + K_{id} v[k] \tag{12.37}$$

En este sistema los parámetros de diseño son K_d y K_{id}, pero para poder determinar dichas matrices de tal modo que el sistema tenga los polos de lazo cerrado deseados, es necesario modificar un poco el procedimiento y las matrices aumentadas vistas anteriormente. Esta modificación de realizó de acuerdo con procedimiento propuesto por el autor Katsuhiko Ogata (1996)[102].

Primero, se puede reescribir la ecuación del integrador (12.36) de la siguiente manera:

$$\begin{aligned} v[k+1] &= v[k] + r[k+1] - y[k+1] \\ &= v[k] + r[k+1] - C_d(A_d x[k] + B_d u[k]) \\ &= -C_d A_d x[k] + v[k] - C_d B_d u[k] + r[k+1] \end{aligned} \tag{12.38}$$

De las ecuaciones (12.35), (12.38) y (12.37) se obtiene que:

$$\begin{aligned} u[k+1] &= -K_d x[k+1] + K_{id} v[k+1] \\ &= -K_d \left(A_d x[k] + B_d u[k]\right) + K_{id} \left(-C_d A_d x[k] + v[k] - C_d B_d u[k] + r[k+1]\right) \\ &= \left(K_d - K_d A_d - K_{id} C_d A_d\right) x[k] + \left(I - K_d B_d - K_{id} C_d B_d\right) u[k] + K_{id} r[k+1] \end{aligned} \tag{12.39}$$

Ya que $u[k]$ es una combinación lineal de los vectores de estado $x[k]$ y $v[k]$, se define un nuevo vector de estados aumentados compuesto por $x[k]$ y $u[k]$ y se plantea un nuevo sistema de acuerdo con las ecuaciones (12.35) y (12.39).

$$\begin{bmatrix} x[k+1] \\ u[k+1] \end{bmatrix} = \begin{bmatrix} A_d & B_d \\ K_d - K_d A_d - K_{id} C_d A_d & I - K_d B_d - K_{id} C_d B_d \end{bmatrix} \begin{bmatrix} x[k] \\ u[k] \end{bmatrix} + \begin{bmatrix} 0 \\ K_{id} \end{bmatrix} r[k+1] \tag{12.40}$$

$$y[k] = \begin{bmatrix} C_d & 0 \end{bmatrix} \begin{bmatrix} x[k] \\ u[k] \end{bmatrix}$$

En este nuevo sistema, los polos de lazo cerrado están determinados por el mismo sistema y no dependen de la referencia. Para poder aplicar directamente la técnica de sintonización de ubicación de polos (ya sea por un desempeño deseado o los polos obtenidos por LQR) se puede considerar que la referencia es constante, o una entrada escalón, tal que $r[k] = r[\infty] = r$ y así el sistema (12.40) se puede reescribir de la siguiente forma:

$$\begin{bmatrix} x[k+1] \\ u[k+1] \end{bmatrix} = \begin{bmatrix} A_d & B_d \\ K_d - K_d A_d - K_{id} C_d A_d & I - K_d B_d - K_{id} C_d B_d \end{bmatrix} \begin{bmatrix} x[k] \\ u[k] \end{bmatrix} + \begin{bmatrix} 0 \\ K_{id} r \end{bmatrix} \tag{12.41}$$

Se observa que, para la entrada escalón, los vectores $x[k]$, $u[k]$ y $y[k]$ tienden a ser valores constantes. Por lo tanto, se obtiene la ecuación (12.42) en estado permanente para el vector integral, la cual confirma un error nulo permanente respecto a la referencia.

$$\begin{aligned} v[\infty] &= v[\infty] + r - y[\infty] \\ y[\infty] &= r \end{aligned} \tag{12.42}$$

De igual manera, el sistema de la ecuación (12.41) en estado permanente se convierte en:

$$\begin{bmatrix} x[\infty] \\ u[\infty] \end{bmatrix} = \begin{bmatrix} A_d & B_d \\ K_d - K_d A_d - K_{id} C_d A_d & I - K_d B_d - K_{id} C_d B_d \end{bmatrix} \begin{bmatrix} x[\infty] \\ u[\infty] \end{bmatrix} + \begin{bmatrix} 0 \\ K_{id} r \end{bmatrix} \tag{12.43}$$

Se define la diferencia entre el estado actual y el estado permanente como un nuevo vector de error de estados y se construye su sistema al restar (12.41) y (12.43).

$$\begin{aligned} x_e[k] &= x[k] - x[\infty] \\ u_e[k] &= u[k] - u[\infty] \\ \begin{bmatrix} x_e[k+1] \\ u_e[k+1] \end{bmatrix} &= \begin{bmatrix} A_d & B_d \\ K_d - K_d A_d - K_{id} C_d A_d & I - K_d B_d - K_{id} C_d B_d \end{bmatrix} \begin{bmatrix} x_e[k] \\ u_e[k] \end{bmatrix} \end{aligned} \tag{12.44}$$

La dinámica del sistema aumentado queda determinada finalmente por los valores característicos de la matriz de estado de esta última ecuación, la cual está en función de los valores del sistema discreto y las matrices de ganancias K_d y K_{id}, puede reescribirse de la siguiente manera:

$$\begin{bmatrix} x_e[k+1] \\ u_e[k+1] \end{bmatrix} = \begin{bmatrix} A_d & B_d \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_e[k] \\ u_e[k] \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} w[k] \tag{12.45}$$

$$w[k] = -\begin{bmatrix} K_d - K_d A_d - K_{id} C_d A_d & I - K_d B_d - K_{id} C_d B_d \end{bmatrix} \begin{bmatrix} x_e[k] \\ u_e[k] \end{bmatrix} \tag{12.46}$$

Este nuevo sistema aumentado discreto se puede simplificar como un sistema realimentado, como se muestra en las ecuaciones (12.47) y (12.48), al definir apropiadamente las nuevas matrices aumentadas.

$$\xi[k+1] = A_{ad}\xi[k] + B_{ad}w[k] \tag{12.47}$$
$$w[k] = -K_{ad}\xi[k] \tag{12.48}$$

$$\xi[k] = \begin{bmatrix} x_e[k] \\ u_e[k] \end{bmatrix} \quad A_{ad} = \begin{bmatrix} A_d & B_d \\ 0 & 0 \end{bmatrix} \quad B_{ad} = \begin{bmatrix} 0 \\ I \end{bmatrix}$$
$$K_{ad} = \begin{bmatrix} K_d - K_d A_d - K_{id} C_d A_d & I - K_d B_d - K_{id} C_d B_d \end{bmatrix} \tag{12.49}$$

Con los polos deseados del sistema previamente definidos, simplemente se debe de seguir el procedimiento visto en la sección 12.5 para realizar su ubicación en este nuevo sistema aumentado discreto y así obtener los valores de la matriz K_{ad} que logran dicha dinámica impuesta. Es importante recordar que si el sistema original es observable y controlable, ambas propiedades se mantienen en este nuevo sistema aumentado discreto. Finalmente, para obtener los valores de las matrices de ganancias K_d y K_{id} necesarias para la ley de control, estas se deben de despejar de la ecuación (12.49) tal que:

$$\begin{bmatrix} K_d & K_{id} \end{bmatrix} = \begin{bmatrix} K_{ad} + \begin{bmatrix} 0 & I \end{bmatrix} \end{bmatrix} \begin{bmatrix} A_d - I & B_d \\ C_d A_d & C_d B_d \end{bmatrix}^{-1} \tag{12.50}$$

Para poder implementar este método en el prototipo armado primero se debe de discretizar el modelo que aproxima su comportamiento. Al igual que los controladores de variables de estado anteriores, para este controlador se utiliza el modelo linealizado (12.10). El periodo de muestreo es de $T_s = 16[ms]$, que es el tiempo que le toma al programa en realizar un ciclo completo de ejecución, se debe discretizar utilizando este periodo. Para ello, existen varios métodos de conversión con los que se pueden obtener representaciones equivalentes discretas a partir de una función continua, por ejemplo: la regla del adelanto, la regla del atraso, la regla trapezoidal, la retención de orden cero (ZOH), la retención de orden triangular (THE), por mencionar algunas. Matlab cuenta con el comando *sysd = c2d(sys,Ts,method)* para discretizar sistemas, tanto funciones de transferencia como en variables de estado. El método que utiliza por defecto es el de retención de orden cero, pero es posible cambiarlo por otro distinto. Se utilizó esta función para obtener la representación discreta del sistema linealizado de la planta mostrado en (12.51).

$$x[k+1] = \begin{bmatrix} 1 & 0.016 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.016 \\ 0 & 0 & 0 & 1 \end{bmatrix} x[k] + \begin{bmatrix} 0.0002514 & 0.0002258 \\ 0.03143 & 0.02823 \\ -0.003984 & 0.003846 \\ -0.498024 & 0.480707 \end{bmatrix} u[k] \tag{12.51}$$
$$y[k] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x[k] + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} u[k]$$

Con esta nueva representación se construye el sistema aumentado de acuerdo a la expresión (12.47), tal que:

$$\xi[k+1] = \begin{bmatrix} 1 & 0.016 & 0 & 0 & 0.0002514 & 0.0002258 \\ 0 & 1 & 0 & 0 & 0.03143 & 0.02823 \\ 0 & 0 & 1 & 0.016 & -0.003984 & 0.003846 \\ 0 & 0 & 0 & 1 & -0.498024 & 0.480707 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \xi[k] + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} w[k] \tag{12.52}$$

El comando *place* de Matlab se aplicó sobre el sistema aumentado (12.52) para obtener la matriz K_{ad}. Los polos deseados utilizados son los mismos que los propuestos en el controlador continuo de retroalimentación de salidas con acción integral de la sección 12.5, sin embargo, estos polos se deben de representar ahora con sus equivalentes discretos utilizando el mismo tiempo de muestreo que la planta. Una vez obtenida la matriz aumentada de ganancias K_{ad} (mostrada en la ecuación (12.53)), se utiliza la expresión (12.50) para obtener las matrices K_d y K_{id} necesarias para realimentar el controlador, tal como lo muestra la ley de control (12.37). Los resultados de estas operaciones se muestran en la Tabla [12.5].

$$K_{ad} = \begin{bmatrix} 0.1893 & 0.2282 & 0.0478 & -0.0259 & -0.6165 & -0.1519 \\ 0.2183 & 0.2789 & 0.0980 & 0.0106 & 0.1530 & -0.8930 \end{bmatrix} \tag{12.53}$$

Sistema	Polos discretos	Matriz K_d	Matriz K_{id}
Linealizado discreto	$P_1 = 0.9841 + 0.0093i$ $P_2 = 0.9841 - 0.0093$ $P_3 = 0.9231$ $P_4 = 0.9216$ $P_5 = 0.8483 + 0.0807i$ $P_6 = 0.8483 - 0.0807i$	$K = \begin{bmatrix} 14.0704 & 3.6122 & -1.6692 & -0.5219 \\ 17.2131 & 4.2089 & 0.5615 & -0.0381 \end{bmatrix}$	$K_{id} = \begin{bmatrix} 0.1893 & 0.0478 \\ 0.2183 & 0.0980 \end{bmatrix}$

Tabla 12.5: Constantes resultantes para el controlador RS con acción integral discreto

Los estados de esta propuesta discreta sufren el mismo problema que en la continua: no pueden ser medidos directamente. Entonces, tal como se muestra en la Figura [12.19], no se está exento de necesitar de un observador para obtener los estados necesarios para la realimentación. El observador (12.30) calculado anteriormente es suficiente para este propósito, pero es necesario utilizar su representación discreta. Para conseguirla se puede utilizar el mismo procedimiento de la Sección (12.4) donde se calculó, pero esta vez se debe utilizar el sistema linealizado discreto (12.51) en lugar del continuo además de discretizar los polos que definen su dinámica. Otra opción más simple es discretizar directamente el sistema de lazo cerrado que define al observador con el tiempo de muestreo deseado. Al utilizar la función *c2d* de Matlab el observador de lazo cerrado discreto queda representado de la siguiente manera:

$$\hat{x}[k+1] = \begin{bmatrix} 0.3015 & 0.0160 & 0.0002 & 0 \\ -5.0888 & 1 & 0.0019 & 0 \\ 0.0002 & 0 & 0.2943 & 0.0160 \\ 0.0019 & 0 & -5.1546 & 1 \end{bmatrix} \hat{x}[k] + \begin{bmatrix} 0.6985 & -0.0002 & 0.0003 & 0.0002 \\ 5.0888 & -0.0019 & 0.0314 & 0.0282 \\ -0.0002 & 0.7057 & -0.0040 & 0.0038 \\ -0.0019 & 5.1546 & -0.4980 & 0.4807 \end{bmatrix} \begin{bmatrix} y[k] \\ u[k] \end{bmatrix} \tag{12.54}$$

$$\hat{x}[k] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \hat{x}[k] + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y[k] \\ u[k] \end{bmatrix}$$

Para comprobar que los valores obtenidos son correctos y el controlador discreto es suficiente para controlar la planta, se construyó el diagrama de bloques de la Figura [12.20] en Simulink, donde se simuló la respuesta de esta propuesta discreta en el modelo no lineal ante condiciones iniciales fuera del equilibrio para comprobar la estabilidad y luego un cambio de referencia con el objetivo de verificar la regulación. Los resultados se encuentran en las gráficas de la Figura [12.21] representados con las cantidades reales, análogas a las que se verían en el modelo físico.

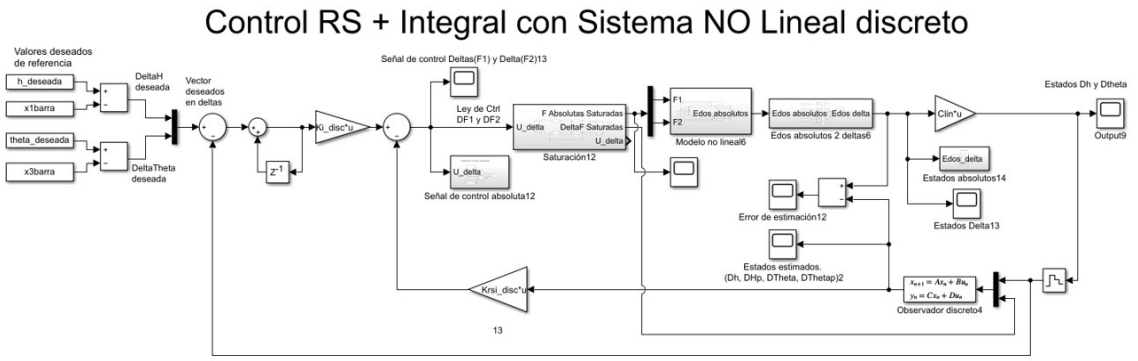
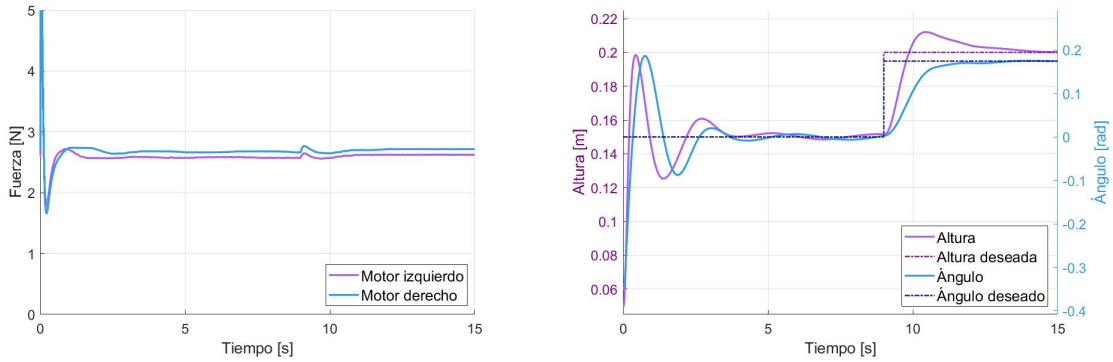


Figura 12.20: *Diagrama de bloques (Simulink): Control RS servo discreto con sistema no lineal*



(a) Modelo no lineal: Señal de control

(b) Modelo no lineal: Variables a controlar

Figura 12.21: *Simulación del control RS servo discreto con el modelo no lineal*

La dinámica es bastante parecida a su contra parte continua, aunque presenta un comportamiento más suave y menos agresiva, tanto en la señal de control como en el desempeño, sin llegar a los límites físicos del sistema. Además, ambas variables a controlar alcanzan un error en estado permanente nulo.

13. Implementación de los controladores

Como ya se mencionó, las variables de control (entradas de la planta) son las fuerzas de sustentación que proporcionan los motores y las variables a controlar (salidas de la planta) son el ángulo de inclinación y la altura del móvil del prototipo: la tarea de medir estas salidas la tienen los sensores seleccionados. Por otro lado, los motores y propelas son los encargados de traducir la señal de control como un fenómeno mecánico que ocasione un cambio. El microcontrolador es el responsable de la interpretación y cálculos donde se involucran todas estas variables junto con las funciones del algoritmo de control y la comunicación, sin embargo, este funciona utilizando voltajes y corrientes por sus diferentes componentes, por lo que es imperativo poder representar las variables de entrada y salida de nuestro sistema de esta forma y así poder trabajar con ellas. Como se mostró en la Sección 6.1.4, los sensores ya cuentan con el transductor y el acondicionamiento necesario y por ello son capaces de realizar esto de forma efectiva con las variables de salida. De igual forma, en la Sección 6.1.2, se mostró la señal PWM mediante la cual se interpretan las entradas.

Existe una relación matemática entre la variable física y la señal eléctrica que la representa la cual no siempre es directa. Esta relación puede describirse de muchas maneras, por ejemplo, con una ecuación matemática conocida como *ecuación característica de comportamiento* o mediante gráficas características. En el caso de los sensores es muy común encontrar estas relaciones en la hoja de datos (Datasheet) que brindan los fabricantes para que las medidas sean interpretadas de forma correcta; uno de los puntos principales de decisión sobre el modelo de los sensores en este proyecto fue la linealidad de esta relación. El caso de los actuadores es un tanto diferente.

13.1. Caracterización de los motores

La relación entre la señal eléctrica proporcionada por el microcontrolador y los actuadores no es fácil de conocer. En la bibliografía e información disponible de estos elementos no se encuentra la ecuación característica debido a que esta relación depende ampliamente de la forma del sistema, los componentes utilizados y sus propias restricciones. Por lo tanto, cada sistema, aunque sea similar, puede tener una relación de voltaje-empuje diferente. Por ello, en ocasiones esta relación se debe obtener experimentalmente; a este proceso se le conoce como *caracterización*.

El microcontrolador del prototipo utiliza una señal PWM con su frecuencia fija en 500[Hz], es decir que solo se necesita caracterizar la relación entre el valor entero de 10 bits del Duty y la fuerza de sustentación generada en Newtons. Para lograr esto se montó un experimento como se muestra en la Figura [13.1].

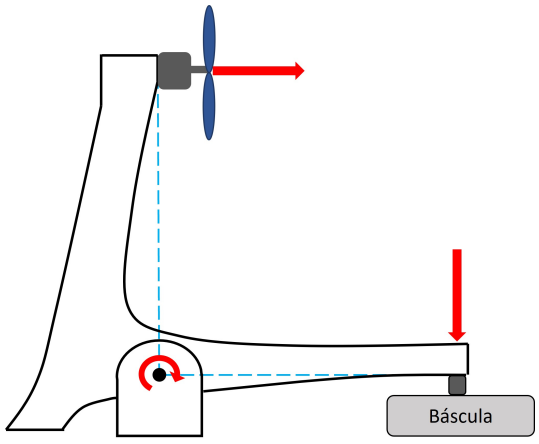


Figura 13.1: *Planteamiento del experimento para la caracterización de los motores*

Se diseñó y fabrico un soporte tipo escuadra con libertad de rotar alrededor de un perno central. En un

extremo se puede colocar el conjunto motor-propela, mientras que en el otro se encuentra una báscula capaz de medir la fuerza mediante una celda de carga. Cuando el motor gira y genera una fuerza en un extremo, se produce un par sobre el perno que a su vez desencadena otra fuerza con un desfase de 90°. Debido a que la distancia entre el perno y el motor es igual a la que hay entre el perno y el punto de contacto con la báscula, la fuerza que se transmite a la celda de carga es aproximadamente la misma que genera el motor.

Mediante esta configuración se hizo funcionar el motor a distintas velocidades utilizando distintos valores de Duty del PWM (siempre dentro del rango de [500 - 1000] para mantener el tiempo en alto entre 1[ms] y 2[ms]). De esta manera se obtuvo una relación entre el valor de PWM y la fuerza producida como se muestra en la Figura [13.2]. La medición varía debido a factores de precisión, exactitud e histéresis de la celda de carga. Aunque se observa que el comportamiento no es completamente lineal, con propósitos de simplicidad, ahorro de la carga computacional del microcontrolador y obtención simple de la relación inversa, se decidió aproximar esta relación mediante una recta.

El coeficiente de determinación R^2 es una herramienta que nos ayuda a conocer que tan bien se puede predecir un resultado, evalúa la calidad del modelo para replicar estos mismos, y la proporción de variación de los resultados que puede explicarse por el modelo. Se calcula por medio de la resta *uno* menos la *varianza residual* sobre la *varianza*. La interpretación que se obtiene es que mientras mas cercano sea su valor a la unidad se considera que las estimaciones del modelo se ajustan bastante bien a la variable real[103]. El coeficiente R^2 resultante es de 0.98, entonces, la regresión lineal se considera muy buena para esta aplicación. La expresión (13.1) es la ecuación característica PWM-Fuerza resultante de este experimento; su inversa nos permite traducir la señal de control a un valor de PWM listo para enviar a los motores.

$$F_m = 0.018465(pwm) - 10.0694 \tag{13.1}$$

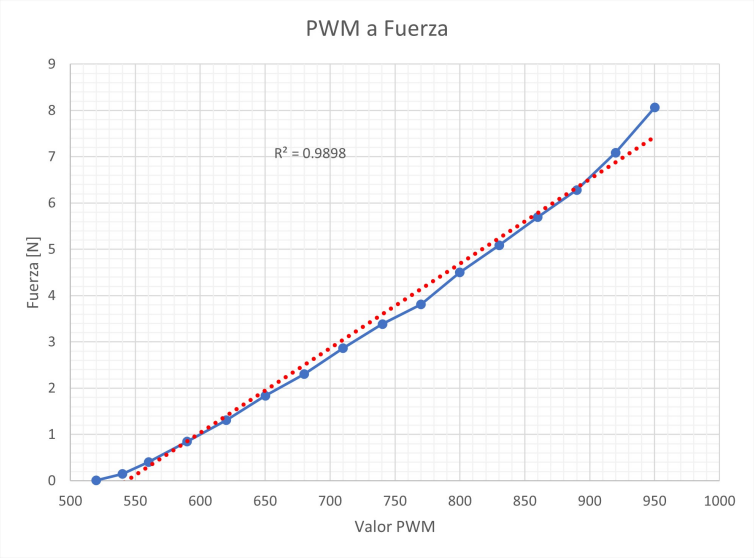


Figura 13.2: *Gráfica de la caracterización experimental del empuje generado por cada motor*

13.2. Programación del microcontrolador

La programación del microcontrolador es una de las últimas etapas del desarrollo del sistema de control; una vez realizada, se pueden comenzar a hacer pruebas sobre el sistema y análisis de resultados para validar el funcionamiento de los modelos de la planta y controlador obtenidos simulados. Es un proceso fundamental que requiere de un profundo entendimiento del problema y conocer el funcionamiento de todo el sistema en donde se encuentra implementado, por ello, es específico de la aplicación. Es aquí donde se jerarquiza el

orden de ejecución de instrucciones y se procesan los datos de las entradas para obtener las salidas. Ambas son consecuencia de los requerimientos y especificaciones dispuestas en la etapa de diseño.

El controlador puede ser programado en la computadora y utilizar el microcontrolador solamente para sensar y manipular los actuadores, o completamente en el microcontrolador del dispositivo. A pesar de que una computadora tenga una frecuencia de trabajo mucho mayor, manejar una conexión inalámbrica mediante Wifi implica que el proceso de enviar datos y recibir la señal de control genere una latencia y el ciclo pueda tomar un tiempo mayor que si se ejecuta en el mismo dispositivo. Aunque la segunda opción supone una mayor carga computacional para el MCU, se decidió implementarlo completamente a bordo del móvil, esto significa que el procesador ubicado en el microcontrolador se dedica al cálculo de las operaciones necesarias para hacer el control, lectura de datos, comunicación y otras funciones fundamentales.

Debido a la limitada capacidad de procesamiento y memoria, el lenguaje que se usa para la generación de código suele ser diferente en un microcontrolador al de una computadora; muchas funciones complejas son limitadas para hacer más eficiente el código y consumir una menor cantidad de recursos. Para mejorar en este aspecto es buena práctica contemplar algunas recomendaciones como simplificarlo lo más posible de forma que sea entendible, tener una buena estructura y organización, además, para evitar la repetición de código es buena idea hacer uso de funciones, clases e instancias.

Para este proyecto se utilizó el lenguaje de programación *MicroPython*[104], que es una implementación delgada y eficiente de *Python 3* e incluye un subconjunto de las librarías estándar de python comprimidas y optimizadas para correr en ambientes restringidos. Esta elección es congruente con el lenguaje usado en la interfaz, lo cual facilita su interacción.

La implementación se compone principalmente de tres programas. El primero contiene solamente los valores puntuales de las variables, parámetros y ganancias tanto del modelado del sistema, la linealización y los valores continuos y discretos de los controladores. El propósito de esto es que estas variables puedan ser llamadas y modificadas en cualquier momento por cualquier función o programa sin correr el riesgo de perder los valores originales.

El segundo programa contiene las funciones y algoritmos que involucran el funcionamiento del controlador. Con la intención de aprovechar la *Programación Orientada a Objetos (POO)*, se definieron clases que contienen todos los *Atributos* y *Métodos* necesarios.

La clase para sensar realiza la lectura directa de los datos de la IMU y obtiene el ángulo mediante un filtro complementario basado en los datos del giroscopio y el acelerómetro, como se explicó en la IMU en la Sección 6.1.4. Además recopila e interpreta la señal del infrarrojo y calcula la distancia mediante la ecuación característica de dicho sensor. Finalmente, filtra ambas lecturas mediante un filtro de conocido como *Filtro* α-β-γ para reducir el ruido que se pueda producir. Los resultados de esta etapa se muestran en la Figura [13.3]; el algoritmo se realizó de acuerdo con lo que dice Alex Becker en su articulo sobre dicho filtro.[105][106]

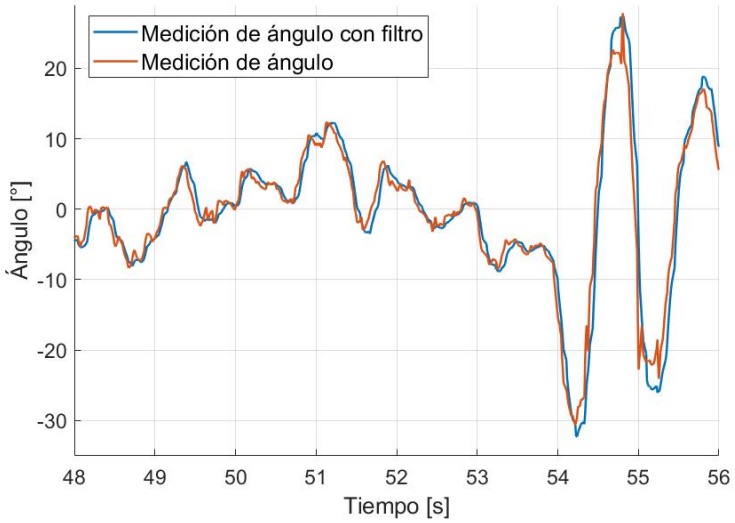


Figura 13.3: *Resultados del filtrado mediante un filtro* α-β-γ

Las clases del PID y del control RS servo contienen el algoritmo obtenido en la sección discreta de cada uno de los controladores, estiman los estados y convierten las variables en deltas de ser necesario; todo mediante las variables guardadas en el primer programa. Este segundo programa también contiene algunas funciones relacionadas con los elementos extra en los diagramas de bloques de los controladores, por ejemplo: la conversión entre el Duty del PWM y la fuerza de sustentación y el bloque para combinar los efectos de los controles PID desacoplados. Actualmente se utiliza un programa secundario donde se definieron las funciones necesarias para realizar las operaciones matriciales necesarias para los cálculos, ya que este lenguaje reducido no permitía el uso de librerías comúnmente utilizadas en Python como NumPy.

El tercer programa o *main* es aquel que inicia y corre de forma cíclica cada vez que se enciende el prototipo. Se encarga de definir los objetos mediante las variables y clases de los dos primeros programas, ejecutar el sensado y controladores, codificar las variables y comunicarse con la interfaz para ejecutar comandos específicos o enviar los valores listos para graficar mediante Wifi. El diagrama de flujo de este último programa, junto con su código, se encuentra en el apéndice I.

14. Resultados del control

Ya que se tienen las distintas propuestas de control programadas en el dispositivo, las ganancias de cada uno bien definidas y la forma de enviar instrucciones inalámbricamente, solo resta probar el funcionamiento. Se realizaron varios experimentos con el propósito de verificar el cumplimiento de los objetivos particulares de toda la sección de *Diseño e implementación del control*.

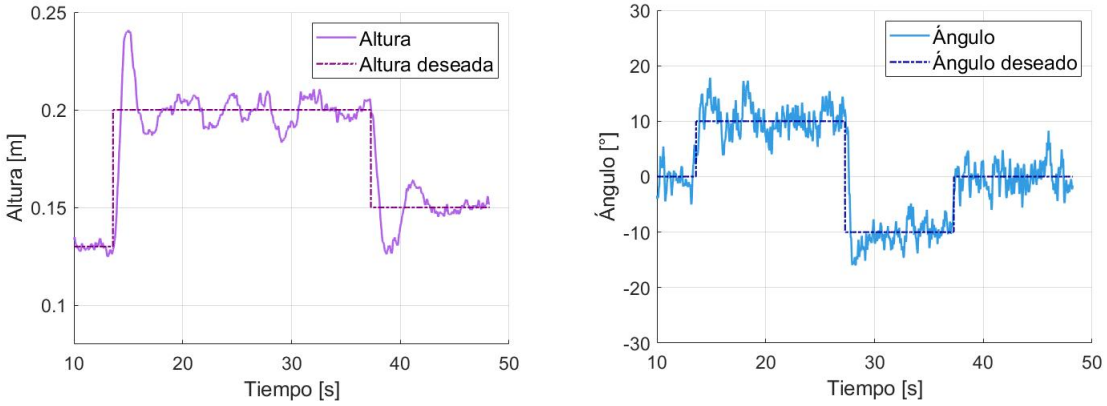
14.1. Resultados del PID

Para evaluar el desempeño de la primera propuesta de control en el prototipo, se implementaron dos controladores tipo PID discreto con la acción derivativa directamente en la señal de salida de cada subsistema como se propone en la Sección 11.5. Se debe recordar que se plantearon dos subsistemas desacoplados simultáneos para describir el comportamiento de la altura y el ángulo. Las ganancias K_c, τ_i y τ_d de ambos controladores fueron obtenidas utilizando la técnica de sintonización de Ziegler-Nichols de lazo cerrado.

Se realizaron dos pruebas, la primera tiene como propósito evaluar la eficacia de los controladores para responder a cambios de referencia; se realizaron tres de ellos de forma continua en la misma prueba. El primero consiste en un incremento simultáneo en los valores deseados, tanto de la altura como del ángulo,

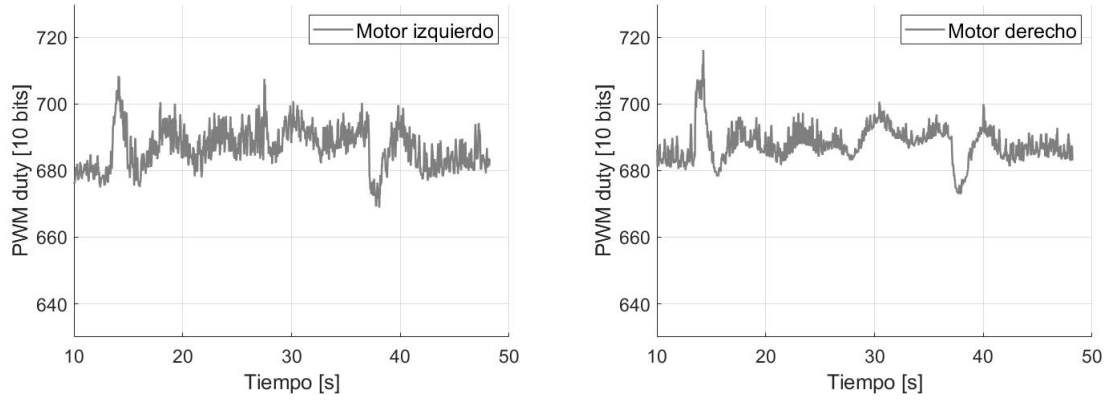
y ocurre aproximadamente en el segundo 14[s]. El segundo cambio de referencia, que sucede a los 27[s], es únicamente para el ángulo que pasa de un valor positivo a uno negativo. Por último, en el segundo 37[s], el sistema debe llegar a su punto de equilibrio en ambas salidas. Los datos obtenidos de esta prueba se recuperaron del archivo generado por la interfaz gráfica y se muestran en la Figura [14.1]. Para representar los datos con mayor claridad se optó por mostrar cada variable de control y variable a controlar en una gráfica por separado.

Esta propuesta consiguió seguir la referencia de forma exitosa y su desempeño se caracterizó por sus importantes sobrepasos (principalmente en la altura) y su rápida respuesta. Se percibe que en el momento de mover la referencia del ángulo sin mover la altitud, los controladores consiguen llevar al ángulo a su respectivo valor deseado sin afectar a la altura significativamente. Además, la señal de control siempre mantiene valores conservadores alrededor del punto de equilibrio sin incrementos repentinos y no presenta la patada derivativa.



(a) Altura medida

(b) Ángulo medido



(c) Señal de control del motor izquierdo

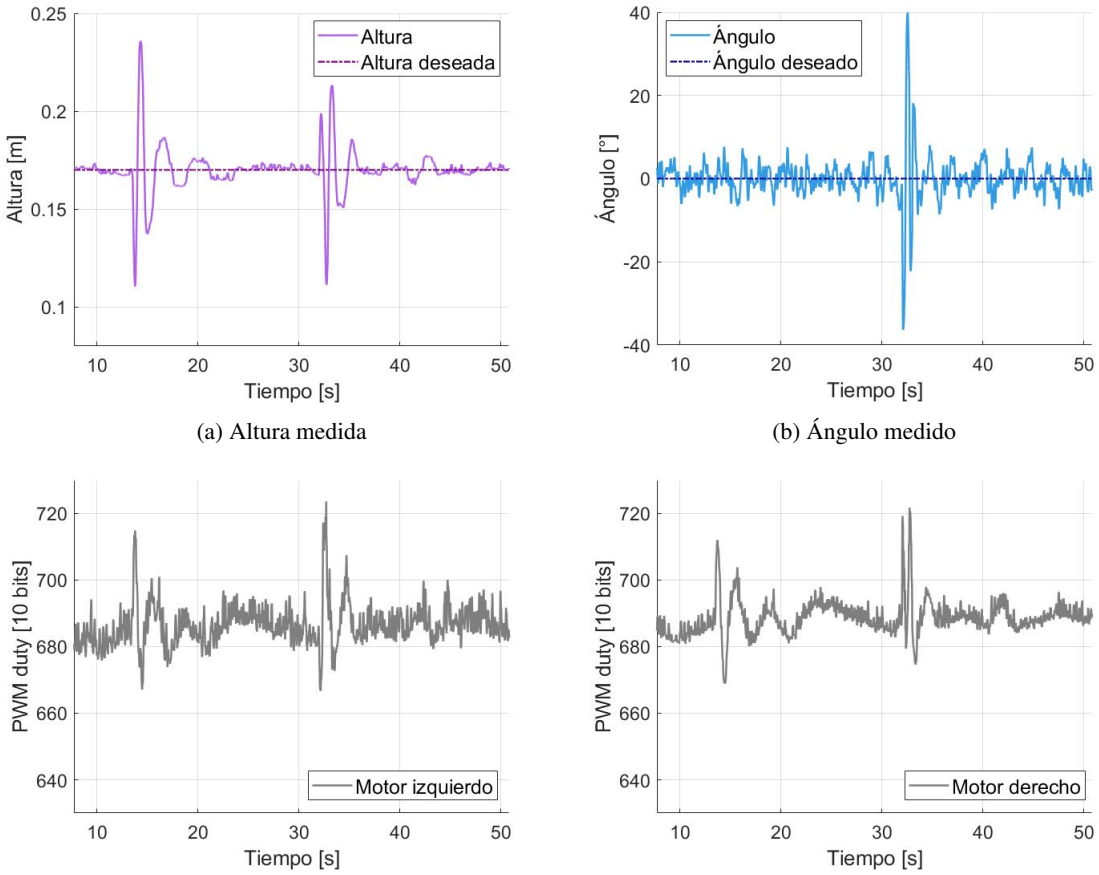
(d) Señal de control del motor derecho

Figura 14.1: *Respuesta ante cambios de referencia con controladores tipo PID discreto en el modelo real*

La segunda prueba tiene como objetivo evaluar la forma en la que el controlador rechaza las perturbaciones. Para ello se mantuvieron las referencias constantes durante todo el experimento y se infundieron dos alteraciones importantes. La primera fue ejercer una fuerza momentánea hacia abajo directamente sobre el centro de la plataforma móvil con cuidado de afectar únicamente a la altura. La segunda fue una fuerza con dirección ascendente en el extremo izquierdo del balancín para afectar su orientación. Los resultados de esta prueba se encuentran en la Figura [14.2].

Esta prueba reveló que, a diferencia de cuando hay un cambio de referencia, el sistema tarda más en re-

cuperarse tras una perturbación, sobretodo si se altera la posición vertical. También, al intentar corregir la alteración angular, se descompensó la altura, por lo que se vio en la necesidad de corregir ambas variables. A pesar de esto, en ambos casos, el prototipo logró rechazar las perturbaciones con éxito. Es importante señalar que con este controlador cada prueba puede tener una respuesta inicial muy violenta si se inicia lejos de los valores deseados. Por ello, antes de cada prueba se movió manualmente el dispositivo hasta una posición próxima a ellos.



(a) Altura medida

(b) Ángulo medido

(c) Señal de control del motor izquierdo

(d) Señal de control del motor derecho

Figura 14.2: *Respuesta ante perturbaciones con controladores tipo PID discreto en el modelo real*

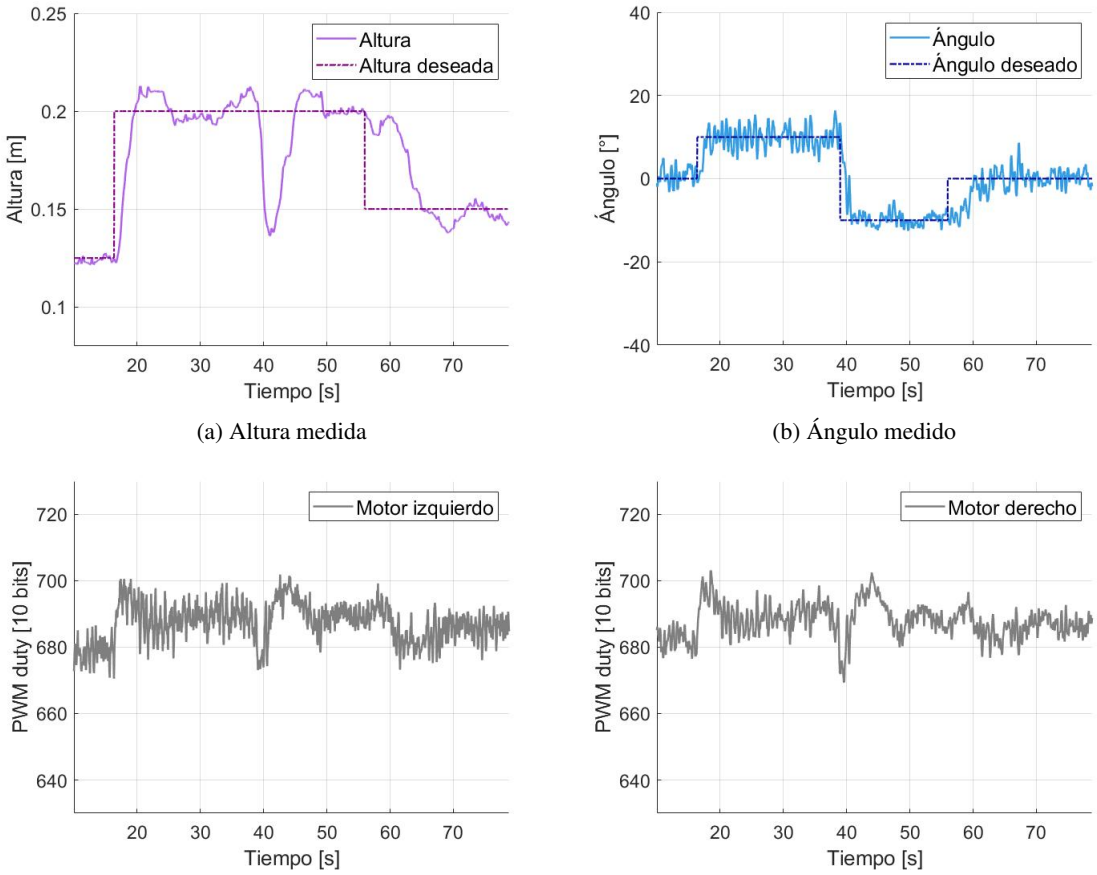
14.2. Resultados del control RS + integral

El mismo procedimiento se utilizó para evaluar la eficacia y desempeño de la segunda propuesta de control. Para esto, se programó el controlador discreto de retroalimentación de salidas con acción integral obtenido en la Sección 12.6. Las matrices de ganancias K_d y K_{id}, junto con el observador discreto de orden completo necesario para poder realimentar los estados, corresponden a los mismos obtenidos en dicha sección.

La primer prueba nuevamente evaluó la capacidad del sistema para seguir un cambio de referencia tipo escalón. Se realizaron tres cambios distintos en los segundos 16[s], 39[s] y 56[s] del mismo experimento. El primero, consiste en un cambio simultáneo de los valores de ángulo y altura deseada, el segundo cambio únicamente es sobre la inclinación y por último se modifican ambas referencias para que el dispositivo llegue a su punto de equilibrio. Las gráficas resultantes de esta prueba se muestran en la Figura [14.3].

En general, las señales de control obtenidas son más suaves y no se alejan mucho del punto de equilibrio. El desempeño es más lento pero mejor controlado; esto se nota en el tamaño del sobrepaso en ambas salidas

y el tiempo de asentamiento necesario para llegar a su nueva referencia. Una de las diferencias principales, comparado con los controladores tipo PID desacoplados, es que ante un cambio de referencia en una sola variable, la señal de control resultante afecta fuertemente a ambas salidas. Este cambio se muestra cerca del segundo 39[s] en la Figura [14.3] (a), donde la altura sufre una descompensación al intentar regular el ángulo, pero después de esto se recupera.



(a) Altura medida

(b) Ángulo medido

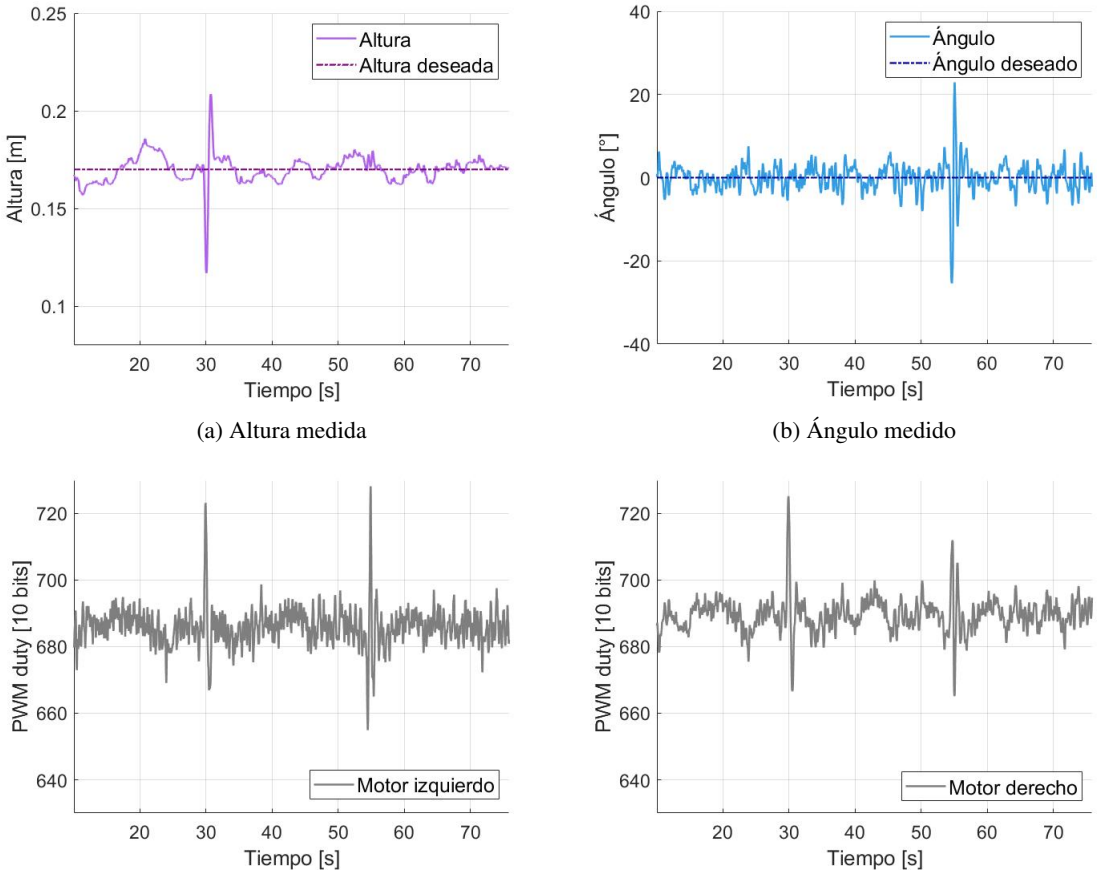
(c) Señal de control del motor izquierdo

(d) Señal de control del motor derecho

Figura 14.3: *Respuesta ante cambios de referencia con controlador discreto RS + integral en el modelo real*

Para la segunda prueba en esta propuesta de control se siguió el mismo procedimiento. Con el propósito de observar la habilidad del controlador para rechazar perturbaciones, se ejerció un cambio súbito a cada variable a controlar durante el mismo experimento. Para perturbar al sistema, al igual que con los PID, al balancín se le ejerció una fuerza vertical hacia abajo por la parte central de manera que no rotara. Después, se consiguió un cambio abrupto en la inclinación, aplicando otra fuerza en dirección ascendente desde el extremo inferior derecho del balancín. El desempeño de esta prueba se encuentra en la Figura [14.4].

Este sistema resultó efectivo para rechazar perturbaciones rápidamente. La señal de control se dispara en picos grandes y de corta duración para compensar estas discrepancias. También, a diferencia del control PID, se nota que, al perturbar el ángulo, la descompensación de la altura es mínima.



(a) Altura medida

(b) Ángulo medido

(c) Señal de control del motor izquierdo

(d) Señal de control del motor derecho

Figura 14.4: *Respuesta ante perturbaciones con controlador discreto RS + integral en el modelo real*

Una característica que se presentó durante las pruebas de ambas propuestas de control es una tendencia a la inestabilidad cuando el sistema se aleja mucho de sus valores de equilibrio, en especial, cuando esto ocurre de manera súbita. Por ello, se evitaron las perturbaciones demasiado violentas durante los experimentos para no comprometer la integridad del prototipo. Otra consecuencia de esto es que, si al iniciar una prueba el dispositivo está muy alejado de sus valores deseados, las primeras señales de control pueden ser demasiado agresivas y tender de nuevo a la inestabilidad. Se recomienda iniciar con precaución cada experimento. El controlador RS + integral es, en particular, más robusto ante estos efectos.

El sistema también presentó vibraciones que añaden ruido indeseado a las lecturas de los sensores. Este efecto se puede apreciar en la Figura [14.4] (a) y (b) de cualquier prueba dentro de esta misma sección. Ambas lecturas cuentan con un filtro de Kalman independiente para mejorar la adquisición de datos. Debido al diseño particular de la IMU, estas vibraciones la afectan de mayor manera y compensarlo con el filtro es buena opción hasta cierto punto, ya que al aumentar los valores de su matriz de pesos, el filtro se vuelve más lento para reflejar cambios en la variable, lo cual retrasa la adquisición de datos afectando directamente al controlador y se convierte en un sistema inestable. Es por ello que se optó por ganancias que, aunque no eliminan por completo el ruido, permiten reflejar los cambios rápidamente.

Se realizó un video para mostrar de mejor forma todos los resultados, si desea consultarlo la liga se encuentra en la sección V de apéndices.

Parte IV

Conclusiones

Con el propósito de cumplir los dos objetivos generales de diseño y control, este trabajo se dividió en dos secciones principales. La primera permitió desarrollar de forma metódica un prototipo VTOL funcional capaz de variar su posición con dos grados de libertad: altura e inclinación, basado en las características necesarias de utilidad, seguridad y practicidad de un banco de pruebas de control experimental, mientras que la segunda brinda una mayor profundización y delimitación sobre las técnicas de control con las que funciona y mejora su entendimiento. Cada una cuenta con sus propios objetivos particulares y resultados individuales derivados de pruebas empíricas e implementación práctica.

La forma, funcionamiento y componentes del prototipo, son resultado de la metodología empleada. Al ser un esfuerzo conjunto de un número distinto de integrantes en diferentes etapas de diseño, esta ayudó a mantener las metas claras, integrar las propuestas de forma efectiva y mantener coherencia durante todo el desarrollo, demostrando así la utilidad de este tipo de estrategias de trabajo y diseño. Se sugiere sobre todo prestar atención a la parte de investigación y planteamiento, ya que debido a suposiciones y falta de experiencia previa se tuvo la necesidad de replantear ciertos aspectos base con el proyecto ya avanzado. Profundizar en estos primeros pasos conduce a un menor número de replanteamientos para llegar a una propuesta funcional.

Los requisitos recolectados y procesados en la definición del sistema ayudaron a enfocar los esfuerzos en puntos críticos. Durante el diseño conceptual se llevó a cabo la identificación y el desglose de las funciones; esto permitió determinar una idea acerca del número y tipo de componentes que se necesitarían adquirir. Se considera que esta etapa, donde se habla de la generación, selección y agrupación de soluciones, fue la que implicó mayor trabajo, cuidado y tiempo de esta sección, ya que se tuvieron que considerar muchos factores como una mayor compatibilidad entre soluciones individuales, la información que se tenía acerca de cada alternativa dada la experiencia del equipo y datos investigados, características físicas, precio, la viabilidad para la producción de las piezas, etc.

Por su parte, la etapa correspondiente al detallado del dispositivo y construcción implicó muchos cambios, inclusive se replanteó la selección de los mismos conceptos, por lo cual se requirió de algunas iteraciones para mejorar el desempeño del prototipo. Tales cambios se debían a la falta de entendimiento de la solución propuesta, problemas en la implementación, disponibilidad de los componentes y otras dificultades no tomadas en cuenta a la hora de proponer soluciones. Sin embargo, gracias a ello, las ideas evolucionaron y se pudo mejorar en muchos sentidos respecto a las propuestas iniciales.

Tras finalizar la construcción del dispositivo fue posible evaluar el resultado mediante la lista de especificaciones y el árbol de objetivos, esto permitió determinar si las características finales del prototipo cumplían o no con lo planteado inicialmente como se observa en la Tabla [14.1]. Con base en ello, se determinó que fueron satisfechas casi en su totalidad. Aspectos como la estructura, la integración y distribución de los componentes, la funcionalidad e intuitivo manejo de la interfaz, el buen desempeño del control y la practicidad (proporcionada por la conexión inalámbrica y las dimensiones), destacan por sobre las demás características solicitadas.

Como parte del mismo proceso, no todos fueron satisfechos o se les dio la misma atención. Puntos de menor relevancia como el cableado, la elección de algunos componentes, la seguridad, el mantenimiento, el ruido, el peso e incluso el precio, son algunos que pudieron ser tratados de diferente forma y es algo en lo que aún se puede mejorar. Por ejemplo, incluir actuadores más pequeños permitiría reescalar el diseño a un dispositivo más liviano y menos costoso.

Resultados				
Especificación		**Referencia**	**Resultado**	**Evaluación**
Objetivos de control		2/2 puntos	2 puntos	Ambos controles implementados tienen un buen comportamiento, logran estabilizarse y regular las variables del dispositivo con un error en estado permanente cercano a cero.
Velocidad de procesamiento		16 [Mhz]	240 [Mhz]	El microcontrolador usado tiene muchas mejores prestaciones a las solicitadas teniendo como referencia las de una placa *Arduino UNO*.
Tiempo de respuesta a solicitud (con latencia)		100[ms]	16[ms]	Dependiendo de la instrucción, este intervalo de tiempo puede ser mayor o menor. Pero en general la respuesta logra ser mucho más rápida que el valor recomendado.
Tiempo de uso continuo		45 [min]	60[min]	El dispositivo soportó largas sesiones de trabajo (4 hr) con pruebas intermitentes de 20 minutos cada una sin inconveniente más que un ligero aumento de temperatura motores y drivers. La prueba más larga realizada fue de 1 hora sin notar repercusiones.
Dimensiones máximas del dispositivo		60x60x60 [cm]	48x58x30 [cm]	Las dimensiones resultantes son menores a las establecidas, pero se encuentran cerca del límite.
Masa total del dispositivo		5 [Kg]	4.725 [Kg]	La masa actual es un 5% menor a la establecida, por lo cual no hay problema en que una persona pueda trasladar el dispositivo.
Masa de la estructura móvil		0.7 [Kg]	0.54 [Kg]	La masa del *frame* tiene un valor de 0.54 [kg], lo cual es notablemente menor a lo esperado y la convierte en una característica destacada en el diseño.
Tiempo de armado		5 [hr]	4-6 [hr]	El montaje completo toma de 4 a 6 horas en completarse debido a los ajustes que se requieren, se considera un punto a mejorar, pero generalmente se encuentra dentro del límite impuesto.
Componentes intercambiables		0.7	0.66	Es un poco por debajo de lo esperado ya que la IMU, piezas únicas como el frame y partes de la base, son de difícil remplazo. Sin embargo, el resto de los componentes que podrían ser susceptibles a fallos pueden intercambiarse con relativa facilidad, lo que nos garantiza un tiempo de vida prolongado.
Respuesta a perturbaciones		3/3 puntos	3 puntos	Varía dependiendo del controlador asignado, pero en general rechaza perturbaciones externas ya sea cambios de referencia, variaciones de carga (leves, moderadas, momentáneas y constantes) además de ruido.
Puntaje de operación		7/10 puntos	8/10 puntos	La operación por medio de la GUI no cuenta con las características de personalización y portabilidad. Sin embargo, logra abarcar el resto de los rubros como cumplimiento de tareas, homogeneidad, funcionalidad, robustez, organización, estética, vocabulario y ayuda. El resultado es una buena experiencia de usuario en términos generales.
Distancia de operación		1-5 [m]	1-20 [m]	El dispositivo se controla totalmente de forma remota, y la distancia de operación únicamente está sujeta a las interferencias físicas del lugar, por esta misma razón el límite máximo puede variar. [136]
Protecciones eléctricas	OTP	70 [°C]	80 ± 5 [°C]	La fuente seleccionada posee la mayoría de las protecciones recomendadas, a excepción de la protección por sobrecorriente. Pero el resto de ellas nos permiten tener una seguridad suficiente respecto a la confiabilidad de la fuente.
	OPP	130-150%	105-125%	
	SCP	si	si	
	OCP	130-150%	NA	
	OVP	110-130%	115-135%	
Nivel de ruido máximo		68 [dB]	80 [dB]	Se identifica como un punto a mejorar ya que, durante la ejecución del control, el ruido de la rotación de las hélices es alto y puede llegar a ser molesto. No existe riesgo para la salud, pero no es recomendable la exposición prolongada a este nivel.

Tabla 14.1: *Evaluación del cumplimiento de las especificaciones del proyecto*

En lo que respecta con la sección de control, se identificaron dos aspectos que probaron ser clave: La obtención del modelo y la discretización. La importancia del modelado matemático (o identificación del modelo de ser el caso) consiste en ser el nexo entre los cálculos y la realidad. Cualquier modelo puede pasar por una etapa de diseño de control y arrojar resultados, pero, a pesar de que los parámetros de control estén bien calculados, si el modelo es erróneo no será funcional. Incluso cambios que pueden parecer insignificantes, como la aproximación del valor de fricción, tienen una gran repercusión al final, sobre todo en el controlador de variables de estado. Varios de los problemas que se presentaron durante esta etapa fueron resueltos al cambiar la propuesta del modelo y no al modificar los métodos de control. La linealización también fue parte importante de este proceso, aun cuando se llegó a considerar muy simplificado, probó ser eficaz.

La simulación con Matlab y Simulink fue una herramienta imprescindible para implementar los controladores, ya que permitió verificar los objetivos de estabilización y regulación sin comprometer el prototipo (esto es particularmente importante en sistemas mecánicos o muy sensibles a cambios repentinos en las entradas). Otras características del sistema implementadas como la solución de la patada derivativa del PID o la función para combinar las señales de los controladores desacoplados se obtuvieron gracias a este análisis previo. Claro que no es posible esperar que un modelo simple represente fielmente a un sistema físico complejo, por lo mismo, las simulaciones y los resultados no empatan completamente; las propuestas de control deben ser robustas para compensar este hecho.

En cuanto a la implementación de los controladores, su discretización fue un parteaguas para el desarrollo del proyecto. Al principio se asumió que la implementación de un controlador continuo con el hardware propuesto sería suficiente para controlar al sistema, pero durante numerosas pruebas se demostró lo contrario. El microcontrolador no es lo suficientemente rápido para entregar una salida que se asemeje a una señal de control continua. Se recomienda siempre implementar un controlador discreto; a menos que sea posible utilizar hardware analógico o que se cuente con un procesador digital suficientemente rápido, como en una tarjeta de adquisición de datos.

Respecto a los resultados obtenidos durante las pruebas en la planta, se aprecia que existen algunas circunstancias que provocan una respuesta violenta debido a la sensibilidad del controlador a la distancia respecto del punto de equilibrio, como iniciar alejado de este o recibir una perturbación que lo desvíe abruptamente; ambos controles tienen un mejor desempeño si se inician cerca de este. Se puede adjudicar a que los controles se diseñaron basados en un modelo que se obtuvo a partir de un punto de equilibrio y mientras más lejos se encuentre de él, este modelo pierde validez al provenir de un sistema no lineal.

Tanto el acercamiento clásico como el moderno demostraron ser efectivos en los objetivos de estabilización y regulación propuestas. En general, el PID demostró seguir la referencia en un menor tiempo pero con un mayor sobrepaso, mientras que el RS + integral mostró una mejor respuesta ante las perturbaciones. El desempeño de ambos controladores se puede mejorar a futuro si se toman en cuenta algunas observaciones. La primera es perfeccionar la adquisición de datos, para ello, se puede cambiar la instrumentación por mejores módulos o adicionar filtros más efectivos, ya sea analógicos o digitales; reducir la vibración del móvil también sería una gran mejora en este aspecto. La segunda es el tiempo de muestreo; esto se puede mejorar al implementar interrupciones internas en el MCU para ejecutar el algoritmo de control de forma constante en lugar de depender del tiempo que tarde la comunicación. Otra forma es reducir en general el tiempo de muestreo, lo cual se puede conseguir al optimizar el código, programarlo en un lenguaje de menor nivel y optimizado para ambientes restringidos (como C) o también programar el controlador en la computadora en lugar del MCU y ejecutarlo, junto con la comunicación, en paralelo con la interfaz.

Al considerar las características del prototipo resultante y los alcances de este trabajo, se considera que es una buena alternativa como sistema de control de laboratorio para estudiantes, profesores, investigadores e ingenieros que busquen aprovechar el dispositivo o bien afianzar los conocimientos prácticos, teóricos y el nexo entre estas áreas de conocimiento, orientado al control automático.

"Cuando un hombre sabe para donde va, el mundo entero se aparta para darle paso"

Bertrand Russell

"Cuando era más joven, restregué mi ignorancia en la cara de los demás. Me apalearon. Para cuando tenía cuarenta mi burdo instrumento había sido refinado hasta un punto de corte fino. Si escondes tu ignorancia de los demás, nadie te golpeará y nunca aprenderás."

Ray Bradbury

"Solíamos mirar al cielo y preguntarnos sobre nuestro lugar en las estrellas, ahora solo miramos hacia abajo y nos preocupamos sobre nuestro lugar en la tierra."

Cooper, Interstellar (2014)

Parte V

Apéndices

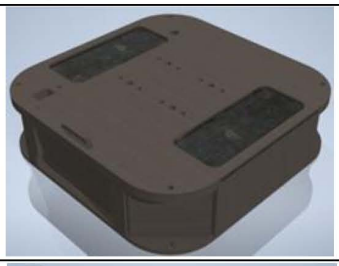
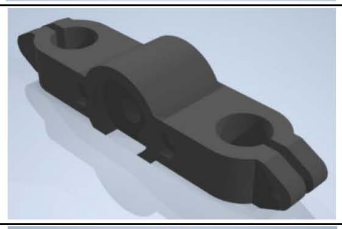
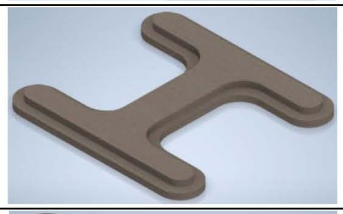
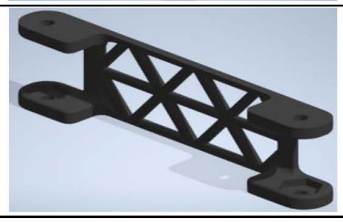
Debido a la cantidad de programas, archivos y documentos generados, en esta sección solo se muestran los más relevantes. En caso de querer consultar el resto de archivos (tales como los modelos CAD o el resto de los programas del Microcontrolador, Interfaz, Matlab, Simulink y Mathematica) los puede solicitar a los autores por correo electrónico:

- Víctor Octavio Romero Rivas: romerorivas.vo@gmail.com
- Saúl González Duardo: saul.gods97@gmail.com

También se realizó un video para mostrar el prototipo final y su funcionamiento. Puede consultarlo en: `https://youtu.be/T4mlffOgfb4`

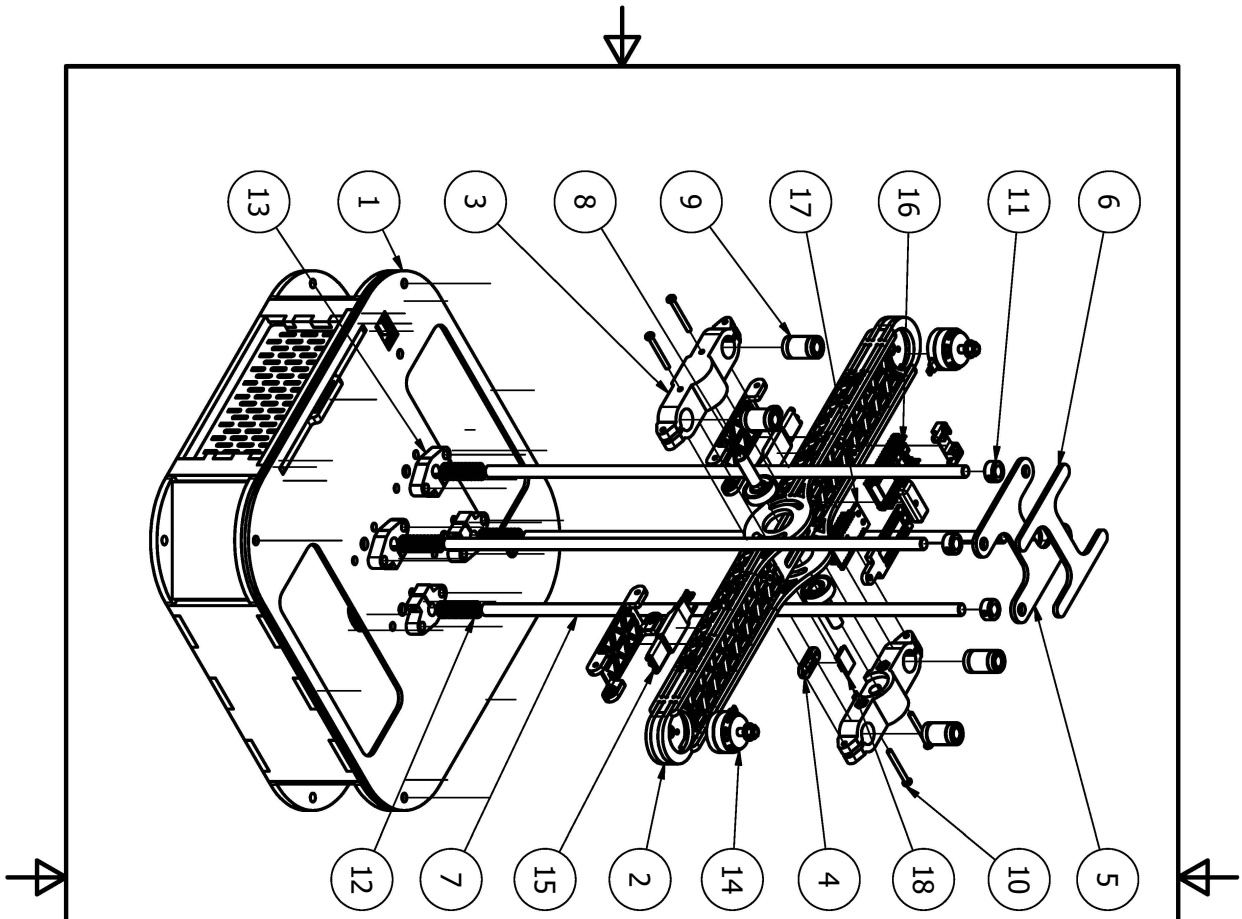
A. Lista de componentes

Componentes Electrónicos		
Microcontrolador		
Modelo	ESP32	
Descripción	Microcontrolador con comunicación wifi y bluetooth incluida. https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf	
Dimensiones	51x27x13 [mm]	
Cantidad	1	
IMU		
Modelo	BMI085 BOSCH	
Descripción	Unidad de medición inercial por comunicación I2C, con shield. https://pdf1.alldatasheet.com/datasheet-pdf/view/1132062/BOSCH/BMI085.html	
Dimensiones	43x20x10 [mm]	
Cantidad	1	
Sensor de distancia		
Modelo	GP2Y0E03	
Descripción	Sensor óptico (ToF) con rango de 4 a 52 con comunicación I2C y analógica. https://pdf1.alldatasheet.com/datasheet-pdf/view/1243994/SHARP/GP2Y0E03.html	
Dimensiones	17x11x5 [mm]	
Cantidad	1	
Motores Brushless		
Modelo	Racerstar BR2212 1400KV	
Descripción	Motor brushless BR2212 1400KV, corriente máxima 20A. https://9k.gg/BMGHH	
Dimensiones	28Dx40 [mm]	
Cantidad	2	
ESC		
Modelo	ESC Genérica	
Descripción	Driver controlador de velocidad para una corriente máxima de 30A, voltaje de entrada 2-4 celdas LiPo. https://uelectronics.com/wp-content/uploads/2020/10/30A-BLDC-ESC-Product-Manual.pdf	
Dimensiones	55x26x13 [mm]	
Cantidad	2	
Fuente		
Modelo	MEAN WELL – SE 600 12	
Descripción	Fuente de alimentación de 600W (12V - 50 A) cuenta con protección contra corto circuito, sobrecorriente, sobretensiones y sobrecalentamiento. https://pdf1.alldatasheet.com/datasheet-pdf/view/259427/MEANWELL/SE-600-12.html	
Dimensiones	247x127x64 [mm]	
Cantidad	1	

Estructura		
Postes		
Descripción	Varilla/barra lisa de acero rectificado.	
Dimensiones	8Dx400 [mm]	
Cantidad	4	
Base		
Descripción	Base de MDF para fuente con tres compartimentos y ranuras para ventilación.	
Dimensiones	296x296 [mm]	
Partes	6	
Balancin		
Descripción	Frame con ranuras para sujetar los motores ESC, sensores y microcontrolador, fabricado con ACP de 4.7mm.	
Dimensiones	420x38x50 [mm]	
Piezas	5	
Abrazaderas		
Descripción	Abrazaderas de PLA hechas en impresión 3D, sujetan los baleros lineales y la estructura del frame.	
Dimensiones	113x25x32 [mm]	
Cantidad	2	
Tapa		
Descripción	Cubierta superior de la estructura fabricada en MDF que sujeta y alinea los cuatro postes.	
Dimensiones	80x75x8	
Cantidad	1	
Sujetadores de PLA		
Descripción	Piezas de PLA para sujetar los componentes electrónicos al frame.	
Cantidad	8	

Elementos mecánicos		
Propelas		
Modelo	8045	
Descripción	Propela de plástico, dos palas con punta redondeada 8pulgadas de diámetro con 45 grados de inclinación. https://www.tutorialdedrones.com/helices-para-drones/	
Dimensiones	205x22x7 [mm]	
Cantidad	1 par	
Rodamiento lineal		
Modelo	LM8UU	
Descripción	Rodamiento lineal con diámetro interno de 8mm y carcasa de acero.	
Dimensiones	15Dx24 [mm]	
Cantidad	4	
Rodamiento		
Modelo	608-RS	
Descripción	Rodamiento rígido de bolas con diámetro interno de 8mm.	
Dimensiones	22Dx7 [mm]	
Cantidad	2	
Resortes		
Modelo	Ajax 74	
Descripción	Resorte de compresión de 9mm diámetro interno, acero templado galvanizado.	
Dimensiones	10Dx50 [mm]	
Cantidad	4	
Collar de retención		
Modelo	C8	
Descripción	Collar de retención Acme 8mm de acero.	
Dimensiones	16Dx8 [mm]	
Cantidad	4	
Soporte para eje		
Modelo	SHF10	
Descripción	Soporte de pared para varilla lisa de 8 mm hecho de aluminio.	
Dimensiones	43x24x10 [mm]	
Cantidad	4	

B. Planos: Piezas y ensamble del prototipo



PARTS LIST			
ITEM	QTY	TITLE	DESCRIPTION
1	1	Ensamble Base	MDF
2	1	Ensamble Frame	ACP y PLA
3	2	Soporte baleros	PLA
4	1	Abrazadera sensor	PLA
5	1	Tapa A	MDF
6	1	Tapa B	MDF
7	4	Barra rectificada	Acero
8	2	Eje rectificado	Acero
9	4	Rodamiento lineal	Acero
10	4	Seguro	Acero
11	4	Collar de retencion	Acero
12	4	Resorte compresión	Acero
13	4	Soporte eje shf8mm	Aluminio
14	2	Motores brushless	
15	2	ESC 30A	
16	1	ESP32	
17	1	IMU BMI085	
18	1	Sensor de distancia	

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Facultad de Ingeniería

Título: **Ensamble General**

Escala: 1 : 4

Diseñó: González Duardo Saúl
Romero Rivas Víctor Octavio

Proyecto: Dispositivo VTOL

Ensamble:

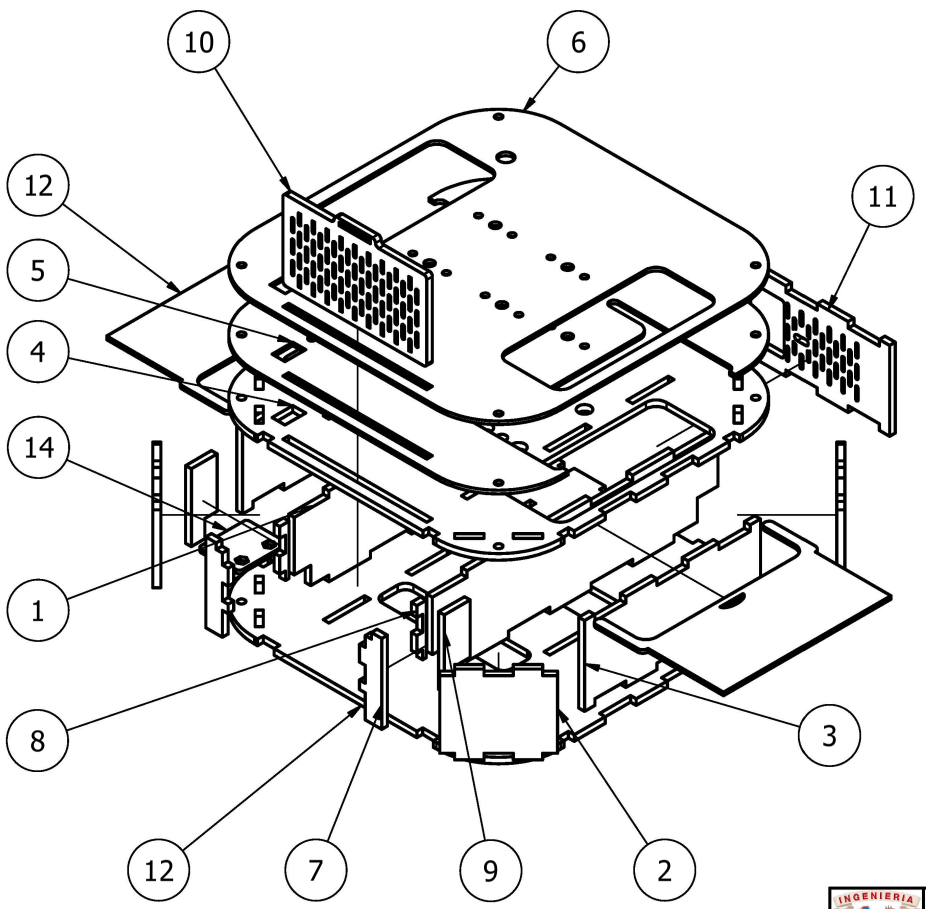
Unidades: mm

Formato: A4

Cantidad: 1

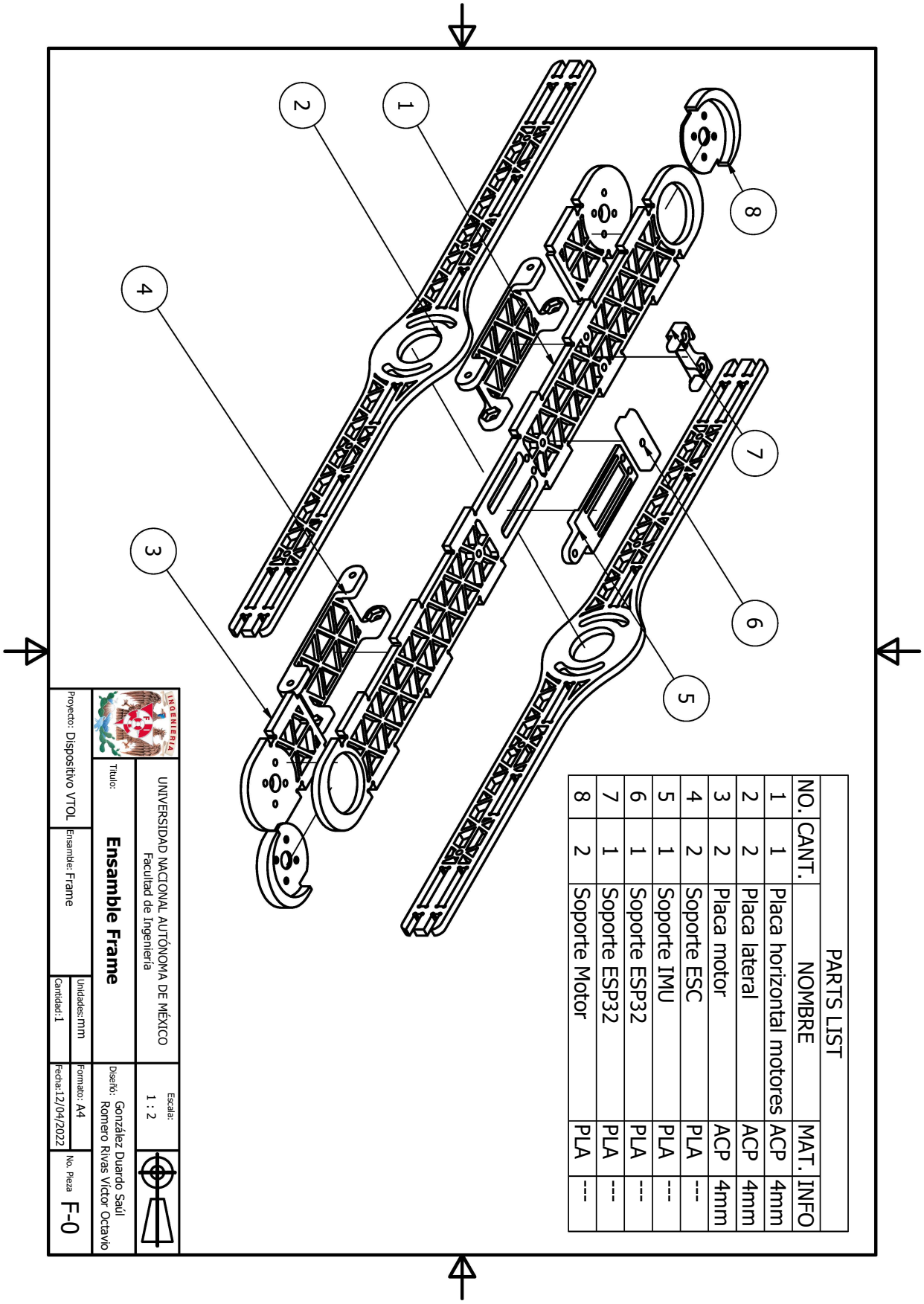
Fecha: 12/04/2022

No. Pieza G-0



PARTS LIST				
NO.	CANT	NOMBRE	MAT.	INFO
1	2	Lateral interior	MDF	5.5mm
2	4	Esquina exterior	MDF	5.5mm
3	2	Lateral exterior	MDF	5.5mm
4	1	Superior 1	MDF	5.5mm
5	1	Superior 2	MDF	3mm
6	1	Superior 3	MDF	3mm
7	2	Frontal 1	MDF	5.5mm
8	2	Frontal 2	MDF	5.5mm
9	2	Frontal 3	MDF	5.5mm
10	1	Puerta deslizante	MDF	5.5mm
11	1	Posterior exterior	MDF	5.5mm
12	1	Inferior	MDF	5.5mm
13	2	Tapa deslizante	MDF	3mm
14	1	Base Fenolica	MDF	3mm

	UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO Facultad de Ingeniería		Escala: 1 : 4	
	Titulo: **Ensamble Base**		Diseñó: González Duardo Saúl Romero Rivas Víctor Octavio	
Proyecto: Dispositivo VTOL	Ensamble: Base	Unidades: mm Cantidad: 1	Formato: A4 Fecha: 12/04/2022	No. Pieza B-0



PARTS LIST

NO.	CANT.	NOMBRE	MAT.	INFO
1	1	Placa horizontal motores	ACP	4mm
2	2	Placa lateral	ACP	4mm
3	2	Placa motor	ACP	4mm
4	2	Soporte ESC	PLA	---
5	1	Soporte IMU	PLA	---
6	1	Soporte ESP32	PLA	---
7	1	Soporte ESP32	PLA	---
8	2	Soporte Motor	PLA	---

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Facultad de Ingeniería

Título: **Ensamble Frame**

Escala: 1 : 2

Diseñó: González Duardo Saúl
Romero Rivas Víctor Octavio

Proyecto: Dispositivo VTOL

Ensamble: Frame

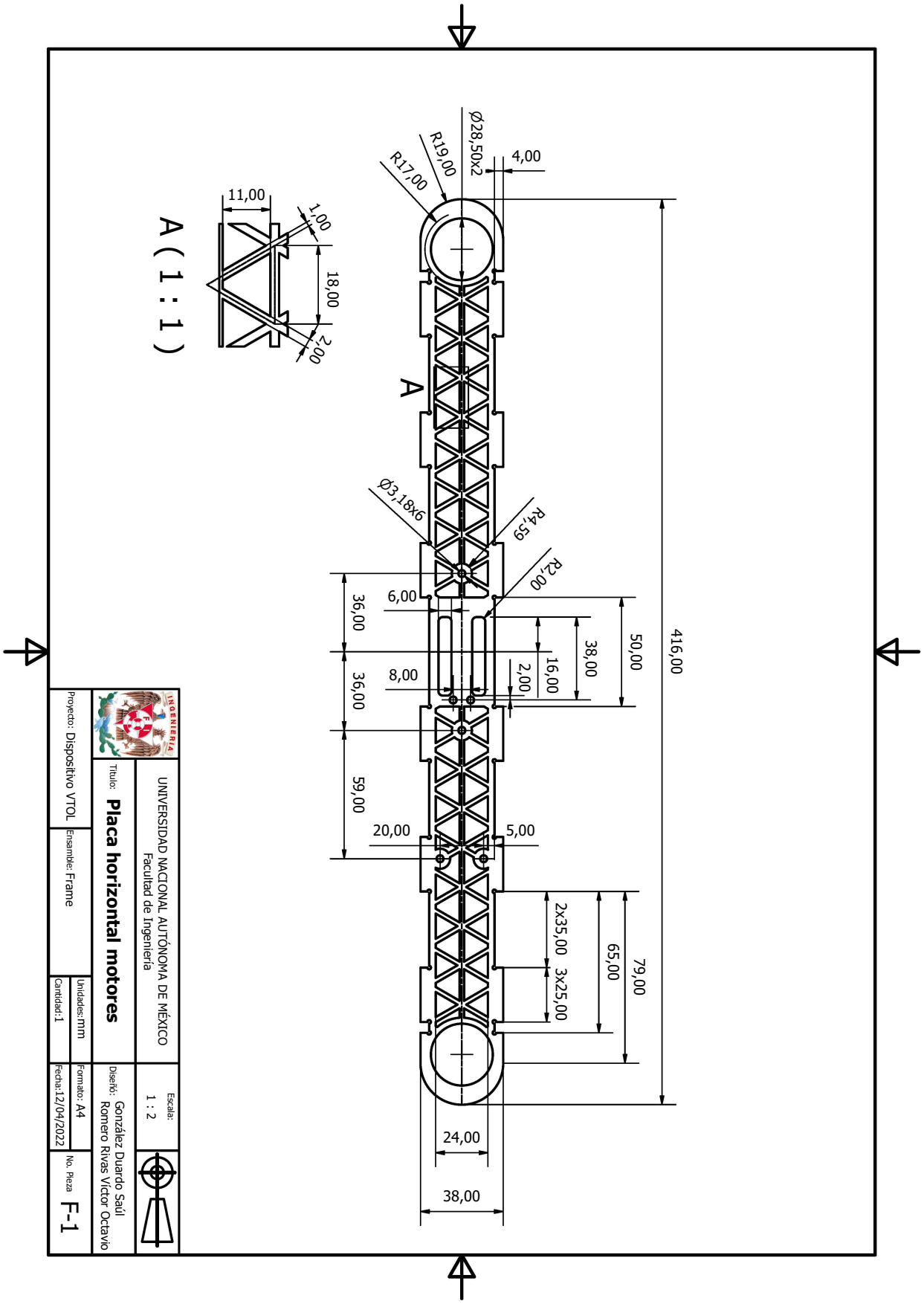
Unidades: mm

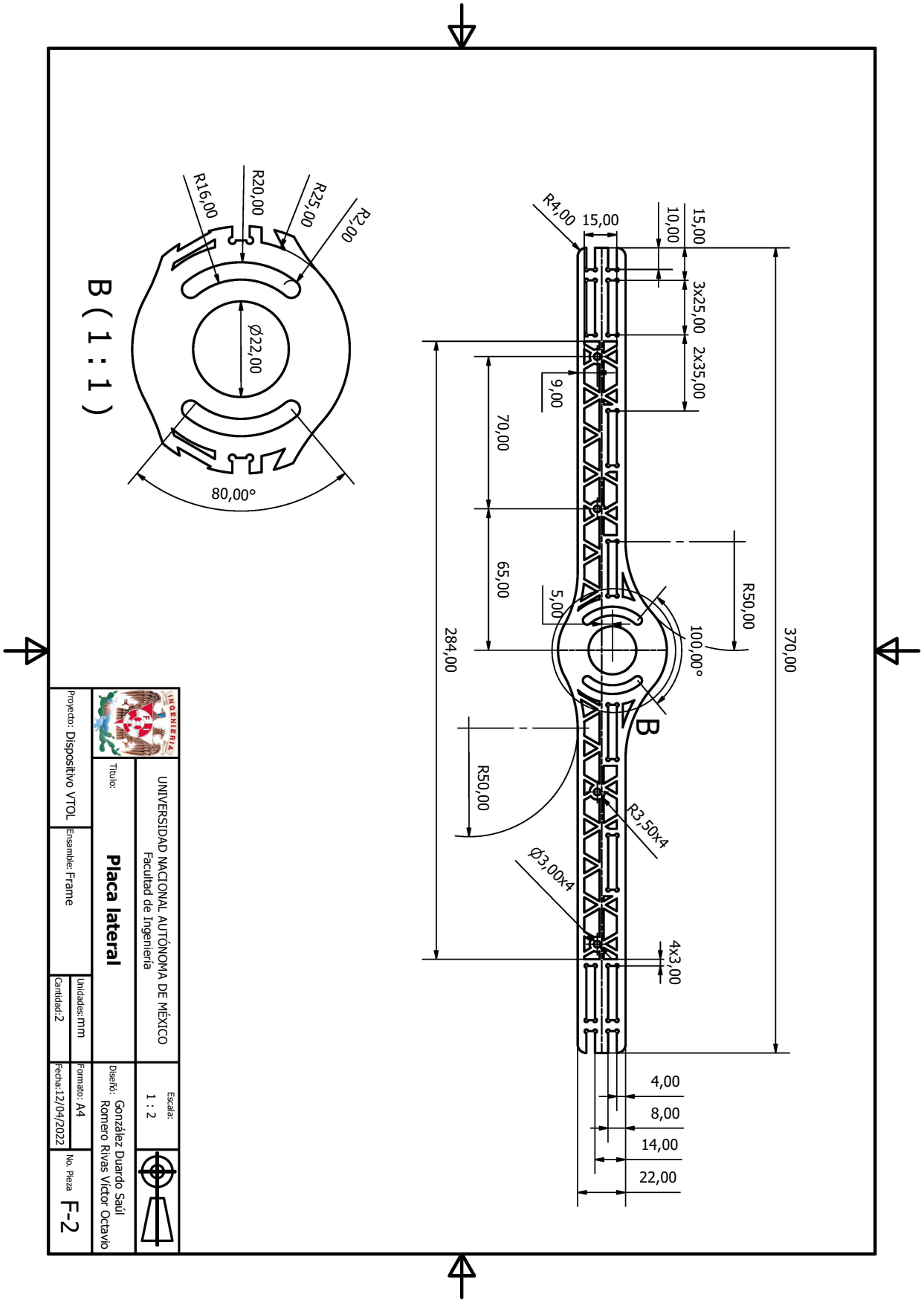
Cantidad: 1

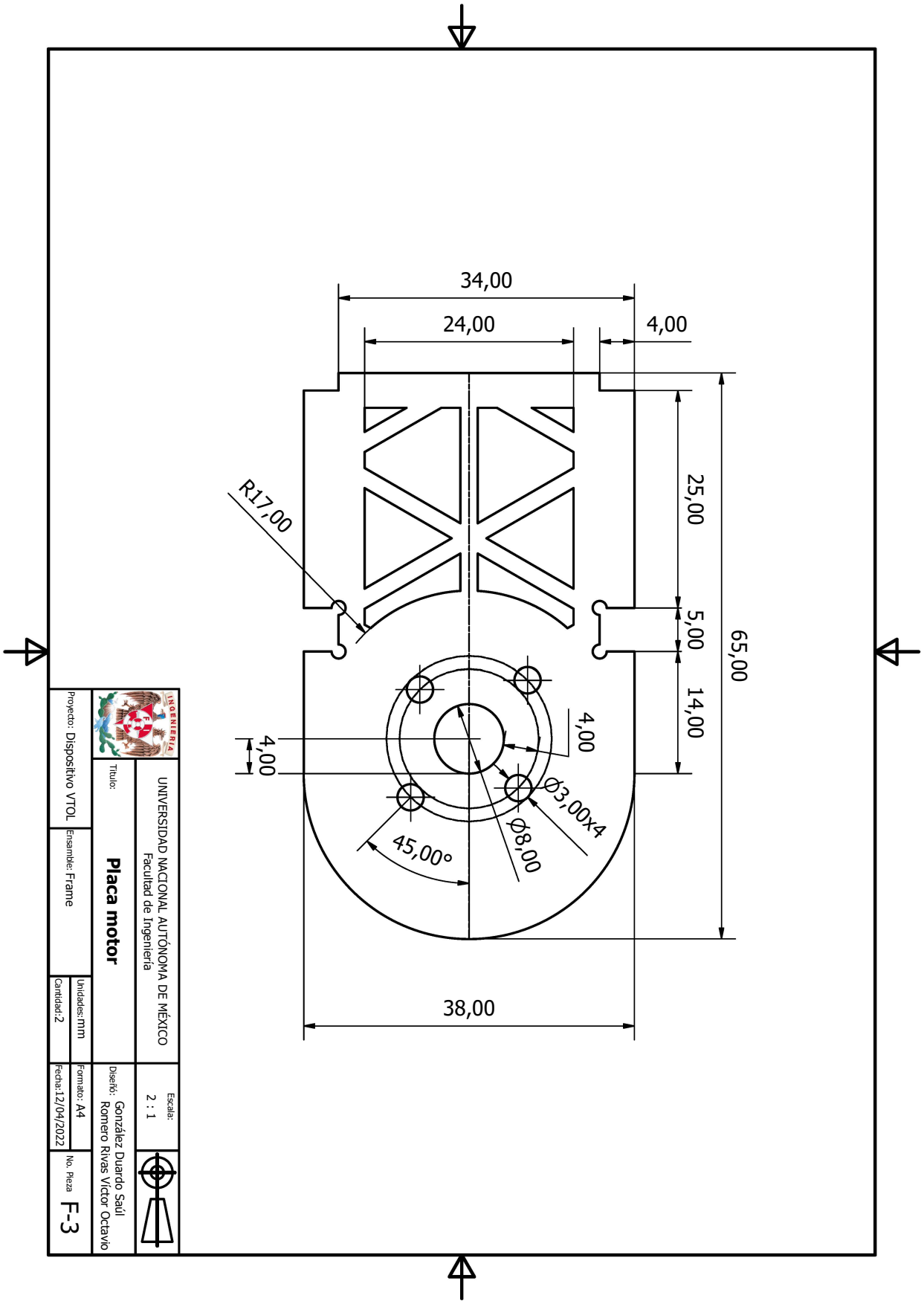
Formato: A4

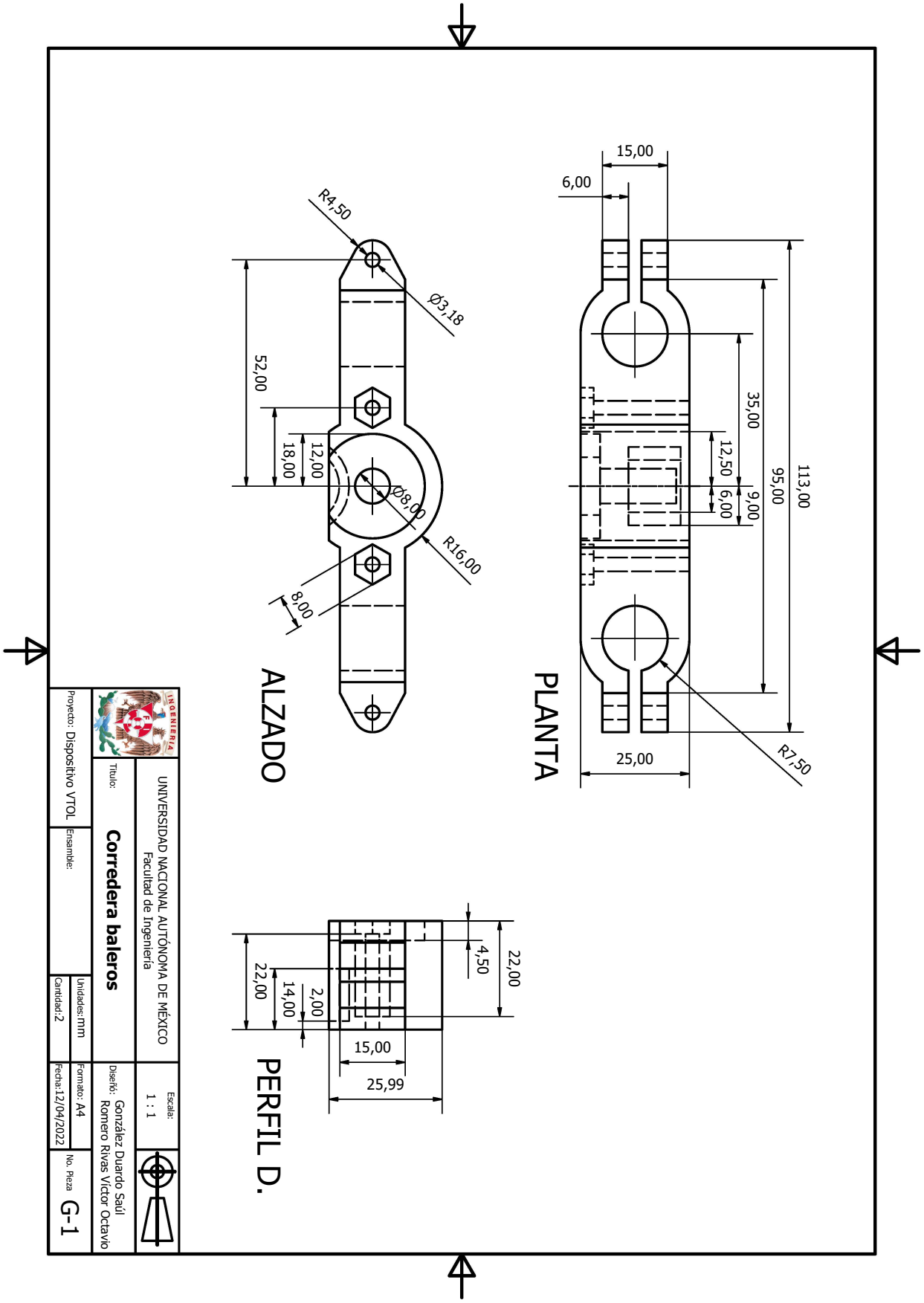
Fecha: 12/04/2022

No. Pieza F-0


Ø28,50x2
R19,00
R17,00
4,00
11,00
1,00
18,00
2,00
A (1 : 1)
A
Ø3,18x6
R4,59
R2,00
6,00
36,00
8,00
36,00
59,00
38,00
16,00
2,00
50,00
416,00
20,00
5,00
2x35,00
3x25,00
65,00
79,00
24,00
38,00
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Facultad de Ingeniería
Título: Placa horizontal motores
Escala: 1 : 2
Diseñó: González Duardo Saúl
Romero Rivas Víctor Octavio
Proyecto: Dispositivo VTOL
Ensamble: Frame
Unidades: mm
Cantidad: 1
Formato: A4
Fecha: 12/04/2022
No. Pieza F-1


R4,00
15,00
10,00
15,00
3x25,00
2x35,00
9,00
70,00
65,00
5,00
284,00
R50,00
100,00°
370,00
B
R50,00
R3,50x4
Ø3,00x4
4x3,00
4,00
8,00
14,00
22,00
B (1 : 1)
R16,00
R20,00
R25,00
R2,00
Ø22,00
80,00°
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Facultad de Ingeniería
Título:
Placa lateral
Escala: 1 : 2
Diseñó: González Duardo Saúl
Romero Rivas Víctor Octavio
Proyecto: Dispositivo VTOL
Ensamble: Frame
Unidades: mm
Cantidad: 2
Formato: A4
Fecha: 12/04/2022
No. Pieza
F-2


34,00
24,00
4,00
25,00
5,00
14,00
65,00
R17,00
4,00
4,00
Ø3,00x4
Ø8,00
45,00°
38,00
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Facultad de Ingeniería
Título:
Placa motor
Escala: 2 : 1
Diseñó: González Duardo Saúl
Romero Rivas Víctor Octavio
Proyecto: Dispositivo VTOL
Ensamble: Frame
Cantidad: 2
Unidades: mm
Formato: A4
Fecha: 12/04/2022
No. Pieza
F-3


PLANTA
15,00
6,00
113,00
95,00
35,00
12,50
6,00
9,00
25,00
R7,50
ALZADO
R4,50
Ø3,18
52,00
18,00
12,00
Ø8,00
R16,00
8,00
PERFIL D.
4,50
22,00
22,00
14,00
2,00
15,00
25,99
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Facultad de Ingeniería
Título:
Corredera baleros
Escala: 1 : 1
Diseñó: González Duardo Saúl
Romero Rivas Víctor Octavio
Proyecto: Dispositivo VTOL
Ensamble:
Unidades: mm
Cantidad: 2
Formato: A4
Fecha: 12/04/2022
No. Pieza
G-1

C. CAE: Tablas de propiedades de los materiales

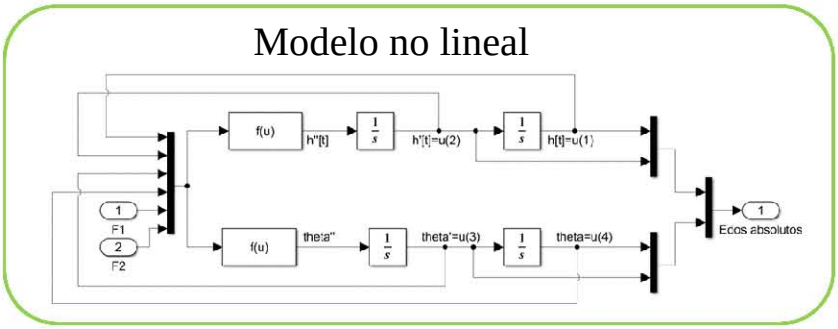
Material	**ACP (4mm)**		
General	Densidad	1.273	g/cm^3
	Resistencia a la fluencia	43.995	Mpa
	Resistencia Máxima a tracción	43.995	Mpa
Tensión	Módulo de Young	40.099	Gpa
	Coeficiente de Poisson	0.3	su
	Modulo cortante	15.423	Gpa
Nombre y No. de pieza	1 Placa inferior motores		
	2 laterales		
	2 Soportes de motor		

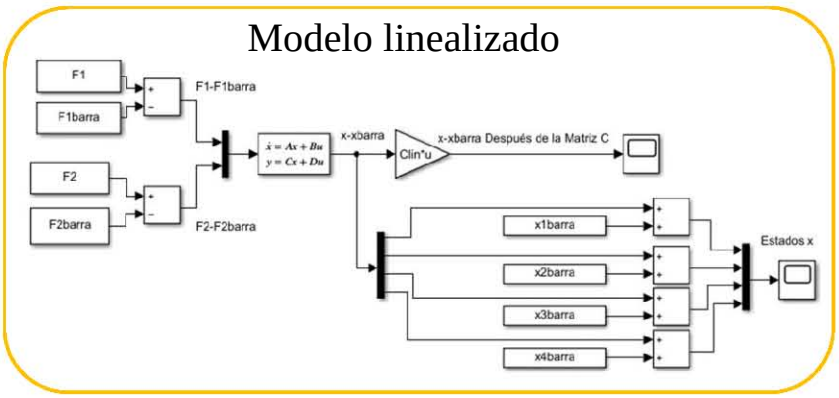
Material	**PLA**		
General	Densidad	1.039	g/cm^3
	Resistencia a la fluencia	30.999	Mpa
	Resistencia Máxima a tracción	35.999	Mpa
Tensión	Módulo de Young	3.099	Gpa
	Coeficiente de Poisson	0.203	su
	Modulo cortante	1.288	Gpa
Nombre y No. de pieza	2 Soportes para ESC		
	1 Soporte IMU		
	2 Soporte ESP32		
	2 Separadores de motor		

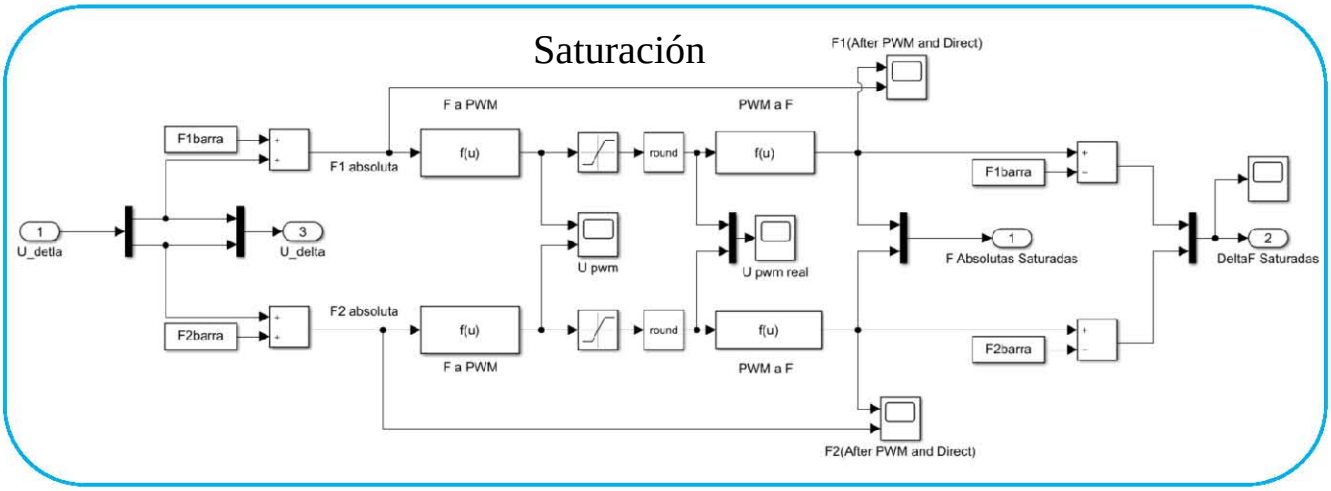
Material	**Acero suave**		
General	Densidad	8	g/cm^3
	Resistencia a la fluencia	250	Mpa
	Resistencia Máxima a tracción	540	Mpa
Tensión	Módulo de Young	193	Gpa
	Coeficiente de Poisson	0.3	su
	Modulo cortante	74.231	Gpa
Nombre y No. de pieza	4 Barras de acero		
	2 Baleros		
	4 baleros lineales		

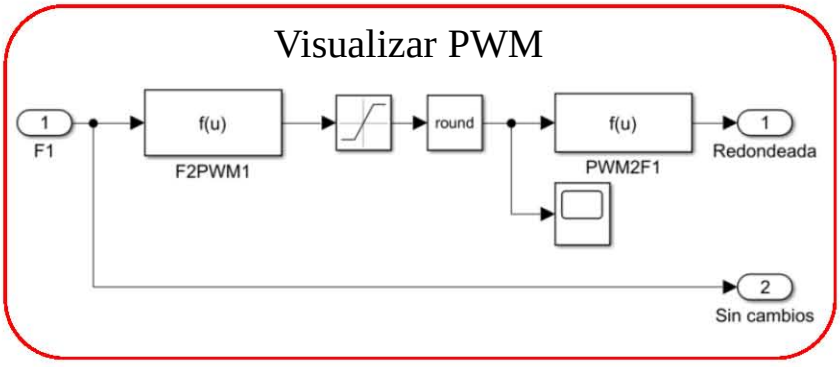
Material	**Acero inoxidable**		
General	Densidad	8	g/cm^3
	Resistencia a la fluencia	250	Mpa
	Resistencia Máxima a tracción	540	Mpa
Tensión	Módulo de Young	193	Gpa
	Coeficiente de Poisson	0.3	su
	Modulo cortante	74.231	Gpa
Nombre y No. de pieza	4 Barras de acero		
	2 Baleros		
	4 baleros lineales		

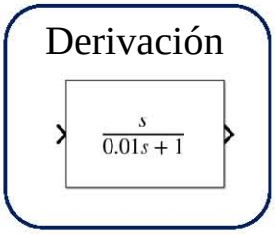
D. Simulink: Diagramas de bloques y subsistemas

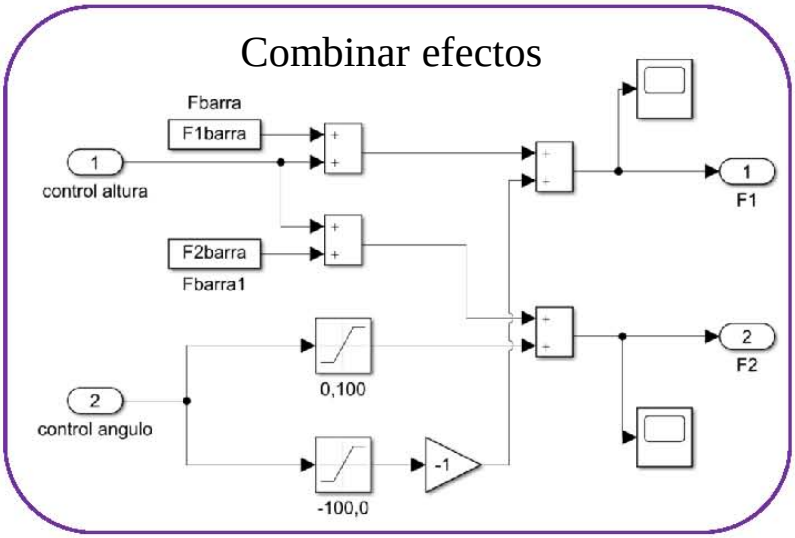


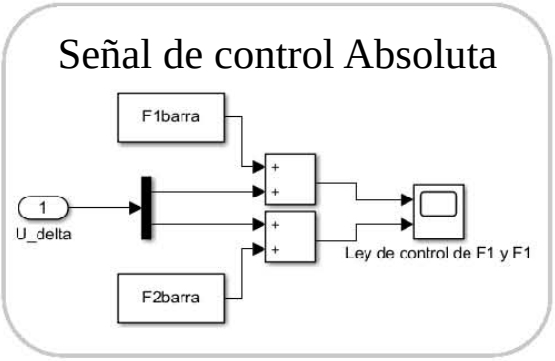


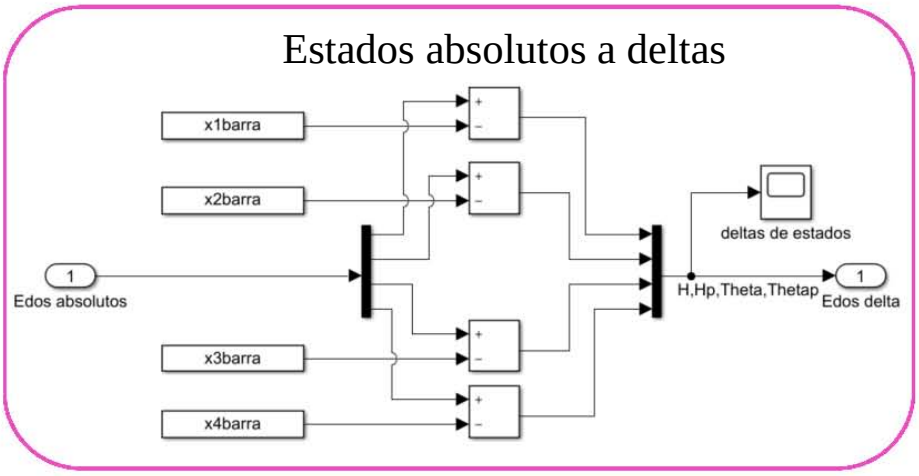


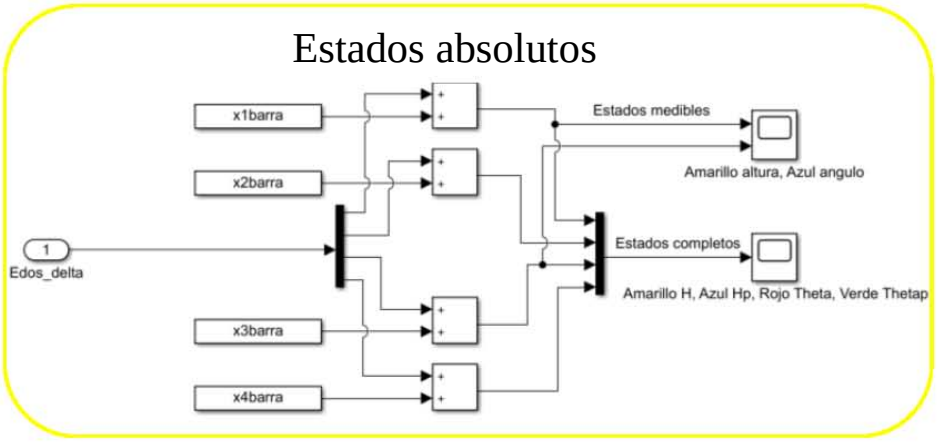


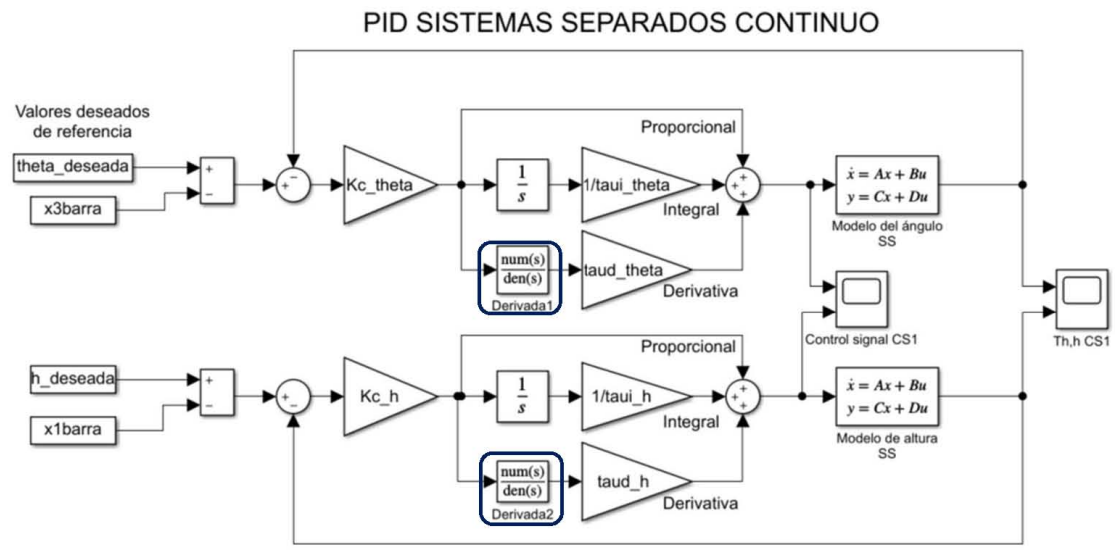


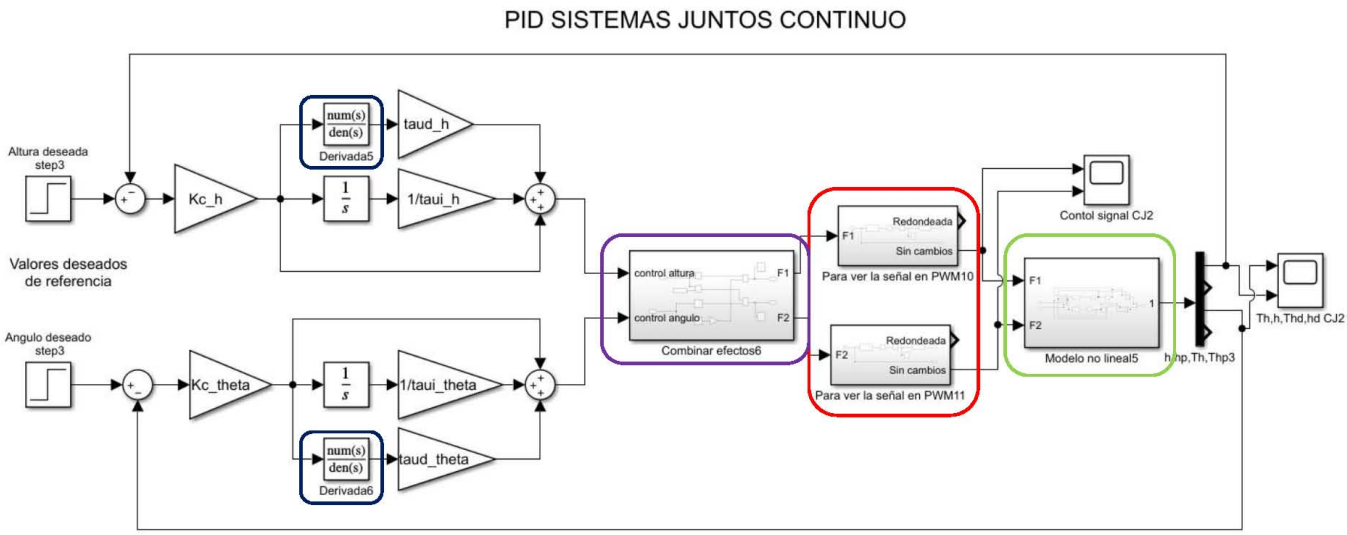

Combinar efectos
Fbarra
F1barra
1
control altura
F2barra
Fbarra1
1
F1
2
F2
0,100
2
control angulo
-1
-100,0

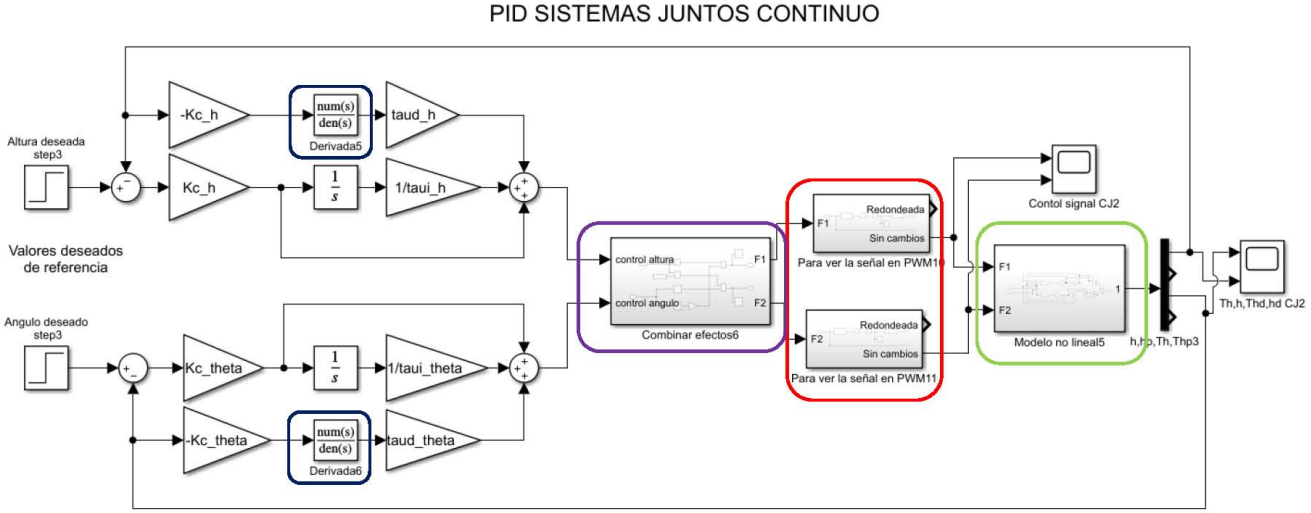

Señal de control Absoluta
F1barra
1
U_delta
F2barra
Ley de control de F1 y F1

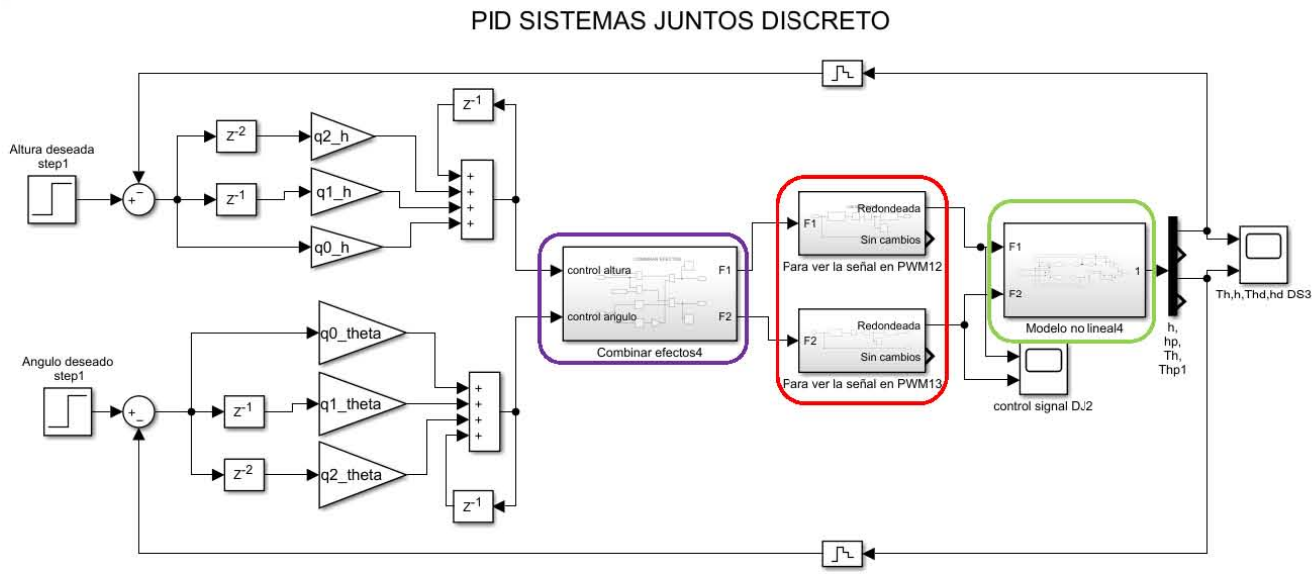

Estados absolutos a deltas
x1barra
x2barra
1
Edos absolutos
x3barra
x4barra
deltas de estados
H,Hp,Theta,Thetap
1
Edos delta

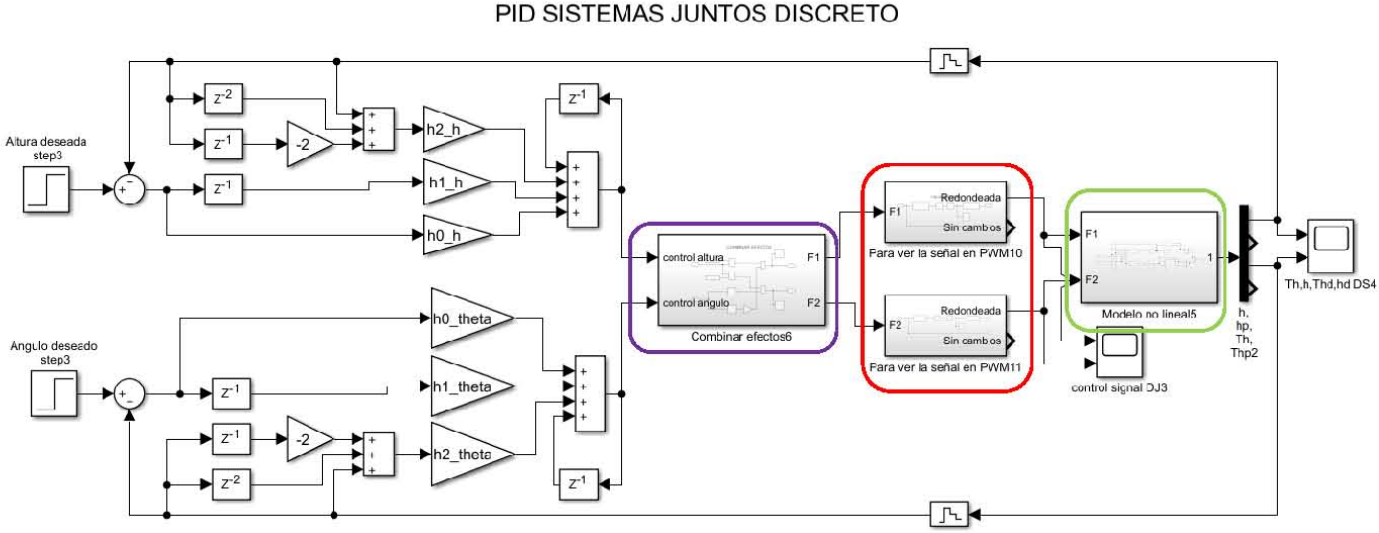

Estados absolutos
x1barra
x2barra
1
Edos_delta
x3barra
x4barra
Estados medibles
Amarillo altura, Azul angulo
Estados completos
Amarillo H, Azul Hp, Rojo Theta, Verde Thetap

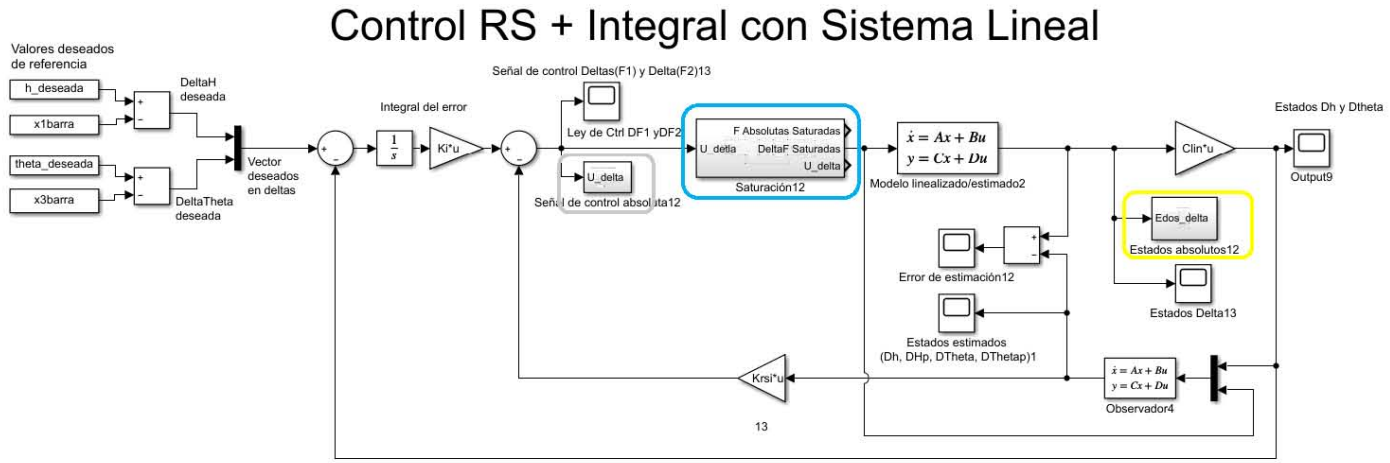

PID SISTEMAS SEPARADOS CONTINUO
Valores deseados de referencia
theta_deseada
x3barra
Kc_theta
Proporcional
1/taui_theta
Integral
Derivada1
taud_theta
Derivativa
Modelo del ángulo SS
Control signal CS1
Th,h CS1
h_deseada
x1barra
Kc_h
1/taui_h
taud_h
Derivada2
Modelo de altura SS

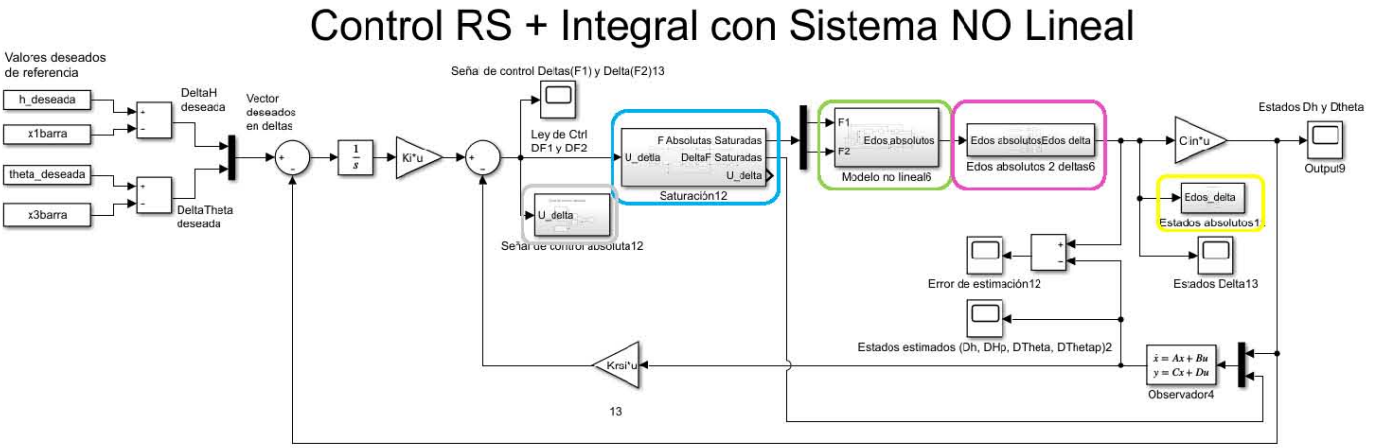

PID SISTEMAS JUNTOS CONTINUO
Altura deseada step3
Valores deseados de referencia
Angulo deseado step3
Kc_h
Kc_theta
taud_h
1/taui_h
1/taui_theta
taud_theta
Derivada5
Derivada6
Combinar efectos6
control altura
control angulo
Redondeada
Sin cambios
Para ver la señal en PWM10
Para ver la señal en PWM11
Contol signal CJ2
Modelo no lineal5
Th,h,Thd,hd CJ2

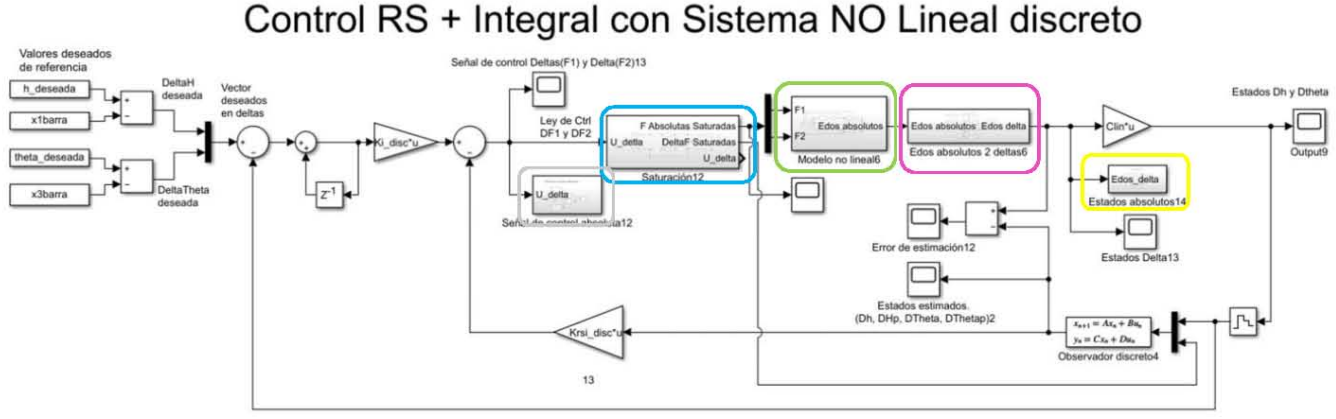

PID SISTEMAS JUNTOS CONTINUO
Altura deseada step3
Valores deseados de referencia
Angulo deseado step3
-Kc_h
Kc_h
Kc_theta
-Kc_theta
taud_h
1/taui_h
1/taui_theta
taud_theta
Derivada5
Derivada6
Combinar efectos6
Redondeada
Sin cambios
Para ver la señal en PWM10
Para ver la señal en PWM11
Contol signal CJ2
Modelo no lineal5
Th,h,Thd,hd CJ2


PID SISTEMAS JUNTOS DISCRETO
Altura deseada
step1
Angulo deseado
step1
q2_h
q1_h
q0_h
q0_theta
q1_theta
q2_theta
control altura
control angulo
Combinar efectos4
Redondeada
Sin cambios
Para ver la señal en PWM12
Para ver la señal en PWM13
Modelo no lineal4
control signal DJ2
h,
hp,
Th,
Thp1
Th,h,Thd,hd DS3


PID SISTEMAS JUNTOS DISCRETO
Altura deseada
step3
Angulo deseado
step3
h2_h
h1_h
h0_h
h0_theta
h1_theta
h2_theta
control altura
control angulo
Combinar efectos6
Redondeada
Sin cambios
Para ver la señal en PWM10
Para ver la señal en PWM11
Modelo no lineal5
control signal DJ3
h,
hp,
Th,
Thp2
Th,h,Thd,hd DS4


Control RS + Integral con Sistema Lineal
Valores deseados de referencia
h_deseada
x1barra
theta_deseada
x3barra
DeltaH deseada
DeltaTheta deseada
Vector deseados en deltas
Integral del error
Ki*u
Señal de control Deltas(F1) y Delta(F2)13
Ley de Ctrl DF1 yDF2
U_delta
Señal de control absoluta12
F Absolutas Saturadas
U_detla
DeltaF Saturadas
U_delta
Saturación12
Modelo linealizado/estimado2
Clin*u
Estados Dh y Dtheta
Output9
Edos_delta
Estados absolutos12
Estados Delta13
Error de estimación12
Estados estimados (Dh, DHp, DTheta, DThetap)1
Krsi*u
13
Observador4


Control RS + Integral con Sistema NO Lineal
Valores deseados de referencia
h_deseada
x1barra
theta_deseada
x3barra
DeltaH deseada
DeltaTheta deseada
Vector deseados en deltas
Ki*u
Señal de control Deltas(F1) y Delta(F2)13
Ley de Ctrl DF1 y DF2
U_delta
Señal de control absoluta12
F Absolutas Saturadas
U_detla
DeltaF Saturadas
U_delta
Saturación12
F1
F2
Edos absolutos
Modelo no lineal6
Edos absolutosEdos delta
Edos absolutos 2 deltas6
Clin*u
Estados Dh y Dtheta
Output9
Edos_delta
Estados absolutos1
Estados Delta13
Error de estimación12
Estados estimados (Dh, DHp, DTheta, DThetap)2
Krsi*u
13
Observador4


Control RS + Integral con Sistema NO Lineal discreto
Valores deseados de referencia
h_deseada
x1barra
theta_deseada
x3barra
DeltaH deseada
DeltaTheta deseada
Vector deseados en deltas
Ki_disc*u
Señal de control Deltas(F1) y Delta(F2)13
Ley de Ctrl DF1 y DF2
U_delta
F Absolutas Saturadas
U_detla
DeltaF Saturadas
U_delta
Saturación12
F1
F2
Edos absolutos
Modelo no lineal6
Edos absolutos Edos delta
Edos absolutos 2 deltas6
Clin*u
Estados Dh y Dtheta
Output9
Edos_delta
Estados absolutos14
Estados Delta13
Error de estimación12
Estados estimados (Dh, DHp, DTheta, DThetap)2
Krsi_disc*u
13
Observador discreto4

E. Matlab: Código para control por PID

15/07/22 04:13 PM C:\Users\totos\Documents\Tesis_Ingeniería\Programas definitivos\Matlab y ...\Script_control_PID.m 1 of 4

```
%{
Código para el controlador de la tésis del Proyecto tipo DRON
En este caso se intenta separar la dinámica en dos partes más simples
y se controla con PID discreto

Alumnos:
   Víctor Octavio Romero Rivas
   González Duardo Saúl

Asesor: Edmundo Gabriel Rocha Cózatl

Derechos Reservados
%}

clc
clear

% ----------------- Definir algunos parámetros ---------------------
%Estos parámetros son para no cometer errores si más adelante se quiere
%cambiar alguna forma de calcular los polos, ganancias y matrices

%Se va a usar la sintonización por ziegler nichols o definir directo los
%valores
PID_zieglerNichols = false;
PID_zieglerNichols_experimentales = false;


%% ********* CONSTANTES PARA LOS MODELOS FENOMENOLÓGICOS *****************

g=9.78; %Gravedad
beta1=0.01;%fricción lineal
beta2=0.01;%fricción rotacional
l=0.185;%largo del brazo [m]
L = l; %Así lo pusimos en el simulink

%punto de equilibrio experimental. Utilizamos este punto como referencia
%para algunas constantes
%Caracterización (500-1000)
F1barra=0.018465*(685)- 10.0694; %N
F2barra=0.018465*(690)- 10.0694; %N
x1barra=0.15;
x2barra=0;
x3barra=0;
x4barra=0;

m3 = 0.360; %masa central
m1 = (F1barra/g) - (m3/2); %Motor izquierdo
m2 = (F2barra/g) - (m3/2); %Motor Derecho
mt = m1+m2+m3; %Masa total

%Valores anteriores (para no perder la referencia)
%m1=0.078; %Motor Izquierdo (Masas ligeramente desbalanceadas 0.054)
%m2=0.084; %Motor Derecho
%m3=0.400;%Masa central (0.346)

J = (m1+m2)*(l^2); %Momento de inercia

%Condiciones iniciales ( solo con modelos edos y no lineal)
theta_inicial = deg2rad(-15);
h_inicial = 0.08; %[m]

%Valores Deseados
h_deseada=0.15; %altura [m]
theta_deseada=deg2rad(0); %ángulo [°] convertidos a radianes

%% ***************** DECLARACIÓN DE LOS MODELOS *************************

%Modelos para la altura
% A_altura = [0,1;0,-(beta1/mt)];
```

```
% B_altura = [0;1/mt];
% C_altura = [1,0];
% D_altura = 0;
% altura_edos_sys = ss(A_altura,B_altura,C_altura,D_altura); %modelos edos
% altura_tf_sys = tf([1/mt],[1,beta1/mt,0]); %modelo función de transferencia

A_altura = [0,1;0,-(beta1/mt)];
B_altura = [0;2/mt];
C_altura = [1,0];
D_altura = 0;
altura_edos_sys = ss(A_altura,B_altura,C_altura,D_altura); %modelos edos
altura_tf_sys = tf([2/mt],[1,beta1/mt,0]); %modelo función de transferencia

%{
Se verificó que si da lo mismo cuando se comprueba con:
[b,a] = ss2tf(A_altura,B_altura,C_altura,D_altura)
syss = tf(b,a)
Por lo tanto los cálculos son correctos
%}

%Modelo para el ángulo
A_angulo = [0,1;0,-(beta2/J)];
B_angulo = [0;L/J];
C_angulo = [1,0];
D_angulo = 0;
angulo_edos_sys = ss(A_angulo,B_angulo,C_angulo,D_angulo);
angulo_tf_sys = tf([L/J],[1,beta2/J,0]);
%La equivalencia de estos sistemas también se comprobó

%% ********** CONTROL PID PARA LOS SISTEMAS ************************

%Modo paralelo

if ~PID_zieglerNichols
    %------------------- PARA EL ÁNGULO --------------------------------
    %Cálculo de las constantes por asignación de polos

    A_theta = angulo_tf_sys.Numerator{1}(3); %angulo: 23.4319   altura: 1.722
    B_theta = angulo_tf_sys.Denominator{1}(2); %angulo: 1.266   altura: 0.0172
    ts_theta = 5; %5  para la altura, 5 para angulo
    sp_theta = 15; %0.5 para la altura, 15 para angulo

    re_theta = 5/ts_theta;
    im_theta = (-pi*re_theta)/(log(sp_theta/100));

    polos_deseados_theta = [-re_theta+im_theta*i,-re_theta-im_theta*i,-10];
    poli_theta = poly(polos_deseados_theta);

    % chido_h = [2.7, 0.4524, 0.1]; %El que quedó puesto en el proyecto
    % chido_theta = [0.84, 1.0, 0.145] %El que quedó en el proyecto

    I_theta = poli_theta(4)/A_theta;
    P_theta= poli_theta(3)/A_theta;
    D_theta = (poli_theta(2)-B_theta)/A_theta;

    % ------------ PARA LA ALTURA ------------------------
    %Cálculo de las constantes por asignación de polos

    A_h = altura_tf_sys.Numerator{1}(3); %angulo: 23.4319   altura: 1.722
    B_h = altura_tf_sys.Denominator{1}(3); %angulo: 1.266   altura: 0.0172
    ts_h = 5; %5  para la altura, 5 para angulo
    sp_h = 0.5; %0.5 para la altura, 15 para angulo

    re_h = 5/ts_h;
    im_h = (-pi*re_h)/(log(sp_h/100));

    polos_deseados_h = [-re_h+im_h*i,-re_h-im_h*i,-10];
    poli_h = poly(polos_deseados_h);

```

```
139     I_h = poli_h(4)/A_h;
140     P_h= poli_h(3)/A_h;
141     D_h = (poli_h(2)-B_h)/A_h;
142
143 else
144     % Para la altura
145     filename='ZN_altura.xls';
146     datos_interfaz=xlsread(filename);
147     plot(datos_interfaz(:,1),datos_interfaz(:,3))
148
149     ku_h = 6.09; %constante K crítica
150     pu_h = 0.55; %Periodo crítico 0.76
151
152     kp_h = (0.6*ku_h)
153     %kp_h = (0.6*ku_h)/2
154     taui_h = 0.5*pu_h
155     taud_h = 0.125*pu_h
156
157     if PID_zieglerNichols_experimentales
158         kp_h = 2.7;
159         taui_h = 0.4524;
160         taud_h = 0.1;
161     end
162
163     P_h = kp_h
164     I_h = kp_h/taui_h
165     D_h = kp_h*taud_h
166
167     %[2.1, 0.38, 0.11]
168     %chido_h = [2.5, 0.4524, 0.085]; %El que quedó puesto en el proyecto
169     %[2.2, 0.3985, 0.0966]
170
171     %Para el angulo
172     figure(2)
173     filename2 = 'ZN_angulo.xls';
174     datos_interfaz2 = xlsread(filename2);
175     plot(datos_interfaz2(:,1),datos_interfaz2(:,2));
176
177     ku_theta = 1.4; %K crítica
178     pu_theta = 1.16; %Periodo crítico [s]
179
180     kp_theta = (0.6*ku_theta)/1;
181     taui_theta = 0.5*pu_theta;
182     taud_theta = 0.125*pu_theta;
183
184     if PID_zieglerNichols_experimentales
185         kp_theta = 0.84;
186         taui_theta = 1.0;
187         taud_theta = 0.145;
188     end
189
190     P_theta = kp_theta
191     I_theta = kp_theta/taui_theta
192     D_theta = kp_theta*taud_theta
193
194     %[0.84,0.58,0.145]
195     %chido_theta = [0.84, 1.0, 0.145]; %El que quedó en el proyecto
196 end
197 %% ********** DISCRETIZACIÓN DEL PID PARA LOS SISTEMAS ******************
198
199 %Tiempo de muestreo
200 % figure(1)
201 % bode(altura_tf_sys)
202 % figure(2)
203 % bode(angulo_tf_sys)
204
205 Ts = 0.010; %segundos
206
207 %Se está utilizando una discretización trapezoidal
```

```
208
209 %Controlador del ángulo
210 % El controlador se discretizó en forma ideal (K,taui,taud);
211 Kc_theta = P_theta;
212 taui_theta = Kc_theta/I_theta;
213 taud_theta = D_theta/Kc_theta;
214
215 q0_theta = Kc_theta*(1+(Ts/2*taui_theta)+(taud_theta/Ts));
216 q1_theta = Kc_theta*(-1+(Ts/2*taui_theta)-(2*taud_theta/Ts));
217 q2_theta = Kc_theta*(taud_theta/Ts);
218
219 PIDz_angulo = filt([q0_theta,q1_theta,q2_theta],[1,-1],Ts);
220
221 %Para el PID con derivada en el sensado
222 h0_theta = Kc_theta*(1+(Ts/(2*taui_theta)));
223 h1_theta = Kc_theta*(-1+(Ts/(2*taui_theta)));
224 h2_theta = -Kc_theta*(taud_theta/Ts);
225
226 %Controlador de la altura
227 % El controlador se discretizó en forma ideal (K,taui,taud);
228 Kc_h = P_h;
229 taui_h = Kc_h/I_h;
230 taud_h = D_h/Kc_h;
231
232 q0_h = Kc_h*(1+(Ts/2*taui_h)+(taud_h/Ts));
233 q1_h = Kc_h*(-1+(Ts/2*taui_h)-(2*taud_h/Ts));
234 q2_h = Kc_h*(taud_h/Ts);
235
236 PIDz_altura = filt([q0_h,q1_h,q2_h],[1,-1],Ts);
237
238 %Para el PID con derivada en el sensado
239 h0_h = Kc_h*(1+(Ts/(2*taui_h)));
240 h1_h = Kc_h*(-1+(Ts/(2*taui_h)));
241 h2_h = -Kc_h*(taud_h/Ts);
242
```

F. Matlab: Código para control por RS+integral


```
%{
Código para el controlador de la tésis del Proyecto tipo DRON
Víctor Octavio Romero Rivas
González Duardo Saúl

Asesor: Edmundo Gabriel Rocha Cózatl

Control RE, RS y RS+int para el sistema completo (4to orden)
Derechos Reservados
%}

clear
clc

%Variables para el comportamieto del script
with_estimated_model = false; %ture -> modelo estimado
                        %false -> modelo linealizado
with_estimated_model_simu = false; %ture -> modelo estimado en la simulación
                          %false -> modelo linealizado en la simulación
with_observer = true;
with_integral = true;

with_lqr = false; %true -> utiliza el control óptimo lqr
              %false -> utiliza la ubicación de polos tradicional

%% DECLARACIÓN DE CONSTANTES PARA EL MODELO FENOMENOLÓGICO

g=9.78; %Gravedad
beta1=0.01;%fricción lineal
beta2=0.01;%fricción rotacional
l=0.185;%largo del brazo [m]
L = l; %Así lo pusimos en el simulink

%Caracterización (500-1000) y punto de equilibrio experimental
F1barra=0.018465*(685)- 10.0694; %N
F2barra=0.018465*(690)- 10.0694; %N
x1barra=0.15;
x2barra=0;
x3barra=0;
x4barra=0;

%Valores aprox esperados
% m1=0.078; %Motor Izquierdo (Masas ligeramente desbalanceadas 0.054)
% m2=0.084; %Motor Derecho
% m3=0.4;%Masa central (0.346)
%Valores para los cálculos
m3 = 0.360; %masa central
m1 = (F1barra/g) - (m3/2); %Motor izquierdo
m2 = (F2barra/g) - (m3/2); %Motor Derecho

%% CONDICIONES INICIALES DEL SISTEMA

c_i_h=0.05; %altura [m]
c_i_hp=0.0000; %velocidad lineal [m/s]
c_i_theta=deg2rad(-20); %angulo [°] (Convertido a radianes)
c_i_thetap=0.0000; %velocidad angular [rad/s]

cond_ini_sislineal=[c_i_h-x1barra, c_i_hp-x2barra, c_i_theta-x3barra,...
    c_i_thetap-x4barra];

%Valores deseados
h_deseada=0.15; %altura [m]
theta_deseada=deg2rad(0); %ángulo [°] convertidos a radianes


%% SISTEMA LINEALIZADO

if ~with_estimated_model
fprintf('Se utiliza el modelo linealizado: \n')
```

```
70 %Modelo fenomenológico sin fricción (de mathematica)
71 %Antes del cambio de pwm
72 % Alin = [0,1,0,0;0,0,0,0;0,0,0,1;0,0,0,0]; %Sin fricción
73 % Blin = [0,0;1.84599,1.71414;0,0;-33.7363,33.0235]; %Sin fricción
74
75 %Despues del cambio de pwm (el chido)
76 Alin = [0,1,0,0;0,0,0,0;0,0,0,1;0,0,0,0]; %Sin fricción
77 Blin = [0,0;1.96443,1.7642;0,0;-31.1265,30.0442]; %Sin fricción
78
79 %Modelo fenomenológico con fricción (de mathematica)
80 % Alin = [0,1,0,0;0,-0.0186321,0,0;0,0,0,1;0,0.00203969,0,0];
81 % Blin = [0,0;1.90095,1.82548;0,0;-11.7314,11.3235];
82
83 eigAlin=eig(Alin);
84
85 else
86 fprintf('Se utiliza el modelo estimado: \n')
87 %modelo obtenido de la identificación del sistema
88 %idsys = load('dying_sunset_model.mat');
89 %idsys = load('emerging_sunrise_model.mat');
90 idsys = load('presaged_eclipse_model.mat');
91 Alin = idsys.planta_est.A;
92 Blin = idsys.planta_est.B;
93
94 end
95 Clin=[1 0 0 0; 0 0 1 0];
96 Dlin=zeros(2,2);
97 sys_lineal = ss(Alin,Blin,Clin,Dlin)
98
99 if with_estimated_model_simu
100     fprintf('Se simula con el modelo estimado \n')
101     Aid = Alin;
102     Bid = Blin;
103
104 else
105     fprintf('Se simula con el modleo linealizado \n')
106     Aid = Alin;
107     Bid = Blin;
108 end
109
110 %% CONTROLABILIDAD Y OBSERVABILIDAD DEL SISTEMA
111
112 MClin=ctrb(Alin,Blin); %Matriz de controlabilidad
113 rangoMC=rank(MClin); %El rango debe ser completo (4)
114 MOlin=obsv(Alin,Clin); %Matriz de Observabilidad
115 rangoMO=rank(MOlin); %El rango debe ser completo (4)
116
117 if (rangoMC==4)
118     fprintf('El sistema es controlable \n')
119 else
120     fprintf('El sistema NO es controlable \n')
121 end
122 if (rangoMO==4)
123     fprintf('El sistema es observable \n ')
124 else
125     fprintf('El sistema NO observable \n')
126 end
127
128 %% CONTROL POR RE
129
130 fprintf('Control por RE \n')
131 if ~with_lqr
132     fprintf('Se calulan los valores por ubicación de polos \n')
133
134     %Polos deseados con base en el desempeño deseado
135     te = 5; %Tiempo de asentamiento [s] (5)
136     sp = 0.5; %Sobrepaso [%] (0.5)
137     n_veces = 5; %veces que se alejan los polos restantes
138
```

```
139     p_real = 5/te; %5 es por regla de 1%
140     p_imaginaria = (-pi*p_real)/(log(sp/100));
141
142     pd1 = -p_real+p_imaginaria*1j; %negativo porque debe ser estable
143     pd2 = -p_real-p_imaginaria*1j;
144
145     %Agregamos otros dos polos no dominantes al menos 5 o 10 veces alejados de la
146     %parte real de los polos deseados complejos conjugados.
147     pd3=-p_real*n_veces;
148     pd4=-p_real*(n_veces+0.1); %no son iguales porque sale mal la función place
149
150     %En caso de que se quieran asignar los polos directamente:
151     %pd1 = 0; pd2 = 0; pd3 = 0; pd4 = 0;
152
153     pd = [pd1;pd2;pd3;pd4]
154
155     %Asignación de polos para sacar las matrices K y K0
156     MatrizK=place(Alin,Blin,[pd1 pd2 pd3 pd4]);
157     MatrizK0=inv(Clin*inv(-Alin+Blin*MatrizK)*Blin);
158
159     K = MatrizK
160     K0 = MatrizK0
161 end
162
163 %% CONTROL POR LQR
164
165 if with_lqr
166     fprintf('Se calculan los valores con lqr \n')
167
168     %Es una forma de calcular la K dependiendo de una función de costo
169     % 4 estados, 2 entradas
170     mu1 = 1; %peso de la altura (30)
171     mu2 = 1; %peso de la velocidad lineal (1)
172     mu3 = 1; %peso del angulo (80)
173     mu4 = 1; %peso de la velocidad angular (1)
174     mu5 = 5; %peso de la entrada 1 (50)
175     mu6 = 5; %peso de la entrada 2 (50)
176
177     %Las matrices de Q deben de ser solo en la diagonal principal
178     Q = [mu1 0 0 0;0 mu2 0 0;0 0 mu3 0;0 0 0 mu4];
179     R = [mu5 0;0 mu6];
180     [Klqr,Plqr,polosLClqr]=lqr(Alin,Blin,Q,R);
181     K0lqr = inv(Clin*inv(-Alin+Blin*Klqr)*Blin);
182
183     polosLClqr
184
185     K = Klqr
186     K0 = K0lqr
187
188     %Para calcular el observador basado en el lqr
189     %Si se desea el observador basado en RE se debe comentar la linea
190     %p_real = abs(real(polosLClqr(1)));
191
192 end
193
194 %% CONTROL POR RS (OBSERVADOR)
195
196 if with_observer
197
198     fprintf('Se utiliza observador \n')
199     %Diseño del Observador
200
201     %Los polos deben ser de 2 a 5 veces más rápidos que el controlador
202     %no demasiado porque se inestabiliza y puede generar ruido
203     n_veces_obs = 10;
204     if with_lqr
205         pobs1 = polosLClqr(1)*n_veces_obs;
206         pobs2 = polosLClqr(2)*n_veces_obs;
207         pobs3 = polosLClqr(3)*n_veces_obs;
```

```
208       pobs4 = polosLClqr(4)*n_veces_obs;
209    else
210       pobs1=pd1*n_veces_obs;
211       pobs2=pd2*n_veces_obs;
212       pobs3=pd3*n_veces_obs;
213       pobs4=pd4*n_veces_obs;
214    end
215
216    pobs = real([pobs1;pobs2;pobs3;pobs4])
217
218    %Debido a la Dualidad de sistemas se puede obtener el observador si se
219    %obtiene el controlador del sistema dual.
220    MatrizLT=place(Alin',Clin',pobs);
221    Lobs=MatrizLT';
222
223    %realizamos las matrices A y B del observador
224    fprintf('Observador (lazo cerrado) \n')
225    Aobs=Alin-Lobs*Clin
226    Bobs=[Lobs Blin]
227    Cobs=eye(4)
228    Dobs=zeros(4,4) %Es 4 porque son 2 de los estados y 2 de las entradas
229
230 end
231
232 %% CONTROL CON ACCIÓN INTEGRAL
233
234 if with_integral
235    fprintf('Se calcula el integral \n')
236
237    %Polos del integrador
238    n_veces_int = 10; %Veces que alejamos los polos extra de los dominantes
239    if with_lqr
240       %Polos para el control integral basados en el lqr
241       %poloInt = [polosLClqr(1);polosLClqr(2) polosLClqr(3) polosLClqr(4)]
242       polosInt = [polosLClqr;
243                   real(polosLClqr(1))*n_veces_int;
244                   real(polosLClqr(1))*n_veces_int]
245
246    else
247       %Polos para el control integral basados en RE
248       %poloInt = [pd1 pd2 pd3 pd4]
249       polosInt = [pd;
250                   pd(1)*n_veces_int;
251                   pd(2)*n_veces_int]
252    end
253
254    %Método para la acción integral según el Nise
255    fprintf('Integral según el Nise \n')
256
257    %Sistema aumentado
258    Aa = [Alin zeros(4,2);-Clin zeros(2,2)];
259    Ba = [Blin; zeros(2,2)];
260    Ea = [zeros(4,2);eye(2)];
261    Ca = [Clin zeros(2,2)];
262    Da = [Dlin zeros(2,2)];
263    sys_aumentado = ss(Aa,[Ba Ea],Ca,Da);
264
265    Ka = place(Aa,Ba,polosInt);
266    Krsi = [Ka(1,1),Ka(1,2),Ka(1,3),Ka(1,4);Ka(2,1),Ka(2,2),Ka(2,3),Ka(2,4)]
267    %de la retro de estados
268    Ki = -[Ka(1,5),Ka(1,6);Ka(2,5),Ka(2,6)] %para la acción integral
269 end
270
271 %% DISCRETIZACIÓN DEI CONTROL RE
272 fprintf('Discretización de los controladores: \n')
273 %Tiempo de asentamiento para todos
274 Ts = 0.016 %[s] (0.01 para que funcionen las gráficas)
275
276 fprintf('Control RE discreto \n')
```

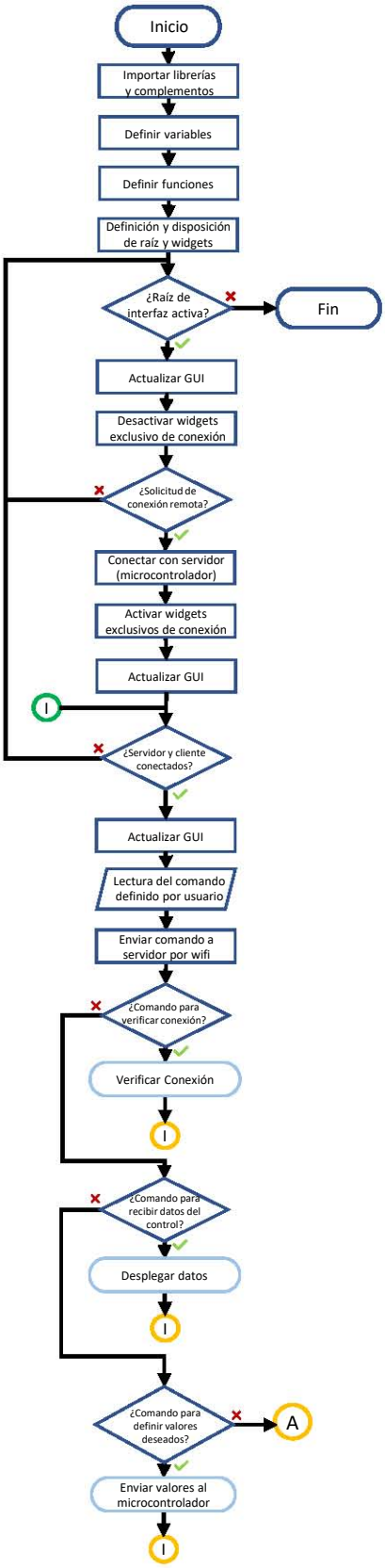
```
277 %Sistema lineal a discreto
278 sys_lin_disc = c2d(ss(Alin,Blin,Clin,Dlin),Ts);
279 Alin_disc = sys_lin_disc.A;
280 Blin_disc = sys_lin_disc.B;
281 Clin_disc = sys_lin_disc.C;
282 Dlin_disc = sys_lin_disc.D;
283
284 %Para los polos discretos
285 if with_lqr
286     %Utiliza los polos de lazo cerrado del lqr
287     polos_disc = exp(polosLClqr*Ts)
288 else
289     %Utiliza los polos de la ubicación de polos
290     polos_disc = exp(pd*Ts)
291 end
292
293 K_disc = place(Alin_disc,Blin_disc,polos_disc)
294 K0_disc = inv(Clin_disc*inv(-Alin_disc+Blin_disc*K_disc)*Blin_disc)
295
296 %% DISCRETIZACIÓN DEL OBSERVADOR
297 fprintf('Observador (Lazo cerrado) discreto \n')
298
299 if with_observer
300     if with_lqr
301         pobs_disc = exp(real(polosLClqr)*n_veces_obs*Ts)
302     else
303         pobs_disc = exp(real(pd)*n_veces_obs*Ts)
304     end
305
306     MatrizLT_disc=place(Alin_disc',Clin_disc',pobs_disc);
307     Lobs_disc=MatrizLT_disc';
308
309     %realizamos las matrices A y B del observador
310     %Lazo cerrado
311     Aobs_disc=Alin_disc-Lobs_disc*Clin_disc;
312     Bobs_disc=[Lobs_disc Blin_disc];
313     Cobs_disc=eye(length(Alin_disc));
314     Dobs_disc=zeros(4,4);
315 end
316
317 %% DISCRETIZACIÓN DEL CONTROL RE+INT
318
319 fprintf('Control RE + int discreto \n')
320
321 if with_integral
322
323     %Sistema aumentado
324     Aa_disc = [Alin_disc Blin_disc;zeros(2,4) zeros(2,2)];
325     Ba_disc= [zeros(4,2); eye(2,2)];
326
327     if with_lqr
328       polosInt_disc = exp(polosInt.*Ts)
329
330     else
331       polosInt_disc = exp(polosInt.*Ts)
332     end
333
334     %polosInt_disc = [exp([-1+0.5929j;-1-0.5929j;-10+5.929j;-10-5.929j;-20;-20]*Ts)]
335     %Puede ser calculado con comando place
336     Ka_disc = place(Aa_disc,Ba_disc,polosInt_disc)
337
338     %Adaptación para valores discretos
339     Km = [(Alin_disc - eye(4)), Blin_disc;
340           (Clin_disc*Alin_disc), (Clin_disc*Blin_disc)];
341     In = [zeros(2,4) eye(2)];
342     Kaux_disc = (Ka_disc+In)/Km;
343
344     %Calcular las matrices de ganancias discretas
345     Krsi_disc = [Kaux_disc(1,1),Kaux_disc(1,2),Kaux_disc(1,3),Kaux_disc(1,4);
```

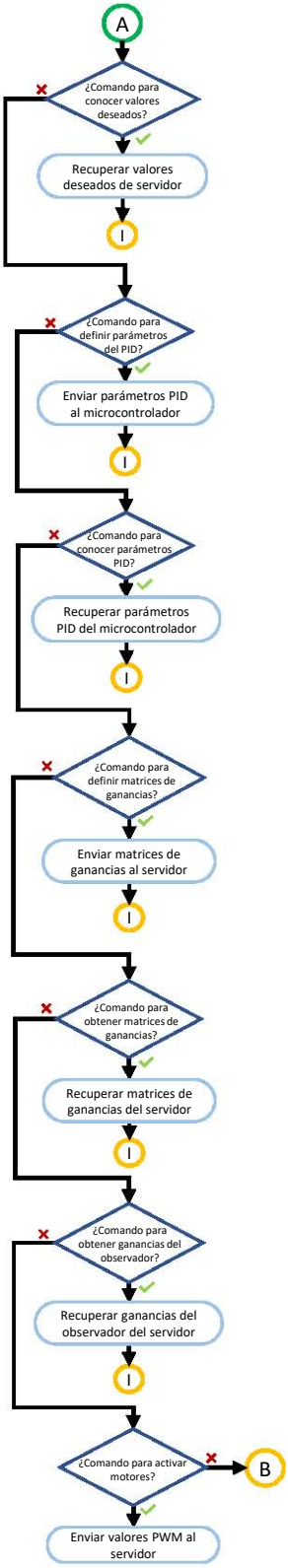
```
346                 Kaux_disc(2,1),Kaux_disc(2,2),Kaux_disc(2,3),Kaux_disc(2,4)]
347     Ki_disc = [Kaux_disc(1,5),Kaux_disc(1,6);Kaux_disc(2,5),Kaux_disc(2,6)]
348 end
349
350 %% PARA OBTENER LAS MATRICES LISTAS PARA PEGAR EN LA INTERFAZ
351 % fprintf('Para copiar y pegar en la interfaz \n')
352 %
353 Kstring = K_disc;
354 K0string = K0_disc;
355 Krsistring = Krsi_disc;
356 Kistring = Ki_disc;
357 Aobsstring = Aobs_disc;
358 Bobsstring = Bobs_disc;
359 MatrizKString = sprintf('[[%0.9f,%0.9f,%0.9f,%0.9f],[%0.9f,%0.9f,%0.9f,%0.9f]]',Kstring')
360 MatrizK0String = sprintf('[[%0.9f,%0.9f],[%0.9f,%0.9f]]',K0string')
361 MatrizKrsiString = sprintf('[[%0.9f,%0.9f,%0.9f,%0.9f],[%0.9f,%0.9f,%0.9f,%0.9f]]',Krsistring')
362 MatrizKiString = sprintf('[[%0.9f,%0.9f],[%0.9f,%0.9f]]',Kistring')
363 MatrizAobsString = sprintf('[[%0.9f,%0.9f,%0.9f,%0.9f],[%0.9f,%0.9f,%0.9f,%0.9f],[%0.9f,%0.9f,%0.9f,%0.9f],[%0.9f,%0.9f,%0.9f,%0.9f]]',Aobsstring')
364 MatrizBobsString = sprintf('[[%0.9f,%0.9f,%0.9f,%0.9f],[%0.9f,%0.9f,%0.9f,%0.9f],[%0.9f,%0.9f,%0.9f,%0.9f],[%0.9f,%0.9f,%0.9f,%0.9f]]',Bobsstring')
365 %
366
367
368 %% Simulación para el control discreto RS
369 %Simulación del algoritmo que se mete al microcontrolador
370 %Par estar seguros de como se debe programar
371
372 %Numero de interacciones
373 nit=1000;
374
375 %Condiciones iniciales
376 u(1,1:nit) = 0;
377 u(2,1:nit) = 0;
378 %Saturación de los motores
379 umax = 0.018465*1000-10.0694;
380 umin = 0.018465*500-10.0694;
381
382 ym(1,1:nit) = c_i_h-x1barra;
383 ym(2,1:nit) = c_i_theta-x3barra;
384
385 v(1,1:nit) = 0;
386 v(2,1:nit) = 0;
387
388 r(1,1:nit) = h_deseada;
389 r(2,1:nit) = theta_deseada;
390 % r(1,500:nit) = 0.2; %Cambio de referencia
391 % r(2,500:nit) = deg2rad(10);
392
393 Xm(1,1:nit) = c_i_h-x1barra;
394 Xm(2,1:nit) = c_i_hp-x2barra;
395 Xm(3,1:nit) = c_i_theta-x3barra;
396 Xm(4,1:nit) = c_i_thetap-x4barra;
397
398 Xm_obs(1:4,1:nit) = 0;
399
400
401 for k=3:nit
402     %Planta
403     Xm(:,k) = Alin_disc*Xm(:,k-1) + Blin_disc*u(:,k-1);
404     ym(:,k) = Clin_disc*Xm(:,k-1)+[x1barra;x2barra];
405
406     %Observador
407     Xm_obs(:,k) = Aobs_disc*Xm_obs(:,k-1) + Bobs_disc*[ym(:,k)-[x1barra;x3barra];u(:,k-1)];
408
409     %Ley de Control
410     u(:,k) = -K_disc*Xm_obs(:,k) + K0_disc*(r(:,k)-[x1barra;x3barra]);
411
412     if u(1,k) > umax
413         u(1,k) = umax;
414         %v(k) = 100;
```

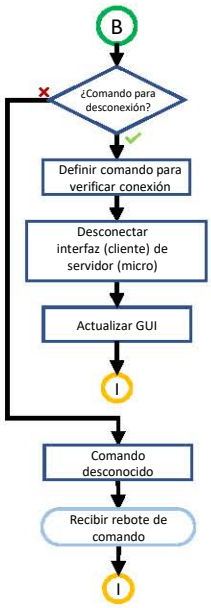
```
415         else if u(1,k)<umin
416                 u(1,k) = umin;
417             end
418         end
419         if u(2,k) > umax
420             u(2,k) = umax;
421             %v(k) = 100;
422         else if u(2,k)<umin
423                 u(2,k) = umin;
424             end
425         end
426
427 end
428 t = 0:Ts:(nit-1)*Ts;
429 figure(1)
430 subplot(2,1,1)
431 stairs(t,r(1,:),'--g','Linewidth',1),hold on
432 stairs(t,r(2,:),'--m','Linewidth',1)
433 stairs(t,ym(1,:),'-y','Linewidth',1)
434 stairs(t,ym(2,:),'-b','Linewidth',1)
435 title('Simulación control RS')
436 xlabel('Tiempo (s)');
437 ylabel('Salidas (metros y rad)');
438 legend('h_d','theta_d','h','theta','Location','SouthEast')
439 grid on;
440 hold
441 subplot(2,1,2)
442 stairs(t,u(1,:),'y','Linewidth',1),hold on
443 stairs(t,u(2,:),'b','Linewidth',1)
444 xlabel('Tiempo (s)');
445 ylabel('Control');
446 legend('u')
447 grid on;
448
449
450 %% Simulación para el control disceto RS + int
451 %Simulación del algoritmo que se mete al microcontrolador
452 %Para estar seguro de como se debe programar
453
454 %Numero de interacciones
455 nit=1000;
456
457 %Condiciones iniciales
458 u(1,1:nit) = 0;
459 u(2,1:nit) = 0;
460 %Saturación de los motores
461 umax = 0.018465*1000-10.0694;
462 umin = 0.018465*500-10.0694;
463
464 ym(1,1:nit) = c_i_h-x1barra;
465 ym(2,1:nit) = c_i_theta-x3barra;
466
467 v(1,1:nit) = 0;
468 v(2,1:nit) = 0;
469
470 r(1,1:nit) = h_deseada;
471 r(2,1:nit) = theta_deseada;
472 r(1,500:nit) = 0.2; %Cambio de referencia
473 r(2,500:nit) = deg2rad(10);
474
475 Xm(1,1:nit) = c_i_h-x1barra;
476 Xm(2,1:nit) = c_i_hp-x2barra;
477 Xm(3,1:nit) = c_i_theta-x3barra;
478 Xm(4,1:nit) = c_i_thetap-x4barra;
479
480 Xm_obs(1:4,1:nit) = 0;
481
482
483 for k=3:nit
```

```
484       %Planta
485       Xm(:,k) = Alin_disc*Xm(:,k-1) + Blin_disc*u(:,k-1);
486       ym(:,k) = Clin_disc*Xm(:,k-1)+[x1barra;x2barra];
487
488       %Observador
489       Xm_obs(:,k) = Aobs_disc*Xm_obs(:,k-1) + Bobs_disc*[ym(:,k)-[x1barra;x3barra];u(:,k-1)];
490
491       %Integrador
492       v(:,k)=v(:,k-1)+(r(:,k)-[x1barra;x3barra])-(ym(:,k)-[x1barra;x3barra]);
493       if v(1,k) > 40
494          v(1,k) = 40;
495       end
496       if v(1,k)< -40
497          v(1,k) = -40;
498       end
499
500       if v(2,k) > 40
501          v(2,k) = 40;
502       end
503       if v(2,k)< -40
504          v(2,k) = -40;
505       end
506 %
507       %Ley de Control
508       u(:,k) = -Krsi_disc*Xm_obs(:,k) + Ki_disc*v(:,k);
509      if u(1,k) > umax
510          u(1,k) = umax;
511           %v(k) = 100;
512       else if u(1,k)<umin
513               u(1,k) = umin;
514           end
515       end
516       if u(2,k) > umax
517          u(2,k) = umax;
518           %v(k) = 100;
519       else if u(2,k)<umin
520               u(2,k) = umin;
521           end
522       end
523
524 end
525 t = 0:Ts:(nit-1)*Ts;
526 figure(2)
527 subplot(2,1,1)
528 stairs(t,r(1,:),'--g','Linewidth',1),hold on
529 stairs(t,r(2,:),'--m','Linewidth',1)
530 stairs(t,ym(1,:),'-y','Linewidth',1)
531 stairs(t,ym(2,:),'-b','Linewidth',1)
532 title('Simulación control RS + int')
533 xlabel('Tiempo (s)');
534 ylabel('Salidas (metros y rad)');
535 legend('h_d','theta_d','h','theta','Location','SouthEast')
536 grid on;
537 hold
538 subplot(2,1,2)
539 stairs(t,u(1,:),'y','Linewidth',1),hold on
540 stairs(t,u(2,:),'b','Linewidth',1)
541 xlabel('Tiempo (s)');
542 ylabel('Control');
543 legend('u')
544 grid on;
545
```

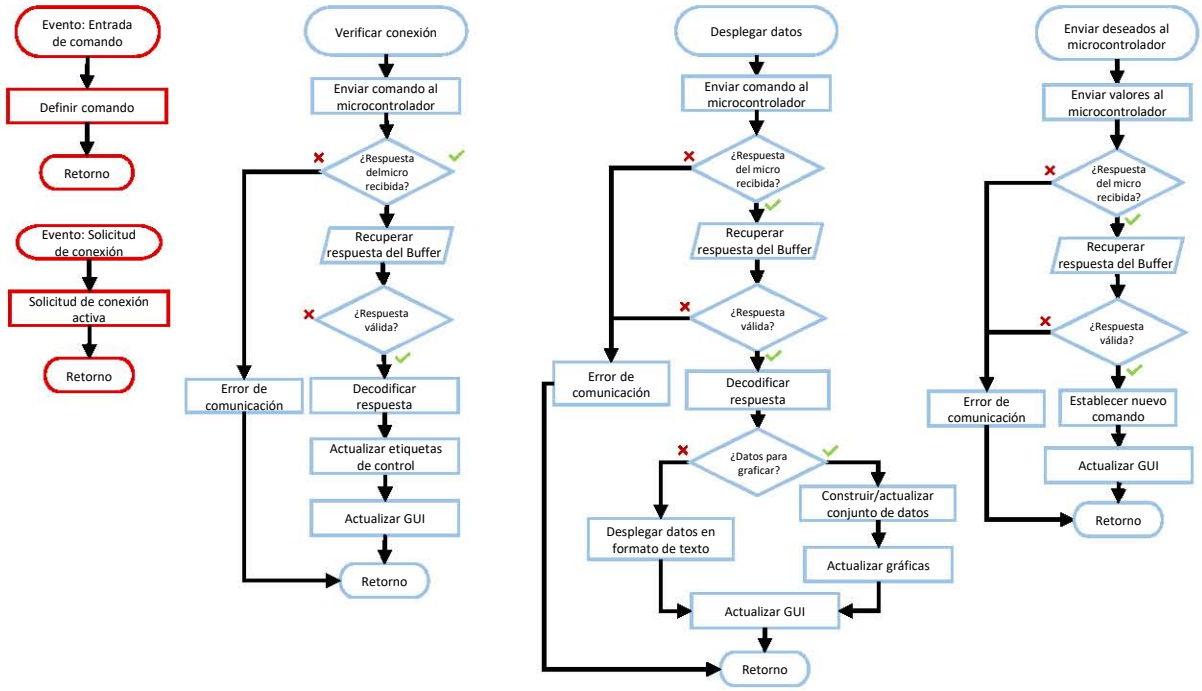
G. Diagrama de flujo: Interfaz

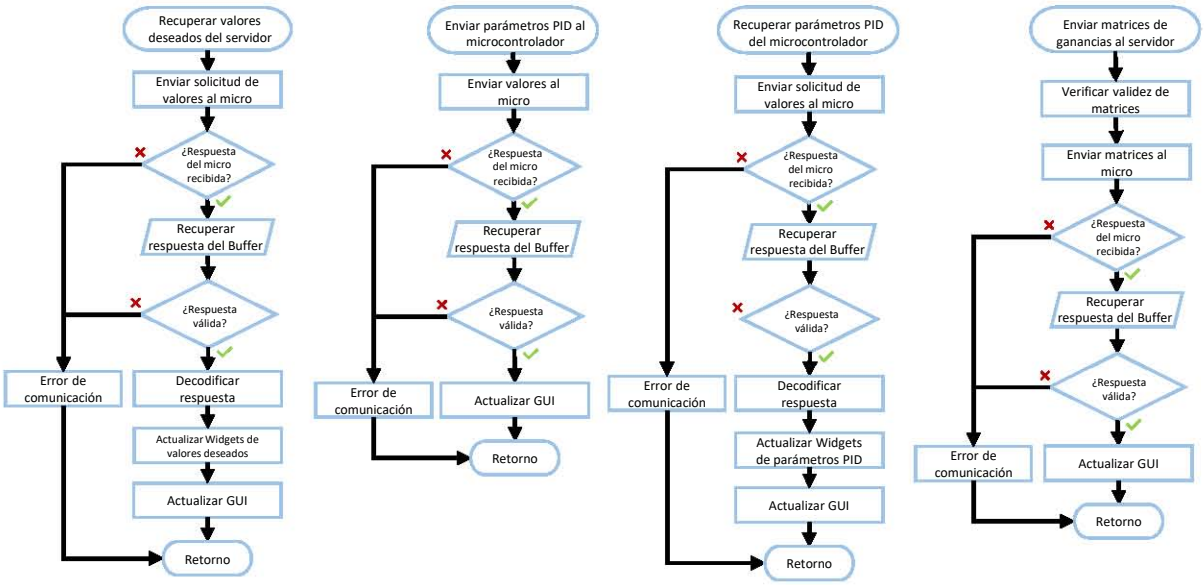


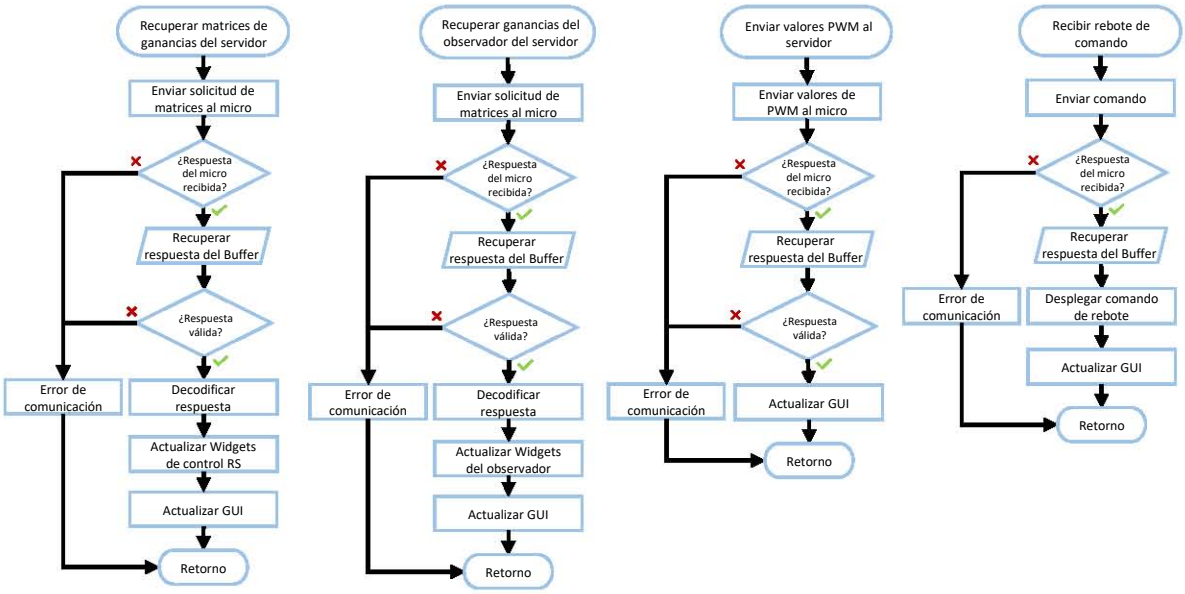




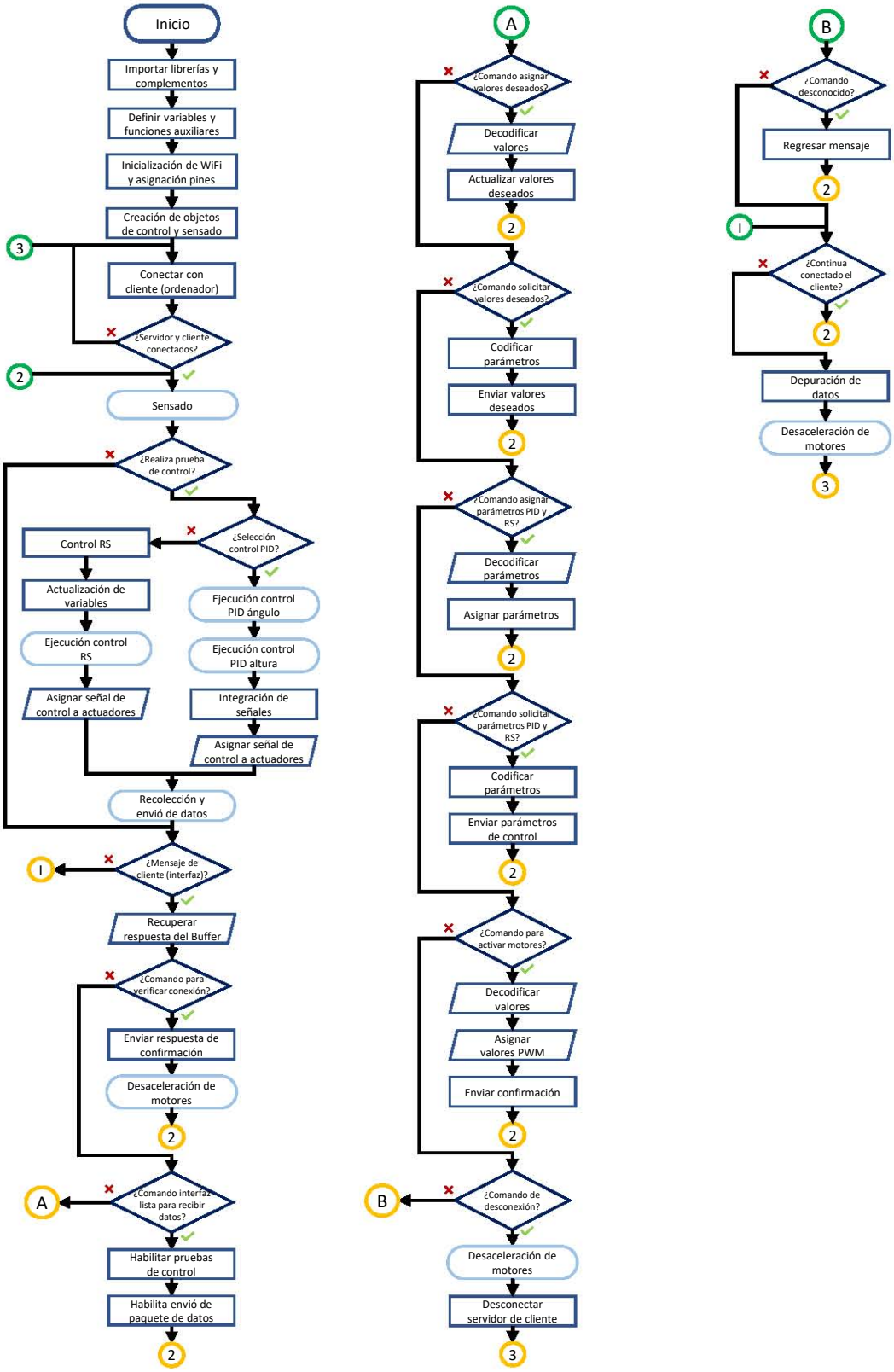
Nota: La definición del comando, la solicitud de conexión y algunas otras funciones se ejecutan como EVENTOS o EXEPCIONES cuando el usuario interactúa con la interfaz en cualquier momento del programa

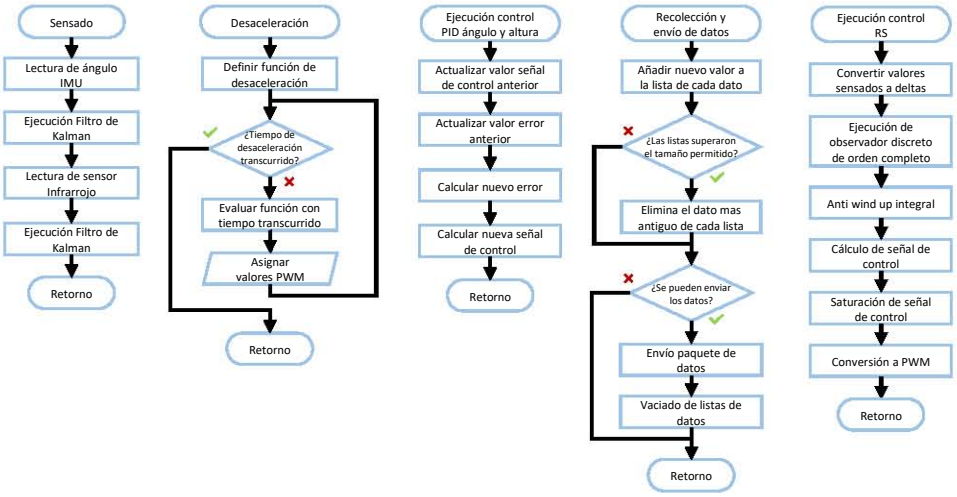

Evento: Entrada de comando
Definir comando
Retorno
Evento: Solicitud de conexión
Solicitud de conexión activa
Retorno
Verificar conexión
Enviar comando al microcontrolador
¿Respuesta delmicro recibida?
Recuperar respuesta del Buffer
¿Respuesta válida?
Error de comunicación
Decodificar respuesta
Actualizar etiquetas de control
Actualizar GUI
Retorno
Desplegar datos
Enviar comando al microcontrolador
¿Respuesta del micro recibida?
Recuperar respuesta del Buffer
¿Respuesta válida?
Error de comunicación
Decodificar respuesta
¿Datos para graficar?
Construir/actualizar conjunto de datos
Desplegar datos en formato de texto
Actualizar gráficas
Actualizar GUI
Retorno
Enviar deseados al microcontrolador
Enviar valores al microcontrolador
¿Respuesta del micro recibida?
Recuperar respuesta del Buffer
¿Respuesta válida?
Error de comunicación
Establecer nuevo comando
Actualizar GUI
Retorno


Recuperar valores deseados del servidor
Enviar solicitud de valores al micro
¿Respuesta del micro recibida?
Recuperar respuesta del Buffer
¿Respuesta válida?
Error de comunicación
Decodificar respuesta
Actualizar Widgets de valores deseados
Actualizar GUI
Retorno
Enviar parámetros PID al microcontrolador
Enviar valores al micro
¿Respuesta del micro recibida?
Recuperar respuesta del Buffer
¿Respuesta válida?
Error de comunicación
Actualizar GUI
Retorno
Recuperar parámetros PID del microcontrolador
Enviar solicitud de valores al micro
¿Respuesta del micro recibida?
Recuperar respuesta del Buffer
¿Respuesta válida?
Error de comunicación
Decodificar respuesta
Actualizar Widgets de parámetros PID
Actualizar GUI
Retorno
Enviar matrices de ganancias al servidor
Verificar validez de matrices
Enviar matrices al micro
¿Respuesta del micro recibida?
Recuperar respuesta del Buffer
¿Respuesta válida?
Error de comunicación
Actualizar GUI
Retorno


Recuperar matrices de ganancias del servidor
Enviar solicitud de matrices al micro
¿Respuesta del micro recibida?
Recuperar respuesta del Buffer
¿Respuesta válida?
Error de comunicación
Decodificar respuesta
Actualizar Widgets de control RS
Actualizar GUI
Retorno
Recuperar ganancias del observador del servidor
Enviar solicitud de matrices al micro
¿Respuesta del micro recibida?
Recuperar respuesta del Buffer
¿Respuesta válida?
Error de comunicación
Decodificar respuesta
Actualizar Widgets del observador
Actualizar GUI
Retorno
Enviar valores PWM al servidor
Enviar valores de PWM al micro
¿Respuesta del micro recibida?
Recuperar respuesta del Buffer
¿Respuesta válida?
Error de comunicación
Actualizar GUI
Retorno
Recibir rebote de comando
Enviar comando
¿Respuesta del micro recibida?
Recuperar respuesta del Buffer
Error de comunicación
Desplegar comando de rebote
Actualizar GUI
Retorno

H. Diagrama de flujo: Microcontrolador




Sensado
Lectura de ángulo IMU
Ejecución Filtro de Kalman
Lectura de sensor Infrarrojo
Ejecución Filtro de Kalman
Retorno
Desaceleración
Definir función de desaceleración
¿Tiempo de desaceleración transcurrido?
Evaluar función con tiempo transcurrido
Asignar valores PWM
Retorno
Ejecución control PID ángulo y altura
Actualizar valor señal de control anterior
Actualizar valor error anterior
Calcular nuevo error
Calcular nueva señal de control
Retorno
Recolección y envío de datos
Añadir nuevo valor a la lista de cada dato
¿Las listas superaron el tamaño permitido?
Elimina el dato mas antiguo de cada lista
¿Se pueden enviar los datos?
Envío paquete de datos
Vaciado de listas de datos
Retorno
Ejecución control RS
Convertir valores sensados a deltas
Ejecución de observador discreto de orden completo
Anti wind up integral
Cálculo de señal de control
Saturación de señal de control
Conversión a PWM
Retorno

I. Micropython: programa principal del microcontrolador


```
#Victor Romero
#Código para el servicio social - Tésis, Dron
#Derechos reservados
"""
Este programa es para comunicar la computadora con el microcontrolador
a traves de comunicacion TCP utilizando wifi.

El microcontrolador se configurara como el servidor y la computadora (Con la interfaz)
como el cliente.

Depende de los comandos recibidos se envía una respuesta distinta al cliente

De forma paralela simempre se realiza el control del dispositivo

El control puede ser RS o PID según lo especificado


Este programa interactua con "Interfaz_Dronazo_V9"
"""

print("Ejecutando el programa micro_evolution_interfaz_V9.py")

from machine import Pin,I2C,ADC,PWM,freq #para poder definir los pines y sus funciones
import network #Para establecer la conexion WLAN
import socket #Para poder enviar y recibir datos tipo TCP
import select #para verificar si existen datos listos para leer en el buffer
import utime #Para las medidas de tiempo
import Coconut_Functions as CF #Script con los métodos de control y sus funciones
import Watermelon_var as wmv #Script con los valores calculados del control (No se deben modificar)
import IMU_BoschSensorTec #Script con las funciones necesarias para leer la IMU bosch IBM085
import math
import minibasicnumpy as npb
import gc #Para evitar "memory allocation failed"

#Para operaciones de matrices
#import minibasicnumpy as npb
#import Vic_IMU #Para la lectura de la imu (módulo personalizado)
##import IMU_BoschSensorTec
#import urandom
#import time


freq(240000000) # Seleccionamos la frecuencia de funcionamiento
gc.enable() #enable garbage collection

#Led que indica que el programa del micro funciona correctamente.
led_status = Pin(2,Pin.OUT) #Pin 2 es el Led interno azul
led_status.on()

#----------------------INICIALIZACIÓN DE VALORES---------------------------

pwm_arming = 500 #Valor de pwm (para el ppm con 1[ms] por pulso)
pwm_inicial = 500

#Inicializamos estos valores para la primer iteración del control
pwm = [[500],[500]]
edos_estimados = [[0],[0],[0],[0]]

#---VARIABLES PARA LA LECTURA DE DATOS-----

angulos = [0.0,0.0] #roll y pitch [rad]
altura = 0.0 #distancia telemetro [m]

#Es necesario llevar el tiempo de ejecución por ciclo para la IMU y el control
tinit = 0.0; #tiempo de inicio de lapso de tiempo en [s]
tstop = 0.0; #tiempo final del lapso de tiempo en [s]
telapsed = 0.0; #diferencia de tiempo entre las lecturas en [s]

#Esta variable es para auxiliar la graficación.
#tamaño de los paquetes de datos para graficacion
len_pack= 14 #longitud del paquete de datos (Por alguna razón la interfaz no aguanta más de 17)
             #Todo funciona bonito en 10
pack_requested = False #Si se solicitó el envió de datos

#En caso de que pase más de un ciclo sin que se soliciten datos. esta variable
#lleva el conteo de tiempo entre cada solicitud de datos (entre cada lectura).[s]
tacumulated = 0.0;
#se decidio enviar los datos por paquetes para mas resolucion.
global pack_telapsed, pack_angulo, pack_altura, pack_edo1, pack_edo2, pack_edo3, pack_edo4
global pack_pwm2, pack_pwm1, pack_ref_altura, pack_ref_angulo

pack_telapsed=[]
pack_angulo=[]
pack_altura=[]
pack_edo1=[]
pack_edo2=[]
pack_edo3=[]
pack_edo4=[]
pack_pwm2=[]
pack_pwm1=[]
pack_ref_altura=[]
pack_ref_angulo=[]

#Como el empaquetado y el envió se hacen en dos lugares diferentes esta bandera
#avisa cuando la interfaz está lista para recibir la info.
```

```
96  global send_pack_flag
97  send_pack_flag = False
98
99  #LLeva el conteo del tiempo sin respuesta, para desconexión por time-out
100 toutini = 0.0 #Inicio del timeout
101
102 #Variable que le permite hacer o no el control
103 control_test = False
104
105 #Variable para saber que control se desea ejecutar
106 which_control = 0 #0 -> PID  1 -> RS
107
108 #Variable para saber si el control es integral en el RS
109 integral_control = False
110
111 #Variable para saber si el observador es de espacio de estados o kalman en el RS
112 isStateSpace = True
113
114 #------------------------- FUNCIONES AUXILIARES -----------------------------
115
116 #Funcion para mandar la tanda de datos
117 def SendDataPack(dataPack):
118     #mandar los datos a la interfaz
119     #%data#tiempo[ms]#angulo[°]#altura[m]#refaltura[m]#refangulo[°]#pwmderecho#pwmizquierdo#edo1#edo2#edo3#edo4
120     #arma el string de respuesta y lo convierte en bytes
121     gc.collect()
122     msg_send = "%data#{}#{}#{}#{}#{}#{}#{}#{}#{}#{}".format(dataPack[0],
123                                                             dataPack[1],
124                                                             dataPack[2],
125                                                             dataPack[3],
126                                                             dataPack[4],
127                                                             dataPack[5],
128                                                             dataPack[6],
129                                                             dataPack[7],
130                                                             dataPack[8],
131                                                             dataPack[9],
132                                                             dataPack[10])
133     #print(msg_send)
134     ret = cliente.send(bytes(msg_send,"utf-8"))
135
136
137 #Función para armar el paquete de datos para graficación
138 def BuildDataPack(data):
139     #data = [permiso de enviar, t,angulo,altura,v_angulo,v_altura,pwmD,pwmI,error_altura,error_angulo]
140     global pack_telapsed, pack_angulo, pack_altura, pack_edo4, pack_edo2, pack_edo1, pack_edo3
141     global pack_pwm2, pack_pwm1, pack_ref_altura, pack_ref_angulo
142     global send_pack_flag
143     #empaquetamiento de los datos especifico para una graficación con alta resolución
144     #vamos a empaquetar los datos de cada ciclo del control
145     #Se empaquetan los datos
146
147     #permiso de enviar,[t,angulo,altura,refaltura,refangulo,pwmD,pwmI,edo1,edo2,edo3,edo4]
148
149     #En teoría los datos de entrada no se necesitan hacer globales
150     pack_telapsed.append(data[0])
151     pack_angulo.append(data[1])
152     pack_altura.append(data[2])
153     pack_ref_altura.append(data[3])
154     pack_ref_angulo.append(data[4])
155     pack_pwm2.append(data[5])
156     pack_pwm1.append(data[6])
157     pack_edo1.append(data[7])
158     pack_edo2.append(data[8])
159     pack_edo3.append(data[9])
160     pack_edo4.append(data[10])
161
162
163
164     #Para controlar la cantidad de datos acumulados y que no se sature el micro
165     if len(pack_telapsed) > len_pack:
166         del pack_telapsed[0]
167         del pack_angulo[0]
168         del pack_altura[0]
169         del pack_ref_altura[0]
170         del pack_ref_angulo[0]
171         del pack_pwm2[0]
172         del pack_pwm1[0]
173         del pack_edo1[0]
174         del pack_edo2[0]
175         del pack_edo3[0]
176         del pack_edo4[0]
177
178
179     #Si la interfaz está lista para recibir datos se envían
180     if send_pack_flag:
181         #print("pack size", len(pack_ref_angulo))
182         try:
183             SendDataPack([pack_telapsed,pack_angulo,pack_altura,pack_ref_altura,
184                           pack_ref_angulo, pack_pwm2,pack_pwm1, pack_edo1,
185                           pack_edo2, pack_edo3, pack_edo4])
186         except:
187             print("La interfaz no estaba lista para recibir datos o falló la memoria")
188
189         #Una vez que se envían se vacian para la siguiente ronda de paquetes
190         pack_telapsed=[]
191         pack_angulo=[]
```

```
        pack_altura=[]
        pack_ref_altura=[]
        pack_ref_angulo=[]
        pack_pwm2=[]
        pack_pwm1=[]
        pack_edo1=[]
        pack_edo2=[]
        pack_edo3=[]
        pack_edo4=[]


        #La información ya se envió y espera la siguiente confirmación
        #Se actualiza la bandera
        send_pack_flag = False


#--------------------------  PROGRAMA PRINCIPAL  ------------------------------

#ACTIVAR EL MODULO WIFI
ap = network.WLAN(network.AP_IF) #objeto network configurado como access point
ap.active(True) #Activamos la conexión
ap.config(essid = wmv.SSID, password = wmv.PASSWORD) #configuramos nombre y contraseña
#Se opbitnen de watermelon

#La direccion IP por default es 192.168.4.1
#Se puede comprrobar con ap.ifconfig()

#ACTIVAR IMU
#Comunicación I2C
imu = I2C(scl = Pin(23),sda = Pin(22))#EN MICRO ESP 32
IMU_BoschSensorTec.Init(imu) #Despertar la IMU

#Para el sensor de distancia
entrada_ADC=ADC(Pin(34))#EN MICRO ESP32
#Atenuamos la señal de entrada. El micro solo aguanta señales de hasta 1[V]
#Se atenúa para que logre reconocer señales de hasta 3.3[V]
entrada_ADC.atten(ADC.ATTN_11DB)

#Pines para motores, frecuancia de pwm
motor_derecho = PWM(Pin(18)) #D1
motor_derecho.freq(500)
motor_derecho.duty(500)
motor_izquierdo = PWM(Pin(19)) #D2
motor_izquierdo.freq(500)
motor_izquierdo.duty(500)

#Se usa el protocolo PPM. Duty max --> 50 - 103
#Con eso se consiguen pulsos de entre [1 - 2][ms]
#En la práctica está alrrededor de 54 - 90

#Inicializamos objeto de control
#controlazoRS = CF.CoconutRS_4n()
controlazoRS = CF.CoconutRS_4n_disc()

controlazoPID_angulo = CF.CoconutPID_nokick(kp = wmv.kp_theta, taui = wmv.taui_theta, taud = wmv.taud_theta,
                            ts = wmv.ts, ref = wmv.theta_deseada,
                            maximo = CF.PWM2F(900) - wmv.F1barra,
                            minimo = -1*(wmv.F1barra - CF.PWM2F(520)))

controlazoPID_altura = CF.CoconutPID_nokick(kp = wmv.kp_h, taui = wmv.taui_h, taud = wmv.taud_h,
                            ts = wmv.ts, ref = wmv.h_deseada,
                            maximo = CF.PWM2F(900) - wmv.F1barra,
                            minimo = -1*(wmv.F1barra - CF.PWM2F(520)))

# controlazoPID_angulo = CF.CoconutPID(kp = wmv.kp_theta, taui = wmv.taui_theta, taud = wmv.taud_theta,
#                             ts = wmv.ts, ref = wmv.theta_deseada,
#                             maximo = CF.PWM2F(900) - wmv.F1barra,
#                             minimo = -1*(wmv.F1barra - CF.PWM2F(520)))
#
# controlazoPID_altura = CF.CoconutPID(kp = wmv.kp_h, taui = wmv.taui_h, taud = wmv.taud_h,
#                             ts = wmv.ts, ref = wmv.h_deseada,
#                             maximo = CF.PWM2F(900) - wmv.F1barra,
#                             minimo = -1*(wmv.F1barra - CF.PWM2F(520)))

# controlazoPID_altura = CF.CoconutPID(kp = wmv.kp_h, taui = wmv.taui_h, taud = wmv.taud_h,
#                             ts = wmv.ts, ref = wmv.h_deseada,
#                             maximo = CF.PWM2F(90) - wmv.F1barra,
#                             minimo = -1*(wmv.F1barra - CF.PWM2F(54)),
#                             compensador = lambda x,y: x*(2-math.cos(y)))

sensores = CF.Sensores()

while not ap.active():
    pass #Esperamos a que se active correctamente el modulo wifi

print ("Configuración Wifi access point exitosa: {}".format(ap.ifconfig()))


try:
    #Ahora activamos los socket que son los que permiten la comunicacion TCP
    addr = socket.getaddrinfo('0.0.0.0',80)[0][-1] #direccion IP y puerto (80)
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #se seleciona protocolo y tipo(ipv4,TCP)
    s.bind(addr) #asigna la direccion al socket
    s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)#Se puede reusar el socket cuando se desconecta
    s.listen(1) #Solo se puede conectar un cliente a la vez

```

```
288
289     #INICIA CICLO INFINITO DEL PROGRAMA
290     while True:
291         print("Escuchando por nuevo cliente")
292         #En la siguiente linea se queda atorado hasta que se conecta un cliente
293         cliente,caddr = s.accept() #aceptamos la conexion con el cliente
294         cliente.setblocking(0) #para que no se bloquee el socket durante la lectura
295         #cliente es un nuevo socket, caddr es la direccion del cliente
296         print("Conexion establecida con",caddr)
297         tinit = utime.ticks_ms()
298
299         while True:
300
301             #Se cuenta el tiempo que llevó el ciclo anterior
302             tstop = utime.ticks_ms()
303             telapsed = (utime.ticks_diff(tstop,tinit))/1000 # tiempo entre iteraciónes en [s]
304             #Para la primer iteración donde el tiempo pasado es cero
305             if telapsed == 0:
306                 telapsed+=0.001
307             tinit = utime.ticks_ms()
308
309             #print(telapsed)
310
311             #Acutalizamos el tiempo entre cada solicitud de datos
312             tacumulated = tacumulated + telapsed
313
314             # ------------------------ SENSADO -----------------------------
315
316             #Primero el sensado (el angulo es negativo)
317             #Regresa la altura en mentros y los angulos en grados
318             #altura, angulos = CF.Sensado(entrada_ADC,imu,angulos,telapsed)
319             altura, angulos = sensores.Sense(entrada_ADC,imu,telapsed)
320             #print(altura,-angulos[0])
321
322
323             ###################################################################
324             # --------------------- CICLO DE CONTROL ------------------------
325             ###################################################################
326
327             if which_control == 0: #If PID
328
329                 if control_test:
330
331                     #Obtener la señal de control para el ángulo
332                     u_angulo = controlazoPID_angulo.Feedback(math.radians(-angulos[0]))
333
334                     #Obtener la señal de control para la altura
335                     u_altura = controlazoPID_altura.Feedback(altura)
336
337                     #Si se utilizara el modelo no lineal o el modelo real
338                     [F1,F2] = CF.PIDCombineEffects(u_altura,u_angulo,wmv.F1barra,wmv.F2barra)
339                     pwm = [[CF.F2PWM(F1)],[CF.F2PWM(F2)]]
340
341                     motor_derecho.duty(int(pwm[1][0]))
342                     motor_izquierdo.duty(int(pwm[0][0]))
343
344                      # ------------------- ESTA ÁREA ES DE COMUNICACIÓN --------------------------
345                     #permiso de enviar,[t,angulo,altura,refaltura,refangulo,pwmD,pwmI,0,0,0,0]
346                     BuildDataPack([telapsed,
347                                    -angulos[0],
348                                    altura,
349                                    controlazoPID_altura.ref,
350                                    math.degrees(controlazoPID_angulo.ref),
351                                    int(pwm[1][0]),
352                                    int(pwm[0][0]),
353                                    0,0,0,0])
354
355
356             elif which_control == 1: #If RS
357
358                 # ----------------------- CONTROL -------------------------
359                 #Solo si se solicita el control desde la interfaz
360                 if control_test: #control_test
361
362                     #El control necesita la altura en metros y los angulos en radianes
363                     #-angulo porque la imu esta alrevez, aqui para no afectar el calculo
364                     #del angulo en el programa IMU
365                     controlazoRS.UpdateValues([[altura],[math.radians(-angulos[0])]])
366
367                     #Feedback (en la primera iteración se usan las condiciones iniciales)
368                     pwm = controlazoRS.Feedback(isIntegral=integral_control) #[[pwm1],[pwm2]]
369
370                     motor_derecho.duty(int(pwm[1][0]))
371                     motor_izquierdo.duty(int(pwm[0][0]))
372
373
374                     # ------------------- ESTA ÁREA ES DE COMUNICACIÓN --------------------------
375                     #permiso de enviar,[t,angulo,altura,refaltura,refangulo,pwmD,pwmI,edo1,edo2,edo3,edo4]
376                     BuildDataPack([telapsed,
377                                    -angulos[0],
378                                    altura,
379                                    controlazoRS.h_deseada,
380                                    math.degrees(controlazoRS.theta_deseada),
381                                    int(pwm[1][0]),
382                                    int(pwm[0][0]),
383                                    controlazoRS.edos_estimados[0][0],
```

```
                                controlazoRS.edos_estimados[1][0],
                                controlazoRS.edos_estimados[2][0],
                                controlazoRS.edos_estimados[3][0]])



            #Verificamos si hay lecturas disponibles en el socket
            ready2read = select.select([cliente],[],[],0.001)#el valor era 0.001
            if ready2read[0]:
                toutini = utime.ticks_ms()
                msg =cliente.recv(1024) #recibe 1024 bytes (basta y sobra por mucho)

                if len(msg)!=0: #si la longitud del mensaje es mayor a cero lo analiza
                    #Las intrucciones que se mandan desde el cliente (computadora):
                    msg = msg.decode("utf-8") #convierte bytes en string
                    command = msg.split("#") #divide el mensaje en un arreglo

                    #Activa el Proceso de control
                    control_test = True if command[0] == "%dataCont" else False

                    if command[0] == "%dataCont":
                        send_pack_flag = True #Avisa que la interfaz está lista para recibir datos


                    #%data se usa para mandar los datos individualmente en un arreglo fuera del ciclo de control unicamente para desplegar el valor
                    elif command[0] == "%data":
                        #OJO: Solo es para el tiempo, angulo, altura, pwm1 y pwm2
                        #     Los demás datos son basura
                        #mandar los datos a la interfaz
                        #%data#tiempo[ms]#angulo[°]#altura[m]#refaltura[m]#refangulo[°]#pwmderecho#pwmizquierdo#edo1#edo2#edo3#edo4

                        #arma el string de respuesta y lo convierte en bytes
                        msg_send = "%data#{}#{}#{}#{}#{}#{}#{}#{}#{}#{}#{}".format(telapsed,-angulos[0],altura,0,0,
                                                                                   int(pwm[1][0]),int(pwm[0][0]),0,0,0,0)

                        ret = cliente.send(bytes(msg_send,"utf-8"))

                    #Para asegurarse que la conexión todavia existe
                    elif command[0] == "%SA?":

                        #Si solo se comprueba conexión y los motores están encendidos se desaceleran
                        if pwm[0][0] > pwm_inicial or pwm[0][0] > pwm_inicial:
                            CF.Desaceleracion(motor_derecho,motor_izquierdo,pwm)
                            pwm[0][0] = pwm_inicial
                            pwm[1][0] = pwm_inicial

                        msg_send = bytes("%SA#{}#{}#{}".format(which_control,integral_control,isStateSpace),"utf-8")
                        ret = cliente.send(msg_send)

                    elif command[0]== "%setdeseado":
                        #formato= %setdeseado#theta#altura
                        controlazoRS.ChangeDesired(float(command[2]),math.radians(float(command[1])))
                        controlazoPID_angulo.ChangeDesired(math.radians(float(command[1])))
                        controlazoPID_altura.ChangeDesired(float(command[2]))
                        print("altura: {}, angulo: {}".format(controlazoRS.h_deseada,
                                                              math.degrees(controlazoRS.theta_deseada)))
                        msg_send = bytes("StillAlive","utf-8")
                        ret = cliente.send(msg_send)

                    elif command[0]== "%getdeseado":
                        #formato= %getdeseado#theta#altura
                        if which_control == 0: #PID
                            msg_send = bytes("%getdeseado#{}#{}".format(math.degrees(controlazoPID_angulo.ref),
                                                                        controlazoPID_altura.ref),"utf-8")
                        elif which_control == 1: #RS4n
                            msg_send = bytes("%getdeseado#{}#{}".format(math.degrees(controlazoRS.theta_deseada),
                                                                        controlazoRS.h_deseada),"utf-8")
                        ret = cliente.send(msg_send)
                        print("Enviados los valores deseados.")

                    elif command[0] == "%pwm":
                        #modificar el pwm
                        motor_izquierdo.duty(int(command[1]))
                        motor_derecho.duty(int(command[2]))
                        #Esto entra en conflicto con la desaceleración del StillAlive
                        #pwm = [[int(command[1])],[int(command[2])]]
                        #Envía señal de que sigue vivo
                        msg_send = bytes("StillAlive","utf-8")
                        ret = cliente.send(msg_send)
                        print("PWM configurado en {},{}".format(command[1],command[2]))


                    elif command[0] == "%disconnect":
                        #desconectar al cliente
                        print("Desconexión solicitada")
                        #Si los motores están encendidos se desaceleran
                        if pwm[0][0] > pwm_inicial or pwm[0][0] > pwm_inicial:
                            CF.Desaceleracion(motor_derecho,motor_izquierdo,pwm)
                            pwm[0][0] = pwm_inicial
                            pwm[1][0] = pwm_inicial
                        break #se sale del ciclo de conexion


                    elif command[0]== "%arming":
                        #modificar el pwm
                        motor_izquierdo.duty(500)
```

```
480                     motor_derecho.duty(500)
481                     print("Se mandó señal de arming")
482                     msg_send = bytes("StillAlive","utf-8")
483                     ret = cliente.send(msg_send)
484
485                 elif command[0] == "%setpidconstants":
486                     #formato = %setpidconstants#kp_theta#taui_theta#taud_theta#kp_h#taui_h#taud_h
487                     controlazoPID_angulo.ChangeConstants(float(command[1]),float(command[2]),float(command[3]))
488                     controlazoPID_altura.ChangeConstants(float(command[4]),float(command[5]),float(command[6]))
489                     which_control = 0 #Ahora el PID es el control utilizado
490
491                     print("Constantes del PID cambiadas:  Angulo [{},{},{}], Altura [{},{},{}]".format(command[1],
492                                                                                                        command[2],
493                                                                                                        command[3],
494                                                                                                        command[4],
495                                                                                                        command[5],
496                                                                                                        command[6]))
497                     msg_send = bytes("StillAlive","utf-8")
498                     ret = cliente.send(msg_send)
499
500
501                 elif command[0] == "%getpidconstants":
502                     #formato = %getpidconstants
503                     msg_send = bytes("%getpidconstants#{}#{}#{}#{}#{}#{}".format(controlazoPID_angulo.kp,
504                                                                                  controlazoPID_angulo.taui,
505                                                                                  controlazoPID_angulo.taud,
506                                                                                  controlazoPID_altura.kp,
507                                                                                  controlazoPID_altura.taui,
508                                                                                  controlazoPID_altura.taud,),"utf-8")
509                     print("Constantes del PID enviadas")
510                     ret = cliente.send(msg_send)
511
512                 elif command[0]== "%setmat":
513                     #formato= %setmat#[[1,2,3,4],[5,6,7,8]]#[[1,2],[3,4]]
514                     #Dimensiones K[2,4],K0[2,2]
515
516                     #Ahora NO utilizamos el control integral
517                     integral_control = False
518
519                     #K= command[1:9]
520                     K= command[1].split(",")
521                     for i,c in enumerate(K):
522                         K[i]=float(c.replace("[","").replace("]",""))
523
524                     controlazoRS.matrizK=npb.array(2,4,K)
525
526                     #K0= command[9:13]
527                     K0= command[2].split(",")
528                     for i,c in enumerate(K0):
529                         K0[i]=float(c.replace("[","").replace("]",""))
530
531                     controlazoRS.matrizK0=npb.array(2,2,K0)
532                     print("Set matrix K -> {}, K0 -> {}".format(controlazoRS.matrizK,controlazoRS.matrizK0))
533                     which_control = 1 #Ahora el RS es el control utilizado
534
535                     msg_send = bytes("StillAlive","utf-8")
536                     ret = cliente.send(msg_send)
537
538                 elif command[0] == "%setmatint":
539                     #formato= %setmatint#[[1,2,3,4],[5,6,7,8]]#[[1,2],[3,4]]
540                     #Dimensiones Krsi[2,4],Ki[2,2]
541
542                     #Ahora utilizamos el control integral
543                     integral_control = True
544
545                     #Krsi= command[1:9]
546                     Krsi= command[1].split(",")
547                     for i,c in enumerate(Krsi):
548                         Krsi[i]=float(c.replace("[","").replace("]",""))
549
550                     controlazoRS.matrizKrsi=npb.array(2,4,Krsi)
551
552                     #Ki= command[9:13]
553                     Ki= command[2].split(",")
554                     for i,c in enumerate(Ki):
555                         Ki[i]=float(c.replace("[","").replace("]",""))
556
557                     controlazoRS.matrizKi=npb.array(2,2,Ki)
558                     print("Set matrix Krsi -> {}, Ki -> {}".format(controlazoRS.matrizKrsi,controlazoRS.matrizKi))
559                     which_control = 1 #Ahora el RS es el control utilizado
560
561                     msg_send = bytes("StillAlive","utf-8")
562                     ret = cliente.send(msg_send)
563
564                 elif command[0] == "%setmatobs":
565                     #formato= %setmatobs#True#[[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]]#[[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]]
566                     #Dimensiones Aobs[4,4],Bobs[4,4]
567
568                     isStateSpace = command[1] == "True"
569
570                     if isStateSpace:
571                         #Aobs = command[2:18]
572                         Aobs = command[2].split(",")
573                         for i,c in enumerate(Aobs):
574                             Aobs[i] = float(c.replace("[","").replace("]",""))
575
```

```
576                             controlazoRS.a_obs = npb.array(4,4,Aobs)
577                             print("Matriz aobs: {}".format(controlazoRS.a_obs))
578 
579                             #Bobs = command[18:34]
580                             Bobs = command[3].split(",")
581                             for i,c in enumerate(Bobs):
582                                 Bobs[i] = float(c.replace("[","").replace("]",""))
583 
584                             controlazoRS.b_obs = npb.array(4,4,Bobs)
585                             print("Matriz bobs: {}".format(controlazoRS.b_obs))
586 
587                         else:
588                             print("Observador State:", isStateSpace)
589                         msg_send = bytes("StillAlive","utf-8")
590                         ret = cliente.send(msg_send)
591 
592                     elif command[0]== "%getmat":
593                         #formato= %getmat
594                         msg_send = bytes("%getmat#{}#{}#{}#{}#{}".format(integral_control,controlazoRS.matrizK,
595                                                                          controlazoRS.matrizK0,
596                                                                          controlazoRS.matrizKrsi,
597                                                                          controlazoRS.matrizKi),"utf-8")
598                         print("Matrices del controlador enviadas")
599                         ret = cliente.send(msg_send)
600 
601                     elif command[0] == "%getmatobs":
602                         #formato = %getmatobs
603                         msg_send = bytes("%getmatobs#{}#{}#{}".format(isStateSpace,controlazoRS.a_obs,controlazoRS.b_obs),"utf-8")
604                         print("Matrices del observador enviadas")
605                         ret = cliente.send(msg_send)
606 
607 
608                     #si no se reconoce el comando solicitado lo rebota
609                     else:
610                         print("rebotando mensaje...")
611                         print(msg)
612                         msg_bounce = bytes("BOUNCE:"+msg,"utf-8")
613                         ret = cliente.send(msg_bounce)
614 
615                 #Cada que se comunica con la interfaz se reinicia el timepo del timeout
616                 #print("com time",tacumulated)
617                 tacumulated = 0
618 
619 
620             if utime.ticks_diff(utime.ticks_ms(),toutini)> 10000:
621                 #La interfaz ya no responde, salir y esperar otro cliente
622                 print("Comunicación perdida")
623 
624                 control_test = False
625 
626                 pack_telapsed=[]
627                 pack_angulo=[]
628                 pack_altura=[]
629                 pack_edo1=[]
630                 pack_edo3=[]
631                 pack_edo4=[]
632                 pack_edo2=[]
633                 pack_pwm2=[]
634                 pack_pwm1=[]
635                 pack_ref_altura=[]
636                 pack_ref_angulo=[]
637                 #Si solo se comprueba conexión y los motores están encendidos se desaceleran
638                 if pwm[0][0] > pwm_inicial or pwm[0][0] > pwm_inicial:
639                     CF.Desaceleracion(motor_derecho,motor_izquierdo,pwm,500)
640                     pwm[0][0] = pwm_inicial
641                     pwm[1][0] = pwm_inicial
642                 break
643 
644 except Exception as e:
645     led_status.off()
646     motor_derecho.duty(500)
647     motor_izquierdo.duty(500)
648 #     if(s):
649 #         s.close()
650 #
651 #     ap.disconnect()
652 #     ap.active(False)
653     print("Algo salió mal",type(e).__name__)
654 
655 except: #Si algo sale mal
656     led_status.off()
657     motor_derecho.duty(500)
658     motor_izquierdo.duty(500)
659     print("Algo salió mal")
```

J. Micropython: programa de los controladores


```python
# -*- coding: utf-8 -*-
"""
Created on Sat Jan 30 17:30:40 2021

@author: Víctor Romero
         Saúl Gonzalez
         Guillermo Martinez
         Rubén Alcantara

 Un script con todas las funciones necesarias para ejecutar el control RE/RS
 y PID en el proyecto de control ICS - Ícaro Control System
"""

#----------------- IMPORTAR LIBRERÍAS Y SCRIPTS -----------------------

#Script pesonal que simula numpy en micropython
#import minibasicnumpy as npb
from minibasicnumpy import *
import Watermelon_var as wmv
import IMU_BoschSensorTec
import utime
from math import sin

#-------------------- DEFINICIÓN DE FUNCIONES -------------------------

def F2PWM(fuerza):
    """
    Es la conversión de fuerza en newtons a PWM de acuerdo a la caracterización
    Parameters
    ----------
    fuerza : float, int
        Fuerza en Newtons

    Returns
    -------
    inner_pwm : Int
        PWM para que el motor brushless produzca la fuerza de entrada
    """
    #Aquí son las conversiones con PWM (500-950)
    inner_pwm = 54.156*fuerza+545.32
    return round(inner_pwm)


def PWM2F(pwm):
    """
    Es la conversión de PWm a fuerza de acuerdo a la caracterización
    Parameters
    ----------
    pwm : Int

    Returns
    -------
    inner_f : float
        Fuerza en Newtons
    """

    #Aquí son las conversiones con PWM (500-950)
    inner_f = 0.01846*pwm - 10.0694
    return inner_f


def SaturadorVector(vector,lim_inferior,lim_superior):
    """
    Recive un vector de 2x1 y satura cada valor individualmente
    Parameters
    ----------
    vector : [[a],[b]

    lim_inferior : Number

    lim_superior : Number


    Returns
    -------
    vector : [[a],[b]] es de minibasicnumpy
    """
    try:
        for i in range(len(vector)):
            if vector[i][0]>lim_superior:
                vector[i][0] = lim_superior
            elif vector[i][0]<lim_inferior:
                vector[i][0]=lim_inferior
        return vector
    except :
        print("something went wrong with SaturadorVector")


def SaturadorSimple(valor,lim_inferior,lim_superior):
    """
    Satura un valor simple entero o flotante

    Parameters
    ----------
    valor : Number
```

```
96
97      lim_inferior : Number
98
99      lim_superior : Number
100
101     Returns
102     -------
103     TYPE
104         Saturated Number
105     """
106     try:
107         if valor<lim_inferior:
108             return lim_inferior
109         elif valor>lim_superior:
110             return lim_superior
111         else:
112             return valor
113     except:
114         print("Something went wrong with SaturadorSimple")
115
116
117 #Función para desacelerar los motores despacio
118 def Desaceleracion(motorD,motorI,pwm_inicial,pwm_final = 500,duracion = 2000):
119
120     t = 0
121     tinit = 0
122     t_final = duracion
123
124     #Va a ser una desaceleración lineal y = mx+b
125     m1 = (pwm_final - pwm_inicial[1][0])/t_final #Motor Derecho
126     m2 = (pwm_final - pwm_inicial[0][0])/t_final #Motor Izquierdo
127
128     #La ordenada al origen es igual al valor inicial
129     tinit = utime.ticks_ms()
130
131     while t<t_final:
132         tstop = utime.ticks_ms()
133         t = utime.ticks_diff(tstop,tinit)
134         #Desaceleración lineal
135         pwmd = int(m1*t + pwm_inicial[1][0])
136         pwmi = int(m2*t + pwm_inicial[0][0])
137         #Para que no mande menos de 50
138         pwmd = pwmd if pwmd >= pwm_final else pwm_final
139         pwmi = pwmi if pwmi >= pwm_final else pwm_final
140         motorD.duty(pwmd)
141         motorI.duty(pwmi)
142
143 #Un objeto que nos ayude a obtener las mediciones,
144 #Esto con el objetivo de poder ponerle filtros como el de kalman
145 #o el filtro de exponential moving average
146 class Sensores:
147
148     def __init__(self):
149
150         #ángulos, esta volteado (es negativo)
151         self.angulo = [0,0] #[°]
152         self.angulos_estimado = [-wmv.c_i_theta,0,0]
153
154         #altura está bien
155         self.altura_estimada = [wmv.c_i_h,0,0] #altura[m],velocidad,aceleración
156
157         #Variables de actualización de los estados
158         #Dependen de la precición del sensor y la confianza en el modelo
159         self.abg_theta = [0.5,0.0,0.0] #Ganancias del filtro de kalman
160         self.abg_h = [0.1,0.0,0.0] #Ganancias del filtro de kalman
161
162
163     def Sense(self, adc_pin,i2c_imu,dt):
164
165         self.angulo_pasado = self.angulo
166         #obtenemos angulos, está volteado (es negativo)
167         #Dentro tiene un filtro complementario
168         self.angulo = IMU_BoschSensorTec.Get_angle(i2c_imu,self.angulo,dt)
169
170         #Hacemos un filtro de kalman (alpha - beta - gama) para la distancia
171         """
172         Predicciones (movimiento de newton):
173             x_siguiente = x + v*dt + 0.5*a*(dt**2)
174             v_siguiente = v + a*dt
175             a_siguiente = a
176
177         Actualización de estados
178             x = x_anterior + aplha(lectura - x_anterior)
179             v = v_anterior + beta((lectura - x_anterior)/dt)
180             a = a_anterior + gama((lectura - x_anterior)/(0.5*dt**2))
181         """
182         #para el angulo
183         angulos = self.angulos_estimado
184         diferencia = self.angulo[0] - angulos[0]
185
186         #filtrado
187         abg_theta = self.abg_theta
188         angulos[0] = angulos[0] + abg_theta[0]*(diferencia)
189         angulos[1] = angulos[1] + abg_theta[1]*(diferencia/dt)
190         angulos[2] = angulos[2] + abg_theta[2]*(diferencia/(0.5*(dt**2)))
191
```

```
        #Prediccion para la siguiente iteración
        self.angulos_estimado = [angulos[0]+angulos[1]*dt+angulos[2]*0.5*(dt**2),
                                 angulos[1]+angulos[2]*dt,
                                 angulos[2]]

        #Para la altura
        #Valor actual
        voltaje = (adc_pin.read()*3.2)/4096 #regla de 3 + offset empiricos
        #La caracterización está en cm, la pasamos a metros
        distancia_medida = SaturadorSimple((-26.80121*voltaje+58.42162)/100.0,0.04,0.4) #en metros

        altura = self.altura_estimada
        diferencia = distancia_medida - altura[0]

        #filtrado
        abg_h = self.abg_h
        altura[0] = altura[0] + abg_h[0]*(diferencia)
        altura[1] = altura[1] + abg_h[1]*(diferencia/dt)
        altura[2] = altura[2] + abg_h[2]*(diferencia/(0.5*(dt**2)))

        #Prediccion para la siguiente iteración
        self.altura_estimada = [altura[0]+altura[1]*dt+altura[2]*0.5*(dt**2),
                                altura[1]+altura[2]*dt,
                                altura[2]]

        #Regresa el ángulo volteado para que funcione igual que la funcion sensado
        return altura[0], [angulos[0],self.angulo[1]]



#-------------------------------------------------------------------------
# ------------ CREAMOS EL CONTROL COMO UN OBJETO --------------------------
#-------------------------------------------------------------------------

#Se crea el control como un objeto para evitar tener globales entre scripts
#y poder tener sus propias funciones y atributos internos


class CoconutRS_2n:
    """
    Control de retro de salidas de segundo orden
    La idea es dividir el modelo en dos modelos de segundo orden y controlarlos
    cada uno por separado

    Es para modelos de segundo orden SISO
    """
    def __init__(self):
        pass


class CoconutRS_4n_disc:
    """
    Control RS MIMO discreto
    Controla el sistema ICS de 4 orden con 2 entradas y 2 salidas

    Capaz de utilizar Control RS y control RS + int

    Timempo de discretización depende de las matrices que se le ingresen
    """
    def __init__(self,h_deseada = wmv.h_deseada, theta_deseada = wmv.theta_deseada):

        #Atributos necesarios para el definir el sistema
        self.a_disc = wmv.Alin_disc
        self.b_disc = wmv.Blin_disc
        self.c_disc = wmv.Clin_disc
        self.d_disc = wmv.Dlin_disc

        self.x1barra = wmv.x1barra
        self.x2barra = wmv.x2barra
        self.x3barra = wmv.x3barra
        self.x4barra = wmv.x4barra
        self.f1barra = wmv.F1barra
        self.f2barra = wmv.F2barra

        # c_i_h = wmv.c_i_h
        # c_i_hp = wmv.c_i_hp
        # c_i_theta = wmv.c_i_theta
        # c_i_thetap = wmv.c_i_thetap

        #Atributos necesarios para definir el control
        self.matrizK = wmv.K_disc
        self.matrizK0 = wmv.K0_disc
        self.matrizKi = wmv.Ki_disc
        self.matrizKrsi = wmv.Krsi_disc

        # h_deseada =  wmv.h_deseada #En metros
        # theta_deseada = wmv.theta_deseada #En radianes

        self.satSup = PWM2F(wmv.saturacionSuperiorPWM)
        self.satInf = PWM2F(wmv.saturacionInferiorPWM)

        #Atributos necesarios para definir el observador del lazo cerrado
        self.a_obs = wmv.Aobs_disc
        self.b_obs = wmv.Bobs_disc
        self.c_obs = wmv.Cobs_disc
        self.d_obs = wmv.Dobs_disc
```

```
288
289         #Condiciones iniciales del observador
290         self.edos_estimados = array(4,1,[0,0,0,0])
291
292         #Señal de control (En Newtons)
293         self.u_signal = array(2,1,[0,0])
294
295         #Integrador
296         self.v_integrator = array(2,1,[0,0])
297
298         #Valores deseados
299         self.h_deseada = h_deseada
300         self.theta_deseada = theta_deseada
301         self.delta_h_deseada = h_deseada - self.x1barra
302         self.delta_theta_deseada = theta_deseada - self.x3barra
303         self.delta_deseados = [[self.delta_h_deseada],[self.delta_theta_deseada]]
304
305     def __str__(self):
306         #Aquí se contrulle una string que va a devolver el objeto si se necesita
307         #imprimir
308         s = "valores deseados: [{},{}], error: [{},{}]".format(self.h_deseada,self.theta_deseada,
309                                                                (self.sensado_presente_delta[0][0]+self.x1barra) - self.h_deseada,
310                                                                (self.sensado_presente_delta[1][0]+self.x3barra) - self.theta_deseada)
311         return s
312
313     def ChangeDesired(self,h_deseada,theta_deseada):
314         #Cambia los valores deseados del control
315         self.h_deseada = h_deseada
316         self.theta_deseada = theta_deseada
317         self.delta_h_deseada = h_deseada - self.x1barra
318         self.delta_theta_deseada = theta_deseada - self.x3barra
319         self.delta_deseados = [[self.delta_h_deseada],[self.delta_theta_deseada]]
320
321     def ChangeMatrix(self, K = 0, K0 = 0, Ki = 0, Krsi = 0):
322         if (K != 0):
323             self.matrizK = K
324         if (K0 != 0):
325             self.matrizK0 = K0
326         if (Ki != 0):
327             self.matrizKi = Ki
328         if (Krsi != 0):
329             self.matrizKrsi = Krsi
330
331     def UpdateValues(self,valores_sensados):
332         #Convertimos los valores absolutos a deltas
333         self.sensado_presente_delta = elementwisemin(valores_sensados, [[self.x1barra],[self.x3barra]])
334
335     def Feedback(self, isIntegral = False):
336         #Obtiene la señal de control siguiente
337         #El control tiene acción integral
338         #Observador discreto
339
340         #Entrada del observador [salidas_delta;señal de control anterior]
341         u_obs_disc = [[self.sensado_presente_delta[0][0]],
342                       [self.sensado_presente_delta[1][0]],
343                       [self.u_signal[0][0]],
344                       [self.u_signal[1][0]]]
345
346         #Observador discreto de lazo cerrado
347         self.edos_estimados = elementwisesum(dot(self.a_obs, self.edos_estimados),
348                                              dot(self.b_obs,u_obs_disc))
349
350         error_actual = elementwisemin(self.delta_deseados, self.sensado_presente_delta)
351         self.v_integrator = elementwisesum(self.v_integrator, error_actual)
352
353         #Aquí puede ir el anti wind-up del integral
354         self.v_integrator = SaturadorVector(self.v_integrator,-25,25) #-25,25
355
356         u_signal = elementwisesum(dot(constantXmatrix(-1, self.matrizKrsi),self.edos_estimados),
357                                   dot(self.matrizKi,self.v_integrator))
358
359         #Anti Wind-up
360         u_signal = SaturadorVector(u_signal,self.satInf,self.satSup)
361         self.u_signal = u_signal
362
363         return [[F2PWM(u_signal[0][0]+self.f1barra)],[F2PWM(u_signal[1][0]+self.f2barra)]]
364
365
366     def COObserver(self):
367
368         #Observador discreto (No se utiliza, se agregó directamente en feedback)
369         #Entrada del observador [salidas_delta;señal de control anterior]
370         u_obs_disc = [[self.sensado_presente_delta[0][0]],
371                       [self.sensado_presente_delta[1][0]],
372                       [self.u_signal[0][0]],
373                       [self.u_signal[1][0]]]
374
375         #Observador discreto de lazo cerrado
376         self.edos_estimados = elementwisesum(dot(self.a_obs, self.edos_estimados),
377                                              dot(self.b_obs,u_obs_disc))
378
379
380
381 #Función por si se desean combinar los efectos de los 2 PIDs para el dron
382 def PIDCombineEffects(u_altura,u_angulo,F1barra,F2barra):
383
```

```
    """
    Esta función combina la señal de control que corresopnde a

    deltaTau:
        Diferencia de fuerza entre los motores F1 y F2 requerido
        por el control de angulo

    deltaFtotal:
        Diferencia de fuerza de ambas sumadas referente a la
        fuerza de equilibro requerida para mantener el modelo
        en el aire

    Entrega como resultado F1 y F2 individualmente

    """
    mangf1 = -1*u_angulo if u_angulo<0 else 0
    mangf2 = u_angulo if u_angulo>=0 else 0

    F1 = F1barra + u_altura + mangf1
    F2 = F2barra + u_altura + mangf2

    return [F1,F2]


class CoconutPID:
    """
    Control PID discreto para modelos SISO

    """

    def __init__(self,kp = 1, taui = 0, taud = 0, ts = 0.016, ref = 0,
                 maximo = 100,minimo = -100, compensador = lambda x,y: x*y):

        self.ref = ref #Referencia deseada [En metros o radianes dependiendo]
        self.ts = ts #Tiempo de muestreo

        #PID ángulo (forma estándar, no paralelo)
        self.e = [0,0,0] #vector de error [e[k-2],e[k-1],e[k]]
        self.u = [0,0] #vector de la señal de control [u[k-1],u[k]]

        self.kp = kp #Ganancia proporcional
        self.taui = taui #Ganancia integral
        self.taud = taud #Ganancia derivativa

        #Definimos las constantes y digitalizamos el control
        self.ChangeConstants(kp=self.kp,taui=self.taui,taud=self.taud)

        #Variables para el antiwindup (recordar que se debe manejar en deltas)
        self.maximo = maximo
        self.minimo = minimo

        #Compensador (siempre es una función lambda)
        self.compensador = compensador


    def Feedback(self,sensed,awp = True,compensador_variable = 1):
        """
        Esta función realiza los calculos del PID.
        Actualiza los datos del error y corre un ciclo de feedback

        input: sensed -> Valor sensado
               awp (anti-windup) -> Prende o apaga el anti-windup
               compensador_variable -> Variable extra para el compensador

        return: señal de control

        """
        #Esta función se debe ejecutar cada periodo de muestreo los más constante posible

        e = self.e
        u = self.u

        #Recorrer (Actualizar) valores de señal de control u
        self.RecorrerArray(u)

        #Recorrer (Actualizar) valores de error e
        self.RecorrerArray(e)

        #Calcular el error actual e[k]
        e[2] = self.ref - sensed

        #Ejecutar el PID para obtener U[k] con o sin anti-windup y saturación.
        if awp:

            deltaU = self.q0*e[2] + self.q1*e[1] + self.q2*e[0]
            deltaU = deltaU if self.minimo < u[0]+deltaU < self.maximo else 0
            lu = u[0] + deltaU

        else:
            lu = u[0] + self.q0*e[2] + self.q1*e[1] + self.q2*e[0]


        u[1] = lu
        #recuperamos lo obtenido para futuras iteraciones
        self.u = u
        self.e = e
```

```
480
481         #Pasarla por el compensador definido y retornar resultado
482         return self.compensador(lu,compensador_variable)
483
484
485     #Para recorrer los valores del array
486     def RecorrerArray(self,l):
487         for i in range(1,len(l)):
488             l[i-1] = l[i]
489
490
491
492     def ChangeDesired(self, newref):
493         """
494         Función para cambiar los valores de referencia del control
495
496         newref -> Nuevo valor de referencia
497         """
498         self.ref = newref
499
500         #Todavia no estoy seguro de si esto tambien se debe reiniciar
501         # self.u = [0,0]
502         # self.e = [0,0,0]
503
504         pass
505
506     def ChangeConstants(self,kp,taui,taud):
507         """
508         Para hacer el cambio de kp = num, taui = num y taud = num
509         y actualizar q0, q1 y q2
510         """
511         ts = self.ts #Para no llamar tanto self
512         self.kp = kp
513         self.taui = taui
514         self.taud = taud
515
516         #Esta condición es porque si taui se hace cero ocurre una división
517         #entre cero
518         if taui == 0:
519             self.q0 = kp*(1+taud/ts)
520             self.q1 = -kp*(1+(2*taud)/ts)
521             self.q2 = kp*(taud/ts)
522         else:
523             self.q0 = kp*(1+ts/(2*taui)+taud/ts)
524             self.q1 = -kp*(1-ts/(2*taui)+(2*taud)/ts)
525             self.q2 = kp*(taud/ts)
526
527
528 #Esta versión del PID tiene la derivada directamente en la medición para evitar
529 #la derivative kick. Está discretizada.
530 class CoconutPID_nokick:
531     """
532     Control PID discreto para modelos SISO
533
534     """
535
536     def __init__(self,kp = 1, taui = 0, taud = 0, ts = 0.016, ref = 0,
537                  maximo = 100,minimo = -100, compensador = lambda x,y: x*y):
538
539         self.ref = ref #Referencia deseada [En metros o radianes dependiendo]
540         self.ts = ts #Tiempo de muestreo
541
542         #PID ángulo (forma estándar, no paralelo)
543         self.e = [0,0] #vector de error [e[k-1],e[k]]
544         self.y = [0,0,0] #vector de sensado [y[k-2],y[k-1],y[k]]
545         self.u = [0,0] #vector de la señal de control [u[k-1],u[k]]
546
547         self.kp = kp #Ganancia proporcional
548         self.taui = taui #Ganancia integral
549         self.taud = taud #Ganancia derivativa
550
551         #Definimos las constantes y digitalizamos el control
552         self.ChangeConstants(kp=self.kp,taui=self.taui,taud=self.taud)
553
554         #Variables para el antiwindup (recordar que se debe manejar en deltas)
555         self.maximo = maximo
556         self.minimo = minimo
557
558         #Compensador (siempre es una función lambda)
559         self.compensador = compensador
560
561
562     def Feedback(self,sensed,awp = True,compensador_variable = 1):
563         """
564         Esta función realiza los calculos del PID.
565         Actualiza los datos del error y corre un ciclo de feedback
566
567         input: sensed -> Valor sensado
568                awp (anti-windup) -> Prende o apaga el anti-windup
569                compensador_variable -> Variable extra para el compensador
570
571         return: señal de control
572
573         """
574         #Esta función se debe ejecutar cada periodo de muestreo los más constante posible
575
```

```
576         e = self.e
577         u = self.u
578         y = self.y
579 
580         #Recorrer (Actualizar) valores de señal de control u
581         self.RecorrerArray(u)
582 
583         #Recorrer (Actualizar) valores de error e
584         self.RecorrerArray(e)
585 
586         #Recorrer (Actualizar los valores sensados y
587         self.RecorrerArray(y)
588 
589         #Calcular el error actual e[k] y el sensado actual y
590         y[2] = sensed
591         e[1] = self.ref - sensed
592 
593         #Ejecutar el PID para obtener U[k] con o sin anti-windup y saturación.
594         if awp:
595 
596             deltaU = self.h0*e[1] + self.h1*e[0] + self.h2*(y[2]-2*y[1]+y[0])
597             deltaU = deltaU if self.minimo < u[0]+deltaU < self.maximo else 0
598             lu = u[0] + deltaU
599 
600         else:
601             lu = u[0] + self.h0*e[1] + self.h1*e[0] + self.h2*(y[2]-2*y[1]+y[0])
602 
603 
604         u[1] = lu
605         #recuperamos lo obtenido para futuras iteraciones
606         self.u = u
607         self.e = e
608         self.y = y
609 
610         #Pasarla por el compensador definido y retornar resultado
611         return self.compensador(lu,compensador_variable)
612 
613 
614     #Para recorrer los valores del array
615     def RecorrerArray(self,l):
616         for i in range(1,len(l)):
617             l[i-1] = l[i]
618 
619 
620 
621     def ChangeDesired(self, newref):
622         """
623         Función para cambiar los valores de referencia del control
624 
625         newref -> Nuevo valor de referencia
626         """
627         self.ref = newref
628 
629         #Todavia no estoy seguro de si esto tambien se debe reiniciar
630         # self.u = [0,0]
631         # self.e = [0,0,0]
632 
633         pass
634 
635     def ChangeConstants(self,kp,taui,taud):
636         """
637         Para hacer el cambio de kp = num, taui = num y taud = num
638         y actualizar q0, q1 y q2
639         """
640         ts = self.ts #Para no llamar tanto self
641         self.kp = kp
642         self.taui = taui
643         self.taud = taud
644 
645         #Esta condición es porque si taui se hace cero ocurre una división
646         #entre cero
647         if taui == 0:
648             self.h0 = kp*(1)
649             self.h1 = -kp*(1)
650             self.h2 = -kp*(taud/ts)
651         else:
652             self.h0 = kp*(1+ts/(2*taui))
653             self.h1 = -kp*(1-ts/(2*taui))
654             self.h2 = -kp*(taud/ts)
```

Parte VI

Referencias

Referencias

[1] C. Bissell, «A History of Automatic Control,» en *Springer: Handbook of Automation*. Springer, 2009, págs. 53-69.

[2] N. S. Nise, *Control Systems Engineering*. USA: Wiley, 2011.

[3] IWM, *A Brief History of Drones*, https://www.iwm.org.uk/history/a-brief-history-of-drones, Accesado: Agosto - 2021.

[4] Wikipedia, *Unmanned aerial vehicle*, https://en.wikipedia.org/wiki/Unmanned_aerial_vehicle, Accesado: Agosto - 2021.

[5] A. Intwala e Y. Parikh, «A Review on Vertical Take Off and Landing (VTOL) Vehicles,» *International Journal of Innovative Research in Advanced Engineering (IJIRAE)*, vol. 2, págs. 186-191, feb. de 2015.

[6] Vaughn College, *National Aviation History Month: Vaughn Reviews Advances in Vertical Takeoff and Landing Technology*, https://www.vaughn.edu/blog/national-aviation-history-month-vaughn-reviews-advances-in-vertical-takeoff-and-landing-technology/, Accesado: Agosto - 2021, 2020.

[7] Wikipedia, *Harrier*, https://es.wikipedia.org/wiki/Harrier, Accesado: Agosto - 2021.

[8] Quanser, *Quanser: innovate, educate*, https://www.quanser.com/careers/, Accesado: Agosto - 2021.

[9] tecquipment, *tecquipment*, https://www.tecquipment.com/control-engineering, Accesado: Agosto - 2021.

[10] Feedback, *Feedback*, http://www.feedback-instruments.com/, Accesado: Agosto - 2021.

[11] Amira, *Amira*, http://www.ict.com.tw/AI/Amira/amira/home.htm, Accesado: Agosto - 2021.

[12] G. E. Dieter y L. C.Schmidt, *Engineering Design*. USA: McGrawHill, 2009.

[13] Mgter. Arq. Nora E. NACIF, *Métodos de diseño*, http://www.faud.unsj.edu.ar/descargas/blogs/apuntes-de-ctedra-mtodos-y-estrategias-de-diseo_Metodos%20y%20Estrategias%20de%20Dise%C3%B1o.pdf, Accesado: Mayo-2022.

[14] Universidad Nacional de San Juan, Facultad de Arquitectura Urbanismo y Diseño, *Proceso de diseño*, http://www.faud.unsj.edu.ar/descargas/blogs/unidad-2-el-proceso-de-diseno_EL%20PROCESO%20DE%20%20DE%20DISE%C3%910.pdf, Accesado: Mayo-2022.

[15] Serafín Castañeda, *El Proceso de diseño: Sueña, construye y disfruta*, Presentación de clase de diseño mecatrónico (UNAM - FI), 2019.

[16] K. T. Ulrich y S. D. Eppinger, «Arquitectura del producto,» en *Diseño y desarrollo de productos*. McGrawHill, 2013, págs. 183-185.

[17] M. Modarres y S. W. Cheon, «Function-centered modeling of engineering systems using the goal tree–success tree technique and functional primitives,» *ScienceDirect*, 1998.

[18] J. P. Cardona, J. J. Leal y J. E. Ustariz, «Mathematical modeling of white and black box in engineering education,» *SciElo chile*, 2020.

[19] Universidad del país vasco, *Fase de diseño de detalle*, https://ocw.ehu.eus/pluginfile.php/44416/mod_resource/content/1/5.4-Fase_diseno_de_detalle.pdf, Accesado: Abril-2022.

[20] Dejan, *How Brushless Motor and ESC Work*, https://howtomechatronics.com/how-it-works/how-brushless-motor-and-esc-work/, Accesado: Junio-2021, 2019.

[21] H. A. F. Bobadilla, I. J. T. Landín y U. R. Carmona, «Diseño, construcción y control de una aeronave tipo dron,» Tesis de mtría., Universidad Nacional Autónoma de México - Facultad de Ingeniería., México, CDMX, 2016.

[22] Digikey, *How to Power and Control Brushless DC Motors*, https://www.digikey.com/en/articles/how-to-power-and-control-brushless-dc-motors, Accesado: Junio-2021, 2016.

[23] Electropedia, *Electric Drives - Brushless DC / AC and Reluctance Motors (Description and Applications)*, https://www.mpoweruk.com/motorsbrushless.htm, Accesado: Junio-2021.

[24] RacerStar, *Racerstar BR2212 1400KV 2-4S Brushless Motor For RC Models*, https://www.racerstar.com/racerstar-br2212-1400kv-2-4s-brushless-motor-for-rc-models-p-54.html, Accesado: Marzo-2022.

[25] Herramientas tecnológicas profesionales, *BeagleBone Black*, https://hetpro-store.com/TUTORIALES/beaglebone-black-pwm/, Accesado: Marzo-2022.

[26] C. G. de la Torre, «Sistema de grabación de señales PPM para aplicaciones de animatrónica,» Tesis de mtría., Escuela Técnica Superior de Ingeniería - Universidad de Sevilla, Sevilla, Esp, 2017.

[27] Desconocido, *30A BLDC ESC*, https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjKjpWo6t_2AhWImmoFHTczA14QFnoECAgQAQ&url=https%3A%2F%2Fwww.optimusdigital.ro%2Findex.php%3Fcontroller%3Dattachment%26id_attachment%3D451&usg=AOvVaw0K8g8cGZgvGU7jrp-ABJkw, Accesado: Marzo-2022.

[28] Comprar Drones Online, *¿Cómo funcionan las hélices de un drone?* https://www.comprardrones.online/como-funcionan-las-helices-de-un-drone/, Accesado: Junio-2021, 2016.

[29] Desconocido, *Lo que hay que saber para elegir las hélices para un cuadricoptero*, https://www.prometec.net/elegir-helices-dron/, Accesado: Noviembre-2022.

[30] ——, *Impact of Jets*, https://gse.ac.in/wp-content/uploads/2020/05/Fluid-Mechanics-7th-chapter.pdf, Accesado: Agosto-2020.

[31] R. L. W. Robert D. Christ, «Navigational Sensors,» *ScienceDirect*, vol. 2, págs. 453-475, 2014.

[32] LidarUSA, *IMU Types*, https://www.lidarusa.com/imu-types.html, Accesado: Abril-2022.

[33] L. Tran, «Data Fusion with 9 Degrees of Freedom Inertial Measurement Unit To Determine Object's Orientation,» Tesis de mtría., California Polytechnic State University, San Luis Obispo, jun. de 2017.

[34] UNAM, *Conceptos básicos de los acelerómetros*, http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/213/A11.pdf?sequence=11, Accesado: Abril-2022.

[35] J. I. Atorino, L. P. Bortolín, E. Rodríguez, R. O. Farías y E. E. Rodríguez, «Experimentos con un sensor de efecto Hall,» *Latin-American Journal of Physics Education*, vol. 3, págs. 606-611, mayo de 2009.

[36] 330ohms, *Sensores Inerciales: Acelerómetros, giroscopios e IMUs*, https://blog.330ohms.com/2017/09/27/sensores-inerciales-acelerometros-giroscopios-e-imus/, Accesado: Abril-2022, 2017.

[37] Luis Llamas, *Cómo usar un giroscopio en nuestros proyectos de arduino*, https://www.luisllamas.es/como-usar-un-giroscopio-arduino/, Accesado: Abril-2022.

[38] Sennav, *AHRS VS IMU*, https://sennavs.com/ahrs-vs-imu/, Accesado: Abril-2022.

[39] Bosch, *IMU: BMI085*, https://www.bosch-sensortec.com/products/motion-sensors/imus/bmi085/, Accesado: Diciembre-2020.

[40] OMRON, *Technical Explanation for Photoelectric Sensors*, https://www.ia.omron.com/data_pdf/guide/43/photoelectric_tg_e_8_4.pdf, Accesado: Julio-2021.

[41] Sharp, *GP2Y0E03 Datasheet*, Sharp Microelectronics.

[42] F. S. Espinosa, «Introducción a los microcontroladores,» en *Los Microcontroladores AVR de ATMEL*. Universidad Tecnológica de la Mixteca, 2012, págs. 15-33.

[43] Jacob Beningo, *10 pasos para seleccionar un Microcontrolador*, https://electrojoan.com/10-pasos-para-seleccionar-un-microcontrolador/, Accesado: Abril-2022, 2022.

[44] A. D. Perez, «Protocolos de comunicación entre microcontroladores.Caso de estudio: Protocolo CAN.,» Tesis de mtría., Universidad Nacional De La Plata, Argentina, Buenos Aires, 2016.

[45] Robots Didácticos, *Descripción y funcionamiento del Bus I2C*, http://robots-argentina.com.ar/didactica/descripcion-y-funcionamiento-del-bus-i2c/, Accesado: Abril-2022, 2019.

[46] Dr. Rubén E-Marmolejo, *Microcontrolador qué es y para que sirve*, https://hetpro-store.com/TUTORIALES/microcontrolador/, Accesado: Abril-2022, 2017.
[47] Centro de innovación CISET, *Wifi Conexión inalámbrica*, https://www.ciset.es/glosario/496-wifi-red-inalambrica, Accesado: Mayo-2022.
[48] CISCO, *¿Qué es un access point?* https://www.cisco.com/c/es_mx/solutions/small-business/resource-center/networking/what-is-access-point.html, Accesado: Mayo-2022.
[49] G. V. Alejandro, «Bluetooth,» Tesis de mtría., Universidad Autónoma del Estado de Hidalgo- Instituto de Ciencias Básicas e Ingeniería, México, Hidalgo.
[50] Wim Hoogenraad, *Wifi y Bluetooth, ambos inalámbricos, ¿cuál es la diferencia?* https://es.itpedia.nl/2018/07/12/wifi-en-bluetooth-wat-is-het-verschil/, Accesado: Mayo-2022, 2018.
[51] Espressif Systems, *ESP32-S3 Series Datasheet*, Espressif Systems, 2022.
[52] Jacob Beningo, *Cómo seleccionar y usar el módulo ESP32 con Wi-Fi/Bluetooth adecuado para una aplicación de IoT industrial*, https://www.digikey.com.mx/es/articles/how-to-select-and-use-the-right-esp32-wi-fi-bluetooth-module#:~:text=El%20ESP32%20se%20ha%20vuelto,desarrollo%20integrado%20(IDE)%20de%20Arduino, Accesado: Febrero-2021.
[53] Direct Industry, *Qué fuente de alimentación eléctrica elegir*, https://guide.directindustry.com/es/que-fuente-de-alimentacion-electrica-elegir/, Accesado: Abril-2022.
[54] Fidestec, *Cómo funcionan las fuentes de alimentación conmutadas I*, https://fidestec.com/blog/fuentes-de-alimentacion-conmutadas-01/, Accesado: Abril-2022.
[55] Mean Well, *SE-600-SPEC*, Mean Well, 2019.
[56] Juan Antonio Soto, *Guía de Protecciones en Fuentes de Alimentación*, https://www.geeknetic.es/Guia/1565/Guia-de-Protecciones-en-Fuentes-de-Alimentacion.html, Accesado: Abril-2022, 2019.
[57] J.C. Torres, *Diseño asistido por ordenador*, https://lsi2.ugr.es/~cad/teoria/Tema1/RESUMENTEMA1.PDF, Accesado: Abril-2022.
[58] Autodesk, *Inventor: Poderoso software de diseño mecánico para tus ideas más ambiciosas*, https://www.autodesk.mx/products/inventor/overview?term=1-YEAR&tab=subscription, Accesado: 10-Noviembre-2022.
[59] STACBOND, *Panel composite de aluminio*, https://stacbond.com/panel-composite-de-aluminio/, Accesado: Mayo-2022.
[60] I. J. L. M. Flores, «Aceros y sus aplicaciones,» Tesis de mtría., Universidad Autónoma de Nuevo León, San Nicolas de los Garza, México, dic. de 1996.
[61] S. Farah, D. G. Anderson y R. Langer, «Physical and Mechanical Properties of PLA, and Their Functions in Widespread Applications — A Comprehensive Review.,» *Elsevier BV*, vol. 107, págs. 367-392, dic. de 2016.
[62] AITIM, *Tableros de fibras de densidad media (MDF)*, AITIM, 2015.
[63] Ana Bonilla, *Herramientas de Diseño e ingeniería*, https://www.bizkaia.eus/Home2/Archivos/DPT08/Temas/Pdf/ca_GTcapitulo1.pdf?hash=b99514a126a592fe18a6b6f514c2624f, Accesado: Mayo-2022, 2003.
[64] R. L. Mott, «Factores de diseño,» en *Diseño de Elementos de Máquinas*. Pearson, 2006, págs. 185-186.
[65] Adrián Macías, *Cálculo e Interpretación del Factor de Seguridad*, https://intelligy.com/blog/2019/05/07/calculo-e-interpretacion-del-factor-de-seguridad/, Accesado: Mayo-2022, 2019.
[66] Autodesk, *Ajuste de la configuración y los controles de malla*, Autodesk, 2021.
[67] Digital Guide IONOS, *¿Qué es una interfaz gráfica de usuario (GUI)?* https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/que-es-una-gui/, Accesado: Marzo-2022, 2021.
[68] Python Software Foundation, *Python*, https://www.python.org/, Accesado: 10-Noviembre-2022, 2022.

[69] Python - Documentation, *tkinter — Python interface to Tcl/Tk*, https://docs.python.org/3/library/tkinter.html#architecture, Accesado: Abril-2021, 2021.
[70] Stephen John Machin, *xlrd - documentation*, https://xlrd.readthedocs.io/en/latest/, Accesado: 10-Noviembre-2021, 2005.
[71] xlwt contributors, *xlwt - documentation*, https://xlwt.readthedocs.io/en/latest/, Accesado: 10-Noviembre-2021, 2002.
[72] David Cortesi, *PyInstaller Manual*, https://pyinstaller.org/en/stable/, Accesado: 14-Noviembre-2022, 2022.
[73] CIDECO, *¿Que es un router CNC?* https://sideco.com.mx/que-es-un-router-cnc/#ventajas-router, Accesado: Mayo-2022.
[74] Mtro. Felipe Díaz Del Castillo Rodriguez, *Lecturas de Ingeniería No26 Impresión 3D, una introducción*, https://asesoriacad.files.wordpress.com/2014/02/unidad-1-introduccic3b3n-al-cad-cam-cae.pdf, Accesado: Mayo-2022, 2018.
[75] centro de laseres pulsados, *¿Como funciona un laser?* https://www.clpu.es/divulgacion/bits/como-funciona-un-laser, Accesado: Mayo-2022.
[76] Ing. Alfredo Carrasco, *Tecnología de corte de Metales con Láser*, Cite Energía, 2016.
[77] Jorge Díaz Salgado, *(Apuntes de la clase de Modelado de sistemas físicos - UNAM FI. Comunicación personal*, Clase presencial, 2019.
[78] Edmundo Gabriel Rocha Cózatl, *(Apuntes de la clase de Control Avanzado - UNAM FI. Comunicación personal*, Clase presencial, 2020.
[79] Sergio Andrés Castaño Giraldo, *Control PID – Acción de Control Proporcional*, https://controlautomaticoeducacion.com/control-realimentado/control-pid-accion-proporcional/, Accesado: Febrero - 2022.
[80] ——, *Acción de Control Integral – Control PID*, https://controlautomaticoeducacion.com/control-realimentado/accion-de-control-integral-control-pid/, Accesado: Febrero - 2022.
[81] ——, *Acción de Control Derivativo – Control PID*, https://controlautomaticoeducacion.com/control-realimentado/accion-de-control-derivativo-control-pid/, Accesado: Febrero - 2022.
[82] ——, *Anti Windup en un Control PID*, https://controlautomaticoeducacion.com/control-realimentado/anti-windup-en-un-control-pid/, Accesado: Febrero - 2022.
[83] Control Guru: Practical Process Control, *PID Control and Derivative on Measurement*, https://controlguru.com/pid-control-and-derivative-on-measurement/, Accesado: Abril - 2022, 2015.
[84] Sergio Andrés Castaño Giraldo, *Sintonía PID por el Método CHR*, https://controlautomaticoeducacion.com/control-realimentado/sintonia-pid-por-el-metodo-chr/, Accesado: Febrero - 2022.
[85] ——, *Método Cohen y Coon de Sintonía para Controles PID*, https://controlautomaticoeducacion.com/control-realimentado/cohen-coon/, Accesado: Febrero - 2022.
[86] ——, *Todo sobre Ziegler Nichols – Sintonia de Control PID*, https://controlautomaticoeducacion.com/control-realimentado/ziegler-nichols-sintonia-de-control-pid/, Accesado: Febrero - 2022.
[87] R. P. Moreno, *Evolución Histórica de la Ingeniería de Control*, dic. de 1999.
[88] Sergio Andrés Castaño Giraldo, *Controladores PID Discreto*, https://controlautomaticoeducacion.com/control-realimentado/controladores-pid-discreto/, Accesado: Febrero - 2022.
[89] L. E. G. Jaimes, *Control Digital: Teoría y Práctica*. Medellín: Politécnico Colombiano Jic, 2009.
[90] C. A. Smith y A. B. Corripio, *Control Automático de procesos: Teoría y Práctica*. México: Limusa, 1991.
[91] S. Domínguez, P. Campoy, J. M. Sebastián y A. Jiménez, *Control en el Espacio de Estado*. Madrid: Pearson Educación, 2006.
[92] Mathworks Help Center, *System Identification Toolbox*, https://la.mathworks.com/help/ident/index.html?s_tid=CRUX_lftnavl, Accesado: Agosto - 2020.

[93] Sergio Andrés Castaño Giraldo, *Control por Realimentación de Estados con Integrador tipo Servo*, https://controlautomaticoeducacion.com/sistemas-dinamicos-lineales/control-por-realimentacion-de-estados-con-integrador-tipo-servo/, Accesado: Mayo - 2022.

[94] Mathworks Help Center, *place*, https://la.mathworks.com/help/control/ref/place.html, Accesado: Agosto - 2020.

[95] O. C. L. Q. Methods, *Classical mechanics: the theoretical minimum*. USA: Prentice Hall International, 1989.

[96] Sergio Andrés Castaño Giraldo, *Índices de Desempeño*, https://controlautomaticoeducacion.com/control-realimentado/indices-de-desempeno/?fbclid=IwAR0BE18JaUM8i7Anz3O-vxeqPO74xbkaJrFSAqBQaQXr5iz-AGafaulsDNg, Accesado: Noviembre - 2022.

[97] Edmundo Gabriel Rocha Cózatl, *Control Óptimo*, Presentación de clase - Control Avanzado, UNAM - FI.

[98] Mathworks Help Center, *lqr*, https://la.mathworks.com/help/control/ref/lqr.html?s_tid=doc_ta, Accesado: Febrero - 2022.

[99] Sergio Andrés Castaño Giraldo, *Observadores de Estados*, https://controlautomaticoeducacion.com/sistemas-dinamicos-lineales/observadores-de-estados/, Accesado: Marzo - 2022.

[100] Elizabeth Villota Cerna, *Control Moderno y Óptimo*, Universidad Nacional de Ingeniería. Lima, Perú, 2009.

[101] Ricardo Julián Mantz, *Observadores de estado*, Universidad Nacional de La Plata, Argentina, 2003.

[102] K. Ogata, *Sistemas de control en tiempo discreto*. México: Prentice Hall Hispanoamerica, 1996.

[103] J. F. López, *Coeficiente de determinación (R cuadrado)*, https://economipedia.com/definiciones/r-cuadrado-coeficiente-determinacion.html, Accesado: Noviembre - 2022.

[104] D. P. George y P. Sokolovsky, *MicroPython documentation*, https://docs.micropython.org/en/latest/, Accesado: Abril - 2022.

[105] Alex Becker, *THE α - β - γ FILTER*, https://www.kalmanfilter.net/alphabeta.html, Accesado: Marzo-2022, 2022.

[106] ——, *KalmanFilter.Net*, https://www.kalmanfilter.net/default.aspx, Accesado: Marzo-2022, 2022.

[107] C. R. Alavala, *CAD/CAM concepts and applications*. Delhi: PHI Learning private limited, 2009.

[108] C.-T. Chen, *Linear System Theory and Design*. New York, NY: Oxford UNiversity Press, 1999.

[109] F. J. R. Ramírez, *Dinámica de Sistemas*. México: Trillas, 1989.

[110] K. Ogata, *System Dynamics*. New Jersey: Pearson, 2004.

[111] A. I. B. Murguía, «Implementación de un controlador externo en un cuadricóptero comercial,» Tesis de mtría., Universidad Nacional Autónoma de México - Facultad de Ingeniería, México, CDMX, 2014.

[112] R. R. Víctor, G. D. Saúl y M. M. Guillermo, «Vertical Take Off and Landing System,» 2020.

[113] Alfredo Lopez, *Teoría General de los Sistemas*, https://www.sesge.org/images/docs/tgs_alopez.pdf, Accesado: Mayo-2022.

[114] G. M. M. Claudia Albornoz Mario M.Berón, «Evaluación de InterfacesGráfica de Usuario,» *Universidad Nacional de San Luis Argentina*,

[115] K. Krykowski y J. Hetmańczyk, «Constant Current Models of Brushless DC Motor,» *Electrical, Control and Communication Engineering*, págs. 19-24, 2013.

[116] C. Jiang y G. Zhao, «A Preliminary Study of 3D Printing on Rock Mechanics,» *Rock Mechanics and Rock Engineering*, vol. 48, ene. de 2014. DOI: 10.1007/s00603-014-0612-y.

[117] A. H. Junco y J. R. M. Fernández, «Design and Implementation of an Attitude and Heading Reference System (AHRS) using Direction Cosine Matrix,» *Revista Cubana de Ciencias Informáticas*, vol. 11, págs. 14-17, ene. de 2017.

[118] Espressif Systems, *ESP32 Technical Reference Manual*, Espressif Systems, 2021.

[119] Ing.Mariangely Talavera, *Aplicaciones CAD/ CAM/ CAE*, https://asesoriacad.files.wordpress.com/2014/02/unidad-1-introduccic3b3n-al-cad-cam-cae.pdf, Accesado: Mayo-2022.

[120] Q-NAP qatar national aluminium panel co., *ACP-PE material specification*, http://qbond.co/images/SpecsPE.pdf?fbclid=IwAR1SnyYFUucyED3JsTSl-UEbXCFcVTN7w1XVEiXDLbwZBMDfsfq7MU2YYns, Accesado: Marzo-2022.

[121] Chao Jiang, *A Preliminary Study of 3D Printing on Rock Mechanics*, https://www.researchgate.net/figure/Youngs-modulus-and-Poissons-ratio-for-the-3D-printed-specimens-in-the-UCS-test_tbl3_271923683, Accesado: Marzo-2022.

[122] M. H. Kevin, «Estudio de las aplicaciones y ventajas que ofrece la impresión 3D en el ámbito de la automoción,» Tesis de mtría., Universidad Politécnica de Cataluña, Barcelona, España, ene. de 2021.

[123] F. J. H. Ramón, «Diseño y construcción de una máquina de control numérico por corte co2 láser de 40 watts para acrílico de hasta 4 mm.,» Tesis de mtría., Universidad Internacional del Ecuador, Quito, Ecuador, sep. de 2014.

[124] G. Plett y J. Lee, eds., *State-Space Models and the Discrete-Time Realization Algorithm*, Lecture notes prepared by G.L. Plett and J.L. Lee, 2018.

[125] G. Plett y J. Lee, eds., *State-Space Models and the Discrete-Time Realization Algorithm*, Lecture notes prepared by G.L. Plett and J.L. Lee, 2018.

[126] Jorge Díaz Salgado, *(Apuntes de la clase de Control Aplicado - UNAM FI. Comunicación personal)*, Clase presencial, 2020.

[127] Dr. Ashish Bagai, *Perspective on DARPA Vertical Flight Initiatives*, https://sites.nationalacademies.org/cs/groups/depssite/documents/webpage/deps_168800.pdf, Accesado: Agosto - 2021, 2015.

[128] Wikipedia, *VTOL*, https://es.wikipedia.org/wiki/VTOL, Accesado: Agosto - 2021.

[129] UAVNavigation, *Inertial Navigation*, https://www.uavnavigation.com/support/kb/general/inertial-navigation-system-and-estimation/inertial-navigation, Accesado: Abril-2022.

[130] Vectornav, *WHAT IS AN INERTIAL MEASUREMENT UNIT?* https://www.vectornav.com/resources/inertial-navigation-articles/what-is-an-inertial-measurement-unit-imu, Accesado: Abril-2022.

[131] lakeshore cryotronics, *Hall (Magnetic) Sensors*, https://www.lakeshore.com/products/categories/magnetic-products/hall-(magnetic)-sensors, Accesado: Abril-2022.

[132] Dejan, *What is MEMS? Accelerometer, Gyroscope and Magnetometer with Arduino*, https://howtomechatronics.com/how-it-works/electrical-engineering/mems-accelerometer-gyrocope-magnetometer-arduino/, Accesado: Abril-2022.

[133] Luis Llamas, *MEDIR DISTANCIA CON ARDUINO Y EL SENSOR GP2Y0E03*, https://www.luisllamas.es/medir-distancia-con-arduino-y-el-sensor-gp2y0e03/, Accesado: Julio-2021.

[134] Ricardo Concepción, *Protocolos de comunicación usados en microcontroladores*, https://www.rjconcepcion.com/podcast/protocolos-de-comunicacion-usados-en-microcontroladores/, Accesado: Abril-2022, 2020.

[135] José Guerra Carmenate, *ESP32 Wifi y Bluetooth en un solo chip*, https://programarfacil.com/esp8266/esp32/, Accesado: Febrero-2021.

[136] Directv, *¿Cuál es el alcance de la señal Wifi del router*, https://www.directv.com.co/ayuda/home/asistencia-tecnica/directv-internet/cual-es-el-alcance-de-la-senal-de-wi-fi-del-router.html, Accesado: noviembre-2022.

[137] Cooler Master, *What are the definitions of OVP,OPP,OCP,SCP,OTP and BOP?* https://landing.coolermaster.com/faq/what-are-the-definitions-of-ovp-opp-ocp-scp-otp-and-bop/, Accesado: noviembre-2022.

[138] *Análisis y Control de Sistemas Lineales*, https://studylib.es/doc/9042204/1.2.1clase4modelosdesistemasmec Accesado: Junio-2020.

[139] Fernando di Sciascio, *Estimación de Estados: Observadores*, http://dea.unsj.edu.ar/control2/Clase09a-Observadores%20de%20estados.pdf, Accesado: Agosto - 2021, 2016.

[140] Wikipedia, *Regla del trapecio*, https://es.wikipedia.org/wiki/Regla_del_trapecio, Accesado: Abril - 2022.

[141] Ignacio Mártil de la Plaza, *William Shockley y la invención del transistor*, https://www.bbvaopenmind.com/ciencia/grandes-personajes/william-shockley-y-la-invencion-del-transistor/, Accesado: Junio - 2022, 2017.

[142] Sergio Andrés Castaño Giraldo, *Sistemas Dinámicos Lineales*, https://controlautomaticoeducacion.com/sistemas-dinamicos-lineales/, Accesado: Marzo - 2022.

[143] Mathworks Help Center, *Linear Model Identification*, https://la.mathworks.com/help/ident/linear-model-identification.html?s_tid=CRUX_lftnav, Accesado: Diciembre - 2021.

[144] ——, *Dealing with Multi-Variable Systems: Identification and Analysis*, https://la.mathworks.com/help/ident/ug/dealing-with-multi-variable-systems-identification-and-analysis.html, Accesado: Diciembre - 2021.

[145] ——, *Resolve Fit Value Differences Between Model Identification and compare Command*, https://la.mathworks.com/help/ident/ug/resolve-fit-value-differences-between-model-identification-and-compare.html, Accesado: Diciembre - 2021.