Universidad Nacional de San Luis

Facultad de Ingeniería y Ciencias Agropecuarias

CÁLCULO NUMÉRICO

Trabajo de promoción

DESARROLLO DE UN DRONE CUADRICOPTERO CON MÉTODOS NUMÉRICOS



Alumnos:

* Banffi Echegaray, Joaquín Alejandro
* Bilbao, Juan Ignacio
* López, Emanuel

Profesores:

* Mgtr. Ing. Menuet, Agustín
* Mgtr. Ing Garciarena Ucelay, Martín

Año 2025

Índice

[Resumen 1](#_Toc215654459)

[Introducción 1](#_Toc215654460)

[Drone 1](#_Toc215654461)

[Métodos numéricos usados 1](#_Toc215654462)

[Método de Runge–Kutta de orden 4 (RK4) 2](#_Toc215654463)

[Método de Gradiente Conjugado 2](#_Toc215654464)

[Método de Newton–Raphson 3](#_Toc215654465)

[Regla de Simpson ⅓ 3](#_Toc215654466)

[Spline cúbico natural 3](#_Toc215654467)

[Mínimos Cuadrados 3](#_Toc215654468)

[Desarrollo 4](#_Toc215654469)

[Control Remoto 4](#_Toc215654470)

[Circuito de potencia 5](#_Toc215654471)

[Circuito de control 6](#_Toc215654472)

[Microcontrolador ESP32 (30 pines) 6](#_Toc215654473)

[MPU6050 – Unidad de medición inercial (IMU) para actitud 7](#_Toc215654474)

[ESC-Driver controlador 8](#_Toc215654475)

[Modelado mecánico/matemático del dron 8](#_Toc215654476)

[Generalidades 8](#_Toc215654477)

[Variables de estado y matrices de rotación 9](#_Toc215654478)

[Dinámica traslacional 10](#_Toc215654479)

[Dinámica rotacional 11](#_Toc215654480)

[Modelo de empuje, torques y matriz de mezcla 13](#_Toc215654481)

[Estrategia de control de actitud y empuje 14](#_Toc215654482)

[Arquitectura del sistema 14](#_Toc215654483)

[Controles en cascada para roll y pitch 15](#_Toc215654484)

[Control PID para r 16](#_Toc215654485)

[Dinámica rotacional, uso de las ecuaciones de Euler 17](#_Toc215654486)

[Generación de empuje colectivo 17](#_Toc215654487)

[Matriz Mixer 18](#_Toc215654488)

[Conversor a PWM y realimentación del sistema 19](#_Toc215654489)

[Diseño del sistema 20](#_Toc215654490)

[Diseño 3D 20](#_Toc215654491)

[Diseño de PCB 22](#_Toc215654492)

[PCB de control remoto 22](#_Toc215654493)

[PCB de potencia 23](#_Toc215654494)

[PCB de control 24](#_Toc215654495)

[Simulación 26](#_Toc215654496)

[Ensayos 26](#_Toc215654497)

[Codigo main, principios, uso de RK4 y sistema de control 30](#_Toc215654498)

[Estructura general del programa principal 30](#_Toc215654499)

[Discretización temporal y definición de estados 30](#_Toc215654500)

[Escenarios de referencia y lazo de simulación 31](#_Toc215654501)

[Implementación numérica del controlador en cascada 32](#_Toc215654502)

[Lazo externo: PI de actitud (ángulo → tasa de giro de referencia) 32](#_Toc215654503)

[Lazo interno: PID de tasas (implementación discreta) 33](#_Toc215654504)

[Cálculo de torques físicos con el modelo mecánico 34](#_Toc215654505)

[Empuje total a partir del throttle 34](#_Toc215654506)

[Evaluación del modelo dinámico y método de integración RK4 35](#_Toc215654507)

[Función dron\_derivadas\_fisicas: calcular X\_punto a partir de X y U 35](#_Toc215654508)

[Método de integración rk4\_paso: avanzar el estado en el tiempo 37](#_Toc215654509)

[Explicación mixer (gradiente conjugado) 38](#_Toc215654510)

[Seguridad de corriente (Newton-Raphson) 39](#_Toc215654511)

[Cálculo de corriente y potencia total (Simpson ⅓) 40](#_Toc215654512)

[Gráficas 41](#_Toc215654513)

[Ventana 3D (Spline) 41](#_Toc215654514)

[Resultados de la simulación 43](#_Toc215654515)

[Algoritmo 54](#_Toc215654516)

[Control Remoto 54](#_Toc215654517)

[Arquitectura general del firmware 54](#_Toc215654518)

[Modelo de cada eje (Axis) y autocalibración 55](#_Toc215654519)

[Procesamiento de señal: filtrado, deadzone y auto-trim 55](#_Toc215654520)

[Normalización y mapeo a variables físicas del dron 56](#_Toc215654521)

[Transmisión inalámbrica con ESP-NOW 56](#_Toc215654522)

[Drone 57](#_Toc215654523)

[Implementación en la placa de vuelo (firmware en C++ sobre ESP32) 57](#_Toc215654524)

[Arquitectura general del firmware 57](#_Toc215654525)

[Recepción de referencias vía ESP-NOW y failsafe 58](#_Toc215654526)

[Sensado inercial: IMU MPU6050 + DMP 58](#_Toc215654527)

[Bucle de control en tiempo real (Ts ≈ 4 ms) 58](#_Toc215654528)

[Control de empuje (altura simplificada) 59](#_Toc215654529)

[Control de actitud en roll y pitch (estructuras en cascada) 59](#_Toc215654530)

[Cálculo de torques y mezclador de motores 59](#_Toc215654531)

[Salida a ESC y depuración 60](#_Toc215654532)

[Conclusiones 61](#_Toc215654533)

[Bibliografía 63](#_Toc215654534)

[Anexo I 64](#_Toc215654535)

# 

# Resumen

(Anexo I)

# Introducción

## Drone

Un dron es una aeronave no tripulada capaz de mantener vuelo controlado mediante la acción coordinada de motores, sensores y algoritmos de estabilización. En este proyecto se desarrolló un drone cuadricóptero, es decir, una plataforma aérea que emplea cuatro motores dispuestos simétricamente. Este tipo de configuración permite generar empuje y control mediante variaciones diferenciales en la velocidad de cada hélice, lo que simplifica la mecánica del sistema y favorece la estabilidad.

El diseño adoptado corresponde a una configuración de vuelo en X, donde los brazos del dron forman un ángulo de 45° respecto al eje frontal. Esta disposición mejora la maniobrabilidad y facilita la interpretación de los ejes de referencia del vehículo. En este marco se controlan los tres ángulos principales de la dinámica de un dron: yaw (giro sobre el eje vertical), roll (inclinación lateral) y pitch (inclinación longitudinal). El control de estos ángulos permite mantener la estabilidad y ejecutar maniobras de traslación y rotación en el espacio.

El vehículo está compuesto por tres subsistemas principales. En primer lugar, la placa de potencia, encargada de distribuir la energía hacia los motores y gestionar el consumo eléctrico. En segundo lugar, la placa de control, que integra los sensores inerciales, el microcontrolador principal y la lógica de estabilización del vuelo. Finalmente, el sistema se alimenta mediante una batería LiPo, seleccionada por su elevada densidad energética y capacidad de descarga, características esenciales para aeronaves ligeras.

El dron implementado utiliza un control remoto propio, diseñado específicamente para este trabajo. El mando emplea dos joysticks analógicos que representan los comandos de aceleración y de los ejes yaw, roll y pitch. La información se transmite mediante el protocolo ESP-NOW, lo que habilita una comunicación inalámbrica directa entre dos placas ESP32 sin necesidad de un punto de acceso intermedio. De este modo, el dron recibe en tiempo real las consignas generadas por el operador y las integra en su sistema de control de actitud y de motor.

En conjunto, esta plataforma permitió aplicar los contenidos de la asignatura de Cálculo Numérico dentro de un sistema mecatrónico completo, abarcando desde el modelado matemático y la simulación dinámica hasta la implementación del control.

## Métodos numéricos usados

En el desarrollo del dron y de su simulación dinámica se aplicaron diversos métodos numéricos con el fin de resolver ecuaciones diferenciales, sistemas lineales acoplados, relaciones no lineales del diseño eléctrico e interpolaciones e integraciones necesarias para el análisis del sistema. La elección de cada método respondió a las características matemáticas del problema y a los requisitos de estabilidad y precisión del modelo.

### Método de Runge–Kutta de orden 4 (RK4)

El comportamiento dinámico del cuadricóptero se describió mediante un conjunto de ecuaciones diferenciales ordinarias que modelan la evolución de la actitud, las velocidades angulares y el desplazamiento del vehículo. Para resolver este sistema se utilizó el método de Runge–Kutta de orden cuatro (RK4), cuya formulación:

con:

permitió integrar el modelo con un error global de orden . Esta propiedad resulta esencial para mantener la estabilidad del simulador frente a no linealidades significativas, tales como los acoplamientos entre los ejes rotacionales y la respuesta diferencial de los motores.

### Método de Gradiente Conjugado

La etapa de mezcla de motores resolvió en cada ciclo un sistema lineal derivado de la matriz de asignación de fuerzas y momentos del dron. Este problema puede representarse como:

donde es la matriz del mixer y contiene el empuje total y los torques requeridos. Para resolver dicho sistema se empleó el método de Gradiente Conjugado, que evita la inversión matricial explícita mediante la actualización iterativa

,

con

,

La implementación incorporó un truncamiento decimal a seis cifras significativas en cada operación intermedia, con el propósito de evaluar el comportamiento del método bajo una precisión restringida, similar a la de sistemas embebidos de punto fijo. Este enfoque permitió obtener combinaciones de velocidades de motor consistentes y estables incluso ante variaciones abruptas en las consignas de control.

### Método de Newton–Raphson

En el análisis de seguridad de corriente y en la evaluación de parámetros eléctricos se debió resolver una serie de ecuaciones no lineales relacionadas con la disipación térmica, las caídas de tensión y la potencia máxima admisible de distintos componentes. Estas relaciones no admiten un despeje directo, por lo que se utilizó el método de Newton–Raphson, definido por:

El uso de Newton–Raphson permitió determinar valores críticos como la corriente máxima admisible en función de la resistencia de la pista, o las condiciones de operación seguras para sensores y conductores sujetos a variaciones térmicas. Su convergencia cuadrática lo hace especialmente adecuado para este tipo de ecuaciones continuas y derivables.

### Regla de Simpson ⅓

El análisis energético del dron y el cálculo de corrientes, potencias y energías acumuladas se realizó mediante la regla de Simpson 1/3. Para un conjunto de valores discretos de una función f(t)f(t)f(t), la integral se aproxima como

Esta metodología ofrece una mayor precisión que otros métodos simples de integración, como el trapecio o los rectángulos, especialmente ante señales eléctricas que presentan fluctuaciones o rizado asociado al funcionamiento simultáneo de los motores.

### Spline cúbico natural

La generación y visualización de trayectorias dentro de la simulación se realizó mediante la interpolación con splines cúbicos naturales. Cada tramo se definió como

imponiendo continuidad en la función, en su primera derivada y en su segunda derivada. Las condiciones

,

características del spline natural, permitieron obtener trayectorias suaves y físicamente coherentes tanto para la animación del dron como para la representación de desplazamientos en el entorno gráfico.

### Mínimos Cuadrados

En la etapa de calibración de sensores y ajuste de parámetros se utilizó el método de mínimos cuadrados, que busca minimizar

Las expresiones analíticas para los coeficientes del ajuste

, ,

permitieron compensar offsets, factores de escala y errores sistemáticos en las mediciones. Este procedimiento aseguró que las señales utilizadas en la simulación y en el control del dron representaran adecuadamente el comportamiento del hardware.

# Desarrollo

## Control Remoto

El control remoto desarrollado para el proyecto constituye la interfaz primaria entre el operador y el dron. Su diseño se basó en la integración de dos joysticks analógicos, cada uno conformado internamente por dos potenciómetros lineales dispuestos ortogonalmente. Estos potenciómetros convierten el desplazamiento mecánico de la palanca en una variación de resistencia, lo que se traduce en un cambio proporcional de tensión. De este modo, cada joystick (Fig. 1) proporciona dos ejes analógicos independientes, permitiendo obtener cuatro señales continúas asociadas a: aceleración general (throttle), giro sobre el eje vertical (yaw), inclinación longitudinal (pitch) e inclinación lateral (roll).



Figura 1: Joystick analógico XY

Para el acondicionamiento y muestreo de estas señales se diseñó una placa de control remoto dedicada, basada en un microcontrolador ESP32. La PCB incorpora entradas analógicas estabilizadas, filtrado pasivo para reducir ruido de alta frecuencia y una referencia común de masa que asegura la correcta lectura de los potenciómetros. El ESP32 digitaliza cada canal mediante su ADC interno, aplicando un muestreo periódico y normalizando los valores a rangos adecuados para su transmisión y posterior interpretación en la placa del dron.

La comunicación entre el control remoto y el vehículo se implementó mediante el protocolo ESP-NOW, un sistema de transmisión inalámbrica de baja latencia desarrollado por Espressif. Este protocolo permite el envío de paquetes sin necesidad de un punto de acceso, estableciendo un enlace directo entre dispositivos ESP32. Su principal ventaja es la reducción del overhead asociado a las capas superiores de Wi-Fi, proporcionando tiempos de transmisión del orden de milisegundos, esenciales para maniobras responsivas y estables. Además, ESP-NOW ofrece un alcance típico de entre 20 y 30 metros en interiores y hasta 70–100 metros en exteriores, dependiendo de la potencia de transmisión, la antena y las condiciones del entorno.

El paquete de datos enviado contiene las lecturas analógicas procesadas de los joysticks junto con un identificador de trama para asegurar la integridad y ordenar la información en el receptor. En la placa del dron, un segundo ESP32 opera como nodo receptor, decodificando los mensajes y entregando las consignas al sistema de control. Este esquema de comunicación garantiza una actualización continua del estado del mando, permitiendo que el dron responda en tiempo real a las variaciones introducidas por el piloto.

En conjunto, el control remoto constituye un subsistema robusto, de baja latencia y alta resolución de entrada, imprescindible para la operación segura y precisa del dron cuadricóptero desarrollado.

## Circuito de potencia

El circuito de potencia provee a los motores de la tensión de la batería, y la corriente necesaria.

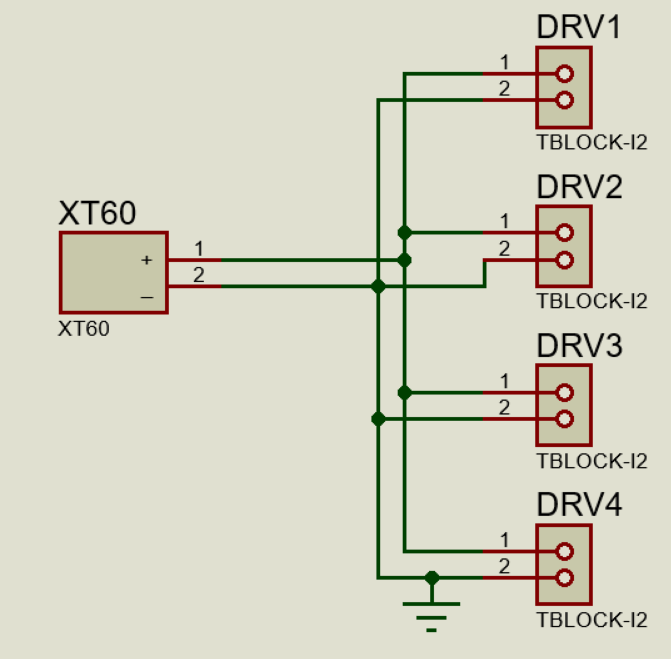


Figura 2: Circuito de potencia

Los controladores electrónicos de velocidad (ESC) poseen dos entradas de potencia, una negativa y otra positiva, para cada ESC se montó una bornera sobre la PDB, estas se representan mediante los SIL que se pueden observar en el diagrama.

El Xt60 hembra correspondiente al esquemático se conecta a la batería de forma directa mediante una extensión hecha con cable siliconado resistente a alta temperatura. El mismo se ensambló con un conector Xt60 macho (este último se enchufa a la hembra montada en placa), y en el otro extremo de la extensión, se empalmo, y soldó un conector Rc Ec3 macho que se enchufa a la hembra de la batería.

A continuación, se puede divisar los conectores Rc Ec3, y Xt60.



Figura 3: Conector EC3



Figura 4: Conector XT60

Las borneras utilizadas en la PDB aguantan 10A cada una, sin embargo, se realizaron pruebas por encima de este valor durante más de 1 minuto, y no se observó calentamiento por parte de estas, ni de las pistas.

## Circuito de control

La placa controladora de vuelo del cuadricóptero se construyó alrededor de tres bloques electrónicos principales: el microcontrolador ESP32 de 30 pines, la unidad inercial MPU6050 y los variadores electrónicos de velocidad (ESC) que gobiernan los motores brushless. En conjunto forman el “la­zo de control físico” que cierra la realimentación entre la dinámica del dron y los algoritmos de control implementados en firmware.

### Microcontrolador ESP32 (30 pines)

El ESP32 se seleccionó como microcontrolador principal de la placa controladora de vuelo porque permite concentrar en un solo dispositivo todas las funciones críticas del dron: adquisición de datos de la IMU, cálculo de las leyes de control, generación de PWM para los motores y recepción de los comandos del piloto. Al ser un microcontrolador de 32 bits con capacidad de operación en tiempo real, dispone de suficiente potencia de cómputo para ejecutar los lazos de control de actitud y de altura a frecuencias del orden de los cientos de hertz, manteniendo al mismo tiempo tareas de comunicación y supervisión.

En el contexto de este proyecto, el ESP32 cumple tres roles principales. Primero, actúa como núcleo de control: recibe las medidas del MPU6050 (velocidades angulares , aceleraciones lineales y ángulos de Euler estimados , ejecuta los algoritmos de control (por ejemplo, PID en cascada) y calcula el empuje total y los torques deseados sobre el dron. Segundo, funciona como generador de señales de mando hacia los ESC: a partir de esos torques y del empuje, el ESP32 aplica la matriz “mixer” para obtener el comando individual de cada motor y lo traduce en señales PWM precisas, respetando los límites de 1000–2000 µs y las restricciones de corriente del sistema.

Finalmente, el ESP32 se encarga de la interfaz con el piloto, recibiendo las órdenes del transmisor remoto mediante ESP-NOW. Esta comunicación inalámbrica de baja latencia permite actualizar continuamente las referencias de roll, pitch, yaw y throttle sin hardware adicional dedicado de radiofrecuencia. De este modo, el ESP32 integra en un único dispositivo la lógica de control, la generación de actuaciones y la recepción de comandos, simplificando la arquitectura electrónica y reduciendo el peso y la complejidad de la controladora de vuelo.



Figura 5: ESP32

### MPU6050 – Unidad de medición inercial (IMU) para actitud

El MPU6050 es una IMU de 6 grados de libertad que integra en un solo encapsulado un acelerómetro triaxial y un giroscopio triaxial. Se lo seleccionó porque proporciona exactamente las magnitudes que necesita la controladora de vuelo para estabilizar el dron:

* Las velocidades angulares alrededor de los ejes del cuerpo, medidas por el giróscopo.
* Las aceleraciones específicas medidas por el acelerómetro.

En este proyecto, el MPU6050 cumple un rol central porque es la fuente de información sobre la orientación y el movimiento del dron. A partir del giróscopo se obtienen las velocidades angulares , que se usan como realimentación en los lazos de control de tasa. Además, el módulo entrega directamente una estimación de los ángulos de Euler (roll, pitch y yaw) en tiempo real, que se utilizan como referencia de la actitud del dron respecto del sistema de referencia inercial.

Es decir, el controlador de vuelo dispone, gracias a este sensor, de:

* para el control de las tasas angulares.
* ya calculados por el propio módulo, para el control de actitud.

El MPU6050 se comunica con el ESP32 mediante el bus I²C, lo que simplifica el cableado y permite trabajar con frecuencias de muestreo elevadas, adecuadas para el control de un cuadricóptero. Desde el punto de vista de implementación, el sistema trata al MPU6050 como un “bloque sensor” que entrega directamente las magnitudes físicas necesarias para el control, sin que sea necesario desarrollar en este trabajo el detalle interno del algoritmo con el que el propio módulo obtiene los ángulos de Euler.

En conjunto, el MPU6050 constituye el núcleo del lazo de realimentación del sistema de control, ya que proporciona en tiempo real las variables de estado (ángulos de Euler y velocidades angulares) que el ESP32 compara contra las referencias para generar las acciones de mando sobre los ESC y, en última instancia, sobre los motores.



Figura 6: Sensor MPU6050

### ESC-Driver controlador

En el contexto del sistema de control, los ESC actúan como actuadores finales del lazo, recibiendo las consignas generadas por el ESP32 (a partir de los errores de actitud medidos por el MPU6050) y ajustando en consecuencia la velocidad de los motores. El ESP32 gobierna cada ESC mediante una señal PWM tipo servo, con anchos de pulso típicos entre 1000 µs y 2000 µs, donde el extremo inferior corresponde al régimen mínimo de giro y el superior al máximo. En el modelo desarrollado se caracteriza esta relación PWM–velocidad angular mediante una ley cuadrática ajustada experimentalmente, lo que permite representar en la simulación cómo una variación en el comando de control se traduce en cambios de empuje en cada hélice.



Figura 7. ESC driver controlador de motores

## Modelado mecánico/matemático del dron

### Generalidades

El dron se modeló como un cuerpo rígido de seis grados de libertad, actuado por cuatro rotores. El objetivo del modelo no es reproducir todos los efectos aerodinámicos finos, sino capturar la dinámica dominante de traslación y rotación para poder diseñar y validar las leyes de control y estimar consumos de potencia.

Se adoptan los siguientes criterios:

* La masa total m se midió en la balanza con el dron completamente armado (batería incluida).
* El tensor de inercia respecto al centro de masa se obtuvo a partir del modelo 3D en *Fusion 360*, usando la herramienta de propiedades físicas. De este modo se incorpora directamente la distribución real de masa (brazos, motores, batería, etc.).

Se consideran dos sistemas de referencia:

* Un sistema inercial fijado al mundo ), con Z vertical.
* Un sistema ligado al cuerpo solidario a la estructura del dron.



Figura 8: Sistema inercial y sistema no inercial

La estructura se considera indeformable (modelo de cuerpo rígido) y no se incluyen, en esta primera etapa, efectos aerodinámicos avanzados (ground effect, arrastre lateral, etc.).

Con estas hipótesis, la dinámica queda gobernada por las leyes de Newton para traslación y por las ecuaciones de Euler para rotación.

### Variables de estado y matrices de rotación

El vector de estado se define como:

donde:

* : Posición del centro de masa en el sistema inercial.
* : Velocidad lineal en el sistema inercial.
* : Ángulos de Euler roll, pitch y yaw del dron.
* : Componentes de la velocidad angular en el sistema del cuerpo.

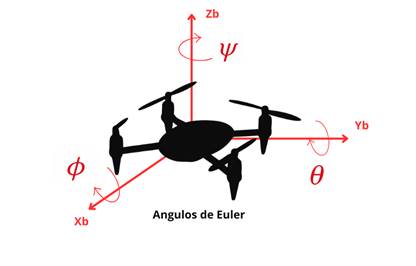
****

Figura 9: Ángulos de Euler en el dron

La relación entre el sistema del cuerpo y el sistema inercial se describe mediante la matriz de rotación:

que transforma vectores del cuerpo al mundo:

Esta matriz se utiliza para proyectar el empuje (definido en el cuerpo) al sistema inercial y para obtener la cinemática de los ángulos a partir de .

### Dinámica traslacional

Sobre el centro de masa actúan principalmente:

* El peso: con en el sistema inercial.
* El empuje total generado por los motores, que en el sistema del cuerpo se modela como



Figura 10: Fuerzas del sistema no inercial

donde ​ es el empuje del motor .

El empuje se proyecta al sistema inercial mediante la matriz de rotación:

y la segunda ley de Newton queda:

Las ecuaciones cinemáticas de posición son simplemente:

Con estas ecuaciones quedan definidos 6 de los 12 estados del dron. En el simulador, estas ecuaciones se utilizan para actualizar la posición y la velocidad del dron en cada instante de tiempo, a partir del estado actual, del empuje total y de la acción de la gravedad.

En el simulador, estas ecuaciones se integran numéricamente mediante un esquema de Runge–Kutta de 4º orden (RK4), utilizando en cada paso de tiempo la actitud actual del dron y el empuje total provisto por el modelo de motores.

### Dinámica rotacional

Para el movimiento de rotación se utilizan las ecuaciones de Euler para cuerpo rígido:

donde:

* es la velocidad angular del dron expresada en el sistema del cuerpo.
* es la aceleración angular del dron expresada en el sistema del cuerpo.
* es el tensor de inercia del dron respecto al centro de masa, obtenido de Fusion 360.
* es el vector de torques netos generado por los cuatro motores.

Despejando:

De esta forma se obtienen .

Para describir la orientación del cuadricóptero se adoptó una parametrización mediante ángulos de Euler en la convención ZYX (yaw–pitch–roll). En esta representación la actitud del cuerpo se obtiene aplicando, en este orden, un giro alrededor del eje Z (como una persona normal que rota sobre su eje Z para cambiar de dirección), luego un giro alrededor del nuevo eje Y (cabeceo) y, finalmente, un giro alrededor del nuevo eje X (alabeo).

Esta elección resulta conveniente por varias razones. En primer lugar, refleja la forma habitual de comandar un dron: el piloto define primero la orientación en (hacia dónde apunta) y, sobre esa referencia, realiza pequeños ajustes de pitch y roll para generar traslaciones horizontales. En segundo lugar, en el régimen de operación normal del vehículo los ángulos de roll y pitch se mantienen relativamente pequeños, mientras que la mayor acumulación de giro se produce en yaw; la convención ZYX mantiene bien condicionadas las ecuaciones en este rango y es la más utilizada en la bibliografía de vehículos aéreos no tripulados, lo que facilita la comparación con otros trabajos.

Bajo esta convención, la relación entre las velocidades angulares del cuerpo y las derivadas de los ángulos de Euler . viene dada por la matriz cinemática :

Donde

En forma equivalente, pueden obtenerse las derivadas de los ángulos de Euler a partir de las velocidades angulares del cuerpo mediante la matriz inversa :

Con

Esta formulación permite escribir de manera consistente las ecuaciones de movimiento del dron tanto en términos de como de y es la base utilizada luego en la implementación numérica del modelo dinámico.

### Modelo de empuje, torques y matriz de mezcla

Cada motor del cuadricóptero genera un empuje aproximadamente proporcional al cuadrado de su velocidad angular:

donde es la constante de empuje obtenida experimentalmente a partir del banco de pruebas motor–hélice–ESC.

Del mismo modo, el par de guiñada (yaw) asociado a cada hélice se modela como:

Donde es la constante de par aerodinámico identificada en los ensayos de banco.

Adoptando una configuración en “X” y tomando como variables de entrada equivalentes el empuje total y los momentos en el sistema de cuerpo,

es posible escribir una relación lineal entre U y el vector de velocidades al cuadrado

Para el cuadro en X utilizado, y asumiendo simetría entre rotores, se obtiene:

Donde es la proyección del brazo de cada motor sobre los ejes e .

Estas expresiones pueden compactarse en forma matricial como:

Con la matriz de mezcla

En esta sección, la matriz M se introduce como parte del modelo de actuadores: describe cómo las velocidades de los cuatro motores se combinan para generar el empuje total y los momentos de control sobre el dron en la configuración en X.  
 En la sección de estrategia de control, esta misma relación se utiliza en sentido inverso: dado un vector de esfuerzos deseado U (salida del controlador en cascada), se resuelve el problema para obtener las velocidades de los motores, respetando las saturaciones físicas de PWM y los límites de corriente.

En la implementación numérica y de control, esta ecuación se invierte mediante un método de gradiente conjugado, lo que permite respetar las saturaciones de PWM y los límites de corriente de cada motor.

## Estrategia de control de actitud y empuje

### Arquitectura del sistema

El sistema de control del dron se organizó en torno a un lazo de realimentación clásico: las referencias de actitud ingresan al controlador, que compara estas consignas con los valores medidos por el sensor inercial MPU6050. A partir de los errores de roll, pitch y yaw, el ESP32 calcula señales de control que, tras pasar por distintos bloques intermedios, terminan modulando la velocidad de giro de cada motor a través de los ESC.

En términos de flujo de información:

* Las entradas del sistema de control son los ángulos de referencia de roll y pitch, la referencia de velocidad de yaw y una referencia Ke de empuje global.
* El MPU6050 cierra el lazo midiendo en tiempo real los ángulos de Euler y las velocidades angulares , que se realimentan a los distintos controladores.
* El ESP32 implementa toda la lógica de control: lazos externos de actitud, lazos internos de velocidad angular, el cálculo de las acciones de control para cada eje y la generación de las consignas hacia los ESC.
* Finalmente, las señales de control se transforman en órdenes de velocidad para cada motor, produciendo el comportamiento de vuelo observado.

Esta estructura permite separar claramente el problema en bloques: primero se corrigen los ángulos deseados, luego se regulan las velocidades angulares del cuerpo y, a partir de allí, se generan las órdenes que se aplican a los actuadores (motores).

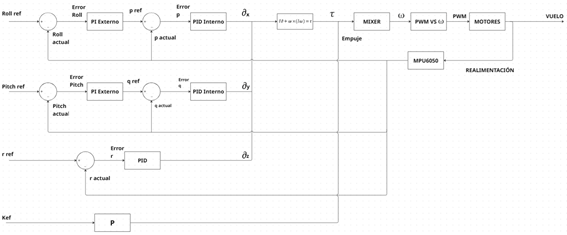


Figura 11: Diagrama en bloques sistema de control

### Controles en cascada para roll y pitch

Para los ejes de roll y pitch se utilizó una estrategia de control en cascada con dos niveles:

1. Lazo externo de actitud, más lento, encargado de que el dron siga los ángulos de referencia.
2. Lazo interno de velocidad angular, más rápido, que actúa sobre las velocidades p y q para ejecutar las correcciones indicadas por el lazo externo.

**Lazos externos (actitud)** Los ángulos medidos por el MPU6050 se comparan con sus consignas:

Cada error alimenta un controlador PI externo:

cuyas salidas son las velocidades angulares de referencia alrededor de los ejes Xb e Yb del dron. Es decir, el lazo externo decide “qué tan rápido” debe girar el vehículo para corregir el error de ángulo.

**Lazos internos (velocidad angular)**

Las velocidades reales p y q medidas por el MPU6050 se comparan con sus referencias:

Estos errores alimentan controladores PID internos, más rápidos, que generan las señales de mando y . En la implementación, estas señales se interpretan como aceleraciones angulares deseadas o esfuerzos de control alrededor de cada eje, que luego se transformarán en torques mediante el modelo dinámico del cuerpo rígido.

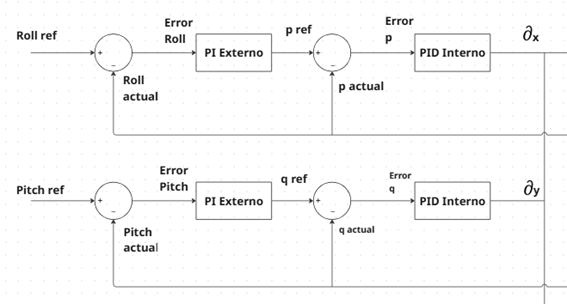


Figura 12: 1er bloque de control, lazos externos y externos de Roll y pitch

### Control PID para r

A diferencia de roll y pitch, el eje de yaw se controla con un único lazo PID actuando directamente sobre la velocidad angular r. En este caso no se implementa un lazo externo de ángulo las referencias provienen del joystick del transmisor como órdenes de “girar a derecha/izquierda” con cierta velocidad, o bien se fijan en cuando se desea mantener el rumbo.

El MPU6050 entrega la velocidad angular real alrededor del eje z del dron, r. Esta se compara con la referencia:

El error alimenta un controlador PID:

cuyo resultado representa la acción de control para el canal de guiñada. En el diagrama de bloques, esta señal sale del PID de yaw y se agrupa junto con y para ser utilizada más adelante en el bloque dinámico (ecuaciones de cuerpo rígido) que generará el torque alrededor del eje z.

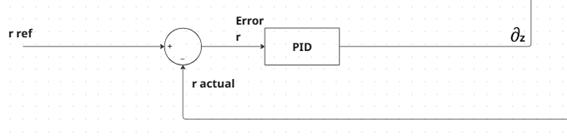


Figura 13: 2do bloque, control PID de r

### Dinámica rotacional, uso de las ecuaciones de Euler

En este bloque se modela la dinámica de rotación del dron como cuerpo rígido en el sistema solidario a los ejes B. La relación fundamental entre aceleraciones angulares, velocidades angulares, tensor de inercia y torques está dada por las ecuaciones de Euler:

La salida de este bloque es el vector de torques totales que deben ser aplicados al cuerpo (suma de los aportados por los cuatro motores).

En el diagrama de bloques, las salidas de los controladores ,se interpretan como aceleraciones angulares deseadas:

De este modo, el Bloque 4 actúa como nexo entre los controladores (que trabajan sobre velocidades/aceleraciones angulares) y el modelo físico del dron, expresando explícitamente qué torques debe generar el sistema de propulsión para imponer la dinámica rotacional deseada.

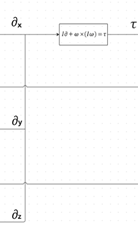


Figura 14: 3er bloque, dinámica rotacional

### Generación de empuje colectivo

Antes de aplicar la matriz mixer es necesario definir cómo se calcula la fuerza de empuje total que debe generar el conjunto de motores. En el diagrama de bloques esta función está representada por el bloque P, ubicado en la rama inferior, que ajusta el empuje alrededor de la condición de hover.

El punto de partida es el peso del dron, mg, que corresponde al empuje necesario para mantenerse en vuelo estacionario. Sobre ese valor se aplica una pequeña corrección relativa Ke​, obtenida a partir de una acción proporcional (bloque P) a la consigna de “subir/bajar” enviada por el piloto mediante control remoto. De este modo, el empuje total se modela como:

donde Ke​ se acota típicamente en un rango reducido ±0,1.

* Si Ke=0, el empuje es exactamente mg y el dron se mantiene próximo al hover.
* Si Ke>0, el empuje supera al peso y el dron tiende a ascender.
* Si Ke<0, el empuje es menor que mg y el dron tiende a descender.

La salida de este bloque es el escalar , que se combina luego con los torques provenientes del Bloque 4 para formar el vector de entrada al mixer.

Este diseño separa claramente el control de empuje colectivo (altura) del control de actitud (roll, pitch y yaw), simplificando el ajuste de los controladores y permitiendo trabajar alrededor de una condición de vuelo estacionario bien definida.



Figura 15: 4to bloque, cálculo de empuje

### Matriz Mixer

En este bloque se distribuyen las órdenes de control generales del dron hacia cada uno de los cuatro rotores.  
 Agrupamos las entradas del mixer en el vector U

Donde es el empuje total deseado y son los torques deseados alrededor de los ejes del cuerpo. La salida del bloque son las velocidades angulares al cuadrado de cada motor:

Para un cuadricóptero en configuración X, la relación estática entre estos vectores puede escribirse como

En lugar de resolver este sistema realizamos la relación inversa para obtener las velocidades angulares al cuadrado en función de las señales de control:

Con:

Con la implementación de obtendremos valores de , a los cuales se les aplica la raíz cuadrada y se obtiene el valor , el cual ingresa al siguiente bloque conversor para obtener PWM. Cabe destacar que al momento de la simulación este cambio de a se realiza con el método numérico del gradiente conjugado, mientras que en el microcontrolador si se usa directamente .

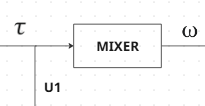


Figura 16: 5to bloque, Matriz mixer

### Conversor a PWM y realimentación del sistema

En el Bloque final se agrupan las etapas que convierten las velocidades angulares deseadas de cada motor en el vuelo real del dron, cerrando el lazo de control con la medida del MPU6050.

En primer lugar, el bloque “PWM vs ” representa la relación estática entre la velocidad de giro del motor y el ancho de pulso PWM que entiende el ESC. A partir de las calculadas por el mixer, se genera para cada canal un comando PWM tipo servo (típicamente entre 1000 y 2000 µs). En la práctica y en la simulación se modela mediante una curva ajustada que captura el comportamiento real de los actuadores, de modo que:

Estos PWM alimentan el bloque “Motores”, que agrupa el conjunto ESC-motor-hélice. En el modelo físico, esta etapa se encarga de transformar el PWM en empuje y torques aerodinámicos, que finalmente producen el vuelo (movimiento del dron en el espacio).

Por último, el bloque MPU6050 cierra el lazo de realimentación: a partir del movimiento resultante del dron, el sensor mide las aceleraciones lineales y las velocidades angulares, y a partir de ellas se estiman los ángulos de Euler. Estas magnitudes se envían nuevamente al ESP32, que las utiliza como estados actuales en los lazos de control de roll, pitch y yaw, cerrando así el ciclo.  
 De esta forma, el Bloque final conecta la salida del diseño de control con el comportamiento real del dron y aporta la información necesaria para mantener estabilizado el sistema.

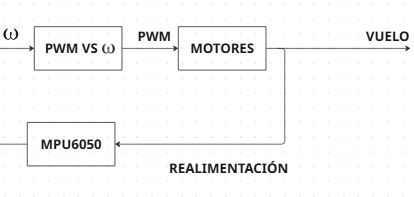


Figura 17: 6to bloque, Conversión PWM y realimentación

# Diseño del sistema

## Diseño 3D

El diseño mecánico del control remoto se desarrolló completamente en CAD 3D, teniendo como objetivo principal lograr una estructura ergonómica, compacta y adecuada para alojar tanto la electrónica como los mecanismos de los joysticks. Las imágenes del modelo muestran la distribución interna de los componentes, la carcasa exterior y las piezas de soporte que garantizan la correcta fijación del hardware.

La carcasa se diseñó en dos mitades principales que permiten un montaje sencillo y seguro. Su geometría incluye refuerzos internos, nervaduras y espesores mínimos calculados para mantener rigidez estructural sin incrementar innecesariamente el peso total. Los soportes internos permiten alojar el ESP32, el módulo de transmisión, la batería y el cableado, manteniendo un orden interno adecuado y reduciendo la posibilidad de fatiga mecánica en los cables.

Los joysticks se montan sobre plataformas dedicadas, con altura y separación optimizadas para una operación confortable. El diseño también incluye orificios de ventilación, pasacables y espacio para fijación con tornillería estándar, lo que permite desmontar el control remoto para mantenimiento o reemplazo de componentes.

Finalmente, el modelo 3D contempla tolerancias apropiadas para impresión en tecnología FDM, evitando deformaciones o interferencias durante el ensamblado. El resultado es un control remoto robusto, funcional y adecuado tanto para ensayos como para uso operativo del dron.

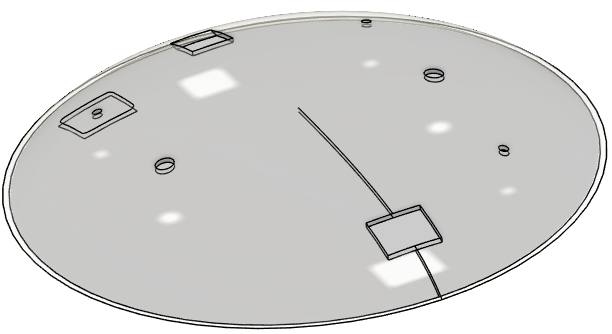


Figura 18: Tapa superior



Figura 19: Tapa inferior



Figura 20: Soporte de motores

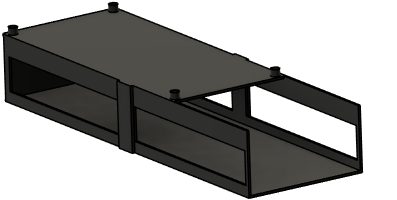


Figura 21: Soporte de batería



Figura 22: Modelo completo

## Diseño de PCB

### PCB de control remoto

La figura 23 muestra la placa de circuito impreso (PCB) diseñada específicamente para el control remoto del dron. Su estructura se organizó para integrar de manera compacta el ESP32, los joysticks analógicos y los elementos de protección y acondicionamiento necesarios para garantizar lecturas estables en los conversores analógico-digitales del microcontrolador.

El ESP32 se ubica en la zona central de la PCB (conectores J6 y J7), permitiendo un ruteo simétrico hacia ambos laterales, donde se conectan los dos joysticks (conectores J4 y J2). Cada joystick entrega dos señales analógicas correspondientes a sus ejes X e Y. Estas señales atraviesan resistencias en serie (R1–R4), cuyo propósito principal es limitar picos de corriente transitorios hacia los pines ADC del ESP32. Este acondicionamiento resulta relevante dado que los potenciómetros pueden generar variaciones abruptas o ruido mecánico durante el movimiento, y se busca evitar sobrecargas o fluctuaciones bruscas en la etapa de lectura.

La placa se alimenta mediante un portapilas externo que utiliza cuatro baterías AA de 1,5 V, entregando un total nominal de 6 V. Dado que el ESP32 opera a 3,3 V, la PCB incorpora un módulo step-down buck LM2596, encargado de reducir la tensión de entrada a un nivel estable y regulado de 3,3 V. Este regulador presenta alta eficiencia y capacidad de corriente suficiente para alimentar tanto al microcontrolador como a los joysticks, asegurando autonomía y estabilidad operativa durante el uso continuo.

En el diseño del ruteo se priorizó trayectos cortos y planos de masa amplios para minimizar el ruido eléctrico en las líneas analógicas. Asimismo, la distribución lateral de los conectores permite una ergonomía adecuada en el dispositivo final, manteniendo los joysticks alineados con la posición natural de las manos del operador.

En conjunto, esta PCB constituye una plataforma robusta y optimizada para la adquisición de señales de mando y su transmisión mediante ESP-NOW hacia la placa principal del dron.

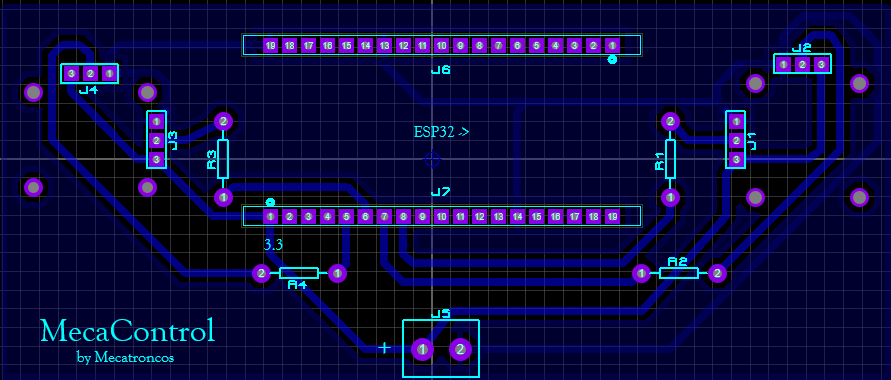


Figura 23: Diseño PCB control remoto

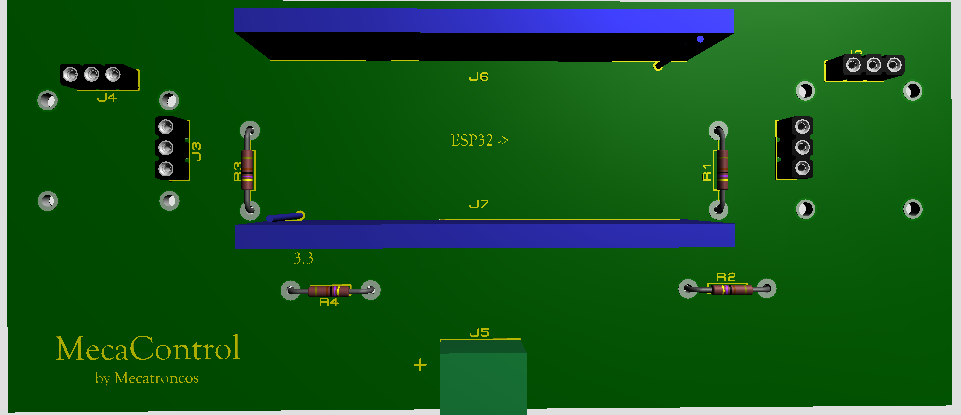


Figura 24: Vista superior PCB control remoto, modelo 3d

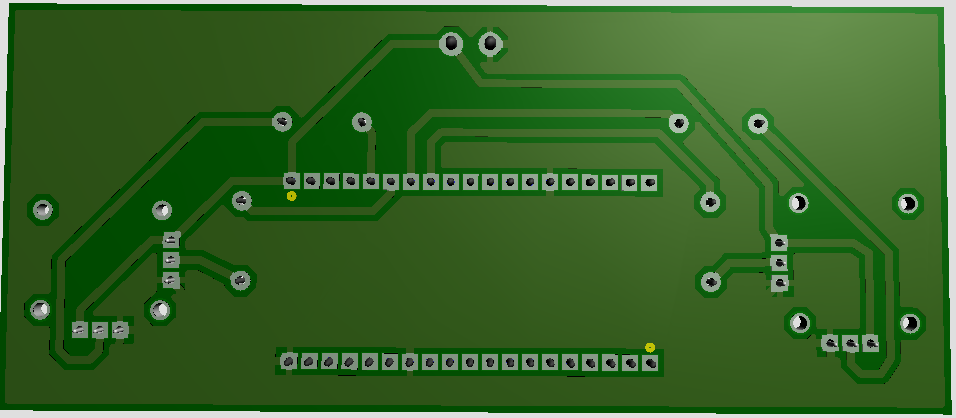


Figura 25: Vista inferior PCB control remoto, modelo 3d

### PCB de potencia

La Placa de Circuito Impreso (PCB) denominada "PDB MecaDron" funciona como el nodo principal de alimentación del sistema de propulsión del dron. Su diseño está críticamente enfocado en la gestión de alta corriente y la distribución eficiente de la energía desde la fuente principal hacia los sistemas motrices. La placa incorpora un conector XT60 (o un footprint equivalente), que está estandarizado en aplicaciones de alta potencia y permite la conexión directa de la batería de propulsión. Este conector es seleccionado por su capacidad para manejar altas corrientes de descarga con una resistencia de contacto mínima, lo cual es fundamental para asegurar un suministro de energía ininterrumpido y fiable durante el vuelo.

La función central de la PDB es distribuir esta energía a los cuatro Controladores Electrónicos de Velocidad (ESC). Para ello, se han dispuesto cuatro conjuntos de pads de soldadura etiquetados como DRV1 a DRV4, que sirven como interfaces de alimentación para los respectivos drivers de motor. La ingeniería del layout es un factor determinante en esta PCB: se han utilizado pistas de cobre de máxima anchura y la ruta más corta posible para enlazar el conector XT60 con cada uno de los pads de salida. Esta estrategia de diseño es vital para minimizar la resistencia eléctrica, lo que se traduce en una reducción de las pérdidas de potencia por efecto Joule y una limitación en el calentamiento de la placa. Además, las rutas cortas contribuyen a mantener una baja inductancia parásita en las líneas de alimentación, aspecto crucial para la estabilidad del voltaje bajo las variaciones bruscas de corriente demandadas por los motores durante las maniobras. Finalmente, la inclusión de orificios de montaje en las esquinas asegura la fijación mecánica de la PDB al chasis.

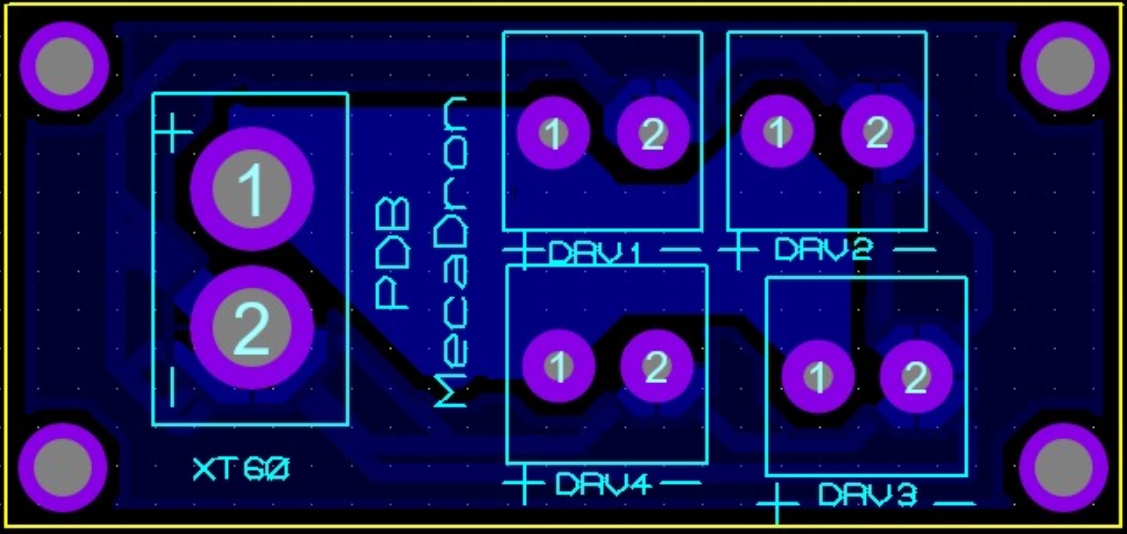


Figura 26: Diseño PCB placa de distribución de potencia

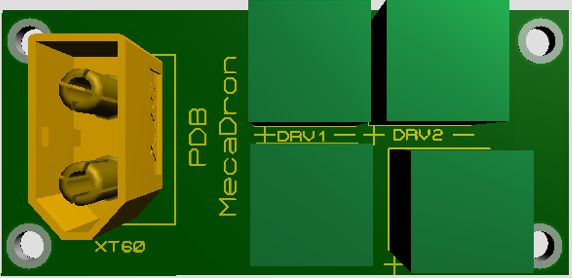


Figura 27: Vista superior PCB placa de distribución de potencia, modelo 3D

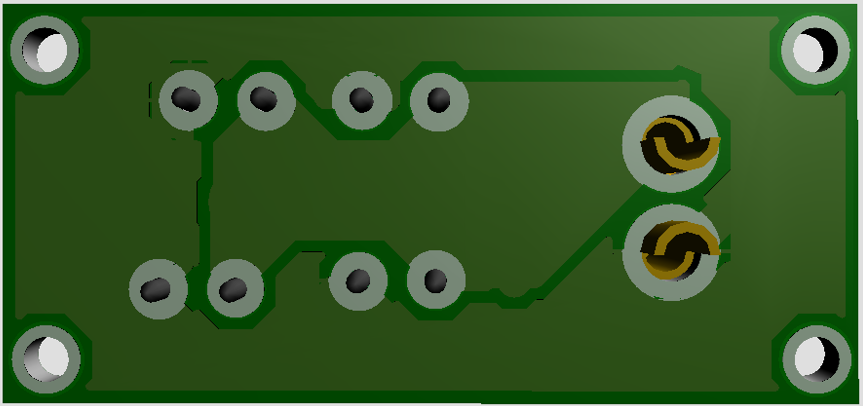


Figura 28: Vista inferior PCB placa de distribución de potencia, modelo 3D

### PCB de control

La Placa de Circuito Impreso (PCB) denominada "Control 1.1 MecaDron" ha sido diseñada como el hardware central del sistema de vuelo del dron. Su principal objetivo es integrar de forma compacta y eficiente los elementos críticos para la navegación y propulsión: el microcontrolador, el sensor de altitud y la interfaz de potencia hacia los motores.

El Microcontrolador Principal (ESP32), ubicado en la zona inferior de la placa, es el encargado de la gestión integral de la aeronave. Este módulo recibe los datos de comando de la unidad de control remoto a través de un enlace de comunicación inalámbrica de baja latencia (típicamente ESP-NOW) y ejecuta el algoritmo de control de vuelo (PID). Las salidas del ESP32 se configuran para generar las señales de Modulación por Ancho de Pulso (PWM), fundamentales para el control de velocidad de los motores.

Para la estabilización y el mantenimiento de la orientación, la PCB integra una Unidad de Medición Inercial (MPU6050). Esta se conecta al microcontrolador mediante el protocolo I²C y proporciona mediciones de aceleración y velocidad angular en los tres ejes. En el diseño del layout se priorizó la minimización de la longitud de las pistas para asegurar la integridad de la comunicación y la precisión de los datos de actitud que alimentan el algoritmo de control.

La interfaz de potencia se realiza a través de cuatro headers de conexión etiquetados como DRV1 a DRV4. Estos conectores sirven como punto de acople para los Controladores Electrónicos de Velocidad (ESC) o drivers de los cuatro motores del cuadricóptero. Cada header cuenta con la línea de alimentación de potencia y una línea de señal (PWM) dedicada, proveniente de un pin específico del ESP32. Un aspecto crítico del diseño es la utilización de pistas de potencia anchas para manejar las altas corrientes demandadas por los ESC, lo que minimiza la caída de tensión y reduce la generación de calor en la propia placa.

Finalmente, para garantizar la fiabilidad del sistema en un entorno propenso a vibraciones y ruido eléctrico, la PCB incorpora planos de masa (GND) amplios en ambas capas. Esta técnica de ingeniería reduce significativamente la interferencia electromagnética (EMI), proporciona una trayectoria de retorno de baja impedancia para las corrientes conmutadas de los motores y ayuda en la disipación térmica general del sistema, asegurando una plataforma de vuelo robusta y estable.

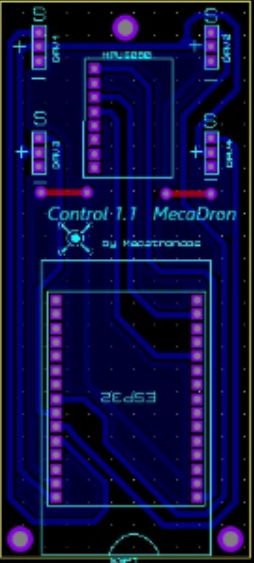


Figura 29: Diseño PCB placa de control

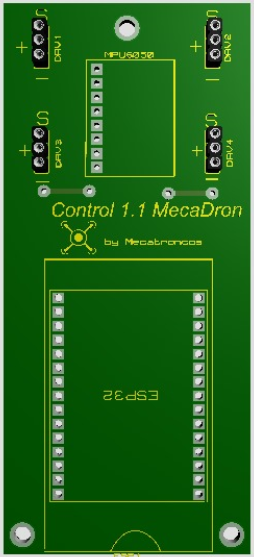


Figura 30: Vista superior PCB placa de control, modelo 3D

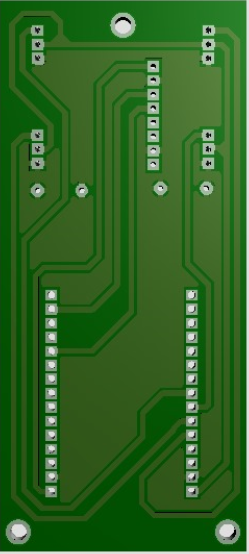


Figura 31: Vista inferior PCB placa de control, modelo 3D

# Simulación

Con el objetivo de validar la estrategia de control propuesta y cuantificar el comportamiento dinámico del dron antes de la implementación física, se desarrolló una simulación numérica completa del sistema. La simulación permite combinar el modelo mecánico (traslación y rotación del cuerpo rígido) con los modelos de los motores, ESC y lazos de control, de manera de anticipar cómo responde el vehículo ante distintos escalones de referencia y perturbaciones sin poner en riesgo el hardware real.

Trabajar en este entorno virtual hace posible ajustar ganancias de los controladores, verificar márgenes de estabilidad, analizar consumos de corriente y empuje disponible, y comparar distintos escenarios de vuelo (hover, cambios de actitud, giros en yaw) de forma rápida y repetible. Solo una vez que el modelo muestra un desempeño aceptable se trasladan los parámetros al ESP32 y al dron físico, reduciendo significativamente el tiempo de puesta a punto y la posibilidad de fallas durante las primeras pruebas de vuelo.

## Ensayos

Los ensayos de medición llevados a cabo tuvieron como finalidad la devolución de los siguientes parámetros:

* RPM
* Amperaje instantáneo
* Tensión de batería
* Temperatura del driver
* Fuerza de empuje de cada motor

Con estos modelos se calculará consumo total (Ah/Wh) vía integración por Simpson 1/3 dentro de la simulación.

Equipo de medición y banco:

* ESP32 (generación de PWM en μS)
* Balanza (empuje estático)
* Tacómetro
* Pinza amperométrica (corriente y temperatura)
* Medición de tensión de la batería
* Soporte rígido y seguro para fijar motor/hélice sobre balanza

La numeración para los motores fue la siguiente, se comenzó por el motor 1, y se continuó por el motor 2.

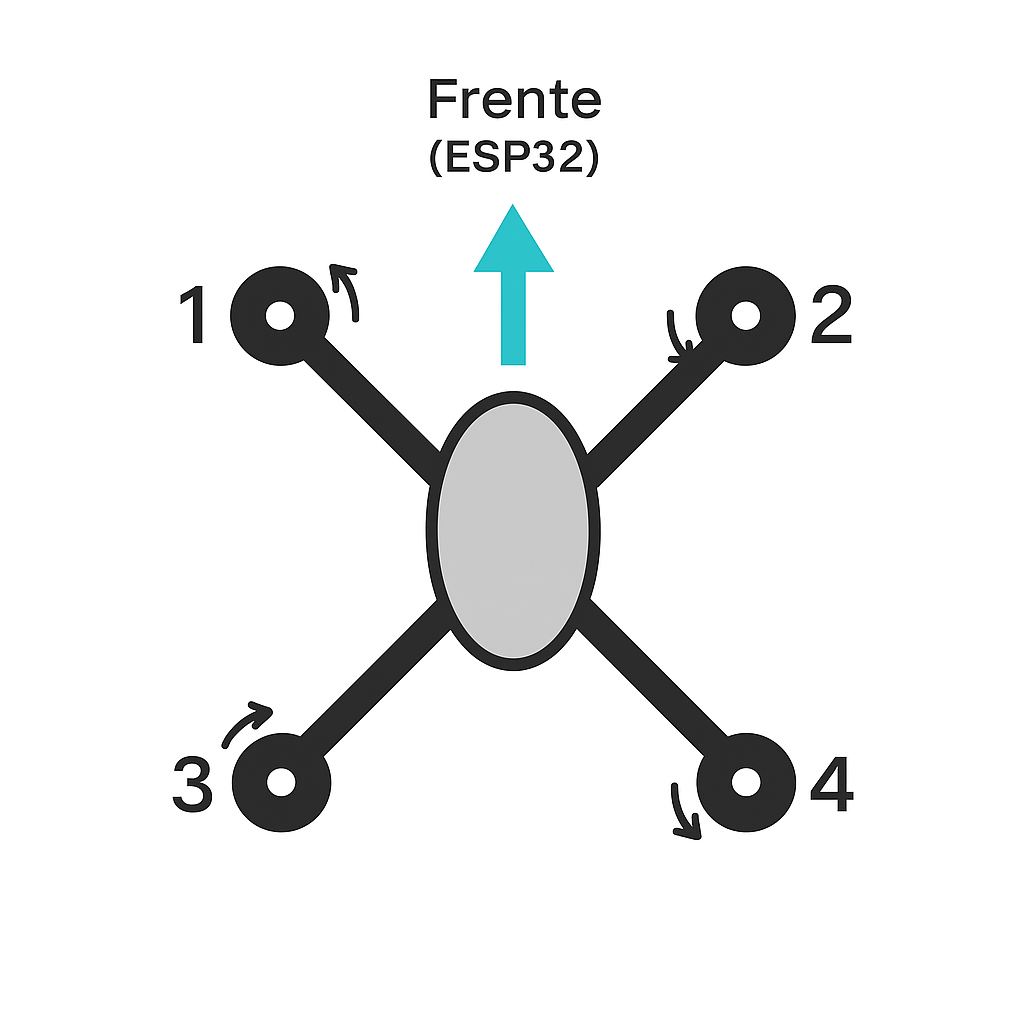


Figura 32: Numeración de motores

El procedimiento consistió en realizar un barrido desde los 1000 us hasta los 2000 us con paso de 50 us. Por cada paso se tomaban 3 mediciones, esto con la finalidad de realizar un promedio, disminuyendo el error debido a inestabilidades en el sistema de medición.

Durante la medición del segundo motor, a más de 14 A, un fallo en el devanado terminó en un corto circuito de uno de los mosfets de la ESC. Es por ello que para el modelado se decidió utilizar la tabla generada con el primer motor, y no seguir poniendo en riesgo más equipos.

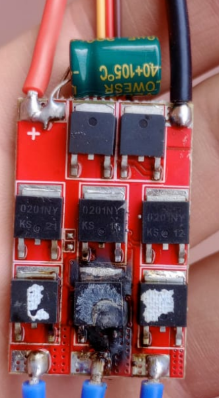


Figura 33: Falla en driver ESC

En el código Ensayos\_calibracion.m se adquieren los datos desde una hoja de datos de excel, y se arman arreglos con los mismos. Los datos compilados se ajustan mediante el algoritmo de Minimos\_cuadrados.m, basándose en regresión cuadrática, se calcula la pendiente de relación entre y la fuerza de empuje, del mismo modo se realiza con la corriente consumida, y el PWM, de modo que, dado un intervalo, se obtiene una aproximación, que relaciona todos los parámetros relacionados con la velocidad angular.

En la línea 208 del código Ensayos\_calibracion.m, se calcula la constante Ke del motor mediante la siguiente fórmula:

Donde Kv es la constante de RPM por Voltio para el brushless.

Luego, se realiza una última regresión para conocer la relación torque y omega cuadrado.

Donde Kq es el par de arrastre que se calcula como c1\*Ke, donde c1 es un factor perteneciente a la regresión exclusiva de la intensidad respecto a omega cuadrado.

Finalmente, la función devuelve las gráficas fuerza de empuje, PWM, corriente, y par de arrastre o torque para omega en algunos casos y omega cuadrado en otros.

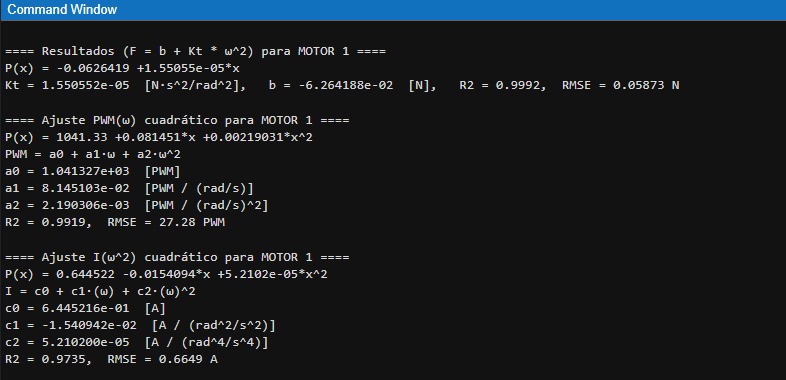


Figura 34: Command Window

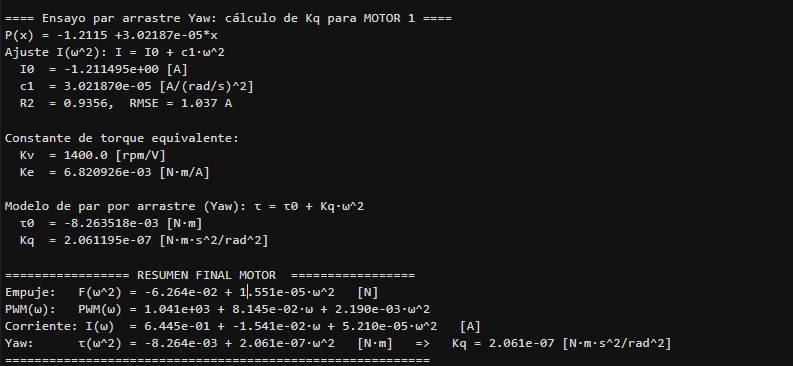


Figura 35: Command Window

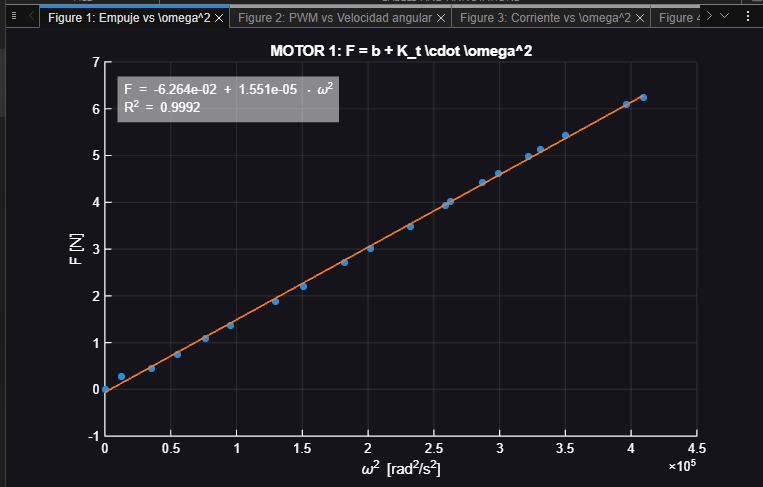


Figura 36: Gráfica empuje vs

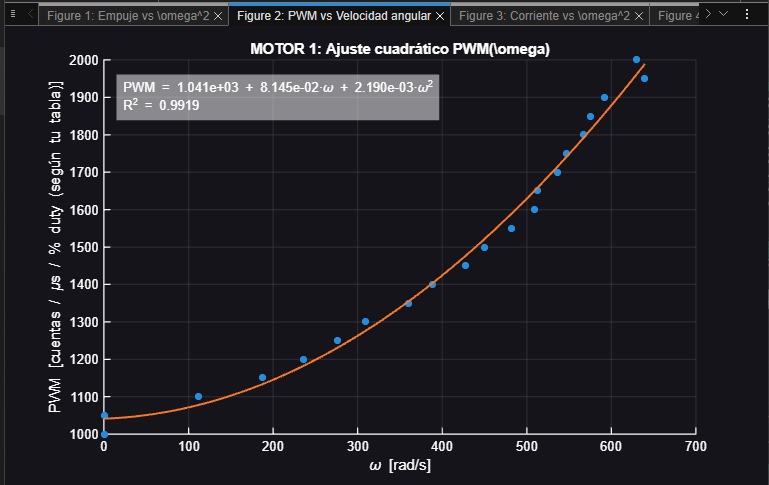


Figura 37: Gráfica ajuste PWM

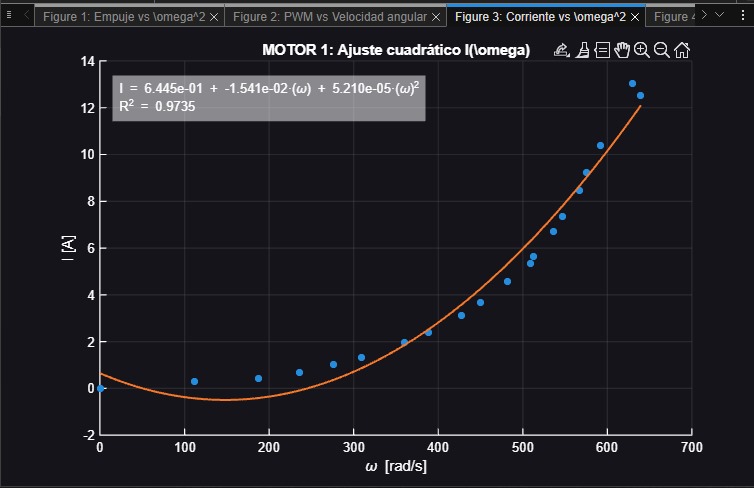


Figura 38: Gráfica ajuste cuadrático

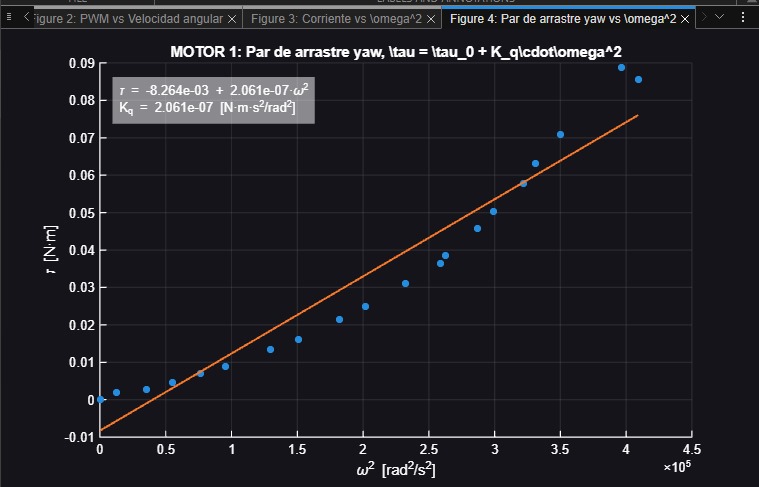


Figura 39: Gráfica par de arrastre

## Codigo main, principios, uso de RK4 y sistema de control

### Estructura general del programa principal

La función simulacion\_dron\_main implementa el “esqueleto” de la simulación numérica del dron. Su objetivo no es solo resolver las ecuaciones de movimiento, sino también reproducir el comportamiento completo del sistema de control y de los actuadores (motores, límites de PWM, consumo eléctrico) sobre una malla de tiempo discreta.

En primer lugar, el programa configura un tema oscuro global para todas las figuras de MATLAB. Esto no afecta al resultado numérico, pero permite visualizar de forma homogénea los distintos gráficos de estados, controles y consumos generados al final de la simulación.

A continuación, se definen los parámetros físicos del vehículo: la masa, la gravedad, el tensor de inercia (matriz ) y la distancia de los motores al centro de masa. Estos parámetros se agrupan en una estructura phys (phys.m, phys.g, phys.I) que se pasa como argumento a las funciones de dinámica y al integrador numérico. Esta encapsulación facilita que el mismo esquema de integración pueda reutilizarse, aunque cambien las propiedades físicas del dron.

También se inicializan las constantes que vinculan la velocidad angular de las hélices y las magnitudes físicas que interesan al modelo: Kt para el empuje y Kq para el par de guiñada. A partir de ensayos previos se ajustan polinomios que relacionan el PWM de cada motor con su velocidad angular (pwm\_poly) y, de forma análoga, polinomios que aproximan la corriente consumida en función de la velocidad (corriente\_cuadratica). Esto permite evaluar numéricamente el efecto de saturaciones de PWM y de límites de corriente sin introducir modelos de motores excesivamente complicados.

Por último, se llama a inicio\_parametros\_control(), que devuelve las ganancias del controlador en cascada (la parte de diseño del PI/PID se detalla en otra sección del trabajo). El programa principal solamente utiliza esos parámetros como “caja negra”: en cada instante, entrega al controlador las referencias y el estado actual, y recibe como salida el vector de entradas físicas.

### Discretización temporal y definición de estados

La dinámica del dron es continua en el tiempo, pero la simulación numérica debe realizarse sobre una malla temporal discreta. Para ello se establece un paso de integración fijo (pasos de h = 0.01 segundos) y el usuario ingresa el tiempo total de simulación T\_total. Con esos datos se calcula el número de pasos

y se construye el vector de tiempos

sobre el que se evaluarán todos los estados y señales. El vector de estados X agrupa en una sola columna las doce variables del modelo:

* Posición del centro de masa en el sistema inercial: X, Y, Z.
* Velocidades lineales: .
* Ángulos de Euler (actitud): roll, pitch, yaw.
* Velocidades angulares del cuerpo: p, q, r.

Es decir

En la simulación se parte de una condición de reposo en el origen, con el dron nivelado (todos los estados iniciales en cero). Para poder analizar después la evolución de la simulación, se preasignan matrices de historial como X\_hist, U\_hist, refs\_hist, etc., que almacenan, para cada instante de la malla temporal, los estados, entradas de control y magnitudes auxiliares de interés (torques, corrientes, potencia, etc.). Esta preasignación es importante desde el punto de vista numérico porque evita realocaciones dinámicas de memoria dentro del bucle principal.

### Escenarios de referencia y lazo de simulación

La lógica de la simulación se implementa en un bucle for k = 2:N. Cada iteración del bucle representa un paso temporal de tamaño . El índice k se asocia al tiempo ti = t(k-1), y sobre ese tiempo discreto se van aplicando distintos escenarios de referencia que emulan el uso de un control remoto real.

En la práctica, lo que se hace es “programar” cómo se movería cada stick del transmisor a lo largo del tiempo: por ejemplo, en cierto intervalo se aplica un pequeño cambio en la referencia de roll, en otro se modifica el throttle para simular un intento de despegue o de vuelo estacionario, y en otro se ordena una rotación en guiñada. Todos estos perfiles de referencia se codifican modificando las variables refs.roll\_ref, refs.pitch\_ref, refs.r\_ref y refs.throttle en función del tiempo .

Desde el punto de vista numérico, estas referencias actúan como entradas en lazo cerrado: en cada instante discreto, el controlador compara las referencias con los estados actuales y genera las entradas físicas que alimentan al modelo dinámico. De este modo, la simulación reproduce la interacción entre un sistema continuo (el dron) y un “piloto virtual” que maneja los sticks del control remoto de forma predefinida, pero actualizando el control en pasos de segundos.

Al final de cada iteración, justo antes de avanzar al siguiente paso de integración, se llama al controlador:

*% ==== Control en cascada → U = [U1; τx; τy; τz] ====*

*[U, parametros\_ctrl, mon] = control\_cascada(refs, X, phys, parametros\_ctrl, h);*

Aquí control\_cascada utiliza el estado actual X, las referencias refs y los parámetros físicos phys para calcular el vector de entradas físicas.

### Implementación numérica del controlador en cascada

El controlador del dron se implementa en la función control\_cascada, que se llama en cada paso de simulación con periodo de muestreo. Desde el punto de vista numérico, puede verse como un conjunto de controladores PI y PID en tiempo discreto, actualizados a partir del error en ese instante y del estado interno del controlador (integradores y errores previos).

La función recibe las referencias refs (equivalentes a la posición de los sticks del control remoto), el estado actual X, los parámetros físicos phys y una estructura params donde se almacenan las ganancias y los estados internos de los controladores (integradores, errores previos, límites, etc.). Devuelve el vector de entradas físicas:

junto con la estructura params actualizada y una estructura de monitoreo mon.

### Lazo externo: PI de actitud (ángulo → tasa de giro de referencia)

El primer bloque del controlador genera referencias de tasa de giro (p\_ref, q\_ref) a partir de los errores de ángulo en roll y pitch. A nivel de código:

*phi = X(7); th = X(8);*

*e\_phi = wrapAngle(refs.roll\_ref - phi);*

*e\_th = wrapAngle(refs.pitch\_ref - th);*

*[p\_ref, params.int.roll] = actualizar\_PI(params.roll\_PI, e\_phi, params.int.roll, h);*

*[q\_ref, params.int.pitch] = actualizar\_PI(params.pitch\_PI, e\_th, params.int.pitch, h);*

Se calcula el error de actitud como diferencia entre el ángulo de referencia y el ángulo medido, usando wrapAngle para evitar saltos bruscos cuando se pasa de π a -π.

La función actualizar\_PI implementa un controlador PI discreto de la forma:

y luego aplica saturaciones tanto al integrador como a la salida.

En código, la lógica principal es:

*I = I\_in + cfg.Ki\*err\*h;*

*I = min(max(I, cfg.I\_min), cfg.I\_max);*

*u\_base = cfg.Kp\*err + I;*

*u = min(max(u\_base, cfg.u\_min), cfg.u\_max);*

Además, se incluye un anti–windup numérico tipo “undo integration”: si la salida U ha saturado y el signo del error tendería a seguir empujando hacia fuera del límite, se deshace la integración del último paso:

*if ( (sat\_hi && err > 0) || (sat\_lo && err < 0) )*

*I = I - cfg.Ki\*err\*h;*

*...*

*end*

Desde el punto de vista de métodos numéricos, esto es importante porque el integrador se actualiza con una regla explícita de Euler (). Sin este mecanismo, si la salida se mantiene saturada mucho tiempo, el término integral podría crecer artificialmente y provocar sobrepasos muy grandes cuando el error cambie de signo.

El resultado de este lazo externo son las tasas de giro deseadas p\_ref y q\_ref, utilizadas luego por el lazo interno.

### Lazo interno: PID de tasas (implementación discreta)

El lazo interno trabaja sobre las tasas de giro. A partir de las referencias p\_ref, q\_ref y refs.r\_ref, se calculan los errores de tasa:

*err\_p = p\_ref - p;*

*err\_q = q\_ref - q;*

*err\_r = refs.r\_ref - r;*

Cada uno de estos errores se procesa con un PID discreto implementado en la función actualizar\_PID, que emula el mismo patrón utilizado en el firmware en C++:

*[alpha\_p, params.int.p, params.err\_prev.p] = actualizar\_PID(params.p\_PID, err\_p, params.int.p, params.err\_prev.p, h);*

Aunque el detalle exacto de actualizar\_PID está encapsulado, numéricamente sigue la misma idea que el PI:

* Acumula un integrador con un esquema tipo Euler.
* Calcula un término derivativo aproximando la derivada del error por diferencia finita:
* Combina las tres contribuciones (proporcional, integral y derivativa), y aplica saturaciones y anti-windup análogo al caso PI.

Este lazo interno tiene una dinámica más rápida (en términos de ganancias) y “traduce” errores de tasa a aceleraciones angulares deseadas

*alpha\_des = [alpha\_p; alpha\_q; alpha\_r]*

Que luego se convertirán en torques físicos usando el modelo rígido del dron.

### Cálculo de torques físicos con el modelo mecánico

Una vez obtenidas las aceleraciones angulares deseadas, el controlador calcula los torques requeridos a partir del modelo dinámico de rotación:

*I = phys.I;*

*w = [p;q;r];*

*tau\_unsat = I\*alpha\_des + cross(w, I\*w);*

Numéricamente, esto implementa la ecuación vectorial:

pero despejada para obtener a partir de y (las aceleraciones angulares). El término de acoplamiento se calcula con cross, y se trabaja en forma matricial con el tensor de inercia.

Luego se aplican saturaciones por componente para asegurar que los torques no excedan los límites que el hardware puede generar:

*tau = [ saturar(tau\_unsat(1), -params.tau\_lim(1), params.tau\_lim(1))*

*saturar(tau\_unsat(2), -params.tau\_lim(2), params.tau\_lim(2))*

*saturar(tau\_unsat(3), -params.tau\_lim(3), params.tau\_lim(3)) ];*

La función saturar es la forma compacta de aplicar un min(max(valor, min), max) a cada componente. Desde la perspectiva numérica, este paso es fundamental porque limita las entradas del modelo antes de la integración RK4, haciendo que el comportamiento simulado sea consistente con las restricciones físicas del actuador.

### Empuje total a partir del throttle

Por último, el controlador calcula el empuje total a partir del valor de refs.throttle. Para ello se parte del empuje necesario para hover:

*U\_hover = phys.m\*phys.g;*

*K\_U1 = 0.1;*

*U1 = U\_hover + K\_U1 \* refs.throttle\*phys.m\*phys.g;*

*U1 = saturar(U1, params.U1\_min, params.U1\_max);*

Donde *U\_hover = mg* es el empuje que compensa exactamente el peso. Luego refs.throttle se interpreta como un valor adimensional en el rango que escala una fracción del peso (mediante *K\_U1*), lo que en la práctica permite simular pequeñas trepadas o descensos con el stick.

Nuevamente, se aplica una saturación para que permanezca dentro de un rango razonable *(U1\_min, U1\_max*).

El vector de entrada física final que sale de este bloque y se usa en el modelo dinámico es:

*U = [U1; tau];*

### Evaluación del modelo dinámico y método de integración RK4

En la sección de modelado matemático/mecánico se obtuvieron las ecuaciones diferenciales que describen la dinámica del dron (traslación y rotación del cuerpo rígido). En la simulación, esas ecuaciones se implementan en dos bloques bien separados:

1. Una función que evalúa las derivadas en un estado dado:

2. Un método numérico que integra esas derivadas para avanzar el estado en el tiempo:

De ese modo, toda la teoría mecánica queda encapsulada en la función, y el integrador numérico se limita a evaluarla varias veces por paso y combinar esos resultados de forma estable y precisa.

### Función dron\_derivadas\_fisicas: calcular X\_punto a partir de X y U

La función dron\_derivadas\_fisicas(X, U, phys) implementa directamente el sistema de ecuaciones diferenciales que se dedujo en la parte teórica. Su objetivo es, dado un estado actual y unas entradas físicas, devolver el vector de derivadas

Primero se “desempaqueta” el vector de estados:

*x = X(1); y = X(2); z = X(3);*

*vx= X(4); vy = X(5); vz = X(6);*

*roll = X(7); pitch = X(8); yaw = X(9);*

*p = X(10); q = X(11); r = X(12);*

y se extraen los parámetros físicos de la estructura phys *m = phys.m; g = phys.g; I = phys.I;*

Las entradas físicas son:

* Empuje total en el eje del cuerpo *U1 = U (1);*
* Torques alrededor de cada eje del cuerpo *tau = U(2:4);*

a) Cinética de traslación y arrastre

Para pasar el empuje del sistema del cuerpo al sistema inercial, se utiliza la matriz de rotación en la función *matriz\_rotacion\_cuerpo\_al\_mundo(roll,pitch,yaw)*, que implementa la rotación ZYX (yaw–pitch–roll):

*R = matriz\_rotacion\_cuerpo\_al\_mundo(roll, pitch, yaw);*

*F\_mundo = R\*[0;0;U1];*

*Fgrav   = [0;0;-m\*g];*

Sobre esa fuerza se añade un término de arrastre (Fdrag), que se modela de forma sencilla con un aporte lineal y uno cuadrático por componente de velocidad:

*kL = 0.5;   % parte lineal*

*kQ = 0.5;   % parte cuadrática*

*drag\_comp = @(vi) (abs(vi)>eps\_v) .\* (kL\*vi + kQ\*abs(vi).\*vi);*

*Fdrag = -[ drag\_comp(vx);*

*drag\_comp(vy);*

*drag\_comp(vz) ];*

Numéricamente, este término introduce amortiguamiento y ayuda a estabilizar la simulación ante velocidades altas, sin necesidad de un modelo aerodinámico detallado.

La aceleración lineal del centro de masa en el sistema inercial se obtiene como:

*acc = (F\_mundo + Fgrav + Fdrag) / m;*

y esto corresponde a las derivadas de las velocidades:

b) Cinética de rotación

Para la parte rotacional se utilizan las ecuaciones del cuerpo rígido alrededor del centro de masa. Se define el vector de velocidades angulares y se calcula:

*w = [p; q; r];*

*hvec = I \* w;*

*w\_punto = I \ (tau - cross(w, hvec));*

Es decir, se implementa numéricamente:

que son exactamente las ecuaciones que ya se dedujeron en la parte teórica, pero en forma matricial.

c) Relación entre (p,q,r) y las derivadas de los ángulos de Euler

El modelo está expresado en ángulos de Euler (roll, pitch, yaw). Para obtener sus derivadas a partir de se utiliza la matriz, calculada en la función *T\_zyx*:

*T = T\_zyx(roll, pitch);*

*Eta = T \* w;   % Eta = [phi\_dot; theta\_dot; psi\_dot]*

Dentro de T\_zyx se introduce un pequeño truco numérico para evitar divisiones por cos(θ) muy cercano a cero (singularidad de gimbal lock):

*cth = cos(pitch);*

*eps = 1e-8;*

*if cth < eps, cth = eps; end*

De este modo se mejora la robustez de la simulación cuando el dron se inclina mucho en pitch.

d) Empaquetado final de las derivadas

Por último, *dron\_derivadas\_fisicas* arma el vector con las derivadas de todas las variables:

*dx = zeros(12,1);*

*dx(1:3)   = [vx; vy; vz];   % ṙ = v*

*dx(4:6)   = acc;            % v̇ = a*

*dx(7:9)   = Eta;            % [phi\_dot; theta\_dot; psi\_dot]*

*dx(10:12) = w\_punto;        % ẇ*

Ésta es precisamente la función que necesita el método RK4 para avanzar la simulación.

### Método de integración rk4\_paso: avanzar el estado en el tiempo

Una vez que se dispone de la función dron\_derivadas\_fisicas para evaluar en un punto dado, la integración numérica se realiza con el método de Runge–Kutta de cuarto orden (RK4), implementado en la función:

*function X\_next = rk4\_paso(X, U, phys, h)*

La lógica es la siguiente: dado el estado actual en el tiempo, y suponiendo que la entrada se mantiene constante durante el intervalo (aproximación de “retención de orden cero”, coherente con un controlador digital), se calculan cuatro pendientes:

*k1 = dron\_derivadas\_fisicas(X,             U, phys);*

*k2 = dron\_derivadas\_fisicas(X + 0.5\*h\*k1,  U, phys);*

*k3 = dron\_derivadas\_fisicas(X + 0.5\*h\*k2,  U, phys);*

*k4 = dron\_derivadas\_fisicas(X + h\*k3,      U, phys);*

*X\_next = X + (h/6)\*(k1 + 2\*k2 + 2\*k3 + k4);*

Donde

* K1: derivada al inicio del intervalo (como en Euler explícito).
* K2: derivada en el punto medio estimado usando K1.
* K3: derivada en otro punto medio, ahora usando K2.
* K4: derivada al final del intervalo, extrapolando con K3.

El nuevo estado se estima como:

lo que da un método de orden 4, mucho más preciso que un Euler simple para el mismo paso de tiempo. En el programa principal se utiliza:

*X = rk4\_paso(X, U\_eff\_from\_forces(f\_eff, Kt, Kq, l), phys, h);*

*X\_hist(:,k) = X;*

Es decir, se usa un vector de entradas efectivas *U\_eff\_from\_forces(...)*, que ya incluye el efecto de la saturación de motores, el mapeo PWM–velocidad y el límite de corriente que será explicado más abajo. Con esas entradas se calcula mediante dron\_derivadas\_fisicas. RK4 integra esas derivadas y entrega el nuevo estado X\_next, que se guarda en el historial para análisis posterior.

Como chequeo adicional, al inicio de la simulación se verifica una sola vez que “dron\_derivadas\_fisicas” devuelva un vector de tamaño correcto (12×1). Esto ayuda a detectar errores de programación antes de ejecutar simulaciones largas:

*if k == 2*

*dx\_test = dron\_derivadas\_fisicas(X\_hist(:,1), U, phys);*

*if ~isequal(size(dx\_test), [12,1])*

*error('dx no es 12x1');*

*end*

*end*

*end*

## Explicación mixer (gradiente conjugado)

En la simulación del cuadricóptero, el mixer resuelve la asignación de velocidades angulares para los cuatro motores a partir de las consignas de control. Esto se modela como un sistema lineal del tipo:

donde:

* es la matriz de mezcla, definida por la geometría del dron y el sentido de giro de cada hélice;
* es el vector de velocidades de motor desconocido;
* es el vector objetivo que contiene el empuje total requerido y los torques en roll, pitch y yaw.

El sistema debe resolverse en cada iteración de la simulación para reflejar los cambios de actitud producidos por las entradas del piloto. Sin embargo, la matriz del mixer presenta acoplamientos entre ejes, y un cálculo directo mediante inversión matricial resulta ineficiente y numéricamente inestable. Por este motivo se implementó el método de Gradiente Conjugado (CG), ideal para matrices simétricas y definidas positivas como las que aparecen en este problema.

La función utilizada, Gradiente\_conjugado(A,b,x0,kmax,use\_trunc6), realiza la iteración clásica del método CG:

La función detiene la iteración cuando:

Como se ve en la imagen:



Figura 40: Bucle if

Lo que garantiza una solución suficientemente precisa para la simulación dinámica sin incrementar el costo computacional.

Ahora vemos cómo se integra el método al mixer del drone, en donde en cada paso:

1. Se arma el vector b con las consignas del piloto
2. Se calcula la matriz física A del cuadricoptero
3. Se ejecuta la función Gradiente\_conjugado para obtener
4. Los valores resultantes se limitan a la gama de operación de los motores
5. El modelo dinámico usa estas velocidades para actualizar la actitud del drone.

## Seguridad de corriente (Newton-Raphson)

Durante el diseño eléctrico del dron surgieron limitaciones asociadas a la capacidad de corriente de las pistas, la protección de los sensores de corriente y la determinación del umbral máximo de operación seguro para los elementos conectados al ADC y a la etapa de potencia. Las expresiones que gobiernan esas condiciones no resultan lineales, por lo cual no pueden resolverse mediante despejes algebraicos simples.

En el código, entre las líneas 145–185 y 647–711, aparecen distintos cálculos donde la variable de interés (corriente máxima admisible, caída de tensión, resistencia equivalente, o potencia disipada) se relaciona con la geometría del conductor y parámetros eléctricos mediante ecuaciones del tipo:

f(I)=R(I)⋅I2−Pmax⁡=0f(I) = R(I) \cdot I^2 - P\_{\max} = 0f(I)=R(I)⋅I2−Pmax​=0

O bien:

f(V)=V−I(V)⋅R=0f(V) = V - I(V)\cdot R = 0f(V)=V−I(V)⋅R=0

Estas expresiones contienen dependencias no lineales, ya sea porque la resistencia depende del ancho y temperatura de la pista, o porque la corriente afecta simultáneamente la caída de tensión y la disipación térmica.

Para resolver estas ecuaciones se implementó el método de Newton–Raphson, un procedimiento iterativo que aproxima la raíz de una función mediante:

xn+1=xn−f(xn)f′(xn)x\_{n+1} = x\_n - \frac{f(x\_n)}{f'(x\_n)}xn+1​=xn​−f′(xn​)f(xn​)​

Este método fue seleccionado por varias razones:

1. Convergencia rápida: su orden de convergencia es cuadrático siempre que la derivada no sea nula, permitiendo obtener valores precisos en pocas iteraciones.
2. Adecuado para funciones derivables, tal como las utilizadas en cálculos de potencia, caída de tensión y densidad de corriente.
3. Evita resolver expresiones algebraicas complejas, lo cual simplifica la implementación en código y reduce probabilidades de error.
4. Permite evaluar condiciones de seguridad bajo diferentes configuraciones simplemente modificando valores iniciales o parámetros del conductor.

En el contexto del dron, Newton–Raphson se utilizó para:

* determinar la corriente máxima segura en función de la resistencia de pista y potencia máxima admisible;
* verificar la capacidad del ADC ante picos o sobre corrientes;
* calcular la disipación térmica bajo distintas condiciones de carga;
* resolver condiciones de diseño en las cuales los sensores o la placa de potencia quedaban cerca del límite operativo.

En cada caso, Newton–Raphson permitió obtener soluciones numéricas confiables, evitando sobreestimaciones peligrosas y garantizando que la simulación y el hardware final respetaran márgenes de seguridad adecuados.

## Cálculo de corriente y potencia total (Simpson ⅓)

Integral(I\_total dt) -> [A\*s]; convertir a mAh con /3600\*1000: En la línea 228 se calcula la corriente total demandada por el dron mediante el algoritmo cumulative\_simpson13(y, h).

Energía: Integral(P dt) -> [W\*s] => Wh con /3600: En la línea 233 se calcula la energía consumida por el cuadricoptero mediante cumulative\_simpson13(y, h).

El algoritmo empleado consiste de dos métodos el de Simpson ⅓ y Trapecio, dependiendo la cantidad de intervalos que posee. Si la cantidad de intervalos es par, se emplea el método de Simpson ⅓, en cambio, si se dispone de intervalos pares, se emplea trapecio.

Sentencia de Simpson ⅓

Si se analiza, el avance de Simpson ⅓ se realiza de a dos intervalos, de k-2 a k, y su aproximación es cuadrática para cada intervalo, lo que hace la sentencia adjuntada es sumar a la integral anterior la actual , es por ello que es acumulativo (cumulative).

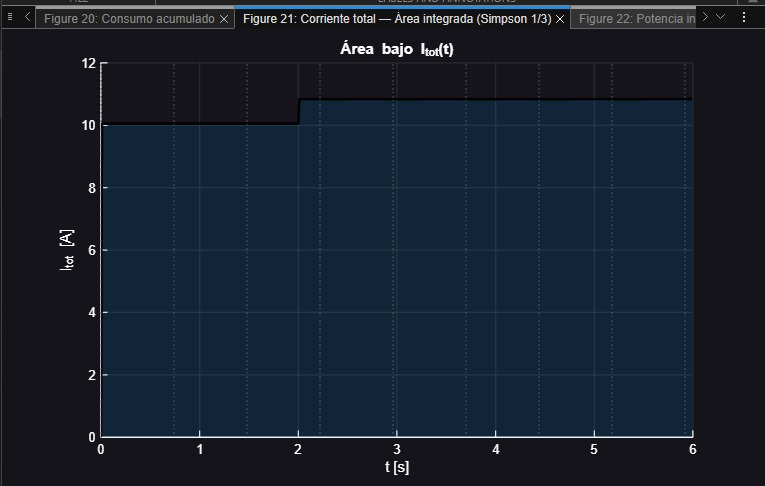


Figura 41: Área integrada de corriente con simpson 1/3

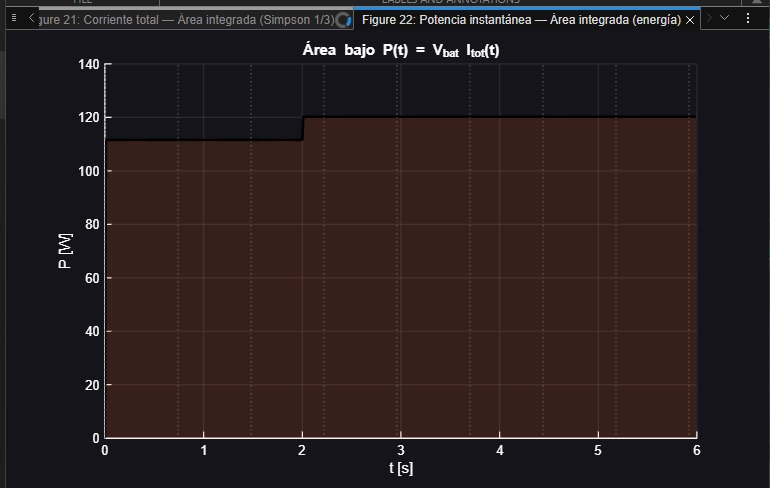


Figura 42: Área integrada de energia con simpson 1/3

## Gráficas

## Ventana 3D (Spline)

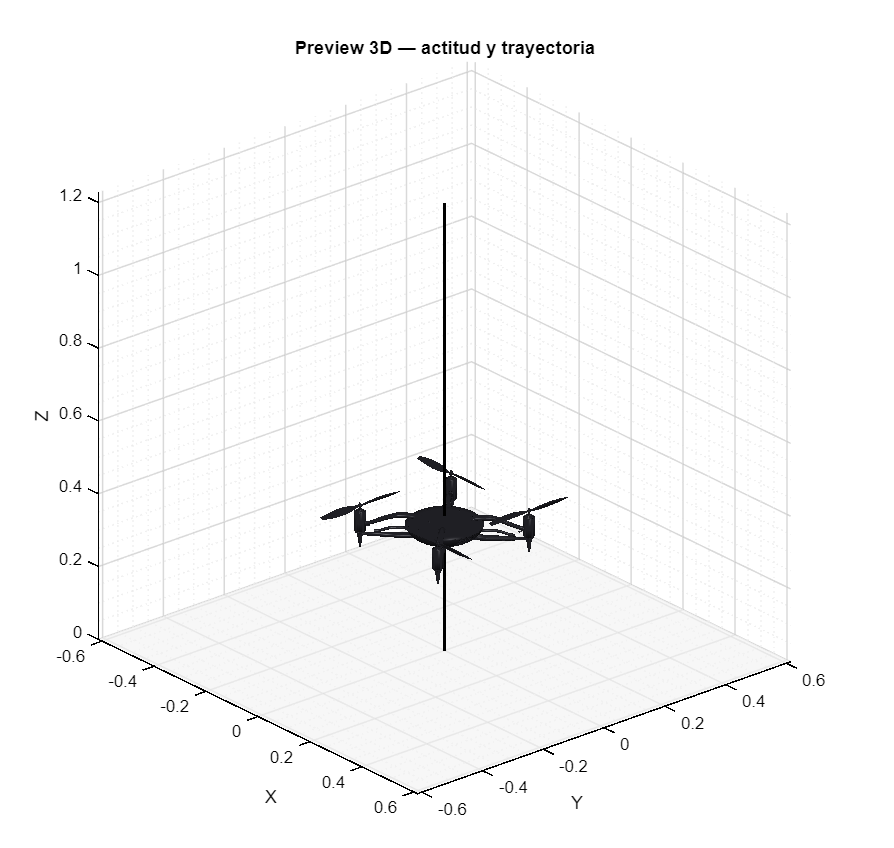


Figura 43: Visualización 3D

El spline suaviza la trayectoria a partir de una secuencia de puntos, su objetivo es producir señales suaves para una animación fluida.

En el apartado correspondiente a 1) Preparar muestras para animación (suave y acotada), se aumenta la cantidad de datos de tiempo, luego en la matriz MxN se introducen las M señales y las N cantidad de datos de cada una de las mismas. En el archivo interpolar\_spline\_natural\_multicanal.m se arma una matriz de ceros de tamaño M por K donde K corresponde a la cantidad de datos de tiempo de animación.

Se interpolan los datos originales con los x medidos y sus respectivos tiempos, y se arma una función en spl, luego se llena la matriz de ceros con los reemplazos de tiempos de animación en la función spl.

El código construir\_spline\_natural.m devuelve la función correspondiente a los datos de t\_sim y X\_sim. Pero antes se debe entender que es un spline cúbico natural.

Esta función es continua, y posee continuidad en su primera y segunda derivada, la condición de naturalidad es:

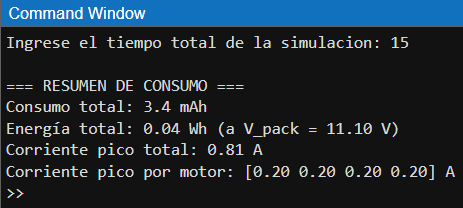
En el código se asegura el formato de fila, se calculan los intervalos, de modo que cada H(i) es la distancia entre nodos (H = diff(X)). El paso siguiente es encontrar cada uno de los coeficientes de los polinomios de tercer grado que tiene la siguiente forma:

Para continuar se debe armar la matriz tridiagonal para encontrar el coeficiente c. Esta matriz se estructura con el for de la fila 24 perteneciente al código construir\_spline\_natural. La diagonal principal se compone del doble de la suma del i-ésimo intervalo h y el anterior, la diagonal adyacente superior se compone de los intervalos i-ésimos, y la adyacente inferior de los intervalos i-ésimos -1.

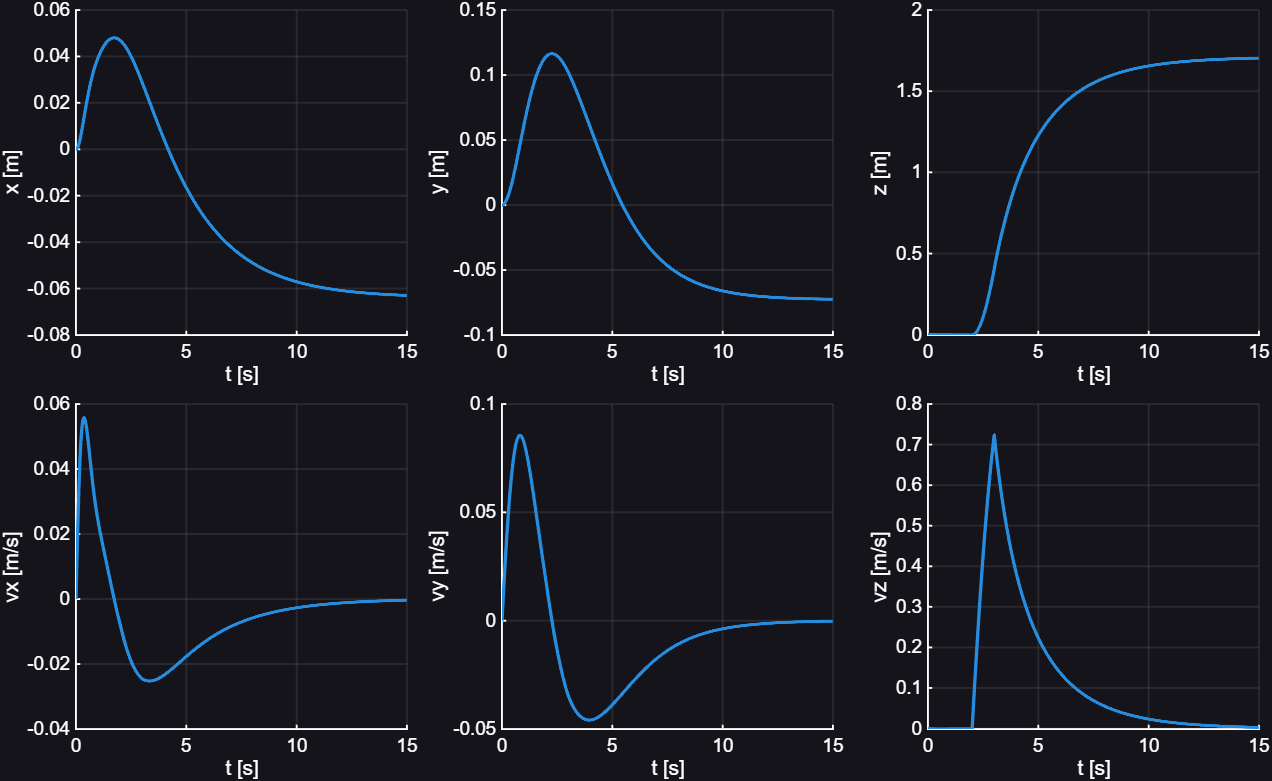
El vector Nx1 devolverá los coeficientes tal que son iguales a 0. Con los c intermedios se calculan todos los otros coeficientes b y d, mediante las siguientes ecuaciones:

## Resultados de la simulación

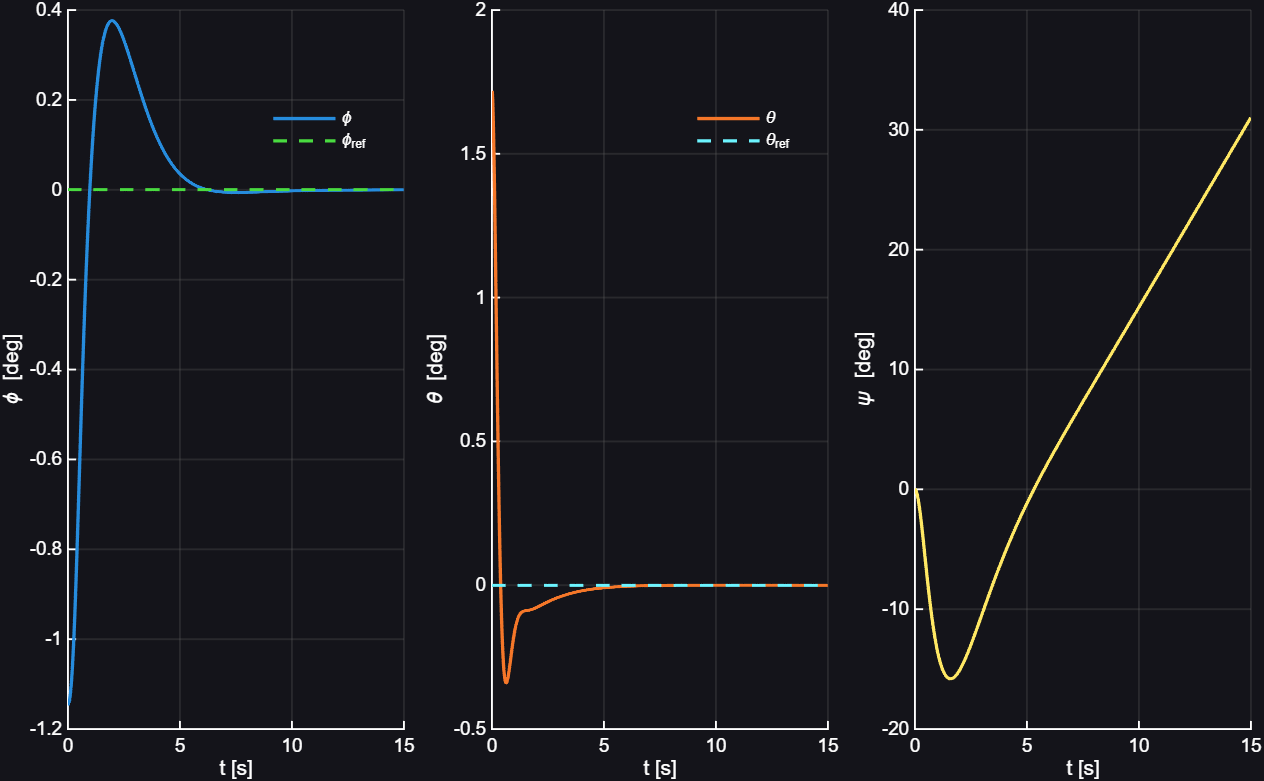
La simulación del cuadricóptero se lleva a cabo desde el código simulacion\_dron\_main.m. El programa solicita al usuario el tiempo de simulación, y mediante la incorporación de las funciones devuelve las siguientes graficas:

 Figura 44

Posiciones y velocidades:

 Figura 45

Actitud del cuerpo rigido:

 Figura 46

Velocidades angulares contra referencias:

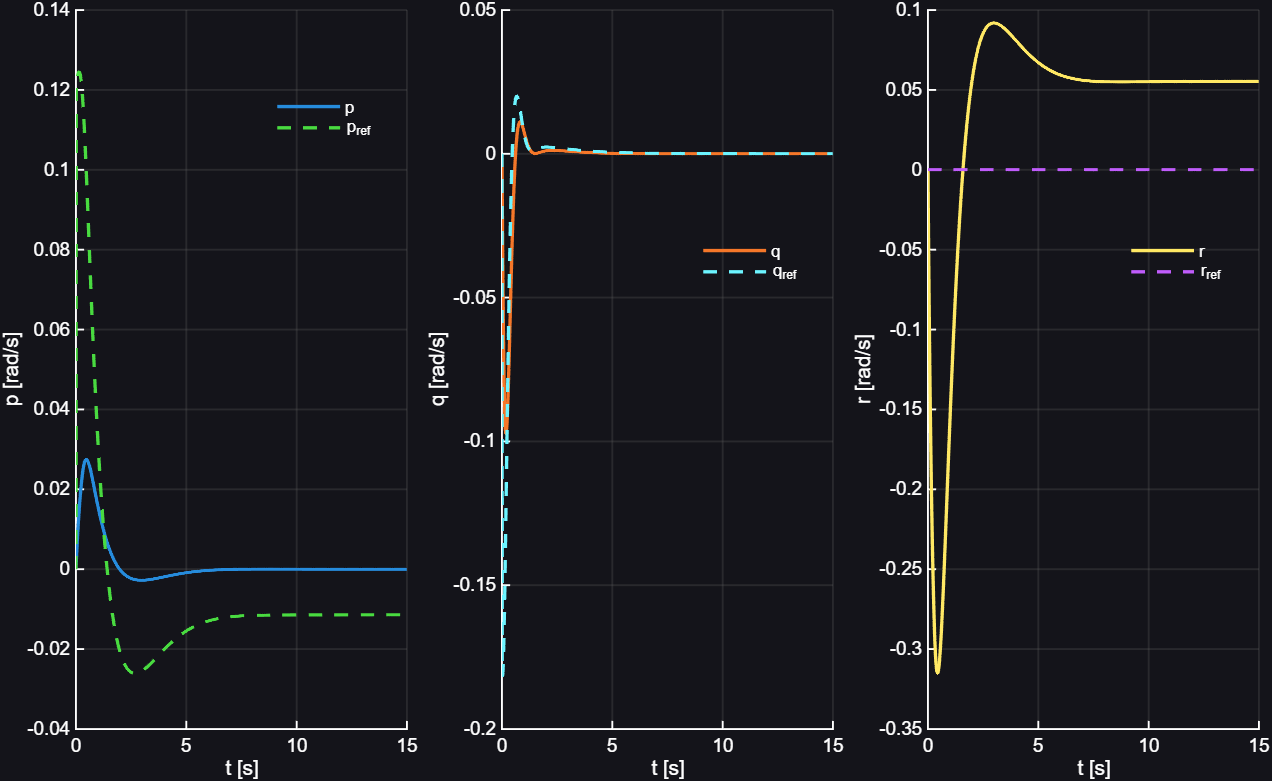
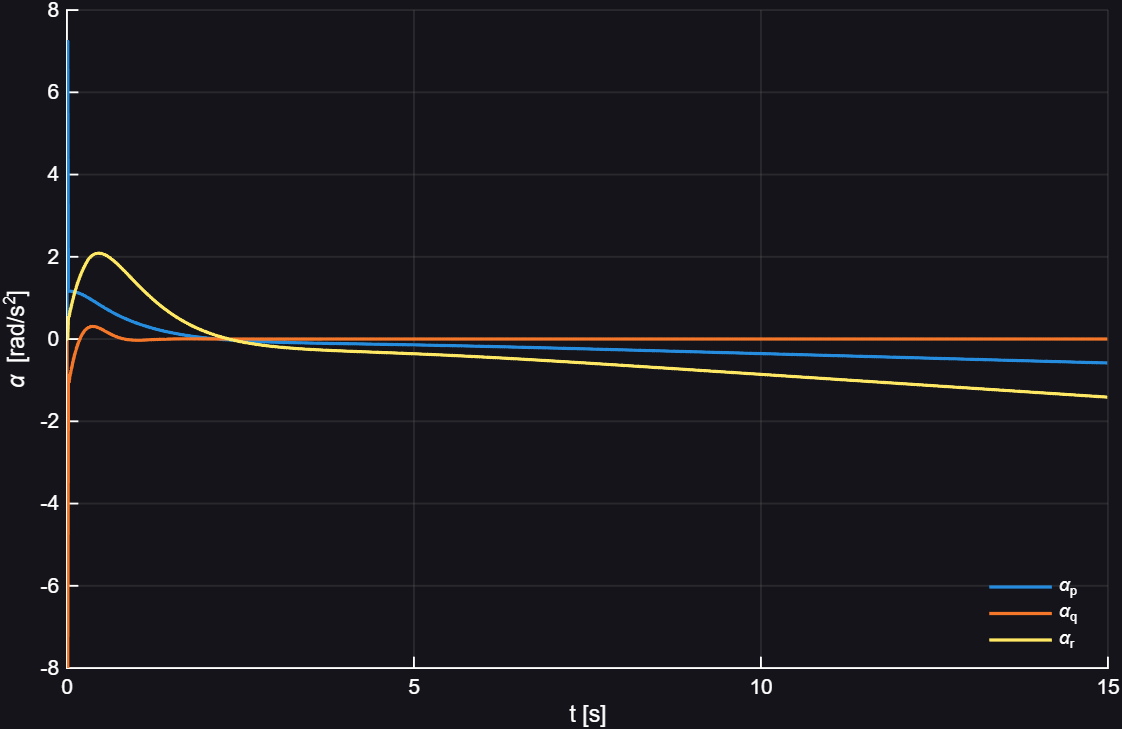
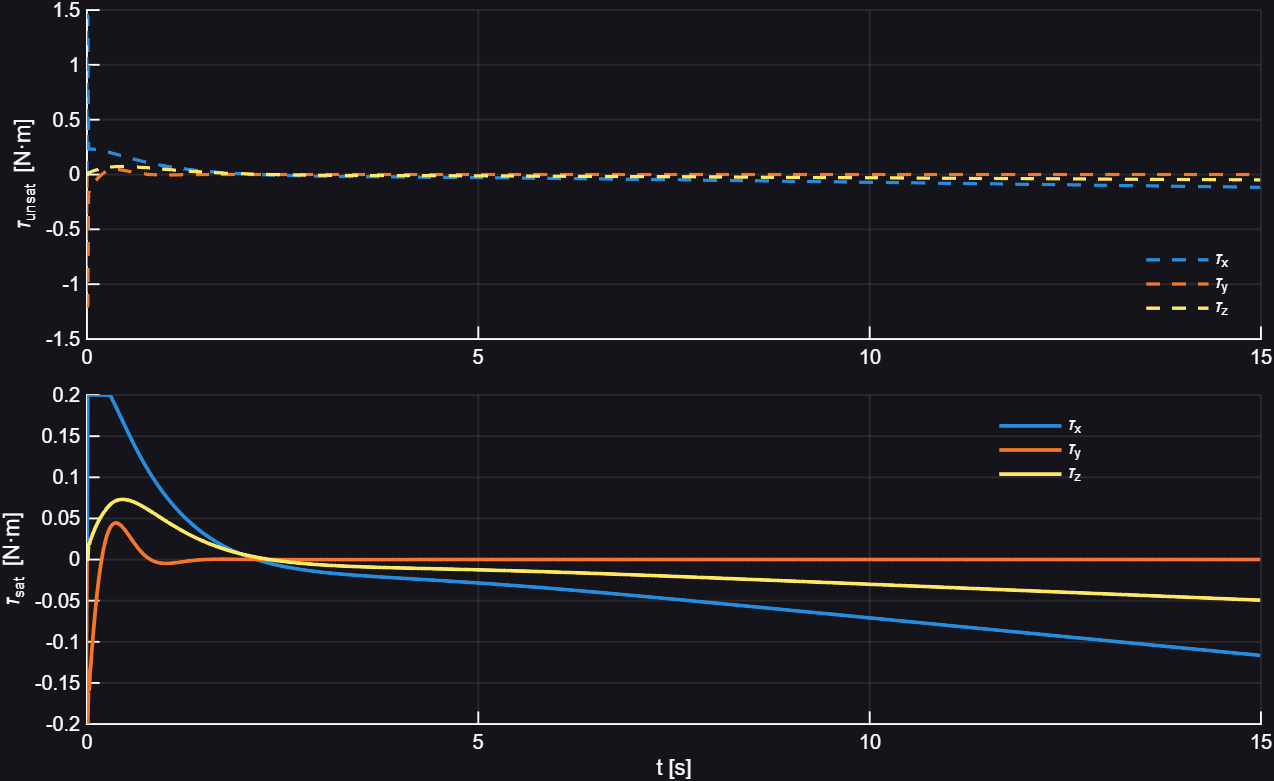


Figura 47

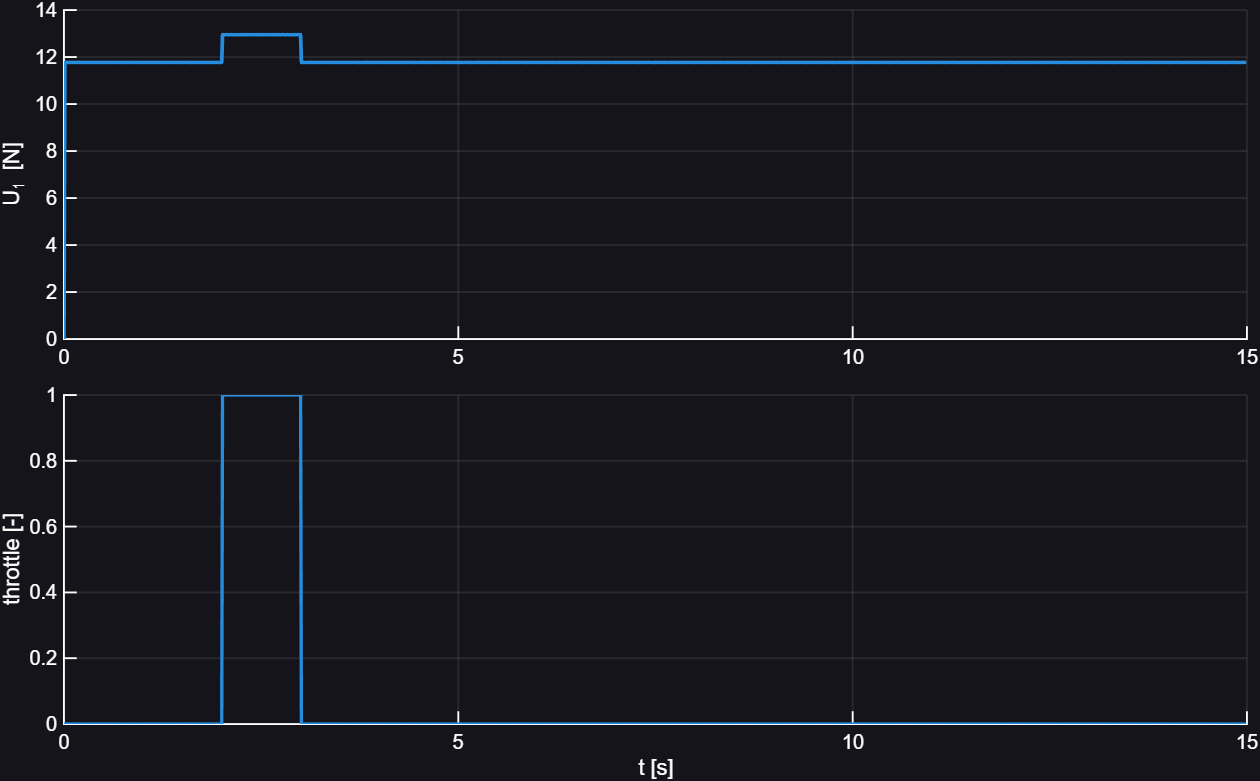
Aceleraciones angulares deseadas:

Figura 48

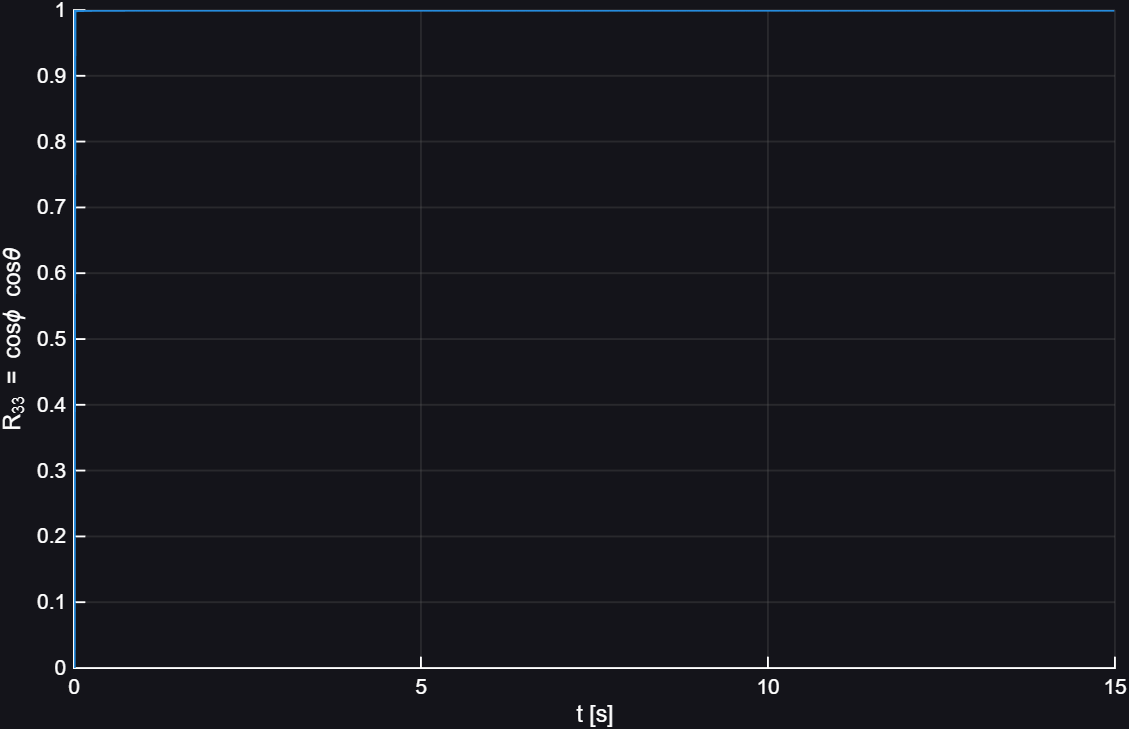
Torques:

 Figura 49

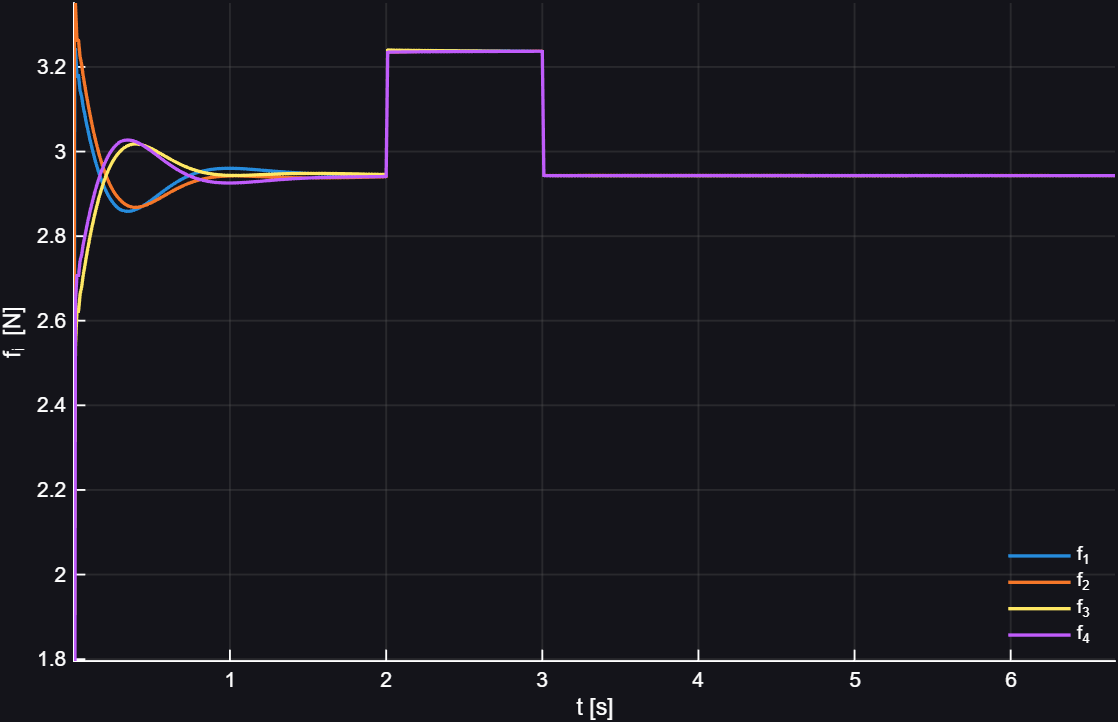
Empuje y throttle

 Figura 50

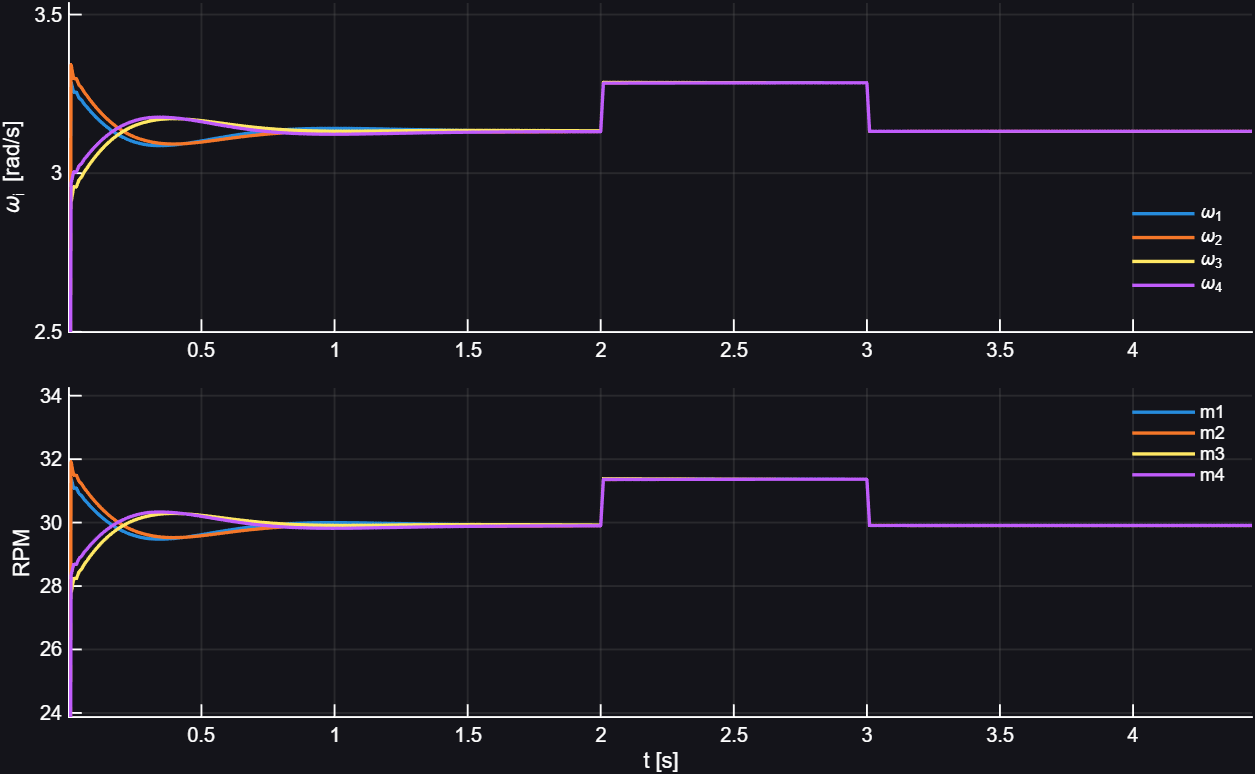
R33:

Figura 51

Fuerza por motor:

Figura 52

Velocidad de cada motor:

Figura 53

Aceleración angular de cada motor:

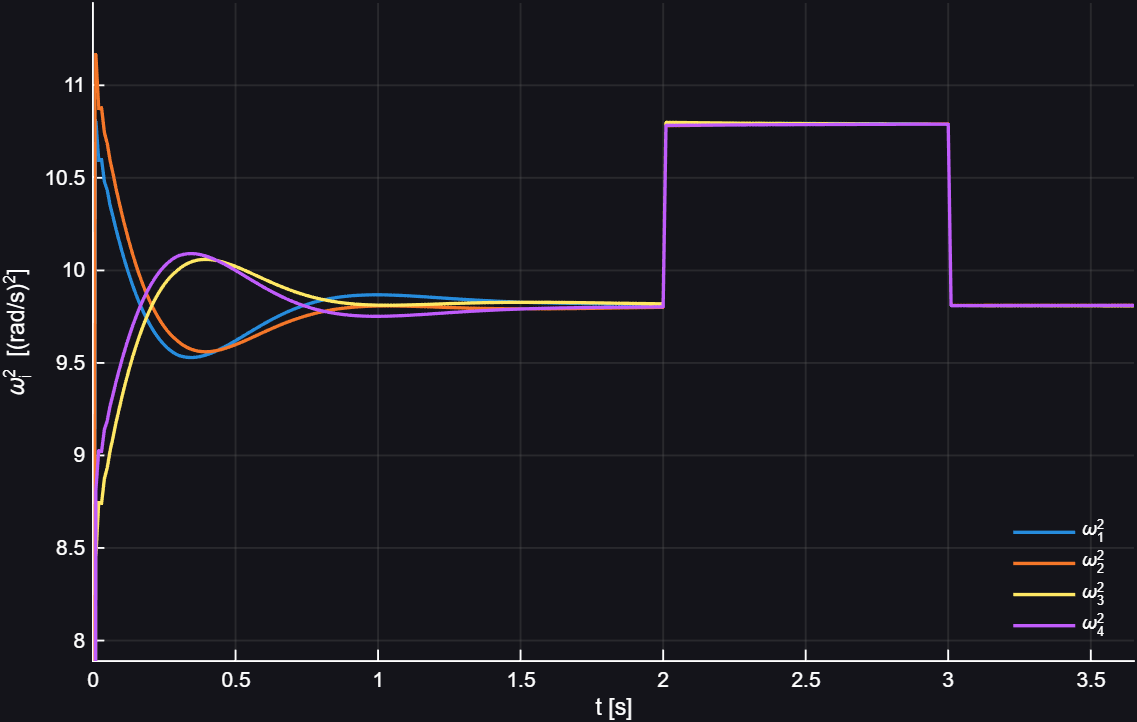
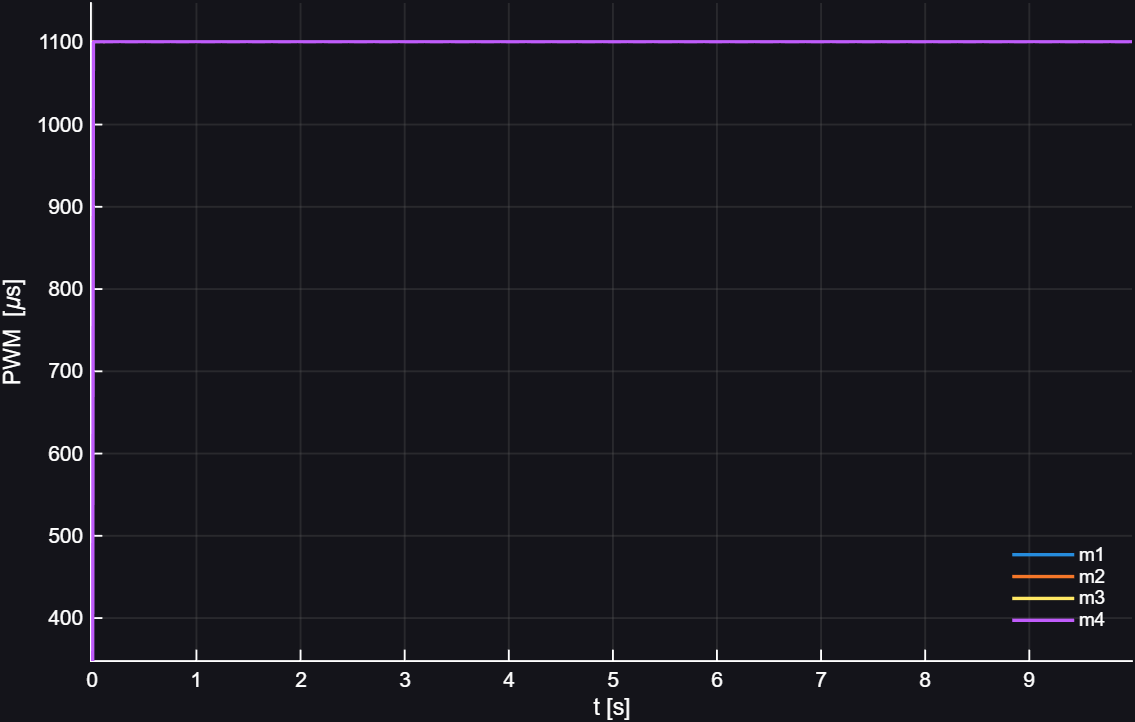
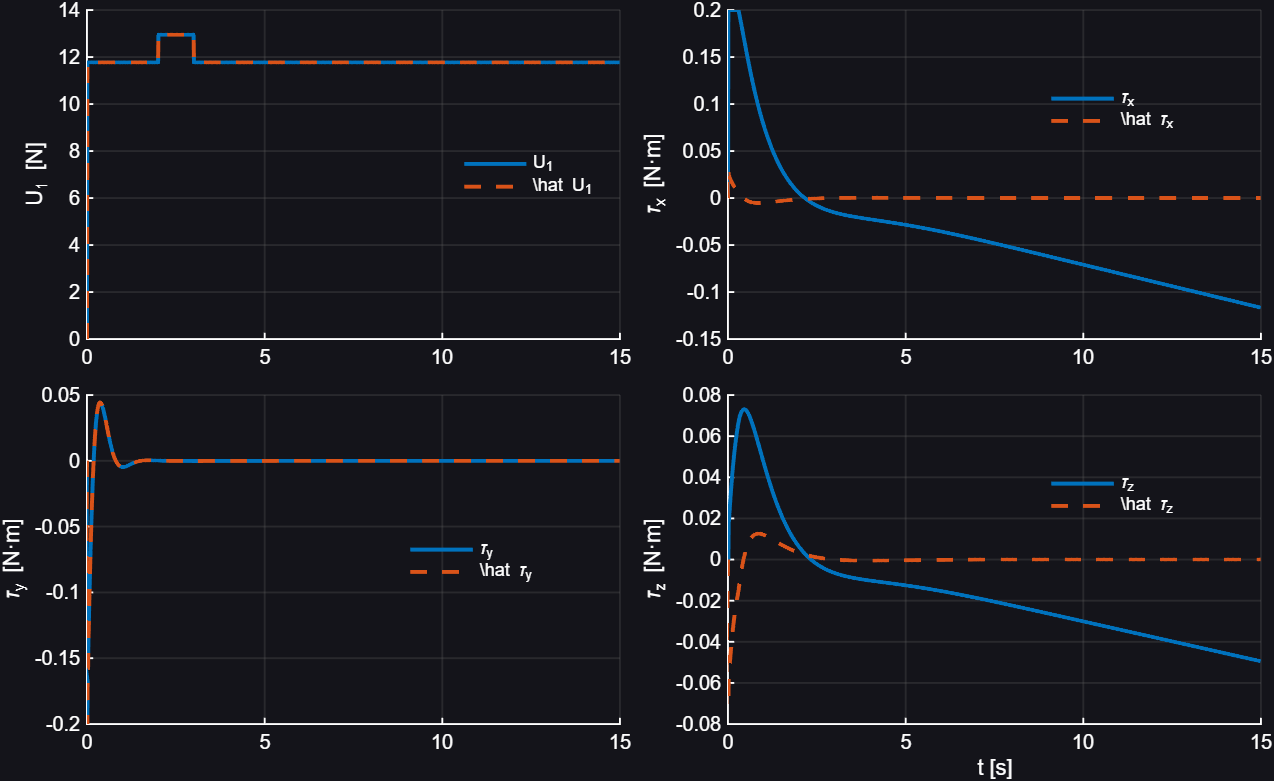


Figura 54

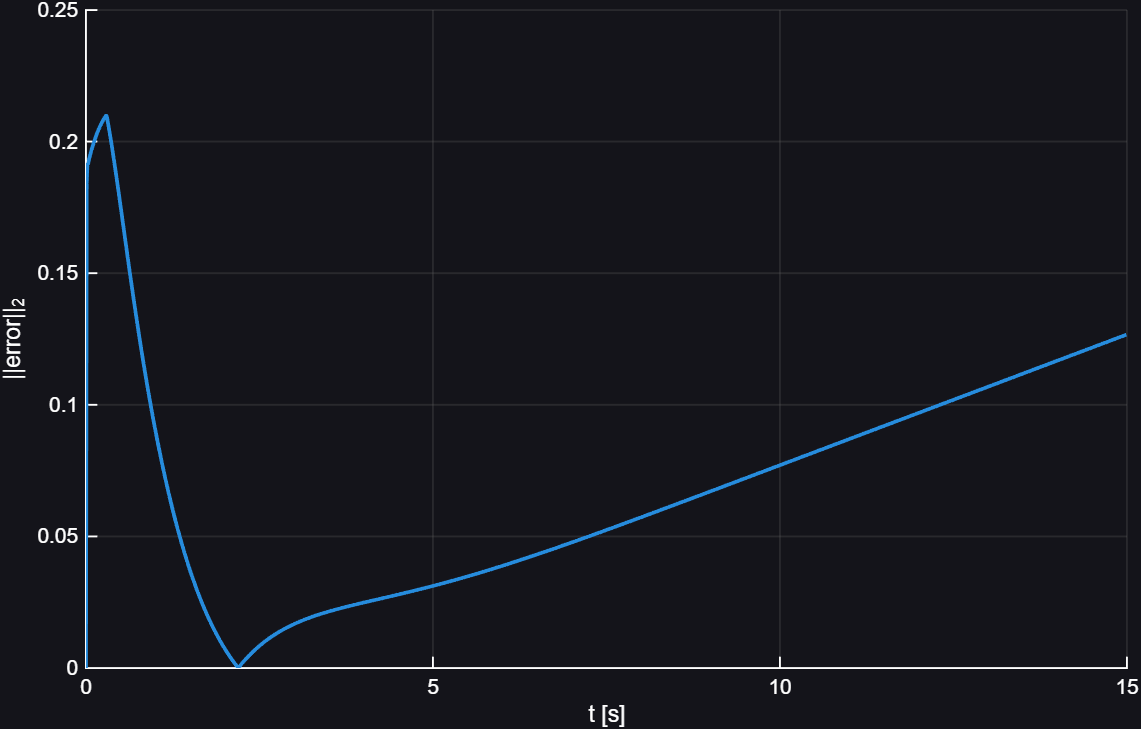
PWM por motor:

Figura 55

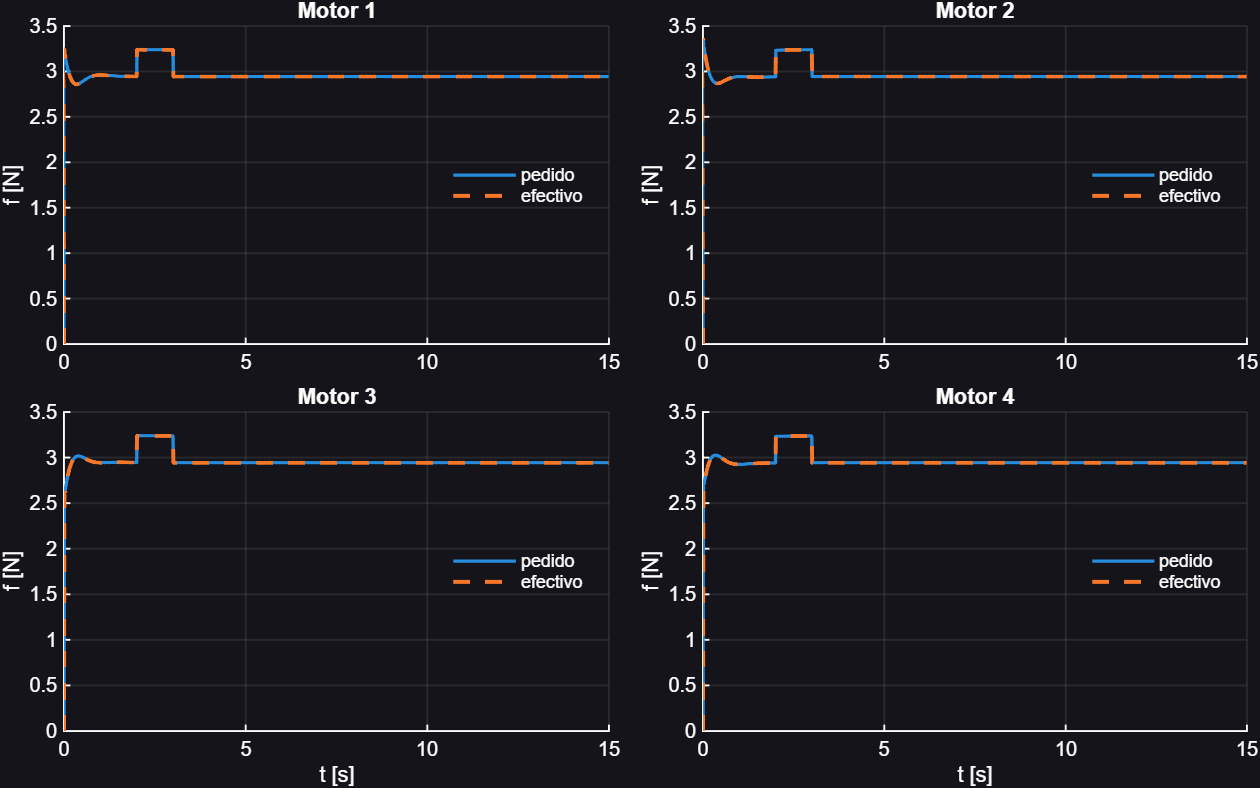
Empuje y torques en cada eje:

Figura 56

Error entre fuerza

Figura 57

PWM solicitado contra medido

Figura 58

Velocidad angular solicitada vs medida:

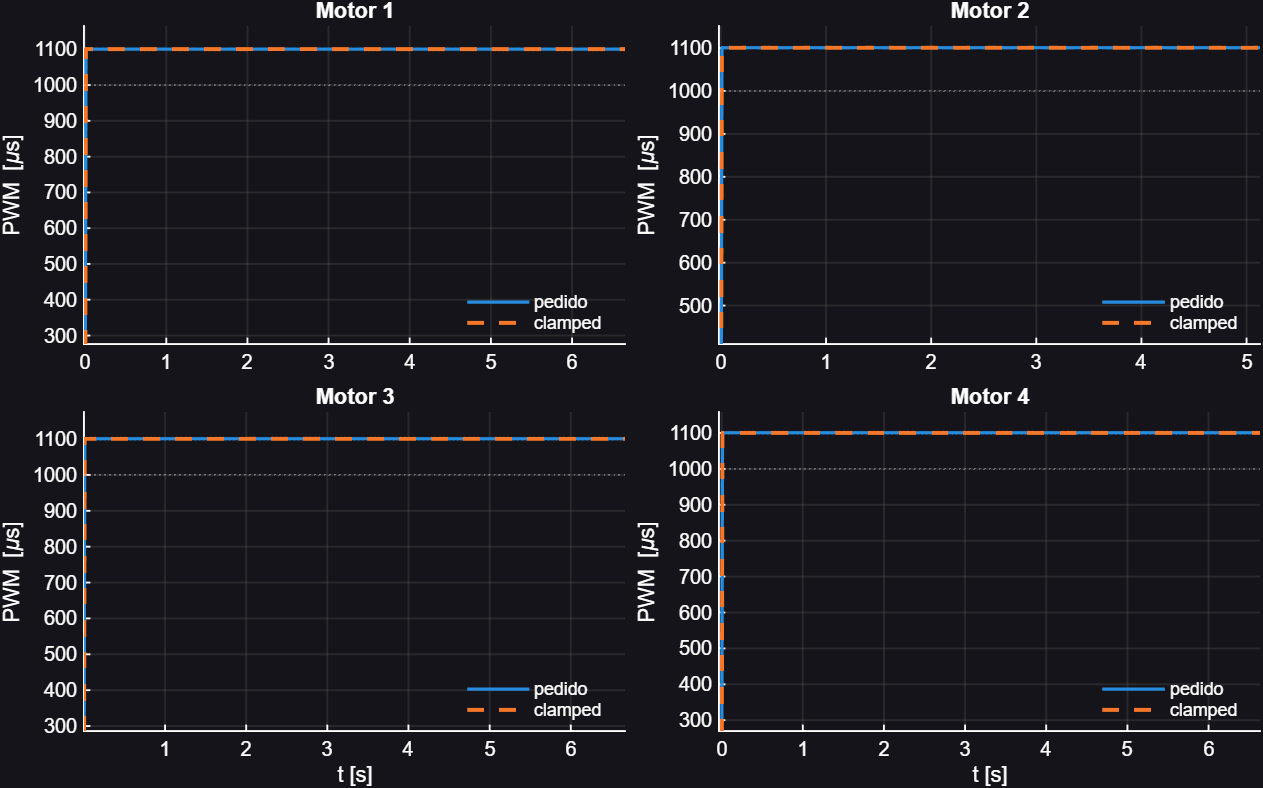


Figura 59

Fuerza solicitada vs medida:

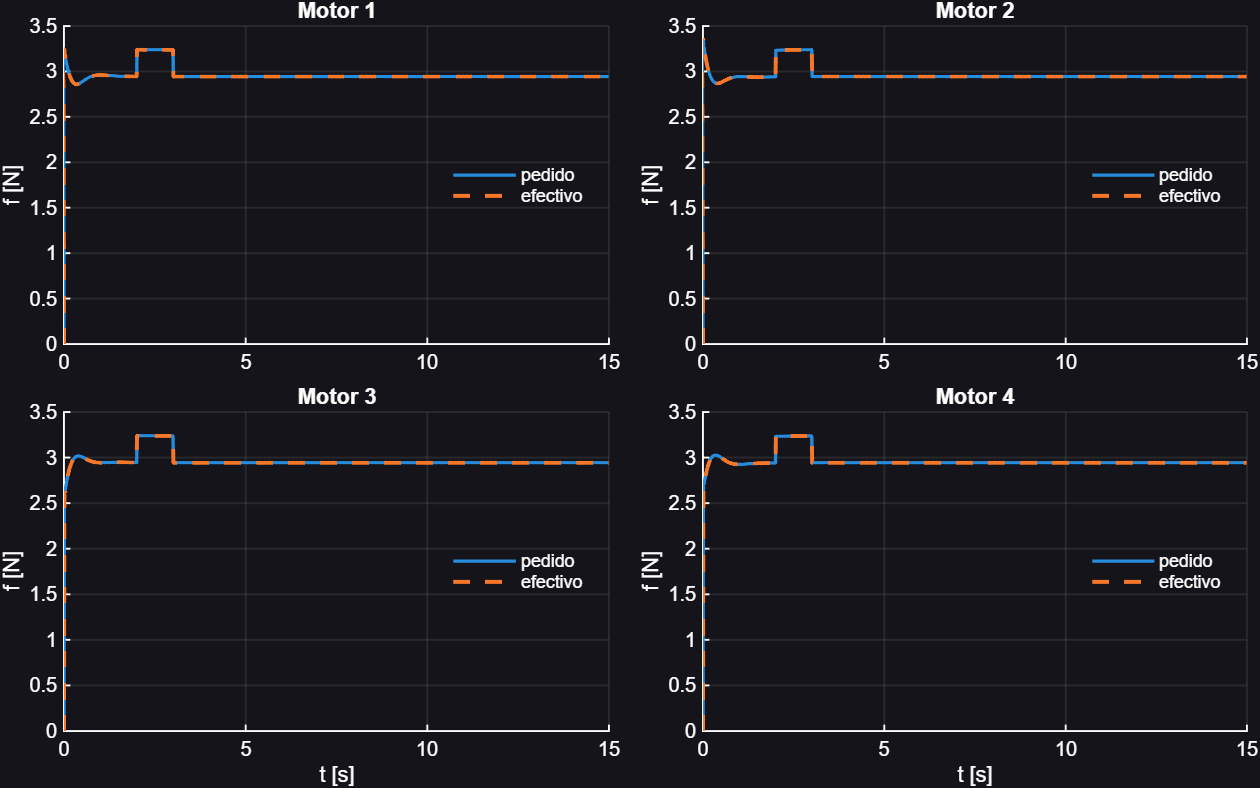


Figura 60

NR:

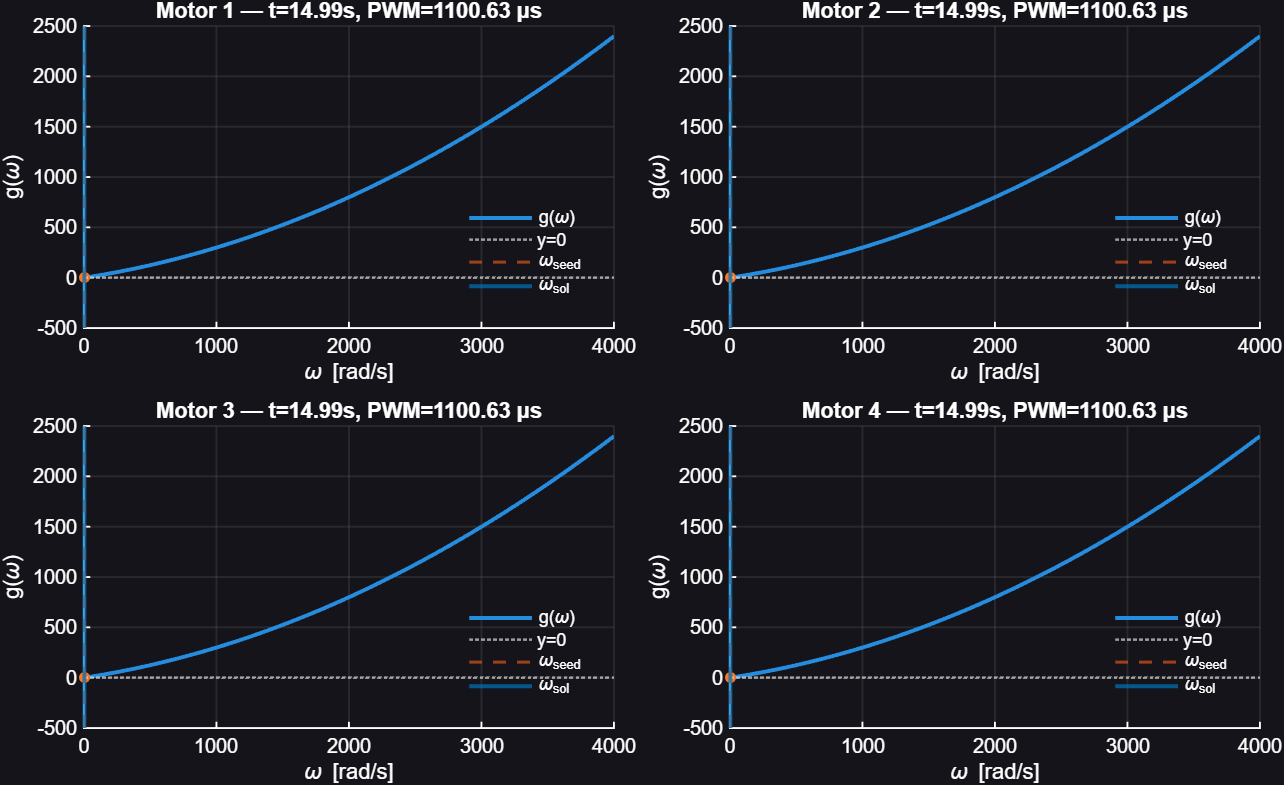


Figura 61

Corriente por motor:

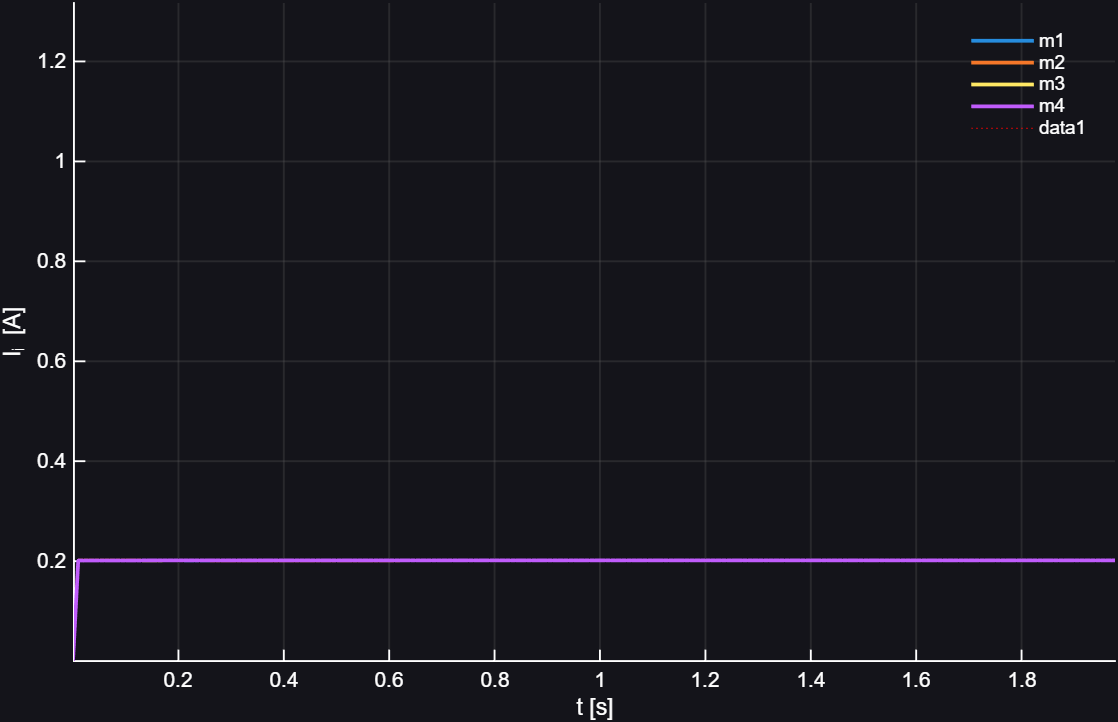
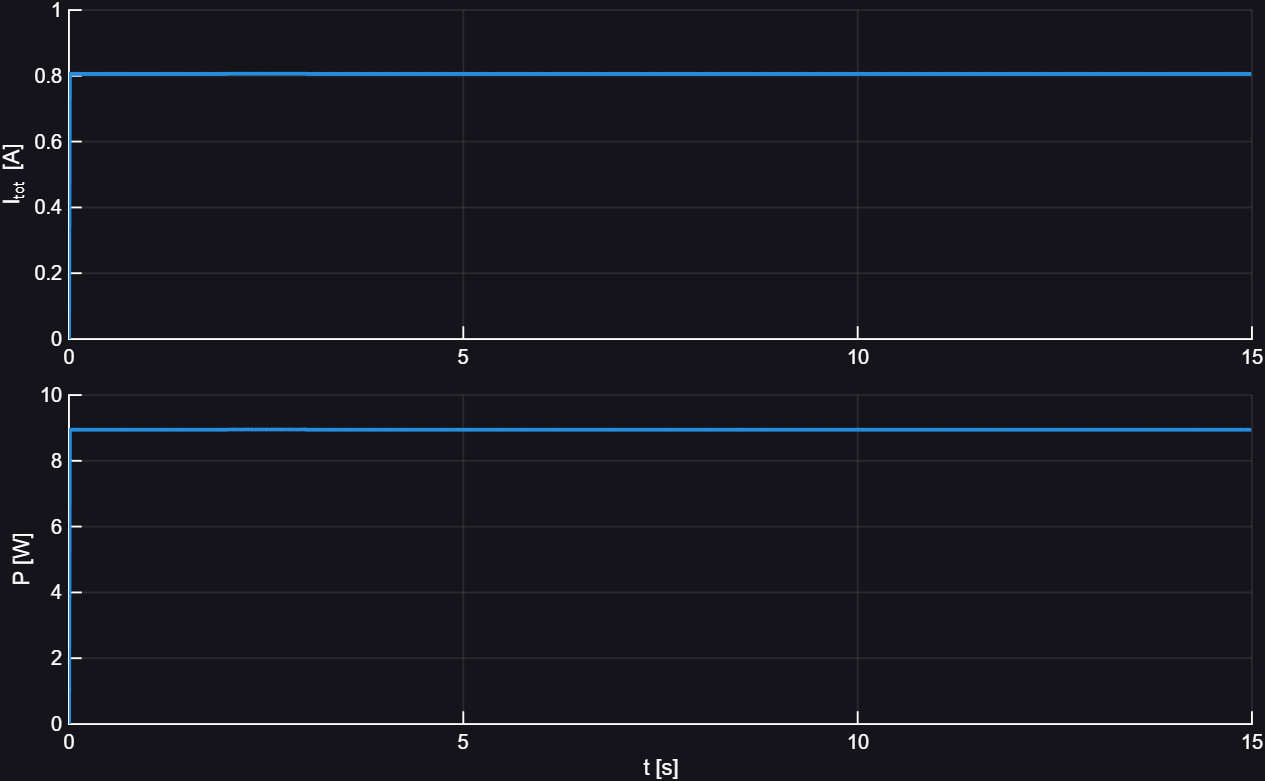


Figura 62

Corriente total y potencia:

Figura 63

Carga y energía demandada:

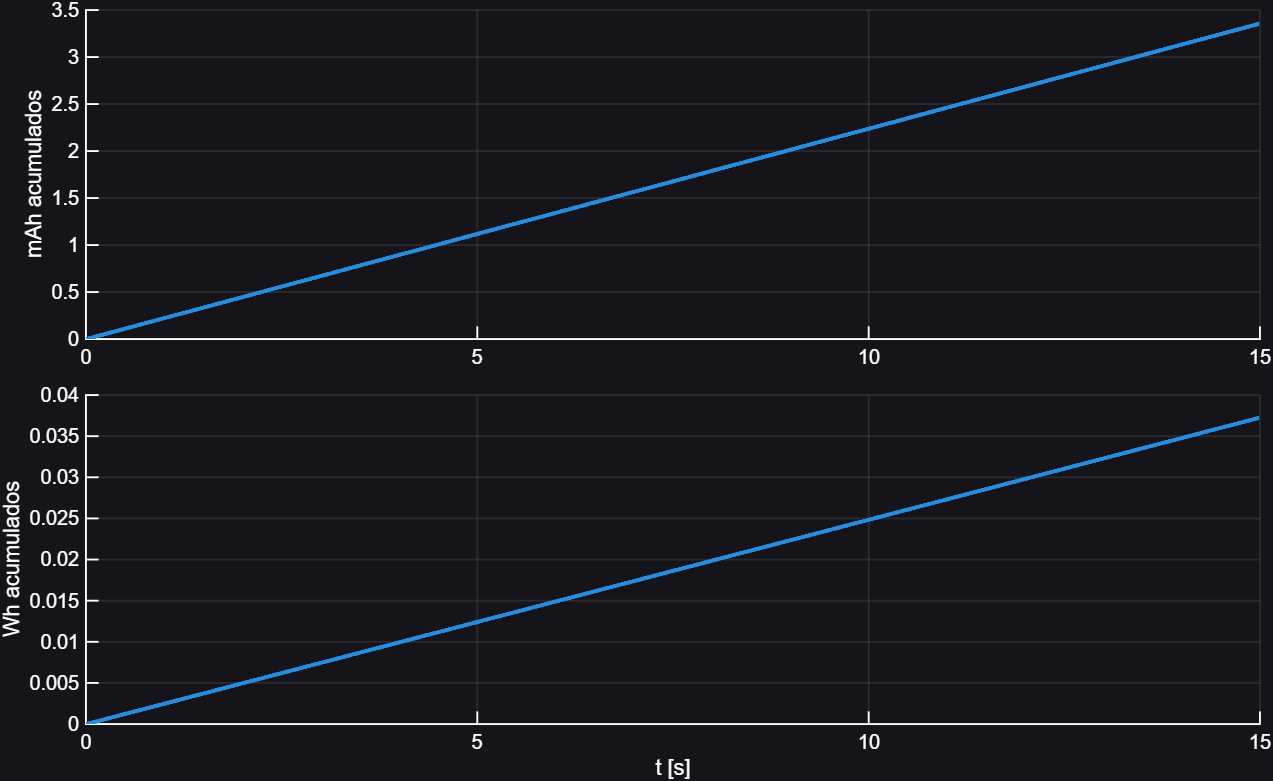


Figura 64

Corriente total:

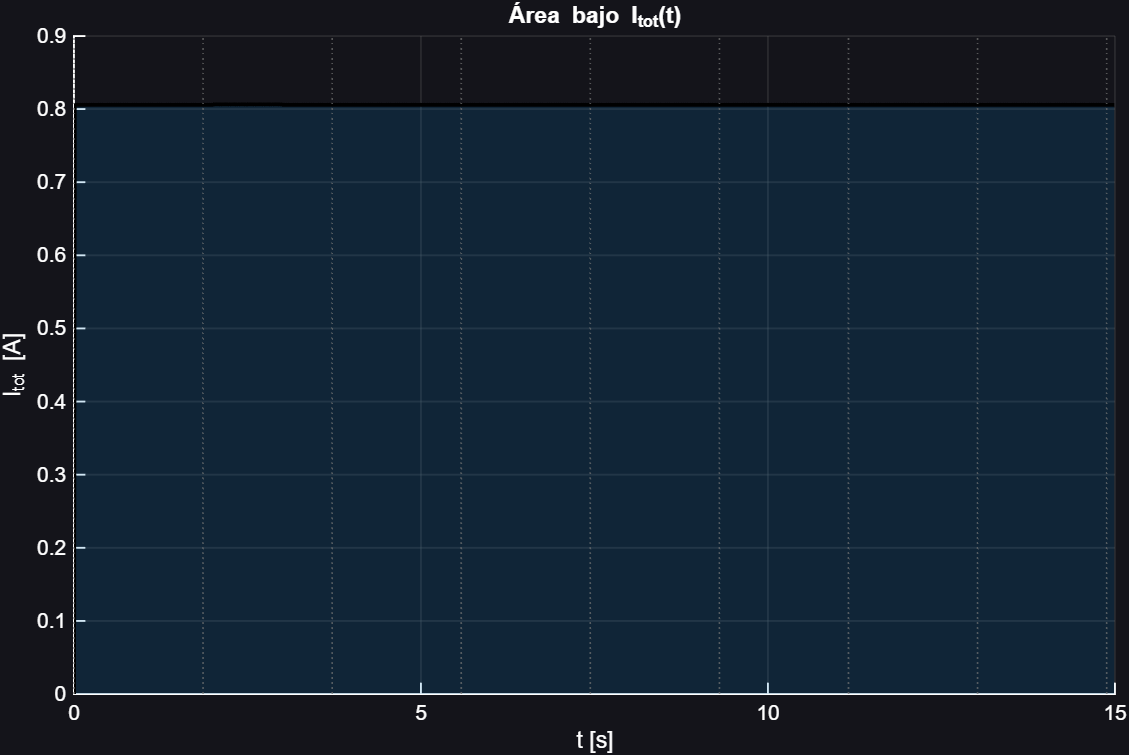


Figura 65

Potencia total:



Figura 66

Los resultados obtenidos en esta simulación corresponden a un tiempo de simulación de 15 segundos, y el siguiente ajuste para los ángulos en el arranque:

Roll:

Pitch:

Yaw:

# Algoritmo

## Control Remoto

El firmware desarrollado para el control remoto del dron se implementó sobre un microcontrolador ESP32 y cumple la función de digitalizar las señales analógicas de los joysticks, filtrarlas, normalizarlas y transmitirlas de forma inalámbrica mediante el protocolo ESP-NOW. Su diseño prioriza la estabilidad de lectura, la compensación automática de desvíos mecánicos en los potenciómetros y una latencia mínima en la comunicación con la controladora de vuelo.

### Arquitectura general del firmware

El programa se divide en cuatro bloques principales:

1. Lectura y procesamiento de señales analógicas
2. Autocalibración y filtros adaptativos de cada eje
3. Mapeo físico a referencias de vuelo (roll, pitch, yaw y throttle)
4. Envío inalámbrico mediante ESP-NOW

El firmware mantiene un ciclo de actualización constante de 100 Hz, garantizando que las referencias enviadas al dron posean baja latencia y variaciones suaves aun con movimiento brusco de los sticks.

### Modelo de cada eje (Axis) y autocalibración

Cada uno de los cuatro ejes del control remoto (LX, LY, RX, RY) se modela mediante una estructura:

* Centro (offset) del potenciómetro
* Filtro exponencial de señal cruda (EMA)
* Bias adaptativo (auto-trim)
* Medición de recorrido positivo y negativo (span\_pos / span\_neg)

Al iniciar el sistema se realiza una autocalibración automática: durante 400 ms se toman 120 lecturas y se calcula el valor de reposo de cada potenciómetro. Esto permite compensar diferencias mecánicas entre joysticks y eliminar la necesidad de calibración manual.

### Procesamiento de señal: filtrado, deadzone y auto-trim

El comportamiento del joystick en la práctica suele presentar deriva, ruido mecánico y variaciones en los extremos del recorrido. Por este motivo cada eje se procesa mediante la función processAxis, que aplica:

a) Filtro exponencial (EMA)

Se suavizan las lecturas crudas del ADC:

Con  
 α = 0.25, lo que filtra ruido manteniendo buena respuesta dinámica.

b) Auto-trim adaptativo

Si el stick está cerca de su posición central, el sistema ajusta lentamente un “bias” interno.  
 Esto corrige derivas propias del potenciómetro con el paso del tiempo.

c) Deadzone

Se anula cualquier valor pequeño alrededor del centro, evitando que vibraciones o imperfecciones mecánicas generen movimiento no deseado del dron.

d) Aprendizaje automático del recorrido

El firmware mide continuamente los valores máximos y mínimos alcanzados por cada eje, actualizando span\_pos y span\_neg.  
 Esto permite:

* Corregir diferencias entre potenciómetros
* Aprovechar todo el recorrido útil
* Obtener una salida perfectamente normalizada en [-1, +1]

### Normalización y mapeo a variables físicas del dron

Una vez procesados los ejes LX, LY, RX, RY, se obtienen cuatro valores normalizados en [-1, 1].

Estas señales se transforman en las referencias físicas que utiliza el dron:

a) Roll y Pitch

Los ejes del joystick derecho controlan la inclinación del dron:

El código convierte estos ángulos a radianes para mantener coherencia con el sistema de control del ESP32 del dron.

b) Yaw (r)

El eje LX controla el giro del dron.  
 Se aplica inversión de signo, tal como el piloto solicitó:

c) Control de altura

El eje LY se interpreta como empuje relativo:

A diferencia de los ángulos, el throttle no se mapea a radianes, ya que se utiliza como una señal proporcional a la variación del empuje total ​ del modelo.

### Transmisión inalámbrica con ESP-NOW

Una vez formado el paquete de datos ControlData, se envía mediante ESP-NOW, un protocolo que permite:

* Comunicación directa ESP32 ↔ ESP32
* Baja latencia (del orden de 1–3 ms)
* Sin necesidad de router o punto de acceso
* Alta confiabilidad para control en tiempo real

El transmisor (control remoto) envía paquetes a 100 Hz con la estructura:

roll\_ref (rad)

pitch\_ref (rad)

r\_ref (rad/s)

control\_altura [-1,1]

En el dron, otro ESP32 recibe estos valores y los integra directamente en los lazos de control de actitud implementados en la placa de vuelo.

## Drone

### Implementación en la placa de vuelo (firmware en C++ sobre ESP32)

Una vez definido y probado el esquema de control en simulación, se implementó el mismo concepto en la placa de vuelo basada en ESP32, utilizando C++ y las librerías de Arduino. Esta parte del sistema ya no integra numéricamente las ecuaciones del dron (eso lo hace el mundo físico), sino que se encarga de:

1. Leer las referencias enviadas por el control remoto.

2. Medir la actitud y las velocidades angulares con la IMU.

3. Ejecutar el lazo de control en tiempo real a una frecuencia fija.

4. Traducir los torques y el empuje deseados en señales PWM para los cuatro ESC.

5. Supervisar el enlace de radio y exponer información de depuración.

En síntesis, el firmware es la versión “en tiempo real” de lo que en la simulación ocurría cada paso de integración.

### Arquitectura general del firmware

El programa se organiza alrededor de dos funciones estándar de Arduino:

* Setup(): se ejecuta una sola vez al encender la placa. Aquí se inicializan:

1. El puerto serie (Serial) para depuración.
2. El bus I2C y el MPU6050 (incluyendo el DMP, filtros y calibración).
3. El módulo Wi-Fi del ESP32 en modo estación y el protocolo ESP-NOW para recibir las referencias desde un control remoto.
4. Los cuatro ESC, fijando su frecuencia (50 Hz) y el rango de PWM (1000–2000 µs).
5. La matriz inversa del mixer (mixInv), que permite pasar de a .
6. Una secuencia de “armado seguro” donde los ESC permanecen varios segundos en mínimo antes de permitir el control.

* loop(): se ejecuta en bucle continuo. Dentro de él:

1. Se actualiza el estado del enlace (nuevos paquetes ESP-NOW y tiempo desde el último).
2. Se leen las mediciones más recientes de la IMU (ángulos y velocidades angulares).
3. Se verifica si ya transcurrió el periodo de muestreo del control (objetivo ~250 Hz).
4. Si corresponde, se ejecuta una iteración completa del lazo de control y se actualizan los PWM de los motores.

En vez de un integrador numérico como RK4, aquí el elemento “que integra” es el propio dron real: el firmware solo decide, cada , qué torques y empuje aplicar según las variables medidas.

### Recepción de referencias vía ESP-NOW y failsafe

Las referencias de roll, pitch, yaw-rate y “control de altura” se envían desde otro ESP32 que actúa como control remoto. Esa información se empaqueta en la estructura:

*typedef struct {*

*float roll\_ref;        // rad*

*float pitch\_ref;       // rad*

*float r\_ref;           // rad/s*

*float control\_altura;  // [-1, +1]*

*} ControlData;*

La función de callback onReceive(...) copia el último paquete recibido en g\_rx\_raw y actualiza un timestamp g\_rx\_last\_ms. En el loop(), en cada pasada se evalúa si el enlace sigue “vivo”:

* Si el tiempo desde el último paquete supera un umbral (RX\_TIMEOUT\_MS = 500 ms), se considera que se perdió el enlace. En ese caso:
  + Se ponen las referencias en cero (roll/pitch nivelados, yaw-rate 0, control de altura 0).
  + Se reinician los integradores del controlador.
  + Se ordena PWM mínimo a los cuatro ESC.
* Si el enlace está OK, se copian los valores a variables internas (roll\_ref, pitch\_ref, r\_ref, control\_altura) aplicando límites físicos (por ejemplo, ±15° para roll y pitch).

De esta manera, el firmware emula de forma robusta el comportamiento de un transmisor de radio, pero con un mecanismo de failsafe integrado.

### Sensado inercial: IMU MPU6050 + DMP

La actitud del dron se obtiene a partir de un MPU6050 utilizando el DMP (Digital Motion Processor) de la propia IMU:

* Periódicamente, se consulta el FIFO del DMP. Cuando hay un paquete completo, se extrae un cuaternión, se calcula el vector gravedad y de allí se obtienen yaw, pitch y roll en radianes (ypr).
* Además, se leen en paralelo los valores crudos del giroscopio (gx, gy, gz), se convierten a °/s usando la sensibilidad del rango seleccionado (±250 °/s) y luego a rad/s. Estos valores se almacenan en el vector PQR, que representa p, q, r.

La idea es clara: los ángulos (roll, pitch, yaw) vienen estabilizados por el DMP, mientras que las velocidades angulares se obtienen más directamente del gyro para alimentar los lazos de control.

### Bucle de control en tiempo real (Ts ≈ 4 ms)

Dentro de loop(), una vez actualizadas referencias y mediciones, se comprueba si ha transcurrido el periodo de muestreo deseado:

*static const float Ts\_target = 0.004f;   // 4 ms → 250 Hz*

*uint32\_t t\_now = micros();*

*float Ts = (t\_now - t\_prev\_us) \* 1e-6f;*

*if (Ts < Ts\_target) return;*

*t\_prev\_us = t\_now;*

Solo cuando Ts supera 4 ms se ejecuta una iteración de control. Esto garantiza que el controlador opere aproximadamente a 250 Hz, de forma periódica y con un paso de tiempo conocido, requisito clave para que los PI/PID trabajen como fueron diseñados.

### Control de empuje (altura simplificada)

El empuje total Empuje no se calcula con un controlador de altura completo, sino con una ley sencilla que parte del peso y lo ajusta según la palanca de “throttle”:

masa \* gravedad es el empuje necesario para compensar el peso.

control\_altura en el rango [-1, +1] modula alrededor de ese valor, permitiendo al piloto mandar trepadas suaves o pequeños descensos.

### Control de actitud en roll y pitch (estructuras en cascada)

Para roll y pitch se implementa en firmware el mismo concepto que en la simulación: un lazo externo PI sobre el ángulo que genera una referencia de velocidad angular, y un lazo interno PID sobre la velocidad angular que genera una aceleración angular deseada.

Por ejemplo, para roll:

* Lazo externo PI (ángulo →p\_ref)  
  + Se calcula el error e\_roll = roll\_ref - roll.
  + Se integra el error con un esquema discreto (I\_roll += Ki\_roll \* e\_roll \*Ts) con límites para evitar windup.  
    Se calcula la referencia de velocidad p\_ref = Kp\_roll \* e\_roll + I\_roll, limitando su magnitud máxima.
* Lazo interno PID (p\_ref →αX)  
  + Se calcula el error de velocidad angular ep = p\_ref - p.
  + Se estima el término derivativo por diferencia hacia atrás D\_p = Kd\_p \* (ep - ep\_prev)/Ts.
  + Se actualiza el integrador I\_p += Ki\_p \* ep \* Ts.
  + Se arma una aceleración angular base alphaX\_base = Kp\_p\*ep + I\_p + D\_p.
  + Se satura alphaX dentro de un rango seguro ±αX\_max.
  + Si la salida está saturada y el error empuja en la misma dirección, se “deshace” la integración de ese tick (anti-windup básico).
  + Se guarda ep\_prev para el siguiente paso.

El mismo esquema se repite para **pitch** (e\_pitch, q\_ref, alphaY) y para **yaw-rate** (er, alphaZ), con la diferencia de que yaw usa un solo PID (no precisa lazo externo de ángulo porque se controla directamente la velocidad de guiñada).

### Cálculo de torques y mezclador de motores

Una vez obtenidas las aceleraciones angulares deseadas, el firmware las traduce en torques físicos utilizando el modelo rígido:

*Vec3 omega    = { PQR[0], PQR[1], PQR[2] };   // (p,q,r) medidos*

*Vec3 alphaDes = { alphaX, alphaY, alphaZ };   // aceleraciones deseadas*

*Vec3 tau      = computed\_torque(J, omega, alphaDes);*

La función computed\_torque aplica exactamente la relación , usando estructuras Mat3 y Vec3 para hacer el código más legible.

Los torques resultantes se saturan a un valor máximo razonable TORQUE\_MAX para proteger tanto la simulación como el hardware:

*float Tx = clampf(tau.x, -TORQUE\_MAX, TORQUE\_MAX);*

*float Ty = clampf(tau.y, -TORQUE\_MAX, TORQUE\_MAX);*

*float Tz = clampf(tau.z, -TORQUE\_MAX, TORQUE\_MAX);*

A partir del empuje total y estos torques se arma el vector de entradas físicas:

*float U[4] = { Empuje, Tx, Ty, Tz };*

y se aplica el **mixer inverso** ya precomputado en mixInv:

*float w2[4];*

*apply\_inverse(mixInv, U, w2);*

*float w[4]  = { sqrtf(w2[0]), sqrtf(w2[1]), sqrtf(w2[2]), sqrtf(w2[3]) };*

* apply\_inverse multiplica el vector U por la matriz B tal que .
* Se asegura que no sea negativo (clamp a 0) para evitar NaN en la raíz cuadrada.

Finalmente, cada velocidad angular de motor se convierte en un ancho de pulso PWM utilizando un polinomio cuadrático ajustado experimentalmente:

*us[i] = pwm\_from\_w(w[i], pwm\_poly[i], us\_min, us\_max);*

donde pwm\_poly[i] contiene los coeficientes (a,b,c) de la relación us=a+b ω+c ω2. Esto permite que el mapa entre “modelo de motor” y ESC real sea consistente con el utilizado en la simulación.

### Salida a ESC y depuración

Con los cuatro valores de PWM en microsegundos (us[0..3]) ya calculados, el firmware los envía a los ESC:

*esc1.writeMicroseconds(us[0]);*

*esc2.writeMicroseconds(us[1]);*

*esc3.writeMicroseconds(us[2]);*

*esc4.writeMicroseconds(us[3]);*

En paralelo, se actualiza una estructura DebugVars g\_dbg con:

* Estado del enlace (edad del último paquete, cantidad de paquetes, MAC).
* Referencias, ángulos y velocidades.
* Estados internos de los controladores (errores, integradores, derivadas).
* Empuje, torques, entradas físicas U, ω2\omega^2ω2, ω\omegaω y PWM.
* Periodo de muestreo efectivo Ts.

La función dbg\_maybe\_print() decide cada ciertos milisegundos si imprime esta información por serie, en formato humano o CSV, permitiendo registrar datos reales de vuelo y compararlos con los resultados de simulación.

En conjunto, este firmware implementa en el ESP32 el mismo esquema de control diseñado y probado en MATLAB/Simulink, pero adaptado a las restricciones del tiempo real: lectura periódica de sensores, control a frecuencia fija, saturaciones y anti–windup discretos, mezcla de motores y generación de PWM. De esta forma, el puente entre el modelo matemático y el dron físico queda completamente cerrado.

# Conclusiones

El desarrollo del dron cuadricóptero, junto con su correspondiente simulación dinámica, permitió integrar en un mismo proyecto los contenidos centrales de la asignatura, abarcando modelado matemático, análisis numérico, diseño electrónico y validación computacional. A lo largo del trabajo se implementaron diversos métodos numéricos (entre ellos Runge–Kutta de orden cuatro, Gradiente Conjugado, Newton–Raphson, Simpson 1/3, mínimos cuadrados y spline cúbico) que resultaron esenciales para resolver ecuaciones diferenciales, sistemas lineales, relaciones no lineales e interpolaciones necesarias para la caracterización del sistema.

Se desarrolló un modelo dinámico completo del cuadricóptero y se implementó un sistema de control remoto basado en ESP-NOW, demostrando la integración entre algoritmos de control, comunicaciones de baja latencia y electrónica embebida. La simulación permitió evaluar con detalle la estabilidad, la distribución de empuje, la respuesta ante diferentes consignas de control y el desempeño del mixer mediante el método de Gradiente Conjugado. Asimismo, el método de Newton–Raphson permitió abordar las ecuaciones no lineales asociadas al diseño eléctrico, principalmente en lo referente a la seguridad de corriente en los componentes del hardware.

Cabe destacar que el dron fue efectivamente construido de forma física (como se muestra en la Fig. 66 y 67), incluyendo su estructura mecánica, el sistema de control, la electrónica de potencia y las comunicaciones. Aún se encuentran pendientes las pruebas experimentales de funcionamiento y la verificación de la simulación frente al comportamiento real. Estas etapas permitirán evaluar la correspondencia entre el modelo matemático y la respuesta física del dron, ajustando parámetros y perfeccionando el sistema de control de acuerdo con los resultados empíricos.

En síntesis, el trabajo demuestra la aplicabilidad real de los métodos numéricos en un sistema mecatrónico complejo, logrando un prototipo físico respaldado por un modelo matemático robusto, una simulación coherente y un diseño electrónico conforme a criterios de seguridad. La integración entre teoría, simulación y construcción física evidencia la importancia del enfoque multidisciplinario propio de la Ingeniería Mecatrónica.



Figura 67



Figura 68

# Bibliografía

[1] Mecánica clásica H. Goldstein. Disponible en: <https://poincare.matf.bg.ac.rs/~zarkom/Book_Mechanics_Goldstein_Classical_Mechanics_optimized.pdf>

[2] Mecánica clásica – Notas de clase. Guillermo Abramson. Disponible en: <https://www.miguelangelvargascruz.com/mecanicaclasicanotasdeclaseabramson_blog_4380.html>

[3] Desarrollo de un modelo matemático, cinemático y dinámico con la aplicación de software, para modificar el funcionamiento de un dron, para que este realice monitoreo automático. Edwin Dimitri Nieto Guerrero; Fernando Alonso Vaca De La Torre. Disponible en: <https://dialnet.unirioja.es/descarga/articulo/7402300.pdf>

[4] Trabajo final de grado. Navegación autónoma de un multi-rotor: control automático del avance. Inés Portolés García. Disponible en: <https://zaguan.unizar.es/record/64167/files/TAZ-TFG-2017-2914.pdf>

[5] Teoría del motor mezclador. Disponible en: <https://cookierobotics.com/066/>

[6] CÁLCULO NUMÉRICO. UNIDAD 1. Solución de ecuaciones no lineales en una variable. Mg. Ing. Menuet Agustín y Mg. Ing. Martín Garciarena Ucelay. Disponible en: <https://moodle4vz.unsl.edu.ar/moodle/mod/url/view.php?id=44200>

[8] CÁLCULO NUMÉRICO. UNIDAD 2. Interpolación polinomial. Mg. Ing. Menuet Agustín y Mg. Ing. Martín Garciarena Ucelay. Disponible en: <https://moodle4vz.unsl.edu.ar/moodle/mod/url/view.php?id=44229>

[9] CÁLCULO NUMÉRICO. UNIDAD 3. Sistema de ecuaciones lineales. Mg. Ing. Menuet Agustín y Mg. Ing. Martín Garciarena Ucelay. Disponible en: <https://moodle4vz.unsl.edu.ar/moodle/mod/url/view.php?id=64754>

[10] CÁLCULO NUMÉRICO. UNIDAD 4. Integrales, Mg. Ing. Menuet Agustín y Mg. Ing. Martín Garciarena Ucelay. Disponible en: <https://moodle4vz.unsl.edu.ar/moodle/mod/url/view.php?id=64753>

[11] CÁLCULO NUMÉRICO. UNIDAD 5. Ecuaciones diferenciales ordinarias. Mg. Ing. Menuet Agustín y Mg. Ing. Martín Garciarena Ucelay. Disponible en: <https://moodle4vz.unsl.edu.ar/moodle/mod/url/view.php?id=64756>

[12] OpenAI. *ChatGPT (GPT-5.1 Thinking)*. Consulta sobre «[Drones, sistemas de control]».

[13] Matlab. Disponible en: <https://es.mathworks.com/products/matlab.html>

# Anexo I

**Desarrollo de un Dron Cuadricóptero utilizando Métodos Numéricos**

**Development of a Quadcopter Drone using Numerical Methods**

Banffi, Joaquín; Bilbao, Juan; López, Emanuel;

Proyecto de promoción / Cálculo Numérico / Ingeniería Mecatrónica

[banffijoaquin@gmail.com](mailto:banffijoaquin@gmail.com) ; [bilbaojuani@gmail.com](mailto:bilbaojuani@gmail.com) ; [loezemanuel@gmail.com](mailto:loezemanuel@gmail.com)

**Resumen:** El presente trabajo describe el diseño, modelado, construcción física y simulación de un dron cuadricóptero, integrando métodos numéricos avanzados y herramientas de análisis propias de la Ingeniería Mecatrónica. Se desarrolló un modelo dinámico completo, se aplicaron algoritmos de integración y resolución de ecuaciones (incluyendo Runge–Kutta 4, Gradiente Conjugado y Newton–Raphson) y se implementó un sistema de control remoto basado en comunicación inalámbrica ESP-NOW. La simulación permitió evaluar la estabilidad, la distribución de empuje y el comportamiento ante distintas consignas. El dron también fue construido físicamente, quedando pendientes las pruebas experimentales y la validación del modelo respecto del comportamiento real. El trabajo demuestra la aplicación práctica de los métodos numéricos en un sistema mecatrónico concreto.

Palabras clave: dron, cuadricóptero, simulación, métodos numéricos.

**Abstract:** This work presents the design, physical construction, modeling, and simulation of a quadcopter drone, integrating advanced numerical methods and analysis tools used in Mechatronics Engineering. A full dynamic model of the vehicle was developed, together with numerical algorithms for integration and equation solving (including Runge–Kutta 4, Conjugate Gradient, and Newton–Raphson). A wireless remote-control system based on ESP-NOW communication was implemented. The simulation enabled the evaluation of stability, thrust distribution, and drone behavior under different inputs. The drone was also physically built, with experimental flight tests and model validation against real-world behavior still pending. The project demonstrates the practical applicability of numerical methods in a real mechatronic system.

Keywords: drone, quadcopter, simulation, numerical methods.