Facultad de Ingeniería de la Universidad de la República

Proyecto de fin de carrera de Ingeniería Eléctrica

Diseño y automatización de un UAV

Autores:
Sebastián Torterolo
Juan Manuel Torterolo
Facundo Cayafa

Tutor:
Ing. Rafael Canetti


6 de agosto de 2016

0.1. Resumen

El objetivo fundamental de este proyecto es diseñar la automatización de un UAV (Unmanned Aereal Vehicle) con configuración de cuadricóptero, mediante técnicas clásicas de control y algoritmos de Redes Neuronales.

Se comienza con el diseño y construcción del robot justificando la elección de cada componente; se arma el artefacto, se ensayan y fusionan los sensores, se caracterizan todos los componentes mecánicos para elaborar un modelo con el cual posteriormente se entrenan las redes neuronales involucradas en el controlador.

Se diseña la inteligencia basándose en el lenguaje de programación $C++$ y en librerías Arduino-compatibles. Finalmente se realizan las pruebas necesarias para verificar los logros alcanzados y compararlos con los objetivos iniciales.

Índice general

Parte I

Introducción

Panorama global del proyecto, antecedentes y objetivos propuestos.

Capítulo 1

Introducción

1.1. Antecedentes

Dentro del instituto de Ingeniería Eléctrica de la Facultad de Ingeniería de la Universidad de la República existe desde hace varios años una línea de trabajo abocada al estudio de tecnología relacionada con la robótica móvil, a través de la que se han diseñado e implementado robots móviles terrestres (SULLA), acuáticos (Pez Robot I y II), y aéreos. En ese marco se han desarrollado diversos proyectos sobre vehículos aéreos no tripulados (UAVs).

El primero de ellos fue AMFO1-Bobby, en donde se diseñó y construyó un avión a escala radiocontrolado con algunas capacidades de vuelo autónomo. Le siguió AUION, continuando el trabajo del primero, resolviendo el diseño del controlador del avión en condiciones de vuelo.

En esta misma línea le siguieron los proyectos uQuad! y uQuad2, en donde se diseñó un sistema de control adaptándolo a una plataforma comercial en el primero y dotándo de capacidades de vuelo autónomo referente a seguimiento de trayectorias en el segundo.

En el presente proyecto se continúa trabajando en la línea presentada, basándose en estos últimos proyectos, se diseña y construye un cuadricóptero, dotando al mismo de una autonomía basada en técnicas clásicas de control y algoritmos de redes neuronales.

1.2. Objetivos específicos

Los objetivos específicos y criterios de éxito propuestos para el proyecto son:

Funcionamiento mecánico del cuadricóptero construido:

- El robot debe disponer de un empuje total de al menos el 150 % de su peso.
- La autonomía de vuelo debe ser al menos de 10 minutos.

Funcionamiento de los algoritmos para seguimiento de trayectorias:

- En vuelos estacionarios debe mantener la altura con un error menor que 1m en la dirección vertical.
- En vuelos realizados fuera del laboratorio (distancias características mayores a 30m) los controladores deben ser capaces de seguir trayectorias con errores menores que 1m en la dirección vertical y menores que 5m en un plano horizontal).

1.3. Organización del documento

El documento se encuentra dividido en 5 partes bien definidas las cuales se describen a continuación:

- Solución adoptada: Se brinda una primera descripción del sistema y las soluciones implementadas. Se detallan uno a uno los componentes utilizados y los criterios de elección de los mismos.
- Modelo físico y Caracterización: Se describen la dinámica y las ecuaciones que rigen el sistema físico y se detallan los experimentos de caracterización de los elementos mecánicos.
- Estimación del estado: Se analizan los sensores utilizados, caracterizando cada uno de ellos y se describe el algoritmo para implementar la fusión sensorial.
- Automatización y control: Se describe en detalle el problema de control y la solución adoptada.
- Implementación. Resultados y conclusiones: Se exponen los experimentos finales de vuelo , criterios de éxito y conclusiones finales.

Parte II

Solución Adoptada

Descripción general del sistema y las soluciones implementadas, detalle de los componentes utilizados.

Capítulo 2

Descripción general

2.1. Descripción del sistema físico

Un cuadricóptero consiste básicamente en una estructura rígida con 4 propulsores compuestos de un motor y una hélice cada uno.

Esquemáticamente se puede observar en la siguiente figura los ejes cartesianos solidarios al vehículo y los ángulos de Euler que describen los grados de libertad del robot y en base a los cuales se desarrollarán todas las acciones sobre el mismo:

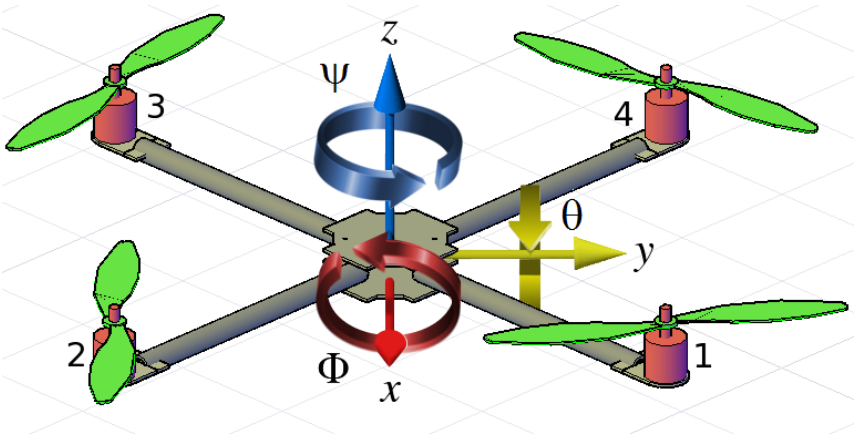


Figura 2.1: El cuadricóptero, ejes y angulos de control

Donde los ángulos Φ (Roll), Θ (Pitch) y Ψ (Yaw) también se conocen como los ángulos de Tait-Bryan, comúnmente usados en aeronáutica (5.1.1).

Las 4 hélices fijas le permiten moverse en sus 6 grados de libertad (3 ángulos y 3 direcciones); con 2 propulsores fijos no es posible tener control sobre todos ellos[1], con 3 propulsores sí es posible pero se encuentra desbalanceado, por lo que 4 propulsores fijos es la configuración mínima para tener momento angular balanceado y control sobre todos los grados de libertad de rotación.

[1]El helicóptero tiene 2 propulsores pero puede inclinar el rotor o cambiar el paso de las hélices en forma cíclica por lo que no se encuentra dentro de esta categoría de vehículos con propulsores fijos.

Cada propulsor ejerce una fuerza sobre el robot, la suma de estas fuerzas se denomina empuje [2], para modificar el estado del rígido y tomar control sobre sus grados de libertad se deben generar desbalances entre empujes y velocidades angulares [3]de cada motor:

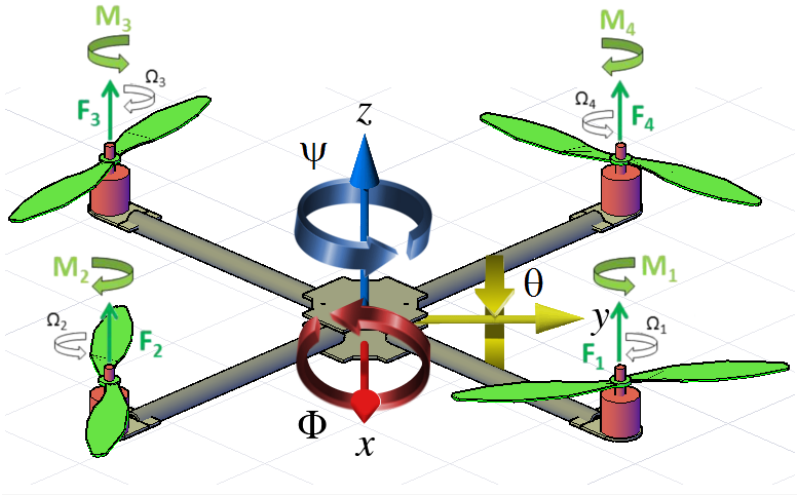


Figura 2.2: Fuerzas (F_i), momentos (M_i momento que el motor ejerce sobre el cuerpo del cuadricóptero) y angulos de giro (Ω_i) de cada motor

Dependiendo de la componente vertical del empuje comparada con el peso del robot, éste se moverá hacia arriba, abajo o se mantendrá suspendido con velocidad vertical nula[4].

Para rotar el vehículo, las acciones de control se pueden resumir del siguiente modo:

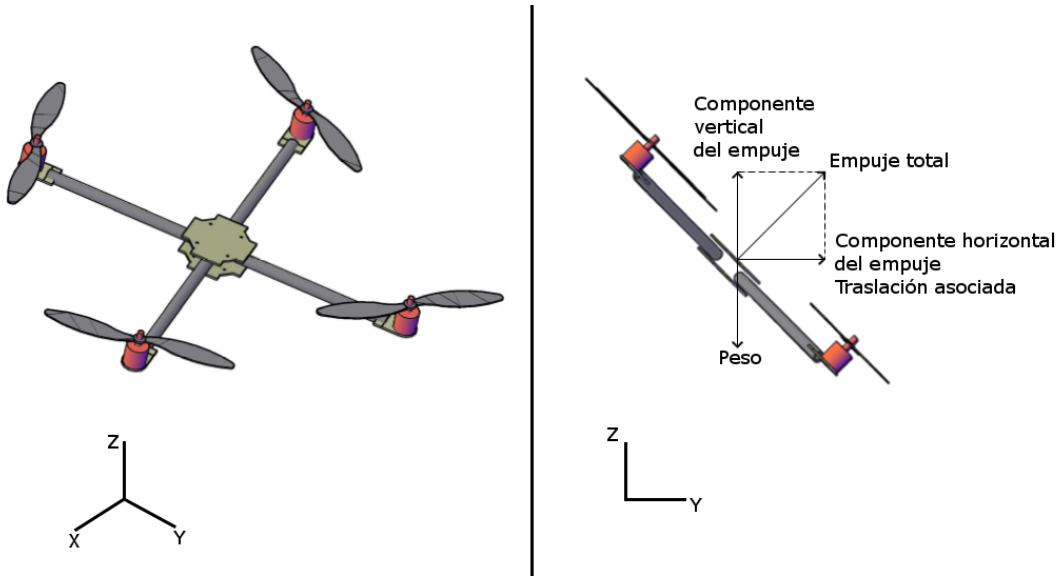
- Giro positivo en torno a y, aumento Θ (Pitch): $\dot{\Omega}_3 + \dot{\Omega}_4 > \dot{\Omega}_1 + \dot{\Omega}_2$
- Giro positivo en torno a x, aumento Φ (Roll): $\dot{\Omega}_1 + \dot{\Omega}_4 > \dot{\Omega}_2 + \dot{\Omega}_3$
- Giro positivo en torno a z, aumento Ψ (Yaw): $\dot{\Omega}_1 + \dot{\Omega}_3 > \dot{\Omega}_2 + \dot{\Omega}_4$

[2]Thrust en inglés.

[3]Tal como se explica en el capítulo 7, la fuerza y el momento que cada propulsor ejerce sobre el vehículo, dependen proporcionalmente con el cuadrado de la velocidad angular del mismo, por lo que generar un desbalance en las fuerzas es necesario realizar los controles necesarios para que giren a distintas velocidades.

[4]Lo que se conoce como “hovering”.

Finalmente la traslación del vehículo se logra inclinandolo, de ese modo se genera una fuerza en el mismo sentido de esta inclinación.



En resumen, el problema fundamental del proyecto es realizar el control de un movimiento traslacional, cualquier movimiento que se desee que el robot ejecute, consiste en lograr una orientación y una velocidad adecuada para los motores. Partiendo de esta consecuencia es que se divide el problema en 2 subsistemas que pueden tratarse independientemente, como se verá en detalle en la sección 14.2, el subsistema angular permite obtener una orientación determinada para el robot, lo que permite fijar la dirección de vuelo, mientras que el subsistema traslacional permite controlar la velocidad y altura de vuelo.

2.2. Interacción funcional del sistema

En el siguiente diagrama se expone la relación funcional entre los componentes del robot:

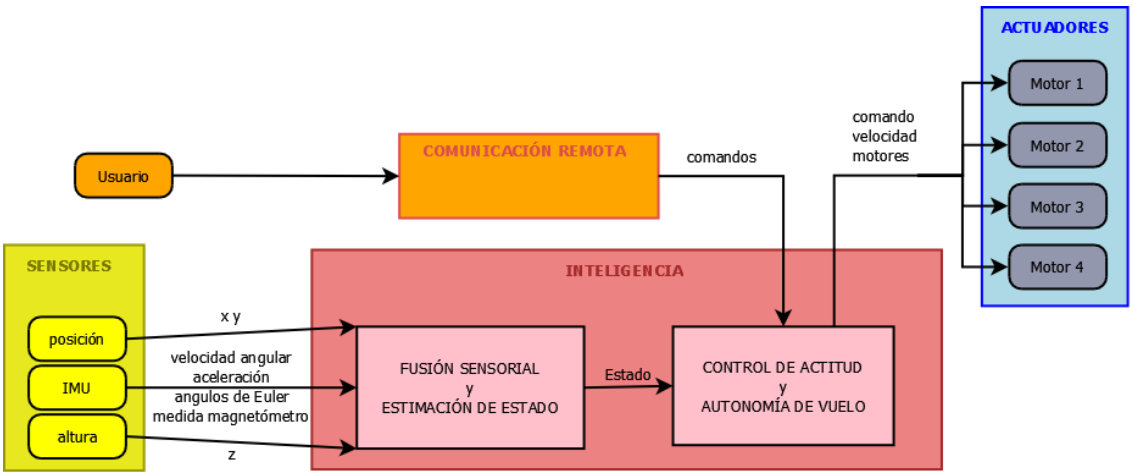


Figura 2.3: Diagrama funcional

El usuario se comunica remotamente con el sistema mediante un cable USB o inalámbricamete vía WIFI, por estos medios transfiere al sistema una serie de comandos que determinaran el comportamiento (el "modo") del vehículo:

- Comando S: Start, habilita al robot a iniciar la secuencia de modos que se describen en los comandos siguientes:
- Comando H: Hovering, dada una altura z deseada, el robot alcanza la misma y se mantiene en ella hasta la llegada de un nuevo comando sin alterar la posición X, Y
- Comando A: Aterrizar, disminuye la velocidad según z progresivamente hasta llegar a una altura cercana a cero [5]
- Comando F: Following, entra en el modo de seguimiento de trayectoria
- Comando Z: Zero, apaga los motores
- Comando C: Caída, en un caso de emergencia en el cual se corte la comunicación con el usuario o con el sensado (por lo que el robot no tendrá información acerca de su estado), se cuenta con un modo de caída en el cual se disminuye progresivamente la velocidad de los motores durante 10 segundos, luego de los cuales se apaga.

El control traslacional parte del objetivo de poder realizar un seguimiento de trayectorias basándose en la definición por parte del usuario de unos puntos, que llamaremos de ahora en más "Waypoints", los cuales tienen a su vez direcciones y sentidos asociadas que determinan ángulos de partida y llegada a los mismos.

Los waypoints son dados de acuerdo a sus coordenadas cartesianas en un sistema de referencia inercial, por lo tanto, para cumplir con este requisito el robot necesita determinar su estado en referencia a este sistema para lo cual sensa el entorno utilizando los sensores que se detallan a continuación.

[5] En el sensado de la altura se tiene un offset por lo que la altura no llega a ser exactamente cero, pero conocido este valor se puede tener la certeza de cuándo el robot ha llegado a tierra.

- GPS: determina la posición absoluta x,y
- IMU: determina la aceleración propia, velocidad angular, ángulos de Euler y dirección del campo magnético terrestre
- Barómetro: determina la altura
- Ultrasonido: determina la altura

La medida del entorno por parte de los sensores siempre conlleva errores e incertidumbres asociadas a ellos, por esta razón se debe trabajar con grados de redundancia que ayuden a minimizar sus efectos en el resultado (estimación del estado del artefacto), es por ello que varios de estos sensores estiman una misma magnitud.

La estimación del estado se realiza mediante la "fusión sensorial" consistente en el filtrado e interpretación de las medidas sensoriales. Este procesamiento se ejecuta en un primer micro controlador llamado Teensy, éste le transfiere los datos procesados ("Estado") a un segundo micro controlador TivaC que será el responsable de realizar el control del vehículo y todo las operaciones de generación y seguimiento de trayectoria basándose en los waypoints.

Finalmente el TivaC le transmite las velocidades angulares a los actuadores (propulsores), quienes son los encargados de modificar el estado del vehículo.

Además del problema principal de control, el proyecto cuenta con varias ramas de elaboración de software entre las que se encuentran la fusión sensorial, estimación del estado, control de actitud, altitud y comunicación. El control mediante redes neuronales representa un desarrollo en software nuevo[6], sin contar con experiencia previa al respecto fue necesario elaborar un controlador PID de implementación conocida y estudiada con anterioridad en la currícula académica con el fin de poder verificar que todo este sistema integrado funcionara correctamente antes de sustituir el PID por la red neuronal[7].

Por lo tanto, si hacemos una ampliación al diagrama de la figura 2.3 tenemos la implementación de 2 controladores como se muestra a continuación[8]:

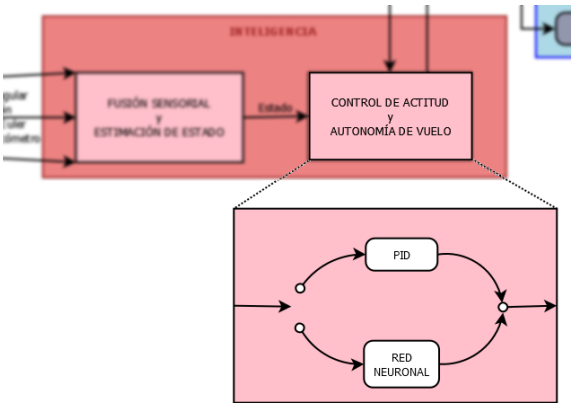


Figura 2.4: Implementación PID - Red neuronal

[6]Recordamos que la implementación del control mediante redes neuronales implicó el aprendizaje de una temática totalmente nueva y búsqueda de información fiable (que hasta elaborar el control no fue posible tener la certeza de la fiabilidad de estas fuentes y la comprensión de las mismas).

[7]Todo el software se desarrolló de forma "modular" para poder intercambiar los controladores clásicos con los basados en redes neuronales.

[8]Estos controladores son independientes y solamente puede estar activo uno a la vez, finalizadas las verificaciones del resto del software se dejó activo el controlador implementado con la red neuronal.

2.3. Autonomía de vuelo

Una vez resueltos los problemas de fusión sensorial, control de actitud y traslación del robot, fue implementado un sistema a un nivel superior para dotar de autonomía de vuelo al vehículo, tomando como referencia la teoría desarrollada por Lester Eli Dubins en 1957 [22] y basándose en la implementación de software desarrollado por el proyecto anterior uQuad2 [2] de planeamiento y seguimiento de trayectorias. A continuación se mencionan los aspectos descriptivos generales para la solución de planeamiento y seguimiento de trayectorias, los algoritmos y fundamentos se exponen en más detalle en el capítulo 13.

2.3.1. Planeación de trayectorias

La generación de trayectorias se elabora en base a los waypoints y a los ángulos de partida y llegada que el usuario designa y por los que el robot debe pasar.

En su teoría, Dubins demuestra que dados dos puntos por sus coordenadas cartesianas en un sistema de referencia inercial (x_1, y_1) y (x_2, y_2) con una orientación dada para cada uno de ellos ϕ_1 y ϕ_2 respectivamente (ángulos de partida y llegada), bajo ciertas hipótesis existe una curva continuamente diferenciable de longitud mínima que une ambos puntos en las direcciones adecuadas y dicha curva es de la forma CSC, donde S representa una trayectoria recta y C un arco de circunferencia de radio fijo, por lo tanto si se define R como el arco de circunferencia recorrido en sentido horario y L al arco recorrido en sentido antihorario se tiene que las posibles curvas que minimizan la trayectoria tienen la forma:

- RSR
- LSL
- LSR
- RSL

Por otro lado Shkel y Lumelsky [23] determinan qué curva corresponde a cada par de waypoints dependiendo de su posición relativa y al cuadrante a los que pertenecen los ángulos de partida y llegada de ellos; una vez determinada la curva de largo mínimo que une 2 waypoints, la concatenación de ellas unirá todos los waypoints definidos por el usuario.

En la figura 2.5 se ve un ejemplo de una sub-trayectoria entre dos waypoints 1 y 2, del tipo RSL. El ángulo de partida para (x_1, y_1) en este caso es $\phi_1 = \pi/2$ y el de llegada en (x_2, y_2) es $\phi_2 = 3\pi/4$. Una hipótesis del planteo de planeamiento de trayectoria para todos los waypoints, es que el ángulo de llegada a un waypoint es el mismo que el de partida para la próxima sub-trayectoria, de este modo no hay un punto anguloso en ningún waypoint.

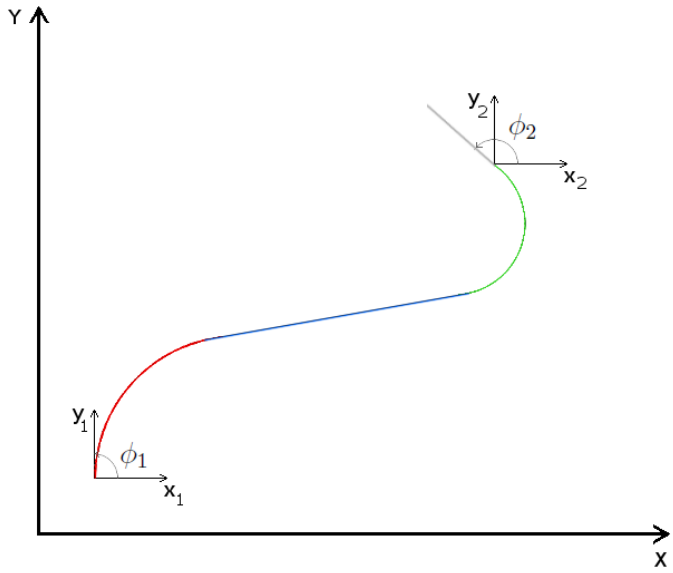


Figura 2.5: Sub-trayectoria RSL entre dos waypoints

2.3.2. Seguimiento de trayectorias

Una vez que se cuenta con una trayectoria a recorrer, es necesario implementar un algoritmo de seguimiento. Esto implica obtener una altura y ángulo Yaw deseado a partir de la estimación del estado actual y de la trayectoria a seguir.

El algoritmo de seguimiento implementado fija el ángulo Roll en cero durante todo el recorrido, el cambio en la dirección de vuelo estará determinado únicamente por el estado del ángulo Yaw. Por lo tanto, la estrategia de seguimiento de la curva plana generada utiliza como variable de control al ángulo Yaw. A partir de la trayectoria a seguir y conociendo la posición en un instante dado, puede calcularse el Yaw deseado que permita acercarse a la trayectoria. Se discretiza la trayectoria generada en sub-waypoints y se sigue a cada uno en forma consecutiva. De esta manera, se determinan puntos con una distancia reducida entre ellos lograndose un mayor control sobre el seguimiento de la trayectoria de acuerdo a Sousa P.B. Sujit y S. Saripalli [24] con su algoritmo de “Carrot chase”.

Capítulo 3

Hardware mecánico y electrónico

3.1. Componentes

En la figura 3.1 se muestra el diagrama eléctrico de la solución adoptada, en las secciones a continuación se desglosa y describen las principales características de cada componente, más adelante en la sección de caracterización se describirán con más detalle.

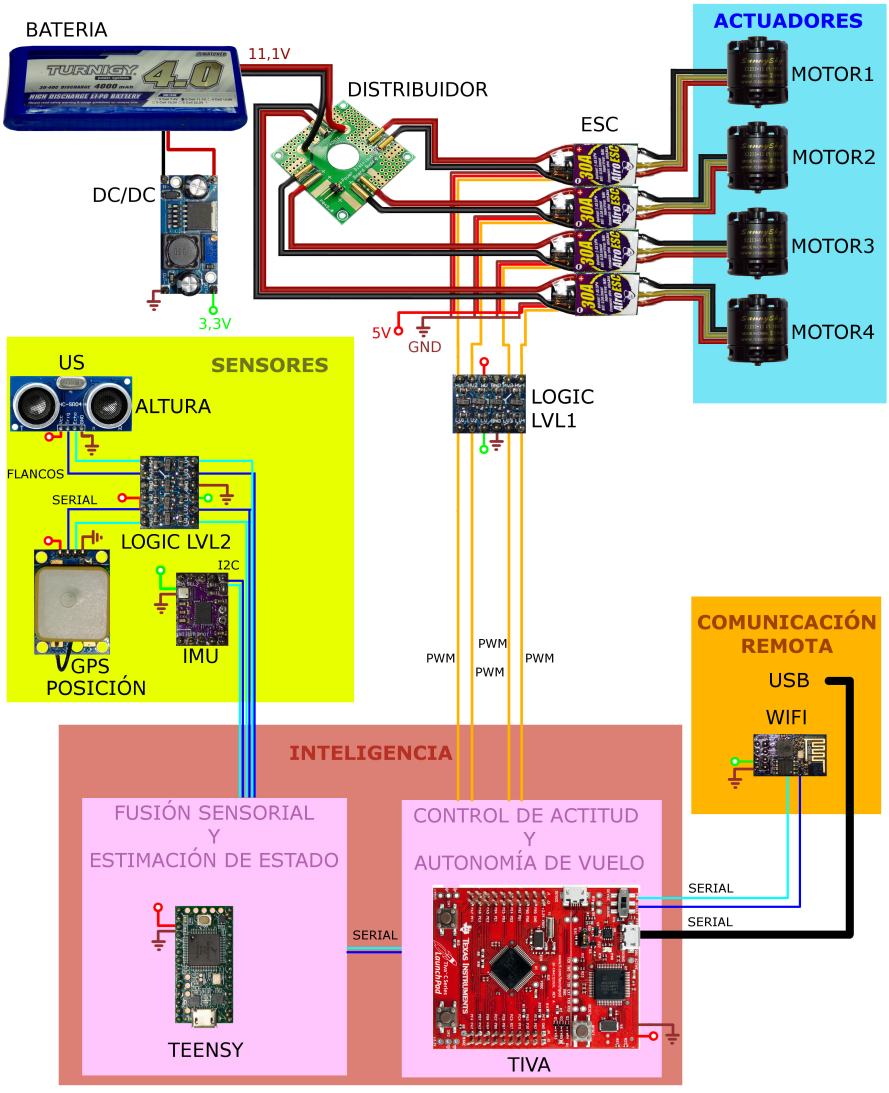


Figura 3.1: Esquema eléctrico de la solución adoptada.

3.2. Componentes básicos

A continuación se describen las principales características de los componentes básicos del sistema.

3.2.1. Estructura

La estructura (Frame en inglés, de ahora en más nos referiremos a ella de éste modo) es la encargada de darle rigidez al artefacto, sobre ella se montan todo el resto de los componentes. Como se verá más adelante, la elección de componentes depende de un compromiso entre la autonomía de vuelo, peso de los componentes y costo de los mismos. Para maximizar el tiempo de vuelo y agilizar el movimiento del cuadricóptero, debemos contar con motores más potentes capaces de trabajar con hélices más grandes y de desplazar mayores volúmenes de aire; como contra parte se debe contar con una batería con mayor potencia, la cual está asociada a la masa de la misma por lo que aumenta sustancialmente el peso del cuadricóptero. El tamaño de las hélices es determinante al momento de seleccionar el frame, se debe lograr una separación adecuada entre ellas, cuanto mayor sea el tamaño del frame, componentes más grandes pueden acoplarse a ella y lograr una mayor eficiencia en el conjunto. Frame seleccionada:

- Modelo: Turnigy Talon Quadcopter (V2.0)
- Material: Fibra de carbón
- Tamaño: 550mm
- Peso: 280gr

3.2.2. Motores

Los motores eléctricos de corriente continua sin escobillas con rotor externo (Brushless Outrunner DC Motors) son comúnmente utilizados en vehículos radio-controlados debido a su eficiencia, potencia, durabilidad y bajo peso. Constan de un rotor de imán permanente y tres pares de bobinas en el estator alimentadas con corriente continua controlada por los ESC, éste se encarga de energizar con sincronía los bobinados generando un campo giratorio.

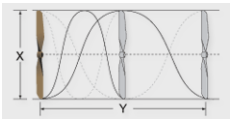
Motores seleccionados:

- Modelo: SunnySky X2212 KV980 II
- KV: 980 RPM/Volt
- Corriente máxima: 15A
- Potencia máxima 150W
- Diámetro exterior: 28mm
- Altura: 28mm
- Peso: 56gr

3.2.3. Hélices

Las hélices giran de forma solidaria al eje del rotor y son las encargadas de ejercer la fuerza de empuje, ésta depende de las dimensiones de las hélices y su velocidad de giro. El fabricante de cada motor da un listado de las hélices recomendadas para el mismo, se eligen las siguientes:

- Modelo: Hobbyking Slowfly Propeller 9x4.7
- Largo: 9" (X)
- Paso: 4,7" (Y)
- Material: plástico



3.2.4. Escs

Los controladores de velocidad (ESC: Electronic Speed Controller) prácticamente quedan determinados al fijar el motor y las hélices. Los ESC seleccionados son:

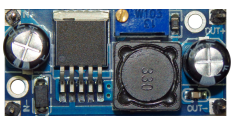
- Modelo: Afro ESC 30Amp
- Corriente: 30A
- Rango voltaje: 2-4s (Lipoly)
- Frecuencia de entrada: 1kHz
- Peso: 26,5gr

3.2.5. Batería

Una vez seleccionados los motores y estimado el peso total del artefacto se elige la batería que cumpla con los requisitos de tensión y maximice el tiempo de vuelo. La batería seleccionada es:

- Modelo: Turnigy 4000mAh 3S 30C Lipo Pack
- Capacidad: 4000mAh
- Configuración: 3S1P/11.1V/3 Celdas
- Descarga: 30C
- Peso: 347gr

3.2.6. DC/DC Converter



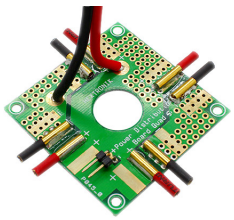
Con el objetivo de proporcionar 3.3V de alimentación a los sensores, se escoge el siguiente conversor DC/DC:

- Modelo: LM2596
- Voltaje de entrada: 4V-35V
- Voltaje de salida: 1.23V-30V
- Corriente max: 3A

3.2.7. Distribuidor

Distribuidor de energía para los ESCs:

- Marca: Hobbyking
- Corriente max: 4x20A
- Peso: 28gr



3.3. Instrumentación

3.3.1. IMU

La unidad inercial de medición o IMU (*Inertial Measurement Unit*) contiene un giróscopo y un acelerómetro, que son los sensores inerciales básicos. Adicionalmente se incluye un magnetómetro para medir el rumbo, a través de la medición del campo magnético terrestre. El acelerómetro y el magnetómetro son sensibles a vectores con dirección y sentido conocidos en el sistema de referencia de la tierra (aceleración de la gravedad y campo magnético terrestre, respectivamente), lo que permite obtener información de la orientación absoluta del vehículo.

En este proyecto se ha utilizado dos modelos de IMU diferentes, con sensores de especificaciones similares, pero con características diferentes en cuanto a los datos de salida.

BNO055

- Fabricante/Modelo: Bosch BNO055 [9].
- Sensores: Sensores: Acelerómetro, Giróscopo, Magnetómetro, Termómetro.
- Permite fusionar internamente las medias mediante un filtro de Kalman extendido.
- Comunicación: i2c.
- Tasa de datos de orientación limitada a $100Hz$.

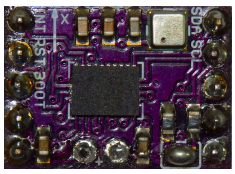


Figura 3.2

MPU9250

- Fabricante/Modelo: InvenSense MPU9250 [10].
- Sensores: Acelerómetro, Giróscopo, Magnetómetro, Termómetro.
- Comunicación: i2c.
- Tasa de datos de sensores hasta $2000Hz$.

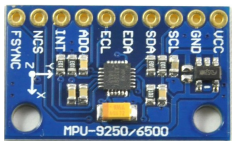


Figura 3.3

3.3.2. Sensor de presión

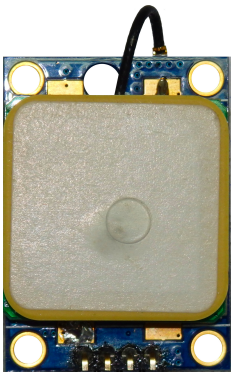
El sensor de presión o barómetro mide la presión absoluta de aire, a partir de la cual se puede calcular la altura sobre un punto tomado como referencia suponiendo un cierto modelo presión-altura para la atmósfera.

- Fabricante/Modelo: Measurement Specialties MS5637 [11].
- Sensores: Sensor de Presión, Termómetro.
- Permite corregir la medida según la temperatura.
- Comunicación: i2c.

El sensor de presión fue adquirido en el mismo PCB que la IMU BNO055 (figura 3.2). Ver [12].

3.3.3. GPS

El GPS (del inglés Global Positioning System: Sistema de Posicionamiento Global) mediante la recepción de señales satelitales permite determinar la posición de un objeto, las características del GPS elegido son:



- Fabricante/Modelo: u-blox NEO-6 [13]
- Protocolo: NMEA0183
- Frecuencia actualización: 10Hz
- Precisión posición: < 3m
- Precisión velocidad: $< 0{,}1$ m/s

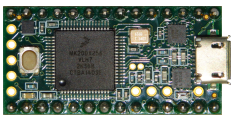
3.3.4. Ultrasonido

El sensor de ultrasonido se utiliza conjuntamente con el barómetro incluido en la IMU para estimar la altura del artefacto:

- Modelo: HC-SR04
- Frecuencia: 40Hz
- Rango mínimo: 2cm - 400cm
- Ángulo de medición: 15º

3.4. Inteligencia

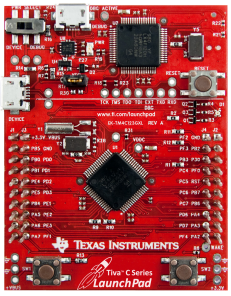
3.4.1. Teensy



El Teensy es un microcontrolador de desarrollo compatible con las plataformas de procesamiento Arduino, completamente programable mediante USB. Se utilizará para realizar la fusión sensorial proveniente de la IMU, GPS y ultrasonido:

- Modelo: Teensy 3.2
- Procesador: Cortex-M4 72-96MHz
- RAM: 64kb
- Digital I/O: 34 pines
- Analógicas In: 21 pines

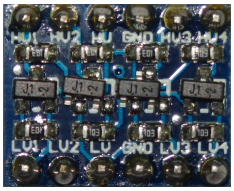
3.4.2. Tiva



El Tiva C es un microcontrolador de similares características al Teensy pero con la ventaja de contar con punto flotante, por lo que se utilizará para realizar las tareas de control que son más exigentes, sus principales características son:

- Modelo: Tiva C
- Procesador: TM4C123GH6PM 32-bit ARM Cortex-M4 80MHz
- RAM: 32kb

3.4.3. Conversor lógico

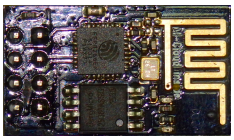


Debido a que tanto el Teensy como el Tiva trabajan con niveles lógicos de 3.3V y GPS y ultrasonido junto al multiplexor trabajan con niveles de 5V es necesarios acoplarlos mediante conversores lógicos bidireccionales:

- Descripción: 4 Channel I2C Logic Level Converter
- Modelo: Genérico
- Operación: Bi-Direccional 5V a 3.3V

3.5. Comunicación

3.5.1. Wifi



Toda la comunicación entre entre el usuario y el robot se hará mediante wifi, las principales características de este módulo son:

- Modelo: ESP8266
- Protocolo: TCP/IP 802.11 b/g/n

3.5.2. Tabla de componentes

Componente	Marca – Modelo
Frame	Turnigy Talon Quadcopter (V2.0)
Motor	SunnySky X2212 KV980 II
Hélice	Hobbyking Slowfly Propeller 10x4.7
ESC	Afro ESC 30Amp
Batería	Turnigy 4000mAh 3S 30C Lipo Pack
Conversor DC/DC	LM2596
Distribuidor	Hobbyking
IMU	BNO055 9-axis motion sensor with sensor fusion
GPS	u-blox NEO-6
US	HC-SR04
Microcontrolador 1	Teensy 3.2
Microcontrolador 2	Tiva C
Conversor lógico	4 Channel I2C Logic Level Converter
Modulo comunicación WIFI	ESP8266

Tabla 3.1: Componentes del cuadricóptero

Capítulo 4

Elección de componentes mecánicos

4.1. Propulsores

Para el diseño físico del robot y por lo tanto determinación y elección de los componentes del mismo, hay varios enfoques, entre ellos elegimos basarnos en una práctica común (thrust to weight ratio) que consiste en considerar que el empuje máximo total de los motores, sea igual al doble del peso del vehículo. De esta manera la aceleración hacia arriba es igual a g. Primeramente se define el peso total que tendrá el cuadricóptero, denominado A.U.W. (All up weight), de la terminología aeronáutica. Se fija este peso objetivo de $1,2kg$, por lo tanto, el empuje total a máxima potencia será de $E = 2,4kg$ fuerza. Así, cada conjunto propulsor (ESC + motor + hélice) debe empujar a máxima potencia $\frac{E}{4} = 600g$ fuerza.

Motor: X2212 KV:980

Technical Datas	
KV	980
Configu-ration	12N14P
Stator Diameter	22mm
STator Length	12mm
Shaft Diameter	3mm
Motor Dimension(Dia.*Len)	Φ27.5×30mm
Weight(g)	56g
Idle current(10)@10v(A)	0.3
No.of Cells(Lipo)	2-3S
Max Continuous current(A)180S	15A
Max Continuous Power(W)180S	150W
Max. efficiency current	(7-12A)>78%
internal resistance	126mΩ

Recommended Prop(inch): Standard 2s-1145/1147, 3s-1047/9047; Max thrust

Tested with SunnySky motor 30A ESC

Prop	Volts (V)	Amps (A)	Watts (W)	Thrust (g)	Efficiency (g/W)
9047	8.5	6.5	55	420	7.63
	10	8.2	82	560	6.82
	11.1	9.6	106.5	680	6.38
	12	11.0	132	740	5.60
1047	7.4	7.4	54.7	480	8.77
	8	8.2	65.6	520	7.90
	10	11.2	112	720	6.42
	11.1	13.2	146.5	870	5.93
1145	8.5	12.0	102	680	6.66
	10	14.2	142	880	6.19
	11.1	17.2	190.9	960	5.02

Figura 4.1: Especificaciones motores seleccionados

Seleccionados los motores con especificaciones un poco por encima de los requerimientos antes mencionados como se ve en la figura 4.1, el fabricante del motor recomienda un conjunto de hélices para optimizar el rendimiento del mismo e inclusive brinda las características del motor en función de ensayos realizados con dichas hélices.

Por lo tanto el conjunto propulsor compuesto por motor, hélice y ESC, queda determinado del siguiente modo:

Componente	Modelo - Características
Motor	SunnySky X2212 KV980 II - Empuje(máx): $870gr$@$13{,}2A$
Hélice	Hobbyking Slowfly Propeller 10x4.7
ESC	Afro ESC $30A$

Tabla 4.1: Conjunto propulsor

4.2. Batería

Una vez determinado el conjunto propulsor, fue elegida una batería que proporcionara la tensión requerida, tratando de maximizar el tiempo de vuelo, teniendo en cuenta que no se sobrepase el peso objetivo inicial[1] (1300gr).

Las características de la batería seleccionada son:

Característica	Valor
Marca	Turnigy
Energía	$4000mAh$
Tensión	$11{,}1V$ 3S (3 celdas de 3.7V c/u)
Capacidad de descarga	30C
Composición	Litio y polímero (LiPo)

Tabla 4.2: Características de la batería

En la tabla 7.1 del capítulo 7 puede verse que en el punto de trabajo del propulsor, éste consume aproximadamente $3A$ para un empuje de $310gr$ fuerza, por lo tanto tendremos un consumo aproximado de $12A$ considerando los 4 propulsores para un empuje total de $1240gr$ fuerza.

Con la batería seleccionada de $4000mAh$ se deduce que el cuadricóptero tendrá una autonomía de aproximadamente 15 minutos [2].

4.3. Frame

Una vez seleccionados el resto de componentes mecánicos, se elije un frame capaz de soportar rígidamente los mismos. La principal característica a la hora de elegirla fue la extensión de los brazos, de modo que las hélices tengan espacio suficiente con un amplio margen para colocar la electrónica en el centro.

[1] Recordamos que la energía que es capaz de brindar la batería, está asociada a su masa, por lo tanto mayor autonomía de vuelo requerirá mayor empuje (y a su vez mayor consumo) de los propulsores; a su vez cuanto mayor sea la masa del artefacto, para unos propulsores dados, menor agilidad tendrá el mismo.

[2] Si consideramos un consumo por encima del punto de trabajo, en el peor caso, un consumo promedio de $16A$ considerando el resto de la electrónica se tienen: $15min = \frac{4A,60min}{16A}$

Además el frame junto con la batería son los elementos más pesados del robot, por lo que teniendo en cuenta esta restricción de la separación entre propulsores y el peso deseado, se selecciona el frame de mejor calidad posible adecuándose al presupuesto disponible.

El frame seleccionado tiene las siguientes características:

Característica	**Valor**
Marca/Modelo	Turnigy Talon Quadcopter (V2.0)
Material	Fibra de carbono
Peso	$280gr$
Longitud (diagonal)	$550mm$

Tabla 4.3: Características del frame

Se sustituyen las patas del frame por otras disponibles del primer proyecto uQuad con el fin de elevarlo, de manera de obtener espacio en la parte baja del mismo para la instalación de la batería[3] y sensor de ultrasonido para medidas de alturas pequeñas[4].

Complementando la estructura por defecto del frame, fue necesaria la construcción de piezas adicionales para el montaje de la batería, posicionamiento de la electrónica y sensores, los detalles constructivos de las mismas se describen el en anexo 22.

El peso total final del vehículo es de 1350 g.

[3]Siendo la batería el elemento más pesado del vehículo, su ubicación es fundamental a la hora de definir el centro de masas del artefacto completo, cuanto más abajo se encuentre el centro de masas, mas estable será el vehículo

[4]El sensor de ultrasonido se utiliza en conjunto con el barómetro para la estimación de la altura.

Parte III

Modelo dinámico y Caracterización

Estudio de la dinámica del sistema físico, caracterización de la plataforma y propulsores.

Capítulo 5

Cinemática y Dinámica

5.1. Introducción

En el presente capítulo se estudia la cinemática y dinámica del robot, con el objetivo de conocer la relaciones entre las variables mecánicas que determinan el estado del artefacto, De esta manera se comienza el abordaje para el problema de control de actitud. Como en cualquier problema mecánico, se comienza por definir los sistemas de referencia a utilizar para tratarlo y se especifican las coordenadas a emplear. Con el fin de hacer más clara la exposición, sólo se mencionan los resultados fundamentales en lo referente a la parametrización de rotaciones en el espacio, dejando los desarrollos para el anexo (19).

Se describe el modelo dinámico del cuadricóptero que se ha adoptado, a partir de ciertas simpificaciones sobre el modelo mecánico de partida (2.1). El modelo así construido es empleado junto con la caracterizacion mecánica (6) y de los propulsores (7) para elaborar un modelo en Simulink de Matlab, el cual sirve para el diseño y simulación de los controladores (V).

5.1.1. Cinemática

Las cantidades de interés en el problema son la orientación del cuadricóptero y la posición de su centro de masas en el espacio, ambas magnitudess medidas respecto a un sistema de referencia inercial llamado *absoluto* S_A, que en este caso se elige fijo a la tierra, la cual para las escalas del problema funciona como inercial con buena aproximación. En este referencial de la tierra se usa un sistema cartesiano de coordenadas (figura 5.1) $\{O, X, Y, Z\}$ tal que el eje X apunta hacia el Este, el eje Y hacia el Norte y el eje Z hacia arriba. El otro referencial de interés es el solidario al cuadricóptero, S_B, donde se elige el sistema de ejes $\{G, x, y, z\}$ mostrado en la figura (5.1), donde G es el centro de masas y el eje x apunta en el sentido normal de vuelo.

Una forma estándar de especificar la orientación es mediante los ángulos de Euler, que son los ánglos de las tres rotaciones elementales (esto es, rotaciones según ejes de sistemas cartesianos intermedios), que aplicadas en sucesión permiten transformar los ejes absolutos en los relativos. Existen muchas convenciones distintas para ángulos de Euler, dependiendo de qué ejes se elija para las rotaciones intermedias y el orden de las mismas. En este trabajo se adoptó la *secuencia aeroespacial*, mostrada en la figura (5.2), la cual permite pasar del sistema (X, Y, Z) al (x, y, z) rotando primero en torno al eje Z el ángulo de *yaw* ψ, luego el ángulo de *pitch* θ respecto al eje intermedio y' y finalmente el ángulo de *roll* ϕ en torno al eje x. En

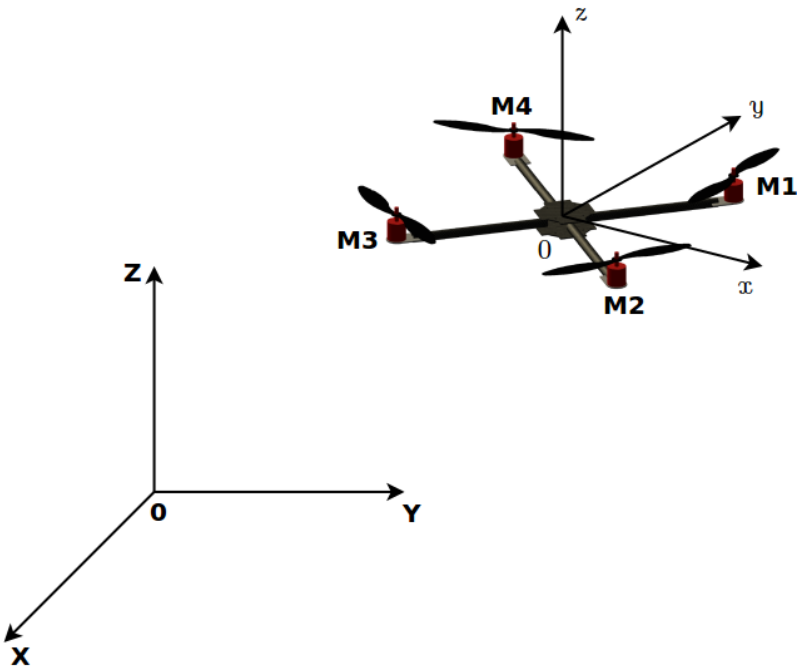


Figura 5.1: Sistemas de coordenadas considerados

esta convención la matriz de cambio de coordenadas del sistema de la tierra (S_A) al del cuadricóptero (S_B) es [25]:

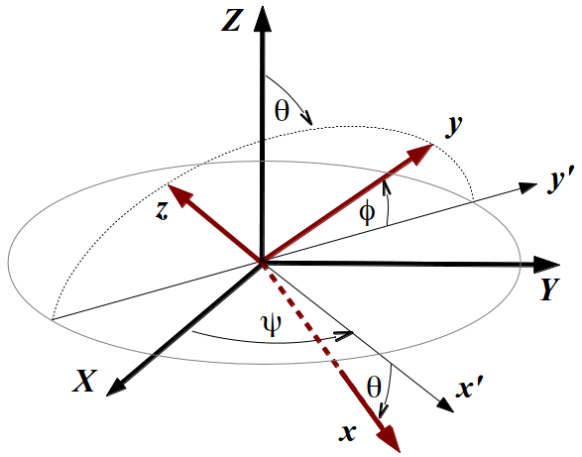


Figura 5.2: Ángulos de Euler en la convención zyx.

$$ {}^{B}_{A}R = \begin{pmatrix} \cos\theta\cos\psi & \operatorname{sen}\psi\cos\theta & -\operatorname{sen}\theta \\ \cos\psi\operatorname{sen}\theta\operatorname{sen}\phi - \operatorname{sen}\psi\cos\phi & \operatorname{sen}\psi\operatorname{sen}\theta\operatorname{sen}\phi + \cos\psi\cos\phi & \cos\theta\operatorname{sen}\phi \\ \cos\psi\operatorname{sen}\theta\cos\phi - \operatorname{sen}\psi\operatorname{sen}\phi & \operatorname{sen}\psi\operatorname{sen}\theta\cos\phi - \cos\psi\operatorname{sen}\phi & \cos\theta\cos\phi \end{pmatrix} \tag{5.1} $$

estando la transformación inversa implementada por su inversa: ${}^{A}_{B}R = {}^{B}_{A}R^{-1} = {}^{A}_{B}R^{T}$. Los ángulos de Euler como parámetros de las rotaciones tienen la ventaja de ser intuitivos e interpretarse inmediatamente como inclinaciones del vehículo según ejes definidos; sin embargo presentan el inconveniente de que parametrizan en forma ambigüa las rotaciones (ver 19). Por otro lado, los algoritmos usuales para estimar la

orientación funcionan en base a parametrizaciones más convenientes, como la matriz de cosenos directores (o DCM, *Direction Cosines Matrix*) [18] y cuaterniones [36], siendo este último el utilizado en este trabajo. Originalmente introducidos como una extensión de los números complejos, los cuaterniones resultan muy adecuados para parametrizar las rotaciones en el espacio, debido a que no tienen singularidades y los cálculos con ellos son computacionalmente menos costosos que con los ángulos de Euler o la matriz de cosenos directores.

Dada la rotación que transforma el sistema (X, Y, Z) en el (x, y, z) por su eje de giro $\hat{n} = (n_1, n_2, n_3)$ y ángulo (antihorario) α, el cuaternión q asociado a esta rotación se define como:

$$q_0 = \cos\left(\frac{\alpha}{2}\right) \tag{5.2}$$

$$q_1 = n_1 \operatorname{sen}\left(\frac{\alpha}{2}\right) \tag{5.3}$$

$$q_2 = n_2 \operatorname{sen}\left(\frac{\alpha}{2}\right) \tag{5.4}$$

$$q_3 = n_3 \operatorname{sen}\left(\frac{\alpha}{2}\right) \tag{5.5}$$

Donde se puede comprobar fácilmente que $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$ (condición de normalización). Esta correspondencia permite efectuar las operaciones de cambio de coordenadas de un vector utilizando el álgebra de cuaterniones. Si la rotación que lleva al sistema absoluto S_A a coincidir con el solidaro al cuadricóptero S_B tiene eje y ángulo de rotación $(\hat{n}, \alpha)$, el cuaternión (normalizado) asociado a esta rotación según (19.5) se denota por ${}^B_A q$ y puede demostrarse que la transformación de componentes de un vector $\vec{v}$ se obtiene de acuerdo al producto (19.7),

$${}^B\vec{v} = {}^B_A q^* \otimes {}^A\vec{v} \otimes {}^B_A q \tag{5.6}$$

donde $\otimes$ indica el *producto de cuaterniones*, (ver 19). La matriz asociada a esta transformación lineal de cambio de base ${}^B_A R$, parametrizada con las componentes de ${}^B_A q$, se escribe:

$${}^B_A R = \begin{pmatrix} 2(q_0^2 + q_1^2) - 1 & 2(q_1 q_2 + q_0 q_3) & 2(q_1 q_3 - q_0 q_2) \\ 2(q_1 q_2 - q_0 q_3) & 2(q_0^2 + q_2^2) - 1 & 2(q_2 q_3 + q_0 q_1) \\ 2(q_3 q_1 + q_0 q_2) & 2(q_3 q_2 - q_0 q_1) & 2(q_0^2 + q_3^2) - 1 \end{pmatrix} \tag{5.7}$$

Si se parametriza la misma rotación mediante ángulos de Euler, la relación con ${}^B_A q$ es:

$$\psi = \operatorname{atan2}\left(2(q_1 q_2 + q_0 q_3), q_0^2 + q_1^2 - q_2^2 - q_3^2\right) \tag{5.8}$$

$$\theta = \operatorname{asin}\left(2q_0 q_2 - 2q_1 q_3\right) \tag{5.9}$$

$$\phi = \operatorname{atan2}\left(2(q_2 q_3 + q_0 q_1), q_3^2 - q_2^2 - q_1^2 + q_0^2\right) \tag{5.10}$$

La relación de las derivadas de los parámetros con la velocidad angular viene dada por las ecuaciones (19.12) para ánglos de Euler, mientras en cuaterniones adquieren la forma más compacta (19.13).

$$\begin{aligned}
\dot{\psi} &= \omega_2 \frac{\sin\phi}{\cos\theta} + \omega_3 \frac{\cos\phi}{\cos\theta} \\
\dot{\theta} &= \omega_2 \cos\phi - \omega_3 \sin\phi \\
\dot{\phi} &= \omega_1 + \omega_2 \frac{\sin^2\phi}{\cos\theta} + \omega_3 \frac{\sin\phi\cos\phi}{\cos\theta}
\end{aligned} \tag{5.11}$$

$$\dot{q} = \frac{1}{2} q \otimes \vec{\omega} \tag{5.12}$$

5.2. Estado

El estado del cuadricóptero como sistema mecánico puede ser separado en una parte translacional y un parte angular o rotacional. El estado translacional viene especificado por la posición y velocidad en el sistema absoluto S_A (Tierra):

$$(X, Y, Z, v_X, v_Y, v_Z)^T \tag{5.13}$$

Para el estado rotacional se debe especificar la orientación y la tasa de cambio de la misma. Una manera de hacer esto es dar los ángulos de Euler y las componentes de la velocidad angular $\vec{\omega}$ en el sistema solidaro al cuadricóptero S_B (que son las que mide el giróscopo):

$$(\psi, \theta, \phi, \omega_1, \omega_2, \omega_3)^T \tag{5.14}$$

la relación entre estas componentes de $\vec{\omega}$ y las derivadas $(\dot{\psi}, \dot{\theta}, \dot{\phi})$ está determinada por 19.12.

En la parametrización de cuaterniones es:

$$(q, \dot{q})^T \tag{5.15}$$

Se debe advertir que, si bien un cuatenión q tiene en principio cuatro componentes, en los cuaterniones que parametrizan rotaciones sólo tres de esas componentes son independientes. Esto impone también un vínculo en la derivada $\dot{q}$. De modo que el estado dado por (5.15) indica efectivamente tres grados de libertad.

5.3. Ecuaciones dinámicas

5.3.1. Primera Cardinal

La parte traslacional de la dinámica corresponde a la traslación del centro de masas y está dada por la ecuación de Newton o primera cardinal en la terminología usada a veces para los cuerpos rígidos:

$$m\vec{a}_G = \vec{F}_1 + \vec{F}_2 + \vec{F}_3 + \vec{F}_4 + m\vec{g} + \vec{F}_d = (F_1 + F_2 + F_3 + F_4)\hat{k} + m\vec{g} + \vec{F}_d \quad (5.16)$$

siendo $\vec{F}_i$ las fuerzas de los propulsores (figura 5.3) y $\vec{F}_d$ la resistencia del aire, que para las velocidades implicadas se puede modelar por:

$$\vec{F}_d = -b\vec{v} \quad (5.17)$$

donde $\vec{v}$ es la velocidad respecto al sistema de referencia del aire, que en caso de ser despreciada la velocidad del viento puede suponerse igual a la velocidad en el sistema de la Tierra S_A.

5.3.2. Segunda Cardinal

Como se menciona en el punto 5.2 las variables de estado consideradas son:

$(yaw, pitch, roll) = (\psi, \theta, \phi)$	Ángulos de Euler.
$\omega = (\omega_1, \omega_2, \omega_3)$	Componentes de la velocidad angular del cuadricóptero en el sistema no inercial.

Planteando el momento angular en el centro de masa G del robot, se tiene:

$$\vec{L_G} = \vec{L_G^q} + \vec{L_G^r} \quad (5.18)$$

Donde:

$\vec{L_G^q}$	Momento angular del cuadricóptero.
$\vec{L_G^r}$	Momento angular de los rotores.

Se plantea el cálculo de cada uno de ellos expresados en el sistema solidario al robot y obviando el subíndice G en los momentos angulares.

El momento angular del cuadricóptero:

$$\vec{L^q} = I_G^q\vec{\omega} = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \quad (5.19)$$

Donde (I_x, I_y, I_z) son los momentos de inercia del robot respecto a los ejes solidarios al mismo y $\vec{\omega} = \omega_1\vec{i} + \omega_2\vec{j} + \omega_3\vec{k}$.

El momento angular total en los rotores es la suma de cada uno:

$$\vec{L^r} = \vec{L^r}_1 + \vec{L^r}_2 + \vec{L^r}_3 + \vec{L^r}_4 \quad (5.20)$$

Se plantea para el rotor 1:

$$\vec{L^r}_1 = I^r_{B1}(\Omega_1 \vec{k} + \vec{\omega}) + m_1 \overrightarrow{(G - B_1)} \times v\vec{}_{B1} \tag{5.21}$$

Donde:

I^r_{B1}	Momento angular en la base del rotor 1.
Ω_1	Velocidad angular del rotor 1.
$v\vec{}_{B1}$	Velocidad de la base del rotor 1.

Teniendo en cuenta que $\Omega_1 >> |\vec{\omega}|$ y que el segundo sumando es también despreciable frente a Ω_1, se tiene para la suma de todos los rotores que:

$$\vec{L^r} \cong I_r(-\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4)\vec{k} = I_r \Omega_r \vec{k} \tag{5.22}$$

En la figura 5.3 se muestra los sentidos para las Ω_i de donde surgen los signos en la ecuación anterior.

Derivamos el momento angular total del cuadricóptero, para plantear la segunda cardinal:

$$\dot{\vec{L}} = \dot{\vec{L}}^q + \dot{\vec{L}}^r \cong I^q_G \dot{\vec{\omega}} + \vec{\omega} \times I_G \vec{\omega} + \vec{\omega} \times \vec{L^r} = \vec{M^{ext}_G} = \vec{M} \tag{5.23}$$

En la ecuación anterior se desprecia la derivada de $\vec{L^r}$ en el sistema no inercial, o sea $\frac{d' \vec{L^r}}{dt} \cong 0$.

Las ecuaciones de la derivada del momento angular, en función de las componentes de $\vec{\omega}$ en el sistema solidario al robot (x, y, z):

$$\dot{\vec{L}} = \begin{cases} I_x \dot{\omega}_1 + (I_z - I_y)\omega_2 \omega_3 + I_r \Omega_r \omega_2 \\ I_y \dot{\omega}_2 + (I_x - I_z)\omega_1 \omega_3 - I_r \Omega_r \omega_1 \\ I_z \dot{\omega}_3 + (I_y - I_x)\omega_1 \omega_2 \end{cases} \tag{5.24}$$

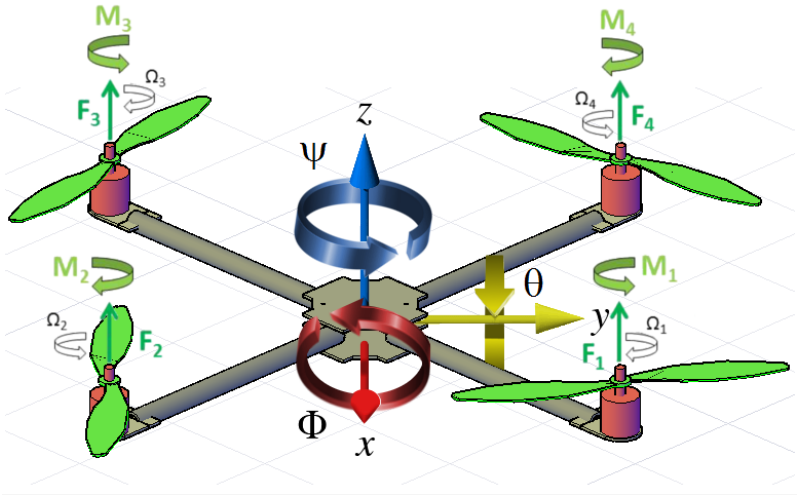


Figura 5.3: Fuerzas y momentos de cada motor

En la figura 5.3 se muestran entre otras cosas las fuerzas de empuje y momentos que cada motor ejerce sobre el artefacto, según las convenciones que se eligieron en

el proyecto, como perfil de vuelo y sistema de coordenadas. La expresión para los momentos en el sistema del robot (x, y, z):

$$\begin{cases} M_x = h_1F_1 - h_2F_2 - h_3F_3 + h_4F_4 \\ M_y = -l_1F_1 - l_2F_2 + l_3F_3 + l_4F_4 \\ M_z = M_1 - M_2 + M_3 - M_4 \end{cases} \tag{5.25}$$

h_i y l_i surgen de considerar el producto vectorial de los radios a G con la fuerza empuje, por lo tanto, son los semi-lados del cuadrado que forman los 4 motores, por la simetría del artefacto. Por tanto $h_i = l_i = l$.

Tomando en cuenta la última consideración:

$$\begin{cases} M_x = l(F_1 - F_2 - F_3 + F_4) \\ M_y = l(F_1 - F_2 + F_3 + F_4) \\ M_z = M_1 - M_2 + M_3 - M_4 \end{cases} \tag{5.26}$$

Considerando entonces los momentos, las ecuaciones para la segunda cardinal resultan:

$$\begin{cases} I_x\dot{\omega}_1 = (I_y - I_z)\omega_2\omega_3 - I_r\Omega_r\omega_2 + M_x \\ I_y\dot{\omega}_2 = (I_z - I_x)\omega_1\omega_3 + I_r\Omega_r\omega_1 + M_y \\ I_z\dot{\omega}_3 = (I_x - I_y)\omega_1\omega_2 + M_z \end{cases} \tag{5.27}$$

La consideración final para la segunda cardinal es que $\Omega_r \cong 0$ ya que se infiere de la ecuación 5.22 cuando las velocidades angulares de los motores son iguales o parecidas, cosa que sucede en estos artefactos en mayor parte del vuelo. Esto último hace a los sumandos con factor Ω_r en la segunda cardinal, despreciables frente a los otros. Entonces:

$$\begin{cases} I_x\dot{\omega}_1 = (I_y - I_z)\omega_2\omega_3 + M_x \\ I_y\dot{\omega}_2 = (I_z - I_x)\omega_1\omega_3 + M_y \\ I_z\dot{\omega}_3 = (I_x - I_y)\omega_1\omega_2 + M_z \end{cases} \tag{5.28}$$

5.4. Consideraciones finales

Las ecuaciones 5.28 y 19.12 son las que interesa resolver para las variables de estado de la actitud del cuadricóptero. En este caso las entradas son las componenetes del momento $\vec{M}$. La traslación viene dada por la primera cardinal 5.16 y las variables de estado son 5.13, en este caso la única entrada es la fuerza neta según $\hat{k}$ en el sistema solidario S_B dada por:

$$F_{neta} = F_1 + F_2 + F_3 + F_4 \tag{5.29}$$

Las componentes del momento $\vec{M}$ vienen dadas por 5.25. Obsérvese que en el problema de control se tiene acceso a modificar las cuatro velocidades angulares de los

motores $\Omega_1, \Omega_2, \Omega_3, \Omega_4$, lo que determina F_1, F_2, F_3, F_4 y M_1, M_2, M_3, M_4 simultáneamente. Por otro lado, para lograr un estádo dinámico arbitrario es necesario poder aplicar una fuerza neta $\vec{F}_{neta}$ y un momento $\vec{M}$ arbitrarios, o sea, seis componentes arbitrarias. Esto no es posible con las cuatro velocidades angulares disponibles de los motores, por lo que no será posible lograr cualquier estado dinámico.

En los capítulos siguientes, se exponen los experimentos necesarios para la resolución, como la determinación de los momentos de inercia (I_x, I_y, I_z) y la caracterización de propulsores, que vincula empuje y torque con la velocidad de los motores. También se exponen las simulaciones de este modelo dinámico realizadas en Matlab2014 (herramienta Simulink).

Capítulo 6

Caracterización Mecánica

En el presente capítulo se exponen los procesos realizados para determinar los parámetros inherentes a la mecánica del artefacto (momentos de inercia, masa total). Se determinaron los momentos de inercia utilizando el modelo del péndulo físico en pequeñas oscilaciones y se determinó la masa del robot armado usando una balanza.

6.1. Momentos de inercia I_x, I_y, I_z:

6.1.1. Descripción del proceso:

Para determinar los momentos de inercia respecto de cada eje del sistema solidario al robot (x, y, z), se utiliza la técnica del péndulo físico. Se hizo oscilar al artefacto en tres posiciones distintas, de manera que en cada caso, el eje de oscilación sea paralelo al eje del momento de inercia que interesa determinar.

Para cada configuración de péndulo:

Se sujeta con la tanza al cuadricóptero por 2 de sus extremos y se lo hace oscilar mediante un eje sin fricción. Con el cronómetro se mide el tiempo total que permanece oscilando, se lo deja oscilar 15 períodos. El período de oscilación será el tiempo total medido dividido la cantidad de oscilaciones (15).

Se repite el proceso 5 veces. Se determinó el período de oscilación de cada péndulo, promediando las 5 mediciones. Con el período del péndulo y la distancia desde el eje de oscilación al centro de masa G, se obtuvo el momento de inercia respecto a cada eje.

6.1.2. Elementos utilizados:

- Cuadricóptero armado.
- Tanza.
- Cronómetro (celular).
- Eje cilíndrico.
- Cinta métrica

6.2. Esquemas mecánicos

En las siguientes figuras se muestran las configuraciones de péndulo para determinar I_x e I_y:

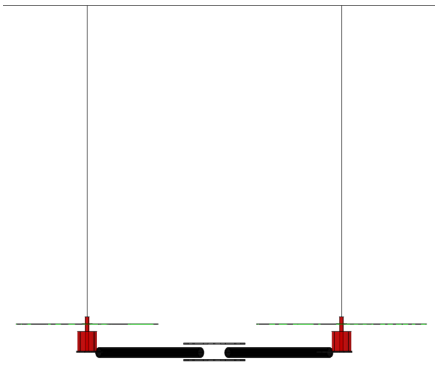


Figura 6.1: x-y: vista frontal

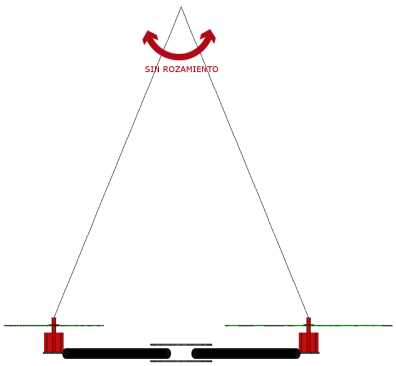


Figura 6.2: x-y: vista lateral

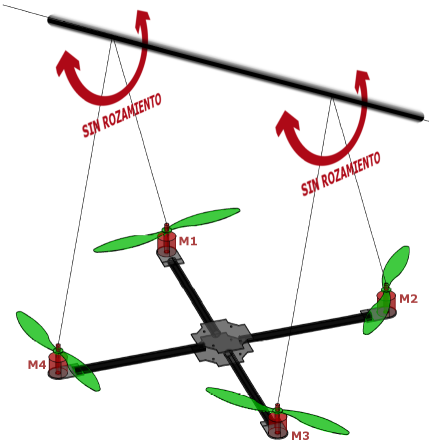


Figura 6.3: Momento según y

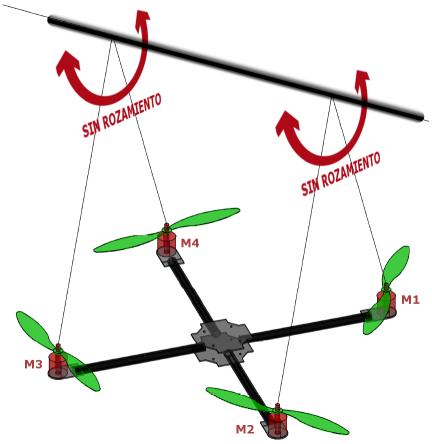
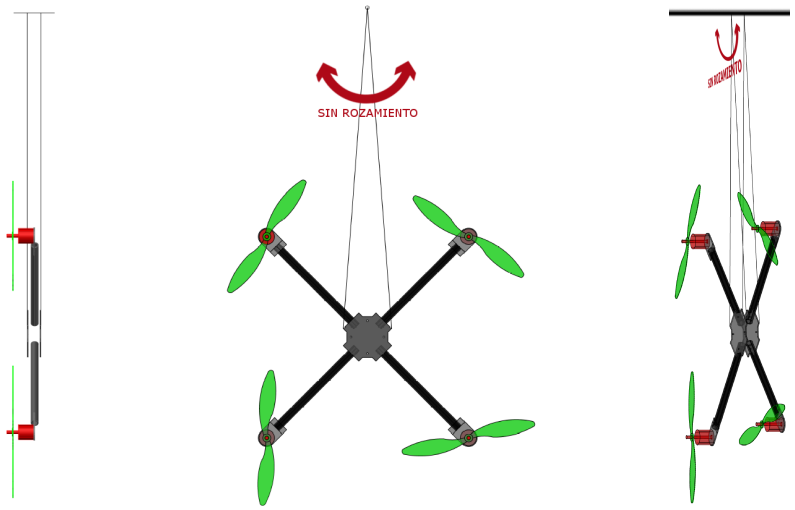


Figura 6.4: Momento según x

Configuración de péndulo para determinar I_z:



6.3. Mediciones y procesamiento de datos

6.3.1. Período

Los datos registrados para el período se muestran en los siguientes cuadros:

I_x		
Nº Osc.	$t_i(s)$	$T_i(s)$
15	23,86	1,591
15	23,78	1,585
15	23,83	1,589
15	23,64	1,576
15	23,54	1,569

Tabla 6.1: Datos registrados (I_x)

I_y		
Nº Osc.	$t_i(s)$	$T_i(s)$
15	22,46	1,497
15	22,29	1,486
15	22,42	1,495
15	22,24	1,483
15	22,34	1,489

Tabla 6.2: Datos registrados (I_y)

I_z		
Nº Osc.	$t_i(s)$	$T_i(s)$
15	26,29	1,753
15	26,49	1,766
15	26,44	1,763
15	26,04	1,736
15	26,38	1,759

Tabla 6.3: Datos registrados (I_z)

Siendo:

- $t_i(s)$ el tiempo total (considerando las 15 oscilaciones)
- $T_i = \frac{t}{Osc}$ el período de cada experimento

Promediando los 5 períodos en cada caso, finalmente:

$$\boxed{T_x = 1{,}582s} \quad (6.1) \qquad \boxed{T_y = 1{,}49s} \quad (6.2) \qquad \boxed{T_z = 1{,}755s} \quad (6.3)$$

6.3.2. Distancia desde eje a G

Se determinó para cada caso la distancia l_G desde el eje de oscilación al centro de masa G del cuadricóptero.

En primer lugar se ubicó el centro de masa del artefacto mediante intersección de verticales. El proceso consistió en colgar el cuadricóptero como se muestra en la figura 6.5. Por la simetría del robot, únicamente fue necesario determinar la distancia dH de la figura 6.7.

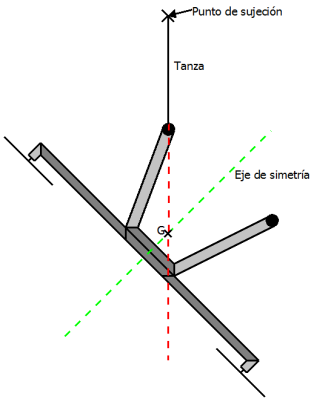


Figura 6.5: Método determinación de G

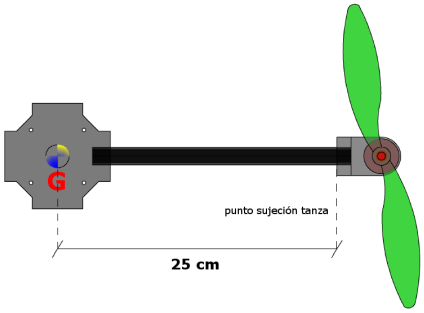


Figura 6.6: Vista superior brazo

Figura 6.7: Vista lateral

En el experimento de I_z se midió directamente l_G mientras que para I_x e I_y por practicidad se tomó la medida de la tanza, en estos casos se llega a l_G de acuerdo a la figura 6.8

$$l_G = \sqrt{L_{tanza}^2 - L_{brazo}^2} + d_H$$

Siendo:

$d_H = 0{,}02m$

$L_{brazo} = 0{,}25m$

Por lo tanto:

$L_{tanzax} = 0{,}610m \Rightarrow l_{Gx} = 0{,}576m$

$L_{tanzay} = 0{,}530m \Rightarrow l_{Gy} = 0{,}487m$

$l_{Gz} = 0{,}730m$

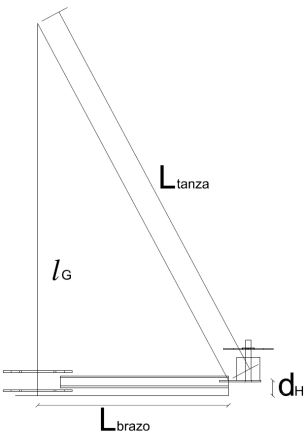


Figura 6.8: calculo l_G

6.4. Resultados

6.4.1. Masa

La masa del artefacto completo es: $m = 1{,}232kg$

6.4.2. Momentos

Partiendo de la ecuación del péndulo físico en pequeñas oscilaciones y el teorema de Steiner tenemos que:

$$\left.\begin{aligned} T &= 2\pi\sqrt{\frac{I_0}{mgl_G}} \\ I_0 &= I_G + ml_G^2 \end{aligned}\right\} \Rightarrow I_G = ml_G\left[g\left(\frac{T}{2\pi}\right)^2 - l_G\right] \tag{6.4}$$

Donde I_0 es el momento de inercia respecto al eje de oscilación.
Finalmente se tiene:

$$\boxed{I_{Gx} = 0{,}032kgm^2} \quad (6.5) \qquad \boxed{I_{Gy} = 0{,}038kgm^2} \quad (6.6) \qquad \boxed{I_{Gz} = 0{,}031kgm^2} \quad (6.7)$$

Capítulo 7

Caracterización de Propulsores

7.1. Introducción

En el presente capítulo se exponen los experimentos de caracterización realizados para los propulsores del cuadricóptero. Se denomina propulsor al conjunto de motor, controlador ESC y hélice utilizada. El objetivo de estos experimentos es caracterizar la fuerza de empuje y el torque que el propulsor realizará sobre el artefacto, en función de la señal PWM enviada al ESC y por lo tanto en función también de la velocidad angular del motor. Otro aspecto perseguido en estos experimentos fue conocer el consumo de los propulsores, información que permite el cálculo aproximado de tiempo de autonomía de vuelo.

7.2. Medida de Empuje

7.2.1. Experimento

Elementos utilizados:

- Motor marca "SunnySky" modelo "X2212 KV:980".
- Hélice Gemfan 947 y 1047.
- ESC marca "Afroesc" modelo 30 A.
- Batería marca "Turnigy" 2200mAh.
- Generador de señales marca "Tektronix" modelo "CFG250".
- Fuente de 5V.
- Osciloscopio marca "Tektronix" modelo "TBS 1062".
- Conjunto fotodiodo (LED IR TSAL6200 y Detector IR TSOP38256).
- Multímetro Extech MN35.
- Balanza electrónica Camry EK9320

Descripción:

En la figura 7.1 se muestra la configuración mecánica del experimento. El motor se colocó fijo a la barra y encendido de manera que empuje la barra hacia abajo. Se muestran en la figura las fuerzas sobre la barra.

En la figura 7.2 se muestra el diagrama eléctrico del experimento. Un sketch de Arduino se encargó de enviar la señal PWM al ESC y mostrar en pantalla el tiempo de pulso en alto de la misma. La velocidad angular del los motores se ajustó variando el ancho de los pulsos de la señal PWM enviada a los controladores. Se varió entonces el ancho de pulso de esta señal entre $1000\mu s$ y $2000\mu s$ con pasos de $100\mu s$. El rango $1000\mu s - 2000\mu s$ corresponde a la calibración que se aplicó a los ESCs, cuyo procedimiento se describe en (20).

La fuerza de empuje que realiza el motor se registró según la lectura de la balanza[1] y se midió en el osciloscopio la frecuencia de la señal cuadrada emitida por el receptor IR. Ya que la hélice pasa dos veces por período, dividiendo la frecuencia medida por 2 se registró la frecuencia correspondiente a la velocidad del motor. La señal eléctrica del generador de señal al LED utilizada es una onda cuadrada de $56kHz$ y $5V$ pp.

Se registró el consumo conectando el multímetro en serie a la batería.

Esquemas de conexión

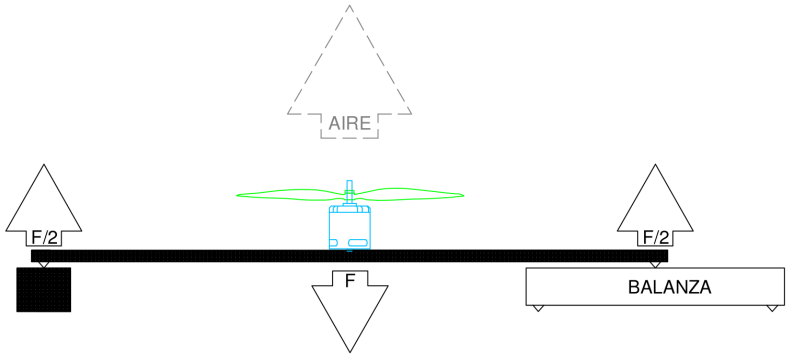


Figura 7.1: Esquema mecánico medida de empuje

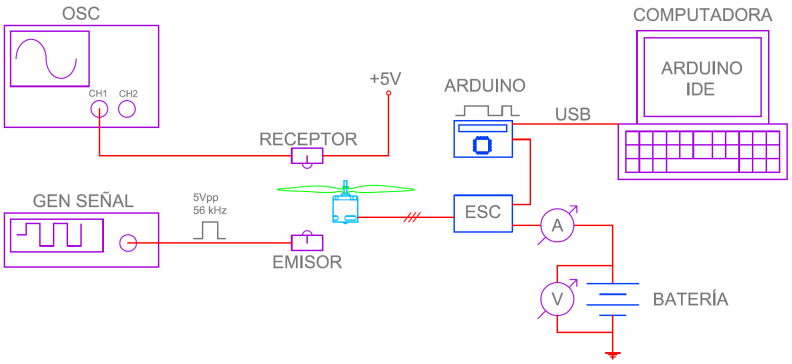


Figura 7.2: Esquema eléctrico del experimento

[1]La lectura en la balanza, tal como se indica en el esquema de conexión mecánico 7.1, equivale a la mitad de la fuerza total que ejerce el motor. No se considera el peso de los componentes ya que la balanza se tara con el peso muerto de los dispositivos previo a encender el motor.

7.2.2. Resultados y Procesamiento de datos

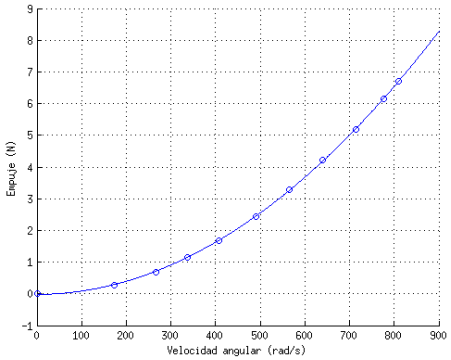
Se muestran los resultados para las hélices 947 que son las que se van a utilizar.

Tabla 7.1: Datos registrados correspondientes a la hélice 947

Ancho pulso (μs)	***Consumo I (A)***	***Voltaje (V)***	***Balanza (g)***	***Frec (Hz)***
1100	0,3	11,5	14	27,4
1200	0,6	11,48	35	42,25
1300	1	11,46	58	53,65
1400	1,6	11,41	86	64,75
1500	2,3	11,35	125	78,15
1600	3,5	11,28	168	90
1700	5	11,18	215	101,75
1800	6,8	11,06	265	113,765
1900	8,7	10,94	314	123,75
2000	9,9	10,87	342	128,85

En la figura 7.3 se muestran los puntos experimentales. El empuje en función de la velocidad angular del motor Ω. Se realiza un ajuste cuadrático de los datos.

Figura 7.3: Empuje vs velocidad angular



A los datos experimentales se les agregó el primer punto correspondiente a $(\Omega = 0, Empuje = 0)$. El ajuste es cuadrático sin forzar por 0.Los coeficientes de la ecuación $a\Omega^2 + b\Omega + c$ son, para este caso:

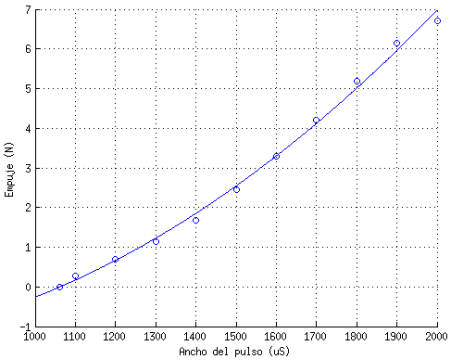
$$\begin{cases} a = 1{,}023 \times 10^{-5} \frac{Ns^2}{rad^2} \\ b = 1{,}373 \times 10^{-5} \frac{Ns}{rad} \\ c = -0{,}01757N \end{cases} \tag{7.1}$$

Se procede del mismo modo para ajustar el Empuje en función del ancho de pulso n, ajuste $an^2 + bn + c$. Se agrega el punto $(n = 1060\mu s, Empuje = 0)$.

Figura 7.4: Empuje vs ancho de pulso

Los coeficientes en este caso:

$$\begin{cases} a = 3{,}267 \times 10^{-6} \frac{N}{(\mu s)^2} \\ b = -0{,}002565 \frac{N}{\mu s} \\ c = -0{,}9608N \end{cases} \tag{7.2}$$



7.3. Medida de Torque:

7.3.1. Experimento:

Elementos utilizados:

- Motor marca “SunnySky” modelo “X2212 KV:980”.
- Hélice Gemfan 947 y 1047.
- ESC marca “Afroesc” modelo 30 A.
- Batería marca “Turnigy” 2200mAh.
- Generador de señales marca “Tektronix” modelo “CFG250”.
- Fuente de 5V.
- Osciloscopio marca “Tektronix” modelo “TBS 1062”.
- Conjunto fotodiodo (LED IR TSAL6200 y Detector IR TSOP38256).
- Multímetro Extech MN35.
- Balanza electrónica Camry EK9320

Descripción:

El motor se monta fijo en una barra articulada sin fricción en un extremo y apoyada en una balanza en el otro extremo. El esquema mecánico del experimento se muestra en la figura 7.5. La balanza mide la fuerza del torque que el motor ejerce sobre la barra. El esquema eléctrico del experimento es el mismo que el del empuje, por tanto valen las mismas consideraciones para este experimento.

Esquemas de conexión

El esquema del experimento se muestra en la figura 7.5.

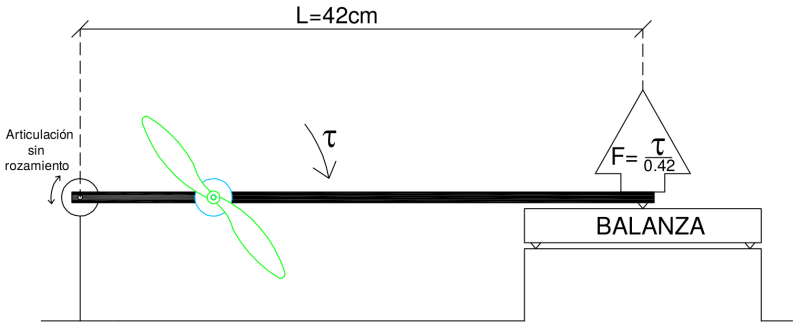


Figura 7.5: Esquema mecánico del experimento

Diagrama eléctrico idéntico al de la medida del torque, ver figura 7.2

7.3.2. Resultados y Procesamiento de datos

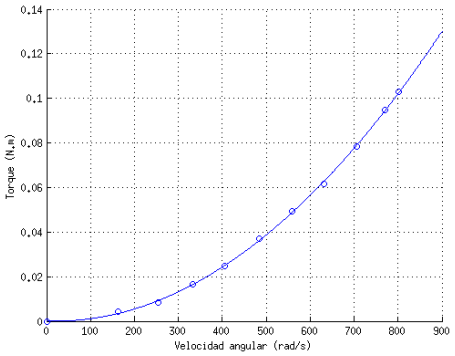
Tabla 7.2: Datos para hélice 947

Ancho pulso (μs)	***Consumo I (A)***	***Voltaje (V)***	***Balanza (g)***	***Frec (Hz)***
1100	0,3	11,42	1	25,75
1200	0,6	11,4	2	40,52
1300	1	11,38	4	52,85
1400	1,5	11,25	6	64,4
1500	2,5	11,19	9	77
1600	3,55	11,12	12	89
1700	5	11,04	15	100,5
1800	7	10,95	19	112,5
1900	9	10,82	23	122,5
2000	10	10,75	25	127,5

En la figura 7.6 se muestran los datos experimentales. No se consideran para el ajuste los dos primeros datos del experimento, por entenderse que el registro 0 en la balanza no es precisamente real con el motor funcionando.
A los datos experimentales se les agregó el primer punto correspondiente a ($\Omega = 0, Torque = 0$). El ajuste es cuadrático sin forzar por 0.Los coeficientes de la ecuación $a\Omega^2 + b\Omega + c$ son, para este caso:

$$\begin{cases} a = 1{,}682 \times 10^{-7} \frac{Nms^2}{rad^2} \\ b = -7{,}075 \times 10^{-6} \frac{Nms}{rad} \\ c = 0{,}0001449 Nm \end{cases} \tag{7.3}$$

Figura 7.6: Torque vs velocidad angular



Se procede del mismo modo para ajustar el Torque en función del ancho de pulso n, ajuste $an^2 + bn + c$. Se agrega el punto ($n = 1060\mu s, Torque = 0$).

Los coeficientes en este caso:

$$\begin{cases} a = 5{,}413 \times 10^{-8} \frac{Nm}{(\mu s)^2} \\ b = -5{,}054 \times 10^{-5} \frac{Nm}{\mu s} \\ c = -0{,}008506 Nm \end{cases} \tag{7.4}$$

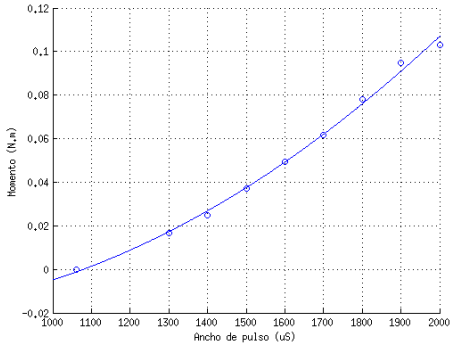


Figura 7.7: Torque vs ancho de pulso

7.4. Consumo y relación entre Ω y n:

7.4.1. Consumo

Para comparar la performance en eficiencia de consumo entre las hélices adquiridas, $947y1047$, a continuación se muestra en las figuras 7.8 y 7.9 el empuje y torque para cada hélice en función de la potencia consumida.

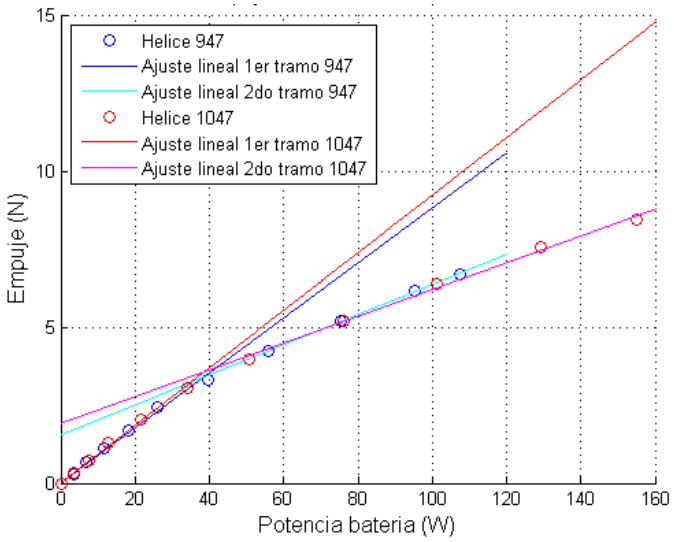


Figura 7.8: Empuje vs potencia eléctrica

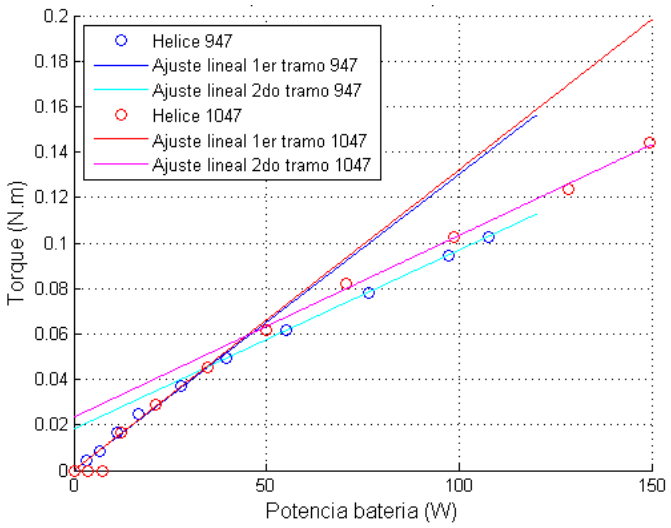


Figura 7.9: Torque vs potencia eléctrica

En ambos casos, los puntos experimentales parecen pertenecer a las mismas rectas, por lo que las hélices parecen presentar la misma eficiencia en consumo, esto es algo que el fabricante especificaba y se verificó con los experimentos realizados.

Alrededor de $40W$ para el empuje y $50W$ para el torque hay una caída en el rendimiento del propulsor, ya que vemos ajustes de datos con pendientes más bajas. En este sentido, mencionamos que la zona de funcionamiento del cuadricóptero ($empujetotal = peso$, de modo de permanecer en el aire), está en el entorno de los $35W$ de consumo con hélice 947.

7.4.2. Velocidad angular y ancho de pulso

En la figura 7.10 se muestra una dependencia lineal entre el ancho de pulso n de la señal PWM enviada al ESC y la velocidad angular del motor Ω.

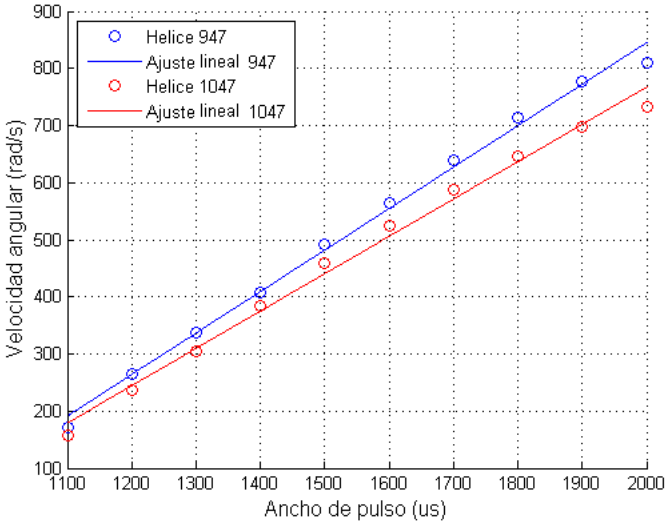


Figura 7.10: Velocidad angular vs pulso

Capítulo 8

Simulaciones

8.1. Modelo en Simulink

Con el robot caracterizado mecánicamente (masa total y momentos de inercia) y la caracterización de los propulsores (empuje y torque en función de la velocidad del motor), se utilizó la herramienta *Simulink* de *Matlab*2014 para poder realizar simulaciones de la dinámica de cuadricóptero.

En la figura se puede ver el diagrama implementado en Simulink.

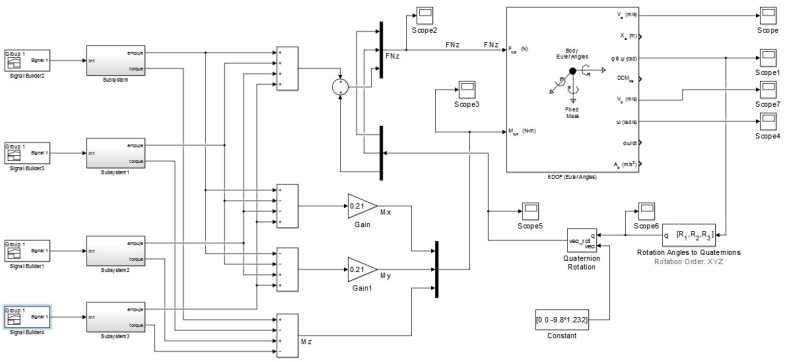


Figura 8.1: Diagrama Simulink para simulaciones

8.1.1. Descripción

Se describe a continuación la figura 8.1:

El bloque $6DOF(Euler\,Angles)$ es el bloque principal, representa la dinámica de la planta, del cuadricóptero. Sus entradas son los momentos aplicados al cuadricóptero por los propulsores según x, y, z y la fuerza neta en los ejes absolutos XYZ. El bloque está configurado con los valores de masa total y momentos de inercia determinados para el robot.

Mediante los tres bloques sumadores de la parte inferior de la figura 8.1 y los bloques de ganancia en caso de ejes x, y, se generan los tres momentos totales según cada eje, acorde a la ecuación 5.26. El bloque de ganancia en el caso de x, y es el parámetro l de dicha ecuación, el cual se midió y vale $0{,}21m$.

Las componentes en los ejes absolutos XYZ de la fuerza neta, se calculan a partir de los ángulos de Euler del cuadricóptero (inclinación) mediante una transformación de los mismos a cuaterniones (apéndice 19.1). En el caso del eje Z, se resta la fuerza peso.

Las salidas de interés del bloque $6DOF(Euler\ Angles)$ son w_1, w_2, w_3, velocidades angulares en cada eje x, y, z respectivamente del sistema del cuadricóptero y los ángulos de Euler.

Los cuatro bloques en la izquierda de la figura cuyas salidas son empuje y torque, representan los propulsores y por lo tanto tienen configurada la caracterización de los mismos, esto es, el empuje y torque se calculan con la forma cuadrática encontrada mediante los parámetros a, b y c determinados en la caracterización. Su entrada será entonces la velocidad del motor n, o sea el ancho de pulso de la señal PWM enviada al ESC.

En capítulos posteriores cuando se ensayan los controladores de actitud, se utiliza todo el bloque de la figura 8.1 como un único bloque (planta) al cual entran las señales n_i PWM de comando a los motores.

8.2. Simulaciones y resultados

Para probar el funcionamiento de los bloques implementados en *Simulink* a continuación se muestra la simulación de la siguiente situación:

- Motores 1 y 2 encendidos a velocidad constante durante $5s$ y luego apagados.
- Motores 3 y 4 apagados.

En la figura 8.2 se ve la evolución temporal de los ángulos de Euler (*Roll* amarillo, *Pitch* en magenta y Yaw en cian. Como es de esperarse *Roll* y Yaw permanecen en 0 y el ángulo *Pitch* va hacia los valores negativos, de forma parabólica hasta los $5s$ (velocidad angular aumenta linealmente) y con pendiente constante luego de los $5s$, donde la velocidad angular es constante, ya que el modelo dinámico en *Simulink* no contempla el rozamiento con el aire, por lo tanto el cuadricóptero queda girando con velocidad angular w_y constante.

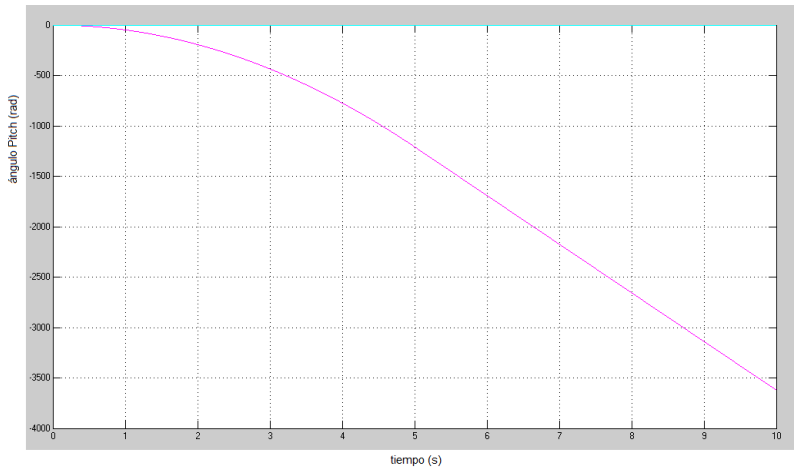


Figura 8.2: Simulación de Simulink para Pitch

La figura 8.3 muestra la evolución temporal de w_y, llamada también w_2.

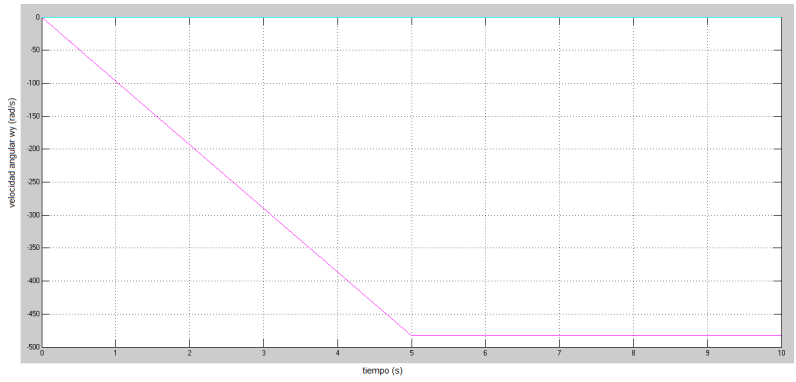


Figura 8.3: Simulación velocidad angular w_y

Parte IV

Estimación del estado

Descripción y caracterización del poder sensorial del cuadricóptero e implementación de la fusión sensorial.

8.2.1. Poder sensorial elegido

Sistema Traslacional

En este caso los tres sensores elegidos y adquiridos para obtener las variables X, Y, Z son GPS, Barómetro y Ultrasonido.

El GPS es un sensor de posicionamiento global, que es capaz de dar la posición en latitud y longitud y una estimación de la altura. Dados los antecedentes de proyectos anteriores, investigación e información del fabricante se conoce que la estimación de la altura del GPS no es buena. Por esto es que se opta por descartarla y tener un barómetro y un sensor de ultrasonido para la estimación o medida de altura.

Por tanto, el GPS será el sensor que dará la estimación de las coordenadas X, Y de artefacto, mediante una conversión de las medidas de latitud y longitud que otorga. Se toma entonces como punto de origen $(0, 0)$ el punto donde se encuentra el cuadricóptero al ser encendido. Para ello se toman al encendido 1500 mediciones de X, Y las cuales se promedian para luego obtener la estimación respecto del origen mediante la sustracción del promedio a la medida.

El sensor de ultrasonido es adquirido por su gran precisión y bajo costo, para la estimación de la altura en los casos de vuelos bajos (menor a $4m$ aproximadamente de altitud), aterrizaje y despegue del cuadricóptero. Para la estimación de altitudes mayores a $4m$, es decir a falta de la medida del ultrasonido, el Barómetro es el sensor elegido. Es un sensor de presión absoluta mediante el cual por medio de una operación se estima la altura de artefacto.

Finalmente entonces, la posición X, Y, Z del cuadricóptero será estimada:

- X, Y por GPS.
- $Z < 4m$ aprox. Ultrasonido.
- $Z > 4m$ aprox. Barómetro.

Sistema Angular

El poder sensorial elegido para este sistema es una IMU (*Inertial Measurement Unit*). En este caso el chip que se adquirió tiene incorporado el barómetro, sensor que se utiliza como ya se mencionó para la estimación de la altura. La IMU se compone de tres sensores, acelerómetro giróscopo y magnetómetro, éstos son fusionados en la propia IMU con filtro de Kalman y se proporciona así la medida de su orientación mediante cuaterniones 19.1 (los cuales son convertidos a ángulos de Euler en la convención elegida mediante las librerías de software desarrolladas). La otra magnitud que da la IMU es la velocidad angular, cuyas componentes están en el sistema de coordenadas de la IMU (equivalente al del cuadricóptero). La velocidad angular medida a partir del giróscopo, será una magnitud importante a la hora de implementar el control de actitud, como se verá más adelante.

Capítulo 9

Caracterización de Sensores

9.1. IMUs

9.1.1. Descripción

La IMUs seleccionadas (3.3.1) se presentan como SiP (System in Package), el cual integra un acelerómetro, giróscopo y magnetómetro en un circuito integrado LGA (Land Grid Array). El acelerómetro y el giróscopo están basados en la tecnología MEMS (*Microelectromechanical systems*) [14], que consiste en la construcción litográfica de mecanismos móviles u oscilantes, que responden a los efectos de las fuerzas ficticias cuando su sistema de referencia (la IMU) es no-inercial. En general las medidas de desplazamiento son realizadas como medidas de capacidad de capacitores formados entre las partes móviles y fijas del mecanismo. Si bien estos sensores son muy accesibles económicamente, presentan la desventaja de tener características de ruido muy pobres, en particular exhiben *offsets* considerables.

Los magnetómetros encontrados en estas unidades funcionan en base al efecto Hall [15], sieno su principio de operación la medición de la diferencia de potencial transversal en un conductor por el cual circula una corriente y está inmerso en un campo magnetico. Una desventaja de los sensores Hall es que las medidas que proporcionan tienen una deriva o *drift* importante que debe ser compensado.

Las tablas 9.1 y 9.2 resumen las pricipales características de las IMUs que se utilizaron.

Característica	**Valor**
Fabricante/Modelo	Bosch BNO055
Acelerómetro	Rangos: $\pm 2g / \pm 4g / \pm 8g / \pm 16g$
Giróscopo	Rangos: de $\pm 125^o/s$ a $\pm 2000^o/s$
Magnetómetro	Rango: $\pm 1300\mu T$ (x,y) y $\pm 2500\mu T$ (z)

Tabla 9.1: Características de la IMU BNO055.

La IMU BNO055 implementa un algoritmo de auto calibración interno que actualiza las constantes de calibración (coeficiente y offset) constantemente. Esta IMU puede operar en varios modos, siendo los más relevantes para esta aplicación los modos *Fusion* y *AMG* (Acelerómetro, Magnetómetro, Giróscopo). En el modo fusion realiza internamente la fusión de acelerómetro, giróscopo y magnetómetro en un microprocesador cortex M0 de 32 bits incorporado que corre un filtro de Kalman extendido y produce datos de orientación parametrizada como cuaterniones 19.2, con

una tasa limitada a $100Hz$. En el modo AMG se puede leer los datos de los sensores (coregidos por la autocalibración) directamente de los registros. Se comprobó que los retardos entre el comando de lectura (i2c) y la transmisión de los datos oscilan en el rango $200\mu s - 6000\mu s$. Esta inconsistencia proviene del *clock stretching*[1] que realiza la interfaz i2c de la IMU y la hace inutilizable si se quiere tener una tasa de datos de los sensores mayor que los $100Hz$ que provee el modo Fusion.

Característica	Valor
Fabricante/Modelo	InvenSense MPU9250
Acelerómetro	Rangos: $\pm 2g / \pm 4g / \pm 8g / \pm 16g$
Giróscopo	Rangos: de $\pm 250^o/s$ a $\pm 2000^o/s$
Magnetómetro	Rango: $\pm 4800\mu T$

Tabla 9.2: Características de la IMU MPU9250.

9.2. Ultrasonido

9.2.1. Descripción

El sensor de ultrasonido se utilizará conjuntamente con el barómetro para determinar la altura del artefacto; mientras que el barómetro al basarse en diferencias de presiones es bueno para medir grandes distancias, el ultrasonido es bueno para medir distancias cortas, de $2cm$ a $400cm$.

Su funcionamiento consiste básicamente en el disparo de un trigger (ver figura 9.1 de [34]) de al menos $10\mu s$, luego del cual se emite un pulso ultrasónico de $40kHz$, éste se refleja en un objeto emitiendo un eco que es recibido por el sensor; este período de tiempo es registrado internamente mediante el ancho de un pulso que luego se utiliza para calcular la distancia mediante la siguiente fórmula:

$$Distancia = \frac{Ancho\ del\ pulso * Velocidad\ del\ sonido}{2} \tag{9.1}$$

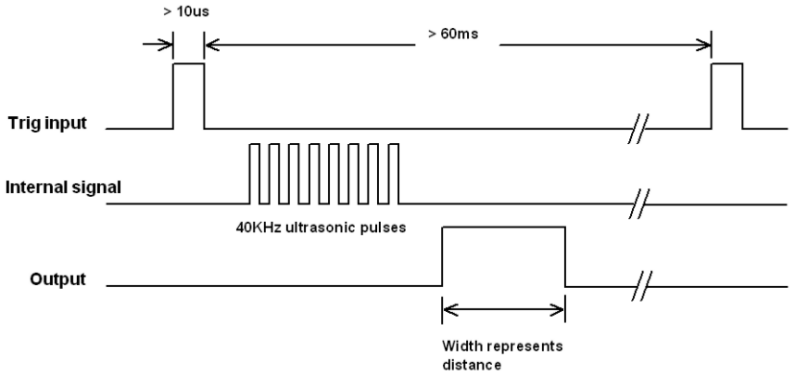


Figura 9.1: pulso ultrasonido

[1]El clock stretching es un mecanismo en el protocolo i2c mediante el cual se permite a un esclavo reducir la velocidad del bus manteniendo baja la señal de reloj (SCL).

9.2.2. Caracterización

El objetivo es caracterizar la respuesta del sensor de ultrasonido en función de la distancia de éste a un reflector.

Elementos utilizados:

- Sensor de ultrasonido HC-SR04.
- Placa de desarrollo Arduino Uno.
- Computadora.
- Cinta métrica.
- Pantalla de metal.
- Morsa.
- Mesa de soporte.

Descripción:

Se dispone el sensor con su eje de emisión en dirección horizontal, fijo a la mesa mediante una morsa. Se extiende la cinta métrica en el piso paralela al eje de emisión y con el cero coincidente con los bordes anteriores del emisor y receptor. La pantalla de metal se usa como reflector, ubicándola frente al sensor y perpendicular al eje de emisión.

Mediante un sketch de Arduino se maneja el disparo del pulso de ultrasonido del sensor y se recibe el rebote, midiendo el tiempo total de viaje en microsegundos. Con el tiempo de viaje se calcula la distancia a la placa reflectora (en cm) y se la despliega en el monitor serial del IDE de Arduino en la computadora. La posición del reflector se varía en intervalos de $10cm$, de $10cm$ a $440cm$. Se registran la posiciones del reflector en la cinta métrica y las distancias correspondientes calculadas por el sketch utilizado: $d = c\frac{\Delta t}{2}$. Se utilizó el valor $c = 343{,}2m/s = 0{,}03432cm/\mu s$.

Esquemas de conexión

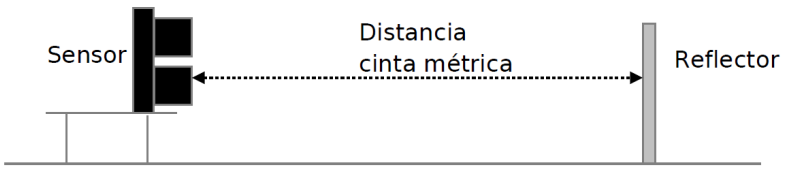


Figura 9.2: Esquema mecánico del experimento

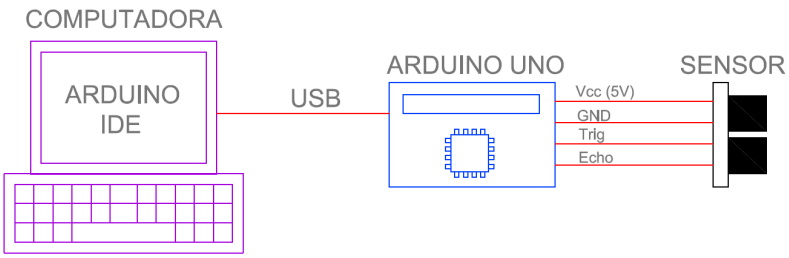


Figura 9.3: Esquema eléctrico del experimento

Procesamiento de datos

En la figura 9.4 pueden observarse los datos registrados contra las medidas tomadas con la cinta métrica. La ecuación que ajusta los datos es:

$$\Delta medida = 1{,}002\Delta sensor + 2{,}737 \tag{9.2}$$

Por lo que se tiene una gran precisión con un offset de $2{,}737cm$

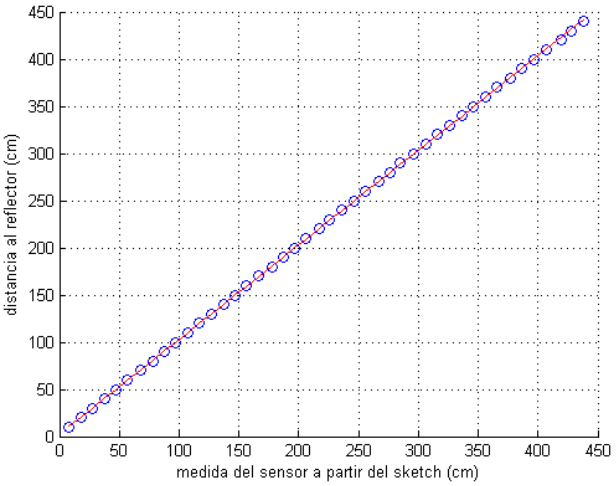


Figura 9.4: Distancia medida vs distancia sensor

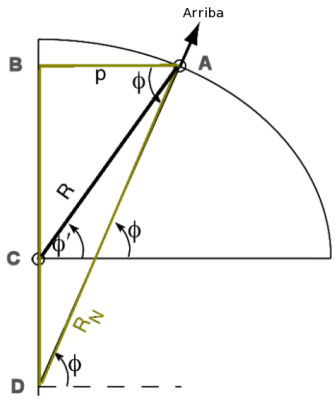
9.3. GPS

9.3.1. Descripción

El GPS se utiliza para determinar la ubicación geográfica del vehículo mediante trilateración satelital. Se utiliza entonces para la ubicación del robot un sistema de posicionamiento global con los errores en la medición que esto implica. El sensor adquirido da las coordenadas en el protocolo NMEA 0183 (National Marine Electronics Association), la comunicación del GPS con el micro Teensy 3.2 es serial con un tiempo de refresco de $100ms$. La medida que proporciona el GPS son los ángulos de latitud y longitud, los mismos son convertidos a coordenadas cartesianas (X, Y) para poder elaborar la autonomía de vuelo con puntos en ese sistema.

Consideraciones a tener en cuenta

Dado que la tierra está aplanada en sus polos, el radio polar es 23 Km menor que el ecuatorial, se debe diferenciar entre las latitudes geocéntricas y geodésicas como se muestra en la figura adyacente(de [33]). En donde:



ϕ representa la latitud Geodésica (WGS84 medida por el GPS, es la utilizada en los mapas)

ϕ' representa la latitud Geocéntrica

Debido a que el radio de curvatura R_N considerado para hallar la latitud es mayor que el radio de curvatura utilizado para hallar la longitud, la precisión en longitud es mayor que en latitud, dado que la medida de cada una se especifica con la misma cantidad de cifras.

9.3.2. Conversión de coordenadas

Para lograr la conversión de los ángulos dados por el GPS a coordenadas cartesianas, se considera el modelo del geoide WGS84 cuyos parámetros son:

- Semieje mayor: $a = 6378137m$
- Cuadrado de la excentricidad: $e^2 = 0{,}006694379990141 = 1 - \frac{b^2}{a^2}$ [2]

Radio de curvatura del paralelo: $R_N = \frac{a}{\sqrt{1-e^2 sin^2(lat_0)}}$
Radio de curvatura del meridiano: $R_M = \frac{a.(1-e^2)}{[1-e^2 sin^2(lat_0)]^{3/2}}$
$x = R_N.cos(lat_0).(long - long_0)\frac{\pi}{180}$
$y = R_M(lat - lat_0)\frac{\pi}{180}$

Siendo lat_0 y $long_0$ la latitud y longitud del origen considerado

9.3.3. Caracterización

El objetivo es caracterizar el GPS registrando medidas estáticas. Es importante caracterizar el GPS en medidas estáticas ya que es cuando presenta mayor error, se conoce de proyectos anteriores que el GPS mejora la performance en movimiento. Si bien es evidente que el cuadricóptero estará en movimiento, interesa conocer el comportamiento estático para poder implementar a la autonomía del robot, la funcionalidad de permanecer en un punto fijo. Por lo expuesto, caracterizando el sensor en posición estática, estamos considerando el peor caso en los errores de la medida. Esto será fundamental a la hora de implementar el filtro de Kalman para la fusión sensorial.

Elementos utilizados:

- GPS u-blox NEO 6.
- Teensy 3.2.
- Notebook.

Descripción:
Se define un punto de inicio que se considera como origen de coordenadas $(0, 0)$.
El GPS se conectada al Teensy y éste a un notebook.
Situados en el punto de origen, se configura el programa para tomar 1500 medidas, las cuales se promedian para obtener las coordenadas convertidas (X, Y) del origen.
Posteriormente se toman 12000 medidas estáticas (20 minutos) que se convierten a coordenadas (X, Y) referidas al punto de origen.

[2] en el elipsoide: a: semieje mayor, b: semieje menor

Procesamiento de datos

En la figura 9.5 pueden observarse los datos registrados correspondientes al experimento estático, el GPS fijo registrando los datos. El punto rojo central representa el origen de coordenadas (resultado de las 1500 medidas iniciales), los puntos azules representan las mediciones. Los círculos concéntricos representados en color negro y rojo tienen radios de 1 y 2 metros respectivamente.

La media y desviación estándar de las medidas en cada coordenada son:

$\mu_X = -0{,}572m$
$\sigma_X = 0{,}753m$
$\mu_Y = 0{,}264m$
$\sigma_Y = 0{,}5568m$

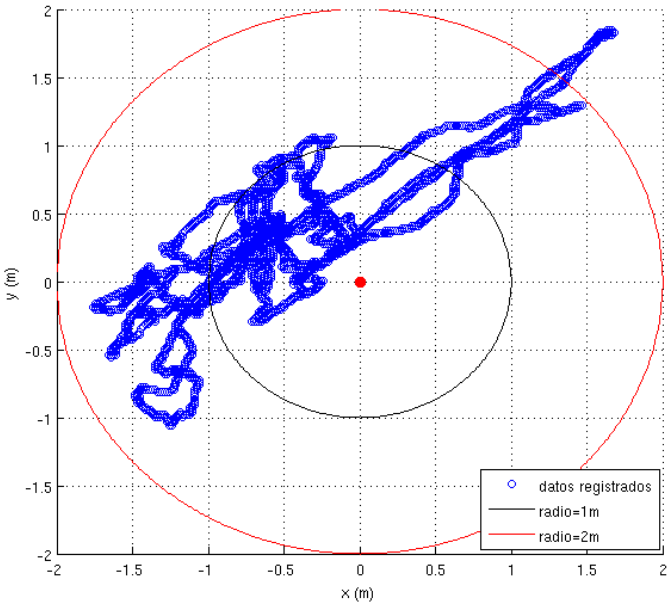


Figura 9.5: Experimento estático

El fabricante asegura que para medidas estáticas, al menos la mitad de los datos tomados estarán dentro de un círculo de 1 metro de radio; en este experimento la cantidad de registros en un radio de 1 metro son 6277 (de un total de 12000 medidas) por lo que se cumple dicha especificación.

La gran dispersión puede deberse al movimiento de los satélites que toma de referencia, cuando los datos registrados salen hacia fuera del circulo de 2 metros de radio podría tratarse de un caso en que el alejamiento de un satélite tenga una consecuencia de una perdida parcial de la señal que se restablece al sincronizarse con otro satélite para realizar la trilateración.

9.4. Barómetro

9.4.1. Descripción

El barómetro mide la temperatura y la presión absoluta. A partir de la medida de presión y conociendo la presión y temperatura a nivel del mar podemos calcular la altura absoluta para el robot. Se elije un barómetro para las medidas de altura ya que el GPS presenta una peor performance para la medida de esta magnitud, el barómetro también presenta sus problemas y particularidades, entre ellas, ya que la altura se calcula a partir de la presión, es sensible a cambios climáticos bruscos. Con la medida de este sensor, fusionada con el GPS vamos a tener la posición absoluta (XYZ) del cuadricóptero

9.4.2. Características

El fabricante del barómetro especifica un algoritmo para obtener las medidas de temperatura y presión del mismo. Este algoritmo fue implementado en lenguaje $C++$ para ser ejecutado en el microcontrolador Teensy. También el sensor presenta varios modos de funcionamiento en cuanto al tiempo de cada nueva medida y esto depende básicamente de la resolución que se requiera para la misma. Si se quiere más resolución se necesita más tiempo para tener una medida nueva, la comunicación con el sensor ya que está incorporado en el chip de la IMU es I2C, se elije el modo de funcionamiento en el cual cada medida nueva de temperatura y presión demora $16ms$ en refrescarse. Por tanto cada nueva medida de altura demora algo mas de $32ms$ con una precisión de décimas de centímetro.

Se parte de la ecuación del modelo de la atmósfera estándar internacional (ISA):

$$P_{ISA}(z) = p_0(1 - \frac{\Gamma}{T_0}z)^{\frac{g}{R\Gamma}} \tag{9.3}$$

donde:

- z es la altura
- R constante de los gases
- p_0 presión a nivel del mar
- T_0 temperatura a nivel del mar
- $\Gamma = -\frac{dT}{dz}$ el gradiente de temperatura suponiéndolo constante de 0 a 11 km
- g gravedad

Para simplificar el procesamiento se considera la atmósfera isoterma (atmósfera con simetría radial, aislada del resto del mundo y en equilibrio termodinámico a una temperatura uniforme T, es decir, cuando $\Gamma = 0$):

$$P_{ISO} = p_0 e^{\frac{-g(z-z_0)}{RT}} \Rightarrow (z - z_0) = h = -\frac{1}{c} L_n \left(\frac{P_{ISO}}{p_0} \right) \tag{9.4}$$

siendo $c = \frac{g}{RT}$

linealizando la ecuación anterior tenemos:

$P_{ISO} \cong p_0(1 - ch) = p_0 - cp_0h$

y finalmente:

$$h \cong \frac{1}{c}(1 - \frac{P_{ISO}}{p_0}) \tag{9.5}$$

9.4.3. Caracterización

El objetivo es caracterizar el barómetro, realizando y registrando medidas del mismo colocado en puntos de altura conocida.

Elementos utilizados:

- Cinta métrica.
- Notebook.
- Barómetro (integrado en la IMU, modelo MS5637).

Descripción:

Con la cinta métrica se mide la diferencia de altura entre pisos consecutivos sobre la escalera de la Facultad de Ingeniería. Se comienza a medir en piso 1. Se registran las medidas entre pisos, junto con los índices de las mediciones.

En las barandas de madera de cada piso, se coloca al barómetro conectado a la Notebook y se espera el tiempo suficiente como para recoger al menos 200 muestras.

Se registran los datos y se avanza hacia el próximo nivel de la escalera.

El procedimiento se repite primero subiendo del piso 1 al 6 y luego bajando desde éste hasta el 1.

Esquema del experimento

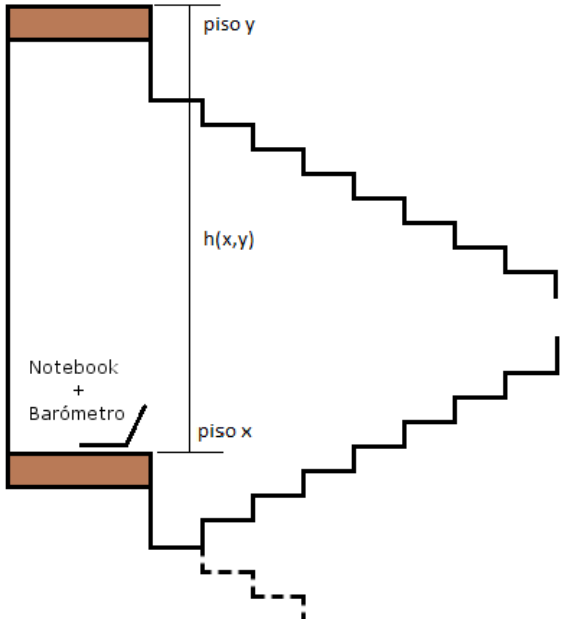


Figura 9.6: Esquema que ilustra los niveles de la escalera en donde se tomaron las medidas

9.4.4. Procesamiento de datos

En la figura 9.7 pueden observarse los datos registrados tanto en la ascenso como en el descenso por los pisos del edificio.

El experimento se realizó de acuerdo al orden mencionado en la "Descripción": se toman medidas en el piso 1 y se avanza de a un piso hasta llegar al 6º; luego se procede a bajar tomando medidas en cada piso inferior hasta llegar al 1º piso nuevamente; antes de tomar las medidas comenzó a llover, se estima que el cambio de presión atmosférica en ese período fue el causante de la diferencia tan notoria en las medidas de ascenso y descenso tomadas en el primer piso.

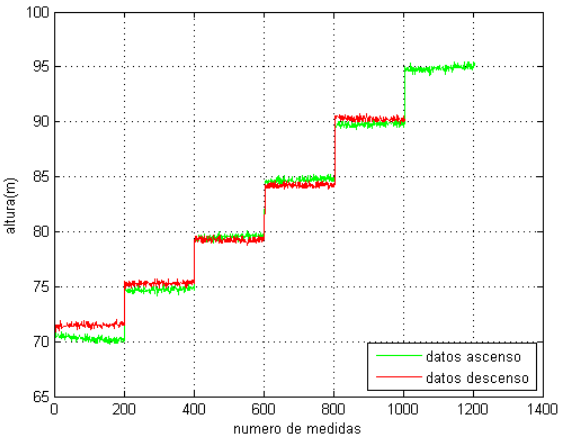


Figura 9.7: Experimento barómetro

La desviación estándar de los datos es:

$$\sigma_{ZBaro} = 0{,}1825m \tag{9.6}$$

Los datos reales (medidos con la cinta métrica) en comparación con los registrados con el Barómetros se muestran en la siguiente tabla:

Tabla 9.3: Datos reales - registrados

Pisos	***Real (m)***	***Barómetro (m)***
1 al 2	4,63	4,46
2 al 3	3,37	4,79
3 al 4	4,63	5,19
4 al 5	4,63	5,07
5 al 6	4,63	5,09

Capítulo 10

Fusión sensorial: Filtro de Kalman

10.1. Introducción

El estado rotacional (velocidad angular y orientación) está determinado directamente por la IMU, por lo que no es necesario un procesamiento posterior (salvo por la conversión del cuaternión a los ángulos de Euler).

Por otro lado, en el caso del estado traslacional, se dispone de medidas del GPS y Barómetro por separado, con un cierto error cada uno, como se expuso en los capítulos anteriores de caracterización.

Otro dato del que se dispone para el movimiento traslacional es el de aceleración absoluta, proporcionado por el acelerómetro de la IMU.

En este capítulo, se presenta una técnica para estimar el estado traslacional (X, Y, Z); (v_X, v_Y, v_Z) utilizando las medidas del GPS, Barómetro y aceleración, basada en el filtro de Kalman lineal. Cabe destacar que para la coordenada Z, siempre se utilizará la medida del sensor de ultrasonido cuando esté disponible, en lugar de la estimación que arroje esta técnica. Esto es debido a que la caracterización del sensor de ultrasonido, ha demostrado que la exactitud de la medida es de $1cm$, que es menor que la proporcionada por la estimación.

10.2. Filtro de Kalman

El filtro de Kalman es un algoritmo que produce una estimación óptima, en el sentido de mínimos cuadrados, de las variables de estado de un sistema cuando éstas obedecen ecuaciones de evolución con ruido y se dispone además de medidas de las mismas, también ruidosas. El algoritmo tiene como hipótesis fundamental que el ruido tiene una distribución Gaussiana de media nula. La utilización del filtro de Kalman en la forma que se presenta, fue sugerida por [45]

10.3. Ecuaciones y operación del filtro

El filtro de Kalman se basa en dos operaciones básicas sobre el estado estimado $\hat{x}$, una *proyección o predicción* dada por la ecuación de evolución del mismo y una *correción* que se lleva a cabo cuando se tiene una medida disponible. Las ecuaciones del filtro de Kalman lineal son las siguientes [16]:

$$G_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \tag{10.1}$$
$$\hat{x}_k = \hat{x}_k + G_k (z_k - H\hat{x}_k) \tag{10.2}$$
$$P_k = (I - G_k H) P_k^{-1} \tag{10.3}$$
$$\hat{x}_{k+1} = \Phi \hat{x}_k + B u_k \tag{10.4}$$
$$P_{k+1}^- = \Phi P_k \Phi^T + Q \tag{10.5}$$

La matrices Φ y B son las que proporcionan la evolución del estado, basado en el estado anterior y en la entrada u.

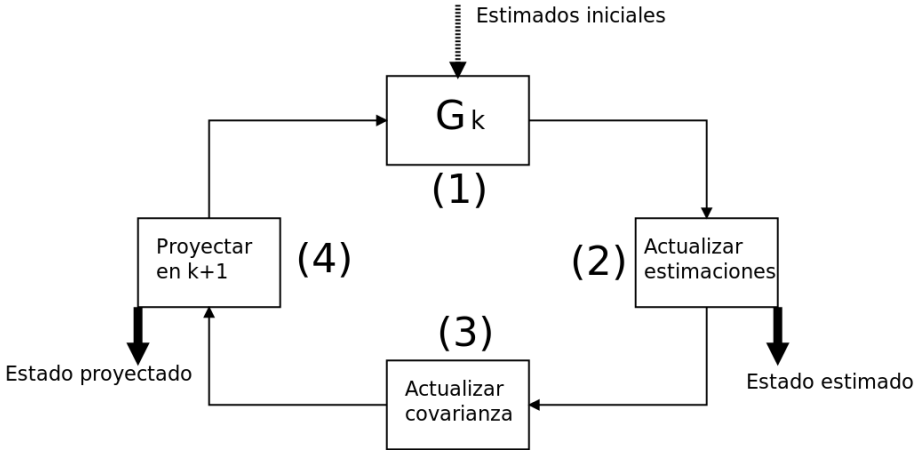


Figura 10.1: Diagrama de operación del filtro de Kalman

Dada la aceleración, medida por la IMU y considerada constante entre actualizaciones separadas por un intervalo DT, las ecuaciones de evolución del sistema son:

$$\vec{v}_t = \vec{v}_{t-1} + \vec{a}_{t-1} DT \tag{10.6}$$
$$\vec{r}_t = \vec{r}_{t-1} + \vec{v}_{t-1} DT + \frac{1}{2} \vec{a}_{t-1} DT^2 \tag{10.7}$$

que se pueden normalizar para tener un vector de estado con dimensiones homogéneas (de distancia) definiendo $\vec{s}_t = DT\, \vec{v}_t$:

$$\vec{\xi}_t = \begin{pmatrix} \vec{r}_t \\ \vec{s}_t \end{pmatrix} = \Phi\, \vec{\xi}_{t-1} + B\vec{u}_t + \vec{w}_t \tag{10.8}$$

donde las matrices se obtienen a partir de las ecuaciones de evolución y son:

$$\Phi = \begin{pmatrix} I_3 & I_3 \\ 0_3 & I_3 \end{pmatrix} \qquad B = \alpha \begin{pmatrix} I_3 \\ 2I_3 \end{pmatrix} \tag{10.9}$$

con $\alpha = \frac{DT^2}{2}$

el término de ruido $\vec{w}_t$ se calcula considerando el error $\delta\vec{a}_t$ en la aceleración:

$$\vec{a}_t \to \vec{a}_t + \delta\vec{a}_t \implies B\vec{u}_t \to B\vec{u}_t + B\delta\vec{a}_t \tag{10.10}$$

entonces $\vec{w}_t = B\delta\vec{a}_t$. La matriz de covariancia Q se obtiene tomando el valor esperado de $\vec{w}_t$:

$$Q = \mathbf{E}\{\vec{w}_t\, \vec{w}_t^T\} = B\mathbf{E}\{\delta\vec{a}_t\, \delta\vec{a}_t^T\}B^T = B\sigma_a^2 I_3 B^T = \sigma_a^2 BB^T \tag{10.11}$$

que se calcula a partir de la expresión para B:

$$Q = \sigma_a^2\alpha^2 \begin{pmatrix} I_3 & 2I_3 \\ 2I_3 & 4I_3 \end{pmatrix} \tag{10.12}$$

La medida de la que se dipone es la posición XY del GPS y la altura h del barómetro (o del sensor de ultrasonido, cuando esté disponible):

$$\vec{z}_t = \vec{r}_t = H\xi_t + \vec{\nu}_t \qquad\qquad H = (I_3 \quad 0_3)$$

de donde se obtiene la matriz de covariancia del ruido de la medida:

$$R = \mathbf{E}\{\vec{\nu}_t\, \vec{\nu}_t^T\} = \mathbf{E}\{\delta\vec{r}_t\, \delta\vec{r}_t^T\} = \begin{pmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_h^2 \end{pmatrix} \tag{10.13}$$

10.4. Implementación y resultados

Primeramente se implementó un prototipo en *Matlab*2014 para probar el funcionamiento del algoritmo. Se generaron trayectorias en una y dos dimensiones, a las que se agregó ruido Gaussiano de media nula para simular errores en las mediciones. Los datos de aceleración y las medidas ruidosas fueron utilizados como entradas al algoritmo del filtro de Kalman.

En la figura 10.2 se muestran los resultados para dos dimensiones, la curva negra es la trayectoria original, los puntos en verde las medidas ruidosas, y la linea azul punteada la estimación que arroja el filtro.

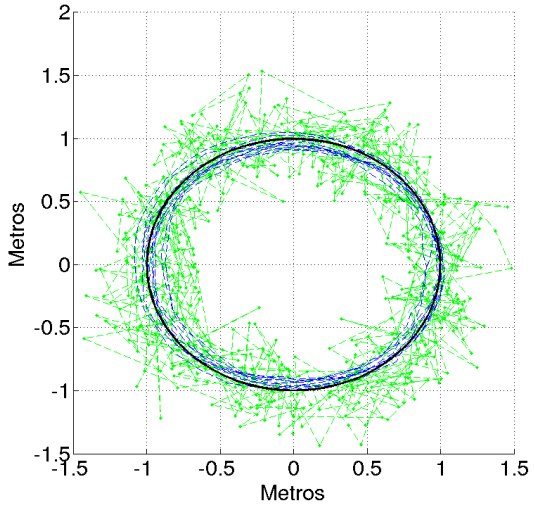


Figura 10.2: Simulación filtro Kalman en Matlab

En segunda instancia se implementó en software el algoritmo del filtro y se programó en el micro Teensy 3.2, encargado de la estimación de todo el estado del cuadricóptero.

El experimento de prueba del filtro en software consistió en una prueba estática del GPS y la IMU (la cual tiene integrado el barómetro en el chip) es decir, tomar medidas de X, Y, Z con los sensores en un punto fijo, durante unos 10 minutos aproximadamente. En la figura 10.3 se muestran los resultados del experimento para las medidas en X, Y. Se puede observar en los resultados que en las medidas del GPS no se obtiene una mejora sustancial en los errores de medida, si bien la curva de puntos X, Y que se obtiene del filtro aparece más suave. Lo que el filtro no puede evitar es el error de deriva que posee el GPS, que depende de otros factores como fluctuaciones atmosféricas que influyen en la propagación de las señales de los satélites, que no pueden ser modeladas como ruido Gaussiano de media nula. El filtro sí parece eliminar los errores aleatorios (la dispersión en corto plazo), es por eso que la curva queda alisada.

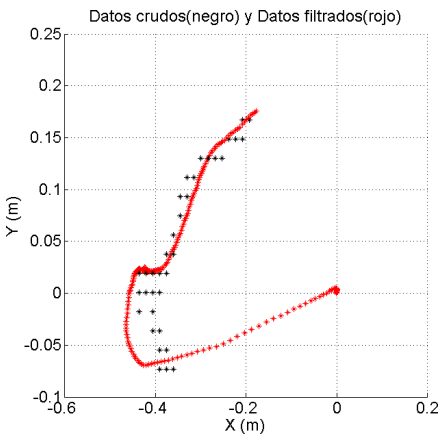


Figura 10.3: Resultados de prueba estática KF, plano XY.

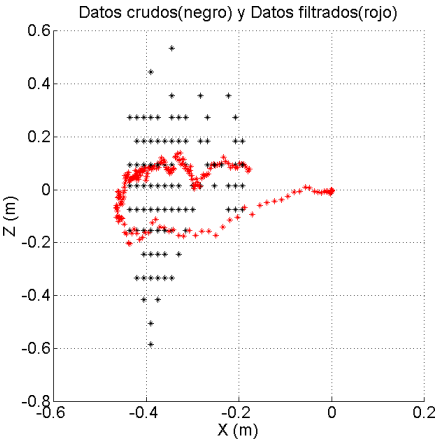


Figura 10.4: Resultados de prueba estática KF, plano XZ.

En la figura 10.4 vemos los resultados en las ordenadas para la coordenada Z del estado traslacional, cuya medida viene del barómetro. En este caso sí vemos una mejora sustancial de la estimación de la altura ya que al tratarse de medidas en un punto fijo, se pasa de tener una dispersión que está aproximadamente entre $[-0{,}6, 0{,}6]$ metros a una entre $[-0{,}2, 0{,}15]$ metros.

Finalmente en la figura 10.5 se muestran la comparación en 3d de los datos crudos y las estimaciones del filtro.

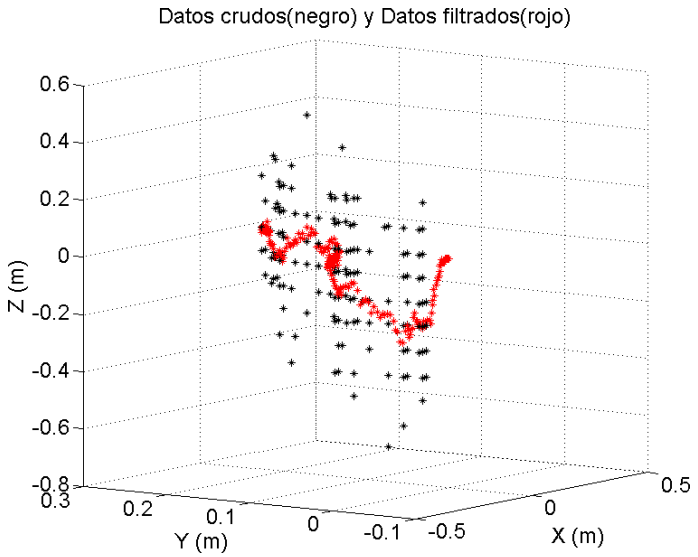


Figura 10.5: Datos crudos y filtrados

Capítulo 11

Fusión sensorial: Filtro Complementario

11.1. Introducción

En el caso de las variables de orientación, al contrario que para la traslación, requiere tasas de muestreo mayor, ya que la estabilidad angular del cuadricóptero demanda a su vez frecuencias del *loop* de control del orden de $100Hz$ o superiores. Si bien el filtro de Kalman produce una estimación óptima, como se mencionó anteriormente, posee la desventaja importante de que requiere una carga computacional elevada para las plataformas de procesamiente usadas en este trabajo, si se desea una tasa de actualización alta. Es por ello que en la estimación del estado angular se recurre a soluciones alternativas, en particular al *filtro complementario* ([18], [36]). La relación entre el filtro de Kalman y el filtro complementario puede consultarse en [17].

11.2. Filtro complementario

Las relaciones (11.1) - (11.3) son el modelo utilizado de las respuestas de los sensores a la aceleración lineal $\vec{a}$ y velocidad angular $\vec{\omega}$ respecto al sistema absoluto y el campo magnético de la tierra B_T. .

$$\vec{a}_{imu} = (\vec{a} - \vec{g}) + \vec{b}_a + \vec{\eta}_a \tag{11.1}$$

$$\vec{\omega}_{imu} = \vec{\omega} + \vec{b}_\omega + \vec{\eta}_\omega \tag{11.2}$$

$$\vec{B}_{mag} = \vec{B}_T + \vec{b}_B + \vec{\eta}_B \tag{11.3}$$

los vectores de *bias* $\vec{b}_a$, $\vec{b}_\omega$ y $\vec{b}_B$ tienen variación temporal lenta y modelan los errores en las bajas frecuencias, mientras que los términos $\vec{\eta}_a$, $\vec{\eta}_\omega$ y $\vec{\eta}_B$ denotan procesos de ruido de frecuencias altas sin componentes en DC (valor medio). Si bien el efecto de los términos de *bias* puede ser reducido en cierto grado sustrayendo el valor medio de una serie de medidas de calibración, es relevante conservarlos en el modelo, puesto que aun teniendo magnitud pequeña sus efectos acumulativos son importantes.

Las medidas crudas entregadas son las componentes de estos vectores referidas al sistema del sensor, la matriz de rotación ${}^B_A R$ proporciona las componentes en el sistema absoluto de acuerdo a (11.4) - (11.6), donde se ha conservado el mismo nombre para los vectores de *bias* rotados.

$$^{A}\vec{a}_{imu} = {}^{B}_{A}R^{T}\,(^{B}\vec{a} - {}^{B}\vec{g}) + \vec{b}_a + \vec{\eta}_a \tag{11.4}$$

$$^{A}\vec{\omega}_{imu} = {}^{B}_{A}R^{T}\,{}^{B}\vec{\omega} + \vec{b}_\omega + \vec{\eta}_\omega \tag{11.5}$$

$$^{A}\vec{B}_{mag} = {}^{B}_{A}R^{T}\,{}^{B}\vec{B}_T + \vec{b}_B + \vec{\eta}_B \tag{11.6}$$

Con estas medidas, la determinación de la orientación puede llevarse a cabo de dos maneras independientes. En una de ellas, a partir de las componentes en el sistema del cuadricóptero (sensor) de $-\vec{g}$ y $\vec{B}_T$ y de las componentes (conocidas) en el sistema de la tierra de esos campos, es posible calcular la rotación (dada por un cuaternión q) que transforma unas en otras, lo que equivale a conocer la orientación. El otro procedimiento consiste en evaluar primero la tasa de cambio $\dot{q}$ del cuaternión q que define la orientación (19.13), e integrarla en el tiempo para obtener un estimado q_t, partiendo de una condición inicial conocida. Si bien cada uno de estos métodos proporciona en principio una solución al problema, estas soluciones no son satisfactorias por separado, debido a las limitaciones inherentes de cada sensor.

En un sistema actuado como un cuadricóptero, las fuerzas de control se ven reflejadas en el acelerómetro, por lo que la aceleración medida no será solamente la debida al vector conocido $\vec{g}$, sino que están contaminadas por el vector desconocido $\vec{a}$ de variación rápida. Además, las vibraciones producidas por los propulsores introducen ruido de alta frecuencia. Es por eso que la orientación del acelerómetro sólo es confiable a largo plazo, lo que en el dominio de la frecuencia implica intercalar un filtro pasa-bajos.

El giróscopo por otra parte es menos suceptible a perturbaciones externas; pero debido a la integración temporal, la existencia de un *bias* hace errónea la estimación al cabo de un breve tiempo. Visto de otra manera, el término $\vec{b}_\omega$ en (11.2) destruye cualquier información en las bajas frecuencias de las medidas del giróscopo; sin embargo, la información de este sensor en las altas frecuencias es confiable. Este problema hace necesario que la estimación dada por el segundo método deba ser procesada por un filtro pasa-altos.

Los filtros complementarios proporcionan en estos casos un método de fusión sensorial eficaz y que no requieren un gran esfuerzo computacional. Sea la magnitud θ y su tasa de cambio ω relacionadas por un integrador de primer orden

$$\dot{\theta} = \omega \tag{11.7}$$

y las medidas z_θ y z_ω de ellas, proporcionadas por sensores con las características (11.1) - (11.3), donde z_θ corresponde a la estimación de la orientación mediante el acelerómetro y magnetómetro descripta y z_ω/s es la estimación a partir del giróscopo. Las medidas pueden ser fusionadas en un estimado $\hat{\theta}$ por la acción del filtro

$$\hat{\theta} = F_1(s)z_\theta + F_2(s)\frac{z_\omega}{s} \tag{11.8}$$

en el cual $F_1(s)$ es pasa-bajos y $F_2(s)$ es pasa-altos. El filtro se llama *complementario* si se satisface la relación

$$F_1(s) + F_2(s) = 1$$

lo que asegura que las frecuencias de corte de los filtros pasa-bajos y pasa-altos son iguales y por tanto el filtro resultante cubre todo el espectro de frecuencia en forma homogénea. La figura 11.1 es el diagrama de bloques de una estructura del filtro complementario adecuada para el diseño e implementación práctica. La relación de la salida $\hat{\theta}$ con las entradas z_θ y z_ω está dada por

Figura 11.1: Diagrama de bloques del filtro complementario.

$$\hat{\theta}(s) = T(s)z_\theta \; + \; S(s)\frac{z_\omega}{s} \tag{11.9}$$

$$\text{con :} \qquad T(s) = \frac{C(s)}{s + C(s)} \tag{11.10}$$

$$S(s) = \frac{s}{s + C(s)} \tag{11.11}$$

donde claramente se cumple la condición de complementariedad $T(s)+S(s) = 1$. La elección más simple (y más común) es la de una realimentación proporcional $C(s) = k$, que lleva a la dinámica en el dominio del tiempo siguiente

$$\frac{d\hat{\theta}}{dt} = z_\omega + k(z_\theta - \hat{\theta}) \tag{11.12}$$

En una implementación digital del filtro esta dinámica debe ser discretizada en un cierto intervalo de muestreo DT. Si se utiliza el esquema simple de diferencia hacia atrás para $\frac{d\hat{\theta}}{dt}$, el estimado a tiempo discreto t se obtiene de la relación

$$\hat{\theta}_t = \gamma z_\theta \; + \; (1 - \gamma)\left(\hat{\theta}_{t-1} + DT z_\omega\right) \tag{11.13}$$

$$\text{con :} \qquad \gamma = \frac{k\,DT}{1 + k\,DT} \tag{11.14}$$

En este trabajo se utilizó el filtro desarrollado en [36], basado en cuaterniones y de bajos requerimientos computacionales. El algoritmo consta de tres pasos escenciales:

1. Cálculo de la orientación a partir del acelerómetro y magnetómetro.
2. Cálculo de la orientación a partir del giróscopo.
3. Fusión de las estimaciones.

El paso 1 es realizado como un proceso de minimización. Sean ${}^{A}\hat{g} = (0, 0, 1)$ las componentes en el sistema absoluto (tierra) de $-\vec{g}$ normalizada y ${}^{A}\hat{b} = (0, b_y, b_z)$ las componentes en el sistema absoluto de $\vec{B}_T$ normalizado. Si ${}^{B}\hat{a}$ y ${}^{B}\hat{m}$ son las medidas normalizadas del acelerómetro y magnetómetro respectivamente, las funciones a valores vectoriales

$$f_g(\hat{q}, \hat{a}) = {}^{B}_{A}\hat{q}^* \otimes {}^{A}\hat{g} \otimes {}^{B}_{A}\hat{q} - {}^{B}\hat{a} \tag{11.15}$$

$$f_b(\hat{q}, \hat{m}) = {}^{B}_{A}\hat{q}^* \otimes {}^{A}\hat{b} \otimes {}^{B}_{A}\hat{q} - {}^{B}\hat{m} \tag{11.16}$$

son medidas de error del cuaternión estimado ${}^{B}_{A}\hat{q}$ respecto del que corresponde a la rotación real que lleva el sistema S_A en el S_B, ${}^{B}_{A}q$. Si se contruye una función objetivo que incluya a ambas

$$f_{gb} = \begin{bmatrix} f_g(\hat{q}, \hat{a}) \\ f_b(\hat{q}, \hat{m}) \end{bmatrix} \tag{11.17}$$

el problema se reduce a minimizar f_{gb}. El algoritmo de optimización utilizado es el método del gradiente en la foma

$${}^{B}_{A}\hat{q}_{\nabla,t} = {}^{B}_{A}\hat{q}_{\nabla,t-1} - \mu_t \frac{\nabla f_{gb}}{||\nabla f_{gb}||} \tag{11.18}$$

donde μ_t es el paso que controla la convergencia. La ventaja de este método es que para esta aplicación se requiere sólo una iteración del método para obtener resultados aceptables. El resultado de este paso es el estimado ${}^{B}_{A}\hat{q}_{\nabla,t}$.

El paso 2 consiste en calcular una aproximación de la derivada de ${}^{B}_{A}\hat{q}$ dada por la ecuación (19.13) con el estimado previo y la velocidad angular medida $\vec{\omega}_t$ (11.19) y utilizarla en un integrador discreto (11.20) para actualizar la estimación

$${}^{B}_{A}\dot{\hat{q}}_{\omega,t} := \frac{1}{2} {}^{B}_{A}\hat{q}_{t-1} \otimes \vec{\omega}_t \tag{11.19}$$

$${}^{B}_{A}\hat{q}_{\omega,t} = {}^{B}_{A}\hat{q}_{t-1} + {}^{B}_{A}\dot{\hat{q}}_{\omega,t} DT \tag{11.20}$$

Finalmente, el paso 3 consiste en la fusión de ${}^{B}_{A}\hat{q}_{\nabla,t}$ y ${}^{B}_{A}\hat{q}_{\omega,t}$ a través de un filtro complementario en la forma (11.13), con algunas simplificaciones adicionales. La elección de los parámetros del filtro puede consultarse en [36]. El algoritmo además puede incluir fácilmente un método de corrección de *bias* para el giróscopo y magnetómetro.

El algoritmo de fusión sensorial para la orientación se emplea en el caso de utilizar la IMU MPU9250 (9.2). En [36] se encuenttra una implementación del mismo en

lenguaje C++, fácilemte adaptable al proyecto si se ajustan los parámetros necesarios. La IMU fue configurada en el rango $\pm 2g$ para el acelerómetro[1] y $\pm 2000^{o}/s$ para el giróscopo. Esta elección es conveniente para que el giróscopo no sature ante cambios muy bruscos de orientación y reduce el ruido en los LSB. De hecho se comprobó que el algoritmo no funcionaba bien con rangos bajos de velocidad angular.

El tiempo de ejecución en la plataforma Tiva C (3.4.2) es de unas decenas de microsegundo.

[1]Se debe observar que los chips ofrecidos por algunos vendedores son *samples* y las salidas del acelerómetro tienen la mitad de bits de sensibilidad que lo manifestado en la hoja de datos.

Parte V

Automatización y control

Abordaje del problema de control y descripción de las soluciones implementadas.

Capítulo 12

Estructura del Sistema de Control

El control del vehículo fue implementado con una arquitectura de capas o jerárquica [21] (p. 1316), basada en tres niveles, que en general comprenden *Planeación*, *Seguimiento* y *Control*. La estructura para este caso en particular se muestra en la figura 12.1, donde cada bloque representa una tarea implementada en software y su salida es comunicada a la capa que tiene debajo.

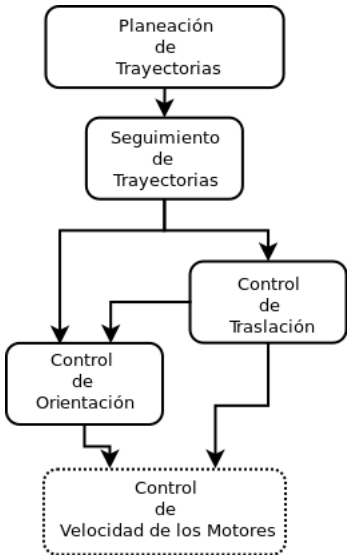


Figura 12.1: Jerarquía del sistema de control.

- Planeación: Esta capa está asociada a la toma de decisiones de alto nivel. Es donde interviene directamente el usuario para fijar los *waypoints* y donde se genera el camino a seguir para unirlos. Este proceso ocurre *offline*, fuera de las plataformas de procesamiento del vehículo. El resultado de este paso es un archivo que codifica las curvas y es lo que se proporciona a la capa de seguimiento.
- Seguimiento: El seguimiento de una trayectoria generada se efectúa a través del algoritmo de *Carrot Chase* de [24]. Este procedimiento realiza primero una discretización de las curvas suministradas por la capa de planeación en puntos llamados *sub-waypoints* y luego cada vez que es invocado calcula el rumbo de vuelo que es necesario seguir, desde la posición en la que se encuentre el vehículo, para alcanzar el próximo *sub-waypoint*, con un cierto margen de error. El resultado a este nivel es un rumbo o ángulo de *yaw* que es tomado por la capa de control. La implementación utilizada de los algoritmos de planeación y seguimiento descriptos partió de la realizada en el proyecto uQuad2 [2].

- Control: La capa de control está vinculada a la orientación y a la estabilidad en el vuelo del vehículo. Se encarga de calcular y ejecutar las acciones sobre los grados de libertad del sistema físico para alcanzar los valores deseados o *setpoints* fijados por las capas superiores, entregando como salida el conjunto de las cuatro velocidades de rotación de los motores deseadas (n_1, n_2, n_3, n_4).

Esta división del problema en capas simplifica el diseño del sistema y la verificación por bloques del funcionamiento.

Capítulo 13

Planeación y seguimiento de trayectorias

13.1. Planeación de trayectorias

Como se mencionó brevemente en el capítulo de "solución adoptada"II, entre Dubins[22], Shkel y Lumelsky[23] logran reducir las curvas de largo mínimo entre 2 waypoints a las del tipo RSR, LSL, LSR y RSL, siendo R un arco de radio fijo recorrido en sentido horario, L un arco de radio fijo recorrido en sentido antihorario y S una trayectoria recta. A continuación se resumen los cálculos para hallar estas curvas parciales (R,L,S) que se realizan en el trabajo de Shkel y Lumelsky, en donde se normalizan las distancias tomando el rario r como unidad, la distancia entre waypoints resulta entonces $d = \frac{D}{r}$ donde D es la distancia real.

Tenemos entonces 3 transformaciones que llevan un waypoint arbitrario $(x, y, \phi) \in \mathbb{R}^3$ en su correspondiente imagen (waypoint destino) en $\mathbb{R}^3$:

$$L_v(x, y, \phi) = (x + sin(\phi + v) - sin\phi, y - cos(\phi + v) + cos\phi, \phi + v), \tag{13.1}$$
$$R_v(x, y, \phi) = (x - sin(\phi - v) + sin\phi, y + cos(\phi - v) - cos\phi, \phi - v), \tag{13.2}$$
$$S_v(x, y, \phi) = (x + vcos\phi, y + vsin\phi, \phi) \tag{13.3}$$

Donde v indica la longitud de cada curva (R,L o S).

Por ejemplo, si tenemos un waypoint inicial con coordenadas $(0, 0, \alpha)$ y uno final $(d, 0, \beta)$, la curva hecha con segmentos L,S y R de longitudes t, p, q respectivamente será tal que:

$$R_q(S_p(L_t(0, 0, \alpha))) = (d, 0, \beta)$$

y tendrá una longitud total de

$$\mathbb{L} = t + p + q$$

De este modo podemos realizar esta composición de transformaciones para cada uno de los casos de nuestro interés:

1. $L_q(S_p(L_t(0,0,\alpha))) = (d,0,\beta)$

$$\begin{cases} p\,cos(\alpha+t) - sin\alpha + sin\beta = d \\ p\,sin(\alpha+t) + cos\alpha - cos\beta = 0 \\ \alpha + t + q = \beta \end{cases} \tag{13.4}$$

la solución de este sistema respecto a los segmentos t, p y q es:

$$\begin{cases} t_{lsl} = -\alpha + arctan\frac{cos\beta - cos\alpha}{d + sin\alpha - sin\beta} \\ p_{lsl} = \sqrt{2 + d^2 - 2cos(\alpha-\beta) + 2d(sin\alpha - sin\beta)} \\ q_{lsl} = \beta - arctan\frac{cos\beta - cos\alpha}{d + sin\alpha - sin\beta} \end{cases} \tag{13.5}$$

y análogamente para el resto de los casos[1]

2. $R_q(S_p(R_t(0,0,\alpha))) = (d,0,\beta)$

$$\begin{cases} t_{rsr} = \alpha - arctan\frac{cos\alpha - cos\beta}{d - sin\alpha + sin\beta} \\ p_{rsr} = \sqrt{2 + d^2 - 2cos(\alpha-\beta) + 2d(sin\beta - sin\alpha)} \\ q_{rsr} = -\beta + arctan\frac{cos\alpha - cos\beta}{d - sin\alpha + sin\beta} \end{cases} \tag{13.6}$$

3. $R_q(S_p(L_t(0,0,\alpha))) = (d,0,\beta)$

$$\begin{cases} t_{lsr} = -\alpha + arctan\frac{-cos\alpha - cos\beta}{d + sin\alpha + sin\beta} - arctan(\frac{-2}{p_{lsr}}) \\ p_{lsr} = \sqrt{-2 + d^2 + 2cos(\alpha-\beta) + 2d(sin\beta + sin\alpha)} \\ q_{lsr} = -\beta + arctan\frac{-cos\alpha - cos\beta}{d + sin\alpha + sin\beta} - arctan(\frac{-2}{p_{lsr}}) \end{cases} \tag{13.7}$$

4. $L_q(S_p(R_t(0,0,\alpha))) = (d,0,\beta)$

$$\begin{cases} t_{rsl} = \alpha - arctan\frac{cos\alpha + cos\beta}{d - sin\alpha - sin\beta} + arctan(\frac{2}{p_{rsl}}) \\ p_{rsl} = \sqrt{-2 + d^2 + 2cos(\alpha-\beta) - 2d(sin\beta + sin\alpha)} \\ q_{rsl} = \beta - arctan\frac{cos\alpha + cos\beta}{d - sin\alpha - sin\beta} + arctan(\frac{2}{p_{rsl}}) \end{cases} \tag{13.8}$$

[1]Se exponen las soluciones, el resto de los calculos pueden encontrarse en [23]

Ahora solo resta determinar cuál es la composición de curvas que minimiza la longitud para cada caso; de acuerdo a [23], se puede determinar esta curva dependiendo del cuadrante en el que se encuentren los angulos de partida y llegada de cada waypoint de acuerdo a la siguiente tabla[2]:

Final Quadrant / Initial Quadrant	1	2	3	4
1	RSL	if $S_{12} < 0$ then RSR if $S_{12} > 0$ then RSL	if $S_{13} < 0$ then RSR if $S_{13} > 0$ then LSR	if $S^1_{14} > 0$ then LSR if $S^2_{14} > 0$ then RSL else RSR
2	if $S_{21} < 0$ then LSL if $S_{21} > 0$ then RSL	if $S^1_{22} < 0$ then LSL if $S^1_{22} > 0$ then RSL if $S^2_{22} < 0$ then RSR if $S^2_{22} > 0$ then RSL	RSR	if $S_{24} < 0$ then RSR if $S_{24} > 0$ then RSL
3	if $S_{31} < 0$ then LSL if $S_{31} > 0$ then LSR	LSL	if $S^1_{33} < 0$ then RSR if $S^1_{33} > 0$ then LSR if $S^2_{33} < 0$ then LSL if $S^2_{33} > 0$ then LSR	if$S_{34} < 0$ then RSR if$S_{34} > 0$ then LSR
4	if $S^1_{41} > 0$ then RSL if $S^2_{41} > 0$ then LSR else LSL	if $S_{42} < 0$ then LSL if $S_{42} > 0$ then RSL	if $S_{43} < 0$ then LSL if $S_{43} > 0$ then LSR	LSR

Figura 13.1: Tabla de camino mínimo según cuadrante de angulo de partida y llegada

[2]Extraída de [23].

Donde :

$$S_{12} = p_{rsr} - p_{rsl} - 2(q_{rsl} - \pi) \tag{13.9}$$
$$S_{13} = t_{rsr} - \pi) \tag{13.10}$$
$$S^1{}_{14} = t_{rsr} - \pi) \tag{13.11}$$
$$S^2{}_{14} = q_{rsr} - \pi) \tag{13.12}$$
$$S_{21} = p_{lsl} - p_{rsl} - 2(t_{rsl} - \pi) \tag{13.13}$$
$$\begin{cases} si\,\alpha > \beta\ S^1{}_{22} = p_{lsl} - p_{rsl} - 2(t_{rsl} - \pi) \\ si\,\alpha < \beta\ S^2{}_{22} = p_{rsr} - p_{rsl} - 2(q_{rsl} - \pi) \end{cases} \tag{13.14}$$
$$S_{24} = q_{rsr} - \pi) \tag{13.15}$$
$$S_{31} = q_{lsl} - \pi) \tag{13.16}$$
$$\begin{cases} si\,\alpha < \beta\ S^1{}_{33} = p_{rsr} - p_{lsr} - 2(t_{lsr} - \pi) \\ si\,\alpha > \beta\ S^2{}_{33} = p_{lsl} - p_{lsr} - 2(q_{lsr} - \pi) \end{cases} \tag{13.17}$$
$$S_{34} = p_{rsr} - p_{lsr} - 2(t_{lsr} - \pi) \tag{13.18}$$
$$S^1{}_{41} = t_{lsl} - \pi) \tag{13.19}$$
$$S^2{}_{41} = q_{lsl} - \pi) \tag{13.20}$$
$$S_{42} = t_{lsl} - \pi) \tag{13.21}$$
$$S_{43} = p_{lsl} - p_{lsr} - 2(q_{lsr} - \pi) \tag{13.22}$$
$$\tag{13.23}$$

Concluyendo con la determinación de la curva que minimiza el recorrido para todos los casos.

Este resultado es válido para el plano (x, y), puede extenderse a 3 dimensiones haciendo que el robot cambie su altura a velocidad constante durante el recorrido entre el waypoint inicial y final de acuerdo a la siguiente ecuación:

$$z = z_i + \frac{l(x,y)}{L}(z_f - z_i) \tag{13.24}$$

Donde:
z es la altura que se está calculando
z_i es la altura del waypoint inicial
z_f es la altura del waypoint final
L es el largo total de la curva entre ambos waypoints
$l(x, y))$ es el largo de la curva hasta el punto en donde se está calculando la altura.

13.2. Seguimiento de trayectorias

A continuación se expone el algoritmo de "Carrot chasing" de Sousa y Saripalli[24] en el cual se basa el presente proyecto para realizar el seguimiento.

Luego de hallada la curva que minimiza el recorrido entre dos waypoints mediante la planeación de trayectorias, se la discretiza para tener mayor control sobre el ángulo de Yaw que será el que se modifica en el siguiente algoritmo, por lo que se limitará el razonamiento a 2 sub-waypoints consecutivos de esta discretización, luego se extiende el programa concatenando pares de ellos hasta recorrer la totalidad de la curva.

En la siguiente figura[3] se observan los parámetros que se utilizan para definir el seguimiento:

- trayectoria deseada entre el waypoint W_i y el W_{i+1}
- p: la posición actual del robot
- q: la proyección ortogonal de p sobre la recta W_i, W_{i+1}
- S: el sub-waipoint al que se desea llegar
- ψ: el ángulo de Yaw actual del vehículo
- d: el "cross track error"[4] del vehículo
- ψ_d el ángulo de Yaw que se desea hallar para corregir la trayectoria

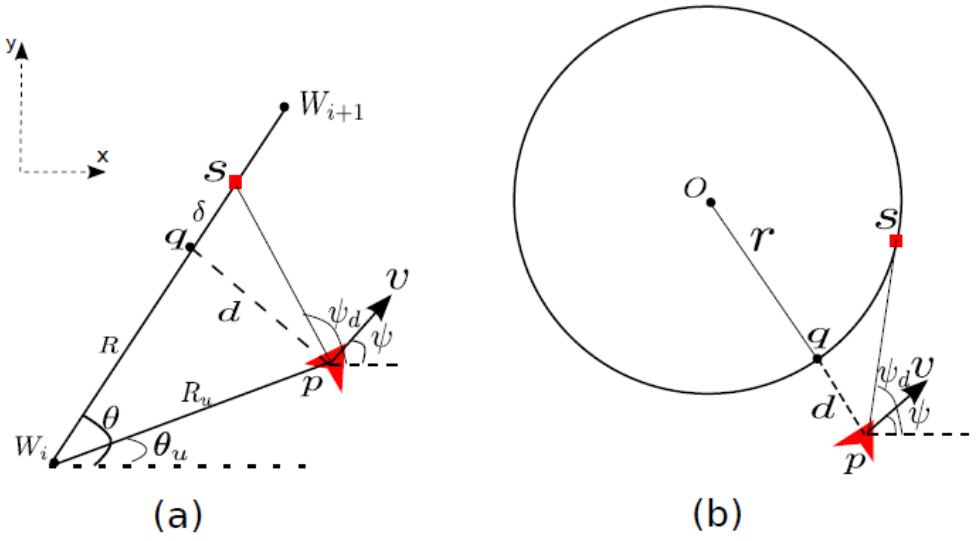


Figura 13.2: Seguimiento rectilíneo (a) y curvo (b)

[3]Extraída de [24]

[4]El error transversal desde la posición del vehículo hasta la trayectoria que debería seguir (de W_i a W_{i+1})

A continuación se expone el pseudo-código correspondiente para cada caso (seguimiento de trayectoria rectilínea o curva):

Trayectoria rectilínea:

1. inicialización: $W_i = (x_i, y_i),\ W_{i+1} = (x_{i+1}, y_{i+1}),\ p = (x, y),\ \psi$
2. $R_u = \|W_i - p\|,\ \theta = atan2(y_{i+1} - y_i, x_{i+1} - x_i)$
3. $\theta_u = atan2(y - y_i, x - x_i),\ \beta = \theta - \theta_u$
4. $R = \sqrt{R^2{}_u - (R_u sin\beta)^2}$
5. $(x'_t, y'_t) \longleftarrow ((R + \delta)cos\theta, (R + \delta)sin\theta)\ s = (x'_t, y'_t)$
6. $\psi_d = atan2(y'_t - y, x'_t - x)$
7. $u = k(\psi_d - \psi)$

Trayectoria curva:

1. inicialización: $O = (x_l, y_l), r, p, \psi$
2. $d = \|O - p\| - r$
3. $\theta = atan2(y - y_l, x - x_l)$
4. $(x'_t, y'_t) \longleftarrow (r\ cos(\theta + \lambda), r\ sin(\theta + \lambda))\ s = (x'_t, y'_t)$
5. $\psi_d = atan2(y'_t - y, x'_t - x)$
6. $u = k(\psi_d - \psi)$

Una vez determinado el ángulo de Yaw deseado ψ_d, se realiza este procedimiento para cada sub-waypoint de la trayectoria finalizando el proceso.

Capítulo 14

Control de movimiento

14.1. Problema del control de movimiento

Desde una perspectiva general, el problema del control de movimiento que se ataca tiene por objetivo seguir una *trayectoria*, que es una curva en el espacio y por ende define los puntos por donde debe pasar el cuadricóptero, sin imponer restricciones de tiempo, es por ello que la velocidad no está fijada y puede ser elegida como parámetro. Por otro lado, seguir la trayectoria implica (como se mencionó en el capítulo de seguimiento de trayectorias) volar en una dirección dada en cada instante. En resumen, la solución del problema planteado puede lograrse sin necesidad de tomar decisiones sobre todos los grados de libertad del cuadricóptero. En efecto, en el lenguaje de los ángulos de Euler elegidos, la orientación puede manejarse de modo que el ángulo de *roll* quede fijo en $0rad$ y el ángulo de *pitch* se mantenga en un valor constante, determinado por la velocidad de vuelo deseada. El ángulo de *yaw* ψ es el único que debe cambiar su valor deseado en cada loop de control. [1] En cuanto a la traslación, el grado de libertad que queda por controlar es el de posición z (altura).

Una vez establecidas estas restricciones, se puede proceder a la resolución del problema acotado reduciéndolo a la implementación de controladores adecuados. Para ello aún es necesario separarlo en sus partes angular y traslacional, que es lo que se explica en el capítulo siguiente.

14.2. Separación del problema

El cuadricóptero como sistema mecánico ha sido reducido en 5 a las ecuaciones de movimiento de un cuerpo rígido. Como consecuencia de ello, el sistema mecánico queda separado naturalmente en dos subsistemas: uno angular y otro traslacional. El subsistema angular está definido por la segunda cardinal en la forma 5.28, tiene como entrada el momento $\vec{M}$ y su salida es el estado de rotación, esto es, la velocidad angular dada por sus tres componentes en el sistema solidario al cuadricóptero $\vec{\omega} = (\omega_1, \omega_2, \omega_3)$ y la orientación dada por los ángulos de Euler y que se simbolizará por $\alpha = (\psi, \theta, \phi)$. El subsistema traslacional está definido por la primera cardinal, tiene como entrada la fuerza neta $\vec{F}^{neta}$ y su salida es el estado de movimiento en el sistema inercial de la tierra, dado por la velocidad $\vec{v} = (v_x, v_y, v_z)$ y la posición $\vec{r} = (x, y, z)$, en el sistema de coordenadas ENU (East, North, Up). El control del subsistema angular permite obtener una orientación dada del cuadricóptero, lo que

[1] Hay que observar que no cualquier ángulo de *pitch* es posible, puesto que a mayor ángulo menor el la fuerza neta en la dirección vertical y un ángulo muy pronunciado puede hacer imposible que el cuadricóptero no caiga.

permite fijar la dirección de vuelo, mientras que el control del subsistema traslacional permite controlar la velocidad y altura de vuelo.

14.3. Componentes del Sistema de Control

El esquema general de control está representado en la figura 14.1 y su funcionamiento es como sigue: un controlador de alto nivel se encarga, a partir de un algoritmo de seguimiento de trayectorias, de decidir una orientación $\alpha_d = (\phi_d, \theta_d, \psi_d)$ (determinada a partir de la dirección de vuelo deseada) y un estado traslacional deseado, que puede ser simplemente la altura de vuelo o también involucrar la velocidad.

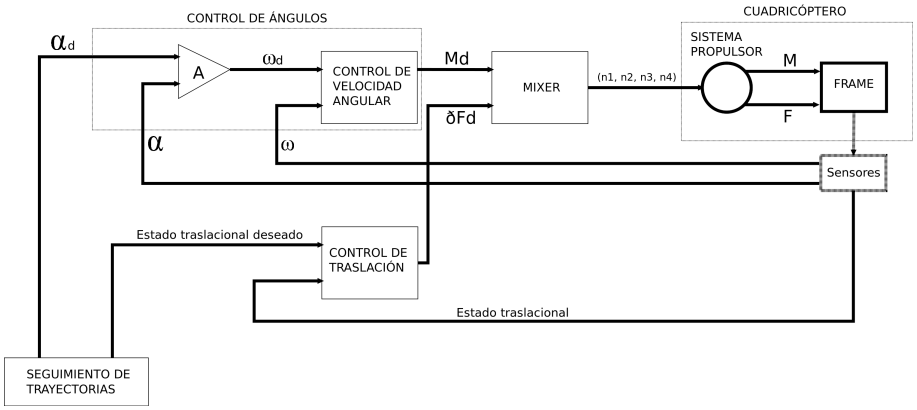


Figura 14.1: Diagrama del sistema de control

14.3.1. Control de Ángulos

El control de ángulos u orientación es la parte más importante de la tercera capa en la jerarquía ilustrada en la figura (12.1) y de todo el sistema en conjunto. Una buena parte del trabajo está relacionada con el diseño e implementación de esta componente. Se trata en detalle en (15).

14.3.2. Control de Translación

Como se dijo al final de (5), no es posible controlar completamente el estado del cuadricóptero actuando sobre las velocidades de rotación de los motores. El control de traslación sólo se encarga de calcular el empuje neto de los motores deseado para mantener una altura y dentro de ciertos límites, la velocidad de vuelo.

14.3.3. Cuadricóptero (planta)

El cuadricóptero se considera formado por el *frame* (cuerpo rígido) y un sistema propulsor. El sistema propulsor se compone de los ESC's, los motores y las hélices y tiene como entradas los niveles de acelerador de cada motor (n_1, n_2, n_3, n_4) codificados como anchos de pulso en PWM. Los anchos de pulso utilizados en los ESC's cumplen:

$$1060\mu s \leq n_i \leq 2000\mu s \qquad i = 1, 2, 3, 4$$

el valor inferior $n_i = 1060\mu s$ corresponde al motor detenido y el valor superior $n_i = 2000\mu s$ corresponde a la máxima velocidad posible. Los sensores fijos al frame junto con los algoritmos de fusión sensorial proporcionan la estimación del estado necesaria para los controladores.

14.3.4. Mixer

El bloque *Mixer* se encarga de tomar el momento deseado $\vec{M}_d$ y la fuerza neta ejercida por los propulsores deseada $\vec{F}_d$ y traducirlo a los comandos PWM n_1, n_2, n_3, n_4 para cada propulsor.

Si las entradas al mixer son nulas, las salidas se fijan en un n de base igual para todos los propulsores n_b, el cual se elige en la mitad del rango, o sea, $n_b = 1500\mu s$. Cuando se tienen entradas $\vec{M}_d$ y $\vec{F}_d$ se calculan en función de éstas las variaciones:

$$\begin{aligned} \delta n_i &= \delta n_s(\vec{M}_d) \qquad\qquad s = x, y, z \\ \delta n_F &= \delta n_F(\vec{F}_d) \end{aligned}$$

que de acuerdo a los sentidos de giro mostrados en la figura 14.2 modifican a los comandos de base n_b de la siguiente manera:

$$\begin{aligned} n_1 &= n_b + \delta n_F + \delta n_x - \delta n_y + \delta n_z \\ n_2 &= n_b + \delta n_F - \delta n_x - \delta n_y - \delta n_z \\ n_3 &= n_b + \delta n_F - \delta n_x + \delta n_y + \delta n_z \\ n_4 &= n_b + \delta n_F + \delta n_x + \delta n_y - \delta n_z \end{aligned} \tag{14.1}$$

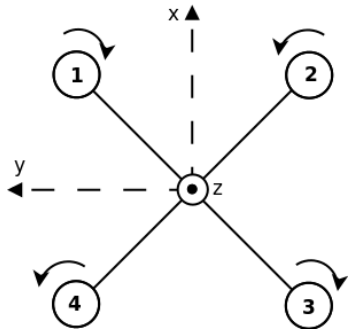


Figura 14.2: Ejes y sentidos de giro de las hélices

Las variaciones $\delta n_F(\vec{F}_d)$ contribuyen variando de forma colectiva el empuje de todos los motores, mientras que las variaciones $\delta n_s(\vec{M}_d)$ $s = x, y, z$ contribuyen al comando para cada motor con el signo adecuado según el momento ejercido por el motor en torno al eje correspondiente. Para calcular las variaciones se hace uso de las características *empuje - pulso* y *momento - pulso* obtenidas para el conjunto ESC-motor-hélice y mostradas en las figuras 14.3 y 14.4. Con el fin de simplificar la implementación en software se ha definido que la salida del bloque de control de translación de la figura 14.1 codifique directamente la fuerza total sobre el empuje de base (δF_d) como la variación δn_F, de modo que sólo resta obtener δn_x, δn_y y δn_z. El valor δn_F se usa primero para mover el punto de operación sobre la curva desde

$n_b = 1500\mu s$ al punto $n_b + \delta n_F$ y a continuación se linealiza la curva en torno a este nuevo punto para obtener δn_x, δn_y y δn_z. Los momentos de los motores en torno a los ejes x e y (M_x y M_y) de la figura 14.2 se obtienen cambiando el empuje que produce cada motor (a través de δn_x y δn_y), mientras que el momento en torno al eje z se obtiene variando el momento de cada motor (a través de δn_z). Estas variaciones siempre se realizan de a pares como indican las ecuaciones (14.1) para no cambiar la fuerza neta dada a través de δn_F. Las ecuaciones de las curvas características son:

$$\begin{aligned} F &= p_1 n^2 + p_2 n + p_3 && \text{empuje del motor} \\ \tau &= t_1 n^2 + t_2 n + t_3 && \text{momento del motor} \end{aligned}$$

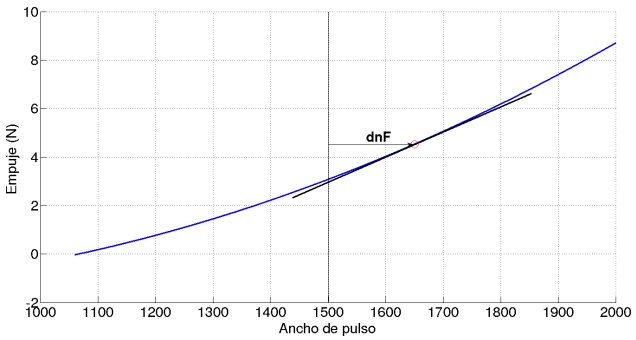


Figura 14.3: Linealización en la curva característica de empuje

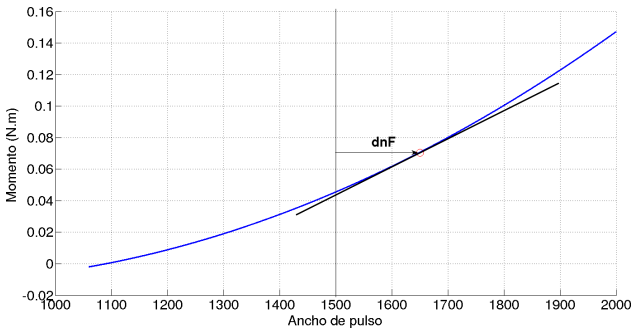


Figura 14.4: Linealización en la curva característica de momento

Calculando las derivadas y evaluando en el nuevo punto de operación $n_0 = n_b + \delta n_F$ se obtiene, ante variaciones δn:

$$\begin{aligned} M_x &= 4\,\delta F\, l = 4l(2p_1 n_0 + p_2)\delta n_x \\ M_y &= 4\,\delta F\, l = 4l(2p_1 n_0 + p_2)\delta n_y \\ M_z &= 4\,\delta M = 4(2t_1 n_0 + t_2)\delta n_z \end{aligned}$$

siendo l el brazo de momentos de los motores respecto al centro de masa. A partir de estas ecuaciones se despeja:

$$\delta n_x = \frac{M_x}{4l(2p_1 n_0 + p_2)}$$
$$\delta n_y = \frac{M_y}{4l(2p_1 n_0 + p_2)}$$
$$\delta n_z = \frac{M_z}{4(2t_1 n_0 + t_2)}$$

Para evitar que ante una entrada al mixer de momentos deseados $\vec{M}_d$ se sobrepasara el límite de $2000\mu s$ en la salida, se impuso la limitación siguiente en los valores de δn:

$$-100 \leq \delta n_i \leq 100 \qquad\qquad i = x, y, z \tag{14.2}$$

y la restricción sobre los valores de δn_F a la entrada:

$$-100 \leq \delta n_F \leq 200 \tag{14.3}$$

En el cuadro (14.1) se ha evaluado el efecto de variar δn en los rangos 14,2 cuando el punto de operación está un uno y otro extremo del rango especificado en 14.3.

δn_F	$4l(2p_1 n_0 + p_2)$	$4(2t_1 n_0 + t_2)$	δM_{xy}	δM_z
200	9×10^{-3}	$0{,}772 \times 10^{-3}$	$\pm 0{,}9$	$\pm 0{,}0772$
-100	$6{,}8 \times 10^{-3}$	$0{,}528 \times 10^{-3}$	$\pm 0{,}68$	$\pm 0{,}0528$

Tabla 14.1: Rangos del mixer

Con los rangos de variación de los momentos obtenidos se fijó el rango correspondiente de entradas admisibles para evitar saturar inadvertidamente a los motores:

$$-100 \leq \delta n_F \leq 200 \tag{14.4}$$
$$-0{,}50\, N.m \leq M_{xy} \leq 0{,}50\, N.m \tag{14.5}$$
$$-0{,}05\, N.m \leq M_z \leq 0{,}05\, N.m \tag{14.6}$$

Capítulo 15

Soluciones para los controladores

15.1. Control Digital y Frecuencia de Muestreo

El sistema de control se implementa en forma digital [27], de modo que las variables de entrada y salida al mismo son números con una precisión finita y las acciones se modifican a una frecuencia de muestreo f_s. Un esquema general de un sistema de control digital se muestra en la figura (15.1)

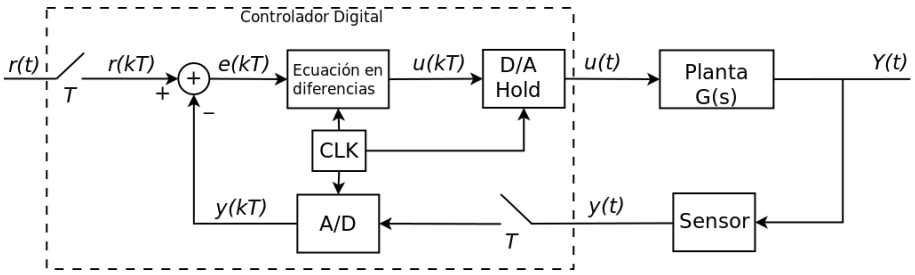


Figura 15.1: Diagrama de un sistema de control digital.

Una computadora digital no puede integrar. Para resolver una ecuación diferencial ésta debe ser priemro aproximada, reduciéndola a una ecuación algebraica que involucre sólo sumas y productos. Es por ello que en el caso discreto el controlador queda implementado como una ecuación en diferencias. Para el diseño del mismo existen escencialemnte dos caminos:

- Emulación: Consiste en aproximar una transferncia continua $C(s)$ (controlador) con un conjunto de ecuaciones en diferencias. Hay efectos básicos que surgen al llevar a cabo esta aproximación, como la introducción de retardo (esto es general para los controladores digitales) que deben ser tenidos en cuenta.
- Métodos directos: Diseño directamente en el dominio digital, sin recurrir a una aproximación. Las técnicas directas proporcionan en general el método más exacto.

En este trabajo fue utilizado el método de emulación para un controlador PID (además de una red neuronal). La principal ventaja de este método es que los parámetros de los controladores pueden ser interpretados según la influencia que tienen en el controlador continuo de partida. Esto es especialmente cierto para un controlador tan común como el PID.

La aproximación del controlador digital consiste en aplicar una discretización a las ecuaciones diferenciales. Para controladores lineales esto se especifica eligiendo el mapeo entre las variables en el dominio de la transformada de Laplace de $C(s)$ (s) y el dominio de la transformada Z del controlador discretizado $D(z)$. Una de las correnspondencias más comunes son:

- Euler hacia adelante: $s \to \frac{z-1}{T_s}$
- Euler hacia atrás: $s \to \frac{1-z^{-1}}{T_s}$
- Trapezoidal: $s \to \frac{2}{T_s}\frac{z-1}{z+1}$

Las más sencillas son las dos primeras. En el caso de una derivada primera la aproximación de Euler hacia atrás para $\dot{x}(t)$es:

$$\frac{dx}{dt} \simeq \frac{x(t) - x(t - T_s)}{T_s} \tag{15.1}$$

Esta transformación tiene la ventaja de que es muy simple y conserva la estabilidad, ya que el semiplano izquierdo en s es mapeado a una región en el interior del círculo unidad en z. Esta propiedad es compartida con la discretización trapezoidal[27].

La elección de la tasa de muestreo (f_s) es en general un compromiso entre varios aspectos del diseño ([27], [28]). Razones para disminuir f_s son el acotamiento del ruido de cuantización, y la reducción del costo del hardware. Por otro lado, entre las principales razones para aumentar f_s están el dotar al sistema de mayor capacidad para rechazar perturbaciones aleatorias en la planta, reducir el retardo asocciado al muestreo, reducir el retardo en la respuesta y hacer que el sistema capaz de seguir señales de entrada con cierto ancho de banda (según el teorema del muestreo). En [27] (cap. 11) y en [28] se recomienda elegir la tasa de muestreo en el rango:

$$20 f_b < f_s < 40 f_b$$

siendo f_b el ancho de banda del sistema en lazo cerrado con un controlador continuo al cual se discretiza. El límite superior está dado por la degradación del error rms en la salida debido al error de cuantización, en microcontroladores con palabras de menos de 12 bits y trabajando en punto fijo. Para procesadores de 32 o más bits resulta ser en general que, a las tasas más altas que alcanzan, la contribución del error de cuantización es despreciable. Debido al desconocimiento de las características del ruido de la planta y del ancho de banda del sistema, se optó por determinar primero una f_s aceptable para el controlador PID discretizado y luego utilizar esa tasa para la red neuronal. Puesto que el microcontrolador utilizado permite números en simple precisión de 32 bits, no se consideró un problema el ruido de cuantización. En un pricipio se utilizó una tasa $f_s = 100Hz$ ($T_s = 0{,}01s$), lo que resultó en errores muy grandes en el seguimiento de una referencia constante de ángulo de orientación. Debido a ello se decidió elevar la tasa de muestreo. Los ESC's empleados aceptan señales PWM de entrada de hasta $500Hz$, de modo que se adoptó como nueva frecuencia de muestreo $f_s = 400Hz$. El período asociado $T_s = 0{,}0025s$ es representable exactamente en punto flotante, lo cual es conveniente porque se lo utiliza directamente en los algoritmos de estimación del estado.

15.2. Cálculo de la velocidad angular deseada

El control de ángulos mostrado en el diagrama (14.1) se hace como cascada de dos bloques. El primero, indicado como una ganancia A, se encarga de calcular una velocidad angular deseada a partir de la diferencia entre los ángulos deseados y los medidos por los sensores. En el caso de ángulos de *pitch* y *roll* la salida es:

$$\omega_{1d} = A_{PR}(\theta_d - \theta) \tag{15.2}$$
$$\omega_{2d} = A_{PR}(\phi_d - \phi) \tag{15.3}$$

Debido a la simetría del cuadricóptero se toma el mismo valor de ganancia (A_{PR}) para ambos ángulos. En el caso del *yaw* por otro lado, la velocidad angular deseada debe ser tal que el giro del cuadricóptero sea recorriendo el menor ángulo entre la dirección deseada y la actual, según se muestra en la figura 15.2.

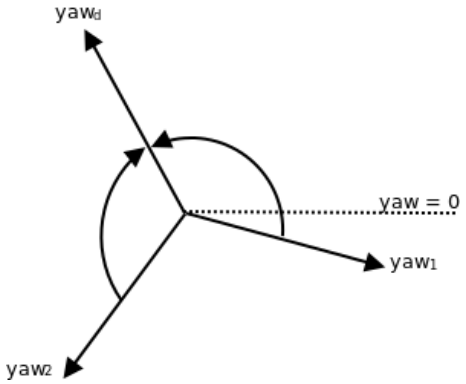


Figura 15.2: Sentidos de giro apropiados para el yaw

Esto se logra tomando una función adecuada de la diferencia $\psi_d - \psi$:

$$\omega_{3d} = A_Y\, \gamma(\psi_d - \psi)$$

la función γ proporciona el menor ángulo (con signo) entre dos direcciones y se puede obtener utilizando la función módulo. Si los ángulos se toman en el intervalo $[0, 2\pi)rad$ y $\Delta = \psi_d - \psi$:

$$\gamma(\Delta) = \quad \text{mód}\ (\Delta + \pi, 2\pi) - \pi \tag{15.4}$$

La ganancia para el *yaw* se denota por A_Y y no es necesariamente igual a la ganancia para los otro ángulos, A_{PR}.

A continuación se encuentra el segundo bloque, que se encarga de controlar la velocidad angular según la referencia $\vec{\omega}_d$ proporcional al error en los ángulos, ya calculada. La razón por la que se ha adoptado esta separación en bloques es que el sistema que debe controlar el controlador de velocidad angular es más sencillo desde el punto de vista de la dinámica. Este sistema tiene como entrada los momentos $\vec{M}$ y como salida la velocidad angular $\vec{\omega}$ y está determinado por la segunda cardinal (ecuación 5.28) y es por tanto de primer orden. Esto lo hace además más accesible el modelado por redes neuronales.

El controlador de velocidad angular fue implementado mediante un controlador PID y un controlador basado en redes neuronales, lo que se explica en los dos apartados siguientes.

15.3. Control de velocidad angular

15.3.1. Control mediante un PID

El controlador Proporcional - Integral - Derivativo (PID) es un controlador de lazo cerrado usado comúnmente cuya salida se compone de tres términos obtenidos a partir del error entre un valor deseado o *setpoint* y la variable que se busca controlar. En el caso continuo, si $y(t)$ es la variable a controlar e y_d es el setpoint, el error en el instante t es:

$$e(t) = y_d \, - \, y(t)$$

la salida del PID es la acción de control $u(t)$ y se calcula como una suma ponderada de términos proporcionales a $e(t)$, su integral y su derivada:

$$u(t) \, = \, P_0 \, e(t) \, + \, I_0 \int_0^t e(\tau) \, d\tau \, + \, D_0 \, \frac{de(t)}{dt}$$

donde P_0, I_0 y D_0 son constantes ajustables.

En el caso discreto, el término integral y el derivativo se sustituyen por sus aproximaciones discretas:

$$u(t_N) \, = \, P_0 \, e(t_N) \, + \, I_0 \sum_{\tau_n=0}^{t_N} e(\tau_n) dT \, + \, D_0 \, \frac{e(t_N) - e(t_{N-1})}{T_s}$$

siendo T_s el intervalo de tiempo entre pasos (período de muestreo).

Como en el problema presente la variable a controlar es el vector velocidad angular $\vec{\omega}$, las ecuaciones anteriores tienen tres componentes o, en forma equivalente, se deben llevar durante la operación tres controladores unidimensionales, uno por cada eje. Debido a la simetría del cuadricóptero, las constantes para los ejes de pitch y roll se pueden elegir iguales, siendo distintas las correspondientes al eje de yaw, se tendrá entonces seis constantes a ajustar:

$$P_{0P} = P_{0R} \qquad P_{0Y} \tag{15.5}$$

$$I_{0P} = I_{0R} \qquad I_{0Y} \tag{15.6}$$

$$D_{0P} = D_{0R} \qquad D_{0Y} \tag{15.7}$$

$$\tag{15.8}$$

Un problema a resolver en las aplicaciones como esta, donde la acción de control que se puede aplicar a la planta es acotada (porque los motores pueden producir empujes y momentos acotados) es el del *windup* [20], [35]. El windup se da cuando la acción de control alcanza la saturación, a partir de ese momento el término integral continua aumentando (y por tanto la acción de control requerida por el controlador) sin que los actuadores (motores en este caso) puedan ejercer esa acción de control. Si

durante esa condición el setpoint cambia a un valor menor, la salida seguirá saturada mientras el integrador no haga descender el término integral a valores adecuados. Esto provoca un retardo indeseable en la respuesta al salir de la condición de windup. Para remediar este problema [37] se restringió el valor que puede tomar el término integral $I = I_0 \sum_{\tau_n=0}^{t_N} e(\tau_n)dT$ dentro de un cierto rango:

$$-MAX_I \leq I \leq MAX_I$$

lo que es equivalente a detener la integración cuando se llega a un extremo del rango, evitando así el windup.

Los valores de la salida del PID de cada eje se limitaron al rango $[-1000, 1000]$ y los del término integral al rango $[-500, 500]$ ($MAX_I = 500$). Estos rangos se mapean a la entrada del mixer a los rangos (14.4) elegidos para los momentos. Para verificar el funcionamiento se construyó un modelo en Simulink (archivo PIDtest_model.slx) que incorpora el modelo hecho para el cuadricóptero, las funciones que implementan los bloques descritos (bloque $\vec{\omega}_d$, PID por eje y mixer) y bloques adecuados para poder manipular los setpoints y las constantes del controlador.

El paso de tiempo elegido para el PID fue $dT = 0{,}01s$, que es la tasa máxima a la que la IMU puede entregar medidas de orientación y velocidad angular, mientras que el paso de integración del modelo se fijó en $10^{-3}s$. Para adecuar estos distintos intervalos se utilizan bloques de muestreo (ZOH - Zero Order Hold) en las conexiones entre las salidas de velocidad angular y orientación del modelo y el PID. El modelo fue utilizado también para realizar el ajuste de las constantes, ya que permite observar las características de la respuesta del sistema controlado.

En la gráfica 15.3 se muestra el resultado obtenido para los setpoints $roll = 0{,}3\,rad$, $pitch = 0\,rad$ y $yaw = -0{,}4\,rad$. En la gráfica 15.4 se muestra el resultado obtenido para los setpoints $roll = 0{,}3\,rad$, $pitch = -0{,}4\,rad$ y $yaw = 5{,}5\,rad$.

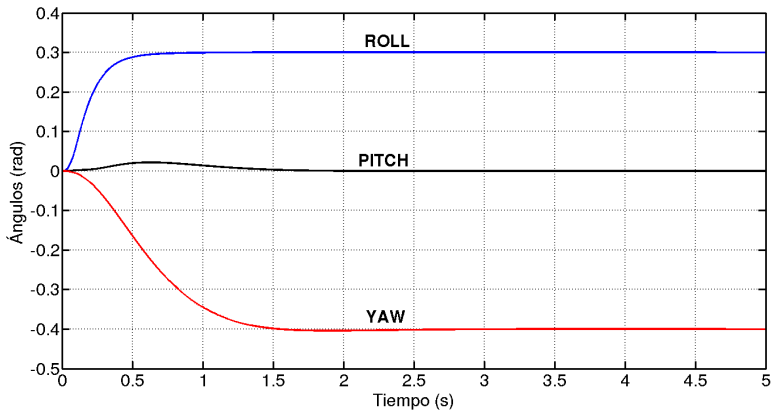


Figura 15.3: Respuesta del sistema para $roll = 0{,}3\,rad$, $pitch = 0\,rad$ y $yaw = -0{,}4\,rad$

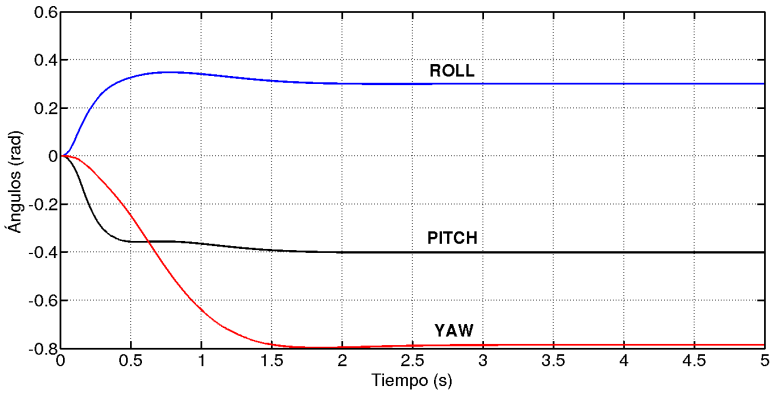


Figura 15.4: Respuesta del sistema para $roll = 0{,}3\,rad$, $pitch = -0{,}4\,rad$ y $yaw = 5{,}5\,rad$

En el modelo el ángulo inicial es siempre $0\,rad$, al imponer un setpoint en el yaw de $5{,}5\,rad$ el movimiento es hacia los ángulo negativos y alcanza el valor final $-(2\pi - 5{,}5\,rad) = -0{,}78\,rad$ que verifica el funcionamiento correcto de la implementación de la función (15.4).

15.3.2. Control con redes neuronales

Control con Red Neuronal única

Para realizar el control de velocidad angular mediante redes neuronales se estudiaron primero diversas alternativas y se eligió la que se consideró más conveniente, dado el problema y las diversas limitaciones que se tienen. A continuación se exponen las alternativas consideradas con comentarios sobre sus ventajas y desventajas.

1. Control por modelo inverso

   En este esquema se utiliza una red neuronal dinámica para aproximar el sistema inverso, mostrado en la figura 15.5.

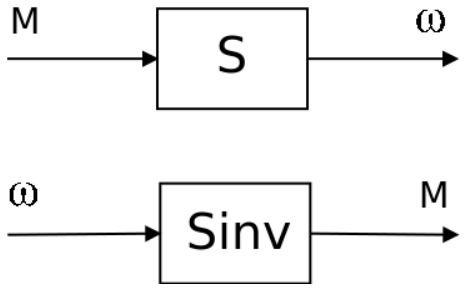


Figura 15.5: Sistemas directo e inverso

   El conjunto de datos de entrenamiento consiste obviamente de entradas $\vec{\omega}$ y objetivos $\vec{M}$, que se pueden generar a partir del modelo directo excitando el modelo del cuadricóptero con una señal de momentos suficientemente variada. La aplicación de esta estrategia depende de que el sistema tenga una inversa definida y tiene la desventaja de que proporciona un controlador en lazo abierto, que no aprovecha las medidas de velocidad angular de la IMU.

2. Control por predicción a partir de un modelo (Model Predictive Control - MPC)

   Este controlador usa un modelo de la planta a controlar para predecir respuestas futuras ante entradas de control potenciales. Con estas predicciones, un algoritmo de optimización calcula las entradas que optimizan el desempeño de la planta de acuerdo a algún criterio. El diagrama de la figura 15.6 ilustra el funcionamiento de este controlador. La variable u' es la acción tentativa de control, y_d es el setpoint e y_m es la respuesta del modelo de red neuronal.

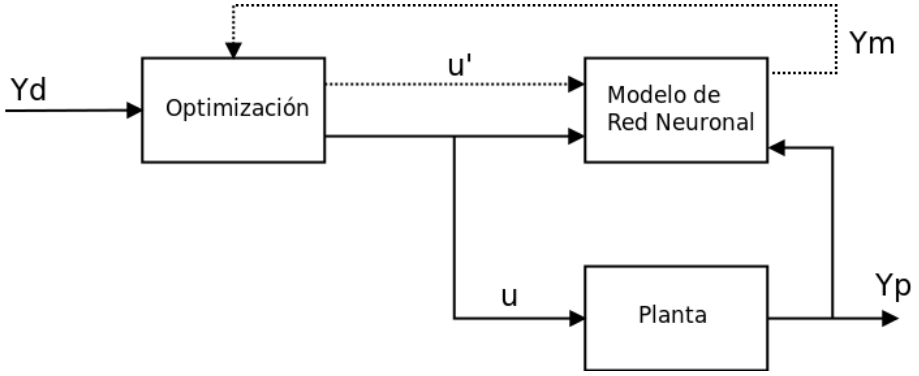


Figura 15.6: Diagrama del controlador MPC

El criterio de optimización consiste en general en minimizar una función de pago con la forma siguiente:

$$J = \sum_{j=N_1}^{N_2} [y_d(t+j) - y_m(t+j)]^2 + \rho \sum_{j=1}^{N_u} [u'(t+j-1) - u'(t+j-2)]^2 \tag{15.9}$$

donde N_1, N_2 y N_u definen los horizontes sobre los que el error y los incrementos de control son evaluados. El parámetro ρ ajusta el peso que los errores en las entradas tienen en la función J.

La principal desventaja de este esquema es que tiene un requerimiento computacional muy pesado, puesto que en cada paso se debe correr un algoritmo de minimización para la función J, lo que implica varias evaluaciones de la respuesta de la red. Esta alternativa fue descartada considerando el poder computacional del que se disponía.

3. Control por modelo de referencia (Model Reference Control - MRC)

Esta arquitectura utiliza una red neuronal como controlador entrenada de modo que la respuesta del sistema controlado sea igual a la de un modelo de referencia elegido de antemano. Para explicar claramente la operación del controlador en este caso conviene describir el proceso de construcción del controlador, cual consiste escencialmente de tres pasos:

Paso 1) Se identifica la planta entrenando una red neuronal que sirve de modelo (al igual que en el caso de MPC), según se ejemplifica en la figura 15.7

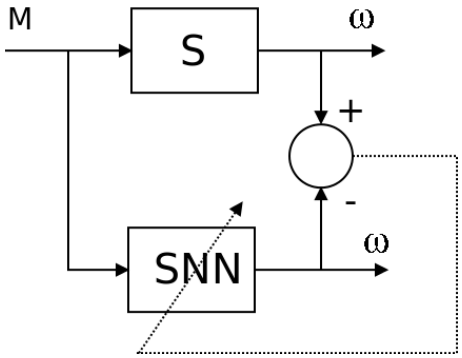


Figura 15.7: Paso 1 de la construcción de MRC

Paso 2) Con la red entrenada en el paso anterior se construye una nueva red agregando los nodos de la red que será el controlador (cNN en la figura 15.8). A continuación se entrena la red controlador (cNN) de modo que la repuesta de la red total ante un conjunto de setpoints $\vec{\omega}_d$ siga a un modelo de referencia elegido (Sref) .

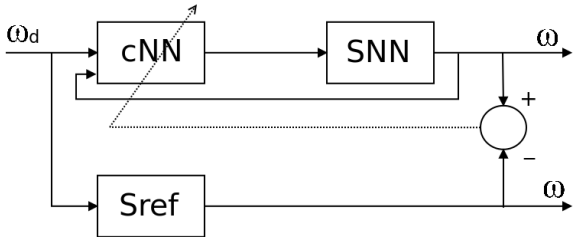


Figura 15.8: Paso 2 de la construcción de MRC

Paso 3) Se construye una red con estructura idéntica a la red cNN y se le asignan los pesos de ésta obtenidos en el entrenamiento del paso 2. El trabajo de diseño consiste escencialmente en la elección del número de capas y número de neuronas por capa de las redes, así como la elección del sistema de referencia Sref. Según la bibliografía consultada, es común elegir como referencia sistemas lineales e invariantes en el tiempo.

Esta arquitectura presenta la ventaja de que proporciona un controlador en lazo cerrado y que sólo requiere evaluar la red cNN por cada paso de control, lo que resulta en una carga computacional no muy elevada bajo la condición de realizar el controlador con un número bajo de neuronas.

La arquitectura elegida fue la de control por modelo de referencia. Para la identificación (paso 1 del proceso de construcción) se hizo un modelo Simulink (archivo tdata_generation_quad_angular_model.slx) en el cual se alimenta al modelo del cuadricóptero construido anteriormente con señales de momento ($\vec{M}$) aleatorias, con un intervalo entre cambios de 0,01s, que es el menor intervalo de tiempo en el que, como se explicó en la descripción del controlador PID, se tienen medidas nuevas de la IMU y por tanto es el menor intervalo del loop de control esperado. Como arquitectura de red se utilizó una red NARX con dos retardos en la entrada y realimentación. En la figura 15.9 se muestra la configuración según la notación de Matlab. Para el entrenamiento se utiliza la configuración serie-paralelo en la que no se usa la red realimentada, sino que se aprovechan los datos objetivo $y(t)$ a la entrada, en lugar de las salidas de la red.

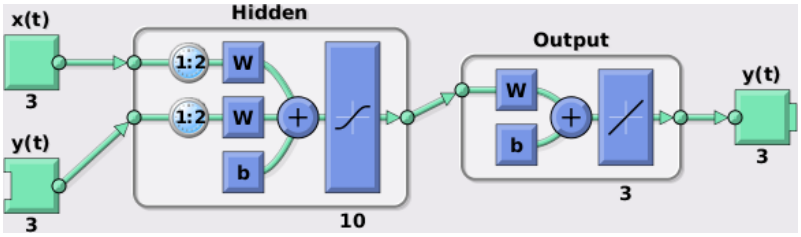


Figura 15.9: Red utilizada para la identificación

Una vez entrenada, esta red se puede incluir en el modelo Simulink para testearla respecto a la planta con una señal de entrada nueva. Previamente a ello se la debe modificar cerrando el loop de la salida a la entrada de *feedback* (la entrada inferior en la figura (15.9), esto se hace con el comando *coseloop* de Matlab. En las gráficas (15.10), (15.11) y (15.12) se muestra un test de la red entrenada frente a una entrada aleatoria de momento durante 50s. Los errores cuadráticos medios son para estos test: $mse_1 = 2{,}7\,rad/s$, $mse_2 = 0{,}66\,rad/s$ y $mse_3 = 1{,}6\,rad/s$

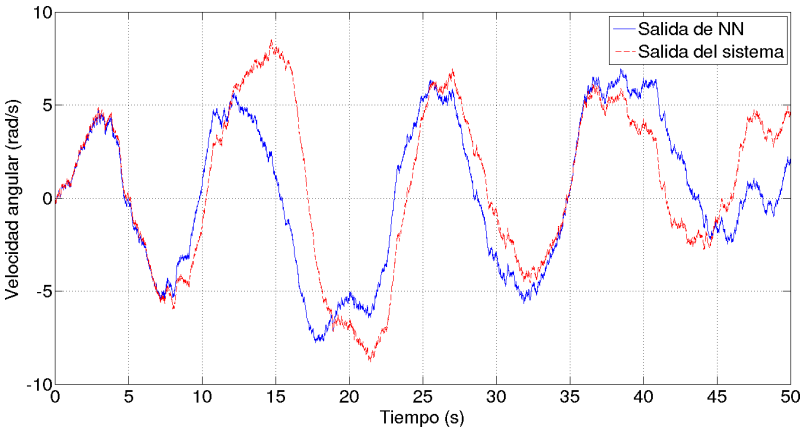


Figura 15.10: Testeo de la red entrenada, componente ω_1

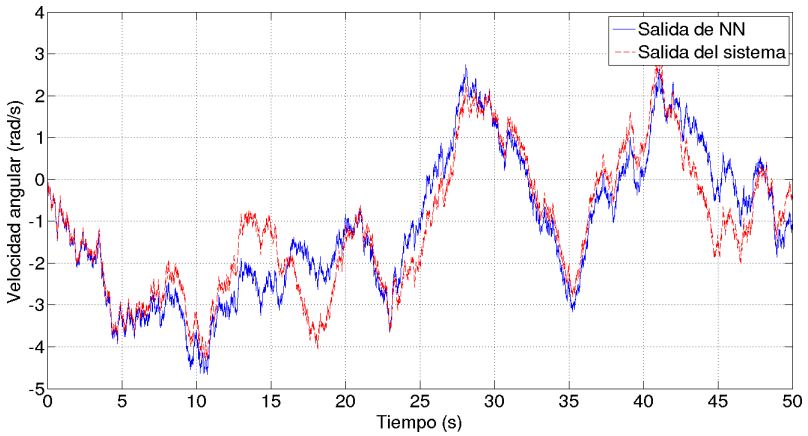


Figura 15.11: Testeo de la red entrenada, componente ω_2

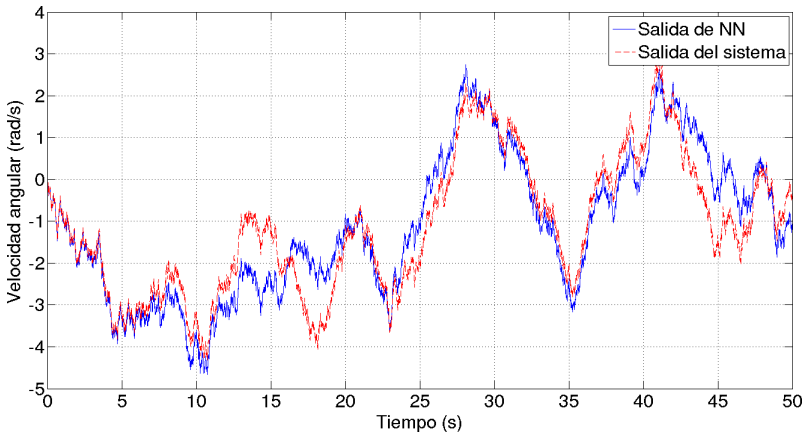


Figura 15.12: Testeo de la red entrenada, componente ω_3

Para el paso 2 se tomó una red controlador con la arquitectura NARX, con cinco neuronas en la capa oculta. La red total se muestra en la figura 15.13. En esta nueva red se debe inhibir el aprendizaje de los pesos correspondientes a la red con la que se ha identificado el sistema, esto se logra en Matlab seteando la bandera *learn* de los pesos a '0'.

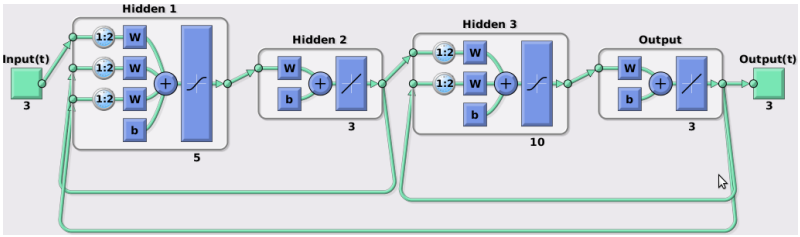


Figura 15.13: Red total para el entrenamiento del controlador

El modelo de referencia tomado es el de un sistema lineal de primer orden para cada eje, siendo idénticos los modelos para los ejes de *pith* y *roll*. Observando las respuestas del sistema controlado con el PID en el tiempo, se eligieron las siguientes transferencias y constantes de tiempo como referencia:

$$H_{PR}(s) = \frac{1}{\tau_{PR}s + 1} \qquad \tau_{PR} = 0{,}2s \qquad \text{Pitch y Roll} \tag{15.10}$$

$$H_{Y}(s) = \frac{1}{\tau_{Y}s + 1} \qquad \tau_{Y} = 0{,}7s \qquad \text{Yaw} \tag{15.11}$$

Los sistemas fueron implementados en un modelo Simulink que calcula las respuestas ante señales de entrada (archivo tdata_generation_quad_angular_refmodel.slx). Para generar un conjunto de datos entrada-salida de entrenamiento se tomaron como entradas escalones aleatorios cada 1s con valores entre $-10\,rad/s$ y $10\,rad/s$ para los sistemas de los ejes de *pitch* y *roll*. Para el eje de *yaw* los escalones se tomaron cada 3s y con valores entre $-3\,rad/s$ y $3\,rad/s$. Estos parámetros fueron elegidos, al igual que las constantes de tiempo, por observación del sistema controlado por el PID. La duración de las señales de entrenamiento fue de 60s. En las figuras (15.14), (15.15) y (15.16) se muestran los resultados de las respuestas de la red total (figura 15.13) contrastadas con las señales de entrenamiento. Los errores cuadráticos medios son: $mse_1 = 0{,}094\,rad/s$, $mse_2 = 0{,}088\,rad/s$ y $mse_3 = 0{,}034\,rad/s$

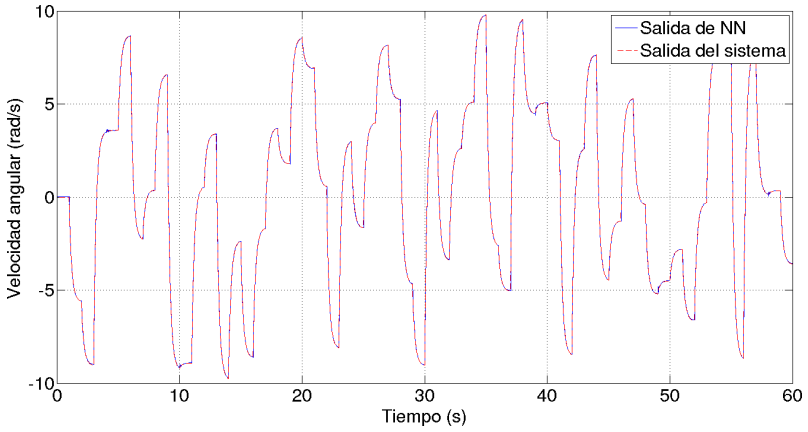


Figura 15.14: Resultado del entrenamiento de la red total, componente ω_1

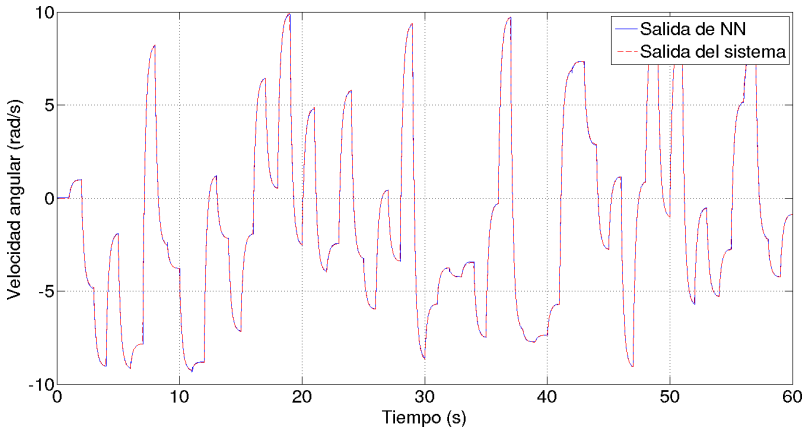


Figura 15.15: Resultado del entrenamiento de la red total, componente ω_2

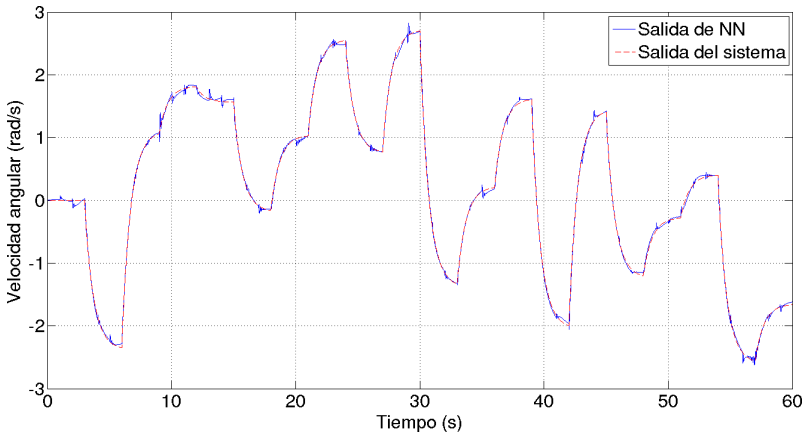


Figura 15.16: Resultado del entrenamiento de la red total, componente ω_3

Para el paso 3 finalmente se debe construir una red de igual estructura que la red controlador y asignarle los pesos correspondientes en la red total obtenidos en el entrenamiento.

Control con una red por eje

La idea es sintetizar el sistema que da la correspondencia momento - velocidad angular mediante una red neuronal, pero sin utilizar un algoritmo de entrenamiento para determinar los pesos, sino elegirlos explícitamente para que la red reproduzca algún procedimiento de integración numérica de las ecuaciones diferenciales. Previo a ello, es posible hacer una simplificación en las ecuaciones dinámicas 5.28:

$$\begin{cases} I_x\dot{\omega}_1 = (I_y - I_z)\omega_2\omega_3 + M_x \\ I_y\dot{\omega}_2 = (I_z - I_x)\omega_1\omega_3 + M_y \\ I_z\dot{\omega}_3 = (I_x - I_y)\omega_1\omega_2 + M_z \end{cases} \tag{15.12}$$

dados los valores típicos de las magnitudes se tiene que $|I_i - I_j| \sim 10^{-3} kg\, m^2$ y $\omega_i\omega_j \sim (10ras/)^2$, siendo $M \sim 1Nm$ se procedió a despreciar en las ecuaciones dinámicas del cuerpo rígido los términos del tipo $(I_i - I_j)\omega_i\omega_j$ frente a M. Esto resulta en las ecuaciones:

$$\begin{cases} I_x\dot{\omega}_1 = M_x \\ I_y\dot{\omega}_2 = M_y \\ I_z\dot{\omega}_3 = M_z \end{cases} \tag{15.13}$$

Estas ecuaciones simplificadas se implementan como mediante red neuronal, para ello se parte del esquema de integración sencillo dado por el método de Euler:

$$\omega_i(t + dt) = \omega_i(t) + \frac{M_i}{I_i}\, dt \tag{15.14}$$

que se puede implementar como un perceptrón solo realimientando la velocidad angular con un retardo de un paso, como se muestra en la figura (15.17). Para este perceptrón, el peso correspondiente a la entrada 1 (Input 1) es $W_1 = \frac{M_i}{I_i}$, el correspondiente a la realimientación de ω es 1 y el bias es $b = 0$.

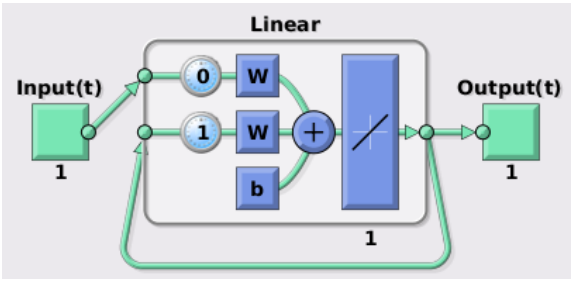


Figura 15.17: Perceptrón que implementa el método de Euler.

A fin de verificar que este método aproximado de integración proporciona buenos resultados para las entradas típicas que se tienen, se simuló en *Matlab* la salida frente a varias entradas de momento M. En la figura (15.18) se muestra a modo de ilustración el resultado para la entrada $M(t) = t\ \sin(t)$. Las gráficas de la respuesta calculada y simulada son prácticamente indistinguibles.

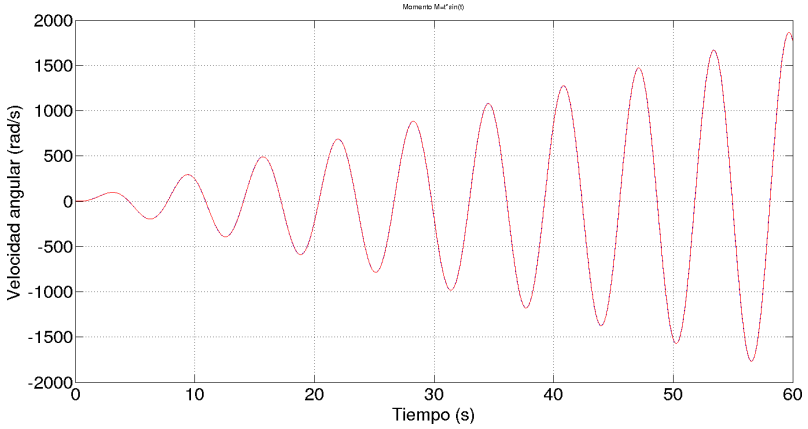


Figura 15.18: Simulación del método de Euler implementado con un perceptrón.

Con los perceptrones así construidos (uno por eje) se construyen sendas redes que constan cada una del controlador a entrenar seguido por un perceptrón de este tipo. El esquema de las mismas se muestra en la figura (15.19).

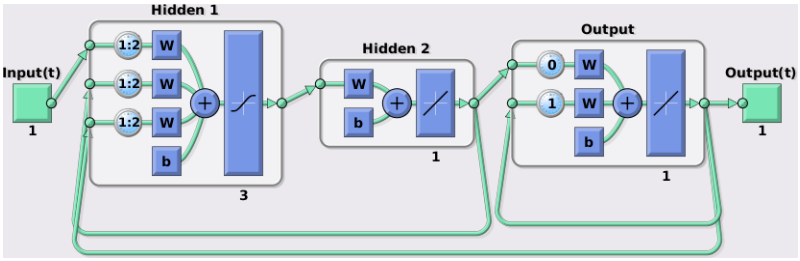


Figura 15.19: Red construida para entrenar el controlador.

Se ha elegido como controlador una red sencilla con tres neuronas en la capa de entrada, función de activación sigmoidal en la capa de entrada y lineal en la de salida. La red total simula el sistema angular controlado y tiene como entradas las velocidades angulares deseadas $vec\omega_d$ y como salida la velocidad angular del cuadricóptero $\vec{\omega}$, ambas según el eje modelado. Para el entrenamiento de esta red se utilizó el patrón de entradas mostrado en la figura (15.20) que está compuesto por escalones de ancho y altura variables fue construido de modo de cubriera toda la región relevante de operación del sistema.

En la figura (15.21) se muestra el resultado para un test de la red entrenada, utilizando como entrada una serie de escalones, esta vez de ancho fijo. Durante la misma simulación se extrajo la salida de momento que produce la red controlador, la cual es mostrada en la figura (15.22). De estos resultados se desprende que los momentos producidos se encuentran en un rango adecuado para ser aplicados al cuadricóptero.

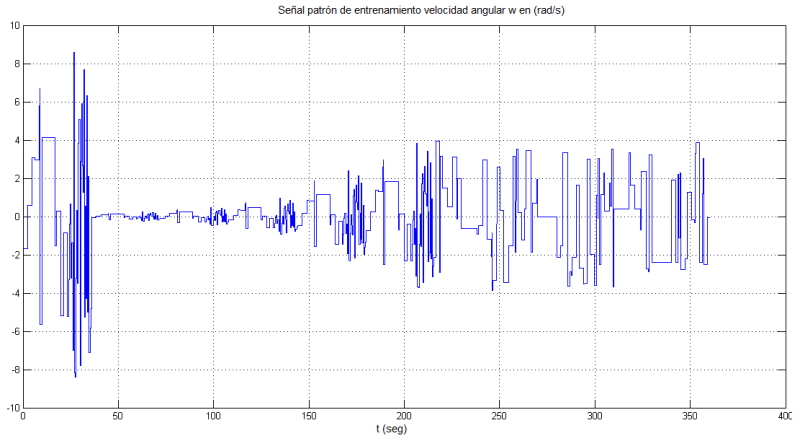


Figura 15.20: Patrón de entrenamiento para las redes por eje.

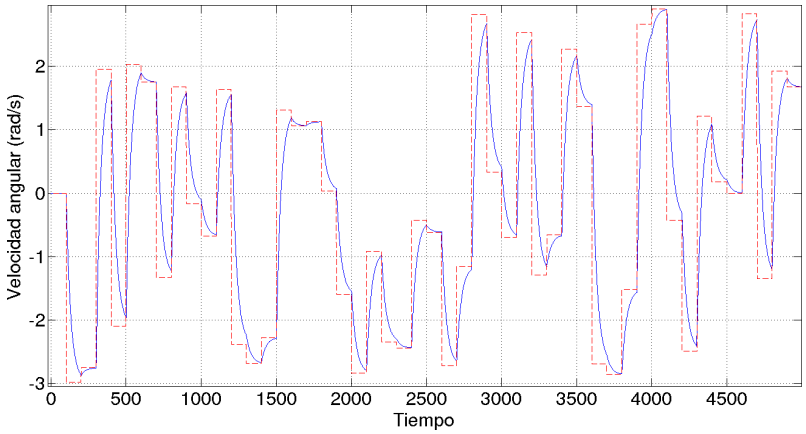


Figura 15.21: Resultado del test de la red entrenada. Tiempo en unidades de $s \times 10^{-2}$.

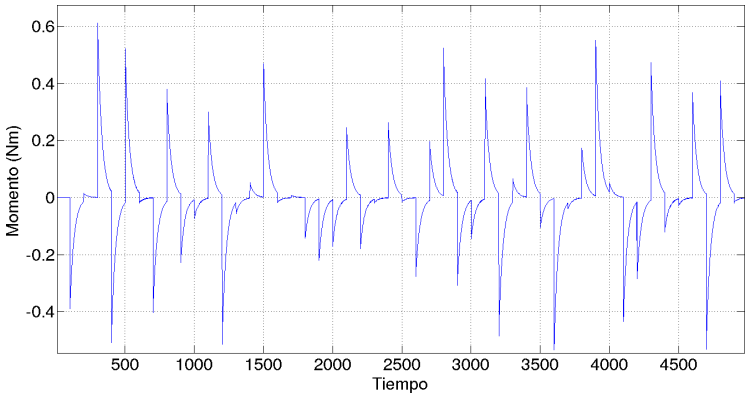


Figura 15.22: Momentos producidos por la red entrenada. Tiempo en unidades de $s \times 10^{-2}$.

15.4. Hover

En el modo de vuelo de *hover* [1] el cuadricóptero debe controlar sus grados de libertad de traslación para permanecer en un lugar del espacio establecido como referencia. Para describir el problema se utilizan los sistemas de coordenadas defini-

[1] El término en inglés *hover* puede ser traducido como "flotar" y se usa específicamente para denotar la acción de permanecer en el aire en un lugar determinado.

dos previamente en la figura (5.1) y que se muestran en la figura (15.23), el sistema $S_A(X, Y, Z)$ es el fijo a la tierra y el sistema $S_B(x, y, z)$ el solidario al cuadricóptero. Es conveniente recordar que el empuje total producido por los motores tiene la dirección y sentido del eje z:

$$\vec{E} = E\hat{z}$$

la fuerza neta está compuesta por este empuje, el peso y la resistencia del aire:

$$\vec{F}^{neta} = \vec{E} + m\vec{g} + \vec{F}_d$$

donde el empuje es la única fuerza sobre la que se puede actuar para efectuar el control.

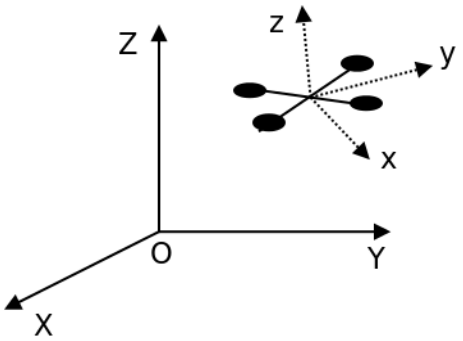


Figura 15.23: Sistemas de referencia fijo a la tierra y fijo al cuadricóptero.

El objetivo es controlar la altura (Z) y la posición horizontal (X e Y) a unos valores deseados (X_d, Y_d, Z_d) mientras se mantiene un ángulo de yaw (orientación) dado ψ_d. En la figura (15.24) se muestra la geometría del problema, (X, Y) es la posición del cuadricóptero y ψ la orientación (del control de la orientación se encarga directamente el bloque de control de ángulos en el diagrama 14.1). La estrategia es desplazar al cuadricóptero en el sentido indicado en cada instante por el vector error de posición:

$$\vec{D} = (X_d - X, Y_d - Y) \tag{15.15}$$

y para ello es necesario inclinar el cuadricóptero de modo tal que el empuje $\vec{E}$ tenga una componente horizontal paralela a $\vec{D}$, o lo que es equivalente, hacer que el versor $\hat{z}$ asociado al eje z del sistema solidario al vehículo tenga una componente horizontal paralela a $\vec{D}$.

La matriz de cambio de coordenadas del sistema de la tierra (S_A) al del cuadricóptero (S_B) en términos de los ángulos de Euler es 5.1:

$${}^{B}_{A}R = \begin{pmatrix} \cos\theta\cos\psi & \operatorname{sen}\psi\cos\theta & -\operatorname{sen}\theta \\ \cos\psi\operatorname{sen}\theta\operatorname{sen}\phi - \operatorname{sen}\psi\cos\phi & \operatorname{sen}\psi\operatorname{sen}\theta\operatorname{sen}\phi + \cos\psi\cos\phi & \cos\theta\operatorname{sen}\phi \\ \cos\psi\operatorname{sen}\theta\cos\phi - \operatorname{sen}\psi\operatorname{sen}\phi & \operatorname{sen}\psi\operatorname{sen}\theta\cos\phi - \cos\psi\operatorname{sen}\phi & \cos\theta\cos\phi \end{pmatrix} \tag{15.16}$$

la inversa de esta matriz (igual a su traspuesta) realiza el cambio de coordenadas del sistema del cuadricóptero (S_B) al de la tierra (S_A):

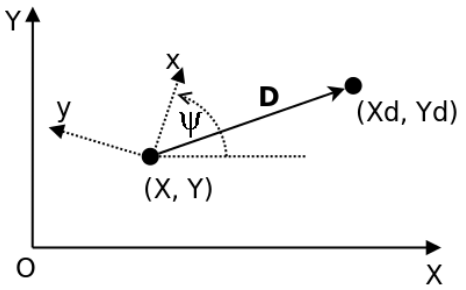


Figura 15.24: Geometría del problema de hover.

$$ {}^{A}_{B}R = {}^{B}_{A}R^{T} = \begin{pmatrix} \cos\theta\cos\psi & \cos\psi\,\mathrm{sen}\,\theta\,\mathrm{sen}\,\phi - \mathrm{sen}\,\psi\cos\phi & \cos\psi\,\mathrm{sen}\,\theta\cos\phi + \mathrm{sen}\,\psi\,\mathrm{sen}\,\phi \\ \mathrm{sen}\,\psi\cos\theta & \mathrm{sen}\,\psi\,\mathrm{sen}\,\theta\,\mathrm{sen}\,\phi + \cos\psi\cos\phi & \mathrm{sen}\,\psi\,\mathrm{sen}\,\theta\cos\phi - \cos\psi\,\mathrm{sen}\,\phi \\ -\mathrm{sen}\,\theta & \cos\theta\,\mathrm{sen}\,\phi & \cos\theta\cos\phi \end{pmatrix} \tag{15.17} $$

las componentes del empuje $\vec{E}$ en el sistema de la tierra son entonces:

$$ {}^{A}\vec{E} = {}^{A}_{B}R\;{}^{B}\vec{E} = E\;{}^{A}_{B}R \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = E \begin{pmatrix} \cos\psi\,\mathrm{sen}\,\theta\cos\phi + \mathrm{sen}\,\psi\,\mathrm{sen}\,\phi \\ \mathrm{sen}\,\psi\,\mathrm{sen}\,\theta\cos\phi - \cos\psi\,\mathrm{sen}\,\phi \\ \cos\theta\cos\phi \end{pmatrix} \tag{15.18} $$

A partir de estas componentes es posible calcular las componentes de la aceleración en el sistema absoluto. Debido a que en el modo de hover las velocidades de desplazamiento son pequeñas, se desprecia la resitencia del aire, luego:

$$ a_X = \frac{E_X}{m} = \frac{E}{m}(\cos\psi\,\mathrm{sen}\,\theta\cos\phi + \mathrm{sen}\,\psi\,\mathrm{sen}\,\phi) \tag{15.19} $$

$$ a_Y = \frac{E_Y}{m} = \frac{E}{m}(\mathrm{sen}\,\psi\,\mathrm{sen}\,\theta\cos\phi - \cos\psi\,\mathrm{sen}\,\phi) \tag{15.20} $$

las inclinaciones (*pitch* y *roll*) se limitan al rango de pequeños ángulos ($\theta, \phi << 1rad$), esto permite aproximar las funciones trigonométricas de esos ángulos en la expresión anterior:

$$ a_X \simeq \frac{E}{m}(\theta\cos\psi + \phi\,\mathrm{sen}\,\psi) \tag{15.21} $$

$$ a_Y \simeq \frac{E}{m}(\theta\,\mathrm{sen}\,\psi - \phi\cos\psi) \tag{15.22} $$

inversamente, dadas unas componentes en S de aceleración en el plano horizontal a_X y a_Y, los ángulos necesarios para producirlas son:

$$ \theta \simeq \frac{m}{E}(a_X\cos\psi + a_Y\,\mathrm{sen}\,\psi) \tag{15.23} $$

$$ \phi \simeq \frac{m}{E}(a_X\,\mathrm{sen}\,\psi - a_Y\cos\psi) \tag{15.24} $$

Resta relacionar las componentes de aceleración a_X y a_Y con el estado del cuadricóptero. Una primera aproximación al problema es hacer que sean proporcionales al error en la posición $\vec{D}$ de (15.15):

$$a_X = K_h(X_d - X) \tag{15.25}$$
$$a_Y = K_h(Y_d - Y) \tag{15.26}$$

sustituyendo en las ecuaciones para los ángulos resulta:

$$\theta \simeq \frac{m}{E} K_h \left[(X_d - X)\cos\psi + (Y_d - Y)\,\mathrm{sen}\,\psi\right] \tag{15.27}$$
$$\phi \simeq \frac{m}{E} K_h \left[(X_d - X)\,\mathrm{sen}\,\psi - (Y_d - Y)\cos\psi\right] \tag{15.28}$$

El orden de magnitud de la constante K_h se puede estimar si se fija el orden de la aceleración horizontal deseada. Se supone que para errores en posición del orden del metro la aceleración sea del orden de $1\,m/s^2$:

$$|X_d - X|,\ |Y_d - Y| \sim 1m \qquad \Rightarrow \qquad |a_X|,\ |a_Y| \sim 1m/s^2$$

esto asegura que el desplazamiento en un intervalo de tiempo de $1s$ es del orden de $\frac{1}{2}a\Delta t^2 \sim 0{,}5m$. Según (15.25), la constante de proporcionalidad es del orden:

$$K_h \sim 1\,s^{-2}$$

En el modo de hover el controlador de altura se encarga de modificar el empuje neto para mantener la referencia Z_d. Debido a que los ángulos de *pitch* y *roll* se mantienen pequeños, el empuje es siempre aproximadamente igual al peso, $E = mg$. En estas condiciones los ángulos calculados mediante las expresiones (15.27) verifican (con $|X_d - X|,\ |Y_d - Y| \sim 1m$):

$$\theta,\ \phi \sim \frac{m}{E}\, K_h\, 1m = \frac{1}{g}\, K_h\, 1m \;=\; 0{,}1\,rad$$

luego, se verifica la hipótesis de ángulos pequeños.

15.5. Control de traslación

El control de traslación implementa un controlador de altura comol algoritmo PID en el error de altura $e = z - z_d$. La salida de este controlador es el valor δn_F de variación colectiva del comando a los propulsores. Este valor es el que toma como entrada el bloque Mixer según se explicó en 14.3.4 y modifica el empuje neto de los cuatro motores:

$$|\vec{E}| = |\vec{F_1} + \vec{F_2} + \vec{F_3} + \vec{F_4}|$$

Opcionalmente es posible computar el ángulo de *pitch* θ de modo que la componente horizontal del empuje $E_x = E\,\mathrm{sen}\,\theta$ iguale en magnitud a la fuerza de rozamiento del aire, modelada como $-b|\vec{v}|$, para tratar de alcanzar una velocidad de vuelo v_d:

$$\theta = \text{asin}\left(\frac{bv_d}{mg}\right)$$

donde la velocidad v_d es fijada por el usuario en las capa de planeación.

Capítulo 16

Resultados

16.1. Introducción

Los controladores desarrollados fueron ensayados experimentalmente en el cuadricóptero, con el objetivo de verificar su funcionamiento y comprobar además que el sistema completo diseñado y construido funcionase correctamente. Para las pruebas en laboratorio se agregó al cuadricóptero una pieza que provee dos ejes de suspensión (para los ángulos de *pitch* y *roll*), los cuales se apoyan en una horquilla por medio de rulemanes, el sistema se ilustra en la figura (16.1).

Figura 16.1: Cuadricóptero en el dispositivo de suspensión.

Esta horquilla permite a su vez el giro según el eje vertical (ángulo de *yaw*) mientras mantiene al cuadricóptero colgado. De este modo se eliminan los grados de libertad de traslación y un grado de libertad de rotación, quedando solamente los ángulos de *pitch* (o *roll*) y *yaw*, que puede ser también eliminado si se impide el giro del dispositivo según la vertical.

En las pruebas realizadas se sometió al sistema a una entrada de ángulos deseados formada por un serie de escalones, cado uno con una duración de 10 a 15 segundos. Para estas pruebas se escribió una versión simplificada del programa de control en

un *sketch* en lenguaje *Arduino* para el microcontrolador Tiva C, que se encarga de generar los escalones, leer el flujo de datos de estado proporcionado por el microcontrolador Teensy y realizar el loop de control. Los valores de las variables de estado relevantes (ángulos de Euler y velocidades angulares), así como los comandos a los motores fueron transmitidos a una PC via un puerto serie y almacenados. El usuario puede enviar comandos de arranque de la prueba y finalizar la ejecución en cualquier instante. Este programa puede utilizar indistintamente el PID o una red neuronal como controlador.

A continuación se muestran los resultados obtenidos para las pruebas con el PID, la red controlador con salidas para todos los ejes y las redes controlador individuales para cada eje. Como se menciona al final de (15.1) se ensayaron dos frecuencias de muestreo distintas: $f_s^{(1)} = 100Hz$ ($T_s = 10ms$) y $f_s^{(2)} = 400Hz$ ($T_s = 2{,}5ms$). Para la primera de ellas fue utilizada la IMU (3.3.1), mientras que para la frecuencia más alta fue utilizada la (3.3.1), capaz de entregar una tasa de lecturas mayor. Las redes neuronales fueron entrenadas por separado para cada una de las frecuencias de muestreo.

En todos los casos de pruebas de ángulos el valor de punto de operación de base de los motores fue $n = 1400us$.[1]

16.2. PID

Para el controlador de ángulos basado en un PID los resultados de *pitch* frente a una entrada escalón son los mostrados en la figura (16.2), para la frecuencia de muestreo $f_s = f_s^{(1)} = 100Hz$. Para esta prueba fueron utilizados los parámetros [2]:

$$\begin{aligned} A &= 2{,}1 \\ P &= 360 \\ I &= 400 \\ D &= 60 \\ MIXER\_RANGE\_PR &= 0{,}5 \end{aligned}$$

En el caso del ángulo de *yaw* los resultados están en la figura (16.3), también para $f_s = f_s^{(1)} = 100Hz$.

[1] Recuérdese que el rango calibrado en los ESCs es $1060us - 2000us$.

[2] Se debe observar para la comprensión de estas constantes que el bloque *mixer* mapea los valores entregados por el controlador a un rango dado por la constante MIXER_RANGE_PR para *pitch* y *roll* y por la constante MIXER_RANGE_YAW para el *yaw*.

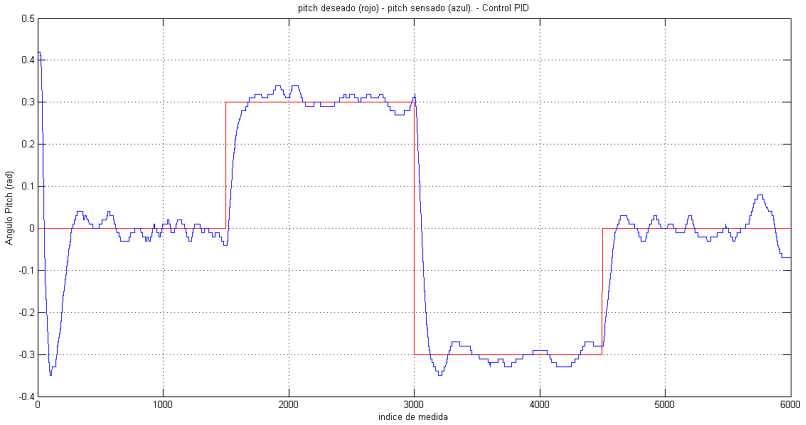


Figura 16.2: Resultados de respuesta a escalones en *pitch* para el PID ($f_s^{(1)} = 100Hz$). Cada unidad del índice de medida corresponde a $10^{-2}s$.

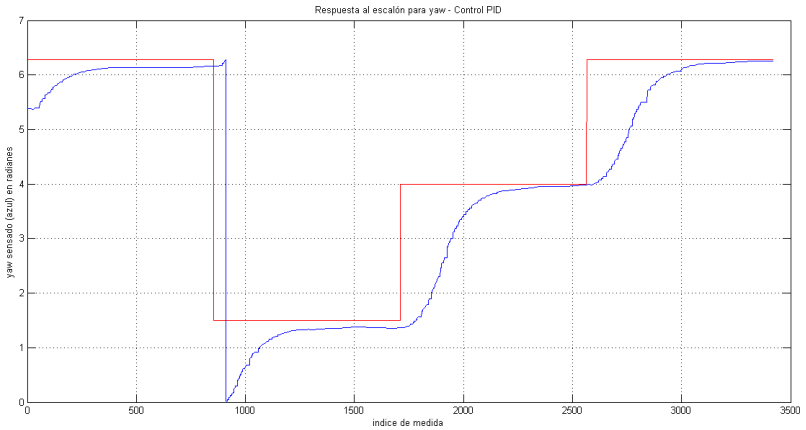


Figura 16.3: Resultados de respuesta a escalones en *yaw* para el PID ($f_s^{(1)} = 100Hz$). Cada unidad del índice de medida corresponde a $10^{-2}s$.

Con los mismos parámetros que las pruebas anteriores, pero ahora con $f_s = f_s^{(2)} = 400Hz$, los resultados para escalones en *pitch* se muestran en la figura (16.4).

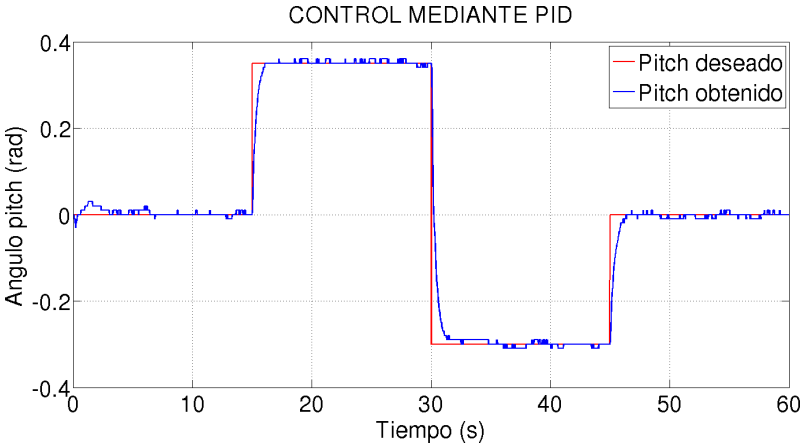


Figura 16.4: Resultados de respuesta a escalones en *pitch* para el PID ($f_s^{(2)} = 400Hz$).

16.3. Red neuronal individual por eje

Los resultados, con entradas similares a las utilizadas con los controladores anteriores, son los de la figura (16.5) para $f_s = f_s^{(1)} = 100Hz$. Los parámetros en este caso son:

$$A = 2{,}1$$
$$CoefMom = 1{,}5$$

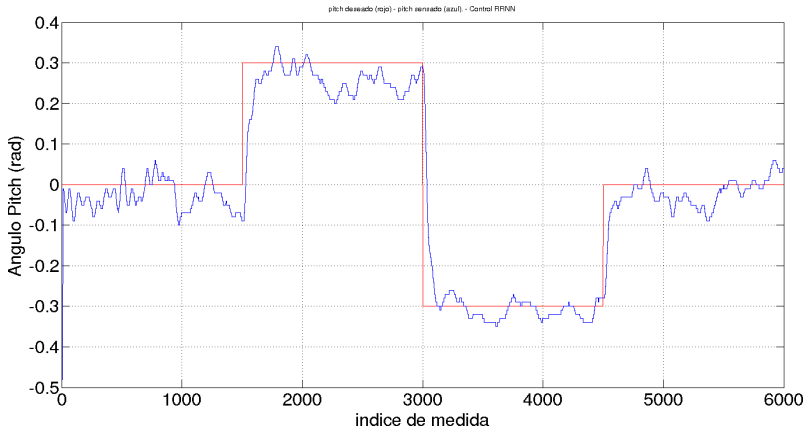


Figura 16.5: Resultados de respuesta a escalones en *pitch* de la red individual por eje ($f_s^{(1)} = 100Hz$). Cada unidad del índice de medida corresponde a $10^{-2}s$.

Mientras que con $f_s = f_s^{(2)} = 400Hz$, los resultados para escalones en *pitch* se muestran en la figura (16.6).

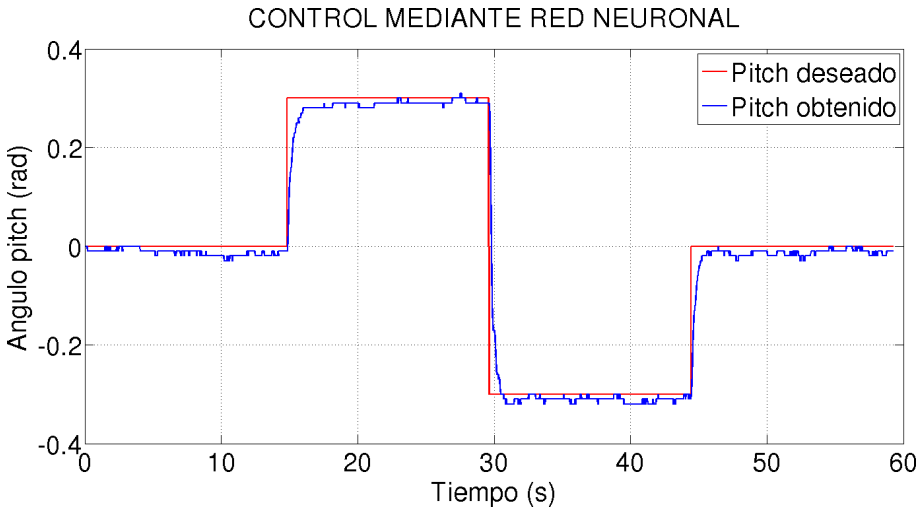


Figura 16.6: Resultados de respuesta a escalones en *pitch* de la red individual por eje ($f_s^{(1)} = 100Hz$).

16.4. Red neuronal con ejes combinados

Para la red neuronal que controla los tres ejes en forma combinada el resultado frente a escalones en *pitch* se muestra en la figura (16.7). El parámetro de ganancia

de ángulos A para esta prueba fue:

$$A = 2{,}1$$

Durante las pruebas se estudió el resultado de agregar una ganancia de momentos a la salida de la red neuronal (que está entrenada para que sus salidas sean momentos). Se observó que con ganancias en el intervalo $1 - 2$ se puede mejorar la respuesta del sistema, además se vio que ganancias muy altas (de 3 en adelante aproximadamente) lo desestabilizaban. El resultado mostrado corresponde a una ganancia de momentos de 1.7 (constante CoefMom en el *sketch*).

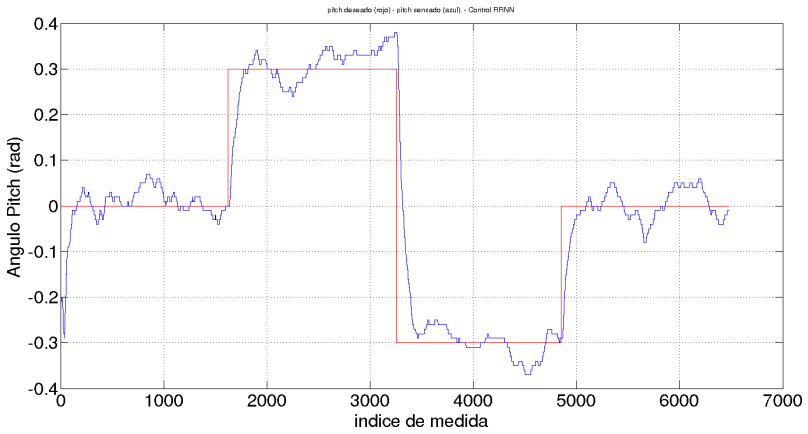


Figura 16.7: Resultados de respuesta a escalones en *pitch* de la red para tres ejes. Cada unidad del índice de medida corresponde a $10^{-2}s$.

En la fase de pruebas se decidió mantener los controladores PID y con redes individuales por eje, lo que simplifica el diseño. Por ello no se realizaron pruebas para la red neuronal combinada con $f_s = f_s^{(2)} = 400Hz$.

16.5. Prueba de altura

Habiendo corroborado el adecuado funcionamiento del control de orientación, se procedió a realizar una prueba del sistema controlando los tres grados de libertad físicos de orientación y la altura. Esta prueba fue realizada dentro del laboratorio, por lo que se desarrolló a baja altura y sin la posibilidad de controlar la posición XY, debido a no estar disponible el GPS. El ensayo consistió en hacer seguir al cuadricóptero rampas de altura, a la vez que controlaba la orientación con referencias cero para todos los ángulos. El resultado se presenta en la figura 16.8.

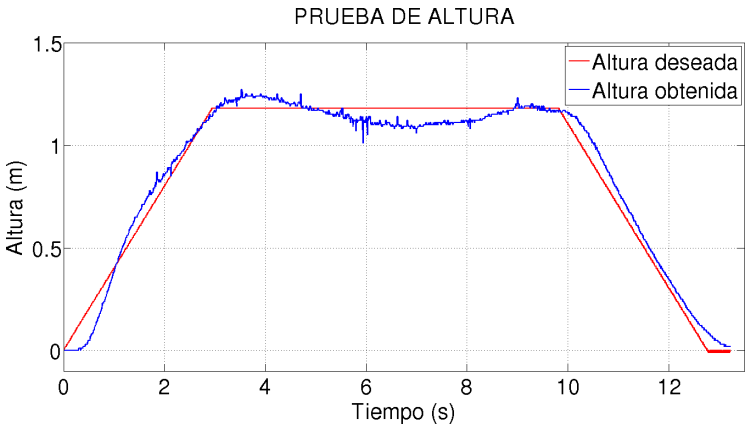


Figura 16.8: Prueba de altura.

16.6. Observaciones

Durante los experimentos se investigaron los resultados de controladores PID y red neuronal por eje con un período de loop de $20ms$ en lugar de los $10ms$ de período de los controladores mostrados aquí. Para ello se debió realizar nuevamente todo el proceso de definición de red modelo y entrenamiento de las redes controlador, mientras que en el caso del PID el intervalo de tiempo del loop es un parámetro de la función de la librería que calcula la salida y no hay que cambiar nada. En ambos casos (red y PID) se observó que no es posible controlar al sistema, resultando en movimientos muy bruscos del cuadricóptero y debiendo detener las pruebas para evitar posibles roturas, sin recabar series relevante de datos. También se observó que las redes entrenadas para un período de loop de $10ms$ no se comportan de manera satisfactoria si se aumenta este período. Basados en estas observaciones y en algunas referencias consultadas [38], [39] se decidió elevar la frecuencia de muestreo, lo que implicó el uso de la IMU 3.3.1.

Parte VI

Implementación en software

Implementación en software.

Capítulo 17

Software

En esta sección se describen la estructura y el funcionamiento del software desarrollado para las dos plataformas de procesamiento utilizadas. En una de ellas (Teensy 3.2) corre el software encargado de la lectura de todos los sensores y la fusión sensorial. En la otra plataforma (Tiva C) corre el software encargado del control. La descripción se hará por separado para los programas de ambas plataformas, sin embargo todo el software desarrollado responde a ciertas características generales comunes:

- Las librerías utilizadas están escritas en lenguaje C++ o en lenguaje *Arduino*.
- Las librerías se integran en un programa principal, en ambos casos escrito en lenguaje *Arduino*.
- Los programas principales se compilan utilizándo entornos de desarrollo (IDE's) similares al de Arduino [40]. En el caso del programa para la plataforma Teensy, se utiliza un *plugin* para el IDE de Arduino [41]; mientras que para la plataforma Tiva C se utiliza el IDE *energia* [42].

17.1. Teensy

El software que corre en la plataforma Teensy se encarga de realizar la lectura de los sensores y ejecutar el filtro de Kalman para estimar la parte traslacional del estado $(\vec{r}, \vec{v})$. El programa consiste de un *loop* principal que se ejecuta cada 10 ms, que es el intervalo de tiempo en el cual la IMU proporciona datos nuevos de orientación y velocidad angular, así como de aceleración. El diagrama de flujo de este programa se muestra en la figura (17.1). Luego del inicio se entra en un loop principal en el cual se leen primero de la IMU el cuaternión (quat, cuatro componentes) que proporciona la orientación y el giróscopo, que proporciona la velocidad angular (gyro, tres componenetes), ambas lecturas son obtenidas en forma *cruda*, con 16 bits por cada componente de las magnitudes medidas. A continuación se lee la UART correspondiente a la comunicación serial con el GPS y se decodifican estos datos, esto es, se extraen de la trama de datos del protocolo del GPS los datos útiles de posición geográfica.

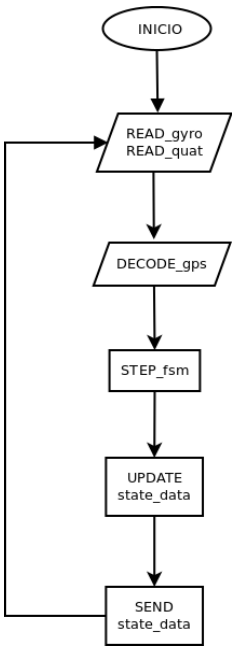


Figura 17.1: Diagrama de flujo del programa de sensores y estimación de estado.

Luego se pasa a leer el barómetro, el cual tiene un retardo de 16 ms entre que se le envía un comando de lectura y tiene el dato disponible para ser leído, esto es así tanto para la medida de presión como para la de temperatura (necesaria para aplicar la corrección de la presión por temperatura). El GPS por otro lado, proporciona datos cada 100 ms, por lo que la estimación del estado traslacional, que implica tener medidas de altura y posición xy puede hacerse cada 100 ms. Para manejar los dos períodos distintos, se implementó una máquina de estados para la lectura del barómetro y fusión sensorial para obtener $(\vec{r}, \vec{v})$, que está representada en la figura (17.2). En cada loop de 10 ms se hace un salto de estado en la máquina, si corresponde. En el estado TRIG_T se envía al barómetro el comando de lectura de temperatura y se almacena el instante de inicio de la máquina, a continuación en el estado READ_T_TRIG_P se espera a completar el retardo $\Delta t_{baro} = 16ms$, se lee la temperatura y se envía el comando de lectura de presión. En el estado READ_P se vuelve a esperar un retardo Δt_{baro} y se lee la presión para terminar calculando la

altura (h) en el estado CALC_h. En el estado READ_ACC se obtiene la aceleración de la IMU (con el vector de aceleración de la gravedad $\vec{g}$ removido) y el estado RUNKF corre un paso del filtro de Kalman. Finalmente hay un estado de espera para completar los 100 ms. Esta realización del proceso de lectura del barómetro como máquina de estados resulta ser bastante flexible en cuanto a la posibilidad de variar la duración del loop principal, porque si es necesario reducir el tiempo de loop es posible agregar más estados que surgan de la separación de las operaciones que se llevan a cabo en uno de ellos en varias subtareas. En este caso, por ejemplo, es posible asignar un estado de la máquina a la ejecución de cada paso del filtro de Kalman, en lugar de correrlo enteramenten en un solo paso.

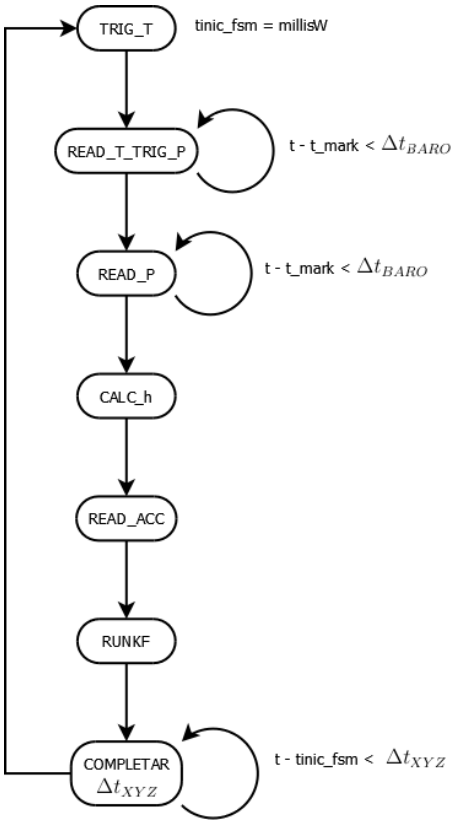


Figura 17.2: Máquina de estados para lectura del barómetro y estimación de $(\vec{r}, \vec{v})$.

Las distintas tareas del programa principal se llevan a cabo utilizando las funciones implementadas en las librerías de C++ escritas para el proyecto o adaptadas. El siguiente es un resumen de la funcionalidad de cada una de ellas.

- **Lectura del barómetro** (BaroMS5637): Esta librería proporciona funciones para solicitar al barómetro las medidas de temperatura y presión y para leer estas medidas de los registros correspondientes mediante el protocolo I2C. Contiene también las funciones para calcular la presión en Pa a partir de los datos crudos y hacer la correción por temperatura. Implementa la función de promediado de medidas para fijar la referencia de presión al inicio.
- **Lectura de la IMU** (SensorBNO055): Se encarga de leer por I2C los datos de orientación (dado por un cuaternión q), velocidad angular y aceleración de la IMU. Incorpora una función para grabar en los registros de calibración de la IMU las constantes en el archivo *calibracion.h*.

- **Lectura del GPS** (GPS_NMEA): Lee los datos provenientes del GPS a través de un puerto serial y hace la conversión de latitud/longitud a coordenadas cartesianas. También realiza el promediado de medidas durante la inicialización para fijar el origen de coordenadas cartesianas.
- **Lectura del sensor de ultrasonido** (HC_SR04): Solicita una medida del sensor de ultrasonido habilita las interrupciones en el pin *echo* del sensor. El sensor genera una interrupción cuando retorna la señal de ultrasonido, que es atendida por esta librería para proporcionar la distancia.
- **Filtro de Kalman** (kfilter): Implementa el algoritmo del filtro de Kalman según se ha explicado. Las matrices del filtro se encuentran definidas en el archivo *kfilter_matrices3d_norm.h*.
- **Utilidades matemáticas** (imumaths, matrix, quaternion, vector): Contienen diversas funciones auxiliares para trabajar con objetos geométricos (vectores y cuaterniones) y matrices. Se utilizan entre otras cosas para llevar a cabo la rotación de las componentes de la aceleración desde el sistema de referencia de la IMU (solidario al cuadricóptero) al sistema de la tierra.
- **Transferencia serial de datos** (EasyTransfer): Se utilizó esta librería (disponible en [44]) para hacer la transferencia de datos al Tiva C a través de un puerto serial. La librería requiere que en ambos extremos de la comunicación estén definidas estructuras iguales de datos. En este caso las estructuras consisten de los vectores velocidad angular, posición XYZ estimada y la orientación dada por un cuaternión (array de cuatro componentes).

El programa principal realiza loops de 10 ms en los cuales lee la aceleración, orientación y velocidad angular de la IMU. Al final de cada loop se corre un paso de una máquina de estados secuencial que se encarga de manejar las lecturas del barómetro, esto es necesario porque luego de requerir una lectura de temperatura o presión es necesario esperar al menos 16 ms para que esté disponible. La máquina de estados hace una transición al cumplirse este intervalo de tiempo a los estados en los cuales son leídas las magnitudes. La máquina se encarga también de leer el GPS y llamar a la función correspondiente para correr un paso del filtro de Kalman cuando tiene datos nuevos. Luego de cada loop es actualizada la estructura de datos y transferida al puerto serial conectado al Tiva C.

17.2. Tiva C

En esta plataforma corre el software de control del cuadricóptero, que escencialmente está implementado como un loop de duración fija. El programa se encarga primero de recibir el estado estimado por el Teensy y convertir los datos crudos de cuaternión a ángulos de Euler y los datos crudos de velocidad angular a radianes por segundo. A partir de estos datos se ejecutan los controladores según el modo de operación en el que se encuentre y luego las salidas de estos controladores son traducidas a comandos que son enviados a los motores. En el loop esta previsto que el usuario pueda intervenir en la ejecución para cambiar el modo de operación o detenerla. En la figura (17.3) se muestra el diagrama de flujo del programa.

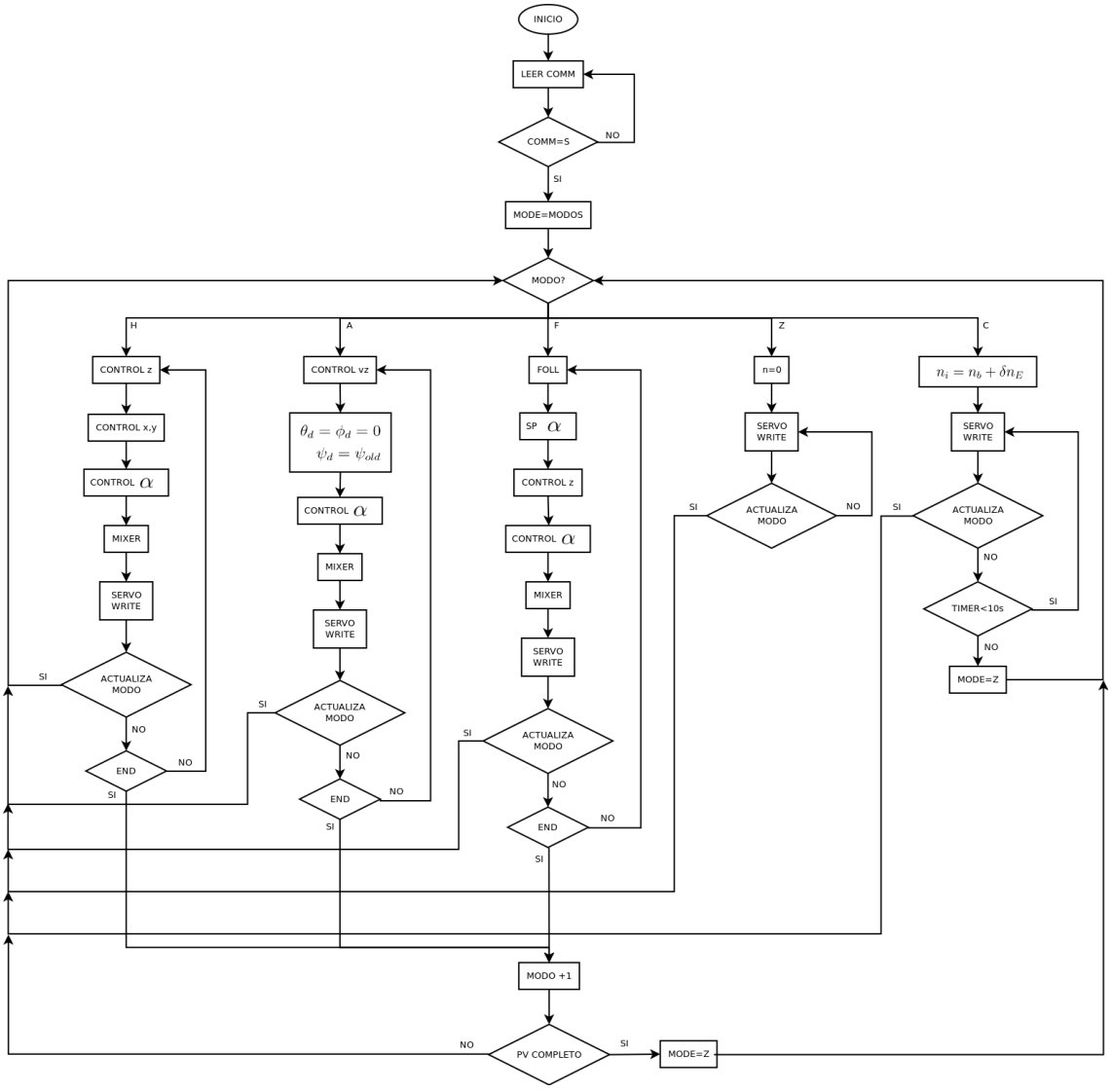


Figura 17.3: Diagrama de flujo del programa de control.

Los modos de operación se simbolizan mediante letras en mayúscula H, A, F, Z, C. Cada modo tiene un conjunto de parámetros que deben ser suministrados y una condición de finalización. Su funcionamiento es el siguiente:

1. Modo de Hovering (H): En este modo el cuadricóptero trata de mantener una posición fija (x, y, z) durante un cierto tiempo.

2. Modo de aterrizaje (A): En este modo se va decrementando la altura gradualmente, hasta llegar a una lectura de cero, que es cuando finaliza el modo. Este modo se apoya fuertmemente en la lectura de altura del sensor de ultrasonido, que tiene una exactitud del orden de un centímetro (superior a la proporcionada por el barómetro con fusión sensorial), lo que evita que el usuario deba intervenir en el proceso.

3. Modo de seguimiento de trayectorias (F): En este modo se ejecuta el algoritmo de seguimiento de trayectorias, que prporciona la dirección de vuelo (ángulo de yaw ψ), se ejecuta además el control de altura y velocidad.

4. Modo de finalización (Z): Detiene los motores completamente y espera un nuevo comando del usuario. Sólo debe ser usado luego de un aterrizaje o prueba de laboratorio.

5. Modo de caída controlada (C): Fue agregado para el caso extremo en que no lleguen datos nuevos de los sensores. Lo que hace es fijar una velocidad de rotación igual en todos los motores que permite que el cuadricóptero descienda pero no en caída libre.

Las librerías que utiliza el programa principal se detallan a continuación:

- **Control de ángulos - PID** (PIDclass): Clase que implementa el controlador PID de velocidad angular. Las constantes asociadas se encuentran en el archivo *pid_params.h*.
- **Control de ángulos - Redes Neuronales** (nnet): En esta librería se implementa la red neuronal dinámica para el controlador de velocidad angular. En el archivo *nnet_params.h* se encuentran definidos los parámetros de la red en arrays y los punteros a esos arrays, utilizados por las funciones de la librería que evaluan la respuesta de la red.
- **Control de ángulos - Redes Neuronales por Eje** (nnetAxis): Esta librería implementa una clase que permite instanciar controladores como objetos, uno por cada eje. En los archivos *cNNi_params.h* (i = x, y, z) se definen los parámetros y pesos de cada uno.
- **Control de altura - PID** (PIDh): Proporciona la clase de la cual se instancia el controlador de altura.
- **Seguimiento de trayectorias** (path_following): Calcula la dirección de vuelo requerida en cada instante para seguir una trayectoria dada por *waypoints*.
- **Control de motores** (mixer): Esta librería se encarga de traducir los comandos de momentos y fuerzas necesarias dado por los controladores en comandos a los motores.

- **Manejo de PWM** (ESCS): Se encarga de inicializar los módulos PWM del Tiva con la frecuencia deseada y conectar las salida a los pines deseados. Suministra las funciones para iniciar los módulos PWM y escribir comandos a los ESCs. Esta librería se ha implementado según las indicaciones proporcionadas por Texas Instruments en [43].
- **Transferencia serial de datos** (EasyTransfer): Usada de modo análogo que en el programa del Teensy, aquí como receptor.
- **Estimación de la orientación** (MadgwickAHRS): Esta librería puede ser usada en caso que se quiera estimar la orientación del cuadricóptero directamente a partir de los datos de acelerómetro, giróscopo y magnetómetro, según se comentó al final del capítulo Estado y poder sensorial. Está basada en [36].

Parte VII

Conclusiones

Se logró diseñar y construir un cuadricóptero que cumple con los objetivos de performance que se habían planteado en un principio. El mismo fue armado a partir de piezas y componentes disponibles en el mercado y piezas de diseño y fabricación propia. Se logró el objetivo de que el peso total se mantuviera cercano al kilogramo de peso, lo que le da un tiempo de vuelo estimado de 20 min, suficiente para el eventual testeo del sistema. Se resolvió adecuadamente el problema del sensado del estado de movimiento y fusión sensorial, esto implicó la elección puesta en funcionamiento de varios tipos de sensores, con diversos protocolos de comunicación y procesamiento de las medidas. La caracterización mecánica del cuadricóptero hecha permitió tener un modelo confiable del mismo, que fue utilizado en el problema de construir un controlador basado en redes neuronales.

En cuanto al procesamiento, se logró determinar plataformas de hardware adecuadas, lo que supuso una búsqueda y selección entre muchas opciones encontradas. Otro desafío que se presentó fue el de la familiarización con distintos entornos de desarrollo de software, algunos de ellos desconocidos en un principio para el grupo. La decisión de dividir las tareas entre dos microcontroladores (Teensy3.2 y Tiva C), uno encargado del sensado y el otro del control, permitió simplificar el software pudiendo concentrarse cada escritura de código en un conjunto de tareas relacionadas; pero a su vez planteó el problema de la comunicación de datos entre ellos y esto permitió entrar en contacto con las distintas soluciones en hardware y software existentes.

La solución del problema de control planteado se realizó por dos caminos, uno de ellos (controlador PID) era conocido sólo en forma teórica, por lo que en el proceso de diseño e implementación se adquirieron varios conocimientos valiosos a partir de problemas que surgen en la práctica de control (como, por ejemplo, el problema del *windup*). El segundo camino (redes neuronales) significó un estudio teórico previo, puesto que era casi completamente desconocido para el grupo. Se logró recorrer el camino desde el aprendizaje del tema hasta la aplicación a problemas de control, terminando en una realización práctica.

Al momento se ha llevado a cabo una prueba de vuelo con el cuadricóptero en el laboratorio, controlando la altura y sin efectuar el control en el plano horizontal (XY), debido a no estar disponible la señal de GPS. Esta prueba fue exitosa, ya que logró validar en un vuelo real el funcionamiento del control angular y parte del control traslacional. Es por esto que la experiencia hasta ahora es alentadora.

Una vez que se pueda hacer vuelos con el cuadricóptero, una de las tareas valiosas a realizar en la de obtener datos de los sensores en un vuelo real. Esto permitiría (además de servir como validación) la utilización de datos reales de la planta para el desarrollo del control basado en redes neuronales.

Parte VIII

Anexos

Capítulo 18

Redes neuronales

18.1. Problema del aprendizaje

La disciplina del *aprendizaje automático* o *aprendizaje de máquina* [1] es una rama de la computación que trata el problema de crear algoritmos y técnicas que permitan a las máquinas (computadoras) *aprender*, que en forma muy general significa que conociendo previamente un cierto número de entradas y salidas deseadas, puedan actuar frente a ciertas entradas sin haber sido explícitamente programadas para ese comportamiento deseado. Dicho de otro modo, si se acepta que hay una relación subyacente f entre un conjunto de entradas X y salidas Y, los programas deben tomar un conjunto de finito de ejemplos, parejas de valores $\{(x_n, y_n)\}$ (con $x_n \in X$ y $y_n \in Y$) conocidas, y generalizar a partir de éstas el comportamiento para cualquier otra entrada X.

18.1.1. Componentes del aprendizaje

En la figura (18.1) [2] se ilustra el esquema general del proceso de aprendizaje. Se intenta aprender una función objetivo desconocida $f : X \to Y$ donde X es el espacio de entrada e Y es el espacio de salida. Se supone que es conocido un cierto conjunto de datos de ejemplos entrada-salida obtenidos mediante la aplicación de la función f, $D = \{(x_n, y_n), n = 1, \ldots, N\}$ con $y_n = f(x_n)$. El algoritmo de aprendizaje A elige una función $g : X \to Y$ de entre un *conjunto de hipótesis* H con un cierto criterio de optimización, de modo que h sea la función que mejor aproxima a f en los datos del conjunto D. A D se lo conoce también como *conjunto de datos de entrenamiento*, puesto que el proceso de aprendizaje implementado por el algoritmo A puede ser visto como un entrenamiento sobre el conjunto de hipótesis H que permite aprender la función f.

[1] El nombre de esta disciplina fue acuñado originalmente en inglés y es *machine learning*.
[2] A partir de [7]

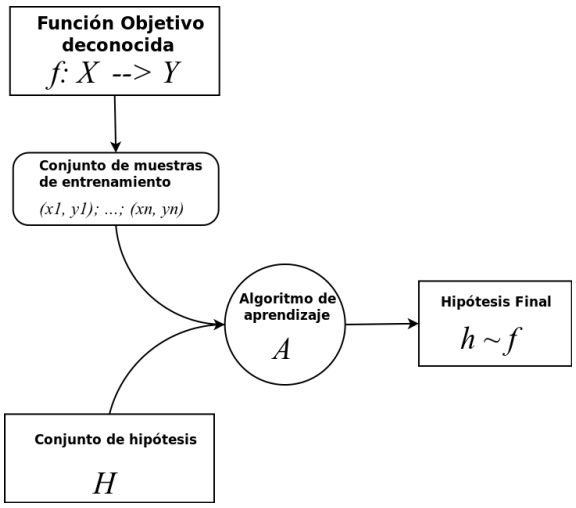


Figura 18.1: Diagrama ilustrativo del proceso de aprendizaje.

Los conjuntos mencionados pueden ser de tipos muy variados en cuanto a los elementos que los forman y tamaño que tengan. Para considerar un caso concreto, puede pensarse en un sistema de reconocimiento de dígitos escritos a mano. Si cada dígito manuscrito se muestrea muy rudimentariamente como una matriz de, por ejemplo, 16×16 pixeles blancos o negros, el espacio de entrada X será el conjunto de todas las combinaciones posibles de puntos blancos y negros en una matriz de 16×16. El espacio Y por otro lado será el conjunto de dígitos $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ y la función objetivo f será la correspondencia entre el dibujo a mano de un dígito (muestreado) y el dígito al cual representa ese dibujo. En este sencillo ejemplo, ambos conjuntos son finitos y el algoritmo de aprendizaje intenta aproximar la función f conociendo un cierto número de ejemplos (imagen, dígito), menor a las 256 combinaciones posibles.

En el caso de este proyecto, la función f es escencialmente la dinámica de un cuerpo rígido (el cuadricóptero) sometido a un cierto conjunto de momentos. El conjunto de entradas X está formado por las funciones momento en función del tiempo $\vec{M}(t) = (M_x(t), M_y(t), M_z(t))$ admisibles (que de hecho son continuas a tramos). Las salidas del conjunto Y son las funciones velocidad angular $\omega(t) = (\omega_x(t), \omega_y(t), \omega_z(t))$ con las que responde la dinámica del rígido a las entradas de momento. La función a aprender es en realidad una *funcional*, puesto que relaciona funciones de entrada con funciones de salida y no valores de variables.

18.2. Redes neuronales

Las redes neuronales son un tipo específico de funciones hipótesis (h) del conjunto H en el proceso de aprendizaje. Su ventaja radica en dos propiedades importantes que poseen. Primero, pueden aproximar cualquier función real f que caiga dentro de un conjunto muy general de funciones, como se expondrá más adelante. Segundo, cuando se trabaja con redes neuronales el algoritmo de aprendizaje A, en la búsqueda de la mejor aproximación a f, opera sobre el conjunto H de una manera suficientemente simple que permite implementarlo en software con un tiempo de cálculo razonable. Inicialmente las redes neuronales fueron modelos inspirados en las redes neuronales biológicas [30] que se encuentran en el sistema nervioso central de los animales. A partir de su introducción, se ha abandonado el objetivo de modelar las redes biológicas y los distintos tipos de redes neuronales han adquirido importancia por sí mismas como aproximadores de funciones y en ocasiones se utiliza el término más adecuado *redes neuronales artificiales*.

18.2.1. Perceptrón

La unidad básica de una red neuronal es el *perceptrón*, cuya estructura se muestra en la figura (18.2) e implementa una función real parametrizada de varias variables. Las entradas $x_1, \ldots, x_n$ son ponderadas por los pesos w_i correspondientes a cada uno de sus canales y luego se le adiciona un offset o bias w_0 para formar una suma ponderada s:

$$s = \sum_{i=1}^{n} w_i x_i + w_0$$

que, definiendo una entrada $x_0 = 1$, se puede escribir en forma compacta como:

$$s = \sum_{i=0}^{n} w_i x_i$$

a continuación esta suma s es provista como entrada a una *función de activación* $\theta(.)$ que da como resultado la salida del perceptrón y.

La función de activación θ puede ser elegida de modos diversos según la aplicación que se quiera hacer del perceptrón. En problemas de clasificación (como en el ejemplo de reconocimiento de dígitos mencionado anteriormente) suelen utilizarse escalones del tipo:

$$\theta(s) = \begin{cases} 1 & s > 0 \\ -1 & s < 0 \end{cases}$$

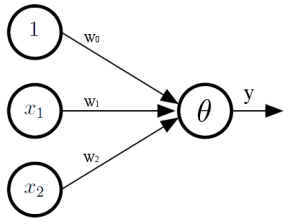


Figura 18.2: Perceptrón.

18.2.2. Estructura de una red neuronal

Una red neuronal está formada por capas de perceptrones interconectadas, como se muestra en el esquema de la figura (18.3), donde las entradas a la red son $x_1, \ldots, x_n$ y las salidas de la red $y_1, \ldots, y_n$ son las salidas de las neuronas en la última capa. Cada círculo indicado con p es un perceptrón y cada flecha lleva asociado el peso de la entrada correspondiente. La red queda definida especificando su arquitectura y el tipo de las funciones de activación en cada capa. Por regla general estas funciones son diferenciables, estando este requerimiento relacionado a los algoritmos de aprendizaje utilizados en ellas, esto permite calcular los gradientes de la salida respecto a los pesos, lo cual es necesario para los algoritmos de aprendizaje.

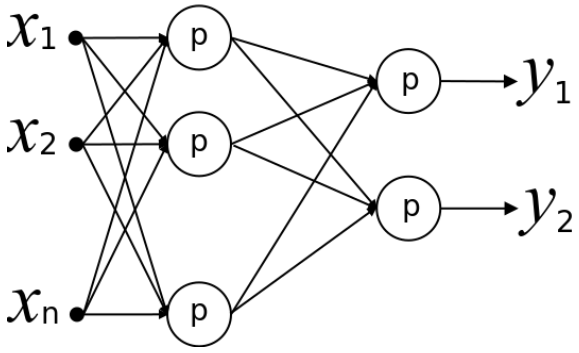


Figura 18.3: Estructura básica de una red neuronal.

Una red neuronal es un *aproximador universal* de funciones de varias variables. Esto queda formalizado en el siguiente teorema:

Teorema de aproximación universal [29]:

Dada una función $f \in C([0,1]^m)$ y $\epsilon > 0$, existe un número natural N, números reales ν_i y ciertos pesos y bias w_i, b_i tales que:

$$F(x) = \sum_{i=1}^{N} \nu_i \theta(\vec{w}_i.\vec{x} + b_i) \tag{18.1}$$

aproxima a f según:

$$|f(x) - F(x)| < \epsilon \qquad \forall x \in [0,1]^m$$

En este contexto los vectores $\vec{w}_i$ son los pesos del perceptrón i y b_i el bias correspondiente. La función de activación *theta* se supone que es continua, acotada y monótonamente creciente. Se puede deducir a partir de 18.1 que esta aproximación es una red neuronal con N perceptrones en la primera capa con pesos $\vec{w}_i$ y uno en la capa de salida con pesos ν_i, siendo la función de activación de esta última lineal de coeficiente 1.

18.2.3. Algoritmos de entrenamiento

Los algoritmos de entrenamiento en el caso de redes neuronales tienen el objetivo de variar los pesos y bias, que son los parámetros que determinan la función h (que es la misma red neuronal) en el espacio de hipótesis H. Se basan en minimizar una función de error:

$$E(w) = \frac{1}{N} \sum_{n=1}^{N} e(h_w(x_n),\, y_n) \tag{18.2}$$

donde h_w representa a la red en H y (x_n, y_n) son las parejas de entrenamiento. La función error e se toma en general como el cuadrado de la diferencia, que es una función diferenciable. Las técnicas de optimización se basan escencialmente en mover los pesos en la dirección contraria al gradiente del error $E(w)$ y se explican en ([32]).

18.2.4. Redes neuronales dinámicas

Las redes descriptas hasta aquí son redes algebraicas, esto es, no tienen memoria. La salida para una entrada dada está determinada sólo por esa entrada y es independiente de las entradas previas a la red. Este no es el comportamiento de un sistema dinámico y por tanto estas redes no son útiles para modelar a estos sistemas. Las redes *recurrentes* poseen al menos un conexión de realimentación, implementando de este modo un tipo de memoria, por lo que son capaces de hacer procesamiento de series temporales y aprender secuencias. Existen varios tipos de redes recurrentes, según la elección de las realimentaciones que pueden consultarse en ([32]) y las referencias allí. En este apéndice sólo se presentará el tipo de red recurrente utilizado en el proyecto, cuyo esquema general se muestra en la figura 18.4.

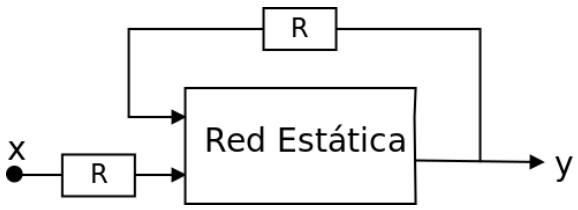


Figura 18.4: Red neuronal recurrente.

Las entradas y salidas son en general vectoriales y los bloques R mostrados implimentan varios retardos de un paso de tiempo. Este tipo de red recurrente se denomina *NARX* por su sigla en inglés: *Nonlinear Autoregresive with eXogenous imputs* y su relación entrada - salida es de la forma:

$$y_{n+1} = F(y_n, \ldots, y_{n-q+1}; x_n, \ldots, x_{n-p+1}) \tag{18.3}$$

donde p y q son la cantidad de retardos en la entrada y en la realimentación, respectivamente.

Capítulo 19

Rotaciones

En este apéndice se presenta un resumen de las distintas representaciones de las rotaciones en el espacio, introduciendo los conceptos relevantes utilizados en la descripción de los grados de libertad de actitud de un cuerpo rígido (como el robot del proyecto) que se mueve en el espacio. Especificar la actitud u orientación de un rígido implica fijar tres objetos matemáticos fundamentales:

1) Un sistema de ejes *fijos* que se toman como referencia. En el caso de este proyecto, estos ejes están fijos a la tierra.

2) Un sistema de ejes solidario al robot. Estos ejes son elegidos según se explicó en 5.

Una vez determinados estos sistemas de ejes, la actitud del robot viene dada por:

3) La rotación que lleva a los ejes fijos a coincidir con los ejes solidarios al robot.

Esto último se apoya en el hecho de que dados dos sistemas de ejes coordenados *directos* en el espacio, uno con versores asociados a los ejes $\{\hat{e}_1, \hat{e}_2, \hat{e}_3\}$ y el otro con versores $\{\hat{e}_1\,', \hat{e}_2\,', \hat{e}_3\,'\}$, siempre existe una rotación que lleva los versores de uno a coincidir con los del otro. Una vez conocida esa rotación, sólo resta proporcionar la traslación que lleva un origen al otro, que corresponde a los tres grados de libertad de traslación. Se proporcionará primero la expresión matemática de una rotación actuando sobre un vector y luego se introducirán los objetos más comúnmente usados para la representación en coordenadas.

19.1. Rotaciones

[25]

Una rotación de eje $\hat{n}$ y ángulo α, simbolizada por $R_{\alpha\hat{n}}$, aplicada a un vector $\vec{r}$ da como resultado un vector $\vec{r}\,'$ que se obtiene según:

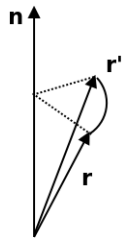


Figura 19.1: Rotación de un vector $\vec{r}$

$$\vec{r}\,' = R_{\alpha\hat{n}}\,\vec{r} = \cos\alpha\,\vec{r} + \operatorname{sen}\alpha\,\hat{n}\times\vec{r} + (1-\cos\alpha)(\hat{n}.\vec{r})\hat{n} \tag{19.1}$$

Para deducir esta fórmula partiendo de la geometría espacial, se utiliza el hecho de que las rotaciones son movimientos *rígidos*, esto es, el módulo del vector rotado es igual al del vector original: $|\vec{r}\,'| = |\vec{r}|$. Por otro lado, si se la toma como *definición* analítica es fácil comprobar esta propiedad calculando directamente el producto escalar $\vec{r}\,'.\vec{r}\,'$ a partir de (19.1). Esta propiedad de las transformaciones lineales implica que se cumple:

$$R^T_{\alpha\hat{n}} = R^{-1}_{\alpha\hat{n}}$$

Cuando se trabaja con sistemas de coordenadas con una terna (base) de vectores asociada, es necesario conocer la ley de transformación de las componentes de un vector cualquiera al aplicar una rotación a los vectores de la base. Las componentes de un vector $\vec{r}$ son los productos escalares con los vectores de la base de versores, luego, dado un vector $\vec{r}$ con componentes r_i en la base original y r'_i en la rotada:

$$r'_i \;=\; \hat{e}'_i.\vec{r} = \left(R_{\alpha\hat{n}}\,\hat{e}_i\right).\vec{r} = \hat{e}_i.\left(R^T_{\alpha\hat{n}}\,\vec{r}\right) = \hat{e}_i.\left(R^{-1}_{\alpha\hat{n}}\,\vec{r}\right) = \left(R^{-1}_{\alpha\hat{n}}\,\vec{r}\right)_i$$

de modo que al rotar los ejes las nuevas componentes de $\vec{r}$ son las componentes del vector que resulta de aplicar a $\vec{r}$ la rotación inversa. Haciendo el cambio $\theta \to -\theta$ en (19.1):

$$\vec{r}\,' \;= R_{\alpha\hat{n}}\,\vec{r} = \cos\alpha\,\vec{r} - \operatorname{sen}\alpha\,\hat{n}\times\vec{r} + (1-\cos\alpha)(\hat{n}.\vec{r})\hat{n}$$

En este punto es conveniente introducir una nueva notación que haga más explícito el sistema en el cual las componentes de $\vec{r}$ se calculan. Se utiliza la expresión ${}^{A}\vec{r}$ para indicar las componentes del vector $\vec{r}$ en un sistema S_A. A la matriz asociada a la rotación $R_{\alpha\hat{n}}$ que lleva el sistema S_A en el sistema S_B se la denota por ${}^{B}_{A}R$. En esta convención resulta para las componentes:

$$ {}^{B}\vec{r} = \vec{r}' = {}^{B}_{A}R \, {}^{A}\vec{r} $$

Donde la matriz ${}^{B}_{A}R$ es:

$$ \begin{pmatrix} \cos\alpha + (1-\cos\alpha)n_1^2 & n_3 \operatorname{sen}\alpha + (1-\cos\alpha)n_1 n_2 & -n_2 \operatorname{sen}\alpha + (1-\cos\alpha)n_1 n_3 \\ -n_3 \operatorname{sen}\alpha + (1-\cos\alpha)n_1 n_2 & \cos\theta + (1-\cos\theta)n_2^2 & n_1 \operatorname{sen}\alpha + (1-\cos\alpha)n_2 n_3 \\ n_2 \operatorname{sen}\alpha + (1-\cos\alpha)n_1 n_3 & -n_1 \operatorname{sen}\alpha + (1-\cos\alpha)n_2 n_3 & \cos\theta + (1-\cos\theta)n_3^2 \end{pmatrix} \tag{19.2} $$

A partir de esta matriz se obtiene la transformación de coordenadas ante cualquier rotación de ejes, en particular justifican las expresiones para las matrices de rotación en términos de los ángulos de Euler.

19.2. Cuaterniones

Las matrices de rotación en términos de los ángulos de Euler implican calcular varias funciones trigonométricas de los ángulos de rotación y esto implica en general que el esfuerzo computacional al utilizar esta representación es elevado. Otro problema que presenta ese tratamiento es que cualquier sistema de ángulos de Euler posee una ambigüedad para algún conjunto de valores de los ángulos. [1]. Una manera de evadir estas complicaciones es describir las rotaciones por medio de otros objetos matemáticos llamados *cuaterniones*.

Los cuaterniones fueron introducidos por el matemático W. R. Hamilton[26] como una extensión de los números complejos $z = x + jy$ y son escencialmente números complejos con *tres unidades imaginarias* i, j, k. En forma análoga a un número complejo, un cuaternión en general tiene la forma:

$$ q = q_0 + q_1 i + q_2 j + q_3 k \tag{19.3} $$

donde las *componentes* q_{0123} son números reales y las nuevas unidades $(1, i, j, k)$ se multiplican de acuerdo a la tabla de multiplicación siguiente:

×	1	i	j	k
1	1	i	j	k
i	i	-1	k	-j
j	j	-k	-1	i
k	k	j	-i	-1

El conjunto de cuatro números reales (componentes) (q_0, q_1, q_2, q_3) se agrupan en una parte *escalar* (q_0) y una parte *vectorial* $\vec{q} = (q_1, q_2, q_3)$, siendo usual la notación $q = (q_0, \vec{q})$. El conjunto de cuaterniones está dotado de una estructura algebraica por las operaciones binarias de suma (+) (componente a componente) y el producto ($\otimes$) de dos cuaterniones $q = (q_0, \vec{q})$ y $p = (p_0, \vec{p})$ que se obtiene de la tabla de multiplicación de las unidades:

[1] Si bien el caso de un cuadricóptero, con la sistema elegido en este proyecto (ZYX) no se presenta esa ambigüedad en condiciones normales de vuelo.

$$q \otimes p = (q_0 p_0 - \vec{q}.\vec{p}, q_0\vec{p} + p_0\vec{q} + \vec{q} \times \vec{p})$$

que no es conmutativo, $q \otimes p \neq p \otimes q$. Estas operaciones satisfacen todas las propiedades necesarias para dar a los cuaterniones una estructura de anillo (en particular $\otimes$ es distributiva respecto a $+$), el neutro de la suma es $(0, \vec{0})$ y el anillo posee una unidad del producto dada por $(1, \vec{0})$. En forma similar a los números complejos, se define la *norma* de un cuaternión $q = (q_0, q_1, q_2, q_3)$ como la expresión:

$$|q| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}$$

y el *conjugado* de $q = (q_0, \vec{q})$:

$$q^* = (q_0, -\vec{q})$$

La operación de conjugación permite escribir la norma $|q|^2 = q \otimes q^*$ y definir el inverso de q según:

$$q^{-1} = \frac{q^*}{|q|^2}$$

La aplicación a rotaciones espaciales proviene del hecho que un vector $\vec{r} = r_1\hat{e}_1 + r_2\hat{e}_2 + r_3\hat{e}_3$ tiene una representación natural como un cuaternión con parte escalar nula:

$$\vec{r} \sim r = 0 \, + \, r_1 i + r_2 j + r_3 k$$

que, abusando de la notación, se denota simplemente por $\vec{r}$. Por otro lado, se puede obtener a partir de la definición que el producto $q^* \otimes \vec{r} \otimes q$ (análogo a una transformación de similaridad para matrices) tiene parte escalar nula y por tanto define siempre un vector asociado en el espacio:

$$\vec{r}' \, = \, q^* \otimes \vec{r} \otimes q \tag{19.4}$$

Dada una rotación de ejes coordenados desde el sistema S_A al sistema S_B, $R_{\alpha\hat{n}}$, con matriz (19.2)asociada ${}^B_A R$, es posible siempre construir un cuaternión ${}^B_A q$ a partir de sus parámetros:

$$\begin{aligned} {}^B_A q &= \cos\left(\frac{\alpha}{2}\right) + \hat{n} \, \sin\left(\frac{\alpha}{2}\right) && (19.5)\\ &= \cos\left(\frac{\alpha}{2}\right) \, + \, \hat{n}_1 \, \sin\left(\frac{\alpha}{2}\right) i \, + \, \hat{n}_2 \, \sin\left(\frac{\alpha}{2}\right) j \, + \, \hat{n}_3 \, \sin\left(\frac{\alpha}{2}\right) k && (19.6) \end{aligned}$$

de modo que el producto (19.4) proporcione en $\vec{r}'$ las coordenadas del vector $\vec{r}$ en los nuevos ejes ${}^B_A\vec{r}$. Esto puede verificarse directamente a partir de (19.4) sustituyendo q por (19.5)

$${}^B\vec{v} \, = \, {}^B_A q^* \, \otimes \, {}^A\vec{v} \, \otimes \, {}^B_A q \tag{19.7}$$

para obtener:

$${}^B\vec{r} \, = \, \vec{r}' \, = \, {}^B_A R \, {}^A\vec{r}$$

con ${}^B_A R$ la matriz (19.2). La ventaja surge al expresar la transformación (19.4) en términos de las componentes (q_0, q_1, q_2, q_3) del cuaternión ${}^B_A q$ construido en (19.5), haciendo esto resulta que la matriz de cambio de base ${}^B_A R$ tiene la expresión:

$$ {}^B_A R = \begin{pmatrix} 2(q_0^2 + q_1^2) - 1 & 2(q_1 q_2 + q_0 q_3) & 2(q_1 q_3 - q_0 q_2) \\ 2(q_1 q_2 - q_0 q_3) & 2(q_0^2 + q_2^2) - 1 & 2(q_2 q_3 + q_0 q_1) \\ 2(q_3 q_1 + q_0 q_2) & 2(q_3 q_2 - q_0 q_1) & 2(q_0^2 + q_3^2) - 1 \end{pmatrix} \tag{19.8} $$

Que es polinómica en las componentes de q y además tiene muchos productos repetidos, lo que simplifica el cálculo de sus componentes. Un punto importante en esta descripción de una rotación es que el cuaternión q debe estar *normalizado*, esto es, $|q| = 1$ lo que puede verse directamente en la def. (19.5).

Si se parametriza la misma rotación mediante ángulos de Euler, la relación con ${}^B_A q$ es:

$$ \psi = \text{atan2}\left(2(q_1 q_2 + q_0 q_3), q_0^2 + q_1^2 - q_2^2 - q_3^2\right) \tag{19.9} $$

$$ \theta = \text{asin}\left(2q_0 q_2 - 2q_1 q_3\right) \tag{19.10} $$

$$ \phi = \text{atan2}\left(2(q_2 q_3 + q_0 q_1), q_3^2 - q_2^2 - q_1^2 + q_0^2\right) \tag{19.11} $$

la condición de normalización es necesaria para la existencia de los ángulos definidos por las relaciones (19.9), (19.10) y (19.11).

Los cuaterniones q y $-q$ describen la misma rotación debido a que, según (19.5), la inversión de signo en q equivale a cambiar $\alpha \to \pi - \alpha$ y $\hat{n} \to -\hat{n}$, y esto define la misma rotación en el espacio. Esta ambigüedad puede evitarse fácilmente si se conviene en tomar siempre $q_0 > 0$, de esta forma existe una correspondencia 1-1 entre las rotaciones y la semi-hiperesfera $|q| = 1, q_0 > 0$.

Finalmente, la relación de las derivadas de los parámetros con la velocidad angular viene dada por las ecuaciones (19.12) para ánglos de Euler, mientras en cuaterniones[25] adquieren la forma más compacta (19.13).

$$ \begin{aligned} \dot{\psi} &= \omega_2 \frac{\sin\phi}{\cos\theta} + \omega_3 \frac{\cos\phi}{\cos\theta} \\ \dot{\theta} &= \omega_2 \cos\phi - \omega_3 \sin\phi \\ \dot{\phi} &= \omega_1 + \omega_2 \frac{\sin^2\phi}{\cos\theta} + \omega_3 \frac{\sin\phi\cos\phi}{\cos\theta} \end{aligned} \tag{19.12} $$

$$ \dot{q} = \frac{1}{2} q \otimes \vec{\omega} \tag{19.13} $$

Capítulo 20

Calibración de controladores ESC

Los controladores de los motores, (ESCs) reciben una señal PWM a partir de la cual setean la velocidad del motor que controlan. La señal PWM enviada al ESC es una señal cuadrada de amplitud $5V$ y $20ms$ de período. A partir del tiempo en alto ($5V$) de esta señal, el ESC setea la velocidad del motor controlado.
Controladores utilizados: ESCs marca "Afroesc" modelo 30 A.

20.1. Calibración por defecto

La calibración de fábrica:
Motor a máxima velocidad: Pulsos de $1860\mu s$ de duración en alto.
Motor a mínima velocidad: Pulsos de $1060\mu s$ de duración en alto.

20.2. Proceso de calibración de los ESC

Fecha de realización: 26/10/2015.

En caso que se desee utilizar señales de distinta duración de pulso que las de defecto, para la máxima y mínima velocidad, el proceso a seguir es el siguiente:

1. Con el ESC apagado, inyectar en su entrada de señal, la señal con duración de pulso en alto que corresponderá al máximo de velocidad. Ejemplo: pulsos en alto de $1600\mu s$ de duración.

2. Encender el ESC, esperar sonido de encendido más 1 pitido final.

3. Inyectar la señal correspondiente a velocidad mínima, por ejemplo, bajar la duración del pulso alto en la señal a $1200\mu s$. Esperar 2 pitidos cortos más 1 pitido largo. Luz LED del ESC en verde significa calibración exitosa.

20.3. Consideraciones

El ESC adiciona $60\mu s$ al seteo de duración de pulso en alto correspondiente al mínimo. Con los valores de20.2, la señal PWM de velocidad mínima configurada es de $1260\mu s$ de duración. Esto lo hace para que el motor no encienda inmediatamente al proceso de calibración. La señal de velocidad máxima es de $1600\mu s$ de pulso en alto.

Una vez configurado el ESC, si se apaga, no vuelve a la calibración por defecto sino que guarda la última calibración seteada. Por lo tanto, para cada nueva calibración que se desee realizar, tener en cuenta en el proceso de calibración que:

- La señal inyectada al encender el ESC (que corresponde a velocidad máxima), debe tener una duración de pulso en alto mayor a la última configurada para la velocidad mínima. De esta manera, es interpretada por el ESC como el nuevo máximo de velocidad. Si esto se cumple, el proceso de calibración se continúa con normalidad, pudiendo setear incluso un nuevo mínimo.
- Si la señal inyectada al encender el ESC tiene una duración de pulso en alto menor a la última seteada para el mínimo, el ESC queda con la última configuración seteada.
- Los límites prácticos de calibración para los ESC son aproximadamente (medidos en laboratorio):

  Mínimo: $900\mu s$

  Máximo: $2100\mu s$

20.4. Calibración a utilizar

La calibración que se va a utilizar es:
Velocidad mínima: Pulsos de $1000\mu s$. (Funciona a partir de $1060\mu s$)
Velocidad máxima: Pulsos de $2000\mu s$.

Capítulo 21

Adquisición de datos en vuelo

Para entrenar la red neuronal se utilizó el modelo matemático hallado con los parámetros mecánicas resultado de la caracterización, de este modo se tiene la ventaja de poder simular entradas a la red que contemplen situaciones extremas, cosa que no podría hacerse directamente con el cuadricóptero físico ya que arruinaría el equipo, sin embargo, modelar "el mundo" que afecta al cuadricóptero es imposible, por lo que a modo complementario de este entrenamiento, se decide realizar el vuelo del artefacto real, mediante un control de radiofrecuencia, y registrar datos en un vuelo real, en un medio controlado pero con entradas y salidas reales.

El controlador de vuelo seleccionado no cuenta con memoria flash y soporte para "blackbox", la çaja negra"que registra los incidentes del vuelo. Por lo tanto se decide utilizar un controlador disponible del grupo del proyecto uQuad2, el CC3D; el mismo tiene integrada una memoria flash de 128KB pero no cuenta con un software asociado (OpenPilot) con compatibilidad de blackbox, por lo que se procede a cambiarle el firmware a la misma para que sea compatible con el software ÇleanFlight"

Elementos utilizados:

- Controlador de vuelo CC3D
- Conversor serial-USB
- PC
- Jumper para cortocircuitar terminales
- conector mainport para CC3D

Procedimiento:

Se conecta el CC3D al conversor serial-USB mediante el conector mainport para CC3D
puertos:

- 5V a 5V
- GND a GND
- TX a TX
- RX a RX

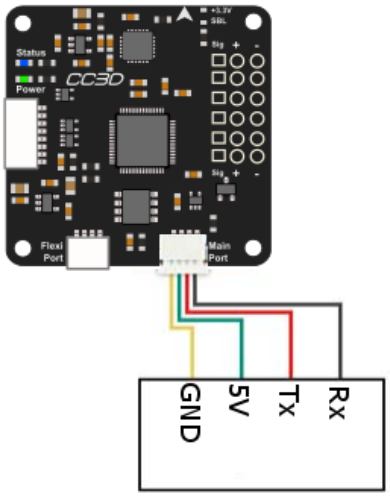
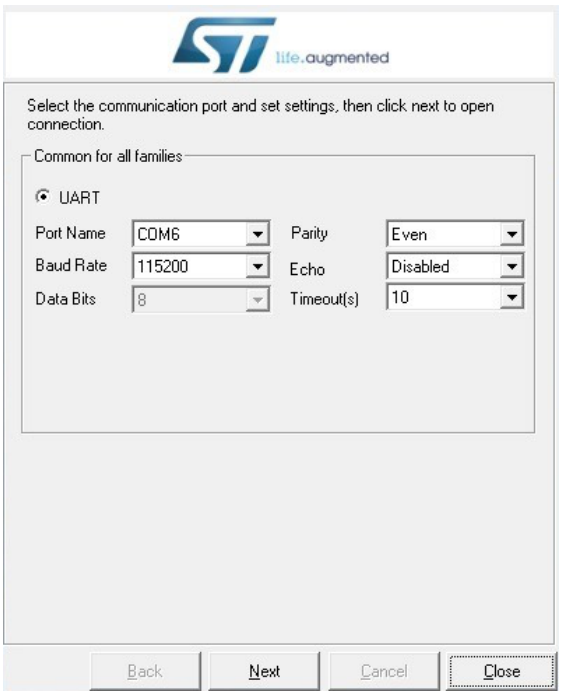


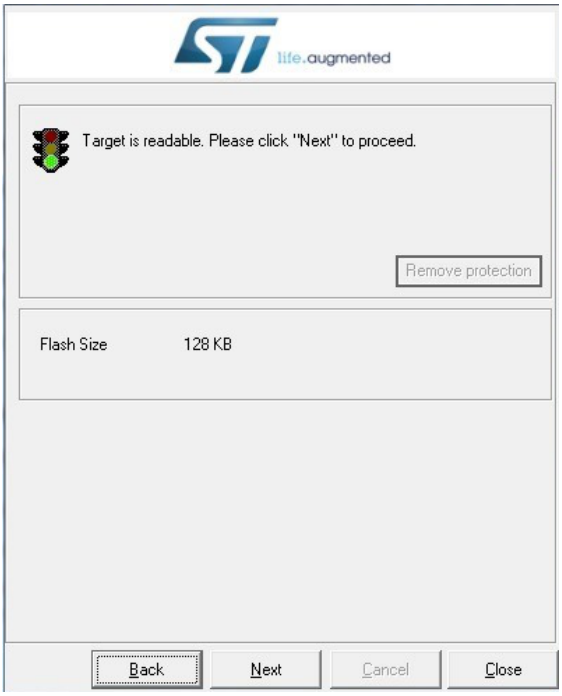
Figura 21.1: mainport CC3D

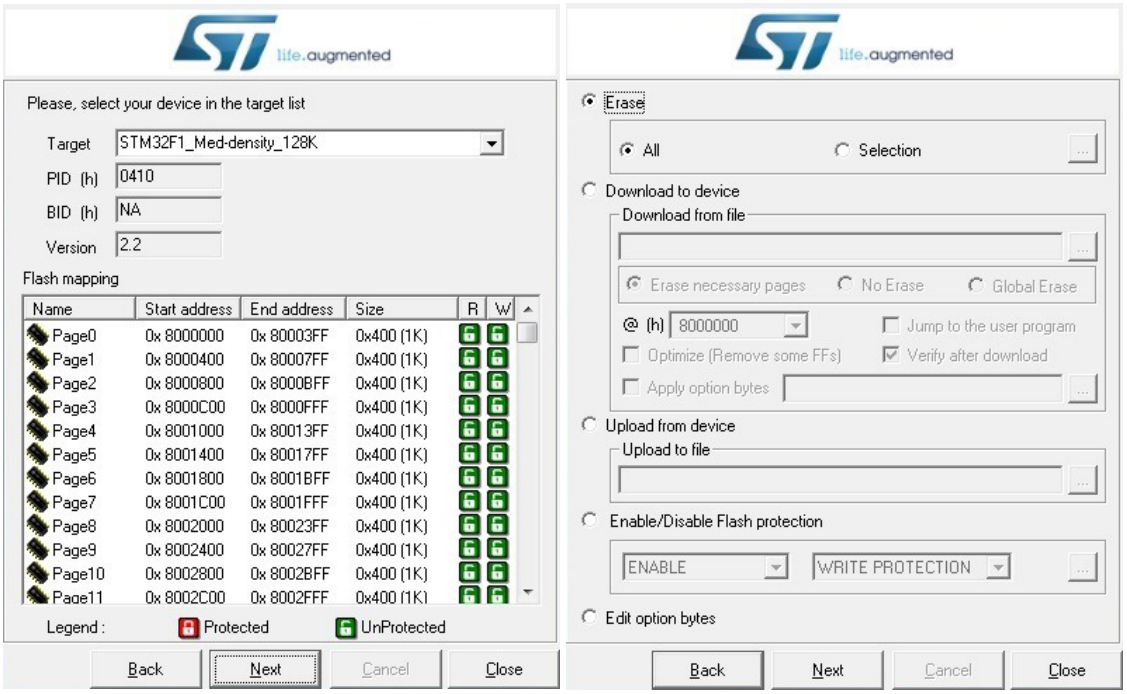
En la PC se ejecuta el software de cambio de firmware "STM Flash Demonstrator".

Cortocircuitar los terminales "3.3Vz "SBLz luego conectar el conversor serial-USB a la PC, de este modo el controlador de vuelo enciende en modo de booteo, puede verificarse que se encuentra en este modo si solamente enciende el LED verde de forma permanente (en modo normal hay dos LEDs encendidos, uno verde y otro azul intermitente).

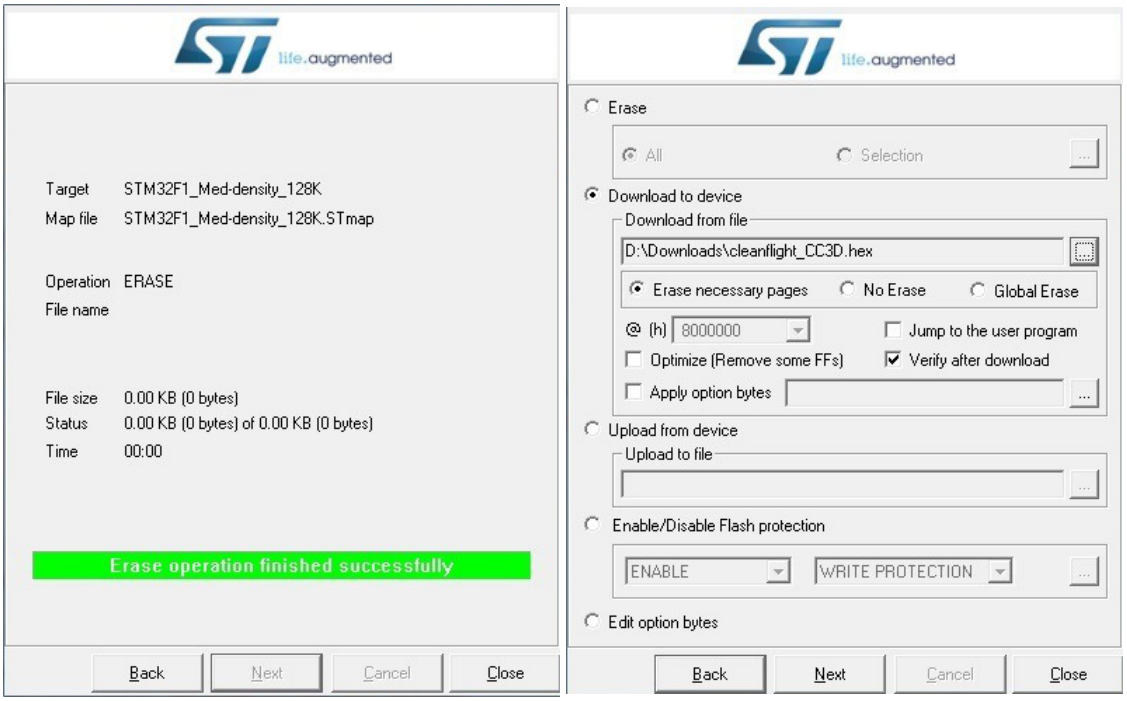
Ejecutar STM Flash Demonstrator en la PC y setear los campos y siguiendo los pasos como se muestra a continuación:

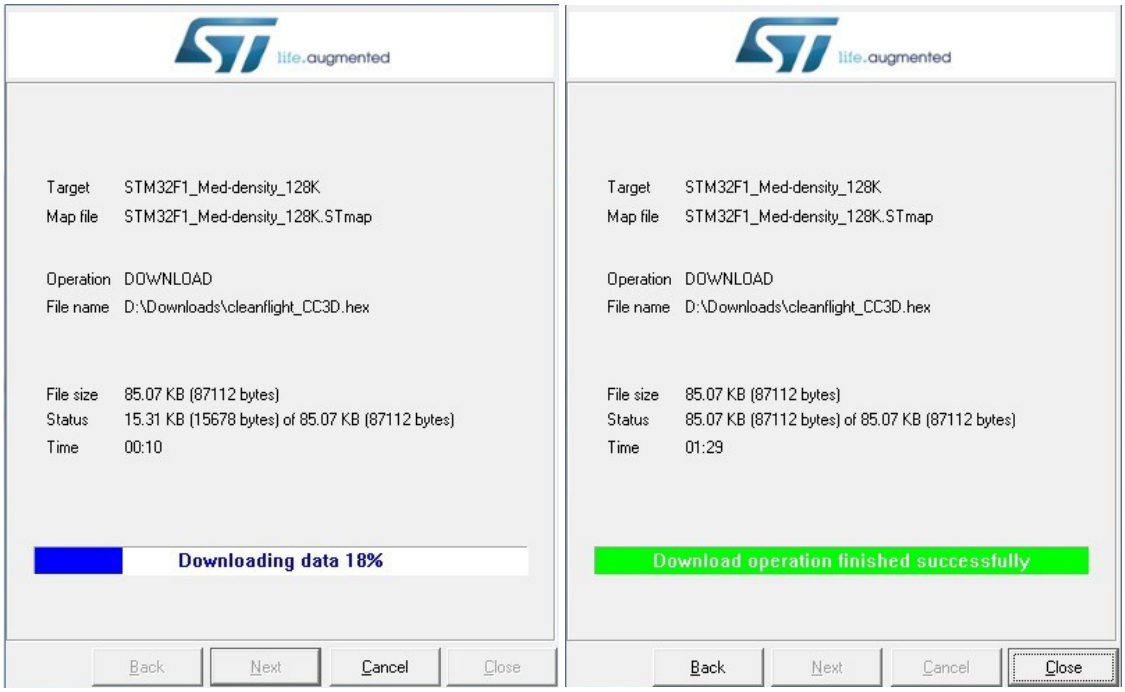






Seleccionar el firmware (con extensión ".hex"):





Finalizado este proceso reiniciar el CC3D, desconectar el conversor serie-USB y el mainport; conectar el CC3D al PC mediante un cable USB-miniUSB y ejecutar el software ÇleanFlight"verificando que reconozca el dispositivo.

Capítulo 22

Piezas hardware adicionales

Para integrar todo el hardware al frame seleccionado fue necesario diseñar piezas de acrílico adicionales de modo de contar con varios niveles en los cuales incluir los sensores y resto de la inteligencia, en este anexo se detallan las principales características de cada una.

22.1. Piezas acrílico

Piezas confeccionadas en acrílico transparente de $2mm$ de espesor en una cortadora láser.

22.1.1. Batería y ultrasonido

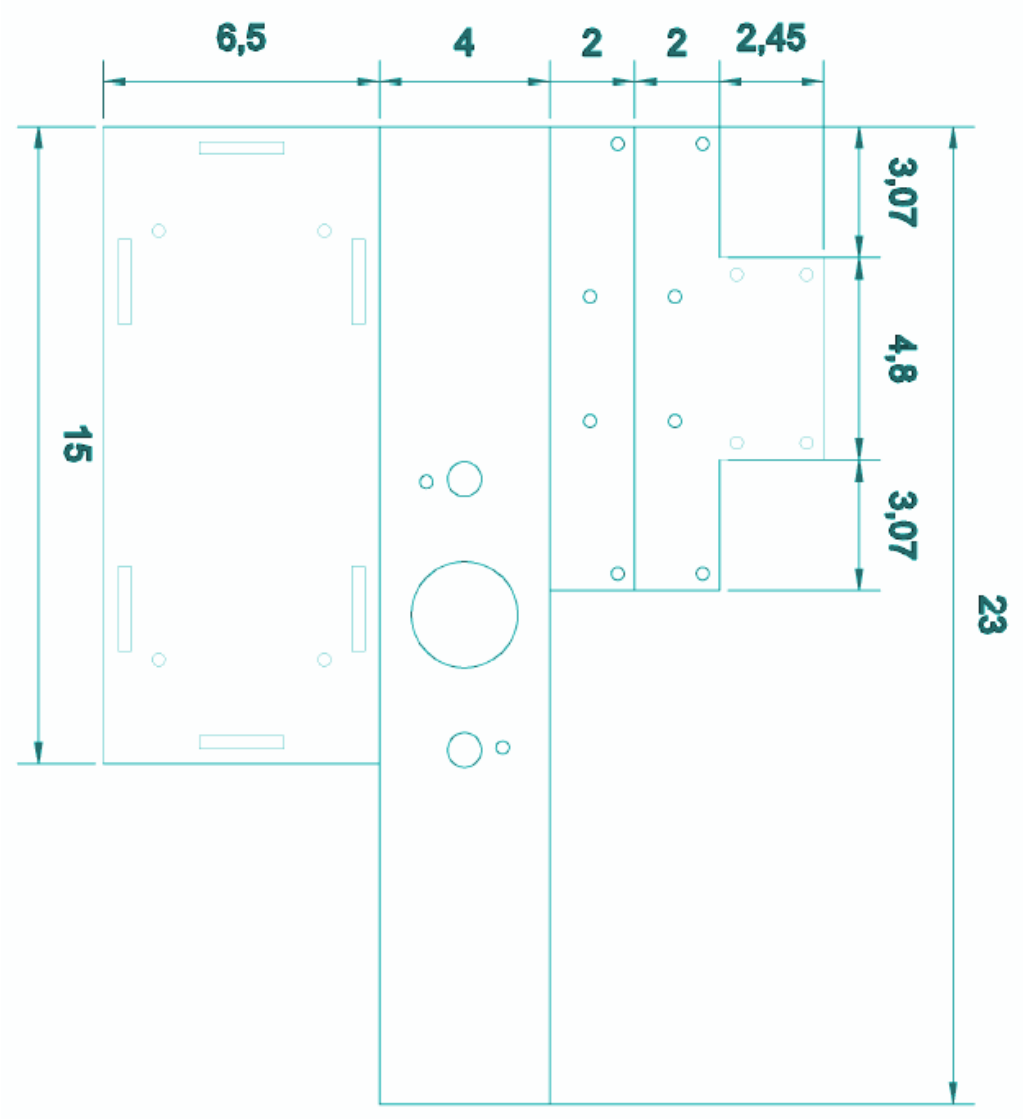


Figura 22.1: Piezas sujeción batería y ultrasonido. Dimensiones en cm

22.1.2. Sujeción Tiva, GPS, Wifi

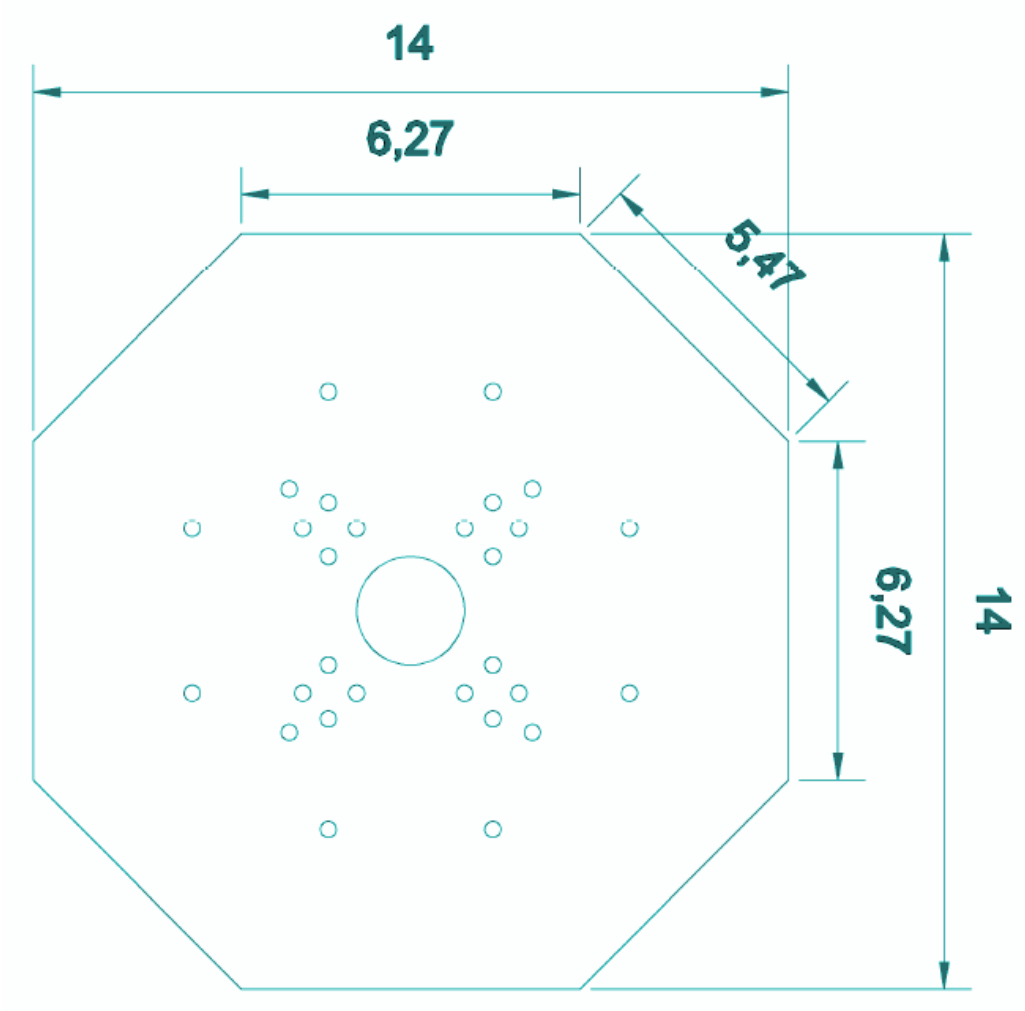


Figura 22.2: Piezas sujeción Tiva, GPS y Wifi. Dimensiones en cm

22.2. PCB

Con el fin de facilitar trabajos futuros sobre el cuadricóptero, se diseña un PCB modular facilitando el acceso a los diferentes componentes electrónicos y sustitución de los mismos. La asignación de cada Pin y ubicación de la electrónica se muestra en la figura siguiente:

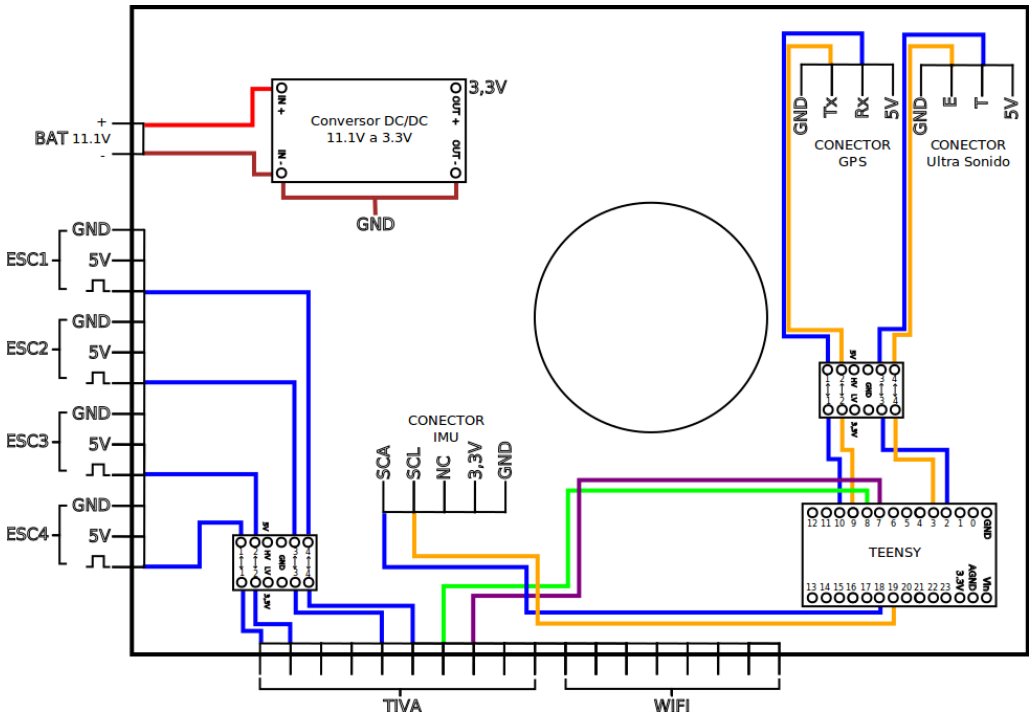
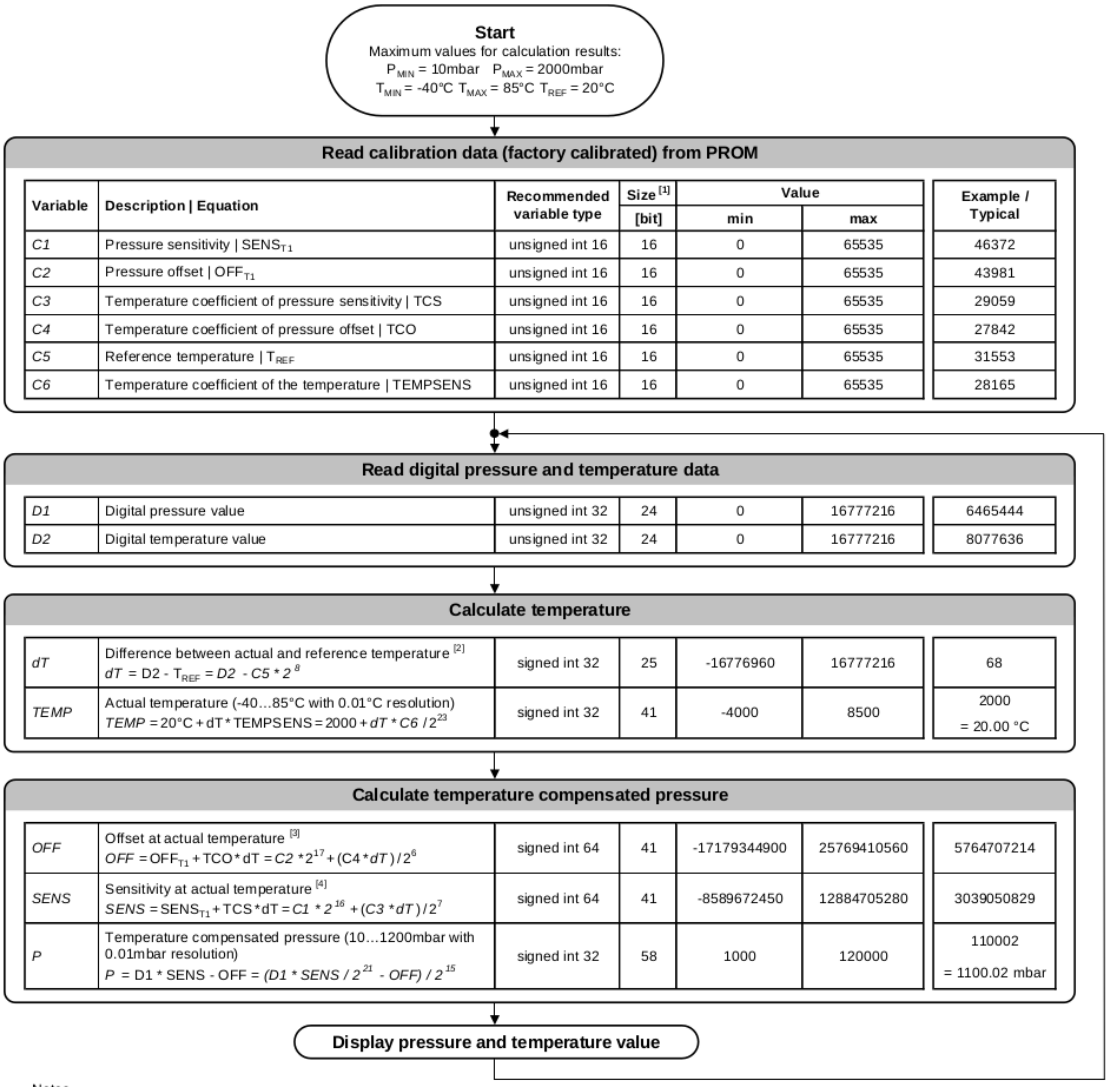
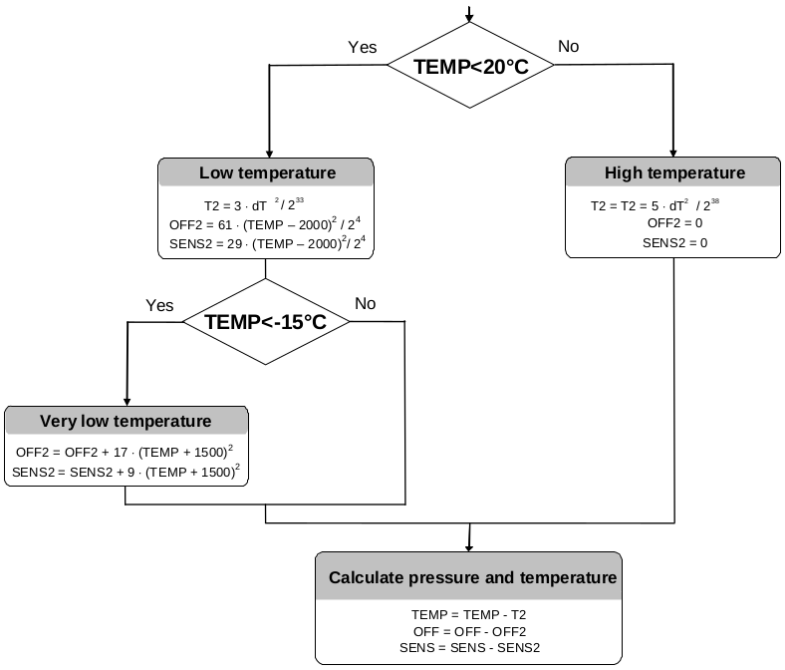


Figura 22.3: PCB modular electrónica varios

Capítulo 23

Algoritmo de medición del Barómetro




Yes
TEMP<20°C
No
Low temperature
T2 = 3 · dT ² / 2³³
OFF2 = 61 · (TEMP – 2000)² / 2⁴
SENS2 = 29 · (TEMP – 2000)² / 2⁴
High temperature
T2 = T2 = 5 · dT² / 2³⁸
OFF2 = 0
SENS2 = 0
Yes
TEMP<-15°C
No
Very low temperature
OFF2 = OFF2 + 17 · (TEMP + 1500)²
SENS2 = SENS2 + 9 · (TEMP + 1500)²
Calculate pressure and temperature
TEMP = TEMP - T2
OFF = OFF - OFF2
SENS = SENS - SENS2

Parte IX

Bibliografía

Bibliografía

[1] Paternain - Rosa - Tailanián, *"Implementación de un UAV con arquitectura de cuadricóptero"*, Facultad de Ingeniería - Universidad de la República, 2012.

[2] Berruti - Falkenstein - Favaro, *"Vuelo autónomo de un cuadricóptero"*, Facultad de Ingeniería - Universidad de la República, 2015.

[3] Hudson - Hagan - Demuth, *Neural Network Toolbox Users Guide*, 2015.

[4] Hudson - Hagan - Demuth, *Neural Network Design*, ebook 2nd Edition.

[5] Norgaard, *Neural Network Based Control System Design TOOLKIT*, Technical University of Denmark, 1997.

[6] Lewis, *Neural Networks in Feedback Control Systems*, The University of Texas at Arlington, 2005.

[7] Mostafa - Magdon-Ismail Tien Lin, *Learning from data*, Rensselaer Polytechnic Institute, USA, 2012.

[8] Grover - Y.C.Hwang, *"Introduction to Random Signals and Applied Kalman Filtering"*, USA, 2012.

[9] https://www.boschsensortec.com/bst/products/all_products/bno055

[10] https://www.invensense.com/products/motiontracking/9axis/mpu9250/

[11] http://www.te.com/usaen/productCATBLPS0037.html

[12] https://www.tindie.com/products/onehorse/bno0559axismotionsensorwith-hardwarefusion/?pt=full_prod_search

[13] https://www.ublox.com/en/product/neo6series

[14] Bernstein, Jonathan, *"An Overview of MEMS Inertial Sensing Technology"*, Sensors Weekly, February 1, 2003.

[15] Ramsden, Edward, *"Hall-effect sensors: theory and applications (2, illustrated ed.)"*, Elsevier, 2006.

[16] Mohinder S. Grewal - Angus P. Andrews, *Kalman Filtering*, Wiley, 2015.

[17] W. T. Higgins, "A Comparison of Complementary and Kalman Filtering," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-11, no. 3, pp. 321-325, May 1975.

[18] Robert Mahony *et. al. Complementary filter design on the special orthogonal group SO(3)*, Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005 Seville, Spain, December 12-15, 2005

[19] Alonzo Kelly, *Mobile robotics: mathematics, models and methods*, USA, 2013.

[20] Michael A. Johnson - Mohammad H. Moradi, *PID Control: New Identification and Design Methods*, 2005.

[21] Siciliano and O. Khatib, *Springer handbook of robotics*, 1st ed. Berlin: Springer, 2008.

[22] Lester E. Dubins, *On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents*, American Journal of Mathematics, 79(3):497-516, 1957.

[23] Shkel - Lumelsky, *Classification of the dubins set*, Robotics and Autonomous Systems 34, 34:179-202, 2001.

[24] Sousa - Saripalli, *An evaluation of uav path following algorithm.*, 2013 European Control Conference (ECC), 2013.

[25] Jack B. Kuipers, *Quaternions and rotation sequences : a primer with applications to orbits, aerospace, and virtual reality*, English, Book, Illustrated edition, 1999.

[26] On Quaternions; or on a new System of Imaginaries in Algebra (letter to John T. Graves, dated October 17, 1843). 1843

[27] G. Franklin, J. Powell and M. Workman, *Digital control of dynamic systems*, 1st ed. Menlo Park, Calif.: Addison-Wesley, 1998.

[28] H. Hirata and J. D. Powell, "Sample Rate Effects on Disturbance Rejection for Digital Control Systems," *1990 American Control Conference*, San Diego, CA, USA, 1990, pp. 1137-1145.

[29] Cybenko G, *"Approximations by superpositions of sigmoidal functions"*, Mathematics of Control, Signals, and Systems, 2 (4), 303-314, 1989.

[30] Warren - Pitts, *"A Logical Calculus of Ideas Immanent in Nervous Activity"*, Bulletin of Mathematical Biophysics. 5 (4): 115–133. 1943.

[31] Martin T Hagan, Howard B Demuth, Mark H Beale, Orlando De Jesús *Neural Network Design*, 2nd ed. 2014.

[32] Simon S Haykin, *"Neural networks, a comprehensive foundation"*, Prentice Hall, 2008.

[33] James R. Clynch, *"Radius of the Earth - Radii Used in Geodesy"*, 2006.

[34] *"HC-SR04 Ultrasonic Range Finder"*, http://accudiy.com

[35] Control, windup: https://en.wikipedia.org/wiki/Integral_windup

[36] http://www.x-io.co.uk/res/doc/madgwick_internal_report.pdf

[37] Control, windup: http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/

[38] https://oscarliang.com/best-looptime-flight-controller/

[39] https://github.com/betaflight/betaflight/wiki/Frequently-Asked-Questions#what-is-oneshot125-oneshot42-and-multishot-and-how-do-these-relate-to-max_throttle-and-looptime-

[40] https://www.arduino.cc/en/Main/Software

[41] https://www.pjrc.com/teensy/teensyduino.html

[42] http://energia.nu/

[43] Getting Started with the Tiva. TM4C123G LaunchPad Workshop: http://software-dl.ti.com/trainingTTO/trainingTTO_public_sw/GSW-TM4C123G-LaunchPad/TM4C123G_LaunchPad_Workshop_Workbook.pdf

[44] https://github.com/madsci1016/Arduino-EasyTransfer

[45] http://statweb.stanford.edu/ candes/acm116/Handouts/Kalman.pdf