UNIVERSIDAD INTERNACIONAL DEL ECUADOR

FACULTAD DE CIENCIAS TÉCNICAS

ESCUELA DE INGENIERÍA MECATRÓNICA

SISTEMA DE TRANSPORTE DE CARGAS MEDIANTE LA UTILIZACIÓN DE DRONES

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN MECATRÓNICA

SEBASTIÁN FRANCISCO CUESTA DELGADO

DIRECTOR: ING. CRISTINA GISELLE OSCULLO NARANJO, MSc

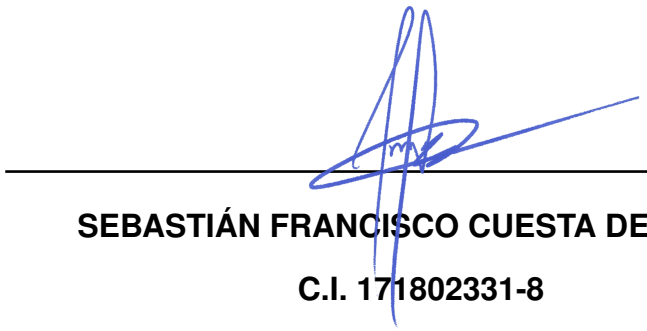
D. M. Quito,

2022

DECLARACIÓN

Yo SEBASTIÁN FRANCISCO CUESTA DELGADO, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que se ha investigado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Universidad Internacional del Ecuador, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por normativa institucional vigente.



SEBASTIÁN FRANCISCO CUESTA DELGADO

C.I. 171802331-8

CERTIFICACIÓN

El docente de la Facultad de Ciencias Técnicas, escuela de Ingeniería Mecatrónica Ingeniera CRISTINA GISELLE OSCULLO NARANJO encargado de la revisión del documento final,

CERTIFICA QUE:

El proyecto de investigación denominado "SISTEMA DE TRANSPORTE DE CARGAS MEDIANTE LA UTILIZACIÓN DE DRONES", fue desarrollado por el Sr. SEBASTIÁN FRANCISCO CUESTA DELGADO y ha sido debidamente revisado y está en condiciones de ser entregado para que siga lo dispuesto por la carrera de Ingeniería Mecatrónica, correspondiente a la sustentación y defensa del mismo.

Cristina Osculos

Ing. CRISTINA GISELLE OSCULLO NARANJO, MSc

DIRECTORA DE PROYECTO

Dedico a mis padres por su motivación y apoyo incondicional.

Agradecimientos

Agradezco a mis padres, Juan y Susana, los cuales me han enseñado que en esta vida vale la pena luchar por las metas, que me han acompañado y me han apoyado en todos los pasos de este camino, por su amor incondicional.

A mis hermanos José Antonio y Rafaela, por siempre estar a mi lado y ayudarme a nunca perder de vista las cosas más importantes.

A mis compañeros con los cuales con los cuales he compartido el camino recorrido en esta carrera y me enseñaron que las respuestas a los problemas se pueden encontrar desde distintas perspectivas.

A mis profesores por el conocimiento brindado en cada una de sus clases, por el cariño a la profesión y por su paciencia.

A los amigos que he hecho en la universidad, muchos de los cuales he llegado a querer como parte de mi familia, por todos los momentos y experiencias que hemos vivido juntos.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

SISTEMA DE TRANSPORTE DE CARGAS MEDIANTE LA UTILIZACIÓN DE DRONES

1. Tema

Implementar un sistema de transporte de cargas mediante la utilización de drones semi-autónomos.

2. Objetivos

2.1. General

Diseñar y construir un sistema de transporte de cargas de hasta 0,5 kg mediante la utilización de drones.

2.2. Específicos

- Investigar y documentar productos similares relacionados al proyecto planteado.
- Investigar el funcionamiento y el control necesarios para maniobrar un dron autónomo.
- Documentar normas y estándares aplicables para la utilización de drones de carga.
- Diseñar y dimensionar mecanismos que sean capaces de transportar una carga establecida de hasta 0.5 kg.
- Construir un prototipo del sistema de transporte con un dron semi-autónomo.
- Establecer y ejecutar un protocolo de pruebas del funcionamiento del sistema.

3. Problema

La pandemia causada por el virus COVID-19 ha cambiado las prioridades actuales de la industria y comercio, debido a la necesidad de continuar el funcionamiento de los diferentes establecimientos comerciales, los cuales se ven afectados por las restricciones de contacto

entre personas, es necesario implementar una nueva forma de realizar entregas minimizando la interacción humana tanto en zonas de difícil acceso como dentro de las ciudades. Al incrementar la demanda de repartidores los costos de entrega aumentan por lo que es necesario un sistema de transporte que permita facilitar y reducir los precios de transporte de productos esenciales.

4. Hipótesis

Implementar un sistema de transporte de cargas utilizando drones que permitirá mover cargas con un peso de hasta 500 g con un nivel de vuelo controlado y sostenido, las rutas a seguir serán generadas mediante un sistema de nodos establecidos por coordenadas GPS, el vuelo se realizará de manera autónoma reduciendo de esta manera los costos de transporte, logrando enviar paquetes con una menor necesidad de contacto humano y facilitando las entregas en áreas de difícil acceso. Las pruebas de funcionamiento se realizarán en la ciudad de Quito.

5. Estudio Teórico

La creciente demanda y popularidad de los UAVs (vehículos aéreos no tripulados) ha llevado al desarrollo de nuevas tecnologías en el área, dando como resultado nuevas aplicaciones para las mismas permitiendo la utilización de este tipo de dispositivos a un público más general, para estos fines se destaca una clasificación especifica de los UAV's, los drones también conocidos como multirotores caracterizados por su maniobrabilidad y versatilidad, lo que los convierte en candidatos ideales para su uso en una amplia variedad de trabajos. Para comprender las ventajas de estos UAV es necesario describir en un ámbito más amplio su clasificación, así como también definir de manera general a este tipo de vehículos.

5.1. Clasificación de los UAV's

Las siglas UAV por su nombre en inglés unmanned aerial vehicle es definido como un vehículo o aeronave el cual no tiene la necesidad de contar con tripulación por lo que debe

ser controlado de manera remota o contar con la tecnología necesaria para poder realizar las maniobras de vuelo de manera autónoma, a lo largo de la historia del desarrollo de los UAV se han generado varias soluciones para lograr estos objetivos generando de esta manera variaciones en la implementación de estos sistemas, dando características únicas para cada una de las clasificaciones de UAV, entre las que podemos encontrar autonomía, dimensiones, peso, capacidad de carga, fuerza de empuje, entre otras, según [20] existen varias formas de clasificar los UAV, de manera general las categorías existentes para las mismas son las siguientes: Micro y mini UAVs de rango corto, UAVs ligeros de rango corto, UAVs ligeros de medio rango, UAVs promedio, UAVs pesados de rango medio, UAVs pesados de largo alcance y aeronaves no tripuladas como la presentada en la Figura 1 , en el caso del presente proyecto se entra en la categoría de Micro y mini UAVs de rango corto, para el caso especifico de los drones es necesario especificar un mayor número de categorías menos generales mostradas en la Tabla 1, estas categorías son propuestas por [14].

Figura 1. MQ-9 Reaper, Fuente: [1]

Tabla 1. Clasificación de drones según su peso, Fuente: [14]

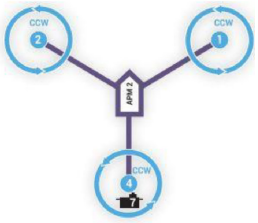
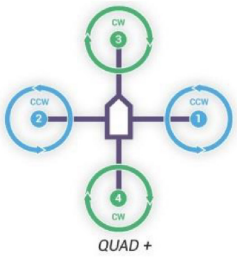
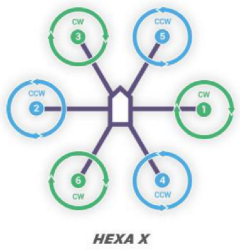
Clase	Tipo	Rango de peso
Clase I(a)	Nano drones	$P \leq 200$ g.
Clase I(b)	Micro drones	200 g $< P \leq 2$ kg.
Clase I(c)	Mini drones	2 kg $< P \leq 20$ kg.

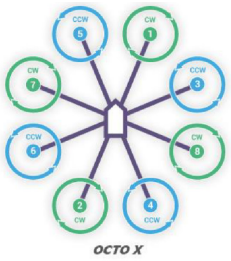
Clase	Tipo	Rango de peso
Clase I(d)	Drones pequeños	20 kg $<$ P $\leq$ 150 kg.
Clase II	Drones tácticos	150 kg $<$ P $\leq$ 600 kg.
Clase III	Drones de ataque	P $>$ 600 kg.

5.2. Características de los drones

Los drones también conocidos como multirotores son populares debido a su bajo costo de fabricación de además de presentar una gran versatilidad permitiendo la adaptación de los mismos para varios fines, existen varios tipos de multirotores, generalmente clasificados por el número de motores que utiliza para maniobrar, esto da características únicas a cada una de las configuraciones presentadas en la Tabla 2 basadas en [15]. Además de las configuraciones estándar también existen otro tipo de configuraciones como las mostradas en la Figura 2.

Tabla 2. Configuraciones estándar, Fuente: [15]

# Motores	Nombre	Ventajas	Desventajas	Imagen
3	Tricopter	Al contar con menor cantidad de motores presenta un costo menor así como también una mayor facilidad en la construcción	Cuenta con baja estabilidad y baja potencia para levantar vuelo por lo que es necesario utilizar baterías de menor tamaño	
4	Quadcopter	Gran cantidad de documentación, mayor capacidad de carga y tiempo de vuelo. la estabilidad no es un problema en esta configuración	Pérdida de capacidad de vuelo en caso de falla de un motor, es necesario motores especializados para levantar cargas	
6	Hexacopter	Capacidad de vuelo limitada cuando falla un motor, mayor capacidad de carga sin la necesidad de utilizar motores especializados	Mayor complejidad de implementación, mayor costo, mayor consumo energético que un Quadcopter	

# Motores	Nombre	Ventajas	Desventajas	Imagen
8	Octocopter	Capacidad de vuelo completa cuando falla un motor, mayor capacidad de carga sin la necesidad de utilizar motores especializados	Mayor peso, costo elevado, mayor consumo energético que un Hexacopter	

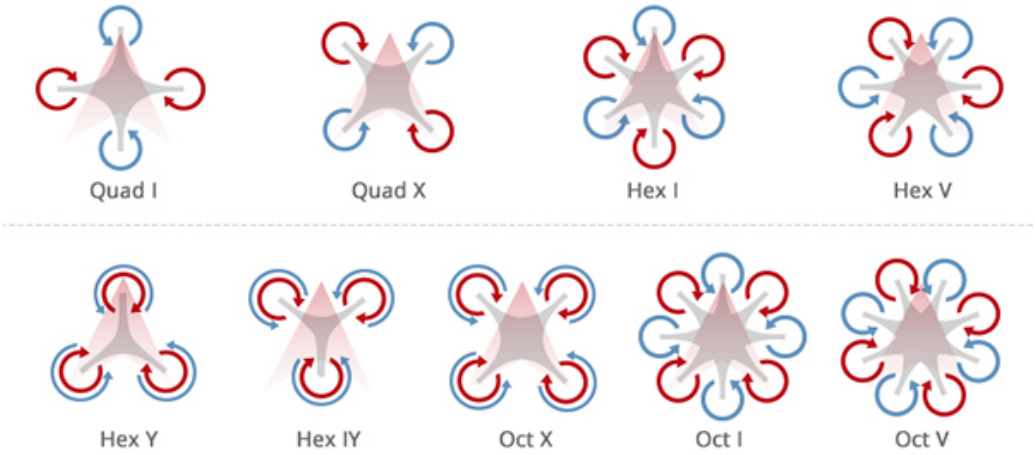


Figura 2. Configuraciones no estándar, Fuente: [2]

Otras formas de categorizar los drones es debido al tamaño de sus hélices las cuales comúnmente se pueden encontrar entre 3 y 10 pulgadas para drones estándar. Existen distintos tipos de control para los drones según [21] se pueden dividir en:

- **Autónomo:** Hace uso de sensores integrados para poder realizar el vuelo a su destino, no depende de un piloto.
- **Monitorizado**: Se requiere un piloto que comande las acciones generales del dron (subir o bajar altura, aumentar o disminuir velocidad) pero el plan de vuelo es preprogramado.
- **Supervisado:** El piloto es encargado de controlar en su totalidad el dron, se pueden realizar algunas tareas de manera automática, como despegues, aterrizajes y mantener la altura.

- **Preprogramado:** se planean las rutas y acciones antes de comenzar el vuelo, esto puede resultar peligroso si no se conoce a la perfección la ruta de vuelo debido a que el dron no podrá reaccionar a escenarios imprevistos.
- **Controlado Remotamente:** el alcance es corto y generalmente se utiliza radiofrecuencia para realizar el control el cual depende en su totalidad del piloto.

5.3. Componentes de un Dron

5.3.1. Motores

Los multirotores se ven en la necesidad de tener una alta capacidad de maniobrabilidad por lo que se ha popularizado la implementación de estos mediante la utilización de motores sin escobillas (brushless) los cuales están compuestos por un rotor (parte móvil) y un estator (parte fija), los elementos que conforman cada una de estas partes se pueden ver en la Figura 3 , estos funcionan con corriente continua, lo cual es un requerimiento para el uso de baterías, este tipo de motor permite un cambio rápido de polaridad dando como resultado el cambio de sentido de giro del motor, estos motores se definen según el factor determinado como kV el cual determina el número de revoluciones por minuto a las que puede girar el motor por cada voltio aplicado al mismo, generalmente también se define el voltaje máximo del motor mediante el tipo de batería para los que se diseñan. Debido a que estos motores no tienen escobillas pueden llegar a un mayor número de RPM y torque, pero para su control es necesario contar con variadores.

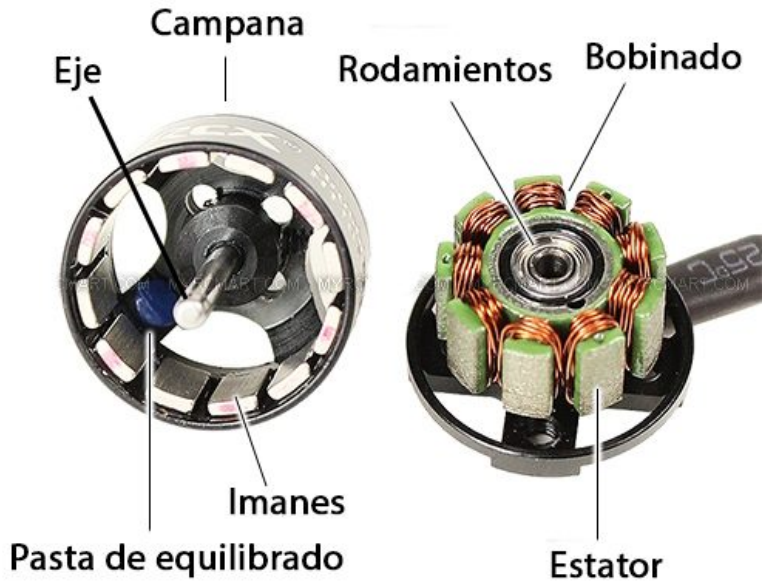


Figura 3. Elementos constitutivos de un motor sin escobillas

5.3.2. Variadores

También conocidos por sus siglas en ingles ESC (Electronic Speed Controller) estos circuitos electrónicos permiten controlar el sentido y velocidad de giro de los motores sin escobillas, gracias a esto se puede realizar frenado dinámico de estos, generalmente el controlador de vuelo se comunica con los ESC y no directamente con los motores, estos son los encargados de dar las señales de control, es importante garantizar el correcto funcionamiento de los variadores debido a que una falla en la sincronización entre el motor y el ESC puede causar que el dron no sea capaz de volar. El diagrama de conexiones se presenta en la Figura 4.

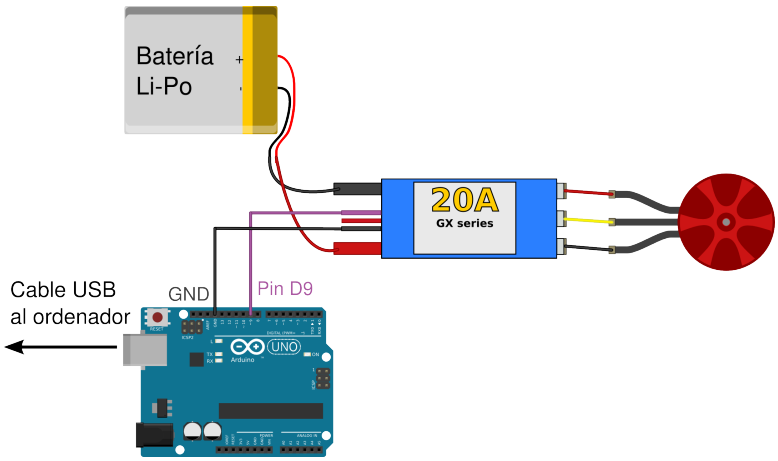


Figura 4. Diagrama de conexiones de un ESC a un módulo Arduino UNO, Fuente: [3]

5.3.3. Propelas

Este componente junto con los motores define la fuerza de empuje del UAV así como también su eficiencia, es decir el consumo energético del sistema aumentará dependiendo del dimensionamiento de las hélices. Estas están definidas por dos parámetros esenciales para el diseño de un dron:

Longitud: Es la distancia entre las puntas de la hélice (también llamado diámetro de la hélice), a mayor tamaño el empuje es mayor, por lo que puede cargar una mayor cantidad de masa, pero un aumento en la longitud implica un mayor consumo energético.

Paso: Es la distancia teórica de avance en el eje de rotación cuando la hélice completa una revolución, un mayor paso generara un más empuje y a su vez consumirá la batería en menor tiempo que una hélice de menor paso.

5.3.4. Baterías

Las baterías más utilizadas en la implementación de UAVs para uso civil son las baterías de iones de litio también conocidas como baterías LIPO, tienen como ventajas su peso reducido así como su elevada capacidad de almacenamiento, estas baterías son populares debido a la densidad energética que estas presentan, sus tasas de descarga alcanzan las decenas de amperios, esto permite el correcto funcionamiento de los motores. La constitución de este tipo de baterías esta determinada por el número de celdas que estas contienen, dependiendo de esto se define la capacidad de carga y voltaje entregado por las mismas.

5.3.5. Controladores de vuelo

Este circuito es el encargado de determinar la posición en la que se encuentra el dron para poder cumplir con las instrucciones de vuelo, esto se logra a través de sensores instalados en el dron, generalmente incorporan un sistema PID para el control de motores y como mínimo un giroscopio para poder determinar la orientación espacial del dron, pero la complejidad de estos circuitos puede aumentar así como su capacidad de control dependiendo de la precisión y necesidades de la tarea a realizarse, puede incorporar sistemas de GPS, sensores de proximidad, y unidades de medición inercial, la implementación de estos

sensores también depende del tipo de control que vaya a utilizar ya que un UAV con control autónomo necesitará más datos para poder cumplir con el plan de vuelo.

5.4. Firmware

El firmware se refiere el programa informático embebido en el controlador de vuelo el cual hace posible el control de la posición, orientación y comportamiento general del dron según las señales de mando que reciba, generalmente cuentan con un controlador PID incorporado en la programación, estos se especializan en diferentes tareas dependiendo de la aplicación que se busca en el dispositivo, esto puede ser vuelo a altas velocidades, actividades de reconocimiento que requieren sensores especializados (como es el caso de la agricultura), tareas de vuelo autónomo por GPS e incluso enjambres de drones, debido a la gran diversidad de objetivos con los que se desarrollan los firmwares de UAVs es importante analizar las opciones existentes actualmente, a continuación se realiza un análisis de las características de las opciones de firmware con mayor relevancia en el proyecto:

5.4.1. Cleanflight

Este firmware de código libre fue creado con el objetivo de generar código fácil de configurar, actualizar y mantener, y de esta manera impulsar el desarrollo de UAVS, cuenta con una amplia gama de funciones, y su constante proceso de actualizaciones permite que sea compatible con la mayoría de los controladores del mercado. Cleanflight es el antecesor de Betaflight y INAV, los dos firmwares más populares en la actualidad. Algunas de sus funciones son las siguientes:

- Múltiples protocolos de control de ESC (Dshot, Multishot y Oneshot)
- Compatible con controladores tipo F7, F4 y F3
- Telemetria
- OSD y VTX
- Configuración de controlador PID durante el vuelo
- Puerto serial

5.4.2. Betaflight

Fundado como una plataforma de pruebas para las versiones beta del firmware Cleanflight, actualmente se considera el más popular entre usuarios de multirrotores debido a su facilidad de configuración, esto se debe a que se cuenta con un software con el objetivo de realizar los cambios necesarios para un UAV específico de manera sencilla, actualmente se encuentra en la versión 4,2, algunas de sus características son las siguientes:

- Modos de vuelo que permiten el uso del giroscopio para auto nivelado
- Control semiautónomo con GPS
- Control PID ajustable
- OSD y VTX
- Telemetría de los ESC
- Puerto serial
- Configuración de la velocidad de reloj del microcontrolador
- Diferentes modos de control de ESC (Dshot, Multishot y Oneshot)

Aunque es un firmware que contiene la mayoría de las características necesaria para la mayoría de las aplicaciones, carece de funciones especializadas como es la configuración de misiones autónomas por GPS, además los cambios que requieran una configuración avanzada pueden ser complicados de realizar, por lo que se recomienda que este software se use en UAVs que no requieran de estas funciones.

5.4.3. INAV

Uno de los predecesores de Cleanflight, este firmware se encuentra orientado a UAVs que tengan como requerimiento la navegación y vuelo autónomo, así como también misiones GPS y establecimiento de bases para el retorno seguro, es comúnmente utilizado en drones de topografía pero al ser de código abierto permite la configuración de este para distintas necesidades. Cuenta con las siguientes características:

- Compatible con controladores de vuelo tipo F4 y F7
- Preconfigurado en su mayor parte
- Modos de mantener posición, mantener altura, volver a base de manera autónoma y mi-

siones GPS

- Compatible con UAVs tipo ala fija, multirrotores, e inclusive drones acuáticos y dispositivos experimentales
- Compatible con sensores tipo GPS, sonar, ESC, de temperatura, tubo de Pitot y lidar
- Protocolos de control de ESC (multishot y Dshot)
- OSD
- Telemetria
- Configuración de filtro para giroscopio
- Programación a través de GUI (soporte para funciones y variables globales)

Aunque es considerado uno de los mejores firmwares para drones autónomos aún tiene como requerimiento que el control que da la señal al dron se encuentre dentro del rango de señal, incluso si se quiere que el UAVs se mueva de manera autónoma, por lo que es una limitante para este proyecto, es importante recalcar que es necesario el uso de un barómetro para la correcta medición de altura.

5.5. Drones de Entrega

Desde su creación, los drones se han desarrollado teniendo en mente un sin número de aplicaciones, desde drones de monitoreo de plantaciones agrícolas hasta drones de rescate pero debido al precio de los mismos no se ha llegado a volver una tecnología común, en los últimos años debido a los avances en miniaturización de circuitos electrónicos y otros mejoramientos en los sistemas necesarios para la implementación de los multirotores como es el poder de procesamiento de los microprocesadores, han conseguido que estos sistemas sean cada vez más accesibles al público general por lo que las aplicaciones comerciales han comenzado a ganar fuerza. En el año 2016 Amazon anuncio públicamente su primera entrega exitosa de productos comerciales utilizando drones, desde entonces la tecnología utilizada para implementar estos sistemas ha ido mejorando día tras día pero en la actualidad sigue siendo una opción poco común debido a la complejidad de implementación de los mismos, pero cada vez esta más cerca de que los drones de entrega de paquetes se vuelvan una realidad para la mayoría de personas del mundo, expandiendo las áreas de

entrega y reduciendo los tiempos de entrega de horas o días a tan solo unos minutos.

5.5.1. Productos Existentes

Amazon Prime Air

Este servicio creado por Amazon permite la entrega de paquetes de pequeño tamaño a poblaciones donde se esta probando el sistema, inicia con una orden a través de la pagina web de la compañía, este pedido es manejado por el personal de una bodega y colocado en el dron de entrega, este dron tiene control autónomo y realiza la entrega en el patio del cliente. Actualmente Amazon cuenta con tres modelos, dos multirotores como se ven en la Figura 5 capaces de llevar cargas de hasta 2,25 kg a una altura de 120 metros, el otro modelo es un híbrido entre un multirotor y un sistema tipo avión, el cual es utilizado para entregas en un radio de 24 km. Estos drones hacen uso de sensores, GPS y visión artificial para el posicionamiento del UAV así como también para detección de obstáculos.

Figura 5. Dron de Amazon Prime Air MK4, Fuente: [4]

Valkyrie Heavy Pro

Este dron desarrollado por la empresa Valkyrie se define como un dron de carga universal lo que quiere decir que cuenta con características que le permiten el funcionamiento continuo durante varias horas sin carga y puede llegar a cubrir distancias de hasta 20 km con una carga de 10 kg, su distancia disminuye a medida que la carga aumenta logrando 15 km con cargas de 20 kg y 3 km con cargas de 30 kg, la carga máxima que puede llevar este UAV es de 50 kg, tiene un modelo tipo quadcopter como se ve en la Figura 6, tiene un

precio mínimo de 3500 dolares para la versión sin accesorios.

Figura 6. Dron Valkyrie Heavy Pro, Fuente: [5]

- Rango/autonomía:
  - 10 kg/20 km
  - 20 kg/15 km
  - 30 kg/3 km
- Tiempo de vuelo: 90 minutos
- Configuración: Quadcopter
- Batería: 44000 mAh, 22V, 25C
- Hélices: 3210
- Peso sin Carga: 7,6 kg
- Navegación: Manual

Delivery Canada Sparrow

Figura 7. Dron Delivery Canada Sparrow, Fuente: [6]

- Rango máximo: 20 km
- Velocidad máxima: 80 km/h
- Carga Max: 4 kg
- Configuración: Octacopter
- Navegación: GPS (Semiautónoma)

A2Z Drone Delivery RDSX

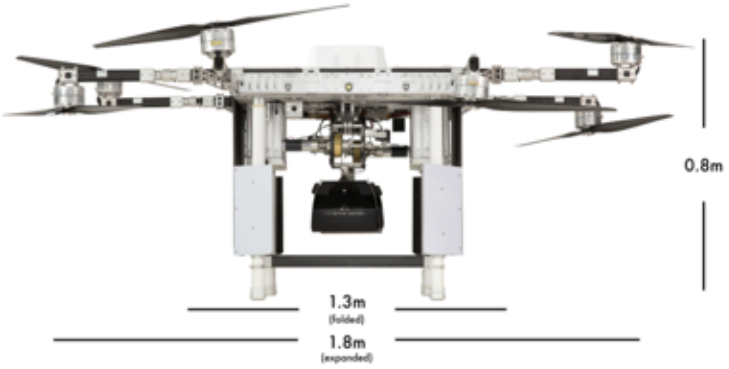


Figura 8. A2Z Drone Delivery RDSX, Fuente: [7]

- Configuración: Octacopter
- Navegación: Manual/ Semiautónoma

- Rango:
    - Radio Control (manual): 1,5 km
    - Estación de control (semiautónomo): 5 km

En la Tabla 3 se resumen las características del los drones de carga comerciales analizados

Tabla 3. Características generales de multirotores de carga disponibles en el mercado

Nombre	Configuración	Rango Max	Carga Max
Amazon Prime Air	Octacopter	24km	2,25kg
Valkyrie Heavy Pro	Quadcopter	20km	30kg
Delivery Canada Sparrow	Octacopter	20km	4kg
A2Z Drone Delivery RDSX	Octacopter	30km	4kg

5.6. Búsqueda de rutas

El Proyecto presenta la necesidad de desarrollar un programa capaz de determinar el camino que debe seguir el UAV para lo cual se debe utilizar un algoritmo de búsqueda, el algoritmo tomará cada base como un punto de ruta por el cual debe pasar, existen dos tipos principales de algoritmos de búsqueda, los conocidos como búsqueda en árbol y algoritmos de búsqueda en grafos, en este caso ya se conocen los nodos de la ruta por lo que los algoritmos basados en grafos son mucho más eficientes, a continuación se describen los algoritmos más utilizados:

5.6.1. Algoritmo de Dijkstra

Este algoritmo comienza con un grupo de nodos adyacentes al nodo inicial, y determina la distancia de estos con el objetivo, una vez encontrado el nodo se realiza el mismo proceso, pero se utilizan los nodos adyacentes al segundo, este proceso se repite hasta llegar al objetivo, este proceso no encuentra la ruta de la manera más rápida ni la ruta más corta, por lo que se suelen realizar iteraciones de este.

5.6.2. Algoritmo A*

Este algoritmo es una mejora del algoritmo de Dijkstra, utiliza métodos heurísticos para determinar el peso de cada nodo y de esta manera mejora las posibilidades de encontrar la ruta más corta de manera rápida y en la primera iteración de manera significativa, debido a esto se pueden eliminar varios nodos del análisis por lo que la velocidad de cómputo también aumenta, es importante calibrar la heurística de manera correcta debido a que si se la hace muy rígida ya no se garantiza la ruta más óptima, para esto suele utilizarse dos tipos de medición de distancia, la primera es la distancia euclidiana entre dos puntos, y la segunda es la distancia Manhattan que se define como la suma de los valores absolutos de las diferencias entre las coordenadas X y Y, para determinar la heurística se realiza una media ponderada de estas distancias, también se pueden agregar factores particulares para cada caso como por ejemplo el trafico en una zona particular, la limitación de este algoritmo radica en su capacidad de manejar rutas multi-agentes debido a que no se toma en cuenta las colisiones entre ellos, únicamente con su entorno. En la Figura 9 se puede ver una comparación entre los resultados de ambos algoritmos.

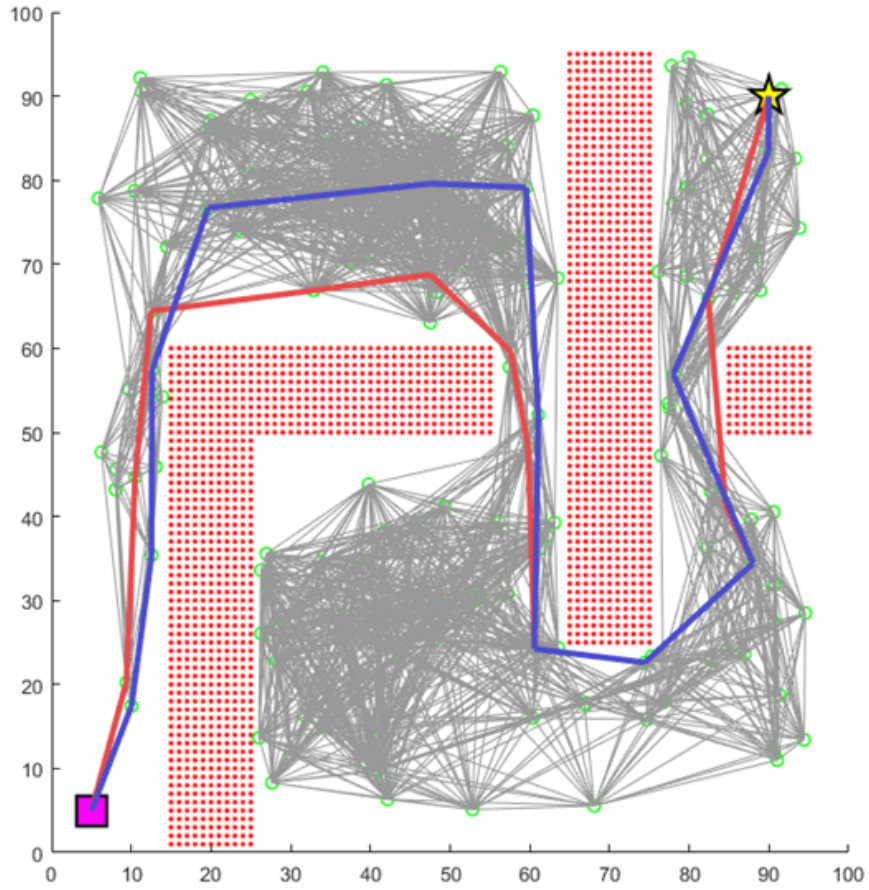


Figura 9. Algoritmo A* (azul) vs algoritmo Dijkstra (rojo)

5.7. Materiales compuestos

Debido a que se debe balancear el peso de la estructura con su capacidad de llevar la carga establecida se decide utilizar materiales compuestos para la estructura, en especifico polímero reforzado con fibra de carbono, los materiales compuestos están conformados por dos partes, la fibra la cual aporta propiedades de resistencia y la matriz permite obtener la geometría deseada así como también soportar los esfuerzos de compresión, transmiten los esfuerzos de corte a las fibras. estos materiales son anisótropicos esto quiere decir que tienen comportamiento direccional en sus propiedades mecánicas, se puede conseguir un comportamiento altamente isotrópico (cuasi isotrópico) si se utiliza laminados con fibras en distintas orientaciones.

5.7.1. Nomenclatura de laminados

Reglas:

1. La orientación siempre se indica de capa superior a inferior en laminados no simétricos y se asigna un subíndice T
2. En laminados simétricos comienza con la capa superior y termina con la capa media (la anterior a la fibra neutra o eje de simétrica), se utiliza un subíndice S para indicar que es simétrico.
3. En laminados simétricos si la ultima capa antes del plano medio no se repite se debe colocar una barra encima del número de capa lo cual indica que esta no se repite
4. Las capas contiguas con el mismo ángulo se indican con un subíndice numérico
5. Si las capas adyacentes se encuentran orientadas en ángulos positivos y negativos se utilizar el termino $\pm\theta$
6. Cuando un sublaminado se repite se debe expresar entre corchetes con un subíndice numérico que indica el número de repeticiones.

Ejemplo:

En la Figura 10 se muestra un ejemplo de un laminado simétrico sin repetición en el plano medio y con sublaminados, su nomenclatura esta descrita en (1).

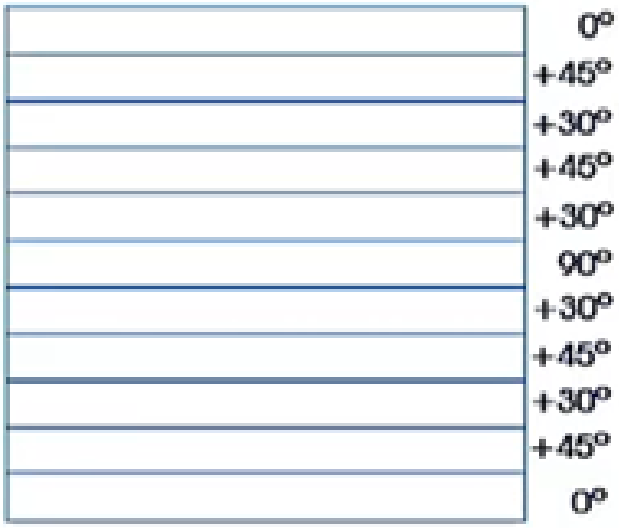


Figura 10. Ejemplo de laminado simétrico, Fuente: [8]

$$[0/\{45/30\}_2/(\bar{90})]_s \quad (1)$$

5.7.2. Clasificación de laminados de material compuesto

- Laminado Equilibrado: Para cada lamina con orientación $+\theta$ existe una con orientación $-\theta$
- Laminado Simétrico: Para cada lamina localizada a una distancia h_i de la fibra neutra, existe una capa con la misma orientación, espesor y composición situada a una distancia $-h_i$ de la fibra neutra
- Laminado antisimétrico: Para cada lamina localizada a una distancia h_i de la fibra neutra y con orientación θ, existe una capa con el mismo espesor y composición situada a una distancia $-h_i$ de la fibra neutra, pero con una orientación $-\theta$
- Laminado unidireccional: Todas las capas cuentan con un mismo ángulo.
- Laminado Angular: Las orientaciones de las capas no coinciden con las direcciones principales (0 grados y 90 grados [X, Y]) por lo que están orientados en ángulos $+\theta$ y$-\theta$, estos laminados presentan una mayor resistencia fuerzas de cortadura.
- Laminado Cruzado: “CROSS-PLY”, las capas siempre ocupan las direcciones principales (0 grados y 90 grados [X, Y])
- Laminado cuasi-isotrópico: Cuenta con simetría en tres planos, son siempre simétrico y equilibrados, la condición de orientación de las capas es que la distancia angular sea la misma tal que $\Delta\theta = 360/n\ [n > 3]$

6. Diseño conceptual

6.1. Análisis de alternativas

Se realiza el análisis de alternativas mediante el método de residuos ponderados, a los parámetros de cada una de las características analizadas se les da un valor entre 0 y 1, siendo 0 una ponderación negativa y 1 una ponderación positiva, estos valores se dan

mediante comparación entre las distintas alternativas por lo que un valor de 0,5 indica que no existe una diferencia significativa entre las alternativas en cuestión.

6.1.1. Configuración de los motores

Se realiza el análisis de la configuración de los motores mediante el método de promedios ponderados para los siguientes criterios:

- **Facilidad de control:** Complejidad requerida para poder controlar el vuelo del dron, aumenta con la cantidad de motores.
- **Estabilidad:** Capacidad de mantener una posición en el espacio y evitar movimientos o vibraciones involuntarios causados por fuerzas externas como puede ser el viento.
- **Redundancia:** Habilidad para mantener el vuelo y maniobrar en el aire en caso de perdida de motores o propelas.
- **Precio:** Costo de implementación de la estructura y componentes necesarios para su funcionamiento
- **Capacidad de carga:** Facilidad con la que se puede levantar peso, para esto se asume que se usaría la misma combinación motor-propela para todos los casos.

Tabla 4. Alternativas para la configuración de los motores

Alternativa 1	Alternativa 2	Alternativa 3	Alternativa 4
Tricopter	Quadcopter	Hexacopter	Octacopter

Evaluación del peso específico de cada criterio

Precio>C. Carga=F. Control>Estabilidad>Redundancia

Tabla 5. Evaluación del peso específico de cada criterio en configuración de motores

	F.Control	Estabilidad	Redundancia	Precio	C. Carga	$\sum+1$	Ponderación
F.Control	-	1	1	0	0,5	3,5	0,25
Estabilidad	0	-	1	0	0	2	0,14
Redundancia	0	0	-	0	0	1	0,07
Precio	1	1	1	-	1	4	0,29
C. Carga	0,5	1	1	0	-	3,5	0,25
					Suma	14	1

Evaluación del peso específico de criterio FACILIDAD DE CONTROL

Alternativa 2>Alternativa 4>Alternativa 3>Alternativa 1

Tabla 6. Evaluación de criterio FACILIDAD DE CONTROL

	Alternativa 1	Alternativa 2	Alternativa 3	Alternativa 4	$\sum+1$	Ponderación
Alternativa 1	-	0	0	0	1	0,10
Alternativa 2	1	-	1	1	4	0,40
Alternativa 3	1	0	-	0	2	0,20
Alternativa 4	1	0	1	-	3	0,30
				Suma	10	1

Evaluación del peso específico de criterio ESTABILIDAD

Alternativa 4>Alternativa 3=Alternativa 2>Alternativa 1

Tabla 7. Evaluación de criterio ESTABILIDAD

	Alternativa 1	Alternativa 2	Alternativa 3	Alternativa 4	$\sum+1$	Ponderación
Alternativa 1	-	0	0	0	1	0,10
Alternativa 2	1	-	1	1	4	0,40

	Alternativa 1	Alternativa 2	Alternativa 3	Alternativa 4	$\sum+1$	Ponderación
Alternativa 3	1	0	-	0	2	0,20
Alternativa 4	1	0	1	-	3	0,30
				Suma	10	1

Evaluación del peso específico de criterio REDUNDANCIA

Alternativa 4>Alternativa 3>Alternativa 2>Alternativa 1

Tabla 8. Evaluación de criterio REDUNDANCIA

	Alternativa 1	Alternativa 2	Alternativa 3	Alternativa 4	$\sum+1$	Ponderación
Alternativa 1	-	0	0	0	1	0,10
Alternativa 2	1	-	0	0	2	0,20
Alternativa 3	1	1	-	0	3	0,30
Alternativa 4	1	1	1	-	4	0,40
Suma					10	1

Evaluación del peso específico de criterio PRECIO

Alternativa 1>Alternativa 2>Alternativa 3>Alternativa 4

Tabla 9. Evaluación de criterio PRECIO

	Alternativa 1	Alternativa 2	Alternativa 3	Alternativa 4	$\sum+1$	Ponderación
Alternativa 1	-	1	1	1	4	0,40
Alternativa 2	0	-	1	1	3	0,30
Alternativa 3	0	0	-	1	2	0,20
Alternativa 4	0	0	0	-	1	0,10
				Suma	10	1

Evaluación del peso específico de criterio CAPACIDAD DE CARGA

Alternativa 4>Alternativa 3>Alternativa 2>Alternativa 1

Tabla 10. Evaluación de criterio CAPACIDAD DE CARGA

	Alternativa 1	Alternativa 2	Alternativa 3	Alternativa 4	$\sum+1$	Ponderación
Alternativa 1	-	0	0	0	1	0,10
Alternativa 2	1	-	0	0	2	0,20
Alternativa 3	1	1	-	0	3	0,30
Alternativa 4	1	1	1	-	4	0,40
				Suma	10	1

Evaluación Final

Tabla 11. Evaluación Final con Resultados por Prioridad

	F.Control	Estabilidad	Redundancia	Precio	C. Carga	$\sum$	Prioridad
Alternativa 1	0,03	0,01	0,01	0,11	0,03	0,19	3
Alternativa 2	0,10	0,04	0,01	0,09	0,05	0,29	1
Alternativa 3	0,05	0,04	0,02	0,06	0,08	0,24	2
Alternativa 4	0,08	0,06	0,03	0,03	0,10	0,29	1
					Suma	1	

Se selecciona la alternativa 2 la cual corresponde a la configuración tipo Quadcopter, como se indica en la Tabla11 existe un empate entre la alternativa 2 y la 4, debido a esto se toma como criterio de precio ya que este es considerado de mayor prioridad, otra de las ventajas es la facilidad de control en esta configuración.

6.1.2. Tipo de control

Se realiza el análisis del tipo de control mediante el método de promedios ponderados para los siguientes criterios:

- **Precio de Implementación:** Se refiere al costo de las distintas partes de cada tipo de control (sensores, microcontroladores, programas, etc.)
- **Experiencia Requerida:** Cantidad de capacitación necesaria por parte del operario del dron.
- **Complejidad:** Cantidad de sistemas necesarios para poder lograr el control del dron, una menor complejidad se relaciona con un menor tiempo de desarrollo.

Tabla 12. Alternativas para el tipo de control

Alternativa 1	Alternativa 2	Alternativa 3	Alternativa 4
Autónomo	Monitorizado	Supervisado	Preprogramado

Evaluación del peso específico de cada criterio

P. implementación>E.requerida>Complejidad

Tabla 13. Evaluación del peso específico de cada criterio en tipo de control

	P. Implementación	E. Requerida	Complejidad	$\sum+1$	Ponderación
P. Implementación	-	1	1	3	0,50
E. Requerida	0	-	1	2	0,33
Complejidad	0	0	-	1	0,17
			Suma	6	1

Evaluación del peso específico de criterio PRECIO DE IMPLEMENTACIÓN

Alternativa 3>Alternativa 4>Alternativa 2>Alternativa 1

Tabla 14. Evaluación de criterio PRECIO DE IMPLEMENTACIÓN

	Alternativa 1	Alternativa 2	Alternativa 3	Alternativa 4	$\sum+1$	Ponderación
Alternativa 1	-	0	0	0	1	0,10
Alternativa 2	1	-	0	0	2	0,20
Alternativa 3	1	1	-	1	4	0,40
Alternativa 4	1	1	0	-	3	0,30
				Suma	10	1

Evaluación del peso específico de criterio EXPERIENCIA REQUERIDA

Alternativa 1>Alternativa 4>Alternativa 2>Alternativa 3

Tabla 15. Evaluación de criterio EXPERIENCIA REQUERIDA

	Alternativa 1	Alternativa 2	Alternativa 3	Alternativa 4	$\sum+1$	Ponderación
Alternativa 1	-	1	1	1	4	0,40
Alternativa 2	0	-	1	0	2	0,20
Alternativa 3	0	0	-	0	1	0,10
Alternativa 4	0	1	1	-	3	0,30
				Suma	10	1

Evaluación del peso específico de criterio COMPLEJIDAD

Alternativa 3>Alternativa 4>Alternativa 2>Alternativa 1

Tabla 16. Evaluación de criterio COMPLEJIDAD

	Alternativa 1	Alternativa 2	Alternativa 3	Alternativa 4	$\sum+1$	Ponderación
Alternativa 1	-	0	0	0	1	0,10
Alternativa 2	1	-	0	0	2	0,20
Alternativa 3	1	1	-	1	4	0,40

	Alternativa 1	Alternativa 2	Alternativa 3	Alternativa 4	$\sum+1$	Ponderación
Alternativa 4	1	1	0	-	3	0,30
				Suma	10	1

Evaluación Final

Tabla 17. Evaluación Final con Resultados por Prioridad

	P. Implementación	E. Requerida	Complejidad	$\sum$	Prioridad
Alternativa 1	0,05	0,13	0,02	0,20	2
Alternativa 2	0,10	0,07	0,03	0,20	2
Alternativa 3	0,20	0,03	0,07	0,30	1
Alternativa 4	0,15	0,10	0,05	0,30	1
			Suma	1	

Se selecciona la alternativa 4 la cual corresponde al tipo de control preprogamado, como se ve en la Tabla17 existe un empate con la alternativa 3 la cual indica un sistema de control supervisado , sin embargo las operaciones autónomas posibles con ese sistema de control son limitadas por lo que la mayoría de decisiones quedan a manos del operador y esto implica un mayor numero de personas y de capacitación para poder implementar el sistema de transporte de cargas.

6.1.3. Controlador

Se realiza el análisis del controlador mediante el método de promedios ponderados para los siguientes criterios:

- **Tareas Autónomas:** Capacidad para almacenar,leer y ejecutar instrucciones preprogramadas que permitan el vuelo sin la necesidad de un operario.
- **Compatibilidad:** Se refiere a la habilidad de utilizar distintos elementos electrónicos que permitan el vuelo así como la cantidad de protocolos de comunicación que permitan leer los datos enviados por los sensores.

- **Precio:** Costo del módulo y de los componentes necesarios para su funcionamiento.
- **Peso:** Cantidad de peso del módulo y sus componentes,un mayor peso disminuye la capacidad de carga del dron.
- **Disponibilidad:** Se refiere a la posibilidad de adquisición del módulo.

Tabla 18. Alternativas para el controlador

Alternativa 1	Alternativa 2	Alternativa 3	Alternativa 4
NVIDIA Jetson	Raspberry pi	Arduino MEGA	Mateksys F722

Evaluación del peso específico de cada criterio

T.Autónomas>Disponibilidad>Precio=Peso>Compatibilidad

Tabla 19. Evaluación del peso específico de cada criterio en controlador

	T. Autónomas	Compatibilidad	Precio	Peso	Disponibilidad	$\sum+1$	Ponderación
T. Autónomas	-	1	1	1	1	5	0,31
Compatibilidad	0	-	0	0	0	1	0,06
Precio	0	1	-	1	0	3	0,19
Peso	0	1	0	-	1	3	0,19
Disponibilidad	0	1	1	1	-	4	0,25
					Suma	16	1

Evaluación del peso específico de criterio CAPACIDAD PARA TAREAS AUTÓNOMAS

Alternativa 1>Alternativa 4>Alternativa 2>Alternativa 3

Tabla 20. Evaluación de criterio CAPACIDAD PARA TAREAS AUTÓNOMAS

	Alternativa 1	Alternativa 2	Alternativa 3	Alternativa 4	$\sum+1$	Ponderación
Alternativa 1	-	1	1	1	4	0,40

	Alternativa 1	Alternativa 2	Alternativa 3	Alternativa 4	$\sum+1$	Ponderación
Alternativa 2	0	-	1	0	2	0,20
Alternativa 3	0	0	-	0	1	0,10
Alternativa 4	0	1	1	-	3	0,30
				Suma	10	1

Evaluación del peso específico de criterio COMPATIBILIDAD

Alternativa 3>Alternativa 2>Alternativa 4>Alternativa 1

Tabla 21. Evaluación de criterio COMPATIBILIDAD

	Alternativa 1	Alternativa 2	Alternativa 3	Alternativa 4	$\sum+1$	Ponderación
Alternativa 1	-	0	0	0	1	0,10
Alternativa 2	1	-	0	1	3	0,30
Alternativa 3	1	1	-	1	4	0,40
Alternativa 4	1	0	0	-	2	0,20
				Suma	10	1

Evaluación del peso específico de criterio PRECIO

Alternativa 3>Alternativa 4>Alternativa 2>Alternativa 1

Tabla 22. Evaluación de criterio PRECIO

	Alternativa 1	Alternativa 2	Alternativa 3	Alternativa 4	$\sum+1$	Ponderación
Alternativa 1	-	0	0	0	1	0,10
Alternativa 2	1	-	0	0	2	0,20
Alternativa 3	1	1	-	0	3	0,30
Alternativa 4	1	1	1	-	4	0,40
				Suma	10	1

Evaluación del peso específico de criterio PESO

Alternativa 4>Alternativa 3>Alternativa 2>Alternativa 1

Tabla 23. Evaluación de criterio PESO

	Alternativa 1	Alternativa 2	Alternativa 3	Alternativa 4	$\sum+1$	Ponderación
Alternativa 1	-	0	0	0	1	0,10
Alternativa 2	1	-	0	0	2	0,20
Alternativa 3	1	1	-	0	3	0,30
Alternativa 4	1	1	1	-	4	0,40
				Suma	10	1

Evaluación del peso específico de criterio DISPONIBILIDAD

Alternativa 3>Alternativa 2>Alternativa 4>Alternativa 1

Tabla 24. Evaluación de criterio DISPONIBILIDAD

	Alternativa 1	Alternativa 2	Alternativa 3	Alternativa 4	$\sum+1$	Ponderación
Alternativa 1	-	0	0	0	1	0,10
Alternativa 2	1	-	0	1	3	0,30
Alternativa 3	1	1	-	1	4	0,40
Alternativa 4	1	0	0	-	2	0,20
				Suma	10	1

Evaluación Final

Tabla 25. Evaluación Final con Resultados por Prioridad

	T. Autónomas	Compatibilidad	Precio	Peso	Disponibilidad	$\sum$	Prioridad
Alternativa 1	0,13	0,01	0,02	0,02	0,03	0,19	4

	T. Autónomas	Compatibilidad	Precio	Peso	Disponibilidad	$\sum$	Prioridad
Alternativa 2	0,06	0,02	0,04	0,04	0,08	0,23	3
Alternativa 3	0,03	0,03	0,06	0,06	0,10	0,27	2
Alternativa 4	0,09	0,01	0,08	0,08	0,05	0,31	1
					Suma	1	

Según la Tabla25 se selecciona la alternativa 4 la cual corresponde al controlador Mateksys F722, este módulo cuenta con un microcontrolador stm32f722ret6 tipo ARM con una señal de reloj interna de 216MHz, muy superior a la alternativa 3 que cuenta con 16 MHz, esto le permite manejar el ciclo PID de manera mucho más efectiva, además el módulo cuenta por defecto con sensores necesarios para el manejo del UAV específicamente un barómetro DPS310 y un giroscopio/acelerómetro MPU6000, ambos ideales para esta aplicación. Su voltaje de operación es de 6V a 36V, con una corriente máxima de 186A, esto es importante debido a que cada motor consume un máximo de 12 A por lo que puede ser utilizado tanto para una configuración tipo octocopter o quadcopter.

6.1.4. Firmware

Se realiza el análisis del firmware mediante el método de promedios ponderados para los siguientes criterios:

- **Tareas Autónomas:** Capacidad para almacenar,leer y ejecutar instrucciones preprogramadas que permitan el vuelo sin la necesidad de un operario.
- **Compatibilidad:** Se refiere a la cantidad de protocolos de comunicación con distintos sensores así como la capacidad de incorporar estos datos en el control PID.
- **Flexibilidad:** Facilidad de modificación del código fuente y de las distintas características del firmware.

Tabla 26. Alternativas para el controlador

Alternativa 1	Alternativa 2	Alternativa 3
Betaflight	Cleanflight	INAV

Evaluación del peso específico de cada criterio

T.Autónomas>Compatibilidad>Flexibilidad

Tabla 27. Evaluación del peso específico de cada criterio en firmware

	T. Autónomas	Compatibilidad	Flexibilidad	$\sum+1$	Ponderación
T. Autónomas	-	1	1	3	0,5
Compatibilidad	0	-	1	2	0,33
Flexibilidad	0	0	-	1	0,17
			Suma	6	1

Evaluación del peso específico de criterio CAPACIDAD PARA TAREAS AUTÓNOMAS

Alternativa 3>Alternativa 1>Alternativa 2

Tabla 28. Evaluación de criterio CAPACIDAD PARA TAREAS AUTÓNOMAS

	Alternativa 1	Alternativa 2	Alternativa 3	$\sum+1$	Ponderación
Alternativa 1	-	1	0	2	0,33
Alternativa 2	0	-	0	1	0,17
Alternativa 3	1	1	-	3	0,50
			Suma	6	1

Evaluación del peso específico de criterio COMPATIBILIDAD

Alternativa 1>Alternativa 3>Alternativa 2

Tabla 29. Evaluación de criterio COMPATIBILIDAD

	Alternativa 1	Alternativa 2	Alternativa 3	$\sum+1$	Ponderación
Alternativa 1	-	1	1	3	0,5

	Alternativa 1	Alternativa 2	Alternativa 3	$\sum+1$	Ponderación
Alternativa 2	0	-	0	1	0,17
Alternativa 3	0	1	-	2	0,33
			Suma	6	1

Evaluación del peso específico de criterio FLEXIBILIDAD

Alternativa 3>Alternativa 1>Alternativa 2

Tabla 30. Evaluación de criterio FLEXIBILIDAD

	Alternativa 1	Alternativa 2	Alternativa 3	$\sum+1$	Ponderación
Alternativa 1	-	1	0	2	0,33
Alternativa 2	0	-	0	1	0,17
Alternativa 3	1	1	-	3	0,5
			Suma	6	1

Evaluación Final

Tabla 31. Evaluación Final con Resultados por Prioridad

	T. Autónomas	Compatibilidad	Flexibilidad	$\sum$	Prioridad
Alternativa 1	0,17	0,17	0,06	0,39	2
Alternativa 2	0,08	0,06	0,03	0,17	3
Alternativa 3	0,25	0,11	0,08	0,44	1
			Suma	1	

Analizando los resultados obtenidos en la Tabla31 se selecciona la alternativa 3 perteneciente a el firmware INAV, este es un proyecto de código abierto escrito en C++ y compatible con microcontroladores ARM, cabe recalcar que este proyecto tiene como objetivo el desarrollo de una solución que permita realizar tareas manera autónoma simplificando el proceso de desarrollo de UAVs, esto lo logra mediante un código base que ofrece gran flexibilidad

para su modificación dependiendo del tipo de vehículo que se este desarrollando, al utilizar este firmware también se gana acceso a herramientas como ese el análisis de los registros de los sensores para extraer la respuesta de escalón unitario de los PIDs. La necesidad del uso del firmware se debe a que existen protocolos desarrollados por empresas privadas de los cuales no existe documentación pública pero si han sido implementados en INAV, un ejemplo de esto es lo que se conoce como Smartaudio el cual permite la configuración del módulo de transmisión de vídeo analógico mediante el microcontrolador. Otro de los puntos importantes de este firmware es la implementación de seguridad ya que permite la configuración de un sistema RTH (return to home) en caso de pérdida de señal.

6.2. Especificaciones del dron para el proyecto

Se establecen las necesidades y requerimientos del dron de carga con el objetivo de realizar el diseño de tal manera que se adapte a cada una de estas, las características planteadas son las siguientes:

- El prototipo debe tener una capacidad de carga de hasta 500 g mientras sea capaz de maniobrar en el aire de manera autónoma.
- Control por radiofrecuencia.
- Se debe utilizar una batería recargable, reemplazable y con voltaje dentro del rango de operación de los componentes electrónicos, en especial tomando en cuenta los motores, propelas y controladores de estos. es importante tomar en cuenta la capacidad de descarga ya que los motores requieren valores de corriente elevadas (2 A - 20 A cada uno, depende del motor y del voltaje utilizado).
- El operador debe ser capaz de visualizar y controlar en todo momento el dron, ya sea por línea de vista o de manera remota.
- Las tareas de entrega deben ser preprogramadas de manera sencilla por lo que se debe tener una base de datos de los posibles puntos de entrega/carga del dron.
- La localidad donde funcionará será Quito.

- Debe ser alimentado con una batería tipo LiPo 3S con un rango de voltaje entre 12,50 - 12,60 VDC.
- Las rutas deben ser calculadas de manera automática a partir de los puntos de salida y llegada.

7. Bosquejo del prototipo

Tomando en cuenta los requerimientos obtenidos en el estudio de alternativas, mediante un proceso de dimensionamiento geométrico iterativo y herramientas de prototipado rápido se realiza un bosquejo que facilite el proceso constructivo. El prototipo cuenta con una configuración tipo quadcopter, se requiere un compartimiento contenedor para la carga así como una estructura para poder montar los componentes electrónicos y mecánicos, se busca minimizar la masa del prototipo sin comprometer la integridad estructural del mismo por lo que la selección de materiales es crítica para cualquiera de las piezas a implementarse. En el caso de ciertos componentes el posicionamiento adecuado permiten el funcionamiento correcto, un claro ejemplo de esto es el módulo GPS el cual debe estar alejado lo más posible a cualquier cable o módulo que genere interferencia electromagnética, así como también no deben existir obstrucciones con los satélites que permiten la geolocalización del mismo. Para poder ensamblar todos los componentes de manera correcta se realiza un modelo CAD en el software Autodesk Inventor 2023, este cuenta con dimensiones reales con lo que se busca representar con la mayor exactitud posible las piezas del prototipo, el modelo se muestra en la Figura 11.

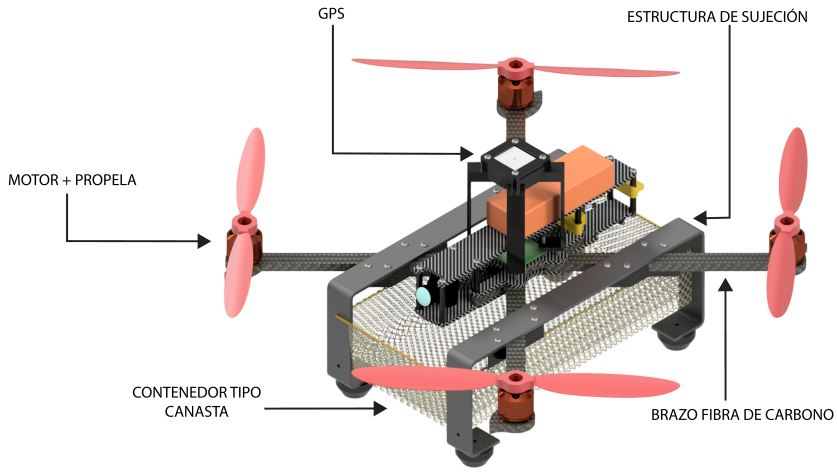


Figura 11. Modelo 3D desarrollado en el software CAD Autodesk Inventor 2023

Las dimensiones del prototipo se basan en su mayoría en las dimensiones geométricas de los componentes electrónicos, excepto las piezas que soportan carga las cuales deben ser diseñadas mediante cálculos y simulaciones para asegurar su correcto funcionamiento. Otra de las consideraciones a tomar en cuenta es el hecho de que no se tiene como prioridad un diseño aerodinámico ya que el precio del prototipo se volvería mucho mayor, además podría resultar en un aumento de peso del UAV.

8. Diseño mecánico

8.1. Diseño de la estructura de sujeción

Se realiza el diseño de la estructura de la sujeción presentada en la Figura 12 la cual será la encargada de soportar el peso de la carga, para poder realizar su análisis se realiza un diagrama de cuerpo libre presentado en la Figura 13, se toma como parámetros iniciales una fuerza P generada por la carga dividida para 4 ya que existe ese número de puntos de sujeción, se tiene como objetivo un factor de seguridad mínimo de 1.10 recomendado por la norma europea para estructuras de aluminio [22].

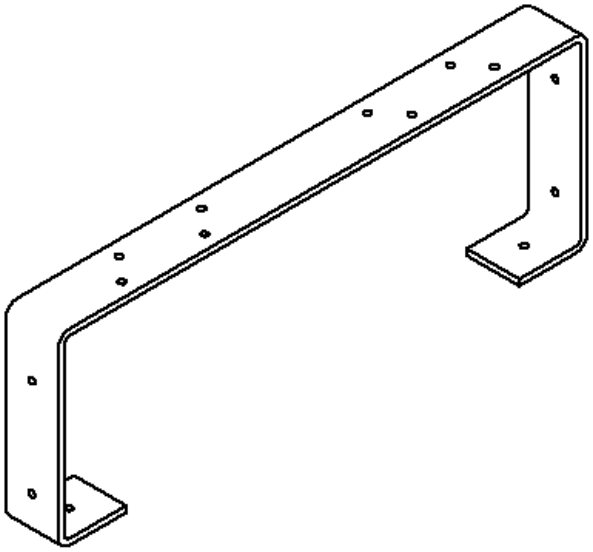


Figura 12. Estructura de aluminio

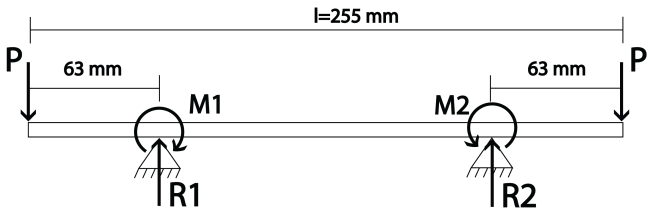


Figura 13. Diagrama de cuerpo libre simplificado de la platina de aluminio

8.1.1. Selección del material

Debido a las restricciones de peso del prototipo y la disponibilidad en el mercado ecuatoriano se selecciona la platina de aluminio marca CEDAL 1267 fabricada de aluminio 6061, su espesor es de 2,40 mm, sus propiedades se presentan en la Tabla 32 pueden ser encontradas en [9], estos datos son utilizados en cálculos futuros.

Tabla 32. Propiedades mecánicas Aluminio 6061, Fuente: [9]

Descripción	Símbolo	Valor
Módulo de elasticidad	E	71,70 GPa
Módulo de rigidez	G	26,90 GPa
Módulo de Poisson	ν	0,33

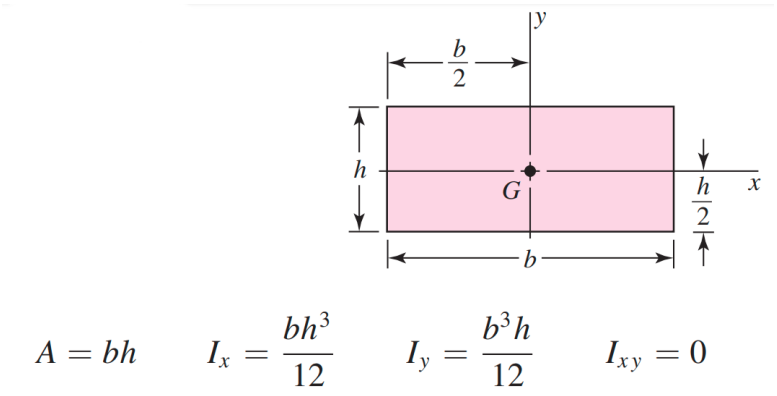
8.1.2. Cálculo de esfuerzos y factor de seguridad

Una vez seleccionado el material y la platina, se realiza diseño estático por lo que se determinan las propiedades geométricas de la sección, la platina CEDAL de modelo 1267 cuenta con las siguientes dimensiones según [23], se utiliza la nomenclatura mostrada en la Figura 14 presentada en [9] :

$$\rho_{lineal} = 0{,}142\ kg/m \rightarrow \textit{Densidad lineal}$$

$$h = 2{,}4\ mm \rightarrow \textit{Espesor de la platina}$$

$$b = 25{,}4\ mm \rightarrow \textit{Ancho de la platina}$$



$$A = bh \qquad I_x = \frac{bh^3}{12} \qquad I_y = \frac{b^3h}{12} \qquad I_{xy} = 0$$

Figura 14. Diagrama para el cálculo de inercia para una barra de sección transversal rectangular, Fuente: [9]

Se calcula el masa total aproximada de la estructura de aluminio utilizando la ecuación 2.

$$masa_{total} = l_{total} \cdot \rho_{lineal} \tag{2}$$

Donde:

l_{total} : *Longitud total de la platina, en* m

ρ_{lineal} : *Densidad lineal, en* kg/m

$$masa_{total} = 1{,}510\ m \cdot 0{,}142\ kg/m = 0{,}215\ kg$$

También se calcula el área A y la posición el centroide utilizando las ecuaciones 3 y 4 respectivamente.

$$A = h \cdot b \tag{3}$$

Donde:

h : *Espesor de la platina, en* mm

b : *Ancho de la platina, en* mm

$$A = 2{,}4\ mm \cdot 25{,}4\ mm = 60{,}96\ mm^2$$

$$centroide = [b/2, h/2] \tag{4}$$

Donde:

h : *Espesor de la platina, en* mm

b : *Ancho de la platina, en* mm

$$centroide = [25{,}40/2; 2{,}40/2]\ mm = [12{,}70; 1{,}20]\ mm$$

Se establece el punto crítico c para el cálculo del factor de seguridad y de los esfuerzos como se muestra en la Figura 15, este punto es seleccionado tomando en cuenta que el esfuerzo normal será mayor en la superficie de la pieza.

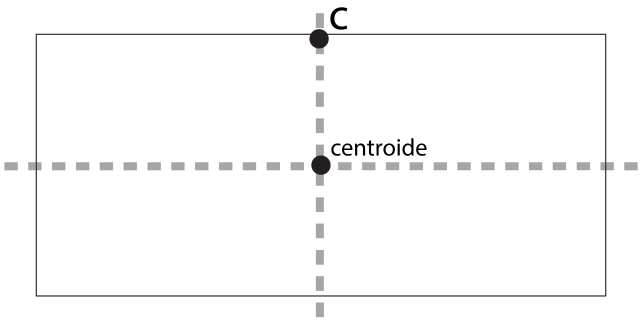


Figura 15. Punto de crítico del componente

Se calcula el segundo momento de inercia de área en dirección x, I_x con la ecuación

(5), y en dirección y, I_y utilizando la ecuación (6).

$$I_x = b \cdot h^3/12 \tag{5}$$

Donde:

h : *Espesor de la platina, en* mm

b : *Ancho de la platina, en* mm

$$I_x = 25{,}4\ mm \cdot (2{,}4\ mm)^3/12 = 29{,}261\ mm^4$$

$$I_y = b^3 \cdot h/12 \tag{6}$$

Donde:

h : *Espesor de la platina, en* mm

b : *Ancho de la platina, en* mm

$$I_y = (25{,}4\ mm)^3 \cdot 2{,}4mm/12 = 3277{,}41\ mm^4$$

El momento de inercia mixto en la dirección xy es nulo. Se calcula el momento polar de inercia J con la ecuación (7)

$$J = I_x + I_y = 3277{,}41\ mm^4 + 29{,}26\ mm^4 = 3306{,}67\ mm^4 \tag{7}$$

Estado de esfuerzos

Se hace uso del diagrama de cuerpo libre mostrado en la Figura 13 para el cálculo del esfuerzo cortante y normal. Primero se identifican las cargas que se están aplicando, en este caso el peso de generado por la carga a transportarse, este se calcula mediante la ecuación (8)

$$P = m/4 \cdot g \tag{8}$$

Donde:

m : *Masa de la carga a transportar, en* kg

g : *Aceleración gravitacional de la tierra, en* m/s^2

$$P = 0{,}5\ kg/4 \cdot 9{,}81\ m/s^2 = -1{,}226\ N$$

Cálculo de reacciones

Se utiliza la ecuación (9) para el cálculo de las reacciones R_1 y R_2 generadas en los puntos de apoyo, al tratarse de cargas simétricas ambas reacciones tendrán la misma magnitud.

$$\sum F_y = R_1 + R_2 - 2P = 0 \rightarrow R_1 = R_2 \rightarrow 2R - 2P = 0 \rightarrow R = P \tag{9}$$

Donde:

R_1 : *Reacción 1, en* N

R_2 : *Reacción 2, en* N

P : *Carga aplicada, en* N

$$R_1 = R_2 = 1{,}23\ N$$

Cálculo de momentos

Se utiliza la ecuación (10) para el cálculo de los momentos M_1 y M_2 generados por la fuerza P .

$$\sum M\ = M_1 + M_2 = 0 \rightarrow M_1 = -M_2 = P \cdot d_x \tag{10}$$

Donde:

P : *Fuerza aplicada, en* N

d_x : *Distancia entre la carga y el punto de apoyo, en* m

$$M_1 = -1{,}226\ N \cdot 0{,}063\ m = -0{,}08\ N \cdot m$$

$$M_2 = -M_1 = 0{,}08\ N \cdot m$$

La distribución del cortante y el momento flector a lo largo de la platina se pueden ver en la Figura 16.

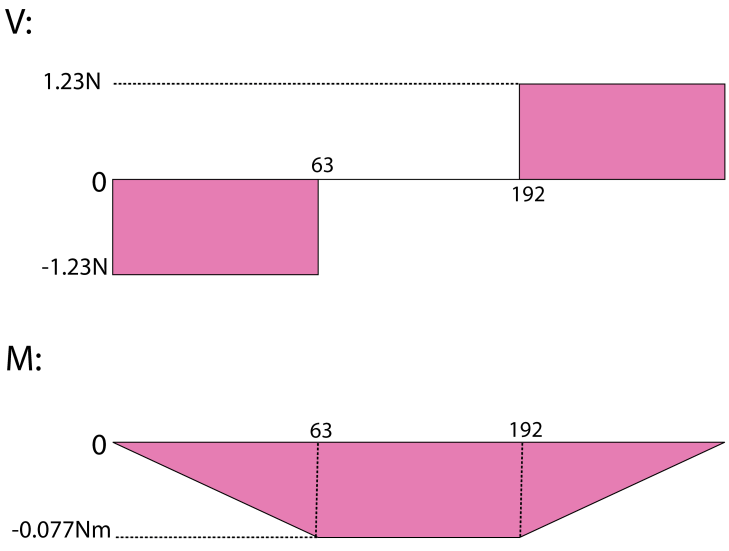


Figura 16. Diagrama de momento flector y cortante sobre la platina de aluminio

Primer momento de área

Se calcula el primer momento de área Q con la ecuación (11)

$$Q = \bar{A} \cdot \bar{y} \tag{11}$$

Donde:

$\bar{A}$: *Área de la sección transversal después del corte por el punto crítico, en* mm^2

$\bar{y}$: *Distancia entre los centroides de A y* $\bar{A}$*, en* mm

$$Q = 60,96\ mm^2 \cdot 0\ mm = 0\ mm^3$$

Esfuerzo cortante

Se calcula el esfuerzo cortante τ en el punto c utilizando la ecuación (12)

$$\tau = (P \cdot Q)/(I_x \cdot b) \tag{12}$$

Donde:

P : *Carga aplicada, en* N

Q : *Primer momento de área, en* mm^3

I_x : *Segundo momento de inercia de área, en* mm^4

b : *Ancho de la platina, en* mm

Debido a que en el punto c el primer momento de área es 0 entonces el valor de τ también es 0.

Esfuerzo normal

Se calcula el esfuerzo normal σ_x generado por el momento flector utilizando la ecuación (13).

$$\sigma_x = (M \cdot d_c)/I_x \tag{13}$$

Donde:

M : *Momento flector, en* $N \cdot m$

d_c : *Distancia perpendicular con el centroide, en* mm

I_x : *Segundo momento de inercia de área, en* mm^4

$$\sigma_x = (-0{,}077\ N \cdot m \cdot 1{,}2\ mm)/(29{,}26\ mm^4) = 3{,}157\ MPa \approx 3{,}16\ MPa$$

Debido a que no existe un momento o carga axial que influya en el esfuerzo en dirección y se establece que el esfuerzo normal σ_y en nulo.

Esfuerzos principales

Para poder calcular el factor de seguridad se requiere conocer los esfuerzos principales σ_1, σ_2 los cuales se calculan mediante la ecuación (14) o el cortante máximo τ_{max} mediante la ecuación (15) dependiendo de la teoría de falla a utilizarse

$$\sigma_1, \sigma_2 = (\sigma_x + \sigma_y)/2 \pm 1/2 \cdot \sqrt{(\sigma_x - \sigma_y)^2 + 4 \cdot \tau^2} \tag{14}$$

Donde:

σ_x : *Esfuerzo normal en dirección x, en* MPa

σ_y : *Esfuerzo normal en dirección y, en* MPa

τ : *Esfuerzo cortante, en* MPa

$$\tau_{max} = 1/2 \cdot \sqrt{(\sigma_x - \sigma_y)^2 + 4 \cdot \tau^2} \tag{15}$$

Donde:

σ_x : *Esfuerzo normal en dirección x, en* MPa

σ_y : *Esfuerzo normal en dirección y, en* MPa

τ : *Esfuerzo cortante, en* MPa

En este caso se está utilizando la teoría de falla por energía de distorsión por lo que se requiere calcular los esfuerzos principales.

$$\sigma_1, \sigma_2 = (3{,}16\ MPa + 0\ MPa)/2 \pm 1/2 \cdot \sqrt{(3{,}16\ MPa - 0\ MPa)^2 + 4 \cdot (0\ MPa)^2}$$

$$\sigma_1 = 3{,}16\ MPa$$

$$\sigma_2 = 0\ MPa$$

Factor de seguridad

Se utilizan los datos presentados en [24] para calcular el esfuerzo de Von Mises σ' aplicado en el punto de estudio mediante la ecuación (16), por lo cual es necesario conocer la resistencia a la fluencia S_y, con este dato se calcula el factor de seguridad η_{ED} mediante la ecuación (17).

$$S_y = 240\ MPa \rightarrow \textit{Resistencia a la fluencia del aluminio}$$

$$\sigma' = \sqrt{\sigma_1^2 + \sigma_2^2 - \sigma_1 \cdot \sigma_2} \tag{16}$$

Donde:

$$\sigma_1 : \textit{Esfuerzo principal 1, en } MPa$$

$$\sigma_2 : \textit{Esfuerzo principal 2, en } MPa$$

$$\sigma' = \sqrt{(3{,}16\ MPa)^2 + (0\ MPa)^2 - 3{,}16\ MPa \cdot 0\ MPa} = 3{,}16\ MPa$$

$$\eta_{ED} = S_y\ /\sigma' \tag{17}$$

Donde:

$$S_y : \textit{Resistencia a la fluencia, en } MPa$$

$$\sigma' : \textit{Esfuerzo de Von Mises, en } MPa$$

$$\eta_{ED} = 240\ MPa/3{,}16\ MPa = 72{,}95$$

Este factor de seguridad es superior al indicado por la norma, pero debido a que se trata de la platina con el menor espesor disponible en el mercado local, se selecciona este elemento

8.2. Diseño de los brazos de fibra de carbono

En este caso se esta trabajando con CFRP tipo T300-3k con un trenzado de 1x1, esto quiere decir que se tiene una resistencia ultima a la tensión de $300\ ksi$ o $2068\ MPa$ y una

calidad 3k utilizada de manera estándar para aplicaciones comerciales, las características del material se describen en la Tabla 33 (pueden existir variaciones entre fabricantes sin embargo estas variaciones suelen ser despreciables).

Tabla 33. Propiedades mecánicas CFRP T300-3k, Fuente: [16]

Nombre	Símbolo	Valor
Resistencia última a la tensión	S_u	2068 MPa
Módulo de elasticidad	E	135 GPa
Densidad	ρ	1,76 g/cm^3

Debido a la complejidad del comportamiento del material, se hace uso del programa de análisis de elementos finitos Autodesk Inventor Nastran 2023 con licencia estudiantil, esto nos permite realizar simulaciones complejas y definir materiales laminados compuestos, una de las ventajas de este programa es su versatilidad ya que nos brinda la opción de modificar el criterio de falla, en este caso se lo configura para utilizar el criterio de Tsai-Hill como se recomienda en [25], esta dado por la ecuación (18), el criterio de falla de Tsai-Hill predice de mejor manera el comportamiento de materiales laminados que otras teorías de falla como puede ser la de Von Mises, se puede ver una comparación entre estas teorías en la Figura 17.

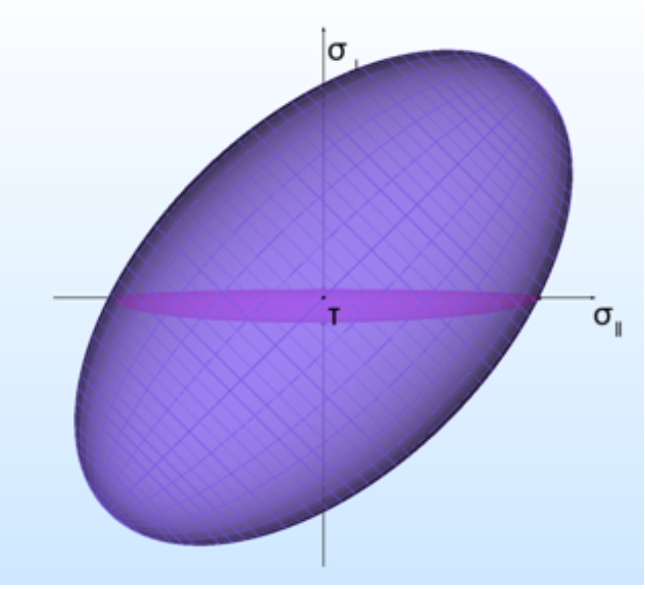


Figura 17. diagrama de esfuerzos máximos: von Mises (purpura) vs Tsai-Hill (magenta)

$$(\sigma_1^2)/(S_1^2) + (\sigma_2^2)/(S_2)^2 + (\tau_T^2)/(S_T^2) < 1 \tag{18}$$

Donde:

σ_1 : *esfuerzo principal en dirección longitudinal*

σ_2 : *esfuerzo principal en dirección transversal*

τ_T : *esfuerzo cortante*

S_1 : *Resistencia a la fractura a tracción máxima longitudinal*

S_2 : *Resistencia a la fractura a tracción máxima transversal*

S_T : *Resistencia a la fractura a tracción máxima en dirección diagonal*

Previamente a las simulaciones se realiza un diagrama de cuerpo libre simplificado basado en la pieza mostrada en la Figura 18 el cual se muestra en la Figura 19, esto con el objetivo de entender de mejor manera cual es el escenario que se debe simular en Autodesk Nastran.

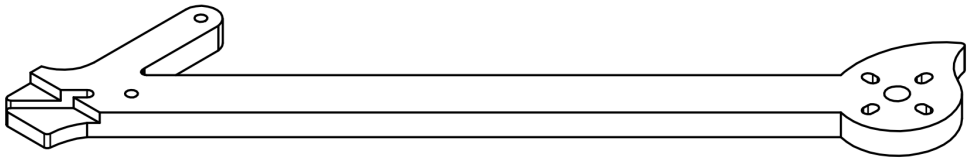


Figura 18. Brazo del UAV

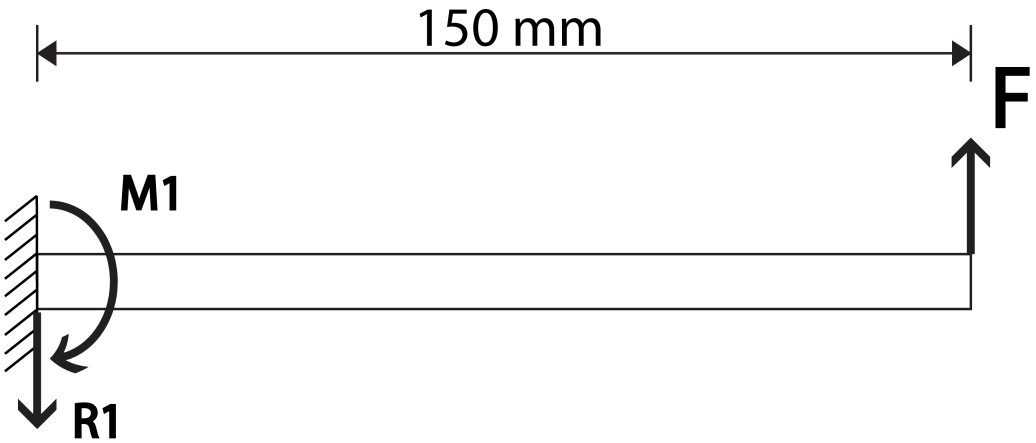


Figura 19. Diagrama de cuerpo libre simplificado del brazo de CFRP

Para obtener resultados útiles en el análisis por elementos finitos es importante configurar los parámetros del material de manera correcta, en este caso se trata de un laminado por lo que es importante conocer su topografía, el material en cuestión tiene un espesor de 7mm con un total de 15 láminas, al ser un laminado cuasi-isotrópico el delta del ángulo de dirección de cada capa se calcula como $45°$ asumiendo que se trata de una estructura simétrica, esto se configura en el programa como se muestra en la Figura 20. La fuerza F es definida por la ecuación (19) .

$$F = (M_{carga} + M_{UAV})/4 \cdot g \tag{19}$$

Donde:

M_{carga} : *masa de la carga a transportar, en kg (0,50 kg)*

M_{UAV} : *masa del prototipo, en kg (1,13 kg)*

g : *Aceleración gravitacional en la tierra, en* m/s^2

$$F = (0,50\ kg + 1,13\ kg)/4 \cdot 9{,}81\ m/s^2 = 4{,}05\ N$$

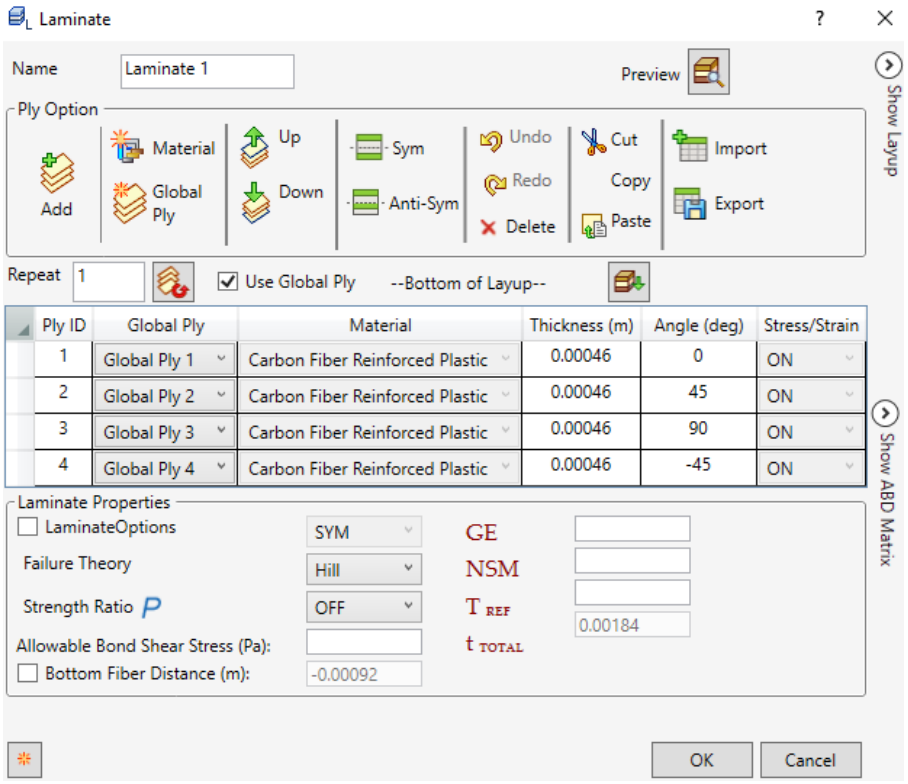


Figura 20. Configuración del laminado cuasi-isotrópico en Autodesk Inventor Nastran

8.2.1. Análisis de desplazamiento

Se realiza un análisis de elementos finitos para determinar el desplazamiento y deformación de la pieza, como se puede ver en la Figura21, el desplazamiento máximo obtenido es de $0{,}013\ mm$.

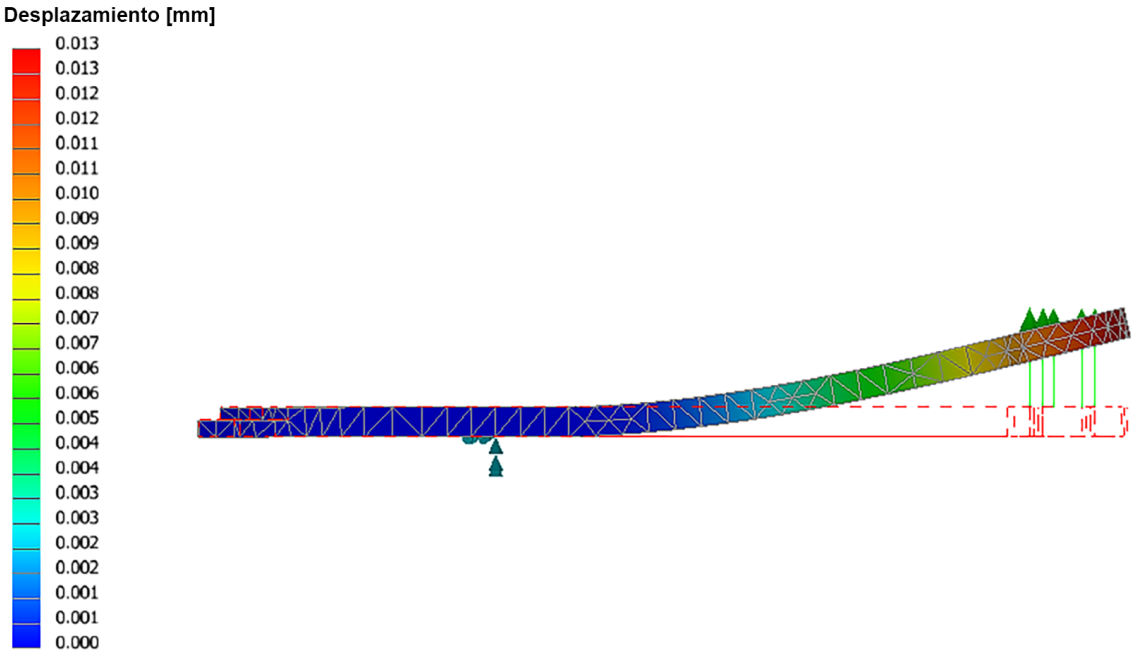


Figura 21. Simulación de deformación en Autodesk Inventor Nastran

8.2.2. Factor de seguridad

Se realiza un análisis mediante el método de elementos finitos para determinar el factor de seguridad, en el caso de los materiales compuestos se debe tomar en cuenta factores como la tendencia a la delaminación del material, debido a esto se utiliza el criterio índice máximo de falla de capa, como se muestra en [26] este factor es menor a 1 indica una falla interna, generalmente por delaminación, mientras que si es mayor a 1 indica una falla en la superficie de la pieza, en este caso el índice máximo de falla de capa es de $1{,}34 \cdot 10^{-5}$ por lo que, en caso de darse, la falla será interna, para poder determinar el factor de seguridad SR que en este caso se conoce como relación de esfuerzos, se realizan los cálculos como se muestra en la ecuación (20).

$$SR = 1/FI \tag{20}$$

Donde:

FI : *Índice máximo de falla de capa, adimensional*

$$SR = 1/1{,}34 \cdot 10^{-5} = 74{,}62$$

8.3. Selección de la combinación motor-propela

Se requiere levantar una masa aproximada de $1{,}6\ kg$ incluyendo el peso del UAV, por lo que se busca que los motores sean capaces de generar un empuje mínimo que sea por lo menos el doble, es decir de $3{,}2\ kg$ de tal manera que los motores sean capaces de maniobrar el dron sin problema, de manera general se suelen dimensionar los motores con una relación empuje-carga de 3, esto simplifica la implementación de los controladores brindando una mayor autoridad al dron en el aire y reduciendo el riesgo de sobrecalentamiento de los circuitos electrónicos, se implementa una relación de 2 para reducir el precio constructivo del prototipo, debido a esto se debe tener presente que el dron tendrá una respuesta más lenta de la esperada por lo que se debe maniobrar de manera más conservadora (sin cambios bruscos de velocidad o dirección).

8.3.1. Cálculos de dimensionamiento de los motores

En el caso de motores sin escobillas el movimiento se genera a partir de la fuerza F_{im} generada por los imanes y las bobinas de este, esta fuerza se puede calcular con la ecuación (21).

$$F_{im} = (B^2 \cdot A)/(2 \cdot \mu_0) \tag{21}$$

Donde:

B : *densidad del flujo magnético, en* T

A : *área del motor, en* m

μ_0 : *permeabilidad magnética entre el estator y el rotor, en* H/m

Si el área se define como el área lateral del motor debido a que el efecto en la parte superior e inferior es despreciable y se establecen las dimensiones del motor como se muestra en la Figura 22 se obtiene la ecuación (22).

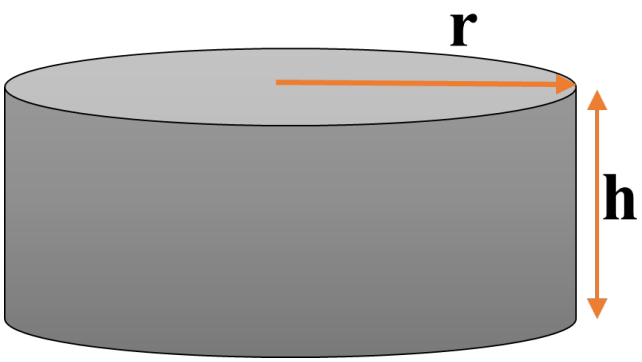


Figura 22. Descripción de las dimensiones del motor

$$F_{im} = (B^2 \cdot (2 \cdot \pi \cdot r \cdot h))/(2 \cdot \mu_0\ ) = (B^2 \cdot (\pi \cdot r \cdot h))/\mu_0 \tag{22}$$

Donde:

B : *densidad del flujo magnético, en* T

A : *área del motor, en* m

μ_0 : *permeabilidad magnética entre el estator y el rotor, en* H/m

h : *Altura del motor, en* m

r : *Radio del motor, en* m

Un vez obtenida la fuerza es necesario obtener el torque del motor T ya que este será el encargado de mover el motor y la propela, este se define como la fuerza multiplicada por la distancia perpendicular, que en este caso corresponde al radio del motor por lo que la ecuación del torque es dada por la ecuación (23).

$$T = F \cdot r = (B^2 \cdot \pi \cdot r \cdot h)/\mu_0\ \cdot r = (B^2 \cdot \pi \cdot r^2 \cdot h)/\mu_0 \tag{23}$$

Donde:

B : *densidad del flujo magnético, en* T

μ_0 : *permeabilidad magnética entre el estator y el rotor, en* H/m

h : *Altura del motor, en* m

r : *Radio del motor, en* m

Debido a que el estándar de fabricación de los motores sin escobillas es utilizar imanes tipo N52 y además su manufactura permite que el cambio en la permeabilidad entre diferentes modelos de motor sea despreciable, la variable que se dimensiona son las dimensiones del motor, en específico el volumen del estator V el cual será directamente proporcional al torque que este puede generar, el volumen entonces está definido como 24

$$V = \pi \cdot r^2 \cdot h \tag{24}$$

Donde:

h : *Altura del motor, en* m

r : *Radio del motor, en* m

En un caso ideal cuando se trabajan con hélices grandes (mayores a 9 pulgadas) se recomienda el uso de motores de la serie 26XX, ya que un mayor diámetro ayudaría a que el motor no se sobre caliente y brindaría más estabilidad al UAV. Sin embargo, estos motores tienden a ser costosos y fuera del presupuesto de este proyecto, así como también cuentan con una disponibilidad baja en el país. Una de las opciones son los motores U3 700KV de TMotors los cuales cuentan con una documentación extensa, estos motores son capaces de mover propelas de hasta 15 pulgadas y generar un empuje de 4kg, sin embargo no se pueden conseguir en el país por lo que se selecciona un motor QWinOut 2212 1000KV debido a su disponibilidad en el mercado, ya que al tratarse de una pieza crítica para el funcionamiento la cual también es conocida por ser el punto de falla mas común de los multirotores es necesario poder adquirir repuestos con facilidad, sus dimensiones se describen de manera estándar por el número del modelo, en este caso se tiene el modelo 2212 siendo 22 el diámetro del estator y 12 la altura del estator en mm por lo que se obtiene 25, estas dimensiones no son ideales ya que el motor puede llegar a sobre-calentarse, esto se deberá tomar en cuenta al momento de operar el dron.

$$V = \ \pi \cdot r^2 \cdot h = \pi \cdot (11\ mm)^2 \cdot 12\ mm = 4561{,}6\ mm^3 \tag{25}$$

Es imposible conocer el torque generado por el motor de manera precisa sin conocer sus características magnéticas o a través de pruebas mecánicas, por lo que se toma la recomendación del fabricante la cual determina que el motor es capaz de mover propelas de hasta 11 pulgadas, si se llegan a realizar pruebas de torque del motor es importante tomar en cuenta la velocidad del motor, ya que debido a las pérdidas generadas por el efecto joule, un motor más lento será más eficiente y capaz de generar más torque.

8.3.2. Cálculos de dimensionamiento de las propelas

Cálculo de la inercia

Se debe determinar la inercia de las propelas para poder saber si el motor será capaz de mover la propela, para poder determinarlo se hace uso de la ecuación 26.

$$I_{total} = n \cdot (m_{pala} \cdot r^2/3) \tag{26}$$

Donde:

n : *Número de palas en la hélice, adimensional*

m_{pala} : *masa de cada pala, en* kg

r : *radio de la hélice, en* m

Se realizan los cálculos para las propelas 1045 y 1245 y se presentan en la Tabla 34.

Tabla 34. Cálculo de inercia para las propelas

Propela	1245	1045
Inercia	$7{,}75 \cdot 10^{-5}\ kg \cdot m^2$	$4{,}84 \cdot 10^{-5}\ kg \cdot m^2$
Área	$0{,}004572\ m^2$	$0{,}00381\ m^2$
Torque mínimo	$0{,}166\ N$	$0{,}125\ N$

Velocidad máxima y ángulo de ataque

Los efectos del flujo del aire sobre la propela son los encargados de generar el empuje necesario para que la UAV pueda despegar y mantenerse en el aire, esto ha dado como re-

sultado el uso de perfiles optimizados para el vuelo, ampliamente estudiados en la dinámica de fluidos especialmente en aplicaciones aeroespaciales, estos perfiles son conocidos en inglés como airfoil y se encuentran como base en el diseño de las propelas, una de las principales características es el coeficiente de empuje C_l, el cual está directamente relacionado con las fuerzas generadas por la interacción con el viento y se define por (27), podemos ver que este coeficiente es dependiente del ángulo de ataque AOA, definido por (28) siendo $\omega \cdot r$ la velocidad tangencial de la punta de la propela también definida como ν, según ([10]) un mayor ángulo de ataque permite obtener un mayor empuje mientras que se cumpla que $AOA < 15°$ debido a que en este punto conocido como ángulo de ataque critico comienza a generarse flujo turbulento debido a la separación del flujo, esto impide la generación de fuerza de empuje y la UAV entra en un estado conocido como "stall" definido en la curva que se muestra en la Figura 23 ,debido a que el flujo del viento afecta el ángulo ataque, incluso si la propela esta bien dimensionada, condiciones adversas pueden generar este efecto.

$$C_l = 2\pi \cdot AOA \tag{27}$$

Donde:

AOA : *Ángulo de ataque, definido como (28)*

$$AOA = \theta_{prop} - atan(V_{in}/\omega \cdot r) \tag{28}$$

Donde:

θ_{prop} : *Ángulo de ataque fijo de la propela definido por su paso, definida como (29)*

V_{in} : *Velocidad del viento perpendicular al "disco" formado por la propela definida por (30), en* m/s

r : *Radio de la propela, en* m

ω : *Velocidad angular de la propela, en* deg/s

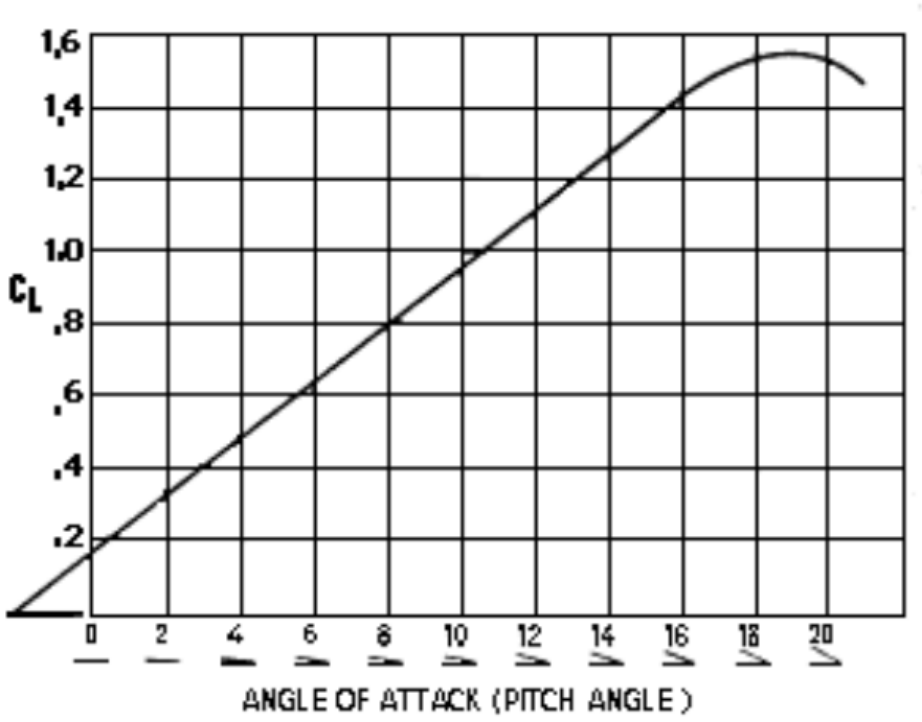


Figura 23. Curva de coeficiente de empuje vs AOA, Fuente: [10]

$$\theta_{prop} = atan(p/\pi \cdot d) \tag{29}$$

Donde:

p : *Paso de la propela, en* m

d : *Diámetro de la propela, en* m

$$V_{in} = 1/2 \cdot \sqrt{m_{AUW} \cdot g/\rho \cdot \pi \cdot r} \tag{30}$$

Donde:

r : *radio de la propela, en* m

m_{AUW} : *masa total del UAV, en* kg

ρ : *densidad del aire, en los cálculos se utiliza la densidad a nivel de la ciudad de Quito*

g : *Aceleración gravitacional en la tierra, en* m/s^2

Se realizan los cálculos del ángulo de ataque para distintos escenarios presentados en la Tabla 35

Tabla 35. Cálculo del ángulo de ataque

	700KV 1245 carga 500g	1000KV 1245 carga 500g	1000KV 1045 carga 500g	2200KV 1045 carga 500g
$V_{in}\ [m/s]$	3,17	3,17	3,47	3,47
$\nu\ [m/s]$	80,44	114,9	95,76	210,66
θ_{prop}	6,81°	6,81°	8,15°	8,15°
AOA	4,54°	5,225°	6,07°	7,2°
$\rho_{Quito} = 0{,}84\ kg/m^3$				

Si bien todos los casos analizados cumplen con el criterio $AOA < 15°$ también existen otras propiedades que deben estar dentro de los rangos aceptables, una de las más importantes es la velocidad máxima de la punta de la propela, para este cálculo se toma la velocidad del sonido como referencia. La eficiencia de la propela alcanza su punto máximo cuando se encuentra cerca de Mach 0.6 , y la velocidad máxima que puede alcanzar se encuentra entre Mach 0.8 y Mach 0.9. En la Tabla 36 se encuentran las velocidades de las puntas de las propelas para los casos analizados anteriormente calculadas mediante la ecuación (31.)

$$\nu_{punta} = r \cdot \omega_{max} / \nu_{sonido} \tag{31}$$

Donde:

ν_{sonido} : *Velocidad del sonido, en m/s*

ω_{max} : *Velocidad angular máxima, en* deg/s

r : *radio de la propela, en* m

Tabla 36. Cálculo velocidad de la punta de la propela

	700KV 1245	1000KV 1245	1000KV 1045	2200KV 1045
ν_{punta}	$Mach$ 0,39	$Mach$ 0,56	$Mach$ 0,47	$Mach$ 1,02
$\nu_{sonido} = 343\ m/s$				

En los casos analizados podemos ver que el último caso no cumple la condición de velocidad máxima de la punta de la propela por lo que no es factible su implementación, esto puede resultar en vuelos inestables, vibración e incluso en rotura de las palas de la propela, en este caso se decide utilizar la combinación 1000KV 1045 con una carga de 0.5 kg para el prototipo ya que cumplen con los requerimientos mínimos de empuje necesario y sus características son satisfactorias tanto en el ángulo de ataque como en la velocidad de la punta de la propela. Es importante tomar en cuenta que si se desean utilizar motores más potentes (capaces de mover propelas de mayor tamaño y por ende generar más empuje) es necesario utilizar baterías de mayor capacidad, lo cual aumentaría el peso de UAV y estos cálculos no serian validos.

Empuje

Una vez determinado el tamaño de la propela que el motor es capaz de mover es importante determinar el empuje que puede generar la combinación hélice motor, para esto es importante conocer los parámetros que gobiernan las propelas. Según [27] puede determinar el empuje generado L utilizando la ecuación (32).

$$L = 0{,}5 \cdot C_l \cdot A \cdot \rho \cdot \nu^2 \tag{32}$$

Donde:

C_l : *Coeficiente experimental de empuje adimensional*

A : *Área de la propela, en* m^2

ρ : *Densidad del aire en* kg/m^3

ν : *Velocidad de la propela, en* m/s

Esta ecuación requiere determinar el coeficiente de empuje de manera experimental por lo que al menos que se cuente con el equipo capaz de realizar pruebas de empuje es imposible determinar el empuje generado por la combinación propela-motor. debido a esto se usan datos del catalogo del fabricante para la elección de propela, se utilizara un modelo 1045 (diámetro de 10 pulgadas y paso de 45 pulgadas) debido a que esto generara aproximadamente 850 g de empuje, lo cual en una combinación tipo quadcopter da como resultado 3.4 kg de empuje máximo . Si se cuenta con un presupuesto mayor se recomienda el uso de un motor T-motor U3 700KV con una propela CF12*4 (1245) la cual brindaría 4 kg de empuje en una configuración tipo quadcopter, es importante tener en cuenta que este cambio elevaría el precio del proyecto de manera significativa debido a que estos motores tienen un costo cuatro veces mayor que los motores seleccionados.

9. Dimensionamiento electrónico y de control

9.1. Diagrama del circuito electrónico

Previo al dimensionamiento del circuito electrónico del prototipo se realizó un diagrama de bloques como se muestra en la Figura 24

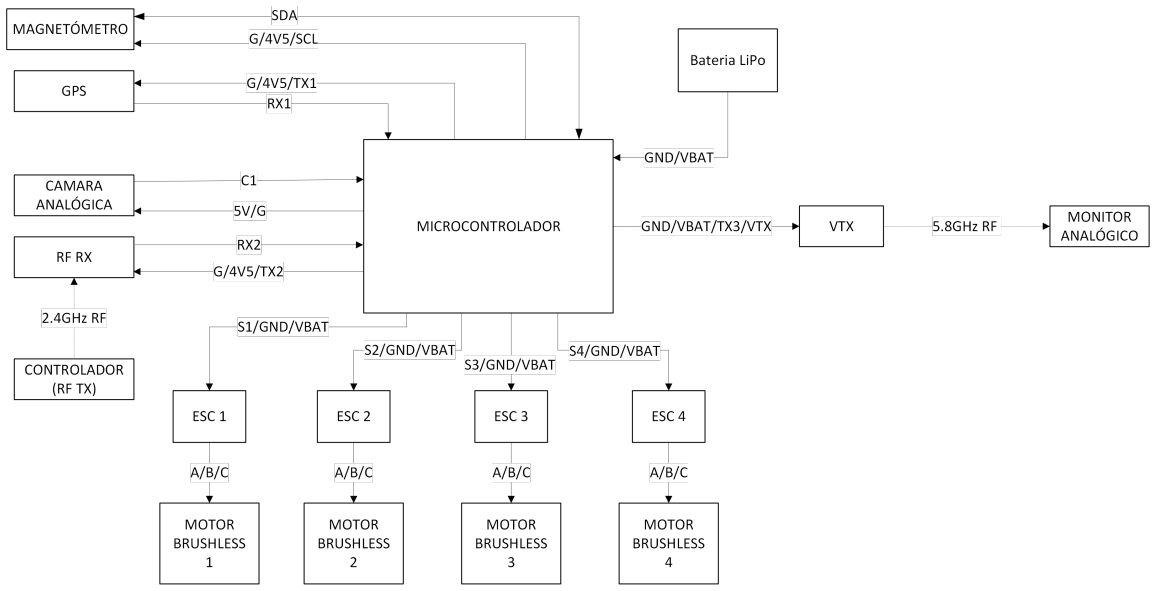


Figura 24. Diagrama del circuito electrónico del prototipo

Los distintos componentes del circuito pueden ser separados en distintas categorías según su función, de esta manera también se realiza un análisis de entradas y salidas del

prototipo como se muestra en la Tabla 37, el número de pines de salida del microcontrolador es menor al total de conexiones debido a que los motores sin escobillas van conectados directamente a los variadores los cuales requieren una única señal de control por cada motor. Uno de los requerimientos más importantes del microcontrolador es la cantidad de puertos seriales, debido a la cantidad de módulos utilizados se requiere 4 puertos seriales independientes (contando el puerto USB) así como también una conexión I2C y una entrada analógica.

Tabla 37. Entradas y salidas necesarias para el funcionamiento del prototipo

Nombre	Cant	Entradas (D)	Salidas (D)	Entradas (A)	Salidas (A)
Sensores					
Magnetómetro	1	1	1	0	0
GPS	1	1	1	0	0
Cámara Analógica	1	0	0	1	0
Comunicación					
RF RX	1	1	1	0	0
VTX	1	1	1	0	0
Drivers					
Variador BLDC	4	0	1	0	0
Actuadores					
Motor BLDC	4	0	3	0	0
Total Conexiones		4	20	1	0
Total Microcontrolador		4	8	1	0

El módulo microcontrolador seleccionado es el Mateksys F722-SE el cual cumple con los requerimientos de entradas y salidas del circuito, además tiene la ventaja de que el módulo cuenta con sensores tipo barómetro, giroscopio y acelerómetro, los cuales son esenciales para el control del dron.

9.1.1. Selección de los componentes electrónicos

Una vez obtenido el diagrama de bloques del circuito se procede a realizar la selección de componentes electrónicos a utilizar en la implementación del prototipo, los elementos seleccionados son los siguientes: módulo GPS + Magnetómetro M80 Pro: Este módulo GPS + magnetómetro es compatible con los sistemas GNSS tipo GPS, Galileo y GLONASS los cuales en conjunto con el magnetómetro QMC5883 permiten obtener datos de geolocalización hasta una velocidad de $500\ m/s$ ($\pm 0{,}05\ m/s$) con un error máximo de $1{,}5\ m$. El módulo es alimentado con $5\ VDC$, una de sus principales características es el hecho de que los datos de los sensores pueden ser obtenidos de manera independiente pero debido a esto es necesario contar con una conexión Serial, así como con una de tipo I2C, la primera para los datos de geolocalización y la segunda para los datos del magnetómetro.

- Acelerómetro MPU6000: este módulo está integrado en la placa del microcontrolador por lo que no son necesarias conexiones externas, cuenta con un sensor tipo giroscopio y acelerómetro, ambos con una resolución de 16 bits y permite obtener datos de los tres ejes de rotación de manera simultánea.

- Barómetro SPL06: este sensor viene integrado en la placa del microcontrolador, es utilizado para mediciones de altura basadas en la presión atmosférica relativa, esto quiere decir que se asume que el prototipo se encuentra a $0\ m$ de altura el momento de ser encendido, la precisión de este sensor es de $\pm 0{,}5\ m$ lo cual es suficiente para esta aplicación. Su voltaje de alimentación es de $3{,}3\ VDC$ y su interfaz de comunicación es I2C.

- Cámara Analógica Foxeer Razer Mini: Con una resolución de 1200 TVL (1280x1024) y un sensor tipo CMOS permiten obtener una imagen clara y una latencia baja sin aumentar el peso del prototipo de manera significativa, su voltaje de operación tiene un rango de $4{,}5\ VDC - 25\ VDC$ lo cual nos da la posibilidad de utilizar el voltaje directo de la batería.

- Receptor de radiofrecuencia FrSky R-XSR: Este módulo tiene una frecuencia de operación de $2{,}4\ GHz$ la cual le permite funcionar con controladores compatibles con el

protocolo ACCESS desarrollado por FrSky, es capaz de recibir valores analógicos en 16 canales programables. Su voltaje de operación es de $4\ VDC$ a $10\ VDC$ y la comunicación con el microcontrolador se da a través de comunicación serial. Una de sus características es la posibilidad de agregar redundancia en la recepción de la señal de control ya que cuenta con la posibilidad de comunicarse con un segundo módulo receptor con una arquitectura tipo maestro- esclavo en caso de ser necesario.

- Transmisor de video analógico (VTX) Xilo Stax: Este módulo trabaja a una frecuencia de $5{,}8\ GHz$ y una potencia de transmisión de $25\ mW$ a $600\ mW$, este valor puede ser variado en tiempo real y de manera remota lo que le permite evitar sobrecalentamiento, la comunicación con el microcontrolador se da a través de puerto serial, su voltaje de operación es entre $7\ VDC$ a $36\ VDC$.

- Variador BLDC (ESC): Se utilizan módulos digitales de $30\ A$, se selecciona esta capacidad debido a que la corriente máxima de los motores es de $12\ A$ de esta manera se disminuye el riesgo de daño, estos módulos se comunican con el microcontrolador utilizando el protocolo digital DSHOT por lo que no se requiere el proceso de calibración de variadores tradicionales.

9.1.2. Dimensionamiento de la batería

Uno de los componentes esenciales del prototipo es la fuente de alimentación la cual tiene relación directa con la velocidad de los motores y por ende de la propela, esto limita el voltaje máximo a $12\ VDC$ por lo cual se selecciona una batería LiPo de 3 celdas, el consumo de los elementos electrónicos que conforman el prototipo esta descrito en la Tabla 38

Tabla 38. Consumo de corriente de cada elemento del circuito

Componente	Consumo [A]
M80 Pro	0,012
FrSky R-XSR	0,07
VTX	0,25
Motores BLDC x4	48

Componente	Consumo [A]
Mateksys F722 SE	0,1
TOTAL	48,432

Debido a la cantidad de corriente necesaria se selecciona una batería con una descarga máxima de 50 A con una capacidad de 2.2 Ah con lo que se calcula la autonomía del prototipo utilizado (33) se obtiene una autonomía mínima de 2 minutos con 44 segundos, la autonomía del prototipo es variable debido a que depende del peso de la carga que se este transportando, una menor carga resultará en un menor consumo energético y mayor tiempo de vuelo.

$$Autonoma = Capacidad_{bat}/I_{consumida} \tag{33}$$

Donde:

$Capacidad_{bat}$: *Capacidad de la batería, en* Ah

$I_{consumida}$: *Corriente consumida, en* A

$$Autonoma = 2{,}2\ Ah/48{,}432\ A = 0{,}0454\ h = 2\ min\ 44\ s$$

9.2. Comunicación con los Variadores (ESC)

La comunicación entre el microcontrolador y los variadores es crítica debido a que esta es la encargada del control de la velocidad de los motores, El controlador cuenta con distintos PIDs que permiten manejar distintas variables, las salidas de estos PIDs serán recibidas por lo que se conoce como el mezclador, este multiplica la salida de los mismos por pesos preestablecidos para cada motor, luego este valor es transformado en el formato correcto para que los variadores sean capaces de leer y ejecutar las acciones requeridas. Existen varios formatos de comunicación explicados a continuación:

9.2.1. Control PWM Analógico (Oneshot)

El control por ancho de pulso hace uso de señales de onda cuadrada en los cuales varía el ciclo de trabajo, en este método existen dos variaciones importantes en cuanto a la interpretación de la señal por parte del variador, en la primera se toma el ciclo de trabajo como variable, es decir un ciclo de trabajo de 0 % será una velocidad nula o frenar, mientras que un ciclo de trabajo del 100 % da como resultado la máxima velocidad posible. En el segundo caso se utiliza una interpretación común en el control de servomotores el cual establece que la variable es el tiempo en alto de la señal, esto quiere decir que nunca se llega a un ciclo de trabajo del 100 % ni a 0 %, generalmente se interpreta un tiempo de $1\ ms$ como freno o velocidad nula y un tiempo de $2\ ms$ como máxima velocidad, este comportamiento se puede visualizar en la Figura 25. Esto conlleva varios problemas, la frecuencia de la señal está limitada por los tiempos establecidos como máximos y mínimos, además el tiempo máximo será un retraso en la señal, otro de los problemas que se presentan en este protocolo conocido como ”Oneshot” es la desincronización entre el microcontrolador y el variador, el error se encuentra en el orden de los ms, por lo que incluso una leve desincronización puede llevar a que la señal no sea interpretada de manera correcta, debido a esto la frecuencia no puede ser aumentada de manera indiscriminada, generalmente se encuentran en el rango de $500\ Hz$ a un máximo de $12\ kHz$, siendo el retraso mínimo de la señal $42\ \mu s$.

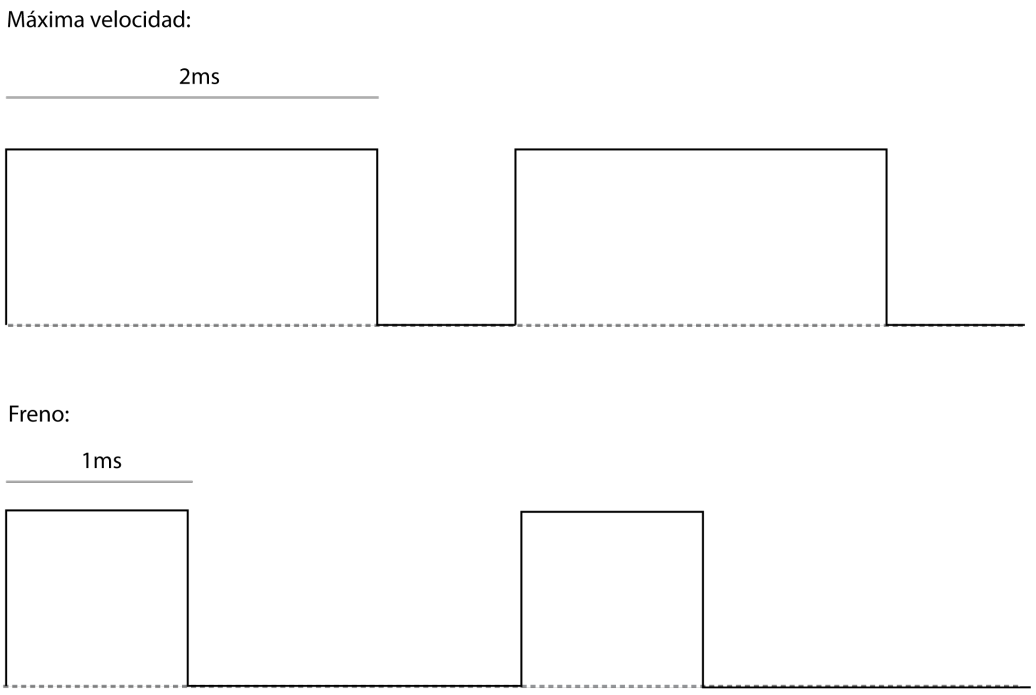


Figura 25. Comportamiento de la señal PWM en Oneshot

9.2.2. Control Digital (Dshot)

Como su nombre lo indica, este método hace uso de una señal digital, esta esta compuesta por 16 bits de los cuales 11 están destinados para el dato de velocidad, su estructura se puede ver en la Figura 26 , por lo que el protocolo tiene una resolución de 2048 pasos, este protocolo cuenta con diferentes velocidad de transferencia que pueden ser configuradas según sea necesario para la aplicación, las posibles configuraciones están presentadas en la Tabla 39

Tabla 39. Características de las distintas configuraciones para el control digital de ESCs, Fuente: [17]

Nombre	Velocidad [kb/s]	Frecuencia [kHz]
Dshot150	150	$9{,}375(106{,}7us)$
Dshot300	300	$18{,}75(53{,}344us)$
Dshot600	600	$37{,}5(26{,}67us)$
Dshot1200	1200	$75(13{,}34us)$

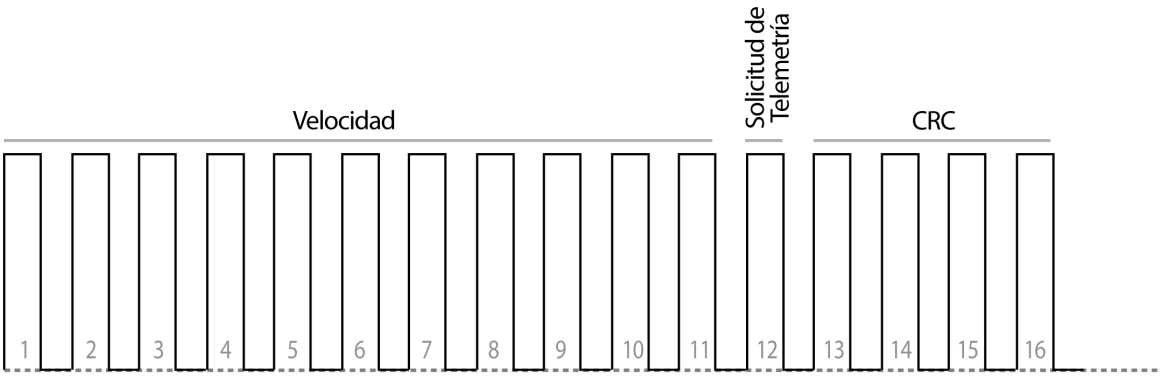


Figura 26. Estructura de un dato enviado mediante el protocolo "Dshot"

Algunas de las ventajas de la utilización de un protocolo digital para la comunicación con los variadores se encuentran en la sensibilidad al ruido la cual es mucho menor que cuando se utilizan protocolos analógicos, además de que este tipo de comunicación es mucho más rápido y no presenta problemas de sincronización, el número de pasos es mayor a la mayoría de los ADC utilizados en el procesamiento de la señal analógica además de que cuenta con detección de errores por medio de CRC-4 (cyclic redundancy check) , el código CRC se agrega en los últimos 4 bits, este código es el resúltate de lógica XOR aplicada al dato que se desea enviar, el receptor aplicará esta misma lógica la cual debe dar como resultado 0000, si el resultado fuera diferente esto indica que son datos corruptos, el uso de este método de detección de errores no aumenta la latencia. La descripción matemática del código CRC es una división binaria con un divisor conocido tanto por el ESC y el microcontrolador, el residuo obtenido por el ESC debe ser 0, esta división se hace transformando el dato en un polinomio, el divisor también será un polinomio, en el caso de CRC-4 el divisor está definido como $x^4 + x + 1$. Para poder transformar el dato en un polinomio se le asigna el termino x^{n-1} de izquierda a derecha al dato tomando en cuenta la posición donde irá el código CRC y se multiplica por el valor correspondiente, por ejemplo:

$$Mensaje : 1011$$

$$10110000 \rightarrow 1x^7 + 0x^6 + 1x^5 + 1x^4 = x^7 + x^5 + x^4$$

$$10110000/10011 \rightarrow (x^7 + x^5 + x^4)/(x^4 + x + 1) = x^3 + x$$

$$R : x^3 + x^2 + x$$

En este caso el resultado de la división binaria es $x^3 + x$ con un residuo $x^3 + x^2 + x$, el residuo establece el código CRC, para el dato 1011 el código CRC será 1110 por lo que el mensaje enviado será 10111110, si al dato enviado se lo vuelve a dividir por el polinomio $x^4 + x + 1$ el residuo será 0, en caso de que se obtenga otro valor quiere decir que existen errores en el mensaje.

9.3. Comunicación por radiofrecuencia

Para la comunicación entre el piloto y el UAV se hace uso de un enlace por radiofrecuencia de 16 canales, en este caso, con una frecuencia portadora de $2{,}4\ GHz$ con modulación FM la cual se realiza mediante el protocolo ACCESS v2 D16 de la empresa FrSky, al ser un protocolo cerrado no se cuenta con descripciones del método de modulación exacto o de encriptación de los datos, de manera general tanto el módulo de transmisión como el módulo de recepción deben ser vinculados para permitir la comunicación a través de los 16 canales con los que cuenta el enlace por radio frecuencia, para poder realizarlo se debe colocar el receptor y el transmisor en modo de vinculación y evitar cualquier tipo de interferencia, la descripción de este proceso se encuentra en [28]. Cada uno de los canales es totalmente programable por lo que puede ser utilizado para distintas aplicaciones, generalmente los primeros 4 canales son reservados para los controles del UAV los cuales son configurados mediante el estándar AETR (Roll [Ail], Pitch [Ele], Aceleración [Thr] y Yaw [Rud]), los nombres de estos canales provienen de su uso en aviación, una vez recibida la señal, el módulo receptor lo transforma en datos que serán enviados por un puerto serial, en contraste a un receptor de radio frecuencia tradicional, no se transforma la señal recibida en ondas PWM lo cual era utilizado para evitar el uso de un microcontrolador. Para la recepción se hace uso de antenas de tipo monopolo de un cuarto de longitud de onda, para poder calcular la longitud de la antena se utiliza (34).

$$L = \lambda/4 = c_{cable}/f \cdot 1/4 = (c/K_s)/f \cdot 1/4 \tag{34}$$

Donde:

λ : *longitud de onda, en* m

c : *velocidad de la luz en el vacío, en* $m/s(299792458\ m/s)$

f : *frecuencia, en Hz* $(2400\ Hz)$

k_s : *factor de velocidad* $(1{,}12)$

c_{cable} : *velocidad de la luz en el cable* $(0{,}89 \cdot c)$

$$L = ((299792458\ m/s)/1{,}12)/(2{,}4 \cdot 10^9\ Hz) \cdot 1/4 = 0{,}112\ m \cdot 1/4 = 2{,}8\ cm$$

En el caso de la antena del módulo de transmisión de radio frecuencia se hace uso de una antena monopolo de un medio de la longitud de onda por lo que se tiene una longitud de 56 mm. Este tipo de conexión tiene un rango máximo aproximado de 1 km en un caso ideal (sin interferencias) pero en aplicaciones reales llega a 500 m, para poder alcanzar distancias mucho mayores se debe cambiar el tipo de conexión, siendo la más eficiente en cuestión de rango y latencia la tecnología LoRa la cual puede alcanzar distancias de hasta 10 km con una potencia de 250 mW, manteniendo la frecuencia de 2.4 GHz, no se hace uso de esta tecnología debido a su baja disponibilidad además de que es considerada tecnología experimental en la comunicación con UAVs.

9.4. Transmisión de video analógico

De manera similar a la comunicación que permite el control por radio frecuencia, se hace uso de la transmisión de video analógico como método de monitoreo del UAV, permitiendo que el piloto tome el control en cualquier momento, incluso cuando no tiene línea de vista, para esto se hace uso de ondas de radio con las siguientes características:

$f = 5{,}8\ GHz \rightarrow$ *Frecuencia portadora*

$\lambda = c/f = c/5{,}8\ GHz = 0{,}0517\ m = 5{,}17\ cm \rightarrow$ *Longitud de onda*

Modulación: Modulación de amplitud en cuadratura (QAM)

Al tener una frecuencia mayor, y por ende una longitud de onda más pequeña, este enlace tiende a fallar a una distancia menor que el del control de $2{,}4\ GHz$. La modulación de esta

onda, como su nombre lo indica, se hace mediante la amplitud de la señal, pero en este caso se requiere transmitir los valores de video siendo estos las intensidades de cada color en un punto dado (RGB) por lo que se utiliza dos señales portadoras desfasadas en 90°, luego estas señales son sumadas para obtener una única señal, esto permite un uso más eficiente del ancho de banda, en el caso del video analógico se construyen estas señales de la siguiente manera:

- Luminancia (Blanco y negro): esta formada por tres valores como se describe en (35).

$$Y = R + G + B \tag{35}$$

- Color: consiste en dos valores dependientes de la Luminancia, el primero descrito por la (36) y el segundo por (37)

$$U = B - Y \tag{36}$$

$$V = R - Y \tag{37}$$

La transmisión de video analógico se realiza en dos etapas, la primera transmite los valores de luz y la segunda la información de color, para poder detectar estas dos etapas se debe tener una frecuencia de sincronización (o de muestreo de la señal) la cual cambia dependiendo de que formato se utilice, $3{,}57954\ MHz$ para NTSC y $4{,}43361875\ MHz$ para PAL, se realiza un pulso de sincronización al inicio de cada línea de video (debido a que el video analógico se maneja de manera entrelazada) y luego se extrae la información. Tanto la cámara como el monitor receptor hace uso de tecnología digital por lo que se debe convertir el video analógico para poder ser mostrado al piloto, la comunicación se muestra en el diagrama de bloques de la Figura 27.

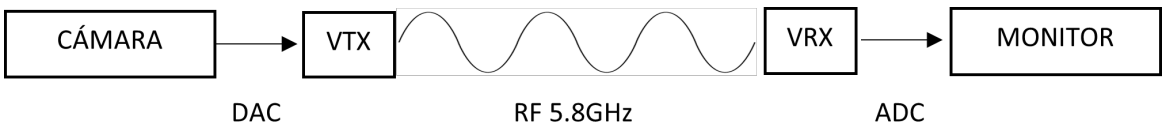


Figura 27. Diagrama de bloques de la transmisión de video analógico

Para la transmisión y recepción de video analógico se hace uso de antenas omnidireccionales con una ganancia de $1{,}6\ dB$ y un ancho de banda de $5{,}5\ GHz$ a $6\ GHz$, este ancho

de banda se debe a que la señal se divide en canales para permitir que existan distintos UAV transmitiendo video sin que exista interferencia, las frecuencias de los canales se muestran en la Tabla 40

Tabla 40. Frecuencias de los canales en Hz para transmisión de video analógico de 5.8GHz, Fuente: [18]

	1	2	3	4	5	6	7	8
A	5865	5845	5825	5805	5785	5765	5745	5725
B	5733	5752	5771	5790	5809	5828	5847	5866
E	5705	5685	5665	5645	5885	5905	5925	5945
F	5740	5760	5780	5800	5820	5840	5860	5880
R	5658	5695	5732	5769	5806	5843	5880	5917

Las antenas utilizadas para la transmisión de video son de tipo omnidireccional, específicamente el modelo AXII de la marca Xilo, su patrón de transmisión se puede ver en la Figura 28.

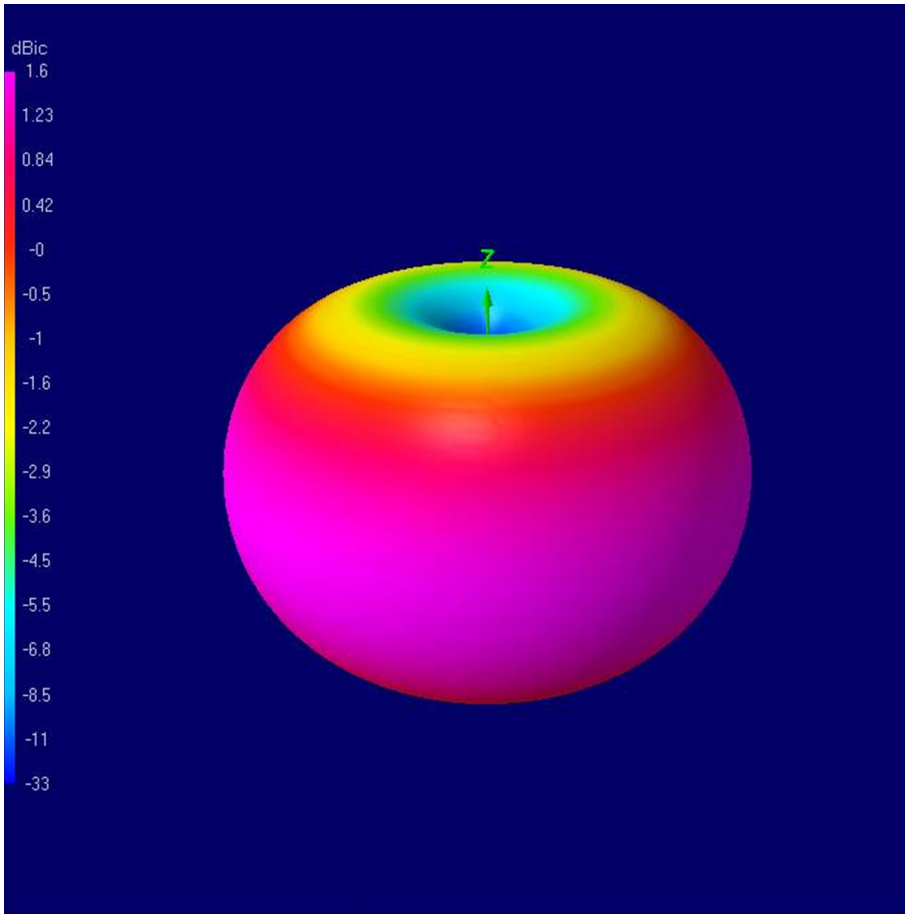


Figura 28. Patrón de transmisión de la antena AXII 5.8 GHz, Fuente: [11]

Utilizando (34) se calcula la longitud de la antena necesaria para la transmisión y recepción de señales de 5.8 GHz, se asume una antena de 1/4 de longitud.

$$L = ((299792458\ m/s)/1{,}12)/(5{,}8 \cdot 10^9\ Hz) \cdot 1/4 = 0{,}046m \cdot 1/4 = 1{,}15\ cm$$

9.5. Sistemas satelitales de navegación global (GNSS)

El acrónimo GNSS hace referencia a “global navigation satellite systems” (sistemas satelitales de navegación global) los cuales tienen como objetivo brindar información de geolocalización de manera precisa, esto implica latitud, longitud y elevación, esto se logra mediante una constelación de satélites los cuales brindan datos de tiempo y basándose en el la hora de llegada conocida como TOA (time of arrival) y el tiempo de transmisión conocido como TOT (time of transmission) se puede calcular la distancia entre el satélite transmisor y el dispositivo receptor, se requiere un mínimo de 4 satélites para poder conocer la latitud, longitud y altura, debido a esto en la implementación en el UAV se configura un requerimiento mínimo de 5 satélites para poder realizar operaciones de vuelo, esto permite que en caso de que se pierda la señal de alguno aun se puede conocer la localización del dron. Existen tres sistemas de localización disponible para ser utilizados en aplicaciones generales, estos son los siguientes:

9.5.1. GPS

Sus siglas provienen de su nombre en inglés “Global Positioning System” (sistema de posicionamiento global) este sistema es mantenido por el gobierno estadounidense desde 1978 (siendo operativo desde 1995) y cuenta con aproximadamente 34 satélites activos, este número varia debido a actualizaciones y mantenimiento del sistema. Su precisión es de 30 a 500 cm dependiendo del número de satélites disponibles.

9.5.2. GLONASS

Sus siglas provienen de su nombre en ruso “Global’naya Navigatsionnaya Sputnikovaya Sistema” (sistema satelital de navegación global) creado originalmente por la unión soviética en 1982, y operativo desde 1995. Existen dispositivos compatibles con GLONASS y

GPS, esto permite aumentar el número de satélites disponibles, así como la cobertura de estos y la precisión de la geolocalización, la constelación de satélites cuenta con 24 satélites opertativos. Este sistema cuenta con una precisión de 2.8 a 7.38 metros, sin embargo, puede llegar a presiones mucho mayores si se usa en conjunto con GPS, debido al posicionamiento de los satélites GLONASS es el sistema más preciso cerca de los polos terrestres.

9.5.3. Galileo

Su nombre completo es “Galileo positioning system” (sistema de posicionamiento Galileo) creado por la Unión Europea y disponible desde 2011, este sistema fue desarrollado para uso militar para poder brindar geolocalización sin depender de satélites de GPS o GLONASS, cuenta con dos versiones, la de uso general con una precisión de 1 metro y un sistema de alta precisión el cual puede llegar a obtener posiciones con un error de 1 cm, existen versiones militares del sistema con encriptación y sistemas de transmisión más robustos, estos últimos están limitados a uso militar. el tamaño de la constelación es de 30 satélites de los cuales 24 están operativos y 6 son considerados satélites de emergencia. Gracias a la cooperación entre la Unión Europea y Estados Unidos, el sistema es compatible de manera comercial con GPS, siendo el conjunto de estos sistemas la mejor alternativa publica para servicios de posicionamiento.

9.5.4. Trilateración

Un componente importante de todos los sistemas de posicionamiento son las estaciones de control de los satélites las cuales con las encargadas de mantener el buen funcionamiento de las constelaciones y corregir cualquier error de cálculo que pueda surgir debido a la órbita lunar y terrestre. Para poder determinar la posición del receptor se calcula el tiempo transcurrido entre que se envió el mensaje y su recepción, esto se conoce como TOF (time of flight), una vez se conocen las distancias a 4 satélites es posible generar esferas desde la posición de cada satélite (la cual es conocida por el receptor y actualizada cada vez que se inicia una conexión), el radio de estas esferas son las distancias a los satélites,

luego se busca los puntos de intersección de estas, si los datos obtenidos son correctos debe existir un único vértice de intersección entre todas las esferas, el cual corresponde a la ubicación del receptor. Este proceso se conoce como trilateración y se puede observar en la Figura 29 , el cual aumenta su precisión a medida que un mayor número de satélites esta disponible, especialmente si lo que se requiere es determinar la posición en el espacio y no únicamente longitud y latitud. El sistema de posicionamiento del UAV utiliza satélites GPS y Galileo para determinar su posición, para poder determinar su orientación hace uso de un magnetómetro.

Una vez se conoce la posición en el espacio, el UAV es capaz de utilizar distintos modos, entre estos se encuentran mantener la posición en el espacio (con una tolerancia que depende de la precisión de la señal del sistema GNSS), regreso al punto de partida (RTH) en caso de perdida de señal y misiones autónomas preprogramadas, para poder realizar esto se hace uso de dos controladores PID, uno para la posición y otro para controlar la velocidad, para asegurar el funcionamiento correcto se debe tener en cuenta la taza de actualización del sistema y la latencia en las comunicaciones.

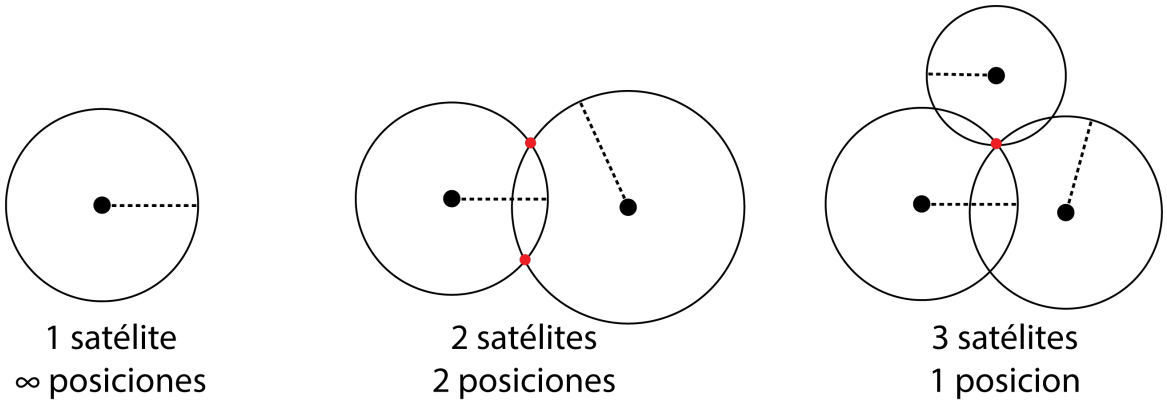


Figura 29. Proceso de Trilateración

Otro punto importante es el sistema de coordenadas geográficas angulares utilizado para representar la posición, este permite representar con dos ángulos internos (su vértice es el centro de la tierra) cualquier punto en la superficie terrestre, sus componentes son:

- Latitud[φ]: ángulo interno medido desde la linea ecuatorial hacia nuestra ubicación, este componente tiene un rango de $-90°$ a $90°$ siendo positivo hacia el norte y negativo hacia el sur.

- Longitud[λ]: ángulo interno medido desde el meridiano 0 (Greenwich), el ángulo de longitud tiene un rango de -180 a 180 siendo positivo hacia el este y negativo hacia el oeste

La norma que establece este estándar es la WGS84 descrito en [29].

9.6. Sistema de control automático

El prototipo requiere un sistema de control automático que permita seguir los comandos dados por el piloto o las instrucciones de la misión que permitan seguir la ruta preprogramada de tal manera que se controle la velocidad de los motores, así como la posición del dron en 3 ejes rotacionales, estos requerimientos deben tomar en cuenta el estado del UAV en cada momento por lo que se requiere un control en lazo cerrado, por lo que se busca implementar controladores PID para realizar esta tarea, este es el controlador más comúnmente utilizado en la industria y está definido por (38) según [30], previo a la implementación del sistema de control se realiza un diagrama de bloques como se muestra en la Figura 30 .

$$u(t) = K \cdot (e(t) + 1/T_i \cdot \int_0^t e(t) \cdot \ dt + T_d \cdot de(t)/dt) \tag{38}$$

Donde:

K : *Ganancia proporcional, adimensional*

T_i : *Tiempo integral, en s*

T_d : *Tiempo derivativo, en s*

$e(t)$: *Error*

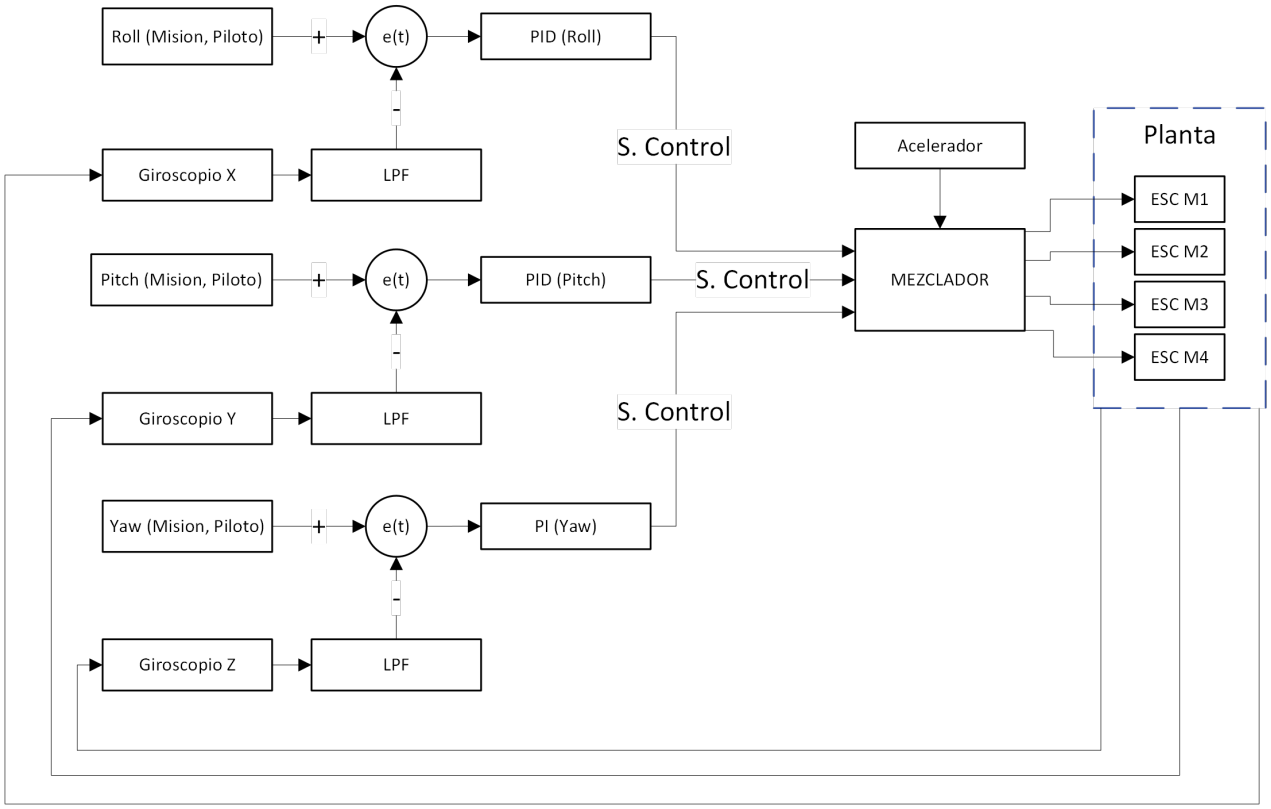


Figura 30. Diagrama de bloques del sistema de control automático

Para la implementación del sistema en el microcontrolador se debe utilizar controladores discretos, esta discretización se define como se muestra en la ecuación (39) el cual utiliza una aproximación bilineal o de Tustin, el tiempo de muestreo es limitado por la velocidad de comunicación entre el microcontrolador y el sensor (800 kHz), sin embargo la señal a medir es menor a 150 Hz por lo que por el teorema de Nyquist se establece la frecuencia de muestreo mínima en 300 Hz, esta da como resultado un tiempo de muestreo máximo Tm de 3.3 ms.

$$u_k = u_{k-1} + q_0 \cdot e_k + q_1 \cdot e_{k-1} + q_2 \cdot e_{k-2} \tag{39}$$

$$q_0 : k_p + k_i \cdot T_m/2 + k_d/T_m$$

$$q_1 : -k_p + k_i \cdot T_m/2 - 2 \cdot k_d/T_m$$

$$q_2 : k_d/T_m$$

Donde:

k_p : *Ganancia proporcional, adimensional*

k_i : *Ganancia integral, adimensional*

k_d : *Ganancia Derivativa, adimensional*

T_m : *Tiempo de muestreo, en* s

9.6.1. Mezclador

Debido a que se requiere controlar 4 motores, pero tenemos únicamente 3 controladores, se hace uso de un “Mezclador”, su función es la de transmitir las señales de control dependiendo de la influencia que tiene un actuador sobre la acción de control, esto se logra mediante la ecuación (40), el vector de pesos para cada motor será diferente, esto depende de la configuración geométrica de estos, en este caso se está utilizando una configuración de tipo quadcopter “X” y en el caso de la rotación en Z (yaw) el sentido de giro de las propelas también tendrá influencia en los pesos.

$$S_M = C_0 \cdot \ a + C_1 \cdot S_{roll} + C_2 \cdot S_{pitch} + C_3 \cdot S_{yaw} \tag{40}$$

Donde:

S_{roll} : *Señal de control para el ángulo roll*

S_{pitch} : *Señal de control para el ángulo pitch*

S_{yaw} : *Señal de control para el ángulo yaw*

a : *Aceleración*

C : *Pesos determinados para los parámetros del motor M*

Antes de establecer los pesos de los motores es importante conocer su configuración y asignar una numeración, en el caso del prototipo se da como se muestra en la Figura 31 , esta es la numeración utilizada comúnmente en el desarrollo de multirotores, siguiendo este esquema se establecen los pesos como se muestra en la Tabla 41 de tal manera que se puedan controlar todas las acciones del dron, los motores responsables de estas acciones están establecidos de la siguiente manera:

- Pitch: para un movimiento positivo (hacia adelante) se utilizan los motores 1 y 3 mientras que un movimiento negativo (hacia atrás) se hace uso de los motores 2 y 4
- Roll: para realizar un movimiento positivo (hacia la derecha) se utilizan los motores 3 y 4 y para realizar un movimiento negativo (hacían la izquierda) se encargan los motores 1 y 2
- Yaw: Este movimiento se basa en la inercia generada por el movimiento rotacional de la propela (razón por la cual se coloca un número igual de propelas con rotación horaria y antihoraria) para un giro positivo (horario) se hace uso de los motores 1 y 4 mientras que para un giro negativo (antihorario) se utilizan los motores 2 y 3.

La salida de los 3 ejes rotacionales debe ser normalizada antes de multiplicarla por la aceleración a ya que no se están trabajando con motores reversibles por lo que el rango de los valores de salida es entre 0 y a, también se establece que en cualquier escenario el valor mínimo de velocidad de cada motor es del 10 %, esto se realiza para aumentar la velocidad de respuesta del UAV ya que un motor en reposo tiene un mayor tiempo de reacción.

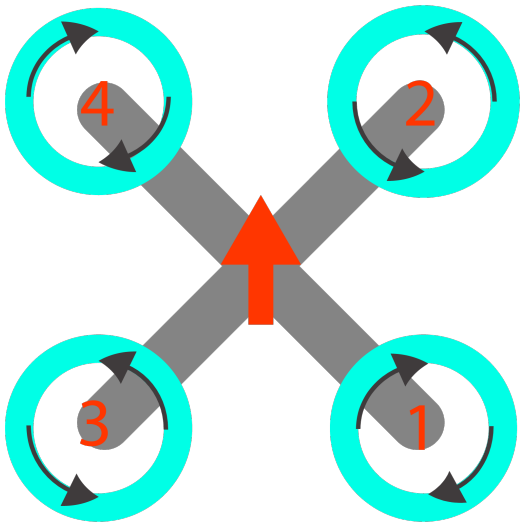


Figura 31. Numeración de los motores del prototipo

Tabla 41. Pesos establecido en el mezclador para un quadcopter en configuración tipo X

MOTOR	C0	C1	C2	C3
1	1	-1	1	1
2	1	-1	-1	-1

MOTOR	C0	C1	C2	C3
3	1	1	1	-1
4	1	1	-1	1

9.6.2. Sintonización de los PIDs

Debido a que existen tres controladores responsables del posicionamiento del dron (los cuales son dependientes unos de otros) la sintonización de los parámetros no se puede realizar de manera efectiva utilizando métodos tradicionales como Ziegel – Nichols o Choen – Coon, debido a esto existe una gran cantidad de metodologías en desarrollo para lograr esta tarea, sin embargo al ser métodos experimentales se optó por modelar el sistema como caja negra como se muestra en el diagrama de bloques de la Figura 32 a través del cual obtenemos un registro temporal de los valores percibidos por los sensores. Para que esto funcione es necesario contar con valores iniciales para los controladores PID, estos se muestran en la Tabla 42 , de esta manera se consigue un vuelo inicial en el que están presentes oscilaciones pero permite la obtención de datos de los sensores y por medio de una proceso iterativo se refinan las ganancias de los controladores hasta obtener un vuelo estable.

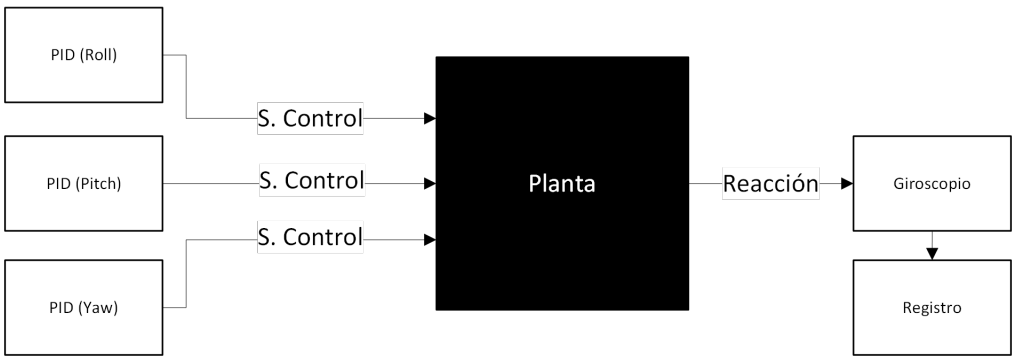


Figura 32. Diagrama de bloques del sistema de caja negra

Tabla 42. Valores preliminares para la sintonización de los controladores PID

EJE	P	I	D
ROLL	40	30	23
PITCH	40	30	23

EJE	P	I	D
YAW	85	45	0

En el caso del control en el eje Z (Yaw) se tiene un control tipo PI esto se debe a que se no se requiere un control tan robusto como uno los otros ejes ya que no se ve afectado por efecto de la gravedad y es menos sensible a las perturbaciones externas, de esta manera se reduce la carga computacional y permite un control más efectivo en los ejes X y Y. En los vuelos preliminares se notó un gran número de oscilaciones por lo que las constantes de los controladores en Roll y Pitch fueron modificados hasta reducirlas hasta el mínimo posible, esto se logra principalmente aumentando el valor de Kd, este último se realiza monitoreando la temperatura de los motores ya que un valor elevado puede causar sobrecalentamiento ya que tiene un efecto de amplificación en frecuencias elevadas (como las causadas por las vibraciones de los motores) de esta manera se llega a los valores mostrados en la Tabla 43 , la afinación de los controladores se realizó utilizando un dron sin carga.

Tabla 43. Valores preliminares para la sintonización de los controladores PID

EJE	P	I	D
ROLL	55	20	60
PITCH	55	20	60
YAW	85	45	0

Una vez sintonizados los controladores se realizan vuelos con el objetivo de obtener datos de los sensores y mediante el uso del programa Matlab se obtiene la respuesta en escalón unitario del sistema (por cada eje) como se muestra en la Figura 33. Se puede ver que se obtiene una respuesta subamortiguada en el caso de los ejes de Pitch y Roll lo cual no es ideal ya que se busca obtener una respuesta críticamente amortiguada, sin embargo un control más agresivo puede resultar en daños en los componentes, con esta afinación los controladores son capaces de seguir las instrucciones dadas por el piloto o las preprogramadas según la tarea que se esté realizando pero no será capaz de realizar maniobras agresivas o llegar a velocidades elevadas. En el eje Z se tiene una respuesta sobre amortiguada, esto no afecta a la maniobrabilidad del dron ya que el tiempo de respuesta es corto

($\approx$ 200 ms).

Para el control de posición mediante GNSS se utiliza un control proporcional sencillo el cual depende de la distancia de la posición objetivo, debido a que las trayectorias utilizadas son lineales este control es sencillo y no presenta mayor complicación, esto también aplica al control de altura en el cual se hace uso del barómetro.

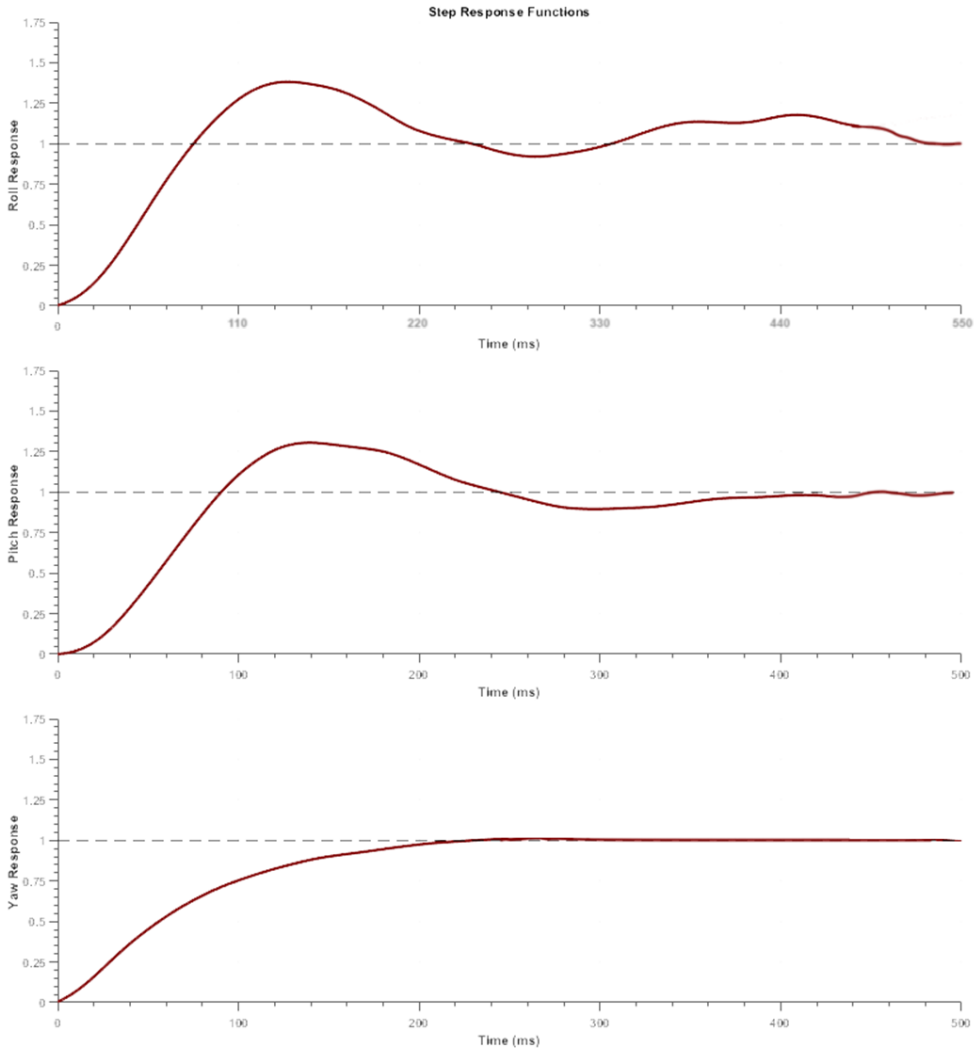


Figura 33. Respuesta en escalón unitario del sistema

9.6.3. Filtrado

Se hace uso de filtros pasa bajo en la salida para lo cual se requiere determinar la frecuencia de corte, para esto se hace uso de la transformada de Fourier rápida (FFT), esta es una optimización de la transformada de Fourier para tiempo discreto (DFT) la cual nos permite convertir los datos obtenidos en el dominio del tiempo en dominio de la frecuen-

cia, esto permite que los cálculos se realicen utilizando el microprocesador ya que el costo computacional de la función FFT es mucho menor. Para la configuración de los filtros pasa bajo se hace uso de los registros de los sensores obtenidos mediante el modelo de caja negra, con la ayuda del software Matlab y la librería Blackbox Explorer se obtiene la respuesta en el dominio de la frecuencia como se muestra en la Figura 34.

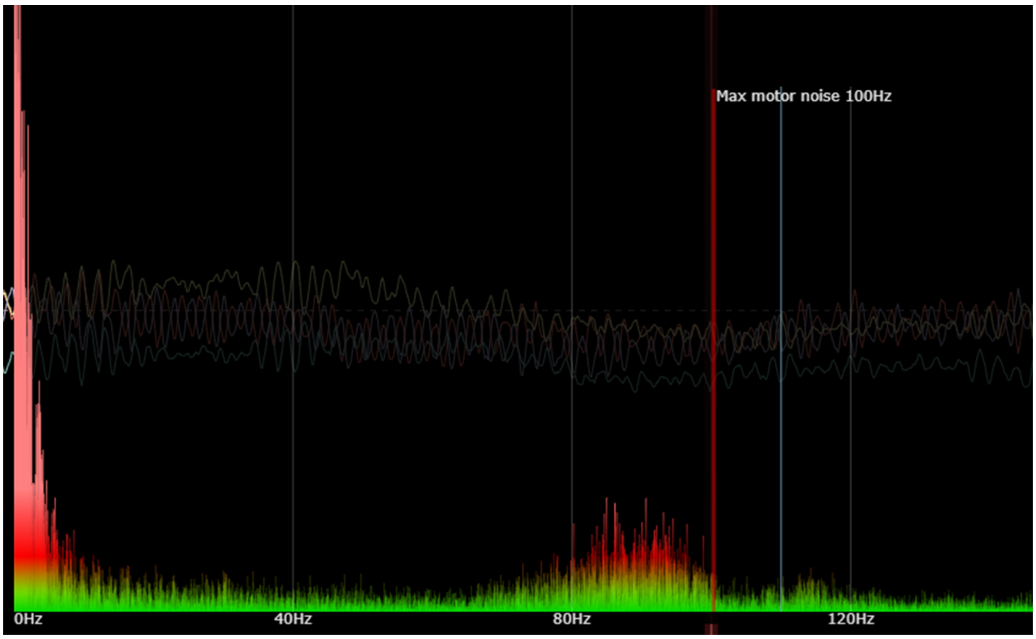


Figura 34. Datos del giroscopio en el dominio de la frecuencia

De esta manera se establece una frecuencia de corte de 60 Hz, uno de los que se tomó en cuenta en el diseño geométrico del prototipo es el hecho de que las propelas de tamaño mayor a 7 pulgadas tienden a bajar la frecuencia en la que se produce el ruido por lo que puede llegar a mezclarse con los datos de movimiento del dron (<24Hz) haciendo que el filtrado sea imposible, esto se logra mediante la reducción de la longitud de los brazos.

10. Desarrollo de la Programación

10.1. Diagrama de funcionamiento del sistema

El sistema implementado para la realización de entregas requiere una infraestructura informática que permita generar rutas de manera sencilla, debido a que el prototipo no cuenta con un sistema de evasión de obstáculos, el operador deberá garantizar que las

conexiones entre los nodos generen una conexión en la que el tránsito sea seguro, esta tarea se vuelve más complicada a medida que el número de nodos aumenta, debido a esto se busca reducir el número de personas capaces de modificar la base de datos (creación y eliminación de nodos) mediante la implementación de sistemas separados para esta tarea y la de generación de rutas. Se realiza un diagrama de funcionamiento del software mostrado en la Figura 35 el cual busca resumir la interacción entre los distintos programas.

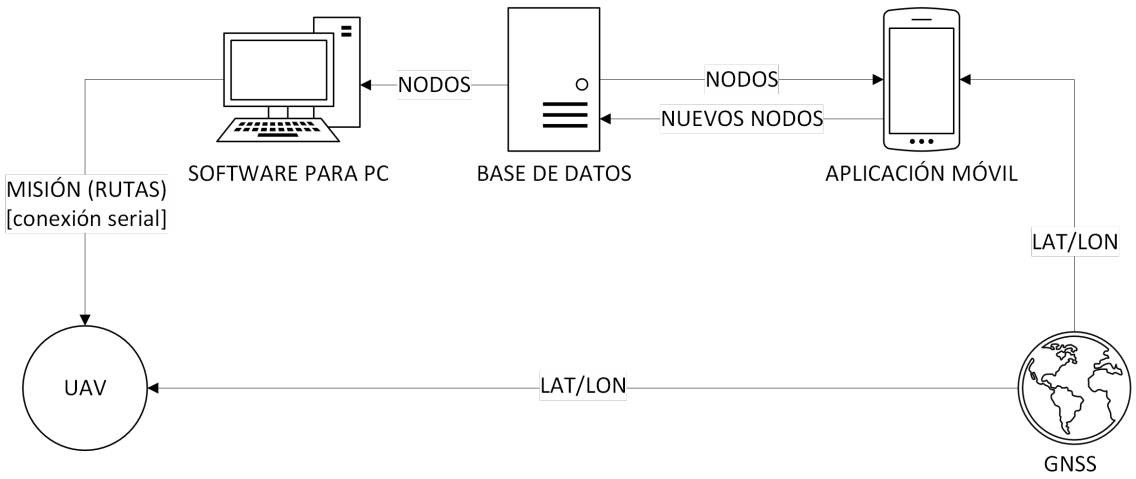


Figura 35. Diagrama de bloques de la interacción entre los programas

Una de las funciones principales del software se relaciona con la visualización de las ubicaciones de los nodos en un mapa, debido a esto es necesario utilizar un proveedor de servicio de mapas el cual pueda ser integrado de manera sencilla en el lenguaje de programación seleccionado es importante recalcar que se utiliza la ubicación en tiempo real del dispositivo mediante tecnología GNSS (en el caso de la aplicación móvil) por lo que el proveedor de mapas debe ser lo suficientemente rápido como para mantener la visualización de manera fluida a medida que el usuario se mueva, debido a estas razones se decide utilizar la API desarrollada por la empresa MapBox, la cual cuenta con la ventaja de ofrecer el servicio de manera gratuita para clientes con aplicaciones no comerciales que cuenten con una densidad de usuarios menor a 25000 usuarios mensuales, si bien la empresa brinda servicios de navegación y generación de rutas, se utiliza únicamente los mapas y coordenadas proporcionadas ya que el UAV utiliza un formato específico para realizar vuelos de manera autónoma. Los mapas obtenidos cuentan con datos de elevación de terrenos y aproximaciones de la geometría de los edificios en zonas urbanas, esto permite que el pro-

grama encargado de la creación de conexiones entre nodos identifique si existen colisiones en la ruta, lo que reduce la posibilidad de choques.

10.2. Aplicación móvil

Se desarrolla una aplicación con el objetivo de manejar la base de datos que servirá para guardar las coordenadas de los nodos los cuales se usarán en el proceso de creación de rutas para generar conexiones, así como también como puntos de despegue y aterrizaje del UAV, para esto se establecen los requerimientos mínimos de funcionalidad de la aplicación, estos son los siguientes:

- La aplicación debe ser capaz de leer y escribir datos en la base de datos
- La aplicación debe permitir que el usuario agregue nuevos nodos a la base de datos
- La aplicación debe permitir que el usuario elimine nodos de la base de datos
- No se pueden agregar nodos con un nombre nulo o repetido.
- La aplicación únicamente permite modificar el área donde se encuentre el dispositivo
- El usuario debe ser capaz de visualizar la ubicación de los nodos en el mapa, así como en formato tipo lista.

Se decide desarrollar la aplicación en el lenguaje de programación C# ya que este cuenta con gran facilidad para generar software multiplataforma, esto permite que se utilice el código desarrollado tanto en Android como en dispositivos con sistemas operativos IOS, debido a la facilidad de desarrollo y las herramientas disponibles, se toma como objetivo principal el sistema operativo para teléfonos móviles Android, el programa utilizado para su implementación es Unity en conjunto con Microsoft Visual Studio. El funcionamiento de la aplicación esta descrito por el diagrama de flujo que se presenta en la Figura 36.

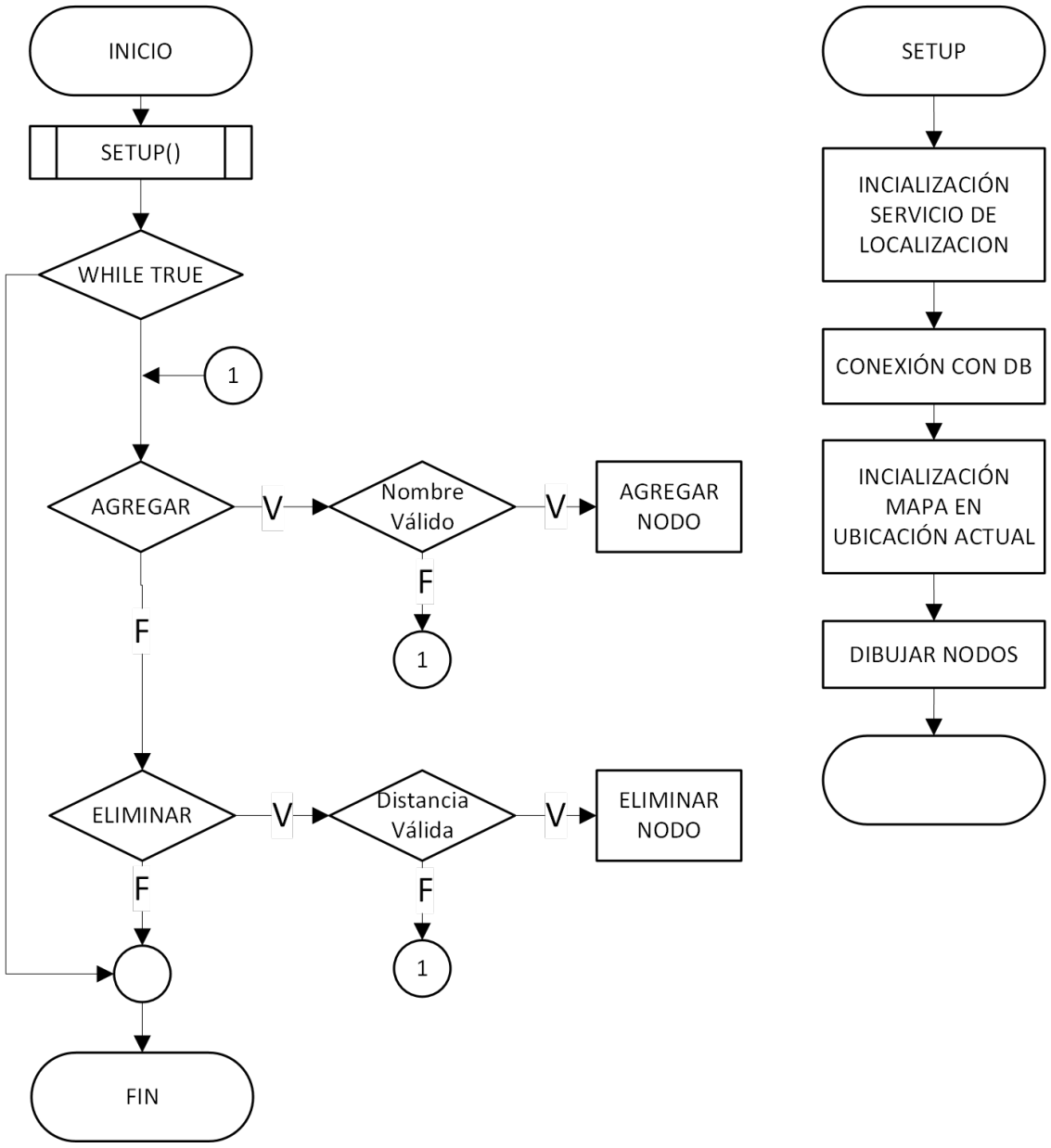


Figura 36. Diagrama de flujo de la aplicación móvil

La interfaz gráfica de la aplicación se desarrolla de manera que sea amigable con el usuario, para que no existan confusiones se muestran únicamente los botones relevantes para las acciones posibles dependiendo del escenario en el que se encuentre, en la Figura 37.

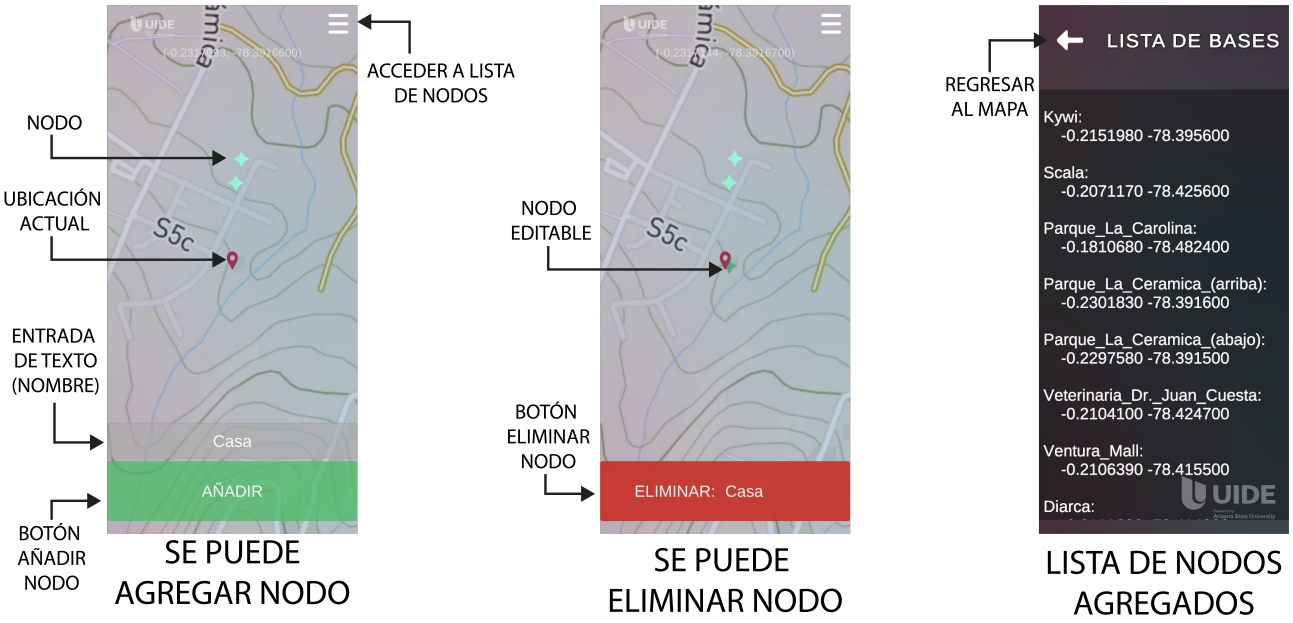


Figura 37. User Interface (UI) de la aplicación móvil

Debido al funcionamiento de la aplicación es importante determinar en qué condiciones se considera que se puede agregar un nodo y en cuales se puede eliminar, esto esta está establecido de tal manera que si nuestra ubicación actual se encuentra a una distancia mayor a $10\ m$, de no cumplirse esta condición quiere decir que existe un nodo en el área, a este se le considera el nodo editable, al utilizar este comportamiento se asegura que los nodos tengan una distancia mínima entre y que no existan solapamiento ya que esto podría causar problemas al momento de generar rutas.

10.3. Programa de creación de rutas

Este programa tiene como objetivo el permitir al usuario seleccionar un nodo de inicio y un nodo final (punto de entrega) para luego generar la ruta de manera automática mediante un algoritmo de búsqueda de rutas, una vez calculado se pasa la información a la memoria EEPROM del UAV lo cual le permitirá realizar el vuelo autónomo, para poder realizar estas tareas el software se desarrolla en el lenguaje de programación C# mediante lo cual ganamos acceso al api utilizada en la aplicación móvil para obtener los mapas (MapBox), este programa tiene acceso a la base de datos de nodos, sin embargo no cuenta con la capacidad de modificar los datos de la misma, una vez se cargan estos datos el programa genera una matriz de adyacencia modificada para lo cual realiza el cálculo de la distancia entre

cada uno de los nodos, una matriz de adyacencia tradicional se compone de la siguiente manera:

$$\begin{matrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{matrix}$$

Donde el numero 1 indica que existe una conexión; La matriz de adyacencia modificada se compone de la siguiente manera:

$$\begin{matrix} 0 & d_{12} & -1 \\ d_{21} & 0 & d_{23} \\ -1 & d_{32} & 0 \end{matrix}$$

En este caso el -1 indica que no existe conexión, todos los otros valores serán llenados con la distancia entre los nodos, tiene simetría en la diagonal principal por lo que $d_{12} = d_{21}$ y $d_{23} = d_{32}$, se utiliza esto ya que permite que no sea necesario recalcular las conexiones o las distancias sin importar si cambian los puntos de inicio o fin de la ruta objetivo. El programa cuenta con una interfaz gráfica (UI) dinámica lo cual implica que sus elementos cambian según los datos brindados por el usuario, sus partes se muestran en la Figura 38

Figura 38. User Interface (UI) del programa de creación de rutas

10.3.1. Búsqueda de rutas

La búsqueda de rutas se realiza mediante el algoritmo A* el cual es una mejora del algoritmo de Dijkstra, su diferencia radica en el uso de una heurística utilizada para calcular los pesos de las conexiones entre los nodos, el diagrama de flujo del algoritmo implementado se muestra en la Figura 39.

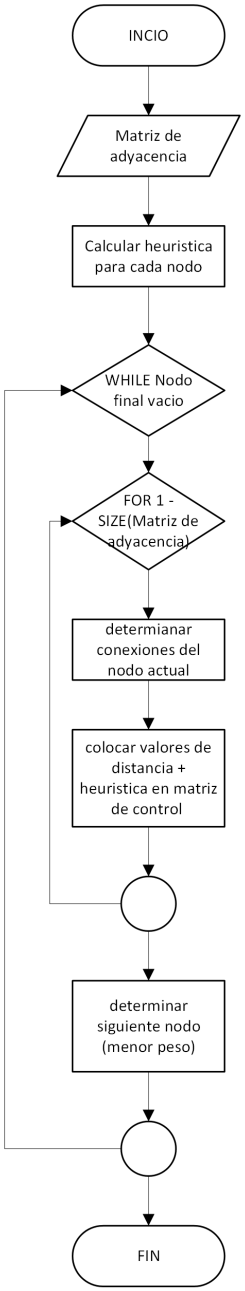


Figura 39. Diagrama de flujo para la implementación del algoritmo A*

Una parte importante para el funcionamiento de este algoritmo es la matriz de control, esta matriz tiene dimensiones 3 x m, donde m es el número de nodos existentes, la primera

columna indica el valor final de distancia del nodo, si este valor es fijado y no se podrá modificar (este valor se fija al momento de convertirse en el nodo actual), el segundo valor es su valor temporal, el cual será su menor valor encontrado hasta el momento, el tercer valor será el predecesor, este número indica el índice del nodo desde el que fue calculada la distancia. Cuando se llega al nodo final objetivo se debe armar la ruta, esto se logra mediante el seguimiento de los predecesores de cada nodo.

10.3.2. Comunicación serial

Una vez encontrada la ruta se debe transmitir por medio del puerto serial hacia el microcontrolador, para esto se sigue el proceso mostrado por el diagrama de flujo de la Figura 40, es importante notar que antes de mandar los comandos que determinan la ruta se debe inicializar el modo CLI (command line interface) el cual permite que el microcontrolador interactúe con los datos de la EEPROM, esto se logra enviando “#” por medio del puerto serial.

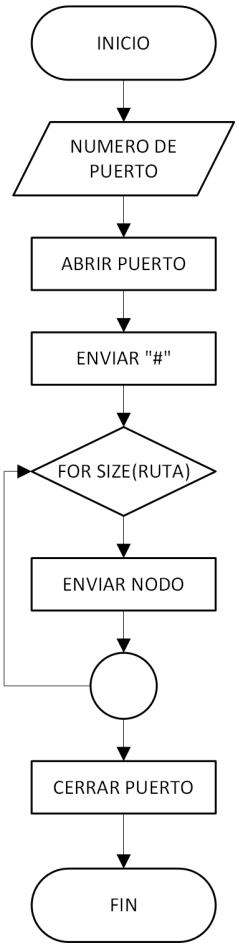


Figura 40. Diagrama de flujo para la transmisión de rutas

El formato para enviar las instrucciones de navegación de la ruta debe contener todos los requerimientos para que sea considerado válido por el microcontrolador, este formato es 'wp' + wpn + acción + latitud + longitud + altitud + velocidad+ 0 0 +flag , estos parámetros se describen de la siguiente manera:

- wpn: número de nodo
- acción:1=movimiento,8=aterrizaje
- latitud: ángulo de latitud del nodo según WGS84
- longitud: ángulo de longitud del nodo según WGS84
- altitud: altura en cm
- velocidad: velocidad en cm/s
- flag: su valor predeterminado es 0, el valor 165 indica el último nodo de la misión

Una vez enviadas las instrucciones, puerto Serial se cierra, permitiendo desconectar el dron. En caso de que la ruta tenga una longitud mayor de la que el UAV es capaz, entonces se crearan sub-rutas con puntos de aterrizaje intermedio que permiten recargar su batería y evitar el sobre calentamiento de los motores.

10.4. Cálculo de distancia a partir de coordenadas de geolocalización

Una de las principales necesidades para el desarrollo tanto del programa de generación de rutas así como en la aplicación móvil es calcular la distancia que existe ya sea entre nodos o entre el dispositivos y los nodos cercanos, para poder realizarlo no se pueden utilizar métodos geométricos o matemáticos tradicionales debido a que las coordenadas hacen referencia a puntos en la superficie terrestre, esto quiere decir que no nos encontramos en un caso en el que se pueda aplicar la geometría euclídea si no se debe hacer uso de geometría elíptica e incluso así no será suficiente ya que el sistema de coordenadas es especializado en geolocalización, sin embargo esto es una problemática conocida por lo que se han desarrollado ecuaciones que permiten calcular esta distancia, cabe recalcar que debido a la forma de la tierra, estos métodos son aproximaciones ya que no nos encontramos en el

caso de una esfera perfecta. Los métodos posibles para el cálculo de la distancia entre dos coordenadas son los siguientes:

10.4.1. Fórmula del Semiverseno (Haversine)

La fórmula de semiverseno asume que la tierra es una esfera perfecta por lo que se trata de una aproximación, sin embargo, los cálculos no tienen un costo computacional elevado y en la mayoría de las aplicaciones el error de cálculo no afecta de manera significativa. Según [31], el semiverseno se define como (41) , esta formulación de la distancia nace a partir de la definición del ángulo central como $\theta = d/r$ donde d es la distancia esférica entre dos puntos y r es el radio de la esfera (en este caso la tierra) como se muestra en la Figura 41, esto implica que $hav(\theta) = hav(d/r)$.

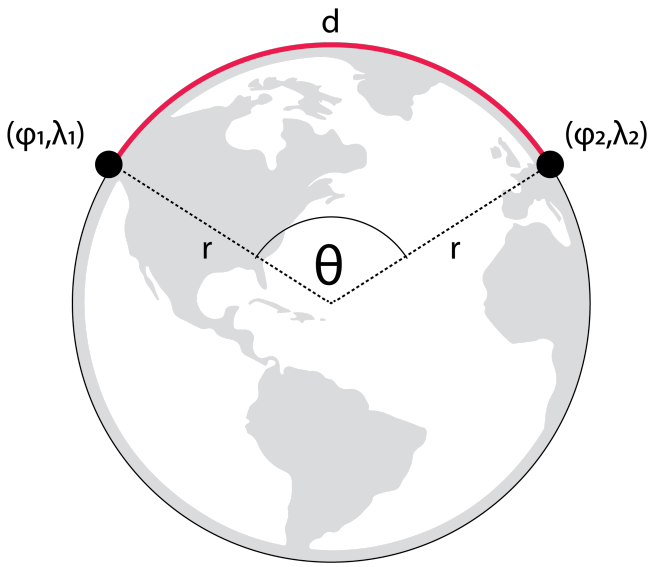


Figura 41. descripción de las dimensiones utilizadas en la formula del semiverseno

Existe una segunda formulación del semiverseno que permite el uso de coordenadas de latitud y longitud, esta está definida como (42) por lo que para obtener el valor de la distancia entre dos puntos de los que se conoce sus coordenadas se debe definir el inverso del semiverseno como (43), al reemplazar (42) en (43) se obtiene la fórmula del semiverseno como se muestra en (44).

$$hav(\theta) = sen^2\ (\theta/2) \tag{41}$$

Donde:

θ : *ángulo central entre dos puntos de una esfera*

$$h = hav(\theta) = hav(\varphi_2 - \varphi_1\ ) + \cos(\varphi_1\ ) \cdot \cos(\varphi_2\ ) \cdot\ hav(\lambda_2 - \lambda_1) \tag{42}$$

Donde:

$$\varphi_1 : latitud\ punto\ 1$$

$$\varphi_2 : latitud\ punto\ 2$$

$$\lambda_1 : longitud\ punto\ 1$$

$$\lambda_2 : longitud\ punto\ 2$$

$$d = r \cdot\ archav(h) = 2 \cdot\ r \cdot\ arcsen(\sqrt{h}) \tag{43}$$

Donde:

d : *distancia esférica entre dos puntos*

r : *radio de la esfera*

h : *formulación del semiverseno para latitud y longitud*

$$d = 2 \cdot\ r \cdot\ arcsen(\sqrt{sen^2\ ((\varphi_2 - \varphi_1)/2) + \cos(\varphi_1\ ) \cdot \cos(\varphi_2\ ) \cdot\ sen^2\ ((\lambda_2 - \lambda_1)/2)\ )} \tag{44}$$

10.4.2. Algoritmo de Vincenty

Este algoritmo recursivo es el que brinda la mayor precisión en el cálculo de distancias, esto inicia desde su formulación la cual asume que la tierra es un elipsoide, lo cual es mucho más certero que la esfera que se asume en las otras formulaciones, uno de los problemas que presenta esta formulación es el hecho de que para puntos antipodales el algoritmo no converge generando un bucle infinito, por lo que generalmente se limita el número de iteraciones a realizarse dando un error al momento de superar el límite, el algoritmo completo con una explicación detallada se puede encontrar en [32] .

En la implementación de los programas se hace uso de la fórmula del semiverseno

debido a que el costo computacional es menor y no se corre el riesgo de generar un bucle infinito que impida el correcto funcionamiento del programa.

11. Construcción y pruebas de funcionamiento

11.1. Proceso constructivo

Para el proceso constructivo se adquieren y manufacturan los distintos componentes necesarios para la estructura mecánica, así como para el circuito electrónico. Este proceso se divide en varias partes explicadas en las siguientes secciones.

11.1.1. Construcción de la estructura principal

La estructura principal del dron se realiza utilizando laminado de fibra de carbono como material, debido a que no se encuentra el material en el país y debido al los precios elevados de las planchas, así como el costo de importación debido al tamaño de las planchas de este material, se opta por utilizar piezas disponibles en el mercado para construcción de drones las piezas seleccionadas corresponden a el modelo XL10 V5 fabricado por la empresa iFlight que se muestra en la Figura 42.

Figura 42. Estructura de laminado de fibra de carbono XL10 V5, Fuente: [12]

Los cálculos realizados en capítulos anteriores fueron hechos tomando en cuenta estas consideraciones. También se hizo uso de manufactura aditiva mediante impresión 3D debido a los requerimientos específicos del dron como es el módulo GPS, el módulo de transmisión de video analógico y el módulo de recepción de radio frecuencia seleccionado, para su fabricación se hizo uso de la impresora 3D “Ender 3 Pro”. En esta etapa también se implementa el circuito electrónico como se muestra en la Figura 43 , para minimizar el peso

de este no se hace uso de un PCB, las conexiones se realizan mediante cable de cobre con cobertura de silicona. La estructura principal ensamblada se muestra en la Figura 44.

Figura 43. Evidencia de implementación del circuito electrónico

Figura 44. Ensamblaje preliminar de la estructura principal y el circuito electrónico

11.1.2. Construcción de la estructura de soporte para la carga

La estructura de soporte de la carga es la encargada de permitir un correcto posicionamiento de la carga a transportar, así como también mantener la integridad de esta, sin embargo, también se debe tener en cuenta el peso de la estructura debido a esto se decide

realizar las piezas utilizando aluminio como material. Las piezas construidas se muestran en la Figura 45 , estas se logran mediante el corte, perforación y doblado de platinas de aluminio 6061.

Figura 45. Pieza de la estructura de soporte para la carga

Para el contenedor se diseñaron y fabricaron 3 diferentes posibles soluciones presentadas en la Figura 46 siendo estas un contenedor estilo caja de MDF, un contenedor tipo cápsula de PVC y un contenedor tipo canasta fabricado de malla de Nylon, de las cuales se seleccionó la ultima debido a que tiene como ventajas un menor costo, es más ligera y es más fácil de utilizar, el volumen de esta es de 1200 cm^3 correspondiente a las dimensiones 23x13x4 cm.

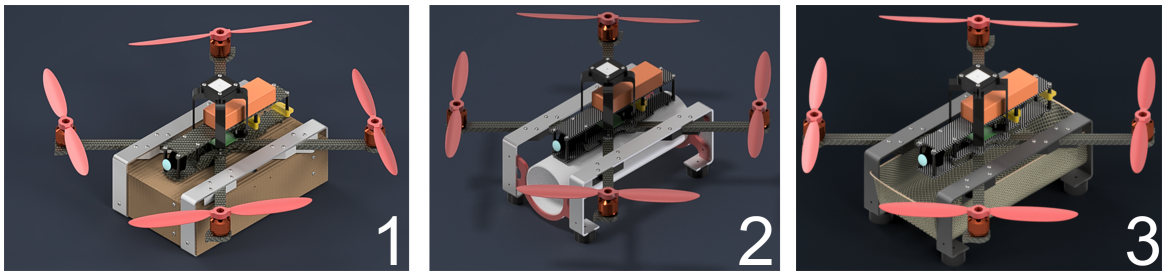


Figura 46. Alternativas contenedor de carga

Una vez se completa la manufactura de la solución seleccionada como se muestra en la Figura 47 se procede al ensamblaje final del prototipo el cual se muestra en la Figura 48.

Figura 47. Elaboración del contenedor tipo canasta

Figura 48. Prototipo final

11.2. Pruebas de funcionamiento

Se realizan pruebas de funcionamiento del dron utilizando distintas cargas, esto nos permite verificar su capacidad de vuelo, así como determinar el comportamiento del consumo de la batería en distintos escenarios, también se busca determinar la precisión del posicionamiento por GNSS y la distancia máxima de transmisión de video.

11.2.1. Prueba de carga

Se realizan vuelos con distintas cargas para determinar la maniobrabilidad del dron con distintas cargas, el escenario para las pruebas es un vuelo de baja altura (de 1 a 2 m) por un minuto en una zona con baja presencia de viento, se determina un total de 6 pruebas entre las cuales se va aumentando el peso 100 g, esto permite analizar de manera progresiva su capacidad de levantamiento, en la Tabla 44 se muestran los datos obtenidos, así como observaciones sobre la presencia de oscilaciones y sobrecalentamiento de los motores.

Tabla 44. Datos obtenidos en las pruebas de carga

Peso [g]	Vuelo	Oscilaciones	Sobrecalentamiento	Maniobrabilidad [%]
0	si	no	bajo	100
100	si	no	bajo	100
200	si	no	bajo	95
300	si	no	medio	90
400	si	si	medio	75
500	si	si	alto	60

Resultados

En la Figura 49 podemos ver que a medida que la carga aumenta, la maniobrabilidad disminuye de manera considerable, esta medida es subjetiva de cada piloto, pero tomando en cuenta la presencia de oscilaciones y sobrecalentamiento se determina que la carga óptima para los motores 2212 1000KV es de 300 g la cual permite mantener un vuelo seguro y evitar daños en el UAV, en el caso de contar con motores de la serie 26XX el problema de so-

brecalentamiento disminuiría de manera significativa. Las oscilaciones que se presentan en el vuelo en cargas mayores a la recomendada son poco significativas sin embargo podrían complicar la realización de tareas autónomas, para reducir este efecto se debe aumentar el termino derivativo de los controladores, pero debido a que este amplifica las frecuencias altas, los motores serian más propensos al sobrecalentamiento, lo cual aumenta el riesgo de daño con cargas de menor peso.

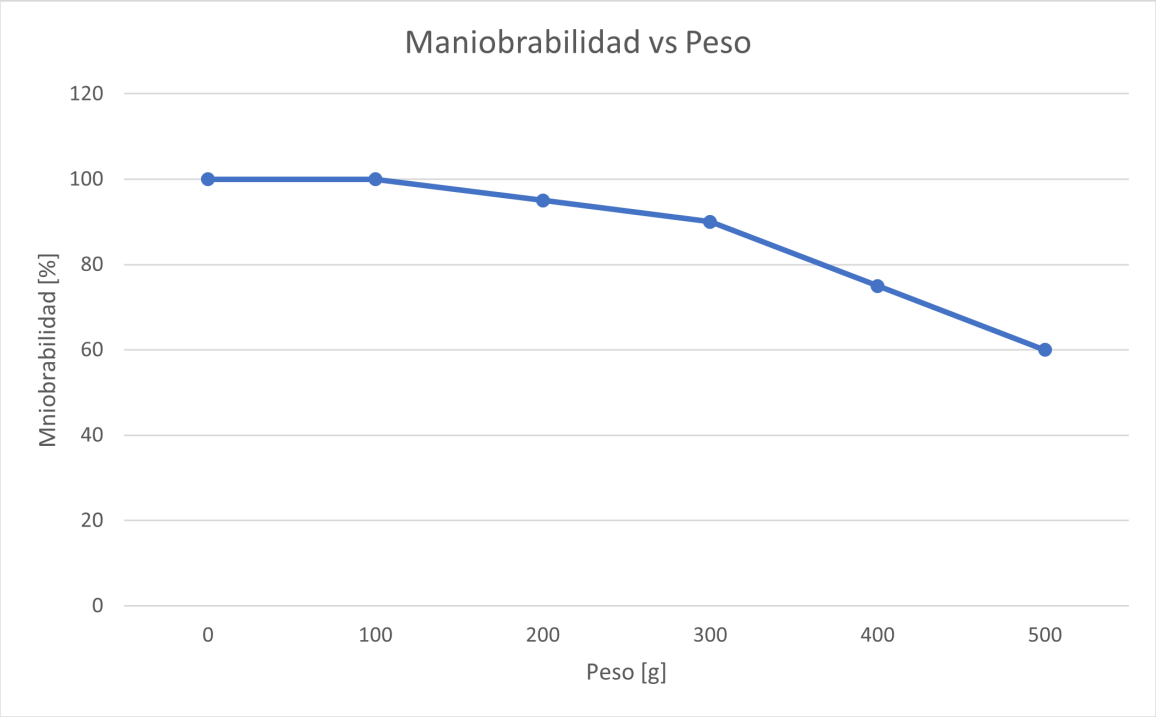


Figura 49. Maniobrabilidad vs Peso

11.2.2. Prueba de batería

Se realiza la medición de consumo de la batería del dron al ser sometido a diferentes valores de peso, las pruebas se realizan con vuelos estáticos de baja altura (de 1 a 2 m), para esto se mide el voltaje final, el voltaje inicial y el tiempo de vuelo, esto se debe a que el dron será capaz de levantar vuelo y la batería se mantendrá en buen estado siempre y cuando el voltaje se encuentre por encima de los 3.5 V cuando se llegue a ese valor se deberá iniciar acciones de aterrizaje debido a que comenzará una etapa de descarga mas rápida de la batería, si se llega a un voltaje aproximado de 3.3 V se verá un menor rendimiento de los motores y en el peor de los casos podrían llegar a detenerse. Los datos obtenidos se muestran en la Tabla 45

Tabla 45. Datos obtenidos en las pruebas de vuelo

Peso [g]	V. inicial [V]	V. final [V]	Tiempo [s]	C. de voltaje [V/s]	T. de vuelo [min]
0	3.92	3.89	35	0.00085	13.61
100	4.12	4.06	38	0.0015	7.38
200	3.89	3.78	50	0.0022	5.30
300	4.05	3.92	51	0.0025	4.57
400	4.06	3.94	43	0.0027	4.18
500	4.17	4.05	39	0.003	3.79

Resultados

Se puede ver que a medida que el peso de la carga aumenta el consumo de la batería también lo hace como se muestra en la Figura 50 , el tiempo de vuelo tiene el comportamiento contrario como se muestra en la Figura 51. Es importante notar que el escenario utilizado para las pruebas es poco demandante (sin presencia de viento y con el dron estático) por lo cual se espera un tiempo de vuelo menor en aplicaciones donde se requiere cambios de velocidad y dirección bruscos, de tal manera que los resultados obtenidos pueden ser considerados tiempos de vuelo máximos en condiciones ideales.

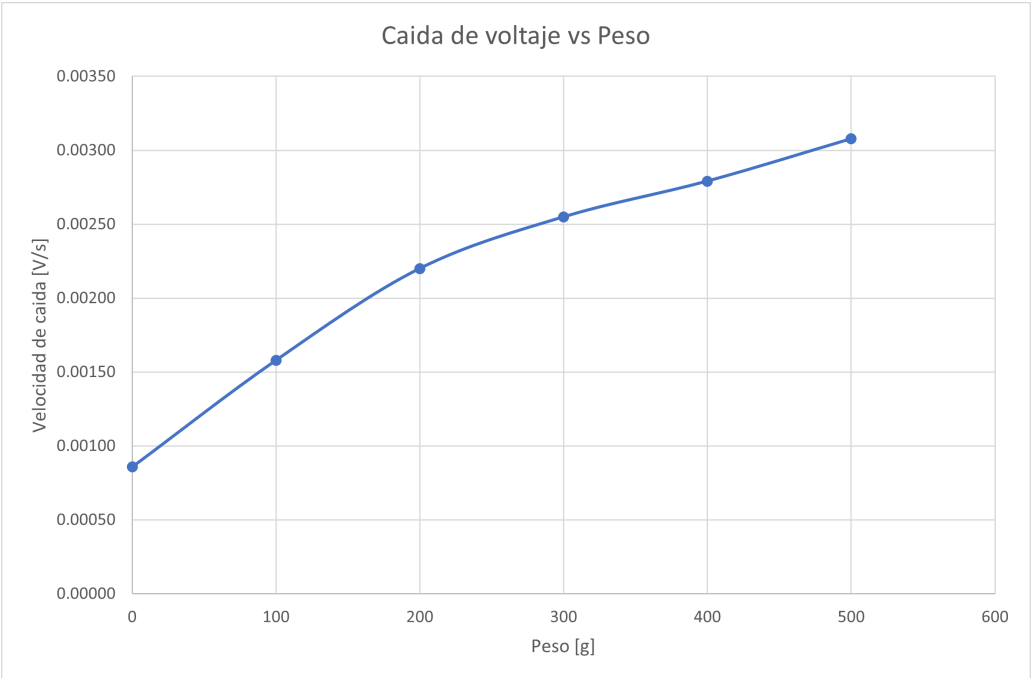


Figura 50. Consumo de batería vs Peso

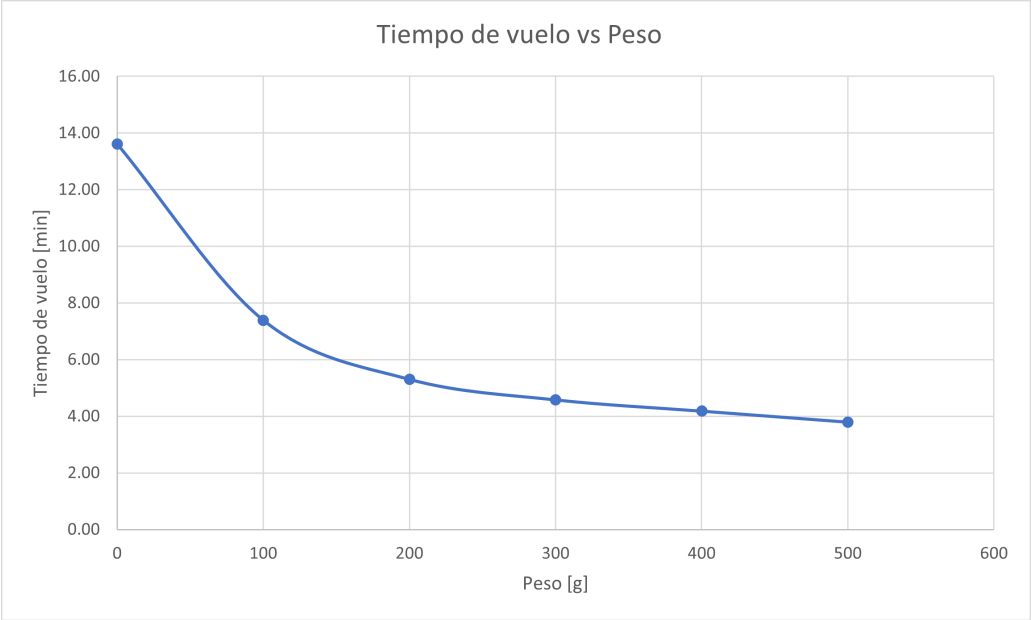


Figura 51. Tiempo de vuelo vs Peso

11.2.3. Prueba de posicionamiento

Se realizan 10 vuelos autónomos con aterrizaje para determinar el error existente en las coordenadas obtenidas mediante GNSS, todos los vuelos se realizaron con una carga de 200 g y un promedio de satélites de 13, los datos obtenidos se muestras en la Tabla 46.

Tabla 46. Datos obtenidos en las pruebas de posicionamiento

Vuelo	Error [m]
1	1.4
2	0.9
3	0.5
4	0.25
5	1.3
6	0.6
7	1.2
8	0.25
9	0.6
10	0.3
Promedio	0.73

Resultados

Con los resultados obtenidos se realiza un análisis del área necesaria para un aterrizaje seguro, como se muestra en la Figura 52 se determina que el área mínima para garantizar un aterrizaje seguro es un círculo de 2.5 m de radio el cual toma en cuenta las dimensiones del dron. Determinar esto es importante ya que es responsabilidad del operador que los nodos agregados por medio de la aplicación cuenten con este espacio mínimo para realizar tareas de manera segura.

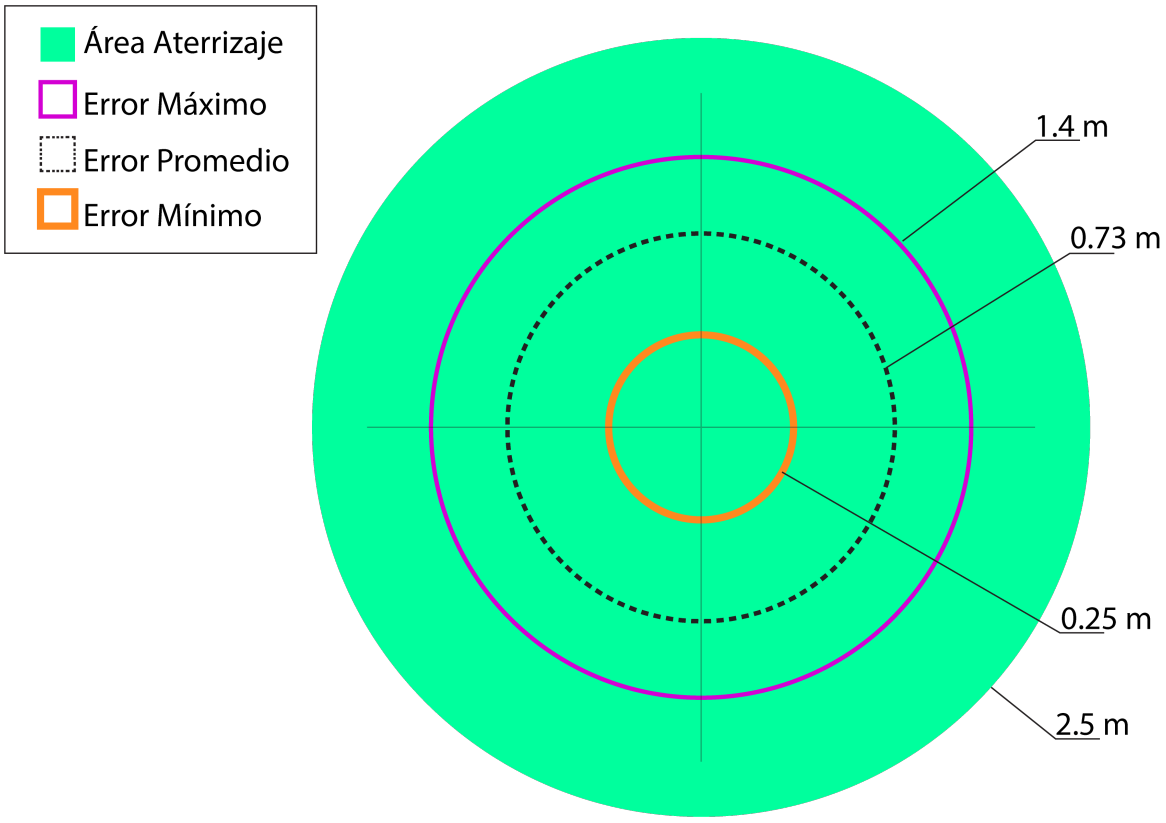


Figura 52. Área de aterrizaje y error de posicionamiento

11.2.4. Prueba de distancia

Se realiza pruebas de recepción de la señal con el objetivo de determinar la distancia máxima de vuelo, la señal a probar es la transmisión de video analógico (5.8 GHz) debido a que cuenta con una frecuencia mayor a la del control (2.4 GHz) lo que se traduce en una menor distancia, las pruebas se realizan para las 4 configuraciones de potencia que el módulo de transmisión es capaz, el terreno utilizado para la obtención de datos permite mantener línea de vista hasta 400 m, por lo que esta distancia es el limite de la prueba,

el cálculo de las distancias se realiza mediante coordenadas de GPS, estos puntos fueron registrados como el sitio en el que se detecta presencia de interferencia lo suficientemente significativa para evitar que el piloto del dron sea capaz de percibir el lugar en el que se encuentra. Los datos obtenidos se muestran en la Tabla 47.

Tabla 47. Datos obtenidos en las pruebas de distancia

Potencia [mW]	Distancia [m]
25	148
100	240
200	320
600	>400

Resultados

Los resultados obtenidos muestran un comportamiento como se muestra en la Figura 53, el comportamiento esperado debe seguir la ley del cuadrado inverso determinada por la ecuación 45, en el último dato se llega al límite del terreno sin interferencia significativa por lo que no se obtiene un número en concreto, sin embargo la presencia de ruido en la señal indica que el limite de la misma es cercano por lo que se recomienda no sobrepasar esta distancia.

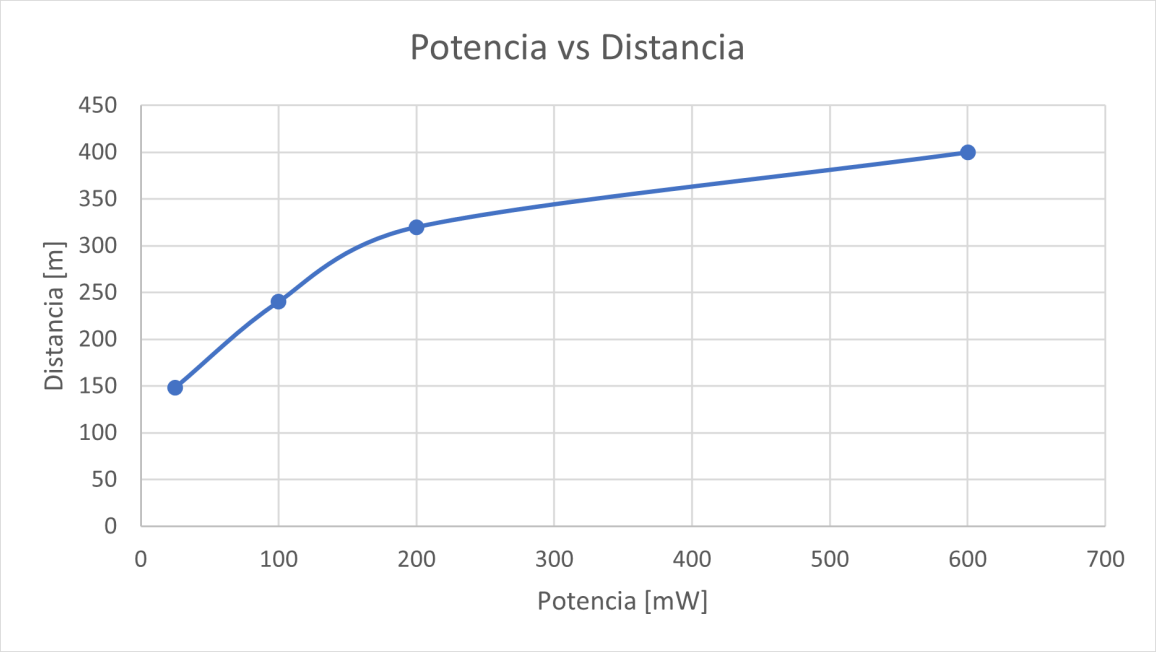


Figura 53. Distancia máxima vs potencia de transmisión de video analógico

$$I = Po/(4 \cdot \pi \cdot d^2) \tag{45}$$

Donde:

I : *Intensidad de la señal, en* dB

Po : *Potencia, en* W

d : *Distancia, en* m

12. Análisis de costos

En la Tabla 48 se detallan los precios de los materiales y servicios utilizados en la construcción del prototipo.

Tabla 48. Costo de los materiales del prototipo

Material/Servicio	Cant.	V. Unitario[USD]	V.Total[USD]	Observaciones
Microcontrolador	1	55	55	Matek F722-SE
Motor BLDC	4	12	48	2212 1000KV
ESC 30A	4	12.5	50	DSHOT
Batería	1	40	40	3S 2.2Ah
VTX	1	25	25	XILO STAX 600mW
RF RX	1	21	21	FrSky R-XSR
Módulo GNSS	1	15	15	M80 Pro
Cámara Analógica	1	30	30	Foxeer Razer Mini
Antena 5.8 GHz	1	20	20	SMA
Estructura CF	1	110	110	XL10 V5
Cable XT60	1	10	10	12AWG
Propela	4	5	20	1045
Platina de aluminio	1	12	12	
Pernos (Global)	1	12	12	
Malla de nylon	1	5	5	
Hilo de nylon	1	3	3	

Material/Servicio	Cant.	V. Unitario[USD]	V.Total[USD]	Observaciones
Control RF	1	90	90	JUMPER T12
Espiral para cable	1	5	5	16AWG
Impresión 3D (global)	1	30	30	Material
Importación (global)	1	100	100	Transporte, Impuestos
Total			701	-

Al total obtenido se le agrega el costo de trabajo de ingeniería siendo este diseño mecánico, programación del prototipo, programación del software de generación de rutas, programación de la aplicación móvil, dimensionamiento, selección e implementación del circuito electrónico y manufactura de las piezas de aluminio. Se estima a un total de 300 horas de trabajo, se toma la tarifa promedio en Ecuador para ingenieros como 5 $/h lo cual agrega al total un costo de 1500$ llevándolo a un total de 2201 $.

13. Conclusiones

- Se realiza la implementación del sistema de transporte de cargas mediante la utilización de drones el cual está constituido por un programa de computadora, una aplicación móvil y el prototipo funcional del dron de carga con dimensiones 475.8 mm x 475.8 mm x 199.2 mm y peso de 1.134 kg el cual es capaz de ejecutar tareas preprogramadas de manera autónoma.

- El programa de generación de rutas es capaz de crear tareas utilizando los datos de la base de datos de nodos en el formato correcto para ser leído y guardado en el dron.

- La aplicación móvil es capaz de acceder y modificar la base de datos de nodos mediante la utilización del sistema de posicionamiento del dispositivo de manera satisfactoria

- El prototipo es capaz de levantar vuelo con una carga con peso máximo de 500 g con riesgo de sobrecalentamiento y falla de los motores, por lo que se establece como peso máximo seguro un total de 300 g.

- El riesgo de falla o colisión del prototipo queda a responsabilidad de operador el cual debe cumplir con la normativa correspondiente para el manejo del prototipo
- Para cumplir con las directivas establecidas por el articulo 6 de la normativa establecida por la DGAC se hace uso de un sistema de transmisión de video analógico de 5.8 GHz por lo que no se debe superar la limitación de transmisión de este al momento de realizar tareas de vuelo.
- El UAV es capaz de llegar a los puntos establecidos mediante coordenadas de geolocalización con un error promedio de 0.74 m, la distancia recomendada de operación del dron a potencia de transmisión de video máxima (600 mW) es de 400 m, la altura máxima comprobada es de 7 m a una velocidad del viento promedio de 8 km/h.
- La recepción del video analógico proveniente del prototipo se puede lograr mediante el uso de un monitor genérico de 5.8 GHz que cuente con la capacidad de utilizar las frecuencias establecidas en este documento
- Para el control del dron se puede utilizar cualquier mando de radio frecuencia de 2.4 GHz compatible con el protocolo de transmisión ACCESS de la empresa FRSKY
- El sistema de ensamblaje se realiza de manera que el mantenimiento del prototipo se pueda realizar de manera sencilla en particular en piezas propensas al daño como son las propelas.
- Cualquiera de las configuraciones de UAV tipo multirotor tienen una gran desventaja en tiempo de vuelo y distancia máxima en comparación con otras alternativas, pero se encuentran mejor situadas en tareas de precisión o en terrenos de difícil acceso.

14. Recomendaciones

- Los motores utilizados se encuentran disponibles en el país sin embargo no son recomendables para aplicaciones de carga debido a ser propensos al sobrecalentamiento por lo que se recomienda el uso de motores de mayor volumen.

- Las baterías tipo LiPo requieren procesos de carga especiales y controlar los rangos de voltaje de cada una de sus celdas por lo que se debe utilizar sistema de carga especializados para evitar daños.
- No se debe bajar de 3.3 V por celda de la batería ya que se corre el riesgo de daño de la batería.
- No se debe operar el prototipo en proximidad de personas, cableado eléctrico y zonas prohibidas por la ley.
- Antes de realizar una tarea de vuelo autónomo o vuelo manual se recomienda comprobar que todos los sistemas se encuentren operativos, así como la integridad física del prototipo.
- En caso de colisión no se debe intentar realizar vuelos hasta comprobar que no existen daños en el prototipo.
- Las piezas con objetivos estéticos se deben mantener al mínimo para minimizar el peso del prototipo.

BIBLIOGRAFÍA

[1] L. Pratt. Mq 9 reaper on patrol. [En línea]. Disponible: https://www.afrc.af.mil/News/Photos/igphoto/2000608244 (2022)

[2] E. Chamorro. Fabdrone. [En línea]. Disponible: https://eduardochamorro.github.io/beansreels/workshops/modulardrone/images/types/1.jpg (2020)

[3] 40a esc brushless speed controller. [En línea]. Disponible: https://www.microjpm.com/products/ad36819/ (2021)

[4] Amazon prime air. [En línea]. Disponible: https://www.aboutamazon.com/news/transportation/amazon-prime-air-prepares-for-drone-deliveries (2021)

[5] Valkyrie heavy pro. [En línea]. Disponible: https://www.valkyrie.pro/ (2019)

[6] Drone delivery canada. [En línea]. Disponible: https://dronedeliverycanada.com/technology/ (2022)

[7] A2z drone delivery. [En línea]. Disponible: https://www.a2zdronedelivery.com/rdsx-overview (2021)

[8] R. Balart, N. Muñoz, L. Quiles, O. Fenollar, y J. Martínez, “Materiales avanzados en ingeniería derivados de laminados compuestos,” 2021. [En línea]. Disponible: https://learning.edx.org/course/course-v1:UPValenciaX+INGLAM201x+1T2021/home

[9] R. G. Budynas, *Fundamentos de diseño mecánico de Shigley*. McGraw-Hill, 2008.

[10] Ángulo de ataque. [En línea]. Disponible: www.grc.nasa.gov/www/k-12/airplane/incline.html [Fecha de consulta: Abril 2022]

[11] Xilo official website. [En línea]. Disponible: http://www.xilodrone.com/ (2021)

[12] [En línea]. Disponible: https://shop.iflight-rc.com/xl10-v5-10-inch-fpv-frame-pro1504

[13] Perfileria de aluminio cedal. [En línea]. Disponible: http://www.cedal.com.ec/index.php/es/categorias-de-productos/perfileria-de-aluminio/normalizados-estandarizados.html (2019)

[14] L. Brooke-Holland, “Unmanned aerial vehicles (drones): an introduction,” *House of Commons Library, UK*, 2012.

[15] I. A. Carrillo Herrera, “Diseño y construcción de un quadcopter controlado por radio frecuencia,” *UIDE*, p. 147, 2013.

[16] Fibra de carbono t300 data sheet. [En línea]. Disponible: https://www.rockwestcomposites.com/media/wysiwyg/T300DataSheet_1.pdf (2022)

[17] Common dshot escs. [En línea]. Disponible: https://ardupilot.org/copter/docs/common-dshot-escs.html (2021)

[18] *VTX XILO STAX USER MANUAL*, XILO. [En línea]. Disponible: https://getfpv-media.s3.amazonaws.com/wysiwyg/files/XILO-STAX-VTX-User-Manual-compressed.pdf

[19] DGAC, “Resolución Nro . DGAC-YA-2017-0104-R,” pp. 1–2, 2017.

[20] M. Hassanalian y A. Abdelkefi, “Classifications, applications, and design challenges of drones: A review,” *Progress in Aerospace Sciences*, vol. 91, no. November 2016, pp. 99–131, 2017.

[21] P. Ruipérez, “Diseño y fabricación de un Dron mediante impresión 3D,” p. 119, 2015. [En línea]. Disponible: https://riunet.upv.es/bitstream/handle/10251/73170/RUIPÉREZ - Diseño y fabricación de un dron mediante impresión 3D.pdf?sequence=5

[22] CEN/TC250, “En 1999-1-1 : 2007,” p. 53, 2007. [En línea]. Disponible: https://www.phd.eng.br/wp-content/uploads/2014/11/en.1999.1.1.2007.pdf

[23] Cedal, *Catalogo de platinas de aluminio CEDAL*, 1st ed., Corporación Ecuatoriana de Aluminio S.A, Ecuador, Diciembre 2021.

[24] F. P. Beer, J. Johnston, y J. T. DeWolf, *Mecanica de materiales*, 7th ed. McGraw-Hill Interamericana, 2017.

[25] I. M. Daniel y O. Ishai, *Engineering mechanics of composite materials*, 2nd ed. Oxford University Press, 2006.

[26] Siemens, *NX Nastran User's Guide*, 1st ed., Siemens, 2013. [En línea]. Disponible: http://www2.me.rochester.edu/courses/ME204/nx_help/index.html#uid:id503311

[27] Nasa. The lift equation. [En línea]. Disponible: https://www.grc.nasa.gov/www/k-12/airplane/lifteq.html [Fecha de consulta: Abril 2022]

[28] F. E. Co., "Frsky 2.4ghz access r-xsr manual," 2020. [En línea]. Disponible: https://www.frsky-rc.com/wp-content/uploads/Downloads/Manual/R-XSR/R-XSR-ACCESS%20-Manual.pdf

[29] S. Malys, R. Wong, y S. True, "The wgs 84 terrestrial reference frame in 2016," 2016. [En línea]. Disponible: https://www.unoosa.org/pdf/icg/2016/icg11/wgd/02wgd.pdf

[30] A. Visioli, *Practical PID Control*, 1st ed. Springer-Verlag, 2006.

[31] N. R. Chopde y M. Nichat, "Landmark based shortest path detection by using a* and haversine formula," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 1, no. 2, pp. 298–302, 2013.

[32] T. Vincenty, "Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations," *Survey Review*, vol. 23, no. 176, pp. 88–93, 1975.

Anexo A: Normativa para la operación de drones en Ecuador

la DGAC (Dirección general de aviación civil) es la entidad encargada del manejo de las actividades aeronáuticas del país por lo que se cuenta con resoluciones para el manejo y regularización de las UAVs, en el Ecuador no se cuenta con una normativa estricta sobre el uso de esta tecnología en la mayoría de los escenarios, según [19] la resolución 251/2015 dictada por la DGAC en el año 2015 se encuentra vigente, esta resolución cuenta con diez artículos que establecen lo siguiente:

Art. 1.- Operaciones en las cercanías de un aeródromo

Se prohíbe la operación de las UAVs en espacios aéreos controlados. La operación de las UAVs se mantendrá durante toda la duración del vuelo, a una distancia igual o mayor a 9 kilómetros de las proximidades de cualquier aeródromo o base aérea militar.

Art. 2.- Altura máxima de vuelo

La operación de las UAVs no excederá en ningún momento una altura de vuelo de 400 pies (122 metros) sobre el terreno.

Art. 3.- Horas de operación

Las UAVs serán operadas solamente en las horas comprendidas entre la salida y la puesta del sol; y en condiciones meteorológicas de vuelo visual, libre de nubes, neblina, precipitación o cualquier otra condición que obstruya o pueda obstruir el contacto visual permanente con la UAVs.

Art. 4.- Responsabilidad por la operación

La persona que opera los controles de las UAVs será responsable por la operación general de la misma durante todo el vuelo, en forma solidaria con el explotador o propietario de la aeronave.

Art. 5.- Integridad fisiológica del operador de una UAVs

Ninguna persona operará los controles de una UAVs si:

- Se encuentra fatigado, o si considera que pudiera sufrir los efectos de la fatiga durante la operación;
- Se encuentra bajo el efecto del consumo de bebidas alcohólicas, o de cualquier droga que pudiera afectar sus facultades para operar los controles de manera segura.

Art. 6.- Funciones de automatización

Si las UAVs tienen la capacidad de realizar vuelo automático, esta función podrá ser utilizada solamente si le permite al operador de los controles intervenir en cualquier momento para tomar el control inmediato de la aeronave.

Art. 7.- Limitaciones

La persona que opera los controles de una UAVs es responsable por asegurarse que la misma sea operada de acuerdo con las limitaciones operacionales establecidas por el fabricante.

Art. 8.- Seguros

El propietario o explotador de las UAVs están en la obligación de responder por los daños causados a terceros, como resultado de sus actividades de vuelo, para lo cual debe contratar la póliza de seguros de responsabilidad civil legal a terceros en los montos mínimos establecidos en la Tabla 49:

Tabla 49. Valor de la póliza de seguros según peso del UAV, Fuente: [19]

Peso	Valor
De 02 a 25 kg	USD 3.000,00
más de 25 kg	USD 5.000,00

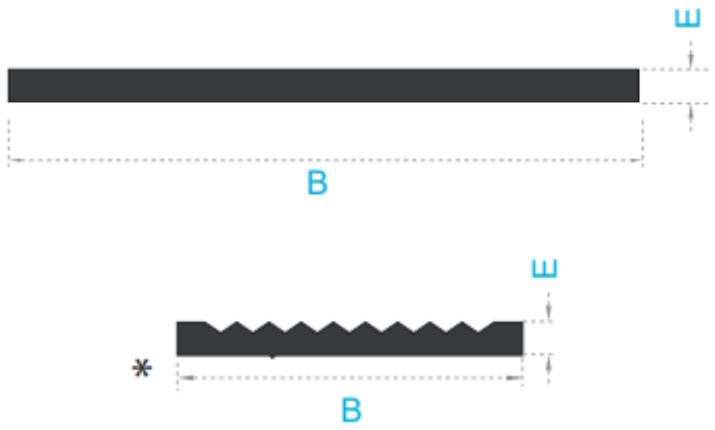
Art. 9.- Cumplimiento con las leyes y reglamentos locales

El cumplimiento de estas disposiciones, no exime al operador de las UAVs de cumplir con las leyes y reglamentos locales aplicables.

Art. 10.- Consideración final

Cualquier aspecto no considerado en la presente resolución, será analizado y resuelto por la Autoridad Aeronáutica Civil.

Anexo B: Catálogo de pletinas CEDAL



REFERENCIA	Lado (B)		Espesor (E)	Peso
	Milímetros	Pulgadas	mm	Kg/m
1002	38.10	1 1/2	3.20	0.331
*1267	25.40	1	2.40	0.142
1423	70.00	2 3/4	1.20	0.228
1767	25.40	1	2.90	0.199
1982	37.70	1 31/64	2.40	0.245

Figura 54. Catálogo de platinas CEDAL, Fuente: [13]

Anexo C: Teoría clásica de laminados (CLT)

La teoría clásica de laminados permite generar modelos matemáticos de los distintos materiales compuestos dentro de esta categoría, esto tiene el objetivo de analizar el comportamiento del mismo utilizando ecuaciones para determinar sus propiedades mecánicas y simulaciones para obtener con mayor detalle el comportamiento de la pieza de interés (tomando en cuenta su geometría). El modelo que rige el comportamiento de laminados compuestos requiere las siguientes variables:

- Geometría del laminado
- N° total de láminas/capas
- Espesor de cada capa
- Materiales de cada capa
- Matriz (E_m, G_m, υ_m)
- Refuerzo (E_f, G_f, υ_f)
- Cantidad de fibra (V_f)
- Ángulo de las fibras (Θ)

Los materiales laminados buscan conseguir un comportamiento isotrópico, pero esto debe ser comprobado matemáticamente mediante el estudio de las propiedades mecánicas del laminado, para obtenerlas se deben seguir tres pasos generales:

1. Modelamiento y análisis matemático de una sola lamina en las direcciones locales 1 – 2 (diagonales).
2. Modelamiento y análisis de una sola lamina en las direcciones locales X - Y.
3. Modelamiento y análisis matemático del material compuesto.

Modelo matemático

Según [8], el modelo matemático para materiales compuestos se realiza tomando en cuenta el diagrama de cuerpo libre presentado en la Figura 55 y la siguiente nomenclatura:

$$h_i \rightarrow \textit{altura de la lamina}$$

$$e_i = h_i - h_{(i-1)} \rightarrow \textit{espesor de la lamina}$$

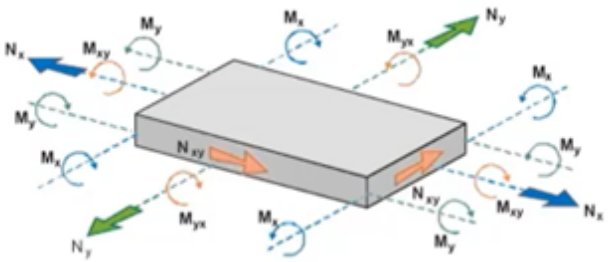


Figura 55. Diagrama de cuerpo libre para el análisis de un laminado, Fuente: [8]

Las características mecánicas de la lámina se dividen en matriz y refuerzo, se debe contar con los datos de módulo de Young o de elasticidad (E_m para la matriz, E_f para la fibra de refuerzo), módulo de cizalladura o corte (G_m para la matriz, G_f para la fibra de refuerzo) y Coeficiente de Poisson (υ_m para la matriz, υ_f para la fibra de refuerzo)

Análisis de esfuerzos y deformaciones

En la Figura 56 se muestra un diagrama de cuerpo libre para el análisis de los esfuerzos en el eje x.

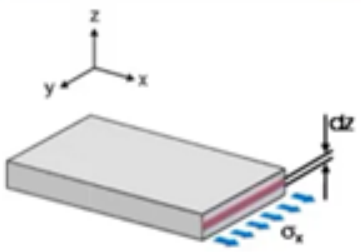


Figura 56. Diagrama de cuerpo libre para el análisis de esfuerzos en un laminado en el eje x, Fuente: [8]

d_z : *elemento diferencial en z*

σ_x : *esfuerzo en eje x*

fuerza total en x está definida como (46)

$$N_x = (\sigma_x * d_z) \tag{46}$$

Matriz de fuerzas:

Esta definida por (47)

$$[N] = ([\sigma] * d_z) \tag{47}$$

Donde:

$[N] \rightarrow$ *matriz de fuerzas*

$[\sigma] \rightarrow$ *matriz de esfuerzos*

Matriz de momentos:

Esta definida por la (48)

$$[M] = ([\sigma] * z * d_z) \tag{48}$$

Deformación a una altura Z (Componentes planar y de curvatura)

$\varepsilon\,(Z) = \varepsilon + z * \ c\varepsilon \rightarrow$ *deformación en el plano*

$c \rightarrow$ *deformación de curvatura*

Esfuerzo a una altura Z

El esfuerzo en Z esta definida por la ecuación (49)

$$\sigma\,(z) = \left[\bar{Q}\right] * [\varepsilon] + \left[\bar{Q}\right] * z * [c] \tag{49}$$

Donde:

$$\left[\bar{Q}\right] \rightarrow \textit{matriz de rigidez global}$$

$$[Q] \rightarrow \textit{matriz de rigidez local}$$

$$[T_\varepsilon] \rightarrow \textit{matriz de transformación de deformaciones}$$

$$[T_\sigma] \rightarrow \textit{matriz de transformación de esfuerzos}$$

La tensión a altura Z también puede ser definida como la multiplicación entre rigidez y deformación como se muestra en la (50).

$$\left[\bar{Q}\right] = [T_\varepsilon] * [Q] * [T_\sigma]^{-1} \tag{50}$$

Expresiones fuerza-deformación

$$[N]_{Totales} = \sum_{i=1}^{i_T} [N]_i = \sum_{i=1}^{i_T} \int_{h_{i-1}}^{h_i} [\sigma]_i * dz$$

$$= \sum_{i=1}^{i_T} \left\{ \left[\bar{Q}\right] * [\varepsilon] \int_{h_{i-1}}^{h_i} dz + \left[\bar{Q}\right] * [c] * \int_{h_{i-1}}^{h_i} z * dz \right\}_i$$

$$[N]_{Totales} = \sum_{i=1}^{i_T} \left\{ \left[\bar{Q}\right] * [\varepsilon] * (h_i - h_{i-1}) + \left[\bar{Q}\right] * [c] * \frac{1}{2} * \left(h_i^2 - h_{i-1}^2\right) \right\}_i$$

$$[A] = \sum_{i=1}^{i_T} \left[\bar{Q}\right]_i * (h_i - h_{i-1})$$

$$[B] = \frac{1}{2} * \sum_{i=1}^{i_T} \left[\bar{Q}\right]_i * \left(h_i^2 - h_{i-1}^2\right)$$

$$N_T = A\varepsilon + B[c] \textit{ (relaciona fuerzas y deformaciones)}$$

Matriz extensional [A]

Interviene en las deformaciones en el plano o extensionales, $[\varepsilon]$. Es la responsable de la rigidez en el plano.

Matriz de acoplamiento [B]

Sí $[B] \neq 0$ entonces $[N]_T$ es la única causante de deformaciones de curvatura que general-

mente se relacionan con la aplicación de momentos sobre el material, es decir, la matriz B acopla deformaciones extensionales (en el plano) y de curvatura.

En la expresión de momento-deformación, la matriz $[B] \neq 0$ da como resultado la aparición de deformaciones extensionales causadas únicamente por $[M]_T$

Expresiones de momento-deformación

$$[M]_T = \sum_{i=1}^{i_T} [M]_i = \sum_{i=1}^{i_T} \int [\sigma] * z * d_z = \sum_{i=1}^{i_T} ([\bar{Q}] * [] * z + [\bar{Q}] * z^2 * [c]) d_z)$$

$$[M]_T = \sum_{i=1}^{i_T} \left\{ \left[\bar{Q}\right] * [\varepsilon] \int_{h_{i-1}}^{h_i} z * dz + \left[\bar{Q}\right] * [c] * \int_{h_{i-1}}^{h_i} z^2 * dz \right\}_i$$

$$[N]_{Totales} = \sum_{i=1}^{i_T} \left\{ \left[\bar{Q}\right] * [\varepsilon] * \frac{1}{2} * \left(h_i^2 - h_{i-1}^2\right) + \left[\bar{Q}\right] * [c] * \frac{1}{3} * \left(h_i^3 - h_{i-1}^3\right) \right\}_i$$

$$[B] = \frac{1}{2} * \sum_{i=1}^{i_T} \left[\bar{Q}\right]_i * \left(h_i^2 - h_{i-1}^2\right)$$

$$[D] = \frac{1}{3} * \sum_{i=1}^{i_T} \left[\bar{Q}\right]_i * \left(h_i^3 - h_{i-1}^3\right)$$

$M_T = B\varepsilon + D[c]$ *(relación entre momentos y deformaciones)*

Matriz de flexión o de curvatura [D]

Interviene en las deformaciones de curvatura o de flexión, [c]. Es la responsable de la rigidez a flexión del laminado compuesto

Expresiones Generales

Paramétricas

$$[N]_T = [A][\varepsilon] + [B][c]$$

$$[M]_T = [B][\varepsilon] + [D][c]$$

$$[A] = \sum_{i=1}^{i_T} \left[\bar{Q}\right]_i * (h_i - h_{i-1})$$

$$[B] = \frac{1}{2} * \sum_{i=1}^{i_T} \left[\bar{Q}\right]_i * \left(h_i^2 - h_{i-1}^2\right)$$

$$[D] = \frac{1}{3} * \sum_{i=1}^{i_T} \left[\bar{Q}\right]_i * \left(h_i^3 - h_{i-1}^3\right)$$

Matriciales

$$\begin{bmatrix} N \\ M \end{bmatrix} = \begin{bmatrix} A & B \\ B & D \end{bmatrix} * \begin{bmatrix} \varepsilon \\ c \end{bmatrix}$$

$$\begin{bmatrix} \varepsilon \\ c \end{bmatrix} = \begin{bmatrix} A & B \\ B & D \end{bmatrix}^{-1} * \begin{bmatrix} N \\ M \end{bmatrix}$$

$$\begin{bmatrix} \varepsilon \\ c \end{bmatrix} = \begin{bmatrix} a & b \\ \beta & d \end{bmatrix} * \begin{bmatrix} N \\ M \end{bmatrix}$$

$[a]$: *matriz de flexibilidad global*

Si $[M] = 0 \rightarrow [\varepsilon] = a * [N] \rightarrow [N] = h_T * [\sigma] \rightarrow [\varepsilon] = a * h_T * [\sigma]$

Características mecánicas elásticas de laminado compuesto

Relaciones esfuerzo deformación $\sigma - \varepsilon$ (direcciones locales 1-2):

$$[\varepsilon_{12}] = [S] * [\sigma_{12}]$$

Matriz de flexibilidad local [S]:

$$[S] = \begin{bmatrix} \frac{1}{E_1} & -\frac{\nu_{12}}{E_2} & 0 \\ -\frac{\nu_{12}}{E_2} & \frac{1}{E_2} & 0 \\ 0 & 0 & \frac{1}{G_{12}} \end{bmatrix}$$

Relaciones esfuerzo deformación $\sigma - \varepsilon$ (Laminado):

$$[\varepsilon] = a * h_T * [\sigma]$$

Matriz de flexibilidad global [a]:

$$a * \ h_T = \begin{bmatrix} \frac{1}{E_x} & -\frac{\nu_{xy}}{E_y} & 0 \\ -\frac{\nu_{xy}}{E_y} & \frac{1}{E_y} & 0 \\ 0 & 0 & \frac{1}{G_{xy}} \end{bmatrix}$$

$$[a] = \begin{bmatrix} a_{11} & a_{12} & .. \\ a_{21} & a_{22} & .. \\ .. & .. & a_{33} \end{bmatrix}$$

Características mecánicas del laminado

Las siguientes ecuaciones describen cada una de las propiedades mecánicas del laminado a partir de la matriz de flexibilidad global.

Módulo de tracción en x:

$$E_x = \frac{1}{a_{11} * h_T}$$

Módulo de tracción en y:

$$E_y = \frac{1}{a_{22} * h_T}$$

Módulo de cortadura xy:

$$G_{xy} = \frac{1}{a_{33} * h_T}$$

Módulo de poisson xy:

$$\nu_{xy} = -E_x * \ a_{21} * \ h_T$$

Módulo de poisson yx:

$$\nu_{xy} = -E_y * \ a_{12} * \ h_T$$

Anexo D: Código de programación

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System;

public class Base : IComparable<Base>
{
   public int id;
   public string name;
   public float lat, lon;
   public string date;
   public Base(int i, string n, float la, float lo, string d)
   {
      id=i;
      name = n;
      lat=la;
      lon=lo;
      date = d;
   }
   public int CompareTo(Base other)
   {
      if (other == null)
      {
         return 1;
      }

      //Return the difference in power.
      return id -  other.id;
   }
```

```
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Networking;
using UnityEngine.UI;

public class BaseDelete : MonoBehaviour
{
    string URL = "https://location-db.000webhostapp.com/removeBASE.php";
    int id=0;
    bool up=false;
    [SerializeField] Button Removebtn;
    [SerializeField] BaseManager bm;
    // Start is called before the first frame update
    void Start()
    {

    }
```

```
public void RemoveBase()
{
    if (!up)
    {
        up = true;
        Removebtn.enabled=false;
        StartCoroutine(Remove());
    }

}
IEnumerator Remove()
{
    id = bm.baseId;
    WWWForm form = new WWWForm();
    form.AddField("fieldF", "id");
    form.AddField("conditionF", id.ToString());
    using (UnityWebRequest www = UnityWebRequest.Post(URL, form))
    {
        yield return www.SendWebRequest();

        if (www.result != UnityWebRequest.Result.Success)
        {
            Debug.Log(www.error);
        }
        else
        {
            Debug.Log("Form upload complete!");
            Debug.Log(www.downloadHandler.text);
        }
```

```
        }
        up = false;
        Removebtn.enabled = true;
    }

    // Update is called once per frame
    void Update()
    {

    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Networking;
using UnityEngine.UI;
using TMPro;

public class BaseInsert : MonoBehaviour
{
    string URL= "https://location-db.000webhostapp.com/insertBASE.php";
    string baseName;
    float  latitud, longitud;
    bool up = false;
    [SerializeField] private locationSaver ls;
    Vector2 loc;
    [SerializeField] TMP_InputField nombre;
    [SerializeField] BasesGetData bData;
```

```
[SerializeField] Button Add;
void Start()
{

}
public void AddBase()
{
   baseName = nombre.text;
   loc = ls.GetLocation();
   latitud = loc.x;
   longitud = loc.y;
   if (baseName != null && baseName != "" && baseName != " ")
   {
      if (name(baseName))
      {
         if (!up)
         {
            up= true;
            Add.enabled = false;
            StartCoroutine(UploadData());
         }
         else
         {
            Debug.Log("Error: Subida en proceso");
         }
      }
      else
      {
         Debug.Log("Error: Nombre Repetido");
```

```
        }
    } else
    {
        Debug.Log("Error: Nombre Vac�o");
    }

}

bool name(string nme)
{
    List<Base> nBases = bData.Bases;

    foreach (Base basex in nBases)
    {
        if(basex.name == nme)
            return false;

    }
    return true;
}

IEnumerator UploadData()
{
    WWWForm form = new WWWForm();
    form.AddField("addName", baseName.Replace(" ","_"));
    form.AddField("addLatitude", latitud.ToString("F7"));
    form.AddField("addLongitude", longitud.ToString("F6"));
    using (UnityWebRequest www = UnityWebRequest.Post(URL, form))
    {
```

```
            yield return www.SendWebRequest();


            if (www.result != UnityWebRequest.Result.Success)
            {
                Debug.Log(www.error);
            }
            else
            {
                Debug.Log("Form upload complete!");
                Debug.Log(www.downloadHandler.text);
            }
        }
        up=false;
        Add.enabled = true;
    }



    // Update is called once per frame
    void Update()
    {


    }
}

using UnityEngine;
using Mapbox.Utils;
using Mapbox.Unity.Map;
using Mapbox.Unity.MeshGeneration.Factories;
using Mapbox.Unity.Utilities;
```

```
using System.Collections.Generic;

public class SpawnBases : MonoBehaviour
{
    [SerializeField]
    AbstractMap _map;

    [SerializeField]
    Vector2d[] _locations;
    [SerializeField] BasesGetData data;

    [SerializeField]
    float _spawnScale = 100f;

    [SerializeField]
    GameObject _markerPrefab;

    List<GameObject> _spawnedObjects;

    bool s= false;
    private void OnEnable()
    {
        BasesGetData.OnValueChange += Spawn;

    }
    private void OnDisable()
    {
        BasesGetData.OnValueChange -= Spawn;
```

```
}

void Spawn()
{
    List<Base> nBases = data.Bases;
    _locations = new Vector2d[nBases.Count];
    _spawnedObjects = new List<GameObject>();
    int i = 0;
    //while (!data.listo) ;
    foreach (Base basex in nBases)
    {
        _locations[i] = new Vector2d(basex.lat, basex.lon);
        var instance = Instantiate(_markerPrefab);
        instance.transform.localPosition = _map.GeoToWorldPosition(_locations[i], true);
        instance.transform.localScale = new Vector3(_spawnScale, _spawnScale, _spawnScale);
        instance.name = basex.name;
        _spawnedObjects.Add(instance);
        i++;
    }
    s=true;
}
private void Update()
{
    if (s)
    {
        int count = _spawnedObjects.Count;
        for (int i = 0; i < count; i++)
        {
            var spawnedObject = _spawnedObjects[i];
```

```
            var location = _locations[i];
            spawnedObject.transform.localPosition = _map.GeoToWorldPosition(location, true);
            spawnedObject.transform.localScale = new Vector3(_spawnScale, _spawnScale, _spawnScale);
        }
      }
   }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine.Networking;
using UnityEngine;

public class BasesGetData: MonoBehaviour
{
   string URL= "https://location-db.000webhostapp.com/baseData.php";
    string [] baseData;
   [SerializeField] public List<Base> Bases = new List<Base>();
   public delegate void ValueAction();
   public static event ValueAction OnValueChange;
   public bool listo = false;
   // Start is called before the first frame update
   void Start()
   {
      StartCoroutine(GetRequest(URL));
   }
   public void Reload()
   {
      StartCoroutine(GetRequest(URL));
```

```
}
IEnumerator GetRequest(string uri)
{
    listo = false;
    using (UnityWebRequest webRequest = UnityWebRequest.Get(uri))
    {
        // Request and wait for the desired page.
        yield return webRequest.SendWebRequest();

        string[] pages = uri.Split('/');
        int page = pages.Length - 1;

        switch (webRequest.result)
        {
            case UnityWebRequest.Result.ConnectionError:
            case UnityWebRequest.Result.DataProcessingError:
                Debug.LogError(pages[page] + ": Error: " + webRequest.error);
                break;
            case UnityWebRequest.Result.ProtocolError:
                Debug.LogError(pages[page] + ": HTTP Error: " + webRequest.error);
                break;
            case UnityWebRequest.Result.Success:
                //Debug.Log(pages[page] + ":\nReceived: " + webRequest.downloadHandler.text);
                string basesDataString = webRequest.downloadHandler.text;
                baseData = basesDataString.Split(';');
                Bases.Clear();
                for (int i = 0; i < baseData.Length - 1; i++)
                {
```

```
                print(separateData(baseData[i], "id:") + " " + separateData(baseData[i], "name:") + " " +
separateData(baseData[i], "lat:") + " " + separateData(baseData[i], "lon:") + " " +
separateData(baseData[i], "date:"));

                Bases.Add(new Base(int.Parse(separateData(baseData[i], "id:")), separateData(baseData[i],
"name:"),float.Parse(separateData(baseData[i], "lat:")), float.Parse(separateData(baseData[i], "lon:")),
separateData(baseData[i], "date:")));

            }

            listo = true;

            if (OnValueChange != null) OnValueChange();

            break;

        }

    }

  }

  string separateData(string row, string field)

  {

    string value = row.Substring(row.IndexOf(field) + field.Length);

    if (value.Contains("|"))

    {

      value = value.Remove(value.IndexOf("|"));

    }

    return value;

  }

  // Update is called once per frame

  void Update()

  {


  }

}


using System.Collections;
```

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class FillList : MonoBehaviour
{
    [SerializeField] BasesGetData data;
    public Text container;
    bool r = false;
    // Start is called before the first frame update
    void Start()
    {

    }
    void Awake()
    {

    }
    // Update is called once per frame
    void Update()
    {
        if (!r && data.listo)
        {
            List<Base> nBases = data.Bases;
            string Lista = "";
            foreach (Base basex in nBases)
            {
                float l = float.Parse(basex.lat.ToString("F7"));
                float m = float.Parse(basex.lon.ToString("F6"));
```

```
            Lista = Lista + " \n " + basex.name + ": \n" + "     " + l.ToString("F7") + " " + m.ToString("F6") +
"\n";
        }
        container.text = Lista;
        r = true;
      }
   }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using Mapbox.Unity.Map;

public class locationSaver : MonoBehaviour
{
   public Vector2 location;
   public Text locationText;
   [SerializeField] AbstractMap map;
   // Start is called before the first frame update
   void Start()
   {

   }

   // Update is called once per frame
   void Update()
   {
```

```
    }
    public Vector2 GetLocation()
    {
        return location;
    }
    public void UpdateLocation(Vector2 nueva)
    {
        //location=new Vector2(nueva.x+0.000001f, nueva.y + 0.000001f);
        location = nueva;
        locationText.text= location.ToString("F7");
        map.UpdateMap(new Mapbox.Utils.Vector2d(location.x, location.y), 14.5f);
    }
}

using UnityEngine;
using System.Collections;
using Mapbox.Unity.Location;
using Mapbox.Unity.Utilities;
using Mapbox.Unity.Map;

public class LocationService : MonoBehaviour
{
    [SerializeField] private locationSaver ls;
    bool isLoc=false;
    public BaseManager bm;

    IEnumerator Start()
    {
```

```
// Check if the user has location service enabled.
if (!Input.location.isEnabledByUser)
    yield break;

// Starts the location service.
Input.location.Start();

// Waits until the location service initializes
int maxWait = 20;
while (Input.location.status == LocationServiceStatus.Initializing && maxWait > 0)
{
    yield return new WaitForSeconds(1);
    maxWait--;
}

// If the service didn't initialize in 20 seconds this cancels location service use.
if (maxWait < 1)
{
    Debug.Log("Timed out");
    yield break;
}

// If the connection failed this cancels location service use.
if (Input.location.status == LocationServiceStatus.Failed)
{
    Debug.Log("Unable to determine device location");
    yield break;
}
else
```

```
    {
        // If the connection succeeded, this retrieves the device's current location and displays it in the Console window.
        Debug.Log("Location: " + Input.location.lastData.latitude + " " + Input.location.lastData.longitude + " " + Input.location.lastData.altitude + " " + Input.location.lastData.horizontalAccuracy + " " + Input.location.lastData.timestamp);
        ls.UpdateLocation(new Vector2(Input.location.lastData.latitude, Input.location.lastData.longitude));

    }

    // Stops the location service if there is no need to query location updates continuously.
    Input.location.Stop();

  }

  void LateUpdate()
  {
    if (!isLoc) {
      isLoc = true;
      StartCoroutine(GetLocation());
      bm.detectBase();

    }
  }

  IEnumerator GetLocation()
  {
    // Check if the user has location service enabled.
    if (!Input.location.isEnabledByUser)
```

```
    yield break;

// Starts the location service.
Input.location.Start();

// Waits until the location service initializes
int maxWait = 20;
while (Input.location.status == LocationServiceStatus.Initializing && maxWait > 0)
{
    yield return new WaitForSeconds(1);
    maxWait--;
}

// If the service didn't initialize in 20 seconds this cancels location service use.
if (maxWait < 1)
{
    Debug.Log("Timed out");
    yield break;
}

// If the connection failed this cancels location service use.
if (Input.location.status == LocationServiceStatus.Failed)
{
    Debug.Log("Unable to determine device location");
    yield break;
}
else
{
```

```
        // If the connection succeeded, this retrieves the device's current location and displays it in the Console window.
        Debug.Log("Location: " + Input.location.lastData.latitude + " " + Input.location.lastData.longitude + " " + Input.location.lastData.altitude + " " + Input.location.lastData.horizontalAccuracy + " " + Input.location.lastData.timestamp);
        ls.UpdateLocation(new Vector2(Input.location.lastData.latitude, Input.location.lastData.longitude));

      }

      // Stops the location service if there is no need to query location updates continuously.
       Input.location.Stop();

      isLoc = false;
   }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class SceneM : MonoBehaviour
{
   public void Lista()
   {
      SceneManager.LoadScene("LIST");
   }
   public void Return()
   {
```

```
        SceneManager.LoadScene("MAIN");
    }
}

using UnityEngine;
using Mapbox.Utils;
using Mapbox.Unity.Map;
using Mapbox.Unity.MeshGeneration.Factories;
using Mapbox.Unity.Utilities;
using System.Collections.Generic;

public class SpawnBases : MonoBehaviour
{
    [SerializeField]
    AbstractMap _map;

    [SerializeField]
    Vector2d[] _locations;
    [SerializeField] BasesGetData data;

    [SerializeField]
    float _spawnScale = 100f;

    [SerializeField]
    GameObject _markerPrefab;

    List<GameObject> _spawnedObjects;

    bool s= false;
```

```
private void OnEnable()
{
    BasesGetData.OnValueChange += Spawn;

}
private void OnDisable()
{
    BasesGetData.OnValueChange -= Spawn;

}

void Spawn()
{
    List<Base> nBases = data.Bases;
    _locations = new Vector2d[nBases.Count];
    _spawnedObjects = new List<GameObject>();
    int i = 0;
    //while (!data.listo) ;
    foreach (Base basex in nBases)
    {
        _locations[i] = new Vector2d(basex.lat, basex.lon);
        var instance = Instantiate(_markerPrefab);
        instance.transform.localPosition = _map.GeoToWorldPosition(_locations[i], true);
        instance.transform.localScale = new Vector3(_spawnScale, _spawnScale, _spawnScale);
        instance.name = basex.name;
        _spawnedObjects.Add(instance);
        i++;
    }
    s=true;
```

```
    }
    private void Update()
    {
        if (s)
        {
            int count = _spawnedObjects.Count;
            for (int i = 0; i < count; i++)
            {
                var spawnedObject = _spawnedObjects[i];
                var location = _locations[i];
                spawnedObject.transform.localPosition = _map.GeoToWorldPosition(location, true);
                spawnedObject.transform.localScale = new Vector3(_spawnScale, _spawnScale, _spawnScale);
            }
        }
    }
}



using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Mapbox.Utils;
using UnityEngine.UI;
public class AStar : MonoBehaviour
{
    [Header("Requerimientos")]
    [SerializeField] BasesGetData data;
    [SerializeField] MapActions mapActions;
```

```
[Header("Conexiones")]
[SerializeField] Slider slider;
[SerializeField, Range(1f, 10f)] public float conDistance=3;
[SerializeField, Range(1f, 20f)] float altura = 15;
[SerializeField] GameObject LinePrefab;
[SerializeField] int conNumber=0;
[SerializeField] bool showConnections = true;
List<GameObject> lines = new List<GameObject>();
float[,] adyacencia;
bool Clines = false;

[Header("A*")]
[SerializeField] float pesoEu = 0.5f;
[SerializeField] float pesoManh = 0.5f;
[SerializeField] GameObject RutaPrefab;
List<GameObject> RutaDrawer = new List<GameObject>();
bool showRoute = false;
bool Croute = false;
bool nr = false;
public int[] ruta = new int[1];

List<Base> nBases;

bool setup = false;
private void OnEnable()
{
   BasesGetData.OnValueChange += Setup;
}
private void OnDisable()
```

```
{
    BasesGetData.OnValueChange -= Setup;
}
// Start is called before the first frame update
void Start()
{


}

// Update is called once per frame
void Update()
{
    if (setup)
    {
        DrawConections();
        DrawRuta(ruta);
    }
}

public void Setup()
{
    if (setup)
    {
        setup = false;
        Clines = false;
        destroyList(lines);
        destroyList(RutaDrawer);
       // showConnections = false;
        showRoute = false;
```

```
        ruta = new int[1];
        Croute = false;
        nr = false;
        conNumber = 0;
    }
    conDistance = slider.value;
    nBases = data.Bases;
    adyacencia = MatrizAdyacencia(nBases);
    setup = true;
}

void destroyList(List<GameObject> l)
{
    foreach(GameObject go in l)
    {
        Destroy(go);
    }
    l.Clear();
}

void DrawConections()
{
    if (!Clines)
    {
        //lines.Clear();
        for (int i = 0; i < conNumber; i++)
        {
            lines.Add(Instantiate(LinePrefab));
        }
```

```
            Clines = true;
            Debug.Log("# conexiones: "+lines.Count);
        }
        if (showConnections)
        {
            int o = 0;
            for (int i = 0; i < nBases.Count - 1; i++)
            {
                for (int j = i; j < nBases.Count; j++)
                {
                    if (adyacencia[i, j] > 0)
                    {
                        Vector2d p1 = new Vector2d(nBases[i].lat, nBases[i].lon);
                        Vector2d p2 = new Vector2d(nBases[j].lat, nBases[j].lon);
                        lines[o].GetComponent<LineRenderer>().SetPosition(0, mapActions.WorldPos(p1) - new
Vector3(0, 0, altura));
                        lines[o].GetComponent<LineRenderer>().SetPosition(1, mapActions.WorldPos(p2) - new
Vector3(0, 0, altura));
                        o++;
                    }
                }
            }
        } else
        {
            for (int i = 0; i < conNumber; i++)
            {
                lines[i].GetComponent<LineRenderer>().SetPosition(0, Vector3.zero);
                lines[i].GetComponent<LineRenderer>().SetPosition(1, Vector3.zero);
            }
```

```
    }

}

void DrawRuta(int[] ruta)
{
    if (!Croute)
    {

        for (int i = 0; i < conNumber; i++)
        {
            RutaDrawer.Add(Instantiate(RutaPrefab));
        }
        Croute = true;
    }

    if (showRoute)
    {
        if (nr)
        {
            for (int i = 0; i < conNumber; i++)
            {

                RutaDrawer[i].GetComponent<LineRenderer>().SetPosition(0, Vector3.zero);
                RutaDrawer[i].GetComponent<LineRenderer>().SetPosition(1, Vector3.zero);
               // Debug.Log("Calles Reset");
            }
            nr = false;
        }
```

```
        int p = 0;
        while (p+1 <= ruta.Length-1)
        {
            Vector2d p1 = new Vector2d(nBases[ruta[p]].lat, nBases[ruta[p]].lon);
            Vector2d p2 = new Vector2d(nBases[ruta[p + 1]].lat, nBases[ruta[p + 1]].lon);
            RutaDrawer[p].GetComponent<LineRenderer>().SetPosition(0, mapActions.WorldPos(p1) - new
Vector3(0, 0, altura));
            RutaDrawer[p].GetComponent<LineRenderer>().SetPosition(1, mapActions.WorldPos(p2) - new
Vector3(0, 0, altura));
            //Debug.Log("Diujado entre "+ nBases[ruta[p]].name +" y "+ nBases[ruta[p+1]].name);
            p++;
        }
    }
    else
    {
        for (int i = 0; i < conNumber; i++)
        {
            RutaDrawer[i].GetComponent<LineRenderer>().SetPosition(0, Vector3.zero);
            RutaDrawer[i].GetComponent<LineRenderer>().SetPosition(1, Vector3.zero);
        }
    }

}

bool Conexion(Base b1, Base b2)
{
    Vector2d p1 = new Vector2d(b1.lat, b1.lon);
    Vector2d p2 = new Vector2d(b2.lat, b2.lon);
    float dist = mapActions.GetDistanceH(p1, p2) / 1000;
```

```
        if (dist <= conDistance)
        {
            bool choque = Physics.Linecast(mapActions.WorldPos(p1) - new Vector3(0, 0, altura),
mapActions.WorldPos(p2) - new Vector3(0, 0, altura), 2);
            if (!choque) return true;
        }
        return false;

    }
    float[,] MatrizAdyacencia(List<Base> b)
    {
        float[,] ady = new float[b.Count, b.Count];
        for (int i = 0; i < b.Count; i++)
        {
            for (int j = 0; j < b.Count; j++)
            {
                ady[i, j] = -1;
            }
        }

        for (int i = 0; i < b.Count - 1; i++)
        {
            ady[i, i] = 0;
            for (int j = i + 1; j < b.Count; j++)
            {
                if (Conexion(b[i], b[j]))
                {
                    ady[i, j] = mapActions.GetDistanceH(new Vector2d(b[i].lat, b[i].lon), new Vector2d(b[j].lat,
b[j].lon)) / 1000;
```

```
                ady[j, i] = ady[i, j];
                conNumber++;
            }
        }
    }
    return ady;
}

float[] Heuristica(float c1, float c2, Base target)
{
    float[] He = new float[nBases.Count];
    float manhattan;
    float euclidea;
    Vector2d p1,p2;
    p2 = new Vector2d(target.lat, target.lon);
    Debug.Log("Heuristica calculada: ");
    for (int i = 0; i < nBases.Count; i++)
    {
        p1 = new Vector2d(nBases[i].lat, nBases[i].lon);
        euclidea = mapActions.GetDistanceH(p1,p2) / 1000;
        manhattan = Mathf.Abs(mapActions.WorldPos(p1).x- mapActions.WorldPos(p2).x)+
Mathf.Abs(mapActions.WorldPos(p1).y - mapActions.WorldPos(p2).y);
        He[i] = c1*euclidea+c2*manhattan;
        Debug.Log(He[i]);
    }
    Debug.Log("----------------------");
    return He;
}
```

```
public void Calcular()
{
    bool e = false; //deteccion de errores;
    Base From = mapActions.GetFrom();
    Base To = mapActions.GetTo();
    int n = nBases.Count;
    float[,] tabla = new float[n,3]; //final temporal predecesor

    Debug.Log("Iniciando A*");

    float[] He = Heuristica(pesoEu, pesoManh, To);
    Debug.Log("Heuristica Calculada");

    for (int i = 0; i < n; i++) tabla[i, 0] = tabla[i, 1] = tabla[i, 2] = 1000000000; //inicializacion con valores grandes


    int actual = nBases.IndexOf(From);
    tabla[actual, 0] = tabla[actual, 1] = tabla[actual, 2] = 0;

    for (int i = 0; i < n; i++)
    {
       Debug.Log("[" + tabla[i, 0] + "," + tabla[i, 1] + "," + tabla[i, 2] + "]");
    }

    Debug.Log("Tabla Calculada");

    Debug.Log("Nodo inicial: " + actual +" ("+ nBases[actual].name+")");
```

```
Debug.Log("Nodo final: " + nBases.IndexOf(To) + " (" + nBases[nBases.IndexOf(To)].name + ")");

#region A*
while (tabla[nBases.IndexOf(To), 0] == 1000000000 && !e)
{
   Debug.Log("Nodo actual: " + actual);
   #region comprobar vecinos
   for (int i = 0; i < n; i++)
   {
      if (adyacencia[actual, i] > 0)
      {
         if(tabla[i,1]> adyacencia[actual, i]+ tabla[actual, 0]+He[i]&& tabla[i, 0] == 1000000000)
         {
            tabla[i, 1] = adyacencia[actual, i] + tabla[actual, 0] + He[i];
            tabla[i, 2] = actual;
         }
      }
   }
   #endregion

   #region siguiente nodo
   Vector2 aux = new Vector2(1000000000, -1);
   for (int i = 0; i < n; i++)
   {
      if (aux.x > tabla[i, 1] && tabla[i,0]== 1000000000)
      {
         aux.x = tabla[i, 1];
         aux.y=i;
      }
```

```
    }

    e= aux.y < 0 ? true:false;
if(e) break;
    tabla[(int)aux.y, 0] = aux.x-He[(int)aux.y];
    actual =(int) aux.y;
    Debug.Log("Nodo siguiente: " + actual);
    #endregion

}

if (e)
{
    Debug.Log("Ruta no encontrada");
    ruta = new int[1];
}

#region encontrar ruta
if (!e)
{
    int i = 0;
    int a = nBases.IndexOf(To);
    ruta = new int[1];
    ruta[i] = a;
    while (a != nBases.IndexOf(From))
    {
        a = (int) tabla[a, 2];
        System.Array.Resize<int>(ref ruta, ruta.Length + 1);
        i++;
```

```
            ruta[i] = a;
        }

        Debug.Log("Ruta encontrada");
        //invertir ruta
        int[] temp = new int[ruta.Length];
        for (int x = 0; x < ruta.Length; x++)
        {
            temp[x] = ruta[ruta.Length - 1 - x];
        }
        ruta = temp;
        string msg = "";
        for (int x = 0; x < ruta.Length; x++)
        {
            msg = msg + "/"+ ruta[x] + "/ ";
        }
        Debug.Log(msg);
        nr = true;
        showRoute = true;
    }
    else
    {
        showRoute = false;
    }
    #endregion

    #endregion

}
```

```
    public void showConns()
    {
        showConnections = !showConnections;
    }

    void OnDrawGizmos()
    {
        if (setup)
        {
            Gizmos.color = Color.gray;
            for (int i = 0; i < nBases.Count - 1; i++)
            {
                for (int j = i; j < nBases.Count; j++)
                {
                    if (adyacencia[i, j] > 0)
                    {
                        Vector2d p1 = new Vector2d(nBases[i].lat, nBases[i].lon);
                        Vector2d p2 = new Vector2d(nBases[j].lat, nBases[j].lon);
                        Gizmos.DrawLine(mapActions.WorldPos(p1)-new Vector3(0,0,altura),
mapActions.WorldPos(p2) - new Vector3(0, 0, altura));
                    }
                }
            }
        }
    }
}

using System.Collections;
```

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
public class updatenumber : MonoBehaviour
{
   Slider slider;
   TMP_Text text;
   // Start is called before the first frame update
   void Start()
   {
      text = GetComponent<TMP_Text>();
      slider = GetComponentInParent<Slider>();
      text.text = slider.value.ToString("F2") + text.text.Replace("###","");
   }

   public void getSliderValue()
   {
      text.text = slider.value.ToString("F2") + " km";
   }

   public void getSliderValuem()
   {
      text.text = slider.value.ToString("F2") + " m";
   }
   public void getSliderValueA(string message)
   {
      text.text = slider.value.ToString("F2") + " "+message;
   }
```

```
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;

public class FillList : MonoBehaviour
{
    [SerializeField] BasesGetData data;
    bool r = false;
    public TMP_Dropdown salida, llegada;
    List<string> ListaM = new List<string>();
    [SerializeField] List<string> ListaLleg= new List<string>(), ListaSal;
    // Start is called before the first frame update
    void Start()
    {
        salida.onValueChanged.AddListener(delegate { actualizarLlegada(); });
    }
    void Awake()
    {

    }
    // Update is called once per frame
    void Update()
    {
        if (!r && data.listo)
        {
```

```
        List<Base> nBases = data.Bases;
        foreach (Base basex in nBases)
        {
            ListaM.Add(basex.name.Replace('_',' '));
        }
        ActualizarSalida();
        actualizarLlegada();
        r = true;
    }
}

void actualizarLlegada()
{
    ListaLleg.Clear();
    foreach (string b in ListaM)
    {
        if(b != salida.captionText.text) ListaLleg.Add(b);
    }
    llegada.ClearOptions();
    llegada.AddOptions(ListaLleg);
    Debug.Log("LLEGADA ACTUALIZADA");
}
void ActualizarSalida()
{
    ListaSal = ListaM;
    salida.ClearOptions();
    salida.AddOptions(ListaSal);
    Debug.Log("SALIDA LISTA");
}
```

```
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
public class updatenumber : MonoBehaviour
{
   Slider slider;
   TMP_Text text;
   // Start is called before the first frame update
   void Start()
   {
      text = GetComponent<TMP_Text>();
      slider = GetComponentInParent<Slider>();
      text.text = slider.value.ToString("F2") + text.text.Replace("###","");
   }

   public void getSliderValue()
   {
      text.text = slider.value.ToString("F2") + " km";
   }

   public void getSliderValuem()
   {
      text.text = slider.value.ToString("F2") + " m";
   }
```

```
    public void getSliderValueA(string message)
    {
        text.text = slider.value.ToString("F2") + " "+message;
    }
}

using UnityEngine;
using Mapbox.Utils;
using Mapbox.Unity.Map;
using Mapbox.Unity.MeshGeneration.Factories;
using Mapbox.Unity.Utilities;
using System.Collections.Generic;

public class SpawnBases : MonoBehaviour
{
    [SerializeField]
    AbstractMap _map;

    [SerializeField]
    Vector2d[] _locations;
    [SerializeField] BasesGetData data;

    [SerializeField]
    float _spawnScale = 100f;

    [SerializeField]
    GameObject _markerPrefab;

    List<GameObject> _spawnedObjects;
```

```
bool s= false;
private void OnEnable()
{
   BasesGetData.OnValueChange += Spawn;

}
private void OnDisable()
{
   BasesGetData.OnValueChange -= Spawn;

}

void Spawn()
{
   List<Base> nBases = data.Bases;
   _locations = new Vector2d[nBases.Count];
   _spawnedObjects = new List<GameObject>();
   int i = 0;
   //while (!data.listo) ;
   foreach (Base basex in nBases)
   {
      _locations[i] = new Vector2d(basex.lat, basex.lon);
      var instance = Instantiate(_markerPrefab);
      instance.transform.localPosition = _map.GeoToWorldPosition(_locations[i], true);
      instance.transform.localScale = new Vector3(_spawnScale, _spawnScale, _spawnScale);
      instance.name = basex.name;
      _spawnedObjects.Add(instance);
      i++;
```

```
        }
        s=true;
    }
    private void Update()
    {
        if (s)
        {
            int count = _spawnedObjects.Count;
            for (int i = 0; i < count; i++)
            {
                var spawnedObject = _spawnedObjects[i];
                var location = _locations[i];
                spawnedObject.transform.localPosition = _map.GeoToWorldPosition(location, true);
                spawnedObject.transform.localScale = new Vector3(_spawnScale, _spawnScale, _spawnScale);
            }
        }
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using System.IO;
using System.IO.Ports;

public class SerialPortManager : MonoBehaviour
{
    [SerializeField] TMP_Dropdown Dropdown;
```

```
public string[] ports;
SerialPort p;

// Start is called before the first frame update
void Start()
{

}
public void GetPorts()
{
    ports = System.IO.Ports.SerialPort.GetPortNames();
    Dropdown.ClearOptions();
    Dropdown.AddOptions(new List<string>(ports));

}

public void Openport()
{
    //loadingIndicator.SetActive(true);
    if (p == null) p = new SerialPort(Dropdown.captionText.text, 115200);
    if (p != null && !p.IsOpen)
    {
        p.Parity = Parity.Even;
        p.StopBits = StopBits.One;
        p.DataBits = 8;
        p.Open();

        while (!p.IsOpen)
```

```
        {
            Debug.Log("waiting: " + Time.time);
        }
        p.DataReceived += new SerialDataReceivedEventHandler(port_DataReceived);
        Debug.Log("puerto abierto... escuchando");
    }
    if (p != null && p.IsOpen)
    {

        p.WriteLine(" # ");
        Debug.Log("frist done");

        float timer = 0;
        while (timer < 60*100)
        {
            timer += Time.smoothDeltaTime;
            Debug.Log(timer);
        }
        p.WriteLine(" # ");
        Debug.Log("CLI ready");

    }
}
public void Closeport()
{
    p.DataReceived -= new SerialDataReceivedEventHandler(port_DataReceived);
    p.Close();
}
public bool Status()
```

```
{
    if(p != null && p.IsOpen) if (p.BytesToWrite>0 || p.BytesToRead>0) return false;
    if (p != null) return p.IsOpen;
    return false;
}
public void SendData(string message)
{
    try
    {

        if(p.IsOpen)p.WriteLine(message);
        while (p.BytesToWrite > 0) ;

    }
    catch (IOException ex)
    {
        // Failed to open com port or start serial thread
        Debug.LogError("IOException : " + ex.Message.ToString());
    }
}
private void Update()
{
    if(p != null && p.IsOpen && p.BytesToRead > 0)
    {
        Debug.Log(p.ReadExisting());
    }
}
private void port_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
```

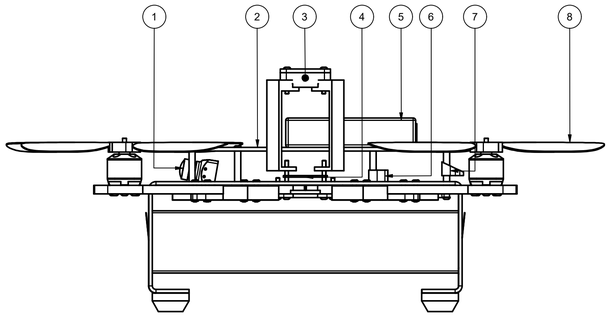
```
        // Show all the incoming data in the port's buffer
        Debug.Log(p.ReadExisting());
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Mapbox.Unity.Map;
using Mapbox.Utils;
using TMPro;

public class MapActions : MonoBehaviour
{
    [SerializeField] AbstractMap map;
    [SerializeField] BasesGetData data;
    List<Base> nBases;
    public TMP_Dropdown salida, llegada;
    public void GoTo()
    {
        Vector2d P1 = GetCoords(salida.captionText.text.Replace(" ", "_"));
        Vector2d P2 = GetCoords(llegada.captionText.text.Replace(" ", "_"));
        Debug.Log("Distancia: " + GetDistanceH(P1, P2)/1000 +"m");
        Debug.Log("Zoom: " + CalcZoom(GetDistanceH(P1, P2)));
        map.UpdateMap((P1+P2)/2, CalcZoom(GetDistanceH(P1, P2)));
    }
    // Start is called before the first frame update
    private void OnEnable()
    {
```

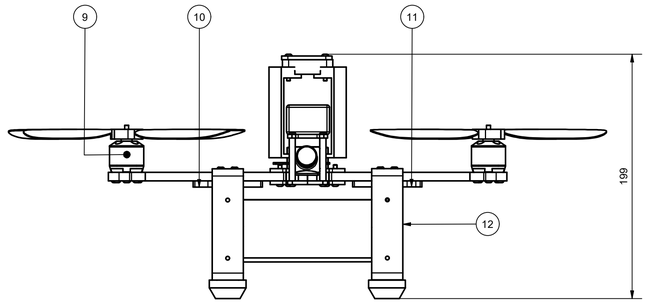
```
    BasesGetData.OnValueChange += ActualizarLista;
}
private void OnDisable()
{
    BasesGetData.OnValueChange -= ActualizarLista;
}
void ActualizarLista()
{
    nBases = data.Bases;
}
Vector2d GetCoords(string name)
{
    foreach (Base basex in nBases)
    {
        if (basex.name == name)
        {
            return new Vector2d(basex.lat, basex.lon);
        }
    } return  new Vector2d(0, 0);
}

public float GetDistanceH(Vector2d P1, Vector2d P2)
{
    //formula de haversine
    float R = 6371000; //Radio de la tierra en m
    float la1 =Mathf.Deg2Rad*(float) P1.x, la2 = Mathf.Deg2Rad * (float) P2.x;
    float Dlat = Mathf.Deg2Rad * (float)(P1.x - P2.x); //delta latitud
    float Dlon = Mathf.Deg2Rad * (float)(P1.y - P2.y); //delta longitud
```

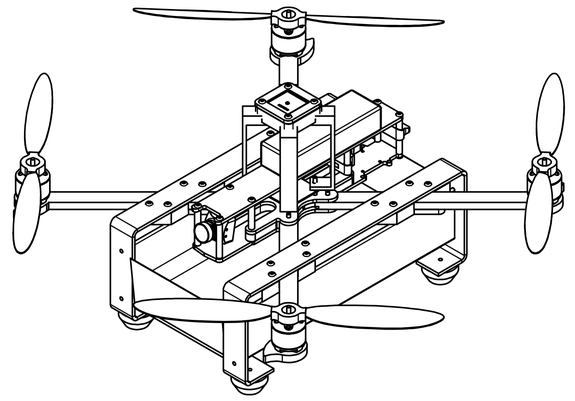
```
        float a = (Mathf.Sin(Dlat/2) * Mathf.Sin(Dlat/2)) + Mathf.Cos(la1) * Mathf.Cos(la2)* (Mathf.Sin(Dlon / 2) * Mathf.Sin(Dlon / 2));
        float c=2*Mathf.Atan2(Mathf.Sqrt(a),Mathf.Sqrt(1-a));
        float d = R*c;
        return d;
    }

    float CalcZoom(float distance)
    {
        //distancia en m
        float zoom = interpolar(1.88f, 13.5f, 10.37f, 12f, distance/1000);
        if (distance / 1000 < 0.5f) zoom = interpolar(0.16f, 17f, 1.88f, 13.5f, distance / 1000);
        if (zoom<2) zoom = 2;
        return zoom;
    }
    float interpolar(float x1,float y1, float x2, float y2,float xN)
    {
        float yN=y1+((y2-y1)/(x2-x1))*(xN-x1);
        return yN;
    }
    public Vector3 WorldPos(Vector2d p)
    {
        return map.GeoToWorldPosition(p);
    }

    public Base GetFrom()
    {
        string name = salida.captionText.text.Replace(" ", "_");
        foreach (Base basex in nBases)
```

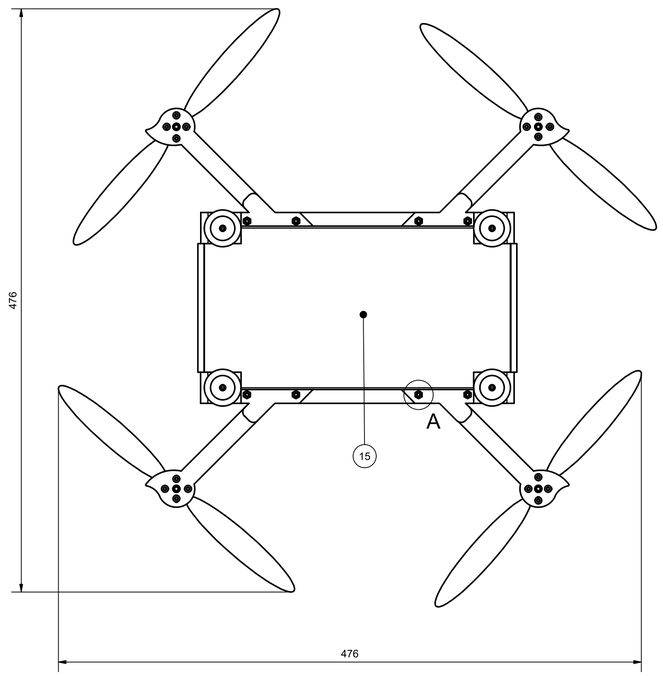
```
        {
            if (basex.name == name)
            {
                return basex;
            }
        }
        return null;
    }

    public Base GetTo()
    {
        string name = llegada.captionText.text.Replace(" ", "_");
        foreach (Base basex in nBases)
        {
            if (basex.name == name)
            {
                return basex;
            }
        }
        return null;
    }
}
```

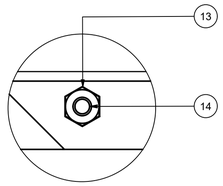
Anexo E: Planos de construcción











DETALLE A
ESC: 2 : 1

VOLTAJE DE ALIMENTACIÓN	
BATERÍA	10.5 VDC - 12.6 VDC (3S)
FRECUENCIA DE OPERACIÓN	
CONTROL REMOTO	2.4 GHz
VIDEO ANALÓGICO	5.8 GHz
DIMENSIONES GENERALES	
LARGO	476 mm
ANCHO	476 mm
PROFUNDIDAD	199 mm
MASA	
TOTAL	1.1 kg
CARGA	
MÁXIMA	500 g

POS	ZONA	DENOMINACIÓN	CANT	MATERIAL	PLANO FORMA	OBSERVACIONES
15	I3	CONTENEDOR DE CARGA	1	-	-	MALLA DE NYLON
14	I7	PERNO M3	36	ACERO	-	LONGITUD 16mm
13	I7	TUERCA M3	32	ACERO	-	-
12	D7	SUJETADOR DE CARGA	2	ALUMINIO	D03-105	-
11	D7	ADAPTADOR CARGA 2	2	PLA	D03-104	-
10	D6	ADAPTADOR CARGA 1	2	PLA	D03-103	-
9	D5	MOTOR SIN ESCOBILLAS	4	-	-	2212-1000KV
8	D4	PROPELA	4	NYLON	-	MODELO 1045
7	D4	SUJECIÓN ANTENA VTX	1	PLA	D03-101	-
6	D3	SUJECIÓN ANTENA RC	1	PLA	D03-102	-
5	D3	BATERÍA	1	-	-	LIPO 3S - 50C
4	D3	CONTROLADOR DE VUELO	1	-	-	-
3	C3	ESTRUCTURA GPS	1	-	D03-003	-
2	D3	ESTRUCTURA PRINCIPAL	1	-	D03-002	-
1	D2	CÁMARA ANALÓGICA	1	-	-	FOXEER RAZER MINI 4:3

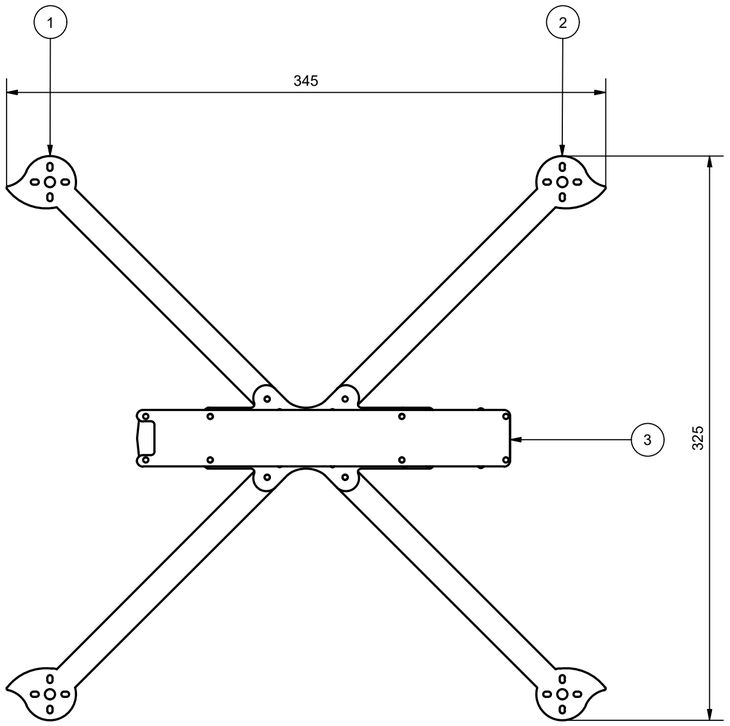
UIDE INGENIERÍA MECATRÓNICA

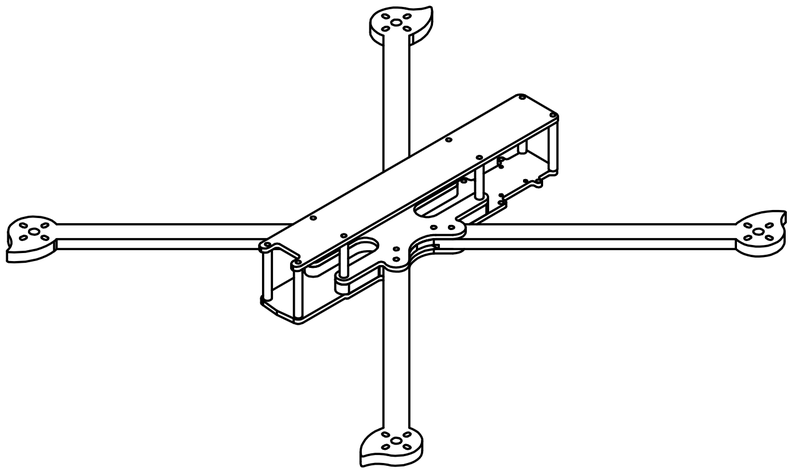
DIB.	CUESTA S.	20/03/2022
DIS.	CUESTA S.	15/02/2022
REV	OSCULLO C.	10/04/2022

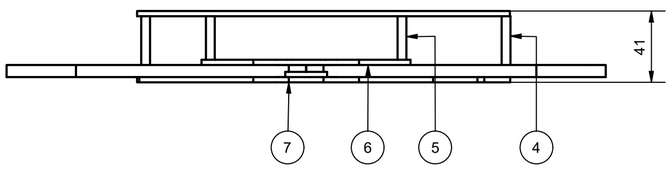
DRON DE CARGA PARA CARGA ÚTIL DE 500 g

D03-001

ESCALA 1:2.5







DIMENSIONES GENERALES

LARGO	345 mm
ANCHO	325 mm
PROFUNDIDAD	41 mm

POS	ZONA	DENOMINACIÓN	CANT	MATERIAL	PLANO FORMA	OBSERVACIONES
7	E4	PLACA INFERIOR	1	CFRP	D03-205	-
6	E3	PLACA MEDIA	1	CFRP	D03-204	-
5	E4	TUBO SEPARADOR 2	4	ALUMINIO 6061	-	M3 x 25mm
4	E4	TUBO SEPARADOR 1	4	ALUMINIO 6061	-	M3 x 35mm
3	B3	PLACA SUPERIOR	1	CFRP	D03-203	-
2	B4	BRAZO 2	2	CFRP	D03-202	-
1	B3	BRAZO 1	2	CFRP	D03-201	-

TABLE

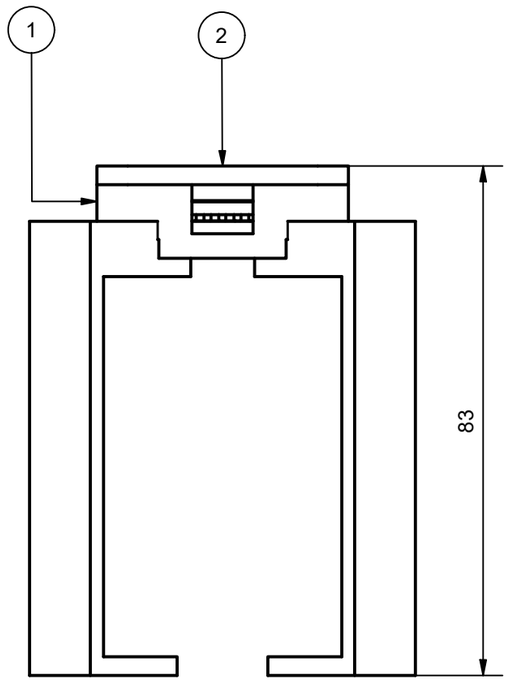
UIDE — INGENIERÍA MECATRÓNICA

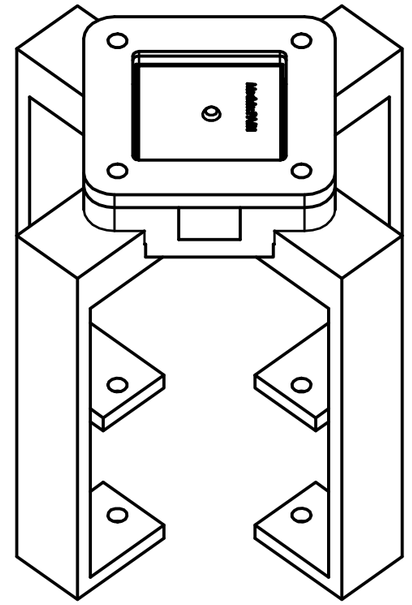
DIB.	CUESTA S.	20/03/2022
DIS.	CUESTA S.	15/02/2022
REV.	OSCULLO C.	29/03/2022

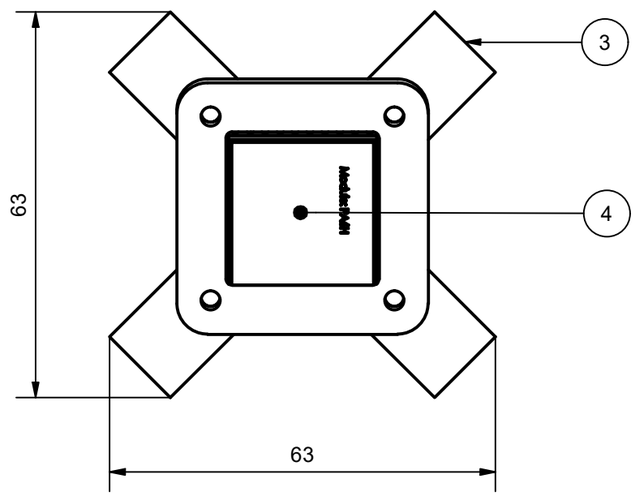
ESTRUCTURA PRINCIPAL

D03-002

ESCALA 1 : 2.5







DIMENSIONES GENERALES

LARGO	62.9 mm
ANCHO	62.9 mm
PROFUNDIDAD	83 mm

POS	ZONA	DENOMINACIÓN	CANT	MATERIAL	PLANO FORMA	OBSERVACIONES
4	E2	GPS	1	-	-	M80
3	D3	BRAZO GPS	4	PLA	D03-303	-
2	B2	TAPA GPS	1	PLA	D03-302	-
1	B2	CONTENEDOR GPS	1	PLA	D03-301	-

UIDE INGENIERÍA MECATRÓNICA

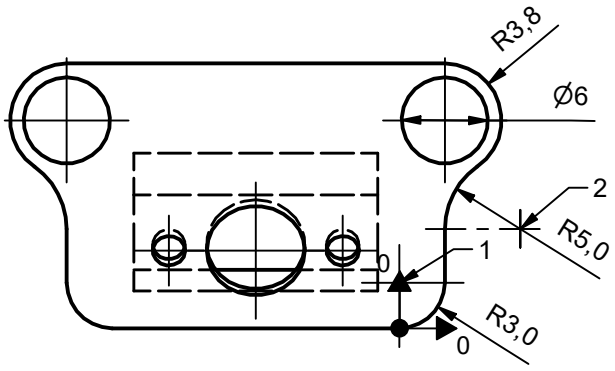
DIB.	CUESTA S.	20/03/2022
DIS.	CUESTA S.	15/02/2022
REV.	OSCULLO C.	29/03/2022

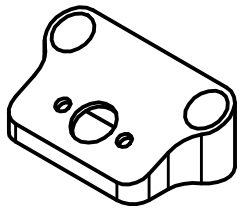
ESTRUCTURA GPS

D03-003

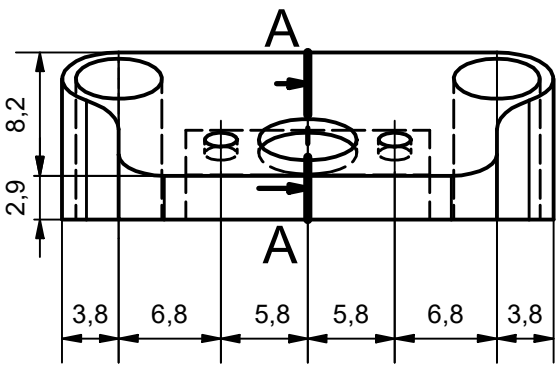
ESCALA 1 : 1

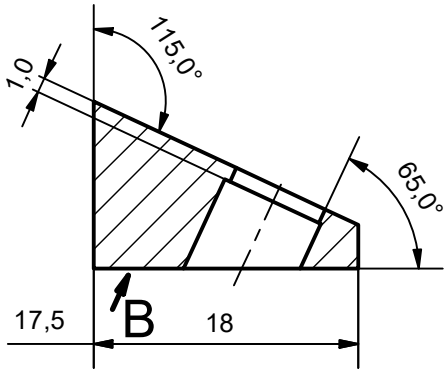
N7



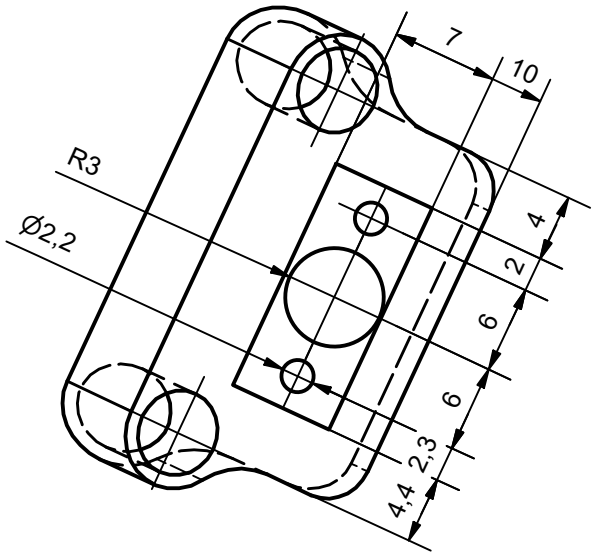


ESCALA: 1:1





SECCIÓN A-A
ESCALA: 2 : 1



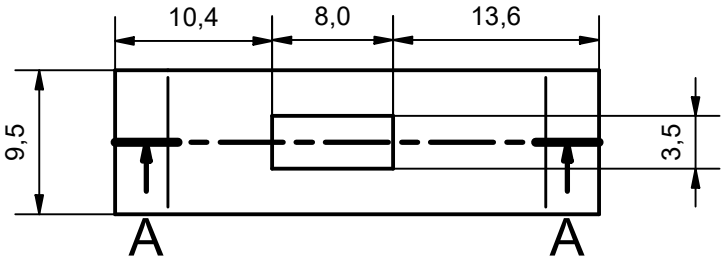
VISTA B

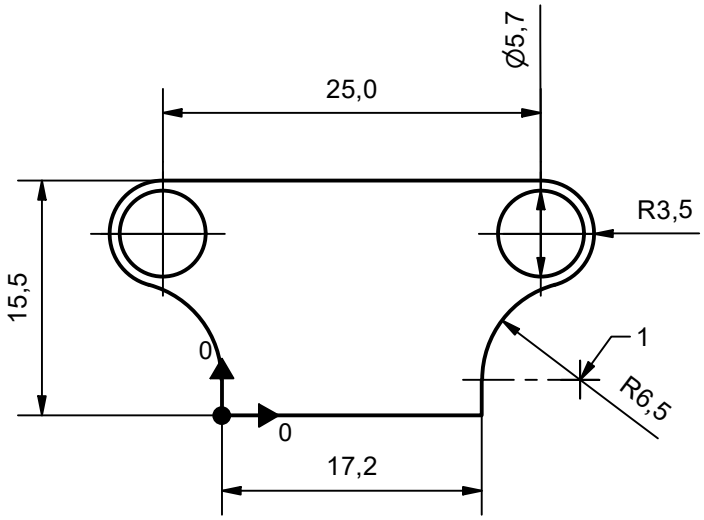
NOTA: IMPRESIÓN 3D
- Temperatura: 210°C
- Relleno: 10%
- Tipo: Cúbico
- Altura de capa: 0.16mm

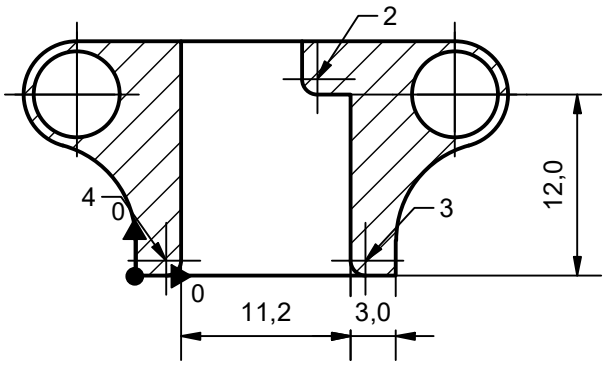
#	X	Y
1	0,0	3,0
2	8,0	6,6

TRATAMIENTO: SIN TRATAMIENTO		UIDE	INGENIERIA MECATRÓNICA			
RECUBRIMIENTO: SIN RECUBRIMIENTO						
MATERIAL: PLA	TOL. GRAL ± 0.1	ESCALA: 2 : 1	DIB.	CUESTA S.	20/03/2022	
			DIS.	CUESTA S.	15/02/2022	
			REV.	OSCULLO C.	10/04/2022	
SUJECIÓN ANTENA VTX		D03 - 101				

N7





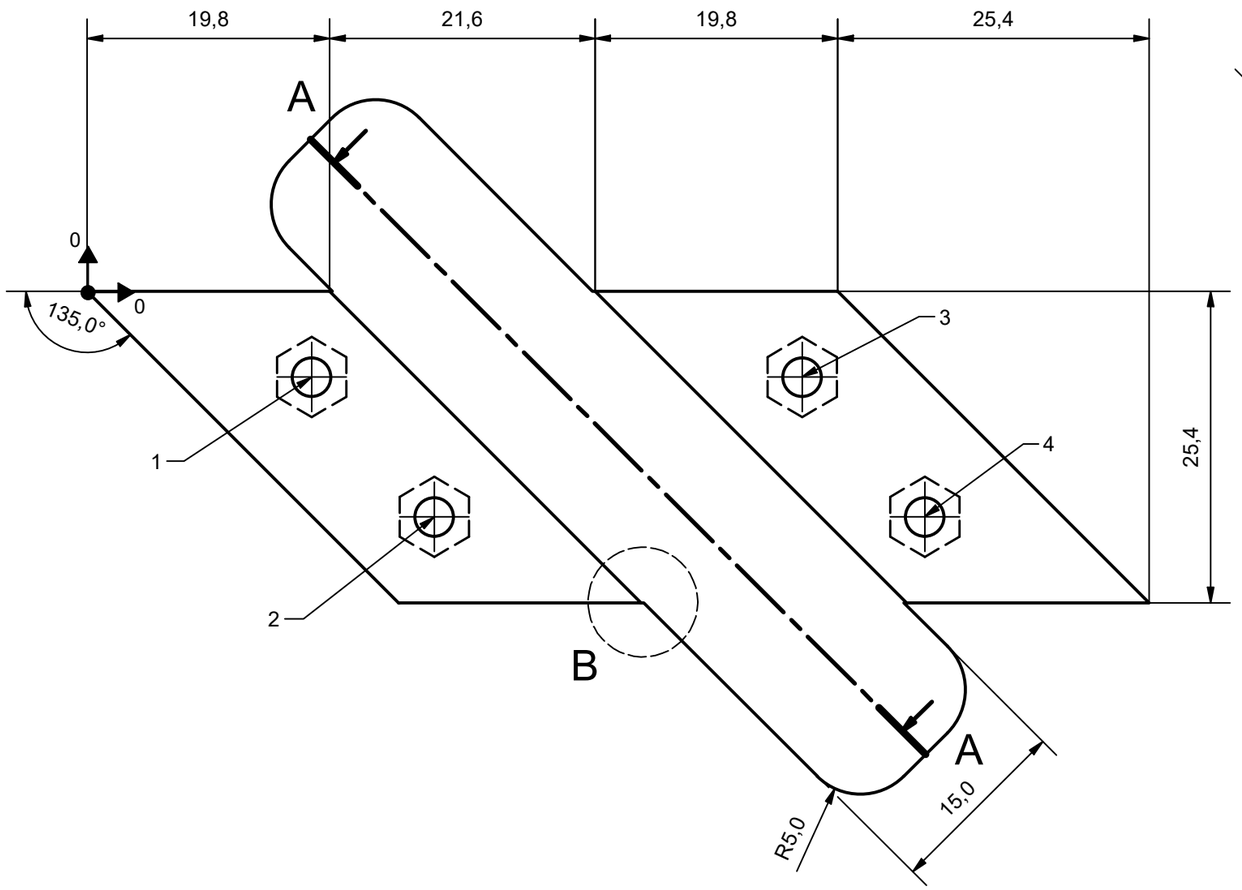


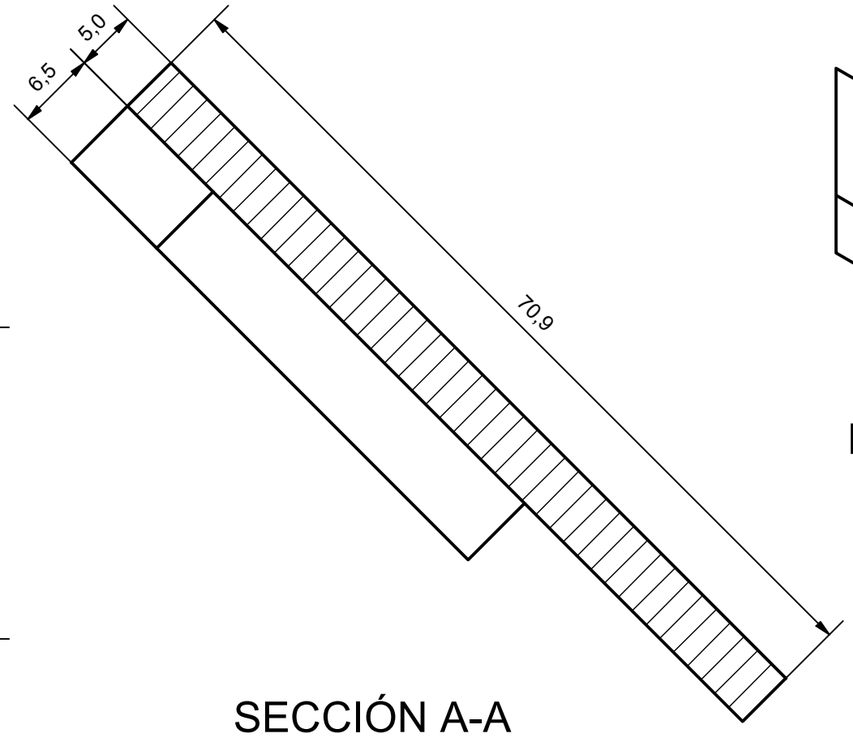
SECCIÓN A-A
ESCALA: 2 : 1

#	X	Y	R
1	23,7	2,3	6,5
2	12,0	13,0	1,0
3	15,2	1,0	1,0
4	2,0	1,0	1,0

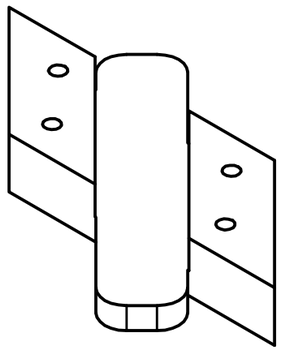
NOTA: IMPRESIÓN 3D
- Temperatura: 210°C
- Relleno: 10%
- Tipo: Cúbico
- Altura de capa: 0.16mm

TRATAMIENTO: SIN TRATAMIENTO		UIDE	INGENIERIA MECATRÓNICA
RECUBRIMIENTO: SIN RECUBRIMIENTO			
MATERIAL: PLA	TOL. GRAL ± 0.1	ESCALA: 2 : 1	DIB. CUESTA S. 20/03/2022
			DIS. CUESTA S. 15/02/2022
			REV. OSCULLO C. 10/04/2022
SUJECIÓN ANTENA RC		D03 - 102	

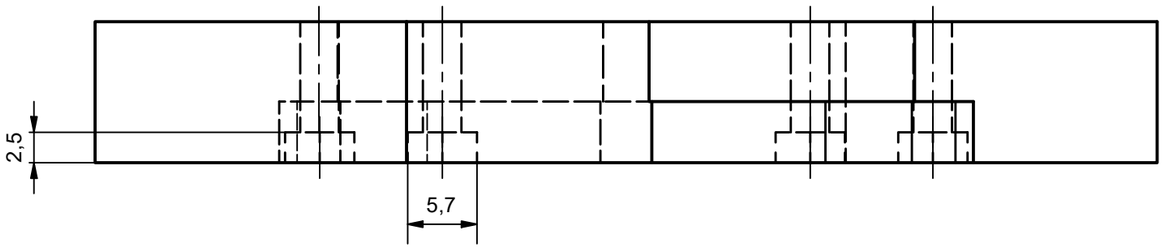


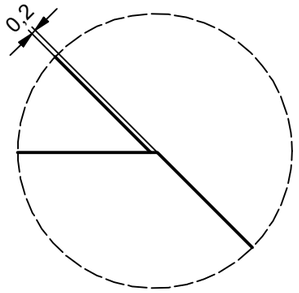


SECCIÓN A-A
ESCALA: 2 : 1



ESCALA: 1:1



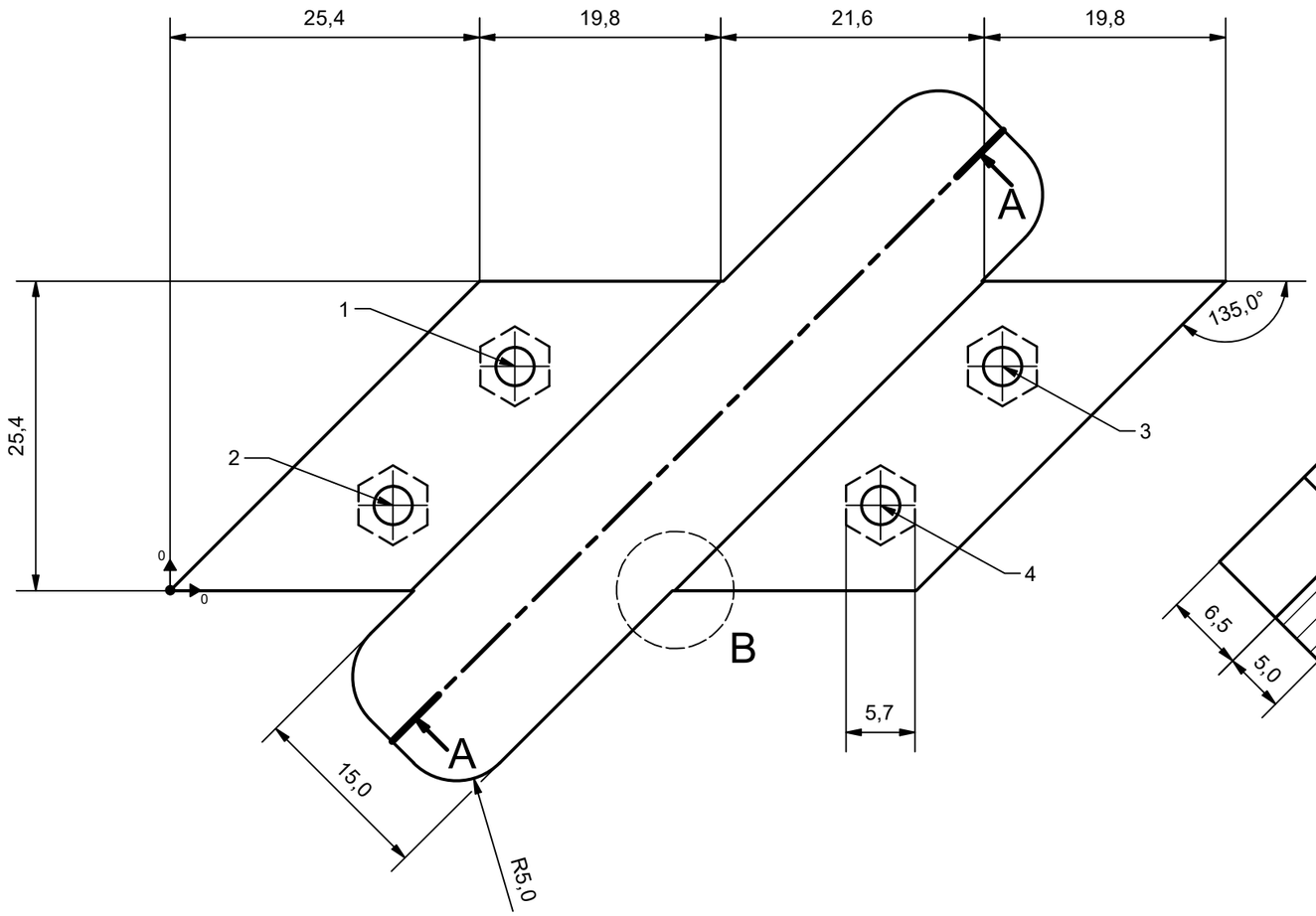


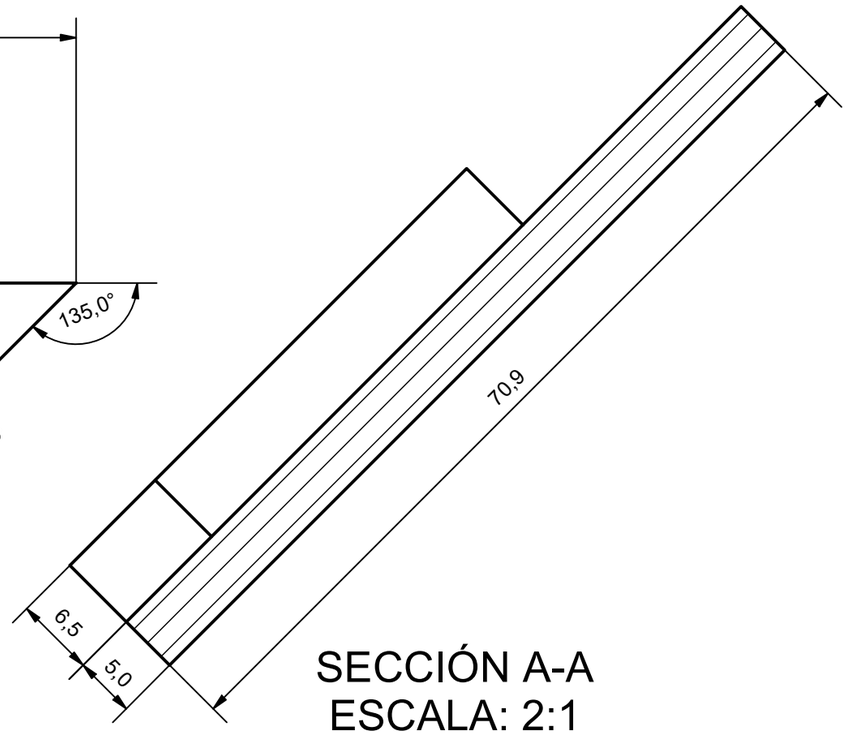
DETALLE B
ESC: 5 : 1

#	X	Y	R
1	18,3	-7,0	3,2
2	28,3	-18,4	3,2
3	58,3	-7,0	3,2
4	68,3	-18,4	3,2

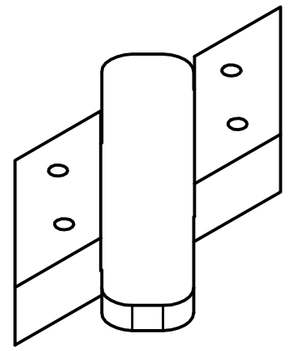
NOTA: IMPRESIÓN 3D
- Temperatura: 210°C
- Relleno: 50%
- Tipo: Cúbico
- Altura de capa: 0.16mm

TRATAMIENTO: SIN TRATAMIENTO		UIDE	INGENIERIA MECATRÓNICA
RECUBRIMIENTO: SIN RECUBRIMIENTO			
MATERIAL: PLA	TOL. GRAL ± 0.1	ESCALA: 2 : 1	DIB. CUESTA S. 20/03/2022
			DIS. CUESTA S. 15/02/2022
			REV. OSCULLO C. 10/04/2022
ADAPTADOR CARGA 1		D03 - 103	

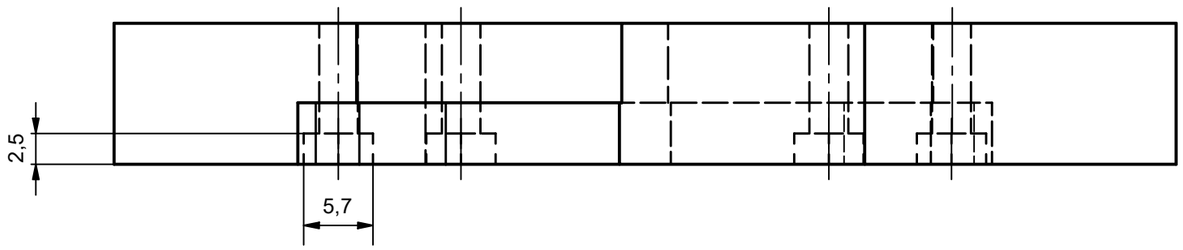


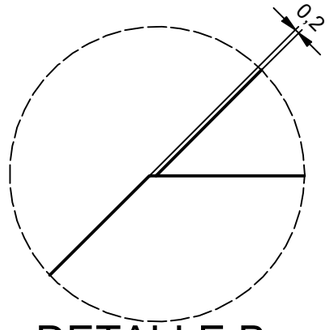


SECCIÓN A-A
ESCALA: 2:1



ESCALA: 1:1





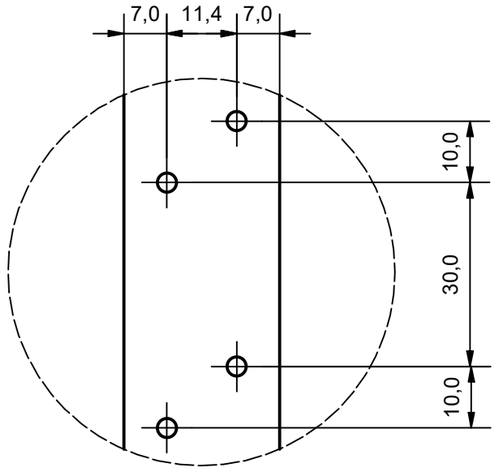
DETALLE B
ESC: 5 : 1

#	X	Y	R
1	28,3	18,4	3,2
2	18,3	7,0	3,2
3	68,3	18,4	3,2
4	58,3	7,0	3,2

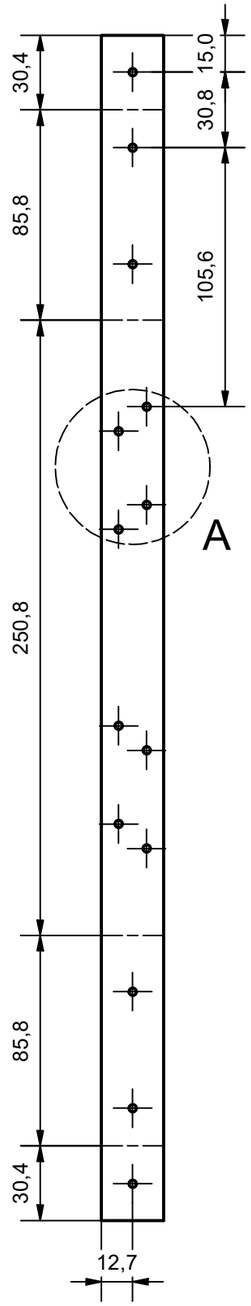
NOTA: IMPRESIÓN 3D
- Temperatura: 210°C
- Relleno: 50%
- Tipo: Cúbico
- Altura de capa: 0.16mm

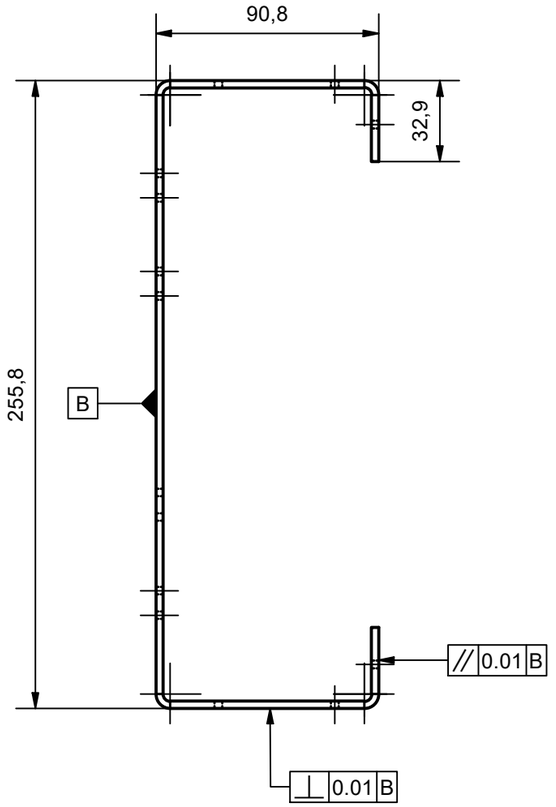
TRATAMIENTO: SIN TRATAMIENTO		UIDE	INGENIERIA MECATRÓNICA			
RECUBRIMIENTO: SIN RECUBRIMIENTO						
MATERIAL: PLA	TOL. GRAL ± 0.1	ESCALA: 2:1	DIB.	CUESTA S.	20/03/2022	
			DIS.	CUESTA S.	15/02/2022	
			REV.	OSCULLO C.	10/04/2022	
ADAPTADOR CARGA 2		D03 - 104				

N8



DETALLE A
ESC: 1:1

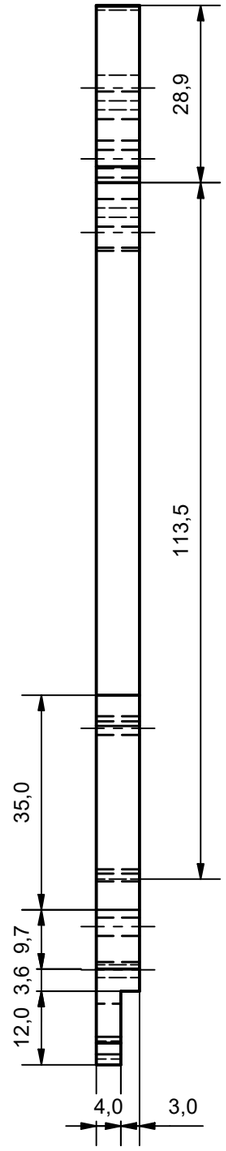


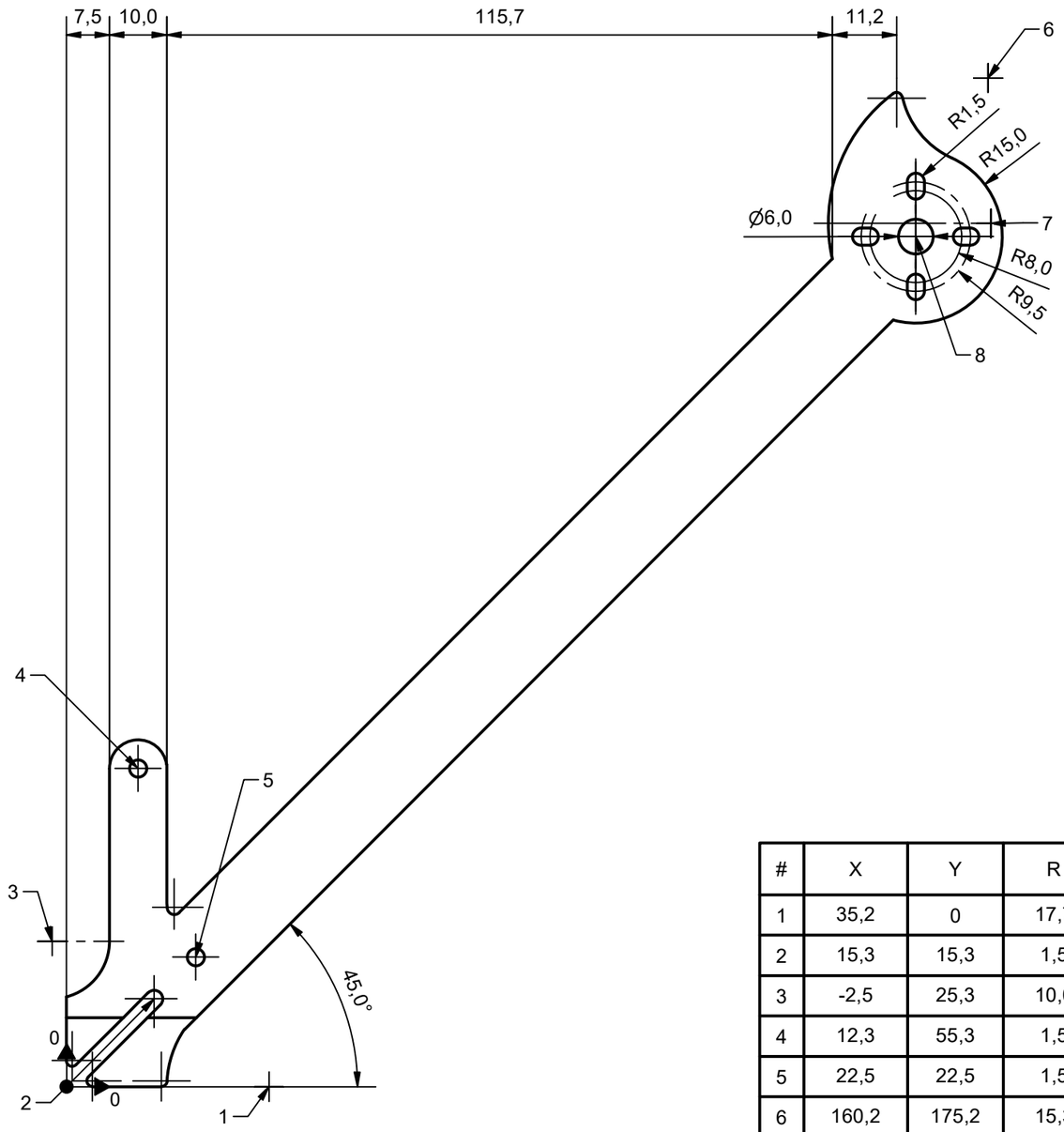


NOTA: Pletina CEDAL 1767
- Espesor: 2.9mm
- Ancho: 25.4mm
- Peso aproximado: 0.199kg/m

NOTA: AGUJEROS
- Diámetro: 3.15mm

TRATAMIENTO: SIN TRATAMIENTO		UIDE	INGENIERIA MECATRÓNICA			
RECUBRIMIENTO: SIN RECUBRIMIENTO						
MATERIAL: ALUMINIO ASTM 6061	TOL. GRAL ± 0.1	ESCALA: 1 : 2.5	DIB.	CUESTA S.	20/03/2022	
			DIS.	CUESTA S.	15/02/2022	
			REV.	OSCULLO C.	10/04/2022	
SUJETADOR CARGA 1		D03 - 105				

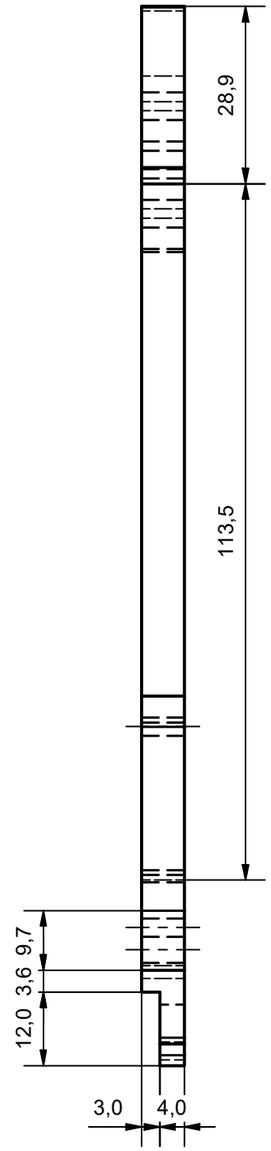


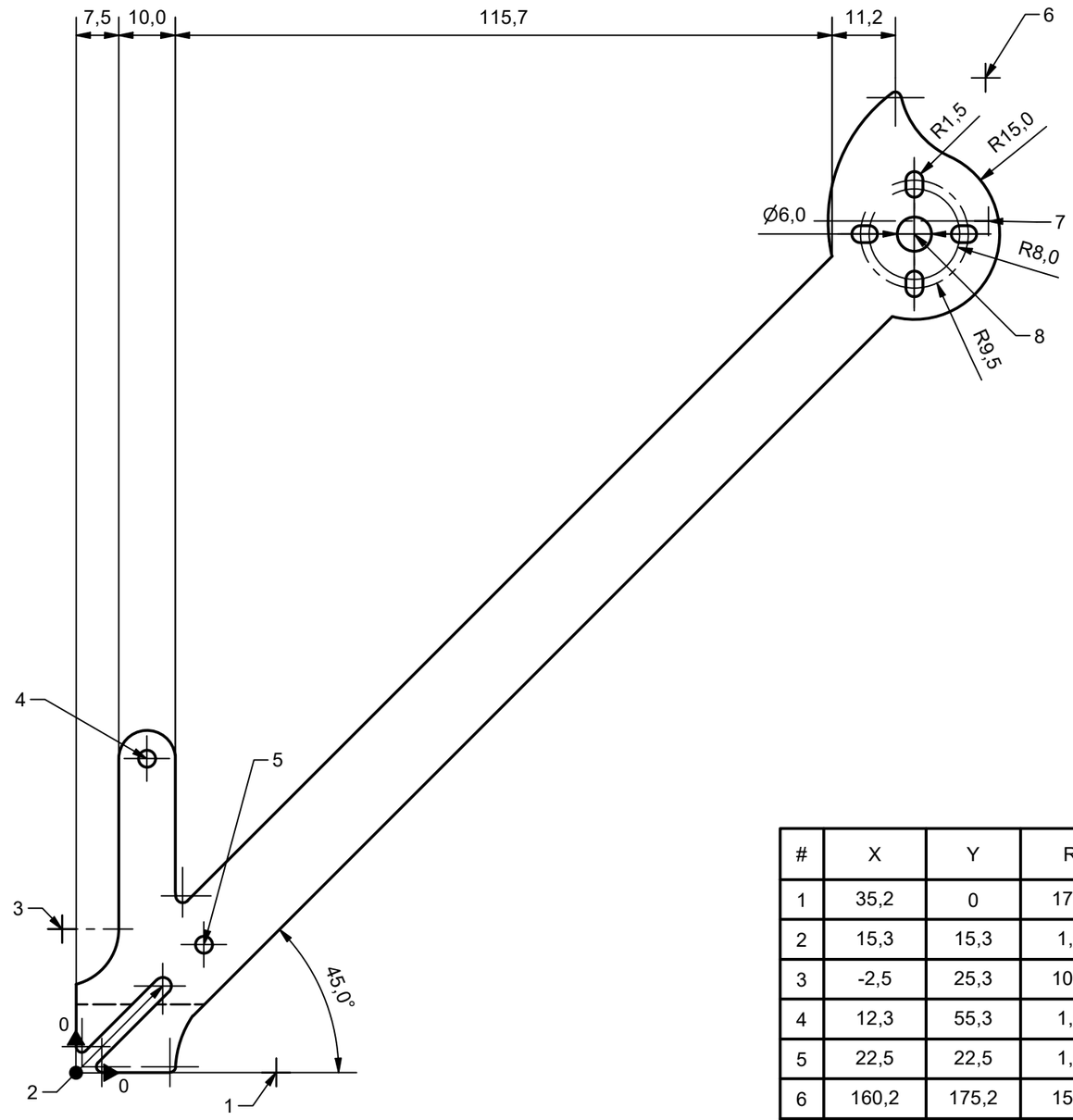


#	X	Y	R
1	35,2	0	17,7
2	15,3	15,3	1,5
3	-2,5	25,3	10,0
4	12,3	55,3	1,5
5	22,5	22,5	1,5
6	160,2	175,2	15,3
7	160,7	149,9	28,2
8	147,7	147,7	-

NOTA: Los redondeos no especificados tienen diámetro: 1mm

TRATAMIENTO: SIN TRATAMIENTO		UIDE	INGENIERIA MECATRÓNICA		
RECUBRIMIENTO: SIN RECUBRIMIENTO					
MATERIAL: CFRP	TOL. GRAL ± 0.1	ESCALA: 1 : 1	DIB.	CUESTA S.	20/03/2022
			DIS.	CUESTA S.	15/02/2022
			REV.	OSCULLO C.	10/04/2022
BRAZO 1		D03 - 201			

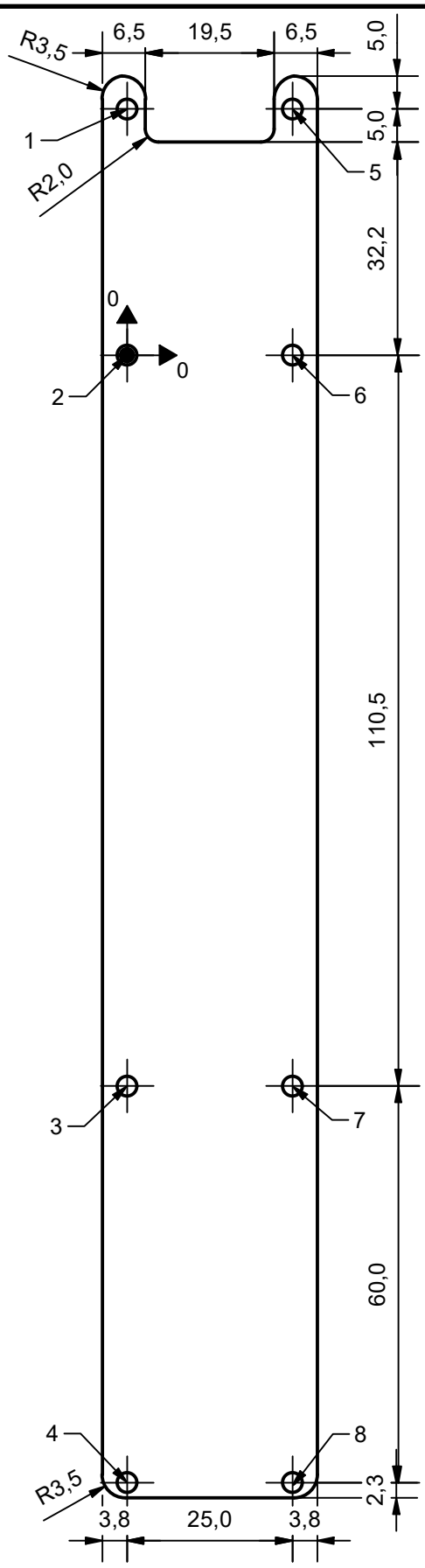




#	X	Y	R
1	35,2	0	17,7
2	15,3	15,3	1,5
3	-2,5	25,3	10,0
4	12,3	55,3	1,5
5	22,5	22,5	1,5
6	160,2	175,2	15,3
7	160,7	149,9	28,2
8	147,7	147,7	-

NOTA: Los redondeos no especificados tienen diámetro: 1mm

TRATAMIENTO: SIN TRATAMIENTO		UIDE	INGENIERIA MECATRÓNICA
RECUBRIMIENTO: SIN RECUBRIMIENTO			
MATERIAL: CFRP	TOL. GRAL ± 0.1	ESCALA: 1 : 1	DIB. CUESTA S. 20/03/2022
			DIS. CUESTA S. 15/02/2022
			REV. OSCULLO C. 10/04/2022
BRAZO 2		D03 - 202	



N6

#	X	Y	R
1	0,0	32,2	3,0
2	0,0	0,0	3,0
3	0,0	-110,5	3,0
4	0,0	-170,5	3,0
5	25,0	32,2	3,0
6	25,0	0,0	3,0
7	25,0	-110,5	3,0
8	25,0	-170,5	3,0

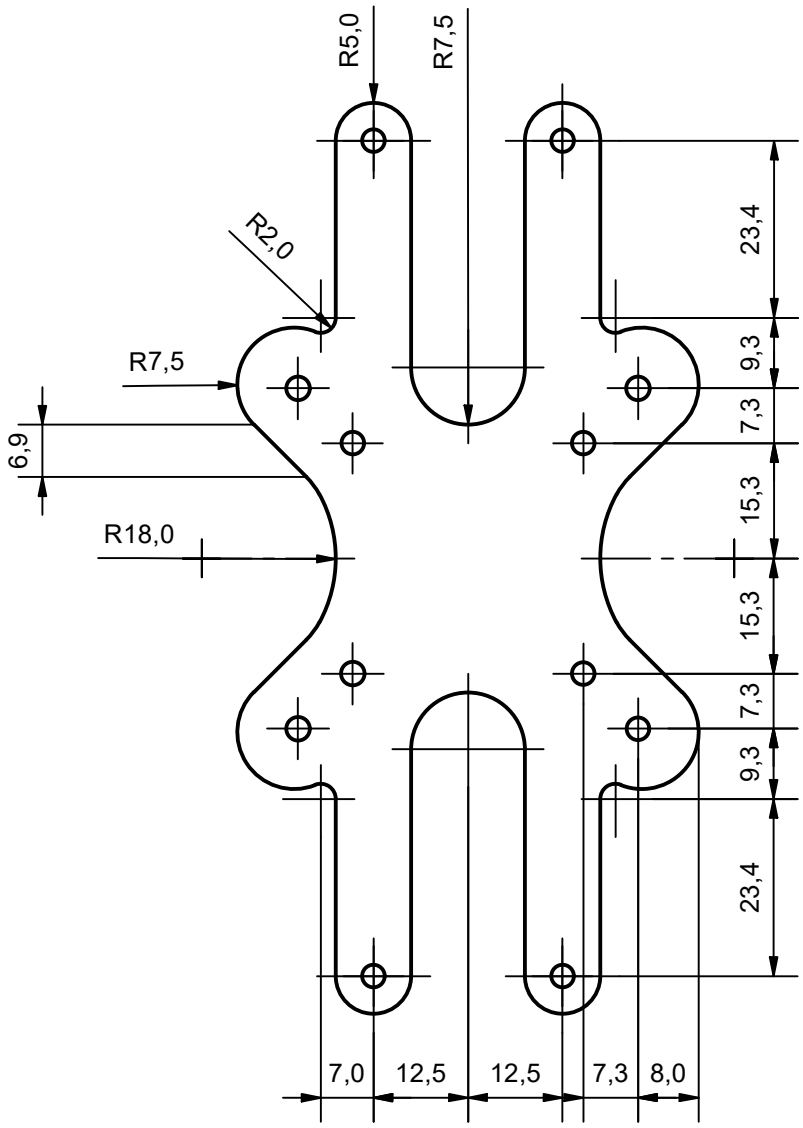
NOTA:
Espesor: 3mm

TRATAMIENTO: SIN TRATAMIENTO		UIDE	INGENIERIA MECATRÓNICA
RECUBRIMIENTO: SIN RECUBRIMIENTO			
MATERIAL: CFRP	TOL. GRAL ± 0.1	ESCALA: 1 : 1	DIB. CUESTA S. 20/03/2022
			DIS. CUESTA S. 15/02/2022
			REV. OSCULLO C. 10/04/2022

PLACA SUPERIOR

D03 - 203

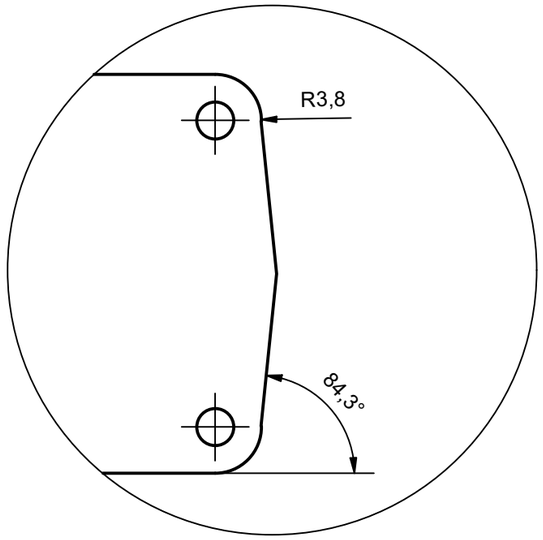
N6



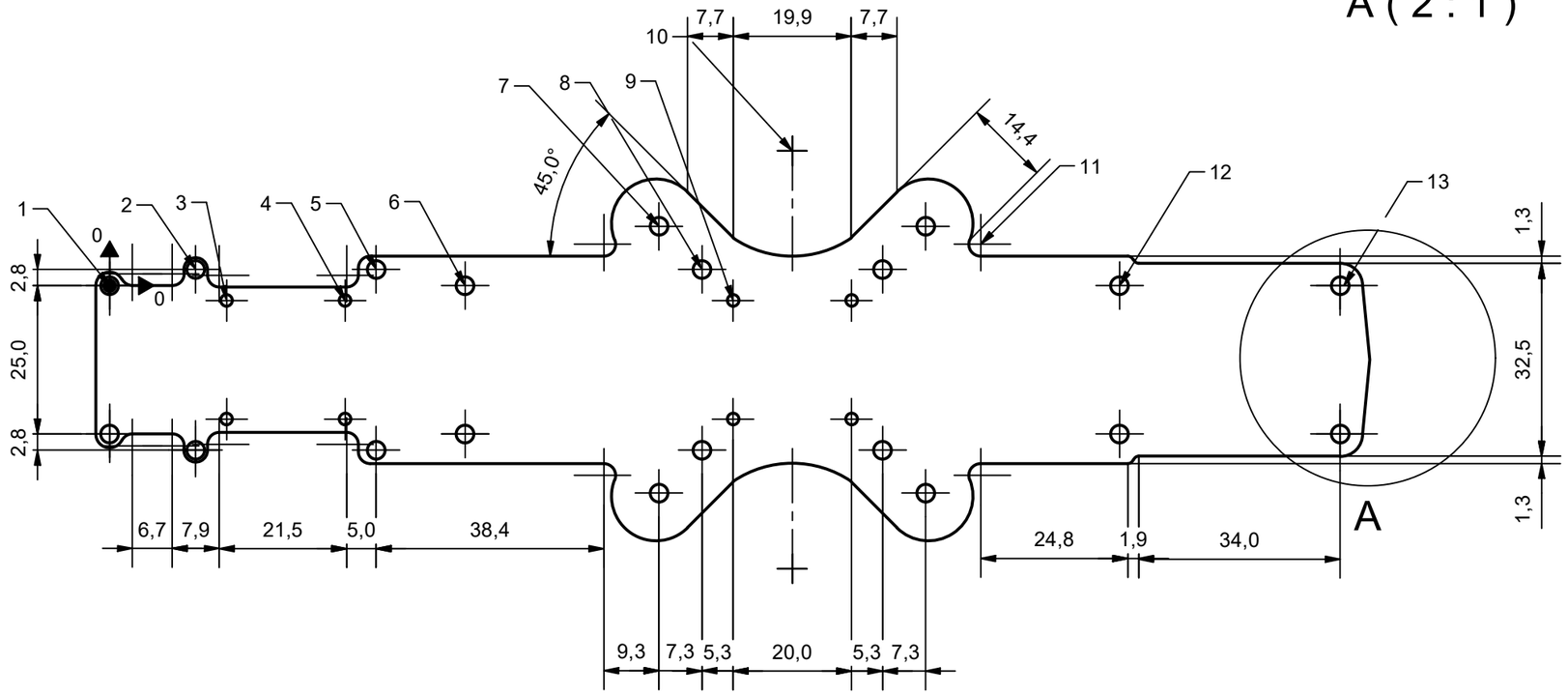
NOTA: todos los agujeros tienen un diametro: 3,2mm

TRATAMIENTO: SIN TRATAMIENTO		UIDE	INGENIERIA MECATRÓNICA			
RECUBRIMIENTO: SIN RECUBRIMIENTO						
MATERIAL: CFRP	TOL. GRAL ± 0.1	ESCALA: 1 : 1	DIB.	CUESTA S.	20/03/2022	
			DIS.	CUESTA S.	15/02/2022	
			REV.	OSCULLO C.	10/04/2022	
PLACA MEDIA		D03 - 204				

N6



A (2 : 1)

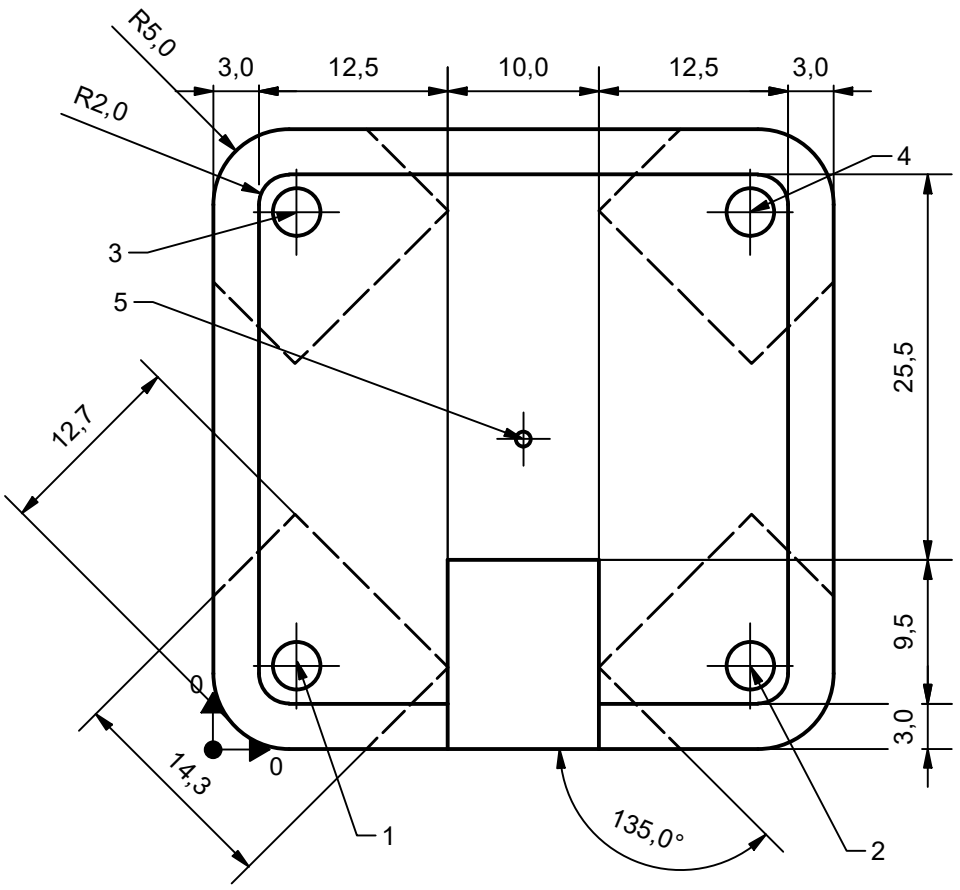


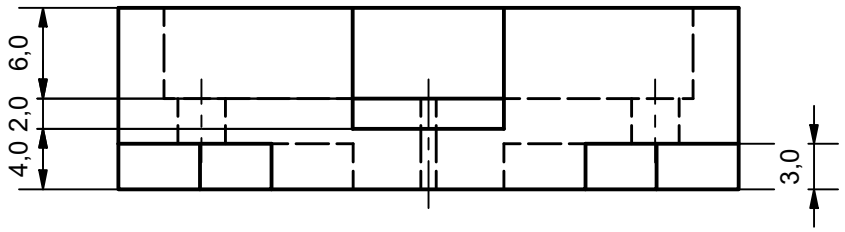
#	X	Y	R
1	0,0	0,0	1,5
2	14,5	2,8	1,5
3	19,8	-2,5	1,0
4	39,8	-2,5	1,5
5	45,0	2,8	1,5
6	60,0	0,0	1,5
7	92,8	10,0	1,5
8	100,0	2,8	1,5
9	105,3	-2,5	1,0
10	115,3	22,7	17,7
11	147,1	7,0	1,5
12	170,5	0,0	1,5
13	207,8	0,0	1,5

NOTA: espesor:3mm
NOTA: simetría en el eje x

TRATAMIENTO: SIN TRATAMIENTO		UIDE	INGENIERIA MECATRÓNICA
RECUBRIMIENTO: SIN RECUBRIMIENTO			
MATERIAL: CFRP	TOL. GRAL ± 0.1	ESCALA: 1 : 1	DIB. CUESTA S. 20/03/2022
			DIS. CUESTA S. 15/02/2022
			REV. OSCULLO C. 10/04/2022
PLACA INFERIOR		D03 - 205	

N7



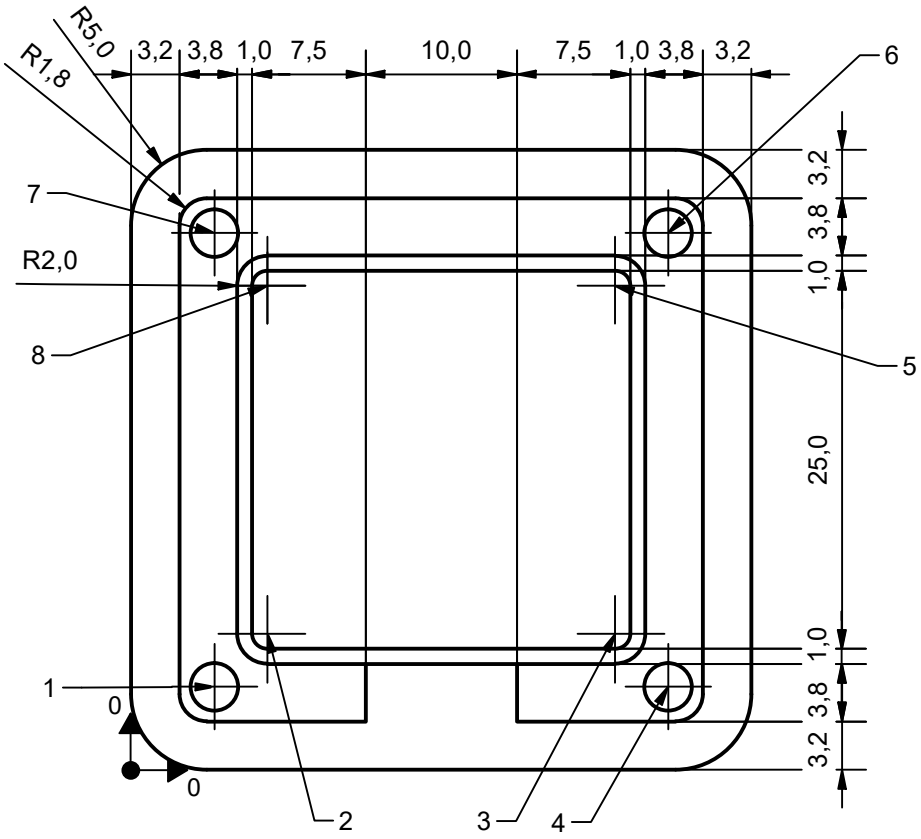


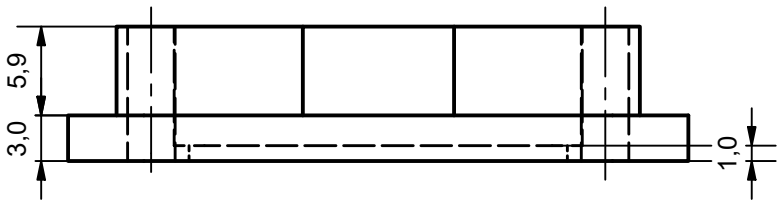
#	X	Y	R
1	5,5	5,5	1,7
2	35,5	5,5	1,7
3	5,5	35,5	1,7
4	35,5	35,5	1,7
5	20,5	20,5	0,5

NOTA: IMPRESIÓN 3D
- Temperatura: 210°C
- Relleno: 10%
- Tipo: Cúbico
- Altura de capa: 0.16mm

TRATAMIENTO: SIN TRATAMIENTO		UIDE	INGENIERIA MECATRÓNICA
RECUBRIMIENTO: SIN RECUBRIMIENTO			
MATERIAL: PLA	TOL. GRAL ± 0.1	ESCALA: 2 : 1	DIB. CUESTA S. 20/03/2022
			DIS. CUESTA S. 15/02/2022
			REV. OSCULLO C. 10/04/2022
CONTENEDOR GPS		D03 - 301	

N7

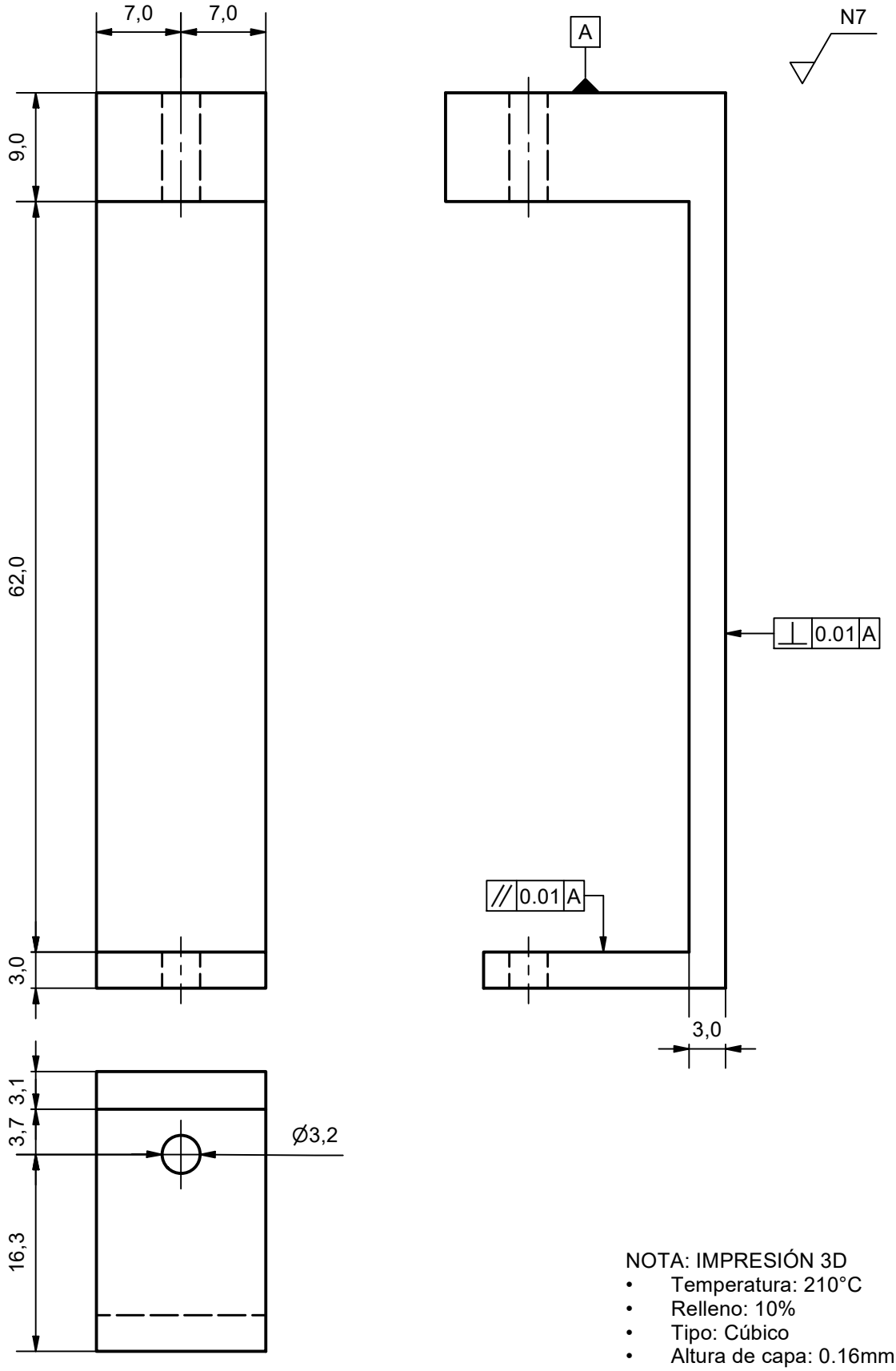




#	X	Y	R
1	5,5	5,5	1,7
2	9,0	9,0	-
3	32,0	9,0	-
4	35,5	5,5	1,7
5	32,0	32,0	-
6	35,5	35,5	1,7
7	5,5	35,5	1,7
8	9,0	32,0	-

NOTA: IMPRESIÓN 3D
- Temperatura: 210°C
- Relleno: 10%
- Tipo: Cúbico
- Altura de capa: 0.16mm

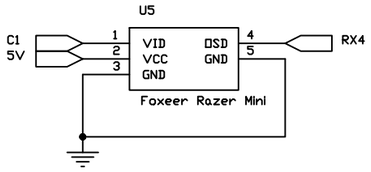
TRATAMIENTO: SIN TRATAMIENTO		UIDE	INGENIERIA MECATRÓNICA
RECUBRIMIENTO: SIN RECUBRIMIENTO			
MATERIAL: PLA	TOL. GRAL ± 0.1	ESCALA: 2 : 1	DIB. CUESTA S. 20/03/2022
			DIS. CUESTA S. 15/02/2022
			REV. OSCULLO C. 10/04/2022
TAPA GPS		D03 - 101	

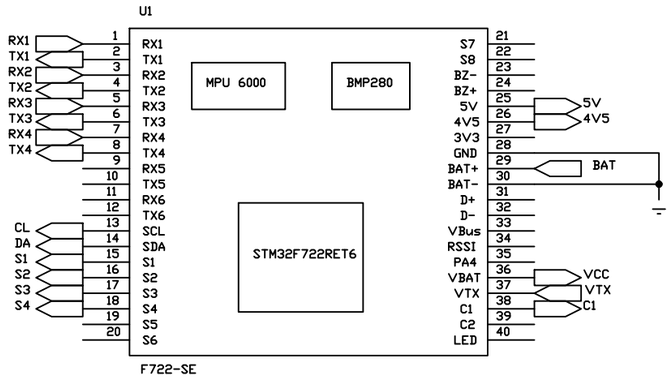


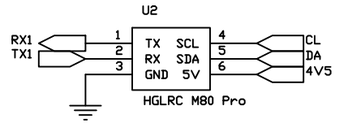
NOTA: IMPRESIÓN 3D
- Temperatura: 210°C
- Relleno: 10%
- Tipo: Cúbico
- Altura de capa: 0.16mm

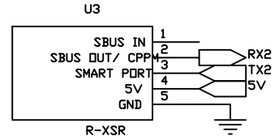
TRATAMIENTO: SIN TRATAMIENTO		UIDE	INGENIERIA MECATRÓNICA			
RECUBRIMIENTO: SIN RECUBRIMIENTO						
MATERIAL: PLA	TOL. GRAL ± 0.1	ESCALA: 2 : 1	DIB.	CUESTA S.	20/03/2022	
			DIS.	CUESTA S.	15/02/2022	
			REV.	OSCULLO C.	10/04/2022	
BRAZO GPS		D03 - 303				

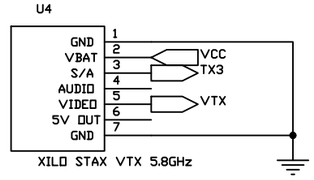
Anexo F: Planos electrónicos

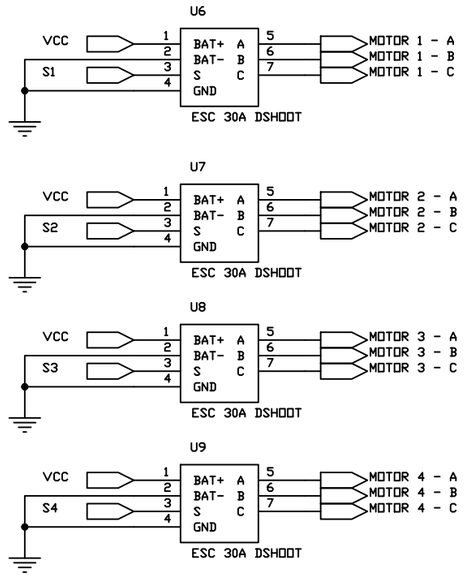


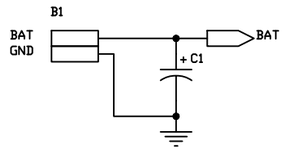


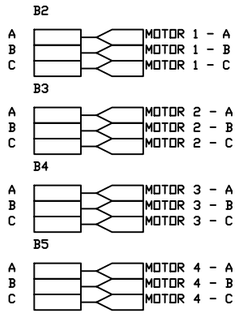












B1 FUENTE DE ALIMENTACIÓN	
BAT	12V
GND	GROUND 0V

UIDE	INGENIERÍA MECATRÓNICA	DIB.	CUESTA S.	20-08-2022
		DIS.	CUESTA S.	15-08-2022
		REV.	OSCULLO C	30-08-2022
DRON DE CARGA		D02-001		ESCALA N/A
PLANO ELECTRÓNICO				N/A

POS.	DESCRIPCIÓN	CANT.	OBSERVACIONES
	ALIMENTACIÓN		
B1	BATERIA LIPO 3S 2200mAh	1	ALIMENTACIÓN PRINCIPAL
	BORNERAS		
B2-B5	MOTORES BRUSHLESS 2212 1000KV	4	MOVIMIENTO DE LAS PROPELAS
	CAPACITORES		
C1	ELECTROLÍTICO 470uf	1	REDUCCIÓN DEL RIZADO
	MÓDULOS		
U1	MÓDULO F722-SE	1	MICROCONTROLADOR
U2	HGLRC M80 Pro	1	GPS Y MAGNETÓMETRO
U3	R-XSR	1	RECEPTOR DE RADIO FRECUENCIA
U4	XILO STAX VTX 5.8GHz	1	TRANSMISOR DE VIDEO ANALÓGICO
U5	Foxeer Razer Mini	1	CÁMARA ANALOGICA CMOS
	DRIVERS		
U6-U9	ESC 30A DSHOT	4	CONTROL MOTOR BRUSHLESS

UIDE

INGENIERÍA MECATRÓNICA

DIB.	CUESTA S.	20-08-2022	
DIS.	CUESTA S.	15-20-2022	
REV.	OSCULLO C.	30-08-2022	

DRON DE CARGA

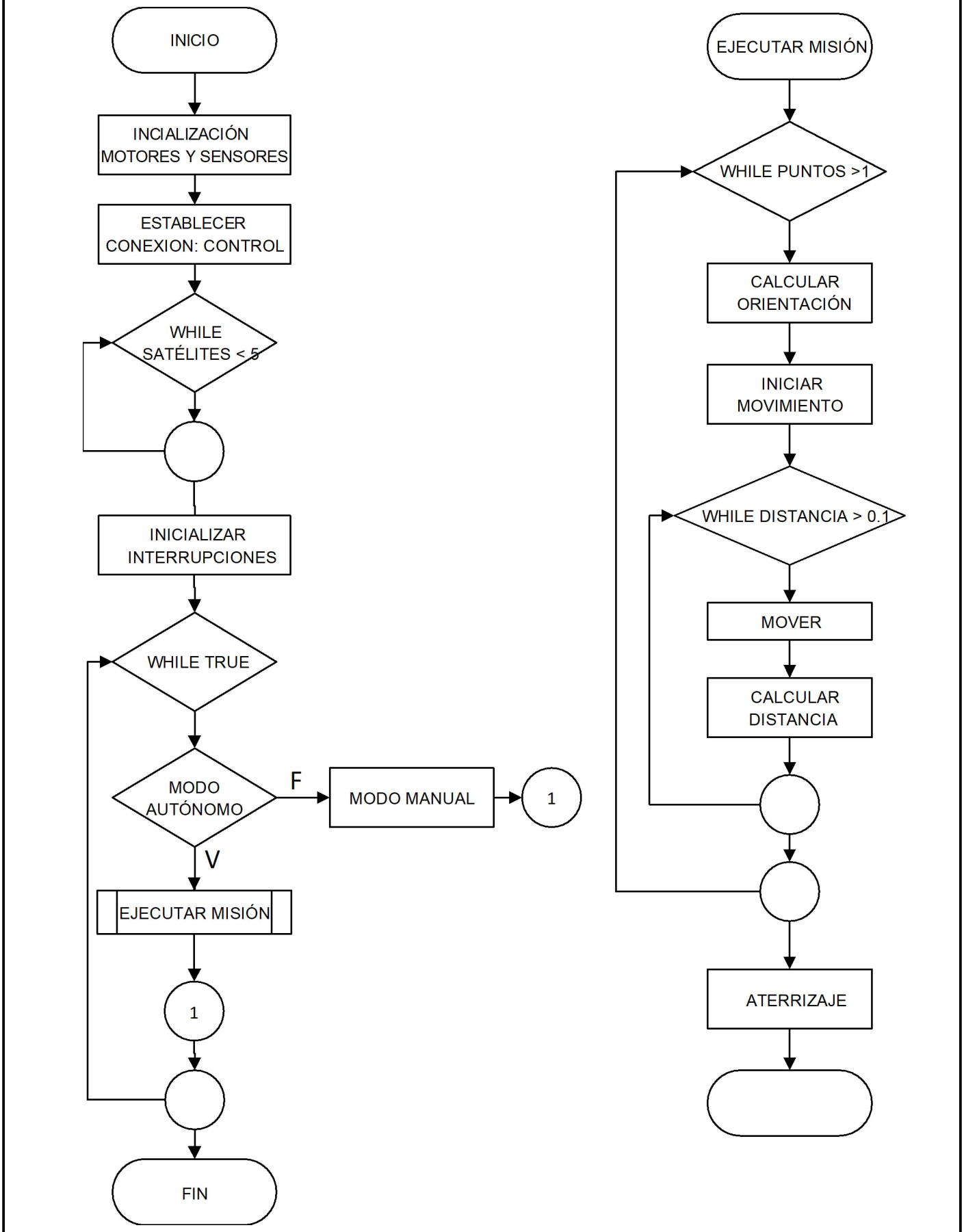
LISTA DE COMPONENTES

D02-101

ESCALA N/A

N/A

Anexo G: Planos informáticos



UIDE	INGENIERÍA MECATRÓNICA	DIB.	CUESTA S.	23-08-2022	
		DIS.	CUESTA S.	10-08-2022	
		REV.	OSCULLO C.	30-08-2022	

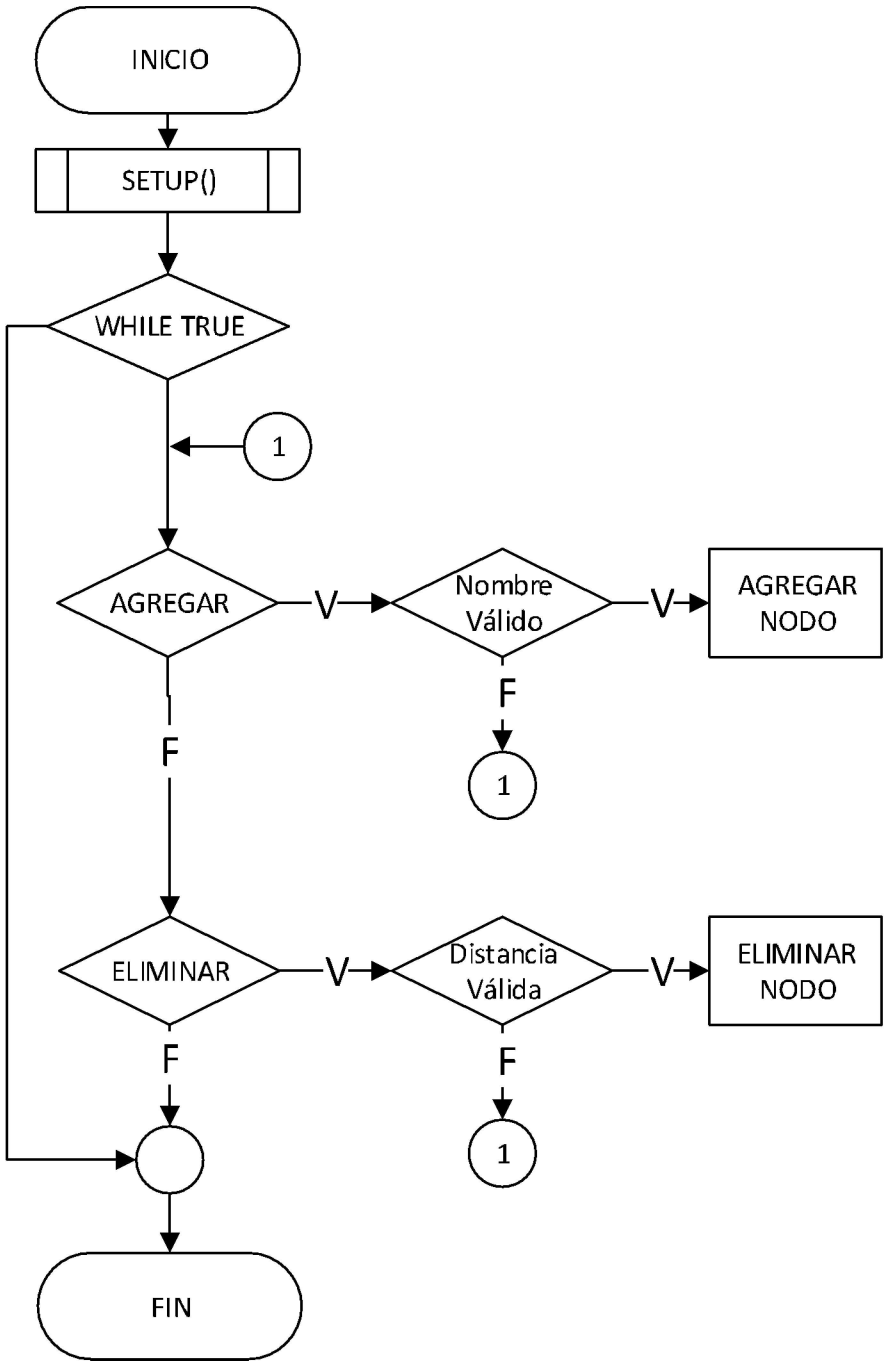
SISTEMA DE TRANSPORTE DE CARGAS DRON DE CARGA

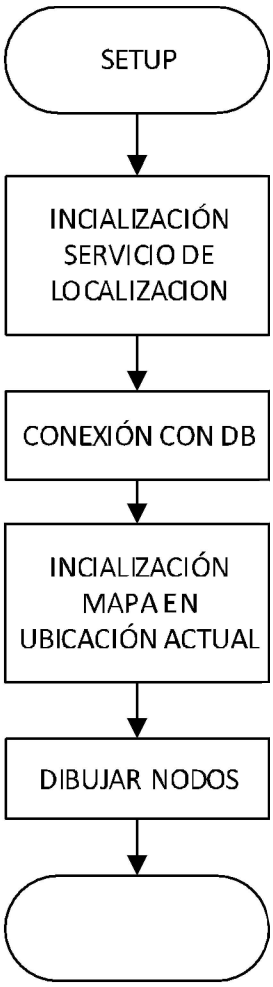
DIAGRAMA DE FLUJO

D01-001

ESCALA N/A

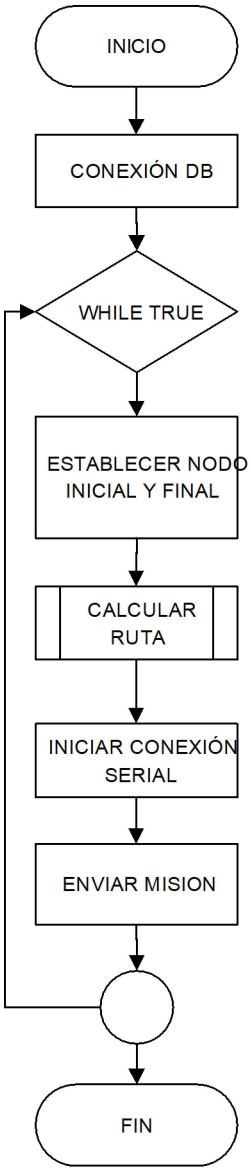
N/A

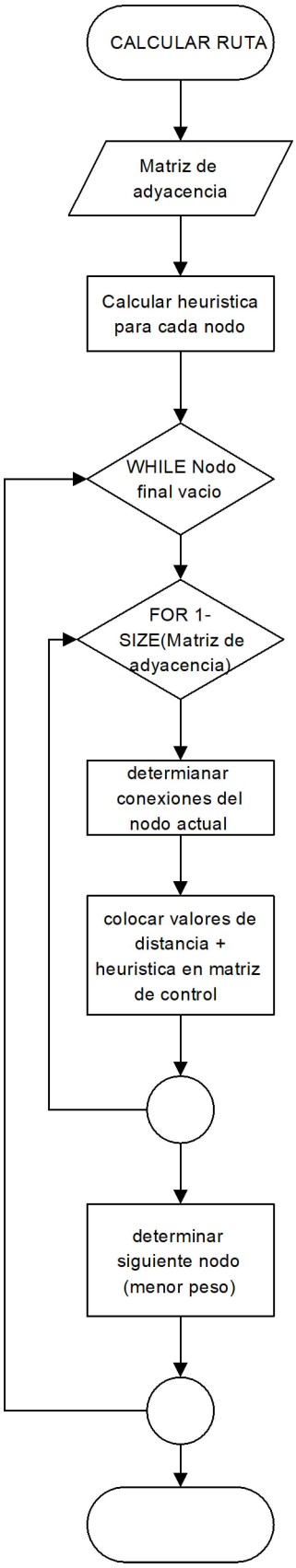




UIDE	INGENIERÍA MECATRÓNICA	DIB.	CUESTA S.	23-08-2022	
		DIS.	CUESTA S.	10-08-2022	
		REV.	OSCULLO C.	30-08-2022	

SISTEMA DE TRANSPORTE DE CARGAS APLICACIÓN MOVIL DIAGRAMA DE FLUJO	D01-002	ESCALA N/A N/A





UIDE	INGENIERÍA MECATRÓNICA	DIB.	CUESTA S.	23-08-2022
		DIS.	CUESTA S.	10-08-2022
		REV.	OSCULLO C.	30-08-2022

SISTEMA DE TRANSPORTE DE CARGAS SOFTWARE PC

DIAGRAMA DE FLUJO

D01-003

ESCALA N/A

N/A

Anexo H: Manual de usuario

MANUAL DE OPERACIÓN Y MANTENIMIENTO

SISTEMA DE TRANSPORTE DE CARGAS MEDIANTE LA UTILIZACIÓN DE DRONES

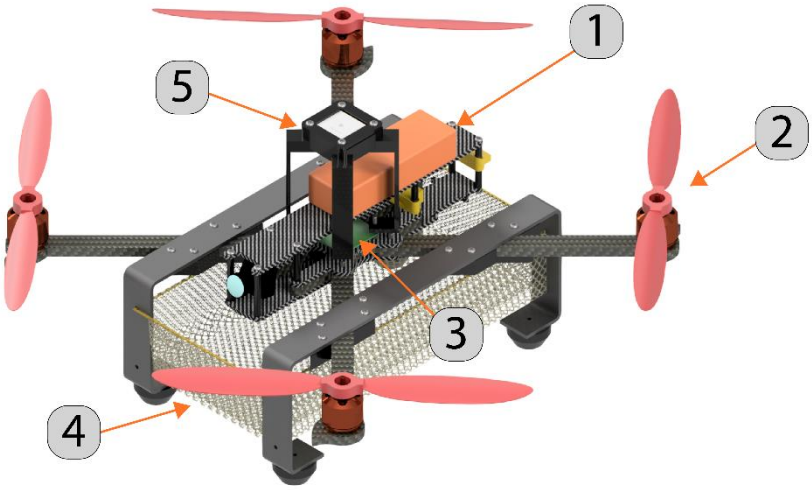
Facultad de Ciencias Técnicas
Escuela de Ingeniería Mecatrónica

UIDE
Powered by
Arizona State University

ESCUELA DE INGENIERÍA MECATRÓNICA

ÍNDICE

PARTES DEL PROTOTIPO



1	BATERÍA

- Batería recargable tipo LiPo 3s de 2200mah de capacidad

2	MOTOR + PROPELA

- Motor 2212 1000Kv + propela 1045

3	MÓDULO MICROCONTROLADOR

- Mateksys F722-SE

4	CANASTA DE CARGA

- Capacidad máxima: 500 g / 1000 cm3

5	GPS + MAGNETÓMETRO

- módulo M80 Pro

6	MANDO DE RADIOFRECUENCIA

- Protocolo D16 ACCESS 2.4 GHz

NOTA: El Sistema debe ser operado utilizando un dispositivo capaz de recibir video analógico de 5.8GHz

ESPECIFICACIONES TÉCNICAS

Batería

- Lipo 3S 2200mah

Peso

- 1.134 kg

Tamaño del prototipo

- 475.8 X 475.8 X 199.2 mm

Capacidad de carga máxima

- 500 g
- 1000 cm3

Conexión de radiofrecuencia

- 2.4GHz D16 ACCESS

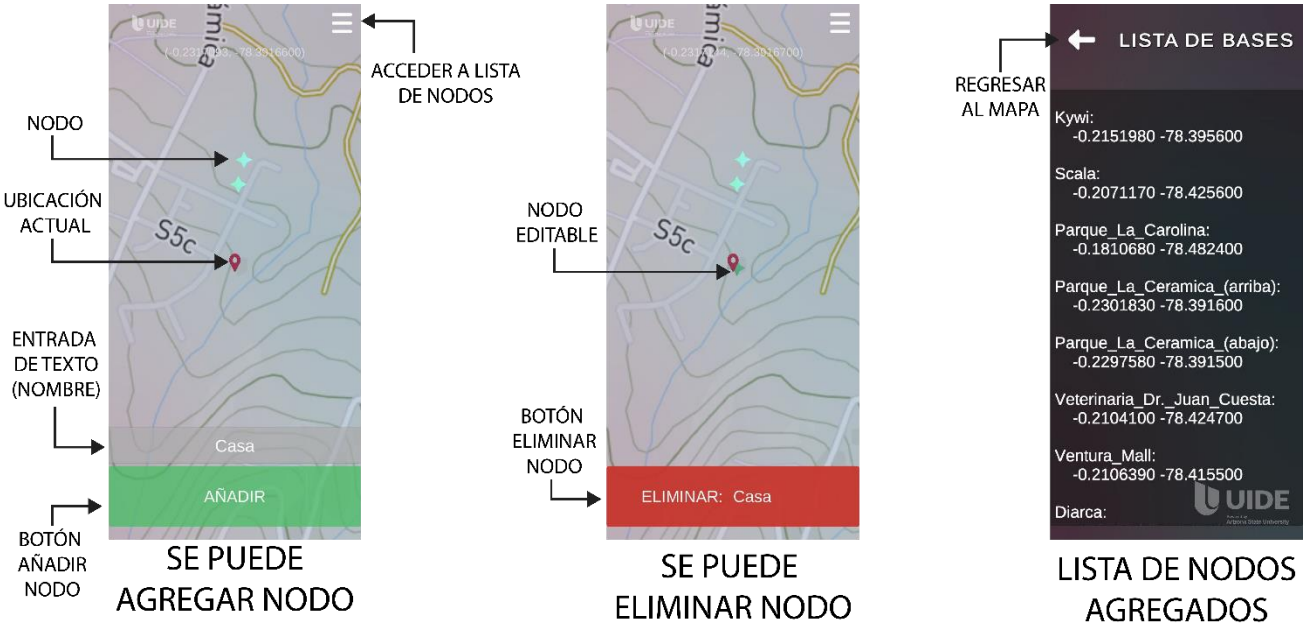
Video Analógico

- 5.8GHz
- Canales:

	1	2	3	4	5	6	7	8
A	5865	5845	5825	5805	5785	5765	5745	5725
B	5733	5752	5771	5790	5809	5828	5847	5866
E	5705	5685	5665	5645	5885	5905	5925	5945
F	5740	5760	5780	5800	5820	5840	5860	5880
R	5658	5695	5732	5769	5806	5843	5880	5917

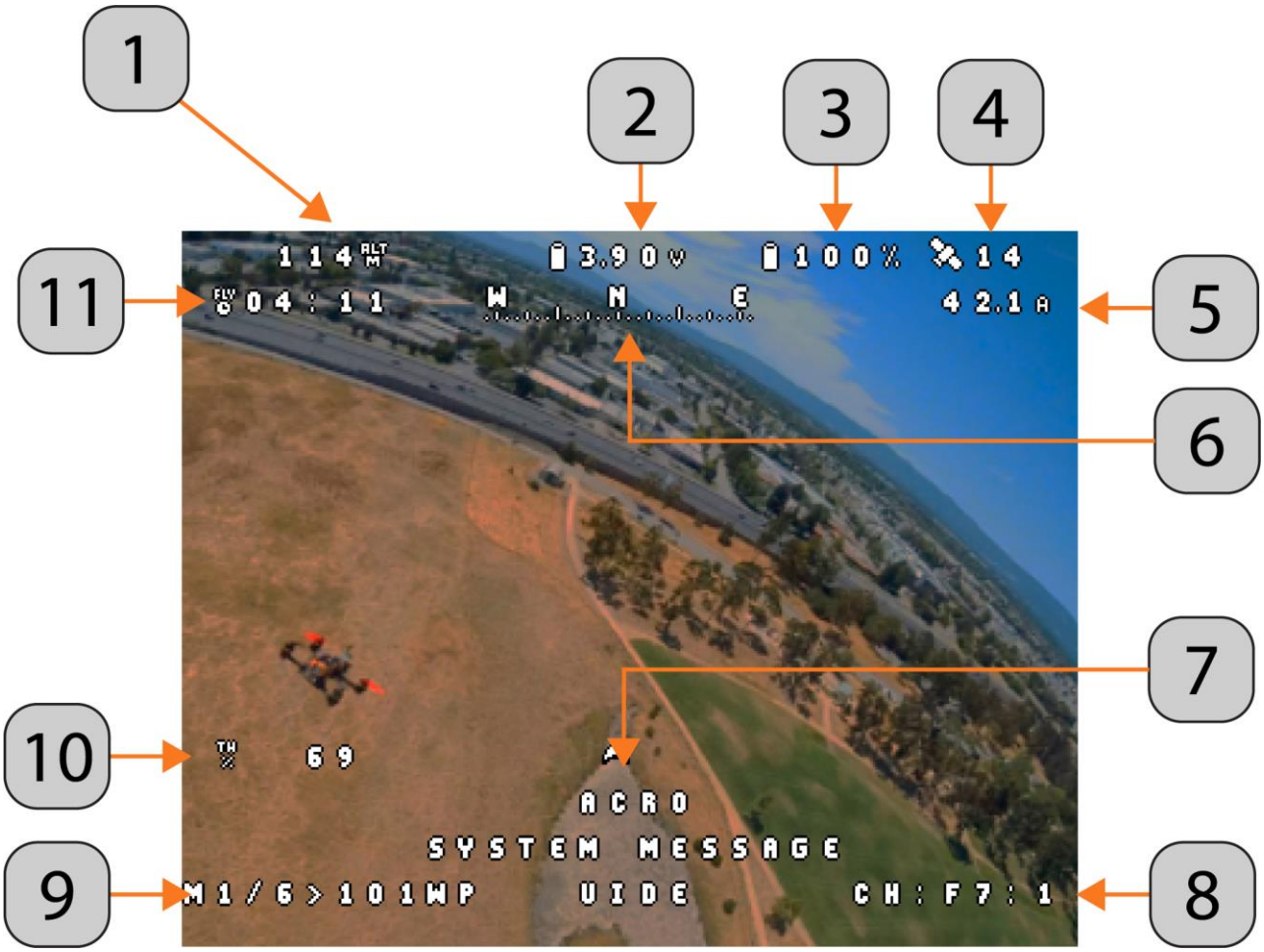
DESCRIPCIÓN DE LAS INTERFACES GRÁFICAS

APLICACIÓN MÓVIL



PROGRAMA DE GENERACIÓN DE RUTAS

DESCRIPCIÓN DEL OSD (OVER SCREEN DISPLAY)

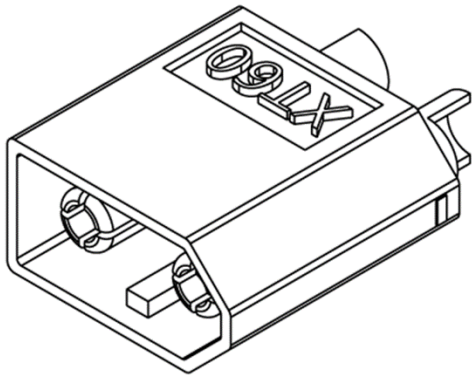


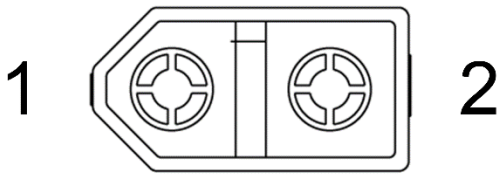
1	Altura medida por el barómetro
2	Voltaje promedio de las celdas
3	Porcentaje de carga de la batería
4	Numero de satélites del sistema GNSS
5	Consumo de corriente actual
6	Brújula
7	Modo de vuelo
8	Canal y potencia del transmisor de video analógico
9	Información de la misión actual
10	Porcentaje de aceleración
11	Tiempo de vuelo/encendido

BATERÍA

CONEXIÓN

El prototipo utiliza un conector XT60 para la conexión de la batería la polaridad del conector se muestra a continuación





1 – NEGATIVO 2 – POSITIVO

CARGA Y ALMACENAMIETO

La carga por cada celda de la batería no debe sobrepasar los 4.2 V ni bajar de los 3.3 V, se recomienda el uso de cargadores especializados que permitan carga “BALANCEADA”.

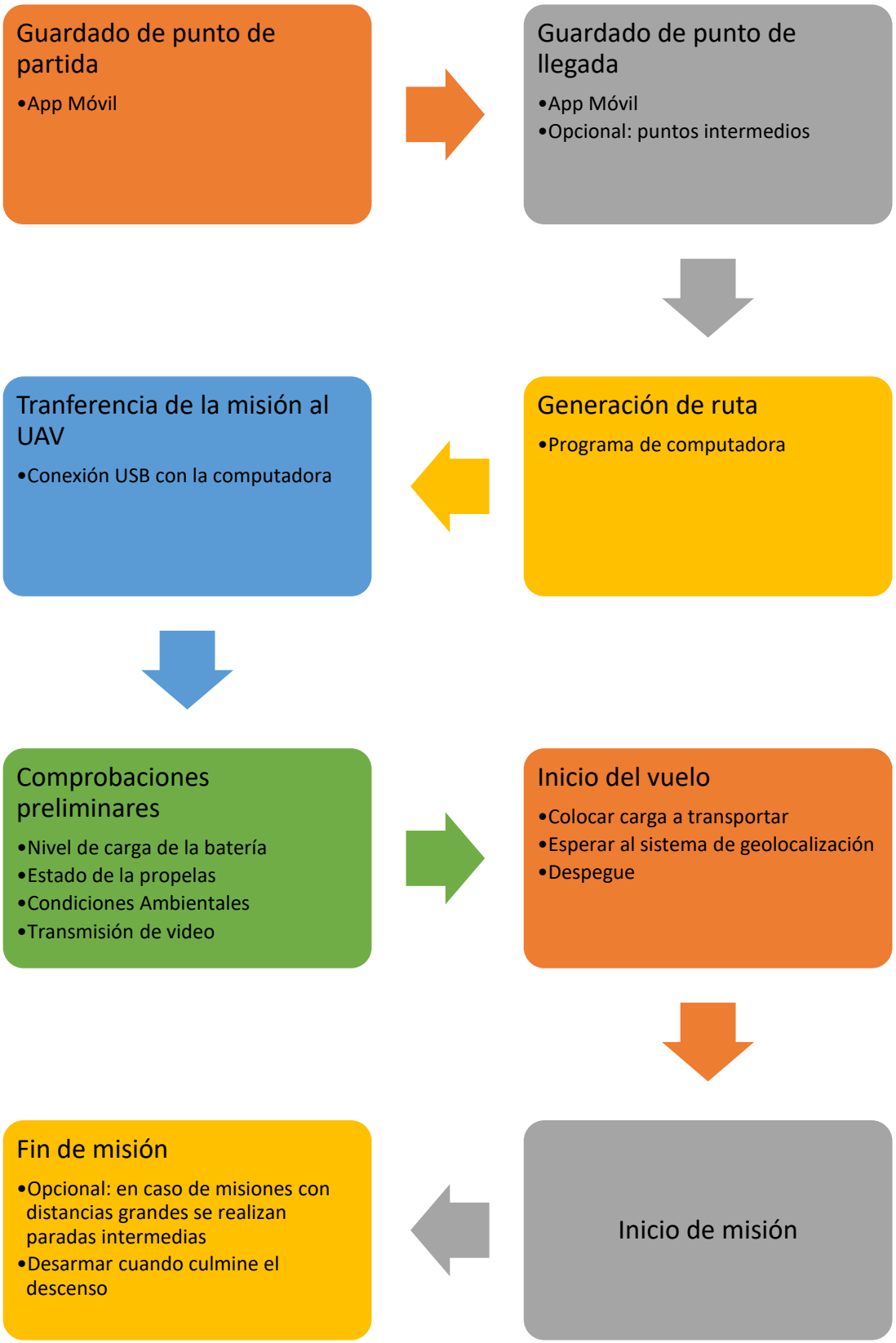
El voltaje de almacenamiento debe ser de 3.8 V para evitar daños en la batería.

EL PRESENTE DOCUMENTO MUESTRA SUGERENCIAS DE VOLTAJES DE OPERACIÓN Y ALMACENAMIENTO SIN EMBARGO SE DEBE CONSULTAR POR VALORES ESPECÍFICOS EN EL MANUAL DEL FABRICANTE DE LA BATERÍA

CONFIGURACIÓN DEL MANDO DE RADIOFRECUENCIA

1	Botón de encendido/apagado
2	Selección de modo de vuelo (Manual, Pos hold, Misiones)
3	Activar "Mantener Altura"
4	Prearmado(Derecha)
5	Pantallla LCD
6	Switch de armado
T	Aceleracion
Y	Yaw (Rotación en Z)
P	Pitch (Rotación en Y)
R	Roll (Rotación en X)

EJECUCIÓN DE LAS MISIONES



MANTENIMIENTO

Las propelas y las antenas son las partes más propensas a sufrir daños por lo que se les debe dar mantenimiento de forma regular, en caso de que las antenas hayan sufrido daños se deben reemplazar antes de proceder con el vuelo.

Las propelas sufren desgaste en cada vuelo por lo que se recomienda hacer un cambio cada 20-30 vuelos o si existe evidencia de deterioro de estas, si el UAV ha sido parte de una colisión se debe realizar una inspección visual de las hélices y en caso de ser necesario reemplazarlas, a continuación, se muestra el orden de ensamblaje de las hélices:

Nota: Si es necesario reemplazar las antenas de recepción de control de radiofrecuencia o componentes internos se debe contactar directamente con el fabricante.

ADVERTENCIAS

No operar el prototipo en condiciones ambientales adversas (Lluvia).

No encender los motores en interiores o cerca de personas.

No encender el prototipo sin conectar la antena del módulo de transmisión de video analógico.

No realizar operaciones de vuelo si las hélices se encuentran en mal estado

No operar el prototipo si los motores muestran signos de sobrecalentamiento

No se deben iniciar acciones de vuelo autónomo si no se cuenta con una recepción clara de la señal de video analógico.

No apagar si el mando de radiofrecuencia si el prototipo se encuentra encendido

RECOMENDACIONES

Realizar el proceso de detección de los satélites GNSS con una batería diferente a que será utilizada en el vuelo ya que este proceso puede durar unos minutos

Si el dron no se encuentra en vuelo mantener la potencia de la transmisión de video al mínimo para evitar daños en el módulo.

No encender el prototipo si existen otros UAVs o dispositivos de transmisión/ recepción de video analógico para evitar interferencia.

Para procesos de calibración, mantenimiento y transporte se recomienda remover las hélices.