DEPARTAMENTO DE CIENCIAS DE LA ENERGÍA Y MECÁNICA

CARRERA DE INGENIERÍA MECATRÓNICA

TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN MECATRÓNICA

TEMA: "DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE PILOTO INTELIGENTE PARA UN VEHÍCULO AÉREO NO TRIPULADO QUE PERMITA EVADIR OBSTÁCULOS, PARA LA GENERACIÓN DE UN ORTOMOSAICO, CON EL FIN DE MONITOREAR ÁREAS DE DIFÍCIL ACCESO"

AUTORES: NOBOA CASTRO, ERICK ALEXANDER
CAMPOS CAMPOS, CRISTHIAN JESÚS

DIRECTOR: Ing. CÓRDOVA CRUZATTY, ANDREA CONCEPCIÓN

LATACUNGA

2018

DEPARTAMENTO DE CIENCIAS DE LA ENERGÍA Y MECÁNICA

CARRERA DE INGENIERÍA MECATRÓNICA

CERTIFICACIÓN

Certifico que el trabajo de titulación, **"DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE PILOTO INTELIGENTE PARA UN VEHÍCULO AÉREO NO TRIPULADO QUE PERMITA EVADIR OBSTÁCULOS, PARA LA GENERACIÓN DE UN ORTOMOSAICO, CON EL FIN DE MONITOREAR ÁREAS DE DIFÍCIL ACCESO."** fue realizado por los señores **NOBOA CASTRO ERICK ALEXANDER y CAMPOS CAMPOS CRISTHIAN JESÚS**, el mismo que ha sido revisado en su totalidad, analizado por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Latacunga, 02 de agosto del 2018

Andrea Córdova

Ing. CORDOVA ANDREA

C.C.1803797222

DEPARTAMENTO DE CIENCIAS DE LA ENERGÍA Y MECÁNICA

CARRERA DE INGENIERÍA MECATRÓNICA

AUTORÍA DE RESPONSABILIDAD

Nosotros, **NOBOA CASTRO, ERICK ALEXANDER y CAMPOS CAMPOS, CRISTHIAN JESÚS**, declaramos que el contenido, ideas y criterios del trabajo de titulación: **"DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE PILOTO INTELIGENTE PARA UN VEHÍCULO AÉREO NO TRIPULADO QUE PERMITA EVADIR OBSTÁCULOS, PARA LA GENERACIÓN DE UN ORTO-MOSAICO, CON EL FIN DE MONITOREAR ÁREAS DE DIFÍCIL ACCESO."** es de nuestra autoría y responsabilidad, cumpliendo con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Consecuentemente el contenido de la investigación mencionada es veraz.

Latacunga, 02 de agosto del 2018

NOBOA CASTRO ERICK ALEXANDER
C.C: 1805223599

CAMPOS CAMPOS CRISTHIAN JESÚS
C.C: 0703029538

DEPARTAMENTO DE CIENCIAS DE LA ENERGÍA Y MECÁNICA

CARRERA DE INGENIERÍA MECATRÓNICA

AUTORIZACIÓN

Nosotros, **NOBOA CASTRO, ERICK ALEXANDER y CAMPOS CAMPOS, CRISTHIAN JESÚS**, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación **"DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE PILOTO INTELIGENTE PARA UN VEHÍCULO AÉREO NO TRIPULADO QUE PERMITA EVADIR OBSTÁCULOS, PARA LA GENERACIÓN DE UN ORTOMOSAICO, CON EL FIN DE MONITOREAR ÁREAS DE DIFÍCIL ACCESO."** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de nuestra responsabilidad.

Latacunga, 02 de agosto del 2018

.....

NOBOA CASTRO ERICK ALEXANDER
C.C: 1805223599

.....

CAMPOS CAMPOS CRISTHIAN JESÚS
C.C: 0703029538

DEDICATORIA

Este proyecto está dedicado a mis padres y hermanos por haberme apoyado incondicionalmente en todo momento, por sus consejos, por sus ejemplos y motivación para poder culminar con esta dura caminata educativa.

Cristhian Jesus Campos Campos

DEDICATORIA

Para Anabel, tu paciencia y cariño me sostuvo durante todo este andar.

Erick Alexander Noboa Castro

AGRADECIMIENTO

En primer lugar, le agradezco a Dios por haberme protegido en todo este tiempo y derramado tantas bendiciones, después a mi familia, mi madre Nancy Campos, mi padre Francisco Campos, mis hermanos y hermanas; los cuales son el pilar fundamental en mi vida.

Cristhian Jesus Campos Campos

AGRADECIMIENTO

A Dios y a mi familia por darme las herramientas necesarias para poder culminar exitosamente todas las metas propuestas.

Erick Alexander Noboa Castro

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

En el presente proyecto se implementa un sistema de piloto inteligente que permite detectar y evadir obstáculos a un UAV de propulsión eléctrica, con capacidades de despegue y aterrizaje vertical, que incorpora a bordo sensores de alto desempeño, gracias a los cuales puede estabilizarse y maniobrar de manera autónoma, al mismo tiempo que le permiten obtener información del ambiente externo a su alrededor. Por otra parte, se establece una región de interés por donde deberá sobrevolar el quadrotor mientras adquiere información. Una vez definida esta región, es necesario ajustar los parámetros de vuelo para que el algoritmo calcule automáticamente la trayectoria más conveniente. Mientras que el UAV avanza en su recorrido, el sistema de piloto inteligente, el cual es el interés de esta investigación, se encarga de monitorear su entorno con la finalidad de poder advertir cualquier obstáculo y tomar exitosamente alguna acción evasiva evitando colisiones y cuidando su integridad sin comprometer el plan de vuelo establecido. Durante el transcurso de esta trayectoria, una cámara de alta definición, incorporada en el fuselaje del UAV adquirirá imágenes periódicamente, las cuales posteriormente servirán para la construcción de un orto-mosaico de esta zona.

PALABRAS CLAVE:

- **PILOTO INTELIGENTE**
- **MAPEO 3D**
- **DRONES**
- **VEHÍCULO AEREO NO TRIPULADO**

ABSTRACT

In the present project, an intelligent pilot system is implemented to detect and evade obstacles to an electric propulsion UAV, with vertical take-off and landing capabilities, which incorporates high-performance sensors on board, thanks to which it can be stabilized and maneuvering autonomously, at the same time that they allow him to obtain information from the external environment around him. On the other hand, a region of interest is established where you must fly over the quadrotor while acquiring information. Once this region is defined, it is necessary to adjust the flight parameters so that the algorithm automatically calculates the most convenient trajectory. While the UAV moves forward, the intelligent pilot system, which is the focus of this investigation, is responsible for monitoring its environment in order to be able to notice any obstacle and successfully take some evasive action avoiding collisions. and taking care of their integrity without compromising the established flight plan. During the course of this trajectory, a camera of high definition, incorporated in the fuselage of the UAV will acquire images periodically, which will later be used for the construction of an ortomosaic in this area.

KEYWORDS:

- **INTELLIGENT PILOT**
- **3D MAPING**
- **DRONES**
- **UNMANNED AERIAL VEHICLE**

CAPÍTULO I

MARCO METODOLOGICO DE LA INVESTIGACIÓN

1.1 Introducción

En el presente proyecto se desarrolla un sistema de piloto inteligente que tenga la capacidad de volar autónomamente y evadir obstáculos empleado en un vehículo aéreo no tripulado. En este caso una de las características más relevantes es el uso de un algoritmo de control, el cual estará encargado de mantener estabilizada la aeronave, además de la detección y evasión de obstáculos con ayuda de ROS (Robot Operating System), como interfaz de comunicación y procesamiento de control para el vehículo aéreo no tripulado.

1.2 Antecedentes

Los vehículos aéreos no tripulados, UAV´s por sus siglas en inglés (Vehículos aéreos no tripulados), han sido motivo de un sin número de investigaciones dada su versatilidad, con aplicaciones que pueden ir desde un ámbito meramente recreacional, como grabar o adquirir imágenes de eventos sociales, hasta requerimientos militares como: monitoreo, vigilancia, seguridad o aplicaciones de espionaje en zonas de interés.

En relación con el desarrollo de aplicaciones, uno de los temas de la Conferencia latinoamericana de sistemas remotamente tripulados UVEX américa 20116, fue el de “Simulación de inteligencia de vehículos aéreos no tripulados (UAV) para supervivencia

militar" donde los autores diseñaron un entorno para simular la capacidad de vuelo de un UAV, en este caso el Parrot AR Drone versión 2.0, además de encontrar y obtener la ubicación geográfica de objetivos militares, usa diferentes algoritmos para procesar las imágenes de sus objetivos (Másum M., A. M., 2013) p161. Por otra parte, los Quadrotores y en sí la navegación con este tipo de vehículos se pueden usar en operaciones como de: evaluación de riesgos de trabajo para prevenir accidentes laborales aplicado a la industria de construcción, como lo propone Alfredo Toriz en su publicación. (Toriz, Raygoza, & Martínez, 2017) los cuales emplean reconstrucción 3D para la identificación y reconocimientos previos de riesgos, a fin de probar su modelo de inclusión tecnológica.

Ecuador es un país que, debido a su variabilidad topográfica, posee diferentes escenarios de los que se requiere un monitoreo, como: quebradas, acantilados, montañas, bosques y hasta la frontera ecuatoriana. Actualmente se realiza estas actividades de monitoreo con el uso de equipo y personal especializado, lo cual conlleva a un elevado costo de operación y mantenimiento, sin mencionar que dicho personal está sometido a diferentes situaciones que exponen su vida a un constante riesgo.

En consecuencia, la implementación de un sistema de control de estabilidad para el vuelo inteligente resulta necesario ya que reduce los errores cometidos por el operador, puesto que ayuda a estabilizar el quadrotor de manera independiente (Olea Hernández & Sánchez L., 2014), dándole la capacidad al vehículo aéreo de seguir una ruta preprogramada, aplicado a la seguridad fronteriza o en situaciones de emergencia. Esto se

ha conseguido mediante diferentes algoritmos de control como el que proponen Fernández en su proyecto “diseño e implementación de un sistema de control asistido para plataforma aérea multi-rotor” (Fernández G., 2015) donde estiman la función de transferencia de cada ángulo de giro del vehículo, por lo que se desarrollan tres controladores PID y se envían órdenes mediante el protocolo de comunicación Mavlink a un programa núcleo (firmware: APM-copter) de código abierto . Por otra parte, también se encuentra el control del UAV en base a un sistema de visión, que estima la posición del vehículo en sus seis grados de libertad, tiene la finalidad de controlar la estabilización del sistema y realizar el rastreo del UAV para conseguir trayectorias controladas.

1.3 Planteamiento del Problema

En Ecuador es bastante común encontrarse con situaciones de alto riesgo, ya sean estragos de la naturaleza como deslaves, derrumbes en carreteras, etc. o también consecuencia de alguna imprudencia de la mano del hombre como por ejemplo incendios forestales. son en estas circunstancias donde se requiere de una pronta inspección para poder tomar una acción correctiva acertada. Este trabajo puede simplificarse significativamente si se aplican nuevas tecnologías como: los vehículos aéreos no tripulados, los sistemas de control robustos, la navegación y localización simultanea; que gracias a su pronto desarrollo se encuentran al alcance de nuestras manos.

Durante estas situaciones de alto riesgo mencionadas el entorno se vuelve muy variable por lo que puede ocurrir cualquier imprevisto, esto representa un riesgo para el vuelo de un vehículo aéreo no tripulado ya que el mismo carece de un piloto a bordo

consciente de los hechos alrededor de la aeronave. Es por esta razón que el UAV requiere de un sistema de piloto inteligente que permita la detección y evasión de obstáculos, protegiendo su integridad sin comprometer la misión o el plan de vuelo establecido.

Por otra parte, los orto-mosaicos que existen en la actualidad poseen un nivel de detalle bajo. Esto se puede evidenciar en uno de los ejemplos más conocidos: “GoogleEarth”, estas imágenes poseen un cierto grado de error con respecto a sus puntos en tierra obtenidos mediante un dispositivo GPS. Lo que acentúa la importancia de la obtención de imágenes con mayor resolución o cantidad de detalle que posteriormente serán necesarias para la construcción de un orto-mosaico.

1.4 Justificación

De acuerdo con el ministro del Interior del Ecuador, solo en el año 2014, la Escuadra de Rescate del Grupo de Intervención y Rescate (GIR) ejecutó 24 operaciones relevantes a nivel nacional. De ellos, 12 en alta, media y baja montaña y 12 rescates en zonas verticales (pendientes o quebradas) (Ministerio del Interior, 2014).

La preparación permanente de la escuadra en técnicas de extracción de víctimas, evacuación de personas, manejo de herramientas de cartografía y manejo del Sistema de Posicionamiento Global (GPS), les permite acudir inmediatamente y brindar el servicio a la comunidad, cuando la naturaleza o el mismo ser humano, son causantes de desastres o accidentes (Ministerio del Interior, 2014). Por Estas razones resulta impor-

tante la implementación de un sistema autómata de adquisición de orto-imágenes que permita el monitoreo previo de áreas de difícil acceso.

Por otro lado, la mencionada irregularidad del terreno, además de una gran variedad de flora y fauna, convierten a gran parte de nuestro país en una zona de difícil acceso, con una alta probabilidad de que la ruta pre-establecida en el plan de vuelo sea interrumpida por algún obstáculo durante el recorrido del UAV. En consecuencia, es necesario dotar al vehículo de la capacidad de tomar una acción evasiva frente a cualquiera de estas situaciones, preservando su integridad hasta cumplir con la misión establecida.

Además, los algoritmos para vuelo inteligente en UAV a partir de un modelo matemático, así como la detección y evasión de obstáculos durante el vuelo que se generen en el desarrollo de esta investigación, aportarán significativamente al avance tecnológico del país y al desarrollo de nuevas aplicaciones relacionadas en el campo de la robótica y la comunidad mecatrónica.

1.5 Objetivo

1.5.1 Objetivo general

Desarrollar e implementar un sistema de piloto inteligente para un vehículo aéreo no tripulado, con capacidad de evadir obstáculos, para la generación de un ortomosaico con fines de monitoreo de áreas de difícil acceso.

1.5.2 Objetivos Específicos

- Determinar el modelo matemático del UAV.

- Simular la respuesta del controlador del UAV en un software computacional.
- Desarrollar e implementar un algoritmo para el piloto inteligente del UAV.
- Implementar un algoritmo para evadir obstáculos en un vuelo automático.
- Generar un orto-mosaico de los datos adquiridos por el UAV.
- Presentar los resultados finales.

1.6 Hipótesis

¿El sistema de piloto inteligente con la capacidad de evitar obstáculos para un UAV, permitirá generar un Ortomosaico mediante el uso de software computacional para el monitoreo de zonas de difícil acceso?

1.7 Estado del arte sistemas de vuelos autónomos con vehículos aéreos no tripulados

Este tipo de sistemas se han desarrollado para dotar a un vehículo aéreo no tripulado de la capacidad de despegar y mantener su estabilidad en el aire mientras realiza maniobras de desplazamiento con rutas planificadas o misiones de vuelo.

Por otra parte, y gracias a los avances en la tecnología, desde que se han introducido giroscopios y acelerómetros en controladores con aplicaciones como el guiado de las bombas alemanas en la Segunda Guerra Mundial (Chicaiza C. & Chuchico A., 2015), se han desarrollado sistemas de control en tierra que se encargan del monitoreo de la posición del quadrotor.

Dada la necesidad, lo que se requiere es aproximar la reacción del conjunto motor-hélice, relacionando su señal de actuación con la velocidad de giro (Rico, Maisterra, & Martínez, 2015), con la finalidad de determinar la fuerza aerodinámica y el par de arrastre que produce cada uno de los cuatro motores cuyas variaciones se traducen en el desplazamiento de la aeronave.

Previo al despegue se define un plan de vuelo, donde se establecen los puntos por los que debe sobrevolar el quadrotor y las acciones que este debe tomar en dichos puntos, la desventaja es que el quadrotor suele seguir este plan con muy poca flexibilidad, debido a que no se puede cambiar este plan una vez que se ha puesto en marcha.

Consecuentemente, en la universidad de Alicante en el año 2015, (Alicante, 2015) investigadores han desarrollado un sistema de control que le brinda mayor autonomía al quadrotor, con un elemento de comunicación a tierra dentro del lazo de control, lo que permite que el plan sea alterado en cualquier punto de la ejecución lo que le permite reestructurar sus funciones de acuerdo con la información obtenida por los sensores.

1.8 Investigaciones Realizadas

1.8.1 Sistema de seguridad que detecta personas y realiza el seguimiento con un UAV.

El proyecto realizado por el estudiante Carlos Paucar en la Universidad de las Fuerzas Armadas ESPE-L (Paucar M. & Morales C. , 2017) tiene como objetivo utilizar vi-

sión artificial para controlar el vuelo de un Quadrotor logrando que este se vuelva autónomo permitiendo la detección de personas en un área determinada. Basado en el Sistema Operativo Robótico ROS el cual se encarga de comandar la aeronave.

1.8.2 Generación de mosaicos georreferenciados

El proyecto realizado por el estudiante Kevin Vásquez en la Pontificia Universidad Javieriana (Vasquez L., 2014), presenta un método con bases matemáticas que permite generar y georreferenciar una orto imagen a partir de datos obtenidos por un vehículo aéreo no tripulado, el cual obtiene un fichero de mayor calidad para distintos sistemas de información geográficos.

1.8.3 Sistema de Piloto Autónomo

El proyecto realizado por los estudiantes Chicaiza Fernando y Chuchico Cristian en la Universidad de las Fuerzas Armadas ESPE (Chicaiza C. & Chuchico A., 2015) se enfoca en la Implementación de un sistema de piloto autónomo basado en plataformas de tipo FPGA (Arreglo de Compuertas Programables en Campo) para la navegación autónoma del vehículo aéreo no tripulado

1.8.4 Generación de cartografía Básica

El proyecto realizado por la estudiante Lady Angulo en la Universidad de las Fuerzas Armadas ESPE (Angulo V., 2014) se desarrolla una metodología para generar cartografías básicas, pero de elevado nivel de detalle a partir de un sistema aéreo no tripula-

do el cual cuenta con sistemas GPS y módulos de orientación IMU empleando un filtro Kalman.

1.8.5 Orto-mosaicos y modelos digitales de elevación

En el presente proyecto se plantea la necesidad de incursionar en nuevos métodos para procesar la información fotogramétrica adquirida a partir de un UAV, debido a que la instrumentación, las características de vuelo y del sensor presentan grandes desafíos al momento de obtener un producto cartográfico adecuado.

1.8.6 Programa de orto-rectificación y georreferenciación de imágenes

El proyecto realizado por el estudiante López de Paz en la Pontífice Universidad Católica de Perú (López de Paz, 2014), se diseña un algoritmo para la generación una orto-imagen orientada a la agricultura de precisión en la producción de azúcar, a partir de imágenes aéreas las cuales se orto-rectifica y georreferencia. El archivo resultante puede ser analizado en un software GIZ para obtener datos relevantes.

CAPÍTULO II

MARCO TEÓRICO

2.1 Vehículos aéreos no tripulados

Un vehículo aéreo no tripulado o también conocido como UAV, denominado así por sus siglas en inglés (Unmanned Aerial Vehicle), es un vehículo motorizado, aéreo y que no posee operador humano a bordo. Usa las fuerzas aerodinámicas para proveerse a sí mismo levantamiento, y puede volar autónomamente o ser pilotado remotamente (Fernández C., 2012).

***Figura 1*.** UAV, modelo Xplus One de Xcraft

Usualmente estos vehículos aéreos tienen la capacidad de desplazarse de forma "VTOL" (del inglés de Vertical Take-Off and Landing), lo que implica que el avión aparte de realizar "despegue y aterrizaje verticales", vuela sin piloto a bordo y puede ser controlado de forma remota o en su defecto programado para ser completamente autónomo.

2.2 Aplicaciones de los UAV´s

Debido a que los UAV´s pueden ser ejecutados de manera remota o de manera autónoma (o con la mínima intervención humana), estas plataformas aéreas no tripuladas se han convertido en un área de investigación y desarrollo muy importante, ya que estas cuentan con un gran potencial para el desarrollo tanto de aplicaciones hacia la sociedad civil, así como a la comunidad militar.

Dentro de las aplicaciones de los UAV podemos encontrar:

- Aplicaciones militares. - generalmente son utilizados en misiones de inteligencia, vigilancia y reconocimiento en zonas de difícil acceso o que pongan en peligro la vida humana.
- Aplicaciones civiles. - Es en el uso civil en donde más se han desarrollado aplicaciones en los últimos años. Como, por ejemplo: en el área de la fotogrametría aérea, el uso de este tipo de vehículos aéreos para la adquisición de imágenes aéreas y captura de datos. Y esto se debe a que se trata de un sistema más novedoso y en pleno auge (Coello R. & Ballesteros A., 2014). Entre las principales aplicaciones civiles se puede mencionar:
    - Topografía: fotografía aérea con realización de mapas
    - Localización de accidentes en lugares de difícil acceso
    - Rápida detección de incendios y el seguimiento de su evolución
    - Control de cosechas, agricultura y paisaje
    - Gestión de crisis originados por desastres naturales, como inundaciones o terremotos

Figura 2. Dron DJI Phantom 3 para usos civiles
Fuente: *(DJI, 2018)*

2.3 Ventajas y desventajas de los UAV´s

Ventajas

La ventaja más importante que se puede destacar de este tipo de plataformas aéreas es el uso en zonas o lugares de alto riesgo o de difícil acceso para el ser humano, además se puede operar de manera autónoma gracias a sus sensores internos. (Paucar M. & Morales C. , 2017).

Desventajas

Por ser plataformas aún en desarrollo existen muchas deficiencias tanto en la transmisión y recepción de datos, como en la seguridad de los firmwares. Otra desventaja muy importante es la poca autonomía de vuelo que poseen estas plataformas debido a la poca capacidad de las baterías.

2.4 Descripción de un sistema UAV

El concepto de UAV, abarca el funcionamiento de todo un sistema complejo, en donde están integrados los segmentos aéreo y terrestre. Las funcionalidades que realiza

cada uno de los segmentos puede variar, dependiendo, del grado de autonomía que tenga el vehículo.

El segmento aéreo: Este segmento incluye la plataforma aérea, la carga útil, parte del sistema de comunicaciones y la estación de control en tierra, en este segmento se asume las tareas de planificación y control de la misión (Valencia A. , 2014).

El segmento tierra: Incluye el sistema de control de la aeronave y su carga, equipos de comunicaciones y se encarga de interpretar la información obtenida de los sensores. (Valencia A. , 2014).

2.5 Clasificación de los UAV´s

Aunque algunos autores aseguran que la clasificación más simple entre los UAV's es si tienen las alas fijas o rotativas, también se las puede representar en dos grandes grupos, de acuerdo con su método de despegue se los clasifica en: despegue vertical o despegue no vertical, como se muestra en la figura 3.

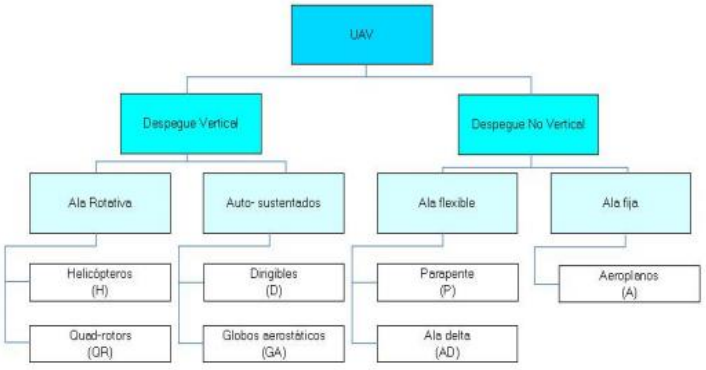


Figura 3. Clasificación de los UAV.
Fuente: *(Mayorga R, 2009)*

La diferencia más notable entre todas estas aeronaves está en la cantidad de movimientos que puede producir dicho vehículo mientras se encuentra suspendida, estos movimientos pueden variar entre un avance frontal, hasta maniobras rotativas o de traslación, así como la capacidad de permanecer suspendido en el aire.

Las aplicaciones y prestaciones de los UAV´s varían mucho de un tipo de aeronave a otra. A continuación, en la tabla 1, se presentan algunas de las características más representativas de cada aeronave. Siendo 1 las características que ofrecen menor prestación y 5 la característica de mayor prestación.

Tabla 1.

Características De Los Principales Tipos De Aeronaves

Característica	Helicóptero	Aeroplano	Dirigible	Quadrotor
Capacidad de vuelo estacionario	3		4	3
Velocidad de desplazamiento	3	4	1	2
Maniobrabilidad	3	1	1	4
Autonomía de vuelo (tiempo)	2	3	4	1
Resistencia a perturbaciones externas (viento)	2	4	1	2
Auto – Estabilidad	1	3	4	2
Capacidad de vuelos verticales	4	1	2	4
Capacidad de carga	3	4	1	2
Capacidad de vuelos en interiores	2	1	3	4

Fuente: (Coello R. & Ballesteros A., 2014)

2.6 UAV de ala rotativa o Quadrotor

Es un vehículo aéreo no tripulado que puede despegar y desplazarse gracias a sus cuatro rotores coplanares, cuyo movimiento depende de la variación de potencia por

parte de cada uno de ellos, consiguiendo así el movimiento deseado. (Blasco , García N., & Reynoso M., 2012) .

La configuración de giro de las hélices de estos vehículos es de forma que, dos giran en sentido horario mientras los otros dos giran en sentido antihorario, como se muestra en la figura 4.

Figura 4. Reacción de torques en cada motor de un quadrotor
Fuente: (Pestana M. J., Sánchez, Saripalli, & Campoy, 2017)

2.7 Funcionamiento de los Quadrotores

Es un sistema sub-actuado con 6GDL (grados de libertad) pero solo 4 entradas, las mismas que pueden ser expresados en función de las velocidades de los motores, permitiendo que las ecuaciones de control sean cuatro ecuaciones independientes. (Casanova , 2015).

Para el control de maniobrabilidad del UAV los motores se mueven en pares. Es decir, un par de motores gira en sentido horario y el otro par en sentido antihorario. Si se desea que el sistema permanezca suspendido en el aire a una altura constante, los 4 rotores deben girar a una misma velocidad, así mismo, si se necesita que el UAV se

eleve o baje de su posición a lo largo del eje Z se debe aumentar o disminuir la velocidad de los 4 rotores equitativamente.

En la siguiente figura se puede apreciar las variaciones necesarias en los giros de los rotores para realizar un movimiento en el quadrotor hacia adelante. Para esto es necesario que los rotores 3 y 4 de la figura 5. tengan una velocidad mayor que los otros dos rotores.

Figura 5. Movimientos básicos de dron: movimiento hacia adelante
Fuente: *(Casanova , 2015)*

En la siguiente figura se puede apreciar las variaciones necesarias en los giros de los rotores para realizar un movimiento en el quadrotor hacia adelante. Para esto es necesario que los rotores 1 y 2 de la figura 6. tengan una velocidad mayor que los otros dos rotores.

Figura 6. Movimientos básicos de dron: movimiento hacia atrás
Fuente**:** *(Casanova , 2015)*

En la siguiente figura se puede apreciar las variaciones necesarias en los giros de los rotores para realizar un movimiento en el quadrotor hacia adelante. Para esto es necesario que los rotores 1 y 4 de la figura 7. tengan una velocidad mayor que los otros dos rotores.

Figura 7*.* Movimientos básicos de dron: movimiento hacia la derecha
Fuente**:** *(Casanova , 2015)*

En la siguiente figura se puede apreciar las variaciones necesarias en los giros de los rotores para realizar un movimiento en el quadrotor hacia adelante. Para esto es necesario que los rotores 2 y 3 de la figura 8. tengan una velocidad mayor que los otros dos rotores.

Figura 8. Movimientos básicos de dron: movimiento hacia la derecha
Fuente: *(Casanova , 2015)*

En la siguiente figura se puede apreciar las variaciones necesarias en los giros de los rotores para realizar un movimiento en el quadrotor hacia adelante. Para esto es necesario que los rotores 1 y 3 de la figura 9. tengan una velocidad mayor que los otros dos rotores.

Figura 9. Movimientos básicos de dron: giro hacia la derecha
Fuente: *(Casanova , 2015)*

En la siguiente figura se puede apreciar las variaciones necesarias en los giros de los rotores para realizar un movimiento en el quadrotor hacia adelante. Para esto es necesario que los rotores 1 y 4 de la figura 10. tengan una velocidad mayor que los otros dos rotores.

***Figura 10*.** Movimientos básicos de dron: giro hacia la izquierda
Fuente: *(Casanova , 2015)*

2.8 Elementos de los Quadrotores

En esta sección se pretende presentar y describir los distintos elementos que conforman un quadrotor básico, a fin de lograr una mejor comprensión de este y sus posteriores aplicaciones.

2.8.1 La estructura o marco

Es la estructura sobre la que se van a montar los demás elementos. Este debe ser de un material rígido de forma que se minimicen las vibraciones ocasionadas por los rotores.

Normalmente los materiales con los que se fabrican suelen ser de fibra de carbono, El poliestireno expandido (EPS), o aluminio; aunque también se pueden utilizar otro tipo de materiales más específicos como fibra de vidrio.

Figura 11. Estructura del quadrotor xcraft plus One, material: Poliestireno expandido (EPS)
Fuente*: (Xcraft, 2017)*

2.8.2 Motores

Los motores utilizados en este tipo de aeronaves son los motores brushless (motores sin escobilla). Este tipo de motores son similares a los de corriente continua en el sentido de que usan bobinas e imanes para mover el eje, pero no cuentan con escobillas en el mismo (Sevilla F., 2014) y también giran a un mayor número de revoluciones que los motores de corriente continua.

En las especificaciones de este tipo de motores se suele incluir como parámetro adicional, el denominado factor Kv. Este factor indica el número de revoluciones por minuto a la que puede girar el rotor por cada voltio que se le aplique.

***Figura 12*.** Motor del quadrotor DJI
Fuente*: (DJI, 2018)*

2.8.3 Controlador electrónico de velocidad

Como se ha explicado anteriormente, los motores utilizados en los Quadrotores son brushless, estos requieren de controladores electrónicos para regular su velocidad de giro. Además, suelen ser multifásicos más comúnmente trifásicos.

2.8.4 Batería

Estas baterías deben suministrar unas intensidades de corriente relativamente altas. Además, conviene que el peso añadido por estas baterías sea el menor posible. Por estas razones las baterías más recomendadas para este tipo de aplicaciones son el tipo LiPo.

Figura 13. Batería LiPo
Fuente: *(Xcraft, 2018)*

2.8.5 Unidad inercial de medidas (IMU)

Es un dispositivo que contiene una serie de sensores que permiten obtener datos de: la velocidad, la orientación y las aceleraciones del cuerpo sobre las que se montan. Estos datos son analizados en el controlador y son procesados, para efectuar los cambios en las velocidades de giro de los motores. La IMU generalmente dispone de una combinación de: 3 acelerómetros, 3 giroscopios, y algunas veces 3 magnetómetros.

2.8.6 Controlador de vuelo

El controlador se puede considerar como el cerebro del quadrotor, puesto que este recibe la información obtenida por los sensores, a través de la IMU, y la procesa modificando las velocidades de los rotores de manera que el quadrotor permanezca en equilibrio en el aire. Estas velocidades se calculan a través de complejos algoritmos.

2.8.6.a Controlador de vuelo comercial: Pixfalcon

Es un módulo de alto rendimiento y de software abierto en el cual funciona un piloto automático adecuado para diferentes plataformas robóticas como: ala fija, multi-rotores, helicópteros. El proyecto Pixfalcon está dirigido a la investigación y desarrollo tanto de las necesidades de aficionados como industriales (3DRobotic, 2018).

Figura 14. Módulo Pixfalcon
Fuente: *(Pixfalcon, 2018)*

Características

Las principales características de este módulo se resumen en la tabla 2.

Tabla 2.

Características Principales Del Módulo Pixhawk

Procesador	Sensores	Interfaces	Otros
Núcleo de 32 bits STM32F427 Cór-tex M4 con FPU	Giroscopio (ST Micro L3GD20H 16 bits)	1x UART (puertos serie), una capacidad de alta potencia	Sistema de alimentación integral con una amplia pro-tección.
168 MHz	acelerómetro / magnetómetro de 14 bits (ST Micro LSM303D)	CAN (con un transcep-tor interno de 3.3V)	Controlador de diodo con control automático ante error.
256 KB de RAM	Acelerómetro / giroscopio de 3 ejes (Invensense MPU 6000)	Entrada compatible con Spektrum	Admite Servos de alta poten-cia (7 V) y de alto voltaje
2 MB Flash	MEAS MS5611 barómetro	▪ Entrada de señal de suma PPM ▪ I2C ▪ SPI ▪ Entradas de 3.3 y 6.6 V ADC ▪ Puerto micro USB interno	Peso y dimensiones: ▪ Peso: 38 g ▪ Ancho: 50 mm ▪ Espesor: 15,5 mm ▪ Longitud: 81.5mm

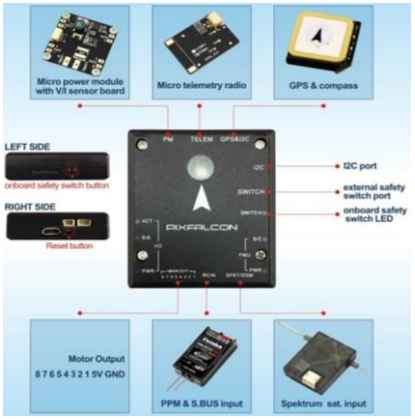


Figura 15. Pin Outs y conectores del módulo Pixfalcon, presente en el dron X plusOne de Xcraft.
Fuente: (Pixfalcon, 2018)

2.8.7 Comunicación piloto-aeronave

Para que un quadrotor se pueda controlar de forma remota, es necesario instalar un sistema de comunicación entre el piloto y la aeronave, para que funcione de manera eficaz y de acuerdo con las prestaciones requeridas.

2.8.7.a Transmisores de radio control

Este sistema está conformado por un transmisor, desde el cual se comunica con el receptor situado en el quadrotor. El receptor decodificará la señal recibida y la enviara al controlador donde se actuará según se requiera. (Sevilla F., 2014).

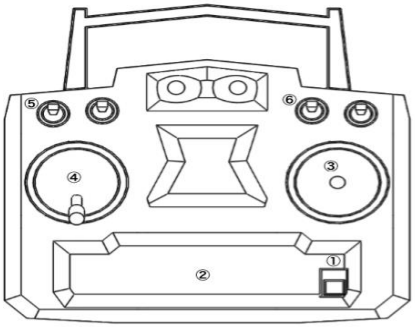


Figura 16. Ejemplo de transmisor del quadrotor, X plusOne
Fuente: *(Xcraft, 2017)*

2.8.8 sensores de distancia

Para que un vuelo autónomo sea posible, es necesario obtener información del entorno próximo a el quadrotor, esta tarea se logra gracias a un conjunto de sensores ubicados en la aeronave. A pesar de la gran variedad de sensores existentes en el mercado, los más utilizados son los sensores infrarrojos, ultrasónicos y recientemente en auge la utilización de cámaras estereoscópicas.

2.8.9 Dispositivos adicionales

Existen una gran variedad de dispositivos que se pueden incorporar a los Quadrotores con la finalidad de mejorar sus prestaciones. A continuación, se detallan los dispositivos más utilizados.

a. Cámara de video

Estas cámaras de vídeo pueden transmitir los datos obtenidos a través de un sistema de comunicación especifico, de forma que lo observado por el quadrotor pueda usarse como ayuda al control del mismo.

b. GPS

Con este sistema de posicionamiento se puede tener una referencia de alta precisión muy útil, la cual no se verá afectada por campos magnéticos externos. Además, permiti-

rá monitorizar la trayectoria del quadrotor sabiendo en todo momento donde se encuentra.

2.9 Selección de componentes

Para la elaboración de las especificaciones de un proyecto, el autor Karl Ulrich en su libro: diseño y desarrollo de productos (Ulrich T., 2013) sugiere como primer paso recopilar e identificar las necesidades del “Cliente”, asegurándose que el proyecto final se enfoque en estas necesidades.

Las especificaciones para el proyecto que se va a desarrollar dependerán de lo que sea técnica y económicamente factible, lo que la competencia ofrezca en el mercado, así como las necesidades del potencial cliente. (Ulrich T., 2013)

El proceso que se lleva a cabo para establecer las especificaciones del proyecto consta de cuatro pasos:

- Elaborar la lista de métricas
- Recopilar información de comparaciones con las ofertas del mercado
- Establecer los valores meta ideales y marginalmente aceptables
- Analizar los resultados y el proceso

2.9.1 Lista de métricas

Es importante considerar que para que una métrica sea de utilidad para el proceso de selección de componentes, esta debe reflejar el grado al cual el proyecto satisface las necesidades analizadas en la etapa de planificación del proyecto. En el caso ideal, debe haber una única métrica para cada necesidad.

En la tabla 3. Se muestra una lista de métricas que pretenden satisfacer las necesidades del proyecto, así como su orden jerárquico de importancia con sus respectivas unidades, donde “1” es poco importante y “5” muy importante.

Tabla 3.

Lista De Métricas

N°	Necesidad	Métrica	Import.	Unidad
1	Amplio tiempo de vuelo	Tiempo de Vuelo por Batería	4	min.
2	Peso Ligero	Peso	3	gr.
3	Detectar Obstáculos	Distancia Detección de obstáculos	5	cm
4	Rápida Respuesta	Velocidad de Procesador	5	GHz
5	Tamaño Adecuado	Medidas	2	cm
6	Amplio Rango de Control	Radio de señal (Rango)	3	m
7	Durabilidad	Factor de seguridad	3	Fs
8	Silencioso	Ruido Producido	2	Decibeles
9	Capacidad de almacenamiento	Memoria	3	Gb
10	Velocidad de vuelo	Velocidad de desplazamiento	4	Km/h
11	Fiabilidad	Error de seguimiento de ruta	4	%

La relación entre las necesidades y las métricas se la puede representar en una matriz necesidades-métricas, en donde las filas de la matriz equivalen a la necesidad del cliente y las columnas corresponden a las métricas. En la tabla 4 se puede observar

esta matriz en la cual, una marca (X) en la celda de la matriz significa que la necesidad y la métrica asociada con la celda están relacionados. (Ulrich T., 2013)

Tabla 4.

Matriz De Necesidades-Métricas

		1	2	3	4	5	6	7	8	9	10	11
		Tiempo de Vuelo por Batería	Peso	distancia Detección de obs.	Velocidad de Procesador	Medidas	Radio de señal (Rango)	Factor de seguridad	Ruido Producido	Memoria Interna	Velocidad de desplazamiento	error de seguimiento de ruta
1	Amplio tiempo de vuelo	X	X									
2	Peso Ligero		X			X						
3	Detectar Obstáculos			X	X							
4	Rápida Respuesta				X							
5	Tamaño Adecuado					X						
6	Amplio Rango de Control						X					
7	Durabilidad		X			X		X				
8	Silencioso								X			

9	Capacidad de almacenamiento									X		
10	Velocidad de vuelo		X								X	
11	Fiabilidad				X							X

2.9.2 Información de comparaciones con las ofertas del mercado

Una muy buena manera de presentar la información recopilada es en forma de una tabla de comparaciones, como la que se muestra en la tabla 5. En la cual se analiza la importancia que tiene cada métrica y se compara los valores reales que ofrecen los fabricantes de los diversos productos existentes en el mercado

Tabla 5.

Comparación De Las Ofertas Del Mercado / Vehículo Aéreo No Tripulado (UAV)

N°	Métrica	Imp.	Unidad	Phanton 4 pro	DJI Mavic Pro	X Plus One
1	Tiempo de Vuelo por Batería	4	12	30	27	20
2	Peso	3	gr	1388	743	1305
3	Medidas	2	cm	35	33.5	82.55
4	Radio de señal (Rango)	3	m	3500	3000	4000
5	Factor de seguridad	3	Fs	4	4	4
6	Ruido Producido	2	Decibeles	-	-	-
7	Velocidad de desplazamiento	4	Km/h	78 -58 -50	64.8	100
8	error de seguimiento de ruta	4	%	2	0.1	2



En la tabla 6. Se puede observar la comparación de las diversas ofertas del mercado (UAV), pero esta vez se analiza el grado relativo al cual estos productos satisfacen las necesidades planteadas al principio, en donde 1 es poco importante y 5 muy importante

Tabla 6.

Comparación De Los Productos Del Mercado (UAV) Basado En La Satisfacción De Las Necesidades

N°	Métrica	Import.	Phanton 4 pro	DJI Mavic Pro	Xplus One
1	Tiempo de Vuelo por Batería	4	4	4	3
2	Peso	3	2	3	3
5	Medidas	2	3	3	4
6	Radio de señal (Rango)	3	3	2	4
7	Factor de seguridad	3	4	4	4
8	Ruido Producido	2	1	1	1
10	Velocidad de desplazamiento	4	3	2	5
11	error de seguimiento de ruta	4	3	4	3
Total			75	71	86

La aeronave Xplus One es la que presenta las mejores características de velocidad de desplazamiento, radio de alcance de la señal de control y debido a sus medidas adecuadas ha sido seleccionado como adecuado para el desarrollo del proyecto de investigación. Además de que presenta una generosa capacidad de carga.

CONTINÚA

Figura 17. Aeronave Xplus One

en la tabla 7. Se puede observar la comparativa de los sensores existentes en el mercado que permitan la detección y la posterior evasión de obstáculos, En donde se las columnas representan a diversos productos existentes en el mercado y las filas representan las métricas analizadas anteriormente.

Tabla 7.

Comparaciones De Las Ofertas/ Sensor Para La Detección De Obstáculos

N°	Métrica	Imp.	Unidad	LeddarTech-Vu8	Estéreo Visión-ZED	Kinect
1	robustez al ambiente*	4	-	4	5	3
2	Peso	3	gr	830	159	1305
3	Distancia Detección de obstáculos	5	m	100	200	3,5
4	Velocidad de adquisición	5	Hz	50	100	16000
5	Angulo de detección	3	°	20	80	57

En la tabla 8. Se puede observar la comparación de las diversas ofertas del mercado para los sensores para la detección de obstáculos, pero esta vez se analiza el grado relativo al cual estos productos satisfacen las necesidades planteadas al principio, en donde 1 es poco importante y 5 muy importante.

Tabla 8.

Comparación de los productos del mercado basado en la satisfacción de las necesidades

N°	Métrica	Imp.	LeddarTech-Vu8	Estéreo Visión-ZED	Kinect
1	robustez al ambiente*	4	4	5	3
2	Peso	3	3	5	4
3	Distancia Detección de obstáculos	5	3	4	1
4	Velocidad de adquisición	5	3	4	5
5	Ángulo de detección	3	1	5	3
	TOTAL		58	90	63

El sensor de estéreo Visión ZED presenta una mayor capacidad de vista a larga distancia, es decir, permite visualizar un obstáculo que se aproxima a una mayor distancia, brindándole un mayor tiempo al procesador para poder tomar cualquier acción evasiva. Por otra parte, su bajo consumo y peso lo convierten en la opción ideal para el desarrollo de la aplicación.

Figura 18. Sensor de estéreo visión ZED
Fuente: *(StereoLabs, 2018)*

2.9.3 Cámara estéreo ZED

Una cámara estereoscópica es una cámara capaz de crear imágenes en 3 dimensiones. La cámara ZED, de StereoLabs (StereoLabs, 2018) . Esta cámara reproduce la forma en la que funciona la visión humana, es decir usando sus dos “ojos” y mediante triangulación. Además, proporciona una compresión tridimensional de la escena que observa, permitiendo aplicaciones donde una plataforma móvil se vuelva consciente de su espacio y movimientos.

2.9.3.a Funcionamiento de la cámara ZED

ZED es una cámara con 2 lentes los cuales captura video de alta definición con un amplio campo de visión y emite 2 transmisiones de video sincronizadas, izquierda y derecha (StereoLabs, 2018).

2.9.3.a Especificaciones

Las especificaciones electrónicas y generales de la cámara se resumen en las tablas 9, 10 y 11 como se muestran a continuación.

Tabla 9.

Especificaciones De La Cámara

Resolución de salida	2x (2208x1242) @ 15fps (2.2K) 2x (1920x1808) @ 30fps (1080p) 2x (1280x720) @ 60fps (720p) 2x (640x480) @ 100fps (VGA)
Formato de salida	YUV 4:2:2
Campo de visión	Máximo 110°
Rango de profundidad	De 50 cm a 20 m

interfaz	Usb 3.8
Campo de visión	Máximo 110°

Tabla 10.

Especificaciones electrónicas de la cámara

Tipo de sensor	1/2.7''
Tamaño de matriz activa	4M pixeles por sensor
Distancia focal	2.8 mm (0.11'') – f/2.0
obturador	Obturador de rodamiento electrónico
Tamaño del pixel	2 um

Tabla 11.

Especificaciones Generales

Dimensiones	175 x 30 x 33 mm
Peso	159 gr
Potencia	380mA /5v alimentado por USB
Temperatura de operación	0 °C a 45°C

2.9.4 Valores objetivo: ideales y marginalmente aceptables

En esta etapa se sintetiza la información recopilada anteriormente, para poder establecer un valor ideal y un valor marginalmente aceptable de trabajo y operación.

Tabla 12.

Especificaciones – Objetivo

N°	Métrica	Imp.	Unidad	Valor Marginal	Valor Ideal
1	Tiempo de Vuelo por Batería	4	min.	>10	>20
2	Peso	3	gr	500 -1500	800 - 1300
3	Distancia Detección de obstáculos	5	cm	70 - 3000	70 - 2000
4	Velocidad de Procesador	5	GHz	> 1.5	> 2.0
5	Medidas	2	cm	< 85	< 500
6	Radio de señal (Rango)	3	m	< 1500	> 2000

7	Factor de seguridad	3	Fs	4	4
8	Ruido Producido	2	Decibeles	> 30	< 20
9	Memoria	3	Gb	>32	128
10	Velocidad de desplazamiento	4	Km/h	> 30	> 60
11	Error de seguimiento de ruta	4	%	2	2

2.10 Modelado Matemático del Quadrotor

El modelo matemático describe la dinámica del movimiento del vehículo según sus 6 grados de libertad, teniendo en cuenta la dinámica de los actuadores primarios, es decir los motores. La propia estructura que poseen los diferentes Quadrotores hacen que las ecuaciones que rigen sus movimientos sean muy independientes unos de otros, en el capítulo 3 se analiza de mejor manera este tema y se enlistan los elementos principales del estudio de estas ecuaciones a fin de obtener el modelo matemático requerido.

2.11 Métodos de control

Debido a que en los últimos años se han aumentado las investigaciones sobre métodos de control de vuelo del quadrotor, han surgido una gran cantidad de algoritmos o sistemas de control. en esta sección se describirá uno de los métodos de control más utilizado, como lo es el PID, además de ser el control con el que se va a trabajar en la presente investigación.

2.11.1 Controlador PID

El controlador PID (proporcional Integrativo Derivativo) es un tipo de control genérico sobre un sistema lineal en lazo cerrado. El controlador PID estima el error calculado a partir de la diferencia entre la salida deseada y la salida medida e intenta minimizar el

error ajustando el mando del sistema. Este controlador está determinado por tres parámetros: Proporcional, integral y derivativo.

La implementación del control PID de un quadrotor se divide en las siguientes partes:

- **Control PID Roll:** encargado del control de giro horizontal, el equilibrio del quadrotor se centra en este eje por lo que un buen control y una adecuada sintonización del PID son críticos para evitar que el vehículo gire.
- **Control PID Pitch:** es el control de elevación, responsable de mantener el avance o retroceso del sistema, con la libertad para poder acelerar o disminuir la velocidad si las maniobras lo requieren.
- **Control PID Yaw:** es el control de giro vertical y evita que las fuerzas ejercidas por las hélices hagan que el vehículo gire sobre su propio eje, pero mantiene una libertad considerable para poder orientar la acción de maniobra izquierda y derecha.
- **Control de Altitud:** Para el control de altitud, la señal de control debe mantener un valor constante de “offset” que actué en sentido contrario y contrarreste la gravedad, para que el vehículo mantenga la altura deseada.

2.12 Sistemas de navegación autónoma

Las soluciones para la navegación autónoma pueden ser muy variadas. Ya sea en función de la tecnología de control empleada, de los equipos existentes en el mercado, así como las técnicas de control de vuelo utilizado.

2.12.1 Navegación autónoma basada en GPS/INS

La navegación basada en GPS/INS (Global positioning system / inertial navigation system) permite al vehículo moverse en rutas preestablecidas de un punto a otro, sin depender de sistemas de comunicación en tierra.

Para obtener el posicionamiento completo se requiere adicionalmente al GPS, un algoritmo inercial que proporciona las variables adicionales de vuelo: la inclinación de los ejes del avión con respecto al eje de referencia “tierra” o AHRS (Altitude and heading reference system) y el sistema de posicionamiento y velocidad del avión (Chicaiza C. & Chuchico A., 2015).

2.12.2 Navegación autónoma basada en sensores de flujo óptico

En general, la navegación basada en sensores de flujo óptico utiliza una cámara de alta velocidad y una resolución de 30 x 30 pixeles. Este sistema de navegación basa su lógica en controlar cada grado de libertad con cada uno de los controladores PID.

2.12.3 Navegación autónoma basada en control por visión

Este sistema de navegación comprende tres componentes: el lazo de control interno (IMU), lazo de control externo (visión) y el algoritmo de planificación de trayectoria. El lazo de control externo es el que se encarga del control del ángulo de dirección y de la posición del UAV. (Chicaiza C. & Chuchico A., 2015).

2.12.4 Navegación autónoma basada en construcción de mapas 3d

La construcción de un modelo 3d del entorno puede considerarse como un prerequisito para la navegación, ya que la seguridad y la confiabilidad de la misión dependen de la exactitud del modelo 3D respecto al entorno. En el mercado existen varios sistemas capaces de proveer información 3D, como es el ejemplo de la cámara estereoscópica ZED.

2.13 Sistema Operativo:

2.13.1 R.O.S (Robotic Operating System)

"ROS Es un sistema operativo de código abierto que permite desarrollar aplicaciones para robots o plataformas de investigación" (ROS.org, 2018). Actualmente este sistema es uno de los más utilizados para programar sobre cualquier plataforma robótica, esto es debido a los servicios estándar que un sistema operativo provee, como lo son: abstracción de hardware, librerías y herramientas, control de dispositivos de bajo nivel, implementación de funciones utilizadas normalmente, gestión de paquetes, entre otros.

Figura 19. logotipo ROS
Fuente: *(ROS.org, 2018)*

2.13.1.a Arquitectura de ROS

ROS se encuentra dividido en tres niveles: Nivel gráfico, nivel de sistemas de archivos, nivel comunitario. En la figura 20 se puede observar mejor esta distribución.

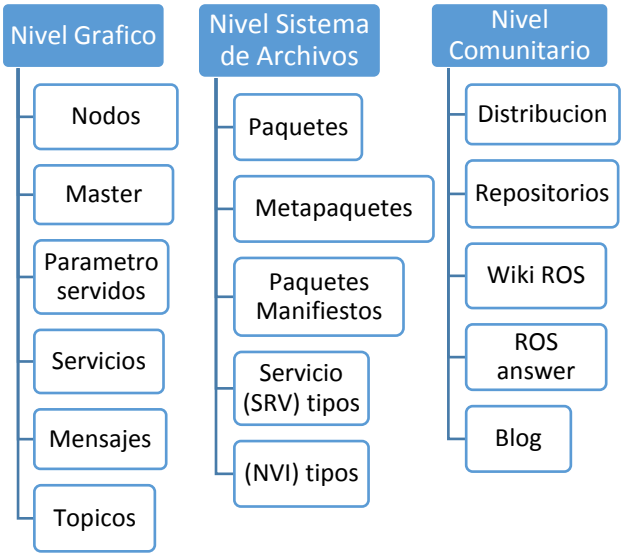


Figura 20. Arquitectura de ROS

- **Nivel Gráfico**

Se trata de un gráfico muy útil en el desarrollo, basado en una red "igual a igual" de los procesos ROS. Los cuales procesan los datos conjuntamente y de una manera sencilla permiten observar las conexiones internas en ROS.

Nodos: Se pueden definir como los procesos ejecutables encargados de realizar el computo en un sistema de control robótico, generalmente existen varios nodos en una misma red, donde cada uno se encarga de ejecutar una tarea en particular.

Master: es el proceso central encargado de mantener un registro de todo el grafo computacional, permitiendo la comunicación entre nodos.

Servidor de parámetros: permite que los datos sean almacenados de manera segura en un lugar central, y este forma parte del Master.

Servicio: Están conformados por 2 partes: un mensaje que es el encargado de preguntar y otro mensaje que es la respuesta. Los servicios se definen mediante archivos SRV, los cuales se compilan en el código fuente de una biblioteca de ROS.

Mensaje: un mensaje es una estructura de datos simple, que puede ser del tipo primitivo (entero, flotante, boolean, etc.) o un arreglo de estos, además es la manera de comunicación entre nodos ya que estos se comunican enviándose mensajes.

Tópico: los mensajes que se envían son generalmente en base a un sistema de publicación o suscripción.

Bags: Son un formato para guardar y reproducir datos de mensajes de ROS

- **Nivel sistema de archivos**

Paquetes: se pueden considerar como la unidad de organización de ROS. Este contiene procesos (nodos), librerías de desarrollo, ficheros de configuración.

Metapaquetes: son paquetes que únicamente sirven para representar un conjunto de paquetes que se encuentren relacionados

Paquete manifiesto: (paquete.xml) su función principal es brindar metadatos sobre un paquete, como lo son: nombre, versión, descripción, dependencias y otra información propia del meta como paquetes exportados. (Tumbaco M. & Quimbita Z., 2014).

(NVI) tipos: definen la estructura de los datos que son enviados por la mensajería de ROS.

Servicio (SRV) tipos: las descripciones de servicio que se encuentran almacenados en un paquete definen la solicitud y la estructura de los datos de respuesta, propios de los servicios ROS. (Tumbaco M. & Quimbita Z., 2014)

- **Nivel comunitario**

Estos recursos periten al usuario intercambiar software, estos recursos son:

ROS Distribuciones: Es una colección de paquetes que se pueden instalar, las cuales tienen un propósito similar a las distribuciones Linux, es decir, permiten una fácil instalación de una colección de softwares.

Repositorios: Ya que se trata de un lugar donde se recopila software de código libre, generalmente asociadas en stack´s (pilas). De esta manera es posible acceder a ellas y descargarlas para su uso. El lugar más usado es Github

Wiki ROS: se trata del principal foro de información sobre ROS en donde se encuentra mucha información relevante. Cualquier persona mediante una cuenta puede contribuir con documentación, o realizar correcciones o actualizaciones de trabajos y más.

ROS answers: Es un sitio de preguntas y respuestas sobre temas relacionados a ROS

2.13.2 openCv

OpenCv es un software libre tanto para uso académico como comercial. Es compatible con sistemas operativos como Ubuntu, Linux y también Windows mediante interfaces como C++, C, Python y Java.

En la actualidad OpenCv, permite realizar muchas tareas como: captura y pretratamiento de los datos de imagen, además permite a los algoritmos de alta complejidad la extracción de características, seguimiento de movimiento y aprendizaje automático, (OpenCV, 2018).

2.13.2.a Estructura de OpenCv

Tiene una estructura modular, es decir, el paquete openCv incluye varias bibliotecas compartidas o estáticas.

- **Funcionalidades básicas:** este módulo define la estructura básica de los datos, incluyendo la matriz multidimensional (Mat) y demás funciones utilizadas por todos los demás módulos.
- **Procesamiento de imágenes:** este módulo permite el filtrado de imágenes lineales y no lineales, transformación geométrica de imágenes (redimensionar, reasignación genérica basado en tablas), la conversión de espacio de color, histogramas, etc.
- **procesamiento de video:** este módulo a su vez contiene otro submódulo de análisis de video que perite realizar la estimación de movimiento y otros algoritmos de seguimiento de objetos. (OpenCV, 2018)
- **calib3d:** Módulo con múltiples algoritmos de visión geométrica, calibración de cámara; algoritmos de correspondencia estéreo y elementos de la reconstrucción 3D. (Paucar M. & Morales C. , 2017)

- **features2d:** Detectores de características sobresalientes, descriptores y comparadores de descriptores.
- **Highgui:** Una interfaz de fácil uso y muy intuitiva para las capacidades de interfaz de usuario
- **Videoio:** Una interfaz fácil de usar para la captura de vídeo y códec de vídeo. (OpenCV, 2018).

2.13.3 ArduPilot

ArduPilot es un software de piloto automático de código abierto que permite controlar cualquier vehículo con mucha flexibilidad únicamente actualizando la última versión del firmware para determinado vehículo. Este firmware posee un conjunto de habilidades en función del hardware que se requiera utilizar; en este caso, un quadrotor. Es por esto que se necesita instalar el firmware denominado Arducopter.

ArduPilot cuenta con características como: múltiples modos de vuelo, fácil configuración al tipo de vehículo, programación a prueba de fallos, etc. Estas tareas se llevan a cabo a través del programa Mission planner, desarrollado por esta compañía. (Jimenez L., 2017).

2.13.3.a software de supervisión del Pixfalcon

Los módulos sobre los que funcionan los pilotos automáticos se pueden probar, calibrar y además simular en softwares diseñados para esto. Uno de estos softwares es el Mission Planner el cual permite la creación de planes de vuelo para los diferentes vehículos no tripulados, utiliza código abierto de ArduPilot.

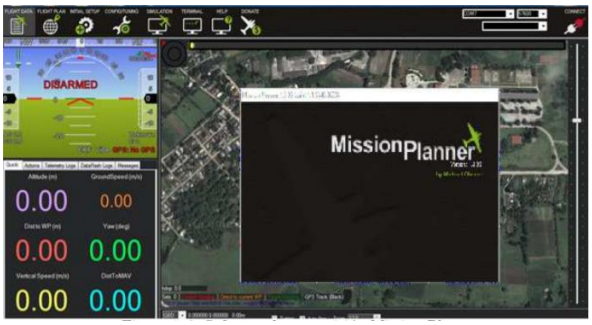


Figura 21. Ventana principal del software Mission Planner

2.13.3.a *Descripción del Mission Planner*

Mission Planner es un software compatible tanto para Windows como para Ubuntu y puede ser utilizado para la configuración o como ayuda para el control de vuelo de un vehículo autónomo. Entre las principales acciones posible a realizar con el Mission Planner tenemos:

- Cargar el firmware en el piloto automático controlador del vehículo.
- Planificar, guardar y cargar misiones autónomas en el módulo de piloto automático.
- Posee una interfaz con un simulador de vuelo en PC.
- Con el módulo de telemetría se puede: supervisar el estado del vehículo mientras está en funcionamiento.
- Ver, analizar, y grabar registros de telemetría, los cuales contiene mucha más información que los registros de pilotos automáticos. (Pelaez V., 2016).

2.14 Sistemas de comunicación en el UAV

Es necesario estudiar el sistema de comunicación interno del UAV, así como la comunicación que se producirá entre el vehículo y la estación de control o llamada

“ground Control Station”. A continuación, se detallan cada uno de los elementos presentes en este sistema de comunicación.

2.14.1 PC

Dado que el objetivo es tener un UAV autónomo, se requiere una buena capacidad de computación. Por ello se decide utilizar un segundo elemento de computación que almacene el programa principal y se comunique con Pixfalcon.

2.14.2 Protocolo de comunicaciones: MAVLink

El protocolo MAVLink llamado así por sus siglas en ingles “Micro Air vehicle Link” fue lanzado en 2009 bajo licencia LGPL para la comunicación de pequeños vehículos no tripulados. (Plaza Rey, 2017).

Generalmente es utilizado para la comunicación entre el quadrotor y la GCS (estación de control en tierra), así como también para la comunicación interna de los subsistemas del vehículo. Mediante este protocolo se puede transmitir datos de: orientación del vehículo, ubicación GPS, velocidad, así como también envió y recepción de ordenes al controlador de vuelo.

2.14.3 MAVProxy

Es una herramienta que sirve como complemento a la estación de control en tierra ya que nos permite reenviar mensajes provenientes del vehículo hacia diferentes estaciones de control en tierra ubicadas en distintas plataformas como ordenadores, tabletas,

etc. También se pueden enviar comandos en tiempo real a el UAV desde una estación de control en tierra.

2.15 Orto-mosaico

La metodología para la elaboración de una orto-imagen se puede dividir en diferentes fases, las cuales son explicadas a continuación:

2.15.1 Selección de Zona de Interés

En esta etapa corresponde determinar cuáles serán los parámetros y condiciones previas al vuelo, se puede decir que en esta fase se realiza un plan de vuelo determinando el área de terreno a analizar (tipo de zona: ciudad, bosque, etc.), todas las condiciones de vuelo del quadrotor, y también cual será la fuente del material proporcionado para la realización de la orto-imagen (fotos o video).

2.15.2 Algoritmos de detección de puntos característicos

Esta fase consiste en la obtención de puntos característicos mediante la utilización de diversos algoritmos. Dentro de los algoritmos más utilizados se encuentran: SIFT (por sus siglas en inglés: “Scale Inavarian Feature Transform”) y SURF (“Speeded Up robust Feautres”). (Peréz C. , 2014).

2.15.3 Extracción de características y emparejamiento

En esta fase se utilizan los puntos obtenidos en la fase anterior mediante los diferentes algoritmos utilizados, y se extraen características de los puntos obtenidos. El méto-

do asociado a estos algoritmos consiste en extraer los valores de intensidad de los pixeles colindantes al punto en forma de bloque (Peréz C. , 2014).

Posteriormente se continua con el emparejamiento, este proceso consiste en aplicar un método que calcule una matriz de distancia euclidiana entre características de puntos, (Peréz C. , 2014) y después asociarlos conforme a la menor distancia.

2.15.4 Obtención de la transformación

Esta etapa consiste en el cálculo de la transformación con los puntos característicos obtenidos en la entapa anterior, en caso de que todos los emparejamientos fueran correctos. Para corregir los errores de asociación de estos puntos y no afectar la calidad de la transformación existen algoritmos que realizan este proceso como por ejemplo el algoritmo RANSAC

2.15.5 *Métodos de unión*

Esta etapa tiene la necesidad de crear una matriz que represente todos los pixeles del fotograma y del mapa, donde cada celda corresponde a un pixel de la imagen correspondiente

2.16 Localización y Mapeo Simultaneo (SLAM)

SLAM llamado así por sus siglas en inglés “Simultaneous Localization and Mapping”, es una técnica que permite crear de manera simultánea un mapa del entorno mientras se conoce la posición del robot en cualquier instante, la cual es una técnica muy utiliza-

da en robótica móvil. El objetivo de la técnica SLAM es la de solucionar los principales problemas que engloban el concepto de navegación. El cual se puede entender como: la capacidad de un robot para movilizarse en un entorno evitando obstáculos mientras se ejecuta una ruta planificada. Las soluciones implementadas de la técnica SLAM se pueden clasificar en base a: el tipo de mapa que se genera, sensor utilizado, algoritmo empleado.

2.16.1 Tipo de mapa que se genera

Los cuales pueden ser de 2 tipos fundamentalmente; mapas de ocupación los cuales definen las zonas del espacio que se encuentran libres y las que están ocupadas por obstáculos para luego representarlas en un entorno bidimensional. (Gil Aparicio, 2008). Y también están los mapas de marcas (Landmark-based maps) los cuales definen la posición de un conjunto de puntos en el espacio con respecto a un sistema de coordenadas globales. En este último se define la posición en un entorno tridimensional.

2.16.2 Sensor utilizado para captar el entorno

En la actualidad existe una amplia variedad de sensores que se pueden utilizar para “visualizar” el entorno como: Sonar, laser en dos o tres dimensiones. Pero recientemente ha surgido un especial interés por las cámaras estereoscópicas, las cuales extraen puntos característicos de las imágenes utilizándolos como Landmarks o marcas visuales en el entorno (Gil Aparicio, 2008).

2.16.3 Algoritmos de SLAM

Las principales técnicas utilizadas en la actualidad para resolver el problema que plantea SLAM se basan en uno de los dos siguientes algoritmos:

- **Filtro de Kalman Extendido (EKF)** El cual considera que el mapa está formado por un conjunto de *landmarks* referidos a un sistema de referencia global y estima un vector de estado de la posición del robot y la posición de las *landmarks* del mapa. (Gil Aparicio, 2008).
- **Filtro de Partículas de tipo Rao-Blackwellized** Comúnmente conocido como FastSLAM, la característica principal de este algoritmo es que emplea un conjunto de partículas para representar la incertidumbre sobre la posición del robot, a la vez que, cada partícula cuenta con un mapa asociado.

CAPITULO III

MODELAMIENTO Y SIMULACIÓN.

En esta sección se analizará el modelo matemático que describe la dinámica del quadrotor según sus 6 grados de libertad, a partir de la formulación Newton-Euler como lo plantea Rico Ramón en su publicación (Rico, Maisterra, & Martínez, 2015) y se detallaran los procedimientos necesarios para identificar experimentalmente los parámetros de dicho modelo matemático.

3.1 Sistema de referencia

Es necesario comenzar definiendo el sistema de ejes para poder definir el movimiento de un quadrotor, existen muchas alternativas dependiendo de la literatura, pero el usado mayormente es el que se presenta a continuación. El cual consta de un sistema de ejes fijo $\{X_{G,}\ Y_G,\ Z_G\}$, que es el sistema inercial, y el otro es un sistema referencia no inercial $\{X_{B,}\ Y_B,\ Z_B\}$.

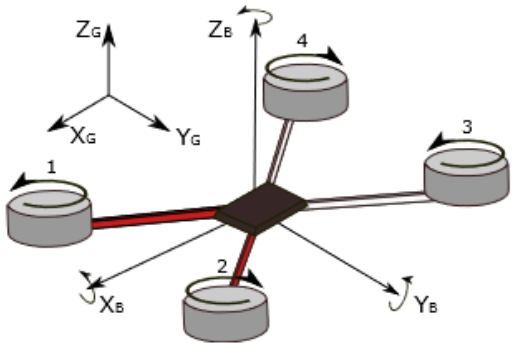


Figura 22. Estructura del quadrotor con sus Sistemas de referencia
Fuente: *(Rico, Maisterra, & Martínez, 2015)*

Dentro del marco de referencia del cuerpo se definen los vectores de velocidades lineales y angulares como se muestra a continuación:

Velocidad lineal:

$$\vec{V} = [u\ v\ w]^T$$

Velocidad angular

$$\vec{\Omega} = [p\ q\ r]^T$$

Para describir el comportamiento de un sólido rígido que se encuentra en movimiento y rotación, lo más recomendable es utilizar las ecuaciones de Euler, ya que estas permiten describir la orientación del cuerpo en función de los ángulos roll pitch, yaw (cabeceo, alabeo, y guiñada).

3.2 Dinámica del movimiento del cuerpo

De acuerdo a la ley de movimiento de Newton, la derivada del momento lineal es igual a la fuerza que actúa sobre el sólido, es decir

$$\frac{d\vec{p}}{dt} = \frac{d}{dt}(m.\vec{v}\,) = m\,.\vec{a}\ = \vec{F}$$

De manera similar se puede encontrar una ecuación análoga para el momento angular (L), el cual resulta del producto entre el momento de inercia (I) del solido rígido, considerando que el cuerpo es simétrico, y la velocidad angular (w) del cuerpo cuando este se encuentra en rotación. considerando que la derivada del momento de inercia con respecto al tiempo es igual 0, la ecuación está dado por:

$$\vec{M} = \frac{d\,\vec{L}}{dt} = \frac{d}{dt}(I.\vec{w}\,) = \frac{d\,I}{dt}(\vec{w}\,) + \frac{d\,\vec{w}}{dt}(I\,) = I\,.\vec{\Omega} \tag{1}$$

El momento de inercia se puede reducir a una matriz diagonal, en función de sus componentes como se muestra a continuación en la ecuación 2.

$$I = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \tag{2}$$

Dado que el sistema de referencia (SR) del cuerpo es un SR no inercial, ya que este se mueve de manera acelerada con respecto al SR fijo, es necesario introducir el efecto Coriolis en la ecuación (1), como se muestra a continuación:

$$\vec{M} = I.\dot{\vec{w}} + \vec{w}\;x\;(I.\vec{w}\,) \tag{3}$$

Y desarrollando los productos vectoriales de la ecuación anterior se obtiene sus componentes como se muestra en la ecuación (4)

$$\begin{aligned} M_x &= I_x\dot{p} + (I_z - I_y\,)qr \\ M_y &= I_y\dot{q} + (I_x - I_z\,)rp \\ M_z &= I_z\dot{r} + (I_y - I_x\,)pq \end{aligned} \tag{4}$$

De la misma manera que para la ecuación del momento angular, se obtiene la ecuación de traslación del quadrotor, es decir aplicando la segunda ley de Newton y añadiendo los efectos de Coriolis. Se tiene las siguientes ecuaciones

$$\vec{F} = \frac{d}{dt}\left(m.\vec{V}\,\right) = m\,\frac{d\vec{V}}{dt} \tag{5}$$

$$\vec{F} = m(\,\dot{\vec{V}} + \vec{w}\;x\;\vec{V}\,) \tag{6}$$

Y desarrollando los productos vectoriales se obtienen sus componentes como se muestra en la ecuación (7)

$$\begin{aligned} F_x &= m(\dot{u} + qw - vr\ ) \\ F_y &= m(\dot{v} + ru - wp\ ) \\ F_z &= m(\dot{w} + pv - uq\ ) \end{aligned} \tag{7}$$

3.3 Rotores

Teniendo en cuenta que el motor del quadrotor es controlado mediante impulsos eléctricos, se modela cada rotor (j) como un conjunto único de motor-hélice. Como se cuenta con cuatro rotores, en este caso: $j = 1, \dots, 4$.

3.3.1 Accionamiento del motor

La relación entre la señal de control del motor PWM, u_{jPWM}, y la velocidad angular del conjunto motor-hélice, w_j, estas señales representarían la entrada y salida del sistema. Para poder analizar la dinámica del sistema de un motor brushless, se debe aproximar a un sistema de primer orden en el dominio s de Laplace. (Rico, Maisterra, & Martínez, 2015). Donde K es la ganancia estática y τ la constante de tiempo del sistema.

$$P_j(s) = \frac{\Omega_j\ (s)}{u_{jPWM}(s)} = \frac{K}{\tau.s+1} \tag{8}$$

3.3.2 Aerodinámica de la hélice

Las hélices en los Quadrotores son consideradas como un perfil giratorio, cuya aerodinámica permite generar una tracción o impulsión, utilizando la potencia que le transmite el conjunto motor-hélice.

Un quadrotor se encuentra sometido a varios efectos aerodinámicos, Los dos principales efectos aerodinámicos que provoca la velocidad de giro del conjunto motor-hélice y la vibración de las aspas del quadrotor son: Par de arrastre (o drag) y Fuerza de empuje (o thrust)

Las ecuaciones que determinan el comportamiento del conjunto motor-hélice son: Fuerza de empuje y momento (par) de arrastre. Estas ecuaciones se pueden modelar como una dependencia cuadrática de la velocidad angular, en donde: k_t es una constante que caracteriza a cada conjunto motor-hélice y depende de los parámetros de cada rotor. k_d es una constante de proporcionalidad y depende de los parámetros de cada motor.

Momento de arrastre debido a cada rotor ($j = 1,2,3,4$)

$$\delta_j = k_d \,.\, (w_j)^2 \tag{9}$$

Fuerza de empuje ejercida por cada rotor ($j = 1,2,3,4$)

$$T_j = k_t \,.\, (w_j)^2 \tag{10}$$

La diferencia entre las velocidades de giro de cada uno de los rotores se traduce en el quadrotor en: tres momentos de giro, según los ángulos de orientación de Euler y también en una fuerza neta de sustentación. (Rico, Maisterra, & Martínez, 2015).

3.4 Pares sobre el Quadrotor

El principio físico que afecta a los Quadrotores es el de “equilibrio de momentos”. Esto ocurre ya que cuando el conjunto motor-hélice gira produce un momento, el cual si no se compensa hará que el quadrotor se encuentre en desequilibrio.

3.4.1 Pares principales

En un quadrotor se generan 4 pares, producidos por cada motor, los cuales se transmiten al cuerpo del vehículo. Para que se produzca este “equilibrio de momentos”, es necesario que los 2 motores opuestos tengan el mismo sentido de giro y justo el contrario en los otros 2 motores restantes. El desequilibrio entre estas fuerzas de empuje T_j generan los pares netos de rotación sobre el cuerpo: roll y pitch. Así mismo, el desequilibrio entre los momentos de arrastre δ_j de cada rotor genera el par neto de rotación en yaw.

roll

$$M_x = \frac{\sqrt{2}}{2}\, l\ (-T_1 + T_2 + T_3 - T_4) \tag{11}$$

pitch

$$M_y = \frac{\sqrt{2}}{2}\, l\ (-T_1 - T_2 + T_3 + T_4) \tag{12}$$

yaw

$$M_z = -\delta_1 + \delta_2 - \delta_3 + \delta_4 \tag{13}$$

Dónde: l representa la distancia entre el eje de cada rotor al centro del quadrotor

3.4.2 Efectos giroscopios

Los rotores producen un efecto giroscópico, el cual es un fenómeno dinámico que se da debido a que los mismo son cuerpos que tienen un movimiento de rotación alrededor de un eje y están sometidos a cambio de orientación por parte del quadrotor. Para cada rotor de manera genérica, el momento dinámico es:

$$\vec{M}^j = I^j \,.\, \dot{\vec{w}}^J + \vec{\Omega}\, x\, (I^j \,.\vec{w}^j\,) \tag{14}$$

Donde: I^j representa el momento de inercia de cada rotor.

Como el eje Z del sistema de referencia del cuerpo es paralelo a los ejes de rotación de los rotores, las componentes correspondientes al resto de ejes serán nulas. (Rico, Maisterra, & Martínez, 2015). Las componentes de $\vec{M}^j$ se reducen a la expresión (15)

$$\begin{aligned} M_x^j &= I_j w_j q \\ M_y^j &= -I_j w_j p \\ M_z^j &= I_j \dot{w}_j \end{aligned} \tag{15}$$

Donde: $I_j = I_z^j$ y $w_j = w_z^j$

El momento angular resultante que se produce sobre el quadrotor, considerando que los momentos de inercia de todos los motores son iguales es:

$$L_R = I_j(w_1 - w_2 + w_3 - w_4) \tag{16}$$

Finalmente, las componentes del momento dinámico del efecto giroscópico son:

$$M_x = L_R q$$
$$M_y = -L_R p \quad (17)$$
$$M_z = \dot{L_R}$$

3.4.3 Par total

Las componentes del par resultante sobre el quadrotor, resultan de la suma algebraica de las ecuaciones (11), (12), (13), los efectos giroscópicos (17). Estas componentes son:

$$M_x = \frac{\sqrt{2}}{2} l\,(-T_1 + T_2 + T_3 - T_4) + L_R q$$
$$M_y = \frac{\sqrt{2}}{2} l\,(-T_1 - T_2 + T_3 + T_4) - L_R p \quad (18)$$
$$M_z = -\delta_1 + \delta_2 - \delta_3 + \delta_4 + \dot{L_R}$$

3.5 Fuerzas sobre el Quadrotor

Para el cálculo de las fuerzas es necesario establecer ciertas consideraciones, como: el quadrotor cuenta con una Estructura rígida y simétrica, el centro de gravedad se encuentra en el centro físico del quadrotor.

3.5.1 Fuerza de Sustentación

La fuerza de sustentación es generada por el movimiento de las hélices, la cual actúa de abajo hacia arriba y cuya dirección es perpendicular al flujo de aire que genera el quadrotor al desplazarse (viento relativo), la representación de esta fuerza se puede observar en la figura 23.

Figura 23. Fuerza de sustentación
Fuente: *(Muñoz, 2018)*

La fuerza de sustentación total ($\boldsymbol{T_z}$) es igual a la fuerza de empuje total ($\boldsymbol{T_j}$), debido a que la fuerza de empuje de cada rotor tiene la misma dirección del eje Z en el sistema de referencia del cuerpo.

$$\boldsymbol{T_z} = \sum_{j=1}^{4} T_j = T \tag{19}$$

Los factores que pueden afectar a la fuerza de sustentación son: la forma del perfil de la hélice, la superficie sobre la que se ejerce la fuerza, la densidad del aire, la velocidad relativa del viento, y el ángulo de ataque de la hélice.

3.5.2 Fuerza de la Gravedad

La fuerza de gravedad se puede expresar en forma de matriz, siendo su dirección perpendicular a la superficie de la tierra, y su sentido hacia abajo (Muñoz, 2018), tal como se muestra en la ecuación (20)

$$\vec{F}_g^G = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \tag{20}$$

3.5.3 Fuerza total

Resulta de la suma algebraica de las componentes de las fuerzas de sustentación (19), de la gravedad (20), como se muestra en la ecuación (21).

$$\begin{aligned} F_x &= F_{gx} \\ F_y &= F_{gy} \\ F_z &= T_z + F_{gz} \end{aligned} \tag{21}$$

3.5.4 Ecuaciones en el sistema de referencia (SR) del cuerpo

Sustituyendo la ecuación (18) en (4) se obtiene la ecuación (22), la cual describe el movimiento rotacional del quadrotor en su sistema de referencia en función de los momentos de inercia.

$$\begin{aligned} \dot{p} &= \frac{l}{I_x}(-T_1 + T_2 + T_3 - T_4) + \frac{g_x}{I_x} \\ \dot{q} &= \frac{l}{I_y}(-T_1 - T_2 + T_3 + T_4) + \frac{g_y}{I_y} \\ \dot{r} &= \frac{l}{I_z}(-\delta_1 + \delta_2 - \delta_3 + \delta_4) \end{aligned} \tag{22}$$

Donde

g_x	$g_x = (I_y - I_z)q\, r + L_R\, q$
g_y	$g_y = (I_x - I_z)r\, p - L_R\, p$

De igual forma, sustituyendo la ecuación (21) en (7), se obtiene la ecuación (23), la cual describe el movimiento lineal del quadrotor en su sistema de referencia.

$$\begin{aligned} \dot{u} &= -qw + vr + \frac{F_{gx}}{m} - B_u\, .u \\ \dot{v} &= -ru + wp + \frac{F_{gy}}{m} - B_v\, .v \\ \dot{w} &= -pv + qu + \frac{T_z}{m} + \frac{F_{gz}}{m} - B_w\, .w \end{aligned} \tag{23}$$

3.5.5 Ecuaciones en el sistema de referencia (SR) de la tierra

Empleando los ángulos de Euler Φ, θ, Ψ (roll, pitch, yaw). Las ecuaciones de movimiento en el Sistema de referencia del cuerpo (26) y (27) se trasladarán al Sistema de referencia fijo de la tierra. (Rico, Maisterra, & Martínez, 2015). Al multiplicar estas matrices de rotación se tiene:

$$R_{x_G}^{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\Phi & sin\Phi \\ 0 & -sin\Phi & cos\Phi \end{bmatrix}$$
$$R_{y_G}^{B} = \begin{bmatrix} cos\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{bmatrix} \quad (28)$$
$$R_{z_G}^{B} = \begin{bmatrix} cos\Psi & 0 & sin\Psi \\ -sin\Psi & cos\Psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

ordenando:

$$R_G^B = RxGB\ .\ RyGB\ .\ RzGB \quad (29)$$

Se obtiene la matriz de rotación entre el SR del cuerpo y el sistema de referencia inercial de la tierra (31). Esta matriz de rotación aplicada sobre (28) permitirá obtener las aceleraciones lineales en el sistema de referencia de la tierra, si se conocen los ángulos de Euler.

Con estas aceleraciones y mediante una doble integración de halla la posición espacial del cuerpo $[x\ y\ z]^T$. Para calcular los ángulos de Euler, después de calcular $[p\ q\ r]^T$ a partir de la ecuación (26), se rotan dichas velocidades de acuerdo con:

$$\begin{bmatrix}\dot{\Phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix}=\begin{bmatrix}1 & sin\Phi\tan\theta & cos\Phi\tan\theta\\ 0 & cos\Phi & -sin\Phi\\ 0 & sin\Phi\sec\theta & cos\Phi\sec\theta\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} \quad (30)$$

Y finalmente se integran las velocidades angulares en el sistema de referencia de la tierra, y de esta forma, se conoce la orientación del cuerpo$[\Phi\ \theta\ \Psi]^T$.

3.6 Identificación de Parámetros

En la presente sección se pretende realizar algunas pruebas a fin de identificar experimentalmente algunos de los parámetros del modelo teórico analizado en la sección anterior (3.3). Para la obtención de los datos se utiliza un atmega que permite el envío de datos al PC mediante conexión serial

3.6.1 Parámetros del rotor.

Para la obtención de datos que permita obtener la curva estática y también el modelo dinámico lineal, analizado en la ecuación (8) del conjunto motor-hélice; se debe construir un pequeño circuito que consta de: un encoder el cual permite medir la velocidad angular (RPM) del motor a distintas señales de entrada y una galga extensiometrica. Posteriormente mediante MATLAB se realiza el tratamiento de la información.

3.6.2 Modelo estático de velocidad (Curva)

Para la elaboración de la curva estática se deben recopilar los valores estacionarios de la velocidad del motor para cada valor de PWM enviado. En la prueba se realizan variaciones en la “potencia” de PWM, es decir valores que van desde el 0% hasta 100%

en intervalos de 5%. Con estos valores Se obtiene la gráfica de la figura 24. Mediante el análisis de esta curva se puede establecer la zona lineal de trabajo de los motores, así como también la validez del modelo dinámico lineal.

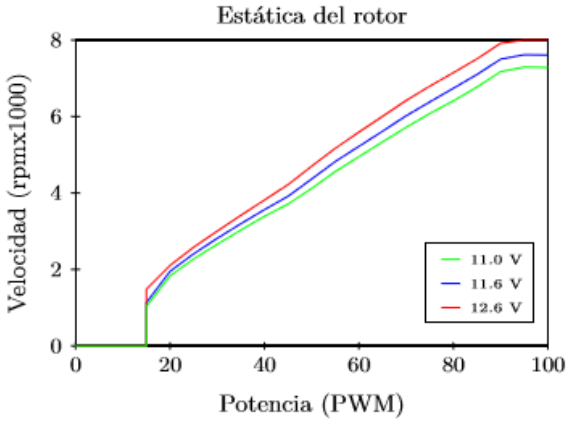


Figura 24 Velocidad del motor Vs PWM

En la figura 24. se puede analizar que existe una zona muerta para las entradas en el intervalo de 0% a 15% y una saturación en el intervalo de 90% a 100% aproximadamente, también se observa una zona con una tendencia lineal entre los intervalos del 15% al 95%. En esta zona lineal de operación se aprecia un incremento notable de la ganancia, más marcado en el intervalo de 40 a 85 %, esta ganancia lineal corresponde al parámetro K en la ecuación (8).

Para una adecuada toma de datos se repite el experimento varias veces, a fin de obtener los datos a distintos valores de carga de las baterías, y observando la perdida de potencia que ocasiona esto en los rotores. La ganancia estática obtenida para este caso experimental es:

$$K = 90$$

3.6.3 Modelo dinámico de velocidad

Para identificar los parámetros del modelo dinámico de la ecuación (8) del conjunto motor-hélice se recogen los datos obtenidos de la respuesta de la velocidad para un cambio a una entrada escalón de 50% a 65% de la entrada de porcentaje de PWM en el instante de tiempo $t = 0{,}15\ seg$. En la respuesta obtenida por Matlab se identifica una constante de tiempo $\tau = 0{,}05\ seg$. Como se puede observar en la figura 25.

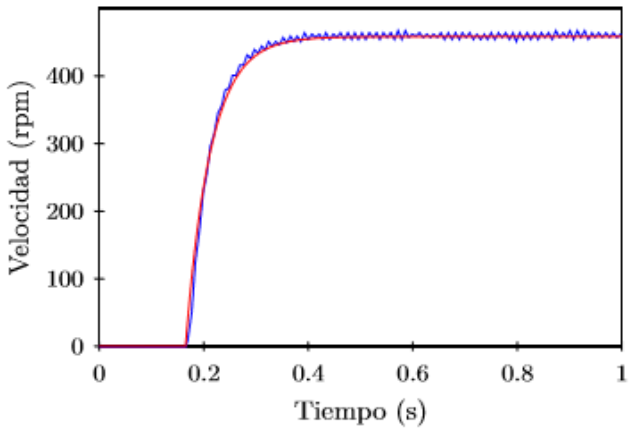


Figura 25. Respuesta dinámica del rotor

3.6.4 Fuerza de empuje

Para la medición de la fuerza de empuje que genera el motor, estudiada en la ecuación (10) es necesario utilizar un banco de pruebas, el cual consiste en: una galga extensiometrica acoplada a la base del conjunto motor-hélice y un encoder que permite obtener datos de velocidad. Como se puede observar en la figura 26.

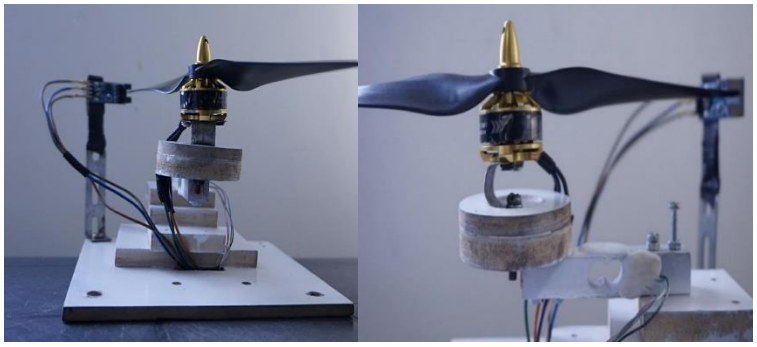


Figura 26. Maqueta utilizada para medir la fuerza de empuje generada

La prueba se realiza variando la señal de control PWM a el motor, se obtiene la velocidad de la hélice y también se registrar la fuerza en ese momento, como se observa en la figura 27. Los resultados velocidad-fuerza de empuje, se muestran en la figura 28. en la cual se logra comprobar el comportamiento cuadrático planteado en la ecuación (10), de donde se identifica:

$$K_t = 4{,}94 * 10^{-6} \frac{N}{(rad/seg)^2} \tag{33}$$

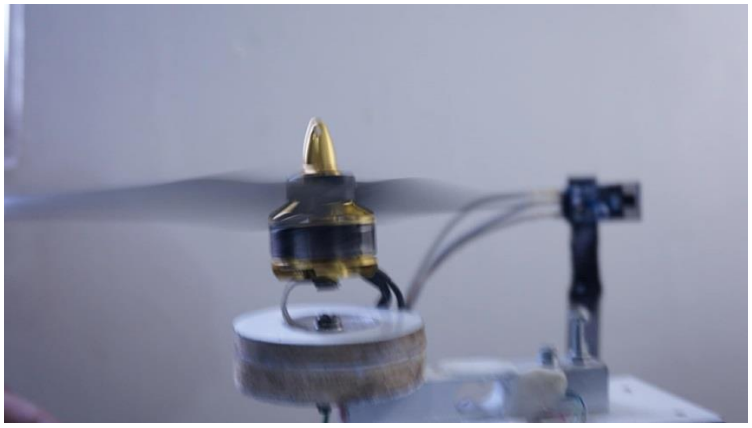


Figura 27. Funcionamiento de motor-hélice para la toma de datos

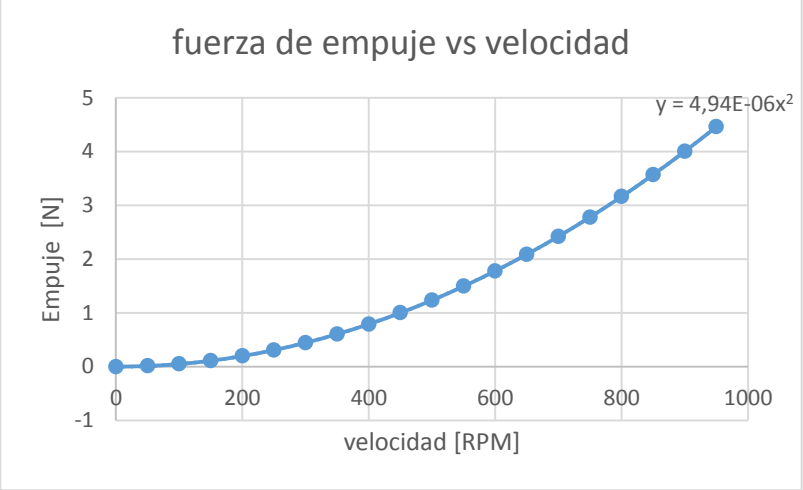


Figura 28. Datos adquiridos en la prueba de fuerza de empuje vs velocidad

3.6.5 Momento de arrastre

De manera similar que para la fuerza de empuje se procede a realizar las pruebas para el momento de arrastre estudiada en la ecuación (9) que ejerce el sistema motor-hélice. El momento de arrastre se calcula multiplicando la fuerza medida por el brazo del quadrotor (distancia mayor) de 43 cm de longitud, obtenido a. distintos valores de PWM. Los resultados se pueden apreciar en la figura 29.

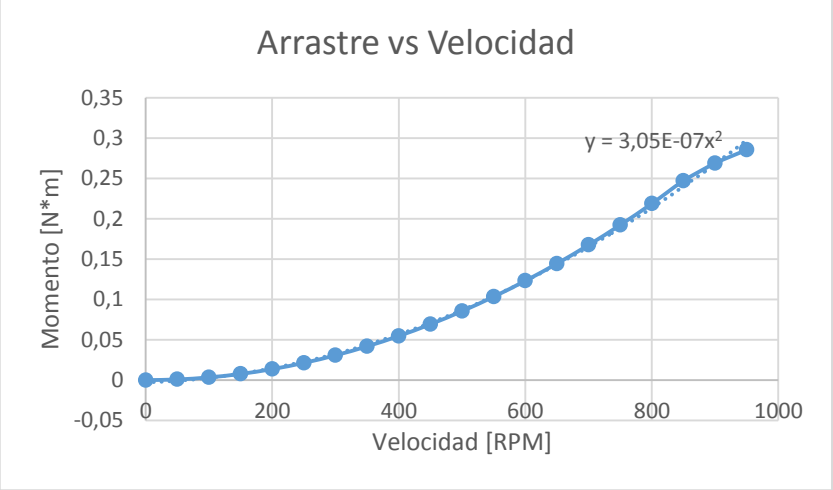


Figura 29. Datos adquiridos en la prueba de fuerza de empuje vs velocidad

3.6.6 Parámetros del solido rígido

Para esta sección, los parámetros se obtienen a partir de un modelo CAD del quadrotor (figura 30) realizado en el programa SolidWorks y algunos de ellos se pueden comprobar experimentalmente.

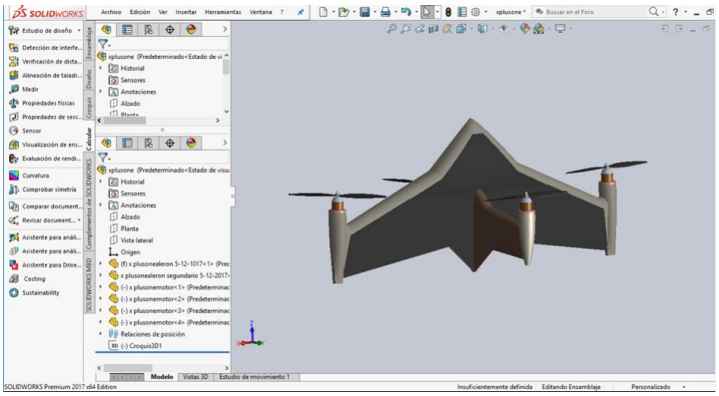


Figura 30. Modelo CAD del quadrotor X plusOne

Con la ayuda de una balanza digital, se registra una masa total del quadrotor de: $m = 1650\ gr$ incluidos también: la cámara y el sensor ZED por lo que es necesario ajus-

tar el material en el modelo CAD para poder obtener los datos correctos, como se puede verificar en la figura 31. La longitud del brazo mayor del Quadrotor se determina en $l = 43\ cm$.

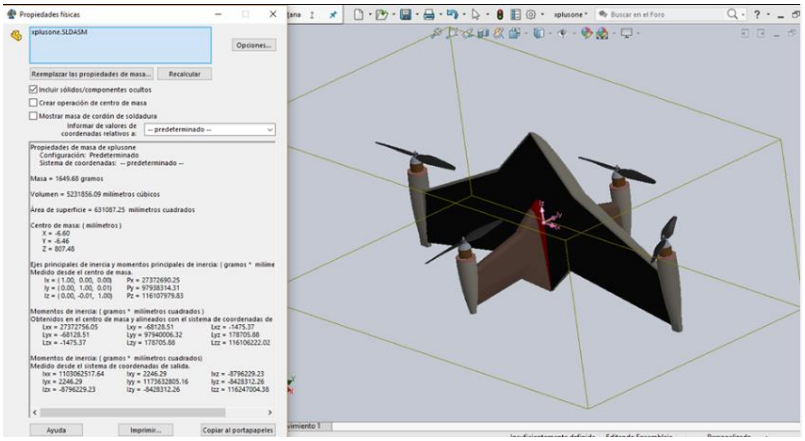


Figura 31. Parámetros del CAD X PlusOne

En la figura 32. se puede observar una tabla de las propiedades físicas del quadrotor, en ella se resumen algunas de las variables que se ocuparan en esta investigación como lo son: centro de masa, masa, momentos de inercia.

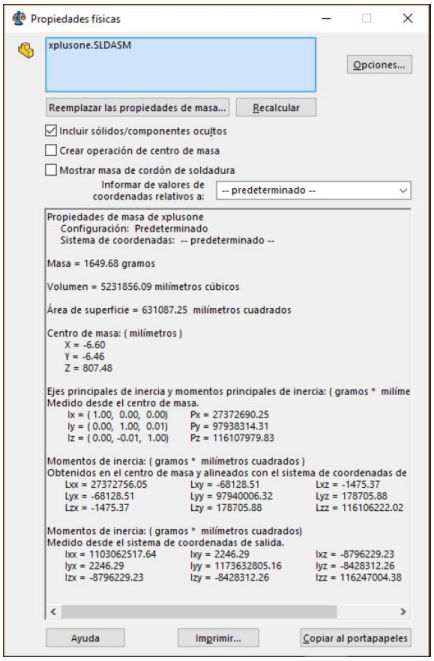


Figura 32. Tabla de propiedades físicas del quadrotor Xplus one

Los valores identificados para el momento de inercia del cuerpo analizado en la ecuación (2) son:

$$I = \begin{bmatrix} 1{,}1030 & 0 & 0 \\ 0 & 1{,}1736 & 0 \\ 0 & 0 & 0{,}11627 \end{bmatrix} kg\,.m^2 \qquad (34)$$

3.6.7 Resumen de parámetros

En la tabla 13 se presentan todos los datos obtenidos experimentalmente y mediante el modelo CAD del modelo teórico.

Tabla 13.

Resumen De Parámetros

Parámetros del Rotor			
Ganancia Estática $K \quad \left[\frac{rpm}{\%}\right]$	Const. / tiempo $\tau \quad [seg]$	Ganancia de empuje $k_t \ [N\frac{s^2}{rad}]$	Ganancia de arrastre $k_d \ [N\frac{m\,.s^2}{rad}]$
90	0,05	$4{,}94 * 10^{-6}$	$3{,}05 * 10^{-6}$
Parámetros del solido Rígido			
Inercia del cuerpo $[Kg\ m^2]$			Masa $m \ [Kg]$
$I_x = 1{,}1030$			$1650\ gr$
$I_y = 1{,}1736$			
$I_z = 0{,}11627$			

3.7 Simulación

Finalmente se realiza la simulación del modelo obtenido con los parámetros que se calcularon, donde se pretende recrear el comportamiento del quadrotor durante una misión de vuelo en un área de difícil acceso, en donde este quadrotor deberá cumplir una ruta dada, volando a una altura constante.

Toda la simulación se ha realizado usando el paquete Simulink de MATLAB, el cual tiene bloques dedicados en los cuales se puede incluir la dinámica rotacional del quadrotor. El Toolbox utilizado es el de Robotics de Peter Corke, el cual ofrece muchas funciones que son útiles para la simulación de la cinemática, dinámica y trayectorias de robots. (Corke, 2018)

El bloque de Simulink “Quadrotor Dinamic´s” que se observa en la figura 33 encerrado en un recuadro rojo, es empleado para implementar la cinemática y dinámica estudiadas anteriormente. Este bloque consta de una función “S”, la cual representa un sistema dinámico. Este bloque cuenta con una sintaxis específica y el usuario debe definir sus propios parámetros.

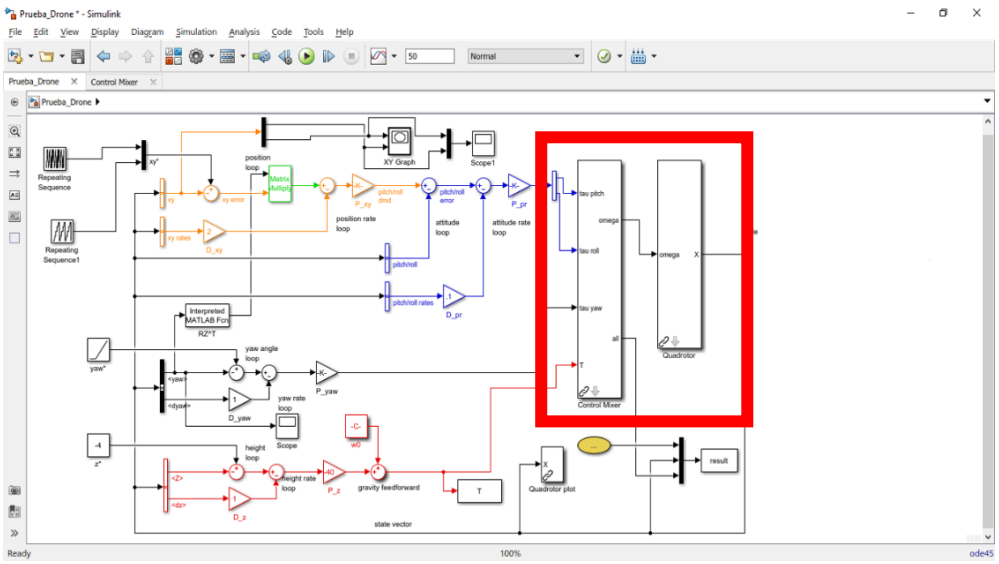


Figura 33. Simulación del modelo de quadrotor

3.7.1 Resultados

En la figura 34 y 35 se pueden observar los resultados de la simulación y respuesta de los parámetros obtenidos. En la primera figura se aprecia la ruta descrita por la aeronave en donde se simula el recorrido requerido para obtener las fotos necesarias para la construcción de una orto imagen.

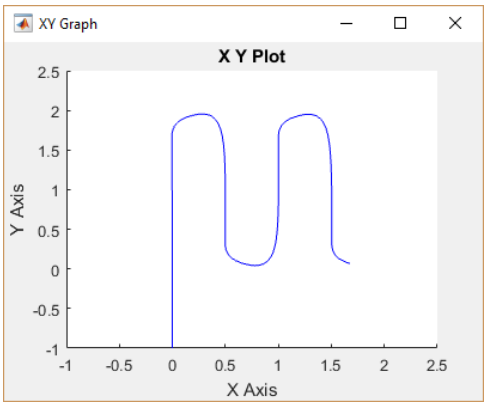


Figura 34. Simulación de la ruta de vuelo

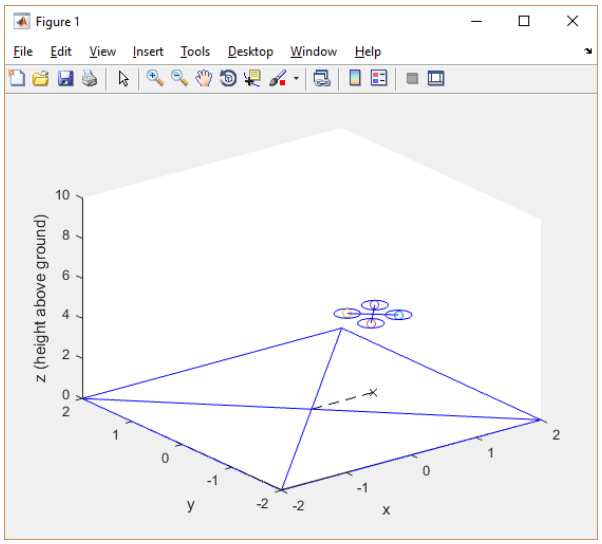


Figura 35. Simulación del despegue y vuelo de la aeronave

CAPITULO IV

ALGORITMO DE VUELO AUTÓNOMO, DETECCIÓN Y EVASIÓN DE OBSTÁCULOS

4.1 R.O.S

Como se analizó en el capítulo 2, ROS es un sistema operativo muy versátil que además ofrece ciertas herramientas, cuenta con una gran comunidad de desarrolladores y presenta muchas ventajas en el desarrollo de software para robots, en este caso un quadrotor. Por estas razones se justifica el uso de este sistema operativo, por ello en este capítulo se explica en que consiste este entorno y las principales herramientas que han sido necesarias para desarrollar esta investigación.

ROS no se puede instalar en cualquier sistema operativo, está limitada a plataformas basadas en Linux, principalmente se suele utilizar Ubuntu. En este proyecto se utiliza el sistema operativo Ubuntu Xenial. Se puede encontrar varias versiones de ROS como: Fuerte, hydro, índigo, kinetic, Lunar, etc. Pero en la versión en la que más se han realizado desarrollos y pruebas sobre navegación autónoma es en la versión Kinetic, es por esta razón que en este proyecto se utiliza esta versión.

Ros se encuentra estructurado por paquetes, los cuales pueden contener uno o varios nodos, cada uno con una función específica, como, por ejemplo: controlar la estabilidad de vuelo de un quadrotor, realizar localización y mapeos simultáneos (SLAM), obtener los datos de un sensor, etc. Estos nodos comparten información entre ellos a través de un sistema de mensajes de ROS.

Para que exista la comunicación entre los diferentes nodos es necesario de un nodo maestro, el cual almacena el registro de información de los nodos que se encuentran activos. Cuando se tiene un conjunto de nodos que trabajan en unión para una función en concreto por ejemplo la navegación, se llama Stack (pila).

Los sistemas de mensajes de ROS utilizan los tópicos (topic's) como ruta y canal de comunicación. Para que dos nodos se puedan comunicar, primero un nodo debe "publicar" un mensaje y el receptor debe estar suscrito para que pueda recibirlo. (Rivas R., 2017).

4.1.1 Herramientas de ROS

Como ya se mención anteriormente ROS ofrece ciertas herramientas que facilitan al usuario el control de diferentes plataformas robóticas, a continuación, se detallarán las que han sido necesarias para la realización de esta investigación.

4.1.1.a Roscore

Es el encargado de la comunicación entre nodos y también ejecutar el nodo master, del que dependen el resto de nodos. (García M., 2017). Cuando se ejecuta roscore también se pone en marcha un servidor de parámetros ROS y un nodo de registro llamado rosout.

4.1.1.a Roslaunch

Esta herramienta permite ejecutar uno o varios nodos, así como también configurar ciertos parámetros del servidor de parámetros ROS. Para la ejecución de esta herramienta es necesario uno o más archivos XML (*.launch) en los cuales se establecen los nodos a ejecutar y se configuran los parámetros.

4.1.1.a Rosrun

Rosrun brinda la posibilidad de ejecutar cualquier aplicación de un paquete sin necesidad de hacerlo en el directorio (Rivas R., 2017).

4.1.1.a Rostopic

Esta herramienta muestra información sobre los tópicos, esta información puede ser: lista de tópicos activos, nodos suscritos, etc. Rostopic cuenta con algunos comandos propios, como por ejemplo rostopic find que encuentra un tópico, o rostopic type que imprime el tipo de información de un tópico.

4.1.1.a Rosnode

La herramienta rosnode muestra información de los nodos como: publicaciones, suscripciones, conexiones, etc. Y al igual que rostopic cuenta con comandos propios similares que los de la herramienta anterior, con la diferencia que rosnode trabaja con los nodos.

4.1.1.a *Rviz*

Rviz es una herramienta de visualización 3D muy útil, este permite visualizar las lecturas de los tópicos que se encuentren activos. En la figura se puede observar la interfaz que ofrece Rviz

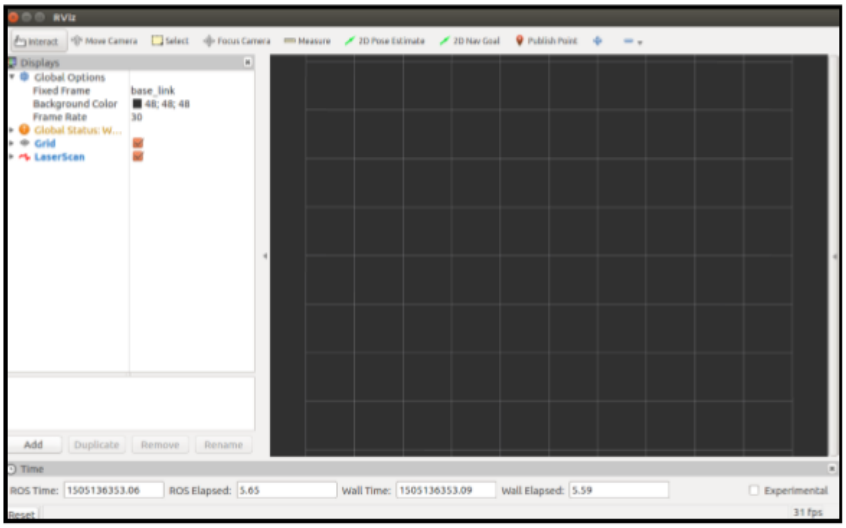


Figura 36. Interfaz de Rviz

Rviz ofrece algunos modos de operación, los cuales se muestran en la barra ubicada en la parte superior de la interfaz, dentro de estos se puede destacar 2 de ellos:

- ***2D pose Estimate:*** es de mucha utilidad cuando se realizan tareas de localización del robot. Este modo permite mandarle al robot una estimación de la posición en la que se encuentra. (Rivas R., 2017)
- ***2D Nav Goal:*** este modo de operación permite enviarle al robot la posición y la orientación a la que debe dirigirse el robot, de esta manera es como se define en un objetivo (goal) para que el quadrotor pueda navegar a la posición deseada.

4.1.1.a *Rqt_graph*

Rqt_graph es una aplicación que permite visualizar el estado del programa que se ejecuta mediante el uso de grafos. Estos grafos muestran la relación entre los nodos y los tipos de mensajes que se publican en los tópicos. Rqt_graph ofrece 3 modos de visualización: solo los nodos, nodos y tópicos activos, y todos los nodos y tópicos. Para ejecutar Rqt_graph se debe ingresar la siguiente línea de comando en una ventana de terminal.

```
$ rosrun rqt_graph rqt_graph
```

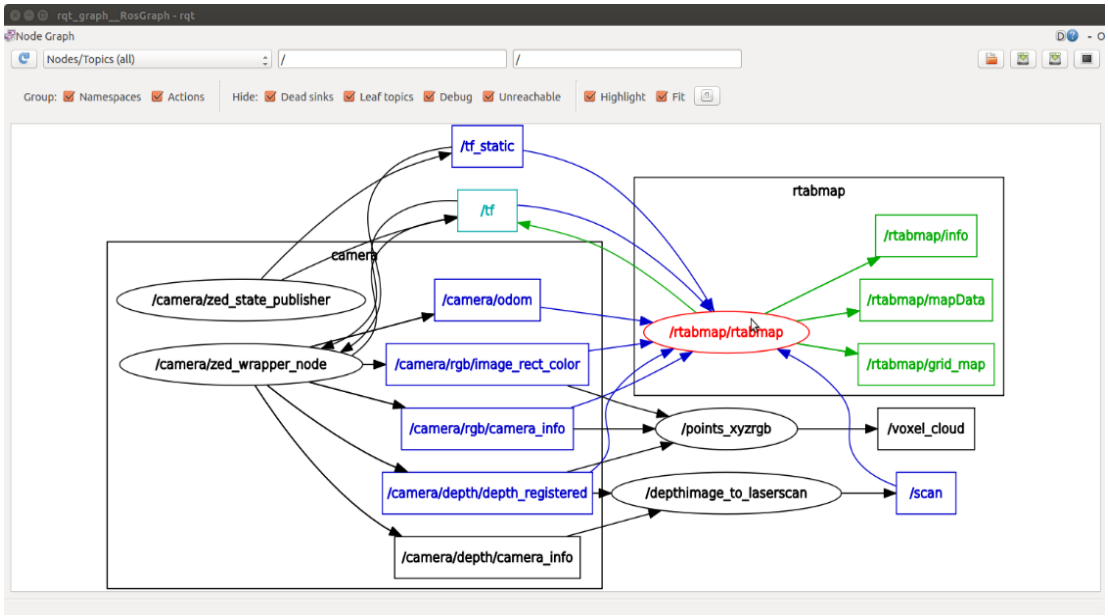


Figura 37. Ejemplo de visualización de las conexiones entre nodos

4.2 Pre-requisitos en el sistema

En esta sección se detallarán las instalaciones y los programas necesarios para el desarrollo del algoritmo de detección y evasión de obstáculos y para la construcción de un orto-mosaico.

4.2.1 Instalación de ros kinetic

ROS kinetic es compatible únicamente con plataformas de Ubuntu Wily y Xenial, es decir versiones de Ubuntu 15.10 y 16.04 respectivamente. Ya que en este proyecto se trabaja con la plataforma de Ubuntu 16.04 Xenial, no presentara ningún inconveniente posteriormente con algún paquete de ROS. A continuación, se resumen los pasos necesarios para la instalación.

1. Configuración de los repositorios de Ubuntu

Es necesario configurar estos repositorios ya que estos facilitan la instalación de nuevo software, a la vez que brindan un alto nivel de seguridad, ya que el software se construye para cada versión especifica de Ubuntu.

Los 4 repositorios principales son: Principal, Universo, Restringido, Multiverso. Se debe configurar en la pestaña “software de Ubuntu” para permitir los canales: “restringido”, “universo”, “Multiverso”.

2. Configuración de fuentes “*list*”

El segundo paso es configurar la computadora para que esta acepte la instalación de software de packages.ros.org.

3. Configuración de “keys”

Se debe ejecutar este comando para poder tener todos los permisos:

4. Instalación

Primero se debe verificar que el índice del paquete “Debian” se encuentre actualizado:

Una de las configuraciones predeterminadas que proporciona ROS para su instalación es:” Desktop-Full Install” la cual permite instalar librearías robot-

genéricas, simuladores 2D/3D, navegación y percepción 2D/3D. para su instalación se ejecuta:

5. Inicializar rosdep

Antes de poder utilizar ROS, es necesario inicializar “rosdep”. Este permite instalar las dependencias del sistema para que se pueda compilar y es necesario ejecutar algunos componentes principales en ROS.

6. Configuración del entorno

Es recomendable configurar las variables de entorno ROS, cada vez que se lanza un nuevo Shell:

7. Dependencias

En los pasos anteriores se ha instalado lo necesario para ejecutar los paquetes principales de ROS, pero para crear y gestionar propios espacios de trabajo ROS, es necesario asignar las dependencias.

En la tabla 14 se resumen los comandos utilizados en los pasos anteriores para la instalación de ROS kinetic. La primera columna representa el número de paso en la instalación y la segunda es el comando que se debe escribir en una ventana de terminal.

Tabla 14.

Resumen De Comandos Para La Instalación De Ros Kinetic

2	$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $ (lsb_release -sc) main"> /etc/apt/sources.list.d/ros-latest.list'
3	$ sudo apt-key adv --keyserver hkp: //ha.pool.sks-keyservers.net: 80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116
4	$ sudo apt-get update
4	$ sudo apt-get install ros-kinetic-desktop-full
5	$ sudo rosdep init
6	$ echo "source /opt/ros/kinetic/setup.bash" >> ~ / .bashrc fuente ~ /.bashrc
7	$ echo "source /opt/ros/kinetic/setup.bash" >> ~ / .bashrc fuente ~ /.bashrc

4.2.2 Instalación De Mission Planner

Lo primero que se necesita hacer es agregar las aptitudes necesarias del repositorio de Mono-proyect.com. Mono permite ejecutar aplicaciones de red en Ubuntu, para esto se ejecuta las siguientes líneas de comando que se muestran en la tabla 15 en una ventana de “terminal”. Al ejecutar el código nos pedirá el password para confirmar con la descarga.

Tabla 15.

Resumen De Comandos Para Instalación De Mono-Proyect

$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 3FA7E0328081BFF6A14DA29AA6A19B38D3D831EF
 	
$ sudo apt install apt-transport-https
 	
$ echo "deb https://download.mono-project.com/repo/ubuntu stable-xenial main" | sudo tee /etc/apt/sources.list.d/mono-official-stable.list
 	
$ sudo apt update

1. instalar Mono

Una vez se haya descargado el archivo, se procede a instalar Mono y sus dependencias

2. Acceder a los Permisos

Lo siguiente que se debe hacer es “habilitar” los permisos para poder conectarnos con nuestro vehículo, para hacerlo se necesita ser miembro del grupo de dialogo de Ubuntu.

Se debe reemplazar la palabra “username” por el nombre de usuario actual en Ubuntu. Y posteriormente se debe reiniciar el equipo para que los cambios sean guardados.

3. descargar Mission Planner

Es necesario descargar la versión de Mission Planner deseada, en este caso se utiliza la versión 1.3.30.

4. Extraer el archivo. Zip

Una vez se haya descargado se necesita extraer el archivo .zip en algún lugar, se debe crear una carpeta en donde se desee que se extraigan los archivos descargados.

5. Ejecutar el Mission Planner

Hay 2 formas de ejecutar el Mission Planner, mediante línea de comando o también se puede crear un archivo ejecutable launch para poder iniciar Mission planner más fácilmente.

En la tabla 16 se resumen los comandos utilizados en los pasos anteriores para la instalación de Mission Planner. La primera columna representa el N° de paso en la instalación y la segunda columna es el comando que se debe escribir en una ventana de terminal.

Tabla 16.

Resumen De Comandos Para La Instalación De Mission Planner

1	$ sudo apt-get install mono-complete festival
2	$ sudo usermod -a -G dialout username
3	$ wget http://firmware.eu.ardupilot.org/Tools/MissionPlanner/archive/MissionPlanner-1.3.30.zip
4	$ mkdir ~/missionplanner;unzip -d ~/missionplanner/ MissionPlanner-1.3.30.zip
5	$ cd ~/missionplanner;mono MissionPlanner.exe
5	$ gnome-desktop-item-edit --create-new ~/Desktop

4.3 Creación de un espacio de trabajo

Para poder empezar a trabajar con el sistema operativo ROS es necesario crear un espacio de trabajo, la cual es en donde se guardarán todos los paquetes que se comunicarán entre si durante la ejecución del programa final.

En una nueva ventana de terminal, se debe ejecutar el comando mostrado, con esto se creará una carpeta de inicio llamada “catkin_ws” con una subcarpeta llamada “src” (source) la cual es la que contiene todos los paquetes de trabajo, estos paquetes se irán detallando posteriormente a más profundidad.

```
$  mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/
$ catkin_make
```

En estas líneas de comando lo que se es crear la carpeta “catkin_ws” y su subcarpeta src, y luego se ubica en la careta creada. El comando “catkin_make” es una herramienta muy práctica que, básicamente permite ejecutar todos los archivos que se encuentre en el espacio de trabajo (catkin_ws). Este comando creara un archivo *.txt llamado “CMakeLists.txt” en la carpeta src. Adicionalmente, se crea en el directorio actual 2 carpetas: una llamada “build” y otra “devel”. La carpeta devel contiene varios archivos de instalación (*.sh)

4.4 Paquetes necesarios

Una de las principales ventajas que presenta ROS es la de la fácil comunicación entre paquetes o nodos, los cuales se van guardando o creando en la carpeta src anteriormente creada. A lo largo del desarrollo de la investigación se irán ocupando algunos

paquetes necesarios para el desarrollo de los algoritmos de detección y evasión de obstáculos, los cuales se explicarán a detalle a continuación.

Algunos de los paquetes son creados y desarrollados por los propios fabricantes de los sensores y controladores, como por ejemplo el caso del paquete creado por StereoLabs para ZED.

4.4.1 ZED ROS Wrapper

El paquete ZED ROS Wrapper fue creado y desarrollado por StereoLabs para la cámara ZED, y permite utilizar la cámara estereoscópica con ROS. Además, da acceso a cierto tipo de datos propios de la cámara como lo son:

- imágenes de la cámara izquierda y derecha
- Nube de puntos (Superficies)
- Mapa de profundidad
- Información de odometría (odometría visual)

Para empezar a trabajar con el paquete ZED ROS Wrapper en el entorno de ROS primero es necesario descargar la última versión del ZED SDK (Software development Kit), los cuáles son un conjunto de archivos y herramientas que proporciona el fabricante para la investigación y el desarrollo de aplicaciones robóticas, este archivo se encuentra en la página oficial de StereoLabs disponible para todos los sistemas operativos, en este caso se utiliza la versión 2.3 (para Ubuntu 16.04).

Para un correcto funcionamiento del archivo ZED SDK primero se deben descargar y actualizar todas las dependencias de CUDA de acuerdo con la tarjeta GPU disponible. En este caso se cuenta con una GPU: Geforce modelo GTX 950M del fabricante Nvidia. En la tabla 17 se pueden observar algunos de los tópicos a los que publica el paquete ZED ROS Wrapper y a que dato depende este mensaje publicado.

Tabla 17.

Tópicos Publicados Por El Paquete ZED Ros Wrapper

Tópico	Procedencia	Descripción
/ zed / left / image_rect_color	Cámara Izquierda	Imagen a color rectificada a la izquierda
/ zed / left / image_raw_color		Imagen a color sin rectificar / izquierda
/ zed / left / camera_info		Datos de calibración /cámara izquierda
/ zed / right / image_rect_color	Cámara Derecha	Imagen a color rectificada a la derecha
/ zed / right / image_raw_color		Imagen a color sin rectificar / derecha
/ zed / right / camera_info		Datos de calibración /cámara derecha
/ zed / depth / depth_registered	Nube de puntos y profundidad	Imagen del mapa de profundidad
/ zed / point_cloud / cloud_registered		Nube de puntos de las imágenes a color registradas
/ zed / odom	Odometría Visual	Posición absoluta 3D y orientación (relativa a zed_initial_frame.)

4.4.1.a Instalación de Toolkit 9.1 CUDA

CUDA (Compute Unified Device Architecture) es una plataforma con una arquitectura de cálculo paralelo desarrollado por la empresa Nvidia la cual accede a la GPU (unidad

de procesamiento gráfico) de la Pc y aprovechándola, y proporcionando un notable incremento en el rendimiento general del sistema.

Antes de proceder con la instalación, se debe verificar que se encuentren instalados los controladores de NVIDIA para Ubuntu en el equipo, a continuación, se detallan los pasos necesarios para realizar esta verificación.

a) Verificar que el entorno (Ubuntu 16.04) se encuentra apto y listo para las instalaciones del kit de herramientas CUDA. Para la actualización del paquete ***apt** y la instalación de las versiones más actualizadas de todos los paquetes instalados.

b) Verificar pre-requisitos como: que se hayan instalado correctamente los controladores de la GPU NVIDIA, verificar su funcionalidad, y que la GPU soporte CUDA. Para esto se utiliza el comando “lspci” el cual muestra los dispositivos que se encuentren conectados actualmente en los buses PCI. Como respuesta, si todo funciona correctamente debería aparecer en la lista de dispositivos los controladores NVDI.

c) Verificar la versión de compilador GCC instalada en el pc, y también la versión del núcleo e instalar los encabezados (headers) del kernel.

Tabla 18.

Resumen De Comandos Para La Verificación De Controladores Nvidia

a)	$ sudo apt-get update
b)	$ lspci | grep -i nvidia
c)	$ sudo apt-get install linux-headers-4.13.0-32-generic

Una vez realizados los pasos anteriores se puede proceder con la instalación del kit de herramientas de CUDA, para la instalación se deben seguir los siguientes pasos.

1. Lo primero es descargar el instalador Base de la página oficial de NVIDIA (Nvidia, 2018). Para esto es necesario registrarse y crear una nueva cuenta de desarrollador NVDI. En esta página se deben escoger los parámetros necesarios para el correcto funcionamiento del kit de CUDA, parámetros como: el sistema Operativo, Arquitectura, Distribución, versión y el tipo de instalación.

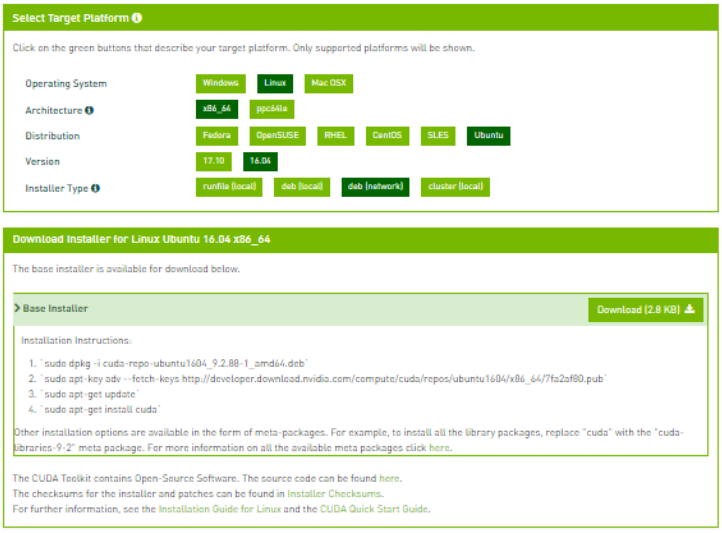


Figura 38. Ventana de parámetros para la descarga del instalador base

En la figura 38 se puede observar cuales fueron las opciones elegidas, según los requerimientos. y cuál fue el archivo base generado por NVIDIA.

2. Usando el instalador base; el cual contiene el repositorio de CUDA desde donde se descargará el kit de herramientas, lo primero es instalar los metadatos del repositorio y luego instalar la clave GPC del repositorio CUDA.

3. Instalar y agregar CUDA a la ruta desde donde se pueda ejecutar directamente los binarios de CUDA

4. En caso de que querer agregar permanentemente CUDA a la ruta, se debe modificar el archivo *.profile, en este archivo se crea el servicio de persistencia de NVIDIA el cual permitirá que nuestra GPU permanezca activa incluso cuando no haya aplicaciones aceleradas por la GPU.

5. Posteriormente se debe habilitar el servicio de persistencia y deshabilitar la regla “udev” predeterminada, y luego debemos reiniciar el sistema.

6. Finalmente se comprueba la versión del Kit de herramientas de NVIDIA CUDA

Tabla 19.

Resumen De Comandos Para La Instalación /Toolkit Cuda

1	$ sudo dpkg -i cuda-repo-ubuntu1604_9.1.85-1_amd64.deb
	$ sudo apt-key adv --fetch-keys
	http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64/7fa2af80
	.pub

2	$ sudo apt-get install cuda $ export PATH=/usr/local/cuda-9.1/bin${PATH:+:${PATH}}

3	$ cd /usr/lib/systemd/system
	$ sudo touch nvidia-persistenced
	$ sudo vi nvidia-persistenced

4	$ sudo systemctl enable nvidia-persistenced $ sudo vi /lib/udev/rules.d/40-vm-hotadd.rules

Después de instalar el Kit de herramientas de CUDA se puede proceder con la instalación del paquete ZED ROS Wrapper.

1. Lo primero es obtener las dependencias necesarias para la librería de nube de puntos.

2. Obtener el paquete del repositorio Github y colocarlo en la carpeta src de Catkin.

3. Finalmente compilar el paquete, para esto es necesario ubicarse en la carpeta raíz de catkin y ejecutar la siguiente línea de comando

Tabla 20.

Resumen De Comandos Para La Instalación /Zed Ros Wrapper

1	$ sudo apt-get install libpcl1 $ sudo apt-get install ros-kinetic-pcl-ros
2	$ cd ~ / catkin / src $ git clone https://github.com/stereolabs/zed-ros-wrapper

3	$ cd ~/catkin
	$ catkin_make zed-ros-wrapper
	$ source ./devel/setup.bash

4.4.1.a Prueba del Paquete

Para ejecutar el paquete ZED ROS Wrapper se necesita abrir una nueva ventana de terminal e ingresamos la siguiente línea de comando. El cual ejecutara el paquete, por lo que es necesario verificar que la cámara ZED esté conectada en un puerto USB 3.0. Una herramienta que nos permite visualizar de mejor manera los topic´s es “rviz”. Ejecutando la siguiente línea de comando se mostrará la imagen a color rectificada de la cámara ZED

```
$ rosrun image_view image_view image: = / camera / rgb / image_rect_color
```

4.4.2 Mavros

EL paquete Mavros funciona como un nodo de ROS, como si fuera una extensión de MAVLink con un proxy UDP (User Datagram Protocol), el cual es un protocolo basado

en el intercambio de datagramas, cuya principal ventaja es el envío y recepción de estos datagramas sin que se haya establecido una conexión previamente.

El paquete Mavros permite la comunicación con la estación de control terrestre (en este caso el software utilizado es MissionPlanner) a través del puerto serie, UDP o TCP, además este cuenta con un sistema completo que actúa como traductor entre ROS y MAVLink. Este paquete también proporciona herramientas de manipulación de parámetros y “waypoints”.

La función que cumple el nodo de Mavros es la de permitir la comunicación entre ROS y el controlador Pixfalcon, a través del Protocolo de comunicaciones MAVLink. Este nodo además permite la configuración de ciertos parámetros, tal es el caso del parámetro “~fcu_url”, y contiene información de la URL de conexión de la unidad de control de vuelo (Pixfalcon) y la velocidad de transmisión.

Por default el parámetro fcu_url viene en la dirección: /dev/ttyACM0:57600, a una velocidad de transmisión de 57600 baudios, pero para que se pueda conectar a el Quadrotor fue necesario cambiar esta dirección, por la que se muestra:

/ dev/ttyUSB0:5700

4.4.2.a Instalación del Paquete Mavros

Existe la posibilidad de instalar este paquete por 2 diferentes métodos: la primera, una instalación binaria la cual genera paquetes pre-compilados desde un repositorio. Y la segunda posibilidad es una instalación desde la fuente la cual nos da acceso a todos los sub-paquetes y nos permite modificarlos, es por esta razón que hemos decidido realizar una instalación desde la fuente.

1. Lo primero que se necesita hacer es instalar MAVLink, usando la referencia de Kinetic para todas las distribuciones de ROS, después se puede instalar Mavros.
2. Posteriormente se crea el espacio de trabajo y sus dependencias, también se requiere instalar el plug-in “GeograhicLib”, necesario para ciertas librerías internas de Mavros, y finalmente se compila desde la carpeta fuente: *catkin_ws.*

Tabla 21.

Resumen De Comandos Para La Instalación /Mavros

1	$ sudo apt-get install python-catkin-tools python-rosinstall-generator -y
	$ rosinstall_generator --rosdistro kinetic mavlink | tee /tmp/mavros.rosinstall
	$ rosinstall_generator --upstream mavros | tee -a /tmp/mavros.rosinstall
2	$ wstool merge -t src /tmp/mavros.rosinstall
	$ wstool update -t src -j4
	$ instalación de rosdep --from-paths src --ignore-src -y
	$ ./src/mavros/mavros/scripts/install_geographiclib_datasets.sh
	$ catkin build

4.4.3 Rtab-map ROS

El paquete de RTAB-MAP es un paquete creado por Mathieu Labbe, bajo la licencia BSD, el cual en su interior funciona con el algoritmo ***RGB-D SLAM*** basado en un detector de cierre de bucle global con algunas restricciones en tiempo real. El algoritmo RGB-D SLAM es uno de los más fiables métodos para realizar SLAM, el cual emplea imágenes en color (RGB) y una capa de profundidad (D por su sigla en inglés Depth).

En general el paquete Rtab-map ROS permite generar nube de puntos 3D de un entorno y también crear un mapa de cuadricula de ocupación 2D, el algoritmo que se corre en su interior extrae las características (features) y lanmarks de la información obtenida por la cámara RGB-D (López Torres, 2016) en este caso la cámara ZED es la que brinda estos datos, y las compara con las características obtenidas en iteraciones anteriores.

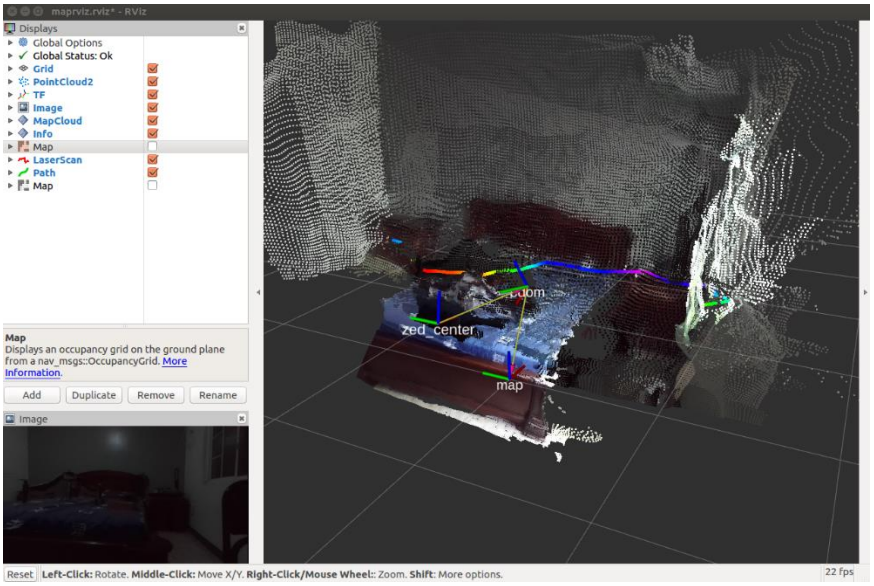


Figura 39. Nube de puntos 3D que genera Rtab-map ROS

En la figura 39 se puede ver la ventana de Rviz, el cual es una herramienta propia de Ros que permite visualizar los tópicos que se están publicando de los diferentes nodos, en la figura se muestra el Tópico “/rtabmap/mapData” el cual genera la nube de puntos 3D del entorno

4.4.3.a Instalación del paquete Rtab-map ROS

Al igual que el paquete anterior existen 2 maneras de instalarlo, en esta ocasión se decidió instalarlo por el método de “instalación binaria”. Para esto se ejecuta la siguiente línea de comando.

```
$ sudo apt-get install ros-kinetic-rtabmap-ros
```

4.4.4 Depthimage to Laserscan

El paquete Depthimage to Laserscan es un paquete desarrollado por Chad Rockey. El nodo de Depthimage to Laserscan convierte una imagen de profundidad a un scanner laser, y genera un escaneo 2D basado en los parámetros que se han definido anteriormente, estos datos son necesarios para la navegación y localización de nuestro quadrotor.

El nodo Depthimage_to_laserscan está suscrito a 2 topic´s, los cuales vendrían a ser parte de las entradas al nodo, estos topic´s son:

- ***Image*** este es el tópico que brinda la imagen de entrada, por defecto viene con la dirección: ***"/camera/depth/image_raw",*** pero para que este nodo reciba las imágenes generadas por el nodo de *ZED ROS Wrapper* es necesario cambiar la dirección por la siguiente

/camera/depth/depth_registered

- ***Camera_info*** Este tópico da la información de la cámara para la generación de la imagen, publica parámetros propios de la cámara como: dimensiones de la imagen captada, resolución, el tipo de imagen, etc. La dirección necesaria para que este tópico reciba estos parámetros de la cámara ZED es:

/camera/depth/camera_info

La salida del paquete Depthimage_to_laserscan se publica en el topic "scan", El cual por defecto tiene la siguiente dirección.

/scan

Este paquete también permite la configuración de ciertos parámetros como:

- Cantidad de filas de pixeles que se usaran para generar el laser
- Tiempo entre escaneo
- Rangos mínimos y máximos de escaneo (metros)
- Identificación del marco de referencia del scanner laser

4.4.4.a *Instalación de Depth to Laserscan*

1. Para la instalación del paquete Depth to Laserscan lo primero que se tiene que hacer es obtener el paquete del repositorio Github y lo colocarlo en nuestra carpeta src de Catkin.

2. Finalmente se compila el paquete, para eso es necesario ubicarse en la carpeta raíz de catkin y ejecutar las líneas de comando indicadas en la tabla 22.

Tabla 22.

Resumen De Comandos Para La Instalación /Depth To Laserscan

1	$ cd ~ / catkin / src
	$ git clone https://github.com/ros-perception/depthimage_to_laserscan.git
2	$ cd ~/catkin
	$ catkin_make depthimage_to_laserscan

4.4.5 Move_base

El paquete move_base fue desarrollado por Eitan Marder-Eppstein, este paquete hace uso de las “Acciones”, servicio de ROS que permite comunicarse con uno o varios nodos a través de la interacción cliente servidor. El cual, dado un objetivo en el entorno real, esta intentara cumplirla haciendo uso de una base móvil. El nodo move_base en su estructura interna, vincula un planificador global y un planificador local los cuales son necesarios para realizar la navegación en un entorno real. Además, el nodo move_base, toma los datos que los sensores del quadrotor captan del entorno y crea dos mapas de cortes 2D, los cuales consisten en una cuadricula de ocupación 2D de estos datos obtenidos.

El paquete move_base permite mover el quadrotor a las posiciones deseadas haciendo uso de la “stack de navegación”, esta stack toma la información de odometría y los datos enviados por los sensores y emite comandos de velocidad para posteriormente enviárselo mediante mensajes a una base móvil. Una figura de la stack de navegación se muestra a en la figura 40.

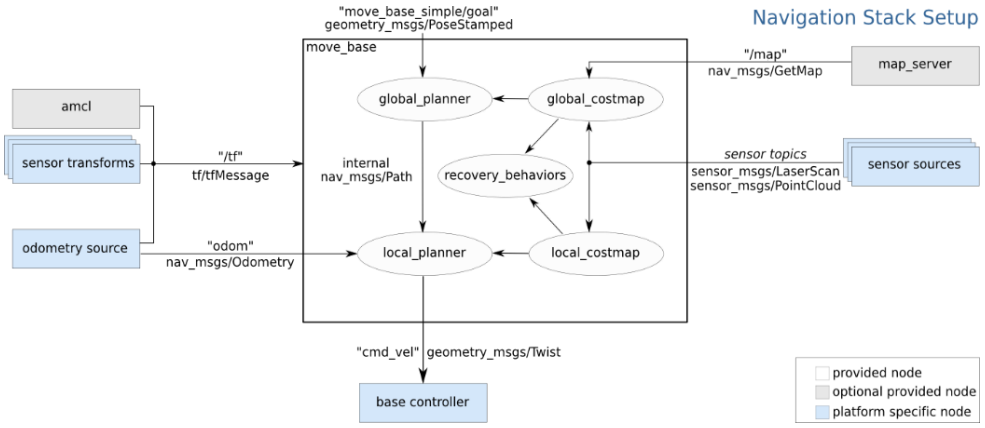


Figura 40. Stack de navegación de ROS

Fuente: (ROS.org, 2018)

El paquete move_base conforma el núcleo de esta stack de navegación conjuntamente con los paquetes “map_server” y “AMCL”, los cuales cumplen la función de: crear un mapa con los datos del escaneo laser y localizarse usando un mapa existente, respectivamente. En la figura 40. Se puede apreciar diferentes recuadros que representan los nodos, los recuadros azules varían según la propia estructura del robot y los senso-

res empleados, los recuadros de color gris son nodos opcionales y los recuadros blancos representan los nodos necesarios los cuales son proporcionados para todos los sistemas.

El nodo Move_base está suscrito al tópico “move_base_simple / goal” el cual proporciona un seguimiento del estado de ejecución del objetivo, el tipo de mensaje que se envía es “geometry_msgs / PoseStamped”. Adicionalmente el nodo move_base se suscribe a tópicos de acciones, cuyos principales tópicos se detallan a continuación:

- ***move_base / goal.*** *Este* es el tópico proporciona un objetivo para que move_base pueda seguirlo en el entorno real. El tipo de mensaje y la dirección que se maneja en este tópico es.

move_base_msgs / MoveBaseActionGoal

- ***move_base / cancel.*** Este tópico permite enviar una solicitud para cancelar un objetivo de ruta específico. El tipo de mensaje y la dirección que se maneja en este tópico es.

actionlib_msgs / GoalID

El nodo move_base también publica en el tópico “cmd_vel”, una secuencia de comandos de velocidad las cuales son interpretadas y ejecutadas por el controlador del robot, el tipo de mensaje que se envía es “***geometry_msgs / Twist***”.

4.4.6 Instalación de Move_base

1. Para la instalación del paquete Move_base lo primero que se debe que hacer es obtener el paquete del repositorio Github y lo colocarlo en la carpeta src de Catkin.

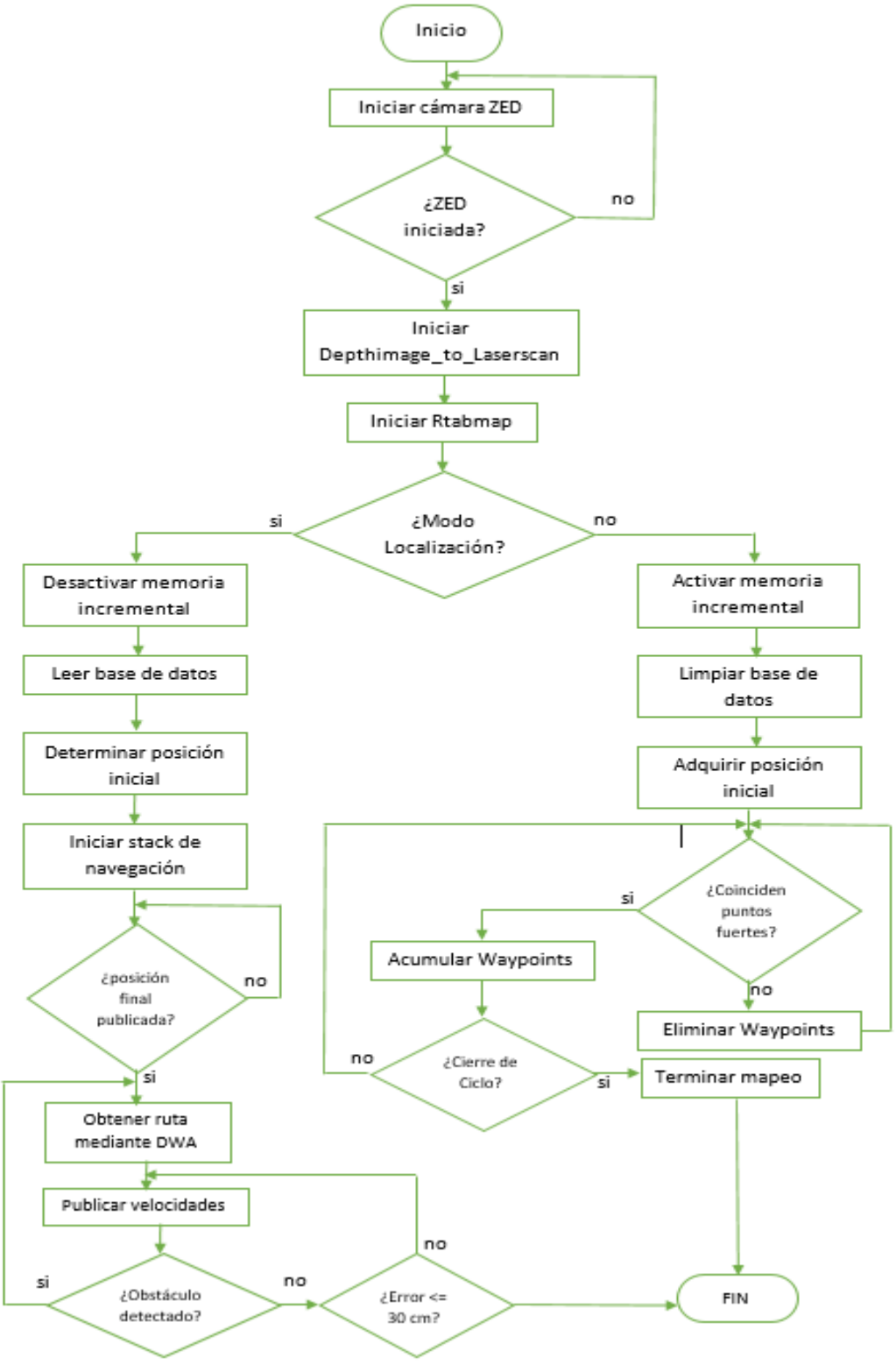
2. Finalmente se compila el paquete, para eso es necesario ubicarse en la carpeta raíz de catkin y ejecutar las líneas de comando mostrado en la tabla 23

Tabla 23.

Resumen De Comandos Para La Instalación /Move Base

1	$ cd ~ / catkin / src
	$ git clone https://github.com/ros-planning/navigation.git
2	$ cd ~/catkin
	$ catkin_make move_base

4.5 Implementación de algoritmo


Inicio
Iniciar cámara ZED
¿ZED iniciada?
no
si
Iniciar Depthimage_to_Laserscan
Iniciar Rtabmap
¿Modo Localización?
si
no
Desactivar memoria incremental
Leer base de datos
Determinar posición inicial
Iniciar stack de navegación
¿posición final publicada?
no
si
Obtener ruta mediante DWA
Publicar velocidades
¿Obstáculo detectado?
si
no
¿Error <= 30 cm?
no
Activar memoria incremental
Limpiar base de datos
Adquirir posición inicial
¿Coinciden puntos fuertes?
si
Acumular Waypoints
no
Eliminar Waypoints
¿Cierre de Ciclo?
no
si
Terminar mapeo
FIN

4.5.1 Conexión

En esta parte se tratará sobre los pasos que se han llevado a cabo para poder comunicar el quadrotor con el entorno de ROS, conexión requerida para realizar las actividades de localización y navegación autónoma.

El quadrotor Xplus One incorpora en su interior un módulo de control Pixfalcon, el cual permite comunicar con una PC a través de su módulo de telemetría. Este módulo de telemetría es de la marca Holybro y trabaja en una banda de 915 MHz.

4.5.1.a Conexión ArduPilot Con ROS

1. Instalar MAVROS. Para un uso más simple, se recomienda instalar RQT ejecutando el siguiente comando en una ventana de terminal.
2. Instalar catkin-tools:
3. Inicializamos SITL para lanzar una nueva instancia de SITL se ejecutar la línea de comando mostrado en la tabla 24.
4. Crear un nuevo directorio El siguiente paso es abrir en una nueva terminal, y crear un nuevo directorio de inicio.
5. Posteriormente se copia el archivo de inicio predeterminado de MAVROS para ArduPilot, luego se debe abrir para editarlo:

Tabla 24.

Resumen De Comandos Para La Conexión / Mavros

1	$ sudo apt install ros-kinetic-mavros
2	$ sudo apt-get install ros-kinetic-rqt ros-kinetic-rqt-common-plugins ros-kinetic-rqt-robot-plugins
3	$ sudo apt-get install python-catkin-tools
4	$ sim_vehicle.py -v ArduCopter --console --map

5	$ cd ardupilot
	$ launch mkdir
	$ cd launch

6	$ roscp mavros apm.launch apm.launch $ gedit apm.launch

Es necesario modificar la primera línea del archivo. launch para poder conectar con SITL cambiándola por la siguiente línea de código:

```
<arg name = "fcu_url" default = "udp: //127.0.0.1: 14551 @ 14555" />
```

Una vez se haya modificado. Se guardan los cambios y se ejecuta mediante:

```
$ roslaunch apm.launch
```

Y finalmente la instalación está completa.

4.5.2 Mapeado

Como ya se estableció anteriormente, es necesario contar con un mapa para poder hacer uso de la stack de navegación de ROS, es decir crear un mapa para que el quadrotor sea capaz de localizarse y navegar en el entorno que se le presente. Para ello es necesario contar con un sensor adecuado que sea capaz de reconocer el entorno.

En este proyecto se utiliza como sensor la cámara estereoscópica ZED la cual responde muy bien ante este tipo de aplicaciones ya que además de generar una nube de puntos y un mapa de profundidad ofrece la ventaja de extraer información de la odometría visual de esta. En la figura 41 se puede apreciar la nube de puntos que se forma al escanear un entorno en el visualizador Rviz.

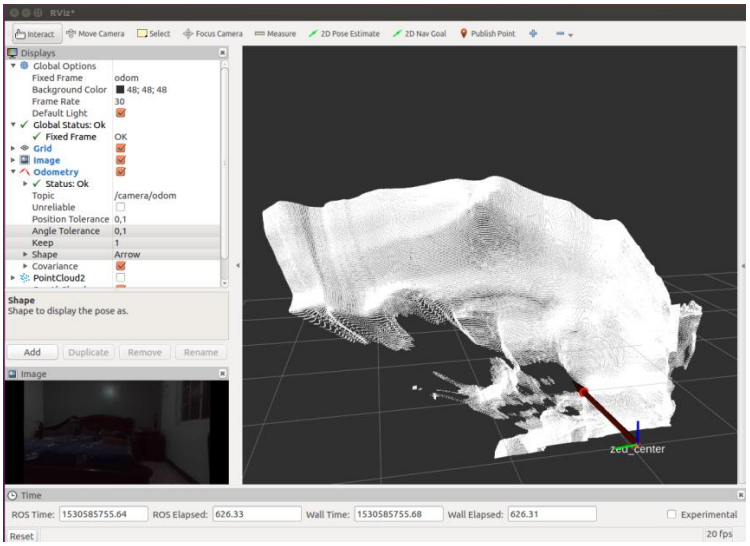


Figura 41. Vista de nube de puntos y odometría en RVIZ que genera la cámara ZED.

Una buena manera de comprender el estado actual del programa es con la ayuda de la herramienta Rqt_graph, en la figura 42 se observa los nodos y cada uno de los tópicos que se encuentran activos al ejecutar el paquete ZED ROS Wrapper.

Figura 42. Grafo de la cámara ZED en el visualizador rqt_raph

El siguiente paso es convertir esta imagen de profundidad generada por el nodo de la cámara ZED a un scanner laser y así poder generar un escaneo 3D y posteriormente el paquete Rtab-map pueda ocupar esta información en la construcción del mapa. Como se revisó en la sección anterior el nodo encargado de realizar esta conversión es el nodo Depthimage_to_Laserscan el cual debe estar suscrito a los tópicos "/camera/depth/depth_registered" y "/camera/depth/camera_info" para luego publicar en el nodo "/Scan". En la siguiente figura se puede observar la comunicación nodos-tópicos antes mencionada en Rqt_graph.

Figura 43. Vista en Rqt_graph de nodos Camera y Laserscan

Ahora si se puede proceder a realizar el escaneo laser de la zona de difícil acceso, en la figura 44 se observa una ventana de en Rviz, en ella se puede visualizar una línea de colores el cual representa el escáner. Esta línea varia su color de acuerdo a la profundidad, en donde rojo representa que está más cerca y el color azul que se encuentra más lejos.

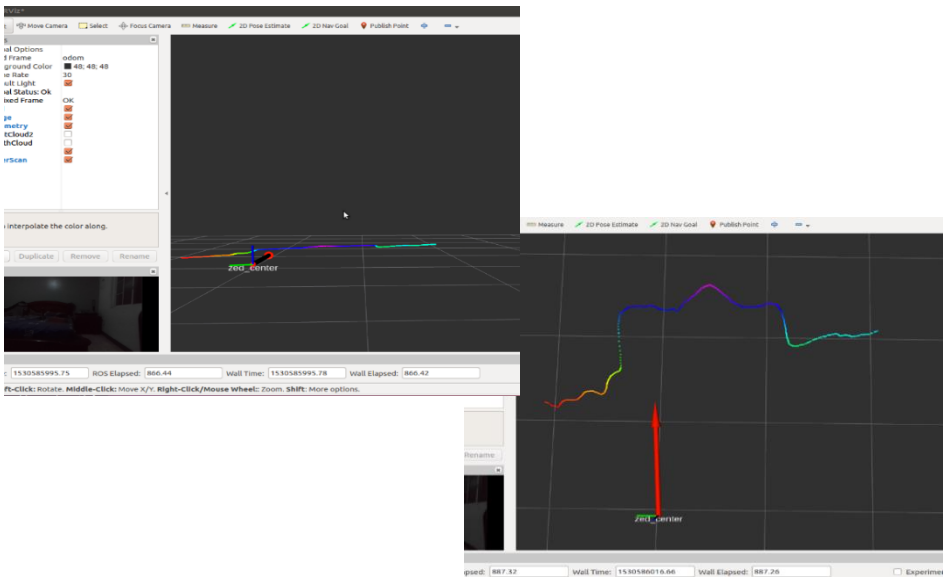


Figura 44. Vista en Rviz del tópico /Depthimage_to_Laserscan

Para que se pueda realizar un buen mapeado es necesario que el quadrotor se desplace por todo el entorno y así la cámara ZED pueda “leer” de manera correcta cada uno de los detalles presentes en esta zona de difícil acceso. En este punto es necesario que el quadrotor se comunique con ROS, es decir que se haya comprobado el correcto funcionamiento del paquete Mavros para que este pueda controlar los movimientos del quadrotor ejecutando un programa base.

Para ello se ejecuta el archivo “***dronemap.py***” (tipo launch) previamente creado, el cual consta de un menú que permite armar el quadrotor, cambiar los modos de vuelo del mismo, ingresar de forma manual usando el teclado la distancia que debe recorrer. Esto se logra modificando el tópico encargado de la posición local e ingresando pasos de 0,5 metros. Para ejecutar este programa se debe ejecutar la siguiente línea de comando:

“ $ Rosrun xplusone dronemap.py”. Este archivo se puede encontrar el código completo en el Anexo A.

Figura 45. Menú principal del programa dronemap.py

Una vez ejecutado el archivo *. launch, se debe ejecutar también el paquete Rtab-map el cual permitirá la creación del mapa global gracias al algoritmo interno que extrae las características y marcas del entorno escaneado. El nodo Rtab-map, para la generación del mapa requiere conocer la posición inicial del quadrotor al empezar a generar el mapa y hacer un seguimiento al sistema de referencia de la cámara ZED conociendo en cada momento su posición, para esto hace uso de los tópicos “tf_stactic” y “/tf”, el nodo

Rtab-map también toma la información publicada en el tópico “/Scan” y los tópicos del nodo ZED ROS Wrapper de la cámara ZED.

Se puede visualizar en la figura 46 el esquema de relaciones entre los nodos y tópicos que se encuentran activos hasta esta parte del programa, haciendo uso de la herramienta Rviz. En esta parte del programa ya se encuentran comunicados y trabajando en conjunto los paquetes y nodos: ZED ROS Wrapper, Rtab-map y /Depthimage_to_Laserscan respectivamente.

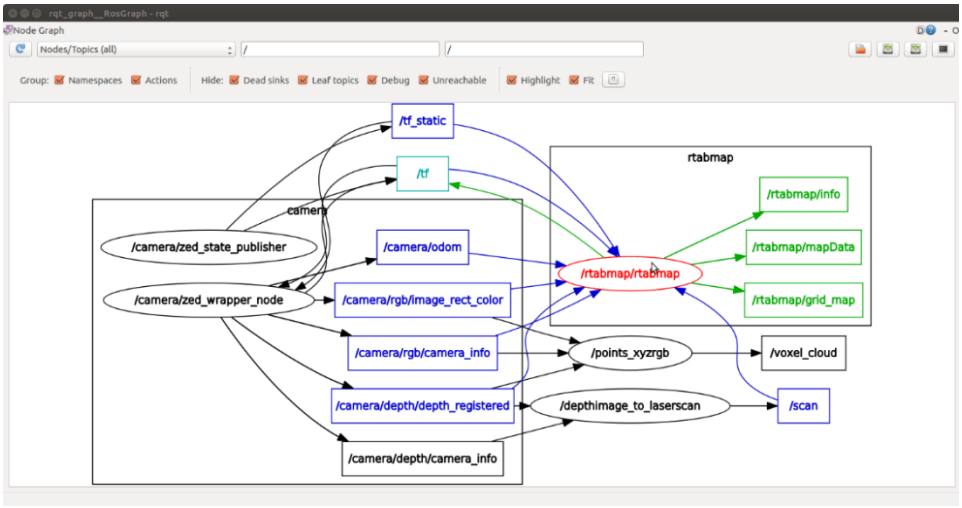


Figura 46. Vista en Rqt_graph de los nodos activos en la etapa de mapeado

Después de ejecutar estos archivos se empezará a crear automáticamente el mapa con los detalles que pueda capturar la cámara ZED y a medida que el quadrotor se vaya desplazando por el entorno se ira generando un mapa global con más elementos y detalles. Una vez se haya realizado un barrido escaneando la zona que se va a mapear este se va a guardar.

En este proyecto se han realizado diferentes mapas de varios entornos, como ejemplo un entorno cerrado el cual es un dormitorio en el que se encuentra una cama, sobre esta se encuentra el quadrotor, en este ambiente el suelo es completamente liso y la presencia de potenciales obstáculos es muy baja a pesar de ser un espacio muy pequeño para el vuelo del quadrotor. En las siguientes figuras se puede observar la secuencia de creación de este mapa en Rviz.

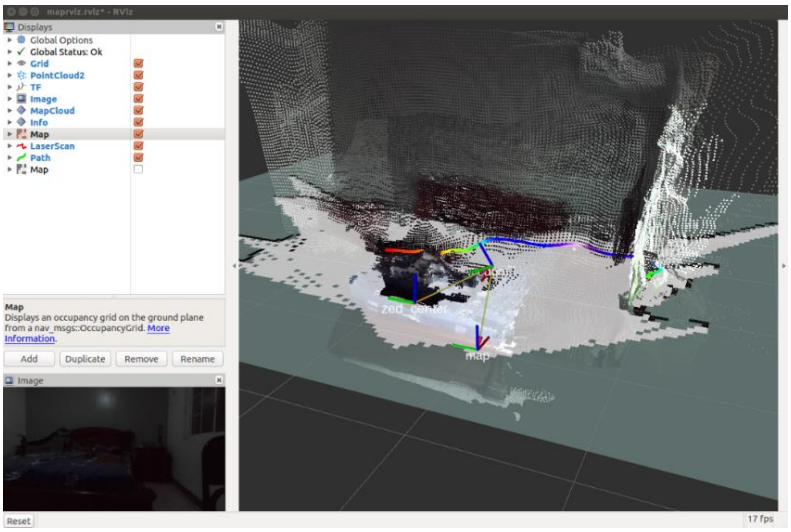


Figura 47. vista en Rviz de la creación de un mapa en interiores

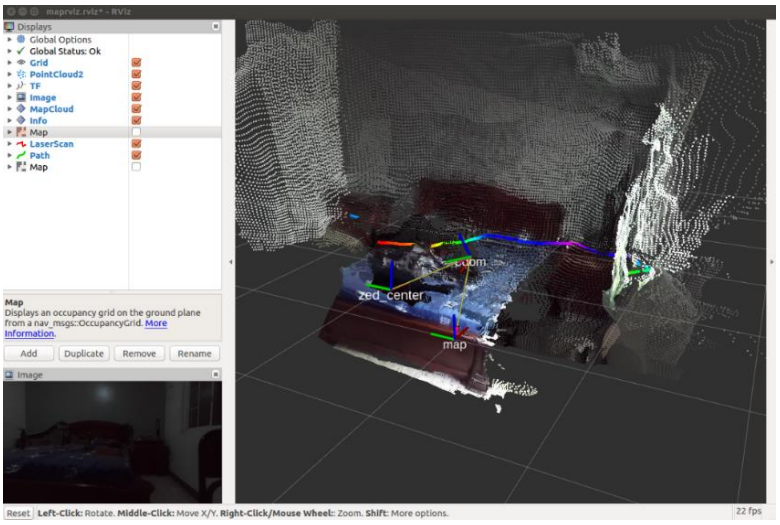


Figura 48. Vista en Rviz del mapa generado en interiores

Otro entorno mapeado es en exteriores, en esta prueba se realizaron 2 mapas: El primero es un mapa de un pequeño parque del colegio 9 de octubre en la ciudad de Machala. Este entorno se caracteriza por tener el suelo irregular, arbustos de mediana altura y la presencia de 3 árboles que podrían ser considerados como posibles obstáculos para el quadrotor, en este entorno se cuenta con espacio suficiente para una adecuada navegación. En las siguientes figuras se puede observar la secuencia de creación de este

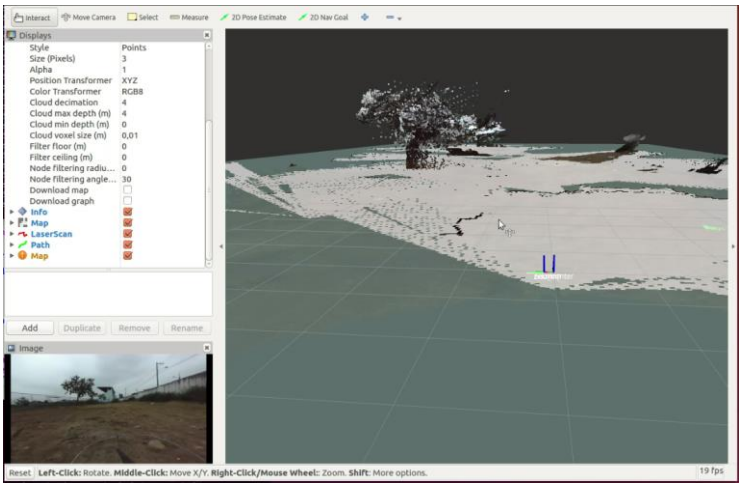


Figura 49 Vista en Rviz de la creación del mapa 1 en exteriores

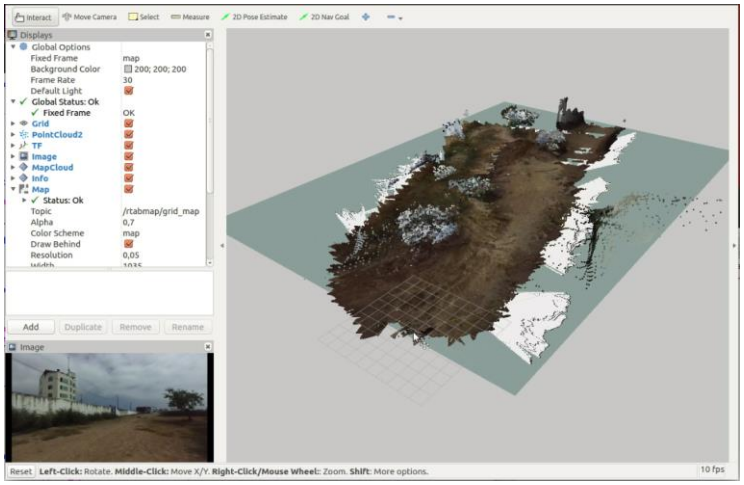


***Figura 50*.** Vista en Rviz del mapa 1 generado en exteriores

El segundo mapa de exteriores es de los patios de la universidad de las Fuerzas Armadas ESPE-l. Este entorno se caracteriza por tener una forma cuadrada, el suelo regular y sin excesivos obstáculos para el quadrotor, además posee una pileta de gran tamaño y se encuentra cercado en gran parte por las paredes del edificio de la ESPE-L. Este entorno es el más adecuado para poder realizar las pruebas de navegación autónoma ya que no solo cuenta con el espacio y la forma necesaria para el movimiento del

quadrotor, sino que brinda la posibilidad de introducir diferentes obstáculos, generando así nuevas rutas de vuelo. En las siguientes figuras se puede observar la secuencia de creación de este mapa en Rviz.

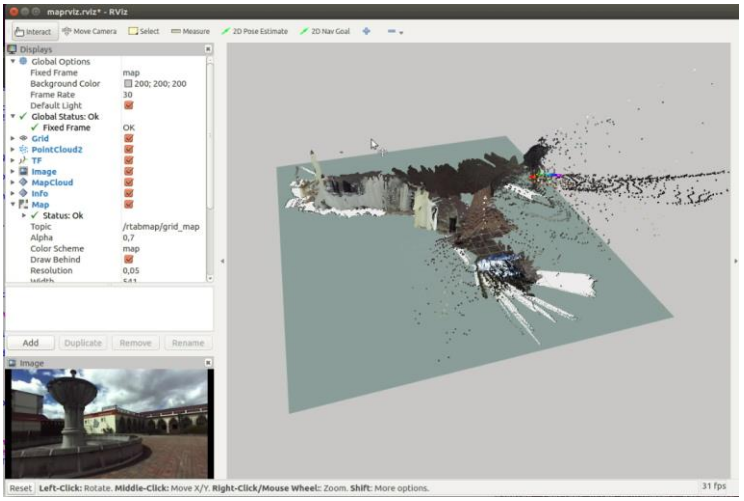


Figura 51. Vista en Rviz de la creación del mapa 2 en exteriores

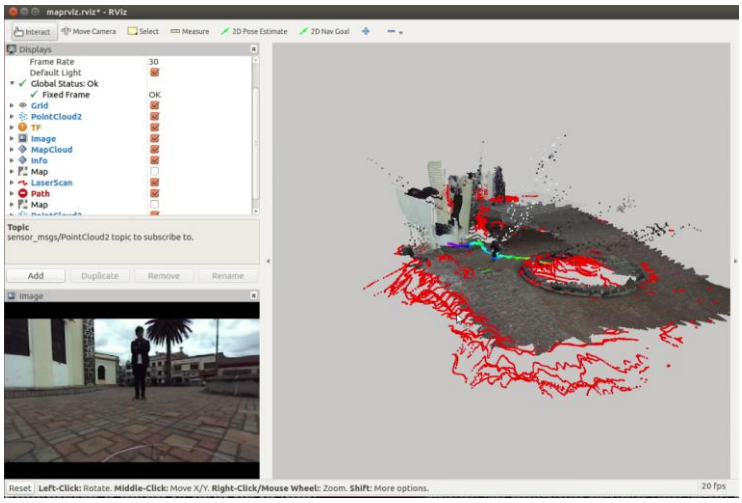


Figura 52. Vista en Rviz del mapa 3D generado en exteriores

4.5.3 Localización

Para que el quadrotor pueda moverse de manera autónoma es indispensable que antes tenga la capacidad de encontrarse dentro del mapa anteriormente generado, a este proceso se le conoce como localización. Este proceso se lleva a cabo dentro del paquete de Move_base el cual realiza una comparación entre los puntos fuertes de las imágenes captadas por la cámara ZED y el mapa global anteriormente creado. El nodo Move_base incorpora en su interior un sistema de localización probabilística para que un robot se pueda desplazarse en 2D. (Rivas R., 2017)

4.5.4 Navegación

El último paso en esta investigación es poder realizar una navegación autónoma en donde el quadrotor pueda detectar y evadir los obstáculos que se le presenten en el entorno. Para poder llevar a cabo este paso se hace uso de la pila de navegación, la cual como ya se explicó anteriormente se necesita información de la odometría del quadrotor, los datos de los sensores y un objetivo a donde el quadrotor debe desplazarse, para que luego se envíen mensajes de velocidad a una base móvil. El tipo de mensaje que se publica para el control del desplazamiento es del tipo “twist” y generalmente se publica en el tópico “/cmd_vel”

Para que se ejecute esta pila de navegación primero se debe ejecutar el nodo move_base y configurar los parámetros necesarios para la navegación como: tamaño del quadrotor, velocidades máximas y mínimas de desplazamiento, rango máximo y mínimo

de detección de obstáculos, tópicos a los q se publica, frecuencia de muestreo, etc. Estos parámetros pueden ser configurados creando 4 archivos (*.Yaml), estos archivos de configuración se pueden encontrar a detalle en el anexo B y son:

- Base_local_planner_params.yaml
- Costmap_common_params.yaml
- Global_costmap_params.yaml
- Local_costmap_params.yaml

El paquete Move_base también incorpora un planificador global y un planificador local. El panificador global combina los datos de odometria con los mapas de costes generados por Move_base. La ruta que se crea en el plan global se calcula antes de que el robot comience a moverse, buscando el camino más corto y utilizando el cálculo potencial cuadrático para proximarse a su objetivo teniendo en cuenta los obstáculos ya conocidos. (Rivas R., 2017). El planificador local dado un mapa crea una trayectoria para que el quadrotor pueda desplazarse de un punto a otro, en el proceso de la creación de esta trayectoria también se crea una función de costes de todas las posibles trayectorias, esta función controla las velocidades lineales y angulares adecuadas para que el quadrotor pueda navegar y una vez detectado y evitado el obstáculo el quadrotor retomara su planificación global.

El planificador local también simula el estado actual del quadrotor, por cada dato de velocidad enviado y predice lo que podría suceder en un periodo corto de tiempo, ade-

más evalúa cada trayectoria utilizando métricas como la proximidad a la que se encuentran los obstáculos, la proximidad al seguimiento de la ruta global, la velocidad de desplazamiento y descarta las trayectorias en donde se produzca una colisión.

Antes de realizar cualquier prueba de navegación es importante verificar que el quadrotor se encuentre comunicado con Ros, es decir que el nodo de Mavros este activo y listo para recibir y posteriormente traducir los mensajes de velocidad que se envían al controlador del quadrotor. En la siguiente figura se puede observar la comunicación nodos-tópicos antes mencionada en Rqt_graph.

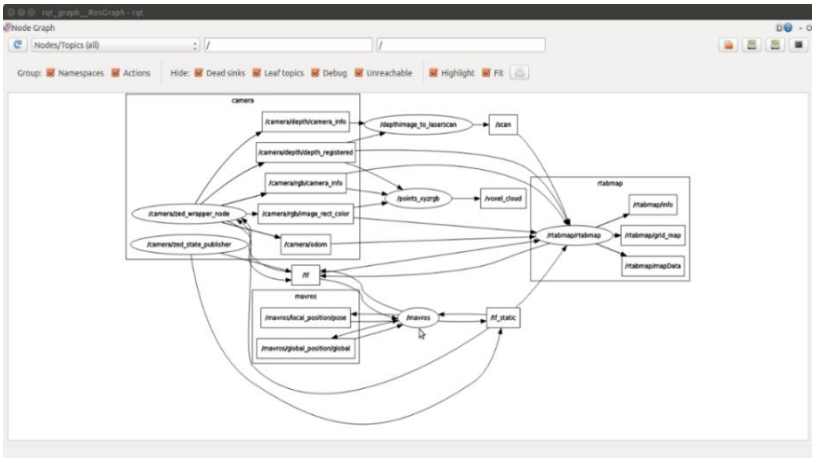


Figura 53. Vista en Rqt_graph de los nodos activos en la etapa de mapeado

Finalmente, para poder realizar una navegación autónoma se debe indicarle al quadrotor un objetivo de posición y orientación con respecto a un sistema de referencia, en este caso el quadrotor se posicionará con respecto al sistema de referencia del origen del mapa. Una de las maneras de especificarle a el quadrotor un objetivo, es con la herramienta que ofrece Rviz “2D Nav Goal” la cual se encuentra en la barra superior del visualizador Rviz, como se observa en la figura 54. El tipo de mensaje que se envía es

"MoveBaseActionGoal" y se publica en el tópico "/move_base_simple/goal" al cual el nodo Move_base se encuentra suscrito

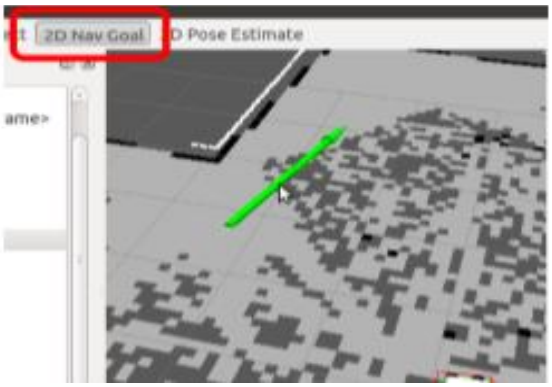


Figura 54. Vista de la herramienta "2D Nav. Goal" del visualizador Rviz.

Para indicarle al quadrotor este objetivo (posición y orientación) simplemente se debe hacer un clic en la parte del mapa a la que se desea llegar, luego aparecerá una flecha de color verde la cual se debe arrastrase hasta que se encuentre en la orientación deseada. En la siguiente figura se puede visualizar el procedimiento antes mencionado.

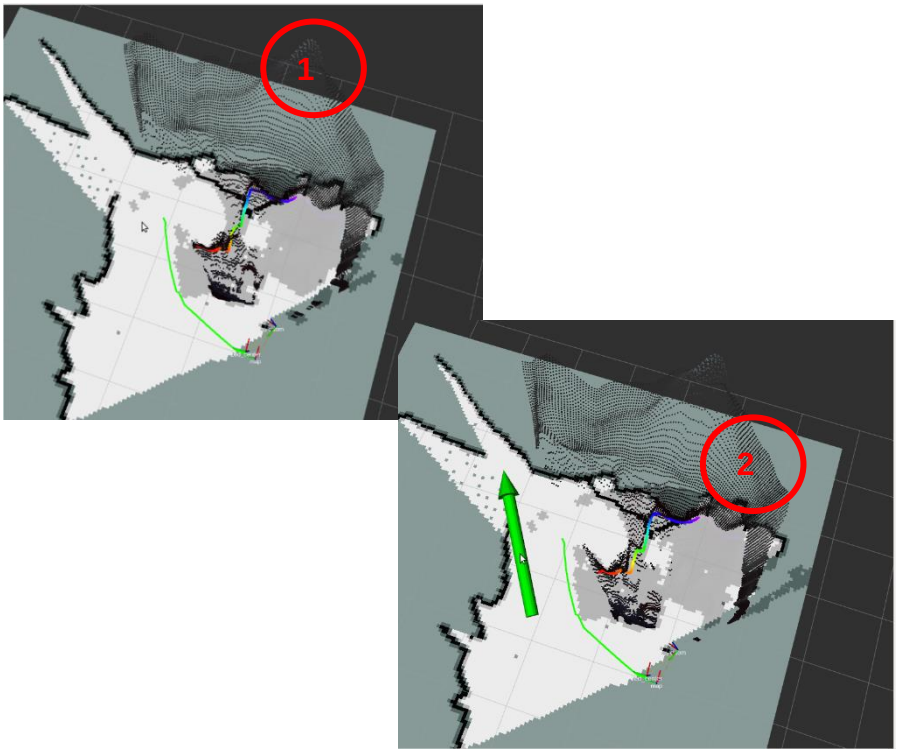


Figura 55. Proceso de asignación de objetivo (posición-orientación)

Una vez marcada la posición objetivo se publica un mensaje en el tópico "/move_base_simple/goal" con dicha información la cual es recibida por el nodo "move_base" quien se encarga de encontrar la ruta ideal realizando múltiples iteraciones con las posibles rutas, una vez definida esta se marca de color verde como se muestra en la figura 56. Para que se desplace el quadrotor se publican velocidades en el controlador de vuelo, disminuyendo el error entre la posición del marco de referencia del quadrotor y la ruta establecida por el nodo.

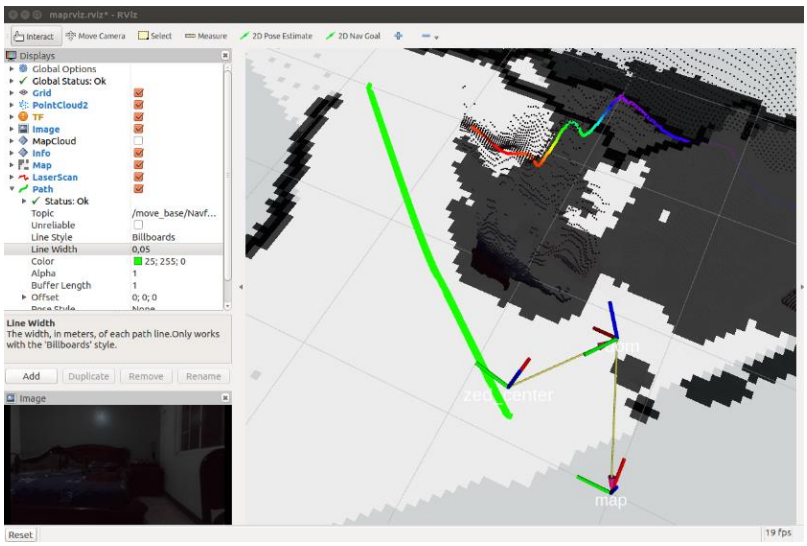


Figura 56. Ruta generada

Se puede visualizar en la figura 57 el esquema de relaciones entre los nodos y tópicos que se encuentran activos para la ejecución final del algoritmo de detección y evasión de obstáculos, haciendo uso de la herramienta Rviz.

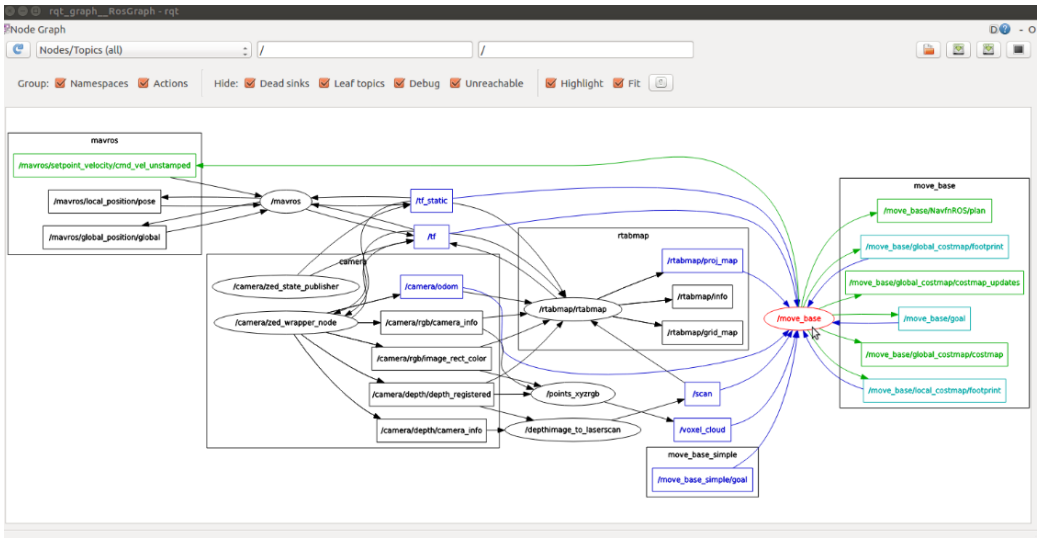


Figura 57. Vista Rqt_graph de programa final

CAPITULO V

ANÁLISIS Y RESULTADOS.

5.1 Área de detección (obstáculos) útil del quadrotor

Para poder determinar cuál es el área en el cual se puede realizar una óptima detección de obstáculos sin comprometer la misión, se realizaron pruebas ejecutando los paquetes de: la cámara (ZED ROS Wrapper), el scanner de laser 2D (Depth to Laserscan) y el mapa 3D (Rtab-Map) y verificando con la herramienta Rviz.

La relación de distancias entre el quadrotor y el obstáculo se las realizó ejecutando el archivo "l***aunch_rtab.launch***" (ruta: ~/catkin ws / src / toscan /launch) y se midió con un flexómetro las marcas. Al ejecutar este archivo, también se ejecutará Rviz y permitirá visualizar el mapa 3D que se ha generado y en él mostrará una línea de colores el cual representa el Scanner 2D con el cual se realiza la detección de obstáculos. Esta línea cambia sus colores de acuerdo a la profundidad medida, cuando se encuentre de color rojo representa un obstáculo para el algoritmo.

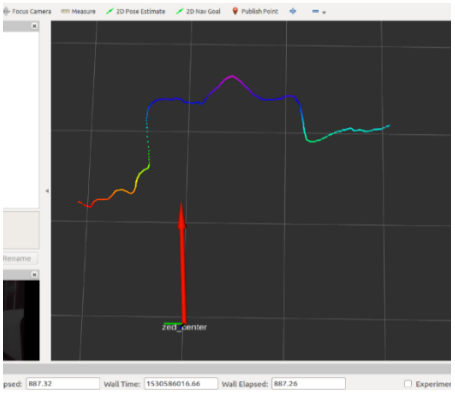


Figura 58. Laser 2D

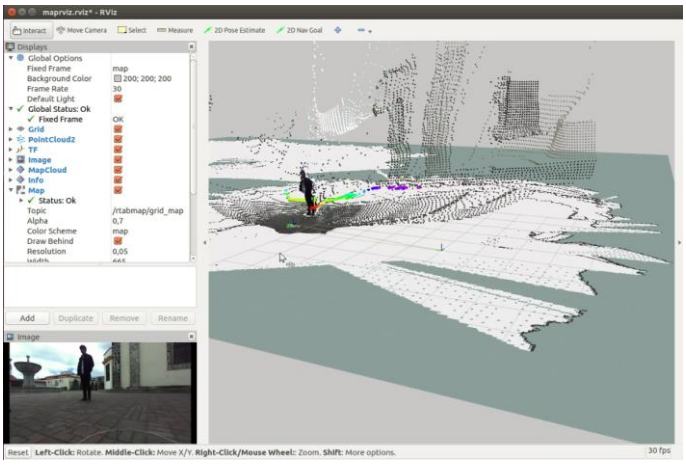


Figura 59. Laser 2D (obstáculo-persona)

Estas pruebas se las realizaron considerando 2 tipos de obstáculos: una caja (dimensiones: 0,5m x 0,6m x 2m) y una persona (mide 1,85 metros de altura y 0,45 metros de ancho). obteniendo las siguientes medidas: una detección máxima de obstáculos a una distancia de 8 m, una detección mínima de obstáculos a una distancia de 1,8 m, detección lateral máxima medida partir del eje central de la cámara a cada uno de los lados de 6 m. se obtiene un área de visión en forma de trapecio, donde se determina un área de detección útil. Cómo se observa en la figura 60.

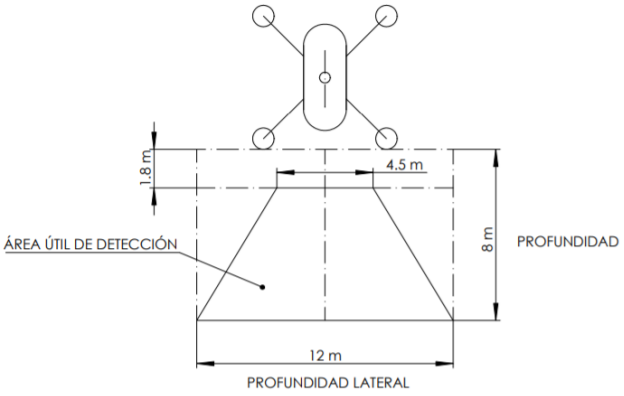


Figura 60. Área útil de detección

5.2 Evaluación del algoritmo de detección de obstáculos.

Para la evaluación del algoritmo de detección de obstáculos se realizaron 20 pruebas a 4 diferentes tramos de distancia, considerando como obstáculos 3 muestras: una caja, una persona, una persona en movimiento. Los datos obtenidos se resumen en las siguientes tablas, en donde se marca con un **+,** sí en Rviz se detecta como obstáculo, caso contrario se marca con un **-.** Se considera una detección cuando un tramo en la línea del scanner laser se encuentra de color rojo.

En la tabla 25 se resume los datos obtenidos con el primer obstáculo, es decir una persona la cual se ha ubicado a diferentes distancias (profundidades) de la cámara; que van desde los 2 hasta los 8 metros, que es la distancia optima de trabajo anteriormente encontrada.

Tabla 25.

Detección de obstáculos (persona)

N ° / Pruebas	**Rango de detección de obstáculos (persona), a profundidad (metros)**			
	< 2	2≤ x <4	4≤ x <6	6≤ x ≤8
1	-	+	+	+
2	+	+	-	+
3	+	+	+	+
4	+	+	+	+
5	-	-	+	-
6	+	+	+	+
7	+	+	-	-
8	+	+	+	-
9	+	+	+	+
10	+	+	+	+
11	-	+	+	+
12	+	+	+	+
13	+	+	+	-

CONTINÚA

14	+	+	+	+
15	+	+	-	+
16	+	+	+	+
17	-	+	+	+
18	+	+	+	-
19	+	+	+	+
20	+	+	+	-
Tot. (+)	16	19	17	14
Tot. (-)	4	1	3	6
Porct (%)	80	95	85	70

De la tabla anterior se tiene como resultados que: a la distancia de 2 metros el algoritmo tiene un 80% de efectividad de detectar este obstáculo, en el tramo desde los 2 metros hasta 4 metros una efectividad de 95%, en el tramo desde los 4 a 6 metros se obtuvo un porcentaje de 85% y en el último tramo de 6 a 8 metros un porcentaje de 70%. De estos se puede observar que existe una mayor efectividad de detección en el segundo tramo (de 3 a 5 m), y también existe un descenso en la detección en el rango mayor a los 6 metros debido a la deformación de la silueta. En la figura 61 se muestra una gráfica con las diferencias de la detección en las distancias antes mencionadas.

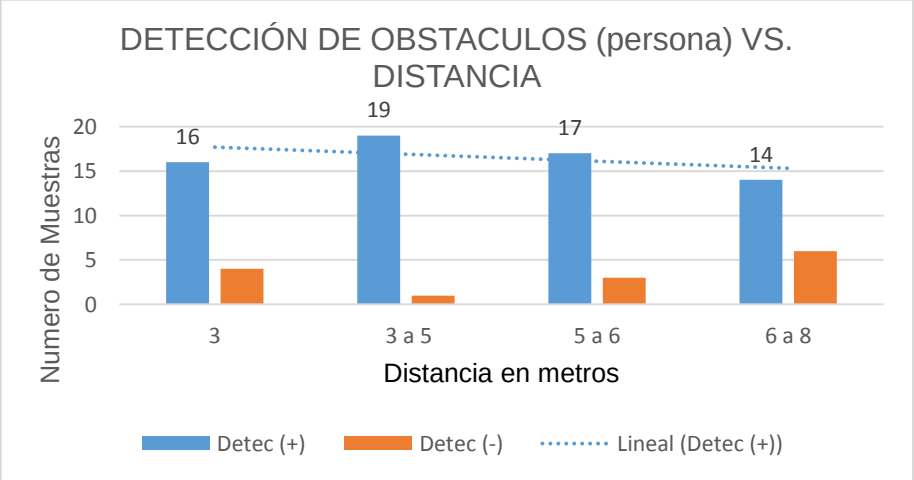


Figura 61 Detección de obstáculos (caja) Vs distancia

En la tabla 26 se resume los datos obtenidos con el segundo obstáculo, esta vez una caja la cual se ha ubicado a diferentes distancias (profundidad) de la cámara que van desde los 2 hasta los 8 metros.

Tabla 26.

Detección de obstáculos (caja) según la profundidad

N ° / Pruebas	Rango de detección de obstáculos (caja), a profundidad (metros)			
	< 2	2≤ x <4	4≤ x <6	6≤ x ≤8
1	+	+	+	+
2	+	+	-	+
3	+	+	+	+
4	+	+	+	+
5	+	+	+	-
6	+	+	+	+
7	+	+	-	-
8	+	+	+	-
9	+	+	+	+
10	+	+	+	+
11	+	+	+	+
12	+	+	+	+
13	+	+	+	-
14	+	-	+	+
15	+	+	-	+
16	+	+	+	-
17	-	+	+	+
18	+	+	-	+
19	-	+	+	+
20	+	+	+	+
Tot. (+)	18	19	16	15
Tot. (-)	2	1	4	5
Porct (%)	90	95	80	75

De la tabla anterior se tiene como resultados que: a la distancia de 2 metros el algoritmo tiene un 90% de efectividad de detectar este obstáculo, en el tramo desde los 2 metros hasta 4 metros una efectividad de 95%, en el tramo desde los 4 a 6 metros se obtuvo un porcentaje de 80% y en el último tramo de 6 a 8 metros un porcentaje de 75%. De estos se puede observar que existe una mayor efectividad de detección en el

segundo tramo (de 3 a 5 m), y también existe un descenso en la detección en el rango mayor a los 6 metros. En la figura 62 se muestra una gráfica con las diferencias de la detección en las distancias antes mencionadas.

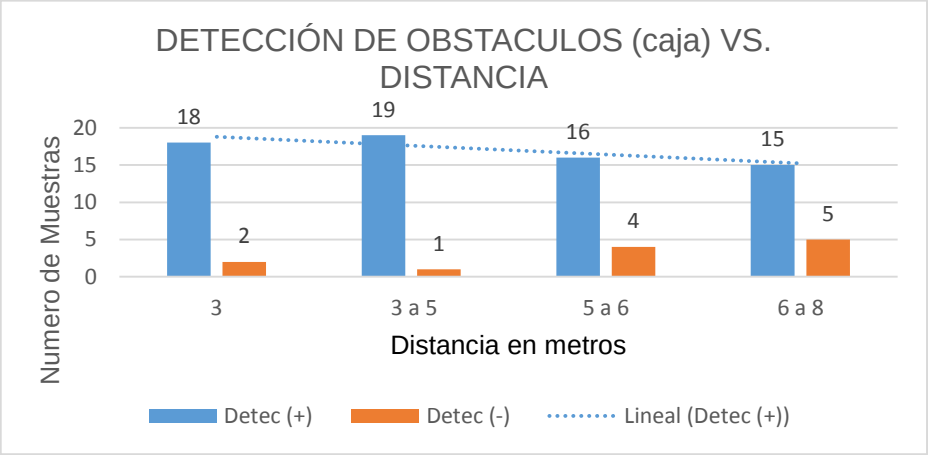


Figura 62 Detección de obstáculos (persona) Vs distancia

En la tabla 27 se resume los datos obtenidos con el tercer obstáculo, esta vez una persona que se encuentra caminando en línea recta hasta salir del campo de visión, simulando obstáculos móviles en un entorno. De manera similar a las 2 pruebas anteriores se ha ubicado a diferentes distancias (profundidad) de la cámara que van desde los 2 hasta los 8 metros.

Tabla 27.

Detección de obstáculos (persona en movimiento)

N ° / Pruebas	**Rango de detección de obstáculos (persona en movimiento), a profundidad (metros)**			
	< 2	2≤ x <4	4≤ x <6	6≤ x ≤8

CONTINÚA

1	+	+	+	+

2	+	+	-	+
3	+	+	+	+
4	+	+	+	+
5	+	+	+	-
6	+	+	+	+
7	+	+	-	-
8	+	+	+	-
9	+	+	+	+
10	+	+	+	+
11	+	+	+	+
12	+	+	+	+
13	+	+	+	-
14	+	-	+	+
15	+	+	-	+
16	+	+	+	-
17	-	+	+	+
18	+	+	-	+
19	-	+	+	+
20	+	+	+	+
Tot. (+)	16	17	14	11
Tot. (-)	4	3	6	9
Porct (%)	80	85	70	55

De la tabla anterior se tiene como resultados que: a la distancia de 2 metros el algoritmo tiene un 80% de efectividad de detectar este obstáculo, en el tramo desde los 2 metros hasta 4 metros una efectividad de 85%, en el tramo desde los 4 a 6 metros se obtuvo un porcentaje de 70% y en el último tramo de 6 a 8 metros un porcentaje de 55%. De estos se puede observar que existe una mayor efectividad de detección en el segundo tramo (de 3 a 5 m), y también existe un descenso en la detección en el rango mayor a los 6 metros. En la figura 63 se muestra una gráfica con las diferencias de la detección en las distancias antes mencionadas.

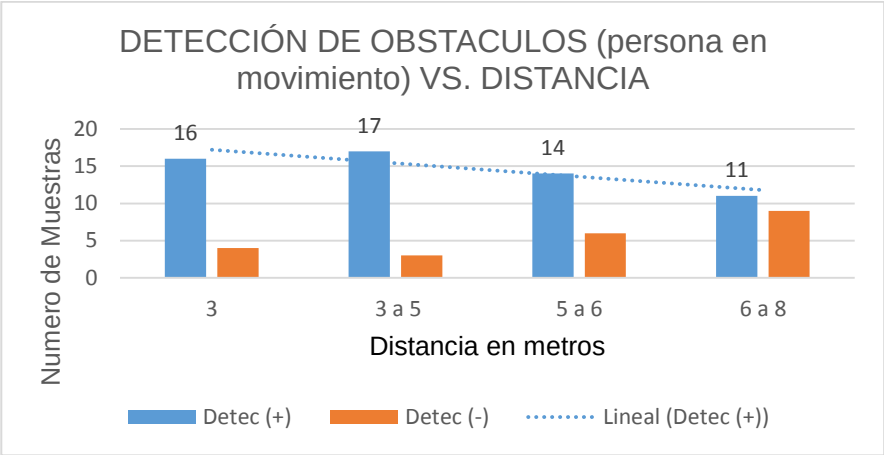


Figura 63 Detección de obstáculos (persona en movimiento) Vs distancia

De las 3 pruebas anteriores se puede concluir que el rango óptimo de trabajo para una correcta detección de obstáculos es desde los 3 hasta los 5 metros ya que en esta distancia se han logrado obtener el mayor número de detecciones de manera eficiente y correcta. Cabe recalcar que los datos fueron tomados a la altura de trabajo del quadro-tor la cual es de 1,5 metros.

5.3 Evaluación del algoritmo de detección de obstáculos dentro de la distancia óptima.

Una vez que se ha definido el rango óptimo de detección se procede a verificar la eficiencia del algoritmo dentro de este rango de distancia. De igual manera se realizaron 25 pruebas, cada una de ellas con los 3 obstáculos anteriormente definidos. En la tabla 28 se resumen los datos obtenidos, donde la última columna representa el porcentaje de efectividad del algoritmo en la detección de los 3 obstáculos.

Tabla 28.

Detección de obstáculos (dentro del rango optimo) según la profundidad

N ° / Pruebas	Rango de detección de obstáculos (persona en movimiento), a profundidad (metros) Obstáculo1 (1)	Obstáculo 2 (2)	Obstáculo 3 (mov)	Porcentaje (%)
1	+	+	+	**100**
2	+	+	-	**66,67**
3	+	+	+	**100**
4	+	+	+	**100**
5	+	+	-	**66,67**
6	+	+	+	**100**
7	+	+	+	**100**
8	+	+	+	**100**
9	+	+	+	**100**
10	+	+	+	**100**
11	+	+	+	**100**
12	+	+	-	**66,67**
13	+	+	+	**100**
14	+	+	+	**100**
15	+	+	+	**100**
16	+	+	+	**100**
17	+	+	+	**100**
18	+	-	-	**33,33**
19	+	+	+	**100**
20	+	+	+	**100**
21	+	+	+	**100**
22	+	+	-	**66,67**
23	+	+	+	**100**
24	+	+	-	**66,67**
25	+	+	+	**100**
Tot. (+)	25	24	19	**68**
Tot. (-)	0	1	6	**7**
Total				**90,67 %**

De la tabla anterior se puede observar que, dentro del rango de distancia de 3 a 5 metros, existe un 90.67 % de efectividad, es decir detección de obstáculos correctos, frente a un 9,33 % de detecciones fallidas, haciendo que este algoritmo sea eficiente. El 9,33 % se considera un error bajo, por lo que se puede determinar que el algoritmo de detección de obstáculos es adecuado dentro de la distancia delimitada y robusto ante los cambios de luz, ya que las pruebas se realizaron a diferentes horas del día sin afec-

tar su funcionamiento. En la figura 64 se puede observar la evaluación de la eficiencia del algoritmo de detección de obstáculos dentro del rango útil de trabajo.

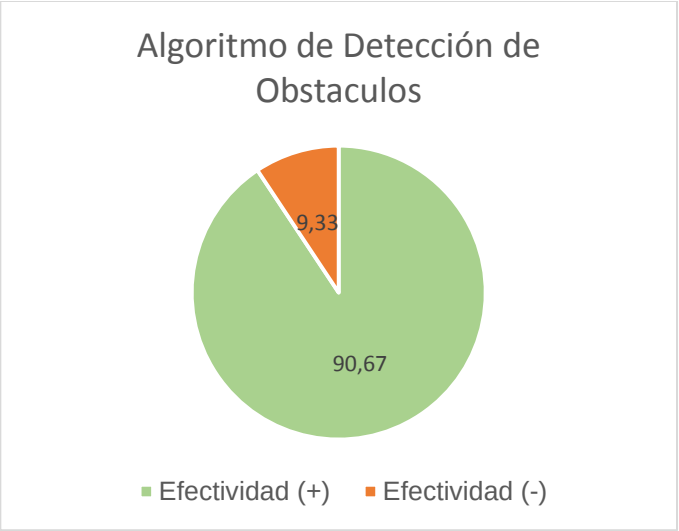


Figura 64. Evaluación de algoritmo / detección de obstáculos.

5.4 Comprobación de hipótesis de detección de obstáculos

- **Hipótesis nula**

Ho= No existe Detección de Obstáculos

- **Hipótesis alternativa**

Hi= Existe Detección de Obstáculos

Margen de error considerado = 5%

Margen de error de tabla = 0,05

$$GDL = (n° \ filas - 1) \ x \ (n° \ columnas - 1)$$

GDL = 2

Tabla 29.

Datos Para Calcular Chi-Cuadrado

	Obsta. 1	Obsta. 2	Obsta 3	Total
Positivo	25	24	19	**68**
Negativo	0	1	6	**7**
Total	**25**	**25**	**25**	**75**

$$\boldsymbol{n} = (n° \; filas) \; x \; (n° \; columnas) - 1 = 5$$

Tabla 30.

Cálculo de frecuencia teórica

n	f	ft	n	f	ft
0	**25**	22,66	3	**1**	2,66
1	**0**	2,66	4	**19**	22,66
2	**24**	22,66	5	**6**	2,66

Donde:

f: frecuencia calculada

ft: frecuencia teórica

$$\mathrm{X_{calc}}^2 = \sum_{i=0}^{n=9} \frac{(f - ft)^2}{ft}$$

$$\mathbf{X_{calc}}^{\mathbf{2}} = \mathbf{8,82}$$

Después de calcular el valor de Chi-cuadrado se compara este con el de la tabla, considerando un error de 0,05. Como se muestra en la figura 65

v/p	0,001	0,0025	0,005	0,01	0,025	0,05	0,1
1	10,8274	9,1404	7,8794	6,6349	5,0239	3,8415	2,7055
2	13,8150	11,9827	10,5965	9,2104	7,3778	5,9915	4,6052
3	16,2660	14,3202	12,8381	11,3449	9,3484	7,8147	6,2514
4	18,4662	16,4238	14,8602	13,2767	11,1433	9,4877	7,7794
5	20,5147	18,3854	16,7496	15,0863	12,8325	11,0705	9,2363
6	22,4575	20,2491	18,5475	16,8119	14,4494	12,5916	10,6446
7	24,3213	22,0402	20,2777	18,4753	16,0128	14,0671	12,0170
8	26,1239	23,7742	21,9549	20,0902	17,5345	15,5073	13,3616
9	27,8767	25,4625	23,5893	21,6660	19,0228	16,9190	14,6837
10	29,5879	27,1119	25,1881	23,2093	20,4832	18,3070	15,9872

Figura 65. *Distribución Chi-cuadrado*

$$\text{Si } X_{calc}{}^2 > X_{tabla}{}^2 = \text{Hi es valida}$$

$$Si\ X_{calc}{}^2 < X_{tabla}{}^2 = \text{Ho es valida}$$

Como:

$$\mathbf{8,82 > 5,9915 = Hi\ es\ valida}$$

5.5 Evaluación del algoritmo de evasión de obstáculos y seguimiento de ruta

Para la evaluación del algoritmo de evasión de obstáculos y seguimiento de ruta se realizan 36 pruebas a 4 diferentes rutas establecidas, considerando como obstáculos 3 muestras: una caja, una persona, una persona en movimiento. Los datos obtenidos se resumen en las siguientes tablas, en donde se marca con un **+,** sí en Rviz se realiza la evasión del obstáculo y el seguimiento de la ruta, caso contrario se marca con un **-.** Se considera una correcta evasión y el seguimiento de la ruta, cuando el quadrotor vuela siguiendo la línea verde trazada (ruta) en el visualizador Rviz con una tolerancia máxima de 0,5 metros y aterriza en el lugar indicado con la orientación correcta.

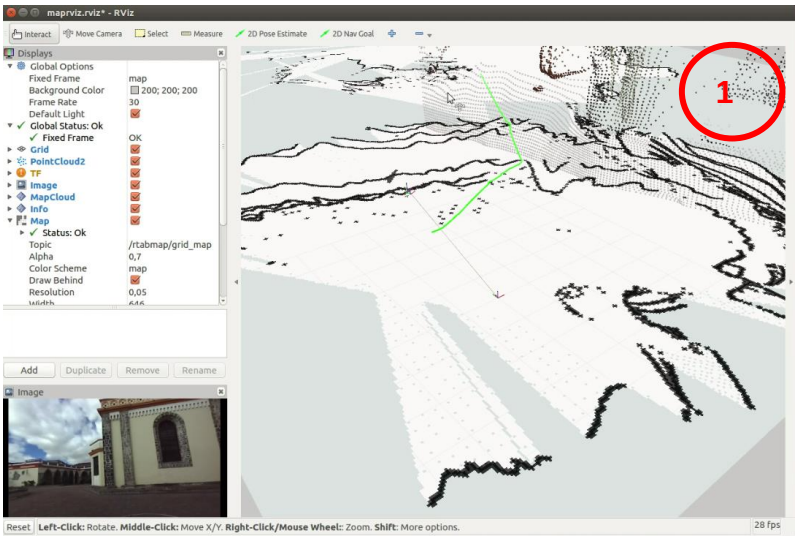


Figura 66. Trayectoria generada N° 1

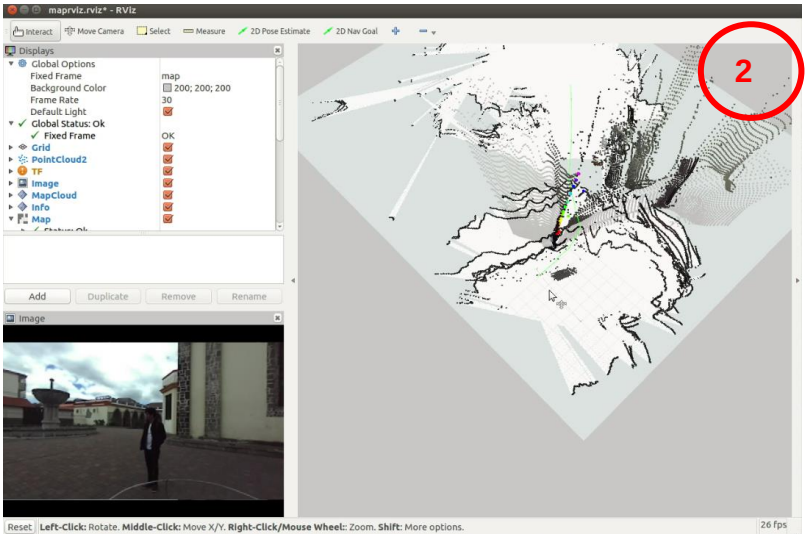


Figura 67. Trayectoria generada N° 2

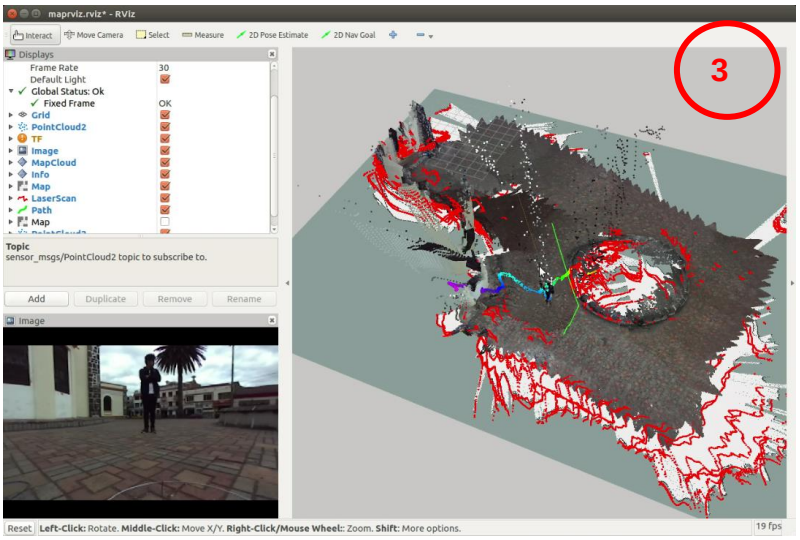


Figura 68. Trayectoria generada N° 3

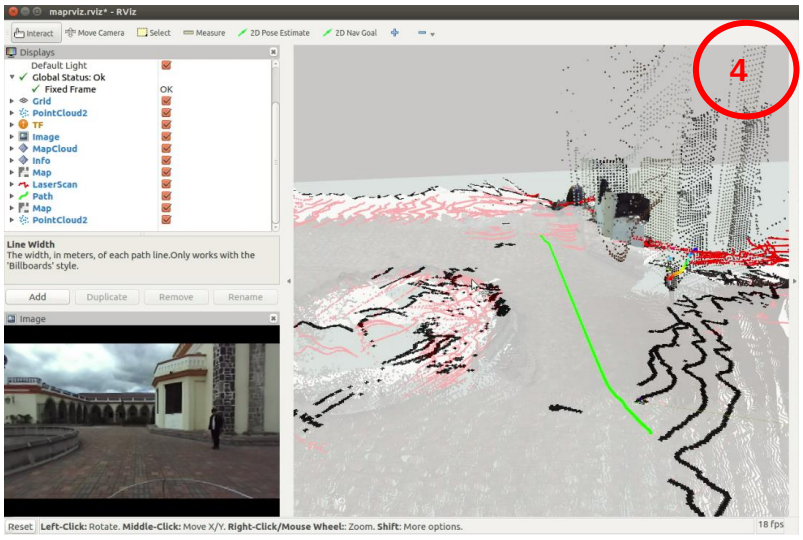


Figura 69. Trayectoria generada N° 4

En la tabla 31 se resume los datos obtenidos con el primer obstáculo, es decir una persona la cual se ha ubicado en diferentes lugares al azar del patio de la ESPE-L, interrumpiendo la posible trayectoria de vuelo del quadrotor en la misión de vuelo que se le ha asignado.

Tabla 31.

Evasión de obstáculos (persona)

N ° / Pruebas	Evasión de obstáculos (persona) y seguimiento de Ruta				
	Trayect. 1	Trayect. 2	Trayect. 3	Trayect. 4	Porcentaje (%)
1	+	+	+	+	**100**
2	+	+	+	+	**100**
3	+	+	+	+	**100**
4	+	+	+	+	**100**
5	+	+	+	+	**100**
6	+	+	+	+	**100**
7	+	+	+	+	**100**
8	+	+	+	-	**75**
9	+	+	+	+	**100**
10	+	+	-	+	**75**
11	+	+	+	+	**100**
12	+	+	+	+	**100**
Tot. (+)	12	12	11	11	**46**
Tot. (-)	0	0	1	1	**2**
T Porct (%)	**100**	**100**	**91,66**	**91,66**	**95,83**

En la tabla 32 se resume los datos obtenidos con el segundo obstáculo, es decir una caja la cual se ha ubicado en diferentes lugares al azar del patio de la ESPE-L, interrumpiendo la posible trayectoria de vuelo del quadrotor en la misión de vuelo que se le ha asignado.

Tabla 32.

Evasión de obstáculos (caja)

N ° / Pruebas	Evasión de obstáculos (Caja) y seguimiento de Ruta				
	Trayect. 1	**Trayect. 2**	**Trayect. 3**	**Trayect. 4**	**Porcentaje (%)**
1	+	+	+	+	**100**
2	+	+	+	+	**100**
3	+	+	-	+	**75**
4	+	+	+	+	**100**
5	+	+	+	+	**100**
6	+	+	+	+	**100**

CONTINÚA

7	+	+	+	+	**100**
8	+	+	+	+	**100**
9	+	+	-	-	**50**
10	+	+	+	+	**100**
11	+	+	+	+	**100**
12	+	+	+	+	**100**
Tot. (+)	12	12	10	11	49
Tot. (-)	0	0	2	1	4
T. Porcen. (%)	**100**	**100**	**83,33**	**91,66**	**93,75**

En la tabla 33 se resume los datos obtenidos con el tercer obstáculo, es decir una persona que va a pasar caminando (interrumpiendo) a una velocidad normal por el frente de la ruta ya trazada, se ha realizado este proceso en diferentes lugares al azar del patio de la ESPE-L. Se consideró una evasión exitosa cuando al pasar la persona y el quadrotor se encuentra a una distancia máxima de 2 metros se recalcula una nueva ruta. Cabe recalcar que los datos fueron tomados a una altura de vuelo del quadrotor de 1,5 metros.

Tabla 33.

Evasión de obstáculos (persona en movimiento)

N ° / Pruebas	**Evasión de obstáculos (Persona en movimiento) y seguimiento de Ruta**				
	Trayect. 1	Trayect. 2	Trayect. 3	Trayect. 4	Porcentaje (%)
1	+	+	+	+	**100**
2	+	+	+	+	**100**
3	+	+	+	-	**75**
4	+	+	-	+	**75**
5	+	+	+	+	**100**
6	+	+	-	+	**75**
7	+	+	+	+	**100**
8	+	+	+	-	**75**

CONTINÚA

9	+	+	+	+	**100**
10	+	+	+	+	**100**
11	+	-	+	+	**75**
12	+	+	-	+	**75**
Tot. (+)	12	11	9	10	55
Tot. (-)	0	1	3	2	9
Porct (%)	**100**	**91,66**	**75**	**83,33**	**87,5**

De los resultados anteriores se puede analizar que en las pruebas hechas con los obstáculos estáticos se tiene una mayor efectividad en el algoritmo, frente a una notable baja en la efectividad del mismo cuando se presentan obstáculos dinámicos. Esto es debido a la baja capacidad de computo que se cuenta, ya que la velocidad máxima de muestreo soportada es de 2 Hz.

Realizando un total de 36 pruebas (entre los tres diferentes obstáculos establecidos) para la comprobación del algoritmo de evasión de obstáculos y seguimiento de la ruta planificada se determinó una efectividad total del 92,36 %, frente a un 7,64%. Teniendo en cuenta que las condiciones medioambientales como el viento, afectan el vuelo normal del quadrotor, y por ende el seguimiento de la ruta generada. Por lo que se puede determinar que el algoritmo de evasión y seguimiento de ruta es eficiente. En la figura 70 se puede observar la evaluación de la eficiencia del algoritmo de detección de obstáculos dentro del rango útil de trabajo.

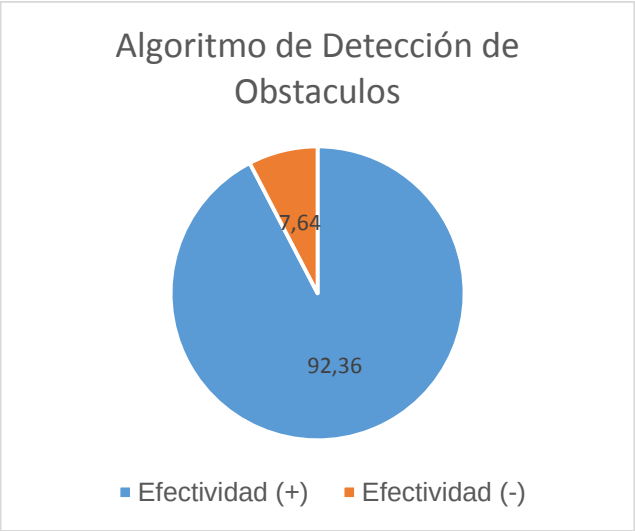


Figura 70. Evaluación de algoritmo de evasión de obstáculos y seguimiento de ruta.

5.6 Comprobación de hipótesis de evasión y seguimiento de Ruta

- **Hipótesis nula**

Ho= No existe evasión y seguimiento de la ruta generada

- **Hipótesis alternativa**

Hi= Existe evasión y seguimiento de la ruta generada

Margen de error considerado = 5%

Margen de error de tabla = 0,05

$$GDL = (n° \ filas - 1)\ x\ (n° \ columnas - 1)$$

GDL = 3

Tabla 34.

Datos Para Calcular Chi-Cuadrado

	Trayect 1	Trayect 2	Trayect 3	Trayect 4	Total
Positivo	36	35	30	32	**133**
Negativo	0	1	6	4	**11**
Total	**36**	**36**	**36**	**36**	**144**

$$\boldsymbol{n} = (n° \, filas) \; x \; (n° \, columnas) - 1 = 7$$

Tabla 35.

Cálculo de frecuencia teórica

n	f	ft	n	f	ft
0	**36**	33,25	4	**30**	33,25
1	**0**	2,75	5	**6**	2,75
2	**35**	33,25	6	**32**	33,25
3	**1**	2,75	7	**4**	2,75

Donde:

f: frecuencia calculada

ft: frecuencia teórica

$$\mathrm{X_{calc}}^2 = \sum_{i=0}^{n=9} \frac{(\mathrm{f} - \mathrm{ft})^2}{\mathrm{ft}}$$

$$\mathbf{X_{calc}}^{\mathbf{2}} = \mathbf{8,974}$$

Después de calcular el valor de Chi-cuadrado se compara este con el de la tabla, considerando un error de 0,05. Como se muestra en la figura 71

v/p	0,001	0,0025	0,005	0,01	0,025	0,05	0,1
1	10,8274	9,1404	7,8794	6,6349	5,0239	3,8415	2,7055
2	13,8150	11,9827	10,5965	9,2104	7,3778	5,9915	4,6052
3	16,2660	14,3202	12,8381	11,3449	9,3484	7,8147	6,2514
4	18,4662	16,4238	14,8602	13,2767	11,1433	9,4877	7,7794
5	20,5147	18,3854	16,7496	15,0863	12,8325	11,0705	9,2363
6	22,4575	20,2491	18,5475	16,8119	14,4494	12,5916	10,6446
7	24,3213	22,0402	20,2777	18,4753	16,0128	14,0671	12,0170
8	26,1239	23,7742	21,9549	20,0902	17,5345	15,5073	13,3616
9	27,8767	25,4625	23,5893	21,6660	19,0228	16,9190	14,6837
10	29,5879	27,1119	25,1881	23,2093	20,4832	18,3070	15,9872

Figura 71. Distribución Chi-cuadrado

$$\text{Si } X_{calc}{}^2 > X_{tabla}{}^2 = \text{Hi es valida}$$

$$\text{Si } X_{calc}{}^2 < X_{tabla}{}^2 = \text{Ho es valida}$$

Como:

$$\mathbf{8,974 > 7,8147 = Hi\ es\ valida}$$

Anteriormente se verificó la funcionalidad y fiabilidad de los algoritmos de detección y evasión de obstáculos por separado efectuando varios planes de vuelo por las diferentes rutas generadas en la plazoleta de la universidad de las fuerzas armadas ESPE-L, obteniendo resultados favorables. Sin embargo, es necesario acoplar estos algoritmos para formar el sistema de piloto inteligente el cual permita realizar un vuelo autónomo por la ruta establecida, permitiendo que el quadrotor pueda adquirir las fotos aéreas para la construcción de un Ortomosaico.

Dado que la aplicabilidad del proyecto es el monitoreo o vigilancia de áreas de difícil acceso mediante un sistema de piloto inteligente, las pruebas para la generación de un Ortomosaico se realizaron en el primer mapa en exteriores generado (parque del colegio 9 de octubre- Machala) el cual asemeja más a "*una zona de difícil acceso*". En esta prueba se sobrevoló esta zona a una altura de 40m y se tomaron 12 fotos, de las cuales se construyó el Ortomosaico que se muestra en la figura 72.

Figura 72. Ortomosaico

CAPITULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1 Conclusiones

- El uso del controlador PIXFALCON en el quadrotor fue de gran ventaja ya que, no solo se puede transmitir los parámetros de vuelo mediante protocolos MAVLink, sino que también, permite la inclusión de algoritmos complejos controlados por un sistema operativo que englobe todas las funciones.
- El software Mission Planner permite configurar y calibrar los sensores del quadrotor como: el giroscopio, GPS y su unidad de medición inercial. Además, facilita cargar la última versión de firmware "*ArduCopter*" disponible para el módulo PIXFALCON.
- Las herramientas SDK de la cámara estereoscópica ZED permiten variar la resolución de trabajo de esta, disminuyendo la cantidad de información que será procesada por la GPU del PC, ayudando a un muestreo más apropiado.
- El procesamiento de imágenes obtenidos de la cámara ZED, consume una gran cantidad de recursos computacionales. Esto se ve reflejado en la velocidad de comunicación entre los diferentes nodos de ros, también limita la velocidad de muestreo para el algoritmo de evasión de obstáculos lo que representa una dificultad al tratarse de obstáculos dinámicos.
- El uso de ROS facilita el desarrollo de algoritmos que permitan realizar las diferentes tareas que comprenden un sistema de piloto inteligente, como el:

control de vuelo del quadrotor, la detección y evasión de obstáculos, usado al momento de generar un vuelo autónomo.

- El sistema operativo robótico ROS, presenta la ventaja de contar con ciertas herramientas que facilitan la localización y mapeado usando algoritmos para SLAM en 2D y 3D; procesos necesarios para la navegación autónoma y evasión de obstáculos utilizando cualquier plataforma robótica.
- ROS al ser un sistema operativo de código abierto, posee un repositorio oficial con varios trabajos desarrollados y probados en sistemas robóticos complejos por la gran comunidad de desarrolladores que aportan a la investigación de nuevos avances en la robótica.
- En este Proyecto se ha logrado comprender algunas de las funcionalidades básicas de ROS, además se ha empleado la librería rospy y roscpp que permiten crear nodos en lenguaje Python y C++, los cuales pueden interactuar entre sí con facilidad.
- El quadrotor presenta problemas de “armado” cuando se encuentra rodeado de edificios ya que no recibe una correcta señal de GPS. Esto causa un fallo en el chequeo previo del estado del UAV antes de que este puede encender sus motores. Por lo que fue necesario desactivar, mediante el software Mission Planner, todas las defensas configuradas de fábrica para poder realizar las diferentes pruebas.
- El desempeño del algoritmo de evasión de obstáculos, el cual se encarga de generar la ruta que debe seguir el quadrotor, depende principalmente de la ca-

lidad de imagen que proporciona la cámara ZED; por lo que se puede concluir que si se trabaja con la resolución más baja que ofrece la cámara y a una frecuencia de trabajo superior a los 15 Hz se puede generar una ruta correcta con tiempo para que el quadrotor pueda responder.

- Para tener un control de vuelo adecuado del quadrotor, sin poner en peligro la aeronave ni la seguridad de los operarios, se deben cumplir algunas condiciones de vuelo necesarias como: baja velocidad de viento relativo y lugares abiertos, ya que debido a su geometría el modelo Xplus One el viento choca en sus paredes y dificulta la estabilidad del mismo cuando se encuentra estático.
- Es necesaria un porcentaje adecuado de traslape entre las imágenes que se adquieren (por parte de la cámara estereoscópica y la cámara de base) para poder generar un mapa 3D y un orto-mosaico útil para el sistema de navegación.
- ROS permite que el sistema de piloto inteligente sea modular, permitiéndole trabajar en cualquier plataforma que se controle a través del protocolo de comunicación MAVLink. Por lo que no está limitado a vehículos aéreos, dándole la capacidad de operar robots terrestres y acuáticos con sistemas de control similares.

6.2 Recomendaciones

- Debido a que el sistema de piloto inteligente requiere de varios programas corriendo al mismo tiempo en la PC que actúa como estación de control en tie-

rra, se recomienda que esta tenga gran capacidad en su memoria RAM y posea al menos una tarjeta de procesamiento grafico NVIDIA de la serie 1050 o superior, para poder responder de manera fluida a los altos requerimientos de los comandos openCV para visión artificial al mismo tiempo que mantiene la estabilidad del UAV.

- Es recomendable que el UAV posea una segunda computadora dedicada a la adquisición de datos en vuelo y la transmisión de los mismos a la estación de control en tierra. Esto ayudaría a alivianar la carga de datos por procesar, mejorando la respuesta del piloto inteligente al aumentar su velocidad del muestreo.
- Tomando en cuenta que se necesita transmitir datos de distintos tipos desde el UAV a la estación de control, es recomendable que se utilice un medio de comunicación diferente al de telemetría, el cual puede ser reemplazado por una red LAN wifi para mejorar el desempeño de los elementos del sistema.

REFERENCIAS BIBLIOGRÁFICAS

3DRobotic. (2018). *Pixhawk*. Recuperado el 3 de mayo de 2018, de www.pixhawk.org/start

Alicante, U. d. (2015). Sistema de control que permite el vuelo autónomo de drones. *San vicente de Raspeig*.

Angulo V., L. V. (2014). Generación de cartografía básica a detalle mediante una metodología de toma con aviones no tripulados (UAV´s). Ecuador, Quito: Universidad de las Fuerzas Aramadas ESPE.

Blasco , X., García N., S., & Reynoso M., G. (2012). Control autónomo del seguimiento de trayectorias de un vehículo quadrotor. España, Valencia: Universidad Politécnica de Valencia.

Casanova , A. (2015). Control difuso del quadrotor AR.Drone 2.0 para el seguimiento autónomo de trayectorias. México, D.F. Oxaca: Universidad Tecnológica De La Mixteca.

Chicaiza C., F., & Chuchico A., C. (2015). Implementación de un sistema de piloto automático basado en una plataforma fpga para la navegación autónoma del vehículo aérero no tripulado de la Universidad de las Fuerzas Armadas. Ecuador, Latacunga: Universidad de las Fuerzas Armadas ESPE-L.

Coello R., A., & Ballesteros A., G. (2014). Fotogrametría de UAV de ala fija y comparación con topografía clásica. España, Madrid: Universidad Católica de Madrid.

Corke, P. (2018). *Robotics*. Recuperado el 12 de Mayo de 2018, de http://petercorke.com/wordpress/toolboxes/robotics-toolbox

DJI. (2018). *DJI Store*. Recuperado el 7 de junio de 2018, de https://store.dji.com/product/phantom-4-pro-v2?vid=43151

Fernández C., J. A. (2012). Planteamiento del desarrollo del sistema aviónica de un UAV por medio del microcontrolador 18F2620. México, D.F: Instituto Politécnico Nacional.

Fernández G., F. (2015). Diseño e Implementación De Un Sistema De Control. Chile, Santiago de Chile: Universidad de Chile.

García M., L. (2017). Navegación sin mapa y mapeado en robótica móvil para entornos no estructurados. España, Sevilla: Universidad de Sevilla.

Gil Aparicio, A. (2008). Construcción cooperativa de mapas visuales mediante un equipo de robots móviles. España, Elche: Universidad Miguel Hernández.

Jimenez L., M. (2017). Montaje y configuracion del drone comercial ERLE-COPTER. España, Sevilla: Universidad de Sevilla.

López de Paz, G. (2014). Diseño de un programa de ortorectificación y georeferenciación de imágenes aéreas aplicadas a campos de caña de azúcar. Peru, Lima: Pontificia Universidad Católica del Perú.

López Torres, P. (2016). Análisis de algoritmos para localización y mapeado simultáneo de objetos. España, Sevilla: Universidad deSevilla.

Mayorga R, R. (2009). Sistema de Navegación para vehículos aéreos cuadricópteros. España, Cataluña: Universidad Politécnica de Cataluña.

Ministerio del Interior, E. (2014). *Gobierno de la República Del Ecuador*. Recuperado el 15 de Enero de 2018, de https://www.ministeriointerior.gob.ec/2014-gir-realizo-24-operaciones-de-rescate-en-montana-y-quebradas/

Muñoz, M. Á. (2018). *Manual de Vuelo*. Recuperado el 12 de Julio de 2018, de http://www.manualvuelo.com/PBV/PBV12.html

Nvidia. (2018). *CUDA toolkit 9.2*. Recuperado el 16 de Mayo de 2018, de https://developer.nvidia.com/cuda-downloads

Olea Hernández, J., & Sánchez L., A. (2014). diseño y construcción de una aeronave de ala rotativa para operaciones de seguridad fronteriza y respuesta a emergencia. México, D.F.: Instituto Politécnico Nacional.

OpenCV. (2018). *OpenCV 3.1.0*. Recuperado el 27 de Marzo de 2018, de www.docs.opencv.org/3.1.0/d1/dfb/intro.html

Paucar M., C., & Morales C. , M. (2017). Investigación e imlementación de un sistema de seguridad fijo y móvil ,mediante un dron, usando visión artificial para detección y seguimiento de personas en un ambiente externo específico. Ecuador, Latacunga: Universidad de las Fuerzas Armadas ESPE-L.

Pelaez V., A. (2016). Análisis del filtro extendido Kalman presente en el pixhawk profundizando en cuanto a su mecanismo de ajuste. Cuba, Santa Clara: Universidad central "Marta-Abreu-2 de las Villas".

Peréz C. , R. (2014). Mapas mediante imágenes tomadas por un UAV. España, Barcelona: Universidad Autónoma de Barcelona.

Pixfalcon. (2018). *Holybro*. Recuperado el 15 de abril de 2018, de http://www.holybro.com/manual/pixfalcon.pdf

Plaza Rey, D. (2017). Navegación autónoma de drones y Automatización de rutas Aplicadas a la Limpieza de Edificios. España, Cataluña: Universidad Politécnica de Cataluña.

Rico, R., Maisterra, P., & Martínez, M. (2015). Identificación de los parámetros de un quadrotor. España, Bilbao: Universidad dela Rioja.

Rivas R., A. (2017). Navegación en interiores con robots de servicios. España, Sevilla: Universidad de Sevilla.

ROS.org. (2018). *CvBridge Tutorials*. Recuperado el 22 de Abril de 2018, de www.wiki.ros.org/cv_bridge/Tutorials/UsingCvBridgeToConvertBetweenROsImagesAndOpenCVImages.com

Sevilla F., L. (2014). Modelado y control de un cuadricoptero. España, Madrid: Universidad Pontificia Comillas.

StereoLabs. (2018). *Documentation*. Recuperado el 8 de Marzo de 2018, de www.stereolabs.com/documentation/overview/getting-started/introduction.html

Toriz, A., Raygoza, M., & Martínez, D. (2017). Modelo de inclusión tecnológica UAV para la prevención de trabajos de alto riesgo, en industrias de la construcción basado en la metodología IVAS. *Revista Iberoamericana de Auomática e informática Industrial RIAI*.

Tumbaco M., D., & Quimbita Z., W. (2014). Diseño y construcción de un prototipo de robot delta con implementación de un cortador laser cnc utilizando la plataforma robotica operating system (ros) para la elaboración de artículos publicitarios. Ecuador, Latacunga: Universidad de las Fuerzas Armadas ESPE-L.

Ulrich T., K. (2013). *Diseño y Desarrollo de productos.* México, D.F.: Mc Graw Hill.

Valencia A. , L. (2014). Generación de cartografía básica a detalle mediante una metodología de toma con aviones no tripulados UAV´s. Ecuador, Quito: Universidad de las Fuerzas Armadas ESPE.

Vasquez L., K. (2014). generación de mosaicos georreferenciados a partir de imágenes capturadas con vehículos aéreos no tripulados. Colombia, Bogota D.C.: Pontificia Universidad Javieriana.

Xcraft. (2018). *X plusOne*. Recuperado el 15 de Julio de 2018, de https://xcraft.io/collections/x-plusone/products/x-plusone-extra-battery-pack

ANEXOS

DEPARTAMENTO DE CIENCIAS DE LA ENERGÍA Y MECÁNICA

CARRERA DE INGENIERÍA EN MECATRÓNICA

CERTIFICACIÓN

Se certifica que el siguiente trabajo fue desarrollado por los señores: **ERICK ALEXANDER NOBOA CASTRO y CRISTHIAN JESÚS CAMPOS CAMPOS**, bajo nuestra supervisión.

Ing. Andrea Córdova
TUTOR DEL PROYECTO

Aprobado por:

Ing. Vicente Hallo
DIRECTOR DE DEPARTAMENTO

Dr. Rodrigo Vaca
SECRETARIO ACADÉMICO