INSTITUTO TECNOLÓGICO DE TUXTLA GUTIÉRREZ

LABORATORIOS DE INGENIERÍA ELECTRÓNICA Y MECATRÓNICA DEL ITTG,

DIVISIÓN DE ESTUDIOS DE POSGRADOS E INVESTIGACIÓN

REPORTE DE RESIDENCIA PROFESIONAL

PROYECTO:

CONTROL DE UN VEHÍCULO AÉREO NO TRIPULADO PARA APLICACIÓN DE MONITOREO AMBIENTAL

PRESENTA:

GÓMEZ RODRÍGUEZ PAULO CÉSAR

ASESOR INTERNO:

Dr. FRANCISCO RONAY LÓPEZ ESTRADA

TUXTLA GUTIÉRREZ, CHIAPAS, JUNIO DEL 2015.

CONTROL DE UN VEHÍCULO AÉREO NO TRIPULADO PARA APLICACIÓN DE MONITOREO AMBIENTAL

Índice general

Capítulo 1

Introducción

En el presente proyecto se desarrolla el control de un vehículo aéreo no tripulado (VANT), para esto se aplica una de las estrategias de control de tipo PID con el cual se realiza el control autónomo en un espacio tridimensional, haciendo uso de un software específico para sitemas robóticos, Robot Operating System (ROS), donde las ventajas de usar este software es por sus paquetes específicos que contiene para el control y simulación de vehículos no tripulados, lo cual facilitará el trabajo en este sentido.

Los VANT durante los ultimos años han recibido considerable atención por parte de investigadores, profesores y estudiantes como un gran avance tecnológico, los ultimos avances en la tecnología han impulsado el desarrollo de diferentes estrategias de control a si como la operación de este tipo de vehículos, una parte importante es el modelo dinámico de un VANT que es el punto de partida para todos los estudios basados en la navegación aérea, también haciendo uso de muchos metodos de control en el cual son implementados en los VANT para un control autónomo, algunos de estos metodos es el controlador PID, control no lineal y controladores LQR entre otros, métodos que requieren información precisa para su aplicación, a si como igual se necesita información de las mediciones de posición y altitud que se pueden obtener haciendo uso de diferentes aparatos de medición como un giroscopio, un acelerómetro, magnetómetro y otros aparatos, como el GPS, sensores ultrasonicos y sensores de presión barometrica.

El propósito de este proyecto es presentar el control autónomo de trayectorias y proporcionar conceptos importantes sobre la navegación autónoma y el control de vehículos no tripulados, para formar una base de investigación innovadora dentro del Instituto Tecnológico de Tuxtla Gutiérrez. Esté se persigue con dos objetivos principales. El primer objetivo es estudiar el modelo matemático de la dinámica de los VANT, el segundo objetivo es el desarrollo de métodos adecuados para la estabilización y control de trayectoria.

En el proyecto se abordará el modelado básico para la navegación autónoma, se hace uso de los ángulos de navegación son un tipo de ángulos de Euler usados para describir la orientación de un objeto en tres dimensiones, si se tiene un sistema de coordenadas móvil respecto de uno fijo, en tres dimensiones, y se desea dar la posición del sistema móvil en un momento dado, hay varias posibilidades de hacerlo, una de ellas son los ángulos de navegación, llamados en matemáticas ángulos de Tait-Bryan, son tres coordenadas angulares que definen un triedro rotado desde otro que se considera el sistema de referencia se definen matemáticamente de forma similar a los ángulos de Euler, pero en vez de usar como línea de nodos el corte entre dos planos homólogos (por ejemplo el xy es el homólogo del xy), se utilizan dos planos no homólogos (por ejemplo xy e yz).

Para la simulación y manipulación de nuestro VANT, así como su control en un entorno tridimensional como se menciono anteriormente se ara uso de ROS ya que este sistema operativo de código abierto mantenido por la Open Source Robotics Foundation (OSRF), proporciona los servicios que caben esperar de un sistema operativo incluyendo las abstracciones de hardware, control de dispositivos a bajo nivel, implementación de utilizades comunes, paso de mensajes entre procesos y gestión de paquetes, tambien proporciona herramientas y librerías para obtener, compilar, escribir y ejecutar código a través de multiples ordenadores, ROS se compone de un número de nodos independientes, cada nodo se comunica con el resto de nodos utilizando el modelo publicador/subscriptor.

ROS es definitivo para el desarrollo de aplicaciones de robots proporciona una arquitectura distribuida

y contiene algoritmos implementados del estado del arte, mantenidos por expertos en el campo todo con una licencia permisiva que en unos pocos años cambiará el panorama de la robótica y es ampliamente adoptado por la investigación, centros educativos y la industria.

Otra de las ventajas que el uso de ROS supone para este proyecto es que contiene paquetes específicos para el control y manipulación de vehículos aéreos no tripulados, lo cual facilitará el trabajo en este sentido. ROS está liberada bajo los terminos de la licencia BSD (Berkeley Software Distribution) y un software open source.

ROS promueve la reutilización de código así los desarrolladores y científicos lograran autonomía sobre los VANT y no tienen que reinventar la rueda todo el tiempo, con ROS podemos hacer esto y mucho más, se puede coger el código de los repositorios, mejorarlo y compartirlo de nuevo es como de esta manera se realizara dicho proyecto y poder generar intereses dentro del campo de la investigación.

1.1. Objetivo general

Realizar una estrategia de control de tipo proporcional, integral y derivativo (PID), para poder realizar el control autónomo del vehículo aéreo en un espacio tridimencional aplicando un software para sistemas robóticos en especifico llamado ROS que nos permitira comunicarnos con el VANT mediante un protocolo de comunicación Wifi.

1.2. Justificación

Se aprovechara una importante característica de nuestro drone que es la forma de comunicación, la cual hace posible una integración con otros sistemas operativos en particular una computadora en la cual sera nuestro centro de mando y a la vez recibir los datos de navegación en tiempo real a través de la misma, todo esto se logrará por medio de una conexión Wifi. La idea principal es la de dar compatibilidad a través de otro sistema operativo. En este caso, elegimos Linux en concreto Ubuntu, ya que nos permite la compatibilidad mediante el uso de un software específico para sistemas robóticos, en concreto es el meta-sistema operativo ROS (Robot Operating System), formado por un conjunto de programas, herramientas y librerías de programación, todos bajo la licencia de software libre y gratuito todo esto nos llevara a obtener nuestro control adecuado para nuestro VANT.

1.3. Hardware: Ar.Drone 2.0

Teniendo en cuenta que nuestro vehículo está conformado por diferentes sensores, microprocesadores y sistemas de propulsión que son más pequeños, ligeros y mas capaces que nunca, llevando a niveles de resistencia, eficiencia y autonomía que sobrepasan las capacidades humanas el cual son de mucha importancia para la navegación y estabilización de los VANT, cabe decir que es un hardware muy completo que genera gran cantidad de aplicaciones ante la sociedad y una de las caracteristicas más importantes es la parte de como recibir los datos de navegación por medio de un protocolo de comunicación conocido como Wifi, la cual nos permite la comunicación entre el vehículo aéreo a una computadora que es el punto de control para nuestro vehículo.

Ademas otras de las ventajas, respecto a otros modelos de quadrotores, esque se distribuye en cualquier comercio especializado o gran superficie, haciendo fácil el acceso a esté tipo de vehículo aéreo como se muestra en la Fig 1.1

Figura 1.1: Ar.Drone 2.0

En los ultimos años ha habido un gran interés en el desarrollo de vehículos aéreos no tripulados, ya que poseen características únicas como: tamaño pequeño, gran maniobrabilidad y relativo bajo precio, haciéndolos atractivos para uso tanto militar como civil en aréas como vigilancia, reconocimiento e inspección en ambientes complejos o peligros.

Capítulo 2

Estado del arte

2.1. Vehículo aéreo no tripulado, UAV.

Los drones o UAV tienen un gran potencial en áreas muy diversas, ya que puede desplazarse rápidamente sobre un terreno irregular o accidentado y superar cualquier tipo de obstáculo ofreciendo imágenes a vista de pájaro y otro tipo de información recogida por diferentes sensores.

Un sistema con múltiples robots UAV es más robusto aún, debido a la redundancia que esto ofrece, permite la cooperación en paralelo entre los drones, ayudándose unos a otros para, por ejemplo, cubrir grandes áreas en exteriores o crear redes de sensores móviles. Estos enjambres de vehículos aéreos no tripulados pueden desplegarse para realizar tareas de búsqueda ante cualquier tipo de desastre natural, como terremotos o ataques terroristas, ayudando a localizar a personas que puedan necesitar ayuda.

Las siglas UAV corresponde en ingles a Unmanned Aerial Vehicle, es decir, vehículo aéreo no tripulado. El origen del desarrollo de estos vehículos pertenece a fines militares, ya que dan la posibilidad de realizar operaciones de alto riesgo, o incluso la de sobrevolar una zona en conflicto para la vigilancia o recojida de información.

Vista la configuración aerodinámica, de apariencia novedosa, que se lleva trabajando en estos particulares sistemas desde hace años, sin embargo, en estos últimos años es cuando su pendiente desarrollo ha sido más exponencial, llegando incluso algunos de ellos a utilizarse de forma recreativa.Existen una gran variedad de UAVs, desde vehículos de grandes dimensiones para altos vuelos y/o grandes distancias de vuelos, hasta pequeñas dimensiones llegando incluso a pocos centímetros.

La investigación a aumentado considerablemente en los ultimos años esa es la razon por la cual existen algunos modelos matemáticos referenciales de este cuadrotor. El desarrollo de un UAV se basa en el tipo de acción que va a llevar a cabo, y la distancia que tendrá que recorrer. A demás de dimensiones específicas, necesita de una tecnología de transmisión y vuelo necesaria para realizar la tarea. En este punto es necesario aclarar que vehículo no tripulado, UAV, no tiene un significado autónomo, sino que estará comunicado desde un operador de tierra, sean pilotos, controladores o cualquier otro tipo de operario relacionado con la monitorización de la aeronave.

Una vez superado el reto de la creación de vehículos no tripulados, se investigó en otro nivel los llamados UAS, Unmanned Aircraft System . Un UAS se trata de la evolución directa de un UAV. Los UAVs pueden estar controlados remotamente desde una estación de tierra por un operador, en cambio, los UAS son autónomos y seguirán una trayectoria ya predefinida, o un vuelo con los recursos de los propios sensores. Existen dos estaciones que pueden manejar información del UAV, la estación de tierra y la estación a bordo del UAV. Dependiendo de cuan autónomo sea el UAV, la estación de tierra realizará más o menos funciones de forma habitual.

2.1.1. Aviones UAV con motor a reacción

En uso militar predominan los UAV de tipo avión, con motores de propulsión a reacción o de pistón. Dentro de los UAV, existe otra denominación para un uso en combate, que es UCAV, Unmmaned Combat Air Vehicle, denominados vehículos aéreos no tripulados de combate.

Un ejemplo de UCAV propulsado con reactores, es el Barracuda proyectado conjuntamente entre España y Alemania, a través de EADS. Fig 2.1

Figura 2.1: Barracuda

Con el mismo fin de desarrollo se encuentra el X-45, creado por la empresa Americana Boeing fue parte del proyecto J-UCAS de DARPA. Fig 2.2

Figura 2.2: X-45

2.1.2. Vehículos UAV, con motor de pistón

UAV con propulsión de tipo pistón se encuentra Predator, sirve principalmente en misiones de reconocimiento, pero además, tiene capacidad ofensiva con la posibilidad de incorporarle dos misiles. En la siguiente imagen además del propio avión podemos ver parte de la cabina de control de tierra, compuesta en su totalidad por una plantilla de 55 personas. Fig. 2.3

Figura 2.3: Predator

2.1.3. Vehículos no tripulados de peso reducido

El más novedoso es el Honeywell RQ-16A T-Hawk, desarrollado por Honeywell, es un pequeño UAV con un motor de gasolina propulsado por un único ventilador colocado en su centro. Su fin es la vigilancia y reconocimiento, este modelo no dispone de armas de defensa u ofensiva, pero tiene un largo alcanzo en proporción a su peso de solo 8.3 Kg. Su reducido peso y tamaño hace que entre dentro de la denominación MAV, del inglés, Micro Aerial Vehicle.

Su creación surgió gracias a la agencia DARPA, responsable de la investigación en el ámbito militar, y actualmente se sigue utilizando en Afganistán por el ejército de los EEUU. Al ser un vehículo no tripulado, no supone riesgo en labores de reconocimiento. Fig. 2.4

Figura 2.4: T-Hawk

En el contexto del desarrollo de vehículos no tripulados, en su mayoría para un fin militar, tanto de grandes dimensiones, o de pequeñas dimensiones como el RQ- 16 T-Hawk, aparecen numerosos modelos para un uso recreativo. Se distribuyen multitud de modelos que se pueden calificar, al igual que los anteriores, en su propulsión o configuración aerodinámica.

Pero el más innovador, fue el Ardrone 1.0 de Parrot, empresa francesa que lo desarrolló y distribuye, con una configuración de cuadricóptero con motores eléctricos.

Se diferencia de otros modelos ya que posee un microprocesador competente para el procesamiento de las señales recibidas a través de una serie de sensores. Además incluye dos cámaras que le permiten captar lo que ocurre a su alrededor. Incorpora un cambió en la estación de control, ya que necesita de un Smartphone para el control del mismo.

Posee una conexión Wi-Fi, y tiene un techo de altitud limitado por la propia conexión Wi-Fi. Debido a la necesidad de una estación de control, también pertenece, en su versión sin modificaciones, al sector de los UAV.

La llegada de un nuevo modelo, el Ardrone 2.0, con diferentes cámaras y sensores, ha sido necesario un nuevo modelo de sistema de control para este novedoso modelo. Fig. 2.5

Figura 2.5: AR.Drone 2.0

La versión 2.0 es más potente en todas sus características importantes para su navegación y estabilización, pero que aun mantiene la esencia de su predecesor con la conexión Wi-Fi, la distribución de sus dos cámaras y las pequeñas dimensiones.

El reto de este proyecto es hacer el control y estabilización en diferentes trayectorias a traves de modelos matematicos ya establecidos a nivel de investigación y principalmente haciendo uso de un software en codigo abierto, en este caso ROS el cual nos aporta diferentes paquetes de navegación y control, y nos permitira hacer la navegación y el control de nuestro vehículo aéreo no tripulado.

2.2. Normativa mexicana CO-AV-23/10

Con base en la circular emitida por la Dirección General de Aeronáutica Civil No. CO-AV-23/10 del estado mexicano se ha definido esta normativa de caracter nacional para impulsar un sector tecnológicamente puntero y emergente. El espacio aéreo está regulado y a veces prohibido para el vuelo Drone en determinados lugares y/o momentos lo que se debe entender, es que no se puede permitir ningún incidente entre paracaídas, ala delta, globo aerostático, planeador, helicóptero, avión de pasajeros o Drone ya que las consecuencias serían trágicas para todos los vehículos y personas involucradas, el respeto de esta normativa significa preservar la libertad que tenemos en México para volar Drones.

Reciben este nombre los vehículos aéreos no tripulados o aeronaves controladas de forma remota, conocidas según sus diversos acrónimos en inglés: UAV, Unmanned Aerial Vehicle y UAS, Unmanned Aerial System. La Organización de Aviación Civil Internacional (OACI/ICAO) a través de la Circular 328 AN/190 del 2011, emitió un pronunciamiento en el cual estableció que RPAS (Remotely Piloted Aircraft Systems) sería la sigla internacional para identificar estas aeronaves. En español son conocidos por el acrónimo VANT (Vehículo Aéreo No Tripulado) o simplemente DRONE, palabra que puede considerarse una adaptación válida al español del sustantivo inglés drone (literalmente abejorro, zángano o zumbador por el ruido que emiten en su operación).

2.2.1. Clasificación

A-AD-S1/S2 Drone recreativo

B-AD-S3/S4 Drone civil equipado con cámara profesional, FPV y/o GPS

Ambos, aparatos de despegue vertical, ala rotativa y multi-rotor que no sean helicopteros: pueden ser tricopteros (3 rotores), quadricopteros (4 rotores), hexacopteros (6 rotores) y octocopteros (8 rotores) controlables en sus tres ejes. Esta clasificación incluye, de manera general, los elementos individuales del

sistema en su conjunto tales como la estación de control en tierra y cualquier otro elemento necesario para facilitar su vuelo, como es el enlace de comunicación (GPS, FPV -FIRST PERSON VIEW-) y el o los dispositivos necesarios para su recuperación. Hay los que se manejan manualmente desde una ubicación remota así como los que vuelan de forma autónoma sobre la base de planes de vuelo pre-programados usando sistemas más complejos de automatización dinámica (despegue, vuelo y aterrizaje automático).

Drone recreativo (Clasificación A-AD-S1/S2)

Vehículo aéreo no tripulado utilizado solo para fines recreativos, con o sin cámara, (Genérica, GoPro, Gear Pro, etc.) que pese menos de 20 kilogramos, exento de bitácora de vuelo, documentos de aeronavegabilidad y autorizaciones de vuelo por los fines que persigue.

Reglas

Este equipo despega, vuela y aterriza solo al alcance de la vista del piloto, por debajo de los 122 metros, en zonas pobladas (siempre que esto no suponga un riesgo claro de daño a otros). El vehículo aéreo deberá portar una etiqueta/placa de 10x5 cm que muestre nombre, dirección y número de teléfono del propietario. Dadas sus características, tiene prohibido volar de noche. Durante el día tiene estrictamente prohibido volar a menos de 3 millas náuticas o 5.5 km de un aeródromo o infraestructura para el despegue o aterrizaje de helicópteros y aviones (espacio aéreo controlado o regulado considerado restringido, peligroso o prohibido).

En esta clasificación entran los vehículos cuyos propietarios no comercializan las fotos o videos que toman, por lo que tampoco deberán entregarlas en forma gratuita como parte de una actividad comercial. Todas las fotos y videos generadas pueden ser compartidas en redes sociales y blogs siempre y cuando se adjunten los datos de indentificación del piloto, mencionando que fueron tomadas para fines recreativos, de investigación o academicos.

Drone civil (Clasificación B-AD-S3/S4)

Vehículo aéreo no tripulado utilizado para actividades especiales, comerciales o multimision (vigilancia, medición, fotografía o video aéreo), que pesa mas de 20 kilogramos y sujeto a la autorizacion de operación, aprobación de tipo, certificado de aeronavegabilidad especial, categoria experimental, matriculacion y registro asi como a responsabilidad civil. .

Reglas

Este equipo puede despegar, volar y aterrizar lejos del alcance de la vista usando una cámara FPV (que retransmite la imagen a tierra -al Piloto- en tiempo real), adicional a la que hace la toma fotográfica o de video (Sony, Canon, Nikon, Black Magic, RED, Kinemini, etc.) solo por debajo de los 122 metros de altura a 30 metros de distancia de personas, vehículos y edificios.

Este vehículo aéreo deberá estar equipado con medios para poder conocer su posición exacta, un sensor barométrico para determinar la altitud y automáticamente evitar exceder la altura permitida así como con un dispositivo capaz de forzar el retorno al punto de partida, en el caso de pérdida de control del piloto. Al igual que el Drone de clasificación A-AD-2014 , este deberá portar una etiqueta/placa de 10x5 cm que muestre el nombre, dirección y número de teléfono del piloto.

Tiene estrictamente prohibido volar a menos de 3 millas náuticas o 5.5 km de un aeródromo o infraestructura para el despegue o aterrizaje de helicópteros y aviones (espacio aéreo controlado o regulado, restringido, peligroso o prohibido).

Todas las vías de desarrollo que se han investigado en el presente proyecto en el apartado Trabajos futuros, necesitan de una normativa para integrar de manera segura los UAV en el espacio aéreo civil. Estas aplicaciones pueden interesar incluso a los propios gobiernos ya que podrían ser misiones de vigilancia de fronteras, recogida datos o patrulla marítima.

Los pioneros en la regulación sobre el espacio aéreo son Suiza, Reino Unido o Australia, legislan las operaciones aéreas mediante normas nacionales. En otros países las operaciones de cualquier UAV requiere de una autorización específica por parte de las autoridades aeronáuticas.

Los principales organismos de gestión y control del tráfico aéreo son, FAA (Estados Unidos) y EASA (Europa) están realizando estudios sobre cómo integrar las aeronaves no tripuladas en sus respectivos espacios aéreos, y por tanto establecer unos mínimos sobre los que certificar estas aeronaves.

La FAA, en conjunción con la Universidad del MIT, estudia dirigir la regulación de las aeronaves no tripuladas realizando estudios de tamaño, masa, probabilidad de fallo o probabilidad de impacto contra aeronaves tripuladas.

Según estos estudios, las aeronaves inferiores en masa 14kg, donde estaría el ardrone 2.0, serían reguladas mediante una serie de normas para el uso civil. Estas normas, serían similares a las reglas seguidas por los aficionados de R/C, no registradas como leyes, y sí como normas de clubes de R/C, basadas en consejos de la FAA. Estas normas, establecen cotas de operación muy bajas, mantener siempre el contacto visual con la aeronave y se restringe su uso fuera del espacio aéreo controlado, lejos de zonas "problemáticas" como sendas de aproximación y despegue de aeropuertos.

Para los UAV de peso superior establecen unas normas homogéneas a las ya existentes para el control del espacio aéreo, aplicando las mismas normas que para aeronaves tripuladas que operan en VFR, como son los ultraligeros. Otra opción sería la de operaciones de UAS del tipo HALE, que vuelan a una cota superior al espacio aéreo que usan los vuelos comerciales.

2.3. Normativa Europea

Por otro lado en Europa, se está trabajando para que UAS de gran tamaño puedan utilizar el espacio civil en 2016. La Unión Europea trabaja para el desarrollo de las aplicaciones civiles de los Sistemas Aéreos Pilotados Remotamente.

Las única regulación existente son las normas de conductas emitidas por las asociación AUVSI, Asociación Internacional de Vehículos no Tripulados. Estas normas buscan realizar unas directrices para concebir seguridad, y acelerará la confianza pública en estos sistemas.

Las normas emitidas por la AUVSI son las siguientes.

Seguridad

• No operaremos UAS de forma que represente riesgo para las personas o propiedades en la superficie o en el aire.

•Aseguraremos que los UAS serán pilotados por personas que están debidamente entrenadas y tienen competencias para operar vehículos o sus sistemas.

• Aseguraremos que los vuelos de los UAS serán realizados solo después de una rigurosa valoración de los riesgos asociados con la actividad. Esta valoración de riesgos, incluirá, pero no limitará.

• Condiciones climáticas en relación con la capacidad de los sistemas.

• Identificación de normalidad de modos de fallo anticipado (perdida de enlace, fallos de potencia, pérdida de control, etc.) y las consecuencias de los fallos.

• Condiciones físicas de la tripulación para operar el vuelo.

• Cumplimiento de las regulaciones de aviación y de la apropiada operación en el espacio aéreo y procedimientos no nominales.

• Comunicación, comando, control y requisitos de espectro de frecuencia del palead (carga útil).

• Fiabilidad, rendimiento y aeronavegabilidad mediante estándares establecidos.

Profesionalidad

• Cumpliremos con las leyes nacionales, estatales y locales, ordenanzas, acuerdos y restricciones relativos a las operaciones de UAS.

• Operaremos nuestros sistemas como miembros responsables de la comunidad aeronáutica.

• Seremos sensibles a las necesidades de las personas.

• Cooperaremos totalmente con las autoridades nacionales, regionales y locales en el despliegue en respuesta a emergencias, investigación de accidentes y relaciones con los medios.

• Estableceremos planes de contingencia para todos los eventos anticipados y los compartiremos abiertamente con todas las autoridades pertinentes.

Respeto

- Respetaremos los derechos de otros usuarios en el espacio aéreo.
- Respetaremos la privacidad de las personas.
- Respetaremos los asuntos del público así como los relativos a las operaciones de los UAS.
- Apoyaremos la mejora del conocimiento y educación pública sobre las operaciones de los UAS.

2.4. Modelado de un VANT

Para obtener el modelo dinámico basado en ecuaciones que describan la posición y orientación, se supone el quadrotor como un cuerpo rígido en el espacio, sujeto a una fuerza principal (empuje) y tres momentos (pares).

A continuación, de una forma más detallada, se describen las fuerzas y pares actuantes sobre la plataforma.

El par para generar un movimiento de balanceo o de roll (ángulo ϕ) se realiza mediante un desequilibrio entre las fuerzas $f2$ y $f4$. Para el cabeceo o pitch (ángulo θ), el des- equilibrio se realizará entre las fuerzas $f1$ y $f3$. El movimiento en el ángulo de guiñada o de yaw (ángulo ψ) se realizará por el desequilibrio entre los conjuntos de fuerzas ($f1$ y $f3$.) y ($f2$ y $f4$.). Se debe a que los rotores 1 y 3 giran en sentido contrario a los rotores 2 y 4. El empuje total produce que el quadrotor se desplace perpendicularmente al plano de los rotores, y se obtiene como suma de las cuatro fuerzas que ejercen los rotores.

Los quadrotor suelen ser sistemas de vuelo de estructura ligera, por lo que el modelo dinámico debe incluir los efectos giroscópicos resultantes tanto del cuerpo rígido rotando en el espacio, como de la rotación de las cuatro hélices. En el cuadro 2.1se describen tales efectos, donde C representan términos constantes, Ω es la velocidad del rotor, JR es el momento de inercia rotacional del rotor alrededor de su eje, l es la distancia del centro de masa a los rotores, J es el momento de inercia del cuerpo rígido y ϕ, θ y ψ son los denominados ángulos de Tait-Bryan.

Donde:

C = Representa terminos constantes

Ω= Velocidad del rotor

JR= El monto de inercia rotacional del rotor alrededor de su eje

L= Es la distancia del centro de masa a los rotores

J= Es el momento de inercia del cuerpo rígido

ф, ϑ y ψ = Ángulos de Tait-Bryan.

Cuadro 2.1: Efectos fÍsicos actuantes

EFECTOS	FUENTES	FORMULACIÓN
Efectos Aerodinamicos	-Rotación de los rotores, - giro de hélices	$C\Omega^2$
Pares Inerciales Opuestos	-Cambio en la velocidad de rotación de los rotores	$JR\Omega$
Efecto de la gravedad	-Posición del centro de masa	L
Efectos giroscópicos	-Cambio en la orientación del cuerpo rígido	$JR\Omega\theta, \phi$
Fricción	-Todos los movimientos del helicóptero	$C(\dot{\phi}, \dot{\theta}, \dot{\psi})$

El cuadrotor es un sistema mecanico subactuado de 6 grados de libertad y 4 entradas de control.

A continuación, de una forma mas detallada, se describen la fuerzas y pares actuantes sobre la plataforma.

En esta sección se presenta la estimación de la orientación y posición inercial del vehículo. El cuadrotor, como solido rígido, esta caracterizado por un sistema de coordenadas ligado a él y con origen en su centro de masas. El sistema de ejes ligado a él está determinado por $B = \left\{\overrightarrow{xL}, \overrightarrow{yL}, \overrightarrow{zL}\right\}$, donde ele eje $\overrightarrow{xL}$ es la dirección normal de ataque del helicóptero, $\overrightarrow{yL}$ es ortogonal a $\overrightarrow{xL}$ y es positivo hacia la derecha

mirando en el sentido positivo de éste, mientras que $\overrightarrow{zL}$ está orientado en sentido ascendente y ortogonal al plano $\overrightarrow{xL}$ O $\overrightarrow{yL}$. El sistema de coordenadas inercial $I = \{\overrightarrow{x}, \overrightarrow{y}, \overrightarrow{z}\}$se considera fijo con respecto a la tierra. Se determina $\xi = \{x, y, z\}$como la posición del centro de masa del helicóptero con respecto al sistema inercial I, y la orientación del vehículo se supondrá dada opor una matriz de rotación R_I: $B \to I$, donde $R_I \in$ SO (3) es una matriz de rotación ortonormal.

la rotación de un cuerpo rígido puede ser obtenida mediante ángulos de Euler, o cuaterniones, por ejemplo. Existen varias definiciones de los ángulos de Euler para representar la orientación relativa de dos sistemas de coordenadas. La mas popular en ingeniería aeroespacial es la convención x, y, z (giro alrededor de x, y', z'') y es conocida como ángulos de Tait-Bryan, también conocidos por ángulos "cardano".

Los ángulos de Tait-Bryan se utilizan para describir una rotación general en el espacio Eucledio tridimensional a través de tres rotaciones sucesivas en torno de ejes del sistema móvil en el cual están definidos. Así, podremos utilizar los ángulos de Tait-Bryan para describir la orientación del cuadrotor.

La rotación de un cuerpo rígido en el espacio se realiza mediante tres rotaciones sucesivas.

1. Rotación según $\overrightarrow{x}$ de ϕ: el primer giro es el correspondiente al ángulo de roll o de balanceo, ϕ, y se realiza alrededor del eje $\overrightarrow{x}$.

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\operatorname{sen}\phi \\ 0 & \operatorname{sen}\phi & \cos\phi \end{bmatrix} \begin{bmatrix} x_L \\ y_L \\ z_L \end{bmatrix} \tag{2.1}$$

2. Rotación según $\overrightarrow{y}$ de θ: el segundo giro se realiza alrededor del eje $\overrightarrow{y}$ a partir del nuevo eje , $\overrightarrow{yL}$con el ángulo pitch o de cabeceo, θ, para dejar el eje $\overrightarrow{zL}$ en su pocición final.

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \operatorname{sen}\theta \\ 0 & 1 & 0 \\ -\operatorname{sen}\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \tag{2.2}$$

3. Rotación según $\overrightarrow{z}$ de ψ: el tercer giro y la ultima rotación corresponde al angulo yaw o de guiñada, ψ, y se realiza al rededor del eje$\overrightarrow{z}$ a partir del nuevo eje $\overrightarrow{zL}$ para dejar el cuadrotor en su posición final.

$$\begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix} = \begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix} \begin{bmatrix} \cos\psi & -\operatorname{sen}\psi & 0 \\ \operatorname{sen}\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} \tag{2.3}$$

Las matrices de rotación definen algebraicamente lo que es una rotación en un espacio 3D considerando un angulo en el que está girando, las matrices de rotación tienen unas propiedades que son importantes de notar:

- Sus ejes de coordenadas son vectores ortogonales (forman un angulo de 90 grados entre ellos).

- Su determinante es 1

- Si se saca la normal de cualquier vector perteneciente a la matriz el resultado es 1 por lo que es una matriz unitaria

- Al ser una matriz ortogonal su transpuesta es igual a su inversa

Pero ¿Para qué usarlo en la robótica?, la respuesta a esta pregunta es que la matriz de rotación define los movimientos de objetos rígidos, por lo que es ideal para su uso en la robótica. Dentro de la robótica podemos pensar que un objeto puede girar sobre diferentes ejes, prácticamente siempre en 3D, por lo que podemos introducir dos nuevos conceptos el de marco global y el de marco de referencia.

El marco global es el punto en donde comienza o está su robot y el marco de referencia es con respecto a que esta girando de manera que se puede obtener coordenadas con respecto a cualquiera de estos dos marcos, en pocas palabras dan posición del robot. Para obtener estas coordenadas se usan transformaciones que involucran a la matriz de rotación.

Las matrices de rotación que representan la orientación del cuerpo rígido rotando alrededor de cada eje y queda expresada como:

$$R(x,\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\text{sen}\phi \\ 0 & \text{sen}\phi & \cos\phi \end{bmatrix} \tag{2.4}$$

,

$$R(y,\theta) = \begin{bmatrix} \cos\theta & 0 & \text{sen}\theta \\ 0 & 1 & 0 \\ -\text{sen}\theta & 0 & \cos\theta \end{bmatrix} \tag{2.5}$$

,

$$R(z,\psi) = \begin{bmatrix} \cos\psi & -\text{sen}\psi & 0 \\ \text{sen}\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

La matriz de rotación completa de B respecto a I, llamada matriz de cosenos directores, viene dada por. $R_I = R(z,\psi).R(y,\theta).R(x,\phi)$

$$R_I = \begin{bmatrix} \cos\psi & -\text{sen}\psi & 0 \\ \text{sen}\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} \cos\theta & 0 & \text{sen}\theta \\ 0 & 1 & 0 \\ -\text{sen}\theta & 0 & \cos\theta \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\text{sen}\phi \\ 0 & \text{sen}\phi & \cos\phi \end{bmatrix} \tag{2.7}$$

$$R_I = \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\text{sen}\theta\text{sen}\phi - \text{sen}\psi\cos\phi & \cos\psi\text{sen}\theta\cos\phi + \text{sen}\psi\text{sen}\phi \\ \text{sen}\psi\cos\theta & \text{sen}\psi\text{sen}\theta\text{sen}\phi + \cos\psi\cos\phi & \text{sen}\psi\text{sen}\theta\cos\phi - \cos\psi\text{sen}\phi \\ -\text{sen}\theta & \cos\theta\text{sen}\phi & \cos\theta\cos\phi \end{bmatrix} \tag{2.8}$$

La matriz de rotación expresada en el sistema de coordenadas B es la traspuesta de R_I, por su propiedad ortonormal, y viene dada por :

$$R_B = \begin{bmatrix} \psi\cos\theta & \text{sen}\psi\cos\theta & -\text{sen}\theta \\ \cos\psi\text{sen}\theta\phi - \text{sen}\psi\cos\phi & \text{sen}\psi\text{sen}\theta\text{sen}\phi + \cos\psi\cos\phi & \cos\theta\text{sen}\phi \\ \cos\psi\text{sen}\theta\cos\phi + \text{sen}\psi\text{sen}\phi & \text{sen}\psi\text{sen}\theta\cos\phi - \cos\psi\text{sen}\phi & \cos\theta\cos\phi \end{bmatrix} \tag{2.9}$$

A partir de la matriz de rotación, se pueden obtener las ecuaciones cinemáticas de rotación del cuadrotor que establecen las relaciones entre las velocidades angulares.

Se una matriz ortonormal $\mathbf{R}$,donde:

$$\mathbf{R}^T\mathbf{R} = \mathbf{I}_n \tag{2.10}$$

y su derivada respecto al tiempo es:

$$\dot{\mathbf{R}}^T\mathbf{R} + \mathbf{R}^T\dot{\mathbf{R}} = \mathbf{0}_n \tag{2.11}$$

definiendo:

$$\mathbf{S} = \mathbf{R}^T\dot{\mathbf{R}} \tag{2.12}$$

se obtiene a partir de la ec.anterior que:

$$\mathbf{S}^T + \mathbf{S} = \mathbf{0}_n \tag{2.13}$$

Donde S es anti-simétrica. La relación entre la derivada de la matriz ortonormal y la matriz anti-simétrica es la siguiente:

$$\mathbf{S} = \mathbf{R}^{-1}\dot{\mathbf{R}} \tag{2.14}$$

Por lo tanto, las ecuaciones cinemáticas para determinar la orientación del helicóptero suponiendo la matriz de rotación, vienen dadas por:

$$\dot{\mathbf{R}}_I = \mathbf{R}_I.\mathbf{S}(\omega) \tag{2.15}$$

donde ω=$[p, q, r]^T$son las velocidades angulares en el sistema de coordenadas ligado al cuerpo rígido y $\mathbf{S}(\omega)(\mathbf{S}(\omega)(\bullet) = \omega x \bullet)$ es la siguiente matriz anti-simétrica:

$$\mathbf{S}(\omega) = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \tag{2.16}$$

Manipulando matemáticamente se obtiene:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \mathsf{sen}\phi\mathsf{tan}\theta & \mathsf{cos}\phi\mathsf{tan}\theta \\ 0 & \mathsf{cos}\phi & -\mathsf{sen}\phi \\ 0 & \mathsf{sen}\phi\mathsf{sec}\theta & \mathsf{cos}\phi\mathsf{sec}\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.17}$$

La variación de los ángulos de Tait-Bryan $(\dot{\phi}, \dot{\theta}, \dot{\psi})$es una función discontinua. Hay que distinguirla de las velocidades angulares en el sistema de referencia asociado al cuerpo rígido (p, q, r) las cuales pueden medirse con giróscopos. Normalmente, se utiliza la IMU, de inglés Inertial Measurement Unit o Unidad de Medida Inercial, para medir las rotaciones y calcular los á ngulos de Tait-Bryan.

La relación entre las velocidades angulares en el sistema fijado al cuerpo y la variación en el tiempo de los ángulos de Tait-Bryan se obtiene a traves de la inversión del Jacobiano de 2.14 y viene dada por:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\mathsf{sen}\phi \\ 0 & \mathsf{cos}\phi & \mathsf{sen}\phi\mathsf{cos}\theta \\ 0 & -\mathsf{sen}\phi & \mathsf{cos}\phi\mathsf{cos}\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.18}$$

El movimiento rotacional del cuadrotor viene dado por las componentes de las velocidades angulares: velocidad angular de balanceo (p) , velocidad angular de cabeceo (q), y velocidad angular de guiñada (r) , sobre los ejes $\overrightarrow{x_L}, \overrightarrow{y_L}$ y $\overrightarrow{z_L}$ respectivamente. Las velocidades angulares son debidas a los pares ligados al cuerpo rígido producidos por las fuerzas externas, las cuales definen momentos en los tres ejes: momento de balanceo (L), momento de cabeceo (M), y momento de guiñada (N) sobre los ejes $\overrightarrow{x_L}, \overrightarrow{y_L}$ y $\overrightarrow{z_L}$ respectivamente.

El movimiento de traslación viene dado por la velocidad $\mathbf{v} = [u_0 v_0 w_0]^T$ en ejes inerciales, y relacionada con la velocidad absoluta del helicóptero expresada en $\mathbf{B}, \mathbf{V} = [u_L v_L w_L]$ mediante la expresión:

Con base a las fuerzas y los momentos que actuan en un cuadrotor, se realizo una estimación del modelo dinámico del cuadrotor Fig. 2.6

Partiendo de la matriz de rotación R_I mencionada anteriormente se tiene que la dinamica de traslación es :

$$\sum F_I = m\dot{V}_I \tag{2.19}$$

donde : V_I=(v_x, v_y, v_z),

de igual manera la dinamica de rotación es expresada de la siguiente manera:

$$\sum M_B = J\dot{\Omega} + \Omega x J\Omega \tag{2.20}$$

donde: J=momento de inercia,$\Omega = (p, q, r)$, vector de velocidad de rotación

Matriz de inercia :

$$J = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \tag{2.21}$$

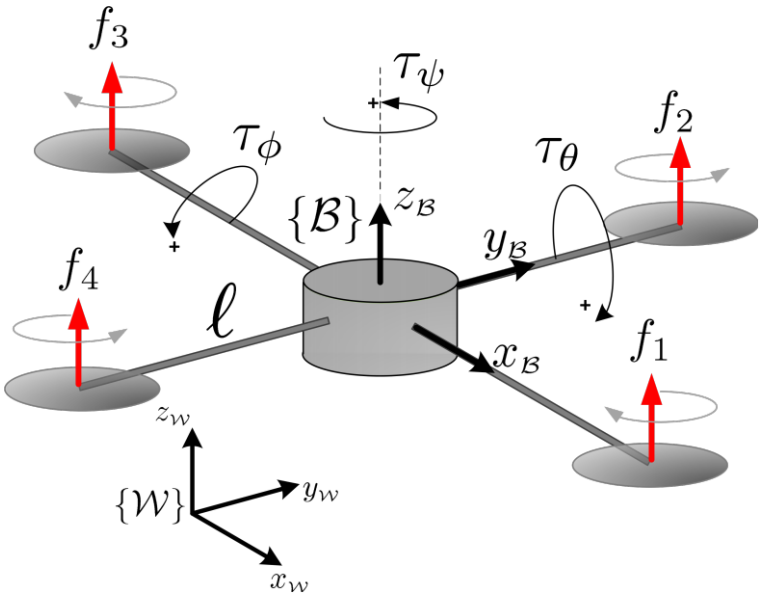

Figura 2.6: fuerzas y momentos 2.6

Velocidad de rotación:

$$\Omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\text{sen}\theta \\ 0 & \text{cos}\phi & \text{sen}\phi\text{cos}\theta \\ 0 & -\text{sen}\phi & \text{cos}\phi\text{cos}\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.22}$$

Entradas: $T = f_1 + f_2 + f_3 + f_4$ peso=fuerzas de empuje

Momentos de los 4 motores :

$\tau\phi = l(f_2 - f_4)$

$\tau\theta = l(f_1 - f_3)$

$\tau\psi = -\tau_{R1} + \tau_{R2} - \tau_{R3} + \tau_{R4}$

Equilibrio de Fuerzas y Momentos :

$$\sum F_1 = \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} + R_B \begin{pmatrix} 0 \\ 0 \\ -T \end{pmatrix} + F_A + F_D$$

$$\sum M_B = \begin{pmatrix} L_\phi \\ L_\theta \\ L_\psi \end{pmatrix} + \begin{pmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{pmatrix} + \tau_A + \tau_D$$

donde:

$$R_B \begin{pmatrix} 0 \\ 0 \\ -T \end{pmatrix} = \begin{pmatrix} (-\text{cos}\psi\text{sen}\theta\text{cos}\phi - \text{sen}\psi\text{sen}\phi)T \\ (\text{cos}\psi\text{sen}\phi - \text{sen}\psi\text{sen}\theta\text{cos}\phi)T \\ (-\text{cos}\theta\text{cos}\phi)T \end{pmatrix}$$

de la dinamica de traslación $\dot{V}_I = \begin{bmatrix} \dot{V}_x \\ \dot{V}_y \\ \dot{V}_z \end{bmatrix}$

$$\begin{bmatrix} \dot{V}_x \\ \dot{V}_y \\ \dot{V}_z \end{bmatrix} m = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + \begin{bmatrix} (-\text{cos}\psi\text{sen}\theta\text{cos}\phi - \text{sen}\psi\text{sen}\phi)T \\ (\text{cos}\psi\text{sen}\phi - \text{sen}\psi\text{sen}\theta\text{cos}\phi)T \\ (-\text{cos}\theta\text{cos}\phi)T \end{bmatrix} + \begin{bmatrix} F_{Ax} \\ F_{Ay} \\ F_{Az} \end{bmatrix}$$

de donde obtenemos:

$$\dot{V}_x = F_{Ax} - [\cos\psi\,\mathrm{sen}\theta\cos\phi + sen\psi\,\mathrm{sen}\phi)]\,\frac{T}{m} \tag{2.23}$$

$$\dot{V}_y = F_{Ay} + [\cos\psi\,\mathrm{sen}\phi - \mathrm{sen}\psi\,\mathrm{sen}\theta\cos\phi)]\,\frac{T}{m} \tag{2.24}$$

$$\dot{V}_z = F_{Az} + g - \cos\theta\cos\phi\frac{T}{m} \tag{2.25}$$

de la dinámica de rotación :

$\sum M_B = J\dot{\Omega} + \Omega x J\Omega$

$J\dot{\Omega} = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \dot{p} & I_x \\ \dot{q} & I_y \\ \dot{r} & I_z \end{bmatrix}$

$\Omega x J\Omega = \begin{bmatrix} i & -j & k \\ p & q & r \\ pI_x & qI_y & rI_z \end{bmatrix} = q_r(I_z - I_y) + p_r(I_x - I_z) + p_q(I_y - I_x)$

$\therefore\ J\dot{\Omega} + \Omega x J\Omega\ =\ \begin{bmatrix} \dot{p} & I_x+ & q_r(I_z - I_y) \\ \dot{q} & I_y+ & p_r(I_x - I_z) \\ \dot{r} & I_z+ & p_q(I_y - I_x) \end{bmatrix}$ de donde podemos decir que $\begin{pmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{pmatrix} + \begin{pmatrix} \tau_{Ax} \\ \tau_{Ay} \\ \tau_{Az} \end{pmatrix} + \begin{pmatrix} L_\phi \\ L_\theta \\ L_\psi \end{pmatrix} = \sum M_B$

y si igualamos y despejamos $\dot{\Omega}$queda de la siguiente manera

$\dot{p}I_x + q_r(I_z - I_y) = \tau_\phi + \tau_{Ax} \Longrightarrow \dot{p} = \frac{\tau_\phi}{I_x} + \frac{[I_y - I_z]}{I_X} q_r + \tau_{Ax}$

$\dot{q}I_y + p_r(I_x - I_z) = \tau_\theta + \tau_{Ay} \Longrightarrow \dot{q} = \frac{\tau_\theta}{I_y} + \frac{[I_z - I_x]}{I_y} p_r + \tau_{Ay}$

$\dot{r}I_z + p_q(I_y - I_x) = \tau_\psi + \tau_{Az} \Longrightarrow \dot{r} = \frac{\tau_\psi}{I_z} + \frac{[I_x - I_y]}{I_z} p_q + \tau_{Az}$

$\therefore$obtenemos lo siguiente.

$$\dot{p} = \frac{\tau_\phi}{I_x} + \frac{[I_y - I_z]}{I_X} q_r + \tau_{Ax} + \frac{I_r}{I_x} q\Omega_r \tag{2.26}$$

$$\dot{q} = \frac{\tau_\phi}{I_x} + \frac{[I_y - I_z]}{I_X} p_r + \tau_{Ay} + \frac{I_r}{I_y} q\Omega_r \tag{2.27}$$

$$\dot{r} = \frac{\tau_\phi}{I_x} + \frac{[I_y - I_z]}{I_X} q_p + \tau_{Az} \tag{2.28}$$

de la matriz de velocidad de rotación Ω se obtiene un sistema de 3 ecuaciones y 3 incognitas (p, q, r) el cual obtenemos $(\dot{\phi}, \dot{\theta}, \dot{\psi})$.

$\Omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\mathrm{sen}\theta \\ 0 & \cos\phi & \mathrm{sen}\phi\cos\theta \\ 0 & -\mathrm{sen}\phi & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$ se obtiene $p = \dot{\phi} - \dot{\psi}\mathrm{sen}\theta$, $q = \dot{\theta}\cos\phi + \dot{\psi}\cos\phi$, $r = -\dot{\theta}\mathrm{sen}\phi + \dot{\phi}\cos\theta\cos\phi$

obtenemos:

$$\dot{\phi} = rtan\theta\cos\phi + p + qtan\theta sec\phi \tag{2.29}$$

$$\dot{\theta} = q\cos\phi - rsen\phi \tag{2.30}$$

$$\dot{\psi} = rsec\theta\cos\phi + qsec\theta sec\phi \tag{2.31}$$

Capítulo 3

Hardware y software del sistema

3.1. AR.Drone 2.0

El Parrot AR.Drone es un vehículo aéreo no tripulado radiocontrolado de uso recreativo civil, funciona propulsado por cuatro motores eléctricos en configuración cuadricóptero y es similar en su estructura básica y aerodinámica a otros modelos radiocontrolados, pero se diferencia de todos ellos en que cuenta con un microprocesador y una serie de sensores, entre los cuales se incluyen dos cámaras que le permiten captar lo que ocurre a su alrededor, más un conector Wi-Fi integrado que le permite vincularse a dispositivos móviles personales que cuenten con los sistemas operativos iOS, Android o Linux. Esto permite controlar al cuadricóptero directamente desde un dispositivo móvil, mientras se reciben por medio de este dispositivo las imágenes y datos de telemetría de lo que los sensores del drone están captando.

El drone fue diseñado originalmente para ser controlado por medio de los productos de Apple con iOS tales como los iPhone, iPad y iPod Touch y para dispositivos Android. Pero debido a que la empresa fabricante del drone liberó los applet de control bajo código abierto pronto aparecieron aplicaciones para otros dispositivos tales como el Samsung BADA y también algunas aplicaciones no oficiales para Symbian, actualmente hay disponible un paquete de desarrollo oficial, de descarga libre, que permite utilizar el drone por medio de una pc portátil equipada con Linux en particular Ubuntu.

La forma de control de los dispositivos incluidos en esté trabajo de investigación es mediante hardware y software, con diferentes aplicaciones. Fig. 3.1

Figura 3.1: AR.Drone 2.0

Otra de sus características, es que no dispone de un mando físico por radiocontrol para su manejo, si no que se vincula a través de su conexión Wi-Fi a un dispositivo móvil con los sistemas operativos iOS de Apple, o Android de Google. Esta conexión le permite ser controlado por los dispositivos móviles e incluso ser controlado a través de la recepción de las imágenes y datos de telemetría obtenidos por el propio ardrone.

Para el uso regular, se necesita tener instalado una aplicación en el Smartphone para el manejo del cuadricóptero. Estas aplicaciones oficialmente solo están desarrolladas para iOS o Android, ya que, en la

actualidad, son los principales sistemas operativos, a nivel de cuota de mercado, en dispositivos móviles. Pero debido a que la empresa Parrot público el control a nivel bajo han aparecido aplicaciones no oficiales en otros sistemas operativos. Fig. 3.2

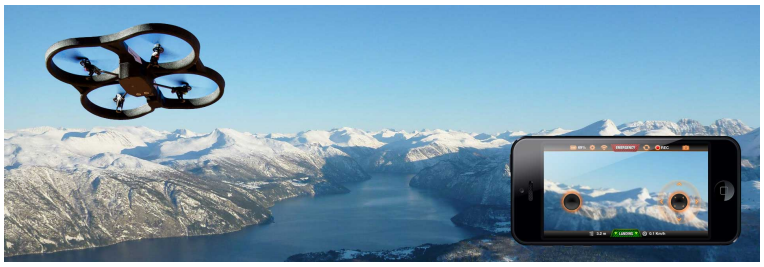

Figura 3.2: AR.Drone comercial

3.2. Especificaciones técnicas del hardware

El núcleo de este ardrone, versión 2.0, es un microprocesador 1GHz de 32 bit ARM Cortex A8 con 800MHz. También dispone de una memoria DDR2 RAM de 1Gb. El sistema operativo introducido es Linux en su versión 2.6.32. Dispone de una conexión USB 2.0 de alta velocidad, para extensión con una memoria extraíble. El microprocesador dispone de un conector Wi-Fi b, g, n. Fig. 3.3

Figura 3.3: Placa madre AR.drone 2.0

Gracias a estas altas características de procesamiento para un vehículo aéreo recreativo, es una buena herramienta para un desarrollo más allá de un pilotaje básico. Además, disponemos de varios sensores que abren el camino de desarrollo para un pilotaje autónomo.

3.2.1. Sistema de propulsión

El ardrone 2.0 es un vehículo aéreo con un movimiento físico propulsado por cuatro motores eléctricos basado en una estructura aerodinámica en configuración de cuadricóptero. Fig. 3.4

Figura 3.4: Motor AR.Drone 2.0

Los cuatro motores de rotor interno sin escobillas, hacen que el dispositivo tenga una alta velocidad de reacción en el aire y pueda, dentro de su categoría de vuelo, ascender a alta distancia. La fuente de alimentación de la que dispone es una batería de tres celdas de Litio-Polímero con capacidad de 1000mAh a 11.1V y capacidad de descarga de 10C, cumple con la normativa de seguridad UL2054. El cumplimiento con la norma UL2054 hace viable una evolución del aparato hacia un vuelo de mayor longitud y tiempo.

Estos cuatros motores son los responsables, a nivel de hardware, para que se mantenga en el aire, con la ayuda de los comandos a nivel de software emitidos por el sistema de estabilización.

3.2.2. Sensores

Centrándonos en las características técnicas de los sensores, dispone de dos cámaras integradas:

• La principal, una cámara frontal en alta definición de 720p a 30 fps. Fig. 3.5

• La segunda cámara, menos potente, es QVGA de 60 fps y con una colocada de forma vertical con orientación descendente.

Figura 3.5: Camara HD frontal

Esta última además de poder ser utilizada para la recepción de imagen, su característica principal es la medición de la velocidad respecto del suelo. A parte de las cámaras, disponemos de una serie de

sensores útiles para el pilotaje del aparato y las principales herramientas para el sistema automático de estabilización.

• Para el control de altitud integra un sensor de ultrasonico, este genera datos de telemetría en función de los cambios de altitud. Sensor de presión con una precisión de ± 10 Pa.

• Acelerómetro digital de 3 ejes con precisión de ±50mg para monitorizar los movimientos de posición.

• Giróscopio de 2 ejes y giróscopio piezoeléctrico de precisión de 2000o/s para los movimientos de Roll, Pitch, Yaw. Fig. 3.6

• Magnetómetro de 3 ejes con 6° de precisión con fines de orientación.

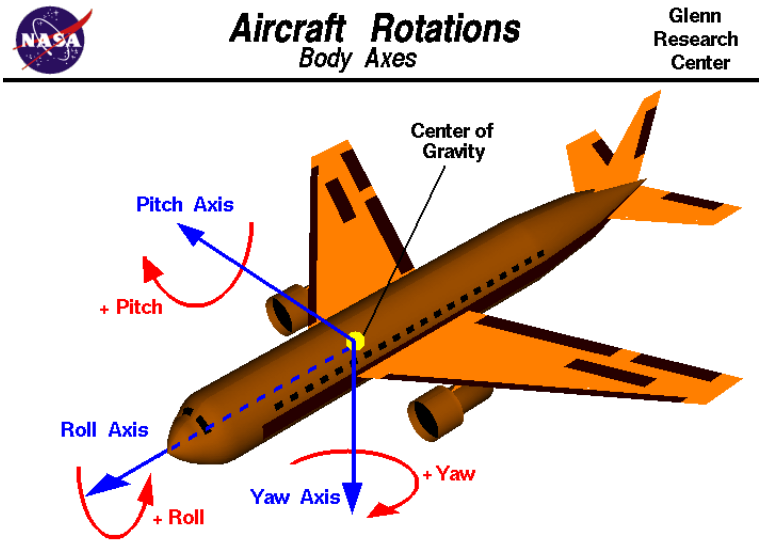

Figura 3.6: roll, pitch, yaw

El conjunto de estos sensores proporcionan toda la información al ardrone 2.0 para la navegación. La fusión de los datos de los sensores proporciona los ángulos de Euler, utilizados para la estabilización. Fig. 3.7

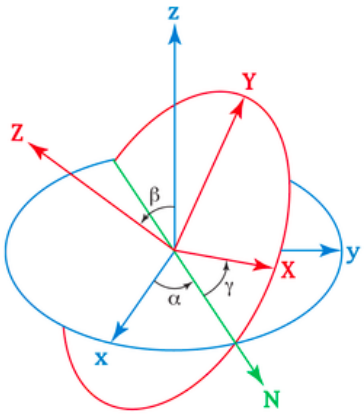

Figura 3.7: Ángulos de Euler

3.3. Robot Operating System (ROS)

ROS fue originalmente desarrolloado en 2007 por el Laboratorio de Inteligencia artificial de Stanford (SAIL) con el soporte del proyecto Standfor AI Robot. Es 2008 se continuó el desarrollo en Willow

Garage, una institución de investigación robótica, con más de 20 instituciones colaborando en el desarollo. En febreo de 2013, ROS se transferió a la Open Source Robotics Foundation.

ROS está liberada bajo los terminos de la licencia BSD (Berkeley Software Distribution) y un software open source. Este gratuito para el uso comercial y de investigación y se podra consultar en la referencia [11].

ROS promueve la reutilización de código, así los desarolladores y científicos no tienen que reinventar la rueda todo el tiempo. Con ROS, puedes hacer esto y mucho más. Puedes coger el código de los repositorios, mejorarlo y compartirlo de nuevo.

ROS es el SDK definitivo para el desarrollo de aplicaciones de robots. Proporciona una arquitectura distribuida y contiene algoritmos implementados del estado del arte, mantenidos por expertos en el campo. Todo con una licencia permisiva que en unos pocos años cambiará el panorama de la robótica yes ampliamente adoptado por la investigación, centros educativos y la industria.

Tenga en cuenta que los nodos de ROS no tienen que estar en el mismo sistema (varios ordenadores) o incluso ser de la misma arquitectura!. Se puede tener un Erle-Brain publicando mensajes y un ordenador portátil suscribirse a ellos. ROS también es de código abierto y es mantenido por muchas personas. Esto hace a ROS muy flexible y adaptable a las necesidades del usuario.

Permite introducir librerías y herramientas para ayudar al desarrollo del software de aplicaciones en el campo de la robótica. Gracias, entre otras cosas, a la abstracción de hardware, los controladores de dispositivo, bibliotecas, visualizadores, paso de mensajes, o gestión de paquetes son algunas de las funciones que nos permite.

Debido a la falta de normativa para el vuelo de estos aparatos, micro UAV, al cual pertenece el ardrone, véase Normativa UAV. Hemos centrado la elección de la vía de desarrollo del sistema, en base a la licencia libre del software.

Actualmente ROS está bajo licencia de código abierto, la licencia BSD, la licencia BSD tiene menos restricciones en comparación con otras como GPL, estando muy cercana al dominio público. Permite el uso del código fuente en software no libre.

3.3.1. ¿ Qué es ROS ?

ROS es un meta-sistema operativo de código abierto para el desarrollo en el sector de la Robótica. Provee diferentes servicios que se esperarían de un sistema operativo, como abstracción de hardware, control de dispositivos en bajo nivel, funciones que se usan comúnmente, comunicación de procesos mediante mensajes y administración por paquetes.

Pero decimos que ROS es un meta-sistema operativo porque además, también comparte características con algunos sistemas middleware (acoplamiento clasificación y envío por el paso de mensajes) y frameworks (callbacks).

Comparando ROS con la mayoría de sistemas middleware y frameworks, ROS impone una leve política restrictiva en el desarrollador, tanto en términos de API, como en problemas de licencia.

La comunidad de este meta-sistema operativo aumenta dado que la curva de aprendizaje no es demasiado alta, y la mayoría del código puede ser fácilmente trasladado, una vez que se entiende lo básico ROS. Fig. 3.8

ROS nos proporciona herramientas y librerías que permiten obtener, construir, escribir y ejecutar programas entre varios computadores, como se ha definido en el apartado anterior, es similar en algunas características con otros frameworks usados en robótica, tales como Player, Orocos, Yarp, Orca, Microsoft Robotics Studio, entre otros.

3.3.2. Objetivo de ROS

El objetivo de ROS es el de implementar todas las conexiones a nivel bajo, para acelerar el proceso de desarrollo, e incluso poder portar las aplicaciones a otros robots gracias a su sistema general de mensajes y nodos. La posibilidad de portar las aplicaciones es una gran ventaja, ya que avances que se realicen para el ardrone, pueden funcionar con otros dispositivos.

Entre otras ventajas tenemos:

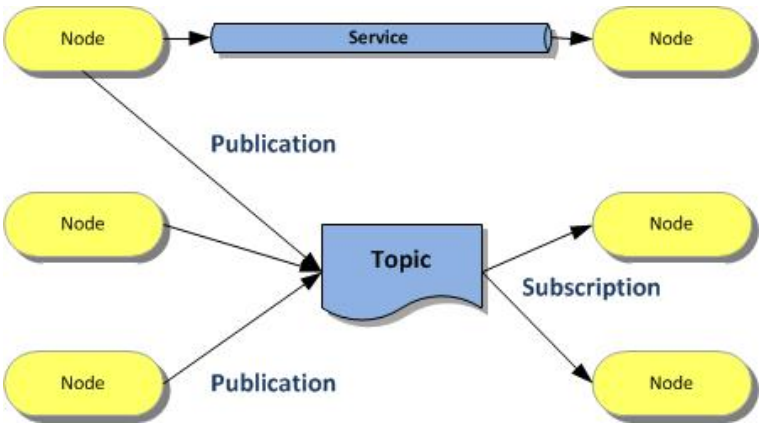

Figura 3.8: Relación nodos y topic

Independencia de idiomas: es fácil de implementar en cualquier lenguaje de programación moderno, el principal es C++. Aunque, ya se ha implementado en Python, C + +, y Lisp, y existen bibliotecas experimentales en Java y Lua.

Test fácil: ROS posee un test, llamado rostest que hace que sea fácil de someter a pruebas.

Escala: ROS es adecuado para sistemas pequeños, pero también, para sistemas grandes y ejecución de los procesos de desarrollo de gran tamaño.

3.3.3. Áreas de aplicación de ROS

Algunas áreas de uso de ROS son:

- Publicación o subscripción de flujos de datos: imágenes, estéreo, láser, control, actuador, contacto, etc.
- Multiplexación de la información.
- Testeo de sistemas.
- Percepción
- Identificación de Objetos.
- Segmentación y reconocimiento.
- Reconocimiento facial.
- Reconocimiento de gestos
- Seguimiento de objetos.
- Comprensión de movimiento
- Visión estéreo: percepción de profundidad.
- Robots móviles
- Agarre de objetos

3.3.4. Entorno de ROS

ROS ofrece una infraestructura de publicación-suscripción de mensajes diseñado para apoyar la construcción rápida y fácil de los sistemas de computación distribuida.

Herramienta: proporciona un amplio conjunto de herramientas para configurar, iniciar, introspección, depuración, visualización, registro, control, y detener los sistemas de computación distribuida.

Capacidades: ofrece una amplia colección de bibliotecas que implementan la funcionalidad de robot útil, con especial atención a la movilidad, la manipulación y la percepción.

Ecosistema: ROS es apoyado y mejorado por una comunidad grande, con un fuerte enfoque en la integración y la documentación.

3.3.5. Conceptos generales de ROS

Para entender el sistema ROS debemos conocer unos sencillos conceptos propios de este sistema. Aunque ROS actualiza anualmente la versión del software estos conceptos no cambian, manteniendo su estructura.

ROS posee tres niveles de conceptos:

- Nivel del sistema de archivos.
- Nivel de Computación Gráfica.
- Nivel comunitario

3.3.6. ROS nivel del sistema de archivos

En este nivel nos referimos a la totalidad de archivos necesarios para el funcionamiento de ROS, no nos referimos al sistema de conexiones o envío de mensajes a nivel interno.

Paquetes: unidad principal en ROS, puede contener procesos de ejecución (nodos), una biblioteca ROS-dependiente, conjuntos de datos, archivos de configuración, o cualquier otro archivo que se útil para la organización de ROS.

Manifiestos: residentes en manifest.xml proporcionan datos sobre un paquete, incluyendo su información de licencia y dependencias, así como información específica del idioma.

Pilas: Pilas o stacks son colecciones de paquetes que proporcionan funcionalidad agregada, como una "stack de navegación".

Manifiestos de pila o stack: se trata del archivo stack.xml, proporcionan datos sobre una pila, incluyendo su información de licencia y sus dependencias en otras pilas.

3.3.7. Nivel de computación gráfica

El gráfico de la computación es la red peer-to-peer de los procesos de ROS que están procesando los datos conjuntamente. Se trata de un gráfico bastante útil en el desarrollo, ya que de una forma sencilla, podemos observar las conexiones internas en ROS.

Los conceptos básicos de computación gráfica en ROS, nos proporcionan los datos de la gráfica de diferentes maneras. Son los siguientes:

Nodos: Los nodos son procesos que llevan a cabo cálculos. Por ejemplo, en el caso de un sistema de control de un robot, en nuestro para el ardrone 2.0, un nodo se encargara del movimiento de navegación, otro de la búsqueda de patrones, y además dispondremos de varios nodos para la navegación autónoma.

Los nodos son ejecutables que pueden comunicarse con otros procesos utilizando temas, los servicios o el servidor de parámetros, el uso de nodos en ROS nos proporciona tolerancia a fallos y separa el código y funcionalidades que el sistema sea más simple, un nodo debe tener un nombre único en el sistema, este nombre se utiliza para permitir que el nodo pueda comunicarse con otro nodo utilizando su nombre sin ambigüedad, un nodo puede ser escrito utilizando diferentes bibliotecas como roscpp y Rospy en el que roscpp es para C ++ y Rospy es para Python.

ROS tiene herramientas para manejar los nodos y nos dará información sobre el mismo, como rosnode, el rosnode es una herramienta de línea de comandos para mostrar información sobre los nodos.

Master: El master ROS proporciona registro de nombres y la búsqueda para los nodos. Sin los nodos no sería capaz de encontrar mensajes entre sí, intercambio, o invocar los servicios.

Parámetro servidor: El servidor de parámetros permite que los datos se almacenarán con llave en un lugar central. En la actualidad es parte del Master.

Mensajes: los nodos se comunican entre sí por el paso de mensajes. Un mensaje es simplemente una estructura de datos, que comprende los campos con tipo estándar (entero, flotante, booleano, etc.).

Topics o temas: Los mensajes toman rutas a través de un sistema de transporte de publicación o suscripción semántica. Un nodo envía un mensaje mediante su publicación a un tema determinado, el tema es un nombre que se utiliza para identificar el contenido del mensaje un nodo que está interesado en un determinado tipo de datos se suscribirá al tema correspondiente puede haber varios editores y

suscriptores concurrentes a un mismo tema, y un único nodo puede publicar y / o suscribirse a múltiples temas.

ROS tiene una herramienta para trabajar con temas llamados rostopic, es una herramienta de línea de comandos que nos da información sobre el tema o publica los datos directamente en la red.

`rostopic bw /topic`: Esto muestra el ancho de banda utilizado por el tema.

`rostopic echo /topic:` Imprime los mensajes de la pantalla.

`rostopic find message_type:` Esto encuentra los temas por su tipo.

`rostopic hz /topic:` Esto muestra la tasa de publicación del tema.

`rostopic info /topic:` Imprime información sobre el tema activo, los temas publicados, los que está suscrito a, y servicios.

`rostopic list:` Imprime información sobre temas activos.

`rostopic pub /topic type args:` Esta publica datos con el tema, nos permite crear y publicar los datos en cualquier tema que queremos, directamente desde la línea de comandos.

`rostopic type /topic:` Imprime el tipo tema, es decir, el tipo de mensaje que publica.

Servicios: El modelo de publicar/subscriptor es un paradigma de comunicación muy flexible, pero en muchos casos el transporte en un único sentido no es suficiente para las interacciones de petición y respuesta que a menudo se requieren en un sistema distribuido. La petición y respuesta se realiza a través de los servicios, que se definen a partir de una de estructuras de mensajes: una para la petición y otra para la respuesta. Un nodo que proporciona un servicio con un nombre y un cliente que lo utiliza dicho servicio mendiante el envío del mensaje de petición y espera a la respuesta. La librería cliente de ROS generalmente presenta esta interacción como si fuera una llamada a procedimiento remoto

Bolsas: Las bolsas es un formato para guardar y reproducir de nuevo mensajes de ROS. Las bolsas son un mecanismo importante para el almacenamiento de datos, como lecturas de sensores, que pueden ser dificilmente adquiridas pero son necesarias para el desarrollo y testeo de algoritmos.

3.3.8. ROS nivel comunitario

ROS permite el intercambio de recursos o información, entre grupos de investigación o universidades, proporcionando vías para el intercambio, algunas fuentes de intercambio son:

• Repositorios: ROS se basa en una red de repositorios de código, donde diferentes instituciones, como programadores o laboratorios, pueden desarrollar y lanzar sus propios componentes de software del robot.

• El Wiki ROS: ROS comunidad Wiki es el foro principal para documentar la información sobre ROS. Cualquier persona puede inscribirse para una cuenta y contribuir con su propia documentación, facilitar las correcciones o actualizaciones, escribir tutoriales véase .

• ROS answers: respuestas a preguntas y respuestas lugar de responder a sus preguntas relacionadas con ROS. No se trata de un foro, sino que es un sistema sencillo de pregunta y respuesta.

• Blog: multitud del blogs destinados al desarrollo de aplicaciones y tutorial de ROS el más importante el blog de Willow Garage, ofrece actualizaciones periódicas.

3.3.9. Instalación de ROS Indigo en Ubuntu (Linux)

Como bien se realizó la investigación detallada sobre ROS, en esta subseción se comprendera de como instalar ROS en la versión 14.04, suponiendo que ya se tiene instalado ubuntu en el equipo en la versión más actualizada, si utiliza normalmente un sistema operativo diferente y usted no tiene una partición libre en el disco duro, también puede instalar Ubuntu en una memoria USB (de arranque), en este caso, se necesitan al menos 4 GB de almacenamiento adicional.

Se comprende las etapas principales de la instalación de ROS indigo para ubuntu.

En primer lugar abrir desde el equipo una terminal y agregar las claves y los registros respectivos:

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu precise main" > /etc/apt/sources.list.d/ros-latest.list'
$ wget http://packages.ros.org/ros.key -O - | sudo apt-key add -
```

```
$ sudo apt-get update
```

Ahora instale ROS y algunos paquetes de útiles:

```
$ sudo apt-get install ros-indigo-desktop-full ros-indigo-joystick-drivers python-rosinstall python-rosdep liblapack-dev
libblas-dev daemontools libudev-dev libiw-dev
```

Crea tu carpeta de espacio de trabajo ROS

```
$ mkdir ~/workshop
```

Añadir comandos de ROS y la carpeta de espacio de trabajo a tu .bashrc : Abra en su editor de texto favorito (por ejemplo, utilizando gedit ~ / .bashrc), y anexar los siguientes dos líneas al final:

```
$ source /opt/ros/indigo/setup.bash
```

```
$ export ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:~/workshop
```

Revise su instalación abriendo una nueva ventana de consola y empezar

```
$ roscore
```

Todo está bien cuando vea la línea "iniciado servicio básico [/ rosout]" . A continuación, puede dejar de roscore pulsando CTRL-C .

Descargue el código fuente en su espacio de trabajo ROS.

```
$ cd ~/workshop $ GIT_SSL_NO_VERIFY=1 git clone https://github.com/AutonomyLab/ardrone_autonomy.git -b indigo
```

```
$ GIT_SSL_NO_VERIFY=1 git clone https://github.com/tum-vision/autonavx_ardrone.git
```

Compilar los paquetes. Tenga en cuenta que el proceso de compilación requiere acceso a Internet y tarda unos 10-20 minutos estos paqutes son para la navegacíon auntónoma de un vehículo aéreo en especial para un Drone 2.0 de 4 rotores.

```
$ roscd ardrone_autonomy
```

```
$ ./build_sdk.sh
```

```
$ rosmake ardrone_autonomy ardrone_joystick
```

3.3.10. Instalación de Driver para ROS indigo

En la plataforma Ubuntu y para ROS Indigo, Hydro y Groovy puede instalar el controlador abriendo una nueva terminal y escribir los siguiente.

```
$ apt-get install ros-*-ardrone-autonomy
```

```
$ apt-get install ros-hidro-ardrone
```

Compilar desde el código fuente el paquete de AR-Drone SDK tiene su propio sistema de construcción que por lo general maneja anchas dependencias del sistema, el paquete ROS depende de estos paquetes ROS estándar: `roscpp, image_transport, sensor_msgs, tf, camera_info_manager,` nav_msgs and `std_srvs.`

La instalación sigue los mismos pasos necesarios normalmente para compilar un conductor ROS usando amento <**http://wiki.ros.org/catkin**> . Clone (o descargar y descomprimir) el conductor a la `src` carpeta de un nuevo o existente amento espacio de trabajo <**http://wiki.ros.org/catkin/Tutorials/create_a_worksp** por ejemplo `~ / catkin_ws / src` , a continuación, ejecute `catkin_make` y compilarlo. Suponiendo que usted está compilando para ROS Indigo :

```
$ cd ~/catkin_ws/src
```

```
$ git clone https://github.com/AutonomyLab/ardrone_autonomy.git -b indigo-devel
```

```
$ cd ~/catkin_ws $ rosdep install --from-paths src -i
```

```
$ catkin_make
```

3.3.11. Ejemplo práctico

Ahora que se ha instalado ROS, para empezar a usarlo se tiene el siguiente ejemplo práctico y concultarlo en [9], usted debe proporcionar su sistema con la ruta donde está instalado ROS, abrir una nueva terminal desde su equipo y escriba el siguiente comando.

```
$ roscore
```

```
$ roscore: command not found
```

Si el comando no se ejecuta le marcara como error donde le dira comando no encontrado, este mensaje es debido a que su sistema no sabe dónde buscar los comandos, para resolverlo, escriba el siguiente comando en una en una nueva terminal.

```
$ source /opt/ros/indigo/setup.bash
```

A continuación, escriba el comando roscore una vez más, y verá la siguiente salida:

```
...
Press Ctrl-C to interrupt
started roslaunch server http://192.168.7.2:52131/
ros_comm version 1.11.10
SUMMARY
========
PARAMETERS
* /rosdistro: indigo
* /rosversion: 1.11.10
NODES
auto-starting new master
process[master]: started with pid [5437]
ROS_MASTER_URI=http://192.168.7.2:11311/
setting /run_id to 8e8dcaf5-fd39-11e4-bff2-6c400899ab38
process[rosout-1]: started with pid [5440]
started core service [/rosout]
```

Esto significa que su sistema sabe dónde encontrar los comandos para ejecutar ROS.

3.4. Simulador interactivo de un VANT

En el desarrollo de este proyecto se realizaron simulaciones que nos permiten visualizar trayectorias realizadas por un VANT, asi como la obtención de datos como son las velocidades en los ejes **x**, **y**, y **z** con respecto al tiempo de vuelo como podemos observar en la siguiente Fig. 3.9 se utiilzó un simulador que nos muestra los diferentes movimientos que puede realizar el vehículo aéreo asi como se realizarón trayectorias y obtención de graficas en 3D, la posición, temperatura y presión este simulador interactivo, es de mucha importancia para entender como se comporta un VANT en un espacio tridimensional, consultar la referencia [3] para poder usar el simulador y poder ser descargado en un equipo siguiendo las instrucciones necesarias.

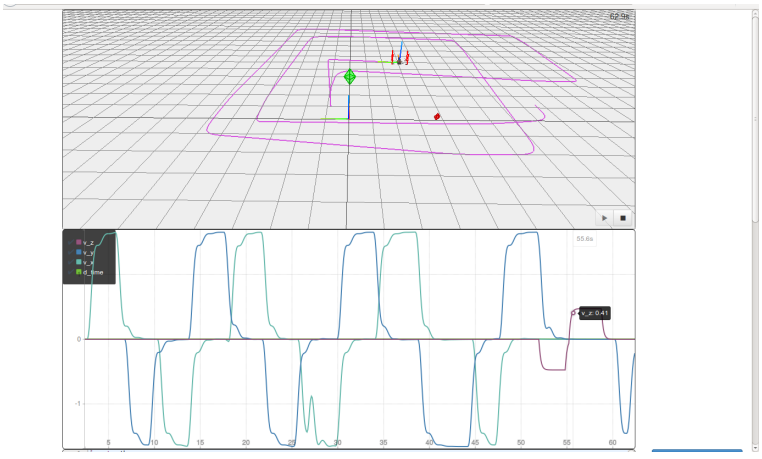

Figura 3.9: velocidades y tiempo **v_x,v_y,v_z** y **d_tim**

El Simulador Interactivo nos genera muchas ventajas para entender realmente como hacer un vuelo en diferentes trayectorias, esté simulador se puede interactuar y también poder descargarlo para su equipo desde

Control de la cámara

- Mantenga pulsado el botón izquierdo del ratón para rotar la cámara

- Mantenga pulsado el botón derecho del ratón para mover la cámara alrededor

- Utilice la rueda de desplazamiento para acercar y alejar

Control del teclado Quadrotor

- Teclas W y S controlan el ángulo de paso

- Teclas A y D controlan el ángulo de balanceo

- Teclas Q & E controlar el ángulo de guiñada

- Teclas de arriba y abajo controlan la velocidad vertical a lo largo de los ejes Z

de esta manera se puede interactuar en el manipulador interactivo con el cual se obtuvo un buen conocimiento de como poder controlar un VANT en un entorno real.

Capítulo 4

Diseño e implementación del controlador

En este capitulo se describen 2 tipos de control para llegar a tener un control autónomo con el VANT pero para poder lograr una interfas entre computadora y el VANT se necesita un software que nos permite la recepción y envío de datos basada en Linux y lograr la navegación autónoma a si como su respectivo control de un punto de posición en un espacio tridimensional, y que la vez el software trabaje con codigo abierto, conforme a lo investigado y visto en los capitulos anteriores fue que se aplicó ROS, y es el entorno en el que se ha finalizado el proyecto a si como también se ara uso de un lenguaje de programación conocido como Python, en el cual nos permite crear nodos para la navegación autónoma y control del VANT.

Se utilizaron varios paquetes el principal se trata del paquete ardrone_autonomy, que es el encargado del envío y recepción de mensajes del VANT con el ordenador en este caso una PC en el cual nosotros podemos observar todos los datos de navegación en tiempo real, de igual forma se tiene el paquete ardrone_python que es donde encontramos diferentes programas y nodos de python donde nosotros podemos modificar la estructuctura de cada programa y hacer uso de ello para poder manipular el propio drone.

4.1. Controlador PID

Un controlador PID es un controlador de tres terminos que tiene una larga historia en el campo de control automático, el nemónico PID se refiere a las primeras letras de los nombres de sus terminos individuales, estos son P por la parte proporcional, I por la parte integral y D por la parte derivativa.

Los controladores PID probablemente sean los mas usados en la industria, esto es debido a su relativa simplicidad y su desempeño satisfactorio en un amplio rango de procesos. Estos han venido evolucionando conforme al progreso de la tecnología y en la actualidad es implementado muy frecuentemente en forma digital

El exito de los controladores PID también es debido al hecho de que usualmente representan el componente fundamental de sistemas de control más sofisticados, que pueden ser implementados cuando una ley de control básica no es suficiente para obtener el desempeño requerido.

El control proporcional-derivativo e integral (PID), es un algoritmo de control para sistemas lineales, la finalidad de este algoritmo es conseguir que el dato de salida sea igual al dato de referencia que se introduzca, a su vez constará de 3 constantes (K_P, K_D, K_I), que se deberán definir según el comportamiento deseado.

La ley de control PID consiste en aplicar correctamente la suma de tres tipos de acciones de control: una acción proporcional, una acción integral y una acción derivativa.

$$U_t = K_p(x_{des} - x_{t-1}) + K_D(\dot{x}_{des} - \dot{x}_{t-1}) + K_I \int_0^t x_{des} - x_{t'-1} dt' \tag{4.1}$$

las constantes K_P, K_D, K_I son las ganancias para cada control y el error es :

$$e = (x_{des} - x_{t-1}) \tag{4.2}$$

donde:

x_{des}= posición deseada

x_{t-1}=posición real

La parte proporcional se obtiene de multiplicar el error de entrada por la constante definida anteriormente (Kp). Esta parte es la que se define como acción-reacción.

La segunda parte es la integral se obtiene de multiplicar la integral del error por la constante Ki. Esta parte del algoritmo se calcula sumando los errores que hemos ido obteniendo, con la finalidad de compensar el error hasta cancelarlo.

La última parte es la derivativa, se obtiene de multiplicar la derivada del error por la constante denominada Kd. La finalidad de este es corregir la sobreactuación que hayamos aplicado para no tener oscilaciones, ya que se anticipa a la salida para disminuir la salida que hayamos obtenido con las 2 partes anteriores.

4.2. Control manual del VANT, basada en Linux

Se trabaja con Linux porque es un Sistema Operativo tipo Unix diseñado para aprovechar al máximo las capacidades de las computadoras PC basadas en el microprocesador i386 y posteriores. Es un SO con capacidades de multiprocesamiento, multitarea y multiusuario. Sin embargo, a diferencia de otros sistemas Unix para PC, usted no tiene que pagar cuantiosas licencias por el uso de Linux.

Linux fue diseñado teniendo en cuenta la portabilidad de las aplicaciones. Linux es compatible con diversos estándares Unix, tales como System V, BSD y los estándares internacionales IEEE POSIX.1 e IEEE POSIX.2, facilitando el desarrollo de aplicaciones para múltiples plataformas.

Una vez establecidos los conceptos anteriores sobre el sistema LINUX se supero el camino de la comunicación de nuestro vehículo aéreo no tripulado para obtener el control basado en Linux, es por eso que se debe avanzar hacia el movimiento a través de una estación remota, en este caso también basada en Linux, y aplicando el sistema de publicación de mensajes de ROS, desde el sistema operativo ROS montado en nuestro sistema linux con todos los controladores instalados se logró hacer la interfaz entre el VANT y poder tener el control adecuado del drone, se dio uso del teclado de la PC para poder hacer los movimientos básicos de navegación y realizar diferentes trayectorias deceadas por el operador.

Ejecutando los siguientes comandos en una nueva terminal de nuestro equipo nos genera una nueva ventana de video, esto quiere decir que ya podemos hacer uso del teclado de la computadora y poder realizar el vuelo deceado en el VANT, en la Fig. 4.1 se puede ver que el VANT esta siendo controlado, consultar [8] donde encontraran el control de un VANT mediante linux y aplicando ROS.

$roscore

$roslaunch ardrone_tutorials keyboard_controller.launch

los controles definidos en ardrone_tutorials/src/keyboard_controller.py son:

E – Pitch Forward

D – Pitch Backward

S – Roll Left

F – Roll Right

W – Yaw Left

R – Yaw Right

Q – Increase Altitude

A – Decrease Altitude

Y – Takeoff

H – Land

SPACEBAR – PARADA DE EMERGENCIA

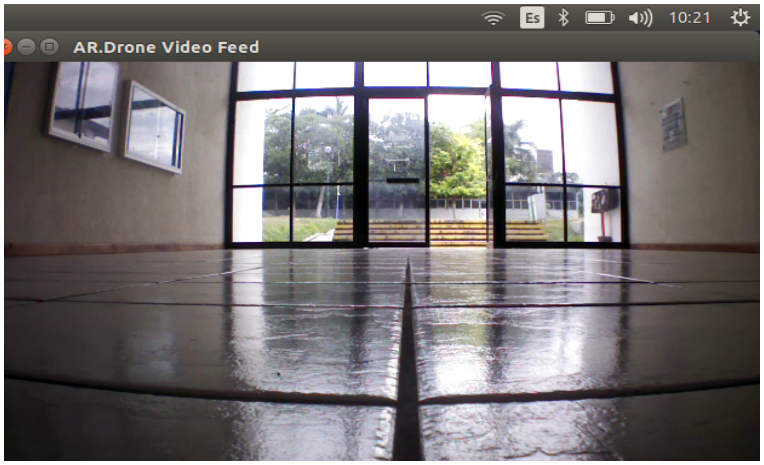

Figura 4.1: Control manual del VANT

Damos click dentro de la ventana de video mostrada anteriormente y pulsando la tecla correspondiente activamos el envío de datos de navegación, y publicar en los diferentes topics para que se produzca el movimiento deseado del VANT y asi tener control absoluto sobre el VANT.

4.3. Estimación de la posición del VANT

Con base a las programaciones realizadas en python, se realizó la trayectoria en tiempo real con nuestro vehículo no tripulado el cual nos permitio obtener nuestros resultados y poder cuestionar nuestras dudas y dar soluciones en la Fig. 4.2 se muestra el vehículo en funcionamiento realizando la trayetoria.

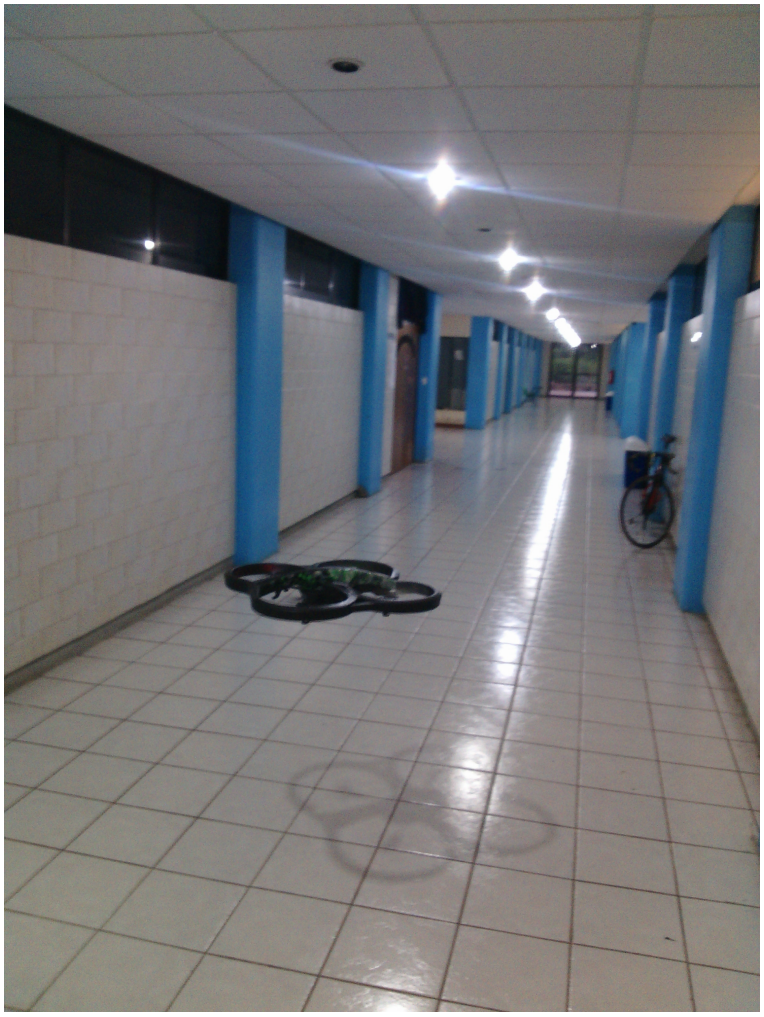

Figura 4.2: Vehículo realizando trayectoria

Para llevar acabo el vuelo y la trayectoria se hizo la siguiente estructura de programación realizada en python en el cual se hizo uso de los ángulos de Euler y matriz de rotación a si como las importaciones de paqutes para la navegaćión autónoma.

```
#!/usr/bin/env python
import robotics1 as rob
import numpy as np
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import math
import rospy
import scipy.io
import roslib; roslib.load_manifest('ardrone_python')
from ardrone_autonomy.msg import Navdata
c = 0
x = 0
y = 0
```

```
i = 0
dx = 0
dy = 0
ta = 0
a = 0
xi = 0
yi = 0
z = 0
T = np.array([[0], [0]])
grados = 0
temp = 0
presion = 0
def callback(navdata):
global ta
global n
global xi
global yi
global x, y, z
global temp
global presion
t = navdata.header.stamp.to_sec()
dt = t - ta
ta = t
T[0] = t
T[1] = dt
dx = navdata.vx * dt
dy = navdata.vy * dt
xi = dx + xi
yi = dy + yi
print("received odometry message: temp=%f presion=%f x=%f y=%f z=%f "%(navdata.temp,navdata.pressure
v = np.array([[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]])
c = np.dot(v,rob.heul2r(navdata.rotX,navdata.rotY,navdata.rotZ,xi,yi,navdata.altd,'deg'))
x = np.append(x, c[0][3]/1000)
y = np.append(y, c[1][3]/1000)
z = np.append(z, c[2][3]/1000)
temp= np.append(temp,navdata.temp)
presion= np.append(presion,navdata.pressure/1000)
try:
if __name__ == '__main__':
rospy.init_node('example_node', anonymous=True)
rospy.Subscriber("/ardrone/navdata", Navdata, callback)
rospy.spin()
mpl.rcParams['legend.fontsize'] = 10
```

```
fig = plt.figure()
ax = fig.gca(projection='3d')
ax.plot(x, y, z, label='projection')
ax.legend()
plt.show()
except KeyboardInterrupt:
print ("terminado")
```

En la siguiente grafica se observa la trayectoria realizada por el VANT. Fig. 4.3

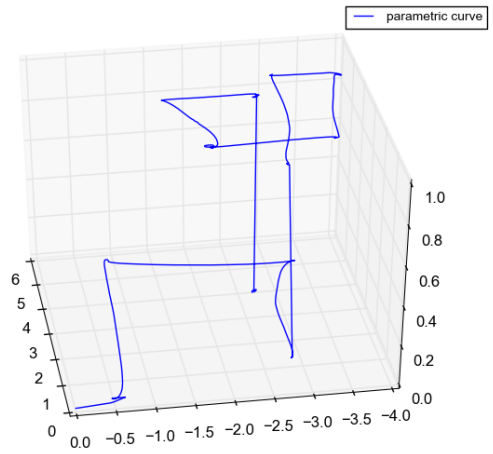

Figura 4.3: Trayectoria recorrida y grafica en 3D.

En esta prueba la trayectoria fue un cuadrado a la que cada punto de trayectoria se le asigno 5m como medida de referencia y como se observa es un recorrido realizado por el VANT en el cual el control de vuelo fue lo esperado a si como la realización del recorrido sobre los puntos marcados, al final de la trayectoria se capturaron los datos necesarios, uno de ellos fue la distancia recorrida por el vehículo en metros (m) y despues de tener esos datos se corroboraron con la ayuda de un flexometro en cual se demuestra que la distancia en metros fue aproximadamente igual considerando un margen de error de un 2 % esto nos conlleva que los resultados y el control son muy factibles para poder generar otro tipo de aplicaciones.

Capítulo 5

Resultados y Discusión

5.1. PID aplicado al VANT

En este capitulo se muestran los resultados obtenidos del control aplicado al VANT, en el cual por medio de sus sensores de medición obtenemos datos de telemetría del vuelo autónomo a si como sus respectivas graficas.

La imagen siguiente el VANT se le aplico la estrategia de control de tipo PID en el cual se le dio un punto de despegue vertical y una altura maxima para los respectivos ejes. Fig. 5.1

Figura 5.1: Control absoluto

En el desarrollo de esta estrategia de control se logró obtener una muy buena estabilidad en vuelo del VANT, se establecio un punto de despegue y desplazamiento de maximo 1m de referencia y despues retomar las diferentes graficas a si como en python se capturó los codigos principales para lograr el control PID, en la siguiente estructura se muestra todo lo que es el control para el VANT.

```
#!/usr/bin/env python
import robotics1 as rob
```

```
import numpy as np
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import rospy
import scipy.io
import roslib; roslib.load_manifest('ardrone_python')
from ardrone_autonomy.msg import Navdata
from std_msgs.msg import String
# note all the distance are in meters and te twist are in m/s
ta = xi = yi = dt = x = y = z = posrl = acel = 0
vela = np.array([[0], [0], [0]])
posa = np.array([[0], [0], [0]])
posdes = np.array([[1], [1], [1]])
#if you wanna change the set point , you need to change the values of below
setpointx = 1
setpointy = 1
setpointz = 1
#Definition of the main class
class UserCode:
def __init__(self):
# Values of Ks
Kp_x = .009 # .75
Kp_y = .01 # .7
Kp_z = 1
Kd_x = .009 # .2
Kd_y = .01 # .2
Kd_z = 1.2
Ki_x = .001 # .001
Ki_y = .001 # .001
Ki_z = .18
self.Kp = np.array([[Kp_x, Kp_y, Kp_z]]).T
self.Kd = np.array([[Kd_x, Kd_y, Kd_z]]).T
self.Ki = np.array([[Ki_x, Ki_y, Ki_z]]).T
def callback(self, navdata):
global x, y, z, ta, xi, yi, ta, posrl, setpointz, setpointx, setpointy, posa, vela
pub = rospy.Publisher('hello_pid', String, queue_size=10)
t = navdata.header.stamp.to_sec()
dt = t - ta
ta = t
# we calculated the differential of distance(x and y)
dx = navdata.vx * dt
dy = navdata.vy * dt
```

```
# we add all the differential of distance to get the real measure of x and y
xi = dx + xi
yi = dy + yi
zi = navdata.altd
print(("received odometry message: x=%f y=%f zi=%f "% (xi, yi, zi)))
#we use the euler angles for calculate the position of our cuaqcopter starting in the origin
v = np.array([[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]]) # if you need start
in other position you change this line
c = np.dot(v, rob.heul2r(navdata.rotX, navdata.rotY, navdata.rotZ, xi, yi, navdata.altd,
'deg'))
posrl = np.array([[c[0][3]/1000], [c[1][3]/1000], [c[2][3]/1000]]) # real position of our
quadcopter
# we store all the position in tree vector each dt
x = np.append(x, c[0][3] / 1000)
y = np.append(y, c[1][3] / 1000)
z = np.append(z, c[2][3] / 1000)
# this vector is for set the setpoint in the plotting
setpointx = np.append(setpointx, posdes[0])
setpointy = np.append(setpointy, posdes[1])
setpointz = np.append(setpointz, posdes[2])
vel = (posrl - posdes) / dt
#acel=(vel-vela)/dt
vela=vel
u = self.Kp * (posdes - posrl) + self.Kd * (np.array([[0],[0],[0]]) - vel) + self.Ki *
(posa-posrl) #control PID
posa = posrl
#print(u)
r1 = str(u[0])
r2 = str(u[1])
r3 = str(u[2])
q = r1.replace("[","")
q1 = q.replace("]","")
q2 = r2.replace("[","")
q3 = q2.replace("]","")
q4 = r3.replace("[","")
q5 = q4.replace("]","")
# we send 3 massages to the second node but we add a flag for indentify the origin of the
messages
L = np.array(['1'+q1, '2'+q3, '3'+q5]) #flag
#print(L)
vec = "%s"%L[0]
#rospy.loginfo(vec)
pub.publish(vec)
vec1 = "%s"%L[1]
```

```
#rospy.loginfo(vec1)
pub.publish(vec1)
vec2 = "%s"%L[2]
#rospy.loginfo(vec2)
pub.publish(vec2)
# This method plot a 3D space
def graf3D(self,x, y, z):
mpl.rcParams['legend.fontsize'] = 10
fig = plt.figure()
ax = fig.gca(projectin='3d')
ax.plot(x, y, z, label='3D')
ax.legend()
plt.show()
# This method plot a 2D space
def graf2D(self,z,setpoint):
plt.plot(z)
plt.plot(setpoint)
plt.show()
# This method plot a 2D space but set 3 graphics in the same window
def subplot(self,x,y,z,setpointx,setpointy,setpointz):
f, axarr = plt.subplots(3, sharex=True)
axarr[0].plot(x, color='r')
axarr[0].plot(setpointx, color='g')
axarr[0].set_title('Control en X')
axarr[1].plot(y, color='c')
axarr[1].plot(setpointy, color='g')
axarr[1].set_title('Control en Y')
axarr[2].plot(z, color='y')
axarr[2].plot(setpointz, color='g')
axarr[2].set_title('Control en Z')
plt.show()
if __name__ == '__main__':
h = UserCode()
rospy.init_node('example_node', anonymous=True)
rospy.Subscriber("/ardrone/navdata", Navdata, h.callback)
rospy.spin()
h.subplot(x, y, z, setpointx, setpointy, setpointz)
scipy.io.savemat('x.mat', mdict={'x': (x)})
scipy.io.savemat('y.mat', mdict={'y': (x)})
scipy.io.savemat('z.mat', mdict={'z': (x)})
#h.graf2D(x,setpointx)
#h.graf2D(y,setpointy)
#h.graf2D(z,setpointz)
```

en la estructura anterior realizada en python se capturaron los datos necesarios para cada control en x,y,z ; como se observa se tiene las ganancias kp,kd y ki para cada control PID en (x,y,z) y respectivos nodos para realizar la navegación y el control autónomo.

se realizarón muchas pruebas para poder llegar a obtener un control adecuado a las ganancias de cada control PID se les proporcionaron los siguientes valores el cual se logró obtener menores oscilaciones y un mejor control para el VANT el cual fue el más factible de todas las pruebas realizadas.

```
Kp_x = .009
Kp_y = .01
Kp_z = 1
Kd_x = .009
Kd_y = .01
Kd_z = 1.2
Ki_x = .001
Ki_y = .001
Ki_z = .18
```

Durante el vuelo se logró obtener un mejor equilibrio para la posición deseada, considerando los errores que nos pueden causar las fuerzas ejercidas por los propios rotores del VANT es de gran ventaja obtener la siguiente grafica Fig. 5.2 en la cual se puede observar el control para cada eje en este caso para x,y,z.

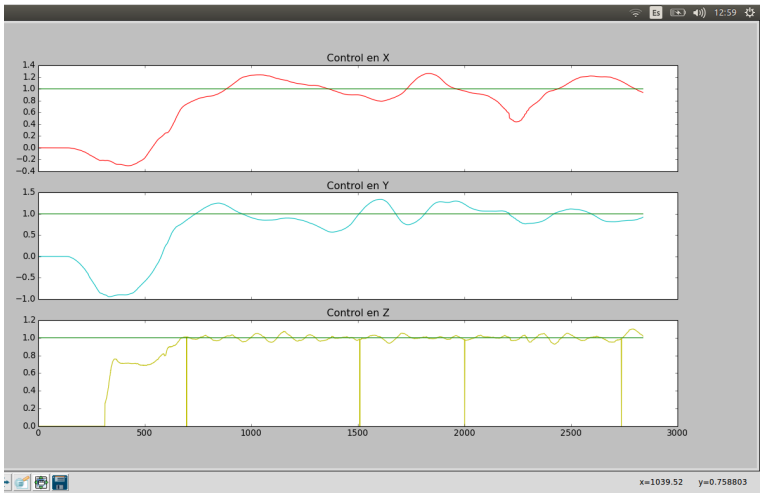

Figura 5.2: Controlador PID para los ejes x,y,z

De la grafica anterior se puede decir que el control fue lo que realmente se esperaba para cada eje se observa que el VANT al realizar el desplazamiento necesario y a la altura maxima de referencia que es de 1m, este se fue equilibrando de la mejor manera como deveria de ser esto demuestra que el control autónomo fue exitoso.

Capítulo 6

Conclusión y Trabajos futuros

Se concluyo que hacer un buen control de un vehículo aéreo nos deja una buena experiencia, demostrada la capacidad del control y de vuelo autónomo de nuestro VANT de cuatro motores, se encuentra disponible de manera absoluta la investigación en estos pequeños aparatos , logramos hacer un buen control de posición asi como un control autónomo, cabe mencionar que los resultados obtenidos son de gran eficiencia y de gran avance para la educación, y de un gran impacto tecnológico, visto el nivel de reacción y la estabilidad gracias a su configuración de cuatro motores, se pueden sugerir multitud de campos de avance.

En el desarrollo del proyecto se ha logrado el control y la manera de publicar y recibir información referida a la navegación, o la telemetría del cuadricóptero. Se obtuvo un punto de inflexión, ya que al conocer el funcionamiento del AR.Drone, y de ROS con el ardrone y los diferentes metodos de control , obtenemos multitud de vías de investigación.

Una vez resuelto el principal objetivo del proyecto, de dotar al ardrone de un software capaz de realizar un vuelo autónomo, se ha investigado en los diferentes caminos de ampliación.

La principal ampliación se centra en la navegación, podríamos dotar al VANT de una navegación autónoma siguiendo una línea recta, al traer integrada una cámara vertical podemos observar el suelo, y con ella la línea guía para realizar ese vuelo.

Además, descubiertas las ventajas de este pequeño vehículo no tripulado con visión por cámara, podemos realizar mapeado 3D a través de una única cámara al dotar la cámara de una de un movimiento físico, como posee el ardrone, podemos llegar a conocer las coordenadas de posición del ardrone en el momento de la toma de imagen.

Cercana a la aplicación obtenida en este proyecto podemos realizar una detección de caras, que puede ser muy útil en tareas de vigilancia o desastres naturales. Sería posible encontrar personas en zonas de difícil acceso, o a modo de vigilante para el reconocimiento de intrusos.

Los cuadricópteros son vehículos aéreos de corta historia, pero con una curva de alto desarrollo en estos momentos. Son innumerables los modelos utilizados en el ámbito militar, mayoritariamente como elemento de vigilancia, pero la mayoría no disponen de vuelo autónomo. Dotando a estos aparatos de un pilotaje autónomo se podrían ganar en seguridad, y en rapidez de reacción, por ejemplo, en la búsqueda de artefactos como minas, e incluso, aparatos autónomos suficientemente potentes podrían rescatar a personas una vez encontradas sin esperar a la llegada de un equipo humano, el muestreo de zonas terrestres de alto riesgo etc.

No se puede estimar la multitud de campos donde estos aparatos pueden avanzar sin embargo el desarrollo de los cuadricópteros es una apuesta segura para el avance en numerosos campos, dando máxima importancia a su característica principal de estabilidad, y posibilidad de reconocimiento del entorno gracias a sensores, entre ellos, las cámaras de alta definición.

Bibliografía

[1] Ferdinand Pierre Beer, E Russell Johnston, José Antonio Gómez-Jurado García, and Juan de la Rubia Pacheco. *Mecánica vectorial para ingenieros*. McGraw-Hill, 1990.

[2] Angulos de Euler. https://es.wikipedia.org/wiki/

[3] edX. https://courses.edx.org/login.

[4] [EN LINEA]. http://upcommons.upc.edu/bitstream/handle/2099.1/12919/memoria.pdf;jsessionid=7d5debbb78f06cf1

[5] Reglamento Mexicano. http://www.aeropuertodrone.com/reglamento.htm.

[6] Katsuhiko Ogata. *Ingeniería de control moderna*. Pearson Educación, 2003.

[7] PDF. https://upcommons.upc.edu/bitstream/handle/2099.1/6930/memoriadef.pdf?sequence=1.

[8] robohub. http://robohub.org/.

[9] Erle Robotics. http://erlerobotics.com/blog/home-creative/.

[10] ROS. http://wiki.ros.org/ros/installation.

[11] ROS. http://www.ros.org/.

[12] tesis. http://tesis.ipn.mx/bitstream/handle/123456789/9717/12.pdf?sequence=1.

[13] Python Tutorials. https://www.youtube.com/watch?v=cjmzdhmhxwu.

[14] ROS Tutorials. http://wiki.ros.org/ros/tutorials.

[15] ROS Wiki. http://wiki.ros.org/.