eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Facultad de Informática
Grado de Ingeniería Informática

Proyecto fin de grado

Ingeniería de Computadores

Experimentos de coordinación multi-robot con quadcopteros

Autor:

Jon Larrañaga Fuerte

Director:

Manuel Graña Romay

2014

informatika fakultatea
facultad de informática

Resumen

El objetivo principal de este proyecto es crear un sistema capaz de controlar varios UAVs y hacer experimentos con ellos de manera coordinada. El UAV utilizado será la plataforma robótica aérea AR.Drone. Estos drones son cuadricópteros con sus cuatro servomotores eléctricos que permiten un control muy robusto de sus maniobras en el aire. El sistema completo estará compuesto por varios drones y un controlador, que en este caso será un ordenador. A partir de la creación de la red compuesta por los drones y el controlador, se detallarán los programas que se han utilizado para controlar los drones, ya sea en los vuelos autónomos o vuelos controlados. El objetivo de estos vuelos será transportar objetos lineales, como pueden ser mangueras o cuerdas, mediante el vuelo coordinado de los drones dotados de sistemas de fijación para que los puedan transportar. Esto es, gracias a un sistema de sujeción que le añadiremos a cada drone probaremos el transporte de varios tipos de cuerdas por dos o más drones a la vez guiados por un solo controlador. Los programas creados tendrán el objetivo de corregir el vuelo e intentar conseguir la estabilidad necesaria para que los drones puedan transportar las cuerdas o mangueras sin perder el control debido a su peso o algún efecto producido por los otros drones, como corrientes de aire inducidas por los rotores. Este proyecto tiene dos partes importantes: La primera es la creación del sistema que nos permite transportar las cuerdas y la segunda es observar y analizar el comportamiento del sistema durante diferentes experimentos.

A la hora de explicar las pruebas experimentales se detallará la situación del sistema con el número de drones y el objeto a transportar. Además, se resumirán los resultados de los experimentos reflejados en valores de parámetros de vuelo recibidos de los drones.

Para acabar, se detallarán las diferentes conclusiones a las que se ha llegado mediante los experimentos y que nos han servido también para escribir las lineas futuras de trabajo que se detallan al final.

Índice

Índice de figuras

Índice de cuadros

Índice de algoritmos

1. Introducción

1.1. Visión general

En la era de la nueva tecnología que avanza a un ritmo frenético, el mundo de la aviación no se ha quedado atrás. Los aviones han evolucionado hasta convertirse en uno de los medios de transporte más utilizados y seguros.

Aparte de los aviones, han ido apareciendo diferentes vehículos aéreos que se utilizan tanto para transporte como para ocio. Esto es, los vehículos aéreos han pasado de tener únicamente la función de transportar personas o mercancías a ser juguetes de control remoto utilizados por niños, creando un mercado que se ha expandido por todo el mundo creándose empresas dedicadas exclusivamente a ello.

Desde que en 1903 los hermanos Wright lograsen volar unos pocos metros con una aeronave propulsada a motor, los cambios que ha sufrido la aeronáutica mundial han sido muy grandes. Las mejoras se han ido sucediendo y se han logrado crear aeronaves con materiales cada vez más ligeros y resistentes, muy diferentes a las primeras aeronaves de madera. Aparte de los materiales, también ha evolucionado el tema del pilotaje, en el que, poco a poco, han ido tomando protagonismo aeronaves controlados a distancia o por ordenadores. Y en este proyecto hablaremos y utilizaremos éstos últimos: los vehículos aéreos no tripulados (UAV Unmanned Aerial Vehicle). Los UAV se han puesto de moda en los últimos años debido a su uso bélico. Es tanta su expansión entre la población que los gobiernos han tenido que promulgar leyes para regular su uso. Por ejemplo, en España la Agencia Estatal de Seguridad Aérea (AESA) es la responsable de controlar el uso de aeronaves controladas por control remoto. Esta entidad ha dictado que no está permitido el vuelo de UAVs en espacios públicos con fines comerciales o profesionales, y, que si un usuario quisiera utilizarlos con algún fin que no sea el de recreo o deportivo, tendrá que requerir una autorización a AESA. En cuanto a los recintos cerrados el uso de estas aeronaves no esta regulado, por lo que se podrán utilizar bajo la responsabilidad del piloto.

1.2. Objetivos

El objetivo general de este proyecto es usar varios drones para conseguir un manejo sencillo de estos vehículos desde un solo computador. La navegación coordinada de varios drones como un único sistema, que llamaremos multidrone, se hará de la forma más fácil posible para la realización de experimentos de campo. La intención de estos experimentos será probar el funcionamiento del multidrone y su uso en una tarea especifica que ocupa al grupo de investigación: el transporte de un objeto casi-lineal flexible, como una cuerda o cable.

Estas ideas se concretan en la siguiente lista de objetivos principales:

1. Puesta en marcha de los AR.Drones
2. Poner en marcha el SDK y los diferentes módulos:
   a) SDK estándar: probar los demos.
   b) Módulos basados en Node.js: Utilizado para acceder a más de un drone a la vez.
3. Crear una red con los drones y el ordenador.

4. Controlar simultáneamente dos o más drones
5. Diseñar el sistema de sujeción de los cables
6. Hacer pruebas de transporte de manguera/cuerdas
   a) dos drones
   b) tres drones

1.3. Alcance

El modelo del drone que utilizaremos para este proyecto será el AR.Drone 2.0 de la marca francesa Parrot. El uso creciente de estos vehículos aéreos ha hecho que este proyecto resulte interesante, ya que se trata de un tema de candente actualidad.

Los usos potenciales de sistemas multidrone son varios: controlar una zona, llevar de manera coordinada varios objetos etc. Como hemos comentado antes, en este proyecto nos ocupamos de transportar objetos casi-lineales simulando una situación en la que el ser humano no pueda transportarlos, como puede ser en casos de incendios o grandes alturas.

1.4. Planificación temporal

En la siguiente tabla se detallarán las diferentes tareas llevadas a cabo para conseguir los objetivos, una descripción de cada tarea y la estimación en horas para llevarlas a cabo.

Tarea	Descripción	Estimación
1. Conocer el proyecto		**50**
a. Recoger información sobre AR.Drone	Funcionamiento, características etc.	25
b. Puesta en marcha del SDK y las demos	Descarga y compilación del SDK y las demos	25
2. Planificación	Planificar el tiempo	**10**
3. Módulos de control		**30**
a. Informarse acerca de los módulos	Informar sobre módulos que controlen varios drones	28
b. Instalación	Instalar los diferentes módulos	2
4. Crear sistema multidrone		**60**
a. Crear la red local	La red que una a los drones y el ordenador	5
b. Controlar el sistema	Control de los drones a la vez desde el ordenador	30
c. Crear programas para corregir vuelos	Programar	25
5. Pruebas		**31**
a. Crear sistema de sujeción	Para poder transportar las cuerdas	1
b. Ejecutar las pruebas	Ejecutar los programas creados para verificarlos, mientras se crean los videos	30
6. Documentación		**120**
a. Crear la memoria	Escribir memoria que describa el proyecto	120
TOTAL		**301**

Cuadro 1: Tareas y estimación

2. Estado del arte

En este capitulo comentaremos el estado del arte relacionado con la coordinación entre UAVs basándonos en [3] como referencia básica. Explicaremos los diferentes pasos que hay que tener en cuenta a la hora de intentar coordinar diferentes UAVs y las principales aplicaciones que puede tener esta coordinación.

En un grupo cooperativo cada uno de los miembros del grupo comparten el mismo objetivo y actúan de acuerdo con el interés común del grupo. La coordinación efectiva a menudo requiere que cada individuo condicione sus acciones al objetivo global. Se pueden adoptar muchas formas de coordinación, pero puede no ser obligatorio que cada miembro esté directamente coordinado con todos los demás para que el grupo tenga un comportamiento coordinado. Cuando queremos crear también un grupo coordinado con UAVs ocurrirá lo mismo.

Para la definición de la cooperación entre UAVs para la vigilancia aérea [3] propone seguir 4 pasos:

- Definir el objetivo de la coordinación y limitaciones
  - Primer paso: Decidir cual es el objetivo de la cooperación
- Definir la función y variables de la coordinación
  - Identificar la información esencial para la cooperación
  - A menudo es posible lograr la cooperación con pequeñas acciones individuales
  - La función de la coordinación será la relación entre el objetivo y la información esencial que debe saber cada drone
- Programa de cooperación centralizada:
  - Derivar la estrategia de cooperación, sabiendo cada miembro del grupo cual es su labor.
- Construcción de consensos
  - Estar seguro de que a pesar de las insuficiencias de la red de comunicación cada drone tiene suficiente información de coordinación.

Las siguientes son aplicaciones principales de la sistemas cooperativos compuestos de UAVs:

- *Cooperative Timing:* Diferentes UAVs tienen que llegar al mismo sitio (el objetivo) a la vez sorteando obstáculos. Esta aplicación de la metodología es interesante para misiones militares. La configuración del mundo viene dada por la ubicación de las amenazas y obstáculos. La variable de decisión es la trayectoria de vuelo. La función de coordinación describe la amenaza y exposición del obstáculo. El objetivo es la de que todos los UAVs atraviesen la zona de obstáculos y que lleguen a su objetivo a la vez.

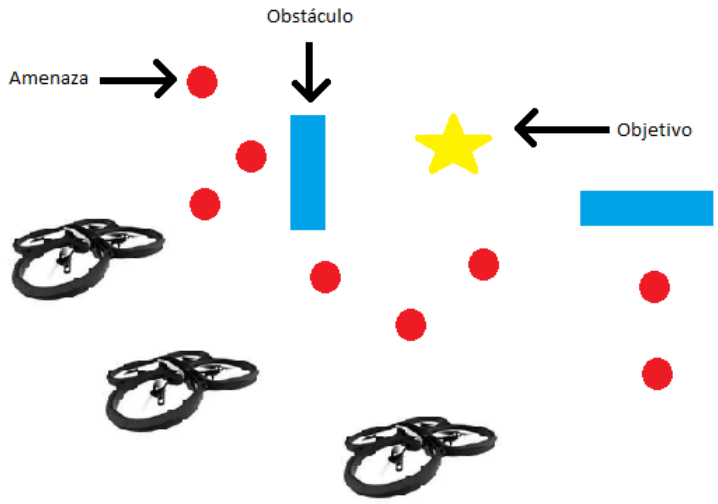

Figura 1: Cooperative Timing

- *Cooperative Search:* Otro ejemplo de cooperación es la búsqueda coordinada. En este problema varios vehículos deben maniobrar alrededor de las amenazas y pasar por encima de los objetivos de observación, mientras mantienen la conectividad de comunicación y evitan colisiones entre ellos o con los obstáculos. El objetivo de cooperación es la de observar el máximo de objetivos utilizando el grupo entero de UAVs mientras atraviesan la zona.

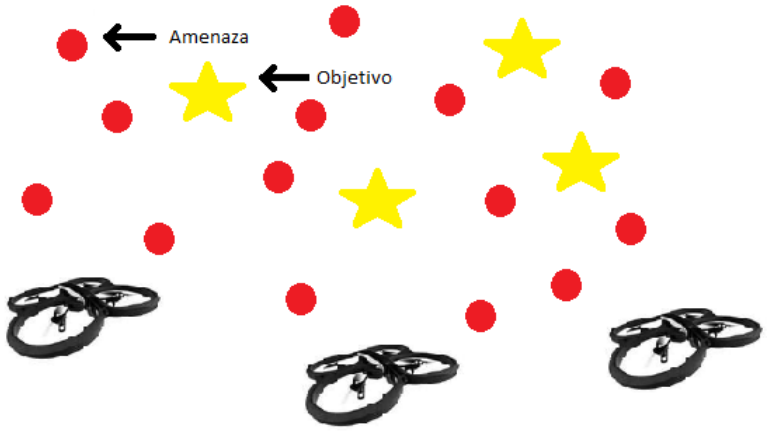

Figura 2: Cooperative Search

- Vigilancia cooperativa de incendios: el último ejemplo es el de la vigilancia de varios UAVs alrededor del perímetro de un fuego. El objetivo es supervisar y realizar el seguimiento del perímetro para comunicar las coordenadas del perímetro a los bomberos. Al tener el alcance de comunicación limitado los UAVs tendrán que trabajar en cooperación para retransmitir el perímetro, esto es, cuando llegan a una base o se acercan a otro UAV se retransmitirán la información.

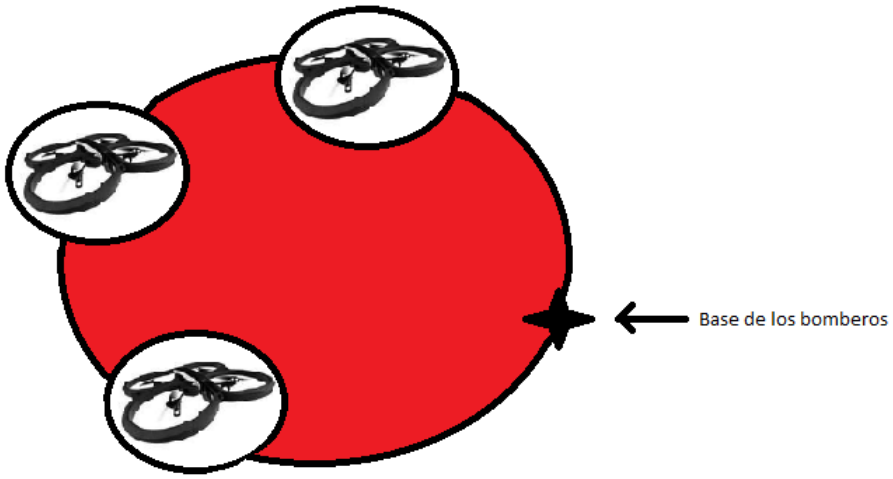

Figura 3: Vigilancia cooperativa de incendios

3. Descripción general

3.1. Drones

Los drones o UAV, conocidos en castellano como vehículos aéreos no tripulados (VANT) fueron creados por militares para su uso no civil, pero como ocurrió con Internet, el avance de las nuevas tecnologías los han socializado trasladando su aplicación a infinitos campos como son el cine o la ciencia. Los VANT tienen una larga historia militar desde antes de la Segunda Guerra Mundial. Durante la Primera Guerra Mundial, fueron desarrolladas las primeras maquetas y ya por los años 40 se utilizaron guiados por radio para entrenar a los soldados con los cañones antiaéreos. Aparte de entrenar a los operarios durante la guerra se consiguió que los drones llevaran explosivos y que fuesen controlados también por radio.

Después de la Segunda Guerra Mundial se empezó a crear drones con el objetivo de reconocer el terreno mientras eran lanzados desde aviones. Estos UAVs tenían programadas diferentes rutas y se convertirían en los primeros drones autónomos.

Con el paso de los años en diferentes guerras, como la de Vietnam o Yom Kippur, se han utilizado los drones con fines bélicos. Luego, en los mediados de los años 80 vendrían los drones con posicionamiento global GPS y largo alcance que han sido utilizados en diferentes conflictos.

Hoy en día, los UAVs son capaces de diferenciar entre aliados o enemigos y se han convertido en maquinas de combate que demasiadas veces ocasionan numerosas victimas inocentes.

Aparte de su uso militar (reconocimiento o ataque) actualmente los UAVs son utilizados con fines científicos o de ocio. Entre estos usos están :

- Extinción o vigilancia de incendios forestales.
- Vigilancia de instalaciones, como centrales nucleares o oleoductos.
- Para vigilar y observar zonas de acceso difícil o restringido, como reservas forestales o estaciones biológicas, luchar contra los cazadores furtivos o analizar los vuelos de las diferentes especies.
- Para tomar muestras en zonas donde la vida humana puede estar en peligro, como puede ser en Chernóbil.
- Para grabar películas. Por ejemplo, en los últimos dos años se han estrenado más de veinte películas que tienen algún plano tomado con drones.

A pesar de sus notables ventajas (uso en zonas peligrosas o la no necesidad de piloto a bordo) el uso de los drones tienen considerables desventajas:

- Los drones pueden ser fácilmente hackeados, posibilitando la opción de que se rompa la comunicación entre el usuario y el drone.
- Debido al balance entre el peso de combustible o batería y la potencia requerida de los servomotores tienen una capacidad de vuelo limitada.
- Debido a su uso masivo de hoy en día se puede utilizar para fines no legales como grabar en zonas privadas.

Para poder controlar los UAVs de forma autónoma, deberán de capturar información sensorial para procesarla. Los sensores de un UAV están relacionados con aspectos típicos de la instrumentalización de una aeronave y pueden recopilar datos relativos a su posición, aceleración, ángulos de giro, velocidad y cualquier otra componente dinámica que intervenga ne la navegación.Los principales sensores de los UAVs son:

- Longitud: La longitud del UAV sobre la Tierra.
- Latitud: Indica la latitud del UAV sobre la Tierra.
- Altitud: El valor métrico que indica la distancia sobre la superficie de su entorno.
- Grados de giro: Sobre cada uno de sus ejes (x, y, z) los giros realizados por el UAV.

Gracias a dispositivos como GPS o brújulas electrónicas, todos estas medidas de sensores se proporcionaran al cliente, ya que los dispositivos estarán instalados a bordo como parte de la instrumentalización.

Según la sofisticación del UAV existirán sensores mas específicos, como pueden ser los sensores de visión o análisis físico del entorno.

El controlador del sistema robótico autónomo tendrá como entrada el conjunto de valores de los sensores implantados en el UAV y podrá proporcionar los valores relativos a las señales de control más adecuados.

3.1.1. Actuadores de un UAV

Como hemos comentado los valores proporcionados por los sensores servirán como entrada para el controlador, que los recopilará, analizará y los empleará para hacer cálculos de las nuevas señales de control que se proporcionará a los actuadores.

Según la morfología del UAV los actuadores serán distintos. Por ejemplo, los UAVs inspirados en aviones emplean alerones y un timón para navegar y cambiar de rumbo. Los UAVs inspirados en helicópteros, sin embargo, poseen una estructura de motores que hacen girar un conjunto de hélices, permitiendo despegar desde un punto fijo, volar, y trazar su rumbo en base a una serie de ángulos de giro.

Entre los actuadores que dispone un UAV están los siguientes:

- Alerones: Los utilizan los UAV basados en aviones y están situados en las alas de la aeronave, facilitando la capacidad de giro de izquierda a derecha de forma rápida.
- Timón: Lo implementan los UAV basados en aviones y se encarga de dotar al UAV de la capacidad de variar su rumbo de forma suave.
- Elevador: variando la potencia de los motores implementados se encarga de proporcionar la capacidad de aumentar o disminuir la posición de altitud del UAV.
- Propulsor: Se sitúa en cada motor del UAV y se encarga de dar potencia a los motores.
- Frenos: Se encargan de reducir la potencia de la aeronave.

3.1.2. Navegación de un UAV

Los mandos de vuelo que permiten la navegación de un UAV, son aquellos mecanismos que están integrados en la aeronave y cuyo objetivo es la de cambiar la orientación y posición en su entorno. Por otra parte, cualquier UAV es capaz de realizar 3 diferentes giros alrededor de sus 3 ejes perpendiculares entre sí, donde su punto de intersección o coordenada $(0, 0, 0)$ está situado sobre el centro de gravedad de la aeronave. Existirán tres ejes que se utilizarán para describir las 3 maniobras que es capaz de hacer el UAV. Los ejes son:

- Pitch: Eje transversal.
- Roll: Eje longitudinal.
- Yaw: Eje vertical.

Teniendo en cuenta estos ejes las maniobras son:

- Cabeceo: Es el movimiento que se realiza alrededor del eje transversal. Se emplea como actuador el elevador que permite el control de los grados de inclinación durante la maniobra de cabeceo. Mediante este movimiento se variará la altitud del UAV, consiguiendo esto gracias a modificar la orientación, esto es, elevando o bajando el morro del avión.
- Alabeo: Es el movimiento que se realiza al producirse una variación alrededor del eje longitudinal del UAV. El actuador que se utiliza para este movimiento son los alerones que están situados en las alas. Gracias a los alerones se produce el giro lateral del UAV. Esto ocurre cuando el alerón de una de las alas sube y el del otro ala baja. Así, cuando el alerón se ha flexionado hacia abajo produce un aumento de sustentación en su ala que provoca el ascenso de esta, mientras que el otro alerón produce en su ala una disminución de sustentación que motiva su descenso.
- Guiñada: Se produce cuando se gira alrededor del eje vertical. Este eje es perpendicular a los otros dos ejes. El actuador que se utiliza es el timón, situado en la parte trasera de la aeronave. Este movimiento permite corregir el rumbo de la aeronave girando alrededor del mismo de izquierda a derecha.

3.2. Cuadricóptero

Los cuadricópteros son UAVs que se elevan y desplazan gracias a sus cuatro motores que están instalados en un marco en forma de cruz. Su autonomía se basa en la energía eléctrica de una batería y utiliza sensores electrónicos para estabilizar el vehículo. Comparando con los helicópteros que no tiene cuatro motores la principal diferencia de los cuadricópteros son que con solo variar la velocidad de uno de sus motores ya se moverá, en vez de utilizar un sistema mecánico para mover el disco del rotor. La desventaja principal de los cuadricópteros es que son más ineficientes que los helicópteros convencionales, ya que necesitan mayor energía para sus cuatro rotores, pero son capaces de aumentar su potencia más rápido gracias a sus baterías tipo LiPo. Estas baterías de polímero de litio permiten una mayor densidad de energía y una alta tasa de descarga.

3.2.1. Movimientos

Como hemos dicho, la capacidad de vuelo y sustentación del cuadricóptero se basa en cuatro motores. Su estructura transversal es bastante más ligera que los motores. Cada hélice está conectado al motor a través de engranajes. Las hélices tienen forma de pala que logran que el aire fluya hacia abajo para conseguir elevarse. Gracias a esto, podremos decir que se trata de una estructura bastante rígida y que lo único que varían son las velocidades en las que giran las hélices. Para explicar los movimientos del cuadricóptero no es necesario hablar de los motores, ya que los movimientos están directamente relacionados con la velocidad de la hélices. Las hélices delanteras y traseras giran en sentido anti-horario, mientras que la izquierda y derecha giran en sentido horario. Esta configuración hace que no sea necesario un rotor de cola. En la Figura 4 se observa un modelo de la estructura de un cuadricóptero en estado hover, esto es, volando pero sin moverse. Esto se logra cuando todas las hélices giran a la misma velocidad.

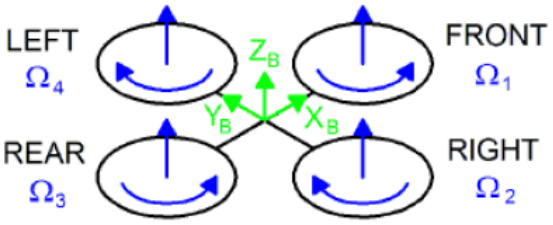

Figura 4: Estructura

En la Figura 4, el esquema de la estructura del cuadricóptero se presenta en negro, la velocidad angular de las hélices en azul y el marco fijo (el sistema de coordenadas) en verde. Además de la velocidad, también de color azul se representan la dirección de rotación y la velocidad en la que se desplazará. Esta última velocidad siempre apuntará hacia arriba. En la imagen anterior todas las hélices giran a la misma velocidad (Ω_H) para lograr contrarrestar la aceleración debida a la gravedad y mantenerse en el aire.

Los diferentes los movimientos del cuadricóptero son los siguientes:

- **Throttle**

La aceleración o deceleración para despegar o tomar tierra ocurrirá cuando se aumente o disminuya la velocidad de todas las hélices en la misma medida. En la Figura 5 aparece la estructura del cuadricóptero mientras se acelera.

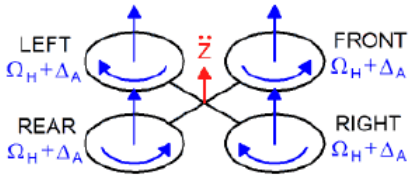

Figura 5: Throttle

En azul se especifica la velocidad de la hélices, que en este caso es igual a $\Omega_H + \Delta_A$ en cada hélice. Δ_A es una variable que representa el incremento respecto a la constante Ω_H .

- **Roll**

Mediante el aumento o disminución de la velocidad de la hélice izquierda y la disminución o el aumento de la velocidad de la hélice derecha el drone se moverá hacia la izquierda o derecha. Esto es, el drone rota alrededor del eje X_b haciendo que el cuadricóptero gire. El estado vertical del drone será el mismo que cuando está estable (hovering).

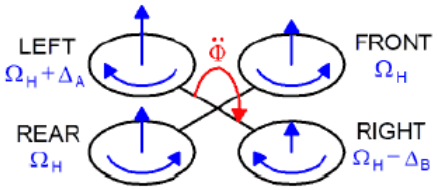

Figura 6: Roll

En la Figura 6 se representa el movimiento roll donde las variables Δ_A y Δ_B son elegidas para que la posición vertical no varíe. Se puede demostrar que para valores pequeños de Δ_A, Δ_A y Δ_B serán parecidos por lo que la inclinación del drone hacia cualquiera de los dos lados no será muy grande y el cambio de dirección no será brusco.

- **Pitch**

Este movimiento es bastante parecido al del Roll pero se consigue mediante el aumento o disminución de la velocidad de la hélice trasera y la disminución o el aumento de la hélice delantera. Esto hace que el drone se mueva hacia adelante o hacia atrás, ya que el movimiento es respecto al eje Y_b que hace el cuadricóptero gire. El estado vertical del drone será el mismo que cuando está estable (hovering) por lo que solo será un movimiento hacia adelante o hacia atrás.

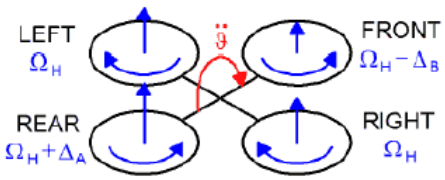

Figura 7: Pitch

Como en el caso de Roll las variables ΔA y ΔB no pueden ser muy grandes para que el movimiento no sea muy brusco y no se produzcan indeseados desplazamientos verticales.

- **Yaw**

Este movimiento se logra al incrementar o disminuir la velocidad de las hélices delanteras-traseras y disminuyendo o incrementado las de la pareja izquierda-derecha. Esto hace que el drone se desplazo respecto al eje Z_b. Esto es, el movimiento Yaw se consigue gracias a que las hélices izquierda y derecha giran en sentido horario y las hélices delantera y trasera en sentido anti horario. Por lo tanto, el helicóptero gira sobre si mismo alrededor de Zb. Como ocurre con los movimientos Pitch y Roll la posición vertical no varía.

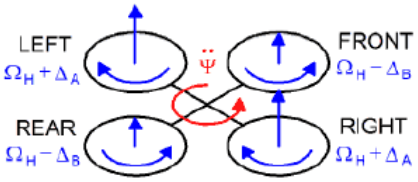

Figura 8: Yaw

Las variables Δ_A y Δ_B no serán muy grandes igual que ocurre con Pitch y Roll, ya que no ocurrirá ningún movimiento vertical.

La Figura 9 ilustra los diferentes movimientos del cuadricóptero según la potencia de cada rotor para una estructura en cruz:

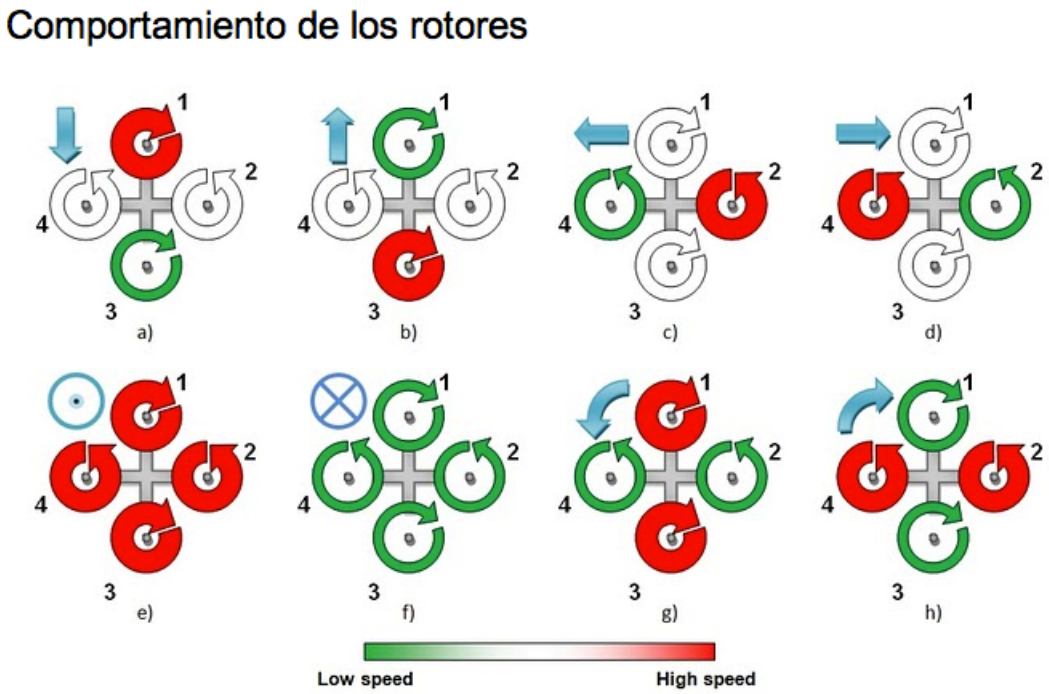

Figura 9: Rotores

Entre los diferentes cuadricópteros que nos podemos encontrar en el mercado están:

- Parrot AR-Drone

Figura 10: AR.Drone

- X-UFO

Figura 11: X-Ufo

- Hubsan X4

Figura 12: Hubsan X4

4. AR.Drone

El AR.Drone es un cuadricóptero no tripulado radio-controlado. Mediante cuatro motores eléctricos y un microprocesador es capaz de volar y con una serie de sensores puede captar lo que tiene alrededor y comunicarse con otros dispositivos. El AR.Drone además dispone de dos carcasas diferentes para cada tipo de vuelo que se quiera hacer: interior o exterior.

Figura 13: Drone

Figura 14: Desplazamiento lateral

Figura 15: Giro

Figura 16: Ganancia o perdida de altura

Entre esta serie de sensores se encuentra dos cámaras y un conector Wi-Fi. Los diferentes sensores de movimiento se encuentran debajo del casco central. Con el conector Wi-Fi se puede controlar el AR.Drone desde otros dispositivos, como PCs, dispositivos móviles etc. Para poder conectarse al drone este crea un red Wi-Fi llamada ardrone_xxx, por lo que el usuario solo tiene que conectarse a esta red. Al dispositivo del usuario el Servidor DHCP del AR.Drone le otorgará una dirección IP, que será la dirección IP propia del drone más un número del 1 al 4.

Mediante el envío de comandos AT se realiza el control y la configuración del drone. Estos comandos se transfieren por el puerto UDP 5556 con una latencia de transmisión habitual de 30 comandos por segundo. La información del drone (la posición, velocidad, altura etc.), llamado *navdata*, son enviados por el avión al cliente por el puerto UDP 5554. Se transfieren con dos frecuencias: 15 veces por segundo en modo Demo y 200 veces por segundo en modo Debug, que sería cuando se crean juegos de Realidad Aumentada.

La secuencia de vídeo es enviada por el AR.Drone al dispositivo del cliente al puerto 5555 y las imágenes pueden ser decodificadas utilizando el códec incluido en el SDK.

Para la transmisión de datos críticos el drone utiliza un canal de comunicación llamado puerto de control. Este canal alternativo se utiliza para recuperar datos de configuración y para reconocer información importante. Estos datos van al puerto de Control TCP 5559.

El sensor de ultrasonidos que viene implementado en la parte inferior del AR.Drone permite determinar la altitud a la cual se encuentra el drone en cada momento. Este sensor emite de forma constante en dirección a la superficie del suelo y dependiendo de la latencia con la que reciba las señales en forma de eco realizará una estimación de la altitud a la cual se encuentra

En cuanto a los sensores basados en visión, se emplean dos cámaras para llevar a cabo la captura de imágenes en tiempo real. Las cámaras son las siguientes:

- Cámara frontal: Con una dimensión de imagen de 640x360 píxeles de tamaño en formato panorámico, esta cámara tiene capacidad de tomar imágenes en alta definición (720 HD). Es la cámara principal del AR.Drone para capturar las imágenes del entorno. Se sitúa en la parte frontal de UAV y se comunica con el procesador de a bordo utilizando la cable de comunicaciones. Es la única cámara capaz de adquirir el campo visual completo del UAV.

- Cámara vertical: Se sitúa en la parte inferior del drone y se utiliza normalmente para lograr una estabilidad al aparato cuando se encuentra en el aire sin realizar ningún tipo de maniobra. La calidad de la imágenes es inferior a la frontal, pero, como en la cámara frontal, el color es RGB.

El cuerpo del drone contiene una de las partes más importantes del dispositivo: la batería de litio que es posible recargar y que permite una autonomía de vuelo de unos 15 minutos. Por otro lado el escudo del armazón protege al drone de los impactos que pueda sufrir durante el vuelo.

4.1. Características electrónicas

En cuanto a las características electrónicas, las tecnologías usadas en el AR.Drone permite al usuario tener un control de precisión extremo y características de estabilización automáticas. Entre tecnologías del AR.Drone se encuentran las siguientes:

- Procesador de 32 bits ARM Cortex A8 @1Ghz
- Linux 2.6.32 -Memoria RAM DDR2 de 1Gbit @200MHz
- Wi-Fi b,g,n
- Conector USB 2.0
- Acelerómetro de 3 ejes con precisión de +-50mg
- Giroscopio de 3 ejes con precisión de 2000º/segundo
- Magnetómetro de 3 ejes con precisión de 6º
- Sensor de presión con precisión de +/- 10 Pa
- Sensor de ultrasonido para medir la altura sobre el terreno

4.2. Actuadores

Los actuadores del AR.Drone coinciden con las 4 hélices motoras y que dependiendo de la velocidad de rotación de cada una de ella se logra desarrollar diferentes maniobras con el drone. También, se puede enviar al AR.Drone instrucciones de navegación predefinidas que al ser ejecutadas se logra, por ejemplo, despegar hasta una altitud predefinida o aterrizar sobre la superficie del entorno. Como se ha comentado anteriormente, en el caso de la navegación en el interior de un edificio se utiliza normalmente un elemento de protección que se coloca sobre la cruceta del AR.Drone para prevenir posibles daños en los motores al colisionar contra algún objeto.

Figura 17: Actuadores del AR.Drone

4.3. SDK

El SDK (kit de desarrollo de software) es un conjunto de herramientas o librerías que permitirá escribir nuestras propias aplicaciones para controlar el drone. Esto lo podremos hacer desde cualquier ordenador personal, iPhone o teléfono móvil Android. La estructura del SDK es la siguiente:

- ARDroneAPI.dox: Se utiliza para generar documentación.
- ARDroneLib: La biblioteca del AR.Drone (la comunicación, los códecs etc.)
- ControlEngine: Archivos para el IPhone
- Docs: La documentación se genera en esta carpeta
- Example: Esta carpeta contiene el código de las diferentes demostraciones.

Esta librería dispone de diferentes capas de abstracción que se encargan de gestionar desde el nivel bajo, donde se sitúa el hardware del drone, hasta el nivel más alto donde las aplicaciones podrán ejecutar las funciones que comunicarán con los sensores y actuadores.

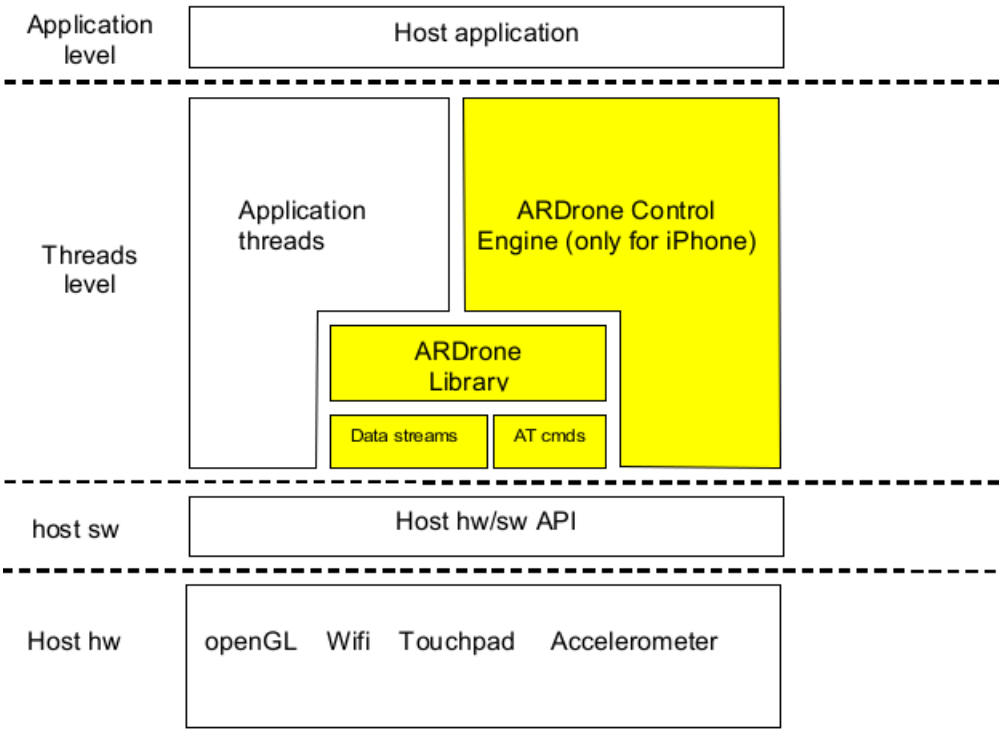

Figura 18: Niveles de abstracción del SDK

4.4. Entorno y puesta en marcha

Para poner en marcha los drones y entender su funcionamiento ejecutaremos los ejemplos del SDK que facilitarán la creación de diferentes programas más adelante.

Los detalles del software y hardware para el funcionamiento del SDK:

- SDK 2.0.1

- Ubuntu 13.10 32 bits
- VirtualBox 4.3.6
- Logitech(R) Precision(TM) Gamepad [0004]
- AR.Drone 2.0

Para utilizar las diferentes demos utilizaremos VirtualBox para simular el Sistema Operativo Ubuntu de 32 bits, que es el necesario para poder ejecutar estos ejemplos. Para ello debemos de montar una imagen .iso en el VirtualBox como en la Figura 19.

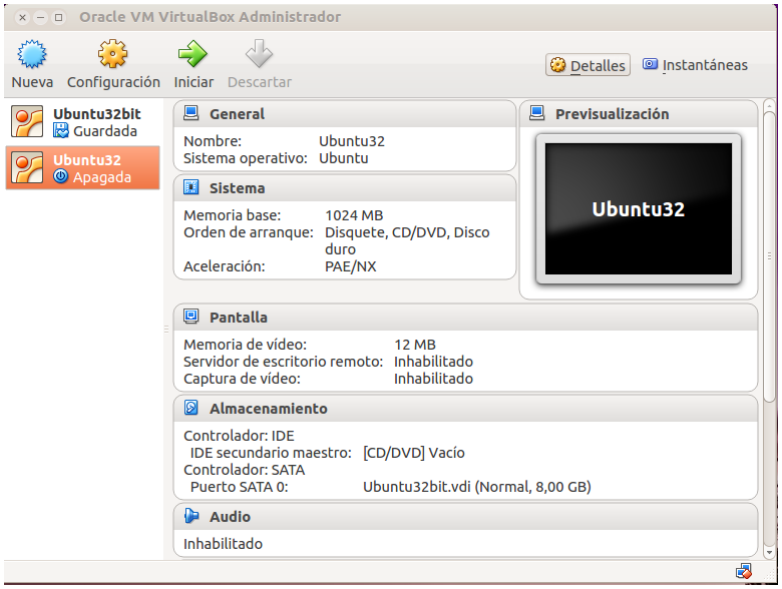

Figura 19: Imagen .iso

Para que funcione el SDK correctamente se deben instalar las siguientes librerías:

- sudo apt-get install libgtk2.0-dev
- sudo apt-get install libsdl1.2-dev
- sudo apt-get install libiw-dev
- sudo apt-get install libxml2-dev
- sudo apt-get install libudev-dev
- sudo apt-get install libncurses5-dev libncursesw5-dev

A continuación,

- En "ARDronelib/Soft/Build/custom.makefile" escribiremos "USE_LINUX=yes" para definir que utilizaremos Linux.
- En "ARDroneLib/Soft/Build/" escribiremos *make* para crear los ejecutables.

Después, en diferentes GENERIC_LIBS habrá que añadir lo siguiente :

- En "ARDronelib/Examples/Linux/video_demo/Build/Makefile":

```
GENERIC_LIBS=-lpc_ardrone -lrt -lgtk-x11-2.0 -lcairo -lgobject-2.0 -lgdk-x11-2.0 –lm
```

- En "ARDronelib/Examples/Linux/Navigation/Makefile" habrá que añadir *–lm* a GENERIC_LIBS.

Los ejecutables se crearán ejecutando *make* en ARDronelib/Examples/Linux y estarán disponibles en color verde.

```
jon@jon-VirtualBox:~/Escritorio/ARDrone_SDK_2_0_1/Examples/Linux/Build/Release$
ls -l
total 941476
-rw-r--r-- 1 jon jon      3446 dic 10 18:52 ardrone2.xml
-rwxr-xr-x 1 jon jon   2598040 dic  4 12:41 ardrone_navigation
-rw-r--r-- 1 jon jon      3446 dic 10 19:04 ardrone.xml
drwxr-xr-x 2 jon jon      4096 dic  4 12:41 common
drwxr-xr-x 2 jon jon      4096 dic 10 18:33 flight_20131210_183300
drwxr-xr-x 2 jon jon      4096 dic 10 18:34 flight_20131210_183426
drwxr-xr-x 2 jon jon      4096 dic 10 18:34 flight_20131210_183435
drwxr-xr-x 2 jon jon      4096 dic 10 18:34 flight_20131210_183449
drwxr-xr-x 2 jon jon      4096 dic 10 18:36 flight_20131210_183606
drwxr-xr-x 2 jon jon      4096 dic 10 18:38 flight_20131210_183828
drwxr-xr-x 2 jon jon      4096 dic 10 18:41 flight_20131210_184152
drwxr-xr-x 2 jon jon      4096 dic 10 18:42 flight_20131210_184243
drwxr-xr-x 2 jon jon      4096 dic 10 18:43 flight_20131210_184300
drwxr-xr-x 2 jon jon      4096 dic 10 18:43 flight_20131210_184322
drwxr-xr-x 2 jon jon      4096 dic 10 18:45 flight_20131210_184457
drwxr-xr-x 2 jon jon      4096 dic 10 19:09 flight_20131210_190859
drwxr-xr-x 2 jon jon      4096 dic 10 19:10 flight_20131210_190941
drwxr-xr-x 2 jon jon      4096 dic 10 19:13 flight_20131210_191301
-rwxr-xr-x 1 jon jon    412268 dic  4 12:41 linux_sdk_demo
-rwxr-xr-x 1 jon jon   2365304 dic  4 12:42 linux_video_demo
-rw-r--r-- 1 jon jon         0 dic 10 17:17 mesures_20131210_171719.txt
-rw-r--r-- 1 jon jon 136668342 dic 10 17:28 mesures_20131210_172239.txt
```

Figura 20: Ejecutables

En cuanto al Wi-Fi, en nuestro caso se utilizarán dos IPs diferentes:

- 192.168.1.1 : La dirección del drone.
- 192.168.1.2 : La dirección del portátil.

Para conectarnos al drone solo hará falta buscar el Wi-Fi correspondiente al AR.Drone como en la Figura 21.

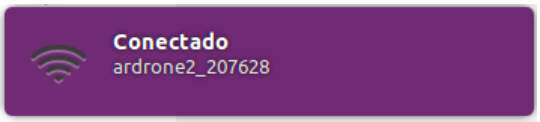

Figura 21: Wi-Fi

Mediante un ping probaremos si hemos logrado conectarnos.

```
jon@jon-VirtualBox:~/Escritorio/ARDrone_SDK_2_0_1/Examples/Linux/Build/Release$
ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=8.50 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.000 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=2.75 ms
64 bytes from 192.168.1.1: icmp_seq=4 ttl=64 time=3.73 ms
64 bytes from 192.168.1.1: icmp_seq=5 ttl=64 time=2.84 ms
```

Figura 22: Ping

4.5. Demos

4.5.1. Linux_sdk_demo

Con esta demo lograremos la información del drone, como su posición y orientación, pero el drone no volará, para tomar datos tendremos que moverlo con nuestras manos.

```
Starting thread ardrone_control
PA : MEMORY SPACE ALLOWED : 40 MB
Thread navdata_update in progress...
Start thread thread_academy_download

   Video stage thread initialisation

Start thread thread_academy_upload
Start thread thread_academy
Academy download stage resumed
Academy download stage paused
Timeout when reading navdatas - resending a navdata request on port 5554
Academy download stage resumed
====================monstrations ====================
Sending default CAT_SESSION settings==================
====================monstrations ====================
Navdata for flight demonstrations ====================
Control state : 0 mVta] 0.000  [Phi] 0.000  [Psi] 0.000
Control state : 0 mVta] 0.000  [Phi] 0.000  [Psi] 0.000
Battery level : 66 mVa] 0.000  [Phi] 0.000  [Psi] 0.000
Orientation   : [Theta] 1648.000  [Phi] -3284.000  [Psi] 153408.000
Altitude      : 0vX] 0.000  [vY] 0.000  [vZPsi] 0.000
Speed         : [vX] 1648.000  [vY] -3284.000  [vZPsi] 153408.000
```

Figura 23: Linux_sdk_demo

4.5.2. Linux_video_demo

Con esta demo se podrá probar la cámara delantera del AR.Drone pero no se controlará el drone y como en la anterior demo lo tendremos que mover con la mano.

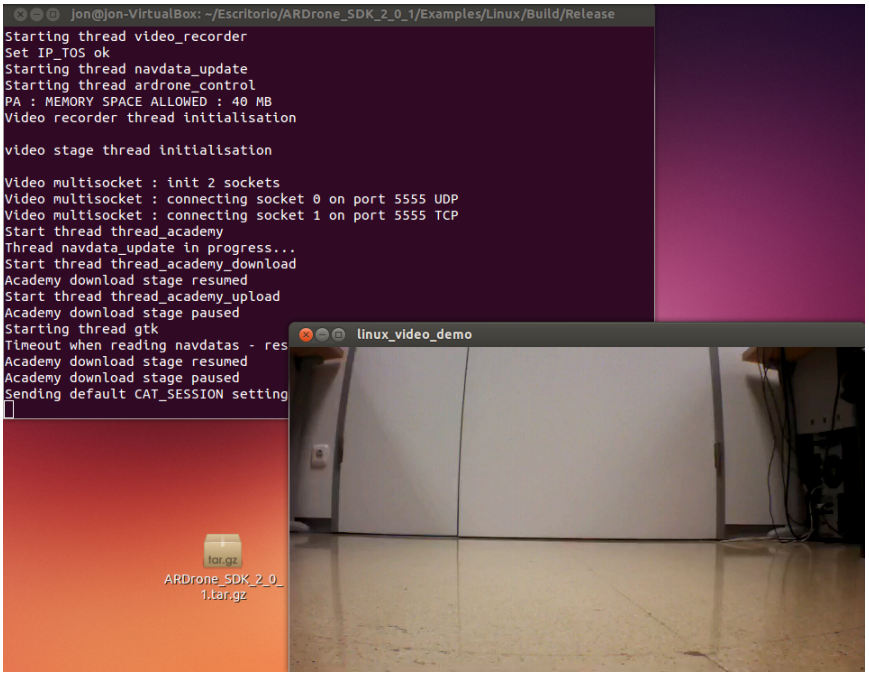

Figura 24: Linux_sdk_demo

4.5.3. Ardrone_navigation

Con esta demo lograremos controlar el AR.Drone con el mando Logitech. En el archivo ardrone.xml aparecerán las controles del Logitech con el que controlaremos el drone:

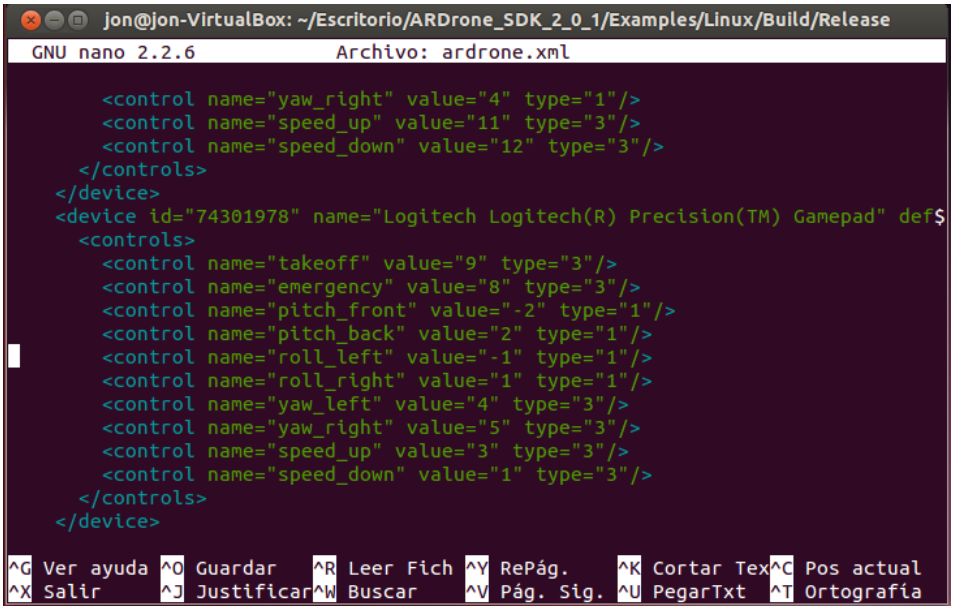

Figura 25: Controles Logitech

También conseguiremos tener en pantalla la información del drone como se puede ver en la figura 22:

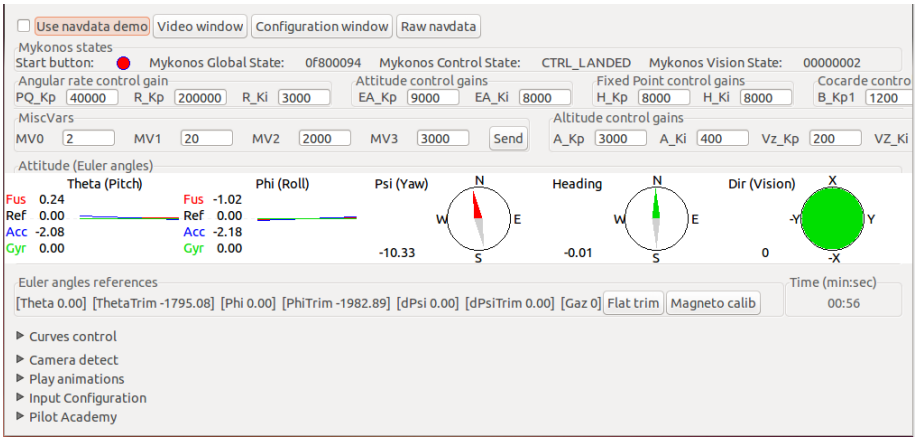

Figura 26: Ardrone_navigation

Pulsando el botón de "Video window" lograremos tener en pantalla la visión de la cámara del drone.

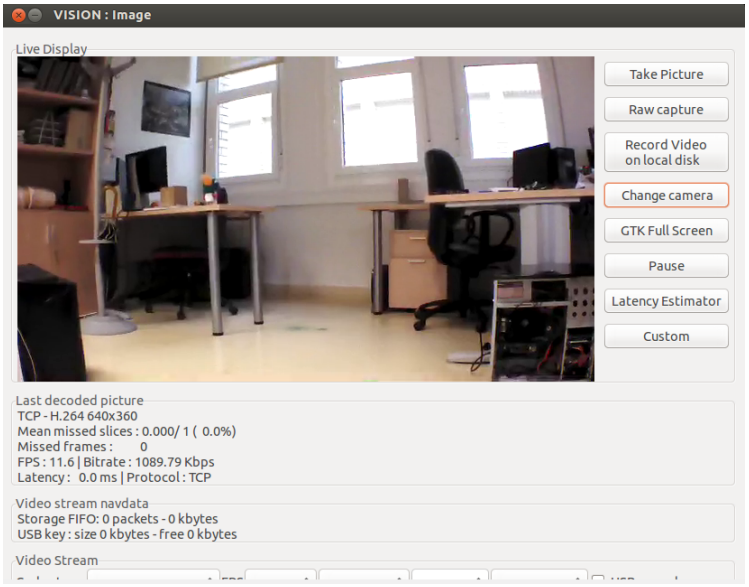

Figura 27: Video window

5. Multi AR.Drone

Nuestra intención es la de conseguir el vuelo y control de más de un drone a la vez con un solo ordenador. Para ello hemos utilizado un cliente Node.js para controlar el AR.Drone.

5.1. Node.js

Basado en un lenguaje de programación Javascript Node.js es un entorno de programación. Trabaja del lado del servidor que cambia la noción de cómo debería trabajar este. Su meta es permitirnos crear aplicaciones altamente escalables y escribir código para poder manejar decenas de miles de conexiones simultáneas en una sola máquina física. Gracias a su gestor de paquetes *npm* se pueden instalar y administrar los módulos de Node que se quieran utilizar. Esto módulos son creados por los usuarios de la comunidad de desarrolladores de Node.js y permiten extender la funcionalidad de un objeto mediante un prototipo. Existen cientos de módulos, que trabajan como paquetes o librerías, que pueden servir de plantillas, APIs, bases de datos...

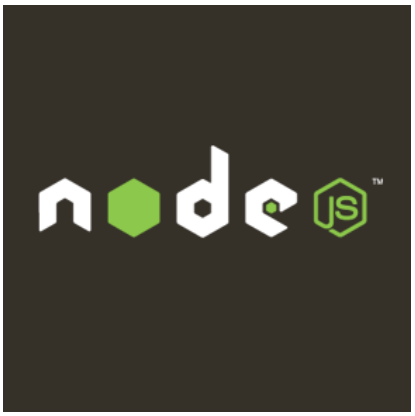

Figura 28: Node.js

5.2. Node ar-drone

Node ar-drone es un módulo de Node.js que nos permitirá controlar diferentes AR.Drones a la vez o por separado. Este módulo expone una API de cliente de alto nivel que trata de soportar todas las características del drone mientras intenta hacerlos más fáciles de utilizar. Para utilizar el Node ar-drone basta con instalarlo ejecutando el siguiente comando:

```
npm install ar-drone
```

5.3. Multidrone

Para conseguir controlar dos drones con el mismo ordenador deberemos observar la configuración de los mismos. Para ello utilizaremos la herramienta telnet:

```
telnet 192.168.1.1
```

Observaremos la configuración del drone desde dentro, donde encontraremos lo mismo que en un sistema operativo Linux:

```
# ls
bin dev factory home licenses proc sbin tmp usr
data etc firmware lib mnt root sys update var
```

Para lograr nuestro objetivo deberemos tener en cuenta la comunicación por Wi-Fi del drone. Este tiene tres tipo diferentes de comunicación:

- Master: Es el forma de comunicación que trae el drone por defecto. Mediante esta configuración nos podremos conectar con nuestro ordenador a la red Wi-Fi que creará el drone. Es la configuración que se utiliza normalmente en cualquier aplicación. El drone activa un servidor DHCP que le otorga una IP (192.168.1.1) y un SSID (ar_drone_XXXXXX).
- Ad-hoc: Parecido que el tipo Master pero el drone creará un red Wi-Fi tipo Ad-hoc.
- Managed: Esta configuración se utiliza para conectar nuestra infraestructura de drones a una red, esto es, para conectar el drone a la red que queramos.

```
WIFI_MODE=`grep wifi_mode /data/config.ini | awk -F "=" '{ gsub(/ */,"",$2);

case $WIFI_MODE in
0)
    WIFI_MODE=master
    ;;
1)
    WIFI_MODE=ad-hoc
    ;;
2)
    WIFI_MODE=managed
    ;;
*)
    WIFI_MODE=master
    ;;
esac
```

Figura 29: Modo Wi-Fi

Nosotros elegiremos el modo Ad-hoc para que podamos conectarnos con nuestro ordenador a la red de los drones.

Para ello tendremos que cambiar el fichero /data/config.ini:

```
vi /data/config.ini
```

Aparte de determinar que tipo de Wi-Fi queremos crear deberemos cambiar la SSID de la red, para que así los dos drones que utilizaremos tengan el mismo SSID y conectarnos desde el ordenador a una sola red.

```
motor4_supplier                 = 1.1
ardrone_name                    = My ARDrone
navdata_demo                    = TRUE
com_watchdog                    = 2
video_enable                    = TRUE
vision_enable                   = TRUE
vbat_min                        = 9000

[network]
ssid_single_player              = multiardrone
ssid_multi_player               = ardrone2_v2.3.3
wifi_mode                       = 1
owner_mac                       = 00:00:00:00:00:00

[control]
accs_offset                     = { -4.4110474e+03 3.8844153e+03 4.0100059e+03 }
accs_gains                      = {  1.9695557e+00 1.2452380e-01 3.6147125e-02  1
```

Figura 30: Network del drone

Después, deberemos asignar a los drones unas IPs diferentes para que no pueda haber conflicto. Por ello, cambiaremos "bin/wifi_setup.sh" para que cada drone cree un IP estática diferente, por ejemplo, daremos a un drone la ip 192.168.1.100 y a otro 192.168.1.200. Será la variable *PROBE* la que deberemos de cambiar que es la determina el ultimo byte de la dirección IP de nuestro drone.

```
OK=0
BASE_ADRESS=192.168.1.
PROBE=100
```

Figura 31: IP del drone

A continuación, comprobaremos que nuestro ordenador se puede conectar al red de los drones y lo probaremos haciendo un PING a ambos drones como se ve en la Figura 33.

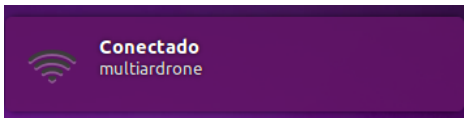

Figura 32: Wi-Fi de los drones

```
jon@jon-VirtualBox: ~
jon@jon-VirtualBox:~$ ping 192.168.1.100
PING 192.168.1.100 (192.168.1.100) 56(84) bytes of data.
64 bytes from 192.168.1.100: icmp_seq=1 ttl=64 time=14.5 ms
64 bytes from 192.168.1.100: icmp_seq=2 ttl=64 time=7.06 ms
64 bytes from 192.168.1.100: icmp_seq=3 ttl=64 time=2.86 ms
^C
--- 192.168.1.100 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 2.866/8.167/14.575/4.844 ms
jon@jon-VirtualBox:~$ ping 192.168.1.200
PING 192.168.1.200 (192.168.1.200) 56(84) bytes of data.
64 bytes from 192.168.1.200: icmp_seq=1 ttl=64 time=4.51 ms
64 bytes from 192.168.1.200: icmp_seq=2 ttl=64 time=4.99 ms
64 bytes from 192.168.1.200: icmp_seq=3 ttl=64 time=1.41 ms
^C
--- 192.168.1.200 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 1.418/3.644/4.999/1.586 ms
```

Figura 33: Ping a los drones

Para poder ejecutar el código cambiaremos la dirección IP de nuestro ordenador a una dirección estática de la red de los drone, como puede ser 192.168.1.101.

```
jon@jon-VirtualBox:~$ sudo ifconfig eth0 192.168.1.101 up
```

Figura 34: IP del PC

5.4. Multidrone vuelo autónomo

Con la intención de lograr un vuelo autónomo de los drones utilizaremos la librería antes mencionada "node ar-drone" con la cual podremos programar el vuelo con la duración que deseemos.

Para ello utilizaremos los diferentes comandos que nos proporciona la librería que los usaremos para dirigir los drones hacia donde nosotros queramos. Entre los distintos comandos están los siguientes:

- client.takeoff() : el drone despegará del suelo.
- client.stop(): todos los comandos se ponen a 0.
- client.land(): el drone toma tierra.
- client.up(speed) / client.down(speed) : el drone ganará altitud o la perderá. El valor enviado será entre 0 y 1.
- client.left(speed) / client.right(speed): Movimiento horizontal hacia la izquierda o la derecha. El valor enviado será entre 0 y 1.
- client.clockwise(speed): El drone girará según la velocidad deseada. El valor enviado será entre 0 y 1.
- client.front(speed) / client.back(speed): tomando la cámara horizontal como referencia, el drone se moverá hacia delante o hacia atrás. El valor enviado será entre 0 y 1.

Aparte de utilizar los comandos para desplazar los drones también tendremos que crear un objecto cliente:

- arDrone.createClient(IP): como opción le podremos mandar la IP del drone que queremos controlar. Si no le enviamos ninguna IP cogerá la IP de defecto(192.168.1.1).

Antes de crear el cliente tendremos que definir que módulo utilizaremos, en nuestro caso será "node ar-drone":

- arDrone = require('ar-drone')

Con todo esto el programa para un vuelo autónomo de un drone quedaría como en la Figura 35.

```
var arDrone = require('ar-drone');
var drone1 = arDrone.createClient();

drone1.takeoff();
drone1.after(5000, function(){
        this.stop();
        this.land();
});
```

Figura 35: Vuelo Autónomo

Donde mandaremos al drone despegar y después de 5000ms le mandaremos pararse y que tome tierra.

Pero si queremos manejar dos drones con un solo programa deberemos de crear dos clientes independientes como en la Figura 36

```
var arDrone = require('ar-drone');
var drone1 = arDrone.createClient({ip : '192.168.1.100'});
var drone2 = arDrone.createClient({ip : '192.168.1.200'});

drone1.takeoff();
drone1.after(5000, function(){
        this.stop();
        this.land();
});

drone2.takeoff();
drone2.after(5000, function(){
        this.stop();
        this.land();
});
```

Figura 36: Vuelo autónomo con dos clientes

5.4.1. Navdata

Al igual que le pasamos comandos al drone también podemos conseguir que el drone nos mande los datos, en la estructura denominada *navdata*, para saber en qué situación se encuentra. Entre estos datos están la altura, la posición (yaw, roll, pitch), el estado de la batería etc. El drone enviará esta información constantemente y la podremos utilizar para programar el dron. En la Figura 37 vemos como lograr que se imprima el *navdata* enviado por el drone.

```
var arDrone = require ('ar-drone');
var drone1 = arDrone.createClient({ip: '192.168.1.100'});

drone1.on('navdata', function(d){
        console.log(d);



});
```

Figura 37: Navdata

Por otro lado, en la Figura 38 vemos como utilizar los datos que nos envía el drone como variables. Por ejemplo, observamos como utilizaríamos la variable de altitud.

```
drone1.on('navdata', function(d){

        if(d.demo){
                if (d.demo.altitude < 1){
                        drone1.stop();
                        drone1.land();}
        }


});
```

Figura 38: Variable altitud

Tendremos que activar las partes que queremos utilizar como variables, ya que el drone sino no nos enviará la información de esa parte, si no que nos enviará todos los datos. Las diferentes partes o paquetes del navdata son los siguientes:

- demo
- time
- rawMeasures
- physMeasures
- gyrosOffsets
- eulerAngles
- references
- trims
- rcReferences
- pwm
- altitude
- visionRaw
- visionOf vision
- visionPerf
- trackersSend
- visionDetect
- watchdog
- adcDataFrame
- videoStream
- games
- pressureRaw
- magneto
- windSpeed
- kalmanPressure
- hdvideoStream
- wifi
- gps

Para que el drone nos mande estos datos tendremos que escribir el programa de la Figura 39:

```
var arDrone = require ('ar-drone')
 , arDroneConstants = require('ar-drone/lib/constants')
 ;

function navdata_option_mask(c){
        return 1 << c;
}

var navdata_options = (

        navdata_option_mask(arDroneConstants.options.DEMO)
        |  navdata_option_mask(arDroneConstants.options.VISION_DETECT)
        |  navdata_option_mask(arDroneConstants.options.MAGNETO)
        |  navdata_option_mask(arDroneConstants.options.WIFI)
        |  navdata_option_mask(arDroneConstants.options.VISION)
);

var drone1 = arDrone.createClient({ip: '192.168.1.100'});
drone1.config('general:navdata_demo', true);
drone1.config('general:navdata_options', navdata_options);
```

Figura 39: Programa para recibir datos

Donde configuraremos el drone para que nos envíe los paquetes navdata que elijamos en la variable navdata_options. En este ejemplo le mandaremos al drone que nos envíe los paquetes demo, vision_detect, magneto, wifi y vision.

5.5. Multidrone vuelo controlado mediante teclado

Para conseguir un vuelo controlado de los drones utilizaremos el teclado del ordenador para guiarlos. La librería "keypress" de Node.js nos ayudará a controlar los drones.

5.5.1. Keypress

Con el módulo Keypress lograremos escuchar la teclas presionadas. Este módulo se basa en que escucha los eventos "keypress" en donde utiliza el proceso "process.stdin" para escuchar la teclas que se pulsan.

Para que el programa escuche las teclas tendremos que definir el módulo que estamos utilizando y tendremos que llamar al proceso stdin:

```
var keypress = require('keypress');
keypress(process.stdin);
```

Figura 40: Keypress

Después, dentro del proceso stdin definiremos una función en donde describiremos diferentes casos en los cuales al ser pulsados ciertos botones el drone se moverá según el comando escrito. Los comandos para el desplazamiento del drone serán los que se definen en el módulo "node ar-drone". Por ejemplo para despegar utilizaremos la barra espaciadora:

```
process.stdin.on('keypress', function(ch, key){

if(key && key.name == 'space' ){
console.log('takeoff');
drone1.takeoff();
drone2.takeoff();
}
});
```

Figura 41: Programa de Keypress

Para terminar solo deberemos definir una tecla para que se termine la comunicación entre los drones y el ordenador como se ve en la Figura 42.

```
if(key && key.name == 'c'){
console.log('Saliendo');
process.stdin.pause();

drone1.stop();
drone1.land();
drone2.stop();
drone2.land();

drone1._udpControl.close();
drone2._udpControl.close();

}
```

Figura 42: Programa para terminar comunicación

6. Transporte de cables o cuerdas

Con el objetivo de transportar una cuerda tendremos que definir los elementos que llevarán a cabo el transporte:

- Vehículo: instrumento que permitirá el traslado de la mercancía, en nuestro caso utilizaremos varios AR.Drones.
- Mercancía: objeto que los vehículos transportarán: nosotros usaremos diferentes cuerdas.
- Infraestructura: lugar donde se ejecutará la actividad. Nuestro transporte se producirá en un entorno cerrado.
- Operador de transporte: se ocupa de guiar los vehículos y en este caso será un computador.

Cuando hablamos de transportar objetos tenemos que tener en cuenta el peso de las mercancía y efecto que este le producirá a los vehículos que lo transporten.

En nuestro caso, utilizando drones como vehículos de transporte tendremos que tener en cuenta que la fuerza vertical deberá de ser igual o mayor que la fuerza ejercida por el peso del vehículo y la mercancía para que permanezca en vuelo. A diferencia de los aviones, los helicópteros no necesitan impulsarse hacia delante para crear la sustentación que les permita crear la fuerza vertical, sino que gracias a los rotores conseguirán mantenerse en el aire. Aparte de eso, los rotores permiten al AR.Drone no tener que utilizar una pista de despegue para ganar velocidad.

En la Figura 43 se detalla el esquema de los movimientos de los cuadricópteros, donde se encuentran las siguientes variables:

- u_1: suma de los empujes de todos los motores.
- Th_1: Empuje generado por el motor delantero.
- Th_2: Empuje generado por el motor trasero.
- Th_3: Empuje generado por el motor derecho.
- Th_4: Empuje generado por el motor izquierdo.
- m: Masa del cuadricóptero.
- g: La aceleración de la gravedad
- l : Longitud media del cuadricóptero.
- x, y ,z : Las tres posiciones o coordenadas.
- ϑ, Φ, ψ : Los tres ángulos de Euler que representan a pitch, roll y yaw.

La suma de los empujes que producirán cada uno de los motores tendrá que ser mayor o igual que la multiplicación entre la aceleración de la gravedad y la masa del drone para que este vuele.

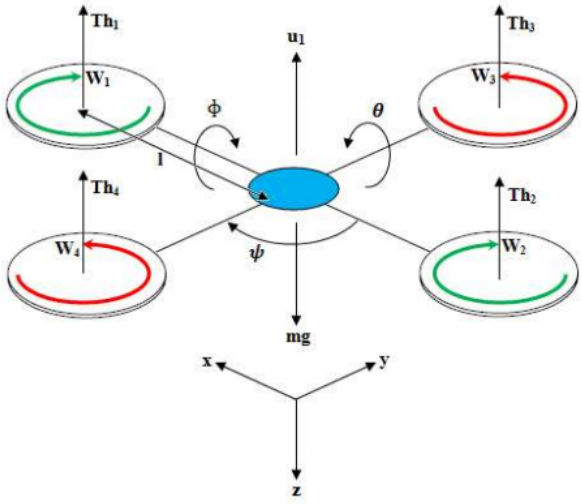

Figura 43: Esquema de un cuadricóptero

Por otra parte, tendremos que tener en cuenta la distancia que hay entre los drones. En nuestro caso, la distancia entre los drones será relativamente corta, ya que el objetivo será la de transportar cuerdas no muy pesadas. Transportaremos estas cuerdas debido a que si utilizamos cuerdas muy pesadas los AR.Drones no tendrán la fuerza suficiente para transportarlos. Aparte de esto, no nos interesa que las cuerdas se arrastren, ya que esto podría suponer un peligro si se encuentran con un obstáculo a ras del suelo.

Debido a la corta distancia, tendremos que tener el cuenta que el viento producido por cada drone afectará a los drones contiguos. Como hemos comentado más arriba el comportamiento de los drones dependerá del peso de su mercancía, por lo que los drones tenderán a moverse por si solos en algunas ocasiones debido al peso o al aire producido por los otros drones.

Para estabilizar la posición de los drones, programaremos los drones para que recuperen su posición. Para ello tendremos en cuenta los ángulos pitch y roll, que serán los que cambiarán cuando el drone se desplace.

Por lo tanto, fijamos un intervalo de valores [-2,2] entre los cuales el movimiento del drone no es apreciable o no es tan exagerado como para que se desplace. Cuando el valor de pitch o roll esté dentro de este intervalo, consideramos que el drone no sufrirá casi variaciones. Tendremos en cuenta los valores de pitch y de roll en el caso que queramos que se mantengan los drones en estado hover, esto es, flotando en el aire pero sin desplazamientos. En el caso de que queramos desplazarnos hacia delante o hacia los lados tendremos que tener en cuenta o no los ángulos en cada caso.

Los cambios efectuados en los ángulos o en las velocidades en su defecto se detallarán en las pruebas, ya que variarán según cada situación y objetivo.

7. Preparación y configuración del entorno experimental

Antes de ejecutar las pruebas, tendremos que montar un entorno que nos servirá para probar el sistema creado y para observar el comportamiento de los drones al ejecutar los diferentes programas. También detallaremos la configuración de un nodo operativo que en nuestro caso será un computador que ejecutará los programas necesarios para el control del AR.Drone a lo largo de todos los experimentos.

El entorno que utilizaremos para algunos experimentos será un entorno cerrado, donde tendremos un nodo operativo que ejecutará los programas que controles los drones. Aparte de este nodo, tendremos los diferentes drones que estarán colocados a una distancia exacta entre cada uno de ellos, ya que, dependiendo de la distancia entre ellos el aire que produce cada hélice de cada drone afectará de diferente manera al otro drone. Como las distancias en un entorno cerrado son pequeñas no habrá problemas con el alcance de la señal Wi-Fi para la comunicación entre el computador y los dos drones. En otros experimentos en cambio, utilizaremos un entorno abierto pero la distancia entre los drones y el nodo operativo será pequeño, debido a esto, no habrá problemas.

Para transportar las cuerdas de los experimentos tendremos que crear un sistema de sujeción de las cuerdas. Para ello deberemos de añadir varios elementos a los AR.Drones. Entre los elementos utilizados están:

- Alambre: Servirá para atar la cuerda con el drone.
- Velcro: Una cinta velcro que servirá para ajustar al drone el alambre.
- Mercancía: Diferentes cuerdas que los drones sean capaces de transportar.

7.1. Nodo operativo

El computador que actuará de nodo operativo es encargado de ejecutar los programar para la navegación de los drones. Las características técnicas del nodo operativo son :

- Hardware:
  - Intel® Core™2 Duo CPU T6400 @ 2.00GHz × 2 , Memoria: 3,8 GiB
  - Controlador: Gallium 0.4 on AMD RV620
- Software:
  - Ubuntu 13.10, AR.Drone SDK 2.0
  - Módulo node-ardrone
  - Módulo keypress

El computador deberá de ser capaz de transmitir datos a través de una red inalámbrica Wi-Fi, para su comunicación remota con los drones como primer paso para la detección y enlazamiento con los mismo, en donde los AR.Drones proporcionarán dos direcciones IP ad-hoc al nodo operativo para el posterior intercambio de mensajes y de información entre ambas partes, como puede ser la información sensorial y las señales de control.

7.2. Cuadro de mando

Utilizaremos un cuadro de mando para obtener información en tiempo real de los valores de los sensores, para así poder saber los resultados de los experimentos. Gracias a este cuadro de mando tendremos la comodidad de poder centrarnos en un único punto toda la información de los sensores para su tratamiento y para poder realizar un análisis de los resultados. El cuadro de mando se basará en un cuadrante de trazabilidad, que llevará a cabo un registro de todas las iteraciones k ejecutadas por parte del bucle de control de la arquitectura, donde se registrarán los valores correspondientes a las señales sensoriales proporcionadas por los AR.Drones. Se registrará una traza por cada iteración k que será cada navdata recibido por los drones. La estructura y formato de cada una de las trazas serán:

- Drone: número o identificador del drone
- {leftRight, frontBack}: roll y pitch
- Altitude: altura en el que se encuentra el drone
- k : número de la iteración

8. Experimentos y resultados

En pruebas iniciales[1] comprobamos la estabilidad de los drones con una cuerda colgando. Como era demasiado larga no se producía el efecto buscado de transporte, ya que la cuerda no se levantaba al aire por completo . Después, utilizamos otras cuerdas más cortas con las que sí logramos el efecto de transporte y que detallamos los experimentos hechos con ellos a continuación:

8.1. Experimento 1

El objetivo de este experimento es observar si el sistema creado para el transporte de una cable o cuerda es capaz de hacerlo y ver los resultados de este experimento.

En este caso, esta prueba se hará para saber si se logra la estabilización de los drones después de despegar cargados con la cuerda.

8.1.1. Ejecución

Para esta prueba utilizaremos dos cuerdas de distinto peso y largura.

- Cuerda 1: Cuerda ligera con 250 cm de largura

Figura 44: Estado inicial Cuerda 1

- Cuerda 2: Cuerda algo más pesada que la Cuerda 1 pero de 150 cm de largura

[1] https://www.youtube.com/watch?v=U3fJkq7tvNg

Figura 45: Estado inicial Cuerda 2

La distancia entre los drones será la máxima que nos permita cada cuerda. Por otra parte, la duración de los vuelos serán de alrededor de 10 segundos que nos permitirán conseguir más de 200 trazas que después las analizaremos.

Figura 46: Vuelo Cuerda 1

Figura 47: Vuelo Cuerda 2

Debido a que a los drones les afecta el viento producido por el otro drone y por el peso de las cuerdas su posición se verá afectada por lo tanto deberemos de corregirlo. En esta prueba como el objetivo es la de lograr la estabilización sin avanzar deberemos de controlar los valores pitch y roll. Para corregir la posición, como hemos explicado antes, designaremos que los drones tengan que tener los valores pitch y roll entre -2 y 2. Si eso no se cumple, el drone deberá de corregir su posición aumentado la velocidad de los rotores que permitirán

girar para que los valores pitch y roll entren entre los valores -2 y 2. Para ello supondremos cuatro situaciones:

- Pitch menor que -2 : Supondrá que el drone se desplaza hacia adelante, por lo que el drone tendrá que volar hacia atrás.
- Pitch mayor que 2 : El drone se moverá hacia atrás y lo corregirá desplazándose hacia adelante.
- Roll menor que -2: El drone se moverá hacia la izquierda, por lo tanto volando hacia la derecha corregirá su posición
- Roll mayor que 2: El drone se desplaza a la derecha y volando hacia la izquierda corregirá su posición.

Estas correcciones se deberán hacer una vez que el drone coja cierta altura, ya que durante el despegue puede suceder cierta inestabilidad.

8.1.2. Resultados

Después de terminar la prueba las trazas conseguidas nos permitirán verificar el correcto funcionamiento del sistema. En los gráficos se recogen los valores de los ángulos pitch y roll y el valor de la altura por cada iteración. Al analizar los gráficos se debe de tener en cuenta que las primeras iteraciones corresponderán al tiempo en el que los drones despegan y que en esos momentos los valores de pitch y roll pueden ser muy superiores o inferiores de [-2,2], ya que debido al velcro y al alambre que sujetan la cuerda en la parte inferior del AR.Drone los valores variarán porque el drone se apoyará en la tierra pero también encima del alambre. Por lo tanto, a la hora de analizar los resultados habrá que mirar a partir de la iteración 70.

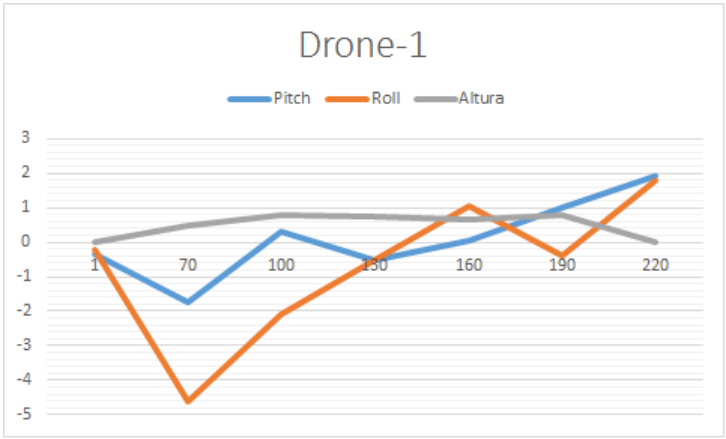

Figura 48: Gráfico del Drone1 con la Cuerda1

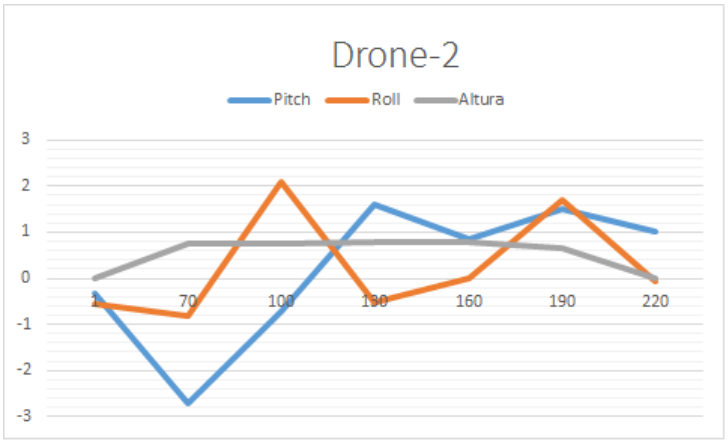

Figura 49: Gráfico del Drone2 con la Cuerda1

En las Figuras 48 y 49 se observan los resultados obtenidos con la Cuerda1 [2]. En ellas se puede ver que en el tiempo en los que los drones han estado volando con la cuerda ligera colgando no se han casi desestabilizado, ya que, en la mayoría del tiempo de vuelo los valores han estado dentro de los valores [-2,2] y si no estaban entre esos valores ha sido por poco tiempo debido a que han corregido su posición.

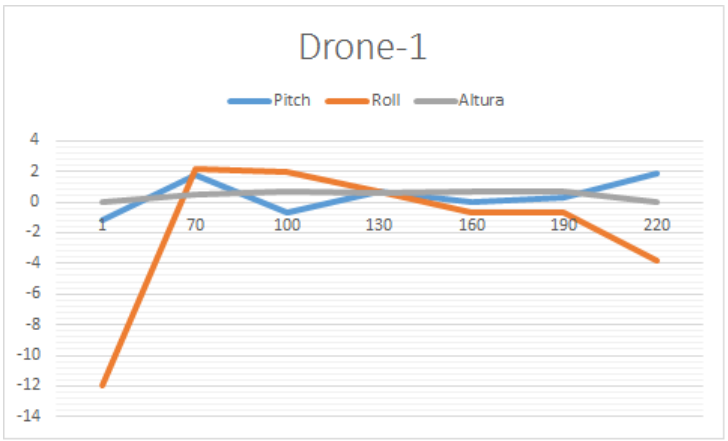

Figura 50: Gráfico del Drone1 con la Cuerda2

[2] https://www.youtube.com/watch?v=0nGUAFjYwes

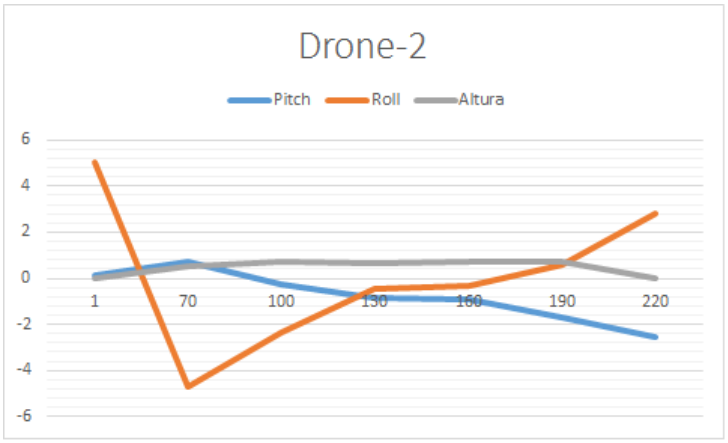

Figura 51: Gráfico del Drone2 con la Cuerda2

Los resultados de la prueba con la Cuerda2[3] se observan en las Figuras 50 y 51. Comparado con los resultados de la Cuerda1 se observa que esta vez a los drones les cuesta más estabilizarse, debido a que están más cerca uno del otro y el viento producido por cada uno de ellos les empuja hacia afuera. A pesar de esto, los drones consiguen corregir su vuelo y se logra cierta estabilidad.

8.2. Experimento 2

En este experimento probaremos el sistema con tres drones. Como mercancía para el transporte de estos drones utilizaremos una cuerda bastante pesada con una longitud suficiente para mantener separados los drones. Con este sistema valdrá para llevar mercancía más pesada que desestabilizarían dos drones.

Comprobaremos si con una cuerda más pesada los tres drones logran estar estables en el aire durante algunos segundos.

8.2.1. Ejecución

Como hemos explicado la cuerda que emplearemos en este experimento será una cuerda de 350 centímetros, por lo tanto la distancia entre cada drone será de 150 centímetros.

[3] https://www.youtube.com/watch?v=aktZTLORkaU

Figura 52: Situación inicial de los 3 drones

El sistema para corregir la posición será la misma que en la prueba 1, donde tendremos que tener en cuenta los valores de pitch y roll.

Figura 53: Vuelo de los 3 drones

De cara a analizar los resultados se deberá de tener en cuenta que el drone de la derecha será el Drone1, el drone del medio será el Drone2 y el de la derecha Drone3.

8.2.2. Resultados

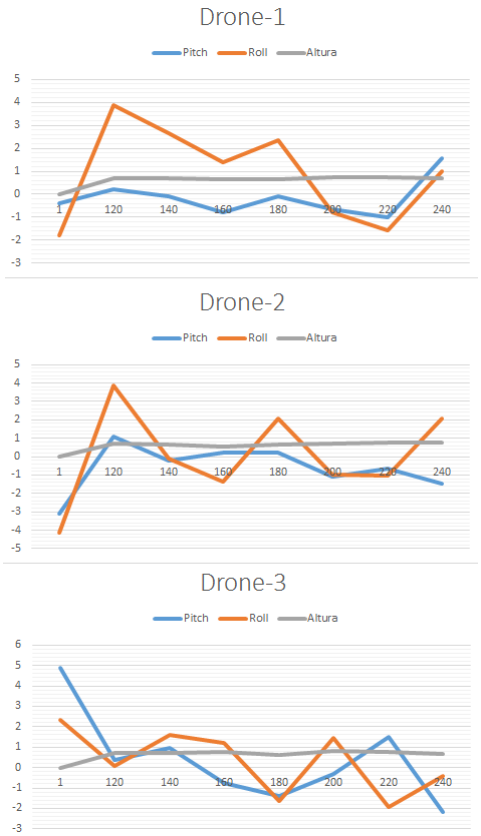

Figura 54: Resultados de los tres drones

Los resultados de esta prueba aparecen en los gráficos de la Figura 54. Para interpretar estos gráficos tenemos de tener en cuenta que en esta prueba los drones no cogerán una cierta altura para tener en cuenta los valores hasta la iteración 120. A partir de este punto se observa que los drones se van estabilizando al entrar los valores entre [-2,2].

Debido a que el Drone2 está en el medio de los tres drones recibirá el viento de los otros dos lo que producirá cierta inestabilidad. Por ello se observa en el gráfico del Drone2 esa pequeña inestabilidad donde el drone intenta arreglar el desplazamiento.

El Drone1 y el Drone3 tendrán una cierta tendencia a volar hacia afuera, esto es, el Drone1 hacia la derecha y el Drone3 hacia la izquierda. Pero esta tendencia no produce que se produzca una gran inestabilidad. Por lo que este sistema responde correctamente cuando se intenta mantener los drones volando estables mientras transportan una cuerda[4].

8.3. Experimento 3

En esta prueba probaremos como en la Prueba 2 transportar una cuerda con tres drones, pero esta vez los drones se desplazarán. Como los drones se tendrán que mover lograr un lugar indoor donde entren los tres drones en horizontal es difícil se ha decidido hacer esta prueba al aire libre, sin tener que estar preocupándose por los golpes.

[4] https://www.youtube.com/watch?v=U-EQwkN3nio

El desplazamiento será hacia adelante por lo que tendremos que tener en cuenta solo el valor del ángulo roll, ya que el ángulo pitch cambiará para que los drones se desplacen hacia adelante.

8.3.1. Ejecución

La cuerda usada será la misma que la prueba anterior, donde la largura de la cuerda será de 350 centímetros y la distancia entre los drones de 150 centímetros.

El tiempo de vuelo será de 8 segundos, en los que los drones se desplazarán a la vez transportando la cuerda.

En esta ocasión hemos cambiado las posiciones de los drones, con lo que el drone del medio será el Drone1, el de la izquierda el Drone3 y el de la derecha el Drone2.

Figura 55: Estado inicial antes del desplazamiento

8.3.2. Resultados

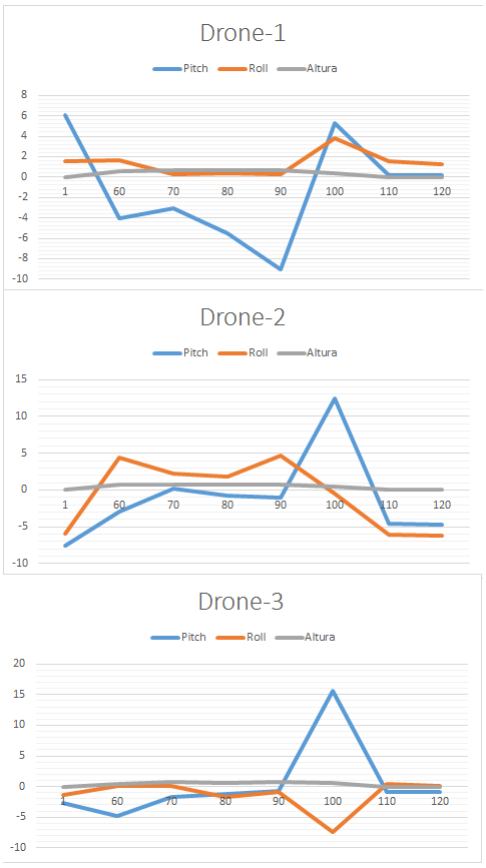

Figura 56: Resultados

El vuelo correcto de los drones se ve reflejado en los gráficos de los resultados de la Figura 56. En ella se observa que los drones no están volando hasta la iteración 60. A partir de entonces el ángulo pitch se mantiene en negativo, logrando así el desplazamiento deseado. Al llegar a la iteración 100 los valores de los ángulos pitch ascienden considerablemente, produciendo que los drones frenen. En cuanto a los valores del ángulo roll se mantienen cercanos a 0, pero si producen un cierto desplazamiento que no se ha podido corregir. A pesar de esto, este ligero desplazamiento no afecta al objetivo [5].

Después de ejecutar esta prueba varias veces, nos dimos cuenta que el Drone1 no conseguía ganar altura debido al peso que tenia que soportar. Por lo tanto se le asignó una velocidad mayor que al Drone2 y Drone3, consiguiendo así que lograse la altura deseada en menos tiempo. Este diferencia de velocidad se observa en los valores del ángulo pitch de la Figura 56, donde en el caso del Drone1 es menor que el de los otros dos

[5] https://www.youtube.com/watch?v=QTmv2Kor2ms

9. Conclusiones

Con el sistema creado hemos podido llevar a cabo nuestros objetivos. Las diferentes pruebas efectuadas nos han dado resultados positivos. El sistema creado podrá ser adaptado a diferentes casos en los que se quieran transportar diferentes cuerdas o cables.

Poder tener en la misma red a los drones y el computador y poder controlarlos ha sido fundamental. Aparte de esto, los módulos utilizados nos han dado la facilidad para crear y controlar este sistema. Además, también nos han permitido crear programas para ejecutar vuelos autónomos y vuelos controlados mediante teclado.

A la hora de analizar todas las pruebas hemos observado que programas creados para ayudar a los drones en la estabilización han funcionado correctamente y que el sistema de sujeción ha aguantado el peso de la cuerda satisfactoriamente.

Por lo tanto las ventajas del sistema creado son los siguientes:

- Facilidad al añadir un nuevo drone: El sistema creado y los módulos utilizados nos facilitan añadir nuevos drones. Estos deberán de ser configurados igual que los demás drones, cambiándoles su SSID y la dirección de red.
- El uso de un solo controlador nos permite tener un control fácil sobre los drones sin tener que estar pendientes de más de un controlador, lo que nos complicaría el manejo del sistema.
- Gracias a los vuelos programados podremos ejecutar los experimentos sin tener que estar manejando el controlador. Estos vuelos serán fáciles de programar y nos darán libertad a la hora de querer hacer otras cosas mientras los drones están volando. Por otra parte, el hecho de poder controlar los drones mientras efectúan el vuelo de manera individual o colectiva servirá para cumplir nuestro objetivos, ya que podremos guiar los drones a nuestro gusto y corregir su vuelo.
- El sistema de sujeción nos permite transportar la cuerda de manera efectiva sin que se produzca ninguna perdida. Además, resulta fácil de modificar puesto que el alambre se podrá amoldar como se quiera.

Aparte de las ventajas hemos detectado diferentes inconvenientes en los experimentos o mientras creábamos el sistema multidrone. Estas desventajas las detallaremos a continuación:

- Al ser el AR.Drone un drone sin mucha fuerza y tener poco peso, el viento les afecta excesivamente. Esto ha dificultado el éxito del experimento del transporte de la cuerda, ya que los drones se han visto afectados por el viento producido por los demás drones.
- El hecho de que el AR.Drone este diseñado para ocio y no para transportar cosas también han afectado a la hora de ejecutar los experimentos, ya que al colgar peso de ellos se han desestabilizado fácilmente. Pero gracias a los programas creados para estos vuelos estas desestabilizaciones son menores. A pesar de esto, los programas que corrigen el vuelo de los drones no han conseguido que el vuelo fuese del todo perfecto, ya que, como hemos observado en las pruebas siempre ha ocurrido algún pequeño desvío. Sin embargo, estos desvíos no han impedido alcanzar el objetivo.

10. Lineas futuras de trabajo

Como complemento de la memoria de este proyecto, en este capítulo detallaremos las futuras líneas de trabajo que se pueden seguir. Debido a que el control de varios drones con un solo controlador, no está muy popularizado las líneas de trabajo futuras pueden ser muy diversas. Algunas de estas están directamente relacionadas con el control de los drones o con diferentes aplicaciones que se pueden llevar a cabo gracias al sistema multidrone. Durante las pruebas realizadas sean ido detallando los diferentes aspectos a corregir para futuros experimentos que mejoren el funcionamiento del sistema. Se proponen las siguientes líneas de investigación futuras:

- Uso de las cámaras para la navegación autónoma: Sin tener que centrarse en el control de los drones una de las líneas de investigación es la de la navegación autónoma de los drones mediante el uso de las cámaras, que se pueden utilizar para visualizar diferentes objetivos y así crear mapas para su navegación autónoma.

- Utilizar diferentes módulos: Con el fin de mejorar y facilitar el control de los drones se pueden utilizar diferentes módulos basados en node.js, entre ellos pueden estar *dronestream* que nos permite tener la visión de nuestro AR.Drone en vivo en nuestro navegador de Internet. Otro de los módulos que pueden ser utilizados es el *Xbox-controller* que nos permite controlar los drones con un mando Xbox.

- Aumentar el número de drones: Nuestras pruebas han sido con tres drones a lo máximo pero se pueden hacer diferentes pruebas con más UAVs. Esto permitirá llevar mercancías más pesadas como pueden ser mangueras o cables.

- Uso de diferentes controladores: No solo se podrá controlar el sistema con un ordenador, sino que, se puede investigar el uso de diferentes controladores, como pueden ser Asterix, Ipad, Arduino etc.

- Uso diferente de la coordinación entre drones: Como hemos explicado anteriormente la coordinación entre drones tiene diferentes usos como pueden ser el control de una zona. La modificación del sistema de AR.Drones para amoldarlo a este objetivo sería uno de los trabajos futuros.

A. Referencias

Como punto de partida para hacer este proyecto se han utilizado diversas referencias y entre ellas se encuentran la tesis de Juan Pablo Fuentes [4] y de Tommaso Bresciani [5], que nos han servido como fuente de ideas e ilustraciones para saber como encaminar nuestro trabajo. Aparte de estas referencias, como punto de partida también hemos utilizado la guía de desarrollo para AR.Drone [7], donde se detallan el funcionamiento del SDK y el de las demos que nos han servido para comprender el funcionamiento de los drones, al igual que nos ha facilitado las cosas a la hora de escribir los programas. Las imágenes utilizadas para explicar los movimientos de los cuadricópteros son sacados de esta guía.

En cuanto al uso de los módulos ha sido de gran ayuda el manual[2] que explica el uso de algunos de los diferentes módulos basados en Node.js. Sin olvidarnos de la web de Nodecopter [1] que recoge todos los módulos que se pueden utilizar.

Cuando hemos tenido que configurar los drones para crear una misma red la información ha sido recogida de dos diferentes webs [6] [8]. Por otra parte, la idea de la coordinación entre UAVs y los ejemplos de estos los hemos cogido del articulo de Randal Beard y su equipo [3], donde se explica lo necesario para la coordinación y lo hemos detallado en el capitulo del Estado del Arte.

Referencias

[1] Nodecopter. `http://nodecopter.com/hack`.

[2] Felix Geisendorfer Andrew Nesbitt and Robin Mehner. *Flying robots*, 2014.

[3] R.W. Beard, T.W. McLain, D.B. Nelson, D. Kingston, and D. Johanson. Decentralized cooperative aerial surveillance using fixed-wing miniature uavs. *Proceedings of the IEEE*, 94(7):1306–1324, July 2006.

[4] Juan Pablo Fuentes Brea. *Arquitectura cognitiva hibrida para la navegacion autonoma de UAVs mediante mapas topologicos visuales.* PhD thesis, UPM, Facultad de Informatica, 2014.

[5] Tommaso Bresciani. *Modelling, Identification and Control of a Quadrotor Helicopter.* PhD thesis, Lund University, Octubre 2008.

[6] Omar Rizwan Eric Smalls and John Backus. Programming multiple parrot a.r. drones on one network with node.js. `http://drones.johnback.us/`, 2013.

[7] N. Brulez S. Piskorski and F. D'Haeyer. Ar.drone developer guide sdk 2.0. 2012.

[8] Pras Velagapudi. Enabling infrastructure wifi on the ardrone with dhcp. `http://www.snowbotic.com/archives/68`, 2011.

B. Códigos anexos

Dentro de este apartado se detallarán las diferentes partes del programa utilizado para llevar a cabo los experimentos. En el Algoritmo 1 se detallan los módulos utilizados en los diferentes experimentos. Estos módulos han sido los mismos en todas las pruebas. Después de esto, se definirán la cantidad de drones que queramos utilizar para hacer nuestros experimentos y por cada uno de ellos habrá que crear un nuevo cliente que permita controlarlo. En en Algoritmo 2 se crearán tres clientes que serán los utilizados para las pruebas.

En los experimentos realizados hemos utilizado dos tipos de control. En el Algoritmo 3 se detalla una parte del programa utilizado para poder hacer un vuelo autónomo y en Algoritmo 4 el programa para controlar los drones mediante el teclado. Por otro lado, en el Algoritmo 5 se definen las partes del *navdata* que se quieran recogen para después utilizarlos y, para finalizar, el Algoritmo 6 contendrá la parte del programa que servirá para imprimir los resultados. También, en este último Algoritmo se detalla la parte del programa utilizado para corregir el vuelo de los drones y que se utiliza para todos los experimentos, variando algunos valores según el objetivo.

Algoritmo 1 Referencias a los módulos utilizados

```
var keypress = require('keypress');
var arDrone = require('ar-drone')
, arDroneConstants = require('ar-drone/lib/constants');
```

Algoritmo 2 Clientes

```
var drone1 = arDrone.createClient({ip : '192.168.1.100'});
var drone2 = arDrone.createClient({ip : '192.168.1.200'});
var drone3 = arDrone.createClient({ip : '192.168.1.1'});
```

Algoritmo 3 Indicaciones para el vuelo programado

```
drone1.takeoff();
drone1.front(0.3);
drone1.after(6000, function(){
this.stop();
this.land();
});
drone2.takeoff();
drone2.front(0.3);
drone2.after(6000, function(){
this.stop();
this.land();
});
drone3.takeoff();
drone3.front(0.3);
drone3.after(6000, function(){
this.stop();
this.land();
});
```

Algoritmo 4 Control de los drones con el teclado

```
keypress(process.stdin);
process.stdin.on('keypress', function(ch, key){
if(key && key.name == 'space' ){
console.log('Takeoff');
drone1.takeoff();
drone2.takeoff();
drone3.takeoff();
}
if(key && key.name == 'l'){
console.log('Land!');
drone1.stop();
drone1.land();
drone2.stop();
drone2.land();
drone3.stop();
drone3.land();
}
if ( key && key.name == 'w'){
console.log('Forward!');
drone1.front(0.1);
drone2.front(0.1);
drone3.front(0.1);
}
if(key && key.name == 'c'){
console.log('Quitting');
process.stdin.pause();
drone1.stop();
drone1.land();
drone2.stop();
drone2.land();
drone3.stop();
drone3.land();
drone1._udpControl.close();
drone2._udpControl.close();
drone3._udpControl.close();
}
process.stdin.setRawMode(true);
process.stdin.resume();
```

Algoritmo 5 Definir las partes del navdata deseados

```
function navdata_option_mask(c){
return 1 << c;
}
var navdata_options = (
navdata_option_mask(arDroneConstants.options.DEMO) |
navdata_option_mask(arDroneConstants.options.VISION_DETECT) |
navdata_option_mask(arDroneConstants.options.PWM)
);
```

Algoritmo 6 Funciones para corregir el vuelo e imprimir resultados

```
drone2.config('general:navdata_demo', true);
drone2.config('general:navdata_options', navdata_options);
var j =0;
drone2.on('navdata', function(d){
if(d.demo){
//Imprimir los resultados entre las iteraciones 20 y 120
if (j == 0 || (j>20 && j<120 )){
console.log(["2,"+d.demo.altitude]);
console.log([d.demo.leftRightDegrees+","+d.demo.frontBackDegrees+","+j]);
}
j = j+1;
//Corrección de la posición del drone
if(d.demo.altitude > 0.5){
if(d.demo.leftRigtDegrees < -2){
drone2.right(0.26);}
if(d.demo.leftRigtDegrees > 2){
drone2.left(0.26);} }
}
});
drone1.config('general:navdata_demo', true);
drone1.config('general:navdata_options', navdata_options);
var i =0; drone1.on('navdata', function(d){
if(d.demo){
//Imprimir los resultados entre las iteraciones 20 y 120
if (i==0 ||(i>20 && i<120)){
console.log(["1,"+d.demo.altitude]);
console.log([d.demo.leftRightDegrees+","+d.demo.frontBackDegrees+","+i]);
}
i=i+1;
//Corrección de la posición del drone
if(d.demo.altitude > 0.5){ if(d.demo.leftRigtDegrees < -2){
drone1.right(0.26);}
if(d.demo.leftRigtDegrees > 2){
drone1.left(0.26);} }
}
});
drone3.config('general:navdata_demo', true);
drone3.config('general:navdata_options', navdata_options);
var x =0;
drone3.on('navdata', function(d) {
if(d.demo){
//Imprimir los resultados entre las iteraciones 20 y 120
if (x == 0 || (x>20 && x<120 )){
console.log(["3,"+d.demo.altitude]);
console.log([d.demo.leftRightDegrees+","+d.demo.frontBackDegrees+","+x]);
}
x=x+1;
//Corrección de la posición del drone
if(d.demo.altitude > 0.5){
if(d.demo.leftRigtDegrees < -2){
drone3.right(0.26);}
if(d.demo.leftRigtDegrees > 2){
drone3.left(0.26);} }
}
});
```