Universidad Autónoma de Baja California.

Facultad de Ingeniería, Arquitectura y diseño.

Ingeniería en electrónica.

Tesis.

“Trayectorias programadas en drones”

Que con el objetivo de cubrir parcialmente los requisitos para obtener el grado de Ingeniero en Electrónica y Comunicaciones presenta: Ricardo Crespo Chávez.

Ensenada, Baja California, noviembre de 2016.

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA.

FACULTAD DE INGENIERÍA, ARQUITECTURA Y DISEÑO

Trayectorias programadas en Drones

TESIS

Que para obtener el grado de Ingeniero en Electrónica presenta:

RICARDO CRESPO CHÁVEZ

Y aprobada por el siguiente comité

Dra. Rosa Martha López Gutiérrez
Co-Directora
Presidente

Dra. Liliana Cardoza Avendaño
Co-Director
Sinodal

Secretario Dr. Adrián Arellano Delgado
Sinodal

Dr. Cesar Cruz Hernandez
Sinodal

Dr. Humberto Cervantes de Ávila
Sinodal

Agradecimientos.

En primer lugar me gustaría agradecerle al Dr. Cesar Cruz Hernández, por brindarme la posibilidad de llevar a cabo el proyecto con todo su apoyo y profesionalidad.

A la Dra. Rosa Martha López Gutiérrez por su gran atención y ayuda en realizar los trámites pertinentes, así como la involucración en esta tesis que me ha instruido.

Al CONACYT de México a través del proyectos de grupos de investigación No. 166654

Y por último, pero no menos importante a mi familia y compañeros de generación así como a todos los maestros que a lo largo de la carrera me formaron como individuo y como profesionista, por su apoyo incondicional, Gracias.

Tabla de contenidos.

Capítulo 1.

1.1 Introducción y objetivos.

Los robots aéreos (Drones) en los últimos años se han convertido en parte importante de la humanidad, debido a qué con ellos se pueden hacer tareas que para el hombre es muy difícil o imposible realizarlo, por tal motivo es importante abordar este tema ya que se pueden desarrollar diversas aplicaciones como lo son el patrullaje en un área determinada, entregar paquetería como lo hace Amazon actualmente o incluso ir a áreas en las que para el ser humano es difícil acceder.

Los modelos de los vehículos aéreos fueron evolucionando acorde a los recursos del tiempo y de la época, agregándoseles accesorios más sofisticados que el mercado brinda. Por su complejidad, el planeamiento, la construcción y el diseño de los modelos a escala estimulan la curiosidad de los jóvenes y adultos.

En 2004 la compañía francesa Parrot comenzó un proyecto llamado A.R. Drone, el objetivo era producir un micro vehículo aéreo no tripulado o UAV para el mercado masivo de los juegos de video y entretenimiento en el hogar. En este proyecto han participado de 5-12 ingenieros de Parrot con el apoyo de técnicos de ingenieros de SYSNAV y Paris Tech para juntar esfuerzos en la construcción del sistema de navegación y el diseño del control [13].

Una de las características únicas es que es una plataforma estabilizada aérea, controlada a distancia a través de una interfaz gráfica fácil de usar que se ejecuta a través de un cualquier dispositivo móvil (Smartphone) o sistema operativo. Este es un proyecto de alta ingeniería es uno de los más importantes ya que utiliza sensores relativamente baratos, empleando algoritmos los cuales hacen que sean muy precisos y con ello tener un control de vuelo estacionario o de avance rápido.

Actualmente en el mercado se pueden encontrar diversos tipos de Drones en un precio regular, debido a esto el interés de jóvenes o investigadores apegados al área de ingeniería se han acercado a ellos para explorar nuevas posibilidades reales de aplicaciones como lo ha hecho google con su “Project Wing” que hasta la fecha solo se utilizan como repartidor de suministros como lo son el agua o croquetas para perro.

El interés está puesto en los cuadricopteros, que “incorpora un sistema de vuelo con cuatro hélices y motores. Con ellos se puede manejar en interiores y son capaces de estabilizarse en un punto, a diferencia de los aviones, lo cual les otorga una gran ventaja

sobre ellos.”[10] Los cuadricopteros no solo cuentan con motores y hélices sino que a su vez cuentan con sensores que pueden medir diferentes variables como lo son la altura en la que están localizados, cámaras de alta definición para tener una mejor visión a la hora de volar, sensores de inclinación y rotación, conexión WiFi, incluso algunos modelos cuentan con GPS con la finalidad de otorgarle al usuario información.

En este trabajo de tesis el objetivo principal es realizar aplicaciones sencillas para dar hincapié y adentrar más en los cuadricopteros realizando programas sencillos que realicen trayectorias básicas recorriendo una distancia en forma de sinusoidales, señal cuadrada, triangular así como realizar círculos, cuadrados etc. con lo cual ir poco a poco adentrando en estos revolucionarios dispositivos y con ello llegar a una aplicación robusta y eficaz en trabajos a futuro.

Se eligió como trabajo de tesis **trayectorias con cuadricopteros** debido a que quería aprender un poco más de programación orientada a objetos y aparte nadie había querido trabajar con ellos sobre todo sería una gran oportunidad para comenzar a realizar aplicaciones sencillas con ellos y con ello motivar en algún futuro a que jóvenes se interesen en querer seguir trabajando y realizar mayores aplicaciones.

En las trayectorias de los cuadricopteros se utilizó programación de alto nivel como lo es Python, JavaScript y C, para la programación en JavaScript se utilizó una herramienta de programación llamada NodeJs y un sistema operativo especial para los cuadricopteros que facilita en gran medida el manejo de dichos dispositivos.

Capítulo 2.

2.1. Antecedentes.

En enero de 1921, La fuerza aérea de Estados Unidos contrató al Dr. George de Bothezat y a Ivan Jerome para desarrollar una máquina que volara de forma vertical. La estructura en forma de "X" de 1678 Kg con seis aspas en cada brazo de 9m. En los extremos de los brazos laterales, se utilizaron dos pequeñas hélices de paso variable para empujar el control hacía donde ir. El Cuadricoptero pesó 1700Kg en el aire y hace su primer vuelo en octubre de 1922. [1]

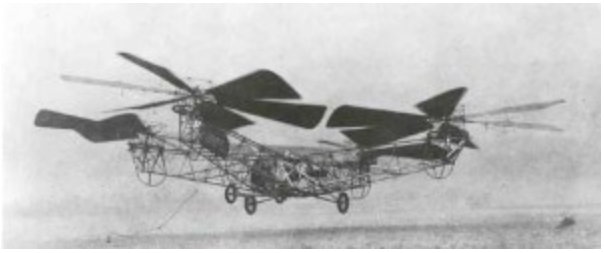


Figura 2.1. Primer cuadricoptero construido por Bothezat y Jerome.

Con esto sin duda se da un gran avance tecnológico al ser el primer cuadricoptero en realizar alzar vuelo con una duración de "1 minuto 42 segundos con 1.8 metros desde el suelo el 18 de diciembre de 1922" [2], con esto se dio un gran paso en el desarrollo de vehículos aéreos.

Los Vehículos aéreos no tripulados (UAV: Unmanned Aerial Vehicle) es una aeronave que vuela sin tripulación humana a bordo, que pueden ser controladas desde una estación base o tener un funcionamiento autónomo mediante un algoritmo prestablecido [3]. Los UAV se remontan desde un poco después de la Primera Guerra Mundial, siendo usados durante la Segunda Guerra Mundial.

Figura 2.2. UAV Predator de los EUA.

En la actualidad con todo el avance tecnológico que se ha venido desarrollando a lo largo del tiempo con la salida del transistor y con ello las compuertas lógicas para con ello llegar a microprocesadores con los cuales se pueden realizar diversas aplicaciones como lo son en este caso los cuadricopteros controlados por radiofrecuencia en el mercado a un muy asequible precio con el cual cualquier persona puede realizar aplicaciones bastante

robustas.

Figura 2.3 Cuadricoptero actual.

Los cuadricopteros utilizados hoy en día cuentan con una gran variedad de sensores con los cuales el usuario puede darse cuenta con base a esa información a parámetros que rodean al cuadricoptero como lo son la altura en la que se encuentra, la velocidad a la que se va, la distancia que recorre, observar el entorno y muchas otras aplicaciones que cuenta para una mejor experiencia. Algo importante a destacar es el tamaño del primer cuadricoptero en 1922 a los que hay hoy en día son muy pequeños, tanto que en un cuarto se puede trabajar con ellos.

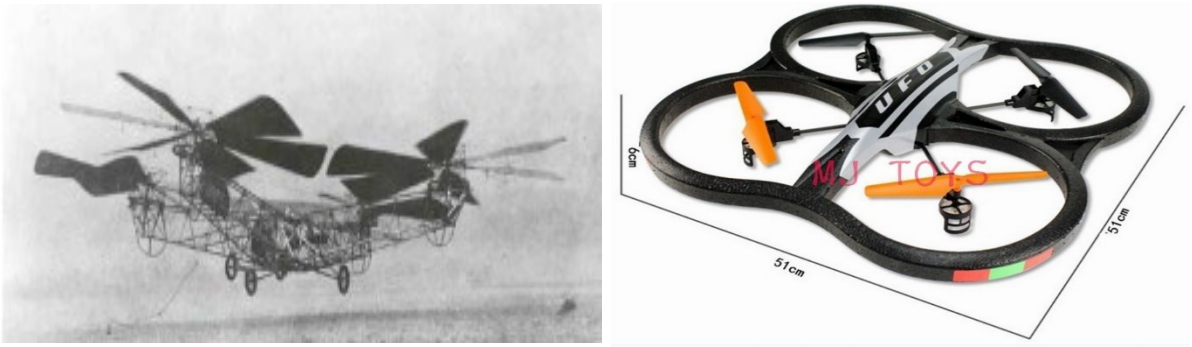


Figura 2.4. Comparativa entre el primer cuadricoptero y los de la actualidad.

La integración de la tecnología de la Segunda Guerra Mundial de UAV a los cuadricopteros tuvo como resultado los Drones y así con ello poder tener el control remoto en un dispositivo y realizar labores que el hombre en ocasiones es imposible realizar o en otro caso entretener a jóvenes y adultos ya que con esa tecnología hace posible que

cualquier usuario sin conocimientos de aeronáutica o con licencia de piloto pueda manejarlos gracias a su tecnología.

"Los UAV han demostrado en diferentes escenarios y especialmente en la Guerra del Golfo y en la Guerra de Bosnia, el gran potencial que pueden tener, en cuanto a la obtención, manejo y transmisión de la información, gracias a la aplicación de nuevas técnicas de protección de la misma. El ejército estadounidense se dio cuenta de la importancia y necesidad de emplear este tipo de plataformas, con el fin de manejar eficiente y discretamente su información, de esta manera en la operación Tormenta del Desierto en 1991, la Armada estadounidense utilizó el sistema UAV Pioneer Israel para suministrar inteligencia a nivel táctico, en Afganistán durante la operación Paz Duradera el sistema UAV Predator realizó misiones de reconocimiento armado y en el 2003 en Irak atacó objetivos de gran valor para la coalición" [5].

2.2. Principios de funcionamiento.

Cada rotor es responsable de una cierta cantidad de torque y empuje en su centro de rotación, Sus hélices no son todas iguales, en efecto se dividen en dos pares: dos de empuje y dos de desplazamiento, que trabajan en contra rotación. Como consecuencia, el torque neto resultante puede ser nulo si todas las hélices giran con la misma velocidad angular, permitiendo que el vehículo se mantenga estable en el aire alrededor de su centro de gravedad [11].

Con la finalidad de definir la orientación se han definido 3 parámetrosdinamicos: Yaw, Pitch y roll.

Yaw: permite girar en 360 grados en su mismo eje.

Pitch: Permite el avanzar o retroceder.

Roll: Permite ir de derecha a izquierda. [12]

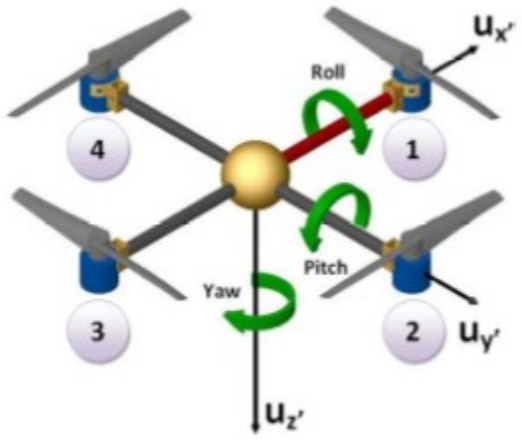


Figura 2.5. Ángulos de Yaw, Pitch y Roll.

2.3. Clasificación:

Existen varias maneras de clasificar a los vehículos aéreos no tripulados. A continuación se presentan algunos de ellos [6].

- Por el tipo de misión para el que fueron diseñados:
  - Simulación de blancos (confundir al enemigo).
  - Reconocimiento de terreno.
  - Combate.
  - Logística (transportar carga)
  - Investigación.
  - Desarrollo.

- Por su alcance:

  En este caso son de corto alcance debido a sus limitaciones, ya que se clasifican algunos por que no alcanzan los 600 metros de altura o los 2 kilómetros de distancia.

- Por el tipo de despegue:
  - Vertical: Como lo son helicópteros.
  - No vertical: Como lo son Aeronaves.

2.4. Aplicaciones.

Como se sabe un cuadricoptero (Dron) es un UAV en la actualidad se le pueden encontrar en diversas áreas y a continuación se dará a conocer algunas de sus aplicaciones en las cuales estos robots pueden hacer de mucha ayuda:

En eventos.

Como ya es costumbre se pueden observar Drones en eventos como mundiales de futbol, recitales, desfiles de moda incluso en jornadas de ingeniería ya han estado. Esto es debido a que pueden estar a una baja altura a diferencia de helicópteros que pueden ser muy peligrosos para las personas y pues claro no muchos tienen el dinero para comprarlo. Sin duda es uno de las aplicaciones en las cual se le ha dado mayor uso ya que las personas desean grabar sus momentos especiales.

Figura 2.5. Imagen de novios capturada por un Dron.

Como repartidor.

En Rusia, Israel y Argentina los Drones se encuentran haciendo envios de pizza. En China, la empresa de correo SF express hace envíos a través de ellos.

Figura 2.6. Dron Repartidor de Pizza.

En situaciones de emergencia para búsqueda de personas.

Esta es una aplicación muy importante debido a que es una fácil y rápida alternativa en el momento que hay un desastre natural en alguna localidad, con ella se pueden localizar a personas que estén en peligro e ir en su rescate de inmediato y a su vez ofreciendo ayuda

para que pueda resistir en lo que la ayuda llega.

Figura 2.7. Dron en área de desastre y rescate de personas.

Vigilancia fronteriza.

Esta es una medida que se debe de tomar para regularizar la entrada ilegal a países, y el gobierno de EUA y España toman cartas en el asunto debido a que no solo es el paso de inmigrantes sino el narcotráfico que hay en la frontera EUA-México.

Figura. 2.8 Drone utilizado para la vigilancia de narcotráfico y cruce de inmigrantes.

En la agricultura.

Un uso que a los agricultores sacan gran provecho a los drones debido a las fotos y videos de alta definición que permiten cámaras que se pueden incorporar a os drones hace factible el monitoreo de grandes áreas, incluso para esparcir pesticidas y fertilizantes en

grandes terrenos como lo es en Asia.

Figura 2.9. Drones en la agricultura.

Como satélites.

Esta investigación pero se quieren utilizar los drones para llevar internar a áreas donde aún no es posible que llegue, funcionarían con energía solar y podrían cumplir la tarea de satélites baratos y en un área específica.

Figura. 2.10. Proyecto con UAV de google para dar internet a todo el mundo.

Patrullaje Urbano.

Como no es posible que la policía esté todo el día en todo lugar o en todo momento, sin duda esta será un gran beneficio para las personas al sentirse un poco más seguras

sabiendo que un Dron está vigilando.

Figura 2.11. Drone de la policía.

Capítulo 3.

3.1 Experimentación.

En esta sección se encontrarán los programas, librerías y código empleado para la realización de esta tesis, así como una ligera descripción de como instalar todo lo necesario para comenzar a trabajar.

3.1.1 Paqueterías usadas.

En esta sección se muestran todas las diferentes paqueterías que se utilizaron para hacer que el ardrone despegue de manera autónoma. Se anexan tanto programas de alto nivel como sistemas operativos hasta las librerías necesarias. [7]

Comencemos por hablar un poco de Ubuntu.

3.1.2 Ubuntu.

Ubuntu es una distribución Linux que ofrece un sistema operativo predominantemente enfocado a ordenadores de escritorio aunque también proporciona soporte para servidores. [8]

Basada en Debian GNU/Linux, Ubuntu concentra su objetivo en la facilidad de uso, la libertad en la restricción de uso, los lanzamientos regulares (cada 6 meses) y la facilidad en la instalación. Ubuntu es patrocinado por Canonical Ltd., una empresa privada fundada y financiada por el empresario sudafricano Mark Shuttleworth.[8]

En este caso Ubuntu será nuestro sistema operativo utilizado ya que es fácil de utilizar, amigable con el usuario y sobretodo ya hay mucho trabajo desarrollado para poder trabajar, además de foros en los que puedes participar y obtener ayuda.

Ahora toca hablar un poco acerca del código a utilizar, el cual se llama Node.Js que es una distribución de Java Script.

3.1.3 Node.JS

Es el programa principal. Este programa es el que ejecuta los diferentes códigos que se escriben para dar las diferentes rutas al ARdrone. Para fines prácticos podemos decir que node.js cumple la función de compilar un código y conectar un sistema de red inalámbrica (En este caso ARDRONE) con una computadora. [7]

Node.js es similar en su propósito a Twisted o Tornado de Python, Perl Object Environment de Perl, React de PHP, libevent o libevde C, EventMachine de Ruby, vibe.d de D y de Java existe Apache MINA, Netty, Akka, Vert.x, Grizzly o Xsocket. Al contrario que la mayoría del código JavaScript, no se ejecuta en un navegador, sino en el servidor. Node.js implementa algunas especificaciones de CommonJS.5 Node.js incluye un entorno REPL para depuración interactiva. [7]

3.1.4 Node-sylvester.

Esta paquetería es necesario instalar para que la librería de ardrone-autonomy funcione. Sylvester resuelve matrices y sistemas de ecuaciones lo cual lo hace una herramienta bastante poderosa a la hora de programar.[7]

3.1.5 Async.

Esta paquetería es necesaria ya que se tiene que instalar para que la librería ardrone-autonomy funcione. Async permite mezclar código JavaScript asíncrono con node.js.[7]

3.1.6 Librerías.

Ahora hablemos un poco acerca de las librerías que se utilizan, las cuales son “ar-drone” y “ardrone-autonomy”.

Ar-drone: es un modulo maestro que contiene gran parte de las instrucciones para contral el ardrone. Basicamente esta librería lo que nos da a entender es que esta librería únicamente se utiliza para realizar comunicación entre el ordenador y el Drone, ya que sus instrucciones son algo limitadas. Esta librería es creada por usuarios de la comunidad social GITHUBE. [7]

Ardrone-autonomy: Esta es otra librería que se instala sobre la librería ar-drone. Es muy importante entender que esta librería funciona a partir de la librería ar-drone, esto

debido a que si no se instala sobre la misma entonces no funcionará y nuestros códigos no se compilarán. En esta librería se tienen las instrucciones que permiten indicarle al ARdrone que se mueva una cierta cantidad de distancia [7].

3.1.7 Proceso de instalación del ambiente de trabajo.

En esta sección se puede encontrar el modo de instalar Ubuntu, node.js y todas las paqueterías necesarias para comenzar a programar.

3.1.7.1 Instalar Ubuntu.

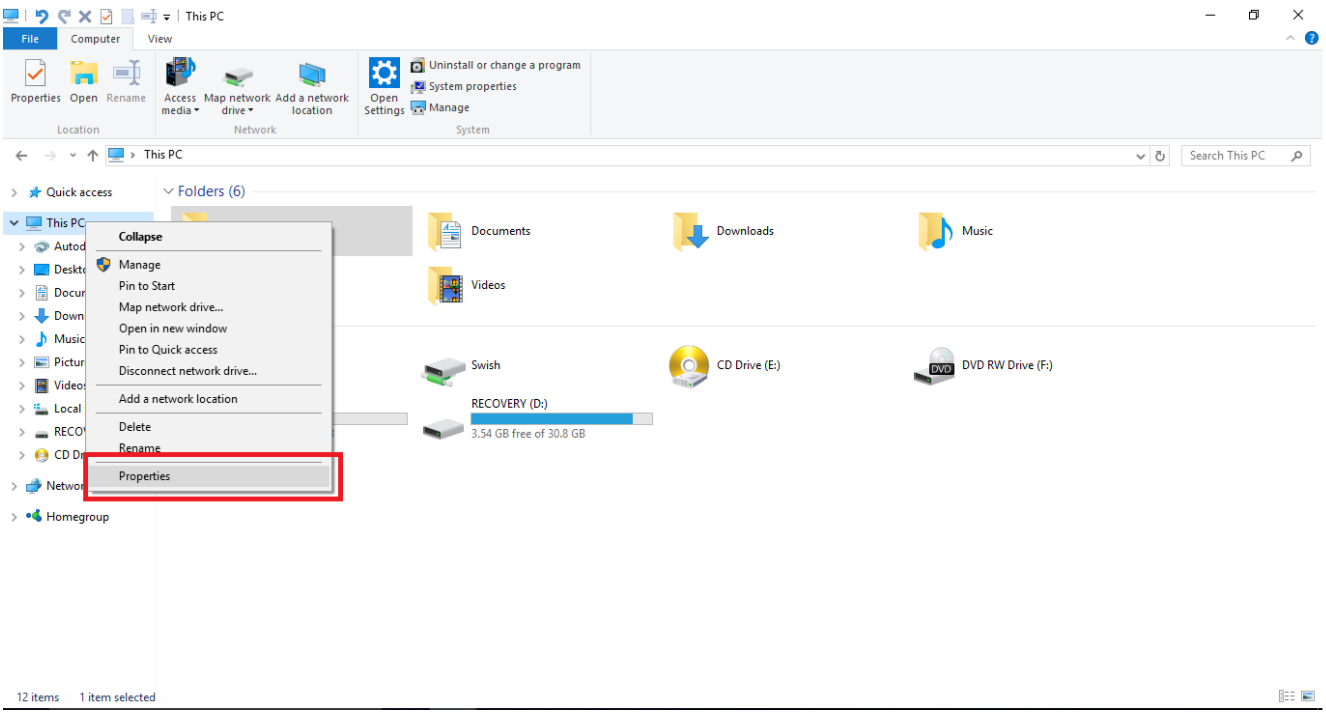
Principalmente hay dos opciones para instalar Ubuntu, una de ellas es crear una máquina virtual en tu computadora (esto únicamente si tu computadora cuenta con espacio en disco duro suficiente y RAM de sobra) que a mi parecer es una de las formas más fáciles de trabajar ya que no hay que crear particiones y es fácil pasar de Windows a Ubuntu sin tener que apagar la computadora e iniciar el sistema operativo, o en su defecto crear una partición en tu disco duro e instalar Ubuntu en una parte en blanco creada especialmente para ella.

En esta sección se verá ambas formas de instalación.

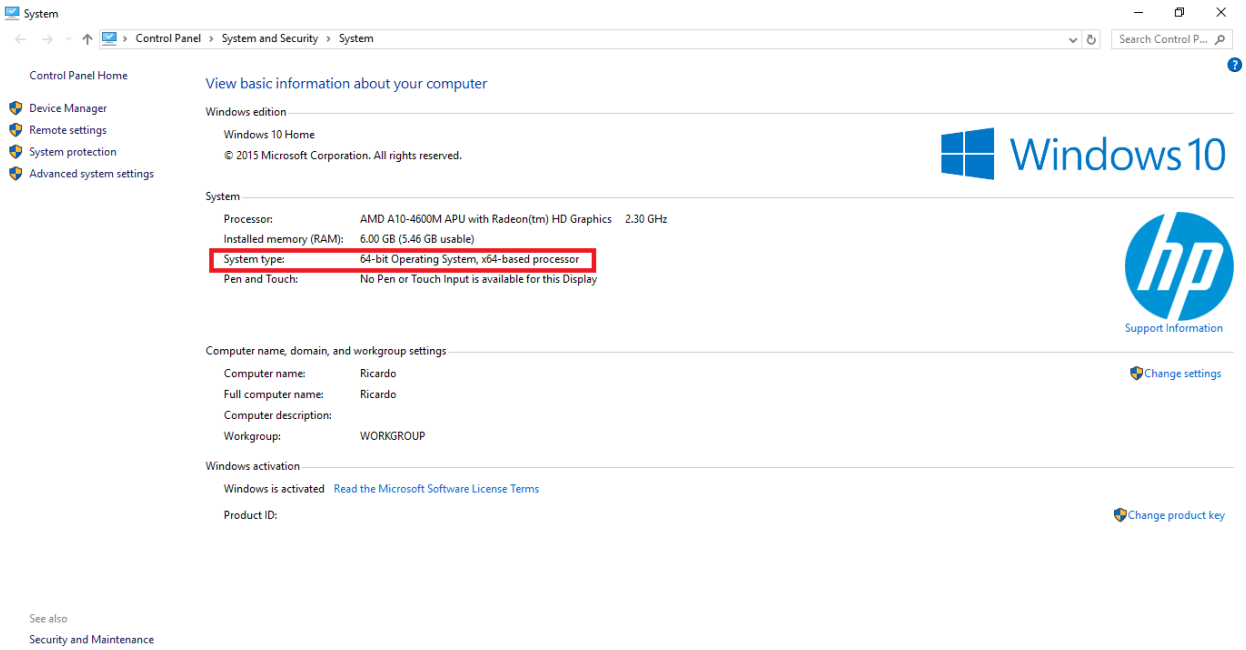
3.1.7.2 Partición en disco.

1. **El primer pasó a realizar es ver que estructura tiene la computadora (32 o 64 bits).**

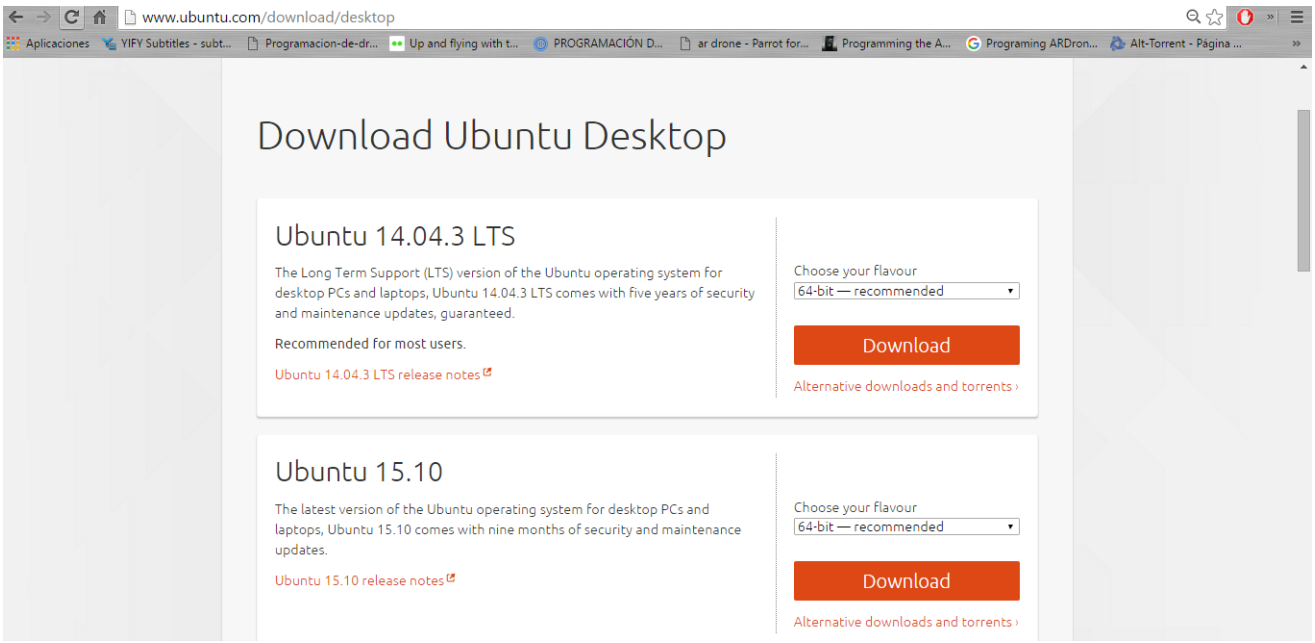
Para ello debes de ir a “mi pc”, dar clic derecho e ir a la opción propiedades.



Una vez que demos clic derecho aparecerá una nueva ventana en la cual encontraras la información de tu sistema operativo.



Ahora que ya tenemos la estructura de programación de nuestro sistema operativo (en mi caso 64 bits) vamos a la página de Ubuntu a descargar el S.O.

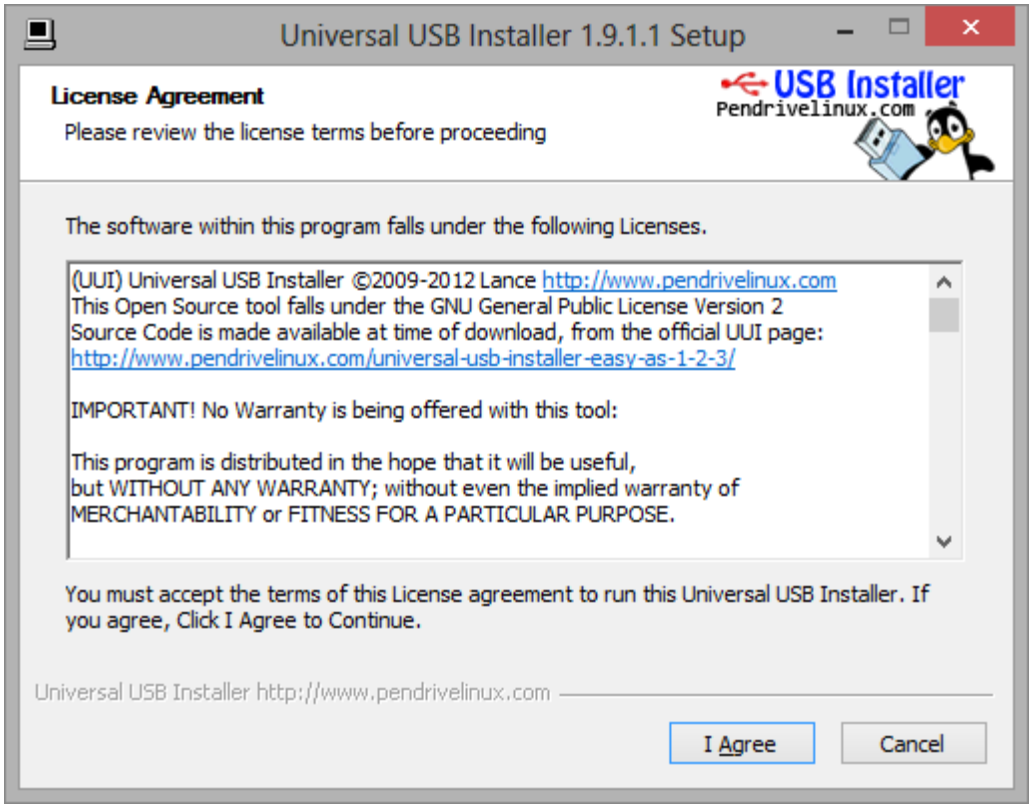


Para esta aplicación se utilizó Ubuntu 12.0.4 así que es recomendable que busquen esa versión en la página oficial de Ubuntu.

2. Crear un pendrive con la imagen del sistema operativo Ubuntu.

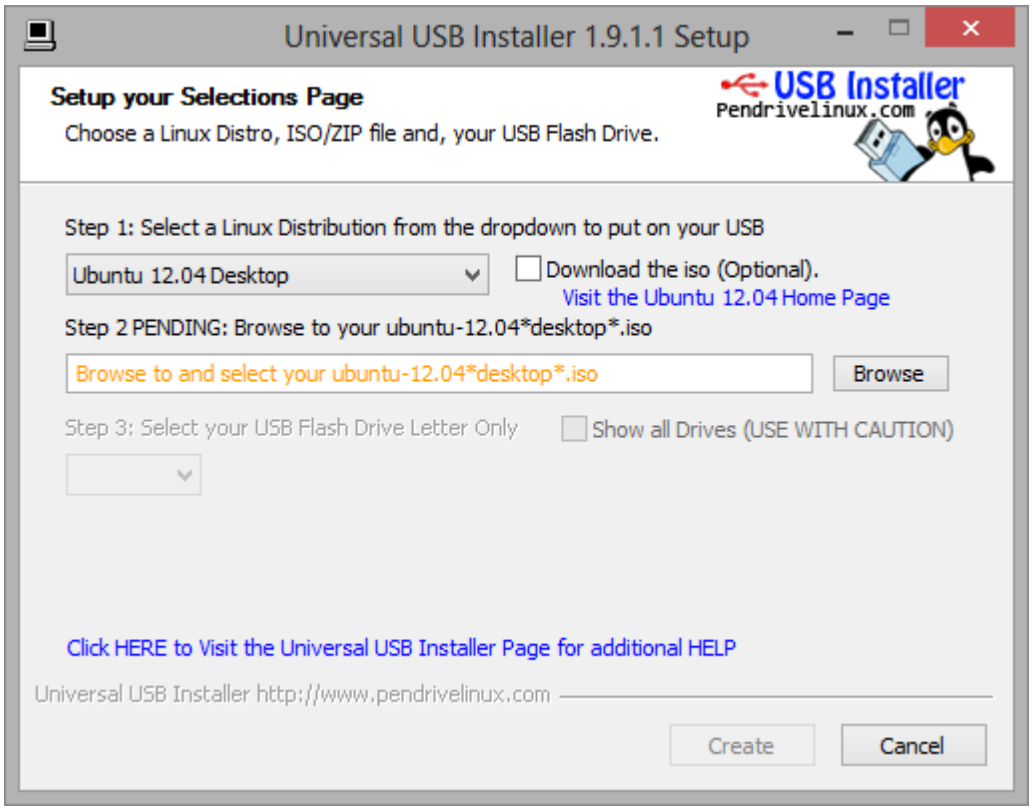
Lo primero que tenemos que hacer después de descargar el sistema operativo de Ubuntu es descargar el Universal USB Installer y, eso, lo podemos hacer desde el siguiente link

http://www.pendrivelinux.com/universal-usb-installer-easy-as-1-2-3/#button

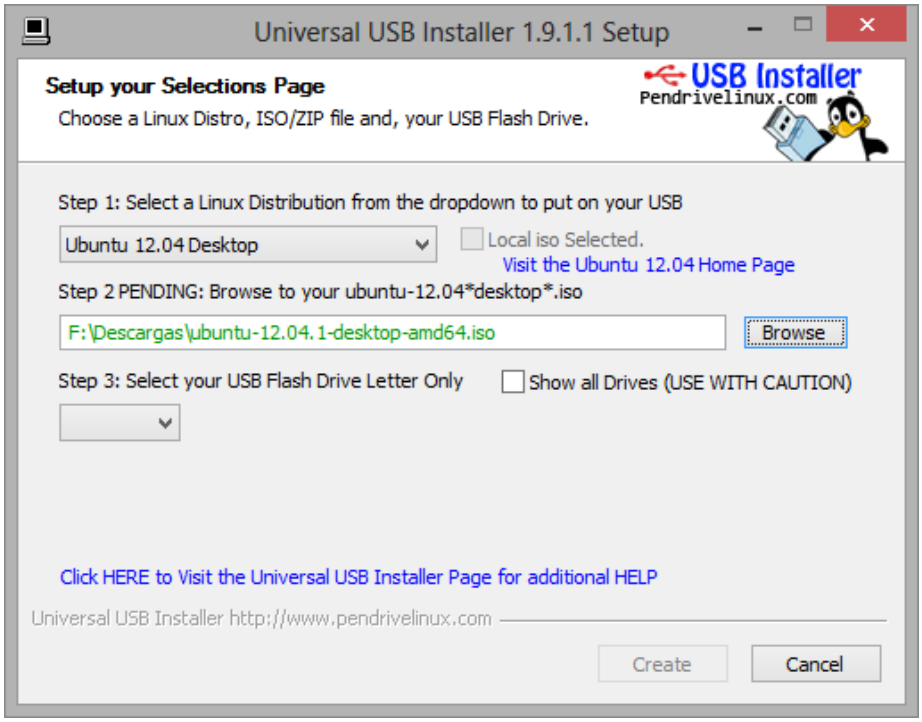


Una vez descargado, lo abrimos (dando permisos de administrador) y aceptamos la licencia:

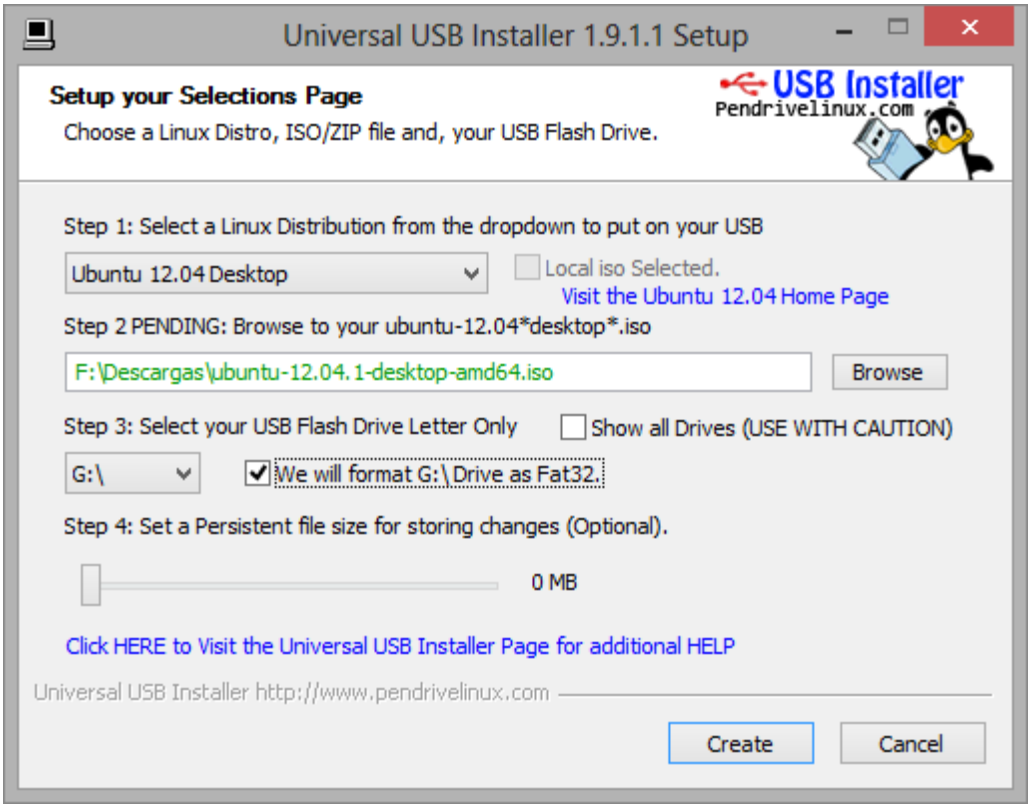
Aceptamos la licencia



Como se puede ver, elegimos en el paso 1 la distribución que queremos grabar en el USB, para este caso, la versión Desktop de Ubuntu 12.04.5 es la que utilizamos en todas las pruebas. Ahora le damos a "Browse" y buscamos donde tengamos la imagen que nos



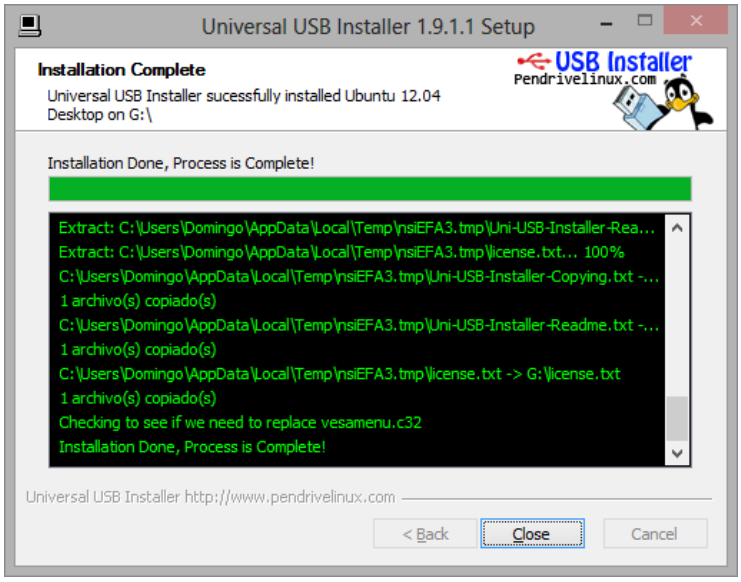
hemos descargado de Ubuntu.



Ahora seleccionamos la memoria USB que haremos booteable.

En este caso era la unidad G:\ pero puede y sea algún otro puerto, debes de elegir bien y ver bien en que unidad está o podrás borrar tu disco duro.

Marcamos la casilla “We Will format...” damos clic en “créate” y aparecerá un mensaje, darás “yes”.



Una vez finalizado el proceso aparecerá una ventana como esta.

3. Ajustar la configuración de arranque en la BIOS

Ubuntu 12.04 se instala siguiendo un sencillo asistente de unos pocos pasos en el que lo único que tiene un poco más de complicación es el particionado del disco. Tanto si vamos a instalar Ubuntu desde una memoria USB deberemos arrancar el equipo con la memoria insertado en él y habiendo modificado la secuencia de arranque en la BIOS. En la mayoría de los sistemas se puede hacer pulsando la tecla F12 mientras aparecen las primeras letras después de encender el ordenador.

Este procedimiento es diferente en todas las computadoras por lo que se espera que cada usuario investigue por su cuenta este procedimiento.

4. Instalar Ubuntu.

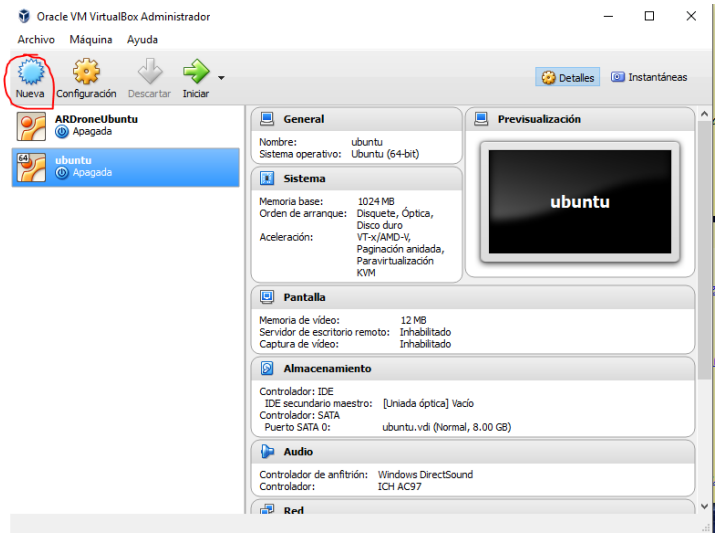
Debido a que la instalación es bastante amigable, no es necesario profundizar en esto. Una vez configurada la BIOS para iniciar desde una USB solo es cuestión de seguir los pasos para la instalación de Ubuntu. Lo que si es importante a considerar es qué debes tener muy en cuenta al momento de hacer la partición y no eliminar por completo el disco duro.

Máquina Virtual.

En este caso únicamente se repite la parte 1 en la cual debes de ver la estructura de tu ordenador. Y haber descargado Ubuntu con anterioridad.

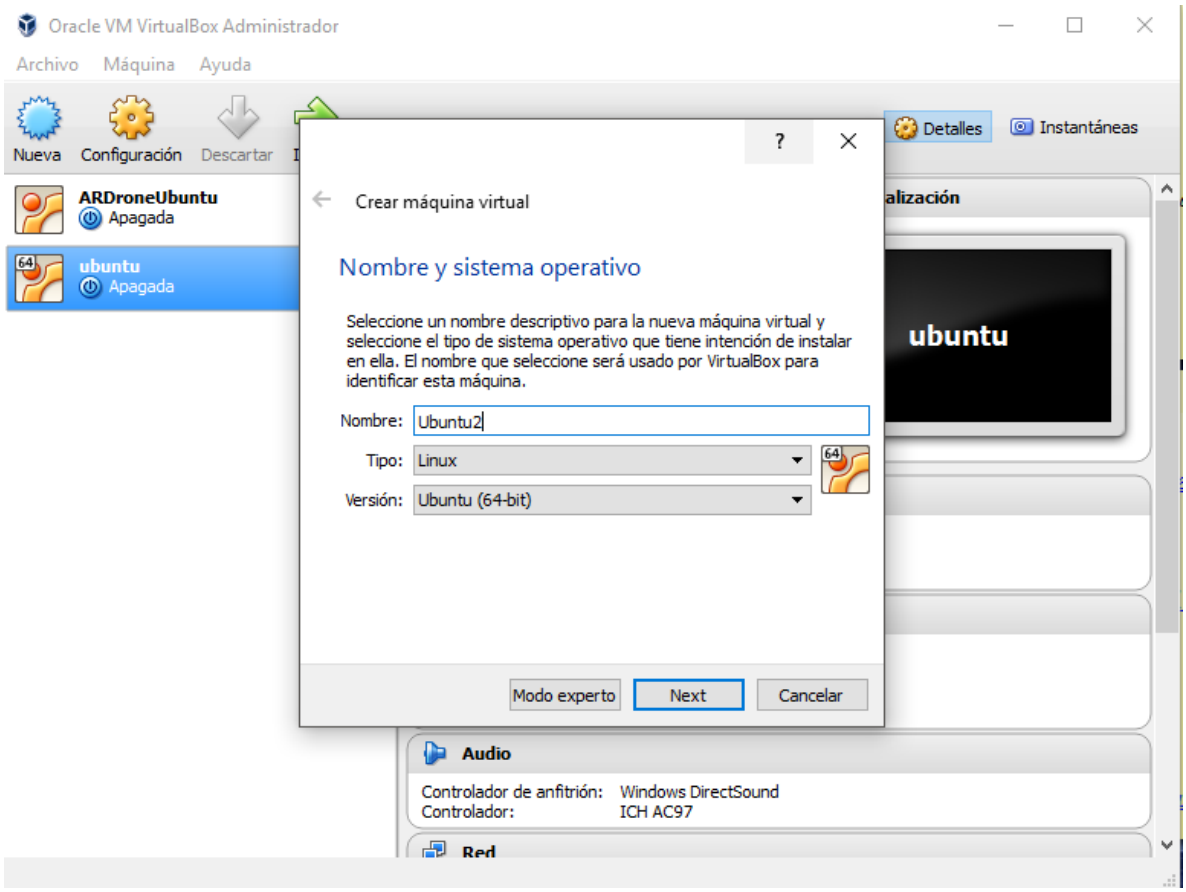
Además de haber descargado Ubuntu se debe de contar con una máquina virtual, en mi caso utilicé “Oracle VM VirtualBox” es una de las máquina virtuales más fácil de utilizar y con un entorno muy amigable.

Una vez descargado e instalado Virtual box se procede a ejecutar el programa y dar

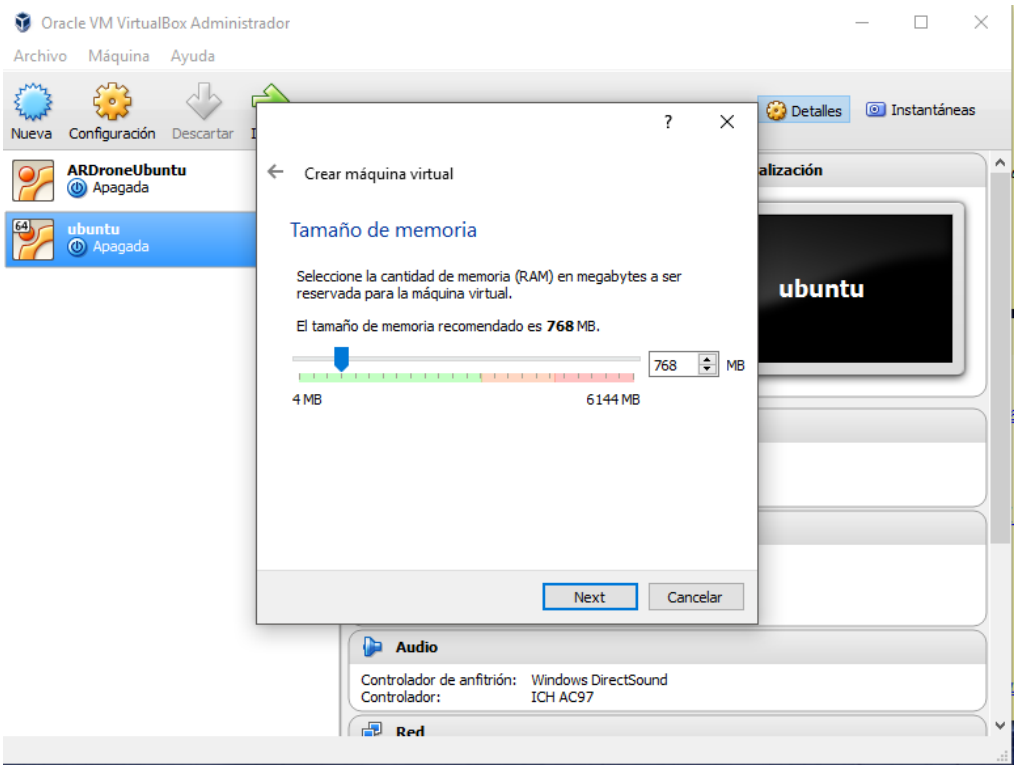


clic en el botón Nuevo.

Una vez que se da clic en “nueva” aparecerá una ventana en la cual escribirás el

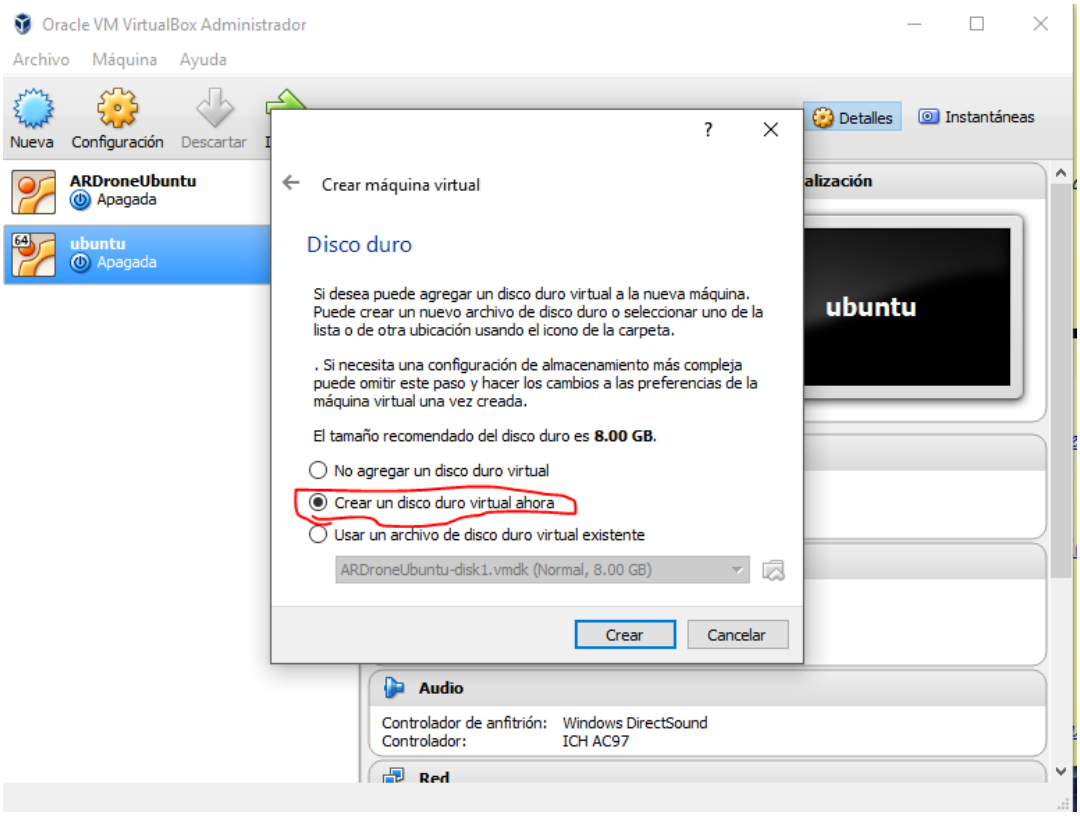


nombre del dispositivo, en este caso lo llamé “Ubuntu 2” debido a que ya cuento con una máquina virtual llamada Ubuntu. Damos “next”



Y aparecerá otra ventana en la cual daremos el tamaño de la memoria RAM que le daremos a la máquina virtual para que utilice.

Damos clic en “next” una vez asignada la memoria RAM a utilizar. Y aparecerá otra

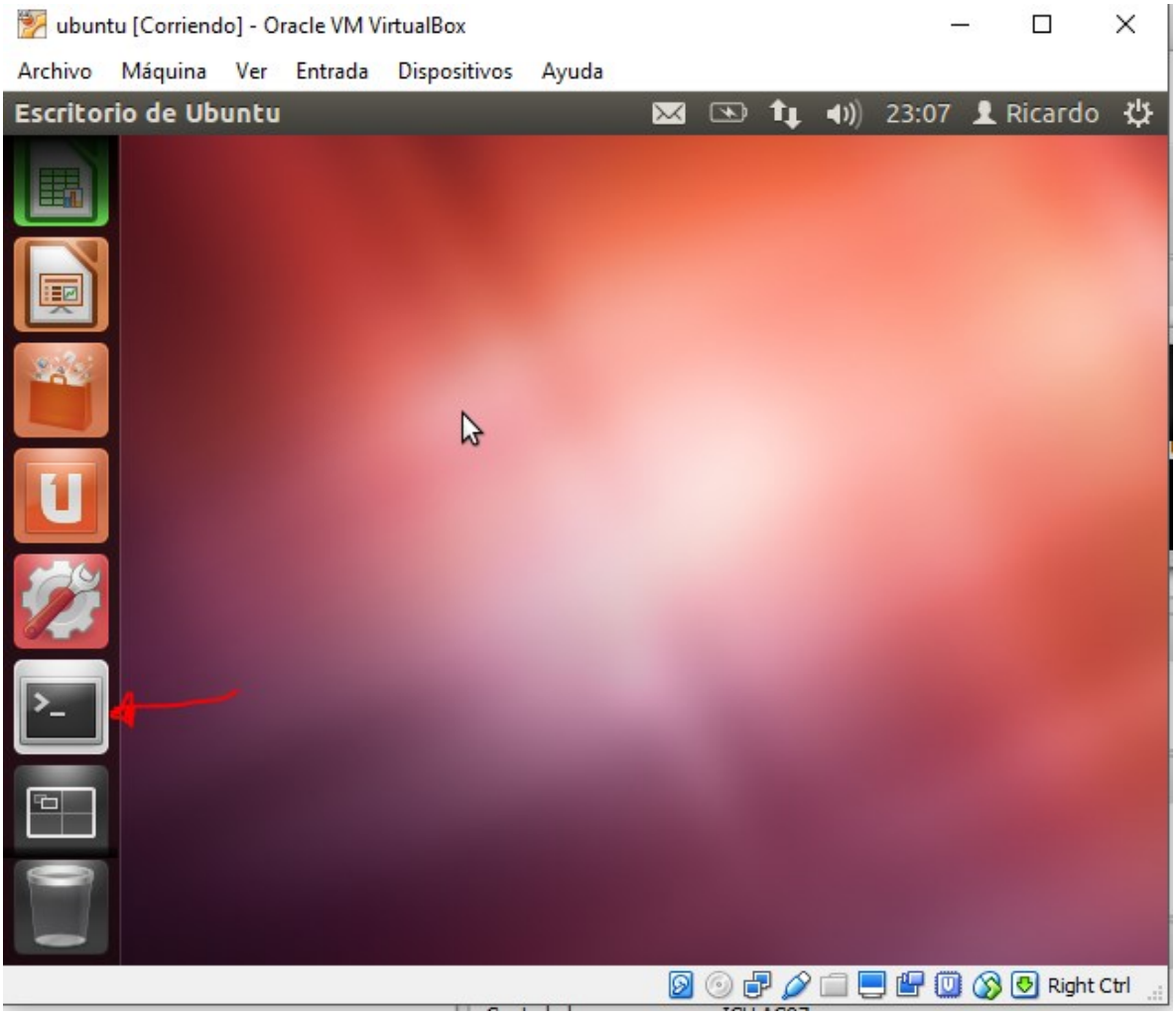


ventana en la cual elegiremos la segunda opción y daremos “next”.

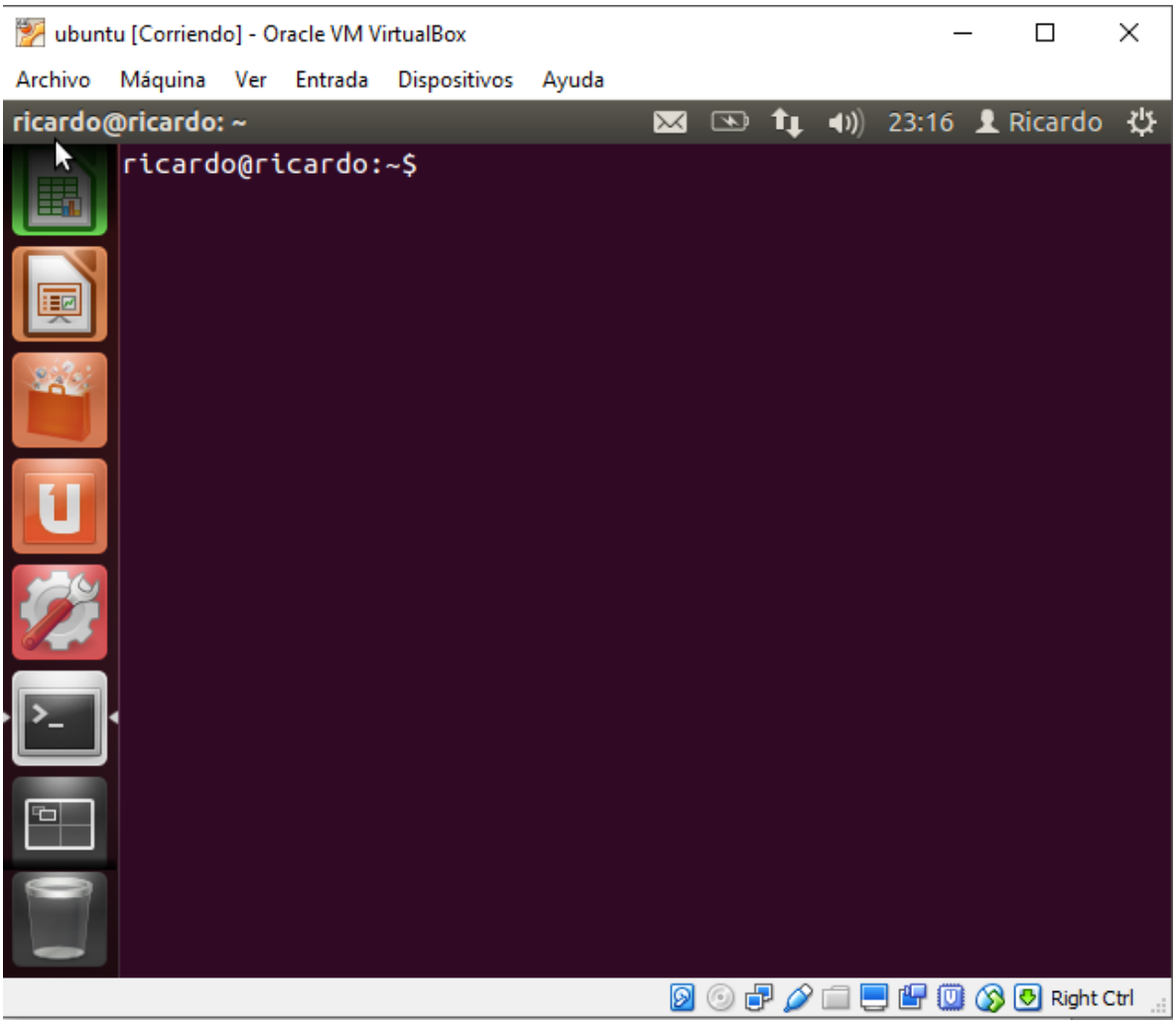
Una vez concluido esto comenzará a crearse el espacio en disco duro y se procederá a realizar el paso 4 de la instalación normal.

3.1.7.4 Instalación de las paqueterías dentro de Ubuntu.

Para la instalación de toda la paquetería siguiente se hará uso de la Terminal de Ubuntu.



Una vez encontrada la terminal, daremos clic y aparecerá una pantalla oscura donde estará el nombre que hayas puesto a la computadora seguido por un signo “$” ahí es donde se escribirán todos los comandos.



De aquí en adelante todo lo que esté escrito en color verde nos indicara que tenemos que escribirlo en la terminal. Se tiene que recordar que cada que se termine de escribir una línea de código es necesario pulsar la tecla “enter” y recordar quitar el signo de “$” que está escrito en este documento.

•Instalar Node.Js

Lo primero es actualizar los repositorios de Ubuntu y obtener los que nos permiten crear paquetes

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install build-essential –y
$ sudo apt-get install git -y
$ sudo apt-get install python libssl-dev -y
```

Una vez hecho esto, descargamos el script de instalación.

```
$ cd /usr/local/src
$ sudo mkdir node
$ cd node
$ sudo wget http://nodejs.org/dist/v0.6.17/node-v0.6.17.tar.gz
$ sudo tar -xzvf node-v0.6.17.tar.gz
```

Lo siguiente es entrar al directorio y ejecutar el archivo.

```
$ cd node-v0.6.17
$ sudo ./configure
```

Se procede a usar el comando make para terminar de instalar node.js

```
$ sudo make
$ sudo make install
```

Esto resultara que tanto node como npm se instalen en la dirección /usr/local/bin

•Instalar librería ar-drone

```
$ npm install git://github.com/felixge/node-ar-drone.git
$ npm install ar-drone
```

•Instalar librería ardrone-autonomy

```
$ sudo apt-get install git git-core
$ cd node_modules
$ git clone https://github.com/eschnou/ardrone-autonomy.git
```

•Instalar librería node-sylvester

```
$ npm install sylvester
```

•Instalar librería Async

$ npm install async

3.1.7.5 Como Programar.

Para escribir un programa nuevo es necesario seguir los siguientes pasos:

1. Conectarse a la red wifi del ardrone. Este procedimiento es igual que cuando nos queremos conectar a una red de internet.

2. Abrir la Terminal

3. Escribir en la pantalla lo siguiente:

$ nano Nombre del archivo.js

Este comando abre el editor de textos llamado nano y crea un archivo con el nombre que se halla escrito. En este editor se escribirá el programa. Si en algún momento se quisiera revisar un archivo que ya se escribió se puede utilizar este mismo código.

Es importante que el nombre del archivo lleve la terminación .js ya que si no la tiene después no podrá ser ejecutable.

4. Se procede a escribir en el editor de texto el programa. Cuando se haya terminado de escribir el programa se debe presionar las teclas CTRL+X, nos preguntara si queremos guardar el archivo y presionaremos la tecla Y, des pues preguntara si quieres guardarlo con el nombre que se había escrito anteriormente, presionaremos la tecla ENTER.

5. Para ejecutar el programa escribiremos el siguiente código:

$ node Nombre del archivo.js

Si todo salió bien el ardron debería de empezar a realizar la rutina que se halla programado. Es importante recordar que cuando se programa utilizando este método los errores de compilación se arrojaran en la pantalla de la terminal.

3.1.7.5.1 Compendio de instrucciones

Instrucciones de la biblioteca ar-drone

arDrone.createClient([opciones])

Regresa un Nuevo objeto de ciente. Opciones incluye

- IP: La IP del Drone. Por defecto es '192.168.1.1'.
- frameRate: La velocidad de fotogramas PngEncoder. Por defecto es 5.
- imageSize: El tamaño de la imagen PngEncoder. Por defecto es nula.

- **client.createREPL()**

Ejecuta una interfaz interactiva con todos los métodos de cliente disponibles en el ámbito active. Additionallyclient resuelve al cliente instantaneamente.

Client.getPngStream()

Regresa un objeto PngEncoder que emite individuales imágenes png depurándolos como "datos" de evento. Hace múltiples llamados a este método regresando el mismo objeto.

client.getVideoStream()

Regresa un objeto TcpVideoStream que emite paquetes tcp como eventos de

client.takeoff(callback)

Inicia el vuelo del Drone.

client.land(callback)

Baja al suelo el Drone.

client.up(speed) / client.down(speed)

Hace ganar altitud o disminuirla. Speed puede ser un valor entre 0 y 1

client.clockwise(speed) / client.counterClockwise(speed)

Causa que el Drone gire. Speed puede ser un valor entre 0 y 1

client.front(speed) / client.back(speed)

Controla el Pitch, con un movimiento horizontal usando la cámara como punto de referencia. Speed puede ser un valor entre 0 y 1.

client.left(speed) / client.right(speed)

Controla el Roll, speed puede ser un valor entre 0 y 1.

client.stop()

Sirve como función de paro de emergencia.

client.calibrate(device_num)

Pide al drone que calibre el dispositivo. Actualmente el firmware del ARDrone soporta un solo dispositivo que puede ser calibrado: El magnetómetro, que es el dispositivo 0.

El magnetómetro puede solo ser calibrado mientras el drone está volando, y la rutina de calibración causa que el drone de una vuelta de 360 grados en su eje.

Instrucciones de la biblioteca ardrone-autonomy

mission.log(path)

Carga los datos de la misión, Borra CSV en el archio dado. Hace que sea realmente útil para depurar el comportamiento de los estados del controlador.

mission.run(callback)

Ejecuta la mission. El llamado tiene la forma de function(err,result) y puede ser desencadenada en caso de error o que finalice la misión.

mission.takeoff()

Hace que vuele el Drone.

mission.forward/backward/left/right/up/down(distance)

Agrega un movimiento a la misión. El drone podrá moverse en la dirección dada siempre y cuando la distancia está escrita en metros, antes de proceder al siguiente paso. El drone podría también intentar mantener todos los demás grados de libertad.

mission.altitude(height)

Agrega un paso de altitud a la misión. Deberá tener que subir antes de hacer el siguiente paso.

mission.cw/ccw(angle)

Agrega un paso de rotación a la misión. Deberá girar un cierto Angulo dado (in Deg) antes de proceder al siguiente paso.

mission.hover(delay)

Agrega un retardo a la misión. El Delay deberá ser en ms.

mission.wait(delay)

Agrega una espera a la misión en ms

mission.go(position)

Agrega un movimiento a la misión. Deberá ir a la posición dada antes de proceder al siguiente paso. La posición es una metal de control tal como {x: 0, y: 0, z:1, yaw:90}.

mission.task(function(callback){..})

Agrega una tarea a la misión. Deberá ser ejecutada previendo antes de que proceda a la siguiente función. Un argumento de callback es pasado a la función, esto deberá ser llamado cuando la tarea haya terminado.

mission.taskSync(function)

Add a task step to the mission. Will execute the provided function before proceeding to the next step.

mission.zero()

Crea un punto cero al cual regresar.

3.1.8 Programación de trayectorias.

En esta sección se encontrará la programación realizada a lo largo de este trabajo de tesis, el cual cuenta con una trayectoria circular, una trayectoria cuadrada, un rombo, trayectoria triangular y un programa para efectuar un paro de emergencia.

Una vez realizados los pasos anteriores lo siguiente es escribir código.

- **Insertar comandos.**

```
var autonomy = require('ardrone-autonomy'); //Declara librerías.
var mission = autonomy.createMission();
const readline = require('readline');

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

do{ //crea un ciclo infinito.
        console.log('\nQue quieres hacer?\n'); //despliega mensajes de opciones.
        console.log('volar');
        console.log('adelante');
        console.log('atras');
        console.log('derecha');
        console.log('izquierda');
        console.log('aterrizar');
        console.log('salir');

        rl.question('\n--------------\n\n', (answer) => {

        if(answer == 'volar'){ //condiciones  las cuales se cumplen dependiendo la opción
                console.log('\nvolar');
                mission.takeoff();
        }
        if(answer == 'adelante'){
                console.log('\nadelante');
                mission.go({x:1,y:0,z:0});
        }
        if(answer == 'atras'){
                console.log('\natras');
                mission.go({x:-1,y:0,z:0});
        }
```

```
            if(answer == 'derecha'){
                    console.log('\nderecha');
                    mission.go({x:0,y:1,z:0});
            }
            if(answer == 'izquierda'){
                    console.log('\nizquierda');
                    mission.go({x:0,y:1,z:0});
            }
            if(answer == 'aterrizar'){
                    console.log('\naterrizar');
                    mission.hover(1000);
                    mission.land();
            }

            mission.run(function (err, result) {

    if (err) {
       console.trace("\nalgo malo pasó: %s", err.message);
       mission.client().stop();
       mission.client().land();
    }
    else {
       console.log("\nMision exitosa!");
       process.exit(0);
          }
          });

          });
}
while (answer =! 'salir'){//termina el ciclo infiito

}

//finaliza el programa.
```

- **Trayectoria circular.**

```
var autonomy = require('ardrone-autonomy');  // Se declara la variable Autonomy y se le asigna la librería “ardrone-autonomy”
var mission = autonomy.createMission(); //Se declara la variable mission y se le asigna la función autonomy.createMission();
mission.takeoff() //Despega el drone

mission.zero()//Marca donde despegó como inicio la posición (0,0)

//Asigna una serie de puntos en “x,y,z” los cuales son representados en metros

 .go({x: 0.5, y: 0.666, z: 0})
 .go({x: 1, y: 1, z: 0})
 .go({x: 1.5, y: 0.666, z: 0})
 .go({x: 2, y: 0, z: 0})
 .go({x: 1.5, y: -0.666, z: 0})
 .go({x: 1, y: -1, z: 0})
 .go({x: 0.5, y: -0.666, z: 0})
 .go({x: 0, y: 0, z: 0})

.hover(1000)  // permanece 1 segundos en el aire
.land(); //aterriza ;

//arroja una mensaje cuando se complete la misión u ocurrió un problema.

mission.run(function (err, result) {

if (err) {

        console.trace("algo malo pasó: %s", err.message);
        mission.client().stop();
        mission.client().land();
}

else {
        console.log("Mision exitosa!");
        process.exit(0);
}
});


//finaliza el programa
```

- **Cuadrado 1x1**

```
Var autonomy = require('ardrone-autonomy');

Var mission = autonomy.createMission();

Mission.takeoff()

        .zero()       // Posiciona el AR drone en el punto de referencia 0,0
        .altitude(1)  // Asigna una altitud de 1 metro
        .forward(1)   // avanza 1 metro al frente
        .right(1)     // avanza 1 metro a la derecha
        .backward(1)  // retrocede 1 metro
        .left(1)      // va a la izquierda 1 metro
        .hover(1000)  // Se queda estático 1 segundo
        .land();      // apaga motores y desciende

//arroja una mensaje cuando se complete la misión u ocurrió un problema.

mission.run(function (err, result) {

if (err) {
        console.trace("algo malo pasó: %s", err.message);
        mission.client().stop();
        mission.client().land();     }
else {
        console.log("Mision exitosa!");
        process.exit(0);
}
});

//finaliza el programa
```

- **Trayectoria de rombo**

```
var autonomy = require('ardrone-autonomy');  // Se declara la variable Autonomy y se le asigna la librería “ardrone-autonomy”.

var mission = autonomy.createMission(); //Se declara la variable mission y se le asigna la función autonomy.createMission();

mission.takeoff() //Despega el drone

mission.zero()//Marca donde despegó como inicio la posición (0,0)

//Asigna una serie de puntos en “x,y,z” los cuales son representados en metros

        .go({x: 1, y: 0, z: 0})
        .go({x: 0.5, y: 0.5, z: 0})
        .go({x: -0.5, y: 0.5, z: 0})
        .go({x: -1, y: 0, z: 0})
        .go({x: -0.5, y: -0.5, z: 0})
        .go({x: 0, y: 0, z: 0})
        .hover(1000)   // Se queda estatico 1 segundo
        .land(); // apaga motores y deciende


//arroja una mensaje cuando se complete la misión u ocurrió un problema.

mission.run(function (err, result) {

if (err) {
        console.trace("algo malo pasó: %s", err.message);
        mission.client().stop();
        mission.client().land();     }
else {
        console.log("Mision exitosa!");
        process.exit(0);
}
});

//finaliza el programa
```

- **Trayectoria triangular.**

```
var autonomy = require('ardrone-autonomy');   // Se declara la variable Autonomy y se le asigna la librería “ardrone-autonomy”.

var mission = autonomy.createMission(); //Se declara la variable mission y se le asigna la función autonomy.createMission();

mission.takeoff() //Despega el drone

        mission.zero()//Marca donde despegó como inicio la posición (0,0)
        .altitude(1.5) //Alza el drone a 1.5 metros del suelo
//Asigna una serie de puntos en “x,y,z” los cuales son representados en metros

         .go({x: 0, y: 0, z: 0}) //Se asegura de que se encuentre en el origen.

        .go({x: 0.5, y: 0.5, z: 0})
        .go({x: 1, y: -0.5, z: 0})
        .go({x: 1.5, y: 0.5, z: 0})
        .go({x: 2, y: -0.5, z: 0}) .go({x: 0, y: 0, z: 0})

        .hover(1000)   // Se queda estatico 1 segundo
        .land(); // apaga motores y deciende


//arroja una mensaje cuando se complete la misión u ocurrió un problema.

mission.run(function (err, result) {

if (err) {
        console.trace("algo malo pasó: %s", err.message);
        mission.client().stop();
        mission.client().land();     }
else {
        console.log("Mision exitosa!");
        process.exit(0);
}
});

//finaliza el programa
```

- **Trayectoria cuadrada.**

```
var autonomy = require('ardrone-autonomy');  // Se declara la variable Autonomy y se le asigna la librería "ardrone-autonomy".

var mission = autonomy.createMission(); //Se declara la variable mission y se le asigna la función autonomy.createMission();

mission.takeoff() //Despega el drone

        mission.zero()//Marca donde despegó como inicio la posición (0,0)
        .altitude(1) //Alza el drone a 1.5 metros del suelo
//Asigna una serie de puntos en "x,y,z" los cuales son representados en metros

        .go({x: 0.5, y: 0, z: 0})
        .go({x: 0.5, y: -0.5, z: 0})
        .go({x: 1, y: -0.5, z: 0})
        .go({x: 1, y: 0, z: 0})
        .go({x: 1.5, y: 0, z: 0})
        .go({x: 1.5, y: -0.5, z: 0})
        .go({x: 2, y: -0.5, z: 0})
        .go({x: 2, y: 0, z: 0})
        .go({x: 0, y: 0, z: 0})

        .hover(1000)   // Se queda estatico 1 segundo
        .land(); // apaga motores y deciende


//arroja una mensaje cuando se complete la misión u ocurrió un problema.

mission.run(function (err, result) {

if (err) {
        console.trace("algo malo pasó: %s", err.message);
        mission.client().stop();
        mission.client().land();     }
else {
        console.log("Mision exitosa!");
        process.exit(0);
}
});

//finaliza el programa
```

- **Paro de emergencia.**

```
var autonomy = require('ardrone-autonomy');
 var mission = autonomy.createMission();
mission.stop();
mission.land();
```

3.2Otra alternativa.

3.2.1 Instalar ROS (Robotic Operative Sistem) y ARdroneXubuntu.

ARDroneXUbuntu es un Sistema operative especialmente diseñado con todas las librerías y varios ejemplos los cuales están enfocados específicamente al ARDrone de Parrot, lo cual es una gran ventaja ya que te ahorra toda la instalación de paqueterías después de instalar el sistema operativo.

La forma en la que se ejecuta este S.O. es mediante una máquina virtual, de nueva cuenta se utiliza virtual box para instalar dicho S.O.

1.- Se descarga ArDroneXUbuntu desde el siguiente link.

http://bit.ly/VqtDql

2.- Una vez descargado el sistema operativo se da doble click en *"ARDroneUbuntu.ova"* y si tienes instalado VirtualBox automáticamente se abrirá el instalador, después de eso solo hay que dar a siguiente a todo, dejar todo configurado tal como está y una vez instalado iniciar la máquina virtual.

Lo primero que se debe de hacer una vez dentro de "ARDRONUBUNTU" es actualizar todos los repositorios lo cual se puede realizar por medio de comandos dentro de la terminal.

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

1.- Terminal

2.- Interfaz utilizada para programar

3.- Aplicación de vuelo.

4.- Actualizar paqueterías (se debe de realizar esto después de actualizar los repositorios).

5.- Barra de búsqueda.

Lamentablemente por la complejidad del código se me hizo muy difícil aprenderlo y realizar aplicaciones básicas con las cuales hacer algo más robusto, únicamente utilicé las aplicaciones que venían de ejemplo dentro del sistema operativo las cuales son muy interesantes a la hora de utilizarlas.

Capítulo 4

4.1 Conclusiones.

Se puede concluir que el Dron realiza la tarea que se le asigna, con el inconveniente qué tiene un tiempo de respuesta a cada instrucción, lo cual hace lento el proceso.

Parrot ha desarrollado muy buenos Drones los cuales son muy útiles, el único problema que puede existir es la estabilidad con la cual vuela el Drone, debido a qué al asignarle un punto si es movido de su lugar, se pierde la total estabilidad del Drone y comienza a responder de manera equivoca.

La programación básica es sencilla, una vez entendido como funcionan programar algunas trayectorias se torna muy sencillo. Por otra parte al querer seguir programando más funciones o utilizar otra forma de programación para aplicaciones más robustas lo torna muy complicado ya que es solo información que se puede conseguir en cursos de paga en otros países.

Python y Node.js junto con Linux es la forma más sencilla de programar los cuadricopteros, al utilizar SDK con Python, el nivel de programación se incrementa pero obtienes mejores resultados.

4. 2 Trabajo a futuro

Como trabajo a futuro se planea la realización de comunicación entre dos drones para así formar maestro-esclavo y aplicar las señales correspondientes y que hagan una tarea determinada.

Al igual se planea comenzar a emplear sistemas caóticos digitales para comenzar a trabajar con caos y señales encriptadas en Drones.

Del mismo modo implementar simulaciones realizadas en Matlab para realizar tareas de patrullaje de un área determinada.

Por último, realizar formación de un grupo de Drones para con ello patrullar un espacio mayor y que se comuniquen entre ellos.

4.3 Bibliografía.

[1] Avistar, Dialogo, "Historia de los cuadricopteros", [Online], Disponible: http://www.aviastar.org/helicopters_eng/bothezat.php

[2] Arducopter, Dialogo, "Tipos de Drones" [online] Disponible: http://cuadricopteros.org/cuadricopteros/

[3] R. MAYORGA, "Sistema de navegación para vehículos Aéreos Cuadricópteros", Proyecto de titulación en ingeniería técnica aeronáutica, Universidad Politecnica Superior de Cataluña. España 2009.

[4] Donweb, Dialogo, "14 usos de Drones", [Online] Disponible: http://agencia.donweb.com/los-14-usos-de-drones-que-seguro-no-conocias/print/

[5] The Forum of the Americas, Diálogo, "Vehículos aéreos no tripulados para vigilar las costas ecuatorianas", [Online]. Disponible: http://dialogo-americas.com/en_GB/articles/rmisa/features/regional_news/2011/10/17/ecuador-unmanned-aircraft

[6] Nadales C. "Control de un quadrotor mediante la plataforma arduino". Proyecto de titulación en ingeniería Técnica de telecomunicaciones, Universidad Politecnica superior de Cataluña. España 2009

[7] Heberto Molina, "ArDrone" Trabajo de investigación Ingeniería en Electrónica, Universidad Autónoma de Baja california, México 2015

[8] LinuxZone, Dialogo, "Descripción de Ubuntu, Descarga y caracateristicas", [Online] Disponible: http://linuxzone.es/distribuciones-principales/ubuntu/

[9] Carlos Fioriti, Dalogo, "Como instalar Node.js y npm en Ubuntu y derivados", [Online] Disponible: https://fioriticarlos.wordpress.com/2014/05/20/instalar-nodejs-y-npm-en-ubuntu/

[10] JdeRobot, , Dialogo, "Programación de Drones", [Online] Disponible: http://jderobot.org/Programacion-de-drones

[11] A. Barrietos, J del Cerro, R. San Martín, C. Rossi, "Vehículos aéreos no tripulados par auso civil", Proyecto de titulación, Universidad Politécnica de Madrid. España. 2012

[12] Brito J. Callou F. "The Navigation and Control Technology inside the AR.Drone micro UAV", [Online]. Disponible: http://cas.ensmp.fr/~petit/papers/ifac11//PJB.pdf

Parrot, "Especificaciones cuadricoptero Parrot A.R. Drone" [Online]. Disponible: http://www.parrot.com