SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

SUBSECRETARÍA DE EDUCACIÓN SUPERIOR
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TUXTLA GUTIÉRREZ

REPORTE DE RESIDENCIA PROFESIONAL

INSTRUMENTACIÓN Y CONTROL DE UN HELICÓPTERO TIPO TANDEM DE 3 GRADOS DE LIBERTAD

Meza López Paulo Ángel
Velázquez Matias Emmanuel

Tuxtla Gutiérrez, Chipas , México
AGOSTO-DICIEMBRE 2017

Tabla de contenido

INTRODUCCIÓN.

Los principios de vuelo y construcción de aviones y helicópteros siempre han sido de gran interés para el hombre. Con el reciente avance tecnológico de las computadoras y el desarrollo de la moderna teoría de control, los sistemas de control automático son un factor importante en el desarrollo de casi todos los nuevos vehículos voladores.

Los vehículos aéreos no tripulados (VANT) han sido un tema de investigación y desarrollo muy activo durante los últimos años [Carrillo et al., 2012]. Pueden ser controlados remotamente o volar de forma autónoma basado en planes de vuelo programados mediante el uso de sistemas de control complejos. Poseen características únicas como: tamaño pequeño, gran maniobrabilidad y bajo precio; haciéndolos atractivos para diferentes tipos de actividades. En los últimos años se ha extendido su uso en aplicaciones civiles, por ejemplo, para la detección y monitoreo de incendios en bosques y ciudades [Salvo et al., 2014], inspección en la construcción de infraestructuras [Roca et al., 2014], toma de mediciones atmosféricas e inspección de líneas de alta tensión.

OBJETIVOS.

Objetivo General:

Implementar un controlador PID para el posicionamiento adecuado de un Helicóptero tipo tándem de 3 grados de libertad montado sobre una plataforma fija.

Objetivos Específicos:

1. Establecer el modelo matemático adecuado que nos permita una descripción viable de la dinámica del sistema del Helicóptero tipo tándem de 3 grados de libertad.
2. Realizar simulaciones del modelo matemático a través de simulaciones en Matlab utilizando simmechanics.
3. Establecer la sintonización adecuada del controlador PID para el control de posición angular de los 3 grados de libertad del sistema y probarlo en Matlab.
4. Instrumentar e implementar un helicóptero tipo tándem sobre una plataforma 3DOF.
5. Implementar el controlador físicamente a través de la tarjeta Navio 2 y raspberry pi 3 acoplados a la plataforma del helicóptero tipo tándem.

1.- FUNDAMENTO TEÓRICO.

1.1.- Modelado matemático

En esta sección se explica el modelo matemático utilizado a lo largo del proyecto de investigación, apoyado de teorías establecidas en el campo de control. Se definen las primeras longitudes, fuerzas, ángulos y pares que actúan sobre el helicóptero durante su funcionamiento. Esto es seguido por un análisis de diagrama de cuerpo libre (FBD) usando la segunda ley de movimiento de Euler. Las ecuaciones que describen el sistema encontrado se transforman a continuación en un modelo de espacio de estados [1].

Los helicópteros con rotores en tándem tienen dos rotores montados uno en frente del otro (a distintas alturas) en el sentido longitudinal del helicóptero, es decir, en tándem. Esta configuración se suele usar en helicópteros de transporte.

Figura 1: "Helicóptero tipo tándem"

En la figura 2 se muestra un esquema básico del sistema de un helicóptero.

1.1.1.- Sistema de coordenadas:

Se ha introducido un sistema de coordenadas de acuerdo con la figura 2. El sistema gira con el equipo alrededor de los ejes de desplazamiento y elevación. La plataforma se fija en el montaje en el punto O. Cuando se describe la posición del helicóptero, se obtienen tres ángulos. Estos se definen como:

•Ángulo pitch. El ángulo que gira el helicóptero alrededor del eje que atraviesa la articulación. Definido como cero cuando el helicóptero está horizontal.

•Ángulo de elevación. El ángulo entre el plano XY y el brazo. Definido como cero cuando el brazo está horizontal.

•Ángulo de desplazamiento. El ángulo que gira la plataforma alrededor del eje T. Definido como cero en el ángulo de inicialización del sensor.

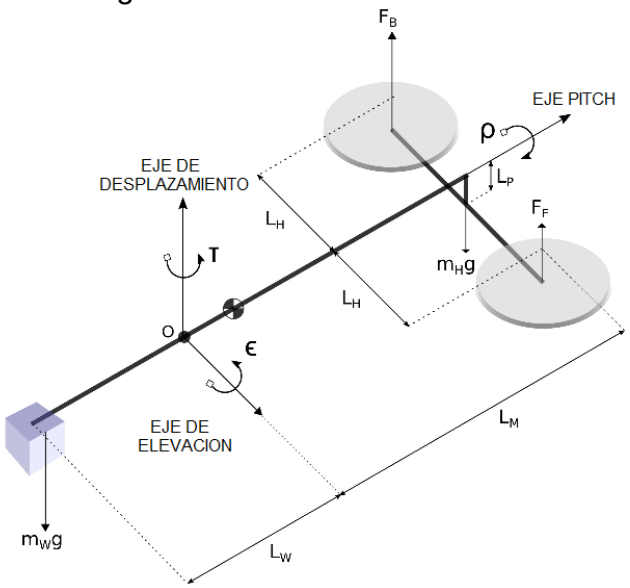

Figura 2: "Modelo de un Helicóptero tándem"

1.1.2.- Distancias:

Las distancias en la figura 2 se definen como:

• L_W: La distancia entre el eje de elevación y el centro de gravedad del contrapeso.
• L_M: La distancia entre el eje de elevación y el helicóptero.
• L_H: La distancia entre el eje de cabeceo y el centro de las hélices.
• $L_ρ$: La distancia entre el eje de cabeceo y el centro de gravedad del helicóptero.

1.1.3.- Masas:

Las masas se definen como:

• m_W: masa del contrapeso.
• m_H: La masa del helicóptero (incluidos los motores, etc.).

1.1.4.- Inercias:

Las inercias se definen como:

• I_ρ: La inercia de rotación alrededor del eje de paso.
• I_ϵ: La inercia rotacional alrededor del eje de elevación.
• I_T: La inercia rotacional alrededor del eje de desplazamiento.

Además, las fuerzas F_F y F_B representan el empuje generado por el motor delantero y trasero respectivamente.

1.2.- Ecuaciones de movimiento.

Para identificar las ecuaciones gobernantes del sistema de un helicóptero, se debe realizar un análisis de diagrama de cuerpo libre en cada uno de los ejes.

1.2.1.- Eje pitch.

En la figura 3 se muestra las fuerzas que actúan sobre el eje de cabeceo o inclinación. Desde la figura, se puede ver fácilmente que el par principal alrededor del eje de cabeceo proviene del empuje generado por los motores. Cuando ρ 6 ≠ 0, la fuerza gravitacional del conjunto producirá también un par alrededor del eje de cabeceo. La fricción de la junta y la resistencia del aire se combinan en un único componente de par M_p. La segunda ley de Euler produce:

$$I_\rho \ddot{\rho} = (F_B - F_F)L_H - m_H g L_\rho sin\rho cos\epsilon - M_{F_\rho} \quad (1)$$

En la ecuación 1 se ha despreciado la fuerza de Coriolis.

1.2.2.- Eje de elevación

En la figura 3, se muestran las fuerzas que actúan en el eje de elevación. El eje de cabeceo ha sido rotado un ángulo de ρ. M_G denota el momento gravitacional. La fricción de la junta y la resistencia del aire se combinan en un único componente de par M_F. Sin embargo, debido a que la plataforma puede estar girando alrededor de los ejes de desplazamiento y elevación, al mismo tiempo, el eje de elevación no puede analizarse de forma aislada.

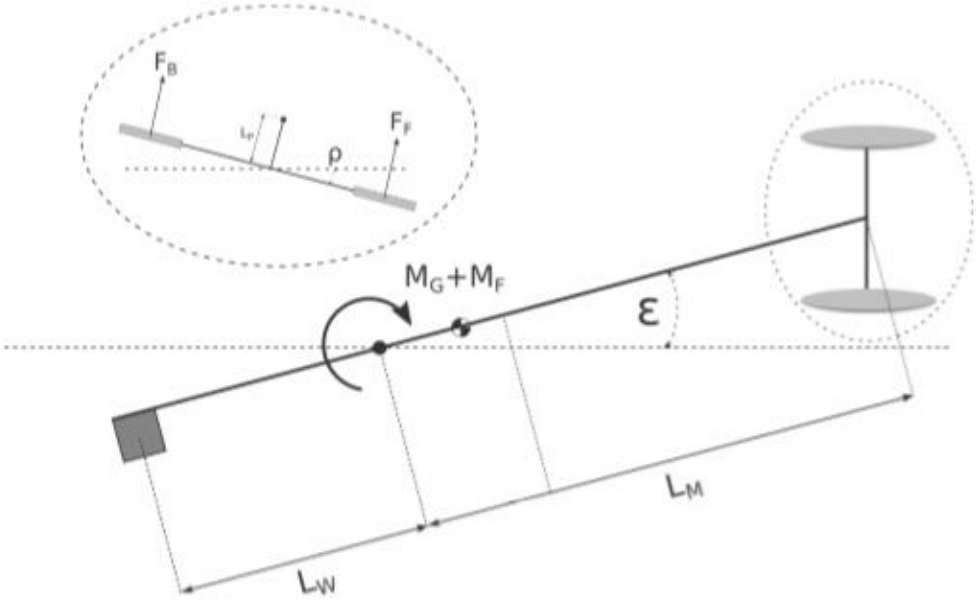

Figura 3: "Eje de elevación FBD. Eje de elevación que señala fuera de la página"

Se introduce un nuevo sistema de coordenadas que se fija en el helicóptero de acuerdo con la figura 4.El sistema de coordenadas local $\in$,ρ, τ' (rojo) se fija en la plataforma. El sistema global X, Y, T (negro) es el marco de referencia principal. El sistema de coordinación local comparte el O con el marco de referencia global (inercial) y gira con el equipo alrededor de los ejes de desplazamiento y elevación.

La segunda declaración de Euler sobre O yields (h_0 que denota un momento angular en el punto O):

$$\Sigma\vec{M}_0 = \frac{d\vec{h_0}}{dt}$$

Dado que el sistema de coordenadas fijado en el helicóptero está girando en comparación con el sistema inercial global, la derivada del momento angular viene dada por:

$$\frac{d\vec{h_0}}{dt} = \left(\frac{d\vec{h_0}}{dt}\right)_{\epsilon\rho\tau'} + \vec{\Omega} \times \vec{h_0}$$

Donde la derivada se evalúa en el sistema de coordenadas local y $\vec{\Omega}$ es el vector de velocidad angular que describe cómo gira el sistema local en comparación con el global. Además $\vec{h_0}$ está dado por:

$$\vec{h_0} = I_0\vec{\Omega} = \begin{bmatrix} I_\epsilon & 0 & 0 \\ 0 & I_\rho & 0 \\ 0 & 0 & I_\tau \end{bmatrix} \begin{pmatrix} \dot{\epsilon} \\ \dot{\tau}\sin\epsilon \\ \dot{\tau}\cos\epsilon \end{pmatrix} = \begin{pmatrix} I_\epsilon\dot{\epsilon} \\ I_\rho\dot{\tau}\sin\epsilon \\ I_\tau\dot{\tau}\cos\epsilon \end{pmatrix}$$

Esto da:

$$\left(\frac{d\vec{h_0}}{dt}\right)_{\epsilon\rho\tau'} = \begin{pmatrix} I_\epsilon\ddot{\epsilon} \\ I_\rho(\ddot{\tau}\sin\epsilon + \dot{\tau}\dot{\epsilon}\cos\epsilon) \\ I_\tau(\ddot{\tau}\cos\epsilon - \dot{\tau}\dot{\epsilon}\sin\epsilon) \end{pmatrix}$$

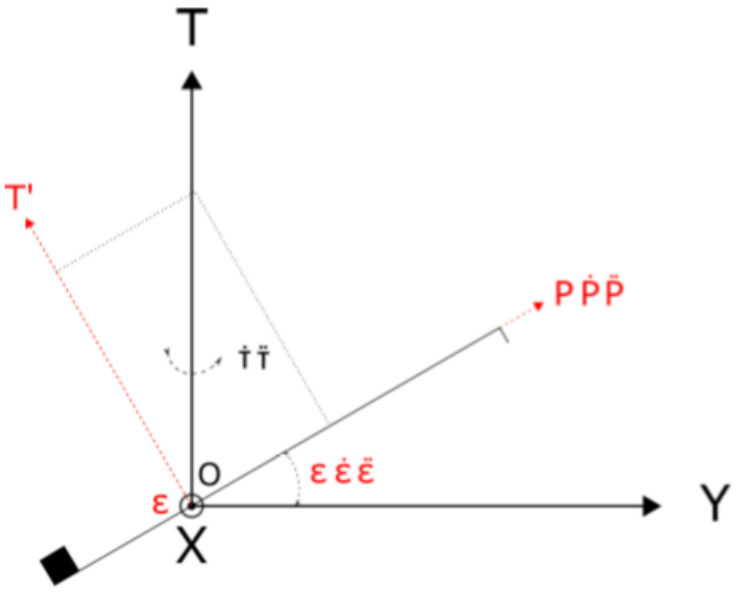

Figura 4: "Sistema de coordenadas local (rojo) y global (negro)"

Seguido del producto punto:

$$\vec{\Omega} \times \vec{h_0} = \begin{pmatrix} (I_\tau - I_\rho)\dot{\tau}\cos\epsilon\dot{\tau}\sin\epsilon \\ (I_\epsilon - I_\tau)\dot{\epsilon}\dot{\tau}\cos\epsilon \\ (I_\rho - I_\epsilon)\dot{\epsilon}\dot{\tau}\sin\epsilon \end{pmatrix}$$

Lo que da el cambio en el momento angular como:

$$\frac{d\vec{h_0}}{dt} = \begin{pmatrix} I_\epsilon \ddot{\epsilon} + (I_\tau - I_\rho)\dot{\tau}^2 \cos\epsilon \sin\epsilon \\ I_\rho \ddot{\tau} \sin\epsilon + (I_\epsilon + I_\rho - I_\tau)\dot{\epsilon}\dot{\tau}\cos\epsilon \\ I_\tau \ddot{\tau}\cos\epsilon - (I_\tau + I_\epsilon - I_\rho)\dot{\epsilon}\dot{\tau}\sin\epsilon \end{pmatrix}$$

El cambio en el momento angular sobre el eje de elevación ahora se identifica como:

$$\frac{dh_{0,\epsilon}}{dt} = I_\epsilon \ddot{\epsilon} + (I_\tau - I_\rho)\dot{\tau}^2 \cos\epsilon \sin\epsilon \qquad (2)$$

Donde el término adicional $(I_\tau - I_\rho)\dot{\tau}^2$ representa el torque centrífugo. El uso de la ecuación 2 junto con la segunda ley de Euler da:

$$I_\epsilon \ddot{\epsilon} + (I_\tau - I_\rho)\dot{\tau}^2 \cos\epsilon \sin\epsilon = (F_F + F_B) L_M \cos\rho - M_G - M_{F,\epsilon} \qquad (3)$$

En la ecuación 3 los pares generados por los motores están descuidados. Son proporcionales a la velocidad de rotación al cuadrado y posiblemente pueden ser bastante significativos. Sin embargo, los motores en nuestro caso están girando en direcciones opuestas contrarrestando entre sí. Claramente, seguirán generando un par neto al girar a diferentes velocidades (cuando se cambia el paso), pero para simplificar este efecto se descuida.

Además, las inercias alrededor del eje de desplazamiento y cabeceo usados en la ecuación 3 son las inercias alrededor de cada eje como se define en la figura 1. Al analizar el eje de elevación, la inercia alrededor del eje de desplazamiento debe compensarse por el hecho de que el brazo se ha separado del soporte mientras que la inercia del tono debe ser compensada por la inercia del brazo más contrapeso y la distancia entre el eje de cabeceo y el eje que atraviesa el brazo. Sin embargo, la diferencia es pequeña, por lo que para evitar tener que introducir nuevas variables, se omite el error.

1.2.3.- Eje de viaje

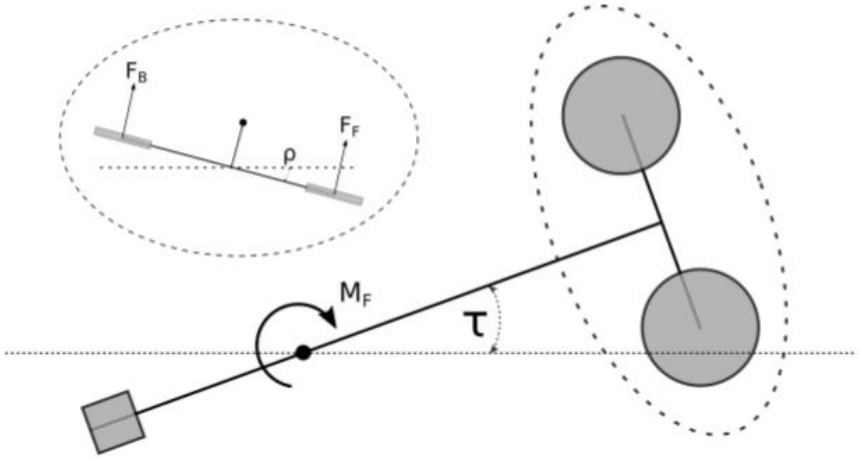

Figura 5: "Eje de desplazamiento FBD. Eje de viaje saliendo de la página"

Las fuerzas principales que actúan en el eje de desplazamiento son el empuje generado por los dos motores. Cuando el ángulo de inclinación es distinto de cero, las fuerzas de empuje tendrán un componente que generará un par alrededor del eje de desplazamiento. El componente aumentará con el aumento de ρ y también disminuirá con un mayor ángulo de elevación.

Contrariamente a las inercias de paso y elevación, la inercia de viaje variará de manera significativa a medida que el helicóptero se mueva. Aproximando el helicóptero y el contrapeso como masas puntuales e ignorando el brazo, la inercia de desplazamiento se puede expresar de acuerdo con la ecuación 4:

$$I_\tau = \cos^2(\epsilon)(m_W L_W^2 + m_H L_M^2) \tag{4}$$

El factor $\cos^2 \epsilon$ causará una reducción bastante significativa (comparada a $\epsilon = 0$) de la inercia incluso para ángulos relativamente pequeños. Para resaltar el hecho de que la inercia del recorrido no es constante sino más bien una función del ángulo de elevación, se expresará como $I_\tau = I_\tau(\epsilon)$. La figura 5 y la segunda ley de Euler producen:

$$I_\tau(\epsilon)\ddot{\tau} = -(F_F + F_B) L_M \cos\epsilon \sin\rho - M_{F,\tau} \tag{5}$$

De forma similar al eje de elevación, los pares de motor se han descuidado.

Aunado al modelado matemático del sistema, es importante elegir un dispositivo para la medición de los parámetros, de tal forma que como vía alternativa de investigación se procedió a unas el shield PXFmini de Erle Robotics en sustitución de la tarjeta Navio 2 que a continuación se describe:

1.3.- Definiciones:

1.3.1.- PXFmini.

El PXFmini es un shield de piloto automático de bajo costo y abierto para la Raspberry Pi que le permite crear un piloto automático listo para volar con soporte para software de vuelo APM para Drones.

Siguiendo la filosofía del DIY (hágalo usted mismo), la empresa española Erle Robotics ha desarrollado PXFmini, una pequeña shield diseñada expresamente para funcionar con Raspberry Pi (Zero, 3, 2B, 1B+). El módulo, que pesa 15 gramos, lleva una IMU de 9DOF (MPU-9250), un barómetro digital (MEAS MS5611), y un sensor ADC para saber el nivel de la batería (ADS 1015). Esto permite al drone volar de forma autónoma si se le añade algún sistema de localización, como un módulo GPS o un sensor láser [5].

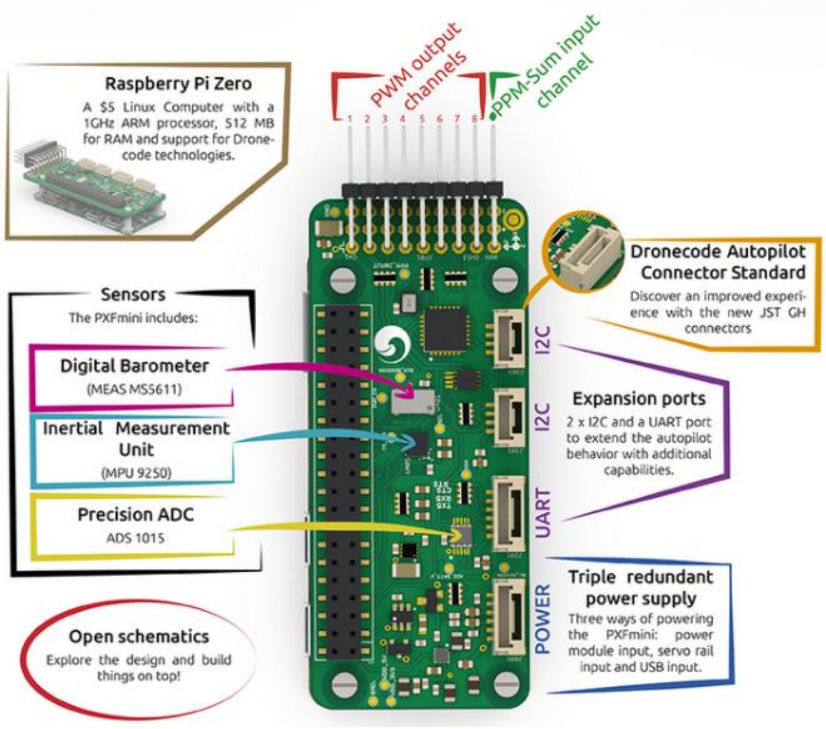

Figura 6: "Componentes de la PXFmini"

El módulo puede configurarse con APM o ROS. Erle Robotics también ha desarrollado un sistema operativo propio para Raspberry Pi.

El shield ha sido diseñado especialmente para la Raspberry Pi Zero, pero también es compatible pin-a-pin con las siguientes placas:

- Raspberry Pi.

- Raspberry Pi 2.

- Raspberry Pi 3.

- Raspberry Pi Zero.

1.3.2.- Sensor MPU-9250 Nueve ejes (Gyro + acelerómetro + brújula) Dispositivo MEMS MotionTracking.

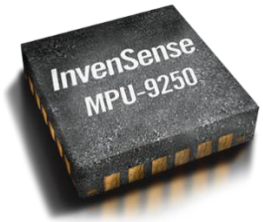

Figura 7: "Sensor MPU-9250"

La MPU-9250 es la unidad de procesamiento de movimiento de segunda generación de 9 ejes de la compañía para teléfonos inteligentes, tabletas, sensores portátiles y otros mercados de consumo. El MPU-9250, entregado en un paquete QFN de 3x3x1mm, es el dispositivo MotionTracking de 9 ejes más pequeño del mundo e incorpora las últimas innovaciones de diseño de InvenSense, permitiendo un tamaño de chip y consumo de energía dramáticamente reducido, al tiempo que mejora el rendimiento y el costo.

El dispositivo MPU-9250 MotionTracking establece una nueva referencia para el rendimiento de 9 ejes con un consumo de energía de solo 9.3μA y un tamaño un 44% más pequeño que el dispositivo de primera generación de la compañía. El rendimiento del ruido giroscópico es 3 veces mejor, y el rango total de la brújula es 4 veces mejor que las ofertas de la competencia.

El MPU-9250 es un sistema en paquete (SiP) que combina dos chips: el MPU-6500, que contiene un giroscopio de 3 ejes, un acelerómetro de 3 ejes y un Digital Motion Processor ™ integrado (DMP ™), capaz de procesar complejos algoritmos MotionFusion; y el AK8963, la brújula digital de 3 ejes líder del mercado. El MPU-9250 es compatible con MotionFusion probado en el mercado de InvenSense. Un solo diseño puede admitir MPU-9250 o MPU-6500, lo que brinda a los clientes la

flexibilidad para admitir cualquiera de los dispositivos en diferentes SKU de productos.

Los controladores de software MPU-9250 son totalmente compatibles con la versión Android 4.1 Jelly Bean de Google y admiten nuevas capacidades DMP de baja potencia que descargan el procesador host para reducir el consumo de energía y simplificar el desarrollo de aplicaciones.

1.3.3.- Sensor de presión MS5611-01BA.

Figura 8: "Sensor de presión MS5611-01BA"

Este sensor de presión barométrica está optimizado para altímetros y variómetros con una resolución de altitud de 10 cm. El módulo de sensor incluye un sensor de presión de alta linealidad y un ADC ΔΣ de 24 bits de potencia ultra baja con coeficientes internos calibrados en fábrica. Proporciona un valor preciso de presión y temperatura digital de 24 bits y diferentes modos de operación que permiten al usuario optimizar la velocidad de conversión y el consumo de corriente. Una salida de temperatura de alta resolución permite la implementación de una función de altímetro / termómetro sin ningún sensor adicional. El MS5611-01BA se puede conectar a prácticamente cualquier microcontrolador.

El protocolo de comunicación es simple, sin la necesidad de programar registros internos en el dispositivo. Las pequeñas dimensiones de solo 5.0 mm x 3.0 mm y una altura de solo 1.0 mm permiten la integración en dispositivos móviles. Esta nueva generación de módulos de sensores se basa en la tecnología MEMS líder y los últimos beneficios de la probada experiencia y conocimiento de MEAS Suiza en la fabricación de grandes volúmenes de módulos de altímetros, que se han utilizado ampliamente durante más de una década. El principio de detección empleado conduce a una histéresis muy baja y una alta estabilidad de la señal de presión y temperatura.

1.3.4.- Sensor de voltaje ADS 1015.

Los dispositivos ADS1013, ADS1014 y ADS1015 (ADS101x) son conversores de analógico a digital (ADC) de precisión, de baja potencia y 12 bits compatibles con

I2C, que se ofrecen en un paquete X2QFN-10 ultra pequeño y sin cables, y un paquete VSSOP-10. Los dispositivos ADS101x incorporan una referencia de voltaje de baja deriva y un oscilador. ADS1014 y ADS1015 también incorporan un amplificador de ganancia programable (PGA) y un comparador digital. Estas características, junto con un amplio rango de suministro operativo, hacen que el ADS101x sea muy adecuado para aplicaciones de medición de sensor de potencia y espacio limitado.

El ADS101x realiza conversiones a velocidades de datos de hasta 3300 muestras por segundo (SPS). El PGA ofrece rangos de entrada de ± 256 mV a ± 6.144 V, lo que permite mediciones precisas de señales grandes y pequeñas. El ADS1015 presenta un multiplexor de entrada (MUX) que permite dos mediciones de entrada diferenciales o cuatro de final único. Use el comparador digital en el ADS1014 y el ADS1015 para la detección de subtensión y sobretensión. El ADS101x funciona en modo de conversión continua o modo de disparo único. Los dispositivos se apagan automáticamente después de una conversión en modo de disparo único; por lo tanto, el consumo de energía se reduce significativamente durante los períodos de inactividad.

1.3.5.- Raspberry pi 3b.

Una computadora pequeña y accesible que se puede usar para aprender programación.

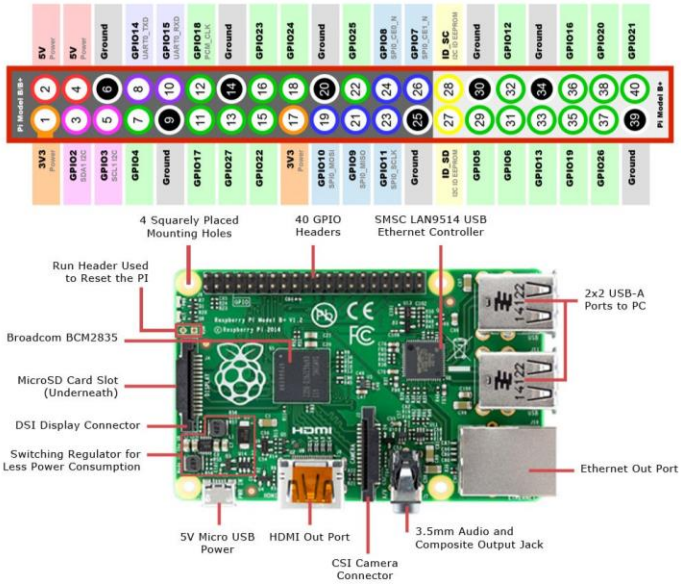

Figura 9: "GPIO Diagrama de PinOut de la Raspberry pi 3B"

La Raspberry Pi 3 es la Raspberry Pi de tercera generación. Reemplazó el modelo B de Raspberry Pi 2.

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1 GB de RAM
- BCM43438 LAN inalámbrica y Bluetooth de baja energía (BLE) a bordo
- GPIO extendido de 40 pines
- 4 puertos USB 2
- Salida estéreo de 4 polos y puerto de video compuesto
- HDMI de tamaño completo
- Puerto de cámara CSI para conectar una cámara Raspberry Pi
- Puerto de pantalla DSI para conectar una pantalla táctil Raspberry Pi
- Puerto Micro SD para cargar su sistema operativo y almacenar datos
- Fuente de alimentación Micro USB conmutada mejorada hasta 2.5A

1.3.6.- Python.

Python es un lenguaje de programación creado por Guido Van Rossum, en comparación frente a otros lenguajes, como Java o C, Python tiene la ventaja de ser un lenguaje interpretado y no necesita una compilación previa, lo que facilita el ciclo de programación y pruebas. A cambio, el rendimiento de la aplicación puede verse afectado. La primera vez que Python ejecuta una aplicación, genera un código intermedio al estilo de java que hace que en sus subsiguientes ejecuciones al rendimiento y la velocidad de ejecución sean mejores.

Figura 10: "Icono de Python"

Python es un lenguaje de tipado dinámico, es decir, que no necesitamos declarar el tipo de cada variable que utilizaremos sino que se inferirá automáticamente en el momento de utilizarla por primera vez. También es un lenguaje fuertemente tipado, lo que significa que, una vez que una variable ha sido utilizada y tiene asignada un tipo concreto, ya solo puede contener datos de ese tipo.

A pesar de ser un lenguaje muy utilizado para crear scripts de sistemas, Python es un lenguaje de propósito general que puede utilizarse también para crear complejos programas orientados a objeto y con interfaces graficas de usuario [2]. Al ser un lenguaje multiplataforma podremos crear aplicaciones que funcionaran en multitud de sistemas operativos, incluso en más plataformas que el propio lenguaje Java.

1.3.7.- ROS.

El sistema operativo del robot (ROS) es un marco flexible para escribir software de robot. Es una colección de herramientas, bibliotecas y convenciones que tienen como objetivo simplificar la tarea de crear un comportamiento robot complejo y robusto en una amplia variedad de plataformas robóticas.

Figura 11: "Icono de ROS"

¿Por qué? Porque crear software robusto y de propósito general es difícil. Desde la perspectiva del robot, los problemas que parecen triviales para los humanos a menudo varían enormemente entre las instancias de tareas y entornos. Hacer frente a estas variaciones es tan difícil que ningún individuo, laboratorio o institución puede esperar hacerlo por su cuenta.

Como resultado, ROS fue construido desde cero para fomentar el desarrollo de software de robótica colaborativa. Por ejemplo, un laboratorio podría tener expertos en el mapeo de entornos interiores y podría contribuir con un sistema de clase mundial para producir mapas. Otro grupo podría tener expertos en el uso de mapas para navegar, y otro grupo podría haber descubierto un enfoque de visión por computadora que funciona bien para reconocer objetos pequeños en el desorden.

1.3.8.- APM Planner.

APM Planner 2.0 es una aplicación de estación terrestre de código abierto para autopilotos basados en MAVlink, que incluye APM y PX4 / Pixhawk que se puede ejecutar en Windows, Mac OSX y Linux.

Figura 12: "Icono de APM Planner"

2.- PROCEDIMIENTO Y DESCRIPCIÓN DE LAS ACTIVIDADES REALIZADAS

Para el desenvolvimiento del proyecto es necesaria la investigación tanto del software como del hardware necesario para el desarrollo del proyecto, es por estas razones que el primer paso será el montaje de la PXFmini y la Raspberry pi3 modelo B, establecer y configurar la comunicación entre ambos dispositivos.

2.1.- Montaje.

Como se había mencionado en el apartado anterior, la PXFmini es compatible pin a pin con los 40 pines de la Raspberry por lo que se procedió al montaje, siendo este el primer reto debido a que la PXFmini está diseñada idealmente para adaptarse a la raspberry pi Zero como se muestra en la siguiente figura.

Figura 13: "Montaje de la PXFmini y Raspberry pi Zero"

Sin embargo, puesto que se tiene a disposición una raspberry pi 3 modelo B es necesario hacer una adaptación diferente a causa de los complementos físicos que tiene implementada la raspberry pi 3b, teniendo como resultado un montaje de la siguiente forma debido a complicaciones de tiempo:

Figura 14: "Montaje de la PXFmini en la Raspberry pi 3B"

Hay que destacar que la forma recomendada para hacer el montaje es la siguiente:

Figura 15: "Montaje ideal de la PXFmini en la Raspberry pi 3B"

Es decir, usar stacking headers en lugar de cables tipo jumper.

Una vez logrado el montaje se verificó que todos los jumpers lograran el contacto pin a pin entre la PXFmini y la raspberry pi. Se necesitó solicitar al proveedor (erle robotics) el sistema operativo raspbian que contiene los drivers y actualizaciones necesarias para que la raspberry logre la comunicación con la PXFmini.

2.2.- Instalación del SO erle.

A continuación se procedió a realizar el proceso de instalación del sistema operativo solicitado:

1. Descarga e instalación de los siguientes softwares: SD Formatter y Win32 Disk Imager.
2. Abrir el programa SD Formatter e introducir una memoria SD clase 10 en la PC.
3. Elegir y formatear la unidad de memoria asignada a la tarjeta SD y cerrar el programa.
4. Abrir Win32 Disk Imager seleccionar la unidad de memoria asignada a la tarjeta SD y buscar el disco virtual (imagen ISO) que contiene el sistema operativo solicitado.
5. Grabar el sistema operativo en la memoria SD y esperar a que el proceso finalice (suele demorarse unos minutos).
6. Introducir la tarjeta SD con el SO en la raspberry.

Para el arranque del minicomputador raspberry pi es necesario conectar un monitor vía HDMI, un teclado y un ratón, estos últimos, mediante los puertos USB disponibles en la raspberry y proveerle alimentación vía micro USB (5v a 2A).

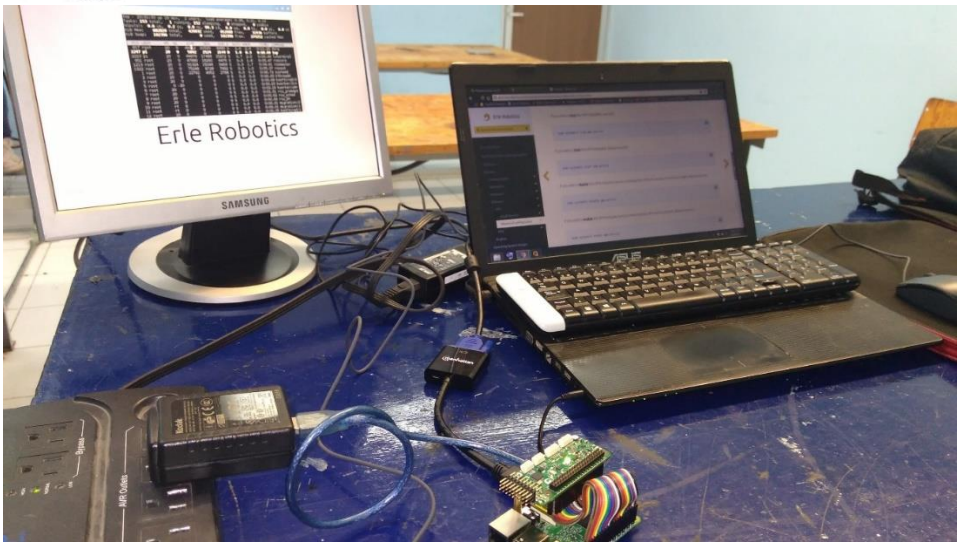

Figura 16: "Setup del sistema"

Figura 17: "Autopiloto (PXFmini y Raspberry pi 3B)"

Arrancar la raspberry solo conectando la alimentación, una vez cargado el sistema operativo. Se realiza la actualización del sistema operativo, para ello, se abre la terminal (consola de comandos) presionando clic derecho en la pantalla de escritorio y eligiendo la opción terminal una vez disponible la terminal se escribe y ejecutan los siguientes comandos:

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

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

Este proceso tiene un tiempo de duración entre 5 y 10 minutos.

Para este caso se sabe que la raspberry pi 3 modelo B cuenta con un módulo wi-fi integrado, sin embargo, este proceso no pudo realizarle puesto que el sistema operativo no permitía la conexión a internet, lo cual representó un retraso para poder

continuar con el proceso de actualización aunado a la poca experiencia y la noción limitada del uso de este tipo de lenguajes (python y ROS) usado en la raspberry.

Con la intención de darle solución al problema de la conectividad a internet se buscó información acerca de cómo configurar el módulo WiFi en el sistema operativo solicitado, obteniendo la siguiente vía de solución presentada en la documentación de la página de erle robotics.

"Existe la posibilidad de conectar el Erle-Brain 3 a una red WiFi. El cerebro será un cliente más de su red local, puede ser útil para conectar múltiples cerebros a la misma red."

2.2.1.-Configuración.

1. Guarde en un archivo el nombre y la contraseña de la red wpa. Adáptelo a su configuración de red, es decir, escriba el nombre y la clave de su red de internet. Como se muestra a continuación;

```
erle@erle-brain-3~$ wpa_passphrase Erle-Robotics 12345> erle.conf
```

2. Mueva la configuración a /etc/wpa_supplicant/ ;

```
erle @ erle-brain-3 ~ $ sudo mv erle.conf/etc/wpa_supplicant/
```

3. Editar /etc/network/interfaces:

```
# Tenga en cuenta que este archivo está escrito para ser utilizado
con dhcpcd.

# Para IP estática, consulte /etc/dhcpcd.conf y 'man dhcpcd.conf'.


auto lo

iface lo inet loopback


auto wlan0

allow-hotplug wlan0

manual iface wlan0 inet

wpa-conf /etc/wpa_supplicant/erle.conf
```

4. Luego reinicie su sistema.

"Tenga cuidado, es posible que su enrutador le asigne una dirección IP diferente. Puede asociar IP / MAC para resolver esto."

Esta guía de solución fue realizada paso a paso, obteniendo un resultado fallido de forma que se buscó otra vía de solución que cumpliera el propósito. Después de un breve proceso de búsqueda y varios intentos se encontró el proceso que da solución al problema descrito a continuación: Se abre la terminal (consola de comandos) presionando clic derecho en la pantalla de escritorio y eligiendo la opción terminal una vez disponible la terminal se escribe y ejecuta el siguiente comando:

```
sudo nano /etc/network/interfaces
```

Seguido de esto nos parece una pantalla como la siguiente:

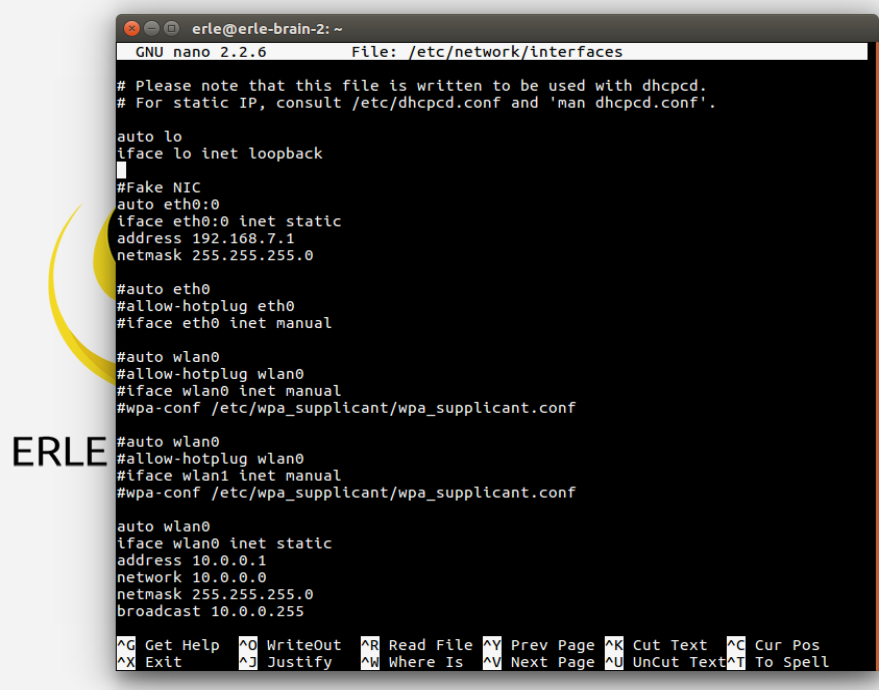

Figura 18: "Configuración del archivo /etc/network/interfaces en modo Host"

Con el cual se accede al archivo que configura la conexión a internet, destacando que la configuración brindada por erle robotics por defecto configura al módulo WiFi de la raspberry como host (anfitrión) lo que quiere decir que la raspberry generará su propia red. Para poder obtener las actualizaciones y acceso a internet se realiza el siguiente cambio en el archivo; descomentar todas las redes lo que implica borrar el símbolo "**#**" al inicio de cada línea, que como resultado en la pantalla de configuración queda de la siguiente forma:

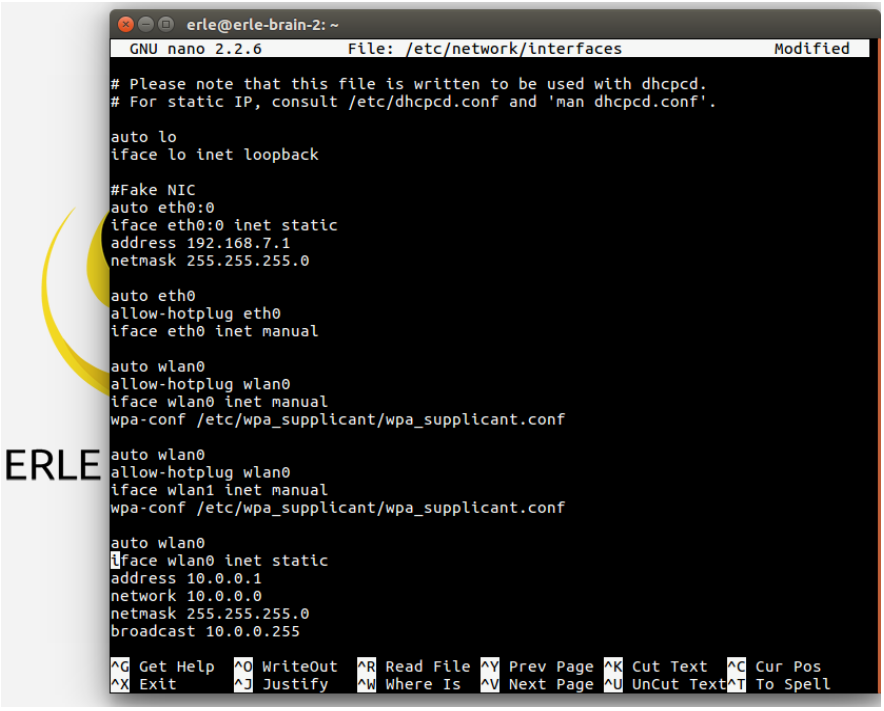

Figura 19: "Configuración del archivo /etc/network/interfaces en modo Cliente"

Se guardan los cambios (ctrl+x, “y”, enter) y se reinicia el sistema ejecutando el comando:

```
$ sudo reboot
```

Luego de esto se podrá tener acceso a las redes que identifique el módulo wi-fi.

Solucionado lo anterior se procede a elegir y conectarse una red de internet para proseguir con la actualización del sistema operativo ejecutando los comandos:

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

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

Finalizado el proceso se debe hallar la forma de iniciar la comunicación entre la raspberry y la PXFmini. Esto se lleva a cabo siguiendo las instrucciones que brinda erle robotics en su documentación, como sigue:

```
$ sudo apt-get install apm-copter-erlebrain
```

Después de instalar el servicio es necesario iniciarlo para comenzar la comunicación, para ello, ejecutamos el siguiente comando:

```
$ sudo systemctl start apm.service
```

Y se corrobora el status de la conexión:

```
$ sudo systemctl status apm.service
```

Con ello la terminal nos indicara el estado como se muestra.

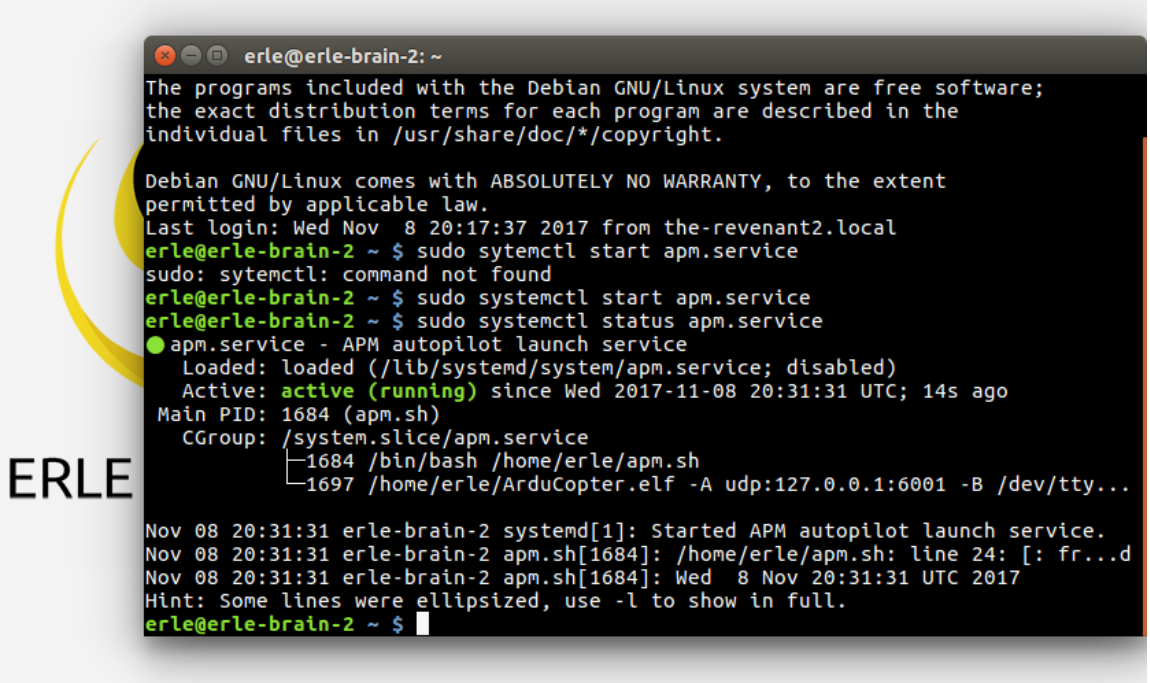

Figura 20: "Imagen de estatus de la PXFmini"

Y podemos observar que la PXFmini está activada “active (running)”

2.2.2.-Indicadores.

También podemos observar el estado de la PXFmini en el mismo dispositivo con ayuda de los leds indicadores:

No lanzado

El piloto automático aún no se lanzó o no se puede lanzar

Si el piloto automático no se inicia, los tres LED se encenderán. También puede ocurrir que el piloto automático no se lance porque hubo un error.

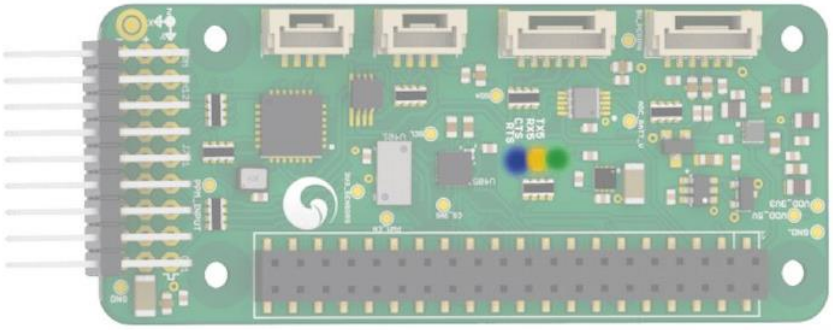

Figura 21: "Autopiloto no lanzado"

Lanzado y no armado

El piloto automático se inicia pero no está armado.

Si el piloto automático se inicia y no está armado, el LED verde se encenderá y el naranja parpadeará. Tenga en cuenta que no todos los vehículos tienen el estado armado / desarmado.

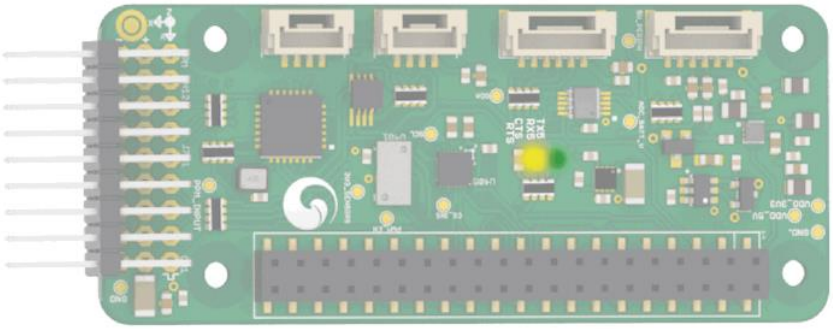

Figura 22: "Autopiloto lanzado y no armado"

Lanzado y armado

El piloto automático se inicia y está armado.

Si el piloto automático se inicia y arma, el LED verde y naranja se encenderá.

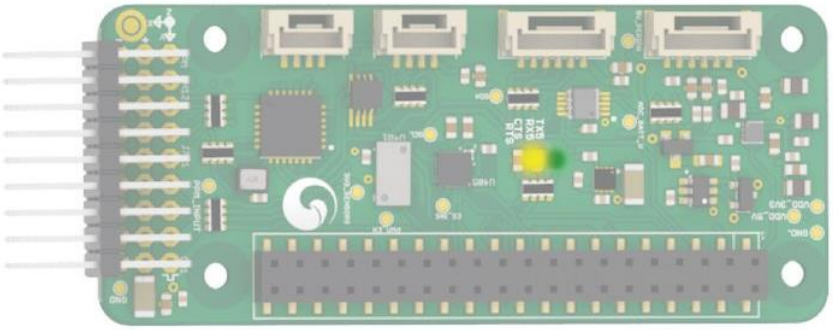

Figura 23: "Autopiloto lanzado y armado"

Teniendo esto en cuenta, se inicia la conexión remota vía ssh entre el autopiloto (Raspberry con PXFmini) y la estación remota (PC).

2.3.- Conexión con el autopiloto vía PuTTY (Windows).

Para ello es necesaria la instalación del software PuTTY para establecer comunicación entre el autopiloto y la estación remota ejecutado en el sistema operativo Windows.

Para poder realizar la conexión vía ssh es necesario que el módulo WiFi del autopiloto se configure como host para que de esa forma genere su propia señal (red).

Al finalizar la instalación y ejecutar PuTTY se presenta la siguiente pantalla.

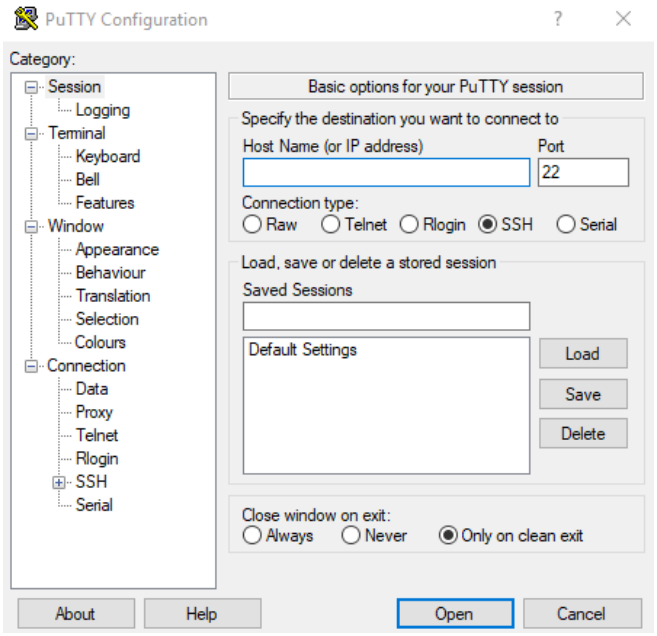

Figura 24: "Visualización de PuTTY"

La cual debemos configurar escribiendo la IP de la Raspberry que encontramos con el comando:

```
$ sudo ifconfig
```

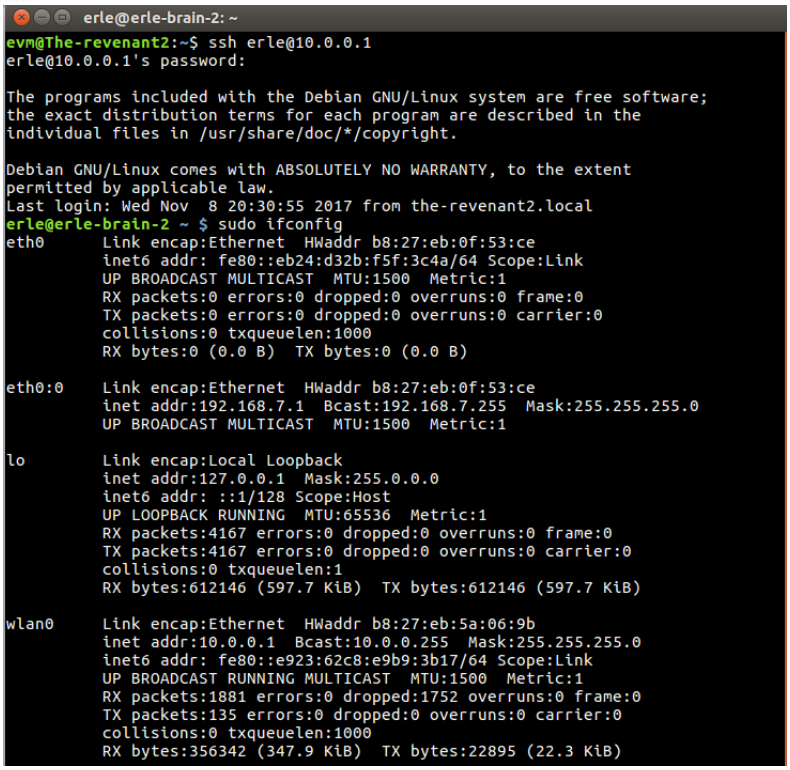

Figura 25: "Visualización de la ip de la Raspberry pi 3B"

Conociendo la IP se accede vía remota a la raspberry ingresando la IP en PuTTY

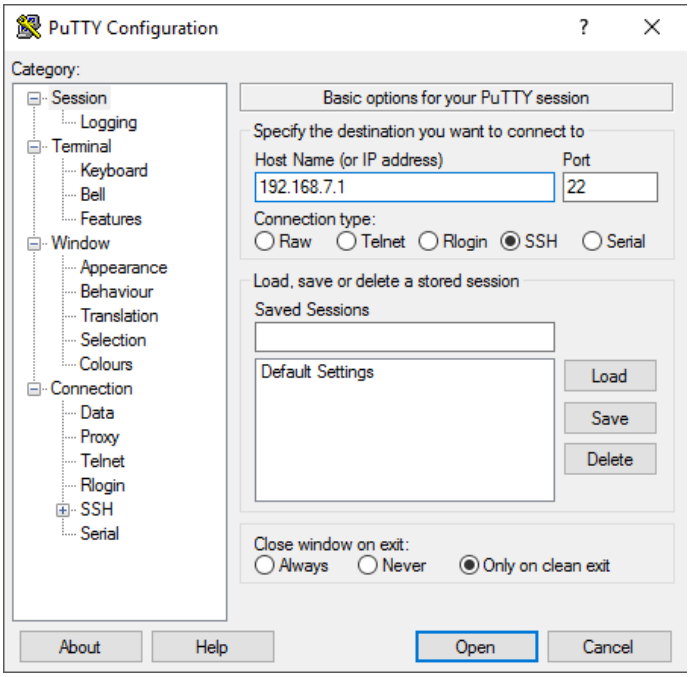

Figura 26: "Ingresando IP en el software PuTTY"

A continuación se abrirá la ventana siguiente, en la que se solicita el nombre de la cuenta y la contraseña con la cual acceder a la raspberry; existen dos cuentas, una para acceder como usuario pi y una más para hacerlo como usuario erle. Se recomienda acceder siempre como usuario erle ya que esta cuenta está configurada por los desarrolladores de la tarjeta PXFmini para optimizar la compatibilidad.

Figura 27: "Visualización de conexión vía ssh en PuTTY"

Debido a que la PXFmini es una tarjeta que se usa dentro del campo de la robótica es consecuente utilizar el metasistema ROS. Los desarrolladores de la PXFmini configuraron la shield con este metasistema por que facilita la programación en robótica, es por eso que, llegado a este punto fue necesario investigar sobre este sistema con el fin de comprender al menos de manera básica tanto la estructura así como el lenguaje.

2.4.- Introducción a ROS.

Para poder hacer una introducción al metasistema ROS es necesario tener en cuenta los siguientes conceptos [2].

Tópico (topic):

Son canales de información entre los nodos. Un nodo puede emitir o suscribirse a un tópico. Por ejemplo, stage (simulador de robots) emite un tópico que es la odometría del robot. Cualquier nodo se puede suscribir. El nodo que emite no controla quién está suscrito. La información es, por tanto, unidireccional (asíncrona). Si lo que queremos es una comunicación síncrona (petición/respuesta) debemos usar servicios.

Un tópico no es más que un mensaje que se envía. Podemos usar distintos tipos de clases de mensajes.

Paquete (package):

El software en ROS está organizado en paquetes. Un paquete puede contener un nodo, una librería, conjunto de datos, o cualquier cosa que pueda constituir un módulo. Los paquetes pueden organizarse en pilas (stacks).

Nodo (node):

Un nodo es un proceso que realiza algún tipo de computación en el sistema. Los nodos se combinan dentro de un grafo, compartiendo información entre ellos, para crear ejecuciones complejas. Un nodo puede controlar un sensor láser, otro los motores de un robot y otro la construcción de mapas.

Pila (stack):

Conjunto de nodos que juntos proporcionan alguna funcionalidad (por ejemplo, la pila de navegación).

Como todo sistema ROS también funciona a través de comandos, que nos permite navegar e interactuar (mover, copiar, modificar y crear) con todos los archivos por medio de la estructura por ejemplo:

roscd: cambia a un directorio de paquete o pila (ej. roscd stage).

roscore: ejecuta todo lo necesario para que dar soporte de ejecución al sistema completo de ROS. Siempre tiene que estar ejecutándose para permitir que se comuniquen los nodos. Permite ejecutarse en un determinado puerto (ej. roscore o roscore -p 1234).

roscreate-pkg: crea e inicializa un paquete. Se tiene que ejecutar desde uno de los directorios válidos para que contengan paquetes. El formato de ejecución es: roscreate-pkg paquete [depen1...] donde depen1 es una dependencia. Por ejemplo, si el paquete que estamos creando va a usar los mensajes estándar y va a usar código c++, debemos indicar las dependencias std_msgs y roscpp.

rosnode: nos proporciona información sobre un nodo. Disponemos de las siguientes opciones:

rosnode info nodo (muestra información sobre el nodo).

rosnode kill nodo (mata ese proceso).

rosnode list (muestra los nodos ejecutándose).

rosnode machine maquina (muestra los nodos que se están ejecutando en la máquina).

rosnode ping nodo (comprueba la conectividad del nodo).

rosrun: permite ejecutar cualquier aplicación de un paquete sin necesidad de cambiar a su directorio. Podemos pasarle parámetros con _my_param:=value (ej. rosrun stage stageros) stage es el paquete y stageros es la aplicación que ejecutamos.

rostopic: permite obtener información sobre un tópico.

rostopic bw (muestra el ancho de banda consumido por un tópico).

rostopic echo (imprime datos del tópico por la salida estándar).

rostopic find (encuentra un tópico).

rostopic info (imprime información de un tópico).

rostopic list (imprime información sobre los tópicos activos).

rostopic pub (publica datos a un tópico activo).

rostopic type (imprime el tipo de información de un tópico).

2.5.- Lectura de datos del autopiloto (Windows).

Lo más importante de la comunicación entre ambos dispositivos es la lectura datos, por lo que se da prioridad a la adquisición de estos datos enviados de la PXFmini hacia la estación remota, para visualizar la lectura de estos datos provenientes de los sensores. Para poder ver la lectura de algún sensor de los disponibles en la shield es importante conocer el uso de Mavros, Mavros es un nodo de ROS extendido de MAVLink con un proxy UDP para las estaciones de control en tierra [6].

Por ende es necesario conocer y ejecutar el comando:

```
$ rostopic list
```

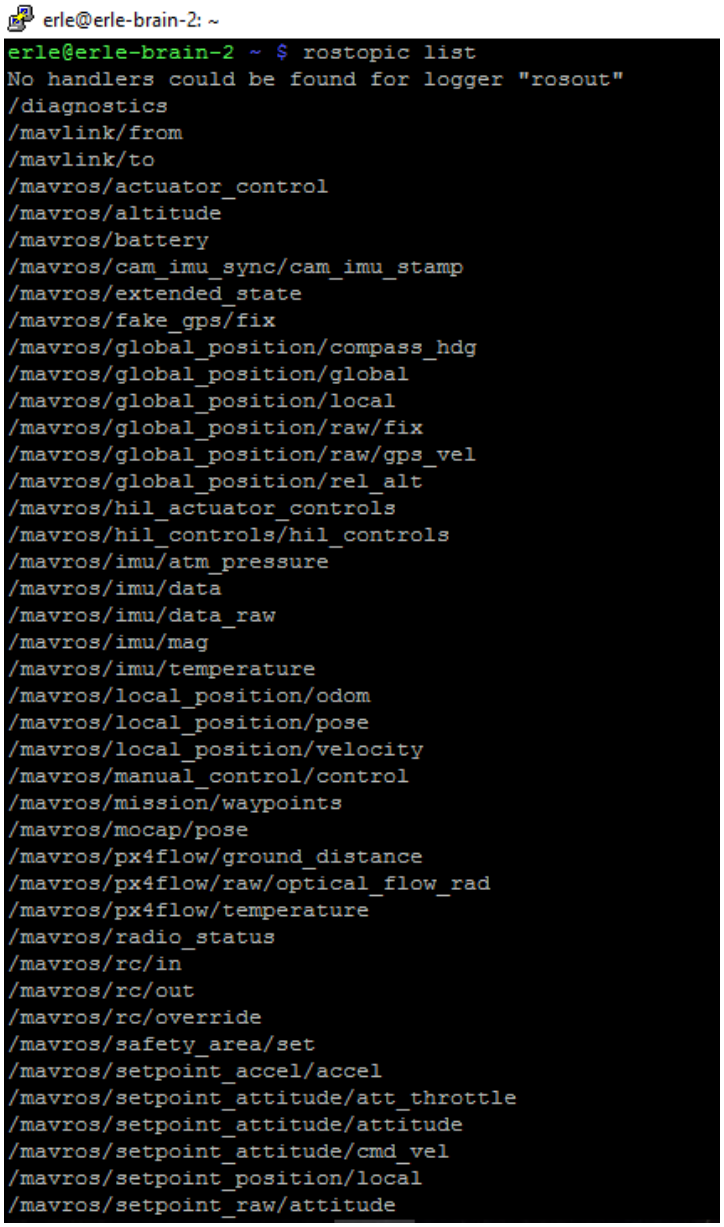

Figura 28: "Listado de tópicos provistos por ROS"

Uno de los datos principales para el control de un drone son los datos provistos por la IMU, para visualizar estos datos se emplea el comando:

```
$ rostopic echo /mavros/imu/data
```

Al ejecutarlo enseguida vemos el censado de la IMU.

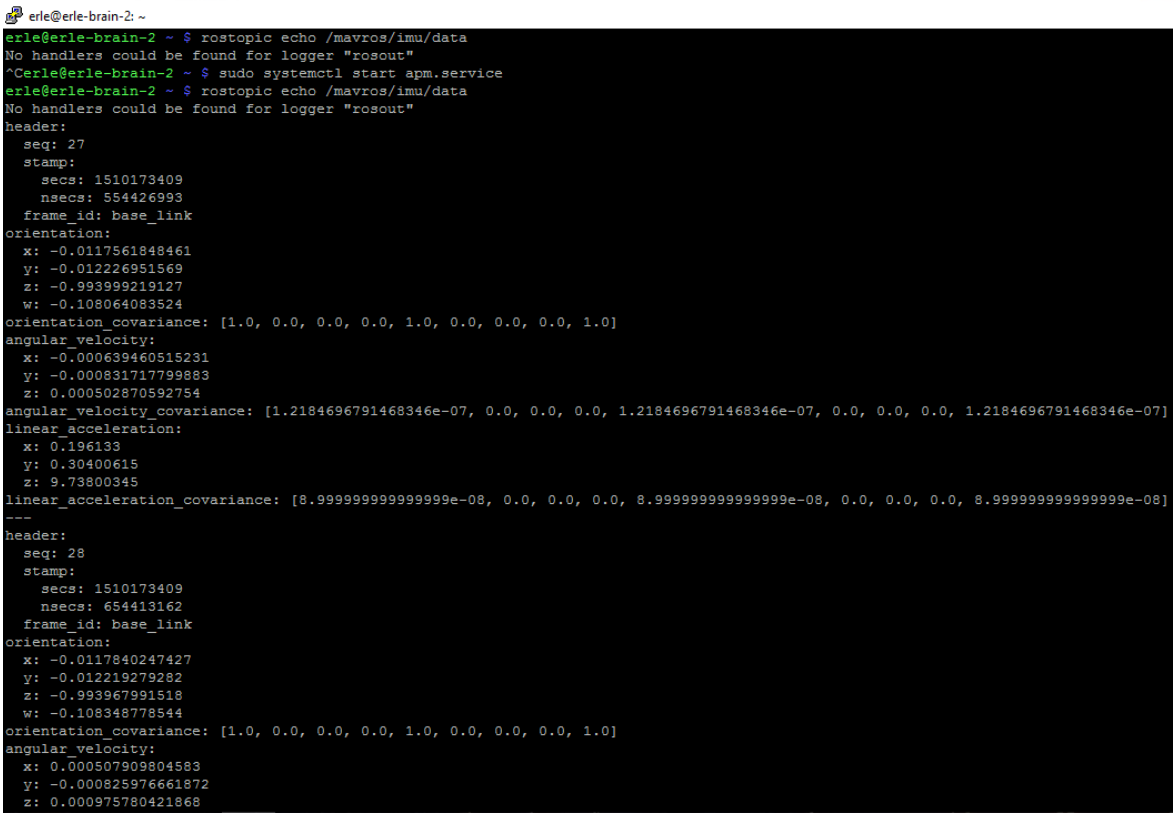

Figura 29: "Ejemplo de una lectura de la IMU"

2.6.- Pruebas con APM Planner (Windows).

Los desarrolladores de la PXFmini sugieren el uso de APM Planner como un recurso para el control del VANT así como para facilitar la calibración del drone. Se procedió a la instalación de APM Planner en el sistema Windows de la estación remota (PC), para enlazar el autopiloto con el software fue necesario conectar la PC a la red emitida por el autopiloto. Hecho lo anterior es posible transmitir y recibir datos entre ambos equipos, el objetivo de este paso es llevar a cabo la demo “Takeoff-land” con los archivos que el desarrollador Erle Robotics proporciona. Cabe destacar que la demo a ejecutarse no viene por defecto en el SO brindado por Erle Robotics, sin embargo, siguiendo una pequeña serie de pasos es posible descargar y configurar estos archivos.

2.6.1.- Creando espacio de trabajo.

Antes de proceder a hacer algo, es necesario crear nuestro propio espacio de trabajo en la cuenta de usuario de Erle Robotics en la raspberry pi. En dicho espacio de trabajo, tendremos todo el código disponible para llevar a cabo el desarrollo de programas y pruebas que sean necesarias, en este caso se usará para almacenar y configurar la demo “Takeoff-land” [3]. Para ver el espacio de trabajo que ROS está usando, use el siguiente comando en la terminal del autopiloto con la ayuda de PuTTY:

```
$ echo $ROS_PACKAGE_PATH
```

En este caso, el resultado muestra la siguiente dirección:

```
/opt/ros/kinetic/share
```

La carpeta a crear está en

```
~/home/erle/catkin_ws/src/
```

Para agregar esta carpeta, usamos los siguientes comandos:

```
$ mkdir catkin_ws
$ cd catkin_ws/
$ mkdir src
$ cd src/
$ catkin_init_workspace
```

Una vez que hemos creado la carpeta de espacio de trabajo, no hay paquetes dentro, sólo el archivo “CMakeList.txt”. El siguiente paso es construir el espacio de trabajo. Para hacer esto, usamos los siguientes comandos:

```
$ cd ..
$ catkin_make
```

Ahora, si escribe el comando “ls” en el directorio, se puede ver las nuevas carpetas creadas con el comando anterior. Estas son las carpetas de desarrollo. Para finalizar la configuración, use el siguiente comando:

```
$ source devel/setup.bash
```

Este paso es solo para volver a cargar el archivo setup.bash. Se obtendrá el mismo resultado si cierra y abre un Shell.

2.6.2.- Ejecución de la demo Takeoff-Land (Windows).

Una vez creado el espacio de trabajo es posible descargar los archivos de la demo "Takeoff-land" para esto se siguen los pasos y funcionará sin problemas [7]:

El primer paso es ejecutar el comando:

```
$ cd /home/erle/catkin_ws/src/
$ git clone https://github.com/erlerobot/ros_erle_takeoff_land
$ cd  ..
$ catkin_make_isolated --pkg ros_erle_takeoff_land
```

Si todo ha funcionado correctamente, se recargan las variables de entorno:

```
$ source devel_isolated/setup.bash
```

Ahora todo está listo para usar la demo. De tal forma que cada vez que se quiera utilizar, se debe conectar la PC y el autopiloto con la ayuda de APMplanner, de igual forma se ejecutan los siguientes comandos:

```
$ source catkin_ws/devel_isolated/ros_erle_takeoff_land/setup.bash
$ rosrun ros_erle_takeoff_land ros_erle_takeoff_land
```

Al continuar con el enlace a realizar entre los dispositivos, y con la ayuda del software APM Planner, se notó el error surgido, es decir, fue posible conectarse a la red del autopiloto pero la interacción entre el autopiloto y APM Planner no tuvo éxito alguno, por lo cual se procedió a llevar a cabo otra vía de solución para lograr la interacción con el autopiloto.

2.6.3.- Ejecución de la demo IMU Visualization (Windows).

Esta segunda alternativa requirió de volver a hacer uso del software PuTTY para visualizar todos los datos de la IMU utilizando la demo "ros erle IMU". De igual forma que con el paquete anterior esta nueva demo requiere su descarga y configuración en el autopiloto de esta forma:

Nuevamente creamos una carpeta que será nuestra área de trabajo, la carpeta crear está en:

```
~/home/erle/
```

Para agregar esta carpeta, usamos los siguientes comandos:

```
$ mkdir imu_catkin_ws
$ cd imu_catkin_ws/
$ mkdir src
```

```
$ cd src/
$ catkin_init_workspace
```

Es importante descargar y compilar el código una vez hecho lo anterior, para lograrlo se ejecuta lo siguiente:

```
$ cd /home/erle/imu_catkin_ws/src
$ git clone https://github.com/erlerobot/ros_erle_imu
$ cd  ..
$ catkin_make_isolated --pkg ros_erle_imu
```

Una vez configurado iniciamos la conexión vía ssh con PuTTY entre el autopiloto y la estación remota y se procede a ejecutar la demo de visualización de la imu. En la terminal de comandos de PuTTY ejecutamos el nuevo paquete instalado, es decir, corremos el programa. Con el los siguientes comandos:

```
$ sudo –s
$ cd ~/imu_catkin_ws
$ source devel/setup.bash
$ rosrun ros_erle_imu visualization.py
```

Después de los constantes intentos en el sistema operativo Windows, los diversos errores que se generaban y sabiendo que este tipo de dispositivos funcionan mejor con sistemas operativos Linux se optó por trabajar en un sistema Ubuntu xenial 16.04 en la pc.

2.7.- Instalación de Ubuntu.

En caso de contar con un disco duro adicional es posible instalarlo en ese espacio, en este caso se debe de crear una partición con memoria libre de la siguiente forma:

- Accedemos al administrador de discos de Windows 10 desde el Panel de control > Herramientas administrativas o empleando el comando “compmgmt.msc” desde la ventana ejecutar.

Y obtendremos una ventana como la siguiente:

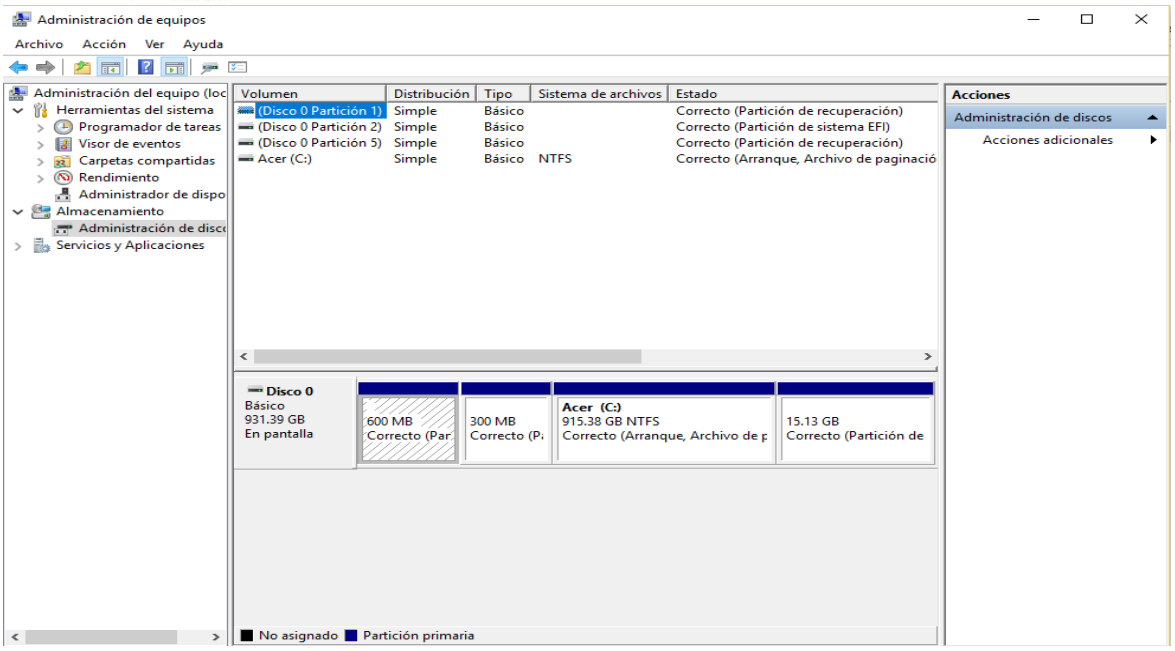

Figura 30: "Ejemplo la ventana para crear particiones"

Tenemos una partición reservada para el sistema y el resto en una partición “C” ocupada con la instalación de Windows 10 que es la que tendremos que reducir para lograr espacio. Para ello pulsamos con el botón derecho sobre ella y marcamos sobre “reducir volumen”.

La herramienta comprobará el espacio libre disponible en la partición “C” y mostrará el máximo en la que la podamos reducir la partición (en nuestro caso 887623 MB. Fijamos en 60000 MB el espacio a reducir y ese será el tamaño de la partición para Ubuntu. El tamaño es lo suficiente para los procesos que se harán.

Veremos en la ventana de administración de equipos cómo rápidamente se habrá creado un espacio en disco adicional vacío. No es necesario crear más particiones ya que el instalador de Ubuntu se encargará de crear las particiones necesarias en el espacio libre que hemos creado.

2.7.1.- Preparando BIOS / UEFI.

Aprovechando las posibilidades de las UEFI, Microsoft activó (a partir de Windows 8) un “sistema de arranque seguro” denominado Secure Boot que obliga a firmar el firmware y el software protegiendo el proceso de arranque del sistema. Varias distribuciones GNU/Linux añadieron soporte para poder instalarse. La opción recomendada es tener el “Secure Boot” funcionando en modo BIOS de esta forma se evitan problemas al instalar sistemas operativos adicionales a Windows 8/10.

Si tienes una placa UEFI y tienes problemas en la instalación, su desactivación se realiza de la siguiente forma:

- Accedemos a UEFI de forma similar que con BIOS, pulsando en el arranque la tecla correspondiente que utilice el fabricante (normalmente Escape, alguna tecla de Función o una combinación de ellas) y buscaremos el apartado de Secure Boot.

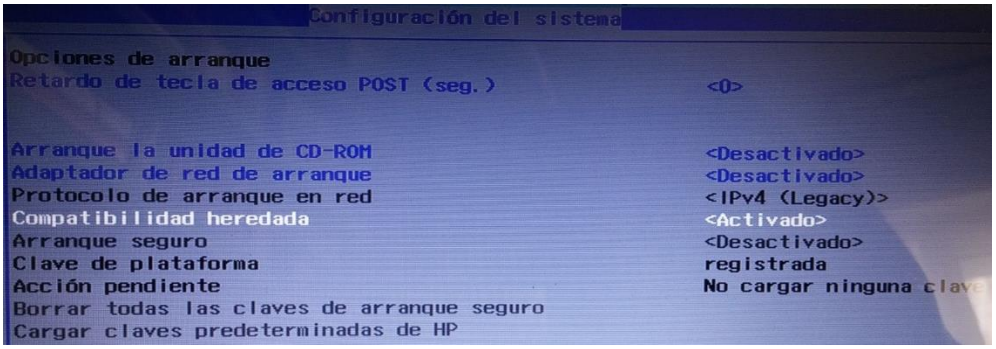

Figura 31: "Ejemplo de la BIOS"

- En nuestro caso es cuestión de un par de pasos como verás en la imagen anterior: Deshabilitar el “Arranque seguro” y habilitar la “Compatibilidad heredada”. Dependiendo del equipo, la placa y la versión de UEFI lo podrás encontrar de una u otra forma pero similar a lo indicado.

Orden de arranque

- Ya que estamos en la BIOS/UEFI, aprovechamos para asegurarnos que la unidad desde donde vamos a instalar Windows 10 esté por delante del arranque del disco duro o SSD. En nuestro caso vamos a utilizar un pendrive USB y por ello lo verás en primer lugar en “Orden de arranque heredada”. También podríamos emplear un DVD. En ese caso lo colocaríamos por delante en el orden de arranque.

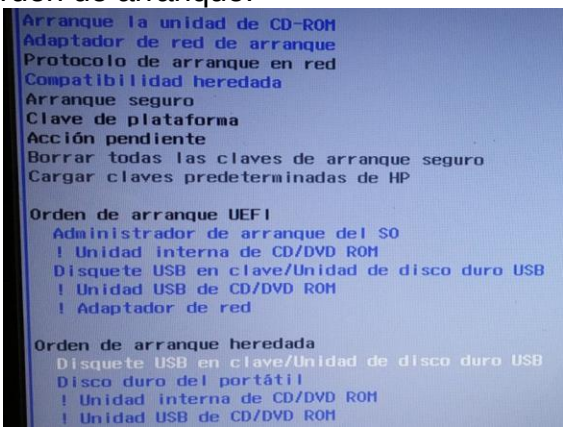

Figura 32: "Selección de Orden de Arranque para la USB"

2.7.2.- Descargando Ubuntu.

Accedemos a la página web oficial que ha preparado Canonical y descargamos la versión que nos interese. En nuestro caso optamos por la imagen .ISO de Ubuntu para escritorio y de 64 bits.

Preparando medio de instalación

Una vez descargado podemos utilizar como decíamos un medio óptico (DVD) o una unidad USB (pendrive o disco externo). Vamos a utilizar ésta última, más rápida (USB 3.0) y versátil.

Para ello utilizamos una herramienta que nunca falta en nuestra colección, Rufus, aunque puedes emplear otras. Descargamos y ejecutamos Rufus con el medio USB insertado que utilizaremos en la grabación. Seleccionamos como tipo de partición MBR para BIOS o UEFI y fijamos la imagen ISO recientemente descargada de Ubuntu. Creamos el medio con los parámetros comentados anteriormente.

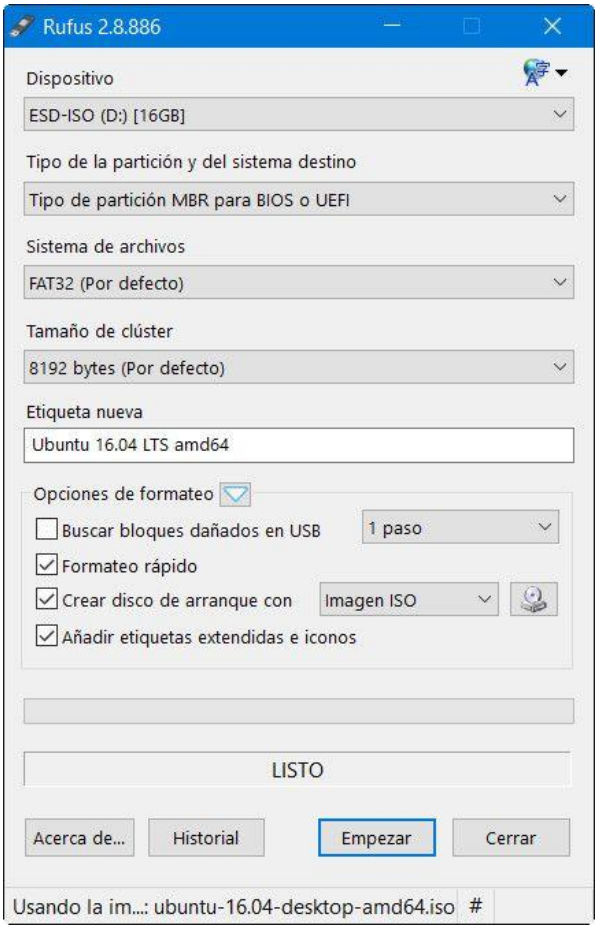

Figura 33: "Interfaz del software Rufus"

2.7.3.- Instalando Ubuntu.

- Colocamos el pendrive que hemos preparado y reiniciamos el equipo. El menú de arranque de Ubuntu nos permite probar el sistema en modo “Live CD” o instalar en disco. Elegimos la segunda opción y continuamos:

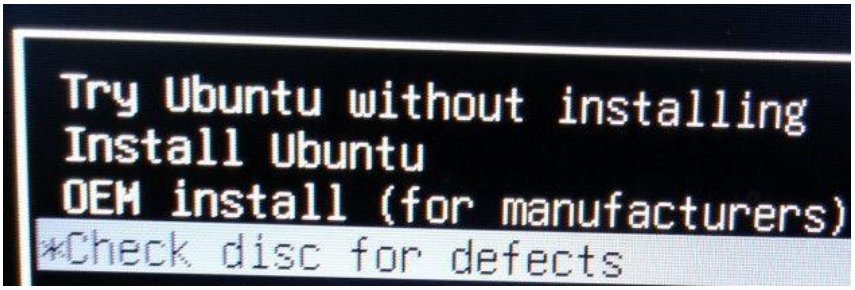

Figura 34: "Selección de partición para el SO"

- Seleccionamos español como idioma de instalación.
- No actives la descarga de actualizaciones ni la instalación de software de terceros. Lo haremos posteriormente desde el mismo sistema.
- La siguiente pantalla sí es importante y refiere al tipo de instalación. Si controlas las particiones en Linux puedes crearlas a tu gusto utilizando la pestaña “Más opciones”, establecer tamaño, punto de montaje o instalación del cargador de arranque. Si no eres un experto no te compliques la vida porque no hay que hacer nada de eso.
- Como verás, el instalador reconoce una instalación de Windows 10 y te ofrece instalar Ubuntu junto a él. Esta es la opción que debes elegir sin tocar para nada las particiones.

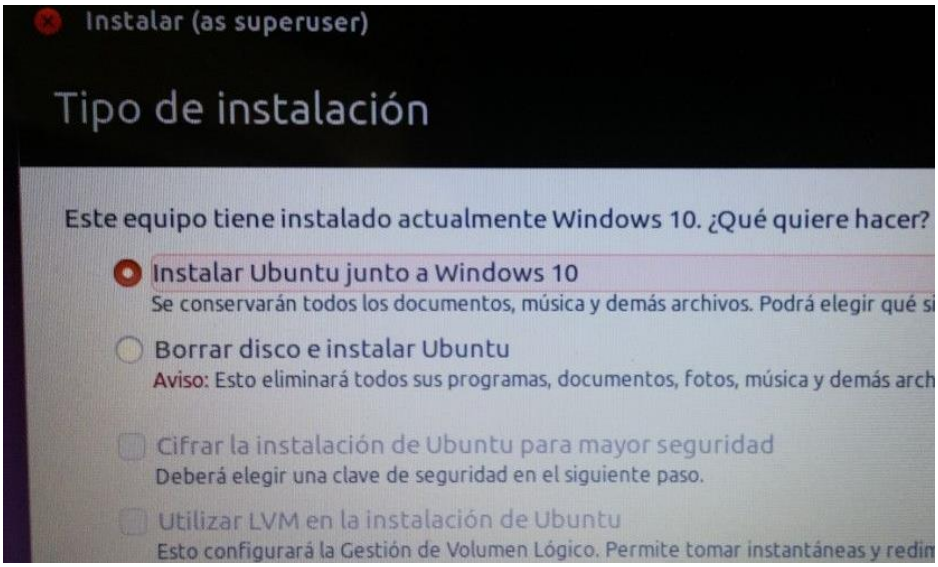

Figura 35: "Selección del tipo de instalación"

- En esa configuración, Ubuntu creará sus propias particiones en el espacio de la SSD (60 GB) que sé dejó libre e instalará su cargador de arranque, en este caso Grub 2. Nosotros tuvimos una advertencia de UEFI que cancelamos porque recuerda, tenemos Windows 10 en modo BIOS y Ubuntu hay que instalarlo de la misma forma porque de lo contrario no podrás arrancar posteriormente el sistema operativo de Microsoft.
- No hay que hacer nada más salvo indicar el nombre del usuario y contraseña de administrador que utilizaremos.

Una vez completada la instalación el equipo reiniciará en el cargador de arranque Grub 2. En el proceso de instalación no detecta Windows 10 pero tranquilo, lo resolveremos con un simple comando en la terminal de Ubuntu. Retiramos el pendrive de instalación e iniciamos Ubuntu.

Como se mencionaba, el gestor de arranque Grub 2 no cargó durante el proceso de instalación Windows 10 u otros sistemas instalados. Eso es lo primero que vamos a resolver de la siguiente manera, abrimos la terminal y ejecutamos el comando:

```
$ sudo update-grub2
```

- Como verás en la imagen, se generará un archivo de configuración de grub que encontrará los sistemas instalados como Windows 10.
- Cerramos la terminal y reiniciamos el equipo para comprobar que el gestor de arranque funciona correctamente y podemos arrancar indistintamente Windows 10 y Ubuntu 16.

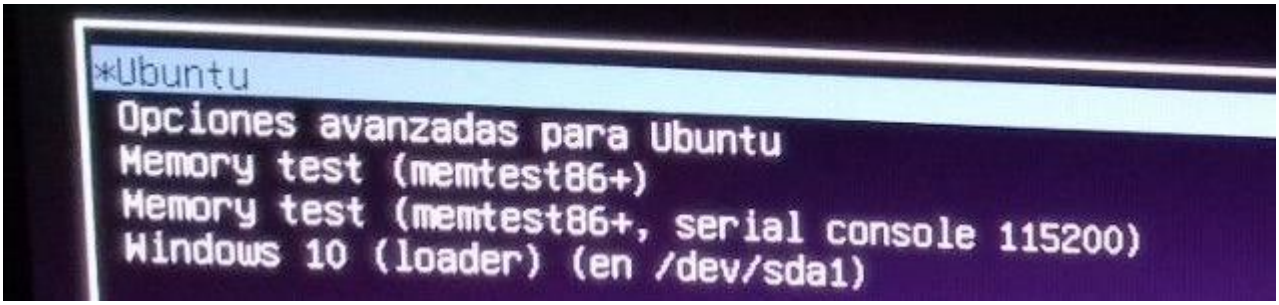

Figura 36: "Interfaz de selección de SO de arranque"

Generado el archivo de configuración por Grub ya tenemos acceso a los sistemas instalados. Desde ahí podremos acceder indistintamente a cualquiera de ellos.

2.7.4.- Instalación de ROS (Ubuntu).

A continuación se procede a instalar el metasistema ROS en Ubuntu, ya que esto permitirá que se logre la comunicación entre la estación remota y el autopiloto, los pasos para la instalación son los siguientes:

1. Instalación.

ROS Kinetic Sólo soporta sistemas Wily (Ubuntu 15.10), Xenial (Ubuntu 16.04) y Jessie (Debian 8) para paquetes debian.

2. Preparar tu sources.list (lista de fuentes).

Preparar tu computadora para aceptar software de packages.ros.org se hace ejecutando el siguiente comando:

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

3. Preparar las llaves.

$ sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116

Si se experimenta errores para conectarse con el servidor de llaves, se puede intentar sustituyendo lo siguiente: hkp://pgp.mit.edu:80 o hkp://keyserver.ubuntu.com:80 en el comando anterior.

4. Instalación.

Primero que nada, hay que asegurarse de que el índice del paquete Debian está actualizado:

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

Existen muchas librerías diferentes y herramientas en ROS. Se provee el metasistema con configuraciones iniciales para dar una introducción amigable. También se puede instalar paquetes ROS de forma individual. En caso de problemas con el siguiente paso, se puede usar los siguientes repositorios en lugar de los mencionados arriba ros-shadow-fixed.

Instalación de escritorio completa: (Recomendada): ROS, rqt, rviz, librerías-genéricas-robóticas, simuladores 2D/3D, navegación y percepción 2D/3D. Con el siguiente comando:

```
$ sudo apt-get install ros-kinetic-desktop-full
```

5. Inicializar rosdep.

Antes de que se pueda usar ROS, se necesita inicializar rosdep. Rosdep le permite instalar fácilmente las dependencias del sistema para la fuente que desea compilar y es necesario para ejecutar algunos componentes principales en ROS. Para ello se ejecuta lo siguiente:

```
$ sudo rosdep init

$ rosdep update
```

6. Configuración del entorno.

Es conveniente si las variables de entorno ROS se agregan automáticamente a su sesión bash cada vez que se lanza un nuevo shell:

```
$ echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc source ~/.bashrc
```

7. Dependencias para construir paquetes.

Hasta ahora, ha instalado lo que necesita para ejecutar los paquetes principales de ROS. Para crear y gestionar sus propios espacios de trabajo ROS, existen diversas herramientas y requisitos que se distribuyen por separado. Por ejemplo, rosinstall es una herramienta de línea de comandos de uso frecuente que le permite descargar fácilmente muchos árboles fuente para paquetes ROS con un solo comando. Para instalar esta herramienta y otras dependencias para crear paquetes ROS, ejecute:

```
$ sudo apt-get install python-rosinstall python-rosinstall-generator python-wstool build-essential
```

Con esto, ROS está instalado y listo para usarse. Hay que destacar que siempre hay que estar pendiente a los cambios de los desarrolladores de la pxfmini, y mantener actualizados los drivers para su uso. Es importante, de igual forma, la instalación del software APM Planner en el sistema Ubuntu para poder proceder a ejecutar las diferentes alternativas de solución.

2.8.- Construyendo APM Planner (Ubuntu).

1. Instala los paquetes requeridos:

Asegúrese de ejecutar `$ sudo apt-get update` primero.

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

```
$ sudo apt-get install qt5-qmake qt5-default \ qtscript5-dev libqt5webkit5-dev libqt5serialport5-dev \ libqt5svg5-dev qtdeclarative5-qtquick2-plugin libqt5serialport5-dev \ libqt5opengl5-dev qml-module-qtquick-controls
```

```
$ sudo apt-get install git libsdl1.2-dev libsndfile-dev \ flite1-dev libssl-dev libudev-dev libsdl2-dev python-serial python-pexpect
```

2. Clona el repositorio en tu espacio de trabajo:

```
$ cd ~/workspace
```

```
$ git clone https://github.com/diydrones/apm_planner
```

3. Construyendo APM Planner:

```
$ cd ~/workspace/apm_planner
```

```
$ qmake apm_planner.pro
```

```
$ make
```

4. Ejecutar APM Planner:

```
$ ./release/apmplanner2
```

5.- Instalación permanente.

```
$ cd ~/workspace/apm_planner
```

```
$ sudo make install
```

Esto colocará el binario en su carpeta / bin / y los archivos correspondientes en / share /

Hecho lo anterior ahora se puede interactuar el meta-sistema ROS instalado en el autopiloto y en la estación remota. Para ello se abre la terminal de comandos en Ubuntu y se inicia la conexión vía ssh con el autopiloto, nuevamente:

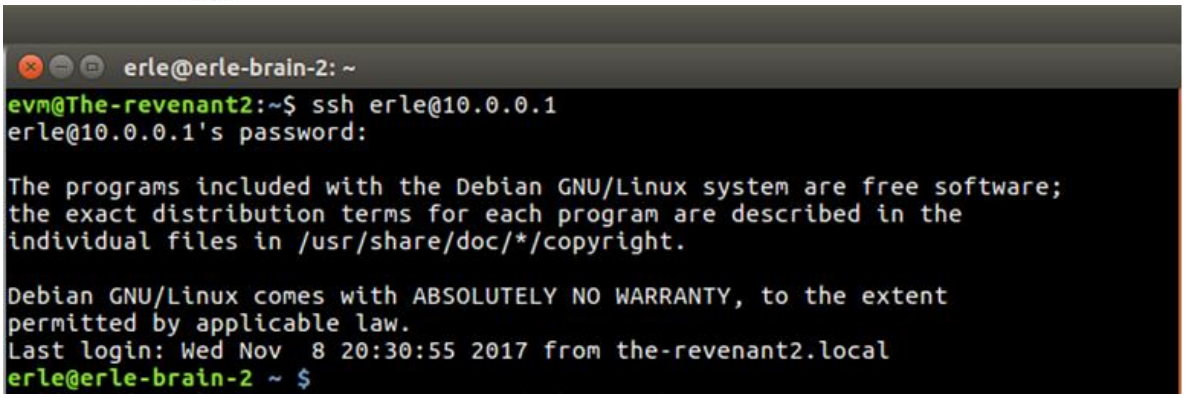

Figura 37: "Conexión vía ssh desde Ubuntu"

2.9.- Ejecución de la demo IMU Visualization (Ubuntu).

Una vez iniciada la conexión se procede a ejecutar la demo de visualización de la IMU. En la terminal de comandos ejecutamos el paquete apropiado, es decir, ejecutamos el programa:

```
$ sudo -s
$ cd ~/imu_catkin_ws
$ source devel/setup.bash
$ rosrun ros_erle_imu visualization.py
```

Se muestra una imagen como la siguiente en una ventana.

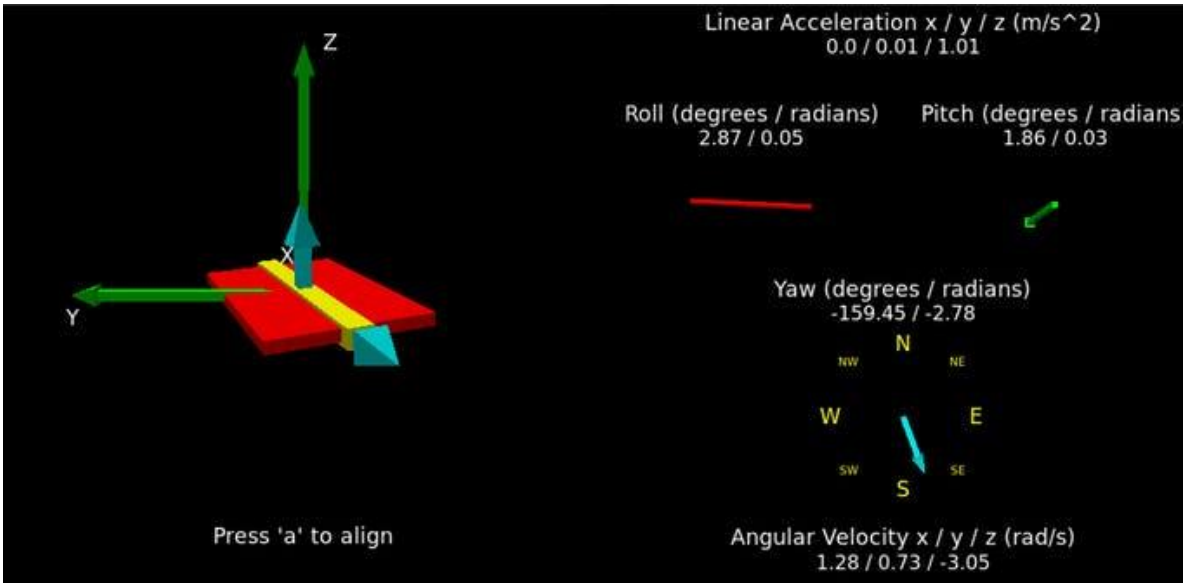

Figura 38: "Visualización de datos del Autopiloto (PXFmini) y graficas 3D"

Llegado a este punto, y confirmando que el autopiloto interactúa con la estación remota mostrando los datos de la IMU se procede a ejecutar la demo Takeoff-Land.

2.10.- Ejecución de la demo Takeoff-Land (Ubuntu).

Para ello, una vez iniciada la conexión vía ssh, en la terminal de comandos se ejecuta el siguiente comando para iniciar APM Planner:

```
$ ./release/apmplanner2
```

Ahora se comienza a usar los comandos para iniciar la demo:

```
$ source catkin_ws/devel_isolated/ros_erle_takeoff_land/setup.bash
```

```
$ rosrun ros_erle_takeoff_land ros_erle_takeoff_land
```

Una vez hecho esto se debe mostrar lo siguiente en APM Planner.

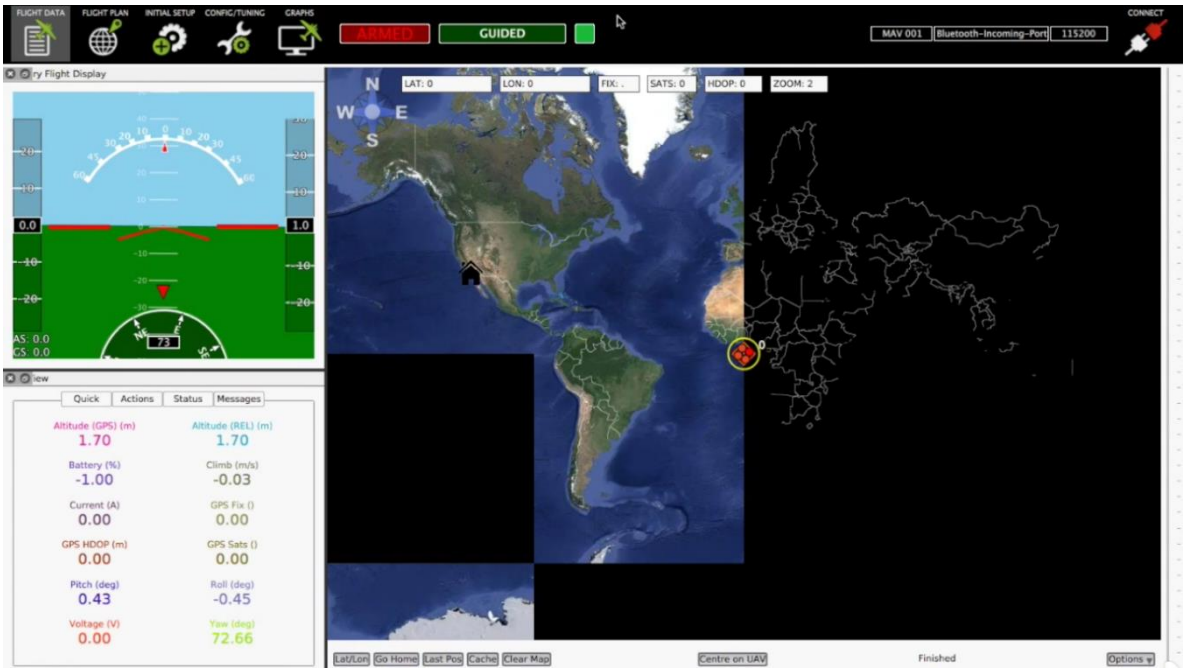

Figura 39: "Interfaz del software APM Planner en Ubuntu"

En este caso hay que aclarar que no se tuvo éxito al ejecutar esta demo, el software y el autopiloto no lograron interactuar. Con todo, se puede presentar una alternativa para el desarrollo del control del Drone, siendo esta el uso de un autopiloto distinto a la PXFmini, como podría ser la shield Navio2.

Ahora bien, con el afán de no interrumpir el avance del proyecto, y también de obtener una idea más clara del funcionamiento del sistema de un Drone tipo tándem y su control, se presenta una propuesta de diseño de una estación de pruebas de control y estabilidad del sistema de un Drone tipo tándem, que consiste en: una base con ejes que simulan 3 grados de libertad para verificar la estabilidad del vehículo, un par de base para motores brushless y un contra peso variable para diversas pruebas.

A continuación se presentan las imágenes del diseño de la estación de pruebas.

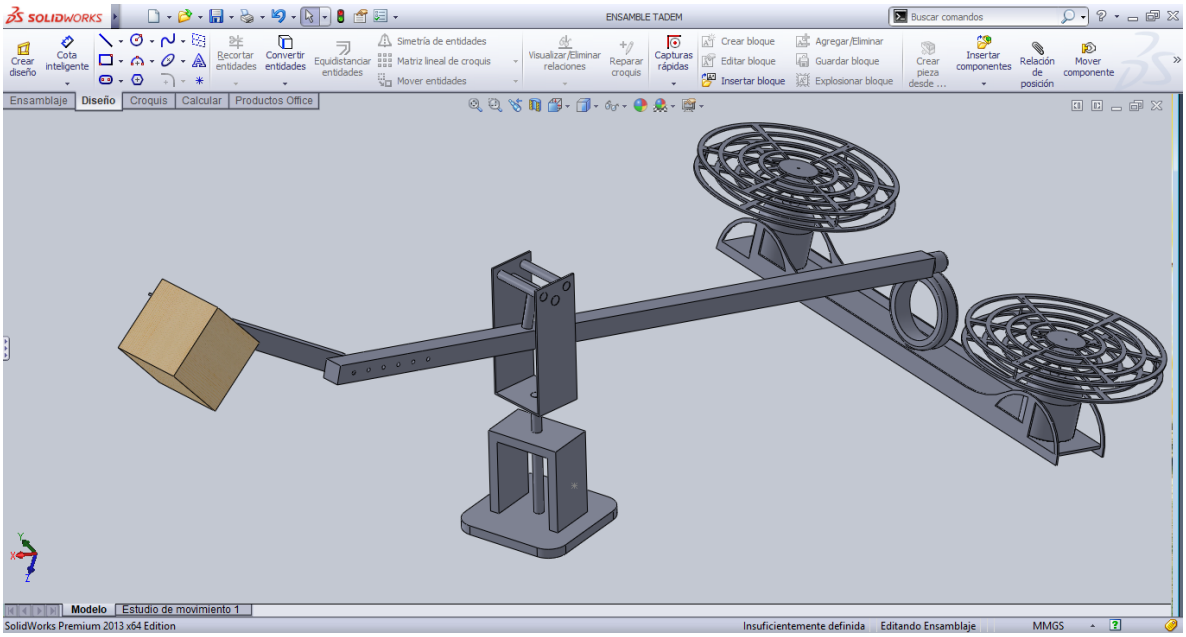

Figura 40: "Diseño del ensamble de la base de pruebas"

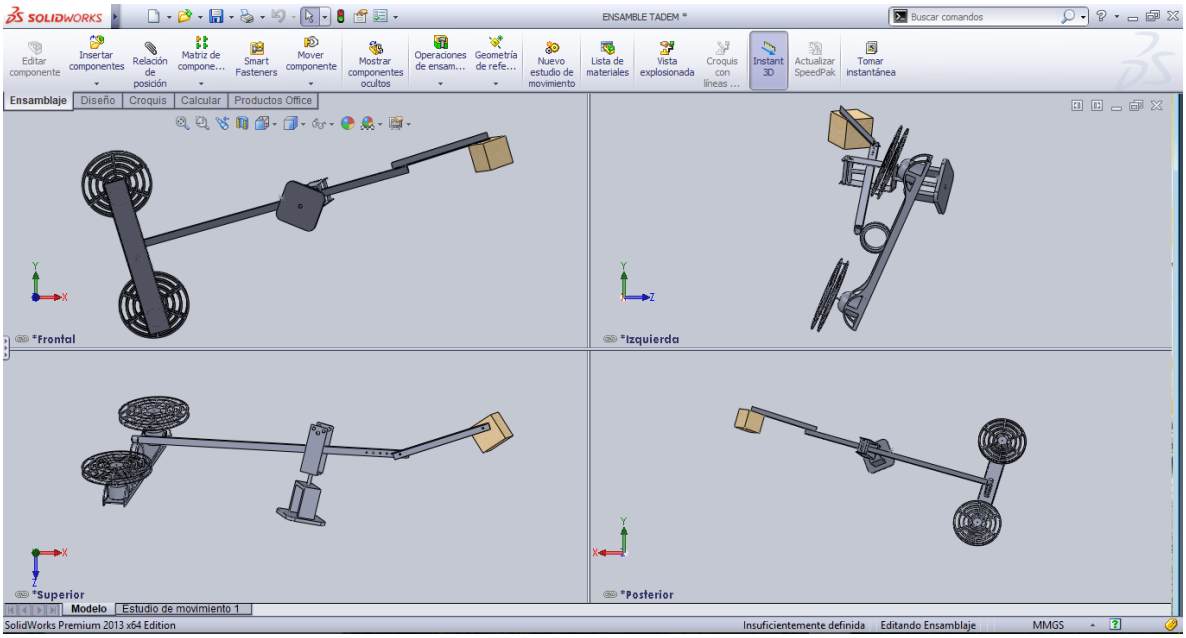

Figura 41: " 4 Vistas del ensamble de la base de pruebas"

Conclusión.

La construcción y control de vehículos aéreos es atractiva y demandante lo que implica que exista una gran diversidad de componentes e incluso de tipos y formas de los mismos vehículos, en este sentido muchas veces es complicado identificar que componentes utilizar debido a que cada marca y cada componente tiene características propias con las que, como usuario, debe familiarizarse esto se resume a una inversión considerable de tiempo para entender el o los lenguajes necesarios para comunicarse con el sistema o dispositivo de interés. Siendo esto un factor importante durante el desarrollo de este proyecto con el uso de la shield PXFmini que resultó desde un primer momento un dispositivo complicado en su ejecución y manipulación, como se mencionó, la primera complicación se presentó desde la instalación del sistema operativo del mismo componente ya que los fabricantes no dan el acceso a la descarga desde la página oficial de la marca Erle Robotics, es decir, no es software libre. Factores como el dicho, se aúnan a la novedad que tiene la Shield y por tanto el soporte limitado que le brindan al usuario.

Por otro lado, la complejidad de un sistema como el propuesto en este proyecto implica que el campo de estudio no se centre en un solo ámbito y permite la interacción con otras ramas y no puramente electrónica, cabe destacar que la movilidad de este tipo de vehículos es importante en su diseño mecánico. Se presentó una propuesta del sistema mecánico mediante Solidworks en el que permite los 3 grados de libertad.

Finalmente, este proyecto es de suma importancia para la comprensión y familiarización con los VANT´s desde un entorno interactivo así como de estudio y de una forma más amigable de entender conceptos de estabilidad y control.

Sugerencias

Sería interesante desarrollar este proyecto ahora con otras shields o autopilotos como la Navio o algún otro dispositivo que cuente con más soporte y que sea de software libre, además quienes se interesen en el desarrollo de un VANT deberán conocer conceptos básicos de lenguajes como Python para poder obtener y entender un resultado a corto plazo, por ejemplo, con la obtención de parámetros o de la telemetría arrojada por el dispositivo que se haya elegido y de esta forma poder utilizar y explotar al máximo las características u opciones que tendremos a nuestro alcance.

Referencias

Bibliográficas:

[1] Bodin, E., & Stenholm, F. (2015). Modelling & Control of a 3DOF Helicopter.

[2] David, B., Peter, W., Jeffrey, E., Allen, D., Chris, M., & Mattew, R. (2011). Learning to Program in Python.

[3] O'Kane, J. M. (2014). A gentle introduction to ROS.

[4] Martinez, A., & Fernández, E. (2013). Learning ROS for robotics programming. Packt Publishing Ltd.

Virtuales:

[5] Erle Robotics (2012 - 2018). “PXFmini”: Vitoria, España. Recuperado de: http://docs.erlerobotics.com/brains/pxfmini

[6] Erle Robotics (2012 - 2018). “Explicando MAVROS”: Vitoria, España. Recuperado de: http://erlerobotics.com/blog/ros-mavros-es/

[7] Erle Robotics (2012 - 2018). “Takeoff and land demo”: Vitoria, España. Recuperado de: http://forum.erlerobotics.com/t/takeoff-and-land-demo-on-pxfmini/1395

Índice de figuras.