Proyecto Fin de Grado
Ingeniería Electrónica, Robótica y Mecatrónica

MONTAJE Y CONFIGURACIÓN DEL DRONE COMERCIAL ERLE-COPTER

Autor: Mario Jiménez León
Tutores: Manuel Vargas Villanueva
Manuel Gil Ortega Linares

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017

Proyecto Fin de Grado
Ingeniería Electrónica, Robótica y Mecatrónica

MONTAJE Y CONFIGURACIÓN DEL DRONE COMERCIAL ERLE-COPTER

Autor:
Mario Jiménez León

Tutores:
Manuel Vargas Villanueva
Profesor titular
Manuel Gil Ortega Linares
Profesor titular

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2017

Proyecto Fin de Grado: MONTAJE Y CONFIGURACIÓN DEL DRONE COMERCIAL ERLE-COPTER

Autor: Mario Jiménez León

Tutores: Manuel Vargas Villanueva
Manuel Gil Ortega Linares

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2017

El Secretario del Tribunal

Resumen

El presente proyecto se ha pretendido que actúe a modo de manual. Un documento en el cual se recopila información y que consta de varias partes.

Por un lado, se presenta *Erle-Copter*, un vehículo quadrotor desarrollado por la compañía *Erle Robotics*, con el cual se ha tenido la suerte de trabajar físicamente. Se expondrán, por tanto, cada uno de los elementos que lo componen, los pasos seguidos para su montaje y su configuración interna necesaria para preparar al vehículo para futuros experimentos asegurando que éste vuela de manera correcta.

Por otro lado, se definen los programas necesarios para la manipulación y disposición de *Erle-Copter* y las distintas funciones que se pueden realizar con los mismos.

A continuación, se explica la configuración del sensor de flujo óptico necesaria y los pasos previos a realizar para un primer vuelo con éxito.

Del mismo modo, se ha recopilado información acerca del autopiloto *ArduCopter*, desarrollado por la compañía *ArduPilot*. Entre ella los controladores que éste utiliza para la manipulación de un quadrotor y los diferentes modos de vuelo de los que dispone. Dicho autopiloto es utilizado por un gran número de vehículos y de forma concreta en nuestro *Erle-Copter*.

Finalmente se expone alguna representación gráfica de las magnitudes a controlar en él, obtenidas a partir de la realización de distintos vuelos, con el objetivo de visualizar su comportamiento durante los mismos.

La mayor parte de la información dedicada a *Erle-Copter* que aquí se expone puede ser extendida y servir de ayuda en diferentes vehículos aéreos de tipo quadrotor.

Abstract

This proyect is able to act like a manual. A document which recopilates information and it is compose in many parts.

On the one hand, *Erle-Copter* is presented. A quadrotor vehicle developed by *Erle Robotics* company, with it we had luck to work physically. It is explained each elements which is composed, its steps required for his assembly and its internal configuration necessary to prepare the vehicle for future experiments.

On the other hand, the programs necesary for the *Erle-Copter´*s manipulation is defined as well as its different functions.

Consequently, it is explained optical flow sensor´s configuration and the previus steps to realized a succesful first flight.

Besides, some information is recopilated about *ArduCopter*, developed by *ArduPilot* company. Each controller which is used for the manipulation of a quadrotor and the differents flight modes.

Finally, some of its magnitudes graphics representations are exposed. Obtained with the realitation of different flights.

This information is dedicated for all tipes of quadcopters.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

A	Amperios
V	Voltios
cm	Centímetros
mm	Milímetros
g	Gramos
kg	Kilogramos
~	Aproximadamente
mAh	Miliamperios hora
"	Pulgadas
KV	KiloVoltio
Hz	Hercios
MHz/GHz	Mega/Giga Hercios
DIY	Do it yourself
ESC	Electronic Speed Control
RPM	Revoluciones por minuto
LiPo	Litio Polímero
CW	ClockWise (Giro en el sentido de las agujas del reloj)
CCW	CounterClockWise (Giro contrario a las agujas del reloj)
APM	ArduPilot Mega
CPU	Central Processing Unit
PCB	Printed Circuit Board
ROS	Robot Operating System
PWM	Pulse-Width Modulation
MAVLink	Micro Air Vehicle Communication Protocol
IMU	Inertial Measurement Unit
UART	Universal Asynchronous Receiver-Transmitter
MP	MegaPixel
R/C	Radio Control
DCM	Direction Cosine Matrix
EKF	Extended Kalman Filter

1 INTRODUCCIÓN

En este primer capítulo de la memoria se exponen, en dos secciones diferenciadas, una breve introducción del conjunto de circunstancias que rodean al proyecto y los distintos objetivos que han sido cumplidos una vez se ha finalizado el mismo.

1.1 Contexto

El presente proyecto se trata de la primera parte de uno de mayor amplitud, que unido al trabajo de fin de grado y master realizado por otros compañeros, posee como principales finalidades el montaje y configuración del vehículo quadrotor *Erle-Copter*, el control del mismo a través de una interfaz gráfica sin necesidad de emisora radio control y una copia, tanto del drone físico como de su control, en el entorno de simulación *Gazebo*.

En su totalidad, tanto el entorno de simulación como la interfaz gráfica de control del drone, se ha preparado en un ordenador de mesa propiedad del departamento de Sistemas y Automática en una máquina virtual con sistema operativo *Ubuntu* 14.04 para que pueda ser utilizado en futuros experimentos y/o proyectos.

El montaje del vehículo se ha realizado en el laboratorio de robótica de dicho departamento, excepto su configuración software, ya que fue necesario conectarlo a Internet a través de un dispositivo router. Por otro lado, las pruebas de vuelo con la emisora radio control, fueron llevadas a cabo en una zona de la escuela habilitada para ello, pudiendo ejecutarlas de manera segura.

1.2 Objetivos

Los propósitos fijados para mi parte del proyecto, se tratan de los pasos esenciales para la posible continuación de éste. Por tanto, los objetivos principales que se han perseguido han sido los siguientes:

- Dar forma física al vehículo mediante la unión de cada una de las piezas que lo componen.
- Establecer la configuración del *software* necesario, utilizado por su autopiloto.
- Confirmar el correcto funcionamiento del drone mediante distintas pruebas de vuelo utilizando la emisora radio control.

Por otra parte, se establecen otros que, aunque de menor importancia, son de gran utilidad a la hora de manipular el quadrotor. Estos son, el control del módulo de la cámara integrada en el vehículo para poder observar la imagen captada en tiempo real, la configuración y comprobación del funcionamiento del sensor de flujo óptico y una introducción al estudio del *software* autopiloto que *Erle-Copter* utiliza con el fin de comprender mejor su funcionamiento.

Todos estos objetivos fueron alcanzados de manera satisfactoria y el camino hacia ellos se expone, a continuación, en los distintos capítulos que componen la estructura de este documento.

2 *ERLE-COPTER*: COMPONENTES FÍSICOS

En este segundo capítulo se describirán todos los elementos que componen fisicamente el vehículo *Erle-Copter* y los sensores de posición que éste utiliza para poder conocer su ubicación en el espacio. A modo de introducción, se muestra una imagen del resultado final del vehículo junto con una serie de características que lo definen.

Ilustración 2-1. *Erle Copter, montaje completo*

La parte delantera del drone y dirección positiva del eje X se corresponde con la parte del frame de color rojizo.

Además, al tener pensado su uso para futuros experimentos, se ha incorporado un pequeño péndulo simple en su parte inferior y se ha modificado la posición del módulo de la cámara con el objetivo de que apunte al mismo para su posterior control. Del mismo modo, se ha colocado una pequeña lámpara de Leds para mejorar la iluminación en esa zona.

A continuación se exponen algunas de las características del vehículo, en ausencia de las patas:

Tabla 2-1. *Características del drone comercial Erle-Copter*

Característica	Descripción
Dimensiones	370 x 370 x 95 mm
Peso	878 g (batería incluida)
Carga útil	1 kg
Distancia entre ejes	730 mm
Hélices	10x4.5 o 9.4x4.3
Tiempo de vuelo	~ 20 min con batería de 5000 mAh
Color del frame	Blanco y rojo

2.1 Componentes esenciales

En esta sección se exponen cada uno de los elementos básicos que constituyen la parte *hardware* de *Erle-Copter*. La gran mayoría han sido obtenidos al comprar el *Erle-Copter DIY kit,* a través de la página web de la empresa *Erle Robotics*, en la cual me he apoyado para obtener información específica de cada uno de ellos. De manera similar se ha hecho con los sensores de posición y el elemento de control de nuestro drone. Estos últimos serán comentados en secciones posteriores.

2.1.1 Chasis

El chasis, también conocido como frame, es la estructura principal de *Erle-Copter* y de cualquier drone en general. Es un elemento de soporte primordial al cual van unidos el resto de componentes y aporta sustentación y rigidez al mismo. En este caso, está formado por cuatro brazos en cuyos extremos se colocarán los motores y dos láminas centrales que sirven de unión y base para el controlador, las patas y la batería, entre otros.

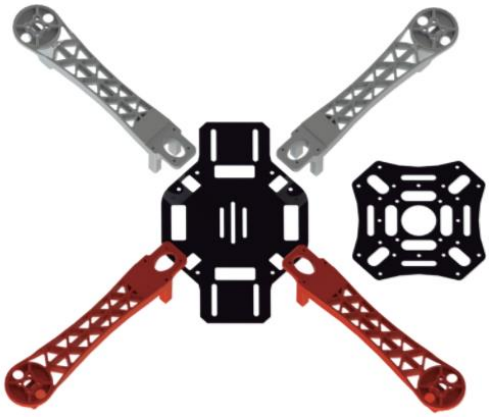


Ilustración 2-2. *Frame F450*

Algunas de las características de este frame son,

Tabla 2-2. *Características del frame F450*

Característica	Descripción
Color	Rojo / Blanco
Distancia entre ejes	450 mm / 17.7 "
Peso	248 g

2.1.2 Motores

Los motores convierten la energía eléctrica proveniente de la batería, en energía mecánica apreciada en forma de rotación y transmitida directamente a las hélices.

Estos motores son los únicos actuadores presentes en *Erle-Copter*. Modificando únicamente su velocidad de giro podemos variar su empuje.

Se han utilizado motores de edición limitada *MN2213* de la marca *T-Motor*. Se trata de motores sin escobillas, que apenas tienen rozamiento y que generan una menor pérdida de calor que hace que aumente la vida útil de los mismos.

A continuación se muestra una imagen de ellos y una breve selección de sus características,

Ilustración 2-3. *Motores MN2213 edición limitada 6º Aniversario*

Tabla 2-3. *Características de los motores MN2213 de T-Motor*

Característica	Descripción
Potencia	950 KV
Eje	3 mm
Peso	58 g
Tamaño estator	22 x 13 mm (diámetro x altura)

Debido a que la alimentación proporcionada por las baterías se trata de una corriente continua, necesitaremos algún dispositivo que permita convertirla en corriente trifásica admisible por los motores y regularla para controlar la velocidad de giro de éstos. Aquí, entran en juego los variadores.

2.1.3 Variadores

Los variadores o *ESCs*, son pequeños circuitos utilizados para variar la velocidad y la dirección de rotación de los motores. Generan la corriente adecuada para que estos aumenten o disminuyan sus *RPM*. Para nosotros es también el elemento que transmite información desde el *Erle-Brain 2* a los motores.

Ilustración 2-4. *Variadores SimonK30A*

A continuación se muestran algunas de sus características obtenidas de la página web de *Erle-Robotics*,

Tabla 2-4. *Características de los variadores SimonK30A*

Característica	Descripción
Corriente máxima	30 A
Rango de voltaje	2-4s *LiPo* (desde 7.4 hasta 14.8 V)
Peso	26.5 g (cables y enchufe incluidos)
Tamaño	50 x 25 x 11 mm

2.1.4 Hélices

Las hélices son el componente encargado de convertir el movimiento de rotación generado por los motores en propulsión.

Tal y como se ha especificado en la Tabla 2-1, las hélices que son utilizadas por *Erle-Copter* pueden ser de 10x4.5 o 9.4x4.3. Siendo el primer número el tamaño de la hélice de extremo a extremo en pulgadas y el segundo su ángulo de inclinación.

Ilustración 2-5. *Hélices CW y CWW*

Tabla 2-5. *Características de las hélices*

Característica	Descripción
Peso	0.100 Kg
Tipo/Material	Estandar/Plástico

2.1.5 Batería

La batería es la principal fuente de alimentación de nuestro drone. Ésta va conectada a *Erle-Brain 2* a través del módulo de potencia.

En el proyecto se cuenta con dos baterías 3S *LiPo* (litio y polímero) de 11.1 voltios y 5500 mAh. La letra S nos revela que se trata de una batería de 3.7 V en cada celda (3 x 3.7 = 11.1 V).

A modo informativo, comentar que las baterías *LiPo* son ligeras, de cualquier forma y tamaño, con una gran capacidad y una tasa de descarga elevada.

Ilustración 2-6. *Batería Floureon 3S LiPo 11.1V y 5500mAh*

2.1.6 Soporte para la batería

Componente de plástico que nos permite anclar la batería al drone y poder cambiarla facilmente.

Ilustración 2-7. *Soporte para la batería*

2.1.7 Módulo de potencia

El módulo de potencia o *Power Module* permite alimentar al controlador e informar del voltaje de la batería y la corriente. Su función es regular el voltaje de salida a un máximo de 5.3 V y a una intensidad máxima de 2.25 A. Además, permite la entrada de hasta 18 V y un máximo de 90 A.

Ilustración 2-8. *Módulo de potencia*

Tabla 2-6. *Características del módulo de potencia*

Característica	Descripción
Tensión máx entrada	18 V
Corriente máx entrada	90 A
Salidas de conmutación	5.3 V y 2.25 A como máximo

2.1.8 Mando radio control

Permite comunicarse con *Erle-Copter* para enviarle los movimientos deseados y poder interactuar con él. A través de sus sticks podemos controlar el empuje, *pitch*, *roll* y *yaw* además de poder armar y desarmar el drone. Como aspecto a destacar, cuenta con 6 canales diferentes.

Se usará también para cambiar de un modo de vuelo a otro mediante sus distintos interruptores.

Ilustración 2-9. *Emisora Turnigy TGY-i6*

2.1.9 Receptor de señal de radio

Recibe los valores del mando emisor y los transmite a *Erle-Brain 2* mediante codificación *PPM* (Modulación por posición de pulsos). Cuenta con seis canales y funciona en una banda de frecuencia de 2.4 GHz.

Ilustración 2-10. *Receptor de señal de radio Turnigy TGY-iA6*

2.1.10 Adaptador inalámbrico *WiFi*

Dispositivo de telemetría que nos permite conectarnos de forma inalámbrica a *Erle-Brain 2*. Éste establece un punto de acceso *WiFi* previamente creado y que, mediante el mismo, permite conectarnos al vehículo desde cualquier ordenador.

En el proyecto se han adquirido dos de estos dispositivos. Uno está incluido en el propio *Erle-Copter* y el otro en el ordenador de mesa establecido en el departamento, configurado del mismo modo, como punto de acceso.

Ilustración 2-11. *Edimax EW-7811UAC*

Tabla 2-7. *Características del módulo WiFi Edimax EW-7811UAC*

Característica	Descripción
Norma *IEEE*	802.11 ac
Ancho de banda	2.4 GHz
Peso	2 gramos / 0.001 libras

2.1.11 Patas

Patas de fibra de vidrio para *Erle-Copter*. Aunque son muy ligeras y apenas hacen variar el peso del vehículo, amortiguan considerablemente el golpe al aterrizar el mismo.

Ilustración 2-12. *Patas fiberglass de Erle-Copter*

Tabla 2-8. *Características de las patas fiberglass de Erle-Copter*

Característica	Descripción
Material	Fibra de vidrio
Altura	200 mm
Anchura entre piernas	325 mm

2.2 Sensores de posición

En esta sección se hará referencia a los sensores de posición que utiliza *Erle-Copter*. Nos centraremos en el *GPS* y el sensor de flujo óptico. Otros sensores de los que se sirve, como el acelerómetro o el giroscopio se encuentran incluidos en el controlador de vuelo *Erle-Brain 2*.

2.2.1 Conjunto *GPS* y brújula

El Sistema de Posicionamiento Glogal (*GPS*) es un sistema de navegación por satélite que nos ayudará a obtener la posición y orientación cardinal de *Erle-Copter*.

En *Erle Robotics* se utiliza un modelo que incorpora un *GPS uBlox Neo-8M* y una brújula digital. En el kit obtenido, ambos dispositivos se encuentran encapsulados en una carcasa para una mejor colocación en el drone y poder aislarlo de posibles interferencias con *Erle-Brain 2*. Dicho módulo, es compatible con el *software* autopiloto *ArduPilot* incorporado en el mismo.

Se muestra a continuación el *GPS* encapsulado y su colocación sobre el controlador *Erle-Brain 2*,

Ilustración 2-13. *Encapsulado GPS-Erle Brain 2*

La brújula digital vendría conectada al controlador a través de un puerto serie *I2C* del mismo mediante un cable *DF13* de 4 pines, mientras que el *GPS* lo hace mediante un puerto *UART*.

2.2.2 Sensor de flujo óptico

Este sensor se trata de una cámara de flujo óptico inteligente que, colocada apuntando hacia el suelo y unida a un giroscopio de 3 ejes, utiliza las diferentes texturas del suelo y otras características visibles para determinar la velocidad de un vehículo aéreo.

El sensor utilizado en el proyecto se denomina *PX4Flow*, en su versión 1.3 y ha sido desarrollado por la compañía *Pixhawk*.

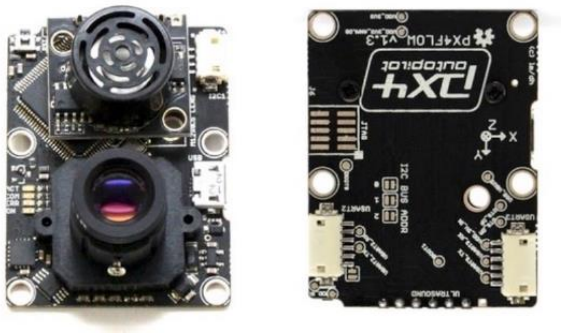


Ilustración 2-14. *Sensor de flujo óptico PX4Flow v1.3*

Este dispositivo cuenta además con la posibilidad de incorporar un sónar para complementar las medidas obtenidas por el sensor. Algunas carácterísticas de éste son,

Tabla 2-9. *Características del sensor de flujo óptico PX4Flow*

Característica	Descripción
Resolución	752 x 480 píxeles
Frecuencia de procesamiento	400 Hz
CPU	*Cortex M4F* de 168 MHz
Longitud de la lente	16 mm
Consumo de energía	115 mA / 5 V
Tamaño	45.5 mm x 35 mm

En *Erle-Copter*, el *PX4Flow* será utilizado para complementar las medidas del *GPS* con el fin de poder realizar vuelos en entornos tanto exteriores como interiores. Como veremos más adelante, será necesario configurarlo para su correcto funcionamiento.

A la hora de su colocación en el *Erle-Copter*, debemos asegurarnos que el eje Y del sensor posee la misma orientación que el eje X de *Erle-Brain 2* (orientación cero) para evitar medidas erroneas. Si se decidiera colocarlo de forma distinta habría que modificar el parámetro interno FLOW_ORIENT_YAW para tener en cuenta esa diferencia de orientación.

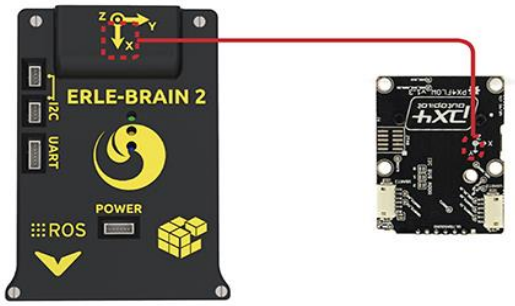


Ilustración 2-15. *Orientación adecuada del sensor de flujo óptico PX4Flow con respecto a Erle-Brain 2*

Por último, conectaríamos el sensor al controlador *Erle-Brain 2* a través de su otro puerto *I2C*, si hacemos memoria recordaremos que el otro puerto *I2C* fue utilizado para la brújula.

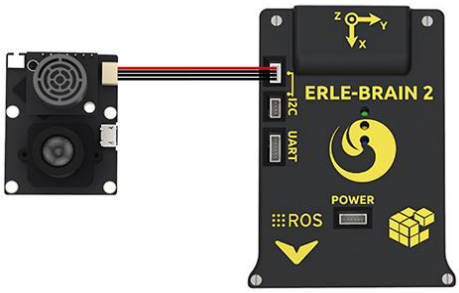


Ilustración 2-16. *Conexión del sensor de flujo óptico PX4Flow al controlador Erle-Brain 2*

2.3 *Erle-Brain 2*

- ¿Qué es *Erle-Brain 2*?

Tal y como se expone en la web del fabricante, se trata de la segunda generación del cerebro robótico artificial basado en *Linux* para la construcción de robots y *UAV* o aviones no tripulados. Cuenta con soporte oficial para *ROS*, sistema operativo para robots, y es compatible con una gran variedad de vehículos. Además utiliza *ArduPilot* como *software* de piloto automático para el control del drone.

Estructuralmente, está compuesto por una *Raspberry Pi 2* unida a una *PCB* denominada *PXFmini.*

- *PXFmini*

Erle-Robotics la define como un escudo de piloto automático de código abierto para *Raspberry Pi,* que te permite crear un autopiloto listo para volar con soporte para *ArduPilot*.

Está diseñado especialmente para *Raspberry Pi Zero*, pero es perfectamente compatible con sus modelos 1, 2 y 3.

Además de mejorar la conectividad mediante una distribución más ordenada e intuitiva de sus pines y conexiones, esta placa posee los distintos sensores internos que el controlador utiliza para el manejo del drone. Estos sensores son los siguientes:

- Sensor de gravedad de 3 ejes.
- Giroscopio de 3 ejes.
- Brújula digital de 3 ejes.
- Sensor de presión barométrico.
- Sensor de temperatura.
- *ADC* para la detección de la batería.

El modelo correspondiente a la *IMU* y al barómetro son, respectivamente, *MPU9250 IMU* y *MS5611 Barometer.*

Cuenta además con tres indicadores Leds (azul, verde y naranja) que configuran un código de estado que nos informa, entre otras cosas, si el piloto automático se encuentra encendido o si el vehículo esta armado o listo para armarse. Esto vendrá explicado más adelante.

A continuación, se muestran dos figuras en las cuales se pueden apreciar las distintas conexiones presentes en la placa así como su unión a la *Raspberry Pi 2*.

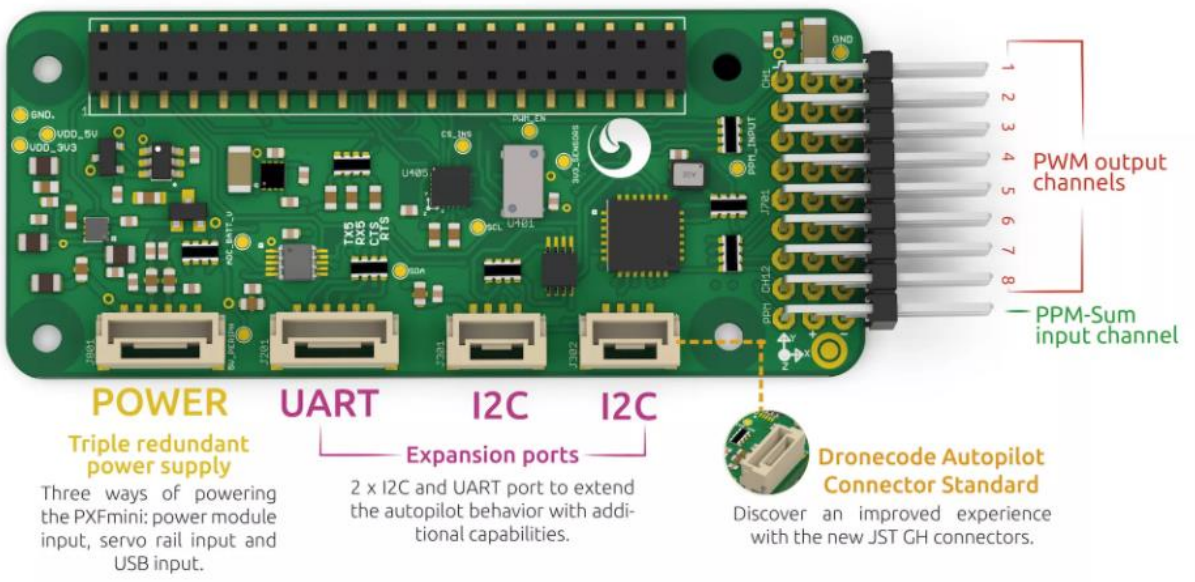


Ilustración 2-17. *Conexiones presentes en la placa PXFmini*

Ilustración 2-18. *Unión de la placa PXFmini a Raspberry Pi 2*

Tabla 2-10. *Dimensiones de la placa PXFmini*

Característica	Descripción
Peso	15 gramos / 0.033 libras
Tamaño	31 x 73 cm

- Conectividad

Al estar compuesto principalmente por una *Raspberry Pi 2,* ofrece mecanismos de conexión tales como puertos *I2C, UART, USB* o *Ethernet*. Además, cuenta con un módulo *PiCamera* de 5 MP para poder obtener en directo una visión de la misma. Más adelante veremos como usarla.

En la siguiente imagen puede observarse los diferentes pines y puertos con los que cuenta *Erle-Brain 2*,

Ilustración 2-19. *Conexiones presentes en el controlador Erle-Brain 2*

- Configuración

Al tratarse de una *Raspberry Pi 2* es necesario introducir en la misma un sistema basado en *Linux (Debian)* a través de una tarjeta SD. Una vez lo tenemos instalado, necesitaremos conectar nuestro *Erle-Brain 2* a Internet para proveerlo de la fisionomía de un quadrotor. De esto nos ocuparemos más adelante.

En la tabla siguiente podemos observar algunas de las especificaciones técnicas de este controlador proporcionadas por *Erle Robotics* en su página web,

Tabla 2-11. *Especificaciones técnicas de Erle-Brain 2*

Característica	Descripción
Dimensiones	63 x 96 x 25 mm
Peso	100 g
Procesador	*ARM Cortex-A7 CPU*, 4 nucleos a 900 MHz
Memoria RAM	1 GB
Cámara	5 MP
E/S	2x*I2C*, *UART*, conexión *POWER*, 4 puertos *USB*
	Puerto *HDMI*, *Ethernet* y conector de audio de 3.5 mm

3 *ERLE-COPTER:* MONTAJE FÍSICO

En este tercer capítulo se explica el procedimiento que se ha seguido para el montaje físico de nuestro *Erle-Copter*.

Una vez recibidos todos los componentes, el primer paso a realizar fue ensamblar nuestro drone. En principio, una tarea fácil ya que en este sentido *Erle Robotics* nos proporciona de manera digital una serie de manuales bastante completos para el montaje de cada uno de sus vehículos autónomos. En nuestro caso, seguiremos las instrucciones presentes en el apartado *"Erle-Copter assembly"* de su página web correspondientes a *Erle-Copter*.

Para adaptar nuestro vehículo a las necesidades que fueron apareciendo a lo largo del proyecto y pensando en experimentos posteriores, una vez montado tal y como dícen las instrucciones, fue modificada la posición de algunos elementos, por ejemplo, sacando el módulo *PiCamera* fuera de *Erle-Brain 2* para que éste apuntase hacia una posible pelota colgada en su parte inferior para su posible reconocimiento.

Ciertos incisos importantes a destacar en el montaje de nuestro drone son los siguientes,

3.1 Posición correcta de los motores

Tal como avisan las instrucciones, en nuestro *Erle-Copter DIY kit* se incluyen motores distintos a los que habitualmente se utilizan y, según el manual proporcionado por la compañía, deben ir colocados en una posición diferente. La colocación correcta de éstos se aprecia en la siguiente imagen,

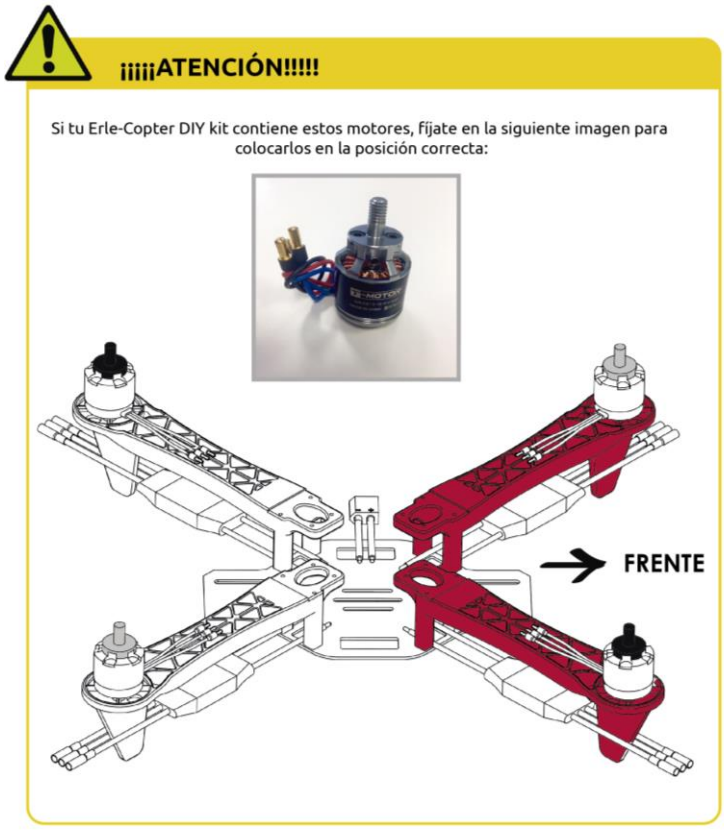


Ilustración 3-1. *Posición correcta de los motores en Erle-Copter*

Teniendo en cuenta la imagen anterior se asocia un número distinto a cada motor según su posición. De este modo, los motores 1 y 2 se corresponden con los motores de punta de color negro y los números 3 y 4 con los de punta de color blanco. Esta distribución se aprecia de mejor forma en esta otra ilustración,

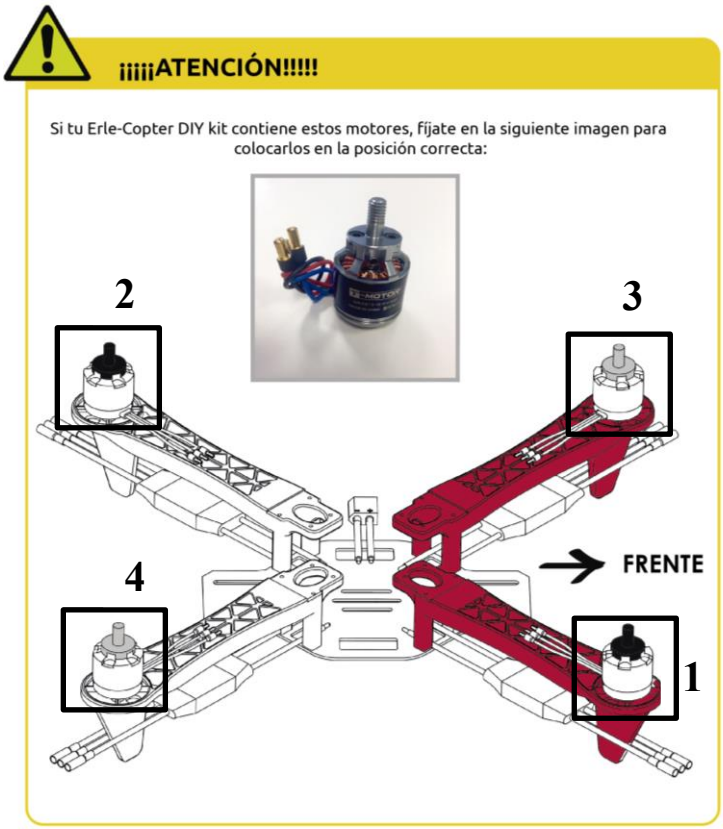


Ilustración 3-2. *Numeración de los motores en Erle-Copter*

3.2 Conexión de los motores a los variadores

Debido al uso de los motores *MN2213*, debemos considerar también una variación en su conexión con los distintos *ESCs*. Al igual que con la posición, para dicha conexión las instrucciones nos avisan, indicando cual es la forma correcta de conectar ambos componentes. Sin embargo, ese aviso induce a error ya que con dicha conexión los motores giran en dirección opuesta a la adecuada. La dirección de giro necesaria para que el drone se controle de forma satisfactoria aparece en la siguiente figura proporcionada por el foro de *Erle Robotics*,

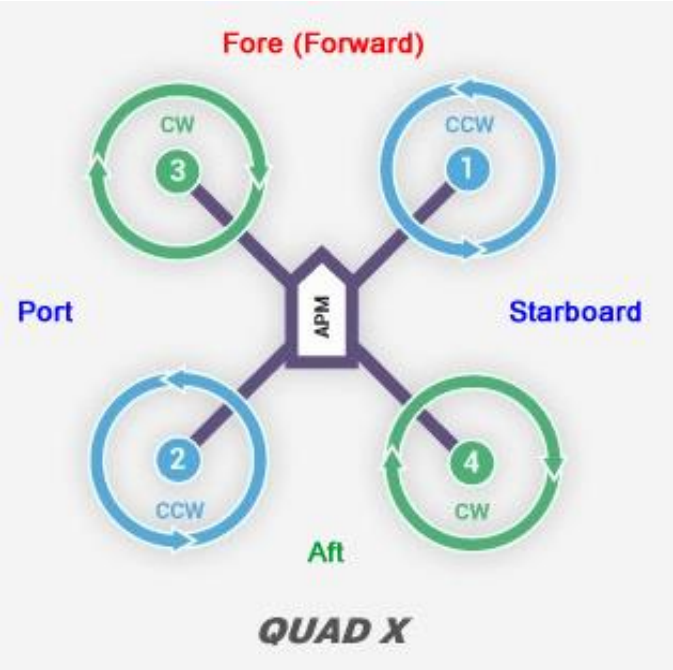


Ilustración 3-3. *Giro adecuado de los motores en un quadrotor*

Teniendo en cuenta la imagen anterior, una configuración general para la conexión entre variadores y motores, independientemente del modelo de éstos y teniendo en cuenta el sentido de giro, puede obtenerse de la siguiente tabla,

Tabla 3-1. *Conexión ESC-motores según el sentido de giro de los mismos*

Sentido de giro	Motores	Conexión
En contra de las agujas del reloj	Uno y dos	+ *ESC* y - Motor - *ESC* y + Motor
A favor de las agujas del reloj	Tres y cuatro	+ *ESC* y + Motor - *ESC* y - Motor

Por lo tanto, siendo de color rojo los cables positivos y de color negro los negativos en ambos componentes, en los motores número 1 y 2 se cruzan los cables para invertir la polaridad, mientras que en los números 3 y 4 se conectan de manera directa a los variadores.

Las conexiones expuestas en la Tabla 3-1 son válidas para cualquier tipo de vehículo quadrotor.

3.3 Conexión de los variadores a *Erle-Brain 2*

Para conectar los distintos variadores a *Erle-Brain 2* debemos tener en cuenta el orden de los distintos pines *PWM* disponibles en el controlador. Puede verse en la siguiente imagen,

Ilustración 3-4. *Orden de los pines PWM presentes en Erle-Brain 2*

Teniendo en cuenta la Ilustración 3-4, los motores irán conectados a los pines *PWM* del controlador de derecha a izquierda. Empezando por el motor número 1 en el extremo derecho, seguido por el 2, 3 y 4.

En el pin de entrada del extremo izquierdo (pin número 14) debe ir conectado el receptor de señal, tal y como puede verse en la siguiente ilustración,

Ilustración 3-5. *Modo de conexión del receptor de señal de radio*

4 *ERLE-COPTER*: PUESTA A PUNTO DE *ERLE-BRAIN 2*

En este cuarto capítulo se describe, por un lado, el *software* compatible y utilizado por *Erle-Brain 2.* Por otro, la manera en la que se configura internamente el controlador desde cero*,* es decir, cómo introducir en él la imagen del sistema operativo que utiliza y cómo instalarle el autopiloto adecuado.

4.1 *Software* del controlador

En esta sección se describe de forma breve el *software* soportado por *Erle-Brain 2* que ha influido en mayor medida a la hora de realizar el proyecto.

En la siguiente imagen, extraida de la página web de *ArduPilot*, se puede observar las numerosas características *software* que *Erle-Brain 2* puede manejar,

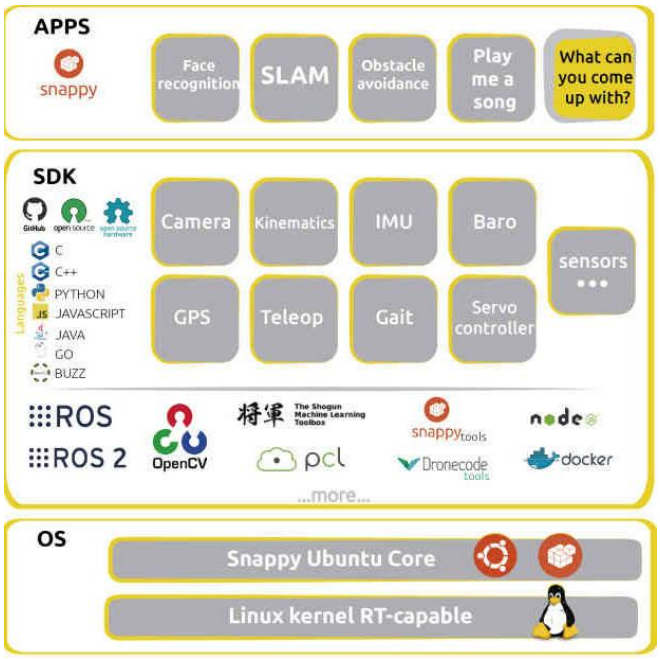


Ilustración 4-1. *Software compatible con Erle-Brain 2*

Teniendo esto en cuenta, a continuación se describen dos de ellos que nos afectan de manera más directa.

4.1.1 *Robot Operating System* (*ROS*)

Tal y como lo describe su página web, *ROS* es un conjunto de librerías *software* y herramientas que ayudan a los desarrolladores a la creación de nuevas aplicaciones para los robots.

Para *ROS,* el *hardware* es irrelevante. Se trata de un *software* de código abierto que nos facilita entre otras cosas controladores de dispositivos, librerías, herramientas de visualización, intercambio de mensajes entre procesos, etc.

Una función muy importante a la hora de utilizar *ROS* en nuestro proyecto, vendrá dada a la hora de controlar el módulo cámara de *Raspberry Pi 2*. Para llevar a cabo este proceso, es necesario instalar *ROS* en un PC con sistema operativo *Ubuntu* 14.04, ya sea en una partición del disco duro o utilizando una máquina virtual.

Erle Robotics cuenta con un entorno de simulación totalmente equipado para poder experimentar de manera segura y de identica forma a como lo haríamos con el drone real.

Los pasos de instalación del entorno se encuentran en la propia página de *Erle Robotics*, en la sección "*Support* → *Docs* → *Simulation* → *Configuring your environment"*. Para este proyecto no es necesario la instalación del entorno completo, pero sí la parte correspondiente a *ROS*.

Las instrucciones para su instalación en su versión *"Indigo"* se encuentran incluidas en los pasos de instalación del entorno. Éstos incluyen las herramientas básicas de *ROS*. Para una gama de paquetes más amplia, se recomienda la realización de los ejemplos presentes en la web.

Ilustración 4-2. *Robot Operating System*

4.1.2 *ArduPilot*

Tal y como lo define *Erle Robotics* en su web, *ArduPilot* es un *software* de piloto automático de código abierto que nos ayudará a controlar cualquier vehículo. Cuenta con características como: múltiples modos de vuelo, fácil de configurar al vehículo que queramos, programación a prueba de fallos, como que se pierda la señal, etc.

Del mismo modo, proporciona un *firmware* o conjunto de habilidades en función del *hardware* que estemos utilizando, en nuestro caso, un quadrotor y por tanto instalaremos el *firmware* denominado *ArduCopter*.

Destacar la flexibilidad con la que cuenta *ArduPilot* para distintos vehículos simplemente actualizando la última versión del *firmware* para uno determinado. Esto se lleva a cabo a través del programa *Mission Planner*, explicado más adelante y desarrollado por la misma compañía.

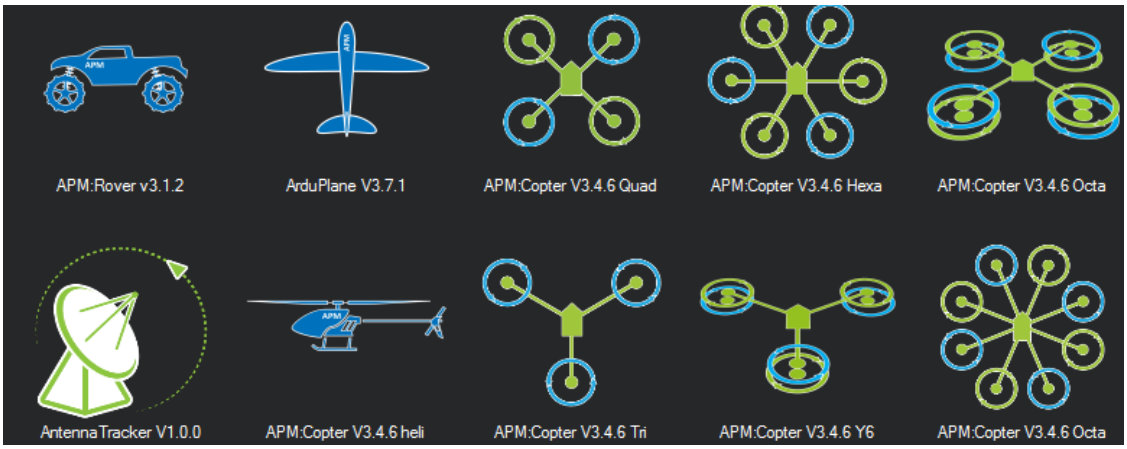


Ilustración 4-3. *Tipos de firmaware para ArduPilot*

4.2 Montaje de la imagen *Frambuesa* en la tarjeta SD

Esta sección resulta sencilla siguiendo los pasos presentes en la página web de la compañía. Pero antes de eso, necesitamos descargar la imagen que utiliza el *Erle-Brain 2* y cuyas principales características se reflejan en la siguiente tabla,

Tabla 4-1. *Especificaciones de la imagen del sistema Frambuesa*

Característica	Descripción
Nombre	*Frambuesa*
Versión	3.4
Fecha de lanzamiento	12/10/2016

Una vez se haya descargado, introducimos la tarjeta SD en nuestro ordenador a través de un adaptador e incluimos la imagen utilizando el programa *Win32DiskImager* (compatible para el sistema operativo *Windows*), tal como nos indican en la web.

Una vez tenemos la imagen preparada en la SD, la introducimos en *Erle-Brain 2*. Ahora mismo, lo que tenemos es un ordenador con un sistema operativo basado en *Linux*. Para aportar la fisionomía de un quadrotor, debemos elegir e instalar en él el autopiloto correspondiente al vehículo para el cual queremos que funcione, en nuestro caso *ArduCopter*.

4.2.1 Elección del autopiloto *ArduCopter*

Para instalar el autopiloto *ArduCopter* en la versión *Frambuesa* de *Erle-Brain 2,* es necesario conectar el controlador a Internet a través de un cable *Ethernet* utilizando un dispositivo router. Del mismo modo, es necesario conectarlo a un monitor mediante conexión *HDMI* junto con un ratón y un teclado *USB*, como se muestra en la imagen a continuación,

Ilustración 4-4. *Conexiones de Erle-Brain 2 a periféricos*

Además, para que *Erle-Brain 2* encienda es necesario alimentarlo utilizando un cable micro *USB*.

En versiones anteriores de la imagen incluida en el controlador, al conectar los periféricos al mismo, aparecía automáticamente en la pantalla, una serie de opciones que perimtían elegir el autopiloto deseado. Esto en la imagen *Frambuesa* cambia mostrando únicamente el escritorio. Una vez en él, abrimos un terminal y ejecutamos el siguiente comando:

sudo apt-get install apm-copter-erlebrain

Y reiniciar.

sudo reboot

Con esto ya tendremos correctamente instalado el piloto automático en el controlador. Este proceso incluye además los parámetros necesarios para volar correctamente nuestro drone con la emisora *TGY-i6*.

4.3 Control del módulo *PiCamera*

Una de las aplicaciones en las cuales haremos uso de la plataforma *ROS* en nuestro sistema *Ubuntu* del PC será para tomar el control de la cámara integrada en *Erle-Brain 2,* cuya posición hemos modificado para que apunte hacía el suelo y poder ver en tiempo real lo que esta está captando.

A continuación se explican los pasos a seguir en ambas plataformas,

- **Dentro de *Erle-Brain 2***

Para poder llevarlo a cabo, es necesario tener instalados en el controlador *Erle-Brain 2* los paquetes que *ROS* utiliza para la manipulación de dicho módulo. Es indispensable, por tanto, conectar el controlador a Internet del mismo modo a como se hizo tras montar la imagen *Frambuesa* en la tarjeta SD para elegir el autopiloto.

Una vez tenemos todo conectado según la Ilustración 4-4, introducimos lo siguiente,

sudo apt-get update

sudo apt-get install ros-raspicam-erlerobotics

sudo apt-get install ros-camera-web-browser-erlerobotics

El primer comando comprueba si existen nuevos repositorios que puedan ser instalados en nuestro sistema y los dos siguientes instalan los paquetes de *ROS* necesarios para la manipulación de la cámara.

- **Dentro de *Ubuntu 14.04* en nuestro PC**

Una vez tengamos instalado *ROS* en nuestro sistema, tendremos las herramientas necesarias parar manipular la *PiCamera* a través de esta plataforma.

Como configuración previa, fue necesario modificar algunos de los archivos presentes en nuestro sistema del siguiente modo:

- Archivo .bashrc: al final de este escribimos lo siguiente,

  ROS_MASTER_URI= http://10.0.0.1:11311

  ROS_IP= 10.0.0.2

Siendo 10.0.0.1 la dirección IP de *Erle-Brain 2,* 10.0.0.2 la de nuesto PC y 11311 el puerto a utilizar. Puede comprobarse en la siguiente imagen,

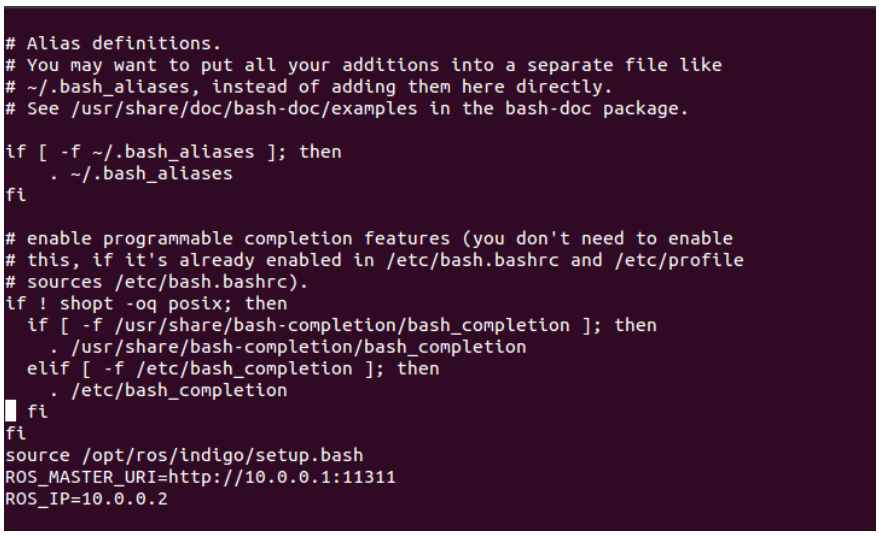


Ilustración 4-5. *Modificación del archivo .basrc*

- Archivo /etc/hosts: colocar debajo de la dirección IP del PC lo siguiente,

  10.0.0.1 erle-brain-2

Del mismo modo que se observa en la imagen inferior,

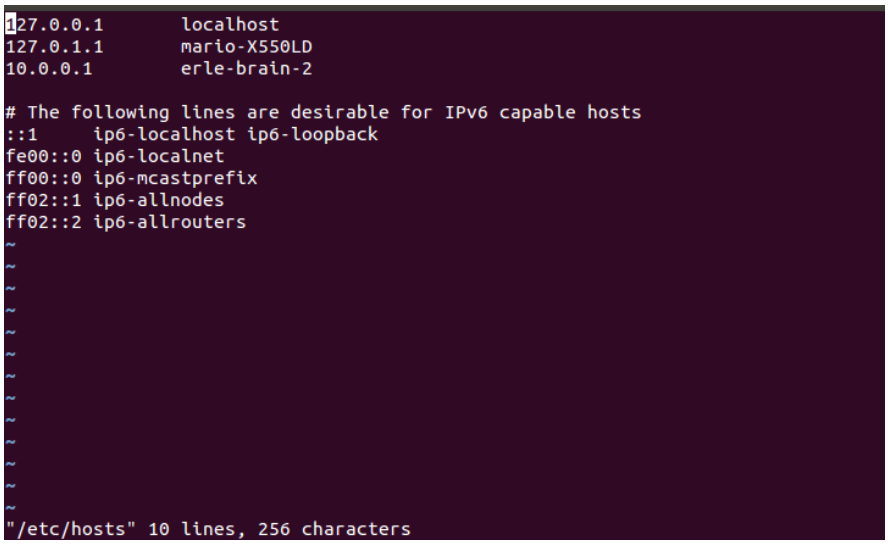


Ilustración 4-6. *Modificación del archivo hosts*

Una vez modificados dichos archivos, para iniciar el control de la cámara, debemos estar conectados a la red *"Erle-Robotics-Frambuesa"* y utilizar los siguientes comandos:

rosservice call /camera/start_capture

Para encender la cámara. En ésta, se encenderá de manera fija un Led de color rojo.

rqt_image_view

Con este comando se abre una nueva interfaz que nos permite ver lo que la cámara está captando en tiempo real.

rosservice call /camera/stop_capture

Para apagar la cámara. El Led rojo dejará de lucir indicándonos que la cámara está apagada.

En la siguiente ilustración puede comprobarse la correcta visualización de la cámara,

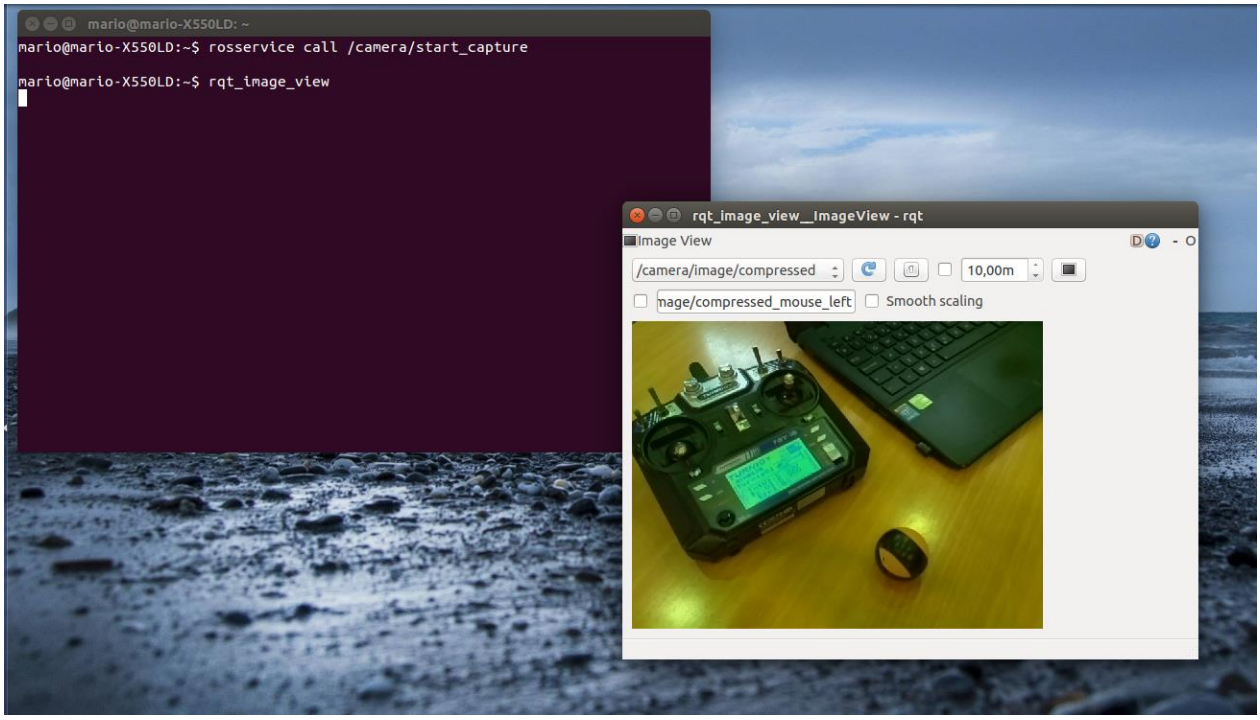


Ilustración 4-7. *Visualización de la imagen captada por la PiCamera utilizando ROS*

5 SOFTWARE

En este quinto capítulo, se incluye una breve descripción de los programas que pueden ser utilizados al manipular un vehículo quadrotor y las funciones que éstos proporcionan. Dichas funciones han sido utilizadas durante la realización del proyecto y serán explicadas de manera más particular aplicadas a *Erle-Copter*.

5.1 Programas utilizados

En la siguiente sección haremos referencia a los programas utilizados durante la realización del proyecto. Otros programas como *Win32DiskImager*, para escribir una imagen en una tarjeta SD, no necesitan de una mayor explicación.

5.1.1 *Mission Planner*

Mission Planner se trata de un programa que usaremos a modo de estación de tierra y que presenta todas las funciones necesarias para utilizar *ArduPilot*. Además nos informará del estado del vehículo en todo momento.

Existen otros programas similares como *APM Planner* o *QGroundControl,* este explicado más adelante.

La version utilizada y probada del programa *Mission Planner* es 1.3.39.

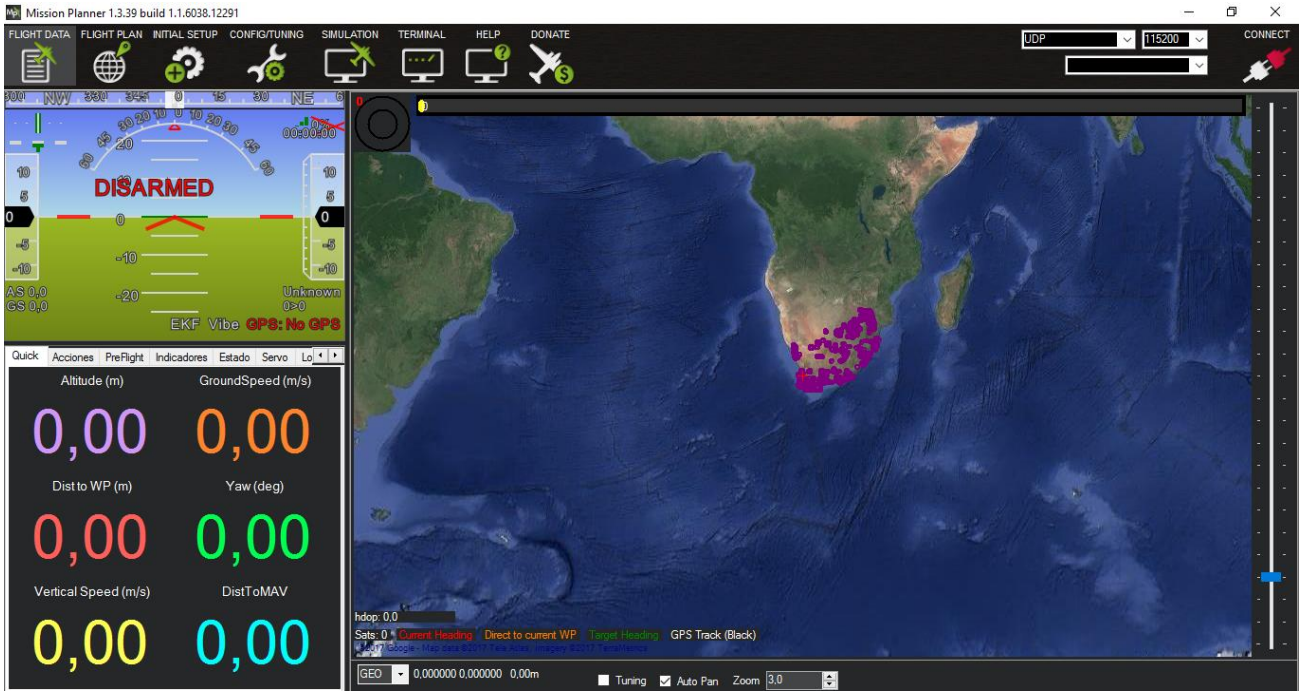


Ilustración 5-1. *Mission Planner v1.3.39*

Entre otras, las actividades que realizaremos con él serán:

- **Conexión *MAVLink* con el controlador.**

En este caso, para poder conectar *Erle-Brain 2* a *Mission Planner* necesitaremos que el PC donde tengamos instalado el programa se encuentre conectado a la red inalámbrica*"Erle-Robotics-Frambuesa"* creada por el controlador *Erle-Brain 2.* Este mostrará automaticamente dicha red una vez haga un largo pitido después de haber sido encendido. El Led azul de la antena *WiFi* brillará de forma intermitente cuando esté disponible.

Una vez estamos incluidos en la red inalámbrica podremos conectarnos a *Mission Planner* a través de un protocolo *UDP*. Para poder efectuar la conexión con éxito, necesitamos conocer la dirección IP de nuestro controlador y el puerto por el cual se va a realizar. Ambos datos vienen incluidos en la siguiente tabla,

Tabla 5-1. *Dirección IP de Erle-Brain 2 y puerto UDP*

Dirección IP	10.0.0.1
Puerto UDP	6000

Para iniciar la conexión, si nos fijamos en la Ilustración 5-1, debemos colocar el número y tipo de puerto en las pestañas de la parte superior derecha de la imagen y pulsar en el icono *"CONNECT".*

- **Modificar, guardar o cargar parámetros de vuelo en el quadrotor.**

Con ello podremos modificar la manera de volar de nuestro drone. Siempre que cambiemos un parámetro debemos pulsar la opción *"Write parameters"* a la derecha de la pantalla, si no el nuevo valor no se guardará.

De manera similar, podremos descargarlos en un archivo *.param* pulsando el botón "*Save*". Estos parámetros pueden ser usados para incluirlos en el entorno de simulación de *Erle-Copter* o simplemente como copia de seguridad.

- **Establecer los modos de vuelo en el canal deseado.**

En la pestaña *"Initial Setup → Mandatory Hardware → Flight Modes"* podremos establecer distintos modos de vuelo para los distintos canales disponibles. Una vez configurados, podremos cambiar de uno a otro utilizando los interruptores del mando control remoto.

- **Preparación del vehículo para su primer vuelo.**

Mission Planner nos ayuda a preconfigurar nuestro *Erle-Copter* para realizar un primer vuelo con éxito. En este sentido, el programa nos permitirá, a través de la pestaña *"Initial Setup",* calibrar la emisora, el acelerómetro, la brújula, habilitar o deshabilitar el *PX4Flow*, etc.

- **Descargar y visualizar *logs* para su posterior estudio.**

¿Qué es un *log*?

Se trata de un registro de todos los acontecimientos que afectan a un proceso en particular. En nuestro caso, quedarán guardados todos los valores captados por la telemetría y cuyas gráficas podremos estudiar.

¿Cómo se guarda, descarga y visualiza un *log*?

Erle-Copter comenzará a tomar medidas en el intervalo en el que éste se encuentre armado. Es decir, al armarse comienza a guardar los valores captados por los sensores, y deja de hacerlo una vez se desarme.

El primer paso para visualizar las gráficas obtenidas es descargar el *log* desde *Erle-Brain 2* a nuestro PC. Para ello, en la pestaña *"Flight Data"* del programa *Mission Planner* buscamos la opción *"DataFlash Logs"* en la interfaz inferior izquierda, tal y como muestra la imagen,

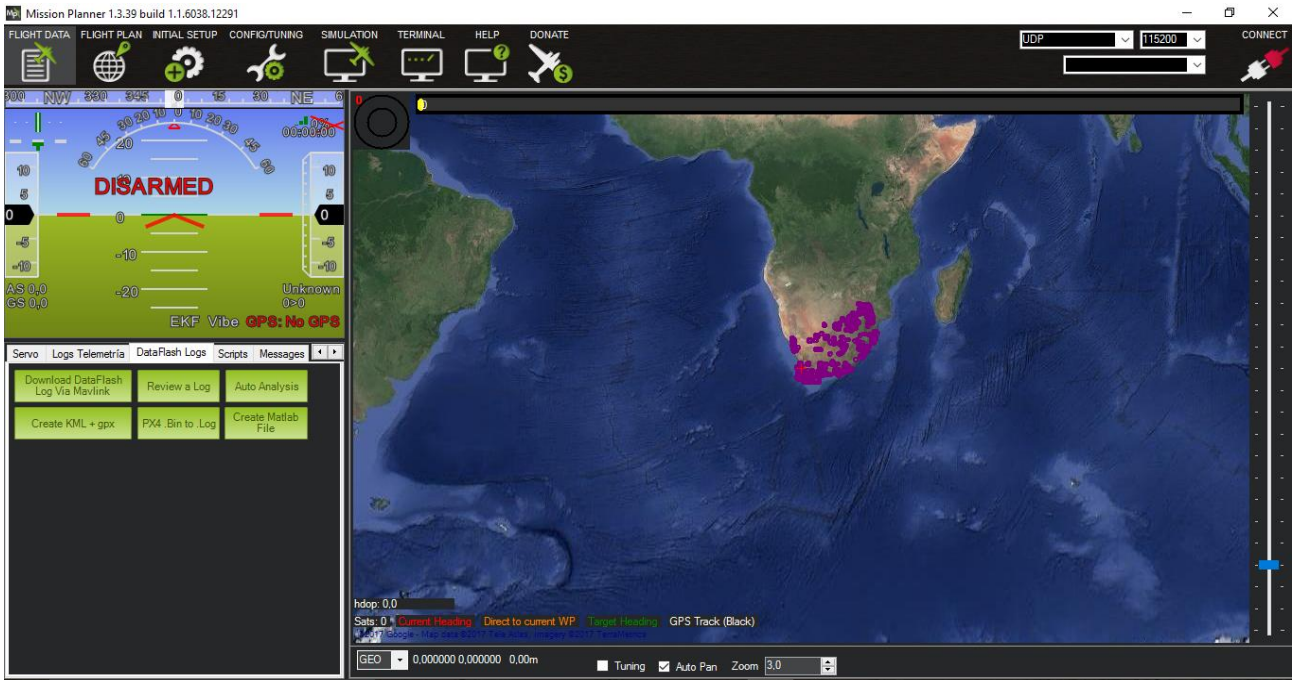


Ilustración 5-2. *Opción DataFlashLogs en Mission Planner*

Pulsamos *"Download DataFlash Log Via Mavlink"*, seleccionamos el log deseado y lo descargamos. *Mission Planner* lo convertirá en un archivo tipo *.bin.log* (archivo en formato de texto) para poder visualizarlo de manera gráfica.

Una vez lo tengamos descargado en nuestro PC, los archivos se guardarán en la carpeta *"QUADROTOR"* → *"1"*. Para visualizarlos, pulsamos *"Review a Log"* y buscamos el archivo *.bin.log* en dicha carpeta.

- **Visualizar un *log* en *MatLab***

Una opción muy útil e interesante, a la par que sencilla, que ofrece *Mission Planner* es la posibilidad de generar un archivo compatible con *MatLab* a partir del *log* registrado en el drone. Fijándonos de nuevo en la imagen 5-2 basta con seleccionar la opción *"Create MatLab File"* y seleccionar el archivo *.bin.log* que se desea convertir. Esto dará lugar a un archivo *.mat* del mismo nombre que podremos abrir y manipular con *MatLab*.

5.1.2 QGroundControl

Al igual que *Mission Planner*, *QGroundControl* actúa como estación de tierra y presenta unas funciones muy similares. La diferencia entre ambos, es que éste ha sido desarrollado por la compañía *PixHawk* mientras que *Mission Planner* lo ha sido por el proyecto *ArduPilot*.

Basicamente, este programa ha sido utilizado para introducir de forma sencilla los parámetros que configuran el *PX4Flow*. Esto viene explicado en el siguiente capítulo.

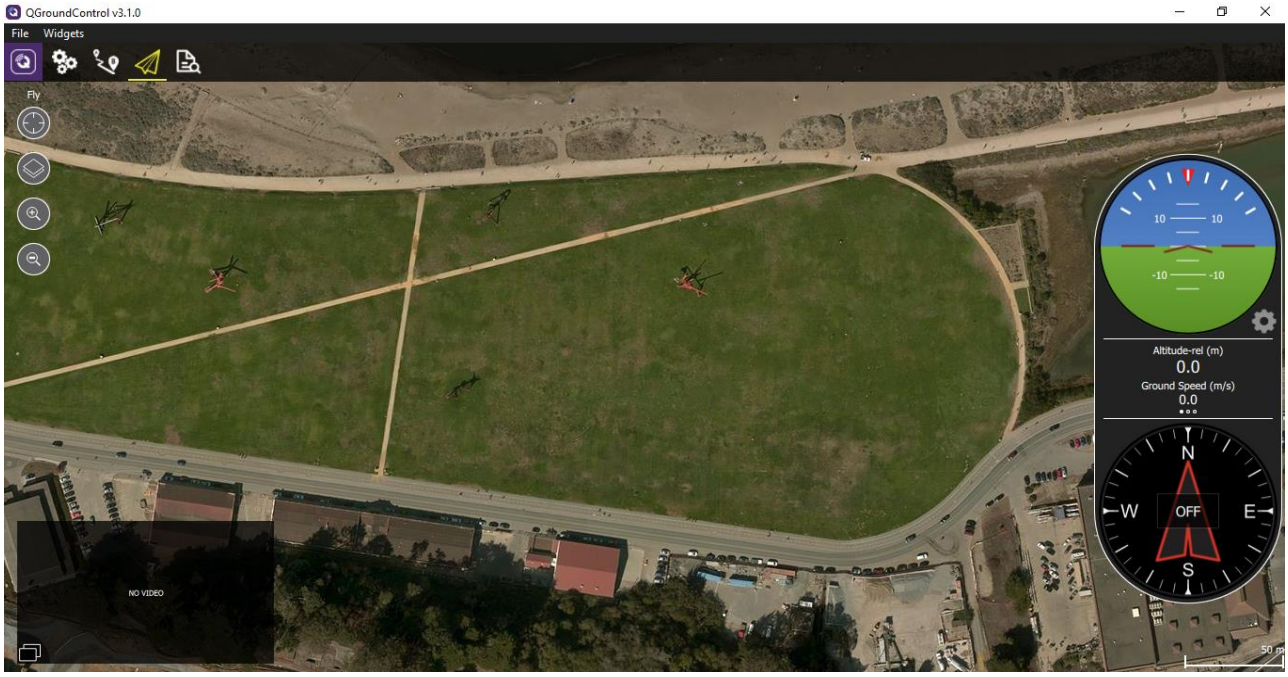


Ilustración 5-3. *QGroundControl*

5.1.3 *Putty* y conexión *SSH* con Erle-Brain 2

Utilizaremos dicho programa, compatible con el sistema operativo *Windows*, en caso de que queramos introducirnos en el sistema del controlador *Erle-Brain 2* de forma remota. *Putty* utiliza el protocolo *SSH* por lo que necesitaremos introducir la dirección IP presente en la Tabla 5-1, pidiendo usuario y contraseña para poder establecer la conexión, los cuales se presentan a continuación,

Tabla 5-2. *Usuario y contraseña para Erle-Brain 2*

Usuario	erle
Contraseña	holaerle

El resto de opciones, como el número 22 del puerto, se dejan por defecto.

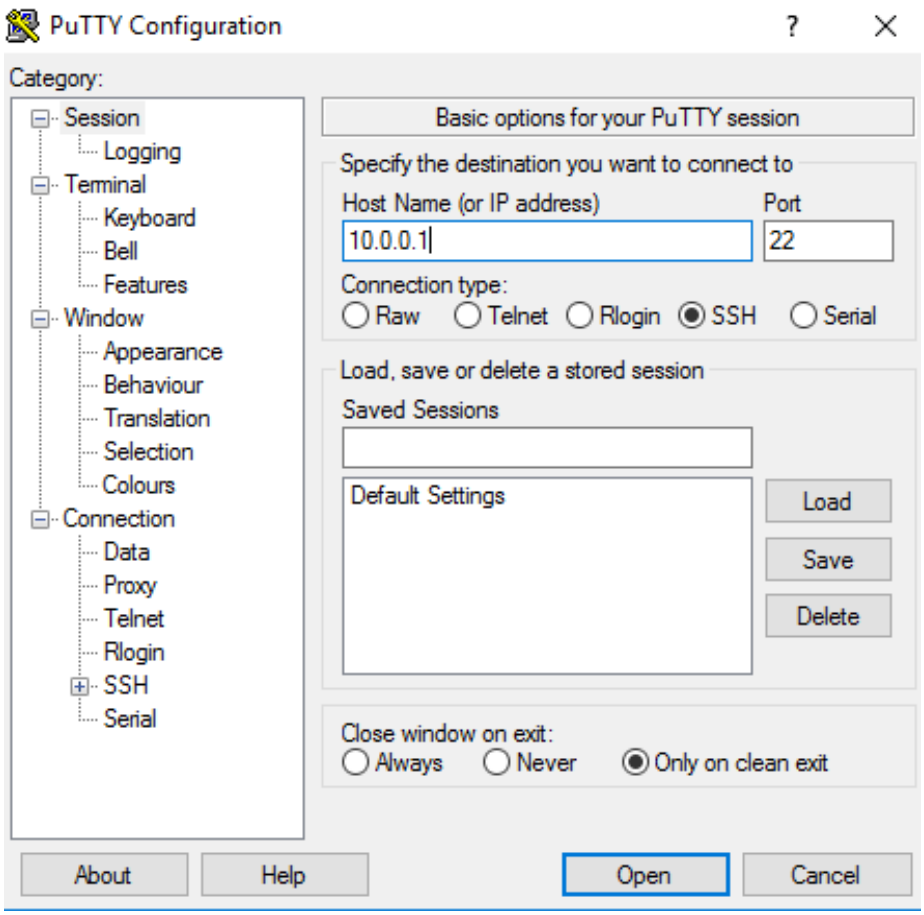


Ilustración 5-4. *Conexión con el controlador Erle-Brain 2 desde el programa Putty*

6 CONFIGURACIÓN DEL SENSOR DE FLUJO ÓPTICO

El sensor de flujo óptico es utilizado para poder realizar vuelos, no solo en espacios abiertos, sino también en cerrados, complementando o sustituyendo las medidas del *GPS*. En este sexto cápitulo, se exponen los pasos seguidos para la configuración del sensor de flujo óptico *Px4Flow* con el que poder obtener medidas apropiadas una vez lo conectemos a *Erle-Copter*.

Para la realización del proyecto hizo falta utilizar dos sensores de flujo óptico en dos versiones distintas.

Por un lado, su versión 1.3.1 se corresponde con el sensor incluido en el kit adquirido junto con el resto de los componentes. Aun habiéndolo configurado de forma correcta, no se consiguió que éste tomara medidas de ningún tipo. Al principio se pensó en una posible incompatibilidad con la nueva imagen *Frambuesa* instalada en *Erle-Brain 2,* pero estando conectado de forma aislada tampoco funcionaba. Por tanto, se llegó a la conclusión de que estaba defectuoso y se decidió utilizar una versión anterior cuyo funcionamiento se había comprobado con anterioridad. Este segundo sensor, en su versión 1.3 funcionaba perfectamente.

En las siguientes ilustraciones podemos comparar las medidas tomadas por cada uno de los *PX4Flow*, ambos fueron conectados de manera individual y mediante cable micro *USB* al programa *QGroundControl*. Una vez dentro de éste, abrimos el desplegable *"Widgtes"* de la parte superior izquierda de la pantalla y ahí seleccionamos la opción *"Analyze"*. Esto nos abrirá una ventana donde podremos observar de forma gráfica los diferentes valores que está captando el sensor en ese mismo momento.

Como se puede observar en las siguientes imagenes, comprobamos cómo el sensor defectuoso no capta valores de velocidad. Sin embargo, el sensor en su versión 1.3 los obtiene de manera correcta.

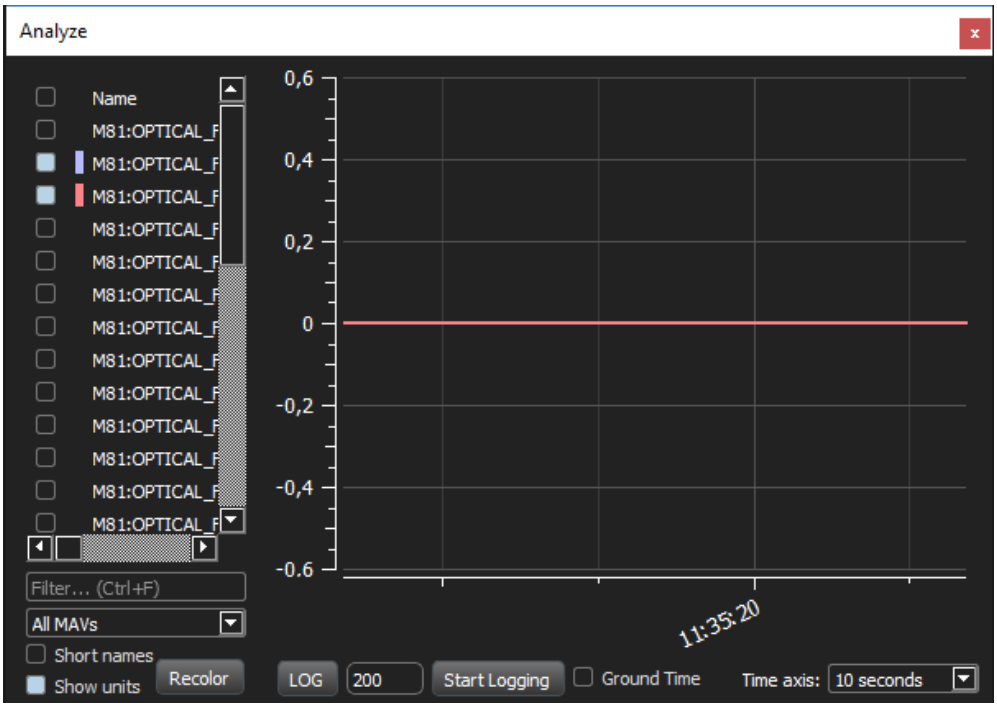


Ilustración 6-1. *Medidas obtenidas por el sensor PX4Flow v1.3.1*

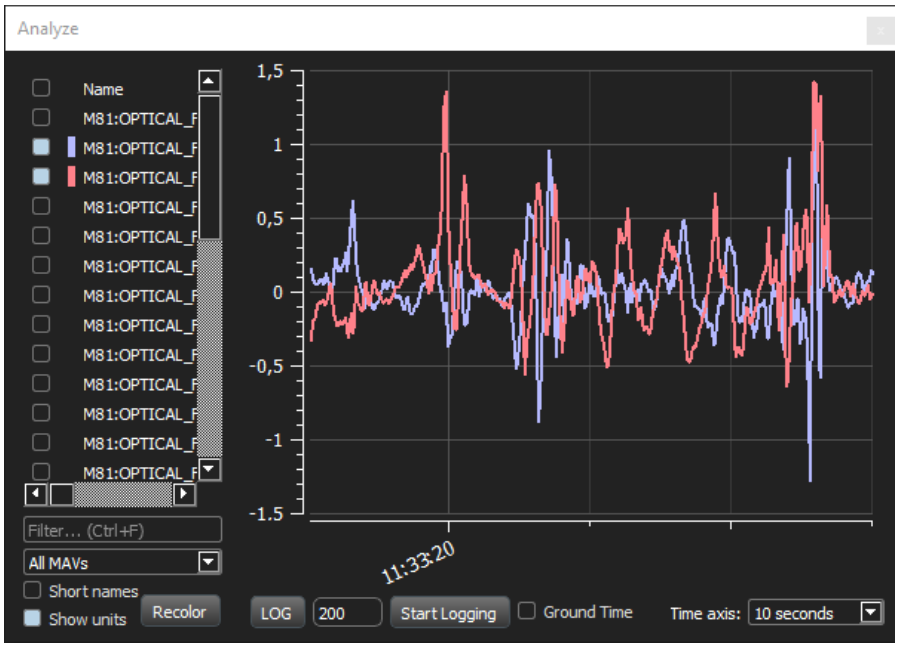


Ilustración 6-2. *Medidas obtenidas por el sensor PX4Flow v1.3*

A continuación se numeran los pasos seguidos, basándome en los encontrados en la página web de *ArduPilot* y *PixHawk*, para la correcta configuración del sensor de flujo óptico.

6.1 Actualización del *firmware* en el sensor *PX4Flow*

Para configurar el sensor de flujo óptico con los parámetros adecuados utilizaremos el programa *QGroundControl* en su versión 3.1.0. Debemos conectarlo a nuestro PC a través de un cable micro *USB* del siguiente modo:

Una vez dentro del programa pulsamos en el icono *"Vehicle Setup"*, cuyo símbolo es un engranaje de tres ruedas dentadas. A la izquierda de esta ventana nos aparecen dos opciones:

- *"Summary"*: si conectamos el sensor teniendo abierta esta ventana, el programa nos mostrará los parámetros que tiene cargados en ese momento.
- *"Firmware"*: conectando el sensor en esta opción podremos cargar en él los parámetros actualizados.

Al conectarlo en esta última, el programa nos preguntará qué versión del *firmware* queremos descargar, seleccionamos la que viene por defecto, "*standar version (stable)"*, y pulsamos "*Ok"*. El programa comenzará a descargarlo de la web del fabricante e introducirlo en nuestro sensor. Una vez cargado completamente, junto a las pestañas "*Summary"* y "*Firmware"* aparecen otras dos, "*PX4"* y "*Parameters"*.

6.2 Compatibilizar el sensor *PX4Flow* con el autopiloto *ArduCopter*

Debido a que el sensor pertenece a la compañía *PixHawk*, los parámetros descargados e incluidos en el mismo viene predeterminados para utilizarse en su propio controlador. Para poder usarse en *Erle-Brain 2* y por tanto hacerlo compatible con *ArduPilot* debemos modificar el parámetro SYS_AP_TYPE en la pestaña "*Parameters*".

Con este parámetro definimos el tipo de autopiloto en el que vamos a incluir el sensor de flujo óptico. Su valor por defecto es 0, compatible por tanto con el autopiloto *PixHawk*. Para que sea compatible con *ArduPilot* debemos asignarle un valor de 12. Este valor aparece en la página web de *ArduPilot* en su sección dedicada al sensor *PX4Flow*.

Los distintos parámetros pueden observarse en la siguiente ilustración,

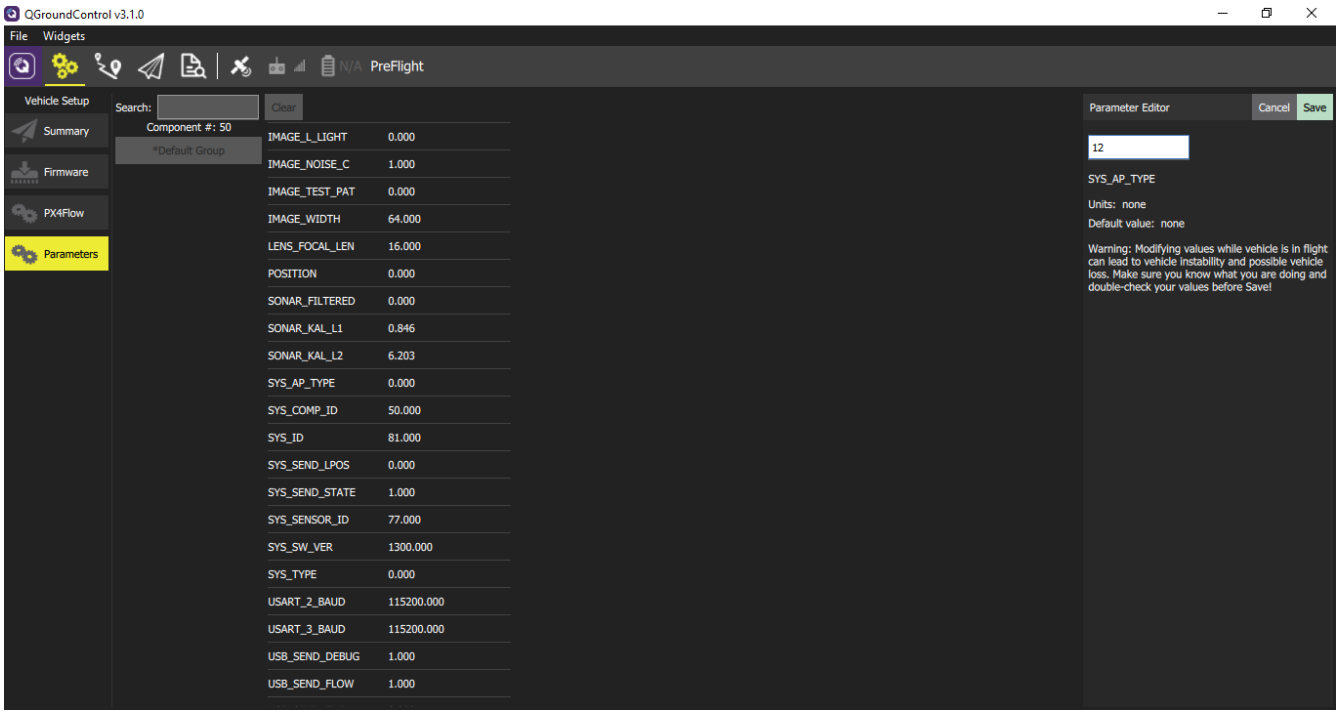


Ilustración 6-3. *Lista de parámetros del sensor PX4Flow en QGroundControl*

6.3 Enfoque de la lente

Estos sensores se envían normalmente sin estar enfocados. Para hacerlo realizamos los siguientes pasos,

1. Después de cargar el *firmware* y modificar el parámetros que determina el tipo de autopiloto, necesitamos enfocar la lente. Para ayudarnos en este proceso se modifica el valor del parámetro VIDEO_ONLY a "1", esto hará que la imagen captada por el sensor se visualice con una mayor resolución, facilitando el enfoque de la cámara.
2. Pulsamos la pestaña "*PX4*" y retiramos la tapa de la lente. Nos aparecerá una nueva ventana donde podremos ver la imagen en tiempo real captada por el sensor.
3. Apuntamos a un objeto de alto contraste situado entre uno y tres metros de distancia.
4. Ajustamos el enfoque de la lente hasta que la imagen aparezca de forma clara y fijamos dicha posición utilizando la rueda roscada que posee la misma.
5. Una vez enfocada la lente, volvemos a designar su valor inicial de cero al parámetro VIDEO_ONLY.

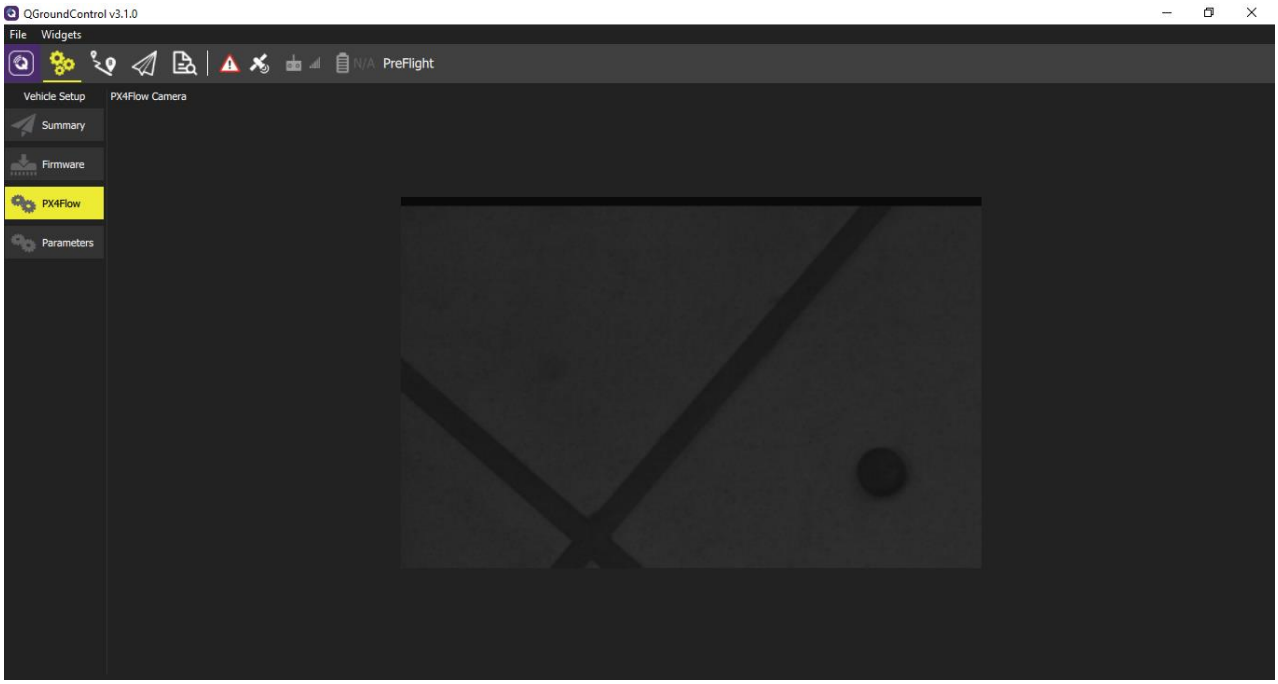


Ilustración 6-4. *Enfoque de la lente del sensor PX4Flow*

En la figura anterior se ha querido visualizar la imagen captada por el sensor de flujo óptico, al enfocar su lente a una distancia de aproximadamente un metro del suelo.

6.4 Conexión con *Erle-Brain 2*

Una vez tenemos el sensor *PX4Flow* preparado podemos conectarlo a nuestro controlador a través del puerto *I2C* que queda libre, como se muestra en la Ilustración 2-16.

Para comprobar que el dispositivo funciona correctamente, comprobamos que la dirección *I2C* del sensor de flujo óptico aparece en el bus. Para esto, conectamos nuestro PC a *Erle-Brain 2* a través del programa *Putty* y escribimos el siguiente comando, proporcionado en la web de *Erle Robotics,* en el terminal:

sudo i2cdetect –r 1

Debería poder verse la dirección 0x42 tal y como muestra esta imagen,

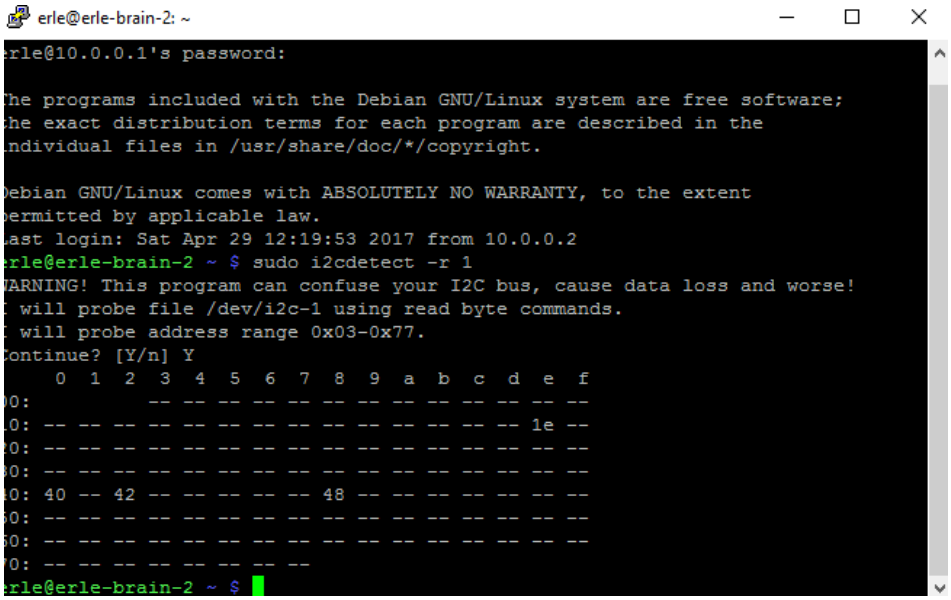


Ilustración 6-5. *Comprobación del puerto I2C utilizado por el sensor PX4Flow*

6.5 Ajuste de parámetros en *Mission Planner*

El sensor puede activarse mientras estemos conectados al programa *Mission Planner*. Para ello, en la ventana *"Initial Setup → Optional Setup → Optical Flow"*, seleccionamos la casilla "*Habilitar el flujo óptico"*. En algunas versiones, esta opción aparece en inglés como *"Enable Optical Flow"*.

A modo alternativo, podemos habilitarlo ajustando el parámetro FLOW_ENABLE a "1" a través de la lista completa de parámetros.

Para que el sensor se inicialice es necesario reiniciar *Erle-Brain 2*.

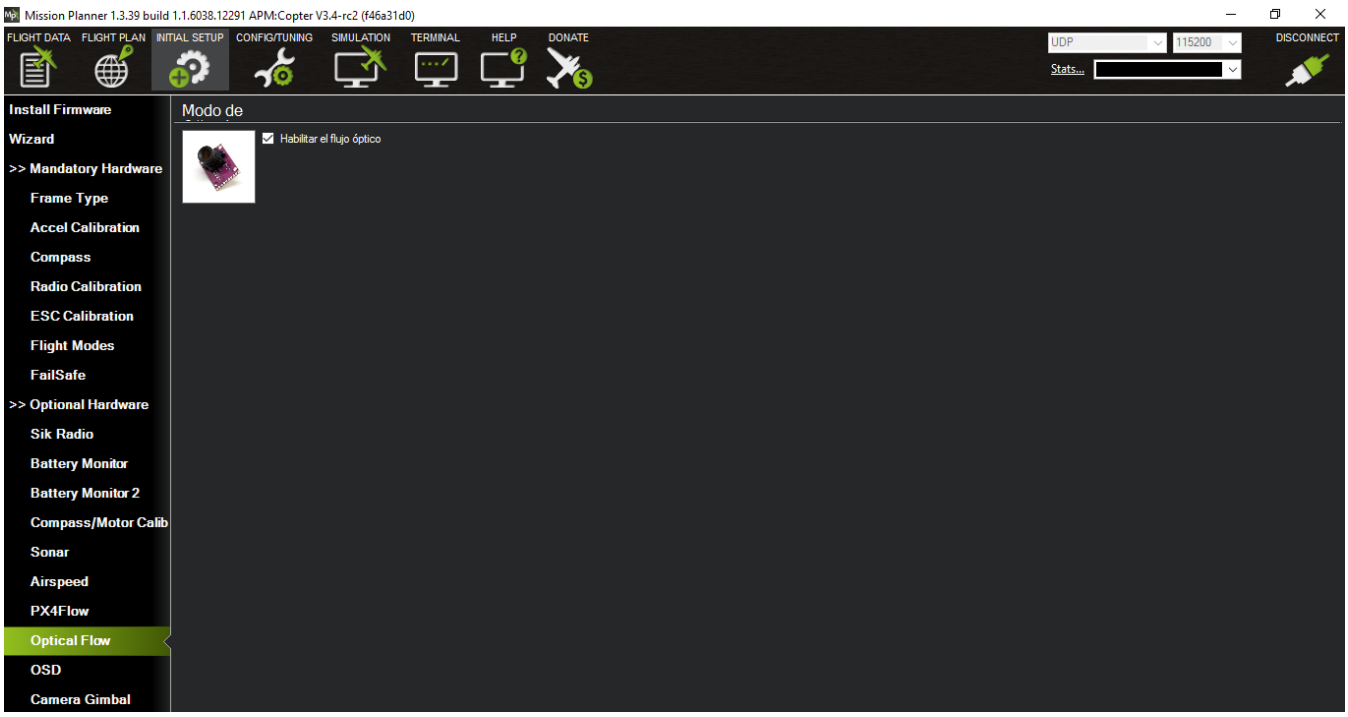


Ilustración 6-6. *Habilitación del sensor PX4Flow en el programa Mission Planner*

6.6 Comprobación del correcto funcionamiento del sensor

Tras reiniciar el controlador podemos comprobar si el sensor se encuentra funcionando abriendo la opción "*Status"* de la pantalla *"Flight Data"*.

Si el sensor está funcionando, los valores *opt_m_x*, *opt_m_y* y *opt_qua* deben ser distintos de cero.

Para nuestro sensor podemos apreciarlo en la siguiente ilustración,

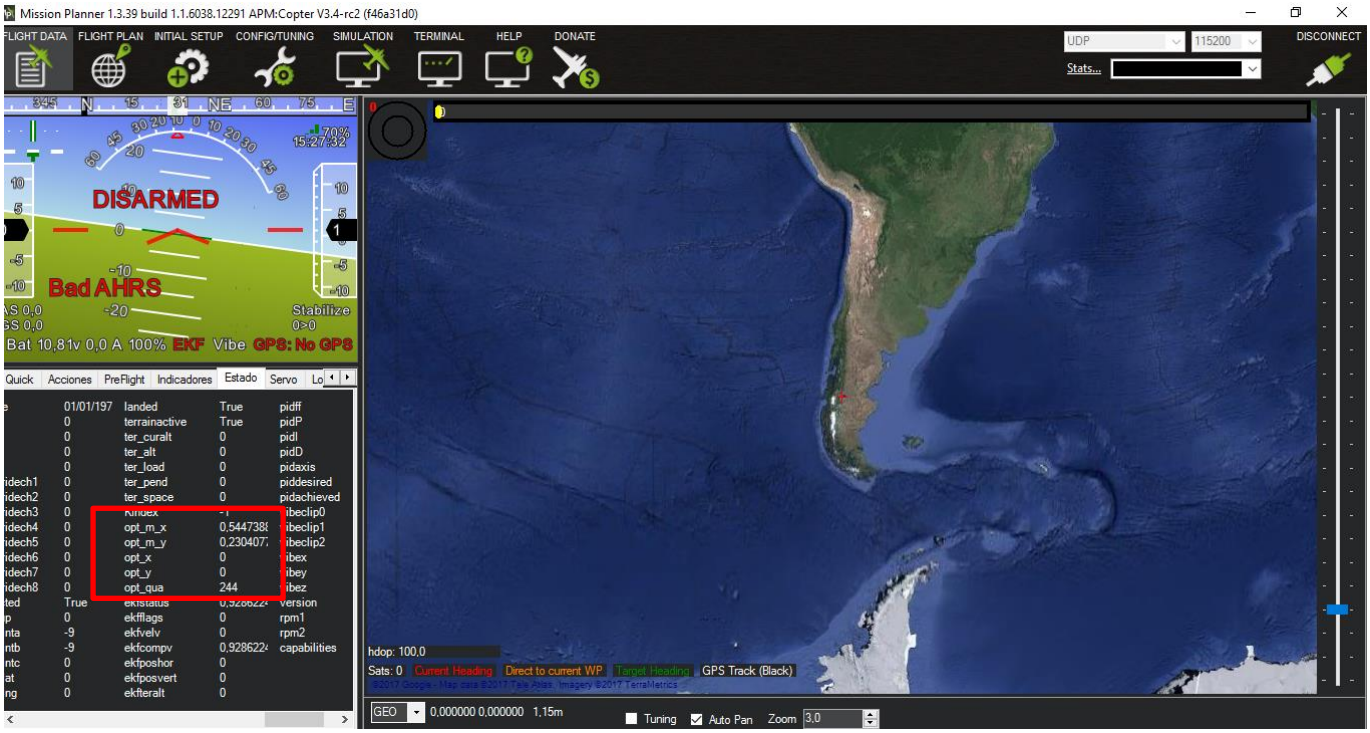


Ilustración 6-7. *Comprobación de la toma de medidas del sensor PX4Flow en Mission Planner*

7 UN PRIMER VUELO

En este séptimo cápitulo, describiremos los pasos previos básicos que se han de establecer para un correcto primer vuelo de *Erle-Copter*.

Es totalmente recomentable que si se carece de experiencia en el pilotaje de vehículos aéreos tipo quadrotor, este primer vuelo se realice en ausencia de las patas, ya que éstas modifican la dinámica del sistema haciendo más complicado su control.

Antes de poder echar a volar el *Erle-Copter,* es necesario realizar una serie de comprobaciones y configuraciones para evitar posibles fallos o accidentes durante el periodo en el que el drone esté en el aire.

Los pasos siguientes se realizan utilizando el mando radio control o a través del programa *Mission Planner*.

- **Arme y desarme del drone**

Para que los motores comiencen a girar y poder elevar el vehículo, es necesario armar el drone. Para iniciar un vuelo, al encenderlo, el stick izquierdo de la emisora correspondiente al acelerador, debe encontrarse en su posición inferior. Después, es necesario esperar a que *Erle-Brain 2* lanze un relativamente largo pitido y el Led de color naranja comience a parpadear. Será entonces cuando *Erle-Copter* está preparado para armarse.

Para armarlo simplemente colocamos el stick izquierdo en la posición inferior derecha hasta que el Led naranja permanezca fijo.

Ilustración 7-1. *Posición del stick izquierdo para armar el drone*

Si lo que queremos es desarmarlo, colocamos el stick izquierdo en la posición inferior izquierda. El Led naranja parpadeará de nuevo.

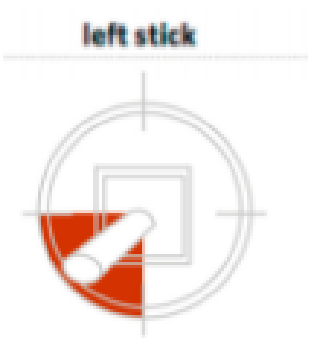


Ilustración 7-2. *Posición del stick izquierdo para desarmar el drone*

- **Calibración de los variadores *ESCs***

Se trata de un paso muy importante y obligatorio de realizar para que el drone vuele correctamente. A continuación se muestran los pasos a seguir para realizar correctamente la calibración.

1. Desconectar la batería y encender la emisora colocando el stick izquierdo en su posición superior al máximo.
2. Conectar la batería. Los Leds del *Erle-Brain 2* azul y naranja se iluminarán siguiendo un patrón cíclico. Esto significa que el drone está listo para entrar en el modo calibración la próxima vez que se conecte.
3. Con el stick izquierdo elevado, desconectar y conectar la batería.
4. *Erle-Brain 2* se encuentra ahora en modo calibración. Los Leds naranja y azul parpadean. Una vez se escuche un largo pitido, el acelerador máximo ha sido capturado y tendremos que colocar ahora el stick izquierdo en su posición mínima, tras lo cual se oirá de nuevo un pitido y la calibración se habrá completado.
5. Desconectar la batería para salir del modo calibración.

- **Comprobar que la dirección de giro de los motores es la adecuada**

Si queremos que el drone despegue, los cuatro motores deben girar en la dirección que indica la Ilustración 3-3, sería muy bueno volver a recordarla.

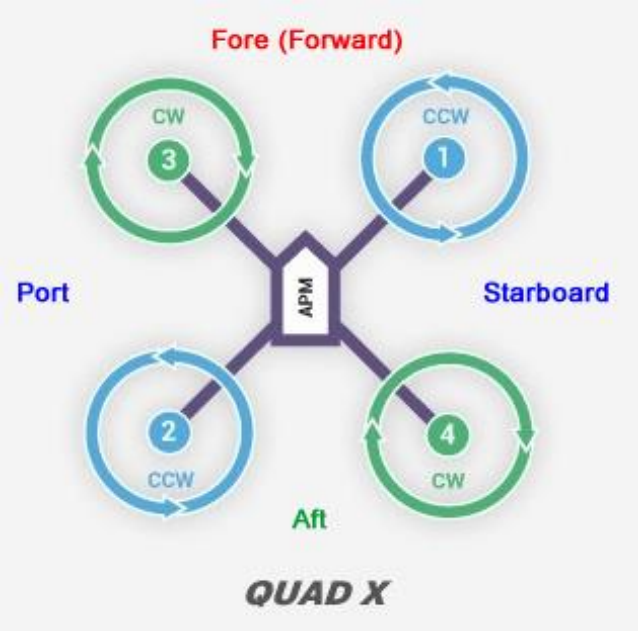


Ilustración 7-3. *Giro adecuado de los motores en un quadrotor*

Este paso se realiza de manera visual y aunque el giro se aprecia mejor con las hélices colocadas, se aconseja retirarlas para evitar que el drone pueda hacernos daño en caso de algún tipo de fallo. En su defecto, puede colocarse un trozo de cinta de un color visible para facilitar dicha comprobación.

- **Comprobación de la conexión de los *ESCs* con el controlador**

Es muy importante comprobar que los distintos variadores se encuentran conectados al controlador *Erle-Brain 2* en el mismo orden que se indica en la Ilustración 3-4. Ya que un fallo en las conexiones con los pines *PWM* harán que el controlador envíe una señal de control al motor que equivocado, provocando fallos en el pilotaje del vehículo que puede acarrear acciones tales como que éste vaya en dirección opuesta o incluso vuelque.

- **Otras comprobaciones**

Utilizando el programa *Mission Planner* podemos comprobar si otros componentes del drone como el acelerómetro o la brújula se encuentra bien calibrados, de no ser así, el programa nos permite hacerlo siguiendo unos intuitivos pasos presentes tanto en la página web de *Erle Robotics* como en el propio *software*.

Una calibración previa recomendable es la de la emisora radio control. Pudiendo hacerse desde *Mission Planner* en la pestaña *"Initial Setup"*, consiste en captar las posiciones máximas y mínimas de los sticks e interruptores presentes en el mando.

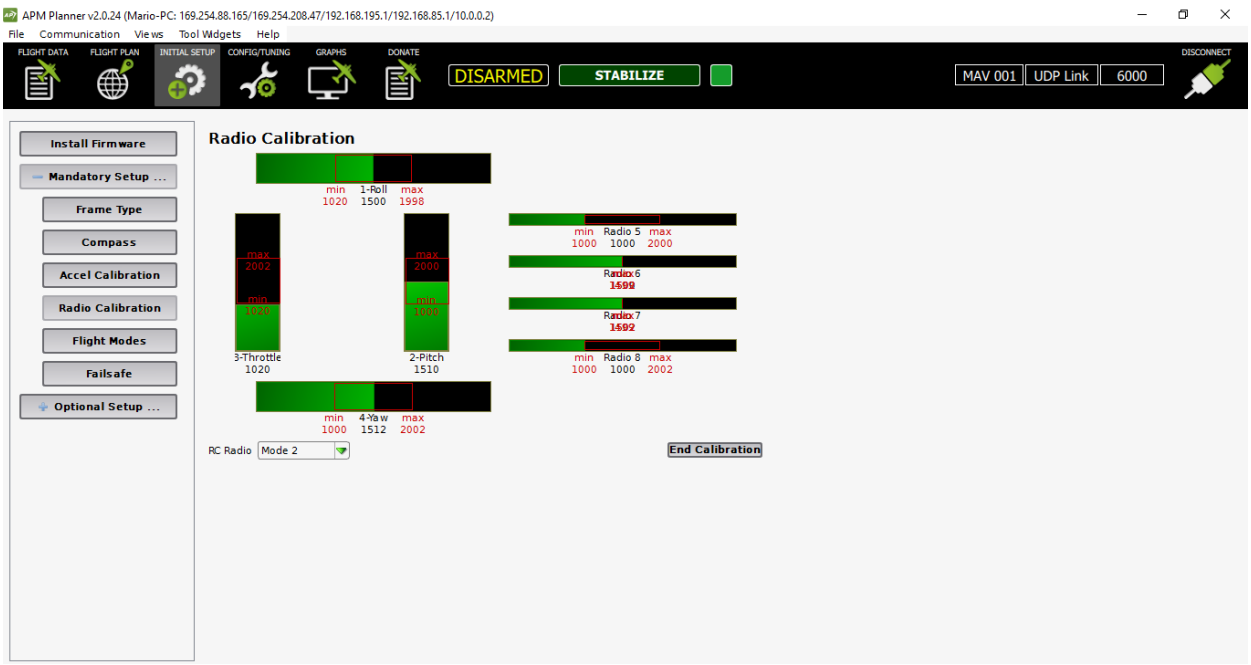


Ilustración 7-4. *Pantalla de calibración de la emisora en APM Planner*

El primer vuelo se realizó en un entorno exterior utilizando sólo las medidas proporcionadas por el *GPS*.

El parámetro interno denominado EK2_GPS_TYPE nos indica qué medidas de los distintos sensores puede utilizar el filtro de Kalman integrado en *Erle-Brain 2*. Si colocamos el valor cero en dicho parámetro, las medidas utilizadas para el control serán únicamente las proporcionadas por el *GPS*. Esto será explicado con mayor detalle en el capítulo siguiente.

Una vez realizado el primer vuelo, me encontré con que el vehículo actuaba de manera "nerviosa", es decir, reaccionaba de manera rápida y brusca a los cambios de inclinación en los ángulos *pitch* y *roll*. Una manera muy sencilla de ajustar esto y adaptar el comportamiento del drone a nuestro gusto, es a través del programa *Mission Planner* o cualquier otra estación de tierra, en su apartado *"Basic Tuning"*. Tal como nos muestra la siguiente imagen,

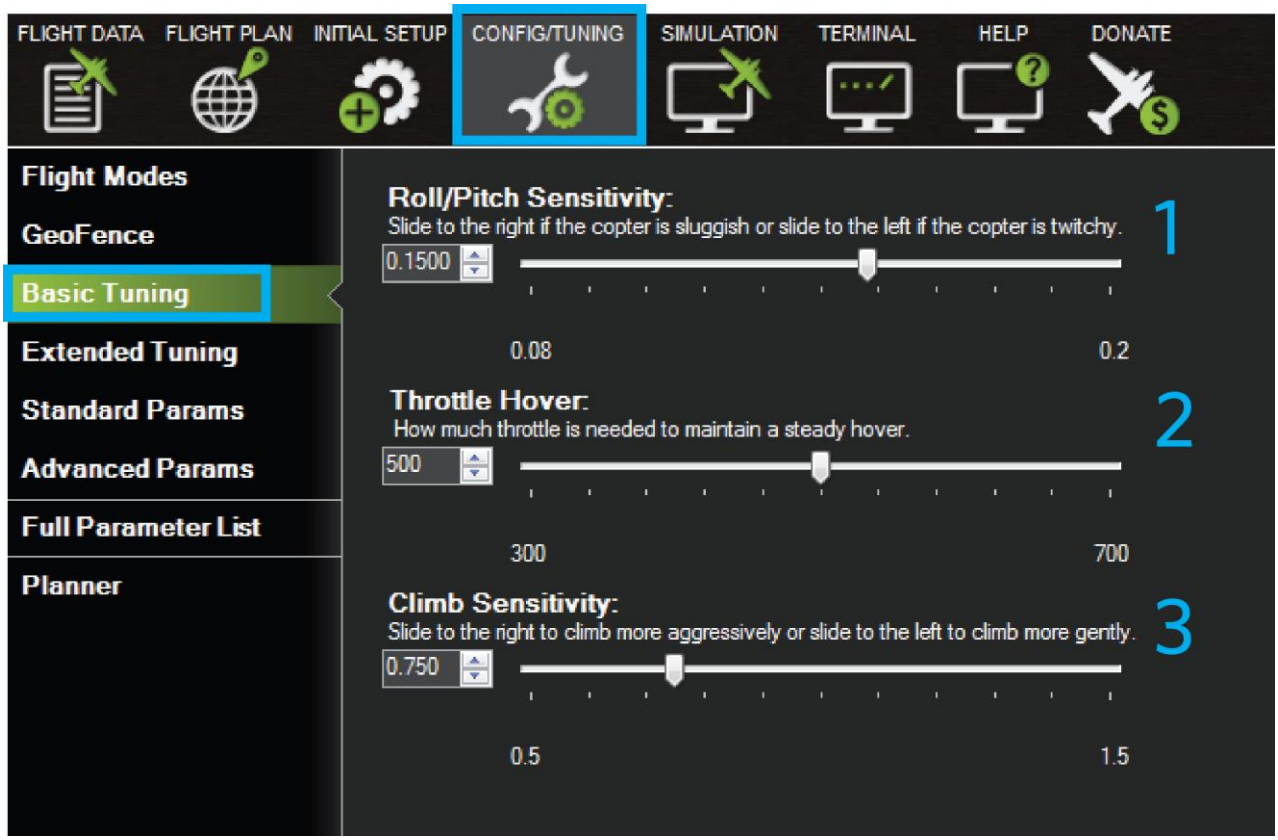


Ilustración 7-5. *Ajustes básicos a través de Mission Planner*

- ***Roll/Pitch* Sensitivity**: si se aprecia un comportamiento del vehículo demasiado "nervioso" en respuesta a los controles del *pitch* y *roll*, debemos deslizar la primera opción hacia la izquierda. Si por el contrario lo observamos lento, la deslizamos hacia la derecha.

Del mismo modo, los dos sliders siguientes sirven para ajustar el acelerador o *throttle*.

- **Throttle Hover**: el quadrotor debe comenzar a ascender cuando el stick del acelerador se encuentre en su posición media, si lo hace con el stick por encima de esa posición, ajustar el slider hacia la izquierda. Si por el contrarío, comienza a ascender con el stick por debajo de su posición media, ajustar el slider hacia la derecha.
- **Climb Sensitivity**: si queremos que el drone gane altitud de manera más agresiva, ajustar el slider hacia la derecha. Si lo que se pretende es que ascienda más suavemente, ajustar el slider hacia la izquierda.

8 CONTROLADORES EN *ARDUCOPTER*

En este octavo capítulo del proyecto, se deja apartado en cierto modo el drone físico y se dan a conocer y comprender en profundidad los distintos controladores que utiliza el autopiloto *ArduCopter* para estabilizar un vehículo quadrotor. De forma paralera, se explicará de manera concisa cómo se ve modificado el vuelo del quadrotor cuando se varía el valor de los parámetros de dichos controladores. Para finalizar, se incluye una introducción al filtro de Kalman extendido aplicado a estos vehículos. La mayor parte de esta información se encuentra publicada de forma más detallada en la página web oficial de *ArduPilot*.

Antes de comenzar, he creído conveniente incluir cierta información básica sobre el funcionamiento de un controlador PID siguiendo el siguiente esquema,

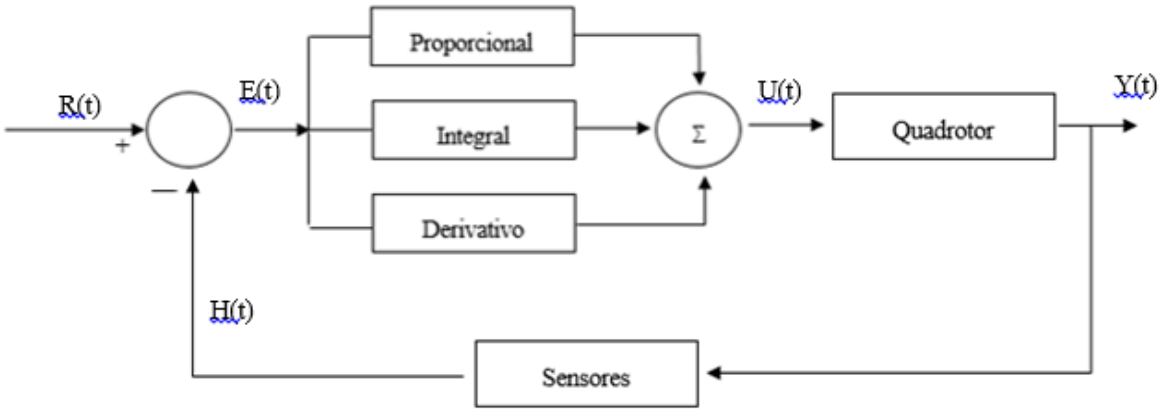


Ilustración 8-1. *Bucle de control PID*

En general, un controlador funciona de la siguiente manera:

Por un lado, tenemos la referencia (R(t)) o el valor deseado para una cierta magnitud. En un quadrotor, dicha magnitud puede ser, por ejemplo, el ángulo de cabeceo o *pitch* del mismo y su valor de referencia puede venir introducido por el piloto a través del mando R/C.

Por otro lado, tenemos el valor real (H(t)) de esa magnitud a controlar, es decir, el valor medido por los sensores de un sistema. En el ejemplo anterior, dicho valor vendría proporcionado por el giroscopio.

Restando estos dos valores, el controlador calcula el error, es decir, la diferencia entre la referencia y el valor real. El objetivo del controlador será conseguir que ese valor sea cero, de modo que la magnitud que se esté controlando se encuentre en el valor de referencia.

Para hacer esto posible, el controlador debe generar una salida de actuación acorde con ese error para tratar de reducirlo.

$$Error = Referencia - Valor\ medido$$

La información captada por los sensores vuelve a introducirse en el controlador mediante el proceso de realimentación. Este proceso también es utilizado para rechazar las posibles perturbaciones que puedan influir sobre el sistema.

8.1 Controladores presentes en *ArduCopter*

Siguiendo con el ejemplo anterior, para modificar el ángulo *pitch*, el controlador debe proporcionar una salida adecuada a través de los actuadores. Estos serán utilizados para modificar hasta seis magnitudas diferentes para lograr controlar la posición y orientación del quadrotor. La ubicación de éste en los ejes x, y, z y su orientación según los ángulos *roll*, *pitch* y *yaw*.

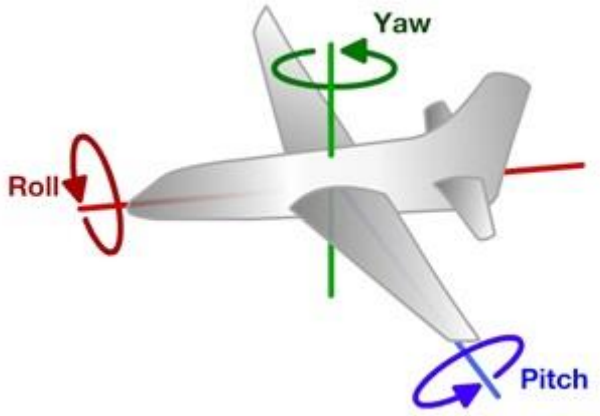


Ilustración 8-2. *Sentido de giro de los ángulos roll, pitch y yaw*

ArduCopter dispone de cuatro actuadores, los cuatro motores con sus respectivas hélices. En ellos, únicamente se varía la velocidad de giro y con ello el empuje. Si cambia la velocidad de un solo motor, se modifican las seis magnitudes.

Los controladores utilizados por *ArduCopter* son filtros PID, un algoritmo de control que utiliza tres parámetros o ganancias para estabilizar el vehículo de manera continua, sean:

- **Proporcional**: depende del error actual. Realiza una corrección proporcional al cambio en ese mismo momento, pero no tiene en cuenta la masa ni la inercia del vehículo.
- **Integral**: depende de los errores pasados y es proporcional a la integral del error. Hay registros de los movimientos del quadrotor. Utiliza esos registros para saber dónde tiene que estar. El término integral posee una estabilización muy lenta y por tanto se utiliza siempre unido a un proporcional.
- **Derivativo**: realiza una predicción de los errores futuros. Es proporcional a la derivada del error.

Los parámetros del controlador son la ganancia proporcional *K*, el tiempo integral T_i y el tiempo derivativo T_d.

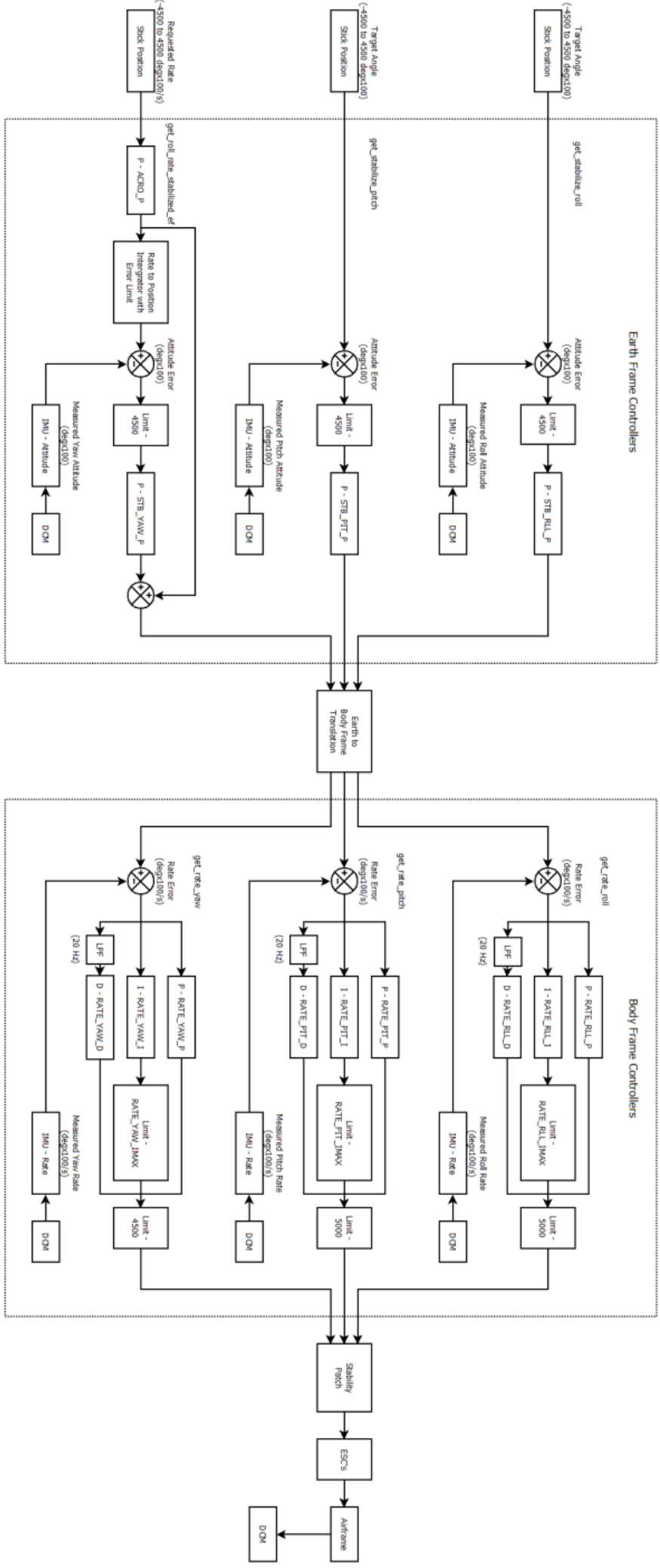
Si se establecen valores del parámetro proporcional demasiado elevados, el quadrotor comenzará a oscilar.

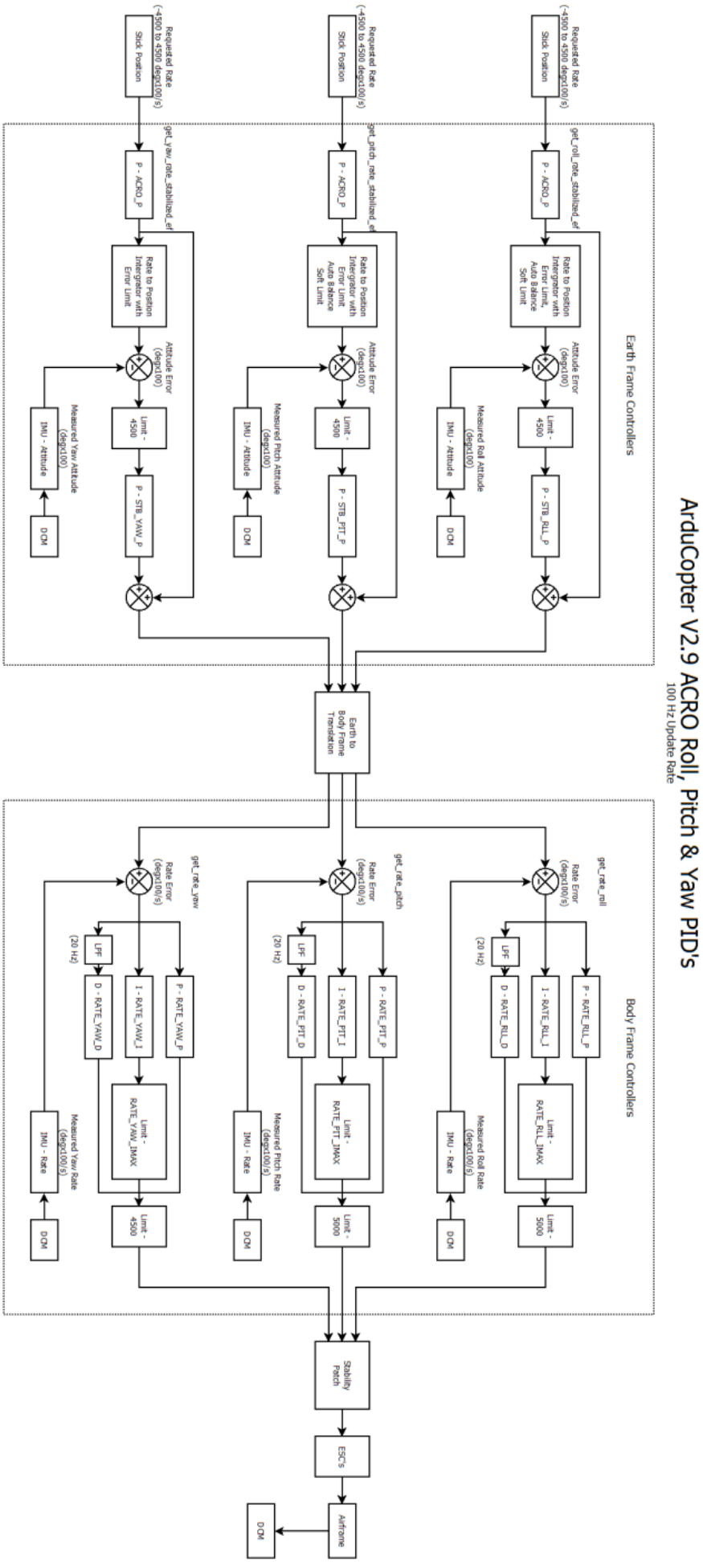
Bajo unas proporciones correctas de estos parámetros, el vehículo volará correctamente estabilizado.

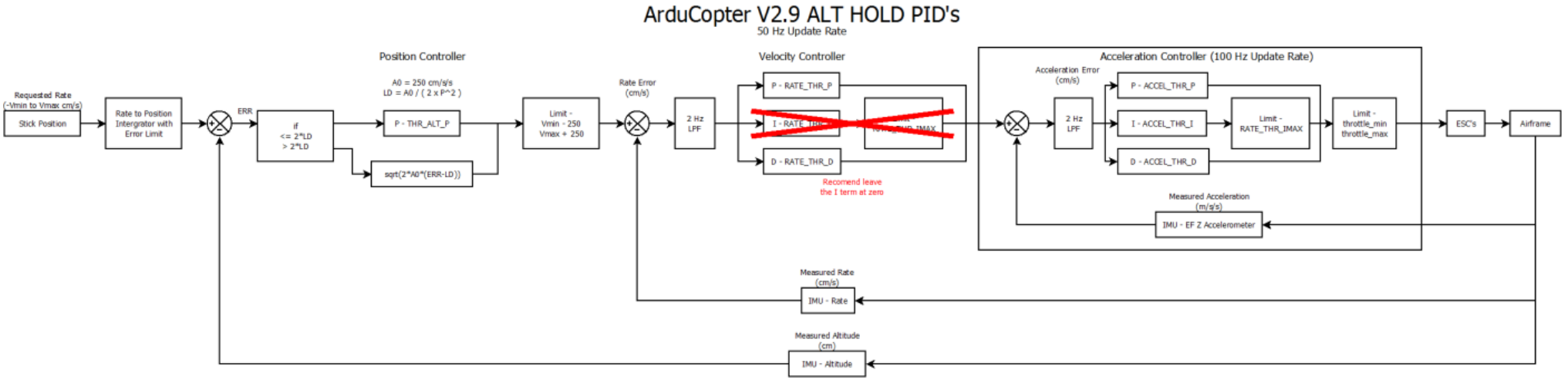
- **Controladores *ArduCopter***

ArduCopter utiliza controladores en cascada. Se define como la configuración donde la salida de un controlador de realimentación es el punto de ajuste para otro, es decir, un controlador se encuentra anidado dentro de otro.

Los esquemas referidos al control se encuentran publicados en el foro de *DIYDrones*. A continuación se expone un duplicado de los mismos.


ArduCopter V2.9 STABILIZE Roll, Pitch & Yaw PID's
100 Hz Update Rate
Earth Frame Controllers
Body Frame Controllers
Target Angle (-4500 to 4500 deg*100)
Stick Position
get_stabilize_roll
get_stabilize_pitch
Requested Rate (-4500 to 4500 deg*100/s)
get_roll_rate_stabilized_ef
P - ACRO_P
Rate to Position Intergrator with Error Limit
Attitude Error (deg*100)
Limit - 4500
P - STB_RLL_P
P - STB_PIT_P
P - STB_YAW_P
Measured Roll Attitude (deg*100)
Measured Pitch Attitude (deg*100)
Measured Yaw Attitude (deg*100)
IMU - Attitude
DCM
Earth to Body Frame Translation
get_rate_roll
get_rate_pitch
get_rate_yaw
Rate Error (deg*100/s)
LPF (20 Hz)
P - RATE_RLL_P
I - RATE_RLL_I
D - RATE_RLL_D
P - RATE_PIT_P
I - RATE_PIT_I
D - RATE_PIT_D
P - RATE_YAW_P
I - RATE_YAW_I
D - RATE_YAW_D
Limit - RATE_RLL_IMAX
Limit - RATE_PIT_IMAX
Limit - RATE_YAW_IMAX
Limit - 5000
Measured Roll Rate (deg*100/s)
Measured Pitch Rate (deg*100/s)
Measured Yaw Rate (deg*100/s)
IMU - Rate
Stability Patch
ESC's
Airframe
DCM


ArduCopter V2.9 ACRO Roll, Pitch & Yaw PID's
100 Hz Update Rate
Earth Frame Controllers
Body Frame Controllers
Requested Rate (-4500 to 4500 degx100/s)
Stick Position
get_roll_rate_stabilized_ef
get_pitch_rate_stabilized_ef
get_yaw_rate_stabilized_ef
P - ACRO_P
Rate to Position Integrator with Error Limit, Auto Balance Soft Limit
Attitude Error (degx100)
Limit - 4500
P - STB_RLL_P
P - STB_PIT_P
P - STB_YAW_P
Measured Roll Attitude (degx100)
Measured Pitch Attitude (degx100)
Measured Yaw Attitude (degx100)
IMU - Attitude
DCM
Earth to Body Frame Translation
get_rate_roll
get_rate_pitch
get_rate_yaw
Rate Error (degx100/s)
LPF (20 Hz)
D - RATE_RLL_D
I - RATE_RLL_I
P - RATE_RLL_P
D - RATE_PIT_D
I - RATE_PIT_I
P - RATE_PIT_P
D - RATE_YAW_D
I - RATE_YAW_I
P - RATE_YAW_P
Limit - RATE_RLL_IMAX
Limit - RATE_PIT_IMAX
Limit - RATE_YAW_IMAX
Limit - 5000
Measured Roll Rate (degx100/s)
Measured Pitch Rate (degx100/s)
Measured Yaw Rate (degx100/s)
IMU - Rate
Stability Patch
ESC's
Airframe
DCM


ArduCopter V2.9 ALT HOLD PID's
50 Hz Update Rate
Position Controller
Velocity Controller
Acceleration Controller (100 Hz Update Rate)
Requested Rate
(-Vmin to Vmax cm/s)
Stick Position
Rate to Position Intergrator with Error Limit
ERR
if
<= 2*LD
> 2*LD
A0 = 250 cm/s/s
LD = A0 / (2 x P^2)
P - THR_ALT_P
sqrt(2*A0*(ERR-LD))
Limit -
Vmin - 250
Vmax + 250
Rate Error
(cm/s)
2 Hz
LPF
P - RATE_THR_P
I - RATE_THR_I
D - RATE_THR_D
Recomend leave
the I term at zero
Acceleration Error
(cm/s)
2 Hz
LPF
P - ACCEL_THR_P
I - ACCEL_THR_I
D - ACCEL_THR_D
Limit -
RATE_THR_IMAX
Limit -
throttle_min
throttle_max
ESC's
Airframe
Measured Acceleration
(m/s/s)
IMU - EF Z Accelerometer
Measured Rate
(cm/s)
IMU - Rate
Measured Altitude
(cm)
IMU - Altitude

A partir del programa *Mission Planner* o cualquier otro utilizado como estación de tierra, es posible ajustar los parámetros de los distintos controladores que posee *ArduCopter*. A continuación se expone una imagen de dicho programa con cada uno de los controladores. Después se explicarán de forma breve,

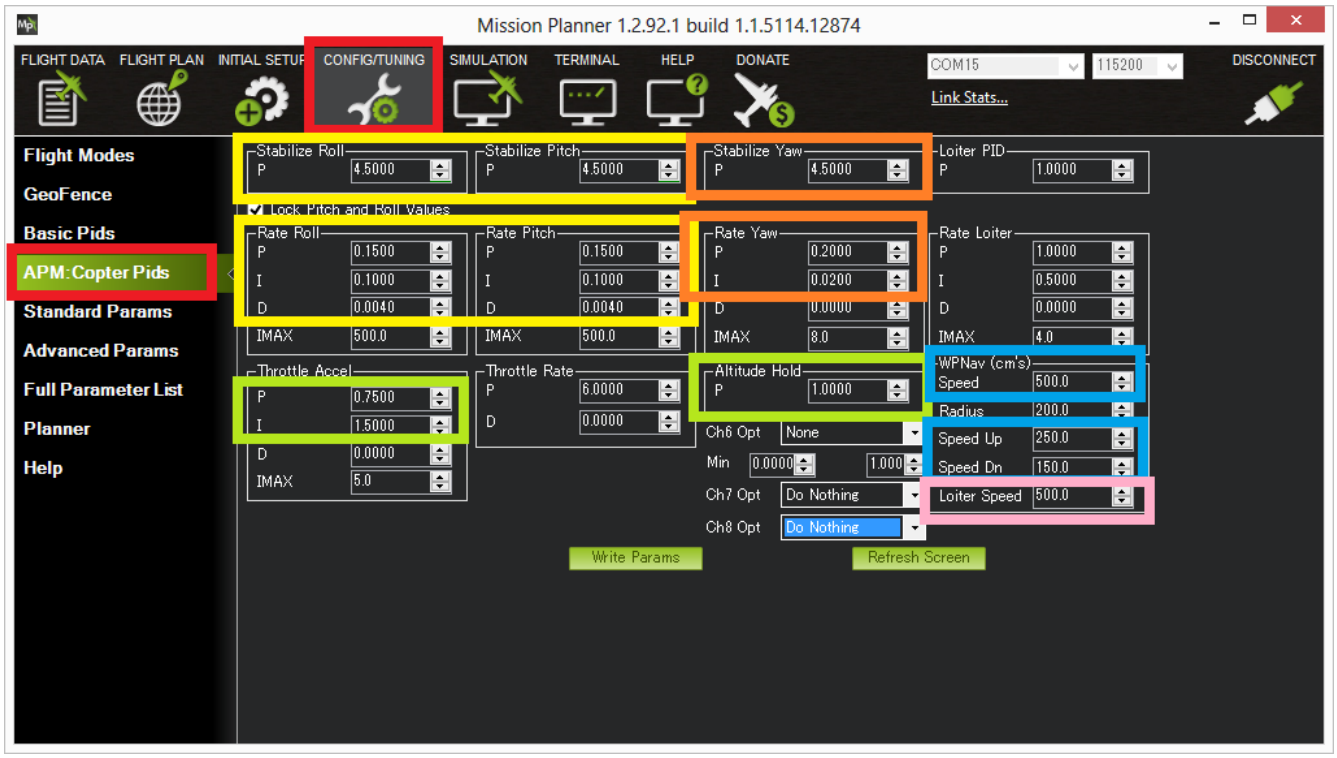


Ilustración 8-3. *Parámetros de los controladores PID presentes en ArduCopter*

Los parámetros que *ArduCopter* incluye de forma predeterminada corresponden al vehículo quadrotor *3DR IRIS*. Estos mismos parámetros son los utilizados por nuestro drone físico *Erle-Copter*, la única diferencia es la configuración del frame. *3DR IRIS* posee una configuración en H mientras que *Erle-Copter* presenta una en forma de X.

Ilustración 8-4. *Drone 3DR IRIS*

Qué controlador está activo en cada momento depende del modo de vuelo. Los cuatro controladores de velocidad, *Rate* (*Roll*, *Pitch*, *Yaw* y *Loiter*) son los controladores internos y más importantes, éstos están activos en todos los modos. Si estos controladores están bien ajustados, el vehículo volará relativamente estable.

El resto de controladores, usan de forma directa o indirecta los internos.

A modo de resumen,

- *Rate (Roll, Pitch, Yaw)*: controladores de velocidad angular. A bajo nivel.
- *Stabilize (Roll, Pitch, Yaw)*: controladores de ángulo. Funcionan por encima de los Rate.
- *Rate Loiter*: controla la velocidad angular en modo *Loiter*.
- *Loiter Speed*: controla la velocidad en modo *Loiter*.
- *Throttle (Rate, Acell)*: controladores de aceleración y velocidad vertical para controlar la altura.
- *Altitude Hold*: controlador de altitud para dicho modo.
- *Nav WP*: es el controlador de navegación que define los puntos objetivo de otros controladores en el modo *Auto*.

Cada uno de estos controladores se explicará en profundidad en el siguiente capítulo, de manera individual utilizados en sus respectivos modos de vuelo.

8.2 Filtro de Kalman extendido

ArduCopter utiliza el algoritmo de filtro de Kalman extendido o *EKF* para realizar una estimación de la posición del vehículo, su velocidad y orientación basándose en las medidas captadas por el giroscopio, el acelerómetro, la brújula, el *GPS* y la presión barométrica.

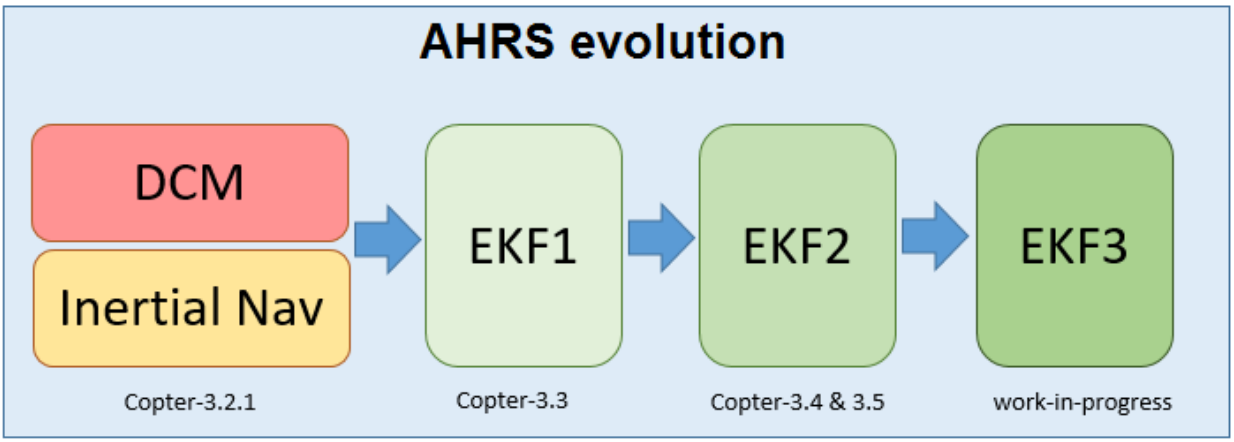


Ilustración 8-5. *Evolución de los distintos filtros usados por ArduCopter*

La principal ventaja que presenta este filtro respecto a otros más simples como, por ejemplo, "*Inertial Nav",* es que mediante la fusión de todas las medidas disponibles, es capaz de rechazar mediciones con un error significativo, de modo que el vehículo es menos susceptible a un determinado fallo que pueda afectar a un único sensor. Del mismo modo, a *EKF* se le puede incluir las medidas de otros sensores externos tales como el lídar o el sensor de flujo óptico.

Otra característica del filtro *EKF,* es que es capaz de estimar desplazamientos en las lecturas de las brújulas de los vehículos, esto hace que sea menos sensible a errores de calibración de la misma.

8.2.1 Principio de funcionamiento

El filtro de Kalman extendido, estima un total de 22 estados. Un estado, es cualquier variable que se desee estimar, como la posición, la velocidad o los ángulos.

A continuación se enumeran los pasos de funcionamiento del filtro de forma resumida,

1. Las velocidades angulares proporcionadas por la *IMU* se integran para calcular la posición angular.
2. Las aceleraciones proporcionadas por la *IMU* se convierten utilizando la posición angular del cuerpo.
3. Estas aceleraciones se integran para calcular la velocidad.
4. Dicha velocidad se integra para calcular la posición.

Los puntos anteriores se refieren al "estado de predicción".

5. Se estima el ruido presente en el giroscopio y el acelerómetro. Los parámetros EKF_GYRO_NOISE y EKF_ACC_NOISE se utilizan para llevar a cabo dicha estimación, sin ella el error continuaría creciendo. Si se aumenta el valor de esos parámetros, la estimación del error en los filtros será más rápida.

   Estos errores estimados se almacenan en una matriz denominada "Matriz de covarianza de estados".

Los pasos del 1 al 5 se repiten siempre que se obtengan nuevos valores de la *IMU* y hasta que esté disponible

una nueva medida en otro sensor.

6. Cuando se recibe una medida del *GPS,* el filtro calcula la diferencia entre la posición predicha en el punto 4 y la posición real.
7. La diferencia obtenida en el punto 6, junto con la "Matriz de covarianza de estados" del punto 5 y el error de medición del *GPS* especificado por el parámetro EKF_POSNE_NOISE, se combinan para estimar la corrección para cada uno de los estados del filtro.

   Hacer que el valor del parámetro EKF_POSNE_NOISE aumente, conlleva que el filtro piense que las medidas proporcionadas por el *GPS* son menos precisas.
8. Al haber tomado una medida, la incertidumbre en cada uno de los estados actualizados se reduce. El filtro calcula la reducción de esa incertidumbre, actualiza la "Matriz de covarianza de estados" y vuelve al paso 1.

➢ La versión actual y estable del filtro de Kalman utilizada por *ArduPilot* es *EKF2* junto con "*DCM*" funcionando en segundo plano. Esto es, si un controlador de vuelo posee dos (o más) *IMUs* operativas, dos "nucleos"/instancias de *EKF*, éstas se ejecutarán en paralelo y cada una de ellas utilizará las medidas de una *IMU* diferente.

8.2.2 Elección de *EKF* y el número de nucleos

A partir de los siguientes parámetros, puede elegirse el tipo de filtro a utilizar por el autopiloto y el número de nucleos en paralelo que usaría *EKF*.

EKF2 por tanto, es capaz de utilizar hasta dos nucleos con dos *IMUs* diferentes. Mientras que *EKF3* es capaz de utilizar hasta 3.

- AHRS_EKF_USE: se ajusta a "1" para usar *EKF* y a "0" utiliza "*DCM"* para el control de actitud y navegación del vehículo (versión 3.2.1 de *ArduCopter*).
- AHRS_EKF_TYPE: se ajusta a "2" para usar *EKF2* para la estimación de la actitud y posición del quadrotor y a "3" para utilizar *EKF3*.
- EK2_ENABLE, EK3_ENABLE: ajustándolos a "1" habilita *EK2* o *EK3* respectivamente.
- EK2_IMU_MASK, EK3_IMU_MASK: se trata de una máscara de bits que especifica cuál de las *IMUs* disponibles se va a utilizar. Se iniciará el nucleo de *EKF* correspondiente según la *IMU* elegida. Los siguientes valores se refieren a *EKF2*, dos nucleos junto con dos *IMUs*.
  1. Inicia un solo nucleo *EKF* utilizando la primera *IMU*.
  2. Inicia un solo nucleo *EKF* utilizando la segunda *IMU*.
  3. Inicia dos nucleos *EKF* en paralelo utilizando la primera y la segunda *IMU*.

8.2.3 Parámetros de ajuste más comunes

Algunos de los parámetros más utilizados si se quiere modificar el filtro de Kalman según las necesidades del piloto son los siguientes,

- EK2_ALT_SOURCE: establece qué sensor se utiliza como principal fuente de información de la altitud.
  0. Utiliza el barómetro (valor por defecto).

1. Se utiliza para entornos en los que las medidas del barómetro incluyen mucho ruido.
2. Utiliza las medidas del *GPS*.

- EK2_ALT_NOISE: por defecto con valor "1". Valores más bajos hacen que se reduzca la dependencia del acelerómetro y se incremente la del barómetro.
- EK2_GPS_TYPE: controla el uso que se realiza con las medidas tomadas por el *GPS*. Puede tomar los siguientes valores,

0. Utiliza la velocidad en 3D y la posición en 2D del *GPS*.
1. Utiliza la velocidad 2D y la posición 2D del *GPS*.
2. Utiliza solo la posición 2D del *GPS*.
3. No utiliza el *GPS*. Se utilizarán las medidas del sensor de flujo óptico si este estuviera operativo.

9 MODOS DE VUELO EN ARDUCOPTER

Una vez que se conocen los distintos controladores que *ArduCopter* utiliza, en este noveno capítulo se explican los modos de vuelo más importantes que puede ofrecer a nuestro vehículo.

ArduCopter cuenta con un total de 14 modos de vuelo diferentes, sin embargo los más usados son los siguientes:

- *Stabilize*
- *Alt Hold*
- *Loiter*
- *RTL (Return to launch)*
- *Auto*

A continuación se explicarán de manera general estos cinco modos y sus respectivos controladores. Una vez conocidos los modos de vuelo básicos, se hará referencia a dos de ellos algo más avanzados, el modo *Acro* y el *Sport.*

9.1 Modo *Stabilize*

Al activar este modo, el piloto tiene el control total del vehículo. Éste controla los motores y el movimiento del drone. El trabajo que realiza la controladora es estabilizarlo, ajustando automáticamente los valores de los ángulos *pitch* y *roll*.

Se trata de un modo que no depende del *GPS*. El drone, puede encontrarse en este modo utilizando unicamente las medidas proporcionadas por la *IMU*.

El piloto debe de estar continuamente regulando la entrada de aceleración para mantener el vehículo a la altura deseada, así como la de los ángulos *pitch* y *roll* para mantenerlo en una posición determinada.

9.1.1 Ajuste del controlador

Este modo se controla con los parámetros PID que aparecen de color amarillo en la Ilustración 8-3. A continuación se explican en detalle,

- *Stabilize Roll/Pitch:* estos parámetros controlan la capacidad de respuesta del quadcopter para dichos ángulos en función de la entrada proporcionada y el error entre los ángulos *pitch* y *roll* deseados y los reales.

  Un valor demasiado elevado hará que el drone oscile bruscamente durante esos giros. De manera opuesta, un valor demasiado bajo hará que el drone se torne lento.

- *Rate Roll/Pitch:* controlan la salida de los motores basándose en la velocidad de rotación obtenida desde el controlador superior *Stabilize*.

  Para vehículos más potentes, estos términos han de ser de menor valor y viceversa.

  - Término P: es necesario que esté bien ajustado. Posee un valor por defecto de P = 0.15 y su aumento provoca que la respuesta en los motores se eleve para alcanzar la velocidad de giro deseada lo antes posible.

- Término I: se utiliza para compensar las perturbaciones que provocan que el drone no se mantenga en esa velocidad deseada.
- Término D: amortigua la respuesta en aceleración del quadrotor.

  Su valor por defecto es D = 0.11 y su aumento puede causar vibraciones indeseadas.

Existen numerósos parámetros que modifican la forma de volar en este modo, explicados en profundidad en la página oficial de *ArduCopter*.

El lazo de control utilizado para este modo viene representado en el capítulo anterior.

9.2 Modo *Alt Hold*

En el modo *Alt Hold*, el controlador mantendrá la altura relativa del vehículo respecto a un punto de referencia utilizando las variaciones en la presión barométrica.

Este modo, por tanto, tampoco depende de las medidas del *GPS*.

Cuando el piloto mantiene el stick de aceleración entre el 40 y 60% de su valor, el controlador mantendrá la altura del vehículo, si no, éste subirá o bajará a una velocidad predefinida de 2.5 m/s. Este valor puede cambiar modificando el parámetro PILOT_VELZ_MAX.

9.2.1 Ajuste del controlador

Este modo se controla con los parámetros PID que aparecen de color verde en la Ilustración 8-3. A continuación se explican en detalle,

- *Altitude Hold:* estos parámetros convierten el error de altitud (diferencia entre la altitud deseada y la real) en velocidad de ascenso o descenso.

  Cuanto mayor sea el valor de estos, más agresiva será la respuesta del vehículo al tratar de mantener la altura.
- *Throttle Rate:* convierten la velocidad de ascenso o descenso deseada en una aceleración hacia arriba o hacia abajo.
- *Throttle Accel:* convierte el error en aceleración en la salida de los motores.

  En estos parámetros, la relación 1:2 entre los términos P e I debe mantenerse siempre, es decir, el valor del término I debe ser siempre el doble que el de P.

El lazo de control utilizado para este modo viene representado en el capítulo anterior.

9.3 Modo *Loiter*

En modo *Loiter*, el piloto puede controlar el drone de manera manual pero cuando éste suelta los sticks, el controlador mantendrá de manera automática la posición actual del vehículo mediante las medidas del *GPS* y su altitud barométricamente.

Es, por tanto, un modo dependiente del *GPS*.

Se trata del modo más sencillo de utilizar, ya que el controlador se encarga del control de toda la dinámica.

Importante: si la brújula no está bien ajustada o sus medidas se exponen a algún tipo de interferencia electromagnética, el drone comenzará a girar en círculos sobre su posición.

9.3.1 Ajuste del controlador

Este modo se controla con los parámetros PID que aparecen de color rosa en la Ilustración 8-3.

Si los parámetros de control de los ángulos *pitch* y *roll* están bien ajustados, el *GPS* y la brújula están correctamente configurados y las vibraciones son aceptables, este modo no requiere de mucha afinación.

9.4 Modo *RTL*

El modo *RTL* o vuelta a tierra, resulta bastante útil cuando ocurre algún tipo de fallo, cuando se está acabando la batería o simplemente cuando el vehículo se sale del rango de control. Cuando algo de esto ocurre, el drone puede iniciar este proceso habiendo sido programado previamente. Durante el mismo, el drone iniciará de forma autónoma un ascenso hasta una altura programada seguido por un desplazamiento en linea recta hasta la vertical del punto de partida, donde el drone descenderá hasta tocar tierra. El punto de partida será aquel en la que se armó el vehículo.

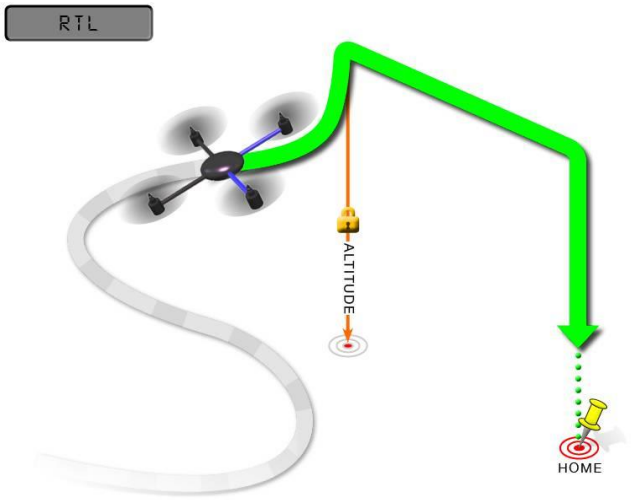


Ilustración 9-1.*Trayectoria seguida por el drone en el modo RTL*

Al igual que el modo anterior, *RTL* es dependiente del *GPS,* por lo que es importante asegurarse de que el vehículo recibe su señal. Esto puede ser comprobado mediante la luz del Led azul del *ArduPilot* que debe ser sólida y fija.

9.4.1 Ajuste del controlador

Las alturas y velocidades del drone durante este recorrido pueden ser modificados cambiando el valor de distintos parámetros, por ejemplo:

- RTL_ALT: este parámetro ajusta la altura que debe tomar el quadrotor antes de iniciar la vuelta a casa. Si es cero, volverá desde la altura en la que se encuentre.

  Su valor por defecto es de 15 m y puede ajustarse desde 1 a 8000 cm.

- RTL_ALT_FINAL: ajusta la altura mínima sobre la vertical una vez que el drone regrese a su posición de armado. Si es cero, volverá directamente a tierra.

  Su valor puede variar desde 1 a 1000 cm.

- LAND_SPEED: tal y como indica su nombre, este parámetro ajusta la velocidad de descenso durante el aterrizaje.

  Esta velocidad es ajustable desde 20 hasta 200 centímetros por segundo.

El resto de parámetros que modifican este modo se encuentran disponibles en la web del controlador.

9.5 Modo *Auto*

Se trata de un modo en el que el vehículo se controla totalmente de forma autónoma. El piloto puede planificar una ruta desde una estación de tierra y mandársela al drone para que la ejecute.

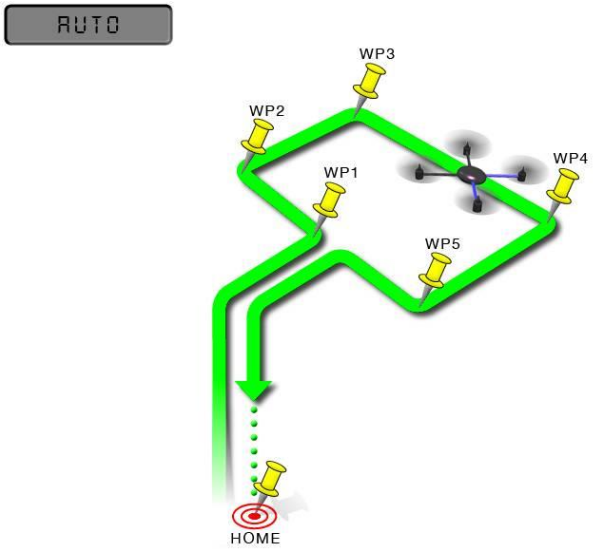


Ilustración 9-2. *Trayectoria seguida por el drone en el modo Auto*

Este modo incorpora el control de altitud del modo *Alt Hold* y el control de posición del modo *Loiter*. Por tanto depende del *GPS* y necesita las mismas condiciones que para dicho modo.

9.5.1 Ajuste del controlador

Este modo se controla con los parámetros PID que aparecen de color azul en la Ilustración 8-3.

Las alturas y velocidades del drone durante este recorrido pueden ser modificados cambiando el valor de distintos parámetros, por ejemplo:

- WPNAV_SPEED: parámetro que ajusta la velocidad máxima horizontal del drone durante el trayecto. Su valor por defecto es 5 m/s, por tanto dicho parámetro se coloca con un valor 500.
- WPNAV_SPEED_UP: ajusta la velocidad de ascenso durante el trayecto.

- WPNAV_SPEED_DN: ajusta la velocidad de descenso durante el trayecto.

9.6 Controlador del ángulo Yaw

El único controlador que queda por explicar es el PID que controla la respuesta del ángulo *yaw*, que aparece en color naranja en la Ilustración 8-3.

Actúa de manera similar al controlador *Stabilize* y normalmente no se necesita un mayor ajuste. Un valor demasiado elevado de estos parámetros hará que el drone oscile y demasiado bajos pueden hacer que éste no pueda mantener su trayectoria.

El parámetro ACRO_YAW_P controla la velocidad de giro del vehículo sobre su eje vertical basándose en la entrada proporcionada por el piloto. Aumentar este parámetro hará que el drone gire más deprisa.

Una vez que conocemos los cincos modos de vuelo más utilizados y los distintos controladores PID que estos utilizan, es el momento de adentrarnos en los modos de vuelo *Acro* y *Sport*. Para cada uno de ellos, se explica a continuación en qué consisten, algunos de los parámetros utilizados para su control y la diferencia entre ambos modos.

9.7 Modo *Acro*

En este modo de vuelo avanzado, los sticks de la emisora radio control son utilizados para controlar la velocidad angular del quadrotor. Si el piloto los suelta, el vehículo mantendrá su posición actual, es decir, no se estabilizará.

Se trata de un modo muy útil cuando se quiere realizar todo tipo de acrobacias de 360º o cuando se desea un control rápido y sueve en *FPV* (*First Person View*).

Posee un acelerador completamente manual, sin compensación del ángulo de inclinación.

Las versiones 3.1 y posteriores de *ArduCopter* poseen una funcionalidad denominada *"AcroTrainer",* que nos facilita el aprendizaje de este modo de vuelo.

➢ **Funcionalidad *"AcroTrainer"***

El parámetro ACRO_TRAINER puede ser modificado con distintos valores:

- 0 = deshabilitado. El piloto controla el quadrotor sin una nivelación automática de éste ni ninguna limitación en su ángulo de inclinación.
- 1 = nivelación automática. El vehículo se nivela automáticamente cuando el piloto suelta los sticks del radio control. La agresividad con la que el drone reacciona a este proceso puede controlarse modificando el valor de los parámetros ACRO_BAL_PITCH y ACRO_BAL_ROLL.
- 2 = nivelación automática y ángulo de inclinación limitado. Por defecto, el vehículo no se inclinará más de 45º. Valor modificable con el parámetro ANGLE_MAX.

9.7.1 Ajuste del controlador

Algunos de los parámetros que nos permiten adaptar este modo de vuelo a nuestras preferencias son:

- ACRO_RP_P: controla la velocidad de rotación en los ejes del *pitch* y el *roll*. El valor por defecto de

este parámetro es de 4.5 y proporciona una velocidad de rotación de 200 º/s.

- ACRO_YAW_P: controla la velocidad de rotación en el eje del *yaw*. Posee un valor por defecto igual al parámetro anterior.
- ACRO_EXPO: se trata de una cantidad exponencial que se aplica a los pilotos que utilizan únicamente el modo acro, por ejemplo, en carreras de drones. Por defecto, este modo es mucho más sensible que los otros, incluso cuando los sticks se encuentran en su posición intermedia. Por tanto, este parámetro permite al piloto, ajustar la respuesta al stick en el control para que se asemeje a la que recibiría en otros modos de vuelo como *Stabilize* o *AltHold*. Reduce la sensibilidad.

9.8 Modo *Sport*

El modo *Sport,* también conocido como "Velocidad controlada a estabilizar" unido a un mantenimiento de la altitud. Está diseñado para volar con visión en primera persona y por lugares complicados, ya que puedes establecer al vehículo un determinado ángulo de inclinación y éste se mantendrá en él.

El vehículo posee una inclinación máxima por defecto de 45º, modificable con el parámetro ANGLE_MAX, al igual que ocurría con el valor 2 de la funcionalidad "*AcroTrainer"* en el modo *Acro*.

La altura del quadrotor se mantiene al igual que lo hace en el modo *Altitude Hold*.

Por lo tanto, la principal diferencia entre ambos modos viene dada en las limitaciones en sus ángulos de inclinación. Mientras que el modo *Sport* es capaz de fijar el vehículo en uno ángulo de inclinación determinado, el modo *Acro* (con el parámetro ACRO_TRAINER a un valor 0, es decir, deshabilitado) proporciona al quadrotor movimientos de 360º en el espacio.

10 REPRESENTACIONES GRÁFICAS DEL VUELO

En en este décimo y último capítulo del proyecto se exponen distintas gráficas obtenidas durante uno de los vuelos realizados al vehículo *Erle-Copter*. En ellas, podremos observar las diferentes señales de referencia indicadas a través del mando radio control, los valores captados por los sensores, etc.

Para la representación de las distintas medidas captadas por los sensores se ha convertido el *log* correspondiente a un archivo compatible con *MatLab*, tal como se explicó en el apartado 5.1.1 de ese mismo capítulo, para una mejor visualización de las gráficas.

10.1 Rendimiento de estabilización del vehículo

Para comprobar los diferentes valores de referencia y de respuesta de los ángulos *pitch*, *roll* y *yaw* debemos descargar y visualizar el *log* correspondiente a dicho vuelo y representar, dentro del apartado *ATT*, información acerca de la actitud del vehículo, las opciones deseadas.

En este caso, se comparan en gráficas diferentes. Por un lado el ángulo *pitch* deseado por el piloto y el ángulo *pitch* real del quadrotor,

- *DesPitch*: ángulo *pitch* de referencia dado por el piloto en grados. Hacia delante, el *pitch* es negativo y hacia atras positivo.
- *Pitch*: ángulo *pitch* tomado por el vehículo. Mismo signo que *DesPitch*.

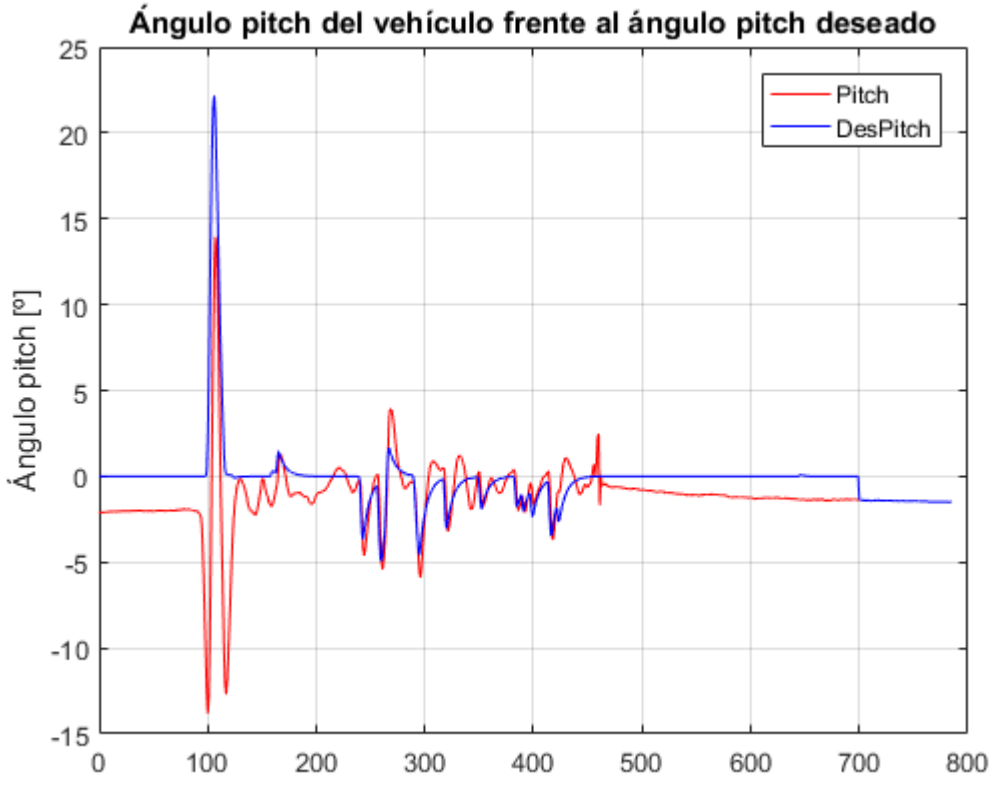


Ilustración 10-1. *Ángulo pitch del quadrotor frente al ángulo pitch deseado*

Y por otro, el ángulo *roll* deseado por el piloto y el ángulo *roll* real del quadrotor,

- *DesRoll*: ángulo *roll* de referencia dado por el piloto en grados. Hacia la izquierda, dicho *roll* es negativo y hacia la derecho positivo.
- *Roll*: ángulo *roll* tomado por el vehículo. Mismo signo que *DesRoll*.

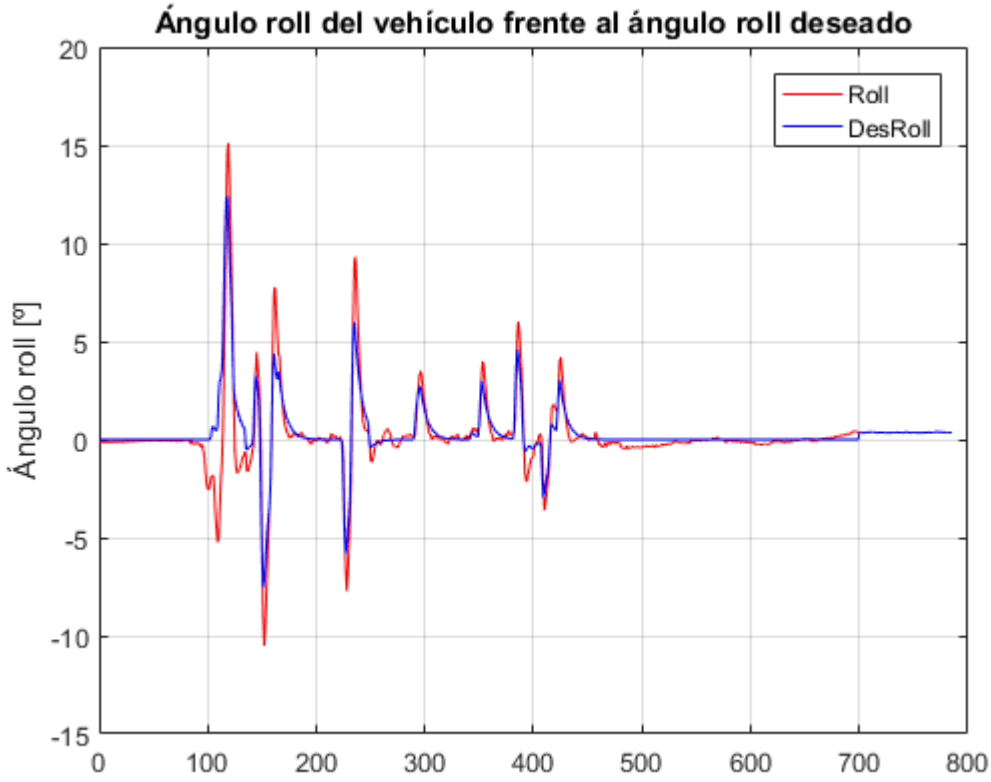


Ilustración 10-2. *Ángulo roll del quadrotor frente al ángulo roll deseado*

Como se puede observar, el valor real de ambos ángulos sigue de forma muy satisfactoria a la referencia proporcionada por el piloto, en este caso yo, mediante el mando radio control.

10.2 Rendimiento del vehículo al mantener la altitud

Al igual que se ha hecho para comprobar la estabilización del vehículo, lo haremos ahora para examinar si *Erle-Copter* ha mantenido la altura deseada al entrar en modo *Alt Hold.*

Ahora, se representan las señales almacenadas en el *log* correspondientes al apartado *CTUM*, es decir, la información de la altitud y la aceleración del vehículo:

- *DAlt:* indica la altitud deseada, a alcanzar por el piloto, en metros, mientras el modo *Alt Hold* se encuentra activo.
- *Alt:* indica la altitud alcanzada por *Erle-Copter* según el *GPS*, en metros, mientras el modo *Alt Hold* se encuentra activo.
- *BAlt:* indica la altitud sobre el suelo según el barómetro.

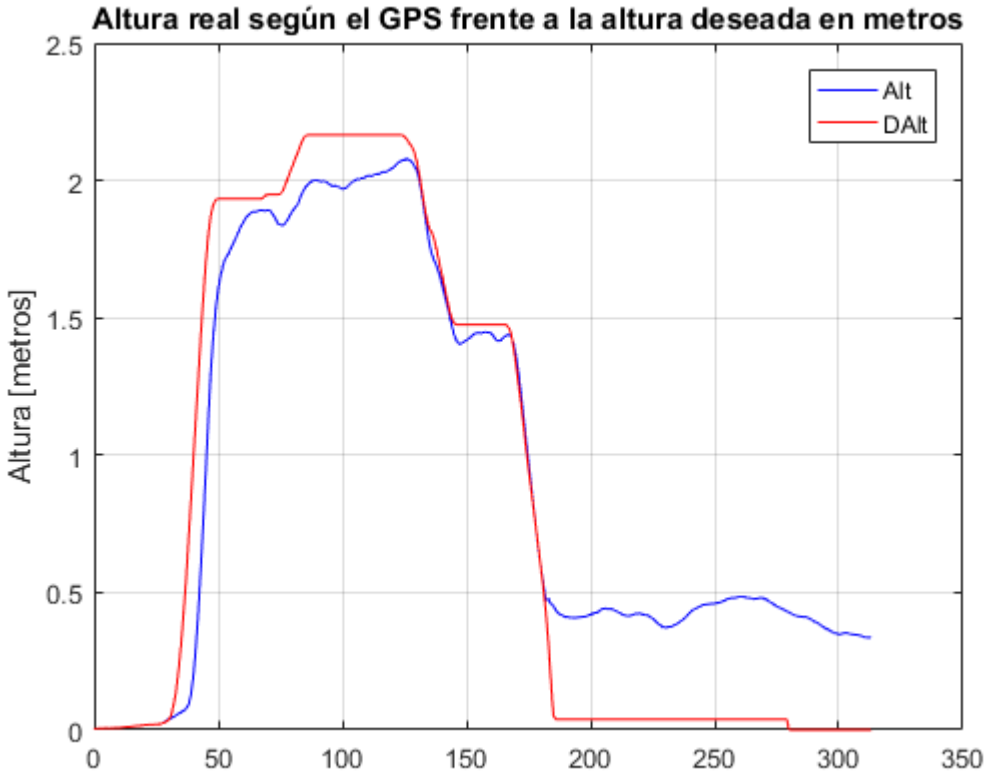


Ilustración 10-3. *Altura del vehículo según el GPS frente a la altura deseada*

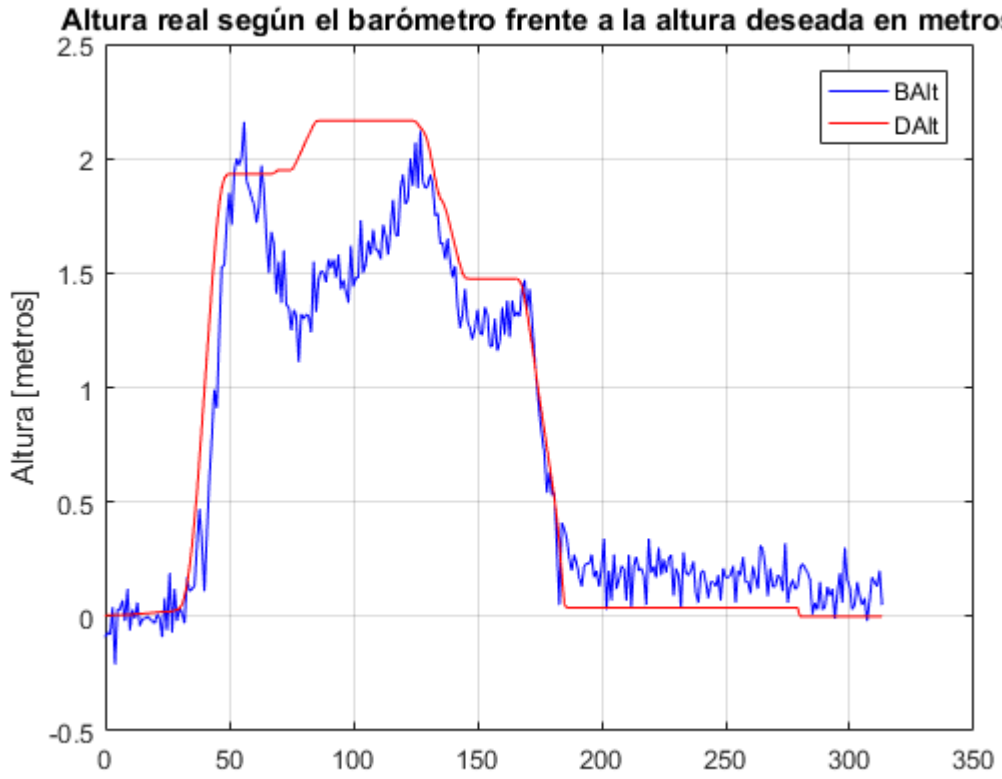


Ilustración 10-4. *Altura del vehículo según el barómetro frente a la altura deseada*

Como se puede observar, en este vuelo, *Erle-Copter* pudo mantener de manera más o menos satisfactoria la altura deseada durante el tiempo en el que estuvo activo el modo *Alt Hold.*

10.3 Medidas proporcionadas por la *IMU*

Otros valores representables, son las medidas obtenidas por los sensores acelerómetro y el giroscopio.

- **Giroscopio**

El giroscopio capta las velocidades de rotación en cada uno de los 3 ejes de posición (x, y, z) en grados/segundo. Los parámetros que ofrecen están información son *GyrX, GyrY* y *GyrZ.*

Se exponen en tres gráficas individuales para su mejor visualización.

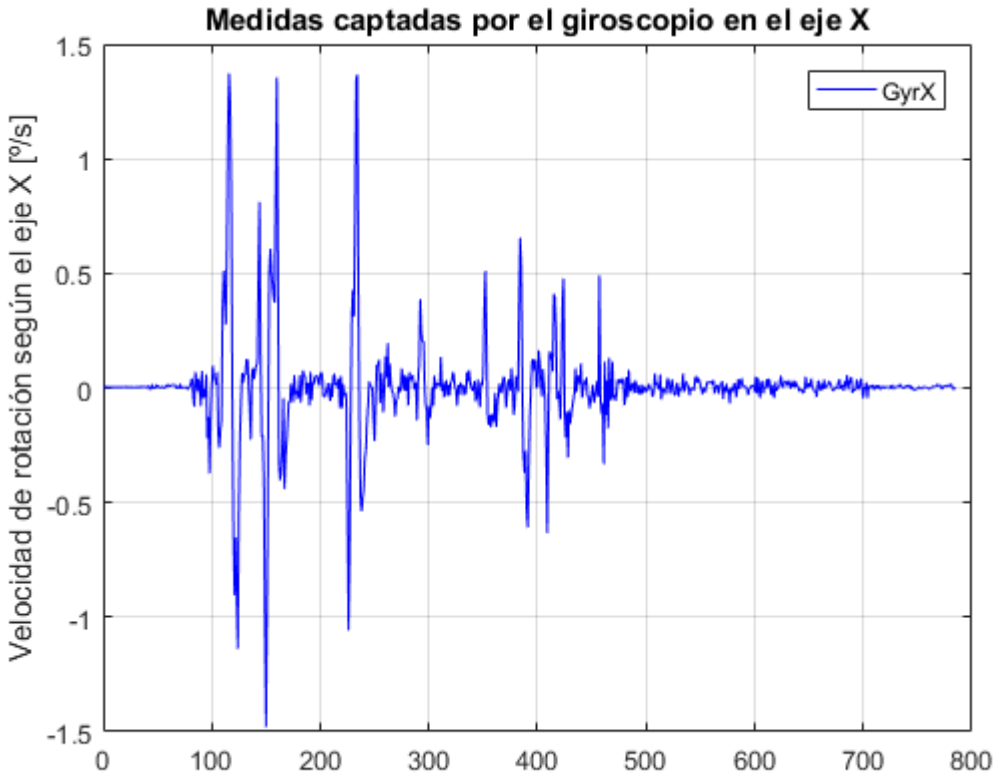


Ilustración 10-5. *Medidas del giroscopio según el eje X*

Ilustración 10-6. *Medidas del giroscopio según el eje Y*

Ilustración 10-7. *Medidas del giroscopio según el eje Z*

- **Acelerómetro**

De manera similar, el acelerómetro capta las aceleraciones en cada uno de los 3 ejes de posición (x, y, z) en m/s^2. Los parámetros que ofrecen están información son *AccX, AccY* y *AccZ.*

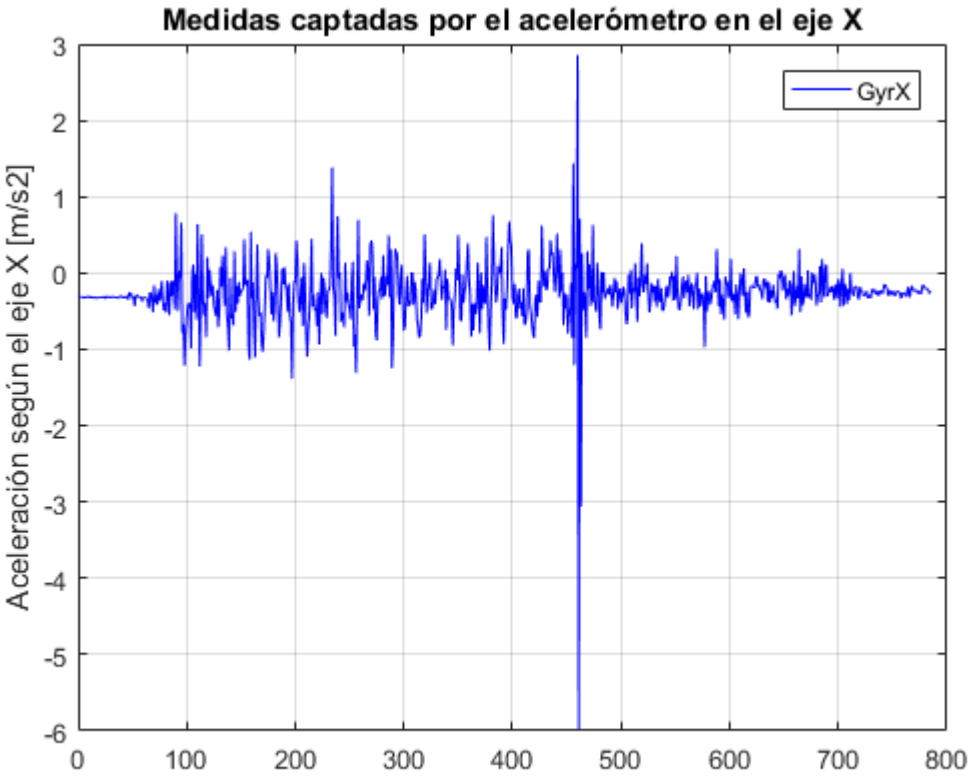


Ilustración 10-8. *Medidas del acelerómetro según el eje X*

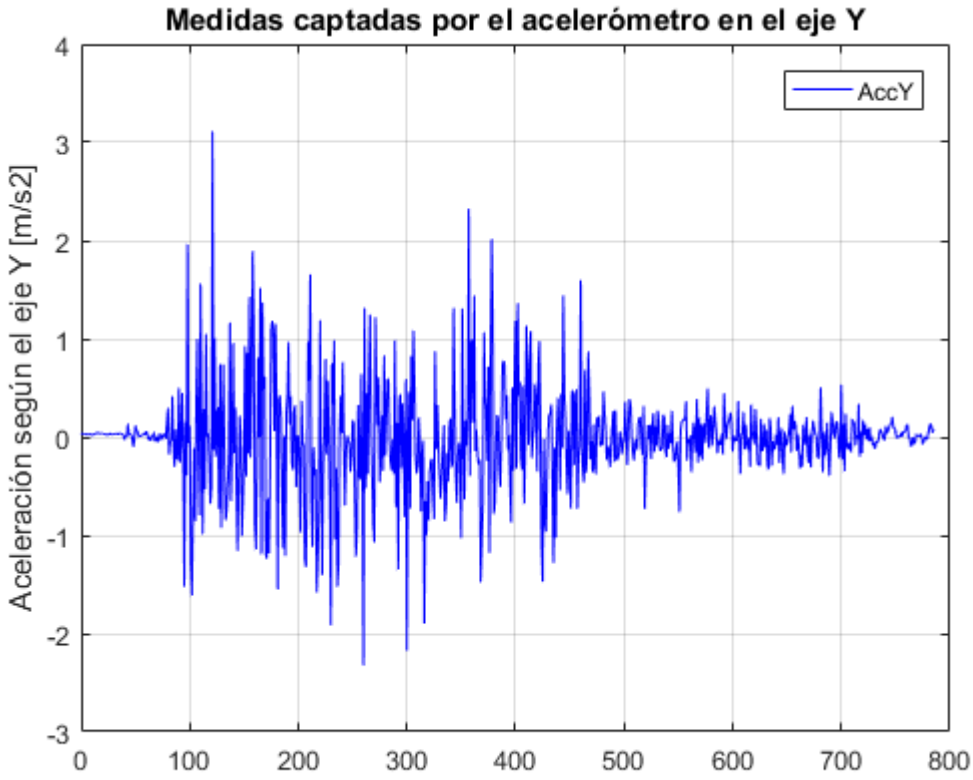


Ilustración 10-9. *Medidas del acelerómetro según el eje Y*

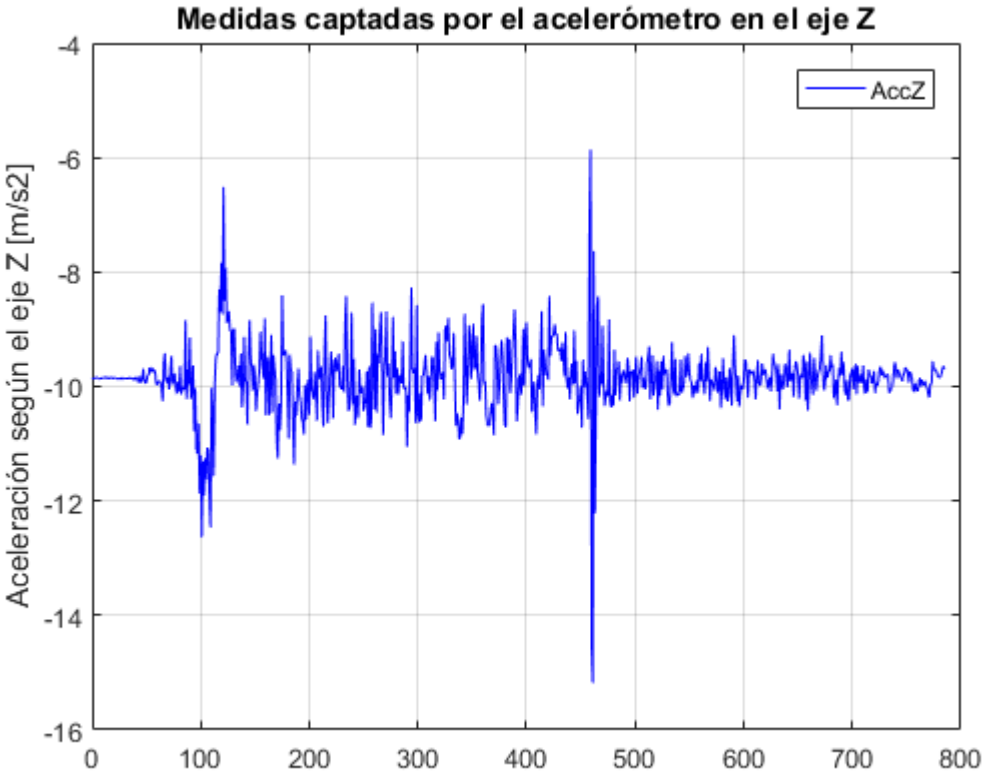


Ilustración 10-10. *Medidas del acelerómetro según el eje Z*

- **Representaciones *ATT***

```
% Representaciones ATT
t = (0:1:785);    % Vector de 786 unidades de longitud

% Ángulo Roll
figure(1)
plot(t,ATT(:,4),'r',t,ATT(:,3),'b'); grid; % Roll (rojo), Roll deseado
(azul)
title('Ángulo roll del vehículo frente al ángulo roll deseado');
legend('Roll','DesRoll');
ylabel('Ángulo roll [°]');

% Ángulo Pitch
figure(2)
plot(t,ATT(:,6),'r',t,ATT(:,5),'b'); grid; % Pitch (rojo), Pitch deseado
(azul)
title('Ángulo pitch del vehículo frente al ángulo pitch deseado');
legend('Pitch','DesPitch');
ylabel('Ángulo pitch [°]');
```

- **Representaciones *CTUN***

```
% Representaciones CTUN

t = (0:1:313);    % Vector de longitud 314 unidades

figure(1)
plot(t,CTUN(:,8),'b',t,CTUN(:,7),'r'); grid; % Altura real según GPS
(azul), Altura deseada (rojo)
title('Altura real según el GPS frente a la altura deseada en metros');
legend('Alt','DAlt');
ylabel('Altura [metros]');


figure(2)
plot(t,CTUN(:,9),'b',t,CTUN(:,7),'r'); grid; % Altura real según barómetro
(azul), Altura deseada (rojo)
title('Altura real según el barómetro frente a la altura deseada en
metros');
legend('BAlt','DAlt');
ylabel('Altura [metros]');
```

- **Representaciones *IMU*. Giroscopio**

```
% Representaciones_IMU
% Giroscopio

t = [0:1:785];  % Vector de 786 unidades de longitud

figure(1)
plot(t,IMU(:,3),'b'); grid; % GyrX
title('Medidas captadas por el giroscopio en el eje X');
legend('GyrX');
ylabel('Velocidad de rotación según el eje X [°/s]');

figure(2)
plot(t,IMU(:,4),'b'); grid; % GyrY
title('Medidas captadas por el giroscopio en el eje Y');
legend('GyrY');
ylabel('Velocidad de rotación según el eje Y [°/s]');

figure(3)
plot(t,IMU(:,5),'b'); grid; % GyrY
title('Medidas captadas por el giroscopio en el eje Z');
legend('GyrZ');
ylabel('Velocidad de rotación según el eje Z [°/s]');
```

- **Representaciones *IMU*. Acelerómetro**

```
% Representaciones_IMU
% Acelerómetro
t = [0:1:785];  % Vector de 786 unidades de longitud

figure(1)
plot(t,IMU(:,6),'b'); grid; % AccX
title('Medidas captadas por el acelerómetro en el eje X');
legend('GyrX');
ylabel('Aceleración según el eje X [m/s2]');

figure(2)
plot(t,IMU(:,7),'b'); grid; % AccY
title('Medidas captadas por el acelerómetro en el eje Y');
legend('AccY');
ylabel('Aceleración según el eje Y [m/s2]');

figure(3)
plot(t,IMU(:,8),'b'); grid; % AccZ
title('Medidas captadas por el acelerómetro en el eje Z');
legend('AccZ');
ylabel('Aceleración según el eje Z [m/s2]');
```

REFERENCIAS

[1] http://erlerobotics.com/blog/ [En linea]

[2] http://forum.erlerobotics.com/top/all [En linea]

[3] http://ardupilot.org/ [En linea]

[4] https://pixhawk.org/ [En linea]

[5] http://diydrones.com/ [En linea]