ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Trabajo de Fin de Grado

Desarrollo de Sistema de Aterrizaje Autónomo Preciso para Multicóptero

AUTOR: Sergio López Milán
DIRECTOR: Juan Carlos Sánchez Aarnoutse
Octubre/2017

Autor	Sergio López Milán
Email del autor	serlopezmilan@gmail.com
Director	Juan Carlos Sánchez Aarnoutse
Email del director	juanc.sanchez@upct.es
Título del proyecto	Desarrollo de Sistema de Aterrizaje Autónomo Preciso para Multicóptero
Descriptores	Multicóptero, Aterrizaje preciso, Pixhawk, Mission Planner, Visión artificial, Raspberry Pi, Pixy, IR-Lock, RTK, Emlid Reach

Resumen

Un GPS comercial que pueda ser equipado en un multicóptero tiene una precisión mayor a 1 m en las mejores condiciones. Dependiendo de las aplicaciones que se tengan que realizar puede ser más que aceptable, pero hay casos en los que esta precisión no es suficiente y es necesario aumentarla con otros métodos.
En el presente trabajo se evalúan con pruebas reales distintas técnicas para incrementar esa precisión con la intención de realizar maniobras de aterrizaje preciso sobre una plataforma.
En este trabajo se estudian dos técnicas distintas para conseguir esos objetos como son la visión artificial y GPS con precisión de centímetros.

Titulación	Grado en Ingeniería Telemática
Departamento	Tecnologías de la Información y las Comunicaciones
Fecha de presentación	Octubre 2017

Tabla de contenido

1. Introducción

1.1. Planteamiento inicial

Un GPS comercial que pueda ser equipado en un multicóptero tiene una precisión mayor a 1 metro en las mejores condiciones. Dependiendo de las aplicaciones que se tengan que realizar puede ser más que aceptable, pero hay casos en los que esta precisión no es suficiente y es necesario aumentarla con otros métodos.
En el presente trabajo se evalúan distintas técnicas para incrementar esa precisión con la intención de realizar maniobras de aterrizaje preciso sobre una plataforma.
En este trabajo se estudian dos técnicas distintas para conseguir esos objetos como son la visión artificial y GPS con precisión de centímetros.

1.2. Objetivos

A continuación, se enumeran los principales objetivos que se aspira conseguir en el desarrollo del trabajo:

- Conocer el funcionamiento de un multicóptero y los componentes necesarios para su montaje.
- Estudiar las distintas tecnologías relacionadas con aterrizaje preciso para obtener las ventajas e inconvenientes de cada una.
- Instalar, configurar y realizar de pruebas con cada una de las tecnologías usadas en aterrizaje preciso.

1.3. Organización de la memoria

Los apartados siguientes de este trabajo se estructuran de la forma descrita a continuación.

Capítulo 2: Dispositivos y tecnologías. Se expone una idea general de los distintos dispositivos y tecnologías que se van a usar a lo largo del trabajo, dando una descripción de cada uno de ellos.

Capítulo 3: Aterrizaje preciso mediante visión artificial. Se muestran las distintas tecnologías probadas en un sistema de aterrizaje que hacen uso de una cámara y procesado de imágenes. En este capítulo se estudia la API Dronekit [1] , la cámara Pixy [2] y el sistema IR-LOCK [3].

Capítulo 4: Aterrizaje preciso con sistema RTK. En este capítulo se presenta la tecnología GPS RTK [4] y se realiza un estudio de las distintas posibilidades en el mercado. Finalmente se hace uso de Emlid Reach [5] para aterrizaje preciso.

Capítulo 5: Conclusiones. Por último, en este capítulo se extraen las ideas principales que se han conseguido con el trabajo junto con los problemas que se han encontrado durante la realización. Además, se exponen las líneas de investigación y mejora para llevar a cabo en un futuro.

2. Dispositivos y tecnologías

En este capítulo se recopilan los distintos dispositivos y tecnologías usadas durante el proyecto, tanto dispositivos hardware como lenguajes de programación y librerías.

2.1. Multicóptero

Un multicóptero, también llamado dron, es un helicóptero que dispone de varios motores para su movimiento, normalmente 3, 4, 6 u 8 motores. También pueden llamarse atendiendo al número de motores que tiene cada uno, siendo un tricóptero, quadcóptero (Fig. 1), hexacóptero y octocóptero (Fig. 2) respectivamente. Pueden presentar diferentes configuraciones atendiendo al lugar en el que están ubicados los motores e incluso si están invertidos o en posición normal, pero esa clasificación cae fuera del propósito de este documento.

Fig. 1 DJI Phanthom 4

Fig. 2 Octocóptero

Para que un multicóptero pueda volar y moverse, es necesario que la mitad de sus hélices se muevan en sentido horario y el resto en sentido antihorario[1]. Hace uso de variadores de velocidad electrónicos (ESC) en los motores para poder adaptar las revoluciones a las que gira cada motor y así poder hacer giros, elevaciones y rotaciones.

Una aeronave posee 3 movimientos (Fig. 3) además de ascender y descender. Estos movimientos son cabeceo (pitch), alabeo (roll) y giñada (yaw). Cada uno de los movimientos se consigue actuando más sobre unos motores que sobre el resto atendiendo al siguiente esquema (Ver Fig. 4).

[1] Obviamente un tricóptero presenta un número impar de hélices por lo que requiere de un mecanismo adicional. La explicación de este mecanismo cae fuera del alcance de los objetivos de este proyecto.

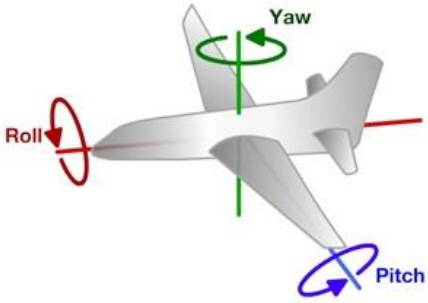


Fig. 3 Esquema de movimientos

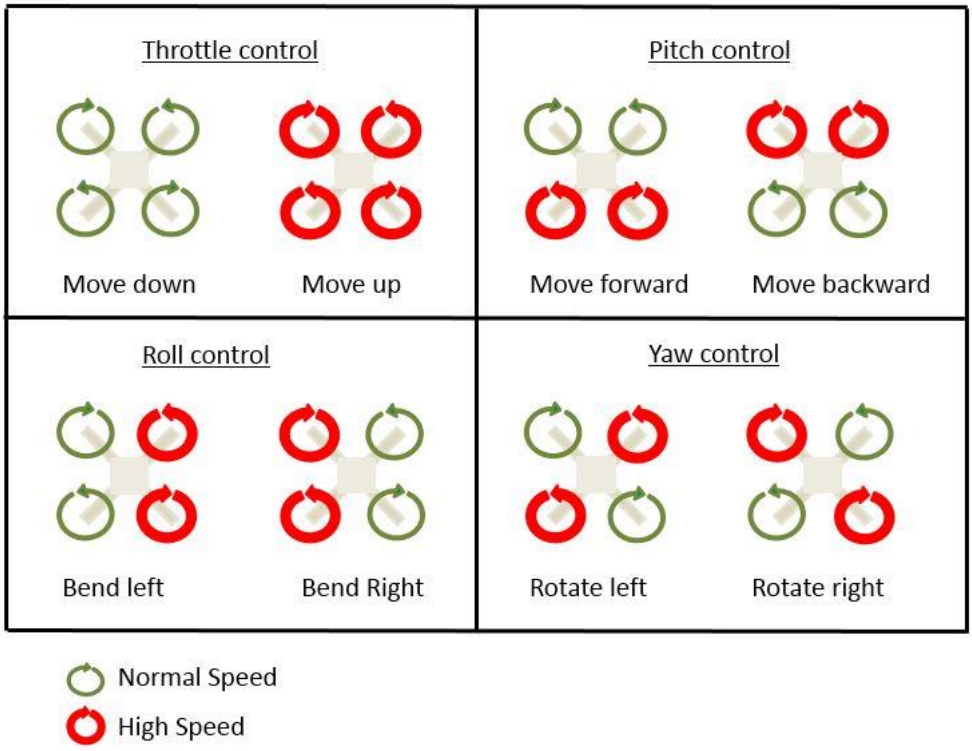


Fig. 4 Rotación de los motores para realizar los distintos movimientos

Una vez entendido el modo de funcionamiento de los multicópteros y su clasificación, se realizará una introducción a la controladora de vuelo que se encarga de transformar las órdenes enviadas mediante el mando de control remoto a los motores, ayudándose de distintos componentes como el GPS, acelerómetro, barómetro, brújula, etc.

2.2. Pixhawk

Existen distintas controladoras de vuelo en el mercado, pero este trabajo se centra en la controladora *Pixhawk* [6] (Fig. 5) desarrollada bajo código y hardware libre por *3D Robotics* [7] y *Ardupilot* [8]. Dicha controladora cuenta con un acelerómetro 3D, giroscopio, barómetro y magnetómetro; y en cuanto a protocolos de comunicación cuenta con 5 UART, bus CAN, SPI, I2C y ADC.

Fig. 5 Conexiones Pixhawk

Para poner en funcionamiento un multicóptero basado en *Pixhawk* son necesarios ciertos componentes adicionales, a continuación se enumeran los imprescindibles:

- Un receptor radiocontrol para actuar sobre el multicóptero.
- Un módulo de potencia para alimentar la controladora mediante la batería.
- Variadores de velocidad electrónicos (también conocidos como ESCs debido a su denominación en inglés – *Electronic Speed Controller)* para modificar las revoluciones de giro de los motores. Evidentemente, serán necesarios tantos como motores tenga la aeronave.
- Opcionalmente, es recomendable equipar al sistema con un GPS con brújula y un módulo de telemetría para poder tener datos en tiempo real del multicóptero en la estación base.

2.3. Pixhawk 2.1

Durante todo este proyecto se ha trabajado con la controladora de vuelo *Pixhawk* comentada en la subsección anterior, aunque en la última fase se ha usado su nueva versión Pixhawk 2.1 [9](Fig 6). Esta última controladora está desarrollada por *ProfiCNC* [10] junto con Ardupilot. Cuenta con mejoras significativas en cuanto a precisión y redundancia lo que hace que sea recomendable en una aplicación profesional.

Esta controladora está dividida en dos partes:

- **Carrier Board**: Se trata de la placa electrónica principal. Cuenta con todas las conexiones para los componentes del multicóptero. Tiene doble redundancia en alimentación y permite conectar dos GPS simultáneamente para mejorar la precisión.

- **Cube**: Cuenta con un sistema de medición inercial (IMU) de triple redundancia. Tiene 3 acelerómetros, 3 giróscopos, 3 magnetómetros y 2 barómetros. Gracias al formato modular, el sistema de medición está separado de la electrónica principal para no producir interferencias. Cuenta con un filtro para eliminar el ruido de las medidas debido a las vibraciones.

Fig. 6 Conexiones Pixhawk 2.1

La gran ventaja de esta controladora para el propósito de este proyecto es la doble conexión de GPS disponible. Gracias a esta característica se puede hacer uso de dos GPS de manera simultánea.

2.4. Mission Planner

Para cargar el firmware en la controladora y poder configurar todos los parámetros del multicóptero se usa el programa *Mission Planner* [11] que se puede descargar de forma libre desde la página de *Ardupilot*. Este programa está disponible tanto para el sistema operativo *Windows* [12] como *Linux* [13].

El programa se divide en varias pestañas a través de las cuales se tiene acceso a los distintos parámetros y configuraciones.

- **Flight Data:** En esta pestaña se encuentran los parámetros que ha definido el usuario para que se muestren, así como un mapa para mostrar en tiempo real la localización del multicóptero gracias al GPS.
- **Flight Plan:** En esta pestaña se muestra un mapa sobre el que pueden crearse misiones automáticas estableciendo *waypoints* a seguir por el multicóptero.
- **Initial Setup:** Esta pestaña a su vez tiene otro sub-menú en el que se puede cargar el firmware a la controladora. Además, una vez que se conecta la controladora al programa a través del puerto USB (o a través de un radioenlace

de telemetría) aparecen dos sub-menús más para configurar el hardware obligatorio y opcional instalado en el multicóptero.

- **Config/Tuning:** Esta pestaña contiene la configuración más avanzada del multicóptero en la que se tiene acceso a todos los parámetros. Será desde esta pestaña donde se configurarían los parámetros de *Precision Landing* o múltiples GPS.

2.5. *Raspberry Pi*

Una *Raspberry Pi* [14] es un computador del tamaño de una tarjeta de crédito desarrollada por la fundación Raspberry Pi en Reino Unido e ideada principalmente para el sector educacional.
Existen varios modelos de *Raspberry Pi* aunque en este trabajo se usa la *Raspberry Pi 3 model B* (Fig. 7) debido a que se trata del último modelo lanzado por lo que cuenta con mayor poder de cálculo.

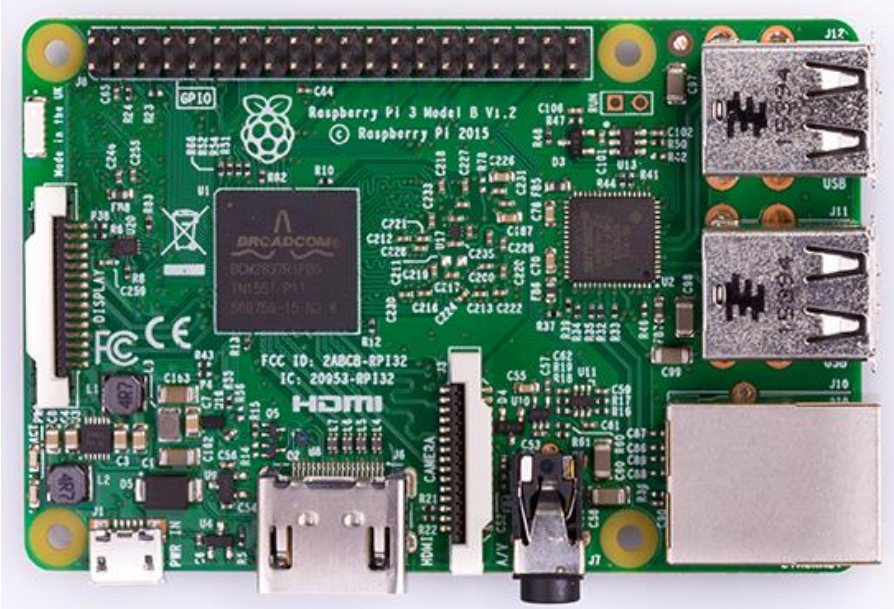


Fig. 7 Raspberry Pi 3 model B

Este modelo cuenta con tarjeta micro-SD, 4 interfaces USB, puerto Ethernet, puerto HDMI, conexión Jack de 3.5mm, GPIO y bus de comunicación para pantalla y cámara. Además, tiene interfaz Wi-Fi y Bluetooth Low Energy.

En cuanto al Sistema Operativo, soporta varios S.O. pero se usará *Raspbian* [15], una versión adaptada de *Debian* [16] ya que es el oficial y está muy documentado.

En este trabajo se hace uso tanto de una *Raspberry Pi* como de una *Raspberry Pi Camera* [17] para procesado de imágenes.

2.6. DroneKit API

Se trata de una plataforma de aplicaciones para multicópteros desarrollada por *3D Robotics*.
Con esta plataforma se pueden conseguir tres tipos de servicios:

1. Aplicaciones web para monitorizar el multicóptero, descargar tanto logs como imágenes o videos en tiempo real a través de protocolos *REST*.
2. Computaciones on-board para realizar cálculos costosos a través de un ordenador de a bordo, y que éste maneje la controladora de vuelo.
3. Desarrollo de aplicaciones destinadas a ofrecer servicios de estación base tanto en móviles como en ordenadores.

Esta plataforma ofrece varios lenguajes de programación para sus distintos servicios. Ofrece *Python* [18] y *Rest API* para desarrollar las aplicaciones web; Python para la computación on-board y aplicaciones de ordenadores; y *Android* [19] para aplicaciones móviles (próximamente ofrecerá la posibilidad de programar aplicaciones para *iOS* [20]).

Mediante esta API se puede acceder a todos los parámetros del multicóptero y actuar sobre él como si se tratase de una emisora de forma remota. También ofrece la posibilidad de crear simuladores de multicópteros (SITL) para probar los códigos desarrollados.

2.7. Pixy

Se trata de una cámara de visión robótica (Fig. 8) capaz de reconocer y seguir varios colores y formas a la vez. Gracias a su poder de procesamiento, es capaz de reconocer una superficie de un color determinado y mediante una de sus interfaces enviar información relativa al objeto como puede ser la posición, el tamaño, etc.

Fig. 8 Cámara Pixy

Esta cámara cuenta con numerosos protocolos de comunicación, así como una salida para servos, siendo muy versátil e ideal para robótica. Cuenta con comunicación mediante SPI, I2C, UART y USB.

- Pixymon

Para acceder a los parámetros de configuración de la cámara se usa el programa *Pixymon* [21] en su versión 2.0.4 ya que con esta versión se tiene acceso a los distintos parámetros de cada elemento y de esta manera poder ajustar el umbral de detección elemento a elemento (en versiones más antiguas se realiza un ajuste general).

2.8. IR-LOCK

IR-LOCK [3] es un dispositivo comercial basado en la cámara Pixy modificada (Fig. 9) para trabajar con un beacon de infrarrojos. Se trata de la misma cámara a la que le han instalado un filtro y lente IR. De esta manera solo detecta infrarrojos. El dispositivo que hace de beacon, llamado *MarkOne* [22] *(Fig. 10)*, está compuesto por una matriz de leds IR. Para este dispositivo se usa la versión 1.0.2 beta de *Pixymon* que recomienda el fabricante, ya que, a pesar de tener bastantes menos parámetros de configuración, viene preparada para detectar el beacon IR *MarkOne*.

Fig. 9 Cámara Pixy modificada

Fig. 10 Beacon MarkOne

2.9. Protocolo MAVLink

MAVLink [23] es un protocolo de comunicación usado en pequeños vehículos no tripulados. Principalmente se usa en la comunicación entre la estación base y el vehículo, aunque también puede utilizarse en la comunicación entre el vehículo y ordenadores de a bordo. En este proyecto se usa este protocolo para la comunicación entre nuestro ordenador de a bordo (*Raspberry Pi*) con la controladora de vuelo además de la estación base.

2.10. Emlid Reach

Se trata de un sistema que hace uso de dos dispositivos GPS y señales de todos los satélites GPS disponibles para aumentar considerablemente la precisión de un GPS convencional obteniendo precisión de centímetro. Uno de los dispositivos debe hacer las funciones de dispositivo base, y se debe situar fijo en un punto. Una vez configurado y puesto en marcha enviará las correcciones de coordenadas a otro dispositivo móvil ubicado en el multicóptero.

Hace uso de la hipótesis de que, si en un punto hay un error en el cálculo de las coordenadas GPS, en otro punto cercano habrá el mismo error, por ello se debe usar en distancias entre ambos dispositivos menores a 10km.

En este trabajo se ha realizado un estudio de los distintos sistemas RTK disponibles en el mercado, entre los que se encuentran *Emlid Reach* [5], *Here+ RTK* [24], *Swift Nav Piksi* [25], *DROBIT GNSS* [26], *Septentrio AsteRx-m UAS* [27], *North RTKITE* [28].

Debido al presupuesto, la elección se limitó a los dos primeros ya que el resto superan los 2000€ de coste, aunque como ventaja proporcionan mayor precisión en un menor tiempo de puesta en vuelo dado que usan tanto la banda L1 como la L2. Además, este trabajo es una toma de contacto con esta tecnología por lo que se asume que si un sistema basado únicamente en L1 es factible (es decir, ofrece una buena precisión pero con el inconveniente de un mayor tiempo de puesta en vuelo), es altamente probable que sea factible con un sistema basado en L1 y L2, ofreciendo un menor tiempo de puesta en servicio.

En cuanto a *Emlid Reach* y *Here+ RTK*, ambos son *open source* y trabajan en la frecuencia L1 únicamente por lo que tienen mayores tiempos de puesta en vuelo.
Se ha elegido finalmente *Emlid Reach* por el hecho de que tanto el dispositivo que actúa de base como el móvil son exactamente el mismo y a la hora de ampliar será más económico que el kit *Here+ RTK*. Además, *Emlid Reach* lleva más tiempo en el mercado lo que repercute en una mayor comunidad a la hora de solucionar problemas.

El kit *Reach RTK (Fig. 11)* incluye dos módulos GPS con antenas con capacidad RTK ya que cada módulo puede ser configurado para base o móvil. Cada uno de estos módulos va equipado con un miniordenador *Intel Edison* para proporcionar potencia de cálculo y conectividad.

Para la configuración de estos dispositivos, los mismos cuentan con un servidor web local y un AP wifi para conectar un ordenador a dicho AP y así configurarlo mediante el navegador a través de su propia aplicación web *ReachView* [29].

Fig. 11 Kit Emlid Reach RTK

- ReachView

Para poder configurar los distintos parámetros y acceder a los gráficos y valores de *Emlid Reach*, los dispositivos implementan un servidor web con un dashboard. Este dashboard cuenta con varias pestañas:

- **Status**: En esta pestaña se muestra la posición en la que se encuentra el dispositivo así como la calidad de recepción de las distintas señales de satélites.
- **RTK settings**: Se tiene acceso a la configuración del modo de funcionamiento de los dispositivos.
- **Correction input**: Selecciona y configura la interfaz a través de la cual recibe las correcciones de coordenadas.
- **Position output**: Selecciona y configura la interfaz a través de la cual envía la posición correcta hacia la controladora de vuelo.
- **Base mode**: Selecciona el modo a través del cual enviará las variaciones de coordenadas a la estación móvil. Además, permite seleccionar la posición exacta de la base.

Una vez vistas todas las tecnologías probadas y empleadas en este trabajo, en el siguiente capítulo se describirá cómo se ha utilizado cada una de ellas: configuración, pruebas y resultados, así como los motivos por los que se descarta, en el caso de que no sea de utilidad.

3. Aterrizaje preciso mediante visión artificial

En este capítulo se tratarán los diferentes sistemas basados en visión probados en este trabajo. Antes de nada se verá cómo se debe configurar la placa Pixhawk para que acepte los datos de estos sistemas. Para cada uno de los sistemas probados, se dará una explicación del mismo, cómo se configura, así como las pruebas o los problemas detectados.

3.1. Configuración Pixhawk

En primer lugar, antes de comunicar la *Pixhawk* con *Mission Planner* se debe instalar el último firmware a la controladora de vuelo. Para ello, en la pestaña *Initial Setup* se selecciona el tipo de multicóptero.
Una vez actualizada, se establece la comunicación con ella para tener acceso a todos los parámetros de configuración de la misma mediante la configuración del puerto y la velocidad de comunicación. En la pestaña *Initial Setup*, se accede al *wizard* que indicará los pasos a seguir para configurarla de una manera sencilla y rápida.

Es importante tener en cuenta que la función *Precision Landing* sólo está disponible de manera oficial a partir de la versión 3.4 de Arducopter por lo que se debe tener esa versión o superior. Además, para hacer uso de esta característica se necesita tener un sonar o lidar (Light Detection and Ranging o Laser Imaging Detection and Ranging) instalado para medir la distancia hasta el suelo.
También es importante saber cuándo actúa esta característica, ya que sólo funciona cuando se active el modo Land y el multicóptero encuentre el patrón en el que debe aterrizar.

La característica de Precision Landing está desactivada por defecto en el firmware. Para activarla se debe entrar en el apartado Config/Tuning -> Full Parameters List y activar a 1 el parámetro PLND_ENABLED. Tras activar esta función, aparece un conjunto de parámetros asociados, como son el tipo de sistema usado, el offset de la posición de la cámara, etc.
En este trabajo se han modificado principalmente dos parámetros: el parámetro PLND_TYPE que determina el tipo de sistema usado, siendo 1 para un ordenador de a bordo y 2 para el sistema IR-LOCK y Pixy. También se ha modificado la velocidad de descenso, LAND_SPEED, que se recomienda poner al mínimo para mejorar la precisión del aterrizaje.

3.2. Aterrizaje preciso con DroneKit API

En una primera aproximación, se optó por el uso de esta tecnología para el propósito de aterrizaje preciso perseguido en este trabajo, ya que ofrece la posibilidad de realizar tratamiento de imágenes y manejar la controladora de vuelo con un ordenador, en este caso una Raspberry Pi con una cámara.

Con esta tecnología se pretende reconocer un cuadrado de un color determinado con procesado de imágenes, obtener el centro de éste y enviar las coordenadas correspondientes a la controladora de vuelo para que posicionara el multicóptero en el centro.

- Instalación y configuración en Raspberry Pi

Se parte de una imagen de Raspbian recién instalada. En primer lugar, se instala Python 2.7 y la librería Dronekit, así como las librerías necesarias para establecer la conexión entre la Raspberry Pi y la controladora.

```
sudo apt-get install screen python-wxgtk2.8 python-matplotlib python-opencv python-
pip python-numpy python-dev python-serial python-pyparsing libxml2-dev libxslt-dev
sudo pip install future
sudo pip install pymavlink
sudo pip install mavproxy
```

Posteriormente se debe deshabilitar el uso del puerto serie por parte del OS, para que así pueda usarlo la librería.

```
sudo raspi-config (Vease Fig.12)
```

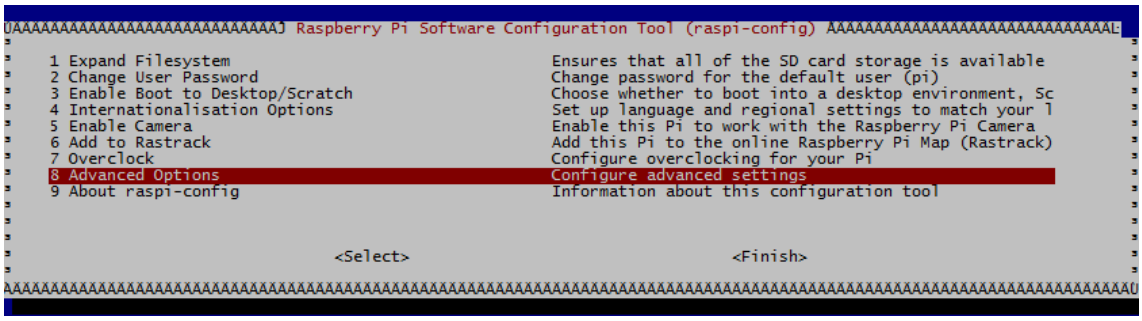


Fig. 12 Menú de configuración Raspbian

Deshabilitar “Serial”, habilitar la cámara e instalar los paquetes de Dronekit API.

```
sudo pip install droneapi
sudo pip install dronekit
sudo pip install dronekit-sitl
```

Una vez instaladas las librerías, la Raspberry Pi está preparada para poder funcionar con la API. Se usará el fichero Landing.py de los anexos, el cual captura imágenes de la cámara y envía la posición calculada al multicóptero.

Las siguientes imágenes (Fig. 13 y 14) muestran el reconocimiento de imágenes que realiza la cámara para detectar la superficie, en este caso una superficie azul oscuro. El punto azul determina el centro de la superficie.

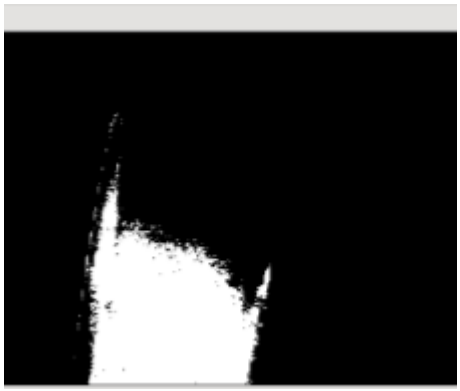


Fig. 13 Imagen tras filtrado de color

Fig. 14 Imagen con superficie detectada

- Problemas

Ante este sistema, el mayor problema encontrado es la ausencia de control por parte del ordenador de a bordo, es decir, la controladora no obedece los controles enviados por la Raspberry Pi. Se realizaron las comprobaciones oportunas y se llegó a la conclusión de que la Raspberry sí envía los mensajes de control, pero no actúa sobre la controladora.

Otro problema encontrado se debe al procesado de imágenes. Se ha realizado un algoritmo básico en el que se reconoce un cuadrado de color azul marino, pero dependiendo de dónde se encuentre, y la luz que le incida, puede no reconocerlo. Además, cuando el multicóptero se posiciona encima, su sombra puede provocar que tampoco reconozca el cuadrado.

3.3. Aterrizaje preciso con Pixy

Tras los problemas encontrados al realizar un procesado de imágenes con una Raspberry Pi y el hecho de no funcionar correctamente la comunicación entre la Raspberry Pi y la controladora, se optó por un dispositivo integrado que realizara el procesado de imágenes y enviara las coordenadas automáticamente, en este caso la cámara Pixy [2].

- Instalación y Configuración de Pixy

La cámara Pixy debe contar con el último firmware disponible (en la fecha de elaboración de este trabajo la versión empleada ha sido la 2.0.8). Para proceder a su actualización, antes de conectar la cámara al ordenador se mantiene pulsado el botón blanco para entrar en el modo programación. En Pixymon [21] aparece 'Firmware upload mode' y una ventana para seleccionar un archivo. Se elige el archivo descargado y se actualiza el firmware.

En la pestaña *configure parameters*, se tiene acceso a guardar las distintas firmas de objetos y modificar sus parámetros de luminosidad. Para guardar una firma nueva se debe pulsar en *set signature* y seleccionar con el ratón el área que se desea guardar. A partir de esto ya es capaz de reconocer ese objeto.

Tras actualizar el firmware, se guarda la firma del color correspondiente con Pixymon. Para ello, en una situación de luminosidad similar a la de trabajo, se enfoca la cámara y se selecciona en Pixymon la superficie a guardar mediante la acción 'set signature 1' y realizando un cuadrado con el ratón.
Además, se debe configurar para que sólo envíe la información de una firma y sólo la mayor superficie mediante los parámetros de configuración *Max block* igual a 1 y *Max box per signatures* igual a 1.
Por último, se establece la salida de datos en I2C mediante *Data out port* igual a 1 y la dirección del dispositivo 0x54.

En cuanto a la instalación de la cámara, se realiza el conexionado tal y como aparece en la imagen inferior (Fig. 15).

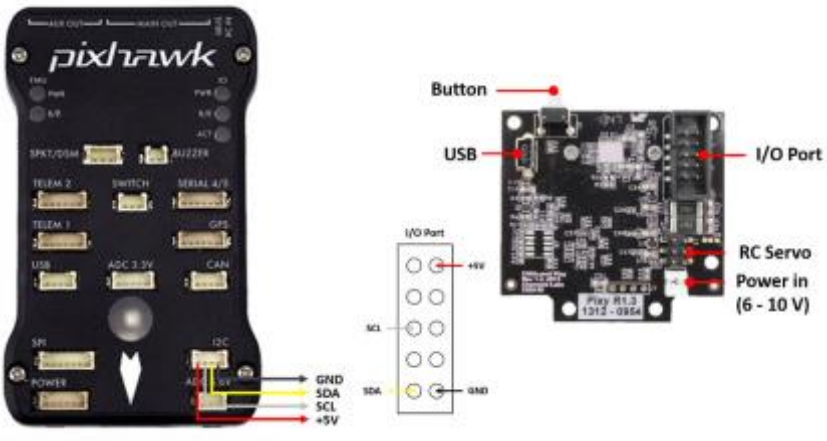


Fig. 15 Pinout de conexionado Pixhawk y Pixy

- Resultados

Con este sistema se ha alcanzado una precisión en torno a 30 cm en el peor de los casos. Se han realizado varias pruebas con varios colores, varios tamaños e incluso colores dentro de otros de mayor tamaño, pero el mejor resultado se ha obtenido con un cuadrado de unos 30cm color azul ya que es unos de los colores que más destaca y mejor distingue la cámara.

- Problemas

A pesar de obtener una buena precisión (menor de 30cm) mantiene el problema de la API anterior, dependiendo el color y las condiciones de luminosidad puede obtener falsas superficies y no aterrizar donde debe.

Además, otro problema añadido al reconocimiento mediante colores es que un vehículo sea del mismo color que la superficie e intente aterrizar sobre el vehículo ya que este objetivo será de mayor tamaño a la superficie en cuestión.

3.4. Aterrizaje preciso con IR-LOCK

Tras los problemas encontrados en el reconocimiento de imágenes por colores tanto en la Raspberry Pi [14] como en la cámara Pixy [2], se ha decidido usar el sistema comercial IR-LOCK [3] basado en infrarrojos. Este sistema cuenta con la ventaja que el software está preparado para detectar el beacon IR y le afectan muy poco las interferencias por lo que siempre lo detecta correctamente.

- Instalación y Configuración de IR-LOCK

Al tratarse de la misma cámara, el procedimiento de actualización de firmware es el mismo que la cámara Pixy, pero en este caso se usa el firmware facilitado por IR-LOCK cuya versión es 1.0.1. De esta manera todo viene preparado para poder trabajar con Pixymon [21] 1.0.2 beta y reconocer nuestro beacon IR MarkOne [22].
También hay que asegurarse de que el puerto de salida es I2C y la dirección es la 0x54.
Una vez configurado, se ajusta la lente de la cámara para que detecte correctamente el beacon a una distancia de 15m. Se puede verificar que reconoce correctamente el beacon ya que la cámara emite una luz roja cuando lo reconoce.

En cuanto al montaje en el multicóptero, se debe proceder de la misma forma que en la cámara Pixy, por tanto, se sigue el esquema superior.

- Resultados

En este caso se ha vuelto a conseguir una precisión en torno a 30cm con un simple beacon bastante más pequeño que la superficie a reconocer mediante colores. Además, no se ve afectado por falsos beacon y por tanto es más seguro frente a errores.

4. Aterrizaje preciso con sistema RTK

Los sistemas anteriores proporcionan una precisión en torno a 30cm que puede ser más que suficiente si se establecen los márgenes adecuados en la plataforma de aterrizaje. Para mejorar esta precisión en mayor medida, se ha hecho uso del sistema GPS preciso *Emlid Reach* [5] con el que se consigue una precisión de escasos centímetros. Con este sistema, para realizar un aterrizaje preciso basta con indicarle las coordenadas de la plataforma de aterrizaje.

- Comunicación con dispositivos Emlid Reach

El dispositivo está configurado de tal manera que, si encuentra una red Wi-Fi guardada anteriormente en él, se conecta automáticamente a ésta. En caso contrario crea un AP Wi-Fi al que es posible conectarse para configurar el dispositivo.

Antes de configurarle una red Wi-Fi, el dispositivo crea un AP Wi-Fi al que conectar un ordenador (o un Smartphone). La contraseña de la red es **emlidreach.** Si se accede a su servidor web local a través de la dirección 192.168.42.1 y puerto 5000[2] se puede añadir una red Wi-Fi con conexión a Internet para permitir al dispositivo realizar sus actualizaciones.

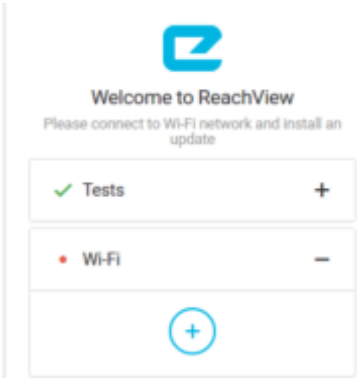


Fig. 16 Interfaz de configuración red inalámbrica

Una vez guardada la red, se pulsa sobre ella para realizar la conexión a la misma del dispositivo (Vease Fig. 16).
En este punto, el dispositivo se reinicia y se conecta a la nueva red indicada. Se accede a su servidor web que contiene el dashboard de configuración a través de su dirección IP de la red local y el puerto 80. Una vez terminadas las actualizaciones, el servidor web muestra el dashboard de configuración (Fig. 17).

[2] En ciertos momentos en los que el servidor web mostrado en el puerto 80 no muestra el dashboard de configuración se debe probar a entrar en el servidor por el puerto 5000 y desde ese punto redirigirse al dashboard mediante un reinicio de la aplicación.

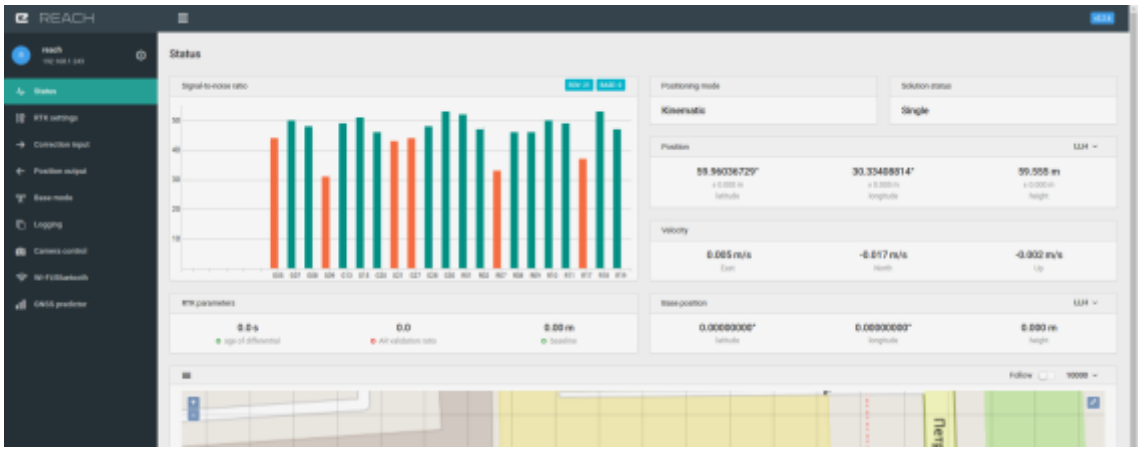


Fig. 17 Dashboard de configuración Emlid Reach

- Instalación del sistema

Este sistema GPS basado en RTK [4] hace uso de dos dispositivos, uno que funciona como base y otro instalado en el multicóptero (denominado *Rover*). (Ver Fig. 18).
El dispositivo base tiene la función de calcular el error en las coordenadas que se encuentra y enviárselo al dispositivo móvil. Éste a su vez, debe corregir su posición mediante el error recibido de la base y enviar la posición corregida a la controladora de vuelo.

Para utilizar este sistema se usará el software Mission Planner como puente entre ambos dispositivos. Por un lado, Mission Planner se conecta al AP Wi-Fi de la estación base para recibir las correcciones de coordenadas. Por otro lado, se conecta el multicóptero por telemetría a Mission Planner de manera que envíe los errores al dispositivo móvil.

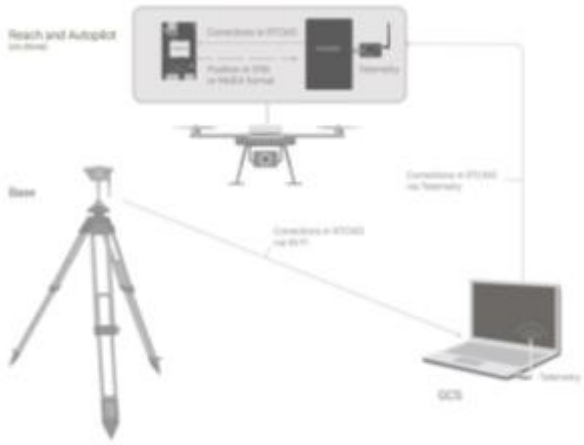


Fig. 18 Esquema de funcionamiento red Emlid Reach

La estación base debe colocarse lo más alejada del suelo y sin obstáculos cerca, por lo que se ha empleado un trípode. Además, para alimentar el dispositivo se usa una batería USB (conocidas también como *power bank*) o el mismo portátil en el que se tiene Mission Planner.

El dispositivo móvil se conecta a la *Pixhawk 2.1* [9] en el puerto GPS 2[3]. El dispositivo *Emlid Reach* tiene dos puertos de comunicación además del puerto micro USB (Fig. 19). Para que se establezca la comunicación con la *Pixhawk 2.1* se debe conectar al puerto que hay junto al micro USB.

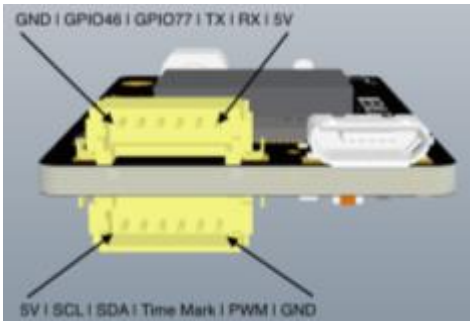


Fig. 19 Pinout Emlid Reach

- Configuración del Rover

Para sincronizar las correcciones recibidas por equipo Rover, mediante la *Pixhawk 2.1* se usa su dashboard para cambiar la interfaz de comunicación en **Correction Input** seleccionando Serial con dispositivo UART a una velocidad de 38400 y formato RTCM3. Por otro lado, para sincronizar las correcciones de posición sobre la *Pixhawk* se cambia la interfaz de comunicación en **Position Output** eligiendo Serial con dispositivo UART a una velocidad de 38400 y protocolo ERB.

- Configuración de la Base

Para configurar el servidor TCP que sirve el equipo base se usa su dashboard. Para ello, se entra en **Base Mode** seleccionando TCP con rol Server y puerto 9000. De esta manera se crea un servidor TCP en el puerto 9000 en la dirección IP correspondiente al equipo base.

- Configuración GPS Inject en Pixhawk

Al igual que ocurría con la característica Precision Landing, el uso de GPS basado en RTK solo está disponible a partir de la versión 3.4 de Arducopter. Además, este GPS

[3] El dispositivo Emlid Reach tiene un consumo medio de 200mA y consumo máximo de 500mA. La controladora Pixhawk 2.1 cuenta con 1A total para compartir entre todos los puertos de comunicación. Se deberán tener en cuenta estas restricciones a la hora de conectar periféricos que consumas bastante a la Pixhawk 2.1 para evitar que el dispositivo Emlid Reach se reinicie en todo momento.

será usado como secundario, de manera que se configura para usar sólo aquel que dé mejor precisión. Es decir, el multicóptero contará con dos GPS.
Se usa Mission Planner como puente entre la estación base y la estación móvil para pasar las correcciones mediante la característica GPS Inject. Para ello se debe conectar por telemetría la Pixhawk al ordenador y éste al AP Wi-Fi de la estación base.

Antes de emparejar el multicóptero con Mission Planner se accede a Initial Setup -> Sik Radio, se desactiva ECC, y para permitir el envío de correcciones sin retrasos mediante telemetría, se configura de manera que se envíen los datos en Raw. Además, también se actualiza el firmware de la telemetría (Fig. 20).

Fig. 20 Configuración Sik Radio

El siguiente paso es configurar la controladora para que reconozca dos GPS. Para ello en Config/Tuning -> Full Parameter List se configurará GPS-TYPE2 a 1, GPS_AUTO_SWITCH a 1 y GPS_INJECT_TO a 1.
Llegado a este punto ya se tiene el dispositivo móvil listo para funcionar con la Pixhawk. Con esta configuración, la controladora de vuelo seleccionará siempre la señal GPS más precisa, que será la que cuente con mayor número de satélites.
En cuanto a la base, se debe realizar la conexión con el servidor TCP que sirve a través de su AP Wi-Fi.
Para que *Mission Planner* actúe como puente entre ambos dispositivos, se habilitarán las opciones avanzadas de *Mission Planner* mediante **Ctrl+F** y en la ventana que aparece se pulsará *Inject GPS.* En la nueva ventana se selecciona *TCP Client* con la IP del dispositivo que actúa como base y el puerto 9000. Tras esto, ya se puede ver en el gráfico las señales recibidas *(Fig. 21)*.

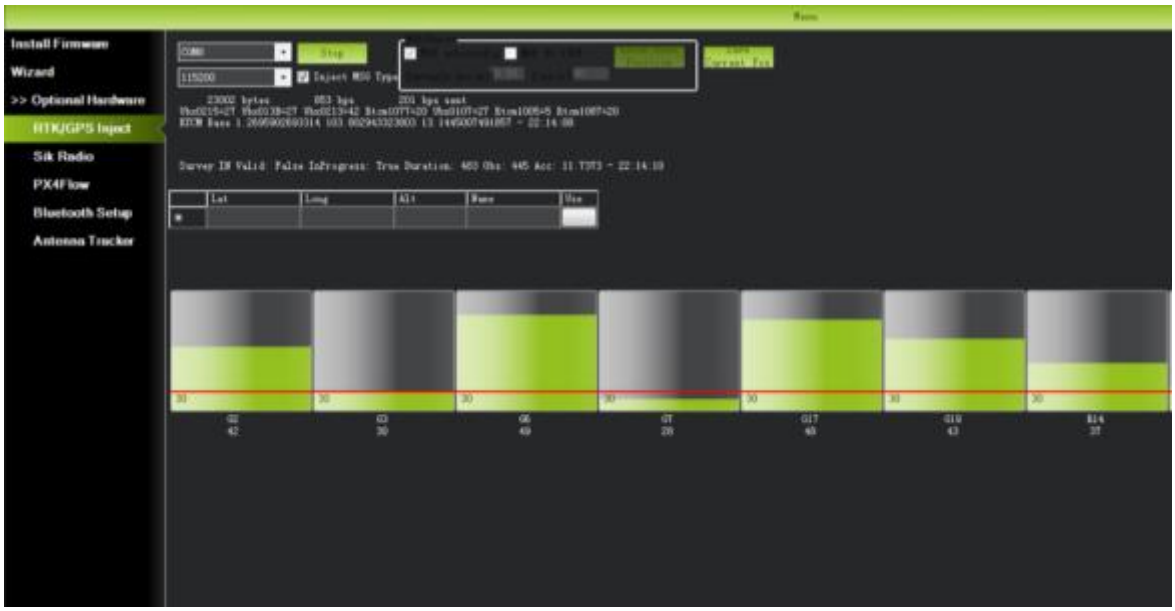


Fig. 21 Función Inject GPS

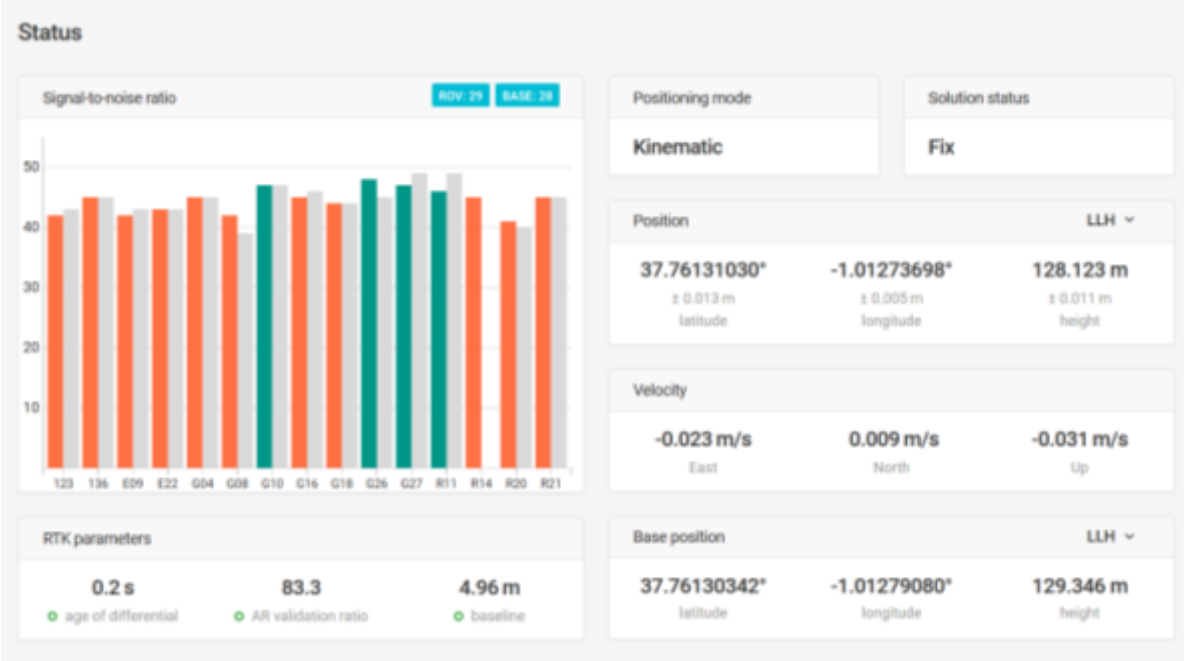


Fig. 22 Emlid Reach estado Fix

- Resultados

Para evaluar la precisión de este sistema se ha establecido un punto de referencia que será la superficie sobre la que aterrizar. Una vez determinada la superficie, se han obtenido las coordenadas y estas a su vez han sido guardadas en el sistema GPS del multicóptero de manera que siempre intentara aterrizar en ese punto.

El sistema cuenta con dos resoluciones de precisión. El estado Fix (Fig. 22) supone una resolución en torno a un centímetro, mientras que el estado Float está en torno a los 15-20cm. El estado depende de la comunicación entre ambos dispositivos que forman el sistema RTK, la distancia entre los dispositivos y la recepción de satélites.

Se ha observado que si el multicóptero realiza movimientos rápidos o bruscos puede producir un cambio de Fix a Float lo que supone una menor precisión. En las pruebas realizadas, se han obtenido aterrizajes en estado Fix teniendo un aterrizaje totalmente preciso sobre la plataforma y en estado Float con una variación de unos 20cm. De cualquier manera, este sistema mejora notablemente los sistemas con visión artificial vistos anteriormente.

5. Conclusiones

El proyecto ha conseguido cumplir el objetivo de realizar aterrizajes precisos con la mayoría de tecnologías estudiadas. Salvo en el caso de la Raspberry Pi junto con su cámara, el resto de tecnologías han conseguido aumentar notablemente la precisión de un GPS convencional.

En este trabajo se han tratados dos tipos de tecnologías como son la visión artificial y el geoposicionamiento preciso. En relación a la visión artificial, se pueden realizar mejoras para optimizar la captación de objetos y el filtrado de errores con el objetivo de evitar reconocer falsos objetos no deseados.
En relación al sistema RTK, se ha conseguido una gran precisión, pero se podría mejorar su fiabilidad usando un sistema más profesional que incluya tanto la banda de frecuencias L1 y L2.

Por último, aunque no cae dentro de los objetivos de este trabajo, en el transcurso del mismo se ha tenido que lidiar con el ensamblado y desensamblado de los drones para probar cada uno de los sistemas y las diferentes controladoras de vuelo, e incluso se han diseñado algunas piezas para la sujeción de los elementos realizadas con una impresora 3D. Además, se ha incorporado a cada uno de los drones utilizados un sistema de visión FPV para que el piloto pueda comprobar en todo momento si el dron está realizando la maniobra de aterrizaje correctamente.

5.1. Líneas futuras de investigación

- Filtros Kalman en Arducopter

El software Arducopter cuenta con un sistema de referencia de posición y rumbo (AHRS) para proporcionar información acerca del rumbo, la posición y la guiñada del multicóptero. El AHRS está compuesto por varios algoritmos entre los que se encuentran los filtros kalman extendidos (EKF).
Los filtros kalman son usados por la controladora para estimar la posición del vehículo, su velocidad y orientación usando de las medidas de los giroscopios, barómetros, acelerómetros, brújula, GPS y velocidad del aire. Además, permite usar medidas de periféricos opcionales tales como sensores ópticos y laser en sus cálculos.
Hay tres versiones de EKF. Las versiones estables de Arducopter usan la versión EKF 2 como fuente primaria de datos para posicionamiento y movimiento del multicóptero. Si la controladora de vuelo cuenta con más de un sistema de medición inercial (IMU), dos instancias distintas de EKF son ejecutadas en cada uno de los IMU de manera paralela y la salida se obtiene de la IMU con mejor resultado.
La versión EKF 3 está disponible, aunque esta deshabilitada en los parámetros de la Pixhawk.

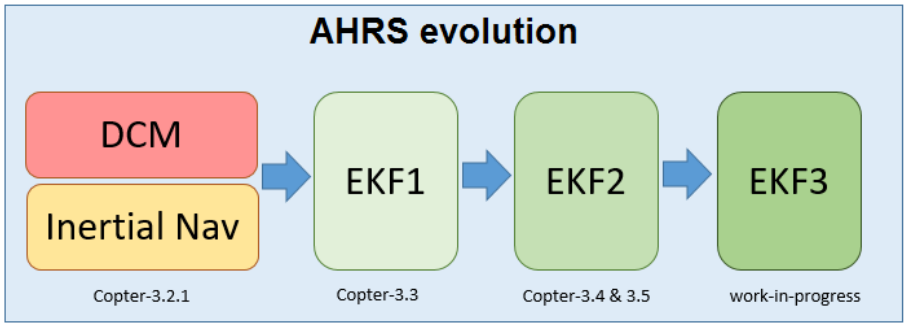


Fig. 23 Evolución algoritmos de detección de posición

Los parámetros más comunes modificados en la configuración de los filtros Kalman son los siguientes:

- **Altitud**: seleccionar la fuente de datos para la altitud entre barómetro, sensor de distancia o GPS
- **Interferencias de altitud**: dar más importancia a la medida del acelerómetro o barómetro.
- **GPS**: seleccionar los datos que se obtienen del GPS.

La ventaja de los filtros Kalman frente a los algoritmos inerciales es el hecho de poder hacer uso de todos los periféricos y sensores para determinar la posición correcta del multicóptero y descartar errores. Cabe la posibilidad de modificar el tipo de filtro usado para tomar decisiones sobre la posición del multicóptero.

- Posibles aplicaciones

Este trabajo se ha enfocado principalmente en conseguir realizar un aterrizaje autónomo y preciso sobre una superficie para realizar distintas operaciones como pueden ser recargarlo o incluso realizar algún tipo de maniobra con algún brazo robótico.

Pero las tecnologías estudiadas no solo pueden servir para lo citado anteriormente, se están desarrollando nuevas aplicaciones en el campo de la mensajería y entrega de paquetes por parte de Amazon, agricultura, etc. En el campo de la salud de he desarrollado algún prototipo de dron capaz de ofrecer socorro a una persona tanto en tierra como en mar, para zonas en las que llegar con personal es más lento que con un multicóptero.

6. Anexos

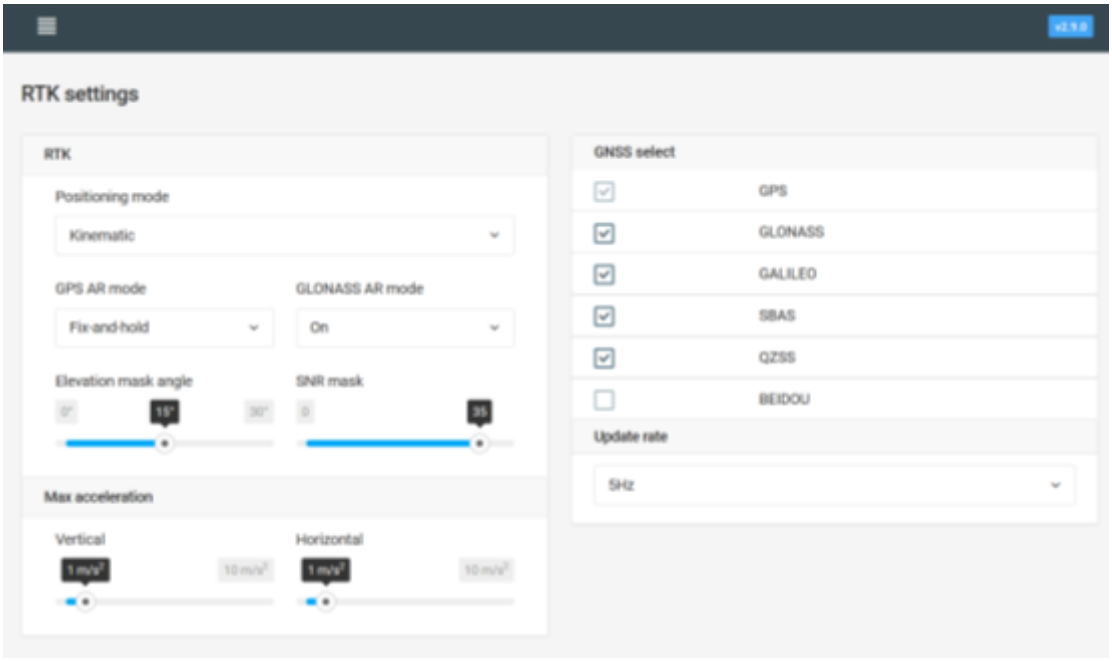
6.1. Landing.py

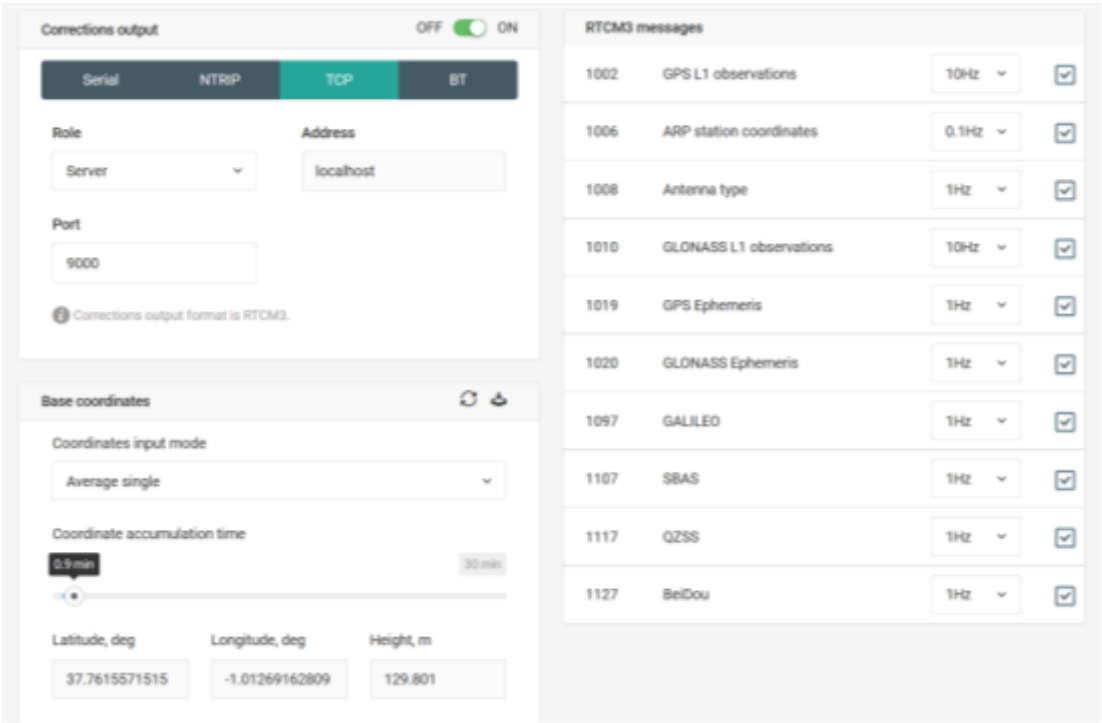
```
#!/usr/bin/env python2
import math

from imutils.video import VideoStream
import imutils
from picamera.array import PiRGBArray
from picamera import PiCamera
import cv2



import time
import numpy as np
from dronekit import VehicleMode, connect

#color settings
hue_lower = 110
hue_upper = 130
saturation_lower = 50
saturation_upper = 255
value_lower = 50
value_upper = 255
min_contour_area = 500 # the smallest number of pixels in a contour

#camera
horizontal_fov = 62.2 * math.pi/180
vertical_fov = 48.8 * math.pi/180
horizontal_resolution = 1280
vertical_resolution = 720

camera = VideoStream(usePiCamera=True, resolution=(1280, 720), framerate=60).start()
#camera = cv2.VideoCapture("./sololink.sdp")
time.sleep(2.0)

vehicle = connect('/dev/ttyS0',wait_ready=True, baud=57600)

def send_land_message(x, y):
```

```
    msg = vehicle.message_factory.landing_target_encode(
        0,       # time_boot_ms (not used)
        0,       # target num
        0,       # frame
        (x-horizontal_resolution/2)*horizontal_fov/horizontal_resolution,
        (y-vertical_resolution/2)*vertical_fov/vertical_resolution,
        0,       # altitude.  Not supported.
        0,0)     # size of target in radians
    vehicle.send_mavlink(msg)
    vehicle.flush()

while(1):
    capture = camera.read()  #Take each frame
    hsvcapture = cv2.cvtColor(capture,cv2.COLOR_BGR2HSV)  #Convert BGR to HSV
    inrangepixels =
cv2.inRange(hsvcapture,np.array((hue_lower,saturation_lower,value_lower)),np.array((hue_upper,saturation_upp
er,value_upper)))  #in opencv, HSV is 0-180,0-255,0-255
    tobecontourdetected = inrangepixels.copy()
    #TODO filter better. binary morphology would be a good start.
    contours,hierarchy = cv2.findContours(tobecontourdetected,cv2.RETR_LIST,cv2.CHAIN_APPROX_SIMPLE)

    contour_sizes=[]
    contour_centroids = []
    for contour in contours:
        real_area = cv2.contourArea(contour)
        if real_area > min_contour_area:
            M = cv2.moments(contour) #moment is centroid
                cx,cy = int(M['m10']/M['m00']), int(M['m01']/M['m00'])
            cv2.circle(capture,(cx,cy),5,(0,0,255),-1)
            contour_sizes.append(real_area)
            contour_centroids.append((cx,cy))

    #find biggest contour (by area)
    biggest_contour_index = 0
    for i in range(1,len(contour_sizes)):
        if contour_sizes[i] > contour_sizes[biggest_contour_index]:
            biggest_contour_index = i
    biggest_contour_centroid=None
    if len(contour_sizes)>0:
        biggest_contour_centroid=contour_centroids[biggest_contour_index]
```

```
    #if the biggest contour was found, color it blue and send the message
    if biggest_contour_centroid is not None:
        cv2.circle(capture,biggest_contour_centroid,5,(255,0,0),-1)
        x,y = biggest_contour_centroid

        send_land_message(x,y)

        #print "Se envia un mensaje"
            print "x ", x
            #print "y ", y

            #cv2.imshow('capture',capture)
            #cv2.imshow('inrangepixels',inrangepixels)

            if cv2.waitKey(1) == 27:
               break

cv2.destroyAllWindows()
camera.stop()
vehicle.close()
```

6.2. Configuración Emlid Reach Base





Bibliografía

DIY Drones - http://diydrones.com
Ardupilot Project - http://ardupilot.org/copter/index.html
Pixhawk Web - https://pixhawk.org
Dronekit API - http://dronekit.io
Python - https://python.org

Referencias

[1] «DroneKit API,» [En línea]. Available: http://dronekit.io.

[2] «Cámara Pixy,» [En línea]. Available: http://cmucam.org/projects/cmucam5.

[3] «IR-LOCK,» [En línea]. Available: https://irlock.com.

[4] «Tecnología GPS RTK,» [En línea]. Available: https://es.wikipedia.org/wiki/RTK_(navegación).

[5] «Emlid Reach,» [En línea]. Available: https://emlid.com/reach/.

[6] «Pixhawk,» [En línea]. Available: http://www.pixhawk.org/.

[7] «3D Robotics,» [En línea]. Available: https://3dr.com/.

[8] «Ardupilot,» [En línea]. Available: http://ardupilot.org/.

[9] «Pixhawk 2.1,» [En línea]. Available: http://www.proficnc.com/content/13-pixhawk2.

[10] «ProfiCNC,» [En línea]. Available: http://www.proficnc.com.

[11] «Mission Planner,» [En línea]. Available: http://ardupilot.org/planner/docs/mission-planner-overview.html.

[12] «Microsoft Windows,» [En línea]. Available: https://www.microsoft.com/es-es/windows/.

[13] «Linux,» [En línea]. Available: https://www.linux.org.

[14] «Raspberry Pi,» [En línea]. Available: https://www.raspberrypi.org/.

[15] «Raspbian,» [En línea]. Available: https://www.raspberrypi.org/downloads/raspbian/.

[16] «Debian,» [En línea]. Available: https://www.debian.org/index.es.html.

[17] «Raspberry Pi Camera,» [En línea]. Available: https://www.raspberrypi.org/products/camera-module-v2/.

[18] «Python,» [En línea]. Available: https://www.python.org.

[19] «Android,» [En línea]. Available: https://www.android.com.

[20] «iOS,» [En línea]. Available: https://www.apple.com/es/ios/ios-11/.

[21] «Pixymon,» [En línea]. Available: http://cmucam.org/projects/cmucam5/wiki/PixyMon_Overview.

[22] «MarkOne,» [En línea]. Available: https://irlock.com/products/markone-v1-

1?variant=18012794371.

[23] «MAVLink,» [En línea]. Available: http://mavlink.org/messages/common.

[24] «Here+ RTK,» [En línea]. Available: http://ardupilot.org/copter/docs/common-here-plus-gps.html#.

[25] «Swift Nav Piksi,» [En línea]. Available: https://www.swiftnav.com/piksi-multi.

[26] «DROBIT GNSS,» [En línea]. Available: https://www.drobit.es.

[27] «Septentrio AsteRx-m UAS,» [En línea]. Available: http://www.septentrio.com/products/gnss-receivers/rover-base-receivers/oem-receiver-boards/asterx-m-uas.

[28] «North RTKITE,» [En línea]. Available: http://gnssrtkmodule.com.

[29] «Emlid ReachView,» [En línea]. Available: https://docs.emlid.com/reach/common/reachview/.