UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

NAVEGACIÓN AUTÓNOMA DE DRONES Y AUTOMATIZACIÓN DE RUTAS APLICADAS A LA LIMPIEZA DE EDIFICIOS

Un Trabajo Final de Grado

Presentado a la Facultad

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

por

Daniel Plaza Rey

En Cumplimiento Parcial

De los Requisitos Para el Grado en

INGENIERÍA DE SISTEMAS ELECTRÓNICOS

Director: Jaime Oscar Casas Piedrafita

Barcelona, Junio 2017

Resumen

Este trabajo recoge el proceso de análisis e implementación de un *software* (SW), basado en *hardware* (HW) de bajo coste, para la automatización de rutas para la limpieza de edificios con UAVs. En una primera parte se recoge el estudio de las opciones disponibles en el mercado para el vuelo autónomo de UAVs, así como el estado del arte actual de la limpieza de edificios. En una segunda parte se recoge el diseño e implementación de un software que permite recorrer la zona a limpiar de forma totalmente autónoma, con las consiguientes pruebas para demostrar su correcto funcionamiento.

Resum

Aquest treball recull el procés d'anàlisi i implementació d'un SW, basat en un HW de baix cost, per a l'automatització de rutes per a la neteja d'edificis amb UAVs. En una primera part es recull l'estudi de les opcions disponibles al mercat, tant de SW com de HW per al vol autònom de UAVs, així com l'estat de l'art de les metodologies de neteja d'edificis. En una segona part es recull el disseny i implementació d'una solució software que permet recórrer la zona a netejar de forma autònoma, amb les corresponents proves per demostrar el seu correcte funcionament.

Abstract

This thesis describes the process of analysis and implementation of a SW solution, based on a low cost HW, to automate routes in building cleaning operatives using UAVs. In the first part of the thesis there is a review of the different options available in the market, both HW and SW, for the autonomous flight of UAVs. In the second part it is described the design and implementation of a software solution that allows performing an autonomous flight all along the surface to be cleaned. At the end, there are described all the tests that have been carried out to prove the correct performing of the software.

Reconocimientos

Quiero dar las gracias en primer lugar a Marc Aicart, coordinador de *HEMAV Academics,* por haber hecho lo indecible desde el primer día por llevar este proyecto hacia delante.

En segundo lugar, agradezco enormemente a mi director, Oscar Casas, el haber aceptado sin pensárselo dos veces, dirigir esta tesis, y por aportar todo un compendio de soluciones allá donde solo había problemas.

No menos importante es la ayuda recibida de los miembros de *Academics*, con especial mención a Sebas, Sandra y Sergio. Sin su ayuda no habría sido posible finalizar este trabajo en un plazo tan limitado.

Quisiera también agradecer a Toni Mas, cofundador de HEMAV, la ayuda técnica prestada, y a los miembros del taller.

Por último quiero dar las gracias a mi familia, por haberme apoyado y animado a finalizar con éxito este proceso, y a mi padre, por haber aportado su sensibilidad literaria a la redacción del proyecto.

Con todos ellos quedo en deuda.

Historial de revisión y aprobación

Revisión	Fecha	Objetivo
0	03/05/2017	Creación del documento
1	15/05/2017	Revisión del documento
2	26/06/2017	Revisión del documento
4	29/06/2017	Revisión final

LISTA DE DISTRIBUCIÓN DEL DOCUMENTO

Nombre	e-mail
Daniel Plaza Rey	daniel.plaza.rey@gmail.com
Jaime Oscar Casas Piedrafita	jaime.oscar.casas@upc.edu

Escrito por: Daniel Plaza		Revisado y aprobado por: Oscar Casas	
Fecha	26/06/2017	Fecha	29/06/2017
Nombre	Daniel Plaza Rey	Nombre	Jaime Oscar Casas Piedrafita
Posición	Autor del proyecto	Posición	Director del proyecto

TABLA DE CONTENIDOS

LISTA DE FIGURAS

1. Introducción y objetivos

En los últimos años hemos sido testigos de un crecimiento extraordinario de los vehículos aéreos no tripulados, comúnmente denominados “drones”. Es tal la diversidad de términos empleados para referirse a estos vehículos que conviene antes clarificar algunos conceptos.

La palabra inglesa *drone* (en castellano, “zángano”) fue acuñada por los británicos a mediados de los años 40 tras crearse el primer vehículo aéreo no tripulado o *Unmanned aerial vehicle* (UAV) con objetivos de observación, y no de combate, en el ámbito militar. El término *drone* (“dron” en su adaptación castellana) engloba un amplio espectro de significados, ya que puede hacer referencia a cualquier vehículo aéreo no tripulado, ya sea pilotado remotamente o tenga navegación autónoma, por ejemplo. De la misma manera, nos referimos exclusivamente a la aeronave, obviando el enlace de comunicaciones y la estación de tierra. Por este motivo se inventó el término UAS, por las siglas en inglés de *Unmanned Aerial System*, que incluye todos los elementos anteriormente citados. Otro de los términos utilizados es *Unmmaned Aircraft* (UA) cuyo significado es equivalente al del UAV. Por último se usa el término *Remotely Piloted Aircraft System* (RPAS) cuando nos referimos a aeronaves no tripuladas pilotadas remotamente. [1] Este último término es el que aparece en la legislación que regula el uso profesional de estas aeronaves actualmente en España, una legislación actualmente escasa y poco desarrollada.

Existe también un término utilizado mayoritariamente por aficionados al radiocontrol, que es el de “multirrotor”. Esta denominación hace referencia al tipo de dron más extendido en el ámbito civil o comercial, que es el que se basa en una serie de motores eléctricos, con hélices en la parte superior del aparato, que le permiten despegar verticalmente y estabilizarse con mucha facilidad. Este tipo de UAV es el que se ha tomado como referencia para el desarrollo de este proyecto.

Debido al rápido avance de las tecnologías UAV y el incremento de su uso comercial aparecen nuevas propuestas en el mercado para el uso de estas tecnologías específicas. La que aquí se desarrolla trata de reducir los costes y sacar los mayores beneficios posibles de los UAVs en el ámbito de la limpieza de edificios. Esta idea surge a partir de la necesidad de una empresa de este sector de optimizar el proceso de limpieza de superficies elevadas en exteriores e interiores de edificios, especialmente en fachadas acristaladas y lugares de difícil acceso. Para ello, solicita a HEMAV S.L., una empresa de servicios basados en la tecnología dron, el desarrollo de un UAV para desempeñar dicha labor y a cuyo estudio pertenece este trabajo.

La primera fase del proyecto encargado a HEMAV S.L. contempla un pilotaje manual de la aeronave por la superficie a limpiar así como el accionamiento de los mecanismos de limpieza. Sin embargo, este TFG pretende dotar a un futuro prototipo de la capacidad para ejecutar toda la operativa de limpieza de forma autónoma, desde el despegue y aproximación inicial hasta el aterrizaje.

Es necesario mencionar que el producto que se ha diseñado para este proyecto no encajaría en la legislación actual sobre el uso de RPAS, ya que como indica este mismo término, los aparatos deben estar pilotados en todo momento y una navegación autónoma está totalmente prohibida, al menos con la norma vigente en el momento de la redacción de este trabajo. La idea es avanzarse a un cambio más que probable de la legislación y poder dar una solución operativa en el momento en que la ley lo permita, lo que, a la vista de lo que avanza el sector puede producirse pocos meses después de la fecha de publicación de este proyecto.

El **objetivo principal** de este trabajo es diseñar una solución de *Software* (SW) en forma de ruta programada, integrada en las plataformas *Hardware* (HW) correspondientes, que se adapte a las necesidades requeridas por una futura operativa de limpieza de edificios. Para ello será necesario hacer un estudio previo de las opciones disponibles hoy en día en el mercado tanto a nivel de HW como de SW.

Es importante mencionar que tanto el estudio como el diseño del sistema recogerán los sistemas de control y comunicaciones en el UAV, dejando fuera los sistemas de propulsión compuestos por motores y *Electronic Speed Controllers (*ESC) y los de distribución de energía, formados por la *Power Distribution Board* (PDB) y la batería. También se deja aparte el diseño e implementación del sistema de limpieza. Por ello, se tratará el UAV como una caja negra con un centro de masas que tendrá que ser capaz de superar una serie de requerimientos que se plantean más adelante. El estudio sobre los elementos anteriormente mencionados se está desarrollando en paralelo a este trabajo por un equipo de 3 personas en la empresa HEMAV, del cual el autor de este TFG forma parte.

En la Figura 1 se presenta el diagrama de Gantt de este proyecto.

				Abril	Mayo				Junio			
	Tarea	Fecha inic.	Fecha fin.	Sem. 4	Sem. 1	Sem. 2	Sem. 3	Sem. 4	Sem. 1	Sem. 2	Sem. 3	Sem. 4
WP 1	Plataformas HW y SW	24-abr	07-may									
	Sistema comunicaciones en UAV	24-abr	07-may									
	Sensores	01-may	21-may									
WP 2	Definición requerimientos misión	15-may	28-may									
	Implementación de la solución SW	15-may	04-jun									
	Simulación e integración del sistema	22-may	11-jun									
WP 3	Verificación final	05-jun	25-jun									

Figura 1. Diagrama de Gantt del proyecto

2. Estado del arte. Operativas de limpieza.

La causa de la creación de este proyecto es la necesidad de una empresa de limpieza de optimizar sus operativas, aprovechándo los avances tecnológicos más novedosos y que más valor puedan aportar a los servicios que ellos prestan.

Por la complejidad del método que se aplica en la actualidad, es necesario un mínimo de 2 operarios para realizar el trabajo de forma segura, aparte de un extenso desarrollo de infraestructura que hace que el proceso sea lento y poco productivo. Hablamos de superficies de entre 15 y 20 m^2 limpiadas por equipo y día.

El proceso realizado actualmente implica un alto riesgo para el operario. I este riesgo a asumir tiene también una gran influencia en los costes. Teniendo en cuenta que cada operario recibe una retribución de 25 € / hora, más un plus de 250 € en concepto de trabajos en elevación y que la maquinaria extra tiene un coste de 500 € / día, hablamos de un coste aproximado por día de 1100 €, lo que resulta claramente excesiva.

Por estas razones, se considera que el uso de UAVs puede ser una alternativa adecuada para la reducción de costos, haciendo más eficiente y segura la limpieza de grandes zonas acristaladas, u otro tipo de superficies, en zonas elevadas. El objetivo es ofrecer a los servicios de limpieza del cliente una herramienta ágil, segura y fiable de limpieza de superficies basada en tecnología dron.

Con esta solución se pretende además aumentar la seguridad de las operaciones hasta llevar la accidentalidad a cero, disminuir los costes relacionados con personal (aumento de eficiencia) e infraestructura y mejorar drásticamente la productividad diaria de cada equipo de limpieza.

En España existe una empresa llamada Cleandrone que ha desarrollado un prototipo de UAV para la limpieza de paneles solares[2]. En el extranjero se han desarrollado varios prototipos tanto de UAVs de limpieza de superficies acristaladas como de robots que se adhieren a la superficie. Uno de estos casos es el del proyecto “Window Cleaning Drones”, que fue presentado al concurso “Drones For Good Award” en 2015[3]. En ninguno de los videos de presentación de estos productos se observa una limpieza continuada de grandes superficies. En el caso de UAVs, la aeronave se aproxima a la superficie, limpia un segmento y vuelve a aterrizar. Por ello, se ve la necesidad de automatizar rutas para grandes superficies de forma continuada para la operativa de limpieza. Para ello el SW y HW utilizado ha de ser capaz de adaptarse a cada edificio o superficie concreta.

3. Estudio de las herramientas disponibles en el mercado para la navegación autónoma de UAVs

3.1. Arquitectura básica de un UAV multirrotor

Existen diferentes tipos de UAV según su tamaño, forma, y funcionalidad. Para este proyecto nos centraremos en el multirrotor ya que dispone de todas las funcionalidades que necesitamos, entre las que destacan su gran estabilidad y facilidad de maniobra, gracias a su estructura.

En todo UAV hay 3 partes bien diferenciadas. En primer lugar tenemos la etapa de alimentación, compuesta por la batería, normalmente hecha de polímero de litio o Li-Po y por una PDB que se encarga de distribuir la energía de la batería a los diferentes elementos del UAV que requieren alimentación. Las PDBs más modernas acostumbran a llevar incorporado un circuito de reducción de voltaje o BEC, que permite obtener de la batería una fuente de voltaje reducido para poder alimentar aquellos elementos del UAV que requieren una potencia menor.

En segundo lugar, tenemos la etapa de potencia, compuesta por ESCs y motores. Estos son los actuadores de nuestro sistema. Su función es ejecutar los movimientos precisos para mover el dron.

Por último, disponemos de una etapa de control formada por una placa controladora de vuelo y una serie de sensores. La placa controladora de vuelo es el cerebro de la máquina y “sensa” y controla todo lo que sucede con el multirrotor. Los sensores son: de un lado el acelerómetro y el giroscopio, que permiten medir la inclinación y orientación del aparato, y por otro, el barómetro y magnetómetro, que miden y permiten mantener una altura determinada y saber la dirección a la que apunta el UAV. Estos sensores son muy importantes para la correcta navegación del UAV. Mientras que los dos primeros son indispensables, los dos últimos pueden ser prescindibles cuando añadimos otros elementos adicionales como el GPS.

Además de lo anterior, siempre tendremos un receptor RF que es el responsable de recibir la señal de radio enviada desde el Control Remoto, que tras interpretar el movimiento realizado por el usuario, lo ha transformado en onda radial. La señal de radio es recibida por el Radio Receptor del multirrotor y transformada en datos que se envían a la controladora de vuelo para que ejecute la instrucción.

3.1.1. Necesidades específicas y limitaciones

La herramienta más utilizada para la ubicación de las aeronaves en el espacio es el GPS. Sin embargo, dada la poca precisión de esta tecnología, no va a ser

muy útil para nuestra tarea. Es por ello que limitaremos su uso al registro de las coordenadas de origen para la vuelta a casa una vez finalizada la misión, así como para la lectura de la altura respecto del suelo durante la misma. En caso de no disponer de buena señal GPS en un lugar concreto, se podrá substituir el despegue y vuelta a casa autónoma por manual. También se hace necesario desarrollar ciertas rutinas que implementen protecciones anticolisión ya que nos moveremos siempre muy cerca de paredes. Por todo esto, necesitaremos sensores para mapear el entorno y podernos ubicar en el espacio.

Existe una gran limitación que es crítica en este tipo de operativas. Se trata de la autonomía del UAV en cuanto a potencia. La opción que más independencia y facilidad de movimiento da a la aeronave es el uso de baterías. Sin embargo, la gran limitación que existe a día de hoy con los UAV es la corta duración de estas. Las baterías de tipo Li-Po, que son las más utilizadas en esta clase de aeronaves tienen una duración máxima aproximada de 20 minutos. Obtener más tiempo de vuelo implica poner más peso. Por ello es necesario valorar otras alternativas. Una de estas es tener un dron “cautivo”, que significa tener la fuente de alimentación en tierra y llevar el dron conectado a un cable que le suministra de forma continua e ilimitada energía para desarrollar sus labores.

Por otro lado se plantea la dificultad de conseguir una buena limpieza. Si bien este hito no puede ser solucionado durante este trabajo, se hace necesario llevar en el UAV instalada una cámara que nos permita ver en tiempo real si la limpieza ha sido correcta o por el contrario hay que repasar alguna zona.

3.2. ¿Qué se entiende por navegación autónoma?

Las aeronaves autónomas son los aparatos voladores capaces de desarrollar una función de forma completamente independiente, sin intervención humana de ningún tipo. Para ello se requiere un entorno conocido a través del cual se diseñan unas rutas para la navegación del UAV o bien un entorno desconocido que es “sensado” por el aparato durante el vuelo. Para ambos casos se requiere una unidad central de inteligencia, en este caso la placa controladora de vuelo, que se encarga de estabilizar la aeronave y procesar los datos que le llegan por distintas vías (emisora RF, telemetría, sensores). Estos son utilizados para decidir y ejecutar las acciones pertinentes para cumplir una serie de objetivos. A su vez, necesitamos una serie de elementos externos que nos permitan recoger información del exterior. Por otra parte necesitamos tener elementos externos como cámaras y sensores que aportan información tanto a la controladora como al usuario, si es necesario, de la situación que rodea al UAV en cada momento.

El hecho de poder automatizar ciertos procesos que tradicionalmente serían llevados a cabo por un piloto, hace que este pueda concentrarse en desarrollar

otras acciones que pueden aportar más valor a la misión, y dejar así las tareas más mecánicas a un ordenador.

En la mayoría de los casos se combina la realización de tareas autónomas por parte del UAV y el control remoto del aparato por parte de un piloto.

3.3. Plataformas existentes

En los últimos años se han ido desarrollando diferentes plataformas para dar la habilidad a los UAVs de poder moverse y cumplir objetivos de forma autónoma.

En muchos casos, el hardware se ha diseñado para dar salida a soluciones software desarrolladas por la comunidad de aficionados tanto a la robótica como al radiocontrol. Por ello se estudiarán varios casos tanto de software como de hardware. A continuación se presentan las opciones más comerciales y que se utilizan en la gran mayoría de UAVs tanto a nivel comercial como de ocio.

3.3.1. Software

El software más utilizado para este tipo de UAVs es ArduPilot. Ardupilot es el software de autopiloto más avanzado y de confianza disponible a día de hoy. Es también el software de autopiloto más testeado. Dada su condición de software libre, está en constante evolución, siempre de la mano del desarrollo tecnológico. Aunque en este caso nos interesa dotar de navegación autónoma a un UAV, ArduPilot está diseñado para trabajar en todo tipo de vehículos, ya sean aéreos, terrestres o acuáticos.

Aparte de los aficionados a la robótica y el radiocontrol, ArduPilot es usado por un gran número de empresas de la industria drone en todo el mundo como 3DR o PrecisionHawk. Ha sido también utilizado para *testing* y desarrollo por una larga lista de corporaciones e instituciones como la NASA, Intel o Boeing, así como por centros educativos.

ArduPilot está formado por un software de navegación alojado en el UAV, normalmente en la placa controladora de vuelo, así como por un software para la estación de control de suelo o Ground Control Station (GCS) que incluye Mission Planner(MP), APM Planner, QGroundControl, MavProxy y otros[4].

3.3.2. Hardware

Existen multitud de plataformas hardware que pueden soportar modos de vuelo autopiloto. La gran mayoría funcionan con ArduPilot, por el hecho de ser ampliamente testeado como se ha comentado. Existen también otras soluciones HW más cerradas, no tan pensadas para desarrolladores, como las que se

comercializan con productos de la marca DJI. A continuación se presentan algunas de las opciones más habituales.

3.3.2.1. APM

Ardupilot Mega (APM) es una placa controladora de vuelo con una IMU de calidad basada en la muy conocida plataforma Arduino Mega. Este autopiloto puede controlar alas fijas, helicópteros multirotor así como helicópteros tradicionales. Soporta 8 canales de radiocontrol con 4 puertos series. ArduPilot Mega consta de una placa principal con el procesador Atmel ATmega2560 (8-bit) y la placa con la IMU que se coloca encima de esta.

La controladora ArduPilot Mega fue diseñada en 2010 por Jordi Muñoz, cofundador de 3DRobotics. Esta placa ha sido de las más utilizadas durante años para la navegación autónoma de todo tipo de vehículos tanto aéreos como terrestres. Con los años ha sido superada por otras plataformas, hasta llegar a dejar de recibir soporte y actualizaciones. Esta placa fue la primera de muchas que se diseñaron para alojar el software ArduPilot[5].

Figura 2. Ardupilot Mega (APM). Fuente: Ardupilot

3.3.2.2. PixHawk

“PixHawk es un proyecto independiente y de hardware abierto que intenta dar soluciones hardware para autopiloto de vehículos de alta calidad a las comunidades académicas, de hobby e industriales a un bajo costo y una alta disponibilidad”, según se dice en su página oficial[6].

Pixhawk se ha convertido en los últimos tiempos en la plataforma preferida para desarrolladores, por las grandes posibilidades que ofrece. Ha superado a su antecesora, APM. En su interior lleva también el software de Ardupilot.

Cuenta con un procesador de 32 bits STM32F427 Cortex M4, y otro coprocesador de 32 bit STM32F103 con 256 Kb de RAM y 2 Mb Flash.

Dispone de una IMU completa formada por giroscopio, acelerómetro, barómetro y magnetómetro, algunos de ellos por duplicado. Incorpora también 5 puertos UART, 2 puertos CAN, 1 puerto SPI y un puerto I2C. Además dispone de numerosas entradas para los distintos tipos de receptores de radio, entre otros.

Figura 3. Controladora de vuelo Pixhawk. Fuente: 3DR

3.3.2.3. Naza

Naza es la solución hardware que ofrece la empresa DJI, referente a día de hoy en UAVs comerciales, con su gama Phantom e Inspire[7]. Esta plataforma ofrece calidad y muy buenas prestaciones RTF, es decir, viene lista para ser volada. Es la opción preferida de aquellos que no quieren ponerse a investigar el funcionamiento de la aeronave y que simplemente quieren salir y volar.

Entre sus defectos destacan el alto precio en todas las versiones disponibles y las pocas posibilidades para desarrolladores. Entre sus ventajas, la posibilidad de empezar a volar tras sacarla de la caja y la facilidad de uso.

Dispone de una IMU completa formada por giroscopio, acelerómetro, barómetro y magnetómetro. No se encuentran especificaciones de procesador en la página oficial de DJI. Exiten varias versiones de esta placa.

Figura 4. Controladora de vuelo Naza. Fuente: DJI

3.3.2.4. Otros

Durante el proceso de investigación para el desarrollo de este TFG he podido observar algunos artículos donde se plantea la posibilidad de utilizar placas de desarrollo como Raspberry Pi 3, que podrían ser utilizadas como controladoras de vuelo sin necesidad de un ordenador extra por poseer bluetooth o wifi incorporados y procesadores más potentes[8][9]. Sin embargo, se han descartado, ya que se intenta utilizar hardware ya testeado para su uso en UAVs y darle una salida comercial.

3.3.3. Plataforma principal escogida para el proyecto

Tras analizar las opciones que nos ofrece cada plataforma se ha optado por utilizar para este proyecto la controladora de vuelo PixHawk por su versatilidad a la hora de ser programada y configurada para diferentes tipos de proyectos con UAVs. Dado que queremos implementar nuestra propia aplicación software para el vuelo de la aeronave, esta placa nos ofrece una amplia gama de formas de comunicación y de control. Se ha descartado la opción de APM ya que Pixhawk es su "sucesora". También se han descartado las controladoras NAZA por su alto precio y por su dificultad de ser programada. El software utilizado será ArduPilot.

3.4. Sistemas de comunicaciones en el UAV

Una vez escogida la plataforma principal, que es controladora de vuelo se hace necesario estudiar el sistema de comunicaciones interno en el UAV, así como las comunicaciones que se producirán entre el aparato y la estación de control remoto o "Ground Control Station" (GCS).

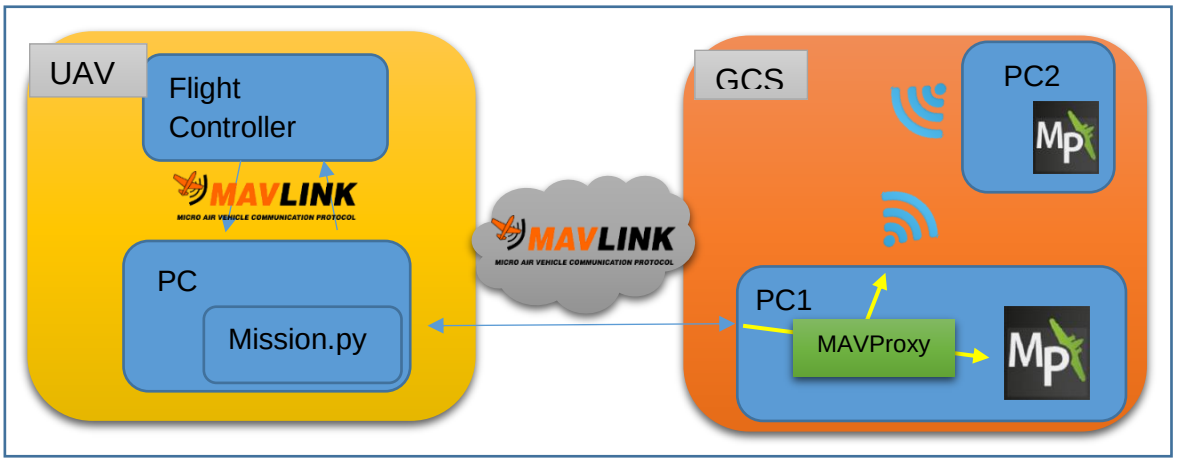

Figura 5. Sistema de comunicaciones UAV

En la Figura 5 se presenta un diagrama de bloques para los diferentes elementos que conformaran nuestro sistema de comunicaciones. Podemos distinguir dos entornos diferenciados. Por un lado, tenemos los elementos que estarán situados en la aeronave. Estos serán la placa controladora de vuelo, Pixhawk, y un pequeño ordenador que contendrá las rutas programadas previamente. Por otro lado, tendremos la estación de control (GCS), en la que podremos ver todos los parámetros importantes sobre el UAV a tiempo real, así como hacer modificaciones en la ruta que se está ejecutando en cada momento.

A continuación se presentan con más detalle cada uno de los elementos que forman este sistema de comunicaciones.

3.4.1. Computadores

Dado que nuestro objetivo es tener un UAV autónomo vamos a necesitar una buena capacidad de computación. Por ello se decide utilizar un segundo elemento de computación que almacene el programa principal y se comunique con Pixhawk.

Tras analizar la información disponible sobre PixHawk y el software ArduPilot, se observa que se recomiendan diferentes plataformas para trabajar como *companion computer* [10]. Algunas de estas son la placa Intel Edison, Raspberry Pi o Nvidia TX1. Por disponibilidad y familiaridad se ha decidido utilizar esta primera aunque por compatibilidad con la Raspberry Pi, todo lo que se desarrolle podrá ser utilizado también en esta última. Ambos pueden alojar en su interior una imagen de Linux y trabajar con Python como se comenta más adelante.

La placa Intel Edison cuenta con un System-on-Chip (SoC) que incluye un procesador dual-core Intel Atom CPU trabajando a 500 MHz y un microcontrolador Intel Quark de 32 bits a 100 MHz. Además dispone de 1 GB de memoria RAM y 4GB de memoria interna. Soporta dual-band WIFI (tanto a 2.4 como a 5 GHz) IEEE 802.11ª/b/g/n.[11]

La Raspberry Pi 3 ha sido actualizada respecto sus predecesoras con un procesador de 64-bit quad-core ARM Cortex-A53 a 1.2GHz. Además, se le ha incorporado WIFI 802.11n y bluetooth 4.1 [12]

3.4.2. Protocolo de comunicaciones: MAVLink

MAVLink o “Micro Air Vehicle Link” es un protocolo para comunicación con pequeños vehículos no tripulados (UV) pequeños. El protocolo MAVLink fue lanzado en 2009 bajo licencia LGPL por Lorenz Meier.[13]

Es utilizado principalmente para la comunicación entre la estación de tierra y el vehículo no tripulado, así como para la intercomunicación entre los subsistemas del vehículo. Este protocolo puede ser utilizado para transmitir datos como la orientación del vehículo, su ubicación GPS y la velocidad que lleva, así como para enviar órdenes a la placa controladora.

3.4.3. MAVProxy

MAVProxy es una potente herramienta software que complementa nuestro GCS. Una de las opciones más interesantes de MAVProxy es que nos permite reenviar mensajes provenientes de nuestro UAV hacia diferentes GCS ubicadas en diversas plataformas como ordenadores, tabletas o teléfonos móviles. De la misma manera, podemos enviar comandos en tiempo real a nuestro UAV desde el GCS.[14]

3.4.4. SITL

El simulador Software in the Loop (SITL) permite ejecutar programas destinados a ser ejecutados sobre distintos vehículos sin necesidad del propio Hardware. Está construido a partir del código fuente de ArduPilot y nos permite observar el comportamiento del código programado sin necesidad de tener nuestro UAV listo y así hacer todas las pruebas deseadas antes de probarlo en la realidad.[15]

SITL permite ejecutar ArduPilot en un PC directamente. Esto es gracias al hecho de que ArduPilot es un software de piloto automático que puede correr sobre diferentes plataformas, como por ejemplo un ordenador. Cuando se ejecuta el SITL los datos de vuelo son recogidos de un modelo dinámico de vuelo de un simulador, contenido dentro del software ArduPilot.

3.4.5. Mission Planner

Mission Planner (MP) es la herramienta que nos permite configurar de forma visual rutas de vuelo para nuestro aparato, así como visualizar toda la información proveniente de la aeronave en tiempo real. A través del protocolo MAVLink recibimos telemetría proveniente del dron con todo tipo de datos sobre el estado del aparato. También nos permite enviar comandos a la aeronave en tiempo real.

4. Sensores disponibles aplicables a la limpieza de edificios

Una vez estudiadas las diferentes plataformas existentes para la navegación autónoma de nuestro UAV y el conjunto de elementos que integrarán el sistema de telecomunicaciones, se hace necesario estudiar y escoger los sensores y elementos externos que van a permitirnos adaptar una simple aeronave autónoma a una operativa tan concreta como es la de limpieza de edificios**.**

Para ello miraremos tres sensores principalmente. Por un lado, tenemos el Lídar, una herramienta que se ha hecho bastante popular en los últimos años para tareas de inspección y estudios de topografía. También valoraremos la posibilidad de usar diversos sensores de infrarrojos y de proximidad, así como de utilizar cámaras y hacer un procesado de las imágenes para evitar obstáculos.

4.1. Detección de obstáculos y distancias

4.1.1. Sensor de ultrasonidos

Los sensores ultrasónicos son detectores de proximidad que trabajan libres de roces mecánicos y que detectan objetos a distancias que van desde pocos centímetros hasta varios metros. El sensor emite un sonido y mide el tiempo que la señal tarda en regresar. Estos reflejan en un objeto, el sensor recibe el eco producido y lo convierte en señales eléctricas, las cuales son elaboradas en el aparato de valoración.

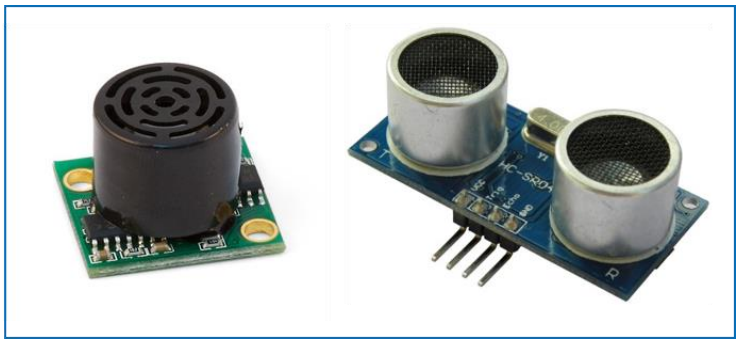

Figura 6. Sensores de ultrasonidos

El BSC-HCSR04, que aparece a la derecha en la Figura 6, es muy utilizado en robótica por su bajo coste (menos de 2 €). Al ser un sensor ultrasónico, trabaja en la banda de los 40 kHz [16]. Según la hoja de especificaciones, tiene un rango de detección de 2 cm a 4 m, a una resolución de 3 mm. El problema que plantea para este proyecto es que apunta a un solo punto, con un ángulo de alcance de 15º, de manera que necesitaríamos unos cuantos para tener vigiladas todas las direcciones.

4.1.2. Lídar

Un lídar (del inglés LIDAR, *Light Detection and Ranging o Laser Imaging Detection and Ranging*) es un dispositivo que permite medir distancias. Utiliza un emisor laser que apunta a un objeto o superficie mediante un haz láser pulsado. Midiendo el retraso entre la emisión del pulso y su detección tras ser reflejado por una superficie, podemos saber la distancia a la que se encuentra.[17]

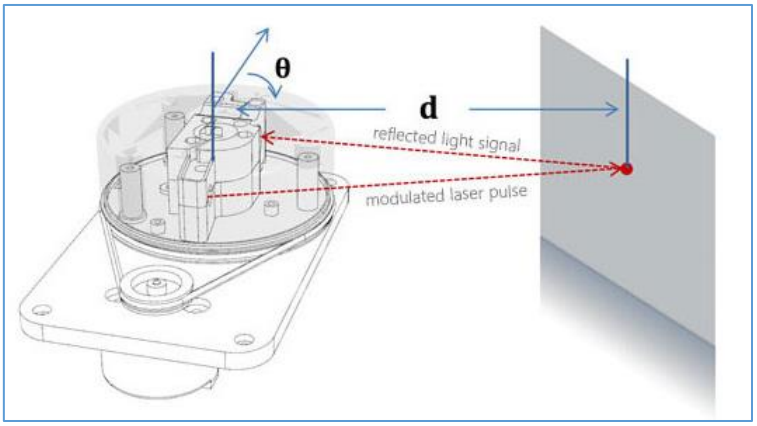

Figura 7. RPLidar 360º. Fuente: www.robotshop.com

El lídar es un sensor que ha ganado importancia en los últimos años gracias a la reducción de tamaño que ha experimentado, lo que hace posible su integración en una plataforma UAV de menos de 25 kg sin problemas. Aunque ha sido ampliamente utilizado para estudios topográficos y de fotogrametría, se está perfilando como una de las principales opciones cuando queremos dotar a un vehículo de funcionalidades de “Obstacle Avoidance”. Para el caso que nos ocupa en este proyecto, será importante, a la hora de limpiar, poder tener una lectura muy precisa de distancia para podernos acercar a la fachada sin llegar a colisionar.

El lídar nos ofrece más precisión a más velocidad respecto los ultrasonidos. Sin embargo, presenta un gran inconveniente, no funciona con superficies acristaladas, ya que el cristal absorbe los rayos láser y por tanto la señal no regresa al lídar de forma que no podemos detectarlo. Dada la disponibilidad de un sensor lídar para este trabajo, se trabajará con el caso de uso de superficies no acristaladas.

Se dispone de un modelo de lídar perfecto para este tipo de operativas, por su reducido peso y tamaño. Se trata del producto RPLÍDAR 360° Laser Scanner. El RPLÍDAR 360º Laser Scanner es un escáner en 2D y 360º de bajo coste. Tiene un rango de detección de entre 20 cm y 6 metros. La resolución es inferior a los 0.5 mm. La nube de puntos en 2D producida puede ser utilizada en mapeo, localización o modelado del entorno. Es capaz de coger 2000 muestras por segundo.[18]

4.2. Supervisión de limpieza y detección de manchas

Para la supervisión de la misión y el análisis de la calidad de la limpieza utilizaremos un sistema de transmisión de vídeo analógico formado por una cámara y un transmisor de vídeo (Figura 8), comúnmente conocido como sistema FPV (*First Person View*). En la estación de tierra dispondremos de un receptor analógico que podremos conectar a un monitor o a un PC. Este tipo de sistemas son los más populares a día de hoy. Su gran ventaja es la baja latencia.

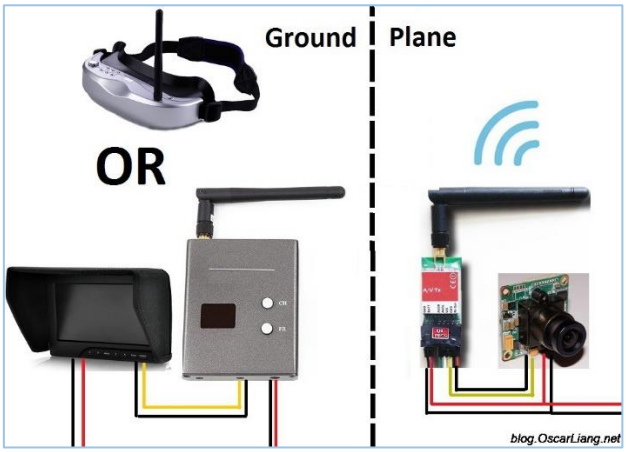

Figura 8. Sistema FPV. Fuente: blog.OscarLiang.net

Empezaremos por la cámara analizando los elementos principales del sistema. En este caso se tratará de una cámara de vídeo normal. Eso quiere decir que se utilizará simplemente para transmitir una imagen real de lo que ve el UAV en cada momento. No utilizamos cámaras térmicas, ni similares.

Existen dos tipos de cámaras que se suelen colocar en UAVs del tipo que se ha comentado. Una forma de clasificarlas es por la resolución que ofrecen.

Figura 9. Tipos de cámara FPV. Fuente: Google imágenes

Lo ideal sería poder utilizar siempre cámaras de alta resolución para cualquier tipo de UAV y de uso. Sin embargo, el principal inconveniente de usar una cámara de alta resolución es su “alta” latencia, que es significativamente mayor

que el producido por cámaras de baja resolución. Es por eso que en casos como el de los drones de carreras, en que lo que se necesita es un retraso mínimo de la imagen debido a la velocidad a la que se mueven, se utilizan cámaras del tipo CCD como la que se muestra en la Figura 9. En este caso tenemos la Foxeer Arrow v2, uno de los modelos más utilizados en drones de carreras. Se trata de una cámara de resolución 600 TVL y con un sensor 1/3 Sony Super HAD II CCD. En este caso el retraso en la recepción es de aproximadamente 30 ms.[19]

Para el caso que nos ocupa, la latencia no es tan preocupante como en el caso anterior. Por eso utilizaremos un cámara deportiva tipo “GoPro” con alta calidad de imagen. En este caso la latencia puede llegar a ser de 120 ms.

El otro elemento del sistema que nos permitirá enviar la imagen del UAV a la estación de tierra es el transmisor de video. Dado que no necesitamos un gran rango de transmisión, debido a la naturalidad de la operativa, no necesitaremos utilizar mucha potencia. Algunas de las potencias más utilizadas por los transmisores de vídeo son 25 mW, 200 mW y 600 mW.

Otro aspecto a tratar es la frecuencia a la que funcionan estos sistemas. Se suele utilizar la banda de 5.8 GHz para no tener interferencias con el sistema radiocontrol que trabaja a 2.4 GHz. En la Figura 10 se ven los canales utilizados y sus frecuencias respectivas en MHz.

CH / FR	CH1	CH2	CH3	CH4	CH5	CH6	CH7	CH8
Fr1 or(A)	5865	5845	5825	5805	5785	5765	5745	5725
Fr2 or(B)	5733	5752	5771	5790	5809	5828	5847	5866
Fr3 or(E)	5705	5685	5665	5645	5885	5905	5925	5945
Fr4 or(F)	5740	5760	5780	5800	5820	5840	5860	5880
Fr5 or(R)	5658	5695	5732	5769	5806	5843	5880	5917

Figura 10. Resumen de frecuencias utilizadas en FPV[20]

Lo ideal es contar con un analizador de espectros en el campo de vuelo para seleccionar el canal menos ocupado y asegurar una alta calidad de señal. En cuanto a la recepción de video en la estación de tierra contaremos con un receptor como el de la Figura 9. Del receptor se utilizan dos cables de alimentación y dos cables con conector RCA de donde obtenemos la señal de video y un polo negativo que podemos conectar al sistema de visualización escogido.

5. Algoritmos de control para la operativa de limpieza de edificios

5.1. Definición de requerimientos e hitos

A la hora de ejecutar una misión de limpieza será necesario establecer una serie de requerimientos a cumplir. Teniendo en cuenta que se trata de una operativa muy cercana a edificios y obstáculos, asegurar la seguridad de la operativa es obligado para el buen fin de la misión. Por ello, es necesario implementar una capa de software dedicada a evitar un choque descontrolado contra las paredes así como para ciertos obstáculos que puedan sobresalir de estas. Este punto es sin duda el más crítico, ya que en este caso, la limpieza, necesitaremos estar en contacto con la pared para limpiarla. Para ello necesitamos asegurar que el dron se mueve en un rango de distancias, respecto de la pared, muy reducido.

Es también importante el concepto de eficiencia. Ajustar los tiempos de cada segmento del vuelo para asegurar la correcta limpieza de la superficie en cuestión, utilizando el menor tiempo posible con el fin de aprovechar al máximo cada vuelo. Para ello es fundamental realizar vuelos de prueba para medir el comportamiento del UAV, en cuanto a velocidades de desplazamiento y tiempos utilizados. Esto es vital para poder reajustar y calibrar los algoritmos finales y dotar de precisión a los movimientos

Finalmente, se implementarán en el programa unos controles que permitan saber cuándo es necesario regresar el UAV a la zona de despegue porque no es seguro seguir volando. Esta situación incluye los casos siguientes:

- Batería insuficiente
- Líquido de limpieza insuficiente
- Superficie total limpiada está cercana a la permitida por seguridad.

5.2. Planificación de las misiones

Se definen las siguientes pautas para la planificación y desarrollo de las misiones:

- Se introducirán en el programa las dimensiones de la superficie a limpiar.
- Esta superficie será dividida en diferentes zonas de manera que, de una forma sencilla, se pueda indicar a la aeronave que ha de relimpiar una zona concreta.
- En caso de que la calidad de la señal GPS sea muy baja el UAV será manualmente pilotado hasta un extremo superior de la superficie. Aun así, se dispondrá de una rutina para ascensión automática.

- Durante todo el programa se ejecutarán de forma periódica rutinas de anticolisión.
- Se definirán varios recorridos posibles, de los cuales se escogerá uno para la operativa de limpieza.
- Una cámara permitirá ver el estado en que quedan las superficies tras ser limpiadas.
- Se ejecutará la "vuelta a casa" cuando se cumpla alguna de las condiciones previamente definidas.

5.3. Implementación del programa

Por motivos de confidencialidad con la empresa HEMAV S.L., no se proporcionan en este documento los códigos desarrollados. Por ello se procede a una explicación descriptiva del programa.

El programa principal encargado de la ejecución autónoma de las misiones ha sido desarrollado en Python. Se utiliza la librería dronekit-python, que facilita las comunicaciones entre la controladora de vuelo Pixhawk y el ordenador de a bordo (Intel Edison/Raspberry Pi 3).

El diagrama de bloques de la Figura 11, presenta a alto nivel, y de forma resumida el funcionamiento general del programa y las órdenes que se envían a la placa controladora. Se utilizan básicamente 3. Estas son: Despegar, Desplazamiento y Vuelta a casa. La primera y la tercera son utilizadas una vez. La orden de desplazamiento es utilizada constantemente para mover el aparato. Es nuestro programa principal el que decide, en función del momento de la misión y el entorno, el tipo de movimiento que se ha de hacer.

Figura 11. Diagrama de bloques del programa principal

5.3.1. Procesado de datos pre-misión

Antes de iniciar la misión es necesario conocer las dimensiones de la pared a limpiar. Las dimensiones son imprescindibles para que el programa adapte la ruta a ejecutar. De la misma manera, necesitamos establecer otros parámetros como la superficie máxima que se permitirá limpiar o la batería mínima antes de regresar el UAV al lugar de despegue. También es posible introducir como

parámetro el tipo de ruta que se quiere ejecutar. Los diferentes tipos de rutas pre-programadas se describen más adelante.

Estos datos serán introducidos como argumentos una vez se ejecute el programa aunque siempre habrán configurados una serie de valores por defecto para estos parámetros. Esto será útil a la hora de simular la operativa.

5.3.2. Inicio de la misión y aproximación a punto inicial

En el caso de que se trate de una misión real y no de una simulación, estaremos pasándole al programa como parámetro el puerto por el que la placa Intel Edison, se conectará con la placa controladora de vuelo. En caso contrario, se ejecutará el SITL.

Antes de despegar, se realizan una serie de comprobaciones que incluyen esperar a que la zona esté despejada, esperar a que el UAV esté preparado para armar los motores y a continuación esperar a que estén armados. Una vez están listos, el UAV despega y se eleva hasta la altura máxima de la superficie en cuestión.

Cuando la aeronave alcance la altitud seleccionada inicia la fase de aproximación. Mediante los datos recogidos por el lídar el UAV detecta dónde está la pared y se acerca hasta estar a la distancia mínima que hemos seleccionado. El UAV será capaz de alcanzar la pared siempre que se encuentre dentro de la distancia máxima a la que es capaz de reconocer un objeto el lídar.

Como punto final a esta primera fase, se ejecuta una rutina llamada calibrateYaw() con la que conseguimos girar el UAV sobre sí mismo (Yaw) para ponerlo exactamente paralelo a la pared, ya que cuando nos dispongamos a limpiar necesitaremos que los aparatos de limpieza estén perfectamente acoplados a la pared.

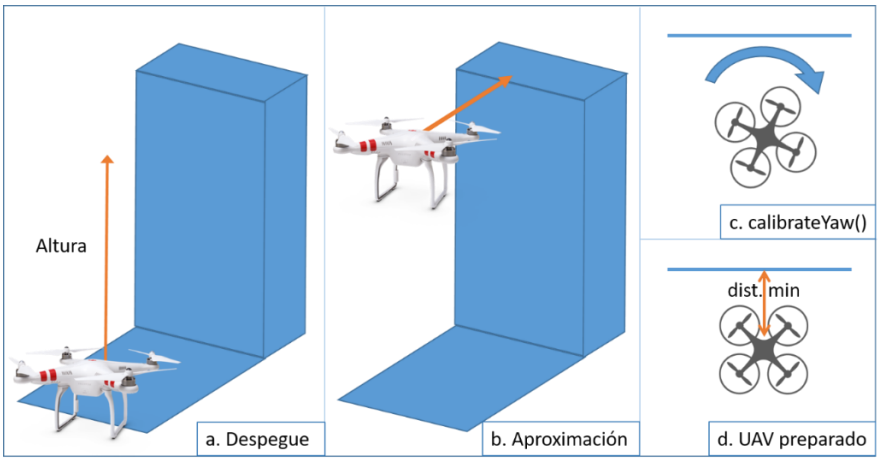

Figura 12. Maniobras de despegue y aproximación a pared

5.3.3. Ejecutar misión

El grueso principal de las labores a desarrollar por parte del UAV se centra en el desplazamiento por la superficie a limpiar siguiendo unas rutas preestablecidas.

El desplazamiento por la superficie se hace con suaves movimientos laterales y verticales. Cada vez que se produce cada uno de estos movimientos se resta la distancia recorrida a las variables *restanteX* y *restanteY.* Se necesita calibrar estos movimientos para que con cada uno de ellos se desplace una distancia aproximada de 1m y de esta manera relacionar el número de movimientos que se necesita hacer con las dimensiones exactas de la superficie: para recorrer 5 m horizontales de pared se harán 5 movimientos de 1m.

5.3.3.1. Evitar colisión

Para asegurar la seguridad de la misión desde el despegue hasta el aterrizaje, en cada iteración que se hace en el programa, e independientemente de en qué fase estemos, se estudian los datos recibidos del lídar y si se detecta un objeto más cerca que la mínima distancia permitida se corrige la posición del UAV, desplazándose en sentido contrario al obstáculo. Mediante software se recogen datos de distancia en 360º y se almacenan en un fichero de texto en la forma {grados Distancia}.

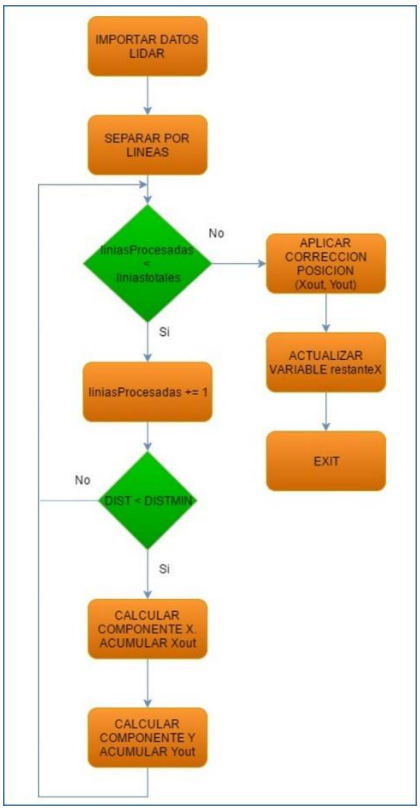

Figura 13. Diagrama de bloques de rutinas anti-colisión

Como se puede observar en la Figura 13, el algoritmo para evitar colisiones se puede dividir en dos partes. Los datos recogidos por el lídar son procesados y almacenados en un fichero de texto. Cuando tenemos los datos separados por ángulo y distancia utilizamos simples reglas trigonométricas para calcular las componentes de profundidad y distancia lateral para cada medida y los vamos acumulando. Al final se hace una media para las dos componentes y se envía un comando a la placa controladora con la información de la rectificación a ejecutar.

Al final se actualiza la variable restanteX. Esta variable guarda la distancia horizontal que falta por recorrer. Por ello, en el caso de encontrar un obstáculo y tener que rectificar la posición, es necesario actualizar esta variable.

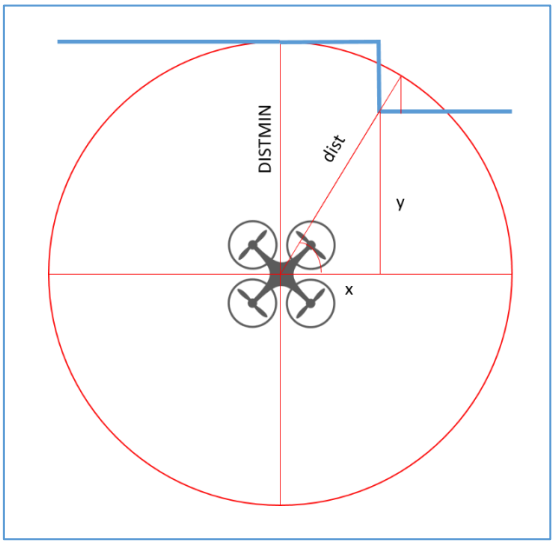

Figura 14. Obtención de componentes para rectificación de posición

5.3.3.2. Modo Seguir pared

Teniendo en cuenta que la finalidad de este proyecto es diseñar una serie de algoritmos para poder limpiar superficies verticales de forma autónoma, se hace necesario que el vehículo esté siempre a la misma distancia de la pared y no sólo que sea capaz de evitar obstáculos. Por ello, para poder limpiar una superficie es necesario que, aparte de separarse de la pared cuando aparece un obstáculo, sea también capaz de acercarse cuando se haya separado demasiado. Para ello se ha desarrollado la función seguirPared(), cuyo funcionamiento es muy parecido al de la función evitarColision(). La diferencia entre estas es que en este caso no sólo es necesario evitar los obstáculos y separarse sino que además hay que acercarse a la pared cuando el UAV se haya separado demasiado con tal de mantener una distancia constante con la pared.

5.3.3.3. Tipos de ruta

Dado que en este trabajo no se pretenden evaluar las técnicas de limpieza más óptimas para una operativa con UAV, se ha decidido diseñar e implementar diferentes tipos de rutas que puedan ser utilizadas una vez se decida este aspecto en futuros trabajos.

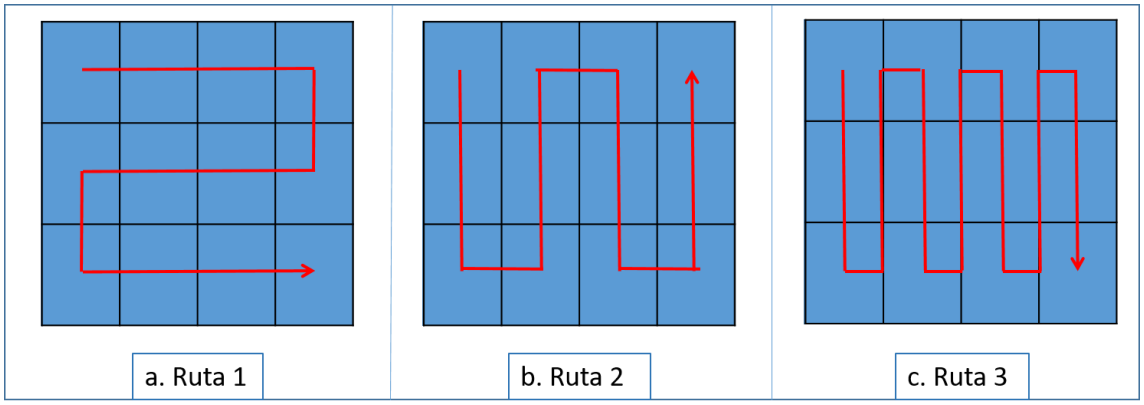

Figura 15. Tipos de ruta

Por eso se han diseñado las rutas que aparecen en la Figura 15. Cada una de estas rutas se adapta a unas necesidades concretas de limpieza.

En el caso de la Figura 15.a el desplazamiento principal es horizontal, y aun siendo la opción que de entrada parece más óptima, no encajaría con una solución de limpieza vertical.

El diseño de la Figura 15.b está pensado para una solución de limpieza vertical, corrigiendo el defecto del tipo de ruta 1. El UAV se desplaza verticalmente por la pared y limpia tanto cuanto sube como cuando baja.

Para el caso en que la limpieza sólo se pueda hacer en sentido bajada se ha implementado la ruta de la Figura 15.c. Esta ruta sacrifica la eficiencia del vuelo ya que debe hacer el doble de desplazamientos verticales que en la ruta 2.

5.3.4. Re-limpieza

A pesar de tratarse de una operativa completamente autónoma, se contempla durante la misión la posibilidad de mandar el UAV a una zona por donde ya ha pasado para realizar una segunda limpieza, en base a una detección visual por parte del operador de una deficiencia en la limpieza. Para ello, dejaremos registrada la última posición en la que se ha realizado la limpieza siguiendo la ruta normal para poder volver una vez finalizado el proceso de re-limpieza. De la misma manera, mantendremos los valores de superficie restante en las dos direcciones vertical y horizontal.

Tras recibir los datos que nos indican la posición a la que hemos de volver, en base a la cuadrícula descrita en el apartado 5.2, se calculan las coordenadas en que se encuentra dicha posición y se da la orden a la placa controladora para dirigir el UAV hacia allí. Como en otras acciones ya comentadas, durante el desplazamiento del UAV hacia la posición indicada, se ejecuta continuamente la función de evitar colision por si algún obstáculo apareciera en el camino.

Una vez el UAV llega a la posición deseada se ejecutaría de nuevo la rutina de limpieza. Hay que recordar, llegados a este punto, que no entra en el ámbito de este trabajo la implementación de dicha rutina. Correspondería a futuros trabajos el diseño de las herramientas de limpieza y su integración en una plataforma UAV.

Finalizada la limpieza, el UAV coge como nueva dirección de destino la que habíamos guardado previamente y al llegar se continúa la rutina normal de limpieza.

5.3.5. Return to launch

La misión de limpieza se ejecutará bajo las condiciones anteriormente presentadas durante un tiempo limitado. A día de hoy esta limitación viene dada principalmente por las baterías, cuyo tiempo de uso es muy reducido. Este tiempo dependerá de diferentes variables como son el peso del UAV o de la cantidad de movimiento que se produzca durante el vuelo. En el caso de utilizar un UAV cautivo, no tendríamos limitación de consumo de potencia y en ese caso podríamos efectuar toda la operativa de limpieza de una vez.

En el momento en que se cumpla alguna de las condiciones predeterminadas para la finalización de la misión se iniciará el proceso de “vuelta a casa” o RTL.

Estas condiciones son: que el nivel de batería sea inferior al recomendado, que el líquido de limpieza disponible sea insuficiente para seguir limpiando o que se haya agotado un tiempo predeterminado. Aunque no se plantea en el esquema principal de la misión, también será posible ejecutar la vuelta a casa de forma manual en cualquier momento.

Al iniciar la misión, queda registrada la ubicación de despegue. En el momento en que se ejecuta la vuelta a casa, estas coordenadas registradas previamente son utilizadas como destino. Durante este proceso, se ejecutará continuamente la rutina de evitar Colision, ya que es difícil saber en qué posición se encontrará el UAV en el momento de ejecutar la vuelta y el escenario de la ruta de vuelta es impredecible. Por ello es necesario sensar continuamente el camino para no chocar.

6. Verificación experimental

Tras desarrollar el programa que automatiza las rutas de limpieza se hace necesario definir los diferentes entornos que se van a utilizar para testear cada parte. Estos entornos son los siguientes:

- Simulación en PC + SITL. Verificación del desarrollo de la ruta.
- Simulación en PC + SITL + Lídar. Verificación del funcionamiento del lídar.
- Intel Edison/Raspberry Pi + Pixhawk + Lídar (en Tierra). Integración del sistema.
- Integración de todos los elementos en UAV

6.1. Simulación del vuelo en PC y SITL (sin lídar)

El primer test que se efectúa consiste en comprobar que cada fase del programa funciona correctamente. Para ello se ha ejecutado en un PC con Sistema Operativo (SO) Windows 10 de 64 bits.

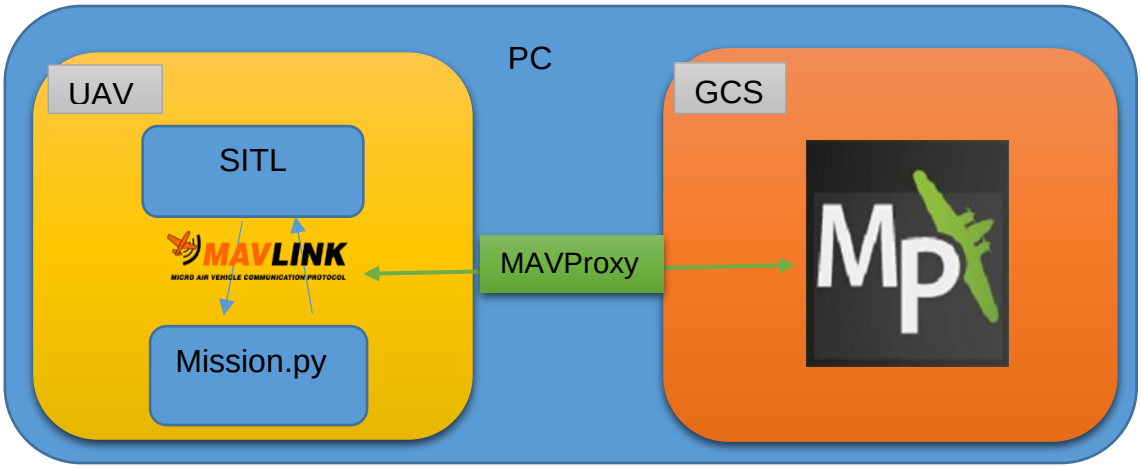

Figura 16. Diagrama de comunicaciones en simulación con SITL

En la Figura 16 se ve un diagrama muy parecido al expuesto al principio del apartado 3.4. Sin embargo, en este caso no tenemos UAV y todo se ejecuta en el ordenador. Por ello, en vez de tener una placa Pixhawk utilizamos el SITL para probar nuestro algoritmo de navegación autónoma, representado en la Figura 16 por Mission.py. A través del programa MAVProxy somos capaces de interceptar la comunicación entre la rutina Mision.py y el SITL. De esa manera podemos ver en el Mission Planner sobre un mapa real lo que pasaría si ejecutásemos nuestro programa en el UAV.

Durante esta simulación no es posible testear algunas funcionalidades como la batería baja o el procedimiento de evitar colisión al no disponer de los datos

provenientes del lídar. Tampoco será posible testear la rutina de aproximación inicial a la pared por la causa anterior, ni la de calibrar *yaw*.

Se utiliza el límite de máxima superficie limpiada (40 m^2 en este caso) para finalizar la operativa de limpieza y ejecutar la vuelta a casa. En Figura AP 1 y Figura AP 2, se presenta la ejecución del programa a través del "cmd" de Windows, con los mensajes que indican la situación del supuesto UAV en cada fase de la misión.

Podemos diferenciar 3 partes. Tras ejecutar el programa con los parámetros requeridos se inicia la conexión con el SITL. A continuación se ejecutan una serie de *tests* para comprobar que el UAV está listo para despegar y a continuación se inicia el despegue. Las comprobaciones incluyen esperar a que el UAV esté listo para armar los motores y a continuación que estos hayan sido armados.

Tras llegar a la parte superior del edificio se inicia el tipo de ruta escogido. Cada línea mostrada es un movimiento ejecutado por la aeronave (Figura AP 1). Se considera que en cada desplazamiento que se produce se recorre 1 m. Para ello es necesario una calibración correcta tras testear el programa en vuelo.

Cuando se supera el límite preestablecido de máximos metros recorridos, se detiene la rutina de limpieza y se ejecuta la vuelta a casa, enviándole a la controladora la orden de cambiar de modo de vuelo a RTL (Figura AP 2). Mientras está volviendo, vamos mostrando la distancia restante hasta casa, calculada a partir de las coordenadas de la posición de despegue y en la que estamos en cada momento. Entre cada una de estas comprobaciones se ejecuta la rutina de evitar colisión. Los valores de altura y distancia hasta casa son recogidos directamente del SITL.

En el momento en que llegamos a la posición de despegue el UAV desarma los motores y finaliza el programa.

6.2. Simulación del vuelo en PC y SITL (con Lídar)

Tras testear el correcto funcionamiento de la ruta principal diseñada, es el momento de integrar el Lídar y dotar a la plataforma de un grado más de inteligencia mediante funcionalidades de "sense and avoid". En esta fase de las pruebas de verificación, se testean las funcionalidades de "esperar a zona despejada", "aproximación inicial a pared" y "seguir pared". Aunque en este caso se utiliza el lídar, el programa está preparado para recoger los datos de un fichero de texto, luego cualquier otro tipo de sensor (p.e. ultrasonido) es fácilmente integrable en el caso que fuera necesario.

Cabe mencionar que en el programa se definen una serie de variables con tal de poder ajustar su funcionamiento a cada situación en función, por ejemplo, de las dimensiones del UAV. Con tal de poder hacer las pruebas cómodamente se han asignado los valores mostrados a continuación:

- Distancia mínima con la pared: 60 cm.
- Distancia máxima con la pared: 75 cm
- Distancia mínima a cualquier objeto para el despegue: 1,5 m

Como se ha comentado en capítulos anteriores, la resolución del lídar es del 1% de la medida de distancia, luego siempre cabe un margen de error en las medidas recogidas. Esto no presenta un problema gracias al diseño del programa. Teniendo esto en cuenta, se pueden ajustar fácilmente los márgenes permitidos para evitar la colisión de la aeronave así como para mantener la distancia con la pared.

La primera funcionalidad se ha comprobado de forma satisfactoria. Tras quitar cualquier objeto del alcance del lídar, se inicia el despegue hasta llegar a la altura seleccionada y se ejecuta la rutina de aproximación a la pared.

En la Figura 17 se pueden apreciar los 3 momentos principales de la aproximación a la pared. Cabe destacar que la parte delantera del UAV estaría mirando hacia la lámina negra. Por este motivo el UAV se mueve hacia delante al detectar la lámina negra que simula el punto más cercano de la pared a la que nos hemos de aproximar.

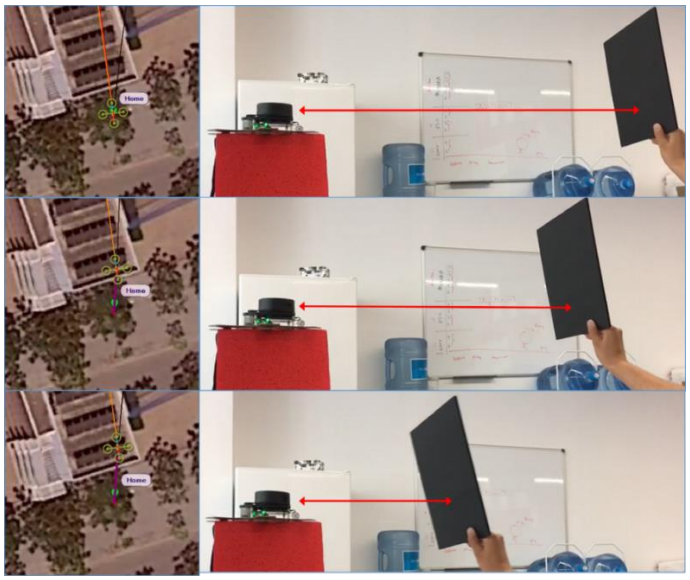

Figura 17. Demostración rutina aproximación a pared

Cuando se alcanza la distancia deseada hasta la pared finaliza la aproximación y empieza la ruta seleccionada. En este punto se comprueba el funcionamiento

de la rutina de seguir pared. Como se ve en las imágenes de la Figura 18, cuando la lámina negra se acerca al lídar por debajo de los 60 cm, el UAV se desplaza para atrás para evitar la posible colisión y mantener la distancia preestablecida. De la misma manera, cuando separamos la lámina por encima de los 75 cm, el UAV se mueve hacia delante para aproximarse a la supuesta pared. En la Figura 18, la línea negra que sale del UAV en cada caso indica la dirección a la que se dirige el UAV.

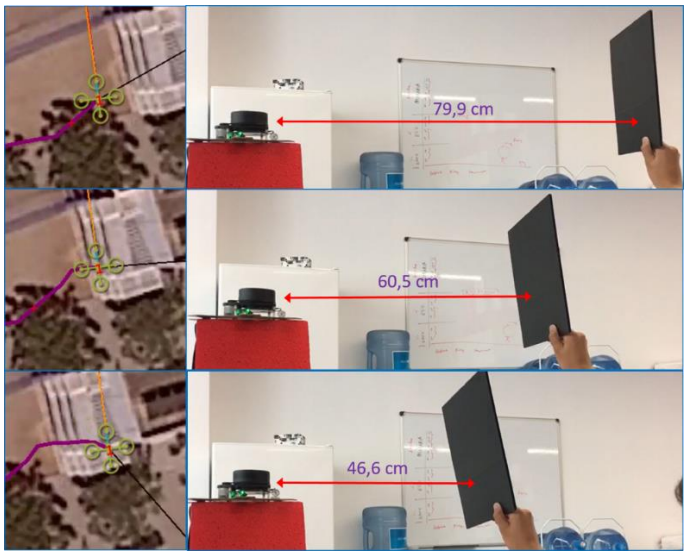

Figura 18. Demostración de rutina de seguir pared

El UAV seguirá la ruta normal con desplazamientos únicamente laterales mientras la distancia entre el lídar y la lámina negra se encuentra entre los 60 y los 75 cm. En el momento en el que situamos la lámina a un lado, simulando un elemento que sobresale de la pared, el UAV deja de desplazarse horizontalmente para separarse de la pared hasta no tener nada a los lados.

6.3. Integración del hardware y software utilizado

Ya se ha comprobado que el programa que se ha diseñado funciona e implementa correctamente las funcionalidades planteadas, al menos en un PC y utilizando el SITL. Llegados a este punto vamos a integrar todo el HW y SW final que va a ir situado en el UAV.

En primer lugar se ha utilizado la placa Intel Edison, en la que se ha cargado un SO Ubilinux(Linux). Se han instalado los paquetes necesarios para la correcta ejecución del programa y su comunicación con la Pixhawk. Sin embargo, ha habido problemas con los drivers del lídar y se ha decidido cambiar a Raspberry Pi. En la Figura 19, se muestra una imagen con la interconexión de los distintos elementos del sistema.

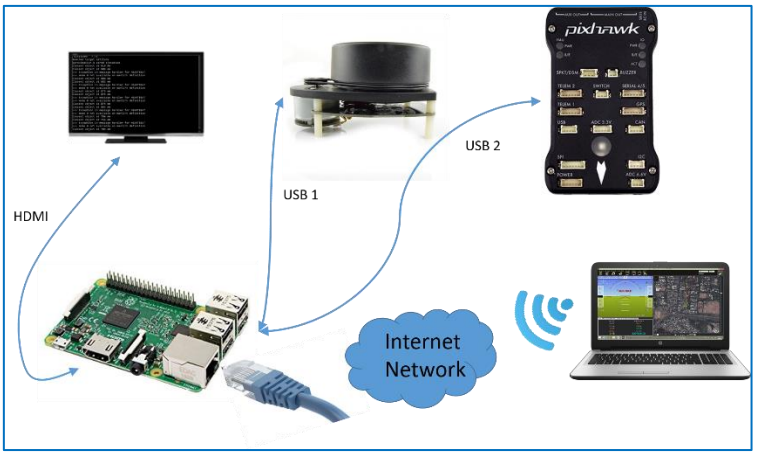

Figura 19. Interconexión de elementos del sistema con HW final

En la Raspberry Pi 3 se ha instalado una imagen de “Raspbian Jessie with Pixel”, la última versión de Raspbian. Se han instalado los paquetes necesarios para poder utilizar python, dronekit y MAVProxy, entre otros. A través de USB conectamos el Lídar y el Pixhawk. También utilizamos una pantalla conectada a través de HDMI para poder visualizar la ejecución del código de nuestro programa. Finalmente, conectamos la Raspberry Pi a Internet a través del puerto Ethernet.

La Raspberry Pi es la encargada de alojar el programa principal de la misión y de estar en constante comunicación con la placa controladora de vuelo dándole órdenes sobre la siguiente acción a realizar. Utilizamos el software MAVProxy para enviar a través de Internet los datos recibidos de Pixhawk. De esta manera podremos visualizarlos en otro dispositivo conectado a la misma red, en este caso un ordenador portátil con Windows 10, 64 bits, utilizando Mission Planner.

En este caso se pretende comprobar que la comunicación entre los dos elementos principales del sistema es correcta y que el programa principal se ejecuta correctamente en la Raspberry Pi. Los resultados de esta prueba son satisfactorios. Los datos son recibidos en el ordenador portátil y se visualizan en MP. El programa principal se ejecuta correctamente en Raspberry Pi.

Como ya utilizamos todos los elementos definitivos y no hay nada simulado se hace difícil comprobar el funcionamiento de las maniobras de despegue o de evitar colisión. Por ello, procedemos al ensamblaje de todo el sistema dentro de una plataforma real UAV para poder probar el sistema en movimiento.

6.4. Ensamblaje y prueba final

Se ha testeado el sistema FPV con una cámara GoPro y una CCD de baja calidad. Se comprueba que la calidad de la segunda no es suficiente para este tipo de operativas y se requerirá una calidad mínima de FULL HD.

Por motivos de falta de tiempo y de medios no se ha podido realizar un vuelo de prueba. Este no es un hecho trivial. Para poder volar bajo la legislación actual es necesario disponer de un espacio habilitado para el vuelo de UAVs, así como de un piloto con el certificado oficial para poder pilotar drones. Dado la no disponibilidad de ningún piloto en el tiempo de que se disponía por parte de HEMAV, se han realizado pruebas en tierra.

Para el ensamblaje final se ha utilizado un chasis DJI F550, 4 ESC TMOTOR de 30 A y motores T-motor. La PDB está integrada en el chasis. Aprovechamos la estructura de este chasis para colocar los elementos principales a 3 alturas como se muestra en la Figura 20. En la tapa superior colocamos el lídar de manera que ningún elemento interfiera en su campo de detección. Colocamos también el GPS/Compás lo más separado posible del lídar para evitar interferencias. En el piso intermedio se ha colocado la controladora de vuelo. En el piso inferior situamos la Raspberry Pi.

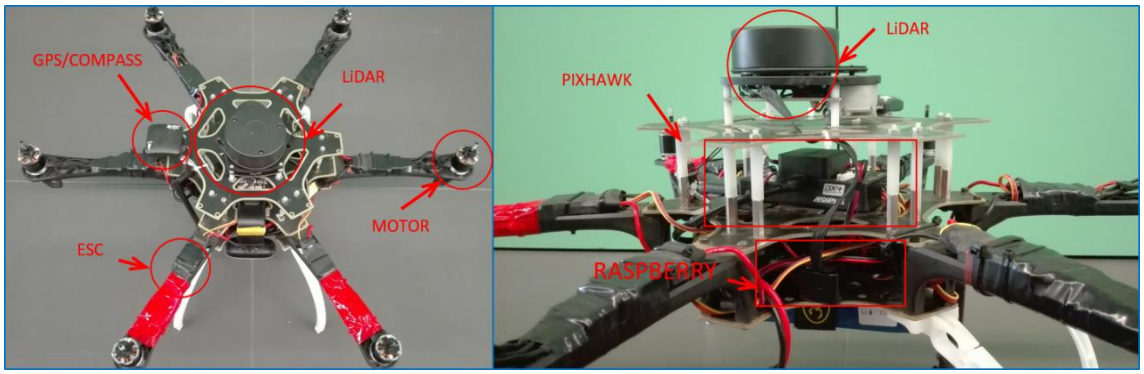

Figura 20. Colocación de todos los elementos en integración final

Una vez montados todos los elementos en el UAV, podremos visualizar en Mission Planner la información proporcionada por la controladora de vuelo en un dispositivo externo gracias a un emisor de telemetría conectado a la Pixhawk y un receptor conectado al dispositivo. Se utiliza como dispositivo externo un ordenador portátil igual que en la prueba anterior. Sin embargo, en este caso no utilizamos una red de Internet sino una transmisión por radiofrecuencia a 433 MHz.

Se han realizado pruebas en Tierra para comprobar el correcto funcionamiento del sistema completo. La forma de comprobarlo ha sido observar la línea negra que se muestra en Mission Planner saliendo del UAV, que indica la dirección que pretende tomar el dron en cada momento.

7. Conclusiones y trabajos futuros

Las empresas de limpieza de superficies verticales necesitan un avance tecnológico que les permita optimizar sus trabajos tanto a nivel de tiempo como de coste. Un UAV puede ser la solución a esta demanda.

Se supera el estado del arte actual. Gracias a la adaptabilidad del programa al tipo de edificio a limpiar, se pueden recorrer grandes superficies antes de devolver el dron a tierra.

Tras estudiar las diferentes plataformas disponibles se observa que Pixhawk es actualmente la más utilizada gracias a su facilidad de uso e integración con el resto de componentes del sistema. Se vuelve a demostrar, que Raspberry Pi es una herramienta muy potente y versátil a la hora de desarrollar todo tipo de proyectos de ingeniería.

Se ha confirmado la posibilidad de dotar de navegación autónoma a drones así como dotarlos de funcionalidades “sense and avoid” con HW de bajo coste. Se ha comprobado que con la solución diseñada es posible realizar este tipo de operativas de forma segura, teniendo en cuenta que, como en toda máquina, siempre existe un porcentaje de impredictibilidad.

Se ha verificado a través de los diferentes test, el correcto funcionamiento de cada uno de los elementos que componen el sistema tanto SW como HW (diseño de ruta, funcionamiento de lídar, integración en Raspberry Pi / Pixhawk, integración en plataforma UAV)

El hecho de automatizar el proceso y asegurar el correcto funcionamiento puede provocar un avance en la legislación actual. Se substituye el error humano por un pequeño error de la máquina. Podría llegar a ser usado por cualquier persona sin conocimientos de pilotaje dron.

Como trabajos futuros se plantea realizar vuelos de prueba para poder calibrar el software, tanto en los límites de distancias definidos, como en los tiempos de ejecución para conseguir movimientos más precisos que aseguren completamente el desarrollo y seguridad de la misión. También se deben probar sistemas digitales de transmisión de vídeo, dada que la latencia producida no es crítica. También será necesario estudiar los efectos y consecuencias del uso de drones en entornos urbanos, tanto por los efectos que estos puedan causar en la sociedad como por las posibles interferencias (por ejemplo, electromagnéticas) que éstos puedan recibir.

8. Costes

El producto que se presenta en este trabajo es una solución software basado en HW existente. El HW utilizado, sirve para testear esta solución, sin embargo no se trata de un producto definitivo. Por ello no se presentan costes.

Se han dedicado 300 horas a esta tesis a un coste de 11 € por hora, hace un total de 3300 €.

Todo el software utilizado es gratuito.

Bibliografía

[1] "RPA, UAV, RPAS, UAS y drones: qué diferencias hay entre ellos | ToDrone." [Online]. Available: http://www.todrone.com/diferencias-hay-entre-rpa-uav-rpas-uas-dron/. [Accessed: 06-May-2017].
[2] "Cleaning the World! - Cleandrone." [Online]. Available: http://www.cleandrone.com/. [Accessed: 10-May-2017].
[3] "Window Cleaning Drones." [Online]. Available: http://www.window-cleaning-drones.com/. [Accessed: 27-Jun-2017].
[4] "ArduPilot :: About." [Online]. Available: http://ardupilot.org/about. [Accessed: 05-Feb-2017].
[5] "ArduPilot Mega - Home." [Online]. Available: http://www.ardupilot.co.uk/. [Accessed: 04-May-2017].
[6] "Pixhawk Autopilot - Pixhawk Flight Controller Hardware Project." [Online]. Available: https://pixhawk.org/modules/pixhawk. [Accessed: 28-Apr-2017].
[7] "DJI - Líder Mundial en Drones/Cuatricópteros con cámara para Fotografía Aérea." [Online]. Available: https://www.dji.com/es/naza-m-v2. [Accessed: 29-Apr-2017].
[8] "PXFmini: cómo crear un dron autónomo con RaspBerry Pi 3 | ToDrone." [Online]. Available: http://www.todrone.com/pxfmini-como-crear-dron-raspberry-pi-3/. [Accessed: 20-May-2017].
[9] "First flight with the Raspberry Pi 3 - DIY Drones." [Online]. Available: http://diydrones.com/profiles/blogs/first-flight-with-the-raspberry-pi-3. [Accessed: 04-Jun-2017].
[10] "Companion Computers — Dev documentation." [Online]. Available: http://ardupilot.org/dev/docs/companion-computers.html. [Accessed: 04-Jun-2017].
[11] "El módulo Intel® Edison | Internet de las cosas | Software Intel®." [Online]. Available: https://software.intel.com/es-es/iot/hardware/edison. [Accessed: 01-Jun-2017].
[12] "Raspberry Pi 3 now on sale!" [Online]. Available: https://www.raspberrypi.org/blog/raspberry-pi-3-on-sale/. [Accessed: 01-Jun-2017].
[13] "MAVLink Micro Air Vehicle Communication Protocol - QGroundControl GCS." [Online]. Available: http://qgroundcontrol.org/mavlink/start. [Accessed: 04-May-2017].
[14] "MAVProxy on Windows 7 — Dev documentation." [Online]. Available: http://ardupilot.org/dev/docs/mavproxy-on-windows-7.html. [Accessed: 01-May-2017].
[15] "SITL Simulator (Software in the Loop) — Dev documentation." [Online]. Available: http://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html. [Accessed: 04-May-2017].
[16] "Sensor de ultrasonidos | Wiki de Robótica." [Online]. Available: http://wiki.robotica.webs.upv.es/wiki-de-robotica/sensores/sensores-proximidad/sensor-de-ultrasonidos/. [Accessed: 25-May-2017].
[17] "drones: una navegación más segura y autónoma con LIDAR." [Online]. Available: http://www.dronepedia.es/blog/drones-una-navegacion-mas-segura-autonoma-lidar. [Accessed: 08-May-2017].
[18] "RPLIDAR 360° Laser Scanner - RobotShop." [Online]. Available:

http://www.robotshop.com/en/rplidar-360-laser-scanner.html. [Accessed: 15-May-2017].
[19] “FPV Camera Latency Testing - Oscar Liang.” [Online]. Available: https://oscarliang.com/fpv-camera-latency/. [Accessed: 30-May-2017].
[20] “Receptor Video Inalambrico FPV 5.8Ghz 40Ch.” [Online]. Available: https://www.asturmodel.es/p-341/Multirotores-y-Accesorios/Sistemas-FPV--Micro-Drones-Carreras/Mini-Receptor-FPV-UFO-58Ghz-40Ch---Raceband. [Accessed: 15-Jun-2017].

Apéndices

1. Ejecución de programa en PC con SITL

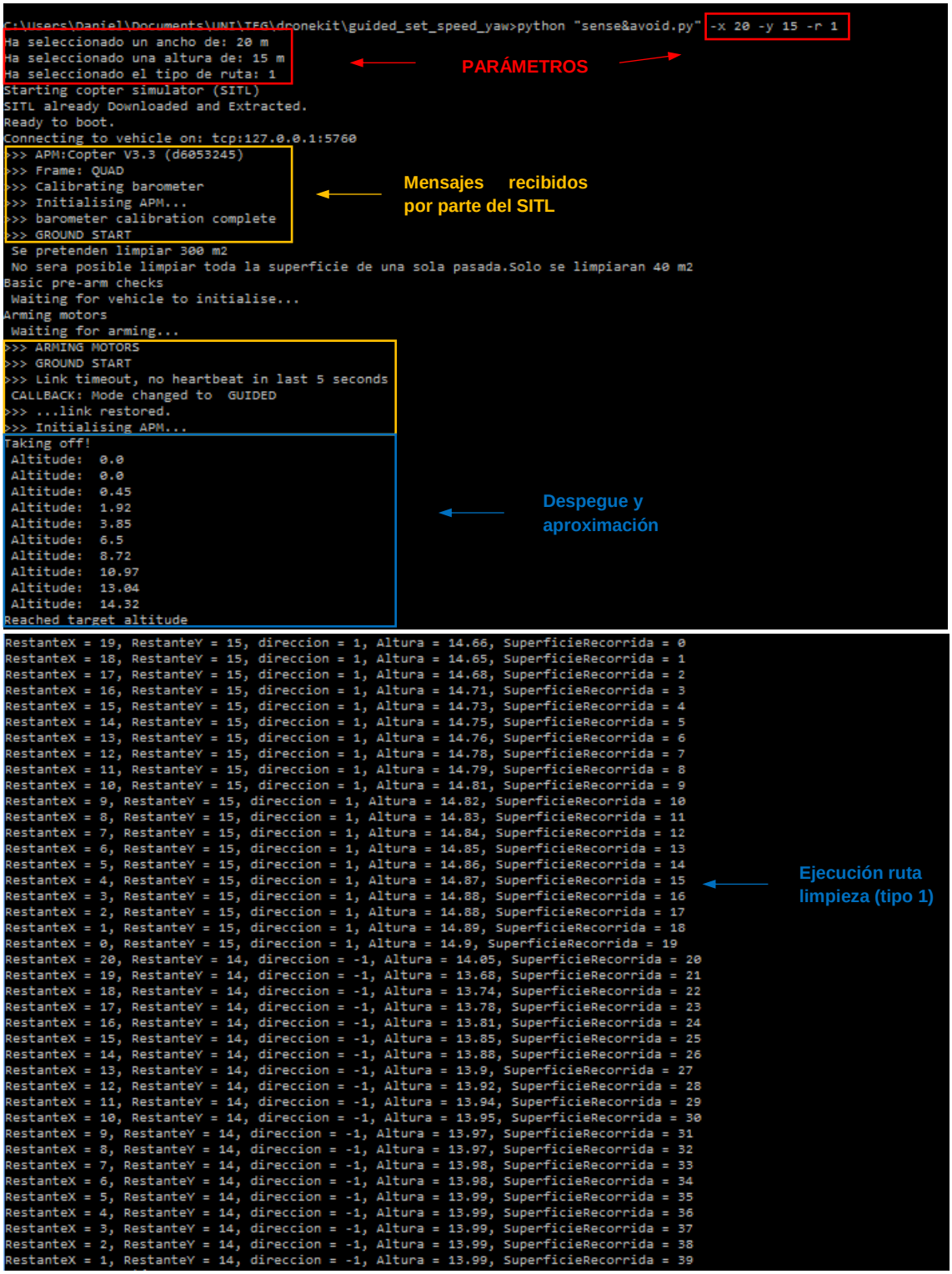

Figura AP 1. Despegue y ruta de limpieza en simulación con SITL

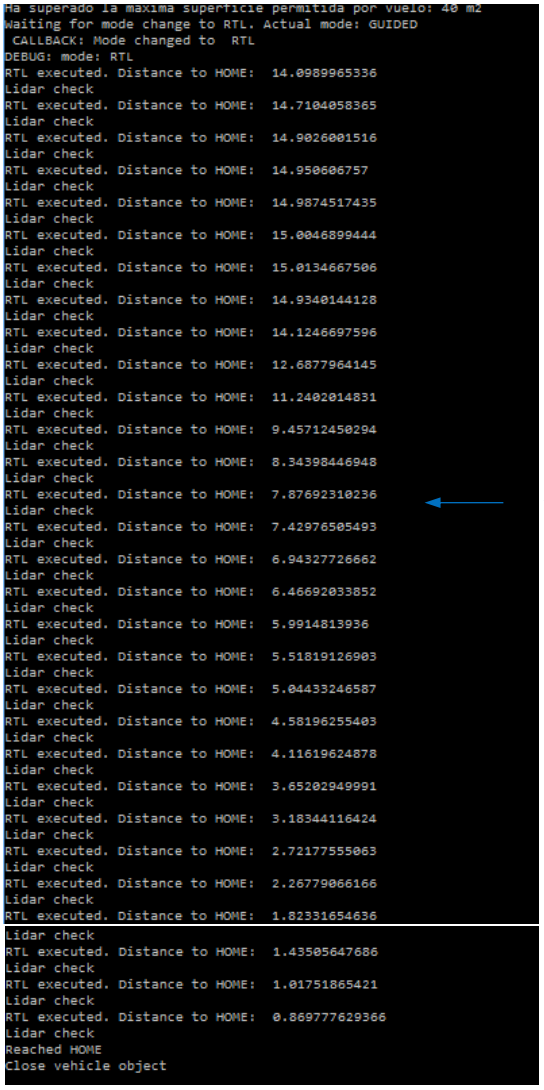

Figura AP 2. Return to launch en simulación con SITL

Glosario

BEC Battery Eliminator Circuit

ESC Electronic Speed Controller

FPV First Person View

GCS Ground Control Station

IMU Inertial Measurement Unit

PDB Power Distribution Board

RPAS Remotely Piloted Aircraft System

RTF Ready to Fly

RTL Return to Launch

SITL Software in the Loop

SO Sistema Operativo

SoC System on Chip

UAS Unmaned Aerial System

UAV Unmaned Aerial Vehicle