Trabajo Fin de Máster
Ingeniería Industrial

Diseño e implementación de un sistema para la gestión de una flota de drones para la inspección de plantas fotovoltaicas

Autor: Alejandro Castillejo Calle

Tutores: Jesús Iván Maza Alcañiz y Aníbal Ollero Baturone

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

Sevilla, 2019

Trabajo Fin de Máster
Ingeniería Industrial

Diseño e implementación de un sistema para la gestión de una flota de drones para la inspección de plantas fotovoltaicas

Autor:
Alejandro Castillejo Calle

Tutores:
Jesús Iván Maza Alcañiz
Profesor Titular de Universidad
Aníbal Ollero Baturone
Catedrático de Universidad

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2019

Trabajo Fin de Máster: Diseño e implementación de un sistema para la gestión de una flota de drones para la inspección de plantas fotovoltaicas

Autor: Alejandro Castillejo Calle

Tutores: Jesús Iván Maza Alcañiz y
Aníbal Ollero Baturone

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2019

El Secretario del Tribunal

Resumen

Este documento presenta una solución diseñada para inspección de plantas solares fotovoltáicas con una flota de multiples UAVs (vehiculos aéreos no tripulados). Se detalla el desarrollo llevado a cabo a nivel hardware y software.

La parte software se divide en dos segmentos: terrestre y aéreo. El segmento terrestre consiste en una aplicación basada en ROS (Robot Operating System) que correrá en la GCS (estación de control de tierra). Esta se encargá de generar una misión de inspección para los distintos UAVs de la flota. Integra una interfaz gráfica de usuario que permite el control y monitorización de la misión en tiempo real desde la propia GCS, además de una API Rest que hace posible el control y monitorización de la misión también de forma remota.

El segmento aéreo contiene el software, también basado en ROS, que correrá en los ordenadores de a bordo de cada uno de los drones. Contiene varios nodos encargados del desarrollo de las distintas fases de la misión, el control de todos los sensores a bordo del dron y la navegación autónoma del vehiculo. El sistema se ha diseñado de forma que sea compatible con los autopilotos de DJI y con aquellos basados en PX4 o Ardupilot.

El documento concluye con los resultados experimentales obtenidos en varias pruebas en simulación y sobre una planta real ubicada junto a la localidad de Utrera (en España).

Abstract

This document presents a solution designed for the inspection of photovoltaic solar plants with a fleet of multiple UAVs (Unmaned Aerial Vehicles). It details the developed hardware and software.

The software is divided into two segments: ground and aerial. The ground segment consists of an application based on ROS (Robot Operating System) that runs in the GCS (Ground Control Station). This will generate an inspection mission for the different UAVs in the fleet. It integrates a graphical user interface that allows the control and monitoring of the mission in real time from the GCS itself, in addition to an API Rest that makes it possible to control and monitor the mission also remotely.

The aerial segment contains the software, also based on ROS, which will run on the on-board computers of each of the drones. It contains several nodes in charge of the development of the different phases of the mission, the control of all the sensors on board the drone and the autonomous navigation of the vehicle. The system has been designed to be compatible with DJI autopilots and those based on PX4 or Ardupilot.

The document concludes with the experimental results obtained in several simulation tests and also in a real plant located next to the town of Utrera (in Spain).

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

1.1 Motivación

El uso de los UAVs para diferentes aplicaciones, tales como vigilancia [2]-[4], transporte [5] e inspección de puentes [6],[7] se ha incrementado significativamente en los últimos años.

Los vehículos aéreos no tripulados se están convirtiendo en un instrumento fiable y útil en los servicios de monitorización y diagnóstico en el campo de la energía [8]. La automatización de determinadas actividades de monitorización está pensada como un elemento esencial para ser competitivos en este mercado. Muchos de los procedimientos manuales y dispersivos que se aplican hoy en día, pronto serán superados por procedimientos centrados en los UAVs. Esto es particularmente cierto en lo que respecta a la inspección de las centrales fotovoltaicas

Recientemente, se han usado vehículos aéreos no tripulados equipados con imágenes térmicas/visuales para la identificación de puntos calientes anormales en los módulos fotovoltaicos. La ventaja de utilizar un sistema de este tipo se deriva de su eficacia en la detección de fallos y el diagnóstico de módulos fotovoltaicos para la inspección de plantas a gran escala. Este enfoque está presente en [9], donde se utiliza una referencia para asociar la imagen térmica/visual terrestre obtenida en distintas posiciones de la planta fotovoltaica con datos geográficos para generar en última instancia "mapas de inspección" estáticos en forma de verdaderos mosaicos ortofotográficos. Otro enfoque para la detección y el análisis automático de módulos fotovoltaicos en imágenes aéreas infrarrojas también se propone en [10]. Además, las plantas fotovoltaicas de la vida real suelen requerir un equipo de varias UAVs para obtener imágenes térmicas en condiciones de irradiación solar comparables.

Este documento se desarrolla a raíz del trabajo de investigación realizado para la Línea 1.1 "Nuevas técnicas de cooperación de UAVs para la inspección desatendida" del proyecto INSPECTOR.

Uno de los objetivos del proyecto INSPECTOR es explotar la capacidad de cooperación entre varios vehículos aéreos no tripulados para la inspección simultánea de una zona. La línea sobre la que se desarrolla este trabajo se centra en la inspección de plantas solares fotovoltaicas.

1.2 Objetivos

El objetivo de este proyecto es explotar la capacidad de cooperación entre varios vehículos aéreos no tripulados para la inspección de plantas solares fotovoltaicas. El enfoque completo tiene que ser validado en experimentos de campo con diferentes multicópteros autónomos equipados con dispositivos heterogéneos como cámaras visuales, infrarrojas, e hiperespectrales.

Por lo tanto, antes de proceder con la implementación del software de la plataforma para el proceso de validación, se requiere un diseño que pueda abordar los diferentes objetivos mencionados del proyecto.

Este documento presenta la arquitectura diseñada para la inspección de plantas solares fotovoltaicas mediante vehículos aéreos no tripulados (UAVs).

La arquitectura descrita en este documento está dotada de diferentes módulos que resuelven los problemas habituales que surgen durante la ejecución de misiones, como la asignación de tareas, la resolución de conflictos y la descomposición de tareas complejas en otras más simples y ejecutables por cada UAV. El enfoque tiene que satisfacer dos requisitos principales: robustez para la operación en escenarios de inspección y fácil integración de diferentes vehículos autónomos de distintos fabricantes.

Se presenta además el diseño software desarrollado para la primera fase del proyecto, cuyo objetivo es generar las misiones de vuelo correspondientes para cada uno de los UAVs. Este software debe ser capaz de

obtener e interpretar la información de la misión de un archivo (perímetro de la planta definido por coordenadas GPS, requisitos de monitorización, parámetros de vuelo e información de sensores), y realizar las siguientes tareas:

- Planificación de la misión.
- Generación de planes de vuelo para cada UAV.

1.3 Produccion científica

A raíz de este trabajo se ha presentado el siguiente articulo en RED-UAS 2019:

"A Multi-UAS System for the Inspection of Photovoltaic Plants Based on the ROS-MAGNA Framework", by Alejandro Castillejo-Calle, José Andrés Millán Romera, Hector Perez-Leon, Jose L. Andrade-Pineda, Ivan Maza, Anibal Ollero

Tambien se han hecho aportaciones en los siguientes:

"ROS-MAGNA, a ROS-based framework for the definitionand management of multi-UAS cooperative missions", By J. A. Millan-Romera, H. Perez-Leon, A. Castillejo-Calle, I. Maza, andA. Ollero, inProc. ofthe International Conference on Unmanned Aircraft Systems (ICUAS).IEEE, June 2019, pp. 1–10.

H. Perez-Leon, J. J. Acevedo, J. A. Millan-Romera, A. Castillejo-Calle, I. Maza, and A.Ollero, "An aerial robot path follower based on the 'carrot chasing' algorithm", By H. Perez-Leon, J. J. Acevedo, J. A. Millan-Romera, A. Castillejo-Calle, I. Maza, and A.Ollero, inFourth Iberian Robotics Conference,2019

1.4 Organiacion del documento

El resto del documento se organiza de la siguiente manera:

En el capítulo 2 describe el procedimiento seguido para generar de forma automática los planes de vuelo, en base a una serie de parámetros que definen la misión y al numero de UAVs conectados con la GCS en ese momento.

En el 3, se trata con más detalle el desarrollo a nivel software que hace posible la planificación y el desarrollo de la misión. Se introduce la capa ADL (Automatic Decision Layer), basada en una máquina de estado que se encarga de desarrollar las distintas fases de la misión.

El cuarto capitulo se centra en la integración de los sensores de a bordo, lo cual se hace mediante la capa software PAL (Payload Abstraction Layer). Esta se encarga del control de los sensores, el almacenamiento de los datos generados por estos y la subida de estos datos a la GCS.

El capitulo 5 muestra la GUI (interfaz grafica de usuario) y la API Rest desarrolladas. Estas permiten el control y monitorización de la misión en tiempo real desde la GCS y de forma remota.

Por último, en los capítulos 6 y 7 se muestran los resultados experimentales obtenitos, y las conclusiones y desarrollos futuros.

2 GENERACIÓN AUTOMATICA DE PLANES

En este capitulo se detalla la solución diseñada para la planificación de la misión de inspección, en base a una serie de parámetros que la definen, y la generación de los planes de vuelo para cada drone.

Este software se ha desarrollado para la GCS utilizando el entorno ROS. Se organiza en varios archivos, escritos en código C++, encargados de la interpretación de la información del archivo de la misión, la planificación de las misiones, la comunicación con los UAVs y la conexión con una interfaz gráfica de usuario (GUI). Este software recibe el archivo de la misión en formato JSON y genera los planes de vuelo para los distintos UAVs en base a una serie de parámetros contenidos en este archivo y los envía a los mismos. Tanto el sofware desarrollado hasta el momento, como las futuras versiones, se pueden descargar desde un enlace en la web interna del proyecto.

2.1 Análisis y definición de requisitos

2.1.1 Toma de decisiones en sistemas complejos

2.1.1.1 Componentes Clave

La "Decisión", relativa a los sistemas inteligentes, se refiere a diferentes mecanismos centrados en el procesamiento autónomo y coherente de una misión (desde simples solicitudes hasta complejas secuencias de tareas de alto nivel), ya sea dentro de un sistema centralizado o distribuido. Se podrían destacar cuatro mecanismos principales:

La asignación de tareas que surge en sistemas multiagente (es decir, compuestos por varias entidades), donde hay que decidir de forma autónoma qué agente debe realizar cada tarea de aquellas que componen una misión. Esto requiere la capacidad de evaluar el interés de asignar una determinada tarea a un determinado agente. Esta operación es especialmente difícil cuando la decision tiene que hacerse teniendo en cuenta los planes individuales actuales de los agentes, así como las tareas que quedan por asignar. Realizar dicha asignación dentro de un sistema centralizado de toma de decisiones requiere disponer de toda información relevante dentro de este sistema central: los modelos, el plan de cada agente y el estado actual de ejecución de las tareas de cada uno de ellos.

La planificación de tareas es elemento central dentro de la toma automática de decisiones. Su objetivo es construir una secuencia de tareas a realizar, con el fin de lograr una misión determinada. Esta misión también podría ser declarada como:

- Una definición simple del estado del mundo que debe alcanzarse después de que se hayan realizado un cierto número de tareas,
- O como una secuencia compleja de instrucciones o estados intermedios del mundo, que el sistema debe alcanzar siguiendo un orden dado (posiblemente parcial).

Dentro de un sistema de agente único, la planificación ya es una pieza compleja de procesamiento y cálculo, ya que la complejidad de encontrar un plan óptimo es generalmente muy difícil en términos computacionales. Esto es especialmente cierto cuando se consideran ventanas de tiempo, o incertidumbres, en el cálculo de los planes.

La coordinación es un proceso que surge dentro de un sistema si determinados recursos (internos o externos) son requeridos simultáneamente por varios componentes de éste. En el caso de un sistema multi-robot, un problema clásico de coordinación es el de compartir el espacio entre los diferentes robots para que cada uno de ellos pueda realizar su plan de forma segura y coherente con los planes de los demás robots.

Por ejemplo, si las misiones en el proyecto INSPECTOR requieren la cobertura completa de una zona

determinada, esta región debería dividirse entre los UAVs disponibles equipados con sensores de acuerdo con sus capacidades relativas (velocidad máxima, autonomía, campo de visión de los sensores, etc.) de tal modo que no haya conflictos entre ellos.

La coordinación del uso compartido del espacio debe llevarse a cabo de forma continua o de manera iterativa durante la ejecución de una misión, ya que determinadas contingencias pueden requerir revisar y actualizar los planes en cualquier momento.

La supervisión se ocupa del control de la ejecución de las tareas de varias maneras:

- Un primer problema es simplemente mantener al sistema al tanto de la evolución del procesamiento de las tareas durante la ejecución de las mismas.
- El segundo es detectar los posibles fallos en las tareas y (si es posible) reaccionar a tales eventos de una manera adecuada.

Por otro lado, la usabilidad de la plataforma puede ser aumentada también si la arquitectura permite diferentes niveles de autonomía. Esto permitiría al usuario decidir si un determinado proceso o acción se realiza de forma manual o autónoma.

Finalmente, se adoptarán los siguientes supuestos en nuestro diseño:

- El operador introduce una misión a una Estación de Control Terrestre (GCS) a través de una interfaz Hombre-Máquina (HMI). Dentro de una misión, el usuario puede designar UAVs para llevar a cabo acciones específicas o partes de la misión.
- Cada UAV debe ser capaz de tomar una serie de decisiones acerca de su propia misión y de interactuar con otros miembros de la plataforma en términos de coordinación si la misión global lo requiere.
- Cada UAV tiene su propia arquitectura interna del sistema, que es transparente para el resto del sistema.

En la siguiente sección, se discute el compromiso entre la toma de decisiones centralizada y la descentralizada, en la medida en que ésta es una cuestión clave en cualquier sistema multirobot.

2.1.2 Toma de decisiones Centralizada/Descentralizada

Una configuración de toma de decisiones centralizada (con una supervisión distribuida mínima) es compatible (por lo menos) e incluso complementaria con una configuración dotada de capacidades de decisión completamente distribuidas. El subapartado anterior enumeraba los principales componentes de la "Decisión": asignación, planificación, coordinación y supervisión. Cualquiera de ellos puede ser desarrollado dentro de un componente central de decisión o entre varios componentes de manera distribuida. Sin embargo, se deben considerar varias cuestiones:

El alcance y la accesibilidad del conocimiento son requisitos preliminares, para permitir la toma de decisiones dentro de un componente central y para asegurar permanentemente la disponibilidad de conocimientos actualizados (relevantes) dentro de este componente central. Este es un requisito importante, ya que requiere la centralización de cualquier conocimiento relacionado con la toma de decisiones de cualquier componente del sistema, y la actualización continua de los mismos, para poder llevar a cabo los procesos de toma de decisiones. Sin embargo, suponiendo que este requisito pueda cumplirse, esto brinda la oportunidad de adoptar decisiones con mayor conocimiento de causa y, por lo tanto, de gestionar las operaciones de la misión de manera más eficiente.

Potencia computacional y escalabilidad. En el caso de un sistema compuesto por varios robots, la cantidad de datos a procesar es bastante grande: el procesamiento de este conocimiento de forma centralizada requiere medios computacionales obviamente potentes.

Por lo tanto, un enfoque centralizado será relevante si:

- Las capacidades computacionales son compatibles con la cantidad de información a procesar.
- El intercambio de datos cumple tanto los requisitos de rapidez (datos actualizados) como de expresividad (calidad de la información que permite una toma de decisiones bien informada).

Teniendo en cuenta las especificaciones del proyecto INSPECTOR, la mayoría de las decisiones serán tomadas de forma centralizada en la GCS.

2.1.3 Arquitectura de la plataforma Multi-UAV

La Arquitectura que se propone deriva de la consideración de dos requerimientos básicos: (i) integración sencilla de UAVs de distintos fabricantes y, (ii) la robustez en las operaciones de inspección en los diferentes escenarios previstos.

Como se ha comentado anteriormente se ha adoptado un enfoque de decisiones centralizado en una GCS, que por razones de usabilidad contará con una interfaz gráfica de usuario (GUI).

La flexibilidad en la incorporación de UAVs se apoya en una abstracción del hardware de un UAV concreto, que permitirá su integración en la plataforma con el único requisito de que el UAV sea capaz de ejecutar un conjunto mínimo de tareas elementales presente en todos los autopilotos actuales.

Como se puede ver en la Figura 1, en cada UAV hay dos niveles principales: el ADL (Automatic Decision Level) y el UAL1 (UAV Abstraction Layer), esta última capa desarrollada por el grupo de investigación GRVC2 (Grupo de Robótica, Visión y Control de la Universidad de Sevilla). El primer nivel aborda la toma de decisiones de alto nivel, mientras que el segundo está a cargo de las llamadas "tareas elementales". En la interfaz entre ambos niveles, la ADL envía solicitudes de tareas y recibe el estado de ejecución de cada tarea, así como el estado del UAV. Ambos niveles se ejecutarán a bordo de cada UAV y se han implementado en C++/Python bajo ROS (Robotic Operating System).

El software de la GCS permite al usuario especificar las tareas a ejecutar por la plataforma, así como monitorizar el estado de ejecución de las tareas y el estado de los diferentes componentes de la plataforma. Este software se ha implementado también en C++/Python sobre ROS. La GCS recibirá del operador de TSK remoto un archivo JSON con las especificaciones de la misión.

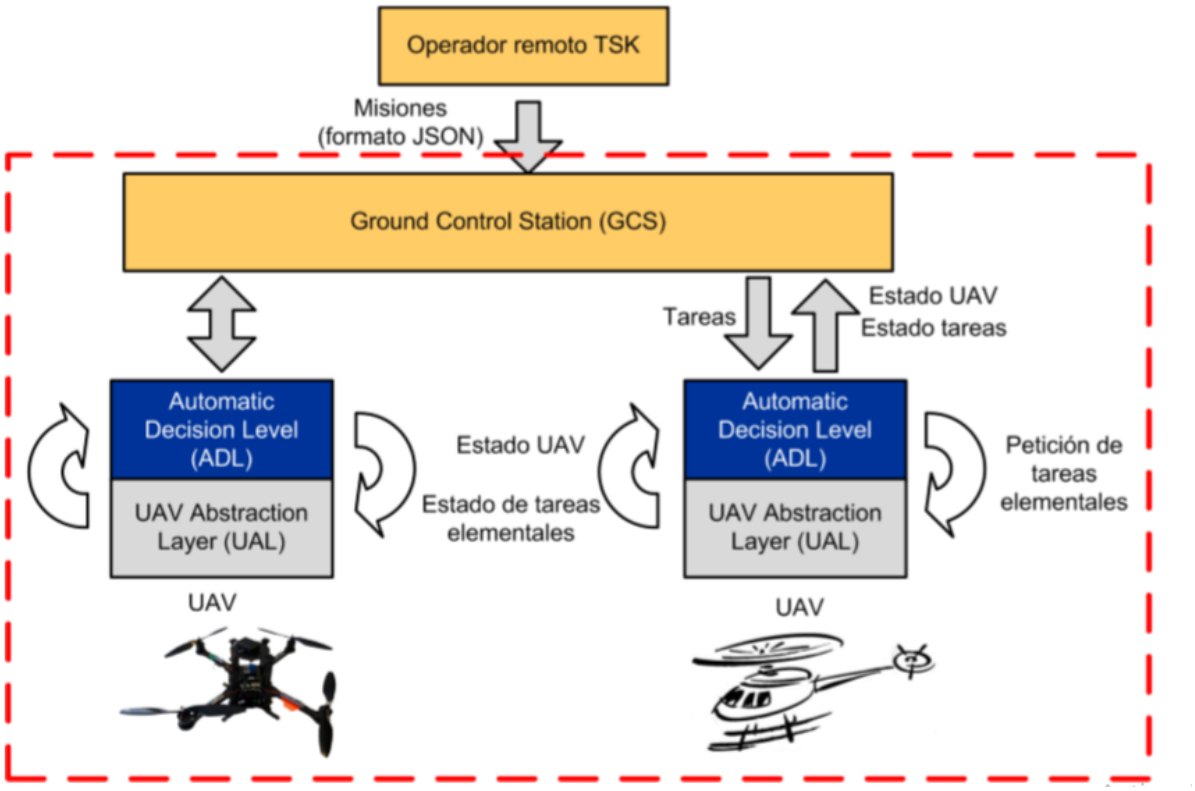

Figura 1. Arquitectura de la plataforma diseñada.

En relación a la capa UAL, su objetivo principal es simplificar el desarrollo y las pruebas de algoritmos de

alto nivel en robótica aérea, intentando estandarizar y simplificar las interfaces con los UAVs para poder utilizar modelos de autopiloto de distintos fabricantes. UAL puede funcionar con UAVs simulados o reales y proporciona llamadas para emitir comandos tales como despegue, aterrizaje y otros de control en posición y velocidad. Aunque la UAL está en continuo desarrollo, existe una versión estable disponible para el uso público y actualmente está siendo utilizada en varios proyectos de investigación europeos por diferentes entidades académicas e industriales.

En UAL se ha definido una interfaz común que recolecta la información y las funcionalidades más utilizadas habitualmente con UAVs:

- Realizar una maniobra de despegue a una altura determinada.
- Ir de la posición actual a un waypoint especificado en un sistema de coordenadas globales o en coordenadas geográficas.
- Ajuste de las velocidades lineales globales y de la tasa de cambio del ángulo de guiñada.
- Aterrizar en la posición actual.
- Volver del modo de vuelo manual al autónomo.
- Ajustar la posición de inicio a la posición actual.
- Obtener la última estimación de la posición del UAV.
- Obtener la última estimación de velocidad del UAV.
- Obtener la última estimación de la matriz de transformación del UAV.

La arquitectura interna de la UAL se puede ver en la Figura 2. La clase denominada Backend establece una interfaz común con la UAL. Con el fin de dar soporte a un autopiloto en particular, se debe desarrollar un back-end específico para ese autopiloto. Este back-end se comunica con el autopiloto y se encarga de las tareas elementales, ofreciendo una interfaz común en el lado del ADL. Por ejemplo, cualquier autopiloto que utilice MAVLink como comunicación (por ejemplo, PX4 y Ardupilot), tiene actualmente soporte a través de un back-end de MAVROS. Además, en el marco del proyecto INSPECTOR se ha desarrollado también un backend basado en el SDK (Software Development Kit) de DJI para dar soporte a los autopilotos de este fabricante.

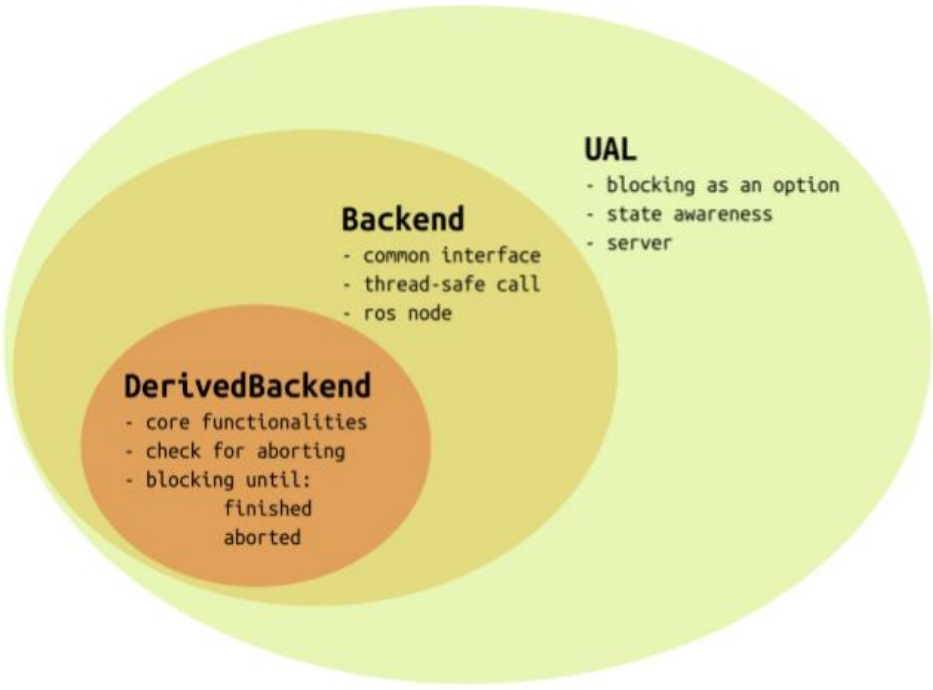

Figura 2. Esquema con las diferentes capas del UAL. Las capas UAL y Backend son comunes, mientras que DerivedBackend es particular del protocolo/autopiloto empleado.

Hay un total de tres back-ends soportados actualmente por la UAL (ver Figura 3). El primero para MAVROS (la adaptación ROS del protocolo MAVLink), que es una forma muy extendida de comunicación con los pilotos automáticos. En segundo lugar, tenemos un back-end "ligero" sólo para simulación, y por último el back-end para DJI desarrollado en el marco del proyecto INSPECTOR.

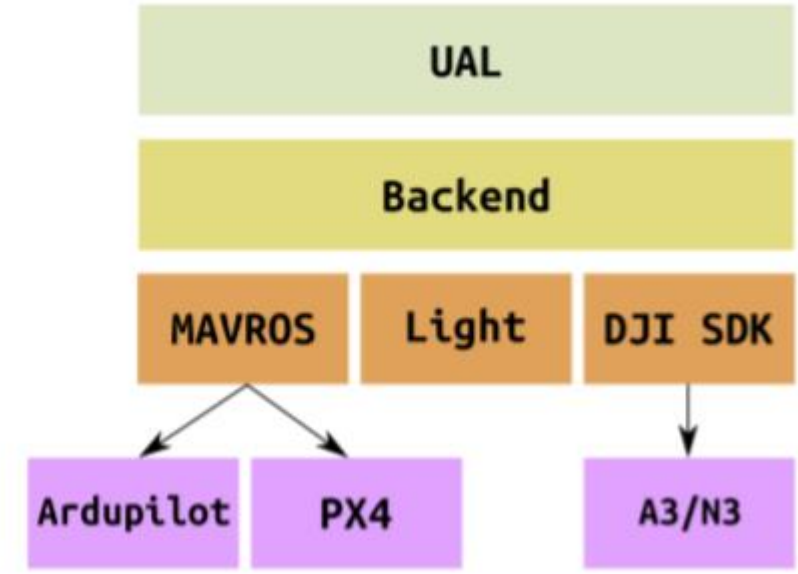

Figura 3. Backends implementados actualmente en UAL.

En lo que sigue, asumiremos que un plan de vuelo para un UAV estará compuesto de una serie de tareas elementales al nivel de la UAL. La **¡Error! No se encuentra el origen de la referencia.**muestra la lista de tareas elementales soportadas en la capa UAL.

takeOff	Realizar una maniobra de despegue a una altura determinada
goToWayPoint	Ir de la posición actual a un waypoint especificado en coordenadas globales
goToWayPointGeo	Ir de la posición actual a un waypoint especificado en coordenadas geográficas
setVelocity	Ajuste de las velocidades lineales globales y de la tasa de cambio del ángulo de guiñada
land	Aterrizar en la posición actual
recoverFromManual	Pasar del modo de vuelo manual a autónomo
setLocalPosition	Ajustar la posición de inicio a la posición actual

Tabla 1. Tareas elementales soportadas actualmente en la capa UAL.

2.2 Obtención de la información de la misión

La información y parámetros necesarios para la generación de la misión se encuentran en un archivo JSON con el siguiente contenido:

- Perímetro de la planta fotovoltaica (definido por las coordenadas GPS de los vértices del polígono que la delimita)
- Requisitos de monitorización y parámetros de vuelo:
    - Altitud de vuelo
    - Solapamiento longitudinal y vertical
    - Velocidad
    - Desviación
    - Variación de altura del terreno
- Información de los sensores

La lectura y obtención de datos del archivo JSON se realiza a través del objeto get_from_json.cpp haciendo uso de un parser fácilmente integrable mediante un solo archivo de cabecera. Este se encuentra ya incluido en el repositorio del programa, por lo que no será necesario la instalación de ningún software adicional para el tratamiento de archivos JSON.

2.3 Generación de planes de vuelo

A partir de las coordenadas del polígono y del resto de parámetros necesarios se obtiene la trayectoria necesaria para realizar, en forma de zig-zag, el barrido de la planta fotovoltaica completa.

Para el desarrollo de esta parte del código se han utilizado como referencia algunos algoritmos del software QgroundControl5 (open source), adaptándolos y añadiendo algunas funciones necesarias para cumplir con las necesidades del proyecto INSPECTOR.

Para optimizar la realización de la misión mediante un sistema multi-UAV se obtiene la longitud de esta trayectoria y se divide entre el número de UAVs, de forma que todos recorran la misma distancia .Luego se genera una lista de waypoints para cada UAV formada por:

- Punto de inicio de la misión.
- Puntos intermedios correspondientes a los vértices del patrón en zig-zag, donde el UAV tendrá que parar, cambiar de dirección y continuar con el siguiente segmento.
- Punto final de la misión.

De esta forma, suponiendo que se tienen n UAVs:

- El primer waypoint del UAV 1 se corresponderá es el punto en el que se inicia la trayectoria completa.
- El primer waypoint del UAV 2 coincidirá con el punto final de la misión del UAV 1, y así sucesivamente.
- El último waypoint del UAV n será el punto final de la trayectoria completa.

De este modo se evitan conflictos durante la realización de los barridos por parte de los distintos UAVs ya que el plan de barrido de cada UAV no se cruza con el de ningún otro. Sin embargo, podrían producirse

conflictos en las trayectorias hacia y desde las rutas de barrido (por ejemplo, cuando el UAV se dirige desde la base a su punto de inicio de misión o en caso de que tuviera que volver porque se agote su batería).

Para evitar esto se ha incluido un algoritmo que genera para cada UAV altitudes diferentes en esos momentos de la misión de forma que siempre estén por debajo o por encima de la altitud de vuelo de barrido (la misma para todos). Además, las altitudes estarán siempre por encima de la altitud mínima de vuelo permitida y guardarán entre ellas una distancia de seguridad. La distribución de dichas altitudes se irá alternando por debajo y por encima de la altitud de barrido. Si se llega a agotar todo el rango disponible entre la altitud de vuelo y la mínima permitida las posteriores altitudes se asignarán todas por encima. En la Figura 4 se puede ver un ejemplo de como sería esta distribución.

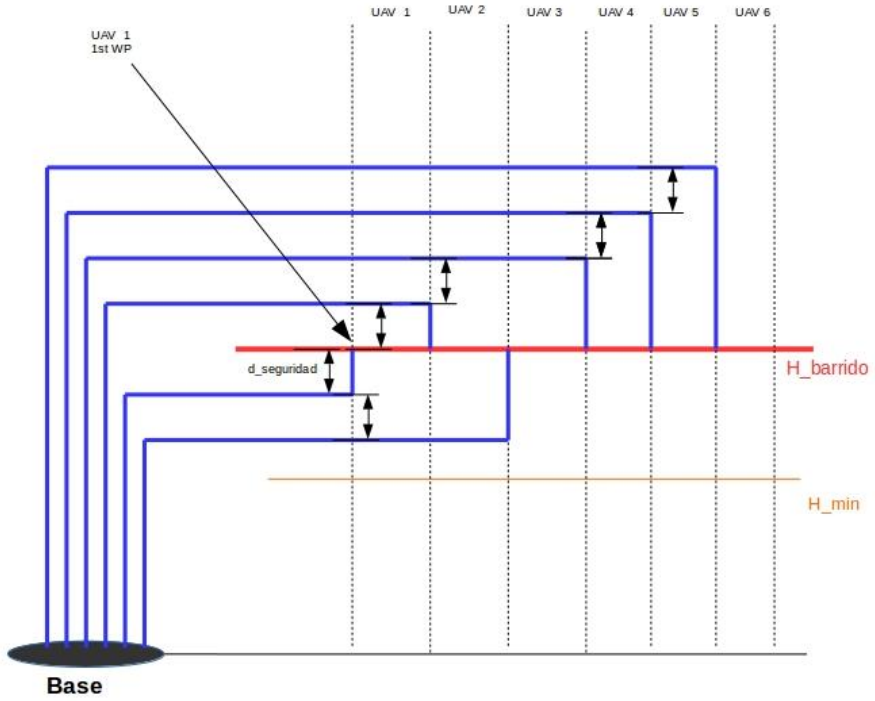

Figura 4. Ejemplo de la estrategia de asignación de altitudes de vuelo para un sistema multi-UAV formado por 6 aeronaves.

3 PLANIFICACION PARA INSPECCION MULTI-UAV DE PLANTAS FOTOVOLTAICAS

En este capitulo se trata la tarea de navegación autónoma de los UAVs para cubrir el área requerida, es decir, la ejecución de forma autónoma de las misiones de vuelo.

Se detalla la estrategia seguida para la planificación de las misiones, así como las implementaciones a nivel de software y hardware que se han llevado a cabo para hacer posible la navegación autónoma.

Esta parte del proyecto se basa en el desarrollo de la capa ADL (Automatic Decision Layer) que se describió anteriormente. Dicho desarrollo se ha basado en una maquina de estado programada utilizando la herramienta de ROS (Robotic Operating System) denominada Smach. Esta máquina de estado se encarga de llevar a cabo la navegación autónoma, y de las tareas de replanificación de la misión en caso de eventos tales como un nivel bajo de batería o la solicitud de abortar la misión de forma manual.

Se ha desarrollado también un sistema de vuelo AGL (Above Ground Level) debido al posible desnivel en las plantas fotovoltaicas entre las distintas zonas de paneles. Este sistema se ha basado en el uso de un altímetro laser.

3.1 Planificación de la misión y generación de planes de vuelo para cada UAV

Este software que se ha desarrollado para la GCS utiliza, también, el entorno ROS. Se organiza en varios archivos, escritos en código C++, encargados de la interpretación de la información del archivo de la misión, la planificación de las misiones, la comunicación con los UAVs y la conexión con una interfaz gráfica de usuario (GUI).

Este software recibe el archivo de la misión en formato JSON, genera los planes de vuelo para los distintos UAVs en base a una serie de parámetros contenidos en este archivo y los envía a los mismos.

3.1.1 Obtención de la información de la misión

La información y parámetros necesarios para la generación de la misión se encuentran en un archivo JSON con el siguiente contenido:

- Perímetro de la planta fotovoltaica (definido por las coordenadas GPS de los vértices del polígono que la delimita)
- Requisitos de monitorización y parámetros de vuelo
  - Altitud de vuelo
  - Solapamiento longitudinal y vertical
  - Desviación
  - Variación de altura del terreno
- Información de los sensores

La lectura y obtención de datos del archivo JSON se realiza a través del objeto *get_from_json.cpp* haciendo uso de un *parser* fácilmente integrable mediante un solo archivo de cabecera. Este se encuentra ya incluido en el repositorio del software, por lo que no será necesario la instalación de ningún programa adicional para el tratamiento de archivos JSON.

3.1.2 Generación de planes de vuelo

A partir de las coordenadas del polígono y del resto de parámetros se obtiene la trayectoria necesaria para realizar el barrido de la planta fotovoltaica completa usando un patrón de barrido en forma de zig-zag.

Para el desarrollo de esta parte del código se han utilizado como referencia algunos algoritmos del software QgroundControl (open source), adaptándolos y añadiendo algunas funciones necesarias para cumplir con las necesidades del proyecto. De esta manera, se genera una lista de waypoints para cada UAV formada por:

- Punto de inicio del barrido.
- Puntos intermedios correspondientes a los vértices del patrón en zig-zag, donde el UAV tendrá que parar, cambiar de dirección y continuar con el siguiente segmento.
- Punto final del barrido.

El plan de barrido de cada UAV se genera además de forma que no se cruza con el de ningún otro. Sin embargo, podrían producirse conflictos en las trayectorias hacia y desde las rutas de barrido (por ejemplo, cuando el UAV se dirige desde la base a su punto de inicio de barrido o en caso de que tuviera que volver porque se agote su batería).

Para evitar esto se ha incluido un algoritmo que genera para cada UAV altitudes diferentes en esos momentos de la misión de forma que siempre estén por debajo o por encima de la altitud de vuelo de barrido (la misma para todos). Además, las altitudes estarán siempre por encima de la altitud mínima de vuelo permitida y guardarán entre ellas una distancia de seguridad. La distribución de dichas altitudes se irá alternando por debajo y por encima de la altitud de barrido. Si se llega a agotar todo el rango disponible entre la altitud de vuelo y la mínima permitida las posteriores altitudes se asignarán todas por encima. En la Figura 4 se puede ver un ejemplo de como sería esta distribución.

En la Figura 5 se puede ver una simulación realizada usando la herramienta Rviz de ROS en una planta de ejemplo con forma rectangular utilizando tres UAVs.

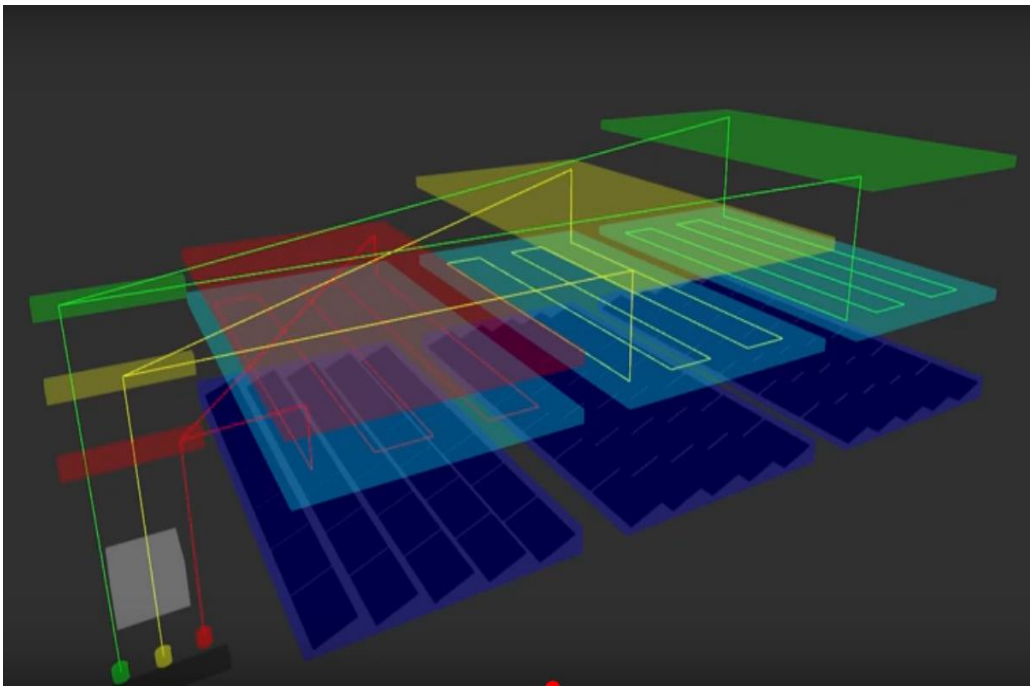

Figura 5. Visualización en 3D con la herramienta Rviz de ROS. Se ha simulado una misión con tres UAVs.

3.2 Automatic Decision Layer (ADL)

Como se ha mencionado en la introducción de este capitulo, en esta parte del proyecto se ha desarrollado la capa denominada ADL (Automatic Decision Layer). Esta se ejecutará en el ordenador de abordo de cada UAV (en nuestro caso será una NUC de Intel) y es la encargada del desarrollo de las distintas fases de la misión.

Por una parte, la ADL se comunica con la GCS, de la que recibe la información necesaria para la misión (waypoints, altitud de barrido y altitud de vuelo fuera del barrido) y las "órdenes" correspondientes (comenzar, pausar o reanudar la misión).

A su vez, la ADL se comunica con las capas UAL (UAV Abstraction Layer) y PAL (Payload Abstraction Layer), también instaladas en el ordenador de abordo. En la Figura 6 se puede ver un esquema de la arquitectura software y hardware del sistema. Como se había detallado en el anterior entregable, UAL se encarga del control del autopiloto A3 de DJI mediante una serie de funciones que permiten abstraerse de la interacción directa con el autopiloto. La capa PAL se encarga del control de las cámaras Sony y Wiris y de enviar en vivo las imágenes de la webcam. Esta se describirá con más detalle en el próximo capitulo.

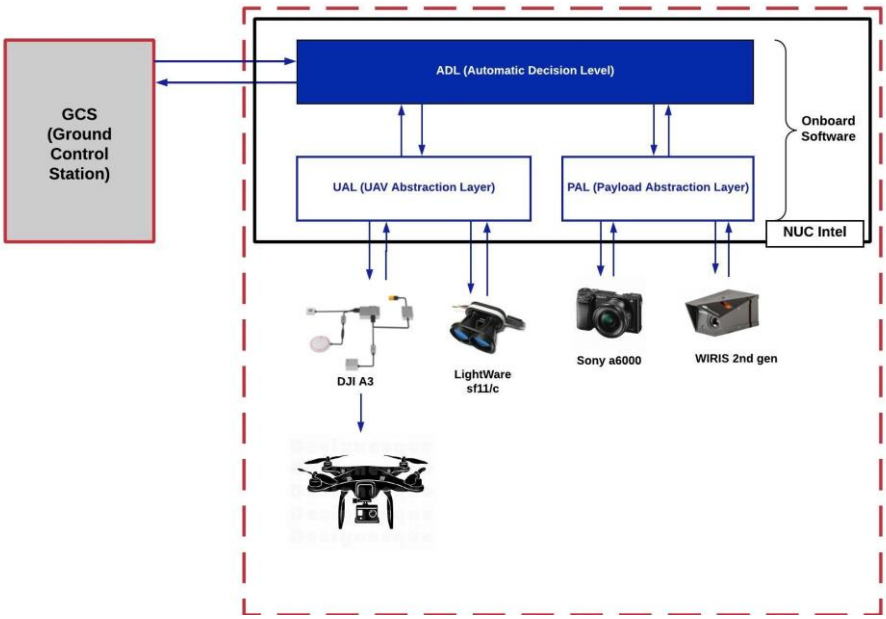

Figura 6. Arquitecturas hardware y software de la solución adoptada.

La capa ADL se ha desarrollado en base a una estructura de máquina de estado haciendo uso de la herramienta de ROS denominada Smach. Se puede ver un esquema de la misma en la Figura 7.

Está compuesta de las siguientes partes:

- **Estado de espera "Standby":** al lanzar la máquina de estado, siempre que no haya ninguna misión ya comenzada, se iniciará en este estado donde queda a la espera de recibir los datos de la misión y la orden de comenzar.
- **Estado de reanudación "Paused Mission":** si una misión ya iniciada se detiene (esto se podría dar porque se esté agotando la batería de UAV o porque se le de la orden de parar de forma manual desde la GCS) la máquina de estado, al ser lanzada de nuevo, se iniciaría en este estado. Desde la GCS se podrá dar la orden de reanudar la misión o cancelarla, volviendo así al estado Standby.
- **Sub-máquina de estado "Mission":** es la parte que gestiona el desarrollo de la misión. Se encarga de la navegación autónoma mediante la interacción con la UAL y de la captura de imágenes en base a los servicios la capa PAL.

  En la Figura 7 se puede ver el detalle de los estados involucrados. Tras despegar a la altura libre de conflictos (WP_00), el UAV iría hasta el waypoint WP_01 a la misma altura; seguidamente pasaría al primer waypoint del barrido; una vez realizado el barrido iría de nuevo a la altura de seguridad para volver hasta las coordenadas desde donde despegó y finalmente aterrizar.

  También puede verse como desde cualquier estado se puede interrumpir la misión a través del servicio "stop_mission". A este servicio se le puede llamar de forma manual desde la GCS y también se activará automáticamente en caso de que el nivel de batería sea excesivamente bajo.

  Como se puede ver, si se detiene la misión durante el estado de barrido antes de comenzar la vuelta a la base se pasaría por el estado "SAVE_CURRENT_POSITION" donde se almacena la posición actual seguida de los waypoints restantes en una nueva lista que permitirá reanudar la misión en el punto en el que se detuvo.
- **Sub-maquina de estado "Download":** se pasará aquí tras terminar una misión, una vez que el UAV haya aterrizado, y se procederá a la descarga automática desde la NUC hacia la GCS de todas las imágenes tomadas durante el vuelo. Una terminado este proceso se pasará de nuevo al estado Standby.

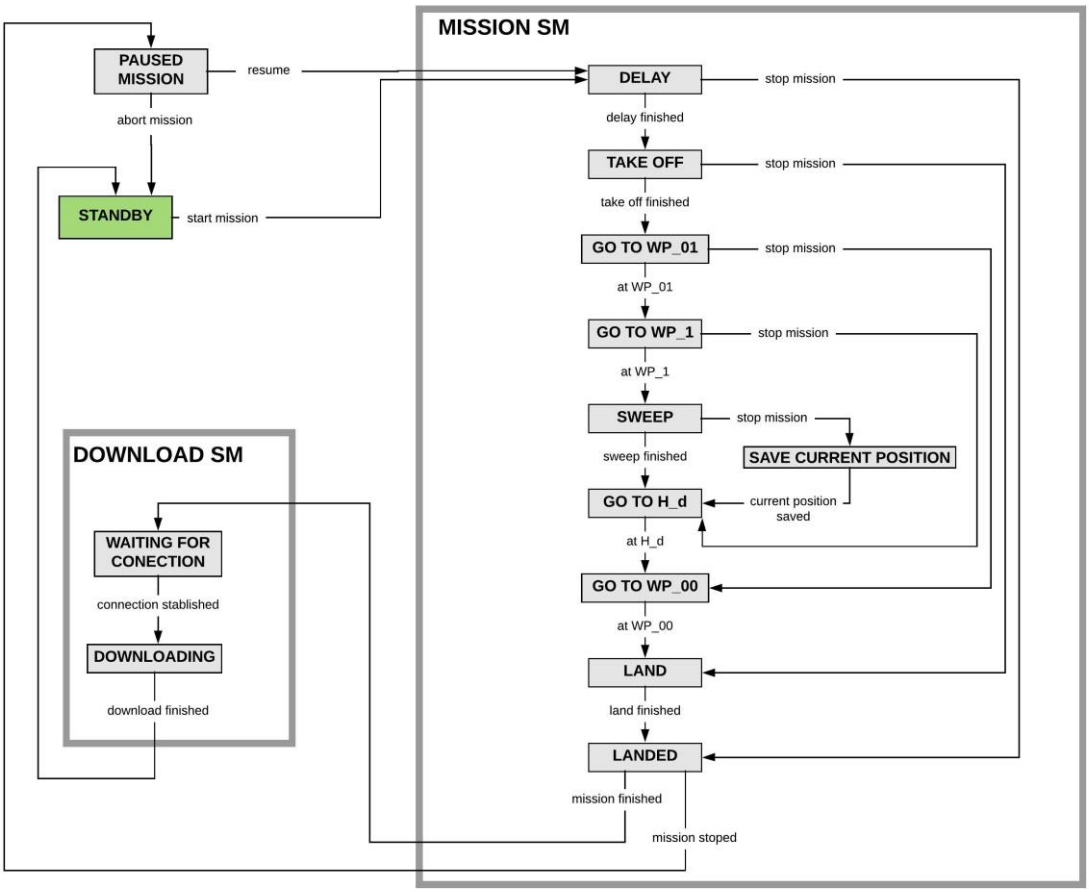

Figura 7. Esquema de la máquina de estados en la que está basada la implementación software de la capa ADL (Automatic Decision Layer) abordo del UAV.

3.3 Vuelo AGL

Teniendo en cuenta que en algunas plantas fotovoltaicas puede haber desniveles en el terreno de hasta 25 metros entre unas zonas de paneles y otras, se plantea la necesidad de que el vuelo se realice en modo AGL.

Hasta ahora el vuelo se realizaba en modo AMSL (Abobe Mean Sea Level), es decir manteniendo una altitud constante sobre el nivel del mar. Esta era la única posibilidad, ya que para el control de altura el autopiloto utiliza únicamente la medida del barómetro que lleva incorporado y del sistema GNSS.

El principal problema que presentaba el vuelo AMSL es que todas las fotografías no se realizarían a la misma altura del suelo, de forma que variaría notablemente el solape y la superficie cubierta por cada píxel. Esto podría ocasionar problemas a la hora de analizar las imágenes mediante el software desarrollado por USAL. Con el vuelo en modo AGL se consigue evitar esto al mantener una altura constante sobre el suelo.

Para conseguir volar en modo AGL ha sido necesario integrar en el UAV un altímetro láser. Se ha optado en concreto por el modelo de LightWare sf11/c (ver Figura 8). Este altímetro tiene un rango de medida de 0 a 120 metros, una resolución de 1 cm y pesa solamente 35 gramos por lo que presenta un buen compromiso entre rendimiento y peso.

Debido a que el autopiloto A3 de DJI no permite la conexión directa de un altímetro externo, se ha desarrollado un *driver* para poder tomar la lectura del sf11/c directamente a través del puerto serie. Además,

se ha integrado en la capa UAL una función que corrige la altitud basándose en la lectura del altímetro láser.

Figura 8. Altímetro láser LightWare sf11/c.

4 CAPTURA DE DATOS DE LOS SENSORES A BORDO

En este capitulo se trata más detalladamente la integración de los sensores a bordo del UAV, centrándose en el desarrollo de la capa PAL (Payload Abstraction Layer). Esto incluye la toma de datos de los sensores del autopiloto A3, del altímetro láser y de las camaras a bordo del UAV.

Se describe el sistema que permite el envío de información correspondiente a la telemetría y de las imágenes en vivo, de cada UAV, a la GCS. Seguidamente, se detalla el postprocesado de estos datos que se llevará a cabo en la GCS una vez finalizada la misión.

Para las cámaras Sony y WIRIS, se muestra la integración a nivel software de que permite controlar la captura de imágenes y la descarga de estas a la NUC.

Por último, se ha incluye el desarrollo que permite subir a la GCS todos los datos de la misión, almacenados en la NUC, al finalizar esta.

4.1 Integración de los sensores a bordo del UAV y captura de datos

Recordando la arquitectura software ya definida en el anterior capitulo (ver Figura 6), la capa PAL será la que directamente interaccione con las distintas cámaras, encargándose de la captura de todas las imágenes y, siempre que la conexión inalámbrica lo haga posible, del envío en vivo de las imágenes de la webcam.

La capa ADL será la que, en función del estado de la misión y de la posición en la que se encuentre el UAV, tome la decisión de iniciar o detener la captura de datos de los sensores, dando estas "órdenes" a la capa PAL. La comunicación entre ambas se realizará a través de servicios ROS. De esta forma se abstrae a la capa ADL de la interacción con las cámaras a más bajo nivel.

En los siguientes subapartados se describen los distintos sensores que se montarán en la plataforma.

4.1.1 Sensores del autopiloto DJI A3

El autopiloto lleva integrados, entre otros, los siguientes sensores que proporcionan todos los datos correspondientes a la telemetría del UAV (posición GPS, altitud barométrica, velocidad, datos de la IMU, etc):

- Barómetro: permite medir variaciones en la presión atmosférica y, en base a este dato, detectar variaciones en la altitud del vehículo.
- IMU (Inertial Measurement Unit): este dispositivo da información acerca de la velocidad, orientación y fuerzas gravitacionales del vehículo usando una combinación de acelerómetros y giróscopos.
- Modulo GNSS (Global Navigation Satellite System) tipo GPS: proporciona información de la localización en coordenadas geográficas y altitud. Aunque el valor de la altitud proporcionado por el sistema GPS no es tan preciso, al combinarlo con la información procedente del barómetro se puede obtener un dato más exacto.

El nodo *dji_sdk_node,* incluido en el SDK (Software Development Kit) que proporciona DJI para ROS, publica estos datos a través de los siguientes *topics* de ROS:

- */dji_sdk/gps_position:* Fusiona la posición global del UAV en latitud, longitud y altitud (en metros). Este valor de altitud es siempre con respecto al punto de despegue del UAV, en el que el autopiloto

establece su referencia de posición local en coordenadas cartesianas. Publica un mensaje tipo *sensor_msgs/NavFix* con una frecuencia de 50 Hz.

- *dji_sdk/local_position:* Posición local en coordenadas cartesianas con marco de referencia ENU (East North Up). Publica un mensaje tipo geometry_*msgs/PointStamped* con una frecuencia de 50 Hz.
- *dji_sdk/gps_health:* Calidad de la señal del GPS. El dato es un entero que varía entre 1 y 5, donde 5 es la mejor condición. Publica un mensaje tipo *sensor_msgs/UInt8* con una frecuencia de 50 Hz.
- *dji_sdk/imu:* Datos de la IMU incluyendo lectura de giróscopo, acelerómetro y estimación de la actitud en referencia FLU (Front Left Up) con respecto al cuerpo del UAV. Publica un mensaje tipo *sensor_msgs/Imu* con una frecuencia de 100 Hz.
- *dji_sdk/attitude:* Actitud del vehículo representada en cuaternios para la rotación del cuerpo en FLU con respecto al marco de referencia ENU. Publica un mensaje tipo *sensor_msgs/QuaternionStamped* con una frecuencia de 100 Hz.
- *dji_sdk/velocity:* Velocidad en referencias ENU. Publica un mensaje tipo *sensor_msgs/Vector3Stamped* con una frecuencia de 50 Hz. Solamente está disponible cuando la calidad de la señal GPS es mayor o igual a 3.

La capa UAL se suscribe a estos *topics* y utiliza algunos de estos datos para el control del UAV.

La capa PAL también se suscribe a estos *topics* y se encarga de recoger y almacenar en un *rosbag* algunos de estos datos junto con sus marcas de tiempo.

4.1.2 Altímetro láser LightWare sf11/c

En el anterior capitulo ya se justificó la necesidad de integrar este sensor, que hace posible el vuelo AGL, y se detalló su integración a nivel software.

El modelo sf11/c de LightWare (Figura 8) se caracteriza por tener un rango de medida de entre 0 y 120 metros, con una resolución de 1 cm.

Al igual que el autopiloto, el altímetro se conecta directamente al ordenador de abordo (NUC de Intel) por puerto serie. Se ha desarrollado un nodo de ROS que obtiene la medida del altímetro y la publica en el siguiente "topic":

- */laser_altitude:* Proporciona como dato la altitud (en metros) con respecto al terreno. Publica un mensaje tipo *std_msgs/Float64* con una frecuencia de 20 Hz.

De la misma forma, UAL se suscribe a este *topic* y utiliza el dato de la altitud para el control del UAV. Asimismo, la capa PAL se encarga de almacenar la información que proporciona el altímetro láser en un *rosbag* junto con sus marcas de tiempo.

4.1.3 Webcam Logitech C270 HD

Se decidió integrar esta cámara (Figura 9) para cumplir el requisito de envío de imágenes en vivo, ya que esto no es posible con la cámara Sony A6000. Se eligió un modelo de Logitech por ofrecer un buen compromiso entre resolución y peso, además de la posibilidad de integración en ROS.

Esta cámara permite obtener imágenes RGB con una resolución ajustable hasta un máximo de 1280x720

píxeles y una tasa de hasta 30 fps para resoluciones menores.

Experimentalmente se ha determinado que la resolución recomendable para hacer el envío de imágenes en vivo desde el UAV durante la ejecución de la misión es de 640x480 píxeles. A esta resolución se pueden obtener imágenes a una tasa aproximada de 18 fps si el enlace de comunicaciones lo permite.

Al igual que los sensores anteriores, también se conecta por puerto serie a la NUC y se hace uso de dos paquetes de ROS para su integración: *image_view* y *usb_cam*. Las imágenes se publican a través del siguiente *topic*:

- */usb_cam/image_raw:* Imágenes de la webcam en formato raw. Publica un mensaje tipo *sensor_msg/Image* a la frecuencia que se haya configurado (en este caso 18 Hz).

La capa PAL puede almacenar esta información en *rosbags* junto a sus marcas de tiempo, de la misma forma que se hace para los otros sensores. Posteriormente desde la GCS se podrá descargar el archivo .bag y extraer de él las imágenes en formato png junto a sus marcas de tiempo correspondiente.

Figura 9. Webcam Logitech C270 HD

4.1.4 Cámara RGB Sony A6000

Con el objetivo de obtener imágenes RGB y poder formar con estas un mosaico de toda el área de la planta se ha integrado en el UAV la cámara Sony A6000 (Figura 10).

Para la integración de esta cámara se ha hecho uso de gphoto2, un proyecto open source para Linux, basado en una serie de librerías, que permite con control a través de usb de más de 2000 modelos de cámaras digitales. Esto supone una ventaja adicional: en el caso de querer utilizar otro modelo de cámara en el futuro, la integración de esta sería bastante simple basándose en el desarrollo ya realizado.

Haciendo uso de algunas de las funciones que proporciona gphoto2, se han incluido en el script de la capa PAL los siguientes servicios de ROS que el control de esta cámara y la descarga de sus imágenes:

- */rgb_camera_connection_service:* Comprueba que la cámara se encuentra disponible e inicia la conexión con esta. En el caso de que no se pueda establecer la conexión devolverá un mensaje de error.
- */rgb_camera_capture_service:* Este servicio abre y cierra un hilo que se encarga de la captura de imágenes con una frecuencia determinada, siempre que previamente se haya establecido la conexión adecuadamente. Al llamar a este servicio se le pasan dos argumentos de entrada: la orden de iniciar o detener la toma de imágenes y la frecuencia a la que se realiza. De esta forma, llamando a este servicio desde la capa PAL se podría ir variando la frecuencia de captura en función de la velocidad del UAV si esto fuese necesario.

En cuando se realiza una captura, esta imagen se va almacenando directamente en el directorio correspondiente a esa misión, dándole al archivo un nombre que incluye la fecha y hora de a captura. Además, los nombres de estos archivos se van almacenando en un fichero csv seguidos de una marca de tiempo de ROS. Esto permitirá, una vez terminada la misión, asociar a cada imagen los datos de telemetría mas cercanos, de la misma forma que se hizo con las imágenes de la webcam.

Es importante destacar que la frecuencia a la que se capturan las imágenes está limitada por el procesamiento de la propia cámara. Aunque este tiempo no es fijo, ya que depende de factores como la iluminación, se ha determinado de forma experimental el intervalo de tiempo mínimo aproximado con el que la cámara es capaz de realizar capturas, en función del tamaño definido para las imágenes. Así, este tiempo es de aproximadamente 2,3 segundos para imágenes de 24 MP, 1,7 para 12 MP y 1,3 para 6 MP.

Figura 10. Cámara RGB Sony A6000 con objetivo 16-50 mm.

4.1.5 Cámara térmica Workswell WIRIS de segunda generación

Para la obtención de imágenes termográficas se ha elegido la cámara WIRIS 2nd generation de Workswell (Figura 11). Esta está especificamente diseñada para ser integrada en UAVs y permite establecer comunicación con el ordenador de abordo a través de ethernet, para la transmisión de imágenes en streaming y el control de la propia cámara, . El control de la cámara se hace a mediante el protocolo de comunicación TCP/IP, mientras que para el acceso a las imágenes almacenadas en la memoria de la cámara se utiliza el standard FTP. La descarga de las imágenes no se puede hacer en justo después de la captura, como se hacía con la cámara RGB, por eso se incluye otro servicio que realiza la descarga de todos los ficheros una vez terminada la misión.

Se han incluido los siguientes servicios de ROS en el script de la capa PAL:

- */thermal_camera_connection_service:* Al igual con la cámara rgb, este servicio comprueba que la cámara se encuentra disponible e inicia la conexión con esta a través de TCP/IP. En el caso de que no se pueda establecer la conexión devolverá un mensaje de error.

- */thermal_camera_capture_service:* Este funciona de la misma forma que su equivalente para la cámara rgb con la diferencia de que no permite descargar directamente las imágenes. Tampoco será necesario crear un registro de estas para su postprocesado, ya que lo realiza la propia cámara.
- */thermal_camera_download_service:* Una vez terminado el barrido, cuando se detiene la captura de imágenes, desde la capa ADL se llamará a este servicio. Este accede mediante comunicación FTP a la memoria interna de la cámara WIRIS, descarga todos los archivos a la NUC y libera la memoria de la cámara.

En el caso de esta cámara no es limitante el intervalo mínimo de tiempo posible entre una captura y otra, ya que es mucho menor que en el caso de la cámara Sony (se puede conseguir una frecuencia de captura de hasta 5 imágenes por segundo).

Figura 11. Cámara termográfica Workswell WIRIS 2nd gen

4.2 Envío de información a la GCS en vivo

Para establecer la comunicación entre la GCS y los distintos UAVs a través de ROS se hace uso de del paquete multimaster_fkie. Este paquete incluye una serie de nodos que hacen posible establecer y gestionar una red multi-maestro en ROS.

La principal ventaja de utilizar un sistema multi-maestro en una red inalámbrica con varios dispositivos que utilizan ROS es que, en el caso de que uno o varios de estos dispositivos pierdan de forma temporal la conexión con la red, todos los dispositivos de la red pueden seguir funcionando de forma independiente, gracias a que todos los nodos que se están ejecutando en cada dispositivo dependen únicamente del nodo master de ese dispositivo.

Haciendo uso de este paquete se puede hacer que cada nodo maestro haga públicos en la red a la que está conectado unos servicios y *topics* determinados a los que podrán suscribirse los nodos maestros del resto de dispositivos.

Para el caso de la telemetría, el maestro que se está ejecutando en la NUC de cada uno de los UAV hará públicos en una serie de *topics* esa información sobre el estado del vehículo, incluyendo un identificador único de cada UAV. El maestro de la GCS se suscribe a estos *topics* y obtiene esta información.

De forma similar se realiza el proceso para obtener las imágenes en vivo. El maestro de cada UAV hace público el *topic* proporcionado por el nodo *usb_cam* precedido de un identificador único para cada UAV. Desde la interfaz gráfica de la GCS se dispondrá de un visualizador proporcionado por el paquete

image_view que podrá suscribirse a estos *topics* y mostrar las imágenes que reciba a través de la red inalámbrica.

4.3 Asociación post-misión de las marcas de tiempo de los datos

Se ha desarrollado un *script* en *python* que se podrá ejecutar desde la GCS para el postprocesado de los datos. Una vez terminada la misión y descargados todos los ficheros .bag grabados durante el vuelo con la información de los distintos sensores, este script procederá a extraer las imágenes del fichero correspondiente a la webcam y asociarle a cada una sus datos de telemetría correspondientes.

En primer lugar, se extraen, como ya se ha mencionado, las imágenes procedentes del fichero de datos de la webcam. Estas imágenes se almacenan en una carpeta junto a un archivo en formato csv que contiene una base de datos con los nombres de todos los archivos de imagen y su marca de tiempo correspondiente. En la Figura 12 se puede ver un ejemplo de dicho archivo csv.

	A	B	C	D	E
1	Time	Header_ sequence	Header_secs	Header_nsecs	Filename
2	1.54782144686105E+018	1274	1547821446	858085805	
3	1.54782144689208E+018	1275	1547821446	889258596	ebcamImage-Image_0001
4	1.54782144692971E+018	1276	1547821446	925681620	ebcamImage-Image_0002
5	1.54782144696153E+018	1277	1547821446	958087104	ebcamImage-Image_0003
6	1.54782144698961E+018	1278	1547821446	988176893	ebcamImage-Image_0004
7	1.5478214470246E+018	1279	1547821447	23603651	ebcamImage-Image_0005
8	1.54782144705705E+018	1280	1547821447	55852120	ebcamImage-Image_0006
9	1.54782144709264E+018	1281	1547821447	89704927	ebcamImage-Image_0007
10	1.5478214471285E+018	1282	1547821447	125529107	ebcamImage-Image_0008
11	1.54782144716062E+018	1283	1547821447	157558079	ebcamImage-Image_0009
12	1.54782144719659E+018	1284	1547821447	193471621	ebcamImage-Image_0010
13	1.54782144722812E+018	1285	1547821447	225288840	ebcamImage-Image_0011
14	1.54782144726064E+018	1286	1547821447	257409509	ebcamImage-Image_0012
15	1.54782144729633E+018	1287	1547821447	293447023	ebcamImage-Image_0013
16	1.54782144732872E+018	1288	1547821447	325600939	ebcamImage-Image_0014
17	1.54782144736081E+018	1289	1547821447	357866833	ebcamImage-Image_0015
18	1.54782144739648E+018	1290	1547821447	393472382	ebcamImage-Image_0016
19	1.54782144742861E+018	1291	1547821447	425687973	ebcamImage-Image_0017
20	1.54782144746436E+018	1292	1547821447	461388489	ebcamImage-Image_0018
21	1.54782144749681E+018	1293	1547821447	493877198	ebcamImage-Image_0019

Figura 12. Información de todas las imágenes extraídas del fichero bag junto a sus marcas de tiempo correspondientes.

Seguidamente, el programa genera otro archivo csv a partir de cada fichero .bag de datos de telemetría. Estos contienen todas las medidas de telemetría tomadas por los sensores (latitud, longitud, etc) y sus respectivas marcas de tiempo (ver Figura 13).

	A	B	C	D	E	F	G	
1	Time	Header_sequence	Header_secs	Header_nsecs	Latitude	Longitude	Altitude	PositionCovarian
2	1.5478214468421E+018	865	1547821446	840972312	43.5204002764	-5.607544816	3.0829532146	(0.0, 0.0, 0.0, 0.0
3	1.54782144690615E+018	866	1547821446	905206024	43.5204019832	-5.6075443443	3.0832736492	(0.0, 0.0, 0.0, 0.0
4	1.54782144692795E+018	867	1547821446	926719079	43.5204025518	-5.6075441872	3.0833804607	(0.0, 0.0, 0.0, 0.0
5	1.54782144699189E+018	868	1547821446	991561710	43.5204048249	-5.607543559	3.0837967396	(0.0, 0.0, 0.0, 0.0
6	1.54782144705627E+018	869	1547821447	55860922	43.5204065284	-5.6075430882	3.0841054916	(0.0, 0.0, 0.0, 0.0
7	1.54782144707793E+018	870	1547821447	77413901	43.520407096	-5.6075429314	3.0842046738	(0.0, 0.0, 0.0, 0.0
8	1.54782144709963E+018	871	1547821447	98827325	43.5204076634	-5.6075427746	3.0843038559	(0.0, 0.0, 0.0, 0.0
9	1.54782144712167E+018	872	1547821447	120473819	43.5204082307	-5.6075426178	3.084403038	(0.0, 0.0, 0.0, 0.0
10	1.5478214471641E+018	873	1547821447	163280104	43.5204093649	-5.6075423043	3.0846014023	(0.0, 0.0, 0.0, 0.0
11	1.54782144718571E+018	874	1547821447	184796550	43.5204099318	-5.6075421476	3.0847005844	(0.0, 0.0, 0.0, 0.0
12	1.54782144720721E+018	875	1547821447	206138478	43.5204104986	-5.607541991	3.0847959518	(0.0, 0.0, 0.0, 0.0
13	1.54782144729329E+018	876	1547821447	292503703	43.5204133306	-5.6075412083	3.085272789	(0.0, 0.0, 0.0, 0.0
14	1.54782144731481E+018	877	1547821447	314011088	43.5204138967	-5.6075410518	3.0853655338	(0.0, 0.0, 0.0, 0.0
15	1.54782144735778E+018	878	1547821447	356991851	43.5204150284	-5.607540739	3.0855486393	(0.0, 0.0, 0.0, 0.0
16	1.54782144744373E+018	879	1547821447	442826656	43.5204172902	-5.6075401139	3.0859143734	(0.0, 0.0, 0.0, 0.0
17	1.54782144748663E+018	880	1547821447	485668695	43.5204184203	-5.6075398015	3.0860898495	(0.0, 0.0, 0.0, 0.0
18	1.54782144750835E+018	881	1547821447	507393133	43.5204189853	-5.6075396454	3.0861775875	(0.0, 0.0, 0.0, 0.0
19	1.54782144753065E+018	882	1547821447	528528273	43.52041955	-5.6075394892	3.0862653255	(0.0, 0.0, 0.0, 0.0
20	1.54782144755102E+018	883	1547821447	550054816	43.5204201148	-5.6075393332	3.0863530636	(0.0, 0.0, 0.0, 0.0
21	1.54782144757273E+018	884	1547821447	571855489	43.5204212437	-5.6075390211	3.0865273476	(0.0, 0.0, 0.0, 0.0
22	1.54782144759433E+018	885	1547821447	593471054	43.520421808	-5.6075388652	3.0866112709	(0.0, 0.0, 0.0, 0.0
23	1.5478214476373E+018	886	1547821447	636429469	43.5204229363	-5.6075385533	3.0867791176	(0.0, 0.0, 0.0, 0.0
24	1.54782144768022E+018	887	1547821447	679278465	43.5204240641	-5.6075382416	3.0869469643	(0.0, 0.0, 0.0, 0.0

Figura 13. Datos de la telemetría del UAV junto a sus respectivas marcas de tiempo.

Por último, se utiliza un algoritmo de comparación para asociar a cada imagen los datos de la telemetría más cercanos a cada una en función de sus respectivas marcas de tiempo. Se genera una copia del archivo cvs que se obtuvo junto con las imágenes, añadiendo nuevas columnas a esa base de datos con la latitud, longitud y altitud asociadas a cada imagen. En la Figura 14 se puede ver un archivo que se ha generado a modo de ejemplo, a partir de una simulación, en el que aparecen las imágenes de la webcam georeferenciadas de manera aproximada.

	A	B	C	D	E	F	G	H	I
1		Time	Header_ sequence	Header_secs	Header_nsecs	Filename	Latitude	Longitude	Altitude
2	0	1.54782144686105E+018	1274	1547821446	858085805		43.5204002764	-5.607544816	3.0829532146
3	1	1.54782144689208E+018	1275	1547821446	889258596	ebcamImage-Image_0001	43.5204019832	-5.6075443443	3.0832736492
4	2	1.54782144692971E+018	1276	1547821446	925681620	ebcamImage-Image_0002	43.5204025518	-5.6075441872	3.0833804607
5	3	1.54782144696153E+018	1277	1547821446	958087104	ebcamImage-Image_0003	43.5204048249	-5.607543559	3.0837967396
6	4	1.54782144698961E+018	1278	1547821446	988176893	ebcamImage-Image_0004	43.5204048249	-5.607543559	3.0837967396
7	5	1.5478214470246E+018	1279	1547821447	23603651	ebcamImage-Image_0005	43.5204065284	-5.6075430882	3.0841054916
8	6	1.54782144705705E+018	1280	1547821447	55852120	ebcamImage-Image_0006	43.5204065284	-5.6075430882	3.0841054916
9	7	1.54782144709264E+018	1281	1547821447	89704927	ebcamImage-Image_0007	43.5204076634	-5.6075427746	3.0843038559
10	8	1.5478214471285E+018	1282	1547821447	125529107	ebcamImage-Image_0008	43.5204082307	-5.6075426178	3.084403038
11	9	1.54782144716062E+018	1283	1547821447	157558079	ebcamImage-Image_0009	43.5204093649	-5.6075423043	3.0846014023
12	10	1.54782144719659E+018	1284	1547821447	193471621	ebcamImage-Image_0010	43.5204104986	-5.607541991	3.0847959518
13	11	1.54782144722812E+018	1285	1547821447	225288840	ebcamImage-Image_0011	43.5204104986	-5.607541991	3.0847959518
14	12	1.54782144726064E+018	1286	1547821447	257409509	ebcamImage-Image_0012	43.5204133306	-5.6075412083	3.085272789
15	13	1.54782144729633E+018	1287	1547821447	293447023	ebcamImage-Image_0013	43.5204133306	-5.6075412083	3.085272789
16	14	1.54782144732872E+018	1288	1547821447	325600939	ebcamImage-Image_0014	43.5204138967	-5.6075410518	3.0853655338
17	15	1.54782144736081E+018	1289	1547821447	357866833	ebcamImage-Image_0015	43.5204150284	-5.607540739	3.0855486393
18	16	1.54782144739648E+018	1290	1547821447	393472382	ebcamImage-Image_0016	43.5204150284	-5.607540739	3.0855486393
19	17	1.54782144742861E+018	1291	1547821447	425687973	ebcamImage-Image_0017	43.5204172902	-5.6075401139	3.0859143734
20	18	1.54782144746436E+018	1292	1547821447	461388489	ebcamImage-Image_0018	43.5204172902	-5.6075401139	3.0859143734
21	19	1.54782144749681E+018	1293	1547821447	493877198	ebcamImage-Image_0019	43.5204184203	-5.6075398015	3.0860898495
22	20	1.5478214752878E+018	1294	1547821447	525868120	ebcamImage-Image_0020	43.52041955	-5.6075394892	3.0862653255
23	21	1.54782144756454E+018	1295	1547821447	561448391	ebcamImage-Image_0021	43.5204212437	-5.6075390211	3.0865273476
24	22	1.5478214475979E+018	1296	1547821447	593880714	ebcamImage-Image_0022	43.520421808	-5.6075388652	3.0866112709
25	23	1.54782144762892E+018	1297	1547821447	625901998	ebcamImage-Image_0023	43.5204229363	-5.6075385533	3.0867791176
26	24	1.5478214476646E+018	1298	1547821447	661469760	ebcamImage-Image_0024	43.5204240641	-5.6075382416	3.0869469643
27	25	1.54782144769665E+018	1299	1547821447	693584809	ebcamImage-Image_0025	43.5204246278	-5.6075380857	3.0870308876
28	26	1.54782144772886E+018	1300	1547821447	725990672	ebcamImage-Image_0026	43.5204251914	-5.60753793	3.0871140957
29	27	1.54782144776463E+018	1301	1547821447	761590869	ebcamImage-Image_0027	43.5204257549	-5.6075377742	3.0871942043
30	28	1.54782144779718E+018	1302	1547821447	794177853	ebcamImage-Image_0028	43.5204280077	-5.6075371515	3.0875146389
31	29	1.54782144783434E+018	1303	1547821447	829537555	ebcamImage-Image_0029	43.5204285705	-5.607536996	3.0875947475
32	30	1.54782144786475E+018	1304	1547821447	861658215	ebcamImage-Image_0030	43.520429696	-5.6075366849	3.0877523422
33	31	1.54782144789808E+018	1305	1547821447	893662761	ebcamImage-Image_0031	43.5204302586	-5.6075365294	3.0878286362

Figura 14. Información de las imágenes georeferenciadas de manera aproximada

Cabe destacar que la georeferenciación de las imágenes se hace de manera aproximada ya que la cámara no admite la posibilidad de emplear ningún mecanismo de sincronización hardware con el autopiloto A3.

4.4 Subida de archivos a la GCS

Se ha definido dentro de la capa PAL un servicio de ROS (*'/data_upload_*service') que permite subir a la GCS todos los datos obtenidos en la misión.

En la última etapa de la misión, una vez se halla terminado de descargar en la NUC las imágenes de la cámara térmica, y siempre que haya conexión entre la NUC y la GCS, se podrá llamar a este servicio.

Este servicio abre una conexión, basada en el protocolo scp, entre la NUC y la GCS; crea en la GCS un directorio con el nombre de la misión correspondiente y un identificador propio del UAV y sube a este todos los datos obtenidos durante la misión (imágenes obtenidas por las cámaras y datos de telemetría)

5 MANEJO REMOTO DE LA FLOTA

En este capitulo se tratan las diferentes herramientas que permitirán el manejo remoto de los UAVs. Por un lado, se encuentra la GUI (Interfaz Gráfica de Usuario) desarrollada para el control desde la propia GCS y, por otro lado, el servicio API Rest para el control desde una ubicación remota (ver parte inferior de la Figura 15).

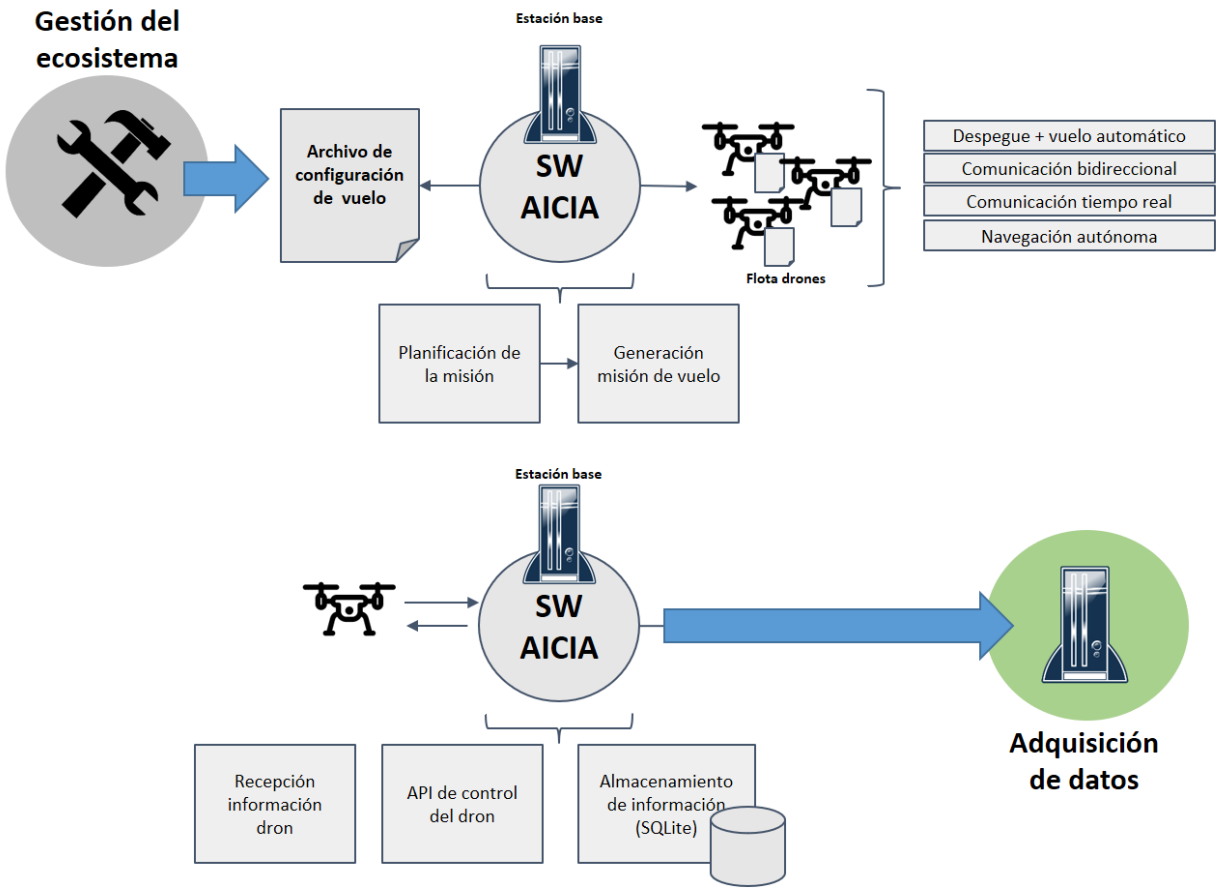

Figura 15. En la parte superior se detalla el esquema de comunicaciones a nivel de la GCS. En la parte inferior de la imagen se representa el esquema de comunicaciones a nivel remoto.

5.1 Software para la generación de misiones y control de los UAVs desde la GCS

Como ya se ha descrito previamente en este documento, el software creado para la GCS se ha desarrollado usando ROS sobre C++. Este se ha diseñado para que sea completamente abierto a distintas interfaces de usuario.

Por un lado, se lanza el nodo principal de la GCS, encargado de la generación de misiones y el control de los UAVs. Todas las funciones de este nodo se controlan a través de servicios de ROS. Esto hace que se pueda utilizar tanto directamente desde la terminal de Ubuntu (útil durante la fase de desarrollo y testeo), como desde cualquier otro programa o interfaz de usuario que se comunique directamente con esos servicios de ROS.

Por otro lado, se podrán lanzar uno o más nodos independientes del anterior que se comunican con él a través de estos servicios de ROS y facilitan al usuario el manejo del sistema. Para este proyecto se ha desarrollado

dos: uno que lanza la Interfaz Gráfica de Usuario en la GCS y otro que hace de servidor de la API Rest.

5.1.1 Servicios de ROS para la generación de misiones y control de los UAVs

En el capitulo 4 se justificó el uso de la herramienta multimaster de ROS. Como se describió, esta permite que distintos 'masters' (el de la GCS y los de las NUCs de cada uno de los UAVs) puedan compartir determinados 'topics' y servicios de ROS.

A continuación, se describen los servicios implementados para la generación de las misiones y el control de los UAVs:

5.1.1.1 Generación de misiones

- */uav_link:* El nodo principal de la GCS genera este servicio. Cada vez que la NUC de un UAV se conecta a la misma red que la GCS se llama a este servicio. A través de éste, cada UAV envía a la GCS su identificador propio, que es almacenado en una lista y utilizado para los servicios de control de ese UAV.
- */create_mission: A*l llamar a este servicio se genera la misión en base a los datos del archivo *mission_file.json* y el número de UAVs disponibles en la lista en ese momento.
- */send_mission_to_uavs:* Cuando se llama a este servicio, una vez creada la misión, se envía a cada uno de los UAVs disponibles su parte correspondiente de esta.

5.1.1.2 Control de los UAVs

Por cada uno de los UAVs conectados aparecerán los siguientes servicios, precedidos del identificador propio de cada UAV. Aparecerán unos u otros dependiendo del estado en el que se encuentre la misión:

- */stby_action_service:* Recordando la descripción de la capa ADL (capitulo 3), al lanzar este nodo, si no hay ninguna misión pausada, la máquina de estados pasa directamente al modo "Standby" donde queda a la espera de recibir la misión. Una vez haya recibido la misión, mediante este servicio se da al UAV la orden de comenzar.
- */stop_service: S*i el UAV se encuentra ejecutando la misión, mediante este servicio se puede dar la orden de detenerla. Cuando se detiene una misión la máquina de estados pasa al modo "Paused_state".
- */paused_state_action_service: E*ste servicio estará disponible cuando la máquina de estados se encuentre en el modo "Paused_state". Este servicio da la opción de reanudar la misión (mediante el mensaje "RESUME_PAUSED_MISSION") o de abortar la misión actual y comenzar una nueva (mediante el mensaje "START_NEW_MISSION"); de esta forma pasaría de nuevo al modo "Standby".

Poe ejemplo, si se tiene un UAV cuyo identificador es "uav_id" aparecerán disponibles los servicios:

- */uav_id/stby_action_service*
- */uav_id/stop_service*
- */uav_id/paused_state_action_service*

5.2 Interfaz Gráfica de Usuario para la GCS

La GUI servirá para el control y monitorización de las misiones desde la propia estación de tierra, haciendo más accesible al usuario las funciones del software de la GCS. Desde aquí se podrá modificar el archivo json con los parámetros de la misión, generar y cargar las misiones, y dar a los UAVs las ordenes necesarias: comenzar la misión, detenerla, reanudarla (después de haberla detenido manualmente o de que se haya detenido de forma automática debido a un nivel de las baterías excesivamente bajo) o abortarla.

5.2.1 Librerías empleadas en la implementación

La GUI se desarrolla sobre rqt[1], un framework de ROS basado en Qt que permite crear interfaces gráficas.

Para el desarrollo de una parte de la interfaz de usuario se ha utilizado, dentro de rqt, la librería Qt-GUI[2].

Se ha utilizado también un plugin de rviz[3] (herramienta de visualización 3D de ROS) integrado dentro de la ventana de rqt que permite visualizar las misiones de los UAVs. Por último, se ha utilizado un paquete llamado *rviz_satellite*[4] que permite cargar mapas en rviz. En este caso se utilizan los mapas libres de OpenStreetMap[5].

5.2.1.1 ROS-rqt

ROS-rqt implementa diversas herramientas para realizar interfaces gráficas de usuario en forma de plugins. Estas herramientas GUI existentes en ROS se pueden ejecutar sin hacer uso de rqt de forma independiente, pero rqt facilita el manejo de estas herramientas permitiendo ejecutarlas todas como plugins dentro de una misma ventana personalizada.

Permite ejecutar varias herramientas GUI existentes en ROS como plugins embebidos dentro de una ventana de rqt y almacenar esta ventana para restaurarla. El usuario puede además crear sus propios plugins para rqt, con Python o C++, basados en la librería Qt-GUI.

5.2.1.2 Qt-GUI

El módulo Qt GUI proporciona clases para la integración de sistemas de ventanas, manejo de eventos, integración OpenGL y OpenGL ES, gráficos 2D, imágenes básicas, fuentes y texto. Estas clases son utilizadas internamente por las tecnologías de interfaz de usuario de Qt y también pueden ser utilizadas directamente, por ejemplo, para escribir aplicaciones utilizando las APIs de gráficos OpenGL ES de bajo nivel.

5.2.1.3 OpenStreetMap

OpenStreetMap está construido por una comunidad que contribuye a mantener datos actualizados sobre carreteras, senderos, cafés, estaciones de tren, etc. a nivel mundial. Dicha comunidad emplea imágenes aéreas, dispositivos GPS y mapas de campo para verificar que el mapa es preciso y está actualizado.

Entre sus colaboradores se encuentran cartógrafos, profesionales de SIG, ingenieros que mantienen los servidores de OpenStreetMap, personal humanitario que mapea las áreas afectadas por desastres y muchos más.

---

[1] http://wiki.ros.org/rqt

[2] https://doc-snapshots.qt.io/qt5-dev/qtgui-index.html

[3] http://wiki.ros.org/rviz

[4] https://github.com/gareth-cross/rviz_satellite

[5] https://www.openstreetmap.org/about

Finalmente comentar que ofrece datos abiertos: cualquiera es libre de usarlos para cualquier propósito siempre y cuando acredite a OpenStreetMap y a sus colaboradores.

5.2.2 Diseño general de la GUI

En la Figura 16 se puede ver una captura de la ventana principal de la interfaz gráfica. El modelo diseñado contiene, a la izquierda, un plugin personalizado que permite el acceso a todas las funciones de la GCS, además del control y monitorización de cada UAV de forma independiente; a la derecha, un visualizador de rviz en el que se mostrarán las misiones una vez generadas.

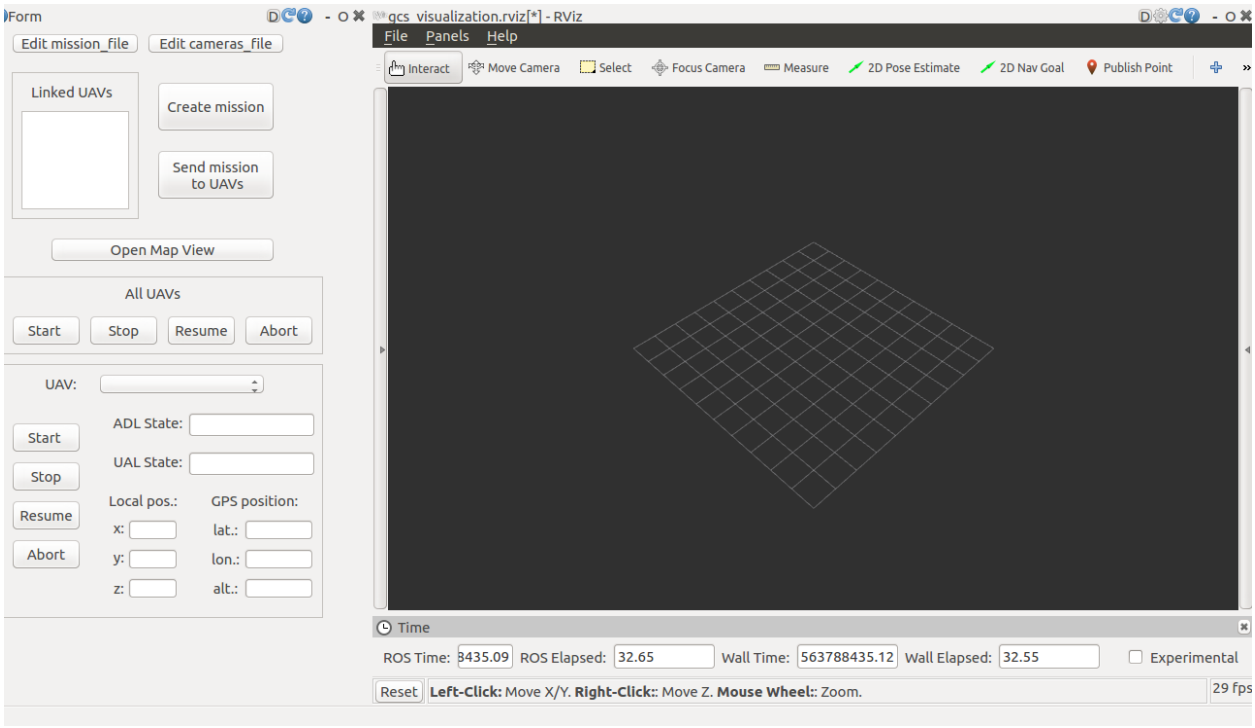

Figura 16. Captura de la GUI diseñada para la GCS en la que aparecen todos los botones disponibles.

5.2.3 Breve manual de usuario de la interfaz

A continuación, se describe brevemente el funcionamiento de la interfaz:

- En la parte superior, los botones "Edit mission_file" y "Edit cameras_file" abren respectivamente los archivos mission_file.json y cameras_file.json con el editor de texto gedit, lo que permite visualizar y modificar su contenido. El primero contiene los parámetros de vuelo que definen la misión. El segundo, también necesario para generar la misión, contiene una lista de las cámaras disponibles y las características de la lente de cada una de ellas.
- En el recuadro con la etiqueta "Linked UAVs" aparece la lista de los UAVs que ya han establecido conexión con la GCS. Al lado, se encuentran los botones "Create mission" y "Send mission to UAVs", que directamente llaman a los servicios del nodo principal de la GCS que permiten, respectivamente, generar la misión para los UAVs de la lista y enviarla a estos. Al lanzar el programa, el botón "Create mission" estará oculto hasta que aparezca al menos un elemento en la

lista de UAVs. De la misma forma, el botón “Send mission to UAVs” estará oculto hasta que se haya generado la misión.

- El botón “Open Map View” abre una ventana de rviz en la que se muestran las trayectorias que seguirán los UAVs (igual a las que se muestran en el plugin de rviz dentro de la ventana de rqt) sobre un mapa de la zona obtenido de OpenStreetMap (ver Figura 19). Se ha diseñado de esta forma debido a que la herramienta *rviz_satellite* no es compatible con el plugin de rviz para rqt, por lo que es necesario abrir una nueva ventana de rviz.
- En el recuadro con la etiqueta “All UAVs” están los botones: “Start”, “Stop”, “Resume” y “Abort”. Estos enviarán la orden correspondiente a toda la flota de UAVs. Cada uno de estos botones se mostrará disponible solo cuando el servicio correspondiente esté disponible para todos los UAVs de la lista. Por ejemplo, el botón “Stop” solo estará disponible si todos los UAVs se encuentran ejecutando la misión y siguen teniendo conexión con la GCS.
- En el último recuadro, aparece un desplegable que contiene la lista de UAVs conectados. Permite el control de cada uno de ellos de forma independiente mediante los mismos botones. De la misma forma que los anteriores, estos solo se mostrarán cuando el servicio correspondiente esté disponible para el UAV seleccionado. Aquí también se podrá visualizar el estado y telemetría del UAV seleccionado.

La Figura 17 muestra una captura de la ventana principal de la interfaz gráfica en un instante en el que se ha generado una misión, pero aún no se ha enviado a los UAVs. Se puede ver cómo, bajo la etiqueta “Linked UAVs”, aparece la lista de los UAVs conectados y, a la derecha, una visualización de la misión generada para los tres UAVs. Se observa como aparece visible el botón “Send mission” pero no aparecen ninguno de los botones para el control de los UAVs ya que, al no haber recibido la misión, todavía no se encuentran disponibles los servicios correspondientes.

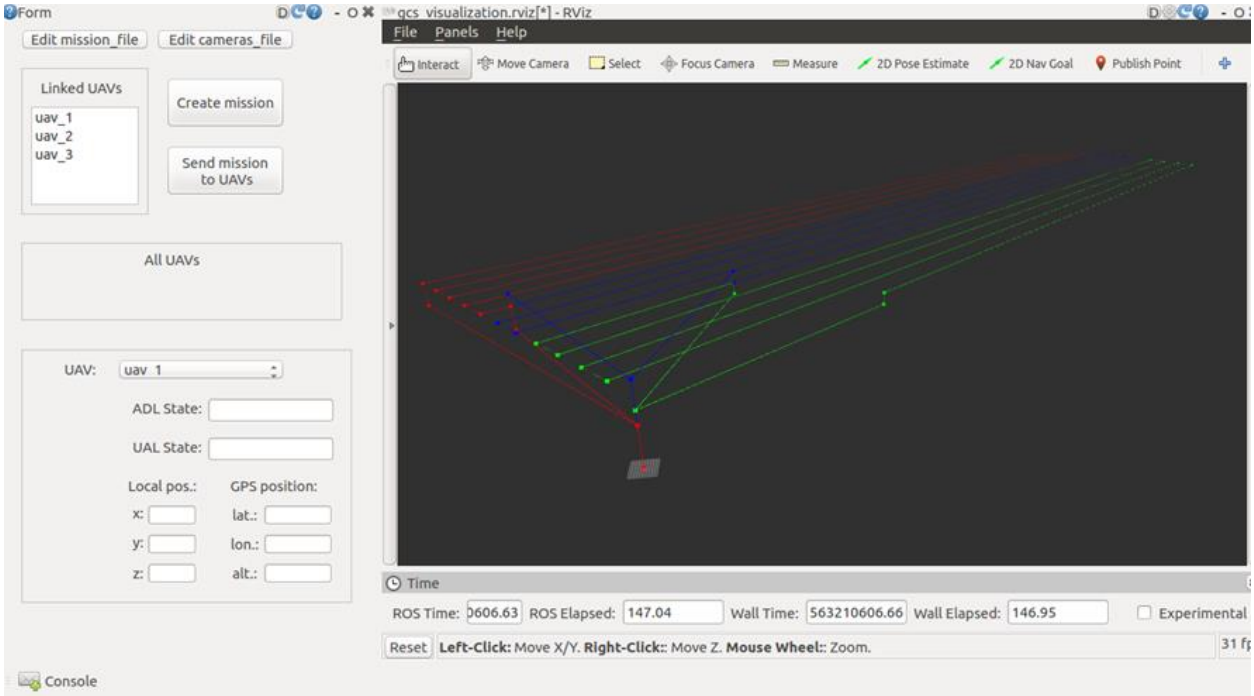

Figura 17. Captura de la GUI tras haber generado una misión para 3 UAVs, donde se pueden apreciar las distintas

altitudes de vuelo.

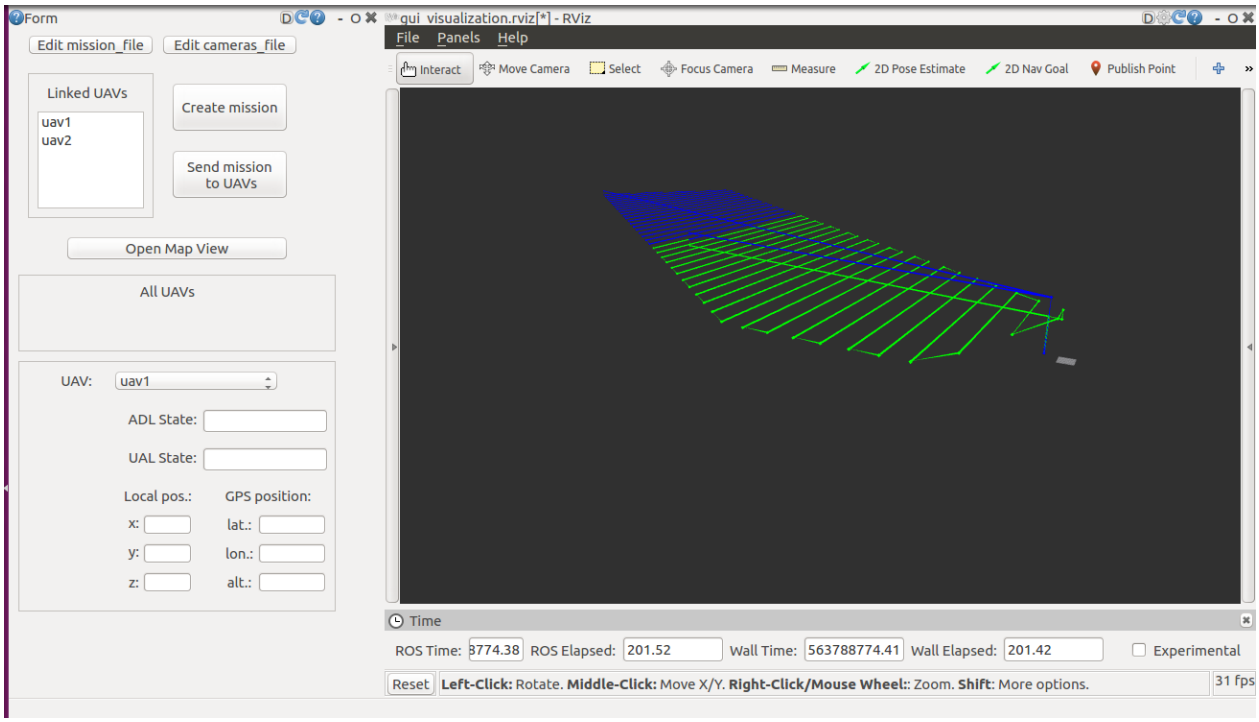

Figura 18. Captura de la GUI tras haber generado una misión para 2 UAVs utilizando las coordenadas de la planta de TSK junto a "El Torbiscal" en Utrera y una dirección de vuelo de 90º con respecto al norte.

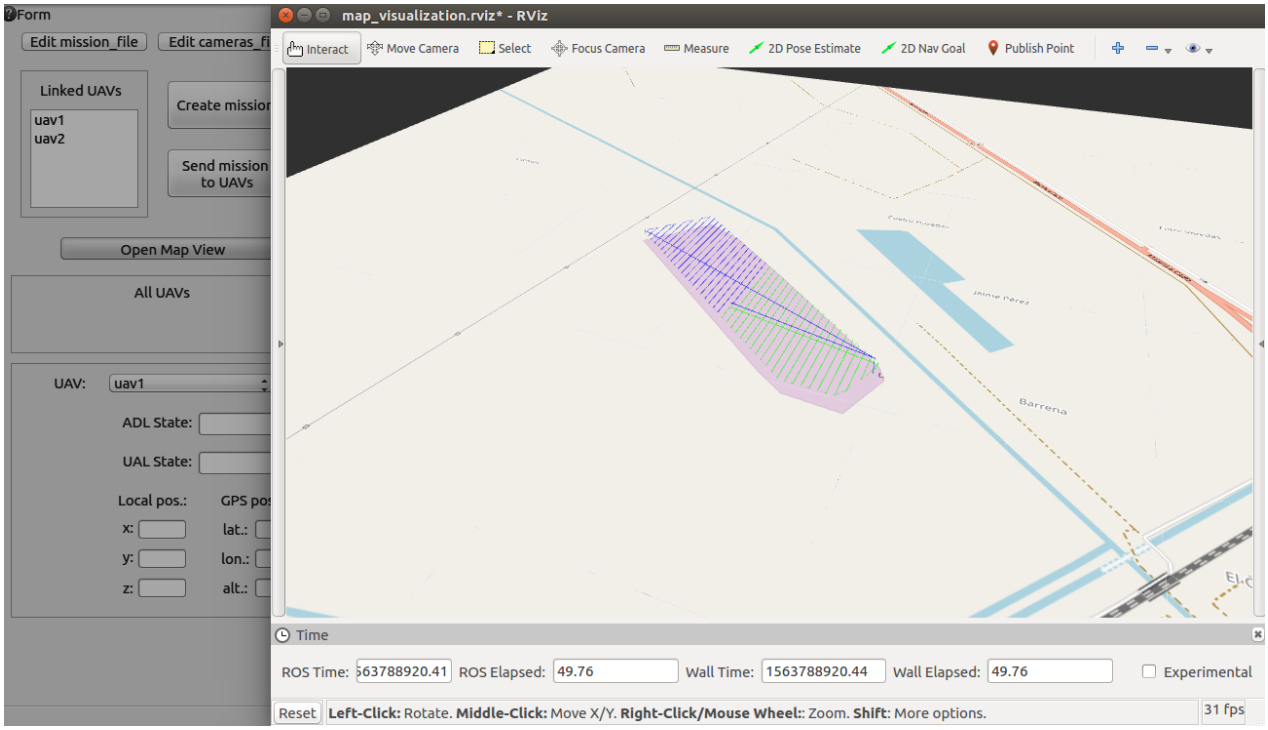

Figura 19. Visualización de la misión sobre el mapa de la zona. Se corresponde a la misma misión mostrada en la Figura 18.

5.3 Integración del Servicio API Rest

Se ha investigado sobre la mejor forma de integrar el software que se ha desarrollado para la GCS con un servicio API REST que permita su manejo remoto.

Se ha concluido que la mejor forma de integrarlo con la arquitectura software que se ha desarrollado, basada en ROS, es mediante ROStful[6], una herramienta que permite tener disponibles servicios y 'topics' de ROS a través de la web mediante REST, haciéndolo compatible con cualquier interfaz desarrollada para la parte del "frontend" que utilice estos mismos mensajes predefinidos.

ROStful también proporciona una interfaz muy simple para el "frontend" que permitiría hacer algunas pruebas (ver Figura 20)

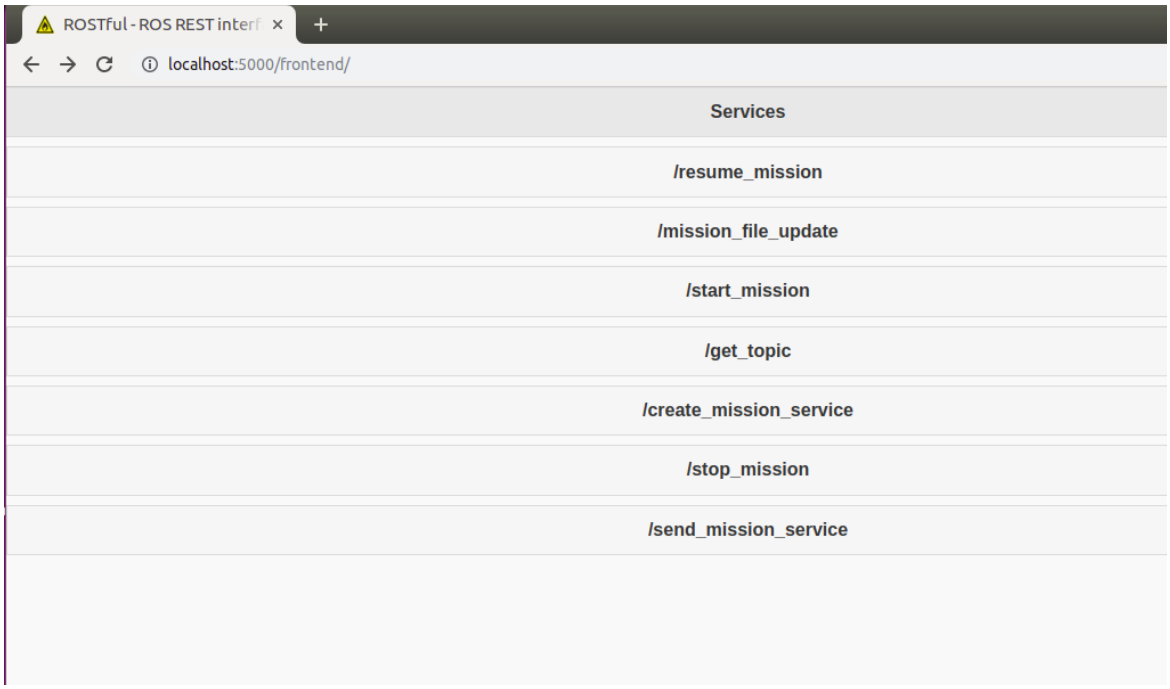

Figura 20. Captura del la interfaz "frontend" de ROStful donde aparecen todos los servicios disponibles a traves de la API Rest

5.3.1 Consulta de 'topics' de ROS a través de API Rest

A través de la API Rest se puede consultar la información ofrecida por cualquiera de los 'topics' de ROS que se encuentran disponibles en ese momento en cada uno de los UAVs que tengan conexión con la GCS.

Por un lado, se tienen los siguientes 'topics' procedentes del autopiloto A3 de DJI:

- */dji_sdk/acceleration_ground_fused*: Aceleración fusionada, con respecto al marco de referencia ENU ("East North Up").

[6] https://rostful.readthedocs.io/en/latest/

- */dji_sdk/angular_velocity_fused:* Velocidad angular fusionada (p,q,r), en referencias FLU ("Front Left Up") con respecto al cuerpo del UAV.
- */dji_sdk/attitude:* Actitud del vehículo representada en cuaternios para la rotación del cuerpo en FLU con respecto al marco de referencia ENU.
- */dji_sdk/display_mode:* Estado detallado del UAV. El tipo de mensaje que publica es un entero que se corresponde con uno de los posibles estados del autopiloto. A continuación, se muestran estos estados y su equivalente entero:

  MODE_MANUAL_CTRL = 0,

  MODE_ATTITUDE = 1,

  MODE_P_GPS = 6,

  MODE_HOTPOINT_MODE = 9,

  MODE_ASSISTED_TAKEOFF = 10,

  MODE_AUTO_TAKEOFF = 11,

  MODE_AUTO_LANDING = 12,

  MODE_NAVI_GO_HOME = 15,

  MODE_NAVI_SDK_CTRL= 17,

  MODE_FORCE_AUTO_LANDING = 33,

  MODE_SEARCH_MODE = 40,

  MODE_ENGINE_START = 41

- */dji_sdk/flight_status:* Estado simplificado del UAV. Publica un entero (0, 1 ó 2) que se corresponde con un posible estado de vuelo. A continuación, se muestran estos estados y su equivalente entero:

  STOPPED = 0,

  ON_GROUND = 1,

  IN_AIR = 2

- */dji_sdk/gps_health:* Calidad de la señal del GPS. El dato es un entero que varía entre 1 y 5, donde 5 es la mejor condición.
- */dji_sdk/gps_position*: Fusiona la posición global del UAV en latitud, longitud y altitud (en metros). Este valor de altitud es siempre con respecto al punto de despegue del UAV, en el que el autopiloto establece su referencia de posición local en coordenadas cartesianas.
- */dji_sdk/imu:* Datos de la IMU incluyendo lectura de giróscopo, acelerómetro y estimación de la actitud en referencia FLU con respecto al cuerpo del UAV.
- */dji_sdk/velocity:* Velocidad lineal en referencias ENU. Solamente está disponible cuando la calidad de la señal GPS es mayor o igual a 3.
- */dji_sdk/height_above_takeoff:* Altura sobre el punto de despegue.
- */dji_sdk/local_position:* Posición local en coordenadas cartesianas (en metros) con marco de referencia ENU, con respecto al punto de referencia de coordenadas locales (se fija como origen aquel desde el que despega el UAV).

También se puede obtener la lectura del altímetro láser mediante el siguiente "topic":

- */laser_altitude:* Altitud medida por el altímetro láser (en metros y con una precisión de 2 cifras decimales)

Además, se dispone de los siguientes "topics" proporcionados por la capa UAL desarrollada:

- */ual/state:* Estado del UAV a nivel de la capa UAL. El mensaje que se publica es un entero que se corresponde con uno de los posibles estados de UAL:

  UNINITIALIZED = 0,

  LANDED_DISARMED = 1,

  LANDED_ARMED = 2,

  TAKING_OFF = 3,

  FLYING_AUTO = 4,

  FLYING_MANUAL = 5,

  LA

  NDING=6

- */ual/pose:* Posición en coordenadas locales. Combina la información de los topics *'/dji_sdk/local_position'* y *'/dji_sdk/attitude'* en un único mensaje.
- */ual/velocity:* Publica la velocidad linear y angular del UAV, procedente de los topics: *'/dji_sdk/velocity'* y *'/dji_sdk/angular_velocity_fused',* en un único mensaje.

Por último, se dispone del siguiente "topic" publicado por el nodo principal de la GCS:

- */uav_list:* Devuelve la lista de identificadores de los UAVs cuyas NUCs tienen conexión en ese momento con la GCS y tienen operativo su autopiloto A3, así como las cámaras WIRIS y Sony.

Se ha creado un servicio, llamado *'/get_topic'* que devuelve el último mensaje publicado por cualquiera de estos "topics". Basta con usar el método POST más el nombre de dicho servicio, pasando como contenido del mensaje el identificador del UAV seguido de una barra (/) y del nombre del topic. Por ejemplo, para obtener la actitud, suponiendo que el identificador del UAV fuese "*uav_id", la petición sería*:

POST /get_topic'

{

" */uav_id/dji_sdk/attitude"*

}

En la Figura 21 se puede ver un ejemplo del uso de este servicio mediante la interfaz para el "frontend" proporcionada por ROStful.

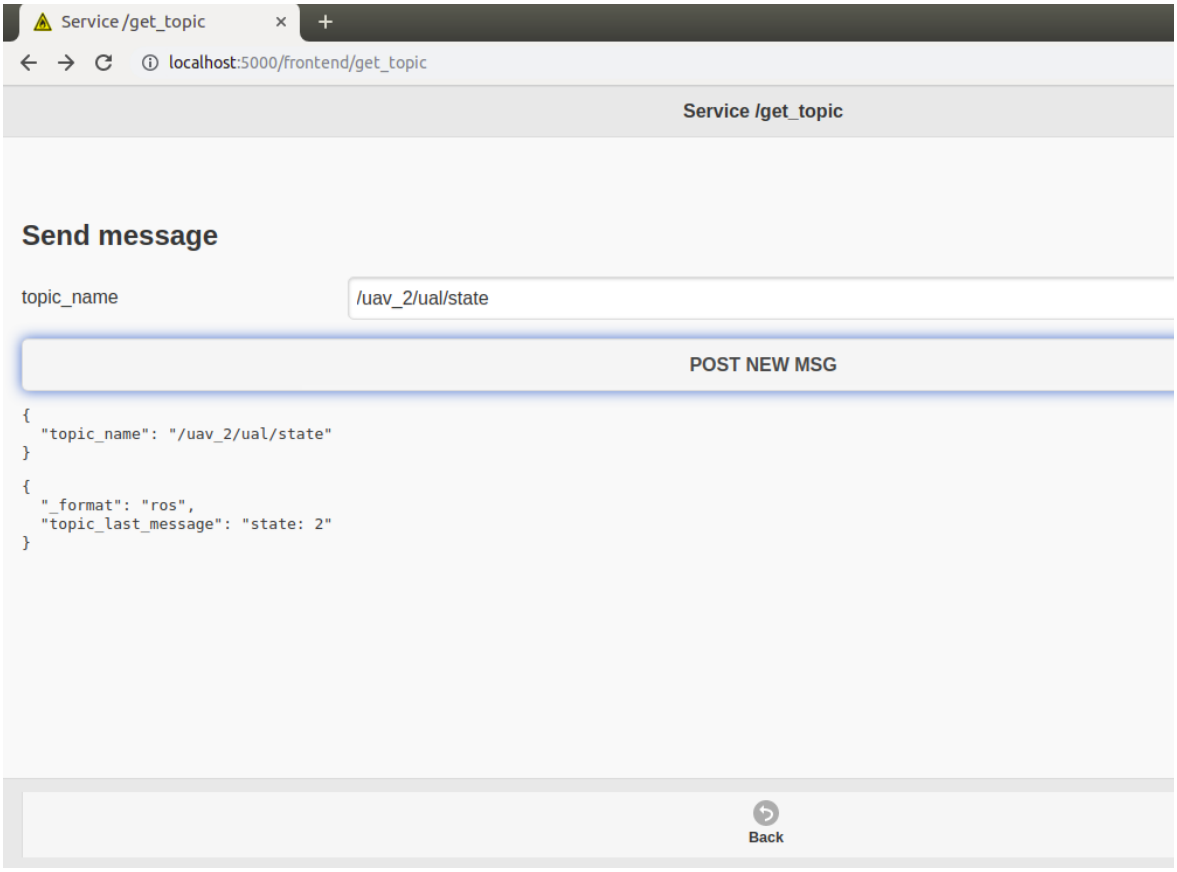

Figura 21. Ejemplo de uso del servicio *'/get_topic'* a través de API Rest para obtener el ultimo mensaje publicado por el "topic" *'/ual/state'* del UAV con identificador *'uav_2'*.

5.3.2 Uso de servicios de ROS a través de API Rest

Se puede llamar directamente a los siguientes servicios de ROS a través de la API Rest, mediante el método POST:

- */mission_file*: A través de este servicio se envia desde el servidor TSK a la GCS el contenido en formato json con los parámetros de la misión (archivo de vuelo).
- /create_mission: El software instalado en la GCS elabora la misión de vuelo en base al número de UAVs conectados en ese momento y de los datos recibidos en el archivo de vuelo.
- */send_mission_to_uavs*: La GCS envía la misión previamente creada a los UAVs. Cada uno recibirá una lista de waypoints (path) y una altitud distinta de seguridad (para el vuelo entre la estación base y su zona de barrido).

Por ejemplo, para actualizar el contenido del archivo de vuelo se utilizaría la siguiente petición:

'POST /mission_file'

{

“contenido del archivo de vuelo”

}

Por otro lado, a continuación, se listan otros servicios que dan la opción de ejecutar la orden en todos los UAVs a la vez o solo en algunos en concreto:

- */start_mission*: da a los UAVs la orden de comenzar la misión.
- */stop_mission*: detiene la misión y cada UAV guarda el punto donde se encontraba para posteriormente poder reanudarla con el servicio *resume_mission*.
- */resume_mission*: reanuda la misión en el caso de que haya sido detenida previamente con el servicio */stop_mission*.
- */abort_mission*: aborta la misión en el caso de que haya sido detenida previamente con el servicio */stop_mission*. Los UAVs seleccionados pasarían a la espera de recibir una nueva misión.

Para esto se les pasa un mensaje, en formato json, con el parámetro "uavs" que podrá ser: "all" (en el caso de querer enviar la orden a todos los UAVs) o una lista con los identificadores de los UAVs seleccionados. Por ejemplo, para dar la orden de detener la misión a los UAVs: uav1 y uav2, se utilizaría la petición:

‘POST /stop_mission’

{

[uav1, uav2]

}

5.4 Volcado en base de datos

Para esta parte se dispone de una aplicación proporcionada por TSK (denominada “pusher”) que se ejecuta en la GCS y se encarga de enviar a los servidores de TSK todos los datos de los sensores abordo (imágenes obtenidas por las cámaras durante la misión y datos de telemetría obtenidos del autopiloto y el altímetro laser) (ver Figura 22).

Los datos de telemetría recogidos de los sensores se almacenarán, una vez descargados en la GCS, en una base de datos tipo SQL. Se ha desarrollado un script en Python que se encarga de recoger estos datos, que durante la misión se habrán ido almacenando en un fichero csv, y pasarlos a la base de datos con el formato deseado junto con su respectivo “timestamp” y el “uuid” establecido en el archivo de misión.

En la Figura 23 se puede ver una captura de la base de datos generada a partir de datos de telemetría recogidos durante uno de los vuelos de prueba. Se han incluido, como primera aproximación, la latitud y la longitud, obtenidas del “topic” del autopiloto A3 de DJI *‘/dji_sdk/gps_position’*. Los “uuids” se han extraído del último modelo del archivo *archivo-mision.json*.

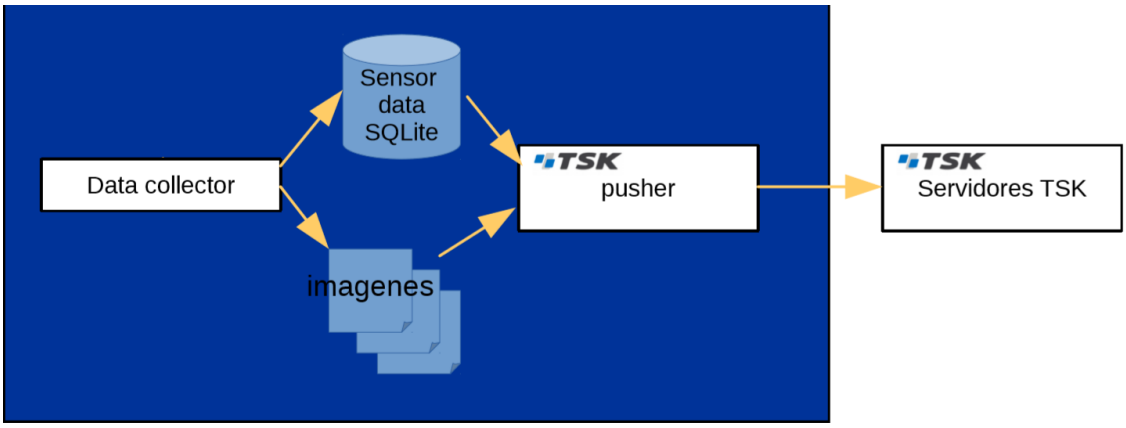

Figura 22. Esquema de la arquitectura encargada del envió de datos desde la GCS a los servidores remotos de TSK

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: sensor_entries — New Record | Delete Record

	id	uuid	timestamp	value
	Filter	Filter	Filter	Filter
1	1	d3b3d0e8-afe3-4630-986c-b8e1cf99b0eb	1.56041970336318e+18	37.0914301512
2	2	e7dc9cae-fce9-4277-b9c7-e0676df7de69	1.56041970336318e+18	-5.8729659628
3	3	d3b3d0e8-afe3-4630-986c-b8e1cf99b0eb	1.56041998868368e+18	37.0910365221
4	4	e7dc9cae-fce9-4277-b9c7-e0676df7de69	1.56041998868368e+18	-5.8722819896
5	5	d3b3d0e8-afe3-4630-986c-b8e1cf99b0eb	1.56041999160983e+18	37.0910492198
6	6	e7dc9cae-fce9-4277-b9c7-e0676df7de69	1.56041999160983e+18	-5.872304435
7	7	d3b3d0e8-afe3-4630-986c-b8e1cf99b0eb	1.56041999448078e+18	37.0910792223
8	8	e7dc9cae-fce9-4277-b9c7-e0676df7de69	1.56041999448078e+18	-5.8723565945
9	9	d3b3d0e8-afe3-4630-986c-b8e1cf99b0eb	1.56041999744015e+18	37.0911209622
10	10	e7dc9cae-fce9-4277-b9c7-e0676df7de69	1.56041999744015e+18	-5.8724299561
11	11	d3b3d0e8-afe3-4630-986c-b8e1cf99b0eb	1.560419999577e+18	37.0911539903
12	12	e7dc9cae-fce9-4277-b9c7-e0676df7de69	1.560419999577e+18	-5.8724879117
13	13	d3b3d0e8-afe3-4630-986c-b8e1cf99b0eb	1.56042000162854e+18	37.0911832963
14	14	e7dc9cae-fce9-4277-b9c7-e0676df7de69	1.56042000162854e+18	-5.8725412317
15	15	d3b3d0e8-afe3-4630-986c-b8e1cf99b0eb	1.56042000354676e+18	37.0912097652
16	16	e7dc9cae-fce9-4277-b9c7-e0676df7de69	1.56042000354676e+18	-5.8725902127
17	17	d3b3d0e8-afe3-4630-986c-b8e1cf99b0eb	1.56042000569175e+18	37.0912399284
18	18	e7dc9cae-fce9-4277-b9c7-e0676df7de69	1.56042000569175e+18	-5.8726431602
19	19	d3b3d0e8-afe3-4630-986c-b8e1cf99b0eb	1.56042000790844e+18	37.0912718125
20	20	e7dc9cae-fce9-4277-b9c7-e0676df7de69	1.56042000790844e+18	-5.8726927577
21	21	d3b3d0e8-afe3-4630-986c-b8e1cf99b0eb	1.56042001014761e+18	37.0913057244
22	22	e7dc9cae-fce9-4277-b9c7-e0676df7de69	1.56042001014761e+18	-5.8727457827
23	23	d3b3d0e8-afe3-4630-986c-b8e1cf99b0eb	1.56042001246457e+18	37.0913383962

|< < 1 - 23 of 1610 > >| Go to: 1

Figura 23. Base de datos tipo SQL generada con los datos de latitud y longitud obtenidos del autopiloto A3 de DJI durante uno de los vuelos de prueba

6 RESULTADOS EXPERIMENTALES

En esta sección se describen simulaciones y experimentos reales que muestran las principales características del sistema desarrollado. Se pueden encontrar varios videos de estos experimentos en el siguiente enlace: https://grvc.us.es/reduas19pv

En primer lugar, se han realizado varias simulaciones utilizando utilizando el simulador de DJI conectado al autopiloto A3 y también con autopilotos de PX4 en modo SITL (Software in the Loop).

Tras verificar que todo funcionaba correctamente en las simulaciones, se ha realizado un experimento de vuelo real en una planta fotovoltaica situada en Utrera (España). Como solo se disponía de un UAV equipado con todas las cámaras, se ha definido una misión en la que se utiliza un UAV real y dos simulados. El objetivo de este experimento fue obtener un conjunto de imágenes visuales y termográficas georeferenciadas de la planta solar para generar un mosaico que sirva para localizar posibles fallos en los paneles a partir del análisis de las imágenes térmicas.

El UAV utilizado para el experimento es un DJI Matrice 600, que integra un autopiloto DJI A3. Este dispositivo tiene un control de altitud preciso basado en GNSS y un barómetro incorporado; sin embargo, la inspección de la planta requiere un control mas preciso de la altitud sobre el nivel del suelo para mantener la misma distancia entre las cámaras y los paneles que se van a inspecciona. Para esto, la plataforma ha sido equipada con un altímetro láser "Lightware sf11/c" que hace posible el vuelo AGL.

A bordo del UAV se encuentran: una cámara tipo webcam, que proporciona imágenes en directo durante la ejecución de la inspección, y las cámaras RGB y térmicas presentadas en el capítulo 4 de este documento.

El experimento consistió en una misión de inspección de toda el área de la planta, volando a 30 metros AGL, con dirección Este-Oest. Esto ha permitido obtener un conjunto de datos de imágenes de la planta para su análisis. Además, se han podido probar distintas funcionalidades del sistema como: la parada manual y reanudación de la mision, y la parada automática cuando las baterías se encuentran por debajo del nivel de seguridad.

En la Figura 24 se puede ver una captura del video grabado durante el vuelo. La Figura 25 muestra dos capturas tomadas de la aplicación DjiGo durante la misión. Esta aplicación está vinculada al piloto automático Dji A3 del UAV y registra, además de otra información, la ruta seguida por la UAS. La primera captura se realiza cuando el servicio de parada se llama manualmente, mientras que la segunda corresponde al momento en que el UAS detiene la misión debido al bajo nivel de batería y regresa al punto de despegue.

La Figura 26 muestra dos imágenes de la cámara RGB tomadas durante el vuelo.

Figura 24. Captura del video que muestra el experimento real en la planta fotovoltaica cercana a Utrera. Disponible en el enlace https://grvc.us.es/reduas19pv#experiment

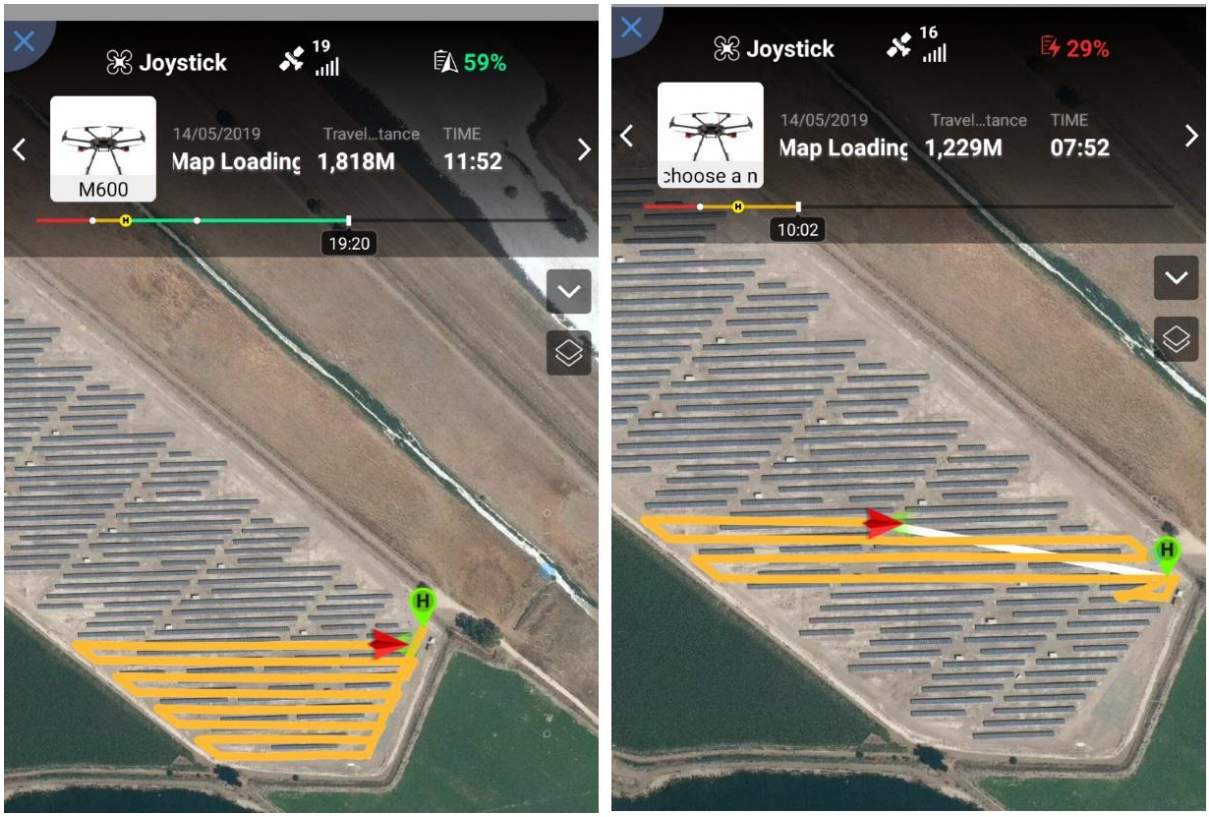

Figura 25. Capturas tomadas de la aplicación Dji Go durante el experimento donde se muestra la trayectoria seguida por el UAV de DJI.

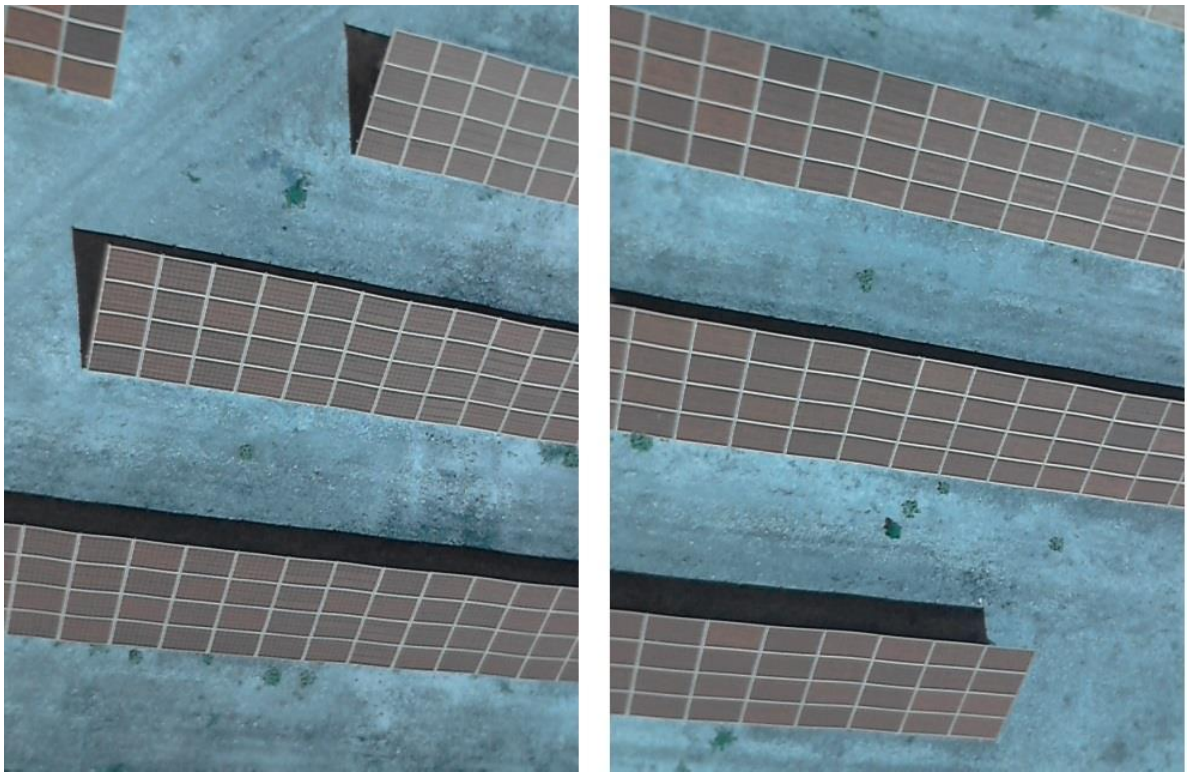

Figura 26. Imágenes tomadas por la cámara RGB durante el vuelo de inspección.

7 CONCLUSIONES Y DESARROLLOS FUTUROS

En este documento se ha presentado la solución diseñada para cumplir con una de las líneas del proyecto INSPECTOR, la correspondiente a la inspección desatendida de plantas fotovoltaicas por una flota de UAVs.

En primer lugar, se ha hecho una ampliación de la capa UAL, que en principio solo era compatible con autopilotos que soportan el protocolo mavlink, para hacerla también compatible con los de DJI; ya que el proyecto requería utilizar estos autopilotos.

Una vez solucionado el problema de la navegación autónoma con autopilotos de DJI, se pasó a desarrollar el software que correría en la GCS, encargado de la generación de misiones de vuelo y el control y monitorización de la flota de UAVs.

Porteriormente, se desarrollaron las capas ADL y PAL. La capa ADL se basa en una máquina de estado que corre en el ordenador de a bordo de los UAVs y se encargá del desarrollo de la misión, comunicándose con UAL y PAL que a su vez se comunican, respectivamente, con el autopiloto y con los demás sensores de a bordo.

Para culminar el proyecto se ha creado una interfaz gráfica de usuario que facilita el control de todo el sistema desde la GCS, permitiendo su uso por el personal de mantenimiento de la planta, y que, además, permite la monitorización en tiempo real de la ejecución de la misión. Adiccionalmente, se ha añadido un servicio que permite hacer todo esto de forma remota a través de una API Rest.

Toda esta implementación se puede encontrar en los siguientes repositorios de GitHub. En el primero se encuentra el software desarrollado para la GCS, mientras que el segundo contiene todo el código que iría a bordo de cada UAV:

https://github.com/AlejandroCastillejo/inspector_gcs

https://github.com/AlejandroCastillejo/inspector_software_uav

Aunque no corresponda a esta parte del proyecto, como futuro desarrollo estaría bien crear una aplicación web para el lado del front-end de la API Rest que haga más cómodo e ilustrativo el uso remoto del sistema.

REFERENCIAS

[1] J. A. Millan-Romera, H. Perez-Leon, A. Castillejo-Calle, I. Maza, andA. Ollero, "ROS-MAGNA, a ROS-based framework for the definitionand management of multi-UAS cooperative missions," inProc. ofthe International Conference on Unmanned Aircraft Systems (ICUAS).IEEE, June 2019, pp. 1–10.

[2] N. Basilico and S. Carpin, "Deploying teams of heterogeneous UAVs in cooperative two-level surveillance missions," in2015IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).IEEE, sep 2015, pp. 610–615. [Online]. Available: http://ieeexplore.ieee.org/document/7353435/

[3] J. J. Acevedo, B. C. Arrue, I. Maza, and A. Ollero, "Adecentralized algorithm for area surveillance missions using ateam of aerial robots with different sensing capabilities," in Proceedings of the IEEE International Conference on Roboticsand Automation, May 2014, pp. 4735–4740. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2014.6907552

[4] F. Balampanis, I. Maza, and A. Ollero, "Coastal areas divisionand coverage with multiple UAVs for remote sensing, "Sensors, vol. 17, no. 4, pp. 808–832, 2017. [Online]. Available: http://dx.doi.org/10.3390/s17040808

[5] K. Kondak, A. Ollero, I. Maza, K. Krieger, A. Albu-Schaeffer, M. Schwarzbach, and M. Laiacker, Unmanned Aerial Systems Physically Interacting with the Environment: Load Transportation, Deployment, and Aerial Manipulation. Springer Netherlands, 2015, pp. 2755–2785. [Online]. Available: http://dx.doi.org/10.1007/978-90-481-9707-177

[6] P. J. Sanchez-Cuevas, P. Ramon-Soria, B. Arrue, A. Ollero, and G. Heredia, "Robotic system for inspection by contact of bridgebeams using uavs, "Sensors, vol. 19, no. 2, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/2/305

[7] L. Yoder and S. Scherer, "Autonomous Exploration for Infrastructure. Modeling with a Micro Aerial Vehicle," in Springer Tracts in Ad-vanced Robotics, ser Springer Tracts in Advanced Robotics, D. S. Wettergreen and T. D. Barfoot, Eds. Cham: Springer International Publishing, 2016, vol. 113, pp. 427–440.

[8] M. Aghaei, F. Grimaccia, C. Gonano, and S. Leva, "Innovative automated control system for pv fields inspection and remote control,"IEEE Transactions on Industrial Electronics, vol. 62, no. 11, pp. 7287–7296, 2015, cited By 36. [Online].Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84944128471&doi=10.1109%2fTIE.2015.2475235&partnerID=40&md5=d61eea91ab90e2d7795e002eccab63a3

[9] J.Tsanakas,L.Ha,andF.AlShakarchi,"Advancedinspection of photovoltaic installations by aerial triangulation andterrestrial georeferencing of thermal/visual imagery,"RenewableEnergy, vol. 102, pp. 224–233, 2017, cited By 17. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84994314516&doi=10.1016%2fj.renene.2016.10.046&partnerID=40&md5=c06ae60131e27d72266b6d737f5682a6

[10] S. Dotenco, M. Dalsass, L. Winkler, T. Wurzner, C. Brabec, A. Maier, and F. Gallwitz, "Automatic detection and analysis of photovoltaic modules in aerial infrared imagery," in Proc. of the 2016IEEE Winter Conference on Applications of Computer Vision (WACV2016).Institute of Electrical and Electronics Engineers Inc., 2016, cited By 15. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84977604150&doi=10.1109%2fWACV.2016.7477658&partnerID=40&md5=cf47152bdcd8d9f62d20777761c8b80c

[11] F. Real, A. Torres-Gonzalez, P. Ramon-Soria, J. Capitan, and A. Ollero, "UAL: an abstraction layer for unmanned vehicles," in2ndInternational Symposium on Aerial Robotics (ISAR), 2018.

[12] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot OperatingSystem," inICRA Workshop on Open Source Software, 2009.

[13] I. Maza, F. Caballero, J. Capitan, J. M. de Dios, and A. Ollero,"A distributed architecture for a robotic platform with aerial sensortransportation and self-deployment capabilities,"Journal of FieldRobotics, vol. 28, no. 3, pp. 303–328, 2011. [Online]. Available: http://dx.doi.org/10.1002/rob.20383

[14] "SMACH - a task-level architecture for rapidly creating complex robotbehavior in ROS," 2019, http://wiki.ros.org/smach, Last accessed on 2019-02-26.

[15] H. Perez-Leon, J. J. Acevedo, J. A. Millan-Romera, A. Castillejo-Calle, I. Maza, and A. Ollero, "An aerial robot path follower based on the 'carrot chasing' algorithm," inFourth Iberian Robotics Conference,2019