ULPGC Universidad de Las Palmas de Gran Canaria | Instituto Universitario de Microelectrónica Aplicada IUMA

Máster Universitario en Electrónica y Telecomunicación Aplicadas (META)

Trabajo Fin de Máster

Diseño de un sistema de compresión y transmisión en tiempo real de imágenes hiperespectrales tomadas desde plataformas de vuelo no tripuladas

Autor: Adán Enrique Jiménez Delgado
Tutor(es): José Fco. López Feliciano
Raúl Guerra Hernández
María Díaz Martín
Fecha: Julio de 2020

IUMA
t +34 928 451 150
+34 928 451 086
f +34 928 451 083
e: iuma@iuma.ulpgc.es
w: www.iuma.ulpgc.es
Campus Universitario de Tafira
35017 Las Palmas de Gran Canaria

Versión: 1 – 20 julio 2020

ULPGC Universidad de Las Palmas de Gran Canaria | Instituto Universitario de Microelectrónica Aplicada IUMA

Máster Universitario en Electrónica y Telecomunicación Aplicadas (META)

Trabajo Fin de Máster

Diseño de un sistema de compresión y transmisión en tiempo real de imágenes hiperespectrales tomadas desde plataformas de vuelo no tripuladas

HOJA DE FIRMAS

Alumno/a:	Adán Enrique Jiménez Delgado	Fdo.:	JIMENEZ DELGADO ADAN ENRIQUE - 45340391P. Firmado digitalmente por JIMENEZ DELGADO ADAN ENRIQUE - 45340391P Fecha: 2020.07.20 17:02:44 +01'00'
Tutor/a:	José Fco. López Feliciano	Fdo.:	LOPEZ FELICIANO JOSE FRANCISCO - 42168444Z. Firmado digitalmente por LOPEZ FELICIANO JOSE FRANCISCO - 42168444Z Fecha: 2020.07.28 11:42:09 +01'00'
Tutor/a:	María Díaz Martín	Fdo.:	DIAZ MARTIN MARIA - 78585289Q. Firmado digitalmente por DIAZ MARTIN MARIA - 78585289Q Fecha: 2020.07.20 17:46:21 +01'00'
Tutor/a:	Raúl Guerra Hernández	Fdo.:	GUERRA HERNANDEZ RAUL CELESTINO - 78518672F. Firmado digitalmente por GUERRA HERNANDEZ RAUL CELESTINO - 78518672F Fecha: 2020.07.20 17:47:25 +01'00'

Fecha: Julio de 2020

IUMA
t +34 928 451 150
+34 928 451 086
f +34 928 451 083
e: iuma@iuma.ulpgc.es
w: www.iuma.ulpgc.es
Campus Universitario de Tafira
35017 Las Palmas de Gran Canaria

Versión: 1 – 20 julio 2020

ULPGC
Universidad de Las Palmas de Gran Canaria

Instituto Universitario de Microelectrónica Aplicada

IUMA

Máster Universitario en Electrónica y Telecomunicación Aplicadas (META)

Trabajo Fin de Máster

Diseño de un sistema de compresión y transmisión en tiempo real de imágenes hiperespectrales tomadas desde plataformas de vuelo no tripuladas

HOJA DE EVALUACIÓN

Calificación: .....

Presidente	Pablo Hernández Morera	Fdo.:
Secretario	Sebastián López Suárez	Fdo.:
Vocal	Sunil Lalchand Khemchandani	Fdo.:

Fecha: Julio de 2020

IUMA
t +34 928 451 150
+34 928 451 086
f +34 928 451 083
e: iuma@iuma.ulpgc.es
w: www.iuma.ulpgc.es
Campus Universitario de Tafira
35017 Las Palmas de Gran Canaria

Versión: 1 – 20 julio 2020

Resumen

La teledetección ha sido un área de conocimiento de interés creciente en las últimas décadas. La posibilidad de estudiar los objetos desde la distancia es una actividad que reporta una gran cantidad de beneficios al tratarse de una acción poco invasiva. En concreto, si esta se combina con aeronaves no tripuladas (UAVs, de sus siglas en ingles *Unmanned Aerial Vehicle*), permite monitorizar grandes extensiones de terreno a un coste muy competitivo frente a otras alternativas. Además, en conjunto con la tecnología espectral, convierte a este trinomio en un reto tecnológico y de investigación que requiere de herramientas altamente sofisticadas para ponerlo en funcionamiento y extraer resultados satisfactorios. En particular, el Instituto Universitario de Microelectrónica Aplicada (IUMA), perteneciente a la Universidad de Las Palmas de Gran Canaria, realiza sendas investigaciones en esta campo, donde se cuenta con un gran conocimiento y aptitudes técnicas.

Este trabajo Fin de Máster se engloba dentro de este campo de conocimiento de interés científico y en él, se han abordado dos problemáticas existentes. Por un lado, se ha tratado la transmisión en tiempo real de las imágenes hiperespectrales tomadas desde las plataformas de vuelo de tal forma que la información captada durante las misiones se encuentre disponible en tiempo real para su estudio y análisis. Sin embargo, esta posibilidad no puede llegar a concretarse sin abordar antes la cuestión de la compresión de los datos, un factor crítico que permite, una vez es capturada la información, reducir el tamaño de esta para que pueda ser enviada a tierra, afrontando así las restricciones existentes en términos del ancho de banda del canal de comunicaciones.

Para ello, se ha diseñado un sistema de transmisión compuesto por una antena rotatoria que se encuentra apuntando al dron en todo momento durante el vuelo, de forma que centralizando las comunicaciones en un router, se establece un enlace de descarga de datos entre la estación de tierra y el UAV en vuelo. Además, para posibilitar dicha acción se ha establecido a bordo un sistema de compresión con pérdidas basado en el algoritmo del estado del arte HyperLCA.

Los resultados obtenidos demuestran el potencial de la solución desarrollada para la problemática descrita. Estos han permitido demostrar que, si bien es cierto que se

logra una compresión en tiempo real, la parte correspondiente a la transmisión de datos es mucho más crítica, lo que a su vez abre la posibilidad de crear una nueva línea de investigación en aras de mejorar los resultados obtenidos y alcanzar de forma óptima el objetivo final de la aplicación abordada.

Palabras clave: *teledetección, UAV, compresión de imágenes, transmisión de datos, tecnología espectral*

Abstract

Remote sensing has been an increasing area of interest for the research community in the last decades. It brings the possibility of studying objects from the distance, which provides a lot of benefits as it is a non-invasive action. Additionally, the emergence of lower-cost Earth observation platforms based on Unmanned Aerial Vehicles (UAVs) has allowed the inspection, surveillance and monitoring of large areas of land. All of this, jointly with the hyperspectral technology, emerges as a very attractive solution that also presents many challenges that require highly sophisticated software and hardware tools in order to extract satisfactory results. In this context, the Institute for Applied Microelectronics (IUMA), belonging to the University of Las Palmas de Gran Canaria, has an extensive experience since it has been researching in this field during the last years.

In particular, this Master Thesis has been oriented to resolve two existing problems around this area of interest. Firstly, the transmission in real-time of the hyperspectral information collected by the aerial observation platforms to the ground segment has been addressed. It permits to send the spectral data as soon as they are sensed in order to be analyzed before the flight mission is completed. However, this possibility cannot be realized without addressing the issue around data compression, a critical factor that allows, once the information is captured, to reduce the size of the information to be managed and thereby, to fulfill the requirements imposed by the restrictive bandwidth of the downlink systems.

On this basis, it has been developed in this work a transmission system composed of a rotatory antenna that permits continuously aiming at the UAV during the flight missions. Through the centralization of communications on a router, a data downlink system is established between the ground station and the flying platform. In order to enable this action, it has been also developed an on-board lossy compression solution based on the state-of-the-art algorithm named HyperLCA, paving the way to the real-time compression of the acquired hyperspectral data in order to overcome the existing bandwidth restrictions.

The obtained results show the potential of the proposed solutions for the issue to be dealt with in this work. They have also illustrated that thought real-time compression

is achieved, the transmission part of the problem is much more critical. Therefore, it gives rise to a new research line of interest in order to analyze the multiple features that could improve the obtained results and to achieve them in the best way the ultimate objective of the system.

Keywords: *remote sensing UAV, image compression, data transmission, hyperspectral technology.*

Índice general

1 Introducción **1**

2 Antecedentes **4**

2.1 Plataformas de vuelo no tripuladas 4

2.1.1 Aplicaciones de los UAV 5

2.2 Tecnología Espectral 6

2.3 Imágenes hiperespectrales 7

2.3.1 Tipos de cámaras espectrales 7

2.3.2 Clasificación y características de las cámaras hiperespectrales 9

2.4 Adquisición de datos con UAV 11

2.4.1 Plataforma de vuelo 11

2.4.2 Gimbal 13

2.4.3 Vuelos autónomos 14

2.4.4 Cámara hiperespectral incorporada 15

3 Compresión de imágenes **16**

3.1 Solución actual y justificación de compresión 16

3.2 Planteamiento de las necesidades de compresión 19

3.3 Algoritmo a emplear 19

3.4 Implementación y optimización del algoritmo ajustado a la solución actual 23

4 Diseño del sistema de comunicaciones **29**

4.1 Vía de comunicación empleada 29

4.1.1 Análisis de los estándares WiFi (802.11) existentes 30

4.1.2 Recepción de la señal 31

4.2 Descripción general del sistema de apuntamiento 32

4.3 Lógica de funcionamiento del sistema 36

4.4 Diseño estructural de la antena 38

4.5 Diseño mecatrónico de la antena 39

4.5.1 Alimentación del sistema en su conjunto 39

4.5.2 Motores paso a paso 42

4.5.3 Unidad de Medida Inercial MPU9250 44

4.5.4 Cálculos de orientación de la antena 45

4.5.5 Módulo de comunicación Bluetooth 61
4.5.6 Módulo de posicionamiento GPS 63
4.6 Interconexión del sistema completo 67
4.7 Lógica de comunicación entre la estación de tierra y el sistema de antenas 69

5 Resultados y líneas futuras **71**
5.1 Resultados del desarrollo del sistema de compresión 71
5.1.1 Análisis del envío de datos sin compresión 72
5.1.2 Resultados de la implementación del compresor en serie . . . 74
5.1.3 Implementación del sistema global de compresión 75
5.1.4 Líneas futuras 79
5.2 Resultados de la implementación del sistema de comunicaciones . . . 81
5.2.1 Análisis de diferentes antenas y su espectro de radiación . . . 81
5.2.2 Montaje físico de la estructura de la antena 86
5.2.3 Interconexión del sistema en su conjunto 87
5.2.4 Líneas futuras 89

6 Conclusiones **93**

Bibliografía **95**

1 Introducción

La vorágine tecnológica vivida en los últimos años ha ocasionado un cambio casi completo en la vida del ser humano. Ha facilitado la forma de vida de la sociedad hasta el punto de hacerla más sencilla, eficaz y productiva. En este contexto, el desarrollo de la ciencia y la tecnología ha permitido que aparezcan herramientas cada vez más potentes que otorgan un gran poder para comunicarse, para hacer negocios, para automatizar procesos mecánicos arduos y pesados, es decir, para mejorar prácticamente todos los aspectos de la vida cotidiana.

De igual modo, campos de estudio como son el Internet de las Cosas (IoT), la Inteligencia Artificial (IA), las redes de comunicación de quinta generación (5G), las ciudades inteligentes (*Smart Cities*) son representativos de la nueva revolución tecnológica que se está viviendo. Estos tienen como objetivo principal dar solución a problemas mediante la aplicación de la ciencia para satisfacer las necesidades o problemas de los usuarios de estas herramientas.

Un nexo común al desarrollo y despliegue de estas tecnologías es la adquisición de datos y su conversión en información. Estos han tomado cada vez un protagonismo mayor en la sociedad en la medida que llegan a ser el negocio de diversas y variopintas empresas; *saber es poder* y esto se ve reflejado en las técnicas de marketing que se nutren principalmente de esta información recogida por las corporaciones. Pero, antes de poder extraer conclusiones sobre ellos, se requiere de una serie de procesos complejos que dependerán, principalmente, de su fuente de extracción, pues no comprenderá el mismo proceso analizar información proveniente de una fuente de vídeo, un sistema electromecánico o una base de datos.

Asimismo, un área muy interesante para la captura de datos son las imágenes. Estas aportan información con muchísima relevancia, principalmente debido a que una imagen es una muestra muy fiel a la percepción del ojo humano de la escena capturada. Concretamente, las personas pueden ver solo en el espectro visible, siendo incapaces de identificar colores más allá del azul, verde, rojo o una mezcla entre ellos. Cada uno de estos colores representa una longitud de onda específica. No obstante, existen otras muchas frecuencias que el humano es incapaz de detectar con sus sentidos. A esta problemática da solución la tecnología espectral, que permite la captura de imágenes desde los 400 nm (color azul) hasta los 15000 nm. Las

bondades que esto aporta son muchísimas puesto que permite obtener en la captura de una escena una alta cantidad de información, permitiendo conocer el objeto bajo estudio con mayor exactitud.

De ello se derivan cantidad de aplicaciones como puede ser en el campo de la medicina, donde se capturan imágenes de partes del cuerpo concretas con la finalidad de analizar la existencia o no de posibles tejidos cancerígenos [16], algo que con otra herramienta visual sería prácticamente imposible de realizar. También se puede aplicar a cuestiones medioambientales como, por ejemplo, la teledetección de plásticos en litorales, para la agricultura[19][17][4] y el análisis de los campos de cultivo donde se detecten posibles enfermedades asociadas, necesidad de regadío, etc.

La toma de estos datos puede ser abordada desde diferentes perspectivas en función de la aplicación que se requiera. En este Trabajo Fin de Máster se hace uso de una plataforma de vuelo no tripulada o dron para proceder a ejecutar la captura de imágenes hiperespectrales, de forma que se pueda obtener una vista de planta del elemento que se analiza y examinarlo de forma adecuada, empleando para ello información espectral. Este proceso tiene como cualidad principal poder barrer grandes áreas de terreno muy rápidamente. No obstante, el hecho de transportar una cámara de estas características a bordo de una aeronave es un proceso delicado si se quiere proceder con éxito en la misión. Así, no solo es de vital importancia el control de la misma, sino también la correcta adquisición de esos datos[20], ya que de ellos dependerá la eficacia de los análisis posteriores.

Actualmente se hace uso de la plataforma que se dispone de la siguiente manera: se lleva el dron a la superficie a analizar, se ejecutan sendos vuelos, se aterriza y se transporta el sistema hasta un laboratorio. A partir de aquí se procede a la descarga de los datos contenidos en los sistemas de almacenamiento del ordenador del dron para, una vez obtenidos, realizar los análisis que se requieran. Esto representa una serie de desventajas que hacen del proceso una actividad poco óptima. Por una parte, la labor de transporte del dron es un proceso muy costoso en tiempo. Por otra parte, la calidad de los datos recogidos se evalúa una vez son descargados en el laboratorio, con lo cual, si estos no fueran satisfactorios se debería volver al punto inicial de la tarea. Por todo ello, es preciso tratar de buscar una solución a la problemática existente.

Con todo, este Trabajo de Fin de Máster persigue el diseño e implementación de un sistema de compresión y envío de datos en tiempo real que permita analizar la información capturada por la cámara hiperespectral a bordo de la plataforma de vuelo según esta es tomada, de forma que se puedan variar los parámetros propicios para hacer que la misión del dron sea lo más eficaz posible.

Para alcanzar este propósito la solución se plantea en dos partes diferenciadas. En primer lugar, los datos que tome el dron deben ser comprimidos a bordo, ya que se trata de grandes volúmenes de información que, para poder tener disposición de los mismos en tierra mientras se está ejecutando una misión de vuelo, deben ser comprimidos previamente a ser enviados. Así, dicha compresión se realizará en el ordenador de a bordo, empleando la GPU que trae integrada[12][18][31]. Por otra parte, en la estación de tierra se diseñará un sistema de antenas que apunte al dron durante todo el vuelo con la finalidad de establecer un enlace radioeléctrico que permita descargar la información capturada hacia tierra. Esto permitiría solventar los problemas anteriormente nombrados, ya que al ser posible disponer los datos capturados en tiempo real, se puede analizar la calidad de las muestras y variar los parámetros necesarios para que la misión sea completada con éxito.

En su conjunto, este trabajo pretende ser una aportación en el campo de la adquisición, procesado y transmisión de imágenes hiperespectrales que, específicamente, se aplicará en el campo de la agricultura de precisión y que, de forma potencial, se podría aplicar a otras áreas de interés científico.

2 Antecedentes

2.1 Plataformas de vuelo no tripuladas

Desde su concepción, las aeronaves empleadas para la adquisición de datos y la observación de la superficie terrestre han sido mayoritariamente pilotadas por equipo humano debido a que su control dista de ser trivial y se requiere un personal altamente cualificado al mando de las mismas. Este hecho presenta una serie de ventajas e inconvenientes. En primer lugar, la seguridad es un factor fundamental, por lo que el hecho de que exista un supervisor humano que tenga bajo control el funcionamiento de una máquina es siempre garantía de certidumbre.

Sin embargo, esta situación se presenta a la par como contraproducente en caso de un comportamiento erróneo, tanto por parte de la persona a cargo como de la aeronave, que puede derivar en nefastas consecuencias, pues ninguno de los dos entes está libre de cometer errores. Otro aspecto a considerar es el tamaño, puesto que una aeronave tripulada por un ser humano habrá de tener, como mínimo, un habitáculo para que el piloto la conduzca. En ocasiones, esto no es lo más óptimo dado que para transportar un paquete de tamaño reducido habría que trasladar tanto al paquete como al piloto, lo que conlleva un gasto de recursos considerables.

Por ello, el hecho propio de la inclusión de un piloto es una consideración compleja y deberá valorarse en función de la aplicación. Tal es así que en 1917, 16 años después del primer vuelo pilotado, se ponía a prueba el primer avión no tripulado y controlado mediante técnicas de radio control con la finalidad de que la aeronave hiciera de bomba voladora. No obstante, nunca llegó a emplearse en la práctica este prototipo que llevaba por nombre *Aerial Target* (AT).

No fue hasta 1943, con la segunda guerra mundial, que se puso en vuelo el FX-1400 por parte de la cancillería alemana. Esta aeronave no tripulada constituía una bomba de 2300 libras que se empleó para atacar navíos. Dicho artefacto sirvió como base para otros dispositivos como los misiles antibuque utilizados hoy día así como otro gran conjunto de armas de precisión guiadas [25].

Como se puede apreciar, los drones (plataformas de vuelo no tripuladas), fueron inicialmente empleados con propósitos bélicos. No obstante, en 2006 tomaron vida

los primeros permisos comerciales por parte de la Administración Federal de Aviación (FAA) de los Estados Unidos. Aquí comenzó una carrera para introducir en el mercado aeronaves de todo tipo. La compañía Parrot, de asentamiento francés, lanzaba al mercado el Parrot AR Drone en 2010 [28] que permitía conectarse al teléfono móvil mediante la tecnología Wi-Fi Direct. Esto supuso un éxito rotundo y permitió dar a conocer al gran público la tecnología de los vehículos aéreos no tripulados (UAV).

Este hecho supuso el pilar en el auge del mercado de los drones. A partir de aquí, hubo un gran despliegue donde se comenzaron a adaptar a todo tipo de aplicaciones, desde monitorización de tráfico, cuidado medioambiental, búsqueda de objetivos mediante la integración de diferentes sensores, propósitos de ocio, métodos de reparto de paquetes y un largo etcétera.

2.1.1 Aplicaciones de los UAV

La vorágine de las plataformas de vuelo no tripuladas ha permitido no solo el despliegue de diversas formas de negocio, sino que ha abierto muchísimas posibilidades a diferentes desarrollos tecnológicos. Así, los diferentes investigadores empezaron a incluir esta valiosa herramienta en sus labores, ya que representa una potente ayuda para sus ámbitos de investigación. Por ello, en el estado del arte se encuentran diferentes tópicos como pueden ser: conducción autónoma[6][36], agricultura inteligente[2], detección de desechos en medios rurales y litorales[37], búsqueda y captura de personas [24] y un amplio espectro de posibilidades relacionadas con drones.

Si bien es cierto que esta operativa podría ejecutarse con otro tipo de instrumentos, como pueden ser aviones o satélites, los UAV se presentan como la solución más apropiada por una serie de motivos. En primer lugar, su coste se reduce muchísimo frente a las otras posibilidades, dado que emplear un avión para sobrevolar una pequeña zona de terreno incurre en grandes costes asociados. En segundo lugar, la rapidez es otro factor decisivo, pues volar un dron no requiere demasiado tiempo frente a las otras alternativas, coronándose como una alternativa de alta disponibilidad y muy accesible al usuario. De igual modo, la posibilidad de volar de forma autónoma crea una capa de abstracción en la que el usuario que quiera proceder a ejecutar una misión de vuelo no necesite un conocimiento extenso del manejo de estas herramientas, pudiendo prescindir de una persona que maneje la aeronave presencialmente.

2.2 Tecnología Espectral

La tecnología espectral es otra área de investigación en la cual se ha profundizado mucho en las últimas décadas. Una aplicación de esta es el campo de la teledetección, donde transportando sensores espectrales a bordo de drones y satélites, se escanean ciertas áreas de interés con la finalidad de tener un conocimiento más profundo del objeto bajo estudio.

Para detallar el funcionamiento de los sensores espectrales se deben especificar una serie de conceptos que permitan un enfoque más dinámico sobre esta tecnología. Así, cuando la luz incide sobre un material, una parte de la radiación penetra en el mismo, mientras que la otra se refleja. Gracias a esto, el ser humano tiene la capacidad de ver, en tanto que observará un objeto de un determinado color en función de la radiación que este refleje.

De igual modo, cabe resaltar que el ojo humano únicamente es capaz de observar los objetos en un determinado número de longitudes de onda, las comprendidas entre 400 nm y 700 nm, tal y como se aprecia en la figura 2.1 donde se recoge una instantánea del espectro electromagnético. No obstante, los objetos reflejan y absorben muchas más frecuencias imperceptibles por las personas. Es en este campo donde la tecnología espectral se ha posicionado como un referente.

Por lo tanto, mientras que la imagen típica que se conoce es de tipo RGB, que únicamente contiene la información de los colores referidos al verde, al azul y al rojo, una imagen multi/hiperespectral presenta información en muchas más longitudes de onda. En particular, toda radiación captada desde los 400 nm hasta 15000 nm se considerará una muestra espectral, abarcando desde el ultravioleta hasta el infrarrojo.

De esta manera, el principal propósito de esta tecnología es obtener la *firma espectral* de un objeto, esto es, el rango de longitudes de onda que refleja y que será único para cada material en función de su composición molecular. Esto representa un gran avance en la medida que permite identificar de manera unívoca los objetos presentes en la escena mediante algoritmos matemáticos. Como se deduce, el campo de aplicación de esta tecnología es muy amplio; esta se aplica en áreas de conocimiento tales como la medicina para la detección de células cancerígenas, en la agricultura para la evaluación del estado de los cultivos, etc. En resumen, la tecnología espectral representa una herramienta de identificación mucho más potente que el ojo humano que abre muchas líneas de investigación en diferentes campos de trabajo.

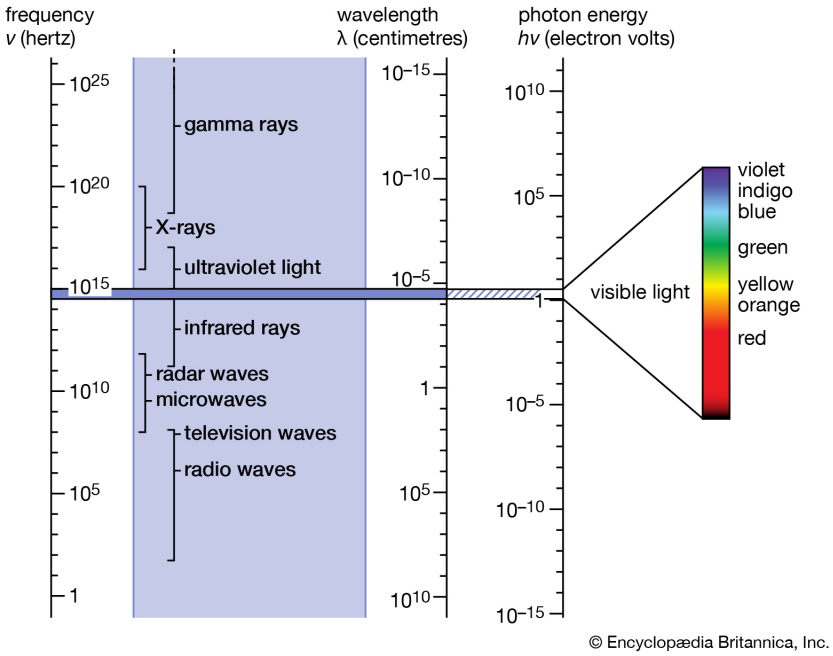


Figura 2.1: Espectro electromagnético

2.3 Imágenes hiperespectrales

Tradicionalmente, las cámaras de uso común han capturado imágenes en tres bandas de longitudes de onda: 400-500nm, 500-600nm, 600-700nm, referidos al azul, verde y rojo, respectivamente. Esto se debe principalmente a que el propósito perseguido por dichas cámaras es capturar la información reproduciendo el funcionamiento del ojo humano lo más fielmente posible. No obstante, con la tecnología espectral se persigue otro propósito que podría considerarse opuesto, es decir, captar lo que el ojo humano no puede, para así poder extraer conclusiones más profundas y certeras acerca del elemento que se precisa analizar.

A su vez, existen en el mercado sensores que pueden captar desde los 400 nm hasta los 15000 nm. En esta línea, se pueden hacer dos grandes clasificaciones. En función del rango de longitudes de onda los sensores serán de tipo CMOS (*Complementary Metal-Oxide-Semiconductor*), InGaAs (*Indium Gallium Arsenide*) o CCD (*Charge-Coupled Device*).

2.3.1 Tipos de cámaras espectrales

Agrupando los distintos sensores se pueden encontrar cinco grandes tipos. En primer lugar, se encuentra el sensor monocromo, este únicamente captura una sola banda proporcionando la radiación reflejada correspondiente a una única longitud de

onda, con lo cual no ofrece demasiada información espectral. En orden creciente de longitudes de onda se encuentran los sensores RGB que capturan desde los 400 nm hasta los 700 nm. Estas imágenes proporcionan poca información a nivel espectral y bastante a nivel espacial. A continuación, se encuentra el sensor espectroscópico, este sensor proporciona desde muchas docenas hasta cientos de bandas espectrales de un único píxel, conteniendo así mucha información espectral pero a cambio, nula información espacial.

En este orden, los sensores multiespectrales serían los siguientes. Estos se caracterizan por captar desde tres hasta diez bandas espectrales, conteniendo información espacial e información espectral limitada. Finalmente, se encuentran los sensores hiperespectrales. Estos contienen desde decenas hasta cientos de longitudes de onda, incluyendo gran información espectral pero menor información espacial. En la figura 2.2 se adjunta una clasificación de los cinco tipos nombrados.

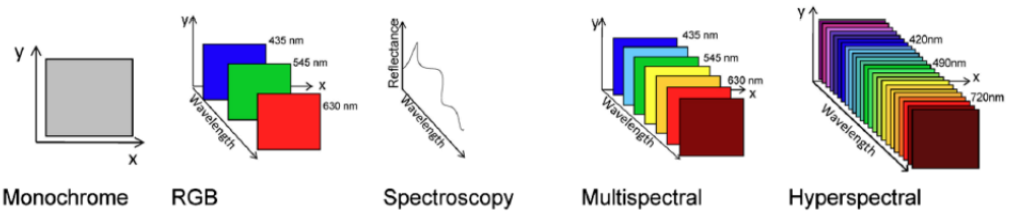


Figura 2.2: Tipos de sensores espectrales

Por tanto, los sensores más interesantes serán los multi e hiperespectrales ya que son los que más información pueden captar. De esta manera, el resultado de la captura de este tipo de cámaras es el conocido cubo hiperespectral mostrado en la figura 2.3.

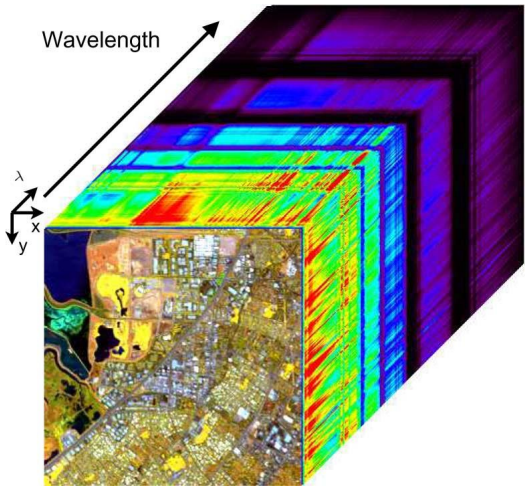


Figura 2.3: Cubo hiperespectral

Como se aprecia, en este cubo se tienen tres ejes, X e Y como dimensiones espaciales, y λ referido a la dimensión espectral. Por ello, para cada píxel de la imagen, tendremos el valor de este mismo píxel en tantas longitudes de onda diferentes como el sensor haya podido captar. Este valor se denomina *reflectancia*, referido a cuanto refleja el objeto de la escena en esa banda determinada. Estas imágenes dan lugar a una gran cantidad de análisis, otro campo de investigación en auge donde se estudian algoritmos de unmixing, clasificación, clusterización, entre otros.

En conclusión, esta tecnología representa una aplicación muy potente ya que permite capturar información más allá del espectro visible, pudiendo llegar a detectar cosas que el ojo humano es incapaz. Dicha técnica se ha utilizado en una amplia variedad de aplicaciones tales como detección de tumores en tiempo real, extracción de objetivos de imágenes en campos como la agricultura, imágenes tomadas desde el espacio, etc.

2.3.2 Clasificación y características de las cámaras hiperespectrales

Atendiendo a la forma de capturar y almacenar información existen cuatro grandes grupos de cámaras hiperespectrales: *staring*, *whiskbroom*, *pushbroom* y *pushbroom*.

Para el grupo de las cámaras de tipo *staring*, se encuentran ejemplares como pueden ser la cámara de rueda de filtros o la cámara de cristal líquido sintonizable. De esta forma, la primera emplea, como su nombre indica, una rueda giratoria que contiene una serie de filtros de interés que permite el paso de las longitudes de onda deseadas. La segunda cámara aprovecha el cristal líquido para hacerlo vibrar a una frecuencia que permita filtrar las longitudes de onda de interés.

Para aplicaciones de teledetección, las cámaras más empleadas son las de tipo *whiskbroom* y *pushbroom*. Como aspecto clave, es necesario reseñar que siempre existirá un *trade-off* entre resolución espectral y espacial. Esto es, una mayor resolución espectral irá siempre en detrimento de la resolución espacial y viceversa. Por ello, es importante atender a la aplicación objeto para emplear la cámara que más se ajuste a las necesidades. Así, las cámaras *whiskbroom* presentarán una mayor preponderancia espectral frente a las *pushbroom*, cuya fortaleza será la resolución espacial.

El mecanismo de funcionamiento de las *whiskbroom* consiste en capturar un único píxel y todas sus bandas. Después de ello, se usa el sistema de espejos mostrado en la figura 2.4 para barrer espacialmente, de forma que se puedan ir capturando los píxeles de la escena. Dicho sistema está formado por un espejo de escaneo que rota según el motor al que está conectado, ocasionando una proyección en el sensor de

la cámara. Para obtener una captura completa de la escena, esta cámara ejecuta dos barridos espaciales diferentes. En primer lugar horizontalmente (línea a línea), y, conforme vaya terminando cada línea, de manera vertical para captar la siguiente. En la medida que esta cámara se transportará sobre una plataforma de vuelo o un satélite, el barrido vertical será ejecutado por el movimiento del mismo.

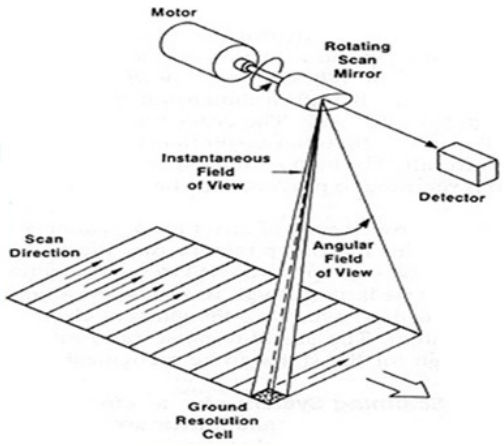


Figura 2.4: Funcionamiento de cámara espectral de tipo whiskbroom

Las cámaras tipo *pushbroom* representan la otra alternativa para usar en aplicaciones de teledetección. La principal diferencia con respecto al tipo anterior subyace en la capacidad de capturar una línea completa sin necesidad de hacer un barrido. Tal y como se muestra en la figura 2.5, la línea de la escena se proyecta sobre el array lineal del sensor y se capturan las bandas secuencialmente. Por ello, la resolución espacial de este tipo de cámaras es mayor que en las whiskbroom, pues se captura una línea del tamaño del array contenido en el dispositivo. De este modo, con un único barrido (el del dron o satélite) se puede capturar la escena completa.

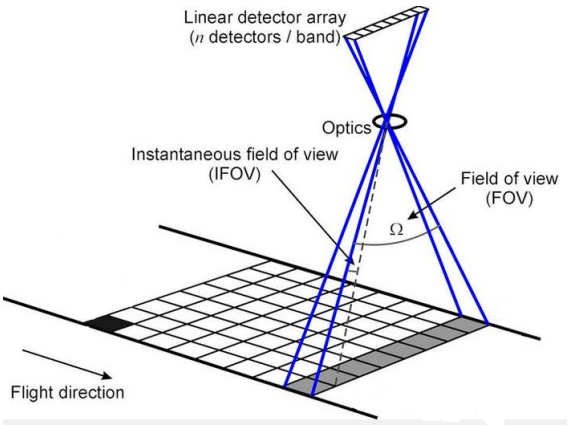


Figura 2.5: Funcionamiento de cámara espectral de tipo pushbroom

Al mismo tiempo, mientras se va capturando la información necesaria, esta necesita ser almacenada en memoria. Dichos datos se salvan de forma secuencial, esto es, un valor tras otro. De este modo, existen tres grandes modos de almacenamiento: *Band Interleave per Pixel* (BIP), *Band Interleave per Line* (BIL) y *Band Sequential* (BSQ).

En el modo de almacenamiento BIP la información se guarda de forma que se registra el primer píxel y el valor del mismo en todas sus bandas. A continuación se procede con el segundo píxel y todas sus bandas y así sucesivamente. Esta forma de capturar es característica de las cámaras de tipo *whiskbroom*. En el tipo BIL, se guarda primero la información de una línea y todas sus bandas, posteriormente se ejecuta para la segunda línea y sus bandas, etc. Este almacenamiento es característico de las cámaras de tipo *pushbroom*. Por su parte, el almacenamiento BSQ consiste en guardar todos los píxeles de una misma banda, a continuación de la siguiente banda y así consecutivamente. Este tipo de almacenamiento es típico de las cámaras tipo *staring*.

Para concluir, dependiendo del orden (*endianess*) en que se acumulen los valores de luminancia de cada píxel se encuentra el formato *Big Endian* o *Little Endian*. El primer modo implica que el bit más significativo será el primero y la segunda forma que el más significativo será el último de cada píxel. Como es natural, esto vendrá dado por las características de la cámara que se emplee y la profundidad de píxel asignada.

2.4 Adquisición de datos con UAV

La tecnología espectral y las plataformas de vuelo no tripuladas forman un binomio excelente, de forma que pueden abordar un gran abanico de aplicaciones. Así, el hecho de poder incluir sensores espectrales en un UAV otorga la capacidad de sobrevolar un área de interés para poder capturar y procesar datos del elemento bajo estudio. No obstante, la ejecución de esta actividad comprende un alto grado de dificultad pues son numerosas las variantes que se deben de tener en cuenta para conseguir resultados adecuados. En particular, el Instituto Universitario de Microelectrónica Aplicada (IUMA) posee una plataforma de vuelo, mostrada en la figura 2.6. Esta cuenta con una serie de elementos característicos que serán detallados a continuación.

2.4.1 Plataforma de vuelo

La plataforma de vuelo es la principal encargada del transporte de los distintos elementos de medida. Por ello, la configuración de la misma representa un punto

Figura 2.6: Plataforma de vuelo con cámara hiperespectral.

crítico que ha de ser tenido en cuenta para que la toma de datos sea exitosa. En la figura 2.7 se muestra el dron empleado [14], se trata del modelo DJI Matrice 600, a este se le incorporarán una serie de elementos que serán detallados a continuación.

Figura 2.7: UAV sin carga útil.

Toda la carga de cómputo del dron, como son el control de la cámara, el almacenamiento de las imágenes y otros requisitos técnicos, son gestionados por el ordenador de a bordo, del cual se realizará un análisis en profundidad en el capítulo 3.1. Sin embargo, en la medida que el dron transportará los diferentes sensores espectrales, los aspectos como el control de la estabilidad se tornan fundamentales ya que de es-

tos dependerá la calidad de los datos capturados. Para ello, se dispone de diferentes sistemas que logran una mayor estabilización del aparato, siendo la controladora de vuelo el más importante.

Esta representa el núcleo central de la aeronave, pues no solamente es la encargada de medir todos los parámetros que afectan al vuelo, como son las distintas fuerzas a las que se ve sometido, altura de vuelo, posición GPS, altitud, etc., sino que también comprende el control de los actuadores y los motores, que serán los encargados de alzar en vuelo la plataforma y controlar que esta permanezca lo más estable posible para ejecutar sus misiones en tiempo y forma. En particular, la controladora empleada es el modelo A3 de la firma DJI [1].

No suficiente con ello, la inclusión de un objeto que estabilice la cámara multi/hiperespectral que transporta el dron es otro elemento clave para añadir mayor solidez a la captura de información.

2.4.2 Gimbal

La correcta estabilización de la cámara de a bordo se consigue con la inclusión de un gimbal. Este es un soporte estabilizador que permite la rotación de un elemento en un único eje, permitiendo un control exhaustivo de los movimientos angulares que realice la cámara.

En términos generales, el gimbal estará compuesto por Unidades de Medida Inerciales (IMU) y motores en cada uno de sus ejes, de forma que usando el acelerómetro y el giróscopo incluidos en ésta, pueda medir las aceleraciones y giros respectivamente. Estas medidas serán empleadas para hacer actuar los motores que actúan en cada uno de los tres ejes, compensando cada movimiento para que el objeto soportado por esta herramienta permanezca inmóvil en todo momento.

Como se puede entender, esto representa un factor fundamental para la cámara de a bordo, ya que permite un mayor equilibrio estableciendo una importante redundancia con los sistemas estabilizadores del dron. No obstante, la cámara hiperespectral necesitará un gimbal de un tamaño considerable que sea capaz de soportar el peso de esta. Así, en la figura 2.8 se recoge el gimbal empleado actualmente para los vuelos [15]. Entre sus características técnicas se encuentra un módulo IMU integrado, un módulo bluetooth, procesador discreto de señales (DSP) de 32 bits, tres modos de operación, facilitando la posibilidad de bloquear un determinado número de ejes y mover el/los otro/s restante/s, etc.

Figura 2.8: Gimbal Ronin MX DJI

2.4.3 Vuelos autónomos

Otro aspecto fundamental para la captura de datos espectrales es dotar a las plataformas de vuelo de la posibilidad de ejecutar vuelos autónomos. De esta forma, el usuario, a través de un dispositivo que puede ser una *tablet*, un móvil o un portátil, planea una misión con sus diferentes *waypoints* (puntos de ruta) para automatizar el vuelo del dron, de forma que no deba estar constantemente un piloto a cargo del vuelo del dron. Esto trae consigo numerosos beneficios: en primer lugar, la planificación del vuelo, lo que permite que el dron siga, en cada instante, una trayectoria marcada. Esto representa una contribución que añade estabilidad al dron pues normalmente la presencia de un piloto será más errónea para seguir una trayectoria predefinida, puesto que el objeto de medida es su ojo y no es tan preciso como las coordenadas GPS que seguiría el ordenador a bordo del dron. Otro aspecto fundamental es la capacidad de observar más eficazmente el terreno sobre el que vuela. Es decir, el usuario contemplará en su dispositivo, mediante una vista de satélite (vista de planta), los puntos de interés sobre los que considera que debe sobrevolar el UAV, de esta forma se conseguirán los datos de manera mucho más eficaz y precisa.

2.4.4 Cámara hiperespectral incorporada

Las cámara hiperespectrales para las que ha sido diseñada la plataforma de vuelo descrita son las que se recogen en la figura 2.9. En la actualidad, se encuentra en uso el modelo FX10 [32] (mostrado en la figura 2.9a), aunque el IUMA también ha procedido a la compra del modelo FX17 [33] para su integración en la plataforma.

(a) Specim FX10 **(b)** Specim FX17

Figura 2.9: Cámaras hiperespectrales de la compañía Specim

En la tabla 2.1 se desglosan las características más significativas de estas cámaras. Asimismo, cabe destacar que ambas cuentan con una interfaz de conexión *GigE Vision*, el cual es un estándar que se introdujo en 2006 para cámaras industriales de alto rendimiento que permite una alta tasa de envío de datos sobre redes de tipo Ethernet. Ambas cámaras cuentan con una profundidad de bit de 12 bits por elemento. En el caso concreto del modelo FX10, el sensor integrado es un CMOS mientras que para el modelo FX17, el sensor es un InGaAs; funcionando en el rango VNIR y NIR respectivamente.

	Specim FX10	**Specim FX17**
Rango espectral (nm)	400-1000	900-1700
Número de bandas espectrales	224	224
Muestreo espacial (Px)	1024	640
Frame Rate máximo (FPS)	330	670
Apertura focal	F/1.7	F/1.7
Peso (kg)	1.26	1.56
Dimensiones (mm)	150x85x71	150x85x75

Tabla 2.1: Características técnicas de las dos cámaras que se emplean en el UAV

3 Compresión de imágenes

En este primer capítulo se trata la parte del proyecto referida a la compresión de imágenes hiperespectrales, donde se aborda la solución actual, las necesidades de compresión, el algoritmo que se empleará según las necesidades indicadas, y cómo adaptar este al funcionamiento de la plataforma de vuelo.

3.1 Solución actual y justificación de compresión

Como ya se introdujo, actualmente el IUMA dispone de una plataforma de vuelo para la adquisición de datos hiperespectrales. El modo de operación que se sigue es el siguiente: se desplaza el UAV al área de interés, se prefijan una serie de configuraciones como son la velocidad de captura en *frames* por segundo, velocidad de vuelo, entre otros. Así, se hacen varios vuelos variando ciertos parámetros para tratar de capturar la máxima información posible que permita realizar un análisis profundo sobre la información recogida. De esta forma, una vez se finalizan los vuelos, se transporta el dron hacia el laboratorio, donde se descargan y analizan todos estos datos. Si la captura no ha sido correcta o la información captada no es del todo satisfactoria, ha de repetirse el proceso con el consumo de tiempo y recursos que ello conlleva, pues no se dispone de una metodología para verificar *in situ* si los datos obtenidos son satisfactorios. Por esta razón, en este capítulo se trata de abordar y solucionar esta cuestión.

Al margen de los sistemas propios encargados del control del vuelo, el UAV incorpora una cámara hiperespectral de cuyo control se encarga un ordenador de a bordo. A estos efectos, pueden emplearse tanto la placa de desarrollo Jetson Nano [26], mostrada en la figura 3.1a, como la Jetson Xavier NX [23], recogida en la figura 3.1b. Las características de este primer dispositivo se recogen en la tabla 3.1. Entre las más interesantes se destaca la inclusión de una GPU de bajo consumo con arquitectura *Maxwell* y con 128 CUDA *cores* de procesamiento. Esta permita gestionar la ejecución automática del vuelo del dron y el proceso de captura de las imágenes hiperespectrales, además de realizar otros procesos de pre-procesado de las imágenes. Adicionalmente, su tamaño, peso y consumo de potencia son bastante reducidos, algo ideal para la aplicación donde se implementa. Por otro lado, las características técnicas del PC Jetson Xavier NX se recogen en la tabla 3.2, donde se resalta una

mayor capacidad de cómputo frente al modelo Nano, con el consecuente aumento de peso y consumo de potencia. No obstante, son iguales en cuanto a tamaño.

(a) PC Jetson Nano **(b)** PC Jetson Xavier NX

Figura 3.1: Opciones de PC OnBoard

GPU	128-core Maxwell
CPU	ARM A57 Quad-core a 1.43 GHz
Memoria	4 GB LPDDR4 a 25.6 GB/s
Conectividad	Gigabit Ethernet
Peso	140 g
Dimensiones	9.906x7.874x2.794 mm
Consumo de potencia	>= 5 Wh

Tabla 3.1: Características técnicas del PC OnBoard Jetson Nano

GPU	384-core Volta
CPU	ARM A57 Quad-core a 2.0 GHz
Memoria	8 GB LPDDR4x a 51.2 GB/s
Conectividad	Gigabit Ethernet
Peso	180 g
Dimensiones	9.906x7.874x2.794 mm
Consumo de potencia	>= 10 Wh

Tabla 3.2: Características técnicas del PC OnBoard Jetson Xavier NX

Como es lógico, esta reducción en peso y potencia generalizada con respecto a un ordenador normal de sobremesa tiene un coste, y es la pérdida de capacidad computacional, algo que se deberá tener en cuenta a la hora de implementar la solución que se persigue.

En cuanto al aspecto más propio de funcionamiento del sistema de captura de imágenes, si bien es cierto que la cámara tiene una velocidad de captura máxima aproximada de 300 FPS, típicamente se suelen realizar misiones cuyo *frame rate* oscila entre 100 y 150 FPS. Para obtener una estimación de la cantidad de datos que se producen por segundo se propone el siguiente ejemplo: una misión de vuelo donde la cámara se encuentre capturando a 150 FPS, cada *frame* capturado contiene 1024 píxeles, 160 bandas y 16 bit por píxel, lo que deriva en un *data rate* que asciende aproximadamente a 393 Mbps. Como se puede deducir, esta tasa de datos es bastante elevada, luego se dificulta la posibilidad de obtener la información en tierra en tiempo real. Como se aprecia en la figura 3.2, se ha realizado una prueba donde se simula el funcionamiento de la cámara hiperespectral y la transmisión inmediata de los *frames* capturados. Se observa que la velocidad de envío es de 20 FPS, que tomando en consideración el tamaño que ocupa cada *frame*, la tasa de transmisión asciende a 56 Mbps aproximadamente, algo insuficiente para el objeto de este TFM. Los detalles del experimento mostrado se abordan en el capítulo 5.

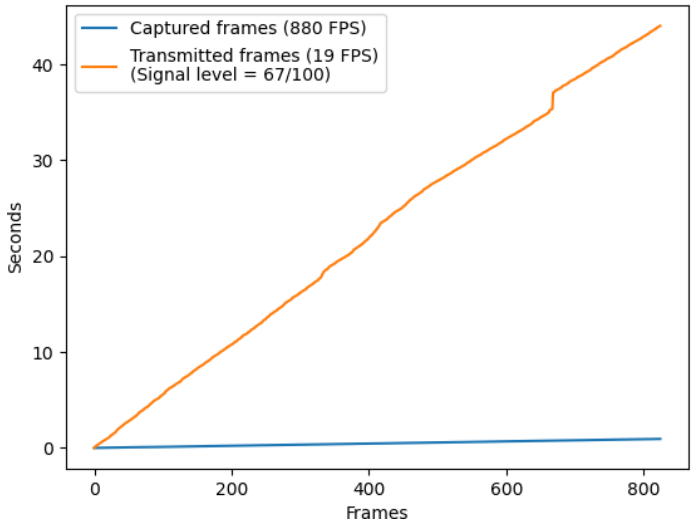


Figura 3.2: Gráfico de transmisión por WiFi entre PC OnBoard y estación de tierra

Por tanto, se puede concluir que existe la necesidad de comprimir los datos capturados antes de ser enviados a tierra para poder cumplir con el objetivo temporal que se persigue en este proyecto. Sin embargo, primero se deben definir estos requisitos de tiempo real adecuadamente. En la medida que no es una aplicación con altas restricciones temporales, se busca la posibilidad de disponer de los distintos *frames* hiperespectrales según van siendo capturados. Sin embargo, y dado que la plataforma de vuelo en su funcionamiento normal estará constantemente alejándose y acercándose del punto de conexión con tierra, no siempre se podrá garantizar la misma tasa de envío de información. En este contexto, la acumulación de cierto número de paquetes de datos a enviar no generaría una problemática grave, siempre y cuando estos sean entregados en un tiempo razonable que permita validar adecua-

damente que la misión de vuelo está siendo correctamente ejecutada en términos de calidad. De igual forma, cabe resaltar que este envío no puede demorar demasiado tiempo, ya que, potencialmente, con los datos recibidos en tierra, se podrán aplicar algoritmos de clasificación, detección de objetivos y agentes anómalos, entre otros, cuyos resultados deberían ser conocidos por el usuario del sistema a la mayor brevedad posible.

3.2 Planteamiento de las necesidades de compresión

Para la búsqueda de un algoritmo de compresión que se adecúe al proyecto que se está desarrollando, se debe tener en cuenta el sistema que se trata. En concreto, la cámara que se emplea para capturar la información espectral es de tipo *pushbroom*, esto es, captura a la vez una línea espacial de píxeles con todas sus bandas espectrales. Por este motivo, se debe emplear un algoritmo de compresión que permita actuar *frame* a *frame*, de forma que puedan ser comprimidos independientemente para ser enviados a tierra inmediatamente después de ser capturados, sin esperar a que concluya toda la misión para ello. Por otro lado, se debe garantizar la posibilidad de prefijar el ratio de compresión, de modo que se pueda certificar que la información nunca es comprimida con un ratio menor al especificado por el usuario con objeto de asegurar que los paquetes de datos son entregados en tiempo y forma.

Así, se debe tener en cuenta los medios hardware que se disponen en la plataforma de vuelo para realizar esta compresión. En tanto que se dispone de un dispositivo de bajo consumo y potencia, las restricciones que se presentan son muy elevadas, con lo cual se deberá buscar un algoritmo que sea *hardware-friendly*, de forma que se consiga el resultado lo más óptimo posible en función de los recursos hardware disponibles. Además, y al incluir la placa de procesamiento una GPU, se tratará de emplear un algoritmo paralelizable, que permita explotar los beneplácitos de esta arquitectura para alcanzar el propósito fijado.

3.3 Algoritmo a emplear

Los investigadores del IUMA han desarrollado el algoritmo de compresión con pérdidas para imágenes hiperespectrales denominado *Hyperspectral Lossy Compression Algorithm* (HyperLCA) que se caracteriza por obtener altos ratios de compresión, un buen rendimiento y una carga computacional bastante reducida. Además, brinda la posibilidad de procesar bloques de píxeles hiperespectrales de forma indepen-

diente, sin necesidad de que haya redundancia espacial entre ellos. De esta forma, se convierte en el algoritmo que más se ajusta a las necesidades planteadas con anterioridad.

Dicho algoritmo se basa en la selección de los píxeles más diferentes de cada *frame* a comprimir, con la finalidad de emplear estos para representar el resto de píxeles existentes mediante el método matemático de proyecciones ortogonales de *Gram-Schmidt* [30]. En particular, este algoritmo está compuesto principalmente por cuatro etapas de procesamiento, tal y como se muestra en la figura 3.3.

En primer lugar, se especifican los parámetros de entrada donde *CR* se refiere al ratio de compresión mínimo exigido por la aplicación, *Nbits o drProj* hace alusión al número de bits que se emplearán para escalar una serie vectores de proyección calculados, *V*, mientras que *BS* (*Block Size*) representa el número de píxeles espectrales a comprimir de manera independiente. Además, para las imágenes que se han empleado para realizar los diferentes test del compresor, el tamaño típico de *Block Size* empleado ha sido de 1024 píxeles, al ser el número de píxeles de cada *frame* independiente capturado por la cámara.

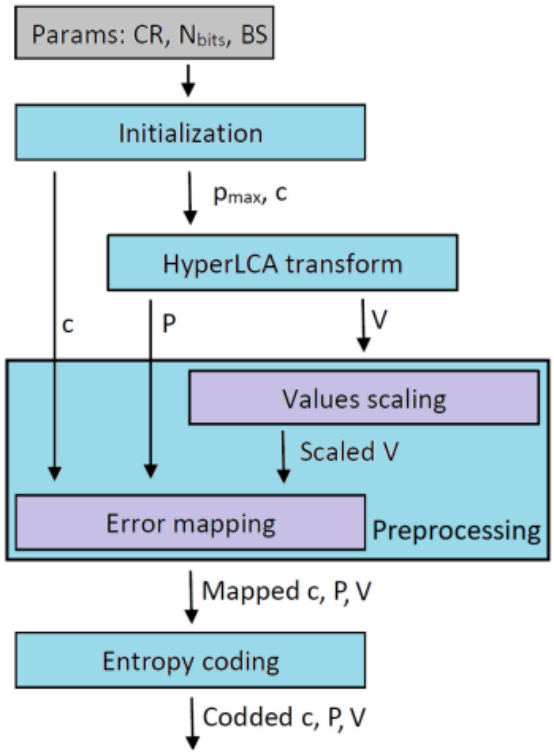


Figura 3.3: Etapas del compresor HyperLCA

La primera etapa, denominada *Initialization* en la Figura 3.3, se encarga de calcular el número de píxeles característicos y los vectores de proyección, *Pmax*, que se extraerán de cada bloque a comprimir de manera independiente. Además, en esta etapa se obtiene el píxel medio, *c*, de los píxeles que conforman cada bloque.

El algoritmo HyperLCA es un compresor basado en transformada. Es por ello que la segunda etapa del proceso de compresión, denominada *HyperLCA Transform* en la figura 3.3, corresponde con las operaciones que realizan dicho proceso de transformación. Esta es la única etapa que introduce pérdidas en el proceso y de la que se obtiene el mayor ratio de compresión. Para comenzar, se resta el píxel medio, *c*, a todos los píxeles del bloque obteniéndose una imagen centrada, M_c. A continuación, comienza un proceso iterativo de tres etapas para extraer los P_{max} píxeles más representativos de cada bloque. En primer lugar, se selecciona el píxel más brillante del bloque, p_i, después se calcula el vector v_i como la proyección de la imagen en la dirección de p_i. Finalmente, la información de la imagen que puede ser representada por el vector seleccionado, p_i, y el vector v_i se resta del bloque original, M_c, quedando en M_c la parte que no puede ser representada por p_i y que, por tanto, representa las pérdidas introducidas por el algoritmo.

La tercera etapa del algoritmo, denominada *Preprocessing* en la figura 3.3, está compuesta por dos partes. La primera, denominada *Value scaling*, realiza un escalado en los vectores de salida de la transformada, *V*, donde los valores de cada uno de sus elementos es redondeado al entero más cercano. La segunda parte de esta etapa, denominada *Error Mapping*, consiste en representar todos los números del vector escalado en enteros positivos de forma que la última de codificación se pueda realizar con mayor eficacia.

Finalmente, la última etapa de este algoritmo corresponde con la codificación entrópica, denominada *Entropy Coding*, la cual se basa en el algoritmo Colomb-Rice [12], donde cada vector de salida se codifica de forma independiente.

Se han realizado varias pruebas con este compresor para valorar si satisface los objetivos del proyecto que se desarrolla. En la figura 3.4, se recogen los resultados de una simulación realizada donde el compresor se ha ejecutado en su versión serie empleando el microprocesador ARM de la Jetson Nano. Como se puede apreciar, se ha especificado una velocidad de captura de 100 FPS y un ratio de compresión de 12, alcanzando un resultado de 5 *frames* comprimidos en un segundo, algo que dista en gran cantidad de los requisitos temporales esperados en este proyecto.

En aras de acelerar el proceso de compresión, el equipo de investigación del IUMA ha realizado una primera implementación paralelizada de este algoritmo sobre GPUs de bajo consumo, en particular, sobre las GPUs embebidas en las plataformas

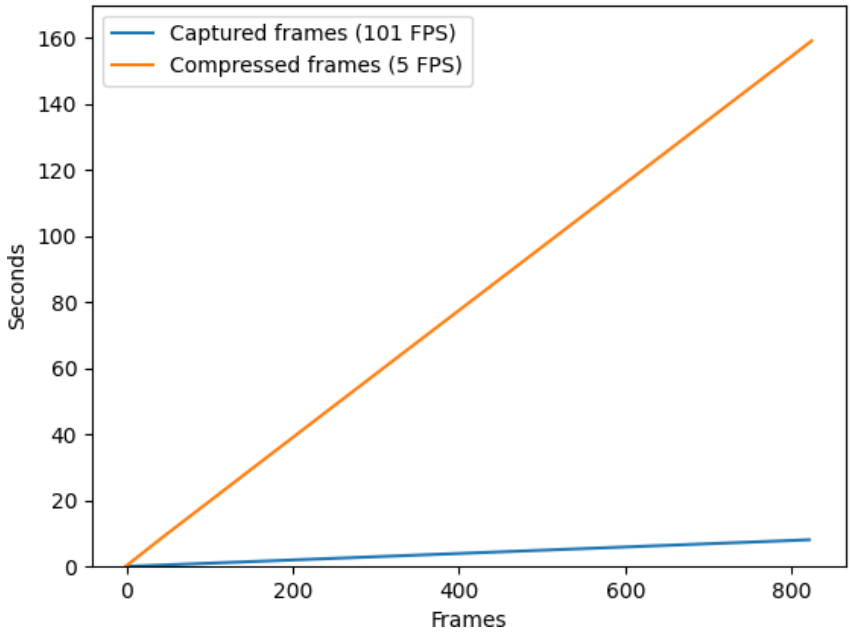


Figura 3.4: Evaluación de compresión serie en Jetson Nano

de cómputo Jetson TK1 y Jetson TX2, cuyos resultados se han analizado en [12]. En total, se diseñaron tres modelos de implementación del algoritmo HyperLCA sobre dichas plataformas de cómputo. El más eficiente, en términos de velocidad de compresión, ha sido aquel en el que las etapas correspondientes con la transformada y el escalado de los vectores, V, se realiza sobre la GPU mientras que las etapas *Error Mapping* y *Entropy Coding* se ejecutan en el ARM de la placa.

Bajo esta casuística, se ha podido verificar correctamente cómo se realiza una compresión en tiempo real de una imagen hiperespectral. El sistema diseñado para realizar estas pruebas se muestra en la figura 3.5. En primer lugar, se han cargado los frames hiperespectrales a ser comprimidos en la RAM del dispositivo. A continuación, mediante el mecanismo de comunicación de memoria compartida (*Shared Memory*) se comparte la información de las imágenes y se envían a los procesos propios de la compresión (transformada en GPU y codificador en CPU). Una vez realizadas estas tareas, se almacena el *bitstream* de los datos comprimidos en RAM y se calcula el tiempo empleado en todo el proceso.

Si bien es cierto que el modelo planteado sirve como prueba de concepto para analizar las ventajas de implementar una versión paralelizada del algoritmo de compresión HyperLCA en GPU, el funcionamiento de la plataforma de vuelo es ligeramente diferente. Conforme la cámara hiperespectral va capturando los *frames* línea a línea, los guarda en un disco de estado sólido (SSD) en el caso de la Jetson Nano o en la tarjeta SD en la Jetson NX, todo ello con la finalidad de evitar posibles pérdidas durante la transmisión de los bloques comprimidos, como podría ocurrir en

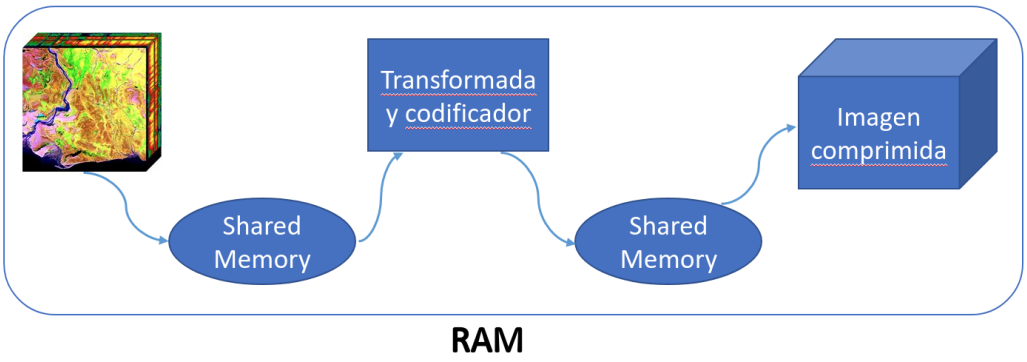


Figura 3.5: Primera aproximación de compresión paralela

un proceso de *streaming* como el anteriormente analizado y propuesto en [12]. Es por ello, que se ha tenido que modificar el modelo de implementación de partida con el objeto de incluir los procesos de guardar la información capturada en un disco SSD y posteriormente, leer los datos a procesar desde los ficheros generados, con la finalidad de almacenar toda la información capturada en caso de no poder ser enviada a tierra satisfactoriamente por cualquier motivo, o para contar con la información original sin pérdidas una vez se llegue al laboratorio en caso de que se desease trabajar con esta.

3.4 Implementación y optimización del algoritmo ajustado a la solución actual

Para dar solución y ajustar el algoritmo de compresión al funcionamiento actual de la plataforma de vuelo, se ha diseñado la arquitectura mostrada en la figura 3.6.

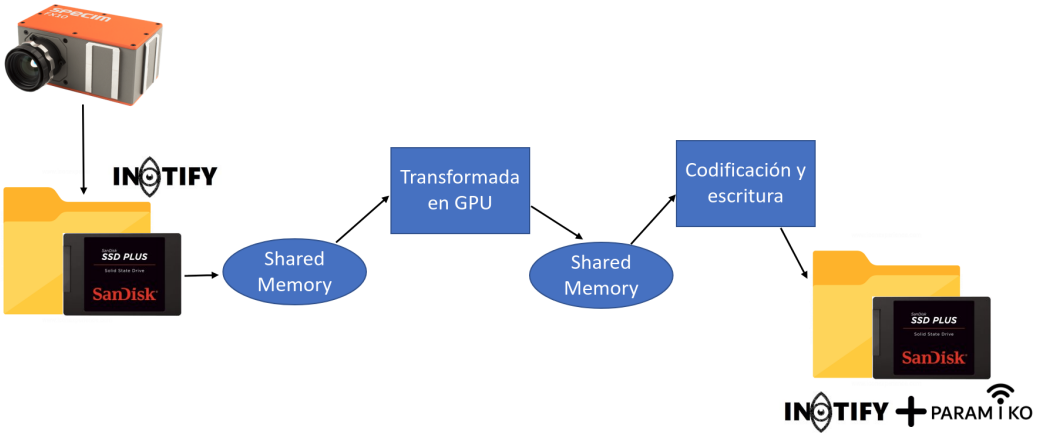


Figura 3.6: Diseño de la arquitectura de compresión y envío de imágenes hiperespectrales

En primer lugar, la cámara se encuentra capturando información espectral y almacenándola en el disco SDD. Para detectar cuando un nuevo fichero se ha generado con la información de un nuevo *frame* capturado, se ha implementado el módulo de Python denominado iNotify [21] en la figura 3.6. Este permite realizar una programación orientada a eventos, particularmente con respecto al evento propio de la creación de estos archivos. Conforme se va creando cada fichero, será necesario ejecutar el algoritmo de compresión. Para ello, y en la medida que la transformada se ejecuta en la GPU y el codificador en la CPU de la plataforma de cómputo, ambos procesos deben comunicarse entre sí. Para solventar esta problemática se ha decidido implementar una memoria compartida que está compuesta de dos partes diferenciadas. Por un lado, se definen unos segmentos de memoria que sirven de sincronización entre los procesos y que consta de los siguientes elementos:

- *Last Captured Frame Index*: índice que indica el último frame capturado por la cámara.
- *Last Transformed Frame Index*: índice que indica el último frame transformado.
- *Last Coded Frame Index*: indica el último frame codificado.
- *Proccess Finished*: indica la finalización del proceso.

De esta forma, los procesos de captura, la transformada y la codificación se podrán comunicar eficazmente, de modo que se lleve un registro adecuado entre los diferentes procesos garantizándose la compresión independiente y en orden de cada *frame* capturado. Definidos los parámetros de sincronización, se han establecido dos segmentos de memoria compartida. Por un lado, *Input Frame Ring Buffer*, donde la cámara hará de productor y la transformada de consumidor. Por otro lado, la región denominada *Transform Output Data* que tendrá como productor a la propia transformada y como consumidor al codificador entrópico. Además, estas regiones de memoria se han definido con un tamaño máximo para 10 *frames*, de forma que no se puedan guardar en memoria compartida nuevos *frames* si se ha producido un retraso en el procesamiento de diez unidades por parte de la transformada o del codificador.

Con el procedimiento descrito se habilita la comunicación y sincronización total entre los procesos. Su modo de funcionamiento a nivel global es el siguiente:

1. Se detecta la creación de un nuevo *frame* con el módulo iNotify y se carga en la memoria compartida así como el índice del mismo.

2. La transformada, que se encuentra en todo momento comprobando si existe un nuevo *frame* capturado, lo comprime escribiendo el bitstream correspondiente y el índice del frame transformado en las posiciones de memoria reservadas para tal fin.

3. Posteriormente, el codificador que sigue un funcionamiento similar, detecta un nuevo *frame* transformado, lo codifica y lo escribe en disco.

4. Una vez realizado todo este proceso, con el módulo Paramiko [38] de Python en conjunción con iNotify, se detecta la creación del frame comprimido que es directamente enviado a tierra mediante conexión SSH.

3.4.0.1 Lógica de funcionamiento del sistema

A continuación, se pasa a detallar de forma minuciosa la lógica detrás de cada proceso que se ha desarrollado para implementar el sistema diseñado. En primer lugar, se muestra el funcionamiento de la parte encargada de reaccionar a la creación de nuevos *frames* hiperespectrales. Para implementar esta librería se sigue el siguiente *workflow*: se crea el objeto iNotify con el que se añade un *watchdog* en la carpeta a monitorizar, en concreto en la que se guardan los *frames* capturados; a continuación se establece un bucle que reacciona a los eventos de creación de archivos, donde cada vez que se crea uno nuevo, se añade a una lista externa donde se va añadiendo cada uno. Finalmente, se comprueba si el último archivo creado coincide con la condición de finalización (archivo creado por el proceso de captura, al finalizar, con un nombre específico), en tal caso se elimina el *watchdog* de la carpeta antes mencionada y concluye el proceso. El código de esta parte se muestra a continuación:

```
# Inotify process
notifier = inotify.adapters.Inotify()
notifier.add_watch(inputFolder)
for event in notifier.event_gen():
    if event is not None:
        if 'IN_CREATE' in event[1]:
            fileName = event[3]
            filePath = event[2] + fileName
            externalFrameList.append(fileName)
            if fileName == finishCondition:
                notifier.remove_watch(inputFolder)
```

Este es el hilo principal de funcionamiento, no obstante, debe existir un hilo de ejecución en *background* que se encargue de gestionar esta lista a la que se adhieren los distintos *frames*. Para ello, se diseña una función que tenga por objeto esta gestión, de forma que esté constantemente verificando si existen nuevos *frames* en la lista, y según se añada uno, se ejecute el proceso de compresión. Este proceso se estará ejecutando hasta que se registre el archivo último que indica la finalización del proceso de captura. En el código que se muestra a continuación se detalla en primer lugar la función de procesamiento de la lista y el lanzamiento de este hilo de ejecución en *background*:

```
# Function to be executed in a parallel thread
def listProcessingThread(frameList):
    thereAreMoreFilesToBeCompressed = True
    while thereAreMoreFilesToBeCompressed:
        while len(frameList) > 0:
            fileName = frameList.pop(0)
            if fileName != finishCondition:
                compressFrame(fileName)
            else:
                thereAreMoreFilesToBeCompressed = False
                break

# Launching the target function in a parallel thread
parallelThread = threading.Thread(target=listProcessingThread, args=(
    externalFrameList,))
parallelThread.start()
```

La función de compresión denominada *compressFrame* a la cual se llama cada vez que se captura un nuevo *frame* se encarga de leer el archivo correspondiente a dicho frame y cargarlo en el buffer de la *shared memory* del cual lee la transformada. Tras hacer esto, la función *compressFrame* incrementa el puntero de la *shared memory* que indica que hay un nuevo frame capturado para notificar a la transformada de que ya puede comenzar a procesar dicho frame. Si la transformada se encontrase procesando un frame anterior, y fuese suficientemente retrasada como para que no haya hueco en el buffer para este nuevo *frame*, la función *compressFrame* esperaría en un bucle *while* hasta la liberación de este recurso. Esto es posible gracias al puntero de la *shared memory* en el que se indica el índice del último *frame* que ha sido procesado por la transformada.

Por otro lado, una vez se han comprimido los *frames*, la última parte del proceso está compuesta por la conjunción del módulo iNotify y Paramiko, de forma que una vez

se detecte la escritura de un *frame* comprimido en el directorio correspondiente del disco SSD, este sea enviado a la estación de tierra. Con respecto a la implementación del módulo Paramiko a nivel de código, se debe especificar tanto el cliente SSH (Secure Shell) para realizar la conexión con la estación de tierra, como el cliente FTP (File Transfer Protocol), encargado del envío de información. A continuación, se llama a la función *put* del cliente FTP que realiza el envío del *bitstream* resultado de la compresión. Para finalizar, una vez se ha comprimido el último frame, el flujo de comunicación del cliente FTP y SSH deben ser cerrados. El código correspondiente se recoge a continuación:

```
# Stablishing the connection to other PC via SSH
client = paramiko.SSHClient()
client.load_system_host_keys()
transport = paramiko.Transport(hostname, port)
transport.connect(username=username, password=password)
sftp = paramiko.SFTPClient.from_transport(transport)

# Function to be executed in a parallel thread
def listProcessingThread(frameList):
    thereAreMoreFilesToBeTransmitted = True
    while thereAreMoreFilesToBeTransmitted:
        while len(frameList) > 0:
            fileName = frameList.pop(0)
            src = os.path.join(inputFolderPath, fileName)
            dst = os.path.join(destinationFolderPath, fileName)
            sftp.put(src, dst) # File transmission
            if fileName == finishCondition:
                thereAreMoreFilesToBeTransmitted = False
                sftp.close()
                transport.close()
                break
```

La elección del módulo Paramiko para gestionar la transmisión de los *frames* comprimidos viene motivada por una serie de cuestiones. En primer lugar, la gran cantidad de documentación disponible existente que ha permitido un rápido desarrollo, así como la posibilidad de resolución de posibles inconvenientes al ser una librería ampliamente utilizada en la comunidad de desarrolladores. Por otro lado, el contar con un gran *set* de funciones a nivel de red, otorga una flexibilidad que permite el prototipado rápido de cualquier solución ideada. Finalmente, la posibilidad de establecer diferentes canales de comunicación a modo de *socket* representa una

solución muy atractiva que puede ser implementada en iteraciones posteriores de este proyecto con la finalidad de lograr una mayor optimización.

4 Diseño del sistema de comunicaciones

En este cuarto capítulo se trata el desarrollo e implementación del sistema diseñado para el canal de comunicaciones que hace posible el envío de los datos capturados por la plataforma de vuelo hacia la estación de tierra. Esta etapa del proyecto es una parte crucial del mismo en tanto que es la que permite que la información captada por el dron, una vez procesada, esté disponible en tiempo real y de forma completa en la estación de tierra para su análisis correspondiente. Así, se tratan diversos aspectos que atañen al diseño de este sistema como son la vía de comunicación elegida, el diseño a nivel estructural, mecánico y electrónico del sistema, detallando cada particularidad de mayor a menor nivel de abstracción.

4.1 Vía de comunicación empleada

Al inicio de este proyecto, únicamente se encontraba diseñada la comunicación entre el UAV y la estación de tierra, ejecutada por vía radio. A través de este enlace se envían comandos hacia la plataforma de vuelo (despegue, aterrizaje, velocidad de vuelo) a lo que esta responde con datos de telemetría: coordenadas GPS (Global Positioning System), altitud, nivel de batería, etc. Al considerarse aprovechar esta vía de comunicación para transmitir los datos espectrales capturados, se desechó por dos motivos: no alcanzar el ancho de banda suficiente y evitar saturar el canal con otros datos que difieran del mero control de la misión de vuelo.

Así, dado que el dron lleva instalado un ordenador de abordo encargado de controlar la misión de vuelo, de gestionar la captura de datos y procesar parte de estos si fuera necesario, se ha decidido que la comunicación entre este y la estación de tierra se realice mediante WiFi. Esta vía de comunicación comprende una serie de alternativas de gran interés. Por un lado, combate las desventajas que presenta la comunicación por radio en tanto que alcanza el ancho de banda necesario para transmitir la información capturada en tiempo real a tierra. En segundo lugar, por su bajo coste y al ser estos dispositivos de red muy demandados, sus precios también son más reducidos que otras alternativas disponibles, permitiendo construir una solución mucho más económica. Por otro lado, existen gran cantidad de librerías

software para este protocolo, lo que posibilita un desarrollo más rápido, eficiente y certero que otras opciones.

No obstante, esta forma de comunicación incurre en una serie de desventajas que deben ser tenidas en cuenta con el objeto de subsanarlas. Deben considerarse dos aspectos clave que permitan que el sistema funcione correctamente: el estándar empleado y las antenas elegidas, de forma que se cumplan los requisitos de ancho de banda y estabilidad en el enlace, garantizándose que los datos son entregados correctamente y en tiempo real en los términos especificados en el capítulo previo de compresión.

4.1.1 Análisis de los estándares WiFi (802.11) existentes

Para poder cumplir con las restricciones de ancho de banda requeridas en este proyecto, es necesario analizar las diferentes posibilidades que ofrece el empleo de esta vía de comunicación y escoger cuál es la más apropiada.

Existen diversas versiones del estándar 802.11 [10], sus principales cualidades se muestran en el cuadro 4.1 donde se hace un desglose de su alcance, velocidad máxima teórica y frecuencia empleada por cada uno.

Estándar	**Frecuencia (GHz)**	**Velocidad máxima teórica (Mbps)**	**Alcance interior -exterior**
802.11	2.4	2	20-100
802.11a	2.4	11	35-120
802.11b	5	54	35-140
802.11g	2.4	54	38-140
802.11n	2.4-5	600	70-250
802.11ac	5	1300	35-200

Tabla 4.1: Características técnicas de los diferentes estándares WiFi

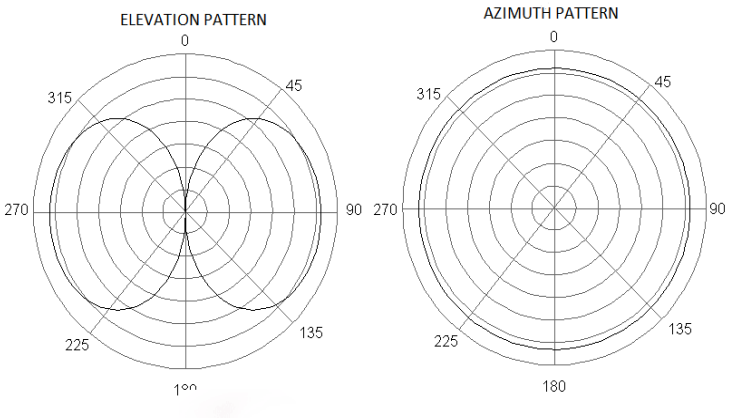
A la vista de estos datos se puede concluir que la versión más interesante es la 802.11n por tener el mayor alcance en exteriores e incorporar la banda de 2.4 GHz, permitiendo la compatibilidad con todas las posibles estaciones de tierra. Asimismo, cuenta con una velocidad máxima teórica de 600 Mbps. Para analizar si esta velocidad es suficiente se remite al lector a la casuística explicada en la Sección 3.1, donde si se obtiene un *data rate* de captura de 393 Mbps y estos datos son comprimidos con un ratio de compresión de 15, obteniendo una tasa de datos resultante de 26.2 Mbps, se asegura la entrega de información en tiempo real. Por tanto, se concluye que la versión 802.11n cumple de forma holgada con estas restricciones.

4.1.2 Recepción de la señal

Como anteriormente se introdujo, otro factor clave en el diseño de esta clase de sistemas es el empleo de una antena apropiada que permita mantener un enlace estable, seguro y lo más veloz posible, en aras de maximizar la mayor cantidad de datos transmitidos entre los dos terminales, antena y UAV.

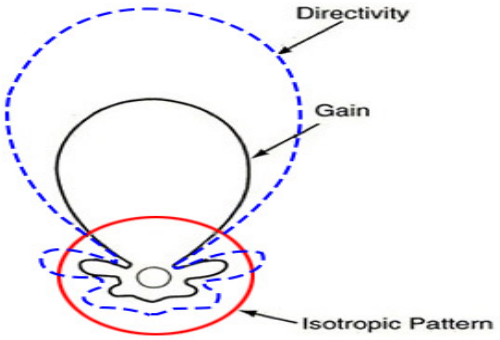
Típicamente, los *routers* incluyen antenas omnidireccionales como la mostrada en la figura 4.1a, caracterizada por tener un diagrama de radiación como el recogido en la figura 4.1b. Analizando el patrón de radiación del ángulo azimuth, se concluye que la densidad de potencia radiada es igual en todos los ángulos, lo que representa una característica intrínseca de este tipo de antena.

(a) Antena omnidireccional



(b) Diagrama de radiación de antena omnidireccional

(c) Antena direccional



(d) Comparación de diagramas de radiación

Figura 4.1: Tipos de antena y sus diagramas de radiación

Estas antenas omnidireccionales son de gran interés y un ejemplar irá incluido en el dron, donde no se dispone de espacio ni energía suficiente para diseñar e implementar un sistema de apuntamiento como sí sucede en la estación de tierra.

No obstante, este tipo de antenas no representa la solución más apropiada para incluir en el segmento tierra, dado que se persigue conseguir la máxima cobertura en torno a un punto concreto, esto es, el UAV en vuelo. Por esta razón, una antena direccional, como la recogida en la figura 4.1c, es la más apropiada. En la figura 4.1d se compara el diagrama de radiación isotrópico de una antena omnidireccional (curva roja) frente a una antena más directiva (curva azul), donde se puede apreciar una focalización mayor de la potencia de la señal en torno a una región de interés.

4.2 Descripción general del sistema de apuntamiento

Una vez se ha decidido de qué forma se realizará la descarga de datos espectrales entre la estación de tierra y el UAV, es necesario resolver el problema acerca de cómo lograr que la antena se encuentre apuntando en todo momento al elemento en vuelo. Como es natural, se incluirán diversos sensores en el sistema de apuntamiento que brinden la información necesaria para el propósito que se persigue, pero antes de pasar a detallarlos, es preciso indicar que la comunicación entre la estación de tierra y la antena se realizará por vía Bluetooth. Para esbozar una idea general del sistema que se pretende desarrollar a nivel de interfaces de comunicación, se muestra en la figura 4.2 un esquemático del mismo, incluyendo los diferentes protocolos empleados entre cada elemento.

A modo de resumen, la estación de tierra estará compuesta por un *router* al que irá conectado el sistema de apuntamiento y un portátil encargado de gestionar tanto el control del vuelo como la recepción de los datos espectrales. De igual modo, la comunicación entre la antena rotatoria y el portátil se realizará por comunicación Bluetooth. Se ha escogido este último protocolo debido a que la estación de tierra puede ser tanto un portátil como una tablet, y el nexo común a ambas es la disponibilidad de WiFi y Bluetooth.

Habiéndose determinado las comunicaciones que se realizarán entre los diferentes elementos, se hace posible el intercambio de información entre los sensores del sistema de antenas y la estación de tierra de forma que se puede ahondar en la cuestión más propia de cómo se logrará que el sistema se encuentre constantemente apuntando a la plataforma de captura de datos.

En primer lugar, será necesario conocer la ubicación de ambos objetos. Como se detalló anteriormente, la plataforma de vuelo está en continua comunicación con la estación de tierra recibiendo órdenes y respondiendo con datos de telemetría, entre ellos su ubicación GPS (latitud y longitud) así como su altitud relativa y absoluta.

En consecuencia, también será necesario conocer esta información del sistema de antenas, esto se realizará mediante la inclusión de un sensor GPS. Con los datos de ubicación de ambos dispositivos se podrá calcular cómo se debe mover la antena para apuntar al dron.

Figura 4.2: Esquema de comunicaciones de la plataforma

Previamente a la descripción de los cálculos teóricos a realizar, es necesario detallar los movimientos que realizará la antena. Tal y como se muestra en la figura 4.4, se ha decidido que esta se mueva horizontal y verticalmente, variando los ángulos de *elevación*, grados con respecto al suelo, así como *azimut*, grados con respecto al norte. Con estos dos movimientos básicos se puede asegurar que la antena apunte al UAV durante todo su vuelo.

En base a esto, se debe establecer un sistema de referencia global. Concretamente, se ha escogido el sistema de coordenadas NED (*North-East-Down*) [8] como se muestra en la imagen 4.4, el cual a partir de los datos de ubicación de la antena y el UAV ofrece como resultado los incrementos en Norte, Este y Abajo (*Down*) en los que debiera moverse la antena para apuntar a la plataforma de vuelo.

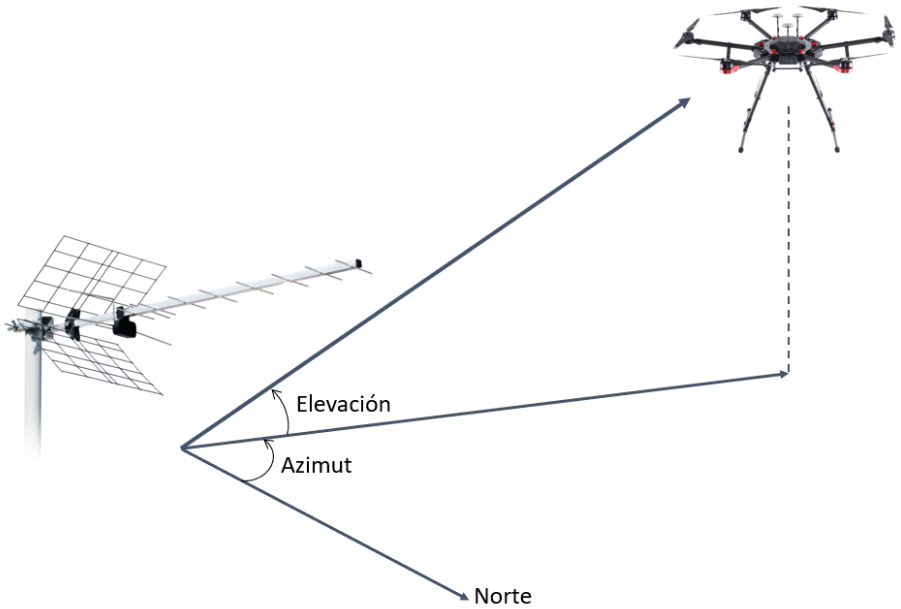


Figura 4.3: Movimientos de la antena para apuntar al dron en vuelo

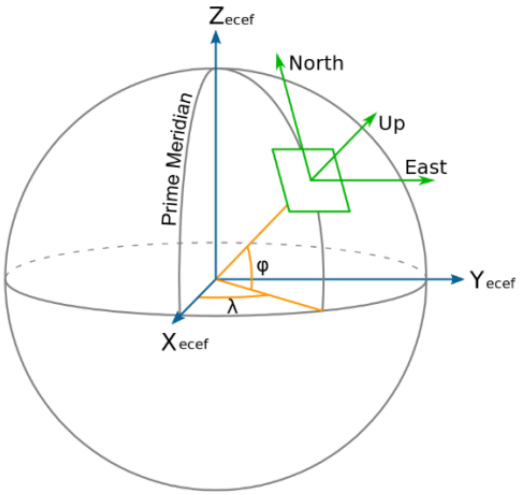


Figura 4.4: Movimientos de la antena para apuntar al dron en vuelo

Esta serie de incrementos deben ser tomados en consideración para ajustar la orientación de la antena y lograr el propósito que se persigue. A modo de ejemplo, si el sistema se encontrara en una superficie totalmente horizontal, el incremento en la coordenada *Este* y la coordenada *Norte* coincidiría con los ángulos de azimuth (λ) y elevación (ϕ) respectivamente, que habría que rotar la antena para posicionarla apuntando al dron en vuelo.

Sin embargo, esta situación es poco realista, en tanto que difícilmente se podrá ubicar esta antena en un plano perfectamente horizontal. Por ello, se debe buscar una forma de solventar dicha problemática. La solución pasa por conocer la orientación de la antena, de forma que se pueda realizar una conversión rápida y fiable entre las coordenadas NED a las que debe moverse la antena y cómo debe variar esta su orientación para conseguirlo. Para este propósito, se debe especificar un sistema de referencia local, que permita *interpretar* las coordenadas NED globales en movimientos angulares locales como son el Pitch, Roll y Yaw mostrados en la figura 4.5.

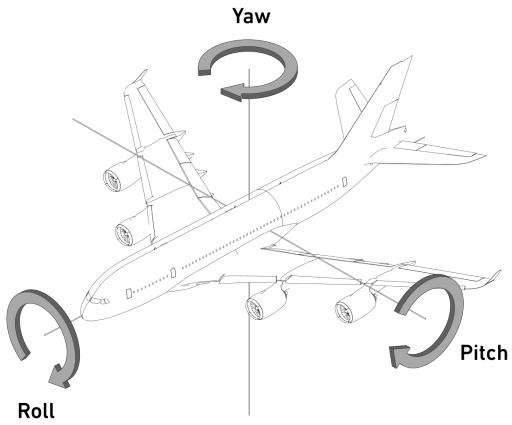


Figura 4.5: Sistema de referencia local en coordenadas Pitch, Roll y Yaw

Para poder calcular estos datos angulares, es necesario disponer de una Unidad Inercial de Medida o IMU por sus siglas en inglés. Este dispositivo se compone de un acelerómetro (para medir aceleraciones lineales), un giróscopo (para las aceleraciones angulares) y un magnetómetro (para cuantificar la intensidad del campo magnético). De esta forma y como se profundizará más adelante, la información extraída de estos sensores será suficiente para conocer los valores locales de Pitch, Roll y Yaw del sistema de apuntamiento. Con todo ello, es posible lograr que la antena sea capaz de ejecutar los movimientos necesarios para apuntar en todo momento al dron mediante su propio sistema de referencia local.

Una vez se ha planteado el problema y su solución de manera global, se procede a su descripción de manera detallada, ahondando en cada uno de los componentes y procesos utilizados en el sistema.

4.3 Lógica de funcionamiento del sistema

Habiéndose descrito todos los componentes que se requieren para el despliegue del sistema de comunicaciones y el funcionamiento general de la antena, el siguiente paso es detallar la lógica de funcionamiento de esta. En la figura 4.6 se recoge un desglose de cada etapa a nivel general.

En primer lugar se debe realizar la calibración del sistema. Este es un paso fundamental que consiste en mover la antena en todas las posiciones posibles con la finalidad de asegurar que los datos tomados por las IMU son los correctos, de forma que se calculen los posibles *offset* que potencialmente puedan afectar a una toma errónea de medidas. Posteriormente, se ejecuta la fase de inicialización cuyo paso más importante es el *homing*. Este consiste en ubicar el sistema en una posición de referencia a partir de la cual se puedan ejecutar órdenes de movimiento partiendo de una posición conocida. Llegados a este punto, el sistema de apuntamiento comunicará dos datos a la estación de tierra: los de orientación y su posición GPS.

Una vez se hayan ejecutado estas tres etapas, comienza el bucle general de funcionamiento. El UAV comunicará a la estación de tierra sus datos de ubicación y altitud relativa (telemetría) a través del canal de radio. Esta realizará los cálculos necesarios para hallar los movimientos que debe ejecutar cada motor y ajustar los ángulos de rotación de la antena, los cuales serán comunicados por Bluetooth al sistema de antena móvil y se accionarán los actuadores que la orientarán hacia la posición actual del UAV. Este bucle continuará hasta que se dé por finalizado el vuelo. Con todo, se consigue que la antena esté constantemente apuntado a la plataforma de vuelo en todo su recorrido dando soporte durante todo el intervalo de tiempo que dure el vuelo.

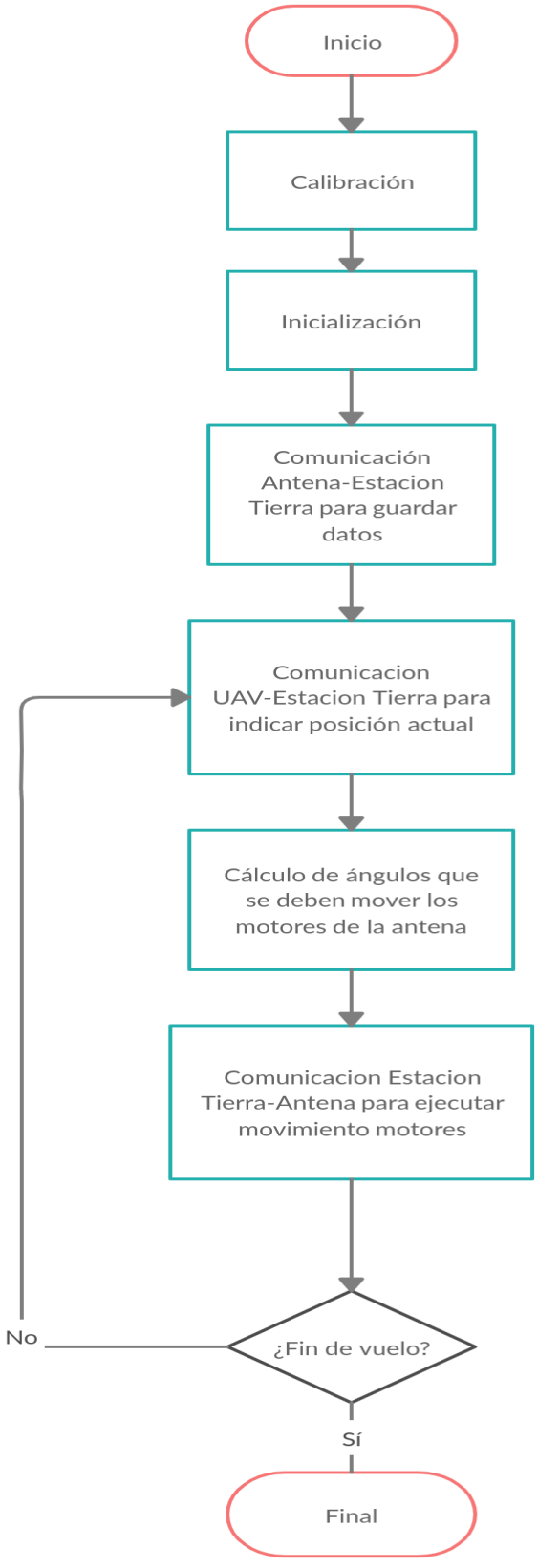


Figura 4.6: Diagrama de funcionamiento del sistema de antenas

4.4 Diseño estructural de la antena

Para escoger correctamente un diseño a nivel estructural que se ajustara a las pretensiones del proyecto que se desarrolla, se ha explorado una solución que permita el movimiento de la antena en dos grados de libertad, elevación y azimuth. De esta forma y tras un análisis exhaustivo, se determinó que el diseño que más se ajustaba a las necesidades y que podía ser impreso en la impresora 3D de la que dispone el IUMA es el recogido en la figura 4.7 [5].

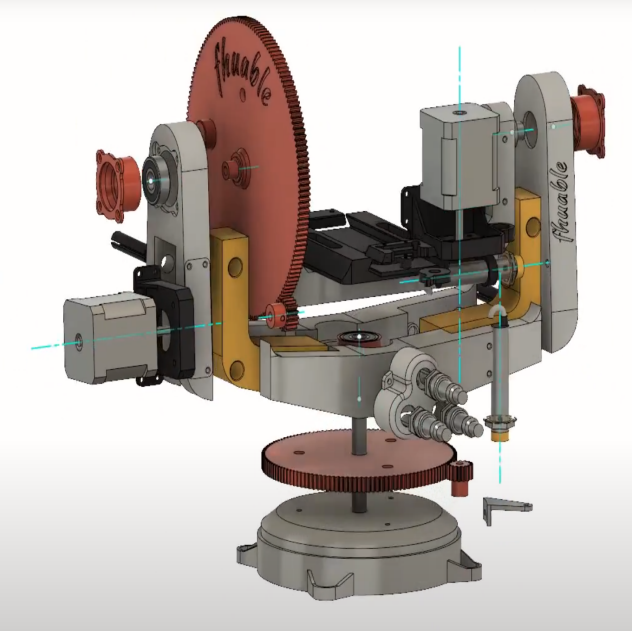


Figura 4.7: Estructura sobre la que irá montada la antena

Como se puede apreciar, está compuesto por dos ruedas dentadas principales, la referida al ángulo de elevación incorpora 160 dientes mientras que la que se sitúa en la base, referida al *azimuth*, está compuesta por 128 dientes. Así, en la medida en que cada eje de los motores tendría incluida una rueda dentada de 16 dientes, las relaciones de transmisión ascienden a 10 y 8, respectivamente. Esto significa, por ejemplo, que 10 vueltas del motor que varía el ángulo de elevación implica una vuelta de la rueda dentada referida a este. Finalmente, para lograr el correcto movimiento de cada motor, serían necesarios cuatro rodamientos en total, dos por cada rueda, de tamaño 26x10x8 mm.

De igual modo, cabe destacar que este diseño ha sido ligeramente modificado, sustrayendo piezas innecesarias a los propósitos de este proyecto e incluyendo otras de elaboración propia que permitieran un mejor ajuste de cada uno de los componentes que integraban la estructura. En conclusión, se escogió esta opción por

ser la que permitía una mayor celeridad para poner en prueba el sistema ideado, de forma que pudieran probarse todos los componentes en su conjunto una vez adheridos al sistema.

4.5 Diseño mecatrónico de la antena

En esta sección se abordarán los aspectos relativos a la electrónica y mecánica que emplea el sistema de antenas desarrollado. Se tratará con profundidad cada uno de los dispositivos que se emplean, detallando no solo sus características técnicas, sino de qué forma se han empleado para conseguir los objetivos de este Trabajo de Fin de Máster. De igual modo, se especificará el código de funcionamiento de cada elemento empleado en su cooperación con el núcleo central del sistema, el Arduino NANO.

4.5.1 Alimentación del sistema en su conjunto

En la figura 4.8 se recoge un desglose de los diferentes elementos que conforman esta parte del proyecto. A la izquierda se encuentra la parte referida a la alimentación del circuito. Se dispone de dos baterías, una para la alimentación del *router* y el amplificador (módulo de telecomunicaciones) y la otra batería para el módulo de apuntamiento de la antena. Como se indicará maś adelante, se podría usar una única batería correctamente dimensionada que provea alimentación a ambos circuitos. Además, se emplean varios convertidores de tipo BUCK cuya función es disminuir el voltaje de salida de las baterías y mantenerlo estable.

Por otro lado se muestra el microprocesador, un ATMEGA328 integrado en una plataforma de desarrollo Arduino, en este caso la versión NANO [3]. De igual modo, se muestran los circuitos de potencia y control. El primero está comprendido por un driver (DRV8825), encargado de proveer la corriente necesaria a las dos bobinas de cada motor para conseguir el movimiento deseado del rotor [27]. Del mismo modo, se disponen los motores escogidos para este proyecto, se trata del modelo NEMA 17, atractivo por tener un tamaño y peso adecuado a la estructura donde se integrará así como por cumplir los requisitos de par necesario para mover la carga que, en este caso, será el elemento radiante o antena. A su vez, el circuito de control está compuesto por la IMU MPU9250, el módulo Bluetooth para la comunicación con la estación base y el módulo GPS.

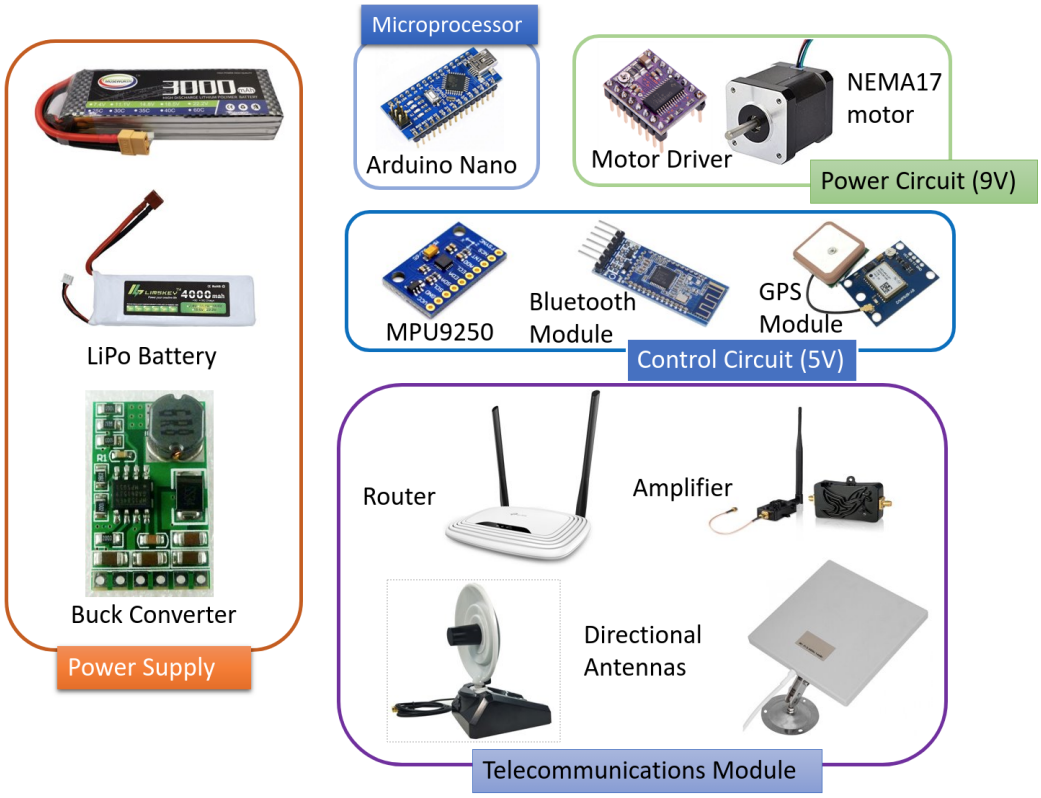


Figura 4.8: Diagrama de los diferentes módulos del circuito

Por último, el módulo de telecomunicaciones contará con tres elementos. En primer lugar, el *router*, al cual se conectarán tanto la estación de tierra (portátil o tablet) como el ordenador de a bordo del dron. Por otro lado, estará la antena, conectada directamente en la interfaz coaxial de este que, junto con un amplificador encargado de aumentar la potencia de la señal recibida, garantizarán una correcta recepción en tiempo real de la información capturada.

4.5.1.1 Dimensionamiento de las baterías

Para un dimensionamiento correcto del sistema de alimentación es de vital importancia conocer con detalle los elementos que más potencia consumen de cada circuito así como el tiempo medio de uso. Cabe destacar que el tiempo máximo de vuelo de un dron ronda los 20 minutos con un único juego de baterías. No obstante, se cuenta con tres juegos para recoger mayor cantidad de datos, con lo cual, el tiempo total de vuelo suma una hora aproximada. En la tabla 4.2 se recoge un desglose de los diferentes elementos del circuito mostrando el consumo de potencia de cada uno.

Elemento	Voltios (V)	Corriente (A)	Consumo energético (Wh)
Router	9	0.6	5.4
Amplificador	9	0.435	3.915
Motores y driver (2)	9	2.2	39.6
Arduino Nano	5	0.28	1.4
IMU	3.3	0.003	0.01
Módulo GPS	5	0.05	0.25
Módulo Bluetooth (kg)	5	0.004	0.02

Tabla 4.2: Consumos de potencia de los elementos de cada circuito.

En base a lo dispuesto en la tabla, se realiza la suma de consumos de potencia para cada circuito. Por una parte, el módulo de telecomunicaciones presenta un voltaje de alimentación de 9(V), con un consumo total de 9.315 (Wh). Por otro lado, en cuanto al circuito de movimiento de la antena, el circuito de control, alimentado a 5(V) precisa una demanda energética de 1.67(Wh), que, comparado con el circuito de potencia, es despreciable en la medida que este consume 39.6(Wh). Por este motivo, se puede realizar una aproximación concluyendo que el circuito de telecomunicaciones consume 10 (Wh) y el referido al sistema de apuntamiento 40 (Wh).

Las baterías que se emplearan para alimentar este sistema serán de tipo LiPo por ser las que se disponen en el laboratorio del IUMA y contar con una serie de ventajas muy provechosas. En primer lugar, su peso es menor que una de Li-Ion con la misma capacidad, llegando a ser hasta cuatro veces más ligera. En segundo lugar, permiten mantener mejor la carga nominal siempre y cuando el voltaje de la batería no disminuya de 3.7(V) por celda. Además, tienen la característica de poder construirse en diferentes tamaños y formatos, de forma que se pueden emplear baterías que se adapten perfectamente a la estructura del sistema. Por último, dado que son las baterías empleadas para los drones, se cuenta con los cargadores, conectores y experiencia necesarios para su uso.

Estas baterías están formadas por celdas, de forma que cada una suele tener un voltaje nominal de 3.7 (V) y uno máximo de 4.2 (V). Estas baterías, una vez cargadas, mantendrán su amperaje mientras van disminuyendo su voltaje de salida. En la figura 4.9 se recoge un ejemplar con una sola celda (4.9a) y otro con dos celdas (4.9b), indicando su voltaje nominal y su capacidad.

El hecho de que el voltaje disminuya una vez cargada la batería representa una desventaja para alimentar los circuitos, por ello se hace necesario emplear convertidores, de forma que a la salida de estos se pueda suministrar un voltaje constante al

(a) Batería de una celda

(b) Batería de dos celdas

Figura 4.9: Baterías de tipo LiPo

circuito que se alimenta, siempre vigilantes de que este no disminuya por debajo del nivel nominal para así preservar la salud de estas.

Con respecto al circuito de potencia, habida cuenta de que su tensión de alimentación es 9 (V) y su consumo de potencia 40 (Wh), se debe buscar una batería cuyo voltaje nominal no descienda de este valor y que proporcione una corriente que haga del producto con su tensión una potencia mayor a la necesitada por el circuito, esto es, mayor de 4.4 (A). Para el caso del módulo de telecomunicaciones, que también se alimenta con 9 (V) y consume 10 (Wh), el amperaje deberá ser superior a 1.1 (A).

4.5.2 Motores paso a paso

En el momento de elegir los actuadores del sistema que se desarrolla, existía la posibilidad de emplear motores de tipo servo o paso a paso. Se escogió esta segunda opción por las diversas ventajas que presentaba en términos de la aplicación que se implementa. En primer lugar, ofrecen un mayor número de polos (entre 50 y 100),

de forma que su movimiento se ejecuta en un sistema de bucle cerrado, permitiendo prescindir de un *encoder* para el control de su posición. Por otro lado, ofrecen un control de movimiento muy preciso, brindando un alto par a velocidades bajas, tienen un coste bajo y están ampliamente extendidos.

Por esta razón, el modelo de motor elegido ha sido el NEMA 17 que se recoge en la figura 4.10. Se emplearán dos unidades cuyos ejes se conectarán a las respectivas ruedas dentadas del sistema de antenas, permitiendo variar los ángulos de elevación y azimuth. En la siguiente sección se indica de qué forma se integrarán en la estructura diseñada.

Este motor cuenta entre sus características con una resolución de 200 pasos por vuelta, permitiendo un ángulo de rotación por cada paso de 1,8º trabajando en modo *full step*; opera a un voltaje nominal de 12 (V), con una corriente nominal por fase de 0.35 (A) y un peso de 0.22 (kg). De igual forma, este actuador debe ir acompañado de un driver que haga de interfaz entre el circuito controlador y el propio motor. Este se encargará de proveer la corriente necesaria que permita ejecutar los movimientos deseados. El modelo escogido ha sido el DRV8825 [27], que se muestra junto con su esquemático en la figura 4.11.

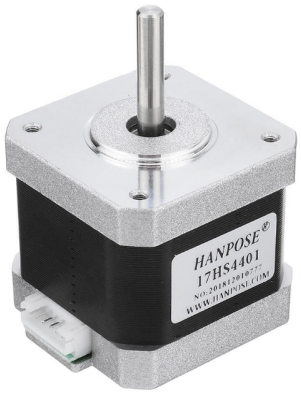
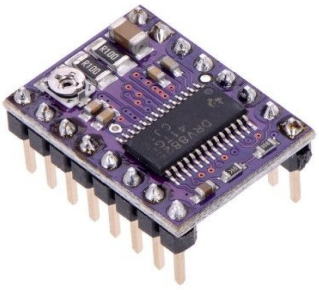


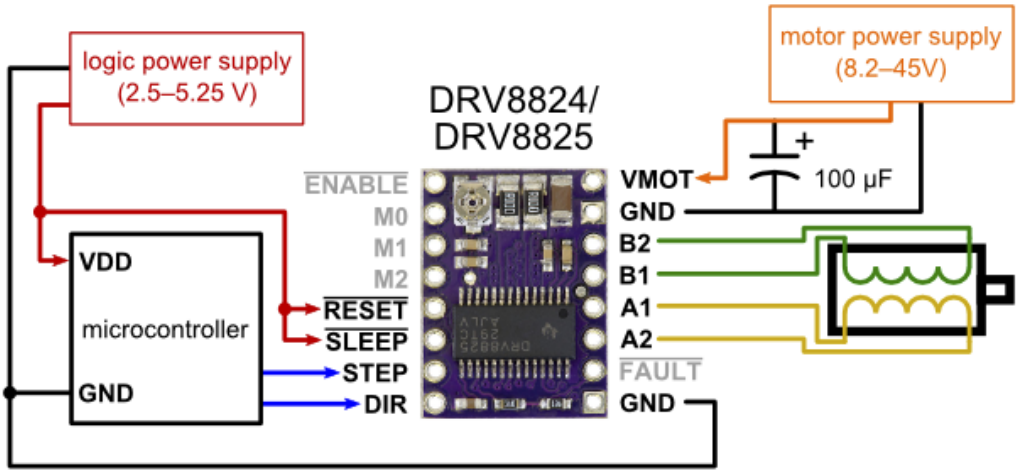
Figura 4.10: Motor paso a paso NEMA 17.

Como se puede apreciar, desde el controlador, en este caso la plataforma Arduino NANO, se controlan los pines de STEP y DIR, que permiten ejecutar los pasos y la dirección de estos respectivamente. De igual modo, se permite la posibilidad de establecer el movimiento mediante micropasos según los pines M0, M1 y M2, permitiendo 8 combinaciones posibles donde se incrementan en un factor x1, x2, x4, x8, x16 o x32 los pasos del motor dependiendo del valor especificado en estos pines. Con ello se consigue la posibilidad de establecer un movimiento mucho más preciso, alcanzando resoluciones de movimiento de hasta 0.056º por paso. Esta opción dota

al sistema de la capacidad de apuntar al sistema en vuelo de una forma mucho más eficaz, donde, habida cuenta de que la antena que se emplea es muy direccional, el hecho de mantener el enlace de comunicación tierra-dron estable es un factor clave para el propósito que se persigue.



(a) Integrado DRV8825



(b) Esquemático DRV8825

Figura 4.11: Driver empleado para el control de los motores

4.5.3 Unidad de Medida Inercial MPU9250

Las Unidades de Medida Inerciales, son dispositivos electrónicos cuya función es la de medir la aceleración lineal y angular empleando un acelerómetro y giróscopo, respectivamente. Ocasionalmente, estas IMU también incluyen la posibilidad de medir la intensidad de campo magnético empleando magnetómetros. La IMU que se ha escogido para este proyecto es el modelo MPU9250 [22]. Se trata de un módulo *multi-chip* que contiene dos dados integrados en un único encapsulado. Uno de ellos contiene el giróscopo y acelerómetro, de tres ejes cada uno, mientras que el otro únicamente incluye el magnetómetro, también de tres ejes. Esto hace de la IMU MPU9250, con la inclusión de un Procesador Digital de Movimiento (DMP), un dispositivo de seguimiento de movimientos. A su vez, integra un bus I2C para su conexión con un procesador encargado de la lectura de sus datos, que en este

caso será un ATmega328, integrado en una plataforma de desarollo Arduino Nano. De igual modo, para la digitalización de los tres ejes se incorporan convertidores analógico-digital (ADC) de 16 bits, permitiendo resoluciones de ±250, ±500, ±1000, ±2000 °/sec para el caso del giróscopo; ±2g, ±4g, ±8g y ±16g para el acelerómetro y ±4800 μT para el magnetómetro.

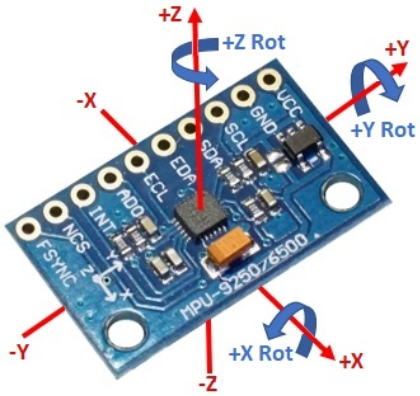


Figura 4.12: MPU9250 junto con un esquemático de sus ejes

La unidad empleada para el desarrollo de este proyecto es la que se muestra en la figura 4.12. En dicha imagen se aprecian los tres ejes sobre los que se cuantificarán las tres magnitudes medidas. Idealmente, si la IMU se encontrara sobre un plano horizontal, el acelerómetro mediría la fuerza de la gravedad en en el eje Z; el magnetómetro daría el valor del campo magnético terrestre en los ejes X e Y, dependiendo de la orientación de este y el giróscopo arrojaría un valor de cero, al no encontrarse sometido a ningún movimiento que ocasione una aceleración angular; no obstante, más adelante se analizará por qué esta situación no es inmediata y cómo se deben realizar una serie de consideraciones para asegurar que las medidas tomadas sean las correctas.

Este elemento proporciona un factor clave para lograr el propósito que se persigue, ya que permite conocer la orientación del dispositivo, que junto con la posición del UAV y la suya propia, permite calcular los ángulos necesarios que debe rotar la antena para apuntar al sistema de vuelo.

4.5.4 Cálculos de orientación de la antena

Como ya se ha comentado, los valores de aceleración lineal e intensidad de campo medidos por la IMU vendrán interpretados en torno a los ejes cartográficos XYZ. A partir de estos valores es necesario conocer la orientación del sistema. Para ello, se emplea un sistema de referencia global, particularmente, el de coordenadas NED,

que permite realizar una rápida conversión desde los ejes cartográficos tal y como se muestra en la imagen 4.13.

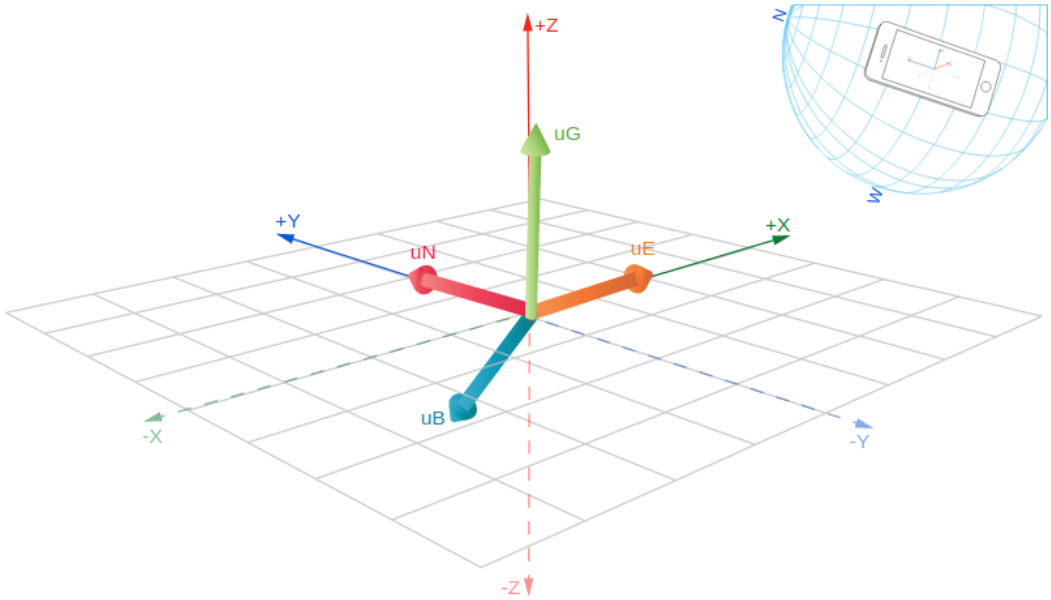


Figura 4.13: Cálculos para la matriz de Rotación

En dicha imagen, se ha dispuesto una IMU en un plano horizontal apuntando hacia el norte. Como se puede observar, existen dos fuerzas fundamentales a las que esta se ve sometida, el campo magnético y la gravedad, mostradas como *uB* y *uG* respectivamente. Cabe destacar el campo magnético se encuentra desviado con respecto al eje Y, esto tiene su explicación en la inclinación magnética terrestre que más adelante se detallará. Así, a partir de estos valores, es posible calcular sus coordenadas NED globales empleando la matriz de rotación, que básicamente consiste en calcular sendos productos vectoriales. En primer lugar, el producto vectorial de la componente de campo (*uB*) con respecto a la componente de aceleración (*uG*) dará como resultado la componente Este (*uE*). Ahora, el producto vectorial de la gravedad con la componente Este, dará como solución la coordenada Norte, y la componente *Down* será directamente la aceleración de la gravedad.

El código propio para hacer los cálculos de la matriz de rotación se muestra a continuación:

```
void CalculateAngles::CalculatePhaseAngleAccelerometerMagnetometer
(float* accelerometerData, float* magnetometerData, float* orientationData) {

    // Initial  data  for  operating
    float  RotationMatrix[SIZE][SIZE] = {{0.0, 0.0, 0.0},
                                         {0.0,  0.0,  0.0},
                                         {0.0,  0.0,  0.0}};

    float  resultVector[SIZE]         = {0.0,  0.0,  0.0};
    float  resultVectorAux[SIZE]      = {0.0,  0.0,  0.0};

    //R1 = crossProduct(crossProduct(A,M),A);
    CalculateAngles::CrossProduct(accelerometerData, magnetometerData,
        resultVector);
    CalculateAngles::custom_memcpy((void*)resultVectorAux, (void*)
        resultVector, SIZE*sizeof(float));
    CalculateAngles::CrossProduct(resultVectorAux, accelerometerData,
        resultVector);

    RotationMatrix[0][0] = resultVector[0];
    RotationMatrix[1][0] = resultVector[1];
    RotationMatrix[2][0] = resultVector[2];

    //R2 = crossProduct(A,M);
    CalculateAngles::CrossProduct(accelerometerData, magnetometerData,
        resultVector);
    RotationMatrix[0][1] = resultVector[0];
    RotationMatrix[1][1] = resultVector[1];
    RotationMatrix[2][1] = resultVector[2];

    //R3 = A;
    RotationMatrix[0][2] = accelerometerData[0];
    RotationMatrix[1][2] = accelerometerData[1];
    RotationMatrix[2][2] = accelerometerData[2];

    //Calculate Norm
    float  r1Norm = CalculateAngles::CalculateNorm(RotationMatrix, 0);
    float  r2Norm = CalculateAngles::CalculateNorm(RotationMatrix, 1);
    float  r3Norm = CalculateAngles::CalculateNorm(RotationMatrix, 2);

    //R1 = R1 / calculateNorm(R1);
```

```
    RotationMatrix[0][0] /= r1Norm;
    RotationMatrix[1][0] /= r1Norm;
    RotationMatrix[2][0] /= r1Norm;

    //R2 = R2 / calculateNorm(R2);
    RotationMatrix[0][1] /= r2Norm;
    RotationMatrix[1][1] /= r2Norm;
    RotationMatrix[2][1] /= r2Norm;

    //R3 = R3 / calculateNorm(R3);
    RotationMatrix[0][2] /= r3Norm;
    RotationMatrix[1][2] /= r3Norm;
    RotationMatrix[2][2] /= r3Norm;

}
```

Con esta matriz de rotación, se es capaz de obtener directamente las coordenadas globales. Así, ya se está más próximo a calcular la orientación del sistema. Para ello, se necesita especificar el sistema de referencia local (*Pitch, Roll y Yaw*), el cual se ha calcula empleando los ángulos de Euler [29], donde a través de sendas fórmulas trigonométricas se logra extraer estos valores angulares tal y como se muestra a continuación:

```
//Orientation from Rotation matrix (R)
    float theta = -asin(RotationMatrix[0][2]);
    float psi   = atan2(RotationMatrix[1][2]/cos(theta),RotationMatrix[2][2]/
        cos(theta));
    float rho   = atan2(RotationMatrix[0][1]/cos(theta),RotationMatrix[0][0]/
        cos(theta));

    float deg1 = rho * RADTODEG;
    float deg2 = theta * RADTODEG;
    float deg3 = psi * RADTODEG;

    //Orientation vector, results
    orientationData[0] = deg1;
    orientationData[1] = deg2;
    orientationData[2] = deg3;
```

De esta forma y dado que la antena deberá realizar una etapa de *homing*, donde se ubique en una posición de referencia, se dispondrá de dos IMU, una ubicada en

la parte móvil de la antena y otra ubicada en la base. Con ello se consigue realizar la lectura de los datos de orientación de la base y de la parte móvil de la antena con la finalidad de que ésta última se oriente hacia la misma dirección que la base, siendo ésta su posición de referencia. Es por ello que en nuestro sistema finalmente existirán el sistema de referencia global de coordenadas NED, y dos sistemas de referencia locales, uno por cada IMU.

Pero antes de realizar estos cálculos y obtener resultados satisfactorios, es necesario que los datos tomados tanto por el acelerómetro como por el magnetómetro sean los correctos y representen fielmente la disposición de la IMU.

4.5.4.1 Magnetómetro

Previamente a tomar las medidas del magnetómetro se deben especificar dos conceptos de gran importancia a tener en cuenta: inclinación y declinación magnética.

Se denomina inclinación magnética al ángulo que toma el campo magnético con respecto a un plano horizontal ficticio en ese punto de la tierra. Así, tal y como se aprecia en la figura 4.14, se tratan los diferentes ángulos de inclinación magnética en función de cada punto geográfico. Para Canarias, este ángulo oscila entre los 30 y 40 grados, esto es, si disponemos la IMU en un plano horizontal, donde únicamente tenga componente de aceleración en Z, perpendicular al plano y, de igual modo, disponemos el eje X apuntando hacia el norte magnético, la componente magnética que se obtendrá no será puramente en X, sino que formará un ángulo de entre 30 y 40 grados entre el eje X y el eje Y.

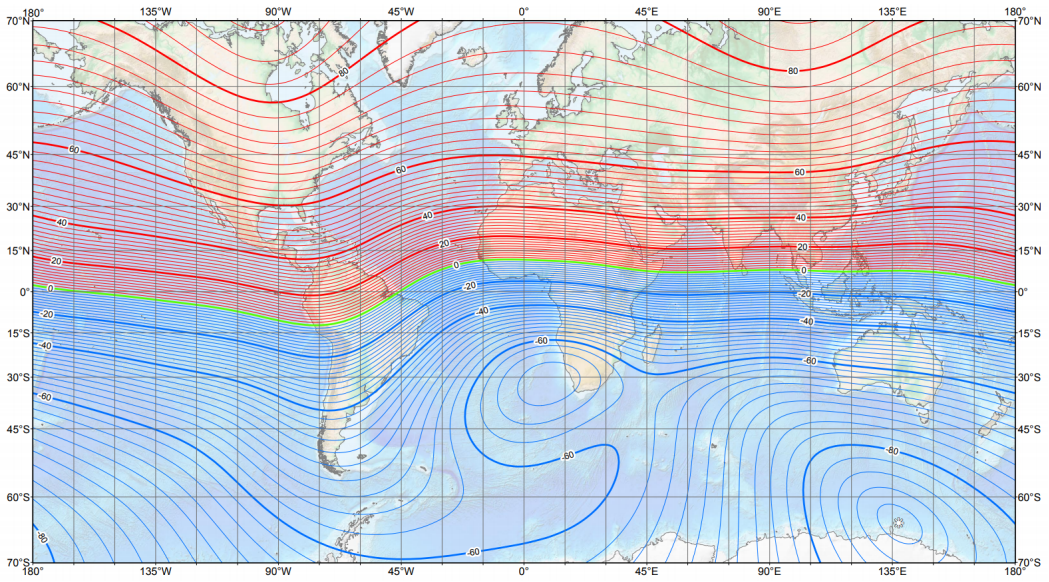


Figura 4.14: Diagrama de inclinación magnética en un mapa geográfico

Por otra parte, la declinación magnética comprende la diferencia entre el norte magnético local y el norte geográfico. Estas dos consideraciones deberán ser tomadas en cuenta para la toma de medidas del magnetómetro para obtener unos resultados satisfactorios que permitan unos cálculos correctos posteriormente.

Como se deduce, no prestar atención a estas casuística puede derivar en una toma de datos que pudiera parecer errónea. Otro caso a considerar es la distorsión magnética a la que pueden verse afectadas las medidas, existen dos tipos: *Hard Iron* y *Soft Iron* [7]. La primera hace referencia a aquella creada por objetos que produzcan un campo magnético, como puede ser un motor, mientras que la segunda es referida a metales que no producen campo pero que sí pueden variar las medidas tomadas, en tanto que distorsionarán la intensidad de campo medida en función de su dirección. Para este proyecto, la IMU ubicada en la parte móvil estará afectada por el metal de los motores, dándose una distorsión de tipo *soft*. No se dará una de tipo *hard* en la medida que la información será capturada antes de poner en funcionamiento a los motores. De esta forma, la calibración que se realice corregirá esta distorsión, asegurando una toma de medidas correcta.

La comunicación del procesador con la IMU se puede realizar mediante SPI o I2C. Si bien es cierto que la primera opción permite el empleo de interrupciones a costa de una velocidad de muestreo menor, se ha escogido la segunda alternativa por su facilidad de programación al considerar que las interrupciones no son de mayor interés para la realización de este proyecto. Asimismo, el bus I2C permite dos modos de operabilidad. El primero consiste en programarlo en modo *master*, donde se permite a la MPU9250 acceder a registros de sensores externos sin la necesidad de un sistema procesador; y, la segunda posibilidad consiste en programar este bus en modo *pass-through*, donde se permite que un procesador externo se comunique con los registros de la IMU directamente.

Los registros de lectura correspondientes al magnetómetro son los que se muestran en la figura 4.15. La medida de cada eje se guardará en dos registros, cada uno de 8 bits (1 byte), de forma que los registros HXL y HXH serán la medida del magnetómetro en el eje X, refiriéndose la L a *lower* y la H a *High*.

Addr	Register name	D7	D6	D5	D4	D3	D2	D1	D0
Read-only register									
03H	HXL	HX7	HX6	HX5	HX4	HX3	HX2	HX1	HX0
04H	HXH	HX15	HX14	HX13	HX12	HX11	HX10	HX9	HX8
05H	HYL	HY7	HY6	HY5	HY4	HY3	HY2	HY1	HY0
06H	HYH	HY15	HY14	HY13	HY12	HY11	HY10	HY9	HY8
07H	HZL	HZ7	HZ6	HZ5	HZ4	HZ3	HZ2	HZ1	HZ0
08H	HZH	HZ15	HZ14	HZ13	HZ12	HZ11	HZ10	HZ9	HZ8
Reset		0	0	0	0	0	0	0	0

Figura 4.15: Mapa de registros correspondiente al magnetómetro

Otro matiz a tener en cuenta para la lectura de estos valores es que se guardan en complemento a 2 y en formato *Little Endian*, esto es, el byte más significativo se posicionará al final y el menos representativo al principio. Además, el rango de medida de cada eje estará comprendido en el intervalo desde -32760 a 32760.

La lectura de los datos se ha realizado mediante el código que se muestra a continuación. En primer lugar, se activa el modo *pass-through*, anteriormente explicado; se habilita el magnetómetro escribiendo en su registro el valor uno; ahora se hace la lectura de siete registros, los 6 primeros correspondientes a los dos registros de cada eje y un último registro que indica la posibilidad de *overflow* en las lecturas. Estos valores se guardan en el vector *buffer*, pasado por parámetros a la función.

A continuación, se verifica si en la lectura de este último registro hay un 1 en la tercera posición, lo cual sería indicador de haber ocurrido *overflow*. Si no lo hay, la variable *resultOk* se actualiza a *true* y se guarda en el vector referido a cada eje su valor correspondiente llamando a la función *decodeMagnetometerBytes*.

```
void ReducedMPU9250::getMagnetometerReading(int16_t* x, int16_t* y, int16_t*
     z, bool* resultOk) {
    // Set i2c bypass enable pin to true to access magnetometer
    I2Cdev::writeByte(devAddr, MPU9250_RA_INT_PIN_CFG, 0x02);
    delay(10);
    // Enable the magnetometer
    I2Cdev::writeByte(MPU9250_RA_MAG_ADDRESS, 0x0A, 0x01);
    delay(10);
    // Read the 6 registers corresponding to the x, y, z axis and the Status
         one
    I2Cdev::readBytes(MPU9250_RA_MAG_ADDRESS,
         MPU9250_RA_MAG_XOUT_L, 7, buffer);
    // Verify that magnetic error overflow did not ocure
    uint8_t HOFL = buffer[6] & 0x08;
    if(HOFL == 0)
    {    resultOk[0] = true;
         x[0] = decodeMagnetometerBytes(buffer[1], buffer[0]);
         y[0] = decodeMagnetometerBytes(buffer[3], buffer[2]);
         z[0] = decodeMagnetometerBytes(buffer[5], buffer[4]);
    }else{
         resultOk[0] = false;
    }
}
```

El código de la función *decodeMagnetometerBytes* se muestra a continuación. En esta función se aborda la lectura de los registros que vienen en complemento a dos, de forma que la función recibe dos argumentos, *highPart* y *lowPart*. A la llamada del método, se debe especificar qué byte es el más significativo y cual menos, tratando la cuestión del *endianness*.Posteriormente, se verifica el bit más significativo del registro, *high*. En caso de ser cero, la conversión a valor decimal es inmediata. En el caso contrario, se cambian todos los bits de cada byte, se realiza la conversión a valor decimal, se suma uno a ese valor y se multiplica por -1.

```
int16_t ReducedMPU9250::decodeMagnetometerBytes(unsigned char highPart,
    unsigned char lowPart)
{
    int value;
    if( (highPart & 1<<7) == 0 )
    {
        value = ((int)(highPart) << 8) | (int)lowPart;
    }else{
        highPart = ~highPart;
        lowPart = ~lowPart;
        value = ((int)(highPart) << 8) | (int)lowPart;
        value = (-1)*(value+1);
    }
    return (int16_t)value;
}
```

Todo este código ha sido escrito en C con el propósito de hacer un diseño más modular y sencillo que permitiera su manejo en Arduino de una forma eficaz y rápida. Por lo tanto, para obtener datos del magnetómetro en Arduino, únicamente ha sido necesario declarar un objeto de la librería diseñada, los vectores para almacenar los valores de lectura y llamar al método indicado.

4.5.4.2 Acelerómetro

Por su parte, la lectura de los datos del acelerómetro es muy similar al caso anteriormente descrito con la salvedad de no emplear complemento a 2. En este caso, los bytes también se leen en *Little Endian*. La lectura de estos se realiza como se observa en la siguiente función.

```
void ReducedMPU9250::getAccelerometerReading(int16_t* x, int16_t* y, int16_t*
     z) {
    I2Cdev::readBytes(devAddr, MPU9250_RA_ACCEL_XOUT_H, 6, buffer);
    *x = (((int16_t)buffer[0]) << 8) | buffer[1];
    *y = (((int16_t)buffer[2]) << 8) | buffer[3];
    *z = (((int16_t)buffer[4]) << 8) | buffer[5];
}
```

Al igual que antes, se emplea el método *readBytes* de la librería I2Cdev, donde se accede al registro de lectura del eje X y se recorren los seis registros siguientes para almacenar los valores de medición de cada eje.

En Arduino la filosofía es similar a la lectura del magnetómetro, donde tras la declaración del objeto y los vectores donde se almacenan los datos, se hace una llamada al método dispuesto en la figura anterior.

4.5.4.3 Toma de medidas empleando Arduino

Una vez se ha definido el modo de lectura y las librerías empleadas para acceder a la información que ofrece la IMU, se procede a realizar la implementación en Arduino. Como se indicó, se ha empleado una plataforma Arduino Nano aunque potencialmente se podría usar cualquier otro modelo. La conexión de ambos dispositivos se muestra en el esquemático de la figura 4.16.

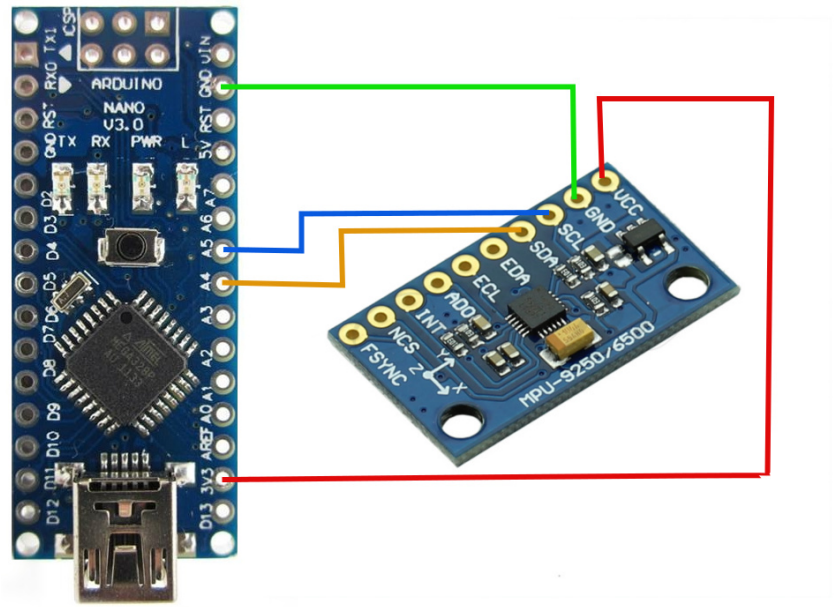


Figura 4.16: Conexión de Arduino Nano con MPU9250

Se han conectado los pines referidos a la alimentación y a tierra donde corresponde, y los pines SCL y SDA de la MPU9250 a los pines A5 y A4 del Arduino Nano, respectivamente. El pin SCL funciona como un reloj serie del I2C y el pin SDA se emplea para la lectura/escritura de los datos seriales de la MPU9250. A su vez, los pines A5 y A4 del Arduino Nano son los pines analógicos de este.

Para la toma de datos del magnetómetro se ha diseñado una pseudo-interfaz de usuario como la que mostrada en la figura 4.17.

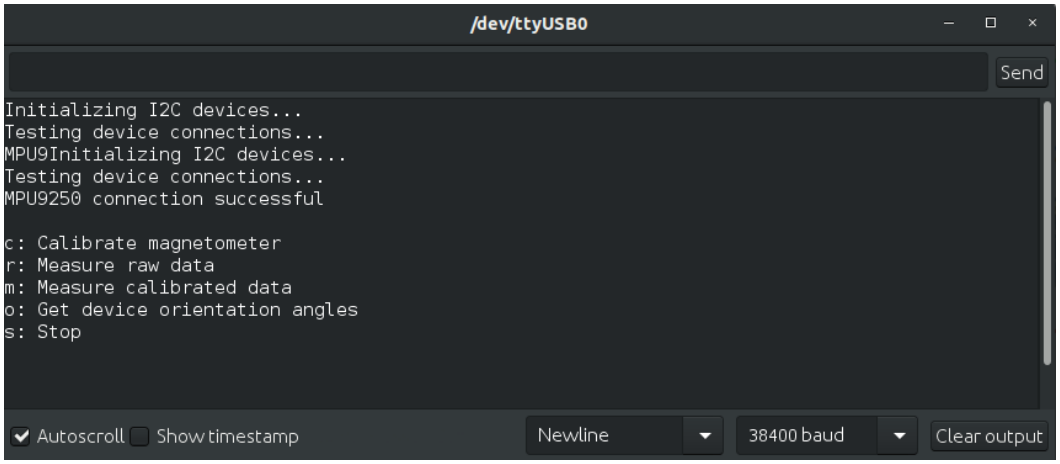


Figura 4.17: Pseudo-interfaz de usuario en Arduino

Aquí se le ofrece al usuario diferentes posibilidades: calibrar el magnetómetro, medir datos en crudo (*raw*), medir datos calibrados, medir los ángulos de orientación (*roll, pitch y yaw*) así como una opción empleada a detener las mediciones. De esta forma, si por ejemplo se escoge la opción 'r', se representa la información análogamente a la figura 4.18.

Estos valores son directamente tomados de los registros y convertidos a decimal. De forma relevante y habida cuenta de que se tomaron con la IMU apuntando al norte (el eje Y) sobre un plano horizontal, se puede apreciar que el mayor valor del magnetómetro se encuentra en el eje Y y la mayor componente de la aceleración se encuentra en el eje Z.

No obstante, esta referencia carece de mayor interés si no se toma en cuenta la calibración del magnetómetro, un factor de gran importancia para que las medidas sean las correctas.

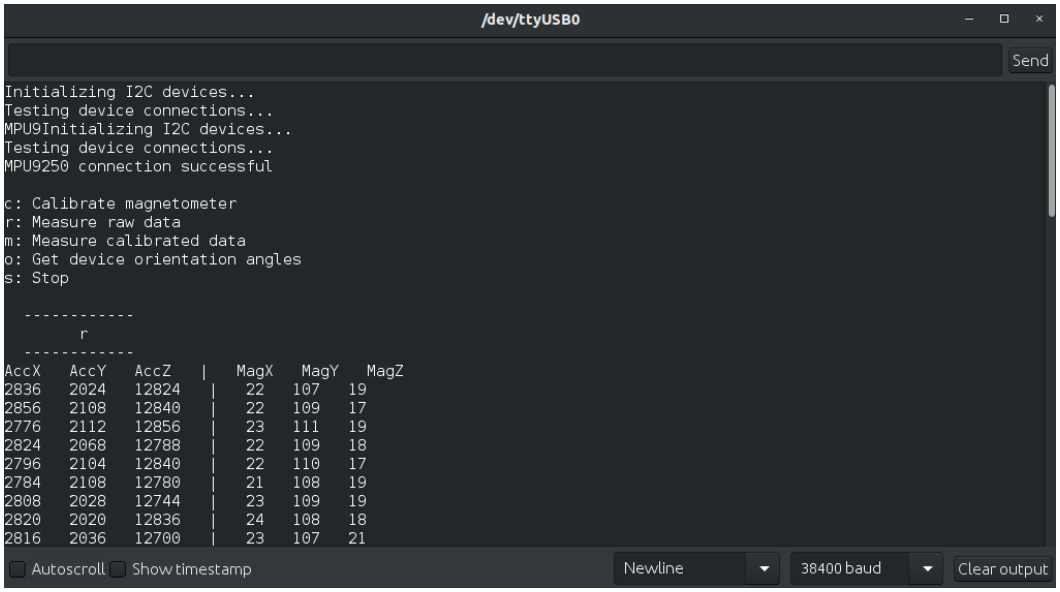


Figura 4.18: Lectura de datos crudos

4.5.4.4 Calibración del magnetómetro

Para calibrar la información provista por el magnetómetro se han tomado los valores en *raw* anteriormente mostrados, se han leído en Matlab, se han analizado y se han ejecutado una serie de operaciones, recogidas en el código a continuación adjunto.

En primer lugar, los datos en *raw* tomados para los tres ejes de la IMU se han centrado, restando su valor por la media de las medidas capturadas y guardado su valor actualizado en la variable *MagReference*. En segundo lugar, se dividen todos los elementos de este vector entre el valor máximo del mismo para obtener un escalado lineal entre -1 y 1, de forma que todas las componentes de cada eje estén comprendidas en este rango y se pueda hacer una comparación más eficaz entre valores crudos y calibrados.

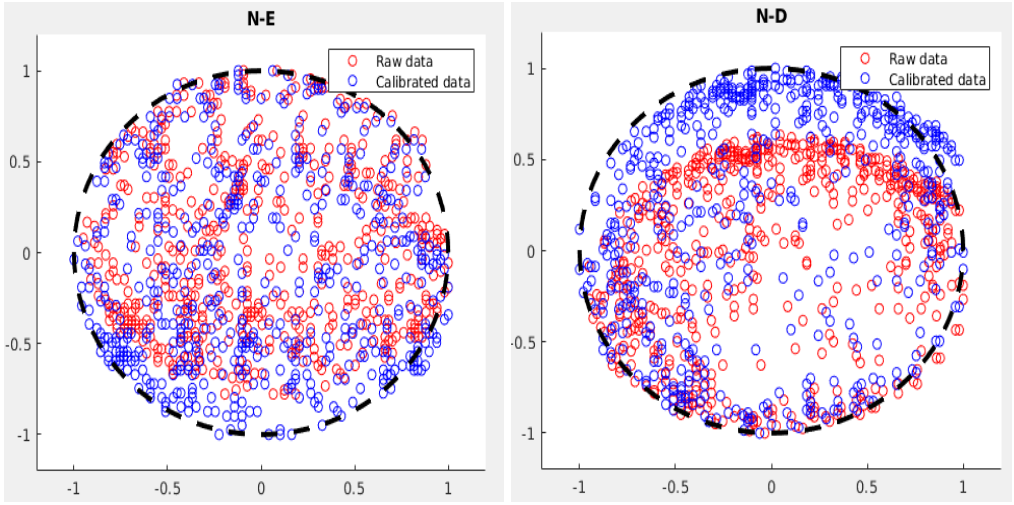
```
% Centralizing the Raw data for comparison
Mx = MagRaw(:,1);
My = MagRaw(:,2);
Mz = MagRaw(:,3);
MagReference = [Mx - mean(Mx(:)), My - mean(My(:)), Mz - mean(Mz(:))];
Mx = MagReference(:,1) / (max(abs(MagReference(:,1))));
My = MagReference(:,2) / (max(abs(MagReference(:,2))));
Mz = MagReference(:,3) / (max(abs(MagReference(:,3))));
MagReference = [Mx, My, Mz];
```

Habiéndose escalado los datos crudos, se procede a la calibración como tal. Se ha empleado el código adjunto. En primera instancia, se calcula el valor mínimo y el valor máximo de cada eje. A continuación, se realiza un escalado que permita tener los valores en el rango [0,1], siendo 0 el mínimo y 1 el máximo. Por último, en el vector *MagCalibrated* se resta 0.5 a todos los elementos de cada vector y se multiplica por 2, de forma que se transforma el rango de valores entre [-1, 1] y se está en disposición de hacer una comparación entre los datos crudos y calibrados.

```
% Calibrating the data
Mx = MagRaw(:,1);
My = MagRaw(:,2);
Mz = MagRaw(:,3);

valuesToAverage = 1;

maxValue = maxk(Mx, valuesToAverage);
minValue = mink(Mx, valuesToAverage);
MxR = (Mx - minValue) / (maxValue - minValue);
minX = minValue;
maxX = maxValue;

maxValue = maxk(My, valuesToAverage);
minValue = mink(My, valuesToAverage);
MyR = (My - minValue) / (maxValue - minValue);
minY = minValue;
maxY = maxValue;

maxValue = maxk(Mz, valuesToAverage);
minValue = mink(Mz, valuesToAverage);
MzR = (Mz - minValue) / (maxValue - minValue);
minZ = minValue;
maxZ = maxValue;

MagCalibrated = [MxR - 0.5, MyR - 0.5, MzR - 0.5]*2;
```

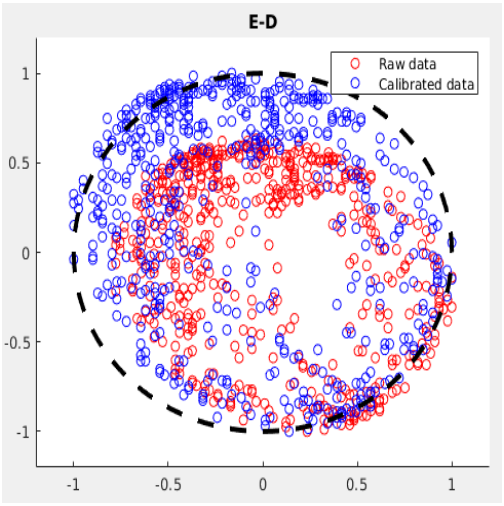
Una vez se han ejecutado estas operaciones, es posible representar los valores de cada vector. Los resultados son los que se muestran en la figura 4.19, estos se han dispuesto en coordenadas NED, refiriéndose a XYZ en cada uno de los ejes. En conclusión, se aprecia como los datos calibrados están correctamente distribuidos en el rango [-1,1], mientras que los datos crudos se encuentran mucho más concentrados. Con

ello, se puede afirmar que la calibración es correcta, y que se está en posición de obtener información fiable del magnetómetro. El siguiente paso es definir estas operaciones en una librería de Arduino para su incorporación en el código.



(a) N-E

(b) N-D



(c) E-D

Figura 4.19: Comparación de datos crudos y calibrados para el magnetómetro

En el código que a continuación se adjunta, se muestra el diseño de la función que se integrará en Arduino, donde se indica al usuario que mueva el dispositivo en todas las direcciones durante un tiempo en el que se tomarán 1000 valores distintos en búsqueda del mínimo y máximo de cada eje. Una vez se realiza este paso, lo siguiente es escribir en la memoria del procesador los datos hallados a la par que se muestran al usuario. Con esto se da por completa la calibración.

```
void ReducedMPU9250::CalibrateMagnetometer()
{
    Serial.println("\n\n Calibration started ");
    Serial.println(" Move the device in all directions for a while ");
    // Initialize the min and max values
    int16_t xMin;
    int16_t xMax;
    int16_t yMin;
    int16_t yMax;
    int16_t zMin;
    int16_t zMax;
    bool resultOk = false;
    while(!resultOk){
        getMagnetometerReading(&Mag[0], &Mag[1], &Mag[2], &resultOk);
    }
    xMin = Mag[0];
    xMax = xMin;
    yMin = Mag[1];
    yMax = yMin;
    zMin = Mag[2];
    zMax = zMin;
    // Calculate the min and max values
    for(long i=0; i<MAGNETOMETER_CALIBRATION_ITERATIONS; i++)
    {
        resultOk = false;
        while(!resultOk){
            getMagnetometerReading(&Mag[0], &Mag[1], &Mag[2], &resultOk);
        }
        if(Mag[0] > xMax){ xMax = Mag[0]; }
        if(Mag[0] < xMin){ xMin = Mag[0]; }
        if(Mag[1] > yMax){ yMax = Mag[1]; }
        if(Mag[1] < yMin){ yMin = Mag[1]; }
        if(Mag[2] > zMax){ zMax = Mag[2]; }
        if(Mag[2] < zMin){ zMin = Mag[2]; }
    }
    // Store the values as float in the class variable
    MagCalMax[0] = (float)xMax;
    MagCalMax[1] = (float)yMax;
    MagCalMax[2] = (float)zMax;
    MagCalMin[0] = (float)xMin;
    MagCalMin[1] = (float)yMin;
```

```
    MagCalMin[2] = (float)zMin;
    // Store the values in the persistent memory for future use
    WriteEepromCalibrationValues();
    // Display new calibration values
    Serial.println("    ");
    Serial.println(" New magnetometer calibration values: ");
    Serial.print(MagCalMin[0]);
    Serial.print("    ");
    Serial.print(MagCalMax[0]);
    Serial.print(" | ");
    Serial.print(MagCalMin[1]);
    Serial.print("    ");
    Serial.print(MagCalMax[1]);
    Serial.print(" | ");
    Serial.print(MagCalMin[2]);
    Serial.print("    ");
    Serial.println(MagCalMax[2]);
    Serial.println("");
}
```

Llegados a este paso, se hace necesaria una función que lea los valores del magnetómetro ya calibrados, para este propósito se ha diseñado el método mostrado en el siguiente código.

```
void ReducedMPU9250::getMagnetometerCalibratedData(float *x, float *y, float
    *z, bool* resultOk)
{
    getMagnetometerReading(&Mag[0], &Mag[1], &Mag[2], resultOk);
    if(resultOk[0])
    {
        x[0] = 2*( ((float)Mag[0] - MagCalMin[0]) / (MagCalMax[0] -
            MagCalMin[0]) - 0.5);
        y[0] = 2*( ((float)Mag[1] - MagCalMin[1]) / (MagCalMax[1] -
            MagCalMin[1]) - 0.5);
        z[0] = 2*( ((float)Mag[2] - MagCalMin[2]) / (MagCalMax[2] -
            MagCalMin[2]) - 0.5);
    }

}
```

En la figura 4.20, se muestra el resultado del código desarrollado, donde el usuario ha escogido la opción de calibración del magnetómetro, tipada con la letra C, ha movido el dispositivo durante un tiempo y, terminado el proceso, se muestran los valores máximos y mínimos. Posteriormente, se escoge la opción de medición de datos calibrados, obteniendo los valores esperados según la orientación del dispositivo. Igualmente se muestran las medidas de la aceleración, a la izquierda el valor normalizado y a la derecha el valor medido en m/s^2.

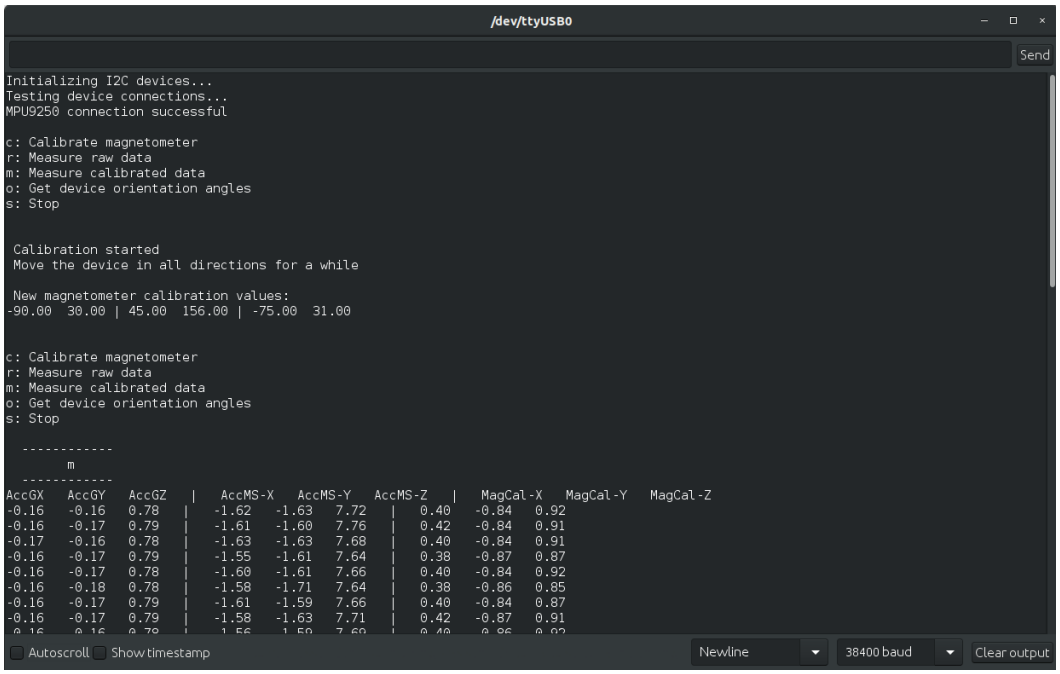


Figura 4.20: Interfaz de Arduino donde se realiza una calibración y medición de datos

A modo de conclusión, en la figura 4.21 se muestran los datos mostrados por el Arduino. Para comprobar el correcto funcionamiento se rota el dispositivo según los ángulos de *Pitch, Roll y Yaw* anteriormente nombrados y se aprecia como el valor angular de cada eje varía correctamente a la par que la orientación del dispositivo. Con todo, se verifican los pasos de calibración, cálculos de matriz de rotación y ángulos de Euler.

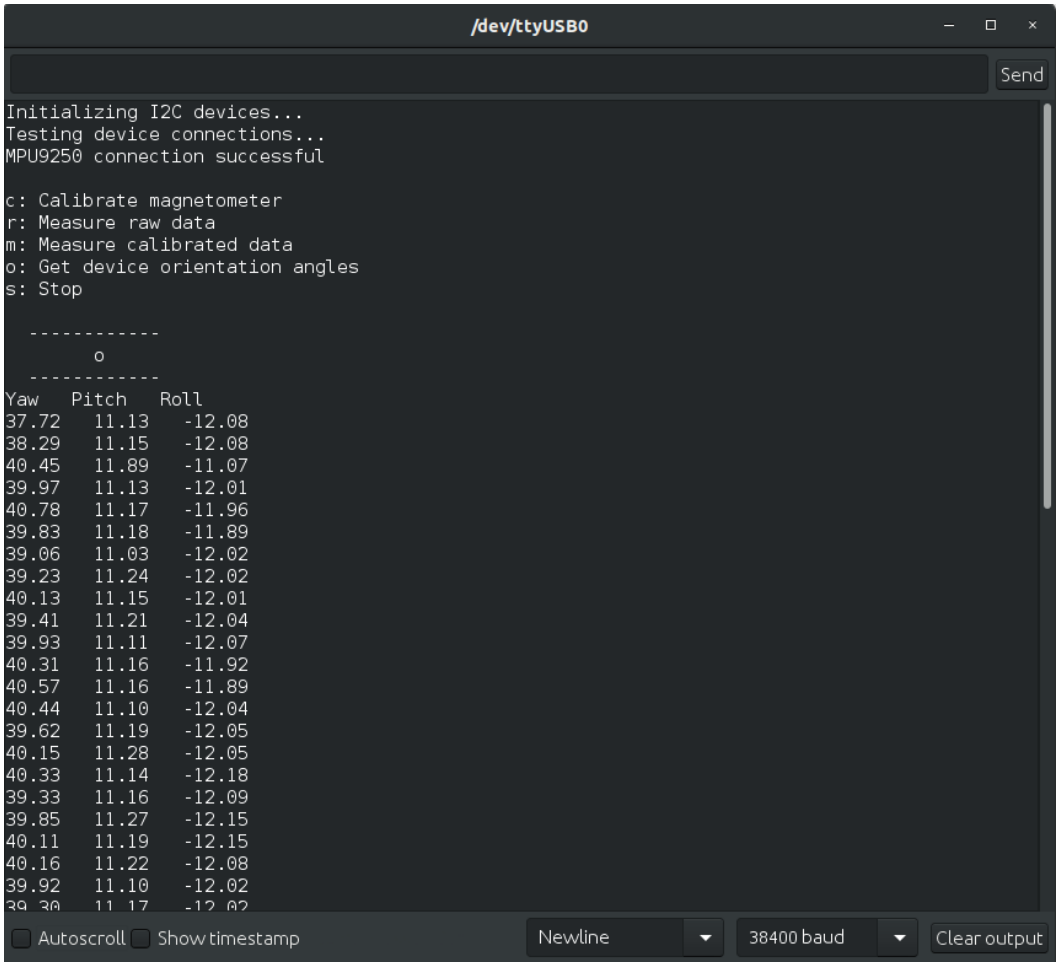


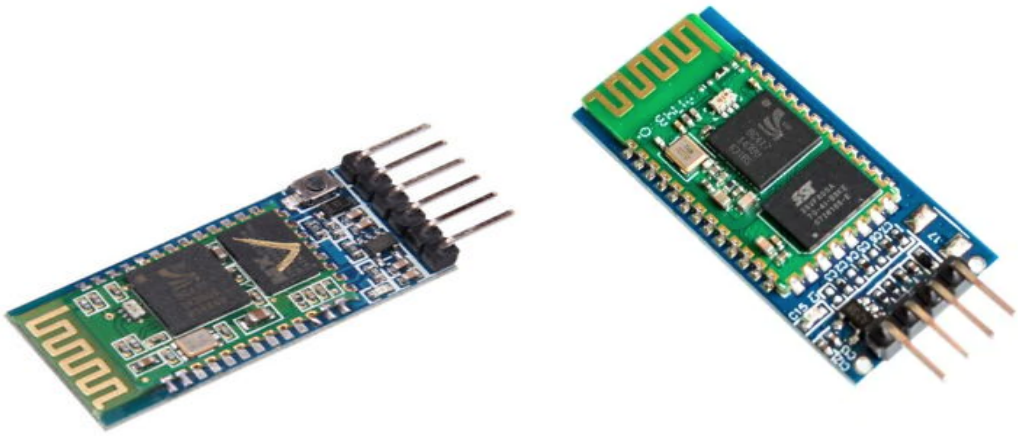
Figura 4.21: Interfaz de Arduino donde se realiza una calibración y medición de datos

Finalmente, cabe destacar que la calibración del acelerómetro es notablemente más sencilla que la del magnetómetro, dado que únicamente se ubica en un plano donde se vaya moviendo en torno a sus ejes (horizontal, vertical, de costado) y midiendo la aceleración de la gravedad. En el caso de no ser la esperada, se deberá modificar el offset de cada registro para lograr los valores deseados.

4.5.5 Módulo de comunicación Bluetooth

A la hora de escoger los dispositivos de comunicación bluetooth se debe especificar un maestro y un esclavo. El primero se incorporó en la estación de tierra mientras que el segundo se instaló en el sistema de apuntamiento. De esta forma, sería el maestro el que estableciera la comunicación con el esclavo mediante un *handshake* típico. Pero antes de poder realizar la comunicación se requiere la configuración de ambos dispositivos para su correcto funcionamiento.

En primer lugar, es necesario diferenciar adecuadamente cada uno de los dispositivos. En la figura 4.22a, se encuentra un ejemplar del dispositivo maestro, y un ejemplar del esclavo en la imagen 4.22b. La principal diferencia física entre uno y otro es la inclusión de seis pines de configuración en el primero y cuatro en el segundo.



(a) Módulo maestro HC05 **(b)** Módulo esclavo HC06

Figura 4.22: Dispositivos bluetooth empleados para la comunicación

Por su parte, el módulo HC05 contendrá dos pines de alimentación, dos de comunicación (TXD y RXD) y dos pines de configuración (STATE Y KEY). Por otra parte, el esclavo únicamente contendrá los pines de alimentación y comunicación. El propósito que se persigue con estos dos dispositivos es configurarlos de forma tal que funcionen como un puerto serie al uso, de forma que una vez configurados, hagan de pasarela donde reciben un dato por su pin correspondiente y automáticamente lo envían al otro dispositivo.

La configuración de estos dispositivos se realiza mediante comandos AT, de forma que se conecta cada dispositivo a un convertidor USB-TTL como el mostrado en la figura 4.23, donde se conecta el pin TX del bluetooth al RX del conversor y viceversa, de modo que se establezca una conexión por puerto serie y se especifiquen los nombrados comandos AT con la finalidad de configurar los dispositivos.

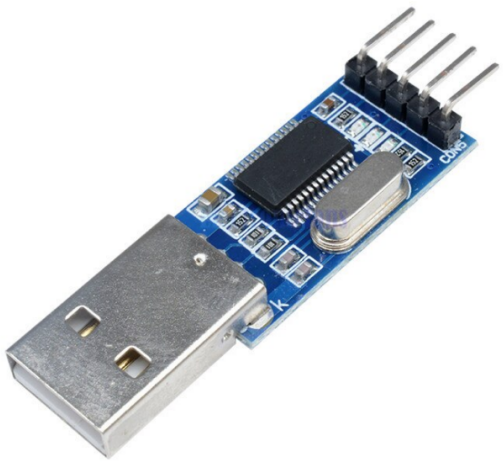


Figura 4.23: Conversor TTL a USB

Una vez realizada la conexión física, los comandos que se emplean son los siguientes:

- AT: comando para iniciar el saludo, devuelve OK
- AT+NAME: devuelve el nombre del dispositivo
- AT+NAME=NombreDispositivo: permite cambiar el nombre del dispositivo
- AT+UART=BaudRate: especifica un *baud rate* de funcionamiento específico

De esta forma, se configuran ambos dispositivos con los mismos parámetros con la finalidad de establecer una comunicación serie sencilla tanto en la parte referida a la estación de tierra como en la antena. En el primer caso, irá directamente conectado a un conversor TTL y en el segundo caso, se realizará la conexión empleado la interfaz de comunicación serie del Arduino Nano y la librería dedicada *SoftwareSerial*, que permite crear un objeto donde a través de la llamada a diversos métodos se gestiona esta comunicación.

4.5.6 Módulo de posicionamiento GPS

Para el cálculo de la ubicación del sistema de antenas, se ha empleado el GPS mostrado en 4.24, se trata de un módulo de la marca U-Blox, incluye una antena de parche, tiene capacidad de llevar un *tracking* de hasta 22 satélites en 50 canales, caracterizado por tener una gran sensibilidad y un consumo muy reducido de 45

mA en funcionamiento. De igual modo, incluye la posibilidad de configurarlo en modo de bajo consumo, disminuyendo su amperaje a 11 mA. Esta característica es de gran interés en la medida que la aplicación que se desarrolla estará constantemente alimentada por una batería, y disminuir el consumo es siempre un objetivo a perseguir.

Figura 4.24: GPS U-Blox NEO-6M

En cuanto a su modo de funcionamiento, incluye un led que se encontrará parpadeando cada segundo en el caso de haber encontrado los datos de posicionamiento, y en el caso de no parpadear indicará que aún se encuentra buscando satélites. Como se puede observar, consta de cuatro pines fundamentales, dos de alimentación y dos comunicación (TX y RX), estos dos pines permitirán establecer una comunicación serie donde el módulo GPS estará constantemente enviando una serie de cadenas que empezarán con los siguientes identificadores típicos: GPGGA, GPRMC y GPGLL. Estas cadenas deberán ser interpretadas para el cálculo de la longitud y latitud correspondientes.

En tanto que el módulo GPS iría directamente conectado al Arduino, existían dos posibilidades, por un lado, interpretar estos datos en el propio microprocesador, y enviar por bluetooth la información ya interpretada, algo que aumentaría el consumo de potencia y aumentaría el uso de memoria. Se desechó esta opción y se optó por multiplexar los pines de bluetooth y GPS, de forma que cuando la estación de tierra solicitara datos de ubicación, el pin transmisor del GPS enviara datos directamente al bluetooth esclavo, de forma que esta información fuera procesada en la estación de tierra, subsanando, por tanto, la problemática indicada.

Para ello, se diseñó una clase Python encargada de interpretar estos datos, la parte más importante de este desarrollo se muestra en el siguiente código:

```
# ---------- RMC Decoding methods ---------- #

    def decodeRMC(self, entireLine):
        subLines = entireLine.split(",")
        latitudeValueString = subLines[3]
        latitudeOrientationString = subLines[4]
        longitudeValueString = subLines[5]
        longitudeOrientationString = subLines[6]
        if latitudeValueString != '' and latitudeOrientationString != '' and
            longitudeValueString != '' and longitudeOrientationString != '':
            self.Latitude_RMC = self.getLatitude(latitudeValueString,
                latitudeOrientationString)
            self.Longitude_RMC = self.getLongitude(longitudeValueString,
                longitudeOrientationString)
            if self.verbose:
                print('RMC (Lat, Long) = ({}, {})'.format(self.Latitude, self.
                    Longitude))
        else:
            self.Latitude_RMC = None
            self.Longitude_RMC = None
            if self.verbose:
                print('RMC no coordinates')



    # ---------- GGA Decoding methods ---------- #

    # TODO: Decode altitude here
    def decodeGGA(self, entireLine):
        subLines = entireLine.split(",")
        if self.verbose:
            print("GGA Altitude: ", subLines[9], subLines[10],sep="")
        latitudeValueString = subLines[2]
        latitudeOrientationString = subLines[3]
        longitudeValueString = subLines[4]
        longitudeOrientationString = subLines[5]
```

```
        if latitudeValueString != '' and latitudeOrientationString != '' and
            longitudeValueString != '' and longitudeOrientationString != '':
            self.Latitude_GGA = self.getLatitude(latitudeValueString,
                latitudeOrientationString)
            self.Longitude_GGA = self.getLongitude(longitudeValueString,
                longitudeOrientationString)
            if self.verbose:
                print('GGA (Lat, Long) = ({}, {})'.format(self.Latitude, self.
                    Longitude))
        else:
            self.Latitude_GGA = None
            self.Longitude_GGA = None
            if self.verbose:
                print('GGA no coordinates')



    # ---------- GLL Decoding methods ---------- #

    def decodeGLL(self, entireLine):
        subLines = entireLine.split(",")
        latitudeValueString = subLines[1]
        latitudeOrientationString = subLines[2]
        longitudeValueString = subLines[3]
        longitudeOrientationString = subLines[4]
        if latitudeValueString != '' and latitudeOrientationString != '' and
            longitudeValueString != '' and longitudeOrientationString != '':
            self.Latitude_GLL = self.getLatitude(latitudeValueString,
                latitudeOrientationString)
            self.Longitude_GLL = self.getLongitude(longitudeValueString,
                longitudeOrientationString)
            if self.verbose:
                print('GLL (Lat, Long) = ({}, {})'.format(self.Latitude, self.
                    Longitude))
        else:
            self.Latitude_GLL = None
            self.Longitude_GLL = None
            if self.verbose:
                print('GLL no coordinates')
```

Aquí se realiza la decodificación de las cadenas, donde se cargan los datos de latitud y longitud correspondientes a cada cadena para, posteriormente, realizar un cálculo de las mismas más preciso. Tanto la gestión del GPS como del Bluetooth fueron integradas en sendas clases con la finalidad de ser importadas desde un programa *Main* que llamara a los métodos de cada una, resultando en un diseño muy modular y escalable.

4.6 Interconexión del sistema completo

Una vez se han abordado los pormenores del sistema diseñado, se procede a mostrar la interconexión del conjunto. Por motivo de simplificación, se muestra cada módulo conectado de forma individual. En la imagen 4.25 se muestra la conexión de los módulos de bluetooth y GPS, de forma que se multiplexan el pin de transmisión del GPS junto con el pin digital D3 de Arduino, la salida del multiplexor se conecta al pin de recepción del bluetooth. Con este proceder se logra el envío de datos GPS directamente por medio del bluetooth cuando la estación de tierra lo solicita, encargándose esta de la decodificación de los respectivos datos.

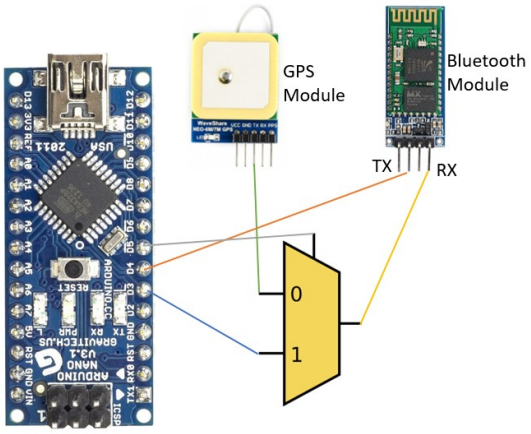


Figura 4.25: Conexión de los módulos de bluetooth y GPS

Por otro lado, se recoge en la imagen 4.26 la conexión entre los *drivers* DRV 8825, el Arduino nano y los motores NEMA 17, se resalta la inclusión de los *shield* donde se conectan los *drivers* que permiten una conexión sencilla entre los módulos por medio de pines macho-hembra.

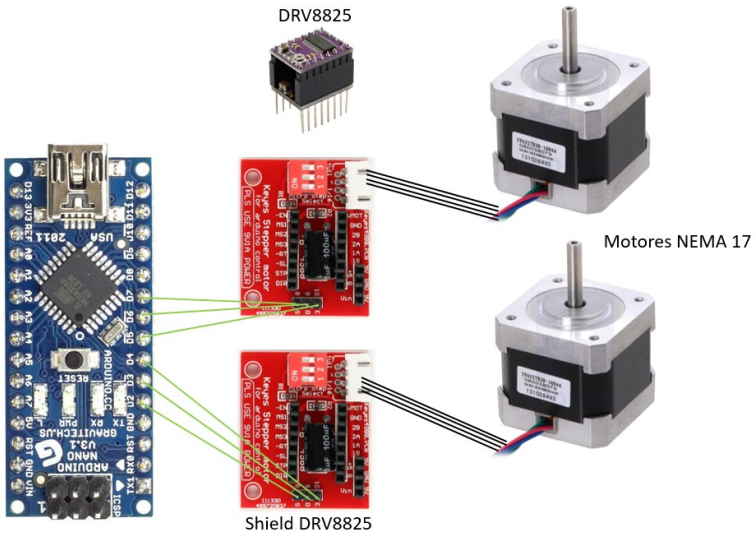


Figura 4.26: Interconexión de los driver y sus motores

Finalmente, en la imagen 4.27, se muestra la conexión de las dos IMU, la ubicada en la base y en la parte móvil, que se conectarán ambas a los puertos I2C del Arduino y se comunicarán por diferentes direcciones cada una.

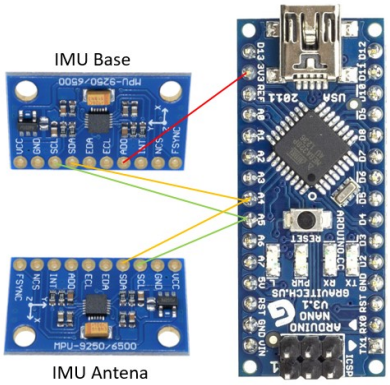


Figura 4.27: Conexión de las IMU

4.7 Lógica de comunicación entre la estación de tierra y el sistema de antenas

Para finalizar, una vez que se ha verificado el funcionamiento de cada componente por separado, el siguiente paso es el diseño de la interfaz de comunicación, que permita ejecutar órdenes desde el segmento de tierra hacia el sistema de apuntamiento. De esta forma, se debe tener en cuenta una serie de requisitos. En primer lugar, se debe tener en cuenta la necesidad de desarrollar un código muy modular, que permita la verificación de cada uno de las etapas por las que debe pasar el sistema. Por otro lado, es muy importante que la ejecución de cada tarea no sea bloqueante en la estación de tierra, en la medida que esta se encontrará no solo recibiendo y enviando datos a la antena, sino que también estará en continua comunicación con el UAV en vuelo, con lo cual este aspecto es vital.

Así, el desarrollo de esta comunicación se ha realizado en Python, en la imagen 4.28 se muestra la interfaz de comandos resultante.

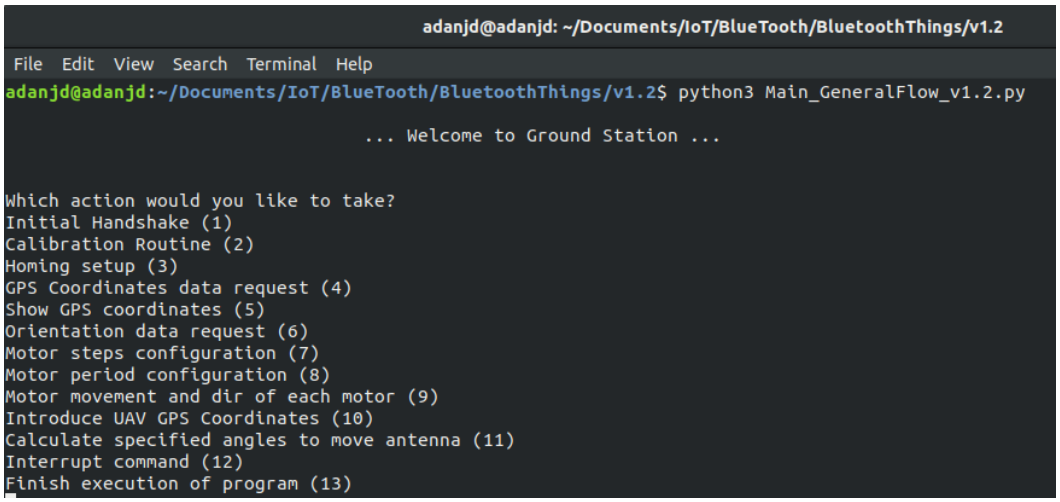


Figura 4.28: Interfaz de comunicación de la estación de tierra

A continuación, se pasa a enumerar y detallar las opciones mostradas:

1. *Handhsake* inicial: primer paso donde se establece comunicación con la antena.

2. Rutina de calibración: se ejecuta la orden donde se pasa al estado de calibración en el que el usuario debe mover la antena y la base en todas las direcciones posibles

3. Rutina de *homing*: se envía la orden de posicionar la antena en una situación de referencia a partir de la cual se puedan realizar los movimientos futuros necesarios

4. Petición de datos de ubicación: se solicita a la antena los datos de ubicación, que son enviados, decodificados y guardados en la estación de tierra

5. Petición de datos de orientación: se envían los datos de ubicación provenientes de la IMU dispuesta en la base

6. Configuración de los pasos del motor

7. Configuración de la velocidad (período) del motor

8. Orden para ejecutar movimientos de los motores y especificar la dirección de cada uno de ellos

9. Petición de las coordenadas de latitud y longitud del UAV

10. Cálculo de los ángulos que debe moverse la antena para apuntar al dron

11. Interrupción de la orden que se esté ejecutando en segundo plano

12. Finalización del programa

Como se aprecia, estas opciones están dispuestas a modo de depuración, de forma que una vez se haya verificado el sistema en plenitud, se podrán prescindir alguna de ellas como son las de movimientos en motores. Además, se respetará la lógica de estados señalada anteriormente, donde, por ejemplo, no se pueda apuntar al dron en vuelo sin haber realizado los pasos previos de calibración y/o homing.

5 Resultados y líneas futuras

En este capítulo se tratan los aspectos relativos a los resultados derivados de las diferentes partes del proyecto. Se desglosará en dos secciones diferenciadas. Por un lado, se tratará la parte referida a la compresión de imágenes y por otro, se abordarán los aspectos relativos a la parte de comunicaciones. Para esbozar los resultados de cada sistema, se detallarán una serie de experimentos que han permitido verificar el correcto funcionamiento de cada módulo, de forma que una vez verificados de forma independiente, se esté en disposición de realizar la interconexión del sistema completo.

5.1 Resultados del desarrollo del sistema de compresión

Para evaluar los resultados del proceso de compresión de las imágenes, se han diseñado varios experimentos que permitan justificar cada uno de los avances que se han realizado. Como ya se ha comentado, las diferentes pruebas se han ejecutado tanto en la placa de desarrollo Jetson Nano como en la Jetson Xavier NX. Cabe recordar que los *frames* espectrales de la Jetson Nano son almacenados en un dispositivo SSD con interfaz SATA-USB3.0 mientras que para la Jetson NX se almacenan en una tarjeta SD. De igual forma, se resalta que mientras la versión destinada al dispositivo Jetson Nano necesita de un adaptador WiFi para poder conectarse a un router, en la versión enfocada a la Jetson NX, la antena viene directamente incluida en la placa. Todo esto permitirá hacer un estudio comparativo y escoge una opción u otra en función de las necesidades.

Concretamente, los dispositivos de red empleados en estas pruebas se muestran en la figura 5.1, el adaptador WiFi [34] que se ha instalado en la Jetson Nano se recoge en la Imagen 5.1a. Dicho adaptador se conecta a la placa con una interfaz USB2.0. El *router* empleado para la transferencia de los datos comprimidos se muestra en la figura 5.1b, que a su vez incorpora las versiones B, G y N del estándar 802.11[35].

(a) Adaptador de red **(b)** Router

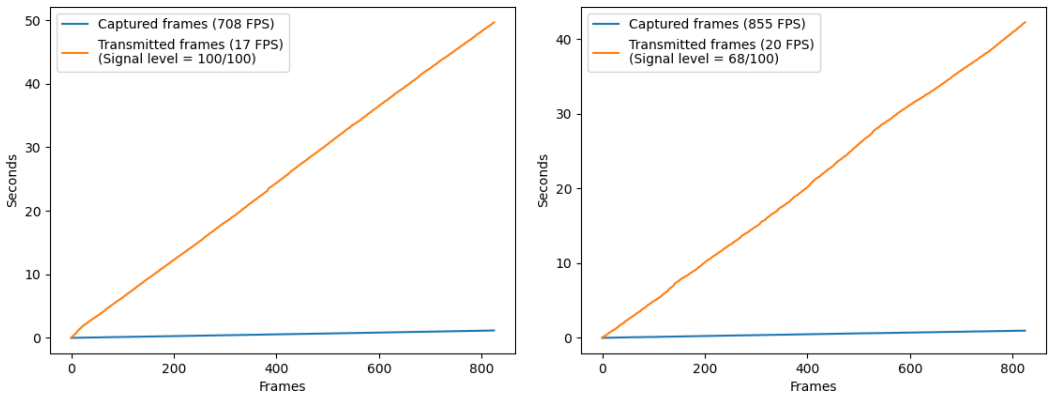
Figura 5.1: Dispositivos de red empleados para ejecutar las pruebas

5.1.1 Análisis del envío de datos sin compresión

En primer lugar, se ha procedido a evaluar la posibilidad de realizar un envío de los datos en crudo, esto es, según la cámara los captura. Para ello, se ha escogido una imagen hiperespectral ya capturada, se ha dividido en sus respectivos *frames*, o líneas de píxeles capturadas en un instante por el sensor hiperespectral durante su barrido, y se ha diseñado un proceso que se encarga de copiar estos *frames* en una nueva carpeta, de tal forma que se simule el proceso de captura.

En primer lugar, se recogen los resultados obtenidos con la Jetson Nano en la figura 5.2a, donde se constata cómo la velocidad de envío dista mucho de la esperada, obteniendo un *frame rate* de envío de 17 FPS. Si se considera que cada *frame* está formado por 1024 píxeles, 160 bandas y 16 bit por píxel, se obtiene un *baudrate* de envío que asciende a 44 Mbps.

Por otro lado, en la figura 5.2b, se recoge el mismo experimento pero realizado sobre la Jetson NX. Este varía ligeramente con respecto al anterior en la cobertura de la señal WiFi, siendo aquí de 68 % con respecto al 100 % del caso anterior. Asimismo, la velocidad de captura de los frames es notablemente más rápida, obteniendo 855 FPS frente a los 708 FPS del caso anterior; algo que responde a la mayor capacidad de cómputo presente en la versión NX, que incorporando un procesador más avanzado contribuye a la aceleración de esta tarea. Además, la velocidad de transmisión alcanza 20 FPS, ligeramente superior al caso anterior pero sin ser aún suficiente para el objetivo del proyecto.



(a) Jetson Nano **(b)** Jetson NX

Figura 5.2: Análisis de envío

Finalmente, cabe destacar que se han realizado tres iteraciones de cada experimento en cada dispositivo, estas se muestran en la tabla 5.1.

Experimento Jetson Nano

Test	Captura (FPS)	Transmisión (FPS)	Cobertura WiFi (%)	Tasa de envío (Mbps)
1	708	17	100	44,56
2	730	17	100	44,56
3	728	16	100	41,94
Media	722	16,66	100	43,68

Experimento Jetson NX

Test	Captura (FPS)	Transmisión (FPS)	Cobertura WiFi (%)	Tasa de envío (Mbps)
1	907	20	68	52,42
2	855	20	68	52,42
3	880	19	67	49,8
Media	880,67	67,67	100	51,55

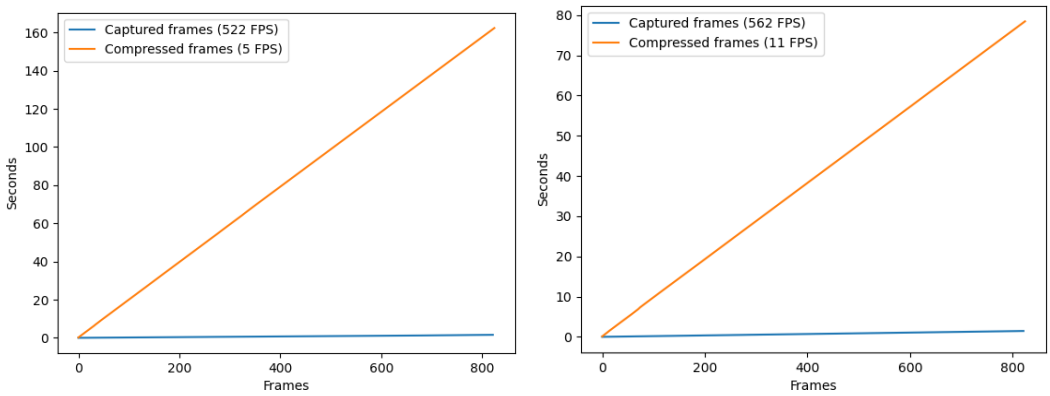
Tabla 5.1: Experimentos de captura y transmisión de *frames*

A modo de conclusión, se establece que la Jetson NX obtiene una mayor velocidad de captura y mayor tasa de *frames* enviados con una menor cobertura WiFi. Este hecho tiene su porqué en las mejoras de capacidad computacional de esta versión con respecto a la Nano. Sin embargo, estos resultados no son concluyentes en vistas a los objetivos del proyecto, resaltando la necesidad de compresión para lograr un envío en tiempo real. También cabe destacar que las velocidades de captura alcanzadas hacen referencia únicamente a la velocidad máxima a la que cada placa mueve los archivos correspondientes a cada *frame* de una carpeta a otra cuando se simula el proceso de captura. No obstante, en una situación real, la velocidad de captura típicamente oscila entre 100 y 150 FPS, de acuerdo con las necesidades de la misión que se esté llevando a cabo.

5.1.2 Resultados de la implementación del compresor en serie

Habiéndose corroborado que el ancho de banda no es el suficiente para realizar el envío de tal cantidad de información cumpliendo los requisitos temporales, se hace necesaria la compresión de estas imágenes. Para realizar este experimento, se ha realizado la implementación serie del compresor HyperLCA con la finalidad de analizar si cumple las restricciones indicadas. De esta forma, se han variado los parámetros de entrada referidos al ratio de compresión y el número de bits usados para escalar cada vector de proyección (Nbits o drProj). En cuanto a este último parámetro, cabe destacar que a mayor número de bits, menor cantidad de iteraciones habrá de realizar el compresor, luego más rápido comprimirá los distintos *frames*. Para proceder, se han analizado los dos casos más extremos, esto es, menor ratio de compresión junto con menor número de bits (drProj), y mayor ratio de compresión y mayor número de bits para escalado de vectores.

Así, en la imagen 5.3 se muestran estas dos iteraciones. En la figura 5.3a se ha establecido el ratio de compresión a 12 y el valor de *drProj* a 8, obteniéndose un *frame rate* de compresión que asciende a 5 FPS; mientras que en la imagen 5.3b se establece un ratio de compresión de 24 y el valor de *drProj* a 12, alcanzando un *frame rate* de 11 FPS, algo superior en tanto que la compresión es mayor, luego, al perderse más información, el proceso es más veloz.

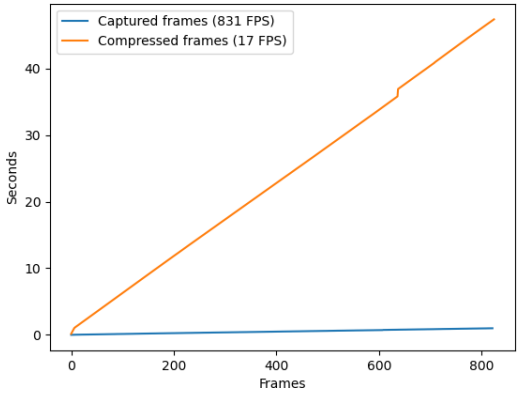


(a) Ratio de compresión 12 y *drProj* 8 **(b)** Ratio de compresión 24 y *drProj* 12

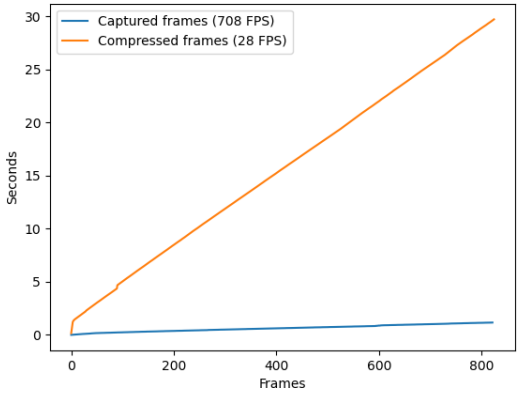
Figura 5.3: Compresión serie en Jetson Nano variando ratio de compresión

Por otro lado, se llevaron a cabo los mismos experimentos en la Jetson NX, cuyos resultados se recogen en la figura 5.4. Como se puede apreciar, tanto las velocidades de captura como las velocidades de compresión son entre dos y tres veces mayores.

De igual modo, se recoge en la tabla 5.1 los resultados de la compresión serie desglosados para cada uno de los dispositivos.



(a) Ratio de compresión 12 y *drProj* 8



(b) Ratio de compresión 24 y *drProj* 12

Figura 5.4: Compresión serie en Jetson NX variando ratio de compresión

Experimento Jetson Nano

Test	RC	drProj	Captura (FPS)	Compresión (FPS)
1	12	8	522	5
2	24	12	562	11

Experimento Jetson NX

Test	RC	drProj	Captura (FPS)	Compresión (FPS)
1	12	8	831	17
2	24	12	708	28

Tabla 5.2: Experimentos de compresión serie

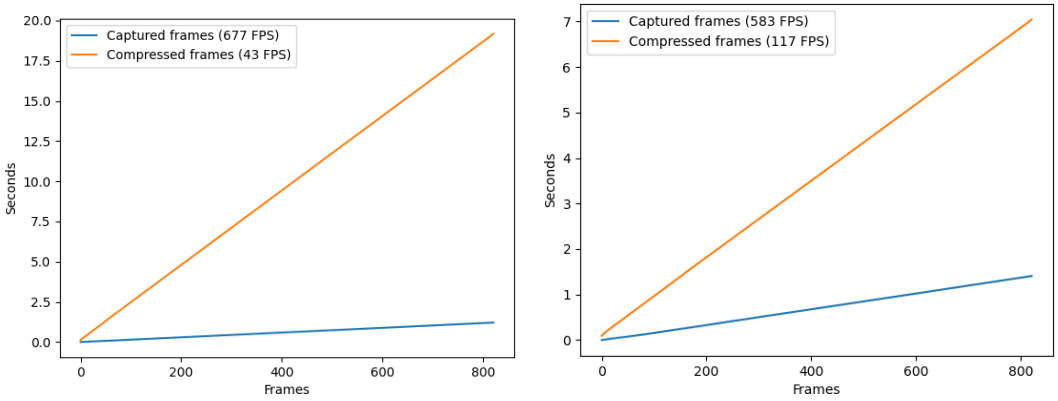
En conclusión, la implementación del proceso de compresión en serie no representa una solución óptima para el problema plantado en ninguno de los dos casos. Por tanto, se requiere de una implementación paralela que permita acelerar el proceso y se logren así los propósitos perseguidos en este proyecto. Estos resultados también demuestran la superioridad computacional del procesador integrado en la Jetson Xavier NX frente al que integra la Jetson Nano.

5.1.3 Implementación del sistema global de compresión

El siguiente paso a evaluar es la implementación paralela del algoritmo de compresión. Para ello, se ha procedido como en la sección anterior, estableciendo escenarios opuestos (más y menos restrictivo) para cada uno de los dispositivos de cómputo empleados.

En primer lugar, se han realizado los experimentos en la Jetson Nano cuyos resultados se muestran en la figura 5.5. Se puede apreciar una gran diferencia entre los *frame rate* de cada experimento, obteniendo un resultado de compresión de 43 FPS para el caso más desfavorable y 117 FPS en el más favorable. En la primera iteración no

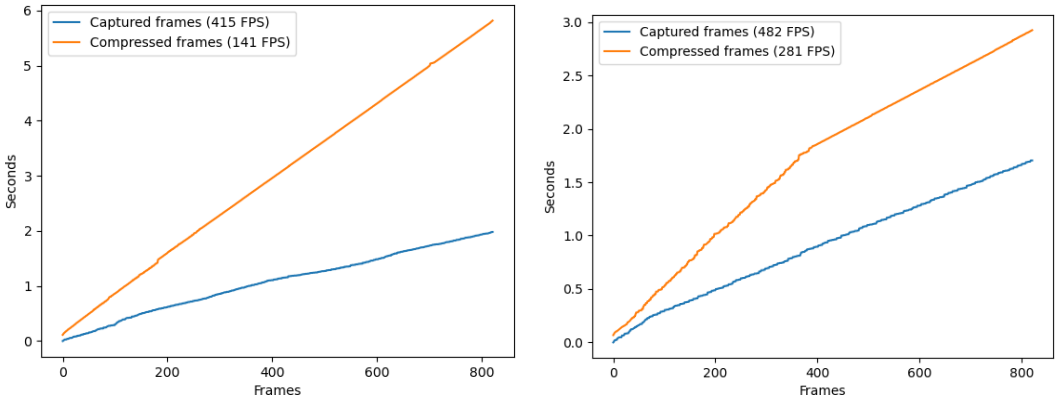
se cumplen los requisitos de tiempo real, ya que como se indicó con anterioridad, las imágenes son normalmente capturadas con un *frame rate* entre 100-150 FPS. No obstante, en el segundo experimento sí que se cumplen estas restricciones, convirtiéndose así en la primera solución apta para ser implementada en el sistema global.



(a) Ratio de compresión 12, drProj 8 **(b)** Ratio de compresión 24, drProj 12

Figura 5.5: Compresión paralela en Jetson Nano variando ratio de compresión y drProj

De igual forma, en la imagen 5.6, se muestran los mismos experimentos que en el caso anterior, donde se destaca que se cumplen las restricciones temporales holgadamente, obteniendo para el primer experimento 141 FPS y 281 para el segundo, de forma que se puede obtener una compresión en tiempo real según se ejecuta la misión de vuelo con cualquiera que sea la configuración de los parámetros de entrada.



(a) Ratio de compresión 12, drProj 8 **(b)** Ratio de compresión 24, drProj 12

Figura 5.6: Compresión paralela en Jetson NX variando ratio de compresión y drProj

En la tabla 5.3 se recogen los resultados detallados de cada experimento. Se extraen dos conclusiones principales de estos resultados. Por un lado, el PC Jetson Nano es capaz de alcanzar los requisitos de tiempo real en la configuración menos restrictiva del compresor, siempre y cuando la cámara se encuentre capturando a un *frame rate* que no sobrepase los 115 FPS. Por otro lado, la versión NX en su implementación

más restrictiva, es capaz de lograr tiempo real hasta alcanzar los 125 FPS, con lo cual, en el resto de configuraciones cumple holgadamente para las prestaciones de la cámara que se emplea.

Experimento Jetson Nano

Test	RC	drProj	Captura (FPS)	Compresión (FPS)
1	12	8	677	43
2			703	43
3			654	43
4	24	12	338	115
5			583	117
6			604	117

Experimento Jetson NX

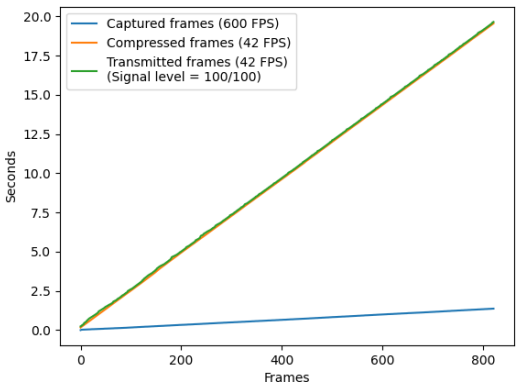
Test	RC	drProj	Captura (FPS)	Compresión (FPS)
1	12	8	251	125
2			415	141
3			445	143
4	24	12	482	281
5			486	292
6			502	288

Tabla 5.3: Experimentos de compresión serie

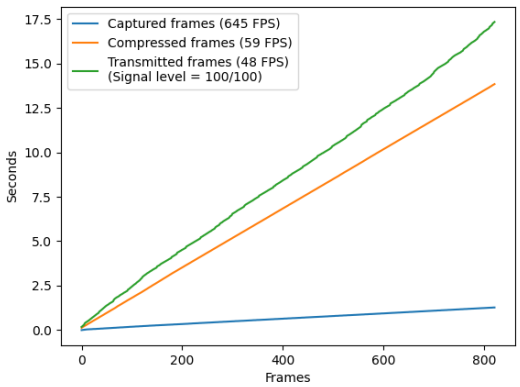
Una vez se ha logrado alcanzar el *frame rate* de compresión necesario, es ocasión de probar el sistema en su conjunto con el objetivo de verificar si se cumplen los objetivos de este proyecto. Así, no solamente se incorporará el compresor, sino también la etapa de transmisión en ambos dispositivos y con diferentes intensidades de conectividad WiFi.

Para proceder, se han diseñado cuatro experimentos para cada dispositivo, donde se han variado tanto los parámetros del ratio de compresión como *drProj*. Tal y como se muestra en la imagen 5.7, se aprecia que la única implementación que cumple con los requisitos temporales en términos de compresión es la recogida en la figura 5.7d, al tener un *frame rate* de compresión ligeramente superior a 100 FPS. Sin embargo, se constata que ninguna de las implementaciones logra obtener una tasa de envío que satisfaga los requisitos marcados. Este hecho puede deberse a la inclusión del adaptador WiFi que necesita el PC Jetson Nano para conectarse a la red, el cual se conecta a la placa con una interfaz USB2.0.

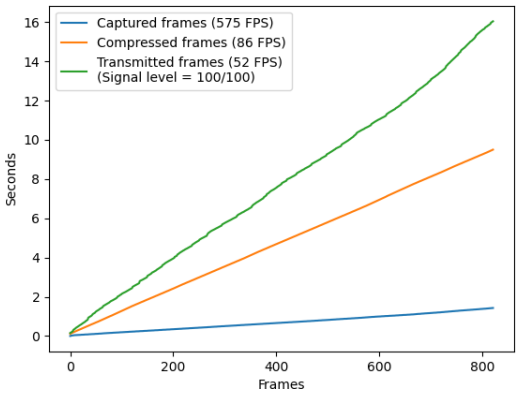
Por otro lado, los resultados de los experimentos realizados en la Jetson NX se muestran en la imagen 5.8, donde se puede apreciar cómo, a excepción del experimento recogido en la imagen 5.8a, todos cumplen holgadamente con los requisitos temporales de compresión, no así con los de transmisión, donde si bien es cierto que se obtienen unos resultados mejores que en el caso de Jetson Nano, estos siguen



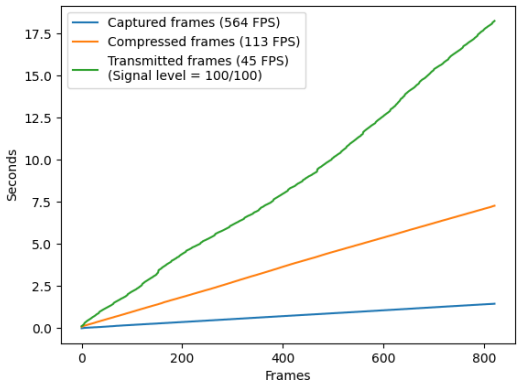
(a) Ratio de compresión 12, drProj 8



(b) Ratio de compresión 12, drProj 12



(c) Ratio de compresión 24, drProj 8

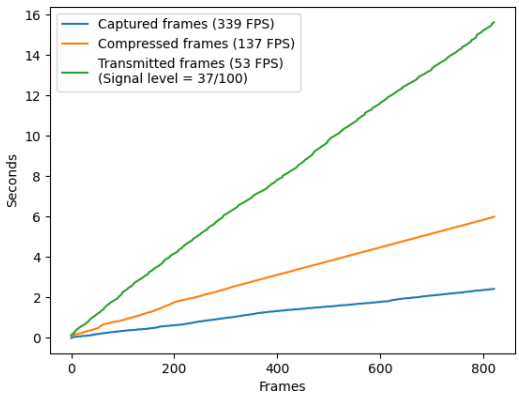


(d) Ratio de compresión 24, drProj 12

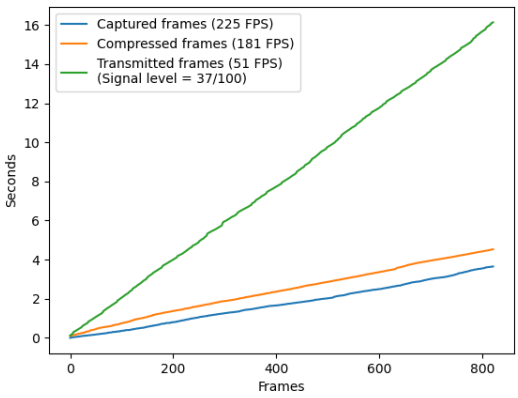
Figura 5.7: Compresión paralela y transmisión en la Jetson Nano variando parámetros de entrada

siendo insuficientes, ocasionando un cuello de botella en el sistema. Este aumento de velocidad puede deberse principalmente al hecho de que la antena WiFi se encuentra integrada en la PCB (Printed Circuit Board), empleando una interfaz de comunicación de mayor velocidad. No obstante, cabe destacar que en el caso de las simulaciones realizadas con la Jetson Xavier NX, la intensidad de la señal WiFi se encontraba únicamente en torno al 37 % de su valor máximo, con lo que un aumento en la intensidad de la misma podría repercutir en una mayor velocidad de transmisión.

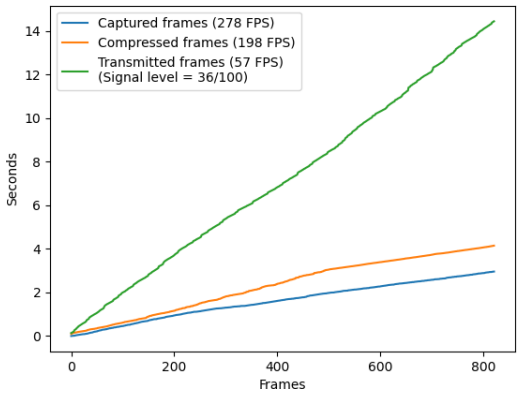
Para finalizar, en la tabla 5.4, se recogen los resultados detallados de cada experimento. Se han realizado tres iteraciones para cada configuración del compresor, obteniéndose unos mejores resultados de velocidad conforme mayor es el ratio de compresión y drProj. Como ya se ha mencionado anteriormente, esto se debe a que un aumento del valor de estos parámetros repercute en un menor número de iteraciones llevadas a cabo por el algoritmo de compresión. Para el análisis propio de las diferentes métricas de calidad del resultado de la compresión, se remite al lector al artículo [13], donde se ha realizado un estudio profundo de esta cuestión.



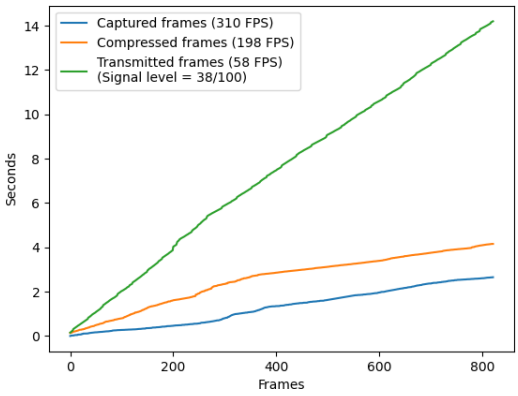
(a) Ratio de compresión 12, drProj 8



(b) Ratio de compresión 12, drProj 12



(c) Ratio de compresión 24, drProj 8



(d) Ratio de compresión 24, drProj 12

Figura 5.8: Compresión paralela y transmisión en Jetson NX variando parámetros de entrada

5.1.4 Líneas futuras

El capítulo de compresión de este proyecto deja varias líneas de investigación abiertas. En primer lugar, sería interesante evaluar la posibilidad de lograr una mayor aceleración de la ejecución del algoritmo de compresión en el dispositivo Jeton Nano, dado que si bien es del mismo tamaño que la versión NX, su consumo de potencia es menor, y este es un factor crítico a la hora de instalar estos dispositivos en plataformas de vuelo. Además, su precio es considerablemente menor, con lo cual se logra una solución global de menor coste.

Asimismo, la posibilidad de sustituir el almacenamiento de la Jetson NX por un SSD en lugar de una tarjeta SD permitiría un incremento de hasta cinco veces con respecto al actual según datos del fabricante, algo que sería de mucho interés para lograr una velocidad aún mayor en este dispositivo que ya de por sí ofrece un gran rendimiento.

Por otro lado, sería interesante valorar la posibilidad de optimizar el algoritmo de compresión, la implementación CUDA de la transformada o sus respectivos kernels,

Experimento Jetson Nano

Test	RC	drProj	Captura (FPS)	Compresión (FPS)	Transmisión (FPS)
1			600	42	42
2	12	8	631	42	42
3			588	41	41
4			645	59	48
5	12	12	635	60	49
6			643	59	45
7			611	88	49
8	24	8	637	87	52
9			575	86	52
10			573	112	47
11	24	12	564	113	45
12			521	114	46

Experimento Jetson NX

Test	RC	drProj	Captura (FPS)	Compresión (FPS)	Transmisión (FPS)
1			353	138	52
2	12	8	339	137	53
3			331	140	54
4			297	161	54
5	12	12	297	182	47
6			225	181	51
7			254	193	51
8	24	8	260	194	64
9			278	198	57
10			310	198	58
11	24	12	271	136	56
12			266	197	62

Tabla 5.4: Experimentos de compresión y envío

o incluso el diseño de la solución de compresión en su totalidad, de forma que se consiga un mayor *speedup*. No obstante, esta es una tarea compleja en la que ya se han realizado numerosas iteraciones tanto en este trabajo como en trabajos previos.

Sin embargo, el punto más importante a resolver es la cuestión de la transmisión de datos, la cual se aprecia que no suele sobrepasar el valor de 60 FPS con un factor de compresión de 24. Esto se puede lograr con un estudio en mayor detalle del sistema, donde se busque emplear dispositivos que incorporen el mismo estándar WiFi, a ser posible desde la versión N en adelante. Con un tratamiento preciso de este asunto así como la incorporación de antenas WiFi omnidireccionales de calidad en el sistema de vuelo, este problema podría ser mejorado en términos de velocidad, logrando así alcanzar los propósitos de tiempo real perseguidos en el transcurso de este proyecto. Otra alternativa a considerar podría ser cambiar la metodología de transmisión, esto es, actualmente se ejecuta el envío *frame* a *frame*, de modo que cada trama enviada no solo contiene el *frame* comprimido, sino todos los demás campos propios del protocolo empleado para la transmisión. Por tanto, una opción

interesante sería valorar la posibilidad de enviar paquetes de *frames*, de forma que cada vez que se agrupen un número determinado (por ejemplo, 5), sean enviados a tierra, reduciendo así el posible *overhead* existente en la transmisión. Por otro lado, también sería interesante analizar el diseño de un *socket* de transmisión (UDP o TCP) de modo que se pueda modificar específicamente para el objeto de la aplicación que se aborda y, por tanto, se reduzcan los *bit* de más que añade el protocolo SSH y que actualmente no se están aprovechando por no resultar de mayor interés.

5.2 Resultados de la implementación del sistema de comunicaciones

Para realizar una evaluación sobre la eficacia del sistema de apuntamiento diseñado en aras de abordar los objetivos del proyecto, se han realizado diversas pruebas que comprenden el análisis de las antenas que se emplearán en términos de potencia radiada, el montaje del sistema mecánico y la excitación de sus motores para verificar su correcto desplazamiento, así como la prueba final del sistema en su conjunto. Debido a la circunstancias particulares producidas por la pandemia del virus COVID-19, las pruebas de vuelo en exteriores se han visto limitadas, con lo cual no se ha podido verificar el sistema en su totalidad pero sí gran parte del mismo. En el desarrollo de esta sección también se indicarán los pormenores ocasionados y pruebas futuras que se deben todavía realizar.

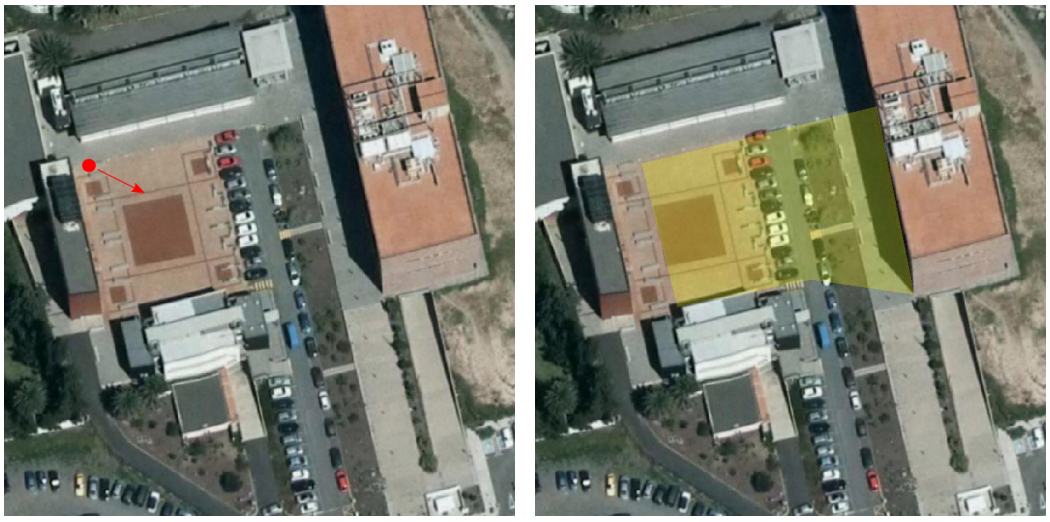
5.2.1 Análisis de diferentes antenas y su espectro de radiación

Para analizar qué antena se adecúa al sistema que se pretende implementar, se ha diseñado un experimento que permita estudiar en profundidad el patrón de radiación de los diferentes elementos radiantes que se disponen. Cabe mencionar que las antenas adquiridas son de muy bajo coste y han sido seleccionadas con el principal objeto de analizar el diagrama de radiación de cada una de ellas y escoger la que más se ajuste a los objetivos del proyecto. Si bien es cierto que su comportamiento puede no ser el esperado dada la poca calidad de las mismas, la teoría explicada anteriormente sobre la necesidad de una antena direccional sigue prevaleciendo y únicamente se indica la motivación de comprar unidades más robustas y fiables cuyo funcionamiento sea más adecuado.

El experimento diseñado es el siguiente: se ha configurado un *router* que, alimentado por una batería, se le intercambiarán diferentes antenas para ver su espectro de radiación. Para ello se emplea un ordenador conectado a esta red wifi local, que

haciendo uso de un GPS captará la intensidad de la señal wifi en cada punto geográfico con una velocidad de muestreo de 1 segundo. Todo ello es gestionado mediante un código Python que se encarga de ir tomando ambos valores simultáneamente. Obtenidos estos datos, se hace un postprocesado con el *software* Matlab que permite mostrar tanto la localización de los puntos medidos como su respectiva intensidad de señal en dBm sobre una imagen aérea de la zona. Con esto, se podrá visualizar el espectro de radiación de la antena.

Por otro lado, el postprocesado diseñado en Matlab se ha preparado para además poder evaluar, de forma cuantitativa y objetiva, las posibles mejoras que se obtendrían cuando la antena no se encuentre en una posición fija sino que se mueva siguiendo al objetivo. Para ello, se calcula una serie de métricas numéricas dentro de un espacio acotado idéntico para todos los test, llamado Región de Interés o ROI, definido previo a la realización de los experimentos. En este caso, las diferentes iteraciones se han realizado en el parking del edificio polivalente II del Parque Científico Tecnológico de Tafira, donde se ubican los laboratorios del IUMA. La figura 5.9 muestra una instantánea del lugar, donde se recoge en la figura 5.9a una instantánea aérea del parking donde se realizaron las pruebas. En ella, se ha destacado con un círculo en rojo la localización dónde se posicionó el *router* con la antena, y con una flecha la dirección aproximada en la que apuntaba esta. En la figura 5.9 se muestra el área de interés, ROI, aplicada a este terreno. Con lo cual, solo los puntos de intensidad WiFi tomados dentro de este área serán considerados para el análisis numérico individual de las prestaciones de cada antena.

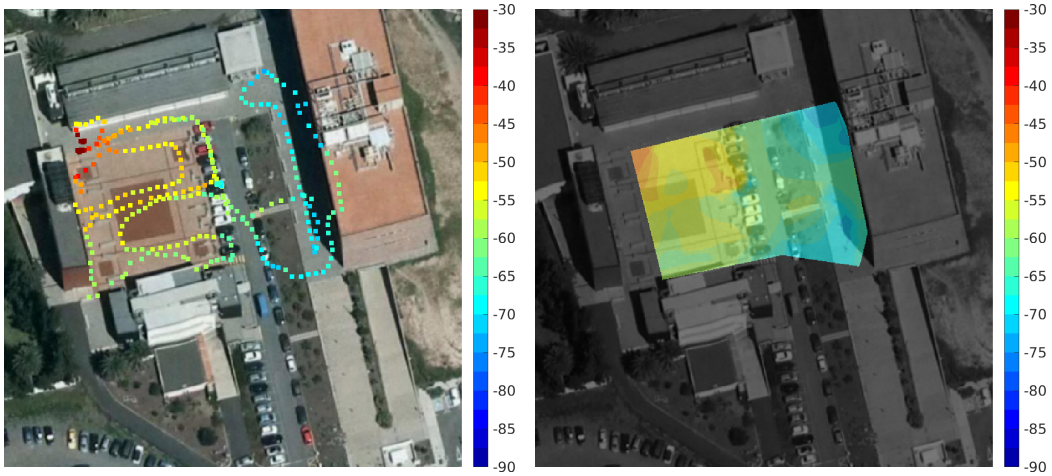


(a) Parking laboratorios IUMA **(b)** Configuración de ROI

Figura 5.9: Establecimiento del área de interés (ROI) en el terreno de estudio

Para comenzar, se muestra en la figura 5.10 la primera iteración del experimento donde se ha probado la configuración básica, es decir, el *router* con su antena omnidireccional tal cual es adquirido en el mercado. En la figura 5.10a se muestran

los diferentes puntos recorridos y en la figura 5.10b, la interpolación de estos puntos, limitando dicha interpolación únicamente al área de interés o ROI. De acuerdo con la Figura 5.10b, se constata el diagrama de radiación típico de una antena de estas características, donde la potencia se distribuye uniformemente en forma de círculos concéntricos.



(a) Puntos recorridos **(b)** Interpolación sobre el área de interés

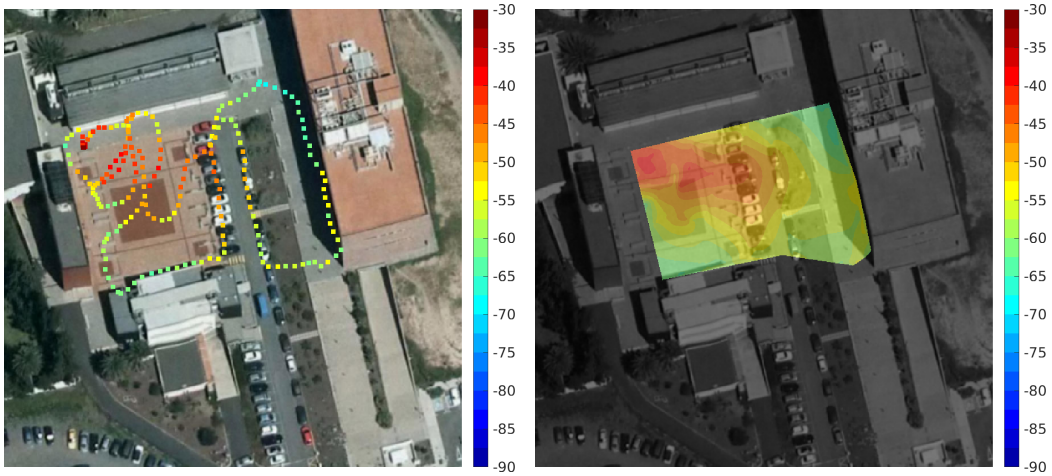
Figura 5.10: Análisis de antena omnidireccional sin amplificador

De igual forma, se han puesto a prueba otras antenas direccionales. Tras analizar varias y constatar la mala calidad de construcción y la poca ganancia conseguida en la mayoría de unidades, se empleó definitivamente la unidad mostrada en la imagen 5.11. Entre las características más interesantes se resalta una ganancia de 14 dBi, el funcionamiento a 2.4GHz, así como unas dimensiones de 17 cm de ancho por 17 de alto. Convirtiendo a esta unidad en una opción muy plausible a ser incorporada al sistema.

Figura 5.11: Antena direccional empleada

Los resultados arrojados tras instalar esta antena en el router se muestran en la figura 5.12, donde se aprecia en 5.12b un patrón de radiación mucho más direccional

que en el caso anterior. Además, se aprecia una mayor intensidad de señal en prácticamente todos los puntos del área de interés.



(a) Puntos recorridos **(b)** Interpolación sobre el área de interés

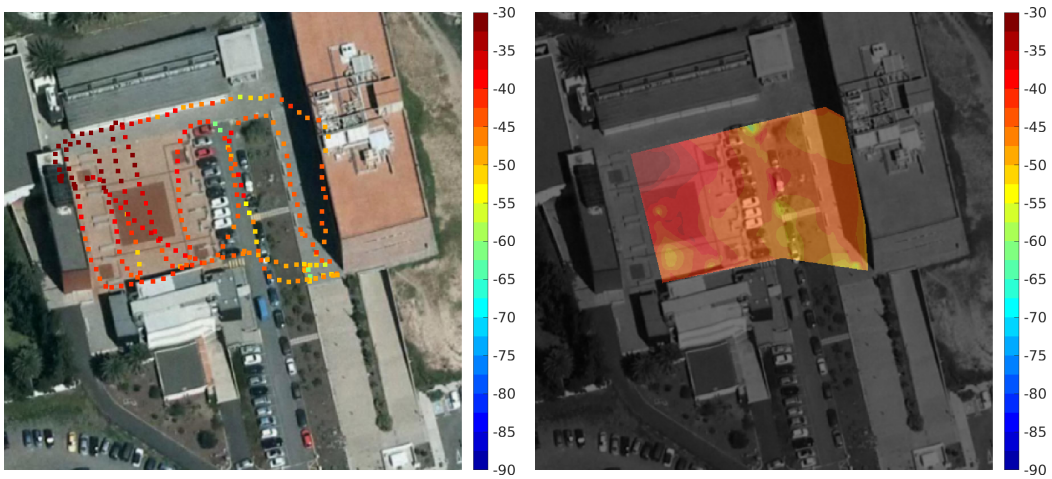
Figura 5.12: Análisis de antena omnidireccional con amplificador

Por otro lado, a la vista de los resultados obtenidos con ambas antenas, se evidencia la necesidad de incluir un amplificador que permita una mayor intensidad de señal en el área de estudio, con la finalidad de asegurar un *baudrate* alto y así asegurar los requisitos de tiempo real. Para ello, se ha empleado el amplificador mostrado en la imagen 5.13a, que es compatibles con las versiones B, G y N del estándar WiFi, funciona a 2.4 GHz, tiene una ganancia de transmisión de 13 dB y de recepción de 10 dB, así como un consumo aproximado de 4W. Además, en la figura 5.13b, se muestra una unidad de doble de potencia (8W), con una ganancia de transmisión de 17 dB y una de recepción de 13 dB aproximadamente, cuya implementación podría ser muy interesante para conseguir mayores coberturas, algo que se evaluará una vez se pueda acceder al campo de fútbol de Tafira.

(a) Amplificador 4W **(b)** Amplificador 8W

Figura 5.13: Amplificador al cual se conecta la antena WiFi

De esta forma, se han vuelto a realizar los experimentos anteriores. En primera instancia, se han obtenido los resultados mostrados en la figura 5.14 empleando la antena omnidireccional. En concreto, se puede apreciar en la imagen 5.14b una muy buena recepción de la señal y el diagrama de radiación isotrópico propio de la antena omnidireccional. Cabe destacar que la máxima potencia de recepción es de -30 dBm, con lo cual se puede apreciar una buena cobertura en todo el área de estudio, solo viéndose disminuida esta en lugares donde no existe visión directa, algo que no sucederá cuando se implemente este sistema en conexión con el dron.

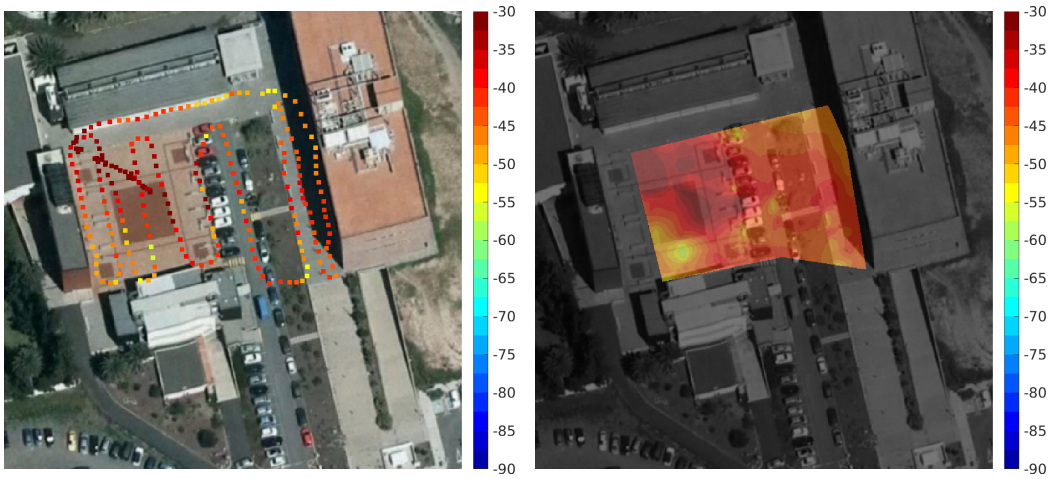


(a) Puntos recorridos **(b)** Interpolación sobre el área de interés

Figura 5.14: Análisis de antena omnidireccional con amplificador

Por otro lado, se muestra en la figura 5.15 el empleo de la antena direccional junto con el amplificador, donde se puede apreciar ligeramente una mejora en la potencia de la señal en la figura 5.15b con respecto a la antena omnidireccional, siendo el patrón de radiación más directivo. Empleando esta última configuración, se ha obtenido -40 dBm de potencia de señal recibida a 40 metros desde el router con visión directa. Un resultado muy aceptable pero poco concluyente ya que no se han podido realizar los test en el terreno habitual de pruebas, algo que se realizará en posteriores iteraciones. Sin embargo, para obtener una conclusión objetiva de estos resultados, es preciso analizar las métricas calculadas.

Estas se recogen en la tabla 5.14. La potencia media de la señal dentro del ROI es la media de los distintos puntos de medida de la intensidad de la señal. Se puede apreciar cómo la configuración que ofrece mejores resultados es en ambos casos la antena direccional, consiguiendo la mejor marca en la implementación con amplificador. Por otro lado, el porcentaje de área cubierta es un indicador de la superficie del ROI de la que se han obtenido valores, ya sea directamente o por interpolación. Finalmente, la última métrica indica la desviación estándar de las diferentes medidas tomadas. Estas métricas adquirirán mayor relevancia cuando sea posible acceder a un área de test de mayores dimensiones donde pueda compararse



(a) Puntos recorridos **(b)** Interpolación sobre el área de interés

Figura 5.15: Análisis de antena direccional con amplificador

la eficacia de la antena direccional apuntando en una dirección fija frente a la eficacia de la misma antena cuando se mueva siguiendo al objetivo. Se espera en este caso tener tanto un aumento considerable en los valores medios de intensidad medidos (dBm) así como una reducción en la desviación estándar de los valores. Estas métricas también serán de interés si se adquiere una antena de mejor calidad y con un patrón de radiación más direccional.

Antena	Potencia media (dBm)	Área cubierta (%)	Desviación estándar
Omnidireccional	-61.99	98.88	0.56
Direccional	-54.3	97.59	0.50
Omnidireccional con amplificador	-43.97	99.09	0.40
Direccional con amplificador	-41.99	96.39	0.39

Tabla 5.5: Análisis de la métrica de potencia media para las diferentes antenas.

Finalmente, se recoge en la imagen 5.16 los dispositivos que se han empleado para realizar las pruebas. En la parte superior izquierda se muestra el ordenador al que se conecta el módulo GPS. Debajo de este, en una caja *self-made* se encuentra el router y el amplificador sobre este. También se puede observar la batería empleada justo debajo del *router*. En la parte derecha, se encuentran las antenas probadas, donde, como se ha indicado, se tomaron como válidas la direccional cuadrada y la omnidireccional por ser las que mejores resultados ofrecían.

5.2.2 Montaje físico de la estructura de la antena

Con respecto al montaje físico del sistema de antenas empleado, se muestra el resultado en la figura 5.17. Tras haber realizado la impresión 3D y el montaje de

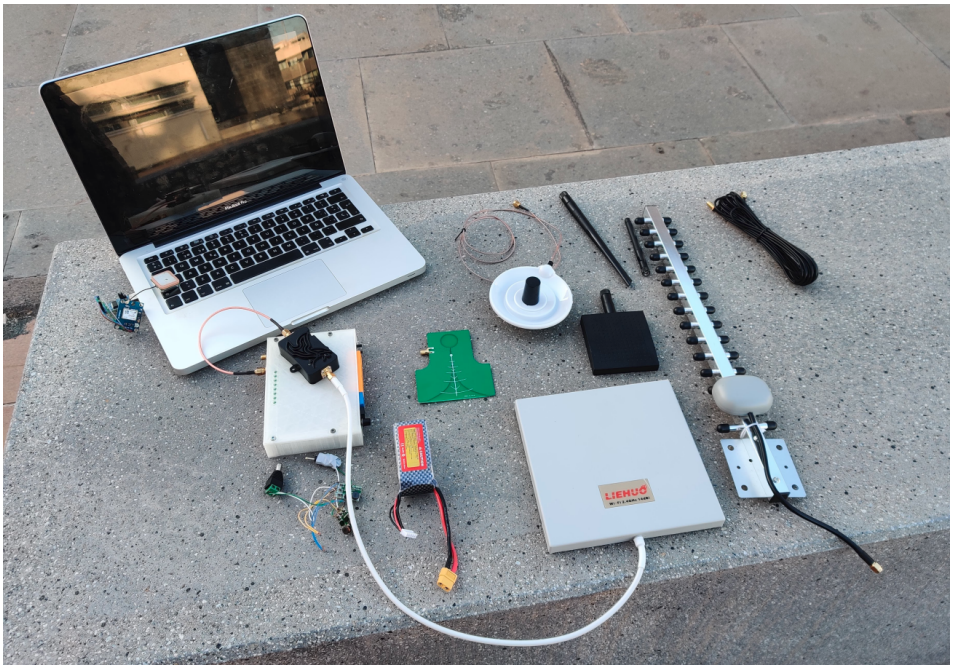


Figura 5.16: Setup del experimento

las piezas que componen la estructura, se han tenido que realizar una serie de modificaciones que permitan ensamblar los distintos componentes en esta. Así, dado que se implementarán diferentes antenas a modo de prueba, el diseño propio de la adaptación de cada una queda abierto a las necesidades particulares de cada antena.

Básicamente se han diseñado dos piezas nuevas, en primer lugar, el engranaje que se conecta al eje del motor mostrado en la figura 5.18a, donde se ha diseñado la cavidad para insertar un tornillo que permita que el engranaje se ajuste correctamente al eje y permanezca completamente inmóvil a la hora de mover el motor. Por otro lado, se ha diseñado la pieza donde se ubica la IMU, cuya posición se escogió la más alejada de los motores con la finalidad de minimizar la distorsión magnética que estos producen. Además, se ha ubicado en el eje horizontal, de forma que esta distorsión sea lo más continua posible y poder realizar una calibración correcta y que pueda modelarse como distorsión de tipo *soft iron*.

5.2.3 Interconexión del sistema en su conjunto

Para concluir, se procede a mostrar el resultado físico del sistema completo. En primer lugar, se muestra en la imagen 5.19 el módulo de comunicaciones que se incorpora en la estación de tierra. Este está compuesto por un bluetooth maestro que se conecta directamente a un conversor TTL a USB, con la finalidad de poder

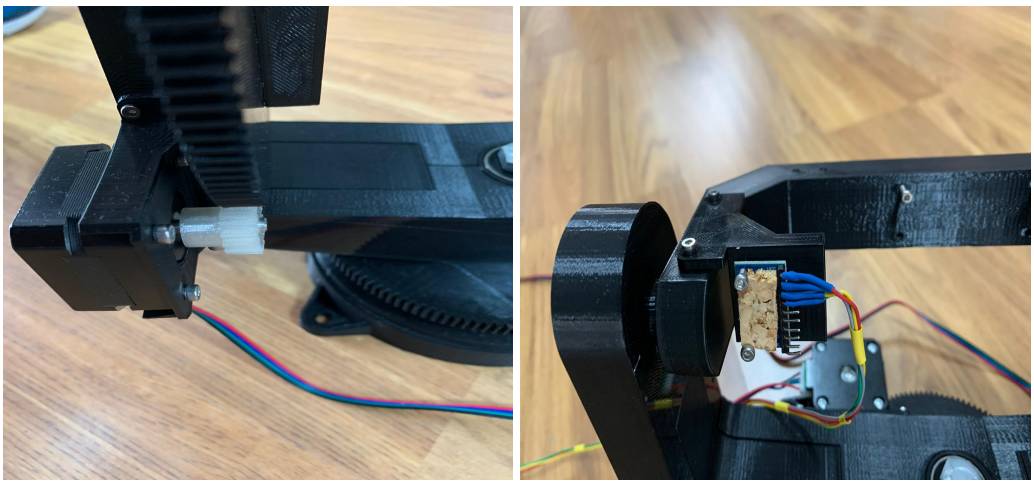
Figura 5.17: Resultado general del montaje de la estructura de la antena

gestionarlo desde el portátil o tablet como si de un puerto serie se tratara. No obstante, en el caso de trabajar con una *tablet*, este dispositivo no es absolutamente necesario ya que la mayor parte de estas integran su propio módulo bluetooth.

Por otro lado, se recoge en la imagen 5.20 el sistema referido al control del movimiento de la antena, a excepción de la propia estructura. En esta se encuentra la interconexión de sus componentes: batería para la alimentación del conjunto, convertidores de tipo buck, interruptor, GPS, IMU, etc.

En la imagen 5.21, se detallan de forma individual todos los componentes que conforman el sistema en su totalidad.

Finalmente, cabe resaltar que aunque la situación promovida por la pandemia mundial sufrida durante los últimos meses ha afectado al desarrollo de este proyecto, se ha verificado de forma satisfactoria todos los pasos que a continuación se detallan: solicitud de coordenadas GPS, solicitud de coordenadas de orientación, calibración del sistema desde la estación de tierra, configuración de parámetros del motor, orden de interrupción desde la estación de tierra sobre los comandos ejecutados en el sistema de antenas y realización de la rutina de *Homming*. Únicamente no se ha podido validar el paso de apuntamiento al dron debido a la incapacidad de realizar estas pruebas de vuelo que normalmente se realizan en el campo de fútbol de Tafira.



(a) Eje del motor **(b)** Soporte para la IMU

Figura 5.18: Modificaciones realizadas para adaptar la estructura

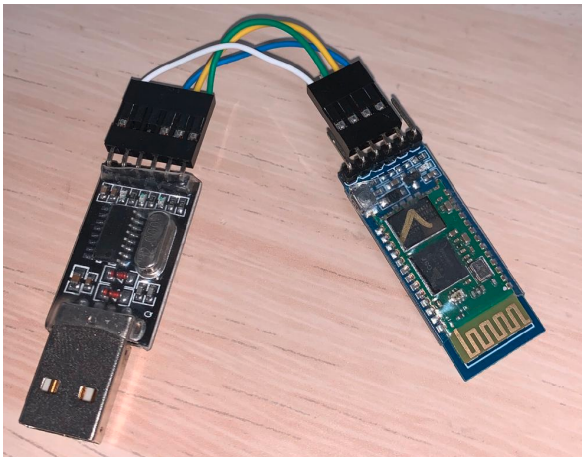


Figura 5.19: Módulo de comunicaciones de la estación de tierra

5.2.4 Líneas futuras

Tras la realización de esta parte del proyecto, surgen gran cantidad de líneas futuras en las cuales ahondar para mejorar el sistema en términos de prestaciones, calidad, precisión y consumo de potencia. En primer lugar y con respecto a la IMU, la posibilidad de emplear algoritmos de calibración más elaborados que doten de mayor precisión a las medidas es un punto muy interesante de estudio. Asimismo, la adquisición de antenas que sean más precisas y robustas también contribuiría en gran medida a mejorar el comportamiento del sistema. Por último, la posibilidad de emplear el sensor giróscopo que lleva incorporado la IUMA ayudaría a establecer un sistema retroalimentado donde motores e IMUs se coordinaran para establecer un movimiento más suave del sistema.

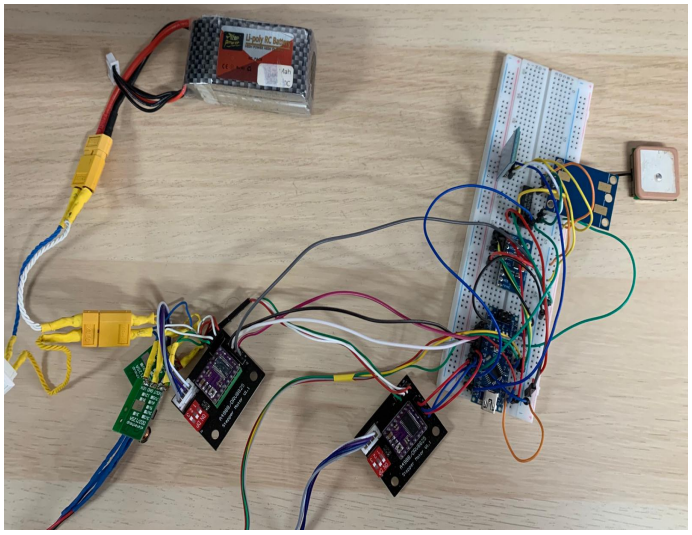


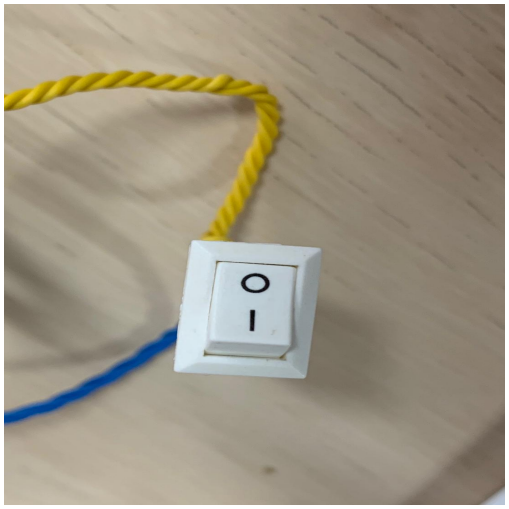
Figura 5.20: Sistema completo del sistema de antenas

Por otro lado, en cuanto a los motores, si bien su funcionamiento cumple lo esperado, convendría analizar la posibilidad de emplear unos de mayor tamaño que proporcionen un torque mayor en el caso de que hubiera que instalar otro tipo de antena de mayor peso. En este punto, el empleo de antenas de mayor calidad es un punto muy importante que podría mejorar en gran cantidad la recepción de la señal, en tanto que las unidades adquiridas son de una calidad baja, distando mucho sus resultados de las especificaciones técnicas mostradas en sus datasheet.

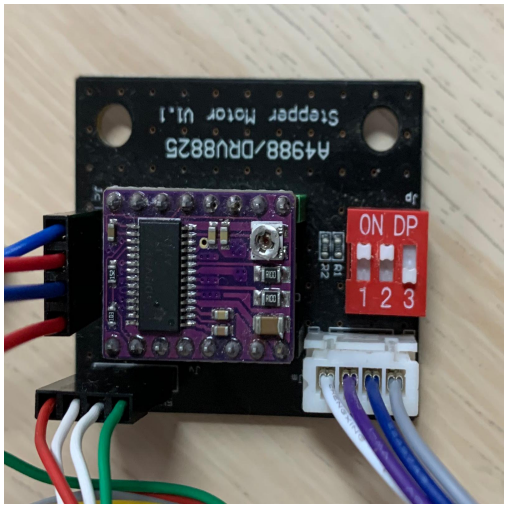
Además, una vez se tenga acceso al campo de pruebas, sería de mucho interés realizar los mismos experimentos con el dron en vuelo, de forma que se puedan identificar puntos débiles del sistema de apuntamiento en distintas situaciones. Así, en la figura 5.22, se muestra la región de interés diseñada para el campo de fútbol de Tafira donde se realizarán las diferentes pruebas implementando el sistema diseñado con el aparato en vuelo. De esta forma, se podrán realizar diferentes barridos donde, con el sistema de antenas móvil, se analice si la potencia de señal recibida en cada punto es la esperada.

Para finalizar, una vez se escoja la antena que más se adapte a las necesidades del problema y en vista a obtener una solución optimizada, se podría realizar un diseño más a medida de la estructura móvil que arrastra a la antena que se emplee, de forma que se reduzca al máximo posible el tamaño del sistema. También sería conveniente condensar en una caja todos los componentes del sistema, así como soldarlos, con la finalidad de dotar al sistema de una mayor fiabilidad y atractivo visual.

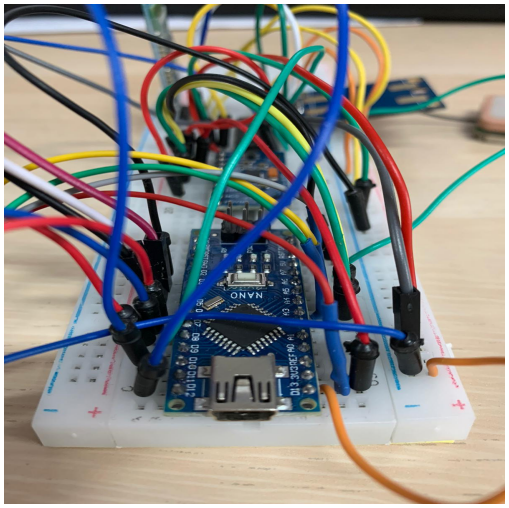
(a) Batería



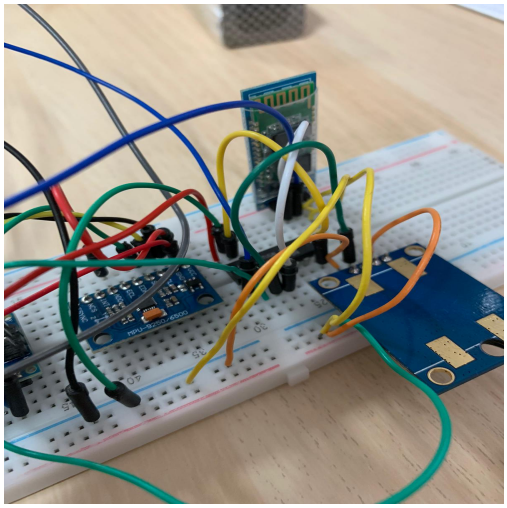
(b) Interruptor



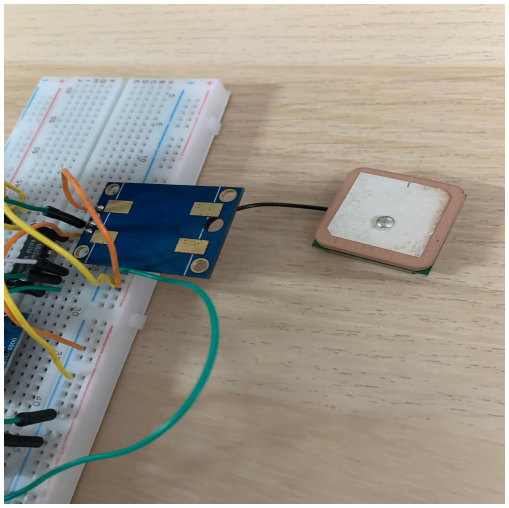
(c) Driver motores



(d) Arduino NANO



(e) IMU y Bluetooth



(f) GPS

Figura 5.21: Componentes del sistema diseñado

Figura 5.22: Terreno de pruebas de vuelo

6 Conclusiones

Para concluir este Trabajo de Fin de Máster, se exponen las siguientes conclusiones según los resultados alcanzados:

- Se ha implementado en distintos sistemas empotrados la versión paralelizada del algoritmo HyperLCA adaptado al actual funcionamiento de la plataforma de vuelo, logrando unos resultados de compresión en tiempo real muy satisfactorios y comprobando su funcionamiento con diferentes configuraciones de PC OnBoard.

- Se ha diseñado un entorno de red local donde se realiza la transmisión de los *frames* comprimidos a tierra, obteniendo unos resultados satisfactorios pero que necesitan ciertas optimizaciones para alcanzar los requisitos esperados.

- Se han analizado diferentes configuraciones de la red, haciendo hincapié tanto en las diferentes versiones del protocolo 802.11 así como realizando un análisis exhaustivo de los diferentes elementos radiantes que pueden incluirse en el router que centraliza las conexiones.

- Se ha diseñado un sistema de antenas móvil, que, apoyándose en una estructura diseñada en 3D, mediante elementos como Unidades de Medida Inercial, motores, GPS, etc. es capaz de realizar los movimientos especificados, obteniendo un control absoluto de la misma y estando preparada para las futuras iteraciones de apuntamiento al dron con el objeto de maximizar la cobertura WiFi.

- Se realizó el diseño de una comunicación Bluetooth entre la estación de tierra y el sistema de antenas, de forma que todos los cálculos de apuntamiento, movimiento de los motores, coordenadas del UAV, entre otros, estén centralizados en la estación de tierra, donde esta se encargue de ejecutar toda la carga computacional y los elementos externos (antena y sistema de vuelo) sean esclavos a la espera de recibir órdenes.

Por ello, tras la consecución de estos objetivos, surgen las siguientes líneas futuras generales de investigación con el propósito de mejorar el sistema en términos de eficacia:

- Análisis profundo de la configuración de red que permita una aceleración de la tasa de envío de los *frames* comprimidos hacia tierra.

- Optimización del grado de paralelismo y reducción de las transacciones de memoria del compresor explotando las posibilidades que ofrece cada PC empotrado, de forma que se puedan conseguir unos resultados satisfactorios y adaptados a los requisitos de funcionamiento de la cámara hiperespectral, de forma que se consiga una mayor celeridad en los procesos que intervienen.

- Implementación de elementos radiantes de mayor calidad, de forma que se consiga una estabilidad superior y una mejor ganancia de la señal logrando la entrega de información en tiempo y forma.

- Adaptar la implementación a las diversas estaciones de tierra que potencialmente se puedan incorporar, de forma que se adecúe tanto a posibles programas en un portátil como aplicaciones para el caso de una tablet.

- Mejora en el movimiento del sistema de antenas, donde se empleen componentes de mayor calidad y fiabilidad, ejecutando cuantas pruebas fueran necesarias para verificar el funcionamiento del sistema bajo cualquier casuística.

Bibliografía

[1]*A3 - MÁXIMA FIABILIDAD. INFINITAS POSIBILIDADES*. https://www.dji.com/es/a3. (Accessed on 07/12/2020) (vid. pág. 13).

[2]Telmo Adão, Jonáš Hruška, Luís Pádua y col. „Hyperspectral Imaging: A Review on UAV-Based Sensors, Data Processing and Applications for Agriculture and Forestry“. En: *Remote Sensing* 2017 (oct. de 2017), pág. 1110 (vid. pág. 5).

[4]Rocio Calderón Madrid, Juan Navas Cortés y Pablo Zarco-Tejada. „Early Detection and Quantification of Verticillium Wilt in Olive Using Hyperspectral and Thermal Imagery over Large Areas“. En: *Remote Sensing* 7 (mayo de 2015), págs. 5584-5610 (vid. pág. 2).

[5]*Camera Gimbal by fhuable - Thingiverse*. https://www.thingiverse.com/thing:3375167. (Accessed on 07/06/2020) (vid. pág. 38).

[6]P. Chen y C. Lee. „UAVNet: An Efficient Obstacel Detection Model for UAV with Autonomous Flight“. En: *2018 International Conference on Intelligent Autonomous Systems (ICoIAS)*. 2018, págs. 217-220 (vid. pág. 5).

[8]*Coordinate Systems for Navigation - MATLAB & Simulink*. https://www.mathworks.com/help/aerotbx/ug/coordinate-systems-for-navigation.html. (Accessed on 07/12/2020) (vid. pág. 33).

[9]*CUDA Runtime API :: CUDA Toolkit Documentation*. https://docs.nvidia.com/cuda/cuda-runtime-api/group__CUDART__EXECUTION.html. (Accessed on 07/14/2020).

[10]*D017532629.pdf*. http://www.iosrjournals.org/iosr-jce/papers/Vol17-issue5/Version-3/D017532629.pdf. (Accessed on 07/12/2020) (vid. pág. 30).

[11]M. Díaz, R. Guerra, P. Horstrand y col. „Real-Time Hyperspectral Image Compression Onto Embedded GPUs“. En: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 12.8 (2019), págs. 2792-2809.

[12]M. Díaz, R. Guerra, P. Horstrand y col. „Real-Time Hyperspectral Image Compression Onto Embedded GPUs“. En: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 12.8 (2019), págs. 2792-2809 (vid. págs. 3, 21-23).

[13]María Díaz, Raúl Guerra, Pablo Horstrand y col. „Real-Time Hyperspectral Image Compression Onto Embedded GPUs“. En: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (2019), págs. 1-18 (vid. pág. 78).

[14]*DJI Matrice 600 Pro - DJI*. https://www.dji.com/es/matrice600-pro. (Accessed on 07/12/2020) (vid. pág. 12).

[15]*DJI Ronin-M - Estabilizador ligero y portátil en tres ejes para cámaras - DJI*. https://www.dji.com/es/ronin-m. (Accessed on 07/12/2020) (vid. pág. 13).

[16]G. Florimbi, H. Fabelo, E. Torti y col. „Towards Real-Time Computing of Intraoperative Hyperspectral Imaging for Brain Cancer Detection Using Multi-GPU Platforms“. En: *IEEE Access* 8 (2020), págs. 8485-8501 (vid. pág. 2).

[17]Jorge Gago, Cyril Douthe, Rafael Coopman y col. „UAVs challenge to assess water stress for sustainable agriculture“. En: *Agricultural Water Management* 153 (ene. de 2015) (vid. pág. 2).

[18]Raul Guerra Hernández, Yubal Barrios, Maria Diaz y col. „A New Algorithm for the On-Board Compression of Hyperspectral Images“. En: *Remote Sensing* 10 (mar. de 2018), pág. 428 (vid. pág. 3).

[19]J. L. E. Honrado, D. B. Solpico, C. M. Favila y col. „UAV imaging with low-cost multispectral imaging system for precision agriculture applications“. En: *2017 IEEE Global Humanitarian Technology Conference (GHTC)*. 2017, págs. 1-7 (vid. pág. 2).

[20]P. Horstrand, R. Guerra, A. Rodríguez y col. „A UAV Platform Based on a Hyperspectral Sensor for Image Capturing and On-Board Processing“. En: *IEEE Access* 7 (2019), págs. 66919-66938 (vid. pág. 2).

[21]*inotify(7) - Linux manual page*. https://man7.org/linux/man-pages/man7/inotify.7.html. (Accessed on 07/13/2020) (vid. pág. 24).

[24]H. Kayan, R. Eslampanah, F. Yeganli y M. Askar. „Heat leakage detection and surveiallance using aerial thermography drone“. En: *2018 26th Signal Processing and Communications Applications Conference (SIU)*. 2018, págs. 1-4 (vid. pág. 5).

[25]L.R. Newcome. *Unmanned Aviation: A Brief History of Unmanned Aerial Vehicles*. Pen & Sword Books Limited, 2005 (vid. pág. 4).

[30]Youcef Saad. *Numerical methods for large eigenvalue problems*. Manchester University Press, 1992 (vid. pág. 20).

[31]L. Santos, R. Vitulli, J. F. López y R. Sarmiento. „GPU implementation of a lossy compression algorithm for hyperspectral images“. En: *2012 4th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*. 2012, págs. 1-4 (vid. pág. 3).

[36]C. Wang, J. Wang, X. Zhang y X. Zhang. „Autonomous navigation of UAV in large-scale unknown complex environment with deep reinforcement learning“. En: *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. 2017, págs. 858-862 (vid. pág. 5).

[37]J. Wang, W. Guo, T. Pan y col. „Bottle Detection in the Wild Using Low-Altitude Unmanned Aerial Vehicles“. En: *2018 21st International Conference on Information Fusion (FUSION)*. 2018, págs. 439-444 (vid. pág. 5).

[38]*Welcome to Paramiko! — Paramiko documentation*. http://www.paramiko.org/. (Accessed on 07/13/2020) (vid. pág. 25).

Recursos Web

[3]Arduino. *Arduino Nano | Arduino Official Store*. (Accessed on 05/11/2020) (vid. pág. 39).

[7]*Compensating for Tilt, Hard-Iron, and Soft-Iron Effects | FierceElectronics*. (Accessed on 07/01/2020) (vid. pág. 50).

[22]InvenSense. *PS-MPU-9250A-01-v1.1.pdf*. (Accessed on 05/09/2020) (vid. pág. 44).

[23]*Jetson Xavier NX for Embedded & Edge Systems | NVIDIA*. (Accessed on 07/12/2020) (vid. pág. 16).

[26]*NVIDIA Jetson Nano Developer Kit | NVIDIA Developer*. (Accessed on 07/12/2020) (vid. pág. 16).

[27]*Pololu - DRV8825 Stepper Motor Driver Carrier, High Current*. (Accessed on 06/21/2020) (vid. págs. 39, 43).

[28]*Quadcopter AR Drone 2.0 Elite Edition | Parrot Official*. (Accessed on 07/12/2020) (vid. pág. 5).

[29]CH Robotics. *Understanding Euler Angles*. (Accessed on 05/11/2020) (vid. pág. 48).

[32]*Specim FX10 – Specim*. (Accessed on 07/12/2020) (vid. pág. 15).

[33]*Specim FX17 – Specim*. (Accessed on 07/12/2020) (vid. pág. 15).

[34]*TL-WN722N | 150Mbps High Gain Wireless USB Adapter | TP-Link*. (Accessed on 07/12/2020) (vid. pág. 71).

[35]*TL-WR841N | Router inalámbrico N a 300 Mbps | TP-Link Iberia*. (Accessed on 07/12/2020) (vid. pág. 71).

Índice de figuras

2.1 Espectro electromagnético 7
2.2 Tipos de sensores espectrales 8
2.3 Cubo hiperespectral 8
2.4 Funcionamiento de cámara espectral de tipo whiskbroom 10
2.5 Funcionamiento de cámara espectral de tipo pushbroom 10
2.6 Plataforma de vuelo con cámara hiperespectral. 12
2.7 UAV sin carga útil. 12
2.8 Gimbal Ronin MX DJI 14
2.9 Cámaras hiperespectrales de la compañía Specim 15

3.1 Opciones de PC OnBoard 17
3.2 Gráfico de transmisión por WiFi entre PC OnBoard y estación de tierra 18
3.3 Etapas del compresor HyperLCA 20
3.4 Evaluación de compresión serie en Jetson Nano 22
3.5 Primera aproximación de compresión paralela 23
3.6 Diseño de la arquitectura de compresión y envío de imágenes hiperespectrales 23

4.1 Tipos de antena y sus diagramas de radiación 31
4.2 Esquema de comunicaciones de la plataforma 33
4.3 Movimientos de la antena para apuntar al dron en vuelo 34
4.4 Movimientos de la antena para apuntar al dron en vuelo 34
4.5 Sistema de referencia local en coordenadas Pitch, Roll y Yaw 35
4.6 Diagrama de funcionamiento del sistema de antenas 37
4.7 Estructura sobre la que irá montada la antena 38
4.8 Diagrama de los diferentes módulos del circuito 40
4.9 Baterías de tipo LiPo 42
4.10 Motor paso a paso NEMA 17. 43
4.11 Driver empleado para el control de los motores 44
4.12 MPU9250 junto con un esquemático de sus ejes 45
4.13 Cálculos para la matriz de Rotación 46
4.14 Diagrama de inclinación magnética en un mapa geográfico 49
4.15 Mapa de registros correspondiente al magnetómetro 50
4.16 Conexión de Arduino Nano con MPU9250 53
4.17 Pseudo-interfaz de usuario en Arduino 54

4.18 Lectura de datos crudos 55
4.19 Comparación de datos crudos y calibrados para el magnetómetro . . . 57
4.20 Interfaz de Arduino donde se realiza una calibración y medición de datos 60
4.21 Interfaz de Arduino donde se realiza una calibración y medición de datos 61
4.22 Dispositivos bluetooth empleados para la comunicación 62
4.23 Conversor TTL a USB 63
4.24 GPS U-Blox NEO-6M 64
4.25 Conexión de los módulos de bluetooth y GPS 67
4.26 Interconexión de los driver y sus motores 68
4.27 Conexión de las IMU 68
4.28 Interfaz de comunicación de la estación de tierra 69

5.1 Dispositivos de red empleados para ejecutar las pruebas 72
5.2 Análisis de envío 73
5.3 Compresión serie en Jetson Nano variando ratio de compresión 74
5.4 Compresión serie en Jetson NX variando ratio de compresión 75
5.5 Compresión paralela en Jetson Nano variando ratio de compresión y drProj 76
5.6 Compresión paralela en Jetson NX variando ratio de compresión y drProj 76
5.7 Compresión paralela y transmisión en la Jetson Nano variando parámetros de entrada 78
5.8 Compresión paralela y transmisión en Jetson NX variando parámetros de entrada 79
5.9 Establecimiento del área de interés (ROI) en el terreno de estudio . . . 82
5.10 Análisis de antena omnidireccional sin amplificador 83
5.11 Antena direccional empleada 83
5.12 Análisis de antena omnidireccional con amplificador 84
5.13 Amplificador al cual se conecta la antena WiFi 84
5.14 Análisis de antena omnidireccional con amplificador 85
5.15 Análisis de antena direccional con amplificador 86
5.16 Setup del experimento 87
5.17 Resultado general del montaje de la estructura de la antena 88
5.18 Modificaciones realizadas para adaptar la estructura 89
5.19 Módulo de comunicaciones de la estación de tierra 89
5.20 Sistema completo del sistema de antenas 90
5.21 Componentes del sistema diseñado 91
5.22 Terreno de pruebas de vuelo 92

Índice de cuadros

2.1 Características técnicas de las dos cámaras que se emplean en el UAV . 15

3.1 Características técnicas del PC OnBoard Jetson Nano 17
3.2 Características técnicas del PC OnBoard Jetson Xavier NX 17

4.1 Características técnicas de los diferentes estándares WiFi 30
4.2 Consumos de potencia de los elementos de cada circuito. 41

5.1 Experimentos de captura y transmisión de *frames* 73
5.2 Experimentos de compresión serie 75
5.3 Experimentos de compresión serie 77
5.4 Experimentos de compresión y envío 80
5.5 Análisis de la métrica de potencia media para las diferentes antenas. . 86