Proyecto final de grado de ingeniería eléctrica

TERMODRON III

Dron autónomo de reconocimiento por termografía

Autores:
Kevin Sosa
Martín Amado
Guillermo Fischer

Tutor:
Rafael Canetti

Instituto de Ingeniería Eléctrica
Facultad de Ingeniería
Universidad de la República

lunes 5 abril, 2021

Agradecimientos

A nuestro tutor, Rafael Canetti, por guiarnos y por los conocimientos compartidos.

A María Emilia Correa, Lucía Torrico y Sofía Montenegro por brindarnos apoyo en todo momento.

Al grupo Termodron II por los consejos, el soporte, y la confianza de dejarnos hacer uso de su proyecto.

A nuestros familiares y amigos que estuvieron presentes brindando apoyo, aliento, y siempre a disposición en el curso del proyecto.

Resumen

El sistema Termodron consiste en un cuadricóptero de vuelo autónomo, con capacidad de detección de focos de calor mediante termografía, evasión de obstáculos, aterrizaje de precisión, recarga inalámbrica, reporte de datos de vuelo en tiempo real, y selección de misiones programadas por el usuario mediante comunicación inalámbrica.

Se realizan mejoras al sistema previo, Termodron II, basándose tanto en las recomendaciones de dicho equipo como también en ideas propias. Estas mejoras incluyen nuevo hardware como ser el controlador de vuelo, GPS, motores y cámara térmica, y software, donde se reescribe la mayoría del código desde cero, organizándolo en librerías separadas por cada funcionalidad o módulo de hardware correspondiente. Además de las mejoras realizadas también se incorporan nuevas funcionalidades al sistema, desde el uso de cámara estereoscópica para evasión de obstáculos, hasta un algoritmo de aterrizaje de precisión mediante reconocimiento de patrones y procesamiento de imágenes en tiempo real, sobre el dron. También se realiza un nuevo diseño de la base acorde a los nuevos requerimientos.

Para comenzar una misión, el usuario configura los diferentes parámetros de la misma mediante una interfaz gráfica, la cual envía la misión a la base y ésta al dron (el cual está aterrizado en la base, monitoreando su estado y las peticiones de misiones entrantes). Algunos de los parámetros a asignar son, el tipo de misión (recorrido de área, reconocimiento de cuerpos calientes, entre otros), coordenadas a recorrer, si se desea recibir datos en tiempo real, si se desea recibir fotos por mail. Una vez recibida la misión, el dron evalúa si está en condiciones para volar (si tiene nivel de batería adecuado, si la última calibración todavía es apta, etc.), y en caso positivo, comienza la misión de forma autónoma, recorriendo los puntos geográficos ingresados, evadiendo los posibles obstáculos que estén en su camino mediante el uso de una cámara estereoscópica y sensores de ultrasonido.

Una vez terminada la misión, retorna a las coordenadas de la base, realiza un aterrizaje de precisión, y, una vez confirmado el aterrizaje y apagados los motores, se ajusta la posición del dron mediante el mecanismo de la base, de forma de lograr un buen acople entre las bobinas del dron y la base para poder comenzar la recarga inalámbrica.

Cuando el dron se encuentra ubicado en la base, el mismo envía un reporte de la misión (además de los posibles reportes en tiempo real que pudieron ser elegidos por el usuario al comienzo de la misión), evalúa su nivel de batería, y de ser necesario, activa la recarga inalámbrica de batería.

Este proyecto requirió una etapa experimental muy extensa para probar todos los componentes y su comportamiento en vuelo. Esta etapa ocurrió naturalmente sobre el tramo final del proyecto y se vio afectada por la pandemia del COVID-19.

Índice general

Capitulo 1

Introducción

El presente informe elaborado para la obtención del título de grado de Ingeniería Eléctrica de la Universidad de la República se refiere al desarrollo de un dron de vuelo autónomo con reconocimiento por termografía. A continuación se detallan los antecedentes y objetivos. Información sobre el presupuesto y cronograma del mismo se puede ver en el apéndice A.4.

1.1. Antecedentes

Según la Real Academia Española, el término "dron" (plural drones), surge como adaptación al español del sustantivo inglés "drone", literalmente "zángano", para referirse a una aeronave no tripulada, debido al sonido similar que emiten.
El término consta de una amplia variedad de sinónimos, dentro de los más conocidos tales como UAV (del inglés, Unmanned Aerial Vehicle), RPV (del inglés, Remotely Piloted Vehicle), RPA (del inglés, Remotely Piloted Aircraft), RPAS (del inglés, Remotely Piloted Aircraft System) y otros tales como SPA, UMA, UAS [1].

Los vehículos aéreos no tripulados (UAV) tuvieron sus comienzos en la primera mitad del siglo XIX con los inventos de Cayley, Stringfellow, Du Temple y Cody, entre otros; estos tenían el fin de desarrollar pequeños modelos no tripulados, con el objetivo de mejorar los diseños de las aeronaves tripuladas.
En 1849, el imperio Austríaco realiza el primer ataque con un UAV, los cuales eran globos que contenían hasta 5 bombas, sobre la cuidad de Venecia.
En 1898, Nikola Tesla, a quien se lo considera como el creador del misil de crucero y de la aviación no tripulada, invento el "Teleautomaton", el primer "dron marino", un vehículo capaz de moverse por control remoto y el cual enviaba señales de radio. En 1919, el "Teleautomaton" resurgió como prototipo de torpedo radio-controlado.
Durante la primer guerra mundial se vio enlentecido el avance en los UAV debido tanto a un mayor enfoque en los vehículos tripulados, como a la falta de tecnología necesaria para

poder desarrollarlos. Sin embargo durante la segunda guerra mundial se vio un avance en las tecnologías de los UAV, donde se realizaron mejoras a los sistemas de control remoto, sensores, tales como barómetros y acelerómetros, autonomía de vuelo, entre otros aspectos [2].

De los años 80 a la fecha es donde se ha visto un mayor avance en la tecnología, debido no solo a su uso militar sino al crecimiento en el uso civil. Los vehículos aéreos se pueden clasificar de varias formas, una de ellas según el uso, tanto militar como civil. Los drones de uso militar con funciones como seguimiento, detección, protección portuaria, entre otras, mientras que los drones de uso civil se pueden re-clasificar como de uso comercial o privado. Dentro del uso privado se encuentran aplicaciones tales como fotografías de paisajes, recreación (por ejemplo, las carreras de drones), y demás.
Dentro del uso comercial se encuentran distintas ramas de aplicación como lo pueden ser Cartografía y Topografía, Riesgos Naturales, Incendios Forestales, Seguridad, Agricultura, entre tantos otros.
En la figura 1.1, se puede apreciar distintos modelos de UAV.

Figura 1.1: Distintos modelos de Drones. Imagen obtenida de **MasScience**.

Dentro del Instituto de Ingeniería Eléctrica de la Facultad de Ingeniería existe una línea de trabajo en el área de robótica, y más específicamente, robótica móvil. Se han implementado robots terrestres como ser SULLA y ROVER [3], acuáticos como el Pez Robot (versión I y II) [4] [5], y aéreos, siendo el primero el AMFO1-Bobby, un avión a escala radio-controlado y con algunas capacidades de vuelo autónomo. Más recientemente se han desarrollado varios proyectos basados en la implementación de drones no tripulados, o UAV .

Dentro de esta rama se comenzó con la serie de proyectos uQuad (I, II, III) [6] [7], los cuales construyeron desde cero un cuadricóptero, abarcando temas desde la mecánica de los motores que permite controlar el mismo en el aire, hasta la electrónica para controlar los motores, llegando en las últimas iteraciones a lograr implementar vuelo autónomo mediante redes neuronales.
Luego, se comenzó el trabajo sobre el Termodron (I, II) [10] [43], el cual consiste en un dron de vuelo autónomo, equipado con una cámara termográfica, con el cual es posible comunicarse mediante la red de telefonía celular, tanto para comandarle misiones (tareas) a realizar (recorrido de un perímetro dado, identificación de cuerpos calientes), como para recibir información del mismo (estado del dron, imágenes capturadas). En la figura 1.2 se presentan tanto el dron implementado por Termodron I como por Termodron II.

Figura 1.2: Termodron I (Izquierda) y Termodron II (Derecha). Imagen obtenida de [10] y [43].

1.2. Objetivos

El proyecto tiene como objetivo llevar a cabo las mejoras al sistema implementado en el proyecto Termodron II, tanto del punto de vista de hardware como de software, con el fin de lograr un prototipo funcional, que logre poner en funcionamiento un sistema autónomo para el reconocimiento de puntos calientes mediante termografía. En particular, se buscará implementar el algoritmo de aterrizaje de precisión, mejorar los enlaces de comunicación, mejorar el control de vuelo del dron, así como los algoritmos para la

definición de misiones y la comunicación con el usuario.

Con este fin se realizan las siguientes mejoras:

- Se incorpora nueva cámara térmica con radiometría.
- Se implementa una interfaz de usuario para mejorar la interacción del usuario con el dron, manteniendo el sistema previo de correo electrónico, dada su robustez.
- Se desarrolla un sistema de aterrizaje de precisión del dron, de forma de lograr un correcto aterrizaje sobre su base.
- Se incorpora una cámara estereoscópica, para la detección de obstáculos, y sensores de ultrasonido como respaldo.
- Se construye un nuevo dron desde cero, incluyendo nuevos componentes, en particular, nuevo controlador de vuelo, módulo GPS, y motores.
- Se diseña la base del dron, de forma de poder alojar el hardware necesario para las funciones como estación de control, comunicación con el usuario y el dron, y para la recarga inalámbrica.

Capitulo 2

Sistema Implementado

2.1. Introducción

Un cuadricóptero es un tipo de helicóptero con cuatro pares motor-hélice que actúan como propulsores, al empujar éstos el aire hacia abajo, se genera una fuerza hacia arriba (llamada empuje). Esta es la idea básica del funcionamiento de un cuadricóptero: al controlar la velocidad de giro de los motores se controla el empuje generado por los mismos. De los cuatro motores, dos giran en sentido horario y los otros dos en sentido antihorario, como se observa en 2.1.

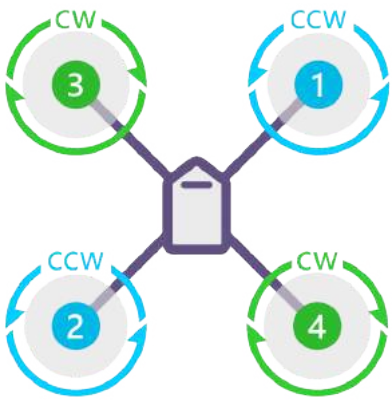

Figura 2.1: Orden de giro de los motores de un cuadricóptero. Imagen obtenida de **Ardupilot**.

Para alterar la altura de vuelo del dron se debe ajustar la velocidad de los motores, ajustando así el empuje generado por los mismos: si el empuje supera al peso del dron, se eleva; si el empuje es menor que el peso del dron, éste desciende; si se igualan se mantiene

la altura. Por otro lado, ajustando la velocidad de un par de los motores, es posible lograr giros respecto a los ángulos de *pitch*, *roll* o *yaw* (dichos ángulos se pueden observar en la figura 2.2) del dron. En la figura 2.3 se muestra cómo las variaciones de velocidad de los motores afectan los giros del dron. Para lograr cambios en el ángulo de *yaw*, se debe aumentar la velocidad de los motores que giran en el sentido opuesto al sentido deseado, es decir, para un giro de *yaw* antihorario, se debe aumentar la velocidad de los motores que giran en sentido horario, y disminuir la de los que giran en sentido antihorario. Para que el dron avance o retroceda se debe cambiar el ángulo de *pitch*, lo cual se logra aumentando la velocidad de los motores traseros y disminuyendo la de los frontales para avanzar, o viceversa para retroceder. Para moverse hacia los costados se aplica el mismo principio que para el avance o retroceso, pero respecto al ángulo de *roll*.

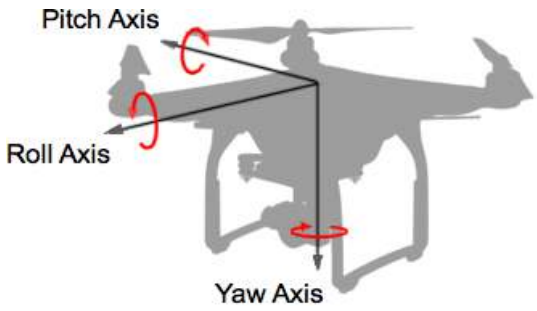

Figura 2.2: Orientación de los ejes de un dron. La dirección de avance del dron es según el eje descrito por el ángulo de *roll*.

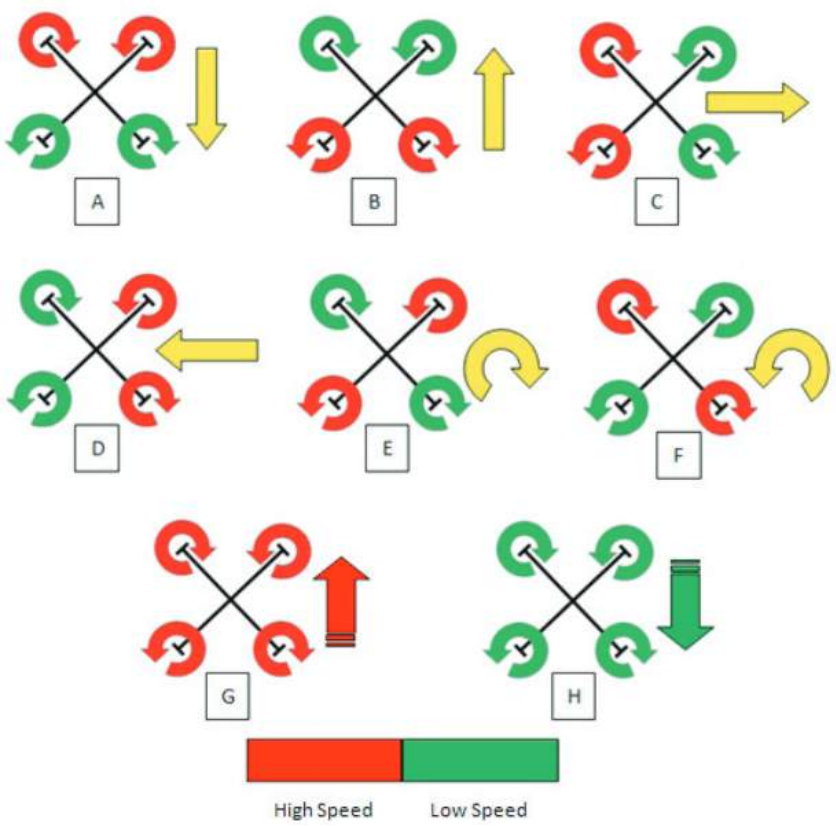

Figura 2.3: Principios de movimiento de un cuadricóptero: (A) avance (dirección de vuelo hacia abajo en la figura), (B) retroceso, (C) movimiento hacia la izquierda, (D) movimiento hacia la derecha, (E) giro en sentido horario, (F) giro en sentido antihorario, (G) ascenso, (H) descenso. En rojo se muestran los motores con alta velocidad y en verde los de baja velocidad. Imagen obtenida de **Física de un quadcóptero**.

2.1.1. Sistema Termodron

El sistema Termodron consta de tres entidades: el dron, la base y la interfaz de usuario. Éstas tres se comunican entre sí de forma inalámbrica previo a comenzar una mision, durante la misma y una vez finalizada la misma.

El dron es un cuadricóptero de vuelo autónomo para relevamiento termográfico mediante una cámara térmica, detección y evasión de obstáculos mediante una cámara estereoscópica, corrección de posición mediante GPS diferencial (mediante un caster[1] ubicado en la fortaleza del cerro), aterrizaje de precisión, recarga inalámbrica y reporte de datos tanto durante el vuelo mediante el uso de Amazon Web Services (AWS), como luego de terminado el mismo. El dron puede observarse en la figura 2.5.

La base consta de dos marcadores fiduciarios que sirven como objetivo y guía para el aterrizaje de precisión, así como un sistema de asistencia al acople de las bobinas de recarga inalámbrica. También funciona como cubierta protectora para el dron y todo el hardware de la recarga inalámbrica y del resto del sistema. Se encarga de recibir la misión

[1]caster: servidor que envía los mensajes de corrección de posición diferencial.

configurada por el usuario, verificar el estado del dron previo al vuelo, y en caso de que esté listo, enviar la misión hacia el dron; además, monitorea el estado de la batería del dron, activando la recarga inalámbrica cuando sea necesario. Para la base se implementó el software que controla la comunicación con el usuario y el dron, se realizó un diseño conceptual de la misma, pero no fue construida.

Finalmente, la interfaz gráfica se implementa para facilitar la configuración de misiones y visualización de los reportes en tiempo real del dron. Dado que la interfaz se comunica mediante correos electrónicos con el resto de las entidades, el usuario puede entonces comandar misiones desde cualquier parte del mundo.

En la figura 2.4 se puede ver un diagrama de bloques del sistema implementado, donde se muestran las tres entidades principales: interfaz, base, dron, así como también los diferentes módulos de cada una de éstas, y las interacciones entre las entidades.

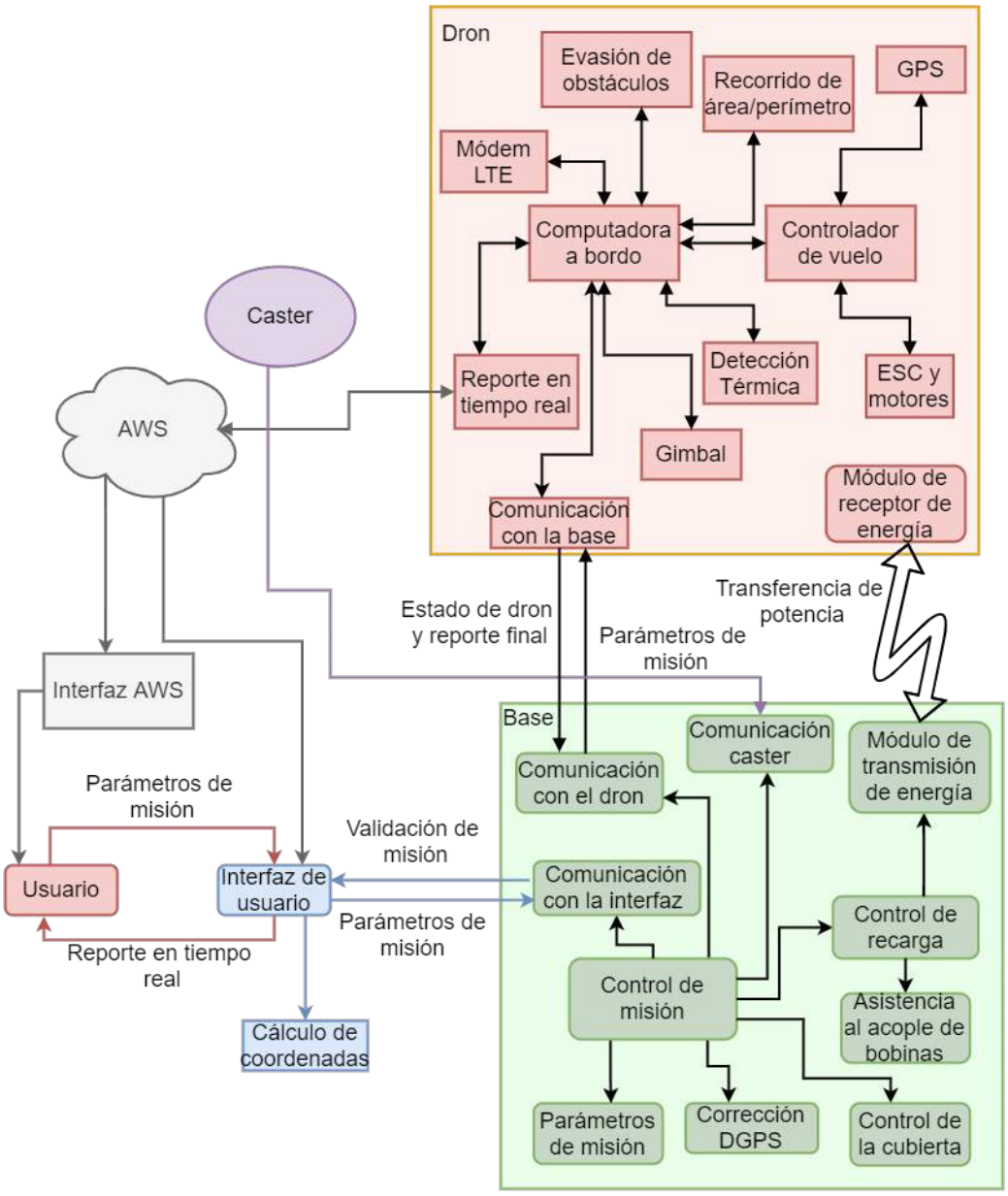

Figura 2.4: Diagrama completo del sistema implementado por Termodron III.

Figura 2.5: Entidad dron implementada.

El funcionamiento del sistema es el siguiente: el usuario utiliza la interfaz gráfica para configurar la misión que desea que el dron realice, ajustando los puntos a recorrer, el modo de recorrido de los mismos (recorrido de un perímetro o barrido de un área), el tipo de detección térmica (detección de cuerpos calientes o de focos ígneos), si se desea o no reportes en tiempo real o reportes por correo electrónico una vez terminada la misión. Una vez configurada la misión, la interfaz envía un correo electrónico al usuario con los parámetros elegidos para su verificación, y también envía uno hacia la base, la cual está en estado de espera por misiones entrantes.

Una vez recibido el correo, la base procede a pedir el estado del dron mediante correo electrónico hacia éste. El dron se encuentra encendido, con los motores apagados, sobre la base, y en estado de espera por un pedido de reporte de estado. Una vez recibido el estado del dron, la base verifica primero el nivel de la batería, en caso de ser menor al 95 % del total activa la recarga inalámbrica, y una vez terminada, pide nuevamente el estado al dron. En caso de que la batería tenga un nivel adecuado, la base procede a verificar el estado general del sistema del dron, revisando así que no hayan otros inconveniente que impidan el vuelo (por ejemplo, falta de calibración de sensores, o poca señal de GPS). En caso de que el dron no esté en condiciones de volar, se notificará al usuario mediante el uso de correo electrónico, para que se realice un chequeo del sistema. En caso de que sí esté listo para volar, se manda la misión al dron y la orden de comienzo.

Recibida la misión, el dron carga los parámetros configurados por el usuario y

procede a realizar el despegue, luego el recorrido (con la toma de fotografías térmicas y evasión de obstáculos, de ser necesarias), y finalmente retorna a la base, realizando el aterrizaje de precisión. Una vez sobre la base y con los motores apagados, notifica a la base que la misión fue completada, enviando los reportes y fotos de la misión. La base entonces activa el sistema de asistencia al acople de bobinas, el cual ajusta la posición del dron para lograr un alineamiento preciso entre la bobina del dron y la de la base, lo cual es clave para una correcta transferencia de energía durante la recarga.

Siendo este proyecto el último en la línea Termodron, se realizaron mejoras en muchos de los componentes y funcionalidades de los proyectos anteriores, a continuación se listan dichas mejoras.

- Mejora del control del dron mediante un nuevo controlador de vuelo.
- Mejora de la autonomía de vuelo, utilizando una batería con el doble de capacidad.
- Mejora de la relación empuje-peso del dron, mediante motores de mayor empuje.
- Mejora del sistema de detección de cuerpos calientes, utilizando una nueva cámara térmica con radiometría.
- Mejora del sistema de detección de obstáculos, utilizando una cámara estéreo junto con sensores de ultrasonido.
- Mejora de la capacidad de procesamiento de datos a bordo del dron gracias a una nueva computadora a bordo.
- Se reescribe el código desde cero (excepto el código para GPS diferencial, el cual se mantiene de Termodron II), manteniendo el lenguaje Python y la arquitectura de *threads* en paralelo.

A su vez, se agregaron nuevas funcionalidades al sistema, los cuales se listan a continuación.

- Implementación de un sistema de aterrizaje autónomo de precisión.
- Implementación de barrido de área[2]: dados los puntos que definen el contorno de un polígono, elegidos por el usuario, se planea una trayectoria para recorrer el interior de dicho polígono, cubriendo toda el área.
- Implementación de un algoritmo de detección de objetos basado en una cámara estereoscópica, con capacidad de detección de objetos que disten hasta 10 m de la misma.

[2]Esta funcionalidad estaba implementada en Termodron I, pero en otro lenguaje de programación. Se decidió utilizar una *script* distinto para este proyecto, en Python.

- Implementación de un algoritmo de evasión de obstáculos.
- Implementación de un algoritmo de detección de cuerpos en cierto rango de temperaturas.

De los sistemas anteriores se mantiene el sistema de recarga inalámbrica y el software para la corrección diferencial de posición satelital (GPS diferencial, o DGPS), ambos de Termodron II.

Para más detalles sobre las prestaciones del sistema implementado referirse al anexo, sección A.4, para más detalles sobre los componentes del sistema ver el capítulo 4, y para detalles sobre el software implementado en el sistema ver el capítulo 5.

2.2. Dron

A continuación se detallan los principales componentes tanto a nivel de hardware, software y aplicación, del dron.

En la figura 2.6 se realiza un diagrama de bloques de los componentes y entidades del sistema, a nivel de hardware como de software.

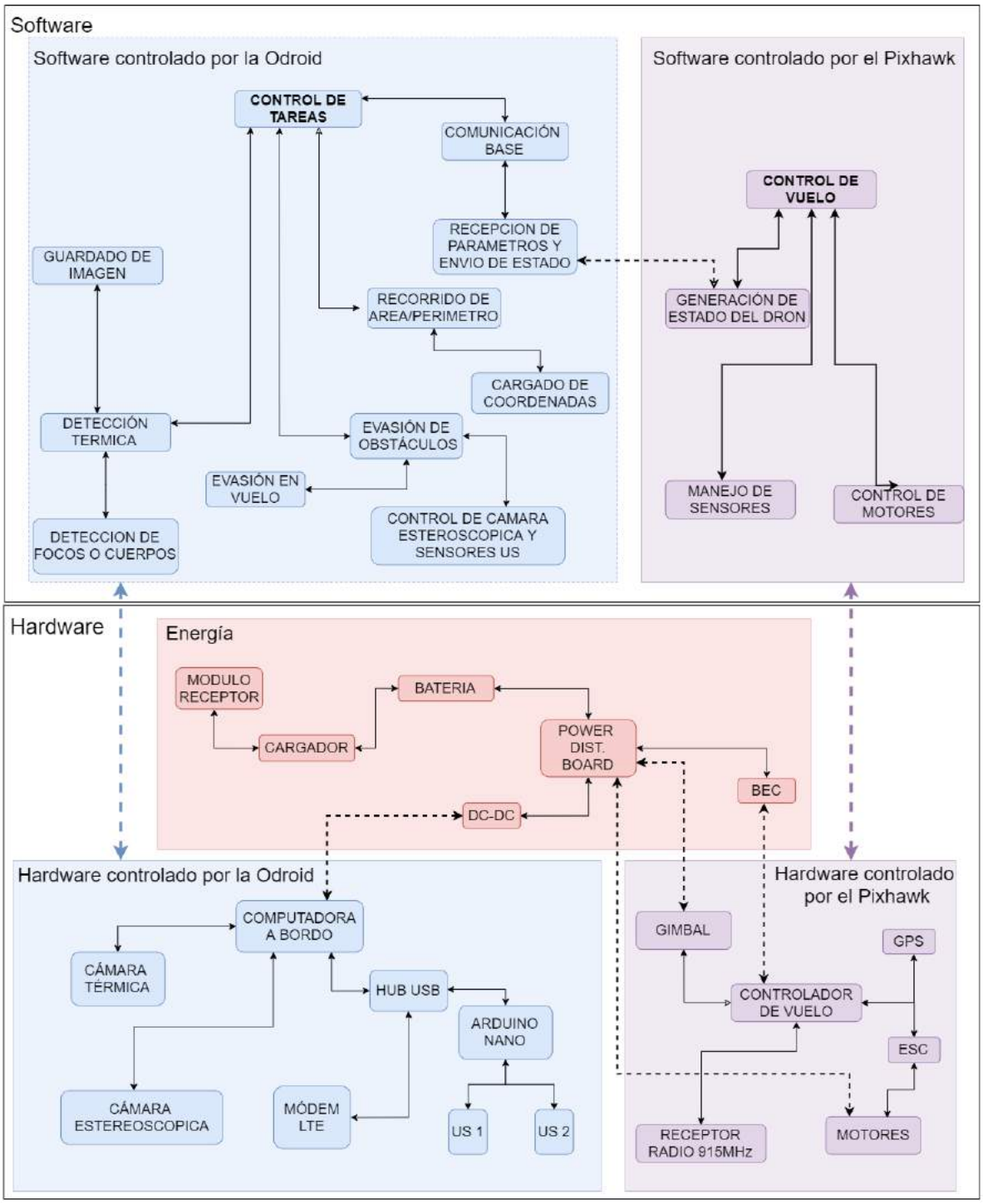

Figura 2.6: Diagrama de bloques de las capas de software y hardware de los componentes del dron. Las flechas punteadas indican objetos de distintos módulos que se relacionan entre sí (por ejemplo, el gimbal es una pieza de hardware controlada por el controlador de vuelo, pero a su vez es alimentado por el Power Distribution Board, que pertenece al módulo de Energía).

2.2.1. Hardware

Como se mencionó anteriormente, se realizaron mejoras de piezas clave de hardware respecto al sistema anterior, y además se construyó un nuevo dron desde cero, manteniendo el sistema de Termodron II para pruebas iniciales mientras se ensamblaba el dron de Termodron III. Este nuevo dron utiliza algunos componentes iguales a los anteriores, como ser el armazón y el controlador de telemetría. Los componentes que no fueron mantenidos de los proyectos anteriores (debido a que existe una nueva versión, o una alternativa con mejores prestaciones) son listados a continuación. Por más información sobre los mismos, ver 4.

- Nuevo controlador de vuelo: Pixhawk 2.1 o Pixhawk Cube
- Nuevos motores, con mayor empuje: Emax mt3510 y correspondientes hélices de fibra de carbono 1447[3]
- Nuevo módulo de navegación satelital: Here 2 GNSS
- Nueva cámara térmica: FLIR Lepton 3.5
- Nuevos sensores de ultrasonido: MaxBotix MB1361 XL
- Nueva computadora a bordo: Odroid XU4
- Nuevo módem 3G/LTE: Huawei E353 HSPA+
- Cámara estereoscópica: Intel RealSense D415

En la figura 2.7 se presenta el diagrama de conexión eléctrica del sistema.

[3] 14 pulgadas de largo y 4.7 pulgadas de avance vertical.

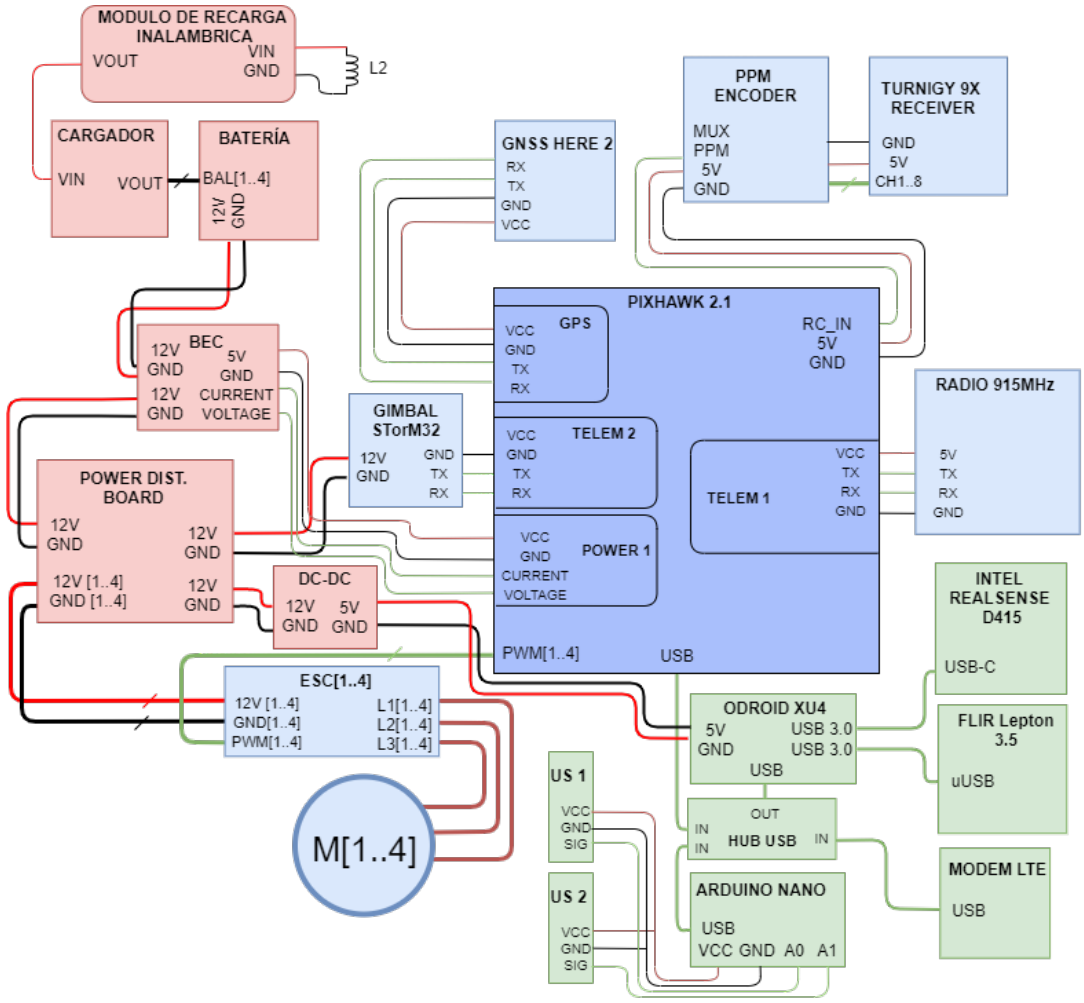

Figura 2.7: Diagrama de hardware de los componentes del dron.

En el capítulo 4 se realiza una descripción detallada del hardware utilizado.

2.2.2. Software

Se realiza un cambio de la computadora a bordo del dron, pasando de un microcontrolador como era el Arduino Due (el cual se programa en lenguaje similar a C++) a una Single Board Computer, la Odroid XU4 (la cual permite programar con distintos lenguajes y entornos de programación). Debido a esto se escribió desde cero todo el software del dron y la base, manteniendo solo el código para la corrección de GPS diferencial de Termodron II. Se utilizó el lenguaje de programación Python para escribir el código, ya que era el lenguaje utilizado por Termodron II para el software de la base, y por la existencia de la librería Dronekit-Python, la cual permite programar el dron y controlarlo desde una computadora a bordo. Además, utilizar Python da acceso al uso de una gran cantidad de librerías de libre acceso que permiten controlar

distintas piezas de hardware, en particular las cámaras, como también permiten el procesamiento de imágenes en tiempo real, y reporte de datos por medio de Amazon Web Service.

Además, se implementaron distintas librerías, separadas por funcionalidad, como lo son:

- libDron
- libStereo
- libThermal
- libDGPS
- libAWS
- libMail
- libArduino

Se detalla su funcionamiento en el capítulo 5.

2.2.3. Sistema de comunicación

Para la comunicación entre el usuario y el conjunto dron-base se utilizan e-mails, dada su robustez, familiaridad de uso por el público general, y facilidad de implementación, así como también se puede utilizar Amazon Web Services (AWS) de forma opcional para la comunicación, y para reportes del dron durante el vuelo. Se tiene además una conexión desde la base hacia el dron mediante la antena de telemetría de 915 MHz para poder enviar los mensajes de corrección de GPS diferencial.

En la figura 2.8 se presenta un diagrama de bloques del sistema de comunicación implementado. En el capítulo 7 se realiza una descripción detallada de los distintos componentes, sus roles y el medio de comunicación utilizado.

Se implementó también una interfaz gráfica con el fin de generar un ambiente sencillo para el usuario donde pueda realizar la configuración de las misiones y recibir un reporte resumido del dron durante la misión.

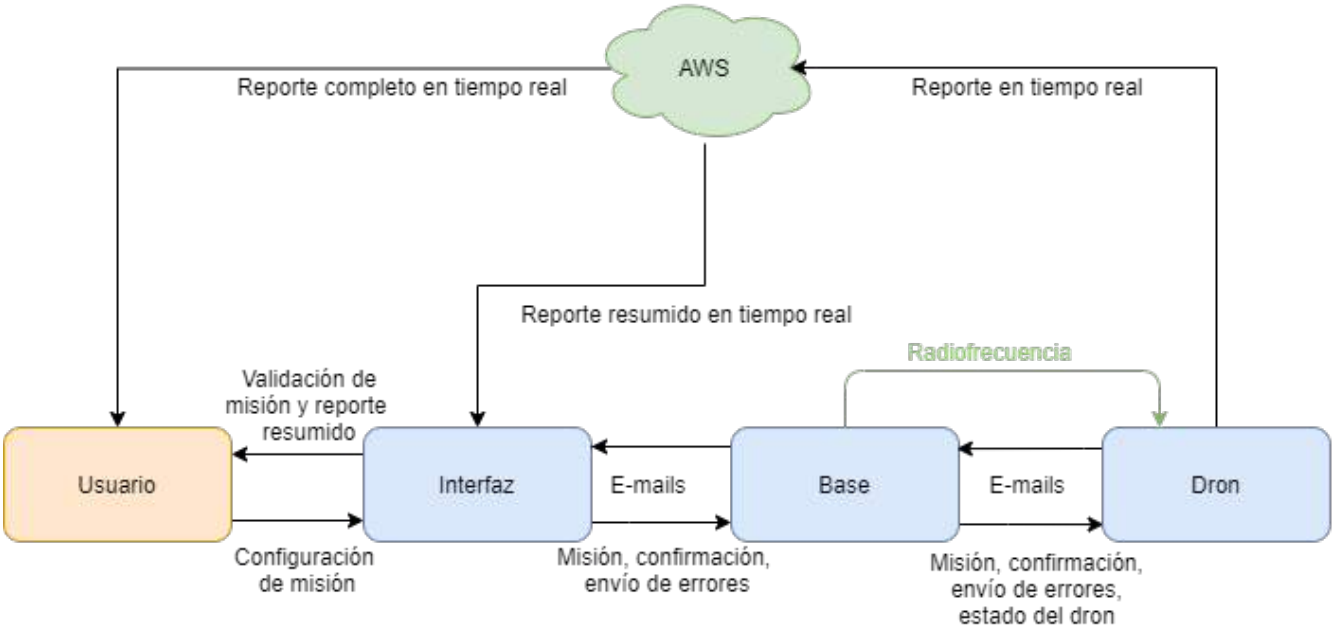

Figura 2.8: Diagrama de bloques del sistema de comunicación.

2.2.4. Aterrizaje de precisión

Dada la necesidad de poder posicionar el dron de forma correcta sobre la base para poder acoplar las bobinas de la recarga inalámbrica, se implementó un sistema de aterrizaje de precisión. El algoritmo utiliza una cámara estereoscópica (Se puede utilizar cualquier cámara, pero se decide utilizar la cámara estereoscópica para esta funcionalidad también ya que también posee una cámara de espectro visible y el gimbal utilizado permite apuntar la misma hacia abajo o hacia adelante) y dos marcadores identificables mediante detección de patrones por software.

El algoritmo se basa en detectar el marcador, obtener su posición respecto a la posición del dron, dirigir el dron hacia el marcador, y, si el ángulo de visión del dron respecto al marcador es menor a cierto valor de umbral se le da la orden de descender una altura determinada, en caso contrario, se mantiene la altura. Una vez alcanzado una altura mínima sobre el nivel del marcador, se cambia el modo de vuelo al dron al modo LAND, el cual disminuye la velocidad vertical del dron de forma lenta, completando el proceso de aterrizaje.

En el capítulo 6 se realiza una descripción detallada del algoritmo implementado, su base teórica y su funcionamiento.

2.2.5. Evasión de obstáculos

Se decide realizar algunas mejoras al sistema de evasión implementado por Termodron II, comenzando principalmente por los sensores de proximidad utilizados. Estos eran

sensores de distancia por ultrasonido, y tenían un rango de detección de 2 metros, lo cual, sumado a la velocidad de procesamiento del Arduino a bordo del dron, limitaba la velocidad de respuesta ante un obstáculo, y por ende, la velocidad de vuelo del dron [10]. Además, dado el principio de funcionamiento de estos sensores, basados en ondas de sonido, los mismos sufren de degradaciones de funcionamiento al ser utilizados en superficies que no reflejen ondas sonoras, u objetos muy pequeños, como por ejemplo, ramas de un árbol con poco follaje. Debido a esto, y dada la incorporación de una computadora a bordo con mayor poder de procesamiento, se decide utilizar un nuevo método de detección de obstáculos mediante una cámara estereoscópica (o cámara estéreo) Intel RealSense D415, agregando además un sensor ultrasónico de distancia con un mayor rango de detección como respaldo a la misma. En el capítulo 4 se entra en más detalle sobre la elección de esta cámara y el sensor de ultrasonido.

El algoritmo de evasión está vinculado directamente al algoritmo de detección de objetos implementado. Este último utiliza la información de imagen de profundidad[4] obtenida de la cámara estéreo, realizando una búsqueda en la imagen de profundidad para obtener sólo los objetos dentro del rango de distancias de interés (entre 0.5 y 9 metros), luego se segmenta la imagen según los puntos que cumplan esta condición y se crea una máscara binaria, mostrando en negro todas las partes de la imagen que están fuera de rango, y en blanco las que no. Luego se aplica una búsqueda de los grupos de píxeles que están dentro del rango de detección deseado, se descartan los que tengan área muy pequeña, para evitar falsos positivos, y se procede a calcular el área en cada contorno. Luego se toma el menor valor de distancia obtenido, se marca el contorno correspondiente a dicho valor, y se muestra en pantalla la distancia mínima obtenida entre todos los contornos analizados. Para más información referirse a la sección **??**.

El cuadro de imagen se divide en tres regiones verticales, aplicándose el algoritmo de detección de objetos a cada región por separado, de forma de obtener una mejor información sobre la ubicación de los objetos en el espacio.

El algoritmo de evasión utiliza la información de distancia de cada sector del cuadro de la cámara, y según si hay objeto detectado y la distancia del mismo hacia la cámara, decide la acción a realizar. En caso de haber dos sectores ocupados, y un sector libre, se ordena mover el dron al sector libre. En caso de haber un sector ocupado y dos libres, se mueve el dron en función de que región está ocupada. En caso de tener los tres sectores ocupados, el dron se gira hacia un costado y si no detecta obstáculos, avanza en esa dirección, luego de una distancia dada, vuele a girar en dirección contraria para evaluar si no hay obstáculos y entonces vuelve a avanzar en el sentido que tenía anteriormente. En las figuras 2.9, 2.10, 2.11 y 2.12 se presenta un diagrama con las diferentes acciones a tomar por el dron según la ubicación de los obstáculos detectados.

[4]La imagen de profundidad contiene en cada píxel el valor de distancia de ese punto hacia la cámara, a menos de un factor de escala propio de la cámara.

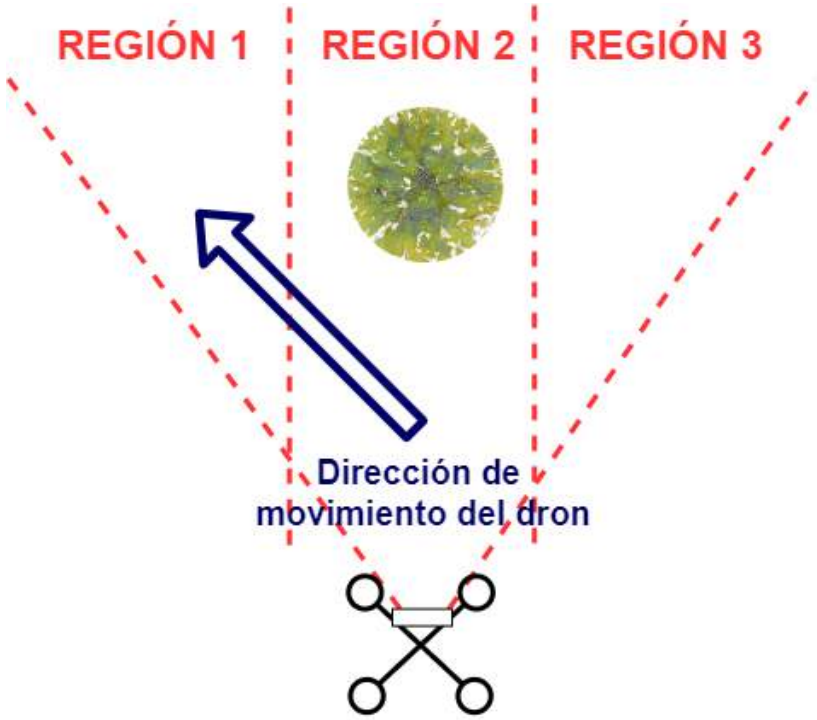

Figura 2.9: Acción a tomar en caso de presentarse un objeto en la región central.

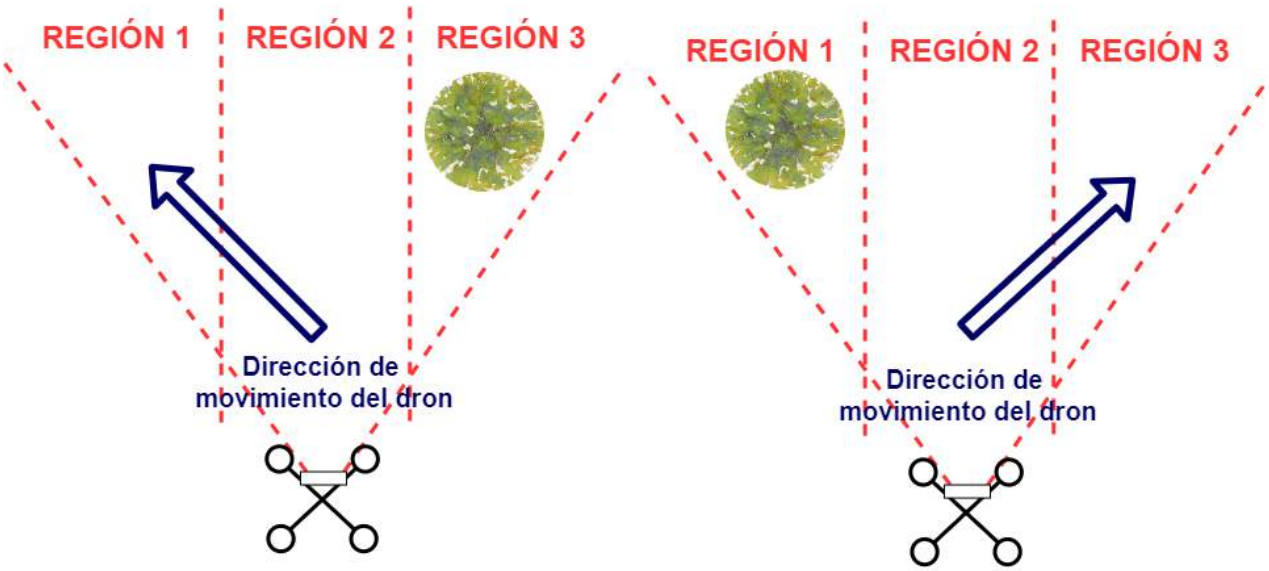

Figura 2.10: Acción a tomar en caso de presentarse un objeto en la cualquier de las regiones laterales.

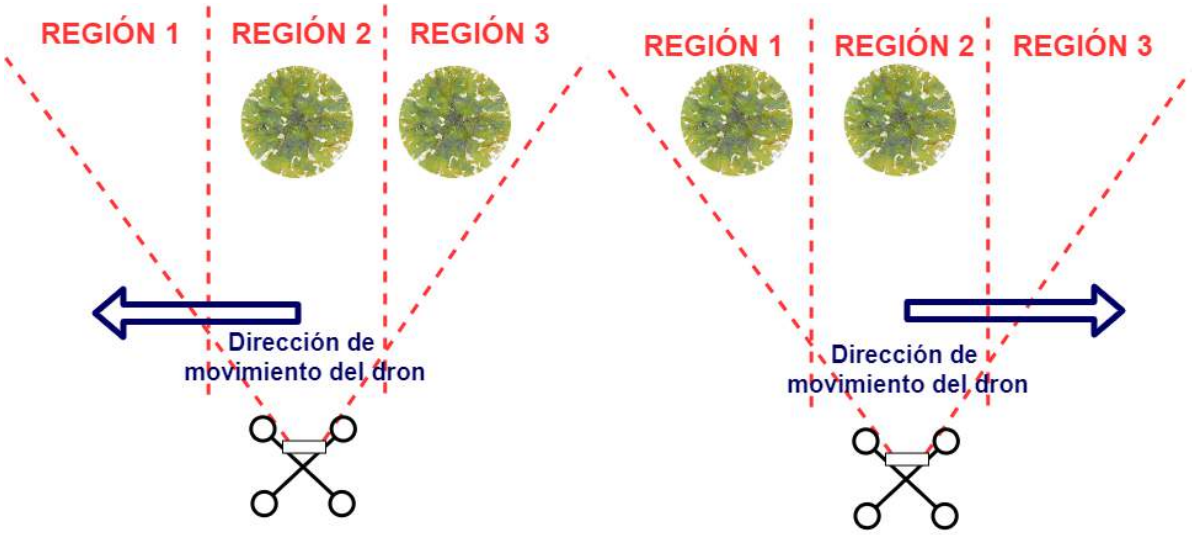

Figura 2.11: Acción a tomar en caso de presentarse un objeto en dos de las regiones.

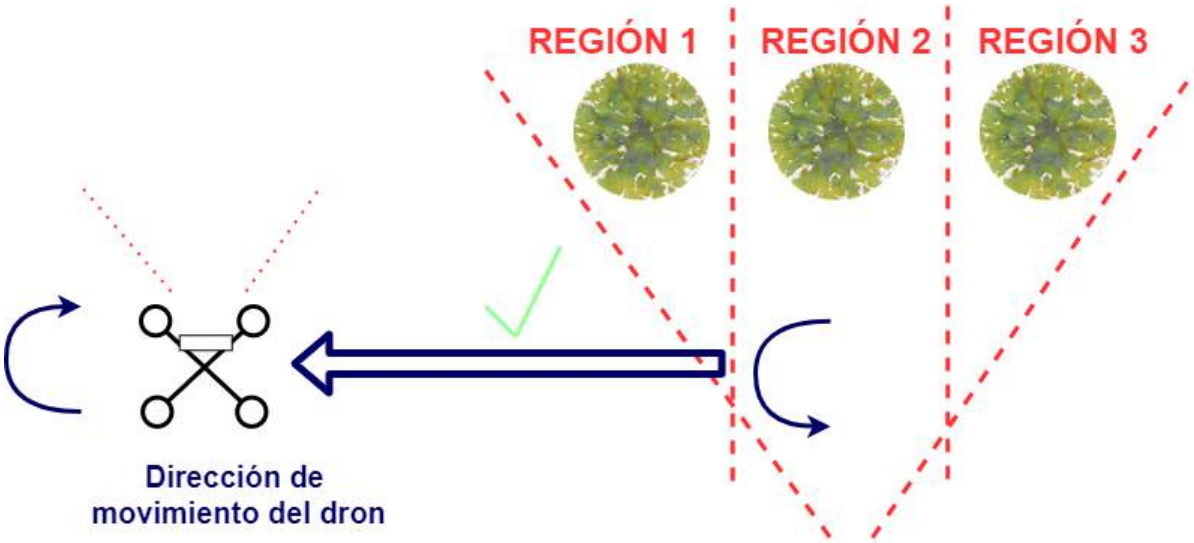

Figura 2.12: Acción a tomar en caso de presentarse un objeto en las tres regiones.

2.2.6. Detección Térmica

El sistema Termodron lleva su nombre gracias a esta funcionalidad, la cual fue implementada por primera vez en el proyecto Termodron I. En este proyecto se mejora el hardware utilizado para la detección térmica, utilizando una nueva versión de la cámara térmica FLIR Lepton, ahora con radiometría[5], también se mejora el algoritmo

[5]La adición de radiometría permite obtener valor de temperatura en cada píxel de la imagen térmica, e independiza estos valores de la temperatura interna de la cámara.

de detección térmica y el hardware en el que se ejecuta dicho algoritmo, en este caso, la computadora a bordo: Odroid XU4.

En los proyectos Termodron I y II el algoritmo consistía en tomar una imagen térmica y hacer la búsqueda del cuerpo más grande cuya temperatura estuviera dentro de un rango de temperatura dado. Una vez encontrado, guardaba la imagen obtenida.

El algoritmo implementado en este proyecto se basa en esta idea, y aprovechando el nuevo hardware, se decide implementarlo en el lenguaje de programación Python 3, gracias a su facilidad de uso y acceso a múltiples librerías de procesamiento de imagen y la disponibilidad de una librería para controlar la cámara térmica FLIR Lepton 3.5.

El algoritmo realiza la búsqueda de objetos[6] cuya temperatura esté dentro de un rango dado. Luego se dibuja el círculo que envuelve al objeto de mayor área entre los detectados y se guarda la imagen. Para evitar capturar el mismo objeto múltiples veces (al moverse el dron sobre éste, por ejemplo) se establece que un nuevo objeto debe estar por lo menos a más de un radio (refiriéndose al radio del círculo que lo envuelve) de distancia del círculo que envuelve al cuerpo hallado anteriormente. El rango de temperaturas en el cual se buscan estos objetos es configurable entre dos modos: modo de detección de cuerpos[7] (temperatura entre 30 y 45 °C), y detección de focos de incendio (temperatura superior a los 100 °C).

En la sección 5.2.4 se realiza una descripción detallada del algoritmo implementado.

En la figura 2.13 se observa una imagen tomada en vuelo utilizando el algoritmo de detección térmica en modo de detección de cuerpos calientes.

[6] Considerando como cuerpo un grupo de píxeles de similar valores de temperatura.

[7] En este caso se utiliza cuerpos refiriéndose a cuerpos humanos, o de animales.

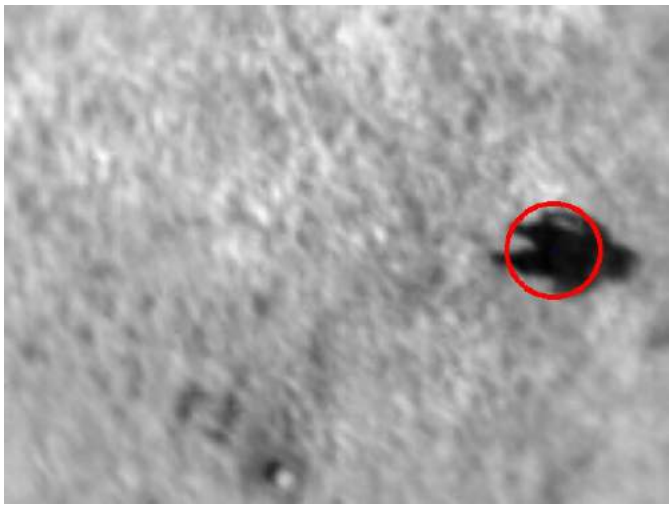

Figura 2.13: Imagen tomada en vuelo durante el día utilizando el algoritmo implementado. Se observa que el cuerpo de la persona es detectado por la cámara, y el mismo está a menor temperatura que el suelo, debido a que fue tomada en verano (temperaturas mayores son colores más claros).

2.3. Base

Como se mencionó anteriormente la base no fue implementada, ya que su implementación no formaba parte del alcance del proyecto, sino que era un objetivo opcional. Se realizó un diseño 3D a modo de concepto, junto con la implementación del software de comunicación necesario para su utilización con el sistema interfaz, base, dron.

La base debe cumplir los siguientes objetivos: contener los componentes de hardware necesarios para la recarga inalámbrica, la computadora que controle dichos componentes de hardware y se comunique con el dron y el usuario, y proteger al dron de las inclemencias climáticas. Todo esto fue tenido en cuenta a la hora de diseñar el modelo 3D de la misma.

El modelo diseñado de la base consiste en un nivel inferior donde se alojan los componentes electrónicos, entre ellos, la computadora principal de la base, la cual se encargará de la comunicación inalámbrica con el usuario y con el dron, además de controlar otros componentes del sistema. También incluye los controladores para los motores que mueven el sistema de alineación del dron y las dos mitades de la cubierta de la base. Se agregan también todos los componentes para la recarga inalámbrica del dron, implementada por Termodron II. Luego, en un nivel superior se tienen los marcadores de referencia utilizados para el aterrizaje de precisión, así como las barras de asistencia al acople de las bobinas, las cuales se encargan de eliminar el pequeño error de posición que el dron pueda haber tenido en el aterrizaje. Finalmente, se coloca una cubierta la cual puede abrirse o cerrarse, para poder proteger al dron cuando está aterrizado.

Por más detalles sobre los componentes utilizados y el sistema mecánico, dirigirse al capítulo 8.

Capitulo 3

Sistema previo a Termodron III

En el presente capítulo se pretende realizar una introducción al sistema previo a Termodron III (Termodron II) detallando hardware, funcionalidades y mejoras realizadas por el mismo. Se realiza esta introducción con el fin de ilustrar el sistema con el que se comienza el proyecto y cuáles son las posibles mejoras a realizar.

Para más detalles respecto al proyecto de Termodron II, referirse a [10].

3.1. Introducción

A continuación se listan las prestaciones del sistema implementado por Termodron II.

- Recarga inalámbrica de la batería en 4 horas.
- Corrección diferencial de la posición por GPS con un error de 2m.
- Evasión de obstáculos ubicados en un rango de 2 m del dron.
- Detección de focos de calor en vuelo.
- Comunicación en tiempo real entre el usuario y el sistema.
- Envío de fotos y estados del dron al usuario.

En la figura 3.1 se puede observar el sistema implementado por Termodron II.

Figura 3.1: Sistema implementado por Termodron II, consiste en el dron y su base (llamada estación de monitoreo y control). Imagen obtenida de [10].

3.2. Hardware

Termodron II reutiliza gran parte del dron implementado por Termodron I, realizando algunos cambios en el mismo buscando más robustez en los componentes y mejor rendimiento. A continuación se detallan los componentes modificados con respecto a Termodron I:

- Se cambia el firmware del controlador de vuelo (Pixhawk 1), pasando de PX4 a ArduPilot.
- Se agrega una antena de telemetría extra para la comunicación entre la estación de monitoreo y control y el dron.
- Se realizó un estudio de los motores, concluyendo con el cambio de los mismos.
- Se actualizaron los sensores de ultrasonido a una versión más nueva.
- Se implementa un gimbal STorM32 para el control de la cámara térmica.
- Se diseña un PCB para conectar los cuatro sensores de ultrasonido al Arduino DUE a bordo.

3.2.1. Evasión de obstáculos

Para la detección de obstáculos, Termodron II utiliza sensores ultrasónicos de distancia, los cuales tienen capacidad para detectar objetos hasta dos metros de distancia. Se cuenta con cuatro sensores de ultrasonido colocados hacia el frente del dron, de forma de poder detectar obstáculos hacia la derecha, izquierda, adelante y abajo. Para la evasión de obstáculos se controla el movimiento del dron por medio de la velocidad de los motores con el fin de evadir el mismo, esto es realizado mediante software por la computadora a bordo.

3.2.2. Detección de focos de calor

Termodron II utiliza la cámara térmica FLIR Lepton 1.5 conectada a un microcontrolador Arduino Due mediante el protocolo SPI.

El algoritmo de detección térmica implementado fija el umbral de intensidad deseada para la imagen térmica, y mediante un recorrido de la imagen busca el cuerpo de mayor tamaño que se encuentre dentro de este umbral. Una vez detectado, notifica a la estación de monitoreo y control y le envía la foto obtenida. El algoritmo se mantiene en ejecución, analizando cada cuadro proveniente de la cámara térmica.

3.2.3. Sistema de comunicación

Termodron II implementó un sistema de comunicación de mayor redundancia respecto a Termodron I, incorporando una segunda antena de telemetría para la comunicación con la estación de monitoreo y control. En la figura 3.2 se presenta el esquema de la comunicación realizado.

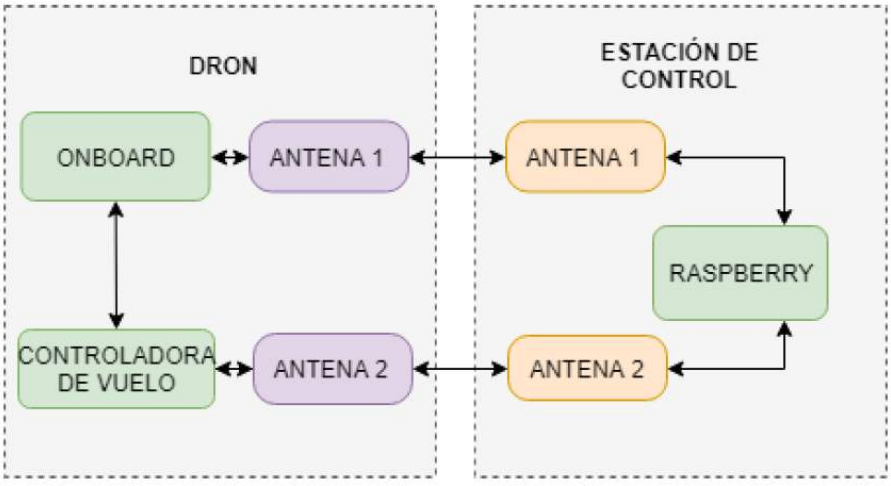

Figura 3.2: Esquema del sistema de comunicación de Termodron II. Imagen obtenida de [10].

El enlace de comunicación entre la computadora a bordo (OnBoard) y la estación de control permite el envío de imágenes, y del estado del dron en tiempo real, mientras que la comunicación de la controladora de vuelo y la estación de control permite establecer un control en tiempo real de la misión y el envío de comandos de emergencia en caso de ser necesarios (aterrizaje y retorno a la base).

La comunicación mediante la computadora a bordo y la controladora de vuelo permite el envío de comandos para la evasión de obstáculos.

3.2.4. Recarga Autónoma

A continuación se detalla uno de los hitos más importantes del grupo Termodron II, la recarga autónoma de la batería del dron.

Termodron II logró diseñar e implementar un sistema de recarga inalámbrica con una eficiencia del 57 % y permitiendo la recepción de $26W$ en el dron, logrando cargar la batería de tres celdas, y 5000 mAh en aproximadamente cuatro horas.

Se recomienda ver [10] para ver en más detalle este tema.

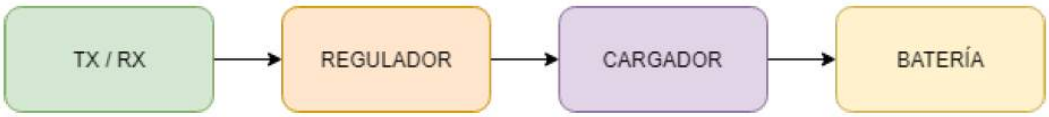

Figura 3.3: Esquema del sistema de recarga autónoma de Termodron II. Imagen obtenida de [10].

En la figura 3.3 se presenta un esquema de la recarga inalámbrica implementada, se destaca que el diseño es externo y se le realizó modificaciones acordes a las necesidades del proyecto. A continuación se explicarán cada uno de los módulos presentes en 3.3.

3.2.4.1. Módulos de TX/RX

Los requerimientos presentados para los módulos de transferencia de energía fueron realizados tomando en cuenta el sistema conformado por los módulos de transferencia de energía, el regulador de voltaje y la posición del dron al aterrizar sobre la base. Estos son:

1. Voltaje de entrada: 12V - 20V.
2. Potencia de salida: La máxima posible.
3. Voltaje de salida: 11V - 18V.
4. Distancia de acople: 0.5cm.

Módulo transmisor

Es el módulo encargado de convertir el voltaje de DC a AC para lograr un sistema de acople inductivo con la placa receptora.
Termodron II realizó las elecciones de componentes para el módulo transmisor mediante el uso del diseño externo, simulaciones realizadas en el software TINA-TI, propiedad de Texas Instruments. Para la construcción utilizó el diseño original, mandó a imprimir la placa PCB y soldó los componentes a mano.
En la figura 3.4 se presenta el layout usado y la placa terminada.

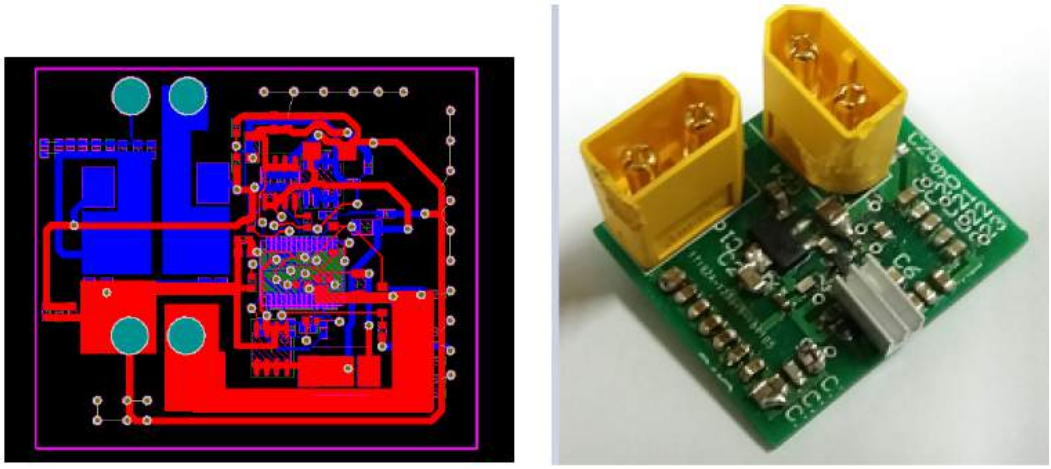

Figura 3.4: Modulo TX implementado por Termodron II. Imagen obtenida de [10].

Modulo receptor

El módulo receptor fue diseñado desde cero por Termodron II, tanto el esquemático como el layout fueron realizados en el programa Altium Designer.
El módulo cuenta con un circuito tanque para el acople inductivo y un puente rectificador de onda completa con un condensador de filtrado para obtener el voltaje DC.
Utilizaron los diodos que generen la menor caída de voltaje posible y soportaran una corriente directa de 8A.
Se elije el condensador de filtrado de forma de obtener un voltaje de ripple lo menor posible.

En la figura 3.5 se presenta el layout y el módulo implementado por Termodron II.

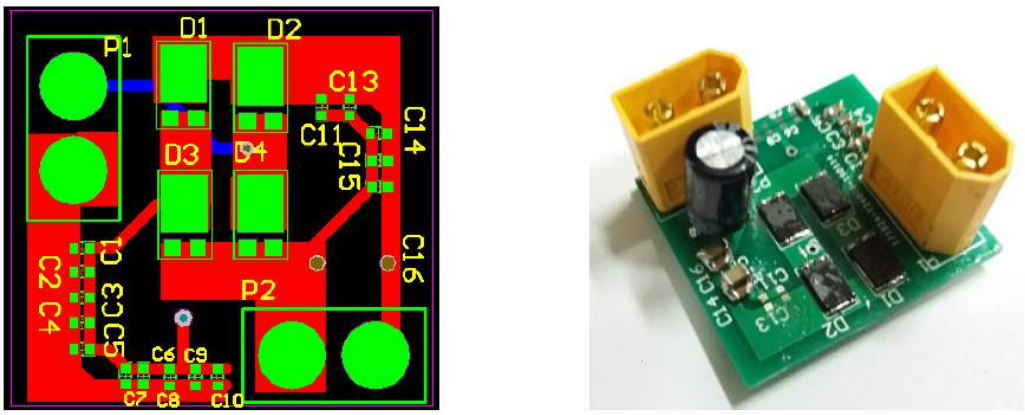

Figura 3.5: Modulo R X implementado por Termodron II. Imagen obtenida de [10].

Bobinas

Las bobinas cumplen la función del acople inductivo entre los módulos receptor y transmisor, un buen diseño de las mismas resulta vital para lograr un buen rendimiento en el sistema.
El diseño implementado por Termodron II busca minimizar las pérdidas de energía en las mismas, para ello buscan minimizar el efecto producido por los siguientes puntos:

1. Acople y alineación.
2. Efecto skin y de proximidad.
3. Patrón del campo magnético.

No se entrará en detalles sobre los puntos antes mencionados, los mismos se pueden encontrar en [10], simplemente se menciona que estos efectos se reducen mediante la implementación de un sistema de acople en la base, la elección del material adecuado para las bobinas y el uso de materiales magnéticos llamados magnetic shields los cuales son de utilidad para influir en la dirección del campo magnético.

3.2.4.2. Módulo Regulador

La elección del regulador se tomó considerando dos puntos importantes del sistema, por un lado el voltaje del sistema es dependiente de la corriente que circule por el mismo y dado que el cargador tiene un rango de voltaje de entrada acotada, el regulador debe ser capaz de entregar un voltaje constante en todo momento, y por otro lado Termodron II consideró la posibilidad de colocar dos o más módulos de recarga en serie, con lo cual seria necesario un sistema para independizarlos sin que se generen efectos como los de la impedancia vista.
Se buscan los reguladores de mayor eficiencia posible, con las siguientes características:

1. $V_{in}(V) = 0$ - 30
2. $V_{out}(V) = 12$ - 18
3. $I_{in}(A) = 0$ - 1,5
4. $I_{out}(A) = 0$ - 1,2

Con las características mencionadas, Termodron II realizó una prueba de eficiencia y de potencia de salida entre dos reguladores del mercado, eligiendo por el regulador XL4015.

3.2.4.3. Módulo Cargador

A la hora de realizar la elección del cargador, Termodron II eligió el cargador comercial con las siguientes consideraciones:

1. Cargadores que implementen el algoritmo de carga CC/CV junto a una recarga balanceada.
2. Cargadores automáticos, que comiencen a cargar cuando se les entrega energía.

El cargador utilizado fue el Tenergy RKI-1122.

3.3. Posibles mejoras sugeridas

Termodron II plantea las siguientes mejoras para una siguiente versión del dron, las cuales se enumeran a continuación:

1. Agregado de una segunda cámara con el fin de utilizarla para el aterrizaje y para poder realizar vídeos del vuelo.
2. Optimización de los componentes y peso del dron.
3. Re-diseño de la base.
4. Mejora de la comunicación Dron-Base.
5. Mejoras en los sensores distancia.
6. Mejoras en el sistema de recarga.
7. Implementación del recorrido de un área.
8. Paneles solares en la base

Estas sugerencias dadas por Termodron II fueron consideradas a la hora de planificar el alcance y el contenido de Termodron III, y varias de las mismas fueron implementadas.

Capitulo 4

Hardware de Termodron III

4.1. Elección de Componentes

Se detallan a continuación los componentes elegidos para el sistema, justificando su elección y realizando una comparación con los otros componentes considerados. En algunos casos se realizaron pruebas para poder decidir qué componente elegir para una cierta función, las mismas se encuentran en el capítulo 9.

Para el dron se utilizaron los siguientes componentes:

- Controlador de Vuelo: Pixhawk 2.1 (Pixhawk Cube)
- Computadora a bordo principal: Odroid-XU4
- Computadora a bordo secundaria: Arduino Nano
- Motores: Emax MT3510
- Electronic Speed Controller (ESC): Emax Simon Series 30A
- Hélices: Hockus Propeller 1447
- Batería: HRB 3S 10000mah 11.1V 25C.
- Cámara Térmica: FLIR Lepton 3.5
- Cámara estéreo: Intel RealSense D415
- Gimbal: STorM32 3-Axis Gimbal
- Sensores de Distancia: MB1361 XL-MaxSonar-AEL1
- Armazón del Dron (frame): Lji ZD550

- Receptor GNSS diferencial: Here 2 GNSS
- Radio telemetría: 3DR 915MHz 100mW
- Receptor Radio: Turnigy 9x 2,4GHz
- Encoder PPM: FPVDrone 8CH PPM Encoder
- Módulo LTE: módem USB Huawei E353 HSPA+
- Distribuidor de energía: Hobbyking Power Distribution mini
- Battery Eliminator Circuit (BEC): Pixhawk 2.1 BEC
- Regulador de voltaje: Pololu 9A 5V DC-DC step down converter
- Cargador: Turnigy 12v 2-3S Basic Balance Charger

4.1.1. Frame - Lji ZD550

El frame utilizado es el LJI ZD550, compuesto mayormente por fibra de carbono y algunas partes de plástico. Su diagonal es de 550 mm, posee soporte para 4 motores, y su peso es de aproximadamente 400 g. Se decide utilizar el mismo modelo de frame que Termodron II, porque se verifica que el peso del dron con los componentes nuevos no se aleja mucho del peso de Termodron II (se agregan aproximadamente 400 g) y porque se tiene una familiaridad con el modelo luego del período de aprendizaje y vuelos de prueba con Termodron II.

En la figura 4.1 se muestra el modelo utilizado.

Figura 4.1: Frame LJI ZD550. Imagen obtenida de **Amazon**.

4.1.2. Controlador de Vuelo - Pixhawk 2.1

El controlador de vuelo es una pieza fundamental del dron, encargado de recibir y procesar los datos de los sensores que miden el estado del dron, y a partir de estos, comandar los motores. Los controladores de vuelo cuentan con procesador, memoria RAM, memoria Flash, sensores (giroscopio, acelerómetro, magnetómetro, barómetro, y en algunos casos termómetro), puertos de comunicación serial (I2C, SPI, UART), conversor analógico-digital y entrada para conectividad con GPS.

Para la elección del controlador de vuelo se toman en cuenta todas estas cualidades mencionadas anteriormente y se comparan tres modelos distintos, Pixhawk 1, Pixhawk 2.1 (también llamado Pixhawk Cube) y Pixhawk 4. En el cuadro 4.1 se presentan las características principales de las distintas versiones del controlador de vuelo.

Característica - Modelo	Pixhawk 1	Pixhawk 2.1/Cube	Pixhawk 4
Procesador	ARM Cortex M4	ARM Cortex M4F	ARM Cortex M7
Velocidad del Procesador (MHz)	168	168	216
Memoria RAM (KB)	256 SRAM	256 SRAM	512 SRAM
Memoria Flash (MB)	2	2	2
Entradas Analógicas	Si (ADC)	Si (ADC)	Si (ADC)
Pines I/O	8	14	16
I2C	1	2	3
SPI	1	1	4
CAN	1	2	2
UART	4	5	5
Voltaje Alim. (V)	5	5	5
Giroscopio	1	1	2
Acelerómetro	1	3	2
Magnetómetro	1	2	1
Barómetro	1	1	1
Dimensiones (mm × mm × mm)	81.5 × 50 × 15.5	85 × 10 × 35	84 × 44 × 12
Peso (g)	38	200	15.8
Precio (US$)	120-200	238	178

Cuadro 4.1: Cuadro comparativo de los distintos modelos del controlador de vuelo.

Algunas observaciones acerca de los controladores: en cuanto al Pixhawk 1, éste es el utilizado por Termodron I y II, se ha estudiado su funcionamiento y se ha logrado ejecutar vuelos con el mismo. Aún así, como se puede observar en el cuadro 4.1, sus prestaciones son inferiores comparadas con las nuevas ofertas. En cuanto al Pixhawk 2.1, éste implementa mejoras significativas en el hardware en comparación con el Pixhawk 1, y, en algunos

aspectos, mejoras sobre Pixhawk 4, ya que el Pixhawk 2.1 presenta triple redundancia en las IMU (unidad de medida inercial), lo que puede ser una mejora significativa sobre el control del dron. Del Pixhawk 4 se destaca que presenta mejoras en cuanto al hardware respecto a ambas versiones (mejor procesador, RAM y flash) pero al ser un dispositivo tan reciente (año 2018) no existe tanto soporte online comparado con dispositivos mucho más usados en el ambiente de los vehículos aéreos radiocontrolados como lo son el Pixhawk 1 y 2.1.

Otro aspecto importante a la hora de tomar una decisión sobre la compra del controlador es el firmware a utilizar, ya que los controladores de la familia Pixhawk son compatibles con dos distintos: Ardupilot y PX4, siendo PX4 utilizada por Termodron I y luego cambiado a Ardupilot por Termodron II. El desarrollador del Pixhawk 2.1 recomienda su uso con Ardupilot mientras que para el Pixhawk 4 se recomienda PX4, esto no implica que no se pueda utilizar el firmware no recomendado, pero en cuanto al soporte puede marcar una diferencia. Además, para el firmware de Ardupilot existe la biblioteca Dronekit-Python [16] la cual permite comunicarse con el controlador de vuelo mediante el lenguaje de programación Python. Dado que el equipo de Termodron II utilizó esta biblioteca para el código de la GCS[1] en su proyecto y la familiaridad con este proyecto durante el período de pruebas iniciales, se decide utilizar Ardupilot y Pixhawk 2.1 como controlador de vuelo.

En la figura 4.2 se muestra el controlador de vuelo utilizado.

Figura 4.2: Controlador de vuelo Pixhawk 2.1 o Pixhawk Cube. Imagen obtenida de **Robot Shop**.

4.1.3. Computadora a bordo

Otro de los principales cambios a realizar se encuentra en la computadora a bordo (On Board Computer). Se cambia el Arduino Due que realizaba el procesamiento a bordo del dron por un sistema que consiste en una SBC[2]: Odroid XU4, y un

[1]Ground Control Station (Estación de Control Terrestre): como Termodron II le llamó a la base en su proyecto.

[2]Single Board Computer; computadora en una sóla placa. Designación usual para este tipo de placas.

microcontrolador: Arduino Nano. Se decide realizar este cambio porque, siendo el Arduino Due un microcontrolador, sus prestaciones son una limitante para el nivel de procesamiento que se quiere realizar; siendo éste aún mayor que en los proyectos anteriores, dada la introducción de una nueva cámara a bordo, como se detallará más adelante.

La Odroid XU4 se encarga de las tareas de mayor consumo de procesamiento, como ser la comunicación con el dron, el control del movimiento del mismo durante las misiones así como también el relevamiento del estado del dron, y el reporte hacia el usuario mediante Amazon Web Services. También se encarga del manejo del gimbal y de ambas cámaras, así como el procesamiento de las imágenes de las mismas. Por otro lado, el Arduino Nano es el encargado de actuar en casos de fallas en la Odroid XU4, realizando el manejo de algunos sensores ultrasónicos de distancia, y un posible reinicio de la computadora principal en caso de fallas.

Computadora a Bordo - Odroid XU4

Para la selección de la computadora principal del dron se hizo un cuadro comparativo con la gran mayoría de las SBC disponibles, comparándose todas sus prestaciones, haciendo énfasis en la memoria RAM, procesadores, conectividad y soporte. De este cuadro se seleccionaron cuatro candidatas: Aaeon Up Board, Nvidia Jetson Nano, Odroid XU4 y Raspberry Pi 3 B+. En el cuadro 4.2 se presentan sus características principales.

Característica \ Modelo	Aaeon Up Board	Nvidia Jetson Nano	Odroid-XU4	Raspberry Pi 3
CPU	Intel Atom x5-Z8350@1.44GHz	AMR Cortex-A57 @1.43GHz	AMR Cortex-A15 AMR Cortex-A7@2GHz	AMR Cortex-A53 @1.4GHz
GPU	Intel HD 400 500MHz	Nvidia Maxwell with 128xCUDA cores@998Mhz	Mali-T628 MP6 @600MHz	Broadcom BCM2837B0.
RAM (GB)	4	4	2	1
Almacenamiento	32GB	16GB	Micro SD	Micro SD
Conectividad	1x Gb Ethernet RJ-45 40-pin GPIO	IEEE 802.11 y 40-pin GPIO	IEEE 802.11 y 30-pin GPIO	IEEE 802.11 y 40-pin GPIO
Dimensiones (mm × mm × mm)	85.6×56.5×16.7	100×47×29	83×58×20	82×56×19.5
Soporte	Medio	Bajo	Medio	Alto
Peso (g)	50	47.9	60	50
Precio (US$)	89	99	84	35

Cuadro 4.2: Cuadro comparativo de las distintas computadoras a bordo.

En el cuadro 4.2 se menciona el soporte de las SBC, (este parámetro es subjetivo) punto clave en la elección de la misma. Se las clasificó como Bajo, Medio y Alto en función de la cantidad de información disponible en la web y de la actividad en los foros disponibles.

De las cuatro SBC mostradas, se destacan dos: la Raspberry Pi 3, que cuenta con la ventaja de tener un alto soporte de la comunidad, y la posibilidad de incorporar un módulo de dos cámaras con el cual se podría implementar visión estereoscópica, pero presenta la desventaja de tener menores prestaciones que las demás. Por otro lado, la Odroid XU4 tiene el mismo factor de forma que la Raspberry Pi 3 y mayor poder de procesamiento.

Ambas son compatibles con las cámaras de profundidad Intel RealSense, las cuales son candidatas para su utilización en el dron. Finalmente, habiendo ponderado todos estos factores, se optó por la Odroid XU4, la cual se presenta en la figura 4.3

Figura 4.3: Computadora a bordo principal, Odroid XU4. Imagen obtenida de **Odroid Wiki**.

Microcontrolador - Arduino Nano

Se optó por un Arduino Nano como computadora secundaria ya que es un microcontrolador de tamaño compacto, barato, de bajo consumo, y con gran soporte online. Su tarea será el manejo de sensores de distancia y de reiniciar la computadora a bordo principal en caso que esta falle.

En el cuadro 4.3 se presentan las especificaciones del Arduino Nano, y en la figura 4.4 se observa la placa.

Característica \ Modelo	Arduino Nano
CPU	Atmel ATmega328
Memoria Flash (KB)	32
RAM (kB)	2
Conectividad	6 PWM, 8 Analógicas
Soporte	Alto
Dimensiones (mm× mm)	45×18
Peso (g)	5
Precio (US$)	12 (pack de 3)

Cuadro 4.3: Especificaciones del Arduino Nano.

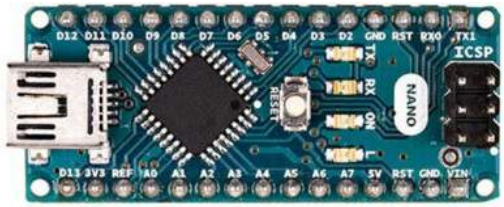

Figura 4.4: Microcontrolador de respaldo, Arduino Nano. Imagen obtenida de **Arduino**.

4.1.4. Motores

Para la elección de los motores se tienen en cuenta distintos parámetros, tanto propios del motor como del dron en su conjunto. El primer parámetro a tener en cuenta es la máxima corriente que el motor puede consumir, ya que esto determinará el tiempo de vuelo del dron, y también la corriente que deban soportar los ESC elegidos.

El segundo parámetro es el empuje del motor, el cual se relaciona también con la corriente que consume y el peso del dron. Se busca el motor con menor consumo de corriente para el empuje en operación (donde el empuje en operación es igual al peso del dron), y además, se recomienda tener una relación empuje-peso igual a 2 para lograr un control fino sobre el movimiento del dron, esto impone otro requisito sobre el empuje deseado de los motores.

Una restricción que se debe tener en cuenta es el voltaje de alimentación de los motores, el cual está dado por el voltaje de la batería a utilizar. En este caso, el voltaje es de 11.1 V, pues se usa una batería de 3 celdas (3.7 V por celda), y se mantiene esta cantidad de celdas debido a que el sistema de recarga inalámbrica diseñado e implementado por Termodron II funciona con este tipo de baterías [10].

Termodron II mejoró el TWR [3] de Termodron I, utilizando motores de mayor empuje y un consumo razonable [10]. En busca de seguir mejorando esta relación se necesitan motores de mayor empuje, siempre sin perder de vista el consumo máximo y el voltaje de alimentación del mismo.

En el cuadro 4.4 se presenta una pre-selección de tres motores analizados. Se debe mencionar que el motor Emax GT2215/12 es el utilizado por Termodron II.

[3] Thrust to Weight Ratio: relación empuje-peso.

Característica \ Modelo	Emax GT2820	Emax MT3510	Emax GT2215/12
Cantidad de celdas	3-4	3-4	3
RPM/V	850	600	905
RPM	7600	4840	7450
$I_{MAX}(A)$	26	15	15
Thrust (g)	1650	1210	1000
Hélices recomendadas	12×06	14×4.7	10×4.7
Dimensiones (mm × mm)	46.5×35	41.5×31.8	36.5×28.5
Peso (g)	140	102	70
Precio (US$)	23.23	32	15.86

Cuadro 4.4: Especificaciones de los motores de la pre-selección.

Al tener la pre-selección realizada se decidió adquirir un ejemplar de cada uno y realizar pruebas con el fin de realizar una mejor decisión. El procedimiento realizado y los resultados se pueden ver en el capítulo 9, sección 9.1, con estos resultados se decidió por el motor Emax MT3510, el cual se muestra en la figura 4.5.

Figura 4.5: Motor Emax MT3510. Imagen obtenida de **Emax Model**.

4.1.5. ESC

Los controladores de los motores o ESC por su sigla en inglés (Electronic Speed Controller) quedan limitados por la corriente máxima que los motores consumen. Se eligen los ESC Emax Simon Series 30A, debido a que pueden tolerar casi el doble que la máxima corriente que consumen los motores, dejando bastante margen de seguridad. Su especificación se presenta en el cuadro 4.5 y se puede observar en la figura 4.6.

Característica \ Modelo	Emax Simon Series 30A
Cantidad de celdas	2-3
$I_C(A)$	30
$I_{MAX}(A)$	40
Dimensiones (mm × mm ×mm)	55×28×7
Peso (g)	28
Precio (US$)	9.24

Cuadro 4.5: Especificaciones de los ESC Emax Simon.

Figura 4.6: ESC Emax Simon Series 30A. Imagen obtenida de **Amazon**.

4.1.6. Hélices

Las hélices se eligieron tomando en cuenta la recomendación del motor seleccionado, estas tienen las características presentadas en el cuadro 4.6.

Característica \ Modelo	Hockus Accessories
Material	Fibra de carbono
Dimensiones (inch × inch)	14×4.7
Peso (g)	20
Precio (US$)	20.19 (2 pares)

Cuadro 4.6: Especificaciones de las hélices seleccionadas.

En la figura 4.7 se presentan las hélices seleccionadas para el dron.

Figura 4.7: Hélices Hockus Accessories 14×4.7. Imagen obtenida de **Amazon**.

4.1.7. Batería - HRB 3S 10000mAh

Previo a la selección de la batería se analizó la batería utilizada por Termodron II. Esta, de marca HRB tiene una capacidad de $5000mAh$, es de 3 celdas y 50C, más información de la misma se puede observar en el cuadro 4.7.

Para la elección de la batería se buscó mejorar la capacidad de la misma, de forma de aumentar el tiempo de vuelo del dron, y como contingencia en caso de usar motores que consumieran más que los utilizados por Termodron II. También se buscó mantener la cantidad de celdas, ya que como se mencionó anteriormente, el sistema de recarga inalámbrica es dependiente de una batería de 3 celdas y se decidió no realizarle modificaciones a dicho sistema. Se busca también mantener la máxima corriente de carga/descarga admisible de la batería, la cual era de 250 A [4].

Se realizó una comparación entre varios modelos distintos. Sus parámetros comparados se pueden observar en el cuadro 4.7.

Característica \ Modelo	Power Hobby	HRB	Ovonic	HRB	HRB
Tipo	Lipo	Lipo	Lipo	Lipo	Lipo
Voltaje de Celda (V)	3.7	3.7	3.7	3.7	3.7
Cantidad de Celdas	3	3	3	3	3
Tasa de Descarga (C)	75	25	50	50	50
Capacidad (mAh)	9000	10000	8000	6000	5000
Dimensiones (mm×mm×mm)	155×44×46	165×64×32	130×40×31	138×46.8×39	155×48×24
Peso (g)	617	640	424	455	376
Precio (US$)	100	84	100 (2 unidades)	50	40

Cuadro 4.7: Especificaciones de las baterías.

[4]Esta corriente se puede calcular como el producto de la capacidad de la batería en mAh por el número C de la batería. Para la utilizada por Termodron II se tiene $5000mAh \times 50C = 250A$

Se decidió por la batería HRB de capacidad 10000mAh, 25C y 3 celdas, por tener el doble de capacidad y ser de la misma marca que la utilizada por Termodron II. La misma se puede observar en la figura 4.8.

Figura 4.8: Bateria HRB 3S, 25C y 10000mAh. Imagen obtenida de **Amazon**.

4.1.8. Placa de distribución de energía

Luego de seleccionados los motores se elige la placa de distribución de energía, la cual será la encargada de repartir el voltaje y corriente de alimentación desde la batería a los motores y los demás componentes como ser las computadoras a bordo, reguladores, y gimbal.

Esta placa debe ser capaz de suministrar la corriente máxima consumida por los motores que, como ya se mencionó, es de 15 A máximo, por lo que se elige la Power Distribution Board Lite de HobbyKing, la cual soporta hasta 20 A de corriente por salida, permite el soldado de varios cables de salida para alimentar otros componentes, y es igual al utilizado por los proyectos anteriores. En el cuadro 4.8 se pueden observar sus características.

Característica \ Modelo	Hobby King
Cantidad de salidas	4
Máxima Corriente (por salida)	20 A
Peso (g)	19.3
Precio (US$)	2.04

Cuadro 4.8: Especificaciones de la placa de distribución.

En la figura 4.9 se presenta la placa de distribución de energía.

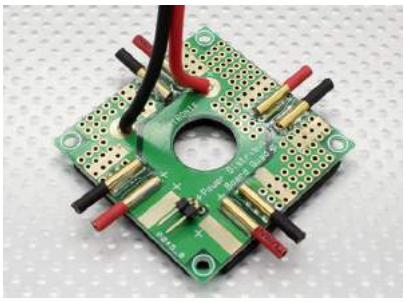

Figura 4.9: Hobby King Power Distribution Board Lite. Imagen obtenida de **Amazon**.

4.1.9. Reguladores de Tensión

Dados los diferentes niveles de tensión requeridos por los diferentes componentes del dron, se necesitan reguladores de voltaje para convertir de los 11.1 V de la batería a 5 V para los circuitos digitales y computadoras a bordo. Para la elección del regulador de voltaje se debe tener en cuenta que la computadora Odroid XU4 se alimenta con 5 V y puede llegar a consumir 5 A, por lo que se seleccionaron dos reguladores distintos, los cuales fueron probados para ver cuál puede proveer la corriente necesaria para la computadora cuando está sometida a grandes tareas de procesamiento. Se halló que ambos pueden proporcionar suficiente corriente, por lo que se decidió utilizar el regulador Pololu D24V90F5, ya que es el que tiene mayor corriente máxima, para tener un margen mayor en caso de picos de corriente. En el cuadro 4.9 se muestran las características de los dos reguladores probados.

Característica \ Modelo	Hiletgo DC-DC Converter XL4015	Pololu D24V90F5
Voltaje de entrada mínimo (V)	4	5
Voltaje de entrada máximo (V)	38	38
Voltaje de salida mínimo (V)	1.25	5
Voltaje de salida máximo (V)	36	5
Corriente de salida máximo (A)	5	9
Corriente de salida máximo recomendado (A)	4.5	8
Eficiencia	96 %	80 % - 95 %
Dimensiones (mm × mm × mm)	54 × 23 × 18	40 × 20.3 × 7.6
Peso (g)	18	4.8
Precio (US$)	8	29.99

Cuadro 4.9: Especificaciones del convertidor DC-DC.

En la figura A.40 se presentan los reguladores de voltajes.

Figura 4.10: Regulador de voltaje Hiletgo. Imagen obtenida de **Amazon**

Figura 4.11: Regulador de voltaje Pololu. Imagen obtenida de **Amazon**

4.1.10. BEC

Se utiliza el BEC[5] que viene incluido con el controlador de vuelo Pixhawk 2.1, el cual es capaz de transformar de 12V a 5V, sensando el voltaje y la corriente. Este es capaz de pasar una corriente máxima de hasta 90A, tiene conectores XT60 para conectar la batería y luego al distribuidor, y cable de alimentación compatible con el controlador de vuelo. En la figura 4.12 se observa el mismo.

Figura 4.12: BEC. Imagen obtenidas de **Amazon**.

4.1.11. Cámara Térmica - FLIR Lepton 3.5

Cámara Térmica

[5]BEC: Battery Eliminator Circuit. Utilizado para alimentar el controlador de vuelo a partir de la batería del dron.

Dadas las pruebas realizadas por Termodron II, y su recomendación de utilizar una cámara térmica con radiometría se decide cambiar la cámara utilizada en los dos proyectos anteriores [10] . A la hora de elegir la cámara térmica se analizaron distintas opciones, de las cuales se pre-seleccionaron tres modelos, se pueden ver en el cuadro 4.10.

Característica \ Modelo	FLIR Lepton 1.5	FLIR Lepton 3	FLIR Lepton 3.5
Resolución (px × px)	80×60	160×120	160×120
Rango de temperatura óptimo	-10°C a 140°C	-10°C a 140°C	-10°C a 140°C
Ángulo de apertura horizontal	50°	57°	57°
Radiómetria	NO	NO	SI
Distorsión	<8 %	<13 %	<13 %
Protocolo de comunicación	I2C	I2C	I2C
Dimensiones (mm × mm 1× mm)	9.67×8.47×5.62	12.70×11.50×7.14	12.70×11.50×7.14
Peso (g)	0.63	0.91	0.91
Precio (US$)	175	239	259

Cuadro 4.10: Especificaciones de las cámaras térmicas.

Se optó por la cámara FLIR Lepton 3.5 ya que cuenta con una resolución de 160×120 (4 veces mejor que la FLIR Lepton 1.5, utilizada por Termodron I y II). Además esta cuenta con radiometría, lo cual la diferencia de la FLIR Lepton 3.
En la figura 4.13 se muestra la cámara térmica seleccionada.

Figura 4.13: Cámara térmica FLIR Lepton 3.5. Imagen obtenida de **GroupGets**.

Breakout Board - Smart I/O Module

La función de la breakout board es generar la interfaz de conexión entre la cámara térmica y una computadora, en nuestro caso, la computadora a bordo. Se decidió por la breakout board PureThermal 2 Smart I/O Module ya que la breakout board 1.4 utilizada por Termodron I y II no es compatible con la cámara FLIR Lepton 3.5 elegida. Además, presenta la ventaja de tener conectividad por USB (a diferencia de la breakout 1.4 que era por SPI y requería la conexión de 9 cables), bastante documentación y soporte por parte

del equipo que la creó, y un diseño compacto, de dimensiones 30×18mm, lo cual la hace ideal para el uso en el dron. En la figura 4.14 se presenta la misma.

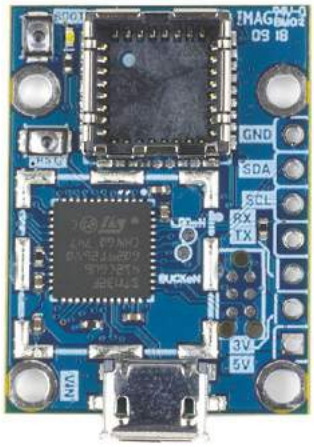

Figura 4.14: Breakout board PureThermal 2 Smart I/O Module. Imagen obtenida de **Digikey**.

4.1.12. Cámara estéreo - Intel RealSense D415

Una de las innovaciones del proyecto Termodron III es la inclusión de una cámara estereoscópica. Esta será la encargada de realizar la detección de obstáculos, y asistirá en el aterrizaje de precisión.

A la hora de seleccionar la cámara a utilizar se buscaron distintas alternativas, mediante investigación sobre el tema de visión estéreo, y considerando aspectos de las mismas como ser: distancia de detección, portabilidad, compatibilidad con las computadoras a bordo consideradas, y soporte online, se logró llegar a dos posibles candidatas. Por un lado la Intel RealSense D415 y por el otro el módulo Stereo Pi. Las principales características de las mismas se detallan en el cuadro 4.11.

Característica \ Modelo	Intel RealSense D415	Stereo Pi Deluxe Kit
Rango Máximo (m)	10	-
Campo de visión	65°x40°x72°	-
Resolución de profundidad	1280x720 up to 90 fps	1280x480 up to 20 fps
Conector	USB-C	15-pin CSI-2 camera connectors
Dimensiones (mm×mm×mm)	99×20×23	90×40×23
Peso (g)	72	-
Precio (US$)	179	199

Cuadro 4.11: Especificaciones de las cámaras Intel RealSense y Stereo Pi.

La cámara Intel RealSense D415 está diseñada para aplicaciones como escaneo 3D y generación de mapas de profundidad. Incluye una cámara de espectro visible, dos

cámaras estéreo de infrarrojo activo, un proyector para obtener información de textura en superficies lisas, un procesador de profundidad D400 para procesar la información de las dos cámaras en tiempo real, todo dentro de una cubierta de aluminio y con un factor de forma pequeño, como se ve en el cuadro 4.11 [12]. Contrario a otras cámaras estéreo basadas en infrarrojo, las cámaras de la familia RealSense D400 funcionan correctamente tanto en luz interior como exterior, lo cual es clave para esta aplicación. Se tiene además un SDK [6] de Intel para el fácil uso y calibración de las mismas, así como librerías para Python de uso libre, y foros de soporte online.

El módulo Stereo Pi tiene como ventajas el hecho de ser compatible con una de las placas de desarrollo más populares en la actualidad, así como su versatilidad para el montaje, dado que son básicamente dos cámaras con cable de cinta en una placa de desarrollo adecuada. Sin embargo, dado que surgió como una campaña de *kickstarter* al comienzo del año 2019, que su soporte hasta el momento es limitado, y que hay una cierta falta de especificaciones (en campo de visión, peso y rango máximo de detección) se decide descartar este modelo frente a la cámara Intel RealSense D415.

Figura 4.15: Cámara Estereoscópica Intel RealSense D415. Imagen obtenida de **Intel**.

4.1.13. Sensor de distancia - MB1361 XL-MaxSonar

Como respaldo a la utilización de la cámara Intel RealSense para la evasión de obstáculos se cuenta con sensores de distancia por ultrasonido. Para la elección de los mismos, se buscó que tuvieran un rango de detección mayor a 4 m, buscando ampliar el rango de los que se encontraban en Termodron II (2 m) [10]. El modelo MBL1361 de XL-MaxSonar fue elegido por tener una distancia de detección máxima de 10 m, y por ser compatible con los controladores de vuelo de la familia Pixhawk. Sus características detalladas se pueden observar en el cuadro 4.12. En la figura 4.16 se presenta el sensor de distancia.

[6] Software Development Kit: kit de desarrollo de software.

Característica \ Modelo	XL-MaxSonar MBL1361
Voltaje nominal (V)	3.2 - 5.5
Máxima Distancia (m)	10
Resolución (cm)	2
Frecuencia de trabajo (KHz)	42
Dimensiones ($mm \times mm \times mm$)	19,9 × 22,1 × 25,11
Peso (g)	5.9
Precio (US$)	58

Cuadro 4.12: Especificaciones de los sensores de distancia por ultrasonido.

Figura 4.16: Sensor de distancia por ultrasonido XL-MaxSonar MBL1361. Imagen obtenida de **MaxBotix**.

4.1.14. GNSS - Here 2

La elección del módulo de navegación satelital se vio ayudada por la elección del controlador de vuelo, ya que, una vez que se decidió por el Pixhawk 2.1, se observó que éste recomendaba su uso con el GNSS [7] Here 2, por lo que se investigó sobre dicho módulo de navegación y se llegó a la conclusión de que era una buena opción dado su precio, soporte, compatibilidad con el controlador de vuelo, y la posibilidad de implementar las técnicas de corrección de GPS utilizadas por Termodron II. Se utilizará GPS como sistema de navegación, ya que fue el utilizado en las dos iteraciones anteriores de Termodron, y es el más predominante en nuestro país. Las características del módulo se presentan en el cuadro 4.13.

[7]Global Navigation Satellite System: sistema de navegación global satelital. Puede ser uno de los siguientes: GPS, GLONASS, BeiDou, Galileo.

Característica \ Modelo	Here 2 GNSS
Procesador	STM32F302
ROM	Flash (NEO-M8N)
Giroscopio	ICM20948
Acelerómetro	ICM20948
Brújula	ICM20948
Barómetro	MS5611
Protocolos	CAN, UART, I2C
Temperatura de trabajo ($°C$)	-40 a 85
Dimensiones (mm×mm×mm)	76×76×16.6
Peso (g)	49
Precio (US$)	95

Cuadro 4.13: Especificaciones del módulo GNSS Here 2.

En la figura 4.17 se presenta el GPS utilizado.

Figura 4.17: GNSS Here 2. Imagen obtenida de **Amazon**.

4.1.15. Modulo de telemetría

Como ya se mencionó, se decidió mantener el modulo de telemetría de 915MHz, de forma de tener redundancia en la comunicación con el controlador de vuelo. Las características principales de la antena están detalladas en el cuadro 4.14 y se puede observar la misma en la figura 4.18.

Característica \ Modelo	3DR Radio Telemetry Kit
Frecuencia (MHz)	915
Potencia de salida (mW)	100
Conector de la antena	RP-SMA
Interfaz	Estándar TTL UART
Peso (g)	5.8
Precio (US$)	25

Cuadro 4.14: Especificaciones de la antena de telemetría.

Figura 4.18: Antena de telemetría 3DR 915MHz. Imagen obtenida de **Amazon**.

4.1.16. Módem LTE

Otra de las innovaciones presentes en el proyecto es la incorporación de un módulo LTE en el dron, aprovechando la mejora de velocidad con respecto a 3G, lo que podría permitir el envío de más datos, fotos y hasta videos hacia el usuario y/o la base. Se nota también que esta tecnología es más rápida que la de radiofrecuencia, por lo que se mejora en este aspecto también, aunque, como ya se mencionó, la robustez de este último sistema es razón para mantenerlo como respaldo.

Se decide utilizar un módem Huawei E353 HSPA+ debido a su disponibilidad en plaza, y que la computadora a bordo corre un sistema operativo Linux, asegurando la compatibilidad del mismo con el módem.

En la tabla 4.15 se notan las características de este módulo y en la figura 4.19 se presenta el mismo.

Característica \ Modelo	Módulo LTE/3G USB
Voltaje de alimentación (V)	5 (USB)
Soporta tarjeta SIM	SI
Protocolos que soporta	SMS,MMS,TCP,UDP,HTTP,FTP,etc.
Velocidad de transmisión	Hasta 21.6Mbps
Dimensiones (mm×mm×mm)	89×27×12
Peso (g)	30
Precio ($)	3000

Cuadro 4.15: Especificaciones del módulo LTE

Figura 4.19: Módulo LTE. Imagen obtenida de **Amazon**.

4.1.17. Receptor de radio - Turnigy 9X

Termodron II cuenta con un control remoto Turnigy TGY9X para poder enviar comandos de emergencia (como ser "retornar al punto de despegue" o "aterrizar") al controlador de vuelo, mediante radiofrecuencia. El control remoto trabaja a 2.4GHz y tiene 9 canales de comunicación.

Se decide mantener el uso de la radio, adquiriendo un nuevo receptor con el fin de no quitar el que estaba presente en Termodron II. Las características del receptor son las descritas en el cuadro 4.16, y en la figura 4.20 se presenta el receptor.

Característica \ Modelo	Here 2 GNSS
Frecuencia de trabajo (GHz)	2.4
Cantidad de canales	9
Dimensiones (mm×mm×mm)	52×35×15
Peso (g)	18
Precio (US$)	15

Cuadro 4.16: Características del receptor radio Turnigy 9X.

Figura 4.20: Receptor radio Turnigy 9X. Imagen obtenida de **Amazon**.

4.1.18. Gimbal - STorM32

En la elección del gimbal se optó por mantener el mismo gimbal utilizado por Termodron II, el STorM32 BGC V1.3. Este cumplirá la función de mantener la cámara estéreo en una posición fija respecto al horizonte, y a la hora de aterrizar sera el encargado de girar la cámara con el fin de que esta esté posicionada correctamente para realizar el aterrizaje.

El gimbal contiene su propio controlador y sensores de medida inercial y se comunica con el controlador de vuelo mediante el protocolo UART con el fin de recibir órdenes. En la figura 4.21 se muestra el mismo.

Figura 4.21: Gimbal STorM 32. Imagen obtenida de **Kiubear**.

Cabe destacar que para la utilidad prevista del gimbal en el dron, no es necesario que posea los 3 motores, por lo cual se quita el motor de *yaw*, reduciendo así el peso del gimbal.

4.1.19. Sistema de recarga inalámbrica

El sistema de recarga inalámbrica fue una parte fundamental del proyecto Termodron II, y dada su gran utilidad para aumentar la autonomía del dron es que se decide mantener el sistema en Termodron III. Se decide dejar fuera del alcance de este proyecto la implementación de variantes o mejoras a dicho sistema de recarga, por lo que se implementará el mismo utilizado en Termodron II. Para tener una idea más detallada del funcionamiento de la recarga y de los componentes utilizados ver [10].

A continuación se detallan algunos de los componentes necesarios para poder implementar este sistema en el dron.

Cargador

Termodron II decidió utilizar un cargador Turnigy Basic Balance Charger con el fin de implementar el método de carga CC/CV[8] y una recarga balanceada [10]. El mismo es

[8]Constant Current/Constant Voltage: método de carga de baterías Li-Po: un tramo es a corriente

compatible con baterías de 2 celdas y 3 celdas, lo cual, como ya se mencionó, fue fijado en 3 celdas y se tomó como un parámetro fijo para la decisión de la batería a utilizar. Además, el cargador implementa la carga automática, lo cual es de vital importancia para que comience a cargar una vez que se le entrega energía. El cargador se presenta en la figura 4.22.

Figura 4.22: Cargador de 3 celdas Turnigy. Imagen obtenida de **Hobby King**.

Regulador de voltaje

Como es detallado por Termodron II, es necesario utilizar un regulador de voltaje de forma de proporcionar al cargador Turnigy un voltaje constante, ya que, debido a las distancias de acople entre las bobinas del sistema de recarga este voltaje no es constante [10]. El regulador utilizado es el XL4015 el cual se detalló en la sección 4.1.9.

4.2. Diagrama de conexión

En la figura 4.23 se presenta un diagrama de conexión de los componentes del hardware.

constante y luego se termina la carga a voltaje constante.

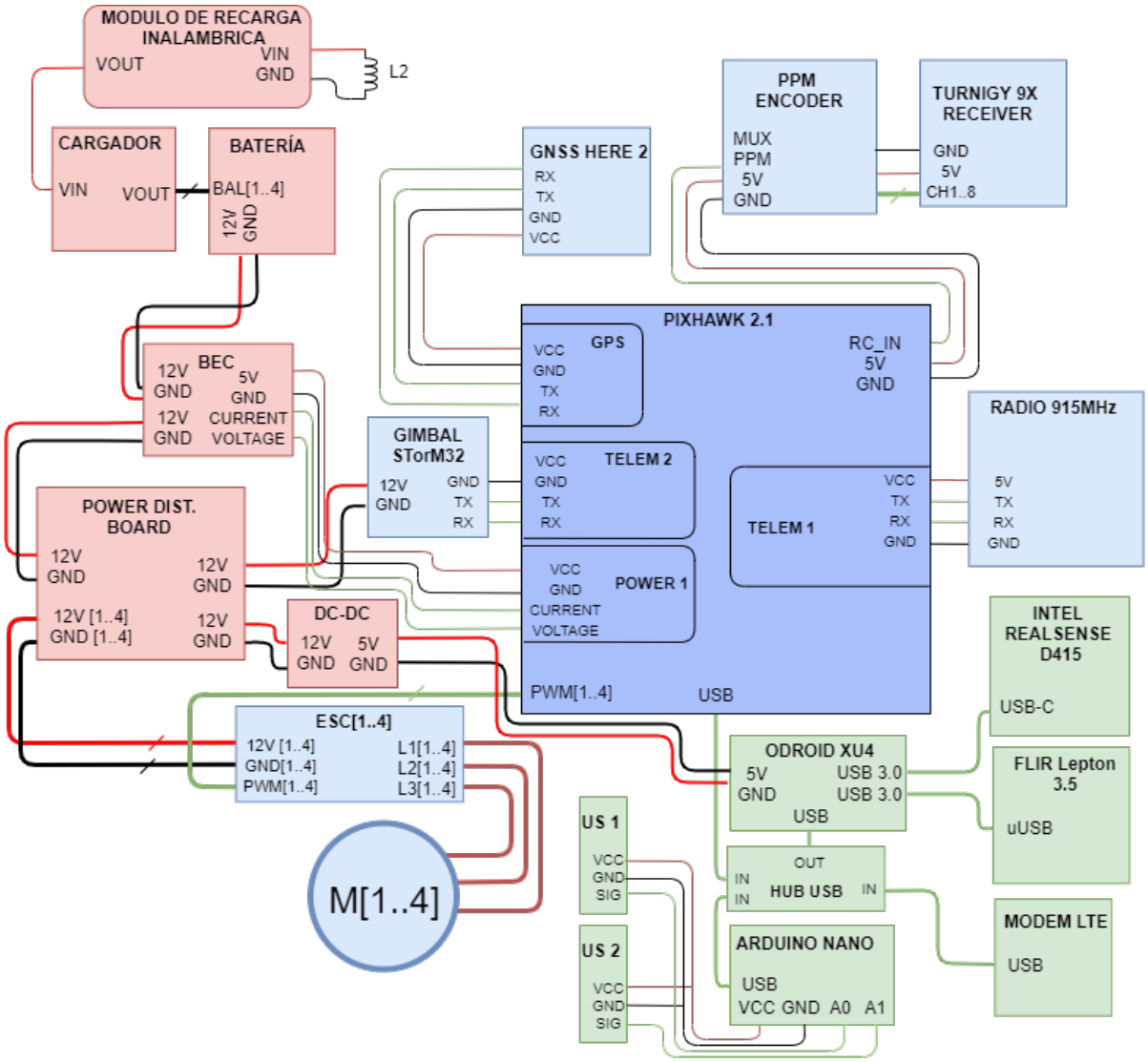

Figura 4.23: Diagrama de hardware de los componentes del dron.

Capitulo 5

Software de Termodron III

5.1. Lenguaje de programación

Se utilizó el lenguaje de programación Python [1], por múltiples razones: dada la existencia de la biblioteca Dronekit-Python, la cual permite la comunicación con un controlador de vuelo con el stack Ardupilot, mediante comandos MAVlink [16], además, fue el lenguaje utilizado por el grupo Termodron II para el código de la Estación de Monitoreo y Control [10], y también por su gran catálogo de bibliotecas útiles para el proyecto y de libre acceso.

5.1.1. Bibliotecas utilizadas

Se listan a continuación las bibliotecas más importantes utilizadas en el software, junto con una breve descripción.

- **Dronekit-Python:** encargada de la comunicación con el controlador de vuelo. Basada en comandos MAVlink [16].
- **pymavlink:** permite el envío de comandos MAVlink a un controlador de vuelo [25].
- **OpenCV:** contiene varias herramientas orientadas a *computer vision* en tiempo real [32].
- **AWS IoT SDK:** permite el envío de datos en tiempo real hacia los servidores de Amazon, y la posibilidad de obtener los mismos en tiempo real [19].
- **pyrealsense2:** versión para Python de la librería *librealsense*. Permite configurar y calibrar cámaras estéreo Intel RealSense y obtener información de distancia en cada pixel de la imagen [33].

[1] Se utilizó Python 3 para todo el código que corre en el dron, y para la mayoría de los que corren en la base, con la excepción del código de DGPS (implementado por Termodron II en Python 2), por razones de incompatibilidad al intentar pasarlo a Python 3

Además, se utilizan otras librerías de terceros, las cuales fueron modificadas para ser adaptadas a las necesidades de este proyecto, éstas son: *lib_aruco_pose* [21], *grid_map_lib* [25], y *NTRIP_client* [54].

Finalmente, se utilizan otras librerías de Python como ser: threading, multiprocessing, argparse, time, socket, JSON, queue, numpy, os, sys, path, math, matplotlib, email.

En la figura 5.1 se presenta un diagrama que muestra la jerarquía de los distintos *scripts* y librerías implementados para la entidad correspondiente al dron.

Se tiene un código principal llamado Dron, el cual importa las librerías: libMail, libAWS, libArduino, libThermal, libStereo, libDron. A su vez, se muestran los *threads* y *scripts* más relevantes de cada librería, pero no se listan todos los códigos de cada una para mantener la simplicidad del diagrama.

Dentro de libDron se encuentra *threadDron*, el cual, a su vez importa el *script*, *precLand*. Éstos se encargan de la realización de la misión (despegue, recorrido y evasión, fotos térmicas) y el aterrizaje de precisión. Dentro de libStereo se tiene *threadStereo* el cual implementa el algoritmo de detección de obstáculos y comunica los valores de medidas de distancia hacia *threadDron*; en libThermal se tiene *threadThermal* el cual se encarga de la detección de cuerpos calientes; en *libAWS* se encuentra *threadAWS* el cual se encarga de la comunicación con *threadDron* para obtener su estado y actualizarlo en el servidor de AWS. En libArduino se encuentra *threadArduino* el cual se encarga de comunicarse con *threadDron* para enviarle los datos de distancia de los sensores de ultrasonido.

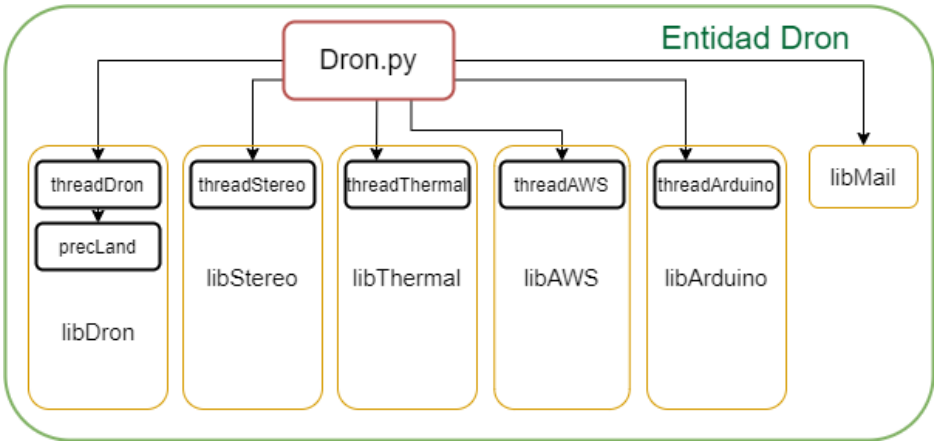

Figura 5.1: Jerarquía del software para la entidad Dron. En rojo se encuentra el *script* principal de la entidad; en amarillo las librerías utilizadas por éste, y en negro los *threads* o *scripts* más importantes implementados en dichas librerías. Las flechas indican cuál *script* invoca a cuál otro. Todos éstos son ejecutados en Python 3

En la figura 5.2 se presenta un diagrama que muestra la jerarquía de los distintos *scripts* y librerías implementadas para la entidad correspondiente a la base. Se compone de un código principal llamado Base, el cual importa las librerías: libMail y libAWS. A su vez, se muestran los *threads* de cada librería (no se muestran todas los *scripts* de cada

librería para mantener la simplicidad del diagrama). El *thread threadAWS* se encarga de obtener el reporte en tiempo real del dron por AWS, si se lo desea. En *libDGPS* se encuentran los dos *scripts* necesarios para el envío de mensajes de corrección diferencial de posición hacia el dron: *DGPS* y *threadDGPS*. Debe mencionarse que éstos últimos son ejecutados en Python 2, ya que no es posible la conversión hacia Python 3, y por eso, y su independencia del resto del código de la base es que se ejecutan por fuera del código principal.

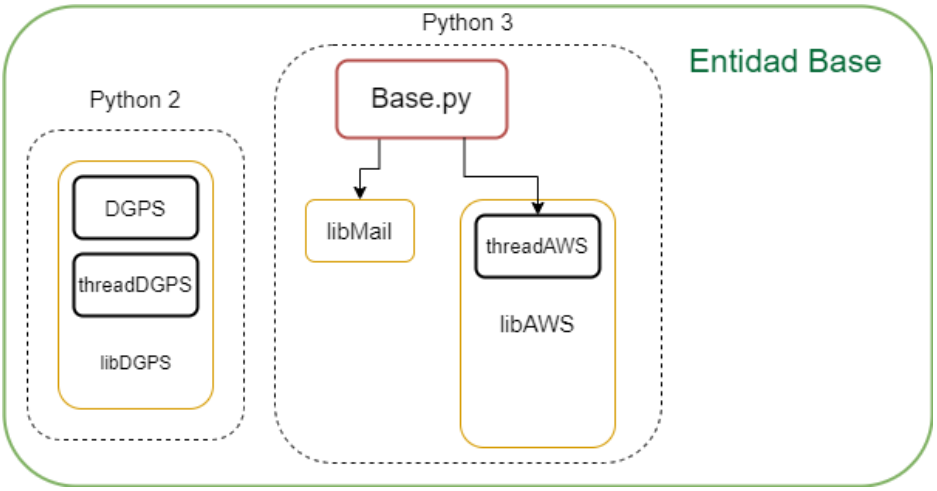

Figura 5.2: Jerarquía del software para la entidad Dron. En rojo se encuentra el *script* principal de la entidad; en amarillo las librerías utilizadas por éste, y en negro los *threads* o *scripts* más importantes implementados en dichas librerías. Las flechas indican cuál *script* invoca a cuál otro. En líneas punteadas se separan los códigos que se ejecutan con Python 2 (los correspondientes a la corrección de GPS diferencial) y los que se ejecutan en Python 3 (los demás *scripts* relevantes a la base).

5.1.2. Arquitectura de software

Para el diseño del software se tuvo en cuenta que se tiene una computadora en la base y una a bordo del dron, por lo que se escribió el código para cada una por separado, teniendo en cuenta las interacciones entre ambas y agregando la lógica de comunicación correspondiente. También se diseñó una interfaz gráfica, la cual puede utilizar el usuario desde su propia computadora para comunicarse con la base, preparar misiones, ordenar que se ejecuten las mismas, ver el estado del dron, y los logs y fotos luego de terminada la misión. En la sección 7 se entra en más detalle sobre cómo se comunican estas tres entidades.

Tanto para la base como para el dron se utiliza un diseño en máquinas de estado, las cuales se pueden ver en las figuras 5.3, 5.4. A continuación se explica el funcionamiento de las mismas.

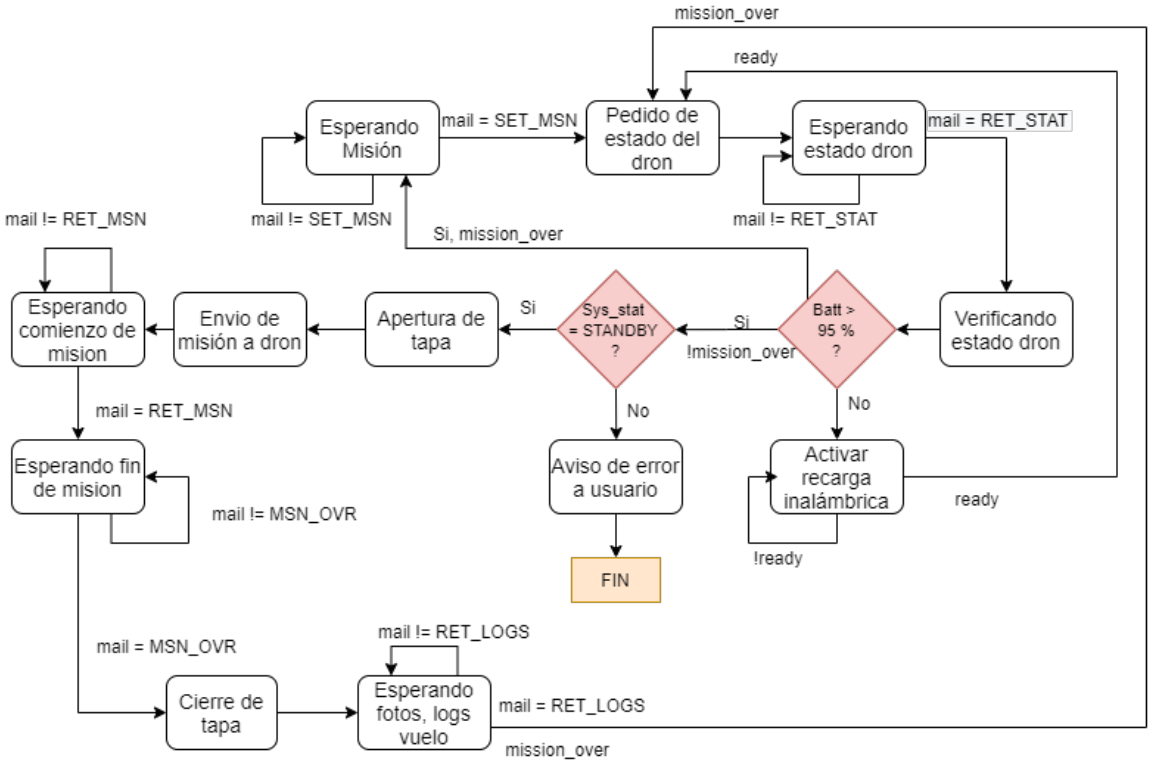

Figura 5.3: Máquina de estados de la base.

La máquina de estados correspondiente a la base comienza (luego de las inicializaciones de parámetros y módulos) por el estado "Esperando Misión", en el cual se monitorean los correos entrantes, filtrando por remitente y asunto (esto se representa en los diagramas de estado con la variable mail y el asunto correspondiente). Una vez recibida una misión[2] desde la interfaz de usuario se procede a verificar que el dron esté apto para volar, para esto se envía un correo al dron con un pedido de reporte, y se pasa a un estado donde se espera por un correo entrante con el asunto correspondiente. Una vez recibido dicho correo desde el dron, se verifica primero el nivel de batería, y si el mismo es menor al 95 % se activa la recarga inalámbrica[3]. Una vez terminada la recarga inalámbrica, se vuelve a pedir el reporte del estado del dron y a esperar el mismo, luego de recibido el correo, se revisa nuevamente el nivel de batería reportado, en caso de ser mayor al 95 %, se pasa a verificar el parámetro *SYS_STAT* provisto por el dron, el cual indica el estado del controlador de vuelo; si dicho parámetro es STANDBY entonces el dron está a la espera de comandos de vuelo, y en caso contrario se tiene algún error en el mismo, por lo que se envía un mensaje de error al usuario/interfaz gráfica y se sale del script, ya que estos errores requieren un chequeo presencial del dron.

[2]Se entiende por misión a los parámetros de configuración de la misma (modo de funcionamiento de cámara térmica, por ejemplo) y los puntos geográficos a recorrer.

[3]Las funcionalidades relacionadas a la recarga inalámbrica y control de hardware de la base son tenidas en cuenta pero no implementadas en código ya que no fueron implementadas en hardware.

Si el estado del dron es adecuado para el vuelo se procede a abrir las barras de asistencia al acople de bobinas y la cubierta del dron, una vez terminado este proceso se procede a enviar un correo al dron con los parámetros de misión, y se pasa a un estado de espera de confirmación de comienzo de misión por parte del dron. Una vez recibida la misma, se pasa a esperar confirmación del fin de la misión (esto incluye el aterrizaje correcto y posterior apagado de los motores del dron, para así poder operar los componentes mecánicos de la base de forma segura), luego se procede al ajuste de las barras de asistencia al acople de bobinas y cierre de la cubierta.

Finalmente, se espera recibir por correo las fotos del vuelo y los logs correspondientes para luego volver a pedir el estado del dron, y, de ser necesario activar la recarga inalámbrica. Teniendo la batería del dron un nivel mayor al 95 % se vuelve al estado inicial de espera de misiones.

Cabe destacar que todos los estados de espera de correos poseen un timer, y en caso de superarse cierto tiempo sin respuesta se toma como una falla en la entidad que debería mandar el correo (en el caso de la base, se asume error en el dron), se envía el mensaje de error al usuario y se sale del script.

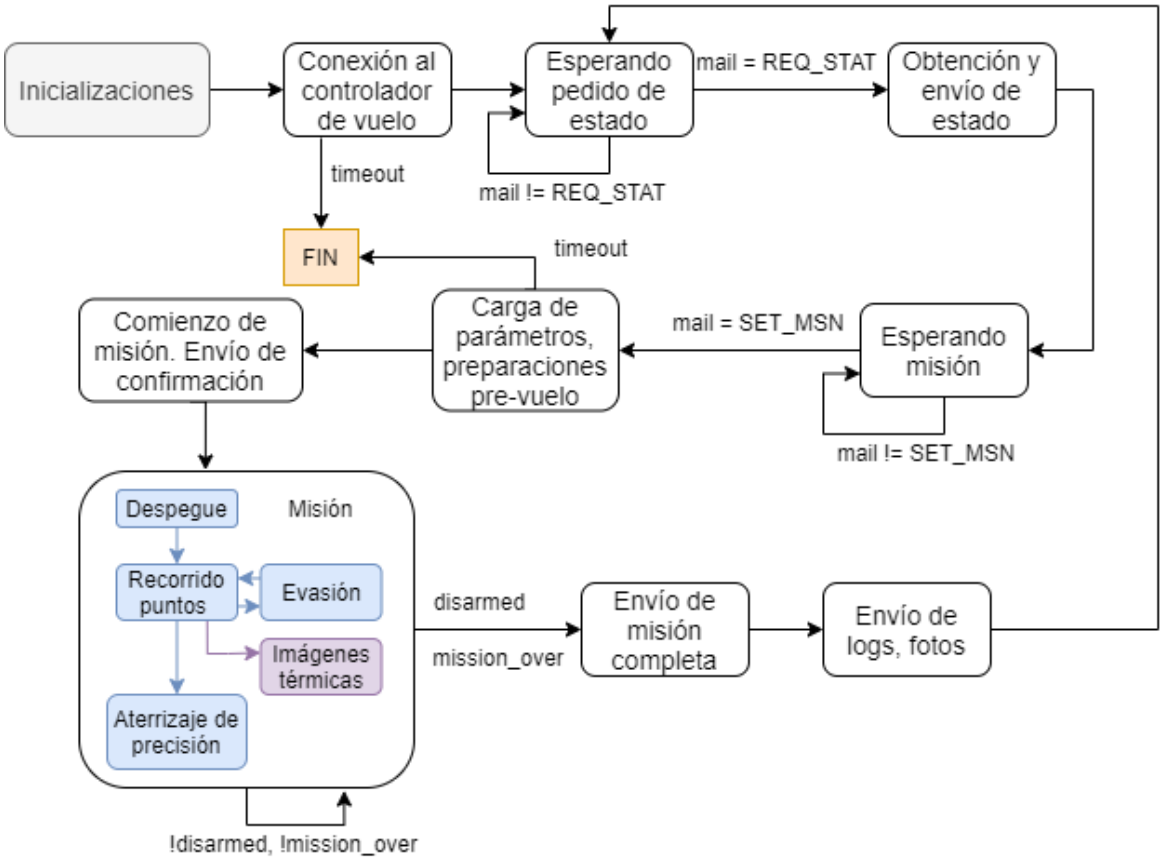

Figura 5.4: Máquina de estados del dron.

La máquina de estados correspondiente al dron comienza realizando las inicializaciones de parámetros y módulos de hardware correspondientes, luego avanza al estado de conexión al controlador, donde se establece la conexión por USB entre el controlador de vuelo y la computadora a bordo, permitiendo así comandar el mismo a través de Python. Se pasa luego a un estado donde se espera recibir un pedido de reporte del estado del dron (nivel de batería, estado del GPS, estado general del sistema, entre otros) desde la base, una vez recibido el correo, se obtiene el reporte del dron y se lo envía por correo hacia la base con el asunto correspondiente.

Luego del envío del reporte se pasa a un estado de espera de la misión a realizar. Durante este estado es posible que se esté realizando la recarga inalámbrica, la cual es controlada por la base. Una vez finalizada la misma, la base enviará la misión al dron luego que éste reporte un estado adecuado para volar.

Una vez recibida la misión, se carga la misma en el controlador de vuelo y se preparan todos los sistemas para comenzar a volar. En este punto, si se detecta algún problema que impida realizar el vuelo, se sale del script. Al salir del script y no enviar ningún mensaje se activará el timeout del estado "Esperando comienzo de misión" de la base, lo cual llevará al envío de un mensaje de error al usuario y el cierre del script de la base también. De esta forma, al ocurrir errores es necesario revisar el sistema para resolverlos y reiniciar las computadoras correspondientes para volver a iniciar las máquinas de estados.

En caso de confirmarse el comienzo de la misión, se avisará a la base de esto y se pasa a ejecutar la misión: comenzando por el despegue, luego se recorren los puntos geográficos seleccionados por el usuario, con la correspondiente evasión de obstáculos si se encuentran durante el vuelo. A su vez se toman imágenes térmicas en caso de encontrar cuerpos o focos ígneos, según corresponda. Una vez terminado el recorrido de puntos geográficos se retorna a las coordenadas de la base, donde se ejecuta el algoritmo de aterrizaje de precisión. Una vez que el dron aterriza y apaga sus motores se envía el correo de misión completada hacia la base, con un posterior envío de las fotos tomadas y los logs hacia la base y el usuario. Finalmente, se vuelve al estado de espera de pedido de reporte del estado del dron.

5.2. Librerías desarrolladas

Dentro de cada entidad (dron, base, interfaz), se decide mantener la estructura de procesamiento en paralelo (en *threads* o hilos) utilizada por Termodron II, de forma de tener librerías de código diseñadas para cada funcionalidad (por ejemplo, una librería para funciones de la cámara estéreo), optimizar el funcionamiento del código, hacerlo más entendible para los futuros usuarios, y facilitar el *debuggeo*.

Dada la naturaleza asíncrona de los hilos ejecutados en paralelo, es necesario una nueva estrategia para compartir datos entre ellos, distinta al uso común de variables compartidas

o variables globales y se opta por dos estructuras de datos: tuberías o *pipelines* y filas o *queues*.

Un *pipeline* es un objeto de la librería *multiprocessing*, el cual, al ser creado, devuelve dos nuevos objetos que llamaremos los "extremos" de la tubería; uno para ingresar datos y otro para retirarlos (o ambos para ambas funciones si se pone en modo *full-duplex*), entonces simplemente se le da acceso a un extremo a cada hilo que se desea comunicar y se envían los datos a la tubería de forma asíncrona entre ellos. Como esta estructura solo permite el intercambio de datos entre dos hilos, también se utilizan las *queues* o filas de la librería *threading*, las cuales son similares a los *pipelines* pero permiten que cualquiera que tenga acceso al objeto *queue* pueda enviar o recibir datos por éste.

Las librerías desarrolladas son:

- libDron
- libStereo
- libArduino
- libThermal
- libMail
- libAWS
- libDGPS

A continuación se tiene una descripción de las librerías principales.

5.2.1. libDron

Esta librería se encarga de toda la comunicación entre la computadora a bordo y el controlador de vuelo. Entre sus funciones se encuentran: conectar con el controlador de vuelo mediante USB, realizar los chequeos de pre-armado, el armado[4], el despegue, recorrido de los waypoints indicados, y la evasión durante el recorrido de los mismos. También controla el gimbal de la cámara, y el aterrizaje de precisión.

5.2.1.1. Aterrizaje de precisión

Se implementa un algoritmo de aterrizaje de precisión para poder asegurar el correcto posicionamiento del dron sobre la base para poder asegurar un acople entre las bobinas del sistema de recarga inalámbrica.

El algoritmo comienza identificando mediante la cámara Intel RealSense D415[5] un marcador ArUco colocado sobre la base, luego se halla la posición relativa del mismo

[4]Armado: encendido de los motores del dron en espera de un comando de despegue.

[5]Se utiliza la misma en su modo de cámara de espectro visible para esta función

respecto al dron, y se le indica al dron ir hacia dicha posición. Se repite este proceso hasta que el dron está a una cierta altura sobre el marcador, donde se le indica al dron cambiar a modo LAND para terminar el último tramo del aterrizaje. Para más detalles sobre el algoritmo implementado ver el capítulo 6.

A continuación se verán en detalle los aspectos de software relacionados con el algoritmo implementado, en especial las bibliotecas y funciones utilizadas.

Se utilizan dos códigos para implementar el aterrizaje preciso, una librería: *lib_aruco_pose* la cual permite la detección y estimación de pose del marcador ArUco deseado; un script: *precLand*, el cual contiene el algoritmo de aterrizaje, y otros dos scripts los cuales son utilizados para la calibración de la cámara a utilizar[6]. Los códigos originales fueron obtenidos de [21] y fueron fuertemente modificados a partir de varias pruebas de campo, buscando lograr resultados mejores, que se ajustaran a las necesidades del proyecto. En particular, se buscó mejorar la precisión del algoritmo original, y el tiempo que tomaba al dron en lograr un aterrizaje.

La librería *lib_aruco_pose* se encarga de definir una clase[7] llamada *ArucoSingleTracker*. Esta clase contiene la función *track*, la cual se encarga de obtener un cuadro de la cámara, convertirlo a escala de grises, buscar el marcador deseado dentro de dicho cuadro y, en caso de encontrarlo, retornar su matriz de rotación y su vector de traslación[8] a partir de las cuales es posible obtener las coordenadas x, y, z del marcador, en centímetros, relativo a la cámara. Una vez obtenida la posición del marcador, se utilizan funciones de esta clase para hallar los ángulos del marcador respecto a la cámara, y para hallar tanto la posición como los ángulos de la cámara respecto al marcador.

Luego, en la imagen se hace un overlay con los ejes correspondientes al marcador (ver figura 6.4), y se agrega también texto indicando la posición y ángulos del marcador respecto a la cámara, así como de la cámara respecto al marcador. Toda esta información desplegada en pantalla es útil para el proceso de prueba y verificación así como para el *debuggeo*, pero no será usada cuando se corra sobre el dron.

Dado que para el algoritmo de aterrizaje implementado no es necesario utilizar tanta información, se modifica la biblioteca para solo obtener las coordenadas del marcador respecto a la cámara, y se agregan otras funciones útiles como ser el guardado de imágenes, o el grabado de video para posterior análisis en las pruebas de campo.

El script *precLand* se basa en el original de [21], el cual conectaba al dron, generaba el objeto *ArucoSingleTracker* y corría un loop de control en el cual se obtenía una imagen de la cámara, se buscaba el marcador en la misma, y, en caso de encontrarlo, se calculaba

[6] Ver A.5.6 para más detalles sobre la calibración

[7] Una clase de Python es un conjunto de funciones propias (métodos) y variables (atributos) que en conjunto cumplen cierta funcionalidad.

[8] Ver 6 para más detalles sobre estos parámetros y su relación con la posición del marcador.

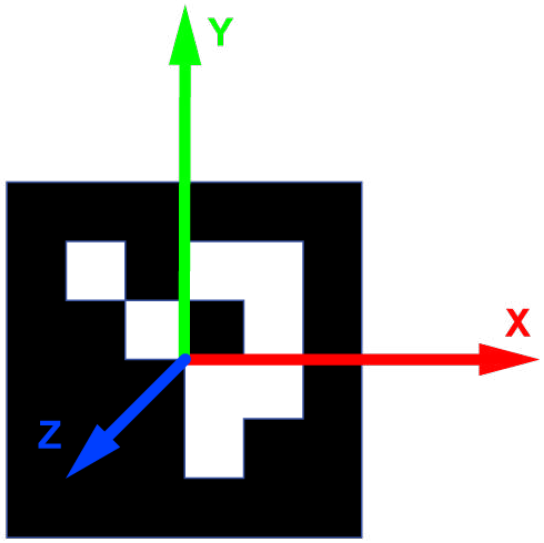

Figura 5.5: Marcador ArUco con su sistema de ejes superpuestos.

el ángulo de visión de la cámara respecto al marcador, y en caso de ser suficientemente pequeño se comanda al dron hacia la posición del marcador, pero con un descenso de altura; en caso contrario se comandaba el dron hacia la posición del marcador pero manteniendo la altura. Una vez que la altura del dron fuera suficientemente baja, se cambiaba a modo aterrizaje y se terminaba el loop.

Dados los resultados de las pruebas de campo que se pudieron realizar con este algoritmo, se decidieron realizar cambios al script original de forma de mejorar el funcionamiento del mismo y agregar respaldos en caso de falla de detección del marcador. En primer lugar, para hacer el código compatible con la arquitectura de software utilizada, se convierte el mismo de un script ejecutable por sí solo a una función capaz de ser importada por el código principal del dron. Luego, como ya se mencionó al hablar de la librería, se agrega la funcionalidad de guardar los cuadros obtenidos de la cámara en video, permitiendo así ver los cuadros capturados por la cámara a la hora de realizar el aterrizaje, y permitiendo corregir errores como pueden ser un alto nivel de vibración que genera cuadros movidos e impiden la detección del marcador.

También se agrega una acción de contingencia para cuando el algoritmo no logra ubicar el marcador (por ejemplo, si hay viento y se mueve el dron de una posición cercana a la base, perdiendo visión del marcador). Esta utiliza un timer, el cual es incrementado luego de que no se detecta un marcador, y vuelve a cero cuando se logra detectar. En caso de que expire el tiempo límite del timer (se encontró experimentalmente que 5 segundos son suficientes para declarar como perdido el marcador), se comanda al dron a subir un 20 % de su altura actual, y, cuando vuelva a encontrar el marcador, no descender, sino ir directo a las coordenadas del marcador manteniendo la altura. De esta forma, se asegura que una

vez que se pierde visión del marcador, al retomarlo el dron intentará centrarse sobre el mismo nuevamente.

Otro cambio realizado es la acción de movimiento del dron una vez encontrado el marcador: una vez que lo detecta, y si el ángulo de visión del dron sobre el marcador es menor a un valor fijo, se indica al dron descender verticalmente, sin cambio de coordenadas. De esta manera este algoritmo alinea al dron con el marcador si se está por fuera de ese ángulo de visión, y de lo contrario simplemente se desciende, acelerando de forma considerable el tiempo de aterrizaje.

Finalmente, se hace un cambio sustancial con el algoritmo original que es agregar otro marcador ArUco, de distinto tamaño, junto al original. Esto se hace porque a grandes alturas puede que no se logre detectar correctamente el marcador si su tamaño no es suficientemente grande (ya que se pierde el detalle de los bits blancos y negros del mismo), y por otro lado, a bajas distancias un marcador muy grande puede no entrar dentro del cuadro imagen, impidiendo también su detección. Entonces, utilizando un marcador de tamaño mayor (27 cm de lado) para la primera etapa del aterrizaje, es decir, a mayores alturas, y luego de cierta altura umbral cambiar en el algoritmo la búsqueda a otro marcador de menor tamaño (16 cm de lado), se logra evitar, en gran parte, las pérdidas de visión del marcador debido a la altura. Además, el uso de un marcador más pequeño en el tramo final del aterrizaje permite que el dron descienda más altura manteniendo visión del mismo, por lo que se puede aterrizar a una distancia aún más cercana sobre el marcador, incrementando la precisión.

Para lograr esta funcionalidad se tuvo que editar nuevamente la librería *lib_aruco_pose* para facilitar la detección de dos marcadores distintos, y evitar la detección errónea de los marcadores, ya que los mismos están situados muy cerca entre sí, quedando ambos dentro del cuadro de la cámara a valores de altura suficientes. Para esto se encontró que aunque se cambiara el número identificador del marcador, podían existir falsas detecciones entre ambos, por lo que se decidió cambiar el diccionario utilizado también, efectivamente cambiando la cantidad de bits de la matriz interna del marcador.

En anexo A.3 se tiene una descripción detallada de las funciones implementadas en esta librería. En el capítulo 6 se entra en detalle sobre el fundamento teórico de la detección de marcadores ArUco, su estimación de pose y el algoritmo de aterrizaje preciso.

5.2.1.2. Evasión de obstáculos

Esta funcionalidad se implementa utilizando funciones de control fino del movimiento del dron, provistas por los códigos ejemplo de *Dronekit* [41]. En particular, se usan las funciones *set_velocity_body()*, la cual permite elegir la velocidad en cada una de las componentes del dron (x,y,z) y se mantendrá la misma durante un segundo, y *condition_yaw()* la cual permite controlar el ángulo de *yaw* del dron.

El algoritmo es activado si se tiene una bandera activa de obstáculo detectado en el *pipeline* correspondiente. Se procede a obtener de otro *pipeline* los datos de distancia a los objetos detectados en cada una de las tres secciones de la imagen de la cámara estéreo

(ver **??** para más detalles sobre la detección de objetos). Una vez obtenidas las distancias de cada sección, se evalúa primero las situaciones críticas, es decir, si se detecta un objeto a una distancia muy cercana (menor a 2 metros), o si se encuentran obstáculos en las tres secciones. Para más detalles sobre el comportamiento del dron durante la evasión referirse a la sección **??**.

5.2.1.3. Barrido de un área

Una de las funciones implementadas en Termodron I fue el barrido de un área, es decir, dado un conjunto de puntos que describen el perímetro de un polígono convexo, el dron debía moverse de forma de cubrir la mayor cantidad de superficie interna a dicho polígono posible, como se puede ver en la imagen 5.6. Esta función no fue implementada por Termodron II ya que hubo un cambio en el firmware utilizado para el dron y esto implicaba un cambio en el lenguaje de programación del mismo.

Para este proyecto se decidió volver a incorporar dicha funcionalidad, y se utilizó un algoritmo obtenido del repositorio de *GitHub* PythonRobotics [25] el cual está implementado en Python e incluye un *script* principal el cual realiza el cálculo del barrido de área y una librería de funciones auxiliares. El *script* principal recibe las coordenadas de los vértices del polígono (en orden, siendo el último punto recibido igual al primero para tener una curva cerrada) y calcula las coordenadas interiores al polígono a recorrer, y su orden. Esto es realizado utilizando un procedimiento muy similar al del algoritmo implementado por Termodron I, siguiendo segmentos de rectas paralelas a uno de los lados del polígono borde, y cambiando dirección al llegar a uno de los bordes, para luego continuar con otro segmento de recta paralelo al anterior, es decir, realizando una especie de *zig-zag*. Esto se puede ilustrar mejor en la imagen 5.6, donde en azul se muestra el polígono borde, y en rojo la trayectoria en *zig-zag* obtenida.

Una vez calculadas las coordenadas a recorrer, éstas son enviadas al dron, el cual irá recorriéndolas en orden hasta finalizar el barrido del área.

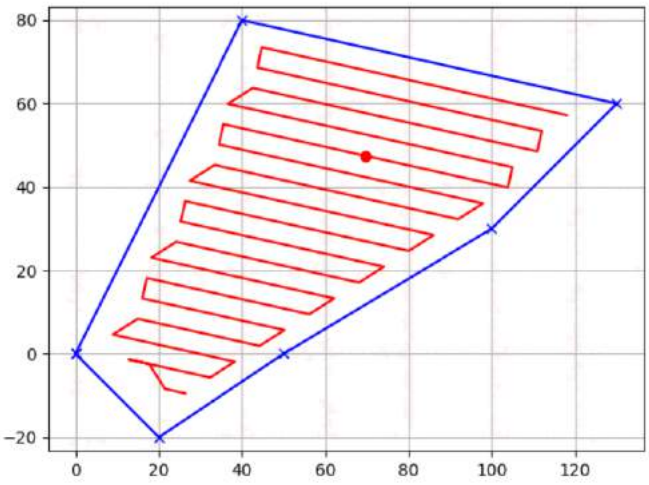

Figura 5.6: Ejemplo de barrido del área. Los puntos en azul describen el borde de la superficie que se quiere barrer. En rojo aparece el recorrido que tomará el dron.

5.2.1.4. threadDron

A continuación se describe el *thread* correspondiente al funcionamiento del dron durante una misión: *threadDron*. Este hilo es iniciado por el código principal del dron una vez que se realizaron todos los chequeos de seguridad previos y se recibió una misión (en la figura 5.4 se puede observar el funcionamiento completo del código principal).

El *thread* comienza con la obtención de los parámetros de la misión mediante la función *set_mission_parameters()*, la cual obtiene, del JSON recibido desde la base, la misión a realizar y los parámetros configurables de la misma. Los parámetros de este documento que son relevantes a este *thread* se listan a continuación. Para más información sobre el documento JSON utilizado para la configuración de misión ver 5.2.6.

- ***homelat***: Latitud de la base
- ***homelon***: Longitud de la base
- ***latitud***: lista que contiene la latitud de cada punto geográfico a recorrer en la misión.
- ***longitud***: lista que contiene la longitud de cada punto geográfico a recorrer en la misión.
- ***altura***: lista que contiene la altura de cada punto geográfico a recorrer en la misión.
- ***thermalMode***: variable que contiene el modo de funcionamiento de la cámara térmica: “cuerpos” para detección de cuerpos calientes entre 35 y 40 °C, o “focos” para detección de focos ígneos con temperaturas mayores a 100 °C.

Luego se llama a la función *set_mission_waypoints*, la cual se encarga de cargar las listas de latitud, longitud y altura al controlador de vuelo. Además, esta función, configura como último punto a recorrer el correspondiente a la base, con el fin de que una vez finalizado el recorrido el dron retorne hacia la misma para luego realizar el aterrizaje de precisión.

Luego, se configura la velocidad de vuelo del dron y se agregan las funciones *listener* o escuchas, las cuales se actualizan cada vez que se realiza un cambio en uno de los parámetros a monitorear. Estos cambios son enviados a una función de *callback* llamada *set_callback*, para realizar el reporte en tiempo real.

Se procede entonces al armado y despegue del dron (previo chequeos de seguridad). Una vez que se despega, se sube a una altura predeterminada de 10 metros y se esperan cinco segundos con el fin de estabilizar el dron, para luego comenzar con la misión. Para esto, se cambia el modo de vuelo a AUTO, y el dron comienza a recorrer automáticamente los puntos geográficos cargados en su memoria. En caso de presentarse un obstáculo, el algoritmo comienza las acciones de evasión, cambiando el modo de vuelo a GUIDED, y, mediante los datos obtenidos por la cámara estereoscópica y la librería libStereo, se le ordena al dron moverse en cierta dirección con el fin de evadir los posibles obstáculos. Luego de terminar la evasión y en caso de no detectarse otro obstáculo, se vuelve a cambiar el modo de vuelo a AUTO, retomando entonces el dron el rumbo hacia el punto geográfico actual.

Una vez finalizado el recorrido de los puntos, el dron retorna a las coordenadas de la base (a una altura de 8 metros) y se comienza el aterrizaje de precisión. Primero se gira el ángulo de pitch del gimbal y luego se llama a la función que implementa el algoritmo de aterrizaje preciso descrito en 6. Una vez terminado el aterrizaje y desarmado el dron, se da por finalizada la misión.

Para más información sobre las funciones mencionadas, ver A.3.

5.2.2. libStereo

Esta librería se encarga del funcionamiento de la cámara estéreo. Se utiliza la librería *librealsense*, en particular su *wrapper*[9] para Python: *pyrealsense2* provista por Intel para uso con sus cámaras de la línea RealSense. La misma puede instalarse usando el instalador de paquetes de Python, *pip*, mediante el comando *pip install pyrealsense2* o se puede compilar la librería directamente desde fuente con los archivos disponibles en el repositorio de GitHub oficial [26].

En particular, esta librería se encarga de implementar el algoritmo para detección de objetos. Basado en el ejemplo *align-depth2color* provisto por Intel para usar con la

[9]Un wrapper de una librería es una capa de código para convertir la librería original a otro lenguaje o interfaz de programación.

cámara estéreo, se adapta el mismo para lograr sensar objetos a menos de una cierta distancia umbral de la cámara [26].

El script creado, *threadStereo*, comienza con la configuración de la cámara, en particular, se configura la resolución a 640x480 píxeles, ya que en mayores resoluciones se lograba un rendimiento[10] pobre en la computadora a bordo. Se configuran diversos filtros propios de la librería: filtro de decimación, el cual realiza un *downsampling* para reducir la complejidad de la información de profundidad obtenida de la cámara, reduciendo también la resolución de la imagen obtenida. Filtro de "llenado de agujeros" o *hole-filling*, el cual se encarga de rellenar la información de profundidad perdida en algunos píxeles, interpolando según los píxeles en un entorno. Luego, se obtiene el parámetro de escala de profundidad de la cámara, el cual relaciona el valor de profundidad de un píxel retornado por la cámara con la distancia del objeto en ese píxel hacia la cámara. Finalmente, se activa el emisor láser de la cámara, el cual agrega información a las superficies que apunta la cámara, mejorando la detección de distancia (en particular, para superficies planas). La activación del emisor láser es opcional y puede ser controlada por el software.

Una vez terminadas las inicializaciones, el script comienza un loop infinito, el cual puede ser terminado mediante una bandera que puede ser activada por otros scripts del sistema (en particular, el script *Dron* es quien controla cuándo activar y desactivar esta bandera). En este loop se obtiene el "cuadro de profundidad", esto es, una matriz de datos del tamaño de la imagen de la cámara, donde cada elemento de la misma contiene la información de profundidad[11] de la cámara hacia ese punto, se aplican los filtros que fueron configurados anteriormente al cuadro obtenido, y se convierte el mismo a un arreglo del tipo *numpy.array*, de forma de facilitar las operaciones a nivel de arreglos gracias a la librería de Python *NumPy*.

Se utiliza la función *where()* de *NumPy* para poder filtrar en el arreglo aquellos píxeles que están dentro del rango deseado de distancias[12] a detectar, obteniéndose un nuevo arreglo donde los elementos del mismo que no estén dentro del rango de distancias deseadas son asignados un valor de 0, y los demás mantienen su valor actual. Se separa este arreglo horizontalmente en tres secciones y se procede a hallar la mínima distancia detectada en cada una de ellas mediante el mismo algoritmo.

Debe ser notado que todo arreglo de *NumPy* puede ser interpretado como una imagen en escala de grises, por lo que se utiliza una librería de procesamiento de imágenes como es *OpenCV* para obtener la distancia mínima detectada en cada sección, y se referirá indistintamente como arreglo o imagen de ahora en más. Se aplica segmentación binaria al arreglo obtenido anteriormente utilizando la función *inRange()* de *OpenCV*, obteniendo

[10] En este caso, rendimiento refiere a la cantidad de cuadros por segundo que logra procesar el algoritmo.

[11] La información de profundidad multiplicada por la escala de profundidad de la cámara da la distancia hacia la cámara, en metros.

[12] En realidad se busca por profundidad en vez de distancia, pero como ya se mencionó ambas están directamente relacionados por la constante *escala de profundidad*.

una imagen donde los píxeles cuyo valor de profundidad estuviera dentro del rango de interés son pintados de blanco (valor 255) y los que están fuera del rango son pintados de negro (valor 0). Luego se hallan los contornos[13] en la imagen binaria con la función *findContours()* de *OpenCV*. Se itera luego en la lista de contornos obtenidos, donde se descartan los de área muy pequeña, y a los restantes se les halla el mínimo rectángulo que los envuelve, para luego tomar la porción del cuadro de profundidad correspondiente a ese rectángulo y buscar dentro de éste sub-arreglo el menor valor de profundidad, usando la función de *NumPy*, *min()* y luego convertir a distancia en metros.

Finalmente, se toma el cuadro de profundidad, se convierte a imagen de color (es decir, imagen de tres canales, azul, verde, rojo) y se le aplica un mapa de color con la función *applyColorMap()* de *OpenCV* para asignar a cada distancia un color distinto. Se agrega también un overlay que indica las tres secciones de la imagen, el rectángulo que envuelve el contorno con la menor distancia detectada hacia la cámara y dicha distancia. Esto se hizo para el testeo y *debuggeo*, no siendo utilizada durante el vuelo, y en ese caso, simplemente se envía por una tubería o *pipeline* una bandera de "obstáculo detectado" y en otra tubería la información de distancia detectada en cada sección de la imagen.

5.2.3. libArduino

Esta librería es implementada para la comunicación entre los sensores de ultrasonido, el Arduino Nano y la computadora a bordo (Odroid-XU4). Como se mencionó en 4.1.13, se cuenta con dos sensores de ultrasonido, uno de los cuales será utilizado en el algoritmo de detección y evasión de obstáculos, y el otro en el algoritmo aterrizaje de precisión. Se utiliza un Arduino como nexo entre la Odroid y los sensores ya que ésta no posee un ADC propio y los sensores de ultrasonido devuelven un voltaje analógico proporcional a la distancia.

Para lograr esta comunicación se cuenta con dos algoritmos, uno que se ejecuta en el Arduino y el otro en la Odroid. El algoritmo ejecutado en el Arduino obtiene las medidas de voltaje analógico en el ADC, luego convierte las mismas a un valor de distancia y las envía mediante el puerto serial (USB) hacia la Odroid. Los datos enviados son una cadena de caracteres y la Odroid se encargará de traducirlos y filtrarlos para obtener las medidas. Esta cadena de caracteres se compone de dos conjuntos de caracteres iguales separados por dos espacios, estos conjuntos de caracteres son de la forma "$S_i =$" (donde el subíndice i toma valores 1 o 2 indicando el sensor), un espacio y la medida, en metros. En 5.2.3 se pueden ver dichas cadenas de caracteres. En la figura 5.8 se presenta el esquema del algoritmo implementado en el Arduino.

$$S_1 = \text{Medida } S_2 = \text{Medida} \tag{5.2.3}$$

[13] Un contorno es una curva continua que envuelve un grupo de píxeles con similares valores o intensidades de color.

En la Odroid, se encuentra ejecutándose en paralelo un algoritmo el cual comienza leyendo el puerto serial correspondiente a la comunicación con el Arduino hasta obtener el fin de carro, luego realiza una conversión de los datos obtenidos de binarios a caracteres, filtra los caracteres buscando las dos medidas de interés, las convierte a enteros y las envía a los algoritmos de evasión de obstáculos y aterrizaje de precisión.

Para convertir los datos binarios a cadenas de caracteres se utiliza la función de Python *str()*, luego para realizar el filtrado y la conversión a enteros se utiliza una iteración en donde se convierte la cadena de caracteres a una lista, mediante la función *split()*, la cual contendrá en cada posición de la lista los caracteres separados por los espacios; en este caso la lista contendrá los siguientes valores $[S_1, Medida, S_2, Medida]$. Luego se recorre la lista y se utiliza la función *isdigit()*, la cual devuelve en una variable entero el dígito correspondiente a las medidas. Para finalizar se envían las dos medidas a los algoritmos mencionados. En la figura 5.8 se presenta el esquema del algoritmo implementado en la Odroid.

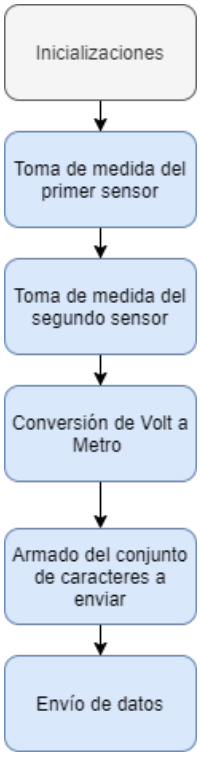

Figura 5.7: Algoritmo implementado en el Arduino para la comunicación con los sensores de ultrasonido.

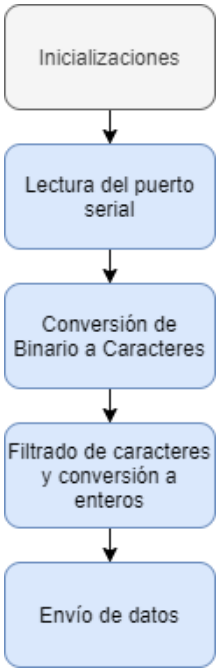

Figura 5.8: Algoritmo implementado en la Odroid para la comunicación con el Arduino.

5.2.4. libThermal

Esta librería es la que implementa el algoritmo de detección de cuerpos calientes, mediante la cámara térmica FLIR Lepton 3.5.

Con el fin de tener un conocimiento sobre el funcionamiento de la cámara térmica se realiza un breve resumen de la funcionalidad de la cámara térmica y de la teoría sobre la radiación infrarroja el cual se presenta en el anexo A.2.

Mas información sobre la tecnología infrarroja y el uso de la cámara térmica FLIR Lepton 3.5 se puede encontrar en [27] [28].

5.2.4.1. Algoritmo de detección de cuerpos calientes

Se implementó un algoritmo similar al utilizado tanto por Termodron I como por Termodron II, en el cual se busca el cuerpo caliente de mayor tamaño y se reporta las imágenes obtenidas del mismo, pero la forma de implementar el mismo cambió sustancialmente, ya que se hace un cambio en el lenguaje de programación; las versiones anteriores de Termodron implementaban el algoritmo en el Arduino a bordo utilizando C++, mientras que en este proyecto se implementa el código en la Odroid XU4 a bordo del dron, utilizando Python 3. Este cambio de lenguaje de programación tiene como principal ventaja el uso de una variada cantidad de librerías de uso público, como lo son *OpenCV* y *NumPy*, las cuales permiten una mejor implementación de algoritmos de tratamiento de imágenes. También se utiliza la librería *purethermal1-uvc-capture* provista por los desarrolladores del modulo de entrada y salida, breakout board[14] utilizada en la cámara térmica, GroupGets [32].

El algoritmo comienza realizando las inicializaciones de parámetros y banderas, luego se procede a conectar la cámara térmica y se toma el primer cuadro de datos (es la imagen térmica provista por la cámara: una matriz de 160x120 donde cada elemento contiene el valor de temperatura del píxel correspondiente) y se comienza un loop infinito dentro del cual se genera una máscara binaria fijando en 1 el valor de los píxeles que tengan valores de temperatura dentro del rango buscado, y 0 en caso contrario. Para ello se utiliza la función *inRange()* de *OpenCV*.

A la máscara se le aplica la función *findContours()*, también de *OpenCV* con la cual se obtienen los contornos de todos los cuerpos presentes en la imagen. Mediante la función *contourArea()* de *OpenCV*, se busca el contorno de mayor área obteniéndose también las coordenadas del centro y el radio del menor círculo que lo envuelve. Se compara la posición de este contorno con el anterior (en caso de ser el primero, no se realiza este paso) y, en caso de que se solapen los círculos que los envuelven, se descarta este cuadro y se vuelve al comienzo del loop. De lo contrario, se sobreescribe la posición del contorno anterior con el actual (evitando así fotografías repetidas del mismo cuerpo caliente) y se procede a convertir la imagen a formato RGB (la imagen original es de un sólo canal, con información de temperatura en cada píxel), se dibuja el círculo que envuelve al contorno hallado mediante la función *circle()* de *OpenCV* y se guarda la imagen.

[14]Breakout board: placa de desarrollo para utilizar hardware específico.

En caso de no encontrar contornos, se procede a tomar un nuevo cuadro y repetir el procedimiento.

Al terminar una misión y una vez en la base y con los motores apagados, la computadora a bordo del dron revisará la carpeta creada por esta librería y, si hay imágenes, las enviará por correo hacia el usuario y el software de la interfaz gráfica.

En la figura 5.9 se tiene un diagrama de bloques del algoritmo implementado.

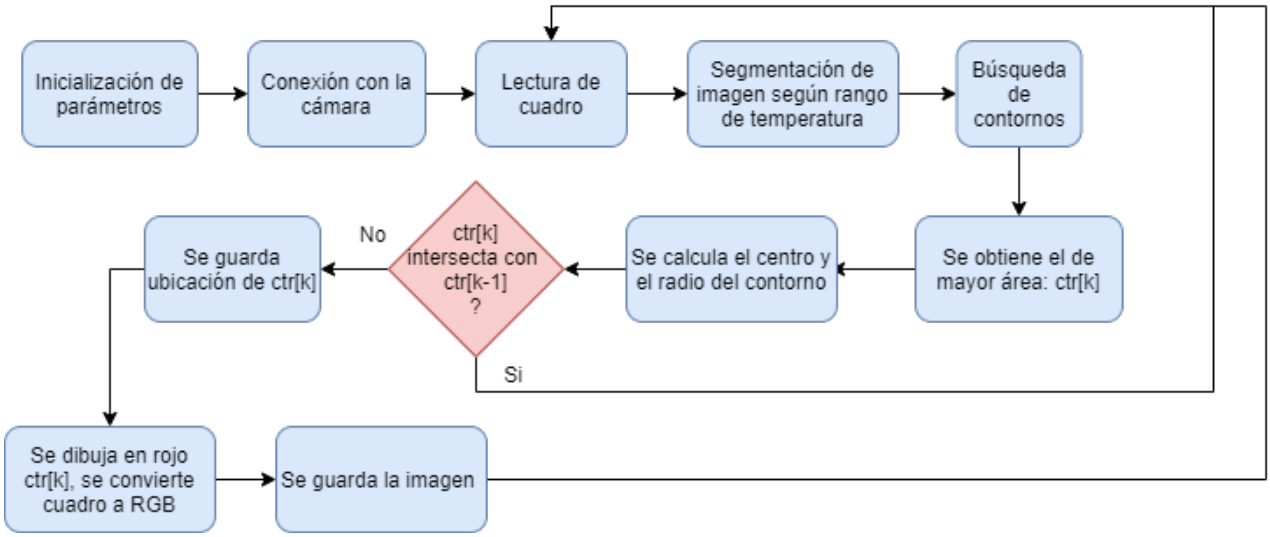

Figura 5.9: Diagrama general del algoritmo implementado para la detección de cuerpos calientes mediante la cámara térmica.

En la figura 5.10 se presenta una imagen ilustrativa del resultado obtenido al implementar el algoritmo, nótese el círculo rojo encerrando el rostro de una persona, y la imagen térmica en escala de grises, con los valores de color más claros correspondiéndose a valores de temperatura más elevados.

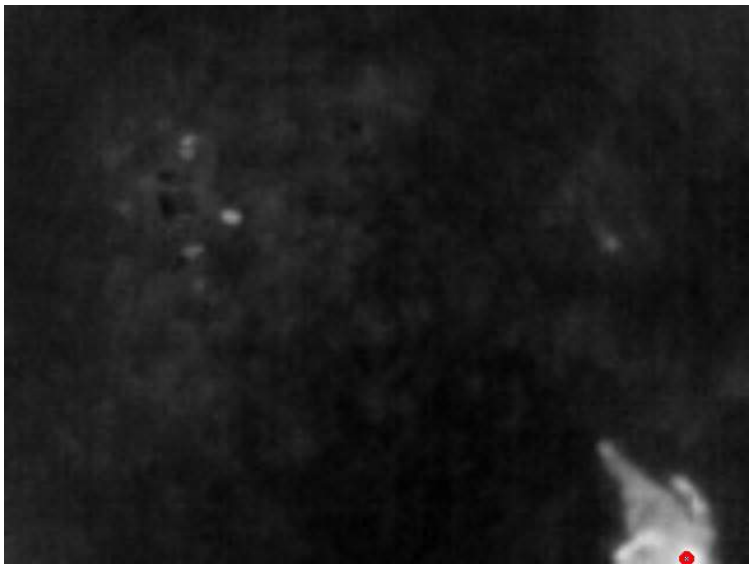

Figura 5.10: Imagen obtenida mediante el uso del código de detección térmica. Se detectan cuerpos cuya temperatura esté entre 30 °C y 40 °C.

5.2.5. libMail

Esta librería es la que contiene las funciones para enviar y recibir correos electrónicos, los cuales servirán para comunicar las tres entidades del sistema: el usuario (mediante la interfaz gráfica), la base, y el dron.

Se crearon cuentas de correo de Gmail para las tres entidades, ya que es posible utilizar el servidor SMTP de Gmail mediante Python para enviar correos electrónicos. Para recibir correos se utiliza IMAP con SSL. A continuación se listan las bibliotecas utilizadas y sus funciones.

- email: librería que permite dar formato a los correos electrónicos a enviar, en particular se utiliza el módulo *mime* para enviar texto plano, texto en html e imágenes
- smtplib: permite realizar conexiones seguras a servidores SMTP con SSL para poder enviar correos electrónicos
- imaplib: permite realizar conexiones a servidores IMAP, realizar identificación segura y filtrar los correos recibidos para su posterior lectura

A partir de dichas bibliotecas se implementan funciones que permiten entrar a una cuenta de Gmail, filtrar los correos no leídos por usuario y/o por adjunto, y descargar los mismos (tanto su texto plano, texto html y sus adjuntos). A su vez se crean funciones para enviar correos electrónicos, con texto plano, texto html e imágenes adjuntas. Se implementan funciones de monitoreo de correos, las cuales mantienen un loop esperando la llegada de correos con asuntos específicos (por ejemplo, la llegada de los parámetros

de una misión previo al vuelo), y también monitoreo del correcto envío de correos. Estas funciones son claves para lograr la comunicación entre las tres entidades del sistema (interfaz gráfica, dron, y base).

La comunicación entre las diferentes entidades del sistema está definida en los diagramas de estado de las figuras 5.3, 5.4, donde una entidad puede realizar un envío de parámetros a fijar (por ejemplo, la base envía al dron las coordenadas a recorrer), o un envío de pedido de información de otra entidad (por ejemplo, la base envía al dron un pedido de reporte de su estado antes de volar), o esperar recibir un correo de una entidad (por ejemplo, el dron espera recibir los parámetros de una misión desde la base para poder comenzar a volar).

Según el estado en que se encuentre una entidad, genera el correo electrónico con un asunto dado de forma que la entidad que reciba dicho correo pueda identificarlo inequívocamente (busca por el correo del remitente esperado, y el asunto esperado según el estado en el que se encuentra), evitando así falsos comandos dados por correos electrónicos de terceros y permitiendo que las máquinas de estado de las entidades dron y base funcionen asíncronamente, ya que sólo recibirá los correos relevantes al estado actual y no los de estados futuros.

5.2.6. libAWS

Esta librería proporciona funciones para conectar con el servidor de Amazon Web Services, ingresar las credenciales correspondientes a cada entidad registrada [15] para poder realizar actualizaciones del estado de la misma, o para poder monitorear el estado de otra entidad. En [36] se tiene un instructivo detallado para crear nuevas entidades y obtener sus credenciales correspondientes.

El monitoreo y reporte del estado son realizados mediante un sistema de publicación/suscripción a distintos *topics* o temas; de esta manera, la entidad que reporta su estado lo hace actualizando el topic correspondiente a su *shadow* (esto es un documento en formato JSON[16] el cual incluye distintos parámetros relevantes al estado de la misma). La entidad que desea monitorear el estado de otra se subscribe al *topic* correspondiente al *shadow* de dicha entidad. Para más información sobre *topics* y *shadows*, ver [35].

Para poder reportar y modificar el estado de una entidad se utiliza el *shadow* de la misma, en la sección *reported*. También se tiene una sección *desired* del *shadow*, en la cual se pueden ingresar cualidades deseadas del estado de la entidad; estos campos se utilizan

[15] Se le llama entidad o *thing* a aquella que posea los certificados y claves públicas generadas por medio de la consola web de AWS, permitiendo así comunicar entidades entre sí, o monitorear el estado de una, entre varias otras funciones.

[16] Estilo de formato de documento de intercambio de datos, el cual es legible por personas y de gran facilidad de procesamiento para las computadoras, funcionando con pares de atributos y valores.

para el seteo de misión (se hace por correo electrónico pero se utiliza el mismo documento JSON para mantener la posibilidad de hacerlo mediante AWS). A continuación se muestra el JSON utilizado en sistema.

```
{
  "state": {
    "desired": {
      "baseLocation": {
        "lat": [],
        "lon": []
      },
      "AWS":"",
      "mailList": [],
      "thermalMode": "",
      "waypoints": {
        "lat": [],
        "lon": [],
        "alt": ""
      }
    },
    "reported": {
      "location": {
        "lat": "",
        "lon": "",
        "alt": ""
      },
      "battery": "",
      "lastHeartbeat": "",
      "armable": "",
      "armed": "",
      "mode": "",
      "gpsStatus": ""
    }
  }
}
```

Se puede observar que el JSON contiene un objeto principal llamado *state* el cual contiene las secciones *desired* y *reported* ya mencionadas. Dentro de la sección *desired* se encuentran los parámetros que el usuario configura para el comienzo de una misión, mediante la interfaz de usuario. Estos parámetros son:

- baseLocation: coordenadas (latitud, longitud) de la base del dron. Se recomienda dejar la base fija y conocer su posición con gran precisión para ayudar al aterrizaje preciso.

- AWS: indica si activar o no reportes en tiempo real por AWS
- mailList: lista de correos electrónicos a donde enviar las fotos y logs luego del vuelo
- thermalMode: modo de detección de la cámara térmica: cuerpos (entre 30 y 45 °C) o focos (más de 100 °C)
- waypoints: coordenadas y altura de los puntos a recorrer por el dron durante el vuelo

Luego, en la sección *reported* se tienen los parámetros que el dron va a reportar en tiempo real, éstos son listados a continuación:

- location: coordenadas y altura de la ubicación actual del dron
- battery: porcentaje de batería restante del dron
- lastHeartbeat: segundos desde que se recibió el último mensaje de *heartbeat* del controlador de vuelo, indicando que éste está conectado y funcionando
- armable: indica si el dron puede ser armado[17].
- armed: indica si el dron está armado o no
- mode: indica el modo de vuelo del dron
- gpsStatus: indica el estado actual del módulo de GPS

Se implementan dos scripts principales: *threadAWS*, para reporte y otro, *AWSClient*, para monitoreo de entidades.

El script *threadAWS* es implementado en el dron, para reportar el estado del mismo en tiempo real. Comienza por obtener las credenciales correspondientes a la entidad a utilizar, luego abre una conexión hacia el servidor de AWS y entra en un loop infinito donde, cada dos segundos, revisa si hay datos nuevos ingresados por el *pipeline*[18] correspondiente. En el caso de utilizarlo en el dron, un extremo del *pipeline* es utilizado por este script y el otro extremo lo tiene el script *Dron*, el cual ingresa por éste los datos de estado del dron en tiempo real.

El script *AWSClient* es implementado en la interfaz de usuario, de forma de poder monitorear el estado del dron en tiempo real. Este script puede utilizarse en cualquier otra plataforma que tenga el SDK AWS-IoT para monitoreo en tiempo real del dron, solo haciendo falta registrar esa nueva entidad en el servidor de AWS. Este script comienza

[17]El armado de los motores es el encendido de los mismos previo al despegue, en el cual giran con una velocidad tal que el dron no levanta vuelo.

[18]Es una estructura de datos en donde se pueden ingresar datos, o removerlos de la misma, similar a una fila o array circular. En el caso de una tubería del módulo *multiprocessing* de Python, sólo se tiene un extremo para ingresar datos, y un extremo para ingresarlos. Las mismas pueden ser single o full duplex

obteniendo las credenciales correspondientes a la entidad a utilizar, luego conecta al servidor de AWS y se suscribe al *topic* correspondiente al *shadow* del dron, de forma de recibir todas las actualizaciones del estado del mismo. Una vez obtenidas las mismas, se pueden enviar por un *pipeline* para compartir con otra función o hilo, o mostrase en el mismo script.

5.2.7. libDGPS

Esta librería, la cual se ejecuta en la base, se encarga de la recepción de los mensajes de corrección de GPS diferencial, la creación de una conexión paralela con el controlador de vuelo mediante radiofrecuencia, y el envío de estos mensajes de corrección hacia el mismo. Para más información sobre el sistema de corrección diferencial de posición satelital (DGPS) ver A.5.3.6, y [10].

Se utilizan dos scripts de Python, ambos utilizados por Termodron II, uno de ellos, *NTRIP_client* es obtenido de [54] y el otro, *threadDGPS* es implementado por Termodron II [10]. Estos códigos se corren en la computadora de la base. A continuación se hace una descripción de las funciones realizadas por cada uno.

El script *NTRIP_client* se encarga de realizar la conexión hacia el servidor o *caster*, ingresar las credenciales del usuario registrado en el sistema REGNA-ROU [56], recibir los mensajes de corrección de posición y redirigirlos hacia un puerto local del equipo (se utiliza el puerto 13320). En este caso se utiliza el servidor ubicado en la Fortaleza del Cerro por ser la más cercana a la Facultad de Ingeniería, donde se realizaron los vuelos del dron.

Luego, el script *threadDGPS* se encarga de iniciar una conexión paralela hacia el controlador de vuelo, utilizando el protocolo MAVlink. Se hace esta conexión de forma paralela y no utilizando la conexión ya existente entre la computadora a bordo y el controlador de vuelo ya que es necesario tener una cierta frecuencia de envío de los mensajes de corrección para que funcione correctamente. Una vez iniciada la conexión con el dron, se procede a abrir el puerto 13320 del *localhost* para recibir los mensajes de corrección recibidos del servidor. Luego, estos mensajes son enviados por la conexión MAVlink hacia el controlador de vuelo mediante radiofrecuencia.

5.2.8. Base

El software de la base del dron utiliza las librerías *libMail*, *libDGPS* y, opcionalmente, puede usar *libAWS*.

Se utiliza *libMail* para tener acceso a las funciones de comunicación mediante correo electrónico, de forma de comunicar con la interfaz de usuario para la obtención de la misión y con el dron para el intercambio de parámetros de estado y de misión previo al vuelo. En la figura 5.3 se pueden observar las distintas funciones de correo electrónico implementadas en la base.

Como se mencionó anteriormente, la base es quien envía los mensajes de corrección de GPS hacia el controlador de vuelo, para esto utiliza las funciones de *libDGPS*.

Finalmente, la base puede utilizar *libAWS* para monitorear el estado del dron, aunque no reporta estos datos, se deja como módulo opcional para testeo.

Debe ser mencionado que el software de control de los componentes mecánicos y de activación de la recarga inalámbrica no fueron incluidos ya que la base fue diseñada pero no implementada.

5.2.9. Interfaz de usuario

Se implementó una interfaz gráfica mediante el uso de Python y la librería *Tkinter*, con la cual se implementa la comunicación entre el usuario y el conjunto dron-base. El objetivo de la misma es facilitar el seteo de misiones por el usuario, y también facilitar la obtención de los reportes en tiempo real, si se desea. En la sección 7.1 se detalla la implementación y el modo de uso de la misma.

Capitulo 6

Sistema de aterrizaje de precisión

Dado que el sistema de recarga inalámbrica implementado por Termodron II requiere una alineación precisa de las bobinas de la base y del dron para poder comenzar la transferencia de energía, es necesario lograr que al terminar una misión el dron pueda aterrizar en una posición deseada en la base, con un mínimo error. Con este fin se implementa un algoritmo de aterrizaje de precisión basado en reconocimiento de patrones. El algoritmo se basa en el implementado en [21], y utiliza la computadora a bordo, la cámara estéreo (en particular, utiliza sólo el módulo de cámara de espectro visible), el controlador de vuelo, y dos marcadores ArUco colocados sobre la posición donde se quiere aterrizar.

El algoritmo se basa en detectar el marcador, obtener su posición respecto a la de la cámara del dron, direccionar al dron hacia el mismo, y, si el ángulo de visión del dron respecto al marcador es menor a cierto valor de umbral se da la orden de descender una altura determinada. Una vez alcanzado una altura mínima sobre el nivel del marcador, se cambia el modo de vuelo al dron a LAND (aterrizaje) completando el proceso de aterrizaje.

Existen dos casos donde el dron puede perder el marcador según su altura: si ésta es muy grande, puede que el software no sea capaz de detectar el marcador correctamente pues éste ocupa muy poco espacio dentro del cuadro de imagen. Por otro lado, también puede suceder que si la altura del dron respecto al marcador es muy baja, éste no pueda entrar completamente dentro del cuadro de imagen, impidiendo su detección por el software. Para mitigar esto es que se utilizan dos marcadores de tamaños distintos, uno de 27 cm de lado para facilitar la detección en el primer tramo del aterrizaje (a grandes alturas) y uno de 13 cm de lado para asegurar que a bajas alturas el mismo entre completamente dentro del cuadro de imagen. El algoritmo cambia cuál de los marcadores debe buscar según la altura del dron respecto al marcador, o su altura de vuelo si no detectó ninguno aún.

El algoritmo también posee técnicas de control para que el dron pueda volver a alinearse con los marcadores si en algún momento pierde visión de los mismos.

Como se mencionó, el algoritmo consta de tres partes claves: la detección del marcador, la estimación de su pose, y finalmente, el loop de control del movimiento del dron. En lo que resta de este capítulo se entrará en detalle en cada una de estas partes, explicando el marco teórico necesario para comprender el software diseñado. Para más detalles sobre las librerías utilizadas, ver capítulo 5, y para una descripción del software implementado referirse al anexo A.3.

6.1. Marcadores ArUco

Los marcadores ArUco son marcadores fiduciarios o de referencia, los cuales permiten realizar la correspondencia entre la posición real de un objeto y la imagen 2D tomada del objeto por una cámara. Los mismos consisten en un cuadrado de borde negro compuesto en su interior por una matriz binaria la cual determina un identificador único para cada ArUco [22]. Existen distintos marcadores ArUco implementados en distintos diccionarios, en este caso se utilizaron dos marcadores: uno del diccionario "Original", de 6 bits, con el número identificador 72, y otro del diccionario "4X4", de 4 bits, con el número identificador 0. En la figura 6.1 se muestran ambos marcadores.

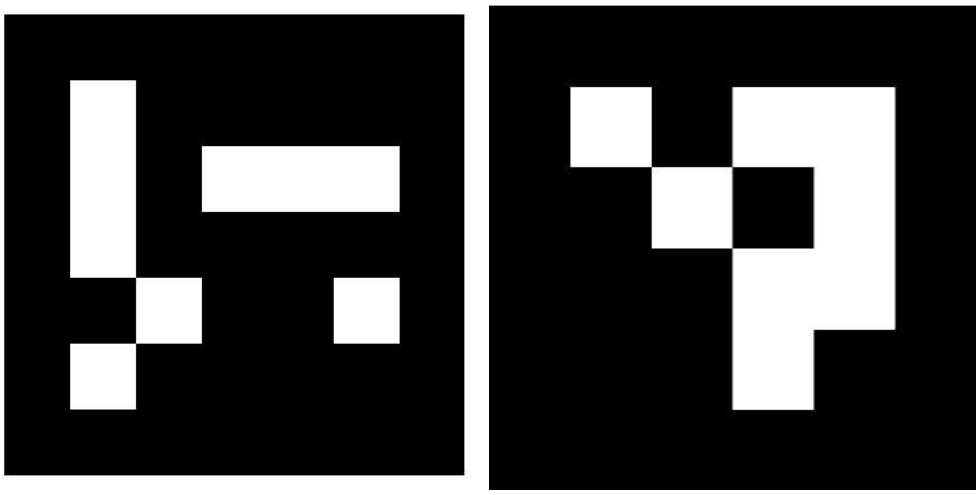

Figura 6.1: Marcadores ArUco: a la izquierda diccionario ArUco Original, 6 bits, identificador 72. A la derecha diccionario ArUco 4x4, 4 bits, identificador 0.

Para la correcta detección de los marcadores es necesario calibrar la cámara a utilizar, es decir, se deben obtener dos parámetros: la matriz de parámetros intrínsecos de la cámara (suponiendo un modelo de cámara estenopeica), y los coeficientes de distorsión debido al lente. En A.5.6 se incluye un instructivo para utilizar el script de calibración.

6.2. Detección de marcador

La detección de marcadores se realiza en dos etapas: primero, se hace un análisis de la imagen para hallar todas las formas similares a cuadrados en la imagen, y luego se analiza la matriz de píxeles interna para determinar si representan un número identificador y si éste es el buscado [22].

La primera etapa es de búsqueda de candidatos, donde se buscan cuadrados o formas similares a cuadrados en la imagen. Esto es realizado mediante una técnica de procesamiento de imágenes llamada umbral adaptativo (*adaptive thresholding*), en la cual cada valor de un píxel es comparado contra un valor umbral dado, y en caso de ser mayor se cambia el color a negro, de lo contrario se cambia a blanco. Este valor de umbral no es fijo (eso es la técnica de umbral simple), sino que es calculado para cada píxel a partir de los valores de los píxeles que lo rodean en un pequeño entorno, de esta forma se logran mejores resultados para imágenes con variaciones de intensidad de luz [23]. Luego se detectan los contornos[1] en la imagen y, aquellos que no sean convexos o no aproximen a un cuadrado (o un cuadrado distorsionado por perspectiva) son descartados. También se aplican otros tipos de filtrado para descartar contornos muy pegados entre sí, o de tamaños muy grandes o muy pequeños [22].

Luego de la etapa de detección de candidatos se procede a analizar cada uno y determinar si realmente contienen un marcador ArUco en su interior, esto se realiza analizando la matriz de bits interna de cada uno. Para obtener esta matriz se realizan transformaciones de forma de eliminar la distorsión generada por la perspectiva de la cámara, obteniendo los marcadores en su forma canónica[2], luego se aplican técnicas de umbral para obtener los bits negros y blancos. Se procede entonces a separar la imagen en celdas según parámetros como el tamaño del marcador y la cantidad de bits del mismo, y luego se determina si cada bit es negro o blanco contando la cantidad de píxeles de cada color hay dentro de la misma. Finalmente se analizan los bits para ver si el marcador pertenece a un cierto diccionario y su número identificador correspondiente [22].

En la figura 6.2 se pueden ver los marcadores detectados a partir de una imagen. En la figura 6.3 se muestran los candidatos rechazados en la misma imagen.

[1] Se entiende por contorno a una curva continua cerrada que envuelve un grupo de píxeles de valor intensidad de color similar.

[2] Se entiende por marcadores en su forma canónica como los marcadores listos para imprimir, i.e. sin distorsión de perspectiva.

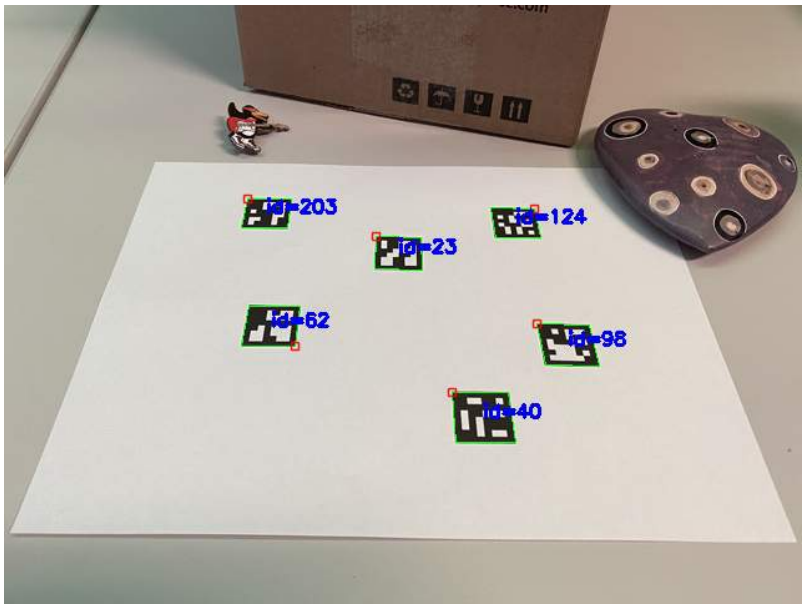

Figura 6.2: Ejemplo de marcadores detectados en una imagen. Se dibuja el contorno, la esquina superior izquierda y el número de identificación de cada uno. Imagen obtenida de **OpenCV**.

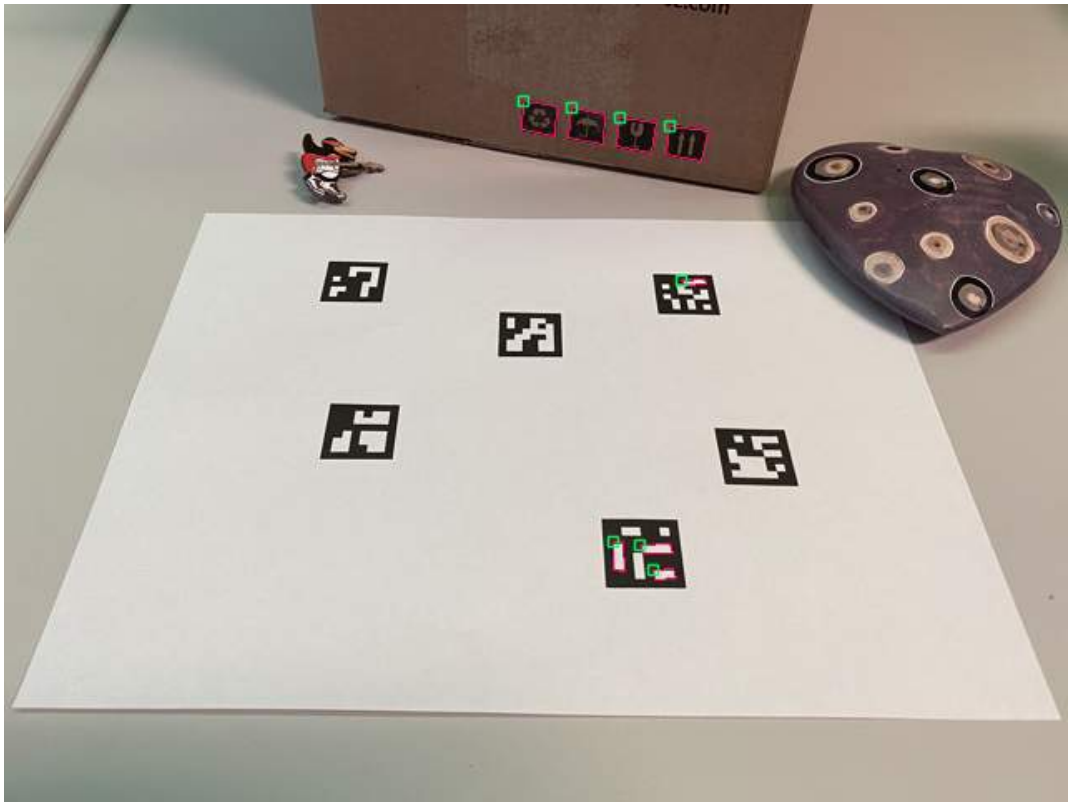

Figura 6.3: Ejemplo de marcadores rechazados en la imagen. Imagen obtenida de **OpenCV**.

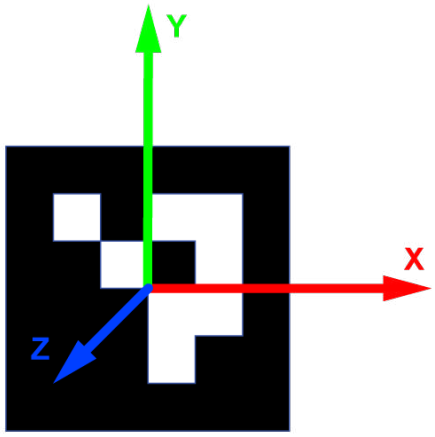

Figura 6.4: Marcador ArUco con su sistema de ejes superpuestos.

Todo esto es realizado por la función de la librería de *OpenCV*, *detectMarkers* la cual devuelve las esquinas de los marcadores y el número de identificación de los mismos [22].

6.3. Estimación de pose

Luego de detectado el marcador se quiere obtener la posición del marcador respecto a la cámara, lo cual es conocido como estimación de pose. Para esto se necesita un sistema de coordenadas solidario a la cámara y un sistema de coordenadas solidario al marcador. En la figura 6.4 se observa el sistema de coordenadas solidario al marcador.

Es necesario entonces obtener la posición del sistema del marcador respecto al sistema de la cámara. Para esto primero se verá cómo convertir la posición de un punto en el mundo (es decir, un punto tridimensional en un sistema de coordenadas global) hacia el plano imagen de la cámara (no confundir el plano imagen de la cámara, el cual tiene su sistema de coordenadas bidimensional, con el sistema de coordenadas de la cámara, el cual es tridimensional).

Aplicando el modelo de cámara estenopeica[3] se puede hallar la la proyección de un punto de un objeto tridimensional (de coordenadas (X,Y,Z)) en el plano imagen (con coordenadas (x,y)):

$$x = -f\frac{X}{Z}, \;\; y = -f\frac{Y}{Z} \tag{6.1}$$

Donde f corresponde a la distancia focal de la cámara. Observar los signos negativos en las igualdades, los cuales representan que la imagen proyectada está al revés en el plano imagen, cuando se sitúa éste detrás del centro de la cámara, o centro de proyección.

[3]El mismo es explicado en mayor detalle en **??**

Esto es fácil de corregir con una rotación de 180º en ambos ejes, o, equivalentemente, multiplicando por -1 las expresiones anteriores.

Pasando a forma matricial se tiene que:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{6.2}$$

Donde, el vector referido al punto proyectado en el plano imagen es expresado en coordenadas homogéneas, por lo que se agrega la coordenada w. Las coordenadas homogéneas son muy útiles a la hora de representar transformaciones como matrices, ya que permiten convertir transformaciones afines en formas matriciales simples, permitiendo componerlas con otras transformaciones lineales. Se entrará más en detalle sobre esto en breve, pero primero se detallará cómo realizar una representación en coordenadas homogéneas de un punto.

Sea el punto bidimensional de coordenadas $(2, 1)$, se puede expresar el mismo mediante coordenadas homogéneas como $(2, 1, 1)$, es decir, agregando una tercera coordenada de valor unidad. También se puede cambiar el valor de la tercera coordenada, pero se debe ajustar proporcionalmente el valor de las dos primeras, es decir, el vector $(2, 1)$ admite una representación en coordenadas homogéneas del tipo $(4, 2, 2)$, $(6, 3, 3)$, etc., en general, las coordenadas homogéneas permiten mapear el vector $(x/w, y/w)$ como (x, y, w).

La igualdad dada por 6.2 es válida solamente cuando coinciden los centros de los sistemas de coordenadas de la cámara y del sistema de coordenadas del plano imagen (también llamado punto principal). Como se puede ver en la figura 6.5, esto no sucede, ya que para procesamiento de imágenes por computadora, la convención es que el centro del sistema de coordenadas del plano imagen sea la esquina superior izquierda. Se tienen entonces tres sistemas de coordenadas: un sistema de coordenadas global, un sistema de coordenadas respecto a la cámara, y un sistema de coordenadas para el plano imagen.

Se debe entonces tomar en cuenta el efecto del desplazamiento del centro del punto principal respecto centro del sistema de coordenadas de la cámara. Suponiendo que éste está en las coordenadas (c_x, c_y), se tiene entonces que:

$$x = f\frac{X}{Z} + c_x, \;\; y = f\frac{Y}{Z} + c_y \tag{6.3}$$

Expresándolo matricialmente, con coordenadas homogéneas:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{6.4}$$

A la matriz que multiplica al punto de coordenadas (X, Y, Z) se le llama matriz de la cámara, o matriz de parámetros intrínsecos y está compuesta por la distancia focal, y la ubicación del punto principal de la misma (todos estos parámetros son propios de cada cámara, de ahí el nombre matriz de parámetros intrínsecos).

Resta entonces saber convertir la ubicación de un punto en el sistema de coordenadas global hacia el sistema de coordenadas de la cámara. Es directo ver que con una traslación y una rotación es posible convertir un punto de un sistema de coordenadas al otro. Entonces, conociendo la distancia entre los centros de ambos sistemas de coordenadas, y los ángulos entre los ejes de uno respecto al otro se puede aplicar la siguiente transformación:

$$x = R(X - C) \tag{6.5}$$

Donde X es el punto a convertir, en el sistema de coordenadas global, x es su proyección en el plano imagen, C es la diferencia de posición entre los centros de los sistemas de coordenadas, y R es la transformación de rotación entre los ejes de los sistemas de coordenadas.

Se puede observar que la transformación 6.5 no es lineal sino afín, ya que tiene un término independiente no nulo, debido a la traslación. Aplicando coordenadas homogéneas a dicha transformación, agregando una dimensión al vector X, es posible entonces expresar 6.5 como una transformación lineal, con su matriz asociada:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{6.6}$$

La segunda matriz que se introduce, correspondiente a la transformación de rotación y traslación es llamada la matriz de parámetros extrínsecos, y está compuesta por la matriz de rotación y el vector de traslación correspondientes a las transformaciones anteriormente mencionadas.

Es entonces posible, conociendo estas matrices, obtener la posición del sistema de coordenadas del marcador respecto a la cámara, estimando así la posición del marcador para el aterrizaje de precisión.

Debe mencionarse que la ecuación 6.6, no incluye los efectos de distorsión que son producidos por los lentes de las cámaras, ya que el modelo de cámara estenopeica en el que se basa no incluye dicho efecto. Sin embargo, las funciones utilizadas para la estimación de pose sí tienen en cuenta los lentes y sus efectos, pero no se entrará en los detalles teóricos de los mismos. Para más información referirse a [22].

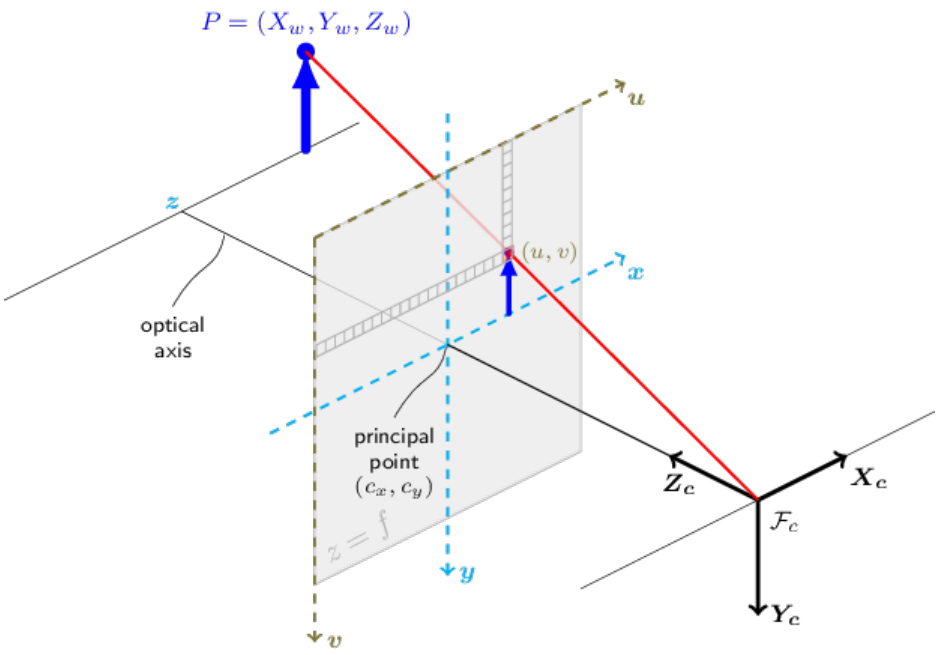

Figura 6.5: Modelo de cámara estenopeica y conversión de punto tridimensional a espacio bidimensional en plano imagen. Imagen obtenida de **OpenCV**.

6.4. Descripción del algoritmo

El primer paso del algoritmo es la detección del marcador, lo cual incluye hallar la posición de las cuatro esquinas, y su identificador. En caso de no detectar un marcador, se incrementa un timer, y al llegar a 5 segundos sin detección de marcador se indica al dron que suba un 20 % de su altura actual hasta un máximo de 8 metro. Esto se hace para que, en caso de que el dron pierda visión del marcador, pueda volver a encontrarlo, ya que al incrementar la altura vertical del mismo, también se incrementa el campo de visión de la cámara sobre el suelo.

En caso de sí encontrar un marcador se comienza el proceso de estimación de la posición del marcador respecto a la cámara, para lo cual se utiliza la función *estimatePoseSingleMarkers()* del módulo *aruco* de *OpenCV*, la cual recibe como parámetro las cuatro esquinas del marcador, el identificador, la matriz de la cámara y los coeficientes de distorsión, y devuelve los vectores de traslación y rotación, los cuales permiten obtener las coordenadas del marcador respecto a la cámara (x,y,z), en centímetros y los ángulos del marcador respecto a la cámara [22]. Cabe mencionar que de estos parámetros obtenidos sólo se utilizan los primeros, siendo los ángulos utilizados durante el testeo para verificar el correcto funcionamiento de la detección, pero podrían ser de utilidad para futuras implementaciones de este algoritmo, permitiendo controlar aún más el movimiento del dron, ajustando su ángulo de *yaw*.

Una vez obtenida la posición del marcador respecto a la cámara, se deben convertir

dichas coordenadas hacia el sistema de coordenadas del dron, los cuales están directamente relacionados con la ubicación del controlador de vuelo. En la figura 6.6 se presenta un esquema de la orientación de los ejes en una aeronave [4] [24]. Cabe destacar que el controlador de vuelo posee una flecha en su carcasa indicando la dirección del vector x, así como también la posee el módulo de GPS utilizado.

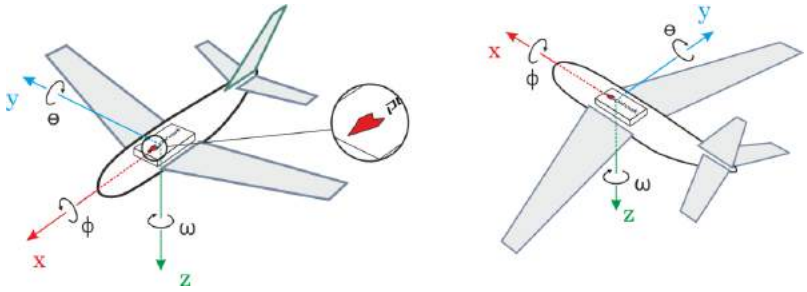

Figura 6.6: Orientación de los ejes en una aeronave (compatible con un cuadricóptero).

El sistema de coordenadas de un marcador ArUco se puede observar en la figura 6.4, y el sistema de coordenadas de una cámara es según la figura 6.7

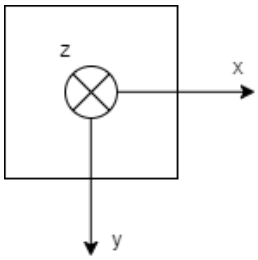

Figura 6.7: Esquema de la orientación de la cámara, vista desde atrás, es decir, la cámara apunta según el eje z positivo, hacia abajo.

Entonces, el cambio de coordenadas a realizar es:

$$X_{DRON} = -Y_{CAMARA} \tag{6.7}$$

$$Y_{DRON} = X_{CAMARA} \tag{6.8}$$

Una vez realizado el cambio de coordenadas, se obtienen las coordenadas NED del dron y se procede a evaluar la altura medida por el dron. Si esta es mayor a 5 metros, se asigna su valor a la variable auxiliar Z; de lo contrario, se le asigna el valor de distancia vertical desde la cámara al marcador. Se hace esta distinción porque el controlador de vuelo estima su altura mediante un barómetro, el cual es menos preciso a bajas alturas.

Se procede a revisar si se cumple alguna de las condiciones de cambio de marcador. Como ya se mencionó al comienzo del capítulo, se utilizan dos marcadores de distinto

[4] Aunque se muestre un avión, los ejes son iguales para el caso de un cuadricóptero.

tamaño, diccionario y número identificador para asegurar la correcta detección de los mismos a pequeñas y grandes alturas. Mediante proceso experimental se determinaron los valores de las alturas de cambio, siendo la primera de 1.5 metros (cuando se cambia del marcador grande al pequeño) y la otra de 2 metros (cuando se cambia del marcador pequeño al grande). Se utilizan dos distancias ya que se encontró que al tener una sola distancia de cambio, cerca de la misma y con las variaciones en las medidas de altura, se realizaban muchos "rebotes" o cambios entre los dos marcadores. De esta manera, al iniciar el algoritmo se busca el marcador grande, una vez que se pasa el umbral de los 1.5 metros, se pasa a buscar el marcador pequeño, y en caso que el dron pierda el marcador y deba subir, superando los 2 metros, se vuelve a buscar el marcador grande.

Una vez finalizado el cambio de marcador, se procede a calcular el ángulo de visión del dron respecto al marcador; esto se puede interpretar como un cono invertido sobre el marcador ArUco donde la generatriz del mismo coincide con el eje z del marcador. Este ángulo de visión funciona como una medida del error en la alineación del dron sobre el marcador en el suelo, y es lo que se utiliza para distinguir entre los dos posibles movimientos del dron.

Luego, se revisa un timer interno, el cual guarda el tiempo transcurrido desde el último comando de movimiento enviado al dron por el algoritmo, en caso de ser este tiempo menor al inverso de la frecuencia de refresco elegida para el algoritmo, se descartan los valores obtenidos y se comienza el algoritmo nuevamente, comenzando por buscar el marcador. En caso de que sí haya pasado un período desde el último comando, se obtienen las coordenadas NED del dron, y, utilizando las distancias del marcador hacia el dron, se procede a hallar las coordenadas NED del marcador.

Se limita la frecuencia de refresco del algoritmo para no saturar al controlador de vuelo con comandos, especialmente dado que el comando utilizado para comandar el dron a la posición del marcador es no bloqueante, es decir, que una vez llamada la función, se puede continuar ejecutando código, y además los comandos de movimiento del dron sobre-escriben a los anteriores. Entonces, una tasa de refresco muy alta implica que cada comando interrumpe al anterior, realizando movimientos muy breves; por otro lado, una tasa de refresco muy baja supone movimientos menos precisos (debido a los posibles errores de precisión en el cálculo de coordenadas, el comando de movimiento y los sensores). Entonces, mediante pruebas experimentales se logra hallar que el valor de frecuencia de 1.5 Hz es adecuado para lograr movimientos precisos pero manteniendo una velocidad adecuada del aterrizaje.

Se verifica luego que el ángulo de visión del dron sobre el marcador sea menor que un valor umbral (8^o para cuando se utiliza el marcador de mayor tamaño, y 10^o para cuando se utiliza el más pequeño), y, en caso positivo, se indica al dron descender una altura fija; de lo contrario, se le indica al dron ir hacia las coordenadas del marcador manteniendo la altura.

Finalmente, se evalúa si la altura del dron es menor al umbral de altura para aterrizar, en caso positivo, se realiza el cambio de modo de vuelo a LAND (aterrizaje) y el dron

completa el aterrizaje, finalizando el algoritmo; en caso contrario, vuelve a comenzar el loop.

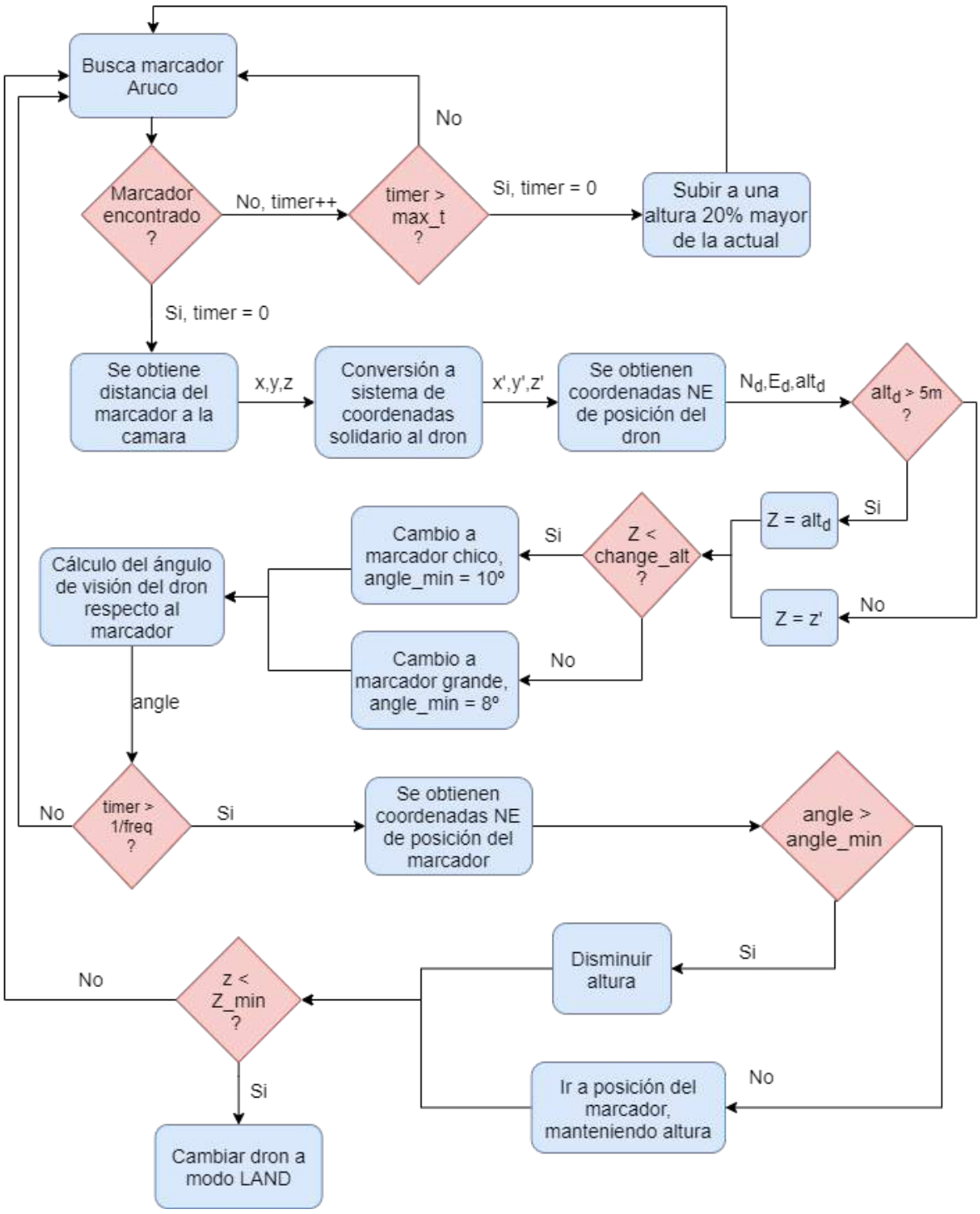

Figura 6.8: Diagrama de bloques del algoritmo de aterrizaje de precisión implementado.

Capitulo 7

Sistemas de comunicación

En el presente capítulo se detallan los sistemas de comunicación utilizados en el proyecto. Termodron III consta de tres entidades que se comunican entre sí: el dron (en particular, la computadora a bordo), la base (la computadora dentro de la misma), y la interfaz de usuario, como nexo entre el usuario y el dron.

Se tienen líneas de comunicación bidireccionales entre la interfaz de usuario y la base, y entre la base y el dron, las cuales utilizan correo electrónico, dada su robustez, y facilidad de uso. Se tiene una línea de comunicación unidireccional desde la base hacia el dron para el envío de los mensajes de corrección de GPS mediante el protocolo MAVlink a través de radiofrecuencia. Se posee también una línea de comunicación entre las tres entidades del sistema, mediante Amazon Web Services. Ésta última es opcional y puede activarse mediante software.

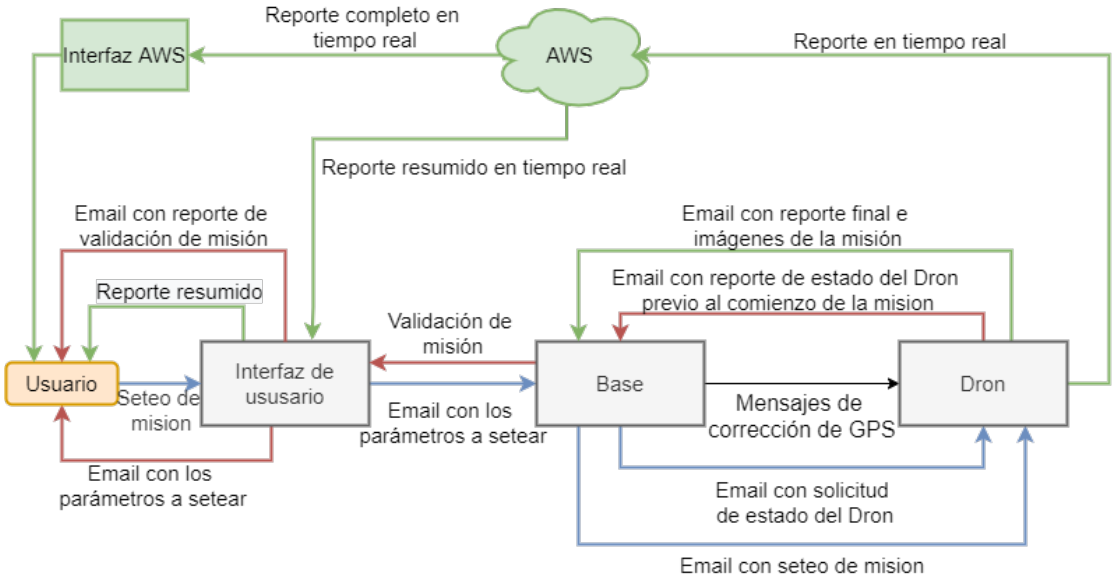

Figura 7.1: Diagrama de funcional del sistema de comunicación.

En la figura 7.1 se presenta el diagrama funcional de la comunicación entre el usuario, interfaz, base y dron.

7.1. Comunicación Usuario - Base: interfaz gráfica

La comunicación entre el usuario y la base se realizó mediante la implementación de una interfaz de usuario, la cual fue diseñada utilizando el lenguaje de programación Python y en particular la librería *tkinter*. Esta librería, la cual viene por defecto con la instalación de Python, es especifica para la creación de una interfaz gráfica, permitiendo incorporar funciones previamente implementadas a distintos botones, así como crear secciones para entrada de texto, ventanas de selección de opciones, entre muchas otras funcionalidades. Para más información referirse a [42].

Para la realización de una misión se comienza mediante la configuración de la misma por parte del usuario vía la interfaz gráfica de usuario, esto incluye la selección de los puntos geográficos a recorrer, la elección de realizar o no un recorrido de un área dada, el modo de funcionamiento de la cámara térmica, y la selección de direcciones de correo electrónico a donde se enviarán los reportes del dron. Una vez hecho esto, la interfaz se encarga de la comunicación con la base mediante correos electrónicos, enviando los parámetros de misión actuales, y eventualmente recibiendo correos de confirmación de la misión, de comienzo de la misma y de su fin.

La interfaz, en su búsqueda de sencillez, cuenta únicamente con tres bloques; el primero es un bloque de acceso y seguridad, luego un bloque principal y por último un bloque de seteo de misión.

7.1.1. Bloque de acceso

Se crea el bloque de acceso (el cual puede ser observado en la figura 7.2) con el fin de limitar la comunicación entre usuario y base solamente al personal autorizado. Luego de ingresar las credenciales se accede al bloque principal.

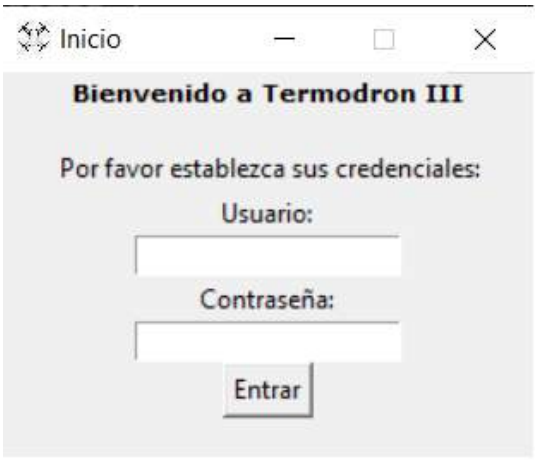

Figura 7.2: Bloque de acceso a la interfaz de usuario.

7.1.2. Bloque principal

El bloque principal se accede directamente luego de ingresar las credenciales en el bloque de acceso. En la figura 7.3 se puede observar este bloque.

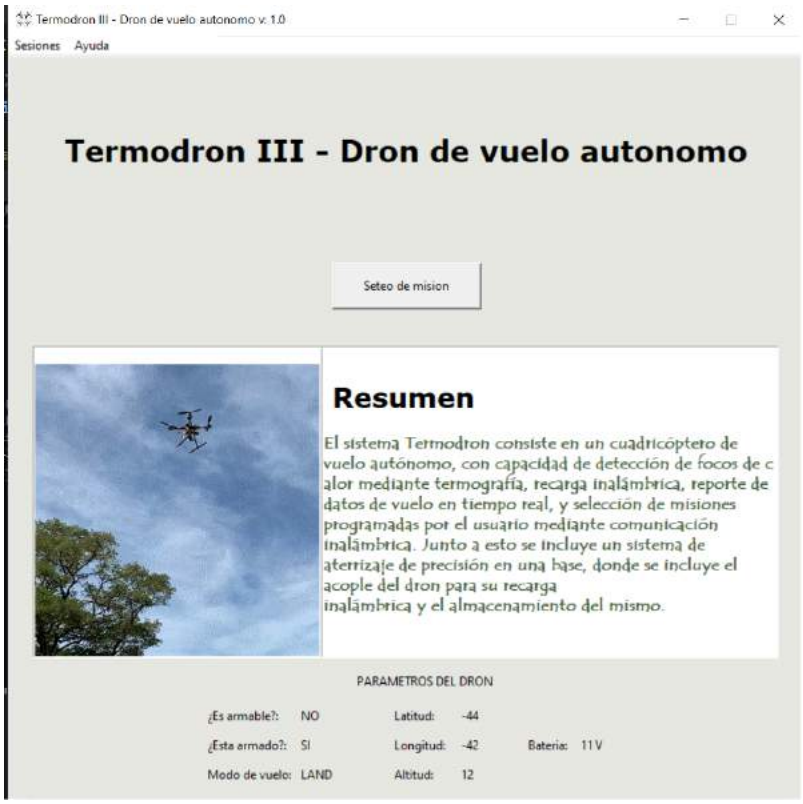

Figura 7.3: Bloque de principal de la interfaz de usuario.

En la región superior se tiene un botón con el cual se accede a la configuración de misiones. En la región media de la ventana se tiene un breve resumen acerca del proyecto Termodron, y en la región inferior se tiene un reporte en tiempo real de algunos parámetros de interés del dron, los cuales, como ya se mencionó, son actualizados mediante AWS, por lo que en caso de no estar activado este módulo opcional, no se mostrarán valores. En caso de activarlo, los parámetros que se reportan en la interfaz son los siguientes:

1. Latitud
2. Longitud
3. Altitud
4. Estado de batería
5. Modo de vuelo
6. Estado del dron: si se se puede armar o no
7. Estado del dron: si está armado o no

Cabe destacar que estos parámetros son a modo de resumen y si se desea tener más información sobre el estado del dron se debe recurrir al uso de la interfaz de AWS.

7.1.3. Bloque de seteo de misión

Este es el bloque más importante de la interfaz de usuario, mediante el mismo se logra la comunicación del usuario con la base, y la configuración de la misión. El bloque se puede observar en la figura 7.4.

A continuación se listan los parámetros de la misión que se pueden configurar en este bloque y se describen brevemente los mismos:

- Funcionalidad de la cámara térmica: se tienen tres posibilidades a elegir: búsqueda de cuerpos calientes (se buscan temperaturas entre 30 °C y 45 °C), búsqueda de focos de calor (se buscan temperaturas superiores a 100 °C), o desactivada.
- Coordenadas a recorrer: se debe cargar un archivo .csv con un formato adecuado indicando las coordenadas (latitud, longitud y altitud) a recorrer (se recomienda que formen un polígono regular), como se muestra en las figuras 7.5, 7.7. Luego, se abre una nueva ventana en la cual el usuario puede elegir el modo en el que se recorren los mismos: recorrido lineal de los puntos ingresados, o barriendo el área dentro del perímetro especificado. En caso de elegirse el modo de barrido de un área, la interfaz calcula las coordenadas del interior del área por donde el dron pasará, utilizando el algoritmo implementado en [25].

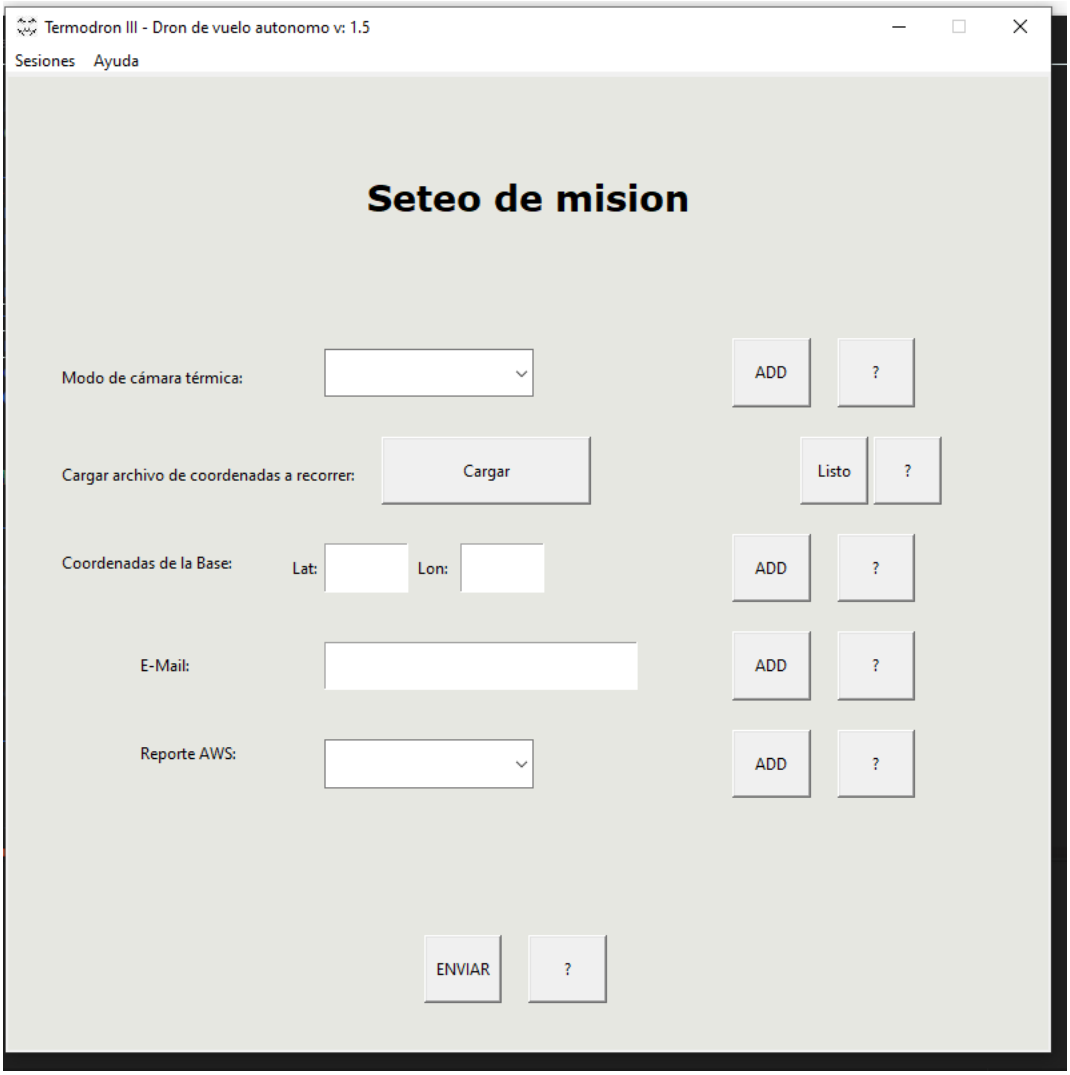

Figura 7.4: Bloque de seteo de misión en la interfaz de usuario, en donde se observan los distintos parámetros configurables.

- Coordenadas de la base: se debe ingresar las coordenadas de la base (latitud y longitud) con el fin de tener la ubicación a donde regresar y realizar el aterrizaje de precisión. Se recomienda conocer la posición de la base con gran precisión y no cambiar este parámetro a menos que se mueva la misma.
  Se asume la altura de la base como la altura a nivel del mar.

- E-mail: se pueden ingresar las direcciones de correo en las cuales se desean recibir los reportes luego de finalizada la misión.

- Reporte AWS: se elige activar o no el reporte en tiempo real por Amazon Web Services. Si no se activa el mismo, no se mostrará el reporte resumido en la interfaz gráfica.

El ingreso de puntos geográficos o *waypoints* es de gran utilidad para cargar recorridos que se usan de forma frecuente. Para esto, en el bloque de seteo de misiones se debe seleccionar el botón de "Sesiones" en la barra superior y luego "Cargar configuración desde archivo", o también se puede presionar el botón "Cargar" en la misma ventana, como se observa en la figura 7.4, luego se abrirá una nueva ventana para elegir el archivo .csv a cargar. Esto se observa en la figura 7.5.

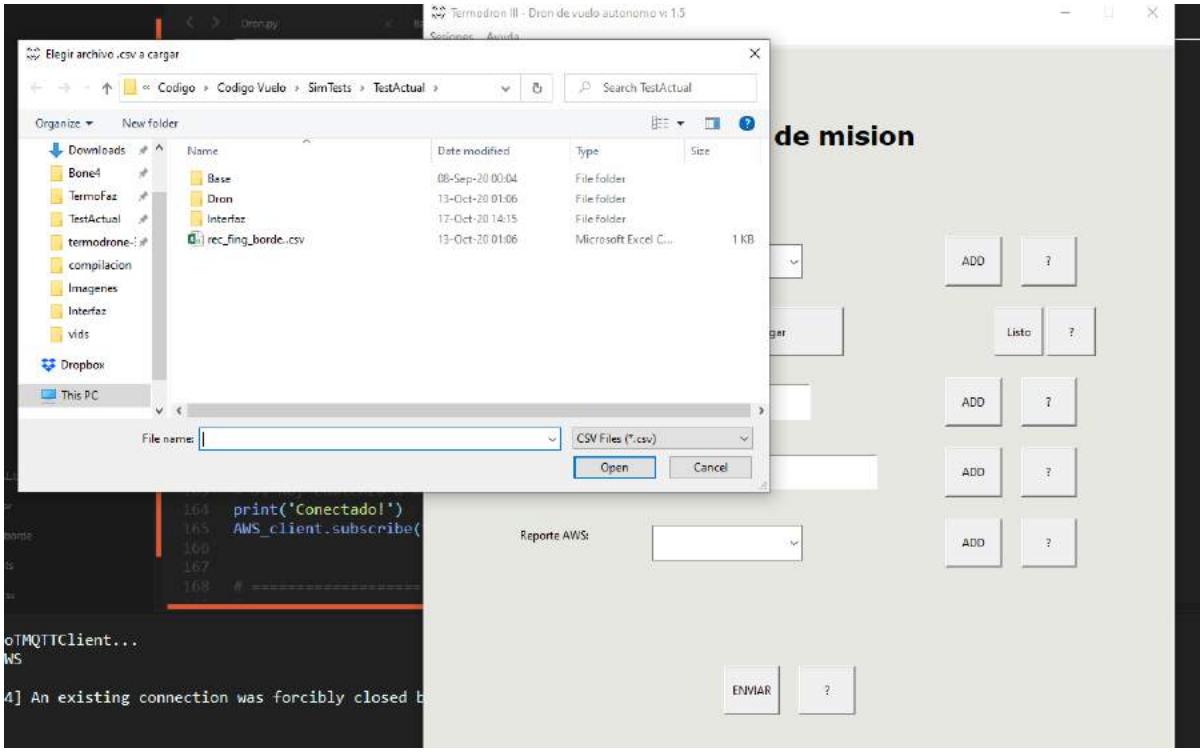

Figura 7.5: Bloque de seteo de misones, es posible cargar los puntos geográficos a recorrer desde un archivo .csv al clickear el botón "Cargar". Se abre una nueva ventana la cual permite elegir el archivo .csv.

El archivo que se carga debe ser de formato .csv (delimitado por comas) y debe contener tres columnas, latitud, longitud y altitud, tal como se muestra en la figura 7.6.

	A	B	C	D
1	-34.9194	-56.1665	5	
2	-34.9194	-56.1664	5	
3	-34.9195	-56.1664	5	
4	-34.9194	-56.1664	5	
5	-34.9194	-56.1665	5	
6				

Figura 7.6: Ejemplo del formato del archivo a cargar con las coordenadas a recorrer.

Una vez elegido el archivo, se debe presionar el botón “Listo”, el cual abrirá una nueva ventana donde se puede elegir el modo de recorrido de los mismos: como un perímetro (es decir, se recorren en el orden que fueron ingresados) o realizar un barrido del área al perímetro ingresado (ver figura 7.7). Para este último los puntos elegidos deben formar un polígono convexo, siendo el primero y el último punto iguales. Se puede especificar la cantidad mínima de puntos a recorrer durante el barrido, al agregar más puntos se logran líneas con menos separación entre ellas. En caso de no especificar un número se usa quince como la cantidad por defecto; este parámetro no es relevante para el primer modo de recorrido. Se debe presionar el botón “Enter” para confirmar la elección y se desplegará una ventana que muestra los puntos ingresados y, en caso de hacer un barrido de área, mostrará el recorrido generado en rojo, como se observa en la figura 7.8.

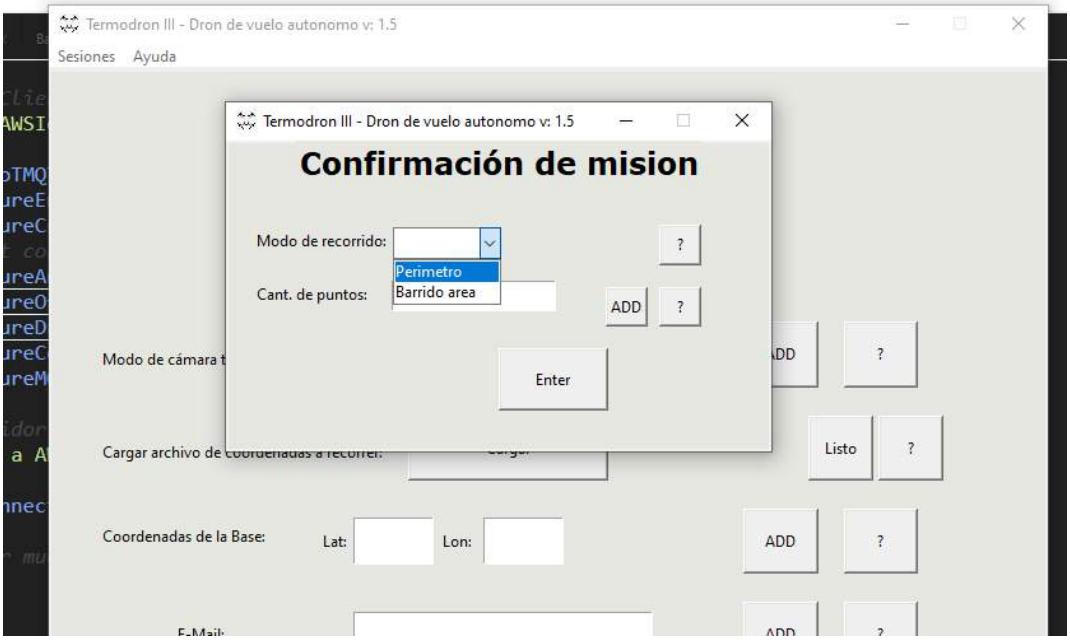

Figura 7.7: Luego de cargar el archivo .csv con los puntos a recorrer se elige el modo de recorrido: perímetro (se recorren en orden secuencial) o barrido de área, el cual genera los puntos adicionales para recorrer en el interior del perímetro especificado.

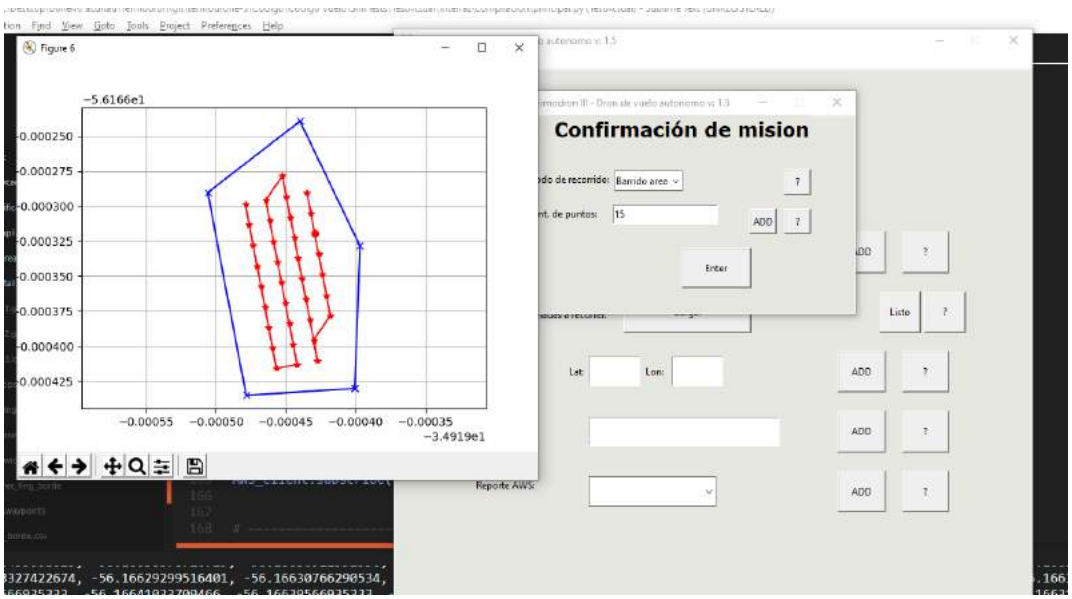

Figura 7.8: Ejemplo del modo barrido de área: el usuario carga un archivo .csv que especifica los puntos del perímetro (azul) y el algoritmo genera el barrido del área interior al mismo (rojo).

7.2. Base - Dron

Una vez recibida la misión, la base se encarga de la comunicación con el dron mediante correos electrónicos. Primero le solicita al dron un reporte de estado, y si el nivel de batería reportado es menor al 95 %, se activa la recarga inalámbrica[1], y una vez terminada ésta, se vuelve a pedir el reporte del dron. Si el nivel de batería del dron es adecuado y se encuentra en un estado armable[2] se envían los parámetros de misión hacia el mismo. Luego, se envía confirmación de la misión elegida a la interfaz y se espera recibir confirmación del comienzo de misión por parte del dron. Finalmente, la base queda en un estado de espera hasta recibir un correo de misión terminada, luego de la cual cerrará la cubierta de la base, y esperará recibir los logs y fotos del vuelo, y luego un reporte del estado del dron, donde, analizará nuevamente el nivel de batería para activar o no la recarga inalámbrica.

Análogamente, el dron comenzará en un estado de espera, hasta recibir un correo solicitando un reporte de estado, luego de enviarlo continúa esperando hasta recibir un correo con parámetros de misión. Se cargan los mismos y una vez listo para comenzar el vuelo, envía una notificación de comienzo a la base y procede a realizar la misión. Una vez comenzada la misión, si se solicitó el reporte en tiempo real por parte del dron, éste lo reportará mediante el uso de AWS, teniendo en la interfaz gráfica un reporte resumido, y

[1]Las funcionalidades que refieren a la recarga inalámbrica o la cubierta de la base no se implementaron, pero se incluye en la teoría del funcionamiento del sistema actual.

[2]Es decir, que puede realizar un vuelo, con los motores listos para ser encendidos.

estando disponible el reporte completo en la consola de AWS IoT. Una vez terminada la misión y habiendo aterrizado y detenido sus motores, envía un correo de fin de misión a la base, seguido por los logs de vuelo, fotos y finalmente otro correo con su reporte de estado.

En las figuras 5.3, 5.4 de la sección 5 se puede ver en más detalle el funcionamiento de las entidades base y dron.

7.3. Amazon Web Services

Se mantiene la plataforma de Amazon Web Services para el reporte en tiempo real del dron, ya que es posible realizar actualizaciones del estado del dron mediante el uso de un documento de texto en formato JSON de máximo 8 kB de tamaño. De esta forma se optimiza el uso de los datos móviles del módulo LTE del dron durante el vuelo. Además, la plataforma AWS provee un API para poder comunicarse con la misma mediante código escrito en Python, de esta forma es que se crean scripts para reporte y monitoreo de entidades (en este caso, del dron).

El reporte en tiempo real es opcional y puede activarse o desactivarse mediante la interfaz gráfica. Además, dada la facilidad de implementación del sistema de AWS, y la utilización de archivos formato JSON para la configuración de misión mediante correos electrónicos, es posible cambiar el sistema implementado a una comunicación completamente por AWS si se desea. Esto no se realizó ya que se prefiere tener la robustez del sistema de correos electrónicos.

Para más detalles sobre los scripts implementados y el documento JSON utilizado para la comunicación y reporte de estados del dron, ver 5.2.6; para más detalles sobre cómo crear un usuario en AWS ver A.5.3.7.

Capitulo 8

Base

En este capítulo se entrará en detalle sobre la base[1] del sistema Termodron III, la cual cumple las funciones de monitoreo del dron, controla el hardware de la misma, así como las funciones de recarga inalámbrica y control de la cubierta protectora.

8.1. Funcionalidades

8.1.1. Asistencia al acople de bobinas

Debido a que el error en la posición del dron aterrizar no es nulo[2], se utiliza un sistema mecánico para el ajuste del dron hacia el centro de la base, para su posterior recarga.

Para esto se utilizan cuatro barras que acomodan el dron hacia el centro de la base. Estas barras son movidas de a pares con un sistema de correas y poleas por un motor NEMA 17 de tal manera que se cierran hacia el centro de la base.

Se utiliza un motor paso a paso NEMA 17, porque es posible controlar por pulsos eléctricos la cantidad de giros del mismo, de esta forma se puede lograr un control fino sobre la distancia a recorrer por las barras. Conociendo el tamaño exacto del dron, simplemente hay que mover las barras desde la posición inicial (abiertas) hasta la posición final (cerradas) en el centro de la base, a una distancia igual al tamaño de la base de las patas del dron. El sistema se puede observar en la figura 8.1.

[1] Llamada GCS o Estación de Control en la Tierra por Termodron I, y Estación de Monitoreo y Control por Termodron II

[2] Referirse a 9.9 para más detalles.

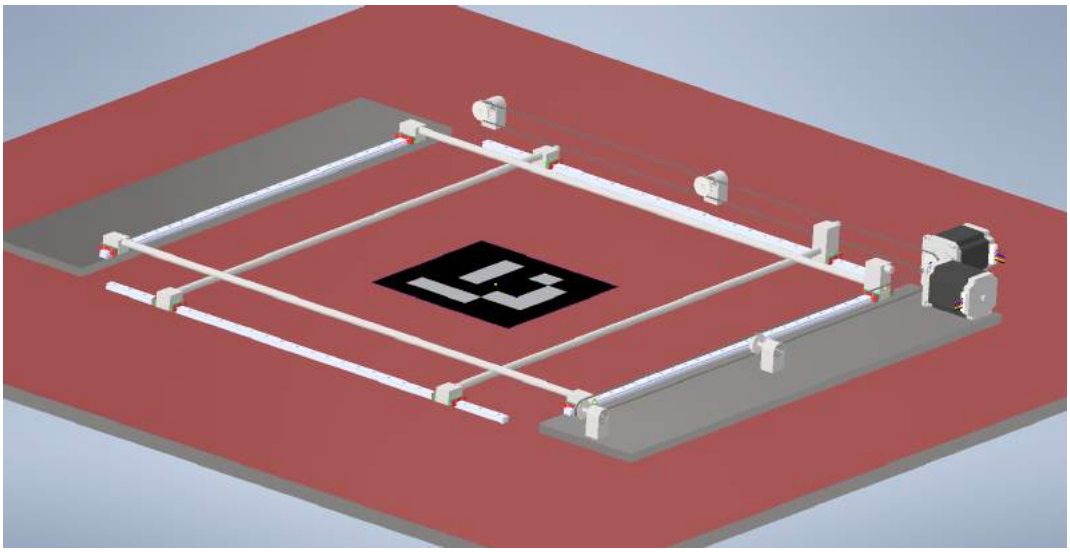

Figura 8.1: Sistema de corrección de posición del dron.

Como se observa en la figura 8.1, las barras ejercen fuerza sobre las patas del armazón del dron de a pares opuestos, por lo que es posible corregir errores en el ángulo de *yaw* del dron respecto al marcador de menos de 30º, en caso de que el dron no aterrice alineado con el marcador ArUco de la base. Aún así, para hacer que el sistema sea invariante por rotación, se puede colocar la bobina de recarga inalámbrica correspondiente al dron centrada entre las patas del mismo.

8.1.2. Recarga Inalámbrica

Para la recarga inalámbrica, como ya se mencionó anteriormente, se utiliza el mismo sistema de Termodron II, el cual fue descrito en el capítulo 3. Por más información referirse a [10]

8.1.3. Control de la cubierta

Debido a que el dron va a operar de forma autónoma todo el año, se tiene la necesidad de protegerlo contra el mal tiempo, ya sea lluvias, vientos, etc. Por este motivo se realiza una cubierta para cumplir con dicho motivo.

Se opta por diseñar una cubierta de dos piezas, con caída para poder evacuar el agua de forma eficiente. Estas dos piezas son movidas por un motor NEMA, al igual que el sistema de acople, es fácil controlar el motor para poder abrir y cerrar el sistema.

En la figura 8.2 se observa la cubierta cerrada, y en la figura 8.3 se la puede observar abierta

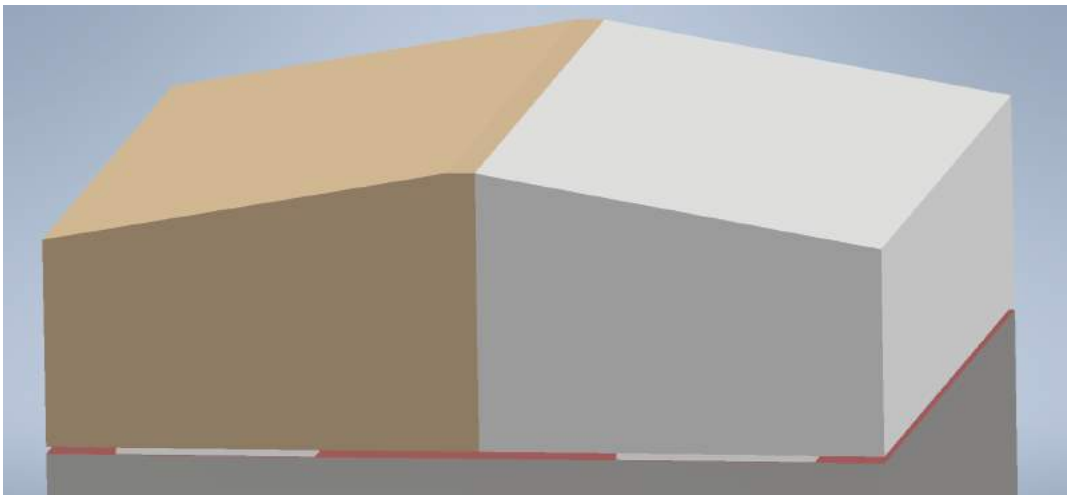

Figura 8.2: Base con cubierta cerrada.

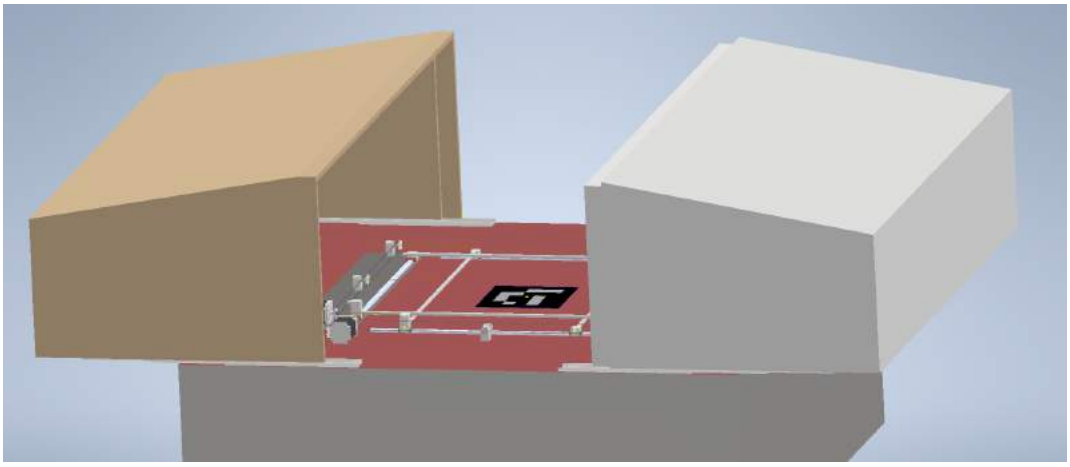

Figura 8.3: Base con cubierta abierta.

8.1.4. Comunicación

Dentro del hardware de la base se tiene una Raspberry Pi 3B+, la cual funcionará como computadora principal de la base, controlando el hardware de la misma así como las comunicaciones con la interfaz de usuario y con el dron. Éstas se realizarán mediante correos electrónicos, como se detalló en la sección 5.2.5. Para más detalles sobre las funciones de comunicación utilizadas, referirse al anexo A.3.

8.2. Modelo 3D

Como se mencionó anteriormente no se construyó la base pero sí se realizó un diseño a modo de prototipo. Para esto se utilizó el programa Autodesk Inventor.

Se comienza diseñando una caja para el almacenaje de la Raspberry Pi 3 (que servirá como computadora principal de la base), los componentes de la recarga inalámbrica, el módem LTE, así como una fuente para el suministro de energía para éstos. Para lograr una buena ventilación se le agregan dos ventiladores, uno para el ingreso y otro para la expulsión del aire.

Luego, pensando en la necesidad de proteger el sistema contra las inclemencias del tiempo, se agrega al diseño una cubierta, de forma de poder cerrar la misma cuando el dron no está en uso, y abrirla previo a un vuelo. Se opta por un diseño de dos piezas para alivianar el peso que deben mover los motores. Una de las tapas tiene un borde en desnivel, de forma de encastrar dentro de la otra, evitando así posibles grietas entre las tapas debido a la tolerancia de los componentes mecánicos que las mueven, y evitando también que entre agua hacia el resto del sistema. Estas piezas que componen la cubierta son montadas sobre un riel y movidas con un motor NEMA mediante un sistema de poleas de tal manera que efectúan un movimiento horizontal sobre la base. Todo esto se puede observar en las imágenes 8.3, 8.2.

En el modelo de la imagen 8.1 se pueden observar los motores NEMA 17, las poleas y las correas GT2, las cuales mueven las barras que desplazan el dron hacia el centro de la base, ayudando al acople de las bobinas de recarga del dron y la base. Se opta por utilizar poleas y correas del tipo GT2 ya que son las utilizadas por las impresoras 3D, por lo que se encuentra una gran variedad de tamaños, amplia disponibilidad en el mercado y mucho soporte online en cuanto a software y controladores.

8.3. Componentes

La elección de componentes se basó en los elegidos por Termodron II. Ya que la función de la base se mantiene, siendo las principales, el acople y posterior recarga del dron, la protección frente a las inclemencias climáticas y la validación y envió de misiones.

1. Motores paso a paso o Stepper, NEMA 17
2. Raspberry Pi
3. Antenas de telemetría 3DR 915 MHZ
4. Fuente AC/DC de 12V
5. Módulo transmisor de la recarga inalámbrica.

NEMA 17

El motor utilizado es un NEMA 17 de 1.2A a 4V, ángulo de paso 18° (200 pasos/revolución) y torque de 3.2Kg-cm. El motor debe ir conectado a un controlador A4988

que permite voltajes y corrientes de hasta 35V y 2A respectivamente.

A ambos componentes se los puede observar en las figuras 8.4 y 8.5

Figura 8.4: Driver A4988

Figura 8.5: Motor NEMA 17

Raspberry Pi

Se utiliza una Raspberry Pi 3 Modelo B+ con sistema operativo Raspbian en su última versión, con las librerías detalladas en el capítulo 5. Por más información sobre las características ver la Tabla 4.2

Antenas de telemetría 3DR 915 MHZ

Se utiliza una antena de telemetría para el envío de los mensajes de corrección del GPS diferencial de la base al dron. Se la puede observar en la figura 4.18.

Fuente AC/DC de 12V

Se utiliza una fuente regulada para poder suministrar energía a los distintos componentes de la base. Se agregará un regulador para los componentes que así lo requieran.

Módulo transmisor de la recarga inalámbrica

Como se mencionó anteriormente se utiliza el mismo sistema de transmisor de energía utilizado por Termodron II. Este proyecto no realizó cambios sobre el mismo. Por más información referirse a [10].

Capitulo 9

Pruebas realizadas

En el presente capítulo se detallan las diferentes pruebas realizadas durante la realización del proyecto. En este se presentan tanto pruebas de caracterización de componentes como pruebas de verificación de algoritmos.

9.1. Caracterización del conjunto motor y hélice

En esta sección se presentan las pruebas realizadas para la caracterización de tres pares motor-hélice con el fin de poder elegir el conjunto motor-hélice que mejor se ajusta a las necesidades del dron. En particular, interesa la relación empuje-consumo de los mismos, pues, como ya se mencionó, interesa mejorar la relación empuje/peso con el menor consumo posible, y con la limitante de mantener el tipo de batería a 3 celdas, debido al sistema de recarga inalámbrica. Para la realización de la prueba se usó como guía la prueba de caracterización de motores realizada por el grupo de proyecto de fin de carrera uQuad III [11].

Por último, se destaca que la prueba fue realizada en conjunto con el grupo de proyecto uQuad IV, e implementada en el laboratorio de control del Instituto de Ingeniería Eléctrica de la Facultad de Ingeniería de la Universidad de la República.

Objetivos

Caracterizar el comportamiento del conjunto motor y hélices, registrando los valores de empuje del motor, velocidad de la hélice, corriente, voltaje y ancho de pulso de la señal PWM.

Esta prueba se realizó durante la elección de los componentes, se tomaron medidas de peso de Termodron II, y, con los cambios en componentes se estimó que el peso de Termodron III en el peor caso sería de 2.8 Kg. Se busco entonces un conjunto motor-hélice que cumpla con las siguientes condiciones:

1. En operación normal entregará un empuje de 800 g a la menor corriente posible
2. Será capaz de entregar un empuje máximo que supere los 1200 g
3. Deberá funcionar con batería de 3 celdas (11.1 V)

Con esto se busca tener una relación entre el empuje de los motores y peso del dron (TWR) de 1.71. Se trabajará con tres motores distintos, estos son, Emax GT2215, Emax GT2820 y Emax MT3510, en donde se destaca que el motor GT2215 es el utilizado por TermoDron II [10].

Descripción del Experimento

El motor se monta en el centro de una barra metálica apoyada sobre una balanza en un extremo, y sobre un soporte fijo en el otro. La velocidad angular de los motores se controla variando el ancho del pulso de la señal PWM envidada a los ESC mediante un Arduino DUE entre 1000μs y 2000μs con pasos de 100μs. Se registraron los siguientes valores:

1. Medida de la balanza
2. Frecuencia con la que se interrumpe el haz de luz del conjunto emisor-receptor IR
3. Voltaje de la batería
4. Corriente por la batería

En las figuras 9.1 y 9.2 se presentan los diagramas mecánicos y eléctricos del experimento realizado. Como se observa en la figura 9.1 la medida de la balanza corresponde con la mitad del empuje que realiza el motor. Por otro lado, como la hélice corta el haz de luz del detector IR dos veces por ciclo, la verdadera frecuencia de la hélice es la mitad de la medida.

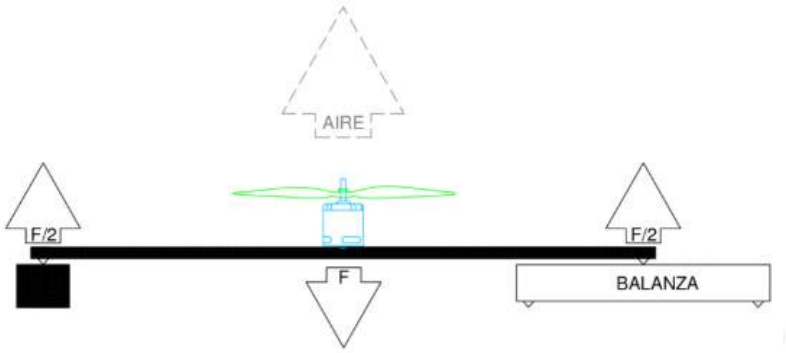

Figura 9.1: Diagrama mecánico de la prueba de caracterización de motores [11].

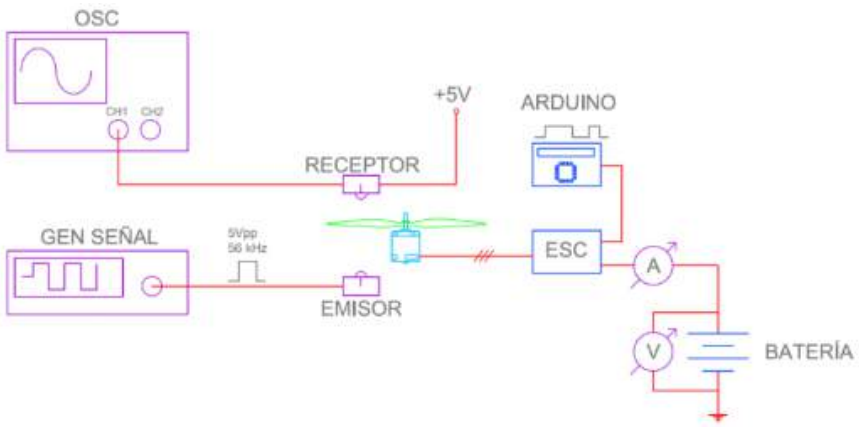

Figura 9.2: Diagrama eléctrico de la prueba de caracterización de motores [11]

Componentes utilizados

Los componentes utilizados para el experimento fueron:

1. Motores: Emax GT2215, Emax GT2820, Emax MT3510
2. Batería: HRB-XF 5000mAh 3S
3. Hélices: NS Propellers Carbon Fiber 12x6, HobbyKing Multirotor Carbon Fiber Propeller 14x4.7, Bgning 10x4.7 Carbon Fiber Propellers
4. ESC: ESC 30A Simon
5. Balanza: Camry EK9320
6. Osciloscopio: Tektronix TBS1062
7. Fuente: Siglent SPD3303C
8. Generador de señales: Tektronix CFG250
9. Pinza Amperimetrica: Fluke 376.

Especificaciones de los motores

A continuación se presentan los datos de los motores que los fabricantes proveen. Estos fueron de utilidad para realizar una primera pre-selección dentro de la gran variedad de motores disponibles en plaza, obteniéndose los tres motores ya mencionados. En el cuadro 9.1 se presenta los datos de los fabricantes [13–15].

Característica \ Modelo	Emax GT2820	Emax MT3510	Emax GT2215/12
Cantidad de celdas	3-4	3-4	3
Kv (RPM/V)	850	600	905
RPM	7600	4840	7450
$I_{max}(A)$	26	15	15
Empuje$_{max}$ (g)	1650	1210	1000
Hélices recomendadas	1206	1447	1047
Dimensiones (mm×mm)	46.5×35	41.5×31.8	36.5×28.5
Peso (g)	140	102	70
Precio (US$)	23	32	16

Cuadro 9.1: Especificaciones de los motores de la pre-selección.

Resultados

Se presentan los resultados obtenidos al realizar la caracterización de cada motor con su hélice recomendada por el fabricante. Se presentan los cuadros con los datos crudos, los cuadros con los cálculos realizados y una serie de gráficas que fueron de interés para la decisión.

Emax GT2215 - Hélice 1047

Para el motor Emax GT2215 se recomiendan hélices 1047, es decir, de 10 pulgadas de largo, y 4.7 pulgadas de avance. Para el experimento se utilizaron hélices de fibra de carbono. En el cuadro 9.2 se presentan los datos crudos obtenidos al realizar la caracterización del motor, mientras que en la tabla 9.3 se presenta los cálculos realizados.

Ancho del pulso	Consumo (A)	Voltaje (V)	Balanza (g)	Frecuencia (Hz)
1000	0.03	12.21	0.0	0.0
1100	0.4	12.21	10	61.5
1200	0.9	12.19	54	93.7
1300	1.5	12.18	89	120
1400	2.0	12.15	120	139
1500	3.0	12.12	154	156
1600	4.0	12.07	200	177
1700	5.7	12.02	258	200
1800	7.8	11.95	319	221
1900	10.1	11.83	385	242
2000	9.9	11.75	370	240.5

Cuadro 9.2: Resultados obtenidos de la caracterización del motor Emax GT2215 con hélices 1047.

Observando la figura 9.3 se observa que el motor utilizado por Termodron II no logra satisfacer la necesidad del proyecto, ya que su empuje máximo es menor al valor necesario

Ancho del pulso	Potencia (VA)	Empuje (g)	RPM
1000	0.0	0	0
1100	4.9	20	1845
1200	10.9	108	2811
1300	18.3	178	3594
1400	24.3	240	4170
1500	36.4	308	4680
1600	48.3	400	5310
1700	68.5	516	6000
1800	93.2	638	6630
1900	119.5	770	7260
2000	116.3	740	7215

Cuadro 9.3: Cálculos realizados para el motor Emax GT2215 con hélices 1047 a partir de los resultados del cuadro 9.2.

de 1200 g.

A continuación se detallan curvas de interés para la decisión de motores y para la caracterización del conjunto motor-hélice, estas son: empuje en función de corriente, para verificar el avance del empuje a medida que incrementa la corriente, y para poder observar el punto de máximo consumo y empuje. La otra curva de interés es de empuje sobre potencia en función de la corriente consumida, y también la curva de potencia en función del consumo.

Emax GT2820 - Hélice 1260

En el cuadro 9.4 se presentan los datos crudos obtenidos al realizar la caracterización del motor Emax GT2820 con hélice 1260 (12 pulgadas de largo y 6 pulgadas de avance), mientras que en la tabla 9.5 se presentan los cálculos realizados.

Como se observa en la figura 9.5 el motor logra superar el empuje máximo de 1200 g, llegando hasta 1680 g bajo 26.6 A de consumo de corriente, superando el requisito de empuje máximo impuesto.

Emax MT3510

En el cuadro 9.6 se presentan los datos crudos obtenidos al realizar la caracterización del motor, mientras que en la tabla 9.7 se presenta los cálculos realizados.

Para el motor MT3510 se observa en la figura 9.7 que no se logra superar el empuje máximo de 1200g y para el caso de un empuje de 800g el motor consume una corriente aproximadamente 6.5A, consumiendo una potencia de 70 VA, lo cual se puede observar en la figura 9.6.

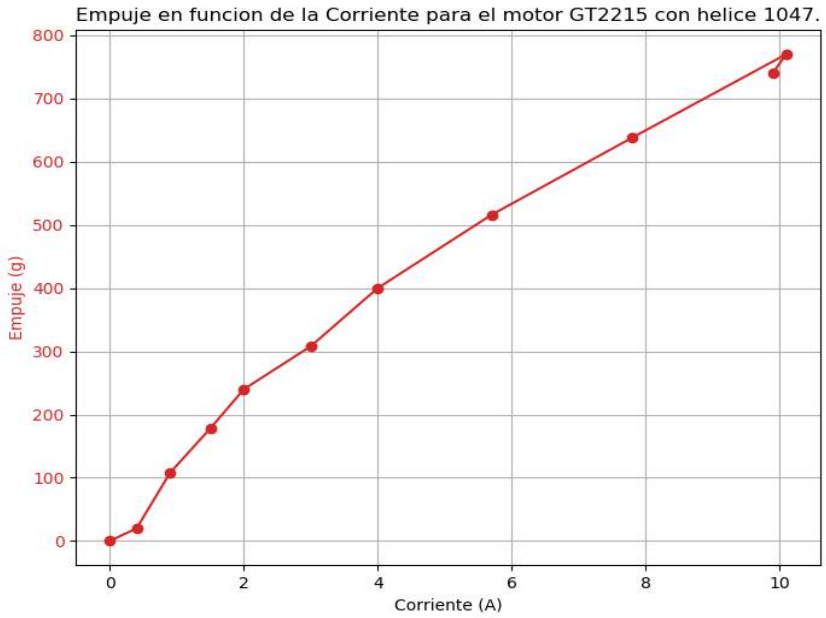

Figura 9.3: Empuje en función de la corriente para el motor GT2215 con hélice 1047.

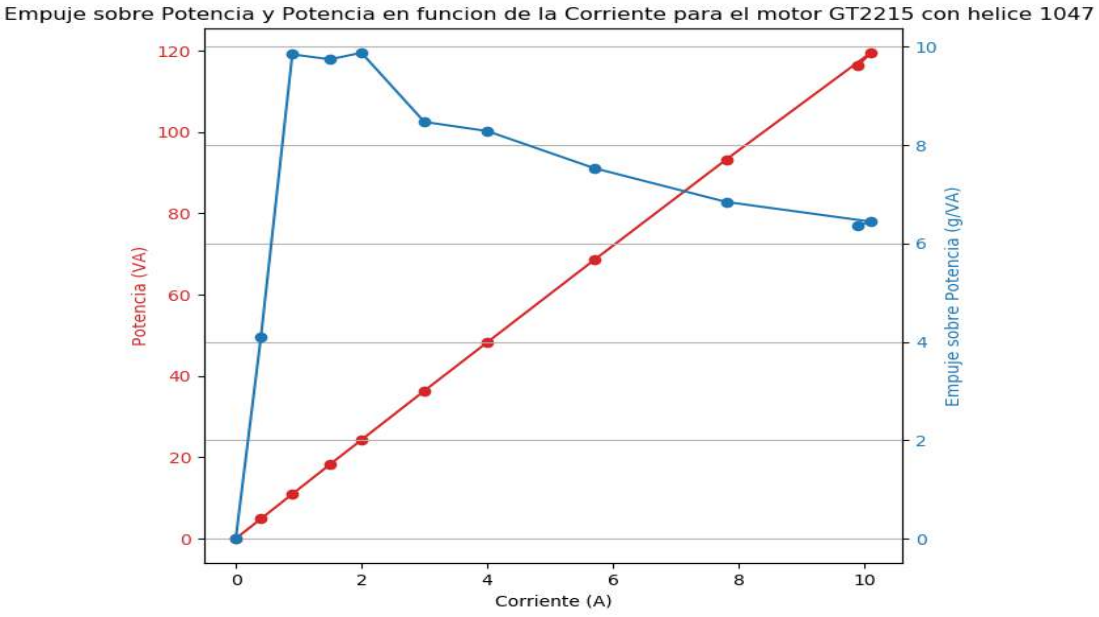

Figura 9.4: Empuje/Potencia y Potencia en función de la corriente para el motor GT2215 con hélice 1047.

Ancho del pulso	Consumo (A)	Voltaje (V)	Balanza (g)	Frecuencia (Hz)
1000	0.0	11.78	0	0.0
1100	0.8	11.75	52	53.6
1200	2.1	11.70	132	84.0
1300	3.7	11.63	210	105.6
1400	5.8	11.56	290	123.5
1500	7.7	11.47	360	136.4
1600	9.9	11.38	430	147.5
1700	13.4	11.27	538	164.2
1800	17.6	11.12	650	179.0
1850	19.8	11.04	700	184.5
1900	22.2	10.96	740	189.8
1950	24.0	10.89	790	196.3
2000	26.6	10.79	840	201.4

Cuadro 9.4: Resultados obtenidos de la caracterización del motor Emax GT2820 con hélice 1260.

Ancho del pulso	Potencia (VA)	Empuje (g)	RPM
1000	0.0	0	0
1100	9.4	104	1608
1200	24.6	264	2520
1300	43.0	420	3168
1400	67.1	580	3705
1500	88.3	720	4092
1600	112.7	860	4425
1700	151.0	1076	4926
1800	195.7	1300	5370
1850	218.6	1400	5535
1900	243.3	1480	5694
1950	261.4	1580	5889
2000	287.0	1680	6042

Cuadro 9.5: Cálculos realizados para el motor Emax GT2820 con hélice 1260 a partir de los resultados del cuadro 9.4.

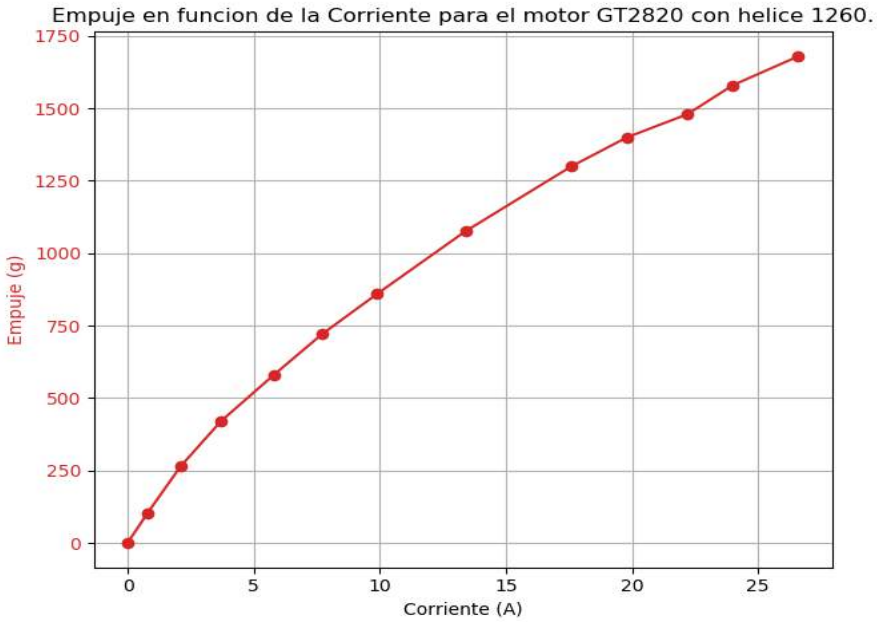

Figura 9.5: Empuje en función de la corriente para el motor GT2820 con hélice 1260.

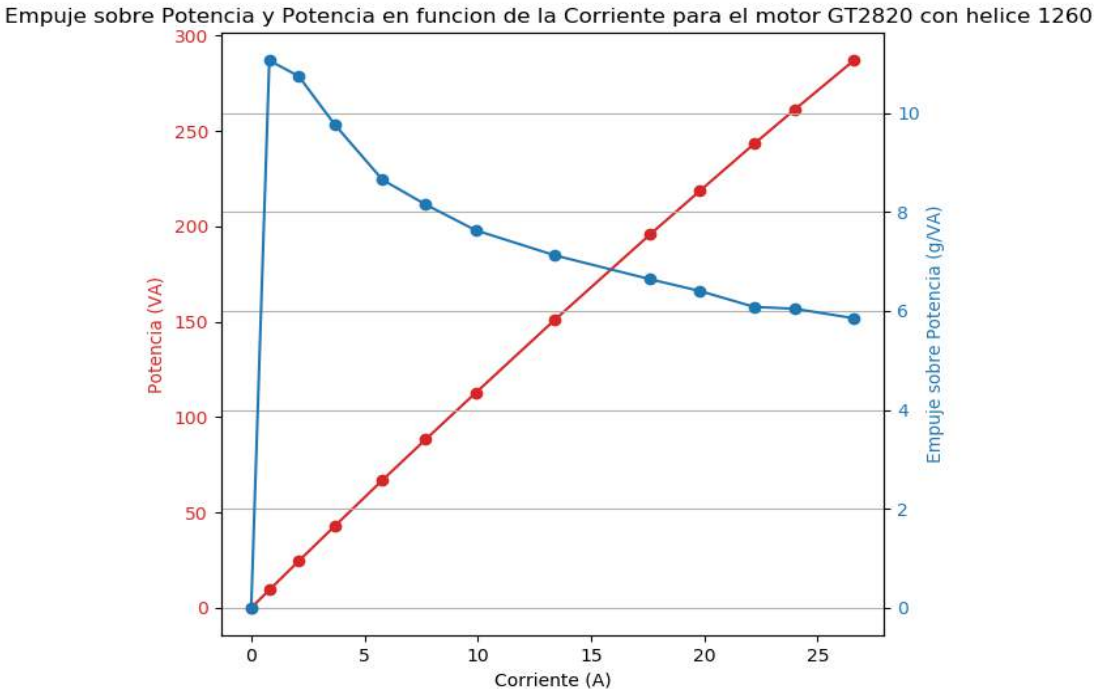

Figura 9.6: Empuje/Potencia y Potencia en función de la corriente para el motor GT2820 con hélice 1260.

Ancho del pulso	Consumo (A)	Voltaje (V)	Balanza (g)	Frecuencia (Hz)
1000	0	11.44	0	0
1100	0.2	11.44	14	28.8
1200	0.5	11.43	47	50.5
1300	0.9	11.42	84	67.9
1400	1.6	11.4	121	81.1
1500	2.6	11.37	180	97.4
1600	4.2	11.32	254	115.5
1700	6.1	11.27	340	131.9
1800	8.4	11.21	416	146.6
1850	9.7	11.14	466	153.4
1900	11	11.09	508	160.4

Cuadro 9.6: Resultados obtenidos de la caracterización del motor Emax MT3510.

Ancho del pulso	Potencia (VA)	Empuje (g)	RPM
1000	0	0	0
1100	2.288	28	864
1200	5.715	94	1515
1300	10.278	168	2037
1400	18.24	242	2433
1500	29.562	360	2922
1600	47.544	508	3465
1700	68.747	680	3957
1800	94.164	832	4398
1850	108.058	932	4602
1900	121.99	1016	4812

Cuadro 9.7: Cálculos realizados para el motor Emax MT3510 a partir de los resultados del cuadro 9.6.

Si bien este motor no logra los 1200g de empuje máximo no se lo descarta ya que para un empuje de 800g es el mas eficiente de los tres.

Conclusiones

Considerando los resultados obtenidos se concluyeron los siguientes puntos:

1. El motor GT2215 logra un empuje que en teoría sería suficiente para el peso estimado del dron de 2.8 kg, pero esto tiene dos inconvenientes: el primero es que dicho empuje es logrado al máximo de consumo y velocidad, es decir, el motor estaría siendo operado al límite de su capacidad, lo cual puede reducir la vida útil del mismo. El segundo es que el mismo no logra un TWR mayor a 1.7, pues su empuje máximo apenas es suficiente para el peso del dron. Por estas razones se descarta este motor.

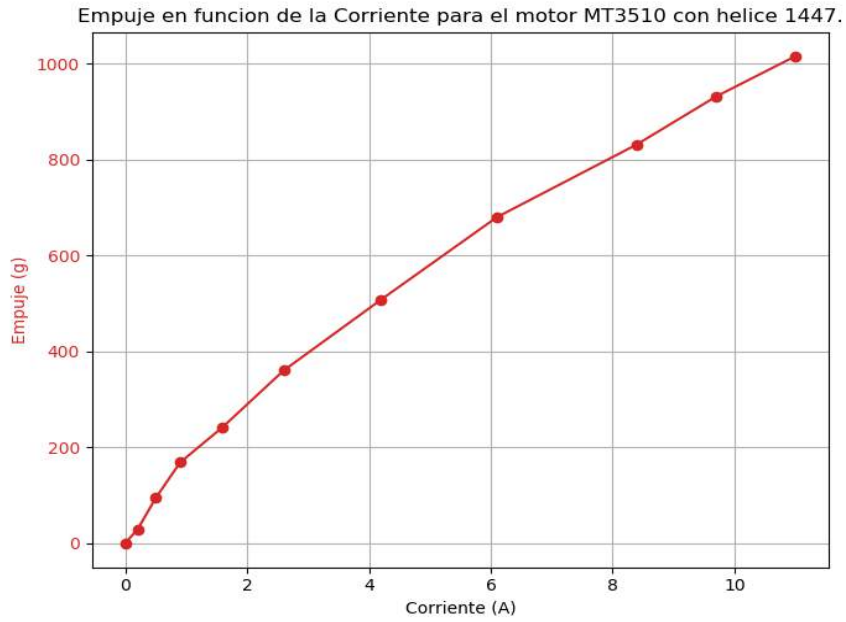

Figura 9.7: Empuje en función de la corriente para el motor MT3510 con hélice 1447.

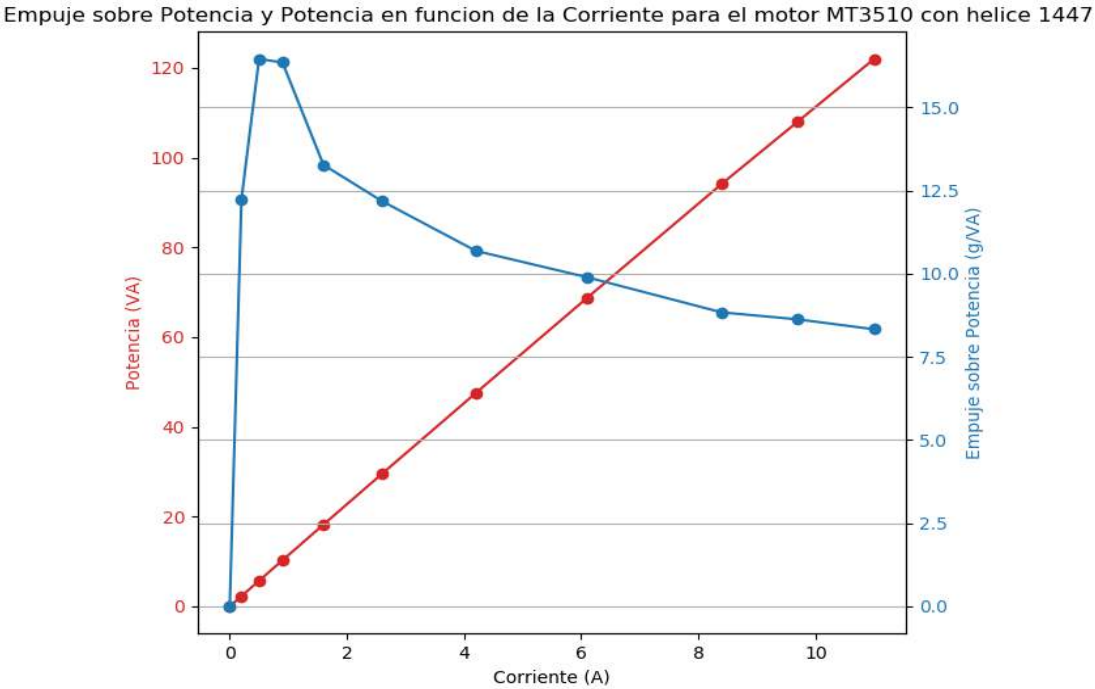

Figura 9.8: Empuje/Potencia y Potencia en función de la corriente para el motor MT3510 con hélice 1447.

2. El motor GT2820 alcanza a entregar el empuje necesario para levantar el dron, y además su empuje máximo puede mejorar el TWR a más de 1.7. Pero, se observa que no es eficiente, ya que para un empuje de 800 g consume 10 A. Un inconveniente que tiene este motor es su peso de 120 g, lo cual sumaría 160 g al peso del dron (contando los cuatro motores).

3. El motor MT3510 logra un empuje máximo que satisface la condición impuesta, y para un empuje de 832 g consume 8.4 A, además, su peso es de 100 g, sumando 80 g al peso del dron.

4. Comparando los pesos de cada motor se observa que el motor MT3510 es 40 g liviano que el GT2820, y también es más eficiente para un empuje de 800 g. Es por esto que se decide utilizar el motor MT3510 en el dron.

9.2. Caracterización de zona muerta de motores

En esta prueba se busca caracterizar la zona muerta[1] de los controladores de los motores, con el fin de poder configurar la velocidad de giro de los motores cuando están armados y la mínima velocidad de giro cuando están en vuelo.

Objetivos

Caracterizar la zona muerta de los motores con el fin de poder configurar la velocidad de giro de los motores cuando están armados y la minima velocidad de giro cuando están en vuelo.

Descripción del experimento

El experimento consiste en ir aumentando el porcentaje de *throttle*[2] que se le entrega a los motores, de a un motor por vez.
Esto se logra mediante el uso del programa Mission Planner[3], en el modo *Motor Test*, el cual se encuentra en *SETUP* y luego *Optional Hardware*.

Una vez que el motor comienza a girar se guarda el porcentaje utilizado y se pasa al siguiente motor. Luego se utiliza el mayor porcentaje de throttle de los cuatro motores,

[1]Es la zona donde el voltaje de entrada no causa que el motor comience a girar. Una vez superado el voltaje de umbral se tiene una relación lineal entre voltaje y velocidad.

[2]Palanca derecha del control remoto.

[3]Mission Planner es un software de control de misión, que permite monitorear, comunicarse y controlar un vehículo que utilice un controlador de vuelo con el firmware ArduPilot.

para la realización de los cálculos de los dos parámetros a configurar.

Según se explica en A.5.5, el parámetro *MOT_SPIN_ARM* se calcula como el máximo porcentaje de *throttle* obtenido de los motores mas un 2 % sobre 100.

El parámetro *MOT_SPIN_MIN* se calcula como el valor del parámetro *MOT_SPIN_ARM* más 0.3.

Componentes utilizados

Para la realización de la prueba, se contó con:

1. 4 motores Emax MT3510
2. Pixhawk Cube 2.1
3. Computadora con Mission Planner
4. ESC modelo Emax Simon Series 30A
5. Antena de telemetría 915 modelo 3DR Radio Telemetry Kit
6. Placa de distribución de voltaje modelo Hobby King
7. Batería HRB 10000mAh 25C

Resultados

Los resultados obtenidos para los cuatro motores, son los presentes en la tabla 9.8.

Numero de motor	% *Throttle*
1	5
2	5
3	5
4	5

Cuadro 9.8: Resultados obtenidos al realizar la prueba de aterrizaje.

Con los resultados obtenidos se calculan los parámetros:

$$MOT\_SPIN\_ARM = \frac{5+2}{100} = 0{,}07 \tag{9.1}$$

$$MOT\_SPIN\_MIN = 0{,}07 + 0{,}03 = 0{,}1 \tag{9.2}$$

Conclusiones

Se logra caracterizar la zona muerta de los ESC, con lo cual se configuran los parámetros de velocidad de giro cuando está armado y la velocidad mínima de giro cuando está en vuelo.

9.3. Aterrizaje de precisión: error

Objetivos

Verificar qué versión del algoritmo de aterrizaje de precisión implementado en la sección 5.2.1.1 provee los mejores resultados, es decir, con menor error. Se mide el error como la distancia desde el centro de la cámara hacia el centro del marcador ArUco. Se considerara un aterrizaje como exitoso si el error es menor a 30 cm.

Descripción del experimento

Se utiliza una laptop para conectar con la computadora a bordo del dron, y desde esta se ejecutará un *script* que realiza los siguientes pasos :

1. Establece una conexión con el controlador de vuelo (Pixhawk Cube) mediante USB.
2. Realiza los chequeos de seguridad necesarios previo al vuelo.
3. Se comanda al dron a volar hasta una cierta altura.
4. Comienza a ejecutar el algoritmo de aterrizaje de precisión.

Una vez finalizado el aterrizaje, y con los motores detenidos se mide el error obtenido.

Se utiliza el control TURNIGY TGY9X remoto para activar el modo LAND, o aterrizaje del dron, en caso de emergencias.

Se busca caracterizar el algoritmo para las mejores condiciones ambientales, es decir, con muy poco viento, de forma de evitar que estas perturbaciones alteren el movimiento del dron. Debido a la época en que se realizan estas pruebas, se vuelve muy difícil lograr coordinar vuelos en días específicos, siendo los mismos realizados los fines de semana, debido a los horarios del grupo. Es entonces que las pruebas se realizan para diferentes velocidades de viento, y en diferentes locaciones.

Componentes utilizados

A continuación se presentan los componentes utilizados para realizar las pruebas.

1. Cuadricóptero Termodron III
2. Computadora a bordo Odroid XU4
3. Cámara Intel RealSense D415
4. Marcador ArUco
5. Laptop
6. Control Remoto Turnigy TGY9X

En la figura se presenta el diagrama físico 9.9 del experimento.

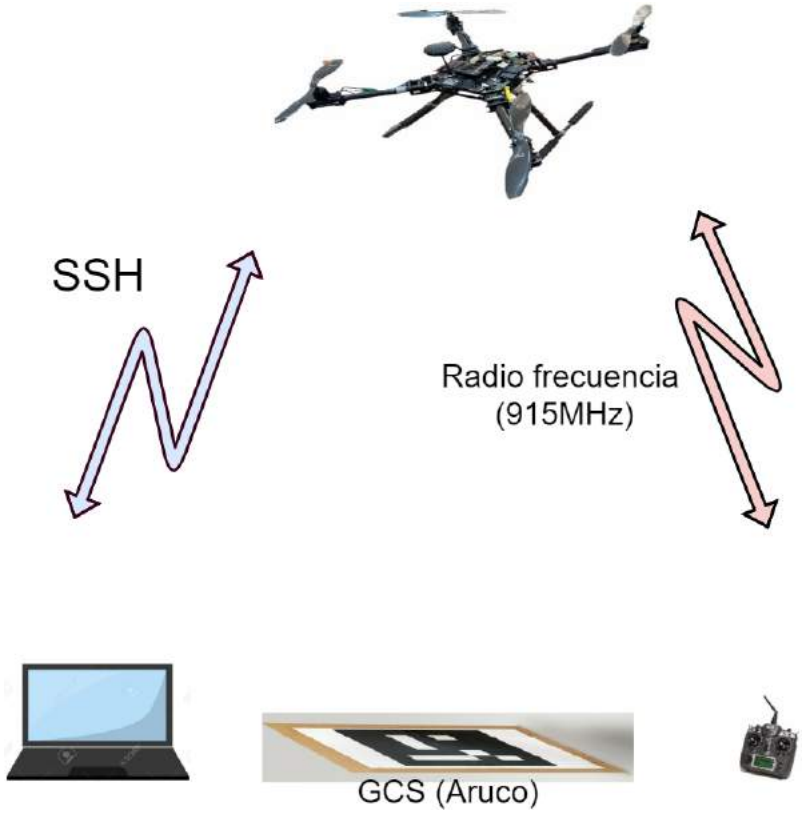

Figura 9.9: Diagrama físico de la prueba a realizar.

Resultados

El algoritmo fue variando a media que se realizaron los experimentos, iterando versiones anteriores según los resultados obtenidos. Los mismos se pueden observar en la tabla 9.9.

Numero de aterrizaje	Error (cm)	Tiempo de aterrizaje (mm:ss)	Cantidad de marcadores	Tamaño de marcadores (cm)	Frecuencia (Hz)	Resolución (píxeles)	Ángulo de descenso (°)	Altura de descenso (cm)
1	15	-	1	16	1	640x480	20	70
2	5	2:20	1	16	1	640x480	20	70
3	15	2:42	1	16	1	640x480	20	70
4	33	4:40	1	16	1	640x480	20	70
5	24	2:50	1	26.5	1	424x240	20	70
6	19	2:24	1	26.5	1	424x240	20	120
7	39	2:33	1	26.5	1	424x240	20	120
8	20	2:16	2	26.5	1	424x240	20	70
9	19	3:20	2	26.5	1	424x240	20	70
10	24	2:50	2	26,5 y 5	1	424x240	20	70
11	26	1:30	2	26,5 y 5	1	424x240	20	70
12	7	1:18	2	26,5 y 10	1	424x240	20	70
13	33	1:38	2	26,5 y 10	1	424x240	20	50
14	3	2:05	2	26,5 y 10	1	424x240	20	50
15	22	3:00	2	26,5 y 10	1	424x240	20	50
16	16	4:48	2	26,5 y 10	1	424x240	20 y 10	50
17	15	1:26	2	26,5 y 10	1	424x240	20 y 10	50
18	15	1:20	2	26,5 y 10	1	424x240	20 y 10	50
19	3	1:12	2	26,5 y 10	1	424x240	30 y 15	50
20	22	1:14	2	26,5 y 10	1	424x240	30 y 15	50
21	15	1:53	2	26,5 y 10	1	424x240	30 y 15	50

Cuadro 9.9: Resultados obtenidos al realizar la prueba de aterrizaje, mientras que se iba ajustando el algoritmo.

Conclusiones

Se logra validar el algoritmo de aterrizaje, realizado varios aterrizajes dentro del rango de error deseado, con unos pocos por fuera del mismo. A la misma vez, se observa que las mejoras introducidas al algoritmo logran mejoras de tiempo y/o precisión.

9.4. Aterrizaje de precisión: tiempo

Objetivos

Acelerar el tiempo que lleva completar un aterrizaje, intentando no degradar la precisión observada en el experimento anterior. Nuevamente se utiliza un proceso iterativo, donde se introducen mejoras al algoritmo a partir de los resultados obtenidos.

Descripción del experimento

Se utiliza una laptop para conectar con la computadora a bordo del dron, y desde esta se ejecutará un *script* que realiza los siguientes pasos :

1. Establece una conexión con el controlador de vuelo (Pixhawk Cube) mediante USB.
2. Realiza los chequeos de seguridad necesarios previo al vuelo.
3. Se comienza a filmar con la cámara, de forma de poder contar con respaldo gráfico de cada aterrizaje.

4. Se comanda al dron a volar hasta una cierta altura.

5. Comienza a ejecutar el algoritmo de aterrizaje de precisión.

Una vez finalizado el aterrizaje, y con los motores detenidos se mide el error obtenido.

Se utiliza el control TURNIGY TGY9X remoto para activar el modo LAND, o aterrizaje del dron, en caso de emergencias.

Se busca caracterizar el algoritmo para las mejores condiciones ambientales, es decir, con muy poco viento, de forma de evitar que estas perturbaciones alteren el movimiento del dron. Debido a la época en que se realizan estas pruebas, se vuelve muy difícil lograr coordinar vuelos en días específicos, siendo los mismos realizados los fines de semana, debido a los horarios del grupo. Es entonces que las pruebas se realizan para diferentes velocidades de viento, y en diferentes locaciones.

Componentes utilizados

A continuación se presentan los componentes utilizados para realizar las pruebas.

1. Cuadricóptero Termodron III
2. Computadora a bordo Odroid XU4
3. Cámara Intel RealSense D415
4. Marcador ArUco
5. Laptop
6. Control Remoto Turnigy TGY9X
7. Cámara y trípode

Resultados

En la tabla 9.10 se presentan los resultados del error al aterrizar y el tiempo de aterrizaje, además de los tamaños de los marcadores y la resolución de la cámara. En la tabla 9.11 se presentan los parámetros del algoritmo utilizados en cada prueba, como son las frecuencias, velocidades y ángulos de descenso (recordamos que se tienen pares de valores pues se tiene una etapa inicial del aterrizaje donde se busca el marcador de mayor tamaño y luego de cierta altura se pasa a buscar el marcador de menor tamaño. Para más detalles ver 6).

Numero de aterrizaje	Tiempo de aterrizaje (mm:ss)	Error (cm)	Cantidad de marcadores	Tamaño de marcador 1 (cm)	Tamaño de marcador 2 (cm)	Resolución (píxeles)
1	33	28	2	26.5	13	424x240
2	27	26	2	26.5	13	424x240
3	45	2	2	26.5	13	424x240
4	89	19	2	26.5	13	424x240
5	32	más de 30	2	26.5	13	424x240
6	27	8	2	26.5	13	424x240
7	25	más de 30	2	26.5	13	424x240
8	29	40	2	26.5	13	424x240
9	32	23	2	28.5	16	424x240
10	100	21	2	28.5	16	424x240

Cuadro 9.10: Resultados obtenidos al realizar la prueba de aterrizaje.

Numero de aterrizaje	Frecuencia 1 (Hz)	Frecuencia 2 (Hz)	Velocidad de descenso 1 (m/s)	Velocidad de descenso 2 (m/s)	Ángulo de descenso 1 (°)	Ángulo de descenso 2 (°)	Altura de descenso (cm)
1	1.5	1.5	0.4	0.1	10	15	80
2	1.5	1.5	0.4	0.1	10	15	80
3	1.5	1.5	0.4	0.1	10	15	80
4	1.5	1.5	0.4	0.1	10	15	80
5	1.5	1.5	0.4	0.1	10	15	80
6	1.5	1.5	0.4	0.1	10	15	80
7	1.5	1.5	0.4	0.1	10	15	80
8	1.5	1.5	0.4	0.1	10	15	80
9	1.5	1.5	0.3	0.1	10	10	75
10	1.5	1.5	0.3	0.1	10	10	75

Cuadro 9.11: Resultados obtenidos al realizar la prueba de aterrizaje.

Conclusiones

Se logra mejorar sustancialmente el tiempo de aterrizaje comparado con el experimento anterior. Se observa que se tiene mayor cantidad de aterrizajes con error fuera del rango aceptable, pero se atribuyen estos a días especialmente ventosos, necesitando una etapa final de pruebas con poco viento para poder verificar esta hipótesis.

9.5. Recorrido de un área

Objetivos

Se busca verificar el correcto funcionamiento del algoritmo de barrido de área utilizado y también verificar que el dron es capaz de recibir las coordenadas de dicho algoritmo, guardarlas en su memoria y recorrerlas de forma exitosa, para luego aterrizar.

Descripción del experimento

Habiendo elegido un área del estacionamiento del instituto de Ingeniería Eléctrica, se toman 5 puntos (pares de latitud, longitud) los cuales definen un polígono convexo y

se ingresan los mismos en el código de barrido de área, el cual produce (en este caso) 24 puntos (pares de latitud, longitud) dentro de la superficie interna de dicho polígono, describiendo un camino que el dron recorrerá, de forma de poder barrer dicha superficie.

Resultados

Se logra que el dron lea las listas de coordenadas a recorrer, las guarde en su memoria y las ejecute de forma autónoma, mediante el modo de vuelo AUTO. Una vez finalizado el recorrido, el dron aterriza en la posición del último punto.

En la figura 9.10 se pueden observar los cuatro puntos (pares de latitud y longitud) iniciales elegidos, los puntos que el dron debía recorrer (los obtenidos del algoritmo de recorrido de área), y el recorrido real (dos colores pues fueron dos vuelos realizados en el mismo recorrido) del dron, el cual fue descargado del controlador de vuelo luego de terminada la misión.

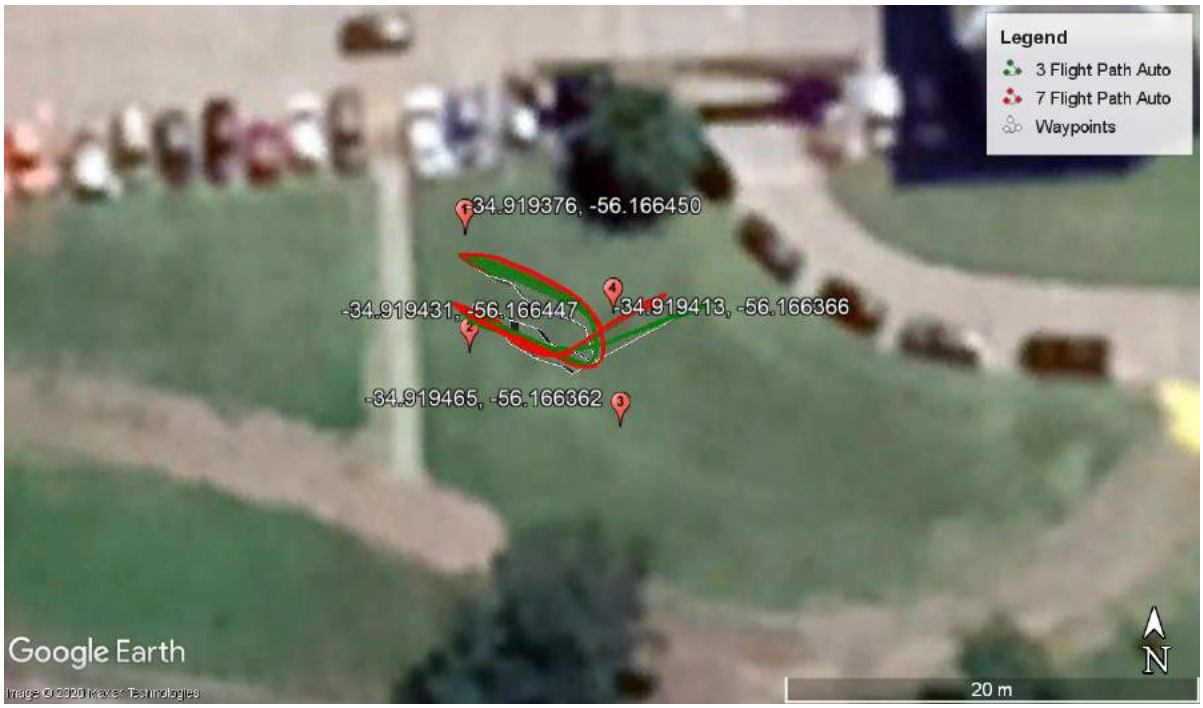

Figura 9.10: Log de vuelo del dron para la prueba de recorrido de área. Los puntos numerados son los que elige el usuario y definen el borde del área a recorrer. En gris se observan los waypoints que generó el algoritmo, y en rojo y verde la posición del dron al ejecutar el vuelo.

Se observan diferencias entre el recorrido del dron real y el supuesto debido a la precisión del GPS del dron (para esta prueba no se utilizó la corrección diferencial de posición), pero las mismas son mínimas, y se considera que el dron recorrió de forma exitosa la superficie.

Conclusiones

Se logra verificar el funcionamiento del script que genera los puntos de recorrido a partir de los puntos que describen el borde de la superficie a recorrer, así como también se logra verificar que el controlador de vuelo es capaz de recibir los puntos, guardarlos y ejecutar una misión pasando por cada uno de ellos con un error despreciable.

9.6. Cámara térmica: verificación de calibración

Objetivos

Se busca verificar el correcto funcionamiento de la cámara térmica, comparando las medidas de temperatura de un cuerpo caliente tomadas por la cámara contra las medidas obtenidas con un termómetro.

Descripción del experimento

Basados en el análisis realizado por Termodron II [10] se utiliza agua caliente como el cuerpo caliente cuya temperatura será medida ya que se comporta casi como un cuerpo negro, es decir, su emisividad es cercana a 1 [34]. Se coloca el agua caliente dentro de un vaso de bohemia y se introduce el termómetro dentro del mismo. Para evitar la estratificación de la temperatura del agua se agita la misma constantemente mientras se toman las medidas. Se coloca la cámara térmica en un soporte vertical de forma de poder apuntar el lente hacia la parte superior del vaso de bohemia, de forma de poder observar el agua dentro del vaso, y no el exterior del vaso, evitando medidas erróneas.

En la figura 9.11 se presenta un esquema del experimento.

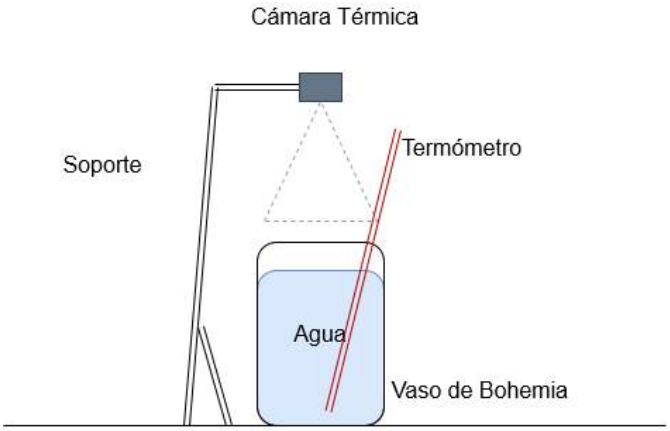

Figura 9.11: Diagrama del experimento de la cámara térmica.

Componentes utilizados

1. FLIR Lepton 3.5
2. Pure Thermal v2.0 Breakout Board
3. Termómetro de mercurio
4. Vaso de bohemia

Resultados

En la tabla 9.12 se pueden ver los resultados obtenidos en las medidas. En la figura 9.12 se pueden observar los puntos experimentales medidos, con el ajuste polinómico de primer orden[4] correspondiente, y con el valor teórico ideal de la pendiente, igual a 1, para el caso en que ambos instrumentos presentaran la misma medida.

Termómetro (ºC)	Cámara Térmica (ºC)	Termómetro (ºC)	Cámara Térmica (ºC)
89	86.0	53	50.0
88	85.3	50	48.1
75	72.0	48	46.5
71	67.2	46	44.1
68	65.3	44	42.0
65	63.3	42	40.0
63	59.2	40	38.5
61	58.0	38	37.2
60	56.1	35	35.0
57	53.9	34	33.5
55	52.4	33	32.1

Cuadro 9.12: Temperaturas medidas por la cámara térmica y por el termómetro.

El valor de los coeficientes del ajuste polinómico obtenido son $m = 0{,}94, n = 0{,}86$, por lo que se observa un valor de pendiente bastante aproximado a la unidad, pero un término independiente de valor no despreciable.

Conclusiones

Se verifica el correcto funcionamiento de la cámara térmica, y del código de Python que implementa la medida de temperatura, obteniéndose un nivel de error aceptable para el rango de temperaturas utilizado. Dado que la fucionalidad de la cámara será para detectar objetos en un rango de temperaturas dado (cuerpos calientes) o para detectar

[4] $y = m \times x + n$, siendo m la pendiente y n el término independiente

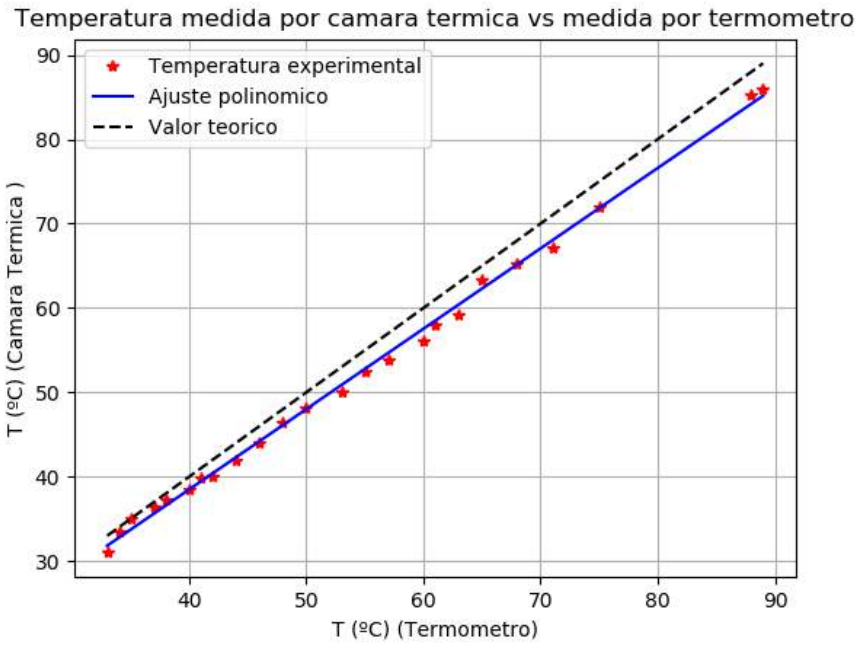

Figura 9.12: Comparación gráfica de temperatura medida por la cámara térmica en función de la temperatura medida por termómetro. Se representa también el ajuste polinómico de los datos experimentales y el valor teórico.

altas temperaturas (focos ígneos), se concluye que la cámara y el código probado son adecuados para la tarea.

9.7. Cámara térmica: verificación del algoritmo

Objetivos

Se busca verificar el algoritmo implementado para la detección de cuerpos calientes y foco de calor.

El algoritmo, como ya fue detallado en la sección 5.2.4, realiza el siguiente procedimiento al tomar una imagen, realiza un filtro en donde obtiene los píxeles con valores en el rango de temperatura deseado y luego realiza operaciones de OpenCV con el fin de obtener el contorno de los píxeles y quedarse con el de mayor área, para luego reportarlo.

Descripción del experimento

El experimento realizado para corroborar el correcto funcionamiento del algoritmo consta de realizar una serie de vuelos donde el dron pondrá en funcionamiento el algoritmo

y se caminará por debajo del dron, si el dron toma las fotografías del personal caminando se considerará exitoso el vuelo y por ende el algoritmo implementado.

Resultados

Se logra verificar el funcionamiento del algoritmo y la cámara térmica a bordo del dron, lográndose detectar cuerpos calientes, señalar los mismos en la imagen y guardarlas. En las figuras 9.13 y 9.14 se presentan dos imágenes obtenidas en los vuelos, una durante el día y otra en la noche. En las mismas se puede observar que si el suelo está a menor temperatura que el cuerpo, el suelo aparece en colores más oscuros y el cuerpo más claros, análogamente para la imagen tomada durante el día, en la cual el suelo tiene mayor temperatura que el cuerpo. Se observó también que si la temperatura del suelo (o del ambiente) es similar a la temperatura del cuerpo a detectar, es difícil distinguir entre ambos, y puede no funcionar adecuadamente.

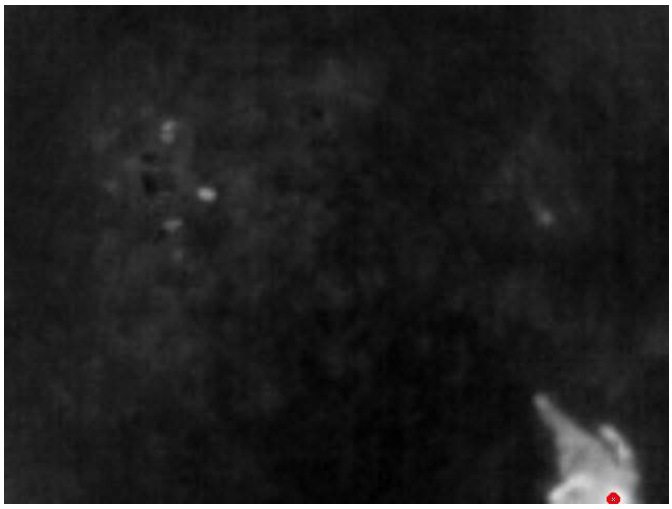

Figura 9.13: Imagen tomada en vuelo durante la noche utilizando el algoritmo implementado. Se observa que el cuerpo de la persona es detectado por la cámara, y el mismo está a mayor temperatura que el suelo (temperaturas mayores son colores más claros).

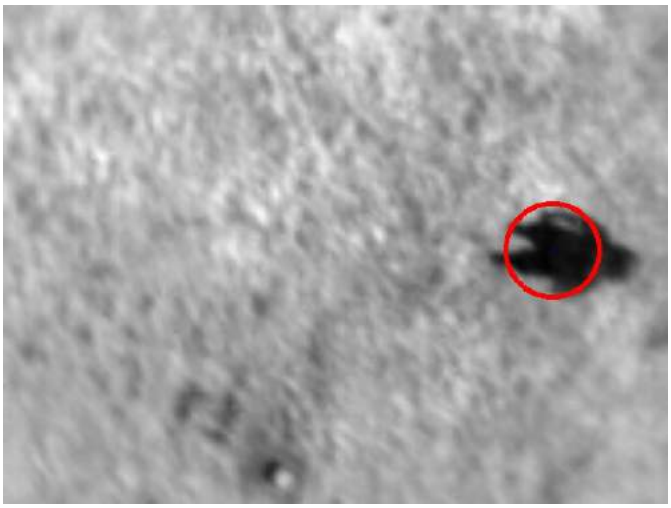

Figura 9.14: Imagen tomada en vuelo durante el día utilizando el algoritmo implementado. Se observa que el cuerpo de la persona es detectado por la cámara, y el mismo está a menor temperatura que el suelo (temperaturas mayores son colores más claros).

Conclusiones

Se confirma el funcionamiento del algoritmo en vuelo. Se observa un caso en el cual puede no funcionar correctamente (suelo de temperatura similar a la del cuerpo a identificar).

9.8. Corrección diferencial de la posición

La posición global del dron en la Tierra es calculada mediante la utilización del sistema global de navegación satelital (GNSS su sigla en inglés), en este caso, se utiliza GPS (Global Positioning System), debido a que es el más utilizado en la región, es el utilizado por ArduCopter, y también por los grupos de proyecto anteriores. Se utilizan los datos previstos por (al menos tres) satélites de órbita media para poder determinar la ubicación del receptor de GPS mediante trilateración [44]. Sin embargo, existe un cierto nivel de error en la posición obtenida, la cual puede variar por distintos factores como ser, cantidad de satélites disponibles, fenómenos ambientales, reflexiones y rebotes de las señales debido a edificios, entre otros. Existen distintas técnicas de corrección de la posición satelital, en este caso se hace uso del GPS diferencial (DGPS), la cual hace uso de una posición fija conocida para corregir en tiempo real la información del GPS. Para más información respecto a DGPS dirigirse a [10] y [43].

El grupo Termodron II implementó DGPS utilizando el servidor, o cáster, ubicado en la fortaleza del Cerro, cuyos mensajes de corrección son luego redirigidos hacia el controlador de vuelo del dron. Se decide mantener el algoritmo implementado por Termodron II, pero con la diferencia de que se realiza una prueba más exhaustiva considerando no sólo dos

puntos de medida, sino que diez [5]. Estos puntos de medida son conocidos y se encuentran en el estacionamiento del Instituto de Ingeniería Eléctrica de la Facultad de Ingeniería de la Universidad de la República, y sus coordenadas fueron obtenidas por el Instituto de Agrimensura de la Facultad, con una desviación de 12 cm en latitud y longitud. Sus coordenadas se pueden observar mapeadas en la figura 9.15.

Figura 9.15: Ubicación de los puntos relevados por el instituto de Agrimensura de la Facultad de Ingeniería, con una desviación de 12 cm en latitud y longitud.

Objetivos

Verificar el funcionamiento de la corrección diferencial de GPS en el nuevo hardware.

Descripción del experimento

Se utiliza el controlador de vuelo (Pixhawk 2.1), junto con el módulo GPS (Here2), conectados a una laptop mediante cable USB, y por radio, en este caso, utilizando una antena de 915 Mhz conectada directamente al puerto de telemetría del controlador de vuelo. Se requieren dos conexiones pues se requiere una vía de comunicación separada para enviar las correcciones diferenciales, de forma de asegurar una tasa de envío alta. Se utilizan los códigos implementados por Termodron II, con leves cambios para poder correrlos en el software diseñado para este proyecto [10].

Se coloca el conjunto controlador-GPS-Radio sobre cada uno de los puntos medidos por agrimensura, (los cuales consisten en bulones clavados en el suelo y con marcas indicando su posición) y se procede a correr los códigos utilizados por Termodron II.

[5]No se midieron los puntos 1 y 6 pues no se encontró el bulón enterrado.

Una vez relevados todos los puntos, se procede a aplicar el código de procesamiento (se creó un código nuevo en Python, ya que el utilizado por Termodron II estaba realizado en Matlab), para verificar el error en las medidas. Este código utiliza la biblioteca de Python, GeoPy, la cual contiene herramientas para el procesamiento de datos de coordenadas globales. En particular, se utiliza la función *distance* en su modalidad *great_circle*, la cual obtiene la distancia ortodrómica, o la distancia de gran círculo, es decir, la distancia más corta medida sobre la superficie de la esfera, entre dos puntos pertenecientes a la superficie de dicha esfera [45] [46].

A partir de los resultados de distancia obtenidos para cada medida se calculan los promedios y desviación estándar y se despliegan en la consola, y se generan gráficas del error en cada muestra.

Componentes utilizados

- Computadora con Dronekit-Python, MAVlink, Python
- Controlador de vuelo Pixhawk 2.1
- Módulo GNSS Here2
- Módulo transmisor de radio 3DR 915 MHz
- Cable micro USB

Resultados

En la tabla 9.14 se pueden observar las coordenadas de los puntos relevados por el Instituto de Agrimensura, junto con la diferencia promedio entre las medidas obtenidas y cada punto, tanto sin la corrección diferencial como con la misma.

Punto	Coordenadas (°,°)	Error (s/corr.)(m)	Error (c/corr.)(m)
0	(-34.91926168, -56.16618294)	8.18	6.29
2	(-34.91948173, -56.16656459)	2.15	1.03
3	(-34.91931211, -56.16735431)	2.13	1.85
4	(-34.91920019, -56.16744623)	2.30	2.35
5	(-34.91918987, -56.16728285)	2.23	1.71
7	(-34.91928451, -56.16669787)	2.25	2.02
8	(-34.91926692, -56.16629787)	2.24	1.07

Cuadro 9.13: Resultados de las pruebas de GPS diferencial. Se observan las coordenadas (medidas por el Instituto de Agrimensura con 12 cm de dispersión) de cada punto, y el error obtenido de la posición del dron tanto con corrección diferencial como sin la misma.

Algunas observaciones que se deben realizar:

- el punto 0 es el que tiene mayor error dado que el mismo está ubicado en una esquina del estacionamiento, es decir, rodeado en dos dimensiones por paredes, lo cual hace perder señal de satélites, y también ocasiona mayores rebotes, por lo que la poca precisión en este punto era de esperarse.
- los puntos que faltan en las pruebas es debido a que no se encontraron los bulones físicamente.
- la precisión de las medidas obtenidas depende de la cantidad de satélites disponibles al momento de realizar la prueba, y esta cantidad a su vez depende de la ubicación geográfica, hora del día, etc. Durante las pruebas siempre se tuvo un mínimo de 14 satélites[6] y un máximo de 18 satélites (dependiendo del punto).
- Se observa que para el punto 4 no se logra una mejora de posición mediante la corrección diferencial, en la figura 9.16 se observa la evolución del error en la medida en función de la cantidad de muestras tomadas. Se puede observar que el error aumenta al aumentar las medidas, por lo que se puede atribuir este aumento a algún factor externo (interferencia de alguna fuente, cambio en número de satélites, etc.).

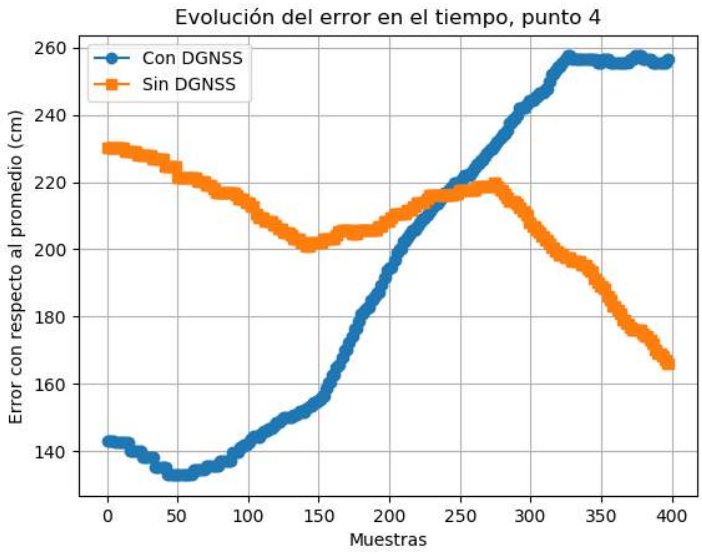

Figura 9.16: Evolución del error (como la distancia entre la posición reportada del dron y el punto medido de forma precisa por Agrimensura) respecto a la cantidad de muestras tomadas, para el punto 4. Se observa un aumento del error en el tiempo, posiblemente debido a un factor externo.

Dado el nuevo hardware, y las nuevas actualizaciones de firmware implementadas por los desarrolladores de ArduCopter, se tiene una mejor precisión sin corrección que la reportada por Termodron II en sus pruebas de GPS (2 metros y 6 metros, respectivamente).

[6]Con menos de 12 satélites se reciben advertencias del controlador de vuelo.

Conclusiones

Los resultados obtenidos son (a menos de un caso patológico conocido y un outlier que debería haberse medido de nuevo) satisfactorios, lográndose verificar el correcto funcionamiento del sistema de corrección diferencial de GPS con el nuevo hardware. Se logra una mejor precisión que la obtenida por Termodron II, gracias al nuevo hardware implementado. Esta corrección será de gran utilidad al terminar misiones, ya que al enviar al dron a las coordenadas de la base (la cual debería ser conocida con gran precisión), se sabe que el mismo no estará a más de un metro de distancia de la misma, lo cual es muy útil para el algoritmo de aterrizaje preciso, el cual se basa en visión de la base desde el dron.

9.9. Validación del algoritmo de detección de obstáculos

Objetivos

Validar el algoritmo de detección de objetos implementado.

Descripción del experimento

Para poder realizar la correcta evasión de obstáculos se implementó un algoritmo de detección de obstáculos. Con el fin de validar el algoritmo implementado se realizaron una serie de pruebas al mismo.

El experimento realizado consiste en la ejecución del algoritmo en una laptop conectada a la cámara estereoscópica, mediante el cual se realizan mediciones apuntando la cámara hacia distintos objetos y comparando contra una cinta métrica, con el fin de validar las distancias obtenidas por el algoritmo.
Se utilizan distintos objetivos para ver el comportamiento del algoritmo, como son paredes, árboles (distintos tipos de árboles, con distintos follajes) y personas.

Para poder realizar el experimento se contó con:

1. Laptop con Python 3.
2. Cámara estereoscópica Intel RealSense D415
3. Cinta métrica genérica de 10 metros.

Resultados

En la siguiente tabla se presentan las distintas medidas obtenidas para los distintos objetos, tanto con el algoritmo de detección de obstáculos como con la cinta métrica.

Objeto	Medida del algoritmo (m)	Medida de la cinta métrica (m)
Árbol de poco follaje	2.29	2.5
Árbol de mucho follaje	3.98	3.9
Persona	2.55	2.75
Pared	2.82	3.1

Cuadro 9.14: Resultados de las pruebas de detección de obstáculos.

En la figura 9.17 se presentan las imágenes de los distintos árboles utilizados para la detección de obstáculos

Figura 9.17: Imágenes a procesar con el algoritmo de detección de obstáculos

En las figuras 9.18 y 9.19 se presentan las imágenes obtenidas del algoritmo de detección de obstáculos para los cuatros objetos medidos.

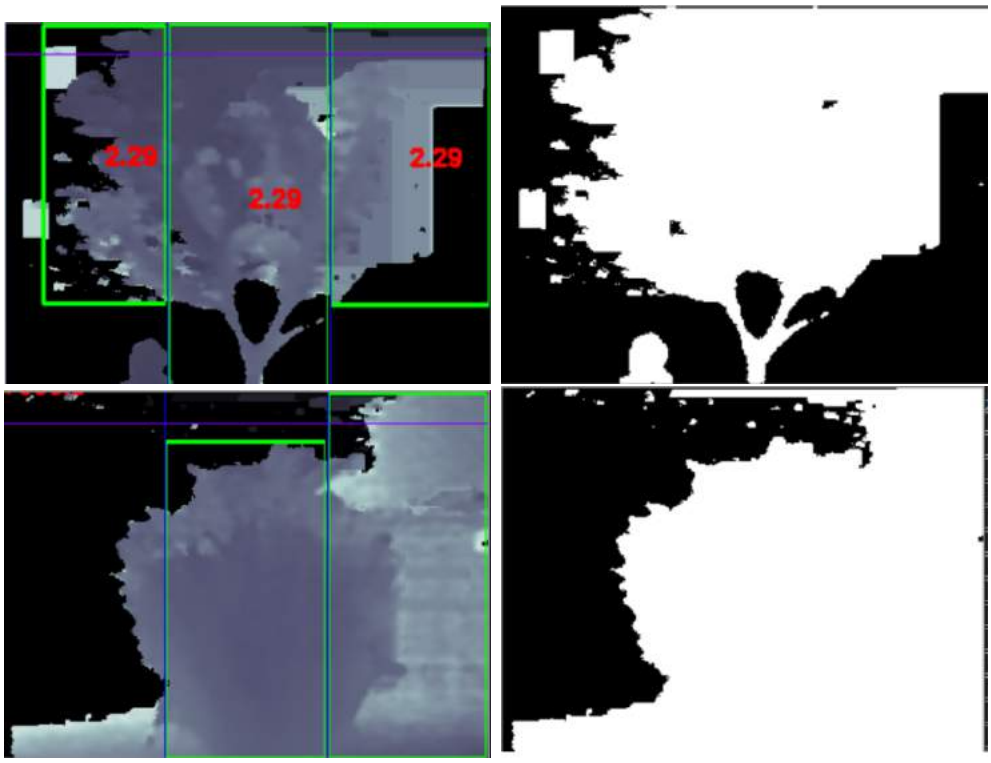

Figura 9.18: Imágenes y mascaras de profundidad del algoritmo de detección de obstáculos

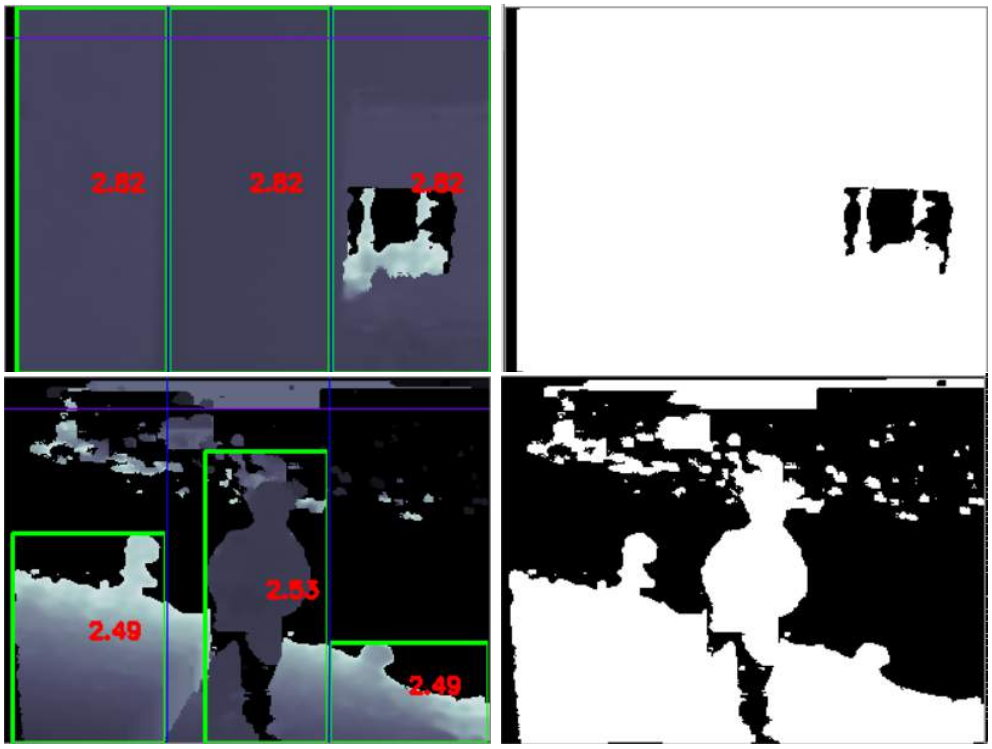

Figura 9.19: Imágenes y mascaras de profundidad del algoritmo de detección de obstáculos

Conclusiones

Se valida el algoritmo de detección de objetos, logrando detectar distintos tipos de árboles, personas y paredes.

Capitulo 10

Conclusiones

Se logra implementar un sistema que cumple con los objetivos planteados, mejorando el sistema previo desarrollado por Termodron I y II, y finalizando con un prototipo funcional.

En cuanto al Dron, se mejoró el control del mismo mediante el uso de una nueva versión del controlador de vuelo, nuevos motores, los cuales mejoraron la relación de empuje-peso, y un nuevo módulo de navegación satelital. Se mejora la autonomía de vuelo del dron por medio del uso de una batería con el doble de capacidad.
Se logró mejorar la calidad de las imágenes térmicas obtenidas por el dron gracias a un nuevo modelo de cámara térmica con resolución cuatro veces más grande, y con radiometría. A su vez, se mejora la detección de objetos durante el vuelo mediante el uso de una cámara estereoscópica, la cual posee un rango de detección más de dos veces mayor que la de los sensores de ultrasonido previamente usados, y además es capaz de detectar una mayor variedad de objetos.
Se mejora la capacidad de procesamiento de datos a bordo del dron gracias a la incorporación de una nueva computadora a bordo, reemplazando el anterior microcontrolador. Esto permite procesar la información de ambas cámaras, comunicarse con el controlador de vuelo, enviar los datos a la base y al usuario por intermedio de la comunicación inalámbrica, todo esto sin depender de enviar datos hacia la base, como sucedía previamente.

En cuanto al software del Dron, el mismo fue escrito desde cero (exceptuando el software para el uso de GPS diferencial), llevando los mismos a lenguaje de programación Python, con una arquitectura de software basada en hilos y procesamiento en paralelo.
Se diseñó e implementó un algoritmo de aterrizaje de precisión mediante el uso de reconocimiento de patrones, utilizando marcadores ArUco y el módulo RGB de la cámara estereoscópica. El mismo logra aterrizajes con errores menores a 30 centímetros y tiempos de medio minuto en condiciones climáticas ideales, y posee técnicas de corrección de posición en caso de pérdida de visión del marcador.
Se diseñó un nuevo algoritmo de evasión de obstáculos, haciendo uso del nuevo hardware,

el cual permite al dron esquivar moviéndose hacia distintas zonas según la ubicación de los obstáculos.
Se mejora el algoritmo de detección térmica, permitiendo detectar objetos según su temperatura, señalarlos en la imagen, y guardar las mismas para su posterior envío al usuario.
Por otro lado, se reincorporó el barrido de área como opción de misión, manteniendo la idea implementada por Termodron I, pero usando un algoritmo en Python.

En cuanto a la comunicación con el usuario, se implementó una interfaz gráfica, con la cual se logra una comunicación simple y robusta entre el usuario, base y dron, así como una visualización previa del recorrido del dron y de los reportes en tiempo real del mismo.

Se implementa un diseño de la base que ayuda a paliar los posibles errores en el aterrizaje de precisión, asistiendo así al acople de las bobinas del sistema de recarga inalámbrica. Se implementó todo el software de la misma, exceptuando los algoritmos de control del hardware correspondiente, debido que no se construyó la misma.

Se destaca el gran alcance abarcado por este proyecto, incorporando módulos de áreas como ser: algoritmos de control, robótica, potencia, desarrollo de software, comunicación inalámbrica, elección e integración de distintos componentes de hardware, tratamiento de imágenes, manejo de distintos tipos de sensores, entre otros. Esto permitió adquirir una gran experiencia de ingeniería en general, no solo de ingeniería eléctrica, y también permitió obtener experiencia más profunda sobre el trabajo en equipo, manejo de proyectos, presentaciones de progreso, trabajo bajo presión y resolución de problemas durante pruebas en el campo, lo cual será de gran utilidad para el futuro laboral.

Finalmente, se quiere mencionar que se logra implementar no solo un proyecto de ingeniería con un alcance amplio, sino también una plataforma de desarrollo para múltiples proyectos a futuro, ya sea el desarrollo de nuevos drones autónomos, como el uso del controlador Pixhawk para el manejo de otro tipo de robot móvil, el uso de cámaras estereoscópicas para detección o modelado de objetos, así como también la detección térmica, ya habiéndose desarrollado un proyecto de detección de temperatura corporal utilizando el modelo de cámara térmica de este proyecto, por este mismo equipo de estudiantes y tutor.

Capitulo A

Apéndice

A.1. Anexo I: Glosario

GNSS: Sistema de navegación satelital global.

GPS: Sistema de posicionamiento global.

DGPS: sistema de posicionamiento global con corrección diferencial.

AWS: Amazon Web Services: conjunto de servicios en la nube, provistos por Amazon.

thing: entidad creada por la consola de AWS-IoT, la cual posee certificados y claves que verifican su identidad en el servidor de AWS.

IoT: Internet de las cosas.

API: Interfaz de aplicaciones.

SDK Kit de desarrollo de software.

TCP Protocolo de control de transmisión.

IMAP Protocolo de acceso a mensajes de internet.

SMTP protocolo para transferencia simple de correo.

SSL capa de puertos seguros.

PWM: Modulación por ancho de pulsos.

ESC: Controlador electrónico de velocidad.

LiPo: Tipo de batería, de litio-polímero.

BEC: Circuito eliminador de batería.

SBC: Computadora de una placa, término utilizado para nombrar una línea de computadoras construidas en una sola placa de circuito, teniendo un factor de forma pequeño.

IMU: Unidad de medición inercial, comprende un acelerómetro, giroscopio, y brújula.

ADC: Convertidor Analógico-Digital, permite convertir señales de voltaje analógico a señales digitales.

Scrollbar: Barra de desplazamiento de la interfaz gráfica.

Pipeline: "tubería", estructura de datos similar a una fila o un stack, donde se pueden ingresar o remover datos de la misma, pero sólo por los dos extremos de la misma.

Queue: "cola": estructura de datos similar a un *pipeline* pero se permite cualquier cantidad de ingresos y/o retiros de datos simultáneos.

API: por su sigla en inglés Application Programming Interface, es un conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

A.2. Anexo II: Funcionamiento de la cámara térmica

La termografía infrarroja es la ciencia dedicada a la adquisición y procesamiento de información térmica de un objeto a partir de la radiación infrarroja emitida por el mismo mediante dispositivos de medida sin contacto [29].

A continuación se presentan algunas consideraciones teóricas previas sobre termografía infrarroja y luego se presentan las principales leyes que rigen el comportamiento de la termografía infrarroja [29].

Se considera que:

1. Cualquier objeto con temperatura superior a los 0 K emite radiación infrarroja .
2. El ojo humano no puede ver este tipo de radiaciones, en la figura A.2 se observa el espectro electromagnético.
3. La termografía infrarroja es una tecnología no invasiva, por lo cual no afecta al objeto a medir.
4. Esta tecnología presenta una gran ventaja en distintas áreas como la seguridad, la veterinaria, la medicina y la agricultura entre otras.

Hay tres formas por la cual se puede disipar la energía radiante; absorción (α_λ), transmisión (ρ_λ) y reflexión (τ_λ), las cuales cumplen:

$$\alpha_\lambda + \rho_\lambda + \tau_\lambda = 1 \tag{A.1}$$

En caso de ser un material opaco, la ecuación se representa de la siguiente forma:

$$\alpha_\lambda = 1 - \rho_\lambda \tag{A.2}$$

En caso de ser un cuerpo negro, la ecuación se representa de la siguiente forma:

$$\alpha_\lambda = 1 \tag{A.3}$$

La ley de Kirchhoff (propuesta por Gustav Kirchhoff en 1859) de la radiación térmica, es un teorema el cual establece que si un objeto se encuentra en equilibrio térmico, su emisividad es igual a su absortividad.

En 1800, el astrónomo Sir Frederick William Herschel descubrió la radiación infrarroja, esto lo realizó mediante el pasaje de luz solar por un prisma y observando que la temperatura aumentaba al pasar de una zona de color violeta a una zona de color roja. Herschel, además observo que la temperatura aumentaba más allá de la zona roja, con lo cual se demostró empíricamente que existen otras formas de luz invisibles al ojo humano [30].

En 1879, Josef Stefan dedujo la ley de Stefan-Boltzmann, la cual establece que la intensidad irradiada W (energía por unidad de superficie y tiempo) de salida de un cuerpo negro es proporcional a la cuarta potencia de la temperatura [30].

$$W = \epsilon \sigma T_e^4 \tag{A.4}$$

donde T_e es la temperatura absoluta de la superficie, σ, de valor $5{,}67x10^{-8}$ $\frac{W}{m^2K^4}$ es la constante de Boltzmann y ϵ es la emisividad; al tratarse de un cuerpo negro la emisividad tiene valor 1. Si se considera un cuerpo gris (cuerpo real), la emisividad es menor a la del cuerpo negro y se encuentra entre 0 y 1 [30].

En 1909, Max Planck propuso la ley de Planck [29], en donde la intensidad de la radiación emitida por un cuerpo negro con una temperatura T y frecuencia v, en equilibrio térmico, se describe por la siguiente ecuación:

$$I(v,T) = \frac{2hv^3}{c^2} \frac{1}{e^{\frac{hv}{KT}-1}} \tag{A.5}$$

donde:

1. h es la constante de Planck, de valor $6{,}62606896x10^{-34}$.
2. σ es la constante de Boltzmann, de valor $5{,}67x10^{-8}$ $\frac{W}{m^2K^4}$.
3. T es la temperatura en Kelvin.
4. c es la velocidad de la luz en m/s.
5. v la frecuencia en Hz.

De la ley de Planck se deduce la ley de Wien: William Wien enunció que la longitud de onda a la que se irradia la máxima energía, multiplicada por la temperatura absoluta T, tiene un valor constante $\lambda_{max}T = 2{,}898X10^{-3}$mK, por lo tanto la longitud de onda a la que se irradia la máxima energía se expresa como:

$$\lambda_{max} = 2898\frac{\mu mK}{T} \tag{A.6}$$

En la figura A.1 se presenta la ley de Wien donde se tienen distintas curvas de intensidad en función de la longitud de onda, para distintas temperaturas [30].

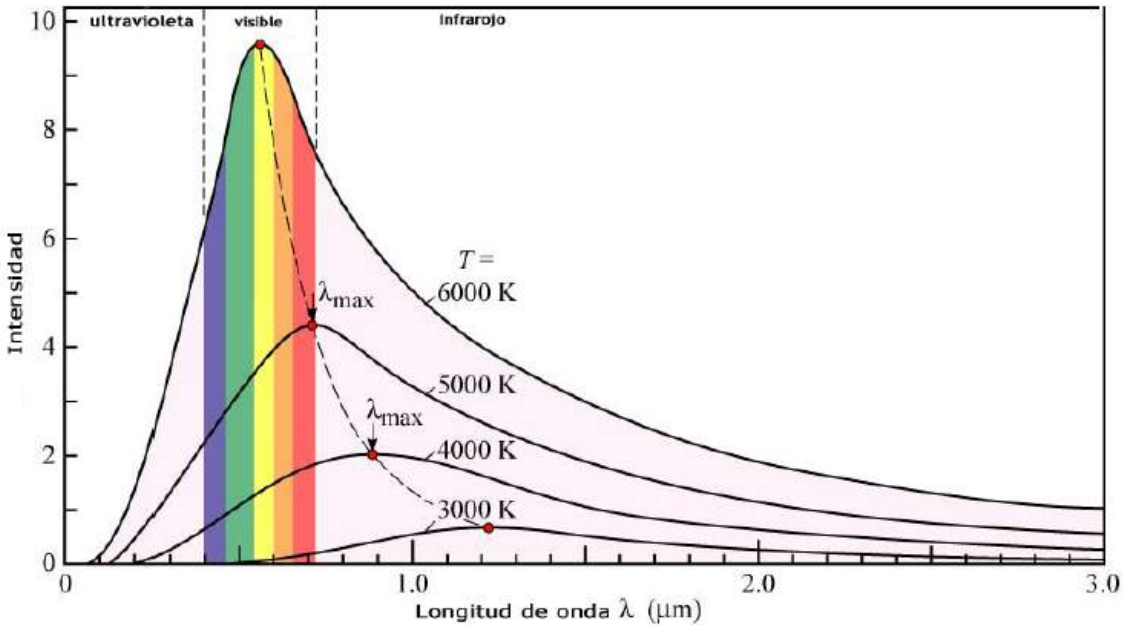

Figura A.1: Ley de Wien.

En la figura A.2 se presenta el espectro electromagnético. Este se compone de varias zonas las cuales comparten las mismas leyes pero se diferencian por su longitud de onda; dentro de estas zonas se encuentra la luz visible, la cual tienen longitudes de onda entre los 400 nm y 700 nm, la infrarroja, la ultravioleta, entre otras.

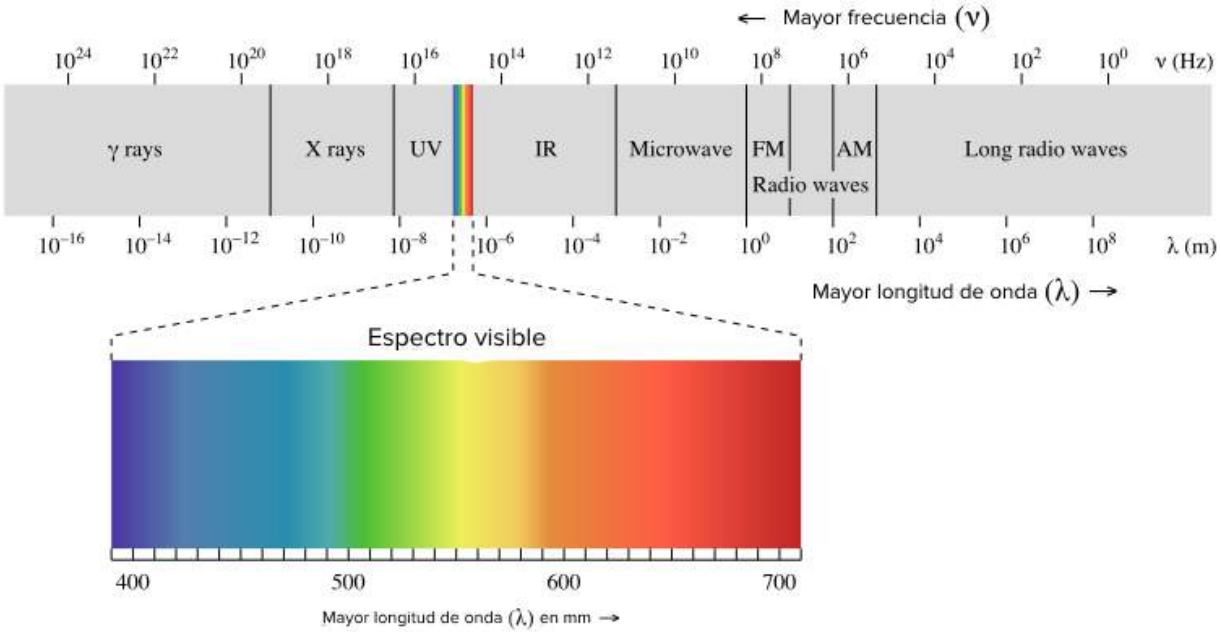

Figura A.2: Espectro electromagnético. Imagen obtenida de **Khan Academy**.

Los ojos humanos son capaces de captar únicamente las longitudes de onda entre los 400 nm y los 700nm, mas allá de esto el ser humano posee receptores que perciben la sensación térmica. Cuanto mayor sea la temperatura de un objeto, mayor sea la radiación infrarroja que emita [30].

El espectro infrarrojo se puede dividir en varias bandas o porciones, estas separadas por la longitud de la onda, las mismas se definen a continuación:

- Infrarrojo cercano: desde 0,8μm a 1.7μm
- Infrarrojo de longitud de onda cercano (SWIR): desde 1μm a 2.5μm.
- Infrarrojo de longitud de onda media (MWIR): desde 2μm a 5μm.
- Infrarrojo de longitud de onda largo (*LWIR*): desde 8μm a 14μm.

No todo el rango infrarrojo es utilizable para las cámaras térmicas debido a que mucho de este espectro es filtrado por la atmósfera. En el espectro visible, de 0,4 μm a 0,7 μm, solo se transmite el 60 % de la radiación emitida. Las regiones comúnmente utilizadas son MWIR y *LWIR*, conocidas como las "bandas de la radiación termica", MWIR se utiliza para medidas de alta temperatura, mientras que *LWIR* para medidas de temperatura ambiente [30]. En el caso de la FLIR Lepton 3.5, esta cámara consta de un sensor *LWIR* para longitudes de onda entre 8μm y 14μm [28].

Basado en estas leyes, se utiliza un sensor llamado microbolómetro para el funcionamiento de una cámara térmica. Este sensor recibe radiación en el rango de onda infrarrojo, calentando su superficie y alterando su resistencia eléctrica. Se obtiene entonces una relación entre la resistencia del microbolómetro y la radiación incidente al mismo, permitiendo entonces, mediante las distintas leyes anteriormente mencionadas, poder relacionar dicha radiación con la temperatura del cuerpo que la emite [30]. La cámara térmica FLIR Lepton 3.5 posee un microbolómetro sin refrigeración VOx (óxido de vanadio) [28].

Las cámaras térmicas también se pueden diferenciar según la arquitectura de la cámara, teniendo dos posibilidades, la primera mediante un sensor único y la segunda mediante el uso de una matriz de plano focal (*FPA*, por su sigla en ingles). Las cámaras de sensor único constan de un "espejo giratorio", donde el sensor mide la energía irradiada por la superficie de un objeto reflejado en el espejo, el cual está girando. El giro del espejo permite lograr un barrido horizontal de las radiaciones.

Un *FPA*, tal cual se usa en la FLIR Lepton 3.5 [28], consta de una matriz de múltiples detectores, donde cada detector provee información de la radiación de cada punto en el plano. Entonces, el funcionamiento de las cámaras térmicas con *FPA* puede describirse como una función $f(x, y)$ donde la amplitud del valor de la función corresponde a la cantidad de energía radiada por la región en el plano caracterizada por x e y.

Debe mencionarse que el objeto presente en el plano no solo presenta la radiación emitida por el mismo, sino que también absorbe y refleja la radiación proveniente de otros cuerpos a su alrededor. La radiación total se puede escribir como:

$$W_{tot} = E_{obj} + E_{refl} + E_{atm} \tag{A.7}$$

donde:

- W_{tot}: Radiación total recibida.
- E_{obj}: Radiación del objeto.
- E_{ref}: Radiación reflejada por los alrededores.
- E_{atm}: Radiación emitida por la atmósfera.

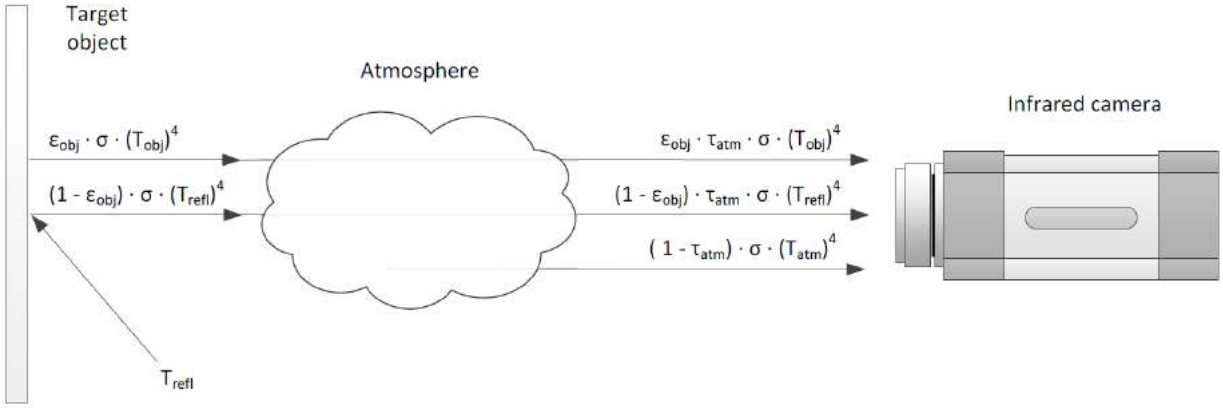

Figura A.3: Modelo de la cámara térmica. Imagen obtenida de [30].

Considerando las leyes ya mencionadas anteriormente y que no toda la radiación emitida por el objeto es recibida por la cámara (es una función que depende de la transmitancia de la atmósfera, τ_{atm}), se tiene que:

$$E_{obj} = \epsilon \tau_{atm} \sigma T^4 \tag{A.8}$$

Los cuerpos grises tienen reflectividad mayor a cero, por ende ellos reflejan la radiación emitida por los alrededores. Se tiene entonces:

$$E_{ref} = \rho_{obj} \tau_{atm} \sigma T_{ref}^4 = (1 - \epsilon_{obj}) \tau_{atm} \sigma T_{ref}^4 \tag{A.9}$$

La radiación emitida por la atmósfera se puede escribir como:

$$E_{atm} = \epsilon_{atm} \sigma T_{atm}^4 = (1 - \tau_{atm}) \sigma T_{atm}^4 \tag{A.10}$$

Sustituyendo las ecuaciones A.8, A.9 y A.10 en la ecuación A.7, se tiene que:

$$W_{tot} = \epsilon_{obj} \tau_{atm} \sigma T_{obj}^4 + (1 - \epsilon_{obj}) \tau_{atm} \sigma T_{ref}^4 + (1 - \tau_{atm}) \sigma T_{atm}^4 \tag{A.11}$$

Despejando la ecuación A.11, se tiene:

$$T_{obj} = \sqrt[4]{\frac{W_{tot} - (1 - \epsilon_{obj}) \tau_{atm} \sigma T_{ref}^4 - (1 - \tau_{atm}) \sigma T_{atm}^4}{\epsilon_{obj} \tau_{atm} \sigma}} \tag{A.12}$$

Como se puede observar en la ecuación A.12, la temperatura del objeto a medir depende de parámetros no conocidos, los cuales son la temperatura de la atmósfera, la emisividad del cuerpo objeto, la temperatura de los cuerpos de los alrededores y la transmitancia de la atmósfera.

La temperatura del ambiente es obtenida mediante el uso de un termómetro, en el caso de la cámara térmica FLIR Lepton 3.5, la misma consta con un termistor del cual no se tiene información. La termistancia de la atmósfera se estima generalmente utilizando

la distancia entre el objeto y la cámara, y la humedad relativa; este valor generalmente es cercano a uno. La emisividad del objeto y la temperatura de los cuerpos del alrededor tienen una gran influencia en la medida a tomar, estos valores se deben calcular de forma precisa, y se pueden encontrar más detalles al respecto en [28].

A.3. Anexo III: Descripción del software implementado

En la presente sección se detallan las principales funciones del software implementado junto a sus variables más importantes, se divide en tres secciones correspondientes al dron, base e interfaz.

En las tres secciones se utilizó el lenguaje de programación Python debido a que es un lenguaje dinámico, fácilmente legible, multiplataforma y con una gran cantidad de librerías de uso libre, las cuales simplifican y optimizan el código.

A.3.1. Software del dron

En la presente sección se detalla el software implementado para el dron, se comenzará detallando la librería DroneKit-Python, la cual es fundamental para el control del dron en vuelo.

A.3.1.1. librería DroneKit-Python

DroneKit-Python es una librería de Python de uso libre y proporciona funciones para facilitar la comunicación con el controlador de vuelo en un vehículo aéreo no tripulado ya sea desde una computadora a bordo mediante un enlace de baja latencia o mediante una estación de control terrestre mediante un enlace de alta latencia por radiofrecuencia. Esta librería es compatible con todas las computadoras a bordo que se comuniquen mediante el protocolo MAVlink y se puede utilizar tanto en Windows, Linux o Mac OS X [47].

La *API* de DroneKit-Python permite las siguientes funciones:

- Conectarse a uno o múltiples vehículos mediante una rutina de Python.
- Obtener información de los vehículos (estado de batería, ubicación, modo de vuelo, etc.) mediante el uso de comunicación serial a través de distintos medios como ser radiofrecuencia o cable USB.
- Control del vehículo mediante comandos MAVlink.

Dentro de las funciones de la librería, se destacan los siguientes comandos o funciones:

- ***connect(connection_string, baudrate)***: función para conectarse al vehículo, donde *connection_string* especifica el puerto serial por el cual se conectará al controlador de vuelo, y *baudrate* indica la tasa de baudios de la comunicación. Esta función devuelve un objeto de la clase *vehicle*, el cual incluye distintos *methods* o funciones propias de la clase.

- ***VehicleMode(flightmode)***: función para asignar el modo de vuelo del vehículo, el cual es especificado por el parámetro *flightmode*. Se debe aplicar esta función al método *mode* del objeto *vehicle* creado, por ejemplo:

```
vehicle.mode = VehicleMode(LAND)
```

- ***armed***: atributo (variable propia del objeto) del objeto *vehicle* con el cual se arma el vehículo al poner su valor en *True*, o se desarma si se le pone como valor *False*, por ejemplo:

```
vehicle.armed = True
```

- ***simple_takeoff(height)***: método para despegar el vehículo (si este ya está armado) hasta cierta altura, la cual se pasa en el parámetro *height*.
- ***groundspeed***: atributo que permite configurar la velocidad de vuelo en m/s para el modo GUIDED.
- ***set_velocity_body(vehicle,vx,vy,vz)***: atributo que permite asignar velocidades en cada uno de los tres ejes del vehículo *vehicle*. Se asigna la velocidad de cada eje según la variable *vx*, *vy*, *vz*, respectivamente, en m/s. Este comando mantiene dichas velocidades por hasta un segundo, si se desea mantener por más tiempo, se debe reenviar constantemente el mismo.
- ***condition_yaw(heading,relative)*** : atributo que permite girar el ángulo de yaw del dron una cantidad de grados igual al valor indicado en el parámetro *heading*. Estos ángulos son globales a menos que se asigne el parámetro *relative* como *True*, en ese caso, el ángulo será relativo al ángulo actual.
- ***rotate(pitch, roll, yaw)***: función dentro del objeto *gimbal* dentro del objeto *vehicle* para rotar el gimbal hacia la posición que se desee, se debe pasar como parámetro los ángulos correspondientes de pitch, yaw, y roll.
- ***add_attribute_listener***: método que permite agregar un *listener* o función escucha a cualquier parámetro que pueda devolver el objeto *vehicle*. La función escucha monitorea los cambios en dicho parámetro, y puede ser utilizada para generar una función *callback* para reportar los cambios en los parámetros monitoreados.
- ***simple_goto(loc)***: método que permite comandar al dron hacia una posición geográfica dada por el objeto *loc*, el cual contiene latitud, longitud y altura. Este método solo puede ser utilizado si el vehículo está en modo GUIDED.

De los códigos ejemplos de DroneKit-Python se utilizan algunas funciones que son listadas a continuación:

- ***get_distance_metres(aLocation1, aLocation2)***: función que devuelve la distancia en metros entre dos ubicaciones especificadas como objetos del tipo *location*.
- ***distance_to_current_waypoint***: función que devuelve la distancia entre la posición actual del vehículo y la ubicación del siguiente punto geográfico en el recorrido (si se utiliza el modo AUTO y los puntos geográficos son cargados al vehículo previamente).
- ***arm_and_takeoff(tgtheight)***: función que permite realizar un despegue de forma segura. Se entiende por esto que se realizan los pasos necesarios para armar y despegar el dron con los chequeos de seguridad pertinentes entre medio. Estos pasos son: chequeo de estado, chequeo si es posible armar, armado, cambio de modo, instrucción de despegue, polling de altura actual hasta llegar a la altura objetivo pasada en el parámetro *tgtheight*.

A continuación se detallan las librerías implementadas para el dron; estas son las librerías que utiliza la computadora a bordo con el fin de llevar a cabo una misión.
Las mismas siguen el orden jerárquico de la figura A.4.

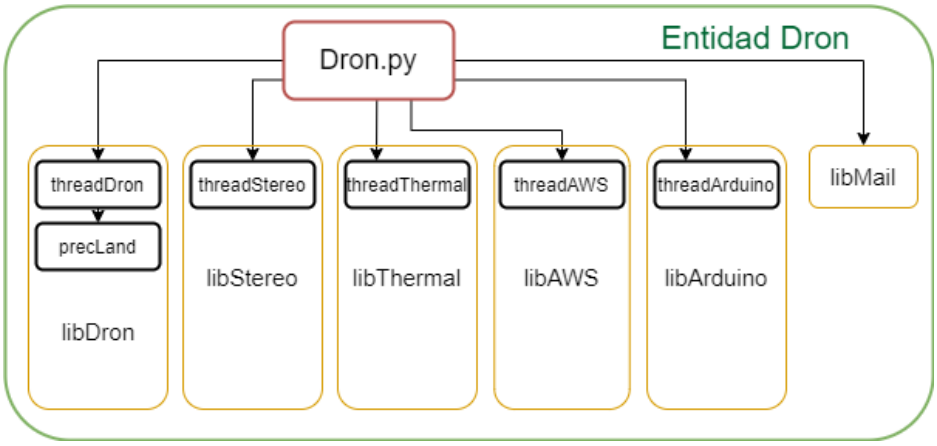

Figura A.4: Jerarquía del software para la entidad Dron. En rojo se encuentra el código principal de la entidad; en amarillo las librerías utilizadas por éste, y en negro los *threads* implementados en dichas librerías. Las flechas apuntan desde el objeto de mayor jerarquía al menor, es decir, los objetos en las puntas de las flechas son incluidos o están dentro de los objetos en el otro extremo de la flecha.

A.3.1.2. libDron

Esta es la librería principal del dron, es la encargada de conectarse con el mismo y realizar el control del dron durante la misión, utiliza los comandos de la librería DroneKit-Python para comandar al controlador de vuelo durante la misma.
La misma se divide en dos scripts, uno de nombre threadDron, el cual realiza el control del dron y la misión; y el otro, de nombre precisionLanding, el cual es el encargado de realizar el aterrizaje de precisión.

A continuación se describen las rutinas implementadas para usar dentro del *thread* correspondiente al vuelo del dron (no debe confundirse este *thread* con el código principal del dron, el cual contiene la máquina de estados para la comunicación con la base y el usuario, y es el que invoca este *thread* una vez que está listo para comenzar la misión).

- ***set_mission_parameters()***: esta función obtiene los parámetros de misión del documento JSON recibido por el dron. En particular, recibe los parámetros de los puntos geográficos a recorrer (latitud, longitud, altura), las coordenadas de la base (latitud y longitud) y el modo de funcionamiento de la cámara térmica.

- ***set_mission_waypoints(lat, lon, alt)***: esta función se encarga de cargar en el controlador de vuelo los puntos geográficos a recorrer en la misión. El último punto siempre corresponde a la ubicación de la base (a altura de vuelo), de forma de retornar a la misma al terminar el recorrido, para luego comenzar el aterrizaje de precisión.

- ***set_callback()***: esta función genera los escuchas de los parámetros a reportar respecto al estado del dron, y luego genera una función *callback* la cual es llamada cada vez que uno de estos escuchas actualiza su valor (porque el parámetro que monitorea cambió su valor). Se utiliza para actualizar un documento JSON con, entre otras cosas, los parámetros correspondientes al estado del dron. Este JSON es luego enviado por AWS para el reporte en tiempo real.

- ***evade_loop()***: esta función engloba todo el algoritmo de evasión de obstáculos descrito en la sección **??**. Es activado cuando se levanta la bandera de obstáculo detectado por el *thread* de la cámara estéreo, y según el valor de distancia reportado por ésta, es la acción a tomar.

- ***startPrecLanding()***: esta función engloba todo el algoritmo de aterrizaje de precisión, el cual es descrito en profundidad en la sección 6.

Finalmente, se describen las funciones utilizadas en el código principal del dron (referirse al diagrama A.4 para una vista más clara de las distintas jerarquías de los *scripts* y librerías implementados). En particular, se describen las funciones de comunicación por correo electrónico implementadas a partir de las funciones *recv_mail_monitor* y *send_mail_monitor* de la librería libMail, simplemente variando el contenido a enviar o el asunto del correo a recibir, de forma de mantener el código final en forma más legible.

- ***wait_for_status_request()***: función de monitoreo de la bandeja de entrada por correos de pedido de reporte de estado del dron hacia la base. En caso de no recibir el correo antes de terminar el tiempo de *timeout* devuelve error.

- ***send_dron_status(statusTag)***: función de envío del estado del dron hacia la base. El estado se envía como texto plano y se pasa a la función a través del parámetro *statusTag* el cual debe ser una variable diccionario de Python, en particular, con el

formato JSON. En caso de no poder enviar el correo antes del tiempo de *timeout* devuelve devuelve error.

- ***wait_for_mission()***: función de monitoreo de la bandeja de entrada por correos de la base con los parámetros de misión. En caso de recibir el correo, retorna en una variable diccionario en formato JSON los parámetros de misión recibidos. En caso contrario devuelve error.

- ***send_mission_begin()***: función de envío de comienzo de misión hacia la base. Envía el string de texto plano "begin". En caso de no poder enviar el correo antes del tiempo de *timeout* devuelve devuelve error.

- ***send_mission_end()***: función de envío de fin de misión hacia la base. Envía el string de texto plano "end". En caso de no poder enviar el correo antes del tiempo de *timeout* devuelve devuelve error.

- ***send_mission_logs(attachments)***: función de envío de logs de vuelo y fotos térmicas de la misión realizada, hacia la base. Se debe pasar la ruta a la carpeta donde se encuentran los archivos a enviar en el parámetro *attachments*. En caso de no poder enviar el correo antes del tiempo de *timeout* devuelve devuelve error.

Para ver más detalles de la máquina de estados del Dron, la cual se implementa en este código principal, ver 5.4.

A.3.1.3. libArduino

Esta librería se utiliza para la comunicación entre los sensores de ultrasonido y la computadora a bordo, mediante el uso de un Arduino Nano como puente entre ambos. La librería se compone de dos algoritmos, uno del lado del Arduino Nano y otro en la computadora a bordo.

Del lado del Arduino Nano, el algoritmo se encarga de tomar las medidas de los sensores de ultrasonido y enviarlas al puerto serial, donde la computadora a bordo estará leyendo los datos correspondientes. El algoritmo se compone de tres funciones, las cuales son:

- ***read_sensor()***: Función encargada de leer los sensores de ultrasonido, utiliza la función provista por Arduino *analogRead(anPin2)*, la cual se le pasa el pin asociado al sensor y devuelve un voltaje proporcional a la distancia.

- ***print_range()***: Función encarada de enviar los datos al puerto serial correspondiente, previamente acondiciona los datos en el string correspondiente para el envío. Se utiliza la función prevista por Arduino *Serial.print(dato)*, donde se le pasa un dato, el cual puede ser string, float o int y la función se encarga de enviarlo al puerto serial.

- ***setup()***: Función encargada de realizar las inicializaciones, definir variables y conectarse al puerto serial correspondiente.

El algoritmo correspondiente a la computadora a bordo fue escrito en Python y se compone tres funciones y un *thread*. Además, se utilizan las librerías propias de Python serial, time, threading y Queue.

Las funciones implementadas son las siguientes:

- ***medida_ultrasonido()***: Función que se encarga de leer los datos del puerto serial, separarlos por espacio y convertir los datos de string a float. Utiliza la función *readline* de la librería serial para leer los datos del puerto serial, la función *split*, la cual se le pasa el carácter de separación a utilizar y separa el string y la función *isdigit* a la cual se le pasan todos los string separados y en caso de ser dígitos los convierte a float.
- ***cola_datos_adelante()***: Función la cual se encarga de poner los datos obtenidos del sensor que apunta hacia adelante en la cola, con el fin de enviarlos al algoritmo de detección de obstáculos. Utiliza la función *put* de la librería Queue.
- ***cola_datos_abajo()***: Función idéntica a *cola_datos_adelante* pero con el sensor que apunta hacia abajo en el dron.
- ***threadArduino()***: *thread* que se encarga de realizar las inicializaciones del puerto, definir las colas para ambos sensores y llamar a las distintas funciones.

A.3.1.4. libThermal

Esta librería es la encargada de realizar la detección de cuerpo calientes o focos de calor. Se compone de dos funciones de la librería *purethermal1-uvc-capture*, una función implementada y un *thread*. Se utilizan destinas librerías de Python, las cuales facilitan el algoritmo, estas se listan a continuación:

- ***threading***: para la creación y uso de *threads*.
- ***datetime, time***: librerías para el manejo de timers.
- ***math***: librería para el manejo de funciones matemáticas.
- ***numpy***: librería para el manejo de arreglos y matrices.
- ***os***: librería para el manejo del sistema operativo (control de ruta de ficheros, creación de carpetas, entre otros).
- ***OpenCV***: librería de procesamiento de imágenes y *computer vision*.

Se destacan las siguientes funciones de OpenCV, las cuales se utilizaron en las distintas funciones implantadas, por mas información sobre las mismas referirse a [48].

- ***normalize(src, dst, alpha, beta, norm_type)***: realiza una normalización del arreglo de entrada *src* y retorna el arreglo normalizado entre los valores *alpha* (mínimo) y *beta* (máximo) en el arreglo *dst*. También tiene como parámetro de entrada el tipo de normalización que se quiera realizar, los cuales se explican en más detalle en la documentación correspondiente [48].

- ***cvtColor(src, dst, code)***: convierte la imagen de un color a otro, tiene como entrada la imagen en *src*, la imagen de salida en *dst* y el código de color al cual convierte, en *code*.

- ***resize(src, dsize, fx, fy, interpolation)***: hace un cambio de escala a una imagen ingresada en *src*, dado el parámetro *dsize* que indica el tamaño de imagen de salida. Los parámetros *fx, fy* son los factores de escala en x e y, respectivamente, y el parámetro *interpolation* determina el tipo de interpolación a realizar para el cambio de escala.

- ***inRange(src, lowerb, upperb)***: dado un umbral de valores compuesto por dos valores de entrada: *lowerb*, *upperb*, y una imagen de entrada (*src*), esta función realiza una segmentación binaria y devuelve una imagen donde los píxeles que estén dentro del umbral serán de color blanco, y el resto de negro.

- ***findContours(image, mode, method)***: función que se encarga de obtener los contornos de una imagen binaria. En el parámetro de entrada *src* recibe la imagen fuente, en *mode* se establece el tipo de método de organización de los contornos retornados (es posible establecer jerarquías entre los diferentes contornos), en *method* se ingresa el método de aproximación de los puntos del contorno (para ser más eficiente, no devuelve todos los puntos que describen el contorno, suprime algunos según este parámetro). Devuelve los contornos detectados como un array de coordenadas [**?**].

- ***contourArea(contour)***: función que se encarga de calcular el área de un contorno.

- ***minEnclosingCircle(area)***: función que se encarga de encontrar el circulo de menor tamaño que encierra un área.

- ***circle(img, center, radius, color)***: función a la cual se le ingresa una imagen, un centro, radio y color; y devuelve la imagen con un circulo dibujado en la posición especificada.

Se utiliza la librería *uvctypes* provista por GroupGets, para el conexionado de la cámara térmica y la obtención de los datos de la misma mediante Python. Además, se utilizan las siguientes funciones de la librería *purethermal1-uvc-capture*, también provistas por GroupGets:

- ***raw_to_8bit()***: e compone de dos funciones propias de OpenCV y una de la librería NumPy, esta función obtiene los datos obtenidos de la cámara térmica y devuelva una imagen de 8 bits. Las funciones que utiliza son las siguientes:
  - *normalize(src, dst, alpha, beta, norm_type)*
  - *cvtColor(src, dst, code)*
  - *right_shift(src,n)*: realiza el corrimiento de n bits a la izquierda, es decir, la división binaria, para cada elemento del arreglo de entrada *src*.
- ***ktoc()***: implementada para convertir la temperatura de grados Kelvin*100 a grados Celsius. Esta realiza la siguiente operación:

$$t_C = \frac{t_K - 27315}{100} \tag{A.13}$$

Las funciones implementadas son:

- ***detTherm(numImg, t_min, t_max)***. La cual recibe los parámetros: *numImg* donde se indica el número de la imagen, utilizado para inicializar el primer objeto detectado, guardando sus coordenadas para luego comparar la posición del siguiente objeto detectado, y asegurarse de no estar detectando el mismo objeto de nuevo. Los parámetros *t_min, t_max* que corresponden temperaturas máxima y mínima en ºC de los objetos a buscar dentro de las imágenes térmicas.

  Devuelve los siguientes parámetros: *detected*: parámetro booleano que indica con True si se detecto un cuerpo caliente o foco de calor, en caso contrario devuelve False, *(x_center, y_center)*: coordenadas del centro del objeto detectado, *radius*: radio del círculo mínimo que encierra al objeto detectado, *img*: imagen capturada con el círculo dibujado.

- ***threadThermal(modo, qFlagThermal, picFolder, showImg, showPrint, saveFrames)***: recibe como parámetros, el modo de funcionamiento de la cámara en *modo*, un objeto del tipo *queue* para poder detener el *thread* mediante otras rutinas externas, el nombre de la carpeta donde se guardaran las imágenes en *picFolder* y distintos parámetros de control para el despliegue de imágenes o el guardado de los cuadros de datos.

  Al comenzar el thread, se revisa el modo de funcionamiento elegido para configurar la temperatura mínima y máxima, en caso de ser cuerpos calientes, estas se configuran en 30 y 45 ºC respectivamente; en caso de ser focos de calor las mismas se configuran entre 100 y 1000 ºC. Luego de configuradas las temperaturas a utilizar en el rango, se procede a la inicialización de la cámara térmica.

  Finalmente, se entra en un loop infinito en el cual se llama a la función *detTherm*, se calcula la distancia entre el centro del objeto detectado y el centro del objeto

detectado previamente y en caso de ser mayor al radio del objeto detectado previamente se guarda la imagen.

A.3.1.5. libMail

La librería libMail se implementó con el fin de facilitar el uso de correos electrónicos entre las tres entidades del sistema (Dron, Base e Interfaz). Esta librería contiene funciones tanto para el envío como para la recepción de correos electrónicos, las cuales son llamadas con distintos parámetros de entrada, dependiendo del correo que se quiera enviar o recibir, el asunto, el texto a enviar, imágenes a adjuntar y si se quieren opciones de testeo y debuggeo en pantalla.

Para la implementación de la librería se utilizan distintas librerías propias de Python como son: smtplib (librería para el envío de e-mails), email (librería para el manejo de los mensajes de los e-mails, no esta destinada al envío de los mismos) y imaplib (para la recepción de correos electrónicos).
Además, se utilizan otras librerías como son os y sys (para el manejo del sistema, creado de carpetas, etc), json (para la creación del archivo a enviar con los parámetros del dron) y time (para el manejo de tiempos en los intentos de envío de correos electrónicos).

Previo a la implementación de las funciones se realiza la configuración de los parámetros de usuarios y contraseñas para las distintas entidades del sistema.

```
dronMail = "termodron.dron@gmail.com"
dronPass = "Termo.3.dron"

baseMail = "termodron.base@gmail.com"
basePass = "Termo3Base"

uiMail = "termodron.interfaz@gmail.com"
uiPass = "termo3.interfaz"
```

Las funciones implementadas se detallan a continuación:

- ***get_subject(mode)***: función a la cual se le pasa el modo de uso del correo y devuelve el asunto del correo. Esta función es de utilidad para el filtrado por asunto en la recepción de los correos.
  El modo del correo puede ser uno de los siguientes:
  - SET_PARAM: Para la configuración de parámetros de misión, el asunto del correo será “Termodron3: - SET MISSION”.

- RET_PARAM: Para la confirmación de los parámetros de la misión, el asunto del correo será “Termodron3: - ACK MISSION”.
- REQ_STATUS: Para solicitarle al dron el estado del mismo, el asunto del correo será “Termodron3: - REQ STATUS”.
- RET_STATUS: Para devolver el estado del dron, el asunto del correo será “Termodron3: - RET STATUS”.
- REQ_MISSION: Para solicitar el comienzo de la misión, el asunto del correo será “Termodron3: - REQ BEGIN MISSION”.
- RET_MISSION: Para indicar que se inicio la misión, el asunto del correo será “Termodron3: - RET BEGIN MISSION”.
- MISSION_OVER: Para indicar que la misión finalizo, el asunto del correo será “Termodron3: - MISSION OVER”.
- REQ_LOG: Para solicitar el envío del log de vuelo, el asunto del correo será “Termodron3: - REQ LOG”
- RET_LOG: Para la devolución del log de vuelo, el asunto del correo será “Termodron3: - RET LOG”
- REPORT: Para el envío del reporte de la misión, el asunto del correo será “Termodron3: - reporte de mision”.

- ***get_search_key(subject, senderMail)***: función a la cual se le pasa como parámetro el asunto del correo y el remitente y la misma genera un string con el formato de correo a buscar. Con esta función se busca optimizar la búsqueda de correos que no se hayan leído previamente en las distintas direcciones de correos electrónicos.

- ***_send(sender, sendPass, recv, body, html_body, subject, attachmentList, debug)***: función a la cual se le pasan los parámetros para el envío del correo electrónico y este, mediante el uso de la librería email genera el correo electrónico y lo envía usando la librería smtplib. A continuación se listan los parámetros de entrada:

  - sender: dirección de correo electrónica del remitente.
  - sendPass: contraseña del correo del remitente.
  - recv: correo electrónico del receptor.
  - body: texto plano del cuerpo del correo.
  - html_body: texto en html del correo.
  - subject: asunto del correo.
  - attachmentList: lista de archivos a adjuntar y enviar.
  - debug: bandera para debuggeo, muestra mensajes en consola.

  Devuelve error si no se logra enviar el correo.

- ***_receive(user, passwd, searchKey, debug)***: función a la cual se le pasan los parámetros para la búsqueda de correos entrantes no leídos, según asunto y cuenta que lo envió. Los parámetros de entrada de la función son: *user* la cuenta donde se reciben los correos, *passwd* la contraseña de dicha cuenta, *searchKey* asunto generado por la función *get_search_key*, *debug* bandera de debuggeo, muestra mensajes en consola. Devuelve error si no se reciben mensajes,
- ***send_mail_monitor(sender, to, plaintext, htmltext, mode, att_folder, timeout, interactive)***: función de monitor de envío de correos. Utiliza la función *_send* en un loop, reenviando el correo en caso de obtener correo. Si se pasa del tiempo indicado por el parámetro *timeout* devuelve error.
- ***recv_mail_monitor(user, mode, sender, timeout, interactive)***: función de monitor de recepción de correos. Utiliza la función *_receive* en un loop, volviendo a revisar los correos no leídos de la bandeja de entrada por el asunto deseado. Si se pasa del tiempo indicado por el parámetro *timeout* devuelve error. Si logra recibir correo, devuelve el contenido de texto plano y la ruta hacia el directorio donde se guardan los archivos adjuntos.

A.3.1.6. libStereo

Se utiliza la librería de *libRealSense* en su *wrapper* de Python: *pyrealsense2* para el manejo de la cámara estereoscópica. Utilizando las funciones de dicha librería, junto con funciones de las librerías de Python: OpenCV, NumPy, Threading y Multiprocessing se implementan las siguientes funciones:

- ***setupRS()***: esta función se encarga de las inicializaciones y configuración de la cámara estéreo. En particular se configuran la resolución a utilizar y los filtros de post-procesmiento a aplicar a los cuadros de profundidad obtenidos.
- ***threadStereo(flagObstSend,datosSend,qFlagStereo,showImg,thMin,thMax)***: este *thread* se encarga de obtener las imágenes de profundidad de la cámara estéreo y realizar la búsqueda de objetos a distancias dentro del rango de interés. La variable *flagObstSend* es del tipo *pipeline* y es utilizada para notificar a otro script si es que se detecta algún objeto; el *pipeline*, *datosSend*, funciona de igual manera, pero envía la información de distancia de los objetos detectados. La variable del tipo *queue*, *qFlagStereo*, se utiliza para comenzar o detener este *thread* por un script exterior. La variable auxiliar *showImg* permite mostrar en pantalla las imágenes de profundidad con los contornos de los objetos detectados y sus distancias. Las variables *thMin*, *thMax* son las distancias mínimas y máximas de detección de obstáculos, en metros.

A.3.1.7. libAWS

Se utiliza el SDK de Amazon Web Services para Python junto con la librería Threading para implementar las siguientes funciones:

- ***myShadowUpdateCallback(payload, responseStatus, token)***: esta función de *callback* es llamada cada vez que se hace una actualización del estado del dron realizando cambios en un documento JSON compartido. Se encarga de asignar los nuevos valores a variables propias, las cuales pueden ser leídas por otras funciones para, por ejemplo, mostrar el reporte en tiempo real de la interfaz gráfica.
- ***threadAWS(queueTag, printUpdate)***: *thread* que se encarga del monitoreo del *queue* correspondiente al documento JSON del dron, una vez realizados cambios en el mismo, se hace una actulización en el servidor de AWS del *shadow* del dron. El parámetro *queueTag* es el *queue* donde se comparte el JSON del estado del dron. El parámetro *printUpdate* es una bandera de debuggeo que habilita imprimir los cambios detectados cada vez que se llama a la función de *callback* anterior.

A.3.2. Software de la base

En el diagrama A.5 se puede observar las distintas jerarquías de los *scripts* y librerías implementadas para la base.

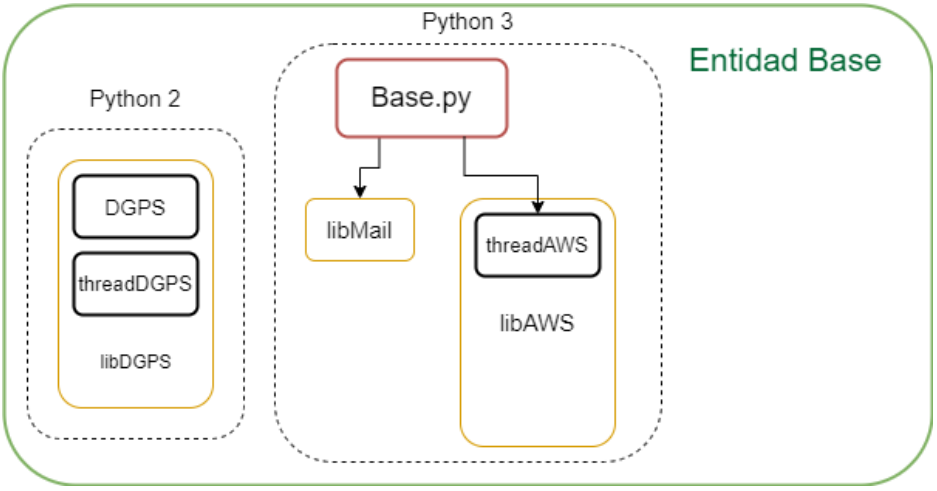

Figura A.5: Jerarquía del software para la entidad Base. En rojo se encuentra el código principal de la entidad; en amarillo las librerías utilizadas por éste, y en negro los *threads* implementados en dichas librerías. Las flechas apuntan desde el objeto de mayor jerarquía al menor, es decir, los objetos en las puntas de las flechas son incluidos o están dentro de los objetos en el otro extremo de la flecha.

A.3.2.1. libDGPS

La base física debe correr dos *scripts* de Python para la corrección diferencial de GPS. Las funciones dentro de estos *scripts* no son incluidas dentro del código principal

de la base ya que este último está escrito para Python 3 y los códigos de DGPS están en Python 2, y no es posible adaptarlo a Python 3.

A continuación se describen estos dos scripts y la forma de utilizarlos:

- ***DGPS()***: se encarga de realizar la conexión hacia la dirección IP del servidor que se use para la corrección diferencial, realizar el pedido HTTP con la estación a la cual conectar, el usuario y la contraseña utilizados. Una vez verificadas las credenciales del usuario y encontrada la estación solicitada, comienza a recibir los mensajes RTCM de corrección del servidor. Luego, se redirige los mismos hacia un puerto abierto del equipo local, en este caso, 13320 del *localhost* para poder recibir los mensajes con el siguiente *script* de la librería, *treadDGPS*. Este *script* debe ser corrido en terminal agregando los siguientes parámetros:

  ```
  python2 DGPS.py -u termodron -p termodron1 -v
  --UDP 13320 rtk.sgm.gub.uy 2101 UYMO
  ```

  Donde luego de -u se indica el usuario, -p indica la contraseña, -v indica modo *verbose* para ver mensajes de error en terminal, –UDP indica a qué puerto redirigir los mensajes, luego se especifica la dirección ip o dirección web del servidor, el puerto del servidor al cual conectar, y finalmente, el *mountpoint* o estación a la cual conectar.

- ***threadDGPS(connection_string)***: este *thread* se encarga de abrir un puerto local para recibir los mensajes de corrección diferencial, realizar una conexión por MAVlink mediante el transmisor de radiofrecuencia conectado al puerto especificado por el string *connection_string*, y luego de confirmada dicha conexión, redirigir los mensajes RTCM recibidos por el puerto 13320 hacia el dron mediante este enlace MAVlink.

A.3.2.2. Código principal de la base

En esta sección se describirán las funciones utilizadas en el bloque principal de la base.

Se implementan varias funciones de recepción y envío de correos electrónicos basadas en las funciones *recv_mail_monitor* y *send_mail_monitor* de la librería libMail, simplemente variando el contenido a enviar o el asunto del correo a recibir, de forma de mantener el código final en forma más legible.

- ***receive_mission_params()***: función mediante la cual se llama a la función *recv_mail_monitor()* con los parámetros específicos para recibir la configuración de la misión. Retorna error en caso de no poder recibir el correo luego de un tiempo de timeout, o el JSON con el contenido de la misión en un diccionario del tipo JSON en caso contrario.

- ***request_dron_status()***: función para enviar un pedido de reporte de estado hacia el dron, previo al vuelo. Devuelve error si no logra enviar el correo luego de un tiempo de timeout.
- ***receive_dron_status()***: función para monitorear correos entrantes por un correo de reporte de estado del dron. En caso de recibirlo retorna un diccionario del tipo JSON con el estado del dron. En caso contrario retorna error.
- ***send_mission(mission_tag)***: función para enviar el documento JSON (en la variable *mission_tag*) hacia el dron una vez que se confirma que el estado del mismo es apto para el vuelo. Retorna error si no se logra enviar el mensaje.
- ***receive_mission_begin()***: función para monitorear correos entrantes en búsqueda de un correo de comienzo de misión del dron. En caso de recibirlo devuelve el string "begin", de lo contrario, devuelve error.
- ***receive_mission_end()***: función para monitorear correos entrantes en búsqueda de un correo de fin de misión del dron. En caso de recibirlo devuelve el string "end", de lo contrario, devuelve error.
- ***receive_mission_logs()***: función para monitorear correos entrantes en búsqueda de un correo de envío de logs de vuelo e imágenes térmicas por parte del dron. En caso de recibirlo devuelve un string con la ruta hacia la carpeta que contiene los adjuntos, de lo contrario, devuelve error.
- ***send_error_msg()***: función para enviar mensaje de error hacia el usuario. Se utiliza cuando alguna de las funciones de comunicación de la base (o del dron) no logran enviar o recibir el correo correspondiente, agotando el tiempo de *timeout* especificado, y deteniendo las máquinas de estado de ambas entidades, requiriendo así un reseteo manual por parte de un operario.

Para ver más detalles de la máquina de estados implementada para la base, ver 5.3.

Debe mencionarse nuevamente que el software encargado de controlar el hardware de la base no fue implementado ya que la misma fue solamente diseñada a modo de concepto y no era un objetivo del proyecto implementar la misma.

A.3.3. Software de la interfaz

En la interfaz se utilizó la librería Tkinter propia de Python, con las principales funciones mencionadas a continuación:

- ***tk.Tk()***: función que genera la ventana principal de la aplicación, se define una variable de la forma "variable" = tk.Tk() con el fin de utilizar la variable para ubicar *widget* en la ventana principal.

- ***tk.mainloop()***: Genera el loop principal con el cual se mantiene la ventana abierta.
- ***tk.Toplevel()***: Genera una ventana nueva.
- ***tk.title()***: Genera el titulo de la aplicación.
- ***tk.Frame()***: Genera un nuevo frame dentro de la ventana donde se utilice, es de utilidad para separar la ventana en distintos espacios.
- ***tk.resizable()***: Función utilizada para poder modificar el tamaño de las ventanas, se pasa como parámetro 1 o 0 para activar o desactivar respectivamente el reajuste del ancho y largo.
- ***tk.Menu()***: Genera el *widget* del menú superior.
- ***tk.add_cascade()***: función auxiliar al menú con el fin generar las distintas opciones de la barra de menú.
- ***tk.add_command()***: Agrega comandos a cada opción de la barra de menú.
- ***tk.Scrollbar()***: función que genera el *scrollbar* con el cual moverse sobre la ventana de la aplicación.
- ***messagebox.showinfo()***: función que se utiliza para desplegar información en una ventana emergente
- ***tk.PhotoImage()***: función con la cual se ingresa una imagen a la aplicación.
- ***tk.Label()***: función para crear un *widget* de texto.
- ***tk.place()***: función que se utiliza para ubicar los *widget* en sus posiciones en la ventana.
- ***tk.Combobox()***: función que despliega un *widget* de tipo lista desplegable, con el cual se puede tener distintos tipos de valores.
- ***tk.Button()***: función con el cual se crea un botón.
- ***tk.Entry()***: función con la cual se crea una entrada de datos, se debe utilizar la función get() para obtener los datos ingresados.
- ***tk.get()***: Se utiliza la función get() cada vez que se quiera obtener datos del tipo Entry.
- ***tk.Text()***: función con la cual se ingresa texto.

- ***tk.StringVar()***: función que crea una variable String la cual se despliega en pantalla, se utiliza la función set() para setear la variable. Por ejemplo se utiliza variables StringVar para configurar las variables del reporte AWS en la interfaz y cada vez que se obtienen datos del servidor AWS se utiliza la variable set para cargar el nuevo dato.

A.4. Anexo IV: Presupuesto y Cronograma

Se contó con un presupuesto de 2000 dolares y originalmente se siguió el cronograma presentando en la figura A.6

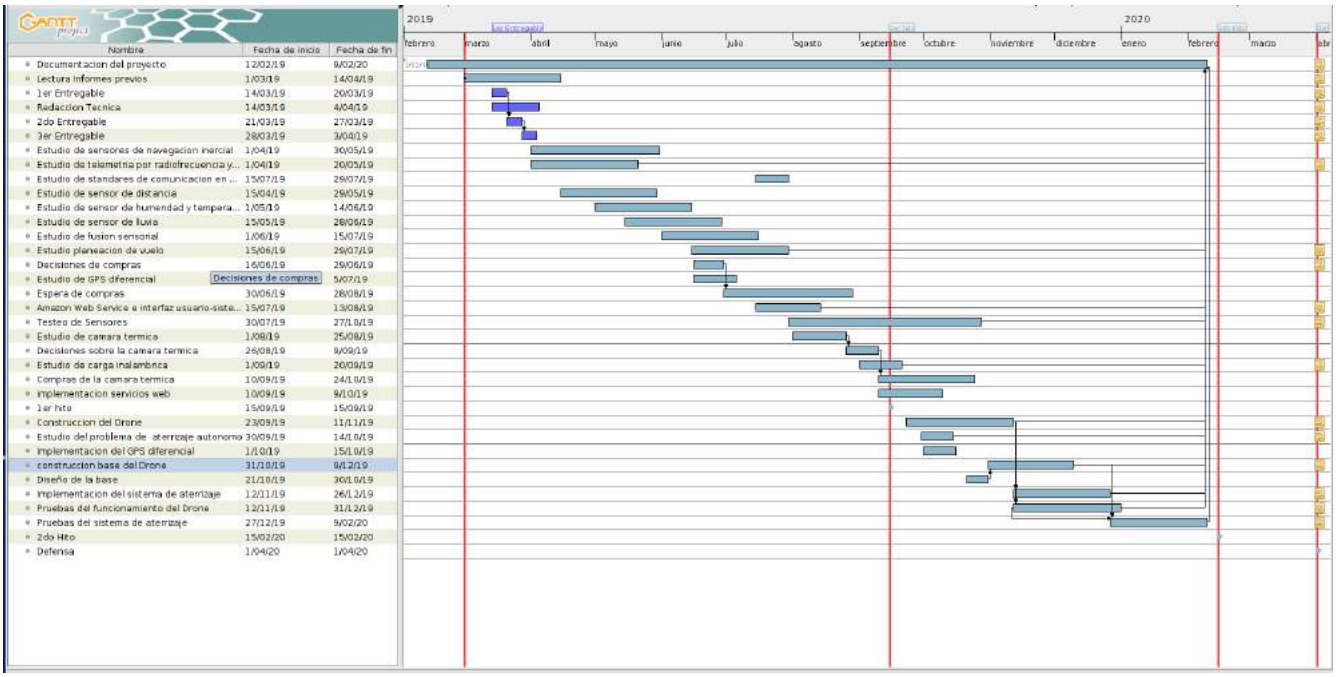

Figura A.6: Gantt realizado para el cronograma del proyecto.

Dada la situación del país debido al virus COVID-19, y la resolución del gobierno y el rector de la Universidad, y ante la imposibilidad de concurrir a la facultad, y, dado que el grupo se encontraba en una etapa cercana al final del proyecto, donde eran necesarias múltiples pruebas, debió detenerse esa fase experimental.
La fecha en la que se detuvo la fase experimental fue el 13 de marzo de 2020.

Durante el período en que no se pudo acceder al edificio de Facultad se planteó el siguiente plan de trabajo:

- Avanzar todo lo que sea posible con la documentación del proyecto, iterando versiones según el feedback obtenido del tutor

- Lograr un diseño conceptual de la base del dron, lo que incluye:
  - Modelado 3D
  - Explicación de sistema mecánico para asistir el alineamiento de la bobina del dron con la de la base para asegurar un buen acople para la recarga inalámbrica.
  - Descripción de los componentes de hardware que la componen
  - Descripción del software que la compone.
- Revisión, corrección y documentación de los códigos escritos hasta el momento

El grupo de proyecto decidió volver a Facultad a realizar pruebas experimentales con el dron el día 21 de mayo de 2020. Las mismas se realizaron durante los fines de semanas, por razones de horario laboral, y manteniendo las medidas de higiene y protección necesarias.

Teniendo en cuenta el tiempo limitado por semana para realizar pruebas, así como una dependencia de las condiciones climáticas, se requiere de más tiempo para continuar esta última fase experimental del proyecto

A.5. Anexo V: Manual de usuario

En el presente manual se describen las prestaciones del sistema implementado, así como los pasos necesarios para poder realizar un vuelo de forma segura y efectiva, y la correcta recepción de los datos al terminar un vuelo, o durante éste.

A.5.1. Instrucciones de uso

En los archivos del proyecto se incluyen tres carpetas que contienen el software de cada entidad del sistema: interfaz, base, dron. El código del dron ya está implementado dentro de la Odroid XU4, y el código de la base se encuentra en la Raspberry Pi. El usuario deberá copiar los contenidos de la carpeta "Interfaz" hacia la computadora donde vaya a correr el programa.

Para utilizar la interfaz se debe clickear en el ícono llamado "Termodron III - UI" y luego clickear en el botón "Seteo de misión". En la nueva ventana puede elegir los parámetros configurables de la misión a realizar. Para más detalles de los mismos ver **??**. El programa es provisto con la visualización de datos por AWS en tiempo real desactivada, ya que el usuario debe crear su cuenta propia y descargar los certificados correspondientes hacia la carpeta "certificates" como se indica en A.5.3.7. En caso de querer generar un nuevo ejecutable con esta funcionalidad activada se debe correr en una terminal [1]

Una vez elegidos todos los parámetros se debe clickear en el botón "Enviar", el cual enviará un correo con la misión elegida hacia las direcciones ingresadas, si las hay, y hacia la base del dron. La Raspberry Pi de la base y la Odroid XU4 deben estar encendidas y conectadas a internet para que puedan monitorear los correos entrantes.

En caso de pérdida o formateo de las computadoras a bordo, se deben quemar las imágenes de los sistemas operativos provistas en los archivos del proyecto hacia las tarjetas SD cada una, utilizando un programa apropiado, como por ejemplo *Etcher*. Luego, se deben agregar los scripts "Base.py", "Dron.py" en las listas de scripts a ejecutar luego del encendido, respectivamente.

A.5.2. Descripción del producto

A.5.2.1. Introducción

El sistema implementado consta del dron (siendo la base un diseño teórico solamente). Sus principales características se listan a continuación.

[1] Se debe tener instalado el módulo *pyinstaller* de Python3, pero se recomienda correr en una de las computadoras de la base o dron, las cuales cuentan con todos los módulos ya instalados.

Dron	
Especificaciones Físicas	
Peso (g)	3000
Armazón (mm)	550[2]
Altura (mm)	350
Velocidad de ascenso máximo(m/s)	5
Velocidad de descenso máximo (m/s)	5
Velocidad máxima horizontal máxima (m/s)	5
Altitud máxima (m)	100
Tiempo de vuelo (min)	10
Batería	
Cantidad de celdas	3
Capacidad (mAh)	10000
Alimentación (V)	11.1
Tasa de carga (C)	25
Peso (g)	600
Motores	
Alimentación (V)	11.1
Empuje máximo (g)	1210
Consumo máximo (A)	15
Peso (g)	102
Hélices	
Material	Fibra de carbón
Dimensiones[3]	14x4.7
Peso (g)	20
GPS	
Modelo	Here 2
Precisión s.corrección (m)	> 2
Precisión c.corrección (m)	$< 1{,}5$
Capacidad de GPS diferencial	Si
Peso (g)	49
Controlador de vuelo	
Modelo	Pixhawk Cube
Procesador	ARM Cortex-M4F
Peso (g)	200

Cuadro A.1: Especificaciones del dron.

Computadora a bordo	
Modelo	Odroid XU4
Alimentación (V)	5
Consumo máximo (A)	6
Procesador	Exynos 5422 Octa big.LITTLE ARM Cortex-A15 @ 2.0 GHz quad-core and Cortex-A7 quad-core CPUs
RAM (GB)	2
Memoria (tarjeta SD) (GB)	64
Cantidad puertos USB	3
Peso (g)	60
Sistema operativo	Ubuntu Mate 18.04
Versión de Python	3.6
Cámara Térmica	
Modelo	FLIR Lepton 3.5
Resolución	160x120
Rango máximo (m)	
Radiometria	Si
Interfaz	SPI, micro USB[4]
Método de detección	Focos de calor o cuerpos calientes
Rango de temperatura para cuerpos calientes (C)	30-45
Rango de temperatura para focos de calor (C)	> 100
Peso (g)	3
Cámara Estereoscópica	
Modelo	Intel RealSense D415
Resolución[5]	640x480 (RGB), 424x240 (stereo)
Rango máximo (m)	10
Interfaz	USB-C
Peso (g)	72
Sistema de aterrizaje de precisión	
Cantidad de marcadores a usar	2
Diccionario marcador 1	ArUco Original
Tamaño marcador 1 (cm)	26.5
ID marcador 1	72
Rango de uso marcador 1	> 2 metros
Diccionario marcador 2	4X4_50
Tamaño marcador 2 (cm)	10

Cuadro A.2: Especificaciones del dron.

ID marcador 2	0
Rango de uso marcador 2	< 2 metros
Velocidad de descenso (m/s)	0.1
Sistema de evasión de obstáculos	
Tecnología	Cámara estereoscópica y sensores de ultrasonido
Modo de detección	Cámara con 3 regiones de detección, sensores de emergencia y algoritmo de toma de decisiones
Rango de detección (m)	9
Velocidad de evasión (m/s)	1

Cuadro A.3: Especificaciones del dron.

En las figuras A.7, A.8 y A.9 se presenta el dron con sus principales componentes, estos se numeran en las imágenes y se detallan a continuación.

Figura A.7: Imagen del dron con sus principales componentes numerados.

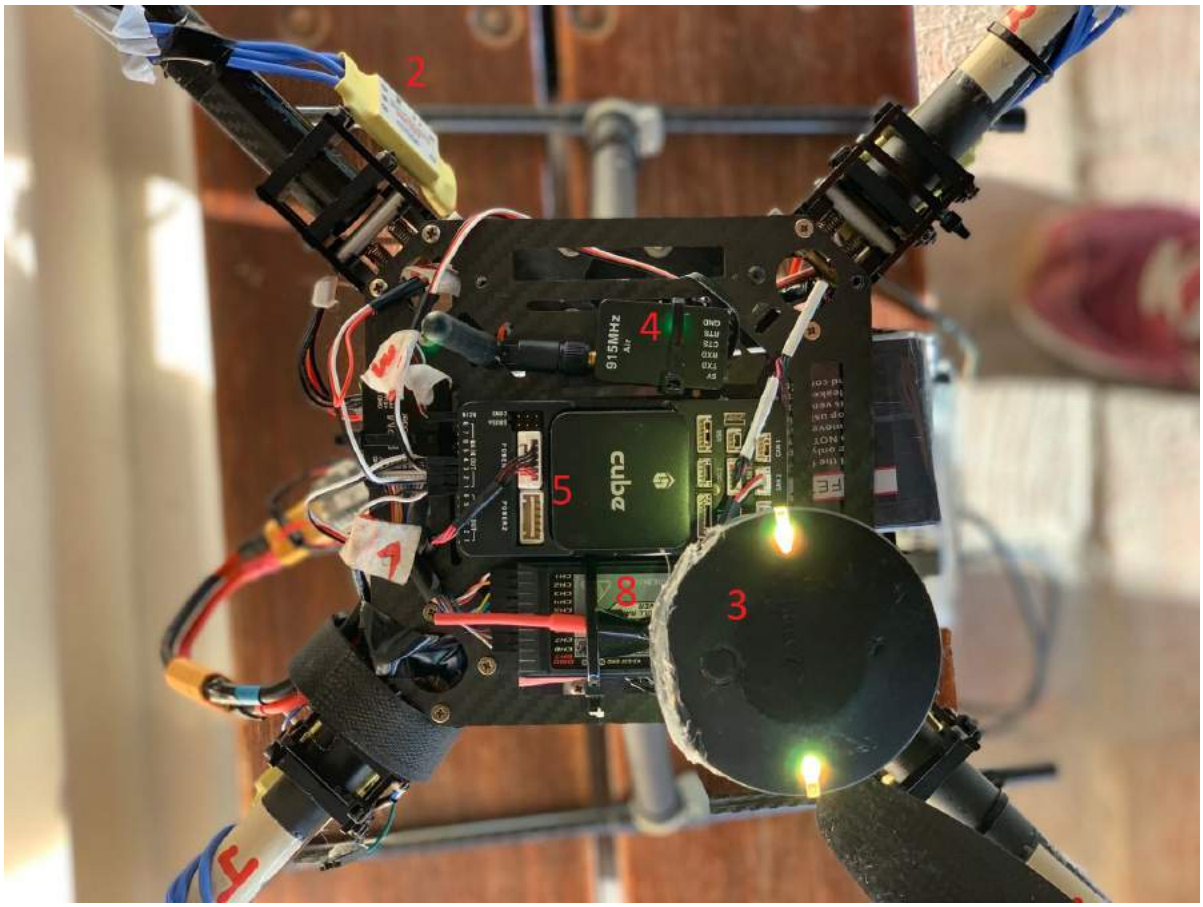

Figura A.8: Imagen del dron con sus principales componentes numerados.

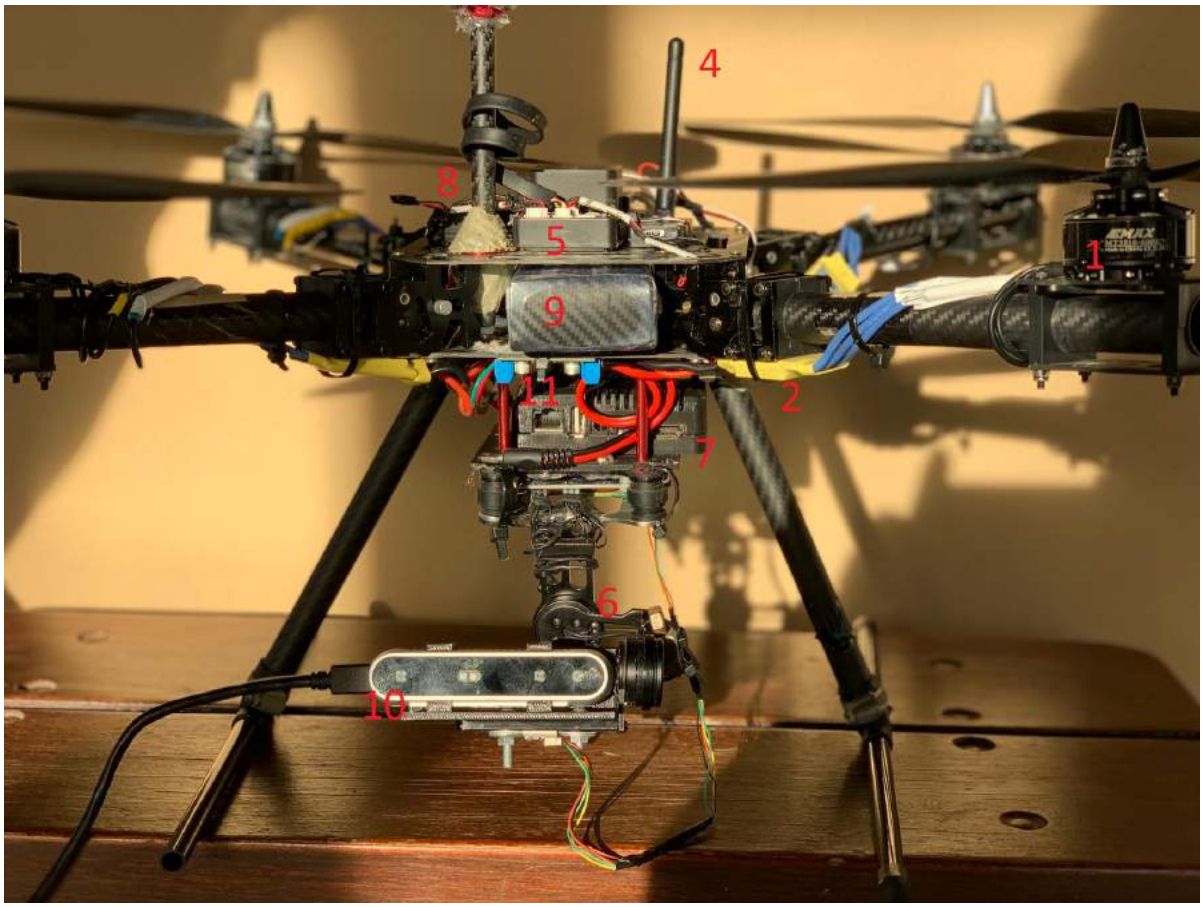

Figura A.9: Imagen del dron con sus principales componentes numerados.

1. Conjunto de 4 motores y hélices
2. Conjunto de 4 ESC
3. Módulo GPS
4. Antena de telemetría 915MHz
5. Controlador de vuelo
6. Gimbal
7. Computadora a bordo
8. Receptor de control remoto
9. Batería
10. Cámara estéreo
11. Regulador de voltaje

A.5.2.2. Características principales

Control de usuario

El dron está diseñado para poder realizar misiones de forma autónoma, no requiriendo que el usuario esté en la cercanía del mismo. Se utiliza para esto el sistema de correo electrónico, o e-mail, mediante el cual el usuario envía hacia la base un documento JSON con los parámetros de la misión a ejecutar. La base monitorea constantemente su casilla de correo hasta recibir dicho documento. La base entonces se comunica con el dron mediante el envio de e-mails para determinar el estado del mismo, en caso de estar listo para volar, le envía por dicho medio el documento JSON y espera un asentimiento del dron para luego enviar la instrucción de comienzo de misión al dron.

Control del dron

El dron cuenta con un controlador de vuelo Pixhawk 2.1 o Pixhawk Cube, el cual se encarga de relevar información de los sensores del dron, y, en base a la información de éstos, comandar los motores del mismo. Este controlador de vuelo se encuentra programado con el firmware de ArduCopter en su última versión (4.0.2 al mes de marzo de 2020).

Por otro lado, el dron también posee una computadora a bordo, en este caso, una Odroid XU-4, la cual se encarga de toda la lógica de comunicación entre los varios periféricos del sistema (cámaras, sensores de distancia, gimbal), el controlador de vuelo, la base, y, opcionalmente, la comunicación directa con el usuario mediante AWS o mail, durante el vuelo. También se encarga del procesamiento de imágenes utilizado por ambas cámaras, tanto para la detección de cuerpos calientes, como para la detección y evasión de obstáculos. Todo el código implementado se encuentra programado en Python.

A.5.2.3. Diagrama del sistema, capas y estados del dron

En la figura A.10 se observa el diagrama del sistema implementado.

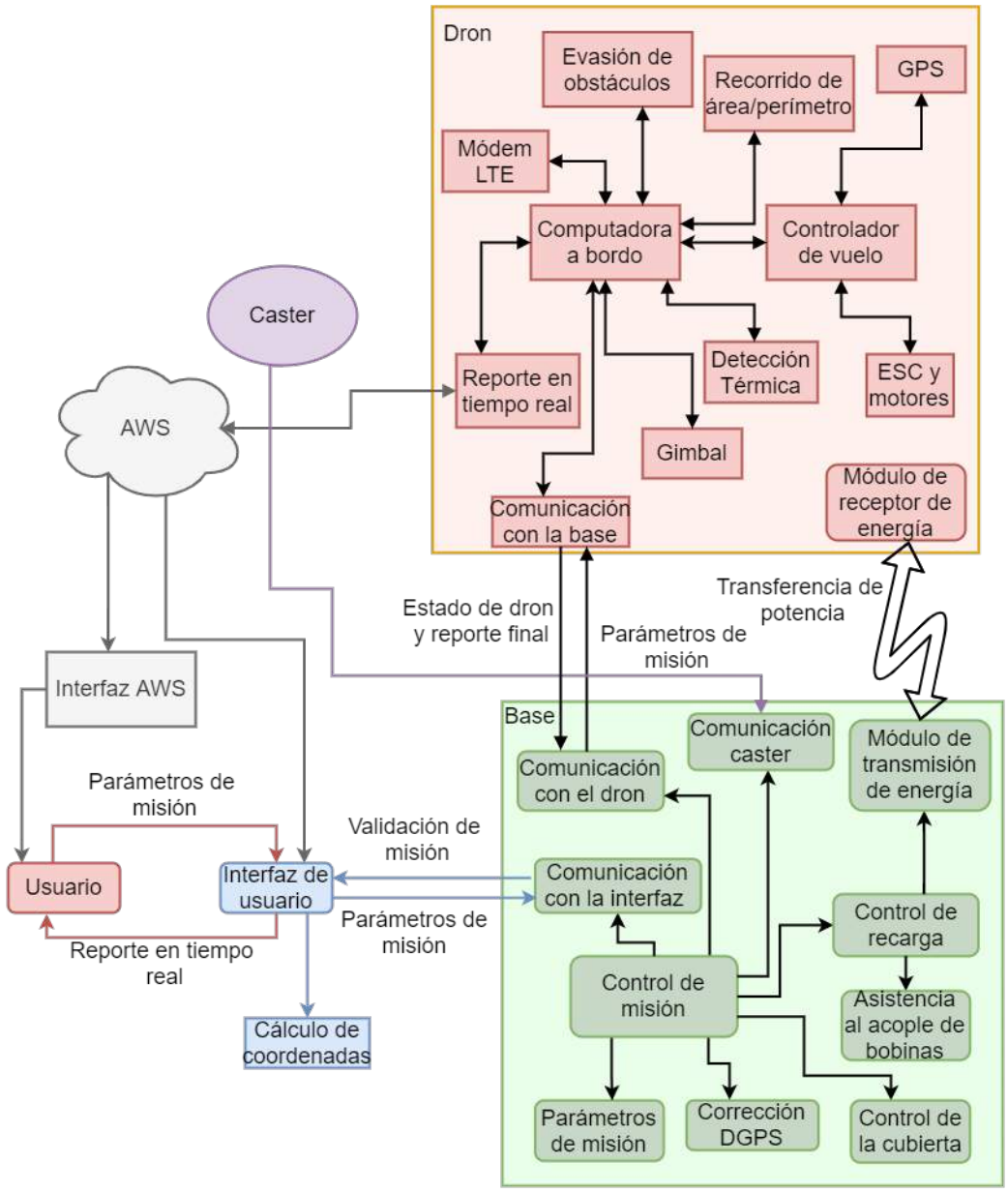

Figura A.10: Diagrama del sistema completo.

En la figura A.11 se observa el diagrama de capas del sistema implementado, donde se observa la relación entre el hardware y el software.

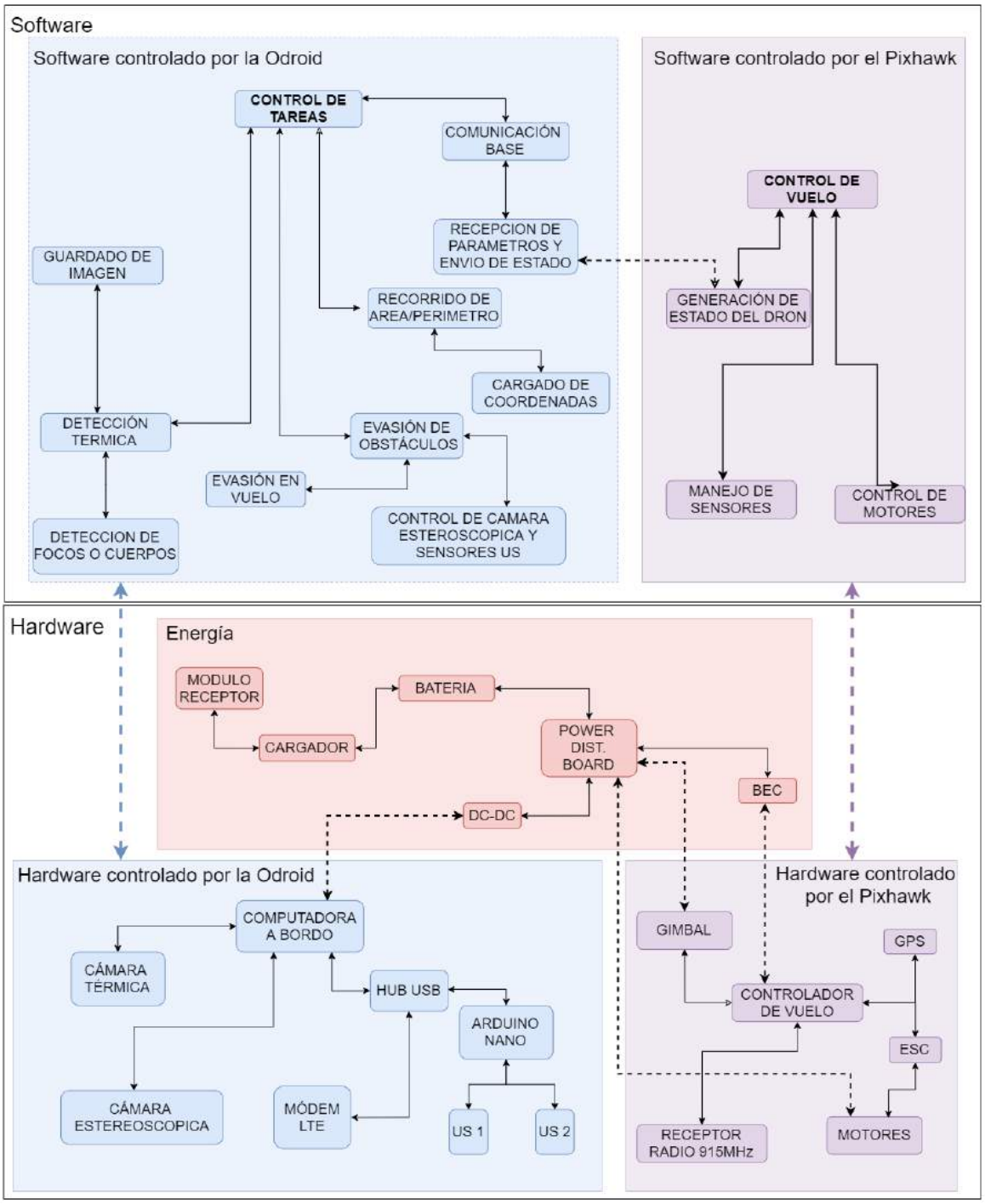

Figura A.11: Diagrama de capas del dron.

En la figura A.12 se observa la maquina de estados del dron y en la figura A.13

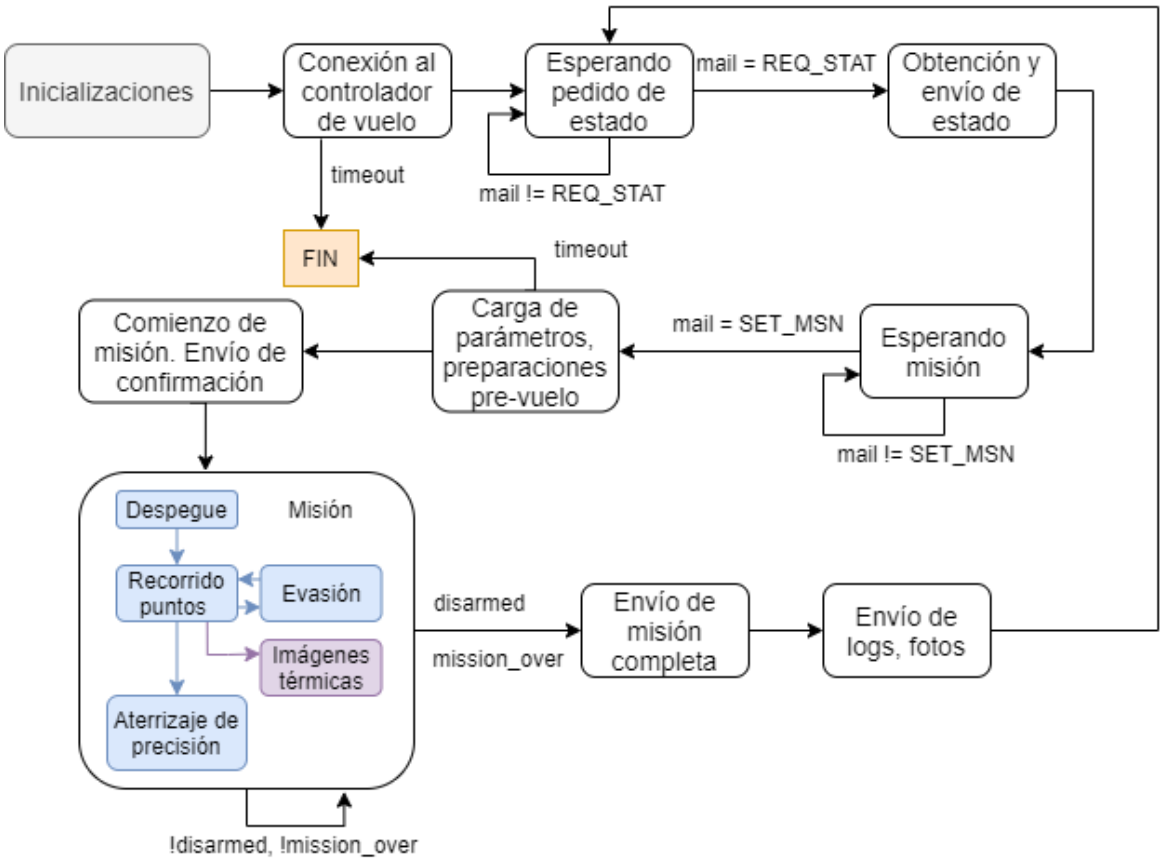

Figura A.12: Máquina de estados del dron.

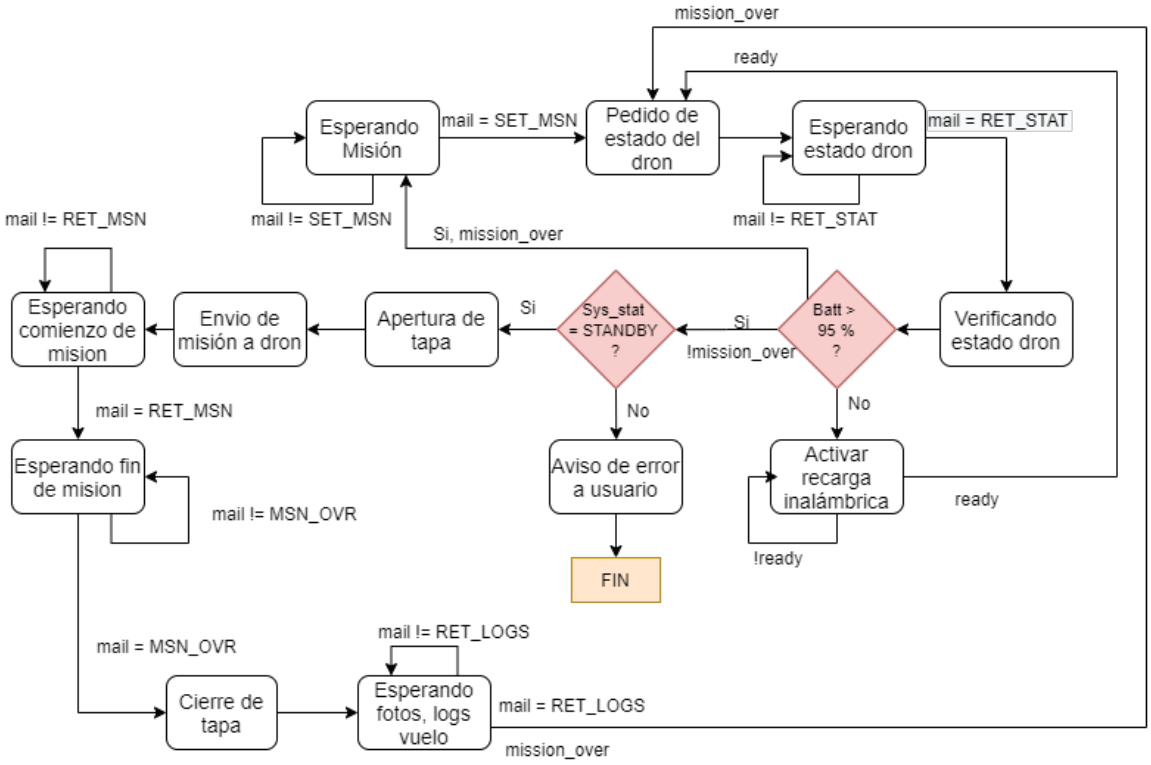

Figura A.13: Máquina de estados de la base.

A.5.3. Funciones del producto

A.5.3.1. Detección térmica

El dron utiliza una cámara FLIR Lepton 3.5 montada en una placa PureThermal V2.0 la cual agrega la capacidad de utilizar la cámara mediante USB, SPI y cargar firmware a la misma. Esta cámara tiene una resolución de 160x120 píxeles, y se devuelve al usuario en formato JPG en escala de grises, con un círculo rojo encerrando el mayor objeto cuya temperatura está en el rango elegido, dentro de la imagen.

Se puede elegir entre dos modos de detección: detección de cuerpos calientes (objetos con temperaturas entre 30 °C y 45 °C), o detección de focos de calor (temperaturas mayores a 100°C). Por defecto se utiliza el modo de detección de cuerpos calientes, para cambiar de modo o desactivar el uso de la detección térmica, se debe seleccionar el parámetro correspondiente durante la configuración de misión.

Si se desea recibir dichas imágenes por mail en tiempo real, se debe especificar esto e incluir los correos de los destinatarios al preparar la misión.

La configuración de estos parámetros se realiza mediante la interfaz gráfica, la cual se detalla en A.5.3.7.

A.5.3.2. Aterrizaje de precisión

El aterrizaje de precisión es parte de toda misión que realice el dron, el mismo le da el cierre a la misión posicionando de forma segura el dron sobre la base con el objetivo se su protección y preparación para la próxima misión.

Para poder realizar el aterrizaje de precisión se elaboro un algoritmo de control del dron mediante el uso de una cámara de espectro visible y dos marcadores ArUco posicionado sobre la base.
El algoritmo realiza un loop en el cual mediante el uso de la cámara, detecta el marcador, obtiene la posición del marcador respecto de la cámara y en caso de encontrarse posicionado sobre la base con un error menor a un cierto umbral configurable se le ordena descender, en caso de no estar posicionado con un error menor al umbral se le ordena al dron centrarse sobre la base.

Para poder realizar un aterrizaje preciso es crucial que tanto el gimbal como la cámara se encuentren calibrados de forma correcta, ver secciones A.5.6 y A.5.7.

A.5.3.3. Detección y evasión de obstáculos

El dron posee una cámara estereoscópica (Intel RealSense D415) y un sensor de ultrasonido para detección de obstáculos, siendo la primera el método primario de detección, y el segundo un respaldo en caso de fallas de la cámara.

Se utiliza una cámara estéreo para poder tener un sistema más robusto de detección de obstáculos, ya que los sensores de distancia usualmente utilizados en aplicaciones de robótica móvil tienen algunos materiales, o condiciones en las cuales no funcionan adecuadamente. Por ejemplo, los sensores de distancia por ultrasonido funcionan enviando un pulso ultrasónico por el aire, el cual rebota en el objeto que esté delante del mismo y vuelve al sensor, dependiendo del tiempo que demora en retornar el pulso se puede estimar la distancia al objeto, pero este método es sensible a materiales que impidan el rebote de la onda sonora incidente, como ser por ejemplo, la corteza de los árboles. Por otro lado, los sensores infrarrojos o láser funcionan con un principio similar a los de ultrasonido, en que miden el tiempo de vuelo de una onda lumínica, en este caso, y por ende también sufren en caso de materiales no reflectivos (como papel, por ejemplo), y su performance se degrada ante la luz solar.

A.5.3.4. Modos de vuelo

El software del controlador de vuelo utilizado, ArduCopter (variante para cuadricópteros de ArduPilot) posee 23 modos de vuelo. A continuación se listan y se describen los utilizados por Termodron III.

- Guided: modo de vuelo para control fino del movimiento del dron. Es el utilizado cuando se desea controlar explícitamente el movimiento del dron, como ser durante el despegue, evasión de obstáculos o el aterrizaje de precisión.

- Auto: modo de vuelo automático donde el dron recorre los puntos geográficos guardados en su memoria previamente.
- Land: reduce la altura del dron hasta llegar al suelo, intentando mantener el descenso en una línea recta vertical. Utilizado para terminar vuelos o como medida de seguridad ante fallas.

Para la calibración y testeo del dron pueden ser útiles los siguientes modos de vuelo:

- RTL: retorna el dron hacia el home, siendo éste el punto donde por primera vez logra obtener señal de GPS (puede ser cambiado por software también). Puede ser utilizado para terminar vuelos y como medida de seguridad ante fallas.
- AutoTune: modo utilizado para la calibración del controlador PID de los ángulos de movimiento del dron (*yaw*, *pitch* y *roll*). Se debe despegar el dron en algún modo, pasar al modo AltHold y luego pasar a este modo para comenzar la calibración del PID. Para más información ver [51].
- AltHold: mantiene la altitud y auto nivela los ángulos de *roll* y *pitch*. Utilizado para calibración de PID con modo AutoTune.

Para más detalles sobre los modos de vuelo ver [52].

A.5.3.5. Registros de vuelos

Los registros, o *logs* de vuelo son registros que genera el controlador de vuelo cada vez que el dron se arma o desarma. Son almacenan en la tarjeta SD del controlador de vuelo y pueden ser extraídos a través de un software de control de misión con el fin de realizar un análisis de los vuelos realizados.

Por defecto el uso de registros de vuelo viene activado y se guarda un registro por cada vez que se arma el dron, pero esto se puede variar modificando los siguientes parámetros:

1. LOG_BITMASK: máscara de bits con el cual se especifica que items van a ser registrados en los logs. Con la máscara en cero se deshabilitan los logs. Mas información sobre los bits de la máscara se puede ver en [53]
2. LOG_DISARMED: si este parámetro tiene valor 1 los logs se comienzan a guardar cuando se energiza el controlador de vuelo y no cuando se arma por primera vez el dron. Este parámetro es útil en caso de fallos al momento de armar el dron.
3. LOG_FILE_DSRMROT: si este parámetro tiene valor 1 al desarmar el dron finaliza el log que estaba siendo registrado, en caso que el parámetro LOG_DISARMED esté con valor 1, el controlador espera quince segundos y vuelve a crear un log nuevo.

Para extraer los registros de vuelo se debe realizar el siguiente procedimiento:

1. Abrir el programa Mission Planner
2. Conectarse al dron mediante cable USB (recomendado) o antena de telemetría.
3. Ir a la pestaña de datos de vuelo, abrir *DataFlash Logs* y luego apretar el botón *Download DataFlash Log Via Mavlink*
4. Luego se despliega una pantalla en donde figuran todos los registros de vuelos y en donde se pueden descargar a la PC.

En la figura A.14 se observan los pasos del procedimiento para extraer datos.

Figura A.14: Procedimiento para extraer los registros de vuelo del controlador. Imagen obtenida de **ArduPilot**.

Una vez extraídos los registros de vuelo es posible analizarlos utilizando el mismo programa, Mission Planner. Para ello, en la misma pestaña *DataFlash Logs* se tienen dos botones (ver figura A.15): *Log Analysis* y *Review a Log*, mediante los cuales se puede realizar un análisis manual de los registros o un análisis automático de los mismos.

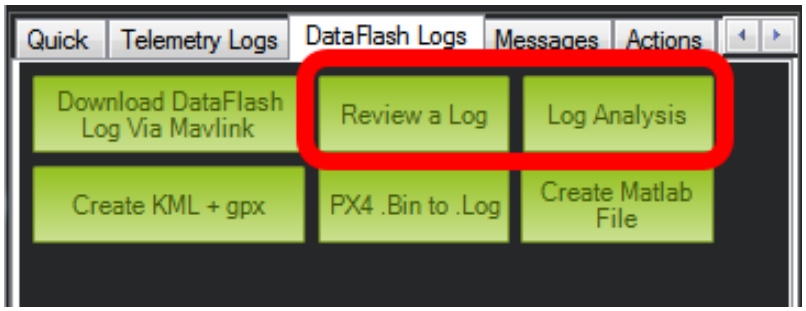

Figura A.15: Botones para realizar el análisis automático y manual de los registros. Imagen obtenida de **ArduPilot**.

El análisis automático es el mas sencillo de los dos, permitiendo un diagnóstico rápido de algunos problemas comunes, pero para realizar un testeo exhaustivo es recomendado el análisis manual de logs. Para hacer uso del análisis automático se debe apretar el botón *Log Analysis*, y se genera un reporte en el cual se indican distintos parámetros y su estado. En la figura A.16 se muestra un resultado obtenido del análisis automático.

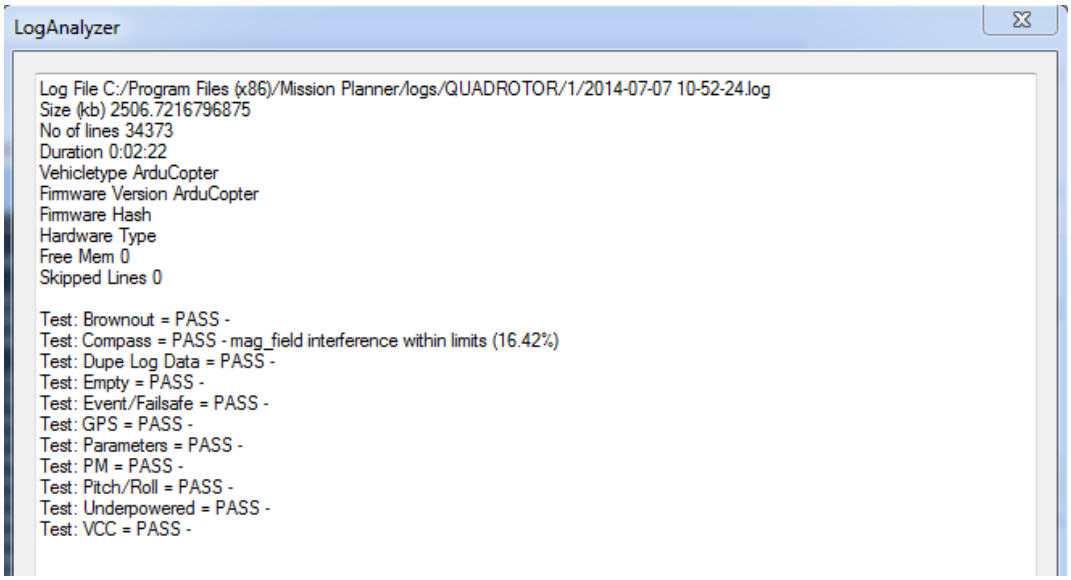

Figura A.16: Ejemplo de análisis del reporte automático. Imagen obtenida de **ArduPilot**.

Para un análisis mas detallado se debe utilizar el botón *Review a Log* mediante el cual se despliega una ventana con la cual se selecciona el registro de vuelo que se desea abrir. Una vez seleccionado se abre una nueva ventana mediante la cual se seleccionan los distintos parámetros y se despliegan los datos con sus respectivas gráficas. En la ventana de gráficas es posible hacer zoom manual, ver mensajes de eventos, errores, y modos de vuelo del dron. Además se tienen tablas en las cuales se indican los valores de cada parámetro que se desee ver. En la figura A.17 se detalla la ventana de modo manual.

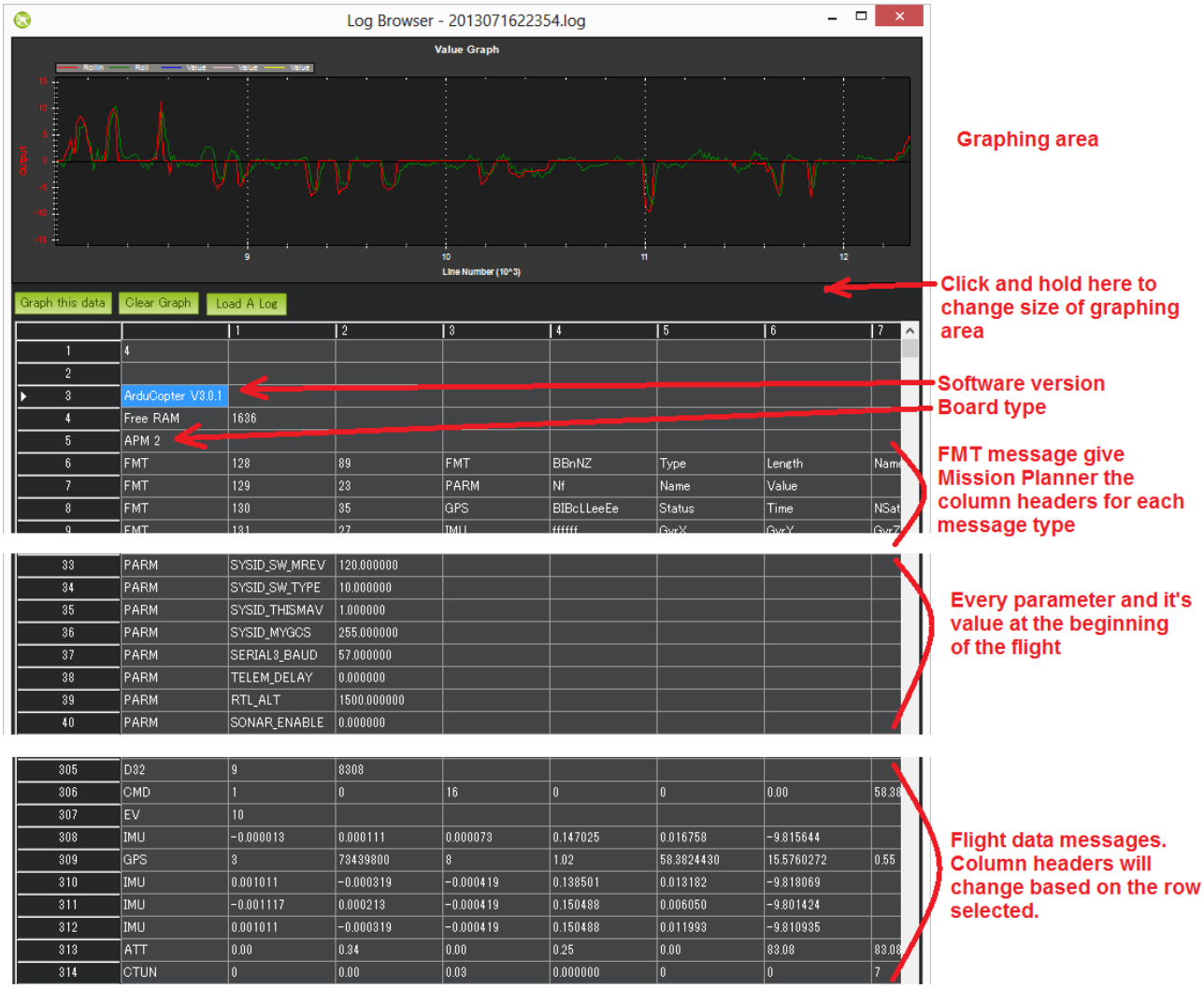

Figura A.17: Ejemplo de análisis del reporte manual. Imagen obtenida de **ArduPilot**.

A.5.3.6. Corrección de posición mediante DGPS

La corrección diferencial de GPS es realizada por la computadora de la base, la cual se encarga de conectar con la estación del Cerro, ingresar las credenciales correspondientes, y recibir los mensajes en formato NTRIP. Esto es realizado mediante un script obtenido de [54]. Luego se utiliza un script implementado por Termodron II que reenvía dichos mensajes hacia el dron mediante una conexión paralela de MAVlink.

Como el dron fue probado en vuelo en el predio de la Facultad de Ingeniería se conecta hacia la estación ubicada en la Fortaleza del Cerro, pero en caso de volar en otro punto del país basta buscar la estación más cercana, y cambiar el *mountpoint* de UYMO al indicado por la estación a utilizar, cuando se corre el script de NTRIP. En [55] se encuentran las estaciones activas en el país, así como su *mountpoint* y estado actual.

En caso de cambiar la estación a la que se conecta, se dejan a continuación las credenciales correspondientes a la cuenta de Termodron en la Red Geodésica Nacional Activa del país (REGNA-ROU) [56]:

- *user:* termodron
- *password:* termodron1

Deberá también cambiarse la estación a la que se contacta, y por ende, el mountpoint respectivo en el pedido que hace el script *NTRIP*. En los archivos del proyecto se incluye junto a los scripts de la librería libDGPS un archivo de texto que indica como utilizar los mismos.

A.5.3.7. Comunicación del sistema

La comunicación del sistema se puede dividir en tres grandes bloques, la comunicación Usuario-Base, la comunicación Base-Dron y el uso de Amazon Web Service. En la figura A.18 se presenta el diagrama de comunicación del sistema.

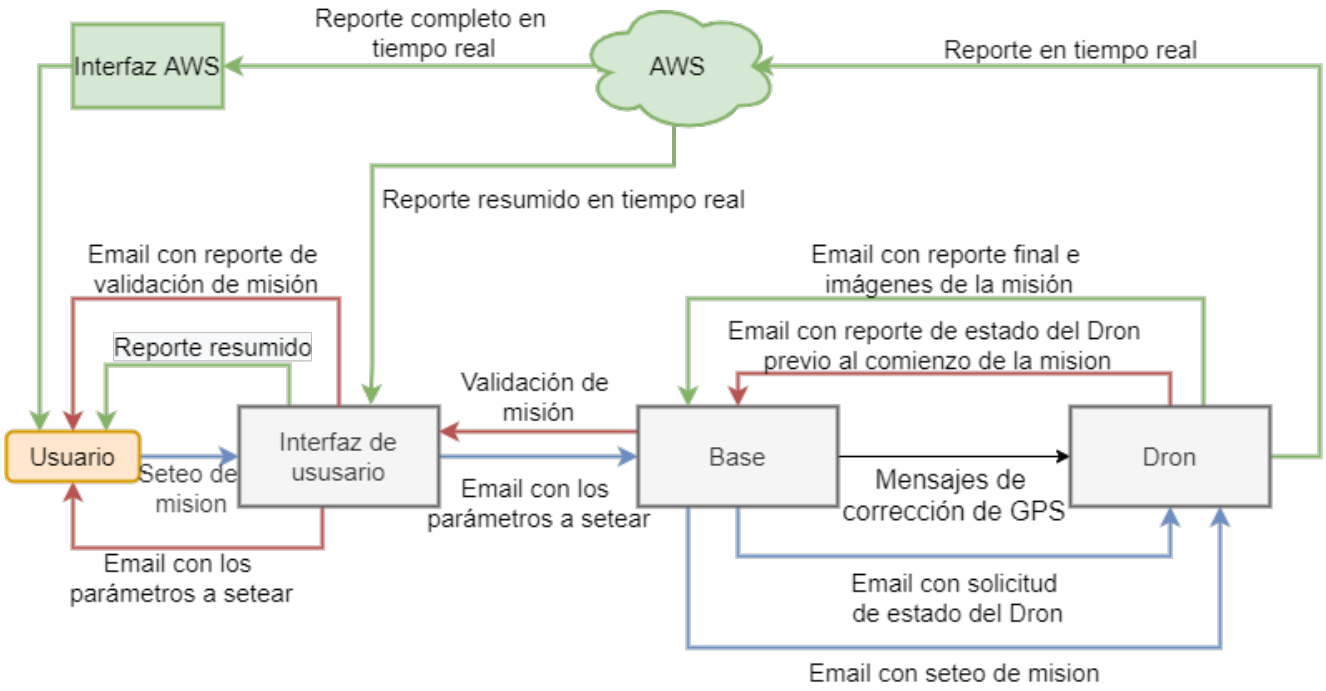

Figura A.18: Diagrama de comunicación del sistema implementado.

Comunicación Usuario - Base

La comunicación del usuario hacia la base se realiza mediante el uso de una interfaz

gráfica y el envío de correos electrónicos.
Se utiliza la interfaz para realizar una configuración de misión que sea sencilla al usuario.

Para poder ingresar a la interfaz se debe utilizar las siguientes credenciales:

1. usuario: termodron
2. contraseña: termodron

Luego para la configuración de la misión se debe acceder al bloque correspondiente de la interfaz, configurar los parámetros (pudiendo cargar las coordenadas desde un archivo .csv) y presionar el botón de enviar. En la figura A.19 se observa el bloque de seteo de misión.

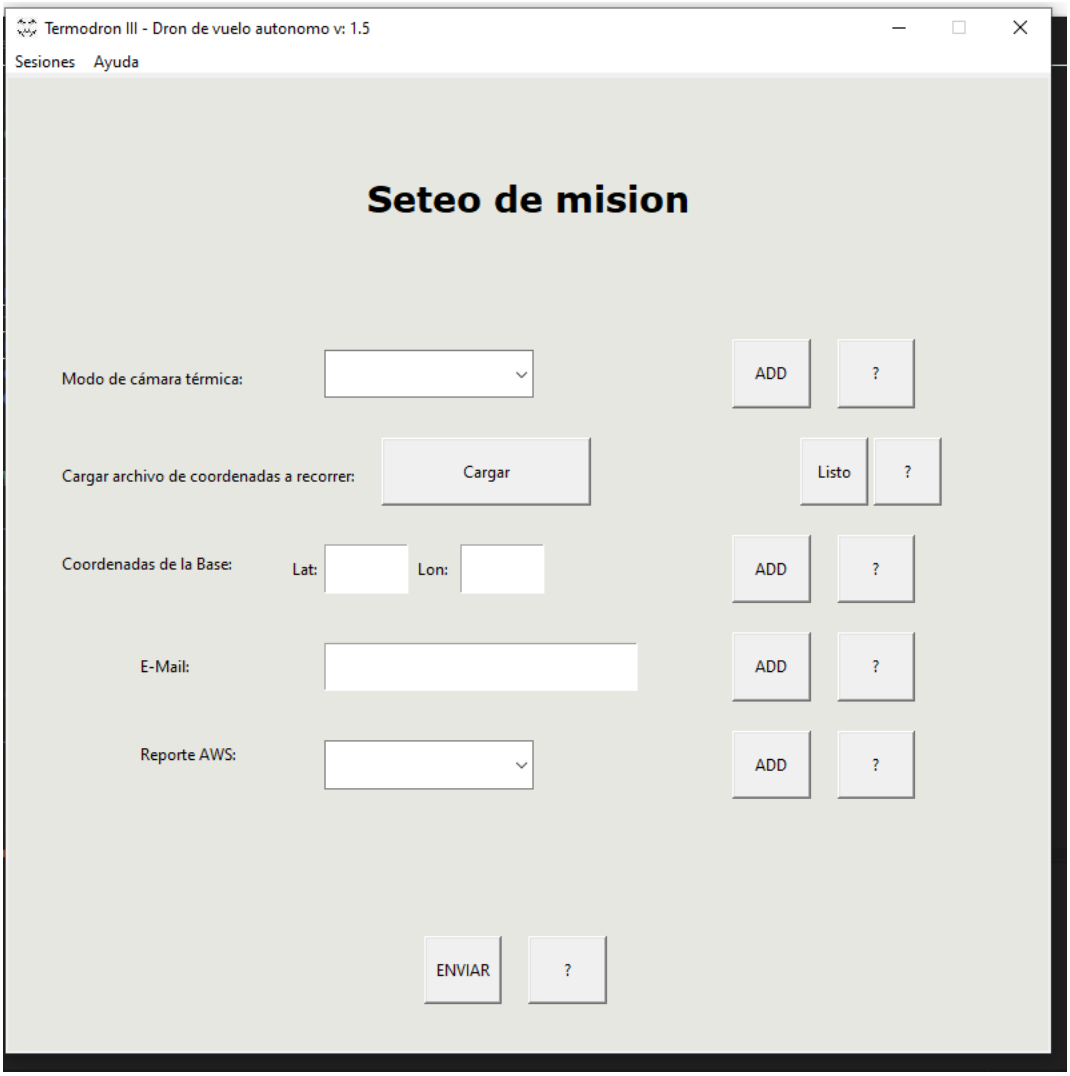

Figura A.19: Bloque de seteo de misión.

La interfaz utiliza el siguiente correo electrónico para el envío y la recepción de correos:

Mail = “termodron.interfaz@gmail.com”

Contraseña = “termo3.interfaz”

Comunicación Base - Dron

La comunicación entre la base y el dron se realiza por dos medios, por un lado se utiliza la radio de telemetría con el fin de enviarle al dron la corrección de GPS y por otro lado se utiliza el correo electrónico con el fin de realizar la comunicación previa y post misión.

Se crean dos correos electrónicos, uno para la base y otro para el dron, los cuales son:

Mail dron = “termodron.dron@gmail.com”

Contraseña = “Termo.3.dron”

Mail base = “termodron.base@gmail.com”

Contraseña = “Termo3Base”

La base se comunica con el dron en dos situaciones, primero para pedirle el estado del mismo (con el fin de ver si esta apto para un vuelo) y luego en caso de estar apto para volar se vuelve a comunicar enviándole los parámetros de la misión.

El dron, también se comunica en dos ocasiones con la base, primero con el fin de responderle a la base la solicitud de estado del mismo y luego al finalizar la misión se comunica con el fin de otorgarle un reporte final de misión y las imágenes obtenidas en vuelo.

Amazon Web Services

El reporte en tiempo real mediante el uso de AWS se habilita con el seteo de la misión por parte del usuario en la interfaz gráfica.
El formato de datos que utiliza AWS para el seteo y reporte de datos en tiempo real es un JSON, el cual se presenta a continuacion:

```
{
  "state": {
    "desired": {
      "baseLocation": {
        "lat": [],
        "lon": []
      },
```

```
      "AWS":"",
      "mailList": [],
      "thermalMode": "",
      "waypoints": {
        "lat": [],
        "lon": [],
        "alt": ""
      }
    },
    "reported": {
      "location": {
        "lat": "",
        "lon": "",
        "alt": ""
      },
      "battery": "",
      "lastHeartbeat": "",
      "armable": "",
      "armed": "",
      "mode": "",
      "gpsStatus": ""
    }
  }
}
```

Se decide mantener el uso del JSON tanto en la comunicación Usuario-Base como en la comunicación Base-Dron para mantener la unicidad en el formato del mensaje a enviar, y permitiendo adaptar fácilmente el sistema a una comunicación única por AWS si se desea en el futuro.

Para crear un usuario en la plataforma AWS se debe ingresar a: `https://console.aws.amazon.com/iot/home?region=us-east-1#/thinghub` o con una simple búsqueda web se encontrará un enlace correspondiente.

Luego de tener una cuenta de AWS el siguiente paso es utilizar el módulo IoT de AWS para crear las *things*[6] respectivas a cada entidad del sistema: interfaz, base, dron, y hallar sus credenciales respectivas. Con estas credenciales es posible conectarse al servidor regional correspondiente de AWS para poder hacer uso del sistema de publicación/suscripción de AWS IoT, y, en particular, los scripts implementados para el sistema Termodron.

A continuación se detallará cómo realizar dicho procedimiento. Se recomienda fuertemente leer la documentación oficial y seguir los pasos indicados en la misma ya que estos

[6]Se le llama *thing* a una entidad la cual posee certificados de autenticidad y claves privadas y públicas, asociadas a una cuenta de AWS.

pueden haber cambiado luego de la fecha de realizado de este documento [57].

1. ingresar a la consola de AWS IoT con la cuenta previamente registrada (usar opción *root user*).

2. En el panel de navegación de la parte izquierda de la pantalla, ir a “Manage” y luego “Things”. Ver figura A.20.

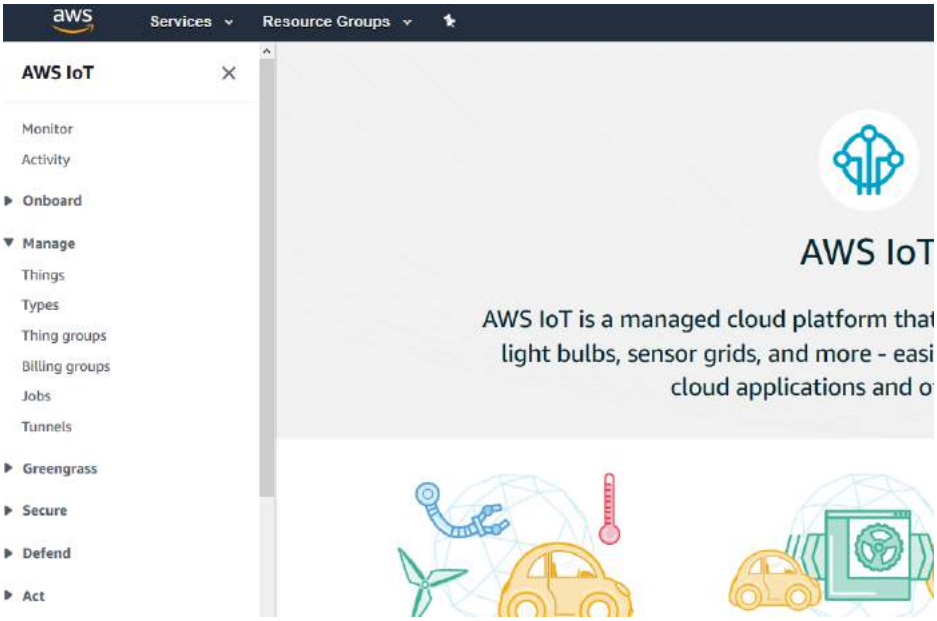

Figura A.20: Paso 2. Imagen obtenida de **Amazon Web Service**.

3. Si no se tiene creada ninguna *thing* todavía aparecerá una ventana con el mensaje “ You don’t have any things yet”, seleccionar entonces “Register a thing”. En caso de tener ya una *thing* se debe seleccionar “Create”.Ver figura A.21.

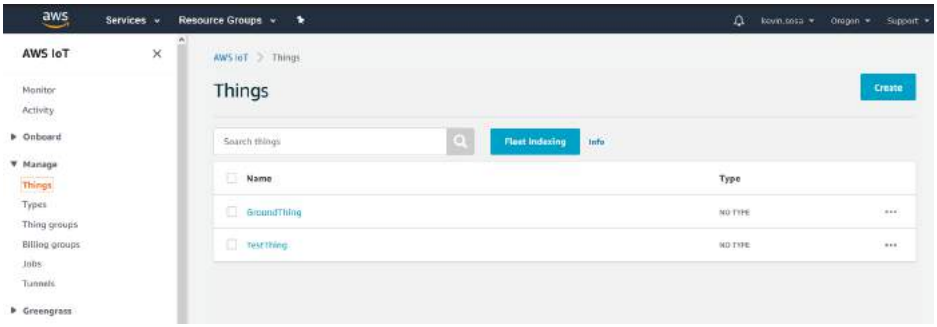

Figura A.21: Paso 3. Imagen obtenida de **Amazon Web Service**.

4. En la siguiente página seleccionar “Create a single thing”. En el formulario “Add your device to the thing registry” se le da un nombre a la *thing* a crear. Este no

puede ser cambiado posteriormente. Clickear el botón “next”. Ver figuras A.22 y A.23.

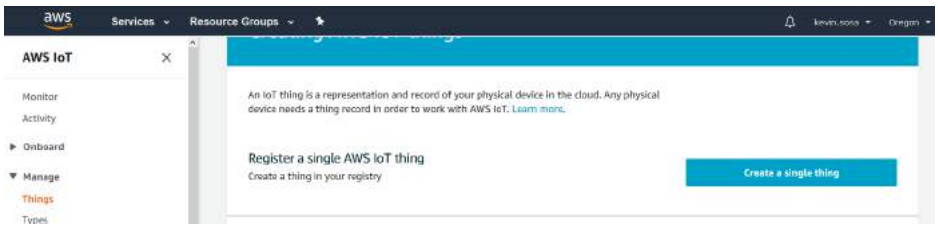

Figura A.22: Paso 4. Imagen obtenida de **Amazon Web Service**.

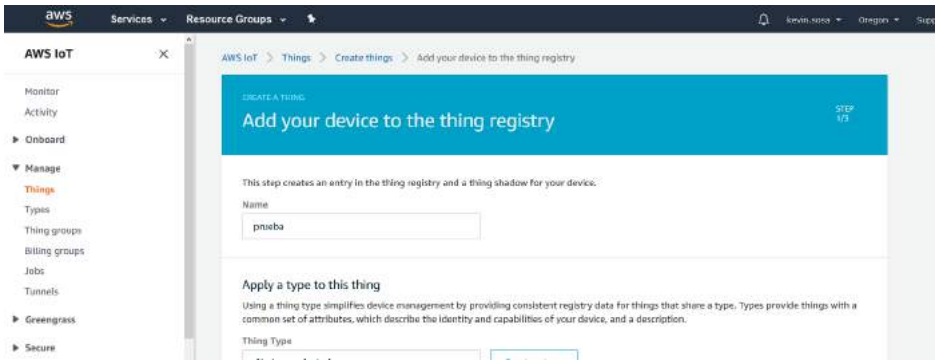

Figura A.23: Paso 4. Imagen obtenida de **Amazon Web Service**.

5. En el siguiente formulario “Add a certificate for your thing” elegir la opción “One-click certificate creation (recommended)” y clickear el botón. Una vez realizada la creación se tendrán links de descarga para los certificados, clave pública, y privada. Los nombres de los archivos descargados pueden modificarse según se crea necesario, mientras no se alteren sus contenidos ni su tipo de archivo. Ver figura A.24

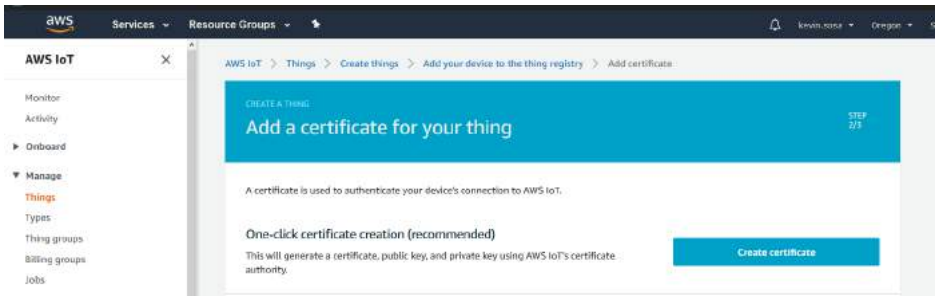

Figura A.24: Paso 5. Imagen obtenida de **Amazon Web Service**.

Esta es la **única** instancia en que se pueden obtener estos archivos, se recomienda respaldarlos inmediatamente.

6. Clickear en el botón “download” para descargar un certificado *root* de AWS. Se abrirá una nueva pestaña , clickear en el link para “Amazon Root CA 1”.

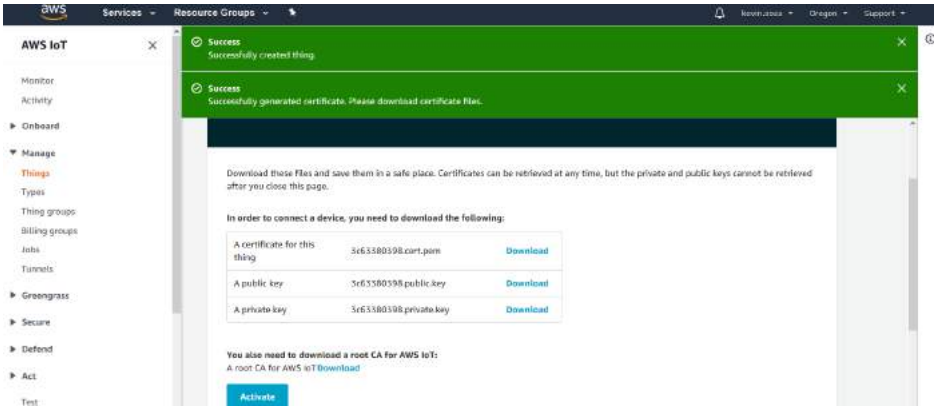

Figura A.25: Paso 6. Imagen obtenida de **Amazon Web Service**.

7. Volver a pestaña anterior y presionar el botón “activate” para finalizar la creación de la *thing*. Una vez realizado, se recibirá un mensaje de confirmación en la pestaña.

8. En la parte inferior de la pestaña, clickear el botón “attach a policy”. Se deberá entonces crear una política para la *thing*, la cual dictará los permisos de la misma para publicación y subscripción con determinadas aplicaciones. Esto es útil para restringir el uso de la misma a usuarios o aplicaciones autorizadas. Para un caso básico o de prueba se puede utilizar la política siguiente (documento en formato JSON):

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:*",
      "Resource": "*"
    }
  ]
}
```

   Esta política dará todos los permisos a la *thing* actual.

9. Finalmente, clickear el botón “Register thing”.

   Para más información, ver: `https://docs.aws.amazon.com/iot/latest/developerguide/iot-moisture-create-thing.html`.

A.5.3.8. Recarga de batería

La recarga inalámbrica del sistema fue implementada por Termodron II, y, si bien esta forma parte del sistema completo, la misma no fue utilizada por Termodron III y no formó parte del alcance del proyecto.
Su utilización si fue tenida en cuenta para el diseño del dron, de la base, el aterrizaje de precisión, y la compra de distintos componentes.

Para más información sobre el módulo de recarga inalámbrica ver [10].

A.5.3.9. Simulación del dron en PC

La simulación del dron previo a la realización de una misión es de gran ayuda para la prevención de posibles fallas o errores al realizar la misión.
Para poder realizar una simulación del dron en una computadora se debe contar con el siguiente equipamiento:

1. Laptop con conexión a internet.
2. Programa Mission Planner.
3. Instalar las librerías:
   a) Dronekit (comando: pip install dronekit)
   b) Dronekit-sitl (comando: pip install dronekit-sitl)
   c) mavproxy (comando: pip install MAVProxy)
4. Cámara térmica en caso de simulación de detección térmica.
5. Cámara estereoscópica en caso de simulación del algoritmo de evasión de obstáculos o aterrizaje de precisión.

El procedimiento para realizar la simulación es el siguiente:

1. Abrir el Mission Planner.
2. Abrir una terminal del sistema y insertar el siguiente comando:
   a) dronekit-sitl copter –home=-Latitud,Longitud,alt,yaw
3. Abrir una terminal del sistema y insertar el siguiente comando:
   a) mavproxy.py –master tcp:127.0.0.1:5760 —out udp:(Direccion IP de la computadora):14551 –out udp:(Direccion IP de la computadora):14550
4. Luego del comando una simulación del dron se conecta automáticamente al Mission Planner.
5. Abrir una nueva terminal del sistema y ejecutar el algoritmo que se desee utilizar.

A.5.4. Preparación del dron previo al vuelo

Para el primer vuelo del dron, o antes de realizar un vuelo con el dron luego de mucho tiempo de inactividad, se recomienda calibrarlo, como se indica en A.5.5, además, es pertinente realizar otros chequeos previos, en particular:

- Batería correctamente conectada y con carga suficiente (este último punto es manejado por el software)
- Hélices sin daños, ajustadas de forma que no haya movimiento de las mismas independiente a los motores, y que las mismas estén colocadas de forma coherente con el sentido de giro de los motores. En la figura A.26 se muestra el sentido de giro de los motores para un cuadricóptero.
- Tapas de los motores: se tienen dos tipos de tapas de motores, grises y negras, cada una con una rosca del tipo opuesta (horaria y anti-horaria). Como los motores del dron giran en dos sentidos opuestos (un par horario y un par anti-horario como en la figura A.26), las tapas de los motores deben colocarse de forma que al girar las hélices las roscas se mantengan apretadas. Es decir, los motores que giran en sentido horario (motores 3 y 4 en A.26) llevan las tapas con rosca anti-horaria, es decir las de color negro. Análogamente los motores de giro anti-horario (motores 1 y 2 en A.26) llevan las tapas de color gris.
- Módulo de GPS correctamente conectado al puerto CAN1 del Pixhawk 2.1, de forma que al conectar la batería del dron el mismo enciende luces azules.

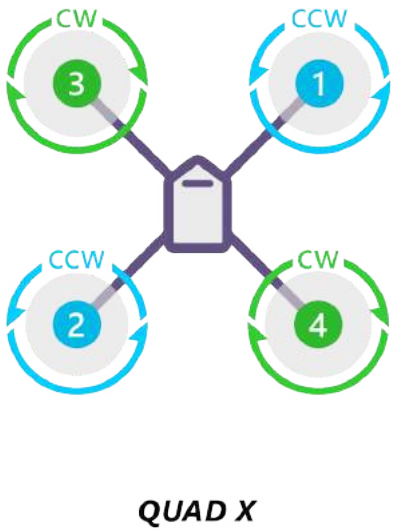

Figura A.26: Sentido de giro de los motores de un cuadricóptero tipo X. También se indican los números de motores correspondientes a los pines de conexión de los ESC en el Pixhawk 2.1.

El módulo de GPS Here2 posee un código de colores en sus luces LED, indicando de esta el estado del dron una vez encendido. Esto es útil para verificar los reportes del estado del dron por software previo al vuelo, así como identificar la posible causa de algún error. En [58] se observan los códigos de colores y sus significados, pero se mencionan a continuación los más comunes en la práctica:

- Parpadeando azul: dron desarmado, sin señal GPS adquirida. Generalmente se observa brevemente luego de encender el dron.
- Parpadeando rojo y azul: inicializando sensores
- Parpadeando amarillo: chequeos pre vuelo fallidos. Se recomienda ver en software la razón de los mismos.
- Parpadeando verde: listo para armar, con señal de GPS adquirida.
- Verde sólido: armado, listo para volar.

A.5.5. Calibración del dron

Previo a un vuelo, es vital confirmar que el estado del dron sea el adecuado para volar, es decir, revisar que la calibración de los sensores del mismo sean adecuadas para volar en las circunstancias del momento. Para hacer este procedimiento se debe elegir un software de control de misión; existen tres opciones: Mission Planner, QGroundControl y APM Planner 2 para Windows, o APM Planner 2 para Linux. Las imágenes que se muestran en esta sección son utilizando Mission Planner, pero existen tutoriales de calibración para QGroundControl también, y APM Planner 2 es muy similar a Mission Planner.

Los sensores a calibrar previo a un vuelo son los acelerómetros y las brújulas, a continuación se detalla el procedimiento para la correcta calibración de los mismos mediante el software Mission Planner. Dichas calibraciones deben ser realizadas en lugares con poca interferencia magnética, y lejos de superificies metálicas, en lo posible, al aire libre.

Se recomienda calibrar estos sensores cada semana o dos semanas, o en caso de cambios climáticos importantes, como ser grandes fluctuaciones en la humedad ambiente.

Información adicional y videos de ejemplo para la realización de las calibraciones se pueden encontrar en [59].

Calibración de acelerómetro

Al comenzar la calibración se debe acceder al Mission Planner, ir a la pestaña *SETUP* y luego a *Mandatory Hardware*. Una vez en *Mandatory Hardware* se debe seleccionar *Accel Calibration* y darle comienzo al proceso de calibración.

En la figura A.27 se observa la ubicación en el Mission Planner de la calibración del acelerómetro.

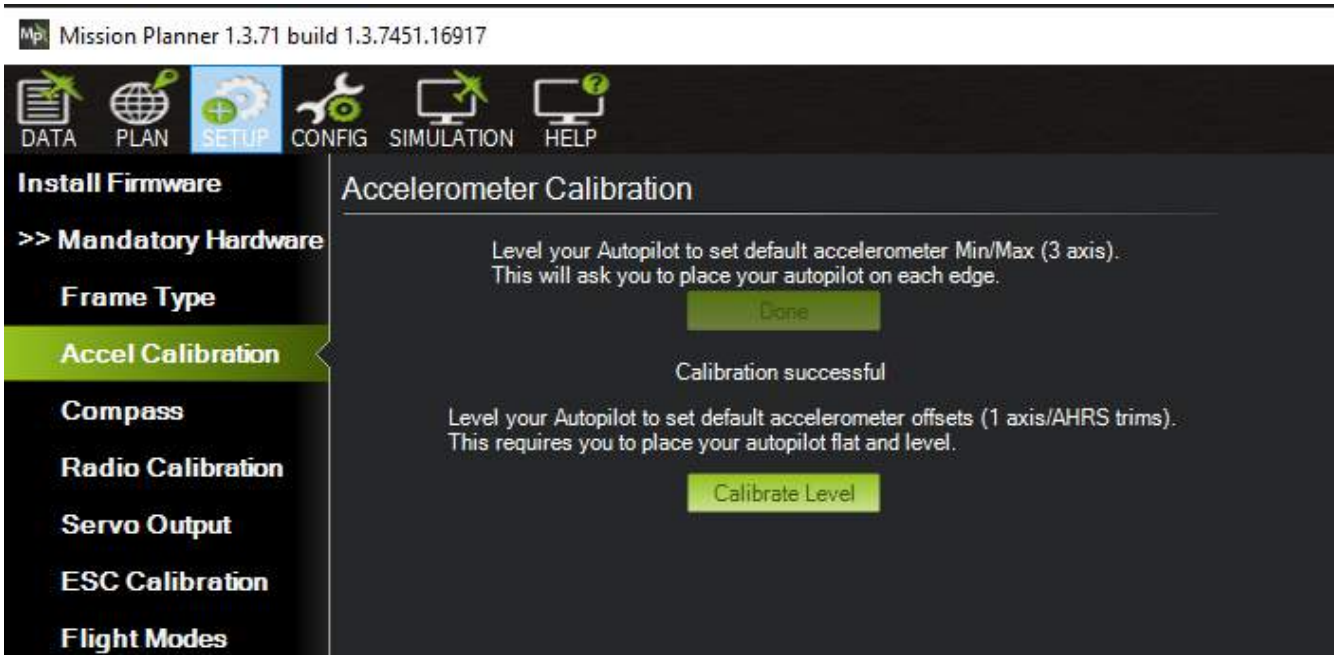

Figura A.27: Ubicación en el programa Mission Planner de la calibración del acelerómetro. Imagen obtenida de **ArduPilot**.

El programa irá solicitando posicionar el dron sobre cada uno de sus lados, debiendo presionar el botón de *DONE* luego de colocar el dron en cada una; una vez finalizadas todas las posiciones, se confirma la calibración.

En la figura A.28 se observan las distintas posiciones en las cuales se debe colocar el dron con el fin de realizar la calibración.

Figura A.28: Distintas posiciones en las que se debe colocar el dron para la calibración del acelerómetro. Imagen obtenida de **ArduPilot**.

Calibración de brújula

Para evitar errores en la calibración de la brújula, la calibración no se debe realizar cerca de superficies metálicas o de equipos que produzcan campos magnéticos, como son computadoras, celulares, fuentes de corriente, etc.

Utilizando Mission Planner, se debe ir a la pestaña *SETUP*, luego en la barra de opciones se elige *Mandatory Hardware* y luego *Compass*. Estos pasos se observan en la figura A.29.

Luego se deben seleccionar cuáles brújulas[7] se desean calibrar en *Onboard Mag Calibration* y apretar el botón de *START*. Una vez comenzada la calibración se debe rotar el dron en todas las direcciones, con un movimiento lento y controlado, observando el aumento en las barras de calibración del Mission Planner, hasta que este de aviso de calibración completa.

Una vez completa la calibración se debe reiniciar el dron.

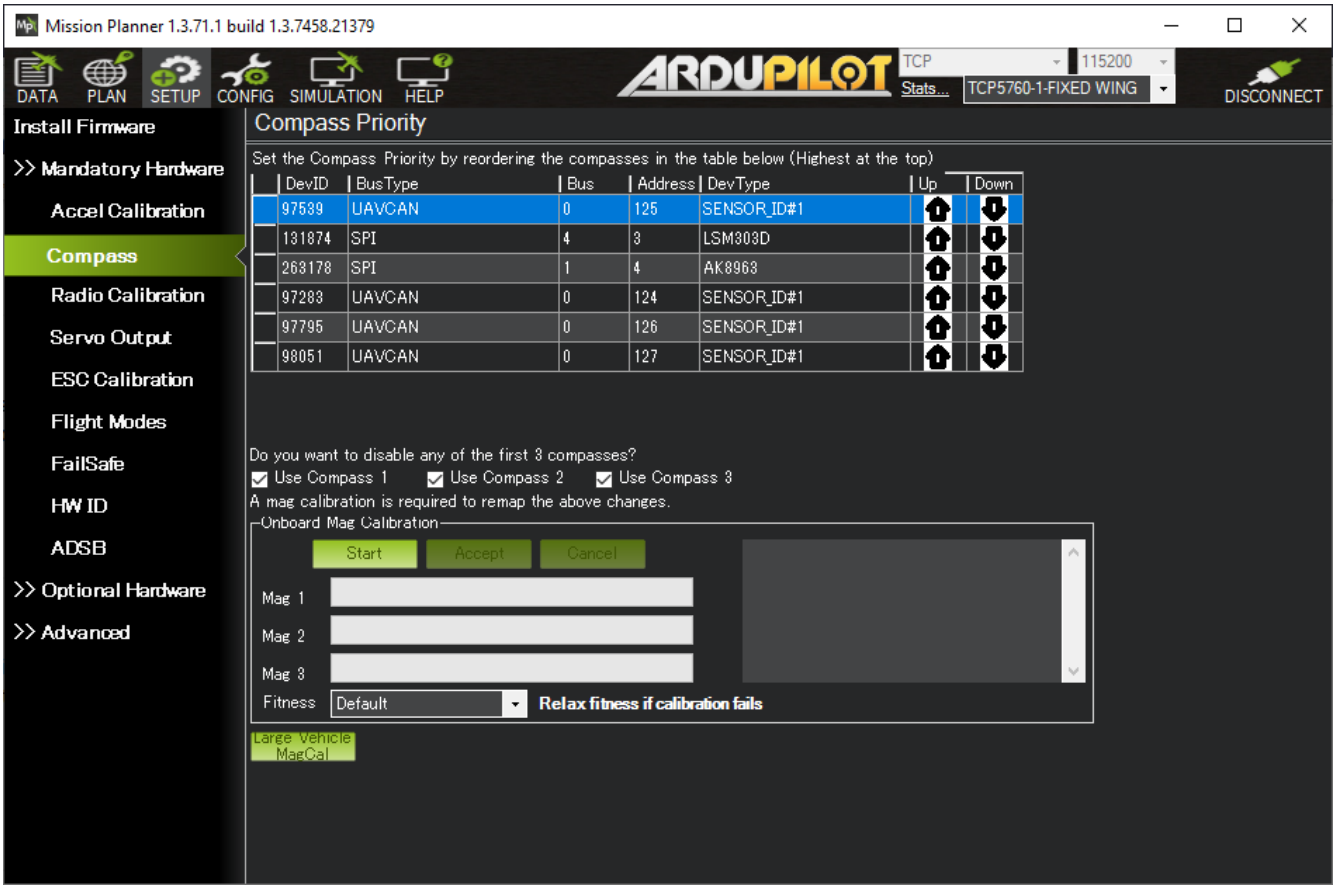

Figura A.29: Ubicación en el programa Mission Planner de la calibración de la brújula. Imagen obtenida de **ArduPilot**.

También puede ser necesario calibrar los *ESC* en caso de cambio de motores, o cambio

[7]Se tienen dos brújulas, una por cada IMU del Pixhawk Cube, y una exterior, dentro del módulo GNSS Here2. Se recomienda sólo utilizar la última, ya que con las dos del Pixhawk no se lograba buena navegación.

en las conexiones al controlador de vuelo.

Calibración de ESC

Para realizar la calibración de los *ESC* se debe utilizar el control remoto, retirar las hélices del dron y no se debe conectar el controlador de vuelo a la *PC*. El procedimiento a realizar es el siguiente:

1. Encender el control remoto con el *throttle* al máximo (ver imagen A.30).
2. Conectar la batería al dron, encendiéndolo. Se observara el cambio en las luces del GPS pasando de azul a un parpadeo rojo, azul y verde. Esto indicará que en el próximo inicio se iniciará en el modo calibración de *ESC*.
3. Desconectar y volver a conectar la batería, manteniendo el *throttle* al máximo.
4. Al conectar nuevamente, y verificar que las luces del GPS están parpadeando rojo, azul y verde, se debe esperar que el dron realice dos pitidos consecutivos.
5. Luego de los dos pitidos consecutivos bajar el *throttle* al mínimo, (ver imagen A.31).
6. Esperar por tres pitidos consecutivos.

Luego de los tres pitidos los *ESC* se encuentran calibrados, y el dron se encuentra en modo de control **manual**, por lo que se debe tener extremo cuidado y no hacer esta calibración con las hélices puestas. Se puede aumentar de a poco el *throttle* con el fin de corroborar el correcto sentido de giro de los motores.

Figura A.30: Paso 1 del procedimiento de calibración de *ESC*, palanca de *throttle* al máximo. Imagen obtenida de **ArduPilot**.

Figura A.31: Paso 5 de calibración de *ESC* palanca de *throttle* al mínimo. Imagen obtenida de **ArduPilot**.

Además de estas tres calibraciones fundamentales, se pueden realizar otras, las cuales son de uso opcional, pero son recomendadas, y se detallan a continuación.

Calibración de la radio

La calibración de la radio tiene como objetivo capturar el valor mínimo y máximo de cada entrada del control remoto con el fin de poder interpretar de forma correcta su valor.

Se deben tener algunas consideraciones, las cuales son:

1. La batería debe estar desconectada y se deben retirar las hélices, con el fin de prevenir posibles movimientos de los motores del dron.
2. Se debe asegurar que el receptor de la radio esté conectado al Pixhawk.
3. Se debe conectar el Pixhawk a la computadora mediante el uso de un cable USB.

El procedimiento para realizar la calibración es el siguiente:

1. Luego de conectado al dron en el Mission Planner, se debe ir a *Initial Setup*, luego *Mandatory Hardware* y por ultimo seleccionar la opción de *Radio Calibration* (ver figura A.32.).

2. Se debe apretar el botón *Calibrate Radio*, y aparecerá una ventana en donde se indica que se debe mover todos las perillas y palancas del control hacia ambos extremos. Al clickear *OK* comienza la calibración, viéndose como se mueven las columnas grises de cada canal del contro, como se observa en la figura A.32.

3. Una vez verificados todos los canales activos del control, se debe apretar el botón *Click when done*, para finalizar la calibración.

Figura A.32: Ubicación en el programa Mission Planner de la calibración de la radio, en donde se observa la ventana previa al comienzo de la calibración. Imagen obtenida de **ArduPilot**.

Figura A.33: Barras de calibración completas para la radio, donde se observa el máximo y el mínimo en cada una. Imagen obtenida de **ArduPilot**.

Configuración de los modos de vuelo en el control remoto

Previo a la realización de una misión, es de gran utilidad configurar distintos modos de vuelo en el control remoto, por ejemplo, en caso de una emergencia en vuelo y ante la necesidad de realizar un *LAND* de emergencia.

Para ello se tiene la posibilidad de configurarlos desde el Mission Planner, en la pestaña *Initial Setup*, luego *Mandatory Hardware* y por ultimo seleccionando *Flight Modes*.
Se tiene la posibilidad de configurar seis modos de vuelos, para lo cual se debe:

1. Encender radiocontrol.
2. Conectarse al dron mediante Mission Planner.
3. Mover las perillas del control, identificando qué canal de PWM corresponde a cada perilla.
4. En cada uno de los seis posibles modos de vuelo a configurar se puede desplegar una lista seleccionado el que se quiere utilizar.
5. Una vez elegidos los modos configurados, clickear en *Save Modes*.

En la figura A.34 se presenta una imagen del Mission Planner en donde se observa la elección de los distintos modos de vuelo.

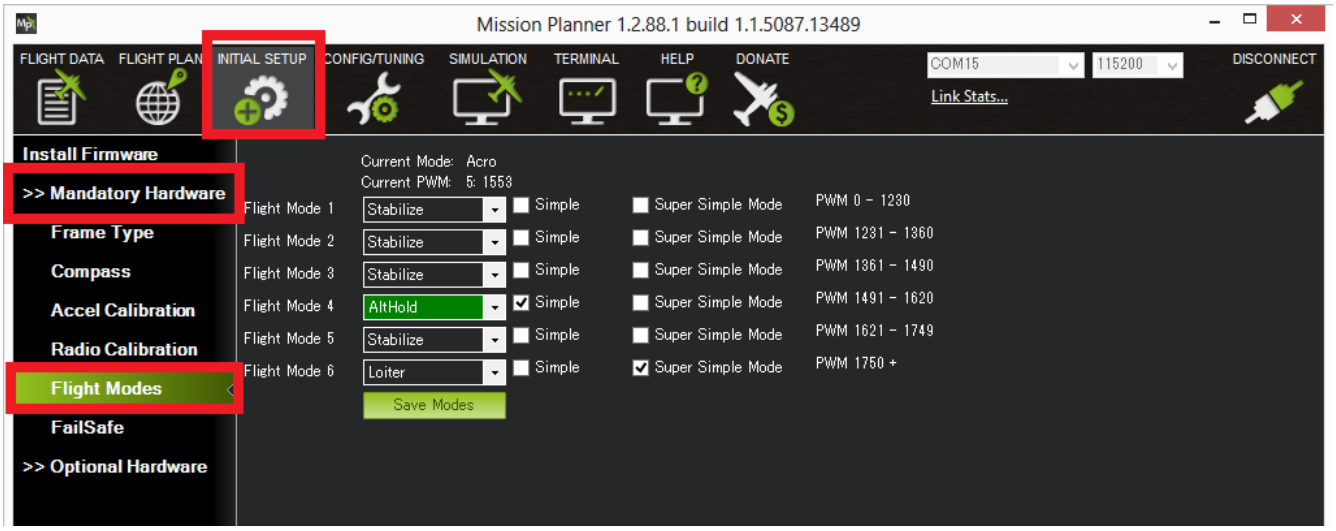

Figura A.34: Selección de los seis posibles modos de vuelo en el control remoto. Imagen obtenida de **ArduPilot**.

Configuración del rango de los motores

Debido a que la mayoría de los *ESC* presentan una zona muerta en su mínimo valor (para los cuales el motor no gira), se puede calibrar dicho rango para que el controlador lo tenga en cuenta a la hora de un vuelo.

Para realizar dicha configuración se debe tener en cuenta los siguientes puntos:

1. Se debe haber realizado la calibración de los *ESC* previo a la configuración
2. Se deben retirar las hélices previo a la configuración

Para comenzar se debe conectar la batería al dron y se debe conectar el dron al Mission Planner mediante el uso de un cable USB o la antena de telemetría.
Una vez conectado se debe dirigir hacia *Initial Setup*, luego *Optional Hardware* y luego *Motor Test*, como se puede ver en la figura A.35.

Luego se comienza a aumentar el porcentaje de *throttle* hasta que los motores comienzan a girar, probando cada motor por separado con los botones *Test Motor A..D*. Teniendo los cuatro motores iguales, el porcentaje de *throttle* podría variar entre un 1 % y un 2 %, se debe seleccionar el mayor porcentaje de los cuatro motores.

Figura A.35: Test de los motores, con el fin de configurar el rango de los ESC. Imagen obtenida de **ArduPilot**.

Una vez obtenido el porcentaje de los motores se procede a configurar los parámetros *MOT_SPIN_ARM* (parámetro que configura la velocidad de los motores cuando se encuentran armados) y *MOT_SPIN_MIN* (parámetro que configura el valor mínimo de la velocidad de los motores en vuelo).
Es importante recordar que cuando el dron se encuentra armado los motores giran a una velocidad inferior a la que giran cuando el dron se encuentra en vuelo.

El parámetro *MOT_SPIN_ARM* se debe configurar como el porcentaje del throttle previamente calculado mas 2 % dividido 100 (por ejemplo, si el valor del throttle fuese 7 %, el valor del parámetro seria 9/100, osea 0,09).
El parámetro *MOT_SPIN_MIN* se calcula con el valor del parametro *MOT_SPIN_ARM*, este debe ser por lo menos 0.03 superior al *MOT_SPIN_ARM* (por ejemplo, si *MOT_SPIN_ARM* fuese 0.09, entonces *MOT_SPIN_MIN* seria 0.12).

A.5.6. Calibración de cámara para detección de marcadores ArUco

Debe ser mencionado que esta calibración debe realizarse solamente si se cambia el tipo de cámara, o la resolución de la misma, de lo contrario ya se proveen los archivos de calibración necesarios para utilizar la cámara Intel RealSense D415 para el aterrizaje de precisión.

Como ya se mencionó en la sección 5.2.1.1, para el aterrizaje de precisión es necesario detectar el marcador ArUco y poder estimar su posición relativa a la cámara. Para poder realizar lo último se requiere obtener los parámetros de calibración de la cámara: la matriz de la cámara y los coeficientes de distorsión producidos por el lente de la misma [22].

A continuación se detalla el procedimiento a seguir para calibrar una cámara para detectar marcadores ArUco, el cual fue obtenido de [21].

Los archivos de calibración son: *save_snapshots.py*, *cameraCalib.py* y *chessBoard_9x6.jpg* y se encuentran en la carpeta "calibracion" dentro de "Codigo/libDron/precLand" del repositorio principal. Dentro de la carpeta "cameraCalib" se encuentran los archivos de la calibración realizada para la cámara Intel RealSense D415.

Primero, se debe imprimir la imagen *chessBoard_9x6.jpg* en una hoja A4. Se debe medir el ancho de la imagen y el ancho de una celda para luego ingresar estos valores en un *script*. Luego, se debe montar la impresión sobre una superficie plana y rígida.

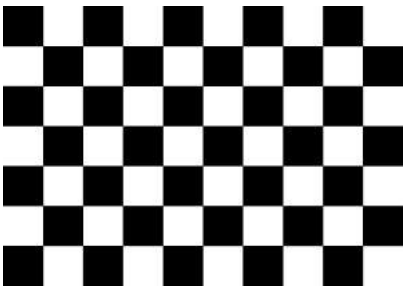

Figura A.36: Imagen chessboard.jpg.

Una vez realizado esto, se deben tomar fotos del tablero de ajedrez en varias orientaciones y sentidos, para lo cual se debe correr el código *save_snapshots.py* en Python 3, especificando los siguientes argumentos mediante línea de comando: –f FOLDER –n FILENAME –w WIDTH –h HEIGHT. Donde FOLDER es el nombre del directorio en el cual guardar las imágenes a tomar (el directorio ya debe existir, en caso de no especificar ninguno se guarda en la misma carpeta), FILENAME es el nombre con el cual se guardan las imágenes, WIDTH es el ancho en píxeles de la resolución de la cámara a utilizar, HEIGHT es la altura en píxeles. Al correr el código se desplegará en pantalla la imagen de la cámara, y se deberá la tecla "s" para guardar una imagen y la tecla "q" para cerrar el programa una vez guardadas suficientes imágenes. Se recomienda tomar entre treinta y cincuenta imágenes del tablero de ajedrez rotado en diferentes ángulos y a diferentes

distancias de la cámara, ya que luego se deberán descartar algunas imágenes previa a la obtención de los parámetros de calibración.

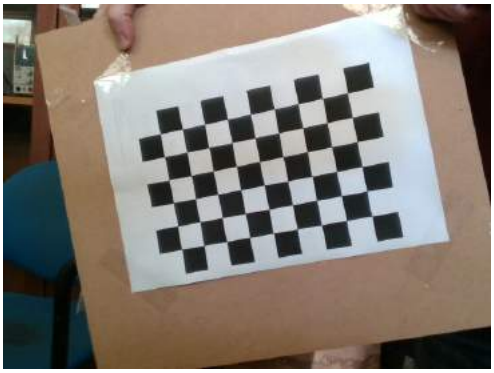

Figura A.37: Distintas orientaciones y sentidos del tablero de ajedrez.

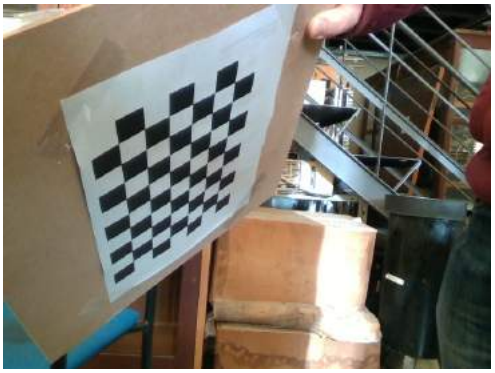

Figura A.38: Distintas orientaciones y sentidos del tablero de ajedrez.

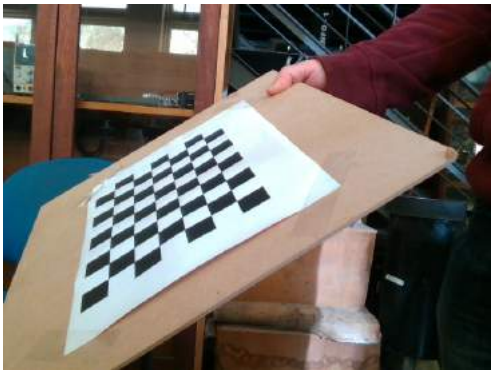

Figura A.39: Distintas orientaciones y sentidos del tablero de ajedrez.

Figura A.40: Distintas orientaciones y sentidos del tablero de ajedrez.

Luego de obtenidas las imágenes se debe correr el código *cameraCalib.py* en Python 3, de la siguiente forma: *cameracalib.py FOLDER IMGTYPE NUMROWS NUMCOLS CELLDIMENSIONS*. Donde FOLDER es la carpeta donde se guardaron las imágenes tomadas en el paso anterior, IMGTYPE es el tipo de imagen a abrir (.jpg por defecto), NUMROWS es el número de filas del tablero de ajedrez usado (9 en la imagen provista), NUMCOLS es el número de columnas del tablero (6 en la imagen provista), CELLDIMENSIONS es el tamaño de una celda del tablero, en mm (se recomienda medir siempre pues puede haber discrepancias en las impresiones).

Luego, se mostrarán en pantalla las imágenes tomadas anteriormente, con un overlay de colores que indica dónde se detectaron esquinas de las celdas. Se debe observar si se detectan todas las esquinas de las celdas correctamente, en caso afirmativo, se presiona "Enter" para guardar esta imagen internamente para el procesamiento, de lo contrario, se presiona "Escape" para descartarla y pasar a la siguiente. Una vez que se hayan analizado

todas las imágenes tomadas, se realizarán los cálculos de los parámetros de calibración, lo cual puede llevar desde tres a quince minutos dependiendo si se corre en una laptop o una SBC.

Figura A.41: Imagen con overlay de colores en la cual se detectan correctamente todas las esquinas de las celdas, y es apta para utilizar.

Figura A.42: Imagen con overlay de colores en la cual no se detectan correctamente las esquinas de las celdas, y no es apta para utilizar.

Una vez finalizados los cálculos, se imprimirán en pantalla los parámetros y se habrán guardado los archivos correspondientes en la carpeta donde se guardaron las imágenes en el paso anterior. Se deben guardar estos dos archivos *cameraMatrix.txt* y *cameraDistortion.txt* en el directorio "cameraCalib" sobreescribiendo los originales (se recomienda mantener un respaldo de los originales).

A.5.7. Calibración del Gimbal

Para mantener la posición de la cámara estéreo estable se utiliza un gimbal STorM32 de 3 ejes. Se elimina el motor que controla el ángulo de yaw, de forma de tener el gimbal siempre apuntando en la dirección de vuelo del dron, lo cual es clave para utilizar la cámara estéreo como método de detección de obstáculos. Para poder realizar esta función el gimbal debe ser balanceado para mantener la posición cuando los motores no están energizados, y luego debe calibrarse el controlador PID del mismo, para poder estabilizarse durante el vuelo. Para realizar esta calibración se utiliza el software provisto por los fabricantes del controlador del gimbal [60].
En la figura A.43 se observa la pantalla principal del software con los parámetros de los distintos motores.

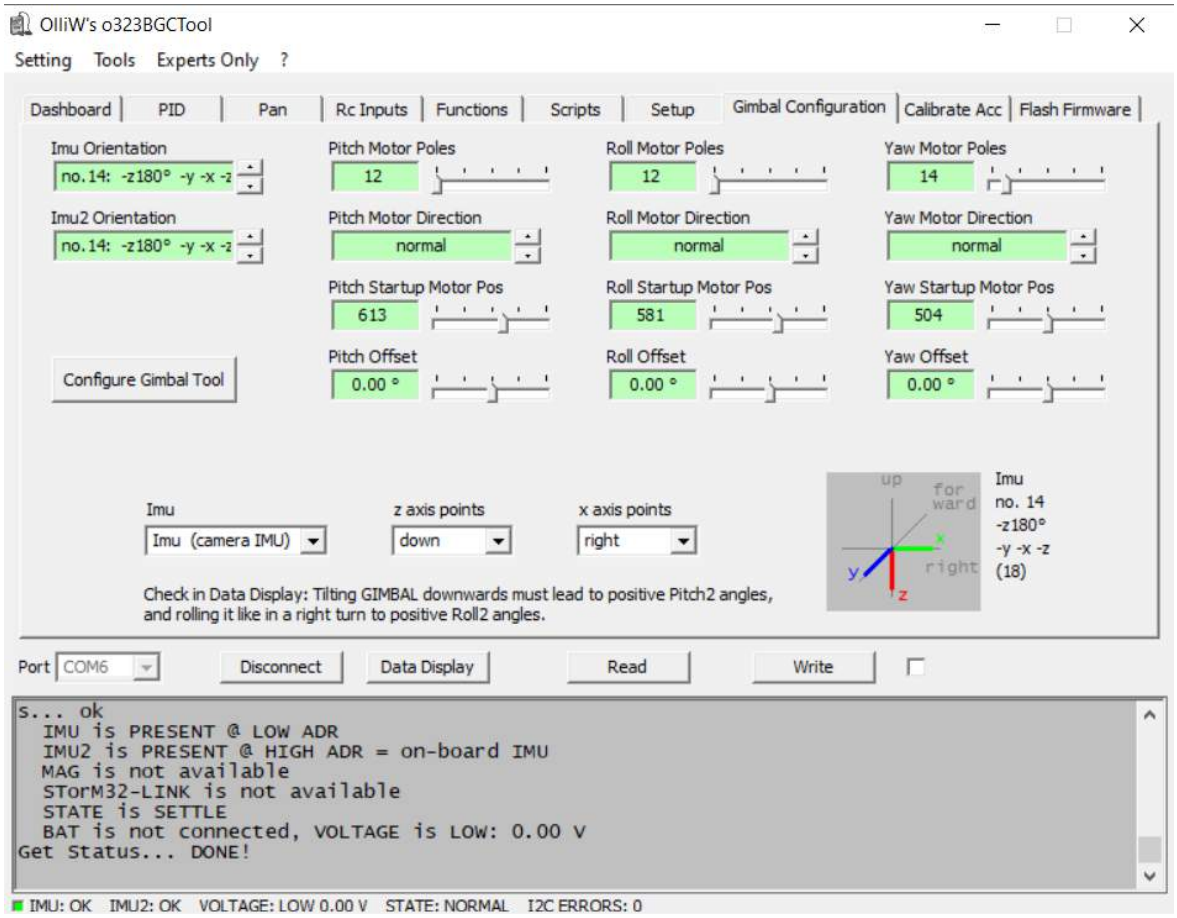

Figura A.43: Pantalla principal de la calibración del gimbal.

La segunda función del gimbal es permitir apuntar la cámara estéreo hacia el suelo, ajustando el ángulo de pitch a -90º, de forma de poder detectar los marcadores ArUco para el aterrizaje de precisión. Para poder lograr esto es necesario poder comunicarse con el gimbal durante el vuelo, esto se logra mediante la biblioteca DroneKit-Python, la cual permite, a través del controlador de vuelo, enviar mensajes hacia el gimbal para controlar los ángulos del mismo. De esta forma, cuando se termina la misión, el dron llega a las coordenadas de la base y procede a comenzar el proceso de aterrizaje preciso, se cambia el ángulo de pitch del gimbal para apuntar la cámara hacia abajo, ya que no es necesaria la función de detección de obstáculos.

En las siguientes figuras se muestra el procedimiento, paso a paso, para la calibración del gimbal.

Como se mencionó anteriormente, el software provee una calibración automática del PID. Para poder acceder a esta, se debe ingresar en la pestaña “Gimbal Configuration”, y luego presionar en “Configure Gimbal Tool” como se muestra en la figura A.43.

Seguido a esto aparece una ventana como la de la figura A.44 donde aparecen algunas

aclaraciones, y a la derecha aparecen tres bloques, donde se puede seleccionar las opciones de calibración. Se puede quitar la marca de "Yaw Motor Position", como se mencionó este motor fue retirado, aunque no es de vital importancia quitarlo, en este ejemplo se mantuvo la opción marcada. Seguido a esto se presiona en "Continue"'.

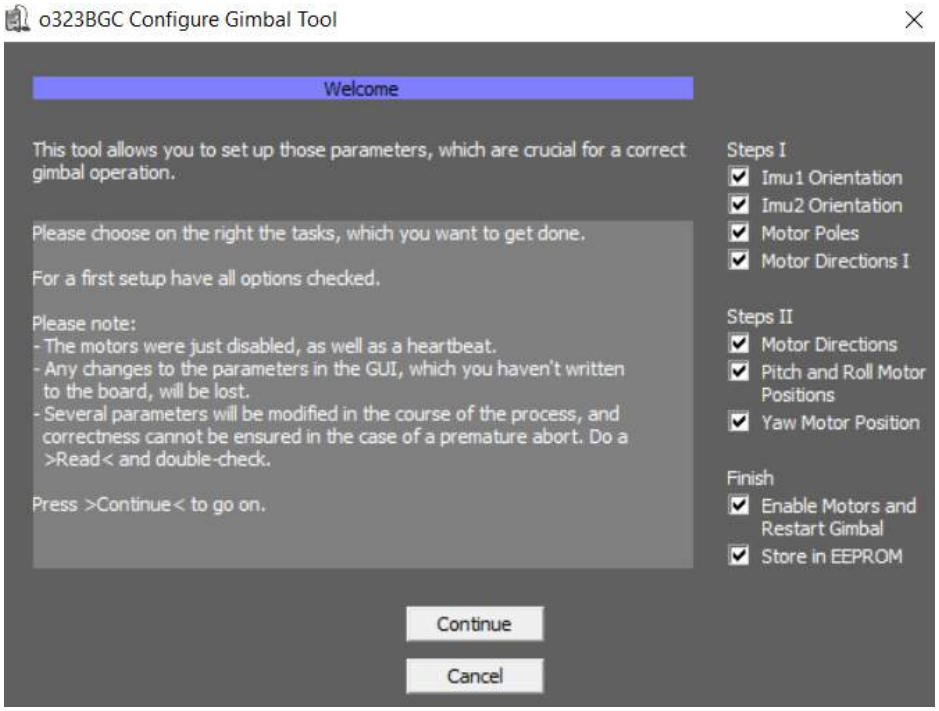

Figura A.44: Pantalla principal para la calibración del gimbal.

Luego aparece la ventana como la de la figura A.45. En este punto se debe colocar manualmente la cámara mirando hacia adelante. Lo más alineado posible con un error de 15^o. Manteniendo esta orientación se presiona sobre el botón "Continue".

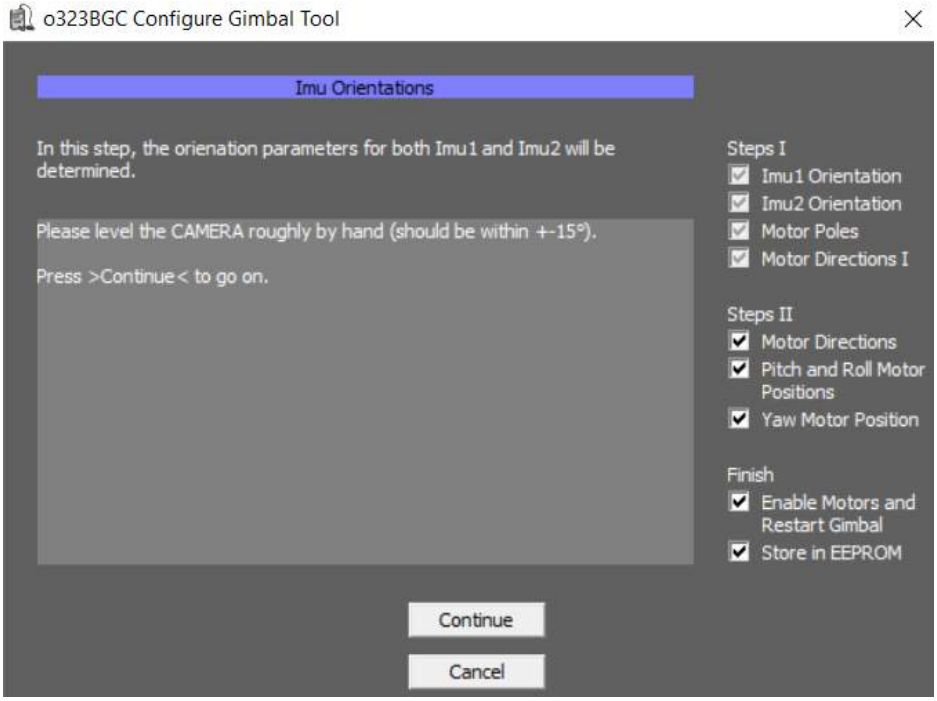

Figura A.45: Configuración de la orientación de las IMU.

Seguido de esto, aparece la ventana de la figura A.46 donde se deben colorar el dron y el gimbal de forma manual a 45^o respecto al suelo y esperar a que cambie la ventana a la de la figura A.47, una vez que cambie se puede volver a colocar el dron en el suelo y presionar en “Continue”

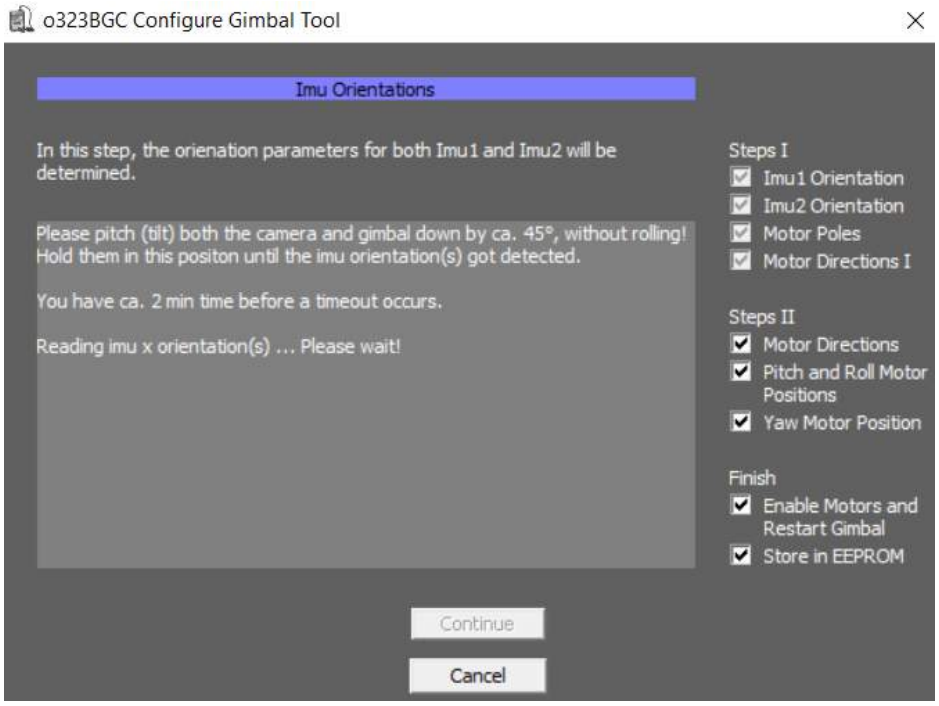

Figura A.46: Configuración de la orientación de las IMU.

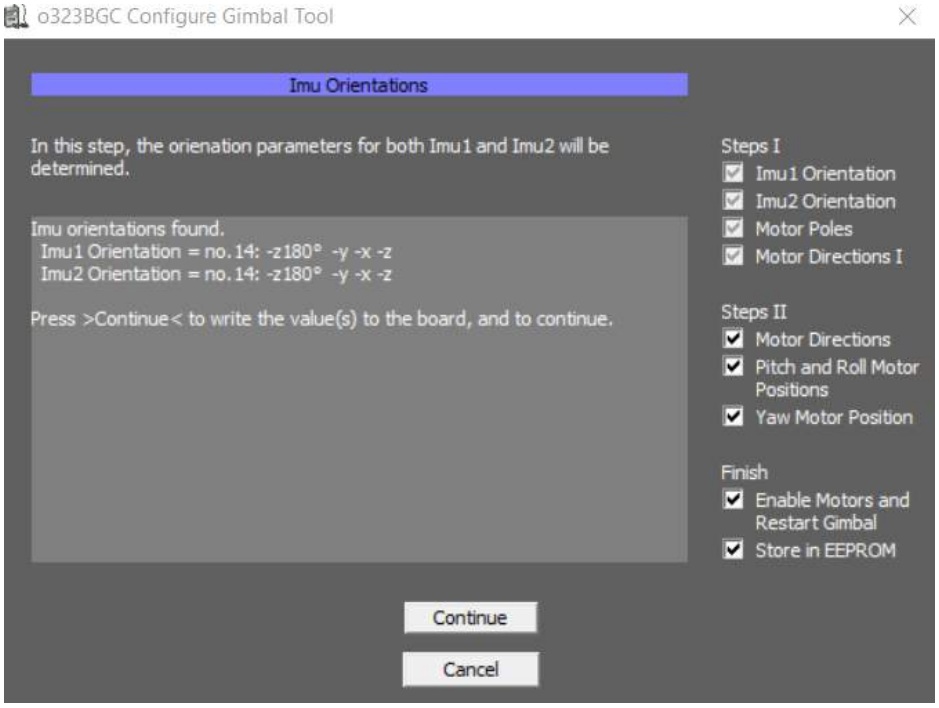

Figura A.47: Configuración de la orientación de las IMU.

Luego, como se muestra en la figura A.48, se elige la cantidad de polos de los motores,

por defecto están en 14, en nuestro caso, pusimos tanto el de pitch como el roll en 12, el yaw se lo puede dejar como está, ya que este motor no se utiliza, luego de seleccionar la cantidad de polos, seleccionar “Continue”.

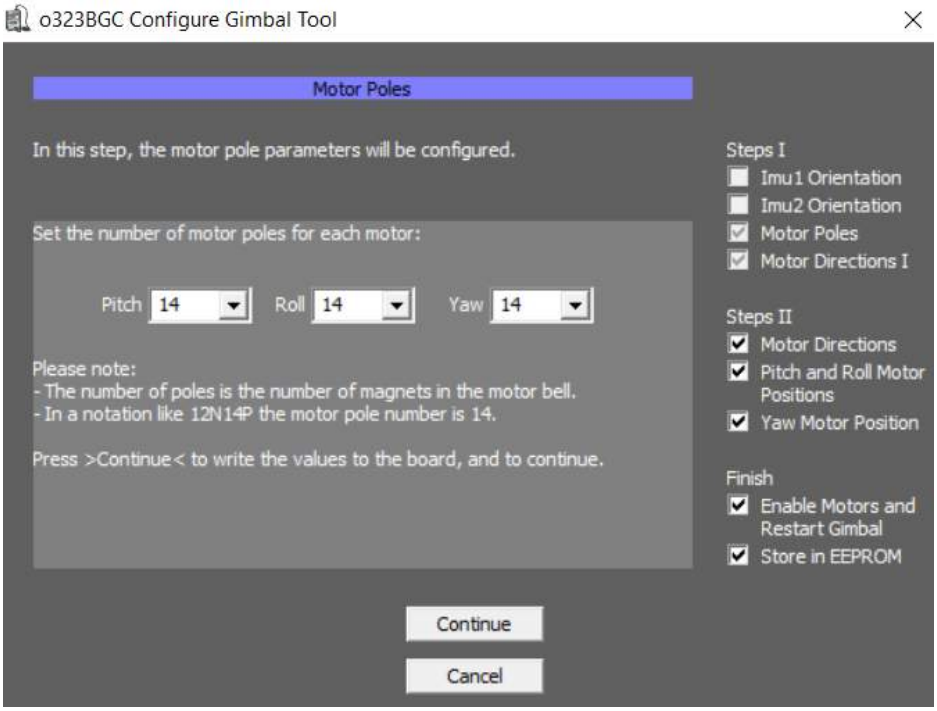

Figura A.48: Configuración de los polos de los motores.

En la siguiente ventana (figura A.49) unicamente se presiona “Continue”

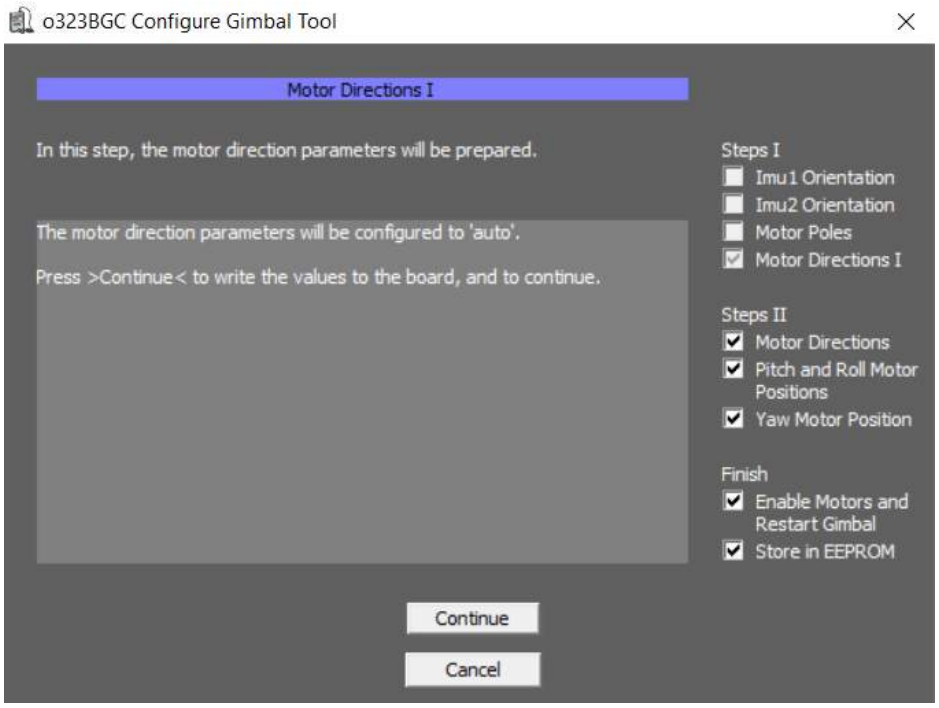

Figura A.49: Configuración de la dirección de los motores.

En este punto se debe corroborar que la batería esté conectada. Como se indica en la figura A.50 el gimbal se reiniciar, para posteriormente poder orientar los motores. Seguido de esto se debe presionar en “Continue”.

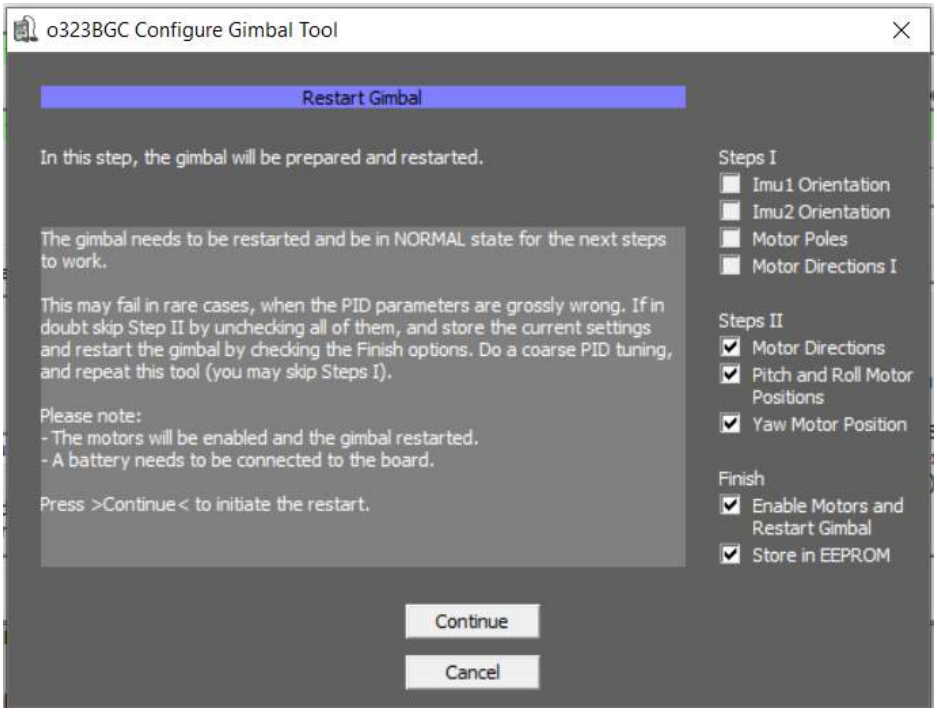

Figura A.50: Reinicio del gimbal.

Luego el controlador se inicializa y nivela la cámara (led verde parpadea), finalizado esto se produce un pitido y el led verde queda encendido. Y se observa la figura A.51

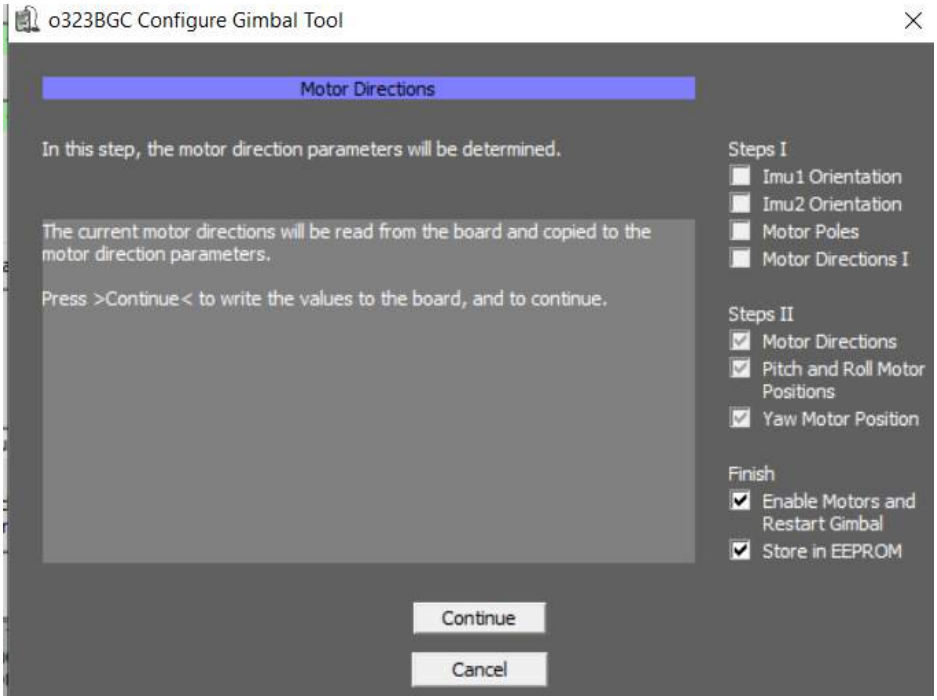

Figura A.51: Configuración de la dirección de los motores.

Para finalizar en las ventanas de las figuras A.52 y A.53 simplemente presionamos “Continue”, como se mencionó anteriormente, la alineación del yaw se podía haber salteado anteriormente, ya que este motor no se utiliza.

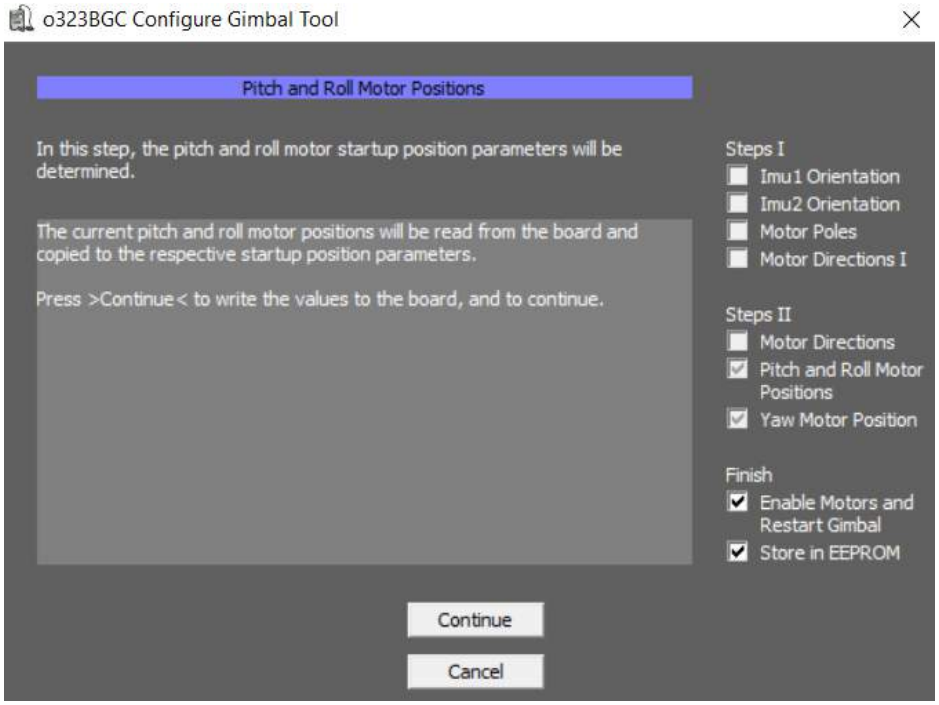

Figura A.52: Configuración de los motores del pitch y roll.

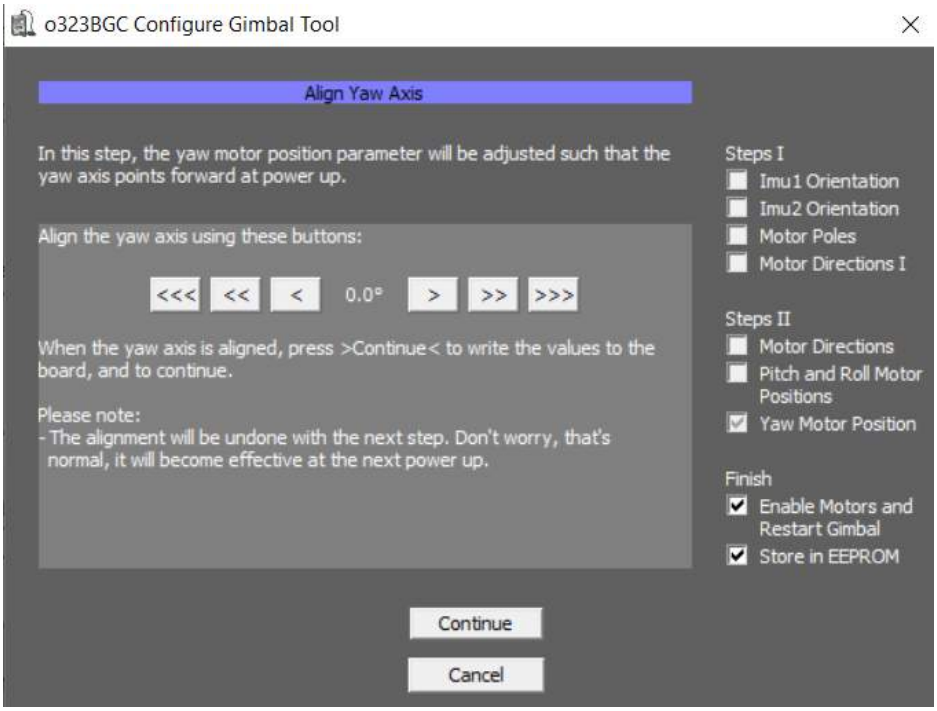

Figura A.53: Configuración de la alineación del eje de yaw.

Por último el controlador guarda en la memoria los parámetros y se presiona "OK" como indica la figura A.54 para terminar el proceso de calibración

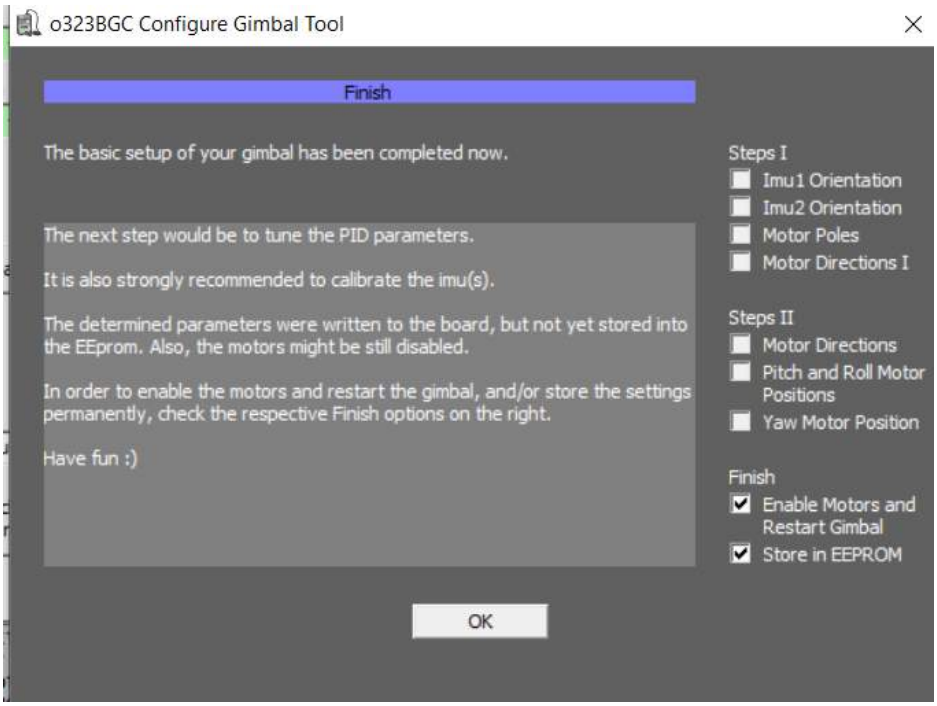

Figura A.54: Proceso de configuración del gimbal finalizado.

Índice de cuadros

Índice de figuras

Referencias

[1] Fundeu BBVA, "Dron".[Online]. Disponible: https://www.fundeu.es/recomendacion/dron-adpatacion-al-espanol-de-drone/

[2] Cristina Cuerno-Rejado, Luis García Hernández, Alejandro Sánchez-Carmona, Adrián Carrio, Jose Luis Sanchez-Lopez, Pascual Campoy, Centro de Automática y Robótica, UPM-CSIC (España) y Universidad Politécnica de Madrid (España), "Evolución histórica de los vehículos aéreos no tripulados hasta la actualidad" [Online]. Disponible: http://oa.upm.es/40803/1/INVE_MEM_2015_203893.pdf

[3] Santiago Bernheim, Agustin Costa, Andres De Luca, Universidad de la República, Facultad de Ingeniería, "Robot Rover : Robot Autónomo Móvil Terrestre", 2019

[4] Diego Astessiano, Pablo Romero, Berardi Sensale, Universidad de la República, Facultad de Ingeniería, "Proyecto Pez Robot",2008

[5] Florencia Blasina, Facundo Artagaveytia, Nicolás Márquez, Universidad de la República, Facultad de Ingeniería, "Proyecto Pez Robot II", 2016

[6] Santiago Paternain, Rodrigo Rosa y Matías Tailanián, Universidad de la República, Facultad de Ingeniería, "uQuad! – Implementación de un UAV con arquitectura de cuadricóptero", 2012

[7] Sebastian Torterolo, Juan Manuel Torterolo,Facundo Cayafa, Universidad de la República, Facultad de Ingeniería, "Diseño y automatización de un UAV", 2016

[8] Wikipedia, "Ángulos de Euler".[Online]. Disponible: https://es.wikipedia.org/wiki/\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{A\global\mathchardef\accent@spacefactor\spacefactor}\accent19A\egroup\spacefactor\accent@spacefactor\futurelet\@let@token\protect\penalty\@M\hskip\z@skipngulos_de_Euler

[9] Intel, "Intel Realsense Depth Camera D415" [Online]. Disponible: https://store.intelrealsense.com/buy-intel-realsense-depth-camera-d415.html

[10] R. De Soto, M. Mendívil, F. Díaz, "TERMODRON 2 Dron autónomo de reconocimiento por termografía", 2019.

[11] S. Torterolo, J.M.Torterolo, F. Cayafa, "Experimento: Medida de Empuje", 2015.

[12] Intel, "Intel Realsense Depth Camera D415" [Online]. Disponible: https://store.intelrealsense.com/buy-intel-realsense-depth-camera-d415.html

[13] Emax Model, "Emax GT2215" [Online]. Disponible: https://emaxmodel.com/catalog/product/view/id/406/s/gt2215/

[14] Emax Model, "Emax Multicopter motor MT3510" [Online]. Disponible: https://emaxmodel.com/catalog/product/view/id/34/s/emax-multicopter-motor-mt3510/category/1783/

[15] Emax Model, "Emax GT2820" [Online]. Disponible: https://emaxmodel.com/catalog/product/view/id/424/s/gt2820/

[16] dronekit, "Dronekit-Python", (2019), Repositorio de GitHub, https://github.com/dronekit/dronekit-python

[17] ArduPilot, "Pymavlink", (2020), Repositorio de GitHub, https://github.com/ArduPilot/pymavlink

[18] OpenCV [Online]. Disponible: https://opencv.org/

[19] AWS, "AWS IoT Device SDK for Python", (2019), Repositorio de GitHub, https://github.com/aws/aws-iot-device-sdk-pythonl

[20] goodrobots, "Vision Landing", (2019), Repositorio de GitHub, https://github.com/goodrobots/vision_landing

[21] Tiziano Fiorenzani, "How Do Drones Work", (2019), Repositorio de GitHub, https://github.com/tizianofiorenzani/how_do_drones_work

[22] Open Source Computer Vision, "Detection of ArUco Markers" [Online]. Disponible: https://docs.opencv.org/3.1.0/d5/dae/tutorial_aruco_detection.html

[23] Open Source Computer Vision, "Image Thresholding" [Online]. Disponible: https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html

[24] Dronecode, "Flight Controller/Sensor Orientation" [Online]. Disponible: https://docs.px4.io/v1.9.0/en/config/flight_controller_orientation.html

[25] AtsushiSakai, "Python Robotics", (2019), Repositorio de GitHub, https://github.com/AtsushiSakai/PythonRobotics/blob/master/PathPlanning/GridBasedSweepCPP/grid_based_sweep_coverage_path_planner.py

[26] Intel RealSense, "librealsense", (2019), Repositorio de GitHub, https://github.com/IntelRealSense/librealsense/tree/master/wrappers/python

[27] GroupGets , "Pure Thermal 2 - Datasheet", https://groupgets-files.s3.amazonaws.com/PT2/PureThermal%202%20-%20Datasheet%20-%201.5.pdf

[28] Flir Lepton , "Flir Leptopn 3 & 3.5 - Datasheet", https://www.flir.es/globalassets/imported-assets/document/lepton-3-3.5-datasheet.pdf

[29] Usamentiaga, Ruben & Venegas, Pablo & Guerediaga, Jon & Vega, Laura & Molleda, Julio & Bulnes, Francisco G. (2014). "Infrared Thermography for Temperature Measurement and Non-Destructive Testing"[Online]. Disponible: https://www.researchgate.net/publication/263862713_Infrared_Thermography_for_Temperature_Measurement_and_Non-Destructive_Testing

[30] de Prada Pérez de Azpeitia, Fernando Ignacio. (2016). "La termografía infrarroja: un sorprendente recurso para la enseñanza de la física y la química"[Online]. Disponible: https://rodin.uca.es/xmlui/bitstream/handle/10498/18501/8-913-Prada.pdf?sequence=4&isAllowed=y

[31] Maldague, X. "Theory and Practice of Infrared Technology for Nondestructive Testing"Wiley: New York, NY, USA, 2001.

[32] GroupGets, "purethermal1-uvc-capture", (2019), Repositorio de GitHub, https://github.com/groupgets/purethermal1-uvc-capture

[33] Pyrealsense2, "LibrealsenseTM Python Bindings"[Online]. Disponible: https://intelrealsense.github.io/librealsense/python_docs/_generated/pyrealsense2.html

[34] "Emissivity Values for Common Materials" [Online]. Disponible: http://www.infrared-thermography.com/material-1.htm

[35] Amazon Web Services IoT Developer Guide, "AWS IoT Device Shadow service" [Online]. Disponible: https://docs.aws.amazon.com/iot/latest/developerguide/iot-device-shadows.html

[36] Amazon Web Services IoT Developer Guide, "Create the AWS IoT thing, certificate, and private key " [Online]. Disponible: https://docs.aws.amazon.com/iot/latest/developerguide/iot-moisture-create-thing.html

[37] Carfagni, Monica & Furferi, Rocco & Governi, Lapo & Santarelli, Chiara& Servi, Michaela & Uccheddu, Francesca & Volpe, Yary. (2019). "Metrological and Critical Characterization of the Intel D415 Stereo Depth Camera"[Online]. Disponible: https://www.researchgate.net/publication/330641397_Metrological_and_Critical_Characterization_of_the_Intel_D415_Stereo_Depth_Camera

[38] K. Hata, S. Savarese. "CS231A Course Notes 1: Camera Models", CS231A: Computer Vision, From 3D Reconstruction to Recognition, University of Standford, 2018. [Online]. Disponible: course_notes/01-camera-models.pdf

[39] University of Auckland, School of Computer Science, "Basics of Computational Stereo Vision" [Online]. Disponible: https://www.cs.auckland.ac.nz/courses/compsci773s1t/lectures/773-GG/topCS773.htm

[40] University of Washington, "Lecture 16Stereo and 3D Vision" [Online]. Disponible: https://courses.cs.washington.edu/courses/cse455/09wi/Lects/lect16.pdf

[41] Dronekit, "Example: Guided Mode Movement and Commands (Copter)" [Online]. Disponible: https://dronekit-python.readthedocs.io/en/latest/examples/guided-set-speed-yaw-demo.html

[42] tkinter, "tkinter — Python interface to Tcl/Tk" [Online]. Disponible: https://docs.python.org/3/library/tkinter.html

[43] S. Torterolo, J.M.Torterolo, F. Cayafa, "Termodron, Dron de vuelo autónomo y reconocimiento por termografía", 2015

[44] Wikipedia, "GPS" [Online]. Disponible: https://es.wikipedia.org/wiki/GPS

[45] "Geopy Documentation" [Online]. Disponible: https://geopy.readthedocs.io/en/stable/

[46] Wikipedia, "Great-circle distance" [Online]. Disponible: https://en.wikipedia.org/wiki/Great-circle_distance

[47] Dronekit, "Open Source Licence" [Online]. Disponible: https://dronekit-python.readthedocs.io/en/latest/about/license.html

[48] OpenCV , "OpenCV modules" [Online]. Disponible: https://docs.opencv.org/4.4.0/

[49] Tiziano Fiorenzani, "OpenCv and Camera Calibration on a Raspberry Pi 3" [Online]. Disponible: https://www.youtube.com/watch?v=QV1a1G4lL3U

[50] Mathworks Help Center, "What Is Camera Calibration?" [Online]. Disponible: https://www.mathworks.com/help/vision/ug/camera-calibration.html

[51] ArduPilot, "AutoTune" [Online]. Disponible: https://ardupilot.org/copter/docs/autotune.html

[52] ArduPilot, "Flight Modes" [Online]. Disponible: https://ardupilot.org/copter/docs/flight-modes.html

[53] ArduPilot, "Complete Parameter List" [Online]. Disponible: https://ardupilot.org/copter/docs/parameters.html#log-bitmask

[54] jcmb, "NTRIP", (2015), Repositorio de GitHub, https://github.com/jcmb/NTRIP

[55] Instituto Geográfico Militar, "Mapa dinámico de estaciones de referencia" [Online]. Disponible: http://www.igm.gub.uy/geoportal/estaciones/

[56] Instituto Geográfico Militar, "Servicios de la REGNA-ROU" [Online]. Disponible: http://www.igm.gub.uy/2016/05/20/servicios-regna-rou/

[57] Amazon Web Services, "AWS IoT: Developer Guide" [Online]. Disponible: https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html

[58] CubePilot, "Here+V2 User Manual" [Online]. Disponible: https://docs.cubepilot.org/user-guides/here+/here+v2-user-manual#here-here-v2-led-meaning

[59] ArduPilot, "Mandatory Hardware Configuration" [Online]. Disponible: https://ardupilot.org/copter/docs/configuring-hardware.html

[60] STorM32-BGC Wiki, "STorM32 NT brushless gimbal controller documentation." [Online]. Disponible: http://www.olliw.eu/storm32bgc-wiki/Downloads