Facultad de Ingeniería de la
Universidad de la República

Proyecto de fin de carrera de Ingeniería
Eléctrica

TERMODRON

Dron de vuelo autónomo y reconocimiento por termografía

Autores:
Damián Vallejo
Agustín Barriola
Ignacio Reyes

Tutor:
Ing. Rafael Canetti

5 de septiembre de 2017

Agradecimientos

A Jimena, Camila, Micaela, Martina y Lucía, muchas gracias por haber acompañado este largo año de trabajo, por haber soportado las horas de ausencia y por el gran apoyo que nos han brindado.

A nuestros familiares, amigos y compañeros de trabajo que también estuvieron apoyando, cada uno desde su lugar con aportes interesantes o simplemente alentando al progreso y éxito de nuestro trabajo.

Resumen

El objetivo de este proyecto es la implementación de un sistema autónomo para el reconocimiento de un punto caliente mediante termografía sensando un área objetivo. Dicho sistema se compone de dos unidades, una es el Dron y la otra es la base en tierra. Estas unidades se comunican entre sí mediante radio, enviando información relevante a ser procesada y/o transmitida al usuario. La comunicación con el usuario se realiza mediante una conexión a internet la cual le permite intercambiar información con el sistema, así como también asignarle tareas programadas vía correo electrónico. El proceso que desarrolla el sistema comienza con la recepción de un correo electrónico enviado por el usuario, delimitando el área a recorrer, el objetivo a cumplir y asignándole una fecha de realización. Una vez alcanzada dicha fecha, el sistema se desempeña sin la necesidad de una intervención externa. Calcula una trayectoria para recorrer el área objetivo y evalúa si se encuentra apto para realizarla. De encontrarse apto, la unidad Dron vuela siguiendo dicha trayectoria, buscando un punto caliente. En todo momento el Dron reporta a la base en tierra su estado y envía imágenes de lo observado. Una vez alcanzado su objetivo, retorna y aterriza en el punto de partida. Por último, reporta todos los datos obtenidos durante el vuelo al usuario vía correo electrónico.

Abstract

The target of this project is the implementation of an autonomous system for the recognition of a hot spot through termography, sensing a target area. This system has two units, one is the Drone and the other one is the ground base. These units communicate with each other by radio, sending relevant information to be processed and/or transmitted to the user. Communication with the users is done through an internet connection which allows them to exchange information with the system and also assign scheduled tasks via e-mail. The process that the system develops begins with the receipt of an e-mail sent by the user, delimiting the area to scout, the objective that it has to comply and assigning a date of completion. Once that date is reached, the system performs without the need for external intervention. It calculates a path to travel the target area and evaluates if it is in condition to start. If found itself ready, the Drone unit flies following that path, looking for a hot spot. The Drone always reports its state to the ground base and sends images of the observed. Once it reaches its target, it returns and lands at the starting point. Finally, it reports all the data obtained during the flight to the user via e-mail.

ÍNDICE GENERAL

Agradecimientos **III**

Resumen **V**

Abstract **V**

1. Introducción **1**

2. Resumen del Sistema Implementado **5**
2.1. Objetivo 5
2.2. Aplicaciones 5
2.3. Descripción funcional 6
2.4. Implementación física de las unidades 7
2.4.1. Unidad GCS 7
2.4.2. Unidad Dron 7

3. Componentes y Ensamblado del Sistema **13**
3.1. Selección de componentes 13
3.1.1. Algoritmo recursivo para dimensionamiento de batería y motores 15
3.1.2. Frame 16
3.1.3. Motores y ESCs 17
3.1.4. Batería 18
3.1.5. Hélices 21
3.1.6. Controlador de vuelo 22
3.1.7. Regulador de voltaje 22
3.1.8. Distribuidor de energía 23
3.1.9. Receptor GNSS 23
3.1.10. Sensores de distancia 24
3.1.11. Shield SD 24
3.1.12. RX-TX 25
3.1.13. Controlador GCS y on-board 25
3.1.14. Shield 3G para GCS 27
3.1.15. Control remoto 27
3.1.16. Sensor de humedad relativa y temperatura 28

3.1.17. Sensor de lluvia . . . 30
3.1.18. Cámara térmica . . . 32
3.2. Ensamblado . . . 32
3.2.1. Ensamblado del Dron . . . 32
3.2.2. Ensamblado de la GCS . . . 33
3.2.3. Sistema armado . . . 36

4. Implementación de los módulos funcionales del Dron 39
4.1. Módulo on-board . . . 39
4.1.1. Hardware on-board . . . 39
4.1.2. Software on-board . . . 41
4.2. Detección térmica . . . 47
4.2.1. Fundamentos de radiación térmica . . . 47
4.2.2. Hardware . . . 48
4.2.3. Software desarrollado . . . 54
4.3. Cálculo de waypoints . . . 55
4.3.1. Sistemas de coordenadas . . . 55
4.3.2. Distancia entre dos puntos . . . 60
4.4. Comunicación Pixhawk y Arduino . . . 63
4.4.1. Conexiones entre Pixhawk y Arduino . . . 63
4.5. Calibración y caracterización de equipos . . . 64
4.5.1. Calibración de los sensores de Pixhawk . . . 64
4.5.2. Calibración de los sensores de distancia . . . 65
4.5.3. Calibración de la cámara térmica . . . 69

5. Implementación de los módulos funcionales de la GCS 71
5.1. Software . . . 71
5.1.1. TaskScheduler . . . 71
5.1.2. Buffer de Arduino . . . 71
5.2. Tareas Implementadas . . . 73
5.2.1. Shield 3G . . . 74
5.2.2. E-Mail . . . 74
5.2.3. Transferencia de archivos al Shield . . . 77
5.2.4. Cálculo de waypoints intermedios . . . 81
5.2.5. Clase Punto . . . 81
5.2.6. Clase Pendiente . . . 81
5.2.7. Algoritmo implementado . . . 81
5.2.8. Raw14 a mapa de bits . . . 82
5.3. Implementación de comunicación con Pixhawk . . . 86
5.3.1. Heartbeat . . . 86
5.3.2. Mensajes secuenciales y Mission Protocol . . . 86
5.4. Comunicación GCS y usuario . . . 87

6. Entorno de Desarrollo 91
6.1. Entorno de desarrollo . . . 91
6.1.1. Piloto automático PX4 . . . 91
6.1.2. Sistema Operativo en tiempo real NuttX . . . 91
6.1.3. Simulador jMAVSim . . . 92
6.1.4. QGroundControl . . . 92

6.1.5. Plataforma IDE de Arduino 93

7. Pruebas **95**
7.1. GNSS vs DGNSS (estático y dinámico) 95
7.1.1. Descripción de las pruebas 95
7.1.2. Desarrollo de las pruebas 95
7.2. Vuelo autónomo 101
7.2.1. Descripción de la prueba 102
7.2.2. Desarrollo de la prueba 102
7.3. Detección de objeto caliente con la cámara Lepton 103
7.3.1. Descripción de la prueba 103
7.3.2. Desarrollo de la prueba 103
7.4. Envío de imágenes a la GCS 103
7.4.1. Descripción de la prueba 103
7.4.2. Desarrollo de la prueba 104
7.5. Envío de imágenes al usuario (vía e-mail) 104
7.5.1. Descripción de la prueba 104
7.5.2. Desarrollo de la prueba 104
7.6. Aterrizaje 104
7.6.1. Descripción de la prueba 104
7.6.2. Desarrollo de la prueba 105
7.7. Alcance de comunicación 105
7.7.1. Descripción de la prueba 105
7.7.2. Desarrollo de la prueba 105

8. Mejoras **107**
8.1. Sistema de carga automático 107
8.2. GCS móvil 107
8.3. Paneles solares en la GCS 108
8.4. Mejora implementación de la GCS 108
8.5. GNSS diferencial 108
8.6. Captura y envío de video 108
8.7. GUI 109
8.7.1. Interfaz web 109
8.7.2. Aplicación móvil 109
8.8. Verificación de convexidad 109
8.9. Conversión de imagen térmica con RGB 109
8.10. Cambio de cámara térmica 110

9. Conclusiones **111**

Anexos **112**

Anexo A. GNSS Diferencial **115**
A.1. Introducción 115
A.2. Método Diferencial 116
A.3. Descripción de NTRIP 117
A.4. Arquitectura de NTRIP 118
A.5. Descripción RTCM 119

Anexo B. Protocolos, Formatos e Interfaces **121**
B.1. Protocolos de comunicación 121
B.1.1. AT Commands 121
B.1.2. Xmodem 123
B.1.3. RTCM 124
B.1.4. MAVLink 126
B.2. Formato de Archivo 127
B.2.1. BMP 127
B.3. Puertos de comunicación 130
B.3.1. SPI 130
B.3.2. I^2C 132

Anexo C. Manual de Usuario **135**
C.1. Partes del producto 135
C.2. Puesta en funcionamiento 135
C.3. Reportes de la GCS 138
C.4. Comandos de usuario 138

Referencias **145**

CAPÍTULO 1

INTRODUCCIÓN

La palabra dron o el acrónimo UAV (Unmanned Aerial Vehicle) son muy conocidas en la actualidad y el diseño de estos sistemas conforma un campo de desarrollo relativamente moderno y en expansión. El término UAV se hizo conocido en los años 90 y describía a las aeronaves robóticas no tripuladas. Mientras que el término dron proviene del inglés "drone" cuyo significado original es zángano (ejemplar macho de la abeja) y luego se utilizó para nombrar a los UAV debido a la similitud del sonido que emiten.

El término UAV no es más que uno entre cerca de una docena de nombres que han ido recibiendo las aeronaves robóticas no tripuladas a lo largo de su existencia. Existencia que tiene su comienzo en la primera mitad del siglo XIX, cuando inventores como Cayley, Stringfellow, Du Temple y otros pioneros de la aviación, construyeron los primeros modelos de aeronaves no tripuladas, previo a intentar desarrollar aeronaves tripuladas[1] [2].

El término UAV cubre un vasto espectro de aeronaves, desde torpedos aéreos hasta los drones recreativos.

En la Figura 1.1 se muestran algunos de los nombres que han recibido estos vehículos a lo largo de la historia. Merece destacar el último, RPAS (Remotely-Piloted Aircraft System), al cual en el año 2011 la Organización de Aviación Civil Internacional, organismo especializado de Naciones Unidas para la aviación civil, publicó su Circular 328 [4] en la cual por vez primera reconoce a las aeronaves no tripuladas como aeronaves. Esto trae consigo todo un marco regulatorio al cual deben adaptarse y de entre todas las posibles tipologías escoge a las que se pilotan de manera remota para ser consideradas como aptas para la aviación civil y no otros tipos (como podrían ser las autónomas). En la Figura 1.2 se muestran algunos ejemplos de aeronaves tipo UAV.

Así se han acuñado los términos que a continuación se detallan y que tienen hoy en día una validez y aplicación internacional casi única en todos los ámbitos. Estos términos son:

- Aeronave pilotada remotamente (Remotely-Piloted Aircraft, RPA): una aeronave en la que el piloto al mando no está a bordo.
- Sistema de aeronave pilotada remotamente (Remotely-Piloted Aircraft System, RPAS): un conjunto de elementos configurables formado por un RPA, su estación de pilotaje remoto asociada (RPS – Remote Pilot Station), el sistema

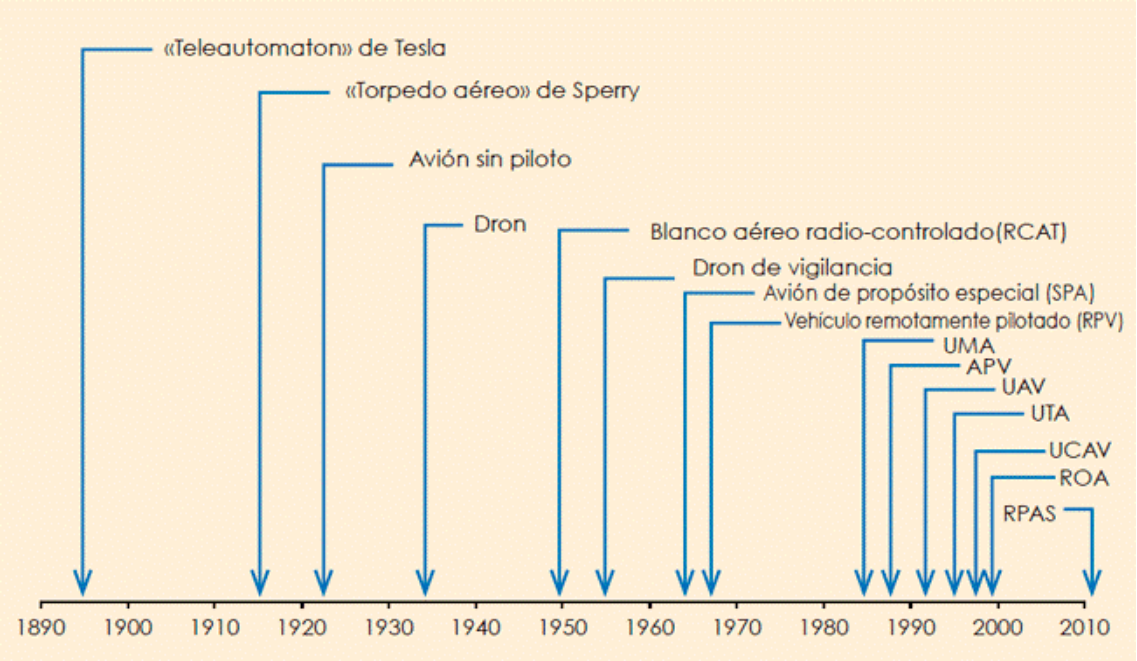


Figura 1.1: Cronología de los nombres aplicados a las aeronaves robóticas. Imagen obtenida de [3].

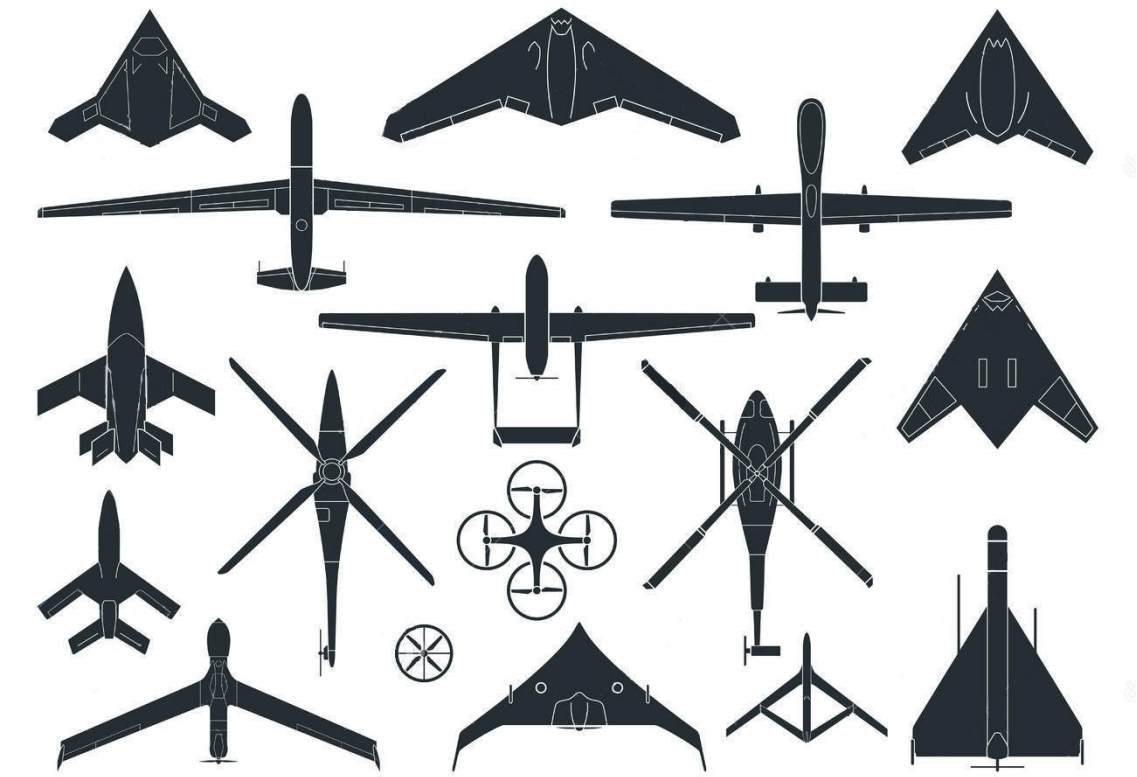


Figura 1.2: Distintos tipos de UAVs. Imagen obtenida de [5].

requerido de enlace de mando y control y cualquier otro elemento requerido en cualquier punto durante la operación del vuelo.

Recién en el año 2015, se empiezan a desarrollar los primeros drones autónomos recreativos, como lo son *Lily* y *Hexo+*[1] los cuales se mantienen a una determinada distancia de una pulsera transmisora, pero son incapaces de evitar objetos. La principal actividad que desarrollan es filmar a las personas mientras se desplazan. Normalmente se utilizan durante actividades deportivas como snowboarding, skiing, sandboarding, mountain bike, etc. Estos son los primeros drones autónomos comerciales que se conocen. Por último, en febrero del 2016, DARPA (Defense Advanced Research Projects Agency), en colaboración con el MIT (Massachusetts Institute of Technology), la Universidad de Pennsylvania, SSCI (Scientific Systems Co. Inc.) y la compañía AeroVironment, presentó su primer dron capaz de volar a una velocidad de hasta 72 km/h de forma autónoma y evitando obstáculos sin necesidad de ajustes o configuraciones. Se trata de un cuadricóptero desarrollado por el programa FLA [6] (Fast Lightweight Autonomy), el cual ha sido fabricado en base a la estructura de un dron DJI Flamewheel 450, pero con un sistema 3D Robotics Pixhawk, quien es el encargado del piloto automático. Dentro de este sistema se cuenta con cámaras, dispositivos de medición de aceleración, sensores LIDAR que utilizan un haz láser para determinar la distancia entre la aeronave y un objeto, además de su propio sonar.

En el Instituto de Ingeniería Eléctrica de la Facultad de Ingeniería de la Universidad de la República existe una línea de desarrollo e investigación que trabaja en UAVs con distintas características y diseños. La línea denominada uQuad [7] [8] [9] lleva desarrollados ya tres drones como proyectos de fin de carrera desde el año 2012. Si bien el trabajo que aquí se presenta no continúa estrictamente la línea de uQuad, aporta un cuarto cuadricóptero de vuelo autónomo al instituto dando paso a siguientes generaciones a continuar trabajando en esta área.

Este proyecto apunta a seguir esta línea de desarrollo en automatización de drones y ponerlos a disposición de las diferentes necesidades de la sociedad, mediante la implementación de un dron que logre detectar objetos a una temperatura diferente a la de su entorno. A su vez contará con sensores que le permitan ubicarse en el entorno y poder desplazarse evitando obstáculos de ser necesario. El dron mantendrá una comunicación constante con la base enviándole información acerca de su posición, cantidad de carga de batería, imágenes capturadas y recibiendo órdenes de aquella. En el caso de encontrar un "punto caliente" el dron reportará a la base, y ésta al usuario, la posición del punto y una imagen del mismo. La comunicación entre el usuario y el sistema (base + Dron) será a través de correo electrónico.

Para la detección térmica de objetos se utiliza una cámara térmica cuyo funcionamiento se basa en la ley de desplazamiento de Wien la cual relaciona la temperatura de un cuerpo con la longitud de onda a la cual el cuerpo irradia mayor potencia.

[1] https://hexoplus.com/product/hexo_drone_3d

CAPÍTULO 2

RESUMEN DEL SISTEMA IMPLEMENTADO

2.1. Objetivo

El objetivo que se propone en este proyecto consiste en la implementación de un sistema autónomo conformado por dos unidades. La primera es un Dron, al cual se le puede solicitar que releve una determinada zona en búsqueda de un objeto que se encuentre dentro de un rango de temperaturas. La segunda es una base en tierra, que establece comunicaciones con: el usuario a través de correo electrónico, un servidor de corrección de posicionamiento para mejorar la estimación de la posición del Dron y finalmente con el propio Dron para enviarle la información de la misión que debe ejecutar y solicitar información acerca del estado de este.

2.2. Aplicaciones

La principal razón por la cual se optó por la implementación de este proyecto fue el gran potencial que tiene a nivel de aplicaciones.

La principal aplicación que se pensó fue el reconocimiento, mediante termografía, de personas en lugares de difícil acceso debido a catástrofes naturales o como complemento en las tareas de rescate (e.g. inundaciones, terremotos, etc.). Otra aplicación interesante es detectar puntos calientes en una línea de transmisión de media o alta tensión. Pensando en Uruguay, la red 3G tiene una gran cobertura a lo largo y ancho del territorio lo cual hace factible el llegar a lugares aislados con la base para que luego el Dron recorra un área alejada. También se puede realizar relevamiento de áreas en búsqueda de animales o bien por seguridad en búsqueda de intrusos. El mantenimiento de un parque de paneles solares relevando la temperatura de los paneles puede ser una aplicación viable, así como el relevamiento de bosques en búsqueda de focos ígneos con el fin de evitar posibles incendios.

2.3. Descripción funcional

El sistema, que se encuentra representado a nivel funcional en la Figura 2.1, inicialmente establece comunicación con el usuario a través de correos electrónicos. Por este medio, la base en tierra, recibe por parte del usuario las coordenadas[1] que definen el perímetro del área a relevar, solicitudes de reportes periódicos durante el vuelo, de un reporte al final de la misión y comandos puntuales (e.g. aterrizar, ir a determinado punto, enviar una imagen). Los correos limitan la influencia del usuario sobre la misión. Una vez iniciada la misión, el usuario pierde el control de las decisiones de vuelo y únicamente podrá detener la misión o solicitar información. Como precaución a la hora de la misión se puede pasar a control manual desde un control remoto.

En determinadas circunstancias, también se envían correos electrónicos al usuario: al comenzar una misión, si una misión fue rechazada o interrumpida y si se localizó un objetivo dentro del área solicitada. Además, se envían correos como respuesta a solicitudes específicas: imágenes de la cámara térmica, bitácora de vuelo al final de la misión y estado general del cuadricóptero con determinado período.

El sistema también se comunica a través de internet con un servidor que proporciona información de corrección de GNSS[2] en tiempo real, comúnmente conocido como GPS diferencial. Con esto se puede mejorar la exactitud en la estimación de la posición del cuadricóptero en el espacio entre dos y tres órdenes de magnitud [10].

La implementación del sistema consta de dos elementos principales, el Dron y la Ground Control Station (GCS).

Las funciones de la GCS son:

- Recibir y procesar los correos electrónicos enviados por el usuario
- Enviar correos electrónicos al usuario con la información que este requiera
- Calcular los waypoints[3] de la misión a partir de los vértices que definen el área a relevar enviados por el usuario y evaluar la viabilidad de la misión
- Recibir del usuario y retransmitir al Dron los parámetros de misión seteados
- Recibir desde el servidor de corrección de posición y retransmitir al Dron las correcciones de GNSS en tiempo real
- Recibir la información de estado del Dron relevada por él mismo

Las funciones del Dron son:

- Ejecutar la misión según lo requerido por la GCS
- Enviar a la GCS la información que esta solicite (e.g. posición, velocidad, estado de la batería)

[1] A lo largo del documento se utilizan los términos coordenadas, puntos o waypoints que si bien todos refieren a un punto en el espacio, están bien diferenciados entre sí. En este caso, las coordenadas enviadas por el usuario corresponden a los vértices del área a recorrer por el Dron.

[2] Sistema Global de Navegación por Satélite: Constelación de satélites que transmite rangos de señales utilizados para el posicionamiento y localización en cualquier parte del globo terrestre.

[3] Los waypoints son las coordenadas de una misión que el Dron deberá recorrer.

- Ejecutar comandos puntuales solicitados por la GCS (e.g. cancelar la misión, enviar una foto, cargar una nueva misión, ir a determinado punto)
- Guardar una bitácora de vuelo con las variables de interés (latitud, longitud, altura, tiempo, carga de batería, imagen térmica, velocidades, consumo de los motores)
- Posicionar la cámara de manera que siempre se encuentre en un plano horizontal
- Estimar su estado en tiempo real y transmitirlo a la GCS

El objetivo del proyecto es entonces la implementación de algunos de estos bloques funcionales y la prueba del sistema completo. Sin embargo, hay bloques cuya implementación escapa al alcance del proyecto; en particular el controlador de vuelo, estimación del estado cinemático y control de gimbal; son implementados por una placa controladora. El desarrollo del software que se ejecuta en dicha placa no es parte de los objetivos de este proyecto.

2.4. Implementación física de las unidades

2.4.1. Unidad GCS

La GCS es la unidad encargada de establecer las comunicaciones con un usuario, con el Dron y con el servidor de corrección de GNSS. En la Figura 2.2 se observa el diagrama con las comunicaciones mencionadas. La GCS evalúa la validez de las solicitudes del usuario y reporta el estado del Dron. Cuenta con conexión a internet, puertos de comunicación y capacidad de procesamiento y almacenamiento que le permiten cumplir con estas funciones.

La GCS está compuesta por dos módulos, una SBC[4], un Shield 3G para la conexión a internet y un transceptor de telemetría para la comunicación de radio con el Dron, tal como se muestra en la Figura 2.3. En la Figura 2.4 se pueden observar los módulos y las funciones que cumple cada uno.

2.4.2. Unidad Dron

El Dron es una unidad conformada por dos partes, un cuadricóptero y un módulo on-board el cual está compuesto por cámara térmica, sensores de proximidad, shield SD y una SBC. A su vez, el cuadricóptero está compuesto por un frame o chasis, hélices, motores, ESC[5], distribuidor de energía, batería, transceptor de telemetría, gimbal, receptor de GNSS y un placa controladora de vuelo Pixhawk [11].

[4]Computadora en una sola placa por sus siglas en inglés.

[5]Controlador electrónico de velocidad por sus siglas en ingles es el componente encargado del control de la velocidad de los motores.

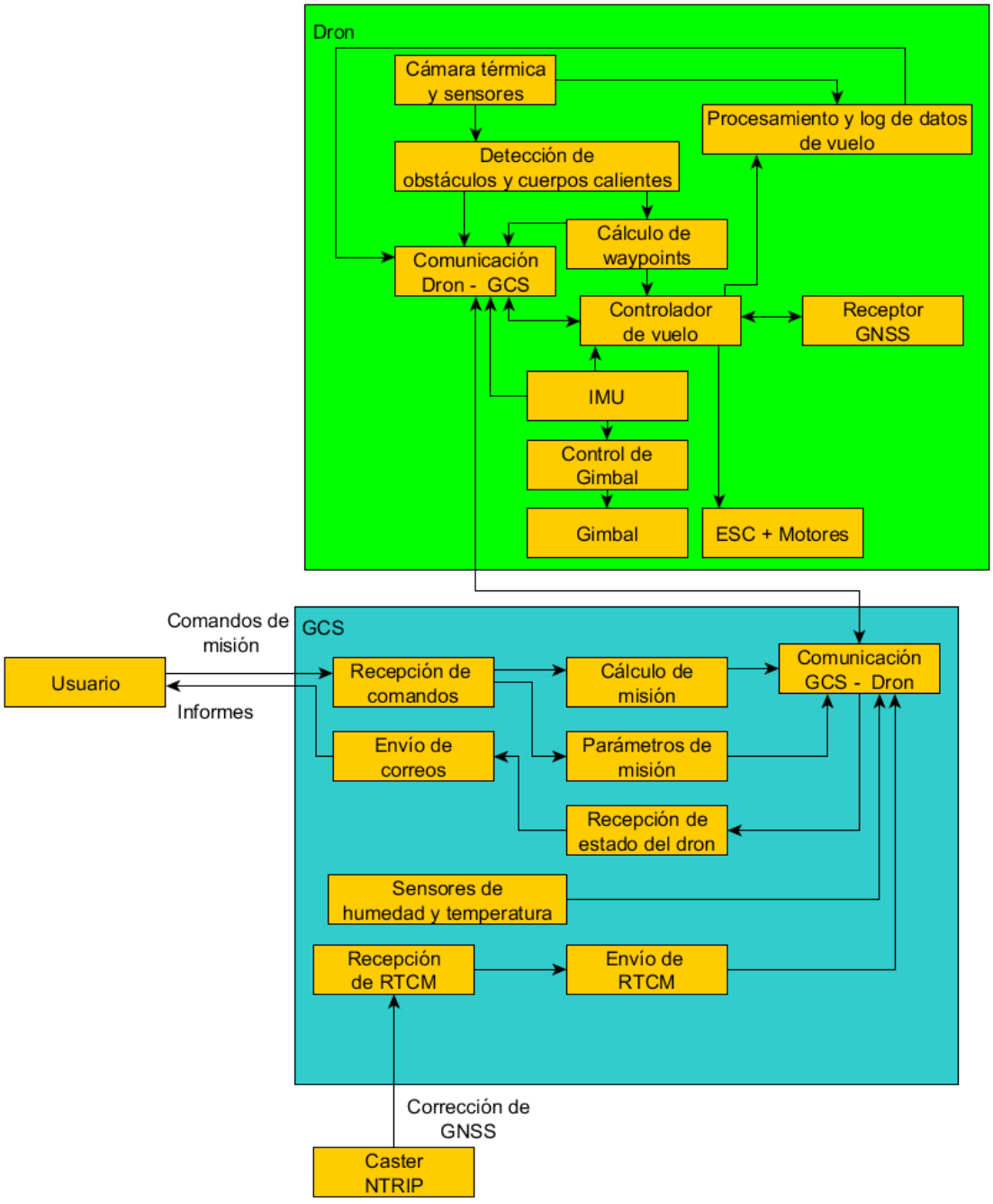


Figura 2.1: Diagrama funcional del sistema.

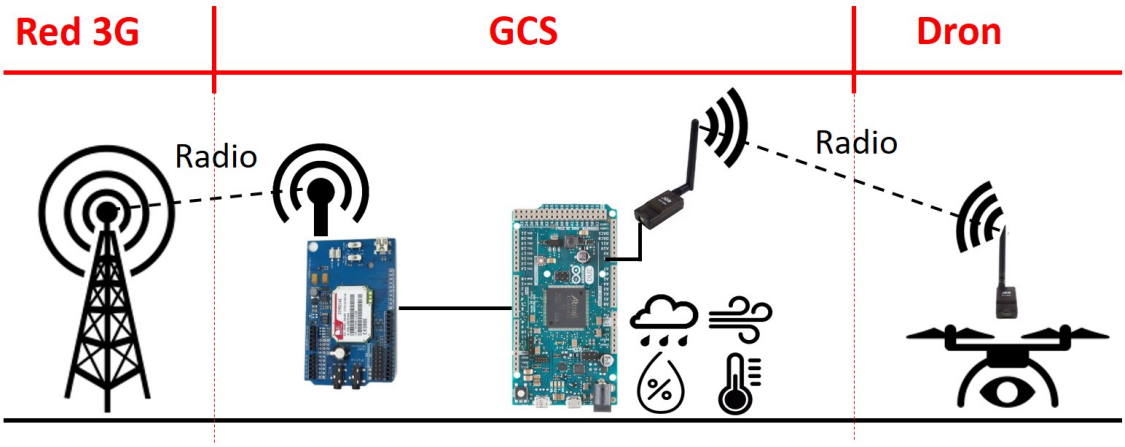


Figura 2.2: Diagrama de comunicaciones de la GCS.

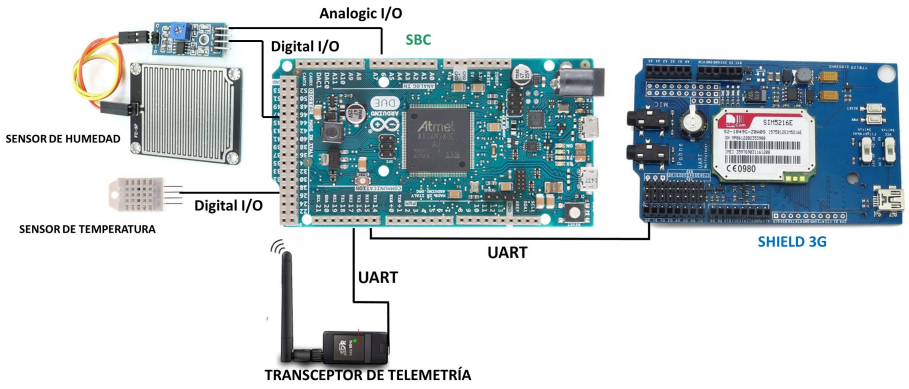


Figura 2.3: Diagrama eléctrico de la GCS.

El Dron es el encargado de realizar un vuelo en forma autónoma, detectar cualquier objeto en el rango de temperaturas definido, enviar información de su estado a la GCS y las imágenes de los objetos detectados. A su vez, el receptor de GNSS es el encargado de procesar la información de corrección recibida desde la GCS para mejorar la exactitud en la estimación de su posición. También es capaz de detectar obstáculos en su vuelo y poder eludirlos para la continuidad de su misión. En la Figura 2.5 se aprecian los componentes del Dron.

El Dron está comunicado con la GCS a través del componente de transmisión-recepción por radio. Desde la GCS es enviada la misión y la configuración inicial de vuelo. Además se envían solicitudes de información de vuelo e imágenes las cuales son atendidas mediante ese mismo canal de comunicación. Internamente, el Dron comunica sus dos partes a través de una comunicación UART[6] entre el controlador de vuelo y la SBC que compone el módulo on-board. Esta comunicación le permite al módulo on-board enviar información e imágenes al Usuario pasando a través del controlador de vuelo y además cambiar la misión si corresponde. El diagrama de los módulos y su comunicación se puede apreciar en la figura 2.4

La controladora de vuelo es la encargada de tomar las decisiones sobre el vuelo del Dron controlando los motores desde los ESC. Posee una IMU[7] la cual cuenta con acelerómetro, magnetómetro, giroscopio y barómetro. Con estos instrumentos y

[6]Universal Asynchronous Receiver-Transmitter, también conocida como Serie.
[7]Unidad de Medida Inercial

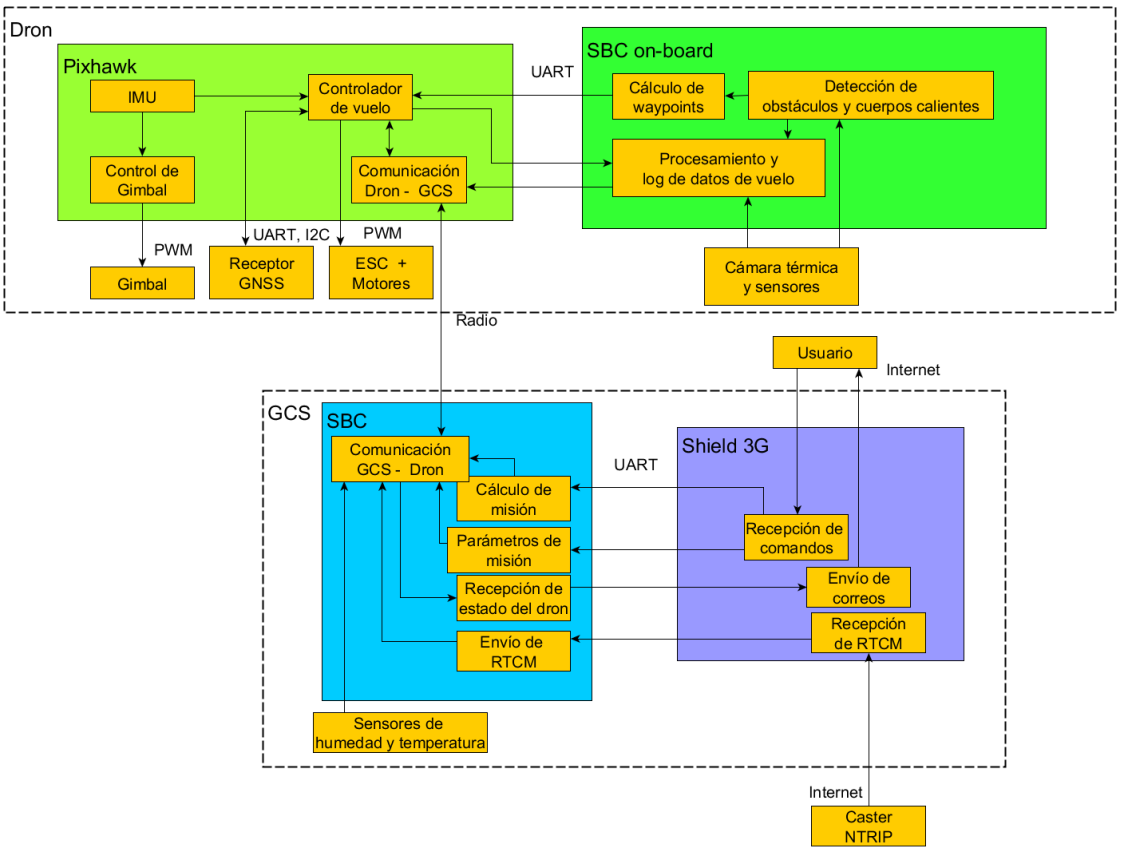


Figura 2.4: Diagrama de los módulos de Hardware que componen cada unidad, las funciones que implementan y los medios de comunicación que utilizan.

el receptor GNSS, se implementa un filtro de Kalman extendido [12] que es capaz de estimar la posición y movimiento de la aeronave para luego tomar decisiones de vuelo. Además, controla el gimbal en el que está montado el módulo on-board y su función es mantener dicho módulo orientado, en todo momento, paralelo al plano del suelo. Esto se logra compensando los ángulos de pitch y roll[8].

La principal tarea del módulo on-board, es la de controlar la cámara térmica y evaluar si en la imagen aparece un cuerpo dentro del rango de detección definido. Mediante un algoritmo, la SBC del módulo on-board recorre la imagen recibida desde la cámara en busca de un cúmulo de píxeles que hayan sensado una temperatura en el rango definido. La imagen térmica es convertida a BMP y almacenada en la memoria SD. Otra tarea del módulo on-board es relevar los sensores de distancia para comprobar si es necesario realizar una maniobra para evitar un obstáculo. Los sensores están ubicados, uno orientado hacia el frente, uno hacia cada lado del dron y uno hacia abajo. Este último sensor colabora con la cámara térmica para realizar un aterrizaje más seguro y preciso. Toda la información que releva el módulo on-board, ya sea el estado de la cámara térmica, estado de las comunicaciones de la SBC, sensores, etc. es almacenada en un archivo log en la memoria SD que se descarga en la GCS una vez terminada la misión y puede ser enviado al usuario si es solicitado.

[8]Estos ángulos, conocidos como ángulos de Euler, son ángulos de giro que permiten la inclinación hacia adelante o atrás y hacia la derecha o izquierda respectivamente. Más adelante en este documento se explican en profundidad estos términos.

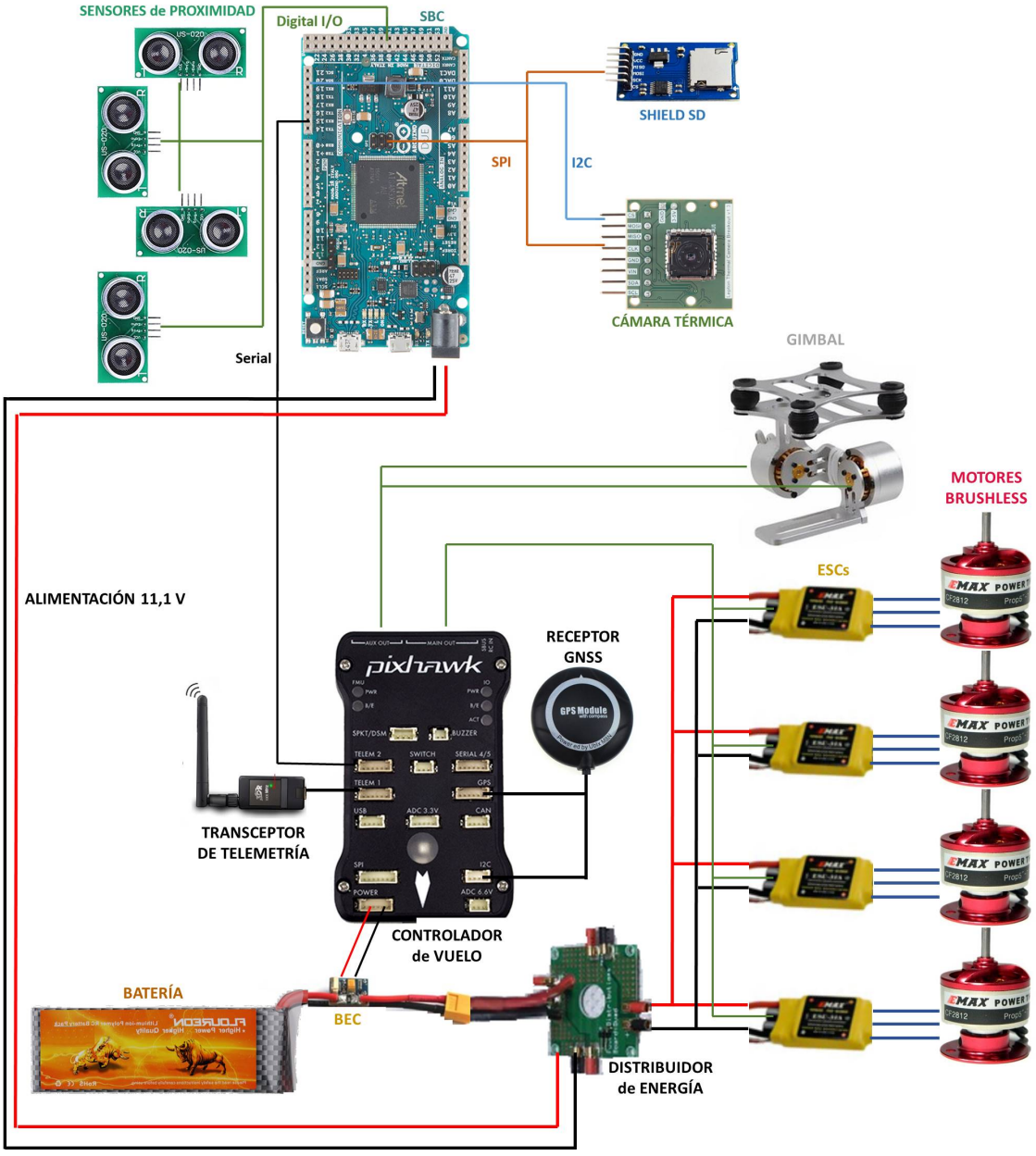


Figura 2.5: Diagrama de componentes del Dron.

CAPÍTULO 3

COMPONENTES Y ENSAMBLADO DEL SISTEMA

En este capítulo se hará una descripción más técnica y profunda del sistema implementado. El sistema se define como la composición de la GCS y el Dron. Cualquier otro elemento o parte descripta en el documento es un actor externo (e.g. el usuario).

A su vez el Dron se descompone en dos partes, el cuadricóptero y el conjunto gimbal, cámara térmica, sensores y placa programable.

A lo que al cuadricóptero se refiere, el mismo debe componerse de una estructura rígida (en forma de + o de x) con cuatro motores en sus extremos dotados de hélices que permitan impulsar principalmente el cuadricóptero hacia arriba venciendo la fuerza de gravedad y en segunda instancia poder mantenerlo en el aire y desplazarlo en cualquier dirección incluyendo el descenso. Para realizar los movimientos se definen lo que se conocen como ángulos de *Euler* o como se conocen en la jerga aeroespacial *Pitch*, *Roll* y *Yaw*. El *Pitch* es el ángulo que permite la inclinación del cuadricóptero hacia adelante o hacia atrás, el *Roll* hacia la izquierda o derecha y el *Yaw* según el eje vertical permitiendo un giro sobre el plano formado por los brazos del cuadricóptero. En la Figura 3.1 se puede observar con claridad cada uno de estos ángulos.

Otros términos también utilizados son *Hover* o *Loiter*, que refieren a mantener al cuadricóptero estático en el aire, y *Throttle* refiere al movimiento ascendente.

El primer paso a dar al momento de realizar este trabajo fue el de seleccionar cada uno de los componentes que integran el sistema.

3.1. Selección de componentes

Para poder implementar el sistema y dotarlo de las funciones mencionadas en el capítulo anterior, es necesario establecer un listado de los componentes físicos que van a conformar dicho sistema y un proceso de selección, comparando distintas soluciones y equipos de varios fabricantes de manera de poder tomar la decisión más acertada a la hora de efectuar la compra de los componentes. Errores en esta etapa se traducen en tiempos de espera de importaciones, pérdida de dinero y hasta reformulación de la solución planificada.

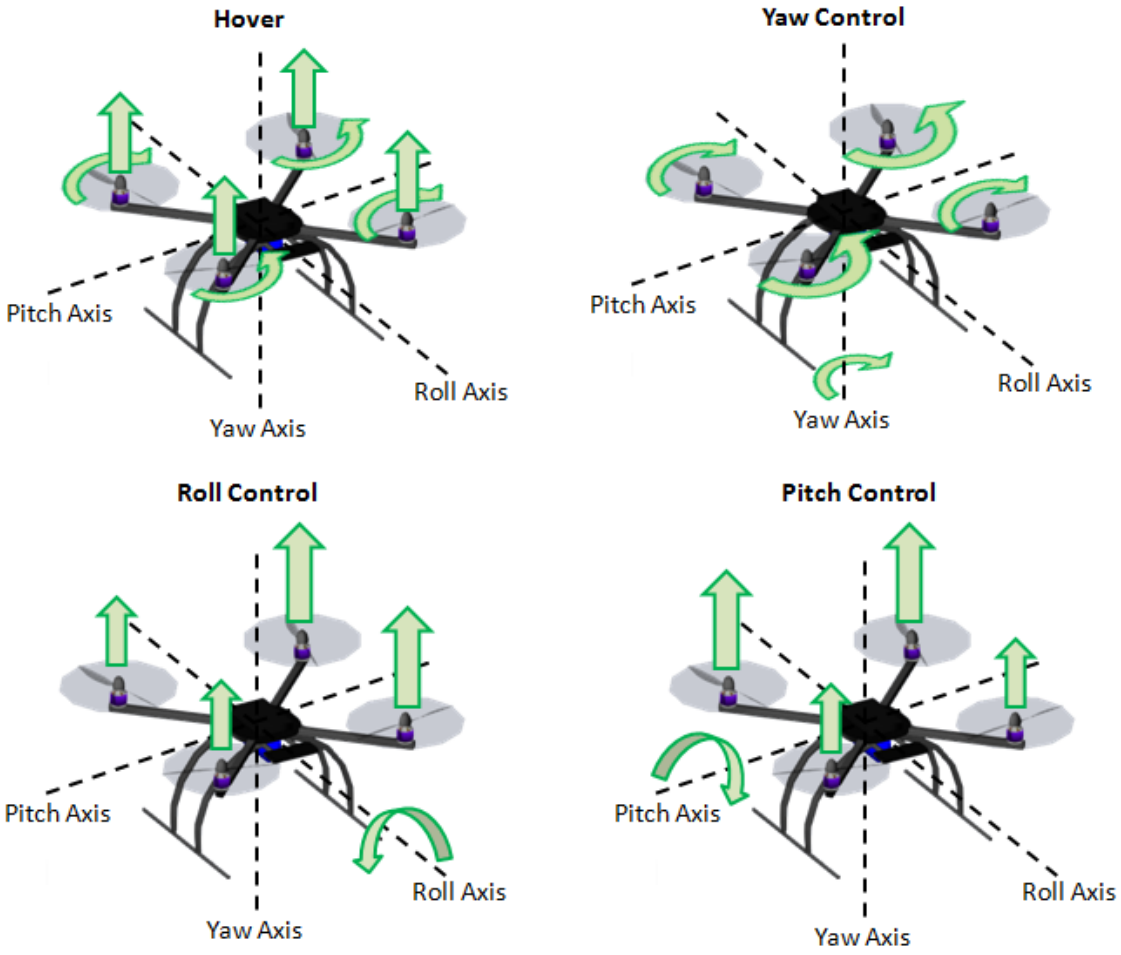


Figura 3.1: Diagrama de ángulos de Euler. Imagen obtenida de [13].

	Promedio	ZOP 10 Ah 3S 25C	TCB 10 Ah 3S 25C	Turnigy 5 Ah 3S 20C	ZOP 15 Ah 3S 35C
Energía específica (Wh/kg)	163.63	169.72	188.14	134.71	161.91

Tabla 3.1: Energía específica de varias baterías

	Promedio	Emax 2822 1200 KV	DST 1200 KV	Turnigy 3632 1200 KV	Turnigy 3020 1200 KV
Pot pico/ Thrust (W/kg)	228.61	226.69	227.69	230.05	230.00

Tabla 3.2: Relación entre potencia pico y thrust de varios motores

Es fundamental dedicar el tiempo necesario al estudio de cada componente y la intercompatibilidad entre los mismos de manera de minimizar la probabilidad de ocurrencia de dichos errores.

A continuación, se ofrece un listado y una breve descripción de los componentes necesarios para construir el sistema y se indican los componentes que se seleccionaron para este proyecto en particular y la justificación de estas elecciones.

3.1.1. Algoritmo recursivo para dimensionamiento de batería y motores

Dada la cantidad de grados de libertad y la falta de un modelo matemático preciso para el dimensionamiento de los componentes, se decide elegir variables relevantes para la selección de la batería y los motores:

- Energía específica (para las baterías, medida en Wh/kg)
- Potencia pico/Empuje (para los motores, medida en W/kg)

Para obtener un valor tipo de estas variables se evaluaron varios componentes de distintos fabricantes. Se pueden observar los resultados en las Tablas 3.1 y 3.2.

También se imponen ciertas condiciones de funcionamiento:

- Aceleración máxima en sentido vertical ascendente igual a la aceleración gravitatoria, o sea que el cociente entre el empuje total de los motores y el peso total del dron debe valer 2.
- Tiempo de vuelo mínimo igual a 10 minutos

- Masa del dron sin la batería de 1.19 kg

Dados los valores antes mencionados se realiza la siguiente iteración hasta obtener una primera estimación de los valores de capacidad de carga de la batería y empuje de los motores óptimos:

$M_{Dron} = 1.19 + M_{Bateria}$

$Th_{Motor} = \frac{M_{Dron}}{2}$

$P_{PicoMotor} = k \times Th_{Motor}$

$I_{PicoMotor} = \frac{P_{PicoMotor}}{V_{Motor}}$

$E_{10min} = \frac{P_{PicoMotor} \times 4 \times 10}{60}$

$C_{Bateria} = \frac{E_{10min}}{V_{Bateria}}$

$M_{Bateria} = \frac{E_{10min}}{e_{Bateria}}$

Donde M_{Dron} es la masa del Dron en kg, $M_{Bateria}$ es la masa de la batería en kg, Th_{Motor} es el empuje del motor en kg, $P_{PicoMotor}$ es la potencia pico del motor en W, k es el promedio de los cocientes entre potencia pico y thrust de la Tabla 3.2, $I_{PicoMotor}$ es la corriente pico del motor en A, V_{Motor} es la tensión nominal del motor en V, E_{10min} es la energía necesaria para mantener al Dron en vuelo al menos 10 minutos en Wh y $C_{Bateria}$ es la capacidad de la batería en Ah.

Como resultado de esta iteración se obtienen los siguientes valores:

$C_{Bateria} = 15.290\ Ah$

$M_{Bateria} = 1.037\ kg$

$Th_{Motor} = 1.136\ kg$

$I_{PicoMotor} = 22.93\ A$

La selección tanto de la batería como de los motores se realizó de modo de que cumplan lo mejor posible con los valores obtenidos en la iteración.

3.1.2. Frame

El soporte mecánico (también conocido como chasis, frame o central cross) del Dron debe ser fuerte y resistente, debe soportar tanto los golpes como las fuerzas realizadas por los motores sin deformarse. Debe, además, ser liviano y aerodinámico

para disminuir los efectos del viento. Por otro lado, también se debe tomar en cuenta que el tamaño sea el apropiado para poder acoplarle todo el hardware y demás elementos constructivos. Dentro de las características que se tomaron en cuenta para esta elección están: bajo peso, dimensiones adecuadas y bajo costo. Se eligió el ZD550 de LJI de plástico, fibra de carbono y pocas piezas de metal, que lo hacen muy liviano (630 g). Su diagonal es de 550 mm y cuenta con brazos retráctiles y tres planos para asegurar las demás piezas. El aspecto del frame se puede ver en la Figura 3.2.

En la elección original de este componente se cometió un error, en la primera compra se trajo un modelo mucho más liviano (200 g) pero sus dimensiones no eran adecuadas. Esto significó que hubo que comprar este nuevo frame más pesado pero la selección del resto de los componentes se hizo pensando en el primer frame. Es por esto que más adelante se verá que los componentes están subdimensionados con respecto a los calculados por el algoritmo de selección.

3.1.3. Motores y ESCs

El motor es uno de los componentes más importantes a la hora de construir un dron. Las cuatro características básicas a tomar en cuenta a la hora de elegir el motor son: el voltaje de funcionamiento, las rpm por Volt (KV), el thrust o empuje y su corriente nominal. Para este tipo de aplicaciones lo usual es utilizar motores sin escobillas con rotor exterior (brushless outrunner). Este es un motor de corriente continua controlado por un ESC[1], elemento que se introduce entre el controlador de vuelo y el motor, y que cumple la función de recibir una señal PWM del controlador para luego entregar una tensión trifásica escalonada al motor. El conjunto motor brushless más ESC sustituye el conmutador y las escobillas de los motores DC por switches de estado sólido que funcionan con una lógica para la conmutación de los bobinados, lo que implica una ventaja ya que no requieren mantenimiento por el desgaste del contacto móvil.

Los brushless de rotor externo son más pequeños y baratos que los de rotor interno y debido a esto los de rotor interno raramente se encuentran en el mercado [14]. El rotor es el elemento magnético permanente y el estator está formado por bobinados como se puede observar en la Figura 3.3. Para detectar la posición del rotor (para conocer la ubicación de los polos magnéticos) se utiliza normalmente sensores de efecto Hall o sensores ópticos y así se genera la señal de control mediante los switches electrónicos. El rotor externo tiene mayor diámetro que el rotor interno por lo que tiene un torque mayor, o lo que es lo mismo un valor de KV menor. Por último, otra ventaja que presentan los motores de rotor externo es que son diseñados para que sus ejes sean reemplazables. En este tipo de motores hay una relación entre las características constructivas. Los KV son inversamente proporcionales al thrust que cada motor puede entregar. Por otra parte, a mayor KV, el fabricante recomienda utilizar hélices de menor diámetro que a un KV menor.

Se decidió que los motores sean de 12 V debido a la disponibilidad de productos en el mercado. Basándose en los uQuad anteriores y dado que este dron es de características similares, se estimó que el peso del mismo será de aproximadamente 1.5

[1]Electronic Speed Controller: es un circuito electrónico cuya función es variar la velocidad de un motor eléctrico, generando energía en tres fases de baja tensión.

Figura 3.2: Frame ZD550 LJI

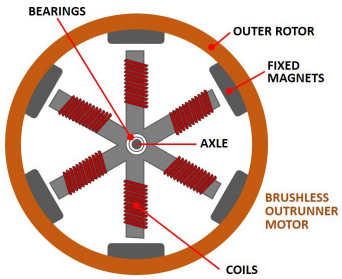


Figura 3.3: Componentes de un motor brushless outrunner. Imagen obtenida de [15].

kg. A medida que se fueron seleccionando los componentes se pudo verificar que el peso es de este orden, aunque en todo momento se sobrestima el peso considerando un peor caso. En la Tabla 3.3 se puede observar un valor estimativo del peso de cada componente y del peso total (en este listado no se incluye la batería). La selección de los motores a utilizar está ligada estrechamente a la elección de la batería, por lo que primero se introducirá este componente y se detallarán los pasos utilizados para llegar a dicha elección.

3.1.4. Batería

Las baterías son una solución al problema de almacenamiento de energía, convirtiendo la energía eléctrica en química, siendo ésta una forma fiable de almacenamiento y permiten independizar a los sistemas de la red de alimentación. El desarrollo en los últimos años de nuevas aplicaciones (vehículos eléctricos, sistemas fotovoltaicos, sistemas de alimentación ininterrumpida, etc.) han propiciado el surgimiento y desarrollo de nuevas tecnologías de baterías, cada vez más eficientes y fiables.

Se pueden clasificar en dos categorías: las baterías primarias, que incluyen a las baterías que no pueden ser recargadas cuando se agotan, a no ser que se repongan las sustancias químicas que las componen; y por otro lado las secundarias, que son baterías que pueden reactivarse si son sometidas al paso, más o menos prolongado, de una corriente eléctrica con sentido inverso al que fueron descargadas.

Las baterías primarias se utilizan principalmente para suministrar potencias ba-

Componente	Peso (g)
Chasis	630
Hélices	60
Motores + ESCs	240
Distribuidor de energía	50
BEC	10
Controlador de vuelo	50
Receptor GNSS	40
Sensores	40
Cámara térmica	10
RX-TX	50
Cámara	10
Total	**1190**

Tabla 3.3: Estimación de pesos de componentes

jas y por ende son de pequeño porte. Las secundarias se fabrican cubriendo un margen muy grande de capacidades que van desde cientos de mAh (capacidades bajas) hasta cientos de Ah (capacidades altas).

Normalmente los drones utilizan baterías de LiPo (Polímero de Litio) por ser estas las que mejor se adaptan a las necesidades del dron. El Litio (Li) es el metal más ligero conocido debido a que cuenta con solo 3 protones, por tanto, su peso atómico es muy bajo, permitiéndole tener un gran potencial químico, lo cual da gran capacidad de almacenaje con poco peso.

Como contraparte, el Litio metálico es muy inestable, y lo es más aún cuando se carga la batería. Esto hace que sean inseguras, por lo que los primeros fabricantes se volcaron a la utilización de formas químicas del Litio que no fueran metálicas.

Entre las ventajas de las baterías de la familia del Litio se tiene que estas no requieren mantenimiento, no tienen efecto memoria y no es necesario realizar un reciclado cada cierto número de cargas. Además, el ratio de autodescarga de una batería de LiPo es menos de la mitad que el de otros tipos de baterías.

Como puntos débiles se encuentran su estructura frágil y la dependencia de un circuito de seguridad para limitar el voltaje que puede alcanzar cada celda durante la carga y la descarga. También tienen un gran envejecimiento, debido a la gran degradación de las capacidades químicas.

Las baterías de LiPo utilizan un polímero sólido como electrolito, el cual permite alcanzar grosores de 1 milímetro y así fabricar baterías muy finas. Como contraposición el electrodo sólido tiene baja conductividad (alta resistencia interna) lo que le impide ofrecer la suficiente capacidad de descarga y eleva su temperatura hasta los 60 °C. Para que esto no ocurra se le agrega un gel al electrolito. Además, están contenidas en una bolsa flexible de aluminio, lo cual las hace más livianas que otros tipos de baterías de fundas rígidas y permite que sean empaquetadas de múltiples formas[16].

En la Figura 3.4 se pueden observar las principales características de los tipos de materiales más utilizados en la fabricación de baterías. Aquí observamos que la batería de LiPo tiene el mayor voltaje por celda, la menor autodescarga, la mayor eficiencia y la mayor potencia específica, lo que la convierte en la más apropiada

Tecnología Parámetros	Pb-ácido	Ni-Cd	Ni-MH	Li-ión [$LiCoO_2$]	LiFe	LI-PO
Voltaje (V/celda)	2v	1.2v	1.2v	3.6/3.7v	3.3v	3.7v
Autodescarga (%/mes)	3%-20%	10%	30%	8%	-	5%
Descarga en continua		10c	8c	1c	26c	20-45c
Descarga por picos	-	-	-	-	52c	30-90c
Mantenimiento	Bueno	Malo	Regular	Fácil	Bueno	Fácil
Ciclos de vida	500-800	1500-2000	300-500	400-1200	2000	>1000
Densidad energética [wh/l]	60-75	50-150	140-300	250-360	220	300
Energía específica [Wh/kg]	30-40	40-60	30-80	100-250	90-110	130-200
Potencia específica [W/Kg]	180	150	250-1000	250-340	3000	7100
Corriente carga rápida [C]	0.4	1- 2	1-2	1	4	1-2
Eficiencia. Carg/Desca	50%-92	70%-90%	66%	80%-90%	-	99.8%
Tolerancia a sobrecargas	-	M. buena	Media	M. mala	Mala	M. mala
Robustez a impactos	Buena	M. buena	Buena	M. mala	Media	M. mala
Altas temperaturas	Media	M. buena	Media	M. mala	Mala	M. Mala
Problemas de ecualización	No	No	No	Si	Si	Si
Seguridad	M. buena	M. buena	M. buena	M. buena	M.buena	Buena
Formato	-	Cilíndrico	Cilíndrico	Prisma	Pris/Cilin	Prisma

Figura 3.4: Características principales de los materiales más utilizados en la fabricación de baterías. Imagen obtenida de [17].

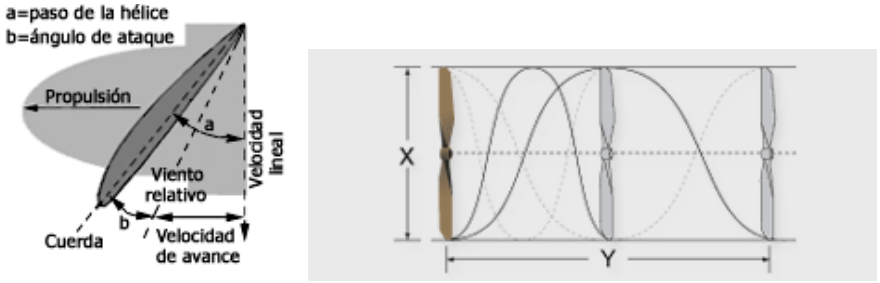
para alimentar el dron. No se debe olvidar sin embargo que son las más inseguras y peligrosas, llegando incluso a explotar ante una sobrecarga o cortocircuito, o incluso si son perforadas.

Se optó por una batería de tres celdas de LiPo (11.1 V) debido a la amplia disponibilidad comercial y que el voltaje máximo solicitado por los componentes del dron es de 12 V.

Dada la elección de la batería, se continuó con la selección de los motores optándose por unos Emax CF2822[18] de 12 V, 1200 KV, que girando a 6100 rpm consume 14.5 A proporcionando un thrust de 745 g.

3.1.5. Hélices

La hélice es una estructura diseñada con el fin de producir una fuerza propulsora al girar alrededor de un eje. Están formadas por varias aspas y el número de estas depende de la función que deba cumplir la hélice y del medio en el que deba desarrollarla. Cada aspa tiene un conjunto de perfiles aerodinámicos que disminuyen progresivamente su ángulo de incidencia desde el eje hasta los extremos. La hélice está acoplada directamente o a través de engranajes o poleas (reductores) al eje de salida de un motor, el cual proporciona el movimiento de rotación. Los perfiles aerodinámicos de una hélice están formados por un ángulo de ataque (respecto al viento relativo de la pala que es cercano al plano de revolución de la hélice) y un paso (igual al ángulo de incidencia). El giro de la hélice acelera el flujo de aire hacia el borde de salida de cada perfil, a la vez que lo empuja hacia abajo, es esta aceleración de una gran masa de aire la que provoca una fuerza de reacción que es la que propulsa el sistema tal como se puede observar en la Figura 3.5.



(a) Ángulos teóricos de una hélice.

(b) X:diámetro de la hélice; Y:pitch

Figura 3.5: Descripción de componentes de una hélice. Imágenes obtenidas de [19] y [20]

Las hélices se fabrican con torsión, esto implica que el ángulo de incidencia varíe de forma decreciente desde el eje hasta los extremos de forma progresiva. Cuando una hélice gira se tiene mayor velocidad tangencial en los extremos que la parte más cercana al eje por lo que es necesario compensar esta diferencia para producir una fuerza de forma uniforme. La "torsión" consigue menor velocidad, pero mayor ángulo en el centro de la hélice y una mayor velocidad pero menor ángulo hacia los extremos, produciendo una fuerza uniforme a lo largo de toda la hélice y reduciendo así las tensiones internas y las vibraciones.

Un punto crítico en el diseño radica en la velocidad con que giran los extremos, porque si está próxima a la velocidad del sonido, se produce una gran disminución en el rendimiento. Este hecho pone límites al diámetro y las rpm de las hélices. La fuerza de propulsión del dron está directamente relacionada con la cantidad de aire que mueve y la velocidad con que lo acelera; depende por tanto del tamaño de la hélice, de su paso, y de su velocidad de giro. Su diseño, forma, número de palas, diámetro, etc. debe ser el adecuado para la gama de velocidades en que puede operar el motor[21].

Las hélices se clasifican básicamente en hélices de paso fijo y hélices de paso variable. Se denomina paso de la hélice al ángulo que forma la cuerda de los perfiles de las palas con el plano de rotación de la hélice. En el caso de las hélices de cuadricópteros son de paso fijo.

Dada la elección del motor y dado que el fabricante recomienda el uso de hélices 1047, se opta por hélices de fibra de carbono cw/ccw (clockwise / counterclockwise) de 10 pulgadas de diámetro por 4.7 pulgadas de paso. Se necesitan igual número de hélices girando en sentido horario que en sentido antihorario para que la componente vertical del torque neto del dron con respecto al centro de masa sea nula. Las hélices largas le dan al dron mayor estabilidad a cambio de girar a una velocidad menor.

3.1.6. Controlador de vuelo

El controlador de vuelo es un componente crucial para el dron. Cuenta con los sensores necesarios para controlar la estabilidad y trayectoria del sistema y con el programa que permite su funcionamiento. A su vez es quien manda la señal a los ESCs para actuar sobre la velocidad de giro de los motores en función de todas las variables que está sensando.

Generalmente todos los controladores de vuelo cuentan con los mismos componentes: CPU, memoria, giroscopio, acelerómetro, magnetómetro, barómetro y entradas y salidas digitales para recibir datos de sensores externos y enviar comandos e información al exterior.

Entre las opciones disponibles en el mercado se evaluaron tres posibilidades y finalmente se decidió por el controlador Pixhawk PX4[22] debido a que es, dentro de las opciones comerciales, el que tiene un mejor procesador y más memoria, lo cual debería ser más que suficiente para todas las tareas de automatización que se desea que el dron realice. La comparativa entre las diferentes opciones se puede observar en la Tabla 3.4. En ella se puede ver que la opción de Pixhawk es la que tiene mayores prestaciones. Además, es importante destacar que es la única que cuenta con aritmética de punto flotante, lo que le da mayor rango de operación a nivel computacional.

3.1.7. Regulador de voltaje

El regulador de voltaje, más conocido por su sigla en inglés BEC (Battery Eliminator Circuit) es un componente electrónico que típicamente forma parte del ESC, pero también puede estar incluido en el receptor de radio control. El BEC permite

Modelo	CPU	Memoria	Redundancia en los sensores
ArduPilot Mega 2.8	8 bits ATMega 2560-16 MHz	8 kB RAM, 256 kB Flash	No
Lynxmotion Quadrino Nano	8 bits ATMega 2560-16 MHz	8 kB RAM, 256 kB Flash	No
3DR PX4 Pixhawk	32 bits ARMCortex M4-168 MHz	256 kB RAM, 2 MB Flash	Si

Tabla 3.4: Comparación entre las diferentes opciones de controladores de vuelo

que un dron pueda llevar una sola batería en lugar de dos (una para fuerza motriz y otra para alimentar el equipamiento de control). Este elemento logra un ahorro de peso al dron.

Un ESC equipado con BEC a menudo incorpora un circuito de baja tensión de corte (LVC) que detecta la caída de tensión causada cuando la batería tiene poca carga. Este sistema disminuye la alimentación del motor con el fin de proporcionar la potencia suficiente al dron para que sea capaz de volver de manera segura al operador. La potencia a la hélice se recorta pero la operación de los controles se mantiene con el fin de poder realizar un aterrizaje de emergencia (dead-stick). Sin esta característica, todo el control se perdería cuando la carga de la batería expiró, probablemente dando como resultado la destrucción del modelo.

Los BEC más simples utilizan un regulador de tensión lineal fijo en su circuito y por lo general se usan cuando el receptor requiere 5 V y un consumo de 1,5 A a 2 A. Cuando la corriente solicitada es de 5 A o más se utiliza un BEC con modo de conmutación, el cual es eléctricamente más eficiente que el BEC de regulador lineal. Esto se debe a que las pérdidas por disipación de potencia en un regulador BEC lineal, para un consumo de 5 A, son del orden de los 35 W y obliga a instalar disipadores de calor de gran tamaño teniendo una eficiencia menor al 50 %. En cambio, un regulador de modo de conmutación puede alcanzar una eficiencia mayor al 90 %. Se pueden colocar condensadores para amortiguar la salida regulada[23].

3.1.8. Distribuidor de energía

Se eligió un distribuidor que cumpliera con las características eléctricas de los motores y la batería, y que fuera solo para 4 motores para no agregar peso extra.

3.1.9. Receptor GNSS

El receptor GNSS elegido es el U-blox NEO-M8[24] como el que se muestra en la Figura 3.6. El NEO-M8, a diferencia de sus antecesores, soporta DGNSS (GNSS diferencial) lo cual permite lograr una exactitud mucho mayor que un receptor de GNSS normal al obtener información de corrección de posición de la base de datos de

la Red Geodésica Nacional Activa de la República Oriental del Uruguay (REGNA-ROU). La estación de la REGNA-ROU conoce con gran precisión su posición (ya que es un punto fijo en la tierra) y conoce su posición por GNSS, por tanto se obtiene el error generado por el GNSS y permite a cualquier sistema corregir su posición simplemente restando dicha diferencia.

Figura 3.6: Receptor GNSS u-blox M8.

3.1.10. Sensores de distancia

Se eligió el sensor de distancia USPRO® US-015 por tener 4.5 m de alcance y un consumo en reposo de 15 mW, el ángulo de apertura es de 15 ° por lo que puede detectar un objeto a esa distancia en un frente de detección de 118 cm, el Dron tiene un ancho de 68 cm por lo que estos sensores no dejan ningún punto ciego en la dirección del movimiento del Dron. Su principio de funcionamiento es por ultrasonido. En la Figura 3.7 se presenta el sensor utilizado.

Figura 3.7: Sensor de proximidad.

3.1.11. Shield SD

Se optó por un shield SD[25] con comunicación SPI para el almacenamiento de toda la información recabada durante las misiones realizadas.

3.1.12. RX-TX

Se eligió el kit de radio telemetría YKS 3DR[26] de 915 MHz del UHF ya que es compatible con Pixhawk y es el más utilizado en el ámbito de cuadricópteros. En la Figura 3.8 se muestran los transceptores con sus antenas. Entre los 902 MHz y los 826 MHz del UHF se reservan para Radioaficionados, Fijo, Radiolocalización y Móvil salvo móvil aeronáutico[2].

Figura 3.8: Transceptores 3DR de 915 MHz.

3DR también utiliza los 433 MHz del UHF que, según la URSEC, se reservan para Radioaficionados y Radiolocalización.

3.1.13. Controlador GCS y on-board

Tanto para el control de la GCS como del módulo on-board se seleccionó la placa programable Arduino dado el bajo precio y versatilidad de sus productos. Entre las opciones de Arduino se opta por el modelo DUE[3] por su mayor capacidad de procesamiento y almacenamiento, mejor velocidad de reloj y tamaño adecuado al propósito de este trabajo. En la Figura 3.9 se presenta la tarjeta Arduino DUE. En la Tabla 3.5 se puede observar otras opciones de Arduino comparadas con el modelo elegido.

[2]Unidad Reguladora de Servicios de Comunicaciones. En el siguiente sitio web se puede observar un mapa con la utilización de los espectros radioeléctricos del Uruguay https://www.ursec.gub.uy/inicio/informacion-tecnica/telecomunicaciones/atribucion-de-frecuencias/

[3]Esquemático de Arduino DUE[27] en https://www.arduino.cc/en/uploads/Main/arduino-Due-schematic.pdf

	UNO	**MEGA2560**	**DUE**
Microcontroller	ATmega328P	ATmega2560	AT91SAM3X8E
Operating Voltage	5 V	5 V	3.3 V
Input Voltage (recommended)	7-12 V	7-12 V	7-12 V
Input Voltage (limits)	6-20 V	6-20 V	6-16 V
Digital I/O Pins	14 (of which 6 provide PWM output)	54 (of which 15 provide PWM output)	54 (of which 12 provide PWM output)
Analog Input Pins	6	16	12
Analog Output Pins	N/A	N/A	2 (DAC)
Total DC Output Current on all I/O lines	40 mA	20 mA	130 mA
DC Current for 3.3 V Pin	50 mA	50 mA	800 mA
DC Current for 5 V Pin	N/A	N/A	800 mA
Flash Memory	32 kB (ATmega328) (0.5 kB used by bootloader)	256 kB (8 kB used by bootloader)	512 kB all available for the user applications
SRAM	2 kB (ATmega328)	8 kB	96 kB (two banks: 64 kB and 32 kB)
Clock Speed	16 MHz	16 MHz	84 MHz
Length	68.6 mm	101.5 mm	101.5 mm
Width	53.4 mm	53.3 mm	53.3 mm
Weight	25 g	37 g	36 g
EEPROM	1 kB (ATmega328)	4 kB	N/A
LED_BUILTIN	13	13	13

Tabla 3.5: Comparación técnica entre Arduino UNO, MEGA2560 y DUE.

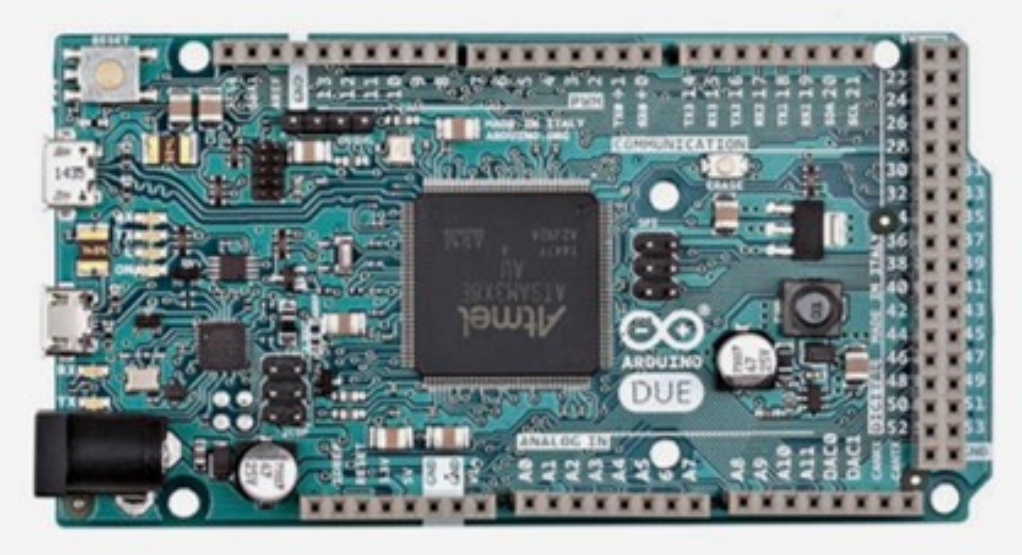


Figura 3.9: Arduino DUE.

3.1.14. Shield 3G para GCS

Se eligió el shield ITEAD[28] 3G GSM por tener comunicación 3G, la cual posibilita el envío y recepción de datos a través de internet. Otras soluciones encontradas en el mercado solo manejaban GSM lo cual, si bien cumple con el requisito de conexión a internet, su velocidad es menor. El shield está basado en un integrado SIM5216E el que es controlado vía comandos AT y es compatible con el controlador elegido (Arduino DUE). Soporta los estándares GSM 850 MHz, EGSM 900 MHz, DCS 1800 MHz, PCS 1900 MHz, WCDMA 850 MHz, WCDMA 1900 MHz y HSDPA.El shield se puede observar en la Figura 3.10.

Figura 3.10: Shield 3G.

3.1.15. Control remoto

Se optó por el control remoto Turnigy TGY9X[29] como el de la Figura 3.11 que trabaja a 2.4 GHz y tiene 9 canales de comunicación para poder hacer pruebas.

Este equipo no formará parte del sistema final pero sirve de ayuda para las pruebas. También es importante tener back-up por razones de seguridad y poder hacer un override al piloto automático en cualquier momento.

Figura 3.11: Control remoto.

3.1.16. Sensor de humedad relativa y temperatura

Se optó por el sensor DHT22[30] que incluye el sensado de temperatura y humedad relativa en el mismo encapsulado. En la Figura 3.12 se observa dicho sensor. El mismo consta de un capacitor de polímero para medir la humedad relativa, un termistor para medir la temperatura y un circuito integrado para el procesamiento y envío de los datos. En la Figura 3.13 se observan el termistor junto a su curva característica y el capacitor. El capacitor está compuesto de un sustrato sobre el que se deposita una película delgada de óxido de polímero o de metal entre dos electrodos conductores. El sustrato es típicamente de vidrio, cerámica o silicio. El cambio incremental en la constante dieléctrica de un sensor de humedad capacitivo es casi directamente proporcional a la humedad relativa del medio ambiente circundante. Este cambio en la resistencia es medido por el circuito integrado. Los sensores capacitivos están caracterizados por bajo coeficiente de temperatura, capacidad de funcionar a altas temperaturas (hasta 200 °C), la recuperación total de la condensación, y una resistencia razonable a vapores químicos. Un termistor es una resistencia variable con los cambios de temperatura. Estos sensores son hechos por sintetización de materiales semiconductores, tales como la cerámica o polímeros con el fin de proporcionar cambios más grandes en la resistencia con sólo pequeños cambios en

la temperatura. El término “NTC” significa “coeficiente negativo de temperatura”, lo que significa que la resistencia disminuye con el aumento de la temperatura.

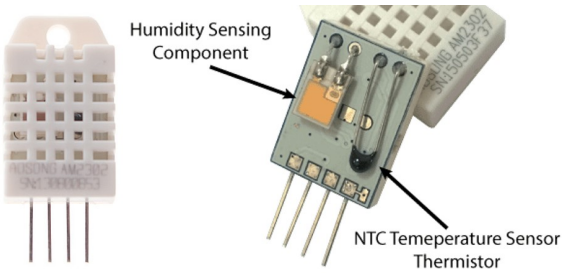
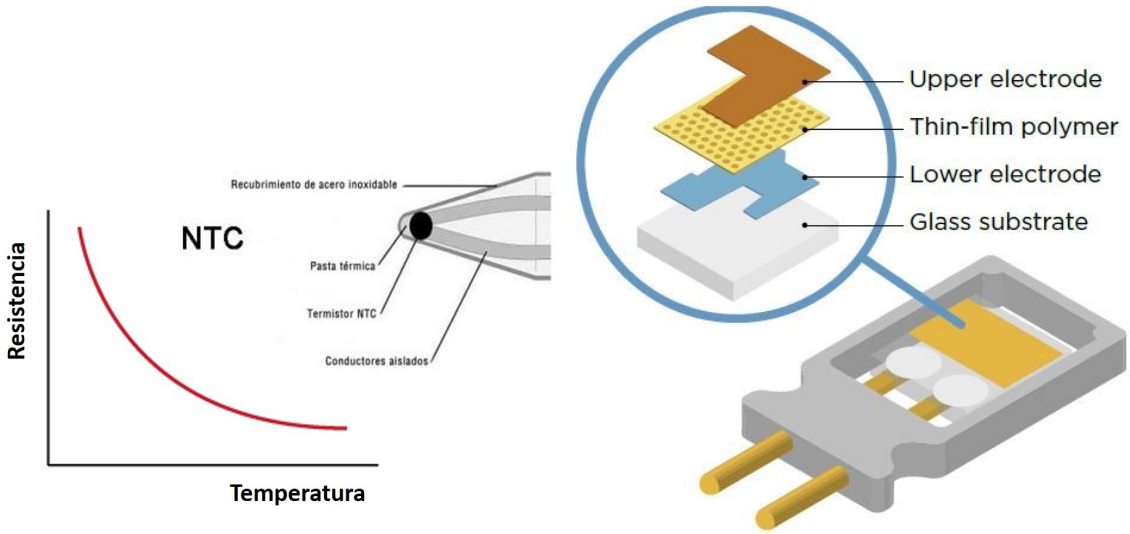


Figura 3.12: Sensor DHT22.



(a) Termistor (b) Capacitor de polimero

Figura 3.13: Componentes del sensor DHT22.

El sensor es calibrado en una cámara con un ambiente controlado, generándose un coeficiente de calibración que es almacenado en la memoria OTP del dispositivo. Soporta un voltaje de alimentación de entre 3.3 y 6 V de corriente continua y un consumo en corriente de entre 1 y 1.5 mA. El rango de temperaturas que puede sensar va desde los -40 °C hasta 80 °C, La precisión es de ±0.5 °C, la resolución es de 0.1°C y toma una nueva medida cada 2segundos.

El rango de humedad relativa que puede sensar va desde el 0

El sensor DHT22 utiliza un solo bus simplificado para la comunicación, en el que se aplica una sola línea de datos para el intercambio de datos y de control de estos en el sistema. El formato de envío de datos envía 40 bits de datos en una transmisión, el bit más alto primero. El diagrama de comunicación correspondiente se muestra en la Figura 3.14.

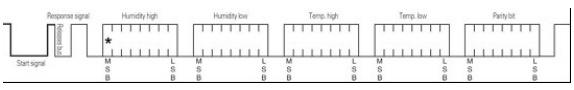


Figura 3.14: Transferencia típica de datos.

Al iniciar el sensor realiza una medida de temperatura y humedad y pasa a modo espera quedando el bus de comunicación en nivel alto debido a la resistencia de pull-up del sensor. La comunicación sigue los siguientes pasos: primero la placa Arduino de la GCS le indica al sensor DHT22 el inicio de una comunicación poniendo en nivel bajo el pin durante al menos 1ms. El bus pasa nuevamente a nivel alto como el efecto de la resistencia pull-up y el sensor pasará de modo de espera al modo de alta velocidad. Luego de que Arduino libera el bus el sensor envía una respuesta que consta de un bajo nivel de 80 ms y un nivel alto de 80 ms para informar que comenzará con la transmisión de datos. El inicio de la transmisión se muestra en la Figura 3.15.

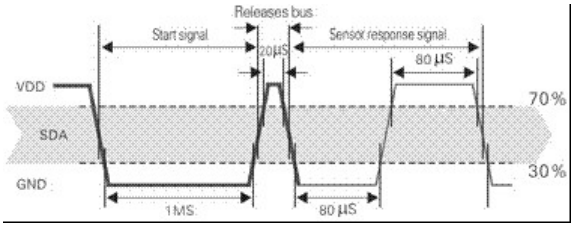


Figura 3.15: Inicio de transferencia datos.

A continuación, envía la cadena de 40 bits de datos en el siguiente orden: humedad byte alto, humedad byte bajo, temperatura byte alto, temperatura byte bajo y 1 byte de check-sum. Si la transmisión de datos es correcta, el check-sum debe ser igual a los últimos 8 bits de:
humedad_byte_alto+humedad_byte_bajo+temp_byte_alto+temp_byte_bajo.
Ahí comienza el envío de los datos, donde cada bit a transmitir comienza con un nivel de voltaje bajo y la longitud de la señal de alto voltaje, determine que el bit es "1.º "0". Un dato "0": nivel bajo durante 50 ms y nivel alto durante 26-28 ms. Un dato "1": nivel bajo durante 50 ms y nivel alto durante 70 ms. En la Figura 3.16 se observa con claridad la diferencia entre ambos datos.

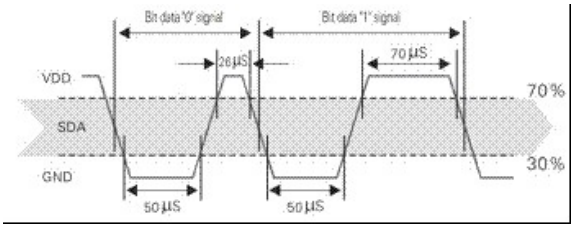


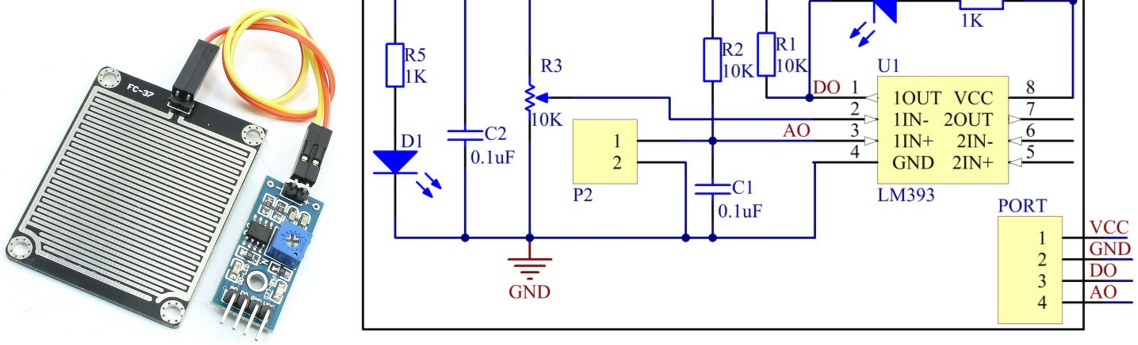
Figura 3.16: Ejemplo de un dato "0ƶ un dato "1".

Luego de finalizado el envío de datos el sensor mantiene el nivel bajo en el bus durante 50ms, pasa a sensar nuevamente la temperatura y la humedad y al concluir esto pasa a modo de espera automáticamente, aguardando por una próxima comunicación.

3.1.17. Sensor de lluvia

Se optó por un sensor de lluvia[31] el cual consta de un tablero de detección y un circuito de ajuste y se puede observar en la Figura 3.17. El sensor de lluvia es

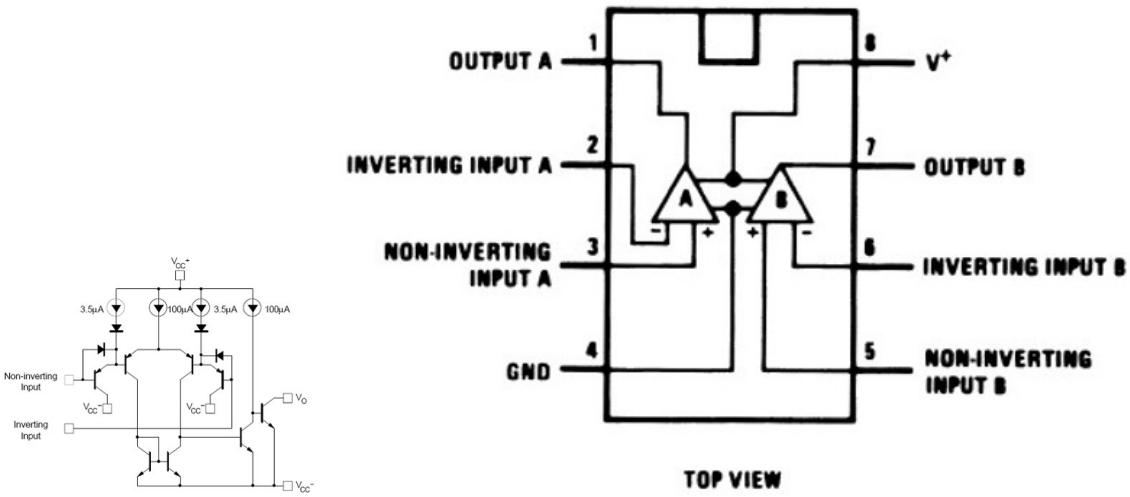
una herramienta que permite de forma sencilla detectar si está lloviendo. El sensor detecta el agua que conecta los circuitos en las pistas impresas del tablero de detección. Dicho tablero del sensor actúa como una resistencia variable que cambiará de 100 $k\Omega$ ohmios cuando está mojado a 2 $M\Omega$ cuando está seco. Por ende, cuanto más húmedo sea el tablero, mayor será la corriente que circulará. A tensión de trabajo es de 5 V y cuenta con dos salidas, una digital y una analógica que indica la salida de tensión. Cuenta con un potenciómetro para ajustar la sensibilidad y utiliza un comparador LM393 de amplio voltaje El circuito compara los voltajes de sus dos entradas y da como salida el mayor de esos voltajes de entrada. El integrado LM393[32] está conformado por dos comparadores. En la Figura 3.17 se observa el diagrama de pines del integrado y un diagrama esquemático del mismo



(a) Sensor.

(b) Diagrama esquemático.

Figura 3.17: Sensor de lluvia.



(a) Diagrama de pines.

(b) Diagrama esquemático.

Figura 3.18: Integrado LM393.

3.1.18. Cámara térmica

Se eligió cámara térmica Lepton de 80x60 pixeles dado su bajo costo frente a otras soluciones Lepton y que cumplía con la necesidad de este trabajo. Otros fabricantes ofrecen soluciones de cámaras térmicas con displays y keyboards que no eran compatibles con este proyecto y que a su vez son de mayor costo. Lograr mejoras en la resolución implica costos elevados. Por otra parte esta solución es compatible con la SBC elegida debido a que utiliza protocolo SPI para el envío de las imágenes y protocolo I^2C para el control de la misma. El shield y un diagrama de sus dimensiones se puede observar en la Figura 4.10.

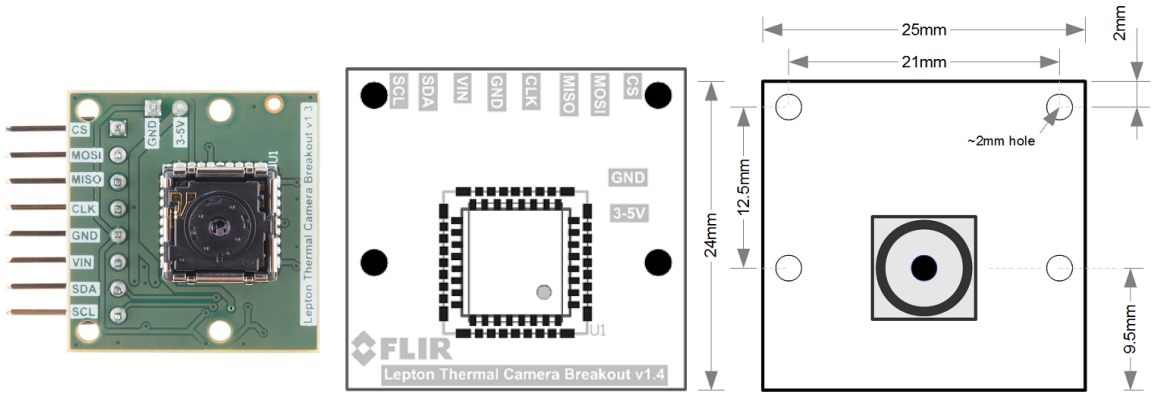


Figura 3.19: Shield Flir Lepton.

Se optó por una cámara Flir Lepton de 80x60 píxeles[33] y 50° de apertura, debido a su tamaño pequeño y bajo peso es ideal para ir montada en el Dron. Además, proporciona una definición acorde al alcance del proyecto. Por otro lado, es la única que se puede adquirir únicamente con un kit de desarrollo sin la necesidad de comprar un sensor profesional (el cual incluye un display), disminuyendo significativamente su costo.

En la Tabla 3.6 se pueden apreciar diferentes modelos de cámaras térmicas y sus principales características como los pixeles, FoV[4] y el precio.

3.2. Ensamblado

En esta sección se describe el ensamblado de todo el sistema discriminándolo en 2 partes, el ensamblado del Dron y el ensamblado de la GCS.

3.2.1. Ensamblado del Dron

Se detalla a continuación el sistema de conexión del controlador Pixhawk con el resto de los componentes. Pixhawk incluye acelerómetro, giroscopio, magnetómetro

[4]Field of View hace referencia al ángulo de apertura de la cámara. Se presentan de a pares y normalmente son Horizontal y Diagonal o Horizontal y Vertical. Basta con 2 de estos valores para definir la apertura de la cámara.

	Pixeles	**FoV**	**Rango (°C)**	**Sensibilidad (mK)**	**Dimensiones (mm)**	**Peso (g)**	**Precio (USD)**
Flir Lepton	80 x 60	51.0 ° x 63.5 °	-66 / +90	<50	11 x 12 x 6	0.55	215
Flir AX8	80 x 60	48.0 ° x 37.0 °	-10 / +150	<100	54 x 25 x 79	25	995
Flir One	80 x 60	50.0 ° x 63.5 °	-20 / +120	<100	72 x 26 x 18	78	250
Seek Thermal	206 x 156	36.0 ° x 36.0 °	-40 / +330	<100	45 x 20 x 15	14	250
Flir TAU2	640 x 512	90.0 ° x 69.0 °	-25 / +100	<60	45 x 45 x 30	72	>2000
Zenmuse XT	640 x 512	90.0 ° x 69.0 °	-10 / +40	<50	103 x 74 x 102	270	>7000

Tabla 3.6: Características principales de diferentes modelos de cámaras térmicas

y barómetro. La alimentación de Pixhawk se realiza desde la batería por medio del BEC, el cual permite entregar 12 V al distribuidor y 5 V a Pixhawk. Al distribuidor se conectan la alimentación de los ESC para luego llegar a los motores. Cada ESC alimenta y controla un motor y a su vez los ESC son controlados por las salidas principales de PWM de Pixhawk.

A Pixhawk se conecta el transceptor 3DR en uno de sus puertos TELEM para la comunicación inalámbrica con la GCS. El puerto TELEM2 se utiliza para la comunicación serial con la placa on-board. El receptor de GNSS se conecta al puerto GPS y su magnetómetro al I2C. El puerto SERIAL 4/5 implementa dos comunicaciones UART, en el 4 está la consola del sistema operativo NuttX y se utilizó a lo largo del proyecto para configuración y debugging. El gimbal es controlado y alimentado por los PWM auxiliares. Todos estos puertos se pueden observar en la figura 3.21, Tanto la cámara como los sensores de proximidad se conectan a la placa on-board.

En la figura 3.20 se puede apreciar el diagrama esquemático del conexionado eléctrico de los componentes del Dron.

3.2.2. Ensamblado de la GCS

La GCS se compone de una placa Arduino DUE conectada al transceptor 3DR para la comunicación con el Dron y a través de un puerto UART con el shield 3G/GPRS para tener conexión a internet a través de la red celular permitiendo el acceso remoto desde cualquier dispositivo, por ejemplo un celular, tablet, notebook, etc. El sensor DHT22 se comunica por un puerto digital, mientras que el sensor de lluvia se comunica por un pin digital y un pin analógico. Ambos sensores son alimentados desde la salida de voltaje de la Arduino DUE.

En la figura 3.22 se puede apreciar el diagrama esquemático del conexionado eléctrico de los componentes del Dron.

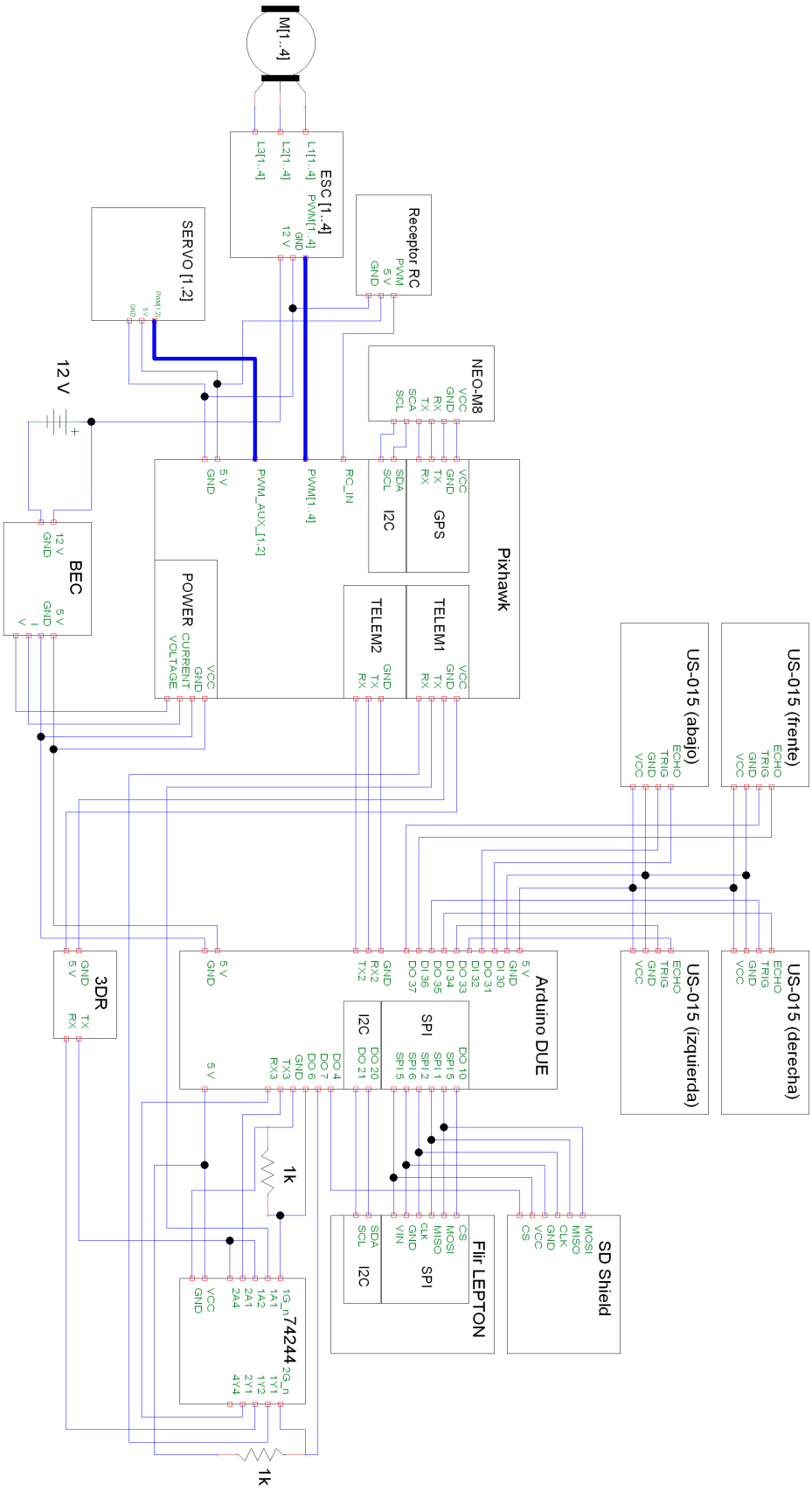
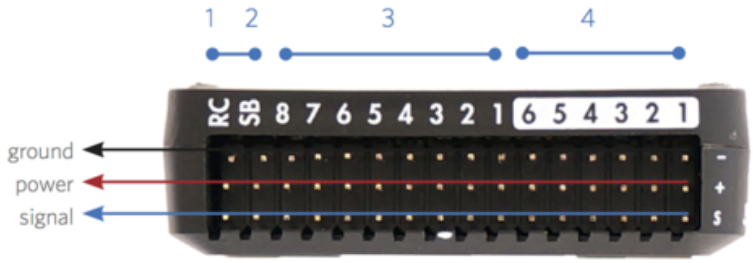


Figura 3.20: Diagrama esquemático del conexionado eléctrico de los componentes del Dron.

1 Spektrum DSM receiver
2 Telemetry (radio telemetry)
3 Telemetry (on-screen display)
4 USB
5 SPI (serial peripheral interface) bus
6 Power module
7 Safety switch button
8 Buzzer
9 Serial
10 GPS module
11 CAN (controller area network) bus
12 I²C splitter or compass module
13 Analog to digital converter 6.6 V
14 Analog to digital converter 3.3 V
15 LED indicator

1 Input/output reset button
2 SD card
3 Flight management reset button
4 Micro-USB port



1 Radio control receiver input
2 S.Bus output
3 Main outputs
4 Auxiliary outputs

Figura 3.21: Puertos de Pixhawk.

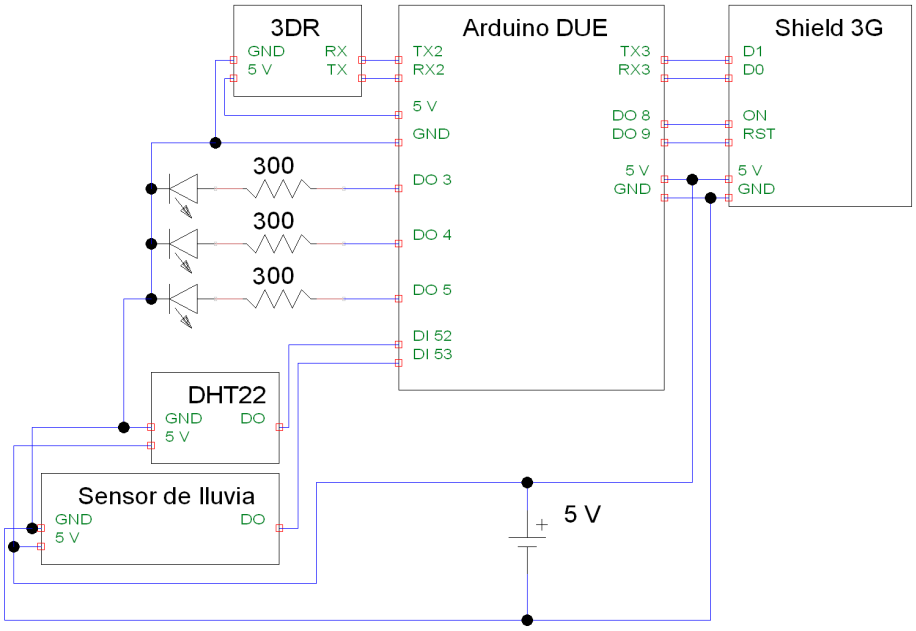


Figura 3.22: Diagrama esquemático del conexionado eléctrico de los componentes de la GCS.

3.2.3. Sistema armado

En la figura 3.23 se puede apreciar el sistema ensablado. Se utilizó una caja estanca para la GCS de manera de proteger tanto sus componentes como el conexionado entre los mismos y se agregaron LEDs de estado (alimentación y actividad en los puertos de comunicación).

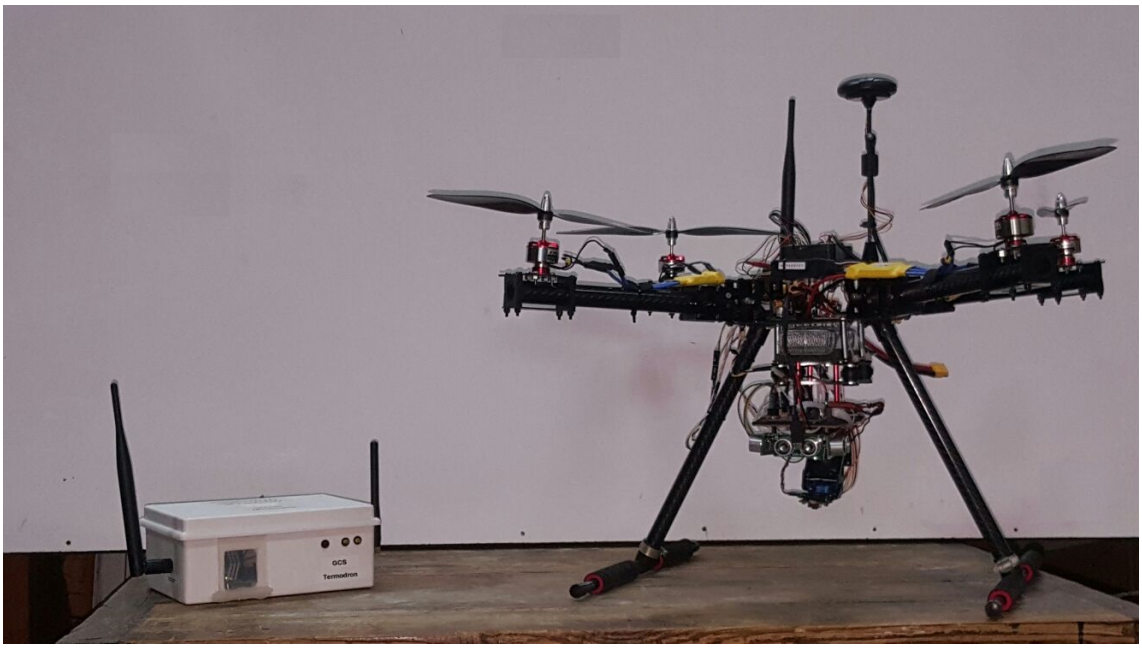


Figura 3.23: Sistema armado. GCS y Dron.

CAPÍTULO 4

IMPLEMENTACIÓN DE LOS MÓDULOS FUNCIONALES DEL DRON

4.1. Módulo on-board

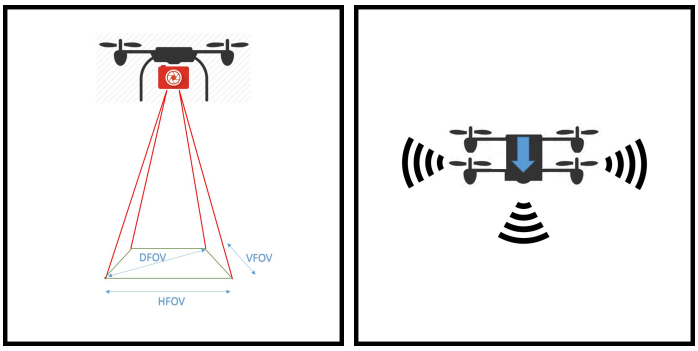
La función principal del módulo on-board es ser interfaz entre los sensores, cámara térmica y shield SD con la controladora de vuelo y también con la GCS a través de la controladora Pixhawk. En la Figura 4.1 se observan las funciónes principales del módulo on-board, estas son:

- Tomar fotografías térmicas e identificar cuerpos que se encuentran dentro del rango de temperaturas objetivo
- Evaluar los sensores de distancia para evitar posibles obstáculos que se puedan encontrar
- Complementar a la controladora de vuelo a la hora del aterrizaje, relevando el estado del sensor de distancia que está mirando hacia el suelo que le da mayor precisión a la medida de la altitud del Dron y también la cámara térmica que le da precisión para encontrar el punto exacto donde debe aterrizar
- Almacenar toda la información de estado de componentes, comunicaciones e imágenes en memoria y a su vez, poder enviarla a la GCS si es solicitada.

4.1.1. Hardware on-board

Para resolver las conexiones de sensores y cámara Flir Lepton con la controladora de vuelo se optó por una interfaz Arduino DUE que pueda recibir todas las señales a sensar y a su vez realizar el control de estos. La Arduino DUE cuenta pines de entrada/salida digitales suficientes para conectar los sensores de distancia. Además, tiene puertos SPI y Wire (I^2C) permitiéndole conectar la Flir Lepton, controlarla y a su vez conectar el shield SD.

El shield SD se utiliza con una memoria SD para el almacenamiento de los datos que se recaban en las diferentes misiones y está conectado vía SPI a la placa on-board. La placa on-board tiene un solo puerto SPI en el cual tiene conectada además del shield SD tiene la Flir Lepton. Para el correcto funcionamiento de la interfaz



(a) Tomar imágenes térmicas

(b) Evaluar sensores de distancia

(c) Asistencia al aterrizar

(d) Almacenamiento de infrmacón

Figura 4.1: Funciones del módulo on-board.

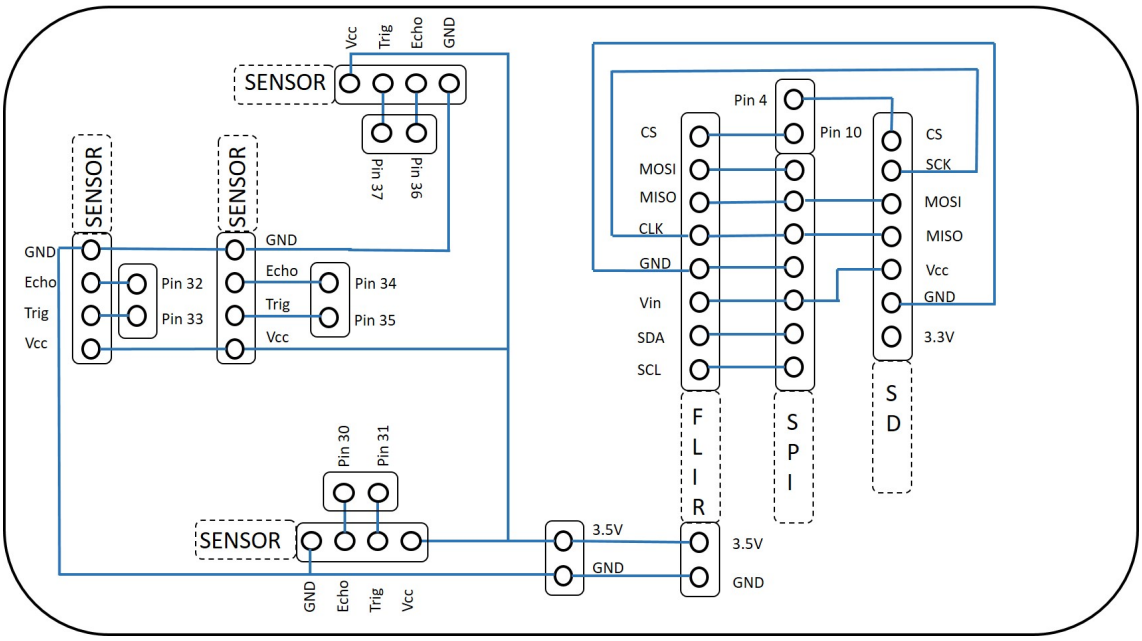


Figura 4.2: Diagrama de conexiones del módulo on-board.

SPI, ambos dispositivos deben tener un pin de Chip Select (CS), y la placa on-board debe ser capaz de establecer comunicación con el dispositivo que corresponda. El shield Flir Lepton cuenta con el pin de CS. Sin embargo, el shield SD tuvo que ser modificado conectando el pin CS del integrado LVC125A, que estaba aislado inicialmente, al pin asignado para el CS del shield SD en la placa on-board.

El diagrama de conexionado puede observarse en la Figura 4.2.

Se diseñó un circuito para permitir la conexión de todos los componentes y que a su vez permita la extracción fácil de cualquiera de ellos en caso de que fuera necesario. Además, este circuito fue realizado en una placa para PCB de un tamaño mayor al necesario para contener el circuito diseñado. Esto fue para que esa placa funcionará de soporte de los sensores, el shield Flir Lepton, el shield SD y la propia on-board. Esto proporciona un plano de trabajo rígido y a la vez liviano.

Finalmente se requería que el plano de trabajo donde están los sensores y la cámara se mantuviera siempre paralelo al plano de la superficie terrestre y así poder estimar fácilmente la posición de los cuerpos calientes detectados. Por esta razón se decidió utilizar un gimbal con dos servos que logran el cometido anterior y están conectados a dos salidas PWM del controlador Pixhawk. Estas salidas están diseñadas de forma de que los servos que se conecten compensen los ángulos de Roll y Pitch.

4.1.2. Software on-board

Al iniciar la placa on-board, están inicializadas todas las comunicaciones que utilizará, estas son: comunicación SPI con Flir Lepton y shield SD, comunicación Wire (I^2C) con Flir Lepton, comunicación Serial con Pixhawk e inicializa los pines digitales para sensar los datos de los sensores de distancia. Además se agendan las tareas que correrán mientras la placa esté encendida. Estas tareas son tres: “heartbeat”, “update” y “states”. Estas agrupan variados procesamientos. La primera (“heart-

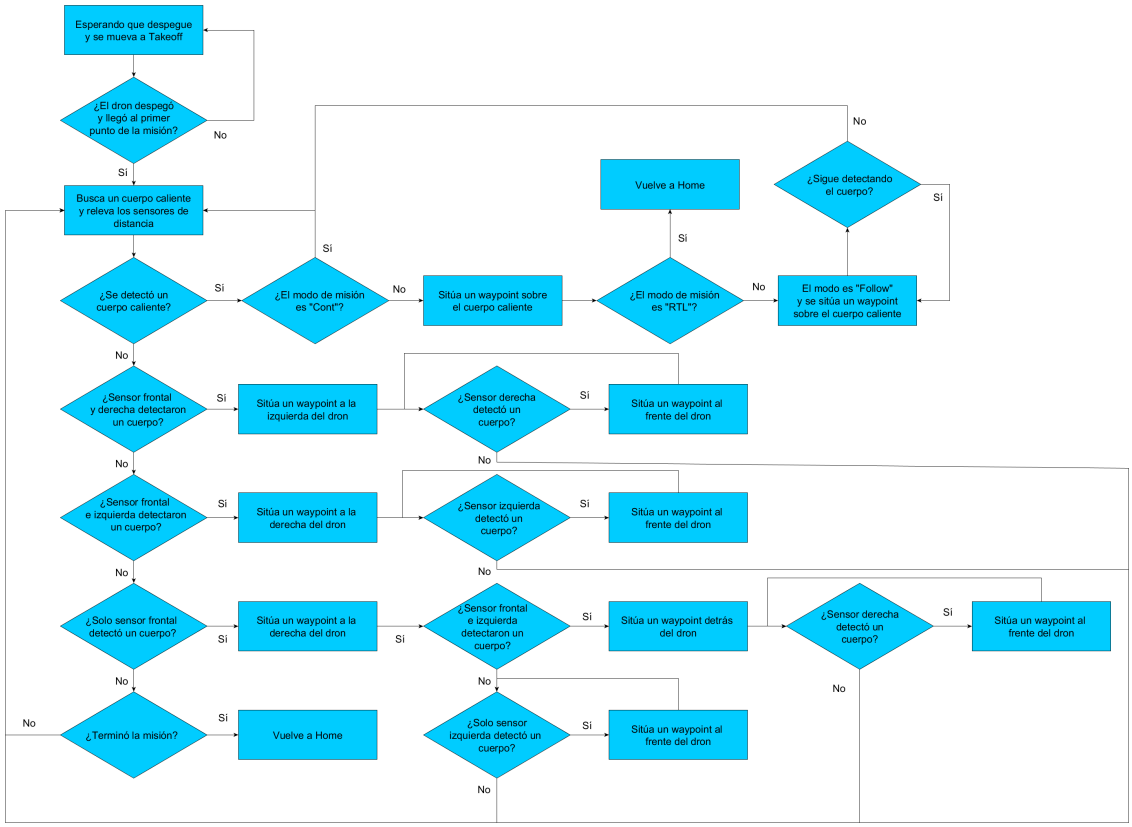


Figura 4.3: Diagrama de flujo del software implementado en el módulo on-board.

beat") simplemente envía un mensaje a la controladora Pixhawk para que sepa que está conectada y funcionando, esto permite que la comunicación se mantenga abierta. La segunda tarea ("update") agrupa funciones de actualización de variables de estado del Dron (consultando a Pixhawk) así como el status de sensores y demás dispositivos que están directamente conectados a la placa on-board. La última tarea ("states") se encarga del manejo de los sensores, consultas, re-cálculos de trayectoria, manejo de imágenes, almacenamiento de datos, etc. Esta última tarea está diseñada a partir de una máquina de estados, la cual puede observarse en la Figura 4.3. Cada vez que la tarea "states" es llamada el programa revisa el estado en el que se encuentra y ejecuta las funciones correspondientes.

El estado inicial ("WaitTakeoff") comienza luego de todas las inicializaciones y mientras el programa se encuentre en este estado simplemente se chequea si Pixhawk llegó al "takeoff" (primer waypoint de la misión) y en caso afirmativo se solicita toda la misión a Pixhawk y se pasa al estado "Checking".

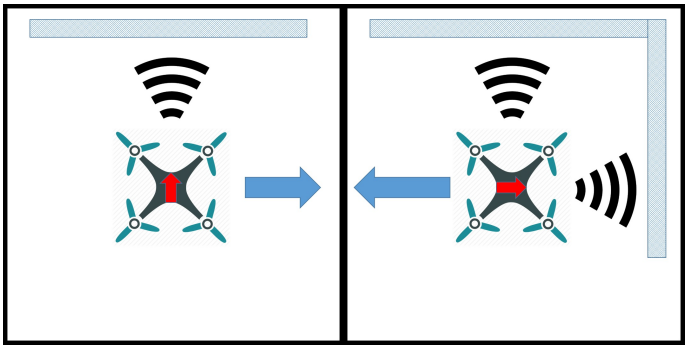
El estado "Checking" comienza tomando una fotografía de la Flir Lepton, luego esta es analizada en busca de cuerpos calientes, se almacena en la tarjeta SD en formato BMP, se envía a la base y después se relevan los sensores de distancia. Por último, se decide el cambio de estado en función de los valores obtenidas. Si se encontró un cuerpo caliente se decide como proseguir en función del tipo de misión, esto es, si la misión se trata de encontrar un cuerpo caliente y seguirlo o retornar a la base, se debe calcular un waypoint sobre el cuerpo, enviarlo a Pixhawk y pasar al estado "BodyDetected", si en cambio la misión se trata simplemente de detectar y relevar las temperaturas en un área, se debe seguir en este estado. Si se detectó un obstáculo con alguno de los sensores de distancia, se debe agregar un waypoint que

permita evitar ese obstáculo y pasar al estado “Sidestep”. Esto implica que, si se detecta un obstáculo delante, se debe agregar un waypoint a la derecha, si el sensor de la derecha no está activado, en caso contrario se debe agregar un waypoint a la izquierda.

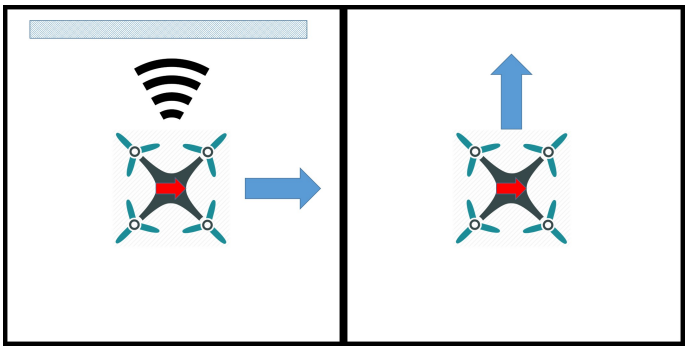
El estado “BodyFollow” envía a Pixhawk el waypoint correspondiente a la posición de la GCS si la misión es encontrar y retornar a la GCS y pasa al estado “Landing”, si en cambio la misión es de seguimiento se debe actualizar la posición del cuerpo detectado y quedarse en este mismo estado, pero si por alguna razón no se detecta ningún cuerpo caliente en la imagen se debe enviar a Pixhawk la misión inicial y pasar al estado “Checking”.

El estado “Landing” es alcanzado una vez que el Dron luego de ser enviado a la base se posiciona sobre esta para implementar el aterrizaje. Este estado es el encargado de aterrizar de Dron, para esto en cada pasaje por este estado se disminuye la altura del dron a la mitad de la altura detectada con el sensor que apunta hacia abajo y si la altura es mayor a la detectada se disminuye a la mitad de la estimada por Pixhawk.

El estado “Sidestep” es el encargado de evitar una posible colisión del Dron y luego retornarlo a la misión. A este estado se llega cuando ya fue detectado un obstáculo y se agregó un waypoint evitándolo y el Dron ya giró completamente hacia el lado donde se agregó el waypoint. Si se llegó por haber detectado solamente algo con el sensor frontal, hay tres posibles casos: primero que nuevamente se detecte algo delante y algo a la izquierda por lo que se deberá indicar al Dron un waypoint que lo lleve hacia la izquierda de su posición inicial, que se detecte solo algo con el sensor izquierdo teniendo que pasarle un waypoint para que siga en la dirección en la que se estaba moviendo y si no se detecta nada se debe retomar la misión. Esto se puede observar en la Figura 4.4.



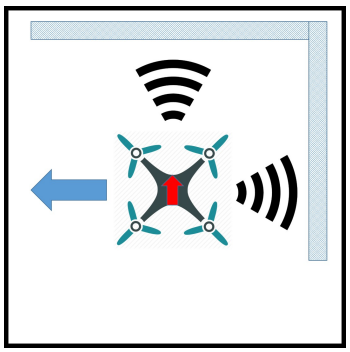
(a) Primera detección (b) Detecta delante



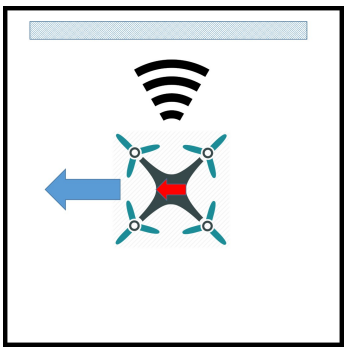
(c) Sigue detectando (d) Deja de detectar

Figura 4.4: Detección solo delante, primero se mueve a la derecha

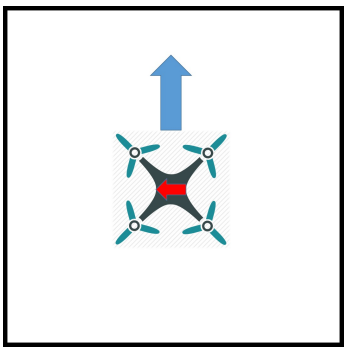
Si se llegó por haber detectado con el sensor frontal y el izquierdo, hay dos posibles casos: primero que se detecte con el sensor izquierdo por lo que se deberá indicar al Dron un waypoint para que siga en la dirección en la que se estaba moviendo y si no se detecta nada se debe retomar la misión. Esto se puede observar en la Figura 4.5.



(a) Primer detección



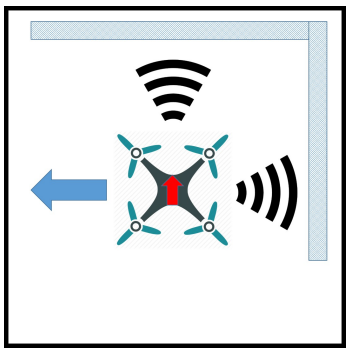
(b) Sigue detectando



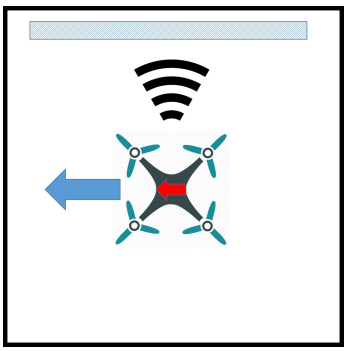
(c) Deja de detectar

Figura 4.5: Detección delante y a la izquierda

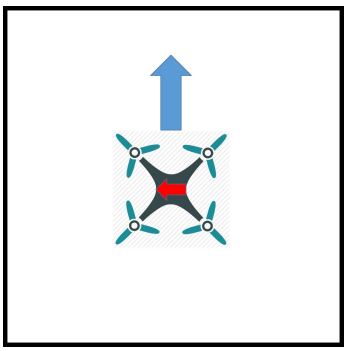
Si se llegó por haber detectado con el sensor frontal y el derecho, hay dos posibles casos: primero que se detecte con el sensor derecho por lo que se deberá indicar al Dron un waypoint para que siga en la dirección en la que se estaba moviendo y si no se detecta nada se debe retomar la misión. Esto se puede observar en la Figura 4.6.



(a) Primer detección



(b) Sigue detectando



(c) Deja de detectar

Figura 4.6: Detección delante y a la derecha

En la Figura 4.7 se indican las referencias a las figuras anteriores.

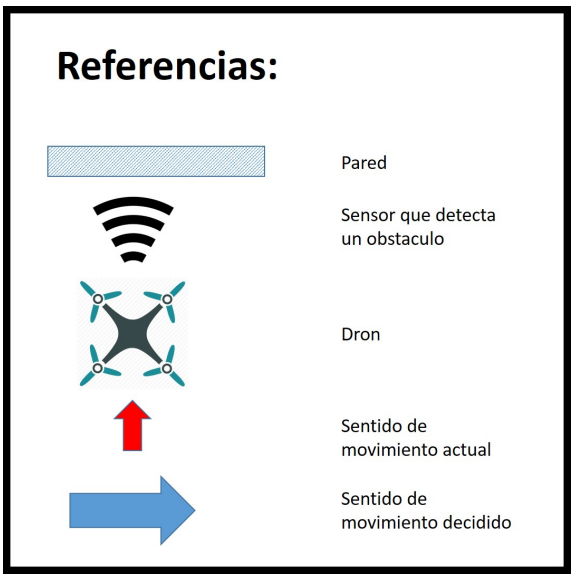


Figura 4.7: Referencias

4.2. Detección térmica

4.2.1. Fundamentos de radiación térmica

La radiación térmica es un tipo de radiación electromagnética. Este tipo de radiaciones cuentan con características intrínsecas como lo son: velocidad de propagación, intensidad, frecuencia de pulsación, fase, polarización, coherencia, divergencia.[34]

La velocidad de propagación en el vacío es $c \approx 3.00 \times 10^8\ m/s$ y es siempre menor cuando atraviesa cualquier otro medio. La longitud de onda se define a partir de la velocidad (v) y la frecuencia de una radiación (f) como $\lambda = v/f$. La frecuencia se mantiene constante al pasar de un medio a otro, pero cambia de longitud de onda debido al cambio en la velocidad de propagación.

Las radiaciones que dan lugar a efectos térmicos en su interacción con la materia están en la banda de $10^{-7} < \lambda < 10^{-4}\ m$.

La materia en estado condensado emite un espectro de radiación continuo debido al movimiento térmico de las cargas contenidas en su interior. La frecuencia de onda emitida por radiación térmica es una densidad de probabilidad que depende solo de la temperatura. Los cuerpos negros emiten radiación térmica con el mismo espectro correspondiente a su temperatura, independientemente de los detalles de su composición. Para este caso, la función de densidad de probabilidad de la frecuencia de onda emitida está dada por la ley de radiación térmica de Planck. La ley de desplazamiento de Wien relaciona la temperatura de un cuerpo con la frecuencia a la que se da el máximo de radiación emitida y la ley de Stefan-Boltzmann establece el total de energía emitida por unidad de tiempo y superficie emisora.

Max Planck, basándose en su postulación de que la emisión de energía electromagnética no era continua, sino que se realizaba en forma de fotones, obtuvo la siguiente expresión para la densidad de potencia irradiada (E) por un cuerpo negro por unidad de área en función de la longitud de onda (λ) y la temperatura (T) como:

$$E(\lambda, T) = \frac{8\pi hc^2}{\lambda^5} \frac{1}{e^{\frac{hc}{k\lambda T}} - 1}$$

siendo la constante de Planck $h \approx 6.625 \times 10^{-34}\ Js$ y la constante de Boltzmann $k \approx 1.381 \times 10^{-23}\ J/K$.

En la Figura 4.8 se observa dicha distribución para distintas temperaturas.

La distribución espectral de Planck presenta un máximo para una longitud de onda λ_{emax}, tal que la temperatura absoluta es:

$$T = \frac{2.9 \times 10^{-3}}{\lambda_{emax}}$$

lo que se conoce como la ley de desplazamiento de Wien.
La integral de la distribución espectral sobre todo el espectro de radiación (e), es la llamada ley de Stefan-Boltzman: $e = \sigma T^4$, con $\sigma = -5.67 \times 10^{-8}\ W/(m^2K^4)$

La intensidad de emisión de un elemento de superficie dA varía según el ángulo con respecto a la normal como: $e_\beta = e_0 cos\alpha$ a lo que se le denomina ley de Lambert.

Como los cuerpos reales no son cuerpos negros, se definen tres características que permiten realizar el modelado. Estas características son: la absortividad, la reflectividad y la emisividad. Se define la absortividad α como el cociente de la energía absorbida por el material sobre la energía total incidente en él. Por otro lado, la reflectividad ρ se define como el cociente de la potencia reflejada sobre la

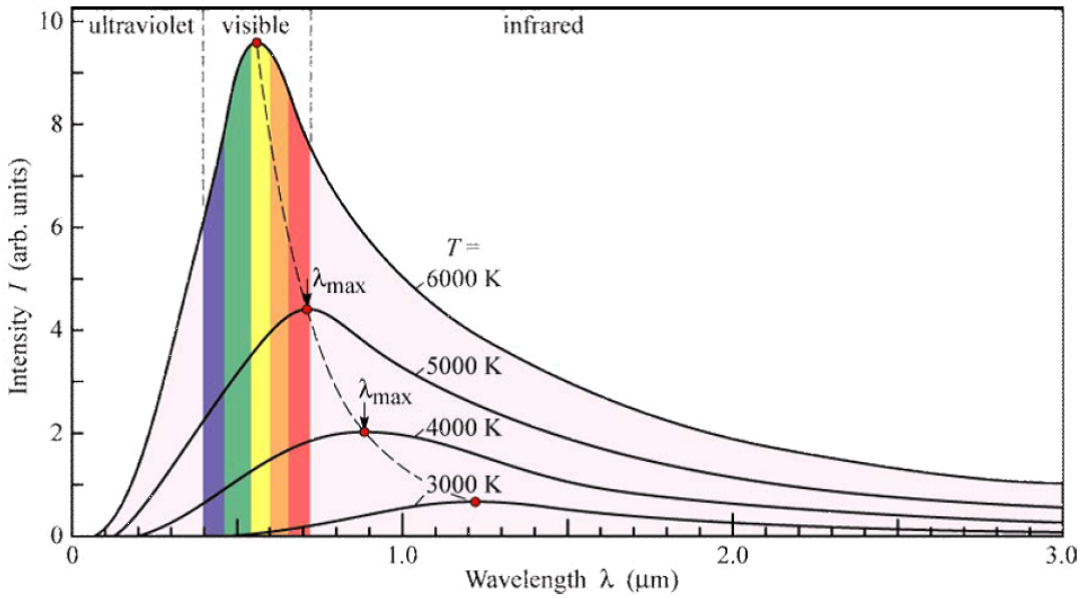


Figura 4.8: Distribución espectral de energías de un cuerpo negro. Imagen obtenida de [35].

potencia total incidente. Por conservación de la energía $\rho = 1–\alpha$. Y por último se tiene la emisividad ϵ, que es el cociente entre la potencia emitida por un cuerpo real y la emitida por un cuerpo negro a la misma temperatura en la misma longitud de onda.

Debido a que poder calcular la temperatura de un objeto a partir de la detección de su energía irradiada implica un cálculo complejo y más aún cuando se intenta realizar en tiempo real es que las cámaras térmicas tienen un elevado costo.

4.2.2. Hardware

Como se mencionó en el capítulo anterior el equipo elegido para realizar la detección térmica de objetos que se encuentran a una temperatura diferente a la de su entorno fue el Flir Lepton. Este es un shield con una cámara LWIR (Longwave Infrared), la cual es la más pequeña y liviana en el mercado y es diez veces menos costosa que las cámaras térmicas tradicionales.

El Flir Lepton contiene una lente avanzada fabricada en forma de oblea, junto con microbolómetro FPA (Focal Plane Array) de 80x60 pixeles activos y procesamiento de imagen térmico avanzado, lo cual le da mayor sensibilidad que las matrices comunes de termopila.

Captura la radiación infrarroja en la banda de longitud de onda de 8 a 14 micras (respuesta nominal) y emite una imagen térmica uniforme, alcanzando una sensibilidad térmica de 50 mK.

Las dimensiones del shield son 10.6 x 11.7 x 5.9 mm y tiene un peso de 0.55 g.

Tiene un lente f/1.1 de doblez de silicio con un FOV (Field of View) horizontal de 51º y uno diagonal de 63.5º, además de un DOF (Depth of Field) de 10 cm hasta infinito.

La cámara cuenta con funciones de procesamiento de imágenes térmicas digitales integradas, las cuales incluyen la compensación térmica automática del entorno, filtros de ruido, corrección de no uniformidad y control de ganancia.

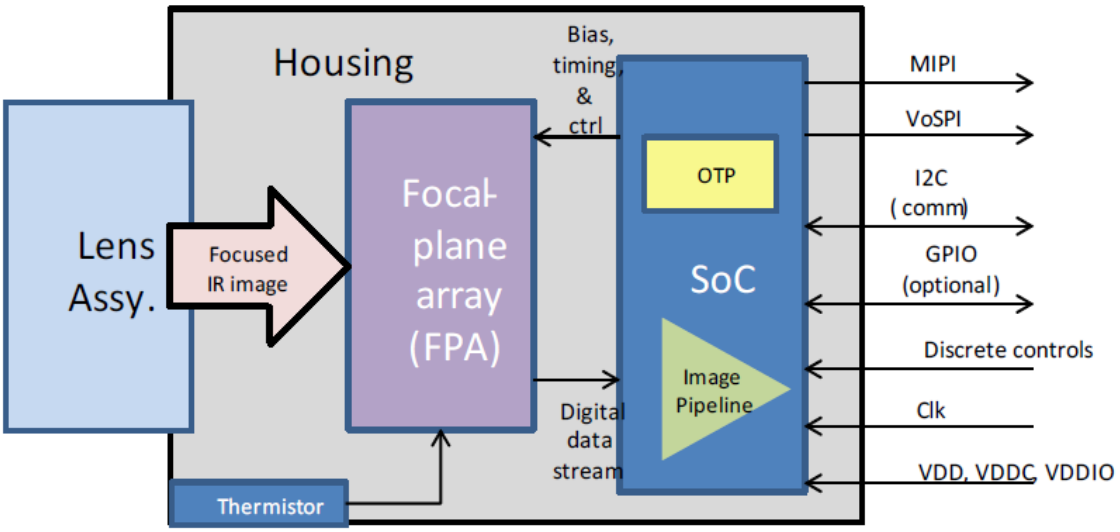


Figura 4.9: Diagrama de funcionamiento del shield Flir Lepton.

Utiliza la interfaz TWI (Two-Wire Interface, la cual es igual a I^2C) para el comando y control y SPI (Serial Peripheral Interface) como interfaz de video.

El consumo eléctrico es menor a 150 mW nominales, siendo siempre menor a 160 mW y cuenta con un modo standby que consume 4 mW.

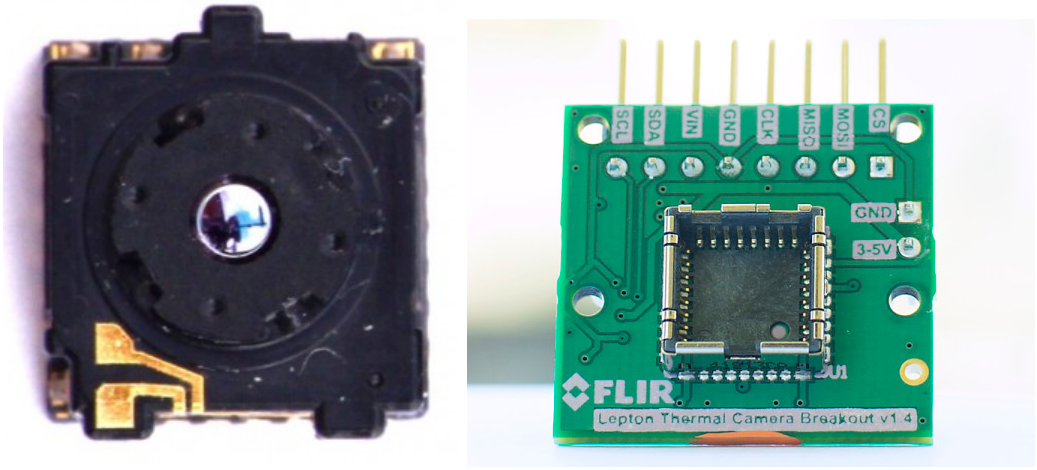
La Figura 4.9 describe el funcionamiento del equipo. El lente enfoca la radiación infrarroja de la escena sobre una matriz 80x60 de detectores térmicos con paso de 17 micras. Cada elemento detector es un microbolómetro de óxido de vanadio (VOx) cuya temperatura fluctúa en respuesta al flujo incidente. El cambio de temperatura provoca un cambio proporcional en la resistencia de cada microbolómetro. El VOx proporciona un alto coeficiente de resistencia de temperatura (TCR) y un bajo ruido, lo que resulta en una excelente sensibilidad térmica y uniformidad estable. La matriz de microbolómetro se hace crecer monolíticamente sobre un circuito integrado de lectura (ROIC) para comprender la matriz completa de plano focal (FPA). Una vez por fotograma, el ROIC detecta la resistencia de cada detector aplicando un voltaje de polarización e integrando la corriente resultante durante un período de tiempo finito denominado período de integración.

El flujo en serie de la FPA es recibido por un sistema en un dispositivo de chip (SoC), que proporciona el procesamiento de señal y el formato de salida.

La cámara tiene una interfaz de 32 pines por lo que se decidió conectarla a una placa de evaluación (BREAKOUT v1.4 de Pure Engineering), compatible con los 32 pines de la cámara Flir Lepton, el cual permite el acceso Two-Wire y el SPI de la cámara y provee de un reloj de 25 MHz de referencia, disminuyendo los pines de la interfaz a solo 10. Esta placa tiene un tamaño de 25 mm x 24 mm y un peso de 5.2 g. Soporta un voltaje de entrada de 3 V a 5.5 V, haciendo a la cámara compatible con más sistemas de desarrollo de software.

En la Figura 4.10 se puede apreciar la cámara Flir Lepton a la izquierda y a la derecha la placa BREAKOUT v1.4.

La SBC Arduino DUE on-board será la encargada del procesamiento y análisis de la imagen térmica. Se conectará el BREAKOUT v1.4 de Pure Engineering a la Arduino DUE según Figura 4.11.



(a) Cámara Flir Lepton (b) Pure Engineering BREAKOUT v1.4

Figura 4.10: Componentes del shield térmico

Breakout Lepton Flir	Arduino Due
CS	PIN 10
MOSI	MOSI (SPI 5)
MISO	MISO (SPI 1)
CLK	CLK (SPI 2)
GND	GND (SPI 6)
VIN	5V (SPI 4)
SDA	PIN 20 (SDA)
SCL	PIN 21 (SCL)

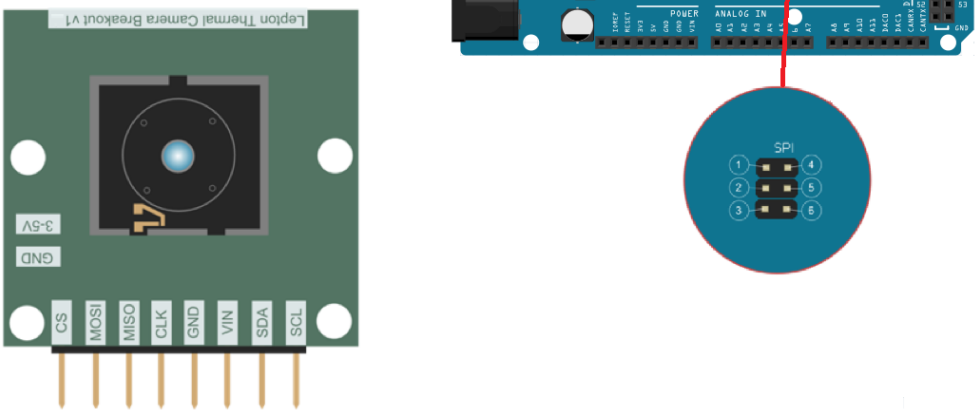


Figura 4.11: Conexionado Arduino DUE on-board y Breakout V1.4.

La cámara Lepton soporta una interfaz de comando y control alojada en TWI (similar a I^2C). La interfaz consiste en un pequeño número de registros a través de los cuales un Master emite comandos y recupera respuestas de la cámara. Los registros de Lepton son de 16 bits. El ID de la cámara es 0x2A. En la Figura 4.12 se pueden apreciar las direcciones de los diferentes registros.

Registro Command

El registro Command de Lepton es un registro de 16 bits situado en la dirección 0x0004. Este registro se utiliza para emitir una orden a la cámara. Al escribir un valor en este registro se inicia el procesamiento de comandos. Es importante asegurarse de que el bit de comando BUSY en el registro Status de Lepton indica que la cámara está lista para aceptar un nuevo comando antes de iniciarlo, de lo contrario la comunicación de la Cámara puede verse comprometida haciendo necesario su reinicio.

La palabra del registro Command está compuesta de 4 campos que se muestran en la Figura 4.13. Esta palabra indica a la cámara qué acción tomar, para formarla el ID de módulo se agrega con la base de ID de comando y con el tipo de comando y de ser necesario con un valor de bit de protección. Los módulos AGC, VID y SYS no requieren que se configure un bit de lo que hace que el valor del bit de protección sea 0x0000. Dado todo esto el ID de Comando se sintetiza de la siguiente manera: ID de módulo + ID de comando Base + Tipo + Valor de bit de protección = ID de comando.

El Lepton ID de Módulo designa el módulo de cámara con el que se operará. Los módulos de Cámara encapsulan atributos y métodos de un subsistema de cámara. Lepton define 3 sub-sistemas y su módulo asociado:

MÓDULO AGC: Este módulo proporciona el comando y control de la operación de control automático de ganancia (AGC) de salida de vídeo. Los datos de vídeo de la cámara pueden procesarse para proporcionar un contraste de escena óptimo utilizando Ecualización de Histograma HEQ o estiramiento de Histograma Lineal.

MÓDULO SYS: Este módulo proporciona información y estado del sistema de cámara.

MÓDULO VID: Este módulo proporciona el comando y control de los datos de vídeo.

Para cada uno de los módulos Lepton, un ID de Comando único identifica un elemento del módulo, ya sea un atributo o una acción. Cada módulo de cámara expone hasta 64 ID de comando asignados a atributos y / o métodos de ese módulo.

Un tipo de comando especifica lo que hace el comando. Estos son “get” (lectura de datos), “set” (escritura de datos) y “run” (ejecución de una rutina).

Una transmisión típica de comandos requiere una secuencia de:

- Sondear el registro de estado hasta que la cámara esté lista para un nuevo comando (bit BUSY en 0).

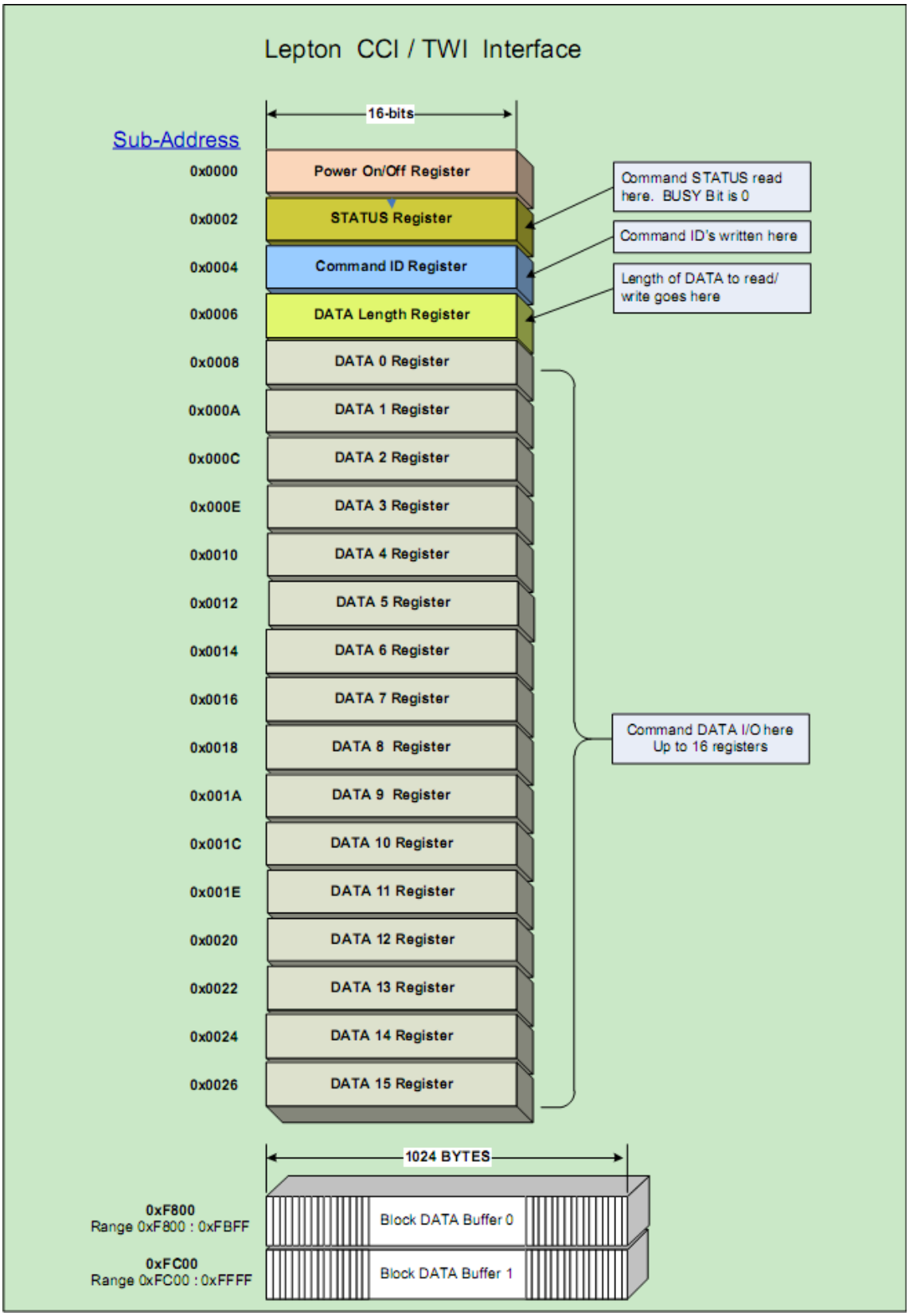


Figura 4.12: Registros de Flir Lepton

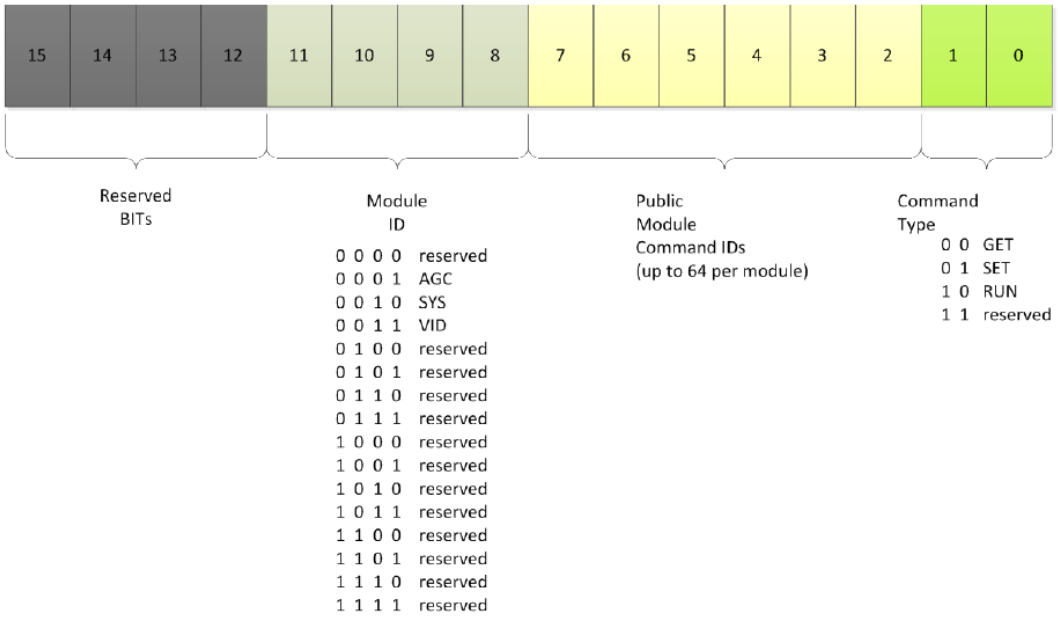


Figura 4.13: Composición de la palabra de comando.

- Escribir datos para enviar a la cámara (si es necesario) en los registros Data o en los bloques Data Buffer.
- Escribir el número de palabras de datos escritas (palabras de datos de 16 bits) en el registro Data Length.
- Escribir el ID de comando deseado en el registro Command.
- Sondear el registro Status para determinar cuándo se ha completado el comando (bit BUSY en 0).
- Leer el código de éxito del registro Status.
- Recupere todas las respuestas requeridas de los registros de datos o bloquee la memoria intermedia de datos.

Registro Status

El registro Status, ubicado en la dirección de registro 0x0002 se utiliza para comunicar el estado del comando y el estado de arranque de la cámara. Cada vez que un host emite un comando a la cámara escribiendo en el registro de comandos, la cámara automáticamente establece en 1 el bit de comando BUSY (bit 0) en el registro de estado. Cuando se completa el comando, el código de respuesta se escribe en los 8 bits superiores del registro de estado. Luego la cámara establece en 0 el bit de comando BUSY. Los bits que la componen se pueden observar en la Figura 4.14.

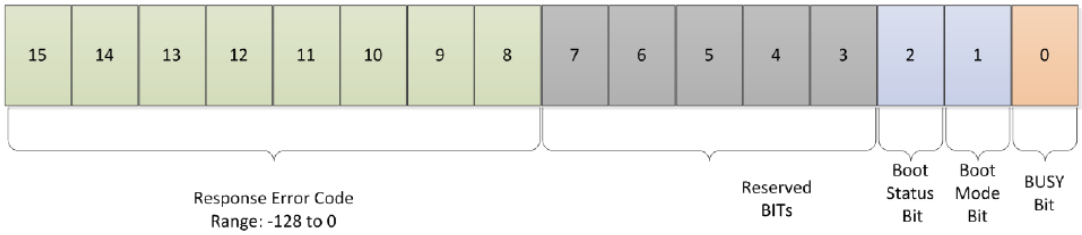


Figura 4.14: Composición de la palabra de status.

Registro Data Length

El registro DATA Length está ubicado en la dirección de registro 0x0006, se utiliza para especificar el número de palabras (registros) de 16 bits que se están transfiriendo.

Registro DATA

Los registros DATA [0-15], ubicados en las direcciones de registro 0x0008 a 0x0026, se utilizan para transferir datos desde y hacia la cámara. Cada registro tiene 16 bits de ancho y hay 16 registros independientes. El modo de incremento automático se utiliza cuando se leen y escriben estos registros. Dado que la interfaz TWI transfiere DATA en palabras de 16 bits, el orden de los bytes se vuelve importante. Esta interfaz sólo admite MSBF (bit más significativo primero). Dentro de cada palabra de 16 bits, los bits 7-0 contienen el MSB (bit más significativo) y los bits 15: 8 contienen el LSB (bit menos significativo).

Cuando se transmiten DATOS que son mayores que una sola palabra (16 bits), los DATOS más grandes se dividen en múltiples palabras de 16 bits. Cada palabra se coloca entonces en múltiples registros DATA con LWSF (palabra menos significativa primero) en el registro DATA inferior.

Para transferencias que exceden los 16 registros de DATOS, la cámara proporciona un búfer de 1 kB. Se utiliza para transferir bloques más grandes de DATOS, tales como tablas de búsqueda de color definidas por el usuario. Estos búferes también se abordan como palabras de 16 bits, por lo que la longitud total de un solo búfer es de 512 palabras. El acceso se trata como una transferencia de múltiples palabras, así como con las palabras menos significativas en las direcciones de memoria inferior.

4.2.3. Software desarrollado

Para comenzar se inician las comunicaciones Wire y SPI de la placa on-board y se setean los pines a ser utilizados. Para la comunicación SPI se configura el modo 3 que es el cual utiliza la cámara y se setea el reloj en 16.8 MHz (resultantes de dividir entre 5 el reloj de 84 MHz de la Arduino DUE). La Flir Lepton tolera un reloj SPI de hasta 20 MHz, por lo que esta elección de reloj es la más rápida posible dada la placa Arduino DUE.

Se realiza una configuracion inicial de la Flir Lepton.

Para realizar una captura en primer lugar se sincroniza la cámara con la placa Arduino (mediante SPI), esto se consigue poniendo en nivel alto el pin SS correspondiente a la cámara y leyendo los frames que llegan hasta encontrar el último de la imagen. Luego de la sincronización se leen de a uno los frames de datos y se almacenan en una variable (que llamaremos image) en la Arduino. Para leer un frame se baja el pin SS se leen dos bytes y se levanta el pin nuevamente y se repite esto hasta que se recorre toda la imagen.

Luego de obtenida la imagen se procede a ubicar pixeles que hayan sensado temperaturas sobre el umbral buscado. Para ello se aplica un algoritmo que recorre toda la imagen buscando pixeles por sobre el umbral. Luego de encontrado se busca si los pixeles que lo rodean están dentro del umbral, y a su vez lo que rodean a estos, así se identifica una mancha de pixeles dentro del umbral. A esta mancha llamaremos cuerpo. Cuando se determina que se detectó el cuerpo en su totalidad se cuenta la cantidad de pixeles que lo componen y se haya el centro del mismo. Este centro es un "centro de masa" del cuerpo dado que se compone del promedio de índices en el eje x de la imagen y el promedio de los índices del eje y. Luego de identificado un cuerpo con estos valores, el algoritmo sigue recorriendo la imagen identificando, si es que los hay, más cuerpos. Se decidió que el cuerpo con mayor cantidad de pixeles es el cuerpo objetivo.

4.3. Cálculo de waypoints

Una vez que se detectó un cuerpo caliente u obstáculo, es necesario comunicar a Pixhawk el waypoint al que deber ir a continuación, ya sea para esquivar el obstáculo o para situarse sobre el cuerpo caliente. Para calcular este waypoint se trabajó en un sistema de coordenadas plano, debido a que tanto la imagen térmica como la distancia medida por el sensor, proveen información con estas características. La distancia medida por el sensor se puede pensar como el largo de un segmento de recta entre el obstáculo y el punto y la imagen térmica como una sección de plano. Como Pixawk utiliza coordenadas esféricas, es necesario convertir de un sistema al otro. A continuación, se ofrece una breve descripción de los sistemas de coordenadas utilizados.

4.3.1. Sistemas de coordenadas

Existen varios sistemas de coordenadas geográficas que permiten determinar la posición de un punto en la tierra. El más utilizado a nivel mundial es el WGS84[1], el cual aproxima la tierra por un elipsoide y tiene como coordenadas la latitud, longitud y altitud. También está el sistema UTM [2], el cual utiliza cuadrantes y coordenadas cartesianas planas para ubicar un punto en la tierra, independiente de su altitud. Este sistema utiliza la clásica proyección de Mercator de la tierra que se ve en los mapas en dos dimensiones.

[1]World Geodetic System 1984

[2]Universal Transverse Mercator coordinate system

WGS84

El WGS84 es un sistema de coordenadas geográficas mundial que permite localizar cualquier punto de la Tierra por medio de tres unidades dadas. Se trata de un estándar en geodesia, cartografía y navegación que data de 1984. Su error de cálculo se estima menor a 2 cm, lo que lo hace el más preciso, razón por la cual el Sistema de Posicionamiento Global (GPS) se basa en él.

Consiste en un patrón matemático de tres dimensiones que representa la tierra por medio de un elipsoide, un cuerpo geométrico más regular que la Tierra. En Figura 4.15 se puede observar la elipsoide.

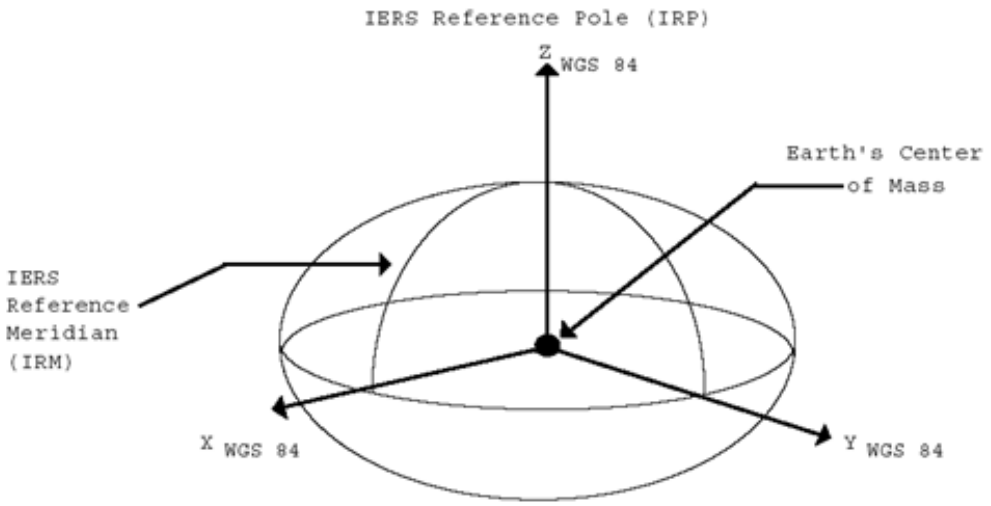


Figura 4.15: Elipsoide WGS84. Imagen obtenida de [36].

El sistema de referencia WGS84 es un sistema global geocéntrico, definido por los parámetros:

- Origen: Centro de masa de la Tierra
- Sistemas de ejes coordenados:
  - Eje Z: dirección del polo de referencia del IERS (The International Earth Rotation Service)
  - Eje X: intersección del meridiano origen definido en 1984 por el BIH y el plano del Ecuador (incertidumbre de 0.005 ")
  - Eje Y: eje perpendicular a los dos anteriores y coincidentes en el origen
- Elipsoide WGS84:
  - Semieje mayor (a): 6378137 m
  - Semieje menor (b): 6356752 m
  - Factor de aplanamiento (f): 298.257223563^{-1}
- Producto constante gravitacional y masa de la tierra (GM): $3.986004418 \times 10^{14}$ m^3/s^2
- Velocidad angular de la tierra (ω): 7.292115×10^{-5} rad/s
- Coeficiente de forma dinámica(J2): $-484.16685 \times 10^{-6}$

UTM

Por una cuestión de practicidad se proyecta este sistema de coordenadas geodésicas (expresado en grados) a algún otro sistema de coordenadas cartesiano de dos dimensiones llamados sistema de proyección típicamente UTM (Universal Transverse Mercator) que se expresan en metros con respecto a un punto de origen arbitrario facilitando cálculos de distancia y superficie.

El UTM utiliza un sistema de coordenadas cartesiano bidimensional para dar ubicaciones en la superficie de la Tierra. En la Figura 4.16 se pueden observar las zonas UTM.

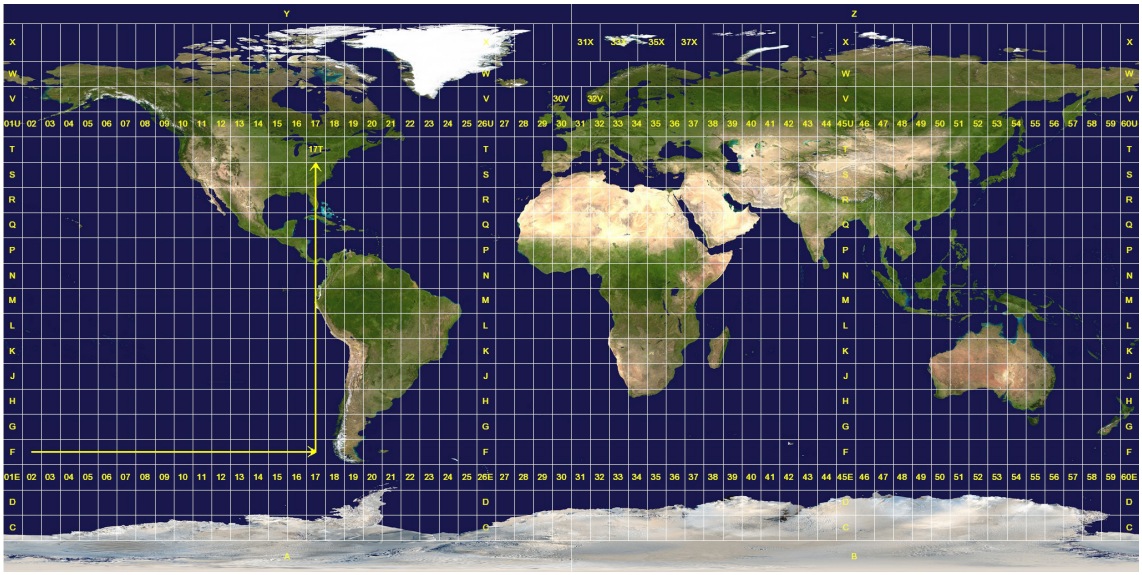


Figura 4.16: Zonas UTM, Imagen obtenida de [37].

El sistema UTM divide la Tierra en 60 zonas entre los 80° S y 84° N de latitud y cada 6° de longitud en ancho (desde los 180° O hasta los 180° E).

Cada una de las 60 zonas utiliza una UTM que puede trazar una región de gran extensión con baja distorsión.

Cada zona está segmentada en 20 bandas de latitud. Cada banda de latitud es de 8 grados de alto y se indica a partir de la letra C a los 80° S, aumentando en el alfabeto inglés hasta la letra X, omitiendo las letras I y O (debido a su similitud con los números uno y cero). La última banda de latitud, la correspondiente a la letra X, se extiende un extra de 4°, para alcanzar los 84° N de latitud, cubriendo así la tierra más septentrional del planeta Tierra.

Una posición en la Tierra está dada por el número de zona UTM y el par de coordenadas planas de este y de la zona norte. El punto de origen de cada zona UTM es la intersección del ecuador y el meridiano central de la zona. Para evitar tratar con números negativos, el meridiano central de cada zona se define para que coincida con 500000 metros Este.

En el hemisferio norte las posiciones se miden hacia el norte desde cero (ubicado en el ecuador). El valor "northing" máximo es de unos 9300000 metros a latitud 84 grados Norte (el extremo norte de las zonas UTM). En el hemisferio sur, disminuyen hacia el sur desde el ecuador hasta unos 1100000 metros a 80 grados sur (el extremo sur de las zonas UTM). En este caso el northing en el ecuador se fija en 10000000 metros así que ningún punto tiene un valor negativo de northing.

Conversión entre sistemas

El receptor GNSS a bordo del Dron utiliza el sistema WGS84 para la estimación de posicionamiento. Por otro lado, el programa de detección térmica devuelve un punto en el plano de la superficie terrestre que puede ser convertido a coordenadas UTM. Para pasar un nuevo waypoint al controlador, éste debe estar en coordenadas WGS84. Por ende, primero se debe pasar la posición actual del Dron de coordenadas WGS84 a UTM. Esto ubicará al Dron en un cuadrante del plano. Luego realizar los cálculos de la posición en el plano a la que se desea ir y por último pasar las coordenadas resultantes de UTM a WGS84 para que sean utilizadas por el controlador.

Para pasar de un sistema a otro existen diversos métodos. Se mencionan solamente los más utilizados:

- Tablas de la Proyección UTM - Dichas tablas se pueden obtener del volumen II de la publicación Servicio Geográfico del Ejército de España (SGE), Sección de Geodesia (1976): Proyección Universal Transversal Mercator, SGE, Madrid.
- Fórmulas de transformación directa del US Army, publicadas en 1973 (véase el USGS Bolletin Num. 1532).
- Fórmulas de Coticchia-Surace, dichas ecuaciones fueron planteadas por Alberto Coticchia y Luciano Surace en el "Bolletino di Geodesia e Science Affini", Num. 1. Estas ecuaciones son las elegidas para implementar la conversión de coordenadas en la placa on-board, debido a que son muy precisas y de fácil implementación.

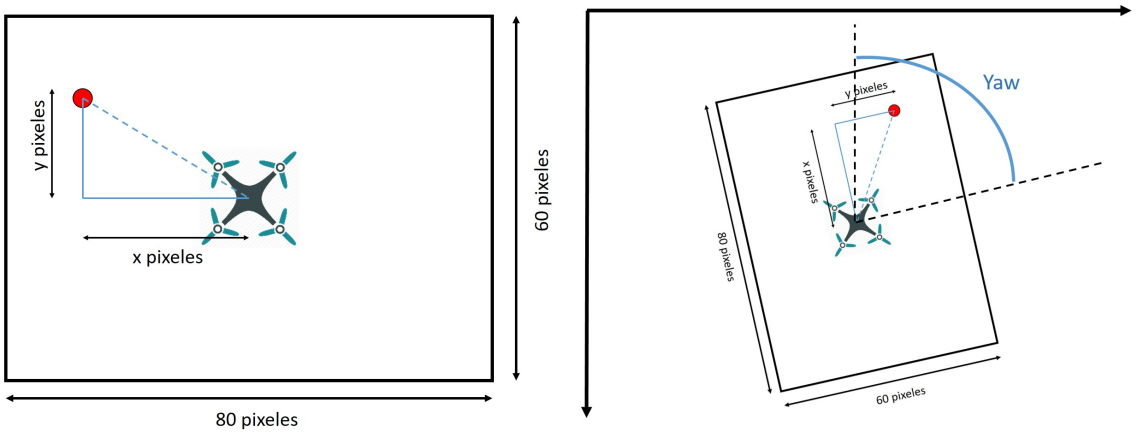
Por ende, para el cálculo de los nuevos waypoint, que modifican la misión, primero se obtiene la posición del Dron en coordenadas WGS84 desde el controlador de vuelo. Luego, utilizando las ecuaciones de Coticchia-Surace, se convierten a coordenadas a UTM. En ese sistema se realiza el cálculo del nuevo waypoint sumando la desviación correspondiente. Por último, se vuelve a convertir el waypoint calculado a coordenadas WGS84 y es enviado a la controladora de vuelo para modificar la misión.

Para calcular un waypoint sobre un cuerpo caliente detectado se parte de los índices del pixel central de la imagen del cuerpo. El plano de la imagen tiene al Dron en el punto central y desde ahí se calcula el punto al que debe moverse para posicionarse sobre el cuerpo encontrado. Luego se realiza una conversión de pixel de distancia a metros de distancia sobre el plano del suelo y por último se agrega la componente de orientación respecto al campo magnético de la tierra (ángulo Yaw) para obtener la posición del punto a agregar en el sistema UTM. Las ecuaciones se pueden observar en la Figura 4.17. El nuevo waypoint queda de la siguiente forma:

$$x[n+1] = x[n] + 0.011h[n]\left((x'_p - 40)\cos\left(yaw[n]\right) + (30 - y'_p)\sin\left(yaw[n]\right)\right)$$

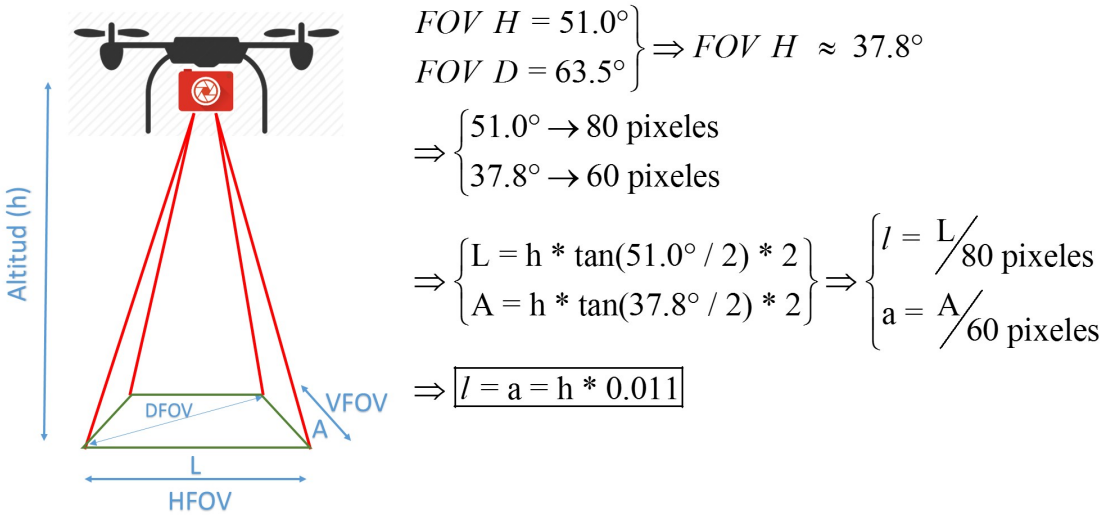
$$y[n+1] = y[n] + 0.011h[n]\left((30 - y'_p)\cos\left(yaw[n]\right) - (x'_p - 40)\sin\left(yaw[n]\right)\right)$$

Donde $x[n+1]$ e $y[n+1]$ corresponden al nuevo waypoint a ser calculado, $x[n]$ e $y[n]$ corresponden al waypoint actual y $h[n]$ y $yaw[n]$ son la altura actual y el yaw actual respectivamente. Todos estos valores referenciados al sistema de coordenadas



(a) Componente distancia en pixeles

(b) Componente orientación



(c) Componente de altura y FOV

Figura 4.17: Cálculo de waypoint dado un cuerpo caliente.

(x, y) del sistema UTM. x'_p e y'_p corresponden a la posición del pixel actual en el sistema de coordenadas (x', y') referenciado al cuadro de la imagen. El factor 0.011 se muestra en la Figura 4.17b.

Para calcular un waypoint generado por una desviación provocada por un obstáculo detectado se utiliza trigonometría tomando en cuenta el ángulo de orientación del Dron respecto al campo magnético de la tierra y las ecuaciones quedan: Para una desviación a la derecha:

$$x[n+1] = x[n] + detour \cos(yaw[n])$$

$$y[n+1] = y[n] - detour \sin(yaw[n])$$

Para una desviación a la izquierda:

$$x[n+1] = x[n] - detour \cos(yaw[n])$$

$$y[n+1] = y[n] + detour \sin(yaw[n])$$

Para una desviación al frente:

$$x[n+1] = x[n] + detour \sin(yaw[n])$$

$$y[n+1] = y[n] + detour \cos(yaw[n])$$

Para una desviación hacia atrás:

$$x[n+1] = x[n] - detour \sin(yaw[n])$$

$$y[n+1] = y[n] - detour \cos(yaw[n])$$

4.3.2. Distancia entre dos puntos

Una de las operaciones que es necesario computar es la distancia entre dos puntos. Dadas la latitud y la longitud de un punto en la tierra, hay varios métodos para calcular la distancia entre ambos [38].

Fórmula del semiverseno

A pesar de que la forma de la tierra considerada en WGS84 es de un elipsoide achatado en los polos, para calcular la ortodrómica[3] entre dos puntos se aproxima la tierra por una esfera y se utiliza la fórmula de semiverseno o haversine. Por definición el semiverseno es:

$$haversin(\theta) = \frac{versin(\theta)}{2} = \frac{1 - cos(\theta)}{2} = sin^2\left(\frac{\theta}{2}\right)$$

La fórmula del semiverseno es un caso particular de una fórmula más general de la trigonometría esférica, la ley de los semiversenos [39]. Dados dos puntos de latitudes ϕ_1 y ϕ_2 y longitudes λ_1 y λ_2 respectivamente y siendo R el radio de la tierra, la distancia entre ellos se puede calcular como:

$$D = 2R\, asin\left(\sqrt{haversin(\phi_1 - \phi_2) + cos(\phi_1)cos(\phi_2)haversin(\lambda_1 - \lambda_2)}\right)$$

[3]Camino más corto entre dos puntos de la superficie terrestre.

Aproximación rectangular

Otro método a priori menos preciso, pero igualmente útil en distancias cortas es la proyección cilíndrica equidistante. Esto es, calcular los arcos de circunferencia dados por las variaciones en latitud y longitud, aproximar el elemento de superficie esférica por un plano y utilizar el teorema de Pitágoras. Con esto se tiene que:

$$D = R\sqrt{(\phi_1 - \phi_2)^2 + \left((\lambda_1 - \lambda_2)cos\left(\frac{\phi_1 + \phi_2}{2}\right)\right)^2}$$

El peso de cada cifra significativa

Para poder saber qué truncamiento es razonable admitir en las coordenadas, es necesario saber qué variación en metros implica una variación en una unidad en cada cifra significativa en la latitud y en la longitud. Esto depende de qué punto de la tierra se esté considerando. El peor caso se da en el Ecuador, ya que es la latitud con mayor diámetro, a mayores latitudes la variación de en un grado implicará menos metros. Una primera aproximación para saber cuántos kilómetros representa un grado es simplemente calcular el cociente entre la circunferencia de la tierra (40075 km) y 360, lo cual equivale a 111.3 km por grado.

De todas maneras se realizó un cálculo con un punto arbitrario en el predio de la Facultad de Ingeniería de la UdelaR para tener la variación en la zona donde el Dron se moverá. Se consideró el punto de latitud y longitud -34.918279 y -56.166426 respectivamente, se varió en una unidad cada cifra significativa y se calculó la distancia con respecto al punto original tanto con el método del semiverseno como con la aproximación rectangular. Los resultados de pueden observar en las tablas 4.1 y 4.2

Latitud (°)	**Distancia semiverseno**	**Distancia rectangular**
-24.9182795	1112 km	1112 km
-33.9182795	111.2 km	111.2 km
-34.8182795	11.12 km	11.12 km
-34.9082795	1.112 km	1.112 km
-34.9172795	111.2 m	111.2 m
-34.9181795	11.12 m	11.12 m
-34.9182695	1.112 m	1.112 m
-34.9182785	11.12 cm	11.12 cm
-34.9182794	1.112 cm	1.112 cm

Tabla 4.1: Distancia con respecto al punto -34.918279 y -56.166426 variando la latitud y manteniendo longitud constante

Observando estas tablas se puede concluir que, hasta el orden de los mil km, es prácticamente igual utilizar el método del semiverseno o el rectangular (con cuatro dígitos de resolución). También se puede concluir que, si se quiere tener resolución del orden de los cm en la estimación de la posición, es necesario tener precisión el séptimo dígito a la derecha del separador decimal en las coordenadas. Los receptores

Longitud (°)	Distancia semiverseno	Distancia rectangular
-46.1664265	911.4 km	911.8 km
-55.1664265	91.18 km	91.18 km
-56.0664265	9.118 km	9.118 km
-56.1564265	911.8 m	911.8 m
-56.1654265	91.18 m	91.18 m
-56.1663265	9.118 m	9.118 m
-56.1664165	91.18 cm	91.18 cm
-56.1664255	9.118 cm	9.118 cm
-56.1664264	9.118 mm	9.118 mm

Tabla 4.2: Distancia con respecto al punto -34.918279 y -56.166426 variando la longitud y manteniendo latitud constante

de GNSS manejan justamente esta precisión. Otra conclusión útil a la hora de hacer estimaciones es que no es tan errado considerar que variaciones en el séptimo digito a la derecha del separador decimal se corresponden uno a uno con variaciones de distancia en centímetros, en el sexto con decímetros y así sucesivamente. Esto último es una aproximación mucho más imprecisa, pero es útil para hacer estimaciones rápidas y que no requieran gran exactitud.

El problema con el punto flotante

Dadas la latitud y longitud de un punto en la tierra, es necesario utilizar un tipo de dato para almacenarlo y operar con él en un procesador. Como se estudió en la sección anterior, es necesario poder representar 9 dígitos, el lugar donde va el separador decimal y el signo. Es por esto que se debe tener especial consideración en cuanto a qué tipo de dato elegir. Normalmente se utilizaría un tipo de dato *float*[4], pero la precisión que este tipo provee no es suficiente para tener precisión de cm en la estimación de la posición. Una segunda opción es usar doble precisión (64 bits), esto sí alcanzaría para tener la resolución deseada pero el protocolo MAVLink está implementado con *float* por lo que esto no es una opción. Otra representación que ofrece MAVLink es la representación de las coordenadas como enteros de 32 bits con signo con unidades de $^\circ \times 10^7$, con esta representación sí se logra la precisión deseada en la representación de las coordenadas.
Para cuantizar el error introducido al utilizar punto flotante de simple precisión, se realizó el siguiente experimento en MATLAB: Se crean dos vectores de latitud y longitud con tipo de dato *double*[5] con 100 muestras cada uno y variación de $1 \times 10^{-7\circ}$ entre sus elementos, se crean dos vectores de tipo *float* truncando los dos primeros y se calcula el error introducido por el truncamiento en cm. En la figura 4.18 se puede observar el comportamiento del error según las coordenadas. El error máximo introducido es de 27.24 cm.

[4]Representación en punto flotante de 32 bits
[5]Representación en punto flotante de 64 bits

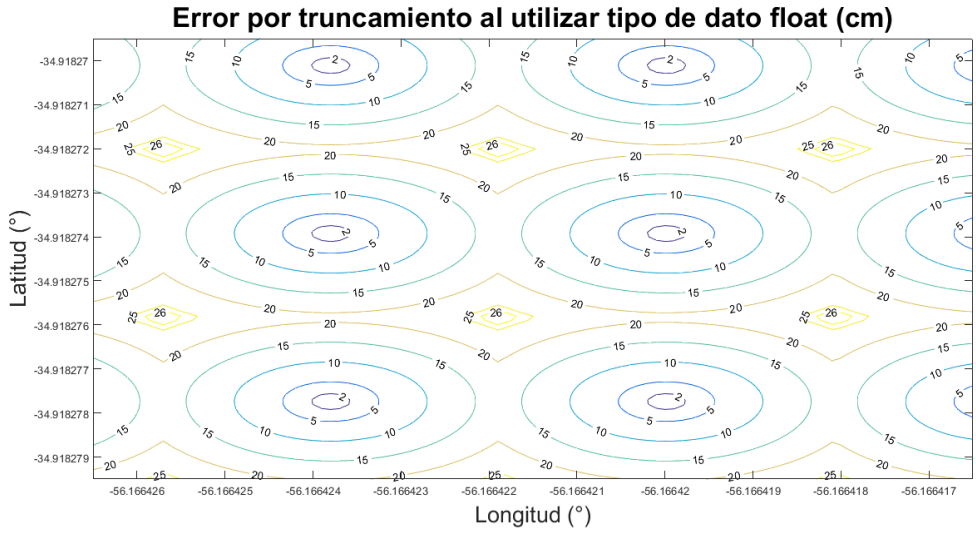


Figura 4.18: Curvas de nivel del error introducido por truncamiento al utilizar punto flotante de simple precisión (32 bits).

4.4. Comunicación Pixhawk y Arduino

Para la comunicación entre Pixhawk y Arduino (tanto para la placa on-board como para la GCS) se adopta el protocolo de comunicación MAVLink, que es el que Pixhawk está programada para utilizar. Esta comunicación se hará a través de puertos serie a una tasa de transferencia de 57600 baudios 8N1 (8 bits de datos, no paridad, 1 de parada).

El protocolo MAVLink consta de un encabezado, el payload y un checksum para chequeo de recepción. En la Figura 4.19 se puede observar la estructura de un mensaje MAVLink.

Para cada uno de las acciones que se implementaron en la comunicación entre Pixhawk y Arduino se consultó la librería de mensajería MAVLink [40]. Más adelante en este documento se detallan algunos de los comandos de mayor relevancia que se tuvo que estudiar para la implementación.

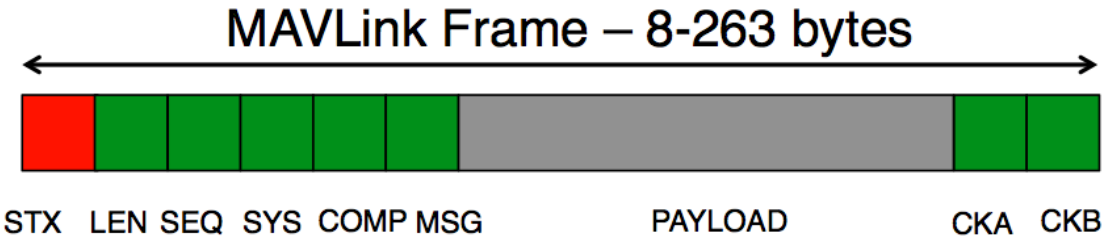


Figura 4.19: Estructura de un paquete MAVLink. Imagen obtenida de [41].

4.4.1. Conexiones entre Pixhawk y Arduino

Pixhawk cuenta con puertos específicamente diseñados para implementar una comunicación serie. Estos puertos son el TELEM 1, TELEM 2 y SERIAL 4/5. Se

utiliza el puerto TELEM 1 para comunicación con la GCS vía radio 915 MHz y el puerto TELEM 2 para la comunicación con la placa on-board que se conectan directamente. Del lado de Arduino se utilizan los puertos serial y para el caso de la on-board comparte la referencia a GND.

Para el envío de información desde la placa on-board a la GCS se utilizó un buffer triestado con el fin de tomar el control del transceptor y así comunicarse con la GCS. Una vez finalizada la comunicación, se libera el transceptor y Pixhawk vuelve a comunicarse con la GCS. Esta acción se utiliza específicamente para el caso en que la placa on-board envía una imagen capturada o logs de información. El integrado utilizado es el 74244 - SN74HCT244N del cual se puede apreciar su diagrama esquemático en la Figura 4.20.

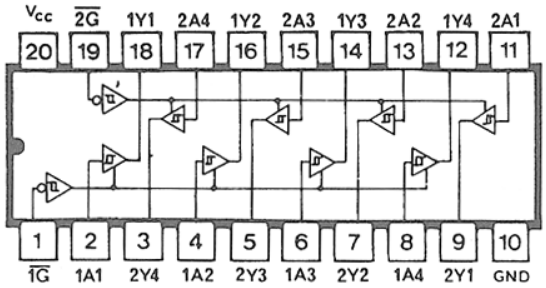


Figura 4.20: Diagrama esquemático del buffer 74244.

4.5. Calibración y caracterización de equipos

Algunos equipos deben ser calibrados para su uso, en particular los que toman algún tipo de medida. En este proyecto se utilizaron sensores de distancia, humedad, temperatura, la cámara térmica y además Pixhawk cuenta con acelerómetro, giroscopio, magnetómetro y el receptor de GNSS cuenta con otro magnetómetro. Todos estos equipos deben ser calibrados para obtener medidas más exactas. En esta sección se detallan los procedimientos y resultados de estas calibraciones.

4.5.1. Calibración de los sensores de Pixhawk

Los sensores de Pixhawk no vienen calibrados de fábrica y es necesario calibrarlos para su funcionamiento. QGC provee una forma de hacerlo siguiendo un procedimiento. A continuación se detalla el procedimiento de calibración que fue utilizado cada vez que se cambió de HIL a modo cuadricóptero en X genérico.

Materiales

- Pixhawk
- Receptor GNSS con magnetómetro
- Sistema de telemetría (dos antenas, receptor y emisor)

- Batería
- Canetígiro[6]
- Nivel
- Notebook
- QGroundControl (QGC)

Introducción

QGC consiste en un software que, corriendo en un ordenador, implementa una GCS (originalmente orientada a Pixhawk, pero que también funciona para otras placas como Ardupilot o cualquier otra controladora de vuelo que use MAVLink) con una interfaz gráfica que permite configurar la placa (e.g. elegir el tipo de vehículo, batería, asignación de los canales de radio, etc.), en particular calibrar los sensores.

QGroundContol requiere que la placa más el magnetómetro externo se coloquen en determinadas posiciones y se realicen determinados movimientos para calibrar los sensores.

Para calibrar el magnetómetro se debe posicionar el sistema en seis posiciones diferentes y rotar el mismo, para el giroscopio se debe posicionar en un plano horizontal y mantener quieto para el acelerómetro lo miso que para el giroscopio, pero sin rotarlo.

Procedimiento

Se fija la placa Pixhawk a un plano con el receptor GNSS, se conectan la batería y el Tx/Rx de telemetría. Se conecta el Tx/Rx en la notebook y se abre QGroundContol.

Dentro de QGC se selecciona Configuration, Sensors y se comienza por el magnetómetro (Compass). En la Figura 4.21 tal se puede observar la GUI de la calibración. Para implementar estos movimientos y posiciones se hace uso del Canetígiro, se fija la placa en una posición y así se consiguen cuatro de las seis posiciones y luego se rota noventa grados para conseguir las otras dos. En la Figura 4.22 se puede observar cómo se llevó a cabo este procedimiento.

Para el giroscopio se requiere que la placa descanse sobre un plano horizontal, esto se hace simplemente dejando uno de los planos de la calibración anterior sin mover el sistema. Para el acelerómetro se repiten los pasos de la calibración del magnetómetro, pero sin rotación.

4.5.2. Calibración de los sensores de distancia

Descripción del equipo y marco teórico

Los sensores de ultrasonido son utilizados para medir distancias mediante el uso de ondas ultrasónicas. El principio de funcionamiento se basa en las propiedades

[6]El Canetígiro es un dispositivo mecánico que permite la libre rotación en el ángulo yaw y variar el roll (o pitch) para conseguir los planos necesarios para la calibración

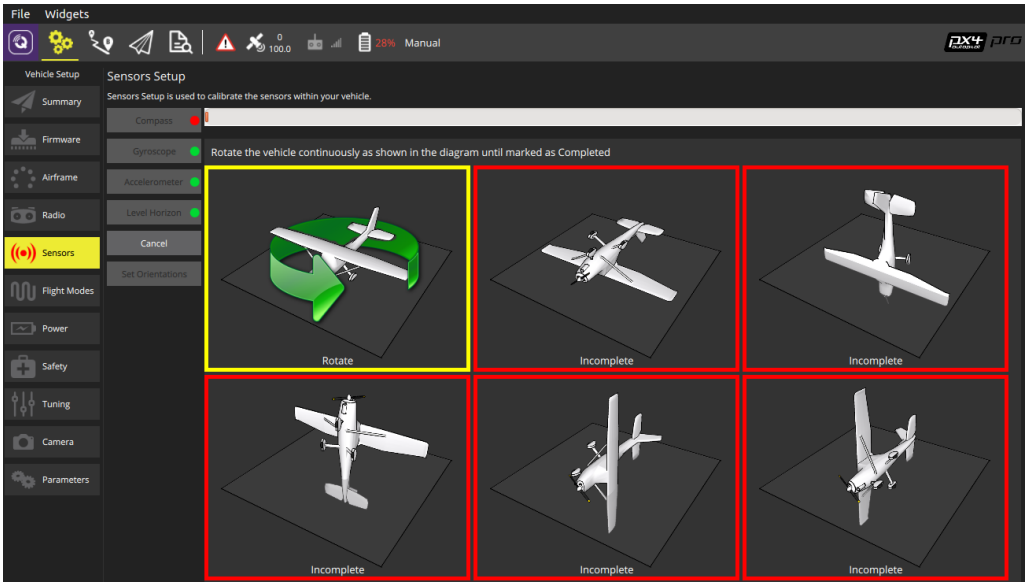


Figura 4.21: GUI de QGC calibración del magnetómetro.

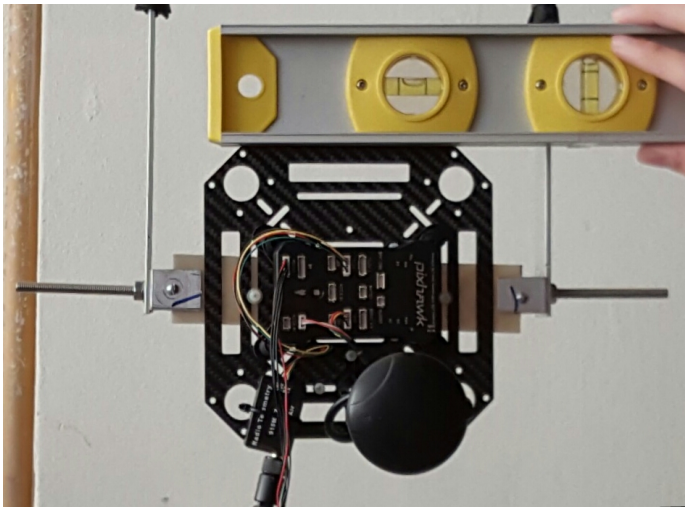


Figura 4.22: Placa montada en Canetígiro para calibración.

magnetoestrictivas de algunos materiales. Una lámina de material magnetoestrictivo tiene la propiedad de, al ser excitada por una corriente eléctrica, deformarse mecánicamente y generar ultrasonidos. Dicho efecto se produce de forma inversa, al recibir dicho material una vibración mecánica se produce una corriente eléctrica. Utilizando este efecto, los sensores emiten una onda ultrasónica excitando una membrana magnetoestrictiva con impulsos eléctricos. Esta onda choca contra los obstáculos y luego el sensor recibe los ecos de esa señal y la convierte en impulsos eléctricos. Existen dos configuraciones de sensores de ultrasonido, una con el emisor y el receptor en el mismo transductor y otra con dos transductores en donde uno es emisor y el otro es receptor. La última configuración es más apropiada para medir distancias cortas debido a que la membrana receptora está lista para recibir la señal reflejada desde el mismo instante en el que se emite la onda, mientras que la primera configuración requiere de un tiempo de bloqueo de la señal emitida para poder recibir la onda rebotada.

La distancia se puede calcular con la siguiente fórmula:

$$D = \frac{tc}{2}$$

Donde D es la distancia, t es el tiempo entre la emisión y la recepción, y c es la velocidad del sonido en el medio, en el caso del aire se considera 343.2 m/s. De todas maneras, hay defectos de fabricación, delays en las interfaces y otro tipo de imprecisiones que hacen que esta ecuación no se cumpla estrictamente lo cual hace necesario calibrar cada equipo.

El sensor tiene un ángulo de detección en el cual es teóricamente sensible, y tiene otros lóbulos secundarios para los cuales es sensible, pero en menor medida. La Figura 4.23 muestra la atenuación de la onda en dB según el ángulo de incidencia. La distancia máxima de detección depende de la frecuencia de la onda ultrasónica

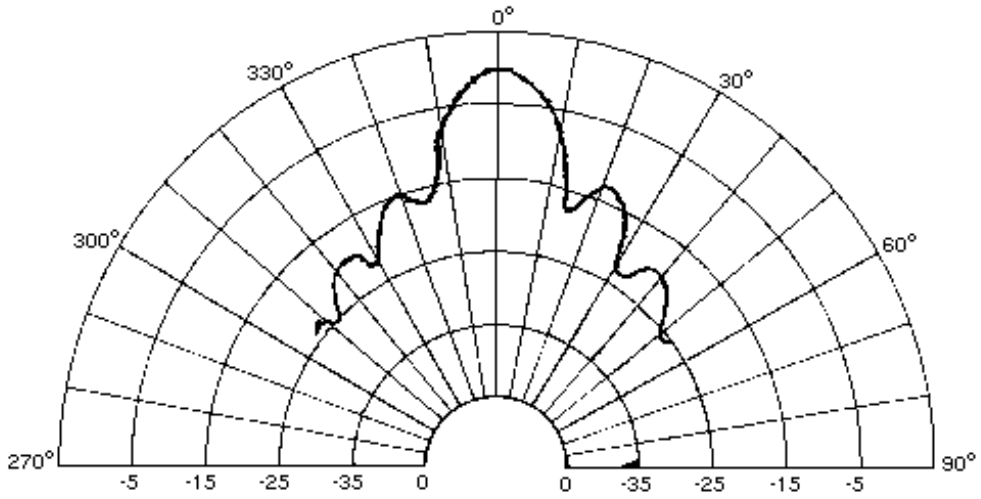


Figura 4.23: Atenuación de la onda en dB según el ángulo de incidencia. Imagen obtenida de [42].

emitida, la electrónica implementada, la membrana y el medio en el que se propaga. Estos sensores tienen una variación despreciable ante variaciones de humedad y son independientes de la presión y la altitud. Por otro lado, la temperatura influye en la densidad del aire y esta a su vez incide sobre la velocidad de propagación de la onda según la siguiente ecuación:

$$V_s = V_{s0}\sqrt{1 + \frac{T}{273}}$$

Donde V_{s0} es la propagación de la onda sonora a 273 K y T es la temperatura en grados Celsius. Otras desventajas de estos sensores tienen que ver con la conformación del obstáculo (irregularidades en la superficie), ángulo de incidencia, ángulo de reflexión, que el objeto se encuentre en la periferia del cono de detección, falsos ecos, etc. Los sensores elegidos para el proyecto fueron los US-015, los cuales son alimentados con 5 V y tienen un consumo de 2.2 mA en reposo y picos de hasta 15 mA. El ángulo de detección es de 15 ° y cuenta con un rango de detección de entre 2 cm y 400 cm. La precisión es de 0.3 cm ± 1 %. El tamaño es de 45 x 20 x 1.6 mm.

Procedimiento de calibración

Para la calibración de estos sensores se tomaron una serie de medidas con respecto a una pared. Los materiales que se utilizaron fueron: una SBC Arduino DUE, sensores US-015, una cinta métrica, una mesa móvil y una notebook.

Se monta la Arduino con un sensor conectado orientado hacia la pared sobre la mesa, se mide la distancia entre el sensor y la pared con la cinta métrica y se registra la lectura del sensor. La salida del sensor es un tiempo en microsegundos. Se tomaron varias medidas y se aproximó la relación tiempo-distancia por el método de mínimos cuadrados de primer orden.

Se comparó el resultado de la calibración con el modelo físico y se observó una diferencia de un 3.66 %. A los efectos de este proyecto, esta diferencia es admisible y podría usarse el modelo físico. De todas maneras, se calibraron los sensores y se utilizaron los parámetros de la calibración. En las figuras 4.24 y 4.25 se pueden observar la linealización y la diferencia entre esta y el modelo físico. También se notó que por debajo de los 20 cm la medida de los sensores no es confiable, esto no representa un problema para el uso que se les da en este proyecto.

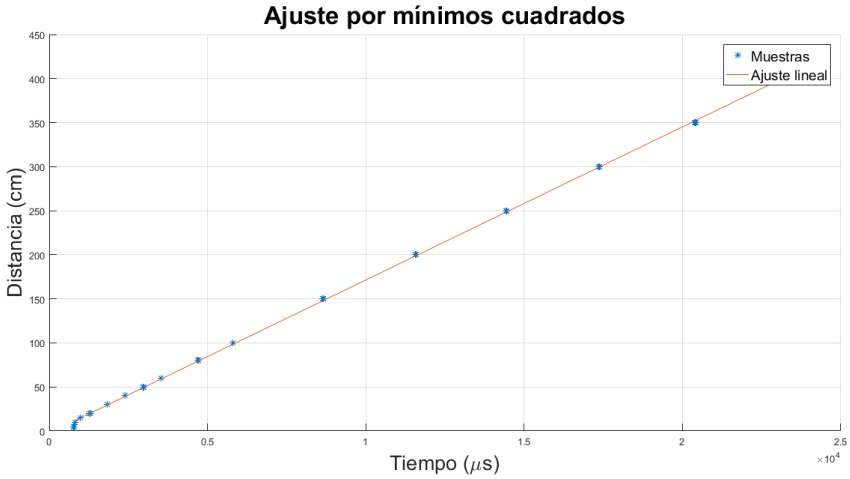


Figura 4.24: Ajuste de la relación tiempo-distancia de los sensores US-015 por mínimos cuadrados de orden 1.

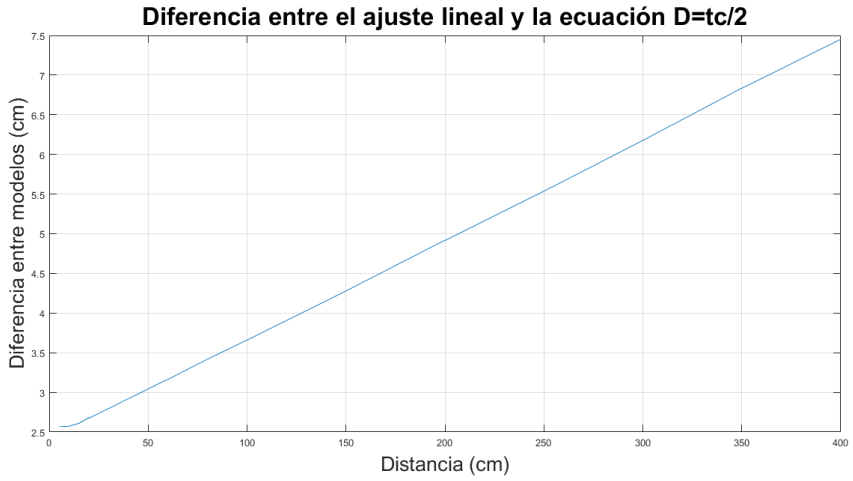


Figura 4.25: Diferencia entre el ajuste y la ecuación física.

4.5.3. Calibración de la cámara térmica

Para la calibración de la cámara térmica se llevó a cabo un procedimiento mediante el cual se tomaron imágenes a un vaso de bohemia conteniendo agua a diferentes temperaturas. La temperatura del agua se midió con un termómetro de mercurio. Se tomaron varias imágenes para una misma temperatura. Estos datos fueron procesados y se obtuvo una caracterización de la salida de la cámara con respecto a la temperatura a la que se encontraba el agua.

Para la caracterización de la cámara térmica se utilizó un modelo de primer orden según recomienda el fabricante [33]. La temperatura medida se puede estimar por la siguiente ecuación:

$$T = aRaw14 + bT_{cam} + c$$

Donde $Raw14$ es el valor reportado por la cámara para determinado pixel y T_{cam} es la temperatura de la cámara. Se realizó una linealización por mínimos cuadrados en MATLAB[7] y se obtuvieron los siguientes resultados:

- a = 0.0327
- b = 27.637
- c = -885.39

En las figuras 4.26, 4.27 y 4.28 se pueden observar las imágenes de termografía del vaso de bohemia para una temperatura de 75 °C y 15 respectivamente °C y los resultados de la regresión lineal en forma gráfica.

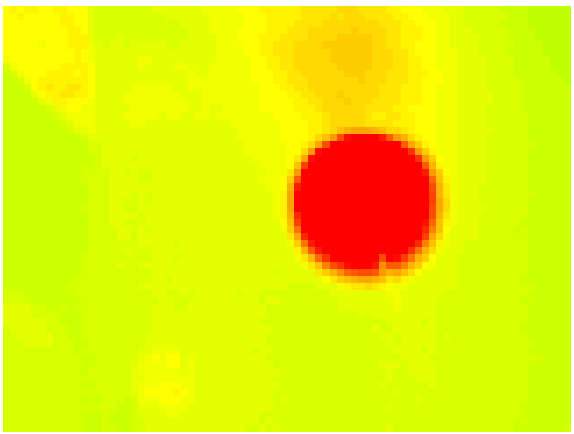


Figura 4.26: Imagen térmica del vaso de bohemia a 75 °C.

[7]MATLAB no cuenta con una función para hacer regresión lineal en dos dimensiones, los créditos van a John D'Errico 2006 por la función polyfitn

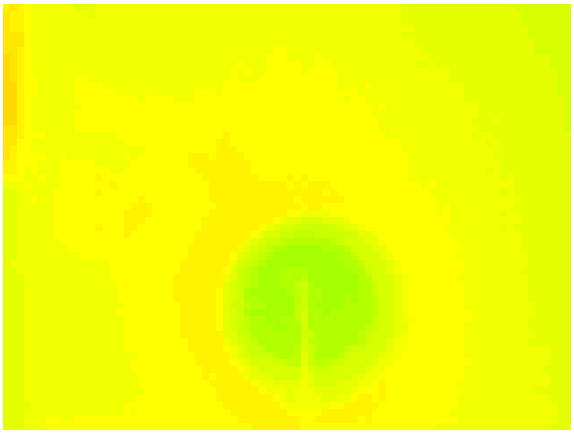


Figura 4.27: Imagen térmica del vaso de bohemia a 15 °C.

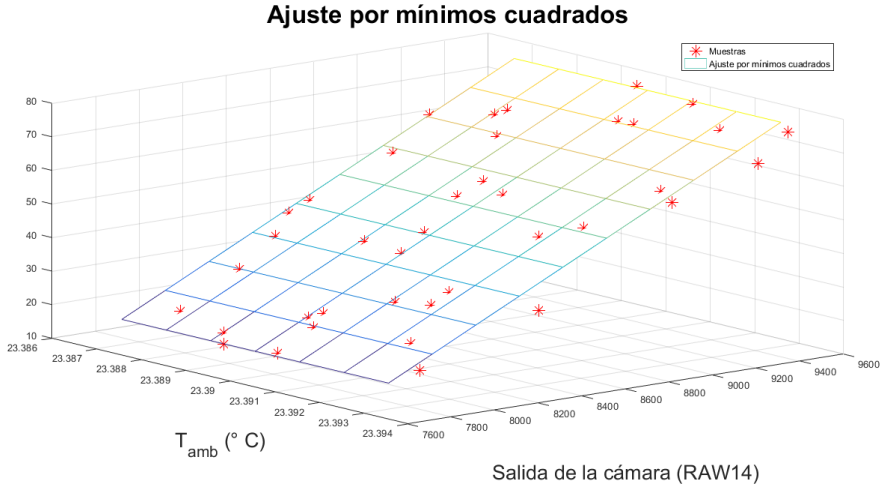


Figura 4.28: Muestras de las mediciones contra el plano que mejor aproxima la relación entre Raw14 y la temperatura de la cámara con la temperatura del punto de la imagen.

CAPÍTULO 5

IMPLEMENTACIÓN DE LOS MÓDULOS FUNCIONALES DE LA GCS

5.1. Software

5.1.1. TaskScheduler

Debido a que el programa necesita cumplir varias funciones simultáneamente, se optó por utilizar un enfoque multitarea para implementarlo. Para esto se utilizó la librería TaskScheduler [43] que implementa las clases Scheduler y Task. La clase Scheduler es la que se ocupa de organizar en el tiempo las tareas y a su vez cumplir la función del "array que las contiene". La clase Task implementa las tareas propiamente dichas, que no son más que funciones que deben ejecutarse con determinado periodo, cierta cantidad de veces y con cierta prioridad con respecto a las demás.

Debido a que el programa a implementar no implicaba gran complejidad a la hora de priorizar tareas ni tampoco se requería que alguna tarea fuera realizada una cantidad determinada de veces, no se utilizaron todas las funcionalidades de esta librería y varias situaciones se resolvieron de manera sencilla.

Por ejemplo, instrucciones que debían ejecutarse una sola vez fueron colocadas en la función *setup()* y todo lo que fuese periódico fue en tareas. Dentro del *loop()* la única instrucción que se ejecuta es el método *execute()* de la instancia de Scheduler del programa de manera de delegar completamente el control sobre las tareas a la misma. En la Figura 5.1 se puede observar el diagrama de bloques del ciclo de vida de una tarea.

5.1.2. Buffer de Arduino

Si bien no se trabajó sobre el buffer de Arduino ni se tuvo en cuenta a la hora del diseño, cabe mencionar que los buffers de los puertos UART de Arduino tienen un tamaño por defecto de 64 bytes. Esto es una gran limitante a la hora de la implementación ya que, si no se tiene en consideración, es muy probable que se

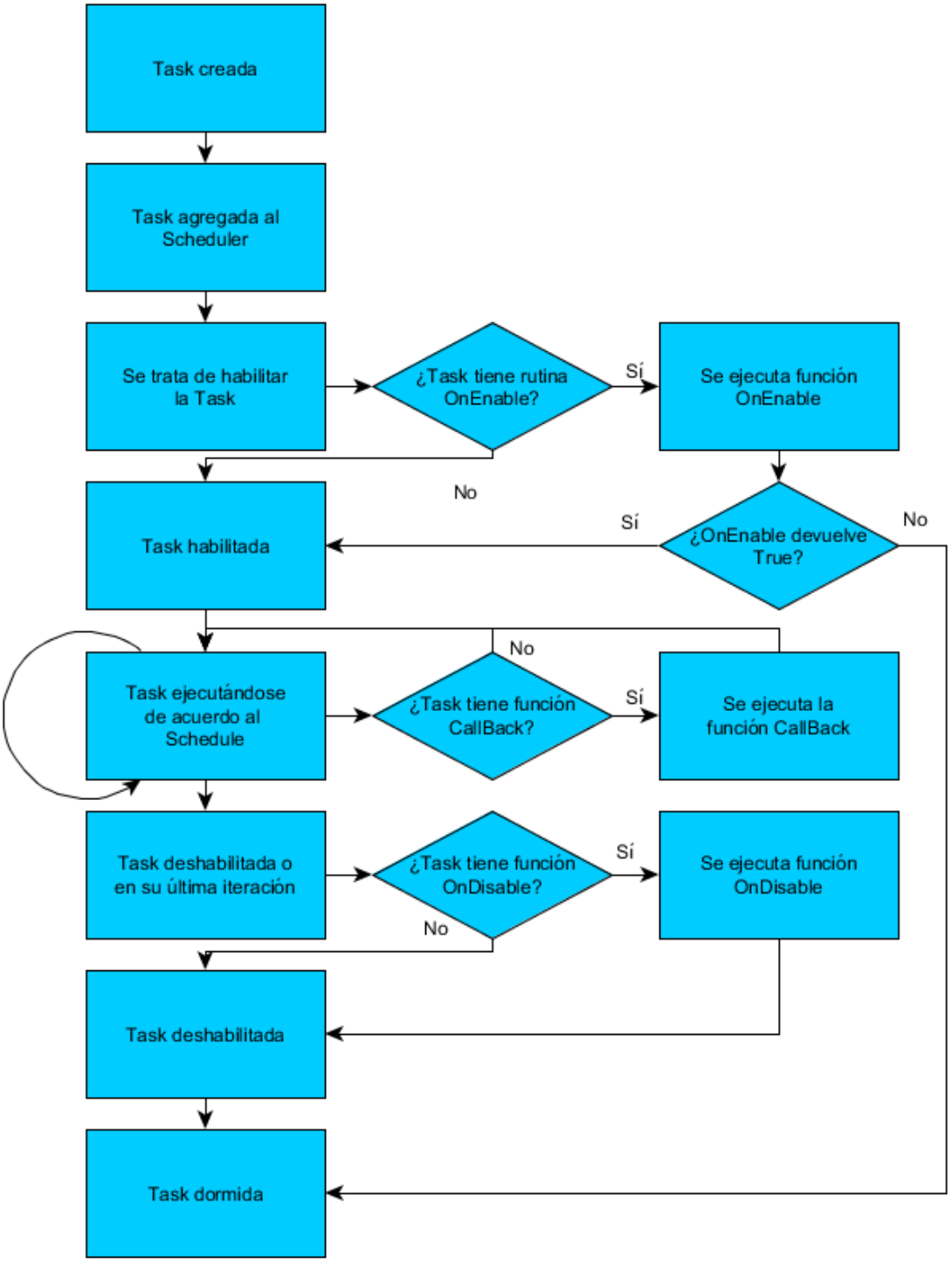


Figura 5.1: Ciclo de vida de una tarea.

colme su capacidad y se corrompan los datos recibidos. Esto llevó a que se eligieran los baudrates y se leyeran los puertos seriales de manera tal de no llegar nunca a saturar un puerto. En particular el stream de RTCM de la estación UYMO es de 9600 baudios constantes, por lo que cuando se están recibiendo correcciones de GNSS se debe tener especial cuidado ya que, si no se lee el puerto por 53 ms, el buffer se llena y los bytes que allí se guardan deben desecharse. Una medida que se tomó para esto fue incrementar al máximo el baudrate entre el Shield 3G y Arduino (19200 baudios) de manera de que cuando se esté leyendo el puerto se pierda menos tiempo. Leer el puerto lo más rápido posible es una condición necesaria pero no suficiente para evitar la saturación, también se debe leer lo más rápido posible sin perjuicio de las demás tareas.

Esto implica que el programa debe tener un enfoque multitarea y justamente la librería *TaskScheduler* brinda el marco para esto. Las demás librerías que hay disponibles para cada tarea no fueron concebidas con estas condiciones por lo que se tuvieron que tomar como punto de partida y reescribir casi completamente en algunos casos. Por citar un ejemplo, la librería para el shield3G implementaba una función *sendATCommand* que imprimía un mensaje en el puerto serial y se quedaba esperando un tiempo (que se le pasa a la función como parámetro) esperando a que le llegue la respuesta. Durante ese tiempo, que podría ser de algunos segundos, se podría llenar el buffer de la comunicación con Pixhawk y esto es inaceptable.

Para enfrentar esta situación se separó la parte de trasmisión de la parte de recepción de datos. Cada vez que es necesario enviar un mensaje y recibir una respuesta una tarea escribe en el puerto serial, levanta una bandera y otra tarea chequea si hay datos disponibles y si son los que eran esperados por la primera, permitiendo que otras tareas se ejecuten durante ese tiempo.

Una estrategia extra que no se utilizó por no llegar a ser necesaria, fue aumentar la capacidad de los buffers de Arduino modificando sus archivos de configuración.

5.2. Tareas Implementadas

Por ser un programa de dimensiones considerables, se optó por segregar las tareas lo máximo posible y mantener un mismo estilo de programación y estructura de manera de poder reutilizar código y entender un bloque nuevo desarrollado por otra persona bajo el mismo estilo. Finalmente, las tareas implementadas son:

- *checkEMail*
- *sendEMail*
- *EnviarReportes*
- *EMailReader*
- *Xmodem*
- *XmodemReader*
- *RTCMReader*
- *RTCM*
- *heartBeat*

- *PX4Reader*
- *Shield3GReader*

De manera de poder probar las funcionalidades del programa por separado, se utilizaron definiciones condicionales (*#define* con *#ifdef*) para agrupar tareas y que no interfieran unas con otras, ya sea en tiempo de ejecución o en utilización de puertos de comunicación.

Las definiciones condicionales que se utilizaron son:

- *RTCMdef*
- *EMaildef*
- *shield3Gdef*
- *PX4def*
- *ConsolaMon*

En la Figura 5.2 se puede observar el flujo del programa principal.

5.2.1. Shield 3G

En el puerto UART que conecta el Shield 3G con Arduino se implementan varios protocolos de comunicación: el control del Shield mediante comandos AT, la recepción de RTCM y la transferencia de archivos por Xmodem. La manera de interpretar los bytes leídos es distinta para cada uno por lo que se programaron tareas distintas para cada caso.

En general se tiene una tarea y otra homónima con "Reader" concatenado para segregar la escritura de la lectura del puerto. Además de esto, dentro de la tarea principal, se implementó una máquina de estados de manera de poder enviar un mensaje, cambiar de estado, prender una bandera y deshabilitar la tarea de escritura y no volver a habilitarla hasta recibir la respuesta que el Reader va a encargarse de leer. Asimismo, las tareas Reader solo hacen algo si hay un byte para leer en el puerto serial (i.e. todo el código está dentro de un *if(serial.avaliable() > 0)*). Los periodos elegidos para los lectores fueron de un milisegundo, no con la intención de que efectivamente se ejecuten con ese periodo sino con el fin de que se ejecuten lo más rápido posible. La inicialización del Shield se hace en la función *setup()* por lo que no fue necesario hacer una máquina de estados porque en ese momento sólo se está haciendo eso. La tarea *Shield3GReader* es entonces la que se encarga de verificar si hay respuesta a los mensajes enviados al shield y su diagrama de flujo de puede observar en la Figura 5.3.

5.2.2. E-Mail

Para implementar la recepción y envío de correos electrónicos a través del Shield3G es necesario realizar una secuencia de comandos, por lo que se escribieron dos tareas, que, si bien podrían haber sido incluidas en el Shield3G y su Reader, fueron separadas para tener mayor claridad.

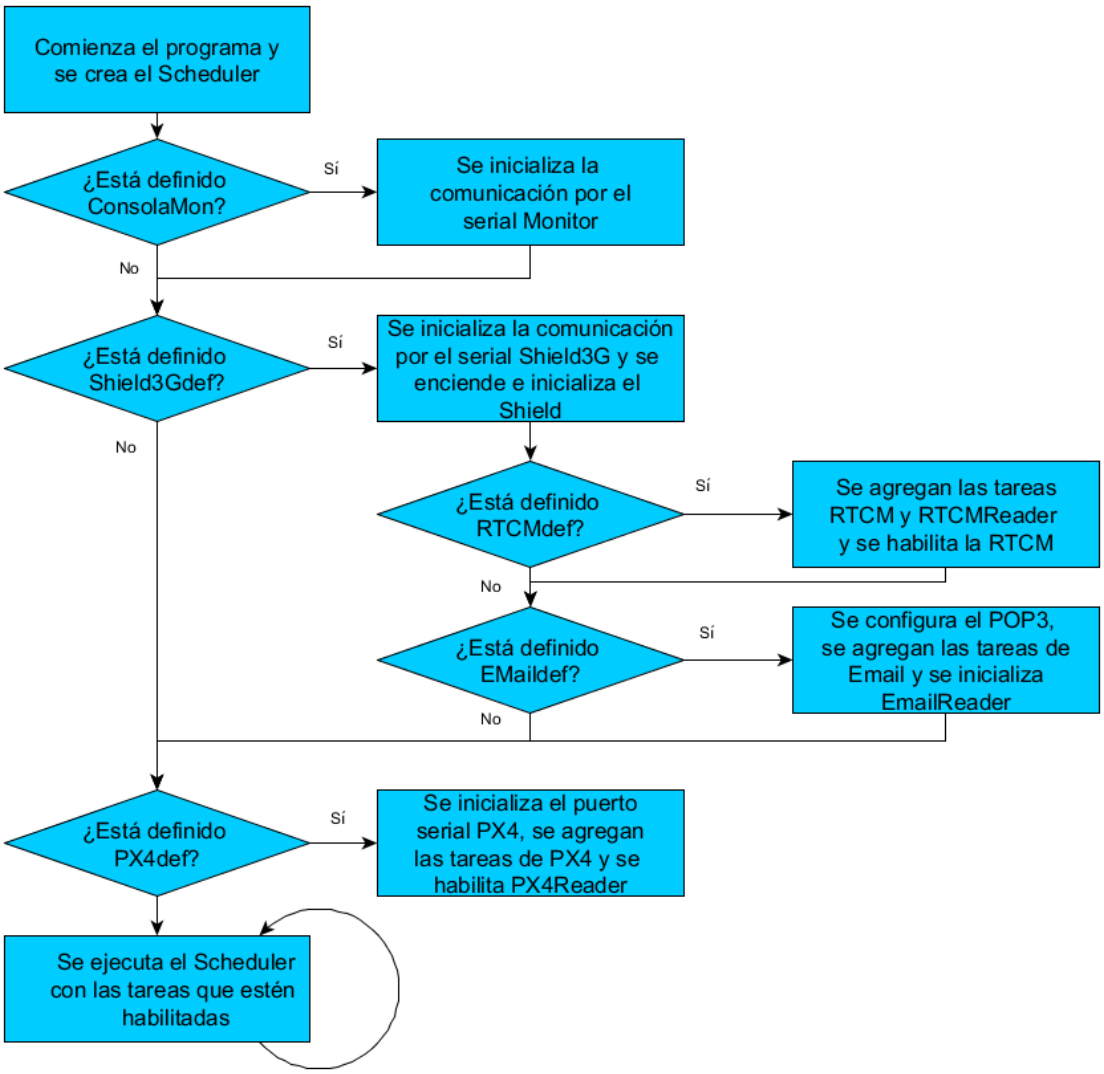


Figura 5.2: Diagrama de flujo del programa principal de la GCS.

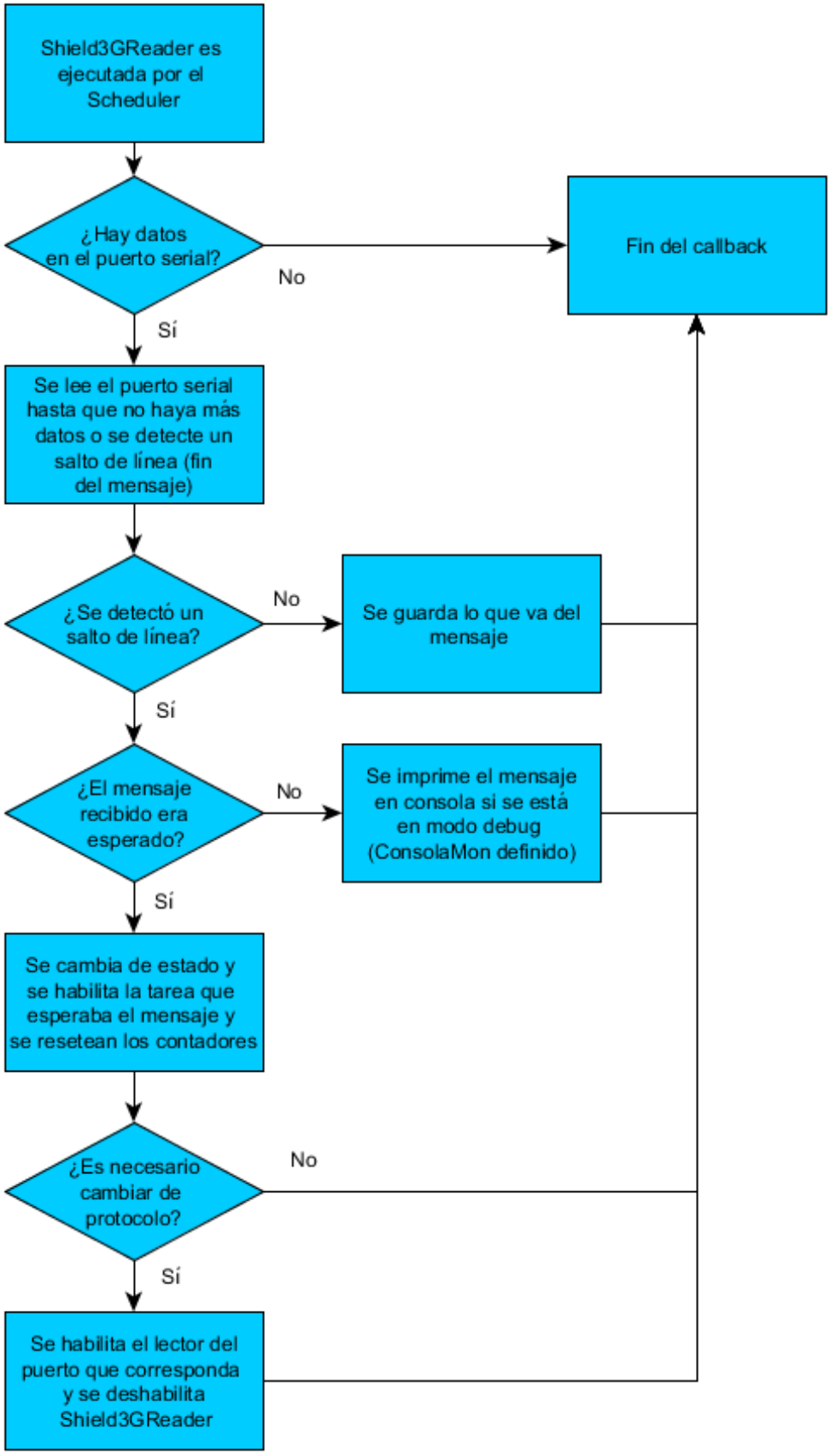


Figura 5.3: Diagrama de flujo de la tarea *Shield3GReader*.

Por un lado, está la tarea *checkEMail*, que en su callback (función que se ejecuta cada vez que se llama la tarea) implementa una máquina de estados. Estos estados fueron llamados POP3STATUS debido a que se utiliza el protocolo POP3 para obtener los correos electrónicos. Primero la tarea ejecuta el comando *AT+POP3IN*, se incrementa una variable que lleva la cuenta de cuántas veces se ejecutó este comando sin obtener respuesta, se inicializa un contador y se deshabilita la tarea, no volverá a habilitarse hasta que *Shield3GReader* reciba la respuesta esperada o los contadores superen el umbral seteado. Esta secuencia es la misma en todas las máquinas de estado de las tareas, cambiando solo el comando que se imprime en el puerto serial. Los parámetros necesarios para la conexión POP3 se setearon durante el *setup()* (i.e. servidor, usuario, contraseña y puerto). Luego se ejecutan los comandos necesarios para obtener el último correo recibido en la casilla, seteando sus respectivos contadores, deshabilitando la tarea y volviendo a ser habilitada y cambiada de estado por *Shield3GReader*, en el caso de salir por timeout no se cambia de estado para que se solicite nuevamente, luego de tres solicitudes se entiende caída la conexión y se deshabilita la tarea. En la Figura 5.4 se puede observar el diagrama de estados de esta tarea.

Mientras se está en el estado POP3_READING se sigue leyendo el puerto y guardando las partes de interés del correo en las variables globales *body* y *subject*. Una vez leído todo el correo se habilita la tarea *EMailReader* para efectivamente atender la solicitud del mail. La tarea *EMailReader* implementa un *switch - case* con el subject y dependiendo del mismo, escribe las variables y realiza las acciones de control necesarias, esto puede ser cargar una misión en el dron, solicitarle que aterrice, vaya a determinado waypoint o contestar al usuario con alguna información (i.e. estado, foto, bitácora de vuelo). Para más información acerca de los correos que un usuario puede enviar consultar el anexo “Manual de usuario”.

5.2.3. Transferencia de archivos al Shield

Para poder enviar archivos adjuntos al usuario (e.g. imagen de termografía), fue necesario implementar el protocolo Xmodem [44]. Como la interpretación de los bytes es diferente a la que se hace en *Shield3GReader*; se optó por leerlos en otra tarea, *XmodemReader*; y por enviar los datos en otra, *Xmodem*. De esta manera se logra segregar todo lo que implica Xmodem y resolver la comunicación de manera ordenada y coherente con el resto del código. Del lado del receptor hay solo tres mensajes, la solicitud de inicio de transferencia (0x43, o “C” por su representación en UTF-8), la confirmación de que el último mensaje fue recibido (0x06, o ACK por acknowledge) y la no confirmación del último mensaje (0x15 o NACK). Esta no confirmación puede darse porque el CRC está incorrecto, no se recibieron todos los bytes de un paquete o los bytes de encabezado son incorrectos. En las Figuras 5.5 y 5.6 se pueden observar los diagramas de estado de las tareas *Xmodem* y *XmodemReader* respectivamente.

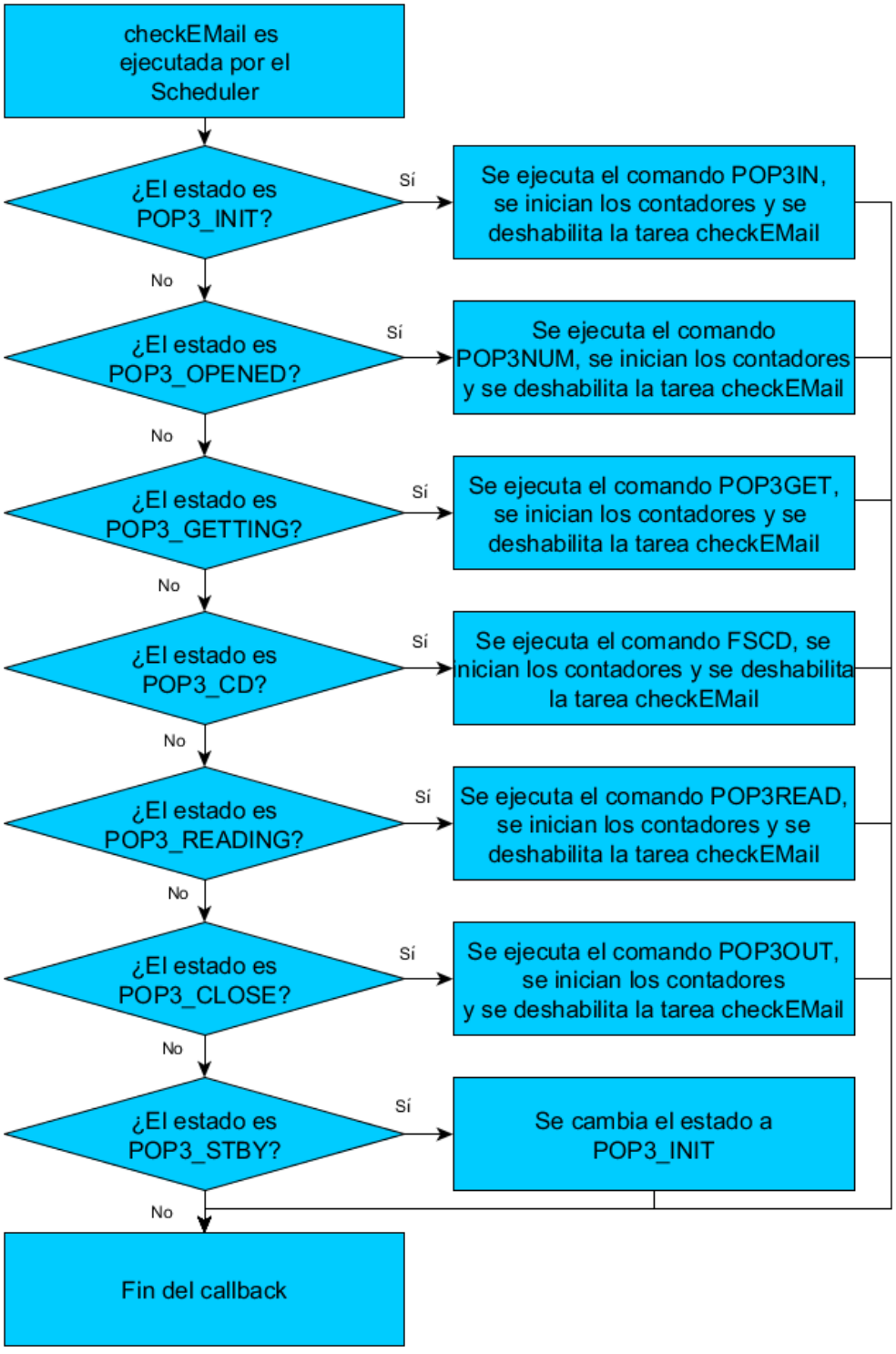


Figura 5.4: Diagrama de flujo de la tarea *checkEMail*.

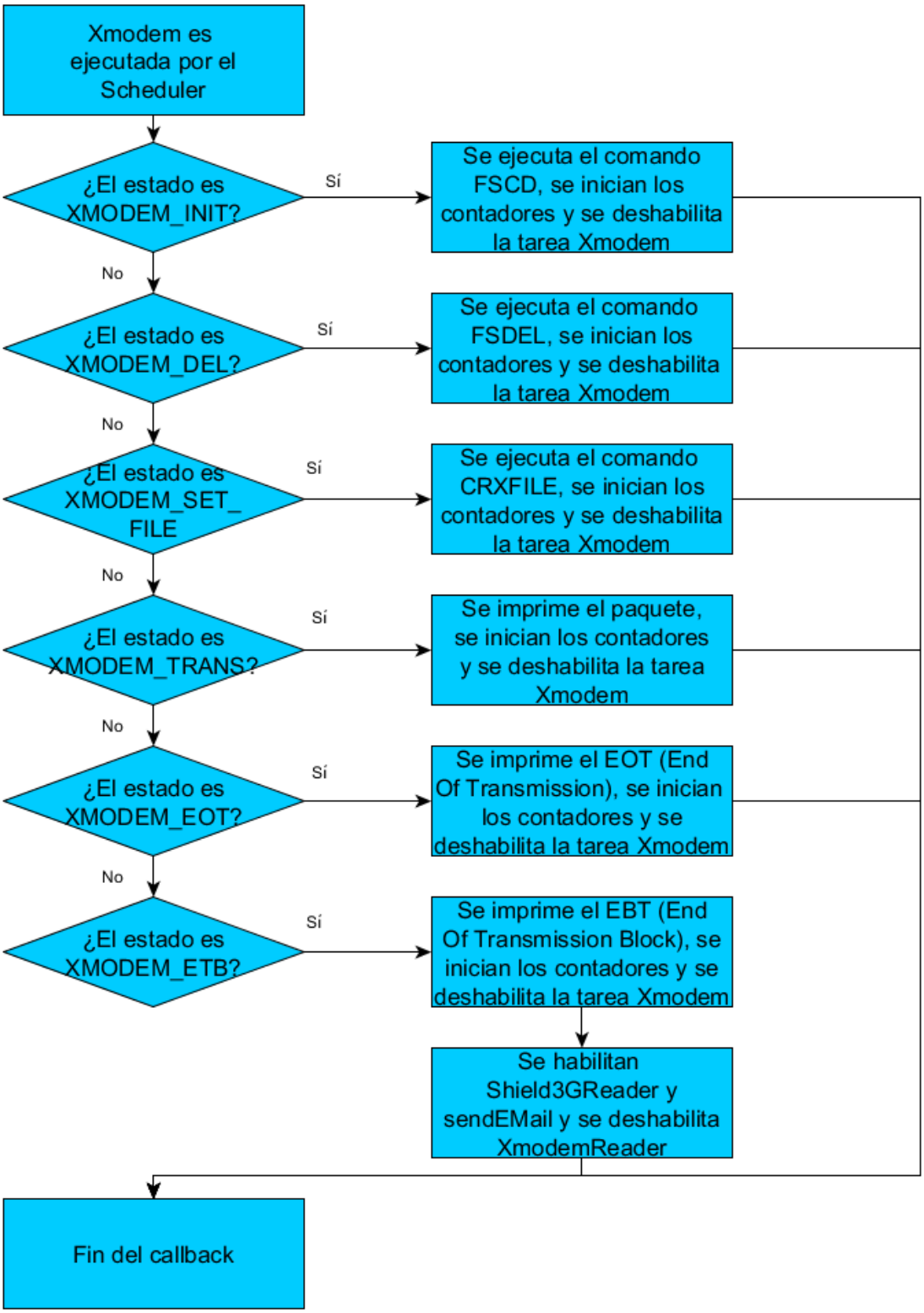


Figura 5.5: Diagrama de flujo de la tarea *Xmodem*.

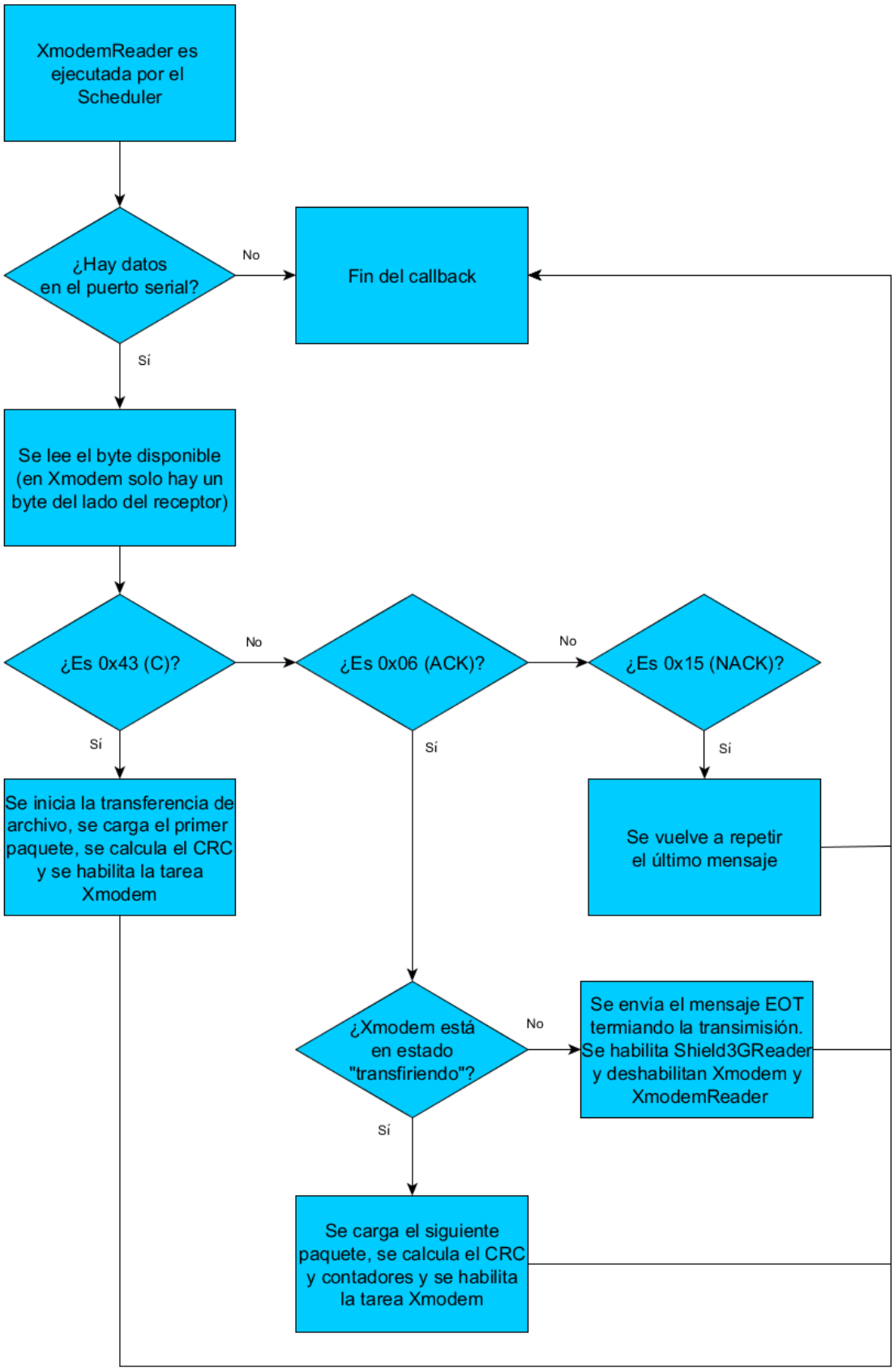


Figura 5.6: Diagrama de flujo de la tarea *XmodemReader*.

5.2.4. Cálculo de waypoints intermedios

El objetivo de una misión es relevar un área determinada en búsqueda de un cúmulo de puntos que sobrepasen determinado umbral de temperatura.
Dicha área está descrita por un perímetro que a su vez es un conjunto de vértices recorrido en determinado orden. Son estos vértices los que el usuario envía a través del correo con asunto "10_LoadArea".
Para poder relevar el área de interés en búsqueda de un punto caliente, se optó por realizar una trayectoria en *zig-zag* paralela al lado del polígono definido por los primeros dos puntos. Para que la misión sea válida, el resultado del algoritmo no debe devolver más de 40 puntos y el polígono debe ser convexo, también se limitó a 10 el número de puntos que definen el perímetro. En caso de detectar que la misión no es válida, se envía un correo al usuario con esta información y se le solicita que ingrese una nueva misión que implique menos puntos o que verifique la convexidad del polígono.
Para poder probar y depurar el algoritmo de cálculo de waypoints intermedios, este se desarrolló en Processing [45], lo que permite contar con un entorno gráfico tanto para ingresar los puntos como para visualizar el resultado del algoritmo.
La librería desarrollada implementa las clases Punto y Pendiente para resolver el problema.

5.2.5. Clase Punto

Punto tiene como atributos longitud, latitud, altura, un valor booleano que indica si cuando el punto fue creado resultó contenido en un intervalo o no y un número que indica la distancia que quedó pendiente de recorrer en la dirección del *zig-zag*.
Los constructores permiten instanciar un punto: dadas sus coordenadas, sobre la recta definida por otros dos a una distancia dada con respecto a uno de ellos, sobre la recta definida por otros dos y que el segmento definido por el nuevo punto y un cuarto punto dado sea paralelo a una pendiente también dada.

5.2.6. Clase Pendiente

La clase Pendiente tiene como atributo un número que representa la pendiente entre dos puntos y de esta manera está definido su constructor y también un valor booleano que indica si la pendiente es infinita (si la latitud de los dos puntos es igual).

5.2.7. Algoritmo implementado

El algoritmo recibe un conjunto de Puntos, Perímetro y devuelve otro, Misión. Se calcula la distancia que debe haber entre las paralelas de manera que cuando el dron recorra la trayectoria toda el área sea barrida por la cámara térmica, y de

manera de tener un margen de seguridad, se utilizó un solapamiento de un 20 % entre las imágenes. Si el dron se encuentra a una altura h, teniendo en cuenta que la cámara tiene un ángulo de apertura de 51°, la distancia entre las paralelas debe ser:

$$D = 1.8 \tan\left(\frac{51^\circ}{2}\right) h$$

Para una altura de 10 metros, la separación es aproximadamente de 8.6 m.
Se implementó una interfaz gráfica donde el usuario puede ingresar los puntos mediante clicks del mouse en una ventana y una vez finalizado hacer clic en el botón "Enviar Misión". Una vez hecho esto, se ejecuta el algoritmo y se calculan los waypoints intermedios. El diagrama de flujo del algoritmo se puede observar el la Figura 5.7 y la GUI del programa hecho en Processing en la Figura 5.8

5.2.8. Raw14 a mapa de bits

Los datos obtenidos de la cámara térmica están en un formato de 14 bits conocido como Raw14, este formato es interpretado como un número proporcional a la diferencia de temperatura entre el pixel que está midiendo y la temperatura de la cámara. Cuando la temperatura medida en el pixel es igual a la temperatura de la cámara, el valor está en el medio del rango de 14 bits ($\sim$ 8192).
Como primer paso, se convirtió el dato de Raw14 a temperatura en grados Celsius, para esto se utilizó el modelo:

$$T = aRaw14 + bT_{cam} + c$$

De la calibración se obtuvieron los valores a = 0.0327, b = 27.637 y c = -885.39.
Se utilizó el formato RGB de 3 bytes para formar la imagen, los dos bytes más significativos representan el contenido de rojo en el pixel, los dos siguientes el verde y los últimos el azul.
Para pasar de temperatura a RGB se designaron umbrales mínimos y máximos de temperatura (0 $°C$ y 40 $°C$), al mínimo y a todos los valores por debajo del mismo se les asignó el verde (0,255,0), al máximo y a todos los valores por encima de este se les asignó el rojo (255,0,0), al punto medio se le asignó el amarillo (255,255,0) y se interpoló linealmente en los intervalos intermedios. Esto es equivalente a moverse por las aristas verde-amarillo y amarillo-rojo del Cubo RGB, el mapeo se puede observar en la Figura 5.9. En la Figura 5.10 se puede observar el resultado de una imagen tomada en el laboratorio.
Finalmente, se crea un archivo BMP con los valores RGB obtenidos. Para esto fue necesario estudiar el formato de archivo BMP y crear un *template* en el programa de manera de poder adjuntarle los bytes de la imagen y generar el archivo.
También se implementó un programa en Processing para probar esta transformación y luego se pasó a Arduino una vez depurado.
Se probaron distintas combinaciones de colores pero finalmente se optó por el verde-amarilo-rojo, en la imagen 5.11 se pueden observar algunas posibles combinaciones.

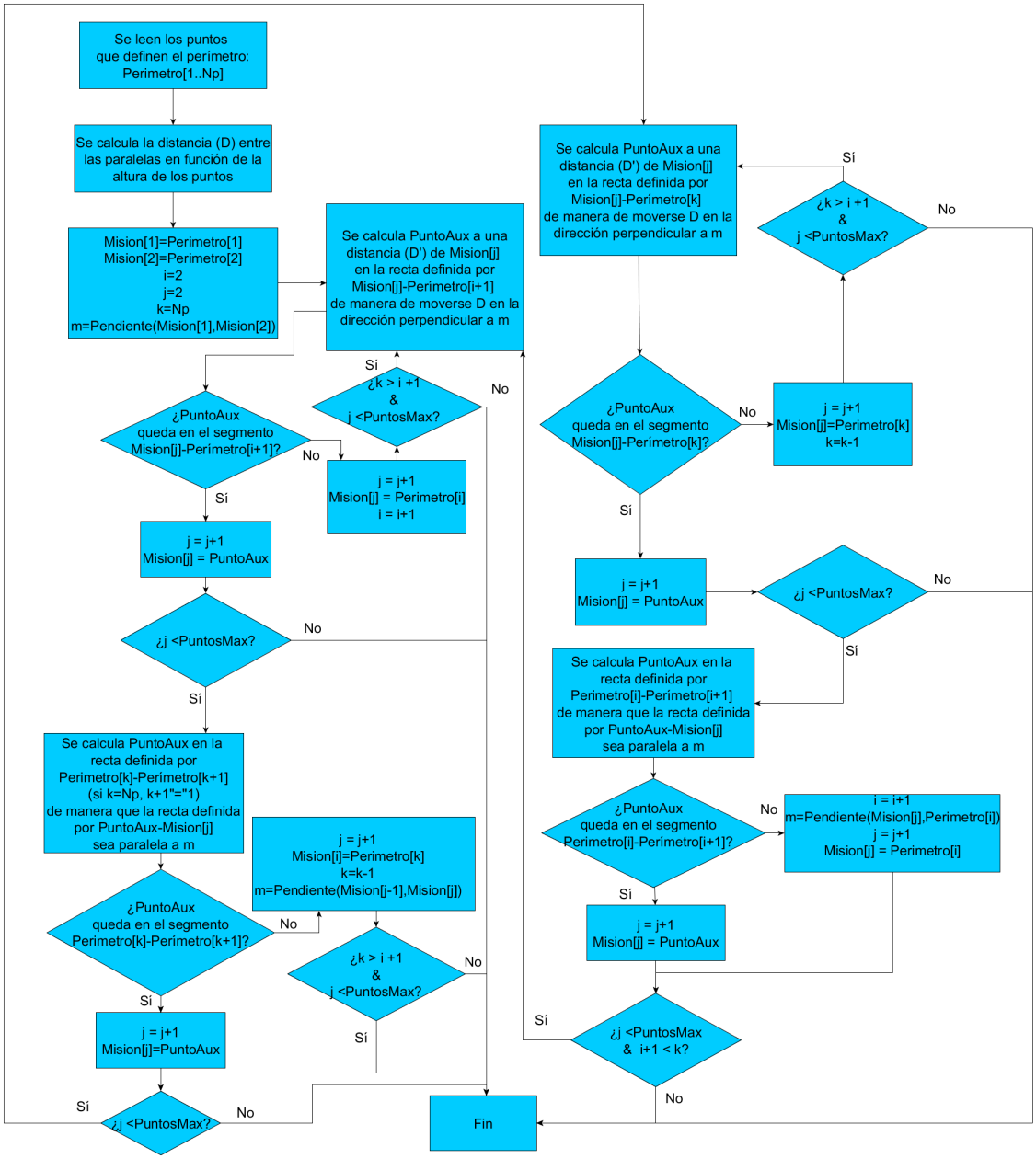


Figura 5.7: Cálculo de waypoints intermedios.

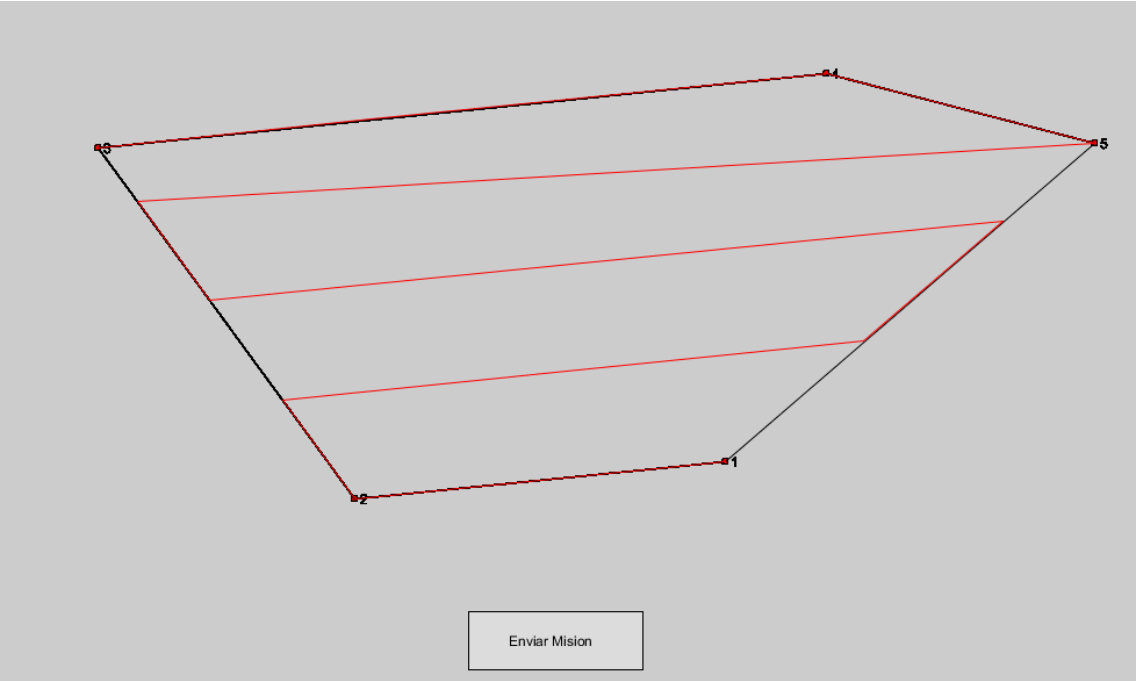


Figura 5.8: Ejemplo de la GUI que calcula los puntos intermedios en Processing. En negro se puede ver el perímetro y en rojo el recorrido de la misión.

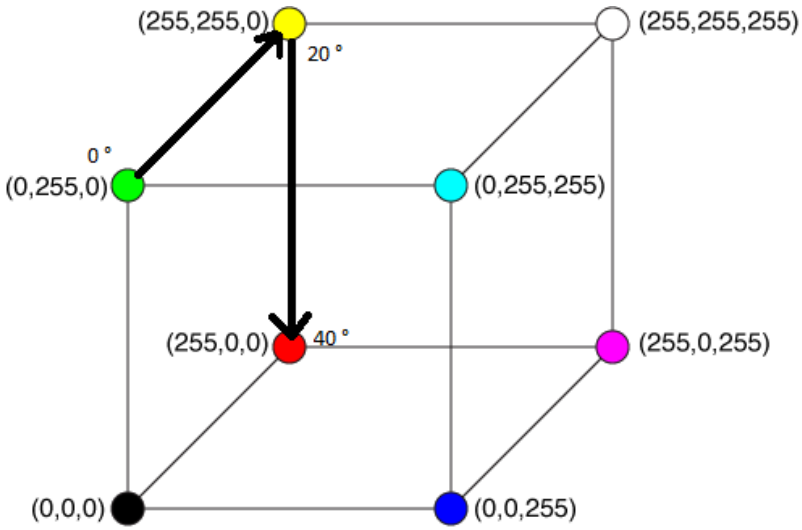


Figura 5.9: Representación del mapeo temperatura-color en el Cubo RGB. Imagen obtenida de [46]

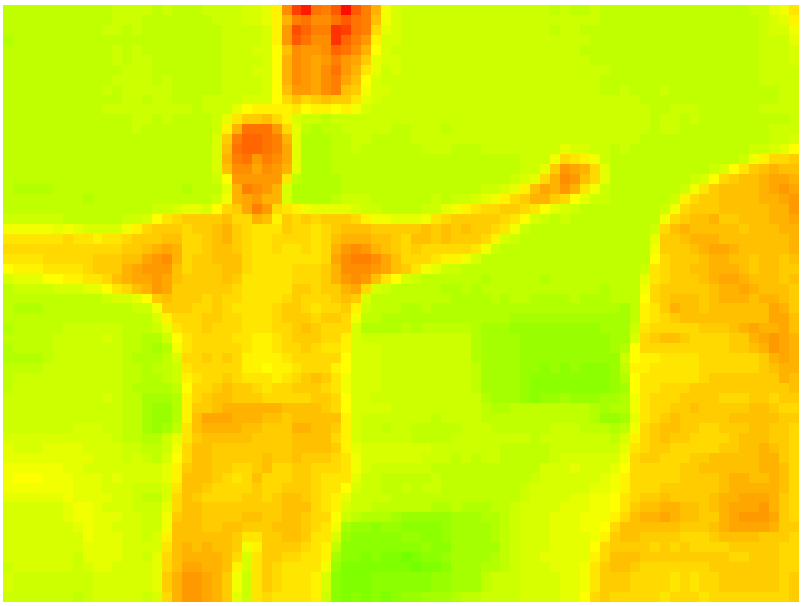


Figura 5.10: Imagen de termografía resultado del pasaje de Raw14 a BMP (80x60 px).

Figura 5.11: Distintas opciones de colores para la imagen térmica formando una imagen al estilo *Pop Art*.

5.3. Implementación de comunicación con Pixhawk

5.3.1. Heartbeat

Una característica de este protocolo es el mensaje Heartbeat. Este mensaje es enviado periódicamente por todos los sistemas involucrados para dar cuenta de que están activos y es esperado por el resto para saber que el sistema que lo envió está activo. Llevando cuenta del Heartbeat se puede saber si alguna de las placas se detuvo o si hay problemas en la comunicación (aunque hay maneras más sofisticadas de evaluar la calidad de la comunicación, como la SNR).

El mensaje Hearbeat permite tomar acción ante su ausencia pero a su vez genera la dificultad de enviar un mensaje periódicamente desde Arduino. Para esto se utiliza la librería TaskScheduler en Arduino, la cual permite implementar esta funcionalidad. Esta librería también se utilizará para solicitar datos de manera periódica a Pixhawk por la GCS para saber el estado de aquella y armar una bitácora de misión, enviar reportes al usuario y cualquier otra tarea que deba realizarse de manera periódica. También se utiliza esta librería para segmentar el loop principal en ambas placas Arduino de manera de que, en lugar de tener un loop corriendo continuamente a la máxima velocidad posible, se tenga una serie de tareas con distintos periodos y prioridades. Con esto se logra exigir lo menos posible al procesador, tener los datos más relevantes más seguido y en definitiva un mayor control sobre los recursos de la placa.

5.3.2. Mensajes secuenciales y Mission Protocol

Si bien el mensaje Heartbeat es periódico y además muchas de las variables de estado del dron se pueden solicitar de manera periódica, hay mensajes que deben ser enviados y recibidos secuencialmente. Es así el caso del Mission Protocol.

El Mission Protocol describe cómo los waypoints se envían y leen de una UAV. El objetivo es garantizar un estado coherente entre el emisor y el receptor. El protocolo consiste en distintas transacciones, cada transacción se ha completado con éxito o no de una manera que el estado anterior de la lista de waypoints en el lado receptor se modifica. Una transacción puede ser iniciada con un mensaje específico sólo cuando ninguna otra transacción está activa entre las dos partes de comunicación. Esto significa que para iniciar una transacción de comunicación ambas partes tienen que estar en estado de reposo.

Después de cada mensaje enviado que exceptúa una respuesta el componente emisor inicia un temporizador. Si no se recibe ninguna respuesta hasta que haya transcurrido un tiempo determinado el mensaje de petición se envía de nuevo. El reenvío se vuelve a intentar varias veces, si no hay respuesta después del último tiempo de espera de reintento, la operación ha fallado. El mecanismo de reintentar significa que todos los componentes tienen que ser capaces de manejar los mensajes duplicados.

Para obtener la lista de waypoints de Pixhawk se debe solicitar la misma mediante el mensaje MISSION_REQUEST_LIST, al cual Pixhawk responde con MISSION_COUNT que es la cantidad de waypoints. Una vez recibida esta información, el solicitante pide el primer waypoint con el mensaje MISSION_REQUEST y Pixhawk lo envía con el mensaje MISSION, esto se repite hasta el último waypoint. Al

recibir el último waypoint el solicitante debe enviar el mensaje MISSION_ACK con el resultado de la transacción. Esta secuencia se puede obervar en la Figura 5.12. Para cargar una misión o un waypoint independiente la secuencia es similar. Este tipo de comunicación difiere de los mensajes periódicos y debe ser tratado distinto, si bien el inicio de la transacción puede ser periódico, la transacción en sí es secuencial.

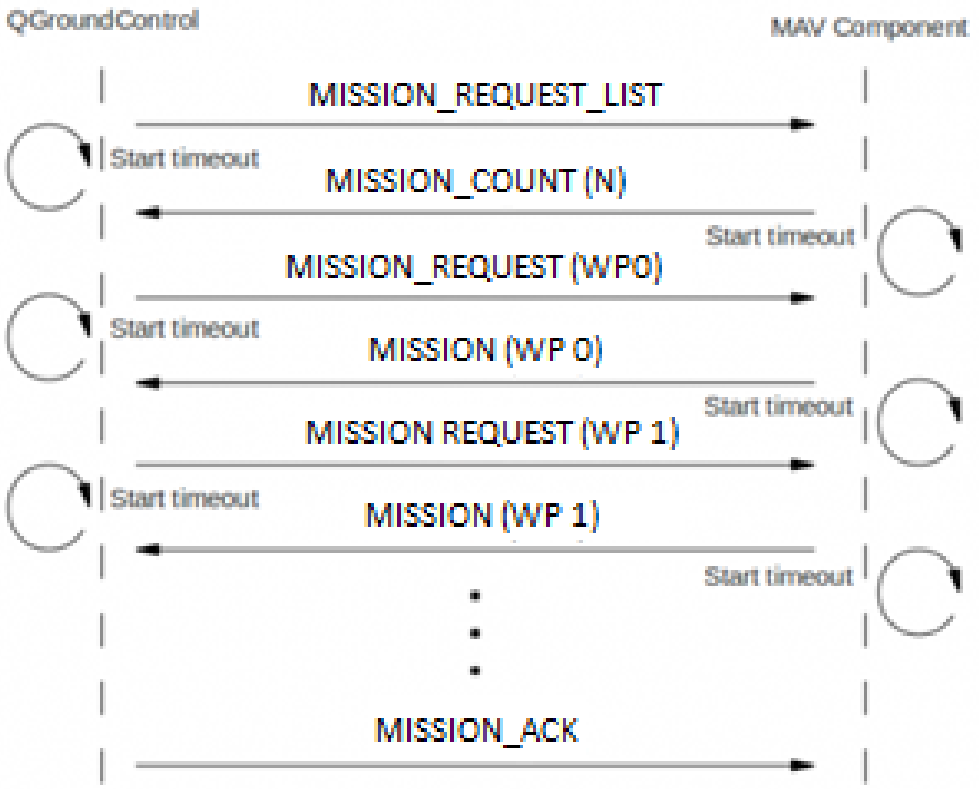


Figura 5.12: Secuencia de mensajes al leer una misión. Imagen obtenida de [47]

5.4. Comunicación GCS y usuario

Se implementa la comunicación entre la GCS y el Usuario a través de correo electrónico con el fin de enviar reportes en función de la información entregada por el Dron o bien ejecutar comandos enviados por el Usuario.

Dicha comunicación se realiza a través del Shield 3G Itead conectado a la SBC que controla la GCS. La comunicación entre el Shield 3G y la placa Arduino se realiza mediante comandos AT[1] lenguaje entendido por el shield. Esto permite el seteo y la ejecución de acciones como por ejemplo enviar un correo electrónico a través de la red 3G. El shield cuenta con un integrado SIM5216E y de una SIM para la autenticación en la red.

La comunicación entre las placas se realizarán a través del puerto serie N° 3 de la Arduino (puertos 14 y 15, Tx y Rx respectivamente) y los puertos D0 y D1 del shield 3G (Tx y Rx respectivamente) tal como se puede observar en la Figura 5.13.

[1]https://es.scribd.com/document/263634315/Simcom-Sim5215-Sim5216-Atc-en-v1-21

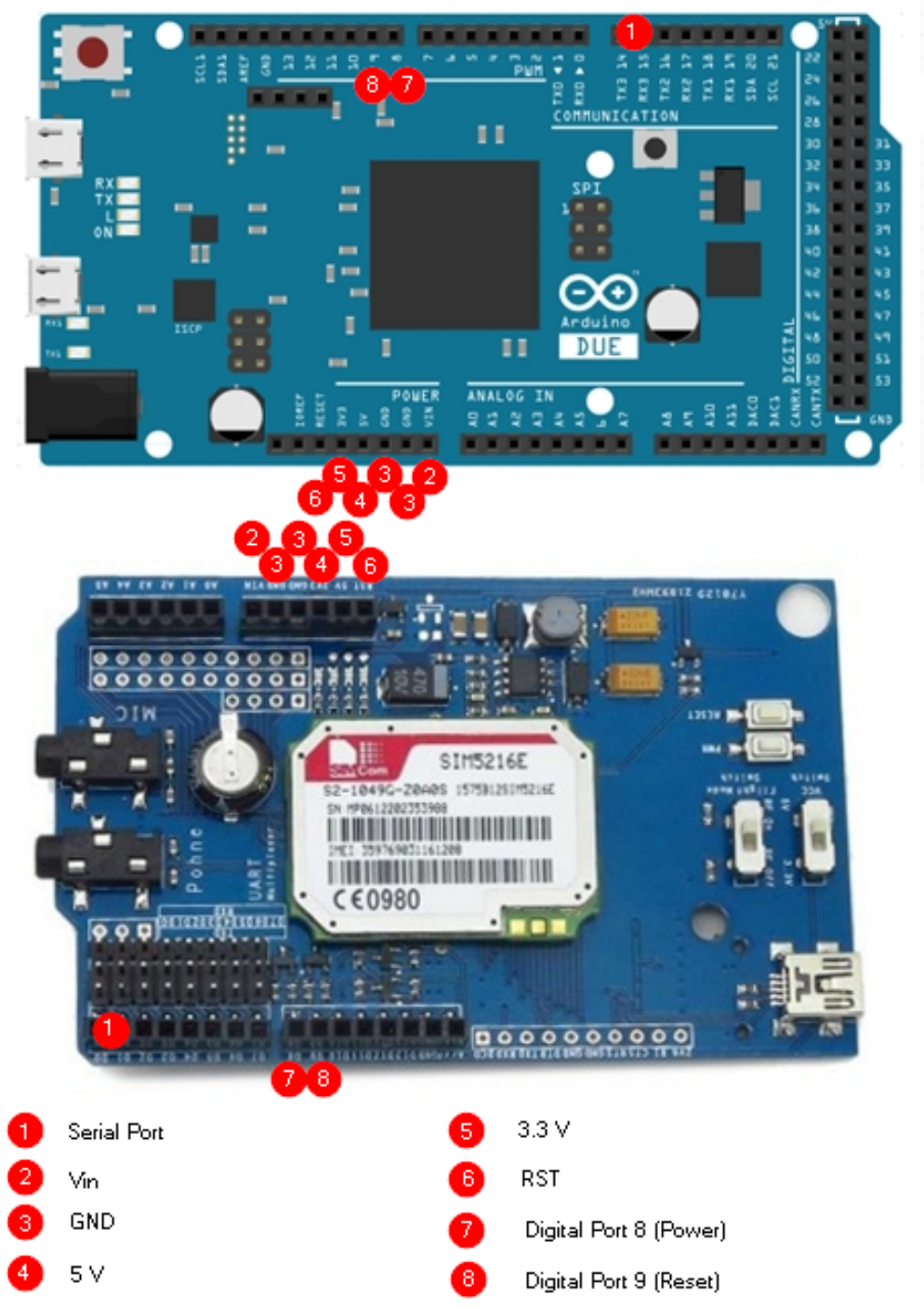


Figura 5.13: Conexionado Arduino DUE con Shield 3G.

Por otra parte, se conectan los puertos V_{in}, 5 V, 3.3 V, GND y RST entre ambas tarjetas.

Para el envío de correo electrónico fue necesario contar con una cuenta de correo y una SIM con un servicio telefónico y de datos (3G).

El equipo utilizado permite también realizar llamadas, video llamadas, grabar y reproducir audio y transferir archivos de las cuales no se hacen uso en este proyecto.

Otra característica de interés de esta placa es la conexión a internet lo cual es utilizada para obtener en tiempo real la información otorgada por la REGNA-ROU para el funcionamiento de GPS Diferencial. El mismo funciona utilizando HTTP sobre una conexión TCP. La misma se detalla en el anexo.

CAPÍTULO 6

ENTORNO DE DESARROLLO

6.1. Entorno de desarrollo

6.1.1. Piloto automático PX4

El firmware que corre sobre Pixhawk, producto del proyecto PX4, implementa un piloto automático y es un proyecto de software libre bajo licencia BSD[1] desarrollado por un equipo internacional y multidisciplinario. No solo está orientado a Pixhawk, sino que también a otras placas controladoras como lo son Snapdragon Flight, Pixfalcon, Intel Aero y Crazyflie [48].

PX4 consiste en dos capas principales; PX4 flight stack, una solución de piloto automático; y el middleware, conjunto de drivers para sensores embebidos y lógica de intercambio basada en publish-subscribe, para conectar estos sensores a las aplicaciones corriendo los controles de vuelo.

El uso del esquema publish-subscribe significa que el sistema es reactivo (se actualiza instantáneamente cuando hay datos nuevos), corre totalmente paralelizado y cualquier componente del sistema puede consumir información de cualquier parte del sistema de manera segura.

Al separar el control de vuelo del hardware completamente, se cuenta con una colección de algoritmos de navegación, guía y control para drones autónomos, ya sean de ala fija, despegue vertical o multicópteros.

6.1.2. Sistema Operativo en tiempo real NuttX

NuttX [49] es un sistema operativo a tiempo real (RTOS) sobre el cual corren los procesos (tasks) que conforman el piloto automático. Pixhawk cuenta con un puerto serial en el cual se tiene acceso a la consola de este sistema operativo, pero se maneja principalmente a base de scripts que están cargados en memoria.

Al bootear el sistema, se corre un script llamado rcS el cual inicializa la placa, sus puertos de comunicación, drivers de sensores, LEDs y los principales procesos que implementan el vuelo autónomo.

[1] Berkeley Software Distribution

6.1.3. Simulador jMAVSim

Si se está desarrollando un piloto automático, es de mucha utilidad contar con un simulador que pueda emular el comportamiento en vuelo de la aeronave. De esta manera se pueden detectar errores sin poner en riesgo el equipamiento sobre el cual correrá el firmware depurado. En el marco del proyecto PX4, se desarrolló un simulador de cuadricóptero, jMAVSim [50]. Este simulador implementa en una interfaz gráfica como la que muestra la Figura 6.1, en la cual se puede observar el comportamiento del dron bajo dos modalidades:

Software In The Loop (SITL)

El tipo de simulación SITL emula el sistema completo, placa y cuadricóptero. En este proyecto no se realizaron este tipo de simulaciones ya que se optó por HITL que es más fiel a la realidad.

Hardware In The Loop (HITL)

En HITL, el firmware corre sobre la placa y la información de los sensores es simulada. Con esto se pueden hacer pruebas en el entorno virtual, pero observando cómo responde la placa a las variaciones del firmware, comunicaciones con la GCS, comunicación con la placa on-board y al control remoto. Esta herramienta resultó muy útil para verificar que partes del sistema final se comuniquen correctamente.

6.1.4. QGroundControl

QGroundControl (QGC) [51] es una interfaz gráfica para el usuario (GUI) para monitorear el estado del dron, que a través de una antena establece comunicación con este e implementa una GCS. Con esto se puede, por ejemplo, cargar una misión al dron, tener información de los sensores, ver en un mapa satelital la posición actual del dron y otras funcionalidades. En la Figura 6.2 se puede observar la ventana de esta interfaz.

QGC es un proyecto independiente de PX4 y busca ser compatible con cualquier piloto automático que implemente el protocolo MAVLink.

A través de QGC se pueden establecer datos específicos de cada placa controladora, por ejemplo: qué tipo de aeronave se está utilizando, capacidad de la batería y constantes de calibración de los sensores. Para este proyecto en particular, QGC fue además utilizada para verificar que las misiones cargadas desde Arduino fueran interpretadas correctamente por el dron y verificar que en las simulaciones HITL el dron pasara por los puntos asignados.

Figura 6.1: Entorno gráfico de jMAVSim.

6.1.5. Plataforma IDE de Arduino

El IDE de Arduino es la plataforma que permite el desarrollo, compilación y carga de un código para productos Arduino. Tanto para la tarjeta on-board como para la GCS, todo el desarrollo fue implementado sobre esta plataforma y debugueado en la misma.

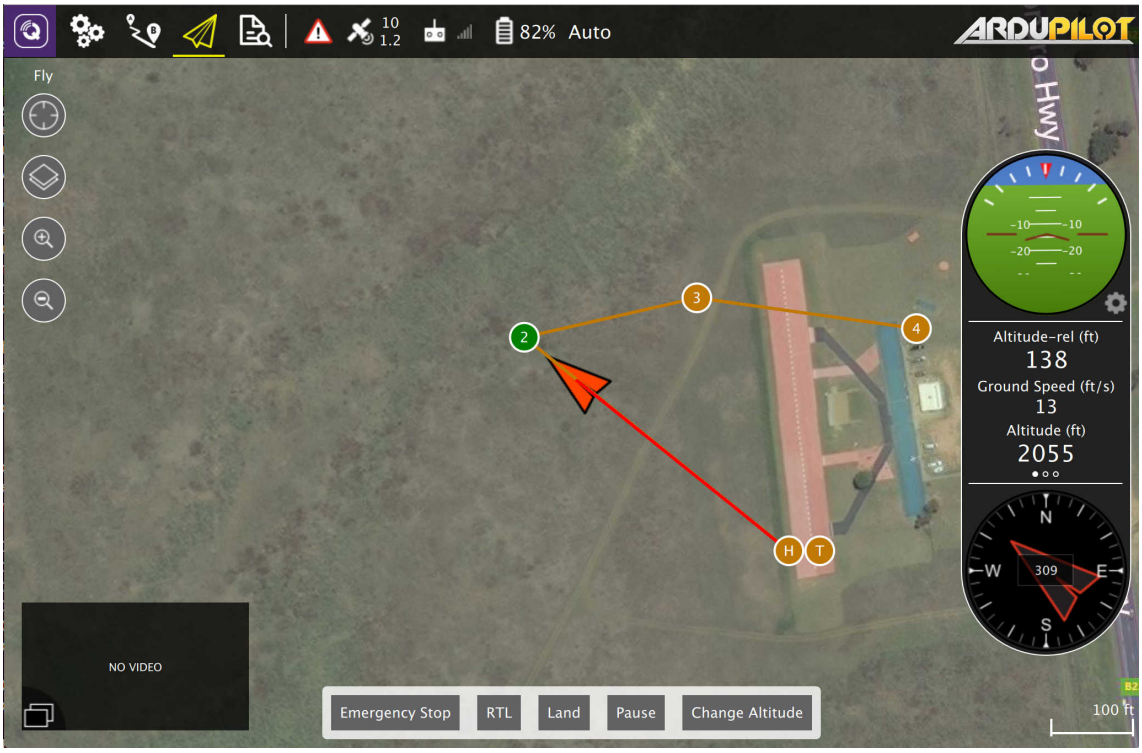


Figura 6.2: Ventana de la GUI QGC. Imagen obtenida de [52]

CAPÍTULO 7

PRUEBAS

7.1. GNSS vs DGNSS (estático y dinámico)

7.1.1. Descripción de las pruebas

Para realizar las pruebas es necesario situarse en una zona con una buena visibilidad de satélites (∼10). Para la prueba estática se sitúa el Dron en un lugar fijo y se registran medidas de posición con y sin corrección diferencial. Para la prueba dinámica se debe efectuar un recorrido con el Dron y registrar medidas al igual que en el caso estático. Es importante que tanto los puntos como el recorrido estén bien definidos de manera de poder repetir la prueba y definir sus coordenadas con mayor exactitud en caso de contar con un geolocalizador más exacto que el que se utilizó en la prueba. También es importante notar que la comparación que se hace en esta prueba no es entre GNSS y DGNSS puramente, ya que la posición que reporta el Dron es la salida de un filtro de Kalman extendido, la cual es fusión entre la posición dada por su receptor de GNSS y los demás sensores de Pixhawk. Esto es importante ya que este filtro es más preciso que solo un receptor de GNSS.

7.1.2. Desarrollo de las pruebas

Las pruebas se realizaron en el estacionamiento sur del edificio de la Facultad de Ingeniería de la UdelaR. Se verificó que la cantidad de satélites visibles por el Dron fue 13 en el punto alejado de la puerta y 6 en el punto en la puerta según QGC. Se eligió un punto en la puerta de manera de no contar con tan buena visibilidad de satélites y poder apreciar cómo esto afecta la precisión de las medidas. En la Figura 7.1 se muestra el recorrido realizado en las pruebas GNSS dinámico. De manera de poder reproducir la prueba y definir con exactitud los puntos, se utilizó la línea que separa los paños del estacionamiento. Se recorrió la misma de principio a fin y se utilizaron sus dos extremos para tomar las medidas estáticas. Se enterraron pernos metálicos en los puntos para identificarlos y personal del instituto de Agrimensura midió con mayor exactitud (12 cm de desviación en latitud y longitud) las coordenadas de los mismos. De esta manera se puede medir no solo la precisión sino también la exactitud de los distintos métodos.

Figura 7.1: Recorrido realizado en la prueba de GNSS vs DGNSS. Imagen obtenida de Google Maps.

GNSS estático

Para el caso estático se tomaron medidas en los dos extremos de la línea que separa los paños del estacionamiento, uno en la puerta de emergencia del instituto y el otro en el extremo opuesto. Se situó el Dron en cada punto y se tomaron medidas con y sin corrección diferencial por aproximadamente dos minutos en todos los casos. Los resultados de estas medidas se pueden observar en las Figuras 7.2 y 7.3. El origen de coordenadas en cada gráfica es el promedio de los promedios de las muestras y la distancia está estimada con la aproximación rectangular[1].

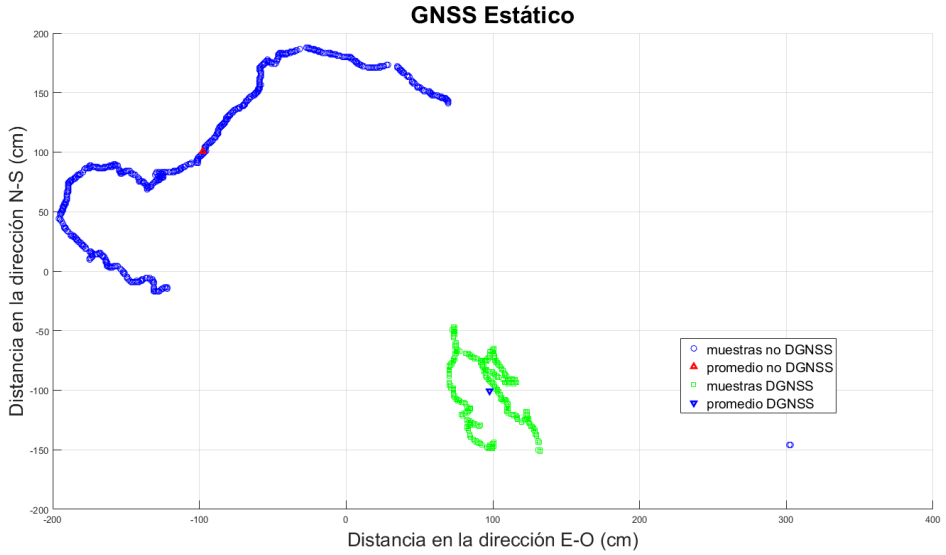


Figura 7.2: GNSS estático en el punto más alejado de la puerta.

De manera de poder comparar ambos resultados, se observó la varianza en metros de las medidas, la evolución en el tiempo de dicha distancia y el error con respecto a la medición tomada por Agrimensura.

[1] $\Delta_{N-S} = \Delta_\phi R$; $\Delta_{E-O} = \Delta_\lambda \cos(\phi_m) R$ donde R es el radio de la tierra, ϕ es latitud, λ es longitud y ϕ_m es la latitud promedio.

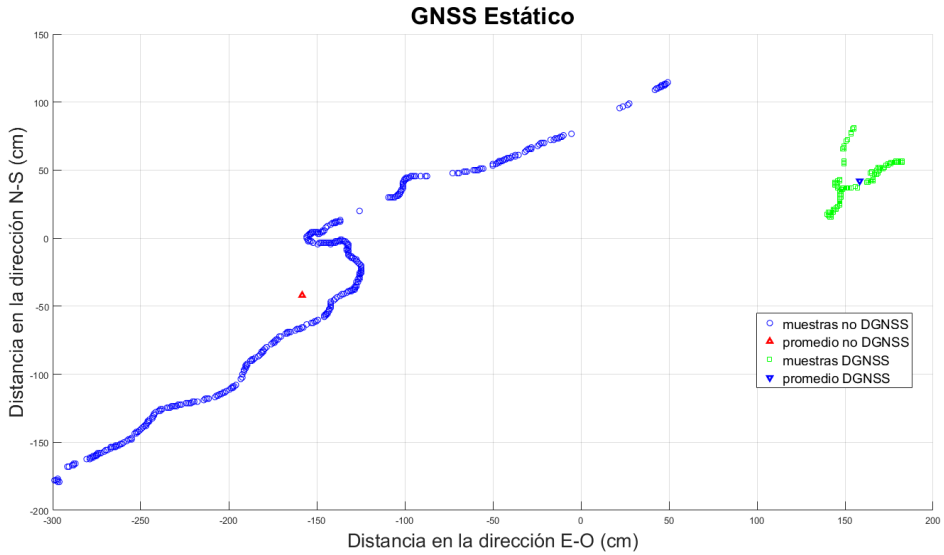


Figura 7.3: GNSS estático en la puerta.

Los resultados que se obtuvieron se pueden observar en la Tabla 7.1.

La evolución de los errores con respecto a los promedios en el tiempo se pueden observar en las Figuras 7.4 y 7.5.

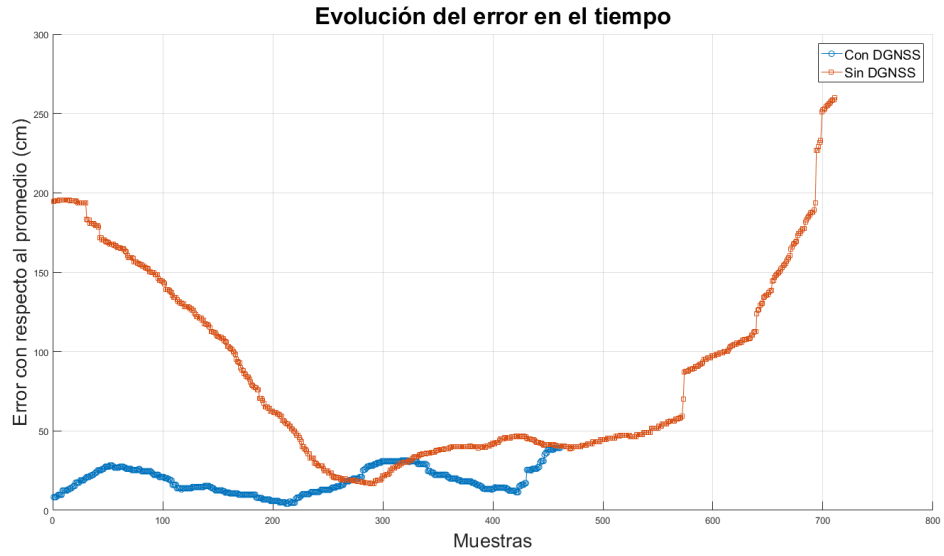


Figura 7.4: Evolución del error con respecto al promedio en la puerta.

GNSS dinámico

Para el caso dinámico se tomaron medidas realizando el recorrido que une los puntos definidos anteriormente. Se tomó el Dron y se trasladó caminando por encima de la línea que separa los paños del estacionamiento. Se realizaron dos recorridos, con y sin corrección diferencial. Los resultados de estas medidas se pueden observar en las Figuras 7.6 y 7.7.

De manera de poder estimar la precisión de las medidas, se midió la dispersión con respecto a la recta que mejor aproxime a las muestras. En el caso sin corrección

	Latitud (°)	**Longitud (°)**	**Dispersión con respecto al promedio (cm)**	**Error con respecto al punto medido por Agrimensura (cm)**
Punto alejado sin corrección diferencial	-34.9192671	-56.1668619	83	339
Punto alejado con corrección diferencial	-34.9192852	-56.1668405	29	61
Punto alejado medido por el instituto de Agrimensura	-34.9192900	-56.1668373	N/A	N/A
Punto en la puerta sin corrección diferencial	-34.9192440	-56.1662328	87	495
Punto en la puerta con corrección diferencial	-34.9192365	-56.1661981	19	312
Punto en la puerta medido por el instituto de Agrimensura	-34.9192617	-56.1661829	N/A	N/A

Tabla 7.1: Posiciones estimadas por los distintos métodos en la prueba

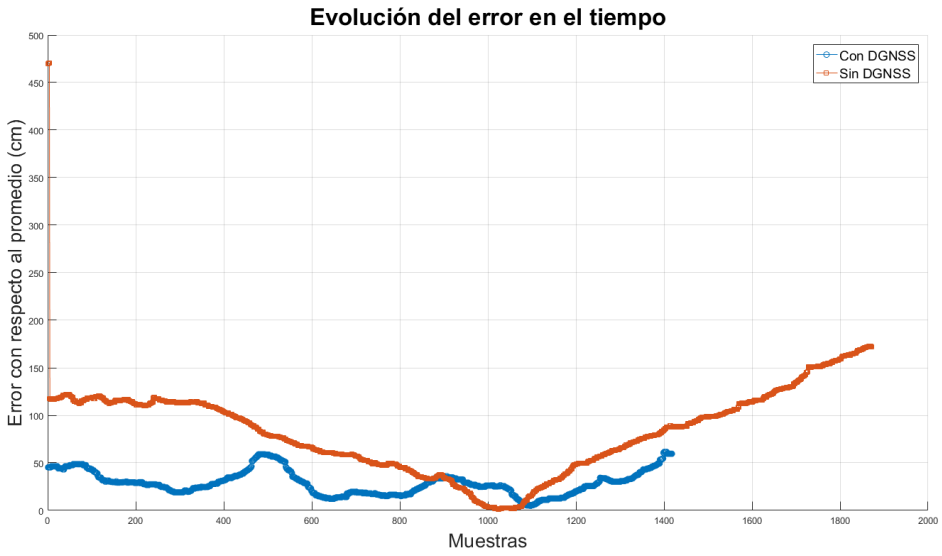


Figura 7.5: Evolución del error con respecto al promedio en el punto alejado de la puerta.

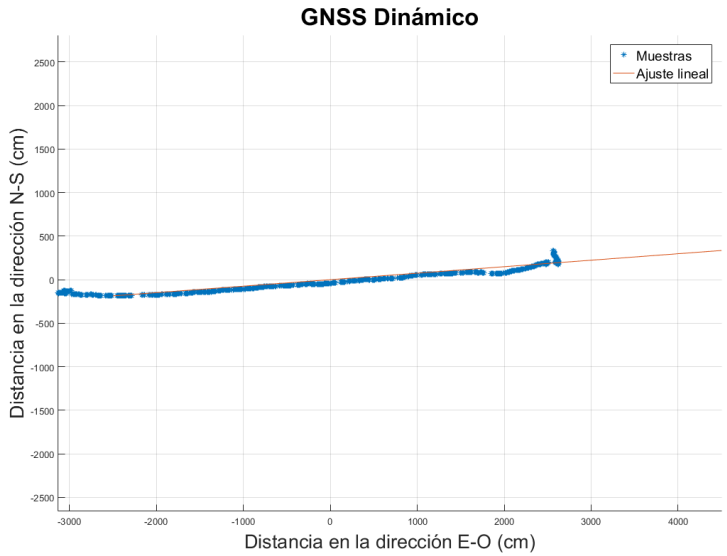


Figura 7.6: Prueba dinámica con corrección diferencial - Recorrido.

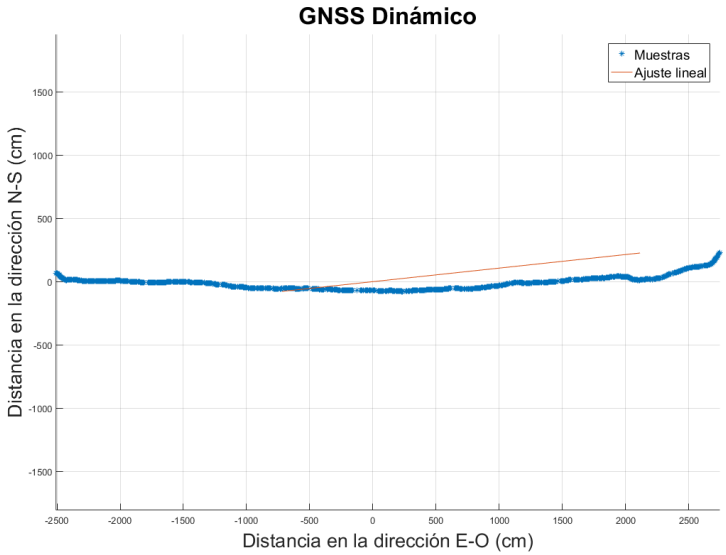


Figura 7.7: Prueba dinámica sin corrección diferencial - Recorrido.

diferencial, la distancia promedio con respecto a la recta fue de 155 cm y en el caso con corrección diferencial fue de 41 cm. También se relevó la evolución del error en el tiempo para ambos casos y los resultados se pueden observar en las Figuras 7.8 y 7.9

Figura 7.8: Prueba dinámica con corrección diferencial - Evolución del error en el tiempo.

Conclusiones

Se pudo observar una notoria mejora con respecto a la dispersión en la estimación de la posición del Dron, tanto en el caso estático como en el dinámico. También se pudo observar una mejora en la exactitud de las medidas en el caso diferencial. Se observó que en el punto con menor visibilidad de satélites empeora la calidad de las

Figura 7.9: Prueba dinámica sin corrección diferencia - Evolución del error en el tiempol.

medidas. No se pudo observar ninguna tendencia temporal en el horizonte de tiempo que se utilizó para relevar las medidas, aunque esto no descarta que pueda haber alguna mejora si se reciben correcciones diferenciales por un periodo más prolongado de tiempo. Como los mensajes RTCM brindan información de varios satélites, es posible que en el periodo que se estudió no todos los satélites visibles por el Dron hayan sido corregidos. Al no tener las medidas de GNSS *raw*, la comparación se vio afectada por el filtro de Kalman que implementa Pixhawk.

7.2. Vuelo autónomo

Las pruebas de vuelo autónomo deben realizarse con cuidado ya que una falla durante el vuelo puede ocasionar daños irreparables en el Dron. Para esto se desarrolló un procedimiento de seguridad que se detallará a continuación. También toda prueba que se realice se hará primero en el simulador jMAVSim con Pixhawk en modo HITL.

Chequeos pre-vuelo

El propósito de este procedimiento previo a cada vuelo es minimizar la probabilidad de falla durante el vuelo, es de suma importancia tener bien definido qué hacer antes y después de cada vuelo para evitar daños físicos y materiales.

Siempre:

- Los motores deben estar firmes. Sujetar la base e intentar moverlos, si se notan sueltos ajustar los dos tornillos que fijan el estator a la base. Repasar el ajuste de los tornillos, con cuidado de no desgastarlos.
- También las hélices deben estar firmes, verificar y ajustar de ser necesario. Para verificar la firmeza de las hélices sujetar el estator y tratar de moverlas.

- La batería debe estar cargada, verificar con QGroundControl, voltímetro o mediante el mensaje 50_GetStatus a la casilla termodron1@montevideo.com.uy

Si se hizo algún cambio en los motores, hélices, cableado, etc.:

- Se debe verificar que las hélices estén correctamente instaladas. Dos hélices giran en sentido anti-horario (1 y 2) y dos en sentido horario (3 y 4). Los índices de los motores están etiquetados en los respectivos ESCs.
- La hélice tiene un lado cóncavo y uno convexo. El cóncavo debe estar hacia abajo y en el sentido de giro de la hélice. Es decir, se debe verificar que si los motores 1 y 2 se giran en sentido anti-horario, el lado cóncavo quede orientado hacia abajo y en el sentido del giro. Lo mismo para los motores 3 y 4 en sentido horario.
- Hacer girar las 4 hélices y verificar que ningún motor tenga más inercia que el resto, de ser así se debe cambiar ese motor. También verificar en este punto que los ejes estén centrados. Los motores deben girar en el sentido correcto. Para verificar esto se enciende el sistema en un lugar donde haya buena visibilidad de satélites. El Dron no volará si no tiene un lock del GPS. Con QGroundControl se puede seleccionar algún modo de vuelo manual (e.g. Manual) y mover un poco el throttle del control remoto sin permitir que el Dron despegue. Con esto los motores girarán y se podrá verificar si es correcto o no el sentido de giro. En caso de ser incorrecto se deben conmutar dos de los cables de alimentación del motor.

7.2.1. Descripción de la prueba

Se separará esta prueba en varias pruebas, en la primera se realizará un despegue sobre el punto Home y un aterrizaje en el mismo punto. En la segunda deberá cargarse una trayectoria simple de solo 2 puntos. La tercera prueba será de recorrer un perímetro dado por sus vértices. En la cuarta y última prueba definir el mismo perímetro de la prueba anterior pero ahora el Dron deberá realizar el barrido del área.

7.2.2. Desarrollo de la prueba

Estas pruebas se realizaron solamente en laboratorio con el Dron en HITL verificando los resultado con el simulador jMAVSim.

En la primera prueba se realiza un ascenso del dron sobre el punto de Home y luego se hace el descenso. Esto se logra a través del mensaje 70_Takeoff.

Una vez que se realiza el Take Off el Dron levanta vuelo hasta una altura predeterminada de 3 metros e inmediatamente después desciende en el mismo punto del cual partió.

En la segunda prueba se realiza una misión simple de 2 puntos. Esto se logra enviando un mensaje 20_LoadMission.

Recibido el LoadMission por parte de la GCS, esta envía los waypoints al Dron y le da Take Off iniciando la misión. El Dron asciende sobre el punto Home (que es el primer waypoint) y luego se dirige al segundo waypoint. Se hace descender el Dron en forma manual.

En la tercera prueba responde al mismo mensaje que la segunda prueba pero formando un perímetro.

La GCS recibe el LoadMission y envía los waypoints al Dron, realiza el Take Off iniciando la misión. El Dron asciende sobre el punto Home y luego recorre el perímetro definido por los waypoints. Se hace descender el Dron en forma manual.

En la última prueba se barre el área definida por el perímetro de la prueba anterior. Para ello se utiliza el mensaje 10_LoadArea.

Una vez recibido el mensaje de LoadArea la GCS calcula la trayectoria a recorrer, envía los waypoints al Dron e indica Take Off iniciando la misión. El Dron asciende sobre el punto Home, luego barre el área en forma de ZigZag culminando en el último vertice del perímetro. Se hace descender el Dron en forma manual.

7.3. Detección de objeto caliente con la cámara Lepton

7.3.1. Descripción de la prueba

Probar la detección de un objeto que esté dentro del rango de temperatura predefinida. Dicha detección deberá interrumpir la misión precargada para mantener su posición sobre el objeto observado tanto si este está quieto o en movimiento. Dicha acción también deberá generar la acción de envío de la imagen y demás información a modo de reporte.

7.3.2. Desarrollo de la prueba

En laboratorio, con el Dron en modo HITL, se le carga una misión la cual se controla desde la herramienta QGC. Durante la misión, se pone una mano frente a la cámara. Se interrumpe la misión y el Dron queda en el punto donde se detectó el cuerpo caliente.

7.4. Envío de imágenes a la GCS

7.4.1. Descripción de la prueba

Entablada la comunicación vía radio se deberá enviar imágenes desde el Dron a la GCS verificando la correcta recepción de las mismas.

Colocar la cámara sobre un objeto con temperatura deseada (una persona, por ejemplo). Observar en la base la recepción de la imagen enviada por la PX4.

7.4.2. Desarrollo de la prueba

En el laboratorio se realiza una captura con la cámara térmica. En la GCS se recibe en la variable *thermal_image*, array de caracteres correspondiente a la imagen. Para el envío al usuario, la GCS convierte dicha variable en un archivo BMP para poder enviarlo como adjunto en un correo electrónico.

7.5. Envío de imágenes al usuario (vía e-mail)

7.5.1. Descripción de la prueba

Una vez recibida una imagen en la GCS, la misma deberá reenviarla en un reporte a un usuario vía correo electrónico. Recibida la imagen de la prueba anterior se deberá verificar la correcta recepción de un correo electrónico con la imagen y un reporte correspondiente.

7.5.2. Desarrollo de la prueba

En laboratorio se realiza la recepción de la imagen térmica por parte de la GCS. La misma realiza la conversión de la variable xxx a un archivo BMP en el shield3G de modo de ser enviado por correo electrónico a la casilla del usuario.

Se logra convertir en forma correcta la variable a un archivo BMP y recibir en la casilla de usuario el correo con la imagen adjunta.

7.6. Aterrizaje

7.6.1. Descripción de la prueba

Verificar la correcta descensión y aterrizaje del dispositivo una vez terminado el vuelo. Existen 3 formas de aterrizaje; fin de misión, por comando GoHome o por comando LandNow.

Para el de fin de misión cargar una trayectoria de un solo punto de modo de que el Dron levante vuelo y luego descienda en el mismo lugar. Observar que dicha acción se realiza correctamente sin afectar la estructura del dispositivo.

Para el GoHome y el LandNow, cargar una misión cualquiera y durante el vuelo del Dron, enviar el comando verificando que realice la acción correspondiente y que descienda correctamente.

7.6.2. Desarrollo de la prueba

La prueba realizada consistió en que el Dron se elevara a 2 metros de altura. Con la herramienta QGC se le cargó la misión de un solo waypoint igual al home. El Dron estuvo unos segundos suspendido y luego procedió a descender. Descendió sin problemas y sin efectos que hicieran que el Dron no lograra posarse correctamente en el suelo.

7.7. Alcance de comunicación

7.7.1. Descripción de la prueba

Para la prueba de alcance se deberá alejar el dispositivo de la base lo más lejos posible y ver en qué punto pierde la comunicación. Dicha prueba se realizará en un sector donde haya vista libre entre el dron y la base. Un ejemplo puede ser la rambla y la distancia se medirá con google earth asumiendo el error que se pueda introducir.

7.7.2. Desarrollo de la prueba

Se lleva el Dron hasta la esquina de la Rambla y Jackson. Por otro lado, se coloca la GCS en la parte de atrás de la facultad como se muestra en la Figura 7.10 y se mide hasta perder la comunicación. La distancia en que se perdió la comunicación es de aproximadamente 545 m.

Figura 7.10: Alcance de comunicación. Imagen obtenida de Google Maps.

CAPÍTULO 8

MEJORAS

A lo largo de este proyecto se fue logrando cada objetivo planteado con dificultades propias de la inexperiencia, desconocimiento, fallas o simplemente por el grado de dificultad, y en algunas ocasiones fue necesario tomar decisiones donde se tuvo que modificar el objetivo, reducir su alcance o bien excluirlo, tal es el ejemplo del sistema de carga.

8.1. Sistema de carga automático

El sistema de carga automático implica desarrollar un sistema mecánico entre el Dron y la GCS que permita cargar la batería del Dron sin la intervención de un usuario. Una vez comenzado el proyecto fue necesario descartar este ítem por la dificultad que presentaba. Sin embargo, se considera una mejora a tener en cuenta para futuros trabajos ya que mejoraría considerablemente la independencia de un usuario en campo para la asistencia del dispositivo.

8.2. GCS móvil

Una segunda mejora es agregarle movilidad a la GCS. Esta funcionalidad permitiría que el Dron tuviese mayor alcanzabilidad en distancia a recorrer ya que la comunicación entre éste y la GCS no se perdería nunca. A su vez, en caso de emergencia, el Dron no tiene que recorrer grandes distancias para llegar a la base, extendiendo el tiempo de vuelo sobre una misión.

8.3. Paneles solares en la GCS

Para las mejoras anteriores, la GCS será la encargada de suministrar carga al Dron, así como contar con su propia alimentación. Para ello una opción es contar con paneles solares que permitan dar energía a la GCS y al Dron al momento de cargarse.

8.4. Mejora implementación de la GCS

Para la implementación de la GCS se utilizó una Arduino DUE que si bien tiene varias prestaciones y capacidad necesaria para el cumplimiento de este trabajo, una tarjeta con sistema operativo Linux tal como lo es la tarjeta Rasberry Pi, sería una mejor solución para el propósito del dispositivo.

8.5. GNSS diferencial

El GNSS diferencial, si bien está implementado y funciona, la forma que se encontró para que funcione con el sistema implementado es apagando el Shield 3G, solución poco práctica por el tiempo que demora el Shield 3G en levantar su sistema (30 segundos aproximadamente). Esto se debe a que no se encontró la forma de interrumpir la conexión TCP con el cáster de NTRIP. Al no poder interrumpir dicha comunicación, el Shield 3G deja de responder a otros comandos AT lo que inhabilita el resto de las funcionalidades del mismo.

Una solución posible es implementar un servidor intermedio que capture los mensajes HTTP desde el cáster para luego poder levantar una sesión TCP entre la GCS y dicho servidor realizando consultas HTTP controladas y no streaming como las entrega el cáster. Dicha solución no fue posible implementarse debido a la falta de tiempo, recursos y del servidor que debería estar comunicándose directamente con el cáster.

8.6. Captura y envío de video

En este proyecto se implementa el envío de imágenes térmicas en forma estática y únicamente cuando hay una detección de un cuerpo caliente o bien por solicitud del usuario. Una gran mejora sería poder enviar video para poder observar un vuelo en tiempo real o bien una vez finalizado el mismo. Para ello es necesario contar con el un dispositivo OSD en el Dron que permita capturar y enviar el streaming de video.

8.7. GUI

Actualmente para que un usuario pueda controlar el sistema debe enviar un correo electrónico con comandos en un formato específico para que la GCS pueda interpretarlos correctamente. Una gran mejora sería implementar una interfaz gráfica que le dé al usuario una forma más amigable de interactuar con el sistema.

8.7.1. Interfaz web

Una interfaz web da al usuario la posibilidad de acceder gráficamente al sistema y poder controlarlo de manera remota contando solo con acceso a internet. Las alertas se pueden seguir entregando vía correo electrónico y/o sms para llegar al usuario lo antes posible.

8.7.2. Aplicación móvil

Hoy en día el desarrollo de aplicaciones móviles ha crecido considerablemente teniendo un gran éxito en todas sus áreas. Sería una excelente opción contar con una aplicación móvil que permita controlar el sistema desde un celular o tablet y así mejorar la experiencia del usuario.

Para implementar estas mejoras es necesario contar con un servidor con conexión a internet que no solo implemente la interfaz web, sino que también proporcione la información a una aplicación móvil, así como la mejora mencionada en la sección GNSS diferencial.

8.8. Verificación de convexidad

Al momento de verificar el área a ser relevada por el Dron, no se está verificando la convexidad de la misma lo que puede llegar a validarse la misión y sin embargo pasar por sectores que queden por fuera del área deseada. Sería una gran mejora poder evaluar la misión y en caso de tener un área cóncava, poder dividirla en 2 o más submisiones de áreas convexas evitando así sobrevolar áreas que no están dentro de la misión.

8.9. Conversión de imagen térmica con RGB

Si bien se utilizó una conversión de los datos de la cámara a RGB para poder observar una imagen térmica, la conversión se hizo de modo lineal del verde al amarillo y del amarillo al rojo. Una mejora puede ser utilizar un algoritmo no lineal que utilice los 3 grados (incluyendo al azul) para mejorar el espectro de colores de la

imagen, haciéndola así más agradable.

8.10. Cambio de cámara térmica

Últimamente FLIR ha anunciado el lanzamiento de una nueva versión de su modelo de cámara Lepton con mejor resolución que la utilizada en este trabajo a 160 x 120 píxeles y mejor FoV (56° x 71°). Flir Lepton 3[1] es una buena opción a considerar en caso de que se quiera mejorar la resolución de las imágenes obtenidas en cada misión.

[1] http://www.flir.com/cores/lepton/

CAPÍTULO 9

CONCLUSIONES

Se logra implementar un Dron de vuelo autónomo con una cámara térmica integrada. También se logra implementar con software desarrollado durante este trabajo, una Estación de Control en Tierra (GCS) que cumple la función de controlar el Dron, recibir información del mismo y comunicarse con un usuario vía e-mail. Si bien no se alcanzó la totalidad del objetivo original, se logra un producto funcional con el que se puede seguir una línea de trabajo con varias mejoras o funcionalidades a incluirle.

Se destaca la gran dificultad en la integración de varios componentes de distintos fabricantes. Este punto demandó varias horas de estudio de protocolos, códigos, formatos, etc. que eran ajenos al conocimiento previo y que fueron de principal necesidad para la correcta integración del sistema.

La selección del controlador Pixhawk fue una excelente elección gracias a su gran versatilidad. Es un producto en continuo desarrollo a lo largo de todo el mundo, de código libre y que otorga flexibilidad al momento de desarrollo de proyectos como el que aquí se presentó.

Respecto a la cámara térmica, cabe destacar que la disponibilidad en el mercado es muy limitada en lo que respecta a la cámara únicamente (se consiguen dispositivos con cámara térmica que incluyen display, keyboard, etc. que no eran útiles para este propósito y que su costo era demasiado alto como para acceder a ellos). Dado esto, se optó por la opción más barata de los productos que permitía el único proveedor de cámaras térmicas en desmedro de una peor resolución de la cámara.

Como placas programables se utilizaron 2 Arduino DUE, una on-board y la otra en la GCS. La elección de la placa DUE a los efectos del producto final fue acertada, pero para la GCS hubiese sido mejor una tarjeta con sistema operativo que permitiera trabajar con multi-tarea tal como se menciona en la sección de mejoras.

En lo que al proyecto en sí refiere, se logró realizar un trabajo acorde a lo planificado generando conocimiento y experiencia en interoperabilidad de dispositivos, protocolos de comunicación, desarrollo de software y diseño de circuitos.

Anexos

ANEXO A

GNSS DIFERENCIAL

A.1. Introducción

El Sistema de Posicionamiento Global (GPS) y el Sistema de Navegación Global Satelital (GLONASS) son sistemas de posicionamiento basados en satélite que proveen servicio global las 24 horas. Estos dos sistemas, en conjunto con otros de menor porte (e.g. Galileo) son llamados Sistemas Globales de Navegación por Satélite (GNSS)[53]. Los GNSS proveen servicios con precisión típica de 5-40 metros. La operación diferencial provee precisión en el orden de los metros y la Cinemática en Tiempo Real (RTK) en el orden de los decímetros y hasta incluso centímetros.

El comité especial RTCM 104 (SC-104) [54], servicio de GNSS diferencial ha estandarizado el protocolo de uso de corrección de posición y que utiliza internet como transporte de la información. Cualquier dispositivo con conexión a internet y que cuente con equipo de GNSS que soporte corrección diferencial, puede lograr mejor precisión en su ubicación. Que el sistema sea de esta manera lo hace simple dada la gran cobertura celular con acceso a internet.

El sistema se puede describir en forma simplificada de la siguiente manera, las Estaciones de Referencia de Observación Continua (CORS por sus siglas en inglés), también llamadas estaciones permanentes, de referencia o fijas, son equipos instalados en puntos del cual se conoce su posición con gran precisión. Las CORS constan de instrumentos geodésico para posicionamiento global de alta precisión que funcionan las 24 hs del día, los 365 días del año, observando continuamente las coordenadas geocéntricas del lugar. Las CORS están conectadas a través de la red GPRS/3G a un servidor o cáster que es el encargado de distribuir la información a un cliente mediante una solicitud HTTP.

El usuario que desea obtener corrección diferencial deberá contar con conexión a internet (en el caso móvil GSM, GPRSn EDGE, UMTS) y un software cliente que permita levantar una sesión TCP a la IP del cáster. La conexión utiliza el protocolo NTRIP [55] (Network Transport RTCM Internet Protocol) para el transporte de la información donde RTCM es el standar de comunicación antes mencionado. NTRIP está basado en HTTP 1.1 lo cual hace sencilla la comunicación desde cualquier dispositivo con conexión a internet. NTRIP fue desarrollado en el año 2004 por la BKG (Boundamt für Kartographie und Geodäsie - Agencia Federal Alemana de Cartografía y Geodesia).

Figura A.1: CORS UYMO del Cerro de Montevideo. Imagen obtenida de [55].

A nivel regional se crea la SIRGAS [56] (Sistema de Referencia Geocéntrico para las Américas) que impulsa el desarrollo de la tecnología NTRIP entre los países miembros.

Para este proyecto se utiliza la CORS situada en la fortaleza del cerro como proveedor de datos RTCM para la corrección diferencia (ver Figura A.1).

A.2. Método Diferencial

El método Diferencial consta de calcular un vector diferencial (dX, dY, dZ) entre un punto base (CORS) y un punto móvil (Dron). Una vez resuelto este vector, se aplica el mismo a las coordenadas del punto base, obteniendo las coordenadas (o bien una mejor aproximación) del punto móvil.

La corrección diferencia se puede realizar de dos formas; postproceso (PP) o en tiempo real [10] (RTK), en este caso se ha utilizado RTK aprovechando la conexión a internet.

Para la corrección en tiempo real es necesario tener un medio de comunicación entre el móvil y el cáster que provee la información de la CORS a la que se está haciendo referencia. Los medios de comunicación son *beacon*[1], radio-módem, satélite de comunicaciones, conmutación telefónica e internet. Para este trabajo se utiliza internet dado el gran despliegue de la red 3G y lo sencillo de poder acceder a ella. Se cuenta con una SIM con un servicio de telefonía móvil que permite el acceso.

[1]El beacon es una emisora que trasmite las señales en las bandas de radio con alcances del orden de los 400-500km. Por lo general se necesita un abono a la señal y comúnmente ofrecen solamente correcciones del orden del metro o inferiores.

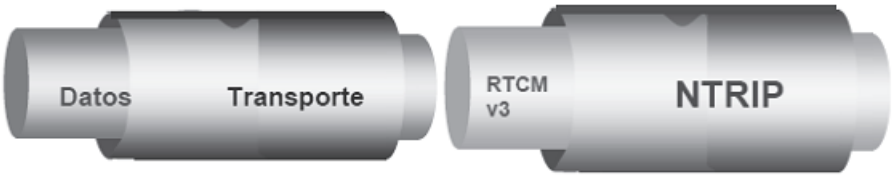


Figura A.2: Data RTCM over Transport NTRIP. Imagen obtenida de [55].

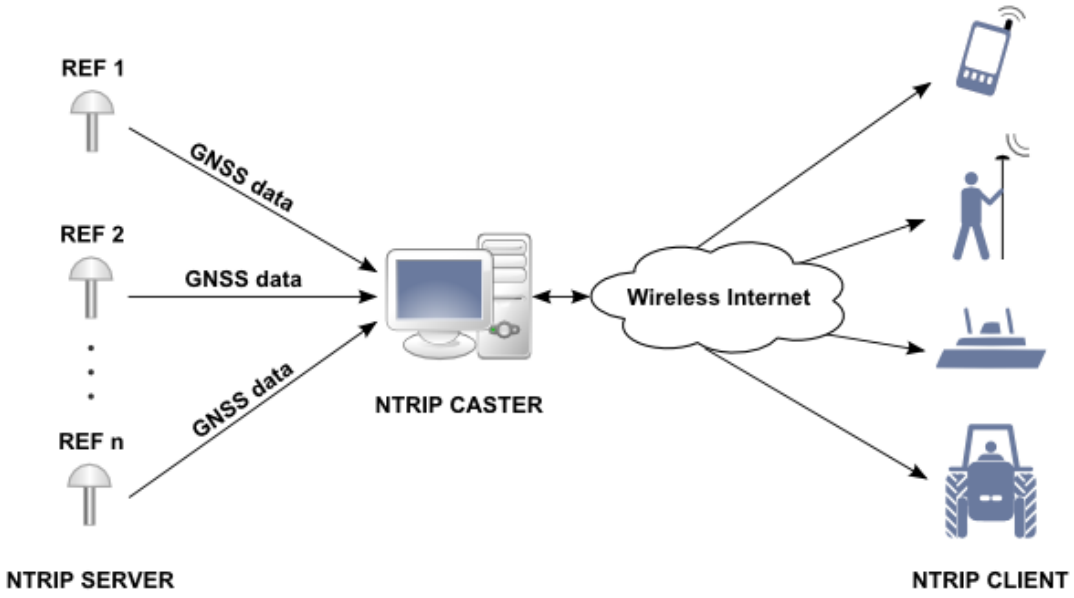


Figura A.3: Diagrama de NTRIP. Imagen obtenida de [57].

A.3. Descripción de NTRIP

Una Red de Transporte de datos RTCM sobre Protocolo de Internet (NTRIP por sus siglas en inglés) permite la comunicación del punto base y el punto móvil sin importar distancias o problemas de visibilidad entre ambos. Una forma de observar este tipo de encapsulación se presenta en la Figura A.2.

Está basado en HTTP/1.1, capa de aplicación que opera sobre TCP/IP. Utiliza el estándar HTTP streaming y es capaz de atravesar firewalls y proxis permitiendo dar acceso al flujo de información brindada por las CORS a todos los usuarios que la soliciten.

Cada CORS recolecta los datos de los satélites en forma continua y los envía a un servidor central (Cáster) donde se calculan los parámetros de corrección que se envía al usuario como se puede observar en la Figura A.3.

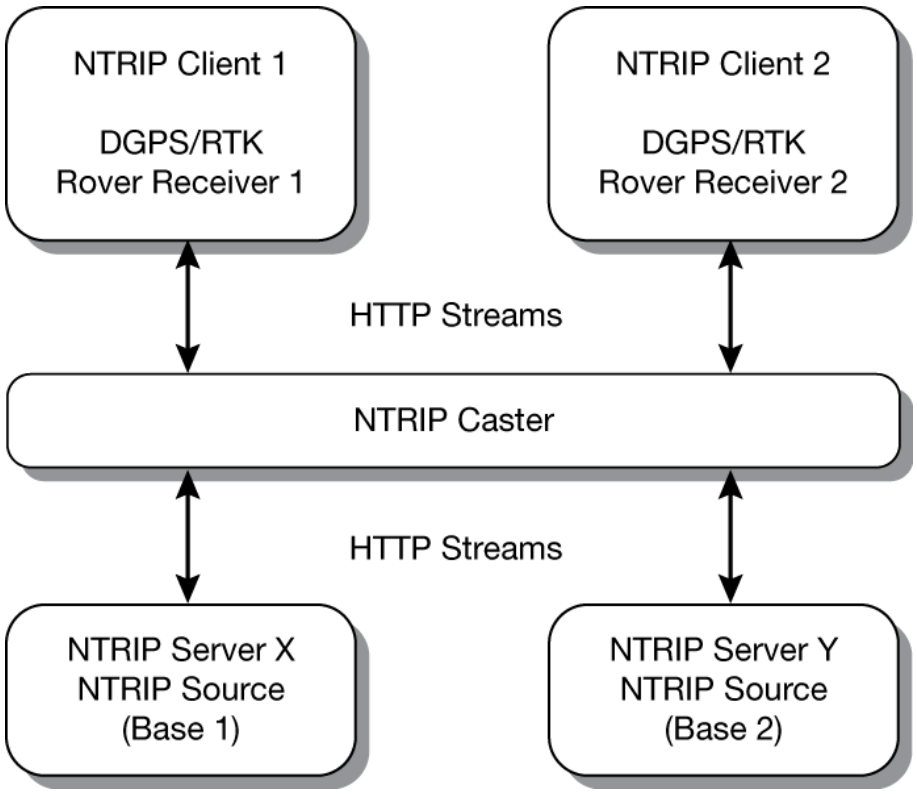


Figura A.4: Arquitectura de NTRIP. Imagen obtenida de [58].

A.4. Arquitectura de NTRIP

La arquitectura de una red NTRIP se puede descomponer en 4 componentes que lo definen (ver Figura A.4).

NtripSource: es el componente de la CORS que implementa las correcciones diferenciales sobre las mediciones de GNSS. Incluye el receptor GNSS y una antena geodésica.

NtripServer: es el componente encargado de recibir los datos desde NtripSource y de entregarlos al NtripCaster. Para ello deberá contar con conexión a internet o bien conexión a un dispositivo que si lo tenga.

NtripCaster: es un servidor con conexión a internet que tiene la capacidad de recibir los datos de todos los NtripServers y tenerlos disponibles para todo NtripClient que desee solicitarlos.

NtripClient: este dispositivo deberá contar con un receptor GNSS con opción RTK (en este caso se utiliza el Ublox NEO-M8 que soporta DGNSS) y contar con conexión a internet o bien un dispositivo que le entregue RTCM y que cuente con conexión a internet. En el diseño de este trabajo la GCS es la que cuenta con la conexión a internet y a través del protocolo MAVLink se transmite por radio el RTCM al receptor GNSS en el Dron.

El flujo de la información se observa claramente en la Figura A.5.

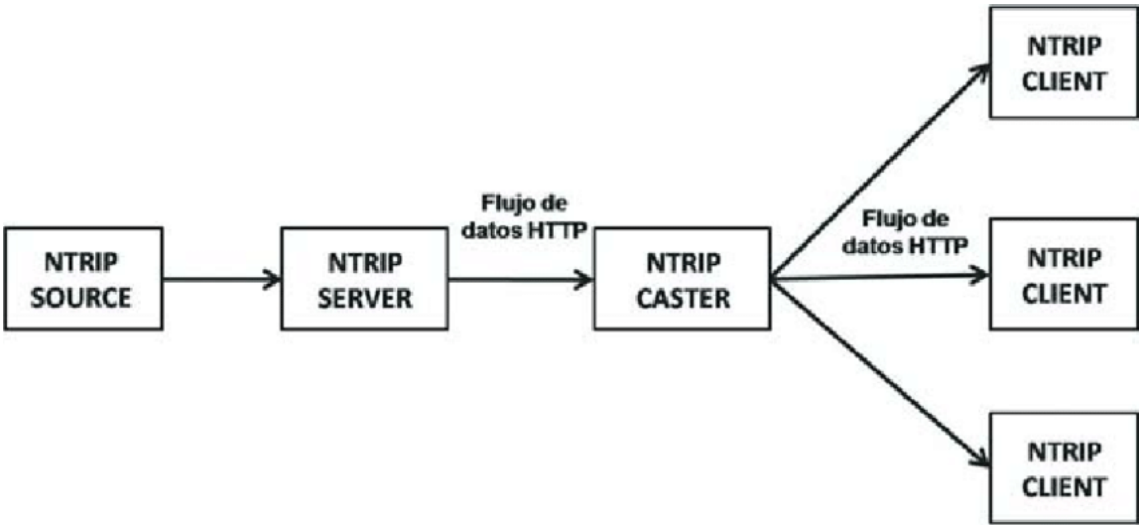


Figura A.5: Flujo de Datos NTRIP. Imagen obtenida de [59].

Preamble	Reserved	Message Length	Variable Length Data Message	CRC
8 bits	6 bits	10 bits	Variable length, integer number of bytes	24 bits
11010011	Not defined – set to 000000	Message length in bytes	0-1023 bytes	QualComm definition CRC-24Q

Figura A.6: Estructura de mensaje RTCM. Imagen obtenida de [54].

A.5. Descripción RTCM

La Comisión Técnica de Radio de Servicios Marítimos (RTCM por sus siglas en inglés) es una organización internacional científica, profesional y educativa sin fines de lucro. Los miembros del RTCM son organizaciones no gubernamentales y gubernamentales. Creada en 1947 como un comité consultivo del gobierno de los Estados Unidos.

Para este proyecto fue necesario entender en forma básica la composición de los mensajes RTCM ya que resultó imprescindible determinar el inicio y fin de los mismos para su retransmisión al Dron. Para ello se recurrió al estándar de RTCM 10403.1 [54].

En la Figura A.6 se puede observar la estructura de los mensajes RTCM. El encabezado se descompone de la siguiente manera; los primeros 8 bits corresponden al inicio de mensaje y corresponde a la palabra D3 en hexadecimal, los siguientes 6 bits son reservados y son todos en 0 y los últimos 10 bits indican el largo del mensaje y se expresa en bytes. Luego del encabezado viene el payload de largo variable (hasta 1023 bytes) y por último 24 bits de CRC.

ANEXO B

PROTOCOLOS, FORMATOS E INTERFACES

B.1. Protocolos de comunicación

A lo largo del proyecto se ha utilizado una gran cantidad de protocolos de comunicación. En muchos casos fue necesario el entendimiento de cada protocolo y el aprendizaje de su uso para lograr el objetivo de este proyecto. También fue necesario el entendimiento del formato de imagen BMP para la creación de archivos de las imágenes térmicas y del estándar de comunicación de los puertos SPI e I^2C. Dado que dicho aprendizaje formó parte del desarrollo del proyecto y ocupó gran parte del tiempo, es que se presenta una breve descripción de cada uno.

B.1.1. AT Commands

Para controlar el Shield 3G se utilizan comandos AT sobre un puerto UART que conecta el shield con Arduino. El Shield cuenta con un integrado SIM5216e de SIMCom [60].

El origen de los comandos AT se remonta a los primeros módems. Fueron utilizados y diseñados por Dennis Hayes y Dale Heatherington[1] en junio de 1981 para el primer módem de PC resolviendo así los problemas de interfaz para permitir a cualquier computadora utilizar un puerto serial estándar para controlar las funciones del módem por software.

ETSI (European Telecommunications Standards Institute) definió comandos del estilo de los AT Commands originales en el standard ETSI GSM 07.07 (3GPP TS 27.007) en el año 1996 y es en este que se basa el integrado de SIMCom para su implementación del protocolo.

Varias de las tareas de la GCS implican comandos AT en su ejecución. A continuación se explica brevemente cada uno de ellos, sus parámetros y función:

[1] http://history-computer.com/ModernComputer/Basis/modem.html

Inicialización del Shield y comandos generales

AT+CREG? - Retorna el estado del equipo en la red. Se utiliza para saber si la tarjeta SIM está registrada en la red local.

AT - Retorna OK si el Shield está disponible.

AT+CPIN=<pin> - Introduce el PIN de la tarjeta SIM para desbloquearla.

AT+CGSOCKCONT=1,"IP",<apn> - Setea el nombre del punto de acceso (APN) para que el dispositivo pueda acceder a la red, en este caso es "prepago.ancel".

AT+CSOCKAUTH=1,1,<user>,<password> - Setea el nombre de usuario y passwords del punto de acceso, en este caso es "bam" en ambos casos.

AT+FSCD=<path> - Cambia la carpeta a path.

POP3

AT+POP3SRV=<server>,<user>,<password>[,<port>] - Este comando es utilizado para setear todos los parámetros y obtener un e-mail del servidor POP3; se deben proporcionar servidor, usuario, password y puerto. Si el cliente POP3 no está libre, el comando simplemente retorna "ERROR". Este comando solo necesita ejecutarse una vez cada vez que se corre el programa.

AT+POP3IN - Se ingresa al servidor POP3 con las credenciales seteadas anteriormente. Este comando y los que siguen, deben ejecutarse cada vez que se quiera leer un nuevo mail.

AT+POP3NUM - Se solicita la cantidad de correos en el servidor.

AT+POP3GET= <n> - Se solicita el enésimo correo del servidor, el shield devuelve el nombre del archivo. Como siempre se mira el último correo, la cantidad de correos es igual a *n*, a su vez el último valor de cantidad de correos es guardado en una variable global para evitar leer dos veces el mismo correo.

AT+POP3READ=0,<path> - Solicita la lectura del correo que se encuentra en path, el 0 indica que es el sistema de archivos local.

AT+POP3OUT - Se hace el logout del servidor POP3.

SMTP

AT+SMTPSRV=<server>,<port> - Se setean el servidor y el puerto SMTP, en este caso son "smtp.montevideo.com.uy" y 25.

AT+SMTPAUTH=1,<user>,<password> - Se loguea con las credenciales en el servidor seteado anteriormente, aquí se ingresa la dirección de correo electrónico y password de la casilla asignada al dron.

AT+SMTPFROM=<user>,<name> - Se setea la dirección y el nombre de quien será el remitente del correo .

AT+SMTPRCPT=<adress>,<name> - Se setea la dirección y el nombre de quien será el destinatario del correo .

AT+SMTPSUB=<subject> - Se setea el subject del correo.

AT+SMTPFILE=1,=<filename> - Se setea el nombre del archivo adjunto del correo.

AT+SMTPBODY - Se indíca que se enviará el body, el shield responde con >> y a partir de ahí se deben imprimir los caracteres codificados en UTF-8 en el puerto y finalizar con 0x1A

AT+SMTPSEND - Se envía el correo con los parámetros seteados anteriormente.

B.1.2. Xmodem

Para transferir archivos de Arduino al Shield se utiliza el protocolo Xmodem, ya que es el que el Shield soporta. Xmodem es un protocolo de transferencia de archivos que fue desarrollado en 1977 por Ward Christensen[61] y posteriormente cedido para su uso público, por lo cual este protocolo también es conocido como "protocolo Christensen". Se trata de un protocolo de transferencia muy sencillo de implementar que gracias a ello alcanzó gran popularidad, actualmente la mayoría de los módems soportan aún este protocolo.

Básicamente el funcionamiento de Xmodem es el de envío de instrucción y espera de acknowledge. Utiliza un conjunto bien definido de instrucciones y sus mensajes son de largo fijo (133 bytes) definidos como se muestra en la Figura B.1. Los 3 primeros bytes, correspondientes al Header, se componen por el tipo de mensaje (primer byte), el número de paquete que se inicia en 1 y una vez alcanzado el valor 255 reinicia su valor a 0, y el tercer byte es el complemento del valor del segundo byte. Entre el byte 4 y el 131 van los 128 bytes de payload y por último dos bytes de CRC.

Cada mensaje se envía del transmisor al receptor y éste último responde ACK o NACK dependiendo si recibió correctamente el mensaje. En la Figura B.2 se muestra

Byte 1	Byte 2	Byte 3	Bytes 4-131	Bytes 132-133
Start of Header	Packet Number	(Packet Number)	Packet Data	16-bit CRC

Figura B.1: Frame de Xmodem. Imagen obtenida de [62].

el flujo de control en una comunicación de Xmodem.

Symbol	Description	Value
SOH	Start of Header	0x01
EOT	End of Transmission	0x04
ACK	Acknowledge	0x06
NAK	Not Acknowledge	0x15
ETB	End of Transmission Block (Return to Amulet OS mode)	0x17
CAN	Cancel (Force receiver to start sending C's)	0x18
C	ASCII "C"	0x43

Figura B.2: Flujo de mensajes de Xmodem. Imagen obtenida de [62].

Para el inicio de envío de mensajes Xmodem, el receptor envía una "C" (0x43) al transmisor indicando que está pronto para recibir mensajes Xmodem. A partir de ese momento el transmisor comienza a enviar los paquetes de largo 133 bytes y el receptor valida los mensajes chequeando que el primer byte tenga valores válidos (SOH, EOT, CAN o ETB), que la suma de los bytes 2 y 3 sumen siempre 0xFF y que el valor de CRC sea el correcto, y contestando con un ACK (0x06) o NACK (0x15). Una vez que el transmisor haya finalizado el envío de mensajes con payload, envía un mensaje EOT (0x04) y la comunicación finaliza una vez que el transmisor recibe el ACK de dicho EOT. En la Figura B.3 se puede observar un ejemplo de intercambio de mensajes Xmodem entre un transmisor y un receptor.

Para el último mensaje con payload, si la cantidad de bytes es menor a 128, se deberá rellenar el payload con 0x00.

B.1.3. RTCM

El protocolo RTCM fue desarrollado con el fin de transmitir información de corrección GNSS tal como se explicó en este documento. La GCS debe poder interpretar dicha información para luego reenviarla al Dron y que el receptor de GNSS pueda procesarla.

Para que la GCS pudiera realizar dicha tarea, se tuvo que estudiar la estructura de los mensajes RTCM. Básicamente un mensaje RTCM tiene la estructura que se muestra en la Figura B.4.

Todo el proceso de recepción de mensajes RTCM se realizan en el Shield 3G y para su inicio se utilizan AT commands. Los siguientes comandos son los utilizados en este proceso:

AT+NETOPEN="TCP",2101 - Con este comando se habilita el puerto 2101 para iniciar una sesión TCP a través de él.

AT+CHTTPACT="201.217.132.178", 2101 - Este comando inicia una sesión TCP en el puerto 2101 del servidor con IP 201.217.132.178 (Cáster del REGNA-ROU).

Sender						Receiver
					<----	"C"
						Times Out after 3 Seconds
					<----	"C"
SOH	0x01	0xFE	Data	CRC	---->	Packet OK
					<----	ACK
SOH	0x02	0xFD	Data	CRC	---->	(Line Hit during Data Transmission)
					<----	NACK
SOH	0x02	0xFD	Data	CRC	---->	Packet OK
					<----	ACK
SOH	0x03	0xFC	Data	CRC	---->	Packet OK
(ACK Gets Garbled)					<----	ACK
					<----	ACK
SOH	0x04	0xFB	Data	CRC	---->	(UART Framing Error on Any Byte)
					<----	NACK
SOH	0x04	0xFB	Data	CRC	---->	Packet OK
					<----	ACK
SOH	0x05	0xFA	Data	CRC	---->	(UART Overrun Error on Any Byte)
					<----	NACK
SOH	0x05	0xFA	Data	CRC	---->	Packet OK
					<----	ACK
EOT					---->	Packet OK
					<----	ACK
ETB					---->	Finished
Finished					<----	ACK

Figura B.3: Ejemplo de intercambio de mensajes Xmodem. Imagen obtenida de [62].

Preamble	**Reserved**	**Message Length**	**Variable Length Data Message**	**CRC**
8 bits	6 bits	10 bits	Variable length, integer number of bytes	24 bits
11010011	Not defined – set to 000000	Message length in bytes	0-1023 bytes	QualComm definition CRC-24Q

Figura B.4: Estructura de mensaje RTCM.

*GET /UYMO HTTP/1.1\r\nUser-Agent: Termodron\r\nAccept: */*\r\nConnection: close\r\nAuthorization: Basic dGVybW9kcm9uOnRlcm1vZHJvbjE=\r\n\r\n* - Una vez establecida la sesión TCP se procede a hacer la consulta HTTP GET la cual comenzará a enviar en forma de streaming mensajes HTTP con RTCM encapsulado.

En respuesta al comando GET de HTTP se reciben mensajes HTTP. La respuesta en AT Command es *+CHTTPACT: DATA, <len>\r\n* donde *len* indica la cantidad de bytes del payload del mensaje HTTP que no es otra cosa que la concatenación de los mensajes RTCM enviados por el Cáster. El tamaño de *len* es menor que 1460. Para cada mensaje HTTP recibido se toma el valor de *len* y luego del salto de línea se comienza a leer el mensaje RTCM. Los primeros 8 bits son fijos y corresponden al valor 0xD3, los siguientes 6 bits no están definidos y están fijados en 0, los próximos 10 bits corresponden al largo de todo el mensaje expresado en bytes, en base a este valor es que se sabe hasta dónde leer el mensaje RTCM que se está leyendo. Una vez completado todo el mensaje se hace un control de CRC con los últimos 24 bits del mismo.

Antes de continuar leyendo el siguiente mensaje RTCM, se procede a enviar el mensaje RTCM anterior al Dron. Para ello se utilizan funciones MAVLink para luego imprimir en el serial que se comunica con el Dron:

mavlink_msg_gps_rtcm_data_pack(system_id, component_id, &msg, flagRTCM, MSGLen ,&bufferMSG[0]) - Esta función empaqueta el mensaje RTCM en un buffer para luego ser enviado. Los mensajes MAVLink de RTCM no pueden superar el tamaño de payload de 180 bytes, en ese caso deberá enviarse el mensaje fragmentado. Luego se imprime en el serial correspondiente el contenido del buffer (buf) seguido del largo (len).

B.1.4. MAVLink

MAVLink (Micro Air Vehicle Link) es un protocolo de comunicación orientado a UAVs. Fue creado en 2009 por Lorenz Meier bajo licencia LGPL (Library General Public License). Es un protocolo que se implementa como una librería "header only", lo que significa que simplemente incluyéndola en el encabezado de un programa a través del comando #include se cuenta con todas sus funciones.

Desde su creación en 2009, ha habido tres versiones de MAVLink (0.9, 1.0 y 2.0). En este proyecto se utiliza la versión 2.0, el detalle de cada mensaje se puede ver en [40].

Los mensajes de MAVLink están formados por "paquetes" de largo variable. Van de entre 8 y 263 bytes. En la Figura B.5 se puede observar la estructura de un paquete. La estructura detallada es como sigue:

- El primer byte indica el inicio de un paquete y es siempre 0xFE (0x55 en versiones anteriores)
- El segundo byte indica el largo del "payload", el payload es la información que contiene el mensaje (e.g. valores del GPS, ángulos de Euler, etc.) y puede

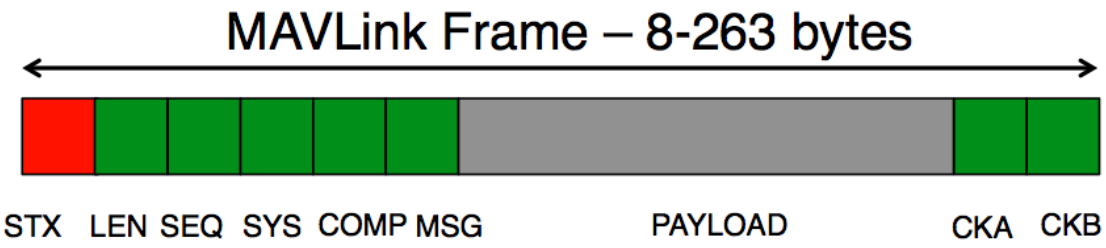


Figura B.5: Estructura de un paquete MAVLink.

valer entre 0x00 y 0xFF

- El tercer byte indica la secuencia del mensaje, es un valor que va aumentando de manera circular de 0x00 a 0xFF y su propósito es detectar la pérdida de mensajes
- El cuarto byte es la ID del sistema, valor que va entre 0x01 y 0xFF que identifica el sistema que envía el mensaje (e.g. GCS, Pixhawk, Arduino on-board)
- El quinto byte es la ID del componoente, ídem que ID del sistema, pero para componentes internos a un sistema (e.g. IMU de Pixhawk)
- El sexto byte es la ID del mensaje, identifica el mensaje dando una manera de interpretar el payload
- Desde el séptimo al antepenúltimo byte está el payload, es efectivamente la información que se quiere enviar
- Los últimos dos bytes son de checksum para detectar errores en la recepción

B.2. Formato de Archivo

Para la transmisión de imágenes desde el Dron al usuario fue necesario entender la estructura de un archivo BMP para poder convertir un conjunto de valores en una imagen entendible para un usuario. Se elige BMP[2] por su sencillez al momento de implementarlo.

B.2.1. BMP

Introducción

El formato BMP de Windows se remonta a más de dos décadas y ha subsistido gracias a su estabilidad y que es un producto Microsoft compatible con todas las versiones de Windows.

La estructura básica del archivo es binaria (en contraposición a un archivo de texto) y se divide en las cuatro secciones siguientes:

[2]http://www.dragonwins.com/domains/getteched/bmp/bmpfileformat.htm

Campo	Tamaño en bytes	Descripción
bfType	2	Caracteres “BM”
bfSize	4	Tamaño del archivo en bytes
bfReserved1	2	Reservado, seteado en 0
bfReserved2	2	Reservado, seteado en 0
bfOffBits	4	Offset al inicio de los datos de Píxeles

Tabla B.1: Campos del encabezado de BMP

- Encabezado (14 bytes)
- El encabezado de imagen (40 bytes)
- La Tabla de Color (la longitud varía y no siempre está presente)
- Los datos de píxeles

Nota: Los datos de imagen se almacenan en formato Little Endian lo cual significa que los bytes de la imagen se almacenan en el orden del byte menos significativo primero y el más significativo último.

Encabezado

El encabezado del archivo BMP contiene 14 bytes tal como se muestra en la Tabla B.1.

Los primeros dos bytes deben ser los códigos ASCII para los caracteres “B” y “M”. Los siguientes 4 bytes indican el tamaño del archivo BMP. BfReserved1 y BfReserved2 son reservados para futuras versiones y se setean en 0. Los últimos 4 bytes del encabezado indican el tamaño del encabezado de la imagen e indica dónde comienzan los datos de imagen contenida en el archivo.

Encabezado de la imagen

El encabezado de imagen para Windows contiene al menos 40 bytes. En la Tabla B.2 se muestra el formato del encabezado básico de imagen de Windows de 40 bytes.

BiSize indica el tamaño, en bytes, del encabezado de imagen. Los siguientes 8 bytes indican el ancho y alto de la imagen en píxeles. Para el alto, los datos se ordenan de abajo hacia arriba por lo que, si el valor del alto es negativo, los datos están ordenados de arriba hacia abajo. El campo biplanos se setea en 1. El bibitcount indica la cantidad de bits por píxel de la imagen. Los siguientes 4 bytes indican la compresión (0 indica que no hay compresión). Bisizeimage indica el tamaño de los datos de píxeles reales en bytes. BiXpelspermeter y BiYpelspermeter es la resolución horizontal y vertical preferida de la imagen respectivamente, en píxeles por metro.

Campo	Tamaño en bytes	Descripción
BiSize	4	Tamaño del encabezado
BiWidth	4	Ancho de la imagen en píxeles
BiHeight	4	Altura de la imagen en píxeles
Biplanos	2	Debe ser 1
BiBitCount	2	bits por píxel - 1, 4, 8, 16, 24 o 32
BiCompression	4	Tipo de compresión (0 = sin comprimir)
BiSizeImage	4	Image Size - puede ser cero para imágenes sin comprimir
BiXPelsPerMeter	4	Resolución preferida en píxeles por metro
BiYPelsPerMeter	4	Resolución preferida en píxeles por metro
BiClrUsed	4	Número de entradas del mapa de color que se utilizan realmente
BiClrImportant	4	Número de colores significativos

Tabla B.2: Campos del encabezado de BMP

Normalmente se setea en 0 indicando que no hay preferencia. BiClrUsed se setea en 0 salvo que la imagen utilice menos de 256 colores (8 bits). Si la imagen utiliza menos de 256, simplemente se indica el número de colores en este campo para menor almacenamiento. BiClr Importante es el número de colores que se consideran importantes al renderizar la imagen. Si este parámetro es igual a cero, todos los colores de la Tabla de colores se considerarán importantes.

Tabla de colores

Si se trata de imágenes con una profundidad de bits de 8 o menos, los datos de píxeles son en realidad un índice en una paleta de colores.

En la versión de archivo BMP utilizada, cada color ocupa 4 bytes. Tomados como un solo valor de 32 bits, los cuatro bytes se ordenan de la siguiente manera: [ZERO] [ROJO] [VERDE] [AZUL]. Debido al formato Little Endian, el valor Blue va primero seguido por el verde y luego el rojo. Un cuarto octeto se setea en 0. Para imágenes de 16 o 32 bits, la Tabla de colores contiene un conjunto de máscaras de bits que se utilizan para definir qué bits de los datos de píxeles se asocian con cada color. Para imágenes de 24 bits, no hay tabla de colores presente.

Los datos de píxeles

Los datos de píxeles se organizan en filas de abajo hacia arriba y, dentro de cada fila, de izquierda a derecha en referencia a la imagen. Cada fila se llama “línea de

exploración". Si la altura de la imagen se da como un número negativo, las filas se ordenan de arriba a abajo.

El número de bytes necesarios para almacenar cada línea de exploración debe ser un múltiplo par de cuatro, de lo contrario se deberá insertar bits nulos de relleno.

1 bit (mapa de bits monocromo) - Cada byte de datos representa 8 píxeles. El bit más significativo representa el píxel más a la izquierda del grupo de ocho.
4 bits (mapa de bits de 16 colores) - Cada byte de datos representa 2 píxeles. El nibble más significativo se asigna al píxel más a la izquierda más píxel del grupo de dos.
8 bits (mapa de bits de 256 colores) - Cada byte representa 1 píxel.
16 bits (hasta 65.536 colores, pero comúnmente 32.768 colores) - Cada píxel está representado por dos bytes. Una representación común es RGB555 que asigna 5 bits a cada color permitiendo 32K colores mientras que deja un pedacito sin usar. Dado que el ojo humano es más sensible al verde, otra representación común es RGB565 que asigna este bit no utilizado al componente verde.
24 bits (16.777.216 colores)- Cada píxel está representado por tres bytes. El primer byte representa la intensidad del color rojo, el segundo del verde y el tercero del azul.
32 bits (hasta 4.294.967.296 colores, pero comúnmente 1.073.741.824 o 16.777.216 colores) - Cada píxel está representado por cuatro bytes. Aunque esto significa que podrían representarse más de cuatro mil millones de colores, muy pocos dispositivos de visualización son capaces de tal resolución. Sin embargo, puede ser deseable almacenar imágenes con una alta resolución tal que se pueda realizar un procesamiento significativo en los datos sin que se produzca una degradación significativa de los errores de redondeo acumulados. Aun así, generalmente es suficiente mantener solamente dos bits adicionales de información por píxel, por lo que la representación RGB101010 asigna 10 bits a cada color. Otra representación común es el RGB888 que simplemente utiliza un formato de 32 bits para almacenar imágenes de 24 bits con el fin de aprovechar la capacidad del procesador para trabajar con datos en bloques de 32 bits de manera más eficiente.

B.3. Puertos de comunicación

B.3.1. SPI

Serial Peripheral Interface, por sus siglas en inglés, es un protocolo asíncrono serie para la transmisión de datos, utilizado por microcontroladores para comunicarse con uno o más periféricos rápidamente y a corta distancia. Además, se puede utilizar para comunicar dos microcontroladores entre ellos. Una conexión SPI cuenta con un dispositivo Master (generalmente un microcontrolador) quien se encarga de controlar los otros periféricos conectados.

Típicamente hay tres líneas de comunicación comunes a todos los dispositivos conectados y una línea específica para cada uno. Las más comunes son:

- *MISO (Master In Slave Out)* - Para enviar datos de los periféricos al Master.

- *MOSI (Master Out Slave In)* - Para enviar datos del Master a cada uno de los periféricos conectados.
- *SCK (Serial Clock)* - Los pulsos de reloj que sincronizan la transmisión de datos son generados por el Master.

La línea específica es:

- *SS (Slave Select)* - El pin en cada dispositivo que el maestro puede utilizar para activar y desactivar dicho dispositivo.

La comunicación se inicia cuando el Master configura el reloj utilizando una frecuencia soportada por los dispositivos esclavos. El maestro selecciona entonces el dispositivo esclavo con el que interactuará colocando un nivel lógico 0 en su línea de selección.

Durante cada ciclo de reloj SPI, una transmisión de datos dúplex completo se produce. El maestro envía un bit en la línea MOSI y el esclavo lo lee, mientras que el esclavo envía un bit en la línea MISO y el Master lo lee. Esta secuencia se mantiene incluso cuando sólo la transferencia de datos es unidireccional.

Las transmisiones normalmente implican dos registros de desplazamiento de un tamaño de la palabra dado (por ejemplo, ocho bits) uno en el maestro y uno en el esclavo; que están conectados en una topología de anillo virtual. Los datos se desplazan generalmente con el bit más significativo primero, al tiempo que cambia un nuevo bit menos significativo en el mismo registro. Al mismo tiempo, los datos de la contraparte se desplazan en el registro de bit menos significativo. Si hay más datos necesita ser intercambiado, los registros de desplazamiento se vuelven a cargar y el proceso se repite. La transmisión puede continuar para cualquier número de ciclos de reloj. Cuando se haya completado, el maestro deja de conmutar la señal de reloj, y por lo general anula la selección del esclavo. Cada esclavo en el bus al que no se le ha activado mediante su línea de selección de chip debe ignorar el reloj de entrada y las señales MOSI y MISO.

En Arduino se utiliza el objeto SPISettings para configurar el Puerto SPI del dispositivo. Se configuran tres parámetros: velocidad máxima de comunicación en Hz, el orden de la información (si primero va el MSB o el LSB) y el modo de la información (hay 4 modos: *SPI_MODE0*, *SPI_MODE1*, *SPI_MODE2*, *SPI_MODE3*). Los cuatro modos se diferencian en si los datos entran o salen en el flanco ascendente o descendente de la señal de reloj de datos (fase del reloj) y si el reloj está inactivo cuando es alto o bajo (polaridad del reloj). Los cuatro modos combinan la polaridad y la fase de acuerdo con la Tabla B.3.

Usualmente una comunicación con interfaz SPI consta de un llamado a la función *SPI.beginTransaction()*, luego escribir el pin SS en nivel bajo, llamar a la función *SPI.transfer()* tantas veces como sea necesario para transferir la información deseada, después se escribe un nivel alto en el pin SS y por último se llama a la función *SPI.endTransaction()*.

Modo	Clock Polarity (CPOL)	Clock Phase (CPHA)	Output Edge	Data Capture
SPI_MODE0	0	0	Falling	Rising
SPI_MODE1	0	1	Rising	Falling
SPI_MODE2	1	0	Falling	Rising
SPI_MODE3	1	1	Rising	Falling

Tabla B.3: Modos de información SPI

Modo	Velocidad de transmisión máxima	Dirección
Standard Mode (Sm)	0.1 Mbit/s	Bidireccional
Fast Mode (Fm)	0.4 Mbit/s	Bidireccional
Fast Mode Plus (Fm+)	1.0 Mbit/s	Bidireccional
High Speed Mode (Hs-mode)	3.4 Mbit/s	Bidireccional
Ultra Fast-mode (UFm)	5.0 Mbit/s	Unidireccional

Tabla B.4: Comparación entre las diferentes opciones de controladores de vuelo

B.3.2. I^2C

Inter-Integrated Circuit, por sus siglas en inglés, es un protocolo serie que requiere de sólo dos cables, pero que puede soportar hasta 1008 dispositivos esclavos. El I^2C está diseñado como un bus maestro-esclavo. La transferencia de datos es siempre inicializada por un maestro y el esclavo reacciona. Es posible tener varios maestros mediante un modo multi-Master, en el que se pueden comunicar dos maestros entre sí, de modo que uno de ellos trabaja como esclavo. El control de acceso en el bus se rige por las especificaciones, de este modo los maestros pueden ir turnándose.

El I^2C utiliza dos líneas de señal: reloj (SCL, Serial Clock) y la línea de datos (SDA, Serial Data). Ambas líneas precisan resistencias de pull-up hacia VDD. Cualquier dispositivo conectado a estas líneas es Open Collector, lo cual en combinación con las resistencias pull-up, crea un circuito Wired-AND. El nivel alto debe ser de al menos 0,7 x VDD y el nivel bajo no debe ser más de 0,3 x VDD. La señal de reloj siempre es generada por el maestro. Para cada modo especificado, está predeterminado respectivamente un pulso de reloj máximo permitido. En la tabla B.4 se muestran las velocidades máximas permisibles de transmisión.

Si el esclavo necesita más tiempo que el dictado por el reloj del maestro, puede mantener, entre la transferencia de bytes individuales, la señal de reloj en nivel bajo (clock-stretching) para frenar de este modo al maestro. Los datos sólo son válidos si su nivel lógico no cambia durante una fase de reloj alta. Las excepciones son el inicio, la parada, y reset. La señal de arranque es un flanco descendiente en SDA mientras SCL se encuentra en nivel alto. La señal de parada es un flanco ascendiente en SDA mientras SCL está en nivel alto. La señal de reset se comporta de igual manera

que la señal de inicio. Una unidad de datos consta de 8 bits (los cuales puede ser interpretados como un valor o como una dirección, dependiendo del protocolo) y un bit de ACK (acknowledge). Este bit ACK es señalizado por un esclavo como NACK (not acknowledge) con un nivel alto, durante un nivel bajo en SDA y el noveno nivel alto de SCL (que sigue siendo generado por el maestro). El esclavo debe poner un nivel bajo en SDA antes de que SCL cambie a nivel alto, de lo contrario otros participantes podrían interpretar esto como una señal de arranque.

La dirección del estándar I^2C es el primer byte enviado por el maestro, en realidad los primeros 7 bits representan la dirección y el octavo bit es el de escritura-lectura (R/W-Bit). Por lo tanto, I^2C utiliza un espacio de direccionamiento de 7 bits, lo cual permite hasta 112 nodos en un bus (16 de las 128 direcciones posibles están reservadas para fines especiales). Debido a la escasez de direcciones, se introdujo más tarde un direccionamiento de 10 bits. Es compatible con el estándar de 7 bits mediante el uso de 4 de las 16 direcciones reservadas. Ambos modos de direccionamiento pueden utilizarse simultáneamente, lo que permite hasta 1136 nodos en un único bus.

El inicio de una transmisión es indicado por la señal de inicio del maestro, seguido de la dirección. Ésta es confirmada por el bit ACK del esclavo correspondiente. En función del R/W-Bit se escriben bytes de datos (datos al esclavo) o se leen (datos al maestro). El ACK es enviado desde el esclavo al escribir, y desde el maestro al leer. El último byte leído es reconocido por el maestro como un NACK, para indicar el final de una transmisión. Una transmisión es finalizada por la señal de parada. Como alternativa, puede ser enviada una señal de reset al arranque de una nueva transmisión, sin necesidad de parar la transmisión anterior con una señal de parada. Todos los bytes son transferidos de esta manera como MSBF (bit más significativo primero).

ANEXO C

MANUAL DE USUARIO

El objetivo de este documento es darle al usuario una guía que le permita utilizar el sistema en su totalidad, desde la puesta en funcionamiento hasta los comandos para el intercambio de información.

C.1. Partes del producto

El sistema debe contener una GCS (Base en Tierra) y un cuadricóptero. La GCS incluye la tarjeta SIM con un servicio de internet.

C.2. Puesta en funcionamiento

Posicionar el Dron próximo al área a ser relevada. En el Dron, conectar la batería al distribuidor de corriente para energizarlo. Colocar la GCS lo más próximo al Dron y considerar que el área a relevar no supere la distancia máxima de comunicación entre el Dron y la GCS (máximo sugerido: 1 km). Energizar la GCS verificando que los leds de las tarjetas queden encendidos. Verificar que en el lugar donde se coloque la GCS haya cobertura 3G.

De operar próximo al sitio de trabajo, se sugiere conectar un laptop a la GCS en el puerto Programming con un conector USB-miniUSB de modo de poder observar todo el proceso en una consola. Para ello utilizar un programa cliente con opción de conexión serial (por ejemplo PuTTY) y levantar consola con velocidad 115200 bps al puerto COM que corresponda.

Una vez iniciada la GCS, un correo electrónico será enviado al usuario configurado por defecto indicando el inicio del sistema. A partir de este punto el sistema está en condiciones de recibir comandos desde el usuario (ver sección de comandos de usuario) para comenzar la operación del sistema.

Cualquier usuario puede enviar una misión a la GCS. La GCS comenzará a responder y enviar reportes a la casilla que haya enviado la misión.

En las Tablas C.1 y C.2 se presentan características del Dron y la GCS respectivamente

En las Figuras C.1 y C.2 se indica el significado de las secuencias de colores de los led del Dron mientras que en la Tabla C.3 se indica la secuencia de parpadeo de

Power	
Alimentación motores	11.1 V
Alimentación integrados	5.0 V
Consumo máx motores	192 W
Consumo máx integrados	1.5 W
Tiempo de vuelo a pot máxima	10 min
Dimensiones	
Peso	1.9 kg
Diámetro	75 cm
Altura	50 cm

Tabla C.1: Especificaciones del Dron

Power	
Alimentación	5.0 V
Consumo máximo	4.15 W
Dimensiones	
Peso	450 g
Volúmen	25x16x7 cm^3
Interfaces	
miniUSB	

Tabla C.2: Especificaciones de la GCS

los leds de la tarjeta Shield3G de la GCS.

Estado del LED	**Estado del shield**
Led Rojo	GCS encendida
Led amarillo izquierdo	Transmisión de datos con PX4
Led amarillo derecho	Transmisión de datos 3G

Tabla C.3: Estado de leds de la GCS

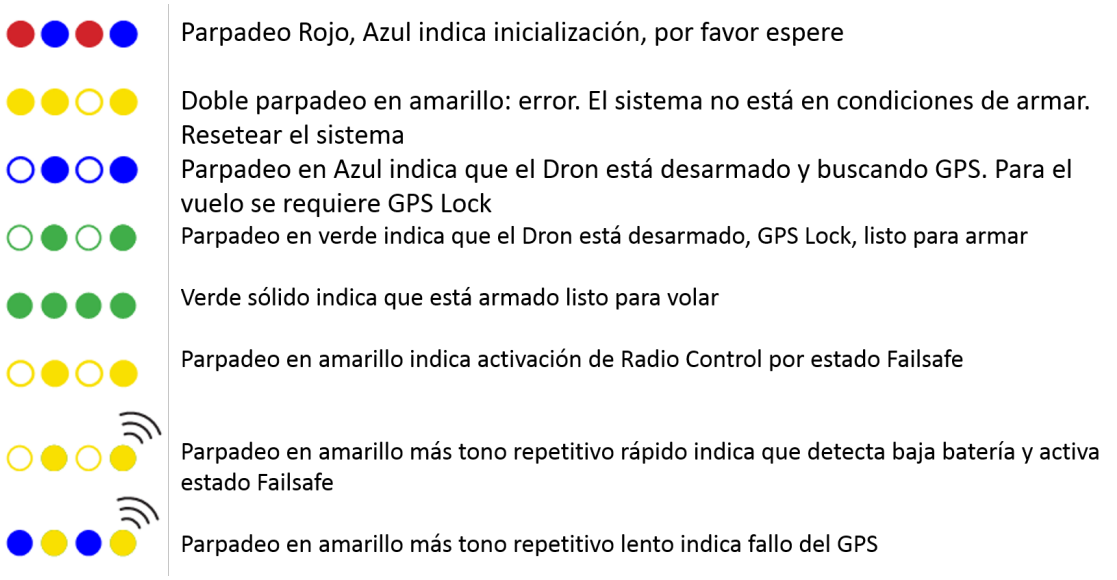


Figura C.1: Leds controlador de vuelo del Dron. Imagen obtenida de [63].

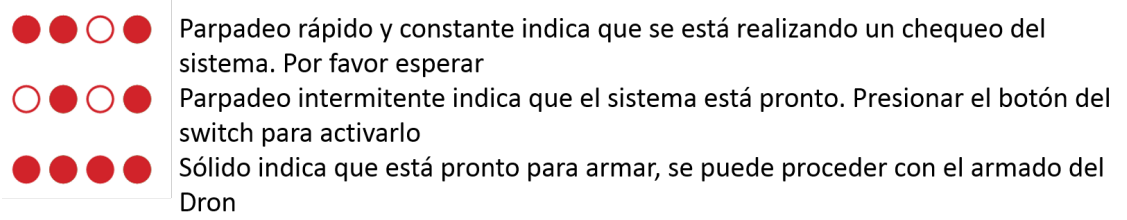


Figura C.2: Led del switch. Imagen obtenida de [63].

C.3. Reportes de la GCS

La GCS reporta al usuario en distintas instancias:

En forma periódica reporta estado de vuelo del Dron como se muestra en el ejemplo:

INICIO REPORTE
LATITUD = -34.1234567 °
LONGITUD = -56.1234567 °
ALTITUD = 5000 mm
BATERIA = 87 %
YAW = 15.000000 °
PITCH = 6.000000 °
ROLL = 0.000000 °
TIEMPO DESDE BOOTEO = 356 s

En el momento en que se detecta un objeto caliente, el Dron envía la imagen a la GCS la cual reenviará la imagen como archivo con extensión BMP.

Una vez finalizada una misión, la GCS enviará un reporte final al usuario con un adjunto con extensión TXT con toda la información relevada por el Dron durante el vuelo.

C.4. Comandos de usuario

En esta sección se definen los comandos que el usuario podrá enviar desde una casilla de correo electrónico personal a la cuenta utilizada especialmente con el propósito de comunicación con la GCS. El detalle de cada comando se describe en este documento y para el buen funcionamiento del sistema se deberá seguir en forma estricta el formato que se especifica.

Cada comando deberá ser enviado en el asunto del correo electrónico a la GCS. El asunto deberá comenzar con un índice, que es el que indica a la GCS lo solicitado por parte del usuario. El resto de la línea podrá ser utilizada libremente por el usuario a modo de referencia.

Se sugiere escribir el asunto del correo en el formato indicado en el listado. En el cuerpo del mensaje se podrá incluir información o no, dependiendo del comando, y siempre terminando con la palabra *FIN;*.

La casilla de correo utilizada es termodron1@montevideo.com.uy creada específicamente para este propósito.

Listado de comandos:

- 00_LoadSettings
- 10_LoadArea
- 20_LoadMisson
- 30_GetPhoto
- 40_GetLog
- 50_GetStatus
- 60_ContinueLoadedMission

- 70_Takeoff
- 80_GoHome
- 90_LandNow

Definición de los comandos

- **00_LoadSettings**

Este comando habilita algunos reportes que serán enviados cada 30 segundos. El no envío de este comando deja sin enviar reportes durante una misión. En el cuerpo del mensaje se deberá indicar los tipos de reportes que el usuario desea recibir. Para finalizar el mensaje se deberá incluir en la última línea la palabra *FIN;*.

Los reportes que se pueden habilitar son los siguientes:

LOGMAIL: TRUE –> El sistema enviará valores de estado, latitud, longitud, altura, velocidades, carga de batería, tiempo de misión, si encontró un cuerpo caliente, modo de vuelo, WP actual, WP siguiente

IMG: TRUE –> El sistema enviará la última foto capturada por la cámara on-board

ALERTAS: TRUE –> El sistema enviará un correo por cada alerta que suceda, despegue, encuentra un cuerpo caliente, encuentra un obstáculo, niveles de batería bajos, aterrizajes forzados con ubicación

MODO_DET: FOLLOW –> Una vez que detecte el objeto caliente, el dron deberá mantenerse sobre este y enviar reporte (este modo es el modo por defecto)

MODO_DET: RTL –> Una vez que detecte el objeto caliente, el dron deberá reportar y retornar al home

MODO_DET: CONT –> Una vez que detecte el objeto caliente, el dron deberá reportar y continuar con el recorrido de la misión precargada

Ejemplo:

para: termodron1@montevideo.com.uy
asunto: 00_LoadSettings

LOGMAIL: TRUE
IMG: TRUE
MODO_DET: RTL
FIN;

- **10_LoadArea**

Este comando indica coordenadas definidas por el usuario. Las mismas representan los vértices de un área a barrer por el Dron. Para el caso de únicamente 2 puntos se realizará un vuelo en línea recta.

La latitud y la longitud deberán ser expresadas en grados y sus decimales (no minutos ni segundos), y la altitud deberá ser expresada en metros y no podrá ser menor a 3 metros. El comando deberá ir en el asunto del correo y en el cuerpo del mensaje deberá respetar el siguiente formato:

<latitud_1>;<longitud_1>;<altitud_1>;
<latitud_2>;<longitud_2>;<altitud_2>;
<latitud_3>;<longitud_3>;<altitud_3>;
. . .
<latitud_N>;<longitud_N>;<altitud_N>;
FIN;

Ejemplo:

para: termodron1@montevideo.com.uy
asunto: 10_LoadArea

47.3997419;8.5495940;5;
45.3997419;8.5495940;5;
45.3997419;6.5495940;5;
47.3997419;6.5495940;5;
FIN;

Nota: El valor de N está limitado de forma tal de no consumir mucha memoria en Arduino. El mismo tiene que ser menor o igual a 20.

- **20_LoadMisson**

Este comando indica los waypoints por los que deberá navegar el Dron. A diferencia de LoadArea que calcula la forma de barrer el área definida por los puntos enviados, LoadMission carga los puntos por los que pasará el Dron en el orden indicado por el usuario. El comando deberá ir en el asunto del correo y el cuerpo del mensaje deberá respetar el siguiente formato:

<latitud_1>;<longitud_1>;<altitud_1>;
<latitud_2>;<longitud_2>;<altitud_2>;
<latitud_3>;<longitud_3>;<altitud_3>;
. . .
<latitud_N>;<longitud_N>;<altitud_N>;
FIN;

Ejemplo:

para: termodron1@montevideo.com.uy
asunto: 20_LoadMission

47.3997419;8.5495940;5;

45.3997419;8.5495940;5;
45.3997419;6.5495940;5;
47.3997419;6.5495940;5;
FIN;

Nota: El valor de N está limitado de forma tal de no consumir mucha memoria en Arduino. El mismo tiene que ser menor o igual a 20.

- **30_GetPhoto**

Este comando es para solicitar la última imagen procesada por el Dron. El Dron guarda solamente la última foto con el fin de encontrar un cuerpo caliente en ella. Para el siguiente chequeo descarta la imagen anterior. El comando deberá ir en el asunto del correo y el cuerpo del mensaje deberá contener únicamente la palabra *FIN;*

- **40_GetLog**

Este comando es para solicitar el log de información del Dron. El comando deberá ir en el asunto del correo y en el cuerpo del mensaje deberá respetar el siguiente formato:

<índiceLog_1>
<índiceLog_2>
<índiceLog_3>
. . .
<índiceLog_N>
FIN;

Ejemplo:

para: termodron1@montevideo.com.uy
asunto: 40_GetLog

0
1
3
FIN;

Los logs definidos son los siguientes:

0: Todos los logs
1: Log de errores
2: Log de GNSS/DGNSS
3: Log de velocidad

- **50_GetStatus**

Este comando es para solicitar información de estado del Dron. El comando deberá ir en el asunto del correo y el cuerpo del mensaje deberá contener únicamente la palabra *FIN;*

- **60_ContinueLoadedMission**

Este comando es para indicar al Dron que continúe con su misión. El mismo solo tendrá efecto en caso de que la misión haya sido interrumpida por alguna razón en particular (objeto caliente que no interesa seguir, luego de un Land, luego de un GoHome, etc.) y siempre que el continuar la misión sea viable. El comando deberá ir en el asunto del correo y el cuerpo del mensaje deberá contener únicamente la palabra *FIN;*

- **70_Takeoff**

Este comando es para indicar al Dron que inicie un vuelo si el mismo se encuentra desarmado. El Dron se elevará sobre su posición de Home y en caso de tener una misión precargada la ejecutará. El comando deberá ir en el asunto del correo y el cuerpo del mensaje deberá contener únicamente la palabra *FIN;*

- **80_GoHome**

Este comando es para indicar al Dron que interrumpa su misión y se dirija a la posición de origen. El comando deberá ir en el asunto del correo y el cuerpo del mensaje deberá contener únicamente la palabra *FIN;*

- **90_LandNow**

Este comando es para indicar al Dron que interrumpa su misión y aterrice en el lugar donde se encuentre. El comando deberá ir en el asunto del correo y el cuerpo del mensaje deberá contener únicamente la palabra *FIN;*

REFERENCIAS

[1] L. R. NEWCOME, *Unmanned Aviation. A Brief History of Unmanned Aerial Vehicles.* American Insitute of Aeronautics and Astronautics. Reston, Virginia (EE.UU.), 2004.

[2] K. P. VALAVANIS and G. J. VACHTSEVANOS, *Handbook of Unmanned Aerial Vehicles.* Springer Science+Business Media. Dordrecht (Holanda), 2015.

[3] 2017. [Online]. Available: http://drones.uv.es/origen-y-desarrollo-de-los-drones

[4] INTERNATIONAL CIVIL AVIATION ORGANIZATION, *Unmanned Aircraft Systems (UAS), Circular 328, AN/190*, 2011.

[5] 2017. [Online]. Available: http://drones.uv.es/origen-y-desarrollo-de-los-drones

[6] Jean-Charles LEDÉ, *Fast Lightweight Autonomy.* Defense Advanced Research Projects Agency, (EE.UU.), 2016. [Online]. Available: http://www.darpa.mil/program/fast-lightweight-autonomy

[7] Santiago PATERNIAN, Rodrigo ROSA y Matías TAILANIÁN, *Implementación de un UAV con arquitectura de cuadricóptero.* Proyecto de Fin de Carrera de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República., 2012.

[8] Joaquíin BERRUTTI, Lucas FALKENSTEIN, Federico FAVARO, *Vuelo autónomo de un cuadricóptero.* Proyecto de Fin de Carrera de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República., 2015.

[9] Sebastián TORTEROLO, Juan Manuel TORTEROLO, Facundo CAYAFA, *Diseño y automatización de un UAV.* Proyecto de Fin de Carrera de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República., 2016.

[10] T. L. DAMMALAGE and L. SAMARAKOON, *Test results of RTK and Real-Time DGPS corrected observations based on NTRIP protocol.* The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences., 2008.

[11] "Pixhawk flight controller hardware project," 2017. [Online]. Available: https://pixhawk.org/

[12] Mohinder S. Grewal and Angus P. Andrews, *Kalman Filtering: Theory and Practice using MATLAB, Second Edition.* Wiley-Interscience, 2001.

[13] 2017. [Online]. Available: http://wiki.theuavguide.com/wiki/File:Multicopter_Dynamics.png

[14] Ward BROWN, *Brushless DC motor control made easy.* Technical report, Microchip Technology Inc., 2002.

[15] 2017. [Online]. Available: https://www.slideshare.net/QuadsForFun/drone-quadcopter-quad-quadricopter-quadrocopter-multirotor

[16] Carlos PEÑA ORDOÑEZ, *Estudio de baterías para vehículos eléctricos.* Universidad Carlos III de Madrid, Mayo, 2011.

[17] Carlos Peña Ordóñez, *ESTUDIO DE BATERÍAS PARA VEHÍCULOS ELÉCTRICOS.*

[18] "Motor emax 2822 datasheet," 2016. [Online]. Available: http://www.merqc.com/files/Datasheet/emax.pdf

[19] 2017. [Online]. Available: http://www.manualvuelo.com/SIF/SIF32.html

[20] 2017. [Online]. Available: http://progettovolo.weebly.com/eliche

[21] Julián MARTÍNEZ DE LA CALLE y José GONZÁLEZ PÉREZ, *Propulsores Marinos.* Servicio de Publicaciones, Universidad de Oviedo, 1997.

[22] "Pixhawk datasheet," 2017. [Online]. Available: https://3dr.com/wp-content/uploads/2017/03/pixhawk-manual-rev7-1.pdf

[23] Henry J. ZHANG, *Basic Concepts of Linear Regulator and Switching Mode Power Supplies.* Linear Technology, 2013.

[24] "U-blox m8n datasheet," 2015. [Online]. Available: https://www.u-blox.com/sites/default/files/NEO-M8_DataSheet_(UBX-13003366).pdf

[25] "Shield sd datasheet," 2012. [Online]. Available: http://www.supertalent.com/datasheets/5_112.pdf

[26] "Radio telemetry 3dr datasheet," 2013. [Online]. Available: https://3dr.com/wp-content/uploads/2013/10/3DR-Radio-V2-doc1.pdf

[27] "Arduino due datasheet," 2015. [Online]. Available: https://www.arduino.cc/

[28] "Shield 3g itead datasheet," 2014. [Online]. Available: https://www.openhacks.com/uploadsproductos/itead_3g_shield_-_itead_wiki.pdf

[29] "Turnigy tgy9x datasheet," 2010. [Online]. Available: https://hobbyking.com/media/file/725056143X2037269X20.pdf

[30] “Dht22 datasheet.” [Online]. Available: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

[31] “Rain sensor datasheet.” [Online]. Available: https://www.openhacks.com/uploadsproductos/rain_sensor_module.pdf

[32] “Integrado lm393r datasheet,” 1999. [Online]. Available: http://www.ti.com/lit/ds/symlink/lm393-n.pdf

[33] “Flir lepton datasheet,” 2014. [Online]. Available: http://www.flir.com/uploadedFiles/OEM/Products/LWIR-Cameras/Lepton/Lepton-3-Engineering-Datasheet.pdf

[34] Dr. Mario Piris Silvera, *FISICA CUANTICA*. Editorial ISCTN. La Habana, Cuba., 1999.

[35] 2017. [Online]. Available: http://www.gravita-zero.org/2016/05/speciale-onde-gravitazionali-i-cento.html

[36] 2017. [Online]. Available: http://ingecivilcusco.blogspot.com.uy/2009/09/sistema-geodesico-mundial-1984-wgs84.html

[37] 2017. [Online]. Available: https://gis.stackexchange.com/questions/141496/which-utm-zone-to-use-for-a-large-country

[38] “Calculate distance, bearing and more between latitude/longitude points,” 2017. [Online]. Available: http://www.movable-type.co.uk/scripts/latlong.html

[39] “Romuald ireneus scibor-marchocki, spherical trigonometry, elementary-geometry trigonometry,” 1997. [Online]. Available: http://www.webcitation.org/query?url=http://www.geocities.com/ResearchTriangle/2363/trig02.html&date=2009-10-25+09:44:36

[40] L. MEIER, “Mavlink common message set,” 2016. [Online]. Available: http://mavlink.org/messages/common

[41] 2017. [Online]. Available: https://erlerobotics.gitbooks.io/erlerobot/en/mavlink/mavlink.html

[42] 2017. [Online]. Available: http://slideplayer.com/slide/10319788/

[43] “Cooperative multitasking for arduino microcontrollers,” 2016. [Online]. Available: http://www.smart4smart.com/TaskScheduler.html

[44] “Xmodem protocol with crc,” 2004. [Online]. Available: http://web.mit.edu/6.115/www/amulet/xmodem.htm

[45] “Processing foundation,” 2017. [Online]. Available: https://processing.org/

[46] 2017. [Online]. Available: http://xstitch.zachrattner.com/HowItWorks.html

[47] 2017. [Online]. Available: http://qgroundcontrol.org/mavlink/waypoint_protocol

[48] “Px4 development guide,” 2016. [Online]. Available: https://dev.px4.io/

[49] “Nuttx real-time operating system,” 2016. [Online]. Available: http://nuttx.org/

[50] “jmavsim,” 2016. [Online]. Available: https://pixhawk.org/dev/hil/jmavsim

[51] “Qgroundcontrol,” 2016. [Online]. Available: http://qgroundcontrol.com/

[52] 2017. [Online]. Available: https://www.gitbook.com/book/donlakeflyer/qgroundcontrol-user-guide/details

[53] NovAtel Inc., *An Introduction to GNSS*. NovAtel Inc., 2015.

[54] Radio Technical Commission For Maritime Services, *RTCM STANDARD 10403.1*. Radio Technical Commission For Maritime Services, 2006.

[55] ——, *Networked Transport of RTCM via Internet Protocol (Ntrip)*. Radio Technical Commission For Maritime Services, 2003.

[56] Tte. Cnel. Norbertino SUÁREZ SILVA, Ing. Agrim. Roberto PÉREZ RODINO y Ing. Agrim. Ricardo YELICICH PELÁEZ, *Red de transporte de datos en formato RTCM, vía protocolo de Internet (Ntrip). Implementación en la región y proyección futura a través de SIRGAS*. Revista Cartográfica 89, 2013.

[57] 2017. [Online]. Available: https://www.alberding.eu/en/ntripcaster.html

[58] 2017. [Online]. Available: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1982-21702015000100043

[59] 2017. [Online]. Available: https://www.researchgate.net/figure/292289557_fig1_Figura-1-Arquitectura-del-NTRIP

[60] “At command set,” 2010. [Online]. Available: http://ec-mobile.ru/user_files/File/SIMCom/simcom_sim5215sim5216_atc_en_v1.05.pdf

[61] “Xmodem,” 2017. [Online]. Available: http://history-computer.com/ModernComputer/Basis/modem.html

[62] 2017. [Online]. Available: http://web.mit.edu/6.115/www/amulet/xmodem.htm

[63] 2017. [Online]. Available: http://www.manualsdir.com/manuals/576767/3d-robotics-pixhawk-autopilot.html?page=10