DESARROLLO DE UN DISPOSITIVO DE MONITOREO EN TIEMPO REAL DE LOS SISTEMAS DE UNA AERONAVE MEDIANTE UN PROTOCOLO DE COMUNICACIÓN INALÁMBRICA.

RTAMS (REAL-TIME AIRCRAFT MONITORING SYSTEM)

JULIAN ANDRÉS AWAZACKO AWAZACKO

UNIVERSIDAD SANTO TOMÁS
FACULTAD DE INGENIERÍA ELECTRÓNICA
SECCIONAL TUNJA
2021

DESARROLLO DE UN DISPOSITIVO DE MONITOREO EN TIEMPO REAL DE LOS SISTEMAS DE UNA AERONAVE MEDIANTE UN PROTOCOLO DE COMUNICACIÓN INALÁMBRICA.

RTAMS (REAL-TIME AIRCRAFT MONITORING SYSTEM)

JULIAN ANDRES AWAZACKO AWAZACKO

TRABAJO DE GRADO PARA OBTENER EL TITULO DE: INGENIERO ELECTRÓNICO

Director de tesis: César Mauricio Galarza Bogotá
Co Director de tesis: Angélica María Salazar Madrigal

UNIVERSIDAD SANTO TOMÁS
FACULTAD DE INGENIERIA ELECTRONICA
SECCIONAL TUNJA
2021

EXONERACION DE RESPONSABILIDADES

El autor se hace responsable de las ideas expresadas en este documento con la intención de exonerar a la Universidad Santo Tomás seccional Tunja y a la Facultad de ingeniería Electrónica de cualquier responsabilidad.

Nota de aceptación

_____

_____

_____

_____

_____

Firma del Director

_____

Firma del Codirector

_____

Firma del Jurado

_____

Firma del Jurado

Tunja, Marzo 2021

DEDICATORIA

Dedico este trabajo primeramente a Dios quien me dio el conocimiento y la fortaleza en todo el proceso durante mi paso por la universidad, por darme la vida y permitirme llegar hasta donde estoy ahora, el momento que todo estudiante anhela.

Dedico este trabajo de grado a mi padre quien me ofreció durante toda la carrera su apoyo incondicional desde el primer día de clases hasta el último; quien me apoya en cada decisión que tomó, una persona que lo da todo por mí y siempre me aconseja, en segundo lugar, a mi madre quien desde el cielo ha colaborado para guiarme en este proceso y ayudarme en momentos que lo necesito desde allá arriba.

Dedico esta tesis a mi hermana, quien ha sido como una madre durante toda mi vida, una persona incondicional que me apoya en cada momento y me ayuda a no decaer, que sin importar la circunstancia esta allí para mi demostrándome su amor y apoyo.

Y por último a todos los docentes y personas cercanas que estuvieron a mi lado durante este proceso ayudándome a culminar con éxito mis estudios.

AGRADECIMIENTOS

Quiero agradecerle a Dios por darme la vida y ser una guía durante el transcurso de mi carrera, darme sabiduría y enseñarme el camino correcto en esta etapa de mi vida.

A mi padre por darme su apoyo incondicional y convertirse en un modelo a seguir como persona, especialmente agradecerle por su esfuerzo y entrega, por estar en los momentos difíciles y a mi madre quien desde el cielo de alguna manera guio e ilumino el camino para cumplir con éxito esta meta.

A mi hermana por sus consejos y enseñarme a tener paciencia en momentos donde era imposible tenerla, por estar siempre ahí a mi lado y a mi cuñado Juan Parra quien ha sido un gran consejero durante mi paso por la universidad.

A las personas que han seguido de cerca este proceso, enseñándome a no tirar la toalla y a perseguir mis sueños.

A los ingenieros de la facultad de ingeniería electrónica, quienes me compartieron sus conocimientos.

Agradecimiento especial a mis tutores de grado quienes me apoyaron y brindaron toda su colaboración para culminar con éxito este proyecto de grado.

TABLA DE CONTENIDO

LISTAS DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

GLOSARIO

- **Acelerómetro:** Dispositivo que cuenta con la capacidad de medir la aceleración de una estructura.
- **Barómetro:** Instrumento que mide la presión atmosférica.
- **Dispositivos:** Conjunto de componentes eléctricos o piezas preparados para ejercer una función específica.
- **Emisor:** Persona o Dispositivo que envía un mensaje.
- **Escalabilidad:** Capacidad que tiene un sistema para crecer y adaptarse sin perder calidad.
- **Giróscopo:** Dispositivo capaz de medir la rotación de una estructura.
- **GPS:** Sistema de posicionamiento global, permite posicionar cualquier objeto sobre la tierra.
- **Hardware:** Parte física de un sistema tecnológico.
- **IMU:** Unidad de medición inercial, generalmente combina los acelerómetros y los giroscopios para entregar mediciones combinadas de estos.
- **IOT:** Internet de las cosas, interconexión de objetos físicos y sensores con otros sistemas con la finalidad de intercambiar datos a través de la red.
- **Lenguaje de programación:** Es el lenguaje que se utiliza para la comunicación entre una computadora y un humano para decirle a esta que es lo que se quiere realizar.
- **Prototipo:** Primer ejemplar que se fabrica de un producto para ejecutar pruebas y que sirve de modelo para fabricar o mejorar el producto.
- **Receptor:** Persona o dispositivo que recibe un mensaje.
- **RPAS:** Sistema de aeronaves pilotadas a distancia, generalmente también se conoce como drone.
- **Sensor:** Dispositivo capacitado para detectar acciones o variables externos.
- **Software:** Programas y rutinas que ejecuta un sistema tecnológico.

- **Telemetría:** Permite la medición y visualización remota de variables de un sistema.
- **UAV:** Vehículo aéreo no tripulado, la diferencia con el RPAS es que este término se usa más que todo en la industria militar.

RESUMEN

El presente documento recopila el trabajo realizado en el proyecto de grado para obtener el título de ingeniero electrónico en la Universidad Santo Tomas Seccional Tunja, el proyecto plantea el desarrollo de un prototipo que sea capaz de monitorear en tiempo real la mayoría de los sistemas de una aeronave, para este caso, se hace el monitoreo a un RPAS (Remotely Piloted Aircraft System) pues su adquisición y/o construcción es más accequible, cabe aclarar que la utilización del RPAS será únicamente para la fase de pruebas, el principal objetivo de este es llegar a ofrecer un soporte en tiempo real al piloto en caso de averías en la aeronave, esto debido a que actualmente el sistema de monitoreo que más se utiliza en la actualidad se ofrece para un mantenimiento preventivo.

Para cumplir el objetivo principal se implementó un algoritmo el cual permite la adquisición y procesamiento de datos obtenidos a partir de los sensores agregados a la aeronave, los cuales fueron; Sensor GPS para conocer la ubicación, altitud, velocidad, rumbo entre otros, un sensor inercial el cual permite obtener la aceleración y los datos de giroscopio de la aeronave y así determinar su orientación actual y el sensor de presión el cual permite calcular la altitud de una manera más precisa y promediarla con el valor que arroja el GPS.

El prototipo entregó una idea del comportamiento al instante de realizar la adquisición y envió de tramas de datos dentro de una aeronave a escala, pudiendo llegar a ser funcional en un avión comercial.

PROLOGO

En la actualidad existen sistemas de seguimiento de aeronaves conocidos como ACARS (Aircraft Communication Addressing And Reporting System), dichos sistemas proporcionan una información limitada a la hora de realizar la adquisición de datos, ya que estos envían información como número de pasajeros, combustible utilizado y no suelen servir de mucho al área de mantenimiento si se quisiera ofrecer un soporte en tiempo real, es por ello que este proyecto presentado como tesis para obtener el título de ingeniero electrónico en la Universidad Santo Tomas Seccional Tunja, propone el desarrollo de un prototipo que sea capaz de monitorear en tiempo real la mayoría de los sistemas de una aeronave y así poder entregar la información necesaria para un buen mantenimiento y monitoreo, para este caso el monitoreo se hace a un RPAS (Remotely Piloted Aircraft System) ya que su adquisición o construcción es más asequible, el prototipo de vuelo dará una idea del comportamiento al momento de realizar la adquisición y envío de tramas de datos dentro de la aeronave, y así demostrar que este puede llegar a ser funcional en un avión comercial.

I. INTRODUCCIÓN

Este trabajo de grado presenta el diseño e implementación de un dispositivo de monitoreo que cuenta con la capacidad de hacer la adquisición de los datos de vuelo de una aeronave. El prototipo inicial adquiere los datos de: presión, temperatura, altitud, velocidad, posicionamiento (GPS), aceleración(x,y,z) y la orientación(x,y,z) de un aeromodelo (RPAS), esto con el fin de probar el comportamiento a escala de algunos de los sensores con los que cuenta una aeronave de aviación comercial, puesto que no se cuentan con los recursos necesarios para acceder a una real.

Se implemento un algoritmo que permitiera al sistema adquirir los datos de los diferentes sensores implementados en la aeronave y reenviarlos por una comunicación inalámbrica a una página web implementada mediante el uso de la tecnología IOT (Internet Of Things). En este documento se muestra la metodología utilizada para el desarrollo del proyecto, junto con los resultados obtenidos y algunas conclusiones.

El prototipo entrega una idea de su comportamiento y como este sería de gran ayuda a futuro para el monitoreo completo de un vuelo comercial, todo esto con el fin de lograr ofrecer un soporte al piloto en tiempo real y lograr solucionar algunas averías en vuelo.

II. JUSTIFICACIÓN

En la actualidad existen sistemas de monitoreo en tiempo real conocidos como ACARS (Aircraft Communication Addressing And Reporting System), pero estos sistemas solo prestan una información limitada para el mantenimiento y operación de las aeronaves, bien sea comerciales o privadas, permitiendo una comunicación sin intervención humana; los mensajes habituales que envía este sistema son: carga de pasajeros, informes de despegues, aterrizajes, cantidad de combustible abordo, información de algunas eventualidades presente con algún sistema de la aeronave, entre otros. Estos sistemas como se mencionó anteriormente ofrecen un mantenimiento preventivo a la aeronave, pues cuando ésta se lleva al área de mantenimiento MRO (Maintenance, Repair and Overhaul) se suelen mirar los datos de ACARS. Adicional a esto se reduce la carga laboral de las tripulaciones de vuelos, pues este hace el reporte de pasajeros y cantidad de carga llevada en cada vuelo. A pesar de ser el sistema de monitoreo que más se usa en la actualidad, éste no es capaz de ofrecer una ayuda en tiempo real al piloto. Ya que se usa más que todo para realizar mantenimientos preventivos a la aeronave y este solo se puede realizar en tierra. Uno de los casos donde el sistema ACARS muestra que no es capaz de ofrecer el soporte en tiempo real y que hay necesidad de ello, fue en el vuelo AIR FRANCE 447 (ICAO INT) donde tristemente a pesar de que este obtuvo los datos de ubicación minutos antes del accidente, no fueron lo suficientemente precisos puesto que el avión fue hallado semanas después y adicional a esto nunca se logró ofrecer al piloto alguna ayuda adicional para cambiar el resultado de esta tragedia. El sistema de monitoreo propuesto en este proyecto busca una interacción mecánico-piloto, logrando así que un mecánico visualice la gran mayoría de los sistemas y las alertas mostradas de la aeronave al piloto en tiempo real, para así lograr sortear una emergencia al poder ofrecerle el soporte y ayuda adecuados.

III. PLANTEAMIENTO DEL PROBLEMA

1. FORMULACIÓN DE PREGUNTAS

¿De qué manera el sistema RTAMS mejoraría el tiempo de respuesta en la solución de averías mecánicas o eléctricas en una aeronave que se encuentre en vuelo?

2. DEFINICIÓN DEL PROBLEMA

Actualmente los pilotos están capacitados para sortear eventualidades que puedan presentarse en la aeronave a través de un manual de referencia POH (Pilot Operating Handbook) (ICAO INT), el cual les indica qué tipo de acciones tomar en caso de fallo; sin embargo, en ciertas ocasiones los pilotos no tienen el tiempo necesario para sortear estos obstáculos o no cuentan con la información suficiente para hacerlo. En algunos casos los pilotos llaman a la estación base para obtener un soporte mayor, puesto que en este lugar cuentan con un manual más extenso y técnicos capacitados, pero este procedimiento conlleva en la mayoría de los casos a que la comunicación sea difícil ya que el piloto tiene que infórmale del fallo bien sea al mecánico o a otro piloto en tierra contando con muy poco tiempo, lo que origina catástrofes aéreas, ocasionando inevitablemente pérdidas de vidas humanas.

3. DELIMITACIÓN DEL PROBLEMA

Se plantea el uso de un prototipo a escala (aeromodelo) y adicional se selecciona un protocolo de comunicación inalámbrica para comunicar el aeromodelo con una estación terrena y así simular una comunicación satelital para poder ver que errores se podrían presentar.

a. Desarrollo del prototipo

El prototipo que se propone realizar corresponde a un modelo a escala, en el cual se utiliza un avión radio controlado con una envergadura de 1.5 metros y un tiempo de vuelo que varía entre 10 y 20 minutos dependiendo su alimentación, adicional a esto se le añaden los sensores de presión, GPS y el sensor inercial para conocer su orientación.

El prototipo también contempla una similitud a escala de los componentes y sistemas que comprenden el modelo a escala real.

Se ajusta el peso y número de componentes que comprenden el prototipo electrónico para no afectar la aerodinámica de la nave, ni para afectar fuertemente el peso y balance de modo que no pudiese volar o ser controlada en el aire correctamente.

b. Tipo de tecnología para la comunicación inalámbrica y cobertura.

El sistema propuesto plantea el uso de tecnología wifi para el acceso a internet, el uso de internet móvil 4G también se puede tener en cuenta ya que este está presente en el área geográfica de prueba, lo que permitirá el levantamiento de resultados reales que beneficiarían el uso de nuestro sistema.

El sistema se complementará con el uso de la tecnología XBee, esto para hacer la adquisición de los datos en vuelo y enviarlos a el dispositivo retransmisor que se encargará de hacer la comunicación a la página web, utilizando bien sea tecnología móvil 4G dependiendo de la cobertura o haciendo uso de la tecnología wifi, pues por peso y balance no se puede poner todos los implementos dentro del UAV.

Haciendo el análisis del uso del internet móvil 4G comparado con el internet satelital, se determina que es factible para el desarrollo del prototipo a escala, ya que es asequible en cuanto a costos e implementación.

c. Limitaciones geográficas.

En cuanto a las limitaciones geográficas, se tiene como área de prueba para la implementación de este prototipo la ciudad de Chía, en la que la red 4G posee una cobertura óptima; no se descarta la posibilidad de que el prototipo pueda también llegar a probarse en algunas zonas donde la tecnología 4G sea una limitante para comprobar la funcionalidad del dispositivo. Adicional a ello en la ciudad de Chía se cuenta con una pista de aeromodelismo la cual es necesaria para realizar el vuelo de la aeronave puesto que volar la aeronave cerca de la universidad no es posible según las regulaciones aeronáuticas de Colombia y es un riesgo para la comunidad

Esta limitante será solo para las pruebas del prototipo, ya que a futuro en caso de que se implementase comercialmente, este deberá usar Internet satelital, sabiendo que esta tecnología está presente en gran parte de los aviones comerciales.

d. Aplicabilidad.

El prototipo que se desarrollará estará enfocado en el ámbito de la aviación comercial, es decir, dirigido a compañías aéreas que se dediquen al transporte bien sea civil o de carga; no se implementará a la aviación militar ni de ningún otro fin.

e. Desarrollo final.

El dispositivo final será entregado como un prototipo, es decir, inicialmente no se ofrecerá para algún fin comercial. Aparte del desarrollo tecnológico se obtendrán productos de investigación como la presentación de un artículo o participación en evento científico.

IV. OBJETIVOS

1. OBJETIVO GENERAL

Desarrollar un dispositivo capaz de monitorear los sistemas básicos de navegación en una aeronave, con el fin de ofrecer un soporte en tiempo real a los pilotos en caso de emergencia desde una estación base.

2. OBJETIVOS ESPECÍFICOS

- Obtener datos de la mayoría de los sistemas de una aeronave arrojados por un prototipo de vuelo RPAS.
- Implementar un protocolo de comunicación inalámbrica para la creación de una aplicación capaz de adquirir y monitorear datos de una aeronave en vuelo.
- Diseñar e implementar un prototipo que adquiera variables de sensores de una aeronave (RPAS) y enviarlos a una página web.

V. MARCO REFERENCIAL

1. ANTECEDENTES

a. Estudio de factibilidad para la creación de una empresa que presta el servicio de sistema de localización y monitoreo de pequeñas y medianas aeronaves (Buitrago Zuluaga & Raigosa Figueroa, 2016)

En el trabajo desarrollado por (Buitrago Zuluaga & Raigosa Figueroa, 2016) se busca localizar y monitorear aeronaves mediante los diferentes sistemas de geolocalización, los autores también desarrollan un estudio de la factibilidad explicando brevemente cómo funcionan algunos tipos de radares utilizados en la aviación colombiana, su déficit en las comunicaciones entre controlador aéreo y los pilotos.

El producto sobre el cual empiezan a trabajar es el sistema de localización de cada aeronave, algo trascendental que viene desde los comienzos de la aviación comercial, es allí donde surgió la necesidad de localizar las aeronaves y por ende crear un instrumento conocido como RADAR, el cual es un sistema que usa ondas electromagnéticas para medir distancias, altitudes, direcciones, entre otros.

b. Plan de negocios para el desarrollo de un sistema automatizado de monitoreo de vuelo para empresas de transporte aéreo (Echeverría, 2010)

La empresa ProAction System en su plan de negocios busca un financiamiento para desarrollar un sistema automatizado de monitoreo de vuelo con el fin de lograr implementarlo en algunas empresas de transporte aéreo, es una actualización a su sistema actual; aquí la empresa quiere desarrollar un producto que llaman “AirSystem V 5.0” con el fin de brindar una herramienta para

la toma de decisiones en el Centro de Control de las aerolíneas, el dispositivo que plantean es capaz de emitir información relacionada con la bitácora de vuelo, tripulación de vuelo, consumo de combustible, rutas regulares de la aeronave, y adicional a eso determina la ubicación actual de la aeronave. Este sistema integra un nuevo subsistema al monitoreo de vuelo de las operaciones aéreas, el sistema informático se desarrollará con herramientas de software Libre de última tecnología basadas en la web, permite un fácil acceso por medio de un navegador de internet.

c. Plataforma web para un sistema distribuido de telemetría de un avión no tripulado (Caballero & Marinelli, 2015)

En el desarrollo de sistemas ANT (Aviones No Tripulados) es importante conocer el estado de las variables de la aeronave durante las fases de diseño, calificación y operación de esta. Por este motivo el subsistema de telemetría de un sistema ANT, debe distribuir la información de las variables de la aeronave para que las mismas puedan ser monitoreadas por diferentes actores y en distintos niveles de toma de decisión. Es aquí donde surge la necesidad de diseñar un subsistema de telemetría que sea capaz de diseminar la información de las variables de la aeronave a distintos puntos geográficamente espaciados, con un tiempo de retraso menor a los 1200 ms y que pueda ser accedido desde cualquier sitio remoto con conexión a Internet. El principal desafío que enfrentaron en este proyecto fue la implementación de la solución mediante herramientas que permitieran entregar desde una plataforma embebida GUI que presta flexibilidad para su operación.

El subsistema distribuido de telemetría para un sistema ANT consta de 30 variables principales y 48 secundarias. Todas las variables deben ser monitoreadas en tiempo real durante las fases de

diseño, calificación y operación del sistema con un retardo máximo admisible de 1200 ms entre segmento aéreo y el edificio de ingeniería u oficina de control.

La arquitectura que utilizan en el sistema se compone por varias capas:

Capa Física: es el enlace aire-tierra toma los datos de telemetría de la aeronave y trasmitirlos a tierra, seleccionaron el módulo XBee XTend, en la aeronave el módulo trasmisor tiene una antena omnidireccional de 3dBi y el receptor una antena de 8 dBi, esto garantiza un enlace de hasta 38 Km.

Estación Terrena: La estación terrena brinda a los usuarios responsables de la operación de un sistema ANT la principal herramienta de monitoreo y control de la aeronave, proporcionando una GUI que permite visualizar en pantalla las variables de vuelo definidas en el apartado de requerimientos del presente artículo.

Servidor TCP/IP: está montado sobre una plataforma embebida NXP LPC1768, con una arquitectura ARM y un núcleo Cortex-M4 [Y01]. Este módulo, recibe la telemetría por medio de una puerta serie y la almacena en un buffer circular de la memoria de acceso aleatorio.

El Servidor Web: proporciona el medio para visualizar los datos de la aeronave desde cualquier computadora con acceso a internet. El servidor se encuentra embebido en una plataforma comercial Raspberry Pi basada en arquitectura ARM y núcleo Cortex-A7. En primera instancia, un cliente TCP/IP de la plataforma Raspberry Pi, establece conexión con el servidor TCP/IP basado en módulo NXP LPC1768 y recibe los datos correspondientes a las variables de telemetría de la aeronave durante la fase de ensayos en vuelo.

Este documento nos proporciona una idea más generalizada de cómo será estructurado nuestro sistema, ya que es uno de los documentos del estado del arte que más se asemeja a nuestro

proyecto, con la única diferencia de que en el proyecto nos hablan de un vehículo aéreo no tripulado, mientras que para el nuestro se plantea para vehículos tripulados teniendo igual un prototipo no tripulado para las pruebas.

d. Secure aircraft communications addressing and reporting system (ACARS) (Estados Unidos Patente nº US 6.677.888 B2, 2004)

Esta invención se relaciona con el trabajo realizado como parte del programa de Ciencia y Tecnología de Uso Dual (DUS & T) según el acuerdo de laboratorio de Investigación de la Fuerza Aérea. En este documento se habla de dispositivos y métodos de comunicación de aeronave a tierra, y en particular a un enlace de datos seguros entre la aeronave y la estación base basándose en tecnología del sistema de direccionamiento e informe de aeronaves conocido como ACARS. Esta solución que se ofrece protege la transferencia de información aeronáutica de extremo a extremo a través del enlace de datos ACARS utilizando técnicas criptográficas basadas en estándares vanguardistas.

La invención muestra unas técnicas específicas para reducir la saturación de frecuencia ACARS mediante un esquema único de codificación y decodificación combinados con un algoritmo de compresión de datos, adicional a esto se proporciona una solución para encriptar el encabezado del protocolo ACARS sin requerir ningún cambio en el equipo.

2. MARCO HISTORICO

Comunicación inalámbrica: En 1864 James Clerk Maxwell propuso la teoría electromagnética de la luz la cual consiste en que una onda electromagnética en un campo magnético tiende a propagarse, luego Heinrich Rudolf Hertz en el año de 1887 logra trasmitir electricidad en forma de ondas electromagnéticas y dejó entrever la posibilidad de producir y transmitir ondas eléctricas a distancia y recuperarlas mediante un aparato receptor.
(Timetoast, s.f.)

IoT (Internet de las Cosas): El internet de las cosas empieza a evolucionar con la convergencia de tres tecnologías distintas; comunicación inalámbrica, sistemas microelectromecánicos (MEMS) y microservicios de internet. Pasando de la comunicación máquina a máquina a volverse una red de sensores que conectan personas, sistemas y otras aplicaciones con el objetivo de la recopilación y visualización de datos.
(Thing Big, s.f.)

Microcontrolador: Fueron inventados por Texas Instruments en la década de los años 70, casi a la par con el primer microprocesador, básicamente los primeros microcontroladores era microprocesadores con funciones de memoria, como una RAM y ROM. Inicialmente el microcontrolador solo lo usaban en Texas Instruments desde 1972 hasta 1974, con el paso de estos años este fue mejorado y puesto a la venta como el TMS1000, con el paso de los años estos dispositivos han ido evolucionando y hoy dia ya se pueden ver aplicaciones en varios sectores

como lo es en el área automotriz, iluminación y dispositivos de consumo de baja potencia. En la figura 1 se observa la imagen de un microcontrolador ATmega644 fabricado por Microchip. (Aycock, s.f.)

Raspberry Pi: La idea inicial fue planteada en 2006, diseñado por un grupo de la Universidad de Cambridge con el objetivo de fomentar el aprendizaje de las ciencias computacionales en niños, estos primeros modelos planteados se basaban en el microcontrolador Atmel ATmega644(Figura 1).
(Teslabem, 2018)

Figura 1. Microcontrolador ATmega644. Fuente: (Teslabem, 2018)

No obstante, fue hasta febrero de 2012 cuando comenzaron las primeras ventas y en sus siguientes 6 meses llegaron a vender 500.000 unidades.

La Raspberry Pi 4 B (Figura 2) cuenta con una CPU BCM2711, Quad core Cortex-A72 de 64 bits a 1.5 Ghz y capacidad de ram de hasta 8GB.

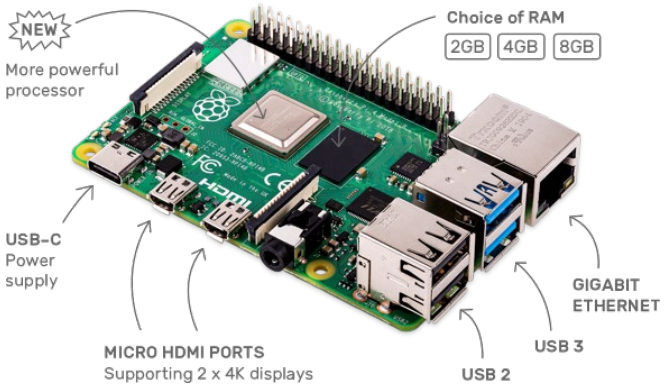


Figura 2. Raspberry pi 4 Modelo B. Fuente: (Fundacion Raspberry Pi, s.f.)

Modulo XBee: Los primeros módulos fueron introducidos bajo la marca MaxStream en 2005 y se usaban el estándar 802.15.4-2003, el tipo de comunicación era punto a punto con un ancho de banda de 250Kbit/s como máximo configurados en una topología de estrella.

(Timetoast, s.f.)

Global Position System (GPS): En 1957 la Unión Soviética lanzo al espacio el satélite Sputnik I, el cual era monitorizado mediante la observación del efecto Doppler de la señal que este trasmitía, debido a ello se comenzó a pensar que la posición del observador se podía calcular de igual modo con el efecto Doppler de una señal trasmitida por un satélite, cuya orbita estuviera determinada con precisión.

El sistema **TRANSIT** fue el primer sistema basado en satélites y su entrada en servicio fue en el año 1965, A principios de los 60 los departamentos de defensa, transporte y la agencia espacial norteamericanas empiezan a tener interés en desarrollar un sistema para determinar la posición de un objeto basado en satélites, dicho sistema tenía que cumplir los requisitos de globalidad, abarcando toda la superficie del globo; continuidad, funcionamiento continuo sin afectarle las condiciones atmosféricas; altamente dinámico, para posibilitar su uso en aviación y precisión.

Inicialmente el sistema GPS tenía el único propósito de ofrecer a las fuerzas militares de Estados Unidos la posibilidad de posicionarse de forma autónoma o individual, con un coste relativamente bajo, con disponibilidad global y sin restricciones temporales. Pero en 1984 un vuelo civil de Korean Airlines fue derribado por la Unión Soviética al ingresar por erro a espacio aéreo restringido, esto causo que el GPS se ofreciera a civiles con un cierto nivel de uso para que al final este se ofreciera globalmente y sin restricciones.

(Sierra, 2012)

Unidad de Medicion Inercial (IMU): Históricamente la navegación inercial no nace si no hasta el siglo XX pero sus inicios empezaron con la invención del giróscopo en el siglo XIX, en la segunda guerra mundial se emplearon por primera vez giróscopos y acelerómetros para guiar misiles. Despues de la guerra hubo un rapido crecimiento puesto que los primeros sistemas consitian en una triada de acelerómetros y giróspos montados en una plataforma la cual mantiene la orientación respecto a un punto.

(Universidad de Sevilla, s.f.)

3. MARCO TEÓRICO

Raspberry Pi: Es un miniordenador de bajo costo y tamaño reducido, destinado principalmente al desarrollo de proyectos y a la estimulación de la enseñanza de las ciencias computacionales.

XBee: Son pequeñas radios que cuentan con la capacidad de comunicarse inalámbricamente entre sí, estos dispositivos utilizan el protocolo de comunicación IEEE 802.15.4 el cual les permite realizar conexiones (Punto a Multipunto; o (Punto a Punto), están diseñados especialmente para aplicaciones que necesiten un alto tráfico de datos con una baja latencia.
(DIGI, s.f.)

Sensor de presión: Es un dispositivo capaz de determinar la presión atmosférica utilizando distintos principios de funcionamiento.

Sensor inercial: Dispositivo que permite medir la aceleración y la velocidad angular en 3 ejes (X,Y,Z), se utiliza principalmente en aplicaciones de captura y análisis de movimiento ya que está compuesto regularmente por un acelerómetro y un giróscopo, este segundo permite conocer la posición actual de un elemento y el acelerómetro permite conocer la velocidad.

GPS: Sistema de Posicionamiento Global originalmente conocido como Navstar GPS, dicho sistema permite posicionar cualquier objeto que se encuentre en la tierra, muchas veces alcanza a tener una precisión de milímetros, sus principales aplicaciones se presentan en la industria militar.

RPAS: Remotely Piloted Aircraft System, Es un vehículo aéreo piloteado remotamente, es decir no tripulado, estos vehículos se suelen emplear en múltiples áreas, como lo son en la industria del aeromodelismo (pilotaje y construcción de aeronaves a escala) o en la industria militar, comúnmente conocidos como UAV o vehículos aéreos no tripulados.

4. MARCO LEGAL

a. Reglamentación Colombiana para aeronaves no tripuladas

RAC 91 reglas generales de vuelo y de operación: Resolución N° 01594 del 07 de junio de 2018 (Unidad Administrativa Especial de Aeronáutica Civil, 2020)

Apéndice 13 (Operación de sistemas de aeronaves no tripuladas UAS y se incorpora a los RAC) Circular 328 AN/190 OACI desde 05 de Agosto de 2019: Según el apéndice 13 se establece la reglamentación para el uso de aeronaves no tripuladas y se especifican las categorías según su operación, para este caso la aeronave que se vuela se encuentra en la clase A (abierta) que dice : Corresponde a la operación de UAS que se encuentren dentro de las limitaciones establecidas en el párrafo (a) de la sección 2.2 de este apéndice y que cuenten con un MTOW (peso de la aeronave al momento de despegar, Maximum Take Off Weight) superior a 250 gr y hasta 25 kg, por lo cual no requiere autorización de la UAEAC dado que su operación representa un riesgo bajo.

Estos aparatos, teniendo en cuenta que en su operación pueden asimilarse a aeromodelos, estarán sujetos, además, a los requisitos y limitaciones establecidos en los literales a), b), d), e), f), g), h) e i) del numeral 4.25.8 de la norma RAC 4 de los Reglamentos Aeronáuticos de Colombia, o de aquella que en el futuro la modifique o sustituya.

VI. DISEÑO METODOLÓGICO

1. ETAPA DE INVESTIGACIÓN

a. Tipo De Investigación

Para el desarrollo del proyecto se plantea una investigación aplicada, ya que lo que se busca concretamente es dar solución a una problemática específica: Monitoreo en tiempo real de los sistemas de una aeronave con el objetivo de ofrecer un soporte en tiempo real al piloto.

En la Figura 3 se puede apreciar un diagrama general del diseño metodológico utilizado en este proyecto.

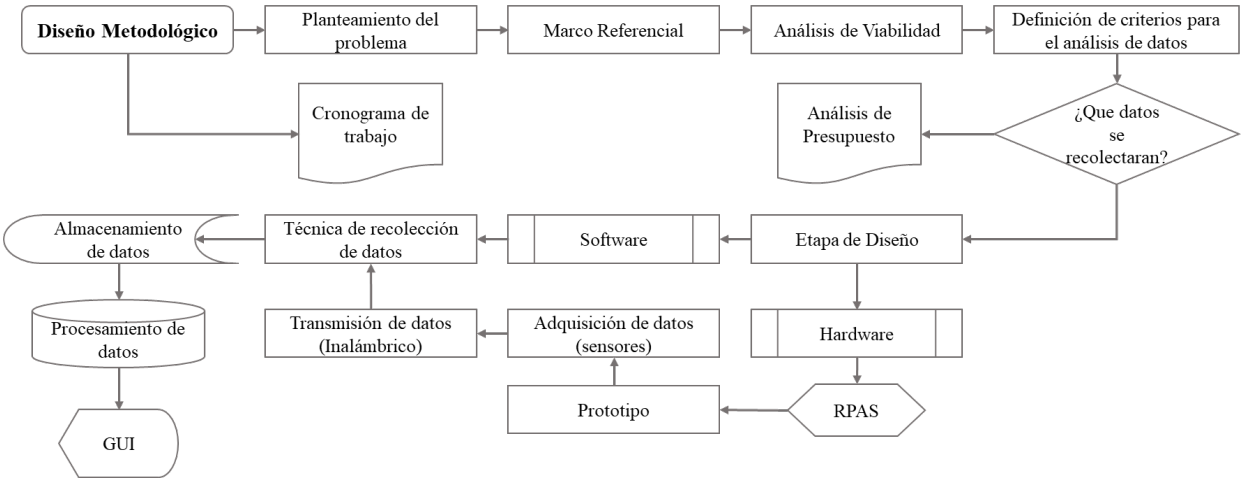


Figura 3. Diseño metodológico del proyecto. Fuente: Autor

b. Análisis Viabilidad

Se evaluará la factibilidad de realizar el proyecto, cuáles han sido sus antecedentes y qué aspectos se podrían modificar para su correcto desarrollo.

c. Evaluación De Tecnologías

Aquí se buscará qué tecnologías han trabajado en temas relacionados al proyecto y se hará una selección de cual es viable para su implementación.

d. Principios De Funcionamiento

Se investigarán los principios de funcionamiento, pues es importante conocer cómo se comporta un avión en vuelo y cuales son algunos de sus factores a mitigar, en qué sistemas de la aeronave nos podríamos concentrar para obtener su información remotamente.

e. Adquisición De Datos

Datos que se plantean adquirir para su procesamiento, tales como: velocidad, altitud, presión… entre otros, también se observaran unas técnicas de recolección y procesamiento de datos.

f. Análisis De Presupuesto

Este análisis va de la mano con la viabilidad, pues de arrojarnos un valor alto limitaría la continuación del proyecto; lo que buscaremos aquí en esta etapa es reducir costes al máximo, como reutilizar algunos de los dispositivos que se encuentran en los laboratorios de la universidad o mirar qué componentes que sean costosos se pueden remplazar por unos de un costo inferior con igual funcionalidad.

2. ETAPA DE DISEÑO

a. Software

Diseño de una GUI para la visualización de los datos de la aeronave, inicialmente se plantea adquirir los datos de un prototipo de vuelo UAV para observar su comportamiento, estos datos se enviarán a una página web para hacer posible su visualización desde cualquier dispositivo con acceso a internet.

b. Hardware

Esta sección consta de dos implementaciones, la primera de ellas es el vehículo aéreo no tripulado, pues de aquí se hace la adquisición de los datos de vuelo ya que es más complejo hacerlo en un avión real.

Posterior a ello, se diseñará el dispositivo principal del proyecto, un hardware que sea capaz de hacer dicha adquisición en tiempo real de los datos del UAV y enviarlos a una página web.

3. ETAPA DE EJECUCIÓN Y EVALUACIÓN

a. Implementación

Se unen los diseños de software y hardware.

b. Ensayos

Evaluación del comportamiento del sistema completo para hacer sus respectivas correcciones.

VII. DESARROLLO

1. CAPITULO 1: INVESTIGACIÓN

Los resultados obtenidos en la parte de investigación no solo abarcan esta sección, puesto que esto se viene realizando desde el principio del proyecto, es decir a este capítulo se le suman los ítems del marco referencial observados en la sección V.

a. Descripción general del sistema ACARS

ACARS (Aircraft Communication Addressing and Reporting System), es un sistema data link (enlace de datos) de transmisión en el rango de frecuencias VHF (Very High Frequency) de 129 MHz – 137 MHz, que básicamente ofrece un medio por el cual las aerolíneas pueden intercambiar datos y comunicarse con las aeronaves de la flota, de la misma forma que es posible intercambiar datos a través de una red digital terrestre. La principal característica de este es su la capacidad de proporcionar datos en tiempo real relacionados con el desempeño de la aeronave y así posteriormente utilizar dichos datos para planificar actividades de mantenimiento.

Estos datos no necesariamente viajan siempre por VHF, sino que también se prevé el uso de HF (High Frequency) y SATCOM (Satellite Communications). Los mensajes son conocidos como enlaces ascendentes y descendentes, los cuales facilitan una comunicación de doble vía para aplicaciones como ATIS (Automatic Terminal Information Service), autorizaciones, informes de tiempo / turbulencia, retrasos de vuelo, también se incluyen informes de datos del motor, posición, carga de pasajeros, combustible abordo.

Esta información suele ser solicitada por la empresa y se recuperan en intervalos de tiempo, estos intervalos y la cantidad de datos que se envían varían dependiendo de la aerolínea, cabe

destacar que antes del ACARS este tipo de información habría sido transferida a través de comunicación por voz VHF.

(Hernandez Carmona & Lopez Monjaras, 2012)

b. Descripción general del internet satelital

Al hablar de comunicación satelital se suele tener en mente dos conceptos, enlaces ascendentes (tierra – satélite) y enlaces descendentes (satélite – tierra), el objetivo de esta comunicación es permitir que estaciones principales (emisor y receptor) se comuniquen entre si utilizando los satélites como estaciones repetidoras.

Internet satelital es un método de conexión a internet mediante ondas electromagnéticas utilizando como principal medio de comunicación un satélite y una estación terrena (Figura 4), durante el transcurso de los años este ha ido evolucionando y creciendo en la industria en general, ya que su cobertura permite conectividad en zonas remotas. Cabe destacar que su principal desventaja suele ser una latencia baja, pero a medida que pasa el tiempo estos sistemas siguen mejorando y a futuro el internet normal como se conoce hoy día pasara a un segundo plano (Duarte Muñoz, 2014). En la Figura 4 se puede observar de manera general el funcionamiento de internet satelital.

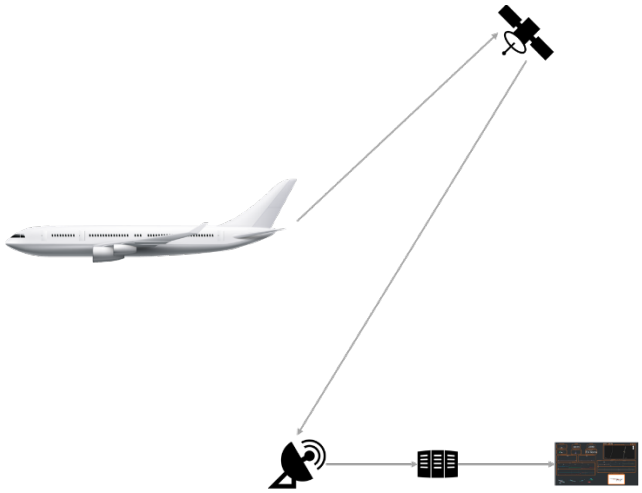


Figura 4. Funcionamiento Internet Satelital. Fuente: Autor

La comunicación satelital hace uso de varias bandas de frecuencia, dentro de las cuales se aprecia el uso de VHF, UHF entre otros, pero para el caso de uso de internet satelital se requiere una banda mucho mayor para ofrecer un mejor ancho de banda, aquí es donde entran las bandas C,X y Ku, donde cada una de ellas tiene una aplicación diferente, por ejemplo la banda C trabaja en frecuencias de 4 – 8 GHz y suele emplearse para la televisión satelital, luego se encuentra la banda X (8 – 12 GHz) su uso es exclusivo para aplicación militar y después se encuentra la banda Ku, que trabaja en frecuencias de 12 a 18 GHz siendo una de las más versátiles del espectro de microondas al proporcionar servicios de banda ancha de buena calidad. El internet satelital alcanza velocidades desde 25 Mbps hasta 100 Mbps inclusive más.

(Duarte Muñoz, 2014)

El proyecto plantea un sistema de monitoreo en tiempo real con el uso de internet satelital, ya que como se menciona las comunicaciones del sistema anterior (ACARS) dependen de la curvatura de la tierra y así las aeronaves vuelen muy alto en algunos puntos del mundo no es común volar regularmente como lo es en el norte de Canadá, adicional a esto se destaca que el sistema RTAMS

no plantea realizar las mismas funciones de un ACARS y en ningún momento reemplazarlo, sino que es un sistema capaz de ofrecer un complemento a la hora de hacer la adquisición de algunos datos que el sistema ACARS no suele obtener, y así lograr cumplir el objetivo de ofrecer al piloto un soporte en tiempo real en caso de emergencias o averías en la aeronave.

Para el desarrollo del proyecto no es posible la implementación en una aeronave real, por ello se plantea el uso de un prototipo a escala (aeromodelo) y adicional se selecciona un protocolo de comunicación inalámbrica para comunicar el aeromodelo con una estación terrena y así simular una comunicación satelital para poder ver que errores se podrían presentar.

c. Protocolos de comunicación inalámbrica

A continuación, se describen algunos de los protocolos de comunicación inalámbrica que se tuvieron presentes para poder probar la comunicación en el modelo a escala con la estación base.

- **SIGFOX:**

Es una red de IOT que plantea ofrecer conectividad a bajo consumo y costo, aportando soluciones M2M (Maquina a Maquina).

Esta tecnología depende de operadores encargados en cada país, para el caso de Colombia lo realizan WDN group (Wireless Network Developments) y ha creado una red de largo alcance y baja velocidad para permitir la comunicación entre los dispositivos conectados sin necesidad de estar directamente ligados a una cobertura de red móvil.

Su funcionamiento depende de otra red distinta que trabaja en 868 MHz, es decir que su cobertura es limitada a zonas donde se encuentren antenas SIGFOX, generando la necesidad de

que si se quiere más cobertura se debe realizar la instalación de equipos repetidores, también cuenta con una interfaz web que permite darle de alta a los equipos ya que estos funcionan por un identificador único en vez de una tarjeta sim (Ferrer, Qué es Sigfox, s.f.).

Para este caso la principal limitante no solo es la cobertura, sino también el número de paquetes que se pueden enviar a la red por día, ya que esta solo permite enviar 140 paquetes con un tamaño de 12 bytes cada uno y también presenta una velocidad de transmisión baja por ello se descarta el uso de este protocolo.

- **LORA:**

(Long RAnge), esta tecnología permite que los dispositivos que se encuentren conectados intercambien paquetes de datos de larga distancia a baja velocidad y con bajo consumo energético.

Su funcionamiento es como una tecnología de modulación para así lograr reducir costos y consumos de energía; ya que la duración de la batería puede llegar a los 3 o 5 años dependiendo del caso de uso. Una de las ventajas que presenta este protocolo es que su señal tiene la capacidad de atravesar edificios fácilmente para así lograr llegar hasta rincones apartados como subterráneos o sótanos (Ferrer, Qué es Lora y Lorawan, s.f.).

Al igual que SIGFOX tambien se descarta, puesto que para su implementación en aplicaciones de tiempo real se encuentra limitada por el numero de paquetes que se pueden enviar.

- **LTE/4G:**

Se basa en la transmisión inalámbrica de datos de banda ancha que principalmente se usa para el acceso a la telefonía móvil o dispositivos portátiles de internet, sin embargo, su cobertura está

asociada al operador de telefonía, los costos de implementación y mantenimiento suelen ser altos y su alto consumo de energía, hacen que solo se tenga en cuenta para el envío de información desde la estación base hasta la página web.

- **ZIGBEE Y XBEE:**

Zigbee es una tecnología inalámbrica que opera en las bandas libres ISM (Industrial, Scientific & Medical) de 2.4 GHz, alcanza velocidades de transmisión de 250 Kbps y un rango de 10 a 75 metros, esta tecnología se caracteriza por operar en redes de alta densidad ayudando así a la confiabilidad de comunicación, ya que entre más nodos existan más rutas alternas hay para garantizar que un paquete llegue a su destino. De Zigbee se desprende una tecnología llamada XBEE la cual modifica el estándar de zigbee y lo acomoda en un paquete más pequeño, los módulos desarrollados por XBee son soluciones integradas que brindan un medio de comunicación inalámbrica para la interconexión entre dispositivos utilizando el protocolo IEEE 802.15.4, esto permite la creación de redes punto a punto y multipunto, los XBee están diseñados para aplicaciones que requieren un alto tráfico de datos y una latencia reducida, estos logran aumentar su alcance dependiendo la antena que se les acople (Garcia, 2012).

Dentro de las ventajas que se encuentran al utilizar los módulos XBee se destaca que estos ofrecen una comunicación inalámbrica de hasta 16 Km ya que no dependen de otra empresa sino de la antena y modulo que se tenga, tampoco presenta un límite de paquetes por día y por eso esta tecnología fue la seleccionada para la comunicación inalámbrica.

Las referencias de la tabla de a continuación dependen de las regulaciones de telecomunicaciones de cada país.

	SIGFOX	LORA	4G/LTE	ZIGBEE
Frecuencia	868 MHz	902 – 928 MHz	1700 – 2600 MHz	850 – 960 MHz
Cobertura	Medio	Medio	Alto	Medio
Límite de transferencia	Si	Si	No	No
Consumo de energía	Bajo	Bajo	Alto	Medio
Ventajas	Aplicación optima en dispositivos que transmiten con poca frecuencia Consumo bajo de energía	Aplicaciones en entornos pequeños Bidireccionalidad Correcto funcionamiento cuando están en movimiento Consumo de energía	Velocidad de transmisión Cobertura en gran parte de Colombia	Ideal para conexiones Punto a punto y multipunto Económico Buena movilidad
Desventajas	Cobertura, depende mucho de donde estén ubicadas las antenas Baja movilidad Límite de transferencia de datos	Latencia alta Cobertura Límite de transferencia de datos	Consumo de energía alto Costos Así como la cobertura es una ventaja, en algunos casos también es desventaja.	Alcance de transmisión depende de la antena

Tabla 1. Tabla comparativa protocolos de comunicación inalámbrica. Fuente: Autor

2. CAPITULO 2: DISEÑO

Para el diseño del prototipo hay que tener en cuenta que no se pueden agregar muchos sensores, esto debido a que el aeromodelo no soporta demasiado peso y hay que tener en cuenta la forma en la que este se distribuirá sobre el mismo. Concepto básico de peso y balance.

a. Diseño del Prototipo

En la figura 5 se puede observar el diseño del prototipo el cual cuenta con. Un microcontrolador el cual se encarga de hacer la adquisición de los datos enviados por los sensores de Presión, GPS e IMU (Sensor Inercial), a su vez este se encarga de realizar él envió de los datos por medio de una comunicación inalámbrica con los módulos XBee, una vez los datos llegan al XBee receptor el cual se encuentra en la Raspberry Pi se podrán visualizar en una aplicación de escritorio diseñada con Python y de allí se enviarán los datos a la página web. La comunicación de los módulos XBee es solo necesaria para la fase de pruebas, puesto que si se llegase a implementar el prototipo en una aeronave real esta tiene que contar previamente con una conexión a internet satelital.

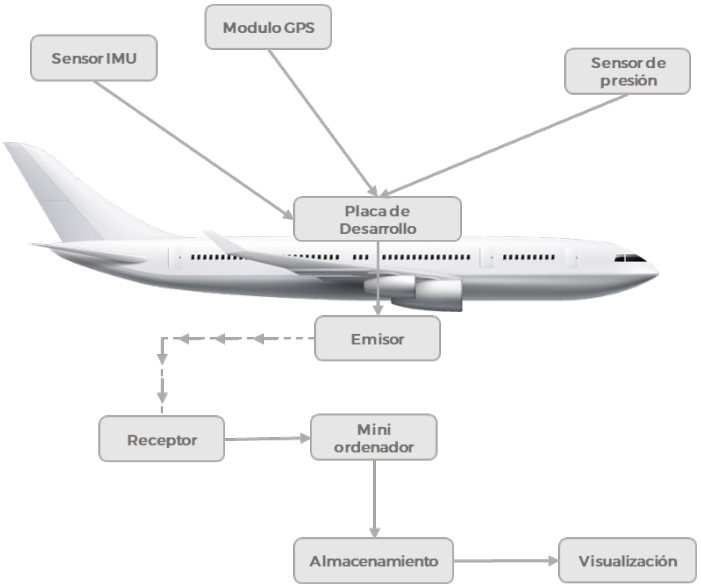


Figura 5. Diseño del Prototipo. Fuente: Autor

b. Sensores y Módulos de adquisición de datos

En esta sección se abordará la selección de los sensores y módulos requeridos para el prototipo, todo esto teniendo en cuenta la investigación previamente realizada

- Placa de desarrollo: Arduino Nano V3

La placa de desarrollo encargada de hacer la adquisición de los datos para posteriormente enviarlos por comunicación inalámbrica a la aplicación de escritorio es un Arduino Nano V3 (Figura 6), el cual gracias a su tamaño y practicidad permite realizar fácilmente lo que se desea. En la Tabla 2 se aprecian las características de este dispositivo.

Placa	**Microcontrolador**	**Memoria Flash**	**Memoria SRAM**	**Memoria EEPROM**
Arduino Nano V3	Atmel ATmega328	32 KB	2 KB	1 KB
Voltaje de operación	**Velocidad De Reloj**	**I/O Digitales**	**Entradas Análogas**	**Canales PW**
5 V	16 MHz	14	8	6

Tabla 2. Características Arduino Nano V3. Fuente: Autor

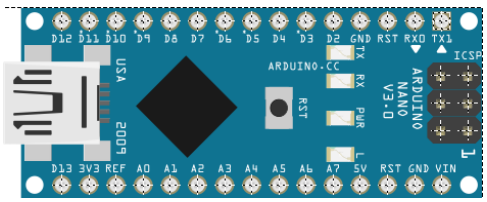


Figura 6. Arduino Nano V3. Fuente: (Fritzing, 2021)

- Sensor Inercial: MPU6050

Inicialmente se había planteado el módulo MPU9250 pero se observó que no era necesaria la adquisición de la magnitud para obtener el rumbo puesto que se puede sacar del módulo GPS y el costo incrementaba un poco, es por ello que se escoge el MPU6050 (Figura 7) ya que este permite medir el movimiento en 6 grados de libertad, combinando un giroscopio de 3 ejes y un acelerómetro también de 3 ejes en un mismo chip (Tabla 3), adicional la interfaz de comunicación que ofrece es I2C la cual permite configurar una variedad de sensores y controlarlo solo por dos cables.

Sensor Inercial	**Rango Acelerómetro**	**Rango Giróscopo**	**Grados de libertad**	**Interfaz de Comunicación**	**Voltaje de Alimentación**
MPU 6050	±2,4,8 y 16 g	±250,500,1000 y 2000 grad/seg	6	I2C	3-5V

Tabla 3. Características MPU6050. Fuente: Autor

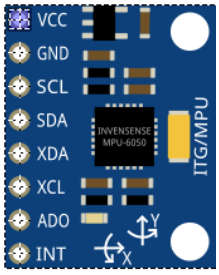


Figura 7. MPU6050. Fuente: (Fritzing, 2021)

- Sensor de Presión: BMP280

El sensor de presión barométrica seleccionado para este proyecto es un BMP280 (Figura 8) el cual cuenta con una precisión de 1hPa según hoja técnica, una comunicación I2C (Tabla 4) y un bajo costo. También permite la adquisición de temperatura con un rango de precisión de 1 ºC

Sensor De Presión	**Rango de Presión**	**Precisión Absoluta**	**Precisión de Temperatura**	**Interfaz de Comunicación**	**Voltaje de Alimentación**
BMP280	300 – 1100 hPa	1 hPa	1 ºC	I2C	1.8 – 3.3V

Tabla 4. Características BMP280. Fuente: Autor

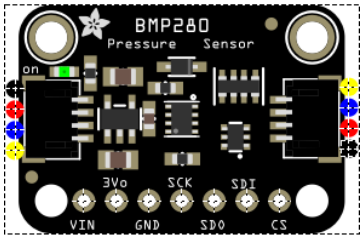


Figura 8. BMP280. Fuente: (Fritzing, 2021)

- Modulo GPS: NEO 6m V2

El módulo GPS seleccionado fue un NEO 6 V2 (Figura 9) ya que su relación costo beneficio es buena y su tamaño reducido hace que sea portable, la comunicación que utiliza es UART (Tabla 5).

Modulo GPS	Altitud Máxima	Velocidad Máxima	Precisión	Interfaz de Comunicación	Voltaje de Alimentación
NEO 6 V2	18.000 mts	515 m/s	Posición:3 mts Velocidad: 0.1m/s	UART	3.3 – 5V

Tabla 5. Características NEO 6m V2. Fuente: Autor

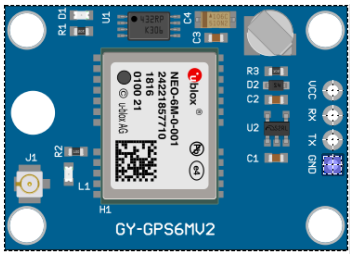


Figura 9. NEO 6m V2. Fuente: (Fritzing, 2021)

A el módulo se le colocara una antena GPS SMA (Figura 10) con una frecuencia de 1575 MHz, esta antena permite mejorar la señal de recepción del GPS.

Figura 10. Antena GPS SMA. Fuente: (Mercado Libre, s.f.)

- Modulo Comunicación inalámbrica: XBee Pro S3B

Como los datos se piensan enviar a una página web y se hace necesario que dichos sean correctos, se requiere que un dispositivo de comunicación inalámbrica envíe estos datos a una estación base y desde allí se envíen a la página web una vez procesados en la aplicación de escritorio, es por ello por lo que se seleccionó un módulo XBee Pro S3B (Figura 11) el cual permite una comunicación inalámbrica de hasta 15 Km dependiendo el tipo de antena, su frecuencia de trabajo es de 900 MHz. En la Tabla 6 se pueden apreciar las especificaciones técnicas del dispositivo

Modulo	**Frecuencia**	**Velocidad de transmisión**	**Interfaz de Comunicación**	**Voltaje de Alimentación**
XBee Pro S3B	900 MHz	20 Kbps	UART	3 – 3.6V

Tabla 6. Características XBee Pro S3B. Fuente: Autor

Figura 11. XBee Pro S3B. Fuente: (Fritzing, 2021)

Las especificaciones técnicas de la antena (Figura 12) que se utilizara junto al XBee se pueden apreciar en la Tabla 7.

Antena	Rango de Frecuencia	Ganancia	Impedancia de Entrada	Potencia
RP-SMA	824 – 960 MHz	10 dBi	50 Ohm	100 W

Tabla 7. Características Antena para XBee. Fuente: Autor

Figura 12. Antena RP-SMA 10 dBi. Fuente: Autor

- Miniordenador: Raspberry Pi 4 Modelo B

Se selecciono una Raspberry Pi 4 Modelo B (Figura 13), la cual cuenta con la capacidad de un ordenador, en esta se hará la visualización de los datos que se adquieran por comunicación XBee y posterior a ello se enviaran los datos a una página web.

Referencia Raspberry	**Procesador**	**GPU**	**Memoria**
Raspberry Pi 4B	Quad-core Cortex-A72	VideoCore VI 500 MHz	4 GB
Conectividad inalámbrica	**Conectividad de red**	**Puertos**	**Voltaje de Alimentación**
Wi-Fi 2.4GHz / 5GHz Bluetooth 5.0, BLE	Gigabit Ethernet	Micro HDMI,USB 2.0, USB 3.0, 40 pines GPIO.	5V 3A

Tabla 8. Características Raspberry Pi 4 Modelo B. Fuente: Autor

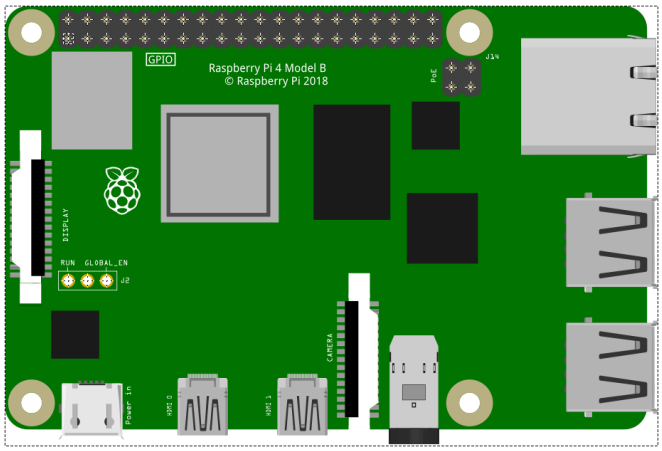


Figura 13. Raspberry Pi 4 Modelo B. Fuente: (Fritzing, 2021)

- Pantalla: Táctil HDMI 7 Pulgadas 800x400

Para visualizar los datos en la estación base se seleccionó una pantalla táctil de 7 pulgadas (Figura 14), la cual permite la interacción con el usuario para que este envíe los datos a la página

web dependiendo la aeronave que se seleccione para hacer el track de vuelo, eso es solo en la parte de pruebas ya que el sistema es totalmente autónomo.

Pantalla	**Resolución**	**Tamaño**	**Voltaje de Alimentación**
HDMI 7 Pulgadas	800 x 400 Px	7 Pulgadas	5V

Tabla 9. Características Pantalla Táctil. Fuente: Autor

Figura 14. Pantalla Táctil 7 Pulgadas HDMI. Fuente: (Vistronica, s.f.)

3. CAPITULO 3: IMPLEMENTACIÓN PREVIA

Una vez seleccionado los módulos y sensores que irán en el prototipo se empieza a realizar el diseño esquemático del sistema (Figura 15).

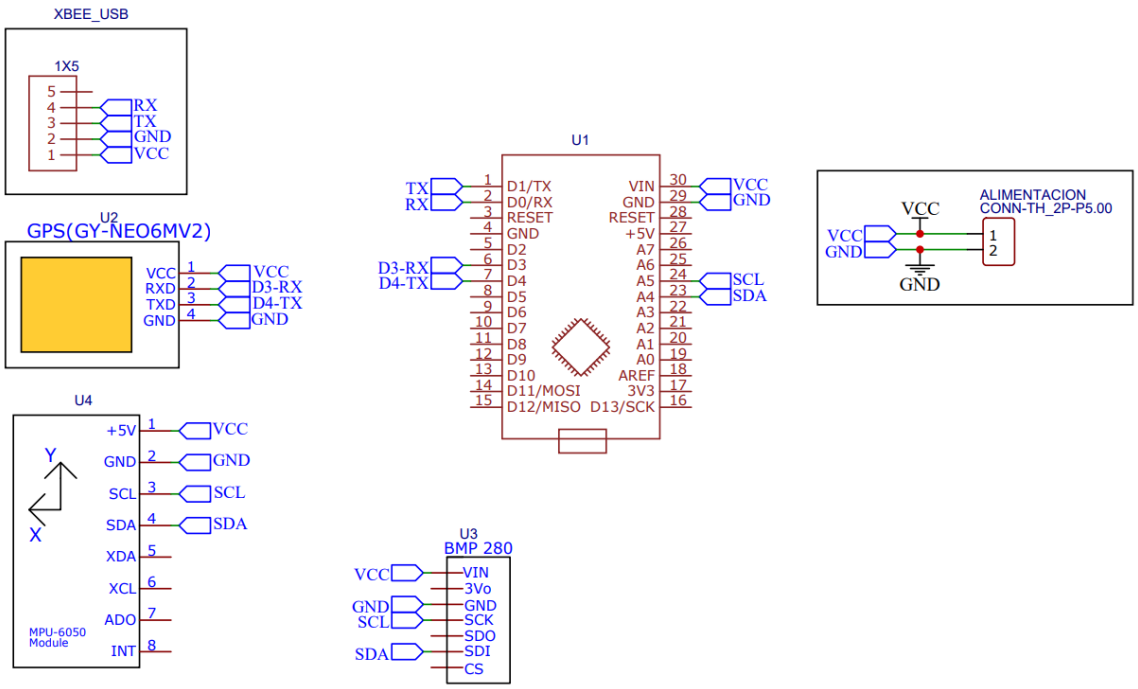


Figura 15. Esquemático Sistema RTAMS. Fuente: Autor

a. Adquisición de la presión: BMP280

Para el sensor de presión se utilizará la librería Adafruit_BMP280 la cual facilita la adquisición de los datos de este sensor, dicho sensor funciona por comunicación I2C es decir con tan solo dos cables se puede adquirir estos datos y a su vez se pueden aprovechar para adquirir datos de otro sensor. La conexión con el sistema se muestra en la Figura 16 y en la Tabla 10 se muestra la conexión de pines.

Color	Arduino Nano V3	BMP280
Rojo	5V – Vin	Vin
Negro	GND	GND
Azul	SCL	SCK
Morado	SDA	SDI

Tabla 10. Conexión pines Sensor BMP280 con Arduino. Fuente: Autor

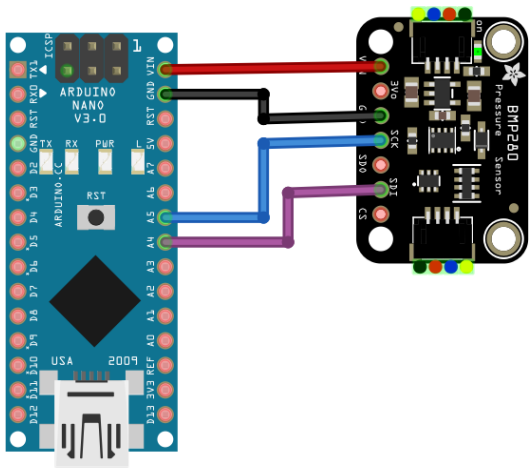


Figura 16. Conexión Arduino y Sensor de Presión. Fuente: Autor

El código para hacer la adquisición de los datos se muestra en la Figura 17.

```
#include <Adafruit_BMP280.h>
#include <Adafruit_Sensor.h>

Adafruit_BMP280 bmp;

void setup() {
  Serial.begin(9600);
  bmp.begin();
  bmp.setSampling(Adafruit_BMP280::MODE_NORMAL,
                  Adafruit_BMP280::SAMPLING_X2,
                  Adafruit_BMP280::SAMPLING_X16,
                  Adafruit_BMP280::FILTER_X16,
                  Adafruit_BMP280::STANDBY_MS_500);
}

void loop()
{
  Serial.println(bmp.readTemperature());
  Serial.println(bmp.readPressure()*0.01);
  Serial.println(bmp.readAltitude(1013.25));
}
```

Figura 17. Código Arduino BMP280. Fuente: Autor

b. Adquisición datos Sensor Inercial: MPU6050:

Con el MPU6050 se hará uso de la librería MPU6050_light (Figura 19) la cual permite hacer la calibración del sensor cada vez que este se vaya a utilizar, así se asegura que los datos siempre estén calibrados antes de hacer su adquisición; también se pueden separar los datos que lleguen en 3 ejes (X,Y,Z) y convertir directamente los datos de giro directamente en pitch, yaw y roll, en la tabla 11 se aprecia la conexión con pines y en la Figura 18 se muestra la conexión.

Color	**Arduino Nano V3**	**MPU6050**
Rojo	5V – Vin	Vin
Negro	GND	GND
Azul	SCL	SCL
Morado	SDA	SDA

Tabla 11. Conexión pines Sensor MPU6050 con Arduino. Fuente: Autor

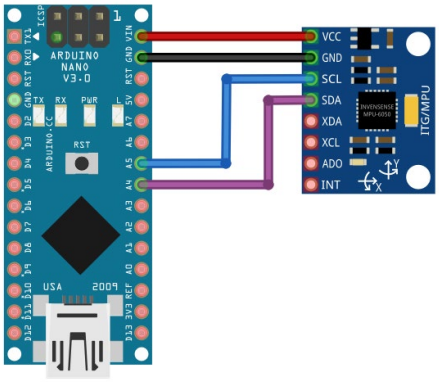


Figura 18. Conexión Arduino y Sensor Inercial. Fuente: Autor

```
#include <MPU6050_light.h>
#include <Wire.h>

MPU6050 mpu(Wire);

void setup() {
  Serial.begin(9600);
  Wire.begin();
  mpu.begin();
  mpu.calcOffsets(true,true);
}

void loop() {
  mpu.update();
  Serial.println(mpu.getAccX());
  Serial.println(mpu.getAccY());
  Serial.println(mpu.getAccZ());
  Serial.println(mpu.getAngleX());
  Serial.println(mpu.getAngleY());
  Serial.println(mpu.getAngleZ());
  Serial.println(mpu.getTemp());
}
```

Figura 19. Código Arduino MPU6050. Fuente: Autor

c. Adquisición de los datos GPS: NEO 6m V2

El módulo GPS hace la adquisición de datos diferente al MPU6050 y el BMP280, para este caso se utiliza la comunicación serial; como los datos una vez adquiridos se enviarán posteriormente por serial mediante el módulo XBee, se hace necesario el uso de la librería SoftwareSerial la cual permite la comunicación serie en otros pines digitales del Arduino, usando el software para replicar la funcionalidad.

También se hace uso de la librería TinyGPS la cual permite decodificar la trama que llega del GPS y adquirir los datos de latitud, longitud y demás por separado. La conexión del GPS con el Arduino se muestra en la Figura 20 en donde se apreciará la conexión Rx y Tx en los pines 3 y 4 del Arduino.

Color	**Arduino Nano V3**	**GPS NEO 6m V2**
Rojo	5V – Vin	Vin
Negro	GND	GND
Gris	D3	Rx
Marrón	D4	Tx

Tabla 12. Conexión pines Sensor GPS NEO 6m con Arduino. Fuente: Autor

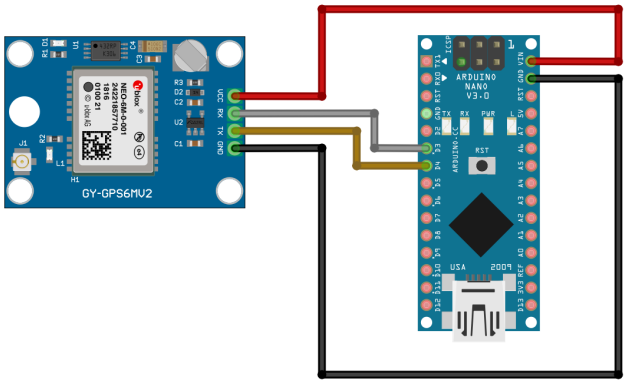


Figura 20. Conexión Arduino y GPS NEO 6m. Fuente: Autor

```
#include <SoftwareSerial.h>
#include <TinyGPS.h>

TinyGPS gps;
SoftwareSerial serialgps(4,3);

int year;
byte month, day, hour, minute, second, hundredths;
unsigned long chars;
unsigned short sentences, failed_checksum;

void setup()
{
  Serial.begin(9600);
  serialgps.begin(9600);

  Serial.println("");
  Serial.println("GPS GY-GPS6MV2 Leantec");
  Serial.println(" ---Buscando senal--- ");
  Serial.println("");
}

void loop()
{
  while(serialgps.available())
  {
    int c = serialgps.read();

    if(gps.encode(c))
    {
      float latitude, longitude;
      gps.f_get_position(&latitude, &longitude);

      Serial.print("A");
      Serial.print(latitude,4);
      Serial.print("B");
      Serial.print(longitude,4);
      Serial.print("C");
      gps.crack_datetime(&year,&month,&day,&hour,&minute,&second,&hundredths);
      Serial.print(day, DEC);
      Serial.print("/");
      Serial.print(month, DEC);
      Serial.print("/");
      Serial.print(year);
      Serial.print("D");
      Serial.print(hour, DEC);
      Serial.print(":");
      Serial.print(minute, DEC);
      Serial.print("E");
      Serial.print(gps.f_altitude());
      Serial.print("F");
      Serial.print(gps.f_course());
      Serial.print("G");
      Serial.print(gps.f_speed_kmph());
      Serial.print("H");
      Serial.print(gps.satellites());
      Serial.print("I");
      Serial.println();
      gps.stats(&chars, &sentences, &failed_checksum);
    }
  }
}
```

Figura 21. Código Arduino GPS NEO 6m. Fuente: Autor

d. Envió de datos XBee: XBee Pro S3B

Para enviar los datos se combinaron los 3 programas mencionados anteriormente como se observa en la Figura 23 y se conectó el XBee Pro S3B a un adaptador serial para hacer el envío de datos más fácilmente (Figura 22).

Color	Arduino Nano V3	Modulo XBee
Rojo	5V – Vin	Vin
Negro	GND	GND
Verde	Rx	Rx
Amarillo	Tx	Tx

Tabla 13. Conexión pines Modulo USB XBee con Arduino. Fuente: Autor

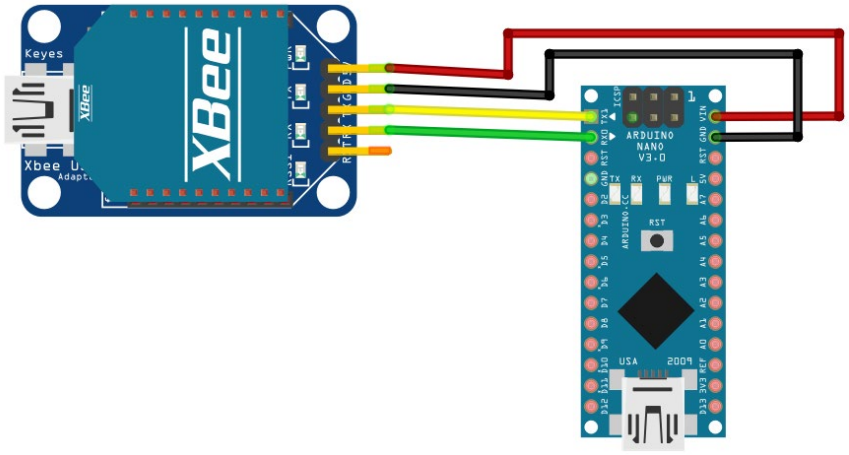


Figura 22. Conexión Arduino y Modulo USB XBee. Fuente: Autor

Para el programa final se implementaron los códigos anteriores mediante funciones para posteriormente llamarlas en el void loop mediante un switch, en la Figura 23 se puede apreciar la sección del void loop, ya que los códigos de las funciones se mostraron anteriormente en las

Figuras 17,19 y 21, a estos códigos se les hizo un ajuste para que se enviaran en forma de trama junto a una letra, la cual será posteriormente identificada para saber a qué variable corresponde el valor que llegue(Figura 24).

```
void loop()
{

  switch (actualiza)
  {
        case 0: //Modulo gps

              neo6();
              if(bandera==3)
              {
              actualiza=1;
              break;
              }
              break;


        case 1: //Sensor de presion

              bmp280();
              if(bandera=1)
              {
              actualiza=2;
              break;
              }
              break;
        case 2: //MPU

              mpu6050();
              if(bandera=2)
              {
                actualiza=0;
              break;
              }

              break;
  }
  delay(200);
}
```

Figura 23. Código Arduino programa Final. Fuente: Autor

```
void bmp280()
{
  Serial.println("I;" + String(bmp.readTemperature()));
  Serial.println("J;" + String(bmp.readPressure()*0.01));
  Serial.println("K;" + String(bmp.readAltitude(1013.25)));
}

void mpu6050()
{
  mpu.update();

  Serial.println("L;" + String(mpu.getAccX()));
  Serial.println("M;" + String(mpu.getAccY()));
  Serial.println("N;" + String(mpu.getAccZ()));
  Serial.println("O;" + String(mpu.getAngleX()));
  Serial.println("P;" + String(mpu.getAngleY()));
  Serial.println("Q;" + String(mpu.getAngleZ()));
  Serial.println("R;" + String(mpu.getTemp()));
  Serial.println("S; Trama completa");
  bandera=2;

}
```

Figura 24. Código Arduino Trama de Datos. Fuente: Autor

e. Emparejamiento Módulos XBee

Una vez se tiene la trama de datos para enviar vía comunicación serial se hace necesario emparejar el emisor y el receptor XBee, para esto se utilizará el programa XCTU que permite modificar los parámetros de los módulos, en la Figura 25 se aprecia la prueba conexión inalámbrica entre los módulos XBee pro S3B

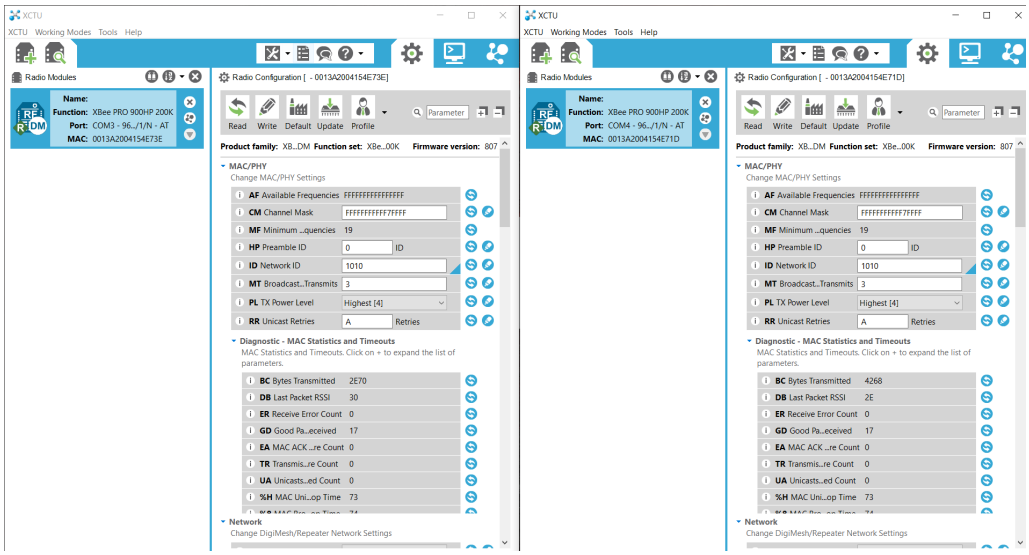


Figura 25. Programa XCTU parámetros de los módulos. Fuente: Autor

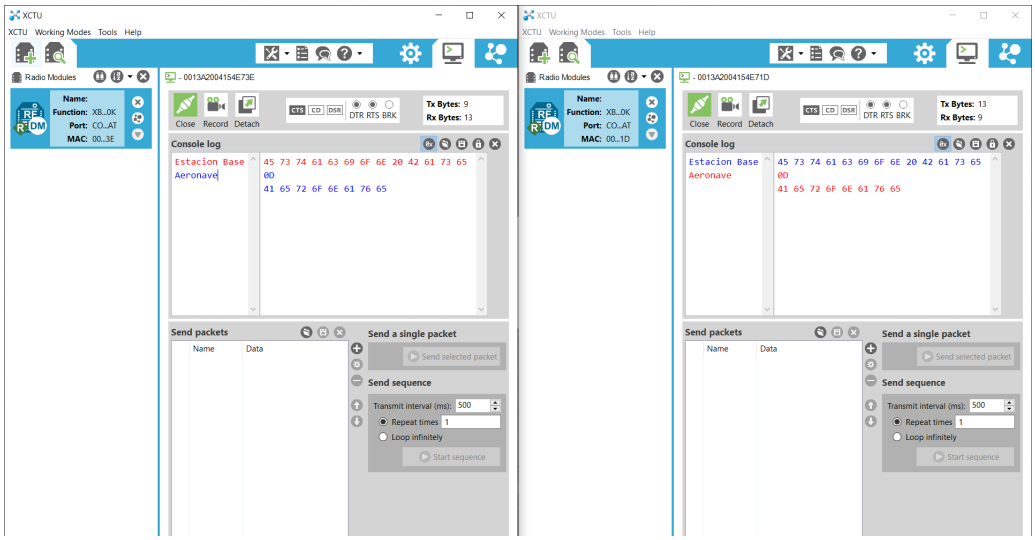


Figura 26. Comprobación Conexión Módulos XBee. Fuente: Autor

En la Figura 26 se observa la comprobación de la conexión entre los dos módulos XBee, la imagen de la derecha corresponde a el módulo que estará en la estación base, de color azul se puede ver el mensaje enviado y de color rojo el mensaje recibido, la imagen de la izquierda es del módulo que va en el avión.

4. CAPITULO 4: PROTOTIPO FINAL

Una vez seleccionados y configurados los sensores y módulos se realiza la implementación del prototipo final como se muestra en la Figura 27.

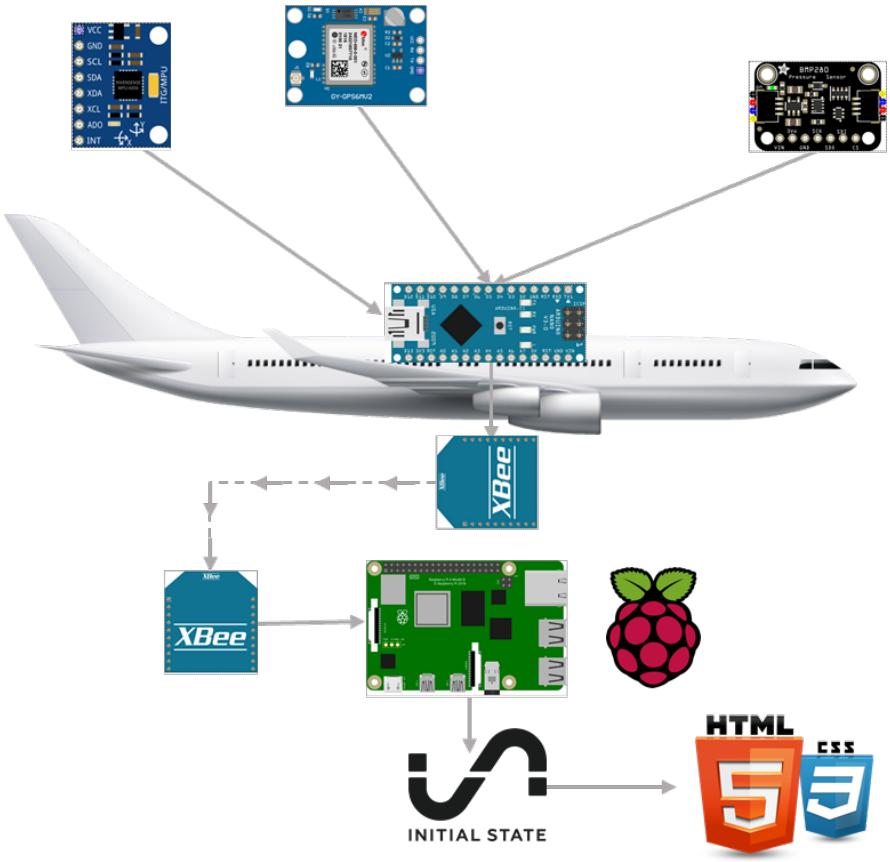


Figura 27. Diseño del Prototipo con los sensores y módulos seleccionados. Fuente: Autor

Como se observa en la Figura 27 el prototipo cuenta con los sensores mencionados anteriormente, adicional se realizó el diseño e implementación de una GUI (Figura 28) programada con Python en la cual se hará la recepción de los datos, esta GUI ira en la Raspberry Pi 4B y desde esta se enviarán los datos a la plataforma de IOT initial state, para posteriormente desarrollar una página web (Figura 29) donde se visualizarán dichos datos.

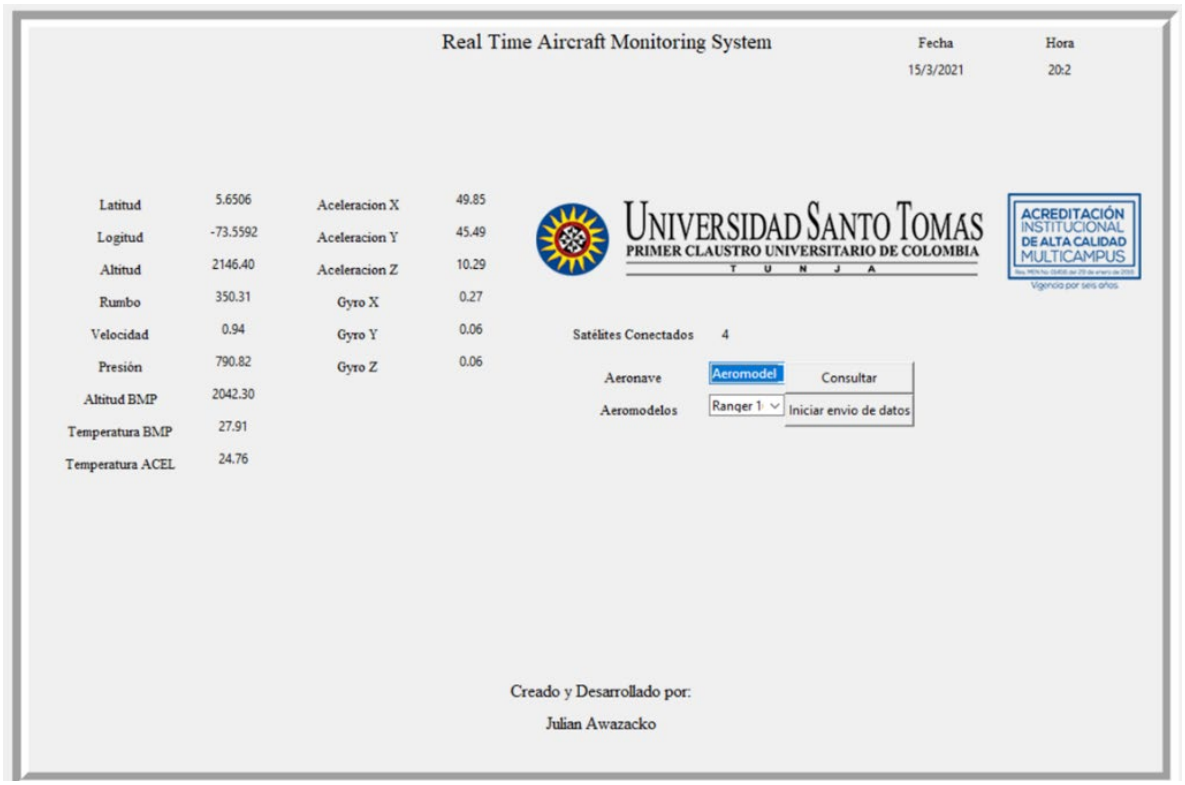


Figura 28. Diseño e Implementación de la GUI. Fuente: Autor

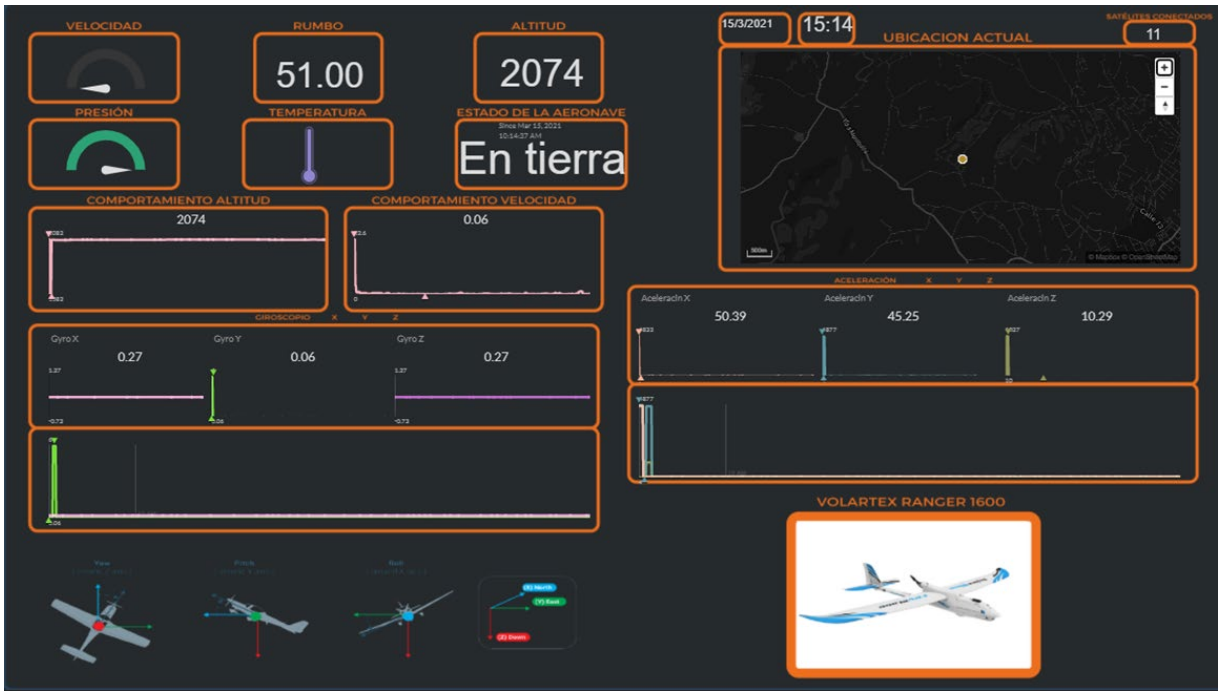


Figura 29. Diseño e Implementación Pagina Web en la plataforma IOT initial state. Fuente: Autor

En la GUI implementada se realiza la adquisición de los datos mediante comunicación serial, es por ello por lo que se hace uso de la librería Pyserial, el programa pregunta por el valor que hay inmediatamente después de la letra que se configuro en la trama de datos como se mencionó en el capítulo anterior, este código se observa en la Figura 30.

```
def get_Serial(self):
    #RECOLECTANDO DATOS DEL PUERTO SERIE Y DECODIFICANDOLOS
    while self.empezar:

        cadena = self.comunicacion.readline().decode("ascii",errors='ignore').strip()

        if cadena:
            pos = cadena.find(";")
            label = cadena[:pos]
            value = cadena[pos+1:]

            if label == "A":
                self.data_latitud.set(value)
                try:
                    self.latitud=float(self.data_latitud.get())
                except ValueError:
                    print("Lectura LATITUD incorrecta, reintentando...")
                    self.get_Serial

            if label == "B":
                self.data_longitud.set(value)
                try:
                    self.longitud=float(self.data_longitud.get())
                except ValueError:
                    print("Lectura LONGITUD incorrecta, reintentando...")
                    self.get_Serial

            if label == "C":
                self.data_fecha.set(value)

            if label == "D":
                self.data_hora.set(value)

            if label == "E":
                self.data_altitud.set(value)
                try:
                    self.altitud=float(self.data_altitud.get())
                except ValueError:
                    print("Lectura ALTITUD incorrecta, reintentando...")
                    self.get_Serial
```

Figura 30. Programación para la Adquisición de los datos por comunicación serial en Python. Fuente: Autor

Una vez adquiridos los datos se programa una interfaz gráfica (Figura 31) haciendo uso de la librería Tkinter, en la interfaz van los datos que se adquieren y también contara con unos label, combobox y un button para enviar los datos a la web.

```
def create_widgets(self):
    #CONFIGURACION DEL LABEL TITULO
    Label(self,text="Real Time Aircraft Monitoring System",fg="black",font=("Times New Roman",15)).place(x=380,y=0)

    #CONFIGURACION DEL LABEL FECHA Y HORA
    Label(self,text="Fecha",fg="black",font=("Times New Roman",10)).place(x=800,y=0,width=80,height=30)
    Label(self,text="0/0/000",textvariable = self.data_fecha).place(x=800,y=30,width=80,height=20)

    Label(self,text="Hora",fg="black",font=("Times New Roman",10)).place(x=930,y=0,width=50,height=30)
    Label(self,text="00:00",textvariable = self.data_hora).place(x=930,y=30,width=50,height=20)

    #CONFIGURACION DE LOS LABEL DE LAS DEMAS VARIABLES
    Label(self,text="Latitud",fg="black",font=("Times New Roman",10)).place(x=30,y=150,width=110,height=30)
    Label(self,text="00000",textvariable = self.data_latitud).place(x=150,y=150,width=80,height=20)

    Label(self,text="Logitud",fg="black",font=("Times New Roman",10)).place(x=30,y=180,width=110,height=30)
    Label(self,text="00000",textvariable = self.data_longitud).place(x=150,y=180,width=80,height=20)

    Label(self,text="Altitud",fg="black",font=("Times New Roman",10)).place(x=30,y=210,width=110,height=30)
    Label(self,text="00000",textvariable = self.data_altitud).place(x=150,y=210,width=80,height=20)

    Label(self,text="Rumbo",fg="black",font=("Times New Roman",10)).place(x=30,y=240,width=110,height=30)
    Label(self,text="00000",textvariable = self.data_rumbo).place(x=150,y=240,width=80,height=20)

    Label(self,text="Velocidad",fg="black",font=("Times New Roman",10)).place(x=30,y=270,width=110,height=30)
    Label(self,text="00000",textvariable = self.data_velocidad).place(x=150,y=270,width=80,height=20)

    Label(self,text="Presión",fg="black",font=("Times New Roman",10)).place(x=30,y=300,width=110,height=30)
    Label(self,text="00000",textvariable = self.data_presion).place(x=150,y=300,width=80,height=20)

    Label(self,text="Altitud BMP",fg="black",font=("Times New Roman",10)).place(x=30,y=330,width=110,height=30)
    Label(self,text="00000",textvariable = self.data_altitudBMP).place(x=150,y=330,width=80,height=20)

    Label(self,text="Temperatura BMP",fg="black",font=("Times New Roman",10)).place(x=30,y=360,width=110,height=30)
    Label(self,text="00000",textvariable = self.data_tempBMP).place(x=150,y=360,width=80,height=20)

    Label(self,text="Temperatura ACEL",fg="black",font=("Times New Roman",10)).place(x=30,y=390,width=110,height=30)
    Label(self,text="00000",textvariable = self.data_tempMPU).place(x=150,y=390,width=80,height=20)
```

Figura 31. Programación de la Interfaz Gráfica en Python. Fuente: Autor

En la Figura 31 se muestra un fragmento del código desarrollado para crear para la interfaz gráfica.

Para demostrar que el sistema es capaz de hacer el track a más de un avión y es capaz de visualizarlo en una página web, dentro de la misma GUI implementada se creó una sección que le permite al usuario seleccionar hacía que bucket irán los datos; Los buckets se crearon en la plataforma de Initial State (Figura 32).

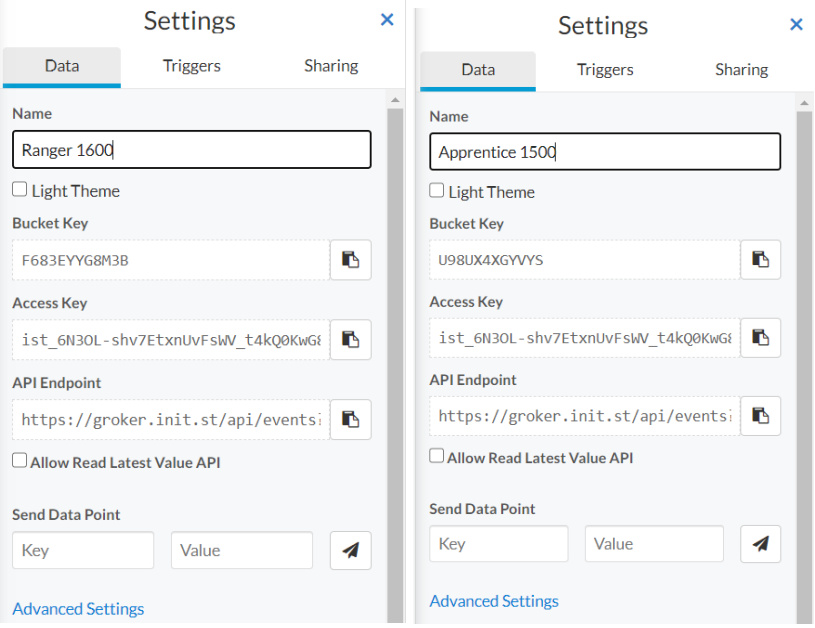


Figura 32. Configuración Bucket Initial State. Fuente: Autor

Los dos buckets implementados harán la recepción de los datos bien sea del aeromodelo Ranger 1600 (Figura 33) o del Aeromodelo Apprentice 1500 (Figura 34).

Figura 33. Aeromodelo Ranger 1600. Fuente: (Air-Rc, s.f.)

Figura 34. Aeromodelo Apprentice 1500. Fuente: (HorizonHobby, s.f.)

Como se mencionó al inicio del documento la implementación del dispositivo se hará en aeromodelos, ya que hacer la implementación en una aeronave real conlleva mayores costos.

a. Descripción Final del Prototipo

A continuación, se describe de manera general el sistema implementado mediante un diagrama de bloques (Figura 35).

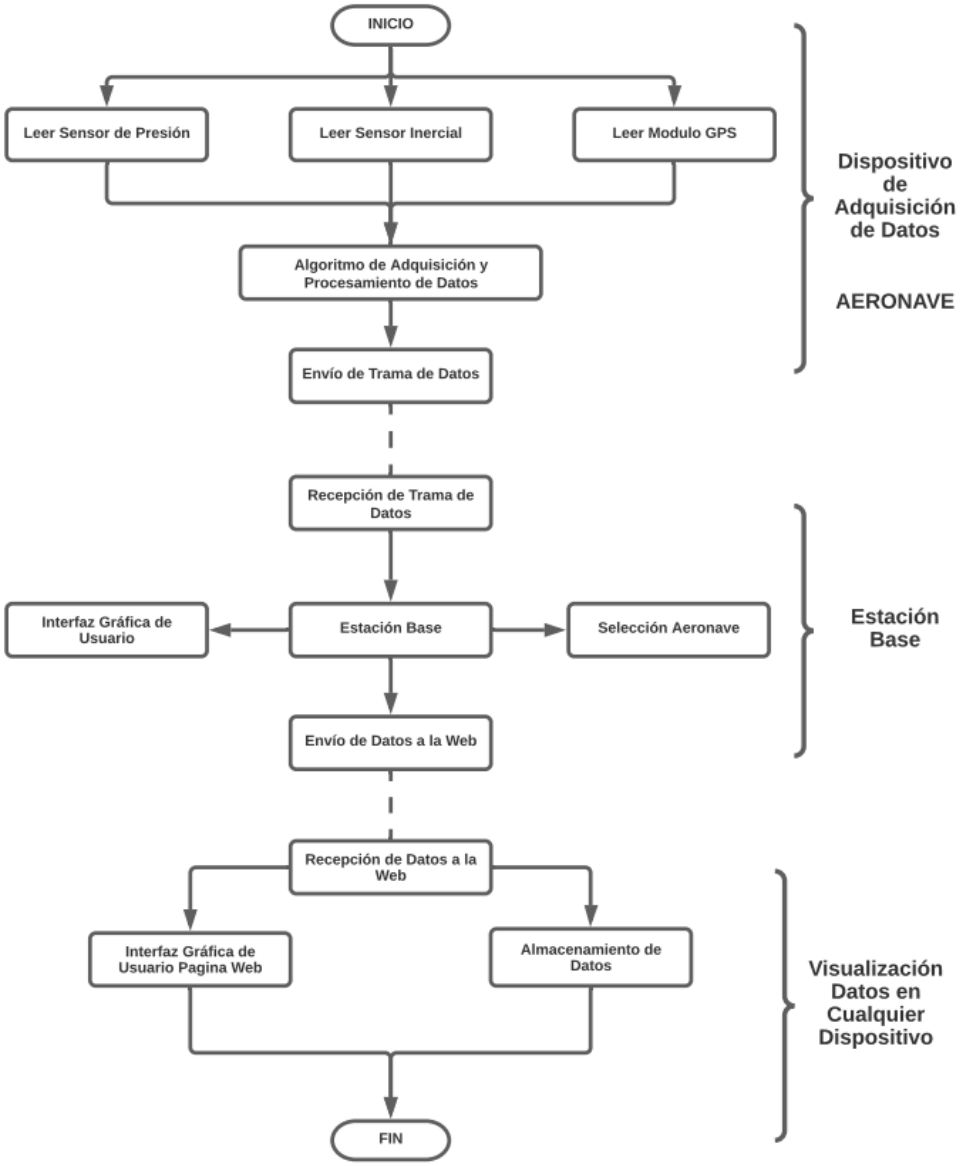


Figura 35. Diagrama General del Sistema. Fuente: Autor

Como se observó en la Figura 35 el sistema general se divide en tres grandes partes, la primera de ellas es donde se realizar la adquisición de los datos de vuelo, luego mediante la comunicación XBee se enviará a la estación base en donde se visualizarán los datos y desde allí se reenviarán a

la página web para que posteriormente se pueda acceder a los datos desde cualquier dispositivo en cualquier parte que cuente con una conexión a internet.

- **Descripción de la Adquisición de los Datos**

A continuación, se muestra el diagrama de bloques del programa implementado en Arduino (Figura 36), en el cual se realiza la adquisición de los datos de los sensores. Para fines prácticos este diagrama se dividirá en varios, empezando por una inicialización de variables y la configuración inicial del programa.

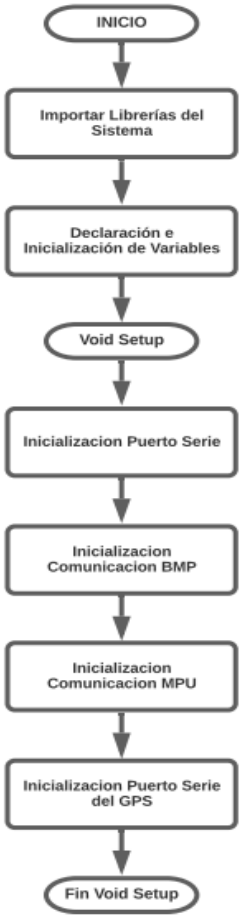


Figura 36. Diagrama de Flujo Inicio del Programa de Arduino. Fuente: Autor

El programa inicia con el llamado de las librerías necesarias para hacer la adquisición de los datos de los sensores correctamente, posterior a ello se inicializan y declaran las variables, luego el programa entra en la función del *void setup* que es donde se especificaran los comandos que se ejecutaran al inicio del sistema, en la Figura 36 se empieza iniciando el puerto serial del Arduino que es el que hará posible que funcione la comunicación serial para enviar los datos, luego se inicia la comunicación serial del GPS, ambas a 9600 baudios, también se inicia el sensor de presión y el sensor inercial; hasta aquí el programa cierra el *void setup* y no se volverá a acceder a este a menos que se reinicie la placa de desarrollo.

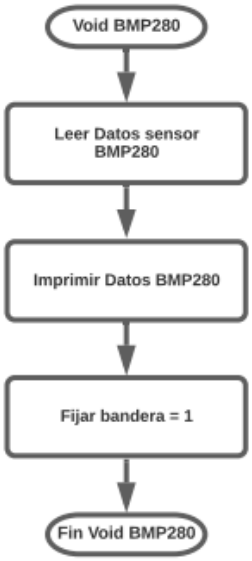


Figura 37. Diagrama de Flujo Función BMP280. Fuente: Autor

La figura anterior (Figura 37) muestra el diagrama de flujo de la función *BMP280* es decir el sensor de presión, en el cual se hace la lectura de los datos, posterior a ello se envían los datos por comunicación serial y se fija una *bandera* en un valor de 1, esto con el fin de que más adelante mediante un switch se le de paso a la siguiente función.

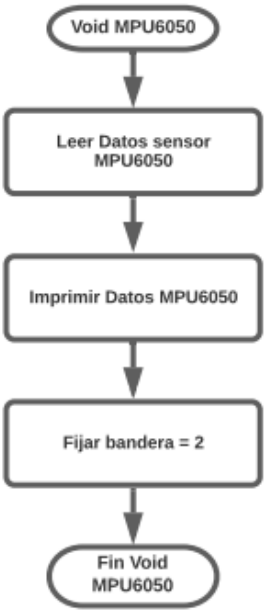


Figura 38. Diagrama de Flujo Función MPU6050. Fuente: Autor

El diagrama de flujo de la Figura 38 representa la adquisición de los datos del sensor inercial, funciona prácticamente igual que el sensor de presión, con la diferencia de que la *bandera* se fija en un valor = 2.

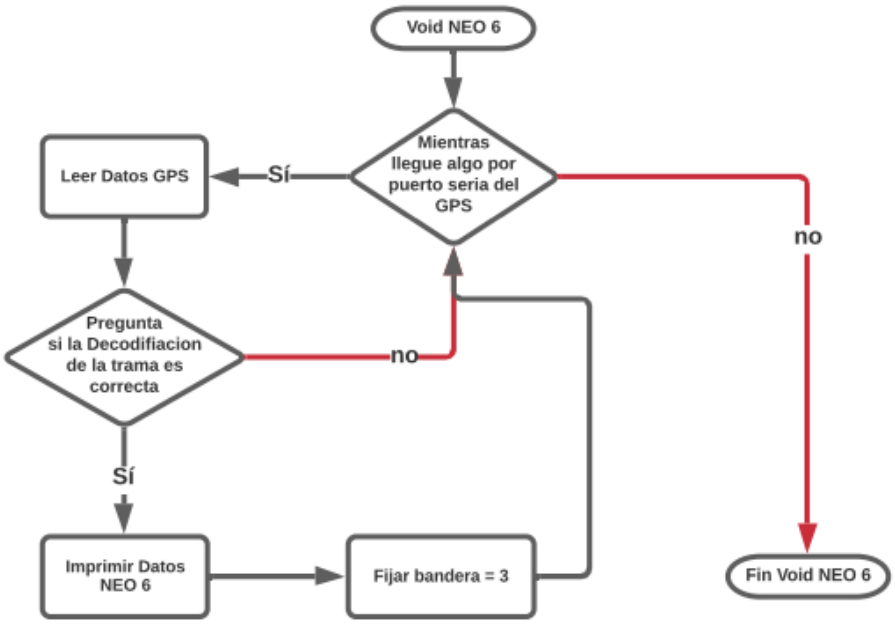


Figura 39. Diagrama de Flujo Función NEO 6m. Fuente: Autor

La función del *GPS NEO 6* (Figura 39) es diferente ya que en esta primero se pregunta si se está recibiendo un dato, de ser así el programa continua y vuelve a preguntar si la decodificación de la trama está en el formato correcto, ya que el módulo GPS envía tramas en diferentes decodificaciones, una vez la decodificación es correcta se imprimen los datos por serial y la *bandera* se fija en un valor = 3.

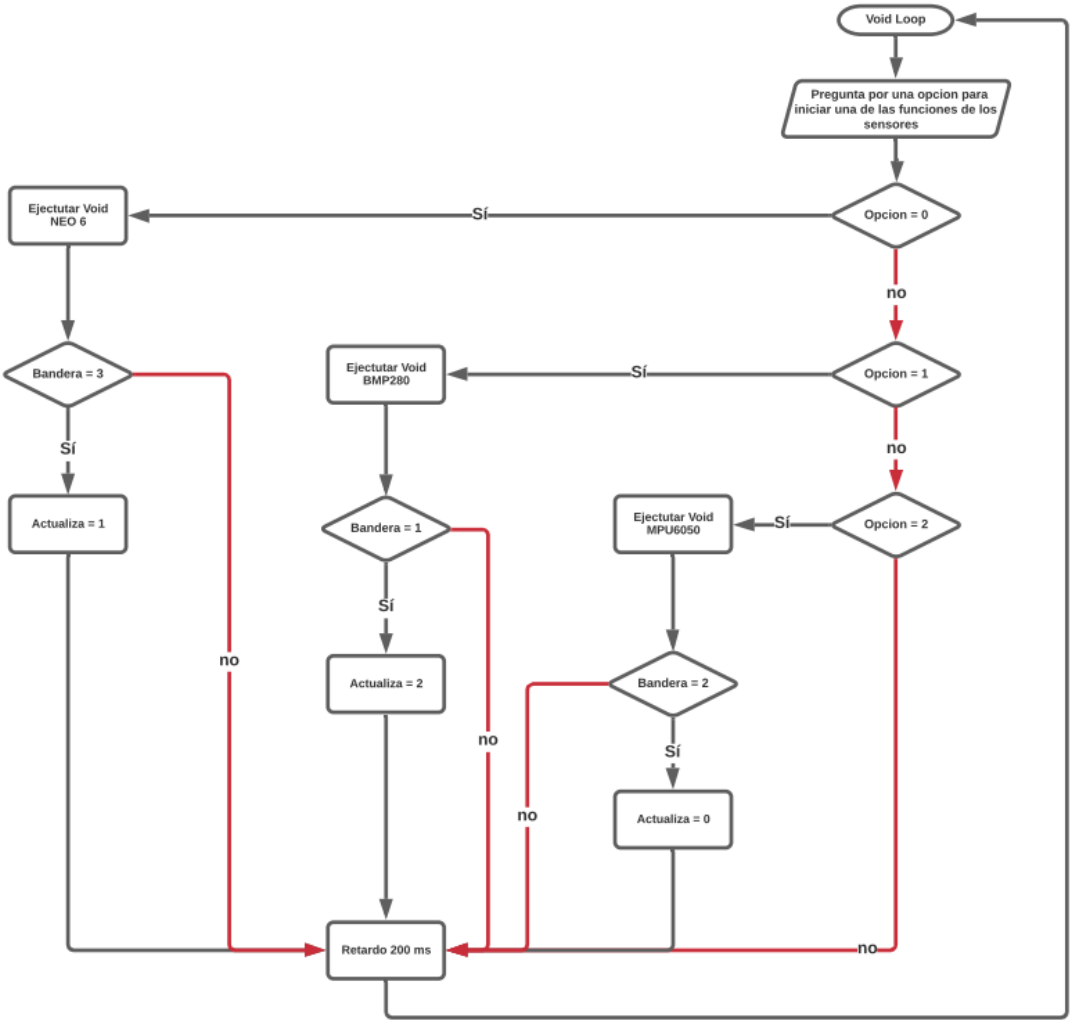


Figura 40. Diagrama de Flujo Función Void Loop. Fuente: Autor

En la Figura 40 se muestra el diagrama de Flujo de la función *Void Loop*, es decir la que se repite siempre, este empieza preguntando por el valor de la variable actualiza, dependiendo el valor empezara a ejecutar la función; para asegurarse de que siempre empiece por la función de GPS, la variable actualiza se inicializo al principio con el valor de 0, una vez entra al case se ejecutara la función GPS, posterior a ello se preguntara por el valor de la *bandera* y si es igual a 3 se actualizara el valor de la función actualiza para que entre al case siguiente. El valor de la bandera es para asegurarse de que el programa termine de adquirir e imprimir todos los datos, para los otros dos case funciona igual.

- **Descripción Estación Base**

El programa de la estación base se encarga de recibir los datos que llegan por comunicación inalámbrica, mostrarlos y luego enviarlos a la página web para que estos puedan ser visualizados en tiempo real desde cualquier dispositivo.

La Figura 41 describe el sistema general del programa de la estación base.

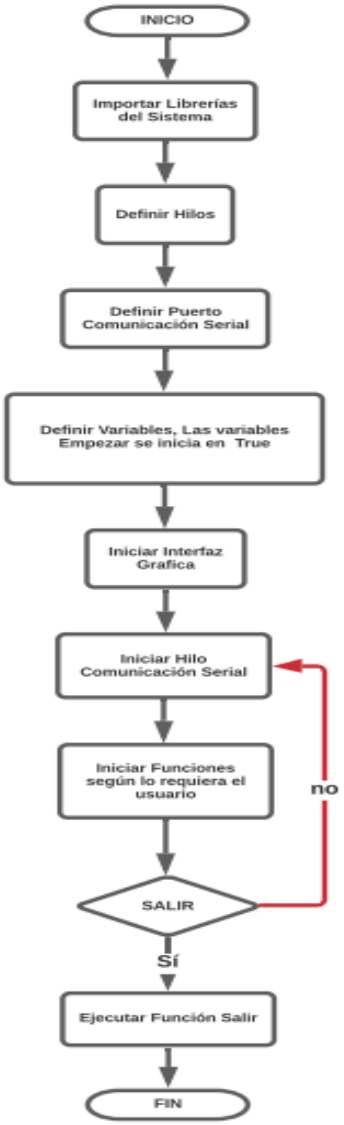


Figura 41. Diagrama de Flujo General Programa Estación Base. Fuente: Autor

Como se observa en la figura anterior el sistema inicialmente importa todas las librerías necesarias para su funcionamiento, luego se definen los hilos de comunicación serial y envió de datos a la web, esto para tener un control más preciso del momento en que se hará dicha adquisición y envió, después se define un puerto de comunicación serial por donde se estarán recibiendo los datos en la raspberry, para este caso si se usara un sistema operativo Windows se definiría como **COM#**, pero como se usa una distribución de Linux basada en debian se usa **/dev/ttyUSB0** se definen las variables que se utilizaran en el sistema y la variable (*empezar*) que se encargara de ejecutar la función de comunicación serial en todo momento dentro de un while, se fija en un valor verdadero para que siempre se esté realizando mientras el hilo de la comunicación serial se halla iniciado, una vez fijadas las variables se inicia la interfaz gráfica y el hilo de comunicación serial, luego se ejecutan las funciones que el usuario vaya necesitando, estas funciones se muestran a continuación y por último, si se cierra la ventana se ejecuta la función salir y el programa finaliza.

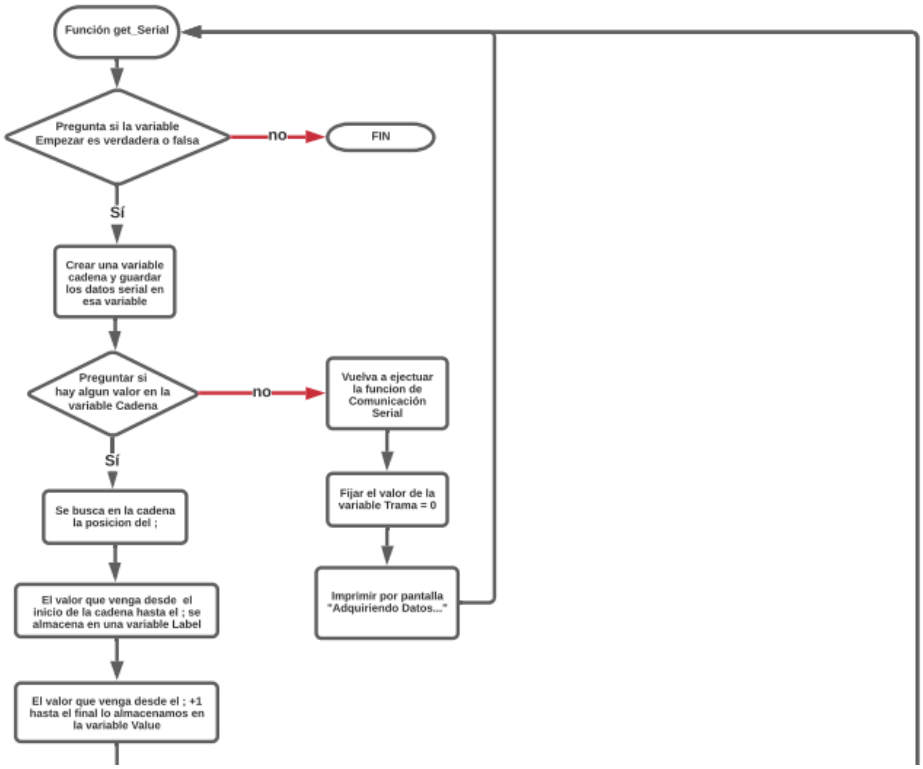


Figura 42. Diagrama de Flujo Función get_Serial Parte I. Fuente: Autor

El diagrama de flujo de la *función get_Serial* se dividió en 3 figuras, el diagrama completo se puede observar en el Anexo 1, en la Figura 42 se muestra la parte I de dicho diagrama de flujo, en esta primera parte se empieza preguntando por el valor de la variable (*empezar*) definido previamente como verdadero al inicializar las variables (Figura 41), una vez se comprueba el valor si es verdadero el programa crea una variable llamada (*cadena*) que es donde se guardaran los datos que llegan por comunicación serial, después se pregunta si dicha variable trae algún valor, en caso de no traerlo el programa vuelve a ejecutar la *función get_Serial* hasta que contenga un valor, fija la variable Trama en 0 e imprime por pantalla "Adquiriendo datos", si la variable trae algún dato empieza a buscar en la trama de datos un punto y coma (;), el valor que venga desde el inicio de la cadena hasta la posición del punto y coma se almacenara en una variable llamada Label, y el valor que venga desde el punto y coma + 1 se almacenara en una variable Value, ejemplo llegan los datos A;34B;22;… el Label será igual a (A o B) y el Value será igual a (34 o 22).

Una vez fijado lo anterior se crea una sentencia IF donde se pregunta por el valor que venga en la variable Label (Figura 43), esto se hace para asegurar que el valor que llega después de esa letra corresponde a la variable que se envió desde el Arduino, en los códigos explicados de este se mostró la forma en la que se enviaban los datos por comunicación serial (Figura 24), una vez consultado el valor de Label guardara el valor que venga después de este en la variable asignada y lo convertirá a dato flotante entrando por un try y añadiendo una excepción en caso de que venga una letra junto al dato puesto que el dato llega tipo char y más adelante algunos datos necesitaran ser operados, este procedimiento se repite hasta la cuando el Label es igual a S, en caso de que el

Label sea igual a C o D no se realizara ninguna conversión a dato flotante ya que estos datos son la Hora y la Fecha y no requieren ser operados.

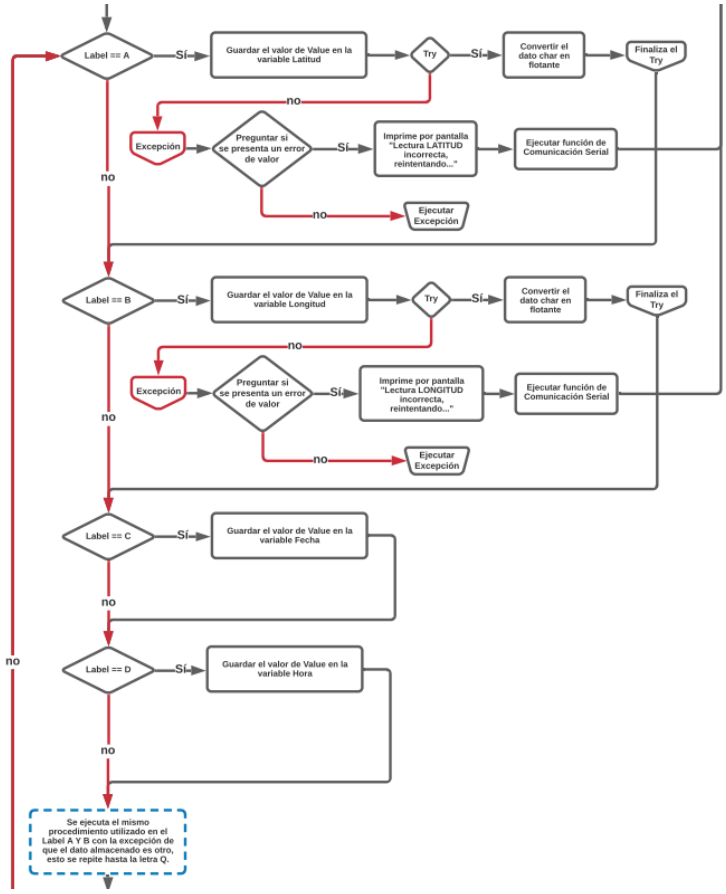


Figura 43. Diagrama de Flujo Función get_Serial Parte II. Fuente: Autor

En la Figura 44 se muestra la última parte de diagrama del flujo la cual lo que hace es que cuando el Label es igual a S se imprimirá por pantalla el mensaje que viene junto a la trama de datos el cual es “Trama completa”, se fijara el valor de la variable *Trama* en 1 (El Por qué se explica más adelante, página 82), operara los datos de altitud del gps y del bmp (sensor de presión) para sacar un promedio y hace lo mismo para el caso de la temperatura entre el bmp y el mpu (sensor inercial), también se pregunta por el valor de la velocidad y en caso de ser mayor a 1 se fija la variable estado en “En Vuelo” esto es un para saber si el avión se encuentra volando o no,

la consulta se puede ajustar haciendo uso de la altitud, posición gps y demás pero para este caso no es necesario ya que al probarse el sistema en un aeromodelo se sabe que automáticamente cuando este gana velocidad ya se encuentra próximo a volar, en caso de que la velocidad sea inferior la variable estado cambiara a “En Tierra” y en ambos casos al finalizar el programa se quedara ciclando hasta cuando la variable *empezar* cambie de True a False.

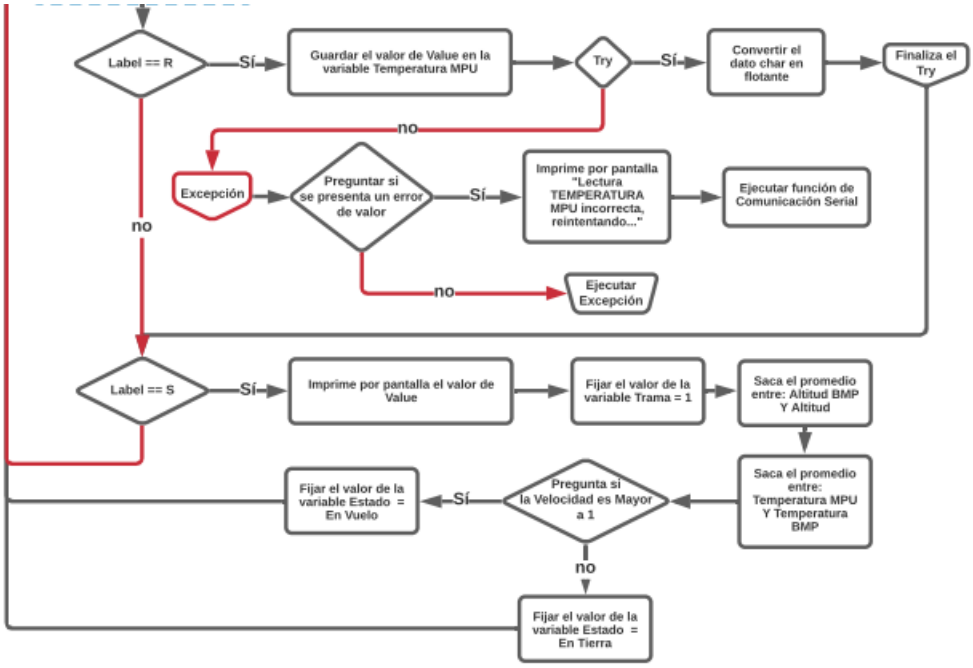


Figura 44. Diagrama de Flujo Función get_Serial Parte III. Fuente: Autor

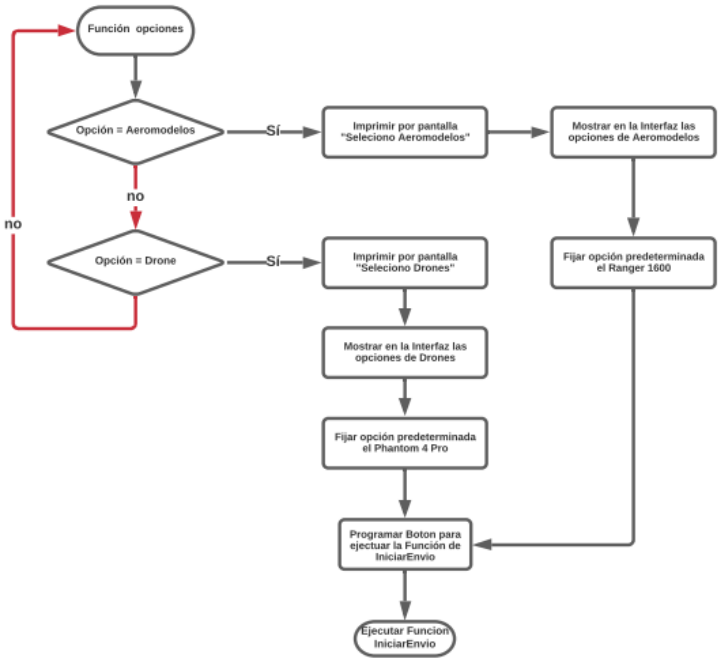


Figura 45. Diagrama de Flujo Función opciones. Fuente: Autor

La figura anterior muestra el diagrama de flujo del a *función opciones* la cual permite a el sistema escoger que tipo de aeronave se volara, para este caso si es un Drone o un Aeromodelo, dependiendo lo que se seleccione el sistema imprimirá por pantalla la selección y programara un botón con el label “consultar” para que ejecute la función de *iniciarEnvio* (Figura 46).

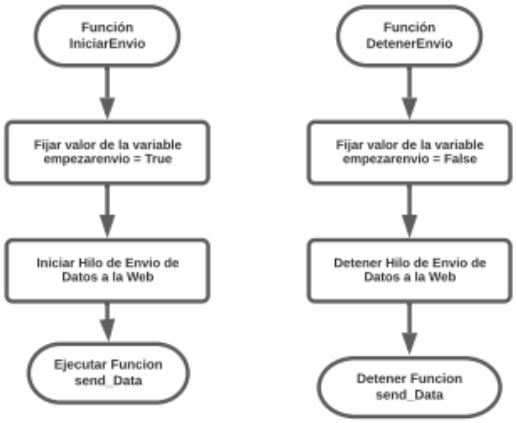


Figura 46. Diagrama de Flujo Funciones Iniciar Envio y Detener Envio. Fuente: Autor

La Figura anterior (Figura 46) muestra el diagrama de flujo de las *funciones iniciar envio* (Izquierda) y *detener envio* (Derecha); La función de la izquierda fija la variable *empezarenvio* en un valor True, para que cuando se llame en el while de la *función send_Data* este se ejecute en todo momento mientras sea True, después se inicia el hilo de envió de datos a la web para que al final se ejecute la *función send_Data*.

Para el caso de la *función detener envío* se hace lo opuesto a lo ya mencionado anteriormente.

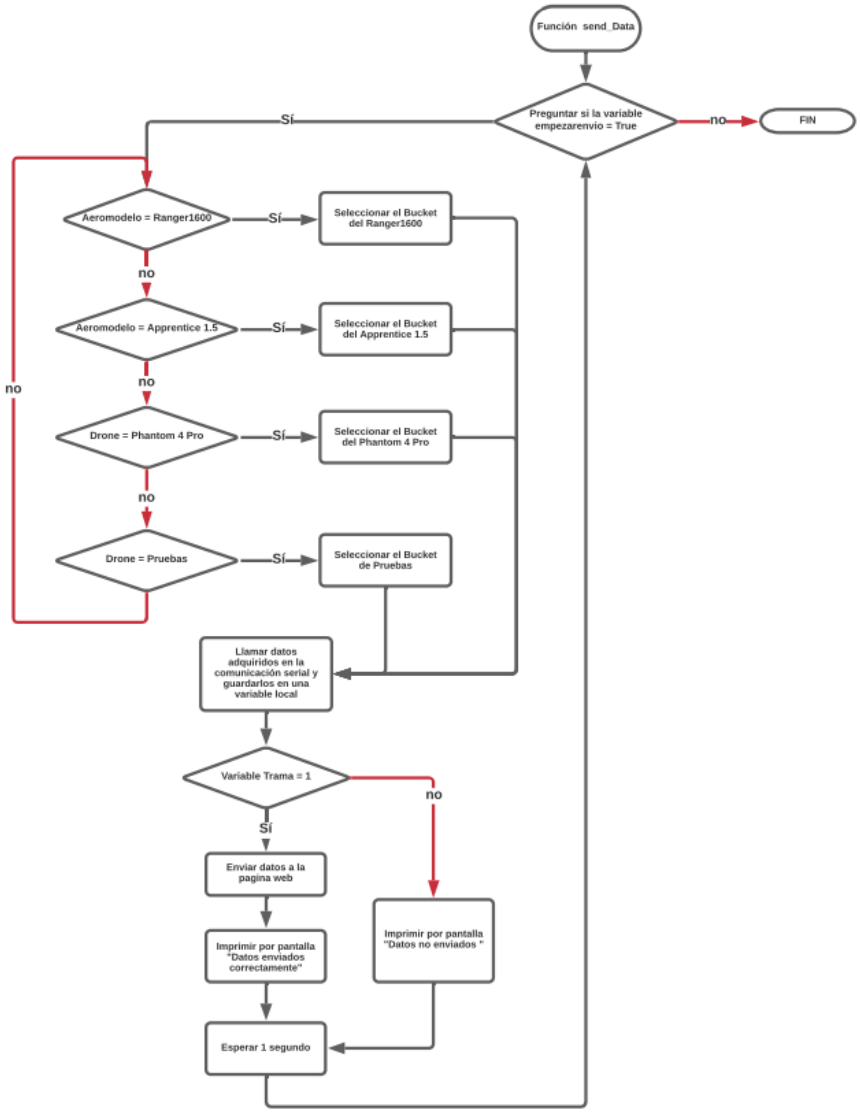


Figura 47. Diagrama de Flujo Funciones send_Data. Fuente: Autor

Para empezar a realizar el envío de datos a la página web primero hay que seleccionar a que bucket irán dichos datos, esto se hace en la *función de send_Data*, su diagrama de flujo se muestra en la Figura 47, el programa inicia preguntando por el valor de la variable *empezarenvio*, esta es una variable que se creó anteriormente y se configuro en la *función iniciarenvío* (Figura 46) para controlar el ciclo while de esta función, una vez se comprueba que el valor es verdadero entra en el ciclo while y según lo que se halla seleccionado en la función de opciones (Figura 45) se mostrara lo siguiente en caso de que se seleccione un Aeromodelo se mostraran 2 opciones ranger 1600 y apprentice 1500 (Figuras 33 y 34 respectivamente), para el caso de la opción Drone aparecerán Phantom 4 pro y una opción de pruebas para no ocupar las otras opciones, dependiendo el aeromodelo o drone seleccionado los datos irán al bucket correspondiente, luego se hace un llamado a los datos adquiridos en la comunicación serial y se almacenan en una variable local ya que si se llama la variable original de tipo self(es decir alcance de clase) la librería streamer no la deja utilizar, después de esto se pregunta por el valor de la variable *Trama* esto es para asegurar que cuando la trama sea igual a 1 los datos que se envíen a la web estarán completos y a su vez se imprime por pantalla que los datos fueron enviados correctamente, en caso de que la variable *Trama* sea 0 simplemente se imprime por pantalla que los datos no se enviaron y se espera un segundo para volver a intentar.

El tiempo de envío de los datos varia, es decir los datos no llegan cada 5 segundos o cada tiempo fijo, si no que van a estar variando ya que su envió depende de si la trama llego completa o no, cabe aclarar que este tiempo no excede los 10 segundos, puesto que el mismo sistema pregunta rápidamente de nuevo por la trama y en las pruebas jamás se presentaron problemas con estos tiempos (ver capítulo de resultados).

Figura 48. Diagrama de Flujo Funciones Salir. Fuente: Autor

Al final cuando se quiere terminar el programa al cerrar la ventana el programa automáticamente ejecuta la función de salir, en esta se asegura que tanto los hilos como la comunicación serial se cierren por completo y se imprime por pantalla que el programa finalizo, esto se visualiza en el IDLE de Python.

- **Diseño e Implementación de la PCB**

  Una vez se tiene todo el sistema previamente probado en una protoboard y funcionando como se explicó en la sección anterior, en base a el diseño del circuito eléctrico (Esquemático Figura 15) se procede a realizar el diseño de la PCB que va dentro de la aeronave.

En la figura de continuación se muestra el diseño final de dicha PCB, el cual utiliza las dos capas de la baquela, esto con el fin de evitar un cruce de caminos entre el pin de alimentación del sensor de presión y el pin de alimentación del módulo XBee, adicional a ello se colocó un logo, el nombre del proyecto y el autor.

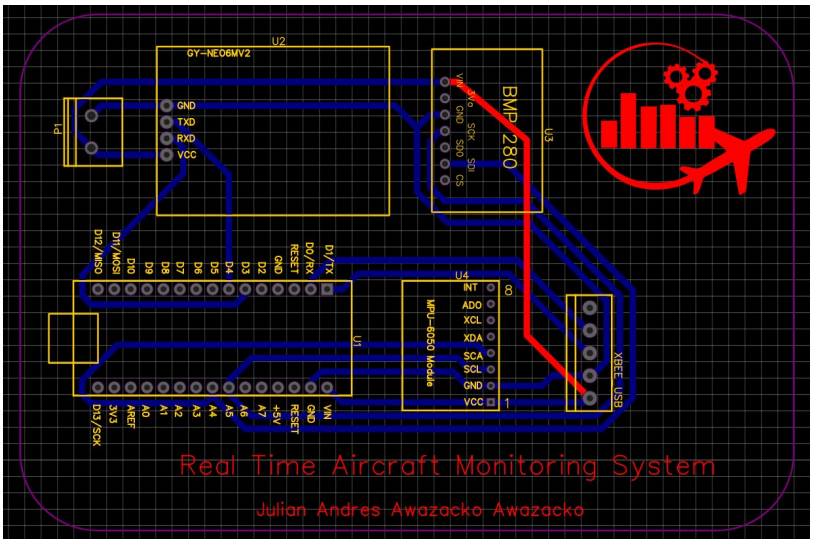


Figura 49. Diseño PCB del Sistema. Fuente: Autor

Los caminos de color azul corresponden a la capa inferior de la PCB y los que están en rojo son de la capa superior, el resultado esperado de esta PCB se muestra en la Figura 50.

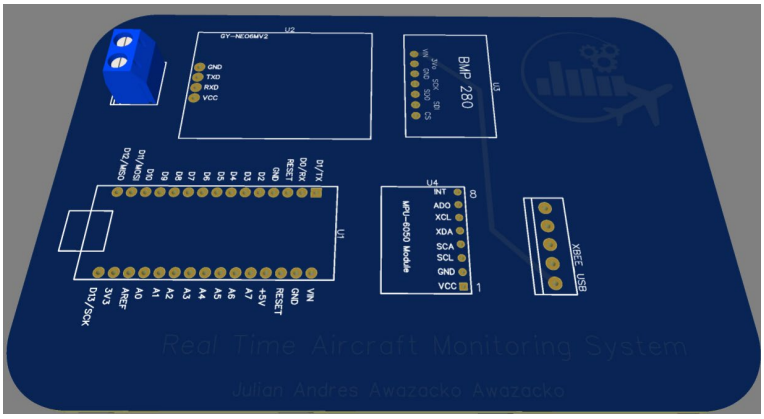


Figura 50. PCB del Sistema en 3D. Fuente: Autor

5. CAPITULO 5: RESULTADOS

a. Implementación Final

Para fines prácticos el desarrollo de la PCB se dejó a cargo de una empresa encargada de imprimir este tipo de circuitos, el resultado que se obtuvo fue el siguiente (Figura 51).

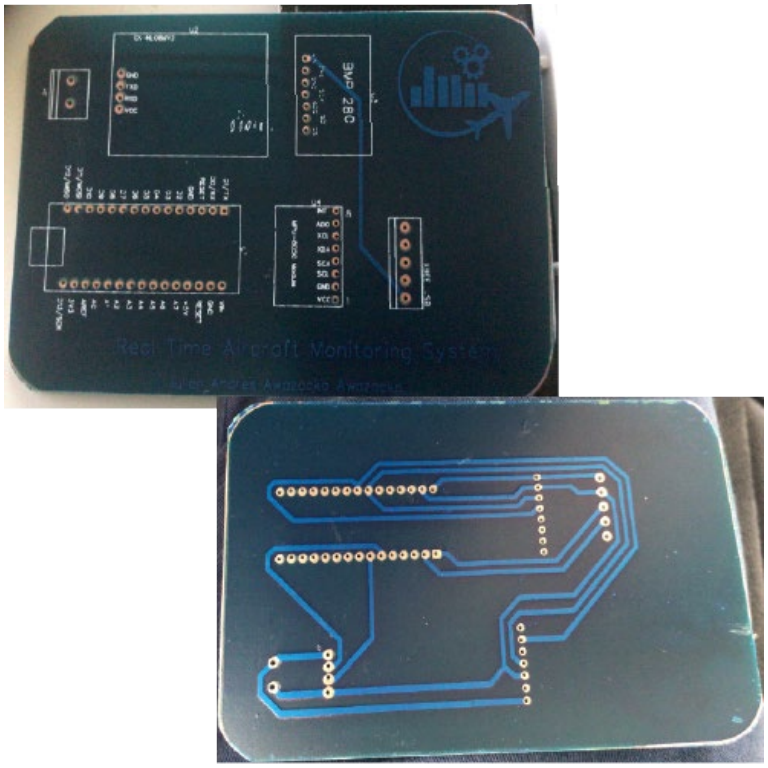


Figura 51. PCB impresa del Sistema. Fuente: Autor

Una vez se cuenta con la PCB, se agregaron los componentes a esta, utilizando regletas de pines (Figura 52) para no soldar los componentes directamente en la PBC y en caso de un fallo en el sistema o de algún componente, solo se haga necesario cambiarlo sin tener que desoldar.

Figura 52. Regleta de Pines. Fuente: Autor

El resultado del sistema puesto ya en la aeronave se puede observar en la Figura 53 y 54, adicional a este sistema se le agrego un fusible de 500mA para que en caso de que exista un choque o una mala conexión en el aeromodelo y se produzca un corto este no queme los componentes.

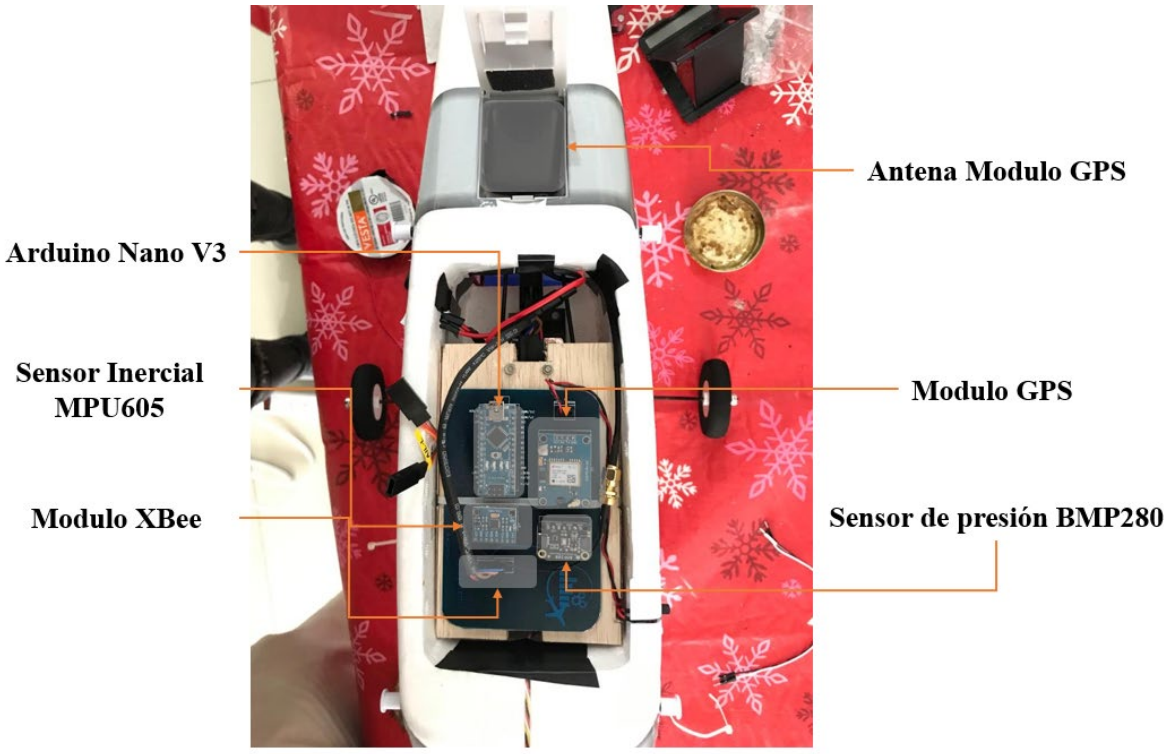


Figura 53. Sistema acomodado dentro del Aeromodelo Parte de Arriba. Fuente: Autor

En la Figura 53 se observa la parte superior de la PCB con los sensores y la placa de desarrollo ya acomodados en el sistema, ahora en la Figura 54 se observará la parte de debajo del sistema que es donde se ubicó el regulador de voltaje (Figura 55) para alimentar todo a 5v y el módulo Xbee ya que en esta posición se logra acomodar de mejor manera la antena mostrada en la Figura 11.

Figura 54. Sistema acomodado dentro del Aeromodelo Parte de Abajo. Fuente: Autor

El sistema esta alimentado con una batería de 6.6V de 2100 mAh (Figura 56).

Figura 55. Modulo LM2596 Regulador de Voltaje. Fuente: (Ferretronica, s.f.)

Figura 56. Batería para alimentar el Sistema. Fuente: Autor

Se selección una batería de 6.6v de 2100mAh generando un amplio margen de corriente para no tener problemas al momento de que la aeronave se encuentre lejos, el consumo máximo que se pudo observar del sistema fue de 187mAh cuando el módulo Xbee se encontraba más lejos. En la selección de la batería también tuvo que ver mucho su forma y tamaño, puesto que era necesario que esta fuera delgada para caber en el fuselaje del aeromodelo.

Figura 57. Parte inferior y lateral del Aeromodelo. Fuente: Autor

En la Figura 57 se observa la parte inferior del aeromodelo con la ubicación de la batería que alimentara el sistema de adquisición de datos y la batería que alimentara todo el sistema del aeromodelo, también se observa la ubicación de la antena y los cables de la antena gps que se le coloco al sistema aquí se tuvo especial cuidado para que no interfirieran con los servomotores de control de las superficies de la aeronave. A la parte derecha de la Figura 57 se ve el sistema lateralmente y se observa un switch que se utiliza para el encendido del sistema de adquisición de datos RTAMS.

b. Resultados Finales y Análisis de datos

Teniendo en cuenta la normatividad aeronáutica colombiana se hace necesario el uso de una pista de aeromodelismo para realizar las pruebas del sistema.

RAC 91 reglas generales de vuelo y operación, Apéndice 13 (Operación de sistemas de aeronaves no tripuladas UAS) Circular 328 AN/190 OACI desde 05 de agosto de 2019 (Ver Marco Legal).

El documento también habla de las limitaciones de las aeronaves que se encuentren en esta categoría, dentro de las cuales se destaca:

- Peso no mayor a 25 Kg
- Velocidad Máxima de 80Km/h
- La aeronave debe tener un alcance de línea de vista con un radio máximo de operación de 500 metros horizontales durante todas las fases del vuelo.
- La operación no puede ser efectuada sobre público, reuniones de personas al aire libre, aglomeraciones de edificios, ciudades u otras aéreas pobladas.
- Operación debe ser en horas diurnas bajo reglas VFR y condiciones VMC
- Altura de vuelo no superior a 123metros AGL (Respecto al suelo)
- El vuelo debe realizarse dentro de espacio aéreo clase G es decir no controlado.

Una vez se cuenta con el conocimiento de dichas limitaciones se realizó la selección del lugar donde se volaría la aeronave, como se pudo observar volar la aeronave cerca de la universidad no es posible ya que la ley lo impide y es un riesgo para la comunidad.

A continuación, se presentan los resultados obtenidos del sistema en las pruebas realizadas el día 30 de marzo de 2021 en la pista de aeromodelismo delta 02 ubicada en la ciudad de Chía Cundinamarca.

Antes de iniciar el vuelo y las pruebas es necesario revisar y ajustar el peso y balance de la aeronave y verificar que el sistema se encuentre funcionando (Figura 58).

Figura 58. Aeromodelo Apprentice 1500. Fuente: Autor

En la Figura 59 se observa el aeromodelo listo a despegar y a partir de este momento el sistema empieza a hacer el track de vuelo de la aeronave, a continuación, se anexan algunas imágenes del funcionamiento del sistema en tiempo real.

Figura 59. Aeromodelo Apprentice 1500 próximo a Despegar. Fuente: Autor

Figura 60. Estación Base. Fuente: Autor

Para la alimentación de la raspberry pi (Estación base) es necesario ubicarse cerca de un tomacorriente, para este caso se ubica cerca de un vehículo como se observa en la figura anterior (Figura 60).

En la Figura 61 se aprecia como la estación base inicia la adquisición de datos de vuelo, para este caso hay que tener presente que la antena receptora del XBee no está 100% en línea de vista, pues existen obstáculos como el techo del vehículo y del parqueadero donde este se encontraba. A pesar de ello el sistema realizo toda la adquisición correctamente.

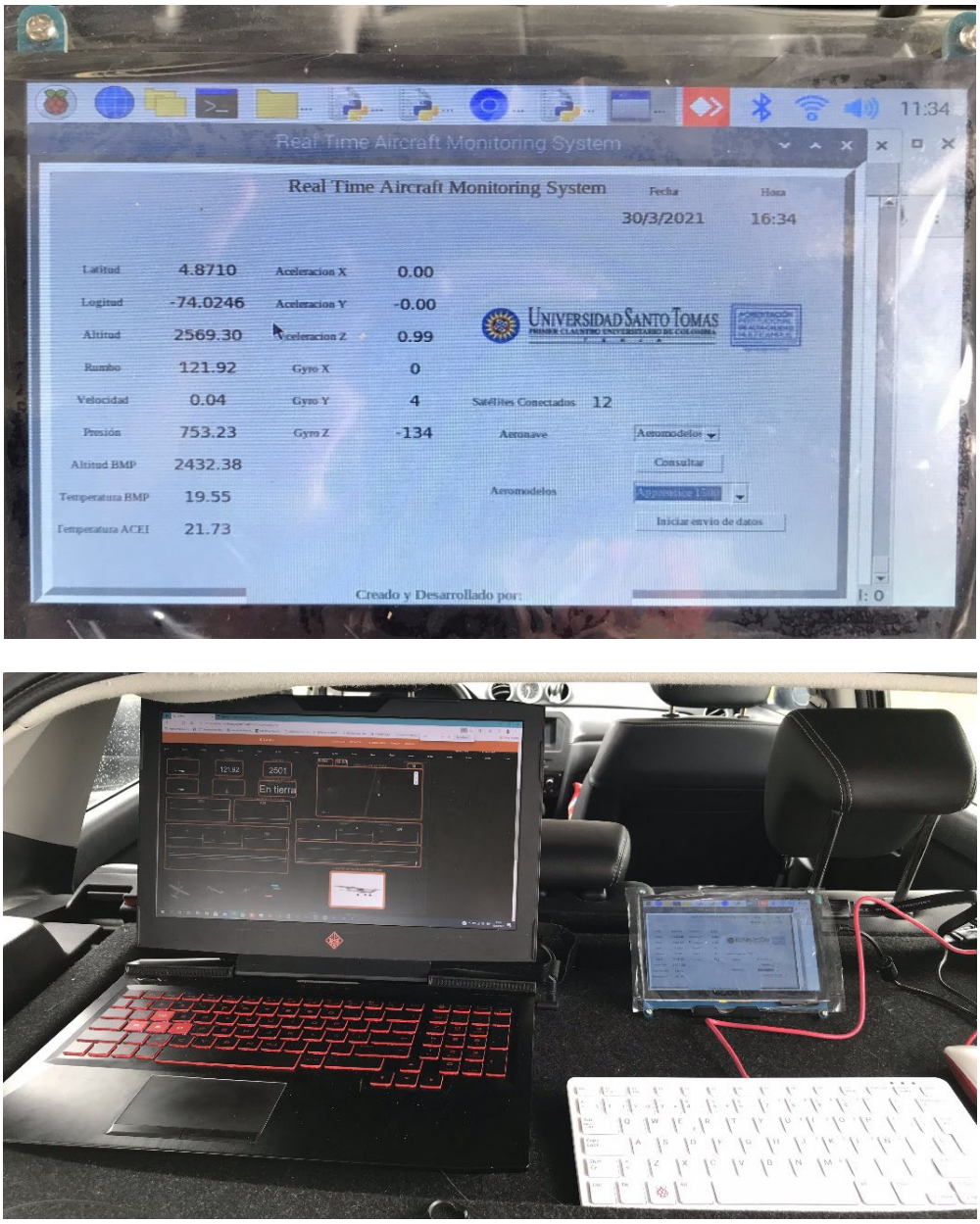


Figura 61. Interfaz Gráfica Estación Base y portátil para visualización de datos web. Fuente: Autor

Ahora una vez que los datos estaban llegando a la estación base se le indica al sistema que empiece a enviar los datos a la página web ***https://www.monitoringsystem.online/es/apprentice.php***, esto con la intención de que puedan ser visualizados en cualquier dispositivo con una conexión a internet, para fines prácticos al lado de

la estación base se acomodó un portátil con conectividad a internet para poder comparar los datos que se encontraban en la estación y en la web.

En la Figura 62 se muestran los datos obtenidos del sistema al momento de que el aeromodelo se encontraba listo a despegar, aproximadamente a las 11:35 am (16:35 zulú), en la parte izquierda de la figura se observa el recorrido que hace la aeronave desde el hangar hasta la pista, también se comprueba que el estado actual de la misma y tiene una orientación de 300 grados, la altitud a la que la aeronave despego fue de 2503 metros.

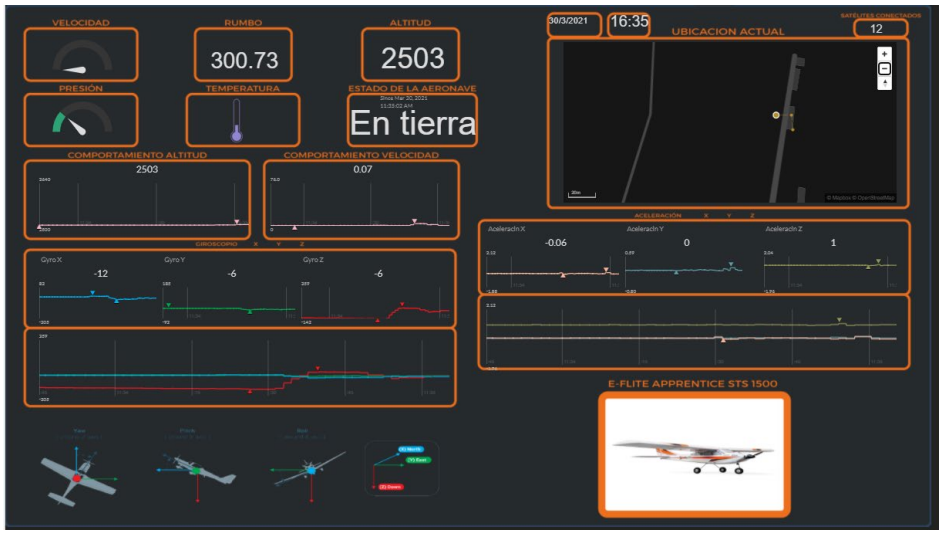


Figura 62. Visualización datos fase de despegue. Fuente: Autor

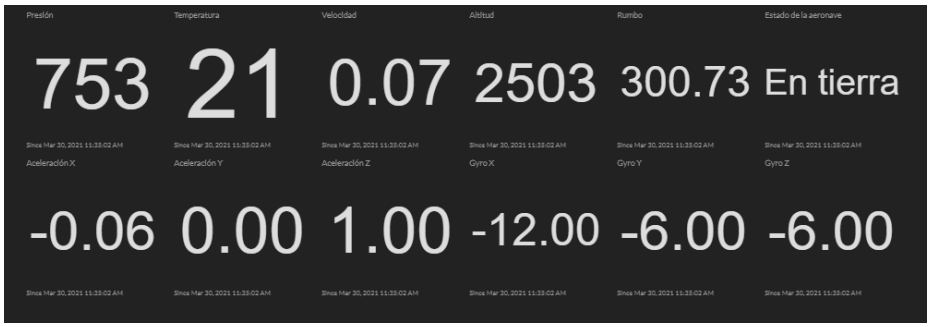


Figura 63. Datos fase de despegue. Fuente: Autor

En la sección de temperatura y presión se observa que no se puede ver el valor actual, para ello deslizando hacia abajo un poco en la página web se pueden observar los datos (Figura 63), aquí se aprecia que la temperatura era de 21 grados y la presión de 753 hPa (Hectopascales), estos datos se comparan con una plataforma de meteorología llamada meteocast (**https://es.meteocast.net/**)para saber más o menos que tanto varían y se observa que la presión y temperatura se encuentran dentro de los rangos esperados, pues cabe aclarar que la página de meteorología hace previsiones y no da un valor exacto, así que se puede decir que los datos más exactos son los arrojados por el sistema implementado en el aeromodelo.

Figura 64. Datos meteorológicos de chía. Fuente: (meteocast.net, s.f.)

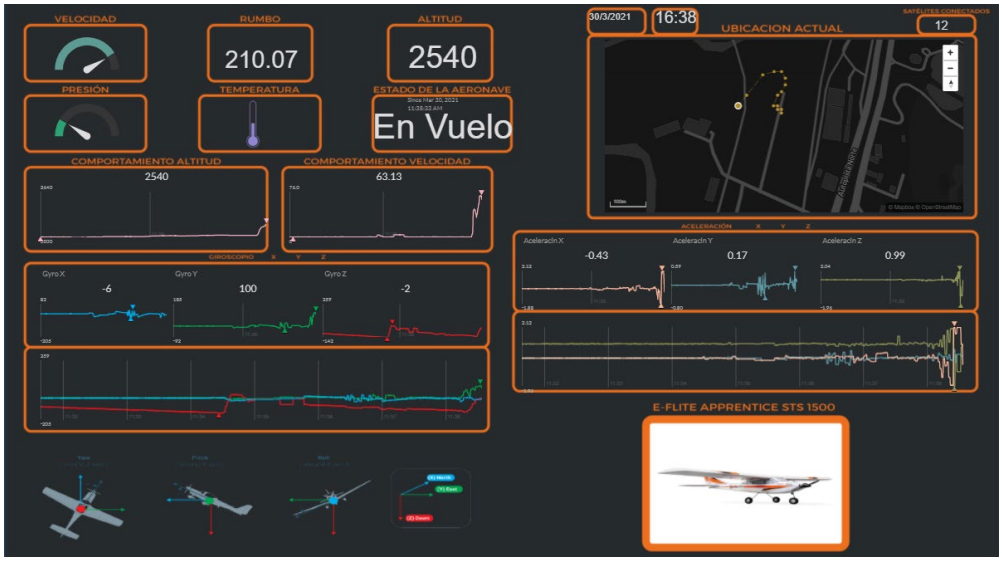


Figura 65. Visualización datos fase de ascenso. Fuente: Autor

En la Figura 65 se observa la fase de ascenso del aeromodelo a una velocidad de 63 Km/h, aquí se puede verificar que los datos del sensor inercial están adquiriéndose correctamente, puesto que se observa que el Giro en el eje Y conocido como Pitch o Cabeceo se encuentran en 100 grados, esto indica un ascenso pronunciado del aeromodelo.

En el documento de soporte técnico de la librería MPU_LIGHT (Fetick, 2021) el autor explica que los datos que se encuentran en color rojo y verde (Figura 66) pertenecen a la velocidad angular y la aceleración lineal respectivamente para los 3 ejes. Estos datos se obtienen directamente del MPU 6050 (Figura 7) y la librería lo que hace es operar los datos como se muestra en la Figura 66, para obtener los valores de los ángulos a partir de la aceleración y el valor de los ángulos desde la fusión de la aceleración con los datos del giroscopio. El autor comenta que los datos calculados a partir de la aceleración son ruidosos y los ángulos calculados a partir de la fusión del acelerómetro y el giroscopio son más precisos.

El valor obtenido finalmente de los ángulos depende del cálculo matemático de la integral de la velocidad angular respecto al tiempo como se muestra en la Figura 66. Este cálculo en nuestro caso, debido a la tasa de actualización del sensor que corresponde a 0.9 microsegundos combinado con el tiempo de ejecución de la estructura del algoritmo secuencial que corresponde tanto a la librería utilizada MPU_LIGHT (Fetick, 2021), como al algoritmo desarrollado en este trabajo para la adquisición de los datos de todos los sensores en el Arduino. Genera un error en el cálculo de los ángulos debido al tiempo de integración entre una medida y la siguiente. El error es más notable cuando la aeronave realiza movimientos bruscos o rápidos y se obtienen mejores resultados cuando los movimientos son lentos o suaves. Se debe considerar también que una vez se ha hecho el cálculo inicial de la integral, el algoritmo ejecuta un filtro en busca de reducir el ruido sobre el

valor obtenido. Para dar solución a este problema se recomienda la adquisición de un sensor INS, que tenga un tiempo bajo de actualización y que suministre directamente el valor de los ángulos.

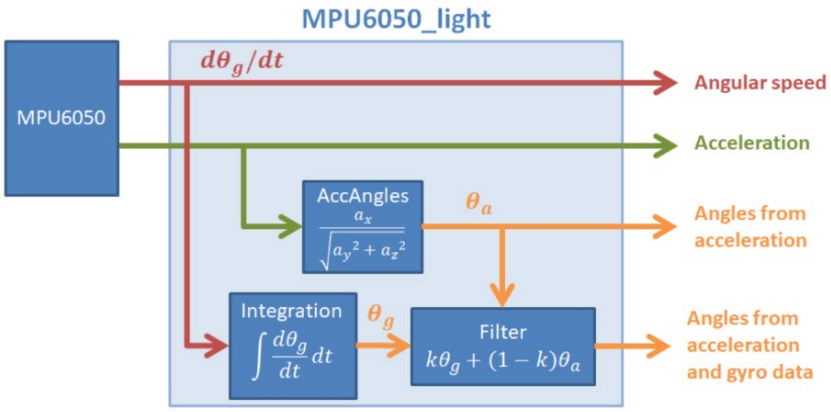


Figura 66. Visión general de los datos accesibles a través de la librería MPU6050_light. Fuente: *(Fetick, 2021)*

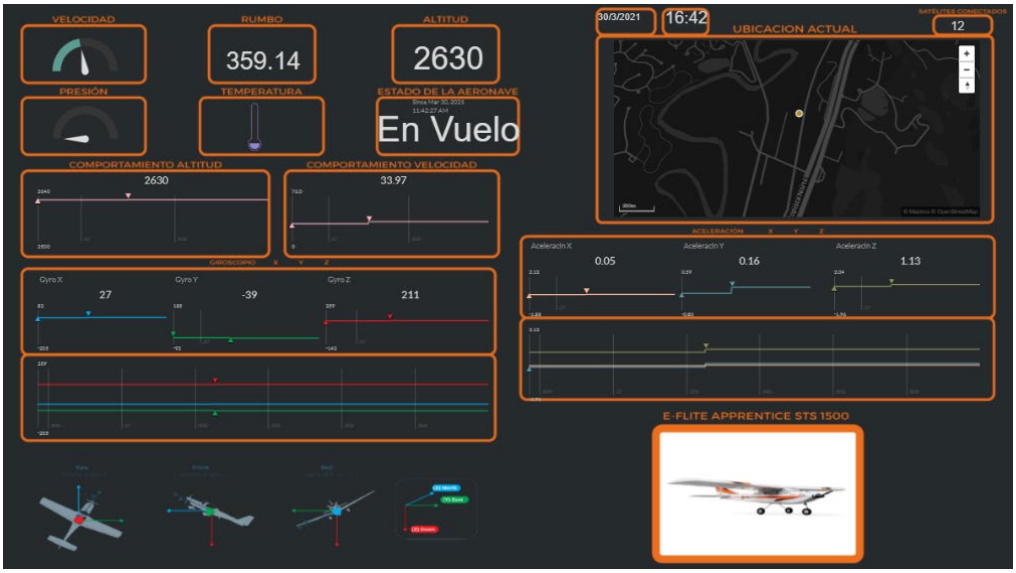


Figura 67. Visualización datos fase de crucero. Fuente: Autor

La fase de crucero es cuando la aeronave alcanza una altitud de vuelo constante, en el caso de los aeromodelos es un poco difícil mantenerla ya que existen variables como el viento que puede elevar o bajar bruscamente la aeronave debido a que tiene un peso liviano, para este caso en la

Figura 67 se muestran los datos que se obtuvieron al momento de alcanzar la altitud máxima la cual fue de 2630 metros, es decir la aeronave alcanzo 127 metros AGL (respecto al suelo) 4 metros por encima de lo permitido, cabe aclarar que esto no quiere decir que se haya entrado en un vuelo ilegal, ya que muchas veces para el piloto del aeromodelo es complejo calcular la altura de la aeronave y en estos casos entre más alto mejor puesto que en caso de un error se tiene más margen para controlar el aeromodelo. También se puede apreciar que la velocidad en ese momento era de 33 Km/h la mitad de la que se obtuvo durante el ascenso y realizando un giro de 27 grados (Gyro X).

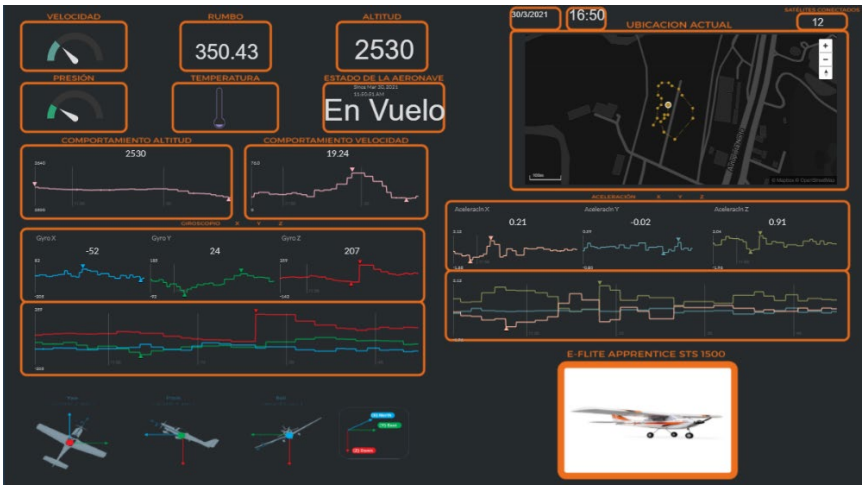


Figura 68. Visualización datos fase de descenso. Fuente: Autor

En la figura anterior se observan los datos al momento del descenso, aquí se destaca que las aeronaves descienden con la nariz hacia arriba y no como se pensaría normalmente con la nariz hacia abajo es por eso que el Gyro en Y es de un valor positivo (24 grados), también se puede apreciar una reducción de la velocidad y la altitud, los datos del Gyro X indican que la aeronave está haciendo un alabeo, bien sea a la izquierda o la derecha y el Gyro en Z indica la guiñada.

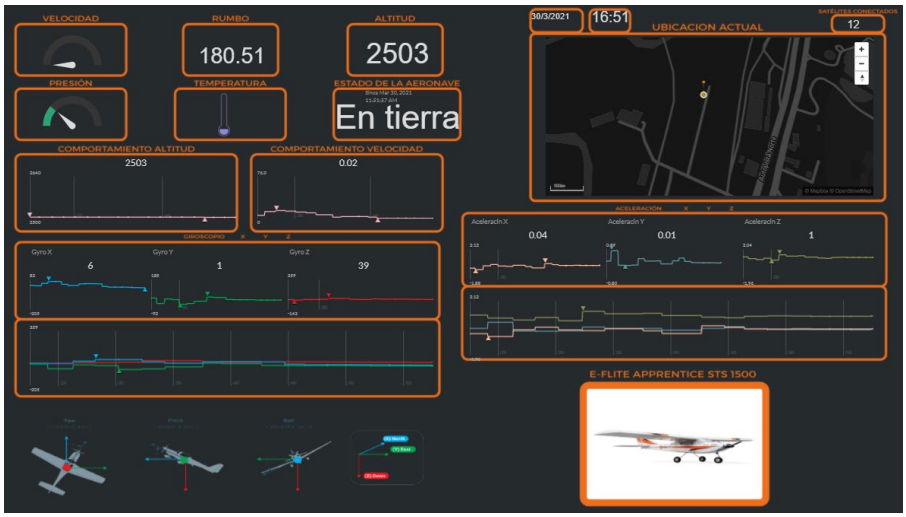


Figura 69. Visualización datos fase de aterrizaje. Fuente: Autor

En la Figura 69 se pueden observar los datos cuando la aeronave se encontraba aterrizando, de aquí se puede calcular el tiempo de vuelo total de la aeronave que se conoce el tiempo de despegue aproximado 11:35am y el tiempo de aterrizaje 11:51 am, un total de 16 minutos de vuelo un tiempo normal para un aeromodelo.

A continuación, se presentan los datos de todo el vuelo (Figura 70) y se recalcan los datos máximos alcanzados en cada una de las variables obtenidas (Tabla 14).

Figura 70. Visualización datos de todo el vuelo. Fuente: Autor

Variable	Mínimo	Máximo
Velocidad	0 Km/h	76 Km/h
Rumbo	0º	360º
Altitud	2503 metros	2630 metros
Presión	742 hPa	753 hPa
Temperatura	19 ºC	21 ºC
Satélites conectados	0	12
Gyro X	-205º	82
Gyro Y	-92º	185
Gyro Z	-142º	359
Aceleración X	-1.88 G	2.1 G
Aceleración Y	-0.8 G	0.59 G
Aceleración Z	-1.96 G	2.04 G

Tabla 14. Comparación datos mínimos vs máximos obtenidos. Fuente: Autor

De la Tabla 14 se puede hacer un análisis más generalizado, ya que para el caso de la aceleración se observan valores de hasta 2G, estos valores máximos se obtuvieron cuando la aeronave realizo movimientos bastante bruscos y se tuvo que realizar una recuperación, es decir en otras palabras cuando estuvo a punto de estrellarse. También se puede comprobar de una manera muy simple que los datos del Gyro en Z fuero correctos ya que estos no pasaron de 360º, el valor daba negativo cuando la aeronave giraba sobre el eje z en sentido antihorario, es decir si había un valor de -360 grados quería decir que la aeronave había realizado un giro completo por su izquierda sobre el eje z, los valores extremos del gyro en Y X se obtuvieron también en los momentos en los que se hizo necesario recuperar la aeronave. Otro dato que se puede analizar y comprobar fue el

comportamiento de la presión, pues a mayor altura menor presión y cuando la aeronave alcanzo los 2630 metros su presión era de 742 hPa (Figura 71).

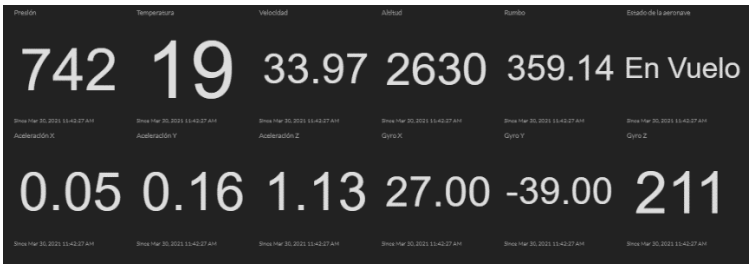


Figura 71. Datos al momento de alcanzar altitud máxima. Fuente: Autor

VIII. CONCLUSIONES

- El sistema desarrollado en este trabajo de grado muestra cómo se puede hacer un análisis de los datos de vuelo de una aeronave y así determinar la situación actual en la que esta se encuentra, ahora imaginemos que tan completo puede llegar a ser una análisis si se lograse obtener la mayoría de los datos de vuelo de la aeronave tales como, combustible abordo, presión de motor, presión de hidráulicos y demás datos que se puedan obtener.

- El prototipo Real Time Aircraft Monitoring System, se desarrolló e implemento de acuerdo con los objetivos establecidos en el proyecto, el uso de módulos GPS y sensores utilizados fue de fácil acceso y presentaron un bajo costo de adquisición, también se demuestra la capacidad que tiene el sistema de ofrecer al usuario la posibilidad de lograr hacer un análisis de todos los datos obtenidos durante todos los instantes del vuelo.

- El desarrollo del prototipo Real Time Aircraft Monitoring System, muestra la viabilidad que tiene el proyecto de servir como herramienta para mejorar la seguridad aérea en caso de emergencias en vuelo, permitiendo la visualización de los datos en tiempo real a un mecánico especializado para que este le ofrezca un soporte más oportuno al piloto.

- Cuando se cuenta con una variedad de sensores es imprescindible realizar la comprobación de cada parámetro para que no se presenten inconvenientes al momento de unir toda la adquisición en una sola, es decir hay que tener presente los tiempos de adquisición de datos de cada sensor, ya que de no tener presente este valor las tramas de datos puede que no se envíen completamente, para mitigar este problema se hizo la implementación del switch que se presentó en la Figura 39 para

que los datos se enviaran a medida que se adquirían y no enviando todos al tiempo como estaba previsto inicialmente.

- El sistema propuesto en este proyecto de grado puede evolucionar de tal forma que se incluyan otras variables a ser monitoreadas, el mejoramiento del sistema da una mayor versatilidad para la toma de decisiones durante un trayecto de vuelo. En la actualidad el sistema más utilizado realiza medición de variables en tiempo real para realizar procesos de mantenimiento preventivo después de ocurrida la incidencia.

- La calidad y precisión de los datos obtenidos depende directamente de las características del sensor que esté haciendo su adquisición, es decir, si se hace una implementación de este sistema con sensores de mejores características, la toma de datos es más precisa. A manera de ejemplo se puede cambiar el sensor inercial por uno que suministre directamente los datos de pitch, roll y yaw, lo cual proporcionara información de mayor exactitud respecto a estas variables.

IX. RECOMENDACIONES Y TRABAJO FUTURO

- Es importante mirar para proyectos futuros en pruebas de aeromodelos usar un sensor inercial que entregue directamente los datos del pitch, roll y yaw, para obtener mejores precisiones al momento de la adquisición, aunque cabe aclarar que en caso de aplicar se en aeronaves reales, estas ya cuentan con estos sensores por ende solo es necesario reescribir el algoritmo de adquisición de estos datos.

- Para futuras modificaciones se podría optimizar el algoritmo de programación para que este sea capaz de enviar alertas por excesos de fuerzas G, velocidad, limitaciones y entre otros, adicional quizá se pueda tratar de que estas alertas sean no solo para la estación base sino que también lleguen automáticamente a correos electrónicos para tener un mayor control, si se utiliza la misma plataforma de visualización initial state esta permite realizar esas alertas ya con los datos adquiridos y no desde el código fuente del programa.

- En caso de dar continuación al proyecto, en la parte de envío de datos a la web sería bueno realizar toda la adquisición directamente en bases de datos diseñadas por el futuro autor, cabe resalta que sería bueno que este proyecto se apoyara junto con un estudiante de ingeniería en sistemas para suplir la falta de conocimiento en el desarrollo de páginas web que el ingeniero electrónico no tiene, en caso de realizar la configuración de una base de datos es importante asegurarse que al momento del llamado de estos datos en la página web se cree una función que actualice los mismos cada cierto tiempo, ya que cuando se realizó esta prueba a pesar de que los datos estaban llegando a la base de datos el tiempo de visualización de estos dependía mucho del time to live (TTL) que

tuviera el hosting utilizado, en otras palabras si se quería ver una actualización de los datos tocaba recargar la página, y en algunos casos para que se visualizara una variación en las barras de la interfaz (Tabla de altitud vs tiempo) tocaba borrar el cache y recargar la página, este fue uno de los motivos que llevo al uso directo de una plataforma de iot como lo es initial state.

X. BIBLIOGRAFÍA

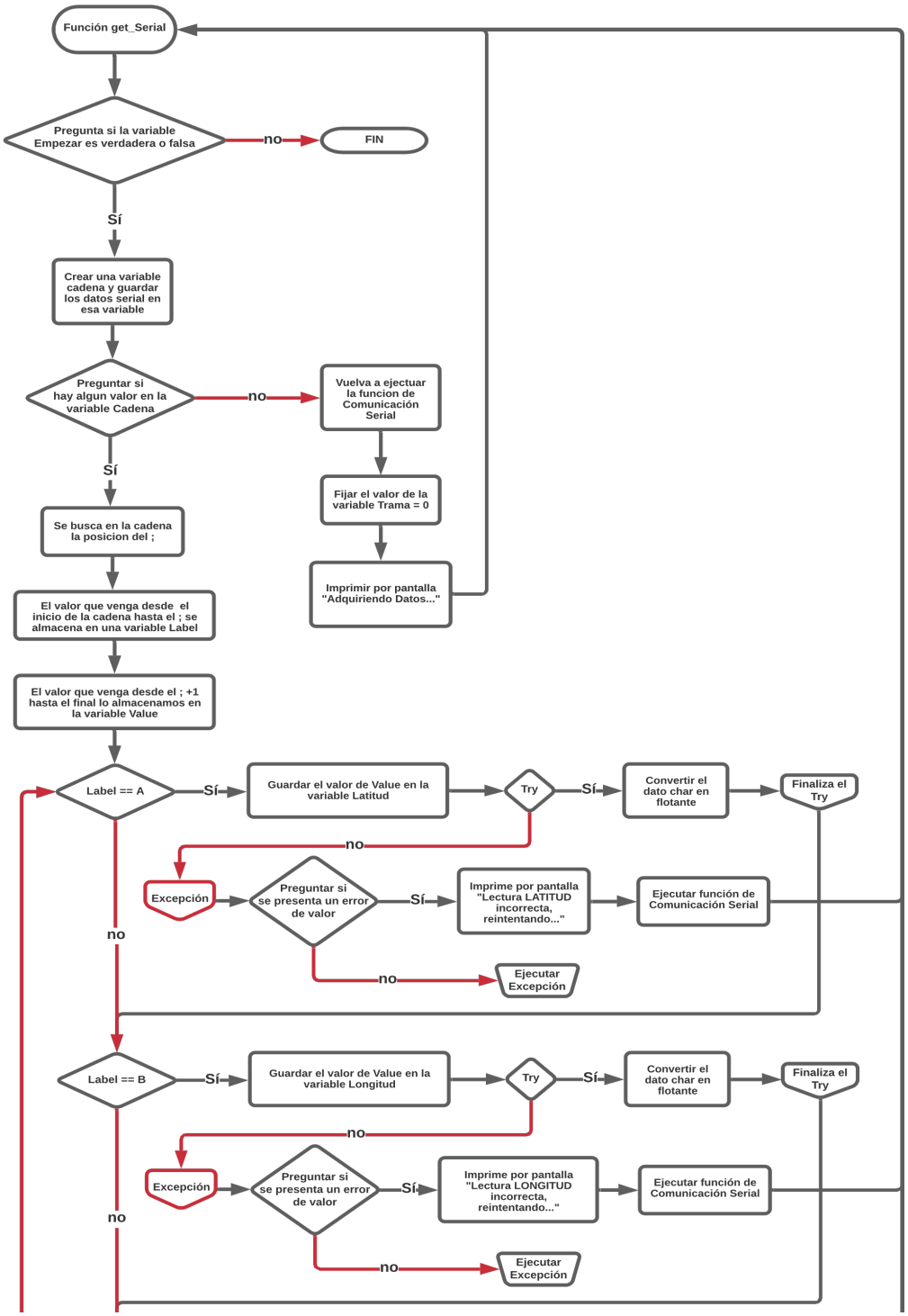
- Buitrago Zuluaga, H. M., & Raigosa Figueroa, M. A. (2016). Estudio de factibilidad para la creación de una empresa que presta el servicio de sistema de localización y monitoreo de pequeñas y medianas aeronaves. Pereira.
- Caballero, G., & Marinelli, J. (2015). Plataforma web para sistema distribuido de telemetría de un avión no tripulado. Cordoba.
- Echeverría, L. M. (2010). Plan de negocios para el desarrollo de un sistema automatizado de monitoreo de vuelo para empresas de transporte aéreo. Quito.
- Roy, A. (13 de enero de 2004). Estados Unidos Patente nº US 6.677.888 B2.
- *Air-Rc*. (s.f.). Obtenido de https://www.air-rc.com/aircraft/Volantex-RC_Ranger-1600_757-7
- Aycock, S. (s.f.). *techlandia.com*. Obtenido de Historia del microcontrolador: https://techlandia.com/diferencia-plc-microprocesador-hechos_547880/
- DIGI. (s.f.). *Xbee.cl*. Obtenido de Que es un Xbee.
- Duarte Muñoz, C. (01 de 10 de 2014). *Hacia el espacio*. Obtenido de http://haciaelespacio.aem.gob.mx/revistadigital/articul.php?interior=209
- Ferrer, V. (s.f.). Obtenido de Que es SIGFOX: https://vicentferrer.com/sigfox/
- Ferrer, V. (s.f.). Obtenido de Qué es Lora y Lorawan: https://vicentferrer.com/lora-lorawan/
- Ferretronica. (s.f.). *Ferretronica*. Obtenido de Modulo LM2596: https://ferretronica.com/products/modulo-lm2596-step-down-dc-dc-1-5v-35v-con-voltimetro?variant=19629234913373&currency=COP&utm_medium=product_sync&utm_source=google&utm_content=sag_organic&utm_campaign=sag_organic&utm_campaign=gs-2020-01-11&utm_source=go
- Fetick, R. J. (Enero de 2021). Obtenido de https://github.com/rfetick/MPU6050_light/blob/master/documentation_MPU6050_light.pdf
- Fritzing. (22 de 2 de 2021). *fritzing.org*. Obtenido de https://fritzing.org/
- Fundacion Raspberry Pi. (s.f.). *Raspberrypi.org*. Obtenido de https://www.raspberrypi.org/products/raspberry-pi-4-model-b/
- Garcia, C. (2012). Obtenido de Zigbee, comunicación para dispositivos: https://sg.com.mx/content/view/310

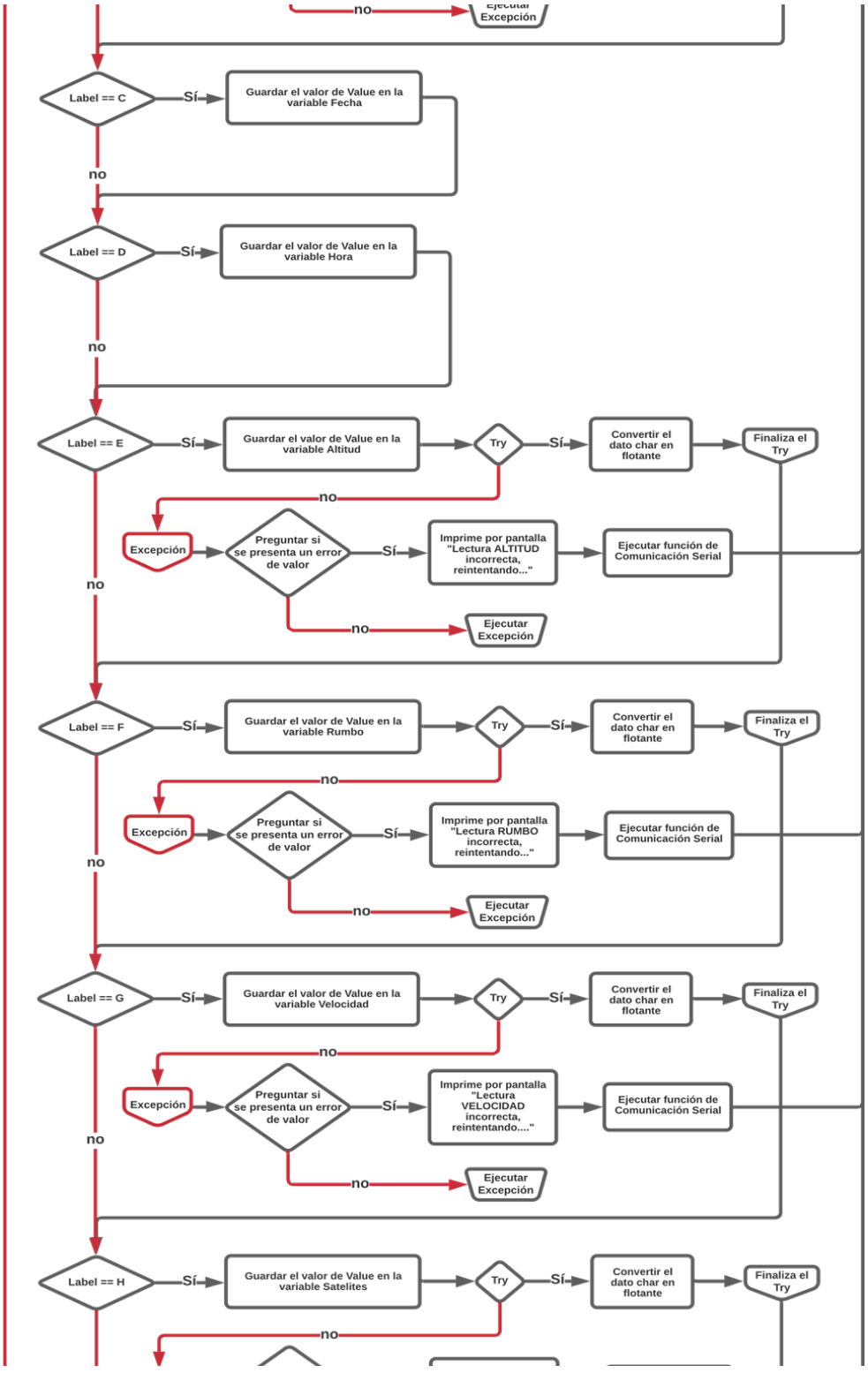
- Hernandez Carmona, G. A., & Lopez Monjaras, C. I. (2012). *Evolución del Sistema ACARS y nueva tecnologia en comunicación aire / tierra en la aviación.*
- *HorizonHobby*. (s.f.). Obtenido de https://www.horizonhobby.com/product/apprentice-sts-1.5m-rtf-smart-trainer-with-safe/EFL3700.html
- ICAO INT. (s.f.). *www.icao.in.* Obtenido de Manual de seguridad de vuelo para operaciones: https://www.icao.int/safety/fsix/Library/GAIN_OFSH_Spanish.pdf
- ICAO INT. (s.f.). *www.icao.int.* Obtenido de DESAPARECIDOS: EL CASO DEL VUELO 447 DE AIR FRANCE: https://www.icao.int/SAM/Documents/2016-EVALGEST/3-INFORME-CASO_TALLER%20VIGILANCIA_%20AIR%20FRANCE%20447.pdf
- *Mercado Libre*. (s.f.). Obtenido de https://www.google.com/url?sa=i&url=https%3A%2F%2Farticulo.mercadolibre.com.co%2FMCO-522129438-antena-gps-157542mhz-conector-sma-navegacion-gps-_JM&psig=AOvVaw07yjEt46TSLBSdY_f38gwo&ust=1616342832358000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCOivwb7jv-8CFQ
- meteocast.net. (s.f.). Obtenido de https://es.meteocast.net/week-forecast/co/chia/
- Sierra, C. (16 de 08 de 2012). *Blog*. Obtenido de El GPS : https://sites.google.com/site/elgpsglobalpositioningsystem/historia-del-gps
- Teslabem. (17 de 10 de 2018). *Teslabem.com*. Obtenido de Historia de la Raspberry Pi: https://teslabem.com/blog/historia-de-la-raspberry-pi/
- *Thing Big*. (s.f.). Obtenido de https://empresas.blogthinkbig.com/breve-historia-de-internet-de-las-cosas-iot/
- *Timetoast.com*. (s.f.). Obtenido de Historia módulos Xbee: https://www.timetoast.com/timelines/historia-de-los-modulos-xbee
- *Timetoast.com*. (s.f.). Obtenido de Historia comunicaciones inalambricas: https://www.timetoast.com/timelines/historia-y-evolucion-de-las-comunicaciones-inalambricas
- Unidad Administrativa Especial de Aeronáutica Civil. (Junio de 2020). *Reglamentos Aeronáuticos de Colombia RAC91.* Obtenido de https://www.aerocivil.gov.co/normatividad/RAC/RAC%20%2091%20-%20Reglas%20%20Generales%20de%20Vuelo%20y%20Operaci%C3%B3n.pdf

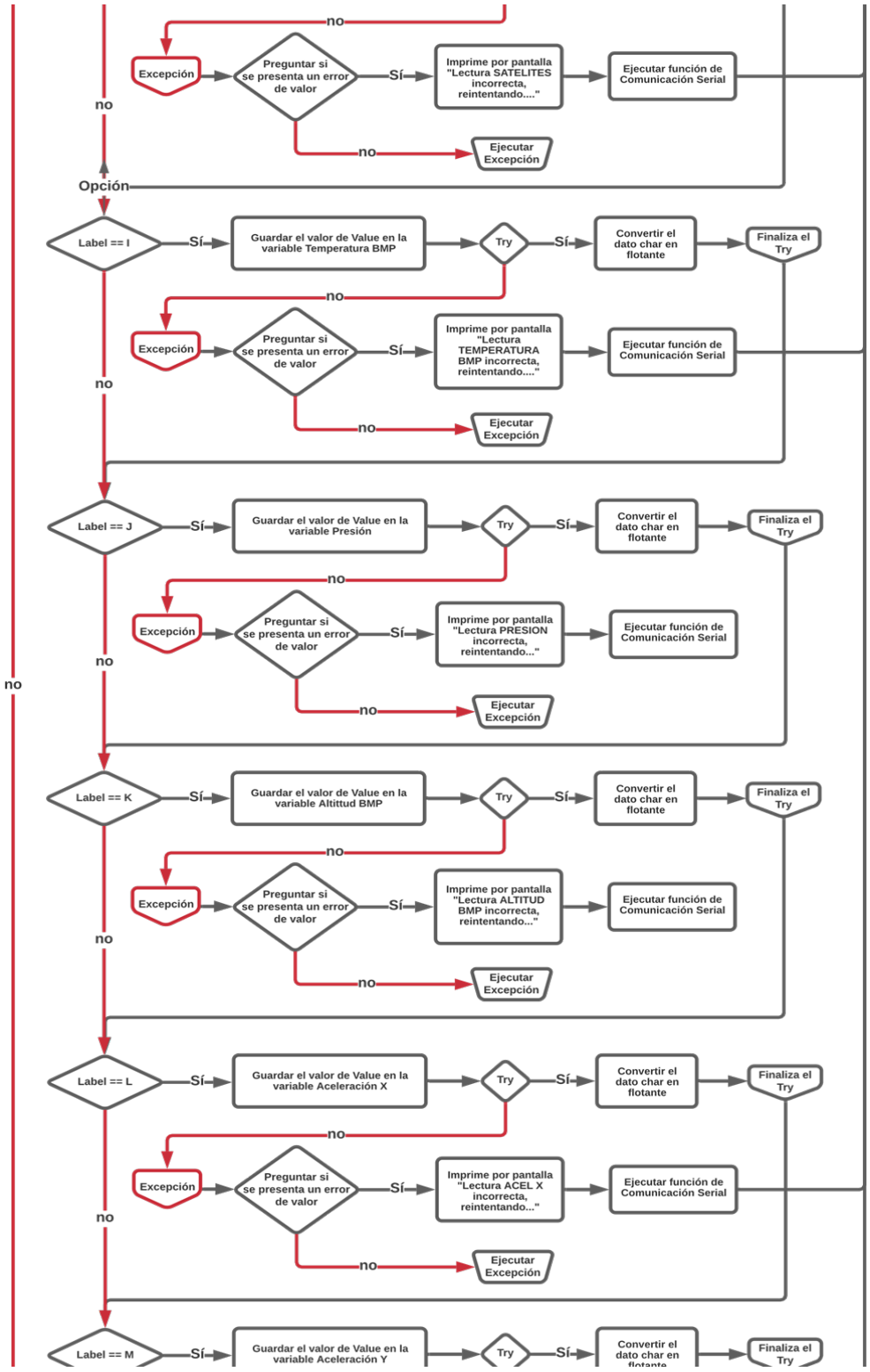
- *Universidad de Sevilla*. (s.f.). Obtenido de Fundamentos de la Navegación Aérea, Sistemas de Navegación Inercial: http://www.aero.us.es/fna/files/T8FNA_print.pdf
- *Vistronica*. (s.f.). Obtenido de https://www.vistronica.com/board-de-desarrollo/raspberry-pi/pantalla-7-inch-800x480-hdmi-touchscreen-raspberry-pi-detail.html

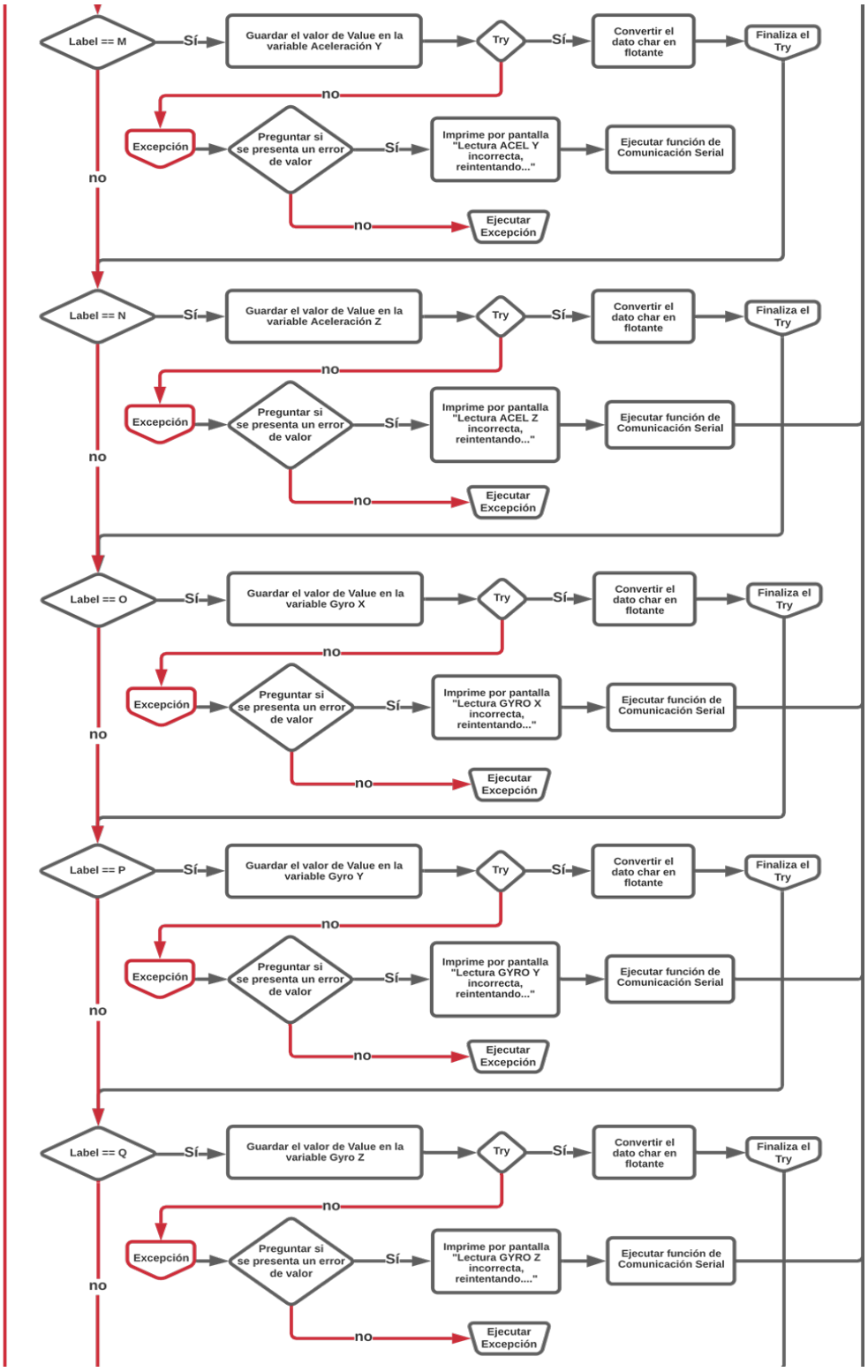
XI. ANEXOS

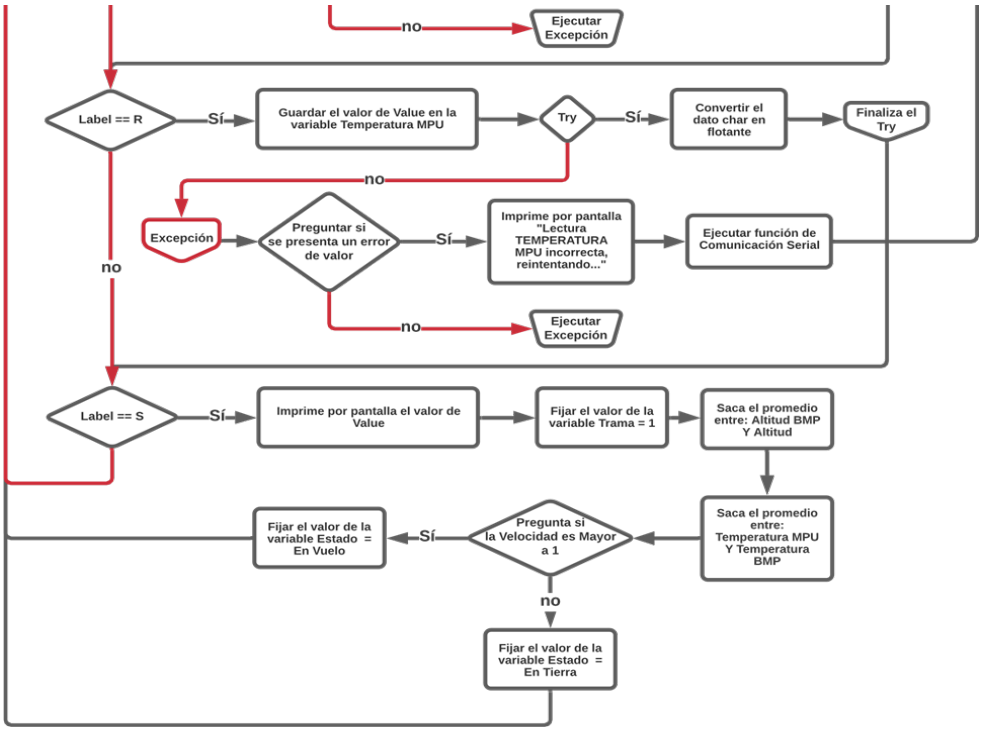
A. Diagrama de Flujo Función Comunicación serial




no
Ejecutar Excepción
Label == C
Sí
Guardar el valor de Value en la variable Fecha
no
Label == D
Sí
Guardar el valor de Value en la variable Hora
no
Label == E
Sí
Guardar el valor de Value en la variable Altitud
Try
Sí
Convertir el dato char en flotante
Finaliza el Try
no
Excepción
Preguntar si se presenta un error de valor
Sí
Imprime por pantalla "Lectura ALTITUD incorrecta, reintentando..."
Ejecutar función de Comunicación Serial
no
Ejecutar Excepción
no
Label == F
Sí
Guardar el valor de Value en la variable Rumbo
Try
Sí
Convertir el dato char en flotante
Finaliza el Try
no
Excepción
Preguntar si se presenta un error de valor
Sí
Imprime por pantalla "Lectura RUMBO incorrecta, reintentando..."
Ejecutar función de Comunicación Serial
no
Ejecutar Excepción
no
Label == G
Sí
Guardar el valor de Value en la variable Velocidad
Try
Sí
Convertir el dato char en flotante
Finaliza el Try
no
Excepción
Preguntar si se presenta un error de valor
Sí
Imprime por pantalla "Lectura VELOCIDAD incorrecta, reintentando...."
Ejecutar función de Comunicación Serial
no
Ejecutar Excepción
no
Label == H
Sí
Guardar el valor de Value en la variable Satelites
Try
Sí
Convertir el dato char en flotante
Finaliza el Try
no


no
Excepción
Preguntar si se presenta un error de valor
Sí
Imprime por pantalla "Lectura SATELITES incorrecta, reintentando...."
Ejecutar función de Comunicación Serial
no
Ejecutar Excepción
no
Opción
Label == I
Sí
Guardar el valor de Value en la variable Temperatura BMP
Try
Sí
Convertir el dato char en flotante
Finaliza el Try
no
Excepción
Preguntar si se presenta un error de valor
Sí
Imprime por pantalla "Lectura TEMPERATURA BMP incorrecta, reintentando...."
Ejecutar función de Comunicación Serial
no
Ejecutar Excepción
no
Label == J
Sí
Guardar el valor de Value en la variable Presión
Try
Sí
Convertir el dato char en flotante
Finaliza el Try
no
Excepción
Preguntar si se presenta un error de valor
Sí
Imprime por pantalla "Lectura PRESION incorrecta, reintentando..."
Ejecutar función de Comunicación Serial
no
Ejecutar Excepción
no
Label == K
Sí
Guardar el valor de Value en la variable Altittud BMP
Try
Sí
Convertir el dato char en flotante
Finaliza el Try
no
Excepción
Preguntar si se presenta un error de valor
Sí
Imprime por pantalla "Lectura ALTITUD BMP incorrecta, reintentando..."
Ejecutar función de Comunicación Serial
no
Ejecutar Excepción
no
Label == L
Sí
Guardar el valor de Value en la variable Aceleración X
Try
Sí
Convertir el dato char en flotante
Finaliza el Try
no
Excepción
Preguntar si se presenta un error de valor
Sí
Imprime por pantalla "Lectura ACEL X incorrecta, reintentando..."
Ejecutar función de Comunicación Serial
no
Ejecutar Excepción
no
Label == M
Sí
Guardar el valor de Value en la variable Aceleración Y
Try
Sí
Convertir el dato char en flotante
Finaliza el Try


Label == M
Sí
Guardar el valor de Value en la variable Aceleración Y
Try
Sí
Convertir el dato char en flotante
Finaliza el Try
no
Excepción
Preguntar si se presenta un error de valor
Sí
Imprime por pantalla "Lectura ACEL Y incorrecta, reintentando..."
Ejecutar función de Comunicación Serial
no
Ejecutar Excepción
no
Label == N
Sí
Guardar el valor de Value en la variable Aceleración Z
Try
Sí
Convertir el dato char en flotante
Finaliza el Try
no
Excepción
Preguntar si se presenta un error de valor
Sí
Imprime por pantalla "Lectura ACEL Z incorrecta, reintentando..."
Ejecutar función de Comunicación Serial
no
Ejecutar Excepción
no
Label == O
Sí
Guardar el valor de Value en la variable Gyro X
Try
Sí
Convertir el dato char en flotante
Finaliza el Try
no
Excepción
Preguntar si se presenta un error de valor
Sí
Imprime por pantalla "Lectura GYRO X incorrecta, reintentando..."
Ejecutar función de Comunicación Serial
no
Ejecutar Excepción
no
Label == P
Sí
Guardar el valor de Value en la variable Gyro Y
Try
Sí
Convertir el dato char en flotante
Finaliza el Try
no
Excepción
Preguntar si se presenta un error de valor
Sí
Imprime por pantalla "Lectura GYRO Y incorrecta, reintentando..."
Ejecutar función de Comunicación Serial
no
Ejecutar Excepción
no
Label == Q
Sí
Guardar el valor de Value en la variable Gyro Z
Try
Sí
Convertir el dato char en flotante
Finaliza el Try
no
Excepción
Preguntar si se presenta un error de valor
Sí
Imprime por pantalla "Lectura GYRO Z incorrecta, reintentando...."
Ejecutar función de Comunicación Serial
no
Ejecutar Excepción



Anexo 1. Diagrama de flujo función comunicación serial.

B. Articulo IV Encuentro Internacional de Investigación Universitaria (ENIIU 2020)

Desarrollo de un dispositivo de monitoreo en tiempo real de los sistemas de una aeronave mediante un protocolo de comunicación inalámbrica.

Rtams (Real-Time Aircraft Monitoring System)

Development of a device for real-time monitoring of an aircraft's systems using a wireless communication protocol.

Awazacko-Awazacko, Julian. Rodríguez, Daniel. Salazar, Angélica.

Universidad Santo Tomas / Estudiante, Tunja, Colombia
Universidad Santo Tomas / Docente, Tunja, Colombia
Universidad Santo Tomas / Docente, Tunja, Colombia

julian.awazacko@usantoto.edu.co

Abstract

Currently there are aircraft monitoring systems known as ACARS (Aircraft Communication Addressing And Reporting System), such systems provide limited information when making the acquisition of data, because these are only acquired when the aircraft is stopped and its information is not good enough to perform comprehensive maintenance to the aircraft, information such as passengers, fuel used do not serve much to the maintenance area is why this project presented as a thesis to obtain the degree of electronic engineer at the University Santo Tomas Tunja Section, proposes the development of a prototype that is capable of monitoring in real time most of the systems of an aircraft and thus be able to deliver the necessary information for good maintenance and monitoring, for this case monitoring is done to a UAV (Unmanned Aerial Vehicle) because its acquisition or construction is more affordable, this is only for the testing phase because the prototype will give an idea of the behavior at the time of making the acquisition and sending data frames within a scale aircraft, and may become functional in a commercial aircraft.

The project is in the development phase, initially proposing all the simulation part with sending and acquisition of the data frames, communicating an aircraft (Simulated in proteus), a prototype software (Application in Java) and monitoring software (Web page). In this phase the information collected by sensors simulated in proteus is sent, which detect the rpm of an engine, the height and the speed, these data are sent to a desktop application created in java, in the mentioned application a data frame is constructed, which, is sent to the web page and there the state in real time of these sensors is shown, visualizing them through the interface of the monitoring system of the navigation prototype.

Keywords

Aircraft, Monitoring, Prototype, System, Wireless

1. Marco Teórico

Monitoreo: Es el proceso sistemático de recolectar, analizar y utilizar información para hacer seguimiento a un proceso determinado.
(ONU MUJERES)
Comunicación Inalámbrica: Es aquella que se realiza a través de un medio no guiado para comunicar dos puntos.
ACARS: es un acrónimo de Aircraft Communication Addressing and Reporting System. Es un protocolo de datos, cuya función es proporcionar conectividad de datos basados en caracteres entre una aeronave y cualquier número de servicio terrestre.
(International Communications Group, 2006)

2. Metodología

La metodología planteada para esta sección se dividió en 3 grandes etapas: Investigación, Diseño e Implementación. En la primera de ellas se plantea una investigación aplicada ya que lo que se busca concretamente con el proyecto es darle solución a una problemática específica, la cual es el monitoreo en tiempo real de los sistemas de una aeronave, aquí se observaran algunos antecedentes que tienen relación con lo mencionado anteriormente, esto con el objetivo de tener una estructuración para el diseño del prototipo y validar su viabilidad, la segunda etapa consta del diseño en sí, tomando como base lo realizado en la fase de investigación con el fin de determinar y optimizar el prototipo tanto en software como en hardware, aunque en este articulo el proyecto se centra en el desarrollo del software, por último se encuentra la fase de implementación, todo esto después de realizar las simulaciones correspondientes y encontrar posibles errores en cuanto al fallo del montaje.

A continuación, se muestra en la figura 1 un diagrama de flujo del diseño metodológico.

Fig1. Diagrama de flujo diseño metodológico

3. OBJETIVOS

- **OBJETIVO GENERAL**
  Desarrollar un dispositivo capaz de monitorear los sistemas básicos de navegación en una aeronave, con el fin de ofrecer un soporte en tiempo real a los pilotos en caso de emergencia desde una estación base.
- **OBJETIVO ESPECIFICO**
  Diseñar e implementar un prototipo que adquiera variables de sensores de una aeronave (uav) y enviarlos a una página web.

4. CUERPO DEL TRABAJO

Etapa 1: Adquisición de datos a través de los sensores de vuelo simulados en proteus, y se envían a una aplicación escritorio en java.

Etapa 2: Se crea base de datos en MySQL, los cuales se visualizan en una página web.

Etapa 3: Los datos recibidos por la aplicación de escritorio se envían a la base de datos creada en la etapa 2.

Etapa 4: Desarrollo de la aplicación web para la visualización de los datos adquiridos a través de los sensores referenciados en la etapa 1.

Fig2. Desarrollo del proyecto – Interfaz escritorio – Interfaz web

CONCLUSIONES

- Debido al TTL (Time to Live) los datos no se pueden visualizar 100% en tiempo real, para mejorar este TTL se debe contar con un hosting más avanzado.
- Las bases de datos que se administran en la nube aseguran un acceso constante a la información desde cualquier ubicación en la que se cuente con acceso a internet, por esta razón esta herramienta nos permite el desarrollo de proyectos con un alto grado de versatilidad.

REFERENCIAS

International Communications Group. (17 de Abril de 2006). www.icao.int. Obtenido de https://www.icao.int/safety/acp/inactive%20working%20groups%20library/acp-wg-m-iridium-7/ird swg07-wp08%20-%20acars%20app%20note.pdf

ONU MUJERES. (s.f.). ONU MUJERES. Obtenido de https://www.endvawnow.org/es/articles/330-cul-es-el-monitoreo-y-la-evaluacin.html

Anexo 2. Articulo IV Encuentro Internacional de Investigación Universitaria (ENIIU 2020).

C. Ponencia Articulo IV Encuentro Internacional de Investigación Universitaria (ENIIU 2020)

Anexo 3. Certificado Ponencia IV Encuentro Internacional de Investigación Universitaria (ENIIU 2020).

D. Articulo revista Ingeniería E investigación

En busca de complementar la divulgación del trabajo realizado, se redactó un artículo en ingles siguiendo el formato establecido por la revista **Ingeniería E investigación** categorizada como Q3 perteneciente a la Universidad Nacional de Colombia, el artículo fue enviado el día 4 de mayo de 2021, ya aprobó la etapa preliminar y se encuentra en el proceso de revisión por parte del comité editorial.

Ingeniería e Investigación
para mí
mié, 12 may. 16:27 (hace 13 días)

INGENIERÍA E INVESTIGACIÓN
Tecnología e innovación con tradición y excelencia

Cordial Saludo,
Estimado Julian A.

Esperamos que este correo le encuentre muy bien.

A la fecha su manuscrito se encuentra en revisión preliminar, en los próximos días le estaremos enviando el resultado. También, le queremos aclarar los tiempos promedios del proceso de evaluación

El proceso de evaluación está conformado por 4 etapas más una etapa final de corrección de estilo y diagramación.

- **Etapa preliminar:** En esta ronda de evaluación el equipo editorial verifica que los autores hayan cumplido con todos los requisitos de postulación entre ellos; formato, licencia, originalidad. Tiempo promedio 5-10 días.
- **Etapa 1:** La etapa 1 de evaluación es realizada por los editores de la revista, y en ella se evalúa la pertinencia y desarrollo del tema. Tiempo promedio de 4-8 semanas.
- **Etapa 2:** Evaluación por parte de los pares asignados por la revista y en el que se evalúa el contenido específico del trabajo. Tiempo promedio 2 a 6 meses.
- **Etapa 3:** Evaluación después de las posibles correcciones sugeridas en primera ronda. Tiempo promedio 2-4 semanas.
- **Etapa final:** Después de la aprobación del documento por parte de los pares se realizan los ajustes finales al documento en cuanto a corrección de estilo y diagramación. Tiempo promedio 1 semana.

Preliminary Review

Participants

The Editorial Board of Ingeniería e Investigación (revii_bog)

Julian Awazacko (awazacko98)

Messages

Note	From
Dear Authors, Your article titled has successfully approved the preliminary review. In the coming days it will be evaluated by the editorial committee. As soon as we have a news we will be informing you. Regards,	revii_bog 2021-06-04 10:44 AM

Add Message

Anexo 4. Articulo revista Ingeniería e Investigación

DOI: http://dx.doi.org/10.15446/ing.investig.vxxnx.xxxx

Development Of A Prototype For Real-Time Scale Monitoring Of Aircraft Systems Through A Wireless Communication Protocol. RTAMS (REAL-TIME AIRCRAFT MONITORING SYSTEM)

Desarrollo De Un Prototipo De Monitoreo A Escala En Tiempo Real De Los Sistemas De Una Aeronave Mediante Un Protocolo De Comunicación Inalámbrica. RTAMS (REAL-TIME AIRCRAFT MONITORING SYSTEM)

Julian A. Awazacko Awazacko, Cesar M. Galarza Bogotá, Angelica M. Salazar Madrigal

ABSTRACT

Currently there are aircraft tracking systems known as ACARS (Aircraft Communication Addressing And Reporting System), these systems provide limited information when it comes to data acquisition, information such as passengers, fuel used do not serve much to the maintenance area if you want to provide real-time support, which is why this project presented as a thesis for the degree of electronic engineer at the University Santo Tomas Sectional Tunja, proposed the development of a prototype that was able to monitor in real time most of the systems of an aircraft and thus deliver the necessary information for good maintenance, monitoring and support in real time, for this case the monitoring was done to a RPAS (Remotely Piloted Aircraft System) since its acquisition or construction is more affordable, the flight prototype gave an idea of the behavior when performing the acquisition and sending of data frames inside the aircraft.

Keywords: Monitoring, **p**rototyping, **s**upport, **w**ireless communication.

RESUMEN

En la actualidad existen sistemas de seguimiento de aeronaves conocidos como ACARS (Aircraft Communication Addressing And Reporting System), dichos sistemas proporcionan una información limitada a la hora de realizar la adquisición de datos, información como pasajeros, combustible utilizado no sirven mucho al área de mantenimiento si se quisiera ofrecer un soporte en tiempo real, es por ello que este proyecto presentado como tesis para obtener el título de ingeniero electrónico en la Universidad Santo Tomas Seccional Tunja, propuso el desarrollo de un prototipo que fuera capaz de monitorear en tiempo real la mayoría de los sistemas de una aeronave y así entregar la información necesaria para un buen mantenimiento, monitoreo y soporte en tiempo real, para este caso el monitoreo se hizo a un RPAS (Remotly Piloted Aircraft System) ya que su adquisición o construcción es más asequible, el prototipo de vuelo dio una idea del comportamiento al momento de realizar la adquisición y envío de tramas de datos dentro de la aeronave.

Palabras clave: Monitoreo, **p**rototipo, **s**oporte, **c**omunicación inalámbrica.

Received: January __th 20xx
Accepted: January __th 20xx

Introduction

This work presents the design and implementation of a monitoring device that has the capability of acquiring flight data from an aircraft. The initial prototype acquires data of: pressure, temperature, altitude, speed, positioning (GPS), acceleration (x,y,z) and orientation (x,y,z) of a model aircraft (RPAS) in order to simulate some of the sensors that a commercial aviation aircraft has, since it does not have the necessary resources to access a real one. In general, the system will be performing this acquisition and it will be sent by wireless communication to a web page implemented through the use of IOT (Internet Of Things) technology.

It is contemplated the use of this system in the future for the total monitoring of a commercial flight in order to give support to a pilot in the shortest possible time knowing all the behavior of the aircraft. Additionally, the methodology used for the development of the project is shown, together with the results obtained and some conclusions.

1.Student of Electronic Engineering, Universidad Santo Tomas, Colombia.
2.Electronic Engineering, Universidad Santo Tomas, Colombia.
3.Electronic Engineering, Universidad Santo Tomas, Colombia.

Ancestors

Web platform for a distributed drone telemetry system (Caballero & Marinelli, 2015): In the development of ANT (Unmanned Aircraft) systems, it is important to know the status of the aircraft variables during the design, qualification, and operation phases. For this reason, the telemetry subsystem of an ANT system must distribute the information of the aircraft variables so that they can be monitored by different actors and at different levels of decision making. This is where the need arises to design a telemetry subsystem that can disseminate the information of the aircraft variables to different geographically spaced points, with a delay time of less than 1200 ms and that can be accessed from any remote site with an Internet connection. The main challenge they faced in this project was the implementation of the solution using tools that would allow delivery from an embedded GUI platform that provides flexibility for its operation.

The distributed telemetry subsystem for an ANT system consists of 30 primary and 48 secondary variables. All variables must be monitored in real time during the design, qualification and operation phases of the system with a maximum allowable delay of 1200 ms between the air segment and the engineering building or control office.

Feasibility study for the creation of a company that provides the service of location and monitoring system for small and medium-sized aircraft (Buitrago Zuluaga & Raigosa Figueroa, 2016): In the work developed by (Buitrago Zuluaga & Raigosa Figueroa, 2016) it is sought to locate and monitor aircraft through different geolocation systems, the authors also develop a feasibility study briefly explaining how some types of radars used in Colombian aviation work, their deficit in communications between air traffic controller and pilots.

The product on which they begin to work is the location system of each aircraft, something transcendental that comes from the beginning of commercial aviation, it is there where the need arose to locate the aircraft and therefore create an instrument known as RADAR, which is a system that uses electromagnetic waves to measure distances, altitudes, directions, among others.

Methodology

The methodology used in this project was divided into 3 main parts, in the first one the whole research was approached, then having an idea of what I wanted, we started to design so that at the end we could make a satisfactory implementation, the complete methodological design can be seen in Figure 1.

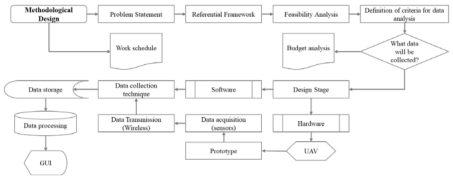


Figure 1. Methodological design of the project.
Source : Author

Investigation Stage

Type Of Research: For the development of the project, an applied research was proposed, since what was specifically sought was to provide a solution to a specific problem: Real-time monitoring of aircraft systems through a wireless communication protocol.

Technologies and operating principles: Technologies that had already worked in this type of project were evaluated and their basic operating principles were investigated to determine which was the most appropriate for this project.

Data processing and analysis: In this section, the data to be acquired for processing and analysis were determined, such as: speed, altitude, pressure, location... among others; in addition, some data collection and processing techniques were observed.

Design Stage

Software: The design of a GUI for the visualization of the aircraft data was carried out, initially it is proposed to acquire data from a UAV flight prototype to observe its behavior, this data will be sent to a web page to make possible its visualization from any device with internet access.

Hardware: This section consists of two implementations, the first one is the unmanned aerial vehicle, since it is where the acquisition of flight data is performed, as it is more complex to do it in a real aircraft.

Subsequently, the main device of the project was designed, a hardware that is able to perform such acquisition in real time of the UAV data and send them directly to a web page.

Implémentation And Evaluation Stage

Implementation and trial and error testing: Once the implementation of the whole system was done, the behavior of the whole system was evaluated to make the respective corrections.

Results

Investigation

ACARS System Overview: ACARS (Aircraft Communication Addressing and Reporting System), a data link transmission system in the VHF (129 MHz - 137 MHz) frequency range, is a system that basically provides a means by which airlines can exchange data and communicate with aircraft in the fleet in the same way as it is possible to exchange data over a terrestrial digital network. The main feature of this system is its ability to provide real-time data on the ground related to aircraft performance and then use this data to plan maintenance activities.

This data does not necessarily always travel over VHF but also provides for the use of HF and SATCOM sub-networks. The messages are known as uplinks and downlinks, facilitating two-way communication for applications such as ATIS (Automatic Terminal Information Service), clearances, weather/turbulence reports, flight delays, engine data reports, position, passenger load, fuel on board are also included. (Hernandez Carmona & Lopez Monjaras, 2012)

This information is usually requested by the company and retrieved at intervals, prior to ACARS this type of information would have been transferred via VHF voice communication.

Satellite Internet Overview: When talking about satellite communication, two concepts are usually kept in mind, uplinks (ground - satellite) and downlinks (satellite - ground), the objective of this communication is to allow main stations (transmitter and receiver) to communicate with each other using satellites as repeater stations.

Satellite Internet is a method of connecting to the Internet through electromagnetic waves using a satellite and a ground station as the main means of communication (Figure 2). Over the years it has been evolving and growing in the industry in general, since its coverage allows connectivity in remote areas. It should be noted that its main disadvantage is usually a low latency, but as time goes by these systems continue to improve and, in the future, the normal internet as it is known today will pass into the background, which is why this is a good candidate for this project.

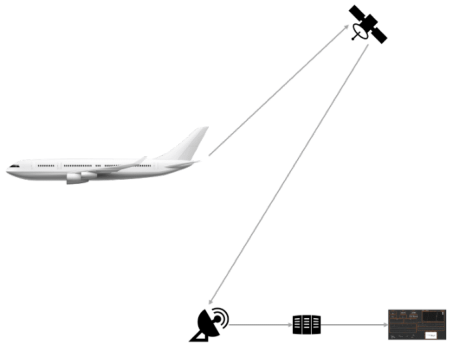


Figure 2. Satellite Internet Operation.
Source : Author

Satellite communication makes use of several frequency bands, within which we can appreciate the use of VHF, UHF among others, but in the case of satellite internet use, a much larger band is required to offer a better bandwidth, this is where the C, X and Ku bands come in, where each of them has a different application, For example, the C band works in frequencies of 4 - 8 GHz and is usually used for satellite television, then the X band (8 - 12 GHz) is used exclusively for military applications and then there is the Ku band, which works in frequencies from 12 to 18 GHz and is one of the most versatile of the microwave spectrum by providing good quality broadband services. Satellite internet reaches speeds from 25 Mbps up to 100 Mbps inclusive more.

The current project proposes a real time monitoring system with the use of satellite internet, since as mentioned above the communications of the previous system depend on the curvature of the earth and even if these aircraft fly very high in some parts of the world it is not possible to fly regularly as it is in the north of Canada, In addition to this, the RTAMS system does not propose to perform the same functions of an ACARS and at no time replace it, but it is a system capable of offering a complement when acquiring some data that the ACARS system does not usually obtain as it is a system focused mainly on preventive maintenance, with the aim of offering the pilot a real-time support in case of emergencies or breakdowns in the aircraft.

For the development of the project, it is not impossible the implementation in a real aircraft, for this reason it is proposed the use of a scale prototype (aeromodel) and additionally a wireless communication protocol is selected to communicate the aeromodel with a ground station and thus simulate a satellite communication in order to see what errors could occur.

The following is a description of some of the wireless communication protocols that were used to test the communication with the base station in the scale model.

SIGFOX: It is an IOT network that offers connectivity at low consumption and cost, providing M2M (Machine to Machine) solutions.

This technology depends on the operators in charge in each country, in the case of Colombia it is carried out by WDN group (Wireless Network Developments) and has created a long range and low speed network to allow communication between connected devices without the need to be directly linked to a mobile network coverage.

Its operation depends on a different network that works in 868 MHz, which means that its coverage is limited to areas where there are SIGFOX antennas, generating the need to install repeater equipment if more coverage is required. It also has a web interface that allows to register the equipment since they work by a unique identifier instead of a sim card (Ferrer, Qué es Sigfox, s.f.).

For this case the main limitation is not only the coverage, but also the number of packets that can be sent to the network per day, since it only allows sending 140 packets with a size of 12 bytes each and also has a low transmission speed, therefore the use of this protocol is discarded.

LORA: (Long RAnge), this technology allows connected devices to exchange long distance data packets at low speed and with low power consumption.

It works as a modulation technology to reduce costs and energy consumption, since the battery life can reach 3 or 5 years depending on the use case. One of the advantages of this protocol is that its signal has the ability to pass through buildings easily to reach remote corners such as subway or basements (Ferrer, Qué es Lora y Lorawan, s.f.).

Like SIGFOX it is also discarded, since its implementation in real-time applications is limited by the number of packets that can be sent.

LTE/4G: It is based on wireless transmission of broadband data that is mainly used for access to mobile telephony or portable internet devices, however, its coverage is associated with the telephone operator, implementation and maintenance costs are usually high and its high energy consumption, make it only be considered for sending information from the base station to the website.

ZIGBEE AND XBEE: Zigbee is a wireless technology that operates in the free ISM (Industrial, Scientific & Medical) bands of 2.4 GHz, reaches transmission speeds of 250 Kbps and a range of 10 to 75 meters, this technology is characterized by operating in

high density networks thus helping the reliability of communication, since the more nodes there are the more alternate routes there are to ensure that a package reaches its destination. The modules developed by XBee are integrated solutions that provide a means of wireless communication for the interconnection between devices using the IEEE 802.15.4 protocol, this allows the creation of point-to-point and multipoint networks, the XBee are designed for applications that require high data traffic and reduced latency, these manage to increase their range depending on the antenna that is attached to them.

Among the advantages of using XBee modules is that they offer a wireless communication of up to 16 km since they do not depend on another company but on the antenna and module that you have, nor does it have a limit of packets per day and that is why this technology was selected for wireless communication.

The following table shows a comparison of the technologies that were considered for the wireless communication part.

Table 1. Comparative table of wireless communication protocols

	SIGFOX	LORA	4G	ZIGBEE
Frequency	868 MHz	902 – 928 MHz	1700 – 2600 MHz	850 – 960 MHz
Coverture	Medium	Medium	High	Medium
Transfer limit	Yes	Yes	Not	Not
Power consumption	Low	Low	High	Medium
Advantages	Optimal application in devices that transmit infrequently Low power consumption	Small environment applications Bidirectional Correct operation when in motion Power consumption	Transmission speed Coverage in most of Colombia	Ideal for point-to-point and multipoint connections Economical Good mobility
Disadvantages	Coverage, very dependent on where antennas are located Low mobility Data transfer limits	High latency Coverage Data transfer limit	High energy consumption Costs While coverage is an advantage, in some cases it is also a disadvantage.	Transmission range depends on the antenna

Source : Author

In the end the selected technology was Zigbee with the support of XBee modules to obtain a greater transmission range, this was selected because it had a good performance and was not limited by the number of packets that could be transmitted per day, in addition to that its implementation has a relatively low cost.

Design

For the design of the prototype, it must be taken into account that it is not possible to add many sensors, because the aeromodel does not support too much weight and it is necessary to take into account the way in which this will be distributed on it. Basic concept of weight and balance.

Prototype Design: Figure 3 shows the design of the prototype which has. A microcontroller which is responsible for acquiring the data sent by the pressure sensors, GPS and IMU (Inertial Sensor), in turn this is responsible for sending the data via wireless communication with the XBee modules, once the data reaches the XBee receiver this is responsible for viewing them on a desktop application designed with Python and from there the data will be sent to the website.

The communication of the XBee modules is only necessary for the testing phase, since if the prototype were to be implemented in a real aircraft it must first have an internet connection.

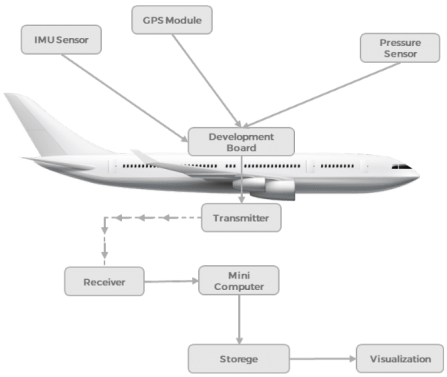


Figure 3. Prototype Design.
Source : Author

Sensors and Data Acquisition Modules: The sensors and modules of the final device are listed below.

- Development Board: Arduino Nano V3 (Arduino, s.f.)
- IMU Sensor: MPU6050 (MPU-6050 Datasheet(PDF) - List of Unclassifed Manufacturers, 2021)
- GPS Module: Neo 6M V2 (NEO-6M Datasheet(PDF) - u-blox AG, 2021)
- Pressure Sensor: BMP280 (BMP280 Datasheet(PDF) - Bosch Sensortec GmbH, 2021)
- Transmitter and Receiver: XBee S3B (DIGI, s.f.)
- Minicomputer: Raspberry Pi 4 Model B (Fundacion Raspberry Pi, s.f.)

Figure 4 shows the schematic circuit of the system, this part belongs only to the sensor information acquisition section. Since in the raspberry pi it is only necessary to connect the XBee Module via USB, no circuit design is required.

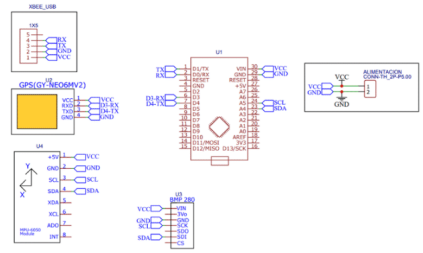


Figure 4. RTAMS System Schematic Circuit.
Source : Author

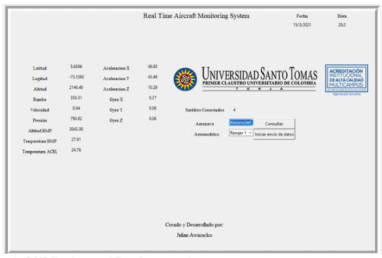


Figure 6. GUI Design and Implementation.
Source : Author

Implémentation

Considering the design phase, the complete implementation of the whole system is carried out. Figure 5 shows the final design of the prototype with the selected sensors and modules. The results obtained from the system will be described below.

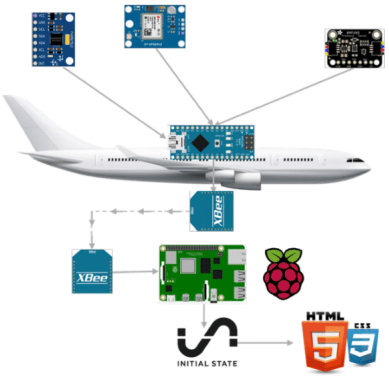


Figure 5. Prototype design with selected sensors and modules.
Source : Author

GUI and web page: A GUI (Figure 6) programmed with Python was designed and implemented to receive the data. This GUI will be installed in the Raspberry Pi 4B and from there the data will be sent to the IOT initial state platform, to later develop a web page (Figure 7) where the data will be displayed.

Figure 7. Website Design and Implementation on the IOT initial state platform.
Source : Author

In the implemented GUI the data acquisition is done through serial communication, that is why the Pyserial library is used, the program asks for the value immediately after the letter that is configured in the data frame as mentioned in the previous chapter, this code is shown in Figure 8.

```
def get_Serial(self):
    #RECOLECTANDO DATOS DEL PUERTO SERIE Y DECODIFICANDOLOS
    while self.empezar:

        cadena = self.comunicacion.readline().decode("ascii",errors='ignore').strip()

        if cadena:
            pos = cadena.find(";")
            label = cadena[:pos]
            value = cadena[pos+1:]

            if label == "A":
                self.data_latitud.set(value)
                try:
                    self.latitud=float(self.data_latitud.get())
                except ValueError:
                    print("Lectura LATITUD incorrecta, reintentando...")
                    self.get_Serial

            if label == "B":
                self.data_longitud.set(value)
                try:
                    self.longitud=float(self.data_longitud.get())
                except ValueError:
                    print("Lectura LONGITUD incorrecta, reintentando...")
                    self.get_Serial

            if label == "C":
                self.data_fecha.set(value)

            if label == "D":
                self.data_hora.set(value)

            if label == "E":
                self.data_altitud.set(value)
                try:
                    self.altitud=float(self.data_altitud.get())
                except ValueError:
                    print("Lectura ALTITUD incorrecta, reintentando...")
                    self.get_Serial
```

Figure 8. Programming for Data Acquisition by Serial Communication in Python.
Source : Author

To demonstrate that the system can track more than one aircraft and is able to visualize it on a web page, a section was created within the implemented GUI that allows the user to allow the user to choose which bucket the data will go to; the buckets were created in the Initial State platform (Figure 9).

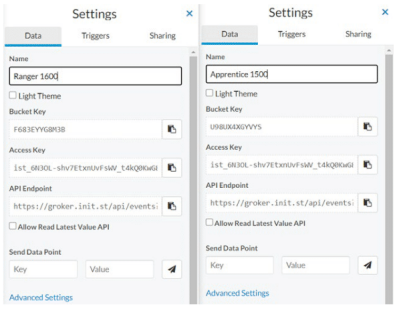


Figure 9. Bucket Initial State configuration.
Source : Author

Final Prototype Description: The following is a general description of the implemented system using a block diagram (Figure 10).

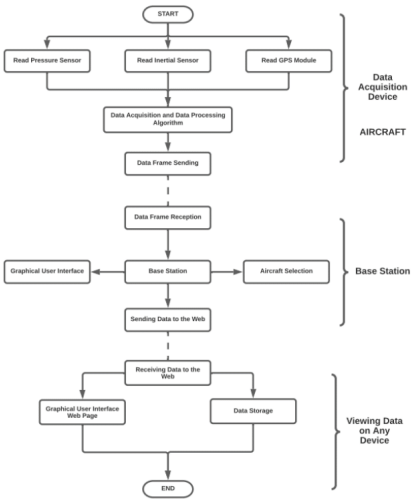


Figure 10. General System.
Source : Author

As shown in Figure 10, the general system is divided into three main parts, the first one is where the flight data acquisition is performed, then through the XBee communication it will be sent to the base station where the data will be displayed and from there it will be forwarded to the web page so that later the data can be accessed from any device anywhere that has an internet connection.

PCB Design and Implementation: Once the entire system has been previously tested on a breadboard and is working as explained in the previous section, based on the design of the electrical circuit (Schematic Figure 4), we proceed to design the PCB that goes inside the aircraft, this PCB uses the two layers of the PCB, this in order to avoid a crossover between the power pin of the pressure sensor and the power pin of the XBee module, in addition to this a logo, the project name and the author were placed.

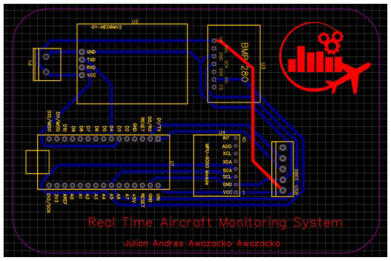


Figure 11. PCB System Design.
Source : Author

Prototype Implementation: In Figure 11 the blue colored paths correspond to the lower layer of the PCB and the red ones are from the upper layer.

The result of the system already installed in the aircraft can be seen in Figure 12 and 13, in addition to this system a 500mA fuse was added so that in case there is a shock or a bad connection in the model airplane and a short occurs, it does not burn the components.

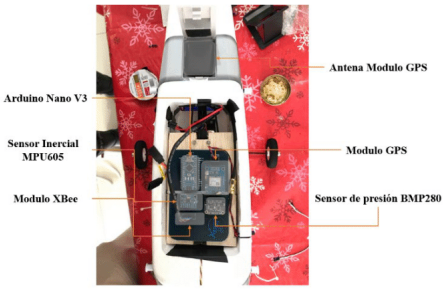


Figure 12. System fitted inside the Model Aircraft Top
Source : Author

Figure 12 shows the upper part of the PCB with the sensors and the development board already placed in the system, now in Figure 13 we will observe the lower part of the system where the voltage regulator is placed to feed everything at 5v and the Xbee module, since in this position the antenna of the XBe module can be placed in a better way.

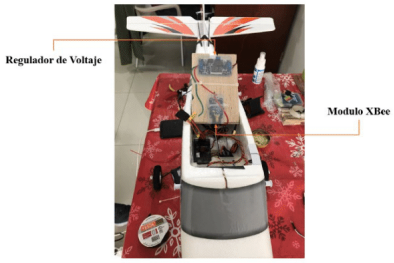


Figure 13. System fitted inside the Model Aircraft Part Below.
Source : Author

Figure 14 shows the lower part of the model airplane with the location of the battery that will feed the data acquisition system and the battery that will feed the entire model airplane system, also shows the location of the antenna and the GPS antenna cables that were placed to the system here special care was taken so that they do not interfere with the servomotors controlling the surfaces of the aircraft. On the right side of Figure 57, the system is seen laterally, and a switch used to turn on the RTAMS data acquisition system is observed.

Figure 14. Bottom and side of the model airplane.
Source : Author

Final Results and Data Analysis: Taking into account the Colombian aeronautical regulations, it is necessary to use a model airstrip to perform the system tests.

RAC 91 general flight and operation rules, Appendix 13 (Operation of unmanned aircraft systems UAS) Circular 328 AN/190 ICAO from August 05, 2019 (See Legal Framework).

The document also talks about the limitations of the aircraft that are in this category, within which are highlighted:

- Weight no more than 25 Kg
- Maximum speed of 80Km/h
- The aircraft must have a line of sight range with a maximum operating radius of 500 horizontal meters during all phases of flight.
- The operation cannot be performed over public, outdoor gatherings of people, agglomerations of buildings, cities or other populated areas.
- Operation must be during daylight hours under VFR rules and VMC conditions.
- Flight altitude not exceeding 123meters AGL (with respect to the ground).
- The flight must be performed within class G airspace, i.e. uncontrolled.Conclusions

Once the knowledge of these limitations was known, the selection of the place where the aircraft would fly was made, as it could be observed, flying the aircraft near the university is not possible since the law prevents it and it is a risk for the community.

The following are the results obtained from the system in the tests performed on March 30, 2021 at the delta 02 model airstrip located in the city of Chia Cundinamarca.

Figure 15. Aero model Ready For Take Off.
Source : Author

Figure 15 shows the model airplane ready to take off and from this moment on the system starts to track the aircraft's flight path.

Figure 16 shows how the base station starts the flight data acquisition, in this case it must be considered that the XBee receiving antenna is not 100% in line of sight, because there are obstacles such as the roof of the vehicle and the parking lot where it was located. In spite of this, the system performed all the acquisition correctly.

Figure 16. Graphical Interface Base Station and laptop for web data visualization.
Source : Author

Now, once the data were arriving to the base station, the system is instructed to start sending the data to the web page **https://www.monitoringsystem.online/es/apprentice.php,** this with the intention that they can be viewed on any device with an internet connection, for practical purposes, a laptop with internet connectivity was placed next to the base station to compare the data that were in the station and on the web.

Figure 17 shows the data obtained from the system at the moment the model aircraft was ready to take off, approximately at 11:35 am (16:35 Zulu), in the left part of the figure the route made by the aircraft from the hangar to the runway is observed, it is also verified that the current state of the runway and has an orientation of 300 degrees, the altitude at which the aircraft took off was 2503 meters.

Figure 17. Takeoff phase data display
Source : Author

Figure 18. Take-off phase data
Source : Author

In the section of temperature and pressure it is observed that the current value cannot be seen, for this sliding down a little on the web page you can see the data (Figure 18), here it is seen that the temperature was 21 degrees and the pressure of 753 hPa (Hectopascals), these data are compared with a meteorology platform called meteocast (https://es. meteocast.net/) to know more or less how much they vary, and it is observed that the pressure and temperature are within the expected ranges, because it should be noted that the weather page makes forecasts and does not give an exact value, so it can be said that the most accurate data are those provided by the system implemented in the aeromodel.

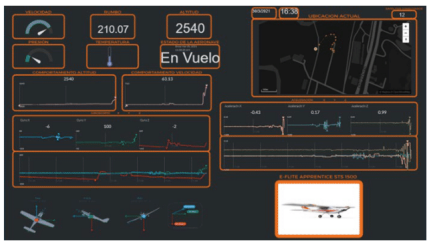


Figure 19. Visualization of the climbing phase data.
Source : Author

Figure 19 shows the ascent phase of the model aircraft at a speed of 63 km/h, here it can be verified that the data from the inertial sensor are being acquired correctly, since it is observed that the Y-axis turn known as pitch is at 100 degrees, this indicates a pronounced ascent of the model aircraft.

In the technical support document of the MPU_LIGHT library (Fetick, 2021) the author explains that the data in red and green color (Figure 20) belong to the angular velocity and linear acceleration respectively for the 3 axes. These data are obtained directly from the MPU 6050 and what the library does is to operate the data as shown in Figure 20, to obtain the values of the angles from the acceleration and the value of the angles from the fusion of the acceleration with the gyroscope data. The author comments that the data calculated from the acceleration are noisy and the angles calculated from the fusion of the accelerometer and gyroscope are more accurate.

The finally obtained value of the angles depends on the mathematical calculation of the integral of the angular velocity with respect to time as shown in Figure 66. This calculation in our case, due to the sensor update rate corresponding to 0.9 microseconds combined with the execution time of the sequential algorithm structure corresponding to both the library used MPU_LIGHT (Fetick, 2021) and the algorithm developed in this work for the acquisition of data from all sensors on the Arduino.

It generates an error in the calculation of the angles due to the integration time between one measurement and the next. The error is more noticeable when the aircraft performs sudden or fast movements and better results are obtained when the movements are slow or smooth. It should also be considered that once the initial calculation of the integral has been made, the algorithm executes a filter in order to reduce the noise on the obtained value. To solve this problem, it is recommended the acquisition of an INS sensor, which has a low update time and directly provides the value of the angles.

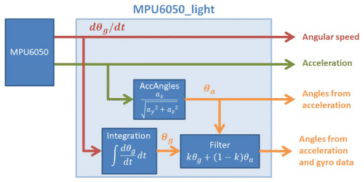


Figure 20. Overview of data accessible through the MPU6050_light library.
Source : (Fetick, 2021)

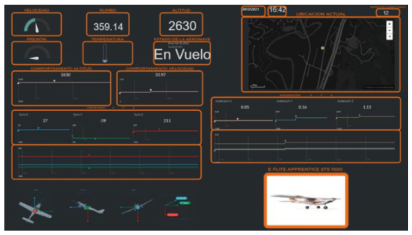


Figure 21. Cruise phase data display.
Source : Author

The cruise phase is when the aircraft reaches a constant flight altitude, in the case of model airplanes it is a little difficult to maintain it because there are variables such as the wind that can raise or lower the aircraft abruptly because it has a light weight, for this case in Figure 21 shows the data obtained at the time of reaching the maximum altitude which was 2630 meters, In other words, the aircraft reached 127 meters AGL (with respect to the ground), 4 meters above the allowed altitude. It should be clarified that this does not mean that it has entered into an illegal flight, since many times it is complex for the model aircraft pilot to calculate the height of the aircraft and in these cases the higher the better, since in case of an error there is more margin to control the model aircraft. It can also be seen that the speed at that moment was 33 km / h, half of what was obtained during the ascent and making a turn of 27 degrees (Gyro X).

Figure 22 shows the data at the moment of descent, here it is highlighted that the aircraft descends with the nose upwards and not as it would normally be thought with the nose downwards, that is why the Gyro in Y is of a positive value (24 degrees), a reduction of speed and altitude can also be appreciated, the Gyro X data indicates that the aircraft is doing a roll, either to the left or to the right and the Gyro in Z indicates the yaw.

Figure 22. Descent phase data display.
Source : Author

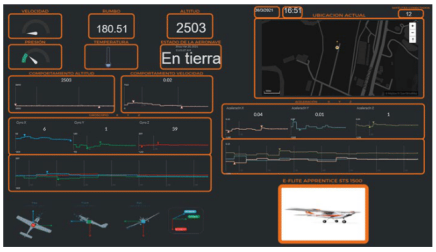


Figure 23. Landing phase data display.
Source : Author

In Figure 23 you can see the data when the aircraft was landing, from here you can calculate the total flight time of the aircraft which is known the approximate takeoff time 11:35 am and landing time 11:51 am, a total of 16 minutes of flight time a normal time for a model aircraft.

The data for the entire flight are presented below (Figure 24) and the maximum data achieved for each of the variables obtained are highlighted (Table 2).

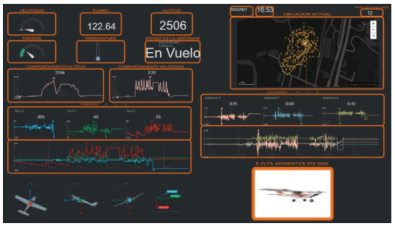


Figure 24. Display of all flight data.
Source : Author

From Table 2 a more generalized analysis can be made, since in the case of acceleration values of up to 2G are observed, these maximum values were obtained when the aircraft made quite abrupt movements and had to perform a recovery, in other words, when it was about to crash. It can also be verified in a very simple way that the data of the Gyro in Z was correct since these did not exceed 360°, the value was negative when the aircraft rotated on the z-axis counterclockwise, that is, if there was a value of -360 degrees it meant that the aircraft had made a

complete left turn on the z-axis, the extreme values of the gyro in Y and X were also obtained at the times when it was necessary to recover the aircraft. Another data that can be analyzed and verified was the pressure behavior, since the higher the altitude the lower the pressure, and when the aircraft reached 2630 meters its pressure was 742 hPa (Figure 25).

Table 1. Comparative table of wireless communication protocols

Variable	Mínimo	Máximo
Velocidad	0 Km/h	76 Km/h
Rumbo	0°	360°
Altitud	2503 metros	2630 metros
Presión	742 hPa	753 hPa
Temperatura	19 °C	21 °C
Satélites conectados	0	12
Gyro X	-205°	82
Gyro Y	-92°	185
Gyro Z	-142°	359
Aceleración X	-1.88 G	2.1 G
Aceleración Y	-0.8 G	0.59 G
Aceleración Z	-1.96 G	2.04 G

Source : Author

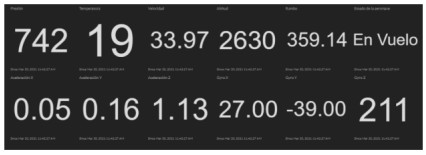


Figure 25. Data at maximum altitude.
Source : Author

Conclusions

- The system developed in this degree work shows how you can make an analysis of the flight data of an aircraft and thus determine the current situation in which this is, now imagine how complete can become an analysis if you manage to get most of the flight data of the aircraft such as fuel on bo ard, engine pressure, hydraulic pressure and other data that can be obtained.
- The Real Time Aircraft Monitoring System prototype was developed and implemented in accordance with the objectives established in the project, the use of GPS modules and sensors used was easily accessible and presented a low acquisition cost, it also demonstrates the ability of the system to offer the user the ability to achieve an analysis of all data obtained during all moments of the flight.
- The development of the Real Time Aircraft Monitoring System prototype shows the viability of the project to serve as a tool to improve air safety in case of in-flight emergencies, allowing the visualization of data in real time to a specialized mechanic so that he can offer a more timely support to the pilot.
- When there are a variety of sensors, it is essential to check each parameter so that there are no problems when joining all the data acquisition in one, i.e. it is necessary to keep in mind the data acquisition times of each sensor, because if this value is not taken into account, the data frames may not be sent completely. To mitigate this problem, the implementation of the switch shown in Figure 39 was made so that the data would be sent as they were acquired and not sending them all at the same time as initially planned.
- The system proposed in this degree project can evolve in such a way that other variables to be monitored are included; the improvement of the system provides greater versatility for decision making during a flight path. Currently, the most widely used system measures variables in real time to perform preventive maintenance processes after the occurrence of an incident.
- The quality and accuracy of the data obtained depends directly on the characteristics of the sensor that is making its acquisition, i.e., if this system is implemented with sensors with better characteristics, the data acquisition is more accurate. As an example, the inertial sensor can be replaced by one that directly supplies pitch, roll and yaw data, which will provide more accurate information regarding these variables.

Acknowledgements

To my father for giving me his unconditional support and becoming a role model as a person, especially to thank him for his effort and dedication, for being there for me in difficult moments and to my mother who from heaven somehow guided and illuminated the way to successfully accomplish this goal.

To my sister for her advice and for teaching me to have patience in moments where it was impossible to have it, for always being there by my side and to my brother-in-law Juan Parra who has been a great advisor during my time at the university.

Special thanks to my undergraduate tutors who supported me and gave me all their collaboration to successfully complete this degree project.

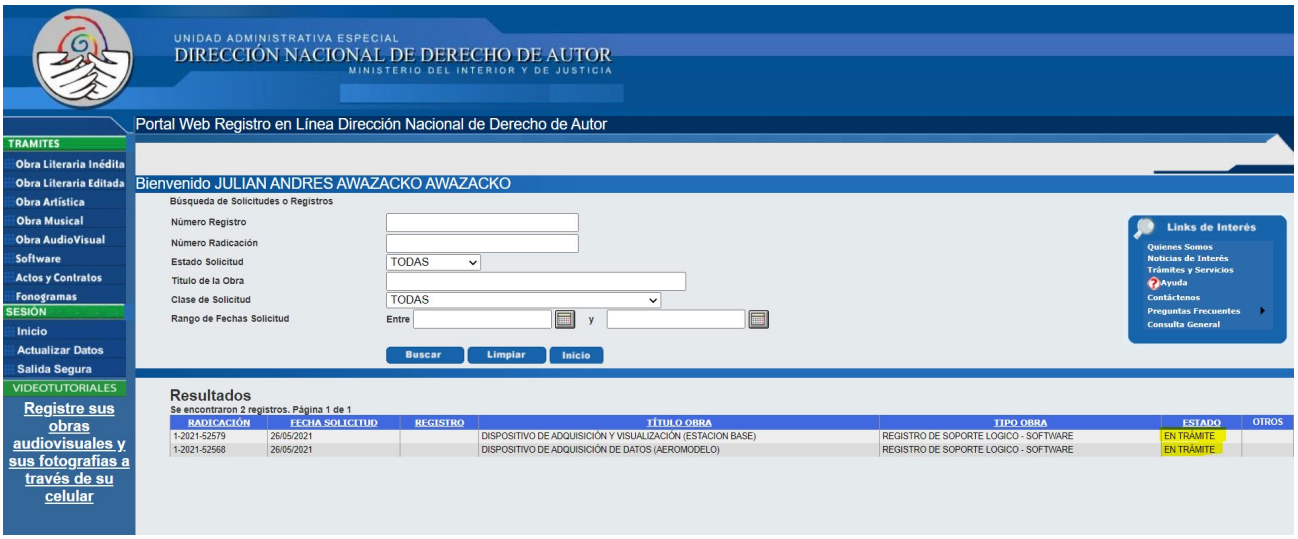
References

- Arduino. (s.f.). Arduino Nano | Arduino Official Store. Arduino Nano. Retrieved May 3, 2021, from https://store.arduino.cc/arduino-nano
- BMP280 Datasheet(PDF) - Bosch Sensortec GmbH. (2021). Retrieved 3 May 2021, from https://www.alldatasheet.com/datasheet-pdf/pdf/1132069/BOSCH/BMP280.html
- Buitrago Zuluaga, H. M., & Raigosa Figueroa, M. A. (2016). Estudio de factibilidad para la creación de una empresa que presta el servicio de sistema de localización y monitoreo de pequeñas y medianas aeronaves. Pereira.
- Caballero, G., & Marinelli, J. (2015). Plataforma web para sistema distribuido de telemetría de un avión no tripulado. Cordoba.

- DIGI. (s.f.). Xbee.cl. Obtenido de Que es un Xbee
- Ferrer, V. (s.f.). Obtenido de Qué es Lora y Lorawan: https://vicentferrer.com/lora-lorawan/
- Ferrer, V. (s.f.). Obtenido de Que es SIGFOX: https://vicentferrer.com/sigfox/
- Fetick, R. J. (Enero de 2021). Obtenido de https://github.com/rfetick/MPU6050_light/blob/master/documentation_MPU6050_light.pdf
- Fundacion Raspberry Pi. (s.f.). Raspberrypi.org. Obtenido de https://www.raspberrypi.org/products/raspberry-pi-4-model-b/
- Hernandez Carmona, G. A., & Lopez Monjaras, C. I. (2012). Evolución del Sistema ACARS y nueva tecnologia en comunicación aire / tierra en la aviación.
- MPU-6050 Datasheet(PDF) - List of Unclassifed Manufacturers. (2021). Retrieved 3 May 2021, from https://www.alldatasheet.com/datasheet-pdf/pdf/517744/ETC1/MPU-6050.html
- NEO-6M Datasheet(PDF) - u-blox AG. (2021). Retrieved 3 May 2021, from https://www.alldatasheet.com/datasheet-pdf/pdf/1283987/U-BLOX/NEO-6M.html

E. Derechos de Autor

Se realiza el proceso de solicitud de registro de software de la aeronave (dispositivo de adquisición de datos) y el software la estación base (dispositivo de adquisición y visualización).



Anexo 5. Derechos de Autor

F. Manual Técnico y Manual de Usuario

MANUAL TECNICO Y DE USUARIO

RTAMS (REAL-TIME AIRCRAFT MONITORING SYSTEM)

JULIAN ANDRÉS AWAZACKO AWAZACKO

UNIVERSIDAD SANTO TOMÁS

FACULTAD DE INGENIERÍA ELECTRÓNICA

SECCIONAL TUNJA

2021

Tabla de Contenido

Lista de Figuras

Lista de Tablas

I. INTRODUCCION

El prototipo RTAMS (Real Time Aircraft Monitoring System) es un sistema que permite adquirir los datos de vuelo de una aeronave, tales como: presión, temperatura, altitud, velocidad, posicionamiento (GPS), aceleración(x,y,z), magnitud(x,y,z) y la orientación(x,y,z), para este caso se explicara el funcionamiento en un aeromodelo (RPAS) esto con el fin de simular algunos de los sensores con los que cuenta una aeronave de aviación comercial, puesto que no se cuentan con los recursos necesarios para acceder a una real. En general el sistema estará realizando dicha adquisición y la esta renviado por comunicación inalámbrica a una página web implementada mediante el uso de la tecnología IOT (Internet Of Things).

En el presente manual se encontrará una guía de uso del sistema RTAMS, se abordarán secciones de especificaciones técnicas, recomendaciones y manual de inicio.

II. OBJETIVO

Informar a los usuarios sobre el funcionamiento del sistema RTAMS y sus especificaciones técnicas.

III. RECOMENDACIONES

- Leer el presente manual antes de iniciar el sistema RTAMS.
- Verifique las conexiones según como se muestra en el diagrama de conexiones del sistema.
- Asegurarse de que el voltaje de alimentación es el indicado antes de iniciar el dispositivo.
- Verificar que la estación base cuente con una conexión estable de internet, preferiblemente superior a 10 Mbps.
- Tener instaladas todas las librerías mencionas en la sección de descripción del sistema.
- En caso de presentar fallos detener el sistema inmediatamente y verificar todos los componentes.

IV. ESPECIFICACIONES TECNICAS

Alimentación	Solo 5V Dc – 1300mAh mínimo
Dimensiones	Largo: 15 Cm Ancho: 10 Cm Alto: 5 Cm
Grado de protección	IP 10
Temperatura de operación	0 – 75 ºC

Tabla 15. Especificaciones Técnicas dispositivo RTAMS.

Dentro de las especificaciones técnicas de la Tabla 1 no se incluyen las dimensiones de la antena del módulo XBee ni de la antena GPS.

Alimentación	Solo 5V Dc, 3A con entrada tipo C
Dimensiones	Largo: 7.1 Cm Ancho: 9.4 Cm Alto: 3 Cm
Grado de protección	IP 10
Temperatura de operación	0 – 75 ºC

Tabla 16. Especificaciones Técnicas estación Base.

Dentro de las especificaciones técnicas de la Tabla 2 no se incluyen las dimensiones de la antena del módulo XBee ni las dimensiones del monitor de visualización, para este caso una pantalla HDMI de 7 Pulgadas.

V. DESCRIPCION DEL SISTEMA

El dispositivo de monitoreo en tiempo real RTAMS es un sistema que está dividido en dos dispositivos un dispositivo de recolección y otro de adquisición y visualización (Estación base), el primero de ellos se encarga de hacer la recolección de los datos de vuelo de un aeromodelo los cuales son:

- Fecha y Hora
- Latitud y Longitud
- Temperatura
- Presión
- Velocidad

- Rumbo
- Aceleración en X, Y y Z
- Giroscopio en X, Y y Z

y enviarlos por comunicación inalámbrica (XBee) a una estación base.

El segundo dispositivo se encarga de recibir los datos mencionados anteriormente y permite la visualización de estos con el objetivo de verificar la integridad de los datos que llegan y para posteriormente enviarlos a una página web para que puedan ser visualizados en cualquier dispositivo con una conexión a internet.

Una vez los datos se encuentran en la página web esta permite realizar un análisis y graficar los datos para ver su comportamiento durante todo un vuelo y generar un historial.

VI. DIAGRAMA DE CONEXIONES

El diagrama de conexiones se mostrará de dos formas, una gráfica y una esquemática.

Dispositivo de recolección

El dispositivo de recolección este compuesto por:

- Arduino Nano V3
- Batería de 5v
- Modulo XBee Pro S3B
- Modulo GPS NEO 6 V2

- Sensor de Presión BMP280
- Sensor Inercial MPU6050

Si no se cuenta con una batería de 5 Voltios exactos se puede utilizar otra de diferente capacidad y a la entrada incluirle un regulador de voltaje para que este entregue los 5V.

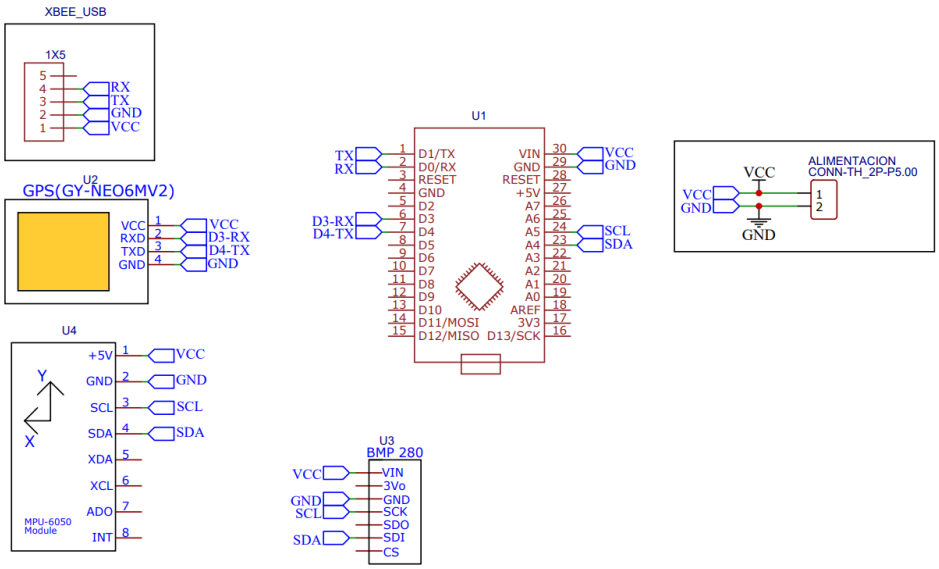


Figura 72. Circuito Esquemático Sistema RTAMS.

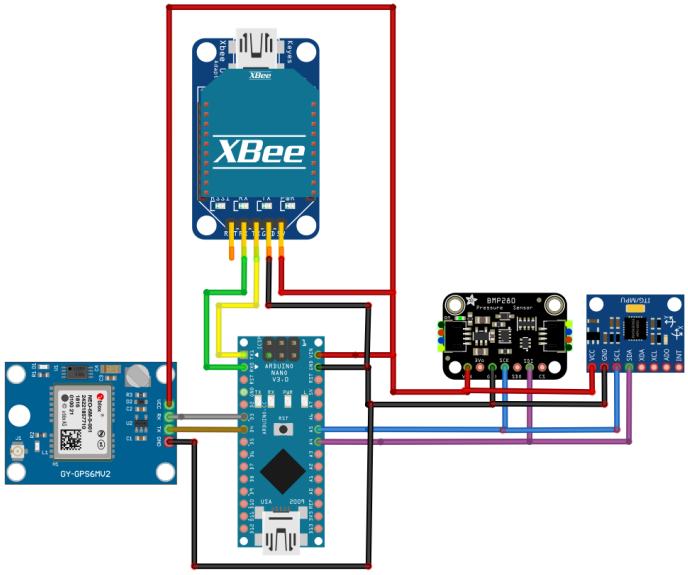


Figura 73. Circuito Grafico Sistema RTAMS.

Color	Arduino Nano V3	Pin Dispositivo	Dispositivo
Rojo	5V – Vin	Vin	TODOS
Negro	GND	GND	TODOS
Gris	D3	Rx	Modulo GPS NEO 6
Marrón	D4	Tx	Modulo GPS NEO 6
Verde	Rx	Rx	Modulo XBee
Amarillo	Tx	Tx	Modulo Xbee
Azul	SCL	SCK ó SCL	BMP280 y MPU6050
Morado	SDA	SDI ó SDA	BMP280 y MPU6050

Tabla 17. Conexiones Circuito Grafico Sistema RTAMS.

Dispositivo de Adquisición y Visualización (Estación Base)

Este dispositivo está compuesto por:

- Raspberry Pi 4 Modelo B
- Pantalla HDMI 7 Pulgadas
- Modulo XBee Pro S3B
- Teclado
- Mouse

Las conexiones de este dispositivo son más sencillas puesto que todas van por USB y la pantalla por mini HDMI.

VII. GUIA DE INICIO

Dispositivo de recolección

Una vez cargado el programa en el microcontrolador Arduino, solo hace falta encenderlo y listo, cabe aclarar que el sistema empezara la recolección de datos una vez se empiece a adquirir datos de latitud y longitud.

Asegurarse de que la conexión entre el del Módulo XBee USB y el XBee es correcta, pues es normal que cuando se mueva bruscamente el dispositivo esta se desconecte.

Dispositivo de Adquisición y Visualización (Estación Base)

Antes de ejecutar el programa principal es importante asegurarse de que se cuenta con Python 3.X en el dispositivo y que este tiene instaladas las siguientes librerías:

- Tkinter
- Pyserial
- Time
- Threading
- ISStreamer

a. Inicialización del programa Main.py y mainFrame.py

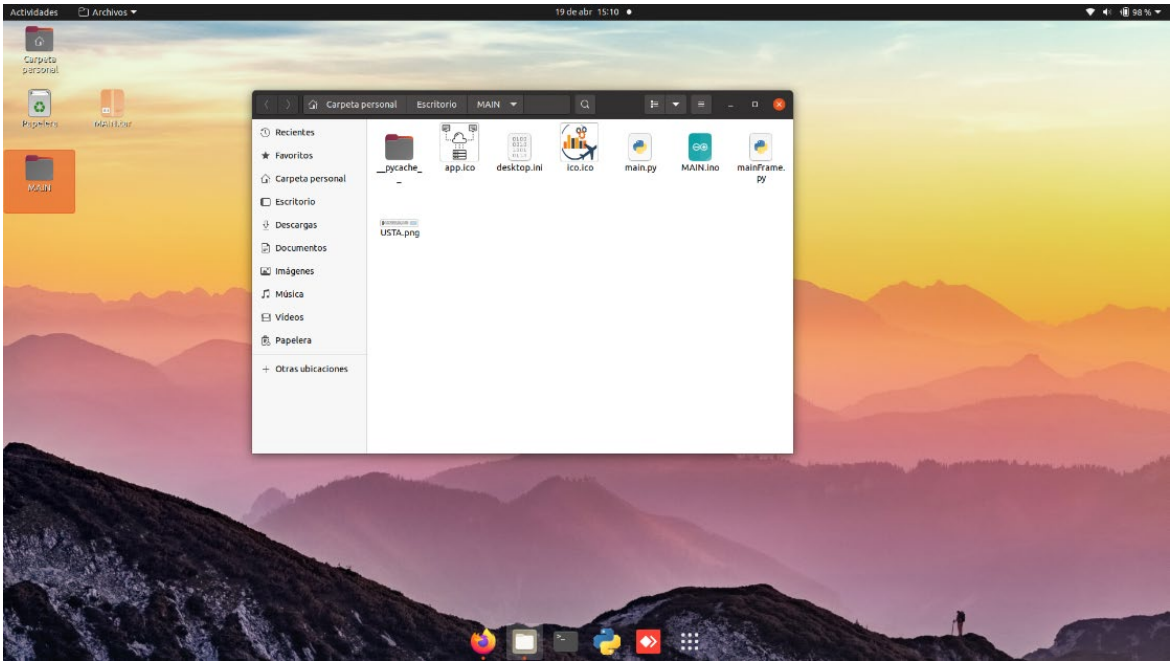


Figura 74. Inicialización Programa Main.py y mainFrame.py.

En la Figura 3 se muestra el contenido de la carpeta que se entregara:

- Iconos y logos del programa.
- 2 archivos de Python, uno es para la ventana principal y el otro es para inicializar el resto de las funciones.
- 1 archivo de Arduino, este es el que se carga en la placa de desarrollo.

Se inicia el programa main.py y el mainFrame.py, el main será el que se ejecutara posteriormente y el mainFrame es para revisar que el puerto serial se declare correctamente (Figura 4). Esto es ya que el programa también puede funcionar en

Windows y allí requiere una configuración diferente, esta configuración se encuentra comentada en el código entregado.

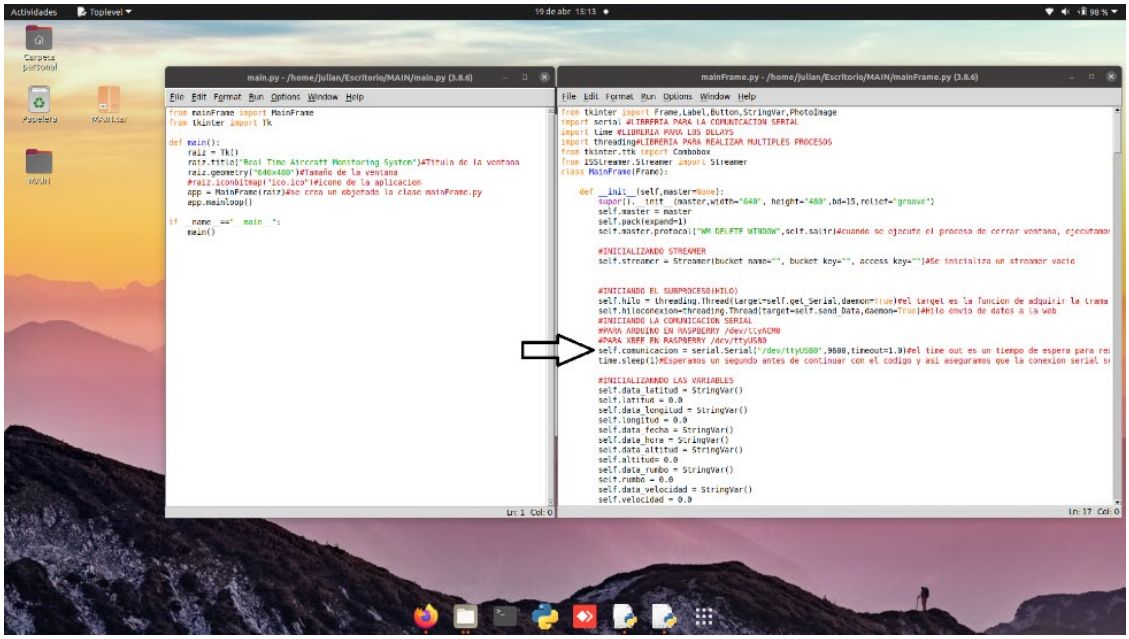


Figura 75. Declaración Puerto Serial.

b. Ejecución del programa main.py

Una vez se comprueba que el puerto serial esta declarado correctamente se ejecuta el programa main.py y se revisa en el Shell de Python que este se encuentre adquiriendo datos y no envié ningún error (Figura 5).

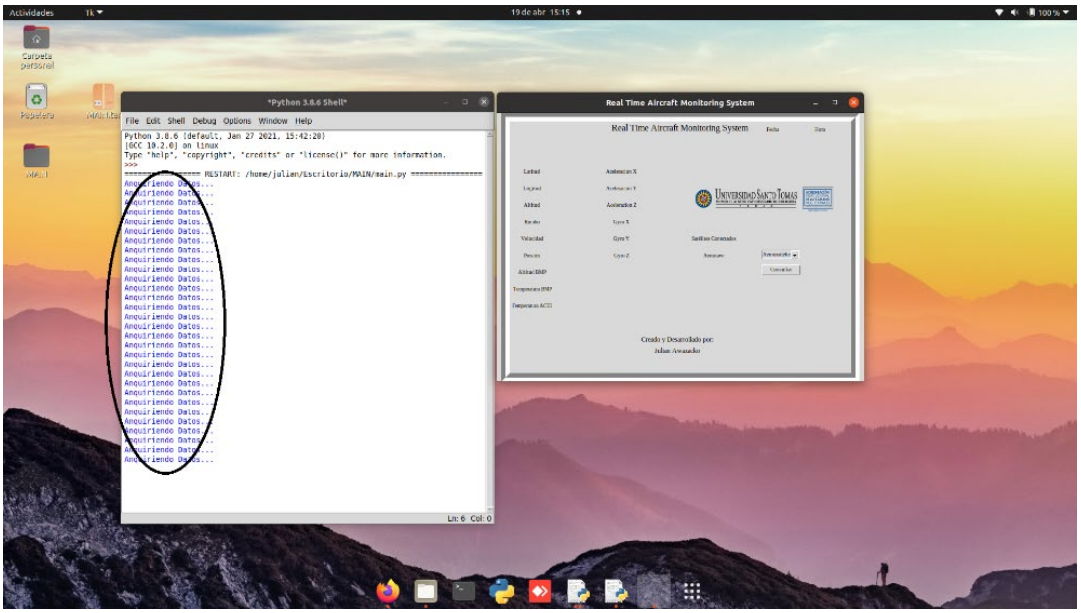


Figura 76. Comprobación programa iniciado correctamente.

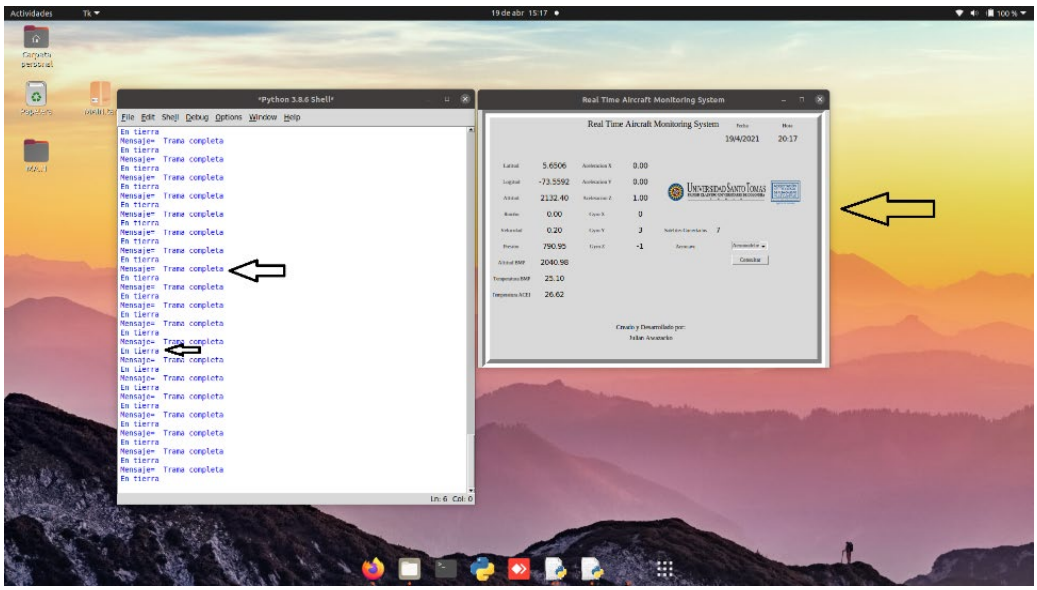


Figura 77. Recepción de Datos.

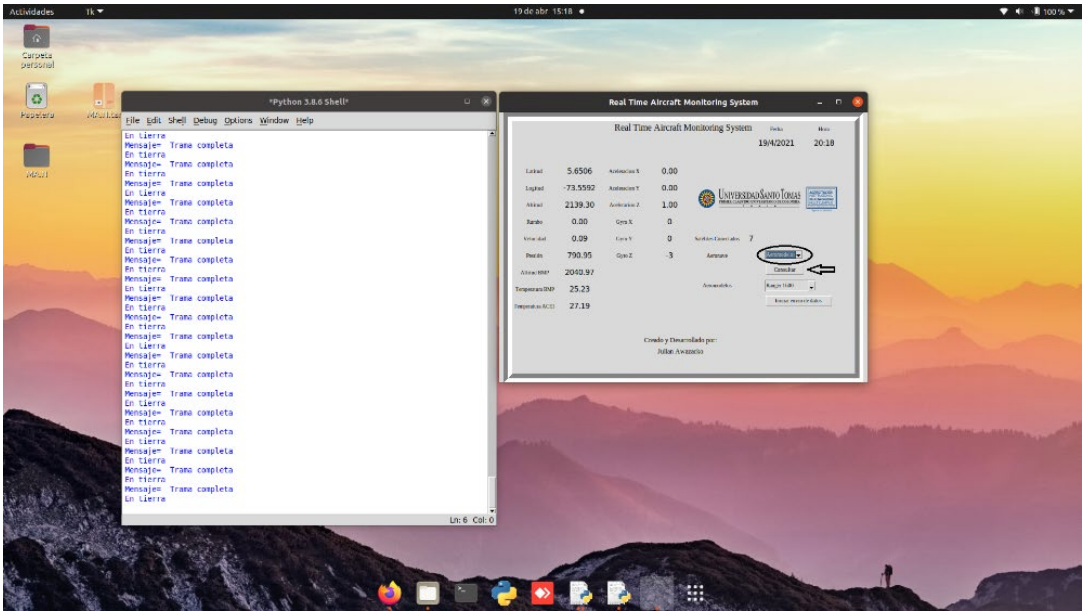
Después de pasado un tiempo se observa que los datos empiezan a llegar (Figura 6), el tiempo que se demora haciendo dicha adquisición es el tiempo que el módulo GPS tarda en hacer la conexión con los satélites, ya que como se menciona anteriormente el Arduino

no empieza a enviar datos hasta que el GPS tenga una conexión establecida correctamente.

Adicional a eso hay que observar que la trama de datos llegue completa y que su estado inicial sea “En tierra” ya que el avión no se encuentra en movimiento.

c. Selección del Bucket de destino

Ahora hay que seleccionar a que Bucket de la página web se quieren enviar los datos, ya que el programa permite enviar datos a diferentes buckets, dependiendo del dispositivo del que se estén adquiriendo los datos se puede seleccionar entre 2 aeromodelos, un drone o un bucket de pruebas.



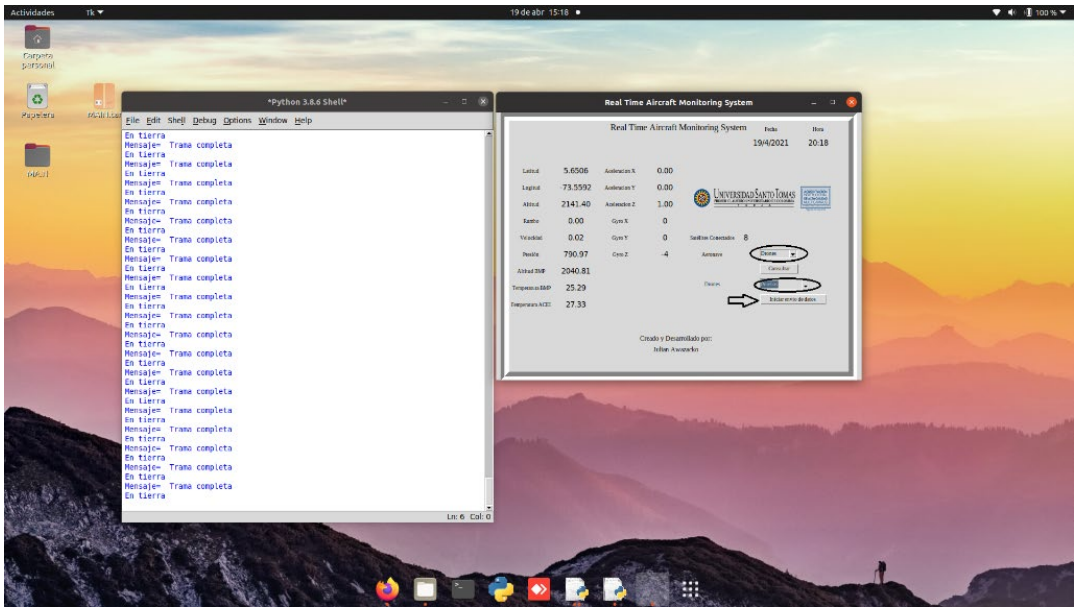


Figura 78. Selección Tipo de Aeronave.

Lo primero que se realiza es seleccionar el tipo de aeronave (Figura 7) a la que se le realizara el track de vuelo, para este caso se puede elegir entre Drone o Aeromodelo. Se seleccionará Drone ya que aquí se encuentra la opción del bucket de pruebas que es a donde se enviaran los datos.

d. Envío de datos a la página web

Una vez seleccionado el tipo de aeronave y el bucket de datos al que irán se presiona el botón de “Iniciar Envío de Datos” y se verifica en el Shell de Python que los datos se están enviando correctamente.

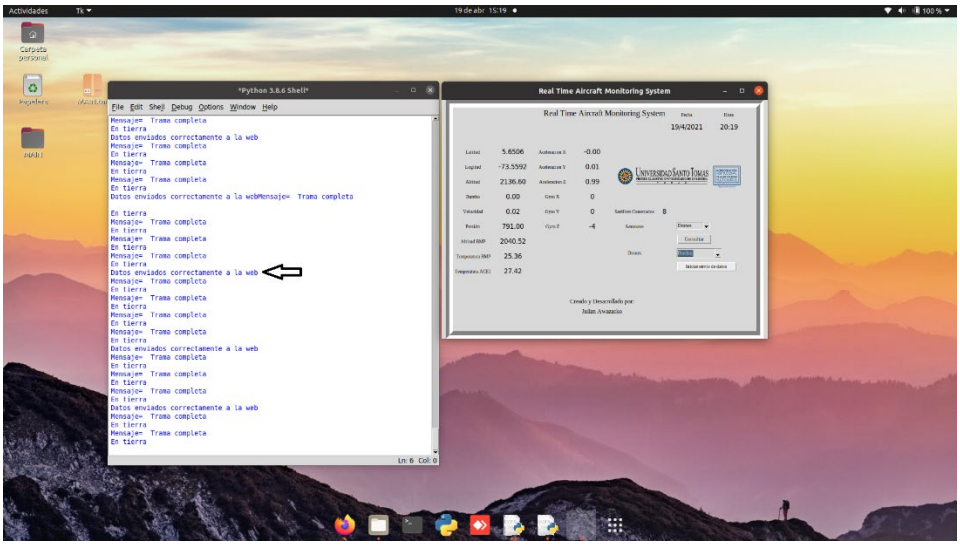


Figura 79. Envío de datos a la web.

Ahora hay que realizar la comprobación de que los datos están llegando al Bucket de destino seleccionado.

Para este caso hay que dirigirse a el perfil de initial State, iniciar sesión y seleccionar dicho bucket (Figura 9).

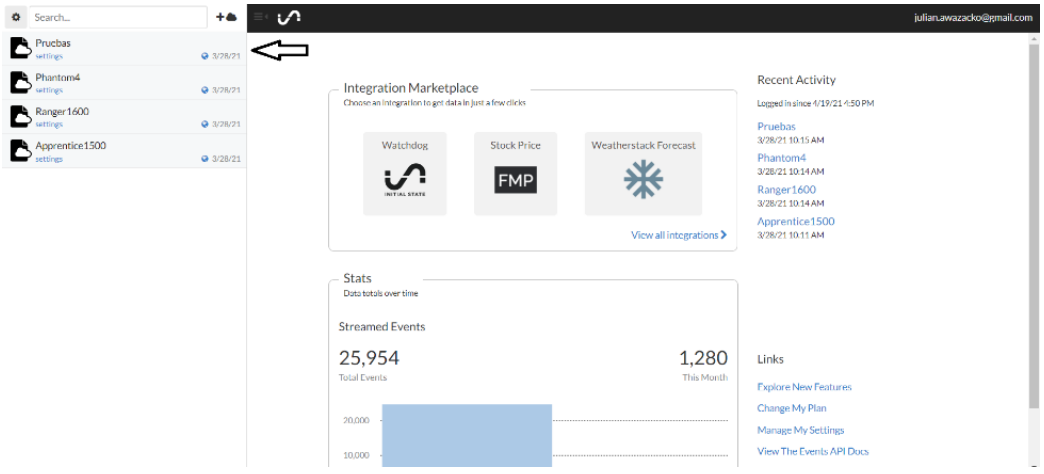


Figura 80. Bucket de pruebas.

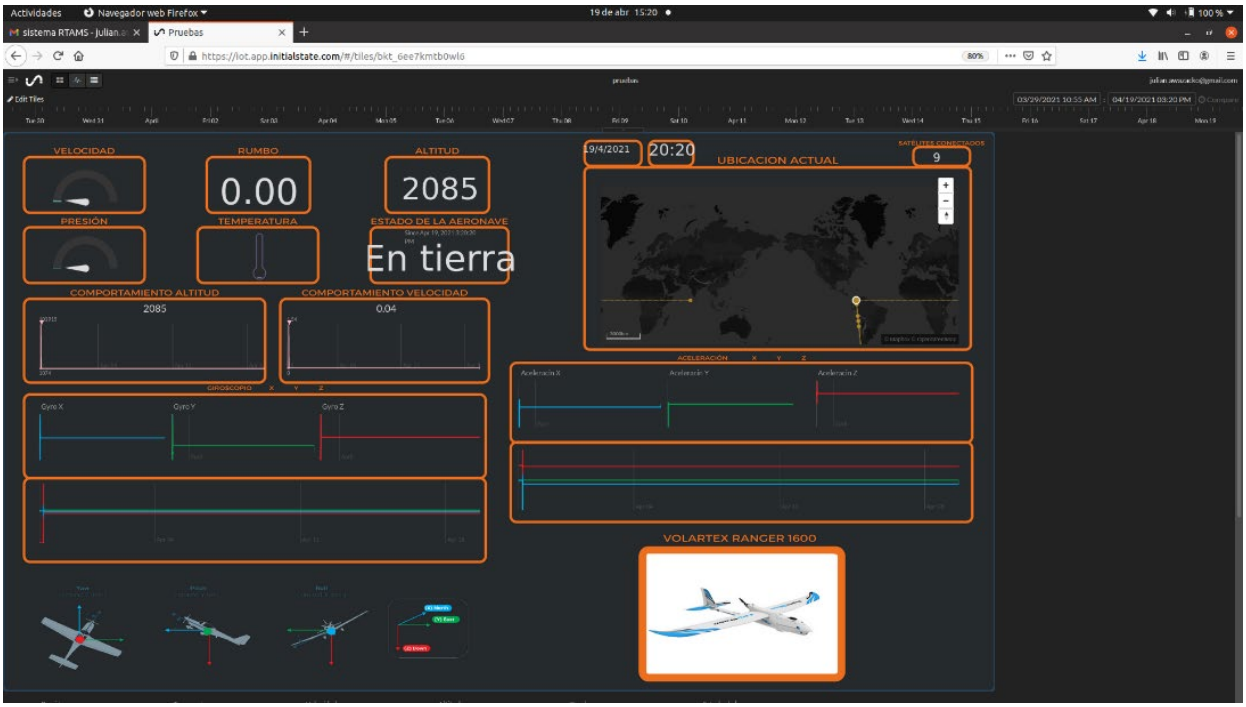


Figura 81. Recepción datos página web.

En la Figura 10 se puede observar que los datos están llegando correctamente y así se comprueba que el sistema en general está funcionando.

e. Finalización del track

Una vez se realiza la recepción de todos los datos se puede cerrar la ventana para finalizar en su totalidad todo el programa.

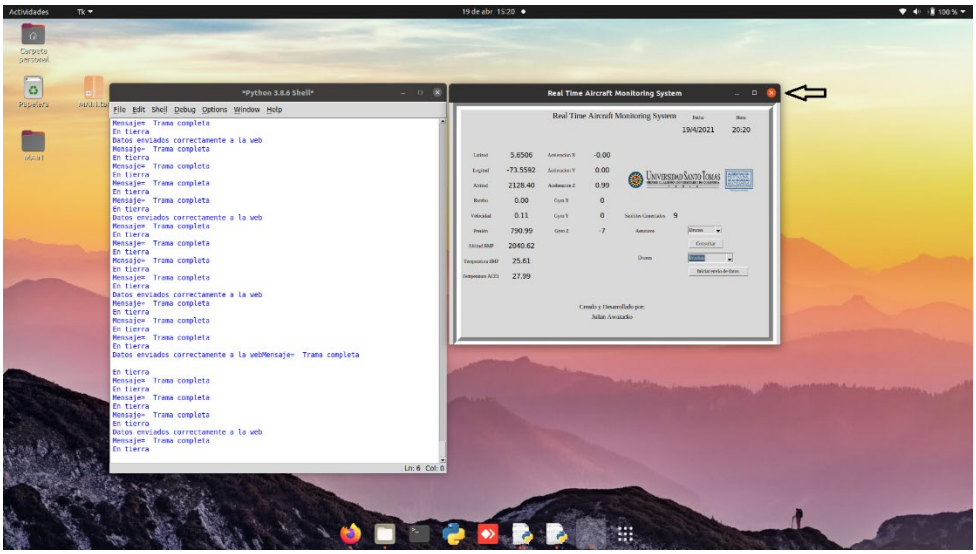


Figura 82. Cierre ventana principal.

f. Para tener en cuenta

No hay que preocuparse si mientras se hace la trama de datos sale un error como el que se muestra en la Figura 12, puesto que el mismo programa cicla hasta que salga del error, este es muy común cuando se llegan muchos datos y no se pueden separar.

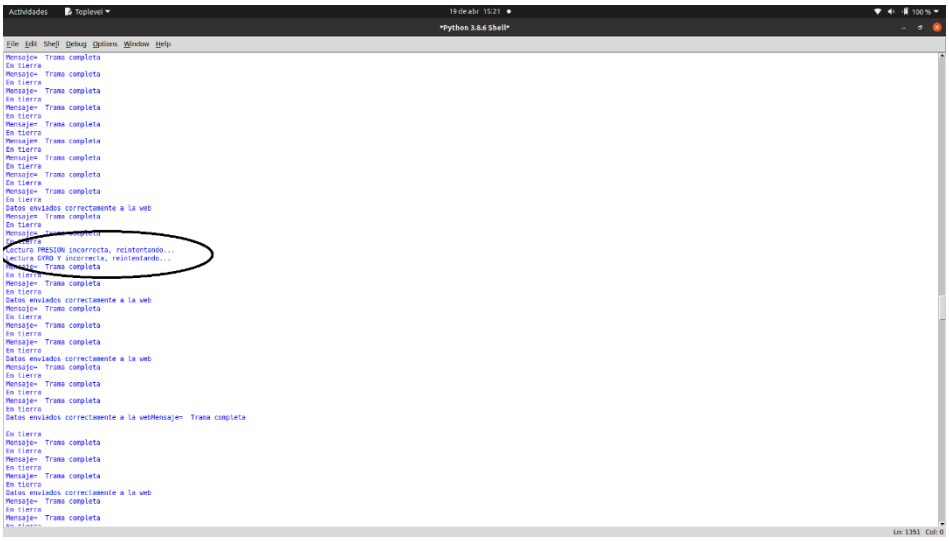


Figura 83. Comprobación errores

g. Añadir buckets al sistema

En caso de que se requiera añadir más buckets para almacenar datos, solo basta con añadir o modificar las líneas de código que se muestran a continuación y se encuentran en la función send_Data del programa mainFrame.py.

```
def send_Data(self):

    while self.empezarenvio:
        if self.Tipo.get() == self.Aeromodelo[0]:
            self.streamer = Streamer(bucket_name="Ranger 1600", bucket_key="3ECCS3R46QW7", access_key="ist_p3fGOFozZgQ66nfU2Z3crqpgZ-6_DH7h")

        if self.Tipo.get() == self.Aeromodelo[1]:
            self.streamer = Streamer(bucket_name="Apprentice1500", bucket_key="8ABNQMJB7QE6", access_key="ist_p3fGOFozZgQ66nfU2Z3crqpgZ-6_DH7h")

        if self.Tipo.get() == self.Drone[0]:
            self.streamer = Streamer(bucket_name="Phantom4", bucket_key="RR5XL37TRYVW", access_key="ist_p3fGOFozZgQ66nfU2Z3crqpgZ-6_DH7h")

        if self.Tipo.get() == self.Drone[1]:
            self.streamer = Streamer(bucket_name="Pruebas", bucket_key="8K8E8X6YX48T", access_key="ist_p3fGOFozZgQ66nfU2Z3crqpgZ-6_DH7h")
```

Figura 84. Editar Buckets

Los datos que hay que añadir los arroja la página de initial State.