
**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN**

# "ANÁLISIS Y DISEÑO DE UN SISTEMA DE CONTROL DE UNA PLATAFORMA AÉREA NO TRIPULADA MEDIANTE UNA APLICACIÓN MÓVIL"

**INFORME DE PROYECTO DE GRADUACIÓN**

**Previo a la obtención del Título de:**

**INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES**

**Presentado por**

**Ruth Sofía Duchi Rivera**

**Guayaquil – Ecuador**

**2014**

# AGRADECIMIENTO

Le agradezco la MSc Patricia Chávez por las invaluables sugerencias y aportes académicos brindados durante el desarrollo del presente trabajo. A mi padre, el Ing. Walter Duchi Silva y al Ing. Aníbal Gamboa por el apoyo incondicional brindado en el desarrollo de este proyecto de graduación.

Ruth Sofía Duchi Rivera

# DEDICATORIA

Dedico este proyecto de graduación a Dios por guiar siempre mi camino para salir adelante y a mi madre Mercedes Rivera Suarez por ser un pilar fundamental en mi vida y un apoyo incondicional en este escalón de mi vida profesional.

Ruth Sofía Duchi Rivera

# TRIBUNAL DE GRADUACIÓN

Prof. SARA RÍOS ORELLANA
SUBDECANA DE LA FIEC

Prof. IGNACIO MARÍN GARCÍA.
DIRECTOR DE PROYECTO DE GRADUACIÓN

Prof. DENNYS CORTEZ ALVAREZ
MIEMBRO PRINCIPAL DEL TRIBUNAL

# DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este informe, me corresponde exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL"

(Reglamento de Graduación de la ESPOL)

_____

Ruth Sofía Duchi Rivera

# RESUMEN

Hoy en día, la necesidad de obtener datos (imágenes, video, temperatura, humedad, cantidad de ruido,…) de lugares sobrevolados es bastante frecuente. Este texto explica claramente cómo se realizó el análisis, diseño e implementación de un sistema de control para una plataforma de vuelo no tripulada mediante una aplicación móvil; en el cual se utilizó el diseño modular, con la finalidad de permitir a nuevos desarrolladores acoplar algún módulo especializado que realice una o varias tareas dependiendo de las necesidades del usuario final. El sistema es independiente del suelo y fácilmente controlable a través de un dispositivo móvil. Con la finalidad de facilitar la comprensión del mismo para el lector, el sistema fue dividido en cuatro partes fundamentales: diseño de la góndola, diseño del canal de comunicación, diseño de los subsistemas de control, y diseño de la aplicación móvil.

Como primer punto tenemos el diseño de la góndola, lo cual se detalla en el capítulo tres. Este capítulo define a la góndola como el contenedor de todos

los elementos necesarios para controlar la navegación de la plataforma de vuelo y considera una cantidad limitada de espacio reservado para instalar el módulo de funcionalidad específica, al que se le llamará módulo secundario.

En segundo lugar tenemos el canal de comunicación, el cual consta de dos enlaces. El primero usando el protocolo *Bluetooth* y el segundo con el protocolo *ZigBee*. Estos enlaces se encuentran unidos por medio del dispositivo intermediario al que llamaremos radio base, el cual cumple las funciones de conmutador y traductor. Esta información es detallada en el capítulo cuatro.

En tercer lugar tenemos el diseño de cada uno de los sistemas de control, detallados en el capítulo cinco. En este capítulo se explica el funcionamiento del subsistema de control de direccionamiento, recuperación, velocidad, así como de los subsistemas de control visual de comunicación y de energía. Para terminar este tema, se hace énfasis en la integración de los subsistemas sobre una placa controladora de propósito generar como es el "*Arduino Mega*".

Finalmente, se detalla en el capítulo seis como fue desarrollada la aplicación móvil; la misma que funciona como una interfaz gráfica entre el usuario y el sistema, generando y enviando datos de control a través de la salida

*Bluetooth* del dispositivo móvil. La aplicación fue desarrollada para el sistema operativo Android OS, utilizando la herramienta de desarrollo "*App inventor*".

Adicionalmente, se realizaron las pruebas correspondientes con cada una de las cuatro partes del sistema. Con el canal de comunicación se realizaron pruebas para determinar el alcance máximo del canal con una confiabilidad de 99.7%. Sobre los subsistemas de control se realizaron pruebas que nos permitieron inferir de manera cualitativa la confiabilidad del sistema; sin embargo no se realizaron pruebas que permitan determinar el valor de fuerza que permite el desplazamiento, puesto que no se contó con el equipo necesario para realizar dicha prueba. Por último la prueba realizada a la aplicación fue una encuesta para determinar de manera cualitativa su facilidad de uso.

# ÍNDICE GENERAL

# ABREVIATURAS

AWG: American Wire Gauge.

BLDC: Brushless DC.

DAC: Dirección General de Aviación Civil.

ESC: Electronic speed Controller.

FEM: Fuerza Electromotriz

IDE: Integrated Development Environment

ISM: Industrial, Scientific and Medical

LED: Light-emitting Diode

OS: Operating System

OSI: Open System Interconnection

PCB: Printed Circuit Board

PNBV: Plan Nacional del Buen Vivir

PWM: Pulse-width Modulation

RF: Radio Frequency

RIM: Research In Motion Limited

SENATEL: Secretaría Nacional de Telecomunicaciones.

SUPERTEL: Superintendencia de Telecomunicaciones

UART: Universal Asynchronous Receiver-Transmitter

ITU: International Telecommunication Union

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# ÍNDICE DE ECUACIONES

# INTRODUCCIÓN

Las plataformas de vuelo no tripuladas nos permiten tener un sistema económico e independiente del suelo, sobre el cual instalar módulos o sensores para satisfacer necesidades específicas de un usuario. Para instalar cualquier tipo de sensor o módulo es necesario contar con el control de la plataforma de vuelo, el cual es el principal objetivo de este proyecto de graduación.

El sistema de control de direccionamiento de la plataforma de vuelo utiliza un dispositivo móvil, al cual se le debe instalar un aplicativo específico diseñado en este proyecto, para que funcione como el control remoto del sistema. Los datos generados por la aplicación móvil viajan a través del canal de comunicación de dos enlace, este canal utiliza *Bluetooth* en el primer tramo (dispositivo móvil – radio base) y *ZigBee* en el segundo tramo (radio base - góndola). En la góndola, por medio de un controlador los datos recibidos (arriba, abajo, derecha, izquierda, adelante, velocidad mínima, velocidad normal y velocidad máxima) son interpretados y ejecutados. El controlador

instalado en la góndola también es capaz de reconocer un dato específico para activar el módulo secundario y ceder el uso del canal a este dispositivo controlador externo. Adicionalmente, el controlador es capaz de identificar cuando el canal de comunicación tiene fallas y activa el subsistema de recuperación, que permite descender la plataforma de vuelo mientras se mantiene girando hacia la derecha. Para concluir, la implementación del proyecto consiste en colocar todos los elementos dentro de la góndola, considerando en su diseño el espacio para el modulo secundario o también llamado dispositivo de funciones especializadas.

# CAPÍTULO 1

# ANTECEDENTES Y JUSTIFICACIÓN

Antes de iniciar con el análisis, diseño e implementación de este proyecto es necesario definir los objetivos, la metodología que se utilizó en el desarrollo del mismo, y además las limitaciones que se tuvo en el proceso de alcanzar dichos objetivos. Por último es necesario conocer qué problema se resuelve al implementar este proyecto y en qué situación económica y social se desarrolló.

## 1.1 Antecedentes.

Las soluciones actuales, que permiten tener un sistema independiente de tierra con la finalidad de obtener datos de zonas sobrevoladas, utilizan como medio de transporte aeronaves, cuyo alquiler tiene un alto costo para el usuario final. Como alternativa viable planteamos la creación de plataformas aéreas no tripuladas con un diseño basado en dirigibles, estas plataformas permiten instalar los módulos necesarios

para cada tarea específica. En nuestro país, la disponibilidad comercial de estos vehículos es limitada, al no existir empresas dedicadas a dicha actividad económica tampoco existe una percepción social de este medio como una solución viable.

## 1.2 Descripción del problema.

Las plataformas aéreas no tripuladas, permiten realizar tareas como: obtener datos geográficos y medioambientales de la zona sobrevolada, utilizando módulos o sensores integrados a la plataforma en base a las necesidades del usuario final (módulo fotográfico, módulo de video, sensores de temperatura o humedad…). Las soluciones importadas disponibles tienen un alto costo, debido entre otros, a la falta de una industria productiva nacional que pudiera competir en la producción de las mismas, y debido a los mercados en los cuales se han enfocado la producción y distribución de dichas plataformas. Los sistemas de control resultan ser sumamente complicados, requiriendo personal especialmente preparado para su manejo y control, siendo esto una limitante importante en el uso de dichas plataformas.

## 1.3 Justificación.

Contar con un sistema de vuelo no tripulado resulta útil, puesto que podemos colocar en el mismo los diferentes módulos y sensores

(monitoreo, temperatura, ruido, humedad…); los cuales nos permiten analizar los daños producidos por una catástrofe climática, visualizar plantaciones o simplemente obtener una toma fotográfica con vista superior para realizar trabajos en el área de producción audiovisual. Para realizar las actividades antes mencionadas (obtención de imagen, temperatura, humedad, cantidad de ruido…) se recurre habitualmente a alquilar una avioneta o helicóptero, con un alto costo debido a los gastos asociados como son: la gasolina que consume el mismo, los servicios prestados por el piloto, los permisos de despegue y aterrizaje y el módulo o sensor necesario según sea el caso; estos gastos pueden ser disminuidos utilizando una plataforma de vuelo no tripulada (dirigible), la cual utiliza menos recursos económicos (recarga de helio, el globo aerostático, un dispositivo móvil para instalar la aplicación y las baterías) los mismos que se van a realizar al momento de la adquisición del dirigible.

## 1.4 Objetivos generales.

Analizar y diseñar un sistema controlador de una plataforma de vuelo no tripulada “dirigible” desde una aplicación móvil.

## 1.5 Objetivos específicos.

Analizar los protocolos y módulos que más le convengan al sistema para una correcta comunicación entre la plataforma de vuelo y la aplicación móvil.

Diseñar un subsistema de traducción que permita el entendimiento entre los dos protocolos seleccionados.

Diseñar la góndola de la plataforma de vuelo con los subsistemas de control de movimiento y sustentabilidad, de recuperación, de energía y de transmisión.

Diseñar los sub-sistemas de control del dirigible, esto incluye control de direccionamiento, control de velocidad, de recuperación y de energía.

Diseñar una aplicación móvil que permita el control de una plataforma de vuelo (dirigible).

## 1.6 Metodología.

En el desarrollo del sistema se identificaron cuatro etapas fundamentales:

Primera Etapa.- Se analizaron los protocolos y módulos de comunicación necesarios para satisfacer las características del sistema (ancho de banda, alcance…) y se seleccionó *ZigBee* para la transmisión de señales de control y *Bluetooth* para enviar los datos generados en el dispositivo móvil como se muestra en la Figura 1.1.

Segunda Etapa.- Se diseñó el módulo de traducción, el cual es un equipo intermediario en la comunicación entre el dispositivo móvil y la plataforma de vuelo, y tiene la finalidad de incrementar la potencia de la señal y al mismo tiempo permite que los protocolos seleccionados (*ZigBee* y *Bluetooth*) comprendan los datos.

Tercera Etapa.- Se tuvo que desarrollar la aplicación móvil, la cual fue realizada en el IDE "*App Inventor*" para poder ser instalada en un dispositivo móvil con sistema operativo Android OS, esta aplicación nos permitió generar y enviar las señales de control para direccionar al dirigible.

Cuarta Etapa.- Se realizó el diseño de una góndola en la cual se colocó los subsistemas de control de direccionamiento, de velocidad y de recuperación, el módulo de transmisión para la comunicación con la aplicación móvil, el subsistema de energía necesario para su funcionamiento y además los motores y las hélices respectivas.

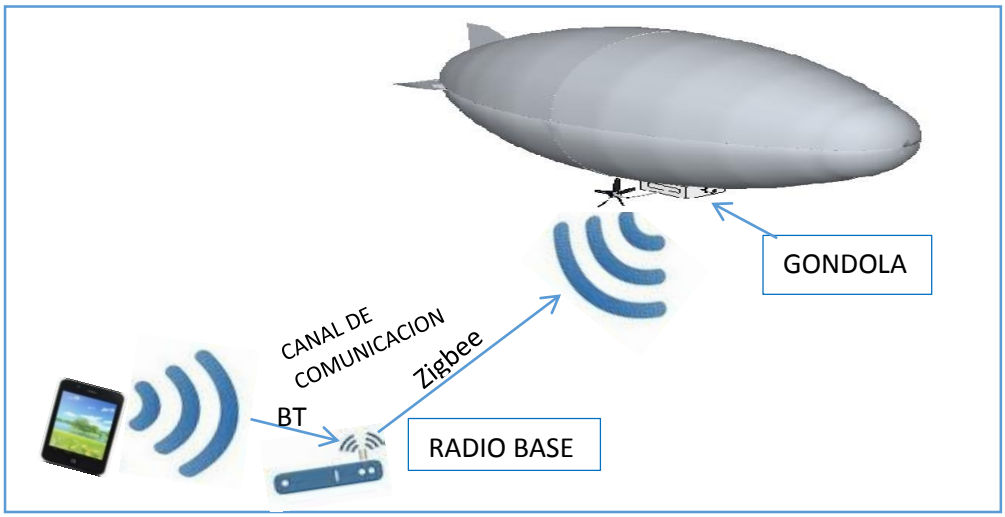



Figura 1.1 : Diagrama general del sistema.

## 1.7 Alcances y Limitaciones.

Se logró establecer correctamente la comunicación entre la aplicación móvil y la plataforma de vuelo, se pudo controlar la navegación de la plataforma de vuelo y además que el subsistema de recuperación permita su descenso; también es importante aclarar que este proyecto se centró en el desarrollo del sistema de control de la plataforma de vuelo desde una aplicación móvil, mas no en el diseño de la plataforma de vuelo, ni en los módulos o sensores que se pudieran colocar en el mismo para realizar alguna tarea. A pesar de los logros alcanzados se tuvo ciertas limitantes económicas y de acceso a la información, la más importante fue la comprar del dirigible físico para realizar las pruebas, pero también se tuvo limitaciones en el presupuesto para la adquisición

de herramientas de desarrollo, sistemas operativos y sistemas de simulación.

# CAPÍTULO 2

# MARCO TEÓRICO

Una descripción general del proyecto fue dada en el capítulo anterior, en la que se habló de las funcionalidades, las aplicaciones y las ventajas que posee el mismo con respecto a otras soluciones. Antes de profundizar en el diseño del mismo; es necesario comprender que el funcionamiento del control de la plataforma de vuelo no tripulada, no se basa únicamente en conceptos referentes a telecomunicaciones, tales como protocolos inalámbricos y módulos de comunicación; sino también en conceptos eléctricos y mecánicos en los que nos basamos para controlar el direccionamiento de la misma. En este capítulo se explica también las partes claves de la operación del dispositivo controlador, el cual permitió garantizar el correcto funcionamiento del sistema. Y finalmente, podremos conocer acerca de los requerimientos legales a los que nos regimos para implementar este proyecto, tanto de la ley especial de telecomunicaciones (regulado por la Superintendencia de Telecomunicaciones, SUPERTEL) como del código

aeronáutico (regulado por la Dirección de Aviación Civil, DAC) que se encuentren vigentes en nuestro país al momento de la publicación del presente escrito.

## 2.1 Conceptos básicos para el entendimiento del proyecto.

Como primer punto de estudio, tenemos los conceptos empleados en el análisis realizado para dimensionar la plataforma de vuelo, en este caso un dirigible. Este análisis involucra leyes físicas de mecánica de fluidos, aerodinámica y equilibrio de cuerpos libres, además de diferentes técnicas de aeromodelismo que puedan ser aplicadas a aeronaves no tripuladas. Considerándose aeromodelismo a la disciplina que utilizando técnicas diversas se ocupa del diseño, construcción y vuelo de objeto más pesados que el aire y de tamaño reducido incapaz de llevar un ser humano [**1**]. Se sobreentenderá que conceptos como: densidad, presión, peso, gravedad están claramente comprendidos.

El Principio de Arquímedes establece que: “Si un cuerpo con densidad $\rho_c$ está parcialmente o totalmente sumergido en un fluido con densidad $\rho_f$, éste ejerce una fuerza B hacia arriba sobre el cuerpo; que es igual al peso del fluido desplazado por mismo” [**2**], para que un cuerpo flote esta fuerza denominada fuerza de flotación o boyante debe ser mayor que el

peso del objeto, por ende la densidad del fluido debe ser mayor que la del objeto.

La segunda ley de Newton es claves en el análisis de la flotación de cuerpos, esta dice que: “La suma vectorial de las fuerzas que actúan sobre un cuerpo son iguales a su masa por la aceleración del mismo, si el cuerpo está en equilibrio esta suma es igual a cero” [**2**].

También es importante enunciar, cómo evaluar la estabilidad de un cuerpo flotante, así como un globo con helio flota en el aire, los cuerpos pueden ser: estables, cuando con cualquier perturbación pequeña se genera una fuerza de restitución que la regrese a su posición inicial; neutralmente estable, porque si alguien lo mueve este quedará en su nueva ubicación, no tiene que seguirse moviendo ni regresar a la posición inicial; o inestables porque con cualquier perturbación este continuará moviéndose.

Con respecto a la estabilidad rotacional y estática. Yunus nos dice que: “Para un cuerpo sumergido o flotante en equilibrio estático, el peso y la fuerza de flotación que actúan sobre él se equilibrarán entre sí, por ende estos cuerpos son estables en la dirección vertical” [**2**]. Y también que: “La estabilidad rotacional de un cuerpo sumergido depende de las

ubicaciones relativas del centro de gravedad G del cuerpo y el centro de flotación B, el cual es el centro de del volumen desplazado; si el punto G está directamente debajo del punto B el cuerpo tiene estabilidad rotacional con lo cual una perturbación rotacional produce un momento de restitución que lo regresa a su posición inicial" [**2**].

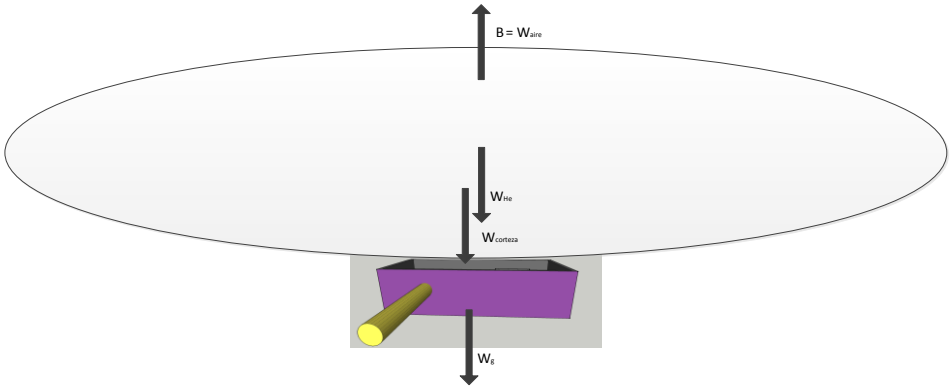



Figura 2.1: Fuerzas que actuan sobre la plataforma de vuelo.

Las fuerzas mostradas en la Figura 2.1 son las que actúan en el análisis del equilibrio estático para el cálculo de las dimensiones del dirigible, generando la ecuación:

$$B - W_{Gondola} - W_{He} - W_{Dirigible} = 0 \qquad (2.1)$$

Dónde:

B: Fuerza boyante

WGondola: Peso de la góndola.

$W_{He}$: Peso del gas con el que se llena el dirigible.

$W_{Dirigible}$: Peso de la corteza del dirigible.

De esta ecuación se obtiene que la cantidad de helio necesaria para levantar la góndola es 1.35 m3, el cálculo detallado de la obtención de este valor se encuentra en el Anexo A.

El arrastre y la sustentación son medidas muy importantes al momento de dimensionar el dirigible; el arrastre es la fuerza que un fluido ejerce sobre un cuerpo en la dirección del flujo y la sustentación es la fuerza resultante generada por las componentes de presión y fuerza de corte normales al flujo, las cuales tiende a mover a un cuerpo en dicha dirección. Estas fuerzas pueden ser medidas utilizando parámetros como el coeficiente de arrastre CD y el coeficiente de sustentación CL, los cuales se expresan matemáticamente en las ecuaciones (2.2) y (2.3). Donde A es el área frontal del cuerpo, $\frac{1}{2}\rho V^2$es la presión dinámica y FD y FL son la fuerzas debidas al arrastre y sustentación respectivamente. [**2**]

$$\mathbf{C_D = \frac{F_D}{\frac{1}{2}\rho V^2 A}} \qquad (2.2)$$

$$\mathbf{C_L = \frac{F_L}{\frac{1}{2}\rho V^2 A}} \qquad (2.3)$$

El segundo punto es explicar el funcionamiento de los protocolos utilizados en el canal de comunicación: *Bluetooth* y *ZigBee*. En la

sección 4.1 se detalla por qué se seleccionaron estos protocolos y como se configuran los dispositivos utilizados.

El protocolo *Bluetooth* es una especificación industrial para redes inalámbricas de área personal, que permite la transmisión de datos a corta distancia. *Bluetooth* funciona sobre tres capas. La capa “radio frecuencia” está conformada por el transceptor físico y sus componentes asociados, los cuales permiten la transmisión de datos sobre la banda ISM a 2.4 GHz. La capa “banda base” es la encargada del control de la comunicación, permite crear dos tipos de redes: *piconet* o *scatternet*. Una *piconet* está formada por un maestro y varios esclavos que pueden ser activos o en espera, mientras que una *scatternet* está formada por dos o más *piconet* entrelazadas por uno o más dispositivos que trabaja como maestro para una *piconet* y como esclavo para otra *piconet*; tal como se muestra en la Figura 2.2. El módulo *Bluetooth* utilizado fue el HC06, el cual tiene una sensibilidad que puede alcanzar los -80 dBm, posee una antena digital inalámbrica construida en 2.4 GHz y tiene un alcance teórico máximo de 1 m sin línea de vista [**3**].

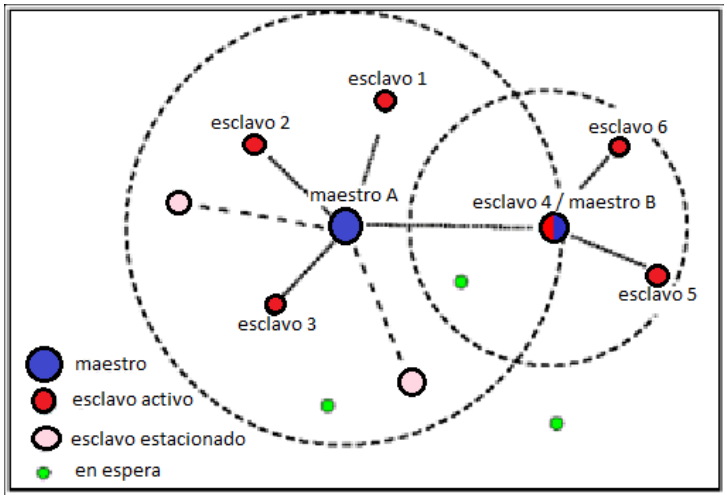



Figura 2.2: Tipos de topología de red permitidas con *Bluetooth*. [4]

Por otro lado, *ZigBee* es un protocolo especializado en la transmisión de señales de control que surgió de una mejora al protocolo 802.15.4; el cual permite tener un dispositivo de bajo consumo, pero que transmite a bajas velocidades. Este estándar define dos tipos de topologías de red: estrella y en pares. En la topología estrella, todos los dispositivos se asocian a un único dispositivo coordinador. Para la topología en pares, se asocia dispositivos de dos en dos hasta interconectar todos los dispositivos de red con la redundancia deseada. Además, este protocolo trabaja sobre dos de las capas del modelo de referencia OSI: Red y Aplicación. La Capa de Red permite múltiples saltos utilizando enrutamiento por redes tipo malla y asocia a la red cifrado y autentificación. La Capa de Aplicación se encarga de iniciar o responder pedidos de enlace y establece una relación segura entre dispositivos; además permite a los nodos exponer valores individuales,

como sensores, conmutadores o actuadores. Los módulos *Xbee Pro* trabajan en la banda de 2.4 GHz con el protocolo de comunicación *ZigBee*, tienen un alcance máximo teórico de 90 metros si línea de vista [**5**].

Finalmente, es muy importante entender el funcionamiento de los motores BLDC (Motores sin escobillas) y también tener una idea clara de la técnica que se utiliza para controlar este tipo de motores; puesto que estos son utilizados en la implementación de los subsistemas de control. La principal característica de los motores BLDC es que no emplean escobillas en la conmutación para la transferencia de energía, debido a que estas producen rozamiento, disminuyen el rendimiento, generan calor, entre otros. Estos motores fueron seleccionados basados en que tienen las siguientes ventajas con respecto a los motores DC convencionales: mejor relación velocidad - par motor, mayor respuesta dinámica, mayor eficiencia, mayor vida útil; en nuestro caso se utilizó motores BLDC de giro externo, porque al tener sus imanes instalados permanentemente en la carcasa externa del motor desarrollan su torque máximo a velocidades más bajas que los BLDC de tipo giro interno, de tal manera que no necesitan reducción, y se pueden acoplar directamente a una hélice. Este tipo de motores puede ser controlado utilizando Conmutación Trapezoidal, en la cual se excita

simultáneamente cada uno de los seis posibles pares dejando siempre una línea desconectada; la velocidad con la que se conmuta dicha excitación a un nuevo par es la que determina la velocidad de giro del motor, puesto que este tipo de controlador de velocidad se basa en que el tiempo entre una conmutación y el cruce por cero es igual al tiempo entre el cruce por cero y la siguiente conmutación. Este método tiene como ventajas su sencillez y facilidad de implementación, por lo cual es más usado en motores pequeños [**6**].

## 2.2 Marco regulatorio de las telecomunicaciones.

La organización mundial más antigua para la regulación de las telecomunicaciones UIT considera a las telecomunicaciones como toda transmisión, emisión o recepción de signos, señales, escritos, imágenes, sonidos o informaciones de cualquier naturaleza por hilo, radioelectricidad, medios ópticos u otros sistemas electromagnéticos; por este motivo este proyecto es considerado un proyecto de telecomunicaciones y por ende es necesario especificar el marco legal que lo regula.

El plan Nacional del Buen Vivir (PNBV) es una carta constitucional que reúne principios, derechos y orientaciones de un nuevo pacto social; entre los derechos del buen vivir con respecto al uso del espectro están

los siguientes: el acceso en igualdad de condiciones al uso de frecuencias del espectro radioeléctrico y a las bandas libres para la explotación de redes inalámbricas; también en este plan se ratifica que el espectro radioeléctrico es un recurso natural sobre el cual el Estado tiene facultades exclusivas y al ser considerado como un sector estratégico se reserva el derecho de administrar, regular, gestionar y controlar, de conformidad con los principios de sostenimiento ambiental, precaución, prevención y eficiencia. Además de esto, se reconoce en la Constitución que el espectro tiene decisiva influencia económica, social, política o ambiental y que se orienta al pleno desarrollo de los derechos e interés social.

La ley especial de telecomunicaciones reformada estipula lo siguiente: con respecto al uso de frecuencias *"El uso del espectro radioeléctrico es necesario para la provisión de los servicios de telecomunicaciones y deberá, en todos los casos, ajustarse al Plan Nacional de Frecuencias"* [**7**], este plan ubica al Ecuador en la región dos y especifica la distribución del espectro de frecuencias para cada tipo de comunicación; con respecto al título habilitante *"El título habilitante para la prestación y explotación de los servicios de telecomunicaciones que requieran de espectro deberá obtenerse obligatoriamente, en forma simultánea, con la concesión del uso del espectro* [**7**]*"*, el cual podrá ser

un permiso o concesión que será entregado por el SENATEL que es el ente regulador; con respecto al uso *“El uso del espectro radioeléctrico para telecomunicaciones podrá consistir en uso privativo, uso compartido, uso experimental, o uso reservado y su asignación, siempre requerirá de una concesión.”* [**7**], por todo lo explicado anteriormente es necesario respetar el plan de frecuencias y tener un título habilitante considerando que el uso del mismo es para fines experimentales y que además se está utilizando el espectro de frecuencias definido en Plan Nacional de Telecomunicaciones como bandas de frecuencias libres, las cuales son compartidas y abiertas pero que requieren un título habilitante denominado licencia simplificada el cual se obtiene mediante un registro con SENATEL, quien es el órgano regulador de las telecomunicaciones en el Ecuador; dado que los módulos de transmisión que se usaron trabajan en la banda de 2.4 GHz se requiere también de una negociación con las Fuerzas Armadas del Ecuador, puesto que antes de que se declaren esta bandas como bandas de uso libre estaban consideradas como reservadas para la fuerza armadas. Cabe recalcar que la necesidad de un título habilitante varía dependiendo del país, puede no ser necesario.

## 2.3 Administración del espacio aéreo.

La administración del espacio ecuatoriano es de gran importancia para el desarrollo de este proyecto, puesto que el código aeronáutico (leyes dispuestas para la administración del espacio aéreo) estipula en el capítulo 2 art. 7 que *"todo piloto al mando de una aeronave que vuele sobre el territorio ecuatoriano está obligado a tener conocimiento de la leyes y reglamentos que rigen el tránsito aéreo en el Ecuador y a cumplirlos"* [**8**] Sabiendo que en el capítulo 1 art. 3 se establece que "El Ecuador tiene y ejerce soberanía plena y exclusiva sobre el espacio aéreo que cubre su territorio y aguas jurisdiccionales" [**8**].

Dado que es obligatorio tener conocimiento de las leyes que rigen el tránsito aéreo dentro del Ecuador vamos a tratar sobre los principales reglamento y leyes existentes en el código aeronáutico que recaen sobre nuestro proyecto.

En la ley de tránsito para aeronaves Titulo 1 capítulo 2 artículos del 4 al 22 donde se establece que:

Art. 4.-*"Es libre el tránsito de aeronaves privadas sobre el territorio nacional siempre que se observen las disposiciones contenidas en este Código, así como las leyes y reglamentos pertinentes."* [**8**]*;*

Art. 5.- *"Nadie puede en razón de un derecho de propiedad oponerse al sobrevuelo de una aeronave. Si la aeronave en vuelo causare daños a los bienes de terceros en la superficie, éstos tendrán derecho a la indemnización correspondiente, de conformidad con lo que se establece en este Código."* [**8**]**;**

Art. 15.- *"De ninguna aeronave podrán arrojarse objetos que causen daño en la superficie salvo en caso de peligro grave."* [**8**]**;**

Art. 16 .- *"Ninguna aeronave podrá transportar, salvo permiso de la autoridad competente, explosivos, municiones, armas de fuego, material bélico o equipos destinados al levantamiento aerofotogramétrico, elementos radiactivos u otros objetos y sustancias que sean expresamente determinados."* [**8**]**;**

Y en el art. 18.- *"Se prohíbe a cualquier aeronave que vuele sobre el territorio ecuatoriano tomar fotografías de cualquier área o zona que haya sido declarada prohibida o restringida por la autoridad competente."* [**8**]**.**

Basándonos en este código, determinamos que: no existe reglamento que exija obtener un título habilitante para poder sobrevolar un lugar, ni tampoco se requiere permiso del dueño de la propiedad para sobrevolarla, pero en caso de causar daños o perjurios a terceros estamos en la obligación de someternos a lo que dice ese código y

entregar la indemnización correspondiente. Obviamente es penado por la ley el trasporte de todo tipo de sustancias psicotrópicas, estupefacientes o cualquier tipo de explosivos.

# CAPÍTULO 3

# GÓNDOLA

El diseño de la góndola consiste en analizar y seleccionar los materiales a utilizar para su construcción, la distribución de los componentes dentro la misma, la forma más adecuada que debe tener considerando que va a navegar en tres dimensiones, entre otros; cada análisis fue realizado basándonos en los conceptos recordados en el capítulo anterior. Profundizando en lo anteriormente expuesto, la góndola o cabina principal sigue la estructura del diseño modular, de tal manera que permite organizar los elementos utilizados en el control de la plataforma de vuelo y reservar espacio dentro de la misma para elementos adicionales; con la finalidad de incluir nuevas funcionalidades al sistema dependiendo de las necesidades del usuario final; es decir, tiene el espacio necesario para instalar dos módulos capaces de realizar tareas independientes compartiendo el mismo canal de comunicación. El módulo que se diseña en los capítulos posteriores es el módulo de control para la navegación de la plataforma de vuelo, al

mismo que llamaremos “módulo 1” y está formado por el controlador, el dispositivo de radio frecuencia, los motores, el eje, la batería y otros. El módulo adicional, al que llamaremos “módulo 2”, debe ser capaz de adaptarse a las especificaciones detalladas en este texto, tanto en dimensiones como en peso. Se recalca también, que la góndola, no solo es un contenedor de elementos, sino también es un protector de factores medio ambientales como son polvo, humedad y vientos.

## 3.1 Estudio y selección de Materiales.

En la selección del material a utilizar en la fabricación de la góndola se consideraron los siguientes temas como fundamentales: El peso de la carcasa de la góndola debe ser el mínimo posible, con la finalidad de minimizar el tamaño del dirigible según se observa en la ecuación (2.1); el material de preferencia debe ser de bajo costo y estar disponible a la venta dentro del país, para evitar retrasos en su construcción; el material debe poderse cortar, pegar y reafirmar con facilidad, puesto que la góndola será elaborada de manera artesanal. En base a esto se decido utilizar madera de balsa revestida de pintura impermeabilizante para incrementar la resistencia mecánica del material y hacerlo resistente a la humedad.

La madera balsa es obtenida del árbol de balso y pertenece a la familia de las bombacáceas. Es una madera tropical con características óptimas por la facilidad con que se la puede trabajar, puesto que puede ser fácilmente cortada y cepillada mediante herramientas cortantes delgadas y agudas. Posee una resistencia mecánica relativamente elevada en relación con su peso liviano. La Densidad cuando está seca es de 0.16 ($g/cm^3$) [**9**].

## 3.2 Diseño de la góndola.

Previo al diseño de la góndola, es necesario determinar qué elementos se utilizarán en la implementación del sistema de navegación para la plataforma de vuelo no tripulada. Estos son el controlador del sistema de navegación, el módulo de RF para transmitir a mayor distancia, un eje metálico de aluminio con dos motores, los cuales serán colocados a cada lado sobre una base en los extremos del eje, un servo motor para rotar al eje, acoplado por medio de engranajes, dos controladores de velocidad electrónicos para los motores derecho e izquierdo y una batería de 11.1 voltios para alimentar el sistema; adicional a estos componentes se tendrá el módulo especializado con su respectivo controlador, los cuales le dan funcionalidades adicionales al sistema de navegación, estas funcionalidades están limitadas al espacio y peso que se deja libre.

En el diseño de la góndola el primer tema a discutir es la forma de la góndola. Para ello es necesario revisar los principios aerodinámicos expuestos en el Capítulo 2.1, donde se estipula que para objetos pequeños que se mueven a bajas velocidades, la resistencia al viento puede considerarse despreciable. Sin embargo, se realizó el diseño utilizando conceptos de aeromodelismo, por lo que se consideró minimizar la magnitud de las fuerzas contrarias al movimiento, una de ellas es la fuerza de arrastre que siente la góndola debido al viento; el coeficiente de arrastre es una medida proporcional a la fuerza de arrastre y nos indica que a medida que reducimos el área transversal con la que choca el viento reducimos la magnitud de esta fuerza. Adicionalmente, la forma de la góndola también fue seleccionada para satisfacer otros criterios, como: facilidad en el ensamblaje, facilidad para colocar amarras y sujetadores, desperdiciar la menor cantidad de espacio posible, entre otros. Para cumplir estos criterios se seleccionó una forma cubica con bordes redondeados, puesto que los elementos instalados en la góndola serán considerados como bloques cúbicos, se desperdiciará menos espacio y al mismo tiempo los sujetadores serán más fácil de colocar.

Una vez fijada una forma simétrica para la góndola es necesario definir las dimensiones y pesos máximos que podrán tener cada uno de los elementos, estos datos se detallan en la Tabla 3.I; adicional a estos componentes se deben tener dos puntos de distribución de energía para 11.1 V y para 5 V.

Tabla 3.I: tabla de elementos contenidos dentro de la góndola

| Elemento | Dimensiones máx. | Peso máx. (g) |
|---|---|---|
| **Controlador del sistema de navegación** | 6 × 2 × 10 cm | 55 |
| **Dispositivo de RF** | 1 × 3 × 4 cm | 10 |
| **servo motor** | 2 × 4.5 × 6 cm | 45 |
| **Controlador de velocidad x2** | 1 × 3 × 6 cm | 30 × 2 |
| **Batería** | 5 × 14 × 2 cm | 265 |
| **Eje** | 44 cm × 0.8 cm | 210 |
| **Dispositivo especializado** | 6 × 11 × 8 cm | 150 |
| **Controlador de dispositivo especializado** | 6 × 2 × 10 cm | 55 |
| **Motores para direccionamiento x2** | 3.5 cm × 3 cm | 50 × 2 |
| **Distribuidor de 11.1 V** | 4 × 3 × 4 cm | 15 |
| **Distribuidor de 5 V** | 2.5 × 2.5 × 2 cm | 20 |

El siguiente punto es la distribución de los componentes, lo cual se realizó según se muestra en la Figura 3.1, el significado de cada una de las etiquetas es mostrado en la Tabla 3.I. En la distribución se consideró que:

- ✓ La antena de módulo de RF (12) debe estar lo más alejado posible de la batería de 11.1 V y de los cables por los que pasan corrientes superiores a 200 [mA].

- ✓ El engranaje del servo motor debe estar a una distancia precisa en el eje (20) para que el servo pueda ser sujetado en la pared lateral de la góndola.
- ✓ Los ESC’s deben estar cerca del eje y del punto de distribución de 11.1 V para realizar buenas prácticas de diseño electrónico, al evitar exceso de cableado.
- ✓ Todos los cables, puntos de distribución y elementos conectados al positivo o negativo de la batería deben estar adecuadamente señalizados, utilizando cable de colores rojo y negro respectivamente.
- ✓ En general, el calibre de los cables debe ser de 16 AWG, de tipo flexible y con protección para evitar crear campos magnéticos alrededor de los equipos de radio frecuencia.
- ✓ Los elementos deben estar fijados con sujetadores para evitar las vibraciones, según la distribución mostrada en la Figura 3.2.
- ✓ La batería debe estar recubierta para evitar interferencias.
- ✓ Finalmente, el último punto a considerar en el diseño es la distribución del cableado. El diseño eléctrico se muestra en la Figura 3.3.

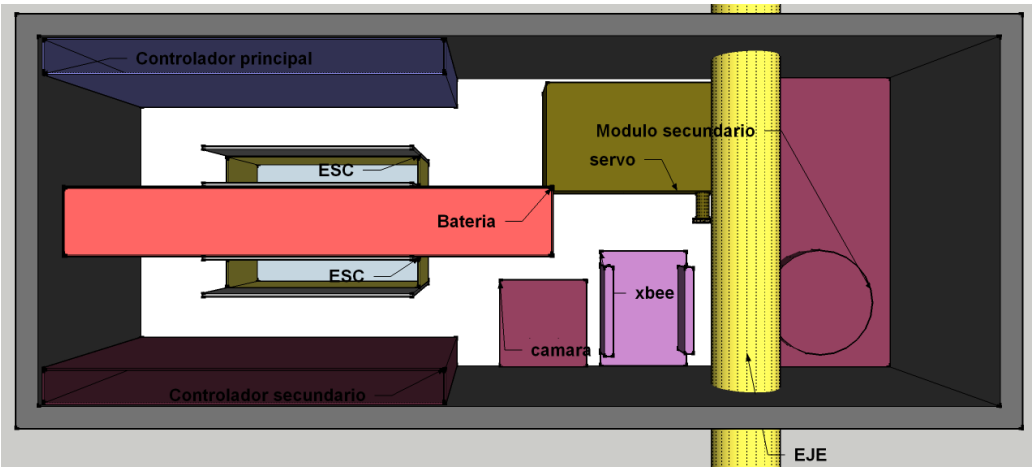



Figura 3.1: Distribución de los elementos dentro de la góndola

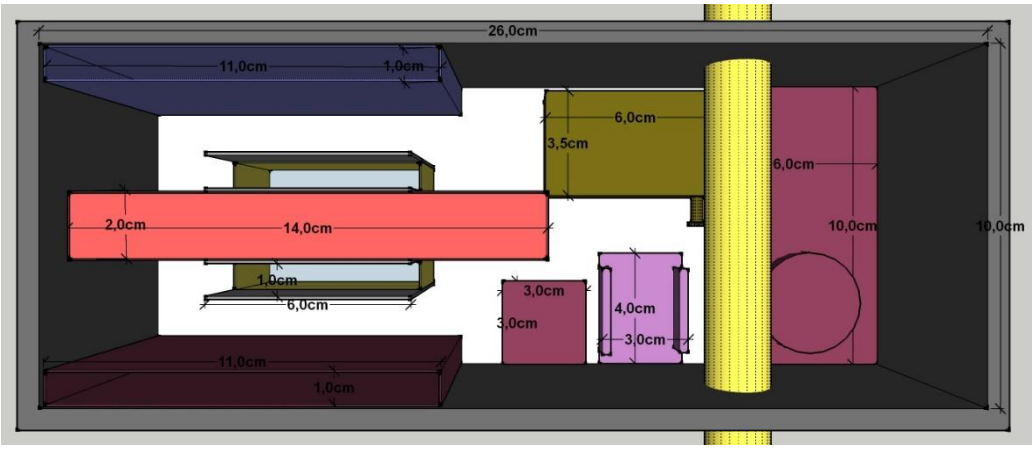



Figura 3.2: Dimensiones de los elementos dentro de la góndola

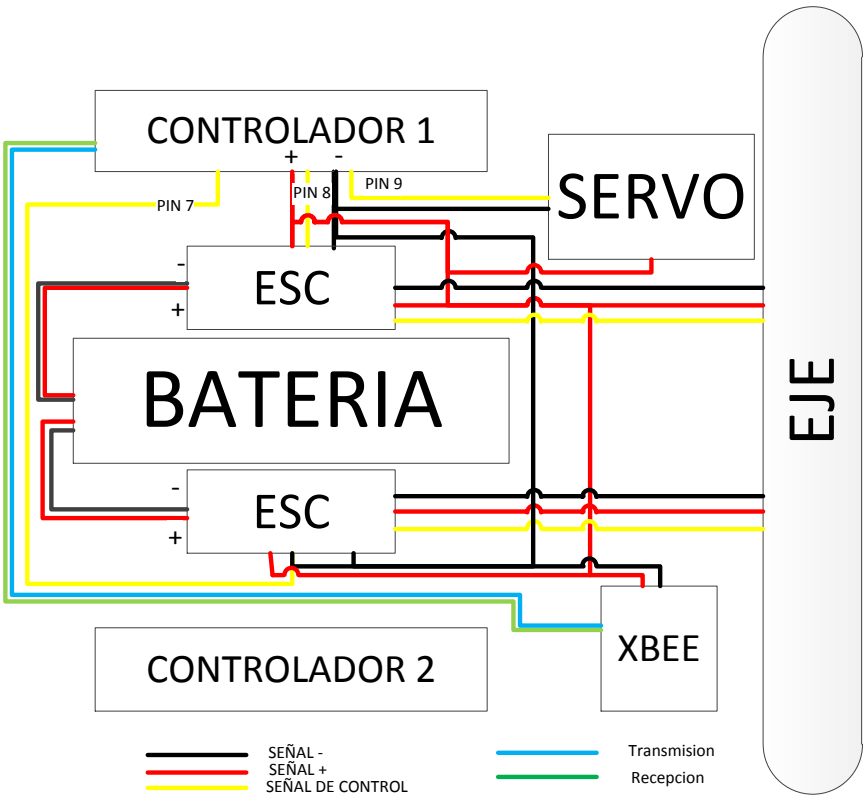



Figura 3.3: Esquema eléctrico de la góndola

## 3.3 Implementación de la góndola.

La góndola fue realizada sobre madera de balsa de forma artesanal, utilizando únicamente una cortadora, papel de lija, pegamento y un taladro. La plancha de madera de balsa que se utilizó para ensamblar la góndola tuvo un espesor de 6 mm y se cortó para que sea una caja de 10 cm de ancho por 27 cm de largo con 7 cm de altura (medidas internas). Todos los componentes fueron distribuidos siguiendo estrictamente el diseño antes explicado y adicionalmente fueron ajustados con sujetadores, pernos y tuercas.

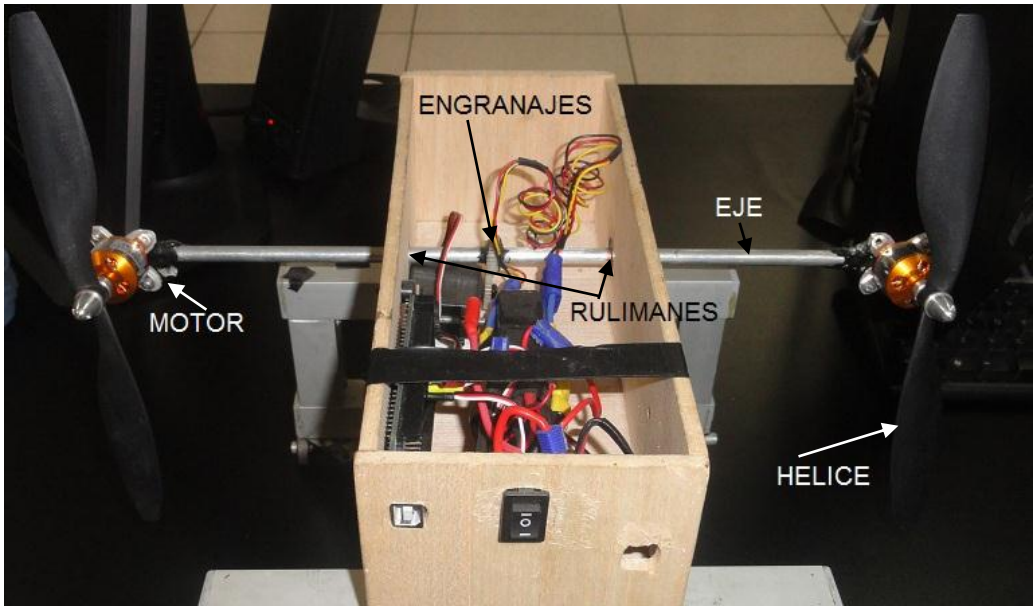



Figura 3.4: instalación del eje y servo motor.

En vista de que el eje gira a baja velocidad, se decidió acoplarlo a la góndola de manera artesanal, sin embargo, en la implementación se consideró a la ubicación de este componente como una parte clave de la góndola, puesto que el eje es la parte fundamental de los subsistemas de control de direccionamiento y recuperación que se explican en el capítulo 5 y además se requiere de simetría para evitar desgastes irregulares. Los rulimanes que sostienen al eje y evitan la fricción en el giro fueron colocados a una distancia de 7.5 cm de la pared y a una altura de 3.5 cm, estos tienen un ancho de 6.5 mm y un diámetro de 1.8 cm, tal como se muestra en la Figura 3.4. El servo motor que se muestra en la Figura 3.4 fue acoplado al eje por medio de engranajes con relación uno a uno.

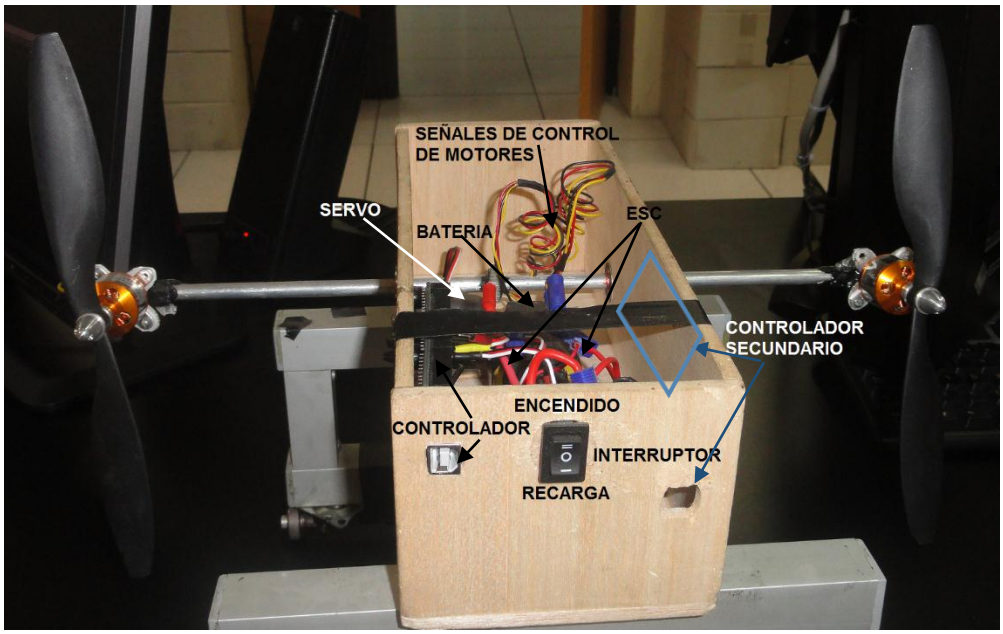



Figura 3.5: Góndola Implementada.

El resultado final se muestra en la Figura 3.5, donde se puede observar que se siguió estrictamente el diseño al momento de ensamblar la góndola y además se dejó espacio libre para el modulo secundario. El producto final tuvo un peso exacto de 965 g sin contar con los componentes del módulo secundario lo cual está dentro del máximo peso que se asumió para el cálculo de las dimensiones del dirigible. Para concluir, recalco que el diseño de la góndola incluye una tapa superior, la cual no se muestra en las fotografías por efectos de visualizar el interior de la misma.

# CAPÍTULO 4

# ENLACE DE COMUNICACIÓN

En los capítulos anteriores, nos dimos cuenta de las posibles aplicaciones que tiene este proyecto de graduación, recordamos y estudiamos muchos conceptos empleados en el diseño de este sistema y vimos en detalle cómo luce físicamente la góndola que podrá ser instalada en el dirigible. A partir de ahora profundizaremos en el diseño y construcción de cada una de las partes de este sistema (enlace de comunicación, subsistemas de control y por último la aplicación móvil).

La parte fundamental del desarrollo del proyecto es el establecimiento del canal de comunicación, puesto que por este se transmiten las señales de control para direccionar la plataforma de vuelo. Este canal consta de dos enlaces como se muestra en la Figura 4.1, la comunicación inicia en la salida *Bluetooth* del dispositivo móvil pasando por la radio base (dispositivo

intermediario) hasta el dispositivo inalámbrico receptor instalado en el dirigible.

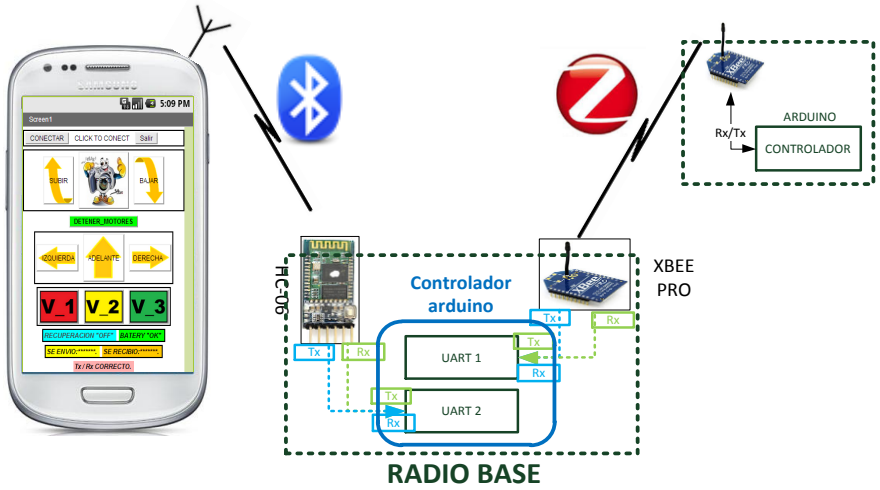



Figura 4.1: Descripción del enlace de comunicación

## 4.1 Selección de las tecnologías y protocolos de comunicación.

El seleccionar correctamente las tecnologías y protocolos de comunicación es la tarea básica para lograr establecer correctamente el canal de comunicación, en esta selección se tuvo en cuenta factores como: compatibilidad entre protocolos, fiabilidad del canal, ancho de banda del canal, rangos de alcance, entre otros. Para lograr seleccionar adecuadamente los protocolos y módulos de transmisión se dividió el estudio en tres partes.

En primer lugar, se analizó las salidas inalámbricas que posee el dispositivo móvil y se encontró la necesidad de tener un dispositivo intermediario que permita incrementar el alcance. En segundo lugar, se realizó el estudio y selección de los protocolos a utilizar en el dispositivo intermediario al cual lo llamaremos radio base de ahora en adelante. Y por último, se seleccionó los módulos de comunicación que mejor se acoplen en la implementación de la radio base.

Las salidas de datos de los dispositivos móviles son: Infrarrojo, *Bluetooth* y *Wifi*. La *salida Infrarrojo* utiliza una serie de LEDs infrarrojos que requiere una alineación correcta entre los patrones de radiación del emisor y del receptor y adicionalmente deben estar lo más cerca posible para que se lleve a cabo la comunicación. Las salidas *Bluetooth* y *Wifi* poseen una capacidad omnidireccional que permiten que exista comunicación inalámbricamente a pequeñas distancia sin la necesidad de una línea de vista.

Continuando con el análisis de las tres salidas antes mencionadas. Para empezar, se descartó Infrarrojo porque tienen como desventaja la necesidad de una línea de vista y además ya no viene integrada en algunos dispositivos móviles. La salida *Wifi* fue descartada, porque los módulos de comunicación que funcionan bajo este protocolo tienen un costo superior a los *Bluetooth*, y además existe escasa información

disponible en la red sobre el uso de esta salida por una aplicación móvil. Finalmente, la salida seleccionada para enviar los datos generados por la aplicación fue *Bluetooth* debido a la disponibilidad de la información existente y compatibilidad en la implementación con la aplicación para el dispositivo móvil, tal como se muestra en la parte superior izquierda de la Figura 4.1.

La radio base es un punto intermediario para nuestro canal de comunicación de dos enlaces, esta tiene como finalidad incrementar el rango de alcance de la señal, de tal manera que podamos controlar el dirigible a una mayor distancia que la permitida por Bluetooth, y además tiene la finalidad de lograr el entendimiento entre el protocolo Bluetooth y el protocolo de mayor alcance ZigBee. El protocolo Bluetooth, tal como se mencionó en el capítulo de marco teórico, permite a los dispositivos comportarse como maestros o esclavos para soportar dos tipos de topología de red, piconet y scatternet, tal como se observar en la Figura 2.2. En nuestro caso tenemos dos dispositivos que se encuentran conectados de la siguiente manera: un maestro que inicia y solicita la comunicación (dispositivo móvil) y un esclavo que acepta formar parte de la red (modulo Bluetooth de la radio base). El segundo protocolo seleccionado para formar parte del canal de

comunicación fue el protocolo ZigBee, el cual permite tener topologías de red punto a punto, punto multipunto y malla.

Los módulos seleccionados fueron el HC06 y el *Xbee Pro*, los cuales funcionan bajo los protocolos *Bluetooth* y *ZigBee* respectivamente. El HC06 tiene un alcance máximo de 10 m y puede transmitir datos desde 1200 bps hasta 1.3 Mbps usando puerto serial UART, este sólo puede funcionar como esclavo y utiliza sencillos comandos AT para su configuración [**10**]. El *Xbee Pro* es un módulo de radio frecuencia que usa el puerto serial UART para transmitir datos a velocidades desde 1200 bps hasta 230 Kbps (Rx y Tx) utilizando un patrón de radiación omnidireccional con alcance teórico de 90 m sin línea de vista; se puede configurar a través de un *hyperterminal* o con el software de programación X-CTU, donde por medio de una interface serial hacia la PC se envían comandos AT. Adicionalmente, el módulo *Xbee Pro* soporta topología de red punto a punto en modo transparente, lo cual significa que los módulos se comporta como un "cable inalámbrico" de tal forma que los datos que se colocan en el buffer del puerto serie pueden ser transmitidos y escritos en el buffer del puerto serie del otro modulo; este módulo de comunicación fue seleccionado por su economía en comparación con otros, su facilidad de configuración y de implementación y obviamente porque se acopla perfectamente a las

características (ancho de banda, alcance, confiabilidad…) necesarias para la implementación de este proyecto. [**11**]

## 4.2 Diseño del enlace de comunicación dispositivo móvil – radio base.

El enlace de comunicación (dispositivo móvil – radio base) permite al dispositivo móvil establecer un enlace punto a punto, por medio de la salida *Bluetooth*, con el módulo *Bluetooth* HC06 que se instaló en el dispositivo intermediario. El diseño de este enlace se refiere a la selección de todos los parámetros a configurar, iniciando en el dispositivo móvil donde se generan los datos hasta el módulo HC06 encargado de entregar los datos al microcontrolador que realiza la función de traductor. La aplicación móvil solo acepta solicitudes de establecimiento de conexión provenientes del módulo HC06 instalado en la radio base, a través del código de bloques que se muestra en la Figura 4.2:  (la variable MAC es la dirección MAC del módulo HC06); el detalle del significado de estos bloques es analizado en el capítulo 6.

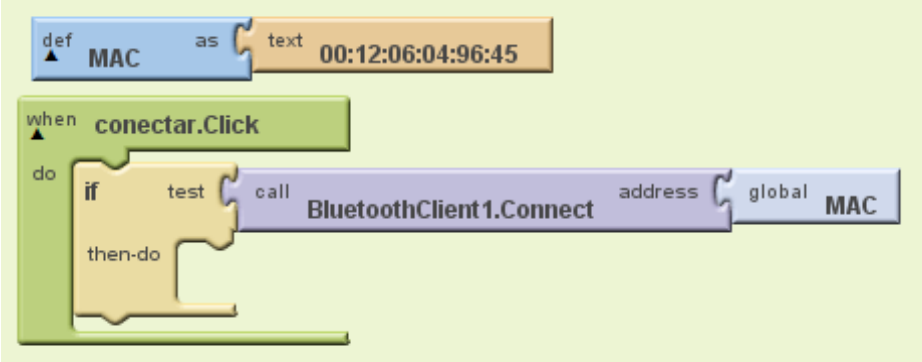



Figura 4.2: Código de bloques de la aplicación que solicitan la comunicación con el módulo HC06

El módulo HC06 con el que se comunica el dispositivo móvil fue configurado por conexión serial USB con la PC, para trabajar a 9600 baudios, sin paridad y con el nombre de Tesis. Los *comandos AT* utilizados para ajustar estos parámetros fueron los siguientes: al enviar un “AT” se recibió un “OK”, lo cual indica que entramos a modo de configuración; al enviar un “AT+BAUD4” se recibió un “OK9600” ajustando la velocidad de transmisión a 9600 bps; al enviar un “AT+NAMETesis” se recibió un “OKname”, ocasionando que el nombre del módulo sea Tesis); y por último se envió un “AT+PN” y se recibió un “OKNONE”, con lo que le indicamos al módulo que trabaje sin paridad. [**10**]

## 4.3 Diseño del enlace de comunicación radio base - dirigible.

En el segundo enlace del canal de comunicación, los módulos fueron configurados en modo transparente, esto quiere decir que todo lo que

ingresa al pin 3 (entrada de datos) es guardado en el buffer de entrada para luego ser transmitido y todo lo que ingresa como paquete RF es guardado en el buffer de salida y luego enviado por el pin 2 (salida de datos). En este caso los parámetros a configurar fueron seleccionados con la finalidad de tener una conexión Punto a Punto compatible con el enlace *Bluetooth*, para esto se utilizó el software *XCTU* y los comandos MY (id del dispositivo local), DL (id del dispositivo con el que se debe establecer la sesión), ATBD (selecciona los baudios), ATCH (selecciona el canal) y ATID (asigna nombre al enlace). Los valores de los parámetros utilizados para tener la comunicación punto a punto son los siguientes: En el ***Xbee Pro*** **A** el parámetro MY = **10** (ID local es 10), DL = **11** (ID del vecino es 11), ATBD**9** (115200 bps), ATCH**C** (use canal C) y ATID**222** (nombre del enlace); y en el ***Xbee Pro*** **B** el parámetro MY = **11** (ID local es 11), DL = **10** (ID del vecino es 10), ATBD**9** (115200 bps), ATCH**C** (use canal C) y ATID**222** (nombre del enlace); una vez realizadas estas configuraciones y conectados los respectivos pines de alimentación, el enlace se encuentra listo para operar. [**11**]

## 4.4 Implementación del canal de comunicación.

El canal de comunicación está formado de dos enlaces punto a punto. El enlace que usa el protocolo *Bluetooth* para enviar los datos generados por la aplicación desde el dispositivo móvil hasta la antena

*Bluetooth* HC06 colocada en la radio base, tal como se muestra en la parte superior izquierda de la Figura 4.1; y el enlace sobre el protocolo *ZigBee*, desde el *Xbee* instalado en la radio base hasta otro *Xbee* instalado en la góndola del dirigible, según se observa en la parte superior derecha de la Figura 4.1, el mismo que permite retransmitir las señales de control con un mayor rango de cobertura que el permitido por la tecnología *Bluetooth* con la finalidad de controlar la plataforma de vuelo a una mayor distancia.

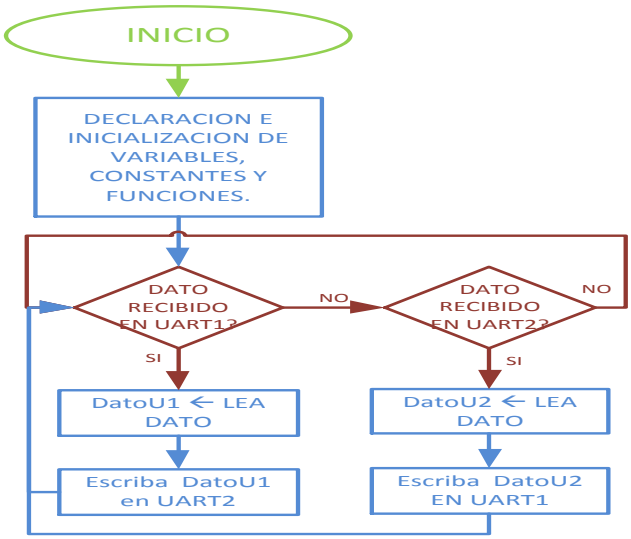



Figura 4.3 Diagrama lógico del funcionamiento de los controladores

Una vez establecidos los diferentes parámetros a configurar en los módulos de comunicación se procedió con la implementación de la radio base, la cual permite tener un solo canal a través de la interacción de los dos enlaces ya mencionados. La radio base funciona como un

interpretador de datos entre los protocolos *Bluetooth* y *ZigBee*. El dispositivo controlador de la radio base permite almacenar en el UART2 (Rx) los datos que se reciben del HC06, para posteriormente escribirlos en el UART1 (Tx) del mismo controlador, y viceversa, los datos que se reciben del UART1 (Rx) conectado al *Xbee Pro* son almacenados para posteriormente escribirlos en el UART2 (Tx) conectado al HC06; es decir, cada que recibe un dato por uno de los puertos seriales, este se reenvía por el otro puerto serial hacia el módulo de comunicación opuesto; tal como se muestra en el diagrama de flujo de la Figura 4.3.

Para finalizar, se realizaron las respectivas conexiones utilizando cable hembra a hembra y una batería de 9 V para realizar las pruebas. Una vez comprobado que el canal de comunicación funciona correctamente se realizó el respectivo diseño en PCB, el resultado de la placa para la radio base puede visualizarse en la Figura 4.4. Las pistas de la radio base fueron realizadas exclusivamente para colocar los módulos *Bluetooth* y *Xbee* junto con la batería de 9 V, sin embargo esta puede ser modificada si la instalación del módulo secundario lo requiera.

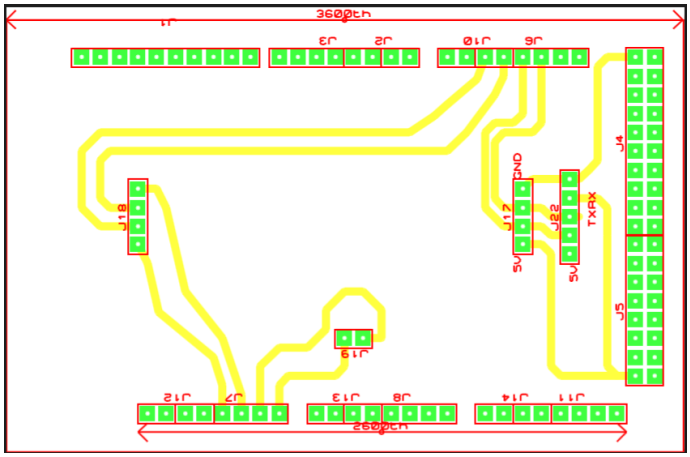



Figura 4.4: Conexiones físicas de la radio base.

# CAPÍTULO 5

# DISEÑO DE LOS SUBSISTEMAS DE CONTROL

Teniendo implementado el canal de comunicación que nos permite transmitir las señales de control a ser interpretadas por cada uno de los subsistemas de control de la plataforma de vuelo no tripulada, la siguiente etapa es el diseño e implementación de estos importante subsistemas de control. Es importante recalcar que la confiabilidad en el manejo del dirigible depende del adecuado funcionamiento, activación y desactivación de estos subsistemas. Para controlar la plataforma de vuelo se utilizan los subsistemas de control de direccionamiento, de velocidad, de recuperación, de comunicación y adicionalmente un control visual del nivel de energía. Este capítulo presenta al lector una descripción completa del controlador de direccionamiento del dirigible, comenzando con las tareas efectuadas por los subsistemas, los diagramas de bloques y de flujo de cada subsistema y además la implementación e integración de todos los subsistemas.

## 5.1 Subsistema de control de direccionamiento del dirigible

Los elementos que conforman el subsistema controlador de direccionamiento del dirigible son mostrados en la Figura 5.1, donde podemos observar que el controlador requiere de un dispositivo adicional para poder activar y desactivar cada uno de los motores y adicionalmente requiere un servo motor que manipule la dirección del eje al que se encuentran acoplados los motores derecho e izquierdo; todo este conjunto de piezas trabajando simultáneamente bajo las instrucciones del controlador permiten generar una fuerza aplicable en diferentes direcciones con la finalidad de manipular la navegación de la plataforma de vuelo según lo deseado por el usuario. Este subsistema controlador es capaz de reconocer, entender e interpretar seis direcciones, las cuales son generadas por el usuario a través de la aplicación móvil y enviadas por el canal de comunicación de dos enlaces hacia la plataforma de vuelo.

Los motores utilizados en el diseño de este proyecto fueron motores BLDC (Motores sin escobillas) de tipo sin sensores. Esta clase de motores requieren de una ESC (Controlador de velocidad electrónico) para generar un torque que permite al mismo tiempo permite controlar la velocidad de giro del motor al aplicar un menor o mayor torque.

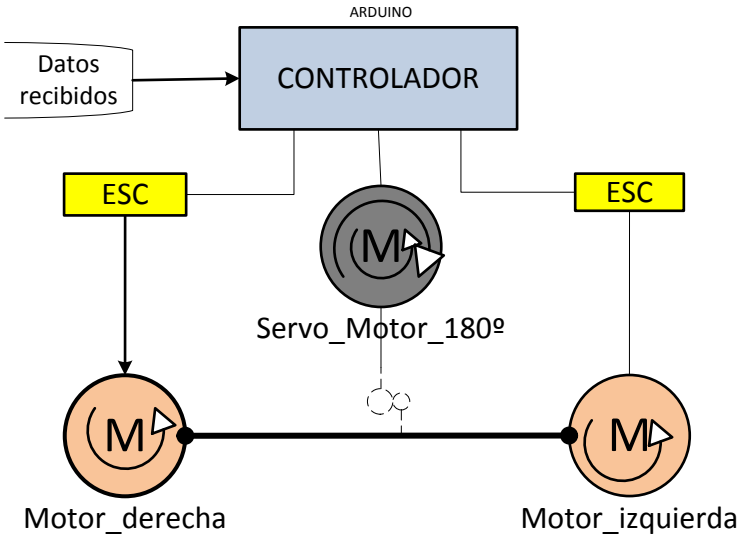



Figura 5.1: Diagrama de bloques del subsistema de control de direccionamiento

Las ESC's, en nuestro caso, usan el método de Conmutación trapezoidal en el cual se excita simultáneamente cada uno de los seis posibles pares dejando siempre una línea desconectada; la velocidad con la que se conmuta dicha excitación a un nuevo par es la que determina la velocidad de giro del motor, puesto que este tipo de controlador de velocidad se basa en que el tiempo entre una conmutación y el cruce por cero es igual al tiempo entre el cruce por cero y la siguiente conmutación. La ESC recibe una señal PWM con tiempos entre 1 ms y 2 ms para variar la velocidad de giro del motor entre 0 kV y 1000 kV (mínima y máxima velocidad teórica para el giro del motor); es importante recalcar, que con este tipo de controlador electrónico lineal, cuando el motor intenta arrancar a pequeñas velocidades la magnitud de la back-FEM es también pequeña lo que

dificulta la detección de los cruces por cero, es decir el control no es aplicado a bajas velocidades.

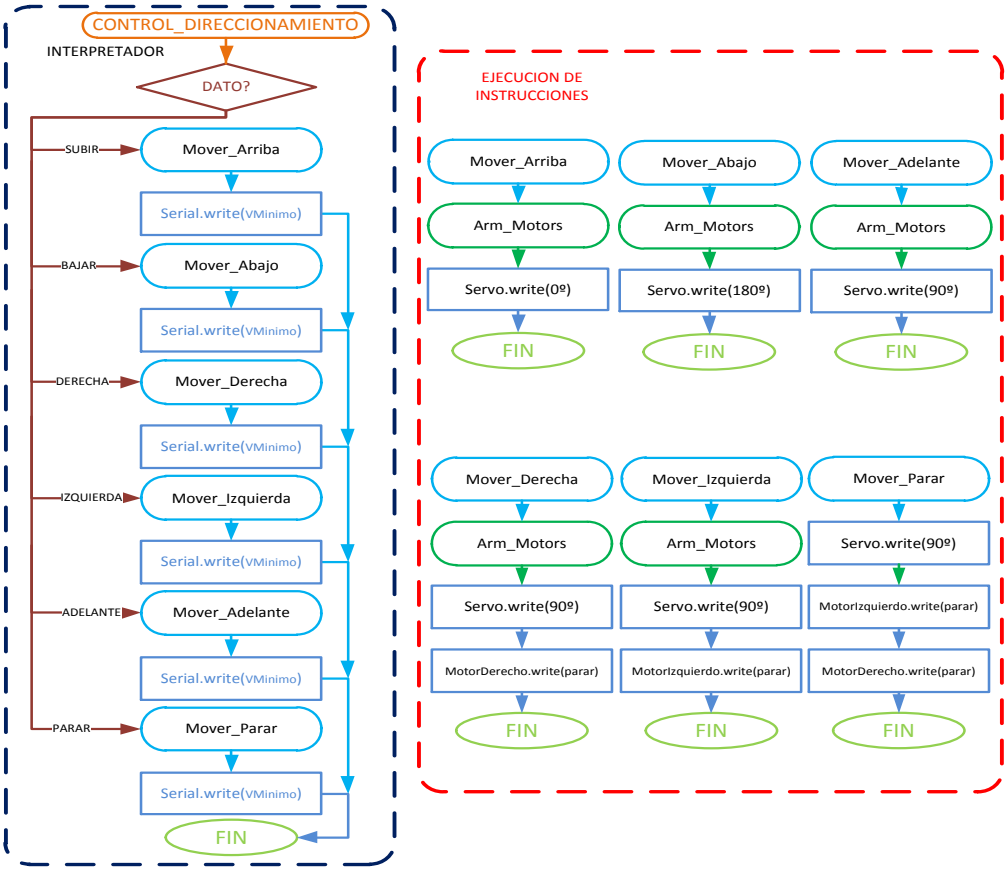



Figura 5.2: Diagrama de flujo del subsistema de control de direccionamiento

La operación logia de este subsistema es representada por medio de los diagramas de flujo mostrados en la Figura 5.2, en el diagrama izquierdo podemos observar que el controlador trabaja en tres etapas. En primer lugar, interpreta el dato recibido, el valor 0xFB es interpretado como subir, 0xED como bajar, 0xEA como derecha, 0XE9 como

izquierda, 0xEC como adelante y 0xEB como detener motores. En segundo lugar ejecuta una lista de acciones dependiendo de la decisión tomada por el interpretador, de tal manera que arma, activa o desactiva los motores con ayuda de la ESC. Por último se acciona el servo motor con la finalidad de girar el eje al que se encuentran acoplados los motores una determinada cantidad de grados.

Las acciones que toma el controlador después de interpretar los datos recibidos se muestran en la Figura 5.1 diagrama izquierdo. Los efectos que tiene cada una de estas acciones son muy importantes para entender el funcionamiento de este subsistema. Para subir se coloca el servo entre 60º y 85º, cada vez que se presiona el botón subir el ángulo del servo incrementa progresivamente 5º, con esto logramos que se alinean los motores para generar una fuerza que puede ser descompuesta en dos; una de ellas en el mismo sentido que el empuje (B), la cual permite elevar la plataforma de vuelo y la otra paralela al plano de tierra para continuar el movimiento hacia adelante. En el caso de desear activar bajar, se coloca el servo entre 95º y 120º con la finalidad de alinear los motores de tal manera que la plataforma de vuelo descienda gracias a la fuerza contraria al empuje que se ha generado, al igual que en la dirección subir se dos fuerzas. Para activar la dirección izquierda, el motor izquierdo se detiene, mientras que al

derecho gira; adicional a esto, el servo se coloca a 90º, de tal manera que el eje de los motores se encuentra paralelo al plano de tierra. Si se requiere girar a la derecha la plataforma de vuelo (el giro se realiza inverso a izquierda) se coloca el servo a 90º y se detiene el motor derecho mientras que el izquierdo es activado; cuando se solicita activar la dirección adelante ambos motores giran en el mismo sentido con el servo en 90º. Y para el último dato, detener los motores, se envía la señal con periodo 1.00 ms a las ESC. Finalmente el controlador reenvía el dato al usuario como confirmación de la acción que se ejecutó.

El armado de la escobilla es una parte clave en el control de los motores Brushless, para ello se requiere enviar a la ESC una señal PWM, en las que los tiempo de alto incrementen cada segundo, desde 1.00 ms hasta el mínimo tiempo de alto para el cual la escobilla no es capaz de detectar una back-FEM lo suficientemente alta como para generar un torque; todo esto se resume en un lazo *for* que escriba en la ESC una señal PWM con tiempo de alto entre 1.00 ms y 1.50 ms tal como se observa en la Figura 5.3.

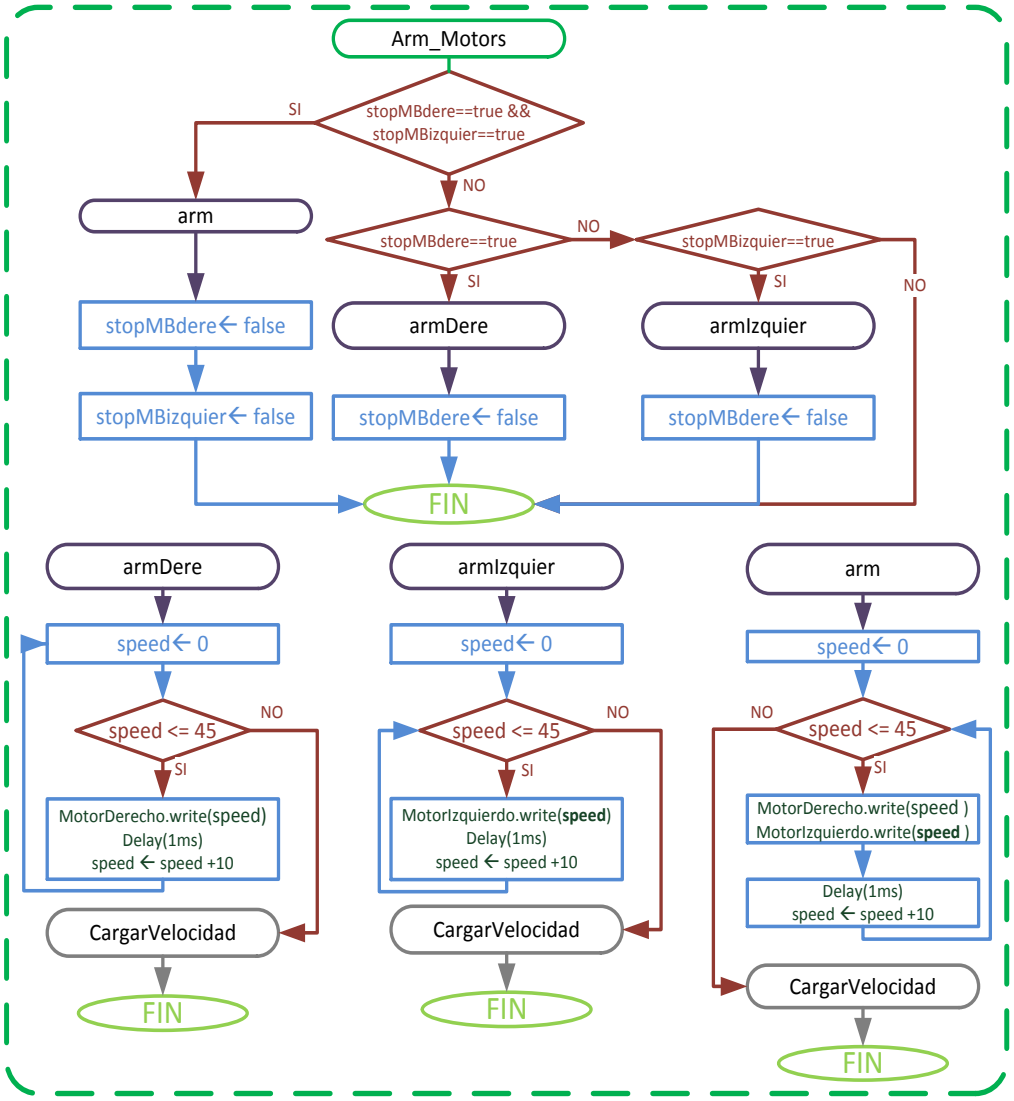



Figura 5.3: Diagrama de flujo del Armado de la ESC

Las ESC son armadas siempre que se ejecuta una dirección, puesto que cada que detenemos un motor es necesario volver a armarla para que este gire; para no intentar armar un motor que se encuentre en movimiento se crearon banderas (stopMBdere, stopMBizquier), las cuales indican que motor se encuentra detenido, de tal manera que por

medio de condiciones *if* se arma los motores individualmente si solo uno de ellos se encuentra detenido o en conjunto cuando ambos fueron detenidos; esto se muestra en la Figura 5.3.

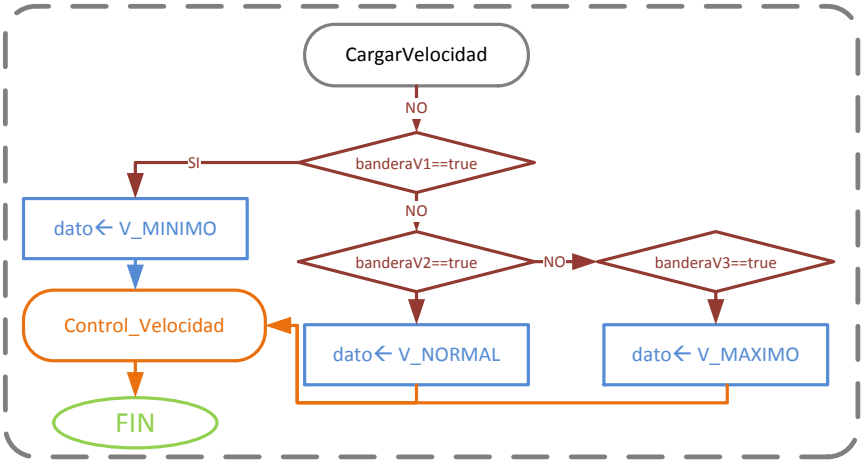



Figura 5.4: Diagrama de flujo de la función cargar Velocidad

En vista de que el armado de las ESC involucra una lista de cambios en las condiciones de velocidad del motor, es necesario actualizar la velocidad del motor al último valor solicitado por el usuario; para realizar esta tarea se llama a la función CargarVelocidad, la cual decide que velocidad cargar utilizando tres banderas (banderaV1, banderaV2, banderaV3) que se activan cuando una velocidad es activada por el usuario. Esto se puede observar con detalle en la Figura 5.4

## 5.2 Subsistema de control de velocidad del dirigible

El subsistema de control de velocidad tiene como objetivo fundamental permitir al usuario utilizar tres velocidades diferentes para navegar al

dirigible. El control de velocidad consiste en interpretar los valores 0xFE, 0xFD y 0XFC generados por la aplicación móvil como V_MINIMO, V_NORMAL y V_MAXIMO, respectivamente; para posteriormente ejecutar la velocidad seleccionada por el interpretador. El movimiento del dirigible con tres velocidades diferentes se logra a partir de controlar la velocidad de giro de los motores derecho e izquierdo, esto ocurre manipulando las ESC. Las ESC permiten variar la velocidad de giro del motor al recibir una señal cuadrada con frecuencia 2 Hz y cuyos tiempos de alto se encuentren entre 1.50 ms y 2.00 ms, en este caso las velocidades baja, media y alta se generan escribiendo señales PWM con tiempos de alto de 1.50 ms, 1.55 ms y 1.60 ms respectivamente.

El diagrama de bloques del control de Velocidad se muestra en la Figura 5.5, donde se puede observar que dependiendo del dato enviado a la función se activa la velocidad solicitada. En este caso se escribe en la ESC del motor derecho e izquierdo 40º, 45º o 50º con la finalidad de que el microcontrolador escriba los PWM con tiempo de alto de 1.50 ms, 1.55 ms y 1.60 ms respectivamente y por ende lograr generar las tres velocidades. Como se explicó en la sección anterior 5.1, para armar las ESC es necesario variar las señales controladoras de la velocidad giro del motor, por lo cual cada que se carga una velocidad baja, media

o alta se activa la bandera banderaV1, banderaV2 o banderaV3 respectivamente; de tal manera que al detener un motor y ponerlo a funcionar nuevamente, el sistema pregunte por cuál de las bandera esta activa y cargue la última velocidad solicitada por el usuario.

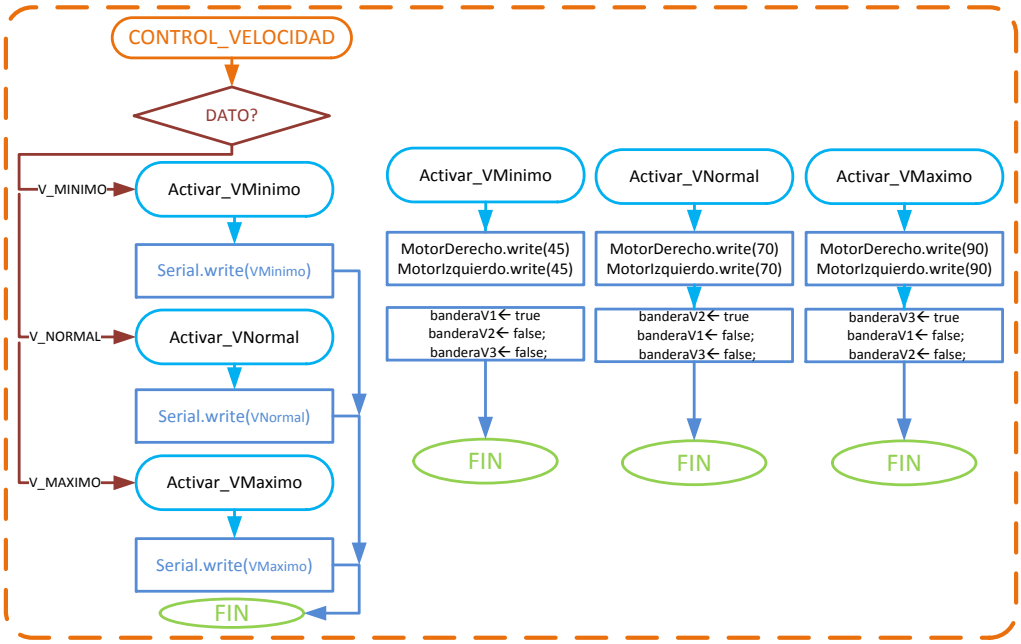



Figura 5.5: Diagrama de bloques del subsistema controlador de Velocidad

## 5.3 Subsistema de recuperación del dirigible.

El subsistema de recuperación del dirigible tiene como finalidad controlar al dirigible en caso de que el enlace de comunicación tenga problemas. El control de recuperación tiene como objetivo principal evitar la pérdida de la plataforma de vuelo, puesto que el helio que contiene el dirigible lo hace sumamente liviano y puede ser fácilmente

desplazado por el viento, cuando sobre el mismo no actúa ninguna fuerza que lo direccione. Este subsistema es capaz de permitir el descenso de la plataforma de vuelo y al mismo tiempo mantenerla girando hacia la derecha utilizando la mínima velocidad.

El diagrama de flujo de este subsistema se muestra en la Figura 5.6; aquí se observa que el proceso inicia armando los motores, en caso de que el usuario haya detenido alguno de ellos; luego se ajusta la dirección del eje al que se encuentran acoplados los motores con un ángulo de 135º para generar una fuerza de empuje descompuesta hacia abajo y hacia adelante; luego se escribe velocidad mínima en el motor derecho y velocidad máxima en motor izquierdo de tal manera que se genere una fuerza hacia abajo y otra hacia la derecha.

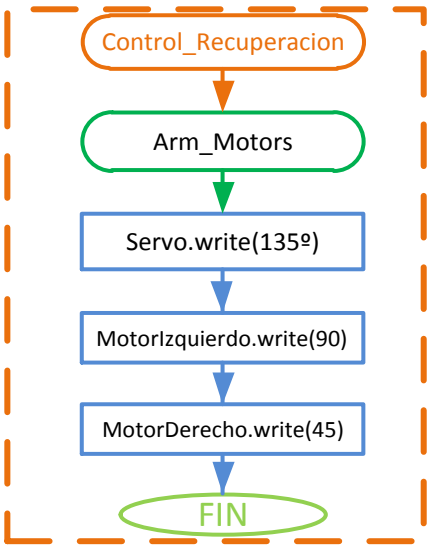



Figura 5.6: Diagrama de flujo del subsistema de recuperación

## 5.4 Subsistema de energía.

El subsistema de control de energía se basa en leer el pin *an0* (entrada analogía de voltaje) del microcontrolador para poder determinar con cuál de los 4 niveles de carga se encuentra operando la batería del sistema. Mientras la batería envía un voltaje de 11.1 voltios, el pin *an0* puede soportar voltajes entre 0 y 5 voltios; por ende el valor de tensión q recibe esta entrada analógica del microcontrolador es un divisor de tensión de la batería ajustado por medio de un potenciómetro a 5 V.

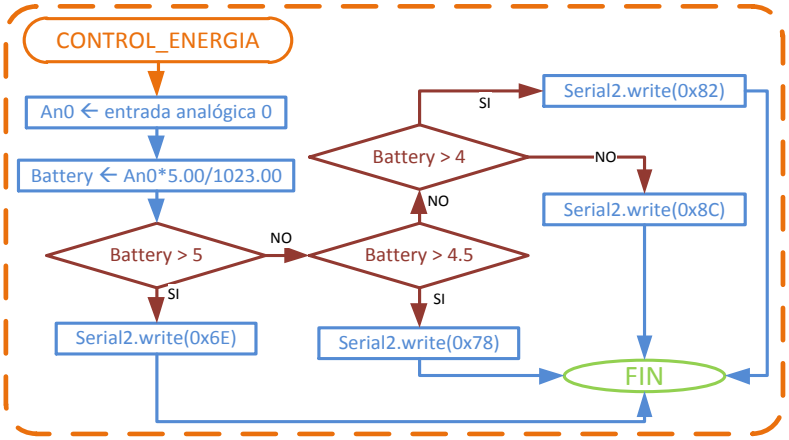



Figura 5.7: Diagrama de bloques del subsistema controlador visual de energía.

El diagrama lógico mostrado en la Figura 5.7 explica el funcionamiento de este subsistema. La primera tarea a realizar es detectar el valor de tensión actual proveniente del divisor de tensión ajustado a 5 V con la batería a plena carga. En segundo lugar se procede a comparar dicho

valor de tensión para saber si es mayor a 5, mayor a 4.5, mayor a 4 o menor a 4, lo cual indica que la carga de la batería está perfecta, buena, mala o agotada respectivamente. Finalmente, la última tarea es informar el estado de la batería al enviar los valores 0xE3 si la carga de la batería está perfecta, 0XE2 si es buena, 0xE1 si es baja, 0xE0 si la batería está agotada, estos datos se envían por medio del canal de comunicación hacia la aplicación móvil, la cual es capaz de entender los valores e informar al usuario el estado de la batería.

Adicional al control de energía se tiene el subsistema de control de comunicación tiene como objetivo determinar si existe algún corte en el canal de comunicación. Un corte puede ocurrir por la desconexión entre la aplicación móvil y la radio base por parte del usuario, con lo cual la aplicación móvil envía el dato 0xE8 antes de realizar la desconexión, al ser identificado este dato se activa el subsistema de recuperación.

Otra manera de tener una desconexión es que los módulos no se encuentren dentro del rango permitido para enviar datos (esta distancia se especifica en el capítulo 7 ) o existe un objeto que genera una sombra entre dos módulos, en cualquiera de los enlaces de comunicación; para identificar este suceso la aplicación móvil envía un *Hello* o un dato de autenticación (0xFF) cada tres segundos, este dato

es recibido por la plataforma de vuelo y reenviado hacia la aplicación móvil.

Finalmente el controlador general es quien determinar cuándo se declara que el canal de comunicación tiene una falla. En el diagrama de flujos mostrado en la Figura 5.8 se puede observar lo explicado anteriormente.

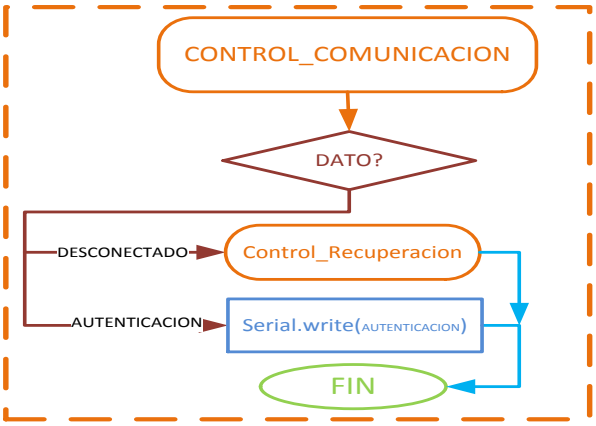



Figura 5.8: Diagrama de bloques del subsistema controlador de la comunicación

## 5.5 Implementación e integración de los subsistemas.

El desarrollo del controlador de la plataforma de vuelo se lo realizó sobre una placa controladora como es Arduino. Este controlador de propósito general utiliza su propio lenguaje de programación basado en C++ y permite dividir la codificación de un proyecto en cuatro partes:

- Declaración de librerías, variables, constantes y funciones necesarias para desarrollar el proyecto.

- Inicialización de todas las librerías, variables, constantes y pines utilizados, a través de la función denominada *void setup( )*.
- El lazo infinito llamado *void loop()*, el cual se encarga de ejecutar las tareas que debe realizar el controlador.
- Y finalmente tenemos el desarrollo de funciones, cada una de las funciones llamadas por desde el *void loop()* fueron previamente explicadas en los subcapítulos anteriores.

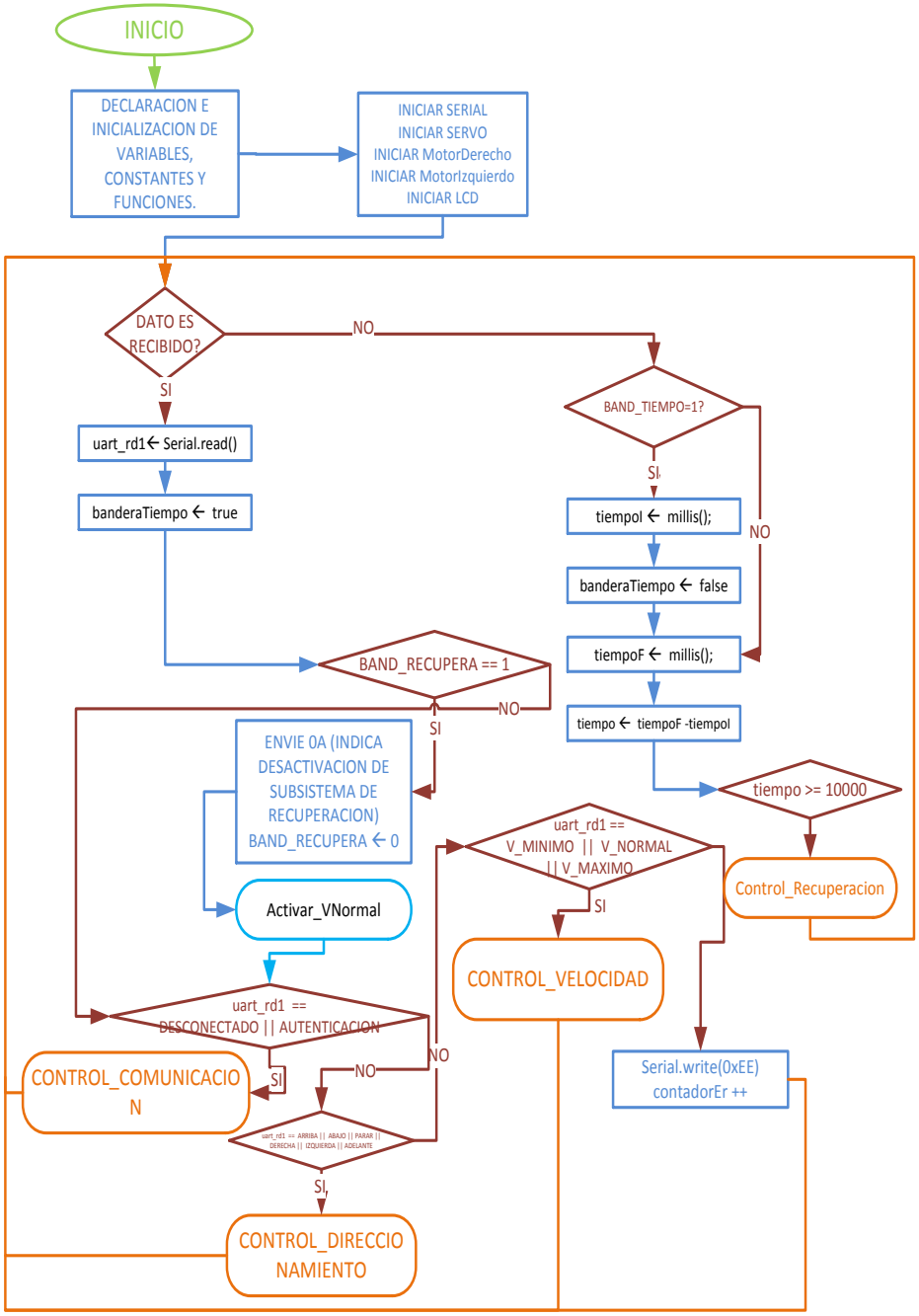



Figura 5.9: Diagrama de Flujos del controlador de la plataforma de vuelo

El funcionamiento lógico del controlador se muestra en el diagrama de flujos de la Figura 5.9. Después de declaradas todas las librerías, constantes, variables y pines a utilizar, se procede a la inicialización de los módulos de comunicación, servos y pines utilizados en este proyecto.

Una vez que el controlador ha ejecutado estas dos tareas, se pregunta si existe algún dato disponible para leer en el puerto serial dos. Si existe dato el controlador se encarga de leerlo, posteriormente decide a que subsistema de control le pertenece el dato y lo activa; si el dato no pertenece a ningún subsistema de control quiere decir que el dato es erróneo. Si no se recibe dato en el puerto serial dos, entonces se activa el contador de tiempo, con la finalidad de determinar por cuanto tiempo no se reciben datos o autenticación; sabiendo que el controlador debe recibir autenticación cada tres segundos, se procede a declarar el enlace como “con falla” cuando no han recibido tres *Hellos* consecutivos, o lo que es igual, el tiempo sin recibir datos es mayor o igual a diez segundos; bajo estas condiciones se activa el subsistema de recuperación.

Cuando el subsistema de recuperación es ejecutado se activa una bandera, la cual nos permite indicar al usuario el momento en el que la

recuperación es desactivada. Una vez activada la recuperación, El controlador la desactiva únicamente si se recibe un nuevo dato. Cuando el nuevo dato es recibido se envía hacia la aplicación móvil el dato 0xE5 para indicar que el subsistema de recuperación de desactivó y se cambia a estado false la bandera de recuperación.

# CAPÍTULO 6

# DISEÑO DE LA APLICACIÓN MÓVIL

La última parte de este proyecto es el diseño y desarrollo de la aplicación móvil, la cual cumple la función de interfaz gráfica entre el usuario y sistema. Para desarrollar la aplicación móvil, en primer lugar se seleccionó el Sistema Operativo y la herramienta de desarrollo o IDE a utilizar. En segundo lugar se definió las funcionalidades de la aplicación móvil, y además, como esta debe interactuar con el módulo HC06 instalado en la radio base. Para finalmente, proporcionar al lector información importante sobre el desarrollo de aplicaciones móviles que utilicen la salida *Bluetooth* para transmisión de datos.

## 6.1 Estudio y selección de la herramienta de desarrollo.

Al momento de seleccionar la herramienta de desarrollo o también llamado IDE en el cual podamos crear nuestra aplicación fue necesario definir el sistema operativo más apto para instalarla. Los sistemas

operativos más vendidos para dispositivos móviles son: i OS desarrollado y patentado por Apple Inc., BlackBerry OS desarrollado por RIM, pero recientemente adquirido por Microsoft y por ultimo Android OS desarrollado por *Open Handset Alliace*, pero posteriormente comprado por Google. I OS es un software propietario y de código cerrado, el cual no permite la instalación del mismo en hardware de terceros y además dificulta que nuevos desarrolladores creen aplicaciones dado que obliga a los mismos a pagar una suscripción para poder desarrollar más el costo de la licencia del IDE para desarrollar aplicaciones; por estos motivos fue descartado. BlackBerry OS al igual que I OS, no permite la instalación del mismo en dispositivos de terceros y se requiere de software licenciado para desarrollar. Por otro lado, Android OS es un software de código abierto y de libre distribución bajo la licencia Apache, el cual fue seleccionado porque da la facilidad al desarrollador de crear, probar e instalar las aplicaciones que desee sin solicitar ningún tipo de suscripción o membresía y además existen IDE's de libre distribución que permiten el desarrollo de las mismas.

Para la selección del IDE utilizado en el desarrollo de la aplicación móvil se analizaron los software "*Basic4Android"*, "*Mono*" para Android, "*Live Code*" y "*App Inventor*"; de estos IDE mencionados se seleccionó App

inventor porque a diferencia de todos los demás es gratuito. “*App inventor*” es una plataforma sencilla de utilizar, puesto que no requiere de conocimientos previos de lenguajes de programación (ni C#, ni .NET, ni C, ni VisualBasic, ni Java). En su lugar utiliza un lenguaje de programación propio de tipo gráfico, donde no se escribe una línea de código, solo se arrastran los bloques y se los dispone para tener una secuencia que realice la acción deseada. Este IDE es impulsado por Google con el fin de que más personas se unan a la familia de Android, usa el browser como centro principal de trabajo, almacenando todo el código en servidores que están disponibles cada vez que entres a internet desde cualquier PC y lo mejor de todo es que al ser gratuito solo se necesita del acceso a internet para desarrollar, probar y descargar la aplicación desarrollada.

## 6.2 Diseño de las pantallas.

Previo al diseño de la interfaz gráfica de la aplicación, es necesario entender el proceso que permite a la aplicación realizar las tareas para las cuales fue desarrollada; en la Figura 6.1 se muestra gráficamente las interacciones que deben ocurrir para que se envíen los datos a la radio base. El usuario interactúa con la interfaz gráfica de la aplicación al presionar cualquier botón de la misma; cada vez que un botón es presionado se genera un evento a ser procesado por el CPU del

dispositivo móvil, con la finalidad de establecer la conexión con el modulo *Bluetooth* ya instalado en la radio base y posteriormente enviar los datos según corresponda por la salida *Bluetooth* del dispositivo móvil.

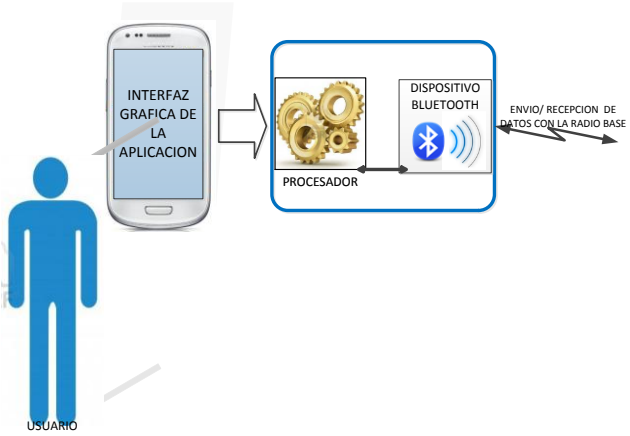



Figura 6.1: Funcionamiento gráfico de la aplicación

La aplicación tiene como objetivo fundamental la generación de señales que controlen el movimiento de la plataforma de vuelo según el usuario lo requiera. El interfaz gráfico de la aplicación deberá permitir a los usuarios interactuar de manera sencilla e intuitiva. La aplicación deberá generar una señal de control con cada evento generado por el usuario, para posteriormente ser enviada por el módulo *Bluetooth* del dispositivo móvil. El dispositivo móvil deberá poder conectarse con la radio base para que los datos sean transmitidos hasta la plataforma de vuelo. Se definirán once tipos de datos a enviar: subir, foto, bajar, izquierda, adelante, derecha, detener motores, velocidad baja, velocidad media, velocidad alta y el dato de autenticación (enviado cada 10 segundos);

antes de transmitir estas señales, el dispositivo móvil deberá estar conectado con la radio base.

Los diagramas de casos de uso son un modelo enmarcado en lenguaje gráfico, que nos permite, una vez capturados los requisitos, representar gráficamente como debe interactuar la aplicación con los usuarios. En nuestro caso, el diagrama de casos de usos que se muestra en la Figura 6.2 nos permite identificar claramente los requerimientos y las tareas que debe realizar la aplicación al interactuar con el usuario, y además nos muestra el orden en que deben ser ejecutadas dichas interacciones para que la aplicación funcione correctamente. En primer lugar, antes de intentar enviar cualquiera de las señales de control se debe establecer la conexión con la radio base. Para finalmente poder enviar las once señales de control establecidas en los requerimientos.

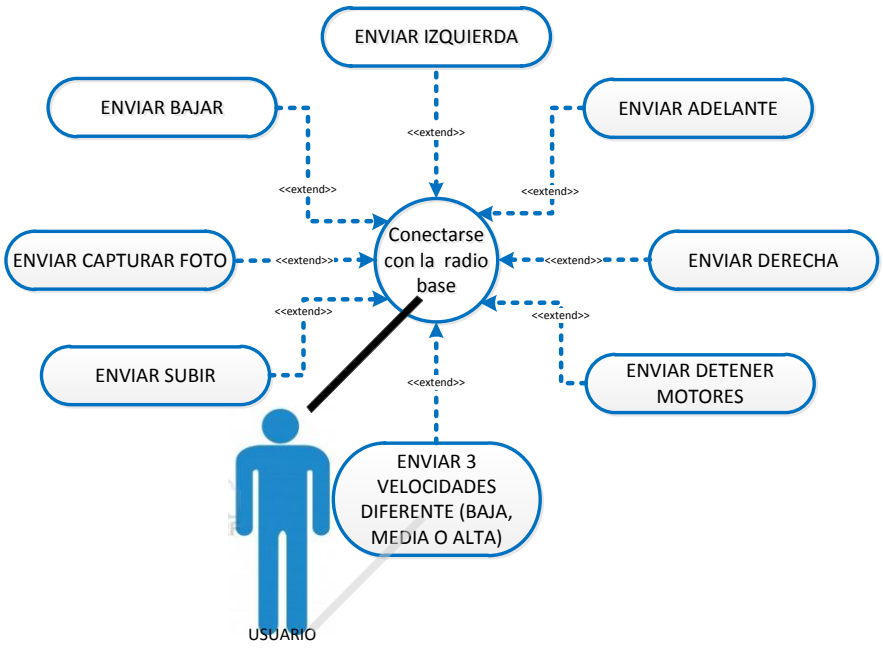



Figura 6.2: Diagrama de casos de usos de la aplicación.


BOSQUEJO DE LAS PLANTILLAS

Figura 6.3: Plantilla de la interfaz gráfica de la aplicación móvil

Siendo ya definidos los objetivos de la aplicación, es posible iniciar con el diseño de las plantillas de la interfaz gráfica, definiendo como serán distribuidos en la pantalla los arreglos y como serán colocados dentro de estos arreglos los botones, imágenes, cuadros de texto, y otros. En la Figura 6.3 se muestra la distribución de las plantillas. En el primer arreglo se colocan los botones de inicio de sesión *Bluetooth* y de cerrar aplicación, en el segundo y cuarto arreglo se colocan los botones que generan las señales de control de movimiento (subir, bajar, izquierda, adelante, derecha) más el botón para la señal de activar módulo complementario, en el tercer arreglo se coloca el botón para detener los motores de la plataforma de vuelo, en el quinto arreglo se colocan los

botones que controlan la velocidad de la plataforma de vuelo y por último en el sexto arreglo se colocan las etiquetas indicadoras del funcionamiento del enlace y de la plataforma.

## 6.3 Implementación de la aplicación.

A continuación vamos a profundizar en ¿Cómo se desarrolló la aplicación?, desde la creación de la interfaz gráfica hasta el desarrollo del código para el funcionamiento de la misma. En la Figura 6.4 se muestra el desarrollo de la interfaz gráfica basada en la plantilla antes definida. La interfaz gráfica consta de seis arreglos horizontales centrados distribuidos en la pantalla; en los cuales se colocan doce botones (conectar, salir, arriba, foto, abajo, parar, izquierda, adelante, derecha, velocidad_1, velocidad_2 y velocidad_3) que en nuestra aplicación pueden generar los eventos pulsar y presionar, más seis etiquetas (etiqueta1, recuperación, batería, enviado, recibido, correcto) que permiten al usuario visualizar el estado en el que se encuentra funcionando la aplicación, el enlace de comunicación y la plataforma de vuelo. Adicional a estos componentes, la aplicación consta de tres componentes no visibles: *BluetoothClient*, *Clock1* y *Clock2*. El componente *BluetoothClient* permite al teléfono establecer un enlace de comunicación (autenticar, enviar y recibir tráfico) con otro dispositivo *Bluetooth*. Los componentes *Clock1* y *Clock2* son temporizadores que

permiten generar un evento en intervalos regulares e tiempo (el intervalo más pequeño es un milisegundos), en nuestro caso estos componentes los vamos a utilizar para preguntar cada segundo si existe un dato a recibir por la conexión *Bluetooth* previamente establecida y para enviar cada diez segundos la señal de autenticación hacia la plataforma de vuelo sobre la misma conexión.

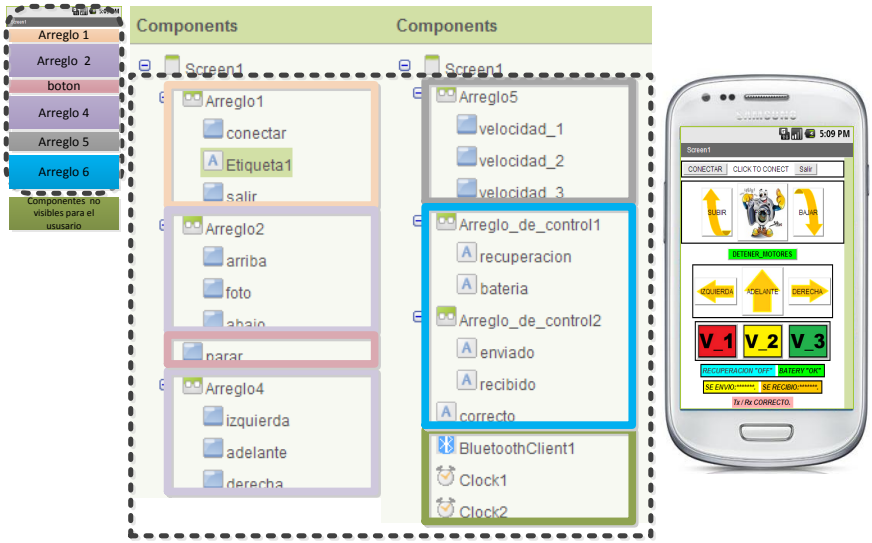



Figura 6.4: Desarrollo de la Interfaz gráfica.

El siguiente paso en la implementación de la aplicación es desarrollar el código que permita al usuario interactuar con los botones, etiquetas y componentes no visibles. Cuando se abre la aplicación la primera función que se ejecuta automáticamente es *Screen1.Inicialize* que se muestra en la Figura 6.5, esta función inicializa al botón conectar y las etiquetas etiqueta1 y recuperación de la pantalla asignándoles texto, color de texto y color de fondo; en nuestro caso, las dos primeras líneas

de código asignan a el botón conectar el texto “CONECTAR” con un fondo color verde, la siguiente línea indica que la etiqueta etiqueta1 se la inicializó con el texto “PULSE PARA CONECTAR” y las tres últimas líneas permiten asignar a la etiqueta recuperación el texto “RECUPERACION OFF” de color azul con un fondo gris oscuro.

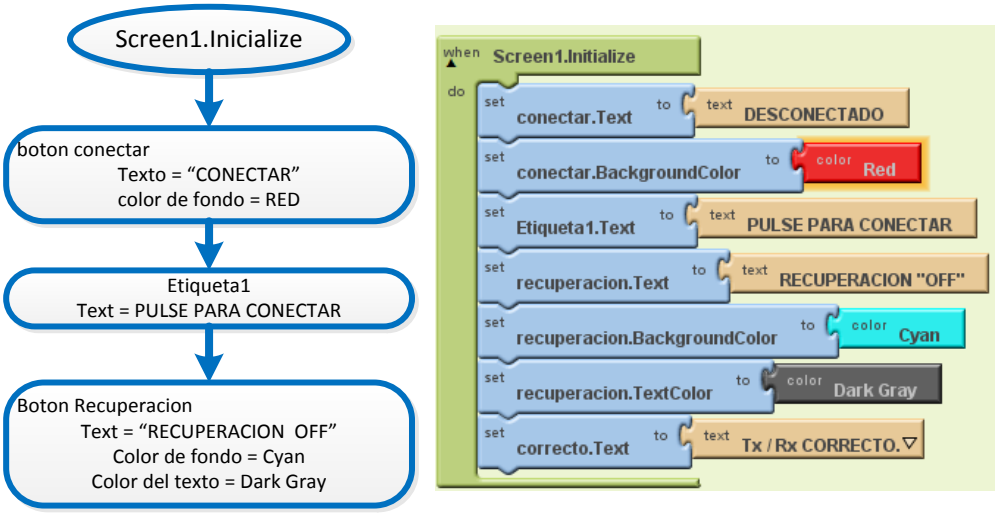



Figura 6.5: Diagrama de Bloques y código en bloques de la función para inicializar la pantalla

En el primer arreglo se colocó el botones conectar, la etiqueta etiqueta1 (permite informar al usuario como iniciar y terminar la conexión *Bluetooth*) y el botón salir. Al realizar el evento, pulsar el botón conectar, se ejecuta la función *Conectar.Click*, la cual fue desarrollada como se muestra en la Figura 6.6 gráfico superior a). Esta función permite conectar el dispositivo móvil con la radio base por medio del protocolo *Bluetooth*; la sentencia *if* permite al bloque *BluetoothClient1.Connect* enviar una solicitud con la dirección MAC

(00:12:06:04:96:45) del módulo colocado en la radio base; si la comunicación es establecida correctamente se ejecuta lo que está dentro de la sentencia *if*: se cambia el texto del botón conectar con "DESCONECTAR" con un color de fondo rojo, el texto de la etiqueta1 se cambia con "PRESIONE DESCONECTAR".

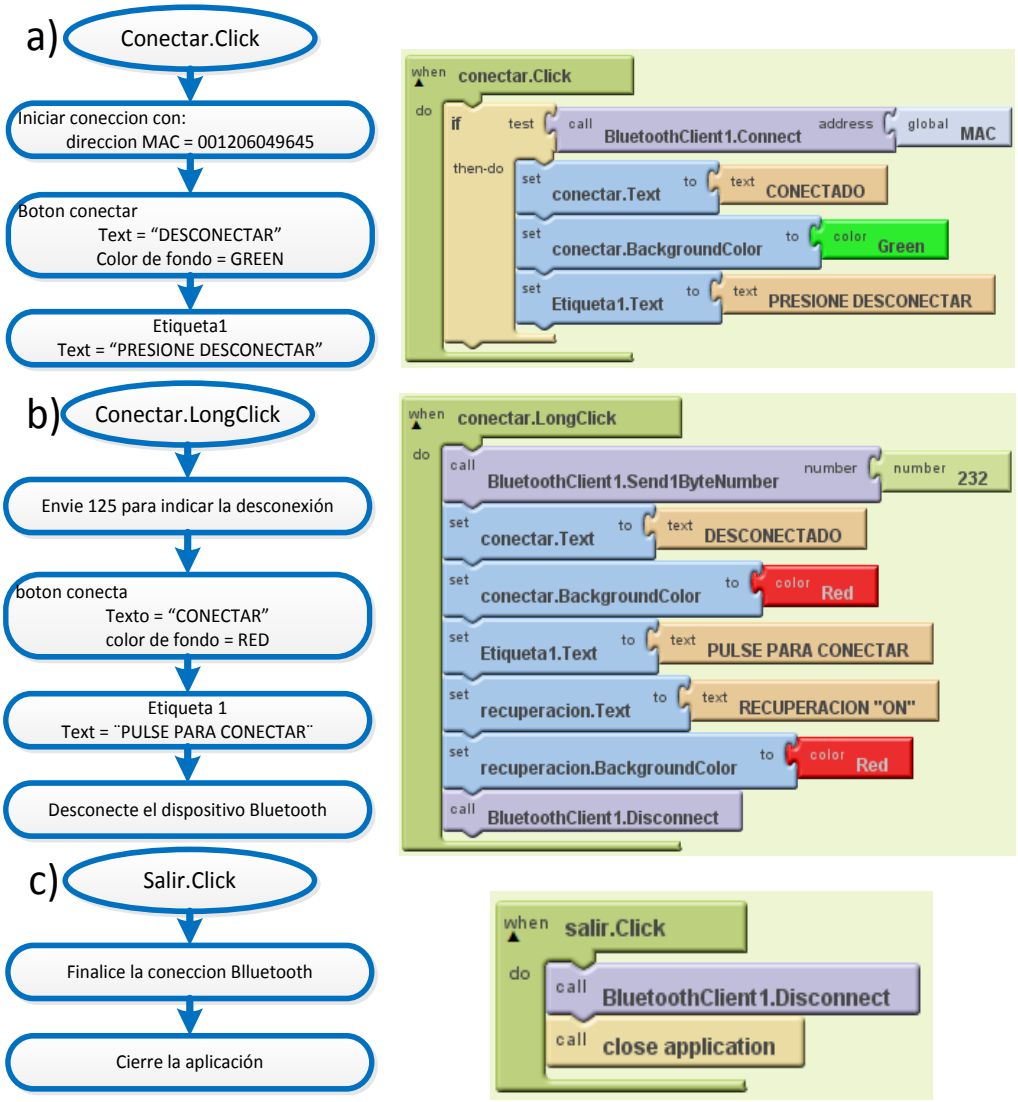



Figura 6.6: Diagrama de bloques y código en bloques de los componentes del primer arreglo

Otra función que se ejecuta con el botón conectar después de iniciada la conexión es *conectar.LongClick*, la cual se muestra en la Figura 6.6 gráfico intermedio b) , esta reconoce el evento presionar y realiza la tarea de desconectar el teléfono móvil de la radio base; antes de la desconexión se envía el número 232 para indicar a la plataforma de vuelo que se procederá con la desconexión, después se actualizan los atributos del botón conectar y la etiqueta Etiqueta1, el texto del botón conectar se cambia por “CONECTAR” sobre un color de fondo verde y el texto de la Etiqueta1 se cambia con “PULSE PARA CONECTAR”; el último bloque del código es el que realiza la desconexión. El último botón del arreglo es el botón salir, cuyo diagrama de bloques y código se muestra en la Figura 6.6 grafica inferior c); al realizar el evento pulsar salir se ejecuta la función *salir.Click*, la misma que efectúa dos tareas desconectar la comunicación por medio de *BluetoothCient1.Disconnect* y cerrar la aplicación por medio de *close application*.

Para el segundo arreglo se colocaron tres botones: arriba, foto y abajo. Estos botones tienen definido el evento pulsar; además permiten actualizar la etiqueta enviado, la cual muestra el valor numérico enviado y el significado del mismo para efectos de pruebas. Al pulsar el botón arriba se ejecuta la función *arriba.Click*, cuyos diagramas se muestran en la Figura 6.7 gráfico superior a), la función envía el número 251 por

la conexión *Bluetooth* y actualiza la etiqueta enviado con el texto “SE ENVIO: ARRIBA” concatenado con “(251)”. El siguiente botón de la lista es el botón foto, donde al pulsarlo se ejecuta la función *foto.Click*; esta función envía el número 231 por la conexión *Bluetooth* y actualiza la etiqueta enviado con el texto “SE ENVIO: FOTO” concatenado con “(231)”, tal como muestran las dos líneas de código en la Figura 6.7 gráfico intermedio b). Finalmente, al pulsar botón abajo de este arreglo, se ejecuta la función *abajo.Click* que se muestra en la Figura 6.7 gráfico inferior c), esta función envía el número 237 por la conexión *Bluetooth* y actualiza el texto de la etiqueta enviado con “SE ENVIO: ABAJO” concatenado con “(237)”.

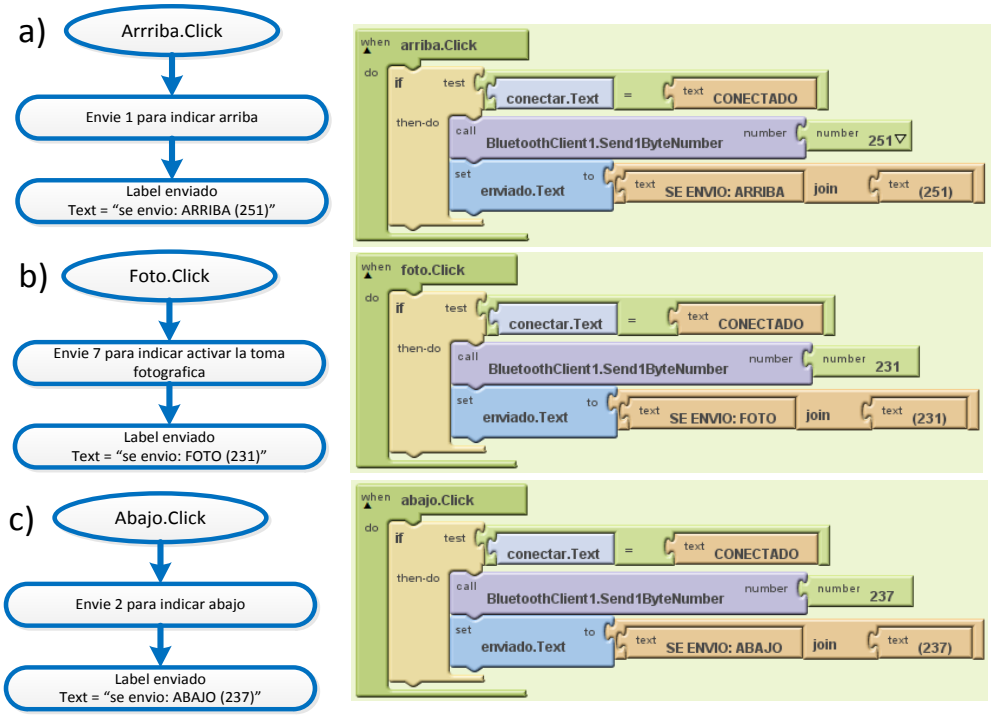



Figura 6.7: Diagrama de bloques y código en bloques de los componentes del segundo arreglo

En la tercera fila de nuestra aplicación se encuentra el botón parar, que permite ejecutar la función *parar.Click* al ser presionado. El diagrama de bloques y código se muestra en la Figura 6.8, este evento realiza dos tareas: enviar el número 235 por la conexión *Bluetooth* y actualizar la etiqueta enviado con el texto “SE ENVIO: PARA” concatenado con “(235)”.

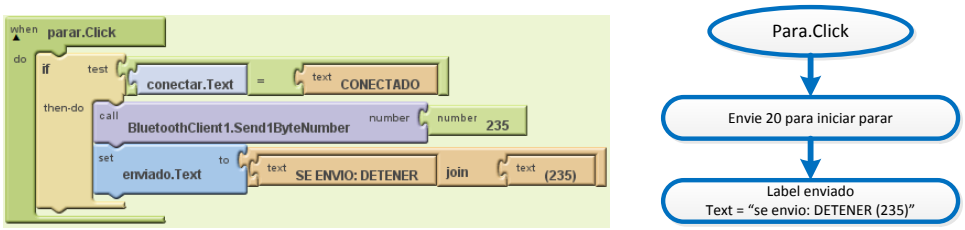



Figura 6.8: Diagrama de bloques y código en bloques del botón parar

El tercer arreglo ubicado en la cuarta fila posee tres botones: derecha, adelante e izquierda; los cuales al igual que en el caso anterior tienen definido únicamente el evento pulsar, con este evento se actualizan los atributos de la etiqueta enviado. Cada vez que se pulsa el botón derecha se ejecuta la función *derecha.Click*, cuyos diagramas se muestran en la Figura 6.9 gráfico superior a), la función envía el número 234 por la conexión *Bluetooth* y actualiza la etiqueta enviado con el texto “SE ENVIO: DERECHA” concatenado con “(234)”. El botón intermedio de este arreglo es el botón adelante, donde al ser pulsado se ejecuta la función *adelante.Click*, cuyas líneas de código y diagrama

lógico se muestra en la Figura 6.9 gráfico intermedio b); esta función envía el número 236 por la conexión *Bluetooth* y actualiza la etiqueta enviado con el texto “SE ENVIO: ADELANTE” concatenado con “(236)”. Finalmente, al pulsar botón izquierda de este arreglo se ejecuta la función *izquierda.Click* que se muestra en la Figura 6.9 gráfico inferior c), esta función envía el número 233 por la conexión *Bluetooth* y actualiza el texto de la etiqueta enviado con “SE ENVIO: IZQUIERDA” concatenado con “(233)”.

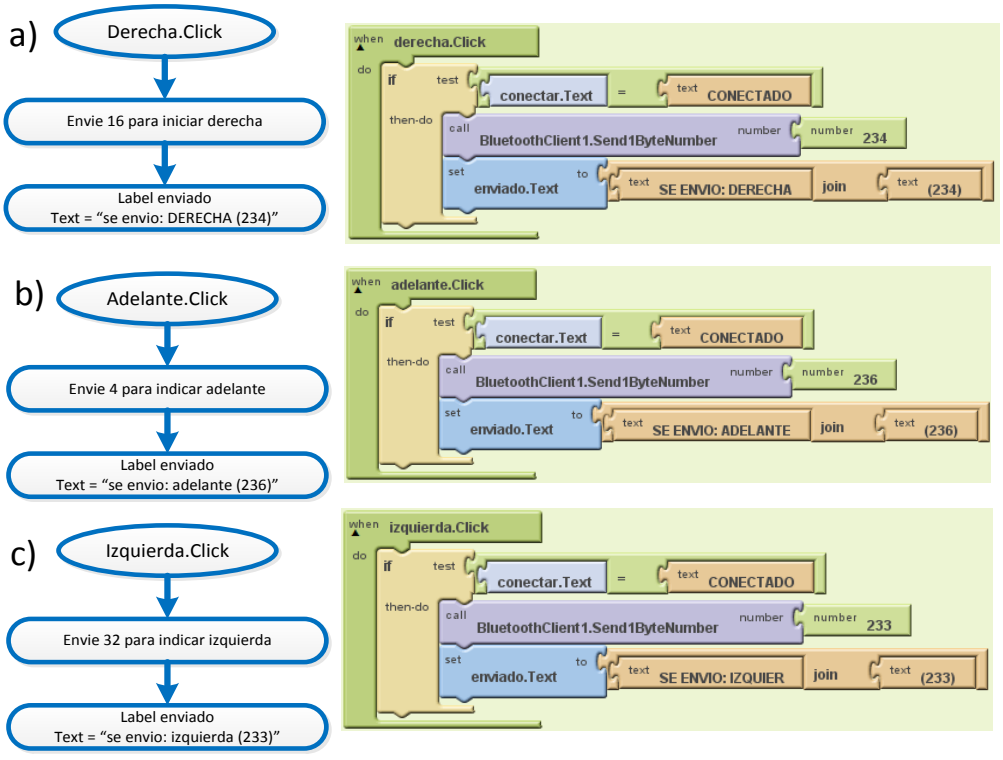



Figura 6.9: Diagramas de bloques y código en bloques de los componentes del tercer arreglo

En la Figura 6.10 se muestran los diagramas lógicos y código de los botones velocidad_1, velocidad_2 y velocidad_3 ubicados en el cuarto arreglo. El gráfico superior muestra el código de la función *velocidad_1.Click* que se ejecuta al pulsar el botón *velocidad_1*, esta función envía el número 254 por la conexión *Bluetooth* y actualiza la etiqueta enviado con el texto “SE ENVIO: V_1” concatenado con “(254)”. El gráfico b) de la misma figura muestra a la función *velocidad_2.Click*, la cual se encarga de enviar el número 253 por la conexión *Bluetooth* (plataforma de vuelo debe cambiar la velocidad al valor medio) y actualiza la etiqueta enviado con el texto “SE ENVIO: V_2” concatenado con “(253)”. El grafico c) en la parte inferior de la Figura 6.10 muestra las tareas que se realizar al ser ejecutadas la función *velocidad_3.Click*, la cual se encarga de enviar el número 252 por la conexión *Bluetooth* y actualiza la etiqueta enviado con el texto “SE ENVIO: V_3” concatenado con “(252)”.

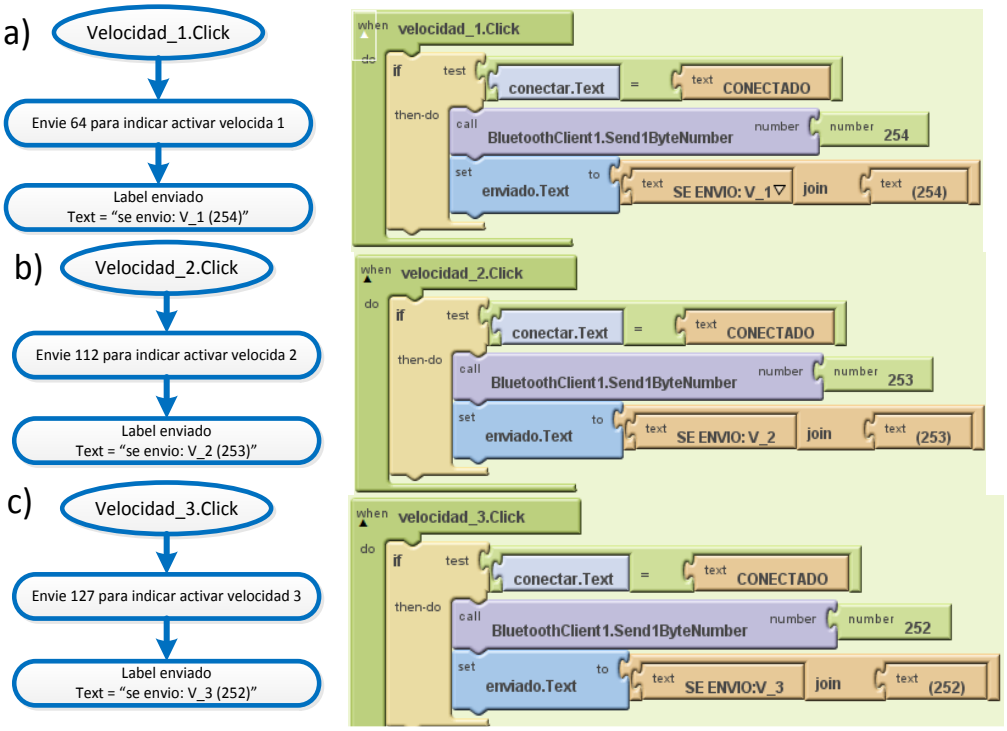



Figura 6.10: Diagramas de bloques y código en bloques de los componentes del cuarto arreglo

Las actualizaciones de las etiquetas son efectuadas cuando se genera un evento presionando cualquiera de los botones o al ocurrir un evento de tiempo con los componentes *Clock1* y *Clock2*.

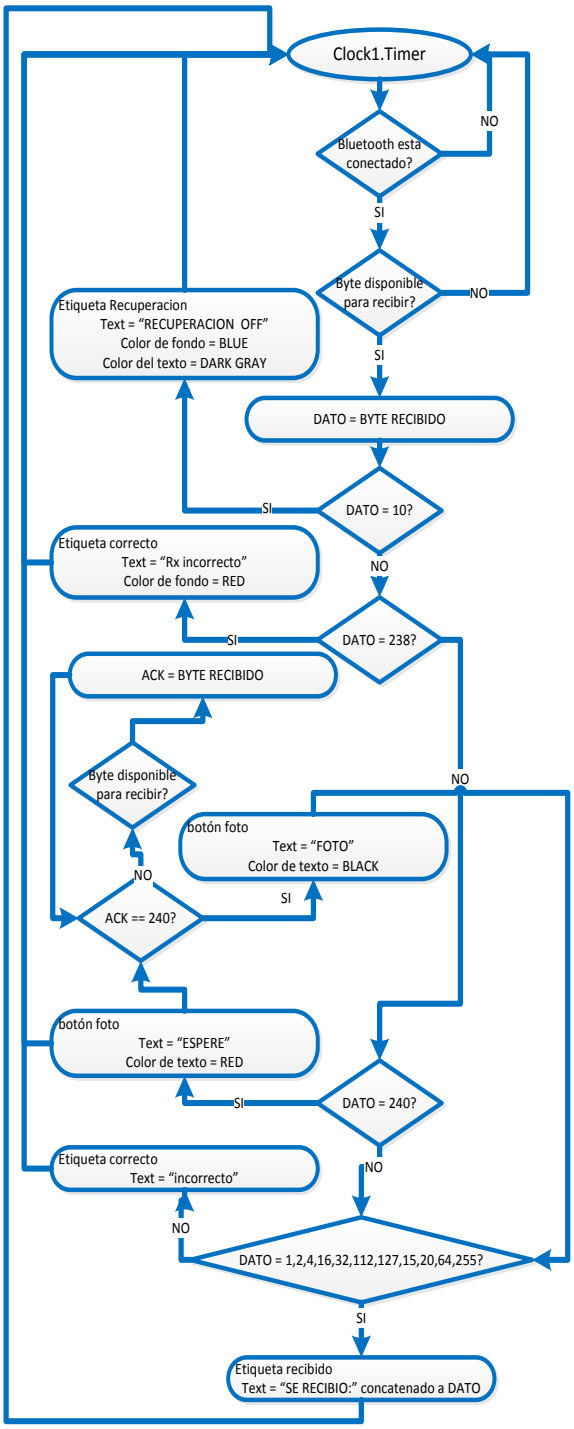



Figura 6.11: Código y diagrama lógico del temporizador Clock1.

La Figura 6.11 muestra el código y diagrama lógico de la función *Clock1*.*Timer*, la cual se ejecuta cada segundo con la finalidad de preguntar si por la conexión *Bluetooth* establecida se encuentra un dato disponible para leer, si existe un dato recibido lo analiza, actualiza las etiquetas adecuadas y se lo muestra al usuario. En la ejecución de esta función lo primero que se pregunta es por la conexión *Bluetooth*, si está disponible la conexión se pregunta por la existencia de un byte para leer, si el byte existe se lo lee y se lo almacena en la variable DATO. Una vez leído y almacenado el dato recibido se lo analiza para actualizar las etiquetas según corresponda; si el dato recibido es 228 actualizamos los atributos de la etiqueta recuperación con "RECUPERACION OFF" como texto, Blue como color de fondo y Dark Gray como color de texto; en caso contrario se pregunta si el dato recibido es 230 se actualizan los atributos de la etiqueta correcto con "Rx incorrecto" como texto y Red como color de fondo; en caso contrario se pregunta si el dato recibido es 251 237 235 234 233 236 254 253 252 232 231 250 249 248 247 246 245 244 243 242 241 240 239 238 o 255 y se actualiza la etiqueta recibido con el texto "SE RECIBIO:" concatenado al valor de DATO; caso contrario se actualiza la etiqueta correcto con el texto "INCORRECTO".

*Clock2.Timer* es una función de temporizador que se ejecuta cada 10 s y permite enviar a la radio base la señal de autenticación que es el número 255, esta señal permite al dirigible conocer cuándo se ha pierde la conectividad y cuando el usuario no desee mover la plataforma de vuelo; el diagrama lógico y código de la ya mencionada función se muestra en la Figura 6.12 aquí se observa la actualización del botón enviado con el texto “SE ENVIO: AUTENTICACION”.

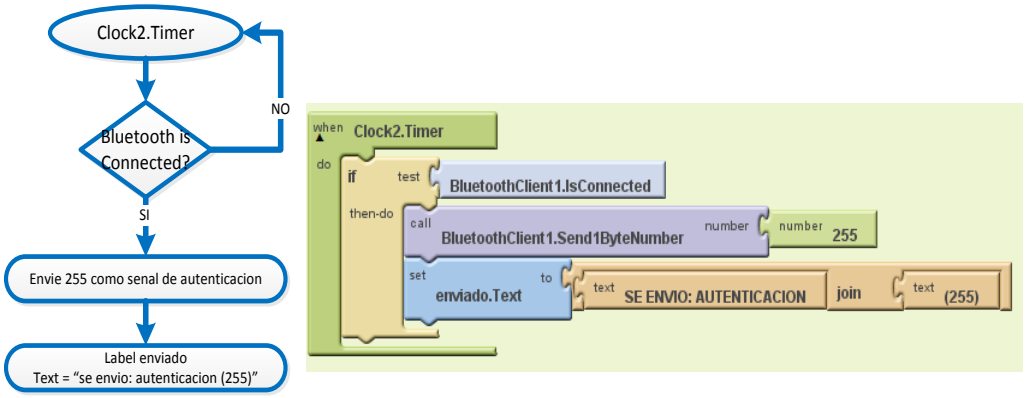



Figura 6.12: Diagrama lógico y código de la función *Clock2.Timer*

# CAPÍTULO 7

# PRUEBAS DE LOS SUB-SISTEMA

Para efectos de probar el funcionamiento de cada uno de los subsistemas de control, se desarrolló este proyecto sobre un controlador de propósito general como es Arduino y se implementó la góndola tal como se explicó en el diseño. En este sistema se realizaron pruebas de tipo cualitativo y cuantitativo, con la finalidad de determinar si el funcionamiento de los subsistemas, el canal de comunicación y la aplicación móvil son confiables y así determinar las limitaciones del sistema.

## 7.1 Prueba del control de direccionamiento.

El subsistema de control de direccionamiento consta de 2 partes fundamentales: los motores sin escobillas y el servo. Ambos cumplen funciones importantes, los motores se encargan de generar la fuerza de empuje necesaria para el desplazamiento de la plataforma de vuelo, mientras que el servo motor se encarga de controlar la dirección en la

cual se aplica esta fuerza. Para comprobar el correcto funcionamiento de este subsistema es importante validar dos temas.

Como primer punto, se validó que el torque que efectúa el servo es suficiente para mover el eje con los dos motores acoplados sin que este sufra algún tipo de desbalance. Para esto se procedió a cargar en el controlador el código que se muestra en el Anexo B, el cual permite ubicar al eje acopado con el servo motor a las posiciones 0º, 90º y 180º cíclicamente cada cinco segundos, esto ocurre por un lapso de tiempo de cinco minutos. Esta prueba fue repetida 30 veces, generando una muestra de tamaño n cuyos resultados también se muestran en el Anexo B. Esta prueba se realizó utilizando la góndola implementada como se muestra en la Figura 3.5; pero se utilizó como fuente de alimentación el cable de datos del Arduino, no la batería propia del sistema. Los resultados mostraron que el eje no se desvía ni un grado en dirección alguna, por ende los torques estructurales que sienten el eje y el servo cuando los motores están en constante movimiento no afectan este acople.

Y como último punto, para las pruebas sobre este subsistema, se comprobó que al ejecutar las direcciones arriba, abajo, derecha, izquierda y adelante los motores sean capaces de proporcionar una fuerza en la dirección solicitada. Esta prueba se divide en dos partes: la

cualitativa, si el subsistema de control de direccionamiento genera una fuerza capaz de mover a la góndola en la dirección indicada (posibles respuestas sí o no); y la cuantitativa, para lo cual se consideró colocar la góndola sujeta a el soporte con 4 ruedas (rulimanes) fijas en una sola dirección mostrado en la Figura 7.1, de tal manera que al colocarla sobre una superficie de baja fricción sea posible medir los metros o en su defecto los grados que la góndola se movió dependiendo de la dirección solicitada. Estas pruebas fueron realizadas sobre el prototipo completo que se muestra en la Figura 3.5 y además algunas fueron realizadas utilizando dos velocidades, tal como se describe a continuación. No obstante, es importante recalcar que ninguna de estas pruebas fue realizada utilizando el canal de comunicación.

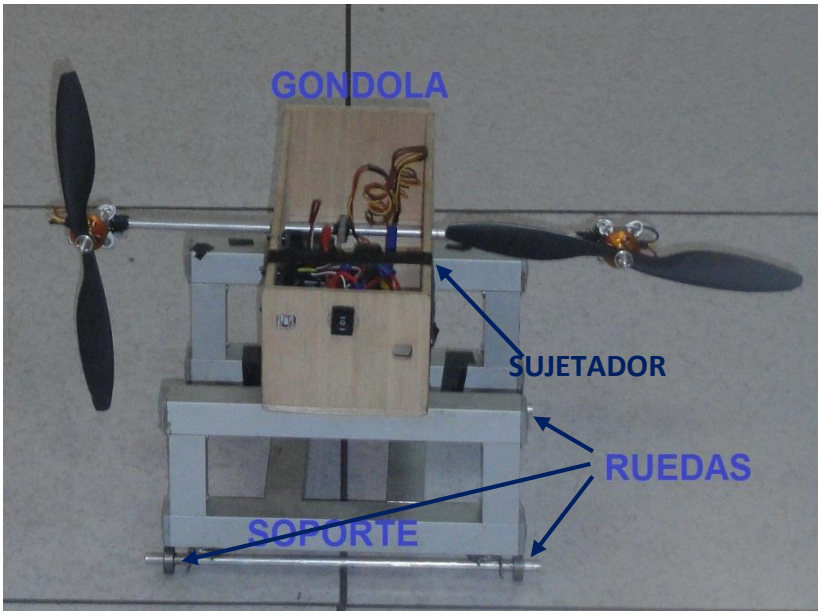



Figura 7.1: La góndola y su soporte para pruebas.

La prueba hacia adelante se realizó cargando en el prototipo el código mostrado en el Anexo C; este código permite activar los motores derecho e izquierdo a la misma velocidad durante 5 segundos cada vez que se reciba una señal de la aplicación móvil, para esta prueba el ángulo del servo motor se fijó en 90º. La señal de la aplicación es una señal de control que me permite ejecutar la dirección adelante después de haber realizado la lectura del dato, la distancia a medir con la cita métrica de 10 m es el desplazamiento hacia adelante que tuvo la góndola mostrado en la Figura 7.2.


Figura 7.2: distancia recorrida en la prueba hacia adelante

En la prueba hacia la derecha se utilizó el código mostrado en el Anexo D, el cual permite activar exclusivamente el motor izquierdo durante 10 segundos con cada recepción de la señal 234, dato que es generado por la aplicación móvil. En este caso se midió con un graduador el ángulo existente entre la posición inicial y final del extremo posterior

izquierdo del soporte de pruebas, tal como se muestra en la Figura 7.3. Estas pruebas fueron realizadas para dos velocidades V2 y V3, tal como se muestra en el Anexo D.

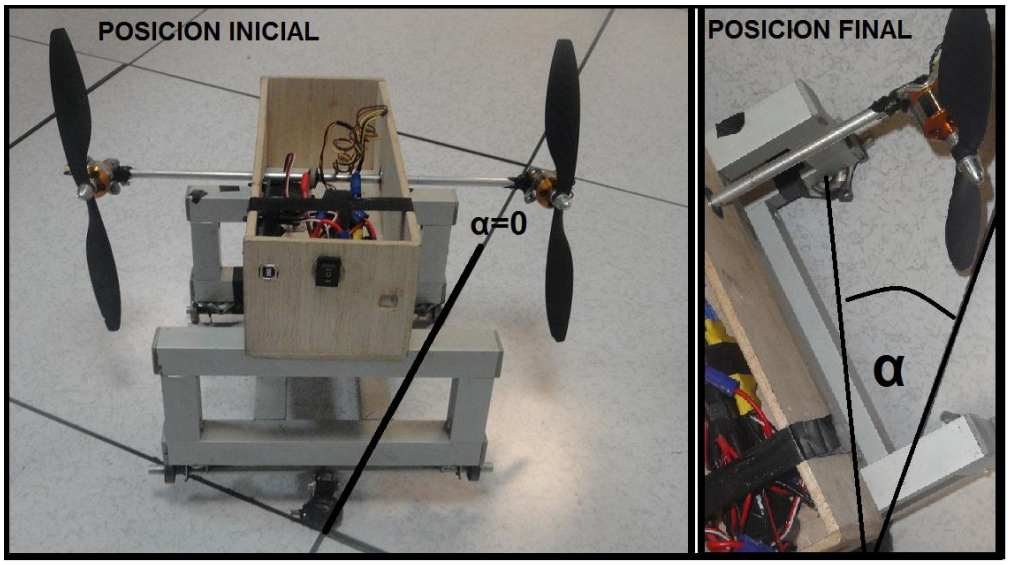



Figura 7.3: Medición del ángulo de giro al activar la dirección derecha.

La prueba de direccionamiento hacia la izquierda es muy similar a la anterior, con la diferencia de que en lugar de activar el motor izquierdo y desactivar el derecho, se activa únicamente el motor derecho. Los datos fueron obtenidos bajo el mismo proceso y con las mismas velocidades. Los códigos utilizados, datos obtenidos y resultados estadísticos calculados se muestran en el Anexo E.

La prueba hacia arriba fue realizada bajo las mismas condiciones que la prueba hacia adelante pero con la diferencia que se ajustó el ángulo del

servo a 135º lo cual provoca que la fuerza que actúa al encender los motores pueda ser descompuesta en dos (una en el plano tierra y otra en dirección hacia arriba). Es importante recalcar que cuando realizamos la prueba hacia adelante solo se generaba una fuera paralela al plano de tierra, por tanto, bajo esta premisa podemos comprobar que existe una fuerza hacia arriba siempre que al activar la dirección la distancia que recorre el carro proporcionalmente menor al cos(45º). Los resultados que se obtuvieron se muestran en el Anexo F, se realizó la prueba exclusivamente en V1, puesto que si demostramos que esto ocurre para V1 podemos inferir que ocurre lo mismo para el resto de velocidades. Adicional y gracias al éxito de esta prueba podemos inferir que lo mismo ocurrirá con la dirección hacia abajo puesto que el ángulo del servo deberá ser acoplado a 45º.

## 7.2 Prueba de recuperación del dirigible.

Aunque el subsistema de recuperación se basa físicamente en los mismos componentes que el subsistema de control de direccionamiento y hayamos determinado en las pruebas anteriores que la parte física de este subsistema funciona correctamente, es necesario determinar de manera cualitativa que el subsistema de recuperación activa los motores de tal manera que genere una fuerza que permita a la góndola

desplazarse hacia abajo con un pequeño giro hacia la derecha, el resultado de estas pruebas se muestra en el Anexo G.

Como vimos en el diseño del subsistema de recuperación este debe generar una fuerza que al ser descompuesta permita el movimiento hacia abajo y hacia la derecha, por tanto en esta prueba se colocó la góndola sobre un soporte con resortes que permitan mantenerla en el aire para observar que estos se estiran al activar el subsistema de recuperación y así validar de manera cualitativa que se genera la fuerza requerida; el resultado de esta prueba fue una pequeña elongación en los resortes que sostienen la góndola y un giro que se observó en la cara frontal de la góndola, al ser una prueba de tipo cualitativa fue realizada únicamente en velocidad mínima y por 10 ocasiones.

Adicionalmente, en vista de que el subsistema de recuperación mantiene activos a los componentes que consumen más energía, se determinó el tiempo máximo que este subsistema puede estar encendido con la batería a plena carga. El código utilizado en esta prueba fue el código completo del proyecto, con el cual para proceder a activar el subsistema de recuperación es necesario únicamente desconectar el *Bluetooth* del teléfono móvil del módulo HC06 de la radio base y por ende puedo mantener activado exclusivamente el

subsistema de recuperación, este código se muestra en el Anexo H. Esta prueba se realizó por cinco ocasiones debido al tiempo que toma realizar una repetición. Los datos obtenidos también se muestran en el Anexo I.

## 7.3 Pruebas en la comunicación.

El canal de comunicación limita en muchos aspectos el funcionamiento del sistema, puesto que el alcance máximo del mismo determina el rango de cobertura sobre el cual se puede controlar la plataforma de vuelo. Para el canal de comunicación se realizó la siguiente prueba: Obtención del alcance máximo del canal de comunicación. Esta prueba se realizó utilizando el software XCTU el cual me permite realizar una prueba de "*lazo de retorno"* sobre el canal, de tal manera que muestra la cantidad y el porcentaje de datos correctos generados en cada prueba; se definió como límite de datos enviados 10000 bytes según se muestra en la Figura 7.4.

Para conseguir el lazo en el canal se creó otra aplicación, la cual reciba datos por *Bluetooth* y los reenvíe por el canal de comunicación hacia el *XBEE* conectado al XCTU. Recalco, que en vista de que la radio solo realiza los procesos de conmutación y traducción, las pruebas sobre el canal de comunicación completo fueron realizadas bajo la premisa de

que la radio base debe ser fijada a una distancia de máximo 1 m con respecto al dispositivo móvil, en este caso la distancia fue de 1 m. Por otro lado el *XBEE* conectado al XCTU a través del Puerto USB de la PC fue movido a diferentes posiciones para determinar la cantidad de datos correctos recibidos a distancias determinadas, tal como se muestra en el Anexo J. Para terminar esta prueba se calculó la confiabilidad del canal para cada distancia como:

$$Confiabilidad = {Datos\ Rx\ correcotos}/{100} \qquad (7.1)$$

Se consideró como aceptable la confiabilidad estándar de todo sistema de comunicación, la cual debe ser mayor o igual a 99.7%; es decir, por debajo de este valor se consideró que no hay cobertura. En base a los datos obtenidos podemos determinar que la plataforma de vuelo puede ser controlada a una distancia máxima de 63 m en línea recta mientras no existan obstrucciones.

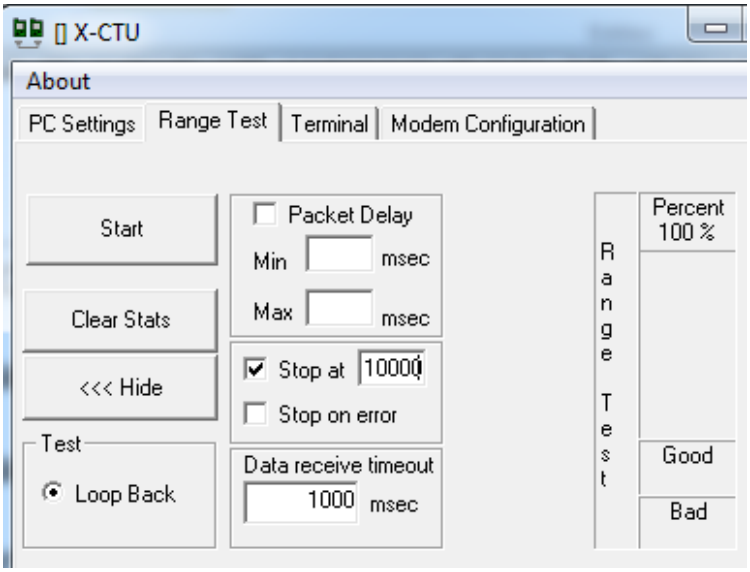



Figura 7.4: Configuración de la prueba de *loop back* en XCTU

## 7.4 Pruebas de la aplicación.

Esta prueba consiste en una encuesta realizada a posibles usuarios y nos permite determinar si la aplicación es fácil de manejar para el usuario final. En la siguiente grafica se muestra la encuesta proporcionada a una muestra de 45 personas. La encuesta mostrada en la Figura 7.5 fue realizada a estudiantes universitarios de la ciudad de Milagro y los datos obtenido se muestran en la el Anexo K. Considerando que el 80 % de los encuestados indicaron que la aplicación fácil de usar, un 13.33 % dijo que era demasiado fácil y un 6.66% indico que es difícil de manejar; podemos afirmar que la aplicación es sencilla de manejar y además que es fácil de entender, dado que los encuestados pudieron realizar todas las tareas de la encuesta mostrada a continuación.

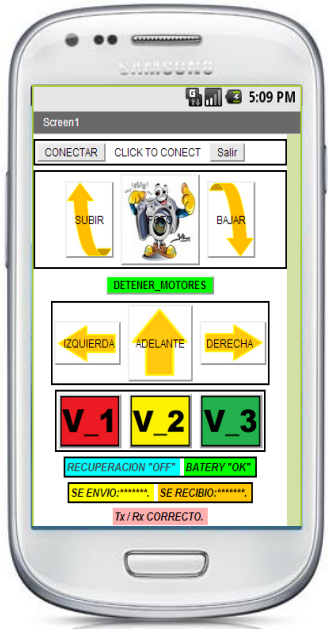
| | | |
|---|---|---|
| Para realizar esta encuesta considere el siguiente evento: Ud. se dispone a controlar una aeronave según se le indique en esta encuesta con la aplicación mostrada en la figura de la derecha. Recuerde que ud solo debe pulsar un botón para que se ejecute la acción requerida. | |  |
| 1 | Conecte el canal de comunicación | |
| 2 | Active la dirección SUBIR | |
| 3 | Active la dirección BAJAR | |
| 4 | Active la dirección ADELANTE | |
| 5 | Active la dirección IZQUIERDA | |
| 6 | Active la dirección DERECHA | |
| 7 | Active la Velocidad 1 = MÍNIMA | |
| 8 | Active la Velocidad 2 = MEDIA | |
| 9 | Active la Velocidad 3 = MÁXIMA | |
| 10 | Desactive los motores | |
| 11 | Active la toma fotográfica | |
| 12 | Salga de la aplicación | |
| Preguntas (complete)<br>En base a las actividades antes realizadas Ud. considera que: | | Extremadamente fácil<br>fácil<br>difícil<br>extremadamente difícil |
| La aplicación es _____ de entender | | |

Figura 7.5: Encuesta para validar facilidad de uso de la aplicación móvil.

# CONCLUSIONES Y RECOMENDACIONES

1. Según pruebas realizadas a los subsistema de control se determinó que al activar la dirección adelante en velocidad mínima durante 5 segundos, la góndola se mueve hacia adelante un promedio de 6.46 m. Esta prueba implica que la góndola se moverá hacia adelante, mas no que se moverá en promedio 6.46 m hacia adelante. Esto es debido a que las pruebas fueron realizadas en el plano (sobre tierra) y por ende bajo condiciones diferentes a las que se tendría ubicando la góndola sobre la plataforma de vuelo. Esto ocurre con todas las demás direcciones, y por ende, podemos concluir que la góndola genera un fuerza a través de los motores para mover al dirigible en el sentido deseado, sin embargo para que esta fuerza supere las fuerzas opuestas al movimiento entre ellas el arrastre, es necesario calibrar el sistema a las condiciones deseadas.

2. Una vez comprobado, gracias a las pruebas antes mencionadas, que los motores son capaces de generar una fuerza en la dirección de

movimiento solicitado (arriba, abajo, derecha, izquierda y adelante), es decir que al activar ambos motores con una pequeña variación de velocidad generaremos un movimiento hacia el lado opuesto al motor con más potencia (en este caso hacia el lado derecho) y además que al colocar una inclinación en el eje de tal manera que los motores apunten hacia abajo. Podemos inferir en función de las pruebas cualitativas (visuales) realizadas sobre el sub-sistema de recuperación que al activar ambos escenarios en conjunto se logró mover la góndola hacia abajo y con pequeño desvío hacia la derecha.

3. El canal de comunicación es el medio de transmisión de las señales de control para direccionar la plataforma de vuelo, por ende, este determina la distancia máxima a la que se puede controlar la plataforma de vuelo sin que se active el sistema de recuperación. Este valor fue determinado en la prueba mostrada en el Anexo J como 63 m de distancia sin obstáculos intermedios.

4. La góndola debe ser colocada sobre un dirigible cuyas dimensiones sean 2.16 m de largo y el ancho 1.11 m según los cálculos realizados en el Anexo A. Es importante recalcar que para que el sistema tenga estabilidad rotacional, el centro de gravedad del fluido desplazado por el dirigible debe estar exactamente arriba del centro de gravedad de la

góndola. Adicional a esto, se debe considerar  a el eje acoplado al servo motor como la parte más crítica en la implementación de la góndola, por tal motivo y para evitar que estos sufran torques estructurales demasiado fuertes que conlleven a desgastes irregulares, desalineación, y por ende un mal funcionamiento del sistema de direccionamiento; es necesario que el eje y los motores acoplados en los extremos se encuentren centrados con respecto a la mediatriz paralela al eje de la de la góndola, por ende en la fabricación del mismo se recomienda utilizar herramientas tales como: nivel, lima, mono vernier (calibrador),papel lija, entre otras.

5. Para  mejoras futuras, es importante recordar que a pesar de que el protocolo *ZigBee* (sobre el cual trabajan los módulos *XBEE*) está diseñado para la transmisión de señales de control, es posible transmitir archivos pesados como imágenes de poca resolución a una baja velocidad utilizando *XBEE*; sin embargo no es recomendable mantener la plataforma de vuelo con el sistema de recuperación activado.

6. Los sistemas de Control están diseñados para trabajar en zonas bajas (20 m.s.n.m.) y donde las corrientes de aire sean suaves, sin embargo es posible que este funcione en zonas altas y con fuertes vientos. Esto es posible gracias a la potencia y velocidad de giro que pueden tener los motores sin escobillas, puesto que los valores ajustados como V1, V2 y

V3 no superna el 70% de las rpm que pueden llegar a dar estos motores; por esto es necesario calibrar variables como velocidad de los motores, dimensiones del dirigible, entre otros, para que el sistema de direccionamiento de la plataforma de vuelo no tripulada pueda trabajar adecuadamente sobre otras condiciones.

7. Es importante recalcar que aunque la aplicación móvil fue catalogada por muchos posibles usuarios como fácil de manejar (80%), es de suma importancia saber que para poder controlar la plataforma de vuelo, el primer paso es activar el canal de comunicación y además que si se presiona el botón desconectar se activara el sub-sistema de recuperación.

# ANEXO A. CÁLCULO DEL VOLUMEN DEL DIRIGIBLE

El cálculo de las dimensiones del dirigible se desarrolla a continuación, para esto nos basamos en la ecuación (2.1) que es la aplicación de la segunda ley de Newton sobre el diagrama de cuerpos libres del sistema mostrado en la Figura 2.1. En el desarrollo se consideró los pesos de los componentes de la góndola mostrados en la Tabla 3.I, los cuales suman 985 g; a esto se adiciona el peso de la corteza del dirigible $W_{Dirigible}$ al cual se le dio un valor de 0.5 kg. Por simplicidad en los cálculos se consideró que el peso máximo del sistema incluido el módulo secundario y la corteza del dirigible es de 1.5 kg que equivalen a 14.7 N.

$$B - W_{Gondola} - W_{He} - W_{Dirigible} = 0$$

$$W_{aire} - W_{Gondola} - W_{He} - W_{Dirigible} = 0; \qquad W = W_{Dirigible} + W_{Gondola}$$

$$m_{aire} \times g - \mathrm{W} - m_{He} \times g = 0; \quad W = 14.7\ N$$

$$\rho_{aire} \times V_{aire} \times g - 14.7 - \rho_{He} \times V_{He} \times g = 0; \qquad V_{aire} = V_{He} = V_{Dirigible}$$

$$\rho_{aire} \times V_{Dirigible} - 1.5 - \rho_{He} \times V_{Dirigible} = 0$$

$$V_{Dirigible} = \frac{1.5}{\rho_{aire} - \rho_{He}}; \qquad \rho_{aire} = 1.293; \quad \rho_{He} = 0{,}1785$$

$$V_{Dirigible} = 1{,}35\ \mathrm{m}^3$$

Hasta el momento tenemos el volumen que debe tener el dirigible para levantar la góndola y tener estabilidad estática, en vista de que es posible ajustar el centro de gravedad de la góndola con el centro de gravedad del dirigible se puede lograr también la estabilidad rotacional.

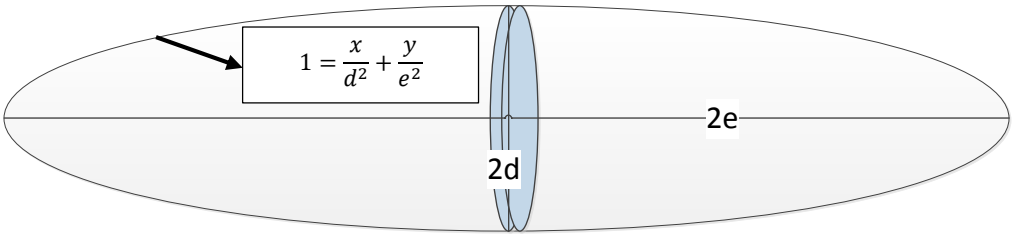



Figura A.1: Forma del dirigible

Continuando con el dimensionamiento del dirigible es necesario determinar las medidas d y e de la elipse rotada en el eje x mostrado en la Figura A.1. Para esto nos basamos la ecuación $1 = \frac{x}{a^2} + \frac{y}{b^2} + \frac{z}{c^2}$ de la cual se obtiene que el volumen de un elipsoide es igual a $V = \frac{4}{3}\,\pi abc$, donde para el caso de nuestra elipse rotada $b = c = d$ y $a = e$. Remplazando se tiene que el volumen de la elipse rotada es $V = \frac{4}{3}\,\pi e d^2$, para este caso en particular se considerará que valor de $e = 2d$, en consecuencia si el volumen del dirigible es 1.35 $m^3$ entonces los valores de $d$ y $e$ son 0.55 y 1.08 respectivamente.

Para finalizar tenemos que el largo del dirigible debe ser 2.16 m y el ancho del dirigible 1.11 m

# ANEXO B. PRUEBA DE ACOPLE EJE - SERVO

A continuación se muestra el código utilizado para validar el acople entre el eje y el servo motor. Este código permite ubicar al eje acopado con el servo motor a las posiciones 0, 90 y 180 grados cíclicamente cada cinco segundos, esto ocurre por un lapso de tiempo de cinco minutos. Una vez trascurridos los 5 segundos el sistema se detiene a la espera de un nuevo dato, esto nos permite medir con un graduador el ángulo de inclinación real que tiene los motores (por software siempre termina en 90º).

```
#include <Servo.h>
#define pinDpwm 7
#define pinServo 8 // PIN AL QUE SE LE ASIGNA LA SEÑAL DE CONTROL.
#define pinIpwm 9
//CALCULO DEL ANGULO DE DESVIACION DEL EJE DESPUES DE QUE EL EJE A PASADO ENTRE LAS POSICIONES 0-90-180 CADA 5 SEGUNDOS DURANTE 5 MINUTOS
Servo myservo;
Servo MotorDerecho;                    // VARIABLE  SERVO PARA CONTROLAR EL MOTOR BURSHLESS DERECHO
Servo MotorIzquierdo;                  // VARIABLE  SERVO PARA CONTROLAR EL MOTOR BURSHLESS IZQUIERDO
unsigned long tiempoI = 0, tiempoF = 0, tiempo = 0;
unsigned int repeat = 0;
int mensage = 0;
boolean iniciar = false;
void setup(){
  Serial.begin(9600);
  myservo.attach(pinServo);
  MotorDerecho.attach(pinDpwm);
  MotorIzquierdo.attach(pinIpwm);
  arm();
  myservo.write(90);
  delay(3000);
```

```
}

void loop(){
  if(Serial.available() > 0){
    mensage = Serial.read();
    if (mensage == '1'){
      iniciar = true;
      MotorDerecho.write(75);
      MotorIzquierdo.write(75);
      repeat++;
    }
  }
  if (iniciar){
    if (repeat <= 150){
      // 5 minutos de constante movimiento entre las 3 posiciones.
      tiempoI = millis();
      while (tiempo < 300000){
        myservo.write(0); // arriba
        delay(5000);
        myservo.write(90); //
        delay(5000);
        myservo.write(180);//abajo
        delay(5000);
        tiempoF = millis();
        tiempo = tiempoF - tiempoI;
      }
      Serial.println(repeat);
      tiempo = 0;
      myservo.write(90); // INICIA EN 90
      MotorDerecho.write(0);
      MotorIzquierdo.write(0);
    }
    iniciar = false;
  }
}
void arm(){
  int speed;
  for(speed = 0; speed <= 75; speed += 5){
    int angle = map(speed, 0, 100, 0, 180);
    MotorDerecho.write(angle);
    MotorIzquierdo.write(angle);
    delay(1000);
  }
}
```

Los resultados de esta prueba se muestran en la Tabla B.I

Tabla B.I: Ángulo final que tuvo el eje después de cada prueba.

| Nº | ϴ | Nº | ϴ | Nº | ϴ | Nº | ϴ | Nº | ϴ |
|---|---|---|---|---|---|---|---|---|---|
| **1** | 89 | 6 | 89 | 11 | 89 | 16 | 89 | 21 | 89 |
| **2** | 89 | 7 | 89 | 12 | 89 | 17 | 89 | 22 | 89 |
| **3** | 89 | 8 | 89 | 13 | 89 | 18 | 89 | 23 | 89 |
| **4** | 89 | 9 | 89 | 14 | 89 | 19 | 89 | 24 | 89 |
| **5** | 89 | 10 | 89 | 15 | 89 | 20 | 89 | 25 | 89 |

# ANEXO C. PRUEBA DE DIRECCIONAMIENTO HACIA ADELANTE

El código utilizado en esta prueba se muestra a continuación. Este permite activar los motores derecho e izquierdo a la misma velocidad durante 5 segundos cuando se reciba una señal de la aplicación móvil, para esta prueba el ángulo del servo motor se fijó en 90º. Los datos obtenidos de esta prueba para las velocidades V1 y V2 se muestran en la Tabla C.I, mientras que los resultados estadísticos de la misma se muestran en la Tabla C.II.

Código para Velocidad 1.

```
#include <Servo.h>
#define pinDpwm 7
#define pinServo 8 // PIN AL QUE SE LE ASIGNA LA SEÑAL DE CONTROL.
#define pinIpwm 9
//medicion de la distancia que recorre la góndola durante 5 s
Servo myservo;
Servo MotorDerecho;  // VARIABLE SERVO PARA CONTROLAR EL MOTOR BURSHLESS DERECHO
Servo MotorIzquierdo;  // VARIABLE SERVO PARA CONTROLAR EL MOTOR BURSHLESS IZQUIERDO
int mensage = 0;
boolean iniciar = false;
void setup(){
  Serial3.begin(9600); // conectar puertos al inverso Rx --> Tx y tx --> Rx
  myservo.attach(pinServo);
  MotorDerecho.attach(pinDpwm);
  MotorIzquierdo.attach(pinIpwm);
  myservo.write(90);
  arm();
  delay(3000);
  velocidad(0);
}
void loop(){
```

```
  if(Serial3.available() > 0){
    mensage = Serial3.read();
    if (mensage == 231){
      iniciar = true;
    }
  }
  if (iniciar){
        velocidad(48);  //V1 = 45
 delay(5000);
        velocidad(0);
            iniciar = false;
}
/*****************************************************************************
* FUNCION_15
* FUNCION QUE PERMITE ARMAR LAS ESC DE LOS MOTORES BURSHLESS
*****************************************************************************/
void arm(){
  int speed;
  for(speed = 0; speed <= 45; speed += 5){
    int angle = map(speed, 0, 100, 0, 180);
    MotorDerecho.write(angle);
    MotorIzquierdo.write(angle);
    delay(1000);
  }
}
void velocidad (int speed){
  int angle;
  angle = map(speed, 0, 100, 0, 180);
  MotorIzquierdo.write(angle);
  MotorDerecho.write(angle);
}
```

Código para Velocidad 2.

```
#include <Servo.h>
#define pinDpwm 7
#define pinServo 8 // PIN AL QUE SE LE ASIGNA LA SEÑAL DE CONTROL.
#define pinIpwm 9
//medicion de la distancia que recorre la góndola durante 5 s
Servo myservo;
Servo MotorDerecho;   // VARIABLE SERVO PARA CONTROLAR EL MOTOR BURSHLESS DERECHO
Servo MotorIzquierdo;   // VARIABLE SERVO PARA CONTROLAR EL MOTOR BURSHLESS IZQUIERDO
```

```
int mensage = 0;
boolean iniciar = false;
void setup(){
  Serial3.begin(9600); // conectar puertos al inverso Rx --> Tx y tx --> Rx
  myservo.attach(pinServo);
  MotorDerecho.attach(pinDpwm);
  MotorIzquierdo.attach(pinIpwm);
  myservo.write(90);
  arm();
  delay(3000);
  velocidad(0);
}
void loop(){
  if(Serial3.available() > 0){
    mensage = Serial3.read();
    if (mensage == 231){
      iniciar = true;
    }
  }
  if (iniciar){
        velocidad(45);  //v1 = 45
 delay(5000);
        velocidad(0);
            iniciar = false;
}
/*****************************************************************************
* FUNCION_15                                                                 *
* FUNCION QUE PERMITE ARMAR LAS ESC DE LOS MOTORES BURSHLESS                 *
*****************************************************************************/
void arm(){
  int speed;
  for(speed = 0; speed <= 45; speed += 5){
    int angle = map(speed, 0, 100, 0, 180);
    MotorDerecho.write(angle);
    MotorIzquierdo.write(angle);
    delay(1000);
  }
}
void velocidad (int speed){
  int angle;
  angle = map(speed, 0, 100, 0, 180);
  MotorIzquierdo.write(angle);
  MotorDerecho.write(angle);
```

}

Tabla C.I: Distancias recorridas hacia adelante en V1 y V2

| | V1 | V2 |
|---|---|---|
| Nº | Z | Z |
| 1 | 5.4 | 7.1 |
| 2 | 5.4 | 7.1 |
| 3 | 5.5 | 7.2 |
| 4 | 5.5 | 7.2 |
| 5 | 5.5 | 7.3 |
| 6 | 5.5 | 7.3 |
| 7 | 5.6 | 7.4 |
| 8 | 5.6 | 7.4 |
| 9 | 5.6 | 7.4 |
| 10 | 5.7 | 7.4 |
| 11 | 5.7 | 7.5 |
| 12 | 5.7 | 7.5 |
| 13 | 5.8 | 7.5 |
| 14 | 5.8 | 7.5 |
| 15 | 5.8 | 7.5 |
| 16 | 5.9 | 7.5 |
| 17 | 5.9 | 7.5 |
| 18 | 5.9 | 7.5 |
| 19 | 5.9 | 7.5 |
| 20 | 6 | 7.5 |
| 21 | 6 | 7.6 |
| 22 | 6 | 7.6 |
| 23 | 6 | 7.7 |
| 24 | 6 | 7.7 |
| 25 | 6 | 8.1 |
| 26 | 6.1 | 8.1 |
| 27 | 6.1 | 8.1 |
| 28 | 61 | 8.1 |
| 29 | 6.1 | 8.1 |
| 30 | 6.1 | 8.1 |
| 31 | 6.2 | 8.2 |
| 32 | 6.2 | 8.2 |
| 33 | 6.2 | 8.3 |
| 34 | 6.2 | 8.3 |
| 35 | 6.3 | 8.3 |
| 36 | 6.3 | 8.3 |
| 37 | 6.3 | 8.3 |
| 38 | 6.3 | 8.3 |
| 39 | 6.4 | 8.4 |
| 40 | 6.4 | 8.4 |
| 41 | 6.4 | 8.4 |
| 42 | 6.4 | 8.4 |
| 43 | 6.4 | 8.4 |
| 44 | 6.5 | 8.4 |
| 45 | 6.5 | 8.5 |
| 46 | 6.5 | 8.5 |
| 47 | 6.6 | 8.5 |
| 48 | 6.6 | 8.5 |
| 49 | 6.6 | 8.5 |
| 50 | 6.6 | 8.5 |
| 51 | 6.7 | 8.6 |
| 52 | 6.8 | 8.6 |
| 53 | 6.8 | 8.6 |
| 54 | 6.8 | 8.7 |
| 55 | 6.9 | 9.4 |
| 56 | 7 | 9.4 |
| 57 | 7 | 9.4 |
| 58 | 7.1 | 9.4 |
| 59 | 7.1 | 9.4 |
| 60 | 7.2 | 9.4 |
| 61 | 7.3 | 9.4 |
| 62 | 7.3 | 9.4 |
| 63 | 7.4 | 9.4 |
| 64 | 7.4 | 9.5 |
| 65 | 7.4 | 9.5 |
| 66 | 7.4 | 9.5 |
| 67 | 7.5 | 9.6 |
| 68 | 7.5 | 9.6 |
| 69 | 7.5 | 9.6 |
| 70 | 7.5 | 9.6 |
| 71 | 7.5 | 9.6 |
| 72 | 7.5 | 9.6 |
| 73 | 7.6 | 9.7 |
| 74 | 7.6 | 9.7 |
| 75 | 7.6 | 9.8 |
| 76 | 77 | 9.9 |

Tabla C.II: Resultados estadísticos de la prueba de direccionamiento hacia Adelante

| | | |
|---|---|---|
| 5.40 | 7.10 | **Min** |
| 7.70 | 9.90 | **Max** |
| 6.46 | 8.41 | **μ** |
| 0.457 | 0.705 | $\sigma^2$ |
| 0.676 | 0.839 | **s** |
| 76.00 | 76.00 | **n** |

# ANEXO D. PRUEBA DE DIRECCIONAMIENTO HACIA LA DERECHA

Este anexo hace referencia al código utilizado en la prueba de direccionamiento hacia la derecha y adicionalmente muestra los datos y los resultados estadísticos obtenidos de estas pruebas.

Código para Velocidad 2.

```
#include <Servo.h>
const unsigned int DERECHA=234, IZQUIERDA=233;
#define pinDpwm 7
#define pinServo 8 // PIN AL QUE SE LE ASIGNA LA SEÑAL DE CONTROL.
#define pinIpwm 9
//medicion de ANGULOS que gira la góndola durante 7 s a la izquierda
Servo myservo;
Servo MotorDerecho;  // VARIABLE SERVO PARA CONTROLAR EL MOTOR DERECHO
Servo MotorIzquierdo;  // VARIABLE SERVO PARA CONTROLAR EL MOTOR IZQUIERDO
int mensage = 0;
boolean dere = false;
void setup(){
  Serial3.begin(9600); // conectar puertos al inverso Rx --> Tx y tx --> Rx
  myservo.attach(pinServo);
  MotorDerecho.attach(pinDpwm);
  MotorIzquierdo.attach(pinIpwm);
  myservo.write(90);
  arm();
  delay(3000);
  velocidad(0);
}
void loop(){
  if(Serial3.available() > 0){
    mensage = Serial3.read();
    if (mensage == IZQUIERDA){
      dere = true;
    }
  }
  if (dere){
```

```
        velocidadI(45);
 delay(10000);
        velocidadI(0);
        dere = false;
  }
}
/*****************************************************************************
* FUNCION_15                                                                 *
* FUNCION QUE PERMITE ARMAR LAS ESC DE LOS MOTORES BURSHLESS                 *
*****************************************************************************/
void arm(){
  int speed;
  for(speed = 0; speed <= 45; speed += 5){
    int angle = map(speed, 0, 100, 0, 180);
    MotorDerecho.write(angle);
    MotorIzquierdo.write(angle);
    delay(1000);
  }
}
void velocidadD(int speed){
  int angle;
  angle = map(speed, 0, 100, 0, 180);
  MotorIzquierdo.write(angle);
  //MotorDerecho.write(0);
}
void velocidad (int speed){
  int angle;
  angle = map(speed, 0, 100, 0, 180);
  MotorIzquierdo.write(angle);
  MotorDerecho.write(angle);
}
```

Código para Velocidad 3.

```
#include <Servo.h>
const unsigned int DERECHA=234, IZQUIERDA=233;
#define pinDpwm 7
#define pinServo 8 // PIN AL QUE SE LE ASIGNA LA SEÑAL DE CONTROL.
#define pinIpwm 9
//medicion de ANGULOS que gira la gondola durante 7 s a la izquierda
Servo myservo;
Servo  MotorDerecho;   // VARIABLE  SERVO PARA CONTROLAR EL MOTOR DERECHO
```

```
Servo MotorIzquierdo;   // VARIABLE  SERVO PARA CONTROLAR EL MOTOR IZQUIERDO
int mensage = 0;
boolean dere = false;
void setup(){
  Serial3.begin(9600); // conectar puertos al inverso Rx --> Tx y tx --> Rx
  myservo.attach(pinServo);
  MotorDerecho.attach(pinDpwm);
  MotorIzquierdo.attach(pinIpwm);
  myservo.write(90);
  arm();
  delay(3000);
  velocidad(0);
}
void loop(){
  if(Serial3.available() > 0){
    mensage = Serial3.read();
    if (mensage == IZQUIERDA){
      dere = true;
    }
  }
  if (dere){
        velocidadI(50);
 delay(10000);
        velocidadI(0);
        dere = false;
  }
}
/*****************************************************************************
* FUNCION_15                                                                 *
* FUNCION QUE PERMITE ARMAR LAS ESC DE LOS MOTORES BURSHLESS                 *
*****************************************************************************/
void arm(){
  int speed;
  for(speed = 0; speed <= 45; speed += 5){
    int angle = map(speed, 0, 100, 0, 180);
    MotorDerecho.write(angle);
    MotorIzquierdo.write(angle);
    delay(1000);
  }
}
void velocidadD(int speed){
  int angle;
```

```
  angle = map(speed, 0, 100, 0, 180);
  MotorIzquierdo.write(angle);
  //MotorDerecho.write(0);
}
void velocidad (int speed){
  int angle;
  angle = map(speed, 0, 100, 0, 180);
  MotorIzquierdo.write(angle);
  MotorDerecho.write(angle);
```

Tabla D.I: Datos obtenidos de la prueba hacia la derecha con V2 y V3

| | v1 | v3 |
|---|---|---|
| Nº | Θ | Θ |
| 1 | 3.3 | 25 |
| 2 | 3.3 | 25 |
| 3 | 3.3 | 25 |
| 4 | 3.6 | 25.1 |
| 5 | 3.6 | 25.1 |
| 6 | 3.6 | 25.1 |
| 7 | 3.9 | 25.3 |
| 8 | 3.9 | 25.3 |
| 9 | 3.9 | 25.3 |
| 10 | 3.9 | 25.5 |
| 11 | 3.9 | 25.5 |
| 12 | 3.9 | 25.5 |
| 13 | 3.9 | 25.9 |
| 14 | 3.9 | 25.9 |
| 15 | 3.9 | 25.9 |
| 16 | 3.9 | 25.9 |
| 17 | 3.9 | 25.9 |
| 18 | 3.9 | 25.9 |
| 19 | 3.9 | 26.3 |
| 20 | 3.9 | 26.3 |
| 21 | 3.9 | 26.3 |
| 22 | 4 | 26.4 |
| 23 | 4 | 26.4 |
| 24 | 4 | 26.4 |
| 25 | 4 | 26.5 |
| 26 | 4 | 26.5 |
| 27 | 4 | 26.5 |
| 28 | 4 | 26.7 |
| 29 | 4 | 26.7 |
| 30 | 4 | 26.7 |
| 31 | 4 | 26.8 |
| 32 | 4 | 26.8 |
| 33 | 4 | 26.8 |
| 34 | 4 | 26.8 |
| 35 | 4 | 26.8 |
| 36 | 4 | 26.8 |
| 37 | 4.1 | 26.8 |
| 38 | 4.1 | 26.8 |
| 39 | 4.1 | 26.8 |
| 40 | 4.1 | 26.8 |
| 41 | 4.1 | 26.8 |
| 42 | 4.1 | 26.8 |
| 43 | 4.5 | 26.8 |
| 44 | 4.5 | 26.8 |
| 45 | 4.5 | 26.8 |
| 46 | 4.5 | 26.8 |
| 47 | 4.5 | 27 |
| 48 | 4.5 | 27 |
| 49 | 4.5 | 27 |
| 50 | 4.5 | 27 |
| 51 | 4.5 | 27.1 |
| 52 | 4.5 | 27.1 |
| 53 | 4.5 | 27.1 |
| 54 | 4.5 | 27.1 |
| 55 | 4.5 | 27.2 |
| 56 | 4.5 | 27.2 |
| 57 | 4.5 | 27.2 |
| 58 | 4.5 | 27.2 |
| 59 | 4.5 | 27.3 |
| 60 | 4.5 | 27.3 |
| 61 | 4.5 | 27.3 |
| 62 | 4.5 | 27.3 |
| 63 | 4.5 | 27.5 |
| 64 | 4.7 | 27.5 |
| 65 | 4.7 | 27.5 |
| 66 | 4.7 | 27.5 |
| 67 | 4.7 | 27.5 |
| 68 | 4.7 | 27.5 |
| 69 | 4.7 | 27.5 |
| 70 | 4.9 | 27.9 |
| 71 | 4.9 | 27.9 |
| 72 | 4.9 | 27.9 |
| 73 | 5 | 27.9 |
| 74 | 5 | 27.9 |
| 75 | 5 | 27.9 |

Tabla D.II: Resultados estadísticos de la prueba hacia la derecha

| | | |
|---|---|---|
| 3.30 | 25.00 | **Min** |
| 5.00 | 27090 | **Max** |
| 4.22 | 26.65 | **μ** |
| 0.166 | 0.65 | $\sigma^2$ |
| 0.407 | 0.80 | **s** |
| 75.00 | 75.00 | **n** |

# ANEXO E. PRUEBA DE DIRECCIONAMIENTO HACIA LA IZQUIERDA

En este anexo se detalla los códigos utilizados en la prueba de direccionamiento hacia la izquierda con los datos obtenidos Tabla E.I a estos datos se los evaluó estadísticamente y se obtuvo como resultado los valores de la Tabla E.II**¡Error! No se encuentra el origen de la referencia.**.

Código para Velocidad 2.

```
#include <Servo.h>
const unsigned int DERECHA=234, IZQUIERDA=233;
#define pinDpwm 7
#define pinServo 8 // PIN AL QUE SE LE ASIGNA LA SEÑAL DE CONTROL.
#define pinIpwm 9
//medicion de ANGULOS que gira la góndola durante 7 s a la izquierda
Servo myservo;
Servo MotorDerecho;  // VARIABLE SERVO PARA CONTROLAR EL MOTOR DERECHO
Servo MotorIzquierdo;  // VARIABLE SERVO PARA CONTROLAR EL MOTOR IZQUIERDO
int mensage = 0;
boolean izq = false;
void setup(){
  Serial3.begin(9600); // conectar puertos al inverso Rx --> Tx y tx --> Rx
  myservo.attach(pinServo);
  MotorDerecho.attach(pinDpwm);
  MotorIzquierdo.attach(pinIpwm);
  myservo.write(90);
  arm();
  delay(3000);
  velocidad(0);
}
void loop(){
  if(Serial3.available() > 0){
    mensage = Serial3.read();
    if (mensage == IZQUIERDA){
      izq = true;
    }
```

```
  }
  if (izq){
        velocidadI(48);
 delay(7000);
        velocidadI(0);
        izq = false;
  }
}
/*******************************************************************************
* FUNCION_15                                                                   *
* FUNCION QUE PERMITE ARMAR LAS ESC DE LOS MOTORES BURSHLESS                   *
*******************************************************************************/
void arm(){
  int speed;
  for(speed = 0; speed <= 45; speed += 5){
    int angle = map(speed, 0, 100, 0, 180);
    MotorDerecho.write(angle);
    MotorIzquierdo.write(angle);
    delay(1000);
  }
}
void velocidadI(int speed){
  int angle;
  angle = map(speed, 0, 100, 0, 180);
  //MotorIzquierdo.write(0);
  MotorDerecho.write(angle);
}
void velocidad (int speed){
  int angle;
  angle = map(speed, 0, 100, 0, 180);
  MotorIzquierdo.write(angle);
  MotorDerecho.write(angle);
}
```

Código para Velocidad 3.

```
#include <Servo.h>
const unsigned int DERECHA=234, IZQUIERDA=233;
#define pinDpwm 7
#define pinServo 8 // PIN AL QUE SE LE ASIGNA LA SEÑAL DE CONTROL.
#define pinIpwm 9
//medicion de ANGULOS que gira la góndola durante 7 s a la izquierda
Servo myservo;
```

```
Servo MotorDerecho;    // VARIABLE  SERVO PARA CONTROLAR EL MOTOR DERECHO
Servo MotorIzquierdo;    // VARIABLE  SERVO PARA CONTROLAR EL MOTOR IZQUIERDO
int mensage = 0;
boolean izq = false;
void setup(){
  Serial3.begin(9600); // conectar puertos al inverso Rx --> Tx y tx --> Rx
  myservo.attach(pinServo);
  MotorDerecho.attach(pinDpwm);
  MotorIzquierdo.attach(pinIpwm);
  myservo.write(90);
  arm();
  delay(3000);
  velocidad(0);
}
void loop(){
  if(Serial3.available() > 0){
    mensage = Serial3.read();
    if (mensage == IZQUIERDA){
      izq = true;
    }
  }
  if (izq){
        velocidadI(50);
 delay(7000);
        velocidadI(0);
        izq = false;
  }
}
/*****************************************************************************
* FUNCION_15                                                                 *
* FUNCION QUE PERMITE ARMAR LAS ESC DE LOS MOTORES BURSHLESS                 *
*****************************************************************************/
void arm(){
  int speed;
  for(speed = 0; speed <= 45; speed += 5){
    int angle = map(speed, 0, 100, 0, 180);
    MotorDerecho.write(angle);
    MotorIzquierdo.write(angle);
    delay(1000);
  }
}
```

```
void velocidadI(int speed){
  int angle;
  angle = map(speed, 0, 100, 0, 180);
  //MotorIzquierdo.write(0);
  MotorDerecho.write(angle);
}
void velocidad (int speed){
  int angle;
  angle = map(speed, 0, 100, 0, 180);
  MotorIzquierdo.write(angle);
  MotorDerecho.write(angle);
}
```

Tabla E.I: Datos obtenidos de la prueba hacia la izquierda con V2 y V3

| | V2 | V3 |
|---|---|---|
| Nº | Θ | Θ |
| 1 | 8.90 | 23.00 |
| 2 | 9.00 | 23.00 |
| 3 | 9.00 | 23.10 |
| 4 | 9.00 | 23.10 |
| 5 | 9.00 | 23.20 |
| 6 | 9.00 | 23.20 |
| 7 | 9.00 | 23.50 |
| 8 | 9.00 | 23.50 |
| 9 | 9.10 | 23.60 |
| 10 | 9.10 | 23.60 |
| 11 | 9.20 | 23.60 |
| 12 | 9.20 | 23.60 |
| 13 | 9.30 | 23.80 |
| 14 | 9.30 | 23.80 |
| 15 | 9.30 | 23.90 |
| 16 | 9.30 | 23.90 |
| 17 | 9.30 | 24.00 |
| 18 | 9.40 | 24.00 |
| 19 | 9.40 | 24.00 |
| 20 | 9.40 | 24.00 |
| 21 | 9.50 | 24.00 |
| 22 | 9.50 | 24.00 |
| 23 | 9.50 | 24.10 |
| 24 | 9.60 | 24.10 |
| 25 | 9.60 | 24.20 |
| 26 | 9.60 | 24.20 |
| 27 | 9.60 | 24.20 |
| 28 | 9.70 | 24.20 |
| 29 | 9.70 | 24.20 |
| 30 | 9.70 | 24.20 |
| 31 | 9.70 | 24.50 |
| 32 | 9.80 | 24.50 |
| 33 | 9.90 | 24.50 |
| 34 | 9.90 | 24.50 |
| 35 | 9.90 | 24.60 |
| 36 | 10.00 | 24.60 |
| 37 | 10.00 | 24.60 |
| 38 | 10.00 | 24.60 |
| 39 | 10.00 | 24.60 |
| 40 | 10.00 | 24.60 |
| 41 | 10.00 | 24.90 |
| 42 | 10.00 | 24.90 |
| 43 | 10.00 | 24.90 |
| 44 | 10.00 | 25.00 |
| 45 | 10.10 | 25.00 |
| 46 | 10.10 | 25.00 |
| 47 | 10.10 | 25.00 |
| 48 | 10.20 | 25.00 |
| 49 | 10.20 | 25.10 |
| 50 | 10.20 | 25.20 |
| 51 | 10.20 | 25.20 |
| 52 | 10.30 | 25.20 |
| 53 | 10.30 | 25.40 |
| 54 | 10.30 | 25.40 |
| 55 | 10.30 | 25.40 |
| 56 | 10.30 | 25.40 |
| 57 | 10.30 | 25.50 |
| 58 | 10.30 | 25.50 |
| 59 | 10.40 | 25.50 |
| 60 | 10.40 | 25.50 |
| 61 | 10.40 | 25.50 |
| 62 | 10.40 | 25.50 |
| 63 | 10.40 | 25.50 |
| 64 | 10.50 | 25.50 |
| 65 | 10.50 | 25.50 |
| 66 | 10.50 | 25.50 |
| 67 | 10.50 | 25.60 |
| 68 | 10.60 | 25.60 |
| 69 | 10.60 | 25.70 |
| 70 | 10.60 | 25.70 |
| 71 | 10.70 | 25.80 |
| 72 | 10.70 | 25.80 |
| 73 | 10.70 | 25.80 |
| 74 | 10.70 | 25.80 |
| 75 | 10.70 | 25.80 |

Tabla E.II: Resultados estadísticos de prueba hacia la izquierda.

| | | |
|---|---|---|
| 8.90 | 23.00 | **Min** |
| 10.70 | 25.80 | **Max** |
| 9.88 | 24.63 | **μ** |
| 0.291 | 0.68 | $\boldsymbol{\sigma^2}$ |
| 0.539 | 0.83 | **s** |
| 75.00 | 75.00 | **n** |

# ANEXO F. PRUEBA DE DIRECCIONAMIENTO HACIA LA ARRIBA

Aquí se muestran los datos obtenidos de las pruebas realizadas cargando el código 1 mostrado en el Anexo B con la diferencia que la línea *myservo.write(90);* debe ser remplazada por *myservo.write(135);* estas pruebas nos permiten determinar si la dirección de la fuerza que ejercen los motores puede ser descompuesta en una hacia arriba y otra hacia abajo con la finalidad de subir al dirigible al mismo tiempo que avanza hacia adelante.

Código para Velocidad 1.

```
#include <Servo.h>
#define pinDpwm 7
#define pinServo 8 // PIN AL QUE SE LE ASIGNA LA SEÑAL DE CONTROL.
#define pinIpwm 9
//medicion de la distancia que recorre la góndola durante 5s
Servo myservo;
Servo MotorDerecho;  // VARIABLE SERVO PARA CONTROLAR EL MOTOR DERECHO
Servo MotorIzquierdo;  // VARIABLE SERVO PARA CONTROLAR EL MOTOR IZQUIERDO
int mensage = 0;
boolean iniciar = false;
void setup(){
  Serial3.begin(9600); // conectar puertos al inverso Rx --> Tx y tx --> Rx
  myservo.attach(pinServo);
  MotorDerecho.attach(pinDpwm);
  MotorIzquierdo.attach(pinIpwm);
  myservo.write(90);
  arm();
  delay(3000);
  velocidad(0);
}
void loop(){
  if(Serial3.available() > 0){
```

```
    mensage = Serial3.read();
    if (mensage == 231){
      iniciar = true;
    }
  }
  if (iniciar){
        velocidad(48);  //V1 = 45
 delay(5000);
        velocidad(0);
            iniciar = false;
}
/*****************************************************************************
* FUNCION_15                                                                 *
* FUNCION QUE PERMITE ARMAR LAS ESC DE LOS MOTORES BURSHLESS                 *
*****************************************************************************/
void arm(){
  int speed;
  for(speed = 0; speed <= 45; speed += 5){
    int angle = map(speed, 0, 100, 0, 180);
    MotorDerecho.write(angle);
    MotorIzquierdo.write(angle);
    delay(1000);
  }
}
void velocidad (int speed){
  int angle;
  angle = map(speed, 0, 100, 0, 180);
  MotorIzquierdo.write(angle);
  MotorDerecho.write(angle);
}
```

Tabla F.I: Datos obtenidos de la prueba hacia la Izquierda

| | V1 |
|---|---|
| Nº | X |
| 1 | 4.37 |
| 2 | 4.42 |
| 3 | 4.43 |
| 4 | 4.44 |
| 5 | 4.46 |
| 6 | 4.47 |
| 7 | 4.49 |
| 8 | 4.5 |
| 9 | 4.5 |
| 10 | 4.5 |
| 11 | 4.52 |
| 12 | 4.52 |
| 13 | 4.52 |
| 14 | 4.52 |
| 15 | 4.53 |
| 16 | 4.54 |

| | |
|---|---|
| **17** | 4.55 |
| **18** | 4.6 |
| **19** | 4.6 |
| **20** | 4.61 |
| **21** | 4.62 |
| **22** | 4.66 |
| **23** | 4.66 |

| | |
|---|---|
| **24** | 4.68 |
| **25** | 4.71 |
| **26** | 4.73 |
| **27** | 4.76 |
| **28** | 4.77 |
| **29** | 4.77 |
| **30** | 4.78 |

Tabla F.II: Resultado estadístico de la prueba hacia arriba.

| | |
|---|---|
| 4.37 | **Min** |
| 4.78 | **Max** |
| 4.57 | **μ** |
| 0.014 | $\sigma^2$ |
| 0.117 | **s** |
| 30.00 | **n** |

Definiendo

$$\text{Z: distancia recorrida hacia adelante} = 6.46$$

$$\theta\text{: Angulo con que se descompon las fueras} = 45º$$

$$\text{X: distancia recorrida con } \theta = 45\ º$$

Por tanto:

$$\rightarrow \text{X} = \text{Z} \times \cos 45º = 4.5679$$

En vista de que la media de los datos muéstrales obtenidos en esta prueba es aproximadamente igual al valor obtenido matemáticamente, podemos afirmar que la prueba fue exitosa.

# ANEXO G. PRUEBA PARA EL SUB-SISTEMA DE RECUPERACIÓN

La prueba de direccionamiento del subsistema de control es una prueba cualitativa que me permite determinar la dirección del movimiento de la góndola al activarse el subsistema de recuperación, es decir validar los motores son capases de generar una fuerza que descompuesta tenga dirección hacia abajo y hacia la derecha. Recalco que esta prueba no permite cuantificar la magnitud de esta fuerza. Los resultados de esta prueba son mostrados en la Tabla G.I.

Tabla G.I : Resultado de la prueba de dirección en estado recuperación

| | MOVIMIENTO | |
|---|---|---|
| **Nº** | **↓** | **→** |
| **1** | SI | SI |
| **2** | SI | SI |
| **3** | SI | SI |
| **4** | SI | SI |
| **5** | SI | SI |
| **6** | SI | SI |
| **7** | SI | SI |
| **8** | SI | SI |
| **9** | SI | SI |
| **10** | SI | SI |

# ANEXO H. CÓDIGO UNIFICADO DEL SISTEMA DE CONTROL DE DIRECCIONAMIENTO

En este Anexo se muestra el código completo desarrollado para el microcontrolador *Arduino Mega*. El código se divide en cuatro partes: declaración de librerías, constantes y variables; inicialización de variables y puertos; desarrollo del código principal; y desarrollo de funcionalidades específicas del sistema.

```
#include <Servo.h>
/*****************************
* DECLARACION DE CONSTANTES *
*****************************/
//DATOS DISPONIBLES PRA RECIBIR
#define ARRIBA 251
#define ABAJO 237
#define PARAR 235
#define DERECHA 234
#define IZQUIERDA 233
#define ADELANTE 236
#define V_MINIMO 254
#define V_NORMAL 253
#define V_MAXIMO 252
#define DESCONECTADO 232
#define AUTENTICACION 255
#define FOTO 231
#define TETA_120 250
#define TETA_115 249
#define TETA_110 248
#define TETA_105 247
#define TETA_100 246
#define TETA_95 245
#define TETA_90 244
#define TETA_85 243
```

```
#define TETA_80 242
#define TETA_75 241
#define TETA_70 240
#define TETA_65 239
#define TETA_60 238
#define DATA_ERROR 230
#define RECUPERACION_ON 229
#define RECUPERACION_OFF 228
// Velocidad de Transmisión
#define Vtransmit 115200
#define VtransmitControl 9600
//PINES POR LOS QUE SE GENERA LA SENAL PWM DE CONTROL DE VELOCIDAD.
#define pinDpwm 7
#define pinServo 8 // PIN AL QUE SE LE ASIGNA LA SEÑAL DE CONTROL.
#define pinIpwm 9
#define pinExterno 30

/************************************
* DECLARACION DE VARIABLES GLOBALES *
************************************/
byte uart_rd1;    // ALMACENA DATO RECIBIDO VIA UART.
byte dataFoto;   // ALMACENA DATO RECIBIDO VIA UART exclusivos del modulo adicional.
Servo myServo;              // VARIABLE  SERVO PARA CONTROLAR EL SERVOMOTOR.
Servo MotorDerecho;                                          // VARIABLE SERVO PARA CONTROLAR EL MOTOR BURSHLESS DERECHO
Servo MotorIzquierdo;                              // VARIABLE  SERVO PARA CONTROLAR EL MOTOR BURSHLESS IZQUIERDO
unsigned int angulo = 90;     // ANGULO QUE SE ASIGNA AL SERVO.

//VARIABLES QUE PERMITEN EL CONTROL PWM.
unsigned int temp=0, velocidad=0; // variables que representan el porcentaje de velocidad y duty cicle respectivamente

//contadores para datos recibidos y erróneos.
unsigned long contadorRx = 0, contadorEr = 0;
// contadores de tiempo.
unsigned long tiempoI = 0, tiempoF = 0, tiempo = 0;

//BANDERAS BOOLEANAS
boolean banderaRecuperacion = false;
boolean banderaTiempo = true;
```

```
boolean datoEsRecibido = false;
boolean stopMBdere = true;
boolean stopMBizquier = true;
boolean canalOcupado = true;
boolean banderaV1 = true;
boolean banderaV2 = false;
boolean banderaV3 = false;

/************************************************
* INICIALIZACION DE PUERTOS, REGISTROS Y VARIABLES *
************************************************/
void setup(){
   Serial1.begin(115200);            // INICIA EL PUERTO SERIE 1 A 115200
bps para modulo adicional.
   Serial2.begin(9600);             // INICIA EL PUERTO SERIE 2 A 115200 bps
para XBEE.
   Serial.begin(VtransmitControl);       // INICIA EL PUERTO SERIE 0 A 9600
bps para control.
   myServo.attach(pinServo);            // ASIGNA CONTROL DEL SERVO EN
EL PIN pinServo.
   MotorDerecho.attach(pinDpwm);
   MotorIzquierdo.attach(pinIpwm);
   Mover_Parar();
   pinMode(pinExterno,INPUT);
}

/************************************
* DESARROLLO DE LA FUNCION PRINCIPAL *
************************************/
void loop(){
  if (Serial2.available() > 0){       // CUESTIONA SI EXISTEN BYTE
DISPONIBLES PARA LEER.
   uart_rd1 = Serial2.read();        // ESCRIBE EL DATO EN uart_rd1.
   Serial1.write(uart_rd1);
   banderaTiempo = true;
   datoEsRecibido = true;
   //CONTRAR  EL DATO RECIBIDO.
   contadorRx = contadorRx + 1;
   //actualizarDisplay(contadorRx, contadorEr); // ACTUALICE EL DISPLAY
   //DESACTIVE BANDERA DE RECUPERACION PUESTO QUE YA SE
ACTIVO DATO.
   if ( banderaRecuperacion){
     banderaRecuperacion = false;
```

```
      Serial2.write(0x0A);      //ENVIA SEÑAL DE DESACTIVACION DE
SISTEMA DE RECUPERACION.
      Activar_VMinimo();
      Mover_Adelante();
    }
    //SI EL DATO PERTENECE AL SUBSISTEMA DE CONTROL DE
COMUNICACION.
    if (uart_rd1 == DESCONECTADO || uart_rd1 == AUTENTICACION){
      Control_Comunicacion(uart_rd1);
    }

    //SI EL DATO PERTENECE AL SUBSISTEMA DE CONTROL DE
DIRECCIONAMIENTO.
    else if (uart_rd1 == ARRIBA || uart_rd1 == ABAJO || uart_rd1 == PARAR ||
uart_rd1 == DERECHA || uart_rd1 == IZQUIERDA || uart_rd1 ==
ADELANTE){
      Control_Direccionamiento(uart_rd1);
    }

    //SI EL DATO PERTENECE AL SUBSISTEMA DE CONTROL DE
VELOCIDAD.
    else if (uart_rd1 == V_MINIMO  || uart_rd1 ==  V_NORMAL || uart_rd1 ==
V_MAXIMO){
      Control_Velocidad(uart_rd1);
    }
    //SI EL DATO PERTENECE A LA TOMA FOTOGRAFICA
    else if (uart_rd1 == FOTO){
                        Serial.print("FOTO.");
      while(digitalRead(pinExterno) == LOW){
                          while (Serial1.available() > 0){       //CUESTIONA SI
EXISTEN BYTE DISPONIBLES PARA LEER.
dataFoto = Serial1.read();
Serial2.write(dataFoto);
Serial.print("-");
Serial.print(dataFoto, HEX);
Serial.print("-");
}
while (Serial2.available() > 0){       //CUESTIONA SI EXISTEN BYTE
dataFoto = Serial2.read();
Serial1.write(dataFoto);
Serial.print("*");
Serial.print(dataFoto, HEX);
Serial.print("*");
}
```

```
     }
                    Serial.print("FIN-FOTO.");
   }
   else { //SI EL DATO RECIBIDO ES INCORECTO.
     Serial2.write(0xEE);    //ENVIA SEÑAL DE QUE EL DATO ES NO
ESPERADO.
                    Serial.print("ERROR.");
     contadorEr=contadorEr + 1;  //INCREMENTAR CONTADOR ERRONEO.
   }
 }
 else{//SI NO SE RECIBE DATO
 //CONTAR EL TIEMPO EN QUE NO SE RECIBE DATOS.
   if (datoEsRecibido){
      if (banderaTiempo){
         tiempoI = millis(); //inicio del tiempo en que no se rrecibe dato
         banderaTiempo = false;
      }
      tiempoF = millis(); // tiempo actual
      tiempo = tiempoF - tiempoI;        // intervalo de tiempo en el que no se
recibe datos.
      if (tiempo >= 10000){    // si han pasado mas de 10 s.
         if (!banderaRecuperacion){
                        Control_Recuperacion();      // active recuperacion.
                    }
      }
                }
 }
 uart_rd1=0;
}
/*****************************************************************************
* FUNCION_1                                                                  *
* Control_Comunicacion RECIBE UN DATO DE 8 BITS GENERADO POR
LA APLICACION MOVIL,*
* LO ANALIZA, ACTIVA EL SUBSISTEMA DE RECUPERACION EN CASO
DE SER NECESARIO Y    *
* ADEMAS REENVIA EL MISMO DATO  POR EL PUERTO SERIAL.
*
*****************************************************************************/
void Control_Comunicacion(int dato){
 switch (dato){
   case DESCONECTADO:
   Serial2.write(DESCONECTADO);
   Control_Recuperacion();
   break;
```

```
    case AUTENTICACION:
    Serial2.write(AUTENTICACION);
    break;
    default:
    break;
  }
}
/*********************************************************************************
* FUNCION_2                                                                      *
* Control_Direccionamiento RECIBE UN DATO DE 8 BITS GENERADO POR LA APLICACION MOVIL,*
* LO ANALIZA, Y LLAMA A FUNCION NECESARIA PARA GENERAR EL MOVIMIENTO SOLICITADO,     *
*  ADEMAS REENVIA EL MISMO DATO POR EL PUERTO SERIAL.
*
*********************************************************************************/
void Control_Direccionamiento(int dato){
  switch (dato){
    case ARRIBA:
      Mover_Arriba();
    break;
    case ABAJO:
      Mover_Abajo();
    break;
    case DERECHA:
      Mover_Derecha();
      Serial2.write(DERECHA);
    break;
    case IZQUIERDA:
      Mover_Izquierda();
                    Serial2.write(IZQUIERDA);
    break;
    case ADELANTE:
      Mover_Adelante();
                    Serial2.write(ADELANTE);
    break;
    case PARAR:
      Mover_Parar();
                    Serial2.write(PARAR);
    break;
    default:
    break;
  }
```

```
}
/*********************************************************************************
* FUNCION_3                                                                      *
* Control_Velocidad RECIBE UN DATO DE 8 BITS GENERADO POR LA APLICACION MOVIL  *
* LO ANALIZA, Y LLAMA A FUNCION NECESARIA PARA AJUSTAR LA VELOCIDAD
* REENVIA EL MISMO DATO POR EL PUERTO SERIAL.
*
*********************************************************************************/
void Control_Velocidad(int dato){
  switch (dato){
    case V_MINIMO:
      Activar_VMinimo();
      Serial2.write(V_MINIMO);
                    Serial.print("V_MINIMO.");
    break;
    case V_NORMAL:
      Activar_VNormal();
      Serial2.write(V_NORMAL);
                    Serial.print("V_NORMAL.");
    break;
    case V_MAXIMO:
      Activar_VMaximo();
      Serial2.write(V_MAXIMO);
                    Serial.print("V_MAXIMO.");
    break;
    default:
    break;
  }
}
/*************************************************
* FUNCION_4                                      *
* ACCIONA EL MOVIMIENTO DEL DIRIGIBLE HACIA ARRIBA *
*************************************************/
void Mover_Arriba(){
  int tmp;
  armMotors();
  tmp = Variacion_servo(angulo,+5);
  angulo = Mover_Servo(angulo,tmp);// CAMBIANDO LA POSICION DEL SERVO
}
/*************************************************
* FUNCION_5                                      *
```

```
* ACCIONA EL MOVIMIENTO DEL DIRIGIBLE HACIA ABAJO  *
************************************************/
void Mover_Abajo(){
  int tmp;
  armMotors();
  tmp = Variacion_servo(angulo,-5);
  angulo = Mover_Servo(angulo,tmp);//CAMBIANDO LA POSICION DEL
SERVO
}
/************************************************
* FUNCION_6                                     *
* ACCIONA EL MOVIMIENTO DEL DIRIGIBLE HACIA DERECHA*
************************************************/
void Mover_Derecha(){
  armDere();
  angulo = Mover_Servo(angulo, 90);
  MotorDerecho.write(0);//MOTOR DERECHO DETENIDO
}
/*****************************************************
* FUNCION_7                                      *
* ACCIONA EL MOVIMIENTO DEL DIRIGIBLE HACIA IZQUIERDA *
*****************************************************/
void Mover_Izquierda(){
  armIzquier();
  angulo = Mover_Servo(angulo, 90);
  MotorIzquierdo.write(0);//MOTOR IZQUIERDO DETENIDO
}
/*****************************************************
* FUNCION_8                                      *
* ACCIONA EL MOVIMIENTO DEL DIRIGIBLE HACIA ADELANTE *
*****************************************************/
void Mover_Adelante(){
  armMotors();
  angulo = Mover_Servo(angulo, 90);
}
/*****************************************************************
* FUNCION_9                                                      *
* ACCIONA EL MINIMO MOVIMIENTO DEL DIRIGIBLE SIMULANDO
DETENERSE *
*****************************************************************/
void Mover_Parar(){
  angulo = Mover_Servo(angulo, 90);
  MotorIzquierdo.write(0);//MOTOR IZQUIERDO DETENIDO
  MotorDerecho.write(0);//MOTOR DERECHO DETENIDO
```

```
}
/*********************************************************
* FUNCION_10                                             *
* ACCIONA LA MINIMA VELOCIDAD DE ROTACION DE LOS MOTORES   *
*********************************************************/
void Activar_VMinimo(){
  temp = 45;              //PORCENTAJE DE VELOCIDAD
  velocidad = map (temp, 0, 100, 0, 180);    //MAPEO ENTRE % DE
VELOCIDAD Y DUTY CICCLE.
  MotorDerecho.write(velocidad);
  MotorIzquierdo.write(velocidad);
  banderaV1=true;
  banderaV2=false;
  banderaV3=false;
}
/*********************************************************
* FUNCION_11                                             *
* ACCIONA LA VELOCIDAD MEDIA DE ROTACION DE LOS MOTORES    *
*********************************************************/
void Activar_VNormal(){
  temp = 48;              //PORCENTAJE DE VELOCIDAD
  velocidad = map(temp, 0, 100, 0, 180);    //MAPEO ENTRE % DE
VELOCIDAD Y DUTY CICCLE.
  MotorDerecho.write(velocidad);
  MotorIzquierdo.write(velocidad);
  banderaV2=true;
  banderaV1=false;
  banderaV3=false;
}
/*********************************************************
* FUNCION_12                                             *
* ACCIONA LA MAXIMA VELOCIDAD DE ROTACION DE LOS MOTORES
*
*********************************************************/
void Activar_VMaximo(){
  temp = 50;              //PORCENTAJE DE VELOCIDAD
  velocidad = map (temp, 0, 100, 0, 180);    //MAPEO ENTRE % DE
VELOCIDAD Y DUTY CICCLE.
  MotorDerecho.write(velocidad);
  MotorIzquierdo.write(velocidad);
  banderaV3=true;
  banderaV1=false;
  banderaV2=false;
}
```

```
/*****************************************************************************
* FUNCION_13                                                                 *
* DEFINE LOS PARAMETROS QUE DEBEN ESTAR ENCENDIDOS CON PERDIDA DE LA *
* COMUNICACION                                                               *
*****************************************************************************/
void Control_Recuperacion(){
  Serial2.write(0xAA);    //ENVIA SEÑAL DE ACTIVACION DE SISTEMA DE RECUPERACION.
  Serial.print("RECUPERACION-ON.");
  armMotors();
  temp = 45;
  MotorDerecho.write(map (temp, 0, 100, 0, 180));
  delay(100);
  temp = 48;
  MotorIzquierdo.write(map (temp, 0, 100, 0, 180));
  delay(100);
  banderaRecuperacion = true;
  angulo = Mover_Servo(angulo, 50);
}
/*****************************************************************************
* FUNCION_15                                                                 *
* FUNCION QUE PERMITE ARMAR LAS ESC DE LOS MOTORES BURSHLESS                 *
*****************************************************************************/
void armMotors(){
  if (stopMBdere || stopMBizquier){
    int speed;
    for(speed = 0; speed <= 45; speed += 5){
      int angle = map(speed, 0, 100, 0, 180);
      MotorDerecho.write(angle);
      MotorIzquierdo.write(angle);
      delay(1000);
    }
    stopMBdere = false;
    stopMBizquier = false;
  }
  CargarVelocidad();
}
/*****************************************************************************
* FUNCION_16                                                                 *
* FUNCION QUE PERMITE ARMAR LA ESC DEL MOTOR BURSHLESS DERECHO
*****************************************************************************/
```

```
void armDere(){
  if (stopMBdere){
    int speed;
    for(speed = 0; speed <= 45; speed += 5){
      int angle = map(speed, 0, 100, 0, 180);
      MotorDerecho.write(angle);
      delay(1000);
    }
    stopMBdere = false;
  }
  CargarVelocidad();
}

/*****************************************************************************
* FUNCION_17                                                                 *
* FUNCION QUE PERMITE ARMAR LA ESC DEL MOTOR BURSHLESS IZQUIERDO
*****************************************************************************/
void armIzquier(){
  if(stopMBizquier){
    int speed;
    for(speed = 0; speed <= 45; speed += 5){
      int angle = map(speed, 0, 100, 0, 180);
      MotorIzquierdo.write(angle);
      delay(1000);
    }
    stopMBizquier = false;
  }
  CargarVelocidad();
}
/*****************************************************************************
* FUNCION_19                                                                 *
* FUNCION QUE PERMITE CARGAR LA VELOCIDAD DE TRABAJO ACTUAL                  *
*****************************************************************************/
void CargarVelocidad(){
  if(banderaV1){
    Activar_VMinimo();
  }
  else if(banderaV2){
                    Activar_VNormal();
  }
  else if(banderaV3){
                    Activar_VMaximo();
```

```
  }
}
/*****************************************************************************
* FUNCION_20                                                                 *
* Variacion_servo RECIBE DOS DATO SDE 8 BITS GENERADOS POR LA APLICACION MOVIL  *
* LO ANALIZA, Y LLAMA A FUNCION NECESARIA PARA AJUSTAR LA VELOCIDAD *
* ADICIONAL REENVIA EL MISMO DATO POR EL PUERTO SERIAL.
*
*****************************************************************************/
int Variacion_servo(int AnguloActual, int movimiento ){
int dato;
  switch (AnguloActual){
    case 60:
      if (movimiento > 0){
        dato = AnguloActual + movimiento;
      }else{ dato = AnguloActual; }
    break;
    case 65:
      dato = AnguloActual + movimiento;
    break;
    case 70:
      dato = AnguloActual + movimiento;
    break;
    case 75:
      dato = AnguloActual + movimiento;
    break;
    case 80:
      dato = AnguloActual + movimiento;
    break;
    case 85:
      dato = AnguloActual + movimiento;
    break;
    case 90:
      dato = AnguloActual + movimiento;
    break;
    case 95:
      dato = AnguloActual + movimiento;
    break;
    case 100:
      dato = AnguloActual + movimiento;
    break;
    case 105:
```

```
      dato = AnguloActual + movimiento;
    break;
    case 110:
      dato = AnguloActual + movimiento;
    break;
    case 115:
      dato = AnguloActual + movimiento;
    break;
    case 120:
      if (movimiento < 0){
        dato=  AnguloActual + movimiento;
      }else{ dato = AnguloActual; }
    break;
    default:
      dato = AnguloActual;
    break;
  }
  Serial2.write(((dato-60)/5)+238);
  return dato;
}
/*************************************************
* FUNCION_n                                      *
* ACCIONA EL MOVIMIENTO DEL DIRIGIBLE HACIA DERECHA*
*************************************************/
int Mover_Servo(int AnguloActual, int NuevoAngulo){
  int i;
  if (AnguloActual <= NuevoAngulo){
    for (i = AnguloActual; i <= NuevoAngulo; i += 1){
      myServo.write(i);
      delay(500);
      Serial.println(i);
    }
  }
  else if (AnguloActual > NuevoAngulo){
    for (i = AnguloActual; i >= NuevoAngulo; i -= 1){
      myServo.write(i);
      delay(500);
      Serial.println(i);
    }
  }
  return NuevoAngulo; }
```

# ANEXO I. PRUEBA DE DURACIÓN DE LA BATERÍA CON SUB-SISTEMA DE RECUPERACIÓN ACTIVO

Al mantener activado el sistema de recuperación iniciando con la batería a plena carga, se obtuvieron los siguientes datos:

Tabla I.I: Resultados de la prueba duración de batería

| Nº | V inicial | DURACION (min) |
|---|---|---|
| 1 | 11.61 | 28 |
| 2 | 11.45 | 25 |
| 3 | 11.32 | 23 |
| 4 | 11.56 | 27 |
| 5 | 11.48 | 25 |

# ANEXO J. PRUEBA DE ALCANCE DEL CANAL DE COMUNICACIÓN.

Los datos obtenidos de la prueba realizada en el canal de comunicación se muestran a continuación explicada en el Capítulo 7.3.

Tabla J.I: Resultados de pruebas de alcance sobre el canal de comunicación.

| Ubicación de la góndola | | | Estado del enlace | |
|---|---|---|---|---|
| **Altura (h)** | **Distancia (x)** | **Z** | **Datos $R_x$ correctos** | **Confiabilidad %** |
| 2.80 | 30 | 30.13 | 10000 | 100% |
| 2.80 | 35 | 35.11 | 10000 | 100% |
| 2.80 | 40 | 40.10 | 10000 | 100% |
| 2.80 | 45 | 45.09 | 10000 | 100% |
| 2.80 | 50 | 50.08 | 10000 | 100% |
| 2.80 | 55 | 55.07 | 10000 | 100% |
| 2.80 | 60 | 60.07 | 9999 | 99.9% |
| 2.80 | 61 | 61.06 | 10000 | 100% |
| 2.80 | 62 | 62.06 | 9999 | 99.99% |
| 2.80 | 63 | 63.06 | 9998 | 99.98% |
| 2.80 | 64 | 64.06 | 9997 | 99.97% |
| 2.80 | 65 | 65.06 | 9996 | 99.96% |
| 2.80 | 66 | 66.06 | 9996 | 99.96% |
| 2.80 | 67 | 67.06 | 9996 | 99.96% |
| 2.80 | 68 | 68.06 | 9995 | 99.95% |
| 2.80 | 69 | 69.06 | 9996 | 99.96% |
| 2.80 | 70 | 70.06 | 9995 | 99.95% |
| 5.60 | 30 | 30,52 | 10000 | 100% |
| 5.60 | 35 | 35,45 | 10000 | 100% |
| 5.60 | 40 | 40,39 | 10000 | 100% |
| 5.60 | 45 | 45,35 | 10000 | 100% |
| 5.60 | 50 | 50,31 | 10000 | 100% |
| 5.60 | 55 | 55,28 | 10000 | 100% |
| 5.60 | 56 | 56,28 | 10000 | 100% |
| 5.60 | 57 | 57,27 | 10000 | 100% |
| 5.60 | 58 | 58,27 | 10000 | 100% |
| 5.60 | 59 | 59,27 | 9999 | 99.99% |

| | | | | |
|---|---|---|---|---|
| 5.60 | 60 | 60,26 | 9998 | 99.98% |
| 5.60 | 61 | 61,26 | 9998 | 99.98% |
| 5.60 | 62 | 62,25 | 9997 | 99.97% |
| 5.60 | 63 | 63,25 | 9996 | 99.96% |
| 5.60 | 64 | 64,24 | 9996 | 99.96% |
| 5.60 | 65 | 65,24 | 9995 | 99.95% |
| 5.60 | 66 | 66,24 | 9995 | 99.95% |
| 5.60 | 67 | 67,23 | 9995 | 99.95% |

# ANEXO K. PRUEBA DE LA APLICACIÓN MÓVIL.

Los resultados de la prueba explicada en el subcapítulo 7.4 basados en la encuesta mostrada en la Figura 7.5 se muestran a continuación.

Tabla K.I: Resultados de la encuesta Figura 7.5

| | Aplicación |
|---|---|
| Nº | calificativo |
| 1 | difícil |
| 2 | difícil |
| 3 | difícil |
| 4 | Extremadamente fácil |
| 5 | Extremadamente fácil |
| 6 | Extremadamente fácil |
| 7 | Extremadamente fácil |
| 8 | Extremadamente fácil |
| 9 | Extremadamente fácil |
| 10 | fácil |
| 11 | fácil |
| 12 | fácil |
| 13 | fácil |
| 14 | fácil |
| 15 | fácil |
| 16 | fácil |
| 17 | fácil |
| 18 | fácil |
| 19 | fácil |
| 20 | fácil |
| 21 | fácil |
| 22 | fácil |
| 23 | fácil |
| 24 | fácil |
| 25 | fácil |
| 26 | fácil |
| 27 | fácil |
| 28 | fácil |
| 29 | fácil |
| 30 | fácil |
| 31 | fácil |
| 32 | fácil |
| 33 | fácil |
| 34 | fácil |
| 35 | fácil |
| 36 | fácil |
| 37 | fácil |
| 38 | fácil |
| 39 | fácil |
| 40 | fácil |
| 41 | fácil |
| 42 | fácil |
| 43 | fácil |
| 44 | fácil |
| 45 | fácil |

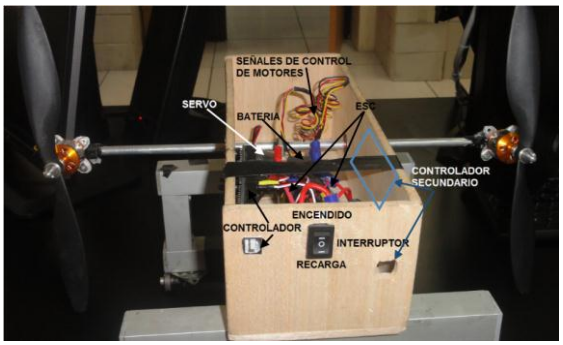
| | | porcentaje |
|---|---|---|
| 3 | **difícil** | **6.66%** |
| 0 | **Extremadamente difícil** | **0%** |
| 6 | **Extremadamente fácil** | **13.33%** |
| 36 | **fácil** | **80%** |
| 45 | **n** | **100%** |

# ANEXO L. ESPECIFICACIONES TÉCNICAS DEL SISTEMA.

*SISTEMA DE CONTROL DE UNA PLATAFORMA AÉREA NO TRIPULADA MEDIANTE UNA APLICACIÓN MÓVIL*

DESCRIPCIÓN GENERAL

Al instalar este sistema en un dirigible cuyo volumen sea 1.35 $m^3$ podemos ser capaces de controlar la navegación del mismo por medio de la interacción con la interfaz gráfica de la aplicación instalable en un dispositivo móvil que utilice *Android OS*.




APLICACIONES

El sistema puede tener muchas aplicaciones gracias a que es capaz de interactuar con un módulo secundario, el cual puede realizar tareas como por ejemplo: tomar fotos (módulo fotográfico), captar ruidos (sensor de sonido), detectar humedad (sensor de humedad) entre otros.

CARACTERISTICAS

*Dimensiones: 10 x 26 cm base; 8 cm altura.
*Velocidad de Transmisión: 115200 bps
*Interruptor de tres estados: encendido, apagado, recarga.
*Rango de direccionamiento: -60º <Θ< 60º
*Voltaje de alimentación: 11.1 V DC.
*Peso: 965 g.
*Alcance: 63 m al aire libre.

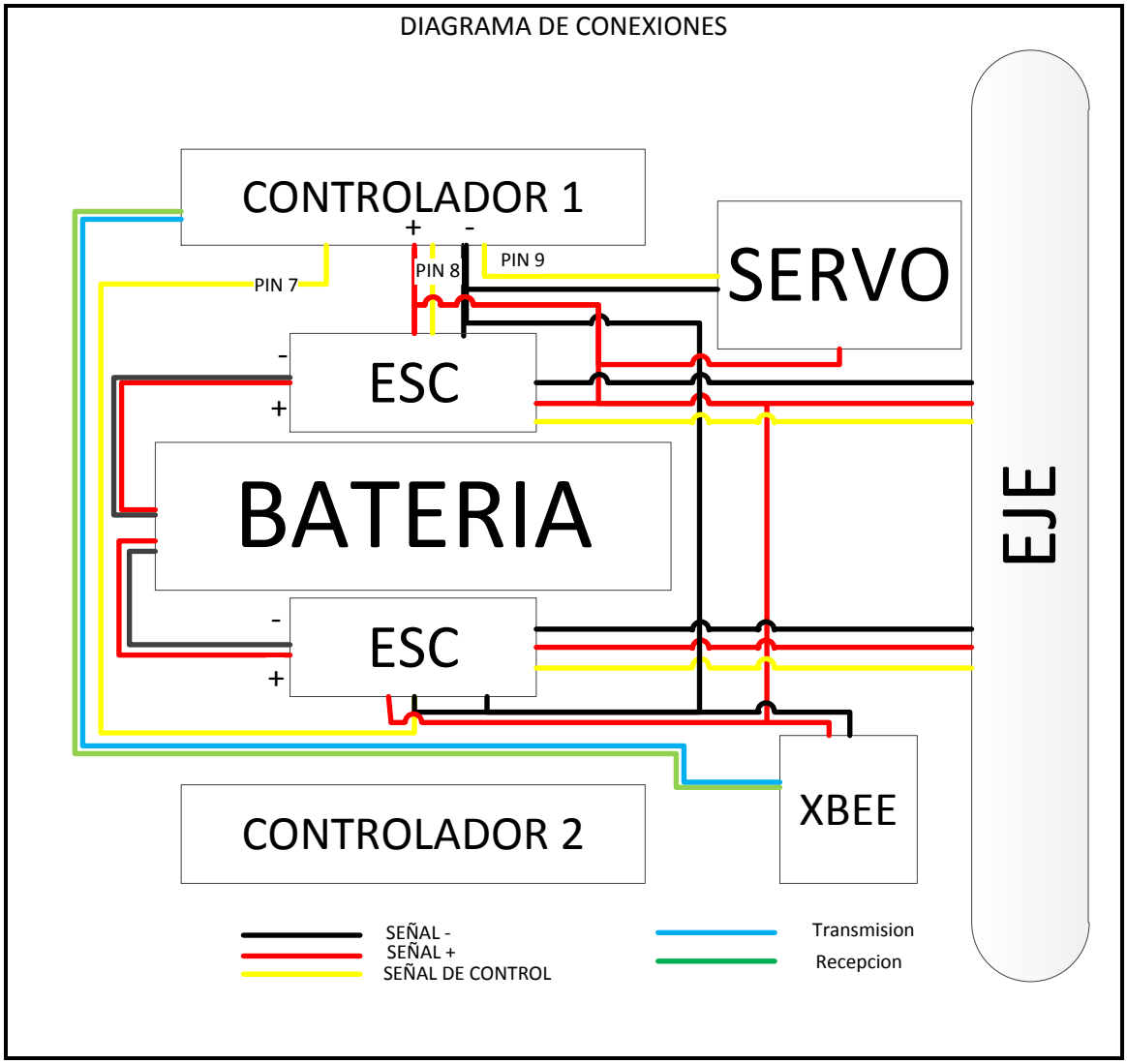


DIAGRAMA DE CONEXIONES
CONTROLADOR 1
+
-
PIN 7
PIN 8
PIN 9
SERVO
-
+
ESC
BATERIA
-
+
ESC
EJE
XBEE
CONTROLADOR 2
SEÑAL -
SEÑAL +
SEÑAL DE CONTROL
Transmision
Recepcion

TABLA DE FUNCIONAMIENTO LÓGICO.

| SEÑAL | CÓDIGO | $\Theta_{SERVO}$ | MOTOR DERECHO | | MOTOR IZQUIERDO | |
|---|---|---|---|---|---|---|
| | | | ON/OFF | % $PWM_{MOTOR}$ | ON/OFF | % $PWM_{MOTOR}$ |
| ARRIBA | 251 | Θ + 5 | ON | - | ON | - |
| ABAJO | 237 | Θ - 5 | ON | - | ON | - |
| DERECHA | 234 | Θ = 90 | OFF | - | ON | - |
| IZQUIERDA | 233 | Θ = 90 | ON | - | OFF | - |
| ADELANTE | 236 | Θ = 90 | ON | - | ON | - |
| DETENER MOTORES | 235 | Θ = 90 | OFF | 0 % | OFF | 0 % |
| V_MÍNIMO | 254 | Θ = 90 | - | 25 % | - | 25 % |
| V_NORMAL | 253 | Θ = 90 | - | 30 % | - | 30% |
| V_MÁXIMO | 252 | Θ = 90 | - | 35 % | - | 35 % |
| DESCONECTADO | 232 | Θ = 50 | ON | 25 % | ON | 27% |
| AUTENTICACIÓN | 255 | - | - | - | - | - |
| FOTO | 231 | - | - | - | - | - |

% $PWM_{MOTOR}$: % de tiempo de alto del PWM aplicado al motor (1ms → 0%, 2 ms → 100%)
$\Theta_{SERVO}$: Angulo del servo después de ejecutada la señal.

CONDICIONES DE OPERACIÓN RECOMENDADAS

| Símbolo | Parámetro | Valor | | | Unidades |
|---|---|---|---|---|---|
| | | mínimo | normal | máximo | |
| $V_{cc}$ | Voltaje de Alimentación | 8 | 11.1 | 12 | V |
| $T_A$ | Temperatura Ambiente | - | 25 | - | ºC |
| Z | Alcance | - | - | 63 | m |
| $W_1$ | Peso del módulo 1 | - | - | 965 | g |
| $W_2$ | Peso del módulo 2 | - | - | 205 | g |

CARACTERÍSTICAS ELÉCTRICAS.

| Símbolo | Consumo real del sistema | Valor | | | Unidad |
|---|---|---|---|---|---|
| | | V_MÍNIMO | V_NORMAL | V_MÁXIMO | |
| I | Corriente de consumo | 2.9 | 4.3 | 5.8 | A |
| V | Voltaje de consumo | 10.96 | 10.90 | 10.85 | V |

Valores obtenidos para un voltaje de alimentación de 11.07 V.

# BIBLIOGRAFÍA


[1] J. M. G. Yamin, «Curso Básico de Aeromodelismo,» APUCA, 18 Agosto 2008. [En línea]. Available: http://www.apuca.com.ar/Curso_Pagina_2.htm. [Último acceso: 26 Agosto 2014].

[2] J. C. Yunus Cengel, Mecanica De Fluidos, McGraw-Hill, 2012.

[3] A. R. Castellano, «Bluetooth.Introducción a su Funcionamiento,» 2012. [En línea]. Available: http://www.dea.icai.upco.es/sadot/Comunicaciones/avanzadas/Bluetooth.%20Introduccion%20a%20su%20funcionamiento.pdf. [Último acceso: 2014].

[4] Nokia, «developer.nokia,» 12 Abril 2007. [En línea]. Available: http://developer.nokia.com/community/wiki/Bluetooth_Protocol. [Último acceso: Agosto 2014].

[5] I. B. Faudot, «recercat.net,» Septiembre 2008. [En línea]. Available: http://www.recercat.net/bitstream/handle/2072/13081/pfc%20ivan%20barneda.pdf?sequence=1. [Último acceso: Agosto 2014].

[6] A. F. M. P. Jorge Mario Cotte Corredor, «DISEÑO DE CONTROL ROBUSTO DE VELOCIDAD DE MOTORES BRUSHLESS PARA ROBÓTICA AEREA,» Junio 2010. [En línea]. Available: http://www.bdigital.unal.edu.co/1896/1/jorgemariocottecorredor.2010.pdf. [Último acceso: Agosto 2014].

[7] ESTADO ECUATORIANO, «REGLAMENTO GENERAL A LA LEY ESPECIAL DE TELECOMUNICACIONES,» Quito, 2001.

[8] CONGRESO NACIONAL: COMISION DE LEGISLACION Y CODIFICACION, «Codigo aeronautico,» abril 2006. [En línea]. Available: http://www.aviacioncivil.gob.ec/wp-content/uploads/downloads/2013/12/C%C3%B3digo-Aeronautico.pdf.

[9] Ecobalsa, «Ecobalsa,» dedicada al manejo, producción y comercialización de madera balsa, 2011. [En línea]. Available: http://www.ecobalsa.com/descargar/FICHA_TECNICA_DE_LA_MADERA_BAL

SA.pdf. [Último acceso: 21 Agosto 2014].

[10] L. Xin, «Guangzhou HC Information Technology,» 4 2011. [En línea]. Available: http://iw.suntekstore.com/office_cache/265/14004375/132885991714004375.pdf. [Último acceso: 11 2013].

[11] A. Oyarce, «MCI Electronics,» 7 2010. [En línea]. Available: http://www.xbee.cl/descargas.html. [Último acceso: 11 2013].