Universidad Zaragoza
1542

Trabajo Fin de Grado

# Diseño e implementación de un sistema electrónico de control remoto para un dirigible

# Design and implementation of an electronic remote control system for an airship


Autor

Sergio Martínez Muñoz

Directores

Édgar Ramírez Laboreo

Eduardo Moya Lasheras


Grado en Ingeniería de Tecnologías Industriales


ESCUELA DE INGENIERÍA Y ARQUITECTURA

2024

# AGRADECIMIENTOS

Agradezco a todas las personas que han contribuido a la realización de este trabajo, permitiendo que se llevara a cabo de la mejor manera posible.

Agradecer a mis directores, Édgar Ramírez Laboreo y Eduardo Moya Lasheras, quienes me han apoyado y dirigido en todo momento con gran entrega y dedicación aportando su experiencia.

También agradecer la colaboración de Gonzalo López Nicolás por haber cedido el dirigible utilizado en este trabajo.

Por último, agradecer a mi familia, el acompañamiento y apoyo durante todos estos años de formación.

# Diseño e implementación de un sistema electrónico de control remoto para un dirigible

## RESUMEN

El presente Trabajo Fin de Grado se ha realizado en la Universidad de Zaragoza. Dicho trabajo aborda el desarrollo de un sistema electrónico de control para un dirigible no tripulado operado mediante radio control (RC), originalmente controlado manualmente con un mando físico. El objetivo de la línea de investigación es conseguir controlarlo manualmente desde un ordenador o un dispositivo móvil, y a largo plazo, de manera autónoma.

En primer lugar, se ha realizado un análisis del dirigible para conocer sus características. El dirigible utilizado es de pequeñas dimensiones (1.4 metros) y utiliza helio como gas para la sustentación. Se han descrito sus componentes y el sistema de comunicación que utilizaba originalmente. Posteriormente, para conocer en detalle esta comunicación, se han obtenido y analizado diversas mediciones sobre su funcionamiento.

A continuación, se han estudiado diferentes alternativas de control y se ha seleccionado la más apropiada. Como solución elegida, **se ha diseñado un sistema electrónico que permite controlar el dirigible a través de Wi-Fi** mediante la generación de señales PWM desde una Raspberry Pi. Para poder realizar esta implementación se ha necesitado un circuito electrónico y se han añadido unos conmutadores para poder alternar entre esta nueva alternativa y el control original con el mando físico.

Seguidamente, se han implementado diferentes funciones en la Raspberry Pi para poder controlar el dirigible. Además, dado que este control se quiere realizar mediante Wi-Fi, se ha diseñado una interfaz gráfica de control integrada en una página web. Para implementarla, se ha desarrollado el diseño web y el sistema de comunicación necesario entre la propia página web y la Raspberry Pi.

Por último, se han expuesto las conclusiones a las que se ha llegado tras la realización del trabajo y algunas de las posibles líneas futuras que se pueden seguir.

# Índice

# Capítulo 1

# Introducción

## 1.1. Contexto y antecedentes

El presente Trabajo Fin de Grado surge de la necesidad de desarrollar un sistema electrónico de control remoto para un dirigible no tripulado de pequeñas dimensiones operado mediante radio control (RC). La integración de un sistema de este tipo mejorará significativamente las capacidades operativas del dirigible, proporcionando mayor precisión y autonomía en sus aplicaciones.

El uso de dirigibles como vehículos aéreos se remonta a finales del siglo XIX, con los primeros modelos tripulados utilizados para transporte de pasajeros, correo aéreo, exploración y misiones militares. Algunos ejemplos populares son el Graf Zeppelin y el Hindenburg. Este último es conocido por el desastre que ocurrió en 1937 (un incendio causado por una fuga de hidrógeno, que se inflamó y destruyó rápidamente el dirigible al aterrizar en Nueva Jersey). Este hecho junto con la llegada de los aviones y helicópteros provocó que la popularidad de los dirigibles descendiera drásticamente.

En la actualidad, el avance de la tecnología ha provocado un resurgimiento del interés en los dirigibles. Estos vehículos podrían ser de gran utilidad como primera ayuda en casos de desastres y como alternativa logística limpia para el transporte de mercancías, al ser más rápidos y económicos que los buques portacontenedores. Algunas compañías de aeronaves están investigando su uso para trayectos cortos, ya que según sus estudios, podrían reducir las emisiones entre un 75 % y 90 % [1, 2].

La empresa aeroespacial Sceye está desarrollando dirigibles estratosféricos alimentados por energía solar con el objetivo de desempeñar diversas tareas como, por ejemplo, transmitir Internet a comunidades remotas. Recientemente, han completado un vuelo diurno en la estratosfera con energía renovable, un “hito fundamental” hacia vuelos de larga duración. Además, planean establecer su centro europeo de producción y operación de plataformas estratosféricas en Teruel, cuyo aeropuerto albergará el hangar más alto de España, y el único especializado en dirigibles [3, 4, 5].

Los dirigibles presentan varias ventajas significativas en comparación con otros vehículos aéreos. En primer lugar, su **capacidad para permanecer en vuelo durante períodos prolongados**, gracias a su flotación con helio, los hace ideales para misiones de vigilancia y supervisión continua. Esta longevidad operativa se complementa con una notable **eficiencia energética**, ya que los dirigibles requieren menos energía para mantenerse en vuelo que, por ejemplo los drones, que dependen exclusivamente de la propulsión motorizada.

Además, los dirigibles tienen un **impacto ambiental muy bajo** durante su operación. No necesitan aeropuertos para aterrizar o despegar, lo que les permite **acceder a áreas remotas o de difícil acceso** de manera más eficiente que los helicópteros. Otra ventaja distintiva de los dirigibles es su **operación silenciosa**. Esta característica es particularmente valiosa en operaciones que requieren discreción, como la observación de fauna o la vigilancia en áreas residenciales, minimizando el impacto acústico y las potenciales molestias para el entorno.

Sin embargo, también es importante considerar las limitaciones inherentes a los dirigibles. Su velocidad de vuelo es generalmente inferior a la de los drones, lo cual puede limitar su aplicabilidad en situaciones que demanden movilidad rápida o respuesta inmediata. Además, los dirigibles son más susceptibles a las condiciones meteorológicas adversas, como vientos fuertes, que pueden comprometer su estabilidad y control durante el vuelo. Asimismo, su mayor tamaño y maniobrabilidad limitada pueden dificultar su transporte y despliegue en entornos con restricciones espaciales o accesibilidad reducida.

En definitiva, a pesar de tener limitaciones, los dirigibles presentan diversas ventajas y utilidades tanto actuales como futuras, lo que los convierte en un campo de estudio atractivo. Algunas de estas aplicaciones son:

- **Vigilancia y seguridad**: Supervisión de grandes áreas, infraestructuras y eventos.
- **Monitorización ambiental**: Seguimiento de fenómenos naturales y recolección de datos sobre la calidad del aire y condiciones climáticas.
- **Investigación científica**: Estudios atmosféricos, ambientales y espaciales.
- **Transporte de carga**: Entrega de mercancías a zonas remotas o de difícil acceso.
- **Turismo aéreo**: Vistas panorámicas de paisajes y fauna desde la cabina.
- **Publicidad y entretenimiento**: Transmisión de mensajes publicitarios y realización de filmaciones aéreas.

En concreto, el dirigible que se va a emplear en este trabajo está diseñado para **uso en interiores** y como **soporte publicitario**. En la figura 1.1 se puede apreciar su aspecto ya inflado y montado:

Figura 1.1: Dirigible empleado en este trabajo

## 1.2. Motivación

La motivación principal de este Trabajo Fin de Grado es la necesidad de **optimizar el control del dirigible disponible**, ya que su manejo y operación presentan dificultades en términos de controlabilidad y precisión debido principalmente a las limitaciones inherentes al propio dirigible (poca potencia de los motores, baja masa/inercia, etc.). Actualmente, se requiere un operador manual que maneje el mando físico, lo que conlleva una dependencia de la habilidad del operador y la posibilidad de errores humanos. En la figura 1.2 se puede apreciar el esquema del sistema original.

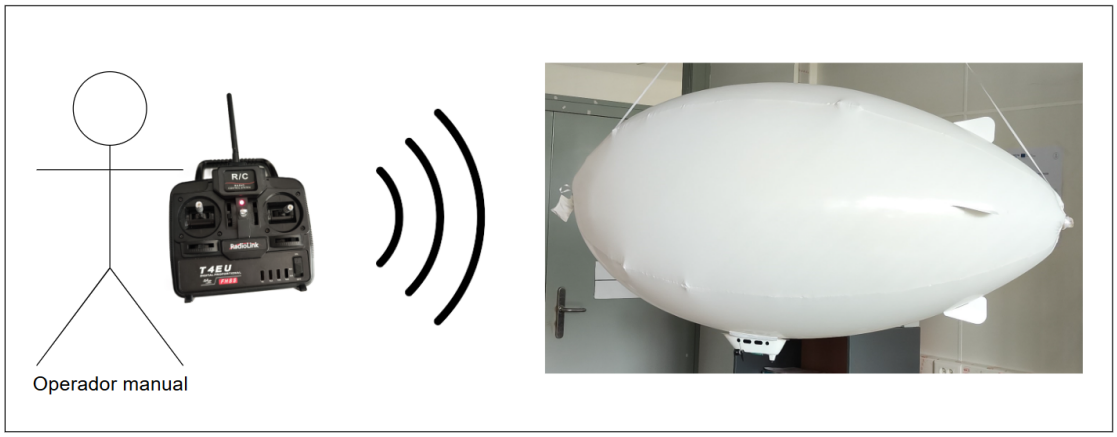



Figura 1.2: Esquema del sistema de funcionamiento original, con operador manual

El objetivo final de esta línea de investigación es lograr que el dirigible se mueva de manera controlada por un sistema automático. Para alcanzar este objetivo, se plantea una estrategia en etapas. Como primer paso, se busca sustituir o complementar la operación manual actual mediante el mando físico, con el control remoto desde un ordenador o un dispositivo móvil.

A largo plazo, el objetivo es desarrollar un sistema de control totalmente automático. Para lograr esto, será necesario implementar diversos tipos de sensores que proporcionen información en tiempo real sobre la posición, velocidad o entorno del dirigible. En la figura 1.3 se aprecia el esquema del sistema objetivo a largo plazo.

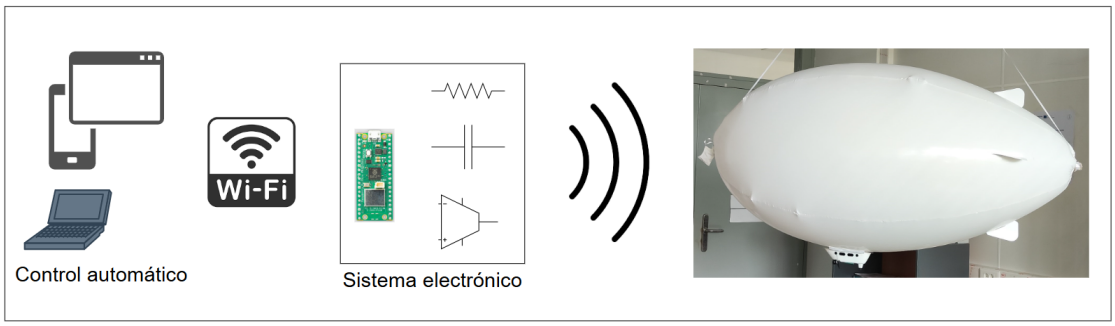



Figura 1.3: Esquema del sistema automático, objetivo a largo plazo

La implementación de un sistema de control automático no solo mejorará la precisión y la controlabilidad del dirigible, sino que también abrirá nuevas posibilidades para su uso en diversas aplicaciones, como pueden ser la vigilancia o la monitorización ambiental, entre otras.

## 1.3. Objetivos y alcance

El objetivo principal del trabajo es **diseñar e implementar un sistema de control** que sustituya o complemente el control manual que originalmente posee el dirigible **para lograr controlarlo desde un ordenador o un dispositivo móvil**. En la figura 1.4 se observa el esquema del sistema objetivo para este trabajo.

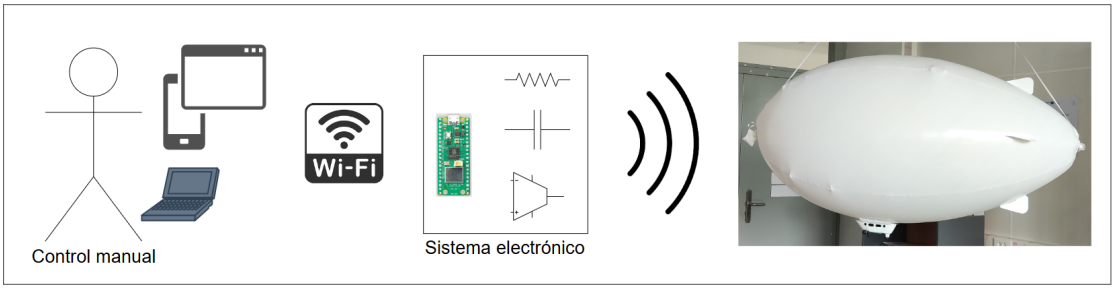



Figura 1.4: Esquema del sistema objetivo para este trabajo, con control manual

Para alcanzar este objetivo, se plantean las siguientes tareas específicas:

1. Descripción general del dirigible.
2. Estudio y mediciones sobre el sistema de comunicación original.
3. Selección de la alternativa de control.
4. Diseño y simulación del circuito electrónico.
5. Montaje del sistema electrónico seleccionado.
6. Implementación del control mediante la Raspberry Pi.
7. Selección e implementación del sistema de comunicación.
8. Diseño de la página web.
9. Pruebas de funcionamiento.
10. Estudio de las conclusiones y posibles líneas futuras.

El alcance de este trabajo abarca la **implementación de un nuevo sistema de control electrónico manual** tras el estudio del sistema de control original. Se han llevado a cabo tareas tanto de *hardware* como de *software*, incluyendo el diseño web.

## 1.4. Herramientas

Aparte del propio dirigible, durante la realización de este Trabajo Fin de Grado se ha hecho uso de un amplio abanico de herramientas para abordar los distintos aspectos del proyecto.

En primer lugar, se ha utilizado el entorno de programación MATLAB junto con su entorno de programación visual Simulink para realizar el modelado y la simulación del circuito que se plantea como alternativa de control.

Posteriormente, para el control del sistema electrónico, se ha empleado el entorno de desarrollo integrado (IDE), Thonny, que está diseñado para programar en MicroPython, un lenguaje optimizado para microcontroladores como el que se usa (Raspberry Pi).

Finalmente, se ha utilizado Visual Studio Code (VSC) como entorno de desarrollo integrado para crear la interfaz de control. Se han empleado los lenguajes estándar para el diseño web, tales como HTML, CSS y JavaScript.

Para la elaboración de la memoria se ha empleado el procesador de textos LaTeX.

## 1.5. Estructura de la memoria

Este Trabajo Fin de Grado está estructurado en los siguientes capítulos:

- **Capítulo 2**: Se analiza el dirigible. Para ello, se hace una descripción general del dirigible y se realizan mediciones sobre el sistema de comunicación original (Tareas 1 y 2).
- **Capítulo 3**: Se selecciona la alternativa de control, se diseña y simula el circuito electrónico y se realiza el montaje completo del sistema electrónico seleccionado (Tareas 3, 4 y 5).
- **Capítulo 4**: Se realiza la implementación del control mediante la Raspberry Pi, se selecciona e implementa la comunicación con la página web y se desarrolla el diseño web (Tareas 6, 7 y 8).
- **Capítulo 5**: Se muestran los resultados tras realizar las pruebas de funcionamiento y se detallan las conclusiones a las que se llega. Además, se exponen posibles líneas futuras para continuar con el proyecto (Tareas 9 y 10).

A continuación, en la figura 1.5, se presenta un esquema visual de los capítulos de la memoria:

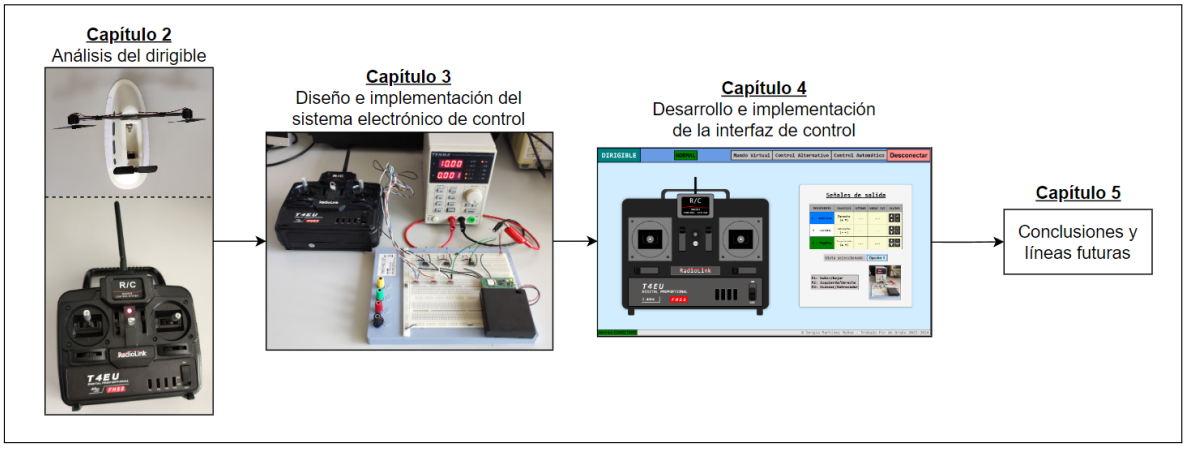



Figura 1.5: Estructura de la memoria

Tras estos capítulos, se aportan los siguientes **Anexos**:

- **Manual de usuario**: Contiene todo lo necesario para poder utilizar el dirigible.
- **Código fuente**: Se detalla alguna parte del código utilizado en el capítulo 4 tanto de MicroPython como de la página web.

## 1.6. Cronograma

En las figuras 1.6 y 1.7 se muestra la distribución temporal de las tareas realizadas y descritas en la sección 1.3. Es importante destacar que el diseño de la página web ha sido la tarea que más tiempo ha requerido.

DIAGRAMA DE GANTT

| | Nombre de tarea | Duración | Comienzo | Fin |
|---|---|---|---|---|
| 0 | **Cronograma TFG** | **132 días** | **lun 08/01/24** | **mar 09/07/24** |
| 1 | Descripción general del dirigible | 7 días | lun 08/01/24 | mar 16/01/24 |
| 2 | Estudio y mediciones sobre el sistema de comunicación original | 10 días | mié 17/01/24 | mar 30/01/24 |
| 3 | Selección de la alternativa de control | 14 días | mié 31/01/24 | lun 19/02/24 |
| 4 | Diseño y simulación del circuito electrónico | 14 días | mar 20/02/24 | vie 08/03/24 |
| 5 | Montaje del sistema electrónico seleccionado | 7 días | lun 11/03/24 | mar 19/03/24 |
| 6 | Implementación del control mediante la Raspberry Pi | 14 días | mié 20/03/24 | lun 08/04/24 |
| 7 | Selección e implementación del sistema de comunicación | 14 días | mar 09/04/24 | vie 26/04/24 |
| 8 | Diseño de la página web | 28 días | lun 29/04/24 | mié 05/06/24 |
| 9 | Pruebas de funcionamiento | 14 días | jue 06/06/24 | mar 25/06/24 |
| 10 | Estudio de las conclusiones y posible líneas futuras | 10 días | mié 26/06/24 | mar 09/07/24 |

Figura 1.6: Cronograma del trabajo realizado (tabla)

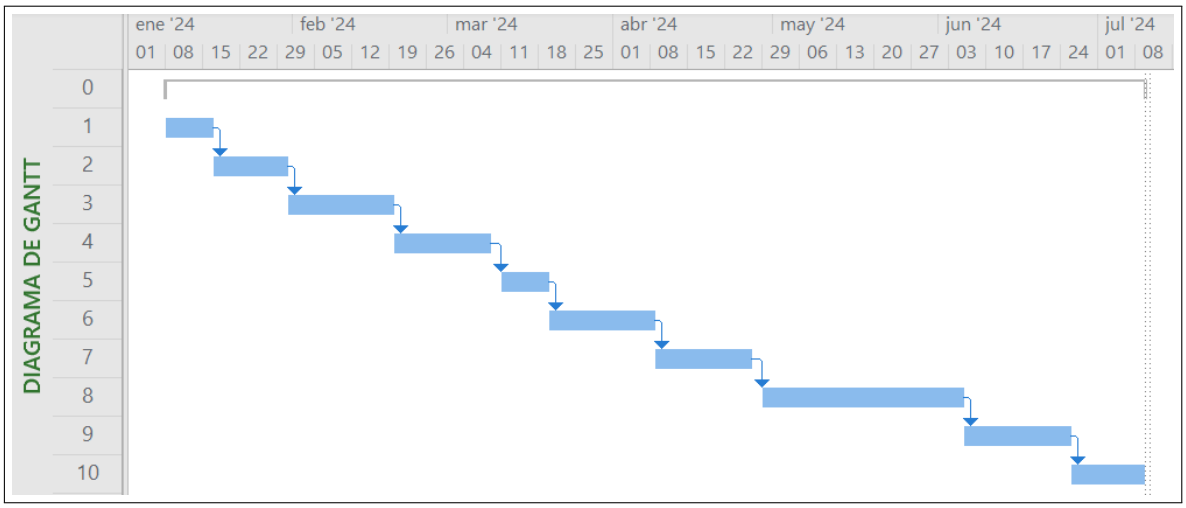



Figura 1.7: Cronograma del trabajo realizado (Diagrama de Gantt)

# Capítulo 2

# Análisis del dirigible

*En este capítulo se presenta el sistema describiendo los componentes del dirigible. Posteriormente, se estudia el sistema de control y su comunicación realizando varias pruebas y mediciones con un osciloscopio. Finalmente, se muestra un resumen con la información más relevante obtenida.*

## 2.1. Descripción general

El dirigible, conocido comercialmente como *Mini Zeppelín RC 1.4 m* [6], es un zepelín con un cuerpo hinchable de 1.4 metros de longitud fabricado en poliuretano. Diseñado para volar en interiores, este dirigible es controlado mediante radio control y está destinado principalmente a ser utilizado como soporte publicitario. El dirigible está formado por:

- **Emisor**: Sistema de control remoto RadioLink T4EU-6 2.4GHz, 6 canales.
- **Receptor**: RadioLink R7EH 2.4G, 7 canales. Se encuentra en la góndola.
- **Góndola**: Está diseñada a escala, tiene una apariencia similar a una cabina de pasajeros y posee 3 motores eléctricos con 3 hélices para controlar el dirigible.


(a) Emisor (Mando) (b) Receptor (c) Góndola

Figura 2.1: Mando, receptor y góndola

– **Cargador y baterías**: Necesarios tanto para el emisor como para el receptor.

  - El mando (emisor) necesita 8 pilas AA alcalinas.
  - La góndola (receptor) necesita 1 pila de litio de 3 V.


(a) Pilas del mando


(b) Cargador del mando


(c) Cargador y pila góndola

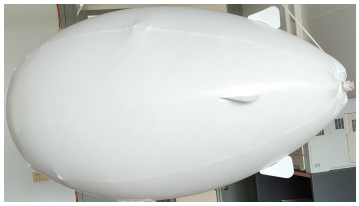
Figura 2.2: Cargadores y pilas del mando y de la góndola

– **Aletas de cola**: Dispone de 4 aletas para mejorar la estabilización y maniobrabilidad.

– **Globo**: Cuerpo hinchable del dirigible.

– **Bombonas de helio**: Para inflar el cuerpo del dirigible y varias recargas es suficiente con 0.2 $m^3$ de helio.

(a) Aleta



(b) Globo

(c) Bombona de helio

Figura 2.3: Aleta, globo y bombona de helio

En el anexo A se proporciona un manual de usuario en el que se explica cómo realizar el montaje y configuración del dirigible para poder ser utilizado.

## 2.2. Descripción del sistema de control

### 2.2.1. Control con el mando físico

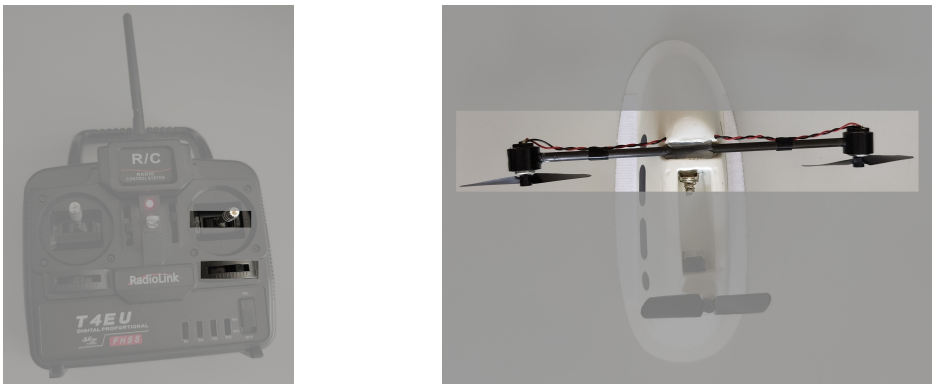
Para poder controlar el dirigible con el mando, se ha de realizar la sincronización entre el mando (emisor) y la góndola (receptor) tal y como se explica en la sección A.3.1 del Anexo A. Este mando cuenta con dos *sticks* para controlar la dirección y la altura del dirigible:

- ***Stick* Derecho**: arriba/abajo ⇒ El dirigible subirá o bajará.
  Controla el sentido de giro de la hélice individual. En la figura 2.4 se muestra su correspondencia con el mando y las hélices.


(a) *Stick* y *offset* del mando (b) Hélices de la góndola

Figura 2.4: Representación gráfica del mando y la góndola para subir/bajar

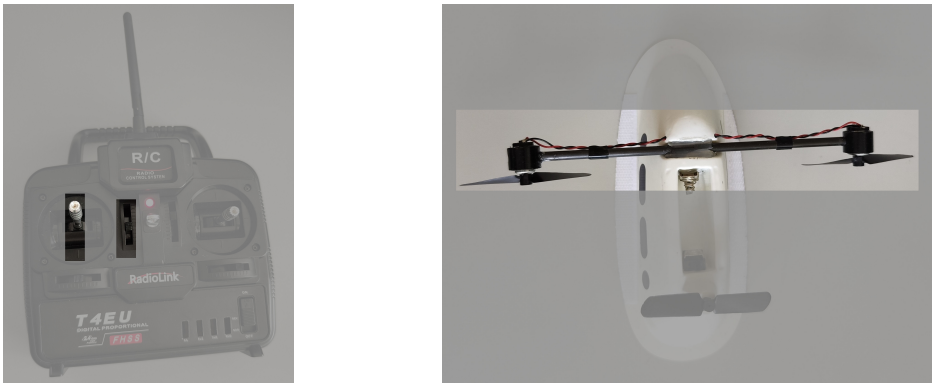
- ***Stick* Derecho**: izquierda/derecha ⇒ El dirigible irá hacia izquierda o derecha.
  Controla la diferencia de velocidades del par de hélices. En la figura 2.5 se muestra su correspondencia con el mando y las hélices.




(a) *Stick* y *offset* del mando (b) Hélices de la góndola

Figura 2.5: Representación gráfica del mando y la góndola para izquierda/derecha

– ***Stick* Izquierdo**: arriba/abajo ⇒ El dirigible avanzará o retrocederá.
Controla la velocidad media del par de hélices. En la figura 2.6 se muestra su correspondencia con el mando y las hélices.




(a) *Stick* y *offset* del mando (b) Hélices de la góndola

Figura 2.6: Representación gráfica del mando y la góndola para avanzar/retroceder

El *stick* izquierdo en las direcciones izquierda/derecha no se utiliza.

Además, el mando cuenta con un *trim* (*offset*) para cada dirección que sirve para regular y ajustar la señal que se emite. En la figura 2.7 se muestran los movimientos de los *trims*, los cuales se deben ajustar para conseguir que no se mueva ninguno de los motores de la góndola.

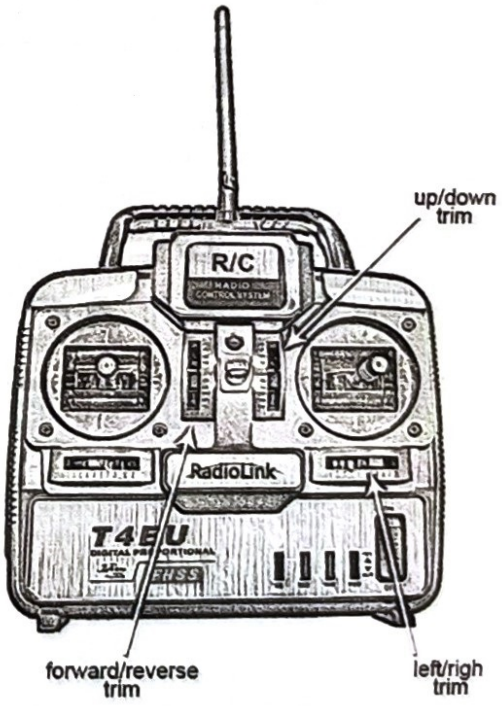



Figura 2.7: Dibujo del mando con el movimiento de los *trims* [7]

### 2.2.2. Características y comunicación del mando (emisor)

El mando *RadioLink T4EU-6 2.4 GHz* [8] es un sistema de control remoto que incluye un transmisor de 4 canales y un receptor de 6 canales. Opera en la banda de frecuencia de 2.4 GHz, comúnmente utilizada para comunicaciones inalámbricas debido a su capacidad de minimizar interferencias. Este sistema emplea modulación FHSS (Frequency Hopping Spread Spectrum), una tecnología que cambia rápidamente entre diferentes frecuencias dentro de la banda de 2.4 GHz, proporcionando una conexión más segura. Gracias a esta tecnología, el sistema puede mantener una distancia de control estable de más de 300 metros.


Figura 2.8: Mando (emisor)

A efectos de este trabajo, no ha sido necesario profundizar más en este aspecto, ya que, como se explica en el capítulo 3, en la solución seleccionada no se ha sustituido este mando (emisor), sino que se le ha realizado una modificación.

### 2.2.3. Características y comunicación de la góndola (receptor)

La góndola del dirigible, tal y como se ve en la figura 2.9, está equipada con tres motores eléctricos, cada uno con su propia hélice, que permiten el control del dirigible. Además, cuenta con un circuito electrónico que ajusta la señal recibida del receptor, permitiendo el control de los motores. En la figura 2.10 se aprecian mejor sus componentes.

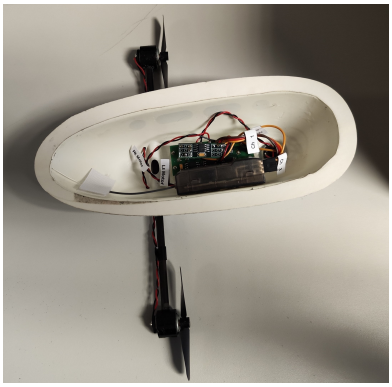
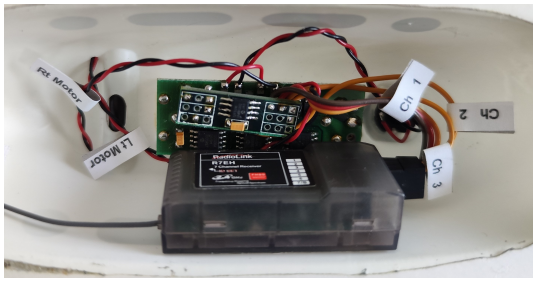


Figura 2.9: Góndola

El componente principal de la góndola es el receptor *RadioLink R7EH 2.4G 7 Canales* [9], un receptor digital de 7 canales que opera en la frecuencia de 2.4GHz y utiliza modulación FHSS para minimizar las interferencias. Este receptor está diseñado para funcionar con el mando (emisor) RadioLink T4EU-6.




(a) Sistema de la góndola


(b) Receptor

Figura 2.10: Sistema de la góndola y receptor

El receptor soporta varios modos de señal: en modo PWM, emite señales PWM en 7 canales; en modo SBUS/PPM, emite señal SBUS en el canal 1, señal PPM en el canal 2, y señales PWM en los canales 3 a 7.

De igual forma que con el mando, no ha sido necesario profundizar más, ya que en la solución seleccionada tampoco se ha sustituido el receptor. En este caso, ni siquiera ha sido necesario modificarlo.

## 2.3. Obtención y análisis de las mediciones sobre el funcionamiento

Como último paso antes de proponer una solución de control alternativa, se han tomado diferentes mediciones para conocer el funcionamiento del dirigible y su comunicación entre el mando y el receptor que controla los motores de la góndola.

### 2.3.1. Señales emitidas por el mando

El mando cuenta con cuatro potenciómetros, de los cuales se utilizan tres, cada uno controlado por su correspondiente *stick*. Para poder medir las señales emitidas por el mando, se ha decidido abrirlo. Para ello, primero se han quitando los tornillos necesarios y luego se han separado las dos partes del mando, tal y como se observa en la figura 2.11.

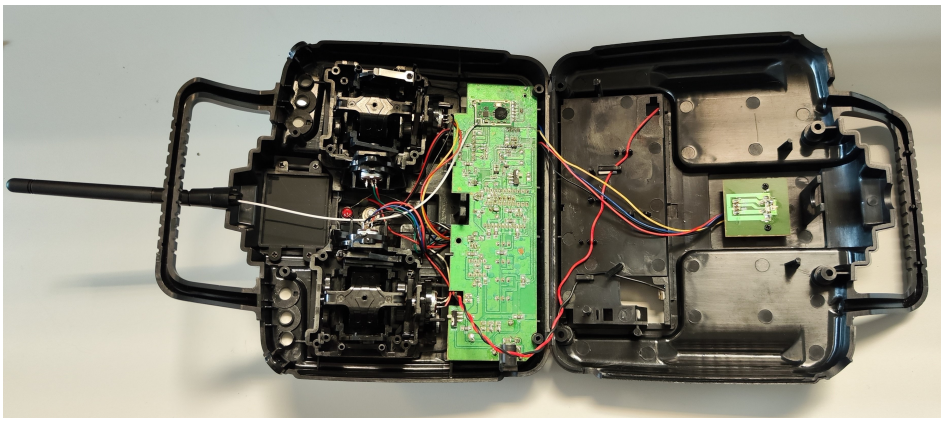
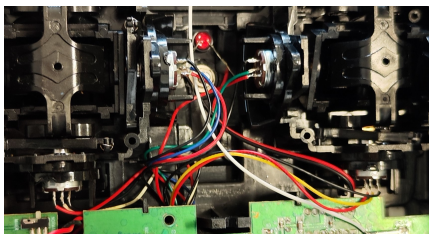
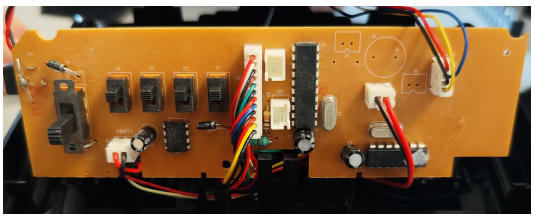


Figura 2.11: Mando abierto, separado en dos partes



(a) Potenciómetros



(b) Circuito electrónico

Figura 2.12: Componentes electrónicos del mando

En la figura 2.12a se observa que de cada potenciómetro salen tres cables: uno rojo que es la alimentación, otro negro que es la tierra y uno de un color diferente para cada potenciómetro, correspondiente a la señal de control que emite.
El cable de señal es azul para el movimiento de subir-bajar, blanco para izquierda-derecha y verde para avanzar-retroceder.

Tras examinar el interior del mando, se ha utilizado un osciloscopio para verificar el tipo de señales que se envían. Para ello, se ha conectado una sonda al cable de la señal de control (el de color diferente) y al de tierra (el negro) de uno de los potenciómetros. Además, se ha conectado otra sonda al otro canal del osciloscopio para medir la señal del cable de alimentación (el rojo). Moviendo el potenciómetro, se ha podido observar la señal en el tiempo, obteniendo el siguiente resultado de la figura 2.13.

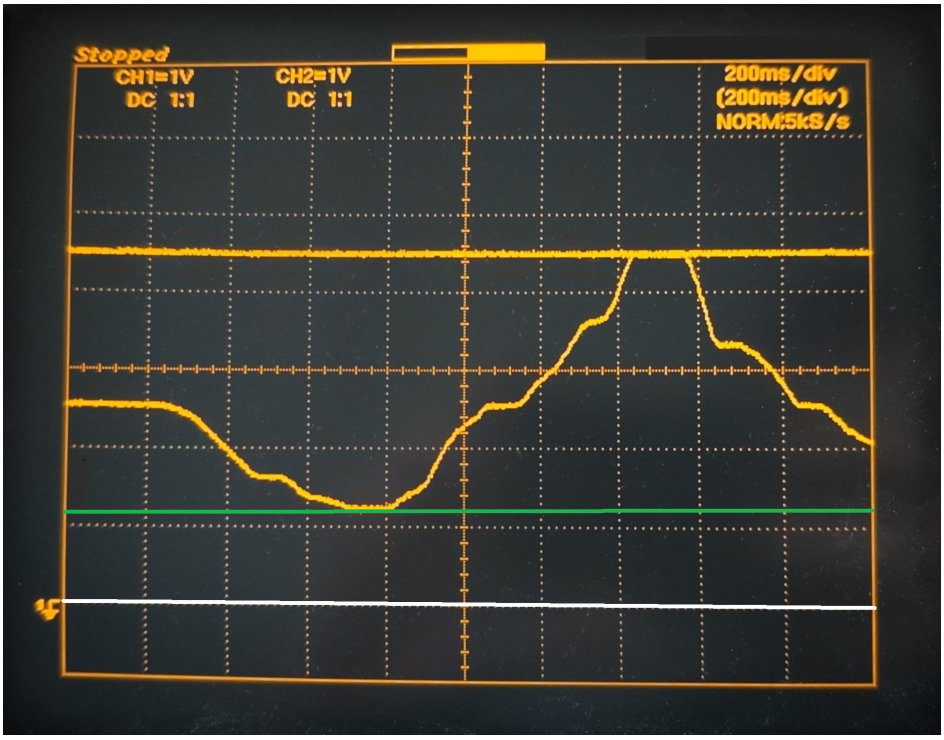



Figura 2.13: Señal de alimentación y de control observada con un osciloscopio

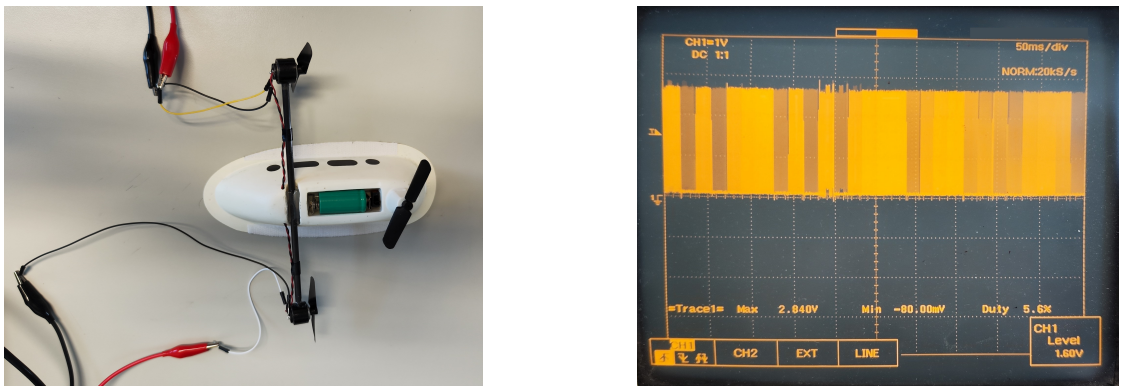
La línea blanca indica la referencia de tierra, es decir, 0 V (cable negro). La línea horizontal naranja de la parte superior está a 4.5 V y representa la señal de alimentación del potenciómetro (cable rojo), que es una señal constante. La señal que aparece entre la línea verde (1.2 V) y la señal de alimentación (4.5 V) es la señal de control (cable de color diferente) y corresponde a la variación de la señal emitida por el potenciómetro mientras se mueve.

Esta señal de control es continua y varía entre 1.2 V y 4.5 V, aunque en el capítulo siguiente se explica que este rango puede variar entre 0 V y 4.5 V, dependiendo del ajuste del *trim* (desplazamiento) asociado a cada potenciómetro.

Este aspecto es importante y se ha tenido en cuenta al seleccionar la solución de control alternativa. En la sección 3.2.1 se muestran los rangos de funcionamiento que se han obtenido para cada uno de los potenciómetros que controlan los motores (envían la señal al receptor, que luego la transmite a los motores para su control).

### 2.3.2. Señales recibidas por los motores de la góndola

La góndola cuenta con tres motores para controlar el dirigible. Para evaluar las señales que reciben se ha utilizado un osciloscopio, conectando la sonda a los motores de la pareja de hélices, tal y como se muestra en la figura 2.14.




(a) Sonda motores góndola (b) Señal motores góndola

Figura 2.14: Sonda en los motores de la góndola y señal medida

Las señales que reciben no son fáciles de interpretar ya que son señales moduladas a tres niveles, probablemente provenientes de un circuito puente en H como el de la figura 2.15. Estas señales son un método eficaz para controlar motores eléctricos, ya que permiten un control preciso tanto de la dirección como de la velocidad.

Este circuito se caracteriza por utilizar cuatro interruptores (transistores o relés) dispuestos en una configuración que forma una estructura similar a la letra H. Dependiendo de cómo se activan o desactivan estos interruptores, es posible invertir la polaridad en el motor, determinando así su dirección de giro. Además, se puede variar la velocidad del motor modificando el ciclo de trabajo de la señal PWM (modulación por ancho de pulso) enviada a los interruptores.

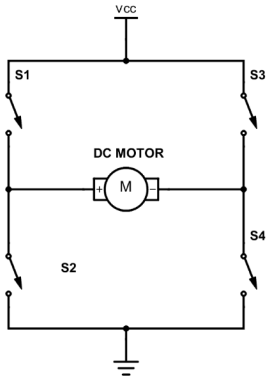



Figura 2.15: Esquema del circuito con configuración en puente en H

A pesar de la dificultad de interpretar la modulación de estas señales, no ha sido necesario profundizar más en su análisis ya que el receptor no ha sido modificado.

## 2.4. Resumen del sistema de control del dirigible

Tras haber descrito y analizado el sistema de control junto la comunicación entre el emisor (mando) y el receptor, se han identificado las siguientes especificaciones:

- **Frecuencia de funcionamiento**: 2.4 GHz. Esta tecnología es un estándar común en la transmisión de señales de radio, puede tener un alcance de hasta varios cientos de metros y tiene una mejor resistencia a las interferencias.
- **Modulación en la transmisión**: FHSS. Requiere una sincronización precisa entre emisor y receptor. Además, es difícil conocer cómo se produce el cambio de frecuencias al emitir las señales.
- **Canales de control en el receptor**: 7 disponibles en el receptor. De los cuales, el dirigible sólo utiliza los 3 primeros canales para su comunicación con el emisor.
  - Canal 1 → Controla el giro a izquierda o derecha.
  - Canal 2 → Controla si sube o baja.
  - Canal 3 → Controla si avanza o retrocede.

  La señales que salen de estos canales van hacia el circuito electrónico de la góndola, donde se adaptan para controlar los motores que regulan el movimiento de las tres hélices.
- **Señal de alimentación de cada potenciómetro**: Es constante de 4.5 V.
- **Señal de control de cada potenciómetro**: Las señales que emite el mando (emisor) son continuas con un rango de 0V a 4.5V. El rango específico de trabajo de cada potenciómetro, considerando el *stick* y el *trim* (u *offset*), es diferente para cada uno de los potenciómetros y se describe en la sección 3.2.1.
  Es importante conocer estos rangos de funcionamiento con precisión, ya que determinan cómo se mueven los motores en cada dirección.
- **Señal recibida por los motores**: La señal que controla finalmente los motores es una señal modulada a tres niveles, difícil de interpretar.

En conclusión, se ha observado que la manera más sencilla de desarrollar un control alternativo es actuar sobre las señales de control de los potenciómetros. En el siguiente capítulo se explica detalladamente cómo implementar este control alternativo. Además, se analizan otras alternativas y se argumenta por qué se ha descartado cada una de ellas.

# Capítulo 3

# Diseño e implementación del sistema electrónico de control

*En este capítulo se detalla el diseño del sistema electrónico alternativo de control, desde su selección y justificación entre las posibles alternativas hasta la implementación y montaje del sistema final. Además, se presenta el circuito electrónico diseñado junto con la simulación realizada para verificar su correcto funcionamiento.*

## 3.1. Selección de la alternativa

El objetivo es implementar un sistema de control alternativo que permita manejar el dirigible desde un ordenador o un dispositivo móvil, por ejemplo, a través de Wi-Fi. Además, se busca conservar la posibilidad de utilizar el control original con el mando físico. Para lograrlo, se han analizado las siguientes opciones como posibles soluciones para implementar este control alternativo.

### 3.1.1. Sustituir el receptor de la góndola

Inicialmente, se ha considerado la posibilidad de sustituir el receptor original por un receptor con tecnología Wi-Fi que fuera capaz de actuar como controlador, utilizando un microcontrolador o una tarjeta de desarrollo, por ejemplo, Raspberry Pi o Arduino.

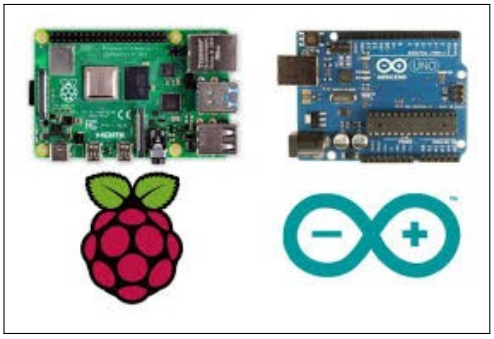


Figura 3.1: Ejemplos de Raspberry Pi y Arduino

Sin embargo, esta opción presenta varios problemas importantes. El primero es que se perdería la capacidad de controlar el dirigible de la manera original con el mando físico, ya que el nuevo receptor probablemente no sería compatible con el mando existente.

Además, es fundamental que el receptor, aparte de recibir señales a través de Wi-Fi, sea capaz de enviar las señales adecuadas para controlar los motores del dirigible. Esto sería complicado de realizar ya que, como se ha explicado en la sección 2.3.2 del capítulo anterior, las señales que reciben los motores que controlan las hélices de la góndola son difíciles de interpretar.

Por lo tanto, tras evaluar y estudiar la manera de solventar estos problemas, se ha decidido **descartar esta opción**.

### 3.1.2. Sustituir el emisor (el mando)

La siguiente idea que se ha planteado es diseñar un sistema electrónico utilizando un microcontrolador o una tarjeta de desarrollo, como una Raspberry Pi o un Arduino, y añadir un transceptor de radiofrecuencia (RF) como el de la figura 3.2 para realizar la comunicación con el receptor de la góndola. Esta solución reemplazaría el emisor original (el mando) por un sistema electrónico que enviaría las señales necesarias al receptor.

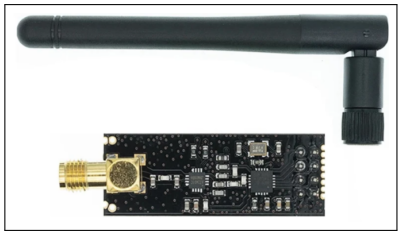


Figura 3.2: Ejemplo de transceptor de radiofrecuencia (NRF24L01)

Esta opción presenta la ventaja de no requerir modificaciones en el receptor, preservando la integridad de la góndola, que es el componente más delicado del sistema. Sin embargo, esta solución implica perder la capacidad de controlarlo con el mando original, lo cual no era el objetivo inicial.

Además, presenta el problema de replicar la comunicación específica entre el emisor y el receptor original. Tal y como se ha explicado en el capítulo anterior, se utiliza la técnica de modulación FHSS, lo cual complica la determinación de las frecuencias y los tiempos de cambio de las señales emitidas. Cada empresa incorpora un *firmware* diferente en sus receptores, por lo que simplemente usar un sistema electrónico y añadir un módulo emisor de RF no sería suficiente para que funcione adecuadamente. Por lo tanto, debido a este motivo, también **se ha descartado esta opción**.

### 3.1.3. Utilizar el cable de entrenamiento

Tras revisar de nuevo el mando disponible, se ha observado que incluye una característica denominada “cable de entrenamiento”. Este sistema conecta dos mandos RC, permitiendo que un instructor tome el control del modelo de forma instantánea si el estudiante comete un error, aumentando así la seguridad.

En los sistemas tradicionales, el cable de entrenamiento conecta físicamente los mandos, mientras que en los sistemas más modernos se utiliza un dispositivo inalámbrico como el de la figura 3.3.

Figura 3.3: Cable de entrenamiento inalámbrico [10]

En la figura 3.4 se muestran las dos configuraciones del cable de entrenamiento: una con conexión física y otra con conexión inalámbrica.


(a) Cable de entrenamiento


(b) Cable de entrenamiento inalámbrico

Figura 3.4: Conexiones con cables de entrenamiento tradicional e inalámbrico [10]

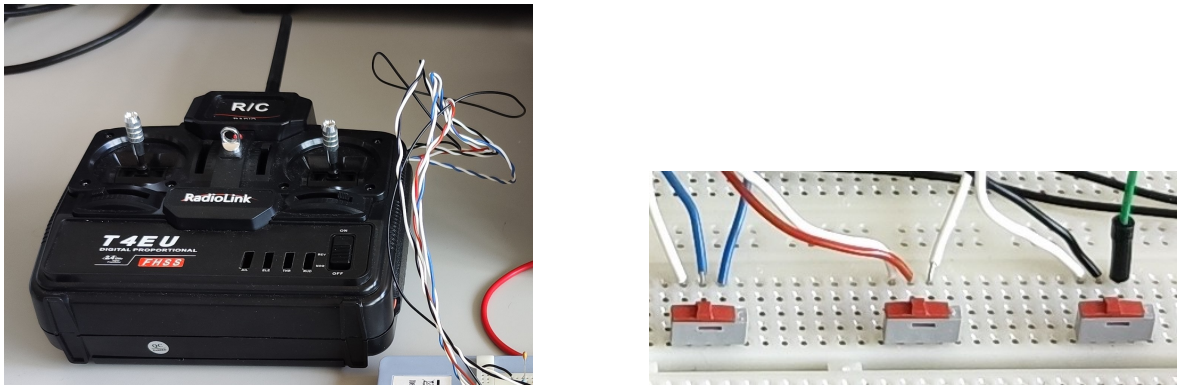
Sin embargo, al no encontrar una manera efectiva de aplicar esta tecnología del cable de entrenamiento, también **se ha descartado esta opción**.

## 3.2. Descripción de la alternativa de control

Hasta ahora, las soluciones planteadas implicaban sustituir algunos de los elementos que realizan la comunicación o utilizar un elemento externo del mando. Sin embargo, tras no llegar a una solución concreta, se ha optado por **modificar el mando (el emisor)** para añadir una etapa intermedia en el proceso de comunicación que facilite la selección entre la señal original de los potenciómetros y una señal alternativa.

Se ha propuesto como solución **utilizar un conmutador para elegir entre el control mediante el mando original o mediante un sistema electrónico**. Para ello, se han desoldado los cables que reciben la señal de control que estaban soldados a la patilla de cada potenciómetro y se les ha añadido un cable más largo para sacarlos fuera. Además, desde esa patilla de los potenciómetros, se ha soldado otro cable para conservar el acceso a la señal de cada uno de ellos y mantener la capacidad de control mediante el mando.

En la figura 3.5 se puede apreciar el mando modificado con los cables fuera y los conmutadores a los que llegan estos cables (dos por cada potenciómetro).




(a) Modificación del mando (b) Conmutadores

Figura 3.5: Modificación del mando y cables en los conmutadores

En la pata central de cada conmutador se encuentra conectado el cable de control correspondiente a cada potenciómetro. A través de este cable, se recibe la señal seleccionada, que puede ser la proveniente de los potenciómetros (modo de funcionamiento original) o la generada por el circuito electrónico diseñado. Esta última señal debe emular las señales de los potenciómetros, permitiendo así que los comandos transmitidos a los motores de la góndola funcionen de la misma manera que si fueran generados directamente por los potenciómetros del mando.

Antes de implementar esta solución, se ha realizado una comprobación alimentando el mando con una fuente externa y variando el voltaje suministrado, verificando que los motores se mueven de manera adecuada.

### 3.2.1. Determinación de los rangos de funcionamiento

Después de verificar que el sistema se comporta adecuadamente, se ha realizado un estudio detallado para determinar los rangos de funcionamiento específicos para controlar la dirección y altura del dirigible. Para ello, se han medido las señales de control de los tres potenciómetros (cables azul, blanco y verde). En la figura 3.6 se puede ver la disposición de cada uno de los tres potenciómetros en el mando.

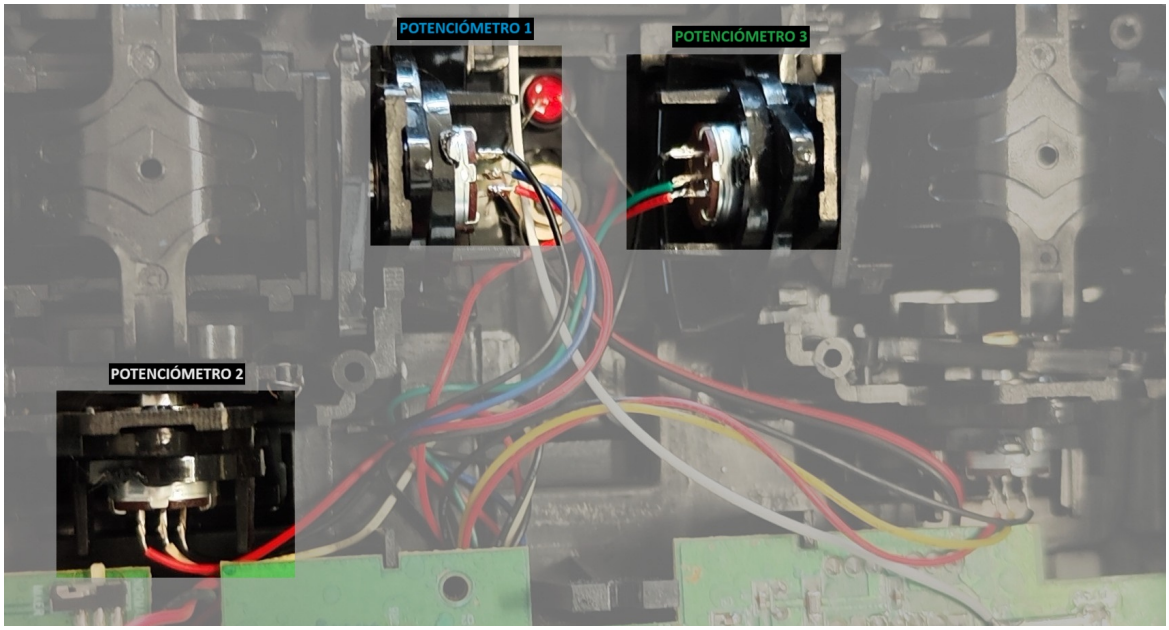



Figura 3.6: Situación de cada potenciómetro en el interior del mando

Se ha observado que no tienen los mismos rangos de funcionamiento los tres potenciómetros y varían significativamente. A continuación, se van a mostrar los rangos específicos para controlar cada uno de los potenciómetros.

**POTENCIÓMETRO 1: Movimiento Subir-Bajar**

| **MOVIMIENTO** | Subir | Parar | Bajar |
|---|---|---|---|
| **RANGO** | 1.2V – 1.95 V | 1.95 V – 2.25 V | 2.25 V – 3.4 V |

Tabla 3.1: Valores medidos desde el potenciómetro 1

Cuando el potenciómetro del mando está dentro del rango de bajar, la hélice gira en sentido horario, mientras que en el rango de subir, gira en sentido antihorario. En el rango de parar, permanece inmóvil.

**POTENCIÓMETRO 2: Movimiento Izquierda-Derecha**

| **MOVIMIENTO** | Izquierda | Parar | Derecha |
|---|---|---|---|
| **RANGO** | 0.55 V – 2.55 V | 2.55 V – 3 V | 3 V – 4.5 V |

Tabla 3.2: Valores medidos desde el potenciómetro 2

Cuando el potenciómetro del mando está dentro del rango de izquierda, la hélice izquierda gira en sentido horario, mientras que la hélice derecha gira en sentido antihorario. En el rango de derecha ocurre lo inverso. En el rango de parar, ambas permanecen inmóviles.

**POTENCIÓMETRO 3: Movimiento Avanzar-Retroceder**

| **MOVIMIENTO** | Avanzar | Parar | Retroceder |
|---|---|---|---|
| **RANGO** | 0 V – 1.25 V | 1.25 V – 1.75 V | 1.75 V – 3.8 V |

Tabla 3.3: Valores medidos desde el potenciómetro 3

Cuando el potenciómetro del mando está dentro del rango de avanzar, las 2 hélices giran en sentido horario, mientras que en el rango de retroceder, las 2 hélices giran en sentido antihorario. En el rango de parar, ambas permanecen inmóviles.

La tensión que finalmente recibe cada una de las hélices laterales depende de los valores de los potenciómetros 2 y 3. De este modo, el potenciómetro 2 controla la diferencia de velocidades entre las hélices, mientras que el potenciómetro 3 regula la velocidad media.

Como se observa en las tablas, a pesar de que el rango de los potenciómetros del mando varía desde 0 V hasta 4.5 V, ocurre que en ciertos rangos satura y al aumentar o disminuir este valor, no varía la velocidad de funcionamiento del dirigible.

Es importante señalar que el rango de control es reducido, lo que puede presentar ciertas limitaciones en la precisión del movimiento. Además, se observa que en los movimientos de derecha y avanzar (indicados en color rojo en las tablas 3.2 y 3.3), la velocidad máxima que se puede alcanzar es significativamente inferior a la que se obtiene en su respectiva dirección opuesta.

También es importante tener en cuenta que no enviar ninguna señal, es decir, mantener las señales de los 3 potenciómetros en 0 V, no garantiza que el dirigible permanezca estático (con las 3 hélices detenidas) ya que tal y como se observa en las tablas 3.1, 3.2 y 3.3, los 0 V están en los rangos de alguno de los movimientos.

Por lo tanto, es necesario enviar un voltaje específico para cada movimiento que haga que el dirigible permanezca quieto (las 3 hélices detenidas). Los voltajes que se van a utilizar se muestran en la tabla 3.4:

| **MOVIMIENTO** | Subir-Bajar | Izquierda-Derecha | Avanzar-Retroceder |
|---|---|---|---|
| **VALOR** | 2.10 V | 2.75 V | 1.50 V |

Tabla 3.4: Valores para parar cada uno de los 3 movimientos

## 3.3. Diseño del sistema electrónico

Para implementar la solución propuesta, es necesario diseñar un sistema electrónico capaz de replicar las señales continuas producidas por los potenciómetros del mando. Este sistema electrónico está compuesto por tres partes diferenciadas.

En primer lugar, se debe seleccionar una tarjeta de desarrollo adecuada que pueda generar las señales continuas requeridas. Esta tarjeta debe ser capaz de producir las variaciones de voltaje necesarias para emular con precisión el comportamiento de los potenciómetros. Posteriormente, es necesario diseñar un circuito electrónico que acondicione las señales generadas por la tarjeta de desarrollo. Este circuito asegurará que las señales sean continuas y que se mantengan dentro de los valores adecuados para su correcto funcionamiento. Por último, se debe elegir un tipo de conmutador que permita alternar entre los dos modos diferentes de control.

A continuación, en la figura 3.7, se presenta un esquema simplificado que ilustra el diseño del sistema electrónico propuesto.

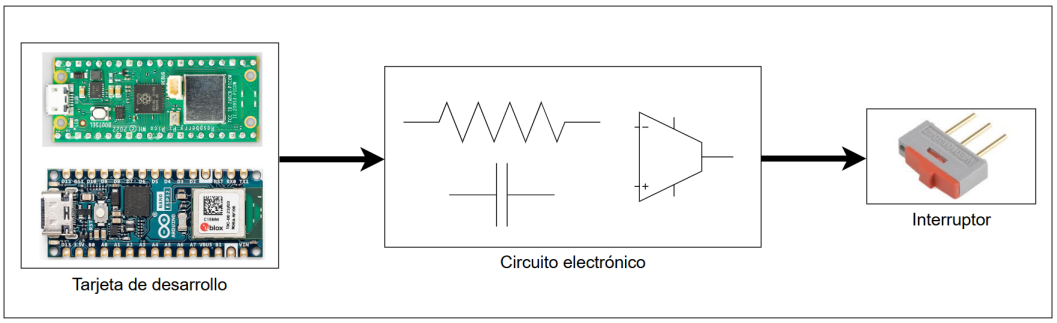



Figura 3.7: Esquema del sistema electrónico

### 3.3.1. Selección de la tarjeta de desarrollo

Las dos opciones principales que se han considerado son la Raspberry Pi Pico WH [11] y el Arduino Nano RP2040 Connect [12]. A continuación, se presenta la tabla 3.5 comparativa:

| Características | Raspberry Pi Pico WH | Arduino Nano RP2040 Connect |
|---|---|---|
| **Microcontrolador** | RP2040 | RP2040 |
| **Memoria SRAM** | 264 KB | 264 KB |
| **Memoria Flash** | 2 MB | 16 MB |
| **Conectividad** | Wi-Fi y Bluetooth | Wi-Fi y Bluetooth |
| **Genera PWM** | Sí | Sí |
| **Voltaje de operación** | 3.3V | 3.3V |
| **Precio** | 9€ | 33€ |

Tabla 3.5: Comparación entre Raspberry Pi Pico WH y Arduino Nano RP2040 Connect

Los requisitos principales que debe cumplir la tarjeta de desarrollo son la capacidad para generar señales PWM y disponer de tecnología Wi-Fi. Ambas opciones cumplen con estos criterios, por lo que la elección entre ellas es indiferente. Sin embargo, tras considerar otros factores como el precio y mi experiencia previa, se ha decidido optar por la **Raspberry Pi Pico WH**.

En la figura 3.8 se puede ver su distribución de pines:

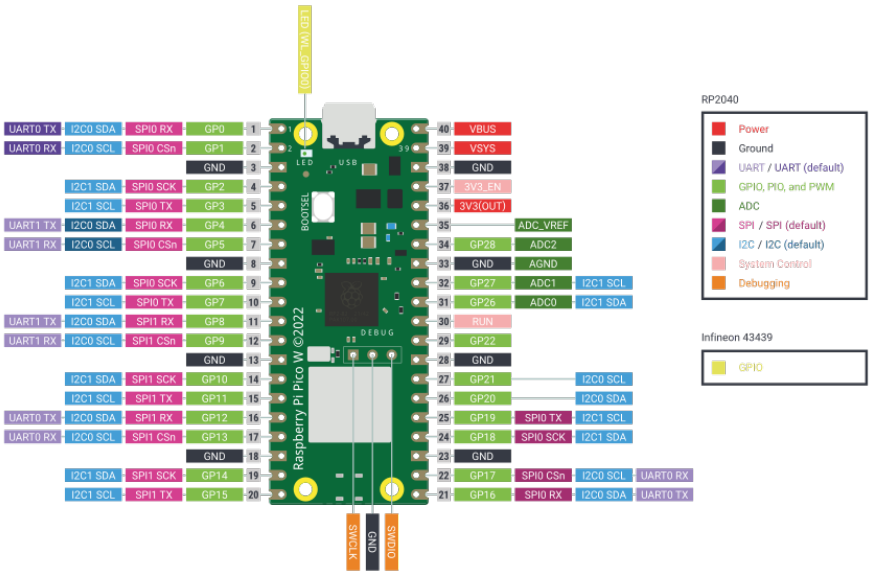



Figura 3.8: Raspberry Pi Pico WH

Este modelo de Raspberry Pi dispone de 26 pines utilizables que pueden generar señales PWM con un voltaje máximo de 3.3 V. Dado que se requiere generar señales en un rango de 0 a 4.5 V, esto supone un problema. Para abordar esta limitación, se ha diseñado un circuito electrónico, que se detalla en la sección 3.3.2.

Además, para alimentarla, se va a utilizar el portapilas de la figura 3.9 con 4 pilas recargables AAA con un voltaje final de 4.8 V, dentro de su rango de alimentación (1.8 V-5.5 V).

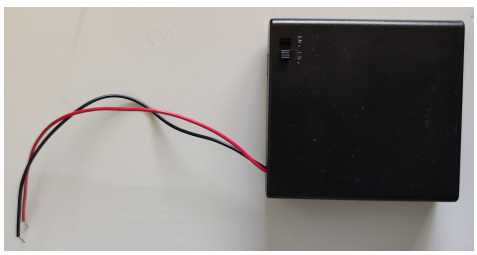


Figura 3.9: Portapilas para alimentar la Raspberry Pi

### 3.3.2. Diseño del circuito electrónico

Este circuito electrónico comienza recibiendo una señal cuadrada de 0 a 3.3 V procedente de la Raspberry Pi, tal y como se muestra en la figura 3.10.

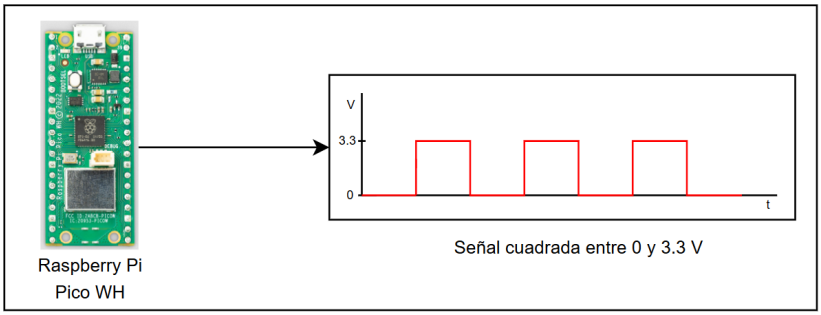



Figura 3.10: Señal PWM salida de la Raspberry Pi

Esta señal está modulada mediante PWM, pero la salida final debe ser un señal continua. Por lo tanto, es necesario un filtro que aproxime y convierta la señal continua aproximadamente lineal con respecto al duty del PWM, siendo aceptable que presente algo de rizado. Además, se debe solventar la limitación de salida de 3.3 V, ya que se requiere un voltaje de hasta 4.5 V (el máximo que generan los potenciómetros del mando). Para esto, se ha optado por utilizar un amplificador operacional configurado como amplificador no inversor.

El circuito se muestra en la figura 3.11 y combina las funciones de filtrado y amplificación y debe ser replicado tres veces, una para cada potenciómetro.

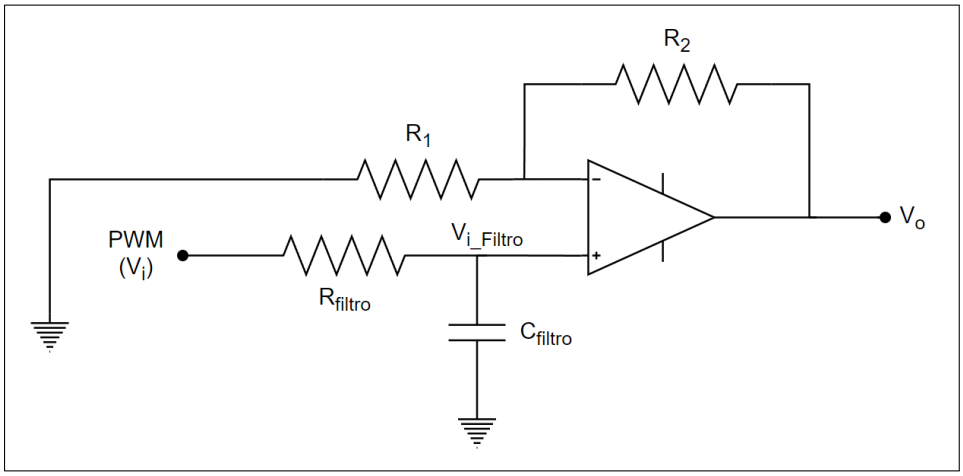



Figura 3.11: Circuito electrónico diseñado

Este circuito genera una señal continua amplificada mediante el siguiente proceso:

**Ciclo de trabajo** (DUTY) → **PWM** (Señal cuadrada de 3.3 V) → **Filtro** (Señal continua de 0V a 3.3 V) → **Amplificador Operacional** (Señal continua de 0V a 4.5 V)

A continuación, se muestra el diseño de las dos etapas necesarias.

**Etapa de filtrado**

Para asegurar el correcto funcionamiento de la señal proveniente de la Raspberry Pi y obtener una señal suavizada, aproximadamente lineal, se ha implementado un filtro pasivo RC de primer orden, de tipo paso bajo, como el de la figura 3.12.

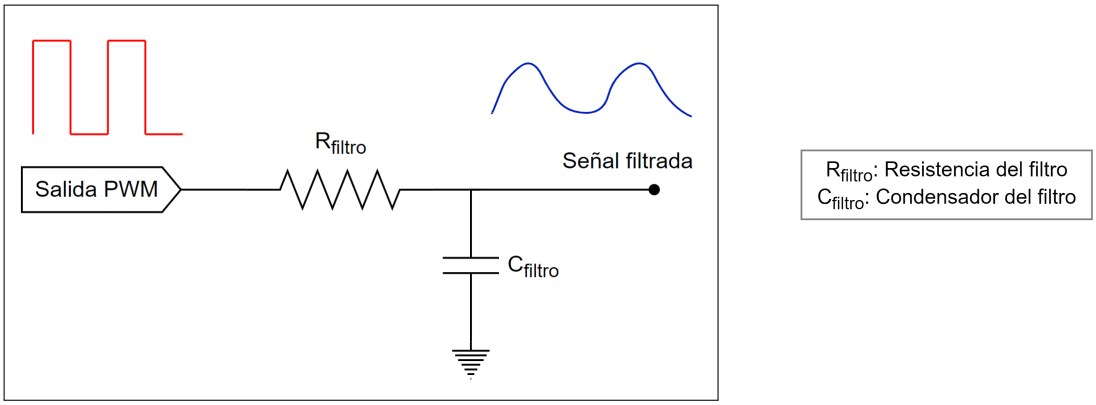



Figura 3.12: Filtro pasivo RC paso bajo de primer orden

Este filtro se caracteriza por permitir el paso de frecuencias bajas mientras atenúa las frecuencias altas, lo que ayuda a eliminar posibles ruidos o interferencias en la señal.

La frecuencia de la señal PWM generada por la Raspberry es 1000 Hz. Para asegurar un buen filtrado, se va a elegir la frecuencia de corte algo más de una década (factor de 20) por debajo de esta frecuencia de 1000 Hz, por lo que será 50 Hz. Además, se va a utilizar un condensador de 0,1 µF, por lo que el único componente por calcular es la resistencia del filtro:

$$R_{filtro} = \frac{1}{2\pi f_c C} \tag{3.1}$$

Sustituyendo los valores anteriores:

$$R_{filtro} = \frac{1}{2\pi \cdot 50 \cdot 0{,}1 \times 10^{-6}} = 31{,}83 \text{ k}\Omega \tag{3.2}$$

Dado que 31,83 kΩ no es un valor comercial, se ha optado por utilizar una resistencia de 47 kΩ, que es la más cercana entre las resistencias disponibles.

**Etapa amplificadora**

Para alcanzar un voltaje de hasta 4.5 V, se ha implementado un amplificador no inversor como el de la figura 3.13. Con el fin de proporcionar un margen adicional, el amplificador ha sido diseñado para ofrecer una salida de hasta 5 V.

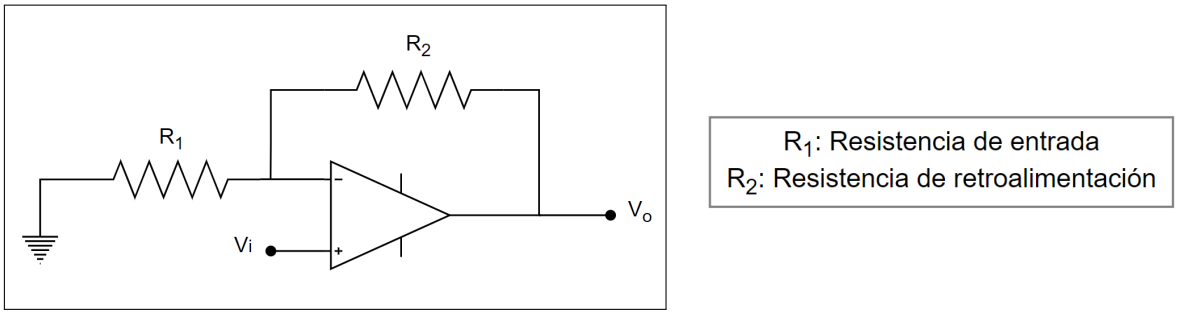



Figura 3.13: Amplificador no inversor

La ganancia del amplificador operacional en un filtro activo no inversor está determinada por la resistencia de entrada $R_1$ y la resistencia de retroalimentación $R_2$. La ecuación para la salida del amplificador es:

$$V_o = V_i \left(1 + \frac{R_2}{R_1}\right) \tag{3.3}$$

Resolviendo esta ecuación para una tensión de entrada de 3.3 V y una salida de 5 V, se obtiene que la relación entre $R_1$ y $R_2$ es:

$$\frac{R_2}{R_1} = 0{,}5151 \tag{3.4}$$

En la práctica, se suelen utilizar resistencias de valores del orden de decenas de kΩ para minimizar la corriente que pasa a través de ellas. Por lo tanto, se selecciona $R_1$ con un valor de 10 kΩ, obteniendo un valor para $R_2$ de 5,1 kΩ.

Para alimentar el amplificador operacional, se requiere una fuente de tensión externa que proporcione un voltaje superior a 7 V. En este caso, la salida de la fuente se ha configurado a 10 V.

### 3.3.3. Selección del conmutador

El último componente del sistema electrónico es el conmutador, que permite seleccionar entre el mando original o el nuevo sistema implementado. Se han utilizado tres conmutadores individuales, uno para cada potenciómetro. Este conmutador debe ser capaz de alternar entre el mando original y el sistema nuevo, permitiendo así controlar un único dispositivo (el dirigible) con dos opciones de conexión.

Para lograr esto, se empleará un **conmutador SPDT** (Single Pole, Double Throw) como el de la figura 3.14. Este tipo de conmutador permite conectar un terminal común (1P) a uno de los dos posibles terminales de salida: 1T (NC, Normalmente Cerrado) o 2T (NO, Normalmente Abierto). Dado que el sistema se monta en una placa de pruebas (Breadboard), se ha optado por utilizar un conmutador DIP (Dual In-line Package), que está diseñado en un formato adecuado para este tipo de montaje.


(a) Esquemático del conexionado (b) Foto del conmutador

Figura 3.14: Conmutador SPDT

Para realizar la conmutación, se necesita conectar el cable de control del mando en la pata central del conmutador. En las patas extremas, se conecta el cable de color proveniente de cada potenciómetro y, en la otra pata extrema, la salida obtenida del circuito electrónico. Además, se debe conectar la tierra de al menos uno de los potenciómetros a la misma tierra del circuito electrónico implementado, mientras que el cable de alimentación de cada potenciómetro debe permanecer conectado tal y como estaba originalmente.

## 3.4. Simulación del circuito

Para comprobar que el circuito que se ha diseñado funciona correctamente, antes de implementarlo físicamente, se ha simulado con la herramienta Simulink (en MATLAB).

La simulación consiste en modelar el circuito diseñado y probar diferentes entradas para simular cómo se comportaría al variar el ciclo de trabajo (Duty cycle), tal como se hará en la implementación real. El circuito se muestra en la figura 3.15 y se observa que se han utilizado 3 entradas diferentes: senoidal, escalón y triangular.

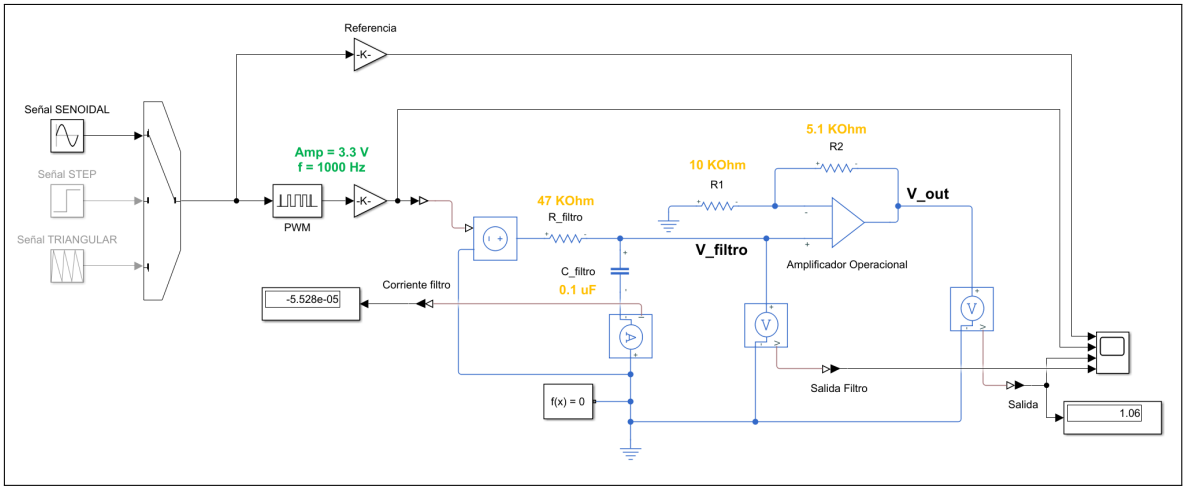



Figura 3.15: Circuito modelado en Simulink

Para la entrada senoidal, se obtienen las señales de la figura 3.16.

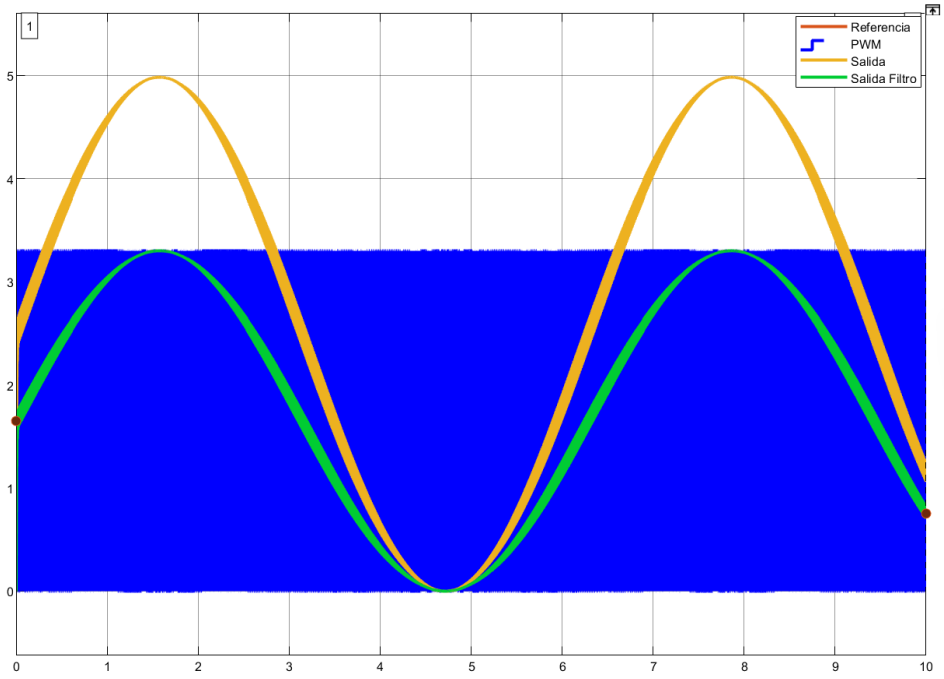



Figura 3.16: Señales para la entrada senoidal

A continuación, en la figura 3.17, se muestra la misma imagen con un zoom para visualizar con mayor detalle el efecto del filtro y el rizado presente en la señal de salida.

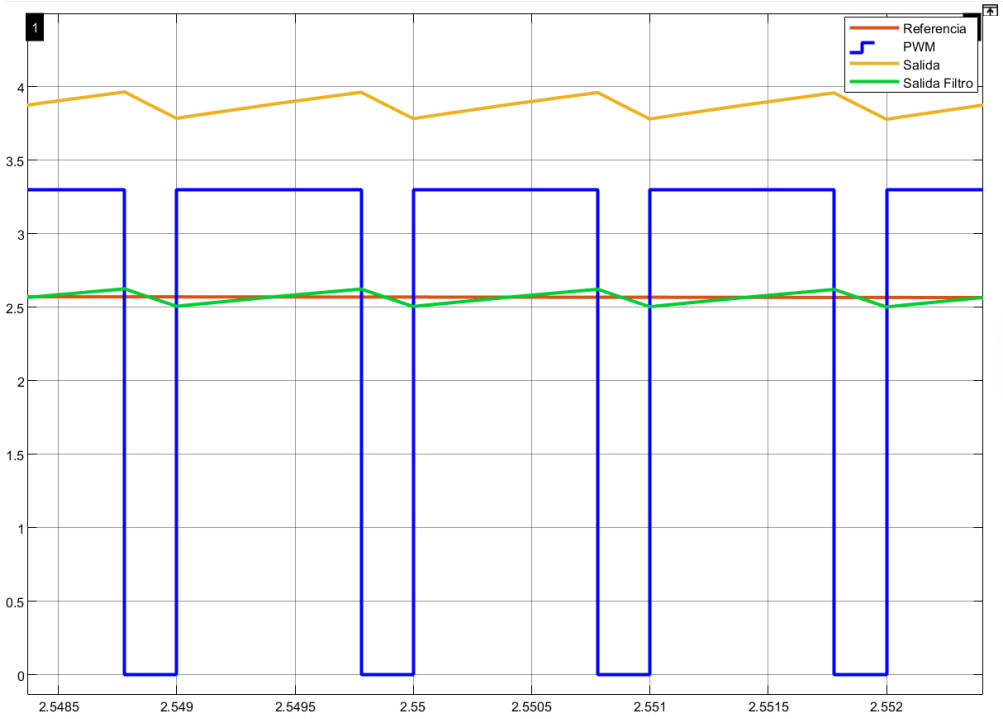



Figura 3.17: Señales para la entrada senoidal con zoom

Se puede observar que la señal después del filtro es similar a la señal de referencia, con un rizado suficientemente reducido.

Para la entrada escalón, se obtienen las señales de la figura 3.18.

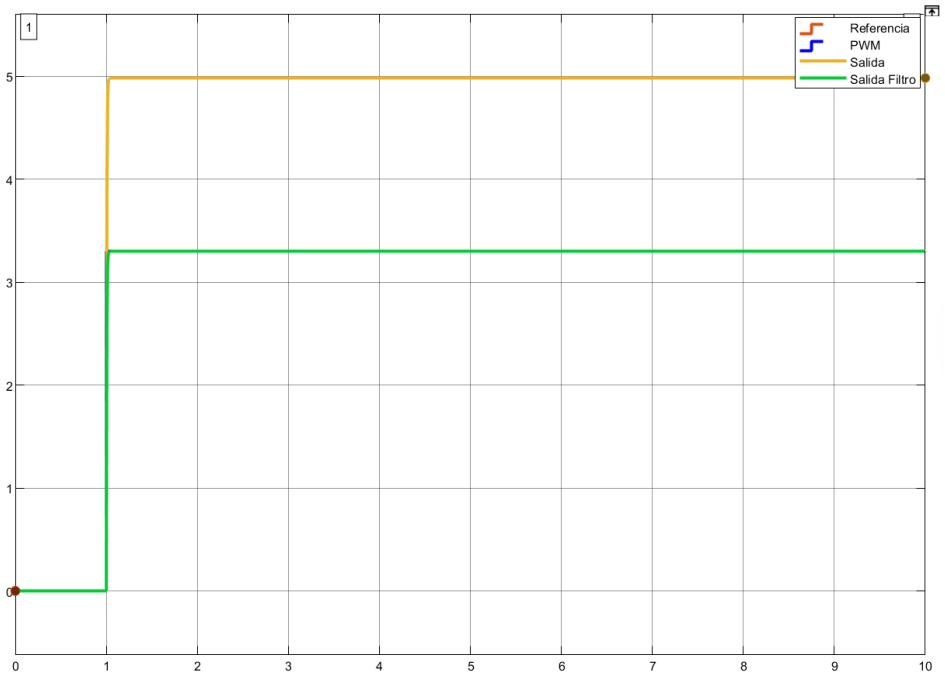



Figura 3.18: Señales para la entrada escalón

En la figura 3.19, se muestra la misma imagen pero con zoom.

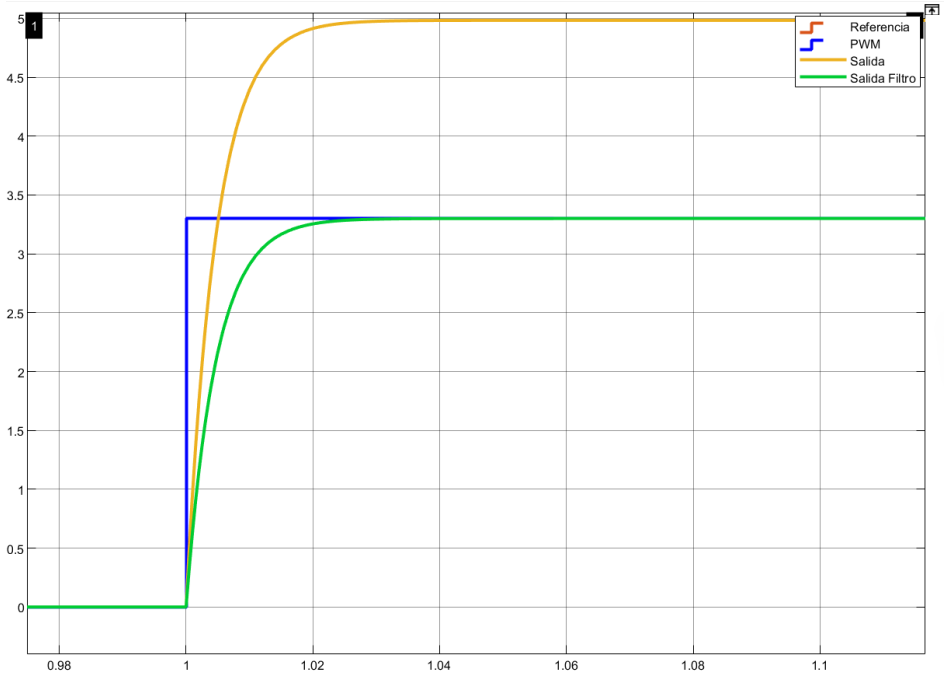



Figura 3.19: Señales para la entrada escalón con zoom

Aquí se aprecia que al buscar reducir el rizado, se produce un aumento del transitorio.

Para la entrada triangular, se obtienen las señales de la figura 3.20.

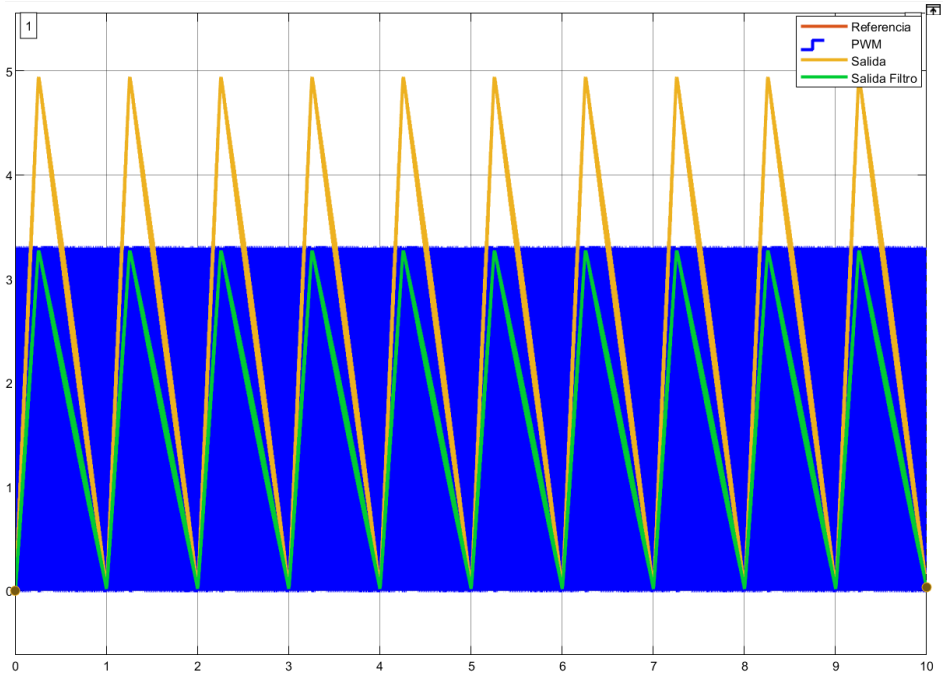



Figura 3.20: Señales para la entrada triangular

En la figura 3.21, se muestra la misma imagen pero con zoom.

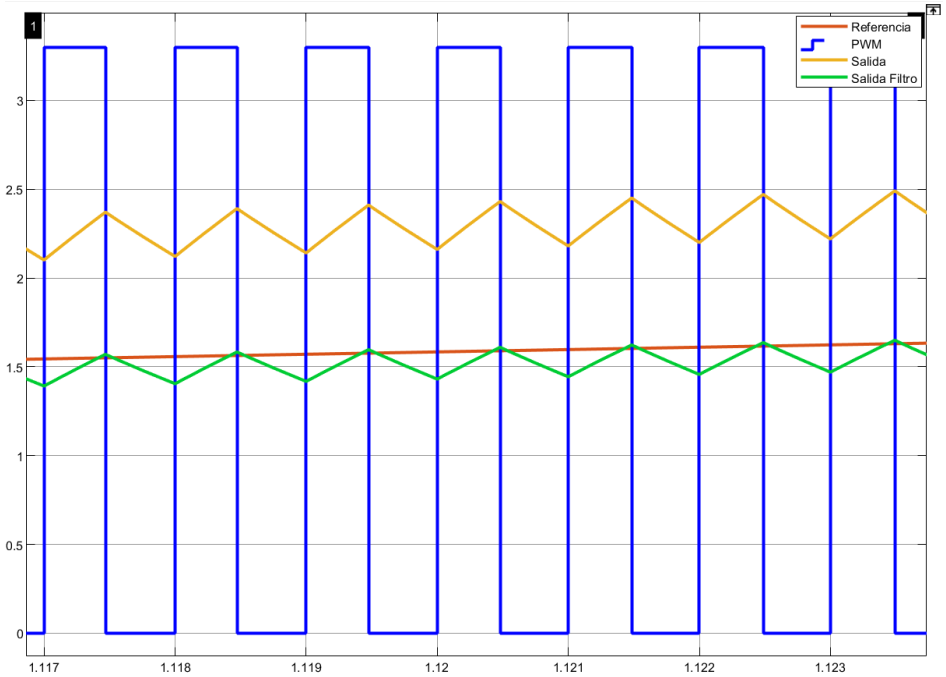



Figura 3.21: Señales para la entrada triangular con zoom

Posteriormente, se muestran las salidas obtenidas para cada una de las 3 entradas con sus valores máximos para comprobar si realmente funcionan correctamente.

En la figura 3.22 para la entrada senoidal.

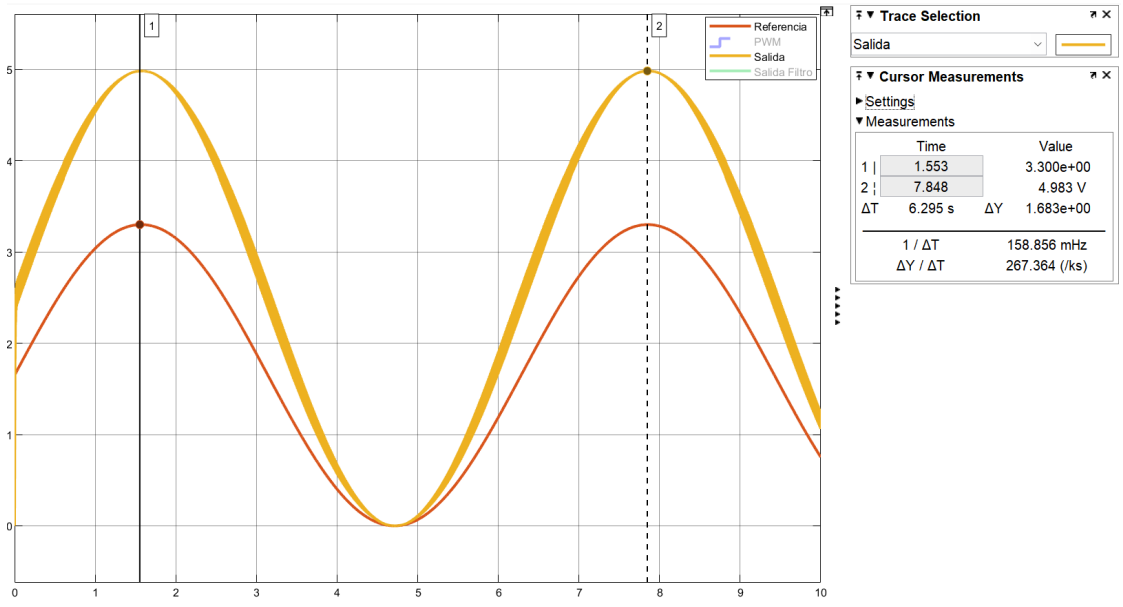



Figura 3.22: Valor máximo de salida para la entrada senoidal

En la figura 3.23 para la entrada escalón.

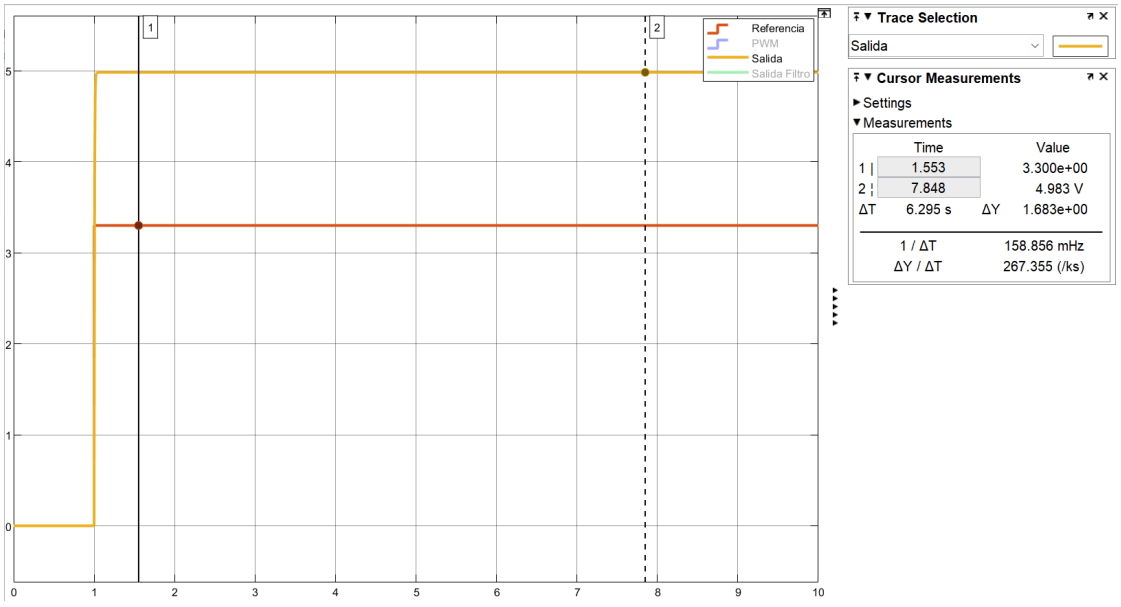



Figura 3.23: Valor máximo de salida para la entrada escalón

En la figura 3.24 para la entrada triangular.

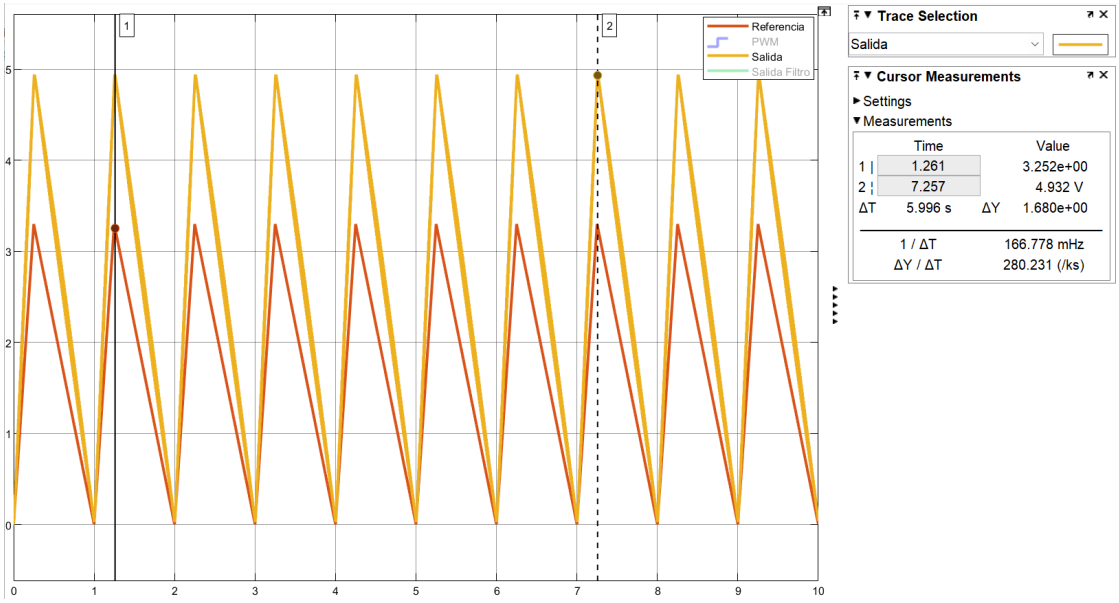



Figura 3.24: Valor máximo de salida para la entrada triangular

Tal y como se detalla en el capítulo 3.3.2, el objetivo es alcanzar una salida de 5 V aplicando la máxima tensión posible desde la Raspberry Pi (3.3 V).

Para las entradas senoidal y de escalón, el valor máximo obtenido es de 4.983 V, mientras que para la entrada triangular es de 4.932 V. Estos valores están muy cerca de los 5 V deseados y son suficientemente precisos para el control de los motores. Las ligeras variaciones se deben a las tolerancias de las resistencias utilizadas, que hace que los valores reales difieran ligeramente de los teóricos.

## 3.5. Representación gráfica del sistema electrónico

En la figura 3.25 se muestra el sistema implementado.

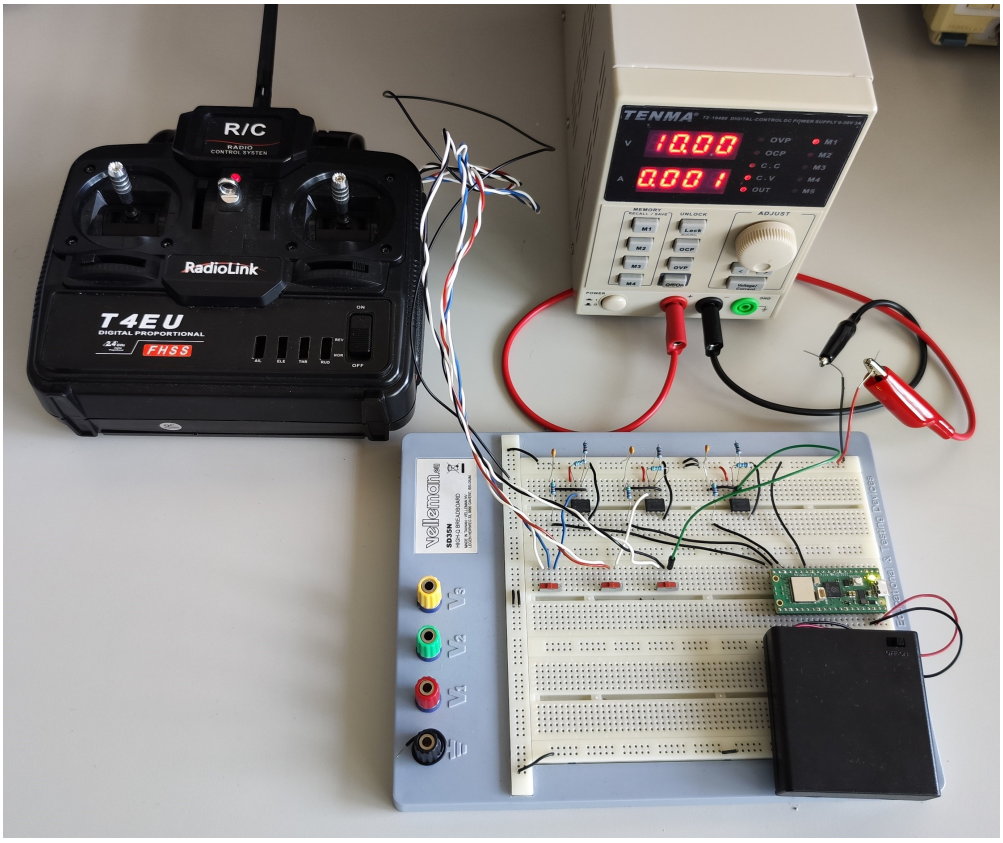



Figura 3.25: Circuito montado y alimentado

Se puede observar que el circuito electrónico está compuesto de tres circuitos individuales para el control de cada movimiento (vertical, lateral y frontal), diferenciados por el color de los cables.

Para obtener instrucciones detalladas sobre la puesta en funcionamiento, se puede consultar el anexo A, que contiene el manual de usuario. Este manual proporciona una guía paso a paso para realizar el montaje correctamente.

En el capítulo 4 se explica el desarrollo de la interfaz de control diseñada para poder controlar el dirigible mediante Wi-Fi.

# Capítulo 4

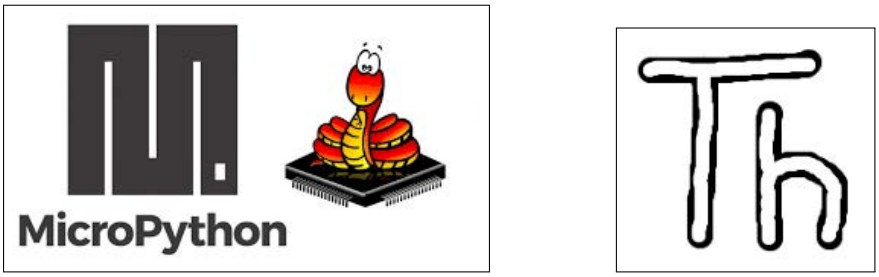
# Desarrollo e implementación de la interfaz de control

*En este capítulo se explica toda la parte software del trabajo. Primero, se explican las funciones que se han implementado para controlar el dirigible. A continuación, el diseño de una página web que permite actuar sobre el sistema por medio de una interfaz gráfica y su comunicación con la Raspberry Pi. Por último, se realiza una descripción de la interfaz de control.*

## 4.1. Implementación del control mediante la Raspberry Pi

Para llevar a cabo el desarrollo del control, se ha optado por utilizar MicroPython, una herramienta diseñada específicamente para funcionar en dispositivos con recursos limitados, como es el caso de la Raspberry Pi Pico.

Como editor, se ha elegido Thonny, un entorno de desarrollo integrado (IDE) ampliamente utilizado para proyectos basados en MicroPython.




(a) MicroPython (b) Thonny

Figura 4.1: Herramientas utilizadas

En esta sección, se clasifican las funciones en tres categorías. La primera categoría incluye las funciones destinadas a gestionar el proceso completo de conexión Wi-Fi. La segunda categoría abarca funciones auxiliares, diseñadas para controlar las características integradas de la Raspberry Pi, como el control del LED y el sensor de temperatura. La tercera categoría se enfoca en las funciones responsables del control del dirigible.

Los archivos correspondientes a estas tres categorías de funciones tienen una extensión de aproximadamente 250 líneas de código.

### 4.1.1. Funciones para la conexión

Para realizar la conexión, se dispone de 2 archivos: *wifiRedes.json* y *wifiConfig.py*. En el primero, se almacenan redes Wi-Fi que se quieran utilizar (nombre y contraseña). En el segundo, se encuentran las funciones necesarias para realizar la conexión Wi-Fi, obtener los credenciales y desconectar la Wi-Fi.

- ***conectar_wifi***: Recibe como parámetros el nombre de la red Wi-Fi y la contraseña. Esta función realiza un intento de conexión durante 30 segundos mientra parpadea el LED cada segundo. Si la conexión es exitosa, el LED se queda encendido al conectarse para indicarlo. En caso contrario, se apaga.

- ***obtener_credenciales***: Esta función recibe como parámetro la red a la que se quiere conectar y tras revisar el archivo *wifiRedes.json*, devuelve el nombre y la contraseña de la red Wi-Fi.

- ***desconectar_wifi***: Esta función realiza el proceso de desconexión y apaga el LED de la Raspberry Pi.

### 4.1.2. Funciones auxiliares

Estas funciones se encuentran en el archivo *raspberryFunciones.py* y son las siguientes:

- ***encenderRaspLED***: Sirve para encender el LED.

- ***apagarRaspLED***: Sirve para apagar el LED.

- ***devuelveTemperatura***: Devuelve la temperatura del sensor de la Raspberry Pi.

### 4.1.3. Funciones para el control del dirigible

En otro archivo, *dirigible.py*, se han implementado diferentes funciones para controlar el dirigible. Para ello, se ha creado un objeto para tener todo encapsulado. Primero, se han configurado los pines para controlar cada circuito (GP13, GP14 y GP15) y su frecuencia, 1000 Hz.

Para controlar el dirigible, se han implementado funciones desde 2 enfoques diferentes. En el primer enfoque, se ha querido controlar exactamente el voltaje que se va a aplicar en la salida:

- ***configurarVsalida***: Esta función recibe como parámetros el motor y el voltaje deseado. Primero, verifica que el voltaje esté dentro de los límites permitidos (0-4.5 V). Si el voltaje excede estos límites, ajusta el valor al máximo o mínimo permitido según corresponda, antes de aplicarlo a la salida.

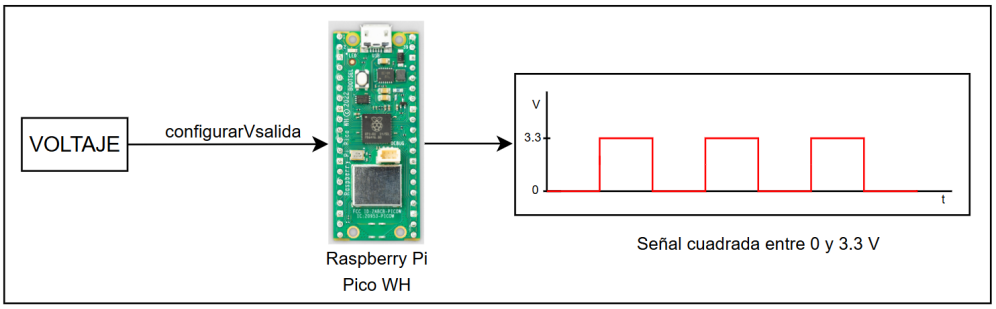



Figura 4.2: Esquema para configurar un voltaje de salida

A partir de esta función, se han implementado 4 funciones para detener los motores individualmente o todos a la vez. Para ello, se hace uso de los rangos de funcionamiento obtenidos en la sección 3.2.1.

- ***DetenerMotorSubirBajar***: Detiene el motor de subir-bajar.
- ***DetenerMotorIzquierdaDerecha***: Detiene el movimiento de los motores de izquierda-derecha.
- ***DetenerMotorAvanzarRetroceder***: Detiene el movimiento de los motores de avanzar-retroceder.
- ***DetenerTodo***: Detiene los 3 movimientos (los 3 motores de las 3 hélices).

En el segundo enfoque, las funciones implementadas corresponden a los seis movimientos posibles del dirigible, recibiendo como argumento un porcentaje de velocidad deseada, que varía entre 0 y 100. Para llevar a cabo estos movimientos, se utiliza la función *configurarVsalida*, junto con los valores obtenidos en la sección 3.2.1.

Estas funciones son: *Subir*, *Bajar*, *Izquierda*, *Derecha*, *Avanzar* y *Retroceder*.

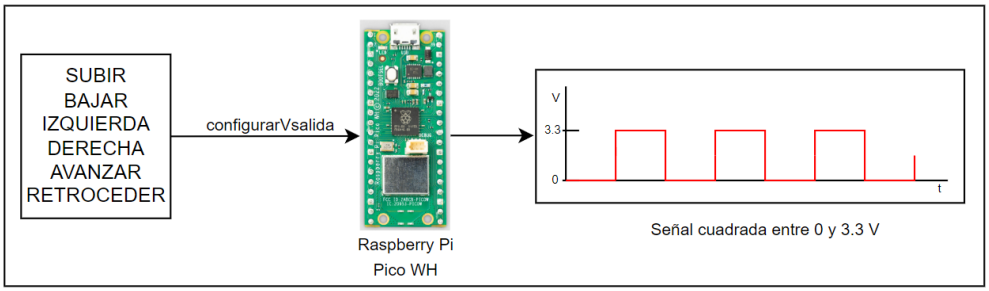



Figura 4.3: Esquema para seleccionar un movimiento

Además de las funciones ya descritas, este archivo también contiene las implementaciones para generar distintas señales, accesibles desde el menú de inicio. Estas señales se han utilizado para llevar a cabo las pruebas iniciales.

## 4.2. Sistema de comunicación

Para que la Raspberry Pi se conecte automáticamente al alimentarla, se utilizan los archivos *boot.py* y *main.py*. En *boot.py*, se encuentra el proceso de conexión Wi-Fi que emplea las funciones descritas anteriormente. Primero, se selecciona la red a la que se desea conectarse. Luego, se obtienen los credenciales mediante la función *obtener_credenciales* y, finalmente, se establece la conexión utilizando la función *conectar_wifi*.

El archivo *main.py*, que tiene aproximadamente 250 líneas de código, es donde se encuentra el código principal y desde donde se gestiona toda la comunicación entre la Raspberry Pi y la página web desde el lado de la Raspberry Pi.

### 4.2.1. Selección del marco web

Tras estudiar diferentes alternativas para la comunicación, **se ha decidido utilizar MicroDot**, un marco web Python diseñado específicamente para ejecutarse en sistemas con recursos limitados, como la Raspberry Pi Pico. La elección de MicroDot se ha basado en su capacidad para gestionar eficientemente la comunicación a través de WebSockets, proporcionando una solución liviana y optimizada.

Para llevar a cabo la comunicación, se ha optado por utilizar Websockets, un protocolo ideal para aplicaciones que requieren intercambio de mensajes en "tiempo real" y actualizaciones asíncronas, como es el caso de este trabajo. Este protocolo permite intercambios bidireccionales *full-duplex* entre un cliente (generalmente un navegador web) y un servidor web.

Para este trabajo no hace falta una comunicación bidireccional pero como el objetivo de la línea de investigación es tener un control automático, sí que será necesario. Por ese motivo, para facilitar trabajo futuro, ya se implementa este tipo de comunicación.

### 4.2.2. Implementación de Microdot

Los archivos Microdot son esenciales para la implementación de la comunicación entre el servidor y la página web.

- **Clase *microdot.py***: Implementa una instancia de aplicación HTTP.
- **Clase *websocket.py***: Procesa un protocolo de enlace de actualización de WebSocket.
- **Clase *microdot.py***: Inicializa el subsistema de plantillas y renderiza una plantilla.

Además de las clases *helpers.py* y *__init__.py*.

En la siguiente sección, se explicará de manera concisa, el método de comunicación entre la página web y el servidor. Mientras que el HTML y el CSS tienen cierta relevancia en el aspecto visual de la página web, es el funcionamiento del JavaScript lo que garantiza una conexión correcta y sin problemas entre las distintas partes.

### 4.2.3. Comunicación entre la página web y la Raspberry Pi

Como se ha comentado anteriormente, el archivo *main.py* es el principal y facilita la comunicación entre la Raspberry Pi y la página web. La comunicación se establece mediante el envío de mensajes desde la página web utilizando JavaScript, cuyo detalle se explorará en la sección siguiente dedicada al diseño de la página web.

En el código de la Raspberry Pi, el proceso consiste en esperar la recepción de mensajes. Dependiendo del contenido del mensaje, se realiza la configuración correspondiente, ya sea para controlar una sección específica o para ejecutar una acción determinada.

## 4.3. Diseño de la página web

### 4.3.1. Entorno de programación

Es importante que la página web esté diseñada de manera adecuada para que se logre un correcto entendimiento y que el dirigible responda de la manera en la que se ha implementado. Para desarrollarla, se ha utilizado Visual Studio Code como entorno de desarrollo, empleando una combinación de HTML, CSS y JavaScript.

- **HTML**: Para establecer la estructura fundamental de la página.
- **CSS**: Para el diseño y la aplicación de estilos para mejorar la presentación visual.
- **JavaScript**: Para agregar interactividad y dinamismo a los diferentes elementos de la página, mejorando así la experiencia del usuario.


Figura 4.4: Lenguajes de programación utilizados para el diseño web

Todo el código es original y ha sido escrito por mí, salvo el pad de la sección 4.3.5 que ha sido adaptado a partir de un recurso de Internet para integrarlo en la página web.

El código HTML ha ocupado unas 600 líneas, el de de CSS más de 2300 y el de JavaScript más de 1500, lo que hace un total aproximado de 4500 líneas de código.

### 4.3.2. Descripción general

La página web consta de tres partes principales: una barra de navegación en la parte superior, un pie de página en la parte inferior y, entre ambos, cuatro secciones distintas. Al acceder a la página, se muestra automáticamente la sección de inicio (DIRIGIBLE).

A continuación, se describen los elementos comunes a todas las secciones (la barra de navegación y el pie de página). Posteriormente, en las secciones 4.3.3, 4.3.4, 4.3.5 y 4.3.6, se detalla el funcionamiento de las cuatro secciones de la página web que permiten los diferentes tipos de control.

#### Barra de navegación

En la figura 4.5 se muestra la barra de navegación, que está en la parte superior de la página y aparece en todas las secciones.

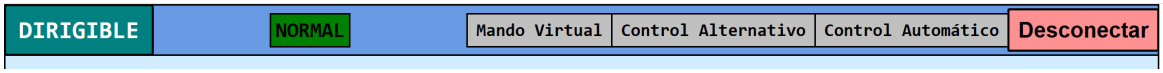



Figura 4.5: Barra de navegación

Esta barra permite desplazarse entre las diferentes secciones. En la figura 4.6 se observa el **botón de inicio** (DIRIGIBLE), situado a la izquierda, para volver a la sección inicial y los **tres botones para seleccionar el tipo de control** que se quiere utilizar.


(a) Botón de Inicio


(b) Botones de Secciones

Figura 4.6: Secciones de control del dirigible

Además, dispone de un **botón de emergencia** que permite detener rápidamente los tres motores en caso de que el dirigible se descontrole o le ocurra algo inesperado. En la figura 4.7 se pueden observar los dos estados de funcionamiento del dirigible. Al inicio, está en el estado NORMAL (de color verde) pero si se pulsa, pasa a EMERGENCIA (de color rojo) y no permite realizar ningún movimiento. Al pulsarlo de nuevo, vuelve al estado NORMAL.


(a) Estado NORMAL


(b) Estado EMERGENCIA

Figura 4.7: Estados de funcionamiento del dirigible

Por último, está el **botón para desconectar** que permite realizar la desconexión. Al pulsarlo, se muestra el mensaje de la figura 4.8 indicando que se está desconectando y que se debe reiniciar el dispositivo para volver a conectarse.

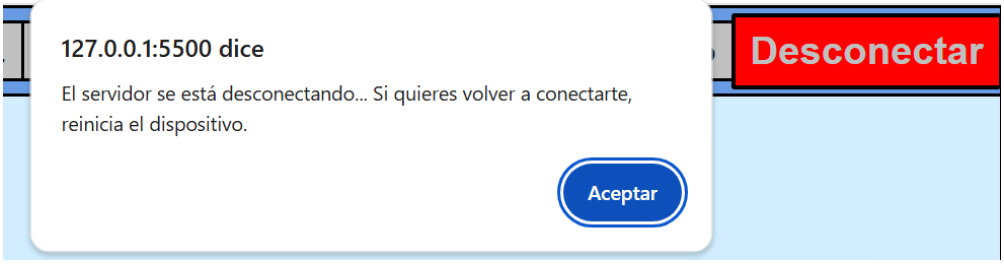



Figura 4.8: Mensaje de desconexión

### Pie de página

En la figura 4.9 se muestra el pie de página. Al igual que la barra de navegación, también aparece en todas las secciones.

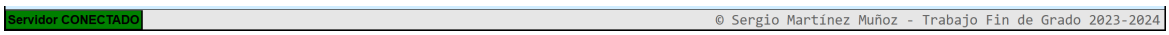



Figura 4.9: Pie de página

En la esquina inferior izquierda, se indica en un pequeño rectángulo el estado del servidor. Se muestra en verde si el servidor está conectado o en rojo si está desconectado, tal y como se aprecia en la figura 4.10.


(a) Servidor conectado


(b) Servidor desconectado

Figura 4.10: Estado del servidor

Esto es importante, ya que si la conexión no es buena, el servidor puede desconectarse. En tal caso, el rectángulo se vuelve rojo y al pulsarlo, se realiza un reconexión para volver a conectarse al servidor.

### 4.3.3. Sección de inicio (DIRIGIBLE)

Esta sección, que se observa en la figura 4.11, es la que se muestra al conectarse. Proporciona un esquema del sistema implementado y sus componentes. Además, indica información general como la fecha, la hora, el tiempo de permanencia en la página y la temperatura.

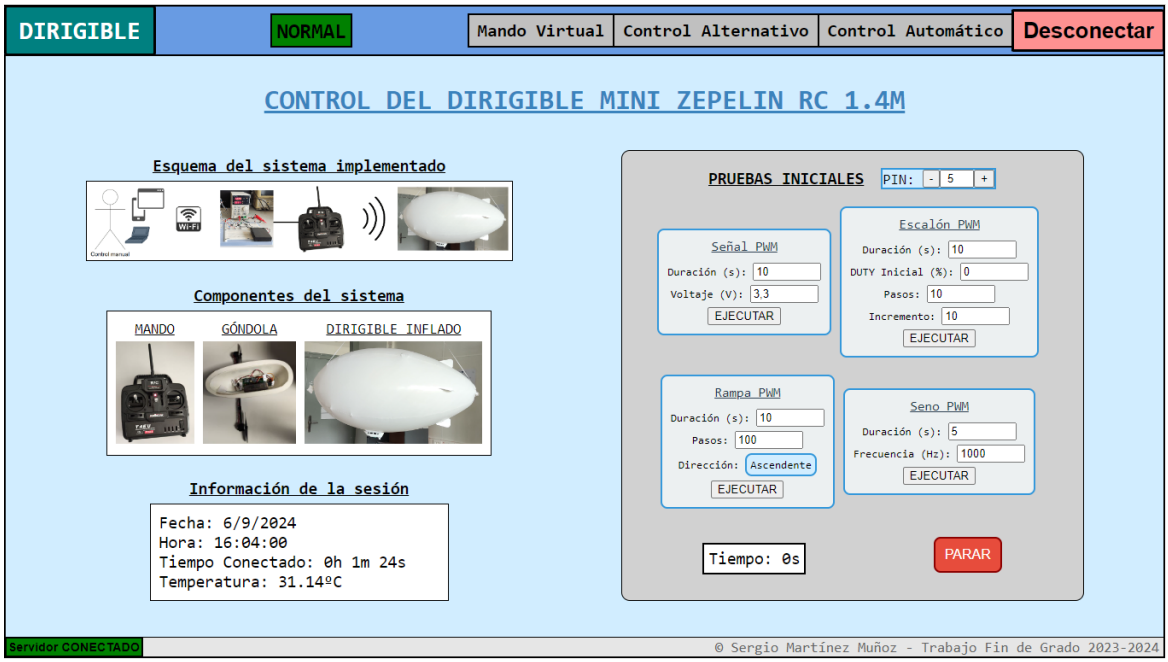



Figura 4.11: Sección de inicio

Asimismo, se ha implementado un apartado dedicado a la realización de las pruebas iniciales. En él, se puede seleccionar el PIN de la Raspberry Pi para generar la señal deseada, así como el tipo de señal. Es posible generar señales PWM, escalón, rampa o seno a partir de una señal PWM generada por la Raspberry Pi, y personalizar la duración y las características específicas de esta señal.

### 4.3.4. Sección Mando Virtual

Esta sección, que se observa en la figura 4.12, ha sido diseñada para replicar el funcionamiento del mando físico original. Este mando virtual es totalmente funcional, permitiendo controlar el dirigible de manera idéntica al mando original, ya que su comportamiento es similar. Además, se ha incorporado un "cuadro de mandos" que muestra en tiempo real el voltaje aplicado en cada movimiento. Este cuadro permite ajustar el voltaje en incrementos o decrementos de 0.10 V o 0.01 V, ofreciendo un control preciso sobre el voltaje aplicado.

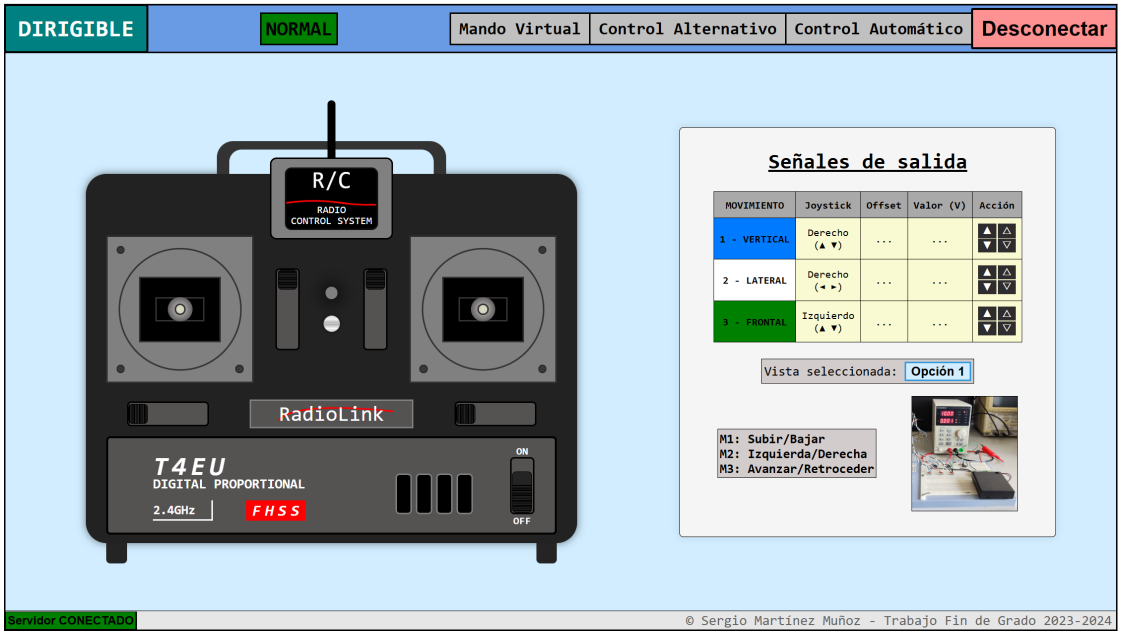



Figura 4.12: Sección Mando Virtual

Para enviar las señales, se pueden mover los *sticks* o cambiar los valores directamente con el "cuadro de mandos". Cabe resaltar, que al variar los valores con los botones de la tabla, se mueven los *sticks* en la dirección correcta.

Al ser totalmente funcional, también están implementados los *offsets*, por lo que el voltaje final que se envía también depende de ellos. Con el *offset* al 0 %, se puede configurar un voltaje entre 0 y 3.8 V y con el *offset* al 100 %, entre 0.7 V y 4.5 V, por lo que el rango total es entre 0 V y 4.5 V, similar al mando original.

| MOVIMIENTO | Joystick | Offset | Valor (V) |
|---|---|---|---|
| 1 - VERTICAL | Derecho (▲ ▼) | 0% | 3.80 |

(a) Señal con el offset al 0 %

| MOVIMIENTO | Joystick | Offset | Valor (V) |
|---|---|---|---|
| 1 - VERTICAL | Derecho (▲ ▼) | 100% | 4.50 |

(b) Señal con el offset al 100 %

Figura 4.13: Voltaje máximo con offset al 0 % y al 100 %

Al acceder a esta sección, el mando está inicialmente apagado y en el "cuadro de mandos" no se muestra ningún valor. Si se pulsa el botón ON/OFF, el mando se enciende y se ilumina el indicador LED en rojo, tal y como se muestra en la figura 4.14.


Figura 4.14: Mando Virtual encendido

Se puede observar que los *sticks* se han desplazado de su posición inicial en el centro. Esto se debe a que, como se ha explicado en la sección 3.2.1, para que estén detenidos los tres motores, es necesario enviar una señal distinta de 0 V. Si no se hubiera configurado de esta manera, al encender el mando, el dirigible se movería. En la figura 4.15 se pueden ver en la tabla los valores iniciales para que el dirigible esté detenido.

| MOVIMIENTO | Joystick | Offset | Valor (V) | Acción |
|---|---|---|---|---|
| 1 - VERTICAL | Derecho (▲ ▼) | 0% | 2.10 | ▲ △ ▼ ▽ |
| 2 - LATERAL | Derecho (◄ ►) | 0% | 2.75 | ▲ △ ▼ ▽ |
| 3 - FRONTAL | Izquierdo (▲ ▼) | 0% | 1.50 | ▲ △ ▼ ▽ |

Figura 4.15: Tabla con los valores iniciales

Además de la tabla principal, existen dos formas alternativas de visualizar el movimiento de cada motor. En la primera visualización, la de la figura 4.16, se muestra un recuadro gris con los movimientos posibles organizados en parejas. Si el dirigible se mueve en una dirección, la palabra correspondiente se ilumina; de lo contrario, ambas palabras aparecen sin resaltar. También se incluye una imagen real del sistema implementado.

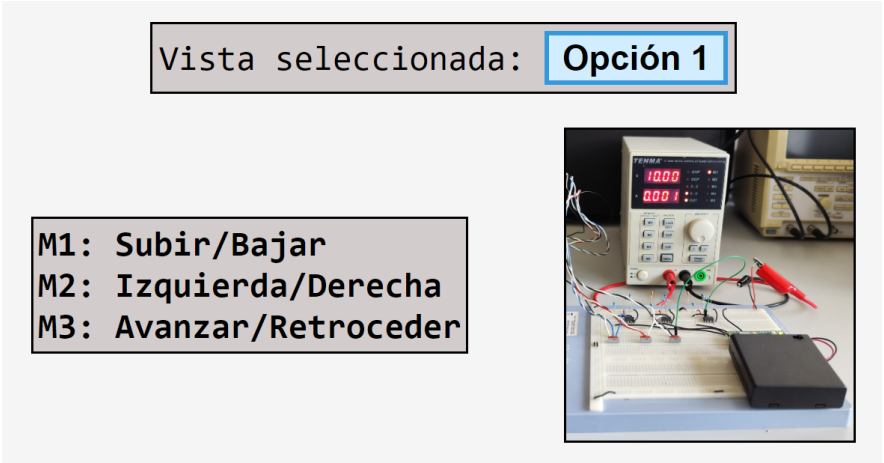



Figura 4.16: Opción 1 de la vista de los movimientos

En la segunda visualización, la de la figura 4.17, cada movimiento se presenta de manera individual, acompañado de su respectiva fotografía. En este caso sólo se muestra la palabra correspondiente al movimiento actual, si el dirigible está detenido, aparecerá la palabra DETENIDO.


Figura 4.17: Opción 2 de la vista de los movimientos

En la figura 4.18 se muestra un ejemplo de movimiento de los *sticks* con la vista 1.

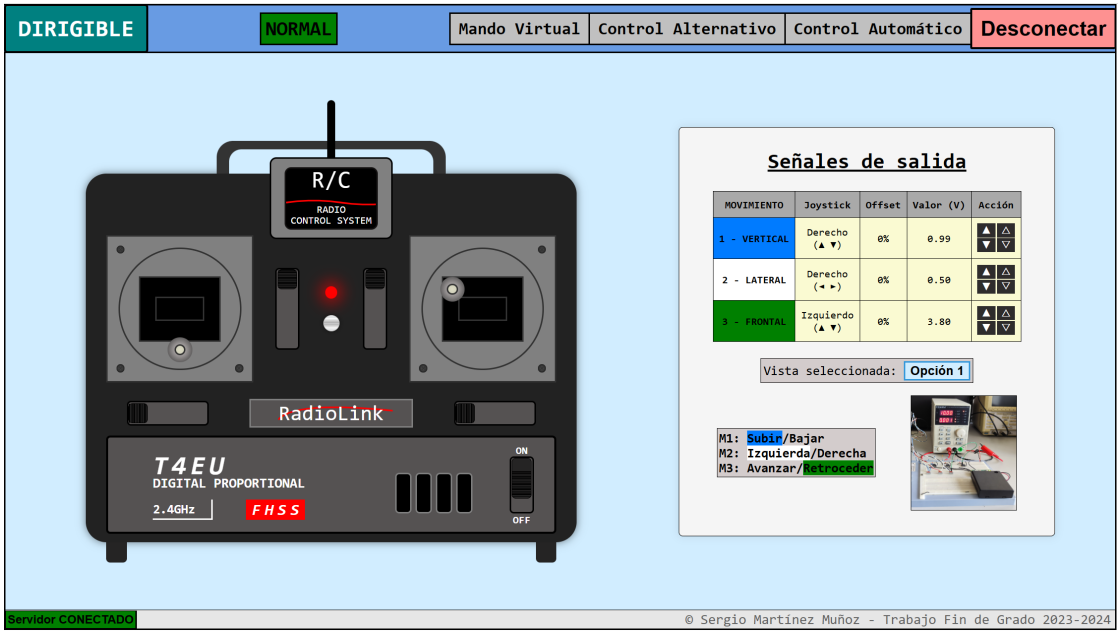



Figura 4.18: Ejemplo de uso del mando virtual con vista 1 y sin *offsets*

En la figura 4.19 se muestra otro ejemplo con la vista 2 y utilizando los *offsets*.

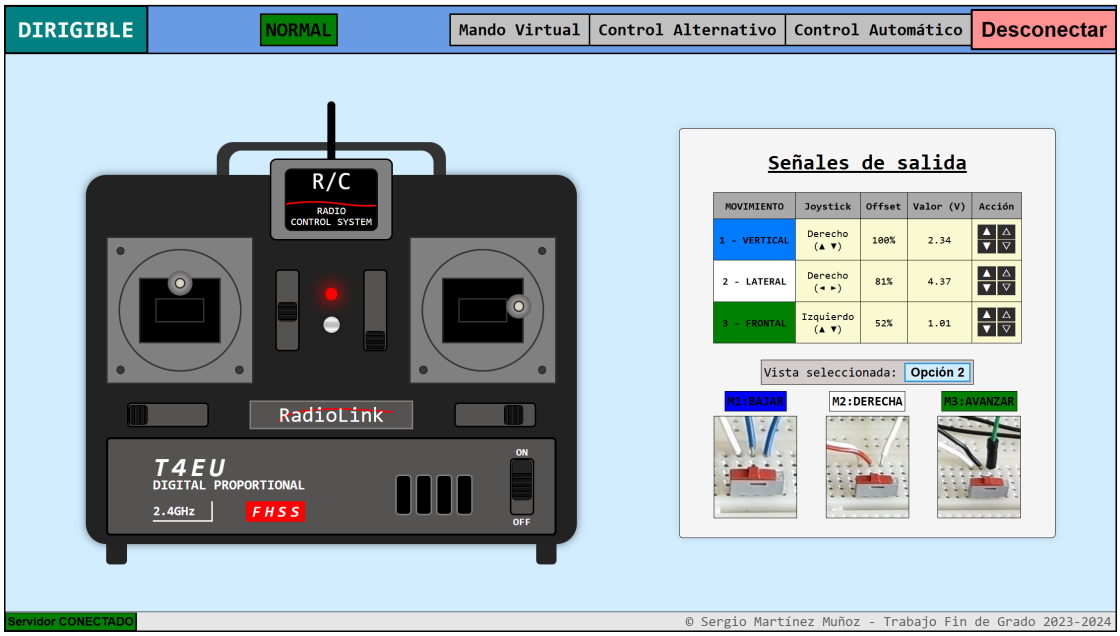



Figura 4.19: Ejemplo de uso del mando virtual con vista 2 y con *offsets*

### 4.3.5. Sección Control Alternativo

Esta sección representa un paso intermedio entre el objetivo final del control automático y la sección anterior de un control manual similar al original. En este caso, se ofrecen dos controles alternativos: por voz y mediante pad y ajuste de altura.

El control con pad [13] permite gestionar el movimiento diferenciando entre el movimiento de altura (subir y bajar) y desplazamientos direccionales (avanzar, retroceder, izquierda y derecha). Mientras se mantiene presionado, el pad controla la dirección del movimiento, y se detiene al soltarlo. La velocidad se ajusta mediante una barra, mientras que la altura se mantiene constante en el porcentaje configurado, con la posibilidad de ajustarla utilizando botones que incrementan o disminuyen la altura.

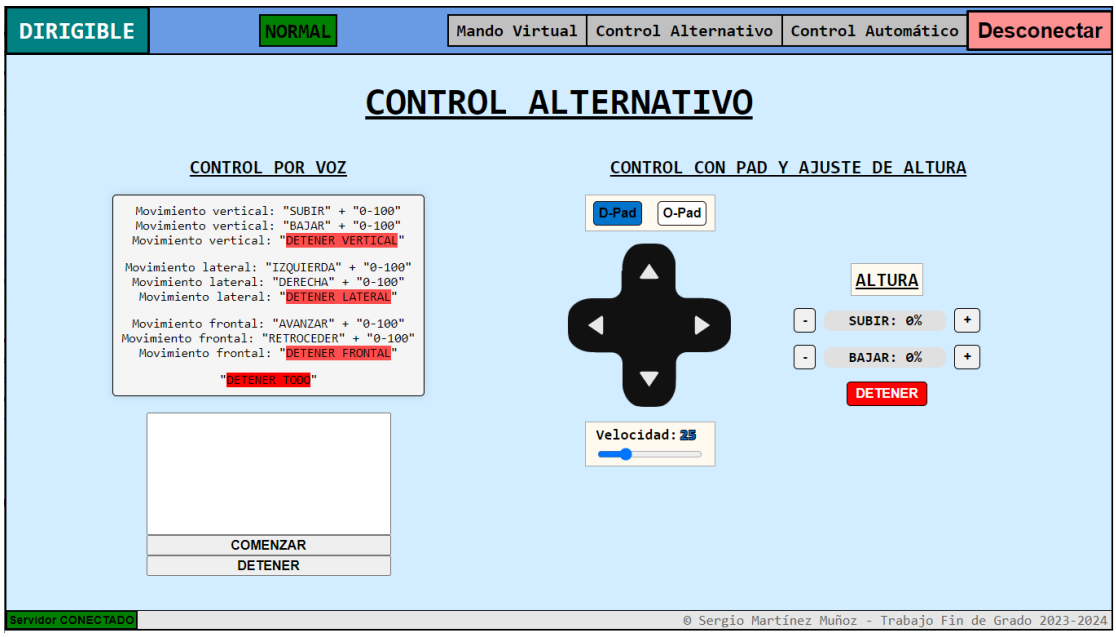
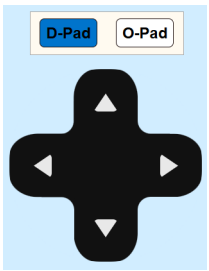
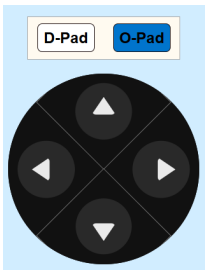



Figura 4.20: Sección Control Alternativo

Se puede observar en la figura 4.20 que hay dos botones, D-Pad y O-Pad, que sirven para cambiar entre los dos pad disponibles. En la figura 4.21 se muestra su comparación.



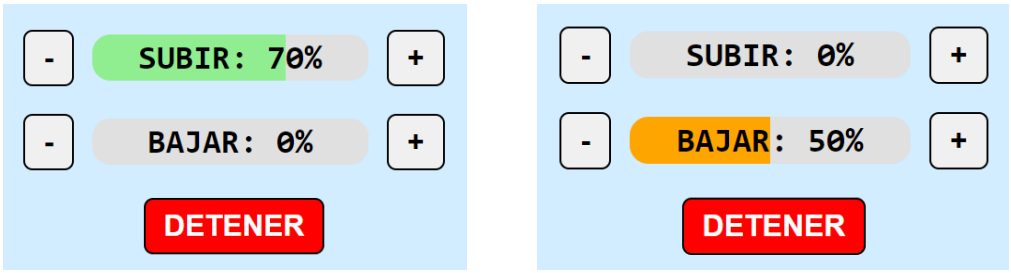

(a) D-Pad




(b) O-Pad

Figura 4.21: Comparación de los dos pad

El control de la altura se realiza mediante las dos barras para subir o bajar que controlan el movimiento de la hélice individual en un sentido u otro y su velocidad. En la figura 4.22 se muestran un ejemplo para subir y otro para bajar. Además, está el botón de detener que pone a 0 % el movimiento que esté activado.




(a) Movimiento para subir (b) Movimiento para bajar

Figura 4.22: Control del movimiento para subir o bajar

Asimismo, se cuenta con un control por voz que permite controlar el dirigible mediante unos sencillos mensajes. Aunque este método no es tan preciso, ya que no siempre detecta todos los comandos y la respuesta es más lenta, es otra alternativa posible de control. Para poder utilizarla, tal y como se indica en el anexo A.4, se debe activar una función experimental en el navegador ya que se tiene que dar permiso para que utilice el micrófono una página web con el protocolo HTTP.

Este control se activa al pulsar el botón COMENZAR, momento en el cual se inicia la escucha. Los mensajes capturados se transcriben en el área de texto mostrada en la figura 4.23, lo que permite verificar el contenido detectado. Para finalizar el control, simplemente se pulsa el botón DETENER, lo que detendrá la escucha y los motores del dirigible.

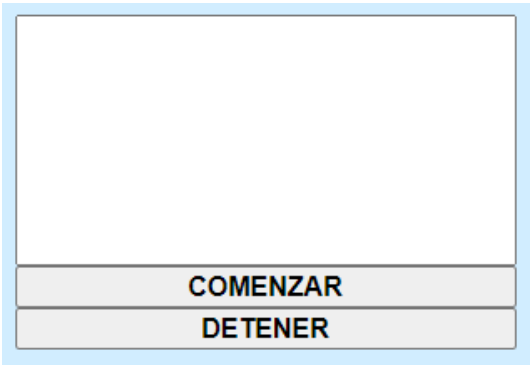



Figura 4.23: Área de texto del control por voz

### 4.3.6. Sección Control Automático

Esta sección está orientada a uno de los objetivos a largo plazo del proyecto: permitir que el dirigible se mueva controlado por un sistema automático. Desde aquí, es posible definir diferentes trayectorias que el dirigible ejecutará de manera autónoma y seleccionar la velocidad a la que se ejecutarán.

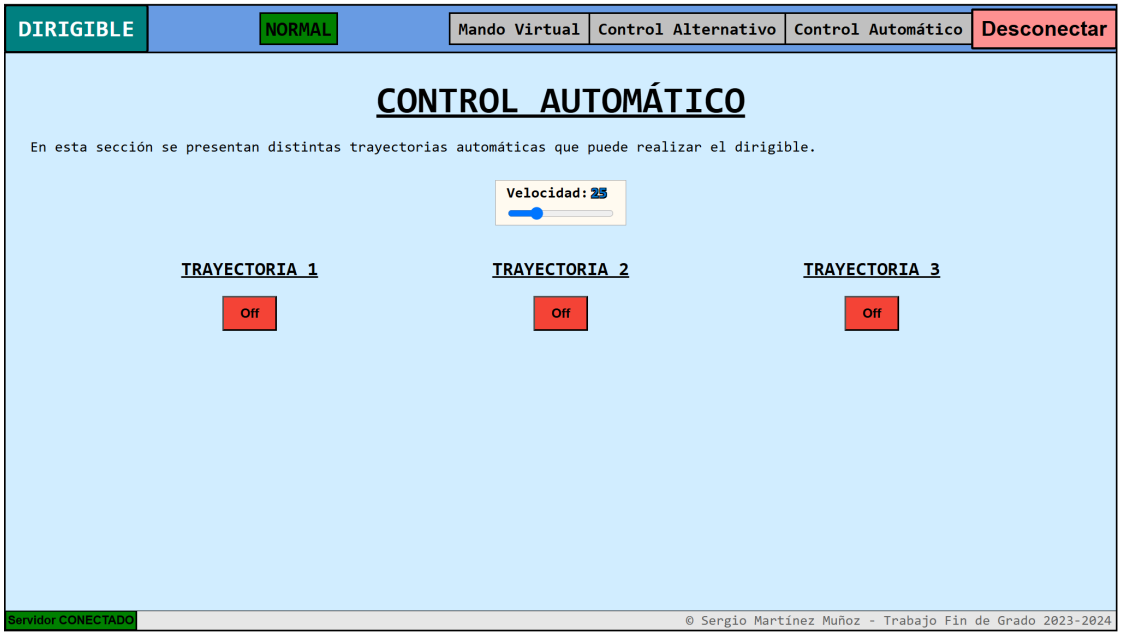



Figura 4.24: Sección de control automática

En la figura 4.25 se aprecia la diferencia de los botones cuando está realizando una trayectoria (en color verde) y cuando está parado (en color rojo).


(a) Trayectoria ON (b) Trayectoria OFF

Figura 4.25: Comparación de trayectoria ON y OFF

Actualmente, dado que el dirigible carece de sensores, el control se realiza en bucle abierto, lo que resulta en una ejecución imprecisa de las trayectorias programadas. Por lo tanto, esta sección está diseñada para facilitar pruebas en trabajos futuros en bucle cerrado, permitiendo una evaluación rápida de las trayectorias programadas.

# Capítulo 5

# Conclusiones y líneas futuras

*En este capítulo se exponen las conclusiones que se han obtenido tras realizar el trabajo. Además, se presentan algunas posibles líneas futuras con el objetivo final de controlar el dirigible de manera automática.*

## 5.1. Conclusiones

Tras implementar la alternativa de control propuesta, en este trabajo se puede concluir lo siguiente:

**Se ha logrado el objetivo** principal que era **desarrollar e implementar un sistema de control remoto**. El sistema funciona correctamente y se puede controlar el dirigible de manera remota desde un ordenador o desde un dispositivo móvil a través de Wi-Fi.

Figura 5.1: Dirigible controlado con la alternativa de control propuesta

En la figura 5.2 se muestra el esquema del sistema utilizado.

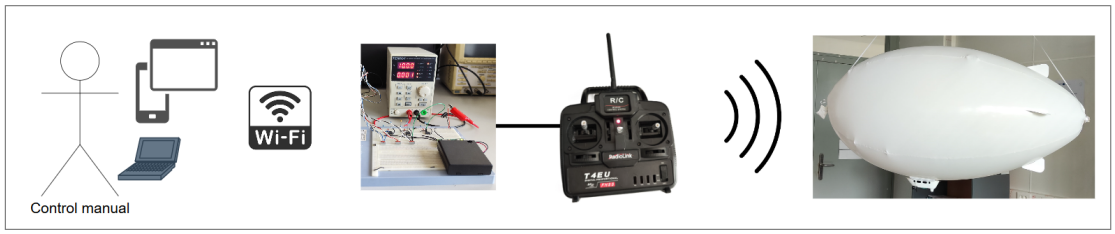



Figura 5.2: Sistema implementado en este trabajo

Además, utilizando los conmutadores del sistema electrónico también es posible tener el esquema original manejando el mando físico, mostrado en la figura 5.3.

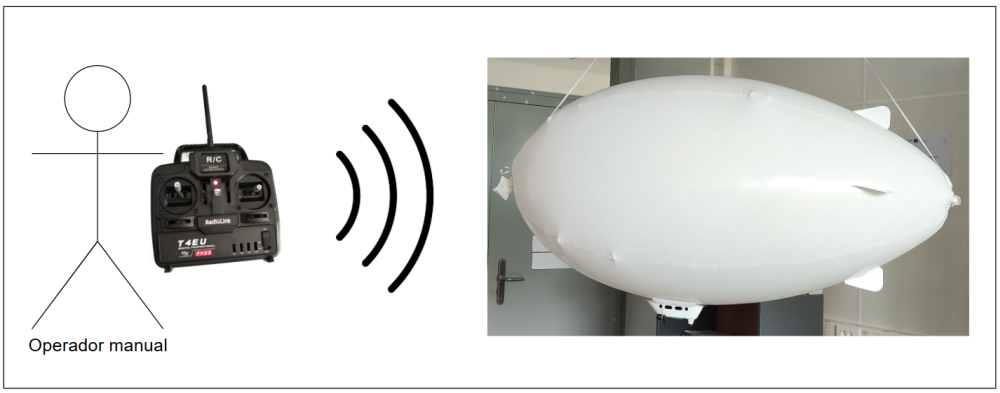



Figura 5.3: Sistema original con operador manual

El sistema de control propuesto ha sido finalmente verificado experimentalmente con el dirigible. Esta verificación también ha servido para observar problemas inherentes a este dirigible que habrá que tener en cuenta en trabajos futuros. Algunos de estos problemas son los siguientes:

- El dirigible es muy sensible a perturbaciones lo que dificulta controlar su vuelo.
- El dirigible tarda mucho tiempo en realizar los movimientos deseados. Esto se debe a que tiene una alta inercia en comparación con la potencia de sus motores, que es muy baja. Aunque el dirigible tiene poca masa y, por lo tanto, poca inercia, los motores tardan mucho en compensarla para ejecutar el movimiento deseado.

A continuación se van a proponer posibles líneas futuras para continuar con este trabajo.

## 5.2. Líneas futuras

Como se ha explicado en las conclusiones, el sistema actual presenta deficiencias en términos de eficiencia y control. La respuesta lenta del dirigible, debida a su alta inercia que limita su capacidad de reacción, afecta negativamente a la maniobrabilidad y precisión del dirigible.

Para abordar estas deficiencias y mejorar el rendimiento del sistema, se propone la siguiente línea de trabajo futura: la **implementación de un controlador que permita un control automático preciso del dirigible** y solucione los problemas actuales de respuesta y maniobrabilidad.

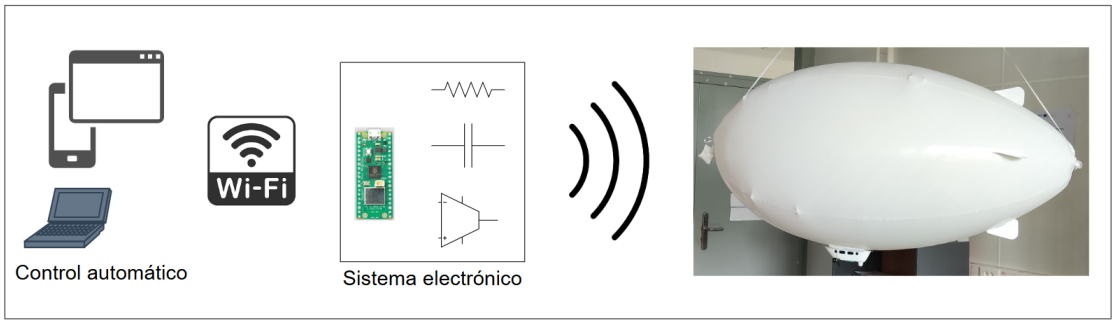



Figura 5.4: Sistema automático a largo plazo

Para implementar un control automático, es fundamental contar con sensores adecuados. Una opción efectiva es incorporar una cámara que ofrezca una vista en tiempo real, facilitando así el control del dirigible. Además, se podría considerar la inclusión de sensores de posición, distancia y velocidad para mejorar aún más la precisión del sistema.

Sin embargo, es importante tener en cuenta que la góndola del dirigible es un componente delicado, y añadir sensores incrementará su peso. Por lo tanto, es crucial seleccionar sensores que sean ligeros y eficientes para evitar sobrecargar la góndola. Otra opción podría ser poner los sensores fuera, evitando así esta limitación.

Además de implementar un controlador, otras líneas de desarrollo pueden ser:

- Diseñar y fabricar una PCB para sustituir el circuito actual, permitiendo su integración dentro del propio mando y mejorando la robustez y eficiencia del sistema.

- Realizar un estudio exhaustivo sobre el tipo de señal emitida por el mando y cómo es recibida por el receptor del dirigible. Explorar alternativas que puedan prescindir del uso del mando físico, buscando métodos de control más eficientes.

# Capítulo 6

# Bibliografía


[1] Iker Morán. Vuelve el zepelín: así es la nueva generación de dirigibles que pretenden reinventar el transporte aéreo. https://www.expansion.com/fueradeserie/motor/2023/08/08/64b917ff468aeb10678b4653.html, 2023. [Web; accedido el 12-02-2024].

[2] Enrique Pérez. Los zepelines están definitivamente de vuelta. Y el más grande de ellos volará por primera vez en España. https://www.xataka.com/movilidad/aeronave-grande-mundo-se-estrenara-espana-air-nostrum-primera-aerolinea-arriesgarse-gigantesco-zepelin-1, 2023. [Web; accedido el 12-02-2024].

[3] Gonzalo García. ¿Es un avión? ¿Una nave espacial? No, es un vehículo eléctrico y solar con múltiples aplicaciones. https://www.hibridosyelectricos.com/aviones/es-avion-una-nave-espacial-no-es-vehiculo-electrico-solar-con-multiples-aplicaciones_75687_102.html, 2024. [Web; accedido el 26-08-2024].

[4] Jorge Heras Pastor. La empresa de zepelines que operará desde Teruel completa con éxito un vuelo diurno en la estratosfera. https://www.elperiodicodearagon.com/aragon/2024/08/20/empresa-zepelines-operara-teruel-completa-107183665.html, 2024. [Web; accedido el 26-08-2024].

[5] Luis Rajadel. El aeropuerto de Teruel tendrá el hangar más alto del país y el único especializado en dirigibles. https://www.heraldo.es/noticias/aragon/teruel/2024/02/09/el-aeropuerto-de-teruel-tendra-el-hangar-mas-alto-del-pais-y-el-unico-especializado-en-dirigibles-1710011.html, 2024. [Web; accedido el 26-08-2024].

[6] Minizeppelín de Radio Control 1.4 m. https://www.publi-zeppelines.com/globos-publicitarios-de-helio/dirigibles-por-control-remoto/mini-zeppelin-de-radio-control-1.4m_246_1_ap.html. [Web; accedido el 09-01-2024].

[7] Publi Zeppelines. *1.4m Mini Zeppelin RC - Manual*. Asturias, España, 2024. Manual de usuario, incluido con el dirigible.

[8] RadioLink T4EU-6 2.4GHz 6channels remote control system. https://www.radiolink.com.cn/doce/product-detail-117.html. [Web; accedido el 20-01-2024].

[9] 2.4GHz 7CH Digital Reciver. https://www.radiolink.com.cn/doce/product-detail-97.html. [Web; accedido el 20-01-2024].

[10] Wireless Trainer Cable. https://www.radiolink.com.cn/doce/product-detail-177.html. [Web; accedido el 05-02-2024].

[11] Raspberry Pi Pico WH. https://www.raspberrypi.com/documentation/microcontrollers/pico-series.html#raspberry-pi-pico-w-and-pico-wh. [Web; accedido el 25-02-2024].

[12] Arduino Nano RP2040 Connect. https://store.arduino.cc/en-es/products/arduino-nano-rp2040-connect?srsltid=AfmBOoq5Q7YZ9sogqaSNsiyXBZvqPD8g5rEavswykDKnRcNGGHrKhf0D. [Web; accedido el 25-02-2024].

[13] CSS only D-pad and O-pad nav. https://codepen.io/tswone/pen/GLzZLd. [Web; accedido el 03-05-2024].

# Lista de Figuras

# Lista de Tablas

# Anexos

# Anexos A

# Manual de usuario

*En este anexo se presentan los pasos necesarios para utilizar el nuevo sistema de control implementado. Este documento sirve como guía para comprender su funcionamiento y ayudar a los compañeros que continúen con el proyecto y futuras mejoras del dirigible.*

## A.1. Preparación del sistema

Para que se pueda controlar el dirigible, se necesita lo siguiente:

- La góndola junto con el cuerpo hinchado. Para inflarlo se deben seguir los pasos explicados en la sección A.2.1 de este anexo.
- El mando junto con el circuito que se ha implementado. Además, se debe sincronizar tal y como se explica en la sección A.3.1 de este anexo.
- La Raspberry Pi Pico WH correctamente configurada. En ella, deben estar los siguientes archivos:
  - **lib** (Carpeta con archivos de configuración)
  - **microdot** (Carpeta con 5 archivos .py)
  - **static** (Carpeta con los archivos de la página web: CSS y JavaScript)
  - **templates** (Carpeta con el archivo de la página web: HTML)
  - **boot.py**
  - **dirigible.py**
  - **main.py**
  - **raspberryFunciones.py**
  - **wifiConfig.py**
  - **wifiRedes.json**

## A.2. Montaje y configuración física del sistema

### A.2.1. Montaje del dirigible

Para realizar el montaje del dirigible se deben seguir los siguientes pasos:

- **PASO 1:** Llenar el cuerpo del dirigible con helio a través de la válvula de inflado hasta que el cuerpo permanece firme y sin arrugas.
  *Se debe cerrar correctamente la válvula tras inflarlo (si entra aire durante esta fase, el dirigible no tendrá suficiente capacidad ascensional para volar) y atarlo con una cuerda a un lugar seguro para prevenir que se escape.*

- **PASO 2:** Unir las 4 aletas al cuerpo, retirar la parte adhesiva y pegar las aletas a unos 30 cm aproximadamente a la parte trasera del dirigible. Sujetar la góndola a los velcros dispuestos en la parte inferior del dirigible. Las ventanillas de la góndola deben estar apuntando hacia la parte delantera del dirigible, es decir, en dirección contraria a las aletas.

- **PASO 3:** Configurar el sistema de control. Para ello, se deben colocar las pilas tanto en el mando (emisor) como en la góndola (receptor).
  *En la sección A.3.1 de este anexo se explicará cómo sincronizar emisor y receptor.*

- **PASO 4:** Equilibrar el dirigible hasta que permanece neutral, es decir, que no suba ni baje. Para ello, se deben colocar unas bolsas tanto en la parte delantera como trasera del dirigible y añadir lastre en ellas. Tras equilibrarlo, se desata el dirigible y se reajusta el lastre.
  *Este paso es importante ya que cuanto más equilibrado esté el dirigible, más fácil será el posterior vuelo.*

Figura A.1: Dirigible inflado

### A.2.2. Montaje del sistema electrónico

El circuito completo está compuesto por tres circuitos individuales, como el de la figura A.2, para el control de cada motor.

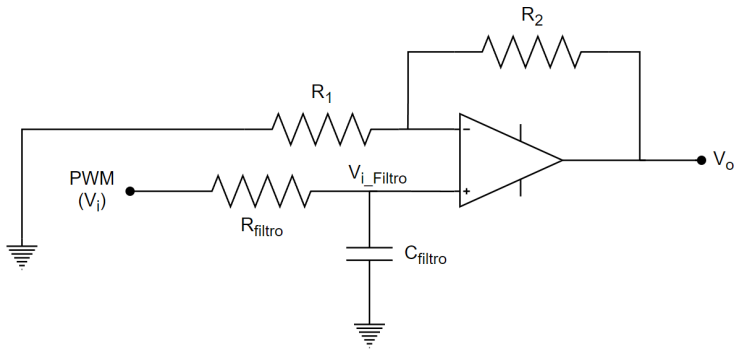



Figura A.2: Dirigible inflado

Cada uno de estos circuitos se diferencia por el color del cable de su salida que va a cada uno de los interruptores. La entrada de cada uno de estos circuitos se genera de un pin de la Raspberry Pi. Los que están configurados son los siguientes:

- El del motor 1, el azul, va al pin GP15.
- El del motor 2, el blanco, va al pin GP14.
- El del motor 3, el verde, va al pin GP13.

El sistema electrónico montado se muestra en la figura A.3

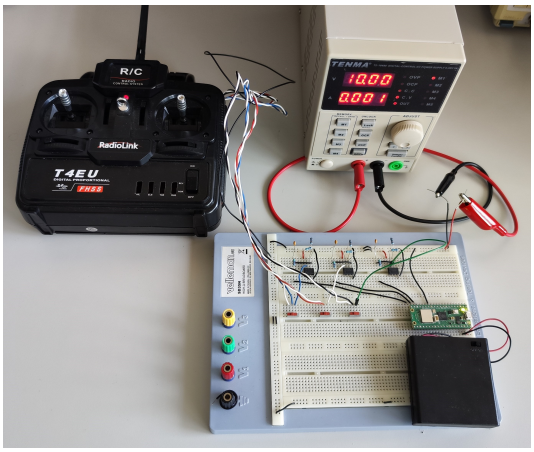



Figura A.3: Modificación del mando con los cables fuera

Como se observa, los cables rojo y negro de la parte superior derecha son los que deben alimentar los amplificadores operacionales utilizados en cada circuito. En el siguiente apartado, A.2.3, se explica como alimentar tanto los amplificadores como la Raspberry Pi.

### A.2.3. Alimentación del circuito electrónico

Para que el circuito funcione, se deben alimentar tanto la Raspberry Pi (si no se conecta al ordenador) como los amplificadores operacionales que tiene el circuito. En la figura A.4 se muestra en la parte superior derecha y en el centro a la derecha, donde se debe alimentar el circuito electrónico.

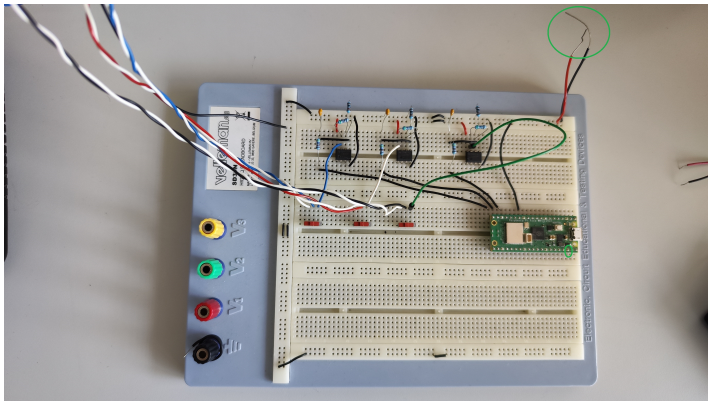


Figura A.4: Circuito indicando donde se debe alimentar

La alimentación de la Raspberry es sencilla, simplemente se tiene que conectar el portapilas. El cable rojo debe ir al pin 39, VSYS, y el cable negro al pin 38, GND, los cuales están señalados en la figura A.5. A pesar de que la Raspberry Pi tiene más pines GND, se debe conectar obligatoriamente a ese pin ya que es la tierra de la fuente externa.

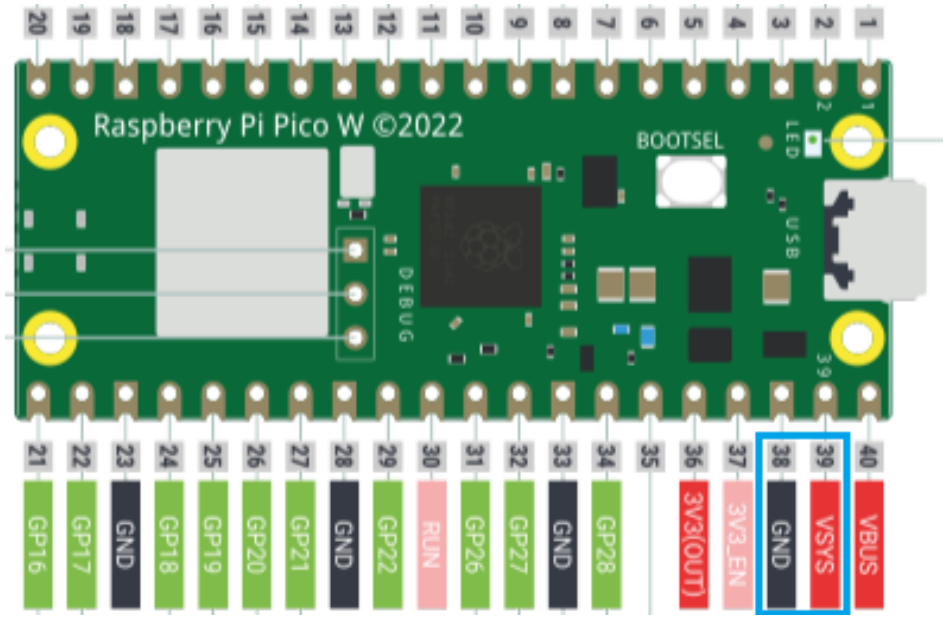



Figura A.5: Pin de alimentación y de tierra de la Raspberry Pi Pico

Se deben introducir las pilas en el portapilas de la figura A.6.

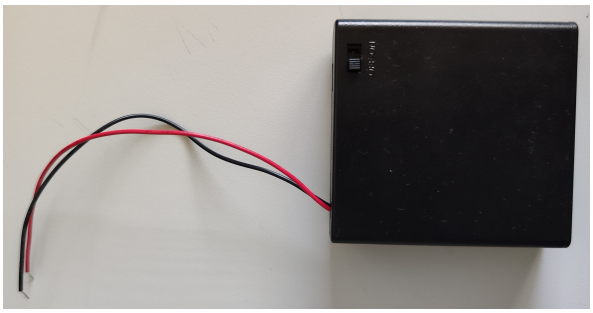


Figura A.6: Portapilas para alimentar la Raspberry Pi Pico WH

La alimentación de los amplificadores es más compleja y se debe utilizar una fuente de tensión externa ya que se necesitan aproximadamente 7 V para que trabajen correctamente. La fuente se debe conectar mediante cables de cocodrilo a los cables rojo y negro y estará configurada a 10 V, igual que en la figura A.7.

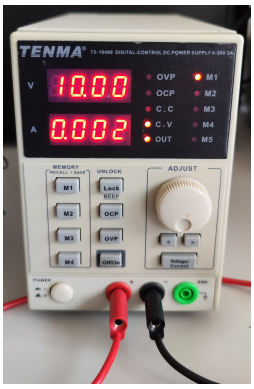



Figura A.7: Fuente de alimentación

Finalmente, el circuito completamente alimentado se muestra en la figura A.8.

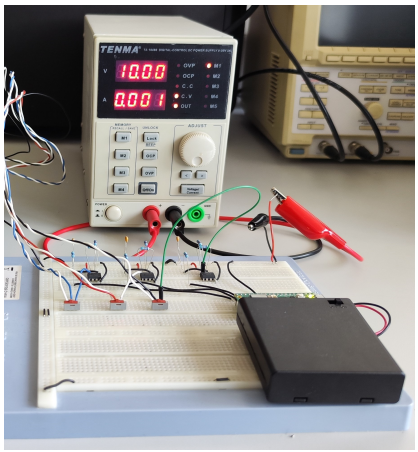



Figura A.8: Circuito alimentado

## A.3. Configuración y control del sistema

### A.3.1. Sincronización entre emisor y receptor

Para poder comunicar el mando (emisor) con la góndola (receptor), se debe encender el emisor y después el receptor, en ese orden. Tras encender el mando, se presiona el botón negro que hay cerca de la antena mostrado en la figura A.9, durante 2 segundos y tras soltarlo comienza a parpadear la luz del receptor. Después de parpadear aproximadamente 8 veces, se conectan.

Figura A.9: Botón negro del receptor

### A.3.2. Elección del control (original o alternativo)

Para que esté en funcionamiento la alternativa que se ha implementado, los 3 interruptores tienen que estar en la posición izquierda, tal y como se muestra en la figura A.10.

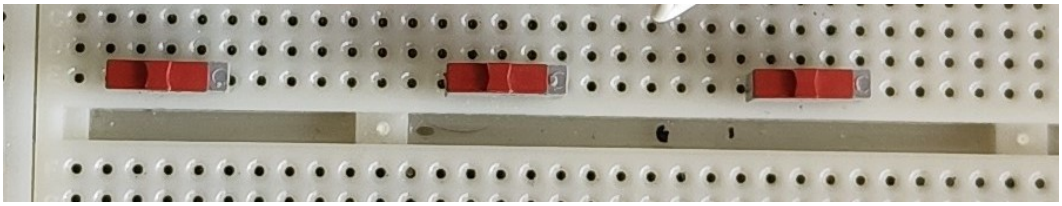


Figura A.10: Posición de los interruptores (Izquierda)

Si por el contrario, se quiere seguir utilizando el mando físico original, deben estar en la posición derecha, mostrado en la figura A.11.

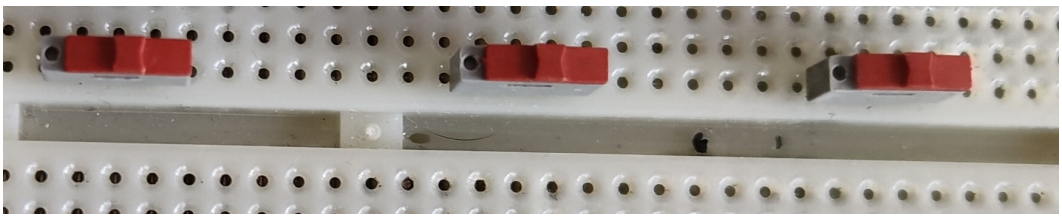


Figura A.11: Posición de los interruptores (Derecha)

Las posiciones izquierda y derecha son respecto a la BreadBoard en su orientación original.

### A.3.3. Configuración de la red Wi-Fi

Para realizar la conexión, se debe abrir el archivo *wifiRedes.json* con cualquier editor de texto y se debe rellenar con los datos de la red Wi-Fi a la que se quiera conectar, nombre (SSID) y contraseña (password). En este archivo, se pueden almacenar múltiples redes y seleccionar, posteriormente, a la que se quiera conectar.

```
{
    "wifi_1": {
        "ssid": "Ingrese el nombre de la red Wi-Fi (1)",
        "password": "Ingrese la contrasena de la red Wi-Fi (1)"
    },

    "wifi_2": {
        "ssid": "Ingrese el nombre de la red Wi-Fi (2)",
        "password": "Ingrese la contrasena de la red Wi-Fi (2)"
    }
}
```

Listing A.1: Archivo de datos de redes Wi-Fi

A continuación, con un editor de texto o con cualquier entorno de programación, se abre el archivo *boot.py*. En la línea donde pone: red_seleccionada = "wifi_1", se selecciona la red cambiando el número de Wi-Fi.

```
from wifiConfig import obtener_credenciales, conectar_wifi

red_seleccionada = "wifi_1" # Se selecciona la red Wi-Fi.
# "wifi_1" --> WIFI CASA
# "wifi_2" --> DATOS MOVILES

# Se obtienen las credenciales de la red Wi-Fi seleccionada.
[ssid, password] = obtener_credenciales(red_seleccionada)

# Se realiza la conexion a la red Wi-Fi con esas credenciales.
conectar_wifi(ssid, password)
```

Listing A.2: Archivo de configuración boot.py

En este archivo se utilizan la funciones obtener_credenciales y conectar_wifi, cuyo código se mostrará en el anexo C.

Una vez realizada la configuración de la red Wi-Fi, el siguiente paso es realizar la conexión para poder controlar el dirigible.

### A.3.4. Conexión a la página web

Para que la conexión sea efectiva, la red seleccionada en la Raspberry Pi debe coincidir con la red a la que está conectado el dispositivo (móvil, ordenador, etc.) desde el cual se desea controlar el dirigible.

Para realizar la primera conexión, se debe ejecutar el archivo *main.py* para conocer la dirección IP (y el puerto que será el 5000) a la que se debe acceder para ver la página web. Para ello, se deberá disponer de un IDE, como por ejemplo, Thonny, que es el que se ha utilizado para este trabajo. Al ejecutar el *main.py*, aparecerá el siguiente mensaje:

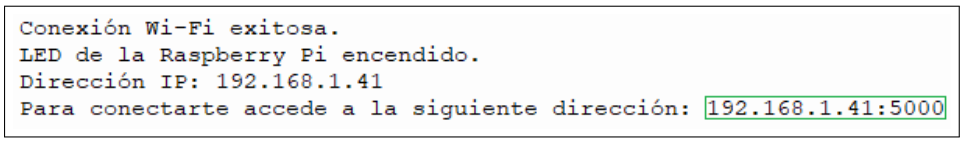



Figura A.12: Mensaje de conexión

La dirección marcada con un recuadro en verde es a la que se debe conectar. En cada caso, aparecerá una dirección diferente si es la primera conexión. En las posteriores conexiones, no hará falta realizar este paso y se podrá acceder directamente a esa dirección sin tener que ejecutar el *main.py* en el ordenador.

## A.4. Explicación del control del dirigible

Una vez realizada la configuración de la red Wi-Fi, la conexión inicial y la alimentación del circuito, ya es posible controlar el dirigible. Para ello, se debe acceder a la dirección que se ha configurado en la sección A.3.4. Desde allí, ya se podrá acceder a todas las secciones de las que dispone la página web y realizar diferentes tipos de control del dirigible.

- **INICIO:** En esta sección, se han realizado las pruebas iniciales. Se pueden generar diferentes señales y seleccionar el pin de la Raspberry Pi sobre el que ejecutarlas.

- **MANDO VIRTUAL:** En esta sección, se puede controlar el dirigible tal y como se controlaría con el mando original. La réplica del mando es totalmente funcional. Además, cuenta con una tabla con la información sobre el voltaje de la señal que se manda y dispone de dos botones para subir y dos botones para bajar con los que se aumenta o disminuye el voltaje de 0.10 V en 0.10 V o de 0.01 V en 0.01 V. También dispone de dos vistas diferentes para visualizar el estado de cada movimiento.

- **CONTROL ALTERNATIVO:** En esta sección, se puede controlar el dirigible de dos formas alternativas. Por un lado, se tiene el control mediante el pad, el cual dispone de un D-Pad (o el O-Pad) para controlar los movimientos direccionales (avanzar, retroceder, izquierda y derecha). Para que funcione se tiene que presionar el pad y si se suelta, se detiene. Por otro lado, están las dos barras con las que se controla la velocidad con la que gira la hélice individual para subir o para bajar, según su sentido de giro.

  La segunda alternativa es el control por voz y mediante unos comandos específicos que se muestran en la página web, es posible controlar el dirigible. Al realizar la conexión a la página mediante el protocolo HTTP, para poder utilizar el micrófono en un navegador, por ejemplo, Google Chrome, se debe activar en la parte de Experiments: “Insecure origins treated as secure” e introducir la dirección web.

- **CONTROL AUTOMÁTICO:** Esta sección, como se ha explicado anteriormente, está en una fase inicial por ser un control en bucle abierto debido a la falta de sensores. En ella, se pueden realizar pequeñas trayectorias pero de manera bastante imprecisa.

# Anexos B

# Código fuente

*En este anexo se presentan las partes más importantes del código que se ha utilizado tanto en la implementación de las funciones de la Raspberry Pi como del desarrollo de la página web.*

## B.1. MicroPython

### B.1.1. Funciones para la conexión

Para realizar la conexión Wi-Fi se han utilizado las funciones implementadas en el archivo ***wifiConfig.py***.

```
# wifiConfig.py
'''
Este modulo se utiliza para realizar la
configuracion de la WiFi. Tanto el nombre de las
redes como sus contrasenias estan en el archivo
wifi_redes.json
'''

import json # Este modulo es una implementacion de JSON
            # (JavaScript Object Notation) optimizada
            # para MicroPython.
import network # Este modulo proporciona funcionalidades
              # para trabajar con conexiones de red,
              # incluida la conexion Wi-Fi.
import time
import sys # Lo usamos para provocar la salida en caso
          # de no conectarse a la Wi-Fi
# Para poder usar las funciones de la Raspberry Pi Pico.
from raspberryFunciones import RaspberryPiPicoWH
raspberry = RaspberryPiPicoWH()

# Logica para conectar a la red Wi-Fi
wlan = network.WLAN(network.STA_IF)

# Procedimiento de conexion a la red Wi-Fi
def conectar_wifi(ssid, password):
    wlan.active(True)
```

```
28      wlan.connect(ssid, password)
29      # Variables de tiempo
30      tiempo_inicio = time.time()
31      tiempo_maximo = 30 # CONDICION MAXIMA DE ESPERA PARA
32                         # LA CONEXION
33      try:
34          while (not wlan.isconnected()) and \
35                ((time.time() - tiempo_inicio) < tiempo_maximo):
36              print(f"Conectando a la red Wi-Fi... "
37                    f"({int(time.time() - tiempo_inicio)}/30 "
38                    "segundos)")
39              raspberry.encenderRaspLED(False)
40              time.sleep(0.5)
41              raspberry.apagarRaspLED(False)
42              time.sleep(0.5)
43
44          if wlan.isconnected():
45              ip = wlan.ifconfig()[0]
46              print("Conexion Wi-Fi exitosa.")
47              raspberry.encenderRaspLED(True)
48              print(f"Direccion IP: {ip}")
49              print(f"Para conectarte accede a la siguiente "
50                    f"direccion: {ip}:5000")
51          else:
52              print("Tiempo de espera maximo alcanzado. No "
53                    "se pudo conectar a la red Wi-Fi.")
54              sys.exit(1)
55
56      except KeyboardInterrupt:
57          print("Interrupcion de teclado. (CONECT. WIFI)")
58          sys.exit(1)  # Salir del programa con un codigo de error
59
60  # Obtener los datos de la red de un archivo JSON.
61  def obtener_credenciales(nombre_red):
62      # Abrir el archivo JSON y cargar la configuracion
63      with open("wifiRedes.json", 'r') as archivo:
64          wifi_redes = json.load(archivo)
65
66      # Verificar si la red especificada esta en la configuracion.
67      if nombre_red in wifi_redes:
68          # Obtener las credenciales de la red especificada.
69          ssid = wifi_redes[nombre_red].get('ssid')
70          password = wifi_redes[nombre_red].get('password')
71          return(ssid, password) # Devuelve las credenciales.
72      else:
73          # Imprimir un mensaje de error si la red no esta en
74          # la configuracion
75          print(f'Error: La red "{nombre_red}" no esta en la '
76                'configuracion.')
77          sys.exit(1)
78
79  # Procedimiento de desconexion a la red Wi-Fi
80  def desconectar_wifi():
81      if wlan.isconnected():
82          wlan.disconnect()
83          while wlan.isconnected():
84              time.sleep(1) # Agregar un pequeno retardo
85          raspberry.apagarRaspLED(True)
```

```
86          print("Desconexion Wi-Fi exitosa")
87      else:
88          print("No estas conectado a ninguna red Wi-Fi.")
```

Listing B.1: Código *wifiConfig.py*

### B.1.2. Funciones auxiliares

Para controlar las características integradas de la Raspberry Pi se han implementado las funciones del archivo ***raspberryFunciones.py***.

```
1  # raspberryFunciones.py
2  from machine import Pin, ADC
3
4  # FUNCIONES PARA LA RASPBERRY PI PICO
5  class RaspberryPiPicoWH:
6      # Funcion para encender el LED de la Raspberry
7      # y mostrar un mensaje.
8      def encenderRaspLED(self, mensaje=True):
9          Pin("LED", Pin.OUT).on()
10         if mensaje:
11             print('LED de la Raspberry Pi encendido.')
12
13     # Funcion para apagar el LED de la Raspberry
14     # y mostrar un mensaje.
15     def apagarRaspLED(self, mensaje=True):
16         Pin("LED", Pin.OUT).off()
17         if mensaje:
18             print('LED de la Raspberry Pi apagado.')
19
20     # Funcion que devuelve la temperatura utilizando
21     # el sensor de la Raspberry.
22     def devuelveTemperatura(self):
23         factor_conversion = 3.3 / 65365
24         valor_bruto = ADC(4).read_u16() * factor_conversion
25         temperatura = 27 - (valor_bruto - 0.706) / 0.001721
26         return temperatura
```

Listing B.2: Código *raspberryFunciones.py*

### B.1.3. Funciones para el control del dirigible

Para controlar el dirigible se han implementado las funciones del archivo ***dirigible.py***. Este archivo también contiene las funciones para generar las señales de las pruebas iniciales, pero no se van a detallar ya que no son relevantes para el objetivo principal.

```
# dirigible.py
# Codigo del Dirigible
from machine import Pin, PWM
from time import sleep, sleep_ms, sleep_us
import utime

class Dirigible_objeto:
    ## __init__: Metodo especial que se llama
    ## automaticamente cuando se crea una nueva
    ## instancia de una clase. Su funcion principal
    ## es inicializar los atributos de la instancia.
    def __init__(self):
        # 1 - Iniciar los motores.
        self._iniciarMotores()
        # 2 - Valores para parar los 3 motores:
        self.valorPararMotorSubirBajar = 2.10 # MOTOR 1
        self.valorPararMotorIzquierdaDerecha = 2.75 # MOTOR 2
        self.valorPararMotorAvanzarRetroceder = 1.50 # MOTOR 3

    ## deinit: Metodo especifico de MicroPython
    ## que se utiliza para desinicializar y liberar
    ## recursos asociados con un objeto.
    def deinit(self):
        print("Desinicializando los pines PWM...")
        utime.sleep_us(1)
        # Desinicializar los pines PWM asociados con
        # los motores.
        self.motorSubirBajar.deinit()
        self.motorIzquierdaDerecha.deinit()
        self.motorAvanzarRetroceder.deinit()

    ## MOTORES
    def _iniciarMotores(self):
        # Asignacion de objetos PWM a los pines
        # especificos para controlar los motores:
        self.motorSubirBajar = PWM(Pin(15, Pin.OUT)) # AZUL (MOTOR 1)
        self.motorIzquierdaDerecha = PWM(Pin(14, Pin.OUT)) #BLANCO(M2)
        self.motorAvanzarRetroceder = PWM(Pin(13, Pin.OUT)) #VERDE(M3)
        # Asignacion de la frecuencia:
        frecuencia = 1000 # Configura el PWM a 1000Hz.
        self.motorSubirBajar.freq(frecuencia)
        self.motorIzquierdaDerecha.freq(frecuencia)
        self.motorAvanzarRetroceder.freq(frecuencia)

    ###### FUNCIONES AUXILIARES #####
    def ajustarVsalida(self, valor):
        # Funcion para asegurar que el valor
        # este dentro del rango valido (0-4.5V)
```

```
49          # (DUTY entre 0%-90%).
50          if valor > 4.5: return 4.5
51          elif valor < 0: return 0
52          else: return valor
53
54      def ajustarPWM(self, duty):
55          # Calcular el valor del PWM segun el DUTY.
56          return min(int(duty/100*65535), 65535)
57
58      ###### FUNCIONES AUXILIARES #####
59
60      ## Configurar salida para un motor segun un
61      ## voltaje deseado dentro de los limites
62      ## permitidos.
63      def configurarVsalida(self, motor, voltios,
64                            mensaje=False):
65          # Averiguar que motor ha llamado a la
66          # funcion.
67          if motor == self.motorSubirBajar:
68              nombre_motor = "MOTOR 1: Subir/Bajar"
69          elif motor == self.motorIzquierdaDerecha:
70              nombre_motor = "MOTOR 2: Izquierda/Derecha"
71          elif motor == self.motorAvanzarRetroceder:
72              nombre_motor = "MOTOR 3: Avanzar/Retroceder"
73
74          # Realizar los calculos correspondientes.
75          vSalidaMaxima = self.ajustarVsalida(voltios)
76          vRaspb = vSalidaMaxima * (3.3 / 5)
77          duty = (vRaspb / 3.3) * 100
78          valorPWM = self.ajustarPWM(duty)
79          motor.duty_u16(valorPWM)
80          # Mostrar mensajes informativos.
81          if (voltios < 0) or (voltios > 4.5):
82              print("La salida de", voltios, "V esta fuera"
83                    " del rango permitido y se ha ajustado"
84                    " a", vSalidaMaxima, "V.")
85          if mensaje:
86              print(f"{nombre_motor} |", "DUTY: "
87                    "{:.2f}% |".format(duty),
88                    "Valor PWM: {:.0f} |".format(valorPWM),
89                    "Voltios Raspberry: {:.2f} V".
90                    format(vRaspb), "| Salida a "
91                    "interruptor: {:.2f} V".
92                    format(self.ajustarVsalida(voltios)))
93
94      # Funciones para detener cada motor por separado
95      # y los 3 a la vez.
96      def DetenerMotorSubirBajar(self, mensaje=False):
97          self.configurarVsalida(self.motorSubirBajar,
98                                 self.valorPararMotorSubirBajar)
99          if mensaje:
100             print("El motor de subir y bajar esta"
101                   " detenido.")
102
103     def DetenerMotorIzquierdaDerecha(self, mensaje=False):
104         self.configurarVsalida(self.motorIzquierdaDerecha,
105                                self.valorPararMotorIzquierdaDerecha)
106         if mensaje:
```

```
107             print("El motor de izquierda y derecha"
108                   " esta detenido.")
109
110     def DetenerMotorAvanzarRetroceder(self, mensaje=False):
111         self.configurarVsalida(self.motorAvanzarRetroceder,
112                                self.valorPararMotorAvanzarRetroceder)
113         if mensaje:
114             print("El motor de avanzar y retroceder"
115                   " esta detenido.")
116
117     def DetenerTodo(self, mensaje=False):
118         self.configurarVsalida(self.motorSubirBajar,
119                                self.valorPararMotorSubirBajar)
120         self.configurarVsalida(self.motorIzquierdaDerecha,
121                                self.valorPararMotorIzquierdaDerecha)
122         self.configurarVsalida(self.motorAvanzarRetroceder,
123                                self.valorPararMotorAvanzarRetroceder)
124         if mensaje:
125             print("El zepelin esta detenido.")
126
127     # Funciones para configurar un motor en una
128     # direccion determinada.
129     def Bajar(self, porcentaje, mensaje=False):
130         voltaje = (1.95 - (porcentaje / 100) *
131                    (1.95 - 1.2))
132         # Entre 1.2V-1.95V (1.95-Lento | 1.2-Rapido)
133         self.configurarVsalida(self.motorSubirBajar,
134                                voltaje)
135         if mensaje:
136             print("El zepelin esta subiendo... "
137                   "(Velocidad: {:.2f}%)".format(porcentaje))
138
139     def Subir(self, porcentaje, mensaje=False):
140         voltaje = (2.25 + (porcentaje / 100) *
141                    (3.4 - 2.25))
142         # Entre 2.25V-3.4V (2.25-Lento | 3.4-Rapido)
143         self.configurarVsalida(self.motorSubirBajar,
144                                voltaje)
145         if mensaje:
146             print("El zepelin esta bajando... "
147                   "(Velocidad: {:.2f}%)".format(porcentaje))
148
149     def Izquierda(self, porcentaje, mensaje=False):
150         voltaje = (2.55 - (porcentaje / 100) *
151                    (2.55 - 0.5))
152         # Entre 0.5V-2.55V (2.55-Lento | 0.5-Rapido)
153         self.configurarVsalida(self.motorIzquierdaDerecha,
154                                voltaje)
155         if mensaje:
156             print("El zepelin se esta moviendo hacia "
157                   "la izquierda... (Velocidad: {:.2f}%)".
158                   format(porcentaje))
159
160     def Derecha(self, porcentaje, mensaje=False):
161         voltaje = (3 + (porcentaje / 100) *
162                    (4.5 - 3))
163         # Entre 3V-4.5V (3-Lento | 4.5-Rapido)
164         self.configurarVsalida(self.motorIzquierdaDerecha,
```

```
165                                 voltaje)
166         if mensaje:
167             print("El zepelin se esta moviendo hacia "
168                   "la derecha... (Velocidad: {:.2f}%)".
169                   format(porcentaje))
170 
171     def Retroceder(self, porcentaje, mensaje=False):
172         voltaje = (1.25 - (porcentaje / 100) *
173                    (1.25 - 0))
174         # Entre 0V-1.25V (1.25-Lento | 0-Rapido)
175         self.configurarVsalida(self.motorAvanzarRetroceder,
176                                voltaje)
177         if mensaje:
178             print("El zepelin esta avanzando... "
179                   "(Velocidad: {:.2f}%)".format(porcentaje))
180 
181     def Avanzar(self, porcentaje, mensaje=False):
182         voltaje = (1.75 + (porcentaje / 100) *
183                    (3.8 - 1.75))
184         # Entre 1.75V-3.8V (1.75-Lento | 3.8-Rapido)
185         self.configurarVsalida(self.motorAvanzarRetroceder,
186                                voltaje)
187         if mensaje:
188             print("El zepelin esta retrocediendo... "
189                   "(Velocidad: {:.2f}%)".format(porcentaje))
```

Listing B.3: Código *dirigible.py*

### B.1.4. Funciones para comunicarse con la página web

Para realizar la comunicación se han utilizado dos archivos. El primero es ***boot.py***, el cual configura el entorno de ejecución. En este caso, obtiene los credenciales y realiza la conexión a la red Wi-Fi.

```
# boot.py
# En el boot.py simplemente leemos el archivo
# JSON y obtenemos el nombre de la Wi-Fi y su
# contrasena.

from wifiConfig import obtener_credenciales, conectar_wifi

red_seleccionada = "wifi_2" # Seleccionamos la red
# Wi-Fi a configurar
# "wifi_1" --> WIFI CASA
# "wifi_2" --> DATOS MOVILES

# Primero obtenemos las credenciales de la red
# Wi-Fi seleccionada.
[ssid, password] = obtener_credenciales(red_seleccionada)

# Despues, nos conectamos a la red Wi-Fi
# utilizando estas credenciales.
conectar_wifi(ssid, password)
```

Listing B.4: Código *boot.py*

El segundo es ***main.py*** y es el archivo que contiene el código principal del programa. Se ejecuta después de *boot.py*. En este caso, se va a detallar únicamente una parte del código.

```
from microdot import Microdot, Response, send_file
from microdot.websocket import with_websocket
from microdot.utemplate import Template

# Para poder usar las funciones del archivo del Dirigible.
from raspberryFunciones import RaspberryPiPicoWH
from dirigible import Dirigible_objeto
from wifiConfig import desconectar_wifi
import sys
import uasyncio
from time import sleep

# Crear una instancia de la raspberry.
raspberry = RaspberryPiPicoWH()
# Crear una instancia del zepelin.
dirigible = Dirigible_objeto()

# FUNCIONES PARA INICIAR EL SERVIDOR Y CARGAR LA PAGINA WEB
app = Microdot()
Response.default_content_type = "text/html"

```

```
22 # Ruta principal
23 @app.route("/")
24 async def index(request):
25     return Template('index.html').render(title='Home')
26
27 # Servir archivos estaticos (CSS y JavaScript)
28 @app.route('/static/<path:path>')
29 async def static(request, path):
30     if '..' in path:
31         # directory traversal is not allowed
32         return 'Not found', 404
33     return send_file('static/' + path)
34
35 # Ruta para obtener la temperatura
36 @app.route("/temperatura", methods=["POST"])
37 async def obtener_temperatura(request):
38     global seccion_seleccionada
39     if (seccion_seleccionada == "inicio"):
40         temp = raspberry.devuelveTemperatura()
41         temp_redond = "{:.2f}".format(temp)
42         return str(temp_redond)
43
44 # Ruta para cerrar el servidor
45 @app.route('/shutdown', methods=["POST"])
46 def shutdown(request):
47     print("Cerrando el servidor...")
48     request.app.shutdown()
49     return 'The server is shutting down...'
50
51
52 # VARIABLES GLOBALES:
53 seccion_seleccionada = "inicio"
54 funcionamiento = "NORMAL"
55
56 # TRAYECTORIAS PROGRAMADAS
57 flag_trayecProgam = False # Variable global para
58 # controlar el bucle de las trayectorias programadas
59
60 async def trayectorias_programadas(velocidadAutomatico):
61     ...
62     ...
63
64
65 # FUNCIONAMIENTO:
66 @app.route("/ws")
67 @with_websocket
68 async def ejecutarComandosDirigible(request, ws):
69     global seccion_seleccionada, funcionamiento, flag_trayecProgam
70
71     ...
72
73     while True:
74         mensajeWebSocket = await ws.receive()
75         # COMPROBAR EL FUNCIONAMIENTO
76         if "funcionamiento:" in mensajeWebSocket:
77             funcionamiento = mensajeWebSocket.split(":")[1].strip()
78
79         if (funcionamiento == "EMERGENCIA"):
```

```
80              # Si hay una emergencia, apagamos el zepelin.
81              dirigible.DetenerTodo()
82
83          else: # (funcionamiento = "NORMAL")
84              if "seccion:" in mensajeWebSocket:
85                  seccion_seleccionada = mensajeWebSocket.split(":")[1]
86                                                              .strip()
87
88              # SECCION 0 - INICIO
89              if (seccion_seleccionada == "inicio"):
90                  ...
91
92              # SECCION 1 - MANDO VIRTUAL
93              elif (seccion_seleccionada == "mandoVirtual"):
94                  ...
95
96              # SECCION 2 - CONTROL ALTERNATIVO
97              elif (seccion_seleccionada == "controlAlternativo"):
98                  ...
99
100             # SECCION 3 - CONTROL AUTOMATICO
101             elif (seccion_seleccionada == "controlAutomatico"):
102                 ...
103
104
105 # INICIO DEL PROGRAMA:
106 if __name__ == "__main__":
107     exit_flag = False  # Bandera para indicar si se
108                        # debe salir del programa.
109     try:
110         app.run()
111     except KeyboardInterrupt:
112         exit_flag = True
113         print("INTERRUPCION DE TECLADO.")
114         desconectar_wifi() # Desconectar la conexion Wi-Fi.
115         raspberry.apagarRaspLED(True) # Apagar el LED
116                                       # de la Raspberry Pi.
117         sys.exit(1)
118     except Exception as e:
119         sys.exit(1)
120         print("Error: ", e)
121     finally:
122         if not exit_flag:
123             print("Has salido del servidor.")
124             dirigible.deinit()
125             desconectar_wifi() # Desconectar la conexion Wi-Fi.
126             raspberry.apagarRaspLED(True) # Apagar el LED
127                                           # de la Raspberry Pi.
128             print("Si quieres volver a conectarte, REINICIA
129                   EL DISPOSITIVO.")
```

Listing B.5: Código *main.py*

El código de las partes donde están los puntos suspensivos (...) se ha eliminado por resumir el contenido del archivo *main.py*.

## B.2. Página web (HTML, CSS y JavaScript)

En la figura B.1 se muestra la estructura de los directorios de la página web.

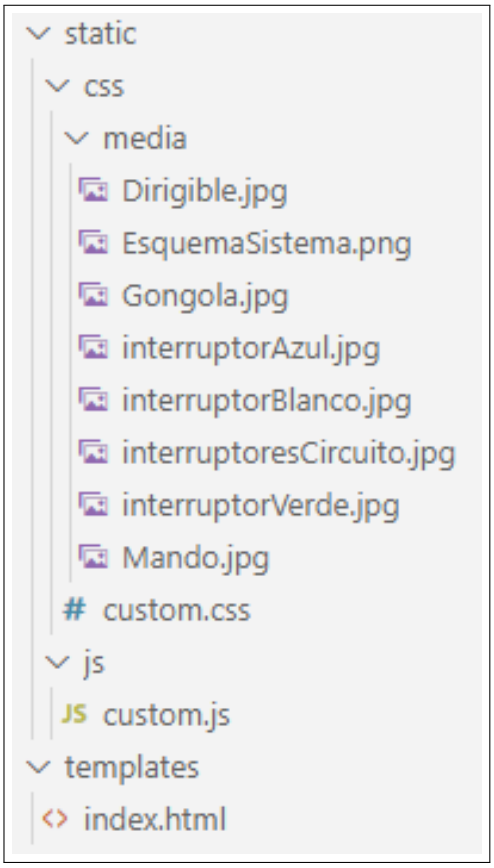



Figura B.1: Estructura de los directorios de la página web

El archivo HTML (index.html) está en la carpeta templates junto con el archivo index_html.py que crea Microdot. El archivo CSS (custom.css) está en la carpeta static, en la subcarpeta css. Dentro de esta carpeta, en la subcarpeta media, se encuentra las fotos que aparecen en algunas de las secciones de la página web. El archivo de JavaScript (custom.js) se encuentra en la carpeta static, en la subcarpeta js.

Al ser un código muy extenso, no se va a detallar el código HTML ni CSS, simplemente se van a mostrar las funciones de JavaScript involucradas en la conexión y en la comunicación con la Raspberry Pi.

Hay más funciones que podrían ser relevantes detallar, sin embargo, para no incrementar aún más el tamaño de los anexos de la memoria, no se incluirán en este documento.

```
1  document.addEventListener("DOMContentLoaded", onLoad);
2
3  // FUNCION PARA LO QUE SE VA A HACER AL
4  // CARGAR LA PAGINA
5  function onLoad() {
6    initializeSocket();
7  }
8
9  /***** VARIABLES GLOBALES *****/  /*IMPORTANTE*/
10 let servidorActivado = false;
11 // Reconectar servidor:
12 const btnConectarServidor = document.getElementById(
13   'btnConectarServidor'
14 );
15 btnConectarServidor.addEventListener("click", function () {
16   if (!servidorActivado) {
17     initializeSocket();
18   }
19 });
20
21 /************************************ CONEXION
22 WEBSOCKET ************************************/
23 // Inicializacion de la conexion WebSocket
24 let websocket;
25 function initializeSocket() {
26   const targetUrl = `ws://${location.host}/ws`;
27   websocket = new WebSocket(targetUrl);
28   websocket.onopen = onOpen;
29   websocket.onclose = onClose;
30   // websocket.onmessage = onMessage;
31   websocket.onerror = onError;
32 }
33 // Eventos WebSocket
34 function onOpen() {
35   console.log("Iniciando conexion al servidor WebSocket...");
36   btnConectarServidor.className = ''; // Quitar todas las clases
37   btnConectarServidor.textContent = 'Servidor CONECTADO';
38   btnConectarServidor.classList.add('btnServActivado');
39   console.log('Servidor conectado');
40   servidorActivado = true;
41 }
42 function onClose(event) {
43   console.log("Cerrando conexion con el servidor...");
44   btnConectarServidor.className = ''; // Quitar todas las clases
45   btnConectarServidor.textContent = 'Servidor DESCONECTADO';
46   btnConectarServidor.classList.add('btnServDesactivado');
47   servidorActivado = false;
48   // setTimeout(initializeSocket, 2000);
```

```
49 }
50 function onMessage(event) {
51   console.log("Mensaje de WebSocket recibido:", event);
52   console.log("Tipo de evento:", event.type); // Tipo de evento
53   console.log("Origen:", event.origin); // Origen del mensaje
54   console.log("ID del ultimo evento:", event.lastEventId); // ID
55 }
56 function onError(event) {
57   console.log("Se encontro un error al comunicarse con el servidor
58               WebSocket..", event);
59 }
60
61 // Envio de mensajes
62 function enviarMensajeWeb(mensaje) {
63   if (servidorActivado) {
64     websocket.send(mensaje);
65   }
66 }
```

Listing B.6: Código de JavaScript (*custom.js*) de la conexión

Esta última función, "enviarMensajeWeb(mensaje)", es la encargada de enviar la información desde la página web hasta la Raspberry Pi que mediante MicroPython la procesa para realizar el control correspondiente del dirigible.