Universitat de les Illes Balears

escola politècnica superior

Treball Final de Grau

GRAU D'ENGINYERIA ELECTRÒNICA INDUSTRIAL I AUTOMÀTICA

Integración de sensores de detección de obstáculos por ultrasonido sobre un vehículo aéreo de bajo coste

MIGUEL ÁNGEL MOLL LLABRÉS

Tutors

Emilio García Fidalgo Alberto Ortiz Rodríguez

Escola Politècnica Superior
Universitat de les Illes Balears
Palma, 25 de julio de 2017

ÍNDICE GENERAL

Índice general i

Índice de figuras iii

Índice de tablas vii

Resumen ix

1 Introducción 1
- 1.1 Robótica aérea en la actualidad . . . 1
  - 1.1.1 Aplicaciones militares . . . 1
  - 1.1.2 Aplicaciones agrícolas . . . 1
  - 1.1.3 Aplicaciones cinematográficas . . . 2
- 1.2 Drones comerciales . . . 2
- 1.3 Sensores de ultrasonidos . . . 3
- 1.4 Objetivos del proyecto . . . 3
- 1.5 Organización del documento . . . 4

2 Módulo basado en ultrasonidos para la detección de obstáculos 5
- 2.1 Vista general . . . 5
- 2.2 Componentes hardware . . . 5
  - 2.2.1 Arduino . . . 6
  - 2.2.2 Sensores de ultrasonidos . . . 6
  - 2.2.3 Transmisión por radiofrecuencia . . . 8
  - 2.2.4 Ar.Drone . . . 10
- 2.3 Diseño del circuito impreso . . . 10
  - 2.3.1 Conexión de los sensores de ultrasonidos . . . 11
  - 2.3.2 Conexión del transmisor de radiofrecuencia . . . 13
  - 2.3.3 Montaje sobre el Ar.drone . . . 14

3 Estrategia de evitación de obstáculos 17
- 3.1 Vista general . . . 17
- 3.2 *Robot Operating System* . . . 17
- 3.3 Obtención de distancias sobre Arduino . . . 18
- 3.4 Software de la estación base . . . 19
  - 3.4.1 *ardrone_autonomy* . . . 19
  - 3.4.2 *srv_teleop* . . . 20

3.4.3 *ardrone_usound_driver* 20
3.4.4 *potential_fields* 21

4 Resultados experimentales **27**
4.1 Experimentos preliminares 27
4.1.1 Respuesta de la función sigmoidal 27
4.1.2 Comportamiento de los sensores 29
4.2 Experimentos sobre un simulador 31
4.2.1 *Gazebo* 31
4.3 Experimentos en un entorno real 41

5 Conclusiones **55**
5.1 Resumen del proyecto 55
5.2 Conclusiones finales 55
5.3 Mejoras futuras 56

Bibliografía **59**

ÍNDICE DE FIGURAS

1.1 Ejemplo de vehículo aéreo no tripulado militar. 2
1.2 UAV diseñado para fumigar campos agrícolas. 2
1.3 *DJI inspire*, UAV diseñado para realizar tomas aéreas. 3
1.4 *Ar.drone*, UAV usado en este proyecto. 3

2.1 Diseño que se propone para el desarrollo del módulo. 6
2.2 Arduino Nano. 7
2.3 Sensor de ultra sonidos MB1043. 8
2.4 APC220. 9
2.5 Esquemático obtenido después de definir todas las conexiones que hay entre los pines de los dispositivos usados. 12
2.6 Recomendación del fabricante para conseguir lecturas de varios sensores secuencialmente obteniéndolas a través de pines analógicos. 13
2.7 Pines que del sensor MB1043. 13
2.8 Diagrama de la planta del dispositivo. 14
2.9 Diseño obtenido. 15
2.10 Distintas vistas del módulo final obtenido. 15

3.1 Estructura modular del diseño software propuesto para este proyecto. . . . 18
3.2 Diagrama de flujo del software creado para realizar la lectura y transmisión de los sensores. 19
3.3 Trama utilizada para transmitir los datos obtenidos mediante los sensores de ultrasonidos en la que I representa el inicio de trama, la N representa la distancia captada por el sensor norte, la S la distancia proporcionada por el sensor sur, la E por el sensor este, la O por el sensor oeste y el salto de línea representa el final de la trama. Las posiciones en blanco son espacios que nos marcarán el final de las lecturas de los sensores. 19
3.4 Arquitectura que representa gráficamente la conexión entre los nodos de ROS del sistema. 20
3.5 Joystick usado durante el desarrollo de este proyecto. 21
3.6 Ejemplo del campo vectorial que crean los obstáculos en función de la distancia a la que nos encontramos. En el gráfico, el punto central es un obstáculo 22
3.7 Ejemplo del campo vectorial resultante al combinar un obstáculo, círculo marrón, con un punto objetivo, círculo azul. 23
3.8 Comportamiento de la función sigmoidal. 24
3.9 Comportamiento de la función sigmoidal utilizada. 24

3.10 Convenio de signos que usa el dron para navegar. 25

4.1 Escenario 1: el dron está expuesto a una pared frontal. 28
4.2 Escenario 2: el dron está expuesto a una esquina. 28
4.3 Escenario 3: el dron está expuesto a una esquina. 28
4.4 Escenario 4: está expuesto a una estructura tipo *callejón sin salida*. 29
4.5 Resultado de la función sigmoidal al ser expuesta al escenario 1 modificando la distancia al obstáculo. El vector azul representa el vector de repulsión resultante. 30
4.6 Resultado de la función sigmoidal al ser expuesta al escenario 2 modificando la distancia a los obstáculos. El vector azul representa el vector de repulsión resultante para los vectores rojos que se corresponden con la repulsión que crea cada obstáculo. 31
4.7 Resultado de la función sigmoidal al ser expuesta al escenario 3 modificando la distancia a los obstáculos. El vector azul representa el vector de repulsión resultante para los vectores rojos que se corresponden con la repulsión que crea cada obstáculo. 32
4.8 Resultado de la función sigmoidal al ser expuesta al escenario 4 modificando la distancia a los obstáculos. El vector azul representa el vector de repulsión resultante para los vectores rojos que representan la repulsión que crea cada obstáculo (el vector de repulsión de la pared vertical coincide con el final). 33
4.9 Escenario para comprobar el correcto funcionamiento de los sensores instalados sobre el dron. 34
4.10 Vectores de repulsión resultantes sobre el robot real durante el recorrido de la figura 4.9. 34
4.11 Ampliación de la línea vertical de vectores situada más a la derecha en la figura 4.10. 35
4.12 escenario que proporciona *tum_simulator* por defecto. 35
4.13 Escenario simulado 1, el cual consta de un único obstáculo al frente. 36
4.14 Escenario simulado 2, el cual consta de dos obstáculo enfrentados. 36
4.15 Escenario simulado 3, el cual consta de dos obstáculos no enfrentados. . . 37
4.16 Escenario simulado 4, el cual consta de dos obstáculos y el vehículo navega entre los dos. 37
4.17 En el escenario 5 se intenta emular el comportamiento del dron en una área completamente cerrada. 38
4.18 Simulación del módulo de ultrasonidos en el *Ar.drone* en *Gazebo*. 38
4.19 Trayectoria al hacer volar el dron en el escenario 1 (figura 4.13), donde el objeto rojo es el obstáculo, la línea azul es el recorrido realizado por el dron, el rombo negro es el punto inicial y el triángulo verde es el punto final. . . . 39
4.20 Recorrido realizado por el dron al volar en el escenario 2 (figura 4.14), donde los objetos rojos son los obstáculos, la línea azul es el recorrido realizado por el dron, el rombo negro es el punto inicial y el triángulo verde es el punto final. 40
4.21 Camino marcado por el algoritmo al volar el dron en el escenario (figura 4.15), donde los objetos rojos son los obstáculos, la línea azul es el recorrido realizado por el dron, el rombo negro es el punto inicial y el triángulo verde es el punto final. 41

4.22 Trayectoria tomada por el dron al ser pilotado en el escenario 4 (figura 4.16), donde los objetos rojos son los obstáculos, la línea azul es el recorrido realizado por el dron, el rombo negro es el punto inicial y el triángulo verde es el punto final. 42
4.23 Camino descrito por el robot al ser expuesto al entorno simulado en el escenario 5 (figura 4.16), donde los objetos rojos son los obstáculos, la línea azul es el recorrido realizado por el dron, el rombo negro es el punto inicial y el triángulo verde es el punto final. 43
4.24 Velocidades obtenidas al hacer volar el dron en el escenario 1 (figura 4.13) 44
4.25 Velocidades obtenidas al hacer volar el dron en el escenario 2 (figura 4.14) 45
4.26 Velocidades obtenidas al hacer volar el dron en el escenario 3 (figura 4.15) 46
4.27 Velocidades obtenidas al hacer volar el dron en el escenario 4 (figura 4.16) 47
4.28 Velocidades obtenidas al hacer volar el dron en el escenario 5 (figura 4.17) 48
4.29 Comportamiento del dron al ser aproximado a la esquina formada por dos paredes. En la figura, los elementos rojos son las paredes, el trazo azul la dirección del dron, el rombo negro el punto inicial y el triángulo verde el final. 49
4.30 (Arriba) Comportamiento del dron al ser aproximado a un obstáculo de frente. En la figura, los elementos rojos son las paredes, el trazo azul la dirección del dron, el rombo negro el punto inicial y el triángulo verde el final. (Abajo) Escenario utilizado para el experimento. 50
4.31 (Arriba) Comportamiento del dron al moverse entre dos paredes paralelas. En la figura, los elementos rojos son las paredes, el trazo azul la dirección del dron, el rombo negro el punto inicial y el triángulo verde el final. (Abajo) Escenario utilizado en el experimento 51
4.32 Contraposición de la velocidad deseada por el usuario contra la velocidad que realmente ha tomado el dron durante el experimento de la figura 4.29. 52
4.33 Contraposición de la velocidad deseada por el usuario contra la velocidad que realmente ha tomado el dron durante el experimento de la figura 4.30. 53
4.34 Contraposición de la velocidad deseada por el usuario contra la velocidad que realmente ha tomado el dron durante el experimento de la figura 4.31. 54

5.1 Dispositivo con el que se podría sustituir el APC220. 57

ÍNDICE DE TABLAS

2.1 Especificaciones del Arduino Nano. 7
2.2 Especificaciones técnicas MB1043. 8
2.3 Especificaciones del APC220. 9
2.4 Opciones de inicialización del APC220. 9
2.5 Especificaciones de la cámara frontal. 10
2.6 Especificaciones de los motores. 10
2.7 Especificaciones electrónicas. 11
2.8 Pines disponibles en el APC220. 14

Resumen

La robótica aérea, en especial el uso de drones, ha sumado importancia en los últimos tiempos debido a la gran cantidad de aplicaciones que posee y la aparición de vehículos de bajo coste de uso recreativo, como por ejemplo el Ar.drone de la empresa Parrot. Estos vehículos están ideados para ser pilotados directamente por un humano, pero no poseen capacidades para evitar obstáculos. Para proporcionar estas habilidades a este tipo de vehículos, en este proyecto se ha desarrollado un módulo basado en ultrasonidos junto con una estrategia que permite dotar de estas capacidades a un dron de uso recreativo. El sistema se ha validado sobre un entorno simulado y sobre una plataforma real, presentando resultados satisfactorios en ambos casos.

CAPÍTULO 1

INTRODUCCIÓN

1.1 Robótica aérea en la actualidad

A lo largo de los últimos años se ha producido una gran revolución en el ámbito de la robótica, en especial, en el campo de la robótica aérea. Gracias a ello, ha habido una notable proliferación de aplicaciones usando esas nuevas tecnologías, lo que ha dado lugar a que las empresas sigan invirtiendo en ellas, llegando a desarrollar tecnologías que serían impensables hace no tantos años. Entre los ámbitos en los que se ha conseguido una mayor importancia, destacan las aplicaciones militares, agrícolas y cinematográficas.

1.1.1 Aplicaciones militares

No es de extrañar que los ejércitos apliquen las tecnologías más vanguardistas para el desarrollo de armamento. La robótica aérea ha sido sin duda alguna una de las tecnologías que más usos ha demostrado tener en este ámbito. Actualmente, los ejércitos se están centrando en el desarrollo de vehículos no tripulados de combate aéreo también conocidos como UCAV (*unmanned combat air vehicle*, ver figura 1.1). Al no estar sujetos a las exigencias que tienen los vehículos aéreos tripulados como pueden ser, entre otros, el blindaje y garantizar el soporte vital (por ejemplo calidad del aire y la presión atmosférica), estas aeronaves poseen un menor peso y dimensión [1]. También presentan la ventaja de que, al no estar tripulados, no se arriesgan vidas humanas. Estos dispositivos se usan mayormente para espionaje en zonas hostiles.

1.1.2 Aplicaciones agrícolas

El uso de UAV (*unmanned air vehicle*) en el ámbito civil no ha gozado de un crecimiento tan significativo como en el militar. No obstante, debido a la gran fiabilidad que presentan las últimas generaciones de drones comerciales, se están empezando a usar en tareas agrícolas tales como control de plagas o fumigación (figura 1.2). Mediante

Figura 1.1: Ejemplo de vehículo aéreo no tripulado militar.

Figura 1.2: UAV diseñado para fumigar campos agrícolas.

el uso de UAV se pueden obtener datos y fumigar de forma más localizada que al usar medios tradicionales (aviones o satélites) gracias a que estos sistemas permiten vuelos más bajos[1].

1.1.3 Aplicaciones cinematográficas

El desarrollo de cámaras con ópticas de gran calidad y tamaño reducido ha permitido que éstas se incluyan en drones, convirtiéndolos en herramientas muy interesantes para las grandes productoras que ven en estos UAV una manera de reducir costes a la hora de realizar tomas aéreas. Tradicionalmente, éstas tenían que ser tomadas por operarios de cámara subidos en un helicóptero con todos los gastos que eso representa. No obstante, no sólo se reducen los costes. Los UAV proporcionan una enorme versatilidad a la hora de hacer tomas ya que su tamaño permite el acceso a zonas que serían impensables usando medios aéreos tripulados. En la figura 1.3, se observa un vehículo aéreo preparado para el desempeño de estas tareas.

1.2 Drones comerciales

Bajo este panorama, han surgido muchas empresas que fabrican una enorme variedad de gamas de drones, como por ejemplo, DJI, Syma y Parrot. Estas marcas están desa-

[1] http://www.interempresas.net/Horticola/Articulos/151745-El-uso-de-robots-en-tareas-agricolas.html

Figura 1.3: *DJI inspire,* UAV diseñado para realizar tomas aéreas.

Figura 1.4: *Ar.drone,* UAV usado en este proyecto.

rrollando conceptos muy interesantes que están recibiendo una gran aceptación en el mercado. Pero estos dispositivos, generalmente, no son capaces de tomar decisiones propias sino que dependen mayoritariamente del operador (excepto para navegar por puntos de paso usando GPS), por lo que no son capaces de evitar obstáculos. Además, estos vehículos son típicamente sistemas cerrados sin posibilidad, a priori, de extender sus capacidades. Un ejemplo de estos dispositivos, es el Ar.Drone (figura 1.4), fabricado por Parrot, el cual es un cuadricóptero civil que, siguiendo el patrón descrito con anterioridad, es programable pero difícilmente ampliable.

1.3 Sensores de ultrasonidos

Existen varios métodos para determinar las distancias a las que se encuentran los objetos en un entorno. Uno de ellos son los sensores de ultrasonidos. Este tipo de sensores no son los más precisos. No obstante, son ampliamente usados en todo tipo de proyectos debido a su robustez, bajo coste y reducido peso.

1.4 Objetivos del proyecto

Teniendo en consideración todo lo dicho anteriormente, en este proyecto se va a desarrollar un sistema de evitación de obstáculos para el robot Ar.Drone basado en ultrasonidos. Éste deberá ser capaz de detectar obstáculos del entorno y decidir la nueva dirección que debe tomar el dron para poder salvarlos sin peligro de colisión.

Para llevar a cabo el objetivo principal del proyecto, se ha desarrollado, en primer lugar, un módulo basado en ultrasonidos que nos permite detectar los obstáculos cercanos al dron. En segundo lugar, se ha desarrollado una estrategia de evitación de obstáculos basada en la información recibida desde el módulo de ultrasonidos que genera los comandos de velocidad necesarios para evitar los obstáculos.

1.5 Organización del documento

A continuación se van a detallar los temas tratados en los capítulos de este documento:

- **Capítulo 2**: En este capítulo, se van a detallar todos los aspectos relacionados con el desarrollo del módulo encargado de detectar los obstáculos mediante sensores de ultrasonidos.
- **Capítulo 3**: En este punto, se van a explicar los algoritmos que se ejecutan en la estación base y que se encargan de determinar las velocidades que debe tomar el dron teniendo en cuenta las lecturas de los sensores de ultrasonidos.
- **Capítulo 4**: En este apartado, se van a mostrar y explicar los resultados experimentales obtenidos bajo simulación y en entornos reales.
- **Capítulo 5**: Para finalizar, se va a incluir un breve resumen del trabajo llevado a cabo, las conclusiones a las que se ha llegado durante y al final del desarrollo del proyecto y se van a proponer posibles mejoras que se podrían realizar en el futuro para mejorar el funcionamiento del sistema creado.

CAPÍTULO 2

MÓDULO BASADO EN ULTRASONIDOS PARA LA DETECCIÓN DE OBSTÁCULOS

2.1 Vista general

En esta sección, se detallará el diseño propuesto para realizar un circuito impreso que permita capturar las distancias de los obstáculos situados en el entorno del dron usando sensores de ultrasonidos y transmitirlas mediante radiofrecuencia a la estación base. Al estar tratando con un robot aéreo, va a ser necesario actualizar constantemente las posiciones de todos los obstáculos que se encuentren cerca de él en su mismo plano de vuelo, por lo que los sensores se deberán colocar de forma que cubran la mayor cantidad de área en todas las direcciones alrededor del dron. También, teniendo en cuenta que las constantes de control del Ar.Drone están calculadas para funcionar con el peso de sus carcasas, no podemos poner pesos muy elevados ya que podrían limitar mucho la capacidad de volar que tiene el vehículo. Es por ello que se decidió usar el mínimo número de sensores, que es cuatro, para poder cubrir todas las direcciones, reduciendo así el peso de la placa. También se ha incluido un microcontrolador para muestrear los sensores de ultrasonidos y un sistema de transmisión mediante radiofrecuencia para enviar los datos a una estación base donde serán procesados y se generará el correspondiente comando de control. En la figura 2.1 podemos ver un diagrama del diseño propuesto. Este circuito va a ir montado sobre la carcasa del Ar.drone diseñada para vuelos en zonas interiores.

2.2 Componentes hardware

Para el desarrollo de este circuito, ha sido necesario usar componentes de diversos fabricantes. En los sucesivos subapartados, se van a describir estos dispositivos y sus especificaciones.

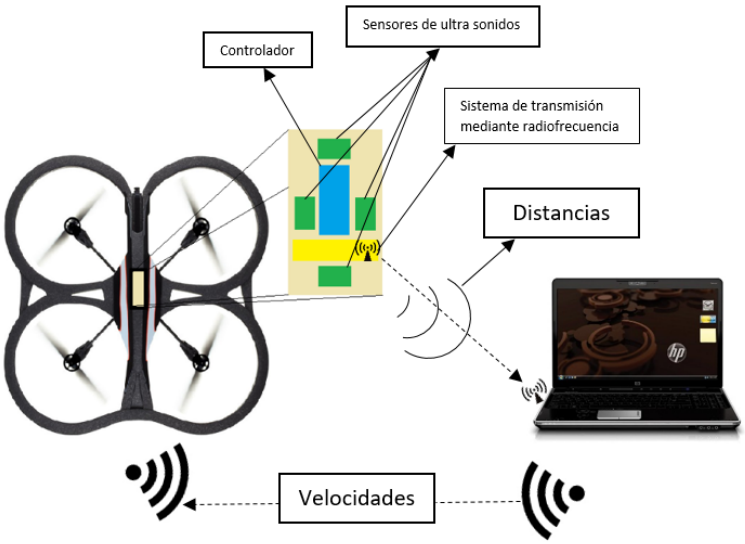

Figura 2.1: Diseño que se propone para el desarrollo del módulo.

2.2.1 Arduino

En el panorama actual, existen una enorme cantidad de microcontroladores. Generalmente, tienen sus propios requisitos de implementación, lenguajes de programación y compiladores, por lo que el uso de uno de ellos implica una inversión de tiempo, al inicio, para su aprendizaje. En estas circunstancias surge Arduino, que propone un sistema de hardware libre, mediante el que cualquiera puede publicar sus diseños para un caso en concreto. Si se observan las distintas placas creadas por este fabricante, se puede apreciar como todas ellas comparten la misma distribución de *pinout*, por lo que la transición a una placa nueva no requiere ninguna curva de aprendizaje en lo referente al hardware. Además, la inclusión de pines analógicos, digitales y pwm (*pulse-width modulation* por sus siglas en inglés) entre otros, hace que estos productos sean muy versátiles. En el caso de este proyecto, se ha decidido usar el modelo Nano (figura 2.2) ya que es el que tiene el peso más reducido y está pensado para ser montado en una PCB (*Printed Circuit Board*). En la tabla 2.1, se observan las características del Arduino Nano.

En el caso de este proyecto, se van a necesitar cuatro pines analógicos para las salidas de los sensores, un pin digital para activar la secuencia de lecturas, un pin de alimentación a 5V y una toma de tierra. El Arduino Nano dispone de ocho pines analógicos, veintidós digitales, uno destinado a la alimentación de 5V y dos tierras, por lo que en este caso, este controlador, va a ser suficiente ya que tenemos suficientes pines de cada tipo para cubrir las necesidades requeridas.

2.2.2 Sensores de ultrasonidos

En la actualidad, existe una enorme cantidad de sensores con los que se pueden obtener información del entorno. Incorporar estos sensores a sistemas de los que esperamos

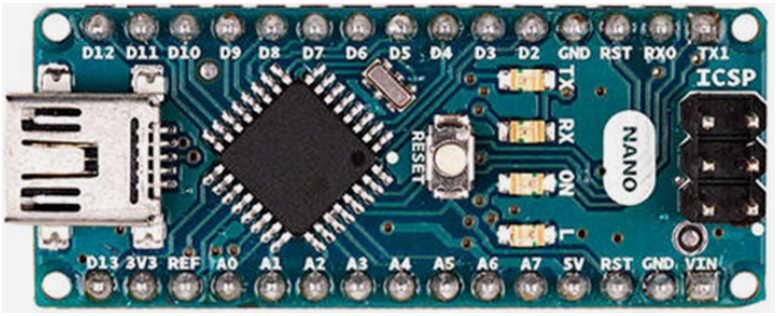

Figura 2.2: Arduino Nano.

Microcontrolador	ATmega328
Arquitectura	AVR
Tensión nominal	5V
Memoria Flash	32 KB
SRAM	2KB
Velocidad de reloj	16MHz
Número de pines analógicos	8
EEPROM	1KB
Intensidad DC para pines I/O	40mA
Tensión de entrada	7-12 V
Número de pines digitales	22
Pines PWM	6
Consumo	19mA
Dimensiones	18 x 45 mm
Peso	7 g

Tabla 2.1: Especificaciones del Arduino Nano.

que tomen decisiones en función de los cambios que se produzcan en su entorno es algo necesario. En el caso que nos ocupa, al querer evitar obstáculos, estos sensores van a tener que medir la distancia que separa el dron de los obstáculos que se puedan encontrar a su alrededor.

Los sensores que se usan para medir distancias se pueden diferenciar, fundamentalmente, en dos grupos: sensores *láser*, los cuales emiten un haz de luz que al encontrar un objeto rebota y miden el tiempo que necesita el haz para realizar el recorrido de ida y vuelta, por lo que sabiendo la velocidad a la que se propaga la luz, se puede determinar la distancia de los objetos; y sensores de *ultrasonidos*, los cuales tienen un funcionamiento bastante parecido a los sensores láser aunque en lugar de emitir haces de luz emiten ondas sonoras. Estos sensores son menos precisos que los que funcionan mediante láser pero mucho más económicos. En esta ocasión, debido a su bajo peso y coste, se van a usar sensores de ultrasonidos. En concreto, el modelo elegido es el MB1043 (figura 2.3) de la gama MaxSonar-EZ fabricados por MaxBotix. Estos sensores

Figura 2.3: Sensor de ultra sonidos MB1043.

Tensión de alimentación	2.5V-5.5V DC
Intensidad nominal	2.5mA a 3.3V o 3.1mA a 5V
Máxima distancia	5m
Mínima distancia	30cm
Rango de temperaturas	-15°C - +65°C
Beam	**60cm a una distancia de 5m**

Tabla 2.2: Especificaciones técnicas MB1043.

han sido elegidos debido a que disponen de un *beam* muy estrecho, por lo que se evita que los sensores puedan dar falsas lecturas al detectar, por error, la misma estructura del dron.

En la tabla 2.2 se observan las características del sensor escogido (para más información véase el *datasheet* del dispositivo[1]).

2.2.3 Transmisión por radiofrecuencia

Para transmitir los datos a la estación base, se utiliza un módulo de radiofrecuencia, en concreto, el APC220 (figura 2.4). Dada la eficiencia de sus sistemas de detección de error, este sistema es apto para funcionar en entornos con muchas interferencias. Gracias a su enorme *buffer*, este dispositivo es capaz de transmitir enormes cantidades de datos pudiendo configurar diversos parámetros como su velocidad de transmisión, entre otros, los cuales van a ser detallados más adelante.

Si se observa la figura 2.4, se ve que estos dispositivos incluyen también un conector USB, cuya función es conectar el APC220 a un ordenador a través de una conexión serie.

En la tabla 2.3 se recogen las especificaciones técnicas del APC220 y en la tabla 2.4 se detallan los parámetros con los que se puede inicializar el componente. Para más información sobre este dispositivo se recomienda consultar su *datasheet*[2].

[1] https://www.maxbotix.com/documents/HRLV-MaxSonar-EZ_Datasheet.pdf
[2] http://image.dfrobot.com/image/data/TEL0005/APC220_Datasheet.pdf

Figura 2.4: APC220.

Velocidad de reloj	418-455MHz
Modulación	GFSK
Intervalo de frecuencia	220KHz
Potencia transmitida	20mw
Sensibilidad recibida	-113dBm@9600bps
Air rate	2400 - 19200bps
UART rate	1200 - 57600bps
Buffer	256 bytes
Humedad	0.1 - 0.9
Temperatura	-30°C - 85°C
Tensión de alimentación	3.5 - 5.5V (El rizado es de 50mV)
Intensidad de transmisión	mayor o igual 42mA@20mW
Intensidad de recepción	mayor o igual 28mA
Intensidad durante hivernación	mayor o igual 5uA
Distancia de transmisión	1000m (en un espacio abierto)
Dimensiones	37.5mm x 18.3mm x 7.0mm

Tabla 2.3: Especificaciones del APC220.

Parámetros	Opciones	Defecto
UART rate	1200,2400,4800,9600,19200,38400,57600	9600bps
Paridad	Disable,Even Parity,Odd Parity	Disable
Frecuencia	418MHz-455MHz	434 MHz
Air Rate	2400bps,4800bps,9600bps,19200bps	9600bps
RF Power	0-9(9 for 20mw)	9(20mw)

Tabla 2.4: Opciones de inicialización del APC220.

Grabación de vídeo	
Cámara	HD 720p 30FPS
Objetivo	Gran angular con una diagonal de 92°
Perfil de codificación básica	H264
Formato de las fotos	JPG

Tabla 2.5: Especificaciones de la cámara frontal.

Motores	
Motores	Motores sin escobillas de tipo inrunner: 14.5 vatios y 28500rpm
Rodamientos	Bolas en miniatura autolubricante
Engranajes	Nylatron

Tabla 2.6: Especificaciones de los motores.

2.2.4 Ar.Drone

Como ya se ha comentado anteriormente, nuestro módulo de detección de obstáculos irá instalado sobre un vehículo Ar.Drone. Este dron genera una red wifi en la que se puede publicar la velocidad que se desea que tome el dron usando un paquete llamado *ardrone_autonomy*, cosa bastante útil en el caso de este proyecto, ya que permitirá modificar el sentido de desplazamiento en función de las distancias detectadas por los sensores de una forma realmente sencilla.

Si se observa la tabla 2.5, se ven las características técnicas de la cámara frontal del vehículo. En la tabla 2.6 se especifican todos los elementos mecánicos que componen el dron. Para terminar, en la tabla 2.7, se observan los distintos componentes electrónicos que incluye el dron. En la parte inferior de la tabla (filas marcadas en negrita) podemos ver los sensores que instala el fabricante. Ninguno de esos componentes va a ser útil para aplicar una estrategia de evitación de obstáculos ya que sólo uno está diseñado para detectar obstáculos (el de ultrasonidos) y está apuntando hacia el suelo para estimar la altura del vehículo.

2.3 Diseño del circuito impreso

El objetivo de esta parte del proyecto es la creación de un circuito impreso o PCB utilizando los componentes comentados en el apartado anterior. Este circuito deberá ir montado sobre la estructura del Ar.Drone e ir midiendo la distancia que separa el vehículo de los distintos obstáculos que pueda haber en su entorno. Por eso, el circuito resultante debe tener unas medidas muy determinadas, ya que de otra forma puede desestabilizar el dron.

Para desarrollar el PCB, se ha recurrido al software llamado *Eagle*, que es un programa para ordenador, creado para desarrollar circuitos impresos de una forma sencilla, pudiendo incluir los componentes que se necesiten (mediante librerías ya creadas) o crearlos desde cero. En este apartado se mostrarán los diseños obtenidos mediante

Electrónica	
Procesador	1 GHz 32 bits ARM Cortex A8
S.O.	Linux 2.6.32
RAM	DDR2 1 GB a 200 MHz
USB	USB 2.0
Wi-Fi	Wi-Fi b g n
Giroscopio	**3 ejes, precisión de 2000°/segundo**
Acelerómetro	**3 ejes, precisión aproximada: 50 mg**
Magnetómetro	**3 ejes, precisión de 6°**
Sensor de presión	**Precisión aproximada 10 Pa**
Ultrasonidos	**Apuntando hacia abajo**
Cámara vertical	**QVGA 60 FPS**

Tabla 2.7: Especificaciones electrónicas.

este software.

Para desarrollar un proyecto en *Eagle*, primero se tiene que crear un esquemático (*schematic*), en el cual se especifica cómo van conectados los pines de los distintos componentes y, a partir de este archivo, *Eagle* representa automáticamente un archivo *board*. Este archivo permitirá al usuario decidir cómo deben ir las pistas del PCB sin poder modificar las conexiones establecidas en el esquemático, es decir, se va a establecer por dónde pasan las pistas sobre el diseño creado previamente. En la figura 2.5, se ve un esquema en el que se define cómo deben ir conectados todos los dispositivos en el circuito impreso. En los sucesivos párrafos se detallarán las conexiones definidas en el esquemático.

2.3.1 Conexión de los sensores de ultrasonidos

Si se observa la figura 2.6, se puede ver como el fabricante recomienda conectar varios sensores para que funcionen secuencialmente. Para ello, se va a conectar una de las salidas digitales del Arduino, en concreto el pin D9 (véase figura 2.2), al pin 4 del sensor (figura 2.7). Con eso, enviando un pulso de al menos 20μs, se consigue que el primer sensor empiece a medir distancia. Al concluir la medición, se detiene, y mediante el pin 2, enviará una señal de disparo que irá conectada al pin 4 del siguiente sensor. Este proceso se repite para todos los sensores exceptuando el último, el cual únicamente recibirá la señal de disparo ya que al volver a empezar el ciclo, el controlador volverá a enviar mediante el pin D9, un nuevo pulso para empezar el ciclo de nuevo. Al observar la figura 2.5, se ve como el primer sensor que efectúa su lectura es el *west*, seguido por *north*, *east* y *south*. Para capturar las lecturas de los sensores, se tuvieron que conectar los pines 3 de los sensores a entradas analógicas del Arduino. Si observamos el esquemático (figura 2.5), podemos ver como, el sensor *west*, ha sido conectado al pin A6, *north* al A5, *east* al A0 y el sensor *south* al pin A4. Se usaron estos pines para facilitar el diseño de la placa.

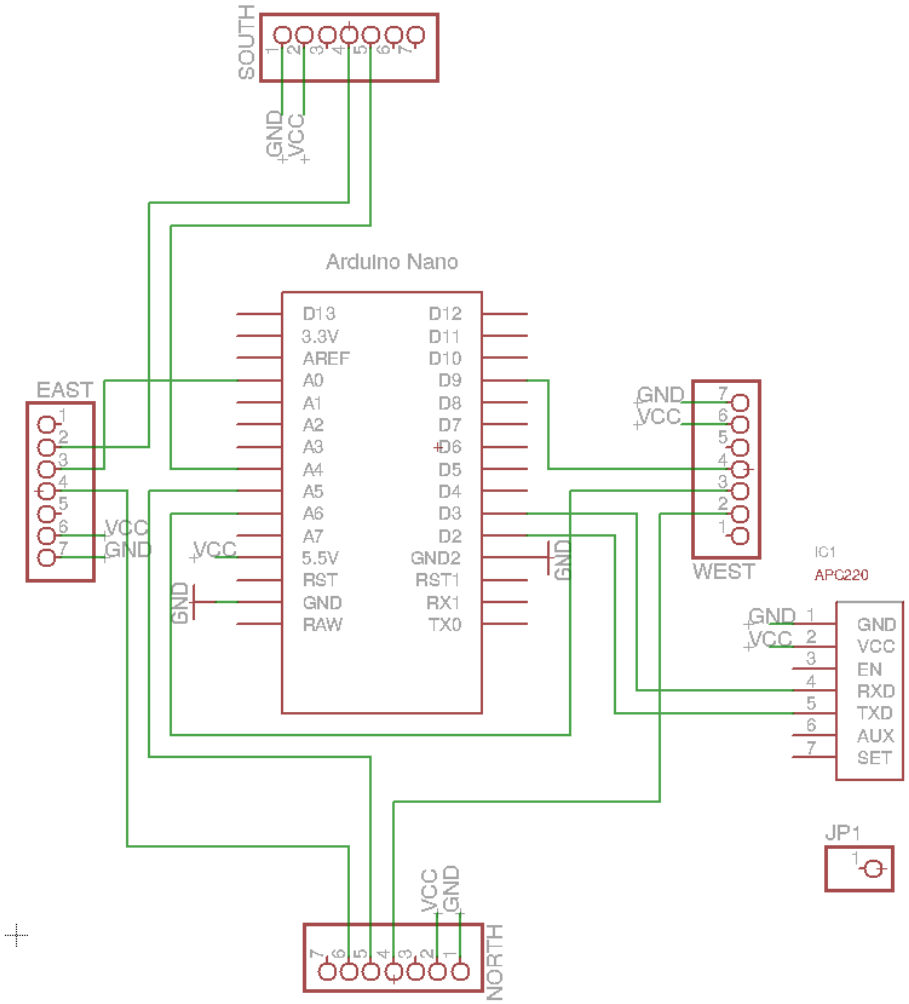

Figura 2.5: Esquemático obtenido después de definir todas las conexiones que hay entre los pines de los dispositivos usados.

Si se observa la última fila de celdas de la tabla 2.2, se puede ver la principal razón por la que se escogieron estos sensores. Al tener un *beam* de 60 cm en su máxima distancia, es muy difícil que se obtengan lecturas del cuerpo del dron ya que, el módulo irá montado sobre la parte central de la carcasa(figura 2.1). Además, la técnica de lectura usada, consistente en ir activando los sensores secuencialmente, evita que se produzcan interferencias entre los sensores, debido a que, mientras uno está obteniendo datos, los otros están inactivos.

Si se observa la figura 2.7, se puede ver la manera en la que están distribuidos los pines en el sensor de ultrasonidos elegido. A continuación se explicará brevemente la funcionalidad que tiene cada uno de esos pines.

- El pin 1 permite conectar el MB1043 a un sensor de temperatura externo.

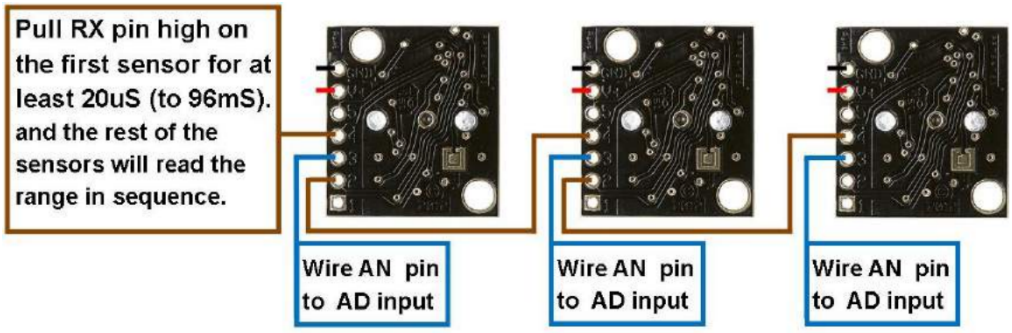

Figura 2.6: Recomendación del fabricante para conseguir lecturas de varios sensores secuencialmente obteniéndolas a través de pines analógicos.

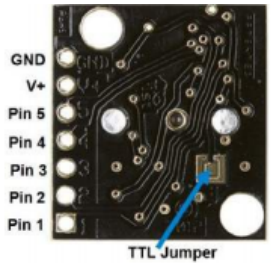

Figura 2.7: Pines que del sensor MB1043.

- El pin 2 proporciona la lectura realizada por el sensor mediante un pulso que representa la distancia. Es por ello por lo que este sensor puede ser usado para activar el siguiente, emite el pulso necesario para disparar el siguiente sensor.
- El pin 3 proporciona la lectura del sensor mediante una señal analógica.
- El pin 4 se usa para activar las lecturas del sensor. Si se deja desconectado o el pin se mantiene a 1, va a medir distancias continuamente. Si está conectado y recibiendo a 0, va a dejar de medir distancias.
- El pin 5 proporciona las lecturas del sensor mediante comunicación serie.
- El pin 6 es el encargado de la alimentación.
- El pin 7 es la toma de tierra.

2.3.2 Conexión del transmisor de radiofrecuencia

En la tabla 2.8 se detalla el *pinout* del APC220. El pin número 1 definido en la tabla se corresponde con el pin que está alineado con la antena del dispositivo (véase figura 2.8).

En la figura 2.5, se observa que los pines TXD y RXD del dispositivo no están conectados a los pines físicos de transmisión por serie del Arduino. En su lugar, se han

Electrónica		
Número de pin	Nombre del pin	Descripción
1	GND	Tierra
2	VCC	Alimentación 3.5V hasta 5.5V
3	EN	>1.6V o sin conectar activo, <0.5V hibernación
4	RXD	Pin de recepción de datos
5	TXD	Pin de transmisión de datos
6	MUX	Para usar el módulo para otras funciones (no usado)
7	SET	Activar los parámetros
8	NC	No conectado
9	**NC**	**Se usa para obtener un mayor soporte mecánico**

Tabla 2.8: Pines disponibles en el APC220.

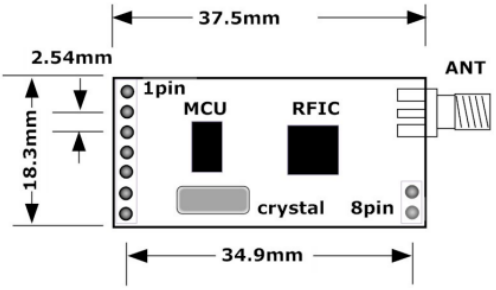

Figura 2.8: Diagrama de la planta del dispositivo.

creado mediante software ya que los físicos son usados para programar el Arduino. Para ello, se han usado los pines digitales D2 y D3. Como se puede ver en la última fila de la tabla 2.8, se ha usado uno de los pines que no tienen propósito para proporcionar un mayor soporte mecánico, ya que al ir montado sobre un vehículo aéreo las vibraciones de éste podrían hacer que se moviera de su sitio.

2.3.3 Montaje sobre el Ar.drone

Una vez finalizado el esquemático, se puede proceder al diseño físico de la placa, es decir, la forma que va a tener y cómo van a ser las pistas. Para ello, como se ha comentado anteriormente, se va a tener que crear una placa, que ha resultado en el modelo que se puede ver en la figura 2.9. La forma del PCB ha sido diseñada para evitar tener espacio innecesario que pueda añadir peso al dron, además de cuidar, en la medida de lo posible, la simetría. Para montar los dispositivos sobre la placa, se ha decidido usar *pinheads*, que nos permiten sustituir fácilmente los componentes, en caso de avería.

Este circuito va a ir montado sobre la parte superior del dron obteniendo el resultado final que se puede observar en la figura 2.10. El peso total del módulo es de 52 g, lo que añade muy poco peso a la plataforma.

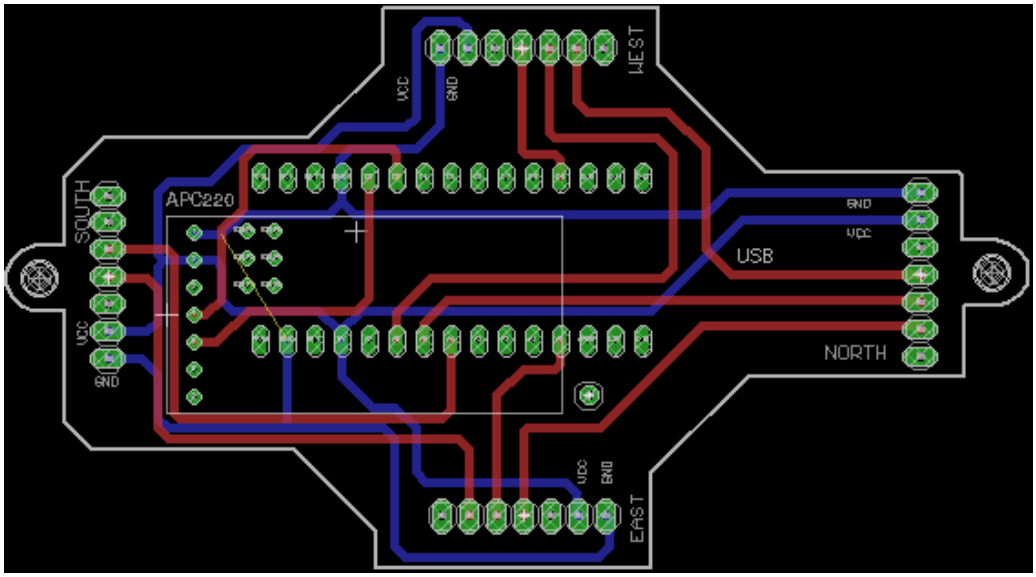

Figura 2.9: Diseño obtenido.

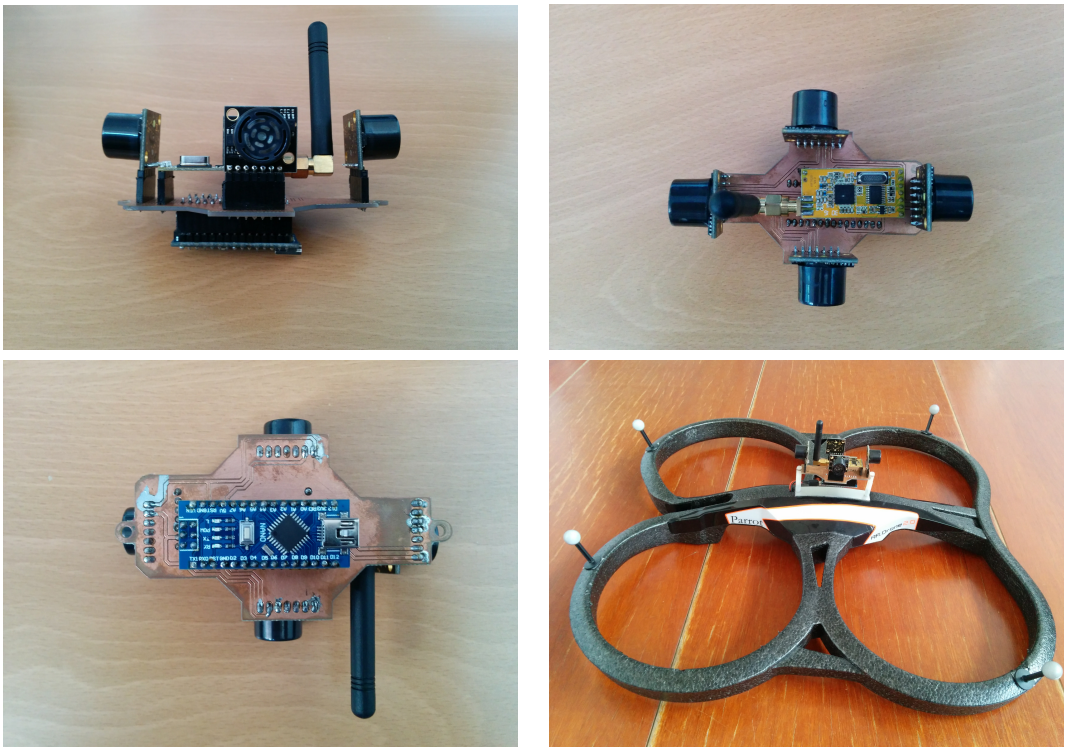

Figura 2.10: Distintas vistas del módulo final obtenido.

CAPÍTULO 3

ESTRATEGIA DE EVITACIÓN DE OBSTÁCULOS

3.1 Vista general

Una vez desarrollado el módulo hardware que nos permite obtener información del entorno, el siguiente paso es desarrollar el software necesario para realizar la evitación de obstáculos. Para ello, las distancias obtenidas a través de los sensores de ultrasonidos son enviadas a la estación base, que se encargará de procesarlas y decidir cómo actuar en función de ellas. Para llevar a cabo esta tarea, el módulo que se encargará de calcular las nuevas componentes de velocidad del dron obtendrá la velocidad que el usuario desea y las lecturas realizadas por los sensores. Usando una estrategia de evitación de obstáculos basada en campos de potencial determinará la nueva velocidad a seguir por el dron. Una vez obtenida esa velocidad, el mismo módulo se encargará de enviarla al dron a través de la red wifi que crea para que éste pueda tomar un nuevo rumbo. En la figura 3.1 se puede ver la estructura de software desarrollada para este proyecto.

3.2 *Robot Operating System*

Como se ha venido detallando a lo largo de este documento, en la actualidad, el número de aplicaciones que incluyen la robótica está sufriendo un crecimiento importante, por lo que la necesidad de un sistema de desarrollo de software era algo de suma importancia. Tratando de dar respuesta a esta necesidad surge *Robot Operating System* (ROS por sus siglas en inglés) que proporciona un entorno de trabajo para que los desarrolladores puedan llevar a cabo aplicaciones de la robótica [2].

ROS, proporciona mecanismos de comunicación entre entidades llamadas nodos, los cuales se comunican entre sí mediante el intercambio de mensajes, publicados en tópicos. Esto permite que los distintos nodos en un mismo proyecto no estén escritos en el mismo lenguaje y ejecutándose en distintas máquinas. Todo este intercambio de información está dirigido a través de un nodo maestro llamado *roscore* que gestiona las

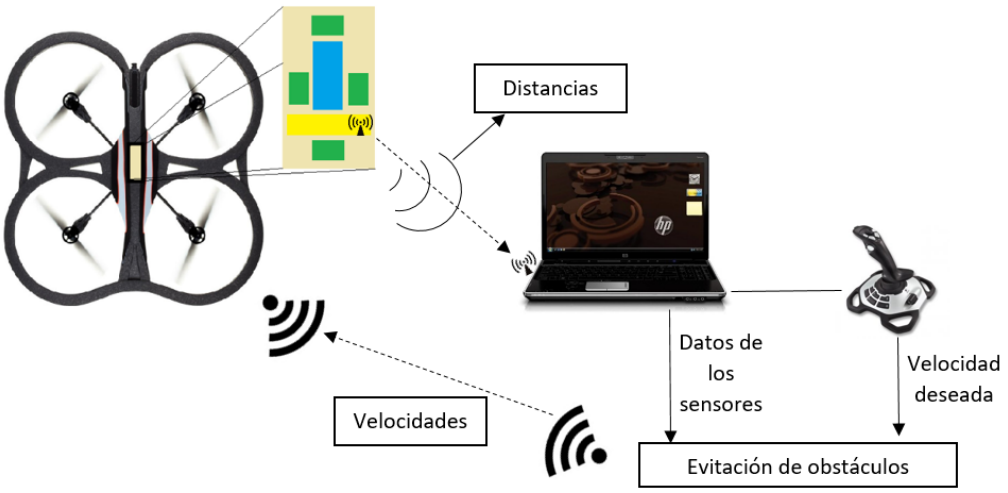

Figura 3.1: Estructura modular del diseño software propuesto para este proyecto.

comunicaciones entre el resto de nodos.

A pesar de ser un concepto muy innovador, la estructura de ROS no es su mayor ventaja. Su punto más diferencial es la enorme comunidad que está constantemente publicando paquetes y librerías relacionadas con diversos ámbitos de la robótica. Por ello, para un mismo problema, pueden existir una enorme cantidad de soluciones distintas, otorgando al usuario la capacidad de elegir la que se adapte más a sus necesidades. Es por este motivo por el que, en el caso de este proyecto, se ha decidido usar ROS como herramienta de desarrollo.

3.3 Obtención de distancias sobre Arduino

Para obtener las medidas de los sensores, se ha desarrollado un programa que dispara los sensores de manera secuencial, crea una trama con esas lecturas y finalmente las envia mediante el APC220. Para ello, se ha usado el Arduino IDE, y una librería ya creada de Arduino llamada *SoftwareSerial*. Esta librería nos permite establecer una comunicación serie usando pines digitales de la placa, por lo que aquellos que están físicamente preparados para ello (los que están conectados a la UART), pueden quedar libres para poder subir programas a la placa sin tener que desconectar los dispositivos que necesitan este tipo de conexión.

En la figura 3.2, se muestra el diagrama de flujo del método usado para obtener las lecturas de los ultrasonidos. Como se puede observar, el orden en el que se producen las lecturas de los datos resulta ser el mismo que se ha usado para conectar físicamente los sensores. Para filtrar los datos, se ha decidido usar un filtro de mediana para cada sensor, con los que se pretende eliminar las falsas detecciones que proporcionan las medidas al hacer, entre otras cosas, cambios de dirección muy bruscos. Para llevar a cabo este

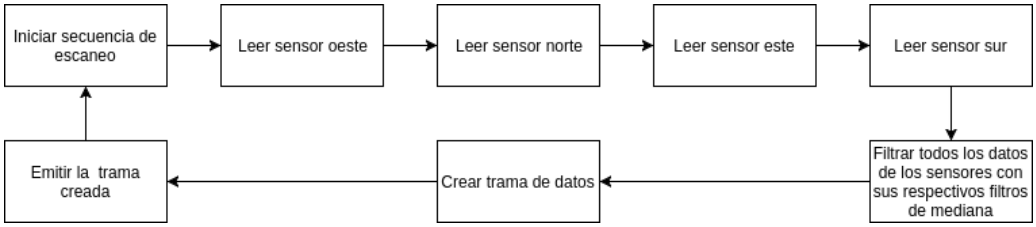

Figura 3.2: Diagrama de flujo del software creado para realizar la lectura y transmisión de los sensores.

Figura 3.3: Trama utilizada para transmitir los datos obtenidos mediante los sensores de ultrasonidos en la que I representa el inicio de trama, la N representa la distancia captada por el sensor norte, la S la distancia proporcionada por el sensor sur, la E por el sensor este, la O por el sensor oeste y el salto de línea representa el final de la trama. Las posiciones en blanco son espacios que nos marcarán el final de las lecturas de los sensores.

filtrado, se ha usado un librería de Arduino creada por un tercero[1], que nos proporciona la mediana de una sucesión limitada de valores. Al haber filtrado todas las medidas de distancia, para obtenerlas en milímetros, se tiene que multiplicar el valor analógico obtenido por cinco (este coeficiente es proporcionado por el fabricante). Tras haber obtenido las lecturas de todos los sensores de ultrasonidos, éstas se agrupan en una trama de datos de cara a enviarse a través de una conexión serie por radiofrecuencia. El formato de esta trama se muestra en la figura 3.3.

3.4 Software de la estación base

Como se ha mencionado en el primer apartado de este capítulo, para el desarrollo del sistema, se ha decidido usar el entorno de desarrollo ROS. Para ello se ha tenido que diseñar una serie de nodos que, mediante la comunicación a través de tópicos, consigan el comportamiento deseado. El diseño final obtenido se puede observar en la figura 3.4. A continuación se explican los detalles.

3.4.1 *ardrone_autonomy*

Este es un paquete de ROS, creado por terceros, que permite establecer comunicación con el dron mediante nodos a través de una red wifi creada por el propio dron, por lo que podremos enviarle órdenes para que despegue, aterrice o varíe su velocidad de vuelo. Este paquete también publica en tópicos información referente al dron como el estado de éste, la información obtenida de los sensores que componen el dron, las imágenes captadas por la cámara o el porcentaje de batería restante.

[1] https://github.com/daPhoosa/MedianFilter

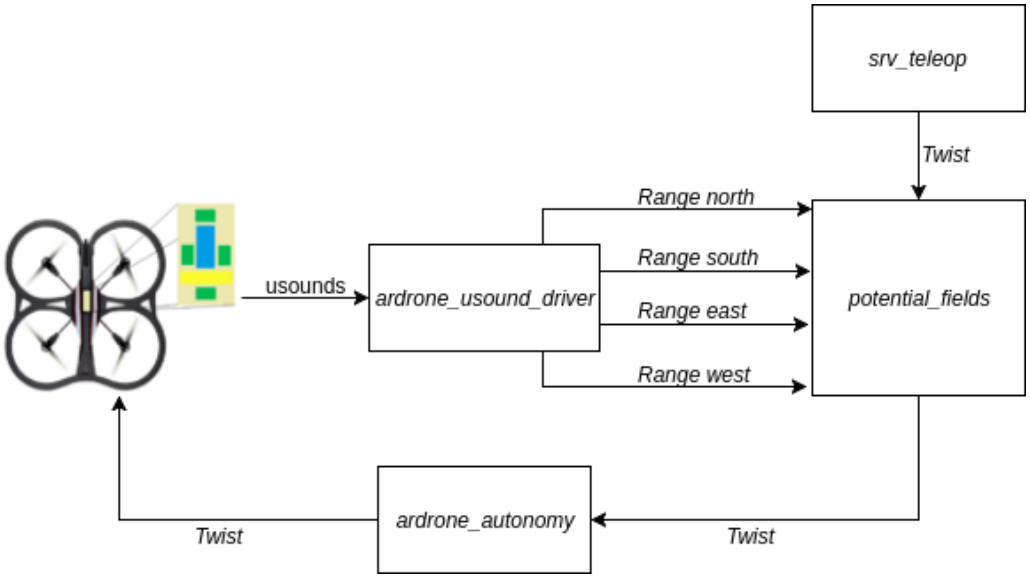

Figura 3.4: Arquitectura que representa gráficamente la conexión entre los nodos de ROS del sistema.

3.4.2 *srv_teleop*

Para definir la dirección que el usuario desea, se va a usar el *joystick EXTREME 3D PRO* de Logitech (véase la figura 3.5). Para ello, se va a tener que usar un *driver*, que obtenga los datos que envía el *joystick* y los publique en tópicos de ROS mediante mensajes de tipo *Twist*. Estos tópicos son aquellos a los que está suscrito *ardrone_autonomy*. En este caso, se ha decidido usar el paquete *srv_teleop* creado por Francisco Bonnín Pascual profesor de la *Universitat de les Illes Balears* y componente del equipo de investigación SRV. Este nodo publica la información del *joystick* mediante un tópico de tipo *Twist* el cuál contiene 6 variables: 3 para las velocidades lineales y 3 para las angulares (*roll*, *pitch*, *yaw*). El uso del *joystick* es necesario porque se trata de un vehículo teleoperado, asistido frente a colisiones a través del hardware/software objeto de este TFG.

3.4.3 *ardrone_usound_driver*

Este nodo ha sido desarrollado para obtener la información que envía periódicamente el módulo físico, decodificarla y transmitir las distancias recibidas a través de tópicos. Para ello, lee la trama que recibe por puerto serie (enviada por el APC220), la decodifica y publica esos valores usando mensajes de tipo *Range* de ROS.

Para leer los datos que llegan por el puerto serie se ha utilizado una librería llamada *LibSerial*, creada por terceros[2], la cual nos permite establecer una conexión serie por el puerto USB deseado, definir la velocidad en baudios de llegada, la paridad, la medida de los caracteres que van a llegar, el protocolo de control de flujo y el número de *stop bits* simplemente invocando a métodos de esta misma clase.

[2]http://libserial.sourceforge.net/x27.html

Figura 3.5: Joystick usado durante el desarrollo de este proyecto.

Tras recibir la trama, ésta se analiza y divide para obtener las diversas medidas de los ultrasonidos. Estas medidas se convierten a enteros y se publican, como ya se ha comentado, en sendos tópicos de ROS como mensajes *Range*.

3.4.4 *potential_fields*

Para que el dron evite obstáculos, se ha desarrollado un nodo que se encarga de tomar los mensajes de tipo *Range* publicados por *ardrone_usound_driver* y los combina con los mensajes de tipo *Twist* publicados por el nodo *srv_teleop* para obtener la velocidad que debe tomar el dron, que es enviada al dron mediante mensajes de tipo *Twist* publicados sobre el tópico *cmd_vel* creado por el paquete *ardrone_autonomy*. Se ha usado un método de evitación de obstáculos basado en campos de potencial.

Método de campos de potencial

Existen dos grandes categorías de arquitecturas de control: deliberativas y reactivas.

- Las arquitecturas **deliberativas**, mantienen una representación del entorno que les permite planificar y replanificar el camino a seguir. El proceso de planificación puede llegar a ser una tarea muy exigente a nivel computacional, por lo que si se produjera algún cambio en el entorno, no se podría generar una respuesta adecuada rápidamente [3].

- Las arquitecturas **reactivas**, a diferencia de las deliberativas, mantienen información mínima del mundo, básicamente se dedican a responder a las lecturas de sus sensores. Se podría decir que hay una conexión directa entre sensores y actuadores. Estos tipos de comportamientos permiten reaccionar rápidamente a los cambios inesperados que se puedan producir en el entorno, por lo que son

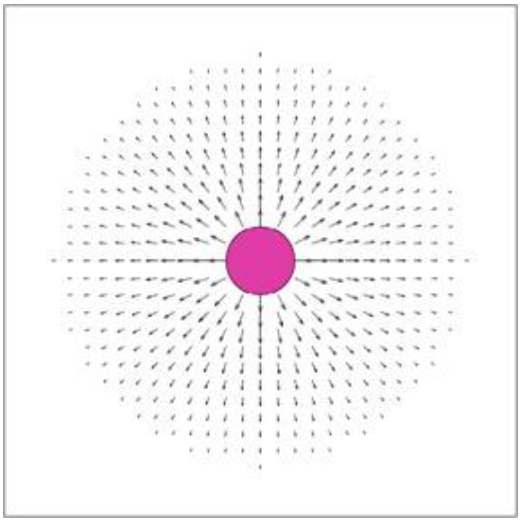

Figura 3.6: Ejemplo del campo vectorial que crean los obstáculos en función de la distancia a la que nos encontramos. En el gráfico, el punto central es un obstáculo

buenos en situaciones con objetos móviles, pero no permiten resolver tareas más complejas como el cálculo del camino óptimo [3].

Una de las técnicas más usadas en navegación reactiva son los campos de potencial [4]. Las estrategias de campos de potencial están basadas en los campos Newtonianos (gravitatorio, electrostático) en los cuales la fuerza del campo es inversamente proporcional a la distancia al elemento que lo crea. En el caso general, se replica este comportamiento por cada obstáculo, de forma que se crean vectores de repulsión que emanan de los obstáculos, cuyo módulo crece con la proximidad de los mismos. El punto objetivo, en cambio, genera vectores de atracción que se decrementan en función de la proximidad a dicho punto. En el caso de este proyecto, el punto objetivo se ha sustituido por los vectores generados por el *joystick* ya que el Ar.drone no posee sistemas para determinar su posición. La figura 3.6 [5] muestra un ejemplo del campo que crea un obstáculo al usar esta estrategia y la figura 3.7 [5] muestra el campo obtenido al combinar un punto objetivo con un obstáculo.

Estrategia implementada

Se ha creado un nodo llamado *potential_fields* que se encargará de recoger los datos publicados en los tópicos que ha creado *ardrone_usound_driver* y los que envía el *joystick* para determinar la dirección que deberá tomar el dron para navegar esquivando obstáculos durante la teleoperación.

Una vez obtenida la información de los sensores, calcula el módulo del vector de repulsión que los obstáculos crean usando una función que nos proporciona el comportamiento deseado, es decir, que a medida que el Ar.drone se acerca a los obstáculos el módulo del vector de repulsión aumenta, que a partir de una cierta distancia la repulsión sea nula y que a distancias inferiores a un límite la repulsión sea máxima. Para ello, se ha decidido usar la función sigmoidal descrita en la ecuación 3.1:

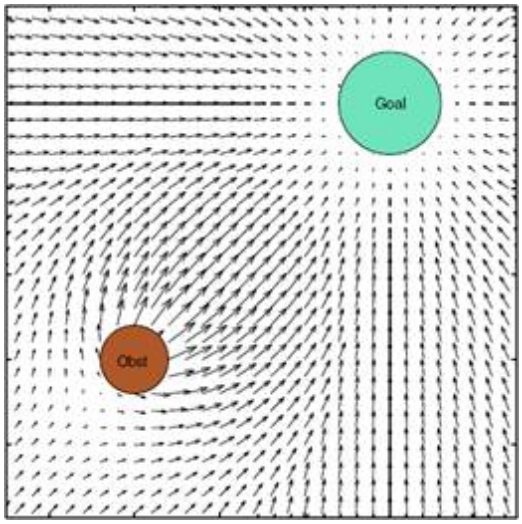

Figura 3.7: Ejemplo del campo vectorial resultante al combinar un obstáculo, círculo marrón, con un punto objetivo, círculo azul.

$$S^*(x) = \frac{1}{1+e^{-a(x-c)}} \tag{3.1}$$

donde a y c son parámetros que determinan, respectivamente, la pendiente y la posición del punto de inflexión y x, en este caso, es la distancia medida por el sensor. Si se aplica directamente la función, se obtiene el comportamiento mostrado en la figura 3.8 que es inverso al que se necesita de acuerdo con la definición de campo de potencial. Por ello, se ha decidido modificar la función para que la repulsión esté inversamente relacionada con la distancia obteniendo la siguiente expresión.

$$S(x) = 1 - \frac{1}{1+e^{-a(x-c)}} \tag{3.2}$$

La función S(x) se aplica sobre las lecturas obtenidas de los cuatro sensores usando $a = 4$ y $c = 1{,}9$ ya que nos proporcionan 3m como distancia mínima a la que el algoritmo empezará a hacer efecto y 1.9m como distancia máxima a la que el dron se puede acercar, por lo que obtendremos una repulsión dentro del rango de detección del dron, que es 5m. Al terminar este proceso, se crea un vector de repulsión cuyas componentes son las que se pueden ver en las ecuaciones 3.3 y 3.4, que posteriormente se normalizan de forma que el vector resultante sea unitario.

$$O_x = S(sur) - S(norte) \tag{3.3}$$

$$O_y = S(este) - S(oeste) \tag{3.4}$$

El convenio de signos ha sido escogido de forma que coincida con el que emplea ROS. Es por ello que los obstáculos en norte y oeste generan vectores de repulsión en sentido negativo y los obstáculos en sur y este en sentido positivo. La figura 3.10 ilustra el convenio de signos que rige la velocidad del dron.

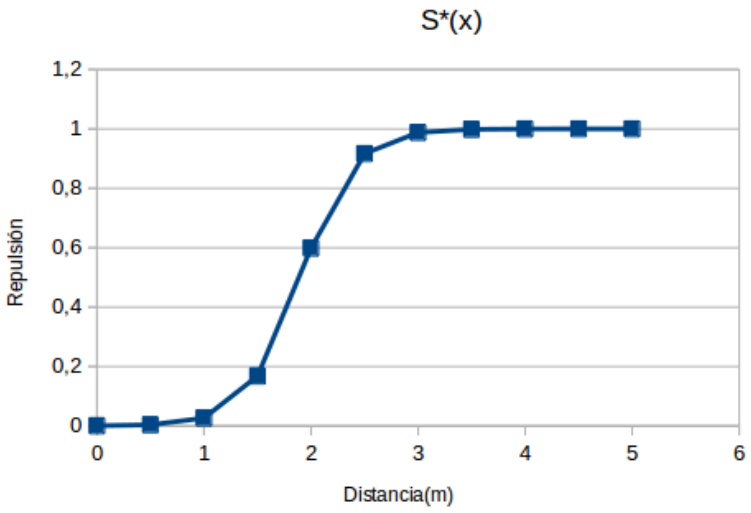

Figura 3.8: Comportamiento de la función sigmoidal.

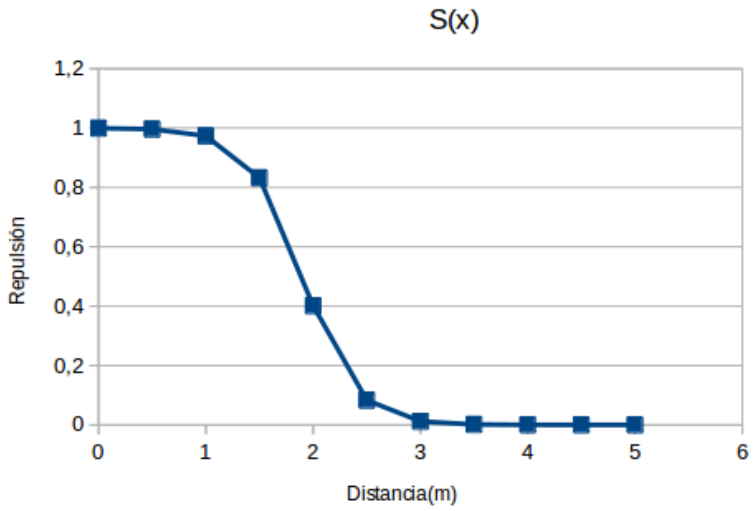

Figura 3.9: Comportamiento de la función sigmoidal utilizada.

Una vez calculada la repulsión que generan los obstáculos sobre el dron, se deben consultar las componentes de la velocidad deseada por el usuario y calcular el vector unitario. Una vez hecho esto, ya se tienen todos los datos para poder calcular la velocidad total, que se obtendrá mediante la ecuación 3.5:

$$\vec{r} = \alpha\,\vec{v} + (1-\alpha)\vec{o} \tag{3.5}$$

donde $\vec{r}$ es la velocidad final, $\vec{v}$ es el vector que marca la velocidad indicada a través del *joystick*, $\vec{o}$ es el vector de repulsión y α es el peso que tendrá el *joystick* sobre la velocidad total: cuanto mayor sea, menos relevante será la repulsión de los obstáculos.

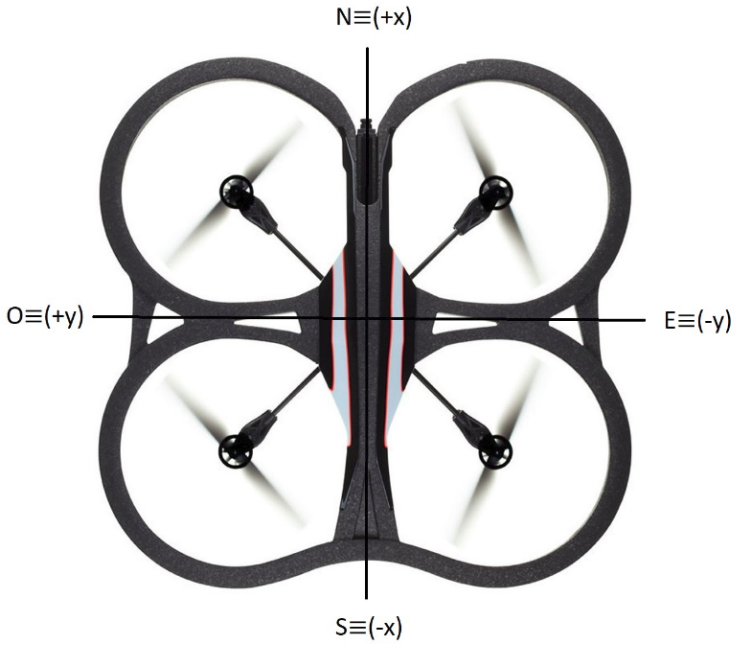

Figura 3.10: Convenio de signos que usa el dron para navegar.

Una vez obtenida la dirección de movimiento, se determina el vector final de movimiento de acuerdo con las ecuaciones 3.6 a 3.8:

$$\theta = \arctan \frac{r_y}{r_x} \tag{3.6}$$

$$S_x = S\cos\theta \tag{3.7}$$

$$S_y = S\sin\theta \tag{3.8}$$

donde S es la velocidad constante a la que se quiere ir, θ es la dirección y (S_x, S_y) son las componentes de velocidad que se transmitirán al dron.

CAPÍTULO 4

RESULTADOS EXPERIMENTALES

4.1 Experimentos preliminares

Antes de comprobar el algoritmo durante un vuelo, se va a comprobar el correcto funcionamiento del algoritmo de evitación de obstáculos de forma teórica y sobre el dron sin volar.

4.1.1 Respuesta de la función sigmoidal

En esta medida experimental se ha desarrollado un script de MATLAB mediante el que se van a representar los vectores de repulsión a los que estaría expuesto el dron en distintos escenarios. Para ello, como se ha comentado anteriormente, se ha creado un script de MATLAB, mediante el que se calcula la función sigmoidal que obtendríamos si el dron estuviera a 1 m, 2m, 3m y 4m de los obstáculos en diferentes escenarios. Una vez calculada esta función, se van a representar las componentes generadas por cada uno de los obstáculos en cada una de las distancias y el vector resultante obtenido de esas componentes.

Escenarios propuestos

Como se ha comentado en el apartado anterior, para llevar a cabo esta prueba se han planteado una serie de escenarios que se muestran en las figuras 4.1, 4.2, 4.3 y 4.4, en las que las líneas rojas representan paredes.

Resultados obtenidos

En la figura 4.5, se observan los vectores de repulsión obtenidos después de aplicar la sigmoidal en el escenario 1 (figura 4.1) a distintas distancias. Al estar expuesto solamente a un obstáculo, se genera sólo un vector de repulsión. Nótese que a medida que la distancia aumenta, el módulo del vector de repulsión disminuye, como era de esperar, ya que el módulo del vector de repulsión está inversamente relacionado con la

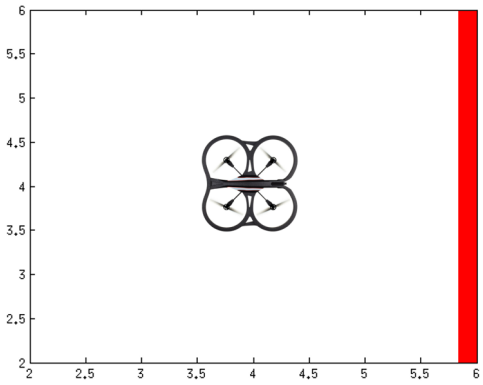

Figura 4.1: Escenario 1: el dron está expuesto a una pared frontal.

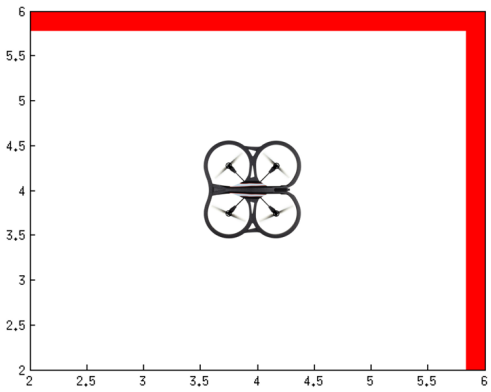

Figura 4.2: Escenario 2: el dron está expuesto a una esquina.

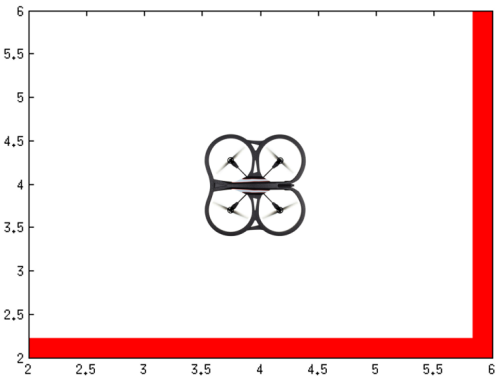

Figura 4.3: Escenario 3: el dron está expuesto a una esquina.

distancia.

En la figura 4.6, se muestra el resultado obtenido después de aplicar la función

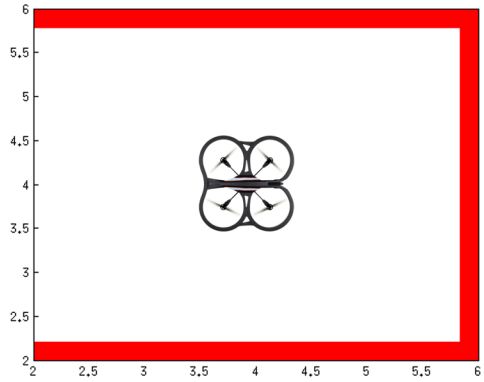

Figura 4.4: Escenario 4: está expuesto a una estructura tipo *callejón sin salida*.

sigmodial en el escenario 2 (figura 4.2) incrementando la distancia. En ellas se puede ver como el vector de repulsión es la resultante de las componentes representadas cómo vectores rojos, que muestran la repulsión que genera cada uno de los obstáculos sobre el dron y, como en el caso anterior, a medida que la distancia aumenta, el módulo del vector de repulsión disminuye.

En la figura 4.7 se pueden ver los resultados obtenidos al realizar las mismas medidas que en apartados anteriores en el escenario 3 (figura 4.3). Los resultados siguen el mismo patrón que las medidas hechas en el escenario 2, es decir, el vector de repulsión obtenido es la resultante de las repulsiones creadas por cada obstáculo y a medida que el dron se aleja de los obstáculos el módulo del vector de repulsión se hace menor.

En la figura 4.8, se representan los vectores de repulsión obtenidos al exponer el dron a una estructura tipo *callejón sin salida* (figura 4.4). Los resultados son vectores de repulsión cuya componente vertical es nula. Esto se debe a que las paredes horizontales se encuentran a la misma distancia del dron, por lo que sus vectores de repulsión se anulan, quedando solamente el vector de repulsión de la pared vertical el cual también decrece en función de la distancia. Los resultados en este caso, también son los esperados.

4.1.2 Comportamiento de los sensores

Para probar el correcto funcionamiento del diseño realizado, se ha planteado un experimento mediante el que se pueda mostrar cómo varía el vector de repulsión en función de la distancia que separa los obstáculos del dron.

Para conseguir esto se ha montado la configuración que se puede ver en la figura 4.9 y se ha movido el dron, manualmente, siguiendo el patrón mostrado en la misma figura.

Para realizar este experimento, se ha deshabilitado el sensor sur para que este no detectara a la persona que iba moviendo el dron, ya que, en caso contrario, se falsearían los datos. Dado que el vehículo no conoce su posición, se ha usado un sistema de

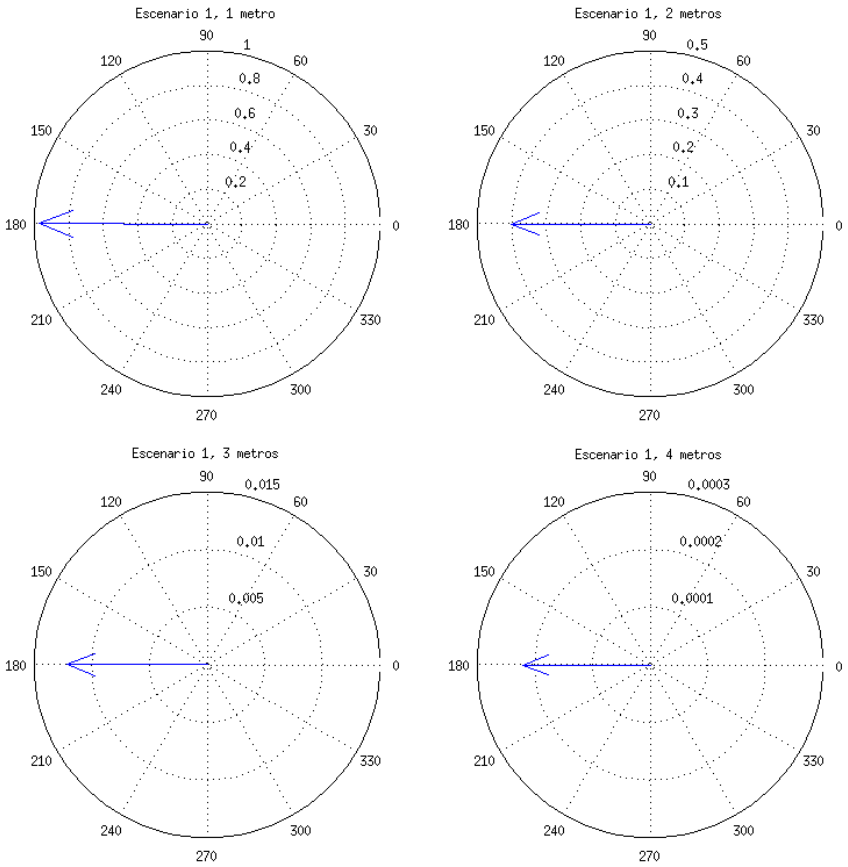

Figura 4.5: Resultado de la función sigmoidal al ser expuesta al escenario 1 modificando la distancia al obstáculo. El vector azul representa el vector de repulsión resultante.

captura de movimiento para conocer la posición del dron. En concreto, se ha usado un sistema *Optitrack* disponible en el laboratorio de investigación de robótica aérea del edificio *Anselm Turmeda* de la *Universitat de les Illes Balears*. Con ello, se han recogido las componentes de repulsión publicadas en tópicos y se han representado en cada uno de esos puntos capturados mediante *optitrack*.

El resultado de este experimento se muestra en la figura 4.10 en la que se puede observar como, a medida que el dron se mueve, los vectores varían su dirección y módulo. Por ejemplo, si observamos la figura 4.11, se ve como a medida que nos alejamos de las paredes, los vectores empiezan a cambiar su dirección. Esto se debe a que el vector de repulsión de una de las paredes está desapareciendo debido a la distancia, mientras que el de la otra se está incrementando.

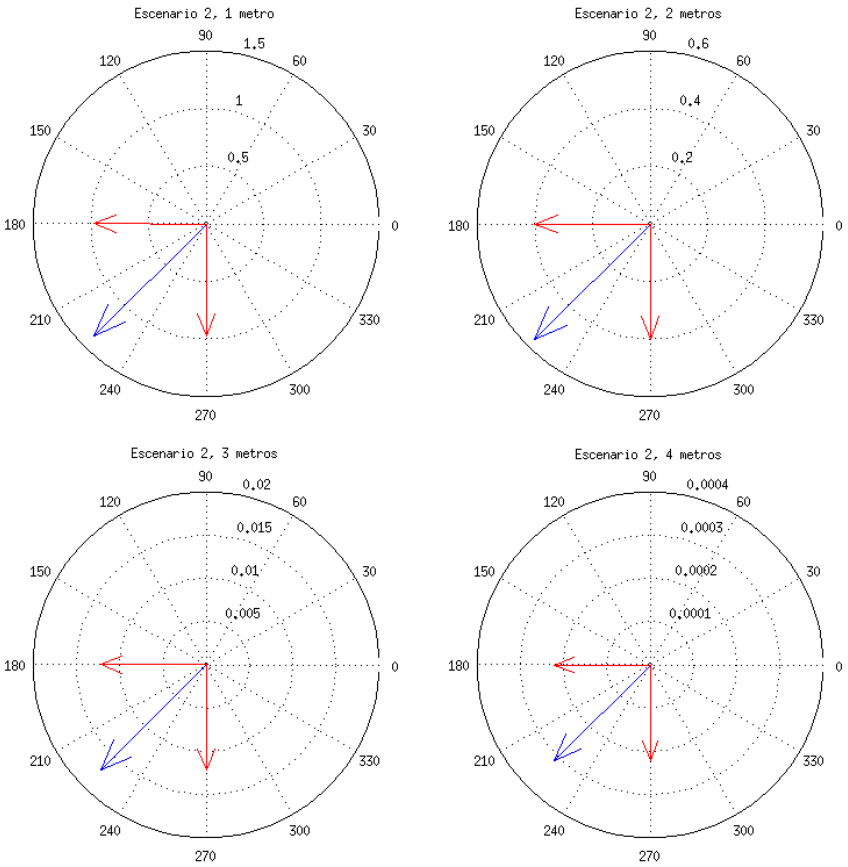

Figura 4.6: Resultado de la función sigmoidal al ser expuesta al escenario 2 modificando la distancia a los obstáculos. El vector azul representa el vector de repulsión resultante para los vectores rojos que se corresponden con la repulsión que crea cada obstáculo.

4.2 Experimentos sobre un simulador

4.2.1 *Gazebo*

En cualquier proyecto relacionado con la robótica, es habitual emplear un simulador antes de probar sobre el robot real. En este proyecto, se decidió usar *Gazebo*, que es una herramienta diseñada para crear y ejecutar simulaciones. Con ella se puede crear tanto el escenario sobre el que se va a ejecutar la prueba como el dispositivo cuyo comportamiento se quiere simular.

Junto con *Gazebo* se ha utilizado el paquete *tum_simulator*. Éste es un paquete de ROS creado para facilitar la simulación de aplicaciones con Ar.Drone en *Gazebo*. Con éste no solamente se pueden hacer vuelos de prueba sino que también se pueden añadir sensores vinculados al Ar.Drone en la posición que el usuario desee.

Al ejecutar el paquete *tum_simulator* con el escenario que se incluye por defecto, obtenemos el entorno que se muestra en la figura 4.12. Al usar este simulador, no sola-

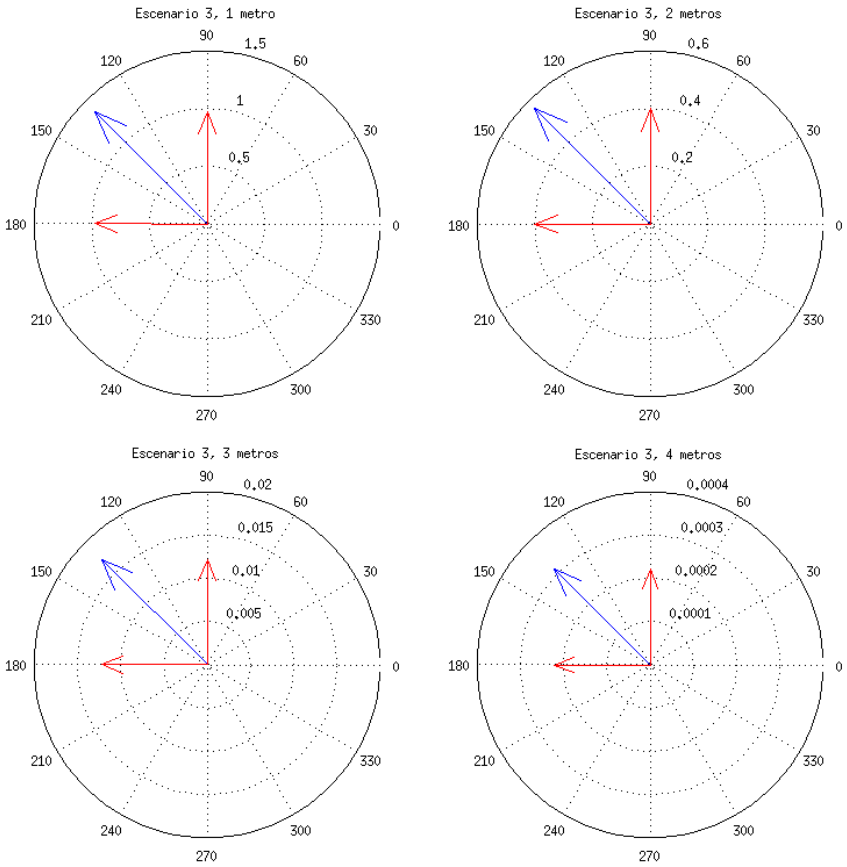

Figura 4.7: Resultado de la función sigmoidal al ser expuesta al escenario 3 modificando la distancia a los obstáculos. El vector azul representa el vector de repulsión resultante para los vectores rojos que se corresponden con la repulsión que crea cada obstáculo.

mente se obtiene un modelo del Ar.Drone, sino que también se crean todos los nodos necesarios para programar el dron; es decir, al lanzar el paquete de simulación, se están creando tópicos en los que publicar las órdenes de despegue, aterrizaje y movimiento que se obtendrían al volar el robot real, por lo que se podrá usar un código pensado para ser ejecutado sobre el dron sin tener la necesidad de incluir ningún cambio.

Objetos creados para la simulación

Para evaluar la estrategia en simulación se han generado diversos escenarios, los cuales se observan en las figuras 4.13, 4.14, 4.15, 4.16 y 4.17. Para emular el comportamiento del módulo real se ha modificado el Ar.Drone del simulador incluyendo cuatro sensores de ultrasonidos, como se puede ver en la figura 4.18.

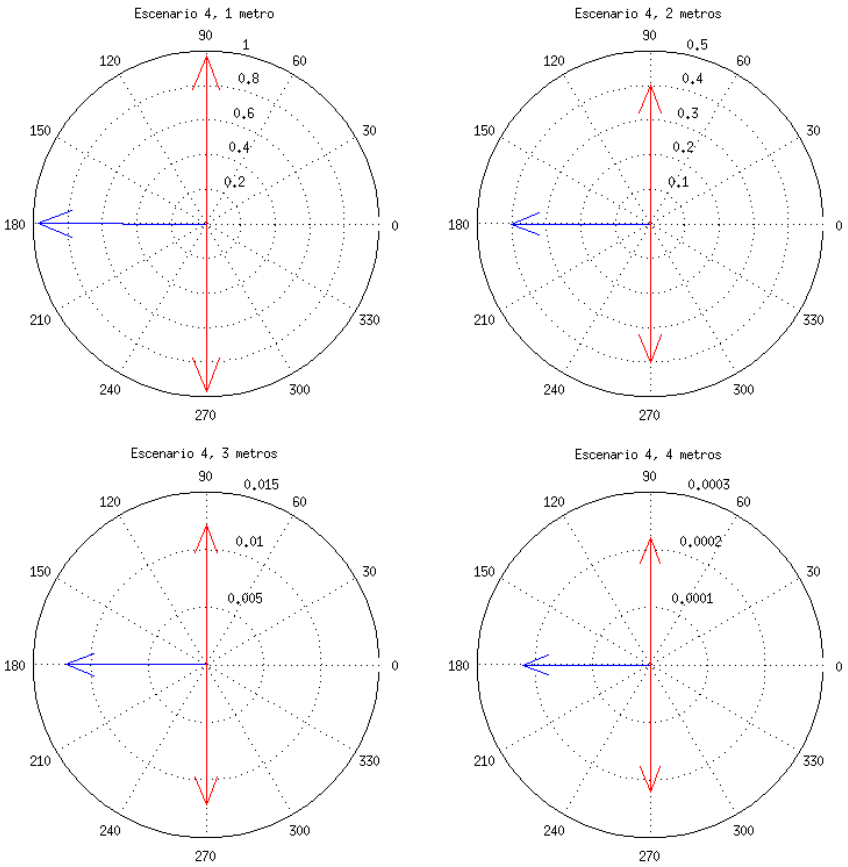

Figura 4.8: Resultado de la función sigmoidal al ser expuesta al escenario 4 modificando la distancia a los obstáculos. El vector azul representa el vector de repulsión resultante para los vectores rojos que representan la repulsión que crea cada obstáculo (el vector de repulsión de la pared vertical coincide con el final).

Resultados obtenidos

En primer lugar, se ha volado el dron hacia los obstáculos creados en los distintos escenarios. Durante estos vuelos, se han almacenado las posiciones del mismo y las de los obstáculos en ficheros para poder comprobar la ruta seguida por el vehículo durante el experimento.

En la figura 4.19, se observa el comportamiento que tendrá el dron al volar hacia el obstáculo del escenario 1 (figura 4.13). Cuando éste se encuentra a una distancia menor que la que establece la sigmoidal, el dron empieza a seguir un patrón de atracción/repulsión. Esto se debe a que, al haber pasado el límite tolerable, el dron se encuentra sometido al campo de repulsión del obstáculo, por lo que va a empezar a alejarse de él. Una vez fuera del límite, este volverá a avanzar hacia el obstáculo hasta que se haya vuelto a acercar demasiado, momento en el cuál el patrón se volverá a repetir. Cuando el dron ha salvado el obstáculo, se puede ver como, al pasar por su lado, el

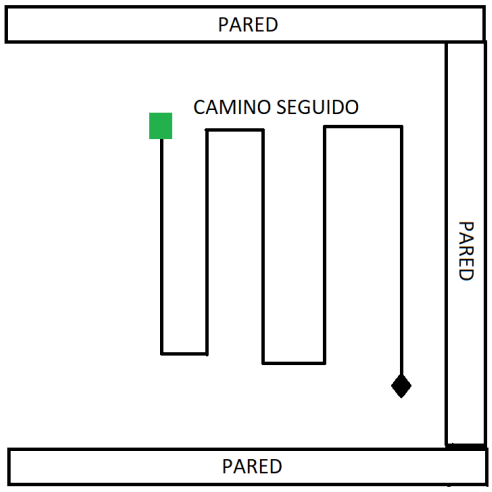

Figura 4.9: Escenario para comprobar el correcto funcionamiento de los sensores instalados sobre el dron.

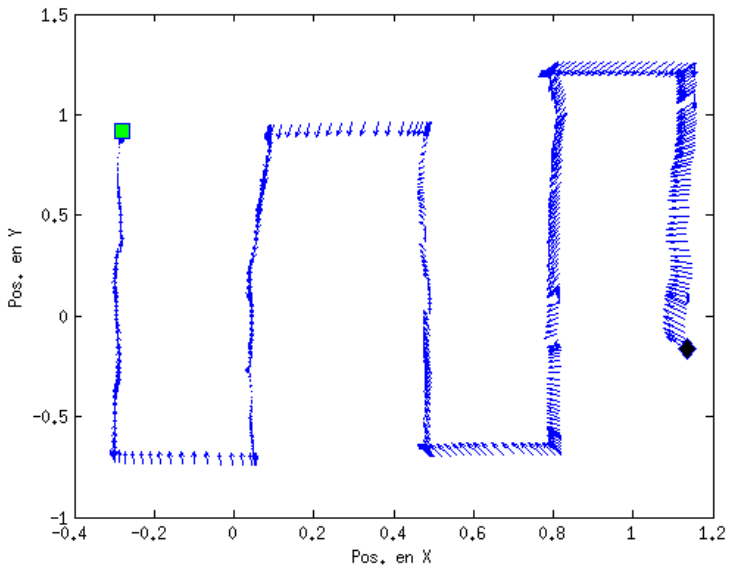

Figura 4.10: Vectores de repulsión resultantes sobre el robot real durante el recorrido de la figura 4.9.

dron también se aleja. Esto es debido a que está a una distancia inferior al límite tolerable y el sensor este ha detectado el obstáculo, por lo que el dron se moverá hacia el oeste.

Si se observa la figura 4.20, se puede ver cómo reacciona el dron al ser llevado entre dos obstáculos (figura 4.14). Como la aproximación no es exáctamente por el centro de los dos obstáculos, detectará uno de ellos antes que el otro, por lo que el dron va a empezar a desplazarse en sentido contrario hasta que el peso del vector de repulsión del otro

Figura 4.11: Ampliación de la línea vertical de vectores situada más a la derecha en la figura 4.10.

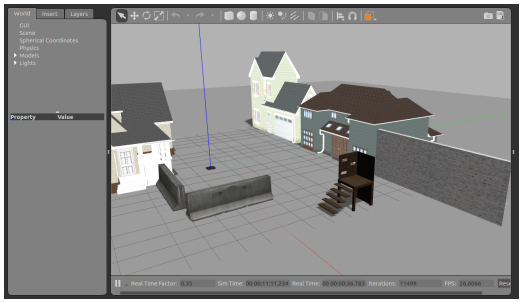

Figura 4.12: escenario que proporciona *tum_simulator* por defecto.

obstáculo sea mayor, momento en el que el dron cambiará el sentido de su movimiento.

Al fijar la atención en la figura 4.21, se aprecia el movimiento del dron al volar en el entorno simulado por el escenario 3 (figura 4.15). Al no estar alineado con el punto inicial del dron, el primer obstáculo no va a representar ninguna amenaza. No obstante, al pasar por su lado, pasa lo mismo que en la simulación del escenario 1: uno de los sensores laterales detecta el obstáculo y al estar a una distancia inferior a la tolerable, se genera un vector de repulsión. Al no detectar más el primer obstáculo, el dron volará siguiendo las instrucciones del usuario. No obstante, cuando detecte el siguiente obstáculo, volverá a ejecutar la maniobra comentada anteriormente.

En la figura 4.22, se observa el comportamiento del dron al volar en el escenario

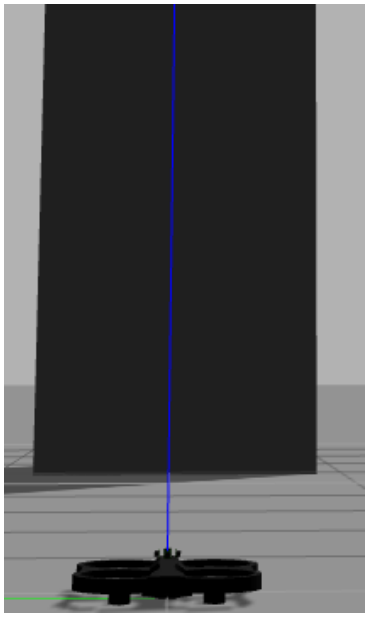

Figura 4.13: Escenario simulado 1, el cual consta de un único obstáculo al frente.

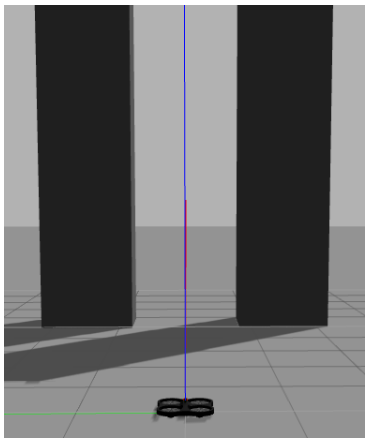

Figura 4.14: Escenario simulado 2, el cual consta de dos obstáculo enfrentados.

que se muestra en la figura 4.16. Para ello, mediante el *joystick*, se envía una velocidad que haga que el dron vuele siempre hacia la misma pared, sin cambiar nunca. Con ello se consigue que el dron vaya cambiando de sentido a medida que se acerca a una pared, ya que como se ha dicho previamente, el vector resultante de repulsión va cambiando a medida que variamos las distancias de los obstáculos.

Al observar la figura 4.23, se puede ver cómo reacciona el algoritmo desarrollado al volar en una zona sin salida (véase el escenario 5, figura 4.17). Como se puede ver en la figura, el dron al moverse por ese entorno, va corrigiendo su velocidad con el fin de alejarse siempre de la pared más próxima.

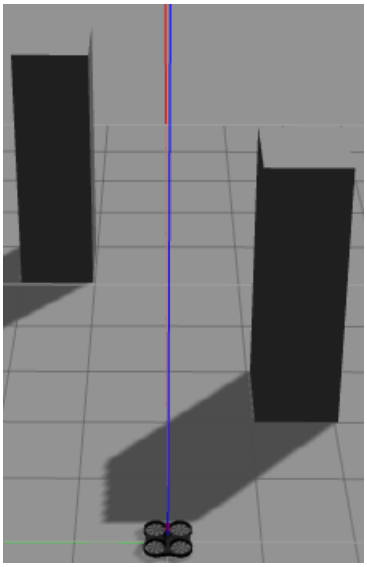

Figura 4.15: Escenario simulado 3, el cual consta de dos obstáculos no enfrentados.

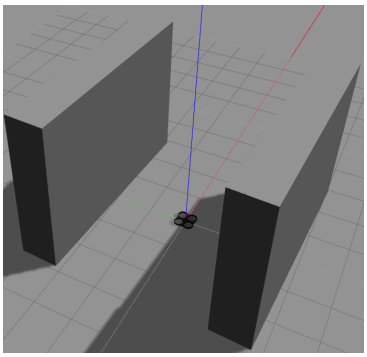

Figura 4.16: Escenario simulado 4, el cual consta de dos obstáculos y el vehículo navega entre los dos.

En la segunda parte de esta sección, se pretende observar la diferencia que hay entre el vector de velocidad recibido a través del *joystick* y el vector enviado al dron, después de aplicar la estrategia de campos de potencial. Para cada uno de los vuelos realizados, se ha obtenido dos gráficos de salida que representan la velocidad obtenida del *joystick* y la que se envía al dron en función del tiempo. Los resultados obtenidos se pueden observar en las figuras 4.24, 4.25, 4.26, 4.27 y 4.28. Debido a que la velocidad a la que se ha establecido que se mueva el dron es superior a la que publica el *joystick*, la representación en la que se pueden ver las velocidades enviadas al dron muestra valores levemente superiores en los casos en los que el dron hace lo que el operario manda.

Si se presta especial atención a la gráfica que representa las componentes x de las velocidades de la figura 4.24, se puede deducir en qué momento se empieza a detectar el único obstáculo del escenario 1 figura 4.13 (recuadro negro), ya que la velocidad que

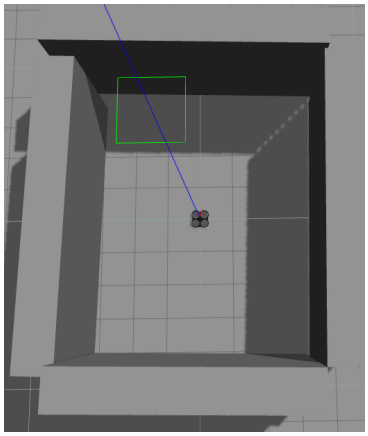

Figura 4.17: En el escenario 5 se intenta emular el comportamiento del dron en una área completamente cerrada.

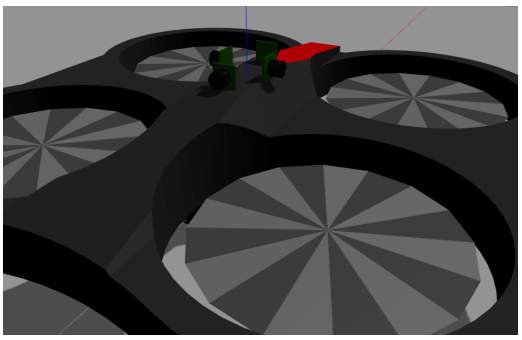

Figura 4.18: Simulación del módulo de ultrasonidos en el *Ar.drone* en *Gazebo*.

se envía al *Ar.Drone* empieza a oscilar para mantener el vehículo alejado del obstáculo; es decir, cuando se entra en el límite de distancia tolerable, se envía una velocidad cuyo sentido es contrario al deseado por el usuario. Al volver a cruzar ese límite, el vector de repulsión deja de actuar, por lo que el dron vuelve a avanzar siguiendo el sentido deseado, hasta llegar al límite citado previamente. Si se observa la figura 4.19, se puede ver cómo responde el dron a estos cambios de velocidad. Este efecto se produce hasta que el dron deja de detectar el obstáculo frontal. Si se observa la gráfica que representa las velocidades en y de la figura 4.24, se puede ver que la velocidad en y, en este caso, tiende a seguir las órdenes que el usuario genera mediante el *joystick*, exceptuando algunos momentos, como cuando el operario está generando una velocidad en y para pasar el obstáculo, pero como el dron ha salido de la zona segura, el ángulo del vector a enviar está cambiando, por lo que la componente y tambien sufre oscilaciones. No obstante, cuando se pasa el obstáculo, y el dron vuela a su lado, uno de los sensores laterales lo detecta, por lo que se calculará una repulsión para alejarlo de él. Este fenómeno puede ser observado en el recuadro verde de la gráfica que representa las velocidades en y de la figura 4.24, momento en el que la velocidad enviada al dron se incrementa enormemente, mientras que la que proviene del *joystick* decrece.

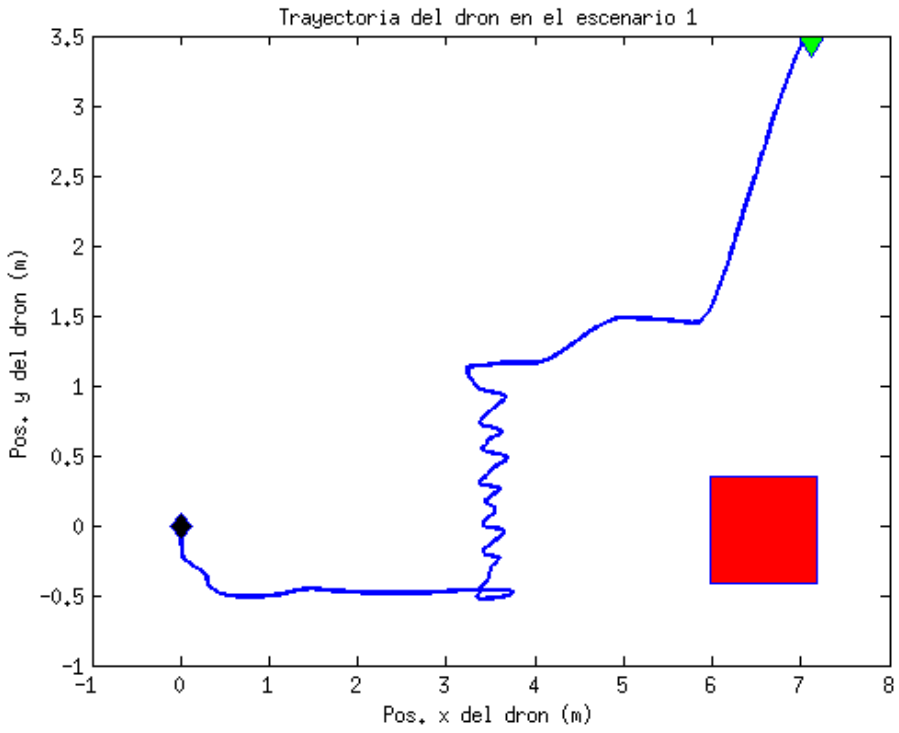

Figura 4.19: Trayectoria al hacer volar el dron en el escenario 1 (figura 4.13), donde el objeto rojo es el obstáculo, la línea azul es el recorrido realizado por el dron, el rombo negro es el punto inicial y el triángulo verde es el punto final.

Si se observa la figura 4.25, se puede ver el comportamiento que tienen los vectores de velocidad cuando se ejecuta la simulación en el escenario 2 (figura 4.14). Si se fija la atención en la gráfica que representa la componente x, se puede ver que la mayor parte del tiempo imita la velocidad marcada por el *joystick*, por lo que seguirá avanzando sin ningún problema. Esto cambia cuando el dron pasa entre los dos obstáculos construidos en este escenario, los cuales generan vectores de repulsión en y, por lo que como se ha explicado previamente, las componentes de la velocidad a enviar van a sufrir fluctuaciones. En el caso de la componente y, la velocidad del dron, imita a la deseada por el usuario. Es a partir del instante marcado por el recuadro negro (momento en el cual se empiezan a cruzar los obstáculos), que el vector de repulsión empieza a hacer efecto. Nótese que el dron va haciendo un efecto rebote. Esto es debido a que al ponerse en medio de los dos obstáculos por primera vez, uno de los obstáculos genera una fuerza de repulsión más grande que el otro, ya que es prácticamente imposible hacer que el dron entre justo por la mitad, por lo que el vehículo va a empezar a moverse evitando ese obstáculo hasta que el otro genere una fuerza de repulsión más fuerte que el segundo, momento en el cual el dron cambiará el sentido de su desplazamiento lateral.

Si se centra la atención en la figura 4.26, se ve la respuesta del algoritmo para el escenario 3 (figura 4.15). Si se observa la gráfica que representa la velocidad en y, se

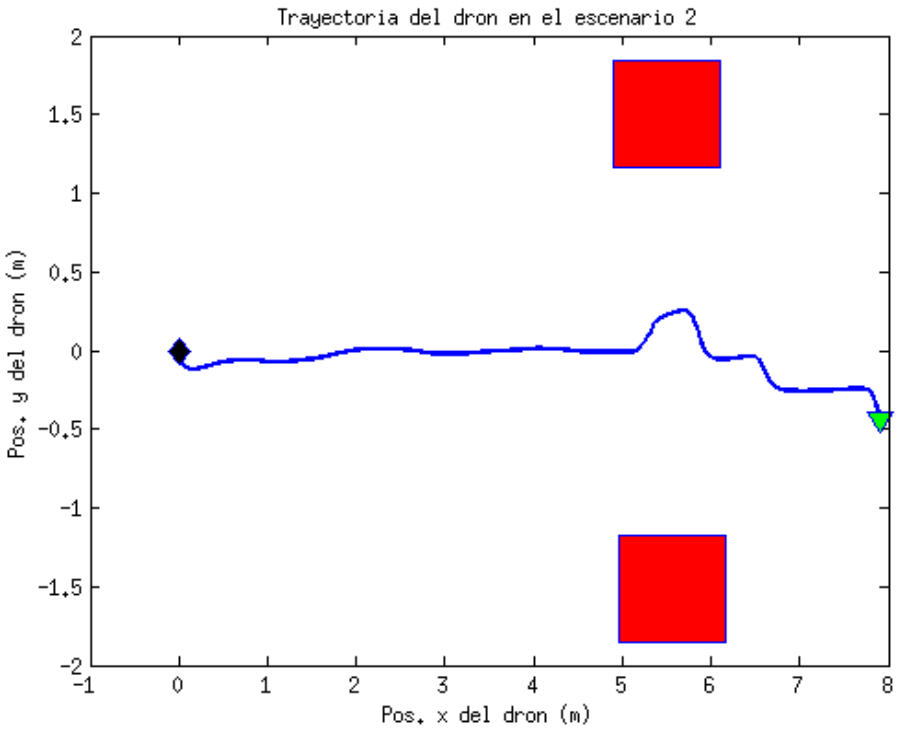

Figura 4.20: Recorrido realizado por el dron al volar en el escenario 2 (figura 4.14), donde los objetos rojos son los obstáculos, la línea azul es el recorrido realizado por el dron, el rombo negro es el punto inicial y el triángulo verde es el punto final.

puede ver que, al igual que en el caso del escenario 1, al pasar junto a un obstáculo, cuando los sensores laterales lo detectan, se crea una repulsión para evitar ese obstáculo (véase el instante marcado por el recuadro negro). A partir del instante marcado por el recuadro verde el dron va a estar expuesto al campo creado por un segundo obstáculo, momento en el que se va a repetir lo acaecido anteriormente: el dron se moverá horizontalmente para mantener la distancia de seguridad establecida.

Al observar la figura 4.27, se puede ver como difiere el vector que se envía al dron del que publica el *joystick*. En este caso, al estar en medio de dos paredes paralelas (véase figura 4.16), mediante el *joystick*, se va a marcar un vuelo para intentar provocar una colisión contra una de las paredes. Es por ello que las representaciones gráficas que marcan la velocidad enviada al dron siguen ese patrón de atracción/repulsión citado previamente. A medida que nos acercamos a una pared, la repulsión de la otra empieza a perder fuerza por lo que la velocidad del dron también cambia.

La figura 4.28 muestra tanto las velocidades que toma el dron como las que el operario envía a través del *joystick* en el escenario 5. Si se observan las componentes por separado, al igual que en los párrafos previos, se puede saber en qué posición está el dron. Por ejemplo, si se observa la velocidad que se envía al dron en y en el instante marcado por el recuadro negro, se puede decir que el dron está pasando cerca de una

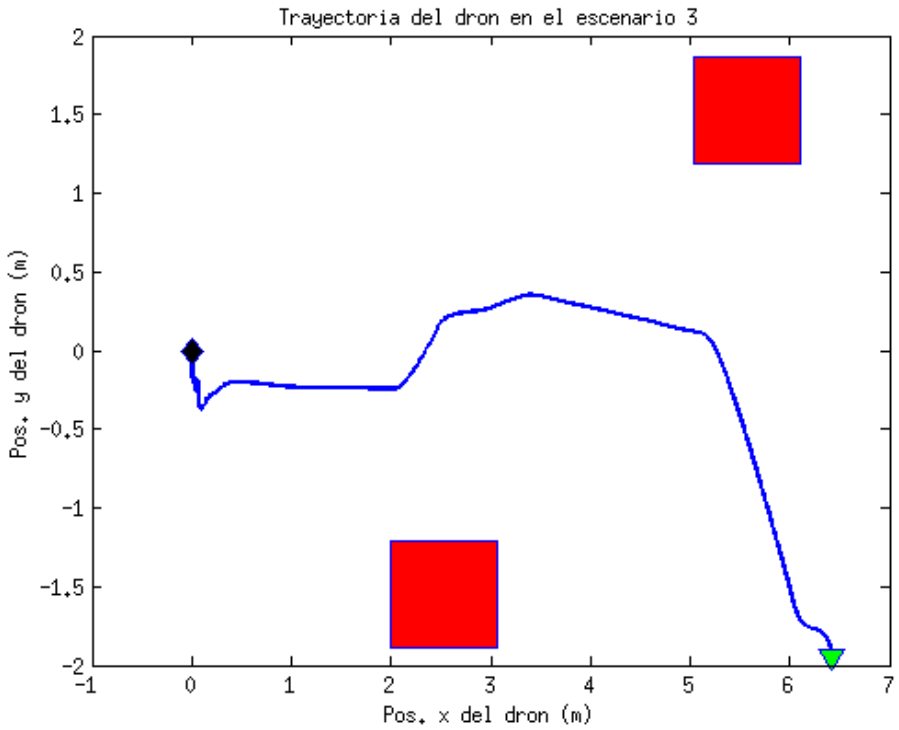

Figura 4.21: Camino marcado por el algoritmo al volar el dron en el escenario (figura 4.15), donde los objetos rojos son los obstáculos, la línea azul es el recorrido realizado por el dron, el rombo negro es el punto inicial y el triángulo verde es el punto final.

pared con el sensor oeste apuntando hacia ella, ya que la velocidad es negativa. Si ahora se fija la atención en la velocidad x, se puede ver justo el momento en el que la repulsión de la pared mencionada ha dejado de hacer efecto, pero está entrando otro obstáculo que está puesto frente el sensor norte (véase instante resaltado mediante un recuadro verde) en el que la velocidad en x cae a -0.5 haciendo que el dron se aleje de la pared, evitando así el peligro. En el recuadro rojo, se puede ver como el dron se ve sometido al patrón de atracción/repulsión que se ha descrito anteriormente en y, por lo que el dron está pasando al lado de una pared con su sensor este apuntando hacia ella, ya que las velocidades de repulsión son positivas.

4.3 Experimentos en un entorno real

Una vez se ha visto que los sensores y algoritmos escritos funcionan correctamente, se va a proceder a volar el dron usando el sistema propuesto. Para ello, como medida de seguridad, ya que hay situaciones que no se pueden prever ni con el simulador, se ha adaptado el nodo para que el algoritmo desarrollado solamente entre en acción en el caso de que se pulse uno de los botones disponibles en el *joystick*, por lo que en caso de observar comportamientos no deseados, simplemente se debe soltar el botón

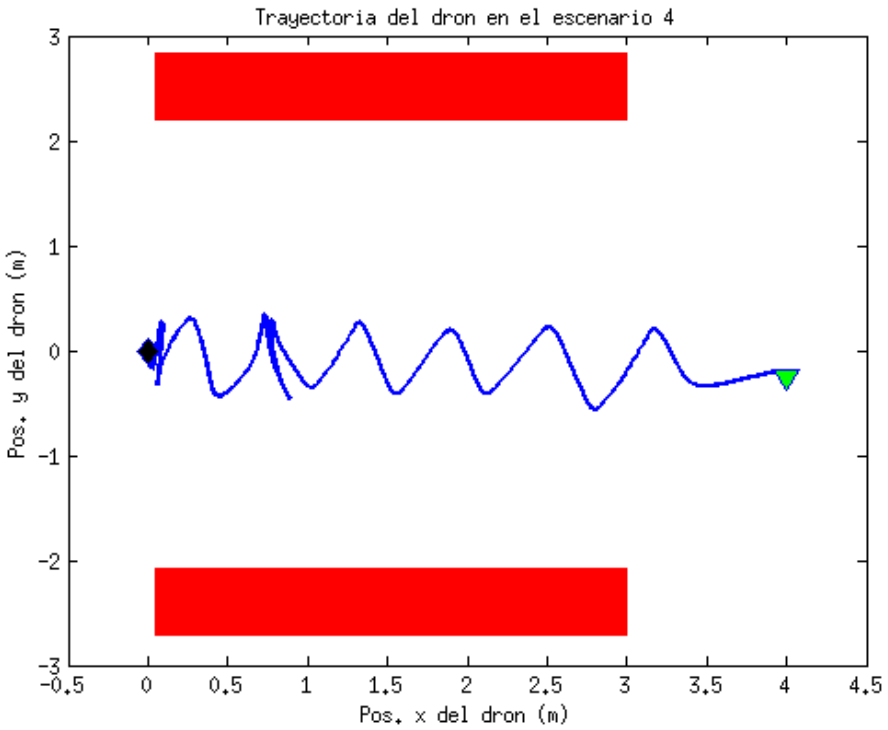

Figura 4.22: Trayectoria tomada por el dron al ser pilotado en el escenario 4 (figura 4.16), donde los objetos rojos son los obstáculos, la línea azul es el recorrido realizado por el dron, el rombo negro es el punto inicial y el triángulo verde es el punto final.

y controlar el dron mediante el joystick. Para realizar esta prueba, se han movido los parámetros de la sigmoidal, a y c, a 10 y 0.75, respectivamente para obtener repulsión entre 0 y 1.5 para adaptarse a las dimensiones del laboratorio.

Para comprobar el correcto comportamiento del diseño final, se ha decidido hacer volar el dron hacia distintos obstáculos montados en el laboratorio de investigación de robótica aérea comentado anteriormente. En este caso se va a seguir tomando las posiciones del dron mediante el sistema *optitrack*. Tras ser procesadas, se ha obtenido las gráficas representadas en las figuras 4.29, 4.30 y 4.31, además de la velocidad marcada por el *joystick* y la enviada finalmente al dron. Al ser un dron de gama baja, la capacidad de vuelo de éste se ha visto muy reducida al añadir cierto peso, por lo que en ciertas ocasiones en los gráficos se pueden ver movimientos extraños producidos por el dron, los cuáles se deben a las correcciones que debe llevar a cabo el usuario para mantener el rumbo.

El primer escenario que se ha propuesto ha sido hacer volar el Ar.Drone hacia una esquina para ver la repulsión que sufre éste en éste entorno. Si se observa la figura 4.29, se puede ver como el dron sufre una repulsión al aproximarse a la esquina. Aunque el comportamiento es similar al esperado, se puede ver como el dron se acerca a las paredes más de lo predispuesto (0.5 m). Esto se debe a que tanto las lecturas de los

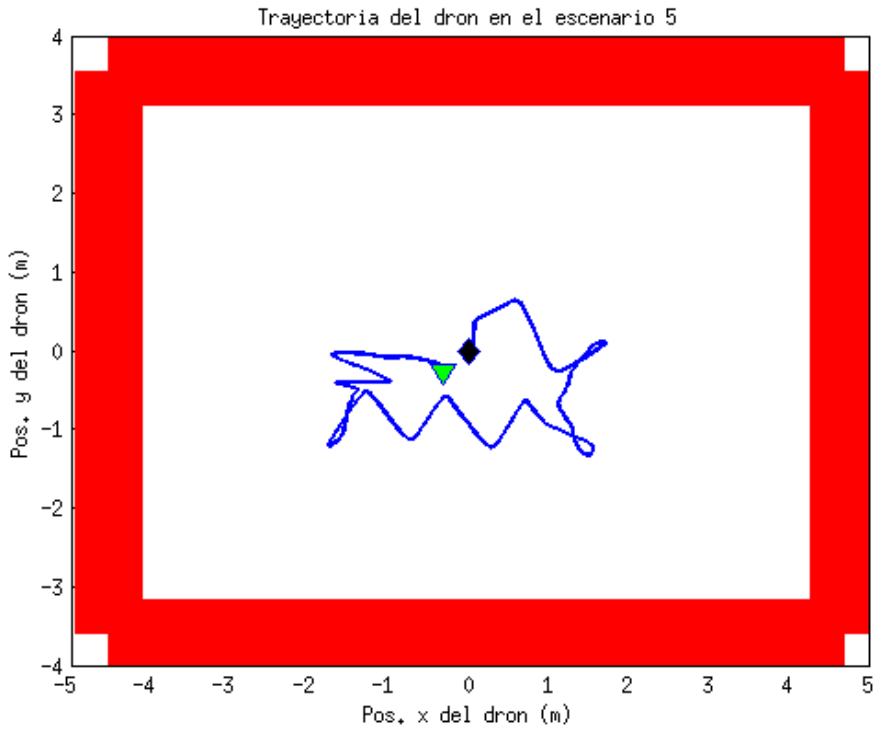

Figura 4.23: Camino descrito por el robot al ser expuesto al entorno simulado en el escenario 5 (figura 4.16), donde los objetos rojos son los obstáculos, la línea azul es el recorrido realizado por el dron, el rombo negro es el punto inicial y el triángulo verde es el punto final.

sensores como las comunicaciones establecidas suponen un retraso, por lo que la repulsión se ejecuta también con cierto retraso.

Para seguir probando el comportamiento del dron, se ha dispuesto un obstáculo en el laboratorio mediante cajas de cartón y se ha hecho volar el dron hacia ese obstáculo. El resultado de dicha prueba se puede ver en la figura 4.30, en la que se puede observar como al llegar a cierta distancia (algo menos de 0.5 m), el dron empieza a experimentar una repulsión que lo aleja del obstáculo.

Para finalizar, se ha hecho pasar el dron entre dos paredes intentando colisionar con ellas. Una vez hecho el experimento, se ha obtenido la figura 4.31 en la que se puede observar como, cada vez que se acerca a una pared, el dron cambia de dirección. Nótese que este es el caso en el que el dron se acerca más a los obstáculos debido al *delay*, el cual ocasiona que el cambio de sentido del vector de velocidad se produzca más lentamente, ocasionando que el dron siga expuesto al vector de repulsión durante más tiempo.

A continuación, como en el caso de las simulaciones mediante *Gazebo*, se va a comparar la velocidad que se está enviando al dron con la que el usuario genera mediante

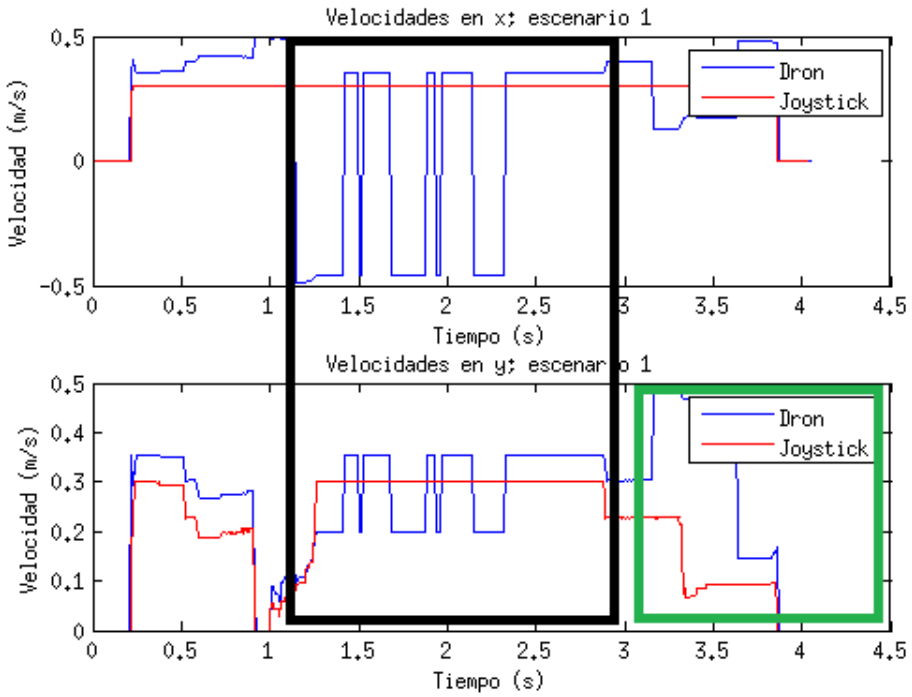

Figura 4.24: Velocidades obtenidas al hacer volar el dron en el escenario 1 (figura 4.13)

el *joystick*.

Si se observa la figura 4.32, se puede ver la velocidad que se está enviando al dron puesta en la misma gráfica que la que el usuario está solicitando cuando el dron está expuesto a la esquina de la pared. En la sección delimitada mediante un rectángulo negro, se puede ver la primera vez que el dron detecta ambas paredes y se empieza a ejercer una pequeña repulsión. En el recuadro verde se observa el momento en el que el dron está más cerca de la pared (véase la figura 4.29) y se ejerce una repulsión durante más tiempo. A continuación, se puede ver como mediante el *joystick*, se corrige el rumbo (acción necesaria debido a la pérdida de capacidad de vuelo a causa del peso añadido) y, en el último recuadro, se puede ver como se termina la repulsión antes de salir del alcance de la pared.

Al fijarnos en la figura 4.33, se observa el punto en el que el dron detecta el obstáculo y empieza su repulsión (recuadro negro).

En la figura 4.34 se puede ver la diferencia entre la velocidad deseada por el usuario y la enviada al dron en el caso de estar volando entre dos paredes. Si se observa el primer recuadro, se puede ver como la orden del joystick es ir hacia una de las paredes, pero el dron al estar expuesto a la repulsión de ésta, cambia el sentido de su movimiento. En el segundo de los recuadros se puede apreciar la segunda repulsión que es en sentido contrario a la anterior ya que ahora el dron está expuesto a la pared que se encuentra enfrentada a la primera.

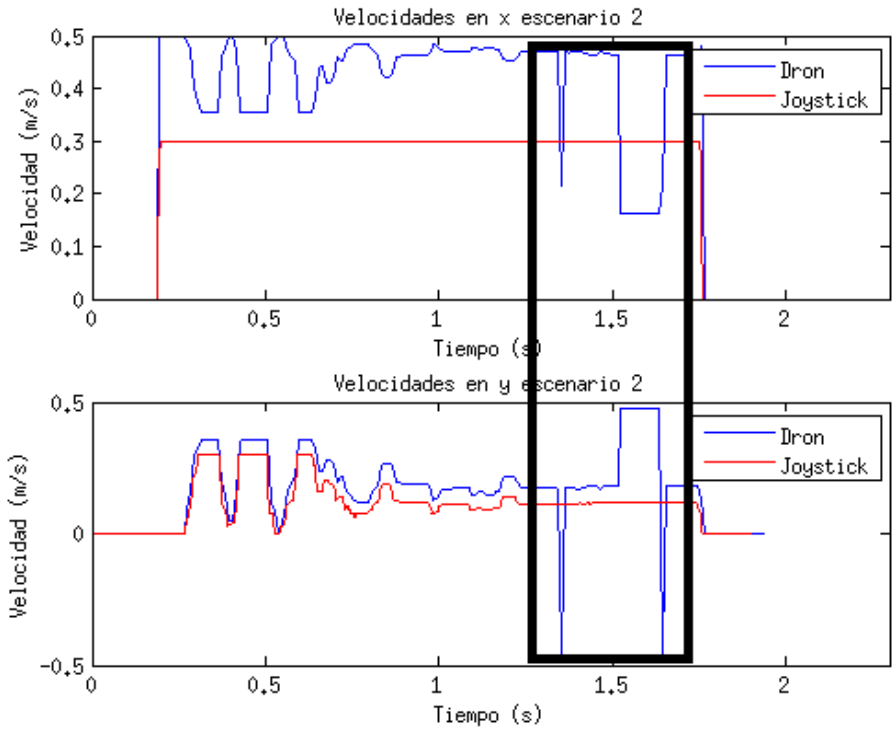

Figura 4.25: Velocidades obtenidas al hacer volar el dron en el escenario 2 (figura 4.14)

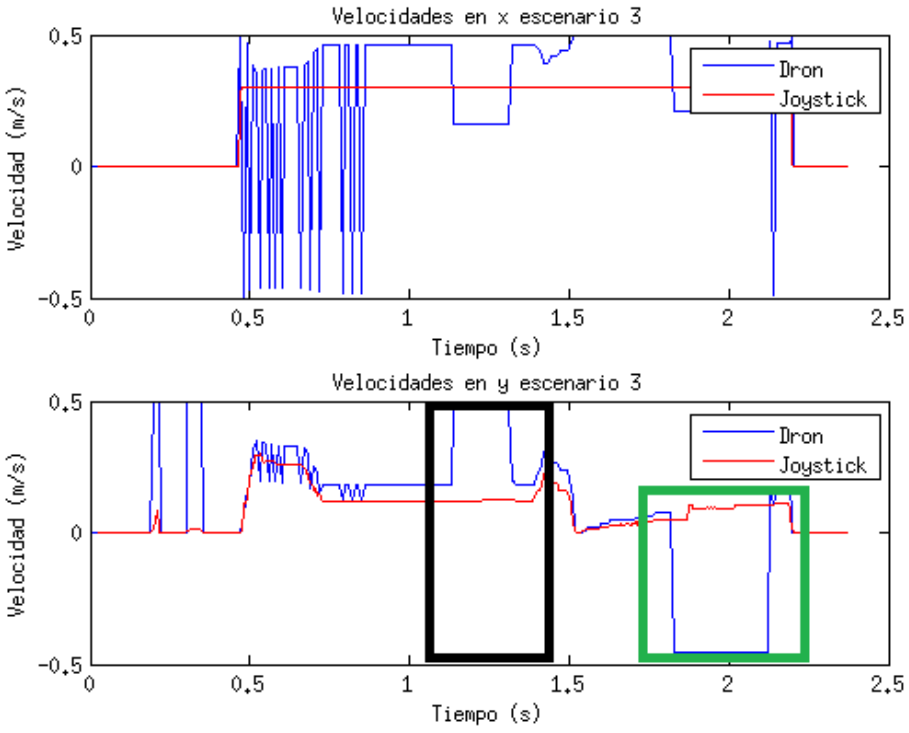

Figura 4.26: Velocidades obtenidas al hacer volar el dron en el escenario 3 (figura 4.15)

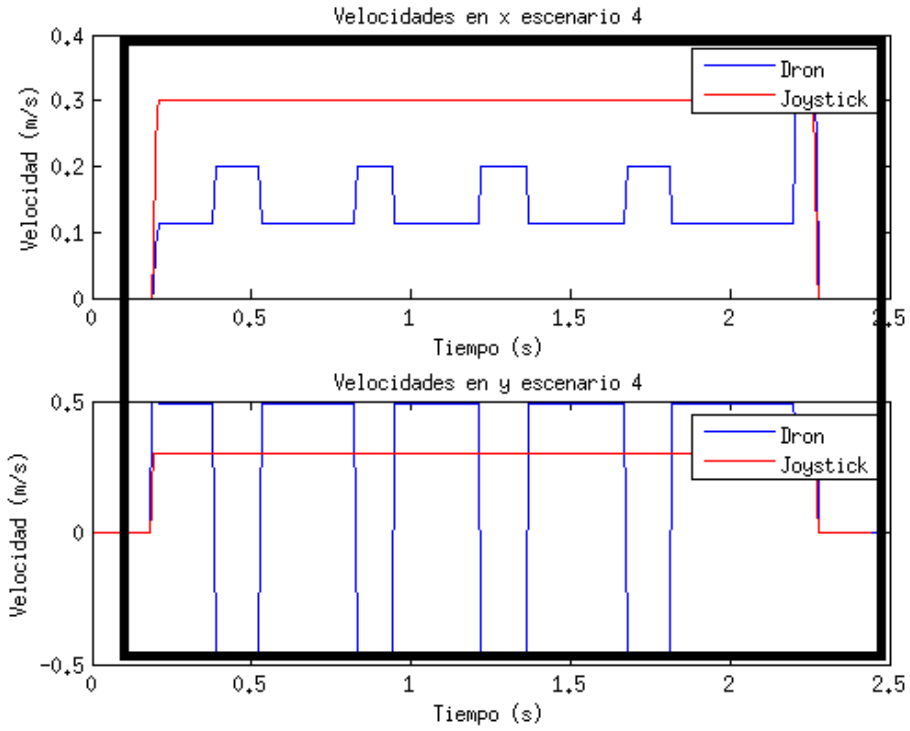

Figura 4.27: Velocidades obtenidas al hacer volar el dron en el escenario 4 (figura 4.16)

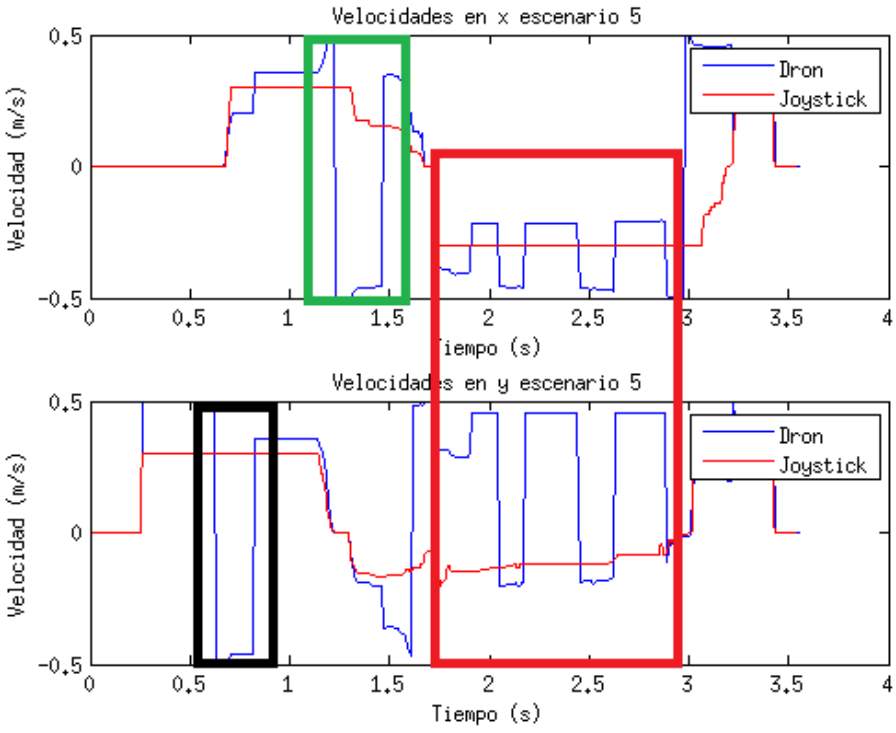

Figura 4.28: Velocidades obtenidas al hacer volar el dron en el escenario 5 (figura 4.17)

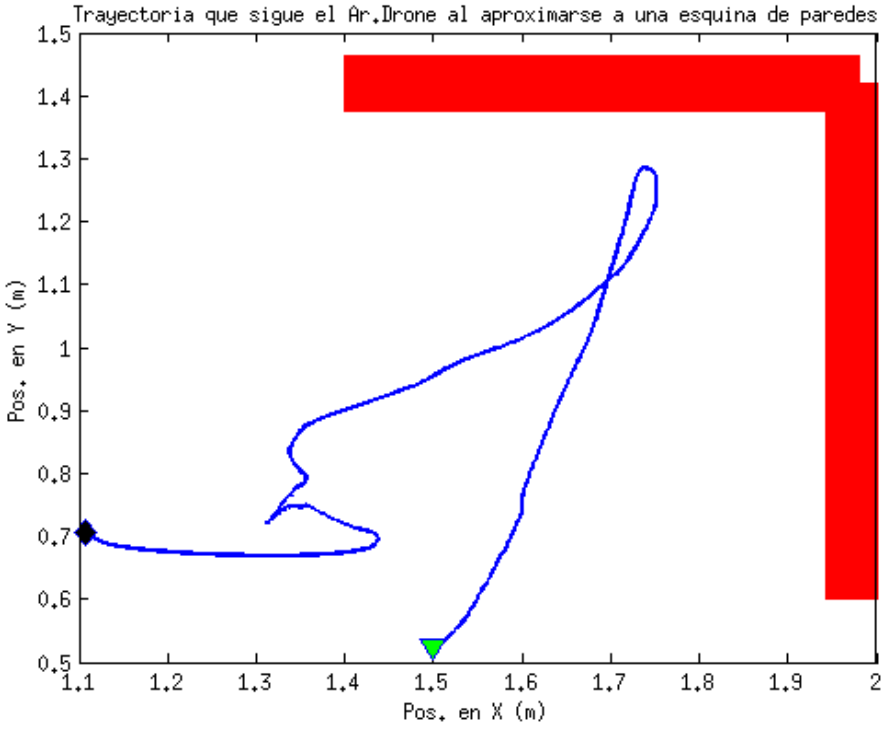

Figura 4.29: Comportamiento del dron al ser aproximado a la esquina formada por dos paredes. En la figura, los elementos rojos son las paredes, el trazo azul la dirección del dron, el rombo negro el punto inicial y el triángulo verde el final.

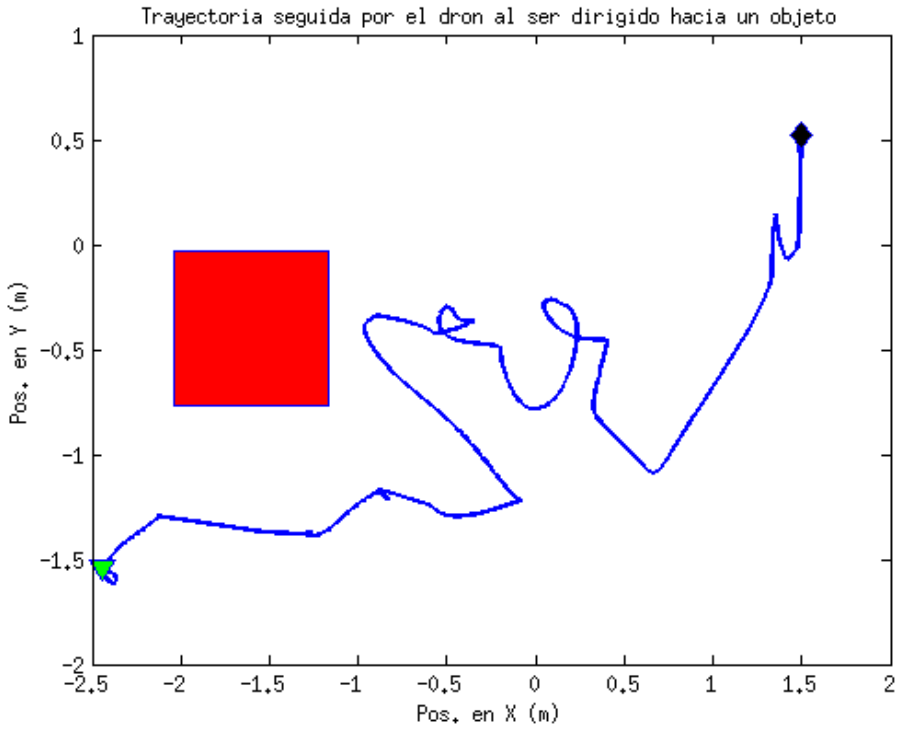

Figura 4.30: (Arriba) Comportamiento del dron al ser aproximado a un obstáculo de frente. En la figura, los elementos rojos son las paredes, el trazo azul la dirección del dron, el rombo negro el punto inicial y el triángulo verde el final. (Abajo) Escenario utilizado para el experimento.

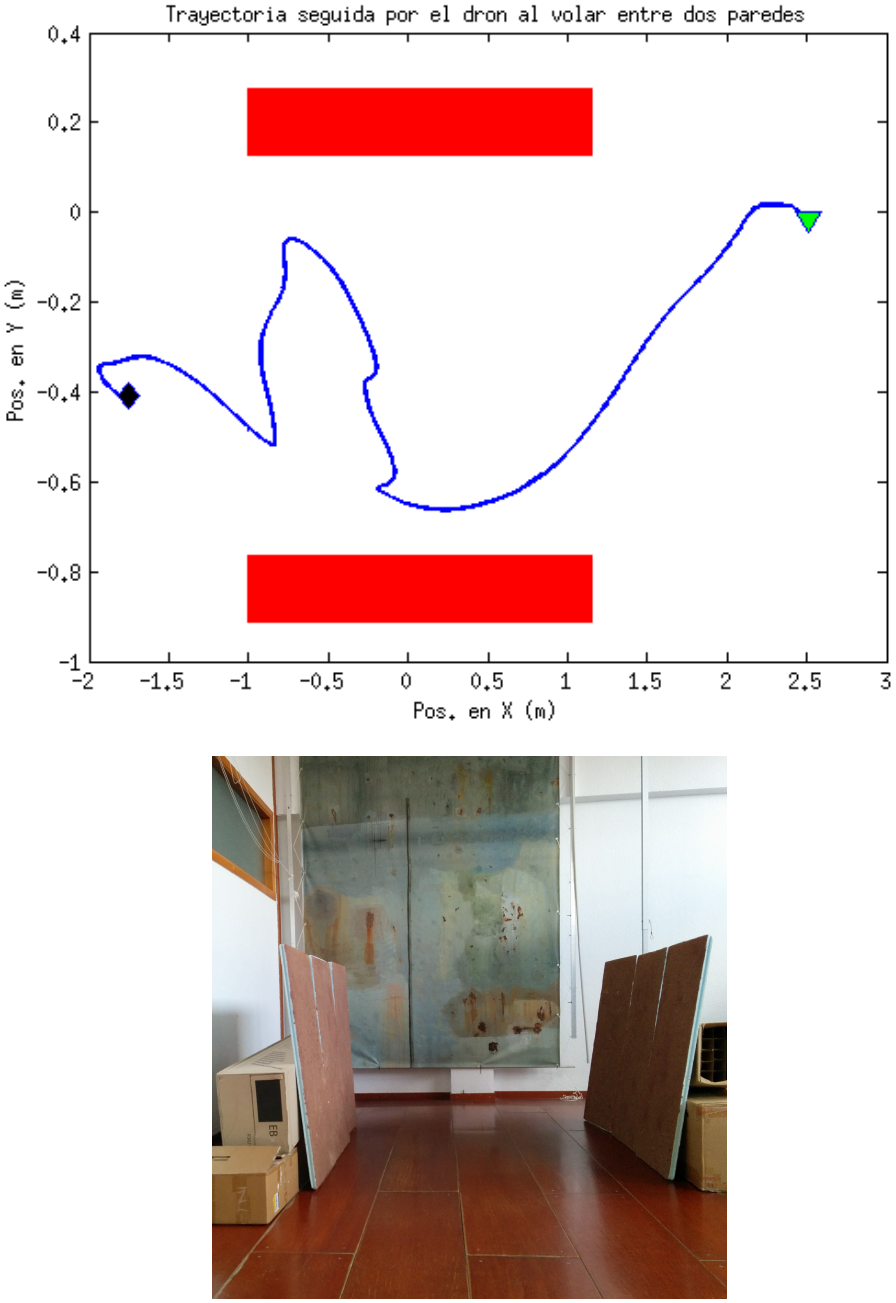

Figura 4.31: (Arriba) Comportamiento del dron al moverse entre dos paredes paralelas. En la figura, los elementos rojos son las paredes, el trazo azul la dirección del dron, el rombo negro el punto inicial y el triángulo verde el final. (Abajo) Escenario utilizado en el experimento

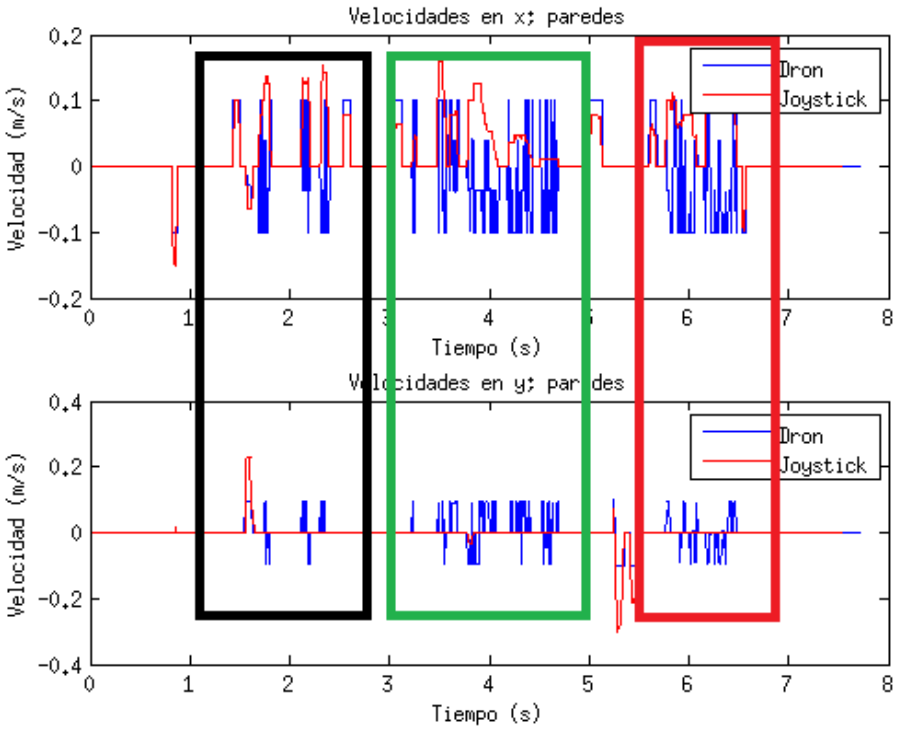

Figura 4.32: Contraposición de la velocidad deseada por el usuario contra la velocidad que realmente ha tomado el dron durante el experimento de la figura 4.29.

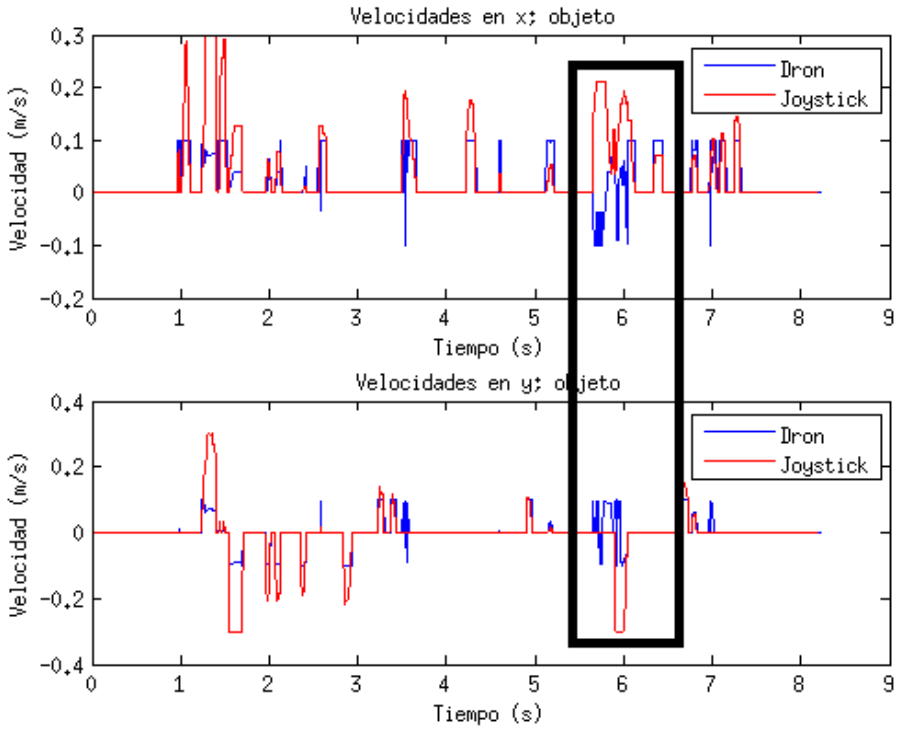

Figura 4.33: Contraposición de la velocidad deseada por el usuario contra la velocidad que realmente ha tomado el dron durante el experimento de la figura 4.30.

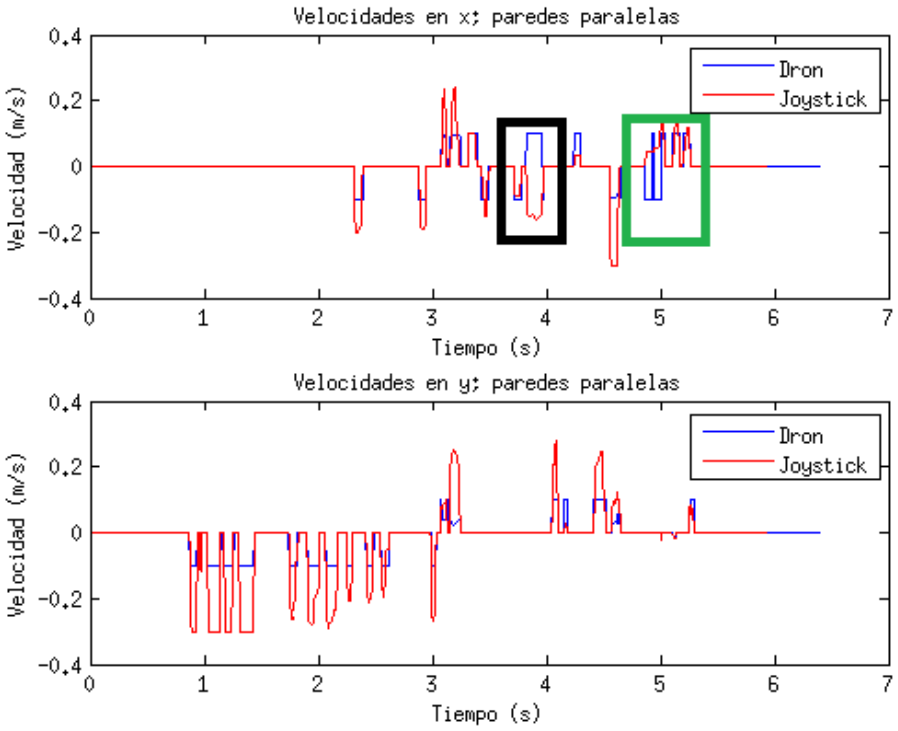

Figura 4.34: Contraposición de la velocidad deseada por el usuario contra la velocidad que realmente ha tomado el dron durante el experimento de la figura 4.31.

CAPÍTULO 5

CONCLUSIONES

5.1 Resumen del proyecto

En este proyecto se ha desarrollado un módulo de detección de obstáculos mediante sensores de ultrasonidos, diseñado para ir instalado sobre una plataforma aérea *Ar.Drone*. Para ello, se ha diseñado un circuito impreso que tiene montado sobre él cuatro sensores de ultrasonidos para obtener medidas del entorno, un microcontrolador basado en Arduino para procesar las mediciones de los sensores y un sistema para establecer comunicación con la estación base mediante radiofrecuencia. La estación base, recoge los datos enviados por el módulo de detección de obstáculos, los procesa mediante una estrategia de evitación de obstáculos y envía comandos de velocidad adecuados al dron. Para ello, mediante ROS (*Robot Operating System*), se ha desarrollado un sistema de nodos con el que se extrae y publica en tópicos la información recibida del dron y se procesa dicha información mediante un algoritmo de evitación de obstáculos, el cuál también obtiene la velocidad deseada por el usuario suscribiéndose al tópico generado por el *joystick*. Con esas dos informaciones (*joystick* y sensores), se determina la velocidad a tomar por el dron, que será transmitida mediante la red wifi creada por el mismo, mediante otro nodo llamado *ardrone_autonomy*.

5.2 Conclusiones finales

Para detectar los obstáculos, se ha optado por usar sensores de ultrasonidos, los cuales para su lectura, exigen un cierto *delay*. Si se usaran los tiempos de retraso recomendados por el fabricante, los sensores escogidos no son una buena opción para detectar obstáculos a velocidad suficiente para evitarlos. No obstante se ha comprobado de forma empírica que estos tiempos se pueden bajar sin afectar las medidas, disminuyendo así el tiempo de detección lo suficiente para que el sensor sea apto para evitar obstáculos.

El módulo de detección de obstáculos creado ha sido montado sobre un Ar.Drone, el cual a pesar de la facilidad que proporciona para variar su rumbo, no tolera un incremento de peso, por lo que no es la opción más adecuada a la hora de expandir sus facultades.

A la hora de transmitir los datos obtenidos mediante el módulo de detección de obstáculos, se ha optado por usar un transmisor por radiofrecuencia, el cual permite conexión a 1km. No obstante, este es el elemento que más peso aporta al dron, lo que como se ha comentado a lo largo de todo este documento, es el mayor problema del prototipo, por lo que en este caso, este transmisor no es el más adecuado. No obstante, el Ar.Drone ha resultado ser capaz de levantar el peso añadido, por lo que aunque no es la mejor, es una solución válida para el caso de este proyecto.

Para evitar los obstáculos se ha usado un algoritmo de campos de potencial, el cual permite aplicar una arquitectura de navegación reactiva, usando solamente información de sensores de distancia, por lo que la capacidad de proceso que requiere es escasa, aunque existen ciertas situaciones en las que el algoritmo no funciona. Esas situaciones son llamadas pozos de potencial. Es por ello que este algoritmo es útil al ser usado en entornos en los que se sabe que no se puede dar ningún pozo de potencial.

5.3 Mejoras futuras

Aunque sea una solución funcional, el sistema propuesto tiene ciertos aspectos negativos sobre los que se podrían aplicar mejoras.

- Aunque se ha intentado maximizar el campo de detección del dron, existen una serie de puntos ciegos. Para disminuirlos, el dron cada cierto tiempo, podría parar su avance y hacer barridos rotando sobre el eje z. De esta forma, si en algún punto hubiera un obstáculo que, mediante su avance normal el dron no lo detectara, al hacer un barrido, éste sería detectado.

- Uno de los principales problemas son las vibraciones a las que se someten los sensores durante los vuelos. Estas vibraciones, se deben al uso de *pinheads*, los cuales no proporcionan una sujeción firme. Si se eliminan estos componentes, se pierde la capacidad de un fácil recambio en caso de avería, pero se consigue que los sensores no tengan juego, por lo que no se verán tan afectados por las vibraciones a las que están sometidos durante el vuelo.

- Para evitar que el usuario tenga que ejercer una velocidad para sobrepasar un obstáculo, se podría aplicar una estrategia para evitar pozos de potencial.

- Para mejorar la navegación del *Ar.Drone*, se podría incluir un sistema que nos permita planificar el camino mediante un mapa y durante la navegación, ceder parte del mando a los sensores de ultrasonidos. Con eso se dotaría el dron de la capacidad de planificar rutas junto con la facultad de responder rápidamente a

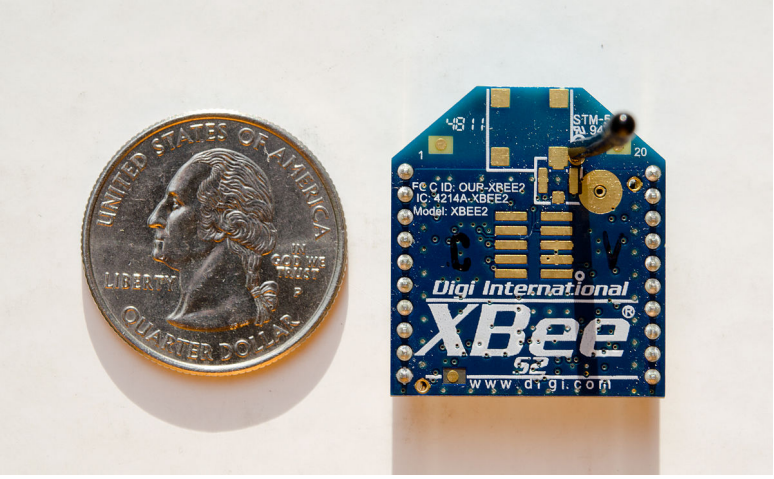

Figura 5.1: Dispositivo con el que se podría sustituir el APC220.

los cambios en el entorno que se puedan producir durante una misión. Es decir, se podría aplicar una arquitectura de navegación híbrida.

- El principal problema en este proyecto es el peso extra añadido al dron. Para tratar de reducirlo, se podría sustituir el APC220 por un Xbee como el que se muestra en la figura 5.1, pudiendo disminuir de 30 a 3g el peso del transmisor de radiofrecuencia.

BIBLIOGRAFÍA

[1] D. Bookstaber, "Unmanned combat aerial vehicles what men do in aircraft and why machines can do it better," Air and Space Power Journal, Tech. Rep., 2000. 1.1.1

[2] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5. 3.2

[3] R. Stelzer, K. Jafarmadar, H. Hassler, and R. Charwot, "A reactive approach to obstacle avoidance in autonomous sailing," *architecture*, vol. 13, p. 16. 3.4.4

[4] R. C. Arkin, "Behavior-based robotics," 1998. 3.4.4

[5] H. Safadi, *Local Path Planning Using Virtual Potential Field*. McGill University School of Computer Science. 3.4.4