



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería Informática
Sistema de Navegación Semiautónomo en
Interiores**



Presentado por Mario Bartolomé Manovel
en Universidad de Burgos — 6 de junio de 2018

Tutores

Dr. Alejandro Merino Gómez
Dr. César Ignacio García Osorio
Dr. José Francisco Díez Pastor

Índice general

Índice general	I
Índice de figuras	III
Índice de tablas	VI
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	2
A.3. Estudio de viabilidad	10
Apéndice B Especificación de Requisitos	21
B.1. Introducción	21
B.2. Objetivos generales	21
B.3. Catálogo de requisitos	21
B.4. Especificación de requisitos	23
Apéndice C Especificación de diseño	33
C.1. Introducción	33
C.2. Diseño de datos	33
C.3. Diseño procedimental	35
C.4. Diseño arquitectónico	38
Apéndice D Documentación técnica de programación	49
D.1. Introducción	49
D.2. Estructura de directorios	49
D.3. Manual del programador	50

D.4. Compilación, instalación y ejecución del proyecto	57
D.5. Pruebas del sistema	63
Apéndice E Documentación de usuario	73
E.1. Introducción	73
E.2. Requisitos de usuarios	73
E.3. Instalación	74
E.4. Manual del usuario	84
E.5. Modo manual	89
Bibliografía	93

Índice de figuras

A.1. Burndown Sprint 1	3
A.2. Burndown Sprint 2	4
A.3. Burndown Sprint 3	4
A.4. Burndown Sprint 4	5
A.5. Burndown Sprint 5	6
A.6. Burndown Sprint 6	6
A.7. Burndown Sprint 7-8	7
A.8. Burndown Sprint 9	8
A.9. Burndown Sprint 11	9
A.10. Burndown Sprint 12	10
 B.1. Diagrama de Casos de Uso	31
 C.1. Diagrama E/R para WebApp	35
C.2. Diagrama Relacional WebApp	36
C.3. Diagrama de Secuencia	37
C.4. Patrón MVC. Imagen de Wikipedia.	39
C.5. Arquitectura del sistema de control de agentes	41
C.6. Estructura de paquetes en agente	42
C.7. Estructura de paquetes en servidor web	43
C.8. Diagrama de clases FrontEnd	45
C.9. Diagrama de clases BackEnd	47
 D.1. PyCharm. Configuración de despliegue	53
D.2. PyCharm. Configuración de intérprete remoto	53
D.3. PyCharm. Configuración de intérprete remoto. SSH	54
D.4. PyCharm. Configuración de intérprete remoto. VirtualEnv	54
D.5. Diagrama de conexión de sensores a RaspberryPi	56

D.6. Clonado de repositorio Git	57
D.7. Ejecución de aplicación web	59
D.8. Ejecución del sistema de control	59
D.9. Ejecución del proyecto desde terminal	60
D.10. Creación de <i>pull-request</i> desde GitKraken	62
D.11. Configuración de SonarCloud	63
D.12. Resultados de SonarCloud	64
D.13. Ejecución de test en intérprete local	65
D.14. Creación de configuración de test de ejecución remota	67
D.15. Configuración de test de ejecución remota	68
D.16. Resultado de cubrimiento de test en intérprete local	69
D.17. Resultado de test en intérprete local	69
D.18. Uso de <i>coverage</i> para recabar métricas de test	71
D.19. Métricas de <i>coverage</i> en SonarCloud	72
E.1. Emisora en modo bootloader	74
E.2. Emisora conectada	75
E.3. Lectura de modelos OpenTX	75
E.4. Creación de modelos OpenTX	76
E.5. Creación de modelos OpenTX. 1	76
E.6. Creación de modelos OpenTX. 2	77
E.7. Creación de modelos OpenTX. 3	77
E.8. Creación de modelos OpenTX. 4	78
E.9. Creación de modelos OpenTX. 5	78
E.10. Creación de modelos OpenTX. 6	79
E.11. Creación de modelos OpenTX. 7	79
E.12. Interruptor SF de la emisora	80
E.13. Creación de modelos OpenTX. 8	80
E.14. Creación de modelos OpenTX. 9	81
E.15. Creación de modelos OpenTX. 10	81
E.16. Escritura de modelos OpenTX.	82
E.17. Acceso a modelos de la emisora.	82
E.18. Acceso a modelo creado en la emisora.	83
E.19. Uso de modelo creado en la emisora.	83
E.20. Inicio de sesión en la aplicación web.	84
E.21. Página principal de control del sistema.	85
E.22. Página principal. Gestión de vídeo.	86
E.23. Página principal. Gestión de vídeo.	87
E.24. Página principal. Gestión de vídeo. Nocturno.	88
E.25. Página principal. Gestión de vídeo. Grabación.	89
E.26. Página principal. Gestión de control manual.	90

Índice de figuras

v

E.27. Página principal. Gestión de control manual. Activación	91
E.28. Herida producida por hélice del <i>drone</i>	92

Índice de tablas

A.1. Costes de personal de desarrollo	10
A.2. Costes totales de asesoría	11
A.3. Costes totales de personal	12
A.4. Costes de hardware	12
A.5. Costes de software	13
A.6. Costes variados	13
A.7. Costes totales del proyecto	13
A.8. Monetización del proyecto.	14
A.9. Gastos totales	15
A.10. Dependencias en <i>backend</i>	17
A.11. Dependencias en <i>frontend</i>	17
A.12. Resumen de licencias.	18
B.1. CU-01. Acceso al sistema	24
B.2. CU-02. Cierre de sesión	25
B.3. CU-03. Gestión del sistema	26
B.4. CU-04. Gestión del streaming de vídeo	27
B.5. CU-05. Grabación de vídeo	27
B.6. CU-06. Control manual	28
B.7. CU-07. Gestión de controles automáticos	29
B.8. CU-08. Gestión del <i>backend</i>	30

Apéndice A

Plan de Proyecto Software

A.1. Introducción

La planificación es, sin duda, una parte fundamental en el ciclo de vida de un proyecto. Con el uso de metodologías ágiles para el seguimiento del proyecto, la planificación inicial se vuelve voluble y cambiante, de forma que la mayor parte de aproximaciones serán precisamente eso, aproximaciones.

En este anexo se detallan los recursos necesarios para el desarrollo del proyecto mediante un estudio de la planificación temporal, y un estudio de la viabilidad del mismo.

El primer apartado, **planificación temporal**, establece un calendario de estimaciones. Cabe destacar que dicho calendario es totalmente flexible. Sí que se ha mantenido la estructura habitual estableciendo una fecha de inicio y una fecha de finalización, y se han tenido en cuenta la duración de las tareas, pero estas son extendibles entre diferentes fases, o *sprints*.

El segundo apartado, **estudio de viabilidad**, detalla la viabilidad del proyecto a través de dos subapartados. El primero detallará la viabilidad económica del mismo, teniendo en cuenta los costes y beneficios previstos. El segundo apartado, relacionado con la viabilidad legal, detallará el contexto legal en el que se regula este proyecto.

A.2. Planificación temporal

Para la realización de este proyecto se ha seguido una metodología ágil basada en Scrum. Dicha metodología aporta flexibilidad, muy necesaria durante el desarrollo software. Si bien está metodología no se ha seguido por completo, ya que el proyecto no ha sido desarrollado por un equipo grande, sí que se ha seguido en esencia la forma que la metodología propone:

- El desarrollo se ha llevado a cabo mediante iteraciones, llamadas *sprints* y revisiones de las mismas.
- La duración de los *sprints* fue de dos semanas. Se consideró hacerlos de una única semana, pero la dificultad del desarrollo de los algoritmos, y sobre todo el amplio estudio que estos requerían, hizo que nos decantásemos por una duración bisemanal.
- Al finalizar cada *sprint* se procedía a realizar una reunión, en la que se comentaba el estado del desarrollo, y se planificaba la siguiente iteración.
- Durante las reuniones de revisión/planificación, se establecían las tareas a realizar en el siguiente *sprint*.
- La monitorización del proyecto se realiza mediante gráficos *burndown*, aunque no son especialmente útiles en fases iniciales, donde el número de *issues* no es alta, y su finalización es muy variable.

Se ha realizado una valoración de la duración de cada *issue* haciendo uso de *story points*, que son una manera de aproximar la duración máxima de las mismas. La estimación temporal dada a los *story points* se ha realizado como una correspondencia a días, es decir, una tarea marcada con 2 *story points* se estima que finalizará al cabo de 2 días de trabajo.

De esta forma se flexibiliza todavía más la metodología, a riesgo de perder precisión. Al no darse una correspondencia absoluta entre los *story points* asignados y la duración, se puede relegar una tarea poco importante para finalizarla en el número de días asignados. Desde luego una tarea sencilla se podrá finalizar antes, y una compleja después, dependiendo de la dedicación, en horas, dedicada a ella.



Figura A.1: Gráfico burndown.

Sprint 1. 06/11/17 - 20/11/17

Este primer *sprint* fue una toma de contacto, y realizado antes del comienzo del semestre, con la intención de tener claro el desarrollo inicial del proyecto. En él se establecieron los requisitos del mismo, y los diferentes algoritmos a implementar. La descomposición de las tareas puede ser consultada en [Sprint 1¹](#)

Como puede verse en la Figura A.1, se relegó el cierre de estas tareas de documentación hasta el final del *sprint*. Se estimaron 4 *story points*, pero se dedicaron al menos 8 días a la búsqueda de información. Dada la carga de trabajo al final del semestre no se pudo dedicar más tiempo al mismo.

Sprint 2. 20/11/17 - 04/12/17

Este segundo *sprint* fue dedicado a continuar con la obtención de información, y realizado antes del comienzo del semestre. El Dr. Alejandro Merino Gómez sugirió la utilización de algoritmos de Campos Potenciales, ver [1], para implementar el sistema de evasión de obstáculos, y proveyó de mucha información relacionada con el mismo. La descomposición de las tareas puede ser consultada en [Sprint 2²](#).

Como puede verse en la Figura A.2, este *sprint* sí que siguió con mayor precisión la forma ideal del gráfico *burndown*. Se estimaron 10 *story points*, pero se dedicaron 9 días a la búsqueda de información. Dada la carga de trabajo al final del semestre no se pudo dedicar más tiempo al mismo.

¹https://github.com/mbm0089/gii_0_17.02_snsi/milestone/1?closed=1

²https://github.com/mbm0089/gii_0_17.02_snsi/milestone/2?closed=1



Figura A.2: Gráfico burndown.



Figura A.3: Gráfico burndown.

Sprint 3. 4/12/17 - 15/01/18

Este tercer *sprint*, más largo por la existencia de las vacaciones de Navidad, fue dedicado a continuar con la obtención de información, y realizado antes del comienzo del semestre. El Dr. Alejandro Merino Gómez sugirió la búsqueda de algún sistema operativo de tiempo real, en previsión de que el sistema operativo de la RaspberryPi no fuera lo suficientemente rápido en sus cambios de contexto. El Dr. José Francisco Díez Pastor proporcionó información sobre búsqueda de rutas. La descomposición de las tareas puede ser consultada en [Sprint 3³](#).

Como puede verse en la Figura A.3, este *sprint* no siguió la forma ideal del gráfico *burndown*, dadas las pocas tareas creadas. Se estimaron 2 *story points*, pero se dedicaron al menos 10 días a la búsqueda de información. Dada la carga de trabajo al final del semestre, y el periodo vacacional, no se

³https://github.com/mbm0089/gii_0_17.02_snsi/milestone/3?closed=1



Figura A.4: Gráfico burndown.

pudo dedicar más tiempo al mismo.

Sprint 4. 17/01/18 - 31/01/18

Este cuarto *sprint*, fue dedicado a conseguir la comunicación entre la RaspberryPi y el drone. Se creó la primera implementación del algoritmo de Campos Potenciales, posteriormente sustituido por *VFH*. En este caso, durante la preparación del *sprint*, expliqué a los tutores las diferentes formas de comunicación que se podían valorar. La descomposición de las tareas puede ser consultada en [Sprint 4⁴](#).

Como puede verse en la Figura A.4, este *sprint* se ajustó bastante bien a la forma ideal del gráfico *burndown* dada la cantidad de tareas. Se estimaron 9 *story points*, pero se dedicaron al menos 12 días a la consecución del protocolo implementado.

Sprint 5. 31/01/18 - 14/02/18

Este quinto *sprint*, fue dedicado a conseguir la emisión de vídeo desde la RaspberryPi. Durante la preparación de este *sprint*, el Dr. César Ignacio García Osorio sugirió realizar un *refactor* de algunas partes del código ya desarrollado para mejorar su legibilidad y mantenibilidad. La descomposición de las tareas puede ser consultada en [Sprint 5⁵](#).

Como puede verse en la Figura A.5, este *sprint* no se ajustó demasiado a la forma ideal del gráfico *burndown* dada la longitud de algunas de las

⁴https://github.com/mbm0089/gii_0_17.02_snsi/milestone/4?closed=1

⁵https://github.com/mbm0089/gii_0_17.02_snsi/milestone/5?closed=1



Figura A.5: Gráfico burndown.



Figura A.6: Gráfico burndown.

tareas creadas. Se estimaron 14 *story points*, y se dedicaron 14 días a lograr los objetivos marcados.

Sprint 6. 14/02/18 - 08/03/18

Este sexto *sprint*, fue dedicado a realizar una implementación que sustituyese al algoritmo de Campos Potenciales. Durante la preparación de este *sprint*, se propuso crear otro algoritmo, dados los problemas que presentaba el algoritmo de Campos Potenciales. El Dr. César Ignacio García Osorio mostró preocupación por las latencias existentes en el *feed* de vídeo proveniente de la RaspberryPi, así que también se comprobó el código, y se buscaron formas de lograr que este fuese más fluido. La descomposición de las tareas puede ser consultada en [Sprint 6⁶](#).

⁶https://github.com/mbm0089/gii_0_17.02_snsi/milestone/6?closed=1



Figura A.7: Gráfico burndown.

Como puede verse en la Figura A.6, este *sprint* no se ajustó a la forma ideal del gráfico *burndown* dada la existencia de una única tarea, y su longitud. Se estimaron 8 *story points*, pero realmente se dedicaron 14 días a lograr los objetivos marcados.

Sprint 7-8. 08/03/18 - 12/04/18

Este séptimo-octavo *sprint* se creó con longitud doble, debido a la necesidad de generar una primera *release* y la cantidad de trabajo requerido para ella. Fue dedicado a finalizar la implementación del VFH, que sustituyese al algoritmo de Campos Potenciales. Se creó una página web que permitía a un usuario iniciar sesión y visualizar el vídeo proveniente del drone asignado, así como su control remoto. Durante este *sprint* se implementó la parte central de la arquitectura de este proyecto.

La descomposición de las tareas puede ser consultada en [Sprint 7-8](#)⁷.

Como puede verse en la Figura A.7, este *sprint* no se ajustó a la forma ideal del gráfico *burndown* dada la longitud de las tareas finales, que provocan que el escalado del gráfico no sea el apropiado. Se estimaron 52 *story points*, pero realmente se dedicaron 14 días a lograr los objetivos marcados. Obviamente, existió solapamiento de tareas. La *release* se finalizó correctamente y está disponible para su uso como versión *alpha*.

Sprint 9. 12/04/18 - 26/04/18

Este noveno *sprint* fue dedicado a comenzar con la implementación del Filtro de Partículas, ver [2], y crear los *scripts* de instalación de los que

⁷https://github.com/mbm0089/gii_0_17.02_snsi/milestone/7?closed=1



Figura A.8: Gráfico burndown.

carecía la primera release.

La descomposición de las tareas puede ser consultada en [Sprint 9⁸](#).

Como puede verse en la Figura A.7, este *sprint* se ajustó bastante bien a la forma ideal del gráfico *burndown*. Sin embargo, al final del mismo, se aprecia la desaparición de tareas. Esto es debido a que se trasladó la implementación del Filtro de Partículas al siguiente *sprint*. No existe una forma de *extender* la duración de una tarea entre *sprints* de forma que se presente en ambos. Se estimaron 6 *story points*, pero realmente se dedicaron 14 días a lograr los objetivos marcados, además se comenzó con el desarrollo del Filtro de Partículas. En un principio se estimó la duración del desarrollo del Filtro de Partículas en 8 *story points*.

Sprint 10. 26/04/18 - 11/05/18

Este décimo *sprint* fue dedicado a la implementación del Filtro de Partículas.

Tal y como se ha explicado con anterioridad, no existe una forma de establecer la misma *issue* en varios *sprints*, de manera que el gráfico *burndown* de este periodo de tiempo se encuentra vacío. El Filtro de Partículas no se finalizó durante el *sprint*, y por lo tanto fue extendido al siguiente.

La duración del desarrollo del Filtro de Partículas se extendió a 21 *story points*.

⁸https://github.com/mbm0089/gii_0_17.02_snsi/milestone/8?closed=1



Figura A.9: Gráfico burndown.

Sprint 11. 11/05/18 - 25/05/18

Este undécimo *sprint* fue dedicado a continuar con la implementación del Filtro de Partículas, y establecer el hardware restante en el drone. Además, se implementaron los controladores PID, ver [3], necesarios para la segunda release del proyecto.

La descomposición de las tareas puede ser consultada en [Sprint 11⁹](#).

Como puede verse en la Figura A.9, este *sprint* se ajustó muy bien a la forma ideal del gráfico *burndown*, dada la gran cantidad de tareas creadas. Sin embargo al final del mismo, se aprecia la desaparición de tareas, debido al problema de extensión de *issues* anteriormente mencionado. Se estimaron 27 *story points*, pero realmente se dedicaron 14 días a lograr los objetivos marcados, además del continuar con el desarrollo del Filtro de Partículas.

Sprint 12. 25/05/18 - 8/06/18

Este duodécimo *sprint* está siendo dedicado a finalizar la documentación del proyecto, y a las pruebas en un entorno real.

La descomposición de las tareas puede ser consultada en [Sprint 12¹⁰](#), aunque no se encuentra cerrado, ni finalizado.

Por el momento se han estimado 30 *story points*, entre los que se cuentan los 21 del Filtro de Partículas, y 8 de ellos se corresponden con la adquisición de métricas de SonarCloud.

⁹https://github.com/mbm0089/gii_0_17.02_snsi/milestone/10?closed=1

¹⁰https://github.com/mbm0089/gii_0_17.02_snsi/milestone/11

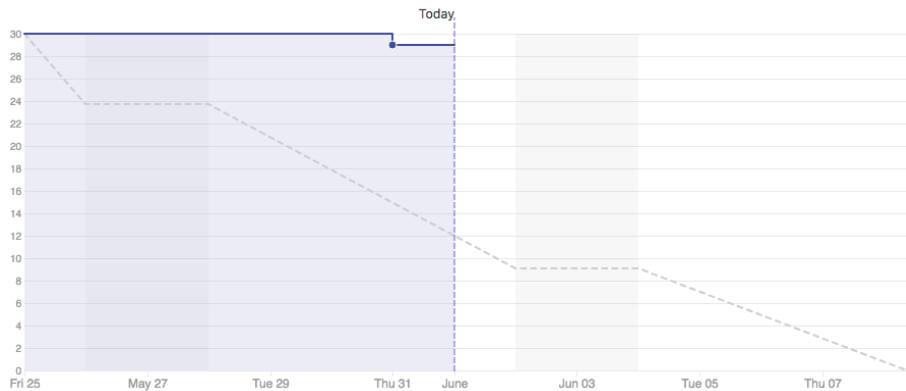


Figura A.10: Gráfico burndown.

Concepto	Coste
Salario mensual neto	1102€
Retención IRPF (13.55 %)	271€
Seguridad Social (29.9 %)	598€
Salario mensual bruto	2000€
Total 4 meses	8000€

Tabla A.1: Costes de personal de desarrollo

A.3. Estudio de viabilidad

Viabilidad económica

En este apartado se detallan los costes y beneficios que podría suponer el proyecto, de desarrollarse con una intención comercial.

Costes

De personal: Para el desarrollo del proyecto tan solo se ha empleado un desarrollador, empleado a tiempo completo durante cuatro meses, de febrero a mayo.

La retribución a la Seguridad Social se ha calculado como un 23,60 % por contingencias comunes, más un 5,50 % por desempleo de tipo general, más un 0,20 % para el Fondo de Garantía Salarial y más un 0,60 % de formación profesional. En total un 29,9 % que se aplica al salario bruto. Ver [4].

Concepto	Coste
Ayudante Doctor	$353,04 \times 0,33 = 117,68\text{€}$
Contratado Doctor Permanente	$339,1 \times 0,33 = 113,04\text{€}$
Contratado Doctor Básico	$389,14 \times 0,33 = 129,72\text{€}$
Total 4 meses	360,44€

Tabla A.2: Costes totales de asesoría

Se estima también la ayuda prestada por los tres tutores, dividiendo un crédito a partes iguales entre ellos. Para lo cual se utilizará la tabla disponible en [5], y haciendo el cálculo de la siguiente manera:

- Sueldo base Ayudante Doctor: 1815,61€. 14 pagas anuales. Total 25418,54€ al año. El cual imparte 24 créditos, de forma que tiene un coste por crédito de 1059,1€ al año. Quedando así en 353,04€ por los 4 meses.
- Sueldo base Contratado Doctor Permanente: 2325,24€. 14 pagas anuales. Total 32553,36€ al año. El cual imparte 32 créditos, de forma que tiene un coste por crédito de 1017,3€ al año. Quedando así en 339,1€ por los 4 meses.
- Sueldo base Contratado Doctor Básico: 2001,27€. 14 pagas anuales. Total 28017,8€ al año. El cual imparte 24 créditos, de forma que tiene un coste por crédito de 1167,41€ al año. Quedando así en 389,14€ por los 4 meses.

De esta forma, y tal y como se ha establecido la división a partes iguales del crédito correspondiente:

De esta manera, los costes totales de personal ascienden a:

De hardware: Para el desarrollo del proyecto se han empleado numerosos elementos de hardware que se detallan a continuación:

De software: Para el desarrollo de este proyecto, se valoran los costes del siguiente software, cuya amortización se ha establecido en 2 años.

Varios: Los siguientes, son costes variados:

Concepto	Coste
Desarrolladores	8000€
Asesoría Doctores	360,44€
Total 4 meses	8360,44€

Tabla A.3: Costes totales de personal

Concepto	Coste	Coste amortizado
Ordenador portátil	800€	66,67€
Emisora	177,79€	11,86€
Drone	Desglose	
RaspberryPi 3B	33,74€	2,25€
Tarjeta de memoria	21,99€	1,47€
Carcasa RaspberryPi	10,99€	0,74€
Cámara PiNoIR	28,99€	1,94€
LED IR	9,99€	0,67€
Controladora de vuelo	38,99€	2,6€
Motores EMAX mt2216	47,84€	3,19€
ESC 20A EMAX	30,53€	2,04€
Hélices de repuesto	10,34€	0,69€
Cargador Baterías	18,63€	1,25€
Sensores Ultrasonidos HCSR04	8,79€	0,59€
Chasis <i>drone</i> con PCB	19,99€	1,34€
Tren de aterrizaje	5,89€	0,4€
Cables	7,09€	0,48€
Tornillos	17,15	1,15€
Hilo de estaño	9,9€	0,67€
Soldador JBC 11W	41,95€	2,8€
Total	1340,58€	89,38€

Tabla A.4: Costes de hardware

Concepto	Coste	coste amortizado
PyCharm IDE	79,6€	13,27€
WebStorm IDE	51,6€	8,6€
GitHub Privado	23,43€	3,91€
SonarCloud Privado	40€	6,67€
Total	194,63€	32,45€

Tabla A.5: Costes de software

Concepto	Coste
Alquiler de oficina	600€
Internet	94,4€
Alquiler nave de pruebas	600€
Total	1294,4€

Tabla A.6: Costes variados

Concepto	Coste
Personal	8360,44€
Hardware	1340,58€
Software	194,63€
Varios	1294,4€
Total	11190,05€

Tabla A.7: Costes totales del proyecto

Totales: El total de los costes es el siguiente:

Beneficios

Por el momento, se trata de un proyecto con fines académicos, y con unos gastos relativamente elevados para la etapa tan temprana en que se encuentran.

Sin embargo, los beneficios podrían ser muy elevados. Se va a considerar un sistema de suscripción anual al servicio de vigilancia mediante *drones*. Dicho servicio proporciona vigilancia mediante *drones*, los cuales estarán

Entorno	Dimensiones	Drones	Coste
Nave	$> 2000m^2$	1	8000€/año
Nave grande	$> 4000m^2$	2	15000€/año
Grupo de naves	$> 15000m^2$	4	28000€/año

Tabla A.8: Monetización del proyecto.

supervisados por personal humano, y su mantenimiento. El servicio se detalla en la siguiente tabla:

La finalidad de establecer las dimensiones y el número de *drones*, se corresponde con la necesidad de realizar una vigilancia adecuada de un recinto de dimensiones determinadas. Es decir, no es lo mismo cubrir una superficie de $1000m^2$ que una de $2000m^2$, por supuesto teniendo en cuenta la distribución interior de las mismas. El último elemento de la tabla A.8, se corresponde con un grupo de naves ubicadas en el mismo recinto, de forma que los *drones* tendrían acceso a cada una de las naves, y a un canal de tránsito entre ellas.

Los costes serían rentables para las empresas, dado que contratar personal de seguridad acorde a la dimensionalidad del negocio es más costoso que mantener los *drones* establecidos en la tabla A.8.

Punto de equilibrio El punto de equilibrio establece el momento en el que se alcanza a cubrir la inversión inicial. Los cálculos propuestos a continuación son una estimación aproximada:

Asumiendo el coste de personal de estos 4 meses en los 8360,44€ resultantes de la tabla A.3, podemos suponer un coste anual de 25081,32€ por empleado¹¹. Se estima que para llevar a cabo este proyecto de forma correcta, y ágil, se deberían emplear a tiempo completo, al menos, 3 desarrolladores y 2 ingenieros en electrónica. Suponiendo unos gastos anuales de personal de desarrollo de 125406,6€.

Se establece que un vigilante/piloto es capaz de atender hasta 8 *drones*. De forma que se añade otro empleado por cada 8 *drones* construidos.

El coste de elaborar un *drone* con las características detalladas en la tabla A.4 asciende, sin contar con la necesidad de una emisora por *drone*, y un ordenador portátil por *drone*, a 362,79€. Sin embargo, teniendo en cuenta

¹¹No se ha eliminado el coste de los tutores.

Concepto	Coste anual
Desarrolladores	125406,6€
Piloto/Vigilante	25081,32€ × <i>drone</i> /8
Hardware	550€ × <i>drone</i>
Mantenimiento	150€ × <i>drone</i>
Seguros	265€ × <i>drone</i>
Formación	30000€
Total anual	(125406,6 + 965 × <i>drone</i> + 30000 + 25081,52 × <i>drone</i> /8) = 319414,2€

Tabla A.9: Gastos totales

que la tecnología empleada no es tan precisa como se desearía, se añade un margen que permitiría mejorar el hardware empleado, estableciendo el total de elaboración de un *drone* en 550€.

Los costes de mantenimiento y reparación de un *drone*, basados en la experiencia personal con estos dispositivos, y en la tecnología empleada, se establece en 150€ anuales por dispositivo. El coste del seguro de responsabilidad civil de la empresa se estima en 265€ por dispositivo, con una cobertura mínima de 300000€, tal y como se estipula en [6], y teniendo en cuenta que este seguro cubriría daños al dispositivo (lo cual permite acotar mejor las estimaciones dadas). Para más información ver [tablas de costes](#)¹².

Además, se incluye un gasto de daño social generado, ver A.3, valorado en 30000€ al año, en concepto de formación para los afectados.

De esta forma, se establece el siguiente resumen de costes, asumiendo que las suscripciones anuales generadas son siempre la suscripción más simple de que se dispone, y que establece un ingreso de 8000€ anuales:

El total anual de la tabla A.9, se extrae de obtener el número de *drones* a través de la siguiente fórmula, donde $x = \text{drone}$ por simplificar:

$$8000x - \left(125406,6 + 965x + 30000 + \frac{25081,52x}{8} \right) \approx 40 \text{ drones/año} \quad (\text{A.1})$$

para cubrir los costes generados.

Responsabilidad Social

Con el auge de la mecanización y automatización de los puestos de trabajo, es obvio que los seres humanos iremos perdiendo cabida en empleos

¹²<https://www.aenus.es/responsabilidad-civil-danos-profesional/>

que un sistema informatizado podría realizar con mayor eficiencia, y sobre todo de forma más económica.

Este proyecto, tiene un gran potencial de causar un daño social importante. El puesto de vigilante nocturno ha provisto de seguridad en muchos entornos laborales, y ha contribuido a generar riqueza protegiendo los bienes de las diferentes empresas que hacen uso de este personal.

Por tanto, si este proyecto avanza y llega a ser utilizado en sustitución de personal humano, se propone impartir formación en pilotaje de este tipo de *drones* a los trabajadores afectados, para que puedan pasar a formar parte de la empresa. De esta forma, el efecto de esta tecnología supondrá una mejora en sus condiciones de trabajo y una evolución en materia de seguridad, tanto para las empresas, como para estos trabajadores.

Viabilidad legal

En esta sección se aborda el contexto legal en el que se encuentra el proyecto, así como todo lo relacionado con las licencias.

Dado que el desarrollo de casi la totalidad de las implementaciones de este proyecto ha sido propia, esta sección será breve en lo que a licencias se refiere.

Software

El proyecto consta de las siguientes dependencias software:

De esta forma, el código desarrollado deberá tener una licencia compatible con MIT y BSD, ambas muy permisivas al respecto. La monetización del proyecto se basa en un sistema de suscripción al mismo, pero la existencia de competencia redundante en una mejora de los productos y servicios, ya que fuerza a las empresas a mantenerse al día. Por tanto, se libera todo el código, tanto del *backend* como del *frontend*, bajo licencia BSD de 3 cláusulas.

Puede leerse un resumen de la licencia en [BSD-3 Clause license¹³](#).

Documentación, imágenes y vídeos

La documentación generada, así como las imágenes utilizadas (salvo dos presentadas en artículos académicos listados en la bibliografía, y sobre las que se ha aportado el crédito correspondiente) y los vídeos han sido de generación propia.

¹³<https://opensource.org/licenses/BSD-3-Clause>

Dependencia	Versión	Descripción	Licencia
Numpy	1.14.2	Se trata de la librería de cálculo científico por excelencia en Python.	BSD
SciPy	1.0.1	Librería de múltiples herramientas para cálculo científico.	BSD
PySerial	3.4	Librería que proporciona acceso a puertos serie.	BSD
Bluetin_Echo	0.1.1	Librería que proporciona una clase para usar los sensores de ultrasonidos HC-SR04.	BSD
UV4L	1.14	Conjunto de drivers para captura de vídeo, y framework WebRTC basado en OpenWebRTC.	BSD

Tabla A.10: Dependencias en *backend*.

Dependencia	Versión	Descripción	Licencia
Flask	1.0	Se trata de un framework para desarrollo de aplicaciones web en Python.	BSD
Eventlet	0.22.1	Librería de comunicación concurrente para Python.	MIT
SQLAlchemy	1.2.6	ORM para Python.	MIT

Tabla A.11: Dependencias en *frontend*.

Recurso	Licencia
Código Backend	BSD
Código Frontend	BSD
Documentación	CC-BY-4.0
Imágenes	CC-BY-4.0
Vídeos	CC-BY-4.0

Tabla A.12: Resumen de licencias.

Por lo tanto se establece una licencia *Creative Commons Attribution 4.0 International* (CC-BY-4.0), que permite compartir, adaptar y distribuir para cualquier propósito la obra, incluso con fines comerciales, siempre y cuando se dé la debida atribución de la autoría del trabajo.

Puede leerse un resumen de la licencia en [CC-BY 4.0¹⁴](#).

Resumen

A modo de resumen de las licencias dadas al material del proyecto, se añade la siguiente tabla:

Marco legal

Más allá de tratarse de únicamente un proyecto software, el uso de *drones* entra en un marco legal muy delicado en España.

La legislación al respecto es de reciente creación, y difiere entre países. A continuación se especifica el marco legal en el que se engloba este proyecto, y las diferentes repercusiones que tiene sobre el uso del mismo.

Legislación sobre Aeronaves

Según el Real Decreto 1036/2017, disponible en [BOE - RD1036/2017¹⁵](#):

- Para operar un **drone de forma profesional**, es necesario ser dado de alta por AESA¹⁶ como operador de *drones*.
- No está permitido sobrepasar los 120m de altura.

¹⁴<https://creativecommons.org/licenses/by/4.0/>

¹⁵https://www.seguridadaerea.gob.es/media/4629426/rd_1036_17_rpas.pdf

¹⁶Agencia Estatal de Seguridad Aérea

- Se permite el vuelo sobre aglomeraciones de personas y en zonas urbanas, siempre y cuando la aeronave no sobrepase los 10 kg. Se debe guardar una distancia de seguridad de 50 m con edificios y personas. El piloto no podrá encontrarse a más de 100 m de distancia.
- Se permite el **vuelo fuera del alcance visual** con aeronaves de menos de 2 kg, y que cuenten con medios para detectar y evitar a otros usuarios del espacio aéreo.
- La aeronave deberá contar con una placa identificativa, que conste de nombre del fabricante, tipo, modelo, número de serie, y nombre del operador y datos de contacto.

Dado el marco establecido en la anterior lista, y dado que el *drone* creado no sobrepasa los 2 kg de peso, tan solo sería necesario contar con personal cualificado, y con una placa identificativa del dispositivo. Además, tal y como se estipula, se hace referencia a *drones* con medios para evitar colisiones, lo cual enmarca este proyecto.

Pese a no mentirse el uso en recintos privados, y acogiendo el proyecto a la normativa más restrictiva, parece sencillo y razonable cumplir con la legislación vigente.

GDPR

La reciente implantación de la **GDPR^{17,18}**, lleva consigo una serie de restricciones y notificaciones a tener en cuenta:

- Se debe señalar públicamente que la zona cuenta con sistemas de vídeo vigilancia.
- Asimismo se debe informar que los datos pueden ser utilizados con fines de mejora del producto.
- Se informará del no tratamiento de datos biométricos recogidos durante sesiones de grabación.
- Se informará de los derechos de cancelación y de acceso a los datos en cualquier momento.

¹⁷<https://www.eugdpr.org/key-changes.html>

¹⁸Reglamento General de Protección de Datos de la Unión Europea

- La transmisión de vídeo se hará de forma cifrada (es requisito de WebRTC, de manera que no es complejo de cumplir), bien mediante HTTPS o túneles TLS.
- Derecho al olvido. Permite que una persona captada por el sistema de vigilancia pueda ejercer su derecho a que los datos sean eliminados, de forma que se detenga su procesamiento o compartición.

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este anexo se recogen las especificaciones de requisitos requeridas para este proyecto. Estas definirán el comportamiento esperado del sistema.

B.2. Objetivos generales

En este proyecto se ha tratado de llevar a cabo los siguientes objetivos de carácter general:

- Diseñar un *drone* capaz de recorrer un espacio, en el que existan obstáculos, de forma segura.
- Diseñar un sistema de acceso al *drone* de forma segura. Tanto para controlarlo de forma remota, como para activar los mecanismos de control automatizados.
- Diseñar una interfaz web que permita la visualización en tiempo real de la cámara del *drone*, así como su control remoto por un operador.

B.3. Catálogo de requisitos

Los siguientes requisitos se derivan de los objetivos generales arriba dispuestos:

Requisitos Funcionales

RF-1 Acceso al sistema: El usuario debe poder iniciar sesión en el sistema.

RF-2 Cierre de sesión: El usuario debe poder cerrar sesión en el sistema.

RF-3 Gestión del sistema: El usuario debe ser capaz de gestionar el sistema completo que hay a bordo del *drone*.

RF-3.1 Gestión del *streaming* de vídeo: El usuario debe poder iniciar/detener el streaming de vídeo.

RF-3.2 Grabaciones de vídeo: El usuario debe poder iniciar/detener y descargar las grabaciones de vídeo.

RF-3.3 Gestión de controles del *drone*: El usuario debe poder activar/desactivar el control manual del dispositivo.

RF-3.3.1 Seguridad automatizada: El *drone* debe presentar mecanismos de autopreservación. Debe protegerse frente a obstáculos.

RF-3.4 Gestión de control automático del *drone*: El usuario debe poder activar/desactivar los mecanismos de control automatizado.

RF-3.5 Registro de acciones: Las acciones llevadas a cabo por el usuario deben guardarse en un registro.

RF-4 Gestión del *backend*: El usuario debe poder acceder al equipo que controla el *backend*.

RF-4.1 Gestión del sistema operativo: El usuario debe poder llevar a cabo las tareas de gestión del sistema operativo como crea conveniente.

RF-5 Registro de sucesos: Intentos de intrusión deben ser registrados.

RF-6 Reconocimiento de dispositivos: La aplicación web debe ser capaz de reconocer un dispositivo (*joystick* o emisora) conectado al equipo, para llevar a cabo el control del *drone*.

Requisitos no funcionales

RNF-1 Usabilidad: El *frontend* ha de ser sencillo de utilizar e intuitivo. En el caso de producirse errores, estos deben ser claramente explicados, o registrados.

RNF-2 Rendimiento: Tanto el *frontend*, como el *backend* deben funcionar de forma eficiente. Retardos elevados podrían dar lugar a problemas muy serios en la ejecución de las ordenes del usuario.

RNF-3 Capacidad y Escalabilidad: Tanto el *frontend* como el *backend* deben ser escalables. El *frontend* debe ser capaz de soportar una asiduidad grande de usuarios, así como permitir añadir múltiples agentes a cada usuario. El *backend*, debe permitir añadir controladores sin que haya que reestructurar todo el sistema.

RNF-4 Disponibilidad: El sistema al completo debe ser robusto. El *frontend* debe estar disponible para su uso el mayor tiempo que sea posible. El *backend* debe ser capaz de recuperarse de errores, y determinar acciones a tomar en el caso de que se produzcan.

RNF-5 Seguridad: Tanto el acceso al *frontend* como al *backend* deben hacerse de forma segura. En ningún momento el *drone* debe guardar imágenes de la zona vigilada. El almacenamiento de contraseñas se hará de forma cifrada, y la comunicación entre el *backend* y el *frontend* deberá ser segura.

B.4. Especificación de requisitos

Esta sección detallará el diagrama de casos de uso, y detallará cada uno de ellos.

Actores

- **Vigilante:** Actor a cargo de la gestión del sistema en producción. Interactúa con el FrontEnd.
- **Admin:** Actor a cargo de la gestión del sistema en desarrollo o en caso de resolución de problemas en producción. Interactúa con el BackEnd.

Casos de Uso

CU-01		Acceso al sistema
Actor	Vigilante	
Requisitos asociados	RF-1, RF-5	
Descripción	Permite al usuario iniciar sesión en el sistema vía web.	
Precondiciones	La base de datos está activa. El usuario dispone de cuenta activa.	
Acciones		<ol style="list-style-type: none"> 1. El usuario accede a la página web. 2. El usuario introduce sus credenciales. 3. Se muestra la página principal con el logo y los botones asociados al streaming de vídeo.
Postcondición		El sistema habrá asignado al usuario el <i>drone</i> correspondiente.
Excepciones		<ul style="list-style-type: none"> ■ Contraseña/Usuario incorrectos (mensaje). ■ No existen <i>drones</i> asociados (mensaje).
Importancia		Alta.

Tabla B.1: CU-01. Acceso al sistema

CU-02 Cierre de sesión	
Actor	Vigilante
Requisitos asociados	RF-2, RF-5
Descripción	Permite al usuario cerrar sesión en el sistema vía web.
Precondiciones	La base de datos está activa. El usuario dispone de cuenta activa. El usuario tiene iniciada una sesión.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede a la aplicación web. 2. El usuario pulsa sobre el enlace ‘Logout’ situado en la parte superior izquierda. 3. Se muestra la página de inicio de sesión con un mensaje de despedida.
Postcondición	El sistema habrá dado por cerrada la sesión de usuario. Se redirige al usuario a la página de inicio de sesión.
Excepciones	El usuario no tenía sesión activa. (redirección a login).
Importancia	Alta.

Tabla B.2: CU-02. Cierre de sesión

CU-03	Gestión del sistema
Actor	Vigilante
Requisitos asociados	RF-1, RF-3
Descripción	Permite al usuario gestionar las acciones del <i>drone</i> vía web.
Precondiciones	La base de datos está activa. El usuario dispone de cuenta activa. El usuario tiene iniciada una sesión. El usuario tiene asignado un <i>drone</i> .
Acciones	<ol style="list-style-type: none"> 1. El usuario conecta un dispositivo con el número de canales necesarios. 2. Se informa sobre el reconocimiento de dicho dispositivo y se muestran los valores de los canales activos. 3. Se muestra el botón de activación del control manual.
Postcondición	El sistema habrá asignado al usuario el <i>drone</i> correspondiente. La aplicación web habrá reconocido el dispositivo conectado. Se dará la posibilidad de activar el control manual.
Excepciones	Dispositivo no reconocido.
Importancia	Alta.

Tabla B.3: CU-03. Gestión del sistema

CU-04	Gestión del streaming de vídeo
Actor	Vigilante
Requisitos asociados	RF-3, RF-3.1 RF-3.5
Descripción	Permite al usuario activar/desactivar el feed de vídeo proveniente del <i>drone</i> .
Precondiciones	La base de datos está activa. El usuario dispone de cuenta activa. El usuario tiene iniciada una sesión. El usuario tiene asignado un <i>drone</i> .
Acciones	El usuario pulsa sobre el botón ‘Start/Stop Streaming’
Postcondición	La aplicación web inicia/detiene el vídeo.
Excepciones	<i>Drone</i> no disponible (mensaje).
Importancia	Alta.

Tabla B.4: CU-04. Gestión del streaming de vídeo

CU-05	Grabación de vídeo
Actor	Vigilante
Requisitos asociados	RF-3, RF-3.2 RF-3.5
Descripción	Permite al usuario grabar el feed de vídeo proveniente del <i>drone</i> .
Precondiciones	La base de datos está activa. El usuario dispone de cuenta activa. El usuario tiene iniciada una sesión. El usuario tiene asignado un <i>drone</i> .
Acciones	El usuario pulsa sobre el botón ‘Record Video’
Postcondición	La aplicación web comienza con la grabación del feed de vídeo.
Excepciones	Feed de vídeo no disponible (mensaje).
Importancia	Baja.

Tabla B.5: CU-05. Grabación de vídeo

CU-06	Control manual del <i>drone</i>
Actor	Vigilante
Requisitos asociados	RF-3, RF-3.3 RF-3.5, RF-3.3.1, RF-6
Descripción	Permite al usuario controlar el <i>drone</i> de forma manual.
Precondiciones	La base de datos está activa. El usuario dispone de cuenta activa. El usuario tiene iniciada una sesión. El usuario tiene asignado un <i>drone</i> . El usuario ha conectado un dispositivo adecuado.
Acciones	El usuario pulsa sobre el botón ‘Enable/Disable Manual’
Postcondición	La aplicación web solicita la activación del control manual al sistema. El sistema se conecta al <i>drone</i> y transmite la información proveniente de la aplicación web.
Excepciones	<i>Drone</i> no disponible (mensaje).
Importancia	Alta.

Tabla B.6: CU-06. Control manual

CU-07	Gestión de controles automáticos
Actor	Vigilante
Requisitos asociados	RF-3, RF-3.4 RF-3.5
Descripción	Permite al usuario activar/desactivar los mecanismos de control automatizado del <i>drone</i> .
Precondiciones	La base de datos está activa. El usuario dispone de cuenta activa. El usuario tiene iniciada una sesión. El usuario tiene asignado un <i>drone</i> .
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona los mecanismos de control automatizado que desea emplear. 2. El usuario pulsa sobre el botón ‘Enable auto’
Postcondición	La aplicación web solicita la activación de los controles automatizados al sistema. El sistema se conecta al <i>drone</i> y transmite la información proveniente de la aplicación web.
Excepciones	<ul style="list-style-type: none"> ■ <i>Drone</i> no disponible (mensaje). ■ Mecanismo automatizado no reconocido (mensaje).
Importancia	Media.

Tabla B.7: CU-07. Gestión de controles automáticos

CU-08	Gestión del <i>backend</i>
Actor	Admin
Requisitos asociados	RF-4, RF-5
Descripción	Permite al usuario gestionar el equipo que controla el <i>drone</i> .
Precondiciones	El usuario dispone de credenciales de acceso al sistema.
Acciones	<ol style="list-style-type: none"> 1. El usuario inicia sesión en el sistema. 2. Se informa sobre el registro de intentos de acceso no autorizado. 3. Se muestra el <i>prompt</i> del sistema a la espera de órdenes.
Postcondición	El sistema operativo habrá registrado el inicio de sesión del usuario. El sistema operativo habrá iniciado sesión con el usuario solicitado.
Excepciones	Acceso denegado, pubkey. (mensaje)
Importancia	Alta.

Tabla B.8: CU-08. Gestión del *backend*.

B.4. ESPECIFICACIÓN DE REQUISITOS

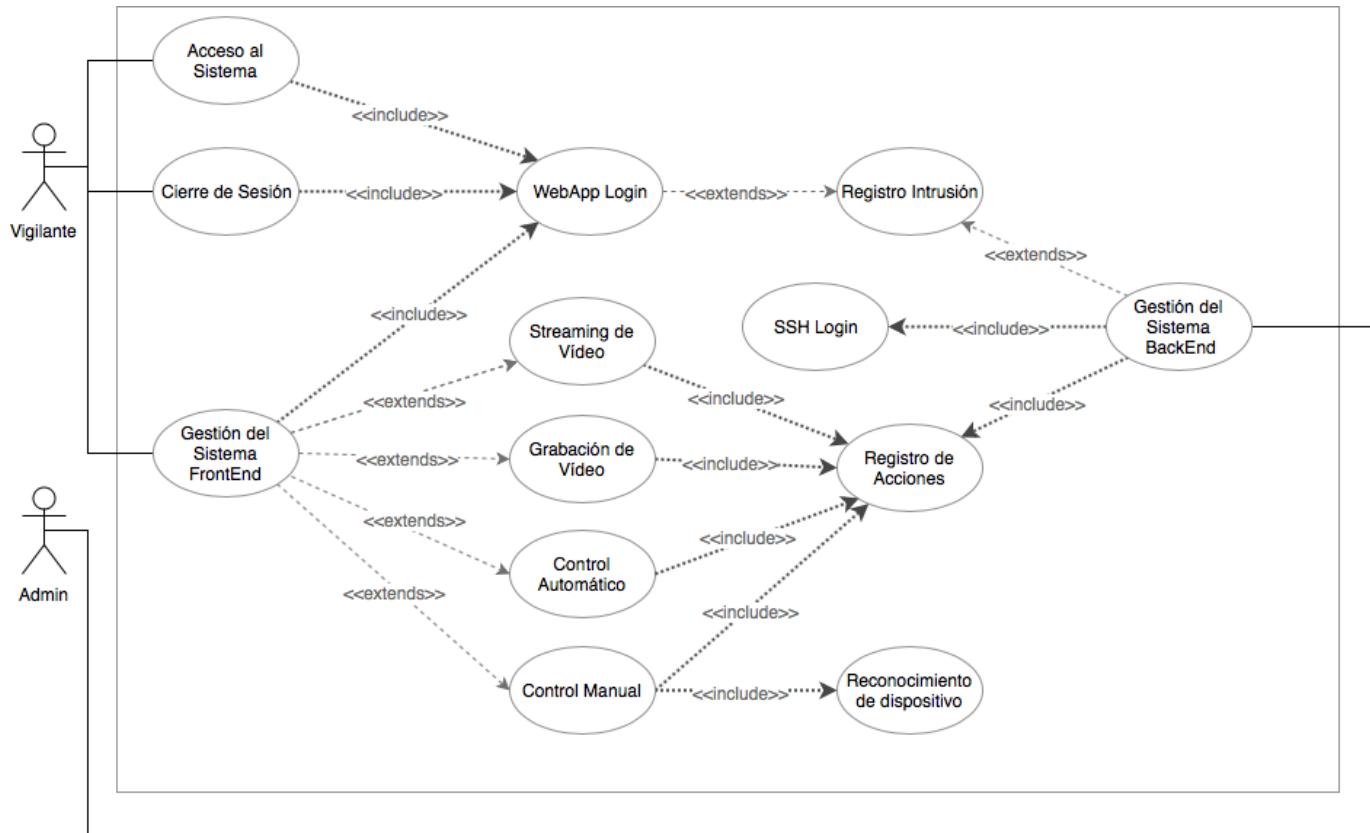


Figura B.1: Diagrama de Casos de Uso.

Apéndice C

Especificación de diseño

C.1. Introducción

En este anexo se definirá la forma en que se han resuelto los requisitos expuestos anteriormente. Se dará una visión detallada de la arquitectura del sistema, diferenciando la aplicación web y el agente, o *drone*.

C.2. Diseño de datos

WebApp

La aplicación visible por el usuario final maneja los siguientes datos, o entidades:

- Usuario: Un usuario dispone de un nombre, apellidos, una dirección de correo electrónico, un nombre de usuario (que utilizará para iniciar sesión), y una contraseña cifrada. Se genera un id automáticamente para cada usuario.
- Drone: Un *drone* dispone de un nombre, una localización, un controlador (que se corresponde con el id de usuario que lo controla), un puerto para el control remoto y un puerto para el *feed* de vídeo. Se genera un id automáticamente para cada *drone*.
- Registro de sesión: La aplicación web guarda registros del inicio/cierre de sesión por parte de un usuario. Dispone del identificador de usuario y de un timestamp.

- Registro de intrusión: La aplicación web guarda registros de inicios de sesión infructuosos. Dispone del usuario utilizado, y un *timestamp*.
- Registro de acciones: La aplicación web guarda registros de las acciones llevadas a cabo por cada usuario. Dispone de un identificador del usuario que realiza la acción, un timestamp y un identificador de la acción realizada.
- Acciones: Se trata de las acciones que realizan los usuarios. Dispone de un identificador y una descripción de la acción.

Agente

El agente basa su funcionamiento en el uso de las siguientes entidades:

- Controladores PID: Los controladores PID se encargan de proveer de valores para los diferentes canales automatizados del agente. Disponen de los coeficientes de las distintas componentes proporcional, integral y derivativa, así como de un límite superior y un límite inferior.
- Sensores: Los sensores de distancia disponen de un pin de disparo, un pin de recepción del eco, y un ángulo.
- Control Remoto: Se trata de un pequeño servidor encargado de recibir las órdenes de un usuario remoto. Dispone de una dirección y un puerto en el que escuchar.
- MSPio: Se trata de la implementación del protocolo MultiWiiSerial-Protocol, el cual permite establecer comunicación con la controladora de vuelo. Dispone de un puerto serie y una velocidad de transferencia.
- Evasión de obstáculos: Se trata de la implementación del algoritmo VFH. Dispone de las siguientes entidades:
 - Malla de histograma: Una representación del entorno del agente. Dispone de las medidas de los sensores, distancias máxima y mínima a cubrir, el mapa del entorno, el tamaño de la ventana a cubrir, el tamaño de las celdas que componen la malla, una medida del error de los sensores y su apertura.
 - Histograma polar: Una representación basada en sectores del entorno del agente. Dispone de una malla de histograma y el ancho de los sectores.

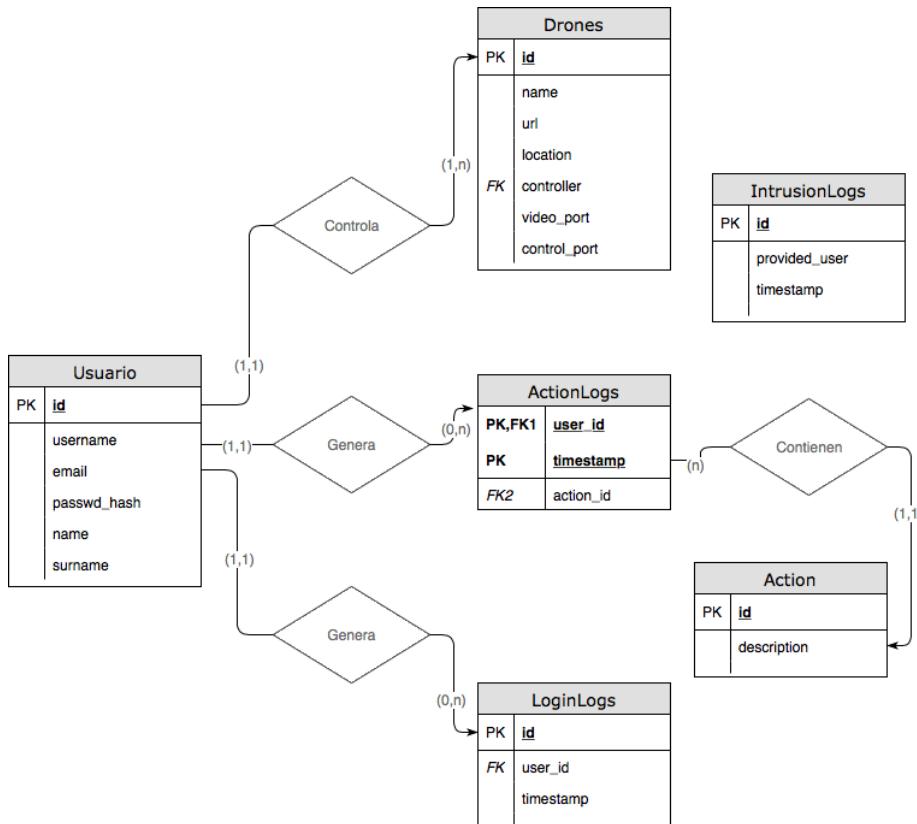


Figura C.1: Diagrama E/R para WebApp.

- Capa de salida: La capa de salida es la que realiza el cálculo de la orientación necesaria. Dispone de un histograma polar, un umbral de navegación segura, y la cantidad de sectores del histograma polar que se requieren para considerar un espacio como ‘amplio’.
- Sistema de localización: Se trata de la implementación del algoritmo del Filtro de Partículas. Dispone de una representación del entorno del agente, medidas de error para el control de rotación, de desplazamiento y de toma de medidas de los sensores, y el número de sectores a generar en cada celda.

C.3. Diseño procedimental

En este apartado se recoge el diagrama secuencial correspondiente a la totalidad del sistema. En él, se representa el flujo de ejecución que puede o no, ver subsección C.4, seguir el sistema creado.

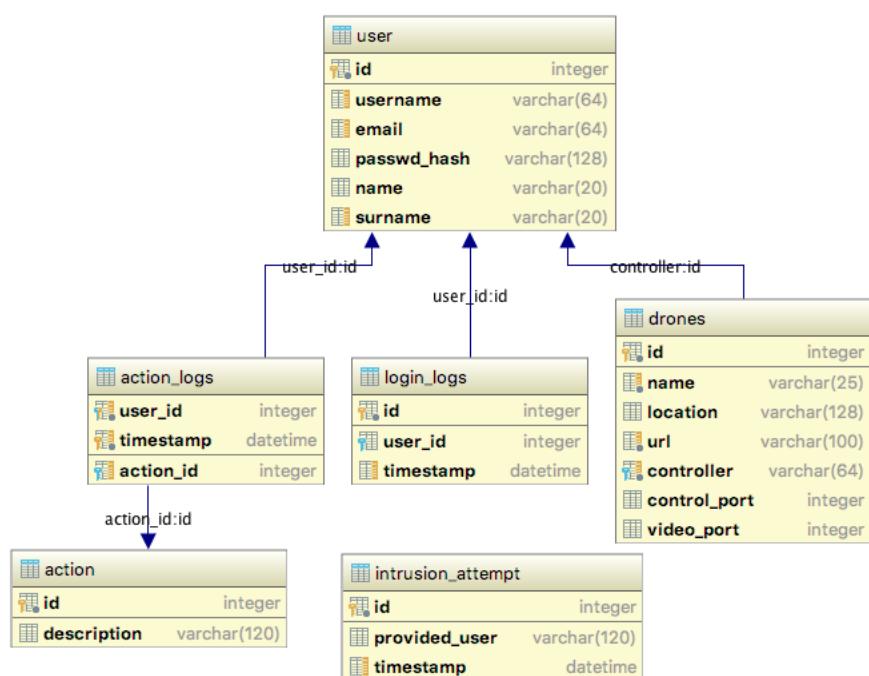


Figura C.2: Diagrama Relacional WebApp. Generado mediante PyCharm.

C.4. DISEÑO ARQUITECTÓNICO

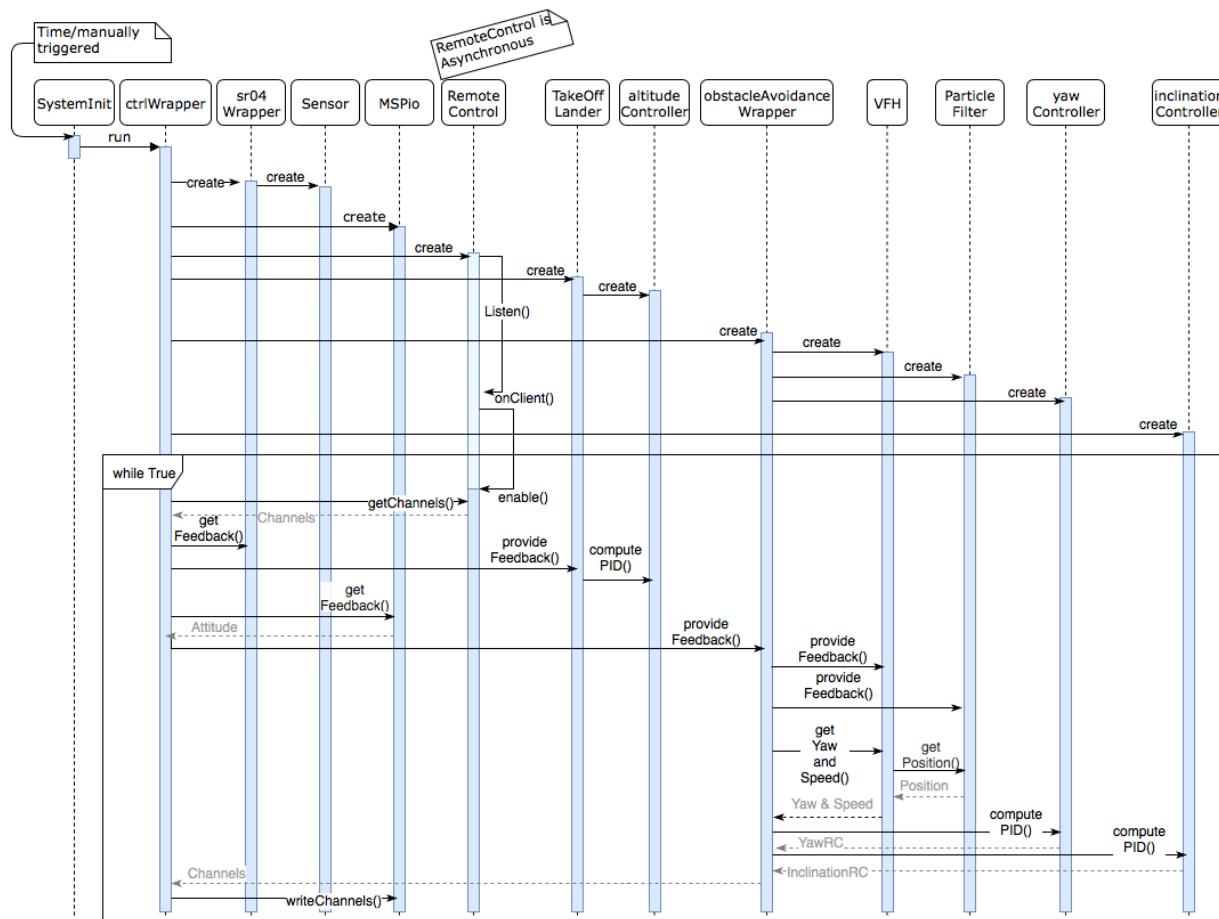


Figura C.3: Diagrama de secuencia del sistema.

C.4. Diseño arquitectónico

En esta sección se dará una aclaración al diagrama C.3, el cual da una idea general del funcionamiento del sistema. Para lograr un código fácil de mantener, y de escalar dado el potencial de crecimiento de las tecnologías relacionadas con este proyecto, se han generado una serie de clases intermedias que actúan a modo de *wrapper*. La interfaz de la aplicación se ha generado siguiendo el conocido patrón MVC¹.

Patrón MVC

Para la parte representada por la aplicación web, se ha hecho uso del habitual patrón MVC (*Model View Controller*), con la intención de separar la capa de lógica de la aplicación (*Controller*), de la interfaz de usuario (*View*) y de los datos manejados (*Model*).

El patrón sigue la siguiente disposición:

- **Model:** El modelo de la aplicación representa los datos a los que la aplicación accederá. Almacena los mismos y los gestiona. En el caso de este proyecto se trata de la base de datos de la aplicación Flask.
- **View:** La vista de la aplicación muestra al usuario una interfaz gráfica sobre la que trabajar. Al recibir acciones, se encarga de solicitar al controlador las medidas necesarias. En el caso de este proyecto se corresponde con la interfaz web de la aplicación Flask.
- **Controller:** El controlador podría considerarse el punto de comunicación entre el modelo y la vista. Se encarga de sincronizar los datos almacenados en el modelo con lo presentado en la vista, y de recibir los eventos generados por la vista y actuar en consecuencia.

Patrón Catálogo

El patrón catálogo no es tan habitual. Se utiliza sobre todo en lenguajes en los que se permite pasar una función o método como parámetro. Por ello en Python, donde todos los elementos son objetos de primera clase, es posible hacer uso de este patrón de forma muy sencilla.

¹Model View Controller. Es un patrón utilizado al diseñar interfaces gráficas, que permite desacoplar las capas de vista, modelo de datos y lógica de aplicación.

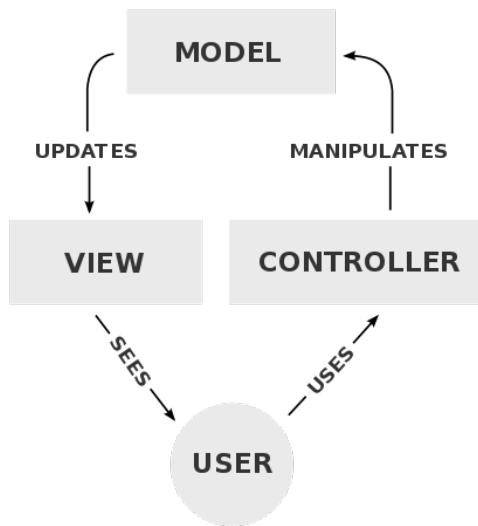


Figura C.4: Patrón MVC. Imagen de Wikipedia.

Permite establecer el comportamiento de una instancia más genérica, estableciendo los métodos generales de esta a métodos especializados de la clase que encapsula. En este caso, se encarga de proporcionar un acceso genérico a los métodos especializados de cada controlador automatizado.

Este patrón es el núcleo de funcionamiento del sistema de control del agente. En el diagrama C.3 puede verse como se instancian numerosos controladores (`RemoteControl`, `TakeOffLander` y `obstacleAvoidanceWrapper`). Dichos controladores, en el momento de ser instanciados por el `ctrlWrapper` se encapsulan en una clase interna denominada `PrioritizedController`. Dicha clase, podría ser considerada como la capa externa de un patrón proxy de protección², el cual permite comprobar si el controlador está activo, si requiere de información de los sensores para su funcionamiento, y si requiere ser bloqueado para usarse (para protegerse de accesos simultáneos en el caso de tener un comportamiento asíncrono). Además, provee de información sobre la *prioridad* del controlador.

La prioridad de un controlador es lo que hace tan flexible el sistema. El diagrama C.3 presenta un uso secuencial de los diferentes `PrioritizedController` generados, sin embargo, el orden de ejecución de cada uno, puede cambiar de forma dinámica. El `ctrlWrapper` basa el procesamiento de los canales de control (Acelerador, Balanceo, Inclinación, Rotación, Armar/Desarmar... etc) en la prioridad de los controladores.

²El patrón proxy de protección controla el acceso al objeto original

Por ejemplo: Supongamos que existen dos controladores, uno remoto y manual (este es el usuario conectado al *drone* a través de la interfaz web) con una prioridad igual a 5, y un controlador de rotación y velocidad autónomo, con una prioridad igual a 10.

La solicitud de los valores de los canales, se realiza por orden de prioridad de menor a mayor. De manera que el controlador automático de rotación y velocidad sobreescibirá la entrada del usuario (solo en los canales controlados), ya que tiene una prioridad superior. Supongamos que el usuario no desea eso. Prefiere ser él quien controle por completo el sistema, así que sencillamente arrancaría el sistema estableciéndose a sí mismo como el controlador más prioritario. Sobreescribiendo de esta forma al controlador automatizado.

Arquitectura general

La arquitectura de la parte visible del proyecto, la aplicación web, no presenta una complejidad elevada. Sin embargo, el sistema de control del agente no es tan intuitivo. Por ello, se presenta el siguiente diagrama C.5 a modo aclaratorio.

El uso de esta arquitectura presenta varias ventajas:

- Los controladores son independientes de la implementación del sistema de control.
- La arquitectura generada puede ser utilizada para controlar cualquier sistema basado en varios sistemas de control que se quieran priorizar.
- Permite llevar a cabo pruebas con mayor seguridad, dado que se puede hacer uso de un controlador manual solo cuando este se active, de manera que los controladores automatizados en pruebas puedan ser interrumpidos en cualquier momento.

Diseño de paquetes

Para organizar las diferentes partes del proyecto, se ha generado una estructura de paquetes que diferencia la ubicación de los mismos (agente o servidor) de forma sencilla.

En la figura C.6 puede verse la estructura de paquetes seguida en el agente. Los paquetes contienen:

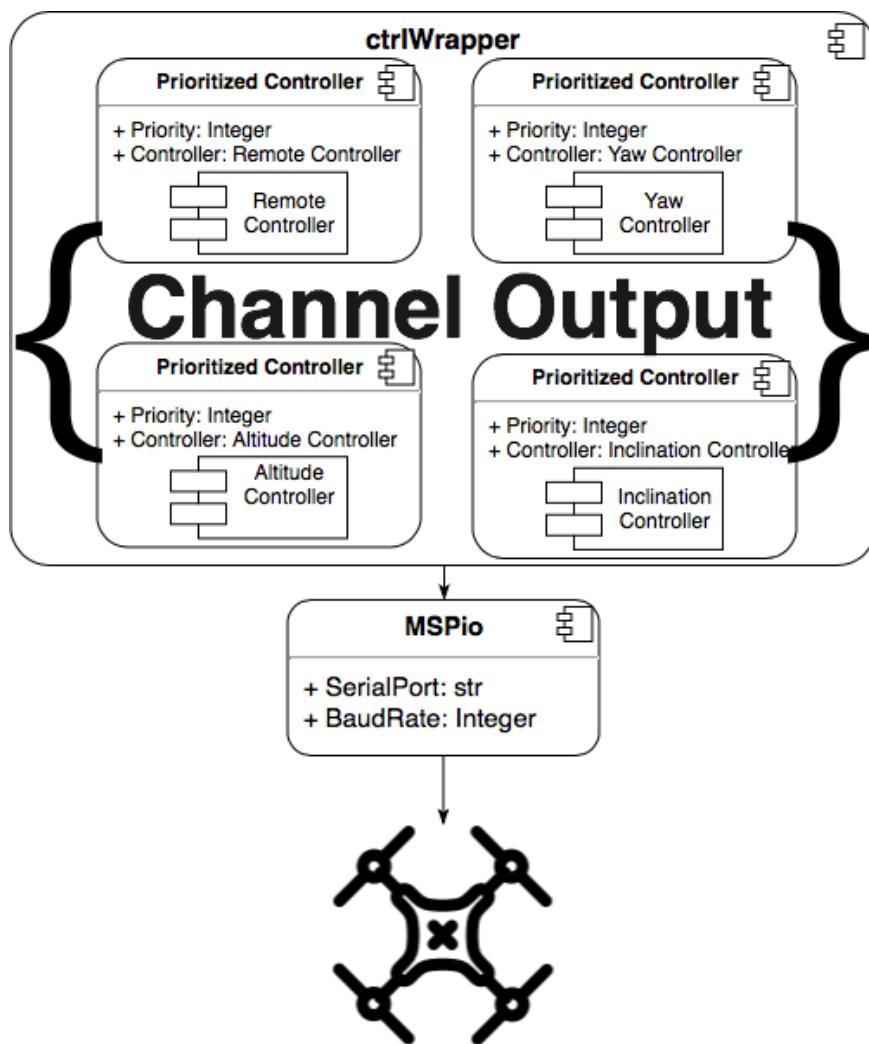


Figura C.5: Encapsulación de controladores mediante controladores priorizados.

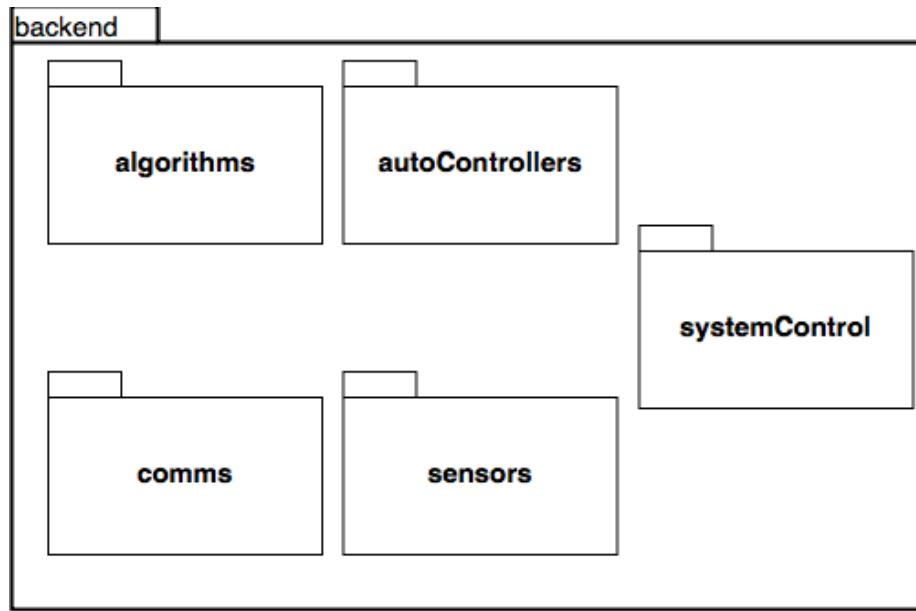


Figura C.6: Estructura de los paquetes en el agente.

- **algorithms**: Las implementaciones de los algoritmos empleados.
- **autoControllers**: Los diferentes sistemas de control automatizados implementados.
- **comms**: Los sistemas de comunicación implementados. Tanto entre el agente y la RaspberryPi, como entre el usuario y la RaspberryPi.
- **sensors**: Las implementaciones de adquisición de datos de los sensores usados.
- **systemControl**: Contiene la implementación del sistema de control del agente.

En la figura C.7 puede verse la estructura de paquetes seguida en el servidor web. Los paquetes contienen:

- **controller**: Contiene la implementación del controlador de la aplicación web.
- **model**: Contiene la implementación del modelo de datos utilizado por el controlador.

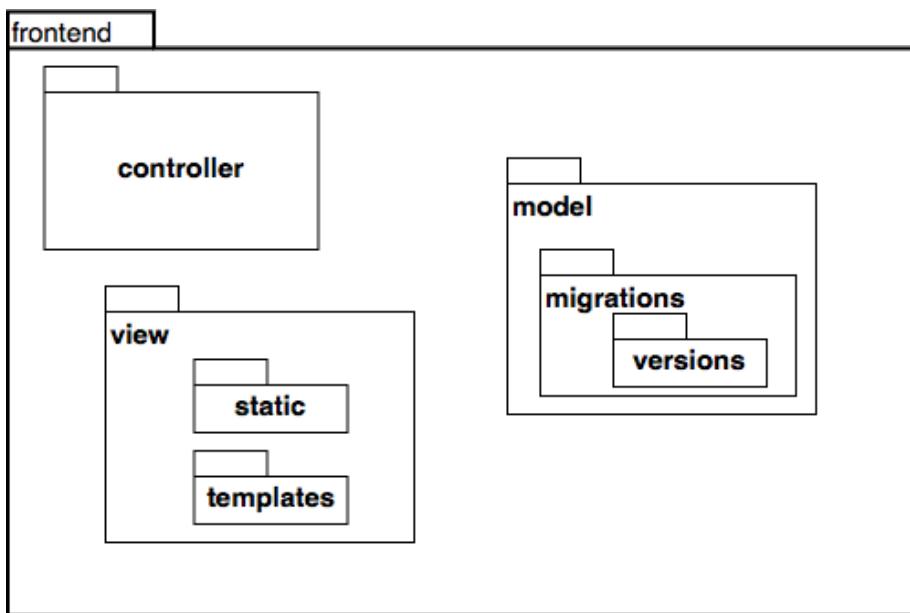


Figura C.7: Estructura de los paquetes en el servidor web.

- *migrations*: Contiene un mecanismo para realizar operaciones de upgrade/downgrade de las diferentes versiones de la base de datos.
 - *versions*: Contiene las versiones de la base de datos
- **view**: Contiene la vista de la aplicación web.
 - *static*: Contiene los elementos estáticos de la aplicación web.
 - *templates*: Contiene las plantillas que representan la interfaz de la aplicación web.

Diseño de clases

Siguiendo con la nomenclatura utilizada en el diseño de la estructura de paquetes, se dará a continuación las características de cada clase implementada:

FrontEnd

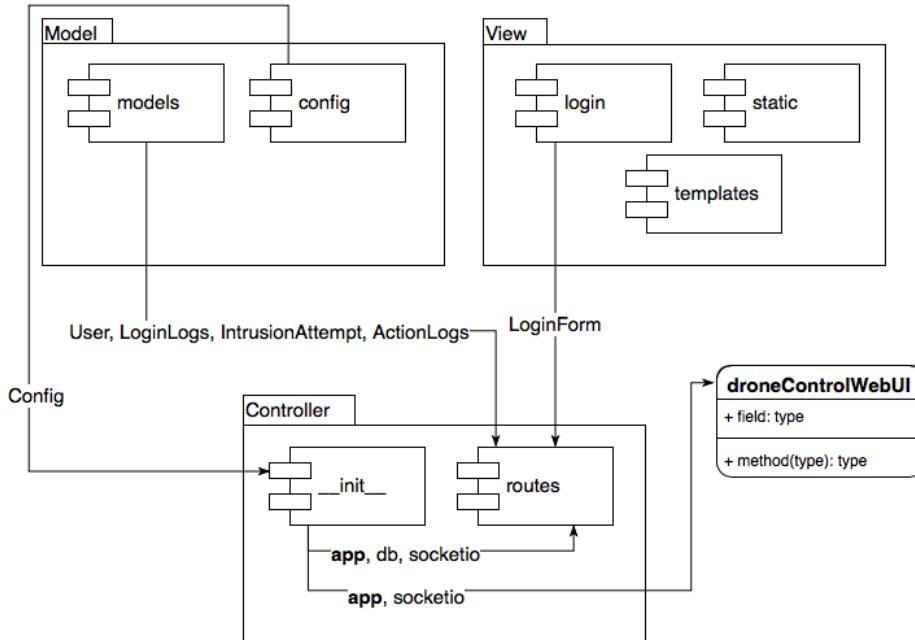
- **controller.__init__**: Establece una serie de opciones necesarias para el arranque de la aplicación web. Rutas de archivos, instancia la

aplicación en sí, establece la configuración, la base de datos, el servicio de login y la capa de sockets utilizada para comunicar la interfaz de usuario de forma asíncrona con la aplicación.

- **controller.routes**: Se trata de la clase encargada del control de la aplicación web. Responde a las solicitudes de los usuarios realizando las acciones correspondientes, y establece la comunicación necesaria entre el agente y el usuario.
- **model.config**: Se trata de una clase que contiene elementos de configuración de la aplicación web. Tal como la ubicación de la base de datos a utilizar, la clave secreta para generar los tokens de la aplicación, etc.
- **model.models**: Se trata de la definición de las entidades a utilizar por el ORM utilizado (SQLAlchemy).
- **model.migrations.env**: Se trata de un script generado por Alembic (una extensión para SQLAlchemy que permite realizar versionado de las bases de datos), y que permite actualizar, o revertir, los cambios realizados basándose en las diferentes versiones generadas.
- **model.migrations.versions.XXXXXX**: Se trata de los diferentes scripts que usará Alembic para actualizar o revertir los cambios realizados en la base de datos.
- **view.login**: Un formulario simple a utilizar por el controlador a la hora de obtener los datos de login.
- **frontend.droneControlWebUI**: El instanciador del servidor web. Se encarga de ejecutar la aplicación web mediante los parámetros especificados.

BackEnd

- **algorithms.Geometry**: Se trata de un módulo auxiliar utilizado por el Filtro de Partículas para realizar cálculos geométricos de forma vectorizada haciendo uso de Numpy.
- **algorithms.ParticleFilter**: Se trata de la implementación del Filtro de Partículas. Se encarga de obtener la posición del agente en el plano.

Figura C.8: Diagrama de clases del *frontend*.

- **algorithms.VFH**: Se trata de la implementación del sistema de evasión de obstáculos.
- **autoControllers.abcControllerPID**: Se trata de una metaclasa³ que define el contrato que deben subscribir las clases que implementen controladores PID.
- **autoControllers.altitudeController**: Se trata de una implementación de la metaclasa abcControllerPID, destinada al control de altitud.
- **autoControllers.inclinationController**: Se trata de una implementación de la metaclasa abcControllerPID, destinada al control de inclinación.
- **autoControllers.yawController**: Se trata de una implementación de la metaclasa abcControllerPID, destinada al control de la orientación.
- **autoControllers.obsAvoidanceWrapper**: Se trata de una clase implementada para encapsular los algoritmos de evasión de obstáculos, y

³En Python las interfaces se denominan *metaclasses*, aunque las capacidades de estas son más extendidas, servirá para referirnos a ellas de aquí en adelante.

generar el cálculo del control de orientación e inclinación para alimentar los controladores PID correspondientes.

- **autoControllers.takeOffLanding:** Se trata de una clase implementada para encapsular el controlador de altitud. Permite tanto el despegue como el aterrizaje, y genera el cálculo del control de altitud para alimentar al controlador PID correspondiente.
- **comms.MultiWiiProtocol:** Esta es la implementación del protocolo MultiWiiSerial Protocol. Dicho protocolo es utilizado por las herramientas de configuración de las controladoras de vuelo de los *drones* actuales, que ejecutan un firmware basado en el controlador Multi-Wii. Se encarga de llevar a cabo la comunicación entre el *drone* y la RaspberryPi, permitiendo el flujo de información en ambas direcciones.
- **comms.RemoteControl:** Se trata del servidor encargado de recibir las órdenes remotas del usuario.
- **sensors.sr04Wrapper:** Se trata de una pequeña clase que encapsula el acceso a los sensores de ultrasonidos utilizados para el cálculo de distancias, para el sistema de evasión de obstáculos.
- **sensors.Sensor:** Se trata de una pequeña clase que representa un sensor de ultrasonidos, incluyendo el acceso a la instancia de la clase *Echo* (perteneciente a la librería *Bluetin_Echo*), así como el ángulo en el que está orientado el sensor, y los pines a los que está conectado.
- **systemControl.ctrlWrapper:** Se trata de la implementación del sistema de control del *drone*. Se posiciona como un mecanismo de separación entre el *drone* y sus sensores, y los diferentes controladores. Se encarga de coordinar, de forma priorizada, los diferentes mecanismos de control del agente, así como de suministrar a aquellos que así lo requieran la información de los diferentes sensores a los que tiene acceso.

Se dispone de un diagrama de clases del *backend* en la figura C.9. Se han añadido colores para tratar de aumentar la claridad del diagrama, de por sí algo complejo.

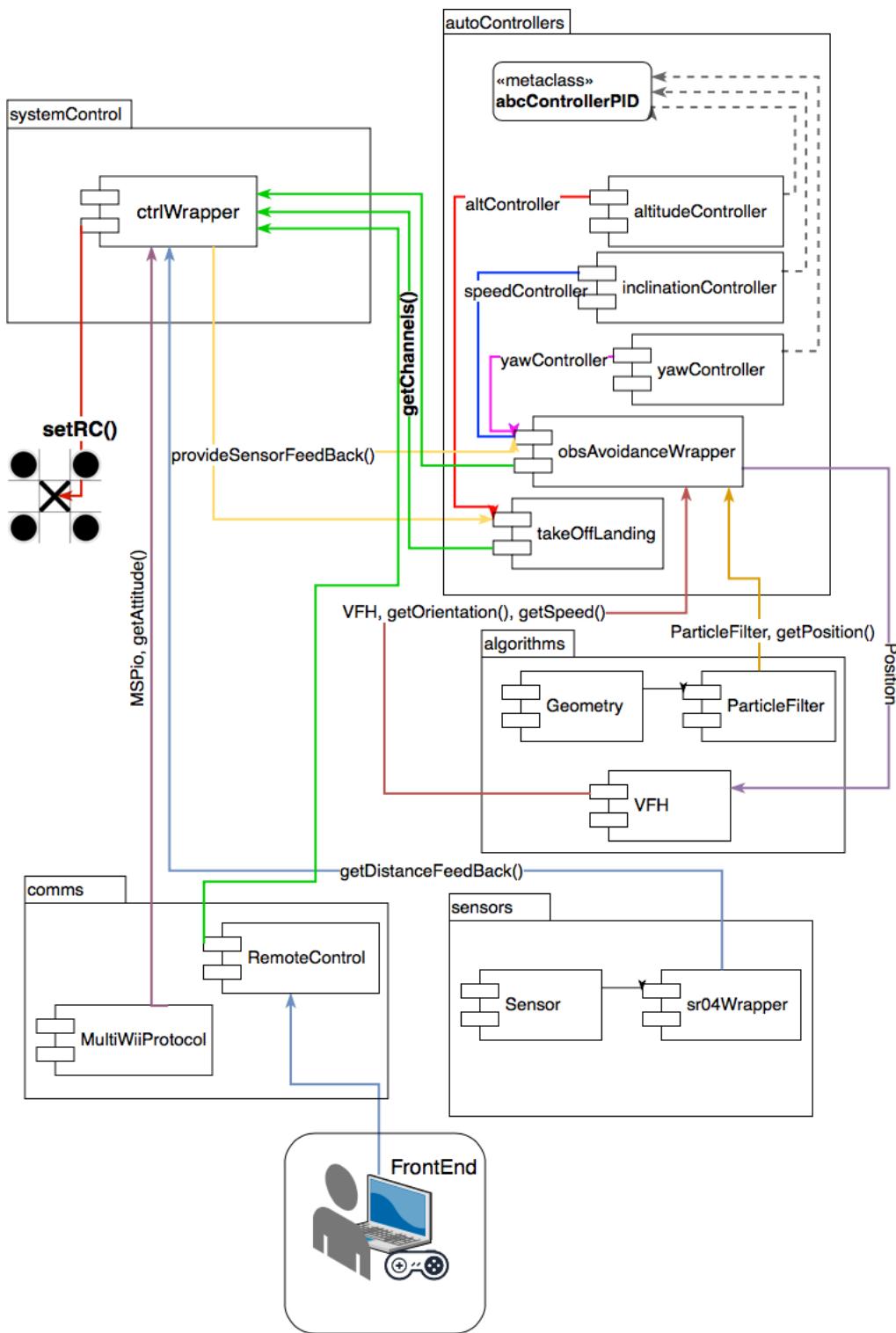


Figura C.9: Diagrama de clases del *backend*.

Apéndice D

Documentación técnica de programación

D.1. Introducción

En esta sección se describirá la documentación técnica de programación del proyecto. Se incluye la estructura de directorios utilizada, la instalación y ejecución del proyecto, así como las pruebas realizadas.

D.2. Estructura de directorios

- **/**: Directorio raíz. Contiene los ficheros de configuración de la instalación mediante *setuptools*, un script de instalación rápida y el fichero README con información sobre el proyecto.
- **/backend/**: Contiene los diferentes paquetes con el código fuente de la aplicación que será ejecutada en el sistema de control del *drone*.
- **/docs/**: Contiene la documentación del proyecto.
- **/frontend/**: Contiene los diferentes paquetes con el código fuente de la aplicación que será ejecutada en el servidor web.
- **/test/**: Contiene algunos test generados para comprobar el funcionamiento de las clases, así como la implementación del algoritmo de campos potenciales, que podría establecerse como *deprecated* en favor del VFH.

D.3. Manual del programador

Para trabajar con el proyecto es necesario tener instalado el siguiente software:

- JetBrains PyCharm IDE.
- Git.
- Un cliente de escritorio para Git (recomendable pero no indispensable), como GitKraken.
- Una shell capaz de ejecutar un script bash. Windows dispone de CygWin, para sistemas operativos con base Unix esta está disponible por defecto.
- Python 3.5+.

El resto de dependencias, referenciadas a continuación, serán instaladas mediante el script de instalación, ver sección [D.4](#).

- **BackEnd:**

- Numpy
- SciPy
- PySerial
- Bluetin_Echo

- **FrontEnd:**

- Flask-Login
- Flask-WTF
- Flask-SQLAlchemy
- Flask-Migrate
- Flask-SocketIO
- Eventlet
- Flask

Además, si se desea llevar a cabo pruebas reales del sistema, es necesario disponer del siguiente hardware:

- Raspberry Pi 3B o similar con sistema operativo Raspbian Stretch.
- Una cámara compatible con RaspberryPi. En este desarrollo se ha hecho uso de una PiNoIR, por no tener filtro de infrarrojos, de manera que se puede usar en momentos de baja luminosidad.
- Una controladora de vuelo compatible con *BetaFlight*¹ o similar.
- 6x Sensores de distancia por ultrasonidos *HC-SR04*.
- Regulador de voltaje para no exceder el voltaje máximo de entrada en la RaspberryPi.
- LEDs infrarrojos.
- Un *drone* sobre el que montar los dispositivos.
- Una emisora compatible con el sistema operativo, y reconocible por este como un joystick. En este desarrollo se ha empleado una emisora *FrSky Taranis X9D Plus*.

A continuación se detalla como instalar y configurar los requerimientos arriba descritos:

RaspberryPi y Raspbian

Este apartado se encuentra en primer lugar, dadas las múltiples referencias al acceso a la RaspberryPi que se darán a continuación.

Para conseguir la última imagen de instalación de Raspbian se puede acceder a [7]. En este proyecto se ha utilizado la imagen *lite* del sistema, que no dispone de interfaz de usuario, y es más ligera. (Extraido del documento de memoria, apartado 3):

Una vez descargado el sistema, este ha sido instalado en una micro SD mediante el siguiente procedimiento por consola²:

- `diskutil list` Permite localizar el dispositivo en el que se encuentra la tarjeta. En nuestro caso `/dev/disk4`
- `diskutil umountDisk /dev/disk4` Permite desmontar el volumen.

¹Firmware que se ejecutará en la controladora de vuelo. Dispone de una aplicación para configurar la misma.

²Realizado en OSX, aunque en Linux es muy similar

- `sudo dd if=raspbian-stretch.img of=/dev/rdisk4 bs=1m` El comando `dd` copia la entrada estándar a la salida estándar. Mediante `if/of` se establece el fichero de entrada/salida. Mediante `bs` se establece el tamaño de bloque a copiar. Se está utilizando `/dev/rdisk4` en lugar de `/dev/disk4` debido a la capacidad de OSX de trabajar con dispositivos en bruto, *raw*, de forma que es posible acceder al dispositivo de forma directa³, sin almacenar en un buffer la lectura del archivo, proporcionando velocidades de escritura/lectura hasta 20 veces más rápidas.

JetBrains PyCharm IDE

El IDE desarrollado por JetBrains es posiblemente el más avanzado para la creación de aplicaciones en Python. Se puede descargar desde [8], escogiendo el sistema operativo, y su instalación se basa en seguir el asistente de instalación.

Sin embargo, si se va a trabajar con una RaspberryPi, es conveniente configurar esta como destino del despliegue del *backend*, de manera que se pueda realizar el depurado de las aplicaciones de forma remota, así como su despliegue.

Accediendo a las preferencias del IDE, en el apartado *Build, Execution, Deployment* y bajo el subapartado *Deployment*, se puede establecer la configuración del dispositivo sobre el que realizar el despliegue, tal y como se muestra en la figura D.1.

Conviene configurar el acceso a la RaspberryPi mediante claves públicas. Para el detalle de como configurar el acceso, y generar las claves públicas ver la subsección D.3.

Se debe configurar también el intérprete remoto para la ejecución y *debug* de aplicaciones. Puede hacerse en PyCharm accediendo a las preferencias del IDE, en el apartado *Project: GII_0_17.02_SNSI* y bajo el subapartado *Project Interpreter*, se puede establecer la configuración del dispositivo sobre el que realizar el despliegue, tal y como se muestra en la figura D.2.

Se especificaría el host al que conectarse via SSH, tal y como se muestra en la imagen D.3.

³Véase `man hdiutil`, sección *DEVICE SPECIAL FILES*

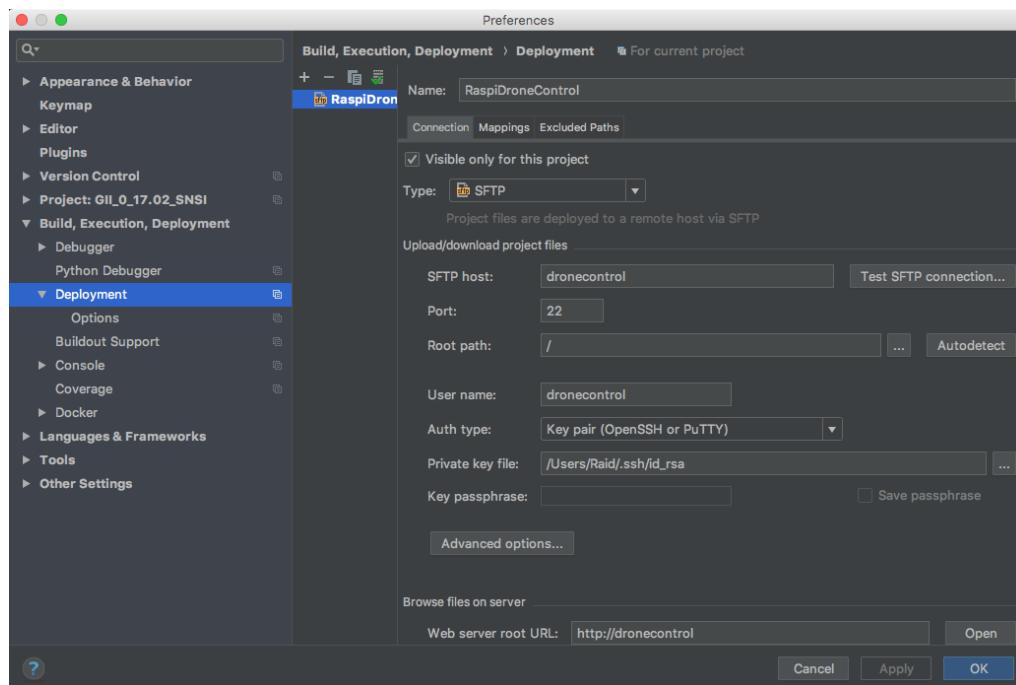


Figura D.1: Configuración del despliegue remoto en PyCharm.

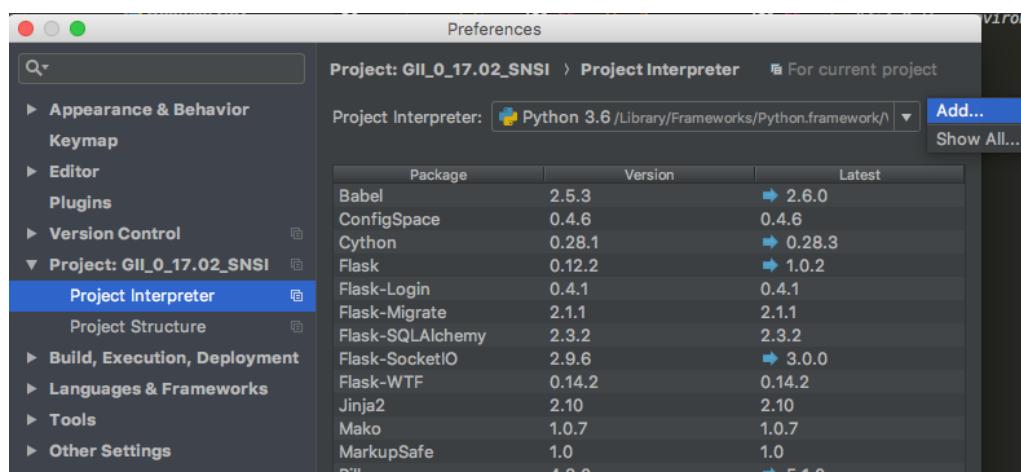


Figura D.2: Configuración del intérprete remoto en PyCharm.

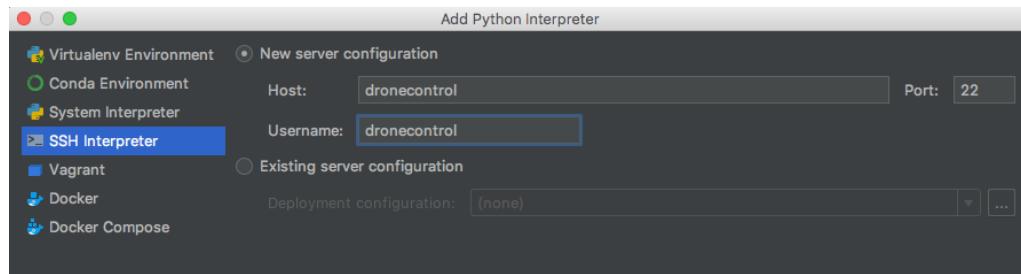


Figura D.3: Configuración del intérprete remoto en PyCharm vía SSH.

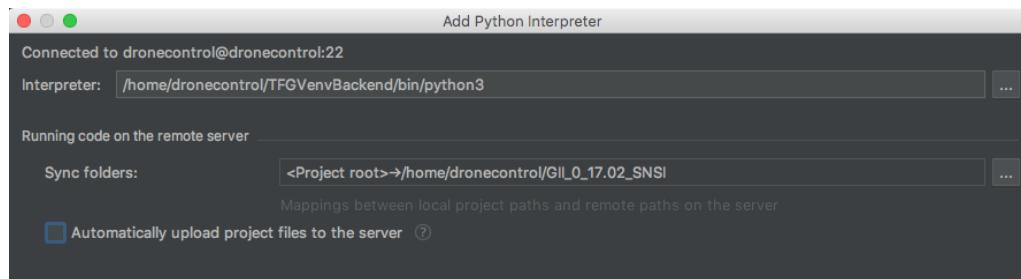


Figura D.4: Configuración del intérprete remoto en PyCharm vía SSH en un *Virtual Environment*.

Y por último, se selecciona el intérprete existente en el entorno virtual generado mediante el script de instalación, ver subsección D.4, y la correspondencia de carpetas, como se muestra en la imagen D.4.

Configuración SSH

Como añadido al anexo, aunque se encuentra disponible en la sección 3 de la memoria, se generan las claves públicas a usar en PyCharm para acceder a la RaspberryPi de forma remota, siguiendo los siguientes pasos:

- Generar el par de claves pública-privada en el cliente mediante el comando `ssh-keygen`
- Copiar la clave pública del cliente en el archivo `.ssh/authorized_keys` de la carpeta `home` del usuario a utilizar en el sistema remoto.
- Editar el archivo `/etc/ssh/sshd_config` para:
 - permitir únicamente acceso a usuarios que envíen una clave pública contenida en `authorized_keys`.

- desactivar el acceso al usuario root, estableciendo la propiedad `PermitRootLogin` a `no`.
- desactivar el acceso por contraseña, estableciendo la propiedad `PasswordAuthentication` a `no`.

Installación de Git

Para hacer uso de las capacidades de un sistema de control de versiones, se utilizará Git. Se puede obtener desde [9]. Aunque para comodidad, se hará uso de un entorno de escritorio, ver subsección D.3.

Installación de GitKraken

GitKraken es un entorno de escritorio para Git. Tiene integración con GitHub y permite hacer una gran cantidad de tareas de forma muy cómoda. Puede obtenerse a través de [10], y su instalación se basa en seguir el asistente, así como la configuración de acceso al repositorio local, y remoto.

Conexión de Hardware

Si se desea probar el proyecto, y desplegarlo en una RaspberryPi con acceso a todos los sensores, se deberían considerar los siguientes pasos:

- Arrancar la RaspberryPi con la imagen grabada, tal y como se explicó en la subsección D.3 y SIN conectar los sensores⁴.
- Una vez arrancado el sistema, se puede acceder a él vía SSH. Se deberá dar acceso al usuario al puerto serie, ya que, por defecto, no lo tiene. Para ello, se debe añadir al usuario al grupo `dialout`.
- Se puede «limpiar» el estado de los pines mediante un simple script de Python, importando `RPi.GPIO` y llamando al método `cleanup`.
- Para conectar la RaspberryPi a la controladora de vuelo, se puede usar un puerto USB, un cable con un extremo USB-A y el otro *microUSB*.
- El diagrama de conexión de los sensores de ultrasonidos puede verse en la figura D.5.

⁴Es muy importante no conectar los sensores hasta estar 100 % seguros de que no existe ningún pin del componente GPIO de la RaspberryPi enviando un voltaje a un periférico conectado a él. Esto podría causar daños en los sensores si el voltaje es aplicado al pin de salida en lugar de al pin de entrada.

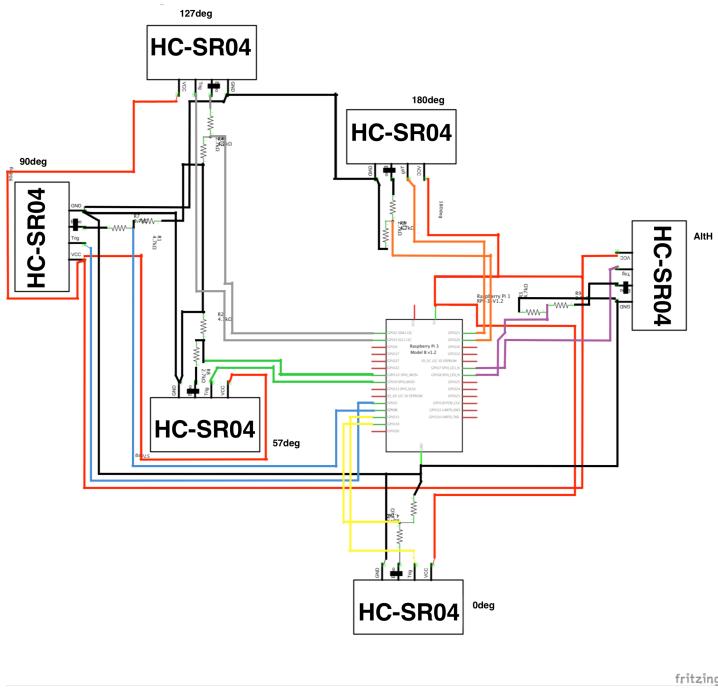


Figura D.5: Vista esquemática del conexionado de los sensores a la RaspberryPi.

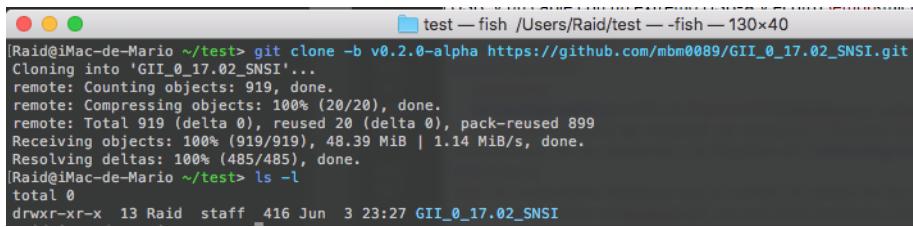
- Es importante destacar que el voltaje de salida de los sensores es 5V, y que, por lo tanto, puede dañar la RaspberryPi, ya que esta funciona a 3,3V. Por ello, se ha de hacer uso de algún elemento para reducir el voltaje o la corriente suministrada. En la vista esquemática proporcionada, existen divisores de voltaje que permiten reducir este voltaje.
- La emisora puede conectarse a un ordenador, y si este la detecta como un *joystick*, puede utilizarse para controlar el *drone* a través de la interfaz web. Para ello es necesario configurar la emisora para que proporcione la salida de los canales a través de la conexión USB.

Obtención del código fuente

Para el desarrollo del proyecto se ha utilizado un repositorio Git local, al que se ha establecido un origen remoto en GitHub. Para clonar el proyecto y poder utilizarlo deben seguirse los siguientes pasos:

1. Abrir un terminal bash.

D.4. COMPILACIÓN, INSTALACIÓN Y EJECUCIÓN DEL PROYECTO



```
[Raid@iMac-de-Mario ~/test> git clone -b v0.2.0-alpha https://github.com/mbm0089/GII_0_17.02_SNSI.git
Cloning into 'GII_0_17.02_SNSI'...
remote: Counting objects: 919, done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 919 (delta 0), reused 20 (delta 0), pack-reused 899
Receiving objects: 100% (919/919), 48.39 MiB | 1.14 MiB/s, done.
Resolving deltas: 100% (485/485), done.
[Raid@iMac-de-Mario ~/test> ls -l
total 0
drwxr-xr-x 13 Raid staff 416 Jun 3 23:27 GII_0_17.02_SNSI
```

Figura D.6: Clonado del repositorio mediante terminal.

2. Moverse hasta el directorio en el que se quiera clonar el repositorio. Para ello puede utilizarse el comando `cd`.
3. Clonar el repositorio mediante el comando:
`git clone git@github.com:mbm0089/GII_0_17.02_SNSI.git` para clonar la rama *master*. Si se desea trabajar en alguna de las ramas en desarrollo, puede usarse el comando
`git clone -b NOMBRE_RAMA git@github.com:mbm0089/GII_0_17.02_SNSI.git`
4. Con esto se iniciará la descarga del repositorio completo.

D.4. Compilación, instalación y ejecución del proyecto

La siguiente sección describirá como instalar las dependencias necesarias, compilar el código y ejecutar el proyecto.

Instalación

El proceso de instalación de dependencias se ha diseñado para que sea totalmente automatizado. Bastará con ejecutar el script llamado *installer.sh* de la siguiente manera:

1. Abrir una terminal bash.
2. Desplazarse hasta la carpeta del proyecto. Puede usarse el comando `cd` para ello.
3. Dar permisos de ejecución al instalador mediante `chmod +x installer.sh`.

4. Ejecutar el instalador usando como parámetros el nombre del entorno virtual⁵ a crear, y el sistema a instalar (bien el *backend* o el *frontend*, tal que: `./installer.sh backendVenv setupBackend.py` , o `./installer.sh frontendVenv setupWebUI.py` .
5. Con esto comenzará la instalación de las dependencias necesarias para la ejecución del proyecto.

Una vez instalado el entorno virtual, este puede configurarse en PyCharm tal y como se explicó en la sección [D.3](#).

Compilación

La compilación del proyecto es una tarea transparente para el desarrollador. Es el propio intérprete quien se encarga de realizar la compilación de las diferentes clases o scripts en el momento de su ejecución.

Ejecución del proyecto

Ejecutar el proyecto a través del IDE es un proceso trivial. Una vez establecido los entornos virtuales a utilizar, tanto en local como el entorno remoto para el despliegue del *backend*, bastará con realizar los siguientes pasos:

- En el caso de la aplicación web, en la raíz de la carpeta `/frontend/` se encuentra una clase llamada `droneControlWebUI.py`. Si se ejecuta esta clase, se lanza el servidor web, y se carga la aplicación web creada. Se puede configurar los parámetros del servidor modificando los valores disponibles en la clase de forma muy sencilla.
- El caso del *backend*, o el código que se ejecuta en la RaspberryPi, es similar. Si se ha seguido el proceso de creación de un intérprete remoto, ver [D.3](#), éste puede ser seleccionado para ejecutar las clases del proyecto. La clase que debe ejecutarse para lanzar todo el sistema de control del *drone* se encuentra en la carpeta `/backend/systemControl`, y tiene el nombre de `ctrlWrapper.py`. De esta manera, bastará con establecer la configuración de ejecución de este script para que utilice el intérprete remoto.

⁵Generalmente, en los proyectos desarrollados en Python, no se instalan las dependencias sobre la instalación base de Python, sino que se usan entornos virtuales. En ellos se crea una copia del intérprete de Python, y se instalan las librerías adicionales sin afectar la instalación del sistema

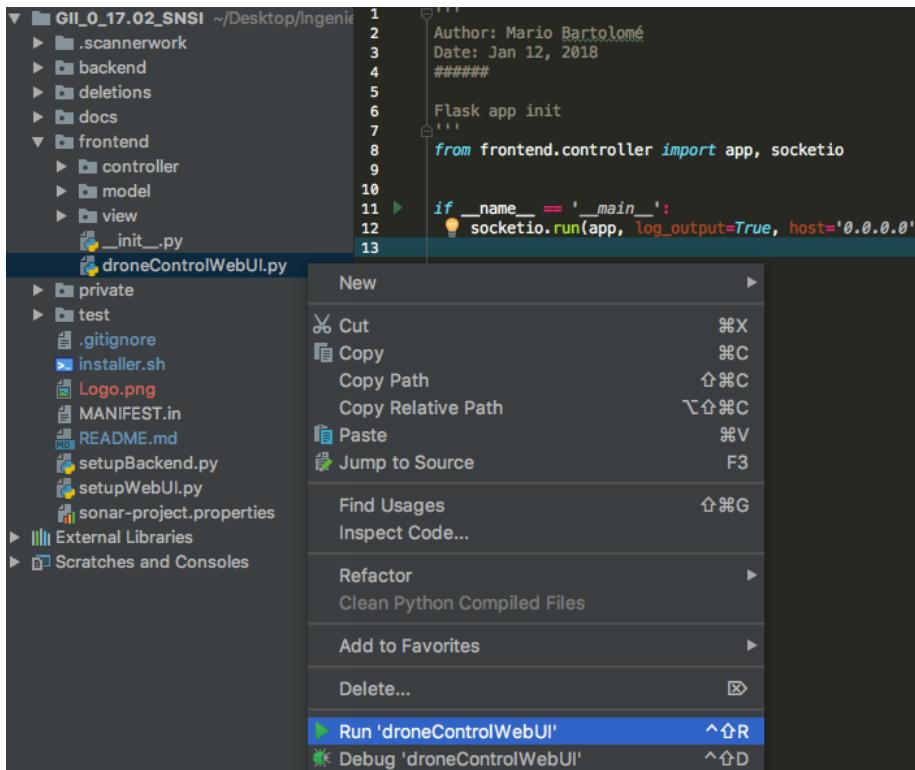
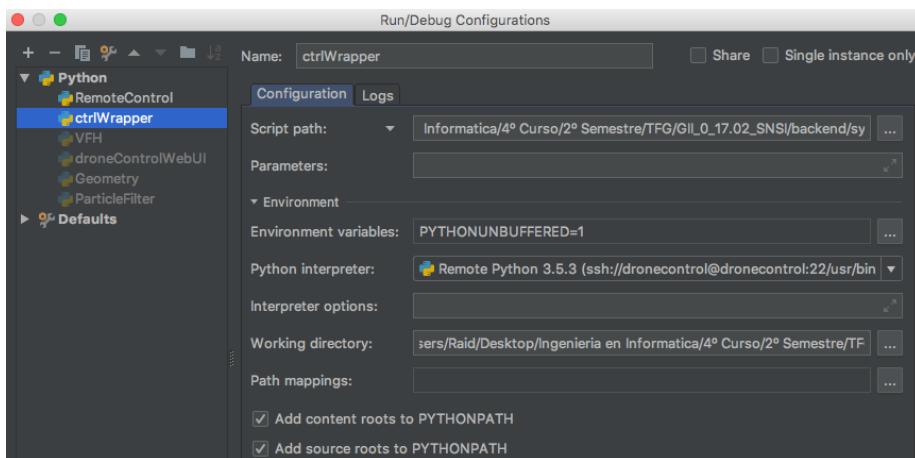


Figura D.7: Ejecución de la aplicación web.

Figura D.8: Ejecución del sistema de control del *drone*.

```
Raid@Mac-de-Mario ~> droneControl
#####
#      All connections are monitored and recorded
#
#      Disconnect IMMEDIATELY if you are not
#          an authorized user
#
#####
Linux UBUDroneSereno 4.9.80-v7+ #1098 SMP Fri Mar 9 19:11:42 GMT 2018 armv7l
Welcome to UBUDroneSereno shell control system.

WARNING! PRIOR TO MAKE ANY CHANGES, MAKE SURE DRONE IS IDLE AND ITS SURROUNDINGS ARE CLEAR.

Last login: Sun Jun  3 22:01:18 2018 from 192.168.1.128
dronecontrol@UBUDroneSereno ~> bash
dronecontrol@UBUDroneSereno:~ $ cd GII_0_17.02_SNSI/
dronecontrol@UBUDroneSereno:~/GII_0_17.02_SNSI $ source ./TFGVenvBackend/bin/activate
(TFGVenvBackend) dronecontrol@UBUDroneSereno:~/GII_0_17.02_SNSI $ export PYTHONPATH=$(pwd)
(TFGVenvBackend) dronecontrol@UBUDroneSereno:~/GII_0_17.02_SNSI $ python3 -m backend.systemControl.ctrlWrapper
RemoteControl server ready to start
Awaiting clients.
Port /dev/ptyUSB0 successfully opened
```

Figura D.9: Ejecución del proyecto desde un terminal bash.

Si se quisiese ejecutar el proyecto desde una terminal, es necesario activar el entorno virtual creado, y de establecer una variable de entorno. Bastará con realizar los siguientes pasos:

1. Abrir un terminal bash.
2. Activar el entorno virtual a utilizar. `source EntornoVirtual/bin/activate`.
3. Establecer la variable de entorno *PYTHONPATH* a la raíz del proyecto. Este paso PyCharm lo hace por defecto. `export PYTHONPATH=$(pwd)` .
4. Ejecutar el script deseado desde la raíz del proyecto.
`python3 -m backend.systemControl.ctrlWrapper` o
`python3 -m frontend.droneControlWebUI`

Añadir nuevos controladores al sistema de control

Añadir nuevos controladores al sistema de control es relativamente sencillo. Para ello se deben seguir los siguientes pasos, teniendo en cuenta el flujo de trabajo de Git. Cabe destacar que durante la realización de este proyecto, y con la intención de simplificar el flujo de trabajo, dado que solo existe un desarrollador, el flujo de trabajo se ha simplificado creando únicamente ramas para las *release* generadas:

1. Crear una nueva rama (*feature branch*) desde la rama de la *release* en la que se está trabajando:
`git checkout -b new-controller RAMA_ACTUAL`
2. Crear una clase para el *controller* a generar: `improvedSpeedController.py`, por ejemplo, que implemente los métodos necesarios. Hacer *commit* del nuevo fichero y la lógica generada.
3. Refactorizar el código para mejorar su calidad. Hacer *commit*.
4. Incluir el nuevo controlador generado en el `ctrlWrapper.py`. Para ello solo se deben proporcionar los métodos requeridos por el controlador principal.
5. Una vez que se ha implementado correctamente el nuevo controlador, se debe sincronizar en la rama del nuevo desarrollador, anteriormente creada, mediante `git push`, y seguidamente incorporar esta rama a la rama de la *release* en desarrollo, y existente en GitHub. Para ello, lo ideal es generar una *pull-request* estableciendo la necesidad de una revisión, pero, como se ha comentado, solo existe un desarrollador, de manera que en el desarrollo se ha simplificado este paso:
`git request-pull nombre1erCommit origin new-controller`⁶

De esta forma se generará una *pull-request* en el repositorio remoto hospedado en GitHub.

Sin embargo, y por comodidad, se ha hecho uso de un cliente de escritorio, que facilita enormemente el flujo de trabajo de Git, de forma que crear una *pull-request* de un *commit* concreto es tan sencillo como dar click derecho sobre el *commit* deseado y pulsar sobre *start a pull-request to origin/ from origin/v0.2.0*, tal y como se muestra en la Figura D.10.

Programación funcional VS Interfaces y diseño por contrato

Se hace notar que la clase `ctrlWrapper`, encargada del control de todo el sistema, podría haber definido una metaclasa, o interfaz, que los controladores a crear pudiesen implementar. Sin embargo, en aras de la programación funcional, y para explotar la flexibilidad de Python, se ha dejado a la discreción del desarrollador el nombrado, firma, e implementación de las diferentes funciones.

⁶El origen del proyecto se habrá establecido a https://github.com/mbm0089/GII_0_17.02_SNSI/

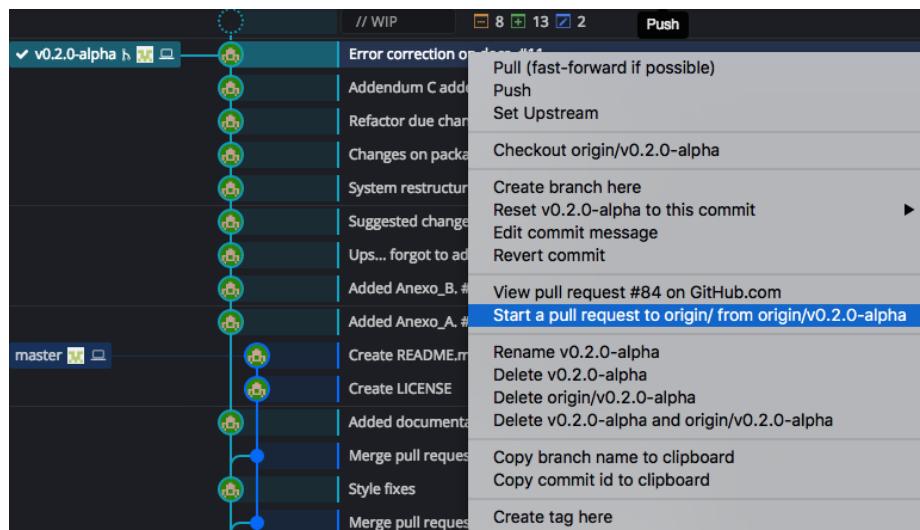


Figura D.10: Creación de una *pull-request* desde el cliente de escritorio GitKraken.

De esta manera, añadir un nuevo controlador a la lista de controladores priorizados que genera el *ctrlWrapper*, requiere pasar como parámetro las funciones requeridas.

Servicios de integración continua

Para llevar a cabo una toma de medidas del código generado, se ha hecho uso de una suscripción a *SonarCloud*. Se trata de un servicio proporcionado por *SonarCloud*, de forma que no es necesario instalar un servidor local, sino que se realiza el escaneo del proyecto y los resultados se envían al servicio en línea de *SonarCloud*.

El escáner a utilizar, se puede descargar de [11].

Para ello, se deben seguir los siguientes pasos:

1. Abrir un terminal bash.
2. Desplazarse hasta la carpeta del proyecto. Puede usarse el comando `cd` para ello.
3. Generar un fichero de configuración de los parámetros requeridos por el escáner de *SonarCloud*, haciendo uso del editor preferido, tal y como puede verse en la Figura D.11.

```

1 sonar.projectKey=GII_0_17.02_SNSI
2 sonar.organization=mbm0089-github
3
4 sonar.sources=.
5 sonar.host.url=https://sonarcloud.io
6 sonar.login=4c7*****b198*****77
7 sonar.cfamily.build-wrapper-output.bypass=true
8 sonar.sources=./backend,./frontend,./test
9

```

Figura D.11: Creación de un fichero de configuración para el servicio online de *SonarCloud*.

Se deberá establecer la clave del proyecto, mediante el atributo **projectKey**. Se deberá establecer la organización del proyecto, mediante el atributo **organization**. Se deberá establecer la ubicación de las fuentes a utilizar, mediante el atributo **sources**. Se deberá establecer el host al que conectarse (el escáner es de *SonarQube*, así que pueden enviarse sus resultados a una instancia de un servidor *SonarQube* local), mediante el atributo **host.url**. Se deberá establecer un *token* para el login, el cual es generado en la creación del proyecto en *SonarCloud*, mediante el atributo **login**. Por último, se establecerán los directorios a tener en cuenta, mediante el atributo **sources**.

4. Ejecutar el escáner de *SonarQube* tal que:

```
sonar-scanner -Dproject.settings=./sonar-project.properties7
```

Seguidamente, entrando en *SonarCloud*⁸, pueden verse los resultados del último análisis, tal y como se muestra en la Figura D.12

D.5. Pruebas del sistema

Las pruebas de software se llevarán a cabo mediante PyCharm, el cual provee de sus propio *framework* de pruebas, que permite llevar a cabo un análisis del cubrimiento obtenido por los diferentes test ejecutados. Se ha creado una serie de pruebas que tratan de cubrir la mayor cantidad de código posible.

⁷Para ello es necesario que sonar-scanner se encuentre en el *PATH* de ejecución

⁸https://sonarcloud.io/dashboard?id=GII_0_17.02_SNSI

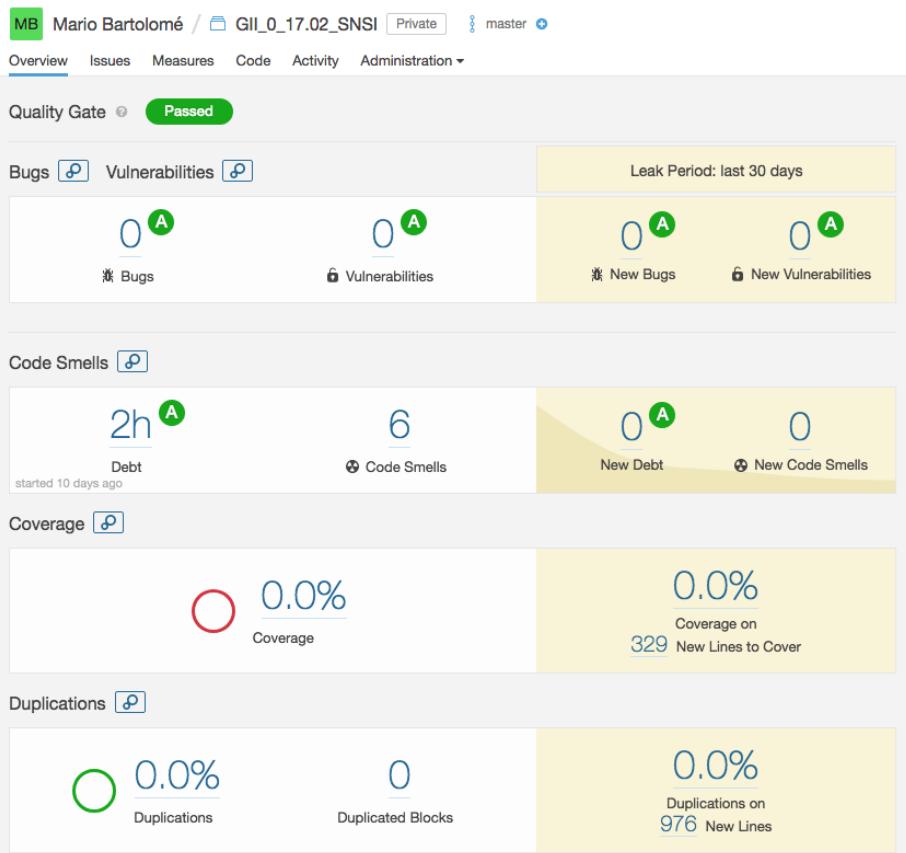


Figura D.12: Resultados del escaneo del proyecto en *SonarCloud*.

No se han creado pruebas para los métodos *setter* o *getter* del desarrollo. Sin embargo, el uso de test más complejos, obtiene resultados idénticos sin realizar específicamente ese tipo de test.

Configuración de los test

En primer lugar es conveniente establecer la configuración de los test. Desde PyCharm es posible ejecutar test en el intérprete local, así como en el intérprete remoto existente en el entorno virtual de la RaspberryPi.

Los test locales se pueden ejecutar de forma sencilla pulsando sobre la carpeta **testing** existente dentro de algunas de los paquetes que componen el sistema:

Tan solo con eso, ya se habrá generado la configuración de ejecución de test de forma local.

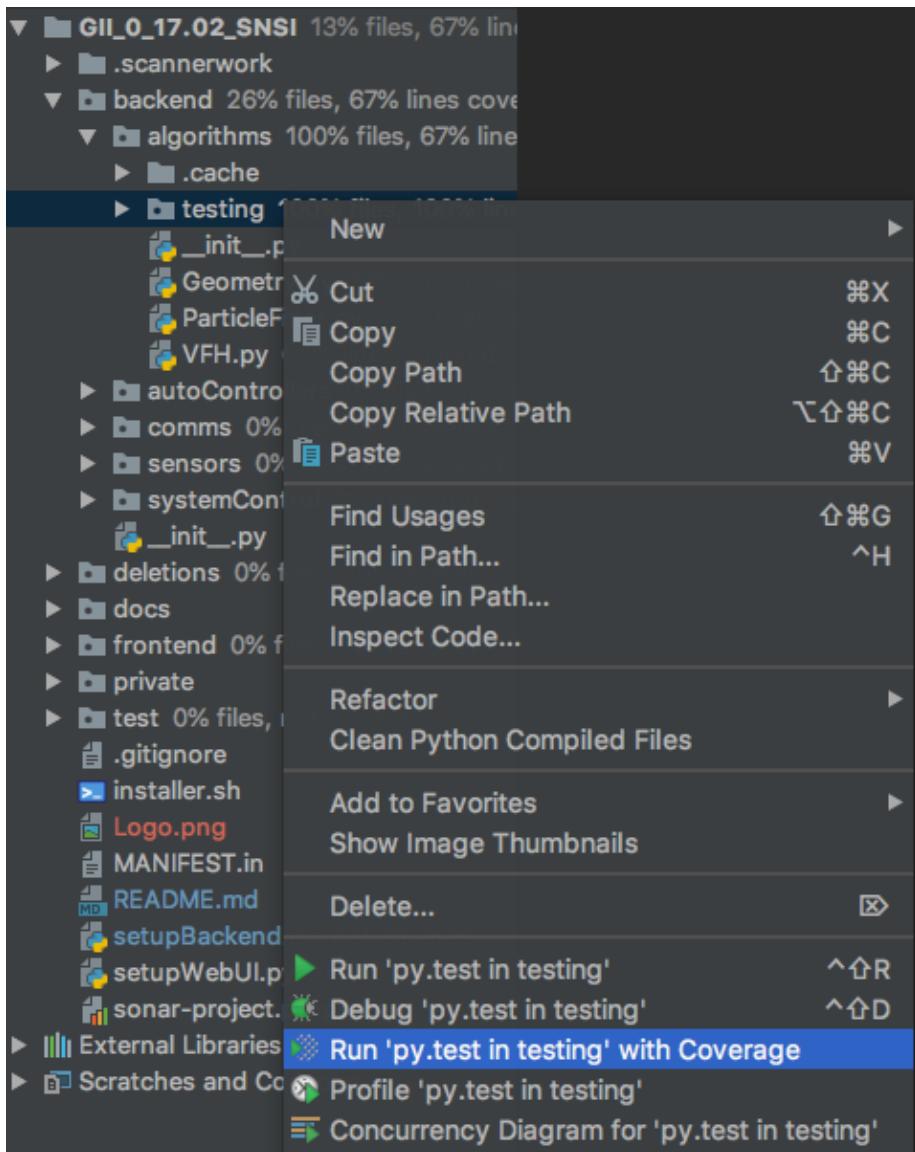


Figura D.13: Ejecución de test y métricas de cubrimiento de forma local.

El caso de los test a ejecutar en la RaspberryPi es algo más complejo. Éstos, requieren de crear las configuraciones para la ejecución de los diferentes test, estableciendo el intérprete remoto existente en el entorno virtual creado mediante los *scripts* de instalación, ver la subsección [D.4](#).

Para ello, se puede dar click derecho sobre la carpeta `testing` de alguna de las características que dependen de las librerías existentes en la RaspberryPi, como, por ejemplo, la contenida en `backend/comms/`. A continuación, se pulsa sobre *Create py.test in test...*, tal y como puede verse en la Figura [D.14](#).

Seguidamente, se debe establecer el nombre y la ubicación del intérprete remoto, tal y como se muestra en la Figura [D.15](#).

Una vez hecho esto, ya es posible ejecutar los test con la misma facilidad que si se tratase de test ejecutados mediante un intérprete local.

Pruebas Unitarias

Las pruebas unitarias comprueban el funcionamiento de una única clase o módulo, sin integrarse con otras.

Una vez ejecutada alguna de las pruebas, haciendo uso de las configuraciones explicadas en la subsección [D.5](#), se mostrarán los resultados de la siguiente forma:

- En el lado derecho de la aplicación, se mostrará el resultado del cubrimiento de los test ejecutados, tal y como se muestra en la Figura [D.16](#).
- En la zona inferior el resultado de dichos test, tal y como se muestra en la Figura [D.17](#).

Pruebas de integración

Sin duda las pruebas más interesantes, y complejas, son las de integración. Éstas, comprueban que la unión entre las clases o módulos se ha llevado a cabo sin producir errores. Es decir, las clases funcionan de forma individual, tal y como se comprobó con los test unitarios, y ahora hay que comprobar que se integran bien entre sí.

Para ello, se pueden ejecutar las pruebas contenidas dentro de la carpeta `/backend/comms` o `/backend/systemControl`. Ambas baterías de pruebas

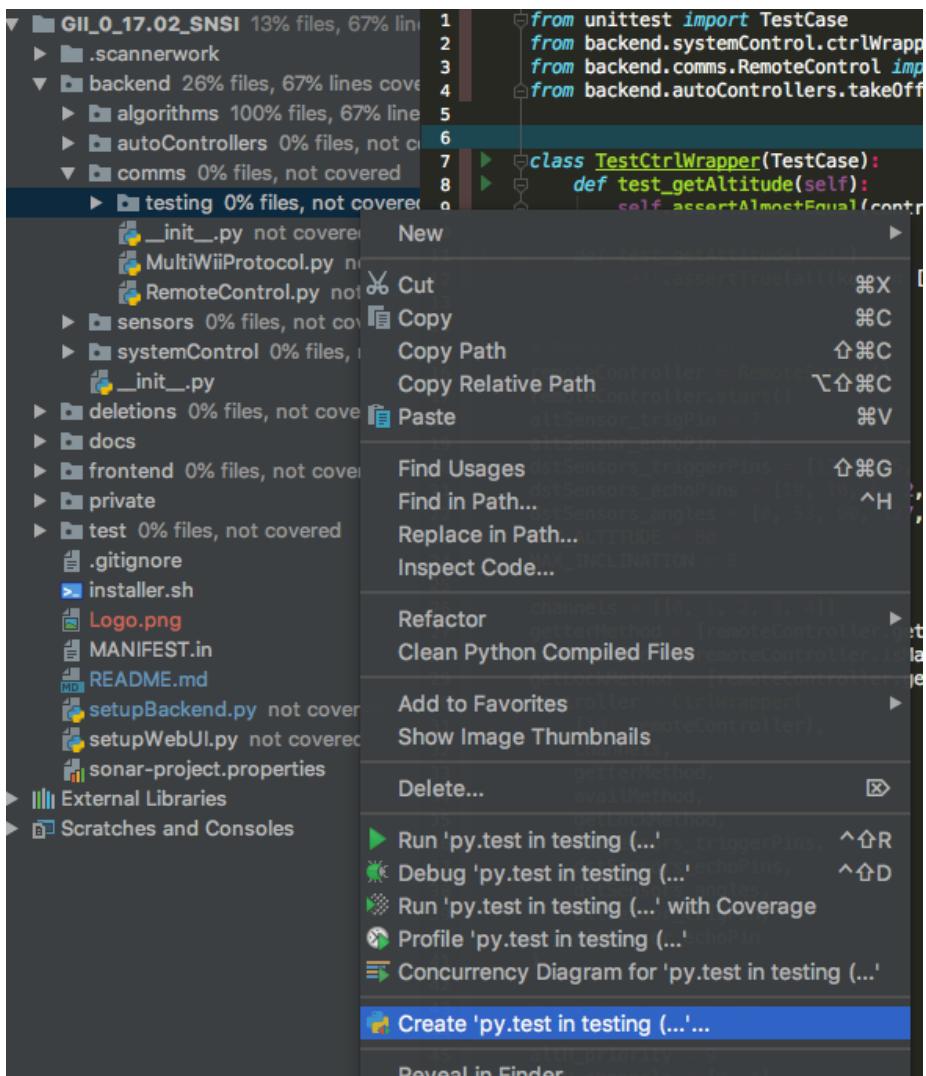


Figura D.14: Creación de la configuración para ejecutar test en un intérprete remoto.

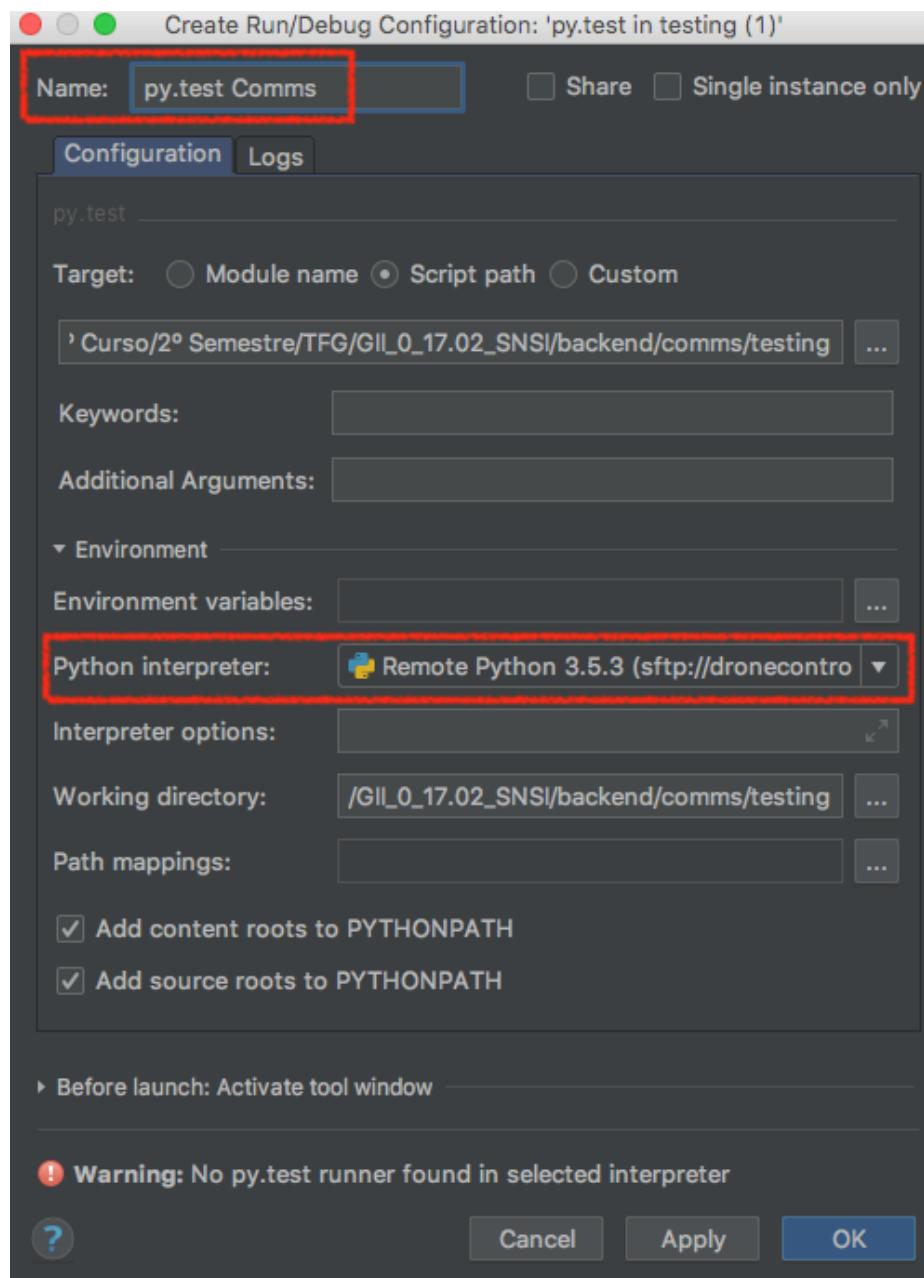


Figura D.15: Configuración para ejecutar test en un intérprete remoto.

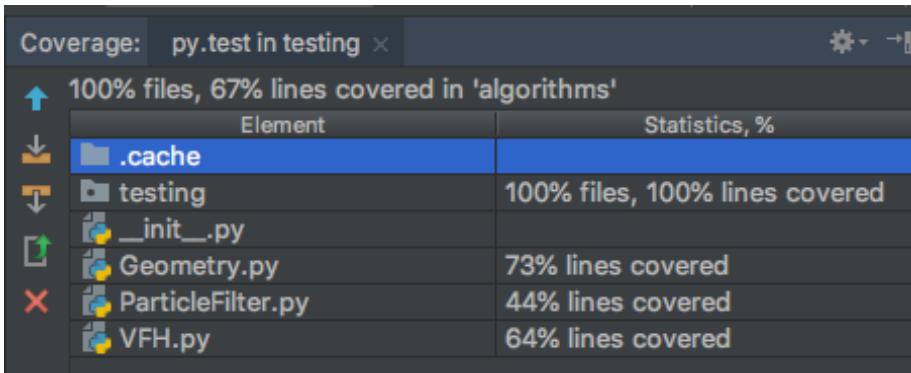


Figura D.16: Resultado de métricas de cubrimiento ejecutado de forma local.

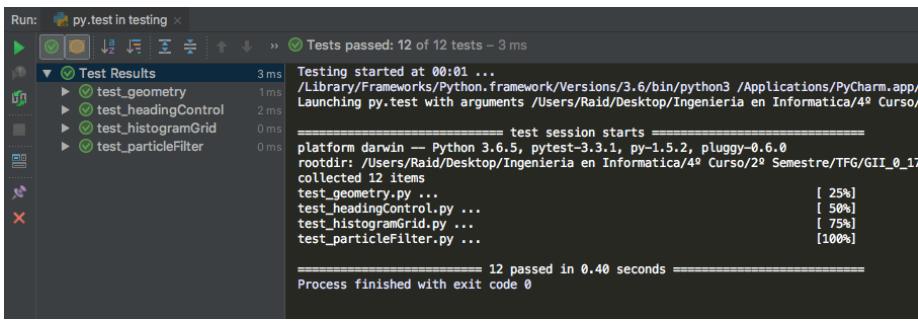


Figura D.17: Resultado de test ejecutado de forma local.

se deben ejecutar en el intérprete remoto, ya que requieren acceder a los sensores de distancia, así como a la controladora de vuelo.

Una vez ejecutadas, se mostrarán los resultados por pantalla.

Cabe destacar, que el funcionamiento de algunas de estas pruebas, sobre todo las basadas en sensores, están condicionadas al correcto funcionamiento de los sensores. Por ejemplo, las contenidas en `/backend/systemControl` tratarán de acceder al sensor de altitud, y comprobará que la altura del *drone*, respecto al suelo, no supera los 12 cm⁹. Un fallo en estos test, puede indicar que el sensor no se encuentra en condiciones óptimas de funcionamiento, o que alguna conexión ha fallado.

De igual manera, las pruebas existentes en `/backend/comms` accederán al puerto serie y establecerán comunicación con la controladora de vuelo, para comprobar que la comunicación con esta es correcta, tratando de leer el estado del *drone*. Un fallo en estos test, puede indicar que la controladora de

⁹El cálculo es aproximado dado el error existente en el sensor. Se admite un error de 1.5 cm.

vuelo no está respondiendo correctamente, que el cable USB que la conecta a la RaspberryPi está defectuoso, o que por algún motivo no se ha podido acceder al recurso.

Por ello, las pruebas de integración llevadas a cabo son susceptibles de revisión y sus resultados no deben tomarse, por sí solos, como definitivos.

Sistema de integración continua

Para llevar a cabo la comprobación continua del desarrollo del proyecto se ha hecho uso de SonarCloud. Tal y como se explicó en la subsección D.4, se dispone de un escáner que llevará a cabo una serie de procedimientos para comprobar las métricas de calidad del sistema en desarrollo.

Sin embargo, el escáner no incorpora ningún mecanismo, en su versión para Python, para llevar a cabo los test, y obtener métricas sobre el cubrimiento de estos.

Para ello, es necesario descargar una herramienta adicional que se integrará con SonarCloud. Dicha herramienta se llama *coverage*, y está disponible a través de *pip*, de forma que su instalación es muy sencilla. Basta con ejecutar `pip install coverage` dentro del entorno virtual del proyecto.

Una vez instalada, se pueden ejecutar las pruebas ejecutando los siguientes pasos:

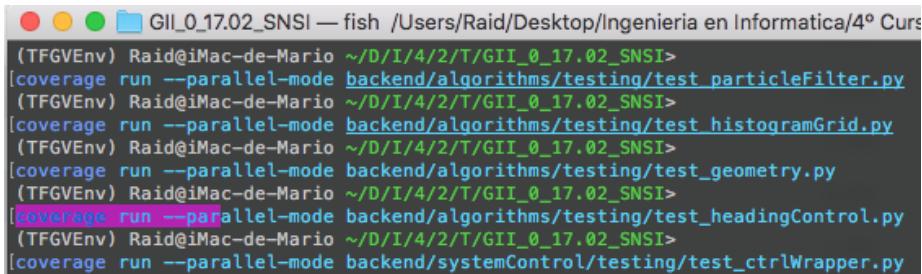
1. Abrir un terminal bash.
2. Desplazarse hasta la ruta del proyecto. Puede hacerse mediante `cd`.
3. Ejecutar el siguiente comando:

```
coverage run -parallel-mode -omit backend/algorithms/testing/test_*
    ↪ -omit backend/comms/testing/test_MSPio.py
    ↪ -omit backend/systemControl/testing/test_ctrlWrapper.py
    ↪ -omit frontend/model/migrations/*
    ↪ -omit backend/algorithms/testing/test_particleFilter.py
```

De esta forma, se habrá ejecutado el test perteneciente a la carpeta `/backend/algorithms/testing` y de nombre `test_particleFilter.py`.

Mediante los diferentes *omit* hemos ordenado a *coverage* que omita esos directorios, para que no trate de medir cuantas líneas de código del test han testado los propios test.

De forma análoga es necesario ejecutar todos los demás test existentes en las demás carpetas, tal y como se muestra en la Figura D.18.



```
(TFGVEnv) Raid@iMac-de-Mario ~/D/I/4/2/T/GII_0_17.02_SNSI>
[coverage run --parallel-mode backend/algorithms/testing/test_particleFilter.py
(TFGVEnv) Raid@iMac-de-Mario ~/D/I/4/2/T/GII_0_17.02_SNSI>
[coverage run --parallel-mode backend/algorithms/testing/test_histogramGrid.py
(TFGVEnv) Raid@iMac-de-Mario ~/D/I/4/2/T/GII_0_17.02_SNSI>
[coverage run --parallel-mode backend/algorithms/testing/test_geometry.py
(TFGVEnv) Raid@iMac-de-Mario ~/D/I/4/2/T/GII_0_17.02_SNSI>
[coverage run --parallel-mode backend/algorithms/testing/test_headingControl.py
(TFGVEnv) Raid@iMac-de-Mario ~/D/I/4/2/T/GII_0_17.02_SNSI>
[coverage run --parallel-mode backend/systemControl/testing/test_ctrlWrapper.py
```

Figura D.18: Uso de *coverage* para recabar métricas de test ejecutados en un intérprete local.

Sin embargo, existe un problema, y es que *coverage* no es capaz de ejecutar test haciendo uso de un intérprete remoto, por lo que las pruebas de integración existentes en el *backend*, y que dependan de hardware disponible en la *RaspberryPi*, no podrán ser llevadas a cabo mediante esta herramienta para ser incluidas en las métricas de SonarCloud.

4. Seguidamente se combinan los resultados de las pruebas mediante `coverage combine`.
5. Después se genera un reporte mediante `coverage report`.
6. Y finalmente se exporta mediante `coverage xml`

A continuación, se puede ejecutar el escáner de SonarCloud, pero antes es necesario añadir una nueva línea a su archivo de configuración. Para ello:

1. Abrir un terminal bash.
2. Desplazarse hasta la ruta del proyecto. Puede hacerse mediante `cd`.
3. Editar el archivo de configuración de SonarCloud mediante el editor de elección.
4. Añadir la siguiente línea:
`sonar.python.coverage.reportPath=coverage.xml`.
5. Guardar y cerrar el archivo.
6. Ejecutar el escáner de SonarCloud mediante
`sonar-scanner -Dproject.settings=./sonar-project.properties`

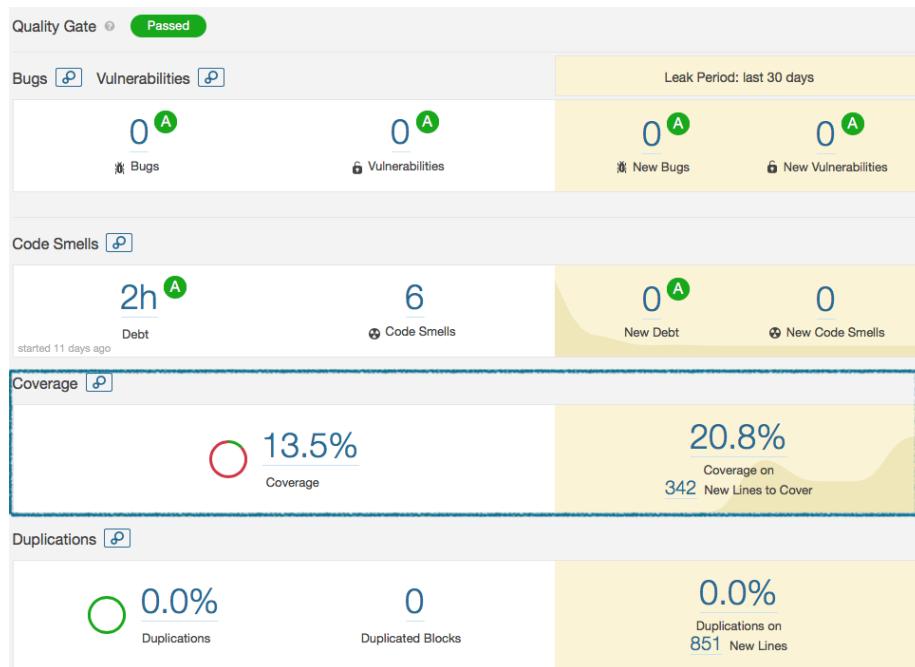


Figura D.19: Actualización de métricas de cubrimiento en SonarCloud mediante *coverage*.

Al poco, en el proyecto de SonarCloud, se verán las métricas actualizadas con la información del cubrimiento de los test realizados por *coverage*. Ver Figura D.19.

Apéndice E

Documentación de usuario

E.1. Introducción

En este apartado, se detallará un manual sobre como hacer uso del sistema implementado.

E.2. Requisitos de usuarios

Los requisitos mínimos para poder hacer uso del sistema completo son:

- Contar con un ordenador con el sistema operativo deseado.
- Disponer de un navegador Web. Se recomienda el uso de *Google Chrome*, *Mozilla Firefox* o *Safari*. Se ha probado el entorno en los tres navegadores. *Chromium* ha presentado problemas en el *parseo* del mensaje *SDP*¹ proveniente del servidor WebRTC que implementa *UV4L*, y no se recomienda. Se desaconseja por completo el uso de *Internet Explorer* o *Edge*.
- *Opcional.* Contar con una emisora detectable por el sistema operativo como un *joystick*, para poder hacer uso de las características de control remoto del sistema.
- Conexión a internet.

¹Session Description Protocol, detallado en la memoria técnica del proyecto

E.3. Instalación

El usuario final accederá a la aplicación a través de una interfaz web, así que no se requiere instalar ningún *software* adicional.

Sí que se detalla a continuación el proceso de conexión y configuración de la emisora para hacer uso del control remoto. Para el desarrollo del proyecto se ha utilizado una emisora *FrSky Taranis X9D plus*. Dicho modelo es tanto hardware, como código abierto.

1. Descargar OpenTX² y seguir el asistente de instalación.
2. Abrir OpenTX.
3. Encender la emisora pulsando la siguiente combinación de botones, los dos botones laterales se deben dirigir hacia dentro, y el botón de encender (central) hacia arriba:

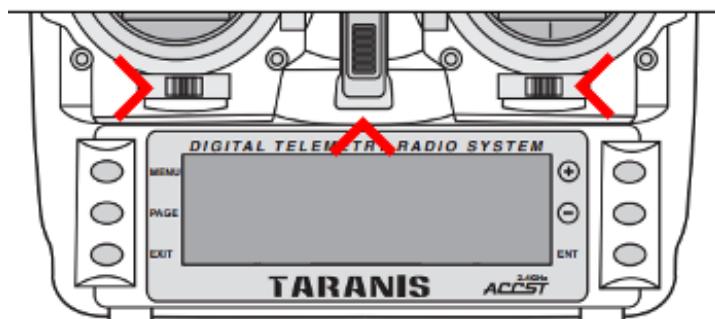


Figura E.1: Encender emisora en modo *bootloader*.

4. Conectar el cable USB a la emisora. El sistema la detectará como un medio de almacenamiento externo, ya que esta emisora dispone de lector de tarjetas microSD para guardar los modelos y configuración de los mismos.

²<http://www.open-tx.org/downloads>



Figura E.2: Conexión de emisora a *OpenTX*

5. Pulsar sobre *Read models and settings from radio* para leer la información de la radio. Tanto los modelos como la configuración.



Figura E.3: Lectura de modelos mediante *OpenTX*.

6. En la lista de modelos, añadir uno nuevo pulsando sobre *Add model*.



Figura E.4: Creación de modelos mediante *OpenTX*.

7. A continuación seguir el asistente dando un nombre al modelo, y seleccionando la opción *Multirotor*.

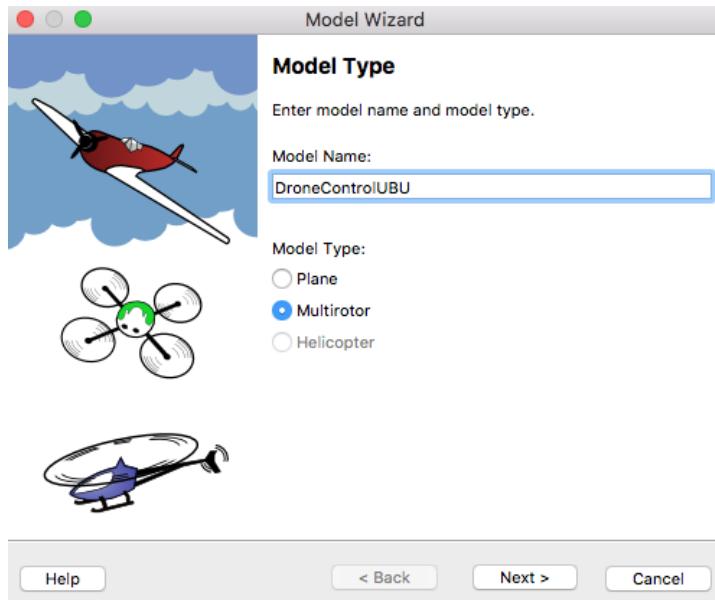


Figura E.5: Creación de modelos mediante *OpenTX*. Asistente de configuración

8. A continuación se muestra el orden de los canales. El sistema está preparado para interpretar el siguiente *mapeo* de canales: 1-Acelerador (*Throttle*), 2-Balanceo (*Aileron*), 3-Elevación (*Elevation*), 4-Rotación (*Rudder*), 5-Armar/Desarmar (*Arm/Disarm*). En esta pantalla solo se muestran los cuatro primeros canales. El canal que permite armar o desarmar el drone se configurará más adelante.

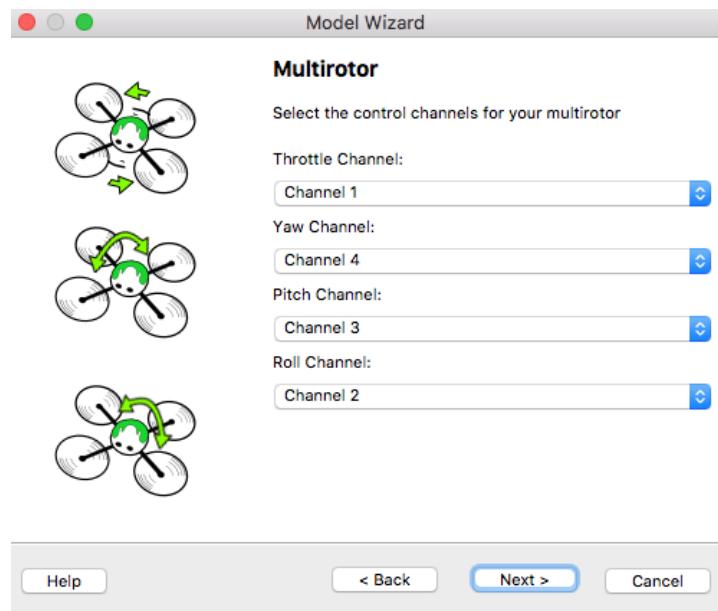


Figura E.6: Creación de modelos mediante *OpenTX*. Asistente de configuración

9. Se guardan los cambios realizados.

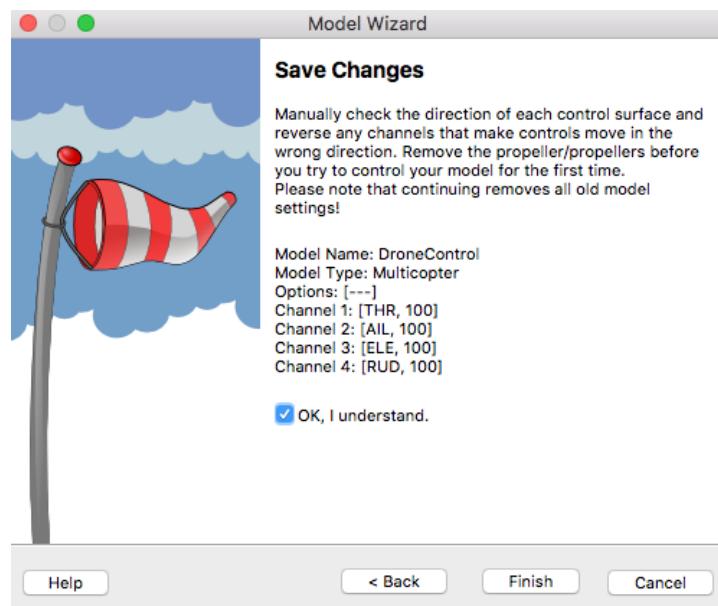


Figura E.7: Creación de modelos mediante *OpenTX*. Asistente de configuración

10. Y ya está casi listo el nuevo modelo.
11. Se da doble click sobre él, para acceder a la configuración avanzada del mismo.

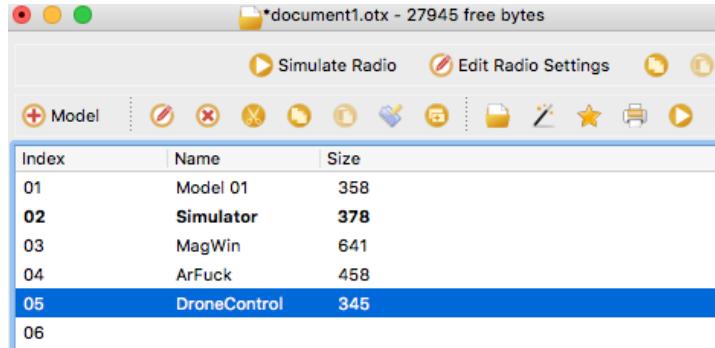


Figura E.8: Creación de modelos mediante *OpenTX*. Asistente de configuración

12. Desplazándonos hasta la parte inferior de la pestaña *Setup*, se establece a «OFF» tanto *Internal Radio System* como *External Radio Module* (desactivando así las opciones de radio, ya que la usaremos como *joystick*). Se establece el *Trainer Port* a *Slave/Jack* para que use el puerto USB como conexión para el *joystick*.

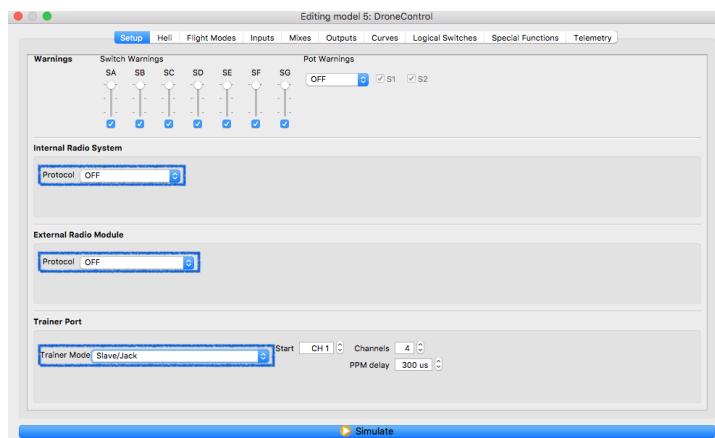


Figura E.9: Creación de modelos mediante *OpenTX*. Asistente de configuración

13. Se accede a la pestaña *Inputs* (en ella se muestran los canales anteriormente configurados) y se añade uno nuevo dando doble click sobre el siguiente espacio vacío, correspondiente a *I5*.

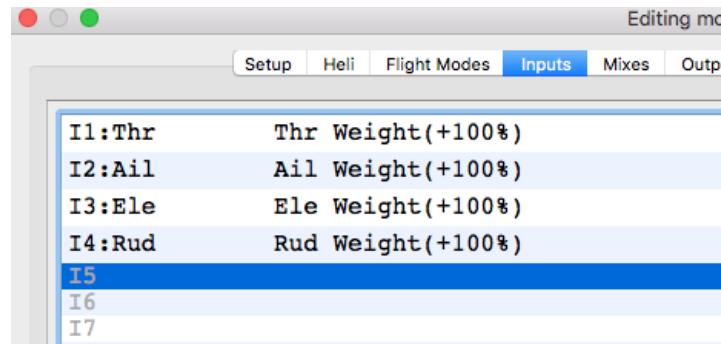


Figura E.10: Creación de modelos mediante *OpenTX*. Asistente de configuración

14. Se muestra ahora la configuración de un nuevo *input*. Se le proporciona un nombre, *Arm* en este caso, y se establece su *Source* (fuente) a uno de los múltiples interruptores disponibles. En el caso de esta configuración, se ha establecido en el interruptor marcado como *SF*, ya que es de fácil acceso.

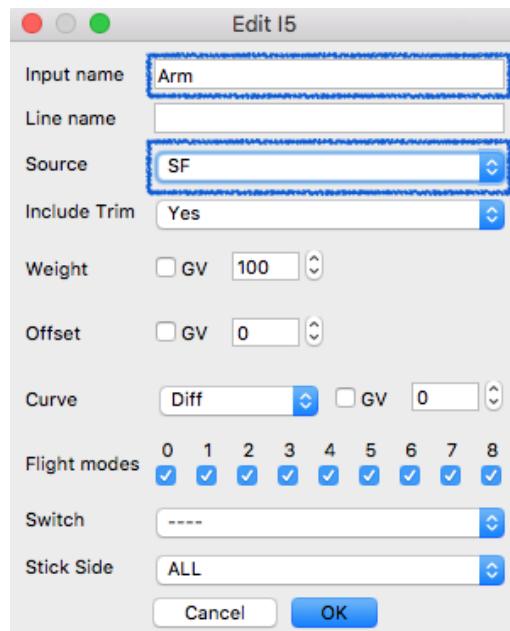


Figura E.11: Creación de modelos mediante *OpenTX*. Asistente de configuración



Figura E.12: Interruptor SF de la emisora de fácil acceso.

15. A continuación se asigna ese *input* recién creado a un canal. Para ello se accede a la pestaña *Mixes* y se selecciona el elemento *CH5* de la lista.

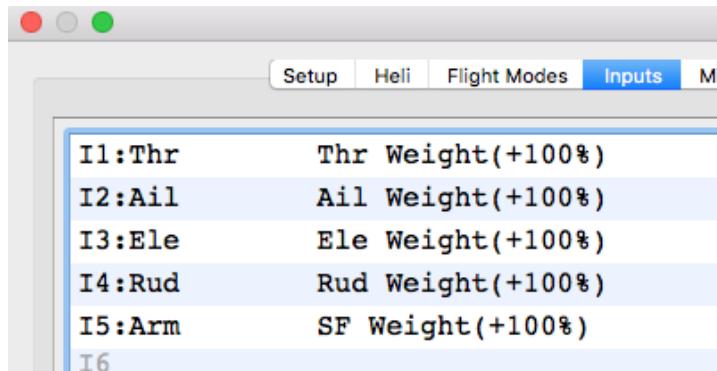


Figura E.13: Creación de modelos mediante *OpenTX*. Asistente de configuración

16. Se da doble click sobre el elemento, y se configura el nombre y la fuente del canal, estableciendo esta al *input* recién creado.

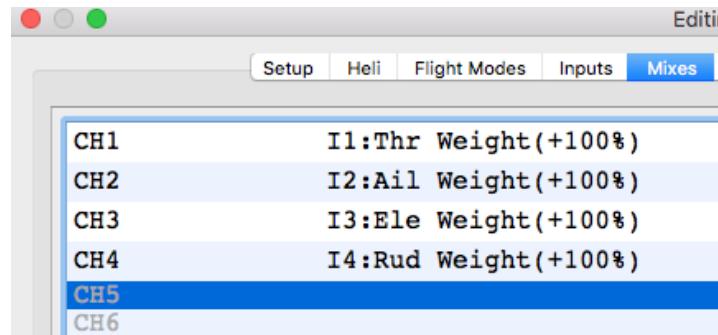


Figura E.14: Creación de modelos mediante *OpenTX*. Asistente de configuración

17. Con esto quedaría listo el modelo creado.

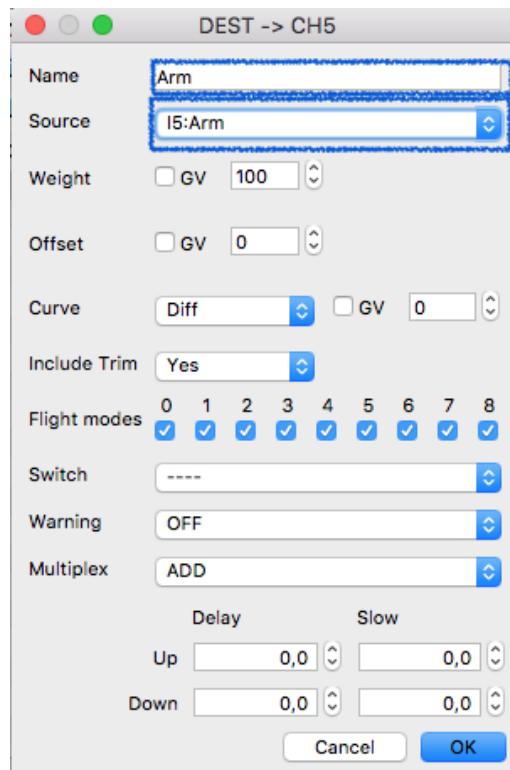


Figura E.15: Creación de modelos mediante *OpenTX*. Asistente de configuración

18. A continuación se escribe el modelo en la configuración de la emisora, pulsando sobre *Write Models and Settings to Radio* y se pulsa sobre *Write to TX* en la ventana emergente.

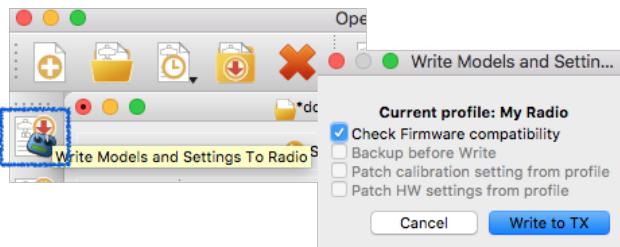


Figura E.16: Escritura de modelos mediante *OpenTX*.

Ahora la emisora dispone de un nuevo modelo, el cual se puede utilizar para manejar el drone a través de la interfaz web. Tan solo habría que seleccionarlo en el menú de la emisora, pulsando la tecla *MENU*, desplazándose hasta el modelo y presionando unos instantes la tecla *ENT*. A continuación se selecciona *Select model*.

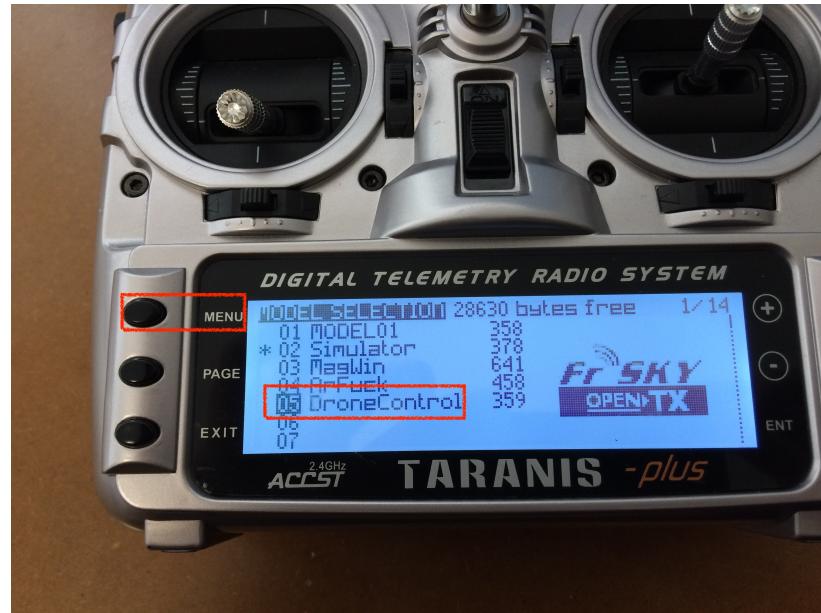


Figura E.17: Acceso a los diferentes modelos de la emisora.



Figura E.18: Acceso al modelo creado en la emisora.



Figura E.19: Uso del modelo recién creado en la emisora.

E.4. Manual del usuario

Para utilizar la aplicación web, el usuario debe disponer de credenciales de acceso. Dichas credenciales solo pueden ser provistas por un administrador del servidor web.

Inicio de sesión

Cuando disponga de ellas, el usuario puede dirigirse a la página creada a tal efecto, e iniciar sesión, tal y como se muestra en la Figura E.20.

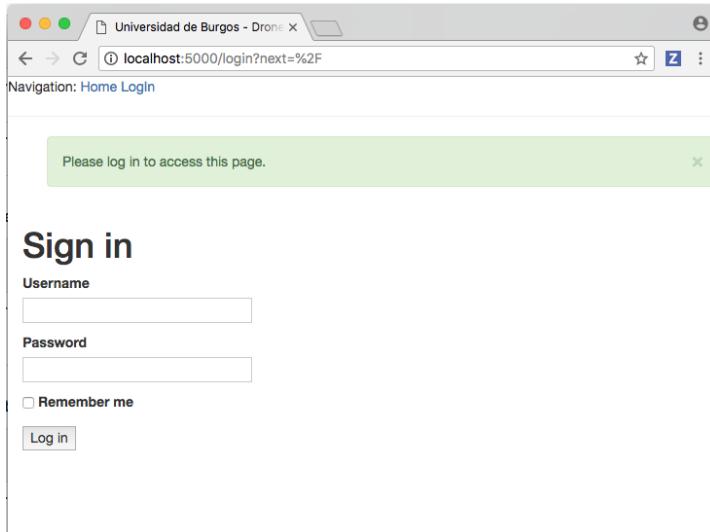


Figura E.20: Inicio de sesión en la aplicación web.

De intentar acceder al *index* de la aplicación web directamente, sin haber realizado el inicio de sesión, redirigirá a la página de *login*, mostrando un aviso informativo.

Si el usuario dispone de credenciales de acceso, y un *drone* asignado, podrá acceder al sistema de control del dispositivo. Cabe destacar que las funcionalidades de sistemas automatizados se encuentran deshabilitadas, a falta de realizar más pruebas de campo.

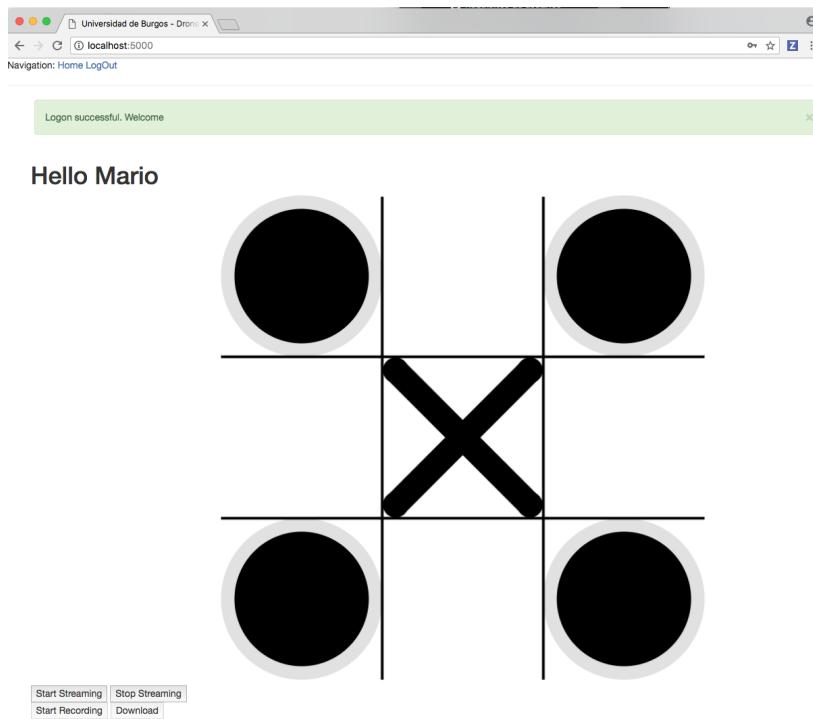


Figura E.21: Página principal del sistema de control remoto del agente.

Activación de vídeo

El *drone* dispone de una cámara que puede ser activada a demanda. Nada más iniciar sesión en la página, la aplicación web conecta con el *drone* y le solicita que informe sobre la disponibilidad de vídeo bajo una configuración predefinida³. De esta forma, en cuanto se solicita iniciar el vídeo, la aplicación ya dispone de la información necesaria para establecer la comunicación con el *drone*.

Pulsando sobre el botón *Start Streaming* se activa el vídeo disponible. Pulsando sobre el botón *Stop Streaming* se detiene la emisión de vídeo.

³Dicha configuración por el momento es fija, y se ha establecido en base a pruebas empíricas. Se hace uso de aceleración hardware para realizar el *encoding* del vídeo a 1280×720 a 15f/s. Será configurable en un futuro para llegar hasta una resolución de 1080p, aunque no se recomienda, dado el alto consumo de ancho de banda.

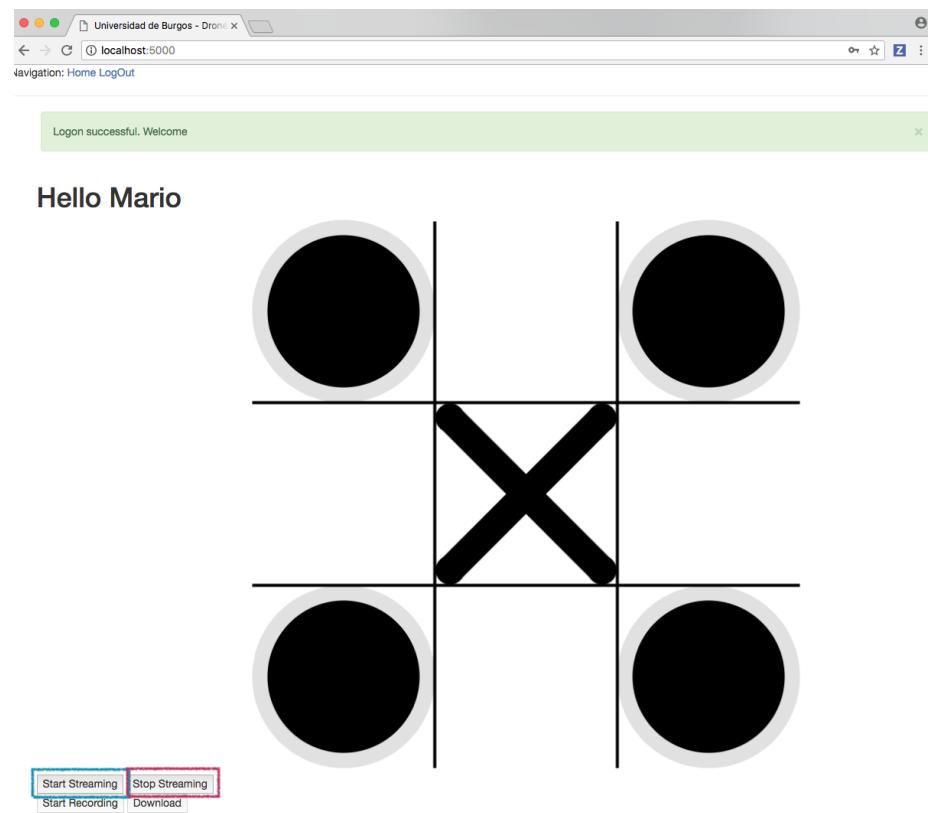


Figura E.22: Página principal del sistema de control remoto del agente. Activación/Desactivación de vídeo.

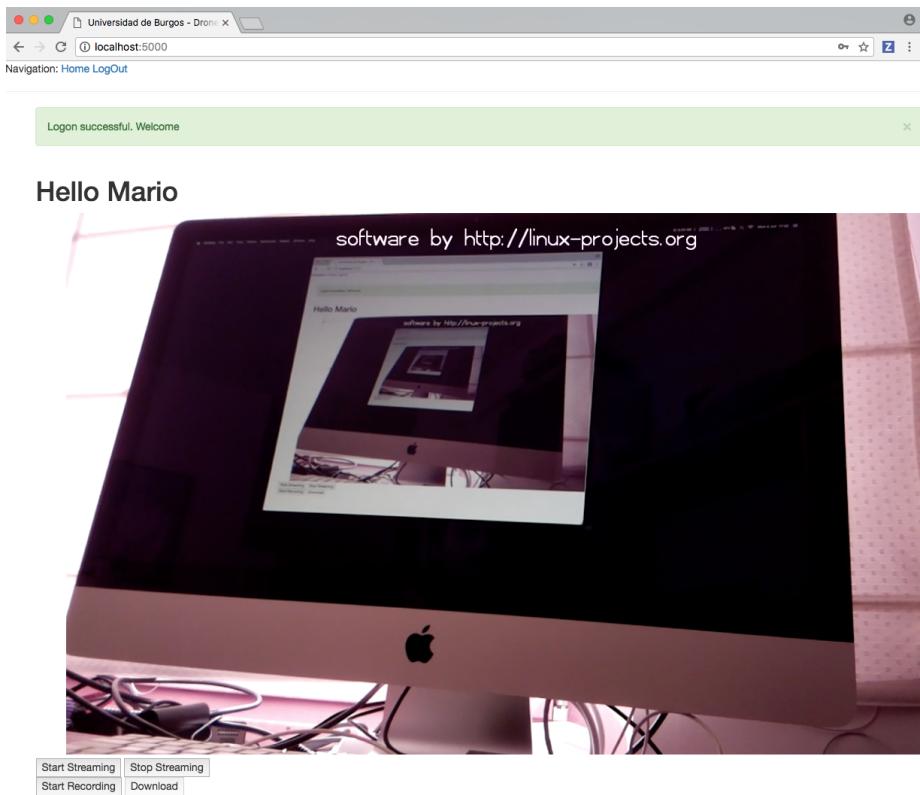


Figura E.23: Página principal del sistema de control remoto del agente. Ejemplo de vídeo con luz diurna.

Cabe destacar que el color del vídeo no es perfecto. Esto es debido a la falta de un filtro de luz infrarroja en la cámara utilizada. Si bien durante el día el tono del color es «rojizo», aporta la capacidad de disponer de visión nocturna haciendo uso de los LED infrarrojos instalados en el *drone*. Dichos LED se activan automáticamente⁴ cuando las condiciones de luminosidad son bajas, y su luz no es visible al ojo humano. Sin embargo, la cámara del *drone* es capaz de captarla. Los anillos de 36 LED infrarrojos aportan, en condiciones de baja luminosidad, una visibilidad perfecta a varios metros de distancia.

⁴Su funcionamiento está basado en un LDR, o *Light Dependent Resistor*, o fotoresistor. Un componente cuya resistencia varía en función de la luz que detecta.

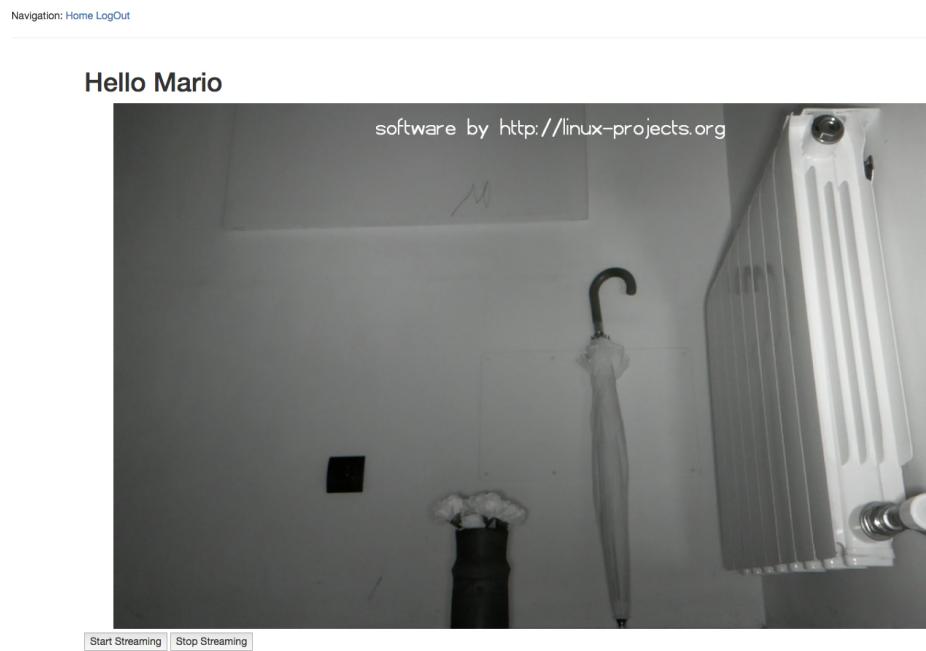


Figura E.24: Página principal del sistema de control remoto del agente. Ejemplo de vídeo con luz infrarroja.

Grabación de vídeo

La interfaz web permite realizar grabación de vídeo. Basta con activar el vídeo, tal y como se explicó en la subsección E.4, y pulsar sobre el botón *Start Recording*. Cuando se deseé detener la grabación se pulsará sobre el mismo botón, que habrá cambiado su nombre a *Stop Recording*. Seguidamente el botón *Download* se habrá activado. Bastará con pulsarlo para descargar un archivo .webm con el vídeo grabado.

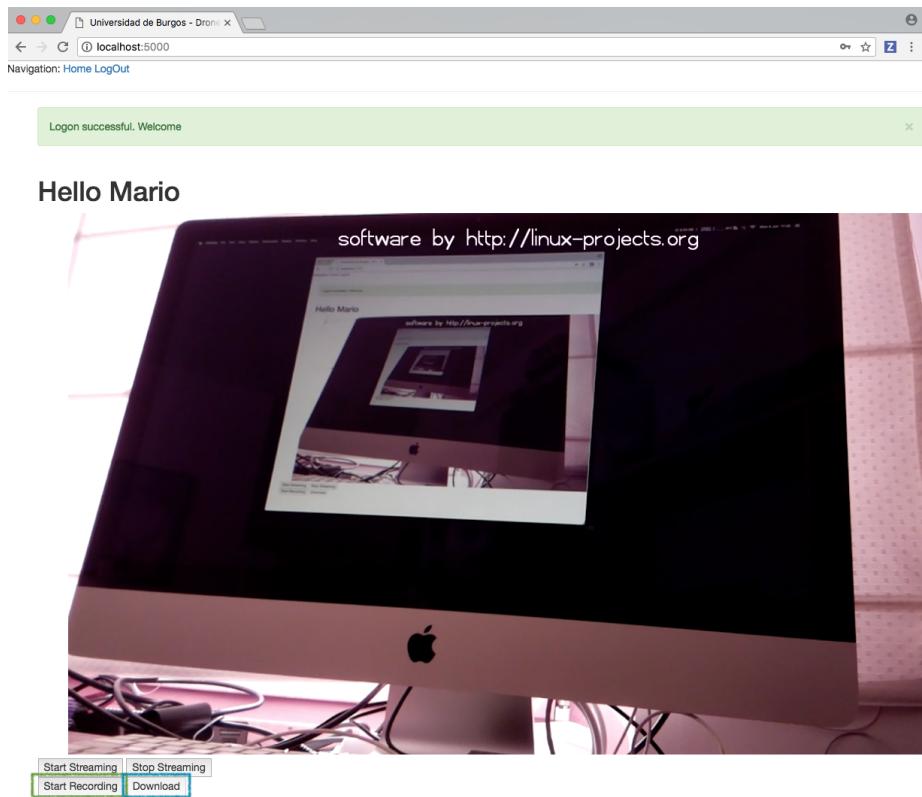


Figura E.25: Página principal del sistema de control remoto del agente. Grabación de vídeo.

E.5. Modo manual

Para activar el modo de control manual del *drone* es necesario haber seguido los pasos para configurar la emisora, ver sección E.3.

En primer lugar se deberá iniciar sesión en el sistema, tal y como se explicó en la sección E.4. Una vez en la página de control del *drone*, se conectará la emisora.

NOTA IMPORTANTE: Para que el sistema de control manual envíe la información de la emisora, y la reconozca, el navegador debe estar activo, es decir, no se deben realizar cambios de aplicación mientras el control manual esté activado, ya que se corre el riesgo de que este se desconecte. Esta restricción es parte de la API de JavaScript para el elemento *Gamepad*,

ver [Gamepad API⁵](#) para más información.

Al conectar la emisora, está será reconocida de forma automática por la aplicación, y se permitirá activar el control manual mediante el botón *Enable Manual*.

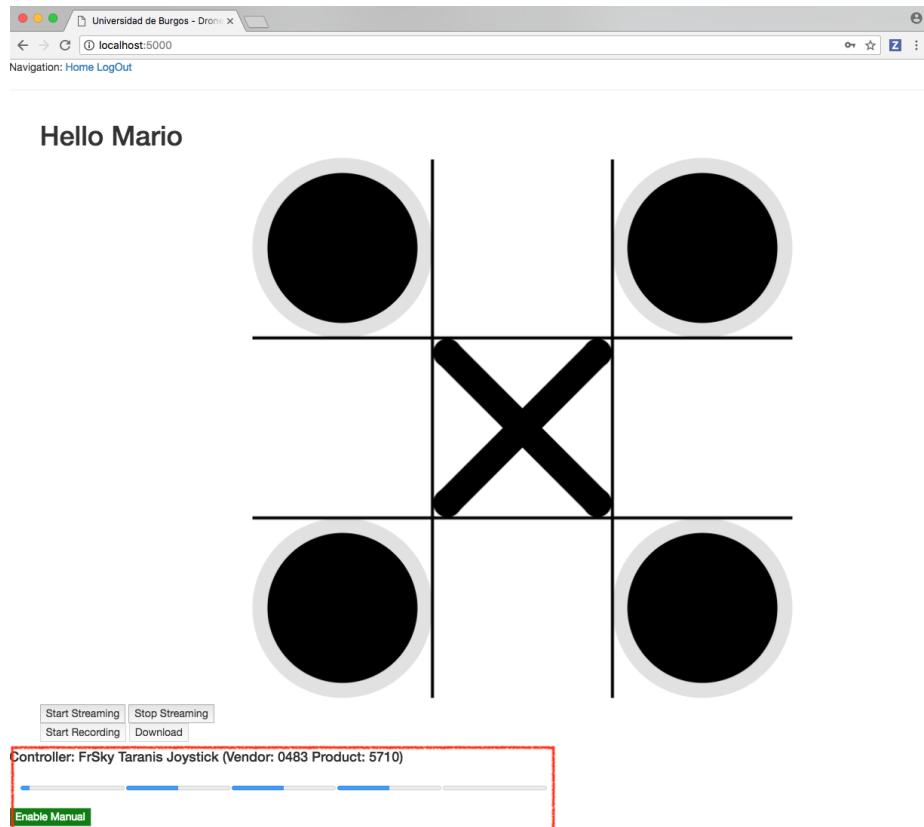


Figura E.26: Página principal del sistema de control remoto del agente. Control manual del *drone*.

Como puede verse en la Figura E.26, se ha reconocido la emisora, y se muestra el valor de los canales en forma de barra de progreso.

Al activar el modo manual, el botón *Enable Manual* cambiará su nombre a *Disable Manual*, y su color a rojo.

⁵https://developer.mozilla.org/en-US/docs/Web/API/Gamepad_API/Using_the_Gamepad_API

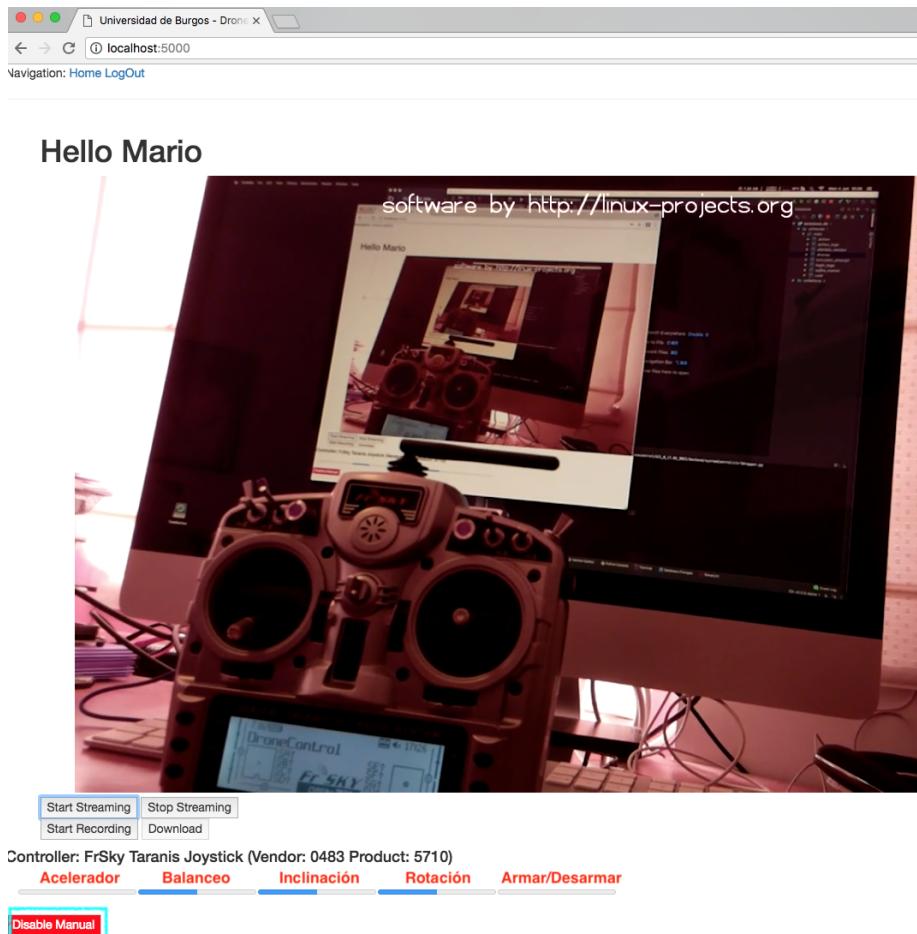


Figura E.27: Página principal del sistema de control remoto del agente. Control manual del *drone*. *Drone Activo*.

NOTA IMPORTANTE: Desde este momento, el control del *drone* se encuentra activo. NUNCA, bajo ninguna circunstancia, active el *drone* cerca de personas o animales. Siempre deben observarse las medidas de seguridad detalladas en la subsección A.3. Este dispositivo no es, ni mucho menos, un juguete. Las hélices giran a una velocidad máxima de 13608 rpm, y pueden causar daños o heridas graves, ver Figura E.28.



Figura E.28: Los experimentos, mejor con gaseosa.

A continuación puede activarse, armarse, el *drone* haciendo uso del interruptor que se designó en la sección E.3. Armar el *drone* supone que las hélices comienzan a girar a velocidad baja.

Se recomienda encarecidamente activar la emisión de vídeo si se va a operar el *drone* de forma remota.

Bibliografía

- [1] Y. K. J. Borenstein, “Potential field methods and their inherent limitations for mobile robot navigation,” 1991. [Online]. Available: https://pdfs.semanticscholar.org/738a/a7b9fda536e7d603e9d84725f38f96b50150.pdf?_ga=2.22475512.2130287822.1523287256-1927681115.1523287256
- [2] I. M. Rekleitis, “A particle filter tutorial for mobile robot localization,” 2003. [Online]. Available: <http://www.cim.mcgill.ca/~yiannis/particletutorial.pdf>
- [3] Wikipedia, “Pid controller,” 2018, [Internet; descargado 9-abril-2018]. [Online]. Available: https://en.wikipedia.org/wiki/PID_controller
- [4] S. Social, “Bases y tipos de cotización 2018,” 2018, [Internet; descargado 1-junio-2018]. [Online]. Available: http://www.seg-social.es/Internet_1/Trabajadores/CotizacionRecaudaci10777/Basesytiposdecotiza36537/index.htm
- [5] UBU, “Retribuciones segun ii convenio de pdi laboral año 2017,” 2017, [Internet; descargado 2-junio-2018]. [Online]. Available: http://www.ubu.es/sites/default/files/portal_page/files/pdi_laboral_2017_2.pdf
- [6] J. C. R, “Ley de navegacion aerea,” 2001, [Internet; descargado 2-junio-2018]. [Online]. Available: http://www.fomento.gob.es/NR/rdonlyres/B8761680-F6C0-48CC-ABCD-3F5CD7480008/71900/RD_37_2001.pdf
- [7] R. Foundation, “Raspbian,” 2018, [Internet; descargado 9-abril-2018]. [Online]. Available: <https://www.raspberrypi.org/downloads/raspbian/>

- [8] JetBrains, “Python ide for professional developers,” 2018, [Internet; descargado 9-abril-2018]. [Online]. Available: <https://www.jetbrains.com/pycharm/>
- [9] Git, “Git,” 2017, [Internet; descargado 9-abril-2018]. [Online]. Available: <https://git-scm.com/>
- [10] GitKraken, “The legendary git gui client for windows, mac and linux,” 2018, [Internet; descargado 9-abril-2018]. [Online]. Available: <https://www.gitkraken.com>
- [11] SonarSource, “Analyzing with sonarqube scanner,” 2018, [Internet; descargado 9-abril-2018]. [Online]. Available: <https://docs.sonarqube.org/display/SCAN/Analyzing+with+SonarQube+Scanner>