Universidad del País Vasco
Euskal Herriko Unibertsitatea

Ingeniaritza Goi Eskola Teknikoa
Escuela Técnica Superior de Ingeniería
Bilbao

Trabajo de Fin de Grado

Implementación SDR de un radar localizador de aviones comerciales mediante la recepción de señales ADS-B

Alumno: Mikel Badiola Landa

Profesor: Manuel Velez

2017-2018

1. Índice

Resumen trilingüe

Proiektu honen helburua ADS-B-aren bidez hegazkin komertzialen detekziorako SDR motako radar bat burutzea da eta, ondoren, irabazitako ezagutza hori erabiliz, hobekuntza bat planteatu, non hegazkinen detekzio tartea handituko den. Helburu hori lortzeko erabiliko diren tresnak aztertuko dira sakonki, hala nola; ADS-B seinaleak jaso, beraien deskodifikazioa burutu eta seinaleen prozesamendua ahalbidetuko dute. Behin tresna guztiak eskuratuak izanik, hardware zein software, kokapen konkretu batzuetan burutuko dira egindako lanari baliotasuna bermatzen dituzten frogak.

El objetivo de este proyecto es implementar un radar SDR para localizador aviones comerciales mediante ADS-B y posteriormente usar ese conocimiento adquirido en crear una mejora que permita una ampliación del rango de detección de los aviones. Para conseguir ese objetivo se analizarán con detenimiento una serie de herramientas que nos permitirán llevar a cabo la recepción de las señales ADS-B, su decodificación y su posterior procesado. Una vez adquiridas las herramientas necearías, que pueden ser tanto de software como de hardware, se realizarán pruebas de campo que nos demuestren la validez de los trabajos realizados.

The purpose of this project is to implement a radar using SDR for commercial aircraft locator by ADS-B and then use this knowledge to create an improvement that allows the extension of the detection area of the aircrafts. To fulfil this objective a series of tools that will allow us to carry out the reception of the ADS-B, the decoding of the signal and post-processing of the data will be analysed carefully. Once the necessary tools have been acquired, which can be either software or hardware, field tests will be carried out to demonstrate the validity of the work done.

Listado de tablas

Listado de imágenes

2. Introducción

El transporte aéreo se ha convertido en uno de los sistemas de trasporte más populares en la tierra, ya sea para transporte de carga o para su uso comercial. Convirtiéndose así en la modalidad de transporte más regulada en el globo terrestre. El avance tanto en las aeronaves como en las infraestructuras han permitido su progreso mundial y parte de este avance inevitablemente está ligado a las telecomunicaciones. Un sistema común de posicionamiento entre los diferentes componentes que toman parte en las rutas de aviación es de vital importancia y a ello ha contribuido el sistema Automático Dependiente de Vigilancia – Difusión.

La tecnología ADS-B funciona como un radar biestático donde el emisor y el receptor están separados, pero además la innecesaria cooperación entre el emisor y el receptor lo convierte en un radar pasivo. Podemos diferenciar dos servicios principales en esta tecnología: ADS-B de emisión (ADS-B out) y ADS-B de recepción (ADS-B in). El servicio de emisión difunde información como la identificación, posición, altitud y velocidad a través de un transmisor a bordo de los aviones. El servicio de recepción puede ser captado por aeronaves, capturando datos FIS-B (Flight information service–broadcast), TIS-B (Traffic information service–broadcast) y, evidentemente, datos ADS-B. Por lo tanto, podríamos diferenciar tres componentes principales que se utilizan en esta tecnología: infraestructuras en tierra, componentes en el aire y procedimientos de operación [1]. El sistema ADS-B está compuesto de tres partes:

- Un subsistema de transmisión que incluye funciones de generación y transmisión de mensajes de la fuente.
- El protocolo de transporte.
- Un subsistema de recepción que incluye la recepción de mensajes e informa al receptor.

El proyecto se centrará en la creación de un subsistema de recepción de datos ADS-B mediante SDR (Software Defined Radio). Una tecnología que nos permite implementar mediante software aquellos componentes típicamente implementados en hardware. SDR proporciona un entorno eficiente, modificable y flexible, puesto que modificando o sustituyendo sus programas de software, o añadiendo otros nuevos, se consigue cambiar sus funcionalidades.

3. Contexto

El sistema ADS-B de emisión es una tecnología en la que un avión determina su posición a través de la navegación por satélite y emite la correspondiente posición junto a más información periódicamente. Como sus siglas en ingles indican; A “Automatic” siempre esta encendido y no es necesario la intervención humana para su emisión, D “Dependent” es dependiente de la posición que le provee un sistema global de navegación por satélite (GNSS), S “Surveillance” proporciona servicios de vigilancia muy parecidos al radar y B “Broadcast” está permanentemente emitiendo señal por aire para que tanto los receptores aéreos como los terrestres puedan recibir la señal. El sistema hace uso del posicionamiento proporcionado por un sistema GNSS de alta precisión, aunque en la práctica el sistema de posicionamiento utilizado por la mayoría de aeronaves es el Sistema de Posicionamiento Global (GPS).

Existen tres enlaces de datos para el ADS-B; modo S “extended squitter” o 1090 ES, transceptor de acceso universal “UAT” y VDL modo 4 (desarrollado por la aviación civil). Para la realización de este proyecto nos centraremos en el enlace más extendido de los tres, el modo S “extended squitter”, puesto que ha sido el enlace seleccionado en la Unión Europea [2] para empezar a implementar el ADS-B de emisión (será obligatorio para todas las aeronaves en 2020 [3]).

Consiste en una extensión del tradicional Modo S del Radar Secundario. Así, la aeronave transmite regularmente mensajes “extended squitter” conteniendo información relevante de su estado. Estos mensajes son denominados paquetes ADS-B y cada paquete envía un tipo de dato correspondiente a la aeronave emisora. Los paquetes pueden albergar los diferentes datos enviados por el transmisor, desde la posición de la aeronave hasta su velocidad. La cadencia de emisión de los paquetes varía dependiendo del tipo de mensaje. Por ejemplo, los paquetes de posición de la aeronave son emitidos con una frecuencia de 0,5 segundos. Los “extended squitters” son transmitidos en la frecuencia de respuesta del secundario, 1090 MHz, y pueden ser recibidos por cualquier aeronave o estación de tierra convenientemente equipada [4]. El hecho de que las respuestas del radar secundario y otros servicios como TCAS (traffic collision avoidance system) trabajen en la misma frecuencia puede llevar a una saturación de la misma [5].

Un ejemplo gráfico de la señal enviada se puede observar en la ilustración 1 y estas son sus características:

- Velocidad de datos → 1 Mbps (1 microsegundo por bit)
- Modulación → Modulación por posición de pulsos (PPM)
- Codificación Manchester
- Un preámbulo de cuatro pulsos para reconocer el inicio del mensaje (8 microsegundos de duración)
- Un bloque de datos de 112 bits (112 microsegundos de duración)

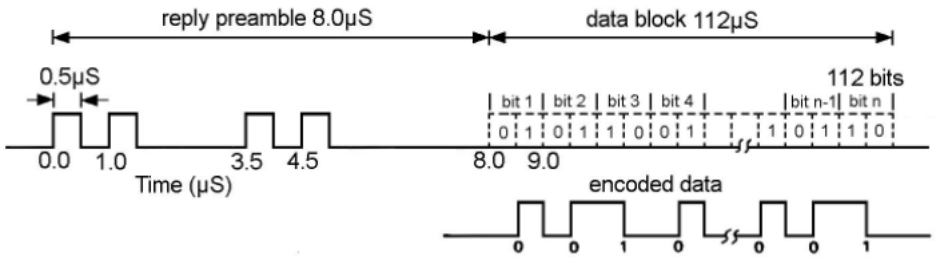


Ilustración 1: Formato de señal ADS-B

Como ya hemos mencionado a nivel lógico estas señales se comportan como paquetes. Conformados por 112 bits que están distribuidos de la manera expuesta en la ilustración 2.

Ilustración 2: Paquetes ADS-B "extended squitter"

- Los primeros 8 bits son de control
    - De ellos los primeros cinco pertenecen al campo “Down Link, DF” que nos proporciona información sobre el tipo de protocolo usado. En nuestro caso como queremos recibir mensajes 1090 ES este campo tendrá un valor de 17, 18 o 19. Aunque no todos los que tengan DF 18 o 19 son interesantes para nosotros.
    - Los otros tres bits están dedicados al campo “Capability, CA” (para DF 17), “Application Field, AF” (para DF 19) o “Control Field, CF” (para DF 18) que nos indica la capacidad de la estación de transmisión.

  Es importante aclarar que para ser un mensaje ADS-B es condición necesaria pero no suficiente tener DF 18 o 19. Para serlo tendrá que tener un CF igual a 0, 1 o 6 en el caso de DF 18 y AF igual a 0 en el caso de DF 19. En el caso de DF 17 en cambio es condición suficiente. [6]

- El siguiente segmento de 24 bits pertenece a la dirección ICAO (International Civil Aviation Organization), que es única en cada aeronave.
- El segmento del mensaje ADS-B está compuesto por 56 bits.
    - Los primeros cinco son para el tipo de mensaje que se está enviando. Como ya hemos comentado dependiendo de este valor el transpondedor enviara el paquete “extended squitter” con una cadencia u otra. [7]
    - El resto pueden variar dependiendo del tipo de mensaje. Pueden ser mensajes que contengan posición, otros que contengan altitud, etc.
- El último segmento para completar el mensaje le corresponde “Parity, PI”, el cual ocupa 24 bits y contiene la información de paridad. Posibilitando la detección y corrección de errores en el decodificador.[8]

Después de esta breve explicación del funcionamiento del ADS-B de emisión, nos centraremos en la explicación de nuestro subsistema de recepción de señales. Antes de nada, conviene recordar una serie de acontecimientos recientes que han influido en la evolución de un tipo de receptores muy utilizados para capturar señales ADS-B.

Hace algunos años explotó en la escena pública de la mano de Eric Fry en marzo de 2010, y luego ampliado por Antti Palosaari en febrero de 2012, el descubrimiento de los dispositivos dongle que originalmente estaban destinados a la recepción de televisión y que podían ser modificados para convertirse en un SDR. Consiguiendo capturar amplios rangos del espectro con una calidad más que aceptable. Esta revelación comenzó con los chips integrados Realtek RTL2832, cuando se descubrió que en su interior tenia ciertos registros y comandos no documentados. Con ellos podía llevarse el chipset a un modo de funcionamiento donde se transfieren las muestras I/Q en banda base sin procesar por un puerto USB al PC. Así se puede decodificar la señal en el ordenador por medio de algún software.

Desde el descubrimiento de esta nueva tecnología muchos proyectos han sido desarrollados, dada la facilidad de implementar un dongle SDR y la obtención del correspondiente software. Tanto su bajo precio como la infinidad de posibilidades que aporta la implementación han promovido una rápida expansión en la comunidad de Internet.

Una de las posibilidades que nos aporta el SDR es la recepción de las señales ADS-B. Así, podemos crear un subsistema de recepción con relativa facilidad. Como ya hemos comentado, la rápida proliferación de este descubrimiento llevo a muchos desarrolladores crear sus propios proyectos y a publicarlos en Internet, donde podemos encontrar infinidad de información al respecto e innumerables dispositivos hardware como software para llevar a cabo esta tarea.

Por lo tanto, la creación del subsistema de recepción SDR para localizar aviones comerciales es el punto de partida para plantear este proyecto.

4. Objetivos

El objetivo principal de este proyecto es la implementación SDR de un radar localizador de aviones comerciales mediante la recepción de señales ADS-B.

Para conseguir nuestra meta, el primer paso será elegir el sistema más adecuado para implementar un subsistema de recepción completo. Para esta elección nos basaremos principalmente en información existente en Internet y trabajos ya realizados por otros que utilizan herramientas parecidas, si no son las mismas, para poder utilizarlo en nuestro proyecto. Estudiar las características de los componentes, tanto hardware como software, seleccionar buenas referencias en Internet, para evitar contratiempos inesperados que retrasen el trabajo, y llevar a cabo pruebas iniciales, principalmente de software. Todo esto será lo que trabajaremos en el estudio de alternativas.

El siguiente paso será la instalación de los componentes seleccionados, comprobando que todo componente seleccionado en el estudio de alternativas funciona correctamente por separado y conjuntamente. Después de la comprobación del correcto funcionamiento, se llevarán a cabo pruebas de funcionamiento sobre emplazamientos específicos para ver la respuesta real del subsistema.

A lo largo de la realización de proyecto se ha visto la posibilidad de ampliar en ciertos aspectos los objetivos iniciales. Entre ellos la búsqueda de una antena que permita mejorar el aérea de recepción respecto a la usada inicialmente. Para confirmar este aumento de la distancia de detección de aeronaves se necesitará de la creación de un software específico que nos demuestre que el objetivo se ha cumplido.

5. Beneficios

El principal beneficio que aporta este proyecto es la posibilidad de implementar y comprender un radar localizador de aviones de manera clara y concisa. El estudio de las alternativas beneficia al lector de este proyecto en poder elegir las opciones que más se adecuen a su objetivo relacionado con algún tema que se trate durante el proyecto, consiguiendo una ayuda que simplificará parte de su trabajo. La descripción de la solución demuestra de qué manera se han utilizado esas herramientas en intentar conseguir los objetivos planteados. Además, la compresión del aspecto teórico de este proyecto también puede ayudar al futuro lector a tener una visión más clara de los fundamentos teóricos del ADS-B y del SDR.

Podría ser de utilidad en el ámbito académico; como sería un laboratorio, donde los alumnos podrían comprender y trabajar de manera práctica y sencilla con el funcionamiento de un SDR o la capacidad de una antena para recibir señales radioeléctricas. También podría ser beneficioso para aquellas personas que tienen intención de hacer una aplicación software utilizando los datos ADS-B que se consiguen a través del SDR. Estos dispondrían de una guía que sería el punto de partida para sus proyectos.

6. Análisis de alternativas

En el análisis de alternativas se analizarán dos aspectos diferentes. Por una parte, la selección del hardware y por otro la del software. En cada una de ellas se establecerán unos criterios de selección los cuales tendrá como objetivo proveer de la solución más eficaz y versátil para llevar a cabo nuestros objetivos.

6.1 Hardware

La primera decisión que tendremos que tomar será qué dispositivo SDR usar para nuestro radar localizador. Para ello, se evaluarán diferentes factores que buscarán la opción más conveniente dependiendo de nuestros criterios. En este caso, en la elección del dispositivo SDR, utilizaremos como factores de decisión el coste del dispositivo, factores técnicos y el soporte que existe de dicho dispositivo en la comunidad de Internet. Este último factor puede facilitar mucho cualquier problema que tengamos a lo largo de la realización de proyecto, resolviendo el problema con mucha mayor facilidad.

Los SDR en general están compuestos de dos partes bastante diferenciadas, una que pertenece al adaptador de radiofrecuencia y otra que está compuesta por un conversor analógico digital. El conversor analógico digital suele estar implementado en un chipset que suele estar controlado por una unidad central de procesamiento. Suelen ser procesadores de propósito general para poder cambiar los protocolos y formas de onda simplemente cambiando el software. Un esquema genérico de un SDR de recepción que se conecta mediante un puerto USB seria como el expuesto en la ilustración 3.

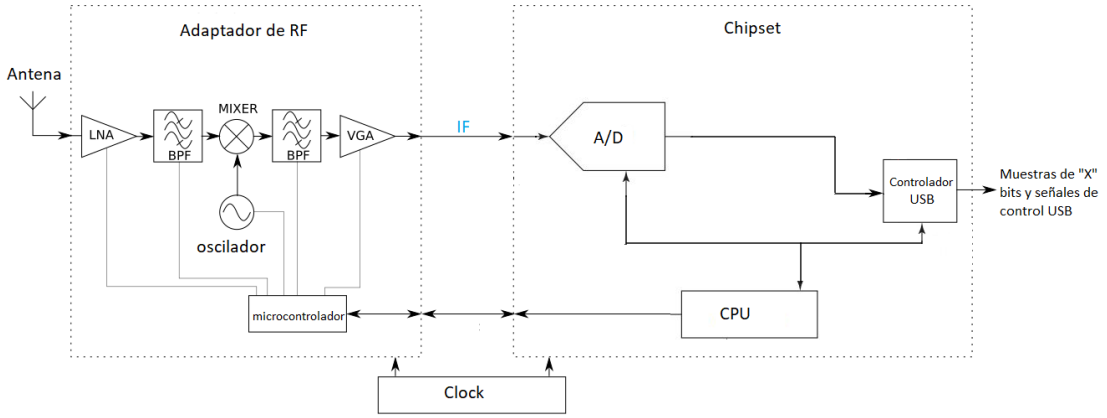


Ilustración 3: Diagrama de un SDR de recepción

Como se ha expuesto en el apartado del contexto, la revelación de Antti Palosaari y Eric Fry supuso toda una revolución en el mercado de los dongle. Provocando una gran cantidad de nuevos proyectos y dispositivos mucho más económicos de lo que estaba acostumbrado un usuario al querer construir un SDR. Por ello, los dispositivos dongle USB basados en chipset RTL2832U formaran parte de nuestra alternativa hardware. También, analizaremos dispositivos

que no contengan ese chipset y que dadas sus características podrían ser completamente funcionales para dar solución a nuestra problemática. Así, como un primer acercamiento, escogeremos dos dispositivos dongle con chipset RTL2832U, pero diferente adaptador de radiofrecuencia, y otros dos que no.

- Dispositivos dongle basados en RTL2832U
    - Elonics E4000
    - SDR-RTL
- AirSpy Mini
- HackRF

Por lo tanto, estos dispositivos serán punto de partida para nuestra elección. Ahora es momento de fijar unos factores de decisión objetivos, que respondan a un criterio técnico y también a un criterio subjetivo como el apoyo que reciben en Internet o de su valor en el mercado. Los factores técnicos deben responder a la pregunta, ¿qué hace a un SDR un buen dispositivo?

Los principales parámetros técnicos que nosotros analizaremos serán los siguientes: rango de frecuencias, ancho de banda (BW), conversor analógico/digital (ADC), chipset, adaptador RF, rango dinámico y precisión de reloj. Algunos de estos parámetros tienen la siguiente importancia:

- Rango de frecuencias → Esta limitado por el tipo de adaptador RF
- Ancho de banda (BW) → En este caso la limitación la impone el chipset (si bien la calidad de los filtros paso banda son importantes). La cantidad de muestras por segundo que es capaz de capturar el conversor analógico/digital, limitará el ancho de banda de la señal captada. Esto se debe al teorema de Nyquist el cual establece una relación de:

$$2f_{muestreo} > f_{\max\_señal}$$

  Aunque para conversores en cuadratura esto no se cumple y queda de esta manera:

$$f_{muestreo} > f_{\max\_señal}$$

  Además, sobrepasar por bastante el límite de Nyquist puede ayudarnos a conseguir el efecto de sobre-muestreado. Este proceso puede mejorar la resolución, reducir el ruido y puede ser beneficioso para evitar el solapamiento (aliasing).
- Conversor analógico/digital (ADC) → Nos especificará la resolución de los bits muestreados. Esto siempre es beneficioso para mejorar la relación señal/ruido.
- Rango dinámico → Es la habilidad del SDR para recibir señales débiles cuando está rodeada por otros fuertes. Con señales de potentes el ADC puede llegar a saturarse si el rango dinámico no es lo suficientemente alto.

Como también hemos mencionado el precio será un factor de decisión. El objetivo principal es la construcción funcional de un radar SDR y por lo tanto un precio excesivamente elevado nos supondría una gran pega para llevar a cabo este objetivo. Por último, el apoyo de la comunidad de Internet a ese dispositivo cobra importancia, ya que la mayoría de información necesaria para llevar a cabo el proyecto proviene de Internet. Además, al ser una tecnología relativamente

nueva, los errores de hardware o la desactualización del software correspondiente puede influir negativamente, por lo tanto, la búsqueda de la solución más eficaz cobra importancia. En la siguiente tabla 1 se hace una comparativa de los factores de decisión.

	Elonics E4000	SDR-RTL	AirSpy Mini	HackRF
Rango de frecuencias	52 – 2200 MHz (con una brecha entre 1100 MHz y 1250 MHz)	24 – 1766 MHz	24 – 1800 MHz	0.1 – 6000 MHz
Ancho de banda	3.2	3.2 MHz	10 MHz	20 Mhz
ADC	8 bits (muestreo en cuadratura)	8 bits (muestreo en cuadratura)	12 bits (muestreo en cuadratura)	8 bits (muestreo en cuadratura)
Rango dinámico	45 dB	45 dB	80 dB	48 dB
Precisión del reloj	0.5 PPM	0.5 PPM	0.5 PPM	30 PPM
Adaptador RF	Elonics E4000	Rafael Micro R820T2	Rafael Micro R820T2 (con mejoras)	RFFC5071
Chipset	RTL2832U	RTL2832U	LPC4370 ARM [9]	MAX5864 [10]
Precio	26 Euro	13.13 Euro	85.46 Euro	258.1 Euro
Otros extras	Menor consumo de energía que RTL-SDR USB 2.0	Muy conocido y una gran comunidad USB 2.0	USB 2.0	Tiene transmisor también USB 2.0

Tabla 1: Comparativas de características de los SDR

El dispositivo llamado HackRF, es el que cuenta con las mejores características de los cuatro. Esta evidente mejora es debido a un chipset y un adaptador RF superior al resto. El chipset MAX5864 tiene un conversor analógico digital de muestreo en cuadratura que soporta hasta 20 MHz de ancho de banda, es decir, las muestras pueden llegar a 20 millones de muestras por segundo. Esta gran cantidad de muestras no son necesarias para nuestro proyecto y además muchos ordenadores con CPU no muy potentes no podrían aprovecharse de esto. En cambio, su resolución no es la mayor de las cuatro opciones, esta menor precisión puede dar lugar a una peor relación señal ruido en algunas situaciones. El rango de frecuencias es también el mayor de todos y satisface nuestra necesidad. Por todo esto y que tenga la función de transmisor el Hack RF tiene el mayor precio de todos.

Si hablamos del Airspy Mini tenemos de un dispositivo más económico, que comparte adaptador de radiofrecuencia con el SDR-RTL. Con ese adaptador su rango de frecuencias consigue captar nuestros 1090 MHz del “extended squitter”. En este caso su chipset es mejor que el RTL1832U, consiguiendo un rango dinámico de 25 dB, un ancho de banda de 10 MHz y una resolución de 12 bits (superior a la del HackRF). Unas características que son más que suficientes para la recepción del ADS-B.

El SDR-RTL fue el dongle que revolucionó el mundo de los SDR, puesto que permitió gracias a su bajo precio el acceso a mucha más gente. Sus características son también suficiente para la recepción del ADS-B, pero es considerable una bajada de características respecto a los anteriores

dispositivos. Como hemos dicho, su precio es muy comedido y la comunidad que tiene en Internet es muy grande. [11]

Analizando el último dispositivo vemos que este se diferencia en el adaptador RF respecto al SDR-RTL. Con el Elonics E4000 capturamos un rango de frecuencias mayor, pero tiene una brecha en un rango de frecuencias intermedias que van desde los 1100Mhz hasta los 1250 MHz. Si bien el resto de sus características responden perfectamente a nuestras necesidades esta brecha roza la frecuencia necesaria para el "extended squitter". Por ello no es recomendable para captar señales 1090 ES.

El primer criterio de selección para descartar será su precio, ya que la facilidad de implementación de nuestro subsistema de recepción de datos ADS-B depende en gran medida del precio por el que se puede adquirir. Por lo tanto, el dispositivo HackRF será descartado. Este dispositivo, aun teniendo unos parámetros técnicos superiores al resto, es de un coste muy superior a sus competidores y tiene unas características que serían desaprovechadas en nuestro caso (como puede ser la transmisión). Por otro lado, tenemos el Elonics 4000 que queda completamente descartado por no tener unas características adecuadas. Los otros dos dongles basados en el adaptador RF R820T2 serán los más adecuados para nuestro propósito, aportan las características necesarias para la recepción de la señal ADS-B 1090 ES. Si comparamos estos dos dispositivos tenemos una gran diferencia, su precio. El SDR-RTL tiene un precio bastante inferior al AirSpy Mini. Aquí nos encontramos ante la disyuntiva de elegir entre un precio mayor pero unas características superiores o un dispositivo más discreto en cuanto a hardware con un precio menor. Como los dos satisfacen nuestras necesidades para la captación de la señal nos decantaremos por el dispositivo más barato. Finalmente, cabe mencionar que SDR-RTL es el dongle más conocido de todos los dispositivos y que goza de gran popularidad dentro de la comunidad de los SDR.

La segunda decisión que hay que tomar respecto al hardware, es la de escoger una antena adecuada para la recepción. El dongle RTL2832U R820T2 seleccionado incluye una antena en su pack de compra y un ejemplo de ella se puede observar en la ilustración 7. Esta antena es específica para la función inicial de los dongle SDR-RTL, que es la recepción de señal DVB-T, pero puede satisfacer las necesidades requeridas por el proyecto [12]. Como hemos especificado en el apartado "Objetivos", en un principio no se preveía la construcción de una antena específica para captar en 1090 MHz. Pero a lo largo del proyecto se amplían los objetivos del proyecto y uno de ellos es la construcción de una antena casera para esta frecuencia específica.

La antena construida es una antena Spider de cuatro patas, que funciona como una antena dipolo cuarto de onda. Las medidas para ella se han calculado de la siguiente manera:

$$\lambda = \frac{c}{f_{1090\ ES}} = 0.275\ \text{m}$$

$$\frac{\lambda}{4} = 68.75\ \text{mm} \approx 69\ \text{mm}$$

6.2 Software

Las decisiones posteriores estarán relacionadas con el software utilizado en nuestro proyecto. Software donde se escogerá el sistema operativo, el driver utilizado para la

modificación del chipset RTL2832U y las aplicaciones utilizadas para la visualización y demodulación de las señales ADS-B. El sistema operativo será dependiente del software escogido, por lo cual, si no queremos descartar aplicaciones debido a nuestra decisión en el SO, la primera decisión será escoger las aplicaciones adecuadas.

El SDR-RTL, lleva integrado un chipset RTL2832U y por lo tanto necesitará la instalación de un driver que le permita transferir las muestras I/Q en banda base sin procesar por un puerto USB al ordenador. Este driver de software libre es la única opción para conseguir que nuestro dongle TV pase a ser un dongle SDR. [13]

Para la decodificación de los datos ADS-B proveniente del dongle USB existen varios programas software disponibles. El factor de decisión será el soporte existente en Internet y la fecha de sus actualizaciones. Estos son los decodificadores más conocidas.

- Rtl1090
- Dump1090
- ADSBSpy

Las nuevas versiones de ADSBSpy no están configuradas para trabajar con nuestro dongle (si lo podría haber hecho con el AirSpy Mini) y aunque existen versiones antiguas que sí lo hacen no tienen soporte. Este hecho nos hace descartar la posibilidad de utilizar el ADSBSpy. Tanto rtl1090 como dump1090 son dos opciones que presentan unas características perfectas para nuestro objetivo. Rtl1090 contiene una interfaz gráfica que nos permite trabajar de una manera más intuitiva y esto para ciertos ámbitos, como puede ser el académico, es una ventaja. En cambio, dump1090 es un software que se controla desde la consola. Dump1090 se caracteriza por su eficacia a la hora de trabajar con los datos ADS-B recibidos y con un mayor rango de opciones con los que poder operar. En detrimento de rtl1090 se puede decir que su última versión es del 15 de diciembre del 2013 y aunque su funcionamiento sea correcto y la versión sea estable es un punto negativo para él. A medida que los objetivos se ampliaron los requerimientos del decodificador ADS-B aumentaron, siendo necesario un software que tuviese más opciones que de los que disponía el Rtl1090. Estos nuevos factores de decisión eran la necesidad de operar con dos dongles a la vez y que se pudiese extraer los datos ya decodificados a un archivo. De modo que, el dump1090 será el decodificador que se utilizará durante el proyecto, tanto para el primer objetivo como para el segundo.

Para la visualización de la señal proveniente del software de demodulación necesitamos un programa que interprete ese flujo de datos y los represente gráficamente en un mapa. Existen varios softwares que pueden darnos esta funcionalidad. En este caso el factor de decisión también estará relacionado con la fecha de actualizaciones, pero además se tendrá que observar la compatibilidad con el software decodificador.

- Virtual Radar Server
- PlanePlotter

El programa que más se ajusta a nuestras necesidades es el VRS, siendo compatible con muchos decodificadores. Ofreciéndonos una flexibilidad bastante amplia en caso de incidencia con el software decodificador. Además, la configuración para recibir los datos del decodificador se puede hacer de una manera muy intuitiva y el propio programa configurara los puertos automáticamente.

La elección del sistema operativo es la última decisión que se tiene que tomar. Todos los programas y drivers tienen la opción de funcionar tanto en Linux como en Windows 10. Como criterio selección he utilizado mi comodidad a la hora de utilizar el SO y la presencia de estos en el ámbito académico. En ambos casos el factor de más peso es el Windows 10 por lo tanto este seré el SO utilizado.

Dentro del software disponible sobre el dump1090 no podemos acudir al software original (de Salvatore Sanfilippo [14]) porque la versión del SO escogida ha sido Windows y fue creado para Linux. Como acabamos de analizar esto sabemos que la versión de Windows también está disponible (de MalcolmRobb [15]). Ahora, estas dos versiones son casi idénticas en cuanto a funcionalidades. Pero varían en ciertas cosas, como puede ser que la versión escogida si soporta paquetes con un DF de 18 y la versión original no lo hace. En la misma explicación de "GitHub" de la versión Windows no se explica que es capaz de decodificar también DF 18, pero si nos adentramos en su código descubrimos que sí lo hace.

Dado el aumento de objetivos fue necesario elegir un lenguaje de programación que nos permitiese operar con los datos recibidos desde el decodificador. Este lenguaje fue Python en su versión 3.6, aunque su elección no respondió a ningún criterio específico. Se observó la posibilidad de utilizar la librería "folium" para la representación gráfica y fue suficiente para decantar la balanza hacia el lenguaje Pyhton.

7. Descripción de la solución

En este apartado se trabajará con la solución adoptada para cumplir los objetivos marcados para este trabajo. Se dividirá en dos partes principales, primero la instalación y creación de todos los elementos necesarios para llevar a cabo las medidas y segundo, un proceso de pruebas de lo que nosotros hemos realizado en las medidas. Puesto que este trabajo ha tenido un objetivo más de lo esperado han sido añadidos dos apartados; con la instalación que se ha realizado y la solución que se ha dado ese nuevo objetivo.

7.1 Instalación

Primero tenemos que instalar el driver Zadig, en este caso se ha instalado la versión 2.3 que es la última hasta la realización de este proyecto. Es software libre y se tiene permiso para su distribución y modificación, siempre que se respete la licencia GPLv3 [16]. Se adjuntará junto a este documento el software para ejecutar el Zadig, aunque es recomendable descárgalo de su página oficial para tenerlo actualizado a la última versión. La instalación es muy sencilla, simplemente ejecutar el zadig.exe y después, en la ventana que nos sale, seleccionar "Options->List all devices". Esta primera interfaz seria como la mostrada en la ilustración 4. Ahí escogemos nuestro dongle e instalamos los drivers. Tenemos que tener en cuenta que nuestro ordenador debe reconocer el dongle insertado y que sea en un puerto 2.0 o superior.

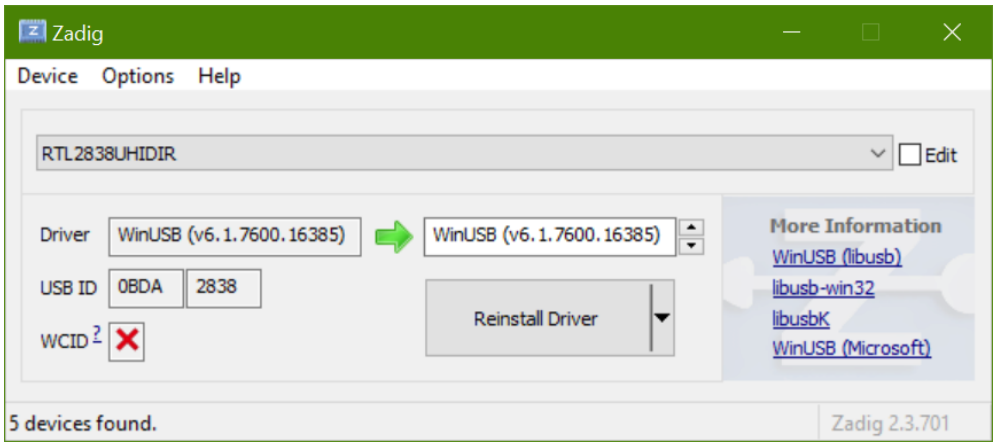


Ilustración 4: Interfaz Zadig

Una vez instalado el driver nos dispondremos a la instalación del software que nos ayudara a decodificar los datos IQ provenientes del dongle. Para este caso, como ya hemos analizado en el apartado de alternativas será el dump1090. También es de software libre y esta liberado bajo la licencia BSD de 3 cláusulas [17]. El software original fue publicado para su uso en Linux, pero hay desarrolladores que han publicado sus trabajos bajo este trabajo que permiten la utilización del software en Windows. Estos trabajos también están sometidos a la licencia BSD de 3 cláusulas y por lo tanto su distribución es posible. En este caso también se adjuntará el software, pero es recomendable descargarlo del link original puesto que los desarrolladores pueden ir actualizándolo con mejoras.

En la carpeta “dump1090-master.zip” viene incluida otra carpeta, también comprimida y llamada “dump1090-win.1.10.3010.14.zip”. Esta carpeta es la que contiene el software que se ejecutará en Windows. Una vez colocado el software en el directorio deseado es recomendable añadir la ruta de toda la carpeta en las variables del sistema (en la variable “Path”), para poder ejecutar el comando desde la ventana de comandos. Una vez añadido sólo necesitamos ejecutar el comando “dump1090” en nuestra ventana de comandos y el programa se ejecutará. Se puede utilizar el comando “dump1090 --help” (como se puede observar en la ilustración 5) para visualizar las opciones que nos aporta el software. Como dump1090 es un programa que utiliza la red es posible que haya que dar permisos al firewall.

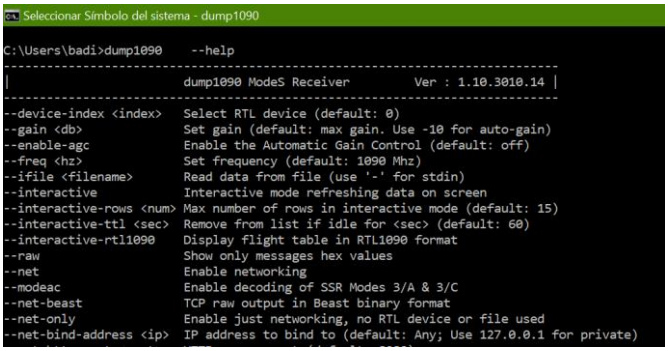


Ilustración 5: Captura con comando "dump1090 --help"

El último paso que hay que tomar para poder visualizar los datos de las aeronaves es la instalación del programa VRS. Se descarga el instalador de la página oficial de Virtual Radar Server, donde podremos encontrar la última versión [18]. No es necesario cambiar ningún parámetro a lo largo de la instalación. Una vez instalado y ejecutado el programa, necesitamos conectar los datos trasmitidos por el dump1090 al VRS. Para ello vamos a “Tools->Options” y en esta nueva ventana en el panel izquierdo en el apartado de “Receivers” añadimos uno mediante el botón “+” verde. Ahora tenemos que configurar el receptor/decodificador, pero VRS cuenta con una herramienta llamada “Wizard”, que nos facilitará mucho este paso. Al seleccionar esto especificamos el tipo de hardware que tenemos (“A software defined radio” en nuestro caso), el programa decodificador que utilizamos (“dump1090” en nuestro caso) y si el SDR está funcionando en el mismo equipo (“Yes” en nuestro caso). Una vez usado este asistente no es necesario configurar nada más en el apartado “Receivers”. Es recomendable especificar, también en la ventana “Tools->Options”, nuestra posición. Lo haremos en el apartado “Receiver Locations” y así conseguiremos situar esa posición como el centro del mapa. Para conseguir visualizar los datos hay que pinchar en la dirección URL que aparece en azul en la ventana principal. Si bien esta opción usa el navegador para mapear los aviones no es necesario estar conectado a Internet. Por defecto nos lo ha configurado para utilizar la dirección IP local “127.0.0.1”. En la ilustración 6 se dispone de una imagen de su interfaz de los puntos importantes para su instalación.

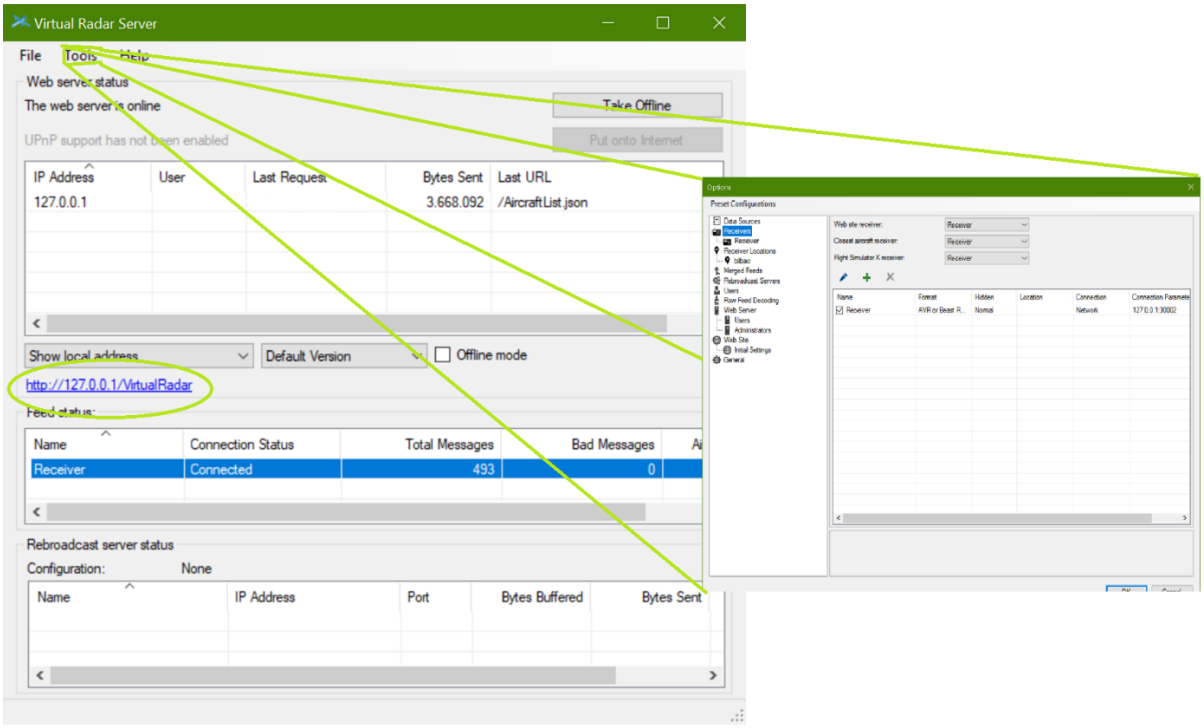


Ilustración 6: Interfaz VRS con indicaciones de su uso

La creación de la antena Spider también será parte de este apartado. Para su construcción se han utilizado materiales caseros, a excepción del adaptador utilizado para ser compatible con el dongle. El dongle utiliza un micro conector coaxial hembra, que tiene el mismo aislamiento eléctrico que los conectores SMB (SubMiniature B), pero con unas dimensiones del 30% inferiores. Este conector es frecuentemente utilizado en los receptores GPS y cuenta con una impedancia de 50 ohm. Una vez obtenido el adaptador de micro coaxial macho a coaxial de televisión hembra, se construye la antena. Para ello se sueldan sobre las cuatro esquinas de una placa de metal cuadrada las cuatro patas de cable rígido, con un ángulo de 45 grados sobre la placa cada una. Esta base se usará como referencia de tierra y estará en contacto con la malla de nuestro cable. Después de añadir las cuatro patas de 69 milímetros a nuestra base, se soldará al núcleo del cable otra pieza de cable (también de 69 milímetros), el cual será el encargado de llevar la información. Finalmente se hará un agujero en nuestra placa metálica para pasar por ahí la pieza que está conectada al núcleo del cable y se soldará a la misma. En nuestro caso esta abertura ya existía en la placa, por lo que no ha sido necesario hacerlo. Es recomendable utilizar un material aislante entre la placa y el cable central, para evitar posibles contactos indeseables. Para una mejor comprensión de la estrecutora general de la antena Spider, en la ilustración 7 se puede observar una comparativa entre las dos antenas.

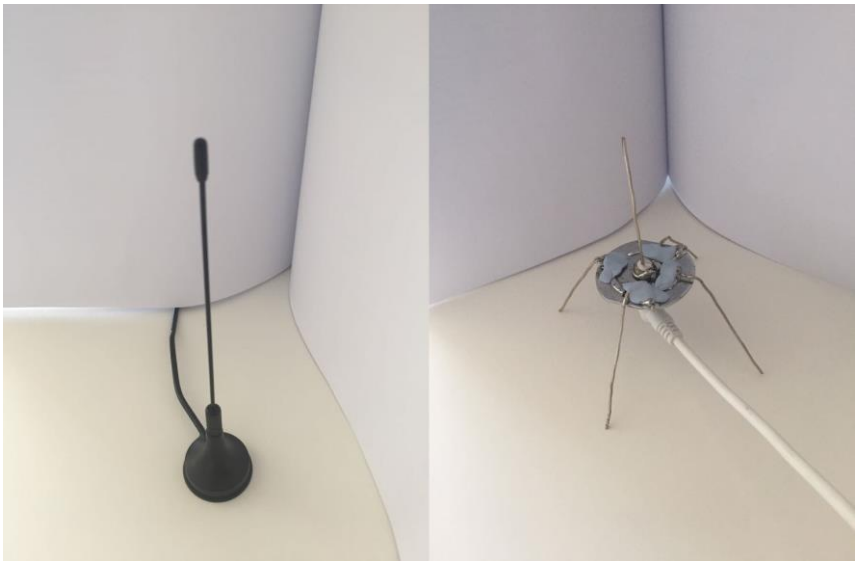


Ilustración 7: Comparativa de las dos antenas; izquierda ant.defecto y derecha ant.Spider

Para realizar comprobaciones del correcto funcionamiento de la antena en 1090 ES se realizarán una serie de mediciones. El instrumento más utilizado para esto es un analizador de redes, el cual está disponible en el laboratorio de la universidad. Se utilizará la función principal del analizador de redes que es medir los parámetros S [19]. Generalmente estas mediciones se realizan en un laboratorio con cámaras anecoicas, y libres de cualquier interferencia electromagnética, pero no disponemos de esta tecnología y tendremos que prescindir de ella. Para evitar la interferencia de cualquier señal externa se ha evitado colocar la antena a la hora de las mediciones cerca de las ventanas. Además, se ha comprobado, mediante un analizador de espectro y utilizando su función “Max hold”, si la frecuencia que se utilizará durante la medición está libre de interferencias. Los datos obtenidos se plasman en la ilustración 8.

Ilustración 8: Captura de pantalla de analizador de espectro con los rangos de frecuencia del ADS-B

Ahora se dará paso a conectar la antena al analizador de redes, previamente adaptado. El valor que nos interesa es el ya mencionado parámetro de dispersión, más específicamente el módulo del parámetro S11 al cuadrado en una escala logarítmica. Parámetro que es el coeficiente de reflexión de la tensión del puerto de entrada. Este coeficiente al cuadrado nos indica la relación entre la potencia reflejada y la potencia incidente, también denominado perdidas de retorno. Nos interesa ver el S11 en la frecuencia que la antena será utilizada, porque un valor muy bajo en dB significa que muy poca señal ha sido reflejada por la antena y en cambio un valor cercano a cero todo lo contrario y eso no es nada recomendable.

Dicho esto, medimos en el analizador de redes en el rango de frecuencias que nos interesa, como es lógico centrada en 1090 MHz y un “span” de 100MHz. Y obtenemos capturas de pantalla que se observan en la ilustración 9 y 10.

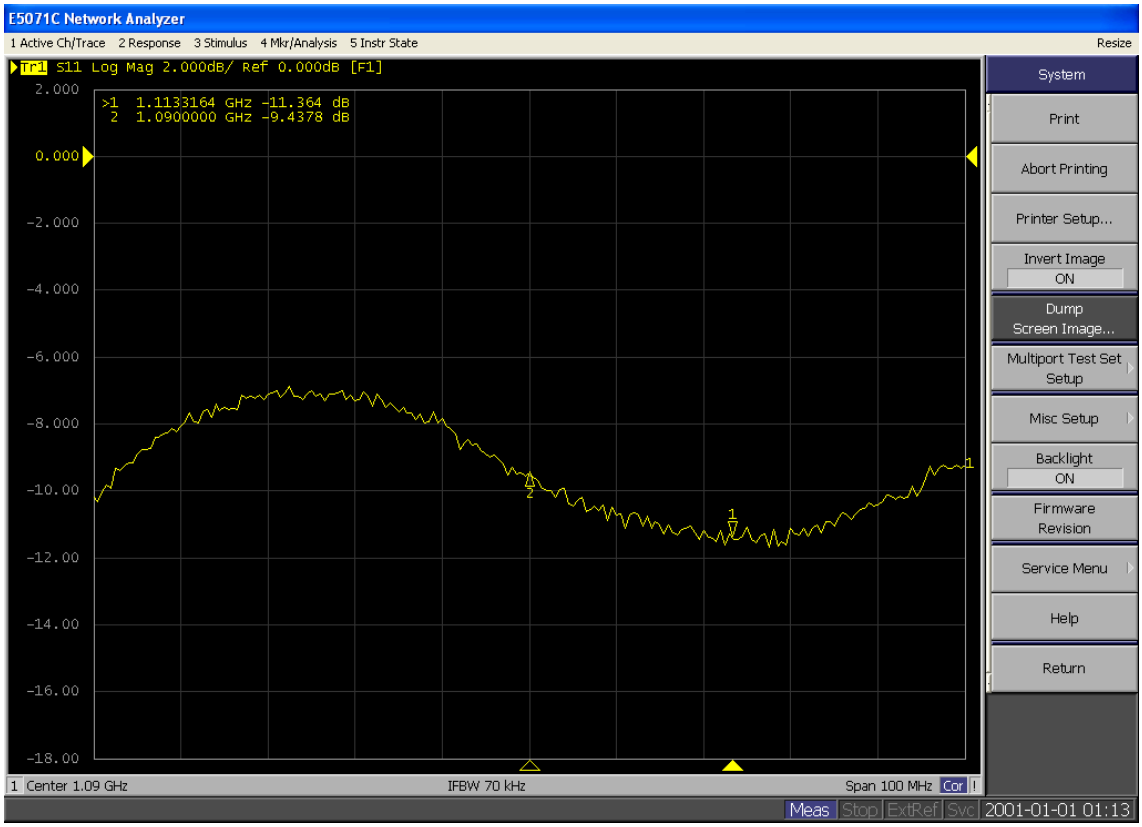


Ilustración 9: Captura de pantalla del analizador de redes con ejes X e Y

En la ilustración 9 se puede ver el resultado obtenido en la pantalla del analizador de redes. Donde en el eje Y encontramos el valor absoluto del coeficiente de reflexión al cuadrado en decibelios y en eje X la banda de frecuencias utilizadas. Por lo que la potencia reflejada, el coeficiente de reflexión y el porcentaje de potencia reflejada serán:

$$10\log(|\Gamma|^2) = -9{,}4378\ dB$$

$$|\Gamma| = 0{,}3373$$

$$|\Gamma|^2 = \frac{P_{recibida}}{P_{incidente}} = 11{,}11\%$$

Ese porcentaje de potencia reflejada es un valor que podemos asumir perfectamente para una correcta recepción de señales. Ahora bien, la frecuencia colindante de 1113 MHz se puede deducir que obtendrá unos mejores resultados, puesto que la potencia reflejada es menor.

$$10\log(|\Gamma|^2) = -11{,}364\ dB$$

$$|\Gamma| = 0{,}2703$$

$$|\Gamma|^2 = \frac{P_{recibida}}{P_{incidente}} = 7{,}31\%$$

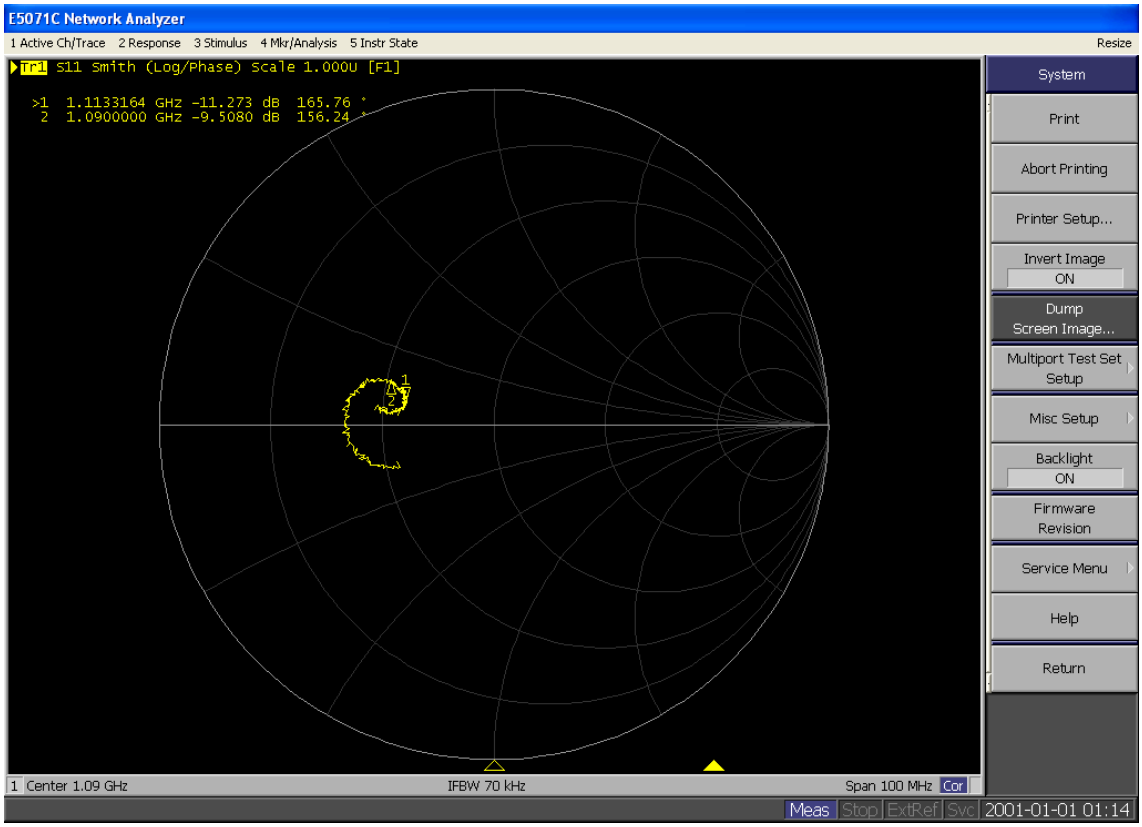


Ilustración 10: Captura de pantalla del analizador de redes con la carta de Smith

También disponemos de la oportunidad de poder visualizar el parámetro S11 con la Carta de Smith en la ilustración 10. Donde el centro del círculo corresponde al coeficiente cero, es decir, un coeficiente de reflexión nulo (ideal) y a medida que nos alejamos esa distancia corresponde al módulo del coeficiente de reflexión. El ángulo representa la frecuencia en la que estamos analizando, por lo tanto, el ángulo más cercano al centro corresponde a la mejor versión de la antena Spider. Como en la imagen anterior, la mejor frecuencia de funcionamiento corresponde a 1.1133 GHz.

$$\Gamma = |\Gamma| \cdot e^{j\theta}$$

En definitiva, la antena Spider tendrá un porcentaje de potencia reflejada (11.11%) bastante óptimo para la labor que vamos a desempeñar (recibir señales de 1.09 GHz).

7.2 Proceso de pruebas

La arquitectura general que tiene todo nuestro subsistema de recepción de señales ADS-B 1090 ES para el primer objetivo en la ilustación 11.

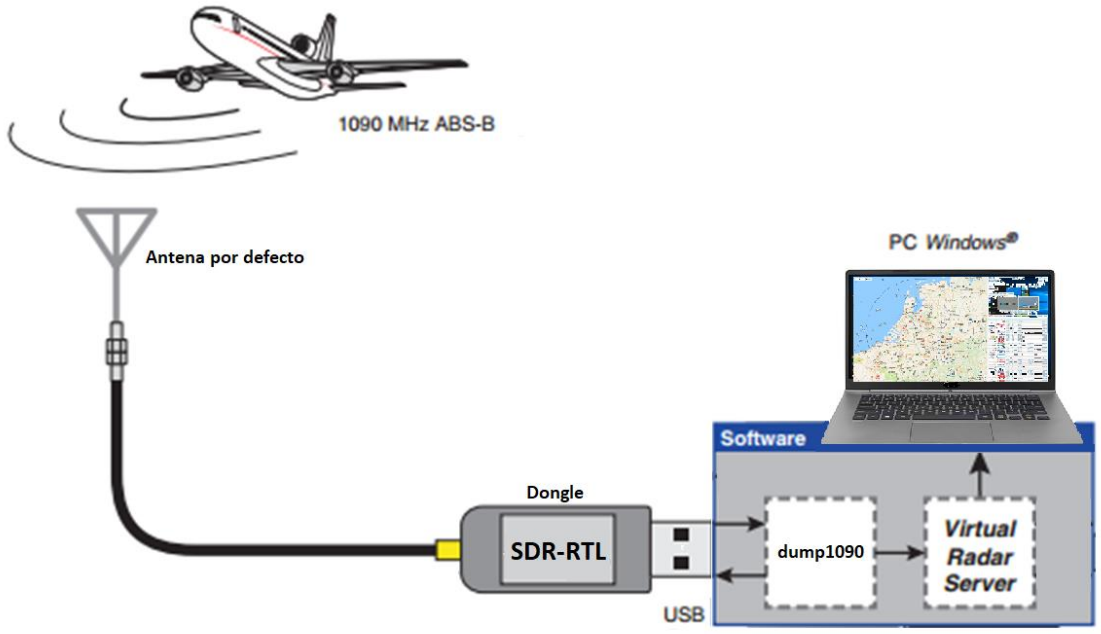


Ilustración 11: Arquitectura del primer objetivo

Las pruebas se realizarán en las coordenadas con latitud 43.3094 y longitud-1.9782 (una aproximación a las coordenadas reales) y utilizando la antena del SDR-RTL por defecto. La localización está situada en lo alto de una azotea y no hay ningún edificio interfiriendo con las posibles señales provenientes de las aeronaves. Se pondrá primero a prueba el dump1090 para comprobar si se capturan los paquetes ADS-B. Ejecutamos el comando “dump1090 --net” y observamos si aparecen paquetes recibidos.

A la hora de realizar estas pruebas de campo uno de los problemas existentes era la falta de control de los elementos que toman partido en la prueba. Las condiciones en las que se realizan no son ideales y en ciertas situaciones pueden darnos problemas como el que ocurre en este caso. En esta localización se observó que debido a una comisaría de policía cercana al lugar algo estaba obstaculizando la recepción de la señal. Probablemente la causa sería un inhibidor de frecuencias que se encontraba instalada en la comisaria. Colocando la antena sin ningún tipo de obstáculo con la comisaria la recepción era totalmente nula y en cambio con un obstáculo entre medias la recepción era inmediata. Para futuras pruebas este emplazamiento será modificado para evitar este problema.

En cuanto empiezan a recibirse los paquetes, estos son mostrados en la ventana de comandos con toda la información acerca de cada uno de ellos. Hay un detalle importante que podemos deducir en este punto, el dump1090 no discrimina los paquetes en función de su campo Down Link, por lo que podemos observar en la ilustración 12 un ejemplo de los paquetes que no son mensajes puramente ADS-B (DF 17, DF 18 con CF 0,1 o 6 y DF 19 con AF 0). Como ya se ha explicado en el apartado del contexto en la frecuencia 1090 MHz también operan las

respuestas del Modo S del radar secundario. Son paquetes que comparten características con el 1090 ES, pero necesitan de una solicitud de mensaje (en 1030 MHz) y, por lo tanto, no son automáticos. De todas formas, los que otorgan la posición y serán utilizados por el VRS si son paquetes con DF 17 o DF 18 (líneas entre 375 y 417).

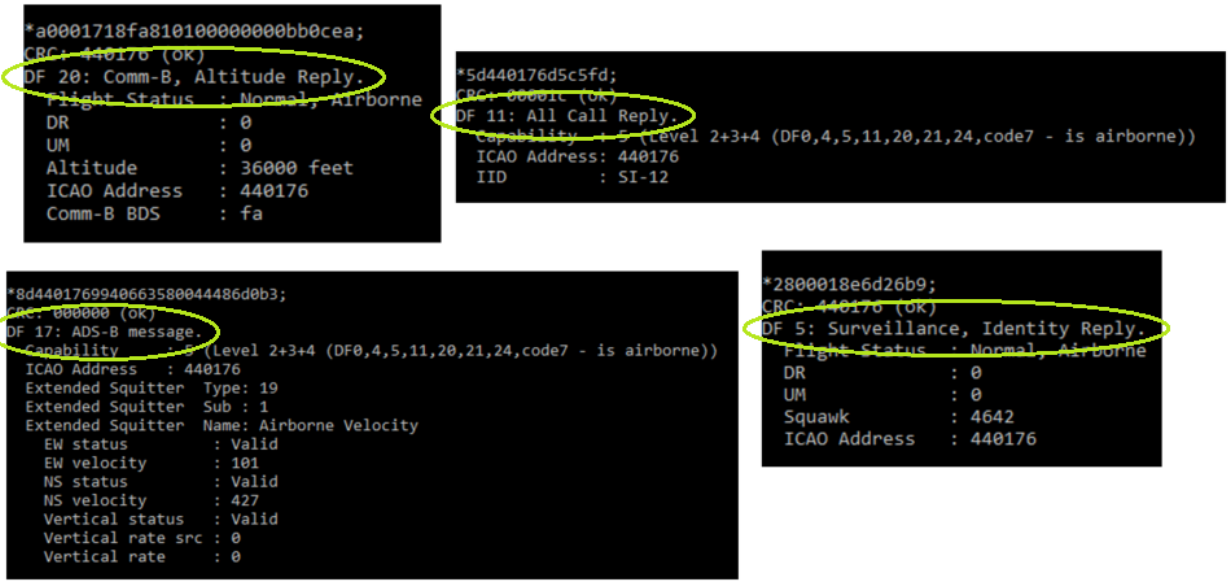


Ilustración 12: Ejemplos de paquetes recibidos por dump1090

Una vez comprobado que tanto el SDR-RTL y el dump1090 funcionan correctamente vamos a disponernos ejecutar el VSR. Como previamente hemos configurado el VRS para nuestro receptor, debería empezar a recibir los paquetes provenientes de él. Lo podemos comprobar en la pantalla principal donde pone “Connection Status” su estado debería ser “Connected”. Es indispensable que el dump1090 este ejecutándose y que la opción de red este activada (--net). Una vez comprobado el estado pinchamos en la URL y se nos abrirá el mapa en el navegador. La ilustración 13 es un ejemplo de cómo se vería el mapa en un navegador, el cual se actualizaría cada segundo.

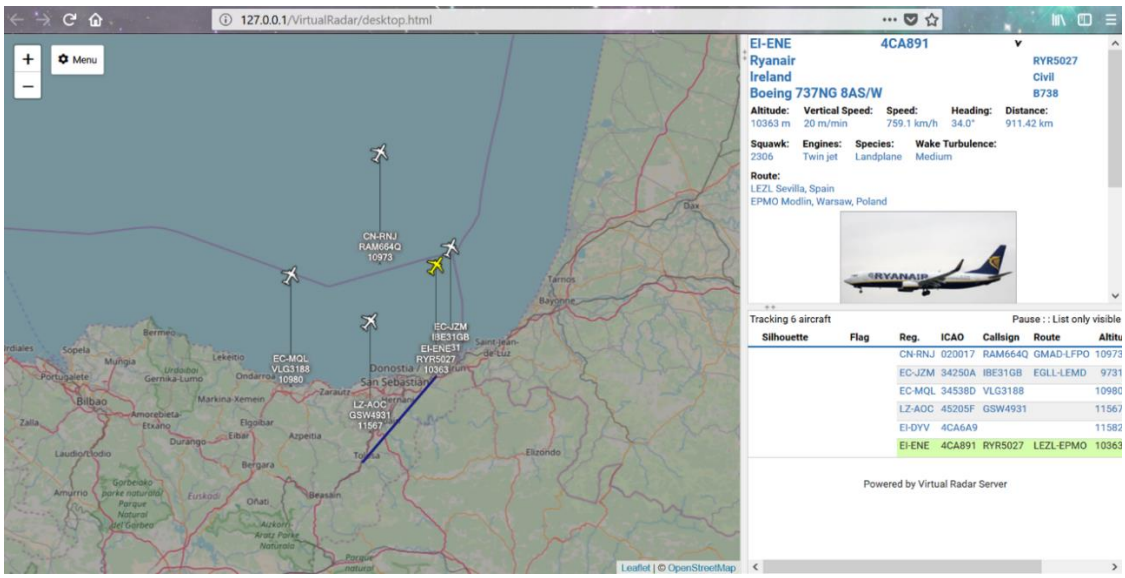


Ilustración 13: Captura de pantalla de lo visualizado mediante VRS

7.3 Instalación del nuevo objetivo

Para dar solución al nuevo objetivo y hacer una comparativa de distancias entre la nueva antena y la que proviene por defecto se ha decidido capturar los datos provenientes del dump1090 para después hacer una comparativa. Para llevar a cabo este objetivo la arquitectura seguida ha sido como la de la ilustración 14.

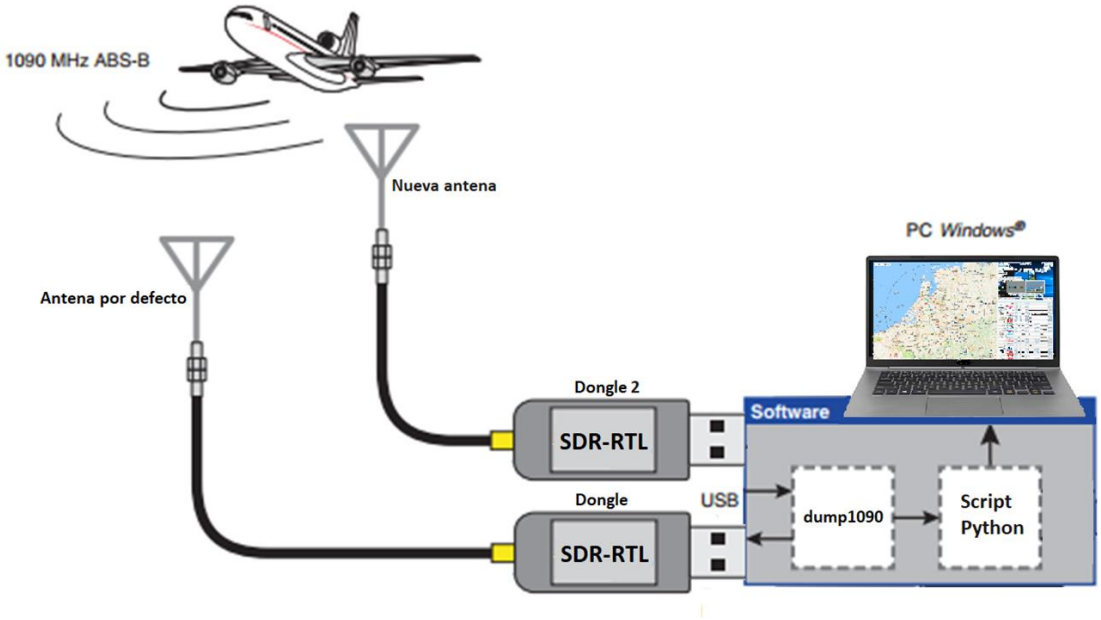


Ilustración 14: Arquitectura del segundo objetivo

El primer paso dado para este objetivo ha sido escribir el script necesario para captar los datos. En él se han capturado los datos provenientes del dump1090 en formato "json", debido a que el decodificador crea un archivo llamado "data. json" y lo envía al host local (127.0.0.1) por el puerto 8080 cada segundo. Este archivo contiene los siguientes datos por cada aeronave:

- Hex → La dirección ICAO que es única en cada aeronave
- Squawk → Un código seleccionado en el transpondedor
- Flight → Identifica un vuelo en particular (una aeronave en particular puede volar en varios vuelos diferentes en un día)
- Lat → Latitud
- Lon → Longitud
- Validposition → Bandera para la posición
- Altitude → Altitud
- Vert_rate → Dato del varioaltímetro
- Track → Ángulo de la aeronave
- Validtrack → Bandera para el ángulo
- Speed → Velocidad de la aeronave
- Messages → Cantidad de paquetes captados hasta el momento
- Seen → Cuanto tiempo ha paso desde que esta visible su última posición

A la dirección local se enviará cada segundo los datos de las aeronaves detectadas, hasta desaparecer del "json" una vez pasados 300 segundos de su último paquete de posición recibido (línea 165).

Mediante el script se guardan los datos cada segundo de posición y altitud de cada aeronave en una lista. En el código del dump1090 (líneas entre 375 y 417) se puede contemplar que en la estructura utilizada por el dump1090 donde guarda la información de los mensajes decodificados sólo se almacena la latitud y la longitud cuando DF es 17 o 18.

Para evitar que los paquetes que no contengan la posición se almacenen, es decir los que tienen DF distinto a 17 o 18 y aquellos mensajes "extended squitter" que no sean los que contienen la posición, se hará uso de la bandera de posición. Básicamente, mediante esta bandera sabremos los paquetes que contienen la posición y los que no. Esto es importante porque los que tienen la bandera bajada no tendrán posición real, pero como dato nos pasarán las coordenadas [0.00 , 0.00]. Con esto evitaremos guardar en la lista esa posición y además tener una lista más optimizada. El problema al saturar la lista con datos no relevantes es, evidentemente, que es información inútil para nuestro objetivo, pero también que la lista puede ser demasiado grande y provocar retrasos en el bucle de captura. Para revisar este tiempo se imprime en pantalla la diferencia de tiempo del inicio y final de cada iteración de bucle. Este número debería rondar el segundo, porque utilizamos la función "sleep" con un segundo cada interacción (dump1090 envía el. "json" cada segundo). Otro problema de optimización que tiene relevancia y se ha evitado en el script, ha sido no repetir la última posición de las aeronaves. Muchas veces ocurre que el "json" sigue albergando los datos de la última posición (teniendo la bandera de posición validada) y nos lleve a guardar las mismas coordenadas más de una vez, saturando la lista con información irrelevante. Al final se consigue una lista con todas las coordenadas de cada aeronave captadas durante el tiempo que está ejecutándose el script, sin tener posiciones repetidas y con valor nulo. El tiempo de duración del script es configurable para cada captura. Todo este proceso se realiza en la función "capturar" del script.

$$nombre\_y\_coordenadas = [[[hex], [coordenadas\_0], [coordenadas\_1] \dots], [[hex], \dots] \dots]$$

La segunda parte del script le corresponde a la función "crear_mapa", en ella se crea el mapa necesario para visualizar las aeronaves de la lista y ver los datos de las máximas distancias detectadas. Esta función se ejecuta después de que toda la lista de las coordenadas esté completada. Como ya se ha comentado en el análisis se hace uso de la librería "folium" [20], que nos permite crear mapas en el navegador de forma sencilla en Python. Para la descarga de la imagen del mapa es necesaria la conexión a Internet, por lo que necesitaremos Internet para la visualizar los datos en un mapa. Se especifica la posición donde tenemos colocado nuestro receptor y después se insertan los datos que se quieren dibujar. Primero se hacen los cálculos de la máxima distancia detectada respecto a nuestro receptor para cada aeronave. Una vez conseguida la distancia máxima de cada aeronave se dibujará un circulo con ese radio para cada una de ellas. El color del círculo será dependiente de la distancia respecto al receptor. Para este cálculo utilizamos la fórmula de Harvesine implementada en Python.

$$d = 2 * R * \text{asin}\sqrt{\sin^2\left(\frac{\Delta lat}{2}\right) + \cos(lat1) * \cos(lat2) + \sin^2\left(\frac{\Delta lon}{2}\right)}$$

Después se dibujará el recorrido de cada aeronave en el mapa con las coordenadas de la lista. Para el color de cada aeronave se utiliza el color que corresponde a su código ICAO en hexadecimal. Esta relación no tiene ningún objetivo concreto, pero dado la compatibilidad de formatos entre el código ICAO en hexadecimal y la paleta de colores en hexadecimal, permite atribuir a cada aeronave un color. Para terminar con la función "folium", se coloca un elemento emergente en el lugar de nuestro receptor el cual contiene; la distancia máxima detectada en toda la lista, la cantidad de aeronaves detectadas y máxima distancia media detectada. Permitiendo visualizar estos datos en nuestro mapa y deduciendo de manera rápida la antena con la mayor distancia detectada. Los datos del "folium" se guardarán en un HTML, que puede ser abierto en el navegador en cualquier momento.

Para terminar el script, se ha creado una función el cual guarda los datos de la lista en un archivo. Estos datos pueden ser útiles para posteriores usos con el procesamiento de datos ADS-B.

Para llevar a cabo esta prueba ha sido necesario el uso de un segundo dongle (con instalación del Zadig incluida) y en el script, realizar una serie de modificaciones para adaptarlo a nuestras necesidades. Como nos interesa que los datos capturados sean exactamente los mismos, la iniciación de la función "capturar" debe ser al mismo tiempo. Esto desde un script, llamando a la función una detrás de otra no es posible. Una posibilidad podría ser llamar al script dos veces de manera manual, pero esta opción no ha sido escogida por la falta de exactitud. Para resolver este problema se ha optado por usar el multiproceso de los procesadores. Esta posibilidad nos la ofrece la librería "Process (from multiprocessing)" de Python.

También hay que mencionar un par de detalles respecto al software dump1090. Al realizar el proceso de pruebas en el primer objetivo, se usaba el comando "dump1090 --net". Si volvemos a utilizar el mismo comando, no pasará automáticamente al segundo dongle. Para resolver esta situación tenemos que modificar el comando y utilizar uno que llame al segundo dispositivo. Insertando "dump1090 --net --device-index 1" damos con la solución. Como es lógico tendremos dos ventanas de comandos abiertas al mismo tiempo que están constantemente mostrando paquetes recibidos en la frecuencia 1090 MHz. Si antes utilizábamos una dirección IP local desde donde se recibían los datos "json" ahora esta dirección no puede ser válida para los dos dongles. Es decir, cada dongle tiene que enviar el archivo "data.json" a una dirección IP diferente, ya que en caso contrario se estarían sobreponiendo constantemente. Para evitar este problema hemos añadido un nuevo comando dump1090 que asigna una nueva dirección local al dongle nuevo. El comando final nos quedaría así: "dump1090 --device-index 1 --net --net-bind-address 127.0.0.100".

7.3.1. Proceso de pruebas del nuevo objetivo

Teniendo todas las herramientas necesarias para realizar las pruebas se ha dado paso a efectuar las pruebas de campo. Estas pruebas se han realizado en una ubicación diferente a la anterior (debido a los errores sobre el emplazamiento anterior) y se reflejaran en una tabla. En ella se apreciarán las diferentes capturas del script, la cantidad de aeronaves detectados, la distancia máxima detectada y la distancia media máxima detectada. Se crearán dos mapas por captura (como los de la ilustración 16), uno realizado con los datos del dongle junto a la antena por defecto y otro con los datos del dongle junto a la antena Spider (Anexos). Ejecutamos los decodificadores para los dos dispositivos y posteriormente el script.

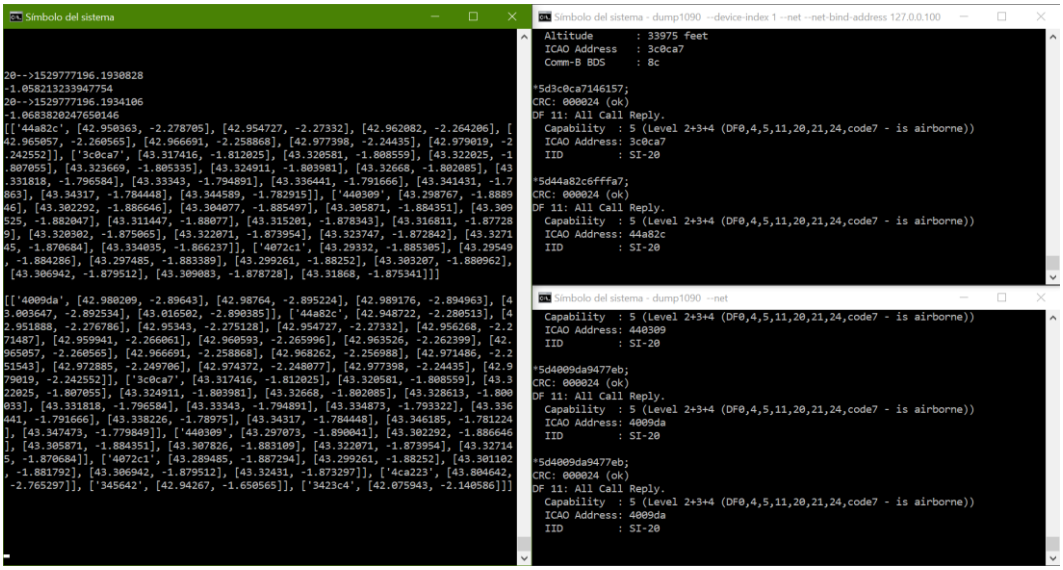


Ilustración 15: Captura de pantalla de las ventanas de comandos al realizar las pruebas de campo

La recepción de paquetes recibidos por el dump1090 es inmediata. Ejemplo en la ilustración 15 con captura de pantalla del momento donde están ejecutándose los dos dongle en dump1090 y el script. El script ha sido ejecutado en cuatro capturas y cada una de ellas con una duración de 750 segundos. La diferencia de tráfico aéreo entre capturas es inevitable, no se puede controlar las condiciones del "ADS-B out". Por ejemplo, los transpondedores de las aeronaves emiten ADS-B con diferentes potencias, debido a la altura o la condición meteorológica, está estipulado un máximo y un mínimo para cada transpondedor y situación [21]. Esto nos obliga a suponer ciertas condiciones para nuestras conclusiones. En cambio, las dos antenas tienen exactamente las mismas condiciones por captura, con una distancia entre ellas despreciable y por lo tanto esta comparativa se pueden deducir conclusiones directas.

Tiempo de captura	Cantidad de aeronaves		Distancia máxima (km)		Distancia máxima media (km)	
	Ant. defecto	Ant. Spider	Ant. defecto	Ant. Spider	Ant. defecto	Ant. Spider
750	15	23	156.7978	156.3191	93.8275	110.0507
750	11	17	120.9168	162.2711	80.4672	93.3888
750	13	25	130.3500	205.4646	83.0422	103.0239
750	21	26	130.0635	217.2986	95.3942	106.8337

Tabla 2: Datos obtenidos en las pruebas de campo con las dos antenas

A primera vista se observa en la tabla 2 que la cantidad de aeronaves detectadas por la antena Spider es superior en todas las capturas. Se puede decir que la antena por defecto ha detectado menos aeronaves en todas ellas y por lo tanto su capacidad de detección es menor. Siguiendo la misma tónica la distancia máxima detectada por la antena Spider ha sido superior a excepción de la primera captura, donde por menos de un kilómetro la antena por defecto ha sido superior. Esta excepción sólo ocurre en la primera captura, pero en el resto de ellas la diferencia se bastante significante. Con la distancia máxima media detectada ocurre igual que con las otras medidas, donde la antena Spider supera en todas las capturas las distancias de la antena por defecto. Esta última medida es un dato más fiable para saber el aérea de detección de nuestras antenas, ya que no es dependiente de una aeronave en concreto que resulte coincidir con la mejor dirección de apuntamiento de nuestras antenas. Además, como ya hemos mencionado, los transpondedores de las aeronaves emiten la señal en un rango de potencias variable (con un máximo y un mínimo) y por lo tanto una detección de una aeronave que envía en el máximo puede darnos a entender que puede detectar todas las aeronaves a esa distancia.

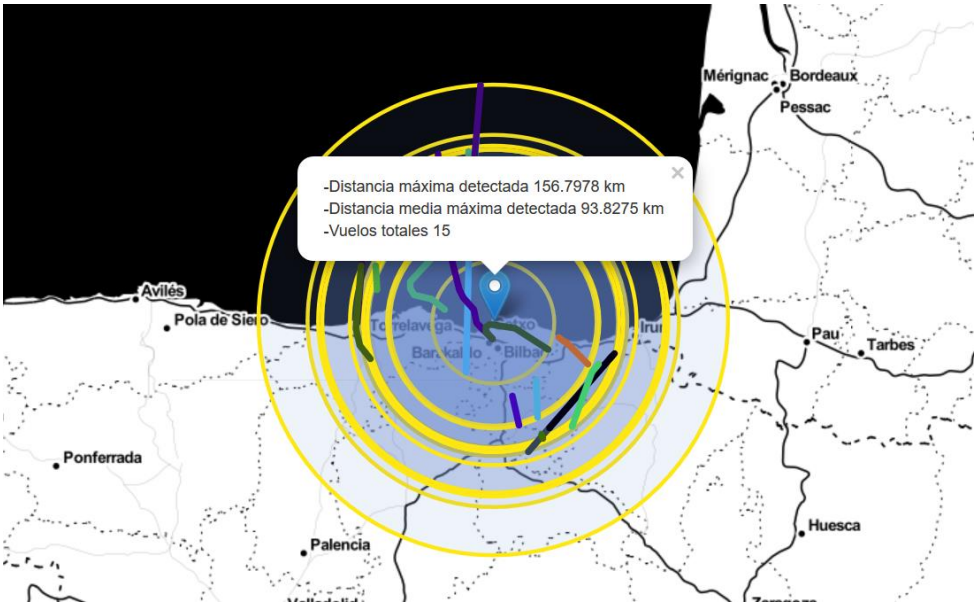


Ilustración 16: Mapa creado en la primera captura con la antena por defecto

Se puede percibir con más detalle en los mapas creados (13.4 y 13.5 en anexos), donde en la antena por defecto se percibe una mayor concentración de circunferencias (refleja la distancia máxima detectada por cada aeronave en una captura) cerca del lugar de recepción. Por el contrario, en los mapas de la antena Spider la concentración de estas circunferencias esta más alejada del centro, derivando en una distancia máxima media mayor. En los mapas se observa perfectamente el recorrido de las aeronaves y su entrada o salida del aeropuerto de Loiu.

Para terminar, hay que mencionar el hecho de estar cerca de un aeropuerto y a una distancia mayor otros dos (Gasteiz y Santander), por tanto, esto aumenta el trafico considerablemente. En la ilustración 16 se puede observar un mayor flujo de aeronaves en los puntos donde están localizados esos tres aeropuertos. Para ver la optimización de nuestro script se han comprobado los tiempos de cada iteración (ilustración 17) y evitar unos tiempos que no

coincidiesen con los aportados por el dump1090 con el “json”. El cálculo para visualizar el tiempo de retardo de cada iteración demuestra que las capturas no han sufrido ningún retraso significativo respecto a las realizadas al principio. Como se observa, en la última iteración la duración es de 1.05562042 segundos, un segundo que corresponde a la función “sleep” y el resto del tiempo del procesamiento del script. El retraso por la lista es despreciable.

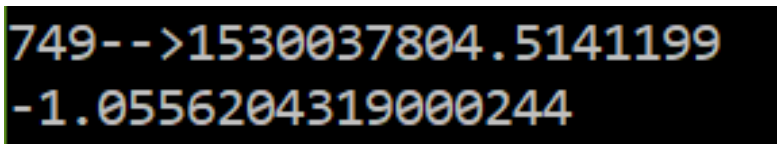


Ilustración 17: Captura de pantalla de los tiempos de bucle de la última iteración de la última captura

8. Descripción de tareas

A la hora de hacer la descripción de las tareas, se tendrá en cuenta que el trabajo se planteó con un objetivo, pero a medida que se analizaba ese objetivo inicial se decidió ampliar el proyecto con un segundo objetivo.

Así, el trabajo tanto del primero como del segundo objetivo se llevó a cabo en diferentes paquetes. En esos paquetes tuvo especial peso el trabajo de investigación previo que hubo que hacer para poder desarrollar las metas. Cada paquete de trabajo tiene un tiempo fijo estimado que hay que cumplir. Después de cada paquete de trabajo se realizaron los entregables del trabajo realizado en ese paquete.

PAQUETES DE TRABAJO

- Análisis del hardware
  En este paquete se analizarán los diferentes dispositivos SDR que hay en el mercado para llevar a cabo nuestro propósito. Se buscará información referente a sus características y se evaluaran otra serie de criterios establecidos.
  Tiempo: 21 días (15 horas)
  Entregable: Informe análisis de alternativas hardware

- Análisis del decodificador
  En este paquete se buscará software que pueda decodificar los datos IQ provenientes del SDR seleccionado. Su elección tiene especial importancia lleva es modificado por la decisión de ampliar los objetivos.
  Tiempo: 13 días (12 horas)
  Entregable: Informe análisis de alternativas software de decodificadores

- Análisis del localizador gráfico
  En este paquete se buscará una alternativa fácil y grafica para poder visualizar las aeronaves decodificadas en un mapa.
  Tiempo: 6 días (5 horas)
  Entregable: Informe análisis de alternativas software de localizadores gráficos

- Análisis del segundo objetivo
  En este paquete se buscará las diferentes alternativas que existen para poder realizar el nuevo objetivo. La correcta selección de las alternativas nos permitirá realizar de una manera mucho más rápida nuestros objetivos.
  Tiempo: 11 días (20 horas)
  Entregable: Informe análisis del segundo objetivo

- Compra del material

En este paquete nos dispondremos a comprar todo el material hardware necesario para nuestro proyecto. Hay materiales que se compraran por internet y otros en tiendas físicas.
Tiempo: 26 días (3 horas)
Entregable: No tiene

- Instalación del objetivo principal
  En este paquete se instalará todo lo necesario para poder realizar el primer objetivo. Tanto la compra de material, como el análisis del segundo objetivo son indispensables antes de este paquete.
  Tiempo: 6 días (4 horas)
  Entregable: Informe sobre la instalación del objetivo principal

- Instalación objetivo secundario
  En este paquete se instalará todo lo necesario para poder realizar el objetivo secundario. Si bien hay software y hardware que se repite en este objetivo, hay nuevos programas que son necesarios.
  Tiempo: 16 días (6 horas)
  Entregable: Informe sobre la instalación del objetivo secundario

- Creación script Python
  En este paquete se programará en Python todo lo necesario para llevar a cabo nuestro objetivo secundario. Es un paquete que necesita de un tiempo relativamente alto por todas las pruebas que hay que realizar durante su creación.
  Tiempo: 21 días (21 horas)
  Entregable: Capturas del script e explicación del código

- Pruebas de campo
  En este paquete se realizarán las pruebas de campo necesarias, una conseguidas todas las instalaciones de los objetivos y probados su funcionalidad se empezará con este paquete. Es un paquete relativamente largo por el tiempo de traslados a las posiciones y porque los errores a la hora de tomar las pruebas son bastantes probables.
  Tiempo: 41 días (25 horas)
  Entregable: Informe sobre las pruebas de campo de ambos objetivos

- Solución a problemas
  En este paquete se tendrá en cuenta el posible tiempo usado en resolver los problemas surgidos durante el resto de paquetes. Pueden ser en cualquier fase del proyecto. Es un paquete de una duración larga ya que los errores pueden llevar a la realización de las pruebas de campo o a la reinstalación de algún software más óptimo.
  Tiempo: 102 días (45 horas)
  Entregable: Informe sobre soluciones adoptadas

- Documentación del proyecto

En este paquete se escribirá todo lo necesario para escribir el informe final sobre el proyecto. En él se releerán todos los informes realizados, se modificará lo necesario y se crearán el resto de apartados para su realización.
optimo.
Tiempo: 175 días (60 horas)
Entregable: Informe final

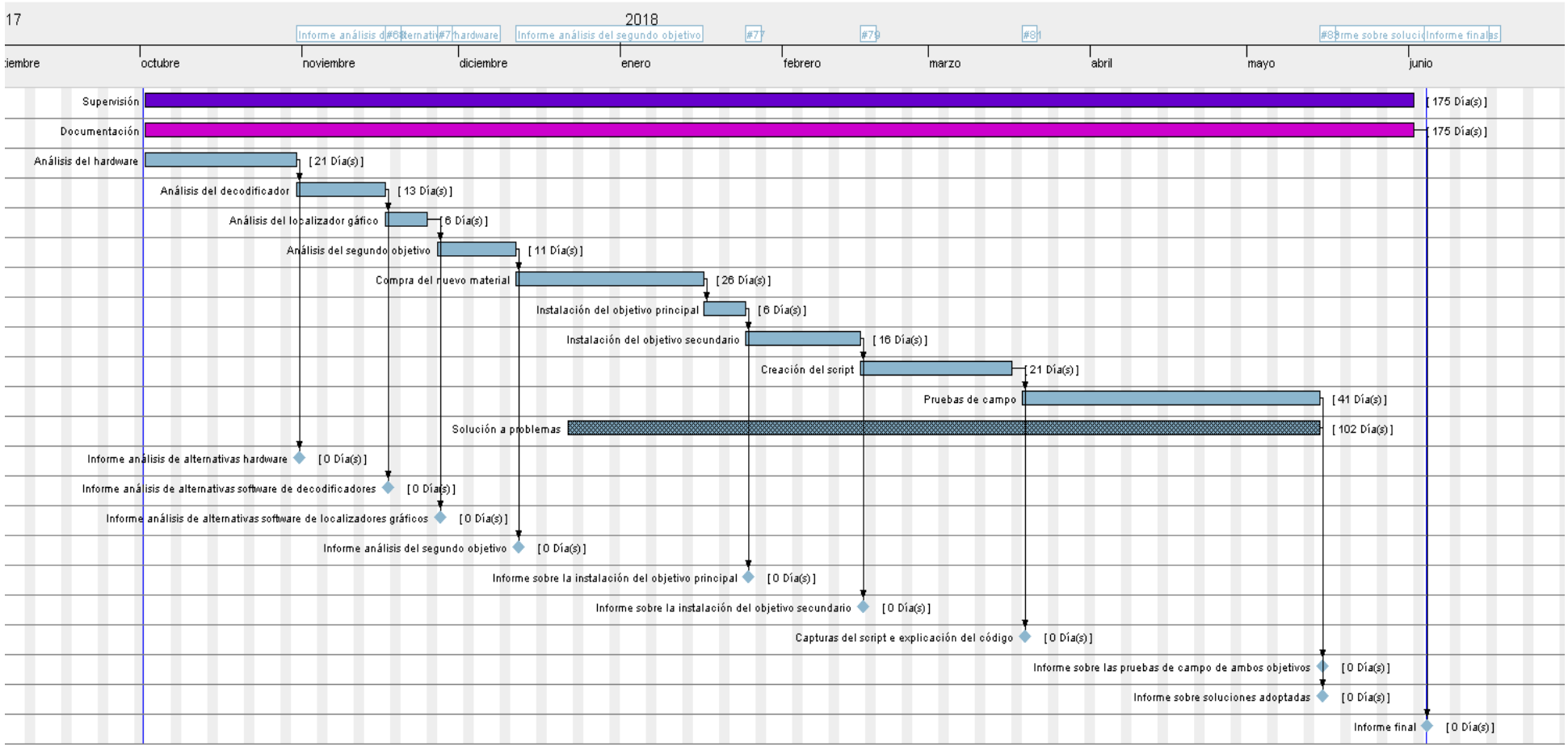


Ilustración 18: Diagrama de Gantt

9. Presupuesto

Para la realización del presupuesto se hará una evaluación tanto del esfuerzo como del material para llevar a cabo este proyecto. Para ello, se tendrán en cuenta los recursos humanos y los recursos materiales que han sido necesitados para la creación de este proyecto.

El trabajo humano ha sido realizado por dos personas y se puede intuir que gran parte de los costes serán derivados de estos. El software utilizado no tiene coste puesto que es software libre. La mayor parte del coste no humano será material, aunque en este apartado tampoco habrá unos costes excesivos, ya que en las alternativas se ha utilizado el precio como un factor de decisión.

9.1 Horas internas

Responsabilidad	Tasa horaria (€/h)
Ingeniero Junior	25
Ingeniero Senior (Director de proyecto)	40

Tabla 3: Tasas horarias

Paquetes	Responsable	Nº de horas (h)
Análisis del hardware	Ingeniero Junior	15
Análisis del decodificador	Ingeniero Junior	12
Análisis del localizador gráfico	Ingeniero Junior	5
Análisis de la solución para el segundo objetivo	Ingeniero Junior	20
Compra del material	Ingeniero Junior	3
Instalación objetivo principal	Ingeniero Junior	4
Instalación objetivo secundario	Ingeniero Junior	6
Creación script Python	Ingeniero Junior	21
Pruebas de campo	Ingeniero Junior	25
Solución a problemas	Ingeniero Junior	45
Documentación del proyecto	Ingeniero Junior	60
TOTAL		216
Supervisión	Ingeniero Senior	124
TOTAL		124

Tabla 4: Horas internas

Responsabilidad	Tasa horaria (€/h)	Nº de horas (h)	Coste (€)
Ingeniero Junior	25	216	5400
Ingeniero Senior (Director de proyecto)	40	124	4960
TOTAL			10360

Tabla 5: Subtotal horas internas

9.2 Amortizaciones

Instrumento	Precio inicial (€)	Valor residual (€)	Vida útil (años)	Uso (meses)	Coste (€)
RTL-SDR (x2)	26,26	8	2	8	6,09
Antena Spider	4	0	5	8	0,53
Ordenador portátil LG	1000	200	5	8	106,67
Soldador	40	15	10	8	1,67
Material de soldadura	5	0	1	8	3,33
VRS	0	0	8	8	0,00
dump1090	0	0	8	8	0,00
Zadig	0	0	8	8	0,00
TOTAL					118,29

Tabla 6: Subtotal amortizaciones

9.3 Gastos

Concepto	Coste (€)
Electricidad	80
Conexión a Internet	200
Transporte para las medidas	10
Material de oficina	20
TOTAL	310

Tabla 7: Subtotal gastos

9.4 Total

Concepto	Subtotal (€)
Horas internas	10360
Amortizaciones	118,29
Gastos	310
TOTAL	10788,29

Tabla 8: Coste total proyecto

Finalmente, el coste total del proyecto es de 10788.29 €. La mayoría del gato proviene de las horas internas realizadas entre las dos personas implicadas. Para la realización del proyecto se había tenido en cuenta el precio de los dispositivos utilizados para adquirir los más asequibles y prácticos posibles. Ese requerimiento ha tenido un impacto directo en el bajo gasto de las amortizaciones.

10. Riesgos del TFG

Todos los proyectos tienen una serie de circunstancias que pueden poner en peligro el correcto desarrollo del mismo. En este apartado se describirán una serie de riesgos que se pueden llegar a dar durante la realización de este TFG.

Para este proyecto, uno de los mayores riesgos es debido a la falta de control de las condiciones del "ADS-B out", porque esas condiciones afectan plenamente a las pruebas de campo que son necesarias a para comprobar los dos objetivos. La falta de control en esa situación puede llevar a equivocaciones a la hora de recibir los paquetes de las aeronaves, pudiendo no recibir ningún paquete por falta de aeronaves cercanas o por causas externas. Si las pruebas se realizan cerca de un aeropuerto, nos aseguramos que probablemente consigamos recibir paquetes ADS-B. En caso de no realizarlas cerca de un punto con gran tráfico aéreo el riesgo de no recibir ningún paquete ADS-B aumenta. Al mismo tiempo la falta de recepción de paquetes ADS-B por causas de mala configuración del software o del incorrecto funcionamiento del hardware son existentes.

La falta de control también puede suponer un riesgo a la hora de comprobar la viabilidad de la antena en el analizador de redes. Para una correcta medición sería necesario el uso de una cámara anecoica, pero no se dispone de ella y por lo tanto hay que buscar una localización donde la antena reciba la mínima cantidad de señales en la frecuencia seleccionada en el analizador.

11. Conclusiones

Este proyecto se inició con el propósito de implementar un radar SDR localizador de aviones comerciales mediante la recepción de señales ADS-B y posteriormente fue ampliado con un segundo objetivo el cual buscaba mejorar la antena existente ampliando el área de recepción de la anterior. Para llevar a cabo esa demostración se decidió hacer una comparativa entre ambas, mediante una serie de mapas y una tabla comparativa.

Para poder llevar a cabo el primer objetivo fue necesario comprender el funcionamiento de ciertos aspectos técnicos del ADS-B y así poder desenvolverse con más facilidad con las herramientas utilizadas. El objetivo se pudo cumplir y se demostró que las herramientas seleccionadas fueron suficientes para localizar las aeronaves mediante las señales ADS-B. En la elección de las herramientas utilizadas, se hizo hincapié en buscar soluciones accesibles a nivel tanto económico como de soporte de la herramienta en cuestión, para poder usar el sistema en entornos como puede ser el académico. Esta posibilidad se ha tenido en cuenta durante toda la realización y el proyecto cuenta con todo lo necesario para poder hacerlo.

Vista la facilidad con la que se alcanzó el primer objetivo y las ganas de ampliar un proyecto interesante desde el punto de vista personal, se decidió ampliarlo con un objetivo que implicase un conocimiento más técnico del sistema ADS-B. Para llevar a cabo el segundo objetivo la creación del script fue un punto esencial que permitió demostrar si nuestra nueva antena cumplía nuestros objetivos. Mediante los mapas creados con el script se pudo comprobar que el segundo objetivo también se llevó a cabo con éxito. Para realizar las pruebas de campo y conseguir los datos necesarios se tuvo que cambiar de emplazamiento respecto al primer objetivo, ya que como se describía en el apartado de riesgos no podemos controlar las condiciones donde se realizan las pruebas y en este caso hubo una interferencia indeseada que nos impedía la correcta recepción de las señales. Durante la realización de este objetivo se indago más en el formato de los paquetes ADS-B, lo que permitió ampliar el conocimiento técnico de los mismos. La utilización de materiales accesibles para facilitar su uso por terceros también estuvo presente durante este objetivo.

12. Fuentes de información

[1] Wikipedia, "Teoría de operación," 2018. [Online]. Available: https://es.wikipedia.org/wiki/Sistema_de_Vigilancia_Dependiente_Automática#cite_note-12.

[2] Requirement Focus Group, "Information from ANC/11 related to ADS-B, ASAS and ACAS," Montreal, 2004.

[3] Eurocontrol, "Avionics requirements for civil aircraft," 2017.

[4] AESA, "El Sistema ADS." [Online]. Available: https://www.seguridadaerea.gob.es/lang_castellano/navegacion/programas/ads/sistema_ads.aspx.

[5] AviationWeek, "FAA Worried About ADS-B, 1090 MHz Interference," *May 2016*, 2016. [Online]. Available: http://aviationweek.com/commercial-aviation/faa-worried-about-ads-b-1090-mhz-interference.

[6] S. C. 186, "Minimum Operational Performance Standards for 1090 MHz Extended Squitter Automatic Dependent Surveillance – Broadcast (ADS -B) and Traffic Information Services – Broad cast (TIS -B)," Washington, DC.

[7] ICAO, "Technical Provisions for Mode S Services and Extended Squitter."

[8] Eurocontrol, "ADS-B for Dummies." .

[9] NXP, "LPC4370 Product data sheet." 2013.

[10] MAXIM, "MAX5864 Product data sheet." .

[11] Community, "RTL-SDR." [Online]. Available: https://www.rtl-sdr.com/.

[12] "DVB-T VHF UHF combo magnetic TV digital antenna." [Online]. Available: http://asian-creation.manufacturer.globalsources.com/si/6008840337938/pdtl/DVB-T-antenna/1153021756/Magnetic-TV-digital-antenna.htm.

[13] P. Batard, "Zadig," *Neosurge*. [Online]. Available: http://zadig.akeo.ie/.

[14] "antirez/dump1090," *Github*, 2012. .

[15] MalcolmRobb, "bovine/dump1090-robb," *Github*. [Online]. Available: https://github.com/bovine/dump1090-robb.

[16] F. S. Foundation, "GNU General Public License," *2016/11/18*, 2016. [Online]. Available: https://www.gnu.org/licenses/gpl-3.0.en.html.

[17] O. Communities, "The 3-Clause BSD License." [Online]. Available: https://opensource.org/licenses/BSD-3-Clause.

[18] A. Whewell, "Virtual Radar Server," *2010*. [Online]. Available: http://www.virtualradarserver.co.uk/PrivacyPolicy.aspx.

[19] "MÉTODOS PARA LA CARACTERIZACIÓN DE ANTENAS." [Online]. Available: http://tesis.uson.mx/digital/tesis/docs/20997/Capitulo4.pdf.

[20] Github, "python-visualization/folium." [Online]. Available: https://github.com/python-visualization/folium.

[21] S. C. 186, "Minimum Operational Performance Standards for 1090 MHz Extended Squitter Automatic Dependent Surveillance – Broadcast (ADS -B) and Traffic Information Services – Broad cast (TIS -B)." Washington, DC, p. 801.

13. Anexos

13.1 Código script

```
import json
import urllib.request
import matplotlib.pyplot as plt
import folium
import numpy as np
from datetime import datetime
import time
import math
from multiprocessing import Process
import pickle


def haversine(lat1, lon1, lat2, lon2):
    rad=math.pi/180
    dlat=lat2-lat1
    dlon=lon2-lon1
    R=6372.795477598
    a=(math.sin(rad*dlat/2))**2 + math.cos(rad*lat1)*math.cos(rad*lat2)*(math.sin(rad*dlon/2))**2
    distancia=2*R*math.asin(math.sqrt(a))
    return distancia

def color_por_distancia(dist):
    if dist < 10:
        return '#fff58e'
    elif 10 <= dist < 20:
        return '#fff163'
    elif 10 <= dist < 15:
        return '#ffee4c'
    elif 15 <= dist < 20:
        return '#ffeb35'
    else:
        return '#ffe711'

def capturar(link,seg,nombre_archivo):

    pos = None
    num = 1
    ordua = time.time()
def capturar(link,seg,nombre_archivo):

    pos = None
    num = 1
    ordua = time.time()
    dif = time.time()
    nombre_aviones=[]
    nombres_y_coordinadas=[]

    while num<seg:

        #Se descarga la información de la dirección "link"
        urlData = link
        webURL = urllib.request.urlopen(urlData)
        data = webURL.read()
        encoding = webURL.info().get_content_charset('utf-8')
        data_json=json.loads(data.decode(encoding))

        for flight in data_json:
            if flight['validposition']: #Para optimizar evitando valores de posicion [0.00 , 0.00]
                try:
                    pos=nombre_aviones.index(flight['hex'])
                    ultima_pos=nombres_y_coordinadas[pos][len(nombres_y_coordinadas[pos])-1]
                    #if ultima_pos != [flight['lat'],flight['lon']]: #Para optimizar evitando repetir la última
                    nombres_y_coordinadas[pos].append([flight['lat'],flight['lon']])
                except ValueError:
                    print("Este avion no esta registrado en la lista, añadiendo...")
                    nombre_aviones.append(flight['hex'])
                    nombres_y_coordinadas.append([flight['hex'],[flight['lat'],flight['lon']]]) #Añade la aeronc

        time.sleep(1) #Como el archivo .json es enviado cada segundo por el dump1090 nos interesa igualar la ite

        print("%s-->%s" % (num, ordua))
        dif = ordua - time.time()
```

```
        print(dif) #Para comprobar el correcto funcionamiento
        print(nombres_y_coordinadas)
        ordua = time.time()
        num = num + 1
        print("\n")

    crear_mapa(nombres_y_coordinadas,nombre_archivo)
    guardar_datos(nombres_y_coordinadas,nombre_archivo)

def crear_mapa(nombres_y_coordinadas,nombre_archivo):
    URUMEA = (43.3094, -1.9782)

    m = folium.Map(location=URUMEA,tiles = "Stamen Toner", zoom_start=10) #Colocación del centro

    print(nombres_y_coordinadas)

    ###CÁLCULO DE MÁXIMA DISTANCIA E INSECIÓN EN MAPA DEL CÍRCULO
    lon1=URUMEA[1]
    lat1=URUMEA[0]
    d=0
    d_max=0
    d_max_max=0
    d_max_suma=0
    d_max_media=0
    cont_aeronaves=0

    for val in nombres_y_coordinadas:
        for val2 in val:
            if len(val2)!= 6: #Se evita utilizar la dirección ICAO
                d=haversine(lat1, lon1, val2[0], val2[1])
                if d > d_max:
                    d_max = d #Distancia máxima conseguida por trayecto
        if(len(val)>2): #Se evitan las aeronaves que solo tienen una coordenada
            d_max_suma = d_max_suma + d_max
            cont_aeronaves = cont_aeronaves + 1
            folium.Circle(location=URUMEA,
                          radius=d_max*1000,
                          color=color_por_distancia(d_max),
```

```
        if(len(val)>2): #Se evitan las aeronaves que solo tienen una coordenada
            d_max_suma = d_max_suma + d_max
            cont_aeronaves = cont_aeronaves + 1
            folium.Circle(location=URUMEA,
                          radius=d_max*1000,
                          color=color_por_distancia(d_max),
                          fill=True,
                          fill_color='#6182ca',
                          fill_opacity=0.1,
                          weight=3
                         ).add_to(m)
            if d_max > d_max_max:
                d_max_max = d_max #Distancia máxima conseguida por captura
        d_max=0

        #Para evitar error si no se ha detectado nada
        if cont_aeronaves!=0:
        d_max_media = d_max_suma/cont_aeronaves

    ###INSERCIÓN EN MAPA DE LOS RECORRIDOS CON COLOR ÚNICO PARA CADA TRAYECTO

    coordenadas=[]
    for val in nombres_y_coordinadas:
        for val2 in val:
            if val2 != [0.0, 0.0]: #Con el uso de 'validposition' esto es redundante
                coordenadas.append(val2)
        color_por_avion='#%s'%coordenadas[0] #Color en función del nombre "hex"/ICAO
        coordenadas.pop(0) #Se elimina el nombre "hex"/ICAO
        k=coordenadas
        folium.PolyLine(locations=k,color=color_por_avion,weight=5).add_to(m)
        coordenadas=[]

    folium.Marker(location=URUMEA,popup= '-Distancia máxima detectada ' + str("{0:.4f}".format(d_max_max)) + ' km<br>
                  '-Distancia media máxima detectada ' + str("{0:.4f}".format(d_max_media)) + ' km<br>' +
                  '-Vuelos totales ' + str(cont_aeronaves) + '<br>').add_to(m)
```

```
        k=coordenadas
        folium.PolyLine(locations=k,color=color_por_avion,weight=5).add_to(m)
        coordenadas=[]

    folium.Marker(location=URUMEA,popup= '-Distancia máxima detectada ' + str("{0:.4f}".format(d_max_max)) + ' km<br>' +
                  '-Distancia media máxima detectada ' + str("{0:.4f}".format(d_max_media)) + ' km<br>' +
                  '-Vuelos totales ' + str(cont_aeronaves) + '<br>').add_to(m)



    print("Guardando en el .html...")
    m.save(outfile=nombre_archivo)

def guardar_datos(nombres_y_coordinadas,nombre_archivo):

    with open(nombre_archivo+"file.txt", 'wb') as fp:
        pickle.dump(nombres_y_coordinadas, fp)

#Se usa en para leer los datos en caso de modificaciones
#def leer_datos(nombre_archivo):
#    with open (nombre_archivo, 'rb') as fp:
#        nombres_y_coordenadas = pickle.load(fp)
#    return nombre_archivo


if __name__ == '__main__':

    print("Hasiera...")

    Process(target=capturar, args=("http://127.0.0.1:8080/data.json",1000,'data_default.html',)).start()
    Process(target=capturar, args=("http://127.0.0.100:8080/data.json",1000,'data_spider.html',)).start()

    print("...Amaiera")

```

13.2 Código de dump1090.c original (líneas mencionadas)

```
    int phase_corrected;         /* True if phase correction was applied. */

    /* DF 11 */
    int ca;                      /* Responder capabilities. */

    /* DF 17 */
    int metype;                  /* Extended squitter message type. */
    int mesub;                   /* Extended squitter message subtype. */
    int heading_is_valid;
    int heading;
    int aircraft_type;
    int fflag;                   /* 1 = Odd, 0 = Even CPR message. */
    int tflag;                   /* UTC synchronized? */
    int raw_latitude;            /* Non decoded latitude */
    int raw_longitude;           /* Non decoded longitude */
    char flight[9];              /* 8 chars flight number. */
    int ew_dir;                  /* 0 = East, 1 = West. */
    int ew_velocity;             /* E/W velocity. */
    int ns_dir;                  /* 0 = North, 1 = South. */
    int ns_velocity;             /* N/S velocity. */
    int vert_rate_source;        /* Vertical rate source. */
    int vert_rate_sign;          /* Vertical rate sign. */
    int vert_rate;               /* Vertical rate. */
    int velocity;                /* Computed from EW and NS velocity. */

    /* DF4, DF5, DF20, DF21 */
    int fs;                      /* Flight status for DF4,5,20,21 */
    int dr;                      /* Request extraction of downlink request. */
    int um;                      /* Request extraction of downlink request. */
```

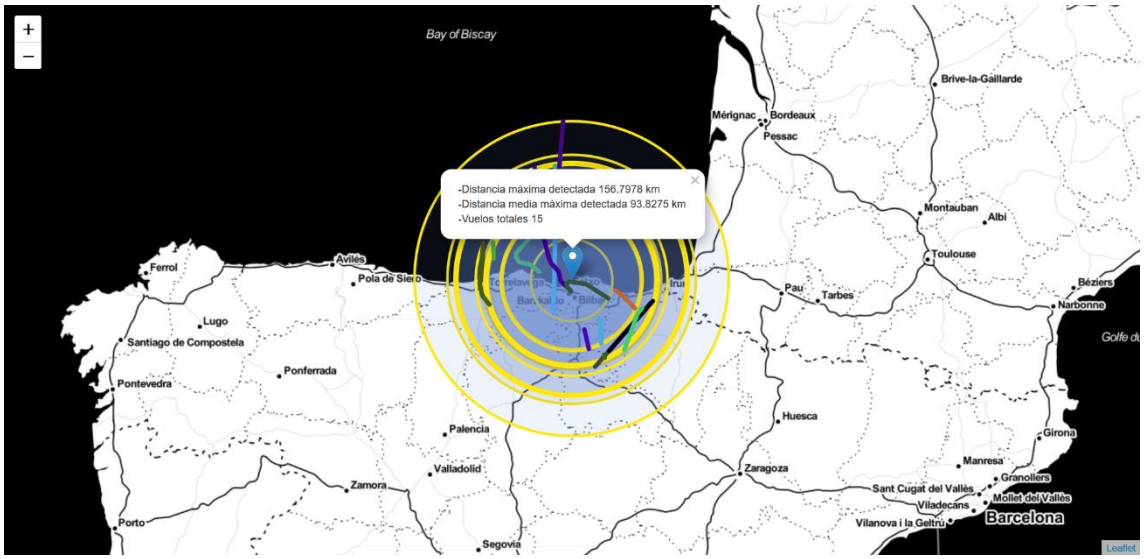
13.3 Código de dump1090.h Windows (líneas mencionadas)

```
154  #define MODES_DEBUG_GOODCRC (1<<3)
155  #define MODES_DEBUG_NOPREAMBLE (1<<4)
156  #define MODES_DEBUG_NET (1<<5)
157  #define MODES_DEBUG_JS (1<<6)
158
159  // When debug is set to MODES_DEBUG_NOPREAMBLE, the first sample must be
160  // at least greater than a given level for us to dump the signal.
161  #define MODES_DEBUG_NOPREAMBLE_LEVEL 25
162
163  #define MODES_INTERACTIVE_REFRESH_TIME 250      // Milliseconds
164  #define MODES_INTERACTIVE_ROWS          22      // Rows on screen
165  #define MODES_INTERACTIVE_DELETE_TTL   300      // Delete from the list after 300 seconds
166  #define MODES_INTERACTIVE_DISPLAY_TTL   60      // Delete from display after 60 seconds
167
168  #define MODES_NET_HEARTBEAT_RATE       900      // Each block is approx 65mS - default is > 1 min
169
170  #define MODES_NET_SERVICES_NUM          6
171  #define MODES_NET_INPUT_RAW_PORT    30001
172  #define MODES_NET_OUTPUT_RAW_PORT   30002
173  #define MODES_NET_OUTPUT_SBS_PORT   30003
174  #define MODES_NET_INPUT_BEAST_PORT  30004
```

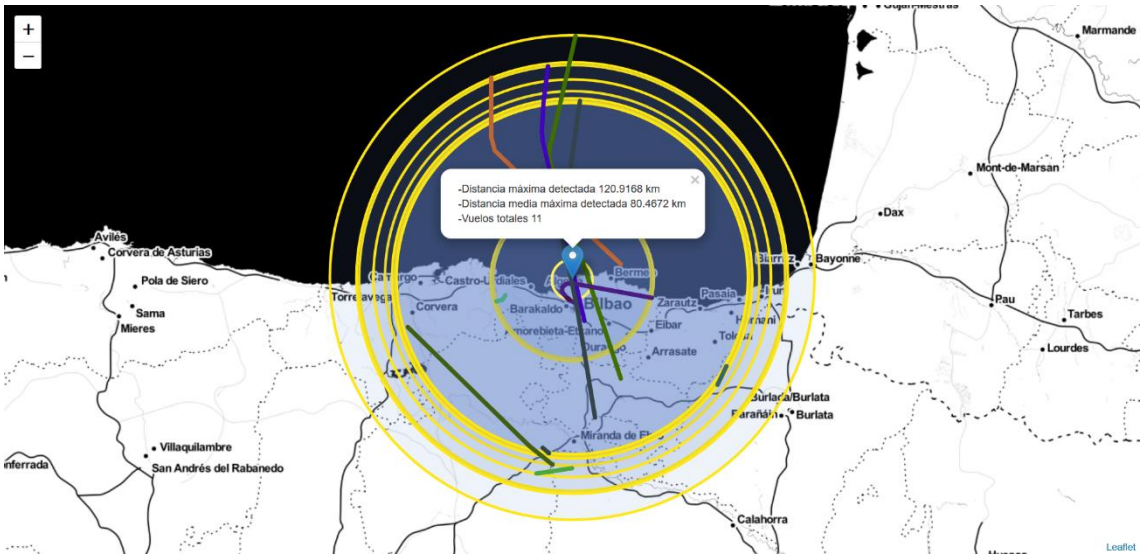
```
387      uint64_t      timestampMsg;                 // Timestamp of the message
388      int           remote;                       // If set this message is from a remote station
389      unsigned char signalLevel;                  // Signal Amplitude
390
391      // DF 11
392      int  ca;                    // Responder capabilities
393      int  iid;
394
395      // DF 17, DF 18
396      int    metype;              // Extended squitter message type.
397      int    mesub;               // Extended squitter message subtype.
398      int    heading;             // Reported by aircraft, or computed from from EW and NS velocity
399      int    raw_latitude;        // Non decoded latitude.
400      int    raw_longitude;       // Non decoded longitude.
401      double fLat;                // Coordinates obtained from CPR encoded data if/when decoded
402      double fLon;                // Coordinates obtained from CPR encoded data if/when decoded
403      char   flight[16];          // 8 chars flight number.
404      int    ew_velocity;         // E/W velocity.
405      int    ns_velocity;         // N/S velocity.
406      int    vert_rate;           // Vertical rate.
407      int    velocity;            // Reported by aircraft, or computed from from EW and NS velocity
408
409      // DF4, DF5, DF20, DF21
410      int  fs;                    // Flight status for DF4,5,20,21
411      int  modeA;                 // 13 bits identity (Squawk).
412
413      // Fields used by multiple message types.
414      int  altitude;
```

13.4 Mapas antena por defecto

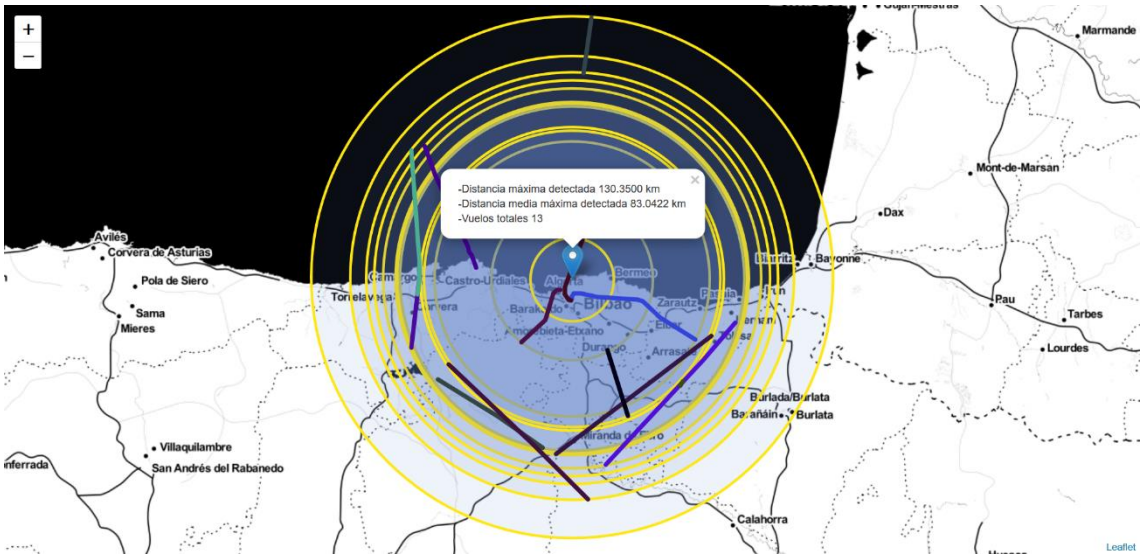
13.4.1. Primera captura



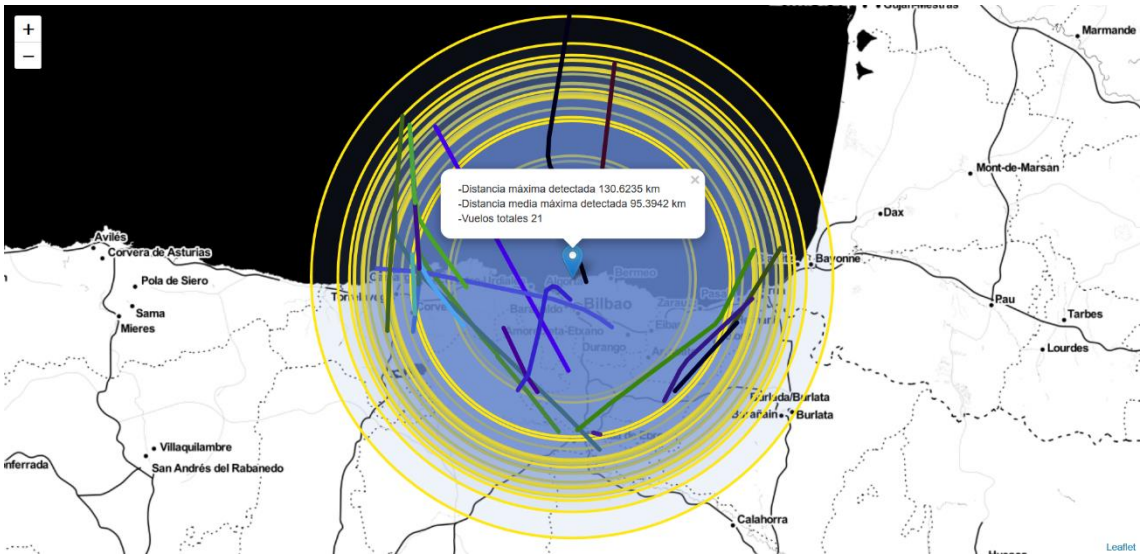
13.4.2. Segunda captura



13.4.3. Tercera captura

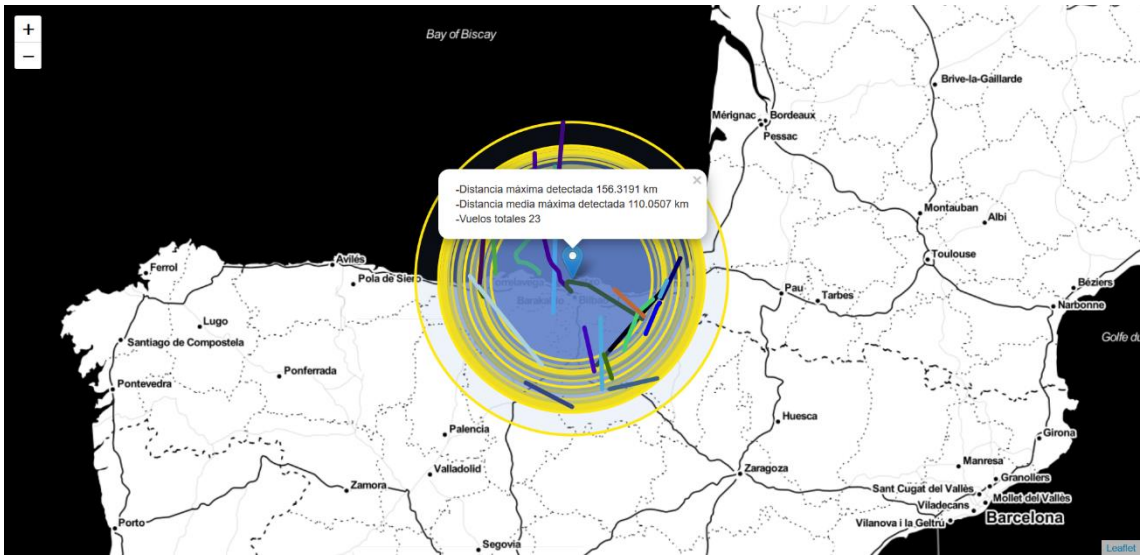


13.4.4. Cuarta captura

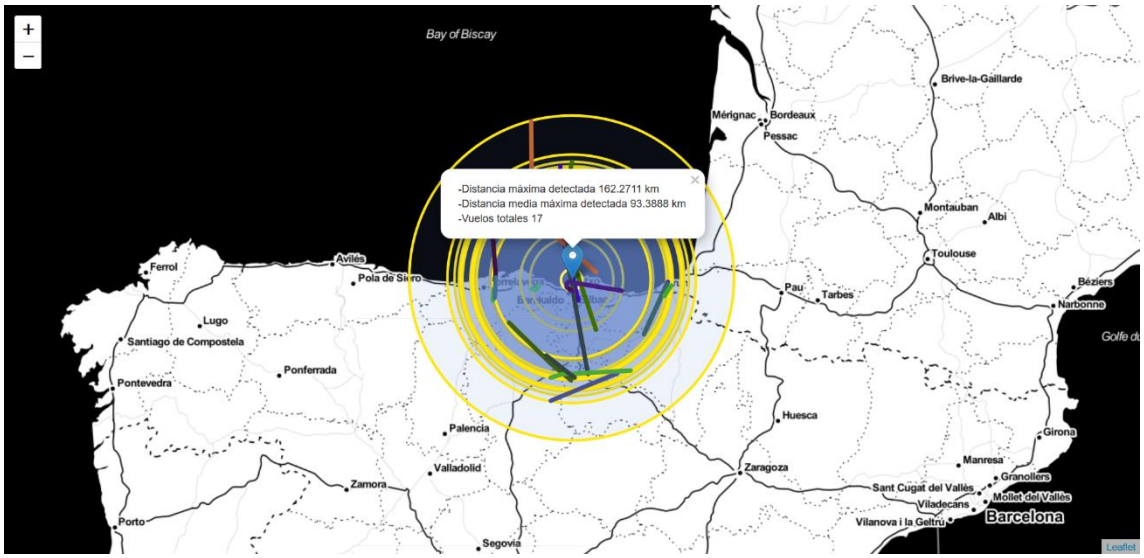


13.5 Mapas antena Spider

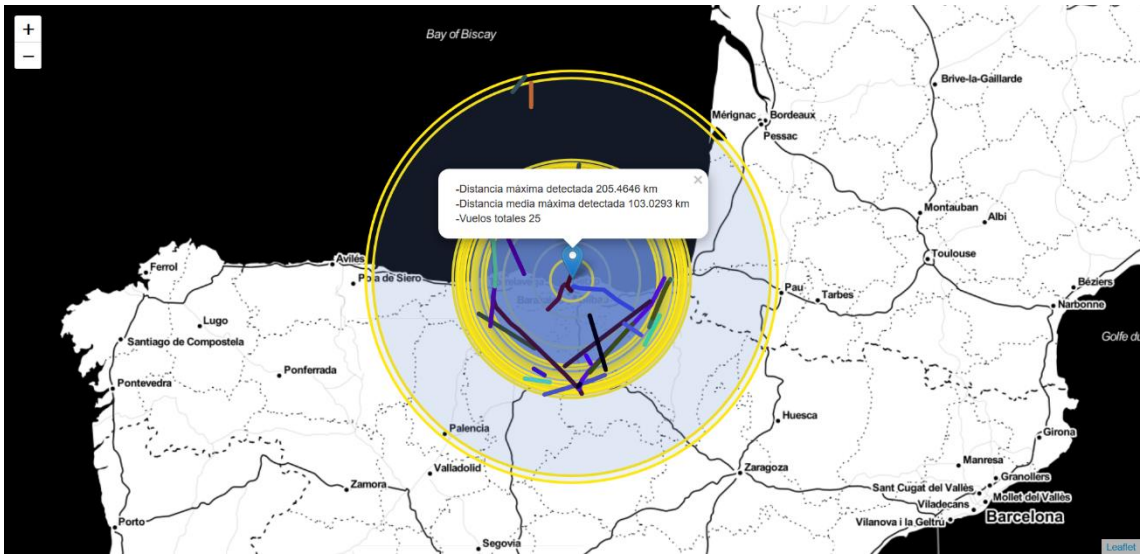
13.5.1. Primera captura



13.5.2. Segunda captura



13.5.3. Tercera captura



13.5.4. Cuarta captura