**Centro educativo**

| **Código** | **Centro** | **Concello** | **Ano académico** |
| --- | --- | --- | --- |
| 15005397 | I.E.S. Fernando Wirtz Suárez | A Coruña | 2022/2023 |

**Ciclo formativo**

| **Código da familia profesional** | **Familia profesional** | **Código do ciclo formativo** | **Ciclo formativo** | **Grao** | **Réxime** |
| --- | --- | --- | --- | --- | --- |
| FP16 | Informática e comunicacións | CSIFC01 | Administración de Sistemas Informáticos en Rede | Superior | Ordinario |

**Módulo profesional e unidades formativas de menor duración (\*)**

| **Código**  **MP/UF** | **Nome** |
| --- | --- |
| MP0373 | Proxecto de Administración de Sistemas Informáticos en Rede  Equivalencia en créditos ECTS: 5.  Código: MP0379.  Duración: 26 horas |

**Profesorado responsable**

| **Titora** | Coello Blanco, Clementina |
| --- | --- |
| **Equipo Docente:** | Corbelle Suárez Clara  Lage Arias, Serxia  Paredes Cereijo, Dolores  Pascual Vázquez, Víctor Alfredo  Villarino Ferreiro, Jaime |

**Alumno**

| **Alumno/a** | Antonio Roca Capdevila |
| --- | --- |

**Datos do Proxecto**

| **Título** | ADS-B , sistema de radar aéreo secundario |
| --- | --- |

**Índice:**

[**Objetivo**](#_heading=h.gjdgxs) **3**

[**Descripción**](#_heading=h.30j0zll) **3**

[**Alcance**](#_heading=h.1fob9te) **3**

[**Planificación**](#_heading=h.3znysh7) **4**

[**Medios a utilizar**](#_heading=h.2et92p0) **4**

[**Presupuesto**](#_heading=h.tyjcwt) **4**

[**Ejecución**](#_heading=h.comz0kndbsjj) **4**

# Objetivo

Módulos tratados durante el desarrollo del proyecto:

**Implantación de sistemas operativos:**

* Búsqueda de los sistemas operativos adecuados al contorno de explotación.
* Instalación, mantenimiento y administración de sistemas operativos clientes.
* Monitorización del rendimiento de sistemas informáticos.

**Planificación y administración de redes:**

* Creación de redes virtuais.

**Fundamentos de hardware**

* Conocimiento y correcta manipulación de los elementos que forman el componente físico y lógico

de los equipamientos.

* Reconocimiento y monitorización de equipamientos, a nivel físico y lógico.
* Puesta en marcha y mantenimiento de periféricos.

**Gestión de bases de datos**

* Planificación y manipulación de datos.

**Administración de sistemas operativos**

* Gestión de la automatización de tareas del sistema.
* Uso de lenguajes de scripting en sistemas operativos libres y propietarios para la administración

de servicios del sistema operativo.

**Implantación de aplicaciones web**

* Instalación y configuración del sistema operativo y los servicios sobre los que se ejecutan las

aplicaciones.

* Creación de documentos web utilizando lenguajes de script de servidor para acceder a la

información almacenada en bases de datos.

**Lenguajes de marcas y sistemas de gestión de la información**

* Uso de linguaxes de marcaxe no tratamento y en la transmisión de información.

# Descripción

Para entender el funcionamiento del proyecto, es necesario saber qué es ADS-B, una vez sepamos en qué consiste esta tecnología, podremos empezar con el proyecto.

**ADS-B** es una abreviación de Automatic Dependent Surveillance – Broadcast (Vigilancia Dependiente Automática – Transmisión).

Automatic : la información se envía periódicamente sin interrogación

Dependent : depende de una fuente de posición en la aeronave (WAAS/GPS)

Surveillance : proporciona servicios de vigilancia parecidos al radar para ATC

Broadcast : se transmite continuamente la información de la aeronave a estaciones terrestres y otras aeronaves.

Cada avión equipado para usar ADS-B calcula exactamente su posición mediante satélites GPS. Es decir, ésta tecnología no depende de los radares como es el caso de los transponder (SSR) tradicionales sino que se basa en GPS, independiente de que también suelen usar los transponders como equipo instalado en las aeronaves.

Después de calcular su posición, el equipo en la aeronave envía un paquete de información, incluyendo su identificación, posición, ruta y velocidad a estaciones terrestres ADS-B y otras aeronaves cada segundo.

Ahora que ya sabemos en qué consiste esta tecnología veamos en qué consistirá el proyecto.

Mediante el uso de una Raspberry Pi 3b, crearemos un receptor de señales ADS-B, que procesará y nos permitirá ver en el mapa las señales recibidas. Una vez montado y configurado el equipo, crearemos un script que nos permitirá enviar avisos, en este caso a través de Telegram, de la detección de aeronaves concretas mediante el acceso a los datos recibidos por medio de nuestro sistema. Posteriormente, crearemos una red privada virtual para poder acceder a nuestro sistema desde cualquier parte, lo que nos permitirá ver, y configurar el sistema desde cualquier parte siempre que estemos conectados a la red.

# Alcance

El alcance del Proyecto estará limitado al área de Galicia debido a los medios técnicos disponibles y a la orografía del terreno. El método de aplicación del mismo sería el mismo en las zonas deseadas donde proporcionar cobertura.

# Planificación

* Creación de una red virtual para control del sistema desde cualquier punto
* Montaje del sistema
* Configuración del sistema
* Elaboración del script de notificación
* Puesta a punto del script mediante testeo del mismo

# Medios a utilizar

* Raspberry Pi 3B \*
* RTL SDR \*
* Cable SMA de baja pérdida\*
* Antena ADS-B 1090

\* podría usarse equipo similar

# Presupuesto

En este caso , la Raspberry debido a que poseo una, no supondría coste de presupuesto, pese a ello, la incluire en el:

* Raspberry Pi o similar(orange pi …) =70€
* RTL SDR =40€
* Cable SMA de baja perdida=Desde 5€
* Antena ADS-B 1090=Desde 5€

# Ejecución

Debido a que usaremos una Raspberry (en adelante me referiré a la Raspberry , pero podría usarse equipamientos similares o dispositivos que monten Linux) el primero de los pasos será comprobar que disponemos de todo el equipamiento necesario para montar, iniciar y configurar el sistema.

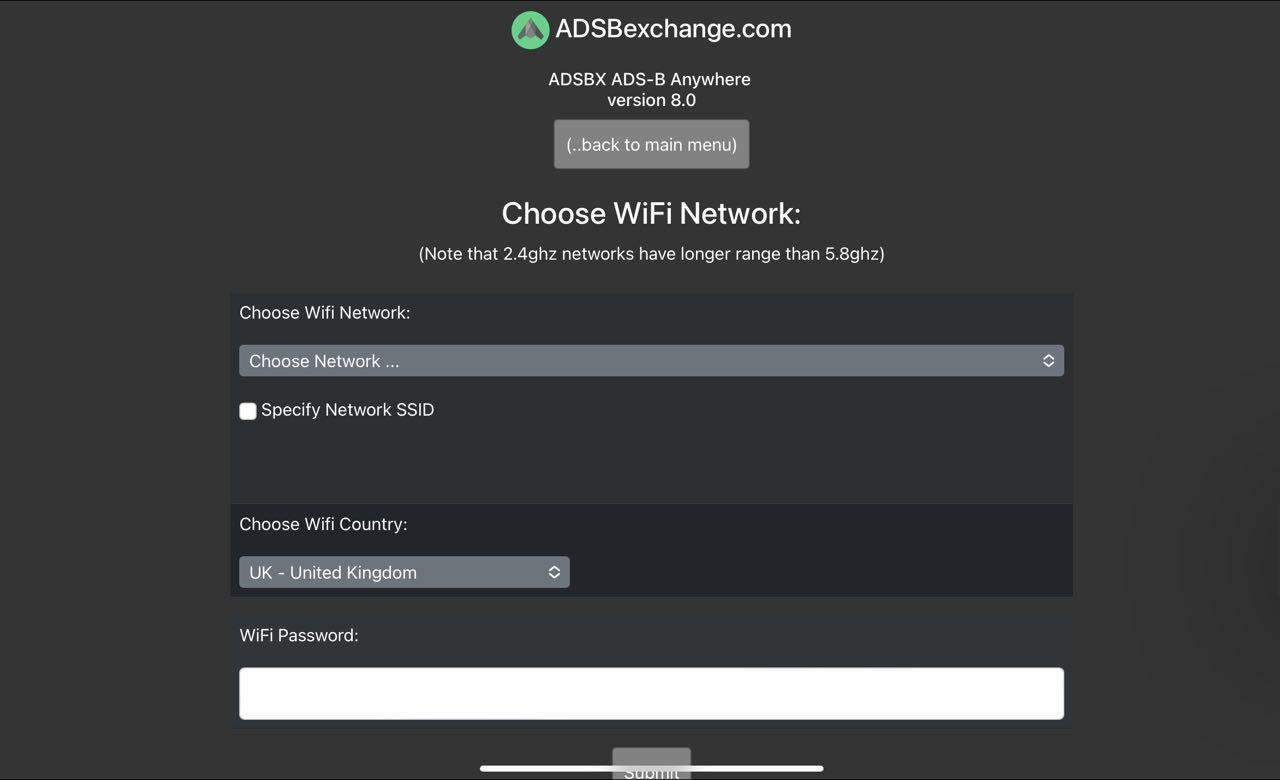
Debemos comprobar que el alimentador de la Raspberry tiene la potencia suficiente para su correcto funcionamiento, en caso de ser un cargador corriente(como los del teléfono móvil) no suministrará la suficiente energía por lo que no llegará a arrancar.

Lo primero que deberemos hacer es grabar una imagen de Raspbian o de ADSBexchange en nuestra tarjeta microSD. Para ello, necesitamos utilizar una herramienta de grabado de imágenes, en nuestro caso, utilizaremos la propia herramienta diseñada por los creadores de la Raspberry, el [Raspberry Pi Imager](https://www.raspberrypi.com/software/) . Una vez descargado, ejecutaremos el programa y en nuestro caso ejecutaremos el proceso de grabado con la imagen de [ADSBexchange](https://www.adsbexchange.com/myip/downloads/adsbx-8.2.220910.zip).

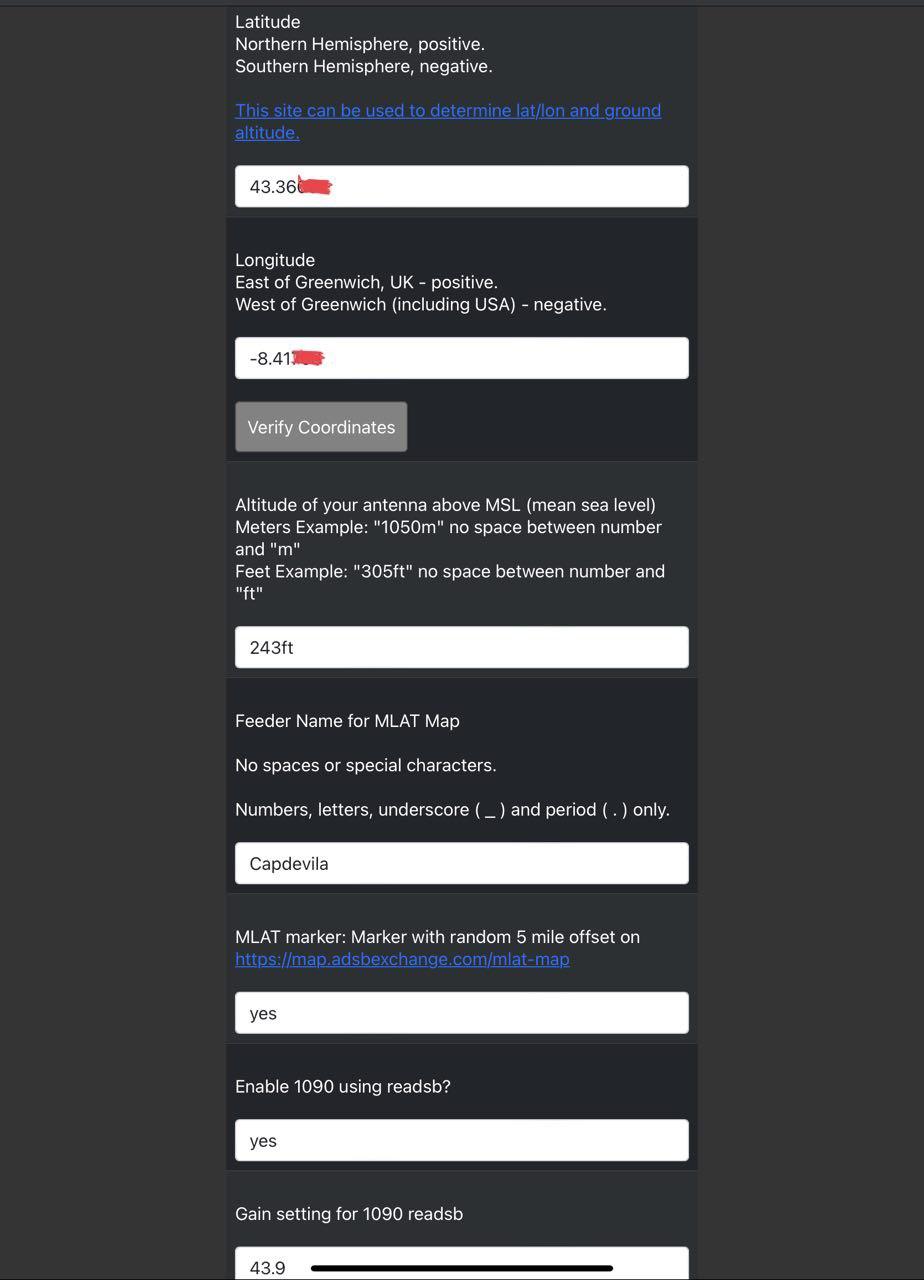
Debemos tener en cuenta a la hora de elegir el software a utilizar, pues en nuestro caso, y debido a las características de hardware de nuestro equipoy comodidad en la interacción con el radar, utilizaremos la imagen de ADSBexchange, pues incluye funcionalidades muy útiles, así como la posible administración del sistema por medio de ssh, es un sistema basado en linux y nos permitirá utilizar más recursos en las funciones de procesamiento de datos del sistema.

Al realizar el primer boot del sistema, podremos detectar una nueva red wifi “ADSBx-config” esta red solo estará disponible los primeros 15min, a la que deberemos conectarnos desde cualquiera de los equipos que dispongamos conectándonos desde nuestro navegador a adsbexchange.local, deberemos seguir las siguientes configuraciones:

En primer lugar configuraremos la red wifi a la que queremos tener conectado el dispositivo, tras introducir los datos y guardar, la red de configuración desaparecerá, y se reiniciará el sistema, ya podremos acceder a adsbexchange.local desde nuestra red. En futuros pasos veremos cómo acceder a esta red desde fuera de la red a la que hemos conectado la Raspberry.

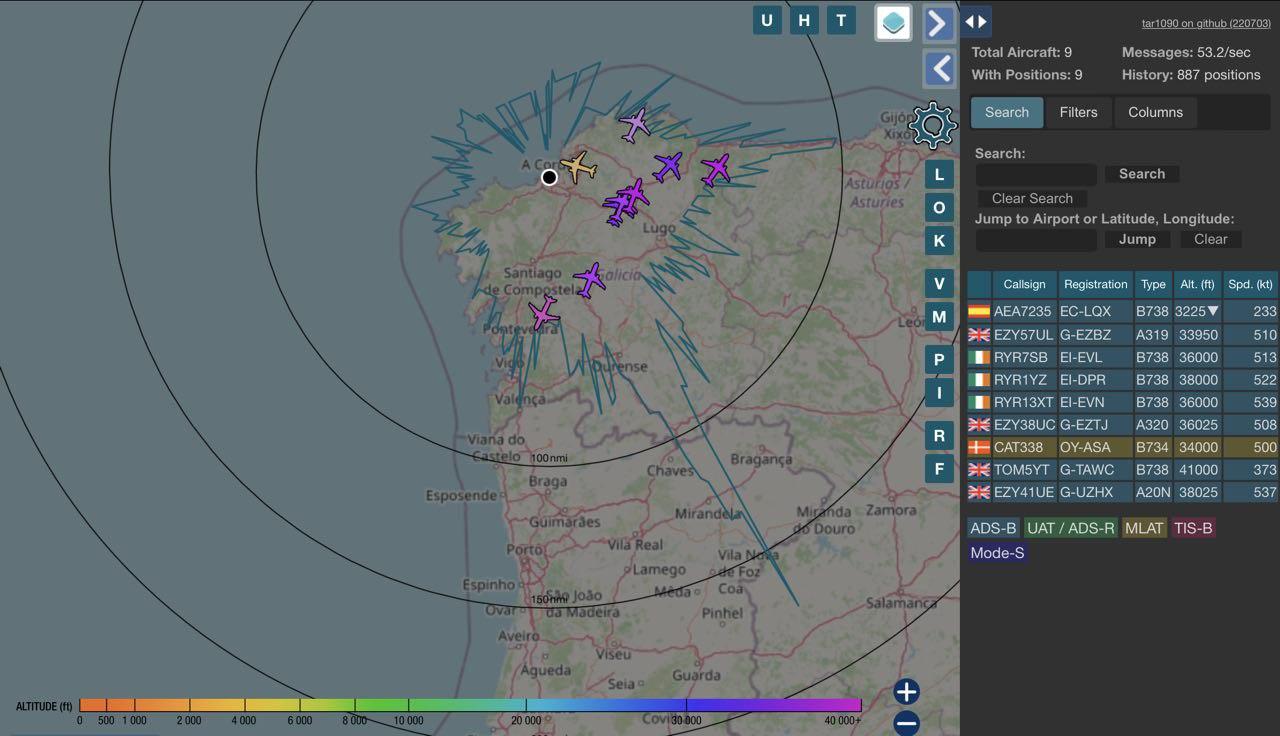


Tras el reinicio, volveremos a acceder a adsbexchange.local y configurar parámetros como la ubicación exacta en coordenadas, la altura, el nombre para el mapa de trazado de MLAT, habilitaremos MLAT, y configuraremos el resto de apartados, aunque muchos de ellos los configuraremos de manera posterior mediante el usos de scripts de optimización.



Guardaremos los datos y se reiniciará de nuevo el sistema. Desde este punto, si volvemos a acceder a adsbexchange.local, y accedemos a 1090 map podremos ver nuestro mapa centrado en nuestra ubicación con las primeras aeronaves detectadas.

Actualmente estamos haciendo feeding a [adsbexchange](https://globe.adsbexchange.com/).



A partir de este momento, realizaremos labores de configuración, mantenimiento y añadir sitios a los que realizar feeding de nuestros datos y veremos algunas de sus ventajas.

Cabe destacar que ADSBexchange es una página libre que no oculta vuelos ni tiene retardo en las señales, como FR24.

**Configuración:**

Lo primero que haremos para configurar algunos de los parámetros y maximizar por software la recepción de señales será conectarnos a la Raspberry via SSH, deberemos encontrar la ip de la Raspberry para ello, podemos usar aplicaciones como FING, donde podremos ver la IP de la misma, para conectarnos a la Raspberry usaremos la consola de comandos o una herramienta como Putty:

**ssh pi@\_\_ip\_raspberry\_\_** siendo (\_\_ip\_raspberry\_\_) la ip de la misma

Nos pedirá una contraseña , la cual será **adsb123**

A continuación, descargamos un script que automatiza la ganancia para que no seamos nosotros los que debamos probar hasta encontrar la mejor configuración de este parámetro, introduciremos :

**sudo bash -c "$(curl -L -o - https://github.com/wiedehopf/adsb-scripts/raw/master/autogain-install.sh)"**

**hash -r**

Tras el comando anterior en que descarga el script, ahora deberemos ejecutarlo con el comando :

**for i in {0..30}; do sudo autogain1090; sleep 120; done**

Con este comando, prueba las distintas ganancias durante dos minutos durante la próxima hora y posteriormente guarda la que mejor resultado ha dado.

**Feeding a flightaware:**

Los siguientes comandos deben ejecutarse todos en el orden siguiente, pese a que algunos de ellos puedan ser redundantes, todos son necesarios para comenzar el feeding.

**URL=""**

**grep -qs -e buster /etc/os-release && URL="https://flightaware.com/adsb/piaware/files/packages/pool/piaware/p/piaware-support/piaware-repository\_6.1\_all.deb"**

**grep -qs -e bullseye /etc/os-release && URL="https://flightaware.com/adsb/piaware/files/packages/pool/piaware/p/piaware-support/piaware-repository\_7.1\_all.deb"**

**wget -O /tmp/piaware-repo.deb "$URL"**

**sudo apt purge -y piaware-repository &>/dev/null**

**sudo rm -f /etc/apt/sources.list.d/piaware-\*.list**

**sudo dpkg -i /tmp/piaware-repo.deb**

**sudo apt update**

**sudo apt install -y piaware**

Tras unos pocos minutos, podremos ir al sitio de [flightaware](https://flightaware.com/adsb/piaware/claim) y solicitar el claim desde nuestra cuenta, lo que nos dará rol business con más funciones y unas estadísticas detalladas de nuestro sistema.

**Feeding a Flightradar24:**

Ahora configuraremos el feeding a FR24, donde nos darán también la suscripción business .

También podremos consultar datos de nuestro feeder, pero la opción más interesante es poder ver el registro de vuelos de las aeronaves.

Comenzaremos descargando el contenido necesario con :

**sudo bash -c "$(wget -O -** [**https://github.com/wiedehopf/adsb-scripts/raw/master/fr24-nopackage.sh**](https://github.com/wiedehopf/adsb-scripts/raw/master/fr24-nopackage.sh)**)"**

El siguiente comando nos abrirá la configuración de comienzo de feeding de FR24, donde introduciremos el siguiente comando :

**sudo fr24feed --signup; sudo systemctl restart fr24feed**

Y las configuraciones que deberemos introducir según nos la soliciten serán las siguientes :

**Would you like to use autoconfig $:**

**no**

**Step 1.3 - Would you like to participate in MLAT calculations? (yes/no)$:**

**no**

**Step 4.1 - Receiver selection:**

**Enter your receiver type (1-7)$:**

**4 (ModeS Beast ... yes really, no it doesn't matter what receiver you're actually running, this means a beast protocol connection to readsb, the decoder at the top of this guide)**

**Step 4.2 - Please select connection type:**

**Enter your connection type (1-2)$:**

**1 (Network ... yes really, it will connect to the decoder at the top of this guide using local loopback)**

**Step 4.3A - Please enter your receiver's IP address/hostname (127.0.0.1 is correct for everyone, means same computer)**

**127.0.0.1**

**Step 4.3B - Please enter your receiver's data port number**

**30005**

**Step 5.1 - Would you like to enable RAW data feed on port 30334 (yes/no)$:**

**no**

**Step 5.2 - Would you like to enable Basestation data feed on port 30003 (yes/no)$:**

**no**

**Step 6 - Please select desired logfile mode:**

**0**

Tras todos estos ajustes, comprobaremos si el funcionamiento e instalación son correctos, ejecutaremos:

**sudo fr24feed-status**

**sudo journalctl -u fr24feed --no-pager**

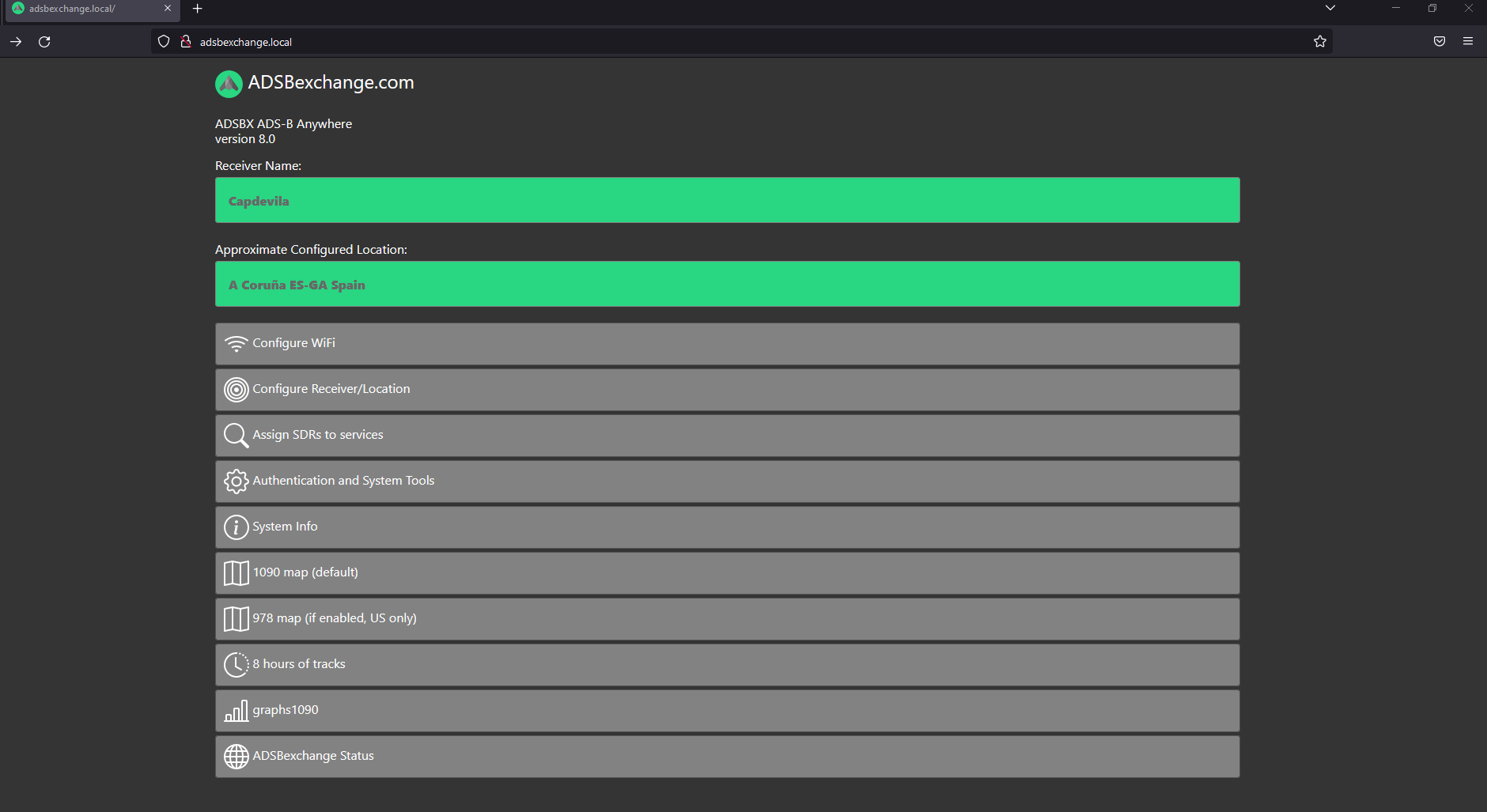
En caso de que la instalación falle o no se configurase correctamente, deberemos desinstalar esta función y volverla a realizar, para desinstalar ejecutaremos:

**sudo systemctl disable --now fr24feed**

**Cómo funciona adsbexchange.local :**

Adsbexchange.local es la interfaz gráfica de nuestro sistema ADS-B en nuestra raspberry . Podremos acceder a ella desde cualquier dispositivo conectado a la misma red que nuestra Raspberry.

Con todo configurado, la ventana que nos aparecerá en ADSBexchange.local será la siguiente:



Como podemos ver en verde podemos ver el nombre del dispositivo, y la ubicación aproximada, cabe destacar que es importante que sea lo más precisa posible, tanto de posición en coordenadas, como de altura, pues de no ser así la altura obtenida de las aeronaves será incorrecta.

En cuanto a las funciones con fondo gris, podemos ver que la primera nos permite seleccionar la red wifi a la que está conectado el receptor.

La segunda de ellas nos permite modificar la ubicación, así como la altura del sistema receptor, debe modificarse cada vez que se cambia el equipo de posición.

La tercera opción, assign SDRs to services, la explicaremos más adelante

La cuarta opción, Authentication and System Tools nos permitirá modificar la contraseña del sistema, en nuestro caso, seguiremos usándola por defecto.

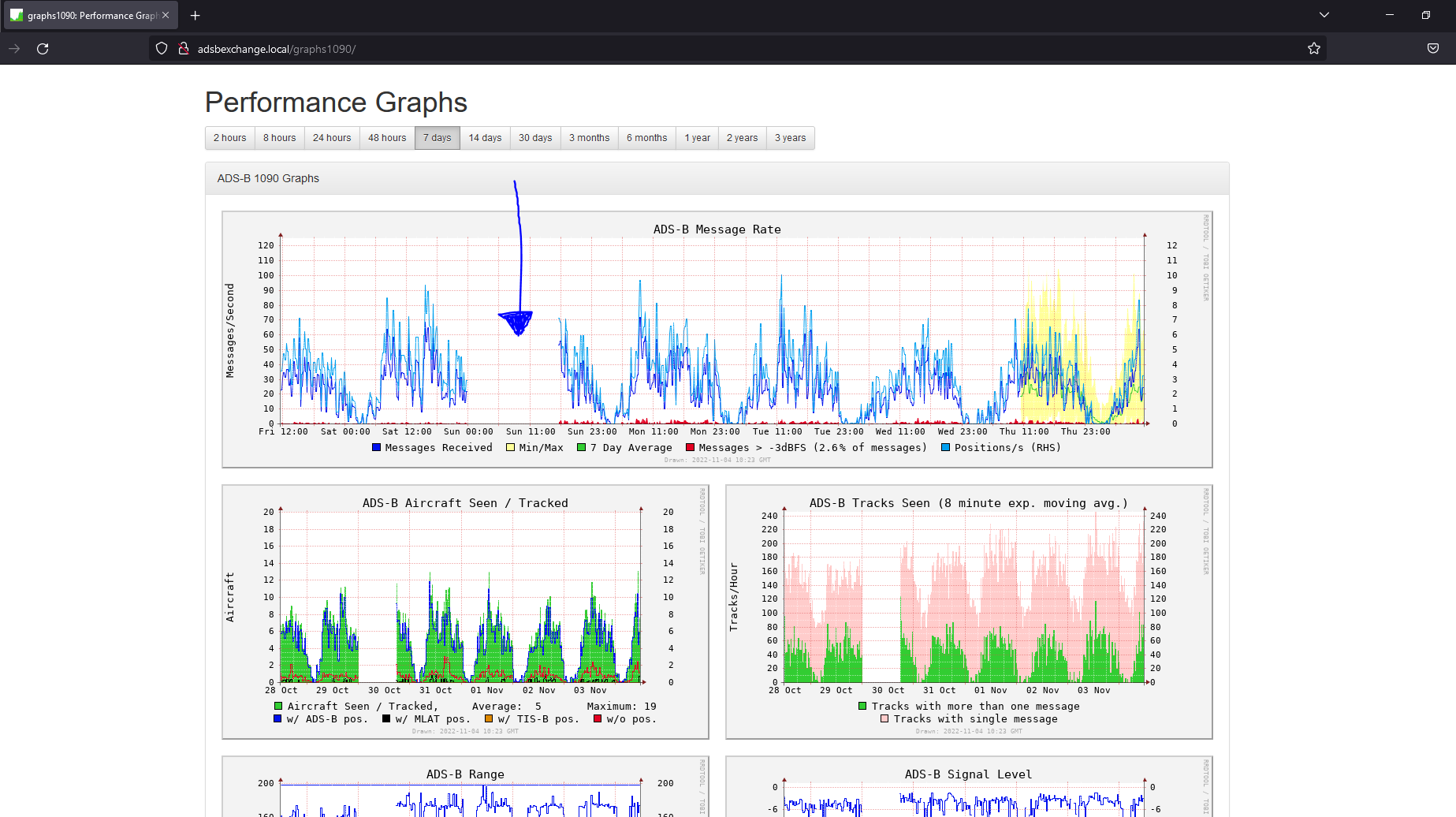
La quinta opción es System info, aquí nos saldrán datos útiles como nuestras ips, la ubicación del feeder, la hora o los paquetes recibidos, simplemente nos permite ver la información del sistema.

La sexta opción es el mapa de Tracking de aeronaves en la frecuencia de 1090Mhz.

La séptima opción es el mapa de 978Mhz, pero es una frecuencia no usada en Europa, por lo que nosotros no le daremos uso, en el caso de EEUU, si se usa y para ello deberemos instalar una nueva antena en otro RTL. En el caso de esta frecuencia en EEUU, las torres de control emiten informes meteorológicos a través de esta frecuencia, por lo que se podría ver la información meteorológica emitida a las aeronaves.

La octava opción es 8 hours of tracks, donde podemos ver el registro en el mapa con las rutas de todas las aeronaves detectadas durante las 8h previas, puede cambiarse este parámetro para que deje un registro de más horas, pero no se recomienda pues podría sobrecargar la memoria del sistema.

La novena opción es tal vez la más interesante tras el mapa donde podemos ver las aeronaves, que son las estadísticas de nuestro receptor, en el podemos ver las aeronaves detectadas y diversos parámetros, que nos ayudaran a ver las ubicaciones más adecuadas. En mi caso, podemos ver un cambio de ubicación en la antena, y la cantidad de paquetes que recibía antes y después, lo que me ayuda a ver qué ubicación era mejor.



Y la última de las opciones es comprobar el status del servicio, donde también podemos obtener el enlace de nuestro mapa para poder verlo desde cualquier ubicación.

**Creación Script notificación aeronaves concretas mediante el uso de los datos obtenidos.**

A continuación crearemos un script en python que permita notificar aeronaves determinadas. Para ello utilizaremos como lenguaje de scripting python, y debido a que el api para la obtención de datos en todo el planeta es de pago, nos limitaremos única y exclusivamente a los datos recopilados por nuestro radar.

Para ello, es preciso que el script sea capaz de monitorizar cambios en el archivo donde se recopilan los datos de las aeronaves. Este fichero es un archivo .json, lo que hace su acceso e interpretación muy sencillos mediante el uso de python.

También necesitaremos de una aplicación que nos pueda notificar en tiempo real, en nuestro caso, haremos uso de los bots de telegram, creando un bot que sea el encargado de notificar mediante un mensaje cuando detecte la aeronave en el listado de aeronaves a notificar.

En último lugar, es un programa que queremos que se esté ejecutando en segundo plano desde nuestra raspberry, por lo que deberemos utilizar una serie de comandos en nuestra consola ssh para el correcto funcionamiento de esta función. Una vez hayamos tenido en cuenta todo esto, podremos empezar la creación del script.

En nuestro caso, nuestro programa notificará de la detección de 2 tipos de aeronaves, en primer lugar notificará de los helicópteros de la DGT, los conocidos como Pegasus, y del helicóptero de salvamento marítimo con base en la ciudad de La Coruña.

Para ello, usaremos dos de los recursos que nos han proporcionado los pasos anteriores. En el caso de nuestro sistema, podremos obtener las matrículas y códigos HEX de las aeronaves que detectemos. Con las matrículas, podremos realizar la búsqueda de las distintas matrículas gracias a nuestro feeding a FlightRadar24, que nos permite al ser feeders buscar las matrículas de las aeronaves y ver los registros.

Mediante este proceso , obtendremos los siguientes resultados:

* EC-LGC #34358d (Pegasus con base en Alvedro a dia 04/11/2022)
* EC-NAA #346105 (Helicóptero de Salvamento marítimo)

Debido a que los helicópteros de la DGT suelen pasar inspecciones en Madrid y que rotan los aparatos, también añadiremos los siguientes aparatos:

* EC-MHV #345358
* EC-LAR #34348c
* EC-LDF #3432c2
* EC-LBD #34329a
* EC-LGD #34358e
* EC-MMF #3452c5
* EC-KXU #343350
* EC-MDO #3445d8
* EC-MHU #345357

Una vez añadidos los aparatos, añadiremos también algunos detalles más de los que queramos recibir notificación, en este caso añadiremos todas las aeronaves con unos callsigns concretos, lo que hará que siempre notifique si la aeronave tiene parte del callsign correspondiente al que nosotros indiquemos.

* Abeja (callsign helicopteros DGT)
* AIB (Airbus)
* HBAL(Globos de google)

También añadiremos en función al squawk, un código que las aeronaves emiten con el fin de transmitir la situación actual de la aeronave.

* 7500 (Secuestro)
* 7600 (Fallo de radio)
* 7700 (EMERGENCIA (avión cayendo , en descenso))

Una vez hecho esto, el primer paso será crear un nuevo bot en nuestro caso en Telegram.

Para ello, nos iremos a Telegram, e iniciaremos una conversación con el bot de telegram encargado de la creación de los mismos **@BotFather.**

Una vez en la conversación, pulsaremos **Start**, y usaremos el comando **/newbot**, donde introduciremos el nombre y el nombre de usuario. El bot nos proporciona un token que será el que introduciremos en la sección del código en la que le decimos que bot queremos que use como medio de notificación. En nuestro caso, necesitaremos también el token de la conversación donde queremos que notifique el bot, por lo que obtendremos también el token de en nuestro caso un canal, donde queremos que el bot emita los avisos.

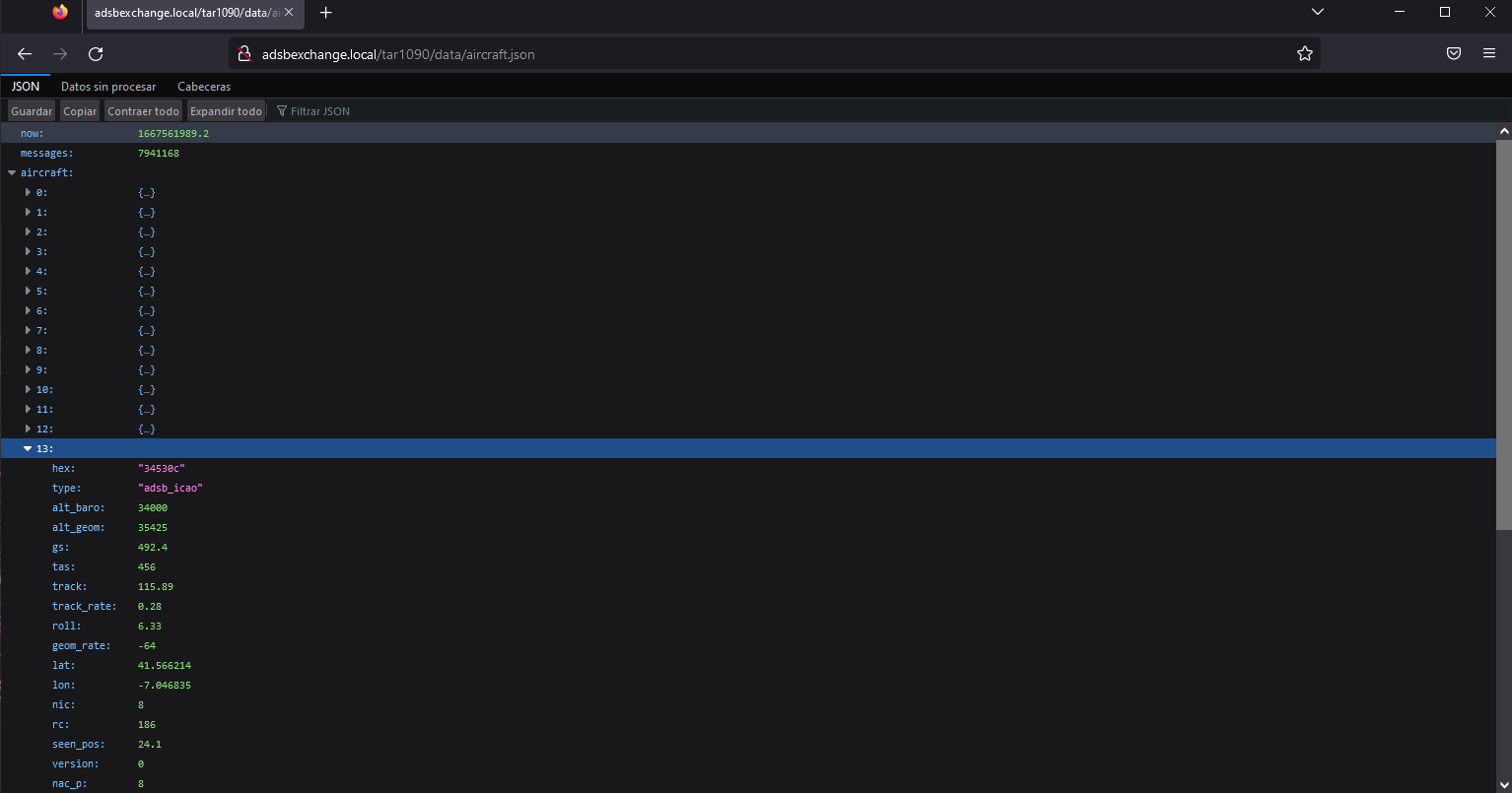
Para obtener el id del canal donde queremos que notifique, nos iremos a la versión web de Telegram, nos situaremos en el canal deseado, en la url nos aparecerá el siguiente formato:

[https://web.telegram.org/z/#-**XXXXXXXXXX**](https://web.telegram.org/z/#-XXXXXXXXXX)siendo el **XXXXXXXXXX** el id de nuestro canal, a mayores antes del id del canal, introduciremos un **-100**, el **-** deberá asignarse siempre , mientras que el 100 se coloca si el canal se ha creado de manera privada, aunque de manera posterior cambie a publica.

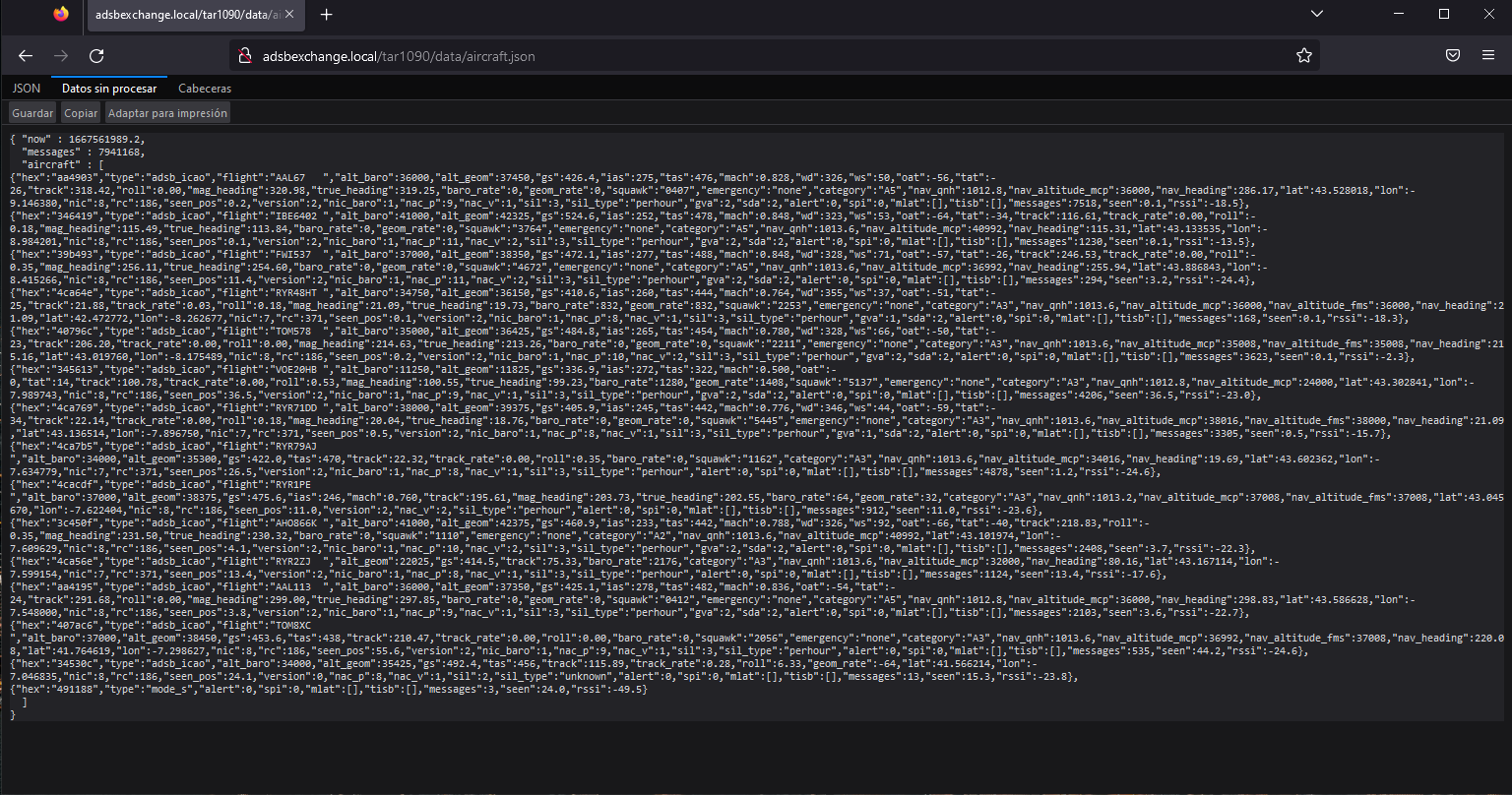
Una vez tenemos los tokens y las aeronaves que queremos poder notificar, deberemos saber dónde se encuentran los datos recibidos por nuestro radar.

Podremos verlos mediante dos formas, la primera sería mediante el navegador entrando en

<http://adsbexchange.local/tar1090/data/aircraft.json>. En esta dirección podremos ver los datos mostrados por el fichero .json

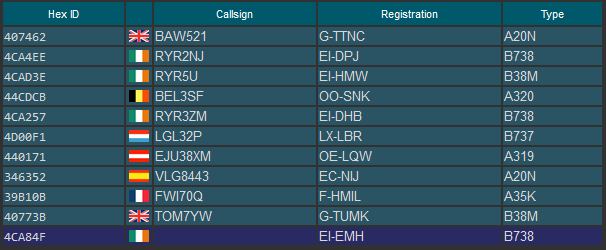


Si clicamos en la opción de datos sin procesar, nos aparecerá el siguiente formato



Como podemos observar, son datos no estructurados, lo que nuestro programa debe hacer, será buscar en este conjunto de datos subcadenas determinadas, en este caso, buscará elementos fijos de cada aeronave, esos elementos son siempre el código HEX y la matrícula de registro de la aeronave.

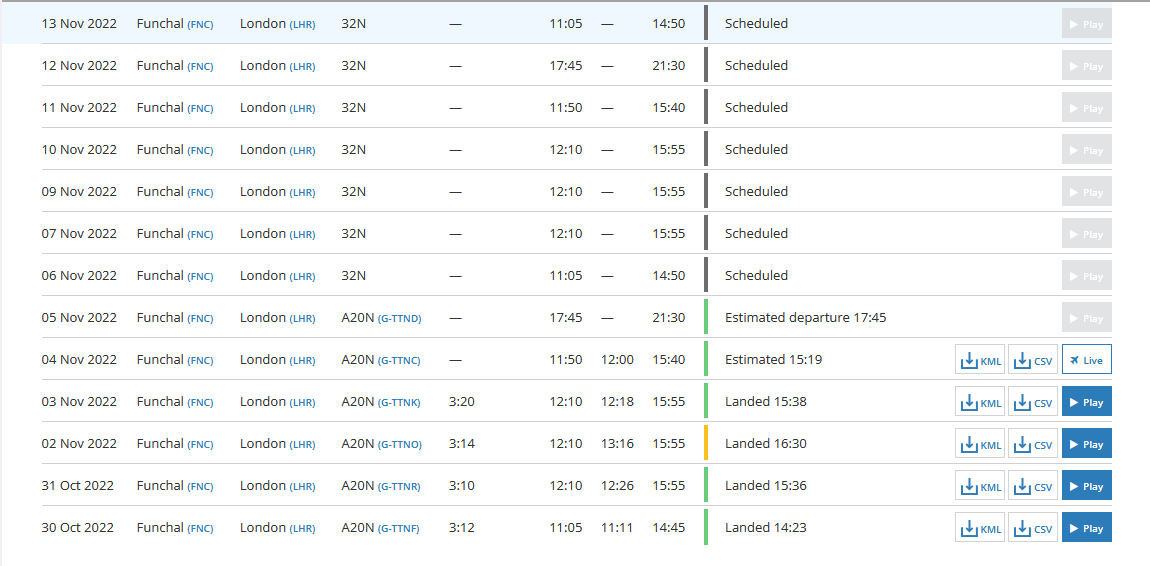
Para que nos hagamos una idea:



Como observamos en la imagen, solo hay dos elementos que son únicos por cada aeronave, el **Hex ID y el código de Registro**, ninguna aeronave puede tener repetidos esos valores.

Sin embargo, el Callsign y el tipo de aeronave si pueden estar repetidos, pues no solamente hay una aeronave de un modelo y el callsign varía en función del vuelo.

Podemos ver los cambios de CALLSIGN en vuelos concretos, por ejemplo, si buscamos el primero de las aeronaves de la lista anterior en Flightradar mediante su callsign (**BAW521**), obtendremos lo siguiente:



Si nos fijamos, podemos ver que siempre realiza en este caso el mismo recorrido, Funchal-Londres, pero nunca es exactamente la misma aeronave pese a ser el mismo tipo, entre el 30 de Octubre y el 4 de Noviembre, la ruta la han cubierto las aeronaves (**G-TTNC, G-TTNK, G-TTNO, G-TTNR , G-TTNF**) y que mañana, dia 5 de Noviembre, la cubrirá el **G-TTND**. Esto ocurre de una manera más visual con los vuelos del 45 Grupo del Ejército del aire, encargado del transporte de autoridades, pese a tener 7 aeronaves, usan una gran cantidad de callsigns pese a tener cada uno su matrícula única. En vuelos nacionales usan la denominación **LINCEXX**, mientras que en vuelos que pasen por espacio aéreo extranjero utilizan la denominación **AME45XX**. Mismo avión pero distinto callsign , por lo que no nos es válido para la búsqueda de la aeronave.

Si nos ponemos a revisar el archivo json, encontramos que de los 2 parámetros que buscamos, encontramos el HEX ID de cada aeronave, ese será el atributo que buscaremos en el archivo para que en el momento de ser detectado, envíe una notificación.

El otro método de visualizar el archivo .json es buscando el archivo de escritura y lectura en el propio equipo. Podremos encontrar el archivo en la ruta : **/run/readsb/aircraft.json**

En nuestro caso, usaremos este segundo metodo de visualizacion, pues solo tendremos que abrir el archivo y que realiza la búsqueda cada pocos segundos, mientras que si seguimos el primer método, tendríamos que hacer un problema que haga webscrapping de esa dirección, lo que crearía un código más largo y engorroso a la hora de realizar el mantenimiento del mismo y posibles mejoras futuras.

El resultado final del código, podemos verlo en la carpeta de entrega del proyecto

**Ejecución del Script en segundo plano.**

El primero de los pasos será conectarnos vía ssh a nuestra Raspberry.

Para ello en windows con putty solo necesitamos conocer la ip de la misma, y posteriormente introducir el usuario y contraseña.

En el caso de las distribuciones linux, el comando será:

**ssh pi@ipRaspberry**

Donde posteriormente introduciremos la contraseña del usuario.

Una vez logueados es recomendable elegir una ruta donde tengamos el archivo a ejecutar.

Debemos tener python instalado

Una vez hayamos elegido la ruta donde queremos ejecutar y guardar el archivo, introduciremos el siguiente comando para ejecutar el programa en 2 plano:

**nohup python3 flights.py </dev/null &>/dev/null &**

Una vez ejecutado, podremos comprobar si se está ejecutando, mediante el siguiente comando:

**ps ax | grep flights.py**

Donde veremos todos los .py en ejecución, cada uno con un número asociado, en caso de querer cortar la ejecución del programa, ejecutaremos:

**kill id**

**Creación de una red virtual para el acceso desde cualquier punto**

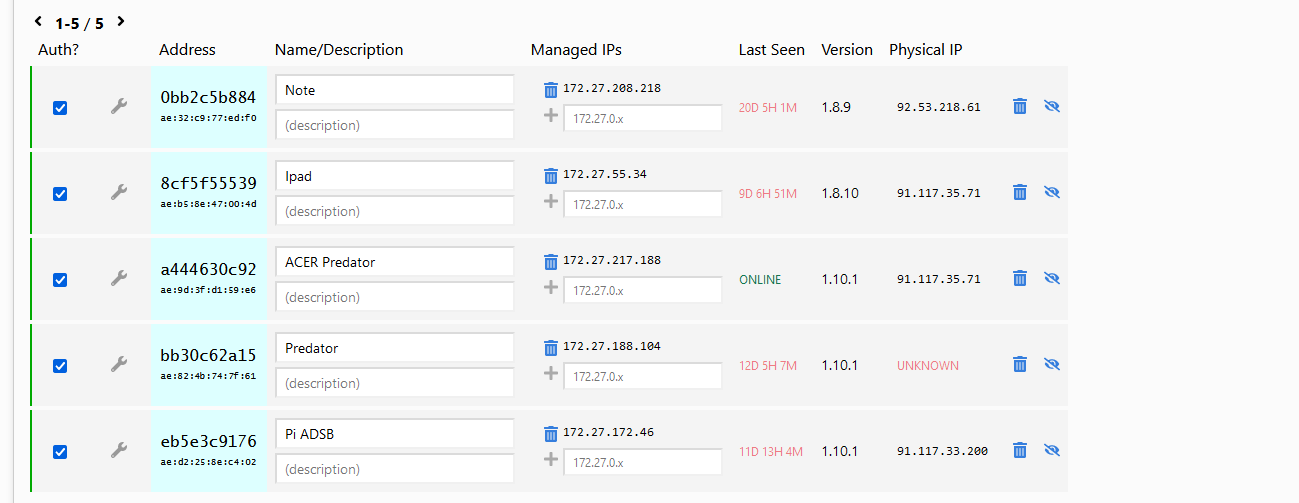
Debido a que solo podemos acceder a todas estas propiedades de nuestra Raspberry desde la red en que está conectada, crearemos una red virtual a la que conectar nuestros equipos y la raspberry para poder acceder desde cualquier parte, sin necesidad de que sea desde nuestra red local. Para ello utilizaremos ZeroTier, con sus herramientas gratuitas de creación de una única red por correo.

Para ello, deberemos descargar ZeroTier en los equipos donde queramos ejecutarla.

En nuestro ordenador, ingresamos en ZeroTier donde nos aparecerá una red ya creada e iremos a miembros, cuando queramos añadir un equipo, iniciaremos ZeroTier en el equipo, y añadiremos la red mediante su código.



Si hacemos scroll hacia abajo dentro de la red , llegaremos a la parte de autorizados , donde podremos ver los siguientes datos



Cada vez que nos conectemos a la red virtual, desde esta pantalla deberemos marcar la casilla de verificación de AUTH, para así darle acceso al equipo a la red.

Para instalar Zerotier en la Raspberry deberemos ejecutar:

**curl https://raw.githubusercontent.com/zerotier/ZeroTierOne/master/doc/contact%40zerotier.com.gpg | gpg --dearmor | sudo tee /usr/share/keyrings/zerotierone-archive-keyring.gpg >/dev/null**

Haremos un update.

**sudo apt update**

Instalaremos zerotier.

**sudo apt install -y zerotier-one**

Ahora añadiremos la raspberry a la red que hemos creado ejecutando:

**sudo zerotier-cli join [NETWORKID]**

y sustituyendo network id por el id de nuestra red que mostramos anteriormente.

Debemos verificar la raspberry desde el panel anterior marcando la opción de auth.

Para que la raspberry se ejecute desde la red virtual, ejecutaremos:

**sudo zerotier-one -d**

Tras este comando, para acceder a la url adsbexchange.local, deberemos escribir desde los equipos conectados a la red virtual la ip de la raspberry en la red local:

Siendo esta en nuestro caso :

172.27.172.46