El presente proyecto pretende desarrollar un servicio WEB mediante el cual tener acceso a las imágenes meteorológicas ofrecidas por la serie de satélites NOAA activos. Para ello, ha sido necesario implementar un sistema de recepción de la señal APT (Automatic Picture Transmission), propia de los satélites NOAA, basado en Software Defined Radio (SDR). El sistema de recepción está ubicado en la estación terrena de GranaSAT.
El trabajo fin de Máster presentado, consistirá en la implementación hardware y software necesaria para la descarga y posterior visualización de las imágenes meteorológicas transmitidas por dichos satélites.

Jennifer López Morillas nació en Granada, España, en septiembre de 1992. Con este trabajo finaliza el Máster en Ingeniería de Telecomunicaciones tras haber finalizado el Grado de Ingeniería de las Telecomunicaciones con matrícula de honor en su Trabajo Fin de Grado en la Universidad de Jaén.

Andrés María Roldán Aranda es el profesor ingeniero a cargo del presente proyecto, así como el tutor del alumno. Actualmente es profesor del departamento de Electrónica y Tecnología de Computadores de la Universidad de Granada.

TRABAJO FIN DE MÁSTER

Servicio WEB para satélites meteorológicos NOAA basado en receptor SDR

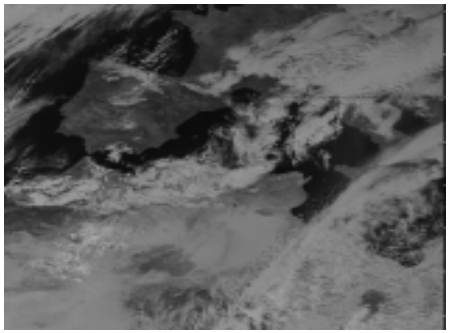
Jennifer López Morillas

INGENIERÍA DE TELECOMUNICACIONES

2016/17

UNIVERSIDAD DE GRANADA

Máster en Ingeniería de Telecomunicaciones



Trabajo Fin de Máster

Servicio WEB para satélites meteorológicos NOAA basado en receptor SDR

Jennifer López Morillas

Año académico 2016/2017

Tutor: Andrés María Roldán Aranda

UNIVERSIDAD DE GRANADA

MÁSTER EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

"Servicio WEB para satélites meteorológicos NOAA basado en receptor SDR"

CURSO: 2016/2017

Jennifer López Morillas

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, 08 de Septiembre de 2017

UNIVERSIDAD DE GRANADA

MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

"Servicio WEB para satélites meteorológicos NOAA basado en receptor SDR"

REALIZADO POR:

Jennifer López Morillas

DIRIGIDO POR:

Andrés María Roldán Aranda

DEPARTAMENTO:

Electrónica y Tecnología de los Computadores

D. Andrés María Roldán Aranda, Profesor del departamento de Electrónica y Tecnología de los Computadores de la Universidad de Granada, como director del Trabajo Fin de Grado de D.ª Jennifer López Morillas,

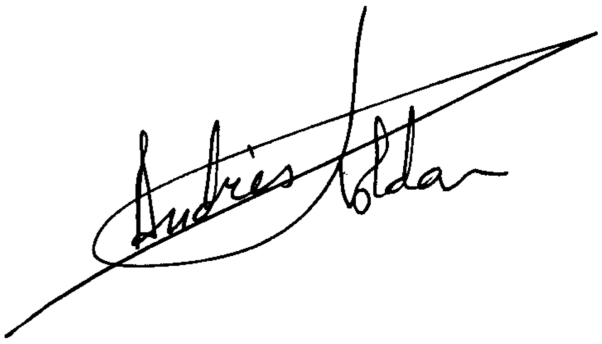
Informa:

que el presente trabajo, titulado:

"Servicio WEB para satélites meteorológicos NOAA basado en receptor SDR"

ha sido realizado y redactado por el mencionado alumno bajo nuestra dirección, y con esta fecha autorizo a su presentación.

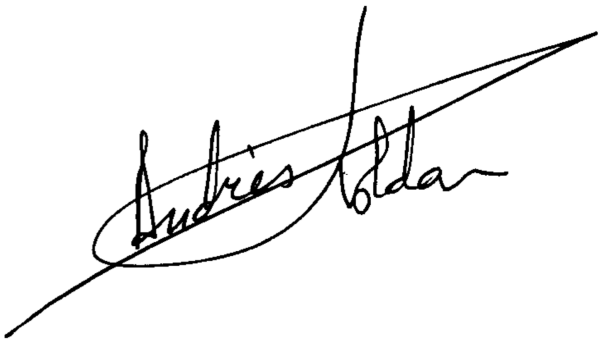
Granada, a 08 de Septiembre de 2017



Fdo. Andrés María Roldán Aranda

Los abajo firmantes autorizan a que la presente copia de Trabajo Fin de Máster se ubique en la Biblioteca del Centro y/o departamento para ser libremente consultada por las personas que lo deseen.

Granada, a 08 de Septiembre de 2017



Fdo. Jennifer López Morillas

Fdo. Andrés María Roldán Aranda

Servicio WEB para satélites meteorológicos NOAA basado en receptor SDR

Jennifer López Morillas

PALABRAS CLAVE:

GranaSAT, satélite, estación terrena, National Oceanic and Atmospheric Administration (NOAA), Automatic Picture Transmission (APT), imagen meteorológica, Software Defined Radio (SDR), CST, downlink, Python, Javascript, HyperText Markup Language (HTML), Razon de Onda Estacionaria (ROE), S_{11}, electromagnético, Half Power Beamwidth (HPBW), ancho de banda, directividad, omnidireccional, Raspberry Pi, RTL_FM, framework, Flask.

RESUMEN:

El propósito principal del presente proyecto es desarrollar un servicio WEB mediante el cual tener acceso a las imágenes meteorológicas ofrecidas por la serie de satélites NOAA de órbita polar que actualmente se encuentran activos. Para ello, ha sido necesario implementar un sistema de recepción y posterior decodificación de la señal APT (Automatic Picture Transmission). Este tipo de señal es la usada por los satélites NOAA para el envío de los datos captados por su sensor AVHRR/3. El sistema de recepción se alojará en la estación terrena facilitada por GranaSAT. Este sistema de recepción estará basado en Software Defined Radio (SDR).

Ya que el objetivo de este proyecto es la creación de un servicio de imágenes meteorológicas, las imágenes recibidas gracias a nuestro sistema de recepción implementado serán almacenadas en el servidor WEB y estarán disponibles para su descarga por un usuario final. El desarrollo de este servicio WEB es otra de las partes centrales del trabajo llevado a cabo. Por lo tanto, el Trabajo Fin de Máster presentado, consistirá en la implementación hardware y software necesaria para la descarga y posterior visualización de las imágenes NOAA.

Este Trabajo Fin de Máster de la titulación de Máster en Ingeniería de Telecomunicación, tiene como propósito el desarrollo de tareas multidisciplinares en el campo de las Telecomunicaciones, donde se utilicen, dentro de lo posible, todos los conocimientos adquiridos durante la formación.

Dedicado a

Mi familia, a la que debo todo su apoyo.

Agradecimientos:

En primer lugar, me gustaría agradecer a mis padres, Santiago y María, y a mis hermanos por su apoyo, animo y buenos consejos en mis años universitarios. Además de a ellos, debo mencionar a mis grandes amigas, Eloísa, Aida y Ana, que han sido el empuje en momentos duros y la voz para creer en mis posibilidades.

De igual forma, agradezco la compañía y buenos momentos pasados con mis compañeros y amigos del Máster, juntos el esfuerzo llevado a cabo se ha hecho más llevadero.

Por último, agradezco a todo el equipo de desarrollo de GranaSAT su colaboración, pero en especial a mi tutor, Andrés María Roldán Aranda, por su tiempo dedicado y su interés por hacernos progresar.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

GLOSARIO

ACARS Sistema de comunicación codificada entre aeronave y estación terrena vía VHF y HF.

ADS-B El ADS-B es un sistema de vigilancia en tiempo real que se utiliza para el control del tráfico aéreo. La información puede ser recibida por las estaciones terrestres de control de tráfico aéreo como un reemplazo para el radar secundario. También puede ser recibida por otras aeronaves para proporcionar conocimiento de la situación y permitir la auto-separación. El avión determina su localización a través de navegación vía satélite..

AIS Sistema que permite a embarcaciones comunicar su posición y otras informaciones relevantes para que otros buques o estaciones puedan conocerla y evitar colisiones. Se trata de un sistema de comunicación radio vía VHF.

APRS Sistema de radio amateur basado en la comunicación de información de la posición de estaciones fijas o móviles utilizando coordenadas Global Positioning System (GPS) u otros datos de telemetría.

ASP.NET Active Server Pages. Lenguaje de programación para Windows. `https://www.asp.net/`.

AVHRR Instrumento de captura de imágenes usado por los satélites NOAA. Está compuesto por 5 módulos ensamblado bajo un único elemento [5]: módulo de escáner, oódulo electrónico, Módulo de radiación de baja temperatura, subsistema óptico, el cual está conformado por un telescopio y la unidad óptica de relé, que permite trabajar en las diferentes bandas espectrales y unidad de placa base.

AVHRR/3 Se utiliza desde su incorporación en la serie de satélites NOAA KLM (NOAA-15, NOAA-16 y NOAA-17). Destaca por estar formado por seis canales, que miden la radiación en cinco bandas simultáneas del espectro electromagnético, de los cuales, tres están dedicados a la captura de los datos APT. Un canal en la región visible de AVHRR/3 es usado para obtener los datos APT durante las horas de luz y un canal IR es usado tanto durante el día y como durante la noche. Un segundo canal IR es programado para reemplazar al canal en el rango visible durante la órbita en zona de oscuridad..

C Lenguaje de programación para desarrollo de software de sistemas y aplicaciones..

CST CST STUDIO SUITE, https://www.cst.com/.

Cubesat Es un tipo de satélitede reducido tamaño para la captura de información espacial. Se construye con módulos con un volumen de un litro y un peso no superior a 1.33 kg.

Drupal Gestor de contenidos de páginas web. https://www.drupal.org/.

Flask Flask es un microframework para Python basado en Werkzeug y Jinja 2. Disponible en: http://flask.pocoo.org/.

GPL La Licencia Pública General de *GNU* o más conocida por su nombre en inglés *GNU General Public License* o simplemente sus siglas del inglés *GNU GPL*, es una licencia creada por la *Free Software Foundation* en 1989 (la primera versión), y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios..

Gpredict Software de predicción de órbitas satelitales. http://gpredict.oz9aec.net/.

GranaSAT GranaSAT is an academic project from the University of Granada consisting of the design and development of a picosatellite (Cubesat). Coordinated by the Professor Andrés María Roldán Aranda, GranaSAT is a multidisciplinary project with students from different degrees, where they can acquire and enlarge the necessary knowledge to front a real aerospace project. http://granasat.ugr.es/.

Java https://www.java.com/es/.

Javascript https://www.javascript.com/.

Joomla Gestor de contenidos de páginas web. https://www.joomla.org/.

JQUERY Biblioteca multiplataforma de Javascript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y GNU is Not Unix (GNU).

JSON es un formato de texto ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript aunque, debido a su amplia adopción como alternativa a XML, se considera un formato de lenguaje independiente.

MATLAB MATrix LABoratory`http://www.mathworks.com/products/matlab/`.

MIRP Es sistema on-board de los satélites NOAA encargado de procesar los datos en diferentes formatos de transmisión. Todo el procesado de MIRP está hecho a partir de datos digitales facilitados por el sensor Advanced Very High Resolution Radiometer 3º Generation, (AVHRR/3). Los datos de sus seis canales son muestreados simultáneamente con una tasa de muestreo de 40 kHz y convertidos en palabras de 10 bits.

Modelo OSI Se tarta del modelo de interconexión de sistemas abiertos (ISO/IEC 7498-1), creado en el año 1980 por la Organización Internacional de Normalización (ISO, International Organization for Standardization). Consta de siete capas, que se nombran a continuación empezando por la más alta en la jerarquía (la capa de aplicación) y siguiendo hacia la más baja (la capa física): Aplicación, Presentación, Sesión, Transporte, Red, Enlace de datos y Física.

NOAA-15 NOAA-k, de la serie de satélites (POES) Polar Operational Environmental Satellites, fue renombrado como NOAA-15 después de su lanzamiento el 13 de mayo de 1998. Con él aparece una generación de satélites con mejoras tecnológicas en su instrumentación. Comenzó a operar el 15 de diciembre de 1998. Su frecuencia de downlink para APT es 137.62 MHz y para High Resolution Picture Transmission (HRPT) de 1702.5 MHz. Se encuentra a una altitud de 807 km y el periodo de su órbita es de 101.1 minutos.

NOAA-18 NOAA-N, pertenece a la última generación de satélites (POES) Polar Operational Environmental Satellites, fue renombrado como NOAA-18 después de su lanzamiento el 20 de mayo de 2005 y empezó a transmitir el 30 de agosto de 2005. Su frecuencia de downlink para APT es 137.9125 MHz y para HRPT de 1707.0 MHz. Se encuentra a una altitud de 854 km y el periodo de su órbita es de 102.12 minutos.

NOAA-19 NOAA-N *Prime*, es el último satélite de la generación de satélites (POES) Polar Operational Environmental Satellites, fue renombrado como NOAA-19 después de su lanzamiento 6 de febrero de 2009 y comenzó a operar el 2 de junio de 2009. Su frecuencia de downlink para APT es 137.1 MHz y para HRPT de 1698.0 MHz. Se encuentra a una altitud de 870 km y el periodo de su órbita es de 102.14 minutos.

Node.js Entorno de ejecución para Javascript. `https://nodejs.org/es/`.

Orbitron Software de predicción de órbitas de satélites. `http://www.stoff.pl/`.

Perl Lenguaje de programación de código abierto. `https://www.perl.org/`.

PHP PHP Hypertext Preprocessor. Lenguaje deprogramación de código abierto. http://php.net/.

POES Satélites meteorológicos de órbita polar: La agencia NOAA y la NASA cooperaron para la construcción y lanzamiento de satélites de órbita polar de uso civil. La NASA se encargó de la construcción, evaluación y puesta en órbita de los satélites conocidos como POES, para posteriormente pasar a ser controlados por la agencia NOAA [2][4]. Actualmente NASA y NOAA trabajan en una nueva generación denominada 'Joint Polar Satellite System' (JPSS), que operará después de que los POES completen su misión.

Python https://www.python.org/.

QTH Código en lenguaje de radioaficionado que define la localización de la *ground station* desde dónde se transmite o recibe.

Raspberry Pi https://www.raspberrypi.org/.

Raspbian Jessie Lite Descarga en: https://www.raspberrypi.org/downloads/raspbian/.

Ruby Lenguaje de programación de código abierto. https://www.ruby-lang.org/es/.

SDRSharp Software de control de SDR, disponible para windows y Linux. Ofrece la instalación de gran variedad de plugins que permiten un mejor control y procesado de la señal recibida.

SGP4/SDP4 Combinación de los modelo matemáticos de propagación orbital *Simplified General Perturbations* y *Simplified Deep Space Perturbations* utilizados para determinar la posición de satélites (POES) Polar Operational Environmental Satellites o cuerpos celestes.

SGP8/SDP8 Mejora de SGP4/SDP4.

StackOverflow https://stackoverflow.com/.

TIOBE TIOBE. The software quality company. https://www.tiobe.com/.

WordPress Gestor de contenidos de páginas web. https://wordpress.org/.

WXtoImg Software para decodificación de señales de satélite. Realiza corrección doppler para la obtención de la imagen enviada por el satélite. Se trata de un software Open Source. http://wxtoimg.com/.

ACRÓNIMOS

ADC Analog-to-Digital Conversion.

ADS Advanced Design System.

AM Amplitude Modulation.

AMSAT Radio Amateur Satellite Corporation.

AoS Acquisition of Signal.

APT Automatic Picture Transmission.

ATC Air Traffic Control.

CMS Content Management System.

CNAF Cuadro Nacional de Atribución de Frecuencia.

CPU Central Processing Unit.

CSS Cascading Style Sheets.

DAB Digital Audio Bradcasting.

DVB-T Digital Video Broadcasting- Terrestrial.

FFT Fast Fourier Transform.

FM Frequency Modulation.

GNU GNU is Not Unix.

GOES Geostationary Operational Environmental Satellites.

GPS Global Positioning System.

GSM Global System for Mobile communications.

HF High Frequency.

HPBW Half Power Beamwidth.

HRPT High Resolution Picture Transmission.

HTML HyperText Markup Language.

HTTP HyperText Transfer Protocol.

IF Intermediate Frequency.

IP Internet Protocol.

IR Infrared.

LAN Local Area Network.

LHCP Left Hand Circular Polarization.

LNA Low Noise Amplifier.

LOS Line of Sight.

LoS Loss of Signal.

NASA National Aeronautics and Space Administration.

NFM Narrow FM.

NOAA National Oceanic and Atmospheric Administration.

NORAD North American Aerospace Defense Command.

PC Personal Computer.

PM2 Production Process Manager.

ppm Part Per Million.

RHCP Right Hand Circular Polarization.

ROE Razon de Onda Estacionaria.

SBC Single Board Computer.

SCP Secure Copy Protocol.

SDP Simplified Deep Space Perturbations.

SDR Software Defined Radio.

SGP Simplified General Perturbations.

SNR Signal to noise ratio.

SO Sistema Operativo.

SOX Sound eXchange.

SWR Standing Wave Ratio.

TCP Transmisión Control Protocol.

TIROS-N Television Infrared Observation Satellite - Next Generation.

TLE Two-Line Element.

UGR Universidad de Granada.

UHF Ultra High Frequency.

VHF Very High Frequency.

WEB World Wide Web.

CAPÍTULO

1

INTRODUCCIÓN

El presente documento desarrolla el Trabajo Final de Máster en Ingeniería de Telecomunicaciones y tratará de poner en práctica conocimientos adquiridos durante dicho máster. El objetivo de este proyecto es diseñar e implementar un sistema de recepción de señales de satélites meteorológicos en tiempo real. Los satélites meteorológicos para los que el sistema ha sido desarrollado son los satélites correspondientes a la serie NOAA. Los datos enviados por dichos satélites en tiempo real son imágenes de la superficie terrestre. Concretamente, este proyecto tratará de decodificar la señal APT recibida en la estación terrena con ayuda de un receptor SDR y obtener la imagen correspondiente mediante software implementado en Python. Se creará una aplicación web que permita la descarga por un usuario final de dichas imágenes.

Este Trabajo Final de Máster ha sido realizado con la colaboración del grupo GranaSAT.

GranaSAT, cuyo logo se muestra en la Figura 1.1, es un proyecto aeroespacial llevado a cabo en la Universidad de Granada, cuyo objetivo es la construcción por estudiantes de un Cubesat. Este proyecto está coordinado por el Profesor Andrés María Roldán Aranda y permite a los estudiantes colaboradores adquirir conocimientos en el campo aeroespacial y una experiencia real en este área. Los equipos y laboratorios utilizados en los proyectos desarrollados en GranaSAT están ubicados en la Facultad de Ciencias de la Universidad de Granada.

1

Figura 1.1 – *Logo GranaSAT*

1.1 Motivación

La Ingeniería de Telecomunicación cubre un gran abanico de sectores en los que desarrollar los conocimientos académicos adquiridos y uno de ellos, por el cual siempre he sentido gran pasión, es la comunicación vía satélite. GranaSAT ofrece la oportunidad de trabajar en el área aeroespacial y desarrollar proyectos que puedan ser utilizados como un producto final. De este modo, GranaSAT pone en práctica la teoría, y de una manera colaborativa, se pretende que la Universidad de Granada alcance expertise en el sector aeroespacial.

El servicio web de satélites NOAA desarrollado, estará disponible para el personal de la Universidad de Granada (UGR), lo que motiva a la elaboración de un proyecto cuyo fin sea ser utilizado por un usuario final. Este servicio web ofrecerá información en tiempo real de los satélites NOAA.

1.2 Estructura del proyecto

En este proyecto, cada capítulo está directamente relacionado con el punto del Proceso de Ingeniería de Sistemas correspondiente. En el diagrama de flujo de la Figura 1.2 hay un esquema de este proceso seguido y en el Capítilo 3 se muestra la trazabilidad temporal seguida en el proyecto en un Diagrama de Gantt.

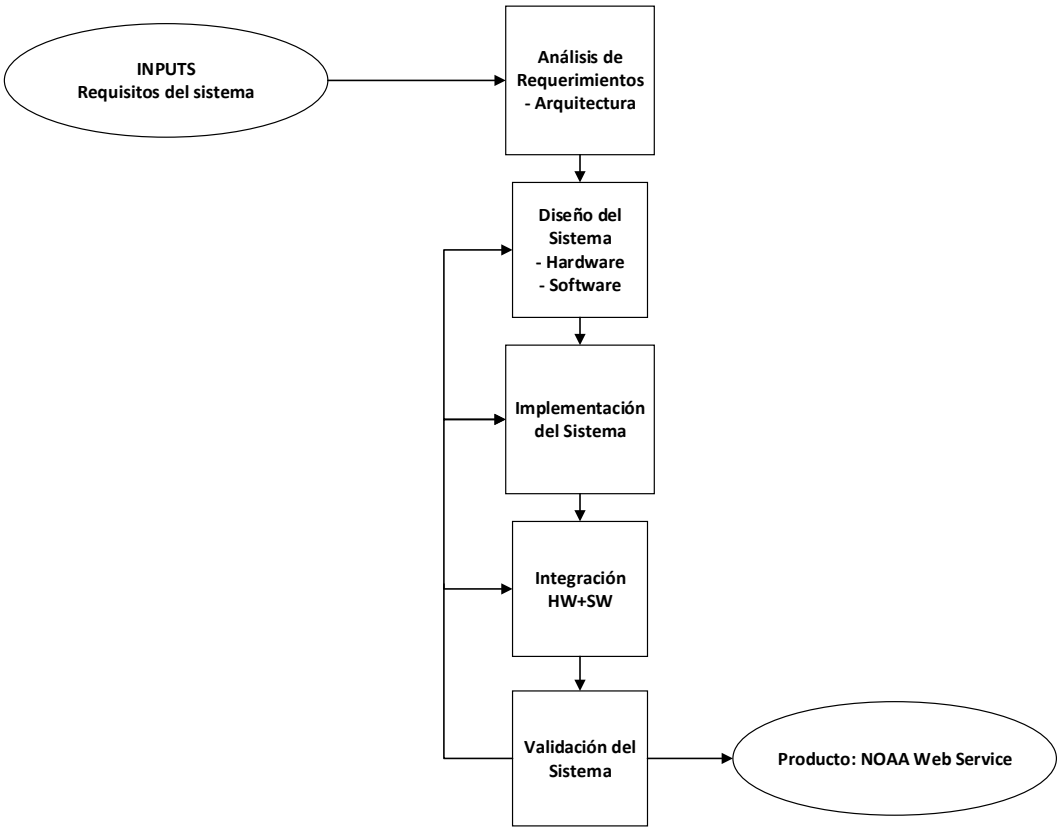


Figura 1.2 – *Diagrama de flujo del proceso del proyecto*

- **Capítulo 2: Objetivos de proyecto**: En este capítulo se observan los requisitos marcados por un supuesto cliente que se tendrán en cuenta para el desarrollo del proyecto y los requerimientos que a nivel técnico un ingeniero extrae para la correcta implementación del sistema y su posterior validación.

- **Capítulo 3: Planning**: Se establece el plan de tareas seguidas para realizar el proyecto, marcando la duración de las mismas. Se puede ver el diagrama de Gantt que ha seguido este trabajo.

- **Capítulo 4: Análisis de requerimientos**: En este capítulo se analizan los requisitos de partida y se realiza un estudio de la tecnología usada en la recepción de señales satelitales, además de analizar las características de los datos transmitidos por los satélites NOAA, extrayendo los inputs necesarios para el diseño del sistema. Además, trazan las pautas que se llevarán a cabo para la correcta evaluación del sistema.

- **Capítulo 5: Diseño del sistema**: Este capítulo desarrolla las pautas llevadas a cabo para el diseño e implementación del sistema software y hardware. Son tenidos

1

en cuenta los requerimientos del Capítulo 2 y su análisis del Capítulo 4. Se analizan diferentes soluciones y se selecciona aquellas que dan un valor añadido al sistema de recepción de imágenes NOAA.

- **Capítulo 6: Integración del sistema final**: Dada la modularidad que tendrá el sistema, es necesario establecer algunas medidas para su integración final software y hardware. Se muestra esquemáticamente como los subsistemas se integran en un único sistema final.
- **Capítulo 7: Evaluación y verificación**: Para llegar al producto final, es necesario seguir un plan de evaluación de los subsistemas que lo componen. En estecapítulo se muestran los resultados obtenidos en los diferentes test de medidas y las acciones llevadas a cabo para una mejora o reimplemntación, si fuera necesaria.
- **Capítulo 8: Resultados y conclusiones**: Finalmente, se muestran los resultados del producto final y se plantean una serie de líneas de trabajo futuras. Igualmente, en este capítulo, se extraen unas conclusiones del desarrollo del Trabajo Fin de Máster realizado.

CAPÍTULO

2

OBJETIVOS DEL PROYECTO

El Trabajo Final de Máster definido en este documento consiste en la implementación de un sistema de ingeniería orientado a producto. Los requisitos fundamentales son los plateados por el cliente final, que será aquel que hará uso del servicio WEB creado. De estos requisitos fundamentales derivan una serie de requerimientos internos técnicos evaluados desde un punto de vista ingenieril, tenidos en cuenta en la etapa de diseño e implementación hardware y software.

2.1 Requisitos

Los requisitos principales que definen la implementación del sistema a nivel funcional son dados por un cliente y se definen a continuación:

- **Recepción de señal de los satélites meteorológicos NOAA:** Para la recepción de las señales satelitales es necesaria la implementación de un sistema de recepción dedicado. La estación terrena constará como mínimo de una antena receptora, equipos de radiocomunicaciones propios para la recepción de las señales APT y una Central Processing Unit (CPU) encargada de la gestión del sistema y procesado de señal.

- **Implementación software para la recepción de imágenes NOAA:** Consiste en el desarrollo de una aplicación que automatice la recepción de señales satelitales en tiempo real durante el pase de cada satélite. El software implementado deberá extraer las imágenes transmitidas por cada uno de los satélites NOAA durante del pase.

- **Descarga de imágenes meteorológicas:** El servicio WEB servirá al usuario las imágenes recibidas de los satélites, que son almacenadas en una base de datos. El usuario podrá descargar las imágenes de diferentes pases.

2.2 Requerimientos técnicos

En base a los requisitos de la sección 2.1 se establecen especificaciones internas del sistema necesarias para el correcto funcionamiento del producto final. Los requerimientos descritos a continuación pueden ser funcionales o no funcionales, atendiendo a si el requerimiento afecta directamente a la funcionalidad del sistema o, por el contrario, son requerimientos basados en especificaciones tomadas por las características intrínsecas de los satélites NOAA o de los dispositivos a usar.

Los requerimientos técnicos se dividen en primarios y secundarios. Los requerimientos primarios son aquellos que, basándose en los requisitos del cliente, den un valor técnico añadido a las necesidades del cliente en el desarrollo del sistema. Los requerimientos secundarios son los derivados de los primarios, siendo estos más específicos.

2.2.1 Requerimientos primarios

(a) **Conocimiento de las características de los satélites NOAA:** Para conseguir la recepción de los datos de los satélites NOAA, es necesario hacer un estudio de esas características que puedan afectar al diseño del sistema: frecuencia de transmisión de cada satélite, modulación y codificación de la señal APT, ancho de banda de la señal.

Además, dado que los satélites NOAA son de órbita polar, conocidos como (POES) Polar Operational Environmental Satellites, será necesario realizar un *tracking* de los satélites para determinar el momento en que el satélite se sitúa sobre la estación terrena y el receptor comienza la escucha de la señal transmitida por dicho satélite.

(b) **Recepción y procesado en tiempo real:** Dado que una especificación fundamental es la correcta sincronización de la recepción de señal con el pase del satélite, es necesario usar un predictor que mediante las efemérides[1] del satélite, en todo momento, el sistema sea capaz de geolocalizar el satélite y activar el receptor durante el pase sobre la estación terrena.

Igualmente, se debe seleccionar la CPU que albergue el servicio con una capacidad de procesamiento adecuada al uso.

(c) **Servidor WEB:** Será necesario el desarrollo de la aplicación del lado del servidor realizando conexiones síncronas o asíncronas con el cliente. Se hará uso del protocolo HyperText Transfer Protocol (HTTP) para la comunicación, perteneciente a la capa de aplicación del Modelo OSI.

[1] Tabla de valores que da las posiciones de los objetos astronómicos en el cielo en un momento dado.

Esta aplicación realizará la tarea de recepción y procesado de señal APT, que para el cliente es invisible.

Se seleccionará el lenguaje de programación conveniente para el desarrollo de la aplicación del lado del servidor, donde las más usuales son PHP, Python, ASP.NET, Perl, entre otras.

(d) **Interfaz con usuario:** En principio el navegador del cliente podría resolver únicamente un código HTML enviado por el servidor, sin embargo, se plantea como un requerimiento primario que el cliente sea capaz de visualizar el *tracking* de los satélites en tiempo real, enviando la información del predictor de localización de los satélites NOAA. Es necesario crear una aplicación en el lado del cliente, mediante un mapa interactivo sobre el que posicionar los satélites en tiempo real. Se estudiará qué lenguaje de programación es conveniente para la aplicación en el lado del cliente donde las más usuales son Java y Javascript.

Se producirá una socket mediante Transmisión Control Protocol (TCP) al servidor situado en la dirección Internet Protocol (IP) de la Universidad de Granada, bajo el dominio `granasat1.ugr.es`

(e) **Recepción de señal:** Se diseñará y fabricará la antena receptora propiamente para satélites NOAA, dada la especificación de su banda de operación. Además, se debe diseñar con especificaciones electromagnéticas acorde a la funcionalidad, como tipo de polarización, ganancia, adaptación en la banda de operación y ancho de haz de caída de potencia -3 dB (HPBW).

(f) **Receptor Software Defined Radio:** Se empleará un receptor SDR por su fácil integrabilidad al sistema de recepción y las ventajas que tiene el diseño con dispositivos que implementan los equipos propios de radiocomunicaciones en software en lugar de hardware. Es necesario realizar un estudio para decidir qué modelo de SDR es más adecuado en nuestro sistema.

(g) **Decodificación de datos APT:** Atendiendo a las características de la señal APT estudiadas, se implementará la decodificación de la señal de audio, obteniendo la imagen resultante.

2.2.2 Requerimientos secundarios

(h) ***Tracking* satelital:** Para la visualización a través de la web de la posición de cada satélite, se utilizará una librería para la edición de mapas interactivos. Primará el uso de librerías Open Source.

(i) **Diseño web:** Para el diseño web se usará HTML y Cascading Style Sheets (CSS). Se estudiará la posibilidad de utilizar algún sistema de gestión de contenido (Content Management System (CMS), por sus siglas en inglés), que facilita la trazabilidad y seguimiento del contenido web. Algunas de las posibilidades más utilizadas son Joomla, WordPress, Drupal.

(j) **Mejoras en recepción:** Se estudiará la necesidad de otros equipos de radiofrecuencia como Low Noise Amplifier (LNA) o filtros paso banda para asegurar recepción de la señal con alta Signal to noise ratio (SNR).

(k) **Corrección de efecto Doppler:** Debido a que los NOAA son satélites de órbita polar, el movimiento relativo de estos sobre la estación terrena, se traduce en una desviación de la frecuencia portadora, que podría ser corregida durante la recepción o en post-procesado.

(l) **Sistema de recepción compacto y económico:** Dado que el proyecto se debe ajustar a las instalaciones y medios económicos facilitados por GranaSAT y la UGR, el sistema implementado debe desarrollarse con vista a la optimización de recursos.

CAPÍTULO

3

PLANNING DEL PROYECTO

De acuerdo a la estructura de proyecto marcada en el Capítulo 1 y la los requisitos de cliente y los requerimientos técnicos evaluados en el Capítulo 2, se realiza el planning del proyecto. El diagrama de Gantt nos ayuda a seguir una organización de tareas y además nos permite el control temporal de las mismas.

En este proyecto existe un único recurso, por lo que muchas tareas que podrían desarrollarse de manera paralela se han tenido que desarrollar en serie.

En la Figura 3.1 se puede ver el planning seguido para la implementación del sistema.

DIAGRAMA DE GANTT

	Nombre de tarea	Duración	Comienzo	Fin
1	**Servicio WEB para satélites meteorológicos NOAA basado en receptor SDR**	**497 días**	**jue 01/10/15**	**vie 25/08/17**
2	**Análisis de requisitos**	**24 días**	**jue 01/10/15**	**mar 03/11/15**
3	Diseño conceptual y funcional del sistema	24 días	jue 01/10/15	mar 03/11/15
4	**Diseño y desarrollo de servicio web**	**200 días**	**mar 03/11/15**	**lun 08/08/16**
5	**Implementación de aplicación tracking satelital**	**64 días**	**mar 03/11/15**	**vie 29/01/16**
6	Programación back-end del propagador satelital	20 días	mar 03/11/15	lun 30/11/15
7	Programación front-end de mapa interactivo con tracking satelital	29 días	mar 22/12/15	vie 29/01/16
8	Programación front-end de información satelital	20 días	mar 26/01/16	lun 22/02/16
9	Implementación de gestión de biblioteca de imágenes	22 días	vie 10/06/16	lun 11/07/16
10	Diseño web HTM+CSS	20 días	mar 12/07/16	lun 08/08/16
11	**Diseño y fabricación de antenas**	**273 días**	**lun 04/01/16**	**mié 18/01/17**
12	Diseño CST Eggbeater	35 días	lun 04/01/16	vie 19/02/16
13	Diseño alimentación Eggbeater	15 días	lun 22/02/16	vie 11/03/16
14	Fabricación antena Eggbeater	20 días	lun 14/03/16	vie 08/04/16
15	Mediciones de parámetros de antena y optimización	15 días	lun 11/04/16	vie 29/04/16
16	Fabricación antena alternativa	4 días	lun 09/01/17	jue 12/01/17
17	Medición de antena alternativa	4 días	vie 13/01/17	mié 18/01/17
18	**Sistema de recepción con SDR**	**110 días**	**mar 01/12/15**	**lun 02/05/16**
19	Análisis de receptor SDR	30 días	mar 01/12/15	lun 11/01/16
20	**Control software de SDR con Raspberry para NOAA**	**80 días**	**mar 12/01/16**	**lun 02/05/16**
21	Análisis librería RTLSDR Linux/Windows	30 días	mar 12/01/16	lun 22/02/16
22	Implementación en Python sistema de recepción	50 días	mar 23/02/16	lun 02/05/16
23	**Decodificación imagen**	**50 días**	**vie 01/04/16**	**jue 09/06/16**
24	Análisis de comportammiendo APT	20 días	vie 01/04/16	jue 28/04/16
25	Análisis alternativas de decodificador APT	20 días	vie 29/04/16	jue 26/05/16
26	Integración decodificador APT	10 días	vie 27/05/16	jue 09/06/16
27	Pruebas de validación del sistema	55 días	lun 02/01/17	vie 17/03/17
28	Reimplementación y optimización según evaluación	40 días	lun 03/04/17	vie 26/05/17
29	Documentación	126 días	vie 03/03/17	vie 25/08/17

Figura 3.1 – *Diagrama de Gantt.*

CAPÍTULO

4

ANÁLISIS DE REQUERIMIENTOS

Siguiendo los requerimientos primarios y secundarios establecidos en el Capítulo 2, estos se deben analizar con detalle para proceder al diseño e implementación del sistema. Este análisis de requerimientos permitirá conocer la tecnología disponible a partir de la cual desarrollar el proyecto. Durante el análisis se determinará qué herramientas son las idóneas, o más interesantes, según diferentes puntos a evaluar como es su ajuste a las especificaciones dadas por el cliente, costo y conocimiento en la materia.

El análisis de requerimientos se desarrolla en tres grandes bloques:

- Extracción de requerimientos no funcionales en base al conocimiento de la señal recibida por los satélites NOAA a tratar. En la sección 4.1 se define la señal APT, su frecuencia downlink, información referente a decodificación, etc. Igualmente, se definen las herramientas disponibles para realizar el seguimiento del satélite de acuerdo a sus efemérides.

  En esta sección se lleva a cabo el análisis detallado del requerimiento a y además se marcan pautas que ayudan a definir los requerimientos b, g y h.

- Análisis físico del sistema caracterizado por la revisión de los elementos hardware y software que lo integran para cumplir con los requerimientos especificados. Se trata de una definición más detallada de los requerimientos funcionales primarios b, c, d, e y f, consistentes en el diseño WEB y el diseño del sistema de recepción de la señal de los satélites NOAA.

Asimismo, se va a mostrar en este capítulo la arquitectura de sistema a nivel global, marcando el punto de partida en el desarrollo del proyecto y además se analizarán los requerimientos secundarios que dan valor añadido al sistema.

Este estudio corresponde a la sección 4.2

- Descripción de pruebas de evaluación y verificación a partir de las cuales determinar si se valida el sistema o subsistemas y se cumple con los requerimientos (véase sección 4.3). Mediante estas pruebas se debe encontrar el origen del error y volver atrás tal y como marca el diagrama de flujos mostrado en la Figura 1.2.

4.1 Análisis de las características de satélites NOAA

4

La denominada misión NOAA, en la cual colaboró la NASA para el lanzamiento de los satélites (POES) Polar Operational Environmental Satellites, comenzó con el prototipo Television Infrared Observation Satellite - Next Generation (TIROS-N), lanzado en octubre de 1978, seguido por el lanzamiento de NOAA-A en junio de 1979, que fue renombrado después de su lanzamiento como NOAA-6. La serie TIROS-N, formada por los satélites NOAA-6 y NOAA-7, fue la primera en incluir el instrumento Advanced Very High Resolution Radiometer, (AVHRR) en su payload. En marzo de 1983 se lanzó el primero de los satélites de la serie *Advanced TIROS-N*, denominado NOAA-E y renombrado como NOAA-8 después de su lanzamiento. Estos satélites son físicamente más largos y tienen más potencia que sus predecesores.

A partir de NOAA-k (renombrado como NOAA-15), lanzado el 13 de mayo de 1998, aparece una generación de satélites con mejoras tecnológicas en su instrumentación. Concretamente, se mejora el sensor Advanced Very High Resolution Radiometer 3º Generation, (AVHRR/3), permitiendo trabajar con mayor capacidad de discriminación entre nubes, hielo y nieve. Los satélites de la serie NOAA de órbita polar activos actualmente son NOAA-19, NOAA-18 y NOAA-15 [4][18]. El resto han sido retirados de servicio o tienen instrumentación importante no operativa.

En esta sección se analizarán los inputs necesarios para atender al requisito a, fundamental para el desarrollo del sistema.

4.1.1 Características del sistema APT

En la siguiente Tabla se muestra la frecuencia de downlink para la transmisión APT de los satélites que permanecen activos:

Satélite	Frecuencia downlink APT
NOAA-15	137.62 MHz
NOAA-18	137.9125 MHz
NOAA-19	137.1 MHz

Tabla 4.1 – *Frecuencias de operación de satélites (POES) Polar Operational Environmental Satellites operativos*

La señal APT es analógica y de más baja resolución que el tipo de transmisión HRPT. Tiene una resolución de pixel de aproximadamente 4 km, frente a 1.09 km de HRPT. Los datos APT son transmitidos continuamente y son recibidos en tiempo real por los equipos de cualquier estación terrestre, que no tienen que ser demasiado sofisticados.

Los datos digitales obtenidos por el instrumento Advanced Very High Resolution Radiometer 3º Generation, (AVHRR/3) pasan a un conversor analógico-digital del sistema Manipulated Information Rate Processor [5], se filtran y se modulan sobre una portadora a 2400 Hz. La señal APT se transmite sobre una portadora Frequency Modulation (FM) modulada por una subportadora Amplitude Modulation (AM) de 2.4 kHz y una desviación de ±17 kHz. El ancho de banda de la señal transmitida es de aproximadamente 40 kHz (ver ecuación 4.1.1), que corresponde con la tasa de muestreo de los seis canales de Advanced Very High Resolution Radiometer 3º Generation, (AVHRR/3).

$$BW = 2 \cdot (17 \text{ kHz} + 2.4 \text{ kHz}) = 38.8 \text{ kHz} \approx 40 \text{ kHz} \tag{4.1.1}$$

En cuanto a la resolución espectral de Advanced Very High Resolution Radiometer 3º Generation, (AVHRR/3), este está formado por seis canales, que miden la radiación en cinco bandas simultáneas del espectro electromagnético, como se observa en la Tabla 4.2. Los dos canales de transmisión APT son multiplexados en tiempo obteniendo un canal de 'video A' (en la región visible durante los pases de día) y un canal de 'video B'. Un canal en la región visible del espectro de radiofrecuencia de Advanced Very High Resolution Radiometer 3º Generation, (AVHRR/3) es usado para obtener la imagen APT durante las horas de luz y un canal Infrared (IR) es usado durante el día y la noche. Un segundo canal IR es programado para reemplazar al canal en el rango visible durante la órbita en zona de oscuridad. Así pues, los canales 2 (visible) y 4 (IR) del sensor Advanced Very High Resolution Radiometer 3º Generation, (AVHRR/3) son los que transmiten en las horas del día y los canales 3 (IR) y 4 (IR) en las pasadas nocturnas (según Tabla 4.2)

Número de canal	Longitud de onda (μm)	Uso de canal
1	0.58-0.68	Nubes durante el día y mapeo de superficie.
2	0.725-1.00	Límites tierra-agua.
3A	1.58-1.64	Detección de nieve y hielo.
3B	3.55-9.93	Mapeo de nubes en la noche, temperatura superficial del mar.
4	10.30-11.30	Mapeo de nubes en la noche, temperatura superficial del mar.
5	11.50-12.50	Temperatura superficial del mar.

Tabla 4.2 – *Características de los canales del sensor Advanced Very High Resolution Radiometer 3º Generation, (AVHRR/3).*

La alta resolución temporal de los NOAA permite el estudio de la evolución de fenómenos atmosféricos, y además, al combinar los tres satélites operativos, se consigue una mayor frecuencia de cobertura de una zona. La resolución temporal hace referencia a la periodicidad en la que se toman imágenes de una misma porción de la superficie terrestre, que en este caso es de 12 horas.

La Tabla 4.3 representa las características de transmisión APT:

Tasa de línea	120 líneas/min
Canales de transmisión	2 transmisores de 6 disponibles
Resolución de imágenes	4 km
Modulación de portadora	Subportadora AM de 2.4 kHz sobre portadora FM
Ancho de banda	38.8 kHz
Frecuencia del transmisor	137 - 138 MHz
Potencia del transmisor	5 W (37 dBm)
Potencia radiada	36.7 dBm
Polarización de antena	Circular a derechas (RHCP)

Tabla 4.3 – *Características de la transmisión APT [7].*

La transmisión APT de los satélites NOAA se compone de dos medias líneas (una por canal A y B), donde cada una corresponde a capturas de diferentes regiones del espectro. Una trama de información está formada por 128 líneas que se envían cada 64 segundos. En la Figura 4.1 y 4.2 se muestra el formato de la transmisión APT.

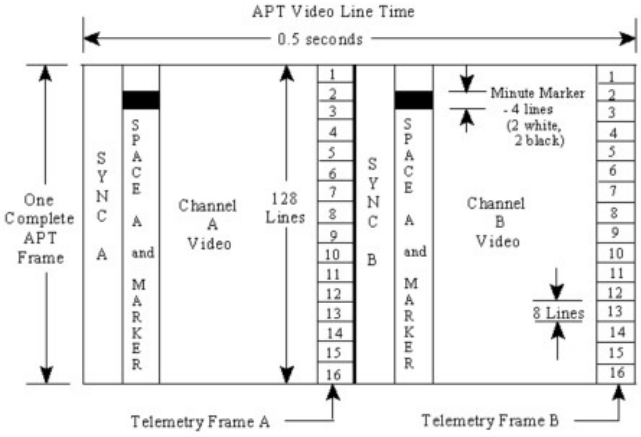
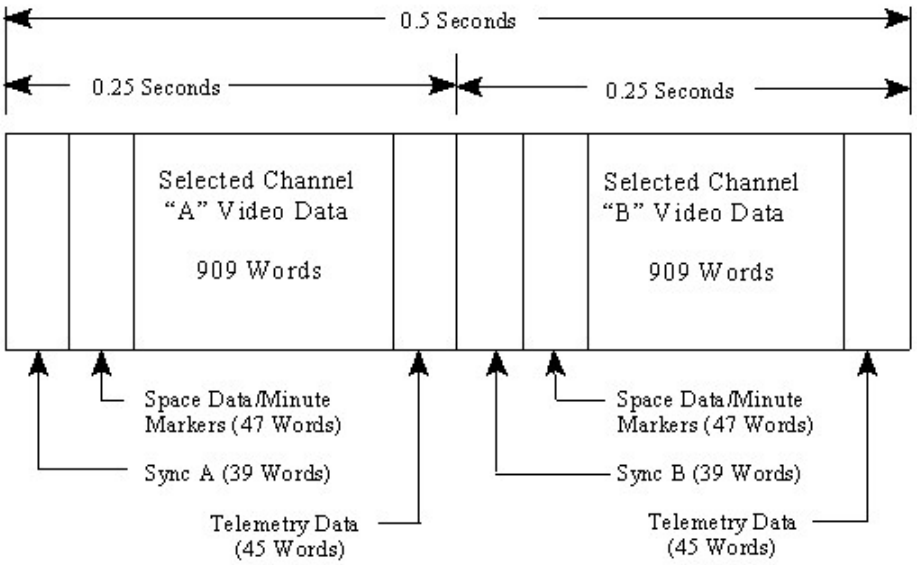


Figura 4.1 – *Esquema del formato APT de los satélites NOAA [7].*



Notes:

1) Equivalent output digital data rate is 4160 words/second
2) Video line rate - 2 lines/second
3) APT frame size - 128 lines
4) Any two of the five (six for group 505) AVHRR channels may be selected for use
5) Sync A is a 1040-Hz square wave - 7 cycles
6) Sync B is an 832-pps pulse train - 7 pulses
7) Each of the 16 telemetry points is repeated on 8 successive lines
8) Minute markers are repeated on 4 successive lines, with 2 lines black and 2 lines white

Figura 4.2 – *Descripción de las características de transmisión de APT[7].*

Estos datos serán los inputs necesarios para alcanzar los requerimientos primarios e y g, consistentes en la recepción y decodificación de las imágenes transmitidas por los satélites.

4.1.2 Two-Line Element

Los satélites (POES) Polar Operational Environmental Satellites de la agencia NOAA ofrecen como ventaja sobre los Geostationary Operational Environmental Satellites (GOES) el dar una cobertura global diaria gracias a que giran en torno a la Tierra 14 veces al día, con un periodo aproximado de 102 minutos, cubriendo en cada imagen un ancho aproximadamente de 3000 km. Además, debido a su baja órbita, con una altura orbital comprendida entre los 830 y 870 Km sobre la superficie terrestre, sus imágenes ofrecen una mayor resolución.

Por otro lado, como inconveniente frente a los GOES, es necesario el seguimiento de la órbita de los satélites (POES) Polar Operational Environmental Satellites. Para calcular la posición de los satélites en el espacio se hace uso de los elementos keplerianos, que son un conjunto de números que definen la órbita satelital en un instante determinado. Los elementos keplerianos principales para definir la órbita satelital son: época, inclinación orbital, ascensión recta del nodo ascendente (RAAN), argumento del perigeo, excentricidad, movimiento medio y anomalía media [9]. Estos elementos vienen dados en dos formatos diferentes: el formato Two-Line Element (TLE) producido por la NASA y North American Aerospace Defense Command (NORAD) y el formato de Radio Amateur Satellite Corporation (AMSAT). En la Figura 4.3 se puede ver la información aportada por el formato TLE. Los TLE están disponibles para su descarga desde Celestrack [22] de forma gratuita.

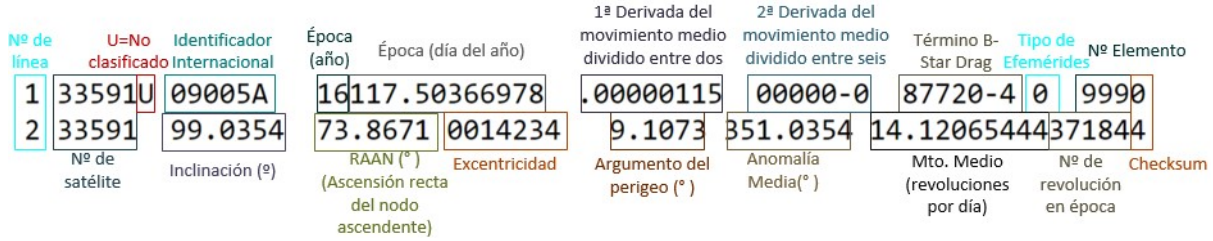


Figura 4.3 – *Formato NASA de Two-Line Element usado con los modelos SGP. Ejemplo de formato de dos líneas de NOAA-19.*

Es posible realizar el seguimiento del satélite a partir de su TLE y empleando modelos matemáticos de propagación orbital como son los cinco modelos SGP, SGP4, SDP4, SGP8 y SDP8. El modelo SGP se aplica en objetos con un periodo orbital inferior a los 255 minutos y Simplified Deep Space Perturbations (SDP) para periodos orbitales mayores a 255 minutos. Es por ello que el modelo SGP4/SDP4 combina ambos. El modelo SGP8/SDP8 fue creado para solventar algunas deficiencias de SGP4/SDP4 para casos especiales de decaimiento orbital, pero sigue siendo el modelo SGP4/SDP4 [27] el usado para operar con TLE. Todos

estos modelos incluyen en sus algoritmos matemáticos las perturbaciones causadas por la forma de la Tierra, los efectos de gravedad de otros cuerpos celestes y la radiación solar.

Existen diversos software que utilizan el modelo de propagación SGP4/SDP4 y los TLE para el cálculo del seguimiento satelital. Entre ellos se encuentra Gpredict, un software con licencia Licencia Pública General disponible para Windows, Linux y MAC OS X, u Orbitron, software libre disponible para Windows. Otros software que ofrecen muy buenas prestaciones, pero propietarios, son *Ham Radio Deluxe* y *SATPC32*.

El servicio propuesto en este proyecto para la descarga y almacenamiento de imágenes meteorológicas también ofrecerá el seguimiento de los satélites NOAA. El cliente debe poder visualizar la posición del satélite en tiempo real. De modo que, se analizará qué tipo de predictores usar, sabiendo que debe interpretar las TLE de los satélites y que los modelos empleados son SGP4/SDP4. A continuación se realizará un estudio de diferentes librerías disponibles de predictores de órbita satelital para finalizar con el cumplimiento del requerimiento a con lo que respecta a *tracking* en tiempo real.

4.1.3 Predictores de órbita satelital

Dado que el servicio WEB a desarrollar se implementará usando Python (tal y como se detalla en la sección 4.2.4.1), se ha realizado una revisión de las librerías disponibles para Python que ayuden en el cálculo del seguimiento de la órbita de satélites (POES) Polar Operational Environmental Satellites. A continuación se definen las librerías encontradas Open Source, cumpliendo con el requerimiento h.

1. **PyEphem:** Proporciona cálculos astronómicos básicos para lenguaje de programación Python. Dada una fecha y una ubicación en la superficie de la Tierra, se calcula la posición de planetas, la Luna, el Sol o distintos satélites dados sus elementos keplerianos (usando TLE en este caso). Se puede acceder a una mayor información y a la descarga de esta librería en [35]. La forma más fácil de instalar es usar el comando *pip*, tras instalar archivos de cabecera de Python. Con esta librería se puede obtener cómodamente la posición del satélite en diferentes instantes temporales.

2. **PySGP4:** Este paquete de Python calcula la posición y la velocidad de un satélite en órbita terrestre, dados los elementos orbitales TLE del satélite de una fuente como *Celestrack* [22]. Implementa la versión más reciente de *SGP4* y se ejecuta regularmente para asegurarse de que sus predicciones de posición de satélite coinciden en 0,1 mm de las predicciones de la implementación estándar del algoritmo en C++. Este error es mucho menor a 1-3 km/día, de modo que los propios satélites se desvían de las órbitas ideales descritas en los archivos TLE. Se puede descargar a través de *Github* [12].

3. **PyPredict:** Es una extensión para Python directamente adaptada de la librería Predict de C de seguimiento por satélite usada en software como Gpredict. PyPredict es una API para la base de código de predicción en C y debe producir resultados

idénticos. Devuelve una observación del satélite relativa a la *groundstation*. Como parámetros de entrada necesita los elementos del satélite TLE y QTH de la *groundstation* en coordenadas (latitud, longitud y altitud). Además de esto, permite definir los instantes de inicio y fin del pase del satélite sobre la estación terrena, futuros pases satelitales y un diccionario completo de información del pase como desviación doppler, velocidad del satélite, identificador NORAD, footprint, entre otros. Está disponible para su descarga desde *Github* en [19].

Analizando las facilidades de uso de las diferentes librerías expuestas. PyEphem y Pypredict resultan ser las más interesante para que la WEB muestre el *tracking* satelital de la manera más cómoda y el receptor de la señal APT se sincronice de manera óptima al pase del satélite. Más adelante, se detallará en qué ha sido empleada cada una de las librerías y las pruebas realizadas con estas.

4.2 Análisis de herramientas empleadas en el sistema

En la Figura 4.4 se plasma de forma visual los diferentes bloques que integran el sistema. En primer lugar, será necesaria una antena receptora en la banda de operación de los satélites NOAA. Esta antena debe estar conectorizada a un receptor que, igualmente, opere en la banda de frecuencia de los NOAA (banda Very High Frequency (VHF)). Se trabajará con SDR. Este sistema de radiocomunicaciones permite un procesado de señal software, ofreciendo así una gran versatilidad, dado que son configurables desde el Personal Computer (PC) del ingeniero. Mezcladores, moduladores, demoduladores y filtros integrantes se implementan en software, con mayores posibilidades de configuración que si se trataran de equipos hardware para un propósito específico. SDR debe ir acompañado de un PC o un equipo de computación embebida. En este caso se contempla la idea de usar una Raspberry Pi, pensando inicialmente en una integración lo más compacta posible del sistema de recepción, como indica el requerimiento i. Con estos dispositivos se conseguirá la recepción de la señal analógica APT y su posterior decodificación para obtener las imágenes tomadas por los sensores ópticos de los satélites meteorológicos NOAA.

Dado que el requisito principal es hacer llegar a un usuario final las imágenes de los NOAA, estas imágenes decodificadas deben ser accesibles. Para ello, se requiere de un Servidor WEB que haga llegar las imágenes a los usuarios mediante TCP/IP. Este Servidor WEB podría ser la propia Raspberry Pi, teniendo acceso a la base de datos de almacenamiento de las imágenes de los NOAA. A continuación se analizarán las posibilidades y se seleccionará aquella que mejor se adapte a los objetivos.

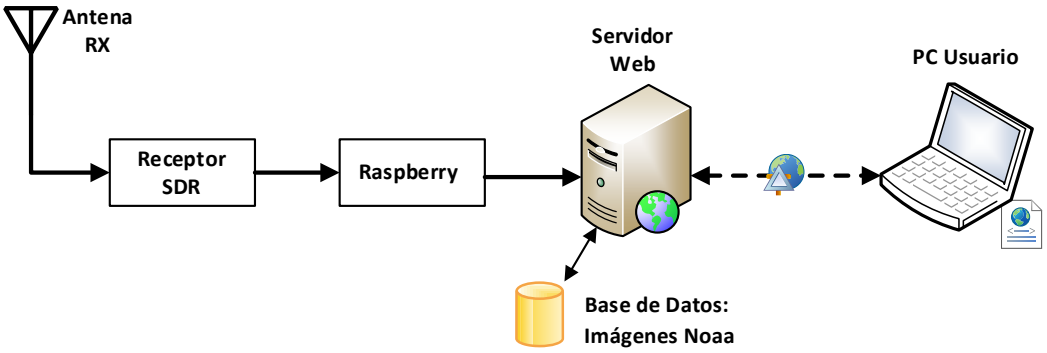


Figura 4.4 – *Arquitectura física general de la recepción y servicio WEB de imágenes NOAA.*

4.2.1 Antena receptora

Un elemento fundamental del proceso en la recepción de señal APT de los satélites NOAA es la antena de recepción. Esta debe seguir unas especificaciones dadas por la señal downlink de los satélites (ver Tabla 4.1) y considerando las señales interferentes en la banda de frecuencia de recepción. Por ello y siguiendo la especificaciones de Cuadro Nacional de Atribución de Frecuencia (CNAF), se debe diseñar una antena que filtre la banda asignada a radio comercial FM (87.5-108 MHz) y la banda para radioaficionados (144-146 MHz), cercanas a nuestra banda de trabajo (137-138 MHz).

Para el cumplimiento del requerimiento e, que trata sobre la recepción de datos procedentes de satélites, es necesario el uso de una antena de polarización circular debido a las ventajas que ofrece con respecto a una antena de polarización lineal. Por lo general, los satélites usan RHCP y no es necesario alinear la dirección de polarización entre antena receptora y transmisora ya que la orientación de la antena del satélite no siempre será la misma, siempre que ambas antenas presenten el mismo tipo de polarización. Esto es debido al efecto Faraday, las transmisiones satelitales experimentan una rotación del plano de polarización. Además, la falta de alineación de la dirección de polarización de ambas antenas exige con polarización lineal márgenes de protección de polarización cruzada mayores que en polarización circular. Otro punto a considerar es que, debido a que los NOAA son satélites de órbita polar, no es necesario que la antena sea muy directiva, pero sí que el ancho de haz sea muy grande, de este modo se podrá recibir la señal del satélite sin necesidad de mover la antena con un sistema de rotores con movimiento en azimut y elevación. Consiguiendo buen HPBW, la antena podrá recibir la señal del satélite cuando esté en linea de vista con la antena de la estación terrena.

Por tanto, el objetivo es diseñar una antena con un diagrama de radiación sectorial en

el plano vertical y omnidireccional en el horizontal que cubra el pase de los satélites NOAA desde que sea visible por el horizonte Acquisition of Signal (AoS) hasta que deje de serlo Loss of Signal (LoS), tal y como se aprecia en la figura 4.5.

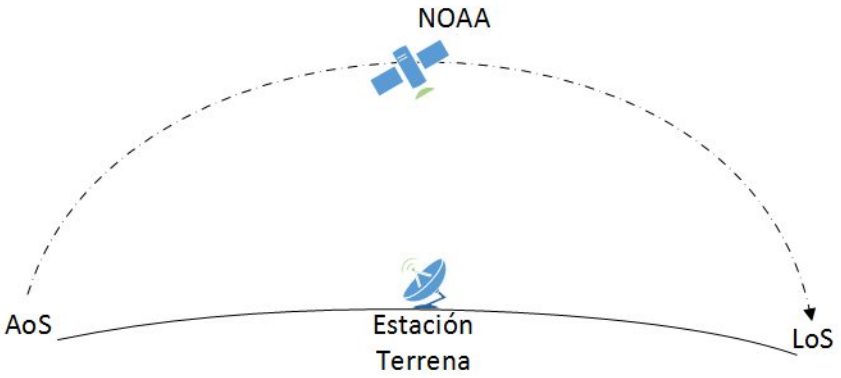


Figura 4.5 – *Trayectoria que debe cubrir el diagrama de radiación de la antena a diseñar.*

A continuación se mencionan posibles alternativas de antenas para recepción de satélites de órbita polar, como los NOAA. Todas ellas cumplen con los requerimientos indicados.

1. **Antena Turnstile:**

   Es una antena de simple diseño y fácil de fabricar. Consiste en dos dipolos de longitud $\lambda/2$ cruzados, como se aprecia en la Figura 4.6. Presenta un diagrama de radiación omnidireccional en el plano horizontal, por lo que es buena candidata para la recepción del APT, aunque no es óptima en pases del satélite de baja elevación.

Figura 4.6 – *Antena Turnstile.*

   Presenta problemas con respecto a la ganancia, ya que apenas aumenta la intensidad de la señal y la recepción se hace complicada.

2. **Antena Cuadrifiliar Helicoidal (QHA):**

Es una antena omnidireccional con un mejor diagrama de radiación que el de *tunstile*. Además no tiene pérdidas de intensidad de señal. En la Figura 4.7 se puede ver cómo es la antena *QHA*.

Figura 4.7 – *Antena QHA.*

El principal inconveniente es que su fabricación es algo más complicada que la *tunestile*. Se ha decidido no fabricar esta antena dado que hay opciones más simples en fabricación (como la *turnstile*), a que GranaSAT ya dispone de una *QHA* para VHF, pudiéndose reutilizar para la banda de operación de los NOAA y no da un valor añadido a esta acción dentro del proyecto.

3. **Antena Eggbeater:**

   Esta antena también cumple con los requisitos para recepción del APT, ya que presenta un diagrama de radiación omnidireccional y es de polarización circular. Consiste en dos loops desfasados 90º hechos de un material radiante. En la Figura 4.8 se puede ver la antena descrita.

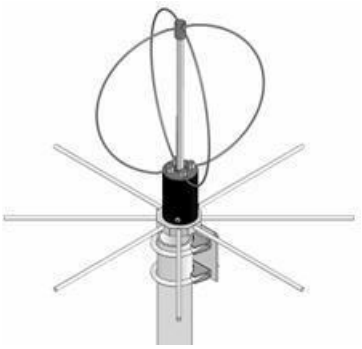


Figura 4.8 – *Antena Eggbeater.*

En comparación con *turnstile*, esta antena presenta las siguientes ventajas:

- Favorece a la recepción en pases de satélite de menor elevación, pero es algo peor en ángulos de 90º.
- El ruido producido por un loop es menor que el producido por un dipolo, aumentando así la relación señal ruido (SNR).

- El loop presenta una ganancia de 1.25dBd (sobre la ganancia del dipolo).

- Es más compacta que la *turnstile*, ya que el reflector de la antena está a una distancia de 1/8 de longitud de onda del elemento radiante, mientras que en la *turnstile* está a 1/4.

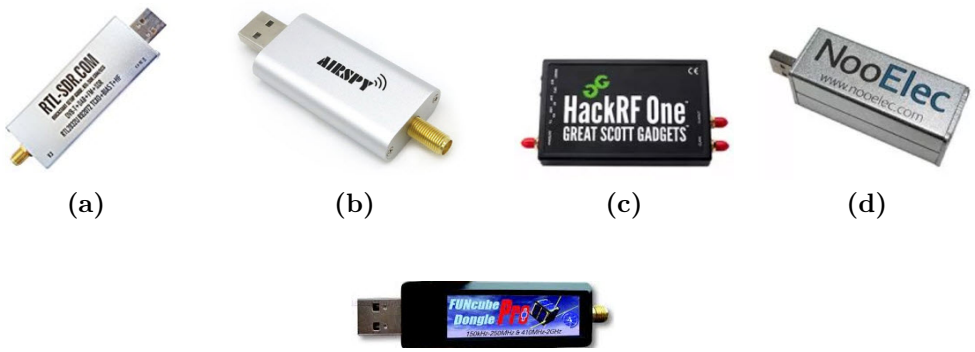
Por otro lado, comparándola con *QHA*, ambas presentan características similares aunque *QHA* presenta un mejor diagrama de radiación, pero mayor dificultad en la fabricación.

4

Finalmente, se decidió optar por el diseño y fabricación de una antena *Eggbeater*. Igualmente, se ha fabricado una antena *turnstile* para comparar ambas y validar el sistema con diversos hardware de recepción.

4.2.2 Receptor SDR

Para la recepción de las señales APT de los satélites NOAA no es necesario equipos demasiado sofisticados o de precio elevado. A continuación se muestran alternativas presentes en el mercado que cumplen con el requerimiento f y el requerimiento l, búsqueda de un sistema de recepción compacto y económico. La siguiente Tabla comparativa ayuda a tomar la decisión a cerca de la alternativa más favorable a nuestra aplicación:



(a) **(b)** **(c)** **(d)** **(e)**

Figura 4.9 – *Modelos SDR: a) RTL-SDR (RTL2832U R820T). b) Mini Airspy. c) HackRF One. d) NooElec NESDR. e) FUNcube dongle Pro+*

	RTL-SDR (RTL2832U R820T)	**Mini Airspy**	**HackRF One**	**NooElec NESDR**	**FUNcube dongle Pro+**
Precio	9 - 20 €	85 - 120 €	270 - 460 €	15 - 40 €	150 €
Frecuencia	500kHz-1.7GHz. (*)500kHz a 24MHz con prestaciones reducidas	25MHz-1.8GHz	1MHz-6GHz	Basado en R820T: 25MHz-1.7GHz. Basado en E400: <500MHz-2.35GHz	150kHz-260MHz y 410MHz-2.05GHz
Software compatibles	SdrSharp, HDSDR, SeeDeR, Gqrx, SdrDx, Sdr-Touch, RTL_FM, RTL_UDP, etc.	SDR#, SDR-Radio, HDSDR, GQRX, GNU Radio, etc.	GNU Radio, SDRSharp, etc.	SdrSharp, HDSDR, SeeDeR, Gqrx SDR, SdrDx, Sdr-Touch, RTL_FM, RTL_UDP, etc.	SdrSharp, Gqrx SDR, Sdr-Radio V2 avance, FDC-DF, SdrDX, DRM, SeeDeR.
Detalles técnicos	Hasta 3,2 MHz de frecuencia de muestreo, RTL2832U/ R820T2 con mayor banda que RTL2832U/ E400 y más económico, tcxo de alta estabilidad (1PPM) según modelo.	Ancho de banda de 6 MHz libre de espurios, tcxo de alta precisión (0.5 ppm), ADC de 12 bits	Transceiver. Software de configuración RX y TX, Software para control de potencia del puerto de antena tx, hardware open source.	Demodulador RTL2832U, chip E400 presenta mejores prestaciones que RT820T, txco de hasta 0.5PPM según modelo.	SAW filtros selectivos en 2m y 70cm, 0.5 PPM tcxo, ADC 16 bits, 192kHz velocidad de muestreo.

Tabla 4.4 – *SDR populares en el mercado.*

Todos estos dispositivos tienen aplicaciones similares: monitorización radio, escaneo de banda ancha, Air Traffic Control (ATC), Automatic Dependent Surveillance-Broadcast,

Automatic Identification System, Aircraft Comunicación Addressing and Reporting System, Automatic Packet Reporting System, APT, radio-astronomía, recepción de telemetría, educación, radio Amateur, y más. Todos ellos son válidos para la recepción de la señal APT de los NOAA. Dicho esto, se selecciona un dispositivo atendiendo principalmente a su precio. Se elige un dispositivo de banda ancha de bajo costo basado en RTL-SDR. El hardware usado es uno de los más baratos del mercado. En la figura 4.10 se puede ver el kit Realtek DVT-T RTL-SDR disponible por un precio no superior a 10 €. El receptor RTL-SDR, a usar por nuestro sistema de recepción de datos APT, consta del chip RTL2832U, un conector USB 2.0 y un conector de antena TV hembra, o bien, el modelo con conector SMA hembra).

Figura 4.10 – *SDR de Realtek RTL2832U y sus componentes integrados en el kit puesto en venta.*

Los equipos SDR definidos, son soportados por diversas plataformas: Linux, Windows, OS X, Raspberry Pi, etc.

El chip RTL2832U, visible en la figura 4.11, es un demodulador COFDM (Coded Orthogonal Frequency Division Multiplexing), una técnica de modulación para transmitir información digital, usada en Digital Video Broadcasting- Terrestrial (DVB-T) y Digital Audio Bradcasting (DAB). También permite recepción de Radio FM. RTL2832U soporta múltiples frecuencias Intermediate Frequency (IF) gracias al cristal (tcxo) de bajo coste que genera una señal de reloj de $\pm 100 ppm$ (partes por millón). Por último, indicar que está empotrado con un conversor analógico digital (Analog-to-Digital Conversion (ADC)) [20].

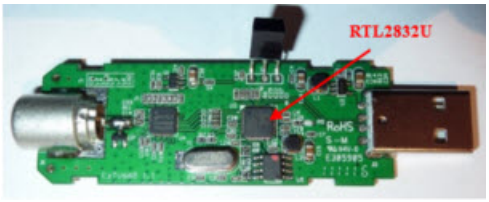


Figura 4.11 – *Interior de SDR de bajo costo.*

A continuación se representa un diagrama de bloques que sigue el dispositivo *RTLSDR*:

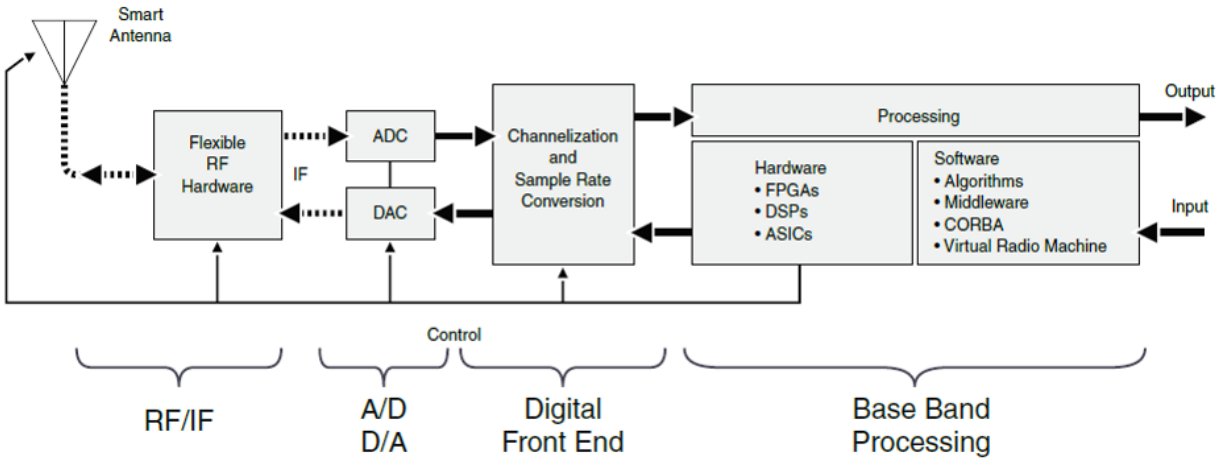


Figura 4.12 – *Diagrama de bloques de RTLSDR [32].*

Existen diversos software con los que RTL2832U es compatible, como se ha visto en la Tabla 4.4. La mayoría de estos se basan en la librería *librtlsdr*. RTL-SDR contiene tanto esta librería, como herramientas de la línea de comandos que realizan funciones de transferencia de datos desde y hacia el dispositivo: *rtl_test, rtl_eeprom, rtl_sdr, rtl_fm y rtl_tcp*. Este software libre está disponible en [17] para su descarga y conocimiento del manejo de sus herramientas.

Algunos de los Software más usados que permiten trabajar con RTL-SDR son los siguientes:

- SDRSharp: software libre para Windows.

- HDSDR: software libre para Windows. Soporta RTL SDR a través del módulo ExtlO.dll.

- GNU Radio: software libre para Linux. Se trata de un potente programa de procesado de señal digital.

4.2.3 Raspberry Pi

En esta sección se van a analizar las posibilidades existentes en el mercado para la selección más apropiada de Raspberry Pi (ver Tabla 4.5). Se ha decidido usar este dispositivo como PC embebido debido a que se trata de un hardware compacto (cumpliendo así la especificación l), que se puede colocar junto a la antena, evitando así el despliegue de cableado y disminuyendo, por tanto, las pérdidas correspondientes.

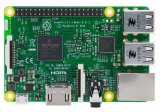
	Raspberry Pi A	**Raspberry Pi A+**	**Raspberry Pi B**	**Raspberry Pi B+**	**Raspberry Pi 2B**	**Raspberry Pi 3B**
						
Precio	21€	17 €	25 €	25 €	35 €	Desde 35 €
SoC (Chip)	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2836	Broadcom BCM2837
CPU	ARM 1176JZF-S a 700 MHz	ARM 1176JZF-S a 700 MHz	ARM 1176JZF-S a 700 MHz	ARM 1176JZF-S a 700 MHz	Quad-Core ARM Cortex-A7 a 900 Mhz	Quad-Core ARM Cortex-A53 a 1.2 Ghz
GPU	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV
RAM	256 MB	256 MB	512 MB	512 Mb	1GB	1GB
USB 2.0	1	1	2	4	4	4
Video	HDMI 1.4, RCA	HDMI 1.4, TRRS	HDMI 1.4, TRRS	HDMI 1.4, TRRS	HDMI 1.4, TRRS	HDMI 1.4, TRRS
Audio	HDMI, Jack	HDMI, Jack	HDMI, Jack	HDMI, Jack	HDMI, Jack	HDMI, Jack
Boot	SD	microSD	SD	microSD	microSD	microSD
Red	-	-	Ethernet 10/100 Mbit/s	Ethernet 10/100 Mbit/s	Ethernet 10/100 Mbit/s	Ethernet 10/100 Mbit/s, WIFI, BT
Consumo	300mA/5V	400mA/5V	700mA/5V	500mA/5V	800mA/5V	2.5A/5V
Alimentación	MicroUSB/ GPIO	MicroUSB/ GPIO	MicroUSB/ GPIO	MicroUSB/ GPIO	MicroUSB/ GPIO	MicroUSB/ GPIO
Tamaño	85.6 x 53.98 mm	65 x 56 mm	85.6 x 53.98 mm	85 x 56 mm	85 x 56 mm	85 x 56 mm

Tabla 4.5 – *Comparativa Raspberry Pi.*

Raspberry Pi 3 B, por sus características y precio, es la mejor opción del mercado.

En cuanto al sistema operativo para Raspberry Pi, hay diferentes opciones. Para uso general, usando Raspberry Pi como PC o servidor, *Raspbian Debian Jessie* es una opción GNU. Existe una versión más ligera de este Sistema Operativo (SO): Raspbian Jessie Lite.

4.2.4 Servidor WEB

En esta sección se analizan los posibles lenguajes de programación para la implementación del servicio y la mejor alternativa como servidor WEB.

Tal y como se ha mencionado, el sistema de recepción será gestionado por una Raspberry

Pi 3 B conectada junto a la antena de recepción de NOAA. De modo que, Raspberry Pi puede ser usada como servidor WEB, además de gestionar la recepción de imágenes NOAA.

Dado que la aplicación debe estar corriendo en un Servidor bajo el dominio `http://granasat1.ugr.es:8004/` (dominio ofrecido por GranaSAT), el servicio WEB debe encontrarse bajo este Servidor. Esto limitaría la ubicación del sistema de recepción, debido a que la Raspberry Pi usada debe encontrarse dentro de la Local Area Network (LAN). De modo que, siendo así, el sistema de recepción no podría cambiar de ubicación, buscando una mejor cobertura de la señal de los satélites.

Así pues, se plantea ubicar el servicio WEB en un equipo Windows o Linux. En la siguiente Tabla se muestran características de ambos SO que conciernen al alojamiento WEB:

	Servidor Linux	**Servidor Windows**
Coste	GNU	Privado
Servidor WEB	Apache, Nginx	Microsoft IIS
Lenguajes	Perl, PHP, Python, Ruby...	VBScript, ASP.NET...
Bases de datos	MySQL, MariaDB	Microsoft SQL Server, Microsoft Access
Gestor de contenidos	WordPress, Joomla, Drupal, etc.	Exchange, aplicaciones .NET, SharePoint

Tabla 4.6 – *Comparativa de Linux y Windows para alojamiento WEB.*

Queda claro que una de las grandes desventajas es la necesidad de licencias para Windows, por lo que el uso de Servidor Linux es la mejor alternativa para nuestro servicio WEB.

4.2.4.1 Lenguajes para desarrollo WEB

Es difícil establecer un criterio sobre los lenguajes de programación más utilizados. Existe una herramienta conocida como el índice TIOBE que consiste en un informe mensual sobre cuáles son aquellos lenguajes más utilizados y los que disponen de una comunidad de desarrolladores más grande detrás que los soportan y desarrollan. En la siguiente Figura se puede ver los lenguajes más usados. La presencia de lenguajes para el

desarrollo WEB ocupa posiciones destacadas. Java, Javascript, Perl, Python, PHP o Ruby son los lenguajes más utilizados para desarrollo backend.

Es necesario complementar los resultados del índice TIOBE con otras clasificaciones que existen en la red. Estudios de StackOverflow [23] ayudan a tener una visión global más realista, ya que se trata de una red para programadores y es más representativa en los desarrollos online que el índice TIOBE.

Jul 2017	Jul 2016	Change	Programming Language	Ratings	Change
1	1		Java	13.774%	-6.03%
2	2		C	7.321%	-4.92%
3	3		C++	5.576%	-0.73%
4	4		Python	3.543%	-0.62%
5	5		C#	3.518%	-0.40%
6	6		PHP	3.093%	-0.18%
7	8	^	Visual Basic .NET	3.050%	+0.53%
8	7	v	JavaScript	2.606%	-0.04%
9	12	^	Delphi/Object Pascal	2.490%	+0.45%
10	55	^^	Go	2.363%	+2.20%
11	9	v	Perl	2.334%	-0.09%
12	14	^	Swift	2.253%	+0.29%
13	11	v	Ruby	2.249%	+0.13%
14	10	vv	Assembly language	2.240%	-0.04%
15	17	^	R	2.105%	+0.59%
16	13	v	Visual Basic	2.096%	+0.08%
17	16	v	MATLAB	2.009%	+0.45%
18	15	v	Objective-C	1.896%	+0.01%
19	21	^	Scratch	1.813%	+0.73%
20	18	v	PL/SQL	1.545%	+0.09%

Figura 4.13 – *20 lenguajes de programación más utilizados según el índice TIOBE [25].*

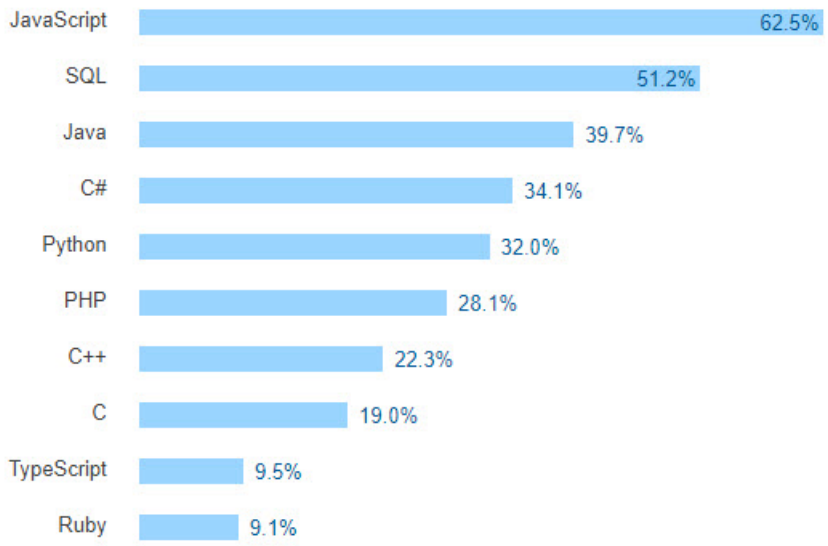


Figura 4.14 – *10 lenguajes de programación más utilizados según StackOverflow [23].*

Dentro de los lenguajes usados, hay que diferenciar aquellos que se usan para la implementación en la parte de servidor y para la interfaz con el cliente.

- **Back-end:**

  Back-end hace referencia al desarrollo WEB del lado del servidor. En cuanto al lenguaje de programación más conveniente y usual (atendiendo a los estudios anteriores), la siguiente Tabla establece una comparativa de estos para tomar la decisión de su elección.

	Java	Javascript	PHP	Python	Ruby
					
Pros	Lenguaje independiente de la plataforma. Gran velocidad. Multitud de librerías estándar (Java API).	Usado en Front-end y Back-end. Node.js usado como servidor WEB ultra rápido, capacidad de abstracción, escalabilidad y eficiencia. Facilidad de comunicación con los navegadores WEB y bases de datos como MongoDB. Es el futuro en desarrollo WEB.	Muy fácil de aprender. Decenas de Frameworks: Symfoly, Laravel, Zend, Cake... Gran demanda laboral. Una comunidad increíblemente grande. Compatible con casi todos los Hostings. Compatible con CMS WordPress.	Fácil de escribir. Rápido. Laboralmente mejor pagado. Fácil de desarrollar. No necesitas un servidor WEB para que funcione. Framwork como Django y Flask. Gestor interno de paquetes instala y borra todos los módulos. Comunidad muy buena.	Fácil uso y lectura de código. Uso de framework orientado a desarrollo WEB llamado Ruby on Rails. Menos competencia laboral.
Contras	Altamente tipado y compilado, haciéndolo poco flexible. Requiere el uso de servidor Tomcat.	Difícil de aprender. Falta de librerías dado a su muy reciente popularidad	Lento, comparado con sus competidores. No genera un bytecode intermedio. Se arregla en versión PHP7, pero no lo suficiente. Más líneas de código para hacer la misma tarea que Python.	Pocos Hostings compatibles. Sin un Framework es muy complejo desarrollar para WEB.	Difícil de aprender. El uso es más restringido y por ello se encontrará menos información en post, foros. Menos ofertas de empleo para Ruby.

Tabla 4.7 – *Comparativa de lenguajes de programación para Back-end.*

Analizando esta Tabla, se descarta en primer lugar el uso de Java y Ruby. Java es poco flexible para desarrollo WEB que se va a realizar, y por su parte, Ruby, pese a ser muy utilizado en desarrollo WEB, es difícil de aprender y está cayendo en desuso. Dado que en el desarrollo de este proyecto se va a aprender un nuevo lenguaje de programación, es de especial interés orientar la decisión atendiendo a la facilidad de implementación, las herramientas disponibles y pensando en un futuro profesional. Así pues, Python parece la mejor alternativa. Python cuenta con una gran comunidad y multitud de librerías estándar (en este punto, Node.js aún está arrancando). Además está ganando en popularidad y las ofertas de empleo ligadas a programadores de Python aumenta. Frente a PHP, es más rápido de escribir y fácil de aprender. Junto a estas ventajas, Python consigue integrar fácilmente librerías escritas en C. En nuestra aplicación, los modelos matemáticos para el cálculo de efemérides, predictores de órbitas satelitales, están escritos en C.

- **Front-end:**

  Cuando se habla de *Front-end* en desarrollo WEB, se está referenciando la interfaz con el usuario. En este caso el el lenguaje usado es interpretado por el navegador.

  Siguiendo con la creciente popularidad de Javascript, este será el lenguaje usado para la interfaz de usuario. Se combinará con el lenguaje de marcado HTML, y el lenguaje de estilo CSS.

  No será necesario el uso de CMS para nuestro desarrollo WEB.

4.3 Plan de evaluación y verificación

Considerando los requerimientos vistos en el Capítulo 2, se desarrolla un protocolo de evaluación del sistema. Llegados a la etapa final del proyecto, el sistema debe ser expuesto a los setup de medidas listados a continuación. Si dichas medidas son verificadas, el sistema se dará por bueno, pero en caso contrario, se deberá analizar las etapas del proceso para encontrar el fallo y, si es necesario, re-implementar.

4.3.1 Evaluación de antena receptora

1. **Medición de antena:** Con ayuda de un analizador de redes, se mide los parámetros de antena S_{11}, impedancia característica y Standing Wave Ratio (SWR), probando que cumple con el diseño y los requerimientos exigidos. Para realizar la medida de la antena, esta debe medirse en condiciones de campo lejano, para que ningún objeto colindante falsee la medida.

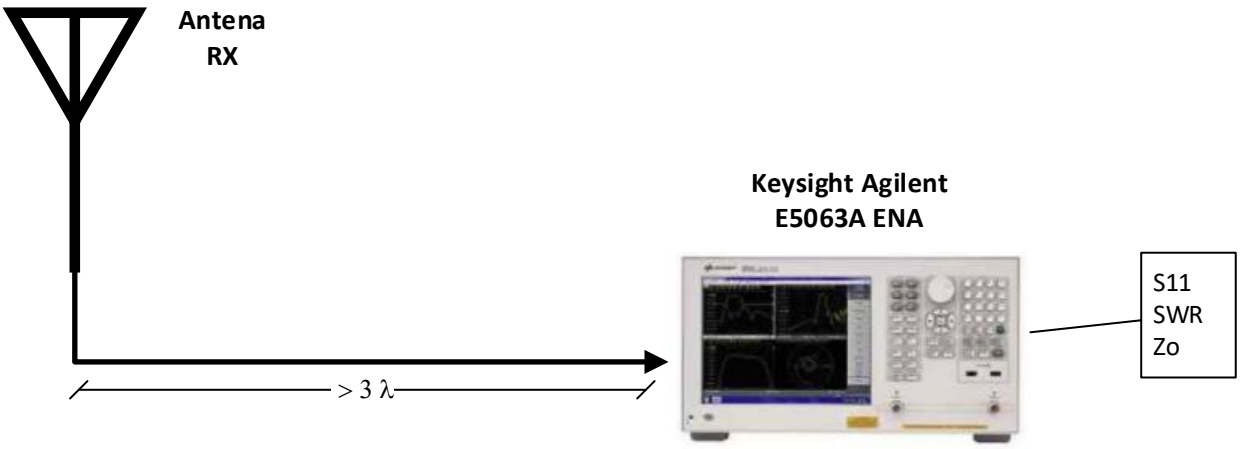


Figura 4.15 – *Setup para medición de antena de recepción.*

Difícilmente se podrá realizar una medición de la ganancia y patrón de radiación de la antena, ya que para ello sería preciso una cámara anecoica, donde arrays de antenas omnidireccionales y un software de medición calculen el patrón.

2. **Recepción de satélite NOAA con antena diseñada:** Empleando un dispositivo SDR conectado a la antena y a un PC mediante su puerto USB, este setup consiste en recibir la señal de los satélites NOAA. El esquema del setup que se debe implementar se puede ver en la Figura 4.16.

   Utilizando Windows como SO, se deben instalar diversos software para el control de SDR y para realizar el seguimiento del satélite. Además, existen software que decodifican la señal APT durante el pase del satélite. Por tanto, se instalará SDRSharp, Orbitron y WXtoImg.

   Además, se probarán otras librerías para el control del dispositivo SDR como RTL_FM y de procesado de audio como Sound eXchange (SOX). Estas librerías son *Open Source*, y se pueden obtener compiladas para Windows.

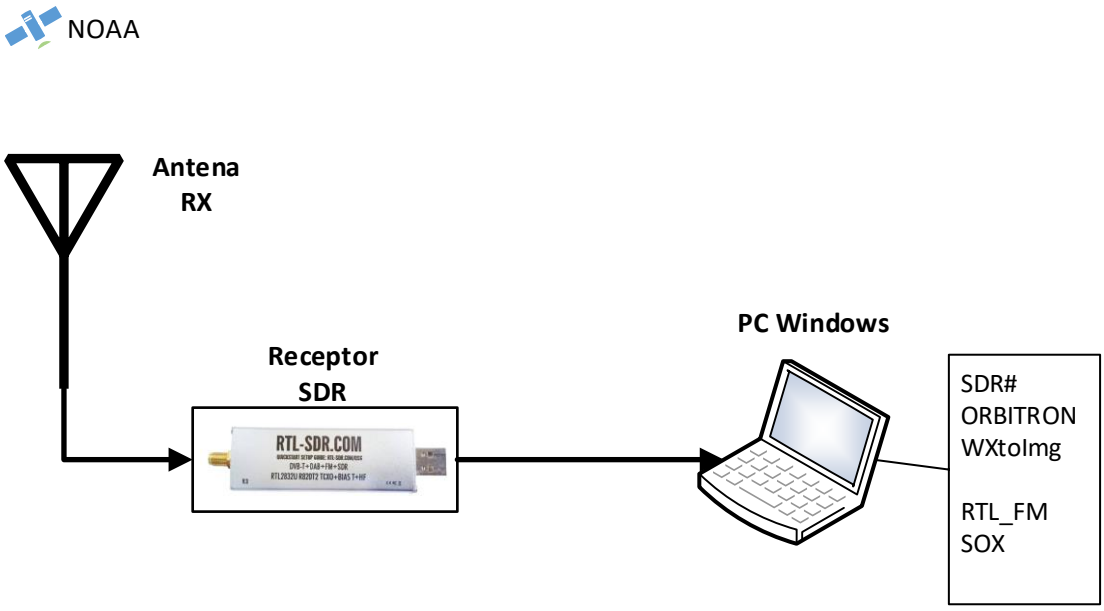


Figura 4.16 – *Setup para recepción de satélites NOAA usando software disponibles para Windows.*

Con este setup de medidas se puede tener una idea del ancho de haz de la antena, comprobando a partir de qué ángulo de elevación del satélite NOAA la antena comienza a recibir señal APT.

4.3.2 Evaluación de receptor SDR

1. **Control de SDR con diversas librerías:** Se estudia el dispositivo con ayuda del software libre SDRSharp. Para ello, se simula un tono con ayuda de analizador de comunicaciones. Es posible implementar un tono igual al APT. La señal APT se transmite sobre una portadora FM modulada por una subportadora AM de $2.4kHz$ y una desviación de $\pm 17kHz$. Esta señal en banda base es subida a la frecuencia RF de downlink de los NOAA.

Las siguientes Figuras muestran los setup a implementar. Dado que la aplicación final va a correr sobre una Raspberry, es importante determinar el comportamiento de las librerías para RTLSDR disponibles.

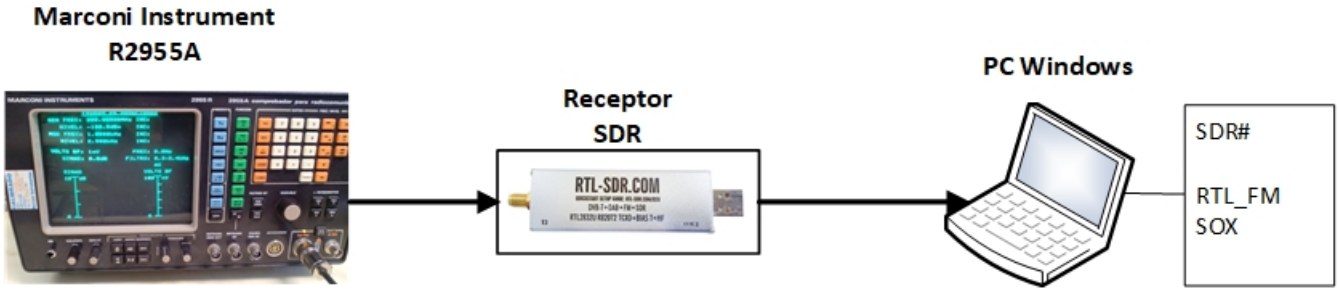


Figura 4.17 – *Setup de pruebas de librerías RTLSDR sobre Windows.*

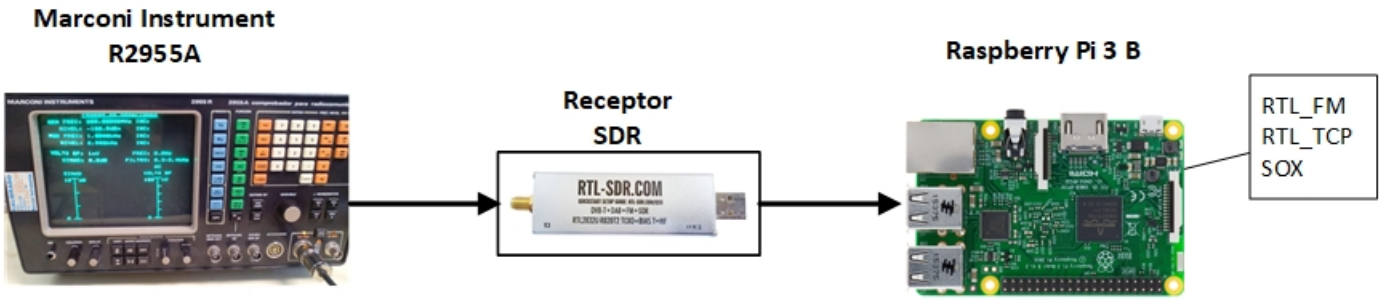


Figura 4.18 – *Setup de pruebas de librerías RTLSDR sobre Raspberry Pi.*

4

2. **Comparativa de diversos receptores SDR:** Los setups mostrados en las Figunas 4.17 y 4.18 serán realizados utilizando diferentes dispositivos SDR disponiblesen el mercado, como son *RTLSDR*, *FunCube Dongle Pro* y *NooElec*. De este modo se comprueba si la señal capturada se ve afectada. Con esta evaluación se busca poder usar el dispositivo más económico del mercado sin que afecte a la calidad de la señal recibida.

4.3.3 Evaluación de predictor de seguimiento de satélites

Existen diversos software que permiten visualizar el seguimiento de satélites. Una vez el servicio WEB ofrezca el *tracking* de los satélites NOAA, esta información puede ser comparada con diversas herramientas existentes, como Gpredict.

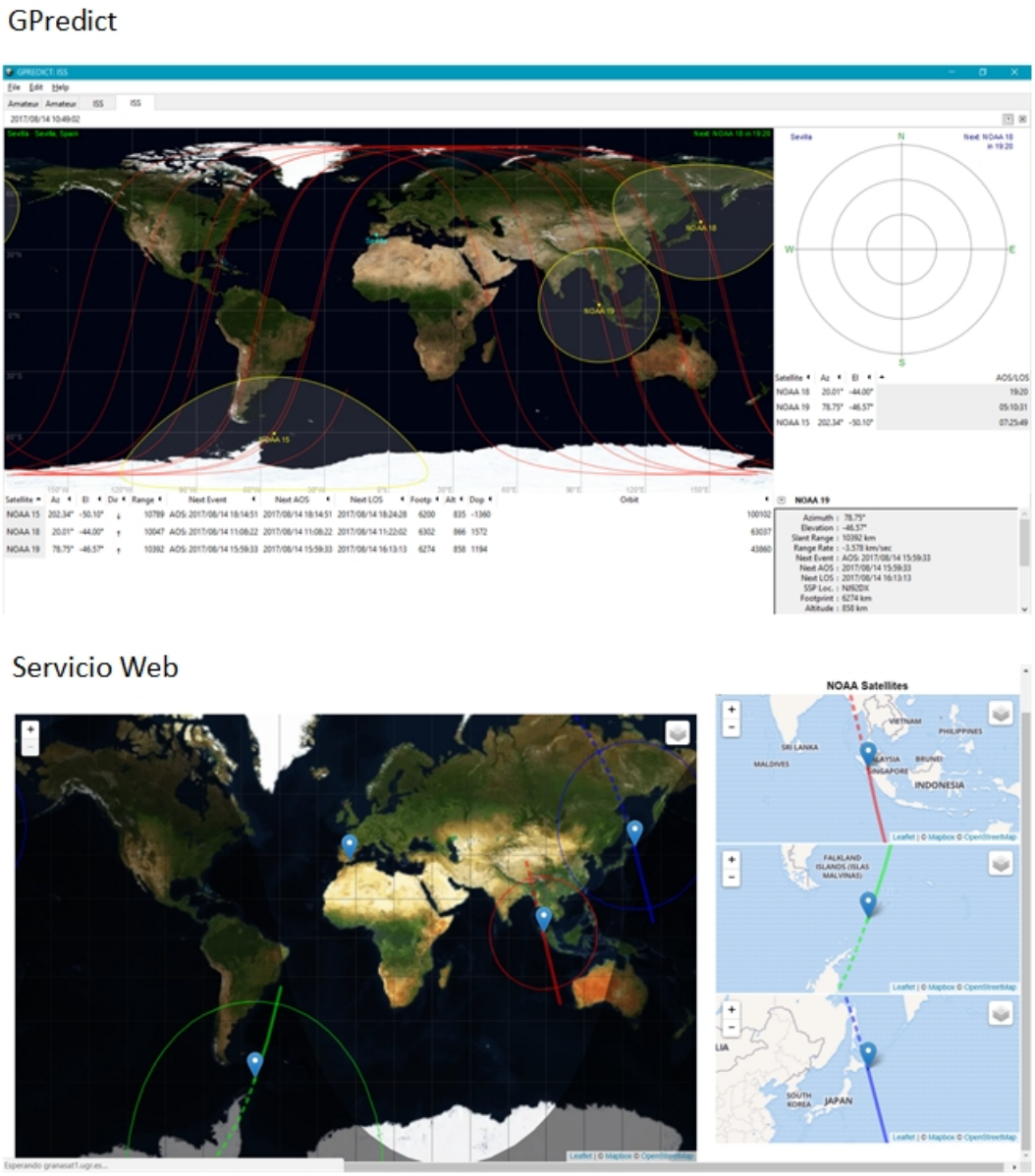


Figura 4.19 – *Comparación de los resultados obtenidos de la posición de satélites en Gpredict y el servicio WEB.*

4.3.4 Evaluación de decodificación de señal APT

Se probará diferentes software de decodificación de la señal APT, prestando especial atención al coste computacional, ya que debe implementarse en tiempo real, una vez el pase del satélite y la descarga de la señal se haya implementado.

Existen diversos software que permiten visualizar el seguimiento de satélites. Una vez el servicio WEB ofrezca el *tracking* de los satélites NOAA, esta información puede ser

comparada con diversas herramientas existentes, como Gpredict.

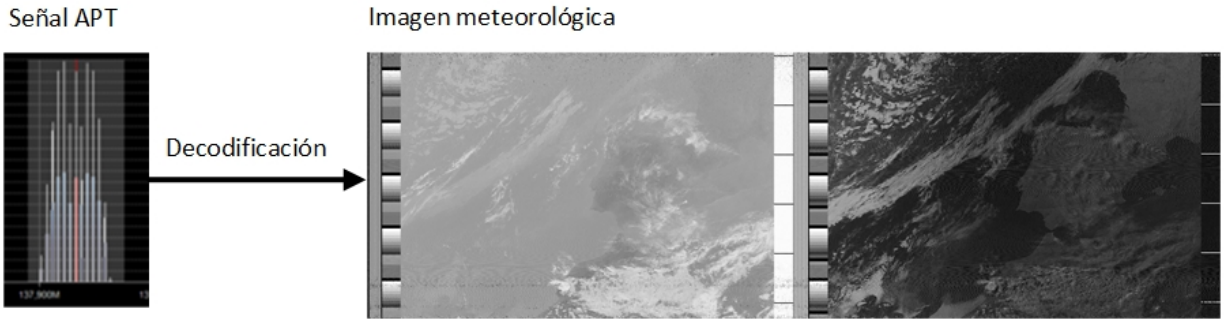


Figura 4.20 – *Decodificación APT.*

Asimismo, las imágenes decodificadas deben almacenarse en una base de datos y estar disponibles para su descarga por el cliente desde la página WEB diseñada. Se comprobará que, efectivamente, el servicio WEB permite este proceso sin ninguna incidencia.

4

4.3.5 Evaluación de sistema completo

Finalmente se evalúa el sistema completo. Para ello se comprueban los resultados obtenidos de los siguientes setups.

1. **Sistema final de recepción:** El sistema consiste en la integración de antena receptra, receptor SDR y Raspberry Pi. En la Raspberry Pi se ejecuta el software de recepción cuando el satélite se sitúa sobre la *ground station*. Mediante comandos *RTL_FM* y SOX se genera un fichero *.wav* con la señal APT demodulada. Este fichero servirá de *input* al decodificador de APT, que se ejecutará igualmente sobre la Raspberry Pi. El *output* del sistema es la imagen meteorológica en formato *.jpg* o *.png* que los sensores de los satélites han capturado en su pase sobre GranaSAT. En la siguiente Figura se muestra el sistema final que será evaluado.

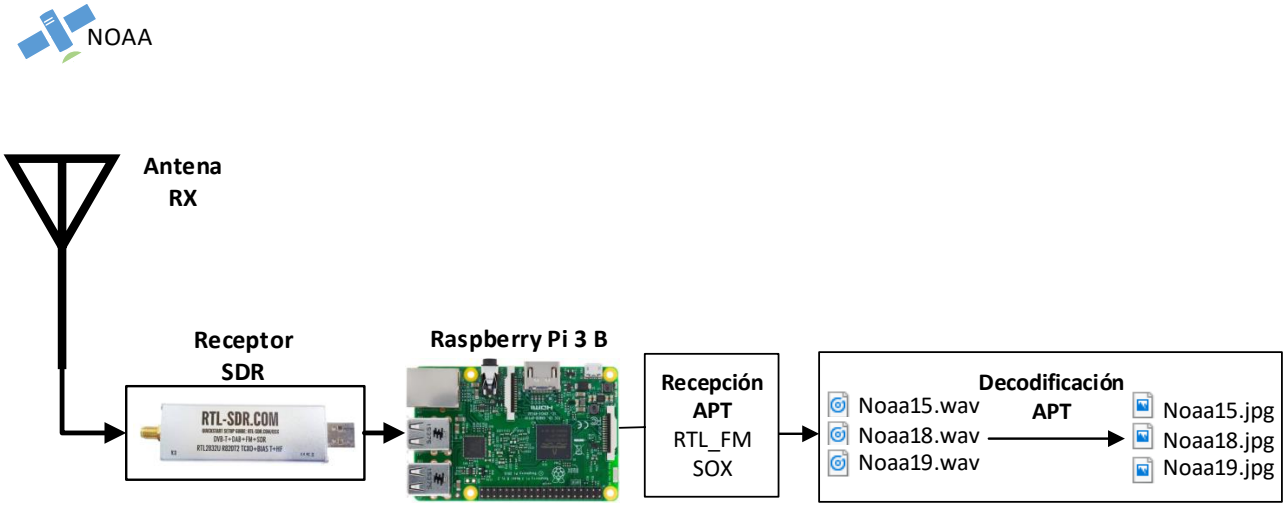


Figura 4.21 – *Sistema receptor final a evaluar.*

2. **Servicio WEB y de descarga de imágenes:** Por otra parte, el objetivo final del sistema es que un usuario tenga acceso a la descarga de las imágenes meteorológicas recogidas en nuestra *ground station*. Se procede a hacer uso de la WEB diseñada, donde, además de permitir visualizar la posición en tiempo real de los satélites NOAA, de obtener información de sus pases futuros y otra información orbital, debe permitir la visualización de imágenes recientes e imágenes que quedan almacenadas en una base de datos dentro del servidor Linux de GranaSAT.

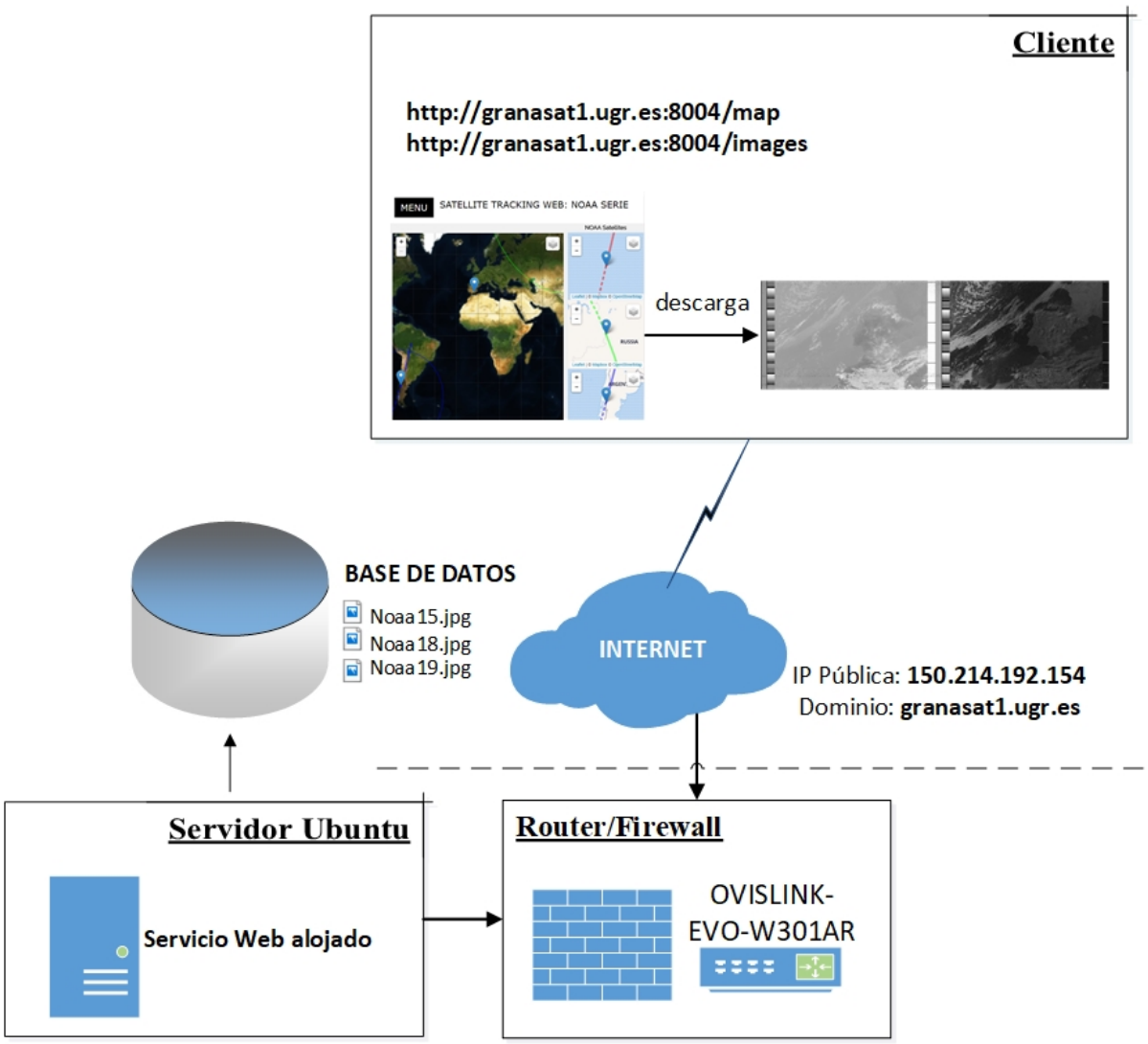


Figura 4.22 – *Servicio WEB de descarga de imágenes a evaluar.*

CAPÍTULO

5

DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

Para la recepción de las señales APT de los satélites NOAA, ya se ha podido comprobar en el Capítulo 4 que no son necesarios equipos demasiado sofisticados o de precio elevado. En el diseño e implementación de este proyecto, la opción más atractiva de receptor SDR es la de un receptor de banda ancha de bajo costo basado en RTL-SDR. El hardware usado es uno de los más baratos del mercado que puede ser encontrado por un precio no superior a los 10€ (como se vio en 4.4). Pese a que existen dispositivos de mejores prestaciones, se pretende un diseño de bajo costo, y a priori, un receptor basado en RTL-SDR es suficiente para la recepción de la señal APT. En la Figura 4.10 se puede ver el kit Realtek DVB-T RTL-SDR disponible en el mercado. El receptor RTL-SDR, usado en nuestro sistema de recepción de datos APT, consta del chip RTL2832U (véase Figura 4.11), un conector USB 2.0 y un conector de antena SMA hembra, pudiendo ser también un conector tipo TV.

Este dispositivo está conectado a la antena receptora que será diseñada y fabricada expresamente para operar en la banda de 137-138 MHz, banda que contiene las frecuencias *downlink* de los satélites NOAA activos (véase Tabla 4.1).

Con ayuda de Raspberry Pi, se controla la recepción de los satélites en tiempo real. Además, se ejecutará un software para la decodificación APT obteniendo la imagen meteorológica. Gracias al reducido tamaño de este ordenador de placa reducida (o Single Board Computer (SBC)), es posible ubicar el sistema de recepción en un punto de buena cobertura sin un gran despliegue de cableado, reduciendo así las pérdidas del cable, ya que

el receptor junto con el PC embebido se pueden colocar junto a la antena.

Las imágenes decodificadas serán almacenadas en una base de datos. El usuario final puede proceder a su descarga a través de la página web creada. Como ya se ha podido ver en la Sección 4.2.4, el servicio web correrá sobre un servidor Linux ubicado en la *gound station* de GranaSAT. La creación del servicio se implementará en Python, haciendo uso del framework Flask. La interfaz de usuario será escrita en Javascript, facilitando la actualización en tiempo real del mapa interactivo que facilita la posición de los satélites al usuario en todo momento.

A continuación se detalla el diseño e implementación de cada una de las partes del sistema.

5.1 Diseño e implementación hardware para recepción

Esta sección se centra básicamente en el diseño de la antena receptora y otros equipos que pueden ser necesarios como LNA o filtros paso banda.

El primer elemento del proceso en la recepción de señal APT de los satélites NOAA es la antena de recepción. Se procede a su diseño atendiendo al análisis de requerimientos de la antena receptora descritos en la sección 4.2.1.

La antena debe seguir las especificaciones dadas por el downlink de los satélites (mostrados en la Tabla 4.1) y considerando la necesidad de filtrar señales interferentes en la banda de frecuencia de recepción. Por lo tanto, se pretende diseñar una antena con un *bandwidth* estrecho, de 137 a 138 MHz (un ancho de banda de 2 MHz).

Para la recepción de datos procedentes de satélites se requiere una antena RHCP, con un *beamwith* (ancho de haz) prácticamente omnidireccional en el plano horizontal, para cubrir todo el pase de los satélites NOAA desde que sea visible por el horizonte AoS hasta que deje de serlo LoS, tal y como se vio en la Figura 4.5. Como la antena de recepción no irá colocada sobre unos rotores para el seguimiento del satélite, esta no debe tener gran directividad.

Debido a la mayor facilidad en el diseño y fabricación de las antenas Turnstile y Eggbeater y al interés en ellas en GranaSAT, se ha decidido proceder a su fabricación. Se ha obtado por fabricar dos antenas diferentes y así determinar cuál es la mejor opción para integrar en el sistema.

5.1.1 Antena Turnstile

Es la antena más fácil de fabricar de las propuestas. Consiste en dos dipolos de longitud $\lambda/2$ cruzados para tener RHCP. Al tratarse de dipolos, presenta un diagrama de radiación omnidireccional, sin embargo, su *beamwidth* no es óptimo para pases de baja elevación y no tiene una buena ganancia, aunque se puede mejorar. La recepción se hace complicada debido a las pérdidas por espacio libre y al fading [1].

[1]Denominado también desvanecimiento de la señal, que representa variaciones temporales en el nivel de la señal y son causados principalmente por variaciones atmosféricas. Es difícil de predecir. Dentro

5.1.1.1 Diseño y fabricación de antena Turnstile

A continuación se describen los pasos seguidos para su fabricación en la banda VHF, en concreto nos interesa la banda de frecuencias de 137 a 138 MHz, cubriendo así las frecuencias downlink de la señal APT de los satélites NOAA activos (NOAA15, NOAA18 y NOAA19).

Los materiales requeridos son los siguientes:

- Barras de aluminio para dipolos.

- Varillas de acero para reflectores.

- Tubo PVC de 40 mm de diámetro (2 m).

- Manguito de 75 mm de diámetro.

- Tapón de PVC de 75 mm.

- Adaptador PVC de 75 mm a 50 mm.

- Adaptador de PVC de 50 mm a 40 mm.

- Pieza aislante para fijar los dipolos.

- Coaxial de 75Ω y 50Ω.

- Conector de TV para SDR o tipo N para medición en analizador de redes.

Seguidamente se detalla las dimensiones de cada uno de los elementos integrantes de la antena y el modo de integración para montar la antena deseada.

a) **Dipolos:** La longitud del dipolo resonante es próxima a $\lambda/2$, algo menor porque la velocidad de propagación por el cobre o aluminio es menor a la del espacio libre.

$$\lambda = \frac{c}{f} \tag{5.1.1}$$

$$L = 0.95\lambda/2 = 0.95c/2F = 142.5/F(\text{MHz})[m]L = (142.5)/(137.5) = 1.036[\text{m}] \tag{5.1.2}$$

De modo que, cada tramo del dipolo será de $51.8cm$.

de los principales tipos de desvanecimientos encontramos desvanecimiento por refracción, por difracción, desvanecimiento por desenfoque de haz, desvanecimiento por múltiple trayecto (multipath). Dentro de las clases de desvanecimiento también encontramos una muy importante denominada desvanecimiento plano. El desvanecimiento causado por múltiple trayectoria es originado cuando dos o más señales han recorrido diferentes trayectos para llegar al receptor, con lo cual se pueden sumar o cancelar en una frecuencia específica dependiendo de la relación de sus fases. En enlaces de baja capacidad este tipo de desvanecimiento origina siempre una reducción del nivel de la señal en el receptor, causado por el ruido térmico, catalogado dentro de desvanecimiento plano. En los sistemas de banda ancha no solo causa distorsión en el nivel de la amplitud de la señal sino que también causan distorsión en banda, a lo que se llama desvanecimiento selectivo.

Los dipolos serán algo más cortos por la separación que habrá entre ellos para el conexionado. Dependiendo de la base de anclaje usada para fijar los dipolos, se sabrá la distancia entre estos. Los dos dipolos forman 90º entre sí. En la Figura 5.1 se puede ver la pieza de metacrilato usada para anclar cada uno de los tramos de los dos dipolos.

(a) **(b)**

Figura 5.1 – *a) Anclaje de dipolos en estructura de sujeción. b) Pieza de metacrilato para sujeción de dipolos.*

Para aislar las conexiones para alimentación de los dipolos (explicadas a continuación) y dar una mayor rigidez a los dipolos, se ha usado un manguito de PVC para tubos de 75 mm (como se puede ver en la Figura 5.1), una transición de PVC de 75 mm a 50 mm para acoplar posteriormente al mástil y un tapón de 75 mm, permitiendo hacer una especie de caja que da solidez a la estructura (véase Figura 5.2).

b) **Reflectores:** Es común usar dos varillas metálicas como reflectores en lugar de una malla. Se colocarán dos varillas en forma de cruz, formando 90º entre sí. La distancia entre dipolos y reflector varía el ancho de haz: $d = [0.22\lambda - 0.45\lambda]$. Cuanto mayor separación, mayor ancho de haz reduciendo la directividad en cenit [2] (beneficiando a pases bajos), mientras que menor separación beneficia a los pases altos. Se ha elegido un valor intermedio:$d = 0.34\lambda = 0.34 * 2.18 = 0.745$ m. La longitud de reflectores viene dada por: $L = \lambda/2 = \frac{300}{2F(MHz)} = 109$ cm.

En la Figura 5.3 se puede ver la colocación de los reflectores en el mástil, hecho con un tubo de PVC de 40 mm de diámetro y 2 m de longitud.

[2]Punto del hemisferio celéste situado sobre la vertical del observador.

(a) **(b)**

Figura 5.2 – *Estructura con dipolos de la antena: a) parte superior, b) parte trasera de la estructura que irá acoplada al mástil con una transición de PVC de 50 mm a 40 mm.*

Figura 5.3 – *Reflectores para antena Turnstile.*

c) **Alimentación:** El dipolo tiene una impedancia característica de 73 Ω [26], que para facilitar los cálculos se considera de 75 Ω. Es necesario conocer la impedancia de antena para la adaptación de impedancias con el cableado y equipos. Se suele adaptar a 50 Ω por compatibilidad con equipos de comunicaciones. Se deben conectorizar los dipolos para que haya polarización circular y además, uno de los dipolos debe estar desfasado 90º con respecto al otro, para lo cual se utiliza un desfasador $\lambda/4$. Para interconexión de dipolos, tomamos de referencia cada elemento según el cuadrante de un reloj: 3, 12, 9 y 6 (ver Figura 5.4). Es importante el modo de conexión ya que obtendremos polarización circular a derechas (RHCP) o a izquierdas (LHCP) según el sentido que tomemos en el conexionado.

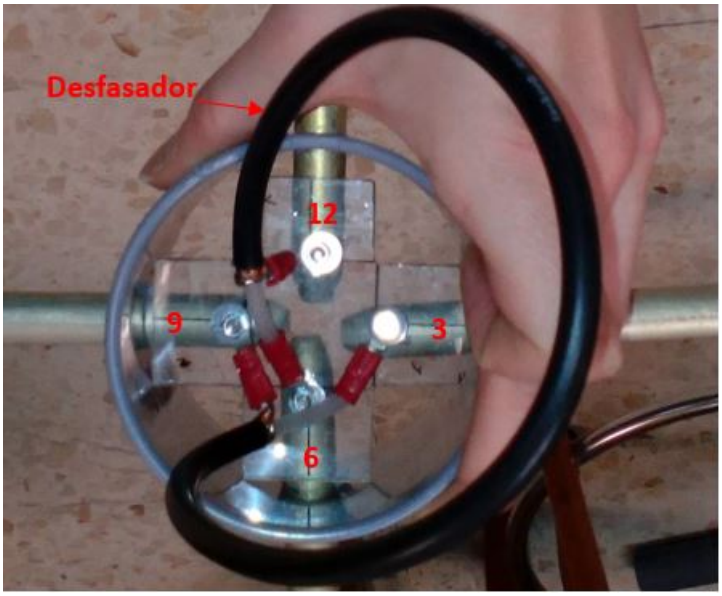


Figura 5.4 – *Desfasador* $\lambda/4$.

- Desfasador $\lambda/4$ con coaxial de 75 Ω: $Longitud = l_\lambda/4 = \lambda/4 * f_v = 218$ cm$/4 * 0.666 = 36.3$ cm Es necesario comprobar el factor de velocidad (f_v) del cable de 75Ω (varía según RG-59). La conexión para conseguir RHCP está representada en la Figura 5.4, donde el vivo de un extremo del tramo de coaxial va a 3 y la malla a 9. El vivo del otro extremo del coaxial va a 12 y la malla a 6. A la salida de ambos dipolos se ve una impedancia de: $Z_L \simeq 75\ \Omega || 75\ \Omega \simeq 37.5\ \Omega$

A continuación se dan varias alternativas para alimentar los dipolos:

- Opción 1 de alimentación: Se puede alimentar con coaxial de 50 Ω (RG-58) directamente. El vivo del coaxial irá a punto 3 y la malla a punto 9. Habría desadaptación: $ROE = 50/37.5 = 1.3$. Esta desadaptación es factible. Es la opción usada en este caso.
- Opción 2 de alimentación: Para alimentar finalmente con coaxial de 75 Ω (RG-59). Es necesario una Q-section de $\lambda/4$. El valor de la impedancia de esta Q-section debería ser: $Z_Q = \sqrt{75 * 37.5} = 53.03\ \Omega$. De modo que se usaría un tramo $\lambda/4$ de coaxial de 50 Ω. La ROE resultante sería similar a la opción anterior.

d) **Simetrización:** Es necesario simetrizar la antena ya que el coaxial es no balanceado y es posible que los dipolos no sean alimentados con la misma intensidad por ambos brazos, resultando un patrón de radiación asimétrico. La forma de simetrizar es mediante la introducción de ferritas en el cable de alimentación o con un balun $\lambda/2$ fabricado con un tramo de coaxial. Con el balún $\lambda/2$ hay que tener en cuenta que la relación de adaptación de impedancias es 4:1, por lo que habría que volver a adaptar. En este caso se ha optado por introducir ferritas (véase Figura 5.5)

Finalmente, se introduce el adaptador de PVC para fijar la estructura de los dipolos al mástil de PVC y el tapón para proteger el cableado. El resultado se vio en la Figura 5.2.

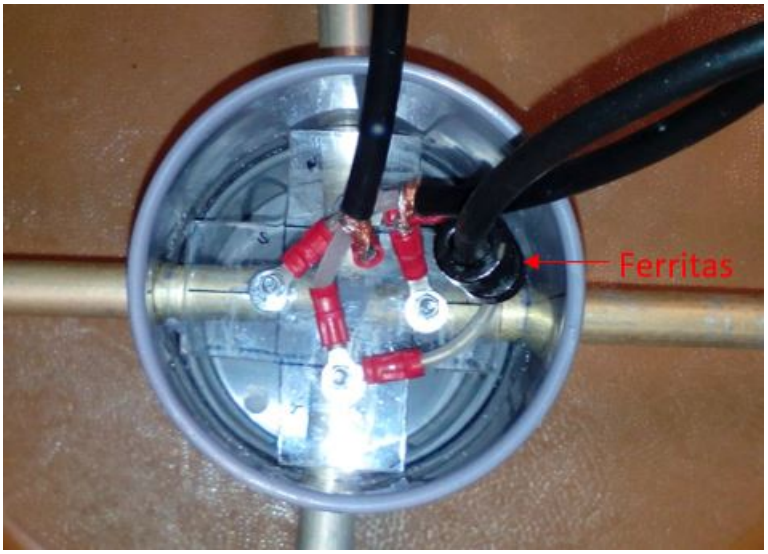


Figura 5.5 – *Inserción de ferritas en la línea de alimentación.*

En la Figura 5.6 se puede ver la antena Turnstile fabricada.

Figura 5.6 – *Antena turnstile fabricada para recepción de señal APT*

5.1.2 Antena Eggbeater

Esta antena también cumple con los requisitos para recepción de señales APT, ya que presenta un diagrama de radiación omnidireccional y es de polarización circular. Consiste en dos loops desfasados 90º hechos de un material radiante, como un cable de cobre. En comparación con turnstile, esta antena presenta las siguientes ventajas:

- Favorece a la recepción en pases de satélite de menor elevación, pero es algo peor en ángulos de 90º, sobre el cenit del satélite.
- El ruido producido por un loop es menor que el producido por un dipolo, aumentando así la relación señal ruido (SNR).
- El loop presenta una ganancia de 1.25 dBd (sobre la ganancia del dipolo).
- Es más compacta que la turnstile, ya que el reflector de la antena está a una distancia de 1/8 de longitud de onda del elemento radiante, mientras que en la tunestile está a 1/4.

Por otro lado, comparándola con QHA, ambas presentan características similares aunque QHA presenta un mejor diagrama de radiación, pero mayor dificultad en la fabricación.

Finalmente, se decidió diseñar y fabricar una antena Eggbeater (véase Figura 5.7) adecuada a la banda de operación de los satélites NOAA y optimizarla para obtener el patrón de radiación deseado y buena adaptación en la banda de interés. Además, se desea que su ancho de banda cubra las frecuencias de downlink de los satélites NOAA, procurando filtrar aquellas frecuencias interferentes contiguas. A continuación se analiza en detalle su diseño.

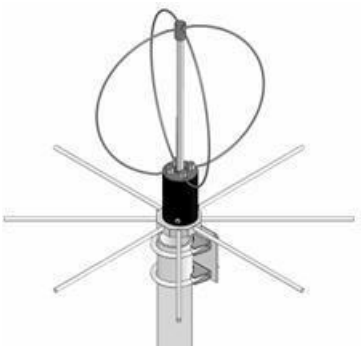


Figura 5.7 – *Antena Eggbeater.*

5.1.2.1 Diseño electromagnético de Eggbeater

Nuestra antena será diseñada para recibir a las frecuencias de los tres satélites NOAA activos (4.1), que consta de la parte activa, que consiste en dos loops desfasados 90º, y los

elementos reflectores que forman el plano de masa.

Para conocer más acerca de antenas de tipo loop, se ha realizado un estudio del arte de su comportamiento electromagnético. En [26], se realiza un estudio teórico de estos elementos radiantes. Las antenas de tipo lazo se pueden clasificar en antenas electromagnéticamente pequeñas y electromagnéticamente grandes. Las antenas electromagnéticamente pequeñas son aquellas cuya circunferencia (su perímetro) es normalmente menor a 0.1λ, mientras que las antenas electomagnéticamente grandes son aquellas cuyo perímetro supera la longitud de onda $C >> \lambda$. Son antenas muy utilizadas en aplicaciones de banda High Frequency (HF), VHF y Ultra High Frequency (UHF), aunque su uso puede llegar a frecuencias de microondas [34].

Se ha demostrado que el patrón de radiación de una antena de tipo lazo, cuya longitud es $C \approx \lambda$ (o cuyo radio es $a \approx 0.1\lambda$), se asemeja al de un dipolo $\lambda/2$ (Figura 5.8), presentando máxima directividad en 0^o y 180^o. Mientras, a medida que aumenta el radio, van apareciendo más lóbulos secundarios, por lo que es algo a evitar.

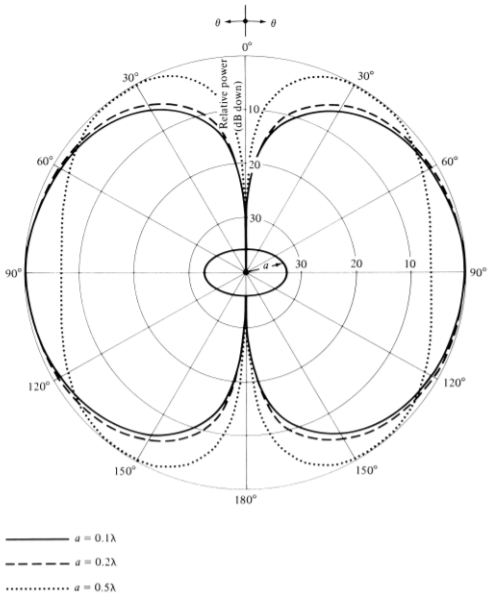


Figura 5.8 – *Diagramas de radiación en plano vertical para un loop circular de corriente constante (a=*0.1λ*,* 0.2λ*, and* 0.5λ*)[26].*

El comportamiento electromagnético depende de de la longitud del elemento radiante y de la corriente a través de él. Generalmente, el modelo matemático para una distribución de corriente constante es para loops pequeños, sin embargo puede ser utilizado para loops de mayor tamaño para la zona de campo lejano (r » a). Siguiendo el modelo de distribución de corriente uniforme, se puede aproximar teóricamente la impedancia de antena y la directividad de esta. En la siguientes gráficas (5.9) se muestra como varía impedancia de radiación y directividad en función a la longitud del lazo circular. Se observa que a partir de que se empieza a considerar electromagnéticamente grande, la

curva cambia de pendiente.

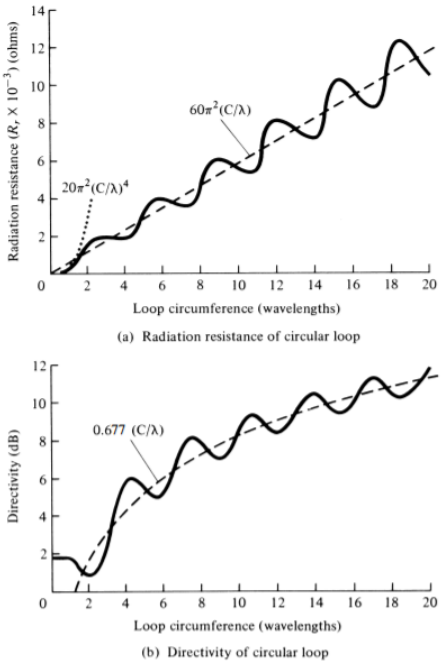


Figura 5.9 – *Resistencia de radiación y directividad para un loop circular de corriente constante.[26]*

5

El perímetro de los loops será de una longitud de onda de trabajo (λ), por lo que la resistencia de radiación será de aproximadamente 200Ω y la directividad de $0.677dBi$. Estas antenas tienen una directividad pobre, por lo que no suelen ser utilizadas como transmisoras, sino como receptoras, donde lo importante es conseguir buena relación señal a ruido.

Los reflectores de la antena permite mejorar la directividad. El factor que influye en el aumento de esta es la distancia de los loops al plano de tierra. Como no se puede proponer un plano de tierra infinito, se propone que los reflectores sean ocho varillas metálicas de longitud $\lambda/4$ en lugar de un plano de metal, así antena será más estable mecánicamente. Mientras, la distancia entre el plano de tierra y los loops será de $\lambda/8$, minimizando el tamaño dela antena, y consiguiendo mejorar la directividad tal y como refleja la siguiente gráfica (5.10), donde se considera un plano de tierra infinito.

Según las especificaciones comentadas, la ecuación 5.1.1 y que la frecuencia central de trabajo es de 137.51 MHz, las dimensiones teóricas de los elementos de la antena diseñada son las siguientes:

- Perímetro de los loops: 2181.7 mm
- Longitud de los radiales: 545.4 mm
- Distancia entre loops y radiales: 272.7 mm

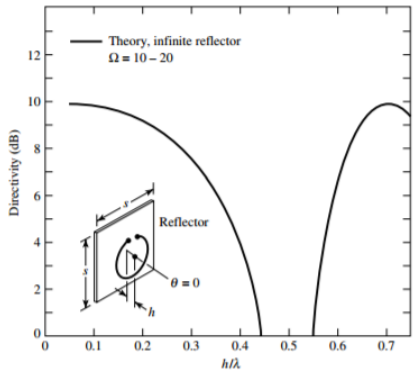


Figura 5.10 – *Directividad de antena de lazo circular, para ángulo de máxima radiación según distancia desde el reflector h/λ. Curva teórica para un reflector de plano infinito.[26]*

A partir de estos resultados se comienza el diseño de la antena receptora. Estas dimensiones se optimizan para unos objetivos concretos para la aplicación.

Para su simulación se han usado software de simulación electromagnética de diseños 3D. A continuación se definen los programas empleados en esta tarea:

- **4nec2:** Es un software libre para Windows basado en herramientas para la creación, visualización y optimización de estructuras geométricas de antenas en 2D y 3D. Genera y compara patrones de radiación en campo lejano y campo cercano permitiendo modelar antenas y representa gráficas de la ganancia y ROE de la antena. Sus funcionalidades son limitadas en comparación con otros software de modelado electromagnético 3D. En [10] puede descargarse la última versión de este software y acceder a un manual de usuario.

- **CST STUDIO SUITE:** Software de simulación electromagnética para diseños electromagnéticos. Las herramientas de CST permiten el diseño y optimización de dispositivos operando en un amplio rango de frecuencias. Sus análisis pueden incluir efectos térmicos y mecánicos, además de simulación de circuitos. Es una herramienta altamente fiable como paso previo a pruebas físicas. CST STUDIO SUITE consta de diversos módulos: CST Microwave Studio, CST EM Studio, CST Particle Studio, CST Cable Studio, CST PCB Studio, CST Mphysic Studio, CST Design Studio.

  En este caso, la herramienta usada para el diseño y optimización de la antena es CST Microwave Studio (CST MWS). Es la herramienta líder para la rápida y rigurosa simulación de dispositivos de alta frecuencia y ofrece simulaciones en el Dominio del Tiempo (Time Domain Solver) y en el Dominio de la Frecuencia (Frequency Domain Solver). Permite el análisis de antenas, filtros, acopladores, estructuras multicapa (como tecnología microstrip), etc. Es posible la importación de estructuras complejas 3D y 2D desarrolladas en otros software de diseño como AutoCAD, CATIA, Solidwork, etc.

  En [16] se puede encontrar documentación y tutoriales de este programa de simulación. CST no es software libre como 4NEC2, pero es de los más usados por sectores de

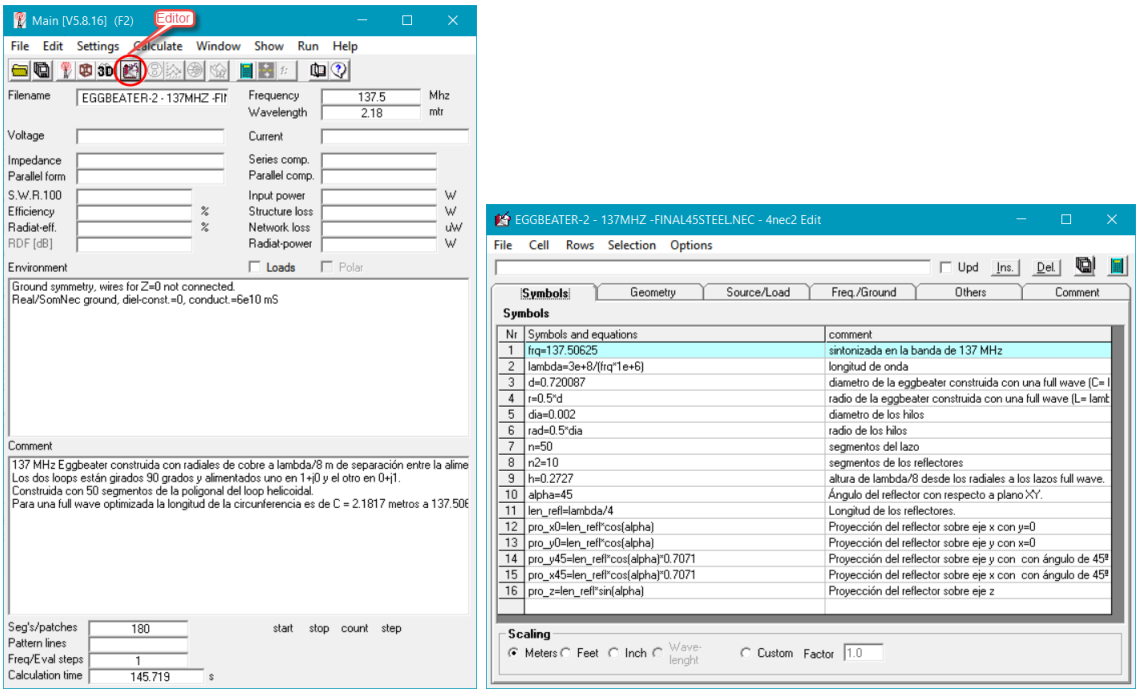
ingeniería.

Mediante estos software se han replicado las especificaciones teóricas mencionadas anteriormente y se ha procedido a optimización de parámetros.

5.1.2.1.1 Simulación con 4NEC2

Este software libre es ideal para simulación de estructuras sencillas, ya que su funcionalidad es algo limitada. Para el caso que nos ocupa, 4NEC2 puede dar como resultado una aproximación rápida al diseño de la antena Eggbeater.

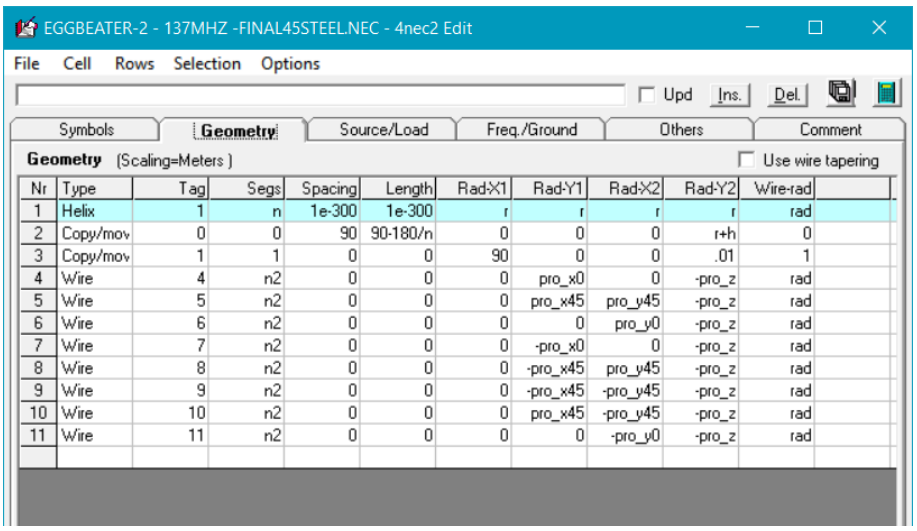
En la Figura 5.11 puede verse la interfaz de trabajo de 4NEC2. Pulsando *'Edit NEC-input file'* se accede a la ventana de diseño de la geometría del elemento radiante, vista en Figura 5.11.b). De este modo, es posible parametrizar la estructura para facilitar sucesivos cambios o acceder al optimizador de 4NEC2. Entre las variables declaradas se encuentra la frecuencia central de la banda de operación, el diámetro de los loops, la longitud de los reflectores, la distancia de estos a los loops y su posición. La posición de los reflectores se ha declarado de forma vectorial, atendiendo a la proyección en el eje x ,y, z. Además, cabe destacar que se ha declarado una variable denominada *alpha* que corresponde al ángulo de inclinación de los reflectores.



(a) **(b)**

Figura 5.11 – *a) Ventana principal de 4NEC2. b) Ventana de edición de estructuras 3D.*

Un punto importante a considerar es que 4NEC2 necesita que se defina el número de segmentos mínimo por cada media longitud de onda de cada elemento. Un mayor número de segmentos implica una mejor consideración de efectos electromagnéticos en zonas críticas. 4NEC2 posee una herramienta de auto-segmentación con opciones de baja, media y alta selección de mínimo número de segmento por media longitud de onda (10, 25 y 100 segmentos). Es recomendable seleccionar esta funcionalidad en la optimización de la antena, donde longitudes y frecuencias varían, adaptándose el número de segmentos a estos cambios. En este caso, se ha declarado el número de segmentos mínimo a considerar en los parámetros de diseño, ya que no hay ninguna zona crítica en la geometría de la antena. En la Figura 5.12 se puede ver la definición geométrica de los diferentes elementos de la Eggbeater, tomando las variables de entrada declaradas.



Nr	Type	Tag	Segs	Spacing	Length	Rad-X1	Rad-Y1	Rad-X2	Rad-Y2	Wire-rad
1	Helix	1	n	1e-300	1e-300	r	r	r	r	rad
2	Copy/mov	0	0	90	90-180/n	0	0	0	r+h	0
3	Copy/mov	1	1	0	0	90	0	0	.01	1
4	Wire	4	n2	0	0	0	pro_x0	0	-pro_z	rad
5	Wire	5	n2	0	0	0	pro_x45	pro_y45	-pro_z	rad
6	Wire	6	n2	0	0	0	0	pro_y0	-pro_z	rad
7	Wire	7	n2	0	0	0	-pro_x0	0	-pro_z	rad
8	Wire	8	n2	0	0	0	-pro_x45	pro_y45	-pro_z	rad
9	Wire	9	n2	0	0	0	-pro_x45	-pro_y45	-pro_z	rad
10	Wire	10	n2	0	0	0	pro_x45	-pro_y45	-pro_z	rad
11	Wire	11	n2	0	0	0	0	-pro_y0	-pro_z	rad

Figura 5.12 – *Geometría de antena en 4NEC2.*

Así pues, se procede a la simulación de los parámetros de antena que son de interés. Para ello, hay que considerar la alimentación de los loops. Como ya se ha comentado, se tratará de una antena de polarización circular, por lo que cada uno de los loops estará alimentado con una fuente de igual amplitud y un desfase de 90º de un lazo con respecto a otro. En la Figura 5.13 se puede ver cómo se introduce en 4NEC2 la alimentación de ambos lazos.

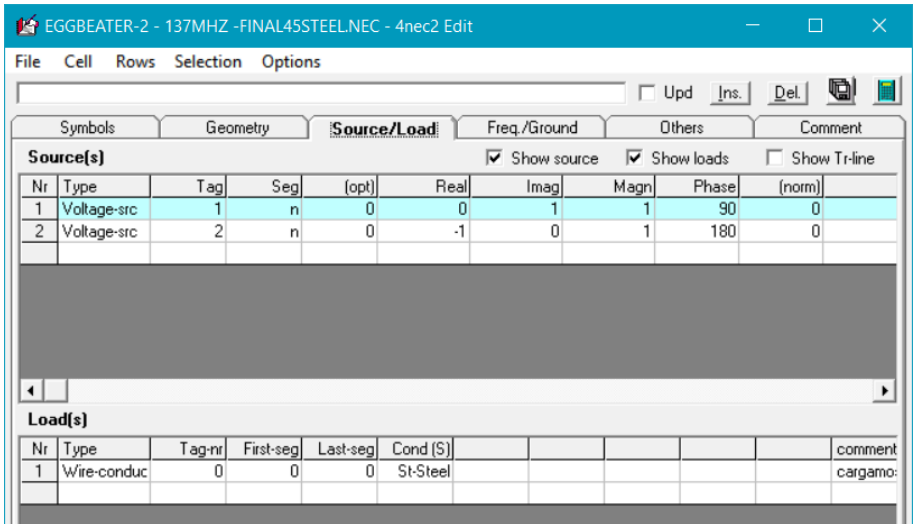


Figura 5.13 – *Alimentación de loops en antena en 4NEC2.*

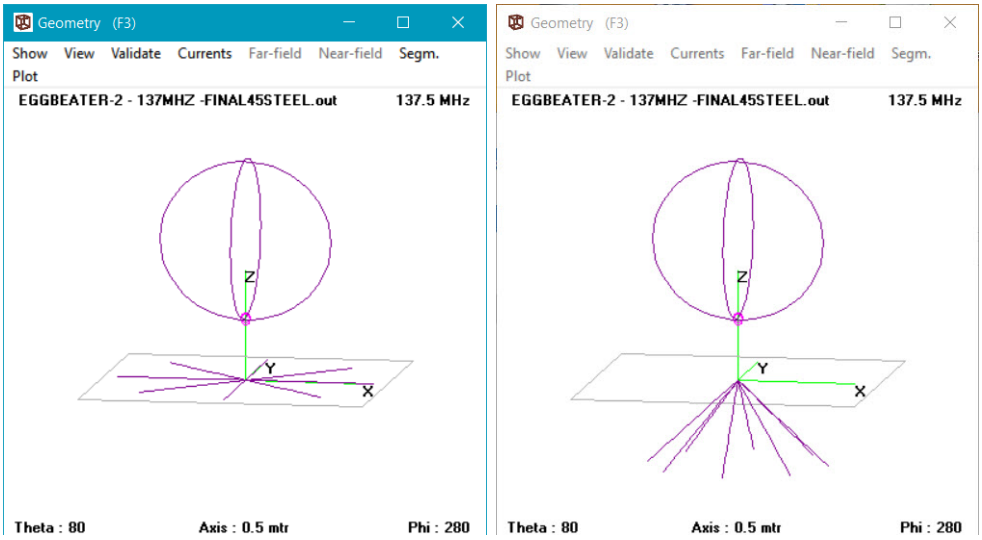
Se ha considerado dar una inclinación a los reflectores, de modo que se consiga ampliar el ancho de haz de la antena Eggbeater. Con un ancho de haz mayor, se podrá optimizar la recepción de datos de satélites en pases de baja elevación. Los reflectores actúan reflejando la potencia radiada y son el elemento que permite mejorar la directividad de la antena. Si se da cierta inclinación a los reflectores, la potencia radiada perderá en directividad pero la antena tendrá un mayor ancho de haz. El ancho del lóbulo principal, se determina como la caída a 3 dB de la potencia máxima radiada del elemento radiante (o caída a la mitad de potencia), denominado por sus siglas en ingles HPBW.

En 4NEC2 se ha realizado una simulación con el cambio propuesto en los reflectores pero no se ha observado grandes cambios en la simulación del patrón de radiación. En la Figura 5.14 se muestra estos resultados. Debido a que 4NEC2 no da una buena fiabilidad, se ha hecho uso del programa CST, que es mucho más profesional y da mayor funcionalidad.

Atendiendo a la Figura 5.14.d), se observa que HPBW de la antena es de aproximadamente 130º, de modo que los pases de elevación muy baja no se obtendría demasiado buena SNR, dada la baja ganancia.

Además de esto, se debe prestar especial interés a la impedancia de radiación de la antena y a su ROE. Como se aprecia en la Figura 5.15, tras la simulación se obtienen una serie de outputs, entre ellos, la impedancia de antena. A la frecuencia de operación, $Z_a = (155 + 0.27j)\Omega$. Esta impedancia es tomada desde uno de los puertos de entrada, es decir, que no considera el paralelo de ambos elementos radiantes desfasados 90º, sino, únicamente uno de ellos. Aún así, la impedancia del loop está afectada por la radiación del loop colindante.

En cuanto al coeficiente de reflexión de la antena, parámetro S_{11}, viene dado por la siguiente ecuación:



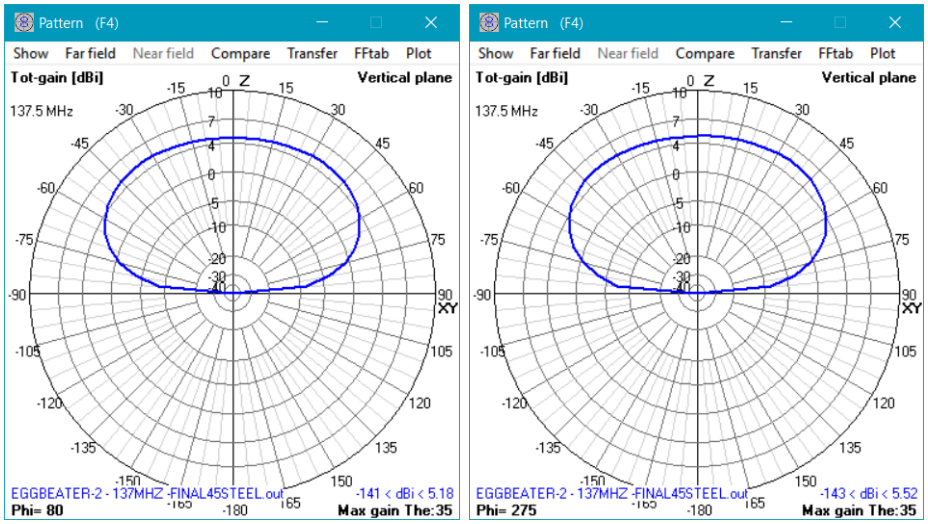


Figura 5.14 – *a) Antena Eggbeater con reflectores con alpha=0º. b) Antena Eggbeater con reflectores con alpha=45º. c)Patrón de radiación en el plano vertical para antena con reflectores con alpha=0º. d) Patrón de radiación en el plano vertical para antena con reflectores con alpha=45º.*

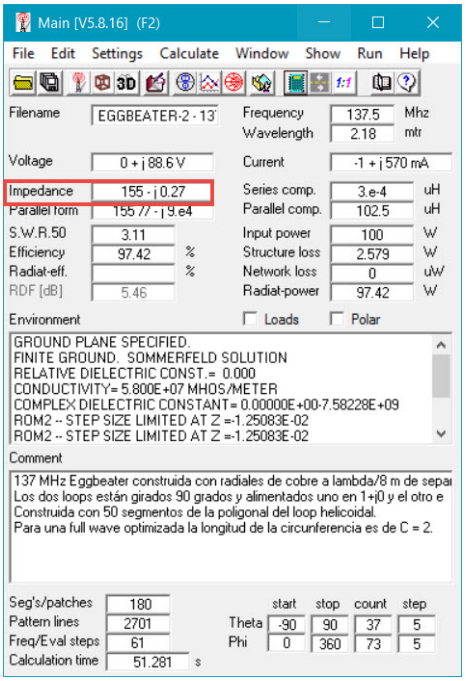


Figura 5.15 – *Parámetros de salid de 4NEC2.*

$$\Gamma = \frac{Z_L - Z_0}{Z_L + Z_0} \tag{5.1.3}$$

donde Z_0 es la impedancia característica de la línea y Z_L es la impedancia de la antena, en este caso. La impedancia característica es 50Ω por adaptación con los equipos de RF y con cableado coaxial de 50Ω. De modo que, atendiendo a esto, a priori se sabe que la antena no estará adaptada, y se puede ver en la Figura 5.16. Igualmente, esto queda reflejado observando la razón de onda estacionaria ROE (o SWR por sus siglas en inglés). En una línea de transmisión coexisten onda incidente y onda reflejada. Cuando la onda incidente y reflejada presentan una interferencia constructiva, la amplitud de la onda resultante viene dada por $V_{max} = V_{incidente} + V_{reflejada}$, mientras que si las ondas son destructivas, se tiene $V_{min} = V_{incidente} - V_{reflejada}$, de modo que SWR se define como:

$$SWR = \frac{V_{max}}{V_{min}} = \frac{1 + |\Gamma|}{1 - |\Gamma|} \tag{5.1.4}$$

Como se puede ver en la Figura 5.16, la razón de onda estacionaria es de 3.11, por lo que más del 25% de la potencia se refleja. Habrá que trabajar en la adaptación de impedancias para mejorar el coeficiente de reflexión (S_{11}).

Finalmente, la Figura 5.17 muestra el patrón de radiación donde se puede ver que la ganancia de antena máxima es de 5.52 dBi en el punto de máxima directividad, tomando como referencia una antena isotrópica.

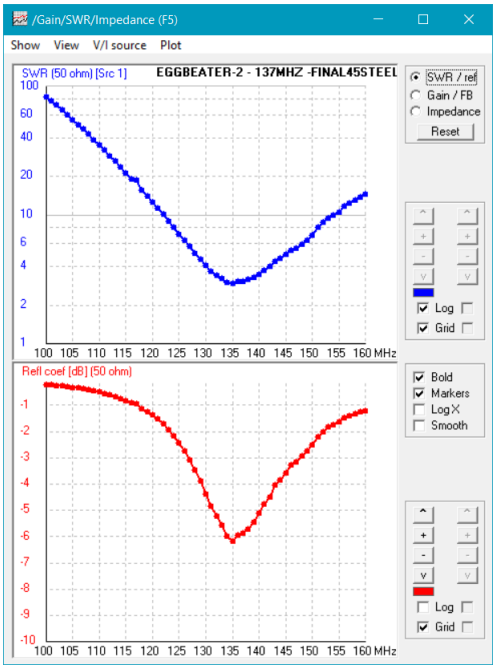


Figura 5.16 – *Gráfica de ROE y coeficiente de reflexión para* $Z_0 = 50\Omega$.

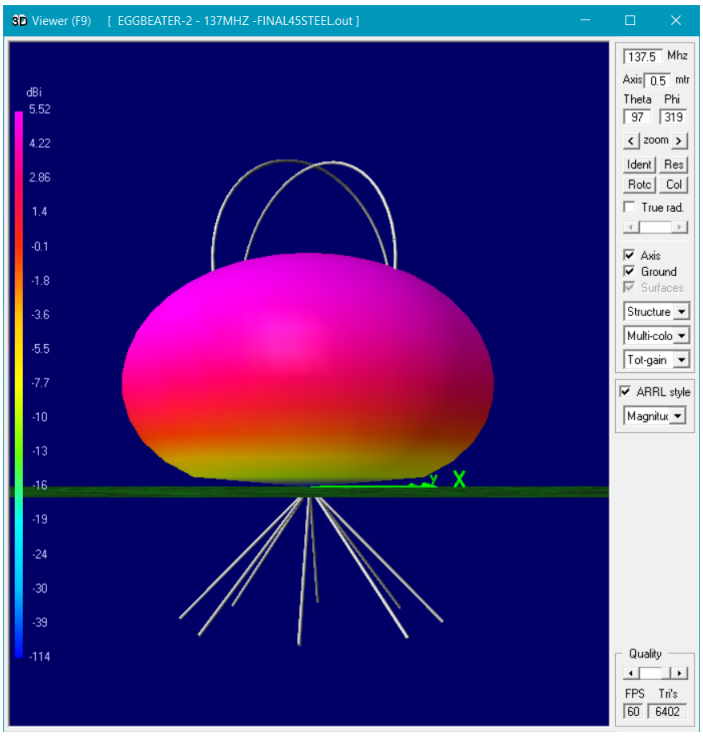


Figura 5.17 – *Patrón de radiación de antena Eggbeater en 3D resultante en 4NEC2.*

Se desea comparar estos resultados con el software CST y dar algún valor añadido a la optimización de la antena.

5.1.2.1.2 Simulación con CST MWS

CST es un software de simulación EM 3D muy usado en el ámbito ingenieril para el diseño y simulación electromagnética de elementos radiantes.

Partiendo de los conocimientos que se han adquirido con la simulación en 4NEC2 sobre la antena Eggbeater, se procede a su comparativa empleando CST. A continuación se detalla el proceso seguido para el diseño y simulación de la antena Eggbeater. Seguidamente se mostrarán los resultados obtenidos en este software.

Manual de diseño y simulación en CST

Se comienza con un diseño básico que se irá mejorando para tener unos resultados más próximos a la realidad. En la Figura 5.18 se muestra el diseño preliminar basándose en los datos teóricos y los optimizados usando 4NEC2. En esta figura se visualiza el entorno de trabajo de CST. En la opción '*Modeling*' del menú principal (vista en la Figura 5.19) se tiene una serie de herramientas para el diseño 3D de la estructura propuesta, consistentes en formas básicas como cubos, cilindros, torus, esferas y conos. A partir de estas estructuras básicas se puede modelar la antena mediante intersecciones de formas, obstrucciones, sustracciones, rotaciones, etc.

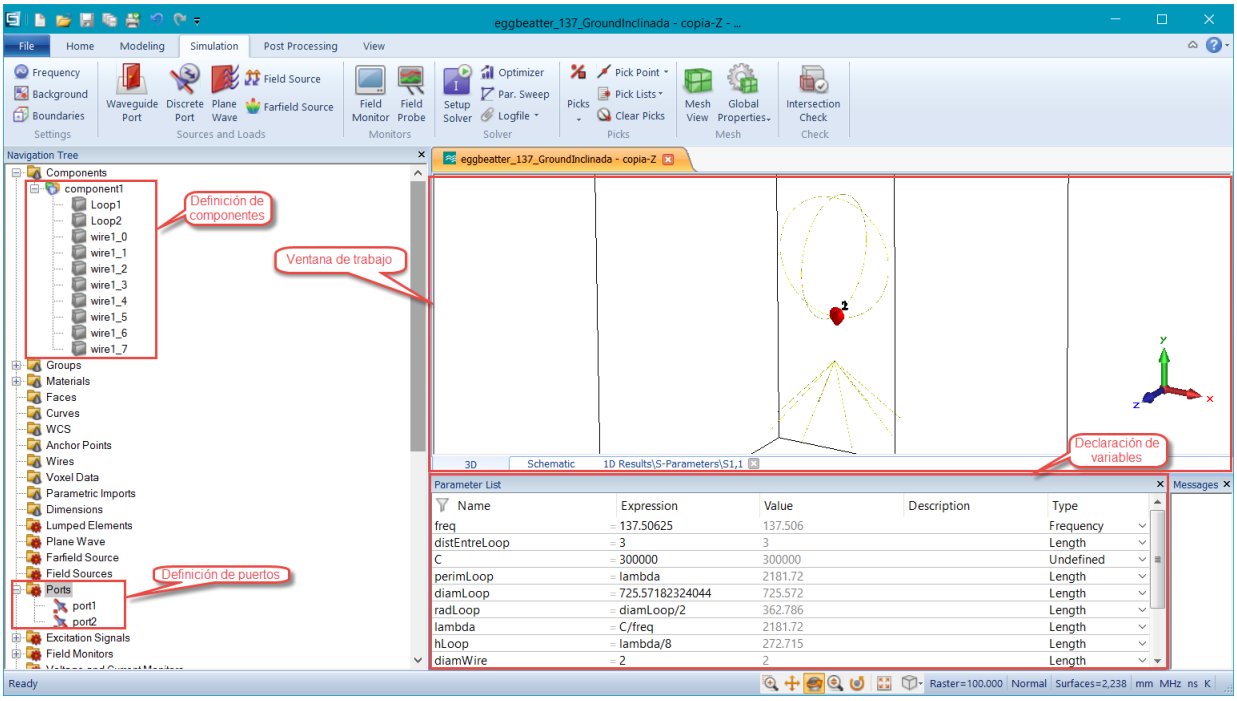


Figura 5.18 – *Diseño preliminar de antena Eggbeater en CST.*

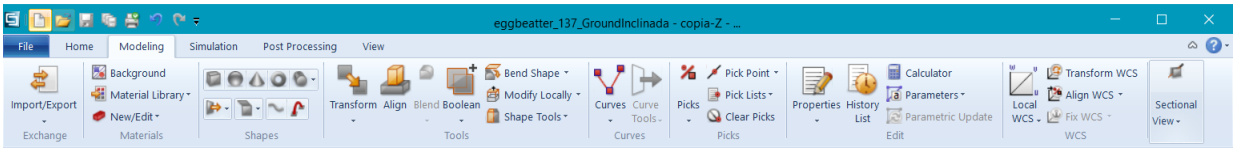


Figura 5.19 – *Menú para diseño 3D en CST.*

En la Figura 5.18 se puede ver que en el lateral izquierdo de la pantalla se localiza el árbol de navegación del diseño. En la carpeta '*Components*' quedan definidos los elementos de la antena creados, pudiendo acceder a ellos para posteriores ediciones. En la parte inferior de la pantalla está la ventana para la declaración de variables. Es muy útil definir la estructura parametrizándola, es decir, se definen sus dimensiones con variables declaradas en esta ventana, de modo que su modificación será rápida y, más importante aún, en el momento de optimización de la antena, estas variables serán modificadas por el software para alcanzar unos objetivos fijados siguiendo un modelo probabilístico seleccionado.

Para la correcta simulación del elemento radiante, hay que considerar una serie de conceptos importantes en CST. Es necesario definir el material de los componentes integrantes de la antena, para lo cual CST cuenta con una amplia librería de materiales con sus especificaciones eléctricas, físicas y mecánicas guardadas. No obstante, si un material no está en la librería, CST da la opción de crear un nuevo material. Del mismo modo que se debe especificar el tipo de material de la antena, también es necesario especificar el material exterior a ella, tratándose de aire en el caso que nos ocupa. '*Background Properties*' es la herramienta para realizarlo y definir el espacio a considerar alrededor de la antena (en este caso, se ajusta el espacio a la antena, como se ve en la Figura 5.18 donde la antena está contenida en una caja).

En los extremos de cada uno de los loops de la antena se debe colocar el puerto de entrada. La alimentación mediante estos puertos en CST se supone simétrica. Para poder introducir el puerto en CST los loops no están completamente cerrados, sino que tienen un corte que deja un espacio en la parte inferior de cada loop. Esta separación es de 2 cm. En este caso, se cuenta con dos puertos de entrada, desfasados 90^{o} uno con respecto al otro para obtener polarización circular y de igual amplitud. En la fabricación produciremos este desfase de 90^{o} mediante un tramo de línea coaxial de longitud $\lambda/4$. Se colocan puertos discretos, que se encuentran en el menú '*Simulation*'. Por defecto este puerto tiene una impedancia de $Z_0 = 50\Omega$, aunque se puede modificar en el momento de editado o en post-procesado.

Antes de simular, se debe especificar la banda de frecuencia para la cual realizar la simulación y las unidades que sigue el diseño (mm, Hz, Kelvin, etc). En la siguiente Figura se observa otra condición que se debe especificar antes de simular. Se trata de las condiciones de frontera, que en este caso, al tratarse de una antena radiando en un espacio de aire, se indica la condición de '*Open (add space)*'.

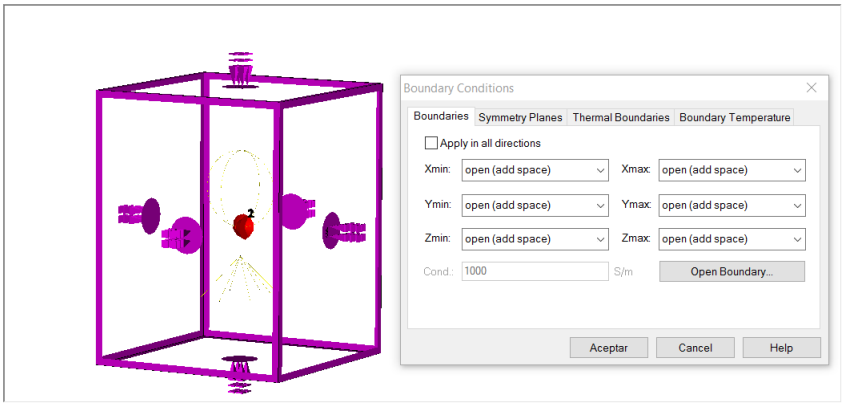
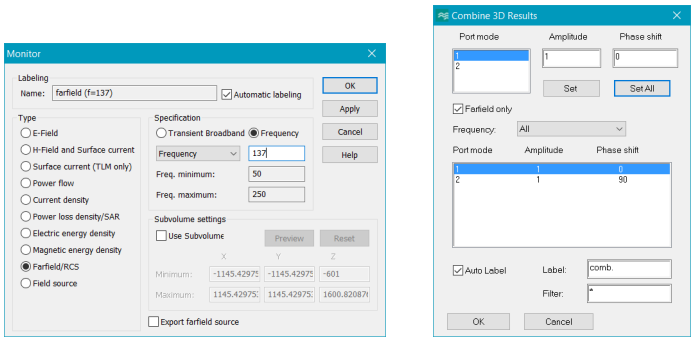


Figura 5.20 – *Condiciones de frontera para un elemento radiante.*

Si se desea información acerca del patrón de radiación de la antena, el campo eléctrico y magnético o la densidad de potencia radiada a una distancia z, se deben seleccionar los monitores de campo correspondientes '*Field Monitor*'. Para el caso descrito, se desea conocer el patrón de radiación en campo lejano a la frecuencia de operación, por lo que se selecciona *Farfield*, como refleja la Figura 5.21. Dado que la antena consta de dos elementos radiantes, será necesario combinar los resultados en post-procesado, introduciendo un desfase en uno de los puertos.



(a) **(b)**

Figura 5.21 – *a) Selección de monitor de campo para el cálculo del patrón de radiación de la antena en CST. b) Combinación de los resultados 3D de ambos loops.*

Los modos de simulación en CST son diversos (ver 5.22), atendiendo a las necesidades del elemento y al estudio que se quiera realizar del mismo. En este caso se ha usado *Integral Equation Solver* que permite asignar el número de muestras en simulación para una banda de frecuencia dada y aumentar este número para frecuencias determinadas, ofreciendo mayor precisión en estas longitudes de onda.

Por último, se debe definir el mallado de la estructura '*Mesh Properties*', aumentándolo

para una mejor simulación en estructuras complejas, pero valorando el coste computacional, que se incrementa exponencialmente.

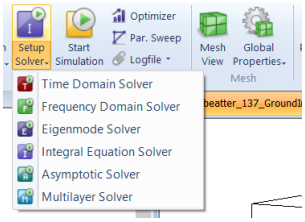
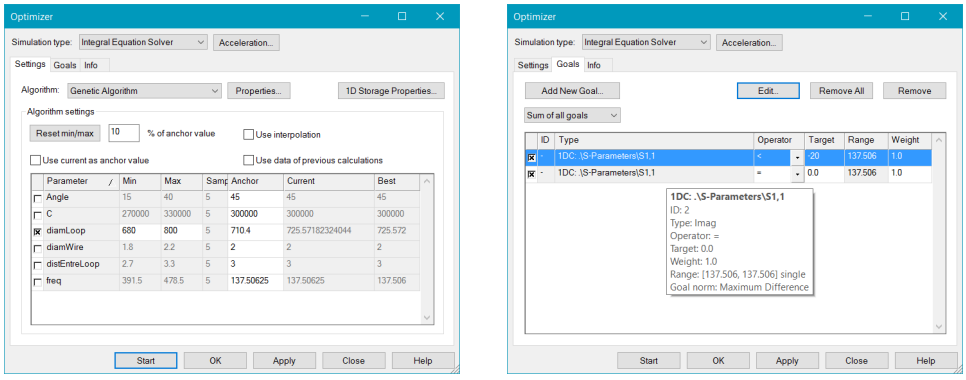


Figura 5.22 – *Modos de simulación de CST.*

Para mejorar la antena, ya sea para sintonizar la frecuencia de operación, aumentar el ancho de banda, hacerla más directiva o aumentar el ancho de haz, se utiliza la herramienta '*Optimizer*' de CST. Esta funcionalidad permite alcanzar un objetivo con modificaciones de las dimensiones de la antena seleccionadas, siguiendo un método probabilístico. En la Figura 5.23 se muestra la ventana de configuración del optimizador. En primer lugar se elige el algoritmo a implementar y se seleccionan las variables que se desean modificar de manera iterativa para alcanzar el objetivo. En este caso, se desea que la adaptación sea menor a -20 dB y que la parte imaginaria del coeficiente de reflexión sea cero para la frecuencia de operación. Una antena se considera adaptada cuando $|S_{11}| < -10dB$, aunque en simulación se debe ir más allá para asegurar la correcta adaptación de la antena. Para que la antena trabaje en campo de radiación, se debe eliminar la parte reactiva del coeficiente de reflexión en la medida de lo posible ($\angle S_{11} = 0$), eliminando el comportamiento inductivo o capacitivo. Para ello, la longitud de los loops, que son los elementos radiantes de la antena, son los elementos a modificar. Para el caso de ampliar el ancho de haz, el ángulo de los reflectores ha sido modificado y finalmente se ha determinado que 45º cumple con los requerimientos.



(a) **(b)**

Figura 5.23 – *a) Ventana de cofiguración del optimizador. b) Definición de los objetivos de optimización.*

El diseño final de la antena en 3D se puede ver a continuación.

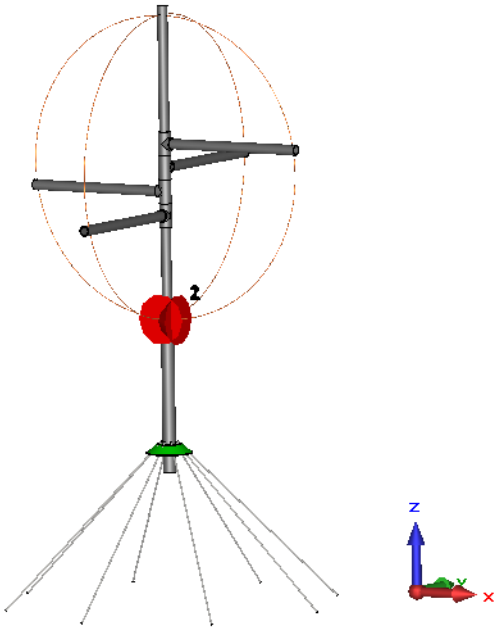


Figura 5.24 – *Diseño final de antena Eggbeater en CST.*

5

Resultados en CST

Para comprobar que se cumplen los requerimientos para nuestra antena es importante visualizar el diagrama de radiación y $|S_{11}|$, el coeficiente de reflexión a la entrada, con la salida terminada en carga adaptada, es decir, $Z_L = Z_0$ (la impedancia de carga es igual a la impedancia característica de la línea) [6]. Si este parámetro es mínimo a nuestra frecuencia de trabajo, se obtendrán las mínimas pérdidas en la recepción del APT.

Las simulaciones resultantes en CST se poden ajustar más a los requisitos pedidos, por lo que se ha hecho uso de la herramienta de CST de optimización, mostrada anteriormente. Finalmente se han conseguido los siguientes resultados:

- $|S_{11}|$ *e impedancia de salida:*

  Ajustando la longitud de los elementos radiantes se consigue una mínima reflexión a la entrada. Los puertos tienen una impedancia característica de $Z_0 = 50\Omega$, por lo que habrá onda reflejada dado que la resistencia de radiación de la antena teóricamente $R_a = 200\Omega$ y según la simulación en 4NEC2 de $R_a = 155\Omega$. Se debe sintonizar la banda de operación, aunque haya onda reflejada (considerando que la antena está adaptada para $S_{11} < -10dB$). Posteriormente habrá que hacer de manera oportuna la adaptación de impedancias para la una línea de transmisión de $Z_0 = 50\Omega$. Además de esto, la reacctancia puede no ser nula, por lo que habrá pérdidas de carácter inductivo

o capacitivo. De modo que, la optimización debe tener como objetivo $\angle S_{11} = 0$ y $|S_{11}| < -10dB$ para ambos loops.

Finalmente se consigue obtener los siguientes resultados:

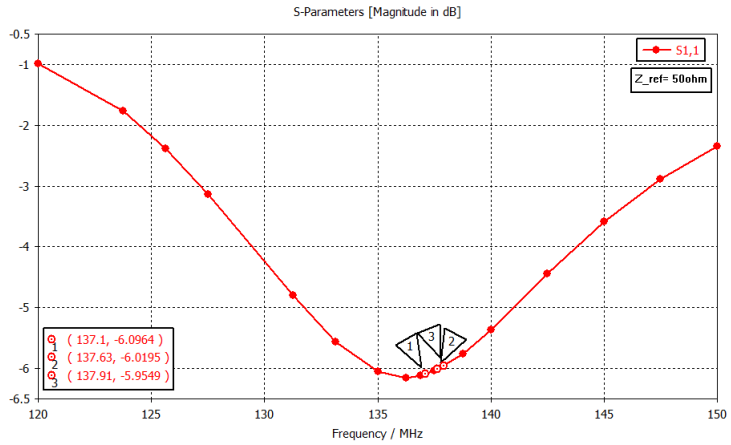


Figura 5.25 – $|S_{11}|$ *de antena Eggbeater sin adaptación de impedancias.*

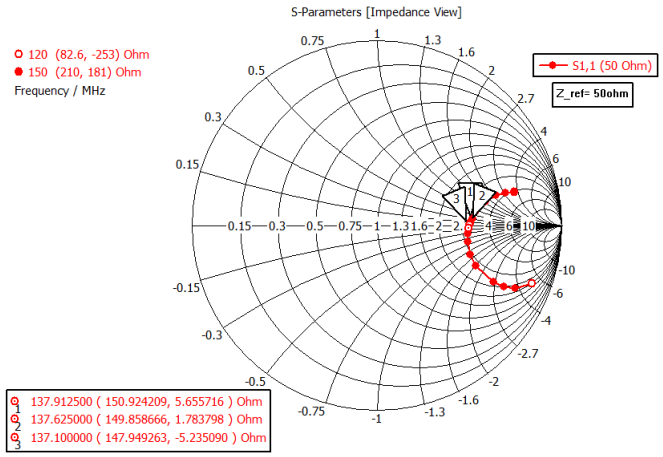
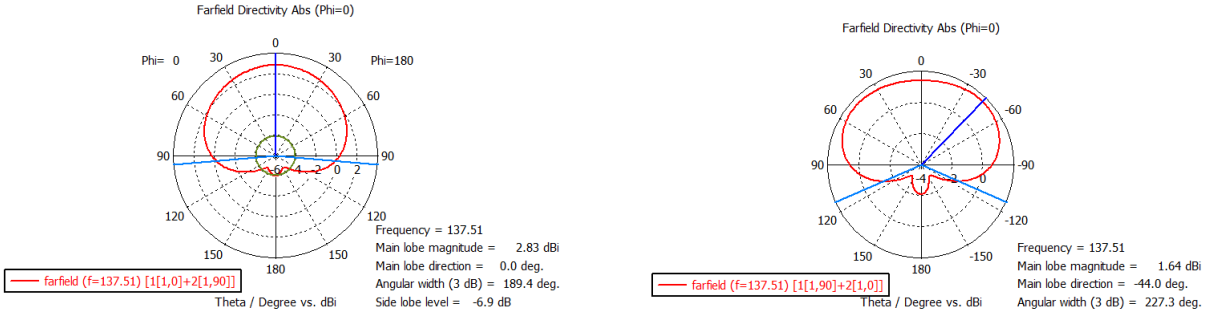


Figura 5.26 – *Impedancia de antena Eggbeater en Carta de Smith.*

- *Ancho de haz, directividad y ganancia:*

  Para maximizar el ángulo de linea de visión directa (Line of Sight (LOS)) con los satélites NOAA con una caída de potencia de 3dB, los reflectores se inclinan un ángulo α aumentando así HPBW. Así pues, los resultados mostrados en la Figura 5.27 verifican que con los reflectores inclinados el ancho de haz aumenta, a costa de perder la máxima directividad en 0^o. 4NEC2 no ofrecía unos resultados demasiado fiables en el cálculo del patrón de radiación, como se puede observar si comparamos esta imagen con la Figura 5.14.



(a) **(b)**

Figura 5.27 – *a) Patrón de radiación para reflectores con un ángulo de inclinación de 0º. b) Patrón de radiación para reflectores con un ángulo de inclinación de 45º.*

Como ya se venía comentando, la máxima directividad de la antena se ve afectada. La antena Eggbeater ha sido diseñada para recepción, y no es necesario tener una ganancia elevada para obtener buenos resultados. Es más interesante tener una antena lo más omnidireccional posible.

En la Figura 5.28 puede verse la ganancia real de la antena. La ganancia no coincide con la directividad, es menor. La definición de directividad no tiene en cuenta la eficiencia de la antena y la supone como un radiador perfecto de energía electromagnética, es decir, la directividad supone a la antena sin pérdidas. En realidad las antenas se construyen con materiales que son conductores imperfectos, igual que material aislante que se utilizan en ellas, por lo que una parte de la potencia suministrada a la antena se perderá en ésta, bien sea por calentamiento a causa de la resistencia de los conductores o por fugas en los dieléctricos, dando como resultado una reducción en la potencia neta y, lo que define la eficiencia de la antena (η_e), cuyo valor está comprendido entre 0 y 1. Se define entonces la ganancia de una antena como 5.1.5:

$$G(\theta, \phi) = \eta_e D(\theta, \phi) \tag{5.1.5}$$

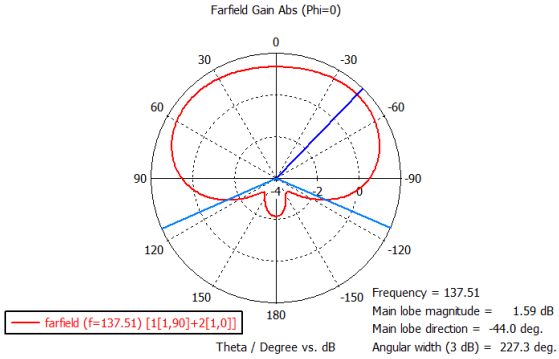


Figura 5.28 – *Ganancia de antena Eggbeater.*

CST ofrece la posibilidad de visualizar el diagrama de radiación en 3D (véase Figura 5.29).

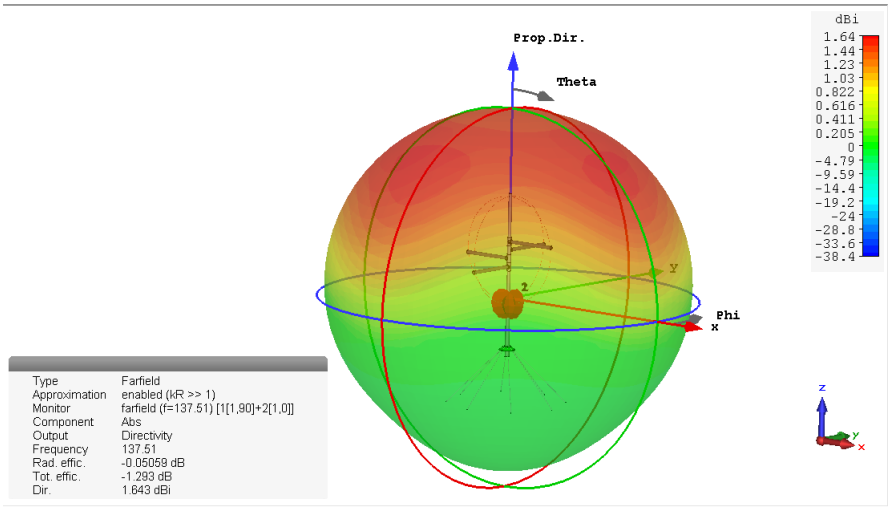


Figura 5.29 – *Patrón de radiación de antena Eggbeater en 3D.*

- *Cross-polarization:*

Otro punto a evaluar es el comportamiento de la antena con respecto la polarización. Se necesita una antena con polarización circular a derechas RHCP, ya que es el tipo de polarización de las antenas transmisoras de los satélites. Pueden existir pérdidas debidas a la polarización cruzada que se deben evitar. Para ello, la radiación en Left Hand Circular Polarization (LHCP) debe ser mínima, nunca será nula. En la Figura 5.30 se puede visualizar la eficiencia de polarización cruzada, demostrando que prácticamente toda la potencia se radia en onda RHCP.

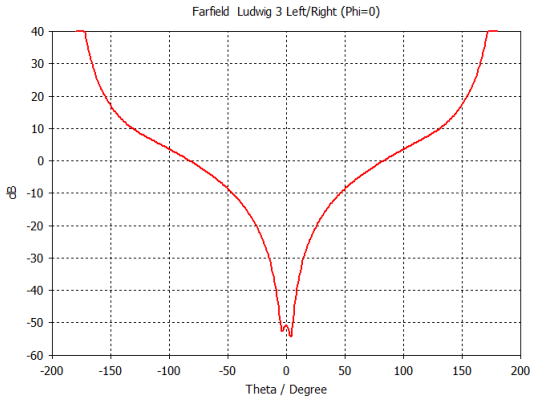


Figura 5.30 – *Patrón de radiación 2D de los componentes de cross-polarization de antena Eggbeater.*

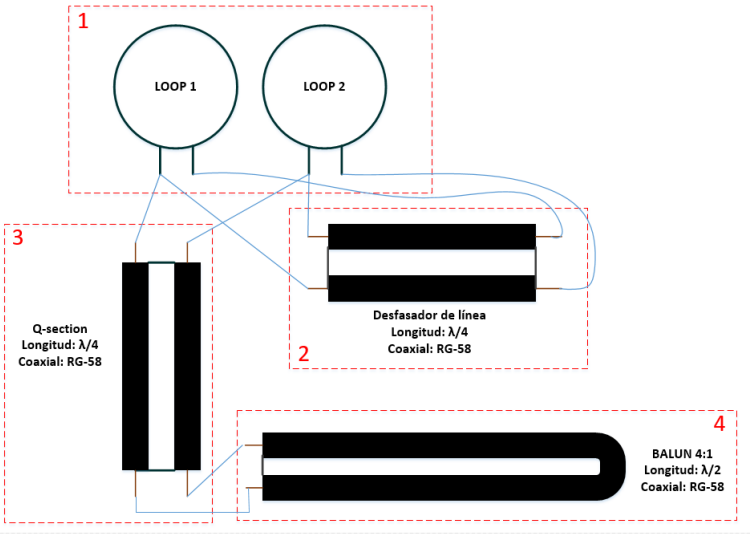


Figura 5.31 – *Esquema de alimentación de la antena Eggbeater.*

5.1.2.2 Desfasador, adaptación de impedancias y simetrización de corriente

Para concluir el diseño de la antena se tiene que adaptar la impedancia de la antena a la impedancia de salida de la línea de transmisión (50Ω), desfasar los loops 90^o uno con respecto al otro y simetrizar la alimentación sobre los loops. Se va a conseguir con el uso de líneas de transmisión, como se muestra en el siguiente esquema.

- Bloque 1: Los loops de la antena tienen una impedancia a la frecuencia central de la banda de trabajo ($f_c = 137.51MHz$) de $Z = (149.43 + 0.235j)\Omega$.
- Bloque 2: Desfasador de línea. Para tener polarización circular es necesario desfasar un loop 90^o con respecto al otro. Se consigue con una línea de transmisión de un cuarto de longitud de onda.
- Bloque 3: Transformador de alta impedancia a la salida del Balun (balanced-unbalanced lines transformer) a baja impedancia de antena. Se ha transformado la impedancia con una línea de transmisión de $\lambda/4$. A continuación se explicará en que consiste la línea Q-section usada.
- Bloque 4: Trasformador Balun 4:1, usado para unir una línea no balanceada de alimentación con una carga balanceada (nuestra antena).

A continuación se va a explicar detalladamente los bloques mostrados en la Figura 5.31:

a) **Desfasador:**

Para desfasar los loops se usa una línea de transmisión de longitud $\lambda/4$, como se puede ver en la Figura 5.32. Una línea de transmisión de un cuarto de longitud de onda desfasa

la señal 90º entre los loops. La polarización circular se produce por la radiación de los dos loops alimentados a j0 y a j90.

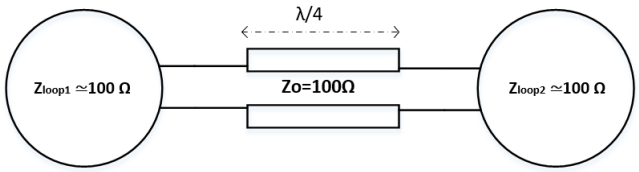


Figura 5.32 – *Conexionado de los loops con la línea desfasadora* $\lambda/4$.

Los loops presentan ambos una impedancia cercana a los 100Ω , de modo que la impedancia característica del coaxial usado como desfasador debe ser 100Ω, según el transformador $\lambda/4$ (5.1.6).

$$Z_0 = \sqrt{Z_1 Z_2} \tag{5.1.6}$$

Esta impedancia característica se puede conseguir fabricando una Q-section. La Q-section consiste en dos cables coaxiales paralelos unidos por sus mallas que tendrá una impedancia característica del doble de la de la línea utilizada para fabricarla. Usando un cable coaxial RG-58 de 50Ω, la Q-section tendrá una impedancia característica de 100Ω.

La impedancia resultante de la conexión de los loops juntos debería ser 50Ω. Esto sería lo ideal para posteriormente conseguir una buena adaptación de impedancias, pero nuestros loops tienen una impedancia de $(149.43 + 0.235j)\Omega$. Para calcular la longitud de estas líneas de transmisión, se calcula la longitud de onda considerando la velocidad de propagación del interior del cable y la frecuencia de trabajo.

$$\lambda_{cable} = \frac{c \cdot v_f}{f} \tag{5.1.7}$$

donde v_f es la velocidad de propagación del cable, que para RG-58 es de 0.66. De modo que:

$Longitud_{cable\lambda/4} = 36cm$

b) **Simetrización:**

Cuando se conecta un dipolo doblado o loop (línea equilibrada) a una línea coaxial no balanceada (línea no equilibrada) ocurre que las corrientes no se distribuyen del mismo modo, circulando por cada brazo del dipolo doblado diferente intensidad [36]. Una buena solución es el uso de BALUN (Bloque 4 en Figura 5.31), que permite convertir líneas de transmisión no balanceadas en líneas balanceadas, cumpliendo con la función de simetrizador de corriente. Además, también es usado para la adaptación de impedancias, ya que algunos BALUN transforman la impedancia de salida.

En este caso, para antenas de tipo loop se suele utilizar BALUN 4:1. Esta relación de transformación se aplica en impedancias de antena de 200Ω e impedancia característica de línea coaxial de 50Ω. De modo que habrá que considerar esta cuestión en la adaptación de impedancias de la antena.

Para su fabricación se ha usado un cable coaxial de $Z_0 = 50\Omega$ RG-58, de una longitud de $\lambda/2$, que considerando la ecuación 5.1.7 se obtiene: $Longitud_{balun} = 72cm$

c) **Adaptación de impedancias:**

Atendiendo a la ecuación 5.1.6 y a la Figura 5.31, la impedancia resultante de un loop en junto con el bloque 2 (correspondiente al desfasador de línea) es $Z = (66.92 - 0.1052j)\Omega$, obtenido calculando el paralelo de la impedancia de un loop y el tramo de línea de 100Ω. Se comprueba que no se obtienen los 50Ω exactos buscados para la óptima adaptación. A continuación se tiene el segundo loop conectado en paralelo a la impedancia resultante. De este modo, se obtiene que:

$$Z_{total} = (66.92 - j0.1052)//(149.43 + j0.235) = (46.2212 - j0.0282)\Omega$$

Es una impedancia de salida cercana a los 50Ω buscados, de modo que se considerarán 50Ω, a la salida del bloque desfasador.

Con la razón de transformación 4:1 del BALUN del Bloque 4, se debe conseguir una impedancia de antena de 200Ω para conseguir la transformación a 50Ω que permite este balun. Para ello, se introduce una nueva línea de adaptación, un nuevo transformador $\lambda/4$. A partir de la ecuación 5.1.6, se obtiene que una impedancia característica de $Z_0 = \sqrt{200 \cdot 50} = 100\Omega$. De modo que se utilizará una Q-section de 100Ω como se muestra en el Bloque 3 de la Figura 5.31.

En base a estos cálculos, se simula el circuito propuesto con ayuda del software de diseño electrónico ADS para comprobar si finalmente se consigue la adaptación de impedancias para la correcta alimentación de la antena.

En primer lugar, haciendo uso de la herramienta de cálculo de líneas de transmisión de ADS '*LineCalc*', se calculan las dimensiones del diámetro de conductor interno y exterior de la línea coaxial de 50Ω (ver Figura 5.33). En A se puede ver la hoja de características del cable coaxial empleado.

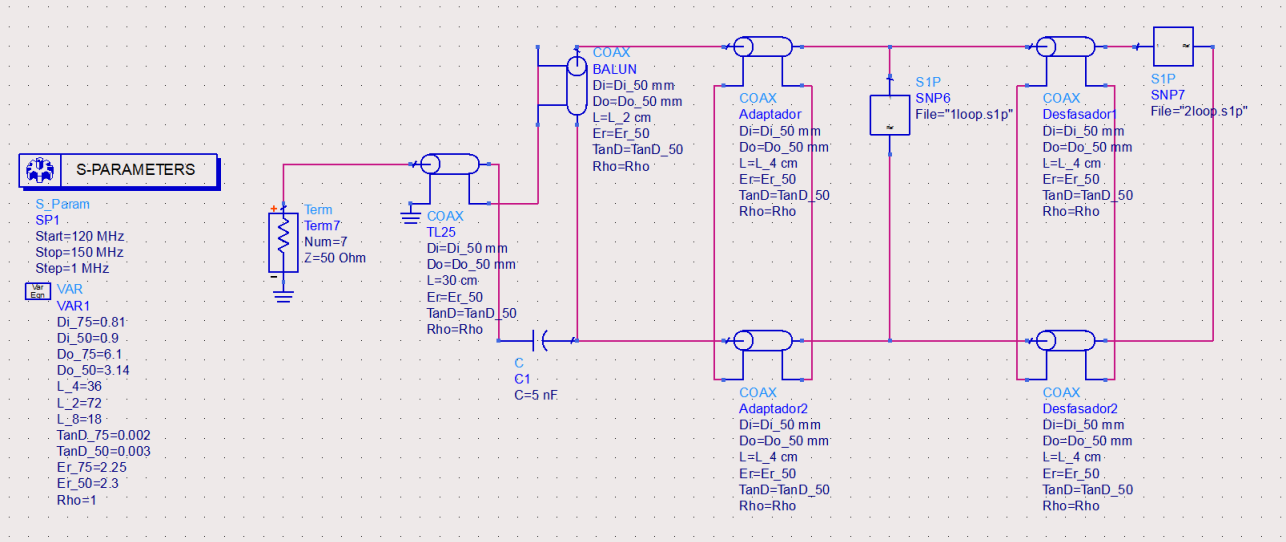


Figura 5.34 – *Esquemático de circuito para la alimentación de la antena en ADS.*

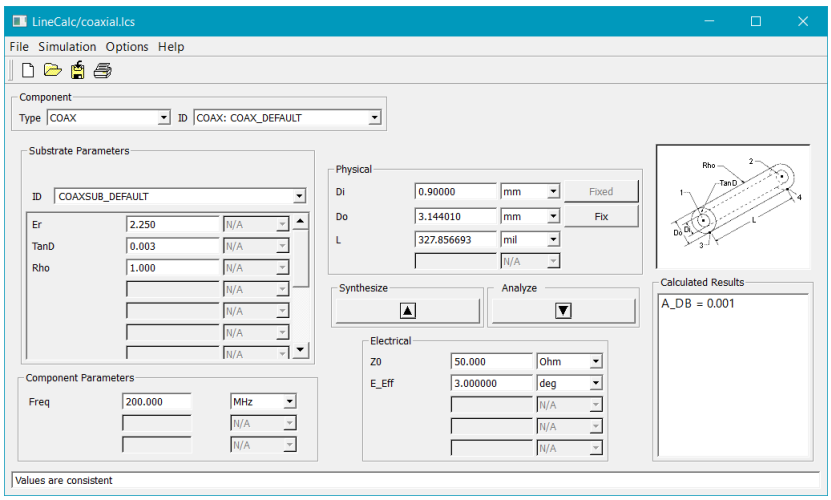
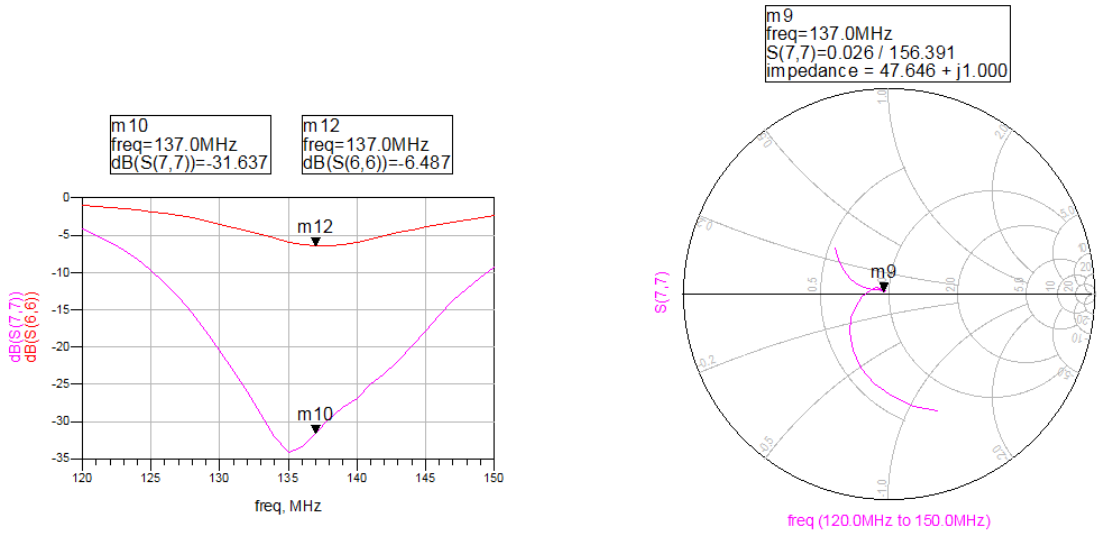


Figura 5.33 – *Herramienta LineCalc de ADS*

Se realiza el esquemático del circuito mostrado en el diagrama de bloque de la Figura 5.31. Se declaran variables correspondientes a las dimensiones del cable coaxial RG-58 y las longitudes de las líneas de transmisión $\lambda/4$ y $\lambda/2$ de a cuerdo a la velocidad de fase del cableado (6.6). Es posible que sea necesario introducir en el circuito una reactancia (capacidad o bobina) para conseguir una impedancia de salida con $jX \approx 0$, reduciendo así las pérdidas reactivas.

Finalmente, el resultado del parámetro S_{11} se puede ver en la Figura 5.35.a), donde se comprueba que para la banda de operación de los NOAA se consigue una adaptación de $S_{11} < -30dB$.



(a) **(b)**

Figura 5.35 – *a) S_{11} calculado con circuito de alimentación de antena en ADS. b) Impedancia de salida calculada en ADS.*

5.1.2.3 Fabricación de Eggbeater

En las Figuras 5.36, 5.37 y 5.38 se muestran los elementos integradores de la antena Eggbeater necesarios para su construcción, siguiendo las especificaciones de la etapa de diseño:

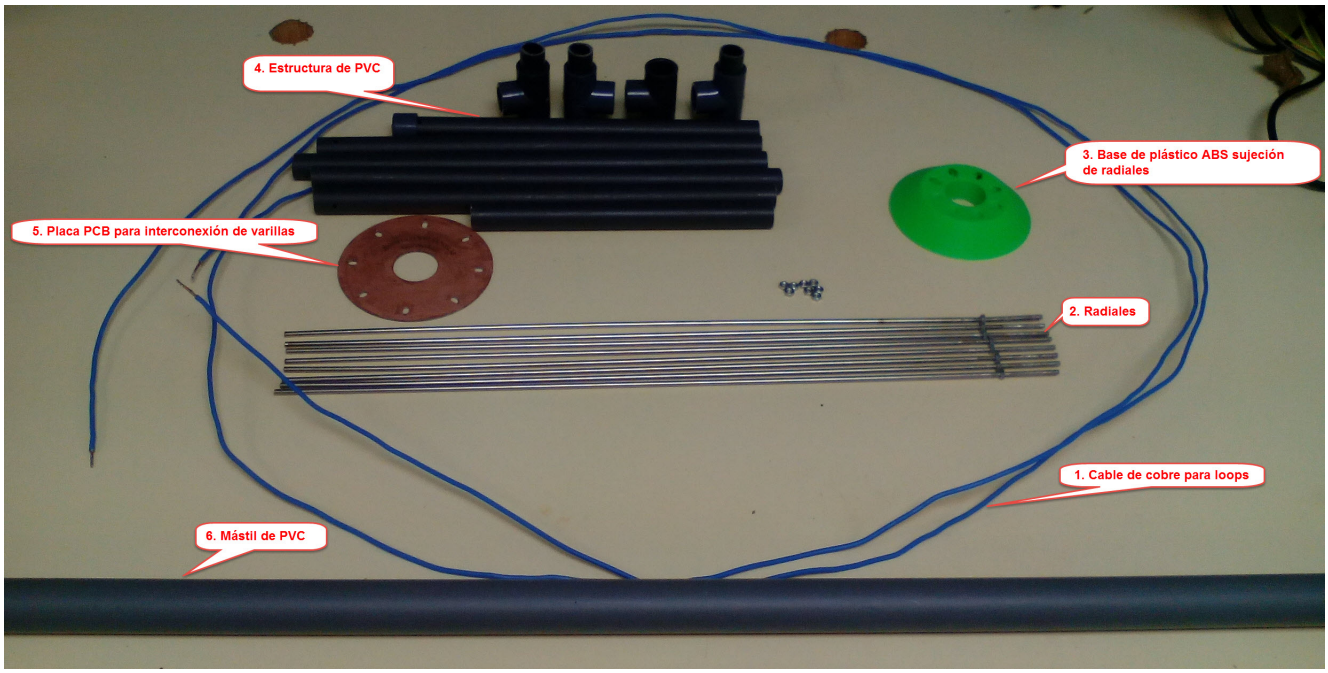


Figura 5.36 – *Elementos de antenna Eggbeater.*

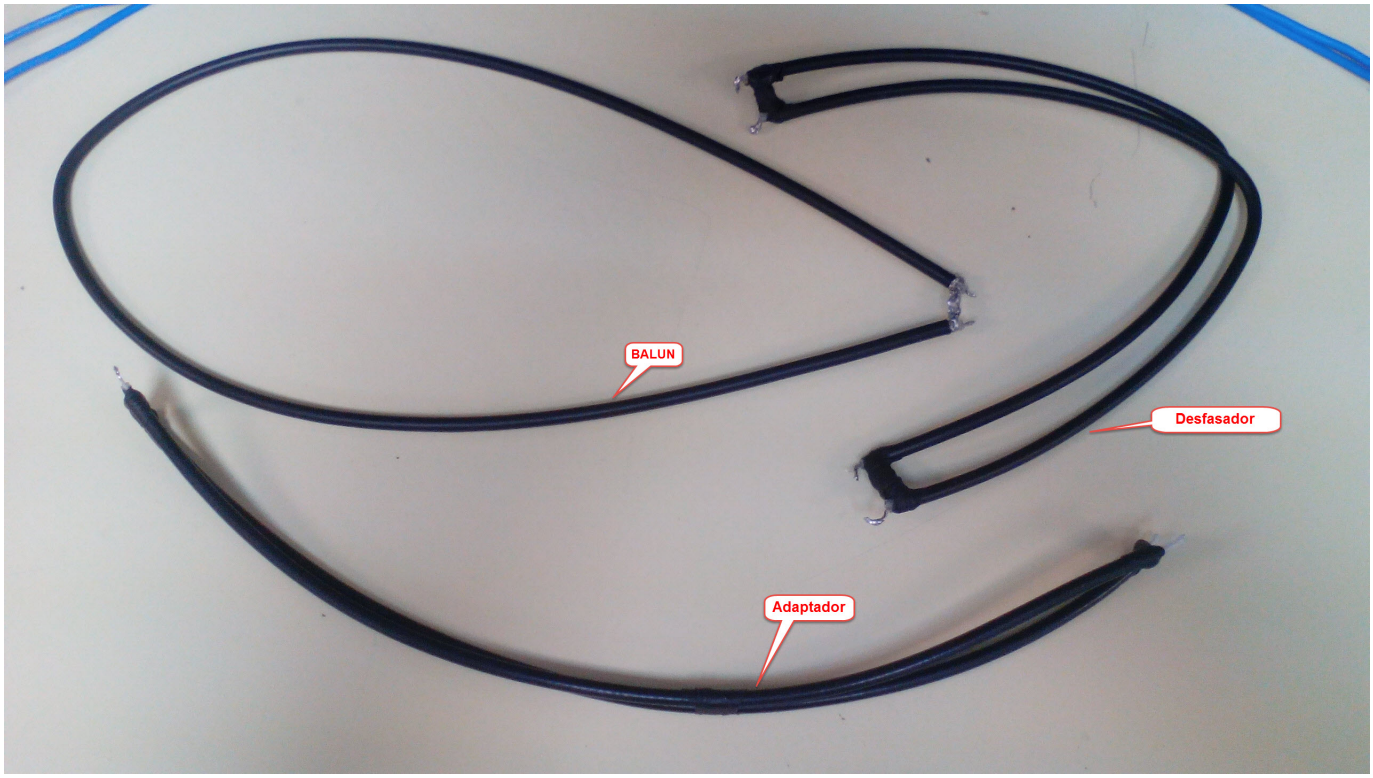


Figura 5.37 – *Cableado de simetría, adaptación de impedancias, y desfasador.*

Figura 5.38 – *Piezas de 2 cm preestañadas para la interconexión de loops líneas de transmisión.*

En la Figura 5.39 se puede ver la interconexión de la línea desfasadora con las piezas PCB a la que ha sido soldada, donde como se puede observar en la Figura 5.38, el cobre ha sido eliminado en la zona central para evitar que los vivos se toquen.

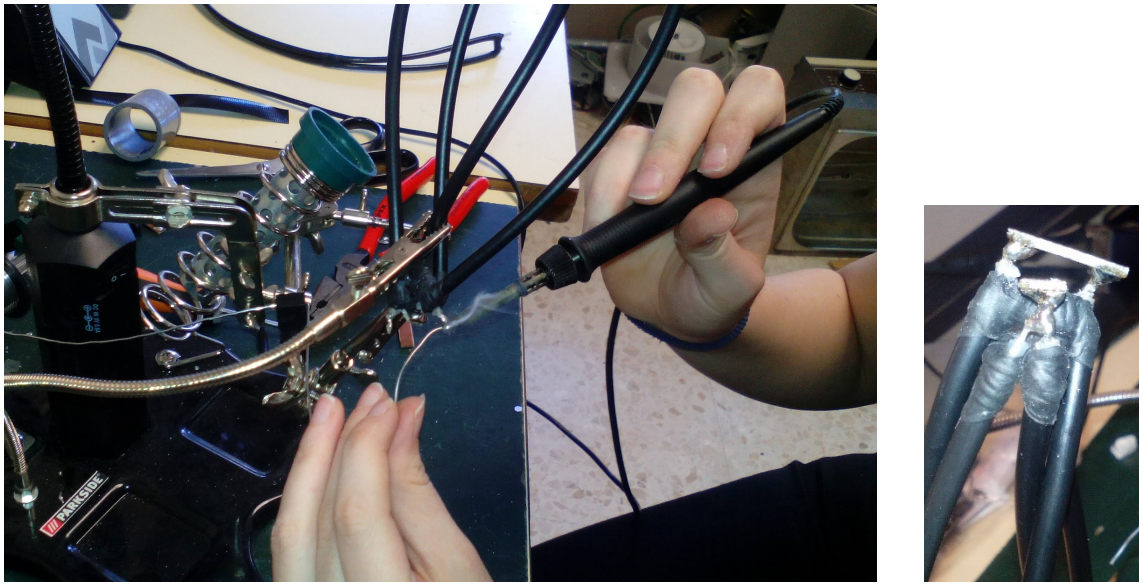


Figura 5.39 – *Soldadura de la línea desfasadora con la pieza de PBC de conexionado de los loops.*

Véase en la Figura 5.40 el montaje de los radiales, la base de sujeción de plástico ABS y el plano de masa fabricado con PCB. Para la fabricación de la base de sujeción primeramente se ha hecho el diseño en Solidworks e impreso con una impresora 3D.

Figura 5.40 – *Montaje del plano de masa.*

Ensamblando las piezas mostradas en el laboratorio de Electrónica de la Facultad de Ciencias de la Universidad de Granada se consigue el resultado que se vio en la simulación CST (Figura 5.24).

Para recibir la señal de los satélites NOAA, la antena ha sido colocada en la azotea de la Facultad de Ciencias de la Universidad de Granada. Para ello ha sido necesario acoplar la antena a un mástil de PVC y posteriormente apoyarlo en una garra metálica anclada a una pequeña caseta. Véase en la Figura 5.41 la realización de un orificio para pasar el cable de alimentación de la antena para estar resguardado. Finalmente, en la Figura 5.42 se muestra la colocación de la Eggbeater.

Figura 5.41 – *Adaptación de una barra metálica a la incorporación de la antena con un orificio para el pase del cableado.*

Figura 5.42 – *Colocación de antena Eggbeater en la terraza de la Facultad de Ciencias de la Universidad de Granada.*

5.2 Servicio WEB para recepción satelital

La señal APT grabada será decodificada para obtener la imagen tomada por el satélite NOAA-19, NOAA-15 o NOAA-18 que pase sobre la estación terrena de la Universidad de Granada (**Locator: IC87CE, Callsign: EB7DZP**). Es posible predecir el pase del satélite, de modo que el receptor SDR permanece a la escucha cuando el predictor lo indique y la señal APT se guarda en un fichero de audio *.wav*. Posteriormente se decodifica esta señal y se obtienen las imágenes, almacenándolas en una base de datos alojada en el servidor.

A continuación se van a detallar las tareas necesarias para la implementación del servicio web creado. El servicio web está alojado en un servidos Linux, Ubuntu 14.4, dado que sobre este SO es GNU. Por otro lado, el control de SDR y el software para decodificación de la señal está corriendo en tiempo real sobre Raspberry Pi 3 B usando el SO Raspbian Jessie Lite.

En la interfaz de usuario, estará disponible el *tracking* en tiempo real de los satélites NOAA, además de otra información de interés para el cliente final, como futuros pases, duración de estos y demás información orbital. A través de la web el usuario puede visualizar las imágenes y proceder a su descarga desde una base de datos.

Para el servidor se ha utilizado Python como lenguaje de programación, ya que se dispone de infinidad de librerías creadas que facilitan la implementación y consigue integrar

fácilmente librerías escritas en C, como los predictores de órbitas de satélites. Mientras tanto, Javascript, HTML y CSS son usados para el desarrollo front-end.

5.2.1 Back-end

Como ya se ha mencionado, el servidor web se ha implementado con Python y Flask como *microframework*. Se ha decidido usar Flask porque es más ligero frente a otros *frameworks* de Python como *Django*, que incorpora multitud de módulos para tareas comunes de desarrollo web. Flask es *Open Source* y proporciona lo mínimo necesario para la creación de aplicaciones web, sin necesidad de instalar *Apache*. Flask es ideal para aplicaciones de contenido estático o *APIs REST*.

Se puede crear una aplicación fácilmente con Flask con pocas líneas, como se ve a continuación:

```
1
2  from flask import
3
4  app=Flask(_name_)
5
6
7  @app.route(``/'')
8
9  def hello():
10
11   return ``hello world..''
12
13
14 if _name_==``_main_'':
15
16   app.run()
```

5

En el Anexo C se muestra las herramientas necesarias para poner en funcionamiento un servicio en Python en el servidor web Ubuntu 14,4. En resumen, será necesario:

-Instalación de Python 2.7 (o posterior).

-Creación de entorno virtual para fácil gestión del servicio.

-Instalación de microframework Flask.

-Descargas de librerías principales para nuestro servicio web dentro del entorno virtual.

Nuestra aplicación web estará corriendo en el puerto 8004, abierto en el router principal para dar salida a la información de este servicio. El script a ejecutar para activar el servicio es *'webServer.py'*. Para daemonizar el servicio se ha utilizado Production Process Manager (PM2), un gestor de procesos de fácil y cómodo uso que permite mediante una simple línea de comando de terminal inicializar un proceso, poner en pausa, pararlo, eliminarlo, consultar su estado o ver los mensajes *logs* generados y errores (ver Anexo D).

5.2.1.1 Propagador de órbita para satélites NOAA

En la sección 4.1.3, se mencionan una serie de librerías en Python que implementan los métodos matemáticos para el cálculo de los *elementos keplerianos* de los diferentes satélites.

Todas ellas emplean el algoritmo SGP4/SDP4 escrito en C. Sin embargo, algunas librerías extraen más información orbital que otras, permitiendo así decantarnos por una u otra de las opciones encontradas.

En primer lugar, dado que cualquiera de los propagadores necesita como *input* el TLE actualizado de cada unos de los satélites, se crea un documento de texto *.txt* que contenga los TLE de los satélites NOAA activos. Este documento será actualizado cada 2 días, descargando la información de la página de *Celestrack* [22]. Como ya se ha mencionado, *Celestrack* ofrece dicha información de forma gratuita y sin necesidad de registro, evitando la necesidad de introducir credenciales para obtener la actualización de la información dinamicamente. Actualmente solo hay tres satélites NOAA activos (NOAA-15, NOAA-18 y NOAA-19. En la actualización de la información, no solo se actualizará el TLE de estos satélites, sino que se añadirá unos nuevas líneas en caso de que otro satélite NOAA vuelva a estar operativo.

Actualización de TLE desde *Celestrack*

```
1
2  def upgrade_TLE():
3
4  #Se seleccionan las líneas de interés de noaa.txt de Celestrack y se copian en un fichero de
     texto.
5
6    TLE_URL = ('http://celestrak.com/NORAD/elements/noaa.txt')
7
8    url = urllib2.urlopen(TLE_URL).read()
9
10   f = open("tleNOAA.txt","a")
11
12   f.write(lineas)
13
14   f.close
15
16
17 def modification_date(filename):
18
19 #Se lee la fecha de última modificación del fichero de texto de TLE
20
21     t = os.path.getmtime(filename)
22
23     return datetime.datetime.fromtimestamp(t)
24
25
26 def TLE():
27
28 #Se actualiza el documento descargando los TLE de Celestrack si la última modificación fue
     hace más de dos días
29
30     f = open("/home/jennifer/webTracking/tleNOAA.txt","r")
31
32     time = modification_date(f.name)
33
34     actual = datetime.datetime.now()
35
36     diff = actual-time
37
38     if diff.days >= 2:
39
40         upgrade_TLE()
```

Otro *input* necesario para los propagadores es la ubicación de la estación terrena (donde se encuentra ubicada la antena de recepción). En código de radioaficionado (Código Q), la ubicación es conocida como QTH. En nuestro caso, GranaSAT tiene el siguiente QTH:

Latitud: 37.1875 (N)

Longitud: 3.7917 (W)

Altitud: 687 m

Para la aplicación web, es interesante que el usuario pueda disponer de toda la información orbital posible: época, elevación, velocidad, footprint, desviación Doppler, etc. Para ello, *Pypredict* [19] es la librería idónea. Fácilmente *Pypredict* proporciona esta información con la siguiente línea,

predict.observe(tle,qth)

donde *tle* es descargado desde *Celestrack* y *qth* es la ubicación de GranaSAT. En la siguiente Figura se puede ver la información orbital extraída de los satélites en el instante que el usuario lo solicita desde el navegador. A continuación se explicará con detalle el desarrollo web en la parte de *front-end*.

Figura 5.43 – *Información de la órbita de satélites NOAA activos mostrada en página web.*

Además, esta librería permite conocer los pases futuros del satélite sobre la *ground station* y la duración del mismo. El usuario puede disponer de esta información también a través de la aplicación web. Las siguientes líneas de código son el modo de conocer la información principal de los futuros pases: fecha, duración y elevación máxima. En la Figura 5.44 se puede ver el resultado ofrecido por el navegador cuando el usuario solicita ver los pases futuros.

———Código para extraer información de *N* número de pases futuros———

```
p = predict.transits(tle, qth)

start=[]

duration=[]

elevation=[]

for i in range(1,N):

    transit = p.next()

    start.append(transit.start)

    duration.append(transit.duration())

    elevation.append(transit.peak()['elevation'])
```

PASSES FUTUROS NOAA

NOAA15

Next Pass (UTC)	Duration	Max. Elevation(º)
'Sat 19 Aug 2017 15:51:11'	'00:07:15'	3.45
'Sat 19 Aug 2017 17:25:56'	'00:14:41'	43.95
'Sat 19 Aug 2017 19:06:31'	'00:13:26'	20.05
'Sun 20 Aug 2017 05:43:20'	'00:13:29'	19.62
'Sun 20 Aug 2017 07:22:37'	'00:14:51'	45.97
'Sun 20 Aug 2017 09:04:40'	'00:07:43'	3.93
'Sun 20 Aug 2017 17:01:35'	'00:13:50'	26.07
'Sun 20 Aug 2017 18:40:51'	'00:14:32'	34.79
'Mon 21 Aug 2017 05:19:06'	'00:11:26'	10.56
'Mon 21 Aug 2017 06:57:33'	'00:15:14'	79.98

NOAA18

Next Pass (UTC)	Duration	Max. Elevation(º)
'Sat 19 Aug 2017 16:20:24'	'00:03:23'	0.7
'Sat 19 Aug 2017 17:53:47'	'00:14:42'	29.65
'Sat 19 Aug 2017 19:34:07'	'00:15:08'	33.83
'Sun 20 Aug 2017 06:18:11'	'00:12:50'	13.98
'Sun 20 Aug 2017 07:57:52'	'00:15:40'	67.55
'Sun 20 Aug 2017 09:40:05'	'00:10:31'	8.12
'Sun 20 Aug 2017 17:42:31'	'00:14:11'	23.64
'Sun 20 Aug 2017 19:22:15'	'00:15:24'	43.51
'Sun 20 Aug 2017 21:09:18'	'00:05:04'	1.25
'Mon 21 Aug 2017 06:06:58'	'00:11:45'	10.22

NOAA19

Next Pass (UTC)	Duration	Max. Elevation(º)
'Sat 19 Aug 2017 13:04:45'	'00:09:22'	6.08
'Sat 19 Aug 2017 14:41:13'	'00:15:27'	56.24
'Sat 19 Aug 2017 16:23:16'	'00:13:13'	16.81
'Sun 20 Aug 2017 03:05:32'	'00:14:41'	28.25
'Sun 20 Aug 2017 04:46:09'	'00:14:48'	34.11
'Sun 20 Aug 2017 06:30:01'	'00:05:02'	1.67
'Sun 20 Aug 2017 12:54:18'	'00:07:46'	3.82
'Sun 20 Aug 2017 14:29:54'	'00:15:17'	43.91
'Sun 20 Aug 2017 16:11:18'	'00:14:05'	21.73
'Mon 21 Aug 2017 02:54:15'	'00:14:07'	22.08

Figura 5.44 – *Fecha y duración de los pases futuros de los satélites NOAA activos mostrado en página web.*

El script creado *'futurePass.py'* devuelve toda la información referente al satélite seleccionado, que incluye datos sobre N pases futuros (según los especificados en la función principal) y datos orbitales en el instante temporal actual.

Disponer de la información de los pases futuros del satélite no solo es interesante para el usuario de la página web, sino que es necesario para el sistema de recepción de la señal APT diseñado. Con esta información el receptor SDR se pondrá a la escucha en el instante preciso. De modo que, la librería *Pypredict* también será instalada en nuestra Raspberry Pi. El dispositivo SDR se activará cuando la elevación del satélite sobre el horizonte sea $> 0^o$, lo que indica el inicio de un pase del satélite. Asimismo, el proceso de escucha del receptor de desactiva cuando finalice el tiempo estimado de duración del satélite. En la sección 5.2.1.2 se mostrará los comandos usados para el control del receptor SDR. Para asegurar que la recepción será buena, se determinará la mínima elevación que debe tener el satélite en su pase sobre la estación terrena para que el receptor sea activado. La decisión sobre la mínima elevación aceptable se determinará en función del ancho de haz de la antena y de la ubicación de la misma, evitando zonas de sombra y procurando siempre que la antena se encuentre en

línea de visión directa con el satélite.

———Código para indicar el inicio de recepción———

```
1
2 info.append(predict.observe(noaa, qth)) #Se devuelve un objeto json
3
4 elevSTR=json.dumps(info) #Codificar JSON. La variable 'info' es un tipo de dato nativo y es
    necesario que sea String.
5
6 elevJSON = json.loads(elevSTR) #Decodificar JSON. Hace posible obtener el valor de la 'key'
    deseada.
7
8 elevation=elevJSON['elevation']
9
10 E_max.append(transit.peak()['elevation']) # Elevación máxima
11
12 if E_max>20 and elevation>0:
13
14 #Activación de receptor SDR para pases de elevación superior a 20grd
```

En la sección 5.2.2 se describirá la tarea de generación de mapas interactivos en los cuales visualizar la posición de los satélites en tiempo real. Para ello también es necesario estas librerías de propagadores. Gracias a ellas se conoce la posición del satélite en tiempo real, su propagación en instantes de tiempo futuros y anteriores y la huella sobre el globo terráqueo que ilumina el satélite (denominado *footprint*). Un indicador sobre el mapa informa en tiempo real de la posición del satélite NOAA, además, la trayectoria de propagación del satélite se dibujará sobre el mapa interactivo y se iré modificando en tiempo real.

Asimismo, el footprint del satélite será dibujado con ayuda de la librería *geometry.py* que permite la transformación de coordenadas esféricas, cartesianas y transforma también a puntos de latitud y longitud. Conocida la latitud y longitud del satélite (gracias a cualquier predictor) y dado también el radio de *footprint*, es posible determinar los puntos cardinales sobre el mapa y representarlos gráficamente.

Además de *Pypredict*, se ha probado la librería *Pyephem* [35] obteniéndose los mismos resultados, aunque esta última no da tanta información de la órbita de los satélites. Sin embargo, facilita la ejecución de la trayectoria del satélite, ya que se obtiene la posición del satélite en instantes futuros y pasados con las siguientes líneas de código:

———Propagación de tracking de satélite en el futuro———

```
1
2 import ephem
3
4 for x in range (0, N):
5
6 # N es el número de instantes futuros del vector temporal a crear
7
8     dateList.append(date_actual+datetime.timedelta(seconds=2*x))
9
10     for z in range(0,len(dateList)):
11
12       tle_rec = ephem.readtle(noaa,tle1,tle2) #Lectura de TLE con Pyephem}}
13
14       tle_rec.compute(dateList[z])
15
16       longitud = degrees(tle_rec.sublong) #Ventor de longitudes
17
18       latitud = degrees(tle_rec.sublat) #Vector de latitudes
```

5.2.1.2 Recepción con RTLSDR

Tal y como se ha mencionado en el Capítulo 4, se seleccionará el dispositivo más económico, ya que todos ellos pueden ser empleados en la recepción de satélites NOAA. En este caso es el receptor SDR basado en RTLSDR con chip RTL2832U/R820T el elegido.

A continuación se muestran las características destacadas:

- Rango de frecuencias sintonizable de 22-2200 MHz.
- No presenta ningún gap, tal y como ocurre con el chip RTL2832U/E400, que presenta un gap en la banda de 1100 MHz a 1250 MHz.
- Selección 3.2 MHz de ancho de banda máximo (2.8 MHz estable)
- 8-bit ADC
- $<$ 4.5 dB de Figura de Ruido (noise figure) del LNA
- 75 Ω ò 50 Ω de impedancia de entrada según conector de antena (SMA o TV).
- Puerto USB 2.0.

Para comenzar a trabajar con *RTLSDR* es necesario calibrar el dispositivo (ver Anexo B). Una vez se conoce su ppm es necesario aprender a controlar su funcionalidad.

Debido a que se va a trabajar con este receptor conectado a Raspberry Pi y a una antena de difícil acceso, el receptor será controlado remotamente en tiempo real mediante la ejecución de un demonio. Gracias al predictor de pases de satélites descrito en la sección 5.2.1.1, el receptor permanecerá activo únicamente durante el pase del satélite de órbita polar sobre la estación terrena.

Para la creación de este servicio, se ha utilizado la librería *'librtlsdr'* disponible en *Github* [31], como versión experimental. Los pasos de instalación están disponibles en la versión previa en [17]. El uso de una versión experimental ha subsanado los problemas que la versión anterior daba. Dichos problemas serán descritos en el Capítulo de evaluación y validación del sistema 7.

Dentro de todos los comandos de control de la librería *'librtlsdr'*, el usado para el servicio implementado es ***rtl_fm***. En [30] se puede encontrar una rápida y básica guía de uso. Para el caso que nos ocupa, en la tabla 5.1 se van a reflejar las diferentes opciones del comando que pueden ser usadas, que puede ser ejecutado desde la terminal y comenzar con la escucha.

De modo que, siguiendo con las indicaciones dadas, se analiza el comando que mejor salida genera. Para ello se ha seguido un minucioso proceso de evaluación, que será mostrado en el Capítulo 7. Cabe destacar que los resultados obtenidos con la opción -r de rtl_fm es un archivo de audio con la señal distorsionada debido a que el remuestreo no ha sido bien

Opción	Descripción
-d	Identificador del RTLSDR utilizado cuando hay varios conectados.
-M	Demodulador (por defecto Narrow FM (NFM)). Opciones: fm, wbfm, raw, am, usb, lsb
-f	frecuencia sintonizada.
-s	Ancho de banda de la señal.
-g	Ganancia en dB
-p	Corrección del error en el cristal en ppm
-E	Múltiples funciones: deemp: Filtro deemphasis wav: Generación de cabecera WAV no-dc: desactivado filtro de bloque DC offset: establecido offset de sintonización (solo para E400) ...
-r	Remuestreo (por defecto muestrea con -s)
-F	Tamaño de filtro FIR. Desactivado por defecto. Valores de 0 a 9.
-h	Ayuda

Tabla 5.1 – *Opciones para el comando rtl_fm*

ejecutado, por ello se evitará el uso de esta funcionalidad, que además está en etapa de pruebas. Por otro lado, si queremos que el comando rtl_fm de como salida un fichero .wav, se obtiene un archivo con un error en la cabecera, aunque puede ser procesado sin problema. Dado el estudio realizado y los problemas existentes con la función de remuestreo, se utiliza rtl_fm junto con el proceso SOX.

SOX es una librería disponible para Linux y compilada para Windows y MacOS. Puede ser descargada y consultada en profundidad en [13]. SOX es una librería de procesado de audio y video. Permite leer y cambiar el formato de archivos (RAW, WAV, MP2/MP3, AVI, MP4, MPEG, etc) además de remuestrear, filtrar, producir multitud de efectos, modificar volumen, etc. Dicho esto, a continuación se muestra los comandos usados en la recepción de cada uno de los satéliles NOAA:

rtl_fm –d 0 –M fm –f 137.62M –s 60k -p 62 -g 40 -E deemp -E wav -F 9 - | sox -t wav - static/Audios/NOAA15.wav rate 11025

rtl_fm –d 0 –M fm –f 137.9125M –s 60k -p 62 -g 40 -E deemp -E wav -F 9 - | sox -t wav - static/Audios/NOAA18.wav rate 11025

rtl_fm –d 0 –M fm –f 137.1M –s 60k -p 62 -g 40 -E deemp -E wav -F 9 - | sox -t wav - static/Audios/NOAA19.wav rate 11025

Se selecciona un ancho de banda de BW=60 kHz, superior al ancho de banda de la señal APT de 40 kHz, según la ecuación 4.1.1. Debido al efecto Doppler, la señal se desplazará en frecuencia. La corrección doppler se efectuará en post-procesado, por lo que se toma un

mayor ancho de banda al necesario.

Además de esto, se indica que el ppm calculado en calibración ha sido 62ppm y se indica una ganancia de 40dB. Este parámetro es muy delicado y habrá que hacer un ajuste en sucesivos pases hasta encontrar el ideal, ya que una ganancia demasiado elevada podría aumentar demasiado el ruido de suelo, produciendo que la señal APT quede enmascarada. Se ha comprobado que con las opciones -E deemp y -F 9 la recepción mejora levemente (estas funciones no son 100% necesarias). Por último, el audio es remuestreado usando la librería SOX, que crea una pista .wav sin error en la cabecera como ocurre sin el uso de SOX.

A continuación se describe los métodos llevados a cabo para el uso de *RTLSDR* en el servicio de recepción de APT.

RTL_FM corriendo en Raspberry PI 3B

Se creará un script en Python y se daemoniza con PM2 [11] (ver características en Anexo D). Pese a ser un gestor de procesos diseñado para Node.js, puede utilizarse para otros lenguajes, ofreciendo buenos resultados y una cómoda gestión del servicio.

En primer lugar, es necesario identificar el instante en el que se produce el pase futuro. En este caso, con la función creada *'futurePass.py'*, se obtiene como salida la fecha de inicio de un nuevo pase, la duración de este, la máxima elevación que alcanza el satélite en dicho pase y más información orbital del satélite. Entre la información orbital que nos interesa se encuentra la elevación en tiempo real del satélite. Como se vio en la sección previa 5.2.1.1, la elevación real del satélite en tiempo real será el indicador de la activación del receptor *RTLSDR*.

Las siguientes líneas indican la activación del dispositivo y el inicio de la captura del audio.

```
1
2  import subprocess
3
4  import os
5
6
7  if E[0][0]>14 and elevation_19 > 0:
8
9        print 'Pase NOAA19. Grabando...'
10
11       t=duration19.split(":")
12
13       durat=int(t[1])*60+int(t[2])
14
15       durat=str(durat)
16
17       arg1='timeout ' +durat+ 's rtl_fm -f 137.1M -s 60k -p 62 -g 40 -E deemp -E wav -F 9 -'
18
19       arg2='sox','-t','wav','-', 'static/Audios/NOAA19.wav','rate','11025'
20
21       argu_input=subprocess.Popen(arg1.split(" "),stdout=subprocess.PIPE)
22
23       argu_input2=subprocess.Popen(arg2,stdin=argu_input.stdout, stdout=subprocess.PIPE)
24
25       argu_input.stdout.close()
26
27       argu_input2.communicate()[0]
28
29       print 'Se salio de SDR'
```

El dispositivo permanecerá a la escucha durante el pase del satélite. El proceso creado con el módulo *'subprocess'* de Python corre de manera independiente y puede tener concurrencia con otros procesos sin problema, igual que si de un proceso iniciado desde el sistema se tratara. Esta sucesión de métodos es la forma de reemplazar el *pipeline* posible desde línea de comandos (|). El método *subprocess.Popen()* ejecuta la orden introducida como primer argumento y captura la salida del sistema almacenándola en el argumento *stdout*, o bien envía información al sistema con el argumento *stdin*, como ocurre con el segundo proceso creado, pasando la salida de *rtl_fm* como entrada a *sox*. El valor de estos argumentos debe ser un archivo o tubería, por lo que usa *subprocess.pipe*, para indicar que un archivo o tubería va a ser abierto. Es importante *arg_input.stdout.close()* después de empezar el segundo proceso para indicar si la tubería está rota, por ejemplo cuando el segundo proceso existe previamente y la salida de *rtl_fm* no será leída. El método *Popen.comunicate* interactúa con los procesos, leyendo datos y espera hasta que el proceso termine, sin realizar otra operación del servio. En este caso, se ha determinado el final del proceso con la función de sistema *'timeout'*, que indica un tiempo para que el proceso sea interrumpido y finalizado por completo. Este tiempo de *'timeout'* es la duración del pase del satélite.

Es esquema de recepción básica explicado se puede resumir observando la Figura 5.45.

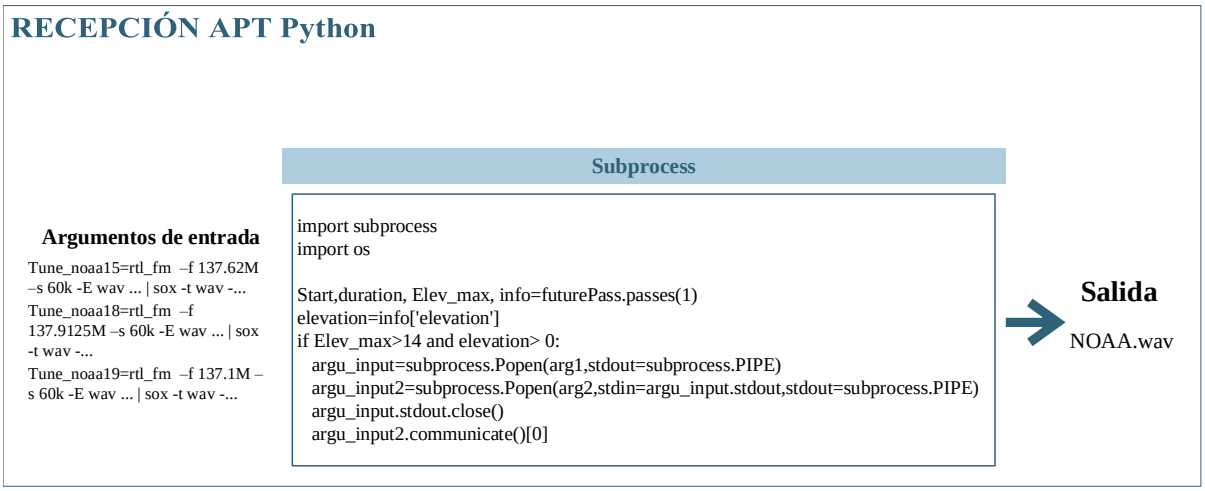


Figura 5.45 – *Recepción de APT básica utilizando el método subprocess.*

Es muy importante que el proceso rt_fm sea finalizado correctamente, ya que no se podrá acceder a otra frecuencia con el dispositivo RTL2832. Como se va a utilizar un solo receptor

SDR, es necesario parar el proceso para iniciar otro nuevo cuando se produzca el pase de un satélite NOAA diferente. Podría usarse la función *Popen.kill()* para asegurar la correcta finalización.

Corrección Doppler

Hasta aquí todo estaría preparado para automatizar la recepción usando Python, sin embargo surge un gran inconveniente que se debe solventar y que ya se ha dejado ver. *Rtl_fm* no permite su modificación una vez el proceso se ha puesto en marcha, de modo que, la corrección doppler moviendo la sintonización de frecuencia en RTLSDR, que es necesaria debido al movimiento del satélite, no podría ser realizada. La solución no es trivial, pero se han encontrado diversas soluciones que se pueden implementar y trabajar en armonía con el servicio en Python [15] [8].

En primer lugar, existe una librería implementada en el lenguaje de programación *Rust* que añade al comando *rtl_fm* la funcionalidad de realizar el seguimiento del efecto doppler, ya que se conecta con la librería *libgpredict*, la cual habría que instalar en la Raspberry Pi o modificar el código fuente *Rust* para que trabaje con la librería de propagador ya implementada. *Libgpredict* es usada en el software Gpredict y sigue el algoritmo SGP4 con el que también trabaja *PyPredict*. Ofrece la posibilidad de corregir el efecto doppler en tiempo real y en post-procesado [15], pero siempre en archivos I/Q. Debido a la dificultad de incorporar un nuevo lenguaje en el aprendizaje y desarrollo de dicho proyecto se descarta ese método.

Por otro lado, resulta interesante poder modificar la frecuencia de recepción para adaptarse a la frecuencia con la desviación por el efecto doppler. Esta frecuencia viene dada por la siguiente ecuación:

$$f = \frac{c + v_r}{c + v_s} f_0 \tag{5.2.1}$$

f es la frecuencia observada, f_0 es la frecuencia downlink del satélite, c es la velocidad de la onda en el medio, v_r es la velocidad relativa del receptor en relación al medio (en nuestro caso se considera fijo) y v_s es la velocidad de la fuente con respecto al medio (positiva se la fuente se aleja al receptor y negativa si se acerca). De modo que la frecuencia aumenta cuando emisor y receptor se acercan entre sí y se reduce si se alejan. El predictor empleado da la desviación doppler a los 100 MHz en tiempo real, de manera que sabiendo la frecuencia downlink del satélite y podemos obtener la frecuencia observada como $f[MHz] = f_0 + freq_{shift}\frac{f_0}{100}$.

El código ofrecido en [8] consigue controlar en tiempo real la frecuencia de recepción del proceso generado por *rtl_fm*. Este código consiste en una aplicación web sencilla mediante la cual el cliente puede modificar la frecuencia de sintonía del SDR, su ganancia y su modulación. Lo que implementa este código es un intérprete en Python de la librería en C *rtl_fm*. Las funciones que nos interesarían en nuestro caso son las siguientes:

rtl_fm_setup_and_go(input_args)

get_frequency()

set_frequency(new_freq)

La función *rtl_fm_setup_and_go(input_args)* inicializa el dispositivo SDR y los argumentos de entrada son las opciones del comando *rtl_fm* vistas, pero sin el proceso sox en *pipeline*, sino que la captura de audio con *rtl_fm* será con fichro de tipo RAW, para poder modificar la frecuencia central RF en tiempo real. Con *get_frequency()* se obtiene la frecuencia a la que el dispositivo está sintonizado y la función *set_frequency(new_freq)* es la usada para modificar la frecuencia de sintonía del dispositivo. De esta manera, como disponemos del valor *Doppler Shift* a 100 MHz, (obtenido con PyPredict) podemos modificar la frecuencia automáticamente durante el seguimiento.

Es necesario ejecutar un hilo (*thread*) para controlar el proceso desde dentro y poder, en este caso, modificar la frecuencia según el efecto doppler. La librería disponible en [8], de la que se parte, utiliza el módulo *threading* de Python [29] para crear con la clase *Thread()* un hilo en ejecución. Si se tienen varios hilos, todos ellos compartirán recursos y esto va a dar problemas en nuestro caso.

Para la aplicación desarrollada, se necesita que se ejecute un proceso independiente para *rtl_fm*, ya que se necesita poder parar y ejecutar el proceso *rtl_fm* para cada pase de los satélites NOAA. El módulo *threading* no es útil en nuestro caso, ya que no es posible interrumpirlo sin que caiga todo el servicio. Para dar solución a esto, existe un módulo con una API similar a *threading*. Se trata de *multiprocessing* de Python [28], que permite crear múltiples procesos a nivel de sistema operativo. En este caso, se crea un proceso con memoria independiente para *rtl_fm*, por lo que será posible pararlo como a cualquier otro proceso del sistema (usando la función *kill*). Se puede ver el esquema en la Figura 5.46 para realizar la recepción de APT con corrección doppler en RF, sin esperar a post-procesado.

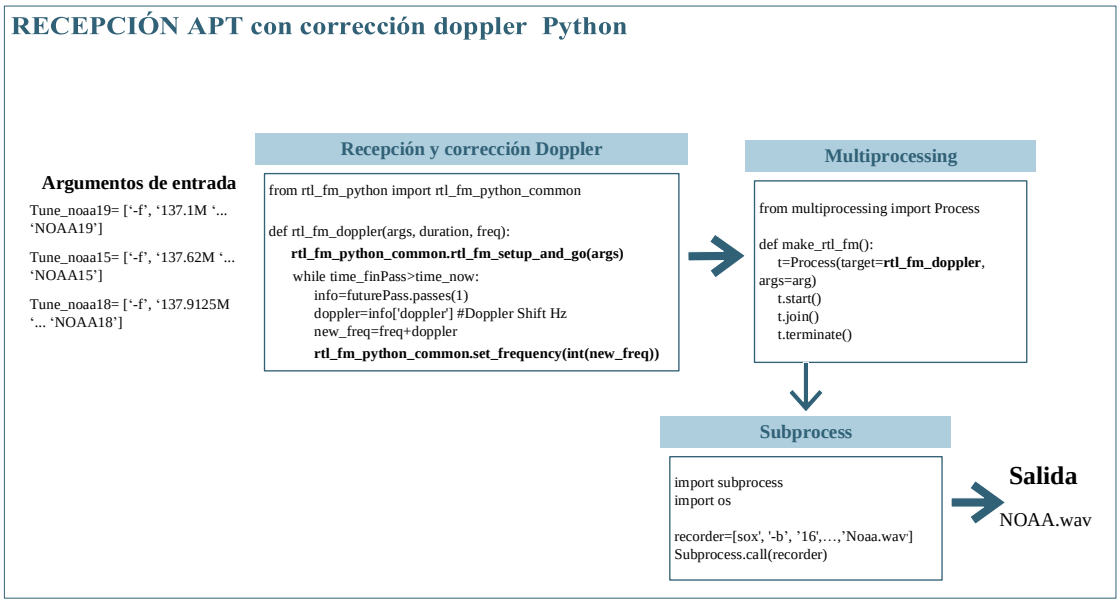


Figura 5.46 – *Recepción con rtl_fm usando interprete Python de [8] para corrección doppler en RF.*

Los pasos a seguir para recibir el APT usando multiprocessing son los siguientes:

- Multiprocessing lanza un proceso con la clase *Process()*.
- Con *'target='* se indica la función que va a ser ejecutada. En este caso se ejecuta *rtl_fm* interpretado en Python para la corrección de la frecuencia sintonizada, corrigiendo el desplazamiento doppler. Esta corrección se puede realizar porque conocemos la desviación de frecuencia doppler (*Doppler Shift*) gracias al propagador *Pypredict*.
- En *'args='* se indican los argumentos de la función, que son las opciones propias de *rtl_fm*.
- Se llama al método *start()* para iniciar la ejecución y *join()* espera a la finalización, que será cuando el satélite finalice su pase.
- Es necesario eliminar el proceso abierto para que en un próximo pase el dispositivo SDR se pueda poner a la escucha creando un nuevo fichero de audio. Esto se consigue con *terminate()* o con *kill(id proceso)*. Una vez el proceso ha finalizado, los datos almacenados en crudo se convierten a .wav con la librería SOX y el módulo *subprocess*. La función *call()* tiene la misma utilidad que *Popen()*. En este caso no se emplea la opción -E wav de *rtl_fm*, sino que se graba y modifica la frecuencia de recepción con los datos en crudo.

Dado que este tipo de corrección doppler ha dado fallos en cuanto a cortes en la recepción de audio debido a la sintonización de la frecuencia en tiempo real. Además, el código desde el que se ha partido, descargado en Github [8], desarrolla un interprete en Python de la versión antigua de *rtl_fm*. Esta versión antigua, disponible en *librtlsdr*, da sucesivos errores que se verán en el Capítulo 7 de evaluación y validación del sistema. El error fundamental es que no está indicado para pequeños anchos de banda, produciendo distorsión en la señal de recepción. Además, se ha probado que la modificación de la ganancia del receptor, también produce distorsión en la señal. Claramente esta librería tiene errores de implementación.

Por ello, finalmente se ha decidido **corregir el efecto doppler en post-procesado**, como se verá en la siguiente sección.

5.2.1.3 Decodificación de datos APT

En esta sección se explica el modo de realizar la decodificación de la señal APT para así extraer la imagen meteorológica y las herramientas usadas para integrar este proceso en el software diseñado. En la Figura 5.47 se puede ver el esquema lógico para decodificar la señal APT, utilizando la información extraída en el Capítulo 4. Se parte de la señal demodulada en FM y convertida a digital, obtenida como salida del receptor SDR.

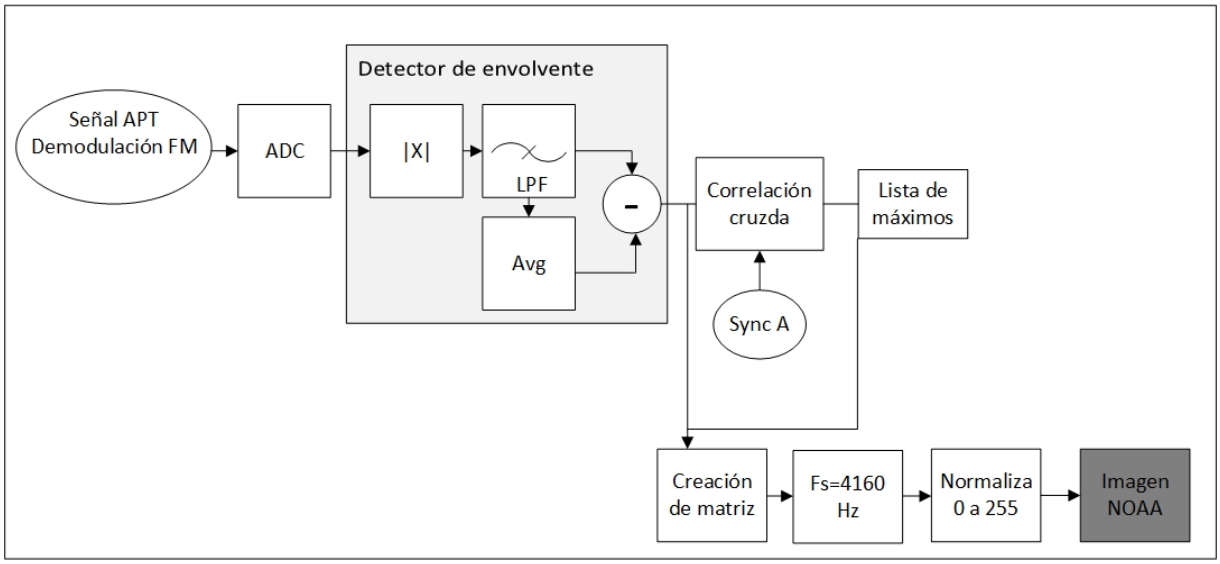


Figura 5.47 – *Esquema de decodificación de señal APT a partir de la señal de sincronización del canal A (visible).*

El primer paso es realizar la demodulación de la subportadora AM. Para ello se usa un detector de envolvente. Este empieza por tomar el valor absoluto de la señal, que actúa como un rectificador; la señal rectificada pasa por un filtro paso bajo, usado para suavizar el rizado de la señal rectificada. La media de la señal filtrada es substraída a la salida directa del filtro pra eliminar los componentes DC de la señal filtrada. Se debe tener especial cuidado con la

frecuencia de corte del filtro paso bajo, ya que una frecuencia demasiado alta no eliminará el rizado. Por lo que, la frecuencia de corte del filtro debe ser ligeramente superior a la frecuencia de la señal (2.4 kHz).

El siguiente paso es obtener la información de la sincronización de uno de los canales. Por ejemplo, se ha tomado la señal de sincronización del canal A, que corresponde al canal del espectro visible. Según se pudo ver en la Figura 4.2, la señal *SyncA* es una onda cuadrada de 1040 Hz, formada por 7 ciclos, con una longitud de 39 palabras. Se realiza la correlación cruzada de la señal demodulada y la señal de sincronización. El máximo punto de la correlación cruzada indica el instante inicial de sincronización. Se obtiene una lista de los puntos máximos que indican el inicio de cada línea de la imagen (que está formada en total por 2080 palabras). Estos puntos máximos están equidistanciados 0.5 s, coincidiendo con la duración temporal de la línea (ver Figura 4.2). Además, se observa un segundo pico dominante alos 0.25s, que muestra que hay una significativa correlación entre el pulso de sincronización A y B. En la Figura 5.48 se puede ver un ejemplo de la correlación cruzada entre *SyncA* y la señal demodulada.

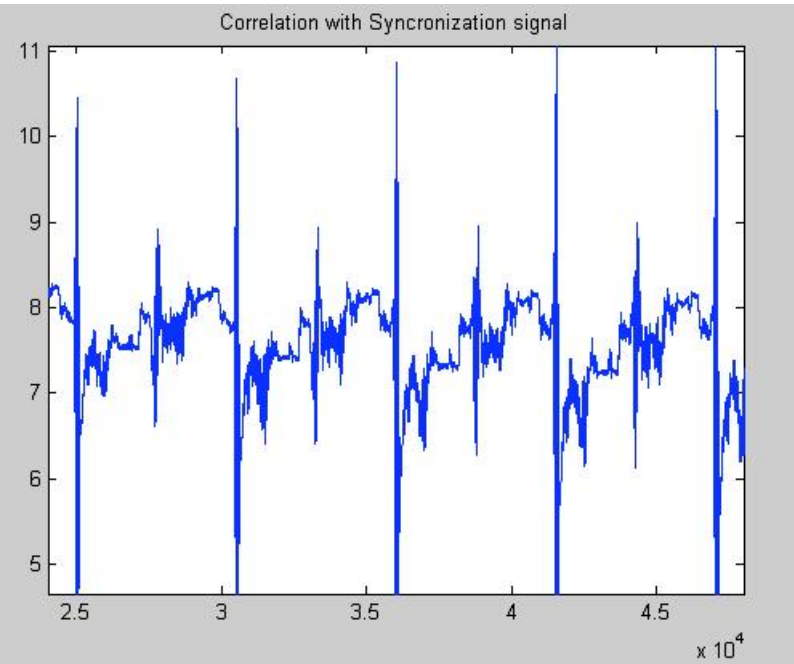


Figura 5.48 – *Gráfica de la correlación cruzada [33].*

Con la lista de puntos máximos estraídos de la correlación cruzada, se puede dividir en secciones la señal demodulada de manera que se conforma la matriz usada para extraer la imagen. Seguidamente se remuestrea los datos en cada fila, para así obtener el número correcto de puntos en cada línea que forma la imagen. La nueva frecuencia de muestreo es 4160Hz y cada línea tiene una duración de 0.5s. Para finalizar, se normaliza la matriz para estar en el rango de 0 a 255 y extraer la imagen resultante.

Una vez entendida la decodificación de la señal y procesado en MATLAB, se estudia el mejor modo de integrar en nuestro sistema.

Tras una búsqueda y estudio de diferentes opciones, se ha optado por el uso de la librería WXtoImg. Este software de decodificación de señales satelitales ha sido probado en Windows, junto con Orbitron y SDRSharp. Orbitron funciona como driver de control de la frecuencia central de los satélites en tiempo real para SDRSharp, sincronizando la escucha del satélite con el movimiento del mismo, corrigiendo las desviaciones por el efecto doppler en RF.

WXtoImg también está disponible para Linux, por lo que puede ser ejecutado en Raspberry Pi. Este software, además de decodificar en tiempo real, como se ha venido empleando en Windows en diferentes pruebas y etapas de validación, puede decodificar archivos de audio WAV. Para ello, únicamente necesita conocer la ubicación de la estación terrena, su QTH, y el inicio del pase del satélite y realiza la decodificación, además de realizar post-procesado para una corrección doppler. WXtoImg está disponible en [14]. A continuación se muestra su sencilla instalación:

pi@raspberrypi: wget http://www.wxtoimg.com/beta/wxtoimg-armhf-2.11.2-beta.deb

pi@raspberrypi: sudo dpkg -i wxtoimg-armhf-2.11.2-beta.deb

pi@raspberrypi: sudo apt-get install alsamixergui

Se debe crear directorio /.wxtoimgrc con: Latitude:37.1875, Longitude:-3.7917, Altitude:687

pi@raspberrypi:/usr/local/bin $ xwxtoimg &

pi@raspberrypi:/usr/local/bin $ wxtoimg

Se aceptan condiciones generales que aparecen al ejecutar la anterior línea de comando. Usando la siguiente línea de comando suficiente para una imagen meteorológica decodificada (ver Figura 5.49. Este software permite mejorar la imagen con diferentes técnicas de procesado de imagen, permitiendo pintar sobre la imagen. En la Figura 5.49 se puede ver el comando utilizado.

```
pi@raspberrypi:~ $ /usr/local/bin/wxtoimg ~/201706101703.wav 201706101803.png
Satellite: NOAA
Status: signal processing.............................
Gain: 14.7
Channel A: 1 (visible)
Channel B: 4 (thermal infrared)
```

Figura 5.49 – *Decodificación de APT con WXtoImg.*

Como se ha mencionado, la decodificación será realizada en Raspberry Pi, corriendo en el servicio principal para recepción de satélites. Para ello, el script de Python que activa

el receptor en el instante del pase del satélite incluirá un nuevo proceso con *subprocess*, llamando al decodificador. Posteriormente, la imagen obtenida será transferida al Servidor Web Linux de GranaSAT. Para dicha transferencia se ha empleado el protocolo Secure Copy Protocol (SCP).

En la siguiente Figura se observa la implementación realizada, para finalizar el receptor de APT. Una vez recibida la señal y creado el archivo WAV, este pasa a ser la entrada del decodificador y la salida es la imagen meteorologica, que es enviada al servidor central de GranaSAT.

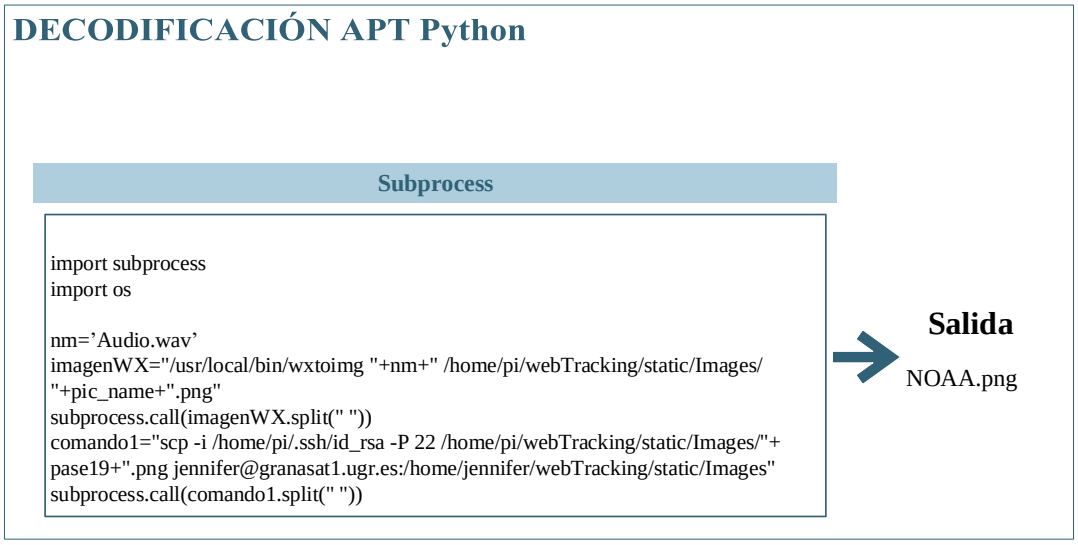


Figura 5.50 – *Esquema de decodificación de APT con WXtoImg y Subprocess.*

5.2.1.4 Base de datos de imágenes decodificadas

Como ya se ha mencionado, las imágenes meteorológicas decodificadas por el sistema receptor que usa Raspberry Pi son enviadas al servidor web. El servidor web dispone de un directorio propiamente para almacenar estas imágenes. Al tratarse de una cantidad de datos que puede ser fácilmente controlada, no ha sido necesario emplear ningún administrador de base de datos como *MySQL*. En un primer momento se comenzó a trabajar con *MySQL* dada la facilidad con la que los datos puedes ser extraídos, almacenados, ordenados, usando el lenguaje *SQL*. Finalmente, ha sido suficiente con gestionar una carpeta donde irán dirigidas todas las imágenes.

Para la transferencia de las imágenes meteorológicas al servidor web se ha empleado el protocolo SCP. El servidor web pedirá la clave de acceso para autorizar la transferencia de información al equipo. Debido a que este proceso debe ser remoto, no se puede introducir la

clave de acceso cada vez que se realice el envío de una imagen. Para que el proceso sea seguro y pueda ser realizado de forma autónoma, se crea un identificador (ID) para una conexión entre raspberry y servidor web sin credenciales:

(raspberrypi)$ ssh -keygen -t rsa - Se crean los archivos id_rsa e id_rsa.pub en el directorio */.ssh*. Seguidamente se copia en el servidor Linux el fichero id_rsa.pub, que será la clave pública para el acceso de nuestra raspberry.

Ahora, desde el servidor:

(GranaSAT)$ cd .ssh

(GranaSAT)$ cat id_rsa.pub » authorized_keys - Se copia el identificador al archivo de claves autorizadas en ssh.

(GranaSAT)$ chmod 600 authorized_keys - Se modifican los permisos del fichero.

De este modo ya se puede realizar la transferencia de imágenes sin que pida credenciales en cada proceso:

(raspberrypi)$ ssh -i /.ssh/id_rsa -P 3389 jennifer@granasat1.ugr.es - Acceso ssh sin credenciales

(raspberrypi)$ scp -i /.ssh/id_rsa -P 3389 /webTracking/static/Images/NOAA.png jennifer@granasat1.ugr.es:/home/jennifer/webTracking/static/Images - Envío de imagen por SCP sin credenciales.

5

Cuando desde el navegador se solicita ver las imágenes meteorológicas, el servidor envía la lista de nombres de archivos disponibles. Para ello, una simple función recorre el directorio y almacena el nombre de los ficheros en una variable del siguiente modo:

```
1
2  ListIMG=os.walk(pathIMG)
3
4  lst=[]
5
6  for root, dirs, files in ListIMG:
7
8    for fichero in files:
9
10        (nombreFichero, extension) = os.path.splitext(fichero)
11
12        lst.append(nombreFichero + extension)
13
14   lst=sorted(lst, reverse = True)
```

La última línea indica que la lista de ficheros es ordenada por por orden alfabético en sentido inverso. Hay que destacar que los ficheros han sido nombrados en el momento de su creación con el nombre del satélite seguido de la fecha completa del inicio del pase. Un ejemplo sería: *NOAA15_20170820153145*

.

Esta lista es enviada al navegador como un tipo JSON (JavaScript Object Notation). Es

el navegador el encargado de interpretar la lista de nombres de archivos para mostrarlas al usuario y permitir la descarga de la imagen. Esto se explicará en 5.2.2.2

5.2.2 Front-end

La interfaz con el usuario será desarrollada en Javascript, combinado con HTML y CSS. Dado que el front-end no será excesivamente complicado en cuanto a diseño web, se ha decidido no usar ningún gestor de documentos CMS e implementar el diseño puramente con HTML y CSS.

Para que la aplicación corra en el navegador, se activa del siguiente modo desde el servidor:

```
1
2 if __name__ == "__main__":
3
4   app.run(host='0.0.0.0',port=8004,debug=True)
```

La aplicación está corriendo en el puerto 8004. Una vez el servicio esté activo, podrá ser utilizado por el usuario desde `http://granasat1.ugr.es:8004/map`, que es su página principal correspondiente a la visualización del *tracking* de los satélites NOAA.

Existen cuatro ventanas que podrán ser abiertas desde el navegador. En primer lugar está, como ya se ha mencionado, la ventana principal, que permite al usuario ver la posición sobre un mapa interactivos de los tres satélites NOAA en tiempo real. En la siguiente sección se describe la implementación del script *'map.html'*. Las siguientes líneas de código muestran la comunicación entre navegador y servidor para responder a la petición del cliente de `http://granasat1.ugr.es:8004/map`.

```
1
2
3   @app.route("/map")
4
5   def map():
6
7       return render\_template('map.html')
```

En esta ventana principal nos encontramos con un MENÚ que redirecciona a otros servicios. El usuario tiene acceso a información sobre los pases futuros de los satélites NOAA-15, NOAA-18 y NOAA-19, información orbital actual de dichos satélites y a la descarga de las imágenes meteorológicas obtenidas por nuestro sistema de recepción situado en GranaSAT

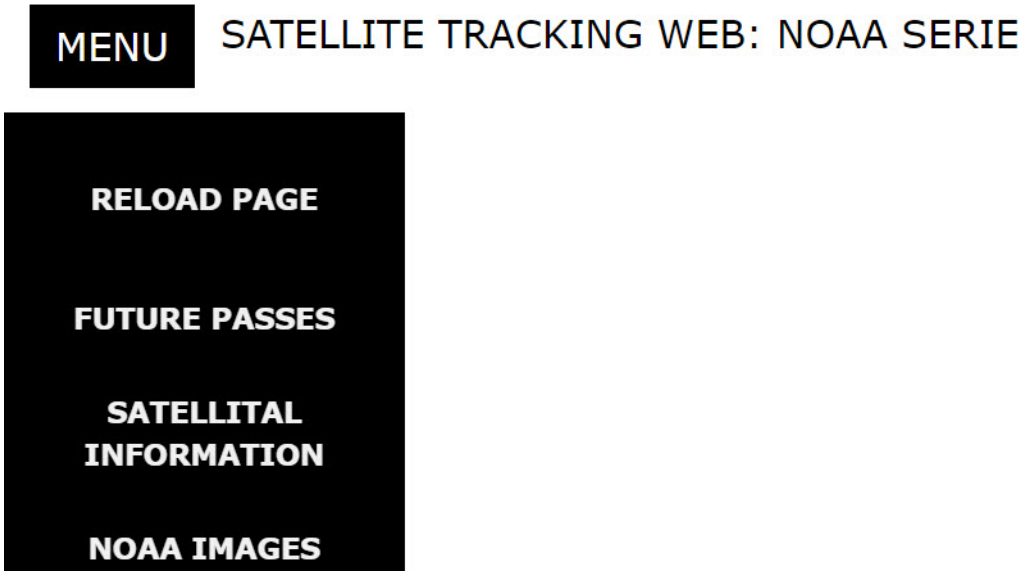


Figura 5.51 – *Redirección a diferentes ventanas desde página principal de aplicación web.*

A continuación se muestra las líneas de código en Python para la comunicación de navegador-servidor para responder a la solicitud de las diferentes aplicaciones.

```
1
2  @app.route("/tbld")
3
4  def tbld():
5
6    name,start,duration, E, info=passes(11)
7
8    return render_template('tbld.html',
9
10                 name=name,
11
12                 pass_start=start,
13
14                 pass_duration=duration,
15
16                 pass_E=E)
17
18
19 @app.route("/index")
20
21 def index():
22
23   name,start,duration, E, info=passes(11)
24
25   json.encoder.FLOAT_REPR = lambda f: ("\%.2f" \% f)
26
27   json.dumps(info)
28
29   return render_template('index.html',
30
31                 info=info,
32
33                 name=name)
34
35
36 @app.route("/images")
37
38 def images():
39
40   return render_template('images.html')
```

En las siguientes secciones se muestra la información correspondiente a cada una de las interfaces a las que el cliente puede acceder.

5.2.2.1 Mapas interactivos

El cliente tiene acceso a la aplicación web creada a través de la URL `http://granasat1.ugr.es:8004/map`. Desde ella puede visualizar la órbita de los tres satélites NOAA sobre un mapa interactivo, en el cual se puede ver además de la ubicación actual del satélite, la zona de cobertura que abarca (conocido como *footprint* del satélite) y la propagación en instantes futuros y anteriores del *'tracking'* satelital.

Estos datos son facilitados por el servidor haciendo uso de la librería *PyPredic* [19] mencionada en la sección 5.2.1.1.

Para la generación del mapa interactivo en el que se muestra el seguimiento de los satélites en tiempo real, se ha utilizado la librería *Leaflet* [24], una librería de código libre para Javascript, que permite el desarrollo de diferentes características sobre mapas interactivos según las necesidades. Los mapas usados en la interfaz del cliente han sido obtenidos del servidor *OpenStreetMap* [3], que utiliza fuentes de datos libres y de *Mapbox* [1], una plataforma para edición de mapas y creación de aplicaciones web con mapas interactivos. A continuación se muestra la creación de la variable que crea el mapa interactivo. Con *Leaflet* se usa el método *'L.tileLayer()'*, para la creación de una capa del mapa elegido de cualquiera de los servidores, *OpenStreetMap* por ejemplo, y con la función *L.Map* se crea el mapa. Un mapa puede tener diferentes capas, es decir, diferentes estilos de mapa tomaos del servidor.

```
1
2       var map_sat = L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', {
3
4         attribution: '&copy; <a href="http://openstreetmap.org/copyright">OpenStreetMap</a>',
5          maxZoom: 18});
6
7
8       var map = new L.Map('mapa_div',{layers: map_sat, minZoom: 2}).setView([0,0],2);
```

Sobre estos mapas creados se coloca la ubicación de los satélites. Así pues, desde el servidor se envía a cliente la ubicación de los satélites mediante diccionarios JSON (JavaScript Object Notation), basados en una palabra clave y su valor. Estos datos son recuperados por el cliente mediante el método *$get()* de la librería jQuery de Javascript. Esta información se actualiza en el tiempo deseado en el navegador, de modo que la información visualizado por el usuario será prácticamente en tiempo real. Por ejemplo, siendo más detallistas, para enviar la posición actual de los satélites y ser leída mediante jQuery, el proceso realizado es en siguiente:

1. En el servidor se enviará la ubicación de los satélites en un vector (latitud y longitud en grados) en una variable *'param_satellites*, el nombre de los satélites operativos y su frecuencia *download*, para ser visualizado en el mapa. Esta información es enviada en un diccionario JSON (JavaScript Object Notation):

```
1
2 @app.route("/paramSAT")
3
4 def paramSAT():
5
6   param_satellites= position_satellites()
7
8   freq, version, ope_NOAA = load_setting()
9
```

```
10   return jsonify({'datos':param_satellites,
11                   'frequency': freq,
12                  'ope_NOAA': ope_NOAA)}
```

2. Se recoge la información en front-end usando jQuery:

```
1 \$.get("/paramSAT", function(data){
2
3         sat = data['datos']
4
5         freq=data['frequency']
6
7         operative_NOAA=data['ope_NOAA'];
8
9        ...
```

3. Con esta información se crea sobre el mapa unos marcadores que posicionan los satélites, ya que se conoce su localización.

```
1
2 for (var i = 0; i < sat.length ; i++) {
3
4        marker =  new L.marker([sat[i][1], sat[i][2]],{title: sat[i][0],}).addTo(map);
```

4. Los datos referentes a la ubicación del satélite son pedidos al servidor usando peticiones GET cada 1000 ms para actualizarlos, usando el método *setInterval()* de jQuery. Se crea una función que modifique los valores de los marcadores creados, y es esta la que será actualizada cada 1000ms. El método de la librería *Leaflet* que realiza el cambio de la localización es *'setLatLng()'*.

```
1
2 var getData = function(){
3
4      \$.get("/paramSAT", function(data){
5
6         sat = data['datos']
7
8         for (var i = 0; i < arrayMarker.length; i++) {
9
10          arrayMarker[i].setLatLng([sat[i][1], sat[i][2]])
11
12        };
13
14       })
15
16     }
17
18
19 setInterval(function(){
20
21       getData()
22
23     },1000
```

De este modo, teniendo los datos y usando las funcionalidades que ofrece la librería *Leaflet*, se genera sobre el mapa interactivo la posición de los satélites.

Para mostrar la propagación del satélite y su footprint se sigue el mismo proceso para recuperar los datos del servidor y se usa la función *new L.polyline(linea[i],color: Color[i]).addTo(map)*, que permite dibuja una línea sobre el mapa según el vector de latitudes y longitudes enviado por el propagador.

En la figura 5.52 se muestra la página principal a la que tiene acceso el usuario. Se han creado tres mapas. En el mayor se observa la posición de la estación terrena GranaSAT y los tres satélites. Haciendo click sobre los satélites se despliega una ventana de información, que da los datos de su posición. Esto se consigue con diferentes funciones ofreciadas por *Leaflet*.

Pasando el ratón sobre cada uno de los indicadores se observa de qué satélite se trata. Los otros tres mapas son un zoom a la posición del satélite.

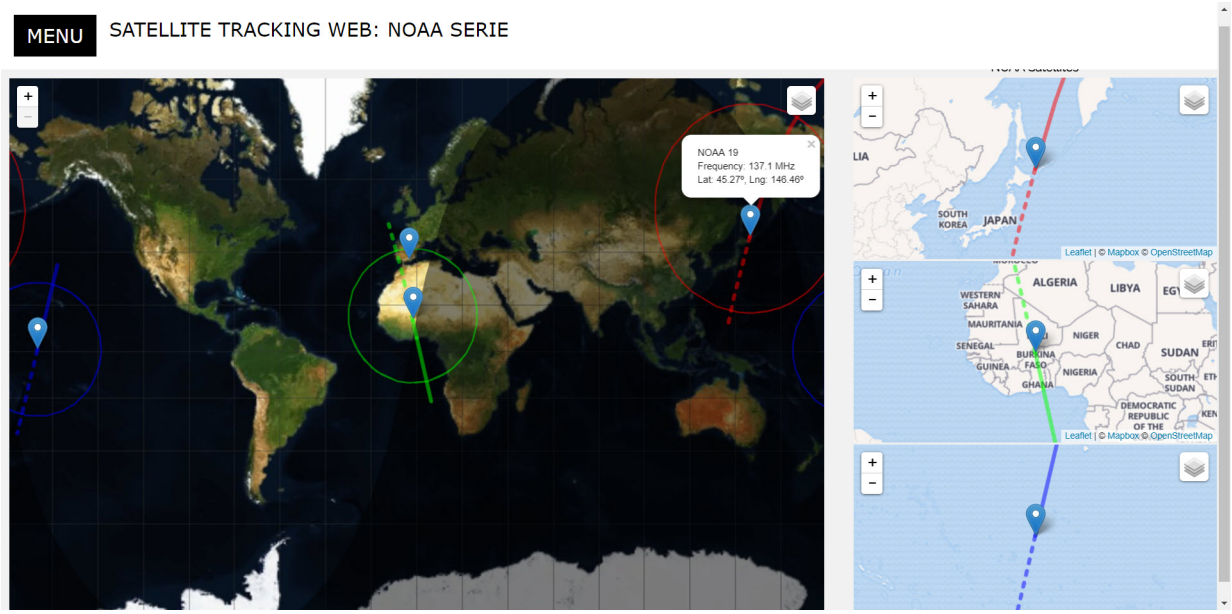


Figura 5.52 – *Página principal de aplicación web desarrollada.*

Como ya se ha mencionado, la zona de cobertura del satélite también es mostrada sobre el mapa interactivo, tal y como se ve en la figura anterior, de modo que el cliente podrá visualizar cuando alguno de los satélites pasa sobre la estación terrena de GranaSAT. Cuando la elevación del satélite sobre la estación terrena es $>0^{o}$, GranaSAT se encuentra dentro de la zona de footprint dibujada (visto en la Figura 5.52.

Sobre el mapa interactivo de mayor tamaño se ha dibujado la zona de sombra de la tierra, conocida como *'terminator'* o *'twillight zone'*, que es una línea en movimiento que separa la zona iluminada por el día y la zona de noche. Y además, se ha insertado una capa que indica las coordenadas terrestres.

5.2.2.2 Descarga de imágenes

Una parte fundamental del proyecto es que el usuario final tenga acceso a las imágenes que se están recibiendo desde la estación terrena. Para ello la aplicación web dispondrá de la opción de la descarga de las imágenes meteorológicas almacenadas en la base de datos del servidor web.

Los nombres de las imágenes guardadas son recogidos con jQuery cómodamente gracias a que son enviados desde el servidor usando tipo de dato JSON (JavaScript Object Notation). Una vez recogidos los nombres de los archivos almacenados, de forma dinámica se genera mediante Javascript elementos HTML, con la función *document.createElement()*,

que permitan que el navegador muestre la imagen y permita a la vez la descarga. Esto se realiza del siguiente modo:

```
1
2  var url="../static/Images/"+imagen_ruta; //'imagen_ruta' es el nombre de la imagen guardada
3  var desc=document.createElement("a") // Elemento 'a' crea rutas de acceso
4  desc.download=imagen_ruta  //Atributo que permite descarga directa.
5  desc.href=url;  // Atributo para introducir url.
6  var elem = document.createElement("img");  // Elemento HTML 'img' muestra imagen.
7  elem.src=url;  // Atributo para tomar la url de ubicación de la imagen.
8  elem.setAttribute("style", "width:60\%");
9  elem.setAttribute("TITLE",date19[z]);  // Con el atributo TITLE se muestra el nombre de la
     imagen al colocar el cursor sobre ella.
10 elem.setAttribute("onclick", "onClick(this)");  // Se crea función que permite ver la imagen
     en toda la pantalla del ordenador al hacer click sobre ella.
11 elem.alt=date19[z]; // Con el atributo 'alt' se muestra el nombre de la imagen ampliada
12 desc.appendChild(elem);  // Es necesario indicar la relación entre elementos HTML creados.
13 document.getElementById("tabla3").appendChild(desc); // Se indexan los elementos creados en
     el contenedor correspondiente creado con HTML.
```

Así pues, se consigue visualizar las imágenes meteorológicas del siguiente modo:

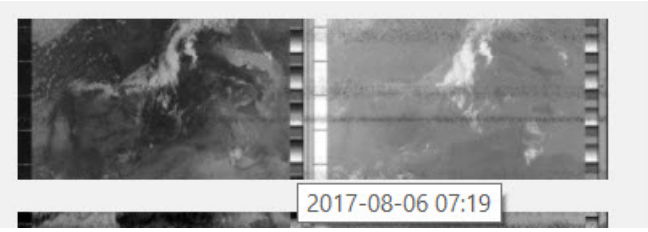


Figura 5.53 – *Imagen mostrada en aplicación web con cursor sobre ella.*

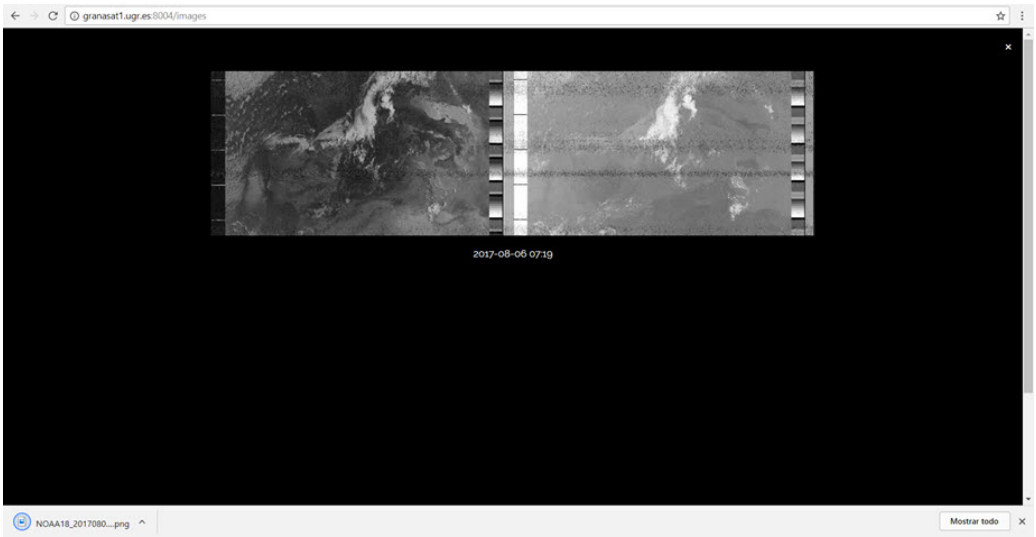


Figura 5.54 – *Imagen descargada haciendo click sobre ella.*

El navegador mostrará las imágenes más recientes, ya que si no habría sobrecarga. El resto de imágenes serán accesibles desde un formulario selector, donde se puede elegir el pase del satélite deseado por fecha. Este formulario ha sido realizado creando elementos HTML de forma dinámica con Javascript mediante *document.createElement(form)*, igual

que en el caso anterior. Como la base de datos se va incrementando en tiempo real, es necesario crear nuevos elementos de esta forma. En la siguiente imagen se muestra el desplegable a utilizar. El selector se implementa con *document.createElement("select")*, seguida de *document.createElement("option")*, como elemento hijo de 'selector', donde las opciones son las imágenes dentro de la base de datos. Haciendo click en 'download' se tiene acceso a la imagen ampliada y permite su descarga.

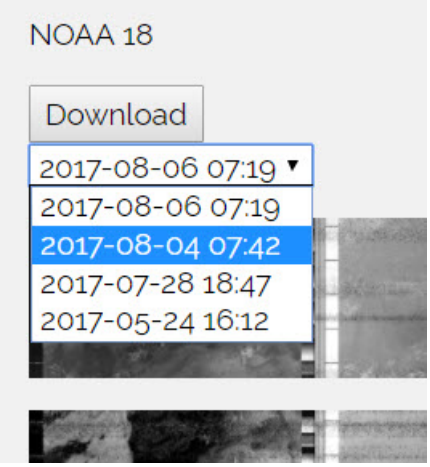


Figura 5.55 – *Creación de desplegable para selección de imagen por el usuario.*

Finalmente, la aplicación de descargas queda del siguiente modo:

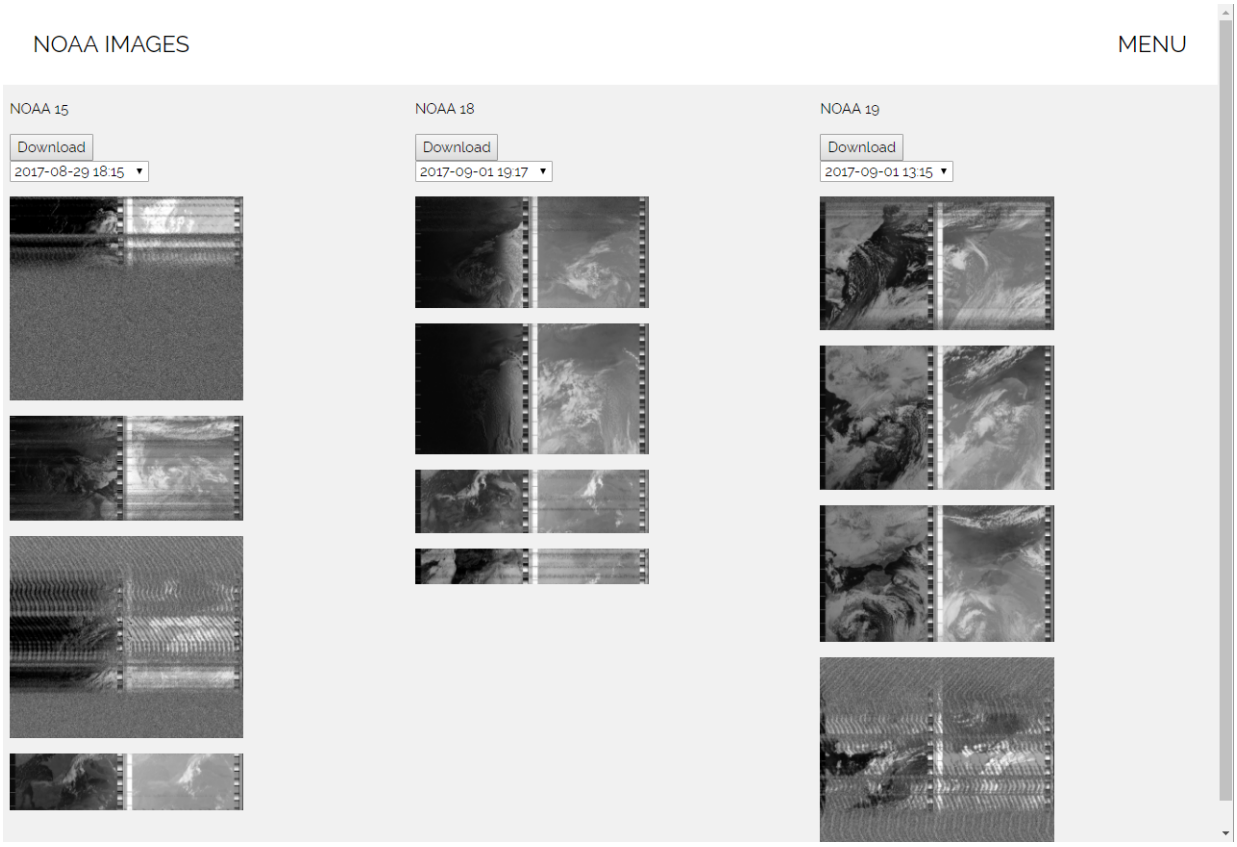


Figura 5.56 – *Servicio para visualización y descarga de imágenes meteorológicas.*

CAPÍTULO

6

INTEGRACIÓN DEL SISTEMA FINAL

El sistema de recepción será ubicado en principio en la terraza de la **Facultad de Ciencias de la UGR**, emplazamiento que tiene asignado el equipo de trabajo de GranaSAT. Este lugar es intercambiable, dada la modularidad del sistema. El sistema de recepción puede ser trasladado sin que afecte al servicio web, gracias a que todo el procesado de la señal recibida se ha implementado en un equipo de reducido tamaño como es Raspberry Pi. Esta señal es enviada usando TCP/IP al servidor web, donde este sí está ubicado en **Facultad de Ciencias de la UGR**.

En posteriores pruebas de verificación se observará que es necesario modificar el emplazamiento del sistema receptor y esto se puede realizar sin problema, solamente con la incomodidad del transporte de la antena receptora.

A continuación se muestra diagrama del sistema final:

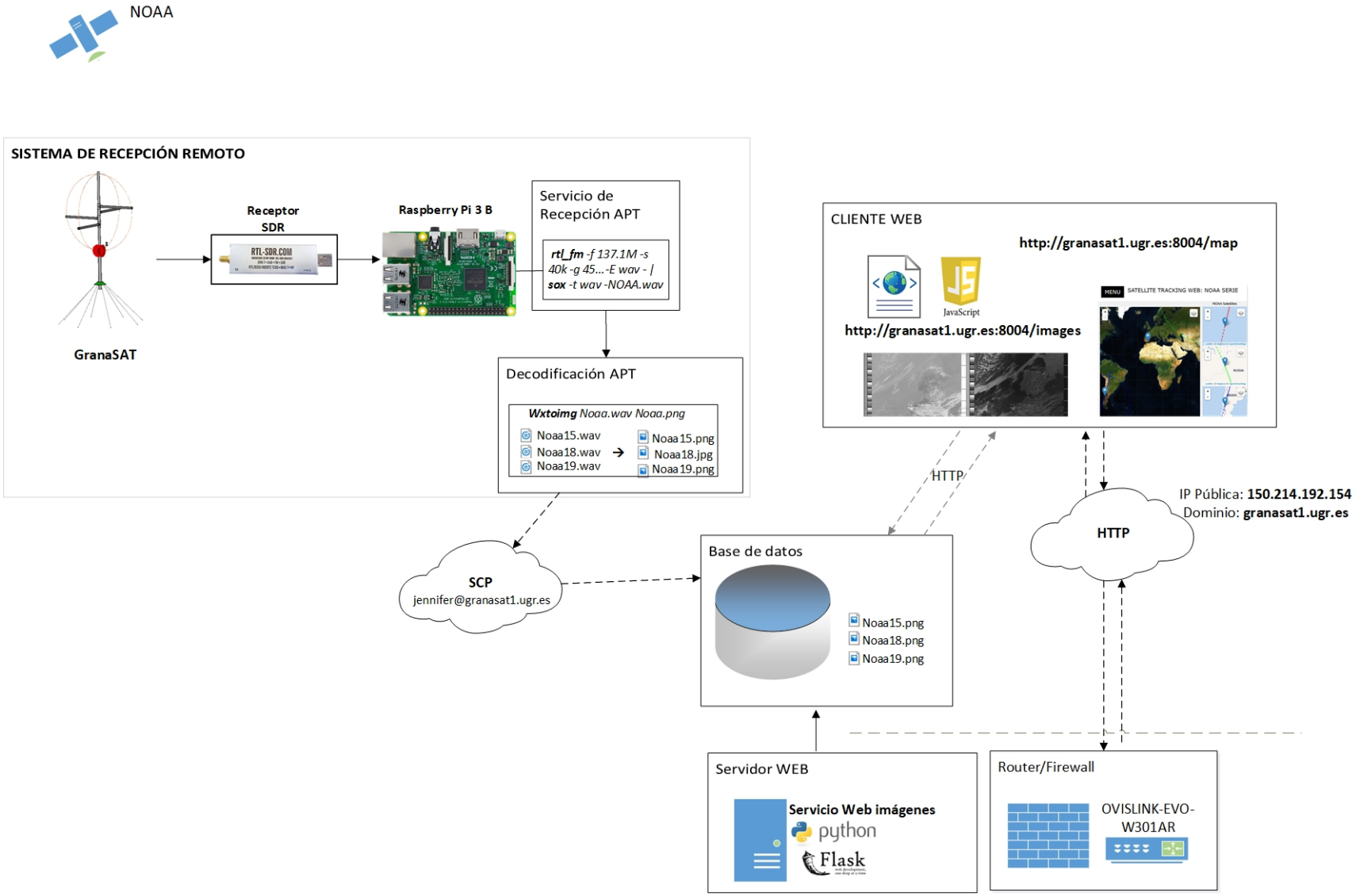


Figura 6.1 – *Pase del NOAA-15 el día 05/08/2017 a las 19:15 con antena Turnstile.*

Como se puede apreciar, el sistema receptor se puede situar en cualquier lugar, siempre y cuando haya conexión a la red, para que mediante el protocolo SCP se envíen las imágenes a la base de datos ubicada en el servidor web. El cliente web puede acceder al servicio creado desde su navegador y descargar las imágenes a través de HTTP, con la funcionalidad de HTML5:

<a ***download=img*** *href=http://granasat1.ugr.es:8004/static/Images/NOAA.png>*

El atributo *download* permite una descarga directa. Esta funcionalidad aún no funciona con todos los navegadores.

Como paso final, queda la evaluación del sistema completamente integrado, una vez los subsistemas hayan sido testados y validados. Los procesos de testeo y sus resultados son comentados en el capítulo 7.

6

CAPÍTULO

7

EVALUACIÓN Y VERIFICACIÓN

Atendiendo al plan de verificación establecido en la sección 4.3, la continuación se pasa a la evaluación del sistema diseñado (mostrado en el Capítulo 5).

7.1 Evaluación de antena receptora. Medición de parámetros de antena

Siguiendo el esquema mostrado en el setup de la Figura 4.15, a antena debe ser medida en campo lejano, sin que haya ningún elemento dentro de su región de radiación o zona Fresnel que pueda falsear la medición. La distancia a partir de la cual se considera que se está en zona de campo lejano o Zona de Fraunhofer viene dada por la siguiente ecuación:

$$R > \frac{2D^2}{\lambda} \tag{7.1.1}$$

Se ha medido el coeficiente de reflexión, la razón de onda estacionaria y la impedancia característica de antena. No ha sido posible realizar una medición de la ganancia y patrón de radiación de las antenas. Sería necesario evaluarlas en una cámara anecoica, o bien, usando un sistema alternativo en un espacio suficientemente grande para que se de la condición de que no haya ningún elemento de interferencia con la antena que provoque reflexiones y usando un sistema de medida con una antena de tipo bocina estandarizada a la frecuencia de recepción de las antenas diseñadas transmitiendo a lo largo de un espacio tridimensional,

permitiendo así poder dibujar el patrón de radiación a partir de la señal recibida por las antenas a medir.

7.1.1 Mediciones de antena Turnstile

La antena ha sido evaluada usando el analizador de redes para medir sus pérdidas de retorno (S_{11}) o su ROE. Ya a priori se sabe que habrá desadaptación, como se vio en la sección anterior. Esta desadaptación es debida a la necesidad de desfasar uno de los dipolos. Los resultados obtenidos se pueden ver en la Figura 7.1, donde se ha obtenido $S_{11} \simeq -9$ dB, equivalente a una $ROE \simeq 2$. Se ha intentado adaptar la antena recortando levemente los dipolos para desplazar la frecuencia de resonancia, pero no ha sido posible mejorar demasiado, ya que se comienza a desadaptar para toda la banda.

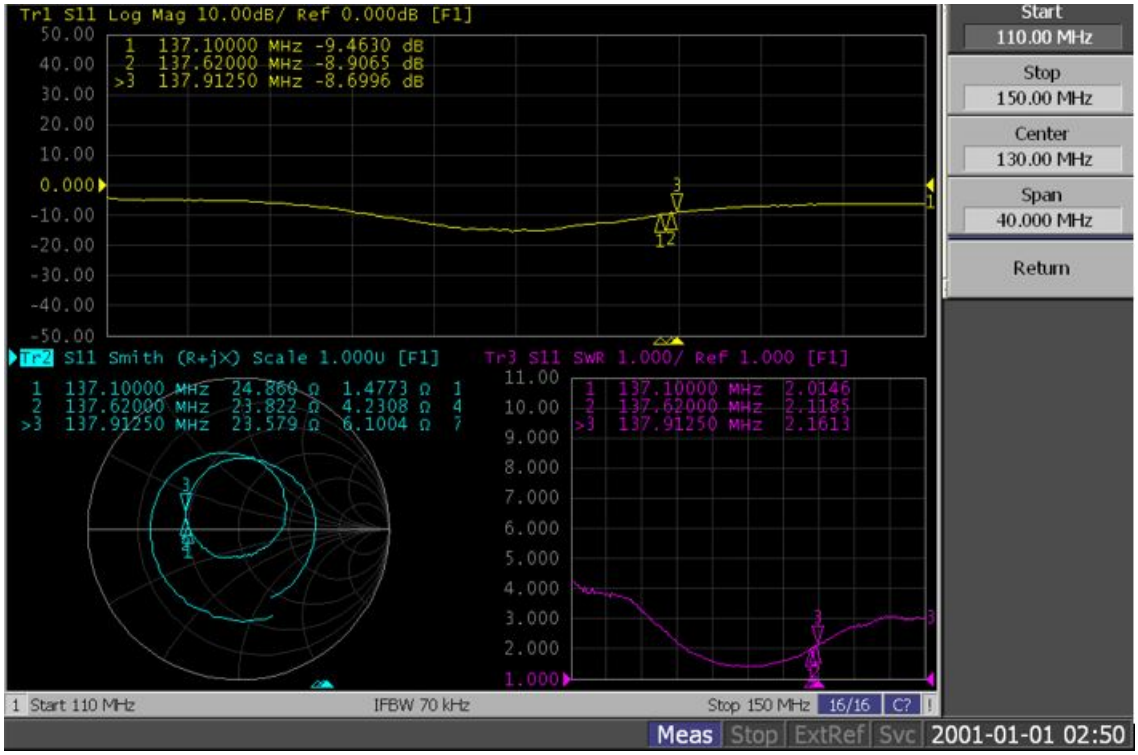


Figura 7.1 – *Medidas resultantes tomadas en el analizador de redes.*

Se ha obtenido una ROE mayor a la esperada, pero aún así, es suficiente para el objetivo de la recepción de señales de los satélites NOAA.

7.1.2 Mediciones de antena Eggbeater

Para concluir, en la Figura 7.2 son mostrados los resultados obtenidos de las mediciones de la antena Eggbeater con un analizador de redes. Se ha analizado el parámetro $|S_{11}|$, la

impedancia de antena y la ROE.

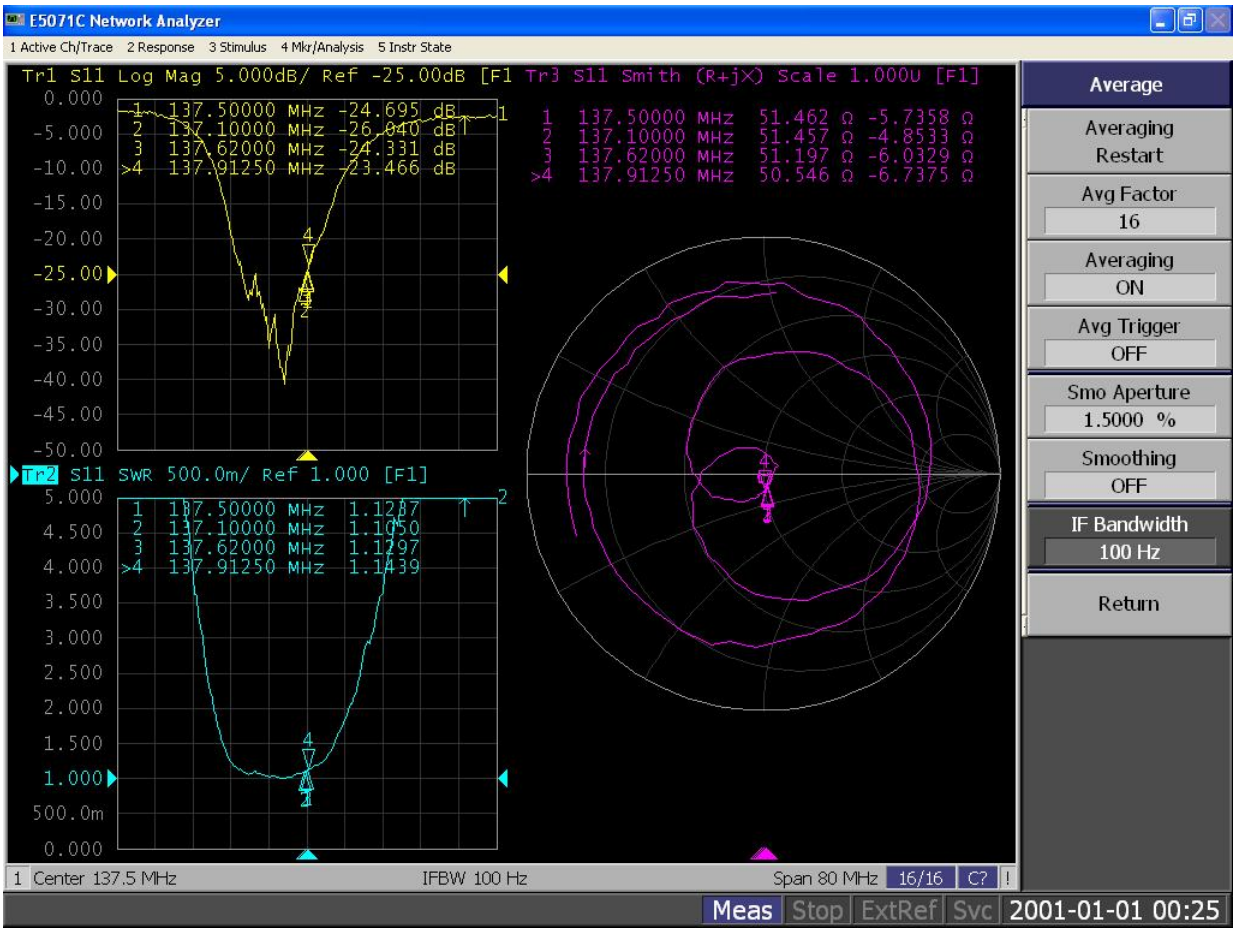


Figura 7.2 – *Mediciones de antena Eggbeater con analizador de redes.*

Como se puede comprobar, tras la adaptación de impedancias, el parámetro $|S_{11}|$ es de -25 dB para toda la banda de recepción de los NOAA y la impedancia de antena resultante es de prácticamente 50 Ω con una parte reactiva pequeña, haciendo que la antena sea casi resonante (cuando la reactancia es nula). Finalmente no se ha añadido una reactancia en serie, como se mostraba en el esquemático de ADS de la Figura 5.34, ya que la parte reactiva es pequeña y no ocasionará grandes pérdidas. Además, la ROE resultante está en torno a 1.13 en la banda de trabajo, un valor muy bueno, ya que es inferior a 1.5.

Los resultados son mucho mejores que los conseguidos para la antena Turnstile vista en la 5.1.1. De modo que, será la antena Eggbeater la usada para la recepción del APT de los satélites NOAA desde la estación terrena.

7.1.3 Recepción de NOAA con antena Eggbeater y Turnstile

En esta sección se compara la recepción dela señal APT de ambas antenas. Es también una buena forma de analizar su comportamiento en cuanto a ancho de haz y directividad, aunque no se esté usando un sistema de medida estándar. El esquema de la Figura 4.16 resume la prueba.

El proceso seguido en la recepción consiste en:

1. RTLSDR debe estar conectado al PC en el que vaya a transcurrir la prueba y a su vez conectado a una antena de recepción. La prueba se va a realizar en un único PC, con el mismo dispositivo RTLSDR, para que no haya duda de que la diferencia en recepciones es únicamente la antena.

2. Inicialización de los software Orbitron, SDRSharp y WXtoImg instalados en Windows. En WXtoImg se selecciona la opción de *'Autorecord'* para que comience a escuchar y decodificar el audio de entrada cuando comience el pase de un satélite. El audio de entrada proviene de SDRSharp, que mediante cable virtual pasa a ser procesado por el software de decodificación de APT. SDRSharp se iniciará cuando Orbitron indique el inicio del pase, dicha comunicación entre ambos software se realiza con el driver *'My DDE Client'*.

3. RTLSDR conmutará su conexión entre ambas antenas durante el pase.

7

A continuación se muestra la señal APT en SDRSharp con ambas antenas durante el mismo pase. La elevación del satélite es de algo más de 20^o y la diferencia de la señal recibida por ambas antenas es notoria. Se observa que hay una diferencia de más de 10 dB entre la señal obtenida usando la antena Eggbeater y la antena Turnstile. Esto indica que la antena diseñada con CST ofrece una calidad mucho mejor que la antena Turnstile, que se ha fabricado siguiendo unos patrones de diseño teóricos, sin considerar a una optimización en diseño.

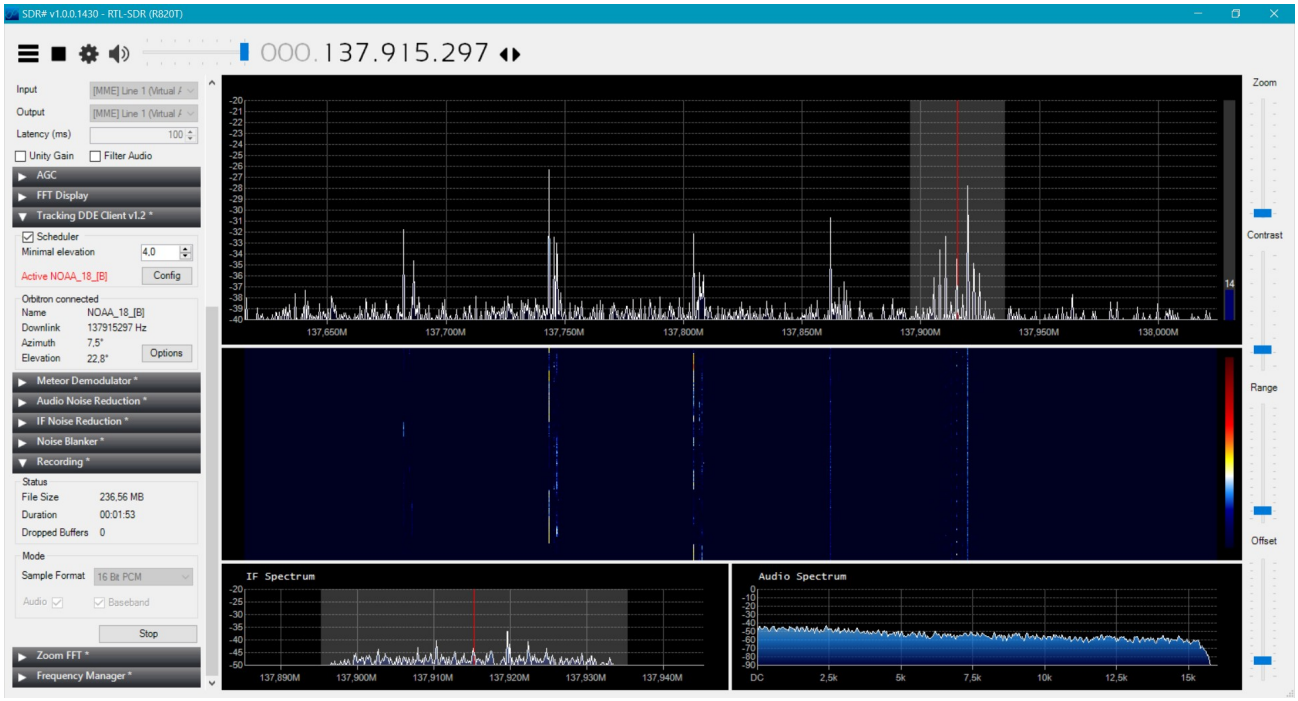


Figura 7.3 – *Recepción conseguida con antena Turnstile del satélite NOAA-18 a 20º de elevación.*

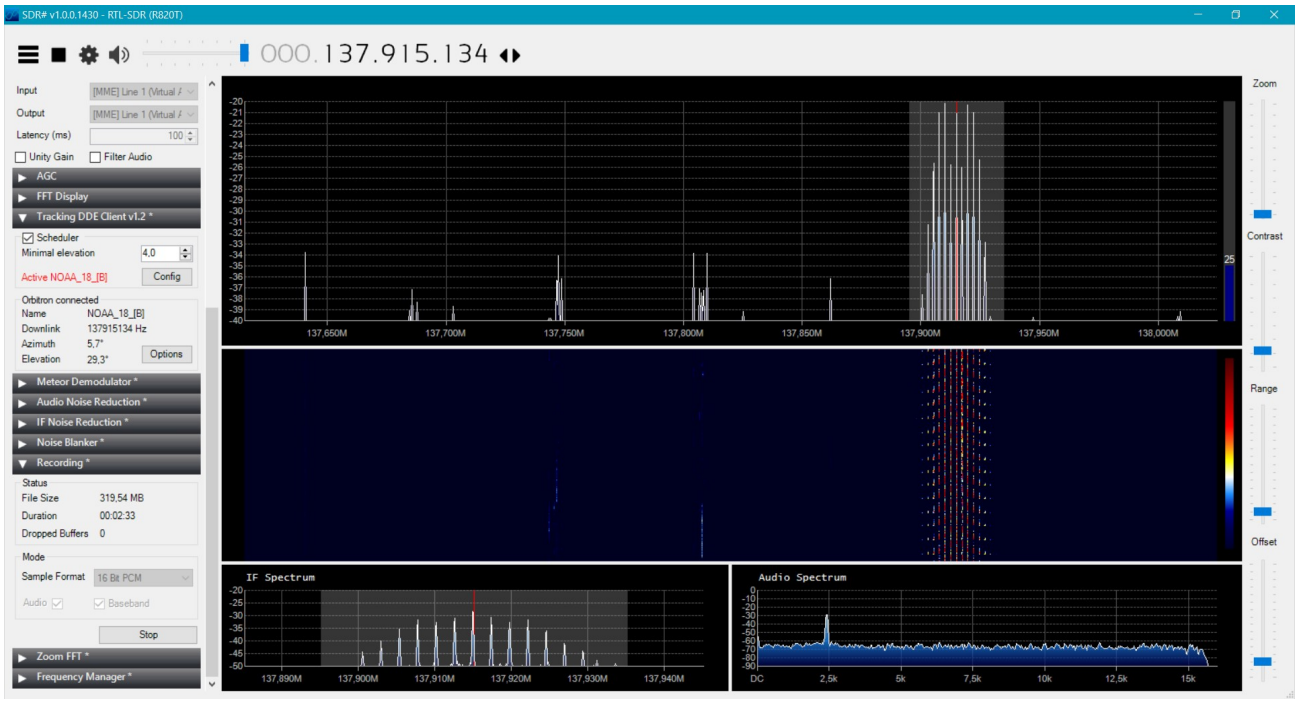


Figura 7.4 – *Recepción conseguida con antena Eggbeater del satélite NOAA-18 a 20º de elevación.*

La señal recibida con esta elevación por la antena Turnstile es tan mala que WXtoImg no es capaz de decodificar la señal, ya que tiene muy bajo SNR. En la Figura 7.5 muestra la diferencia de la decodificación de la imagen al conmutar las antenas receptoras en RTLSDR. Se observa que la antena Eggbeater presenta un gran ancho de haz ya que a 20º recibe la señal perfectamente. Se ha probado que su recepción es buena desde antes de los 10º de elevación. La decodificación final del pase es la mostrada en la Figura 7.6, donde la parte central de la imagen es la decodificación de la señal turnstile. Se aprecia como empeora la SNR.

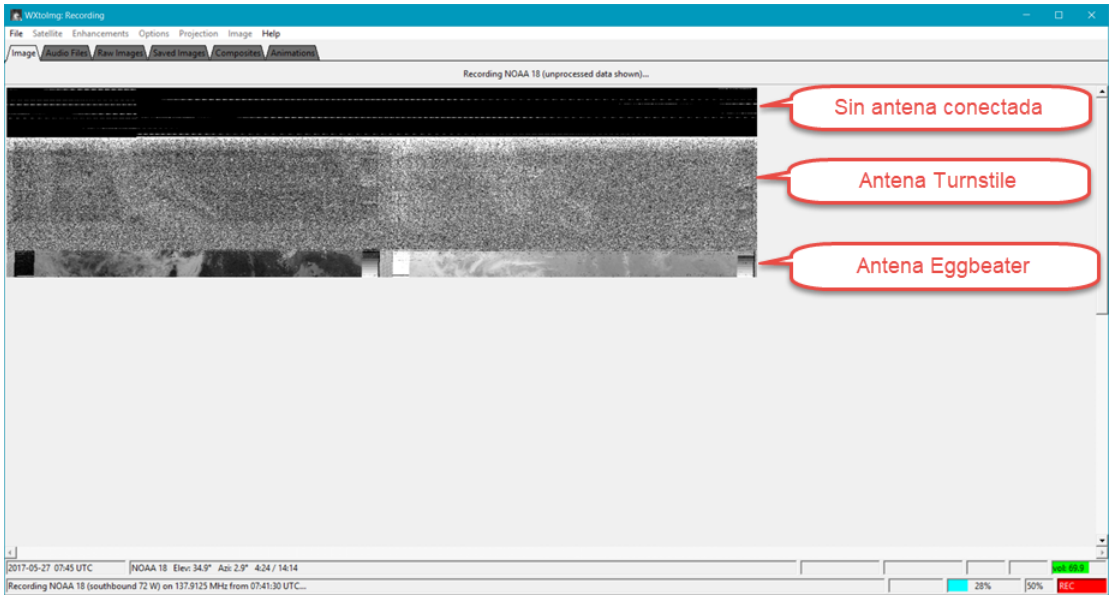


Figura 7.5 – *Comienzo de decodificación con WXtoImg.*

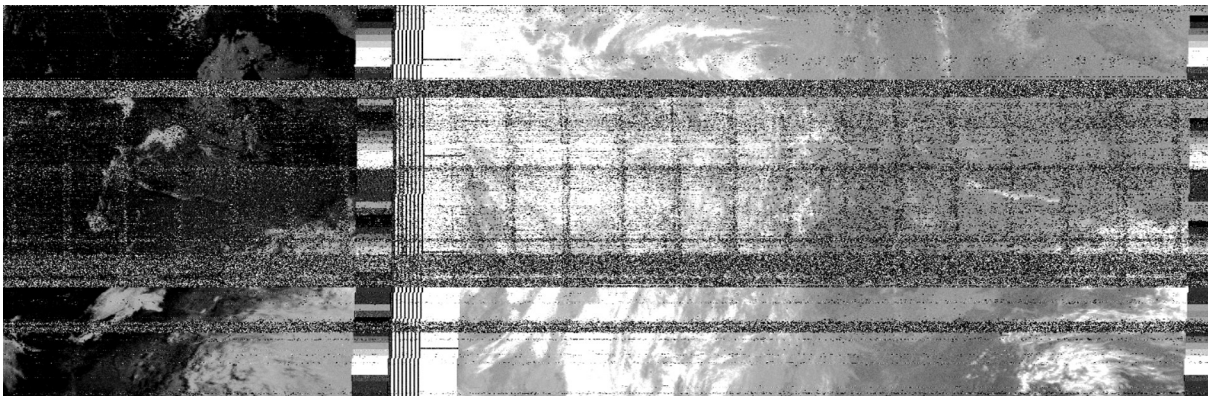


Figura 7.6 – *Decodificación con WXtoImg comparando ambas antenas en el mismo pase.*

Se preveía el peor resultado de la antena Turnstile dadas las medidas con el analizador de redes realizadas. Además, la antena Eggbeater fue optimizada para tener un gran ancho

de haz (beamwidth) y se ha demostrado que se ha conseguido.

7.2 Evaluación de receptor SDR

En esta sección se desarrolla el plan de evaluación planteado en 4.3.2.

7.2.1 Librerías de control de SDR

El software SDRSharp ha sido probado en el anterior plan de evaluación directamente con la antena y la recepción del satélite comprobando que funciona como lo esperado. Además de esto, los dispositivos SDR son calibrados con ayuda de este software y un generador de señal, descrito en Anexo B junto con otras alternativas.

Se evalúa por tanto la librería *libretlsdr* en [17] (con versión v0.5.3-69) y su nueva versión en pruebas de [31] (con versión v0.5.3-88). Estas librerías serán probadas tanto en Windows, con la versión v0.5.3-69 compilada, y en Linux. Para finalizar se probará sobre Raspberry Pi.

También será necesario instalar la librería *SOX*. Ha sido probada la versión para Linux disponible en el repositorio y compilada para windows 14.4.2 [13].

Para esta etapa de pruebas de utiliza el analizador de comunicaciones **Marconi Instrument R2955A**, como se vio en los esquemas 4.17 y 4.18. Las pruebas se han realizado para un tono de 1 kHz con una portadora de 300 MHz y el tono de APT (2.4 kHz modulado en FM sobre una portadora de 137.1 MHz con una desviación de 17 kHz).

Figura 7.7 – *Señales de test con analizador de comunicaciones.*

1. ***librtlsdr*+SOX en Windows:**

   En Windows se ha probado la única librería compilada, que corresponde con la versión antigua de [17]. En Windows se debe trabajar en dos fases. Primero se graba un fichero RAW usando un comando sencillo, con la terminal en el directorio de descarga de la librería compilada para *RTLSDR* se ejecuta el siguiente comando (la tabla 5.1 muestra las opciones básicas de *rtl_fm*):

   rtl_fm -f 137.9125M -s 60k -p 60 -g 20 grabacion

A continuación, el archivo RAW se convierte a WAV usando la librería SOX. Desde la terminal, ahora posicionados en el directorio de SOX compilado:

sox -e s -c 1 -b 16 -r 60k -t raw – Noaa.wav

Se puede cambiar la frecuencia de muestreo con SOX con la opción *'rate'*.

Los resultados han sido los siguientes:

- Se obtienen buenos resultados seleccionando un ancho de banda de 60 kHz.
- Al cambiar el ancho de banda y ponerlo inferior (como a 40 kHz) la señal se distorsiona.
- Comparando la recepción de esta señal generada con SDRSharp con un ancho de banda de 60kHz y la grabación con RTL_FM+SOX, esta última tiene peor SNR.
- La opción -E wav crea un fichero con cabecera errónea.

Las pistas entregadas en la carpeta 'Pruebas de Evaluación' son:

- 60k_137100000Hz_AF: SDRSharp a 60kHz. A 40 kHz también se obtiene buena señal y a menores anchos de banda.

- gen2_60k_40dB: rtl_fm+sox. Señal ruidosa.

- gen3_40k_40dB: rtl_fm+sox. Grabación tomando 40 kHz de ancho de banda. El pulso se escucha distorsionado o movido en frecuencia.

2. ***Librtlsdr*+SOX en Linux:**

   En estas pruebas se usó el tono de 1 kHz a 300 MHz y se probó que una variación de la ganancia también distorsionaba el pulso, además de la variación del ancho de banda vista en las pruebas en Windows. El comando usado es:

   rtl_fm -f 300M -s 60k -p 60 -g 20 | sox -e s -c 1 -b 16 -r 60k -t raw – grabacion.wav

   Las pistas entregadas en la carpeta 'Pruebas de Evaluación' son:

   - SDRSharp_60k_300MHz_AF:SDRSharp. Buena grabación.

   - gLinux_60k_40dB: Mal audio con ganancia 40 dB.

   - gLinux_60k_20dB:Mal audio con ganancia 20dB.

   - gLinux_60k_50dB:Audio algo mejor con ganancia 50dB.

3. ***Librtlsdr*+SOX en Raspberry**

   Se prueba en Raspberry grabando el tono del generador. Se realizan diversas pruebas para comprobar cómo afecta la librería SOX y *RTL_FM* al combinarse.

   Con PLAY de SOX se escucha el tono perfectamente usando el siguiente comando desde la terminal:

   rtl_fm -d 0 -f 137.1M -s 60k -p 46 -g 40 | play -e s -c 1 -b 16 -r 60k -t raw -V1 –

   Ha sido, porque con un ancho de banda menor se produce distorsión del tono.

Archivo	RTL_FM	SOX	Comentario
Record1.wav	rtl_fm -d 0 -f 137M -s 60k -p 46 -g 40 record1 (Grabada en formato raw)	Procesado en Windows con SOX	Buen audio
Record2.wav	rtl_fm -d 0 -f 137M -s 60k -p 46 -g 40 | sox -e s -c 1 -b 16 -r 60k -t raw - record2.wav	Todo junto con pipeline en Raspberry	Mal audio
Record3.wav	rtl_fm -d 0 -f 137M -s 60k -p 46 -g 40 record3	sox -e s -c 1 -b 16 -r 60k -t raw – record1 record3.wav (en Raspberry)	Buen audio
Record4.wav	rtl_fm -d 0 -f 137M -s 60k -p 46 -g 40 record4.wav	-	Fallo en la creación de archivo

Tabla 7.1 – *Batería de pruebas en Raspberry Pi 2B+ con rtl_fm de [17]*

A continuación se detallan los diferentes comandos usados y las observaciones de una batería de pruebas en Raspberry Pi B+ para verificar librería SOX:

Una nueva conclusión es que ejecutar los comandos rtl_fm y sox usando *'pipeline'* no ha dado buen resultado. Este problema se corrige utilizando el proceso *'timeout'* al inicializar la recepción. Este proceso resulta erróneo sobre Raspberry Pi 2B+ debido a una velocidad de procesado limitada. No ocurre esto usando Raspberry Pi 3B. De modo que se selecciona este equipo.

Se han realizado pruebas similares con la librería RTLSDR más actual [31]. En este caso se comprueba que mejoran los problemas visualizados anteriormente. Ya no afecta la variación de la ganancia y del ancho de banda del dispositivo como ocurría con la versión más antigua. El tener un correcto software y hardware seleccionado permite poder ajustarla ganancia del dispositivo, entre otros parámetros.

Archivo	RTL_FM + SOX	Comentario
gen_RaspJenn1.wav	**timeout 20s rtl_fm -f 137.1M -s 40k -g 45 -p 62 -E wav -E deemp -F 9 - | sox -t wav - gen_RaspJenn1.wav rate 11025**	Buen audio con ganancia 45 dB
gen_RaspJenn2.wav	timeout 20s rtl_fm -f 137.1M -s 40k -g 20 -p 62 -E wav -E deemp -F 9 - | sox -t wav - gen_RaspJenn1.wav rate 11025	baja SNR con ganancia 20 dB
gen_rasp2Jenn3.wav	timeout 20s rtl_fm -f 137.1M -s 40k -g 45 -p 62 -E wav -E deemp -F 9 gen_rasp2Jenn3.wav	Buen audio pero error en cabecera WAV

Tabla 7.2 – *Pruebas en Raspberry Pi 3B con rtl_fm de [17]*

Finalmente se selecciona el comando marcado como el óptimo para la recepción con SDR.

4. **Testeo de *RTL_TCP*:** Este comando de la librería RTLSDR es usado para recepción desde un equipo remoto. El equipo con el receptor SDR conectado transmite el audio con el protocolo TCP/IP. Se activa el dispositivo del siguiente modo:

   rtl_tcp -a 0.0.0.0 -p 1234

   Desde otro equipo dentro de la LAN,usando SDRSharp se escucha el audio enviado desde el equipo remoto. Se ha observado latencia al visualizar el espectro con SDRSharp, pero el audio grabado con SDRSharp es bueno, incluso con muy bajo ancho de banda. Desde SDRSharp se controla los parámetros del dispositivo SDR en remoto.

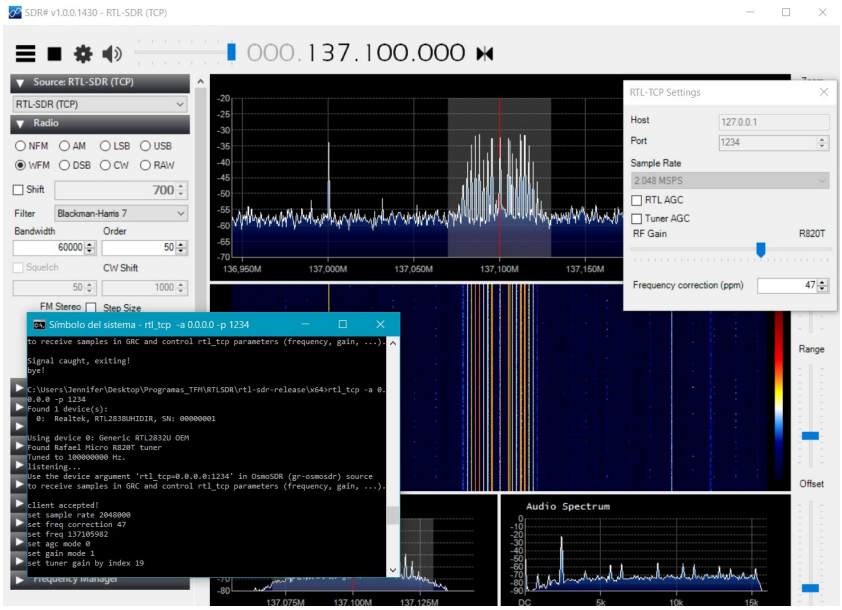


Figura 7.8 – *Uso de RTL_TCP.*

7.2.2 Pruebas con diferentes dispositivos

Se han utilizado diversos dispositivos *RTLSDR* de bajo costo y se han comparada con un dispositivo *RTLSDR* con ppm=0, con un precio mayor ya que su oscilador es mejor.

Los resultados obtenidos son muy similares, por lo que no hay motivo para desechar estos dispositivos de tan bajo costo.

En las pistas entregadas en la carpeta 'Pruebas de Evaluación' se ha introducido el audio *'gen_RaspAlb1.wav'* generado usando *RTLSDR* con ppm=0 y el comando final. Se aprecia que el audio es bueno.

FUNcube Dongle Pro+ ha sido probado en SDRSharp obteniendo igualmente buenos resultados, pero necesita una librería diferente a *'librtlsdr'*.

7.3 Evaluación del predictor en aplicación web

Otra de las evaluaciones realizadas es la comprobación de que el servicio de predicción de pases de satélites devuelve unos resultados correctos. Se ha comparado con el software Gpredict, actualizando sus keplerianos previamente para que no haya errores. En la Figura 7.9 se comprueba visualmente que el footprint de los tres satélites coincide. Además, si hacemos click sobre uno de los satélites en la aplicación web se despliega una ventana con la información de las coordenadas geográficas del satélite. En la Figura 7.10 se prueba que coincide la medida la ubicación de satélite NOAA-15. Hay una mínima diferencia que se debe a que la actualización de la información de la aplicación web sucede cada 1000 ms y hay un mínimo desfase temporal.

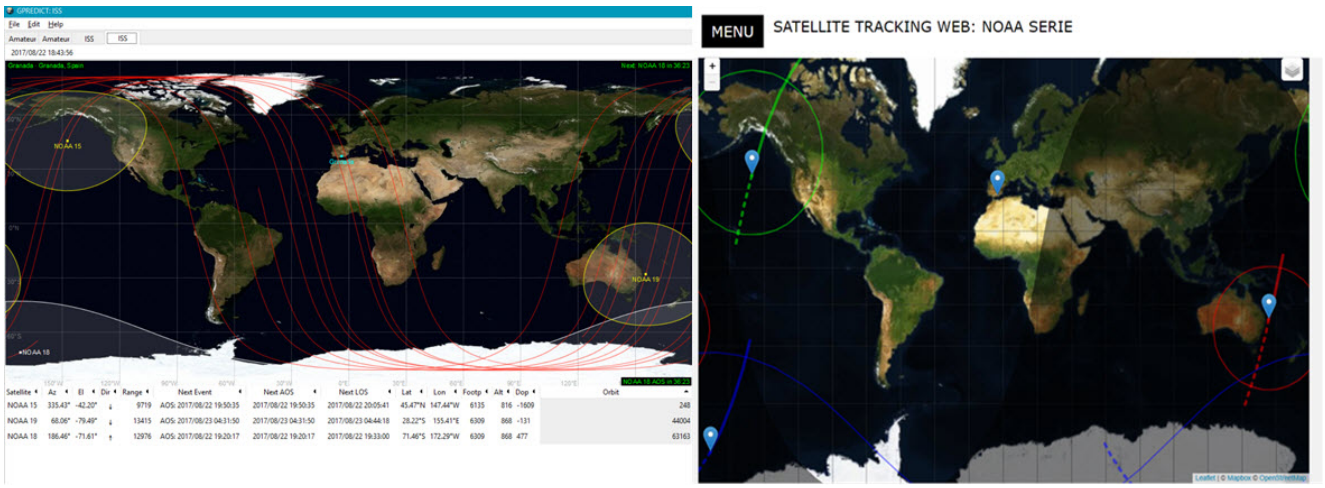


Figura 7.9 – *Comparación de posición y footprint de satélites NOAA desde página web y Gpredict.*

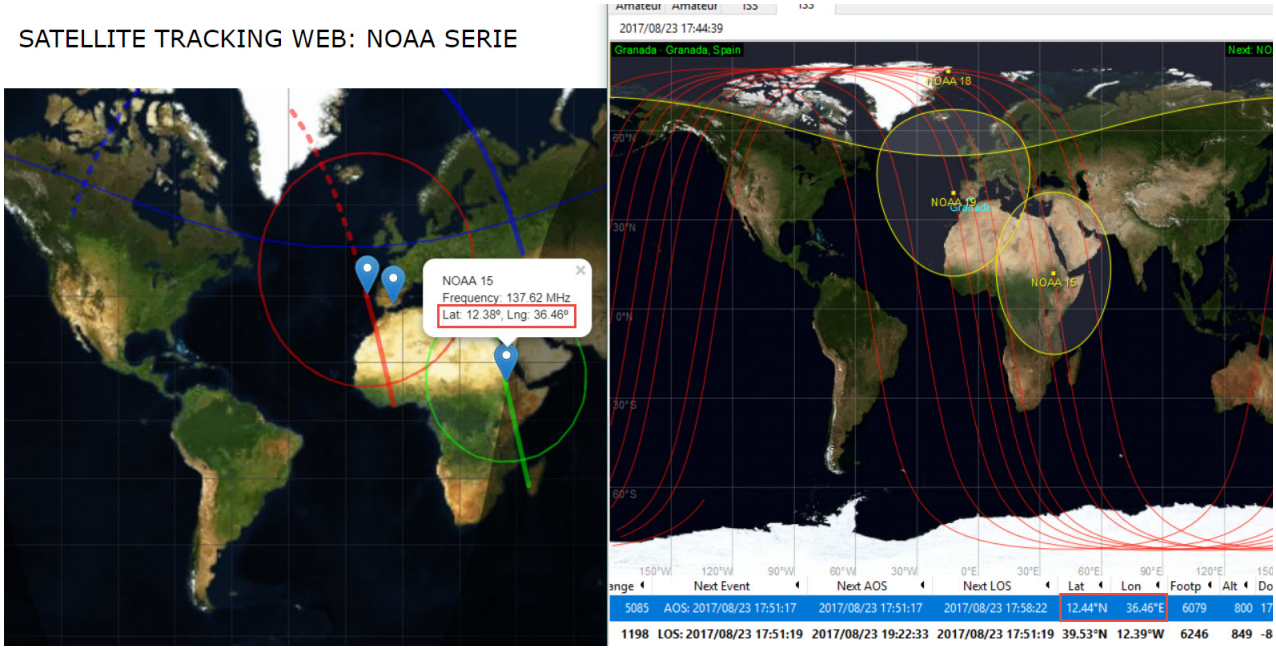


Figura 7.10 – *Comparación de posición de NOAA-15 desde página web y Gpredict.*

En la siguiente Figura se observa la comparación de los pases NOAA desde la página web y desde Gpredict. Prácticamente no hay diferencia (sólo segundos) y esta es debida a que el cálculo de los keplerianos varía debido a las perturbaciones terrestres, y dicho cálculo puede verse afectado si se realiza en diferentes instantes temporales.

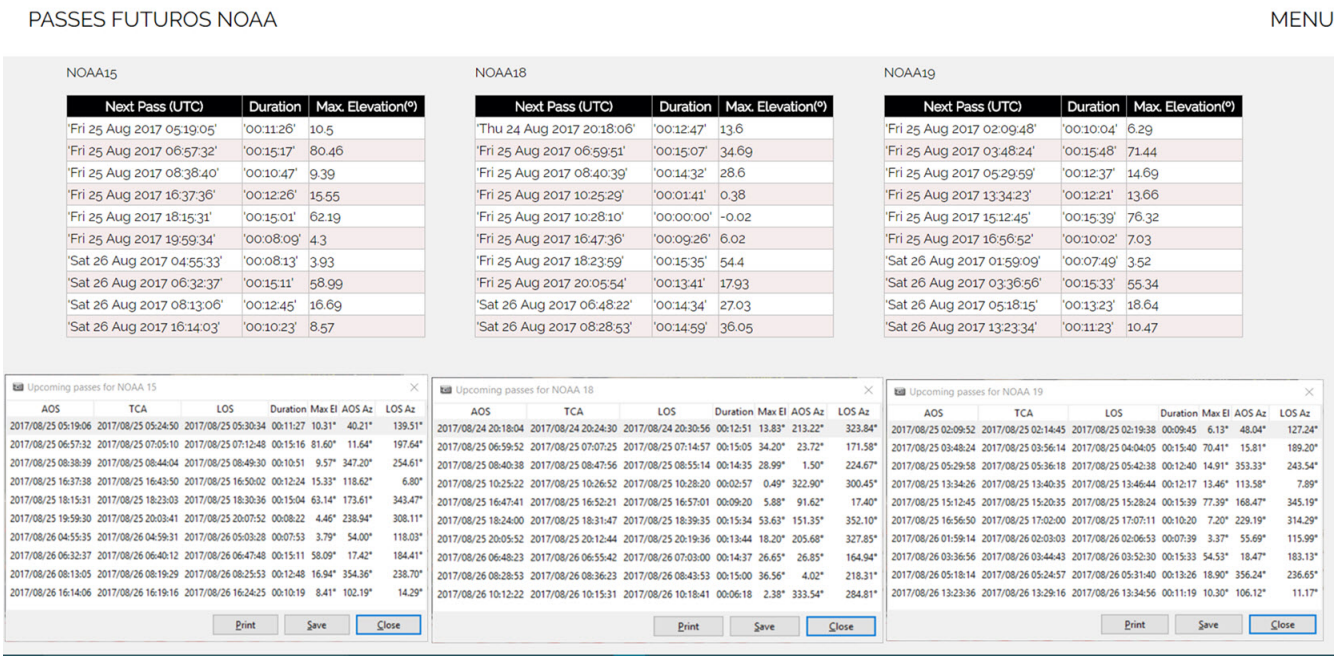


Figura 7.11 – *Comparación de los pases futuros desde página web y Gpredict.*

7.4 Evaluación de decodificación de la señal APT

Una buena SNR es fundamental para que la señal pueda ser decodificada y obtener una buena imagen del pase. El ruido puede deberse a interferencia de otras señales próximas a la banda de operación de los NOAA, obstáculos en la línea de visión directa, ruido eléctrico en la alimentación del receptor, pérdidas por desadaptación de impedancias, pérdidas por cross-polarization, etc. Estos factores deben ser examinados a lo largo del diseño, implementación y validación del sistema.

Una vez la señal es recibida con buen SNR, se procede a su decodificación.

Para la decodificación APT se ha seguido el esquema mostrado en 5.47. En un primer momento, fue empleado un script escrito en Python para la decodificación de dicha señal, fin embargo, su ejecución en Raspberry Pi producía un mensaje de error de memoria interna ocupada. Esto se debe a que la librería para la implementación de la Fast Fourier Transform (FFT) no está optimizada, ocasionando un error de memoria en dicho cálculo. De modo que, la pista de audio debía ser enviada al servidor principal y allí realizar la decodificación de la señal. Esta implementación suponía un retardo considerable en el servicio de recepción, ya que el envío de un archivo de audio de 15 min retrasa la ejecución.

Así pues, se decidió emplear un software suficientemente testado y optimizado. Se ha empleado WXtoImg con resultados muy satisfactorios, igual que los obtenidos si se emplea para Windows. A continuación se muestran algunas de las imágenes obtenidas con la decodificación desde este software usando la antena Eggbeater y la antena Turnstile. Se

comprueba nuevamente que la antena Eggbeater ofrece mejores resultados.

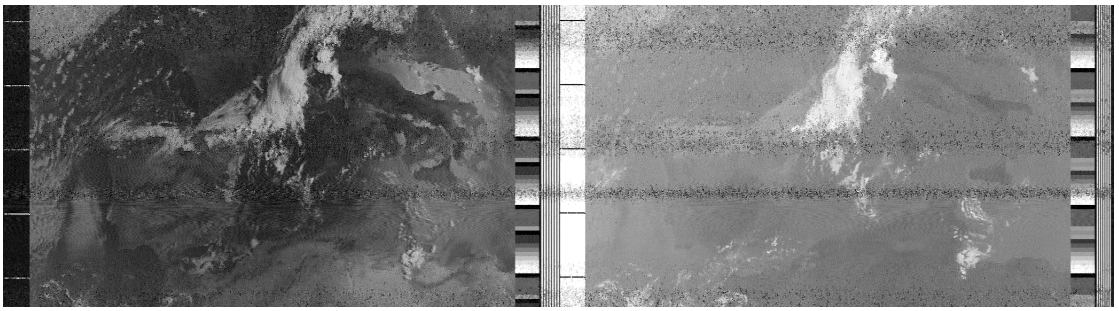


Figura 7.12 – *Pase del NOAA-18 el día 06/08/2017 a las 21:35 con antena Turnstile.*

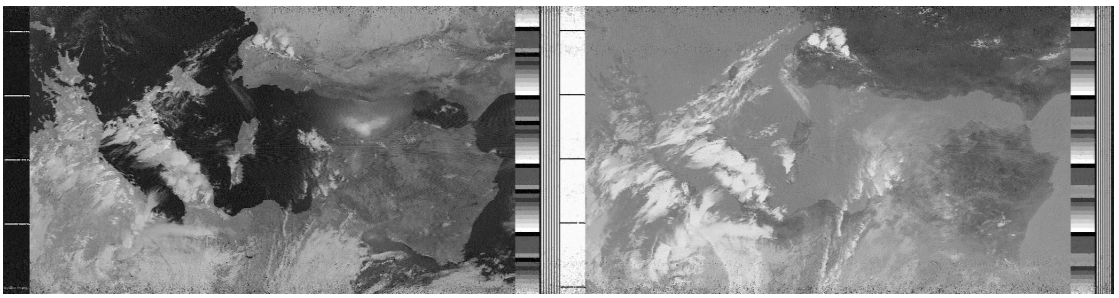


Figura 7.13 – *Pase del NOAA-19 el día 26/07/2017 a las 16:15 con antena Turnstile.*

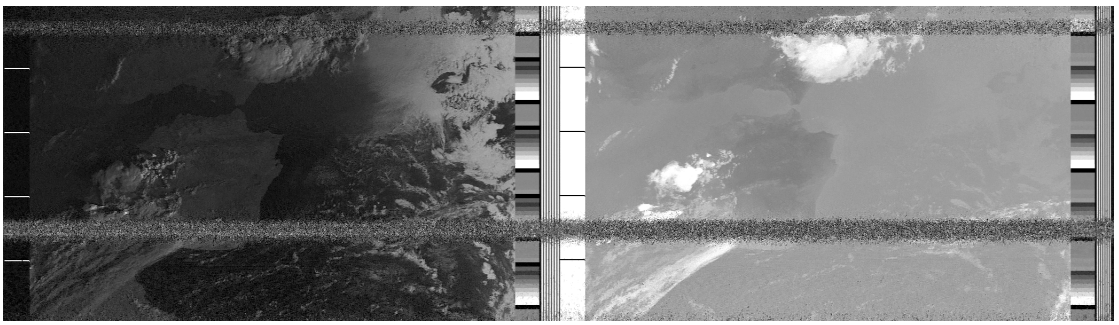


Figura 7.14 – *Pase del NOAA-15 el día 05/08/2017 a las 19:15 con antena Turnstile.*

El efecto doppler es corregido directamente por el software de decodificación empleado en nuestro servicio en banda base. El efecto doppler no ha podido ser corregido en frecuencia RF durante la recepción de la señal, tal y como se pretendía en un principio en el análisis del problema. Una vez ejecutado el proceso *rtl_fm* se ha podido llegar a controlar modificando su frecuencia y ganancia como parámetros de entrada, sin embargo, no se ha conseguido controlar de manera óptima dado que la técnica empleada para dicho proceso producía interrupciones en la captura de audio, es decir, en el instante de modificar la frecuencia

central para corregir el efecto doppler se produce un micro-corte en el audio.

7.5 Evaluación del sistema completo

7.5.1 Evaluación del sistema de recepción final

El esquema mostrado en 4.21 muestra el sistema de recepción final. En las anteriores secciones se ha ido diseccionando cada una de las partes que lo conforman y rediseñando si los resultados no han sido favorables. De modo que, cuando se integran todas las piezas como indica el esquema 4.21, el resultado debe ser favorable.

Sin embargo, la imagen obtenida presentaba muy mala SNR. Dado que las demás pruebas habían pasado los test de validación, el problema del ruido con el que la señal era capturada tenía que deberse al emplazamiento del sistema de recepción, la terraza de la Facultad de Ciencias de la UGR.

Las pruebas para validar la antena recibiendo la señal de los satélites NOAA fueron realizadas en el mismo lugar salvo que usando el PC+SDR y la señal era decodificada sin problema, así que la terraza es un buen sitio para recibir sin interferencias. De modo que todo indica que el problema se debe a ruido eléctrico. Dado que la Raspberry Pi alimenta el dispositivo SDR, se prueba a alimentar la raspberry con su fuente de alimentación oficial de Raspberry Pi 3B (5.1 V, 2.5 A).

El problema continuaba, de modo que se hicieron medidas del amperaje y tensión que consume el equipo. Las figuras siguientes muestran que el equipo no consume en ningún momento los 5.1V a 2.5A con SDR conectado y en funcionamiento, de modo que queda demostrado que no hay problema con el modo en el que la raspberry está siendo alimentada.

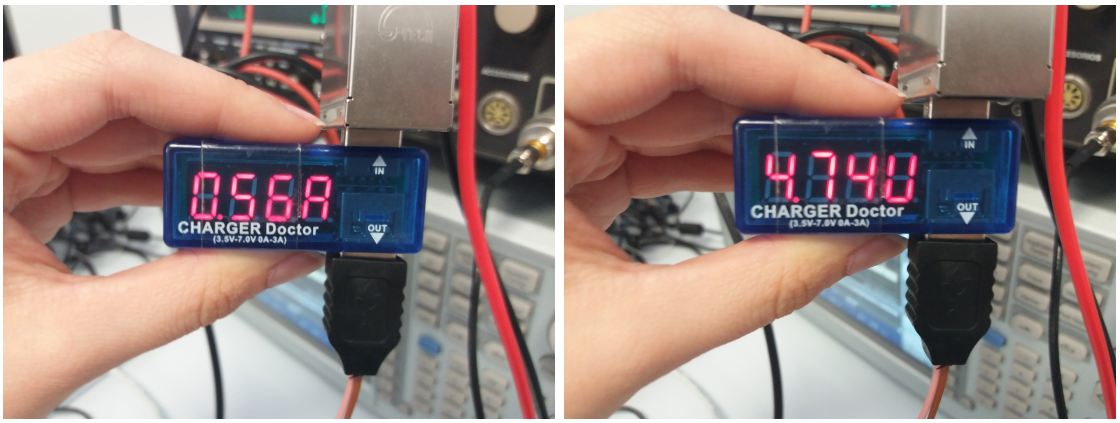


Figura 7.15 – *Medidor de corriente (a izquierda) y tensión (a derecha).*

Finalmente se comprueba si el problema es externo al sistema. Efectivamente cuando el sistema de recepción cambia su ubicación, la recepción satelital es perfecta. El problema

era ocasionado por la instalación eléctrica de GranaSAT. Se ha realizado un despliegue de red eléctrica desde el laboratorio de GranaSAT hasta la terraza de la Facultad de Ciencias, siendo esta red utilizada por multitud de equipos de comunicaciones. En la Figura 7.16 se puede ver la toma de red eléctrica para la antena. De modo que, nuestro en nuestro equipo se introduce ruido eléctrico que enmascara la débil señal APT

Figura 7.16 – *Alimentación de sistema receptor en GranaSAT.*

Ahora el sistema receptor ha sido trasladado al edificio IMUDS del PTS (Parque Tecnológico de la Salud), que será el próximo emplazamiento del equipo GranaSAT.

Figura 7.17 – *Antena trasladada a edificio IMUDS.*

7.5.2 Evaluación del servicio web final

El servicio web queda validado haciendo uso de él y comprobando que cumple con los requisitos impuestos por cliente. Efectivamente con esta aplicación el cliente tiene acceso a la base de datos de imágenes meteorológicas creadas en el servidor web, permitiéndole visualizar las imágenes de los pases de los NOAA sobre la estación terrena.

La descarga de imágenes es realizada mediante HTTP, con el atributo 'download' de HTML sobre una ruta, en este caso la ruta de la ubicación de la imagen en el servidor.

7.5.3 Evaluación de la integración HW+SW

La integración del sistema final (véase 6.1) consiste en el envío de las imágenes capturadas por el sistema receptor, compuesto por antena + receptor SDR + Raspberry Pi, al servidor web de Linux, con dominio **granasat1.ugr.es**.

Con los subsistemas evaluados de forma independiente, el único punto de fallo es la transferencia de información de un servidor a otro. En este caso, desde Raspberry Pi se implementa en el servicio de recepción el comando para dicha transferencia. Finalmente,

usando SCP tal y como se indicó en 5.2.1.4, el sistema queda finalizado y sin error en dicho proceso.

CAPÍTULO

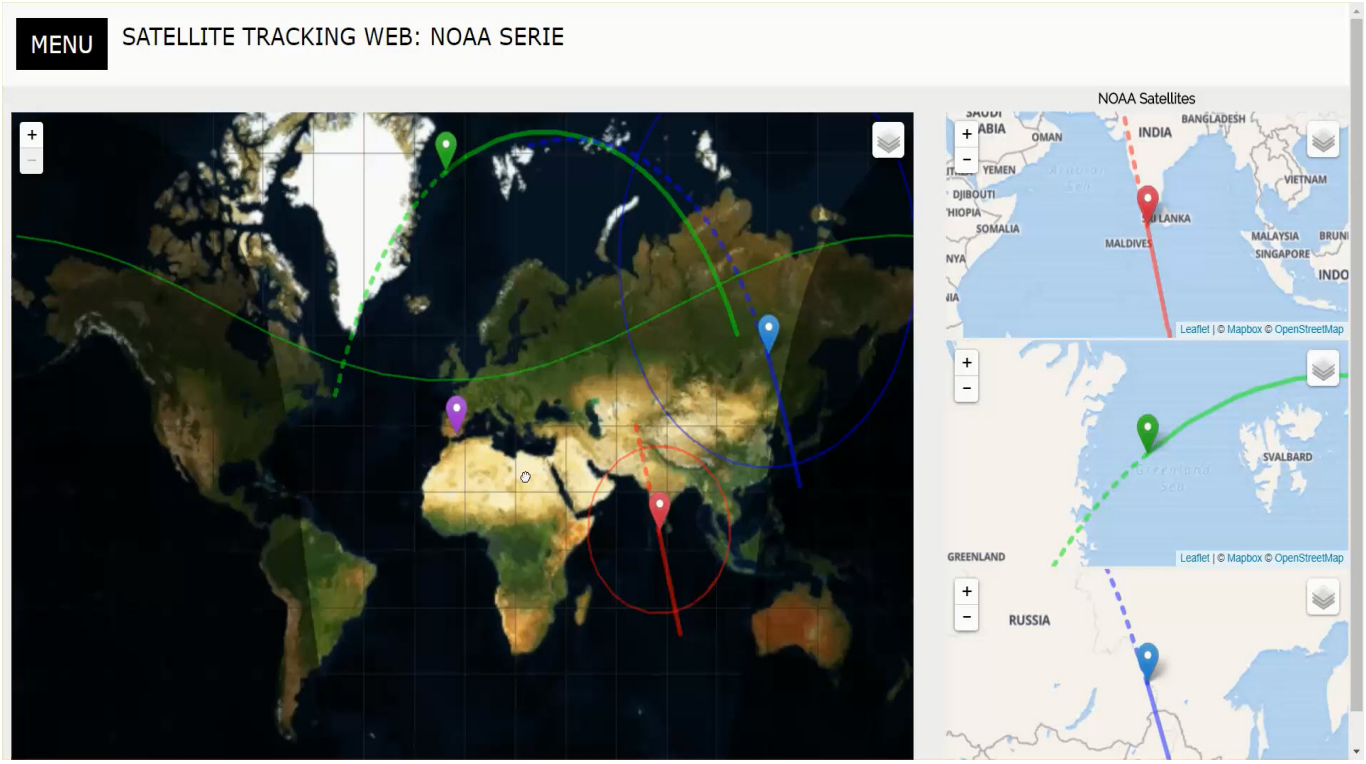
8

RESULTADOS Y CONCLUSIONES

8.1 Resultados

Siguiendo los requisitos se ha desarrollado un sistema a nivel software y hardware para la obtención de imágenes meteorológicas. Un usuario podrá descargar y visualizar en su navegador las imágenes que ofrecen satélites meteorológicos de órbita polar.

En el siguiente vídeo indexado se puede visualizar la aplicación web desarrollada. Esta aplicación permite el seguimiento de los satélites NOAA activos, que son el NOAA-15, NOAA-18 y NOAA-19. Desde el Menú desplegable se tiene acceso a la hora y fecha de los próximos diez pases de los tres satélites, además de otra información de sus órbitas polares. Es en este instante en el que el sistema receptor se activará permitiendo la captura de la señal APT de los satélites y y decodificando la señal para obtener la imagen meteorológica correspondiente. Igualmente, desde el navegador se puede proceder a la descarga de estas imágenes. En la Figura 8.1 se observa la colocación del sistema receptor en GranaSAT, pudiendo ser una localización diferente dada la modularidad del sistema.



Video 8.1 – *Video de aplicación web.*

Figura 8.1 – *Sistema receptor en GranaSAT.*

8

El servicio web está activo y disponible al usuario accediendo a `http://granasat1.ugr.es:8004/map`.

De los resultados finales de este proyecto se puede decir que los objetivos y requisitos

han sido cumplidos satisfactoriamente. El plan de evaluación prefijado ha permito que se resuelvan deficiencias y se optimice el sistema en cuanto a antena receptora, software de recepción e implementación web.

La evaluación de los subsistemas integrantes ha permitido detectar deficiencias que se han subsanado. Entre estos destaca que se ha conseguido determinar que la librería para el control de RTLSDR de [17] no demodula eficientemente señales con pequeños anchos de banda y además distorsiona la señal con cambios de ganancia mediante la opción para control de ganancia. La librería [31] consigue corregir ambos fallos, aunque algunas de sus opciones siguen sin funcionar de manera óptima, como la creación de archivos WAV, donde se produce error en la creación de la cabecera. Además, la comparación entre las dos antenas fabricadas demuestra que un Eggbeater ofrece mayor ancho de haz, menores pérdidas de retorno y un patrón de radiación prácticamente omnidireccional en todo el plano horizontal. Mientras, la antena Turnstile debería ser optimizada con CST para mejorarla, ya que los resultados no han sido demasiado satisfactorios en comparación con la antena Eggbeater.

8.2 Líneas futuras

Como futuros trabajos, siguiendo la línea de este proyecto, surgen varios aspectos interesantes a mejorar y optimizar. Igualmente puede ser el primer paso para el desarrollo de sistemas nuevos que podrían usar este sistema basado en SDR.

La antena diseñada y fabricada tiene un patrón de radiación en su plano horizontal. Esto permite que no sea necesaria la integración de rotores que realicen un seguimiento del satélite. Con este tipo de antenas se obtiene buenos resultados para la recepción de los NOAA, como se ha comprobado. Sin embargo, para la recepción de otro tipo de satélites (POES) Polar Operational Environmental Satellites, las antenas óptimas son antenas con alta directividad. Con antenas omnidireccionales no se consigue una gran ganancia, ya que su densidad de potencia se distribuye en un gran ancho de haz. Utilizar un sistema de rotores que controlen azimut y elevación es lo ideal para el seguimiento de satélites (POES) Polar Operational Environmental Satellites en tiempo real y con grandes resultados gracias a la gran directividad que pueden tener estas antenas montadas en rotores. Por lo tanto, un valor añadido podría ser este cambio de antena receptora.

Por otro lado, dado a la evaluación realizada del sistema de recepción, se han detectado errores de implementación software de la librería 'librtlsdr' en [17]. Se podría realizar un análisis de esta librería y reimplementar los módulos que dan problemas, tales como, la utilización de pequeños anchos de banda, el error en cabeceras WAV y el problema de control del proceso para el cambio de frecuencia en tiempo real. Poder implementar el cambio de frecuencia en tiempo real para evitar el efecto doppler nos facilitaría la recepción de la señal APT y su decodificación.

Además de esto, el servicio WEB creado podría mejorarse utilizando Apache en lugar del microframework Flask, permitiendo que la aplicación WEB soporte las peticiones de

un mayor número de usuarios. En estos momentos está limitada a muy pocos usuarios. Dado que el navegador actualiza la información del *Tracking* del satélite cada segundo, un número alto de usuarios crea un cuello de botella impidiendo que la aplicación muestre toda la información y produciéndose retardos en la carga del servicio desde el navegador. Este aspecto podría optimizarse implementando la WEB con cliente-servidor más ligados entre sí, como son Node.js y Javascript

Otra posible línea de trabajo es, a partir de este desarrollo, trasladarlo a la recepción de otros satélites de órbita polar, adaptando el módulo de recepción. Habría que adaptar la antena para la frecuencia de downlink del nuevo satélite, adaptar los comandos de recepción y revisar si el receptor SDR es el adecuado para la frecuencia de sintonización, además de descargar los TLE del satélite a escuchar. El desarrollo modular del sistema haría que no fuera necesaria la re-implementación completa del sistema.

Por último, sobre las imágenes meteorológicas obtenidas se puede abrir una línea de trabajo en procesado de imagen, consiguiendo la mejora de esta con tratamiento de señal y consiguiendo la identificación de fenómenos meteorológicos a partir de patrones determinados.

8.3 Conclusiones

Se ha llevado a cabo un proyecto de ingeniería de sistemas en el que se ha alcanzado un producto final. GranaSAT ha ofrecido la posibilidad de desarrollar un producto poniendo en manos del alumno sus equipos e instalaciones para optimizar su trabajo. Con este proyecto, que ha permitido trabajar en desarrollo software y en hardware, se han empleando herramientas y metodologías que han conseguido aumentar los conocimientos, aplicar los adquiridos en la formación académica y favorecer el crecimiento profesional.

Con este proyecto he aprendido nuevos lenguajes de programación, nuevos software de ingeniería que no se han empleado durante la carrera y nuevos conceptos del sector aeroespacial con los que las telecomunicaciones están muy ligadas. Por lo tanto, se puede afirmar que el Trabajo Fin de Máster desarrollado ha cubierto los objetivos principales del alumno, que son demostrar conocimientos adquiridos y aumentar sus capacidades como ingeniero.

Finalmente destacar que, gracias a este proyecto y a la posibilidad de implementarlo como parte del equipo GranaSAT, he descubierto un gran interés por el sector aeroespacial y en particular por la ingeniería de antenas, abriéndome las puertas en el mundo laboral. El poder trabajar en diferentes ramas de las telecomunicaciones durante el desarrollo del proyecto me ha permitido orientar mi futuro profesional desde el comienzo hacia mis preferencias. Además, destaco de este tiempo empleado en el desarrollo del proyecto, el gran compañerismo de todo el equipo GranaSAT.

ANEXO

A

DATASHEET RG-58

1

GESCABLE

RG058

RG-Cables acc. to MIL-C-17F and MIL-C-17G

Application

see product overview

Standards

acc. to MIL-C-17F, MIL-C-17G

Flame resistance

acc. to IEC 60332-1

Construction

Inner conductor	stranded copper wires, tinned, diameter 0.90 ± 0.01 mm
Insulation	PE, diameter 2.95 ± 0.05 mm
Outer conductor	
Copper braid	tinned, 96% optical coverage
Sheath	PVC, altern. FRNC, diameter 4.95 ± 0.10 mm black

Mechanical properties

Minimum bending radius	without load	5 x outer diameter
	with load	10 x outer diameter
Temperature	during operation	-40° C to + 85° C
	during installation	-15° C to + 55° C
Corrosivity	only for FRNC type	IEC 60754-2

Electrical properties

at 20°C

Loop resistance		≤ 50 Ω
Characteristic impedance		50 Ω ± 2 Ω
Velocity ratio		66 %
Mutual capacitance		100 nF/km
Transfer impedance		36 mΩ/m
Operating voltage		1.8 kV_{rms}
Test voltage	Inner/Outer conductor	5.4 kV_{rms}

GESCABLE

RG058

Electrical data

at 20°C

Frequency (MHz)	Attenuation (dB/100m)	Max. power rating (Watts) (at ambient temperature 25°Cand max. inner conductor temperature of 70°C)	Return loss (dB) several peaks are allowed	
	nominal	maximum		
10	4.2	750	**Frequency (MHz)**	
100	15.7	230	50-100	≥ 28
200	23.0	180	100-300	≥ 27
400	34.5	110	300-500	≥ 26
1000	60.0	65	500-1000	≥ 25

All other requirements acc. to MIL-C-17F, MIL-C-17G

Technical data

Product code	Desig-nation	Type	Brand name	Outer diameter	Weight	Standard delivery length	Drum size	Gross weight	Copper content	Tensile force
				mm	kg/km	m	*PWD/ring	kg		
1002717	2YCY	0.9Lz/ 2.95z	M17/28-RGO58	4.95	37	1000/100	400/120/ 280	39/3.7	20.4	120
1002919	2YCH	0.9Lz/ 2.95z	M17/28-RG058 FRNC	4.95	38	1000/100	400/120/ 280	40/3.8	20.4	120

*PWD (Plywood drum)

1

ANEXO

B

CALIBRACIÓN RTLSDR

Calibrar el dispositivo *RTLSDR* consiste en calcular su ppm, de manera que ajuste el oscilador para sintonizar la frecuencia RF deseada. Existen diversos modos de calibración que se describen a continuación:

1. **Calibración haciendo uso del software *Kalibrate*:**

   Este software está disponible para Linux y Windows en [21]. Se ha descargado la versión para Windows.

   Este software es una aplicación de consola, así que, para comenzar con la calibración, nos dirigimos desde la terminal de Windows al directorio dónde se ha guardado dicho software descargado. El dispositivo *RTLSDR* debe estar conectado al PC y a su correspondiente antena. *Kalibrate* usa la señal de una estación móvil Global System for Mobile communications (GSM), por lo que comienza escaneando los canales y buscando aquel de mayor potencia. Permite escanear las bandas GSM850, GSM900, EGSM, DCS and PCS. Solo es necesario encontrar un canal con una elevada potencia, por lo que no se requiere escanear todas las bandas.

   A continuación se muestran las opciones del comando:

   Se escanea en algunas bandas GSM seleccionando una ganancia alta y un ppm inicial para el dispositivo. Se espera a la respuesta y se escoge el canal que mayor potencia reciba. En la Figura B.1 se observa los resultados obtenidos cuando se ejecuta la aplicación para dos bandas diferentes, GSM900 y EGSM, permitiendo seleccionar el canal 117 para nuestra calibración.

Opción	Descripción
-s	Banda de escaneo (GSM850, GSM900, EGSM, DCS, PCS)
-f	Frecuencia cercana a la banda
-c	Canal de la estación base GSM
-b	Band indicator (GSM850, GSM900, EGSM, DCS, PCS)
-g	Ganancia en dB
-d	Indicador de dispositivo RTLSDR
-e	Error de frecuencia inicial en ppm
-v	Verbose
-D	Debug activo
-h	Ayuda

Tabla B.1 – *Opciones para el comando kalibrate*

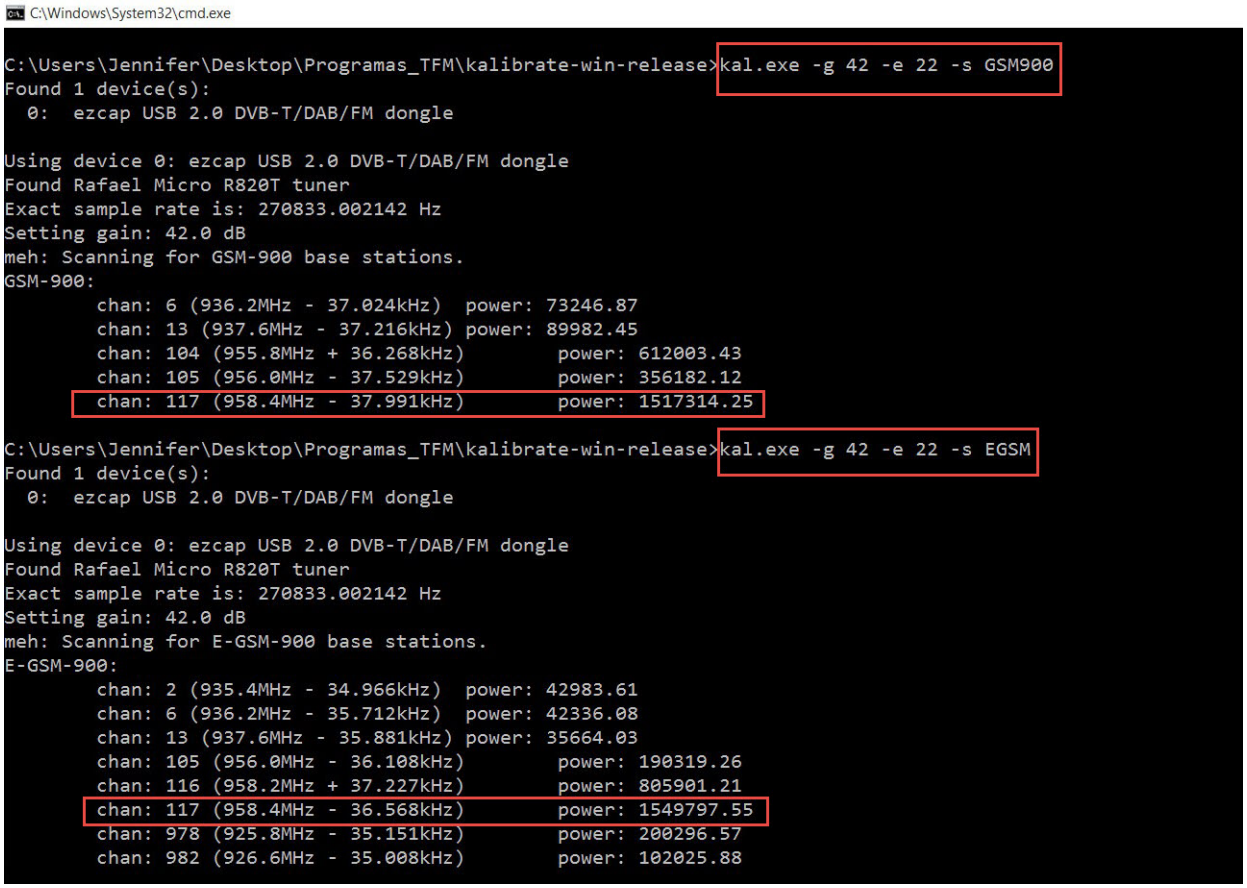


Figura B.1 – *Selección del canal con máxima potencia de recepción.*

RTLSDR no ha encontrado canales transmitiendo para la banda GSM850. Para las bandas DCS y PCS, nuestro RTLSDR no está sintonizado, son frecuencias demasiado altas. Esto se puede comprobar en la imagen B.2.

A continuación, seleccionando el canal 117, es posible calcular el offset (en ppm) de

```
C:\Users\Jennifer\Desktop\Programas_TFM\kalibrate-win-release>kal.exe -g 42 -e 22 -s GSM850
Found 1 device(s):
  0:  ezcap USB 2.0 DVB-T/DAB/FM dongle

Using device 0: ezcap USB 2.0 DVB-T/DAB/FM dongle
Found Rafael Micro R820T tuner
Exact sample rate is: 270833.002142 Hz
Setting gain: 42.0 dB
meh: Scanning for GSM-850 base stations.
GSM-850:

C:\Users\Jennifer\Desktop\Programas_TFM\kalibrate-win-release>kal.exe -g 42 -e 22 -s DCS
Found 1 device(s):
  0:  ezcap USB 2.0 DVB-T/DAB/FM dongle

Using device 0: ezcap USB 2.0 DVB-T/DAB/FM dongle
Found Rafael Micro R820T tuner
Exact sample rate is: 270833.002142 Hz
Setting gain: 42.0 dB
meh: Scanning for DCS-1800 base stations.
[R820T] PLL not locked for 1808770000 Hz!
error: usrp_source::tune

C:\Users\Jennifer\Desktop\Programas_TFM\kalibrate-win-release>kal.exe -g 42 -e 22 -s PCS
Found 1 device(s):
  0:  ezcap USB 2.0 DVB-T/DAB/FM dongle

Using device 0: ezcap USB 2.0 DVB-T/DAB/FM dongle
Found Rafael Micro R820T tuner
Exact sample rate is: 270833.002142 Hz
Setting gain: 42.0 dB
meh: Scanning for PCS-1900 base stations.
[R820T] PLL not locked for 1933770000 Hz!
error: usrp_source::tune
```

Figura B.2 – *Error en bandas DCS y PCS.*

nuestro dispositivo (Figura B.3).

```
C:\Windows\System32\cmd.exe
C:\Users\Jennifer\Desktop\Programas_TFM\kalibrate-win-release>kal.exe -e 41 -c 117 -v
Found 1 device(s):
  0:  ezcap USB 2.0 DVB-T/DAB/FM dongle

Using device 0: ezcap USB 2.0 DVB-T/DAB/FM dongle
Found Rafael Micro R820T tuner
Exact sample rate is: 270833.002142 Hz
meh: Calculating clock frequency offset.
Using GSM-900 channel 117 (958.4MHz)
        offset   1: -18454.65
        offset   2: -18462.92
        offset   3: -18468.08
        offset   4: -18473.25
```

Figura B.3 – *Comando para el calculo del error de frecuencia en ppm.*

Hay que prestar especial atención en que el resultado del offset en ppm aumenta con el dispositivo en 'caliente', es decir, cuando lleva un tiempo prolongado operativo. Es por ello que, no se debería tomar esta medida cuando esté recién activado. En la Figura B.2 se observa el resultado final.

2. **Calibración con SDR#:**

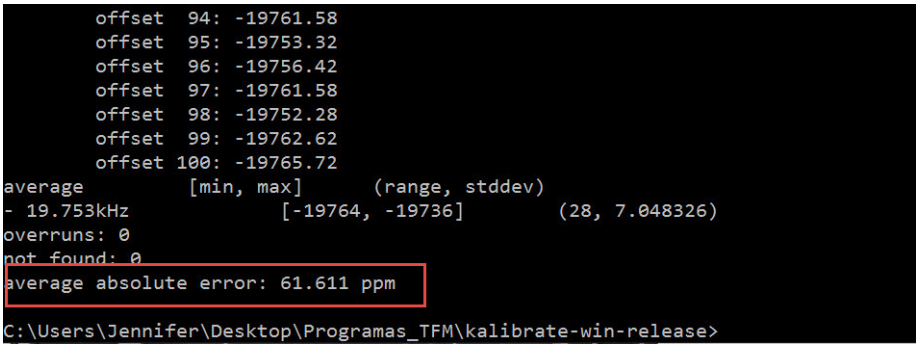


Figura B.4 – *Resultado de error de frecuencia.*

Es posible calibrar el dispositivo directamente con el software *SDR_*. Con el dispositivo RTLSDR conectado a nestro PC y un cable coaxial conectado a un generador de señal que genere un tono, es posible calibrar el equipo.

En este caso se ha empleado un generador de comunicaciones *Marconi Instrument R2955A*. Se genera una señal a una frecuencia RF de 300 MHz y un nivel de -100 dBm. En la la Figura B.5 se puede ver el tono generado. Can ayuda de la configuración de *SDR#* se modifica la opción de error de frecuencia en ppm hasta conseguir que la frecuencia oscilador coincida con la frecuencia central (véase ajuste en la Figura B.6.

Figura B.5 – *Señal generada con equipo de comunicaciones.*

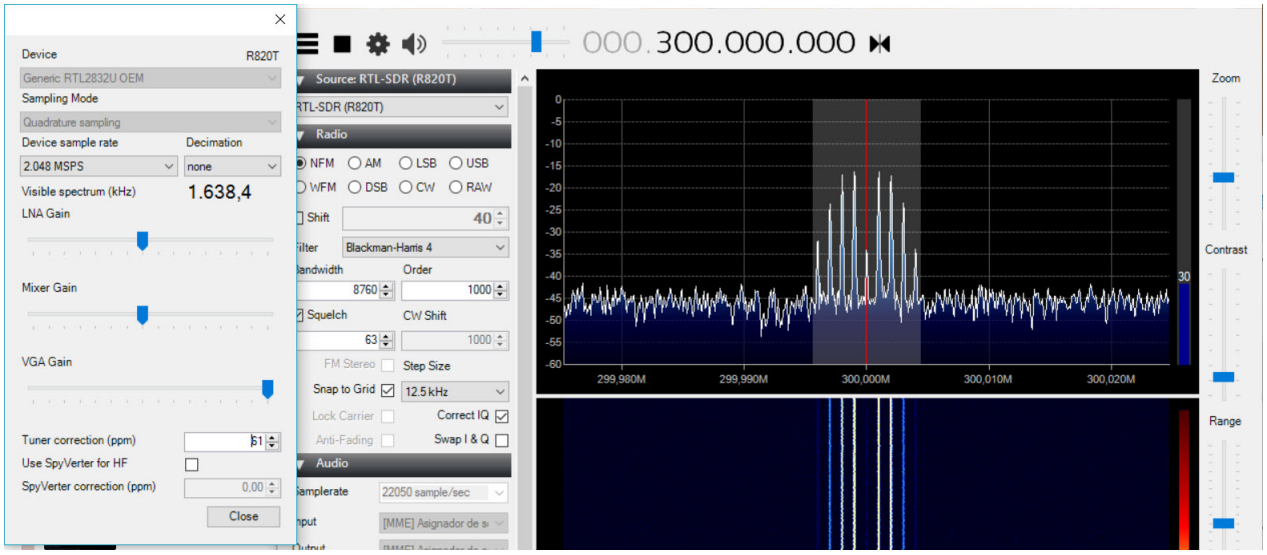


Figura B.6 – *Ajuste de ppm con SDR#.*

Se obtiene 61 ppm, coincidiendo con la calibración anterior.

3. **Calibración con RTL_TEST**

   La librería librtlsdr disponible en [17] para su descarga provee una serie de comandos de control de *RTLSDR* muy útiles. Uno de ellos es el comando ***rtl_test -p***, mediante el cual se consigue obtener el offset del dispositivo. Es la forma más rápida, pero menos precisa, ya que el resultado oscila entre valores próximos al ofset real.

2

ANEXO

C

INSTALACIÓN DE ENTORNO WEB CON PYTHON

Para crear el servicio con Python es necesario instalar una serie de paquetes. El servicio correrá en SO Linux. A continuación se muestra las instalaciones y configuraciones necesarias.

1. **Instalación del paquete Python:** Para nuestro servicio se ha usado la versión de Python 2.7 (podría usarse Python 3.4. Si se trabaja en Windows es necesario añadir una nueva variable de entorno con la ruta *C:/Python27* en el campo *Path* de *Variables de entorno*.

   Con la herramienta *PIP*, herramienta para gestión de paquetes de Python, será más sencillo trabajar, por ello también la instalamos en nuestro servidor:

   $ sudo apt-get install python-pip python-dev build-essential

   $ sudo pip install –upgrade pip

2. **Instalación de entorno virtual:** A partir del gestor de paquetes PIP, se instala *Virtualenv*, como se muestra a continuación. El objetivo es aislar nuestro servicio del sistema principal, que puede tener librerías o recursos que entren en conflictos con las de nuestro servicio.

   $ sudo pip install virtualenv

   $ mkdir webTracking - Se crea el directorio de proyecto llamado *webTracking*.

$ cd webTracking

$ virtualenv venv - Se crea el entorno virtual.

En *venv* se encuentran los directorios *bin/, lib/ e include/*. En el directorio bin/ se encuentran los ejecutables necesarios para interactuar con el entorno virtual. En include/ se encuentran algunos archivos de cabecera de C (con extensión es .h) necesarios para compilar algunas librerías de Python. Y finalmente en lib/ se encuentra una copia de Python así como un directorio llamado site-packages/ en el cual se almacenan los paquetes Python instalados en el virtualenv.

Seguidamente, se activa en entorno virtual de Python, de manera que ya se puede trabajar en él, creando script, instalando nuevas herramientas (que serán solamente accesibles desde virtualenv) y usando las instaladas dentro de virtualenv.

$ cd webTracking

$. venv/bin/activate, o bien, *$ source venv/bin/activate*

(venv)$ - Este prompt indica que se está dentro del entorno virtual. Para salir de él simplemente se escrive el comando *deactivate*.

En el directorio del proyecto creamos diferentes carpetas que van a servir para organizar nuestra aplicación web. Las carpetas principales son *'templates'*, donde se colocarán los ficheros HTML y la carpeta *'static'* que servirá para guardar librerías .js y .css y donde se crea la carpeta *'static/Images'*para almacenar las imágenes de satélites.

3. **Instalación de Flask:** Dentro de nuestro entorno virtual se instala el framework ligero Flask.

   $ pip install Flask

   Para el servicio creado, este framework será suficiente. Siempre se puede montar sobre Apache fácilmente el servicio usando el módulo *WSGI Apache*.

4. **Instalación de paquetes:** Los paquetes y librerías de trabajo son instaladas desde el entorno virtual. Después de ello se importan los paquetes en el archivo de texto *'requirements.txt*. Con este archivo es posible mover el servicio de máquina e instalar los paquetes que han sido necesarios en su creación desde *'requirements.txt*.

   (venv)$ pip freeze > requirements.txt - Se importan los paquetes que se han instalado.

   (venv)$ pip install -r requirements.txt - Instalación de paquetes en nuevo entorno virtual.

Gracias a estas pautas, nuestro servicio podrá ser puesto en funcionamiento rápidamente. Los pasos serán similares para Raspberry Pi

ANEXO

D

INSTALACIÓN Y USO DE PM2

Se ha decidido trabajar con PM2 por la fácil gestión que realiza sobre un servicio web. Aunque inicialmente está indicado para aplicaciones Node.js, puede ser empleado para aplicaciones con Python indicando en la cabecera del script el interprete de Python usado. En nuestro caso se indica:

#!/home/jenifer/webTacking/venv/bin/python

Pasa su instalación, los pasos a seguir son los siguientes:

1. Instalación de node.js y npm: *$ sudo apt-get install nodejs npm*
2. Instalación del gestor de procesos: *$ npm install pm2@latest -g*

Con estos paquetes ya se puede puede gestionar cualquier servicio. Se crea en la instalación el directorio *.pm2* que contiene las carpetas *pids* y *logs*. Estas carpetas almacenan el pid de cada proceso que gestiona PM2 y los mensajes de información y de error de los procesos. De una manera cómoda se puede acceder a ellos.

Desde la terminal se puede controlar el servicio con los siguientes comandos:

$ pm2 start wbServer.py - Inicia el proceso y le de un identificador numérico.

$ pm2 status - Imprime una lista de los procesos abiertos con pm2 y muestra su estado (on, off).

$ pm2 stop 0 - Para el proceso con identificador 0. También podría usarse su nombre directamente.

$ pm2 restart 0 - Reinicia proceso.

$ pm2 logs - Muestra los mensajes *logs*.

$ pm2 flush - Elimina mensajes logs.

$ pm2 delete 0 - Elimina el proceso y con ello sus archivos logs y pid.

De este modo, en cualquier momento se puede ver el estado del servicio con una simple línea en la terminal del servidor, al que se puede acceder con *ssh*.

ANEXO

E

PRESUPUESTO

E.1 Costes de electrónica

A continuación se plasman los costes de materiales de electrónica para cada una de las partes integrantes en el sistema. Principalmente estos costes han ido a parar a los equipos integrantes en el sistema receptor.

Componente	**Coste(€)**
Raspberry Pi 3B	30.16€
Memoria microSD 16 Gb	4.95€
Cargador Raspberry	6.6€
RTLSDR	7.4€
+IVA(21%)	10.31€
TOTAL	**59.42€**

Tabla E.1 – *Coste del receptor Raspberry Pi + SDR*

Además del equipo receptor propiamente, para el sistema de recepción al completo ha sido necesaria la fabricación de la antena. En este caso han sido fabricadas dos antenas receptoras. Su fabricación ha supuesto gastos en componentes electrónicos.

Componente	**Coste(€)**
Conector TV (x2)	5.2€
Conector N	7.69€
Cable 50Ω($10m$)	9.1€
+IVA(21%)	4.62€
TOTAL	**26.61€**

Tabla E.2 – *Coste de componentes electrónicos en antenas*

E.2 Costes de mecánica

Las antenas Eggbeater y Turnstile fabricadas han tenido unos costes en mecánicas mostrados en la siguiente tabla.

Componente	**Coste(€)**
Reflectores de acero (5m)	13.45€
Tubos de PVC (varios)	15.7€
Manguitos PVC	1.89€
Tapones PVC	2.15€
+IVA(21%)	6.97€
TOTAL	**40.16€**

Tabla E.3 – *Coste de componentes mecánicos en antenas*

5

E.3 Costes Software

En la siguiente tabla se muestra el coste de licencias adquiridas.

Software	Propietario de licencia	**Coste(€)**
CST	GranaSAT	Free
4NEC2	Jennifer López	Free
SolidWorks	GranaSAT	Free
ADS	Departamento de electrónica	Free
Matlab	Departamento de Electrónica	Free
Microsoft Visio 2013	UGR	Free (DreamSpark)
Microsoft Excel 2013	UGR	Free (DreamSpark)
Miktex	Jennifer Lóez	Free license
Microsoft Project	Jennifer López	Trial license
Putty	Jennifer López	Free license
Mobaxterm	Jennifer López	Free license
Nuhertz Filter Solutions v14.0	Jennifer López	Free license
Gpredict	Jennifer López	Free license
Orbitron	Jennifer López	Free license
SDR#	Jennifer López	Free license
WXtoImg	Jennifer López	Free license
SumatraPDF	Jennifer López	Free license
TeXnicCenter	Jennifer López	Free license
	TOTAL	0

Tabla E.4 – *Costes Software*

E.4 Recursos humanos

Se considera que este proyecto ha tenido una duración de aproximadamente un año y medio. El trabajo ha sido desarrollado por un ingeniero junior. En la siguiente Tabla se refleja el salario correspondiente al empleado si su jornada laboral es de 5 horas diarias.

Salario	10 €/h
Días trabajados	378 días
Horas trabajadass	1890 horas
TOTAL	**18900 €**

Tabla E.5 – *Coste de empleado ingeniero junior.*

Este ingeniero ha estado bajo la supervisión de un ingeniero senior. Si se fija las horas trabajadas por el ingeniero senior en dicho proyecto en 2 horas semanales, el coste del empleado se refleja a continuación:

Salario	50 €/h
Semanas trabajadas	52 semanas
Horas trabajadas	156 houras
TOTAL	**78200 €**

Tabla E.6 – *Coste de empleado ingeniero senior*

El total en recursos humanos ha sido:

Ingeniero Junior	18900 €
Ingeniero Senior	5200 €
TOTAL	**26700 €**

Tabla E.7 – *Coste total en recursos humanos*

REFERENCIAS

[1] Mapbox. Website. https://www.mapbox.com/.

[2] Noaa, 'noaa heritage missions'. http://poes.gsfc.nasa.gov/noaa-heritage.html.

[3] Openstreetmap. Website. https://www.openstreetmap.org/#map=1/72/-128.

[4] Noaa, 'noaa polar orbiter data user's guide', Agosto 2008. http://www.ncdc.noaa.gov/oa/pod-guide/ncdc/docs/podug/index.htm.

[5] Noaa klm user' guide: Advanced very high resolution radiometer/3. NOAA KLM Website, March 2009. http://www.ncdc.noaa.gov/oa/pod-guide/ncdc/docs/klm/html/c3/sec3-1.htm.

[6] Parametros s y carta de smith. Vida Teleco, Febrero 2009. https://vidateleco.wordpress.com/2009/02/16/parametros-s-y-carta-de-smith-parte-1/.

[7] Noaa klm user' guide: Apt. NOAA KLM Website, Noviempbre 2013. http://www.ncdc.noaa.gov/oa/pod-guide/ncdc/docs/klm/html/c4/sec4-2.htm.

[8] An api and web application to interact with a running instance of rtl_fm. github. Website, Junio 2014. https://github.com/th0ma5w/rtl$_$fm$_$python.

[9] The seven (or eight) keplerian elements. AMSAT, 2014. http://www.amsat.org/amsat-new/tools/keps_detail.php.

[10] 4nec2. NEC Website, Diciembre 2015. http://www.qsl.net/4nec2/.

[11] Pm2. production process manager for node.js. Website, 2015. http://pm2.keymetrics.io/.

[12] sgp4 python. Website, 2015. https://pypi.python.org/pypi/sgp4/.

[13] Sound exchange. Website, 2015. http://sox.sourceforge.net/.

[14] Wxtoimg. software to decode apt and wefax signals from weather satellites. Website, 2015. http://www.wxtoimg.com/beta/.

[15] Command line utility that takes iq data stream as input and produces doppler corrected output stream based on tle. github. Website, Marzo 2016. https://github.com/cubehub/doppler.

[16] Cst studio suite. CST Website, 2016. https://www.cst.com/.

[17] Osmocom. rtl-sdr. Website, 2016. http://sdr.osmocom.org/trac/wiki/rtl-sdr.

[18] Poes operational status. POES Web, April 2016. http://www.ospo.noaa.gov/Operations/POES/status.html.

[19] Pypredict python. Website, 2016. https://pypi.python.org/pypi/pypredict/1.2.

[20] Realtek r2832u. Website, 2016. http://www.realtek.com.tw/products/.

[21] Rtl-sdr.sceners.org. kalibrate-rtl. website, may 1st 2016. Website, 2016. http://rtl-sdr.sceners.org/?p=193.

[22] Celestrack, Enero 2017. https://celestrak.com/.

[23] Developer survey results 2017. stack overflow. Website, 2017. https://insights.stackoverflow.com/survey/2017.

[24] Leaflet. Website, Agosto 2017. http://leafletjs.com/.

[25] Tiobe index for july 2017. Website, 2017. https://www.tiobe.com/tiobe-index//?6671423=1.

[26] Balanis, C. A. Antenna theory. analysis and design, 2005.

[27] D. Vallado, P. C. y. e. a. Revisiting spacetrack report 3. *AIAA 2006-6753* (Agosto 2006).

[28] Fundation, P. S. Multiprocessing - process-based 'threading' interface. Website, 2016. https://docs.python.org/2/library/multiprocessing.html#module-multiprocessing.

[29] Fundation, P. S. Threading - higher-level threading interface. Website, 2016. https://docs.python.org/2/library/threading.html.

[30] Keen, K. Rtl_fm guide. Website, 2017. http://kmkeen.com/rtl-demod-guide/.

[31] Keenerd. Rtl-sdr experimental branch github. Website, 2016. https://github.com/keenerd/rtl-sdr.

[32] Laufer, C. The hobbyists guide to rtl-sdr: Really cheap software defined radio. http://www.qsl.net/yo4tnv/docs/The%20Hobbyists%20Guide%20To%20RTL-SDR%20-%20Carl%20Laufer.pdf.

[33] Mbaka, K. S. E. Weather satellite imaging system. Final Report for EECS 502, Mayo 2006.

[34] Milligan, T. A. Modern antenna design, 2005.

[35] Rhodes, B. C. Pyephem home page. Website, 2008. http://rhodesmill.org/pyephem/.

[36] Ricalde, E. M. Antenas lineales con alimentacion simetrica. Documents.mx, Febrero 2015. http://documents.mx/documents/antenas-lineales-con-alimentacion-simetrica-imprimir.html.