UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE UPV INGENIEROS DE TELECOMUNICACIÓN

Desarrollo de un terminal AIS lowcost para seguimiento de barcos

Autor: Héctor Delás Castellá

Tutor: Dr. Carlos Enrique Palau Salvador

Fecha: 25 de junio de 2018.

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2017-2018

Valencia, 24 de julio de 2014

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

VLC/ CAMPUS VALENCIA, INTERNATIONAL CAMPUS OF EXCELLENCE

EMAS Gestión ambiental verificada REG.NO. ES-CV- 000030

Resumen

En este trabajo final de grado se ha abordado el diseño, simulación e implementación de una aplicación software para la geolocalización de barcos cercanos a la costa, el volcado y análisis de la información a una base de datos y su representación, además de la adaptación hardware para la apropiada recepción mediante una antena, un receptor AIS y una Raspberry Pi que se usará a modo de decodificador y terminal de la aplicación.

Con el fin de conseguir un terminal AIS de bajo coste, se han utilizado elementos baratos y accesibles aunque limitados para operaciones y requisitos de mayor exigencia. Todo este conjunto se ha probado en condiciones reales cumpliendo los requerimientos y siguiendo la metodología de Marine Traffic logrando no solo un decodificador sino un terminal que logra mostrar la posición y la información más relevante de los navíos gracias a los dispositivos AIS incorporados con mínimas pérdidas en la información.

Resum

En aquest treball final de grau s'ha abordat el disseny, simulació i implementació d'una aplicació software per a la geolocalización de barcos pròxims a la costa, la bolcat i anàlisi de la informació a una base de dades i la seua representació, a més de l'adaptació hardware per a l'apropiada recepció per mitjà d'una antena, un receptor AIS i una Raspberry Pi que s'usarà a manera de descodificador i terminal de l'aplicació. A fi d'aconseguir un terminal AIS de baix cost, s'han utilitzat elements barats i accessibles encara que limitats per a operacions i requisits de major exigència. Tot este conjunt s'ha provat en condicions reals complint els requeriments i seguint la metodologia de Marine Traffic aconseguint no sols un descodificador sinó un terminal que aconseguix mostrar la posició i la informació més rellevant dels navilis gràcies als dispositius AIS incorporats amb mínimes pèrdues en la informació.

Abstract

In this final project of degree it has been carried out the design, simulation and implementation of a software application for the geolocation of ships near coast, the dump and analysis of information to a database and its representation, in addition the hardware adaptation for the appropriate reception by means of an antenna, an AIS receiver and a Raspberry Pi that will be used as decoder and terminal for the application.

In order to achieve a low cost AIS terminal, cheap and accessible elements have been used, although limited for more demanding operations and requirements. All this set has been tested in real conditions fulfilling the requirements and following the methodology of Marine Traffic achieving not only a decoder but a terminal that manages to show in real time the position and the most relevant information of the ships thanks to the AIS devices incorporated with minimal losses in information.

1.3 Estructura

Capítulo 1 Introducción y objetivos

1.2 Introducción

Debido a la situación actual la información es un bien muy preciado, en este sentido las telecomunicaciones realizan el papel de transmitir la información de manera fiable, autenticada y completa. Cada vez aparecen más aplicaciones para transmitir y recibir puesto que la base de todas las telecomunicaciones es esa información que se está transmitiendo de un punto a otro.

En el marco de las comunicaciones marítimas, una simple transmisión de información permite ubicar de manera exacta un barco, saber el tipo de barco que es, el destino, las dimensiones de este etc.

Esto no solo puede utilizarse para saber dónde se encuentra cada barco a modo de radar por ejemplo, si se realiza un análisis más exhaustivo podemos llegar a predecir la velocidad exacta a la que debe ir un barco desde cierto punto para entrar a puerto sin tener que esperar a que una dársena quede libre, lo que permite a las portuarias y las empresas encargadas de los barcos ahorrarse miles de euros.

Todo esto se realiza mediante tramas AIS donde no solo se transmite de manera codificada la posición del barco sino otros datos relevantes que podrán utilizarse para diversos fines, sin embargo no se utiliza tecnología demasiado novedosa y complicada por ello se plantea utilizar un terminal de bajo coste como es la Raspberrry Pi que a pesar de no tener una gran capacidad de procesamiento permite funcionar como receptor y a la vez gestionar los datos recibidos.

Por todo ello la metodología es la que se lleva utilizando en las comunicaciones marítimas desde hace años, sin embargo AIS facilita los estándares para la decodificación de sus tramas y permite recibir de manera muy sencilla los datos transmitidos por los barcos puesto que solo necesitas una antena y su receptor.

Finalmente este trabajo está enfocado a la recepción de la información y a su tratamiento, en concreto al almacenaje en una base de datos local, guardada en la Raspberry y una interfaz simple para mostrar la posición actual de los barcos captados y la trayectoria de cada uno.

1.2 Objetivos

El objetivo de este trabajo es conseguir diseñar un receptor AIS mediante elementos de bajo coste totalmente operativo para crear una base de datos de los barcos cercanos a la costa y de las estaciones base para su posterior representación. Para lograr este objetivo se ha dividido el trabajo en cinco metas principales:

- El primer objetivo es conseguir captar tramas desde los barcos para ello necesitamos una antena que se adecue a los requisitos necesarios

- Paralelamente es necesario conseguir su decodificación de las tramas más significativas por lo que necesitaremos programar un decodificador e instalar todos los módulos y herramientas que se requieran en la Raspberry.

- A continuación, se debe diseñar una base de datos para almacenar los datos provenientes de las tramas.

- Posteriormente, debemos encontrar una manera de mostrar en un mapa los datos de los barcos de forma dinámica y realizar una interfaz que simplifique el proceso.

- Por último debemos poner a prueba a la Raspberry Pi para saber hasta cuantas tramas podría soportar a la vez.

Capítulo 2 Estándar AIS y especificaciones técnicas

2.1 Antecedentes del estándar AIS en las comunicaciones marítimas.

En 1991 se presenta el primer documento acerca de las especificaciones del que sería el estándar AIS(Automatic Identification System) sistema de identificación automática, se trata de un estándar internacional y de libre uso, bastante adoptado actualmente por las embarcaciones en el ámbito marítimo, este estándar permite conocer la posición de los barcos de nuestro entorno y obtener una amplia información sobre ellos, como su nombre, velocidad, rumbo, MMSI, estado actual de navegación y mucha más información.

Pese a haber sufrido diversas modificaciones no fue hasta 2002 cuando la IMO (International Maritime Organization) con el objetivo de estandarizar y regularizar las normas adoptadas por el convenio SOLAS (Safety Of Life At Sea) exigió el uso obligatorio de AIS para las embarcaciones con un peso mayor a 500 toneladas, los buques que superen las 300 toneladas y realicen travesías internacionales y cualquier embarcación de pasajeros. Quedan exentos los barcos militares, barcos de recreo con una eslora inferior a 45 metros y los pesqueros.

Además fue incorporado al sistema GMDSS (Global Maritime Distress and Safety System) vigente desde 1999, el cual está integrado por un conjunto de equipos y protocolos con el objetivo de incrementar la seguridad en los viajes marítimos y en caso de emergencia reducir el tiempo de rescate.

El estándar AIS presenta una serie de ventajas por lo que fue escogido para ser usado en embarcaciones profesionales:

Es fácil de implementar, mantener y actualizar ya que es una decodificación sencilla y no requiere un gran procesamiento.

Bajo o inexistente coste operativo.

Estandarización internacional, como se ha mencionado anteriormente este estándar se ha implementado en casi todas las comunicaciones marítimas.

Cobertura entre 20 millas náuticas y 100, dependiendo de la altura de las antenas y obstáculos.

Robustez frente a las condiciones meteorológicas adversas, debido a la longitud de onda en la que trabaja y la simplicidad de la codificación lo hace ideal para el uso en comunicaciones marítimas.

Por todas estas ventajas poco a poco se está implementando cada vez más en barcos pesqueros ya que puede considerarse como un elemento más de seguridad del barco.

Una de las ventajas que le ha hecho ganar terreno estos últimos años es que debido a la altura de los radares en las embarcaciones grandes o a acantilados puede causar que los navíos más pequeños no sean detectados y por ello se requiere de otro sistema para su detección como radares de onda media y corta, sin embargo AIS emite dentro de la banda de VHF por lo que no importa los obstáculos ya que al no ser que haya una gran distancia y esté fuera del alcance estas pequeñas embarcaciones serán detectadas. Por último uno de los grandes inconvenientes y a la vez ventaja de los radares convencionales es que es complicado identificar los blancos ya que pueden confundirse con rocas, faros etc... Sin embargo AIS simplemente identifica a los barcos pero aparto más información que un radar sobre estos

Otra de las cualidades es la robustez que ha adquirido el sistema desde 2004, fecha en la comenzo a implantarse es que a diferencia de por ejemplo del GSM es que los fallos en las estaciones base no implican una caida total del sistema, pues cada estación es autogestionada y esto es debido gracias al esquema de multiplexación temporal SOTDMA(Self-Organized Time Division Multiple Access) el cual permite ordenar los mensajes y manejar más de 4500 mensajes por minuto y sus correspondientes actualizaciones cada 2 segundos.

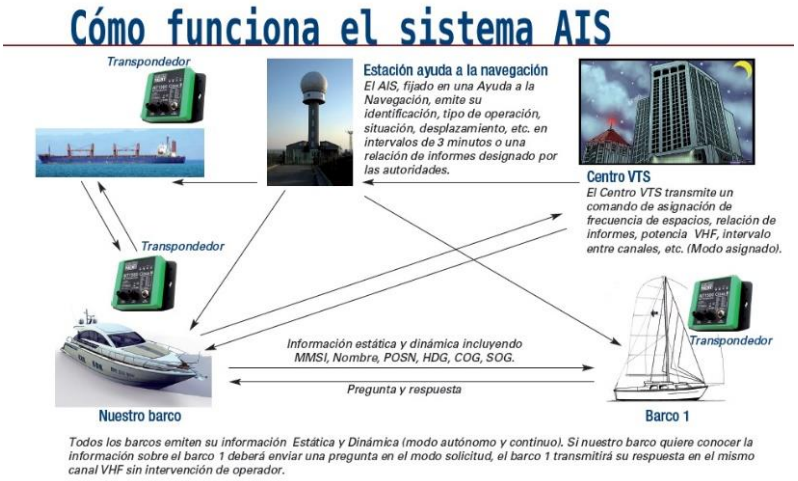

Figura 1. Funcionamiento del sistema AIS.

En la actualidad el sistema AIS esta provisto de un servicio satelital para la identificación de buques y presenta cobertura global por lo que los satelites actuán a modo de estación base en las zonas donde hay cobertura escasa o lejos de la costa, como son el caso de Orbcomm y exactEarth que redireccionan los mensajes hasta las estaciones base AIS que a la vez transmiten los datos que pueden ser utilizados por ejemplo para salvamento marítimo.

2.2 Clasificación y esquema del estándar AIS

En este apartado vamos a profundizar en las cuestiones técnicas del estándar AIS para saber que requesitos hay que cumplir, en que condiciones y como aprovecha el proyecto este sistema.

El estándar AIS está recogido en la recomendación ITU-R M.1371-5. En esta recomendación de la ITU se establece en primer lugar las características del estándar y de la banda de frecuencia

utilizada VHF correspondiente a la banda marítima móvil, a continuación se explica la técnica de multiplexación utilizada SOTDMA que combina TDMA (Time Division Multiplex Access) y por último especifica elementos importantes de la autenticación de los mensajes y la seguridad relativa a estos.

Esta sección del capítulo va a estar centrada en explicar las diferentes capas y protocolos del estándar, su arquitectura y su esquema de modulación.

2.2.1 Esquemas de multiplexación.

En las comunicaciones que se realizan mediante el sistema AIS los barcos emiten su informe consecutivamente por un canal común ya que cuando existe un gran tráfico de estos no es posible asignar un canal para cada barco lo cual supondría muchos problemas ya que podrían suponer interrupciones, necesidad de regulación y una multiplexación compleja para poder escuchar a la vez.

Los tiempos de emisión son de aproximadamente de 3 centésimas de segundo, por ello cuando son muchos los participantes el canal puede verse saturado por ello al igual que se ha utilizado en telefonía móvil el tipo de modulación es TDMA (Time Division Multiple Access).

De modo que este principio solo podría aplicarse y funcionar si existe una sincronización temporal exactamente determina y válida para todos los transpondedores AIS, es decir que la sincronización garantice que no hay emisores transmitiendo en la misma frecuencia.

Por ello para garantizar esta sincronización el reloj de los barcos están ajustados a la milésima y la exactitud de la hora suministrada por el GPS garantiza que todos los barcos participantes sepan cuando empieza y termina un slot de tiempo.

Por lo tanto esa división de las tramas en el tiempo no permite tener un solo canal sino que tienen 2250 slots o rendijas en un minuto es decir que si cada trama tiene 2250 rendijas de tiempos iguales 2250/60=27 milésimas de segundo cada slot de tiempo, sin embargo esto permite el uso del sistema en aquellos lugares donde el tráfico marino es mucho más denso. Únicamente se necesita un procedimiento para que los transpondedores AIS se organicen entre sí de tal manera que no ocupen slots ajenos, para ello se utiliza SOTDMA (Self-Organized Time Division Multiple Access) consiste en una variante del TDMA, ofrece un tratamiento sin perturbaciones.

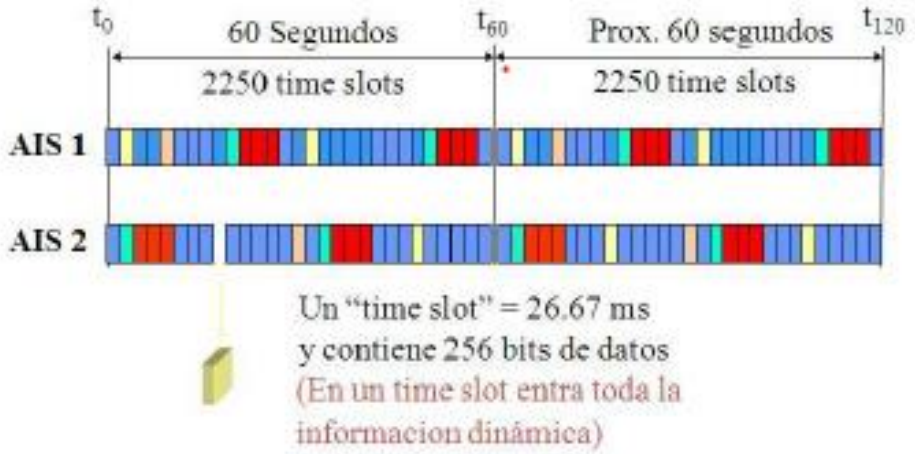

Figura 2. Subdivisión de slots en una trama AIS.

2.2.2 SOTDMA

El funcionamiento de SOTDMA es el esquema de acceso TDMA más complicado, está definido por AIS y pueden encontrarse sus especificaciones técnicas en ITU-R M.1371-4 anexo 2. Básicamente esta técnica de multiplexación se utiliza cuando se transmite en un slot determinado de tiempo y este se encuentra ocupado por lo que primero escucha y después envía, es decir, las embarcaciones conocen al cabo de un minuto la posición, el rumbo, y la velocidad de los barcos que estén dentro del alcance del VHF y a partir de estos datos se genera el directorio dinámico, además como el informe de posición de cada barco aporta también información sobre la reserva de la rendija de tiempo para el siguiente minuto UTC completo.

Gracias a todos estos datos emitidos se obtiene un frame map, el esquema de ocupación de los slots de tiempo en los siguientes minutos UTC, este esquema es memorizado por el transpondedor y actualizado constantemente, existen mensajes que ocupan más de una rendija por lo que la reserva de tiempo se repite tantas veces hasta que el mensaje AIS se ha emitido por completo.

Con todo ello SOTDMA provee de manera dinámica y autónoma una asignación dinámica del canal de transmisión y garantiza que:

Ningún barco bloquee el canal AIS durante mucho tiempo.

Cada barco pueda emitir sus datos a intervalos cortos de tiempo evitando saturar el canal.

Todos los barcos pueden actualizar sus esquemas de actividades.

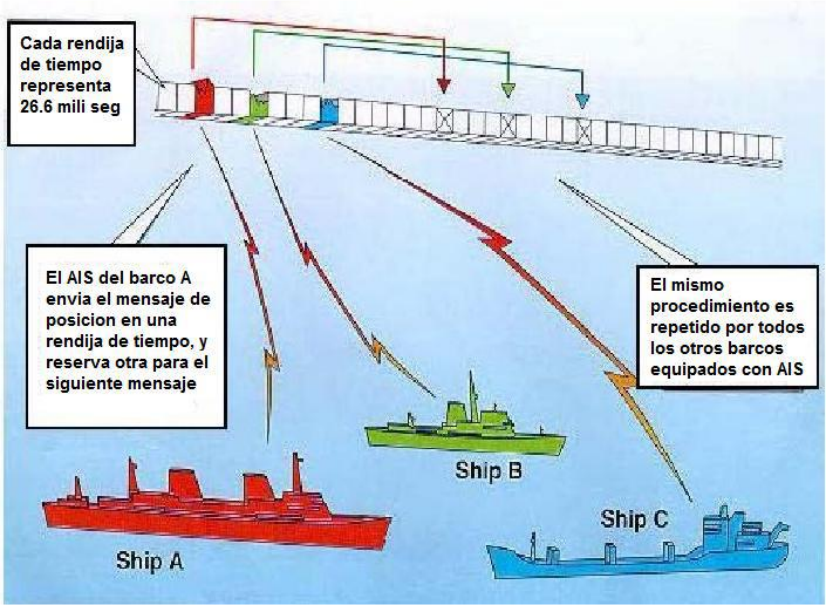

Figura 3. Funcionamiento de SOTDMA entre varias embarcaciones.

En la práctica, la capacidad de este sistema es casi ilimitada, ya que ha sido diseñado para permitir la comunicación entre las zonas con el tráfico marítimo más denso existente en el mundo.

2.2.3 Protocolos AIS

El procedimiento TDMA es usado en telefonía móvil y regula la asignación sin colisiones de un número limitado de canales para un gran número de usuarios. El AIS usa el procedimiento TDMA para asignar rendijas de tiempo en protocolos especiales con los que controla al emisor VHF.

Se emplean principalmente 5 protocolos diferentes dependiendo de la situación del barco y la zona de navegación para seleccionar las rendijas de tiempo:

RATDMA (*Random Time Division Multiple Access*): Este protocolo se encarga de las primeras comunicaciones del aparato AIS puesto que va a hacer su primera emisión, la cual servirá para avisar de su presencia. Para ello escuchará las dos frecuencias AIS durante un minuto, guardara las rendijas ya reservadas y emitirá una reserva aleatoria y cambia al protocolo SOTDMA

SODTMA (Self-*Organized Time Division Multiple Access*)

ITDMA (*Incremental Time Division Multiple Access*): Se emplea para anunciar mensajes AIS relevantes de seguridad.

FATDMA (*Fixed Access Time Division Multiple Access*): se emplea para ayudas a la navegación

CSTDMA (*Carrier Sense Time Division Multiple Access*): Se usa para el AIS de la clase B y trabaja con limitaciones

2.2.4 CSTDMA

El esquema CSTDMA (Carrier Sense Time Division Multiple Access) se emplea como técnica de acceso al medio para transpondedores AIS clase B. Estos transpondedores deben ser compatibles con SOTDMA y garantizar que, en todo momento, SOTDMA tenga prioridad sobre cualquier otro esquema.

Este esquema TDMA no necesita sincronismo UTC directo para operar correctamente, por lo que emplea sincronismo UTC indirecto. De esta manera, el sincronismo entre las estaciones que emplean este esquema deriva de otras estaciones con sincronismo UTC directo (por GPS).

En cuanto a su funcionamiento, las estaciones con CSTDMA monitorizan continuamente el nivel de ruido de fondo en los canales AIS y toman como referencia este nivel de ruido. De esta forma, se logra medir el nivel de la señal al inicio de cada slot.

Usando una asignación aleatoria de slots, se mide el nivel de señal al inicio del slot, seleccionado aleatoriamente, y toma la siguiente decisión:

- Si el nivel de señal es superior al nivel de ruido de referencia, asume que ese slot está en uso y, por tanto, no accede al mismo.

- Si el nivel de señal es inferior al nivel de ruido de referencia, asume que ese slot está libre y, por tanto, accede al mismo.

La principal desventaja de este sistema, frente a la ventaja económica indiscutible que supone la integración de este esquema de acceso sobre transpondedores AIS, es que requiere la existencia de estaciones AIS que empleen otros esquemas de mayor calidad, como SOTDMA, para poder sincronizarse con el resto de estaciones.

2.3 Análisis de las capas del estándar AIS y bandas de frecuencia.

El sistema AIS dispone de dos canales de frecuencia marina VHF (156.00 MHz – 162.025 MHZ):

Canal AIS 1 (161.975 MHz) denominado internamente como AIS A.

Canal AIS 2 (162.025 MHz) denominado internamente como AIS B.

El ancho de banda emitido es de 25 KHz, En VHF es bastante común que se produzcan perturbaciones en el canal, por lo que podría suponer un peligro a la hora de detectar otros barcos y sería un sistema inseguro, por ello utiliza dos canales separados, esto permite que en caso de avería de uno de los receptores el sistema siga funcionando, sin embargo por improbable que pueda ser, en situaciones especiales el aparato AIS de clase A está equipado con un tercer receptor que se mantiene a la escucha en el canal 70 VHF DSC de 156.525 MHZ.

En resumen el AIS deberá ser capaz de funcionar con una anchura de banda de canal de 25 kHz o 12,5 kHz, de conformidad con la Recomendación UIT-R M.1084 y el Apéndice S18. La anchura de banda deberá de ser de 25 kHz y deberá emplearse en alta mar, mientras que la anchura de banda de canal de 25 kHz o 12,5 kHz deberá utilizarse si así lo define la autoridad competente en las agua territoriales, tal como se describe en el § 4.1 y en el Anexo 4.

2.3.1 Alcance del sistema AIS

Cuando hablamos de alcance en el sistema AIS, no es la distancia donde las ondas de VHF llegan, sino la máxima distancia posible donde es posible decodificar y procesar las señales emitidas.

Las características estándar para un radar de un barco medio son 20 millas náuticas de distancia y la potencia es normalmente 2000 vatios, aparte de los factores que influyen en los radares convencionales además de la altura, la reflexión del blanco, la cobertura de tierra. Sin embargo AIS no decodifica una señal debilitada por reflexión sino que recibe la señal directamente del barco.

La banda VHF marino tiene un alcance generalmente de 25 millas, cuando la propagación se produce en línea recta y la potencia de emisión es suficientemente alta para distinguir una señal de voz del ruido.

Estas son las principales características de los aparatos de comunicación que un barco estándar tiene abordo. Por ello el AIS ofrece ciertas ventajas aunque opera en VHF su alcance puede llegar a más de 60 millas náuticas, debido a que no tiene que comprender palabras sino mensajes digitales, además los receptores tienen una sensibilidad de -112 dBm por lo que captan señales muy débiles.

2.3.2 Capa física

El diagrama de comunicaciones del sistema AIS corresponde al del al del cuadro (Figura 4) basado en el modelo de referencia OSI (OPEN System Interconnection) es decir al igual que en muchos otros apartados de las comunicaciones consta de 7 capas o niveles. El proyecto se enfoca principalmente a desarrollar el nivel de presentación y aplicación dado por hecho que un nivel de sesión no tendría sentido para el objetivo que se quiere alcanzar.

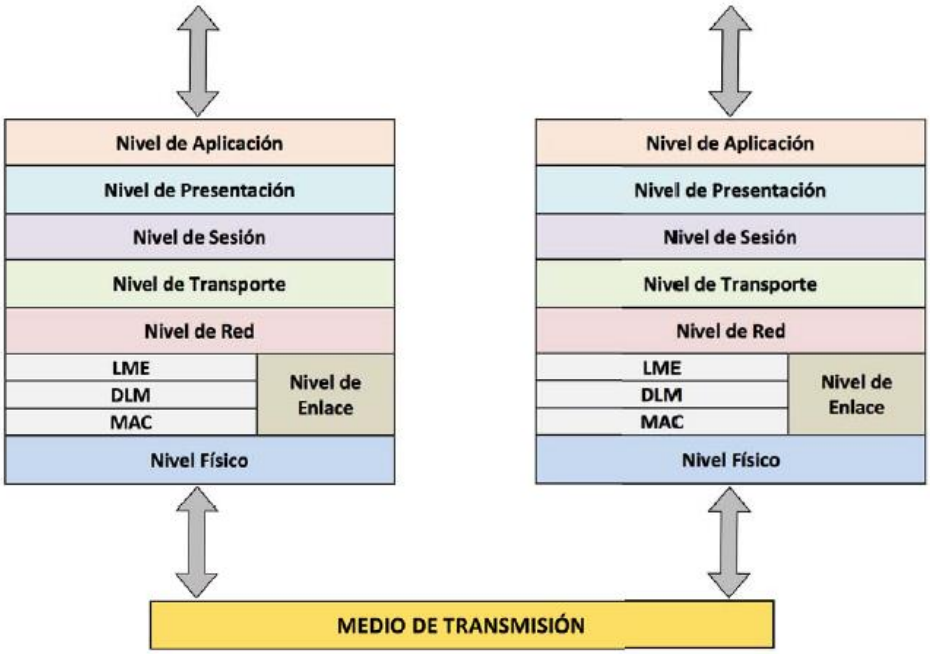

Figura 4. Esquema general de la arquitectura de red del estándar AIS.

En primer lugar la capa física se encarga de transferir cadenas de bits provenientes de un emisor al exterior.

La tasa binaria de transferencia de bits es de $9600 \pm 50 \times 10^{-6}$ bit/s, consta de una potencia de salida de transmisión de 25W como máximo, se utiliza una codificación de datos NRZI y una modulación GMSK/FM de lo cual se hablara en el apartado siguiente.

La subcapa de enlace de datos proporciona los siguientes procedimientos de:

– activación y liberación del enlace de datos;

– transferencia de datos;

– detección y control.

En la capa de red se realiza básicamente el envío y recepción de paquetes de datos.

Por último en lo que respecta a las capas de Transporte, Sesión, Presentación y Aplicación, la única exigencia del estándar es que se implementen de forma coherente a los paquetes de datos que son intercambiados desde el Nivel de Red.

2.3.2 Codificación y modulación

El esquema de codificación utilizado para los sistemas AIS es NZRI (*Non Return to Zero inverted*) el cual mantiene constante el nivel de tensión durante la duración de un bit, en el caso de AIS la forma de onda es especificada aplicando una modificación al nivel cuando aparece un 0 en el tren de bits, esto permite al AIS que la corriente eléctrica necesaria sea de baja señal.

A continuación (Figura 5) podemos observar como es el esquema de modulación y codificación del sistema AIS, una vez los bits estén codificados se modulan utilizando GMSK (Gaussian minimum shift keying) para su posterior modulación FM y su transmisión.

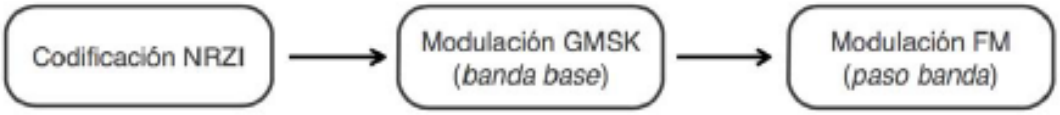

Figura 5. Esquema de codificación y de modulación en el estándar AIS.

El esquema de modulación correspondiente al sistema AIS es la modulación por desplazamiento mínimo gaussiano con modulación de frecuencia, adaptada a la anchura de banda (GMSK/FM), GMSK es un tipo particular de modulación MSK de banda base donde se emplea un filtrado previo para reducir el ancho de banda y con ello eliminar las componentes relativas a los lóbulos secundarios del espectro.

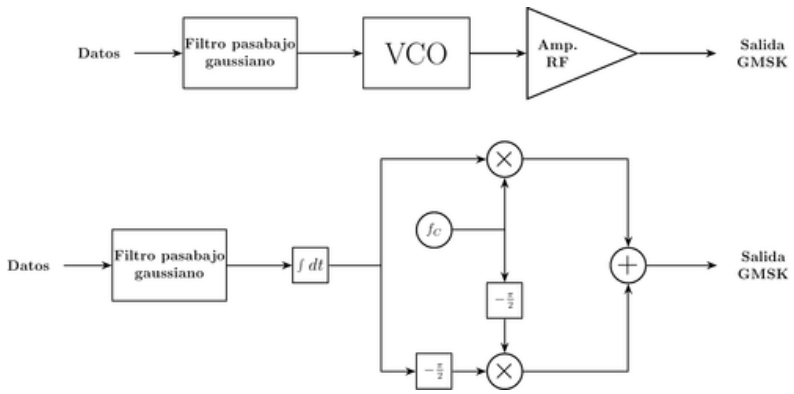

Figura 5. Esquema del modulador GMSK.

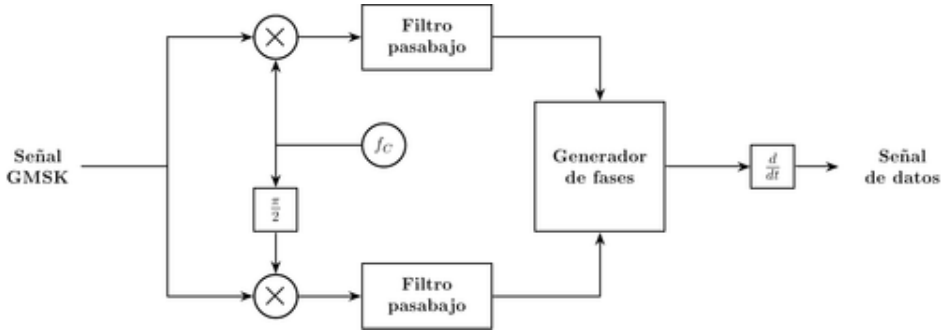

Figura 6. Esquema general de modulación FM para AIS.

El producto BT del modulador utilizado para la transmisión de datos debe ser de 0.4 como máximo si opera en el canal de 25 kHz y de 0.3 en el de 12.5 KHz, en el caso del demodulador usado en la recepción de datos el BT máximo es de 0.5 en el de 25 kHz y 0.3 o 0.5 en el canal de 12.5 kHz.

En la (Figura 7) podemos apreciar la diferencia entre los dos valores, al utilizar un factor BT=0.5 la respuesta al impulso del filtro se extiende 2 periodos de bit en cambio con un factor BT=0.3 llega a los 3 periodos, podemos deducir que la diferencia entre los dos factores radica en el IES puesto que si el factor BT es de 0.3 la interferencia entre símbolos será mayor. Sin embargo la tasa binaria máxima aceptada por el filtro para una misma frecuencia de corte es menor.

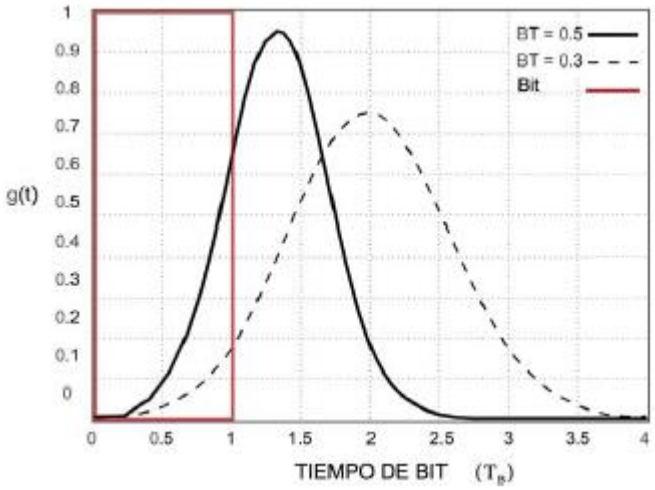

Figura 7. Respuesta al impulso de un filtro gaussiano con BT = 0.3 y BT = 0.5.

Por último el esquema de modulación del estándar AIS, a nivel de sistemas, se completa con la implementación de un modulador o demodulador GMSK en banda base conectado a un modulador o demodulador FM como podemos ver en la (Figura 6), respectivamente. De esta forma, se engloban las virtudes de la modulación GMSK con la robustez frente al canal radio de la modulación FM sobre un mismo sistema.

2.4 Sincronización y organización de las estaciones

En el apartado 2.2 se ha explicado que protocolos se utilizan para la sincronización de tramas y como SOTDMA permite tener una organización de canales donde existe una gran afluencia de datos, sin embargo el proceso de sincronización a nivel de enlace debe estar establecido por la fuente UTC debido a la fiabilidad que aporta. Si esta fuente no está disponible, las siguientes fuentes de sincronización externas, deben servir para la sincronización de la fase del intervalo de tiempo y de la trama:

– una estación con hora UTC, es el primer referente que se debe tener.

– Una estación de base designada como semáforo.

– Otras estaciones sincronizadas con la estación de base.

– Una estación móvil que sea designada como semáforo.

Si más de una estación puede funcionar como semáforo, entonces aquella estación que tenga un mayor número de estaciones recibidas se deberán convertir en estación semáforo de tal manera que sirvan como fuente de sincronización activa. En caso de que haya varias estaciones con el mismo número de estaciones recibidas, aquella que tenga el MMSI más bajo se convierte en la estación semáforo activa.

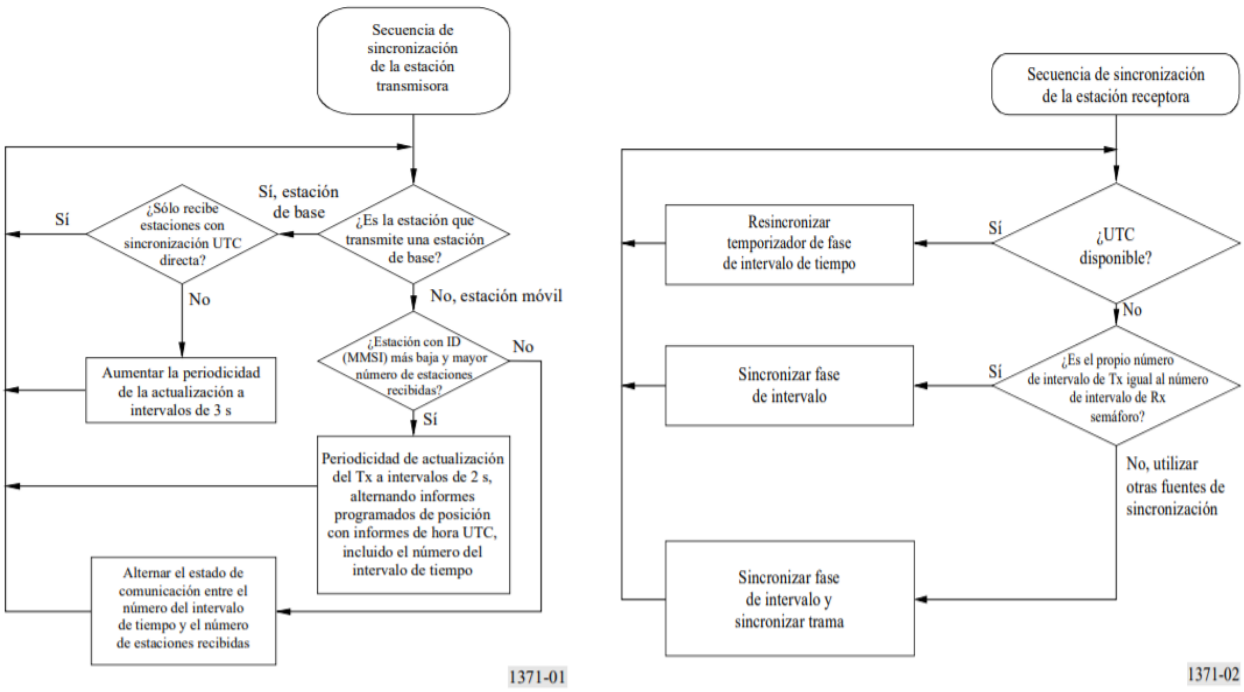

Figura 8. Sincronización de las estaciones transmisoras y receptoras (ITU-R 1371)

Hay que diferenciar dos tipos de sincronización de fase de intervalo de tiempo y de trama

La sincronización de fase de intervalo de tiempo es el método que utiliza una estación base para sincronizarse mediante los mensajes recibidos de otras estaciones, consiguiendo mantener un gran nivel de sincronización y evitando de esta manera que los extremos de los mensajes coincidan en el mismo slot y superponiéndose mutuamente.

La sincronización de tiempo de trama es el método por el que una estación base utiliza el número de intervalo de tiempo actual de ella misma o de otra estación, adoptando el recibido.

Esto es lo que se conoce como sincronización UTC descrita anteriormente en el apartado 2.2.2

La estación base operará la misma periodicidad hasta que detecte una o más estaciones que carezcan de sincronización UTC directa. Cuando ocurra la estación base emitirá con una frecuencia de actualización de 3 s transmitiendo los informes a las estaciones base más frecuentemente.

Las estaciones con acceso directo o indirecto al UTC realizan una sincronización ininterrumpida de sus transmisiones respecto de la fuente del UTC.

La subcapa MAC (Medium Access Control) se encarga de garantizar que la comunicación del Nivel Físico por el medio de transmisión se realice de una forma fiable.Para ello, el estándar AIS incluye las referencias de tiempo común entre las estaciones AIS que integran la red.

Todas las estaciones AIS deben estar sincronizadas bajo una referencia de tiempocomún denominada UTC (Universal Time Coordinated), que viene dada por la red de satélites GPS (Global Position System).

Hay diferentes formas de acceder al sincronismo UTC por partes de las estaciones, diferenciando entre dos formas elementales: sincronismo UTC directo y sincronismoUTC indirecto (Figura 2.10).

Sincronismo UTC directo: el sincronismo UTC deriva de la propia red GPS.El transpondedor AIS debe incluir un receptor GPS que tome esta referencia temporal para sincronizarse a un tiempo común entre estaciones.

Sincronismo UTC indirecta: el sincronismo UTC es heredado de aquellas estaciones que se sincronizan de forma directa con la red GPS.

2.5 Sentencias AIS

Hasta el momento existen 27 tipos de mensajes AIS, aunque conforme se necesitan se añaden mensajes por lo que hay hasta 63 posiciones reservadas para añadir mensajes, están numerados y cada uno tiene asignado un ID que sirve para diferenciarlo ya que cada trama requiere un tratamiento diferente a la hora de codificar y decodificar debido a que la estructura varía en función de la trama.

2.5.1 Mensajes NMEA

Podemos caracterizar estos mensajes en tres tipos según la información que nos aporta.

- **Información estática:** está referida a todos aquellos parámetros propios de una embarcación que permiten identificarla, de manera unívoca, frente al resto de estaciones es decir datos que definen al barco y se mantienen igual:

  - MMSI (*Maritime Mobile Service Identity*).
  - Nombre de la embarcación.
  - Tipo de embarcación.
  - Dimensiones físicas.
  - Bandera del país de origen de la embarcación.

- **Información dinámica**: está referida a la información variable durante una travesía y que es de especial importancia para un correcto control de las operaciones marítimas provistas del sistema AIS:

  - Posición.
  - Velocidad.
  - Trayectoria.
  - Estado de la navegación.
  - Destino.

- **Información sobre el viaje**: el usuario puede introducir manualmente algunos datos relevantes acerca de la travesía marítima y que no son monitorizados de forma automática por el sistema AIS.

- **Mensajes breves relacionados con la seguridad**: un mensaje de seguridad es un mensaje que contiene una advertencia importante sobre las condiciones de navegación o meteorológicas.

En este trabajo se particularizará en las tramas de tipo 1,2 y 3 correspondientes a los datos más importantes para la geolocalización de barcos, la de tipo 4 correspondientes a las estaciones base y por último las de tipo 5 que están referidas a los datos estáticos de los barcos y los datos de la travesía del barco. Se ha particularizado en estas tramas debido a que son las más importantes y aportan la suficiente información para geolocalizar los navíos.

ID mensaje	Denominación	Descripción	Bits de datos
1	Position	Mensaje de posición de embarcaciones	168
2	Assigned position	Mensaje de posición de embarcaciones	168
3	Position	Mensaje de posición de embarcaciones	168
4	Base station report	Posición, hora UTC fecha,rendija de tiempo de una estaación base	168
5	Static and voyage data	Datos estáticos y relacionados con la travesía de embarcaciones(SOLAS) de clase A	424

Tabla 1. Mensajes NMEA tratados.

En cuanto a la frecuencia con que se emiten estos mensajes depende del tipo de información que se quiera transmitir y la velocidad de estos como podremos ver en el (Tabla 2)

- Información estática: Cada 6 minutos en casos normales y cuando se solicite.
- Información dinámica: En función de la velocidad del navio.
- Información relacionada con la travesía y la navegación: Cada 6 min, cuando haya una modificación y cuando se solicite.
- Mensajes de seguridad: Cuando sean necesarios.

Tipo del barco	Intervalo de información
Barco anclado	3 min
Barco en movimiento de 0 a 14 nudos	12 s
Barco en movimiento de 0 a 14 nudos con cambio de derrotero	4 s
Barco en movimiento de 14 a 23 nudos	6 s
Barco en movimiento de 14 a 23 nudos con cambio de derrotero	2 s
Barco en movimiento a más de 23 nudos	3 s
Barco en movimiento a más de 23 nudos con cambio de derrotero	2 s

Tabla 2. Frecuencia de envío de tramas en función de la velocidad del navío.

2.5.2 Formato NMEA 0183

Los mensajes AIS de datos internos del sistema que contienen una gran cantidad de información, parte de la información es para el usuario, pero la gran parte son usados para controlar y configurar el sistema. Sin embargo en este proyecto se ha tratado la parte más dirigida al usuario, debido a que la mayoría de las tramas sirven para reservar rendijas de tiempo, gestión de enlace de datos etc. Las 5 primeras sentencias nos dan los datos de posición de los barcos, las estaciones base y los parámetros más importantes de los barcos.

En cuanto al formato de las tramas utilizan el estándar NMEA 0183 definido en la recomendación ITU-R M.1371, este formato común para la transmisión de datos entre dispositivos electrónicos destinados a equipos marítimos combina ciertas especificaciones eléctricas con los datos recibidos por GPS, permitiendo así la comunicación en tiempo real entre embarcaciones y estaciones marítimas. El estándar utiliza un formato ASCII de 6 bits (Tabla 3) donde solo se emplean 64 caracteres, este formato para las representaciones AIVM/AIVDO de los mensajes radio AIS han sido establecidos por la IEC (International Electrotechnical Commission).

Table 3. Sixbit ASCII

000000	0	"@"	010000	16	"P"	100000	32	" "	110000	48	"0"
000001	1	"A"	010001	17	"Q"	100001	33	"!"	110001	49	"1"
000010	2	"B"	010010	18	"R"	100010	34	"""	110010	50	"2"
000011	3	"C"	010011	19	"S"	100011	35	"\#"	110011	51	"3"
000100	4	"D"	010100	20	"T"	100100	36	"$"	110100	52	"4"
000101	5	"E"	010101	21	"U"	100101	37	"%"	110101	53	"5"
000110	6	"F"	010110	22	"V"	100110	38	"&"	110110	54	"6"
000111	7	"G"	010111	23	"W"	100111	39	"\'"	110111	55	"7"
001000	8	"H"	011000	24	"X"	101000	40	"("	111000	56	"8"
001001	9	"I"	011001	25	"Y"	101001	41	")"	111001	56	"9"
001010	10	"J"	011010	26	"Z"	101010	42	"*"	111010	58	":"
001011	11	"K"	011011	27	"["	101011	43	"\+"	111011	59	";"
001100	12	"L"	011100	28	"\"	101100	44	","	111100	60	"<"
001101	13	"M"	011101	29	"]"	101101	45	"-"	111101	61	"="
001110	14	"N"	011110	30	"\^"	101110	46	"."	111110	62	">"
001111	15	"O"	011111	31	"_"	101111	47	"/"	111111	63	"?"

Tabla 3. Tabla de decodificación ASCII de 6 bits.

Los mensajes se presentan como una ráfaga de datos AIVDM que son transmitidos hasta un host como datos binarios ASCII y se codifican a través de los protocolos de red TCP/IP o UDP/IP.

Un ejemplo de mensaje NMEA 0183 sería:

!AIVDM,1,1,,B,34ReJ651h0OvQd2FT4:P0CUn0000,0*0F

Este es el tipo de trama que se recibe en los receptores AIS y los que el programa decodifica en este caso es una trama de tipo tres la cual nos aporta información importante como la posición, velocidad o el identificador del barco, sin embargo la longitud de las tramas NMEA varía según el tipo, por ejemplo otro tipo de trama que se trata son las de tipo cinco las cuales no solo tienen una longitud distinta sino que se tratan de dos tramas sumadas, es decir una trama compuesta:

!AIVDM,2,1,4,A,548RFv02>`p8iU=P001@td5@4000000000000016@H@<<6S70AmPC1CP,0*2F

! AIVDM,2,2,4,A,j@@000000000000,2*7ª

Sin embargo en todas las tramas se puede asociar cada parte de la trama separada con comas con un tipo de información ofrecida la cual es la misma en casi todas, en la (Tabla 4) se describe el significado de las diferentes partes de un mensaje con formato NMEA 0183 como el que se

presenta en los ejemplos anteriores. Los campos del mensaje y su descripción en esta tabla se ordenan conforme a una lectura del mismo de izquierda a derecha.

Descripción	Mensajes NMEA
Tipo de mensaje NMEA	!AIVDM
Número de líneas del mensaje	1
El número del fragmento en este caso el primero	1
Es un mensaje secuencial para para las tramas compuestas	Espacio vacio
El canal de radio utilizado	B
La carga útil donde se encuentra la información	34ReJ651h0OvQd2FT4:P0CUn0000
Fin de los datos AIS	0*
CheckSum NMEA	0F

Tabla 4. Descripción de los campos de las tramas NMEA.

2.5.3 Decodificación del payload

En cuanto a la decodificación del payload cada trama tiene su propia decodificación y campos de datos debido a que cada una tiene un tratamiento diferente dependiendo de los datos a transmitir, en el apartado 2.3.2 se trata como se ha realizado la decodificación de la carga de datos a nivel de programación sin embargo antes debemos entender la parte matemática y teórica que hay implícita.

Como se ha mencionado antes, en este proyecto se ha abordado únicamente hasta la trama cinco puesto que el objetivo era la geolocalización de barcos mediante un terminal de bajo coste, por ello solo son necesarias las 5 primeras tramas las cuales serán descritas para posteriormente explicar el programa diseñado para su tratamiento

Tramas tipo 1,2 y 3:

Para este tipo de tramas la decodificación es idénticamente igual por ello utilizaremos una de tipo 1 y nos quedaremos con la parte correspondiente al payload, utilizaremos de ejemplo esta trama:

!AIVDM,1,1,,A,15O86n001TJ3KutH8ar@<h;l06Hh,0*5D

En primer lugar separaremos los 48 caracteres ASCII del payload, una vez separados cogeremos carácter a carácter y se deberá traducir a binario donde cada carácter es representado por 6 bits como se muestra a continuación (Tabla 6):

Payload : 15O86n001TJ3KutH8ar@<h;l06Hh

1	000001
5	000101
O	011111
8	001000
6	000110
n	110110
0	000000
0	000000
1	000001
T	100100
J	011010
3	000011
K	011011
u	111101

t	111100
8	001000
a	101001
r	111010
@	010000
<	001100
h	110000
;	001011
l	110100
0	000000
6	000110
H	011000
h	110000

Tabla 5. Decodificación de cada carácter a binario 6 bits.

Una vez tengamos la decodificación binaria debemos concatenar todos los bits para crear una cadena larga de un solo mensaje, una vez tengamos la cadena entera de datos debemos pasar a decimal cada tupla de bits divididos según la longitud que ocupe el dato es decir:

Field	Len	Description	Member	T	Units
0-5	6	Message Type	type	u	Constant: 1-3
6-7	2	Repeat Indicator	repeat	u	Message repeat count
8-37	30	MMSI	mmsi	u	9 decimal digits
38-41	4	Navigation Status	status	e	See "Navigation Status"
42-49	8	Rate of Turn (ROT)	turn	I3	See below
50-59	10	Speed Over Ground (SOG)	speed	U1	See below
60-60	1	Position Accuracy	accuracy	b	See below
61-88	28	Longitude	lon	I4	Minutes/10000 (see below)
89-115	27	Latitude	lat	I4	Minutes/10000 (see below)
116-127	12	Course Over Ground (COG)	course	U1	Relative to true north, to 0.1 degree precision
128-136	9	True Heading (HDG)	heading	u	0 to 359 degrees, 511 = not available.

Tabla 6.Campos correspondientes al payload descritos en la tabla de decodificación.(Anexo I Tabla 6)

Según la tabla de decodificación de cada trama (Tabla 6), cada dato viene representado por una longitud fija de bits por lo que en el caso de la trama que estamos tratando sería:

0000|010001010111110001000|0001|10110110|0000000000|0|000101011111001000000011011011000

Para extraer el mmsi que corresponde al identificativo de los barcos extraer de la cadena concatenada del bit 8 al 37 y convertir a decimal:

01010111110010000011011011000 → 368183000

En el caso del estado de navegación, la velocidad de giro, la velocidad o la precisión, se procedería de la misma forma:

Estado de navegación 0001 → 1 donde en el anexo I (Tabla 7) podemos ver la tabla para saber a que estado de navegación corresponde.

Velocidad de giro 00000000 → en cuanto a las velocidades de giro están especificadas en el anexo I donde en este caso significaría que no estaría girando.

Velocidad 0001100100 → 100 donde la velocidad es dada en la décima parte de un nudo por lo que correspondería a 10 nudos.

En genera exceptuando los parámetros de longitud y latitud, el resto se decodifican de la misma forma y cada uno debe ser interpretado de acuerdo a las tablas de decodificación establecidas.

Por ello veamos los parámetros de longitud y latitud

Para la longitud deberemos convertir a decimal la tupla de bits del 61-88

11010000011011011111110111110 → 84336574

Sin embargo hay que tener en cuenta que la longitud y la latitud pueden ser negativas por lo que habrá que tener en cuenta el signo, lo cual se pudo deducir después de comprobar con diferentes decodificadores online que la longitud al tratarla sin signo se salía del mapa, por ello en este caso como el primer número es 1 deberemos realizarle el complemento a 2 quedaría:

01011111001001000000100000 1 al invertir los bits y se le debe sumar uno

Con lo que obtendríamos la tupla binaria que nos interesa

01011111001001000000100000 10 → 49881154

Después de comprobar teóricamente que esta decodificación era correcta comprobándolo en los mapas se podían decodificar todos los campos.

La longitud y la latitud están representadas en unidades de 0.0001 minutos por lo que la longitud transmitida será:

4988.1154 minutos/60= 83.135256666666667 grados

La longitud puede tener sentido este o en este caso oeste debido al signo negativo por lo que sería: -83.135 grados o 83.135 grados oeste.

Para la latitud deberemos convertir a decimal la tupla de bits del 89-115 y se procedería de la misma manera que con la longitud:

0100000011011011000 →

Teniendo en cuenta que en este caso no tiene signo se decodificaría directamente a decimal y dividiendo entre 10000 y 60 dando 42.179375 grados positivos es decir 42.179375º norte.

Tramas tipo 4:

Las tramas de tipo cuatro son muy similares a las de tipo 1,2 y 3 con alguna pequeña variación de los datos transmitidos debido a que son transmitidas por estaciones base, la tabla de decodificación la podremos encontrar en el Anexo I.

En este caso la trama de tipo 4 tiene un nuevo parámetro la fecha, que a diferencia de las tramas emitidas por los barcos están tienen una fecha donde se indica el da y la hora en la que ha sido transmitido.

La decodificación en este caso sería igual que para los parámetros que no dependen del signo utilizando el diccionario de ASCII de 6 bits y una vez obtenida la cadena concatenada obtendríamos la fecha exacta decimal y se representaría en el formato indicado en (Tabla 7). En cuanto a los demás parámetros se seguiría la misma dinámica y se debe consultar los valores obtenidos en la tabla de decodificación.

38-51	14	Year (UTC)	year	u	UTC, 1-9999, 0 = N/A (default)
52-55	4	Month (UTC)	month	u	1-12; 0 = N/A (default)
56-60	5	Day (UTC)	day	u	1-31; 0 = N/A (default)
61-65	5	Hour (UTC)	hour	u	0-23; 24 = N/A (default)
66-71	6	Minute (UTC)	minute	u	0-59; 60 = N/A (default)
72-77	6	Second (UTC)	second	u	0-59; 60 = N/A (default)

Tabla 7. Campos de fecha descritos en la tabla de decodificación de la trama tipo 4. (Anexo I)

Tramas tipo 5:

Debido a que la trama 5 es una trama compuesta debemos tener en cuenta que la posición de los parámetros cambiara, pasemos a explicar como se decodifica este tipo de tramas. Tomaremos como ejemplo está trama:

!AIVDM,2,1,1,A,58UQ<802@Vj9TaIWV20pE@PE8h4pB1@T@F22220l2Hk8865E0<Tm<p 8888888888,0*46

!AIVDM,2,2,1,A,8888880,2*25

Como se ha explicado anteriormente este tipo de trama NMEA está compuesta de dos tramas es debido a que en una sola sería demasiado larga, en este tipo de tramas no se aporta la posición, sino parámetros de tiempo, dimensiones del barco, nombre del barco,tipo de barco y destino.

Siguiendo la tabla de decodificación obtenemos los primeros parámetros comunes en el resto de tramas el MMSI y el indicador de repetición

La versión del AIS usado, el número de IMO y el símbolo de llamada se pueden interpretar a partir de la tabla de decodificación situada en el (Anexo I). En concreto en la versión AIS únicamente existen 3 versiones actualmente. El parámetro de símbolo de llamada y el nombre del barco tienen una decodificación distinta, debido a que hay que traducir las tuplas binarias a caracteres ASCII de 6 bits, por lo que en el caso de este ejemplo:

40-69	30	IMO Number	imo	u	IMO ship ID number
70-111	42	Call Sign	callsign	t	7 six-bit characters
112-231	120	Vessel Name	shipname	t	20 six-bit characters
232-239	8	Ship Type	shiptype	e	See "Codes for Ship Type"

Tabla 8. Campos correspondientes a la trama tipo 5.

La trama pasada a binario sería:

```
000101001000100101100001001100001000000000000010010000100110110010001001100100101001011001100111100110000010000000111000010101010000100000010
010010001100000000100111000010010000001010000100100010000010110000010000010000001000000001101000000100110001100110010000010000000110000101010
010100000000011001001001101010011001110000010000010000001000000100000010000001000000100000010000001000000100000010000001000000100000010000000000

Vamos a formatear_trama5
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
Estamos en formatear_trama5
011001001010010110011001111001100000100000
001110000101010100001000000010101001000110000000100111000010010000001010000100100010000001011000001000001000000100000100000
['001110000101010100001000000010101001000110000000100111000010010000001010000100100010000001011000001000001000000100000100000']
este campo mide
42
YJVY9
este campo mide
120
NETHERLAND TIDE
52
19
51
8
8
1
8
10
21
0
5.0
este campo mide
120
ST3
0
0
576212000
Barco 576212000 no registrado
```

Figura 9. Extracto de la decodificación hecha por el programa.

Podemos observar que es demasiado larga para poder analizarla tupla a tupla,sin embargo se puede apreciar que a diferencia de las otras tramas en esta es posible que no hayan sido transmitidos correctamente todos los parámetros descritos.

A continuación observamos que el parámetro correspondiente al tipo de barco ocupa una tupla de 8 bits por lo que podrá derivar hasta 255 tipos de barcos, actualmente están definidos 100 tipos diferentes de navíos, por lo que cada número es asignado a un barco en este caso sería uno de tipo 52 lo que correspondería a un TUG (remolcador) como podemos comprobar en la tabla. (Anexo I Tabla 11, Figura 10).

52	Tug

Figura 10. Tipo de barco extraído de la tabla de tipo de barcos (Anexo I, Tabla 11)

Los demás campos son numéricos por lo que el formato sería el mismo descrito anteriormente, simplemente al pasar a decimal los campos de las dimensiones del barco, la fecha y el calado hay que poner las unidades especificadas en la tabla de decodificación.

2.6 AIS tipo A y B

Actualmente existen dos clases diferentes para el estándar AIS, en su origen cuando en 2001 se definió AIS únicamente existía una clase la A, pero años más tarde en 2005 se definió el nuevo estándar, esto es debido a que el sistema AIS es capaz de dirigir más de 2000 paquetes por minuto, si consideramos que los barcos más veloces emiten cada 2 segundos, en zonas con

abundante tráfico se producirían cuellos de botella. Además las empresas de equipamiento de embarcaciones de recreo querían vender equipos AIS, por ello estos aparatos debían de ser más baratos que los sistemas AIS de un petrolero. Estos factores propiciaron el desarrollo de la nueva clase B.

Sin embargo, está nueva clase carece de algunos de los privilegios que tiene la clase A debido a que está dirigido a embarcaciones SOLAS, por lo que siempre tienen acceso al sistema, ya que debido a su tamaño y a la peligrosidad del cargamento abordo, siempre tienen acceso al sistema, a diferencia de las embarcaciones de recreo de la clase B que deben esperar para emitir hasta que haya rendijas libres. Es decir la clase B funciona de forma similar que la clase A pero está limitado de modo que los buques SOLAS de la clase A siempre tienen preferencia.

2.6.1 Diferencias principales

- AIS Clase A: sometido a las instrucciones marcadas por el SOLAS. Es el equipo que deben instalar los barcos a los que se les exige en atención a la norma internacional. Debido a sus características está sometido a homologación por parte de la Administración (en España la DGMM).
- AIS Clase B: aprovecha la tecnología de identificación de barcos, pero con menores prestaciones y menores requisitos tecnológicos. Es el equipo que la UIT (Unión internacional de telecomunicaciones) recomienda instalar en los barcos en los que nos es obligatorio la "clase A". Este equipo es el que establece la DGMM como obligatorio en las embarcaciones utilizadas para impartir formación náutica.

	Clase A	Clase B
Especificación del sistema	IMO AIS	IMO AIS limitado
Protocolo	SOTDMA	CSTDMA
Potencia de emisión máxima	12,5 W	4 W
Tasa de repetición	cada 10 s	cada 30 s
Emisión mensajes de seguridad	si	no necesario

Tabla 9.Diferencias entre las especificaciones técnicas de los AIS clase A y B

En el cuadro se puede apreciar las principales diferencias entre los sistemas AIS de clase A y B, sin embargo en el caso del proyecto desarrollado no afectará a su elaboración debido a que se aplica a emisores, por lo que nuestro receptor no tendrá que tener en cuenta las características particulares de cada uno, sin embargo es importante definir las diferencias.

Capítulo 3 Diseño hardware

En este capítulo se abordara el proceso realizado, para obtener todos los componentes necesarios para la captura de tramas especificando brevemente las especificaciones técnicas necesarias para el caso de una raspberry, ya que se han explicado de forma general durante el capítulo 2.

3.1 Raspberry Pi 3 como receptor AIS

Raspberry Pi es un computador de placa reducida destinada a la realización de prototipos electrónicos para una gran variedad de propósitos, de bajo coste, el software es open source, siendo su sistema operativo oficial una versión adaptada de Debian, denominada Raspbian, aunque permite usar otros sistemas operativos.

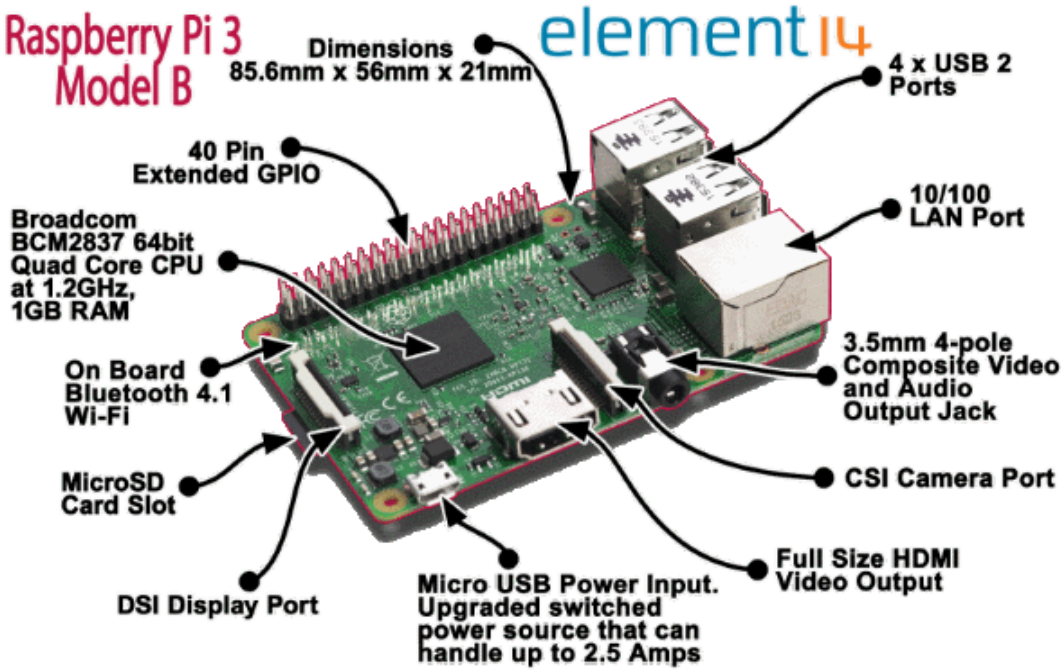

Figura 10. Conexiones disponibles en la Raspberry Pi 3

Una de las razones por las que se ha escogido una Raspberry Pi como procesador de las tramas recibidas es por su versatilidad y libertad a la hora de programar y que no haya ningún problema con el terminal, su asequible precio reduciendo el coste de lo que normalmente un terminal AIS costaría, donde el terminal más barato cuesta 700 euros y por último el rendimiento que puede dar a largo plazo.

Por tanto, debido a su fácil manejo y la posibilidad de instalar las librerías específicas para el proyecto podemos optimizar el uso de la raspberry únicamente para el uso como receptor. Es decir tiene las ventajas que podemos encontrar en un ordenador a la hora de operar con ella, donde no se requiere una gran capacidad de procesamiento y las ventajas de un terminal al ser de dimensiones reducidas.

3.1.1 Especificaciones técnicas

Podemos destacar a la Raspberry Pi por su nuevo procesador, un ARM Cortex A53, un procesador de cuatro núcleos a 1.2GHz de 64 bits y que, según sus datos, tiene un rendimiento 10 veces superior al de la Raspberry Pi original y un 50% más que la Raspberry Pi 2, el modelo anterior.

La Raspberry Pi 3 consta de conectividad inalámbrica WiFi y Bluetooth algo con lo que sus predecesoras no contaban. Además de un puerto Ethernet y entrada hdmi.

Se ha provisto para este proyecto de una capacidad de 32 Gbs en la tarjeta SD y una RAM de 1 GB por lo que habrá que tenerlo en cuenta al gestionar las diferentes tareas a realizar por el programa.

Cabe destacar las dimensiones: 85 x 56 x 17 mm puesto que está característica nos permite una gran movilidad, además está provista de 4 puertos USB 2.0 donde estará conectado el receptor AIS.

3.1.2 Capacidad y rendimiento

Los 4 núcleos que posee la Raspberry pi serán cruciales a la hora de realizar el programa debido a que permitirá subdividir las diferentes tareas de captación, procesamiento y almacenamiento en la recepción de las tramas AIS mediante el uso de hilos o threads, lo cual ayudara a la fluidez de ejecución y a una visualización más limpia.

Además cabe recordar la importancia de la capacidad de almacenamiento debido a que estará expuesta a tiempos prolongados de captura y posiblemente sin acceso a internet o a un servidor donde poder descargar los datos. Por lo que con una tarjeta de 32 Gbs como la que es usada es posible que al cabo de un tiempo prolongado sin un tratamiento correcto quede sin espacio para guardar más tramas.

3.2 Comparativa con terminales AIS

La gran diferencia entre el terminal de “low cost” y el receptor AIS podría ser la diferencia de precio sin embargo una placa base de estas características permite realizar algo más que la mera representación de datos en un mapa, debido a que a partir de la base de datos, la información puede ser útil para análisis, geolocalización, ayuda a la navegación o incluso para espionaje. En la imagen podemos observar un receptor estándar para la captura de tramas pantalla (Figura 11).

Figura 11. Dispositivo de recepción AIS, modelo Comar AIS-NMEA

Sin embargo a nivel físico podemos recalcar 5 puntos fundamentales donde difieren y son similares:

- Otra de las diferencias que presentan a nivel de hardware sería el equipo necesario puesto que en un transpondedor AIS o en un receptor, el monitor y los dispositivos externos para el control del programa viene ya integrado.

- Existe una diferencia de consumo bastante mínima entre los dos tipos de receptores, puesto que la Raspberry PI 3 a pleno rendimiento puede llegar a consumir 1.8W y existen terminales AIS que consumen de 0.5 W el que menos a 6 W los más potentes de la gama media o 40 W sí también funciona como transmisor. Por ello la Raspberry ofrece un valor estándar adecuado a casi todas las situaciones que se puedan dar en alta mar o para la recepción en tierra.

- En cuanto a nivel de recepción el "Daisy AIS Receiver" (receptor usado para la demodulación de tramas) nos otorga un nivel de sensibilidad de -110 dbm lo cual es muy similar a la dada por un receptor AIS la cual es de -107 dbm.

- La tasa de transmisión (BaudRate) es exactamente el mismo para todos los terminales 38400 baudios es decir el número de cambios de señal o símbolos que ocurren por segundo es el mismo, aunque el receptor Daisy nos permite tener una tasa de 9600 y 4800.

- La conexión entre la antena y terminal se realiza mediante USB por lo que por una parte presenta la ventaja de su fácil conectividad y utiliza un estándar muy popular, sin embargo el uso de este tipo de conexión puede sumar ruido al canal y existen perdidas de conexión momentáneas pero no supone un problema debido a que el timeout es lo suficientemente largo como para que no suponga un problema y no se note diferencia entre la recepción de tramas.

3.3 Equipo de recepción

En este apartado se abordara la explicación de los dispositivos utilizados en y montaje del sistema de recepción "lowcost" para la captura, decodificación y almacenaje de tramas AIS, con su posterior representación en mapa.

- Antena:

El primer accesorio necesario para la recepción es la antena, en principio cualquier antena marina VHF que pueda cubrir las frecuencias de 156MHz – 163 MHz puede emplearse con un transpondedor o receptor AIS, por ejemplo una antena de televisión que cubre las bandas VHF y UHF sirve perfectamente si se gira 90 grados para que los dipolos estén posicionados de manera vertical.

La antena usada para este proyecto ajustándose a las especificaciones es la Pacific Aerials Seamaster Classic 3 '6db VHF UltraGlass Antenna. Dnde se ha

En la tabla 10 se presentan los parámetros que caracterizan a la Pacific Aerials Seamaster Classic seleccionada para el prototipo hardware:

Parámetros	Valor
Banda de frecuencia(MHz)	136-175
Ancho de banda(MHz)	5
Ganancia(dbi)	3
Impedancia(Ω)	50
Polarización	Lineal
Tipo de conector	SMA
Longitud (cm)	150

Tabla 10. Especificaciones técnicas de la antena usada.

Cabe destacar que en caso de que la instalación del receptor se haga en un barco deberá existir una separación suficiente entre la antena VHF del equipo AIS y la antena VHF utilizada para el equipo de radio para evitar posibles interferencias, por ello lo recomendable es instalar la antena de la radio debe montarse en lo alto del mástil en la verga

- Cable:

En cuanto al cable utilizado se pueden utilizar PL 259 macho o un TNC pero en nuestro caso utilizaremos un BNC macho para conectar al receptor AIS, es importante destacar que el cable no debe superar los 5 metros de longitud debido a las pérdidas que puede ocasionar y sumadas al ruido introducido por el canal radio puede hacer que las tramas no lleguen completamente.

Para evitar daños al Daisy receiver la antena debe estar instalada como mínimo a 3 metros de la antena, además es importante apartarlo de equipos de alta potencia debido a su sensibilidad.

• DAISy Ais Receiver

Podemos definir el dAISy como un receptor fiable de un único canal para captura de tramas AIS. Es el receptor elegido para este proyecto debido a su sencillez, bajo coste, eficiencia y tamaño.

Con una potencia de consumo muy baja, menos de 100 mW en modo de recepción, no se requiere ningún driver para su instalación simplemente hay que conectarlo en este caso a la Raspberry, podemos ver en la figura 12 las principales características del receptor.

Baud rate	38400
Data bits	8
Parity	None
Stop bits	1
Flow control	None

Figura 12. Especificaciones técnicas del dAISy AIS receiver.

Por defecto está configurado para 38400 baudios(Alta velocidad NMEA) pero es totalmente configurable a otros baudrate.

Es necesario habilitar los puertos del dispositivo al que se conecte, en este caso para la Raspberry no hizo falta desde consola habilitar ningún puerto, sin embargo en el programa se debe considerar la posibilidad de que la raspberry se le puedan haber cerrado los puertos por lo que se debería notificar al usuario.

El dAISy AIS receiver posee un sistema de leds que nos permite saber el estado en el que está sin tener que acceder al menú por consola, los estados son 5 que se resumen brevemente en el estado de recepción normal, donde se indica mediante una pequeña luz verde cada cinco segundos, la captación de un paquete AIS con un flash verde prolongado y la luz rojo que nos indicará si existe algún problema con el sistema o con el paquete recibido.

3.3.1 Montaje

Una vez disponemos de todo el equipo procedemos al montaje el cual no tiene ninguna dificultad y se explicara brevemente en este apartado.

La antena debe estar conectado al receptor dAISy mediante el cable BNC, a continuación el receptor AIS ira conectado a la Raspberry mediante un cable USB a micro USB. Obviamente la alimentación será distribuida por la Raspberry por lo que tendrá que conectarse a una fuente de alimentación, disponemos de varios puertos USB donde le deberemos conectar el USB y en caso de que haya algún dispositivo externo para el manejo del terminal y una entrada hdmi para poder observar la captación y representación de barcos.

En la imagen podemos observar como están conectados los diferentes componentes como están ubicados y que función realiza cada uno de ellos:

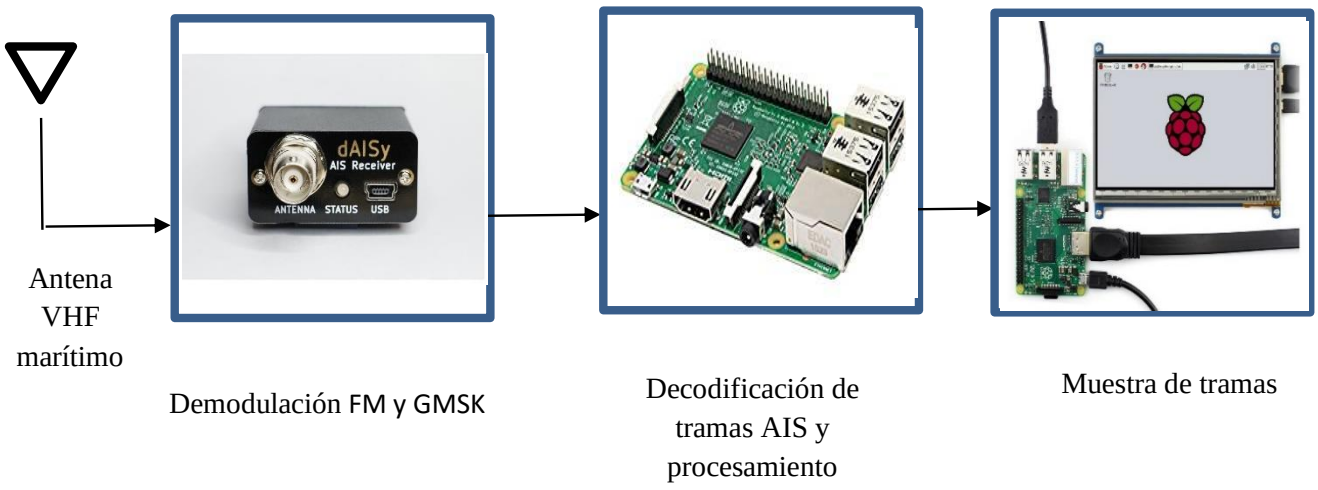

Figura 13. Diagrama de montaje del terminal AIS.

Como se ha explicado anteriormente se deben tener en cuenta el emplazamiento donde se va a instalar debido a que no es lo mismo situarlo en un lugar costero en tierra que en un barco donde

existen varios equipos de radio que pueden verse afectado mutuamente creando interferencias entre ellos. Por tanto esto afectará a la situación de los dispositivos

3.3.2 Ruido

EL ruido es tratado en esta sección principalmente debido a que es tratado de manera explícita en el apartado de modulación, sin embargo, explicaremos en este apartado como el estándar AIS trata el ruido en el canal radio y como nosotros al realizar el montaje del terminal podemos evitar ruido indeseable.

En primer lugar no es el mismo ruido para los AIS de tipo A que para los de B principalmente debido a la potencia de los barcos que utilizan un tipo u otro por ello nos centraremos en los de tipo B.

Las estaciones con CSTDMA (Apartado 2.2.3) monitorizan continuamente el nivel de ruido de fondo en los canales AIS y toman como referencia este nivel de ruido. De esta forma, se logra medir el nivel de la señal al inicio de cada slot.

Usando una asignación aleatoria de slots, se mide el nivel de señal al inicio del slot, seleccionado aleatoriamente.

- Si el nivel de señal es superior al nivel de ruido de referencia, asume que ese slot está en uso y, por tanto, no accede al mismo.

- Si el nivel de señal es inferior al nivel de ruido de referencia, asume que ese slot está libre y, por tanto, accede al mismo.

La principal desventaja de este sistema, frente a la ventaja económica indiscutible que supone la integración de este esquema de acceso sobre transpondedores AIS, es que requiere la existencia de estaciones AIS que empleen otros esquemas de mayor calidad, como SOTDMA (AIS de tipo A), para poder sincronizarse con el resto de estaciones.

Además, una mala medida del nivel de referencia de ruido en el canal puede ocasionar que las estaciones que integran CSTDMA interfieran sobre los slots en uso y provocar la colisión de paquetes. Es por ello por lo que normalmente se emplea en receptores ya que el ruido del canal radio es inevitable, por lo que en los receptores suele implementarse un filtro pasivo paso bajo eliminando el ruido de alta frecuencia.

Sin embargo en la implementación de nuestro terminal debemos procesar el ruido a nivel de software es decir es posible que los bits de las tramas sean erróneos sin embargo se debe ajustar a unas características muy específicas por ello será fácil detectar las tramas erróneas.

Por último debido al sistema utilizado, el receptor es capaz de discriminar las tramas que no llegan completas o donde existen algún bit sobrante y las elimina por lo que siempre llegan tramas ajustadas a la longitud reglamentaria.

Capítulo 4 Diseño Software.

En este capítulo se abordará el proceso de creación de un software adecuado para la interpretación de las tramas AIS, en particular las cinco más importantes para el objetivo de este proyecto como se ha mencionado anteriormente, para ello se ha dividido este capítulo en seis apartados diferentes abordando el proceso seguido para la elaboración del programa, además de explicando los diferentes problemas encontrados, se explicarán las herramientas de las que se han hecho uso para poder conseguir el objetivo.

4.1 Configuración de la Raspberry e introducción a Python 3

En primer lugar la Raspberry viene sin ningún sistema operativo debido a que la tarjeta de memoria debe ser introducida por el usuario, por ello debemos instalarle nosotros el sistema operativo.

Hay dos maneras de instalar Raspbian, una es usando directamente la imagen de Raspbian y otra es usando NOOBS. Esta última opción es la más sencilla y permite instalar también otras distribuciones Linux para Raspberry Pi, pero también ocupará un espacio adicional en la tarjeta SD.

Para el proyecto se utilizó la manera más sencilla mediante NOOBS (New Out Of Box Software) simplemente se formateo la tarjeta SD con formato FAT32 y se copiaron los archivos que nos descarguemos de la página oficial de Raspberry Pi. Después se introdució la SD en la Raspberry, se conecta todo e iremos seleccionando las opciones que nos ofrezca el asistente de instalación. Con el primer inicio de nuestra Raspberry Pi con Raspbian, se muestra rapi-config, una herramienta de configuración de Raspbian para configurar las opciones principales. Una vez se ha instalado el sistema operativo se procede a instalar los programas necesarios para programar,

El lenguaje que se ha empleado principalmente para la realización del proyecto es Python 3, Python 3 es un lenguaje de programación dinámico, multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma. Por ello a la hora de programar tiene una gran libertad y no requiere de muchas líneas de código hacer una función de complejidad media.

Además hay que tener en cuenta que Python es un tipo de lenguaje que requiere de un indentado muy específico, es decir al no utilizar corchetes ni ningún delimitador para las sentencias de código. El sangrado comienza un bloque y su ausencia lo termina. Esto quiere decir que el espacio en blanco es significativo y debe ser consistente.

En este ejemplo (Figura 14) del código del programa la función decodificar está sangrado a cinco espacios. No tienen por qué ser cinco, el único requisito es que sea consistente. La primera línea que no esté sangrada queda ya fuera de la función.

```
1747
1748 def decodificar(data):
1749     longdatar = len(data)
1750     contador=0
1751     for i in range(longdatar):
1752         caracter = data[i:i+1]  # En esta variable vamos guardando caracter a caracter la carga util para tratarla
1753         if str(caracter) ==",":
1754             if contador==4:
1755                 caractertipo=data[i+1:i+2]
1756                 #print(str(caractertipo))
1757                 tipo(caractertipo,data)
1758                 #print("hemos vuelto de guardar y decodificar")
1759                 return
1760             else:
1761                 contador=contador+1
1762
1763 def tipo(caractertipo,data):
1764         #print(data)
1765         if str(caractertipo)<"5" and str(caractertipo)>"0":
```

Figura 14. Código extraído del módulo de decodificación. (Anexo II)

Lo mismo ocurre con los for o los if en este caso el primer bucle del for abarca toda la función desde que empieza el bucle en la línea 1751 hasta que empieza la siguiente función en la 1763, por lo que esto puede suponer una facilidad a la hora de programar puesto que no hay que estar tan pendiente de cerrar las construcciones, sin embargo si de que no haya ningun espacio mal puesto.

Python es un lenguaje de programación a diferencia de otros como Java, interpretado por lo que para poder ejecutar un Programa con la extensión de Python .py es necesario una consola y utilizar el módulo de Python instalado en la raspberry (Figura 29)

Sin embargo es posible después crear un ejecutable para que su acceso sea directo como en Windows donde presionando un icono podemos abrir el programa.

4.1.1 Paquetes de Raspbian y librerías de Python

Una de los primeros paquetes que se instalaron han sido fue Pip, necesaria para poder instalar las libreriass necesarias en Python. Pip viene del acrónimo recursivo, “Pip Installs Packages” o “Pip Installs Python”. Esto significa que este programa hace una gestión de paquetes y se utiliza para instalar y desinstalar software que se ha escrito para Python.

Existen varias maneras de instalar y gestionar paquetes in Python, pero una de las más sencillas y altamente recomendable es el uso de la herramienta por consola de PIP.

PIP permite instalar/actualizar/desinstalar cualquier paquete de Python, así como saber la lista de paquetes instalados (y los desactualizados) desde la línea de comandos. Para instalar pip en Ubuntu o Debian, ejecutamos los siguientes comandos en la consola:

```
sudo apt-get install python-pip
sudo python3 get-pip.py
```

Una vez hemos instalado PIP podemos descargar cada uno de los paquetes que necesitaremos para realizar el programa, en realidad las librerías de Python no deben ser descargadas todas al principio, principalmente porque estas ayudan a resolver problemas que van surgiendo es decir

en un primer momento no se necesitan, es a la hora de crear un mapa o conectar el programa con la base de datos por ejemplo.

La siguiente librería que fue descargada e instalada fue la wx Python, la cual se eligió para hacer una interfaz gráfica, sencilla en Python. Permite el desarrollo rápido de aplicaciones gráficas, este es uno de los módulos más importantes pero a la vez requiere bastante capacidad no solo de procesamiento sino de almacenaje en comparación con los demás.

```
sudo apt-get install python-wxgtk2.8
```

```
$ sudo apt-get install build-essential tk-dev libncurses5-dev libncursesw5-dev libreadline6-dev libdb5.3-dev libgdbm-dev libsqlite3-dev libssl-dev libbz2-dev libexpat1-dev liblzma-dev zlib1g-dev
```

Uno de los módulos imprescindibles es el pySerial para poder realizar correctamente la conexión USB con el receptor, es decir este módulo encapsula el acceso al puerto serie, además permite abrir, configurar,cerrar o comprobar los puertos en el caso del programa únicamente utilizamos la parte de la librería para abrir uno de los cuatro puertos USB disponibles.

Al igual que para las anteriores librerías se utilizará el módulo pip para instalar pySerial

```
pip install pyserial
```

4.2 Módulo de decodificación.

El módulo de decodificación fue el primero que se implementó, utilizando simplemente la interfaz de la consola se comprobaba si la decodificación sea hacía correctamente.

Anteriormente en el apartado 2.5 se ha explicado la estructura de una trama NMEA, sus campos y como se decodifica teóricamente, en este apartado se explicara cómo se realizó la decodificación de cada una de las tramas.

En primer lugar se debía capturar las tramas que aparecían por consola y convertirlas a formato texto para poder tratarlas, para ello se idearon dos maneras de introducción de tramas una que fuera manual para controlar la decodificación y la trama introducida y así saber que obtener al decodificarla y por otra parte la directamente recibida de los barcos, es decir, debido a que las tramas recibidas por los barcos no se pueden controlar sería complicado diseñar este módulo sino se controlaba la entrada y la salida. Por ello mediante un textfiled y llamando al método GetValue (Figura 15) se puede introducir la trama manualmente.

```
def Capturar(self,e):

    global data

    data = self.aivmtext.GetValue()
    longdata = len(data)
    decodificar_data(data)  # si tiene la longitud adecuada tratamos la trama
```

Figura 15. Código extraido del módulo de entrada datos. (Anexo II).

En el caso de la recepción por consola de las tramas procedentes de los barcos se debe en primer lugar encontrar el puerto en el que se encuentra conectado el dAISy AIS receiver, en el caso de la raspberry Pi, donde se debe acceder al puerto en el que se ha conectado, existen dos rutas posibles al puerto USB una es la que aparece en la figura 16 donde según el directorio de raspbian el USB está situado en /dev/ttyACM y el nombre del puerto, la otra manera de acceder al puerto USB es /dev/ttyUSB + número de USB.

```
for iPuerto in range(0, 4):
    try:
        # Puerto que vamos a probar
        Puerto = '/dev/ttyACM' + str(iPuerto)

        # Velocidad
        Velocidad = '38400'

        # Probamos a abrir el puerto
        Puerto_actual = serial.Serial(Puerto, Velocidad)
        print (Puerto_actual)

        # Cambiamos el estado de la variable para saber si lo hemos encontrado
        pEncontrado = True

        # Salimos del bucle
        break
    except Exception as e:
        # Si hay error, no hacemos nada y continuamos con la busqueda
        pass

# ¿Hemos encotrado el puerto
if pEncontrado:
```

Figura 16. Código extraido del módulo correspondiente a la conexión con el USB. (Anexo II)

Se implementó un bucle para determinar a qué puerto podría estar conectado, debido a que no tiene porque ser uno fijo, una vez se detecta el puerto donde está conectado el receptor se procede a capturar los datos mediante el método readline() iremos leyendo línea a línea(Figura 17), únicamente se deberá aplicar la codificación utf8 debido a los símbolos que hay.

Como podemos observar en la imagen en la variable puerto_actual se alamcena las variables donde se indica el puerto y la velocidad de tasa de bits.

```
def CapturaData(self):

    while self.alive.isSet():

        data = Puerto_actual.readline()                # Capturamos la trama en la variable data
        data = data.decode('UTF-8', 'replace')   # Codificamos la trama como 'UTF-8' (eliminamos las secuencias de escap

        if data != "":                              # Si data no está en blanco mostramos la trama

            print(data,  end="")
            #print ("pasamos a decodificar")# Así evitamos el salto de linea
            decodificar(data)
```

Figura 17.Código extraído del módulo capturar data. (Anexo II).

Para programar la parte de decodificación se utilizó la introducción manual de tramas por los motivos mencionados anteriormente, por ello una vez tengamos la trama el siguiente paso es diferenciar y separar los diferentes campos de esta. En el programa cada trama se ha tratado por separado por lo que aunque todas las tramas tengan las mismas partes existe una función para cada uno donde se separan los diferentes campos, se pensó de está manera ya que en las tramas

NMEA de tipo 5 por ejemplo hay partes que no están vacías y está compuesta por dos tramas por lo que sería mucho más difícil la escritura de una función estándar para todas las tramas que tratarlas una a una, además de aumentar el procesamiento requerido.

Por lo que lo primero será dirigir el tipo de trama introducida o recibida a la función correspondiente, existe dos funciones diferentes para separar los campos y tres para la decodificación del payload por lo que el proceso desde la entrada de una trama hasta que llega a su decodificador especifico.

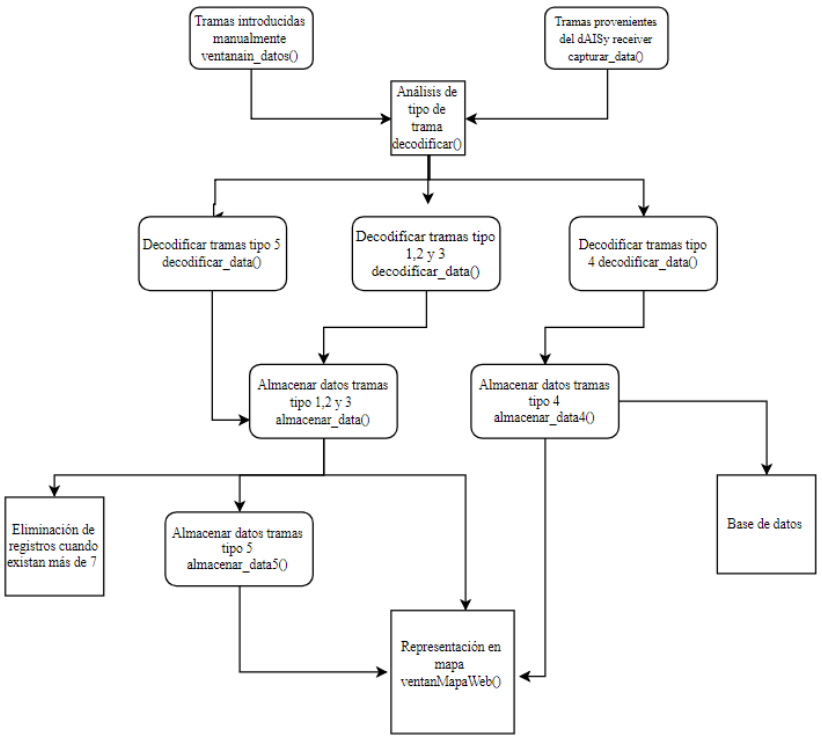

Figura 18. Diagrama del programa.

En primer lugar vamos a analizar la parte del código dedicado a seleccionar el tipo de trama que es y que decodificación es requerida para esa trama, lo cual consta de dos partes:

En la primera se discrimina utilizando las partes separadas por comas y se extrae únicamente el payload, debido a que la longitud de las tramas recibidas y según su tipo pueden variar se ideo un sistema para que independientemente de la largaría de la trama o si llegaba erróneamente se supiera a que tipo pertenece por ello, se utilizó el sistema de la partes separadas por las comas, ya que como se ha explicado en el apartado 2.5.3 la estructura de los campos de las tramas AIS es fija, por ello el quinto campo siempre corresponderá a la carga útil.

```
1748 def decodificar(data):
1749     longdatar = len(data)
1750     contador=0
1751     for i in range(longdatar):
1752         caracter = data[i:i+1]  # En esta variable vamos guardar
1753         if str(caracter) ==",":
1754             if contador==4:
1755                 caractertipo=data[i+1:i+2]
1756                 #print(str(caractertipo))
1757                 tipo(caractertipo,data)
1758                 #print("hemos vuelto de guardar y decodificar")
1759                 return
1760             else:
1761                 contador=contador+1
```

Figura 19. Código extraído del módulo de decodificación. (Anexo II).

En la segunda parte una vez tenemos el tipo y el payload separado podemos enviar esa información, según el tipo, a un decodificador u otro, por lo que en caso de que el tipo sea menor que 5 será enviado al decodificar_data(), dedicado únicamente para las tramas 1,2,3 y 4 o al decodificador dedicado para las tramas de tipo 5 en el cual cabe destacar un pequeño detalle puesto que al ser compuesta es necesario guardar también la siguiente trama (línea 1771) y mediante la función Split volvemos a separar las partes de cada trama y enviamos los dos payloads sumados.

```
1763 def tipo(caractertipo,data):
1764         #print(data)
1765         if str(caractertipo)<"5" and str(caractertipo)>"0":
1766                 decodificar_data(data)
1767                 return
1768         else:
1769             if str(caractertipo)=="5":
1770                 data1=data
1771                 data2= capturar_data()
1772                 datar_1= data1.split(',')
1773                 datar_1_1 = datar_1[0]
1774                 datar_1_2 = datar_1[1]
1775                 datar_1_3 = datar_1[2]
1776                 datar_1_4 = datar_1[3]
1777                 datar_1_5 = datar_1[4]
1778                 datar_1_6 = datar_1[5]
1779                 datar_1_7 = datar_1[6]
1780
1781                 datar_2 = data2.split(',')
1782                 datar_2_1 = datar_2[0]
1783                 datar_2_2 = datar_2[1]
1784                 datar_2_3 = datar_2[2]
1785                 datar_2_4 = datar_2[3]
1786                 datar_2_5 = datar_2[4]
1787                 datar_2_6 = datar_2[5]
1788                 datar_2_7 = datar_2[6]
1789                         #datar_2_6 = datar_2[5]
1790                 datar = datar_1_6 + datar_2_6
1791                 decodificar_data5(datar)
1792                 return
1793             else:
1794                 print("trama no soportada capturaremos otra")
1795                 return
```

Figura 20. Código extraído del módulo de decodificación. (Anexo II)

Una vez el payload ha sido enviado la siguiente fase será la decodificación, por ello trataremos cada decodificador por separado. Sin embargo antes de ello todas las tramas necesitan traducir los caracteres de 6 bit ASCII de la trama a binario, por ello deben de tener un diccionario o un traductor para poder convertir los datos.

Por ello Python facilita mediante diccionarios implementar está traducción. Básicamente, un diccionario es una estructura de datos y un tipo de dato en Python con características especiales que nos permite almacenar cualquier tipo de valor como enteros, cadenas, listas e incluso otras funciones. Estos diccionarios nos permiten además identificar cada elemento por una clave (Key).

En la imagen inferior se puede observar el diccionario diseñado para el programa, donde está representado el valor ASCII y binario de los caracteres siguiendo la metodología vista en el apartado 2.5 y utilizando el diccionario proporcionado por la ITU-R (Tabla 3)

```
1860
1861 def caracter_abit(c):
1862     """En este diccionario almacenamos los valores de 6 bits en formato string
1863     de todos los caracteres que pueden formar una trama AIS"""
1864
1865     caracter_6bits = {"0":"000000","1":"000001","2":"000010","3":"000011","4":"000100","5":"000101","6":"000110",
1866     "7":"000111","8":"001000","9":"001001",":":"001010",";":"001011","<":"001100","=":"001101",">":"001110","?":"001111",
1867     "@":"010000","A":"010001","B":"010010","C":"010011","D":"010100","E":"010101","F":"010110","G":"010111","H":"011000",
1868     "I":"011001","J":"011010","K":"011011","L":"011100","M":"011101","N":"011110","O":"011111","P":"100000","Q":"100001",
1869     "R":"100010","S":"100011","T":"100100","U":"100101","V":"100110","W":"100111","`":"101000","a":"101001","b":"101010",
1870     "c":"101011","d":"101100","e":"101101","f":"101110","g":"101111","h":"110000","i":"110001","j":"110010","k":"110011",
1871     "l":"110100","m":"110101","n":"110110","o":"110111","p":"111000","q":"111001","r":"111010","s":"111011","t":"111100",
1872     "u":"111101","v":"111110","w":"111111"}
```

Figura 21. Diccionario de decodificación en Python.

4.2.1 Decodificación de las tramas tipo 1,2,3 y 4.

Para realizar la decodificación de las tramas se debe recorrer toda la trama y transformar cada carácter ASCII de 6 bits a su equivalente en binario (Imagen de abajo).

```
1813     for i in range(28):
1814         caracter = carga_data[i:i+1]  # En esta
1815         tbinario = caracter_abit(caracter)
1816         #print (tbinario)
1817         tramabinaria = tramabinaria + tbinario
1818
1819     #print (tramabinaria)
1820     if tipo_trama == '4':
1821         #print ("Es una trama de tipo 4")
1822         tramabinaria4 = tramabinaria
1823         #print (tramabinaria4)
1824         formatear_trama4(tramabinaria4)
1825     else:
1826         formatear_trama(tramabinaria)
1827
1828     return
1829
```

Figura 21. Bucle que traduce de ASCII a binario.

A continuación se declararan unas variables globales que serán sobrescritas con cada trama nueva que se deba decodificar, esto permitirá tener acceso en cualquier punto del programa al valor de la variable.

Por ello cada trama tendrá una declaración de variables distinta en el caso de las tramas de tipo 1,2 y 3 tendrán las correspondientes a los campos de la trama.

Una vez declaradas las variables y traducidos todos los caracteres a binario tenemos como se ha explicado un cadena de bits concatenadas, la longitud de los campos de las tramas es un valor obtenido de las tablas de decodificación y por lo tanto hay que imitar el proceso descrito teóricamente.
Se asocia a cada variable el número de bits correspondientes al campo y este deberá ser traducido a decimal, exceptuando longitud y latitud que requieren un tratamiento especial.

Por ejemplo, mostramos como se convertirá el campo longitud a decimal y de igual modo se procede con la latitud, como se ha explicado en la parte teórica estos campos pueden ser

positivos o negativos, por lo que hay que tener en cuenta que en caso de que sea negativo (precedido por 1) se le debe aplicar el complemento a 2 antes de pasarlo a decimal.

En primero lugar se comprueba si es negativa, averiguando el valor del primer bit del campo, en caso afirmativo se le debe aplicar el complemento a 2, lo cual se tuvo que diseñar ya que en Python no se encontró ninguna función que te realizará dicho proceso.

Cuando se obtiene el número en decimal con o sin signo se debe obtener en el formato adecuado por ello se divide entre 600000 ya que como se ha explicado anteriormente la longitud y la latitud son dados en 1/10000 minutos y para pasarlo a grados debe dividirse entre 60.

A continuación (Figura 22) se explica brevemente como se consiguió implementar la función complemento a 2

```
2417  def complemento2(binario):
2418
2419      b = binario
2420      l = len(b)
2421      d = ""
2422
2423      # Pasamos los 0 a 1 y los 1 a 0
2424      for i in range(l):
2425          if b[i] == '0':
2426              c="1"
2427              d = d + c      # Vamos cr
2428
2429          else:
2430              c="0"
2431              d = d + c          # Vamo
2432
2433      # Sumamos 1 para acabar el compl
2434      r=int(d,2)+1
2435
2436      # El resultado lo ponemos en neg
2437
2438      r = -r
2439      return r
```

Figura 22. Código correspondiente al módulo del complemento a 2.

Recorremos la tupla binaria bit a bit comprobando cual vale 0 y cual 1, en caso de que sea 0 se convertirá en 1 y los bits que sean 1 se convertirán en 0, una vez realizada la inversión de bits debemos sumarle 1 a toda la tupla, puesto que trabajamos en binario, Python permite el manejo de números y cadenas en otras bases de manera muy sencilla por lo que simplemente sumándole 1 indicando que la tupla es binaria r=int(d,2)+1 se consigue el resultado esperado. Se convierte a entero decimal y se devuelve en negativo para el siguiente tratamiento.

Para los demás parámetros se deben de pasar directamente a decimal para ello en Python existe una sentencia sencilla que lo consigue hacer como se ha visto en el complemento a 2 las funciones de int o float permiten no solo convertir un tipo de dato de string a entero sino que el método añade la posibilidad de convertirlo a la base que se desee.

Simplemente es necesario indicar la base a la que se debe transformar y dividirlo o multiplicarlo en caso de que sea necesario para adecuarlo al formato, como es en el caso de la velocidad o el rumbo.

Finalmente una vez tenemos todos los campos convertidos a un formato legible e interpretable los almacenaremos conjuntamente de manera concatenada para su posterior uso en la base de datos, en el mapa etc.

La decodificación de la trama 4 se realiza exactamente de la misma forma puesto que los campos son similares y el que quizás tenga un formato más diferente aunque la misma manera de decodificar sean los campos de las fechas. (Figura 23).

```
2074    campo_latitud = tramabinaria4[107:134]
2075    campo_EPFD = tramabinaria4[134:138]
2076    campo_spare = tramabinaria4[138:148]
2077    campo_raim = tramabinaria4[148:149]
2078    campo_sotdma = tramabinaria4[149:168]
2079
2080    typ = int(campo_tipo, 2)
2081    rep = int(campo_repeticion, 2)
2082    mmsi = int(campo_MMSI, 2)
2083    anyo = int(campo_anyo, 2)
2084    mes = int(campo_mes, 2)
2085    dia = int(campo_dia, 2)
2086    hora = int(campo_hora, 2)
2087    minuto = int(campo_minuto, 2)
2088    segundo = int(campo_segundo, 2)
```

Figura 23. Código correspondiente a la decodificación de la trama tipo 4.

Pero debido a que cada uno de los parámetros de la fecha está dividido en un campo no es necesario realizar nigún tratamiento especial por lo que se pasará a decimal de la misma manera y se concatenará.

```
2102    trama_formateada4 = (typ, rep, mmsi, anyo, mes, dia, hora, minuto, segundo, longitud, latitud, calidad, EPFD, spare, raim)
```

Figura 24. Array para guardar los campos deocdificados y enviarlos a otras funciones

4.2.2 Decodificación tramas tipo 5

Como se ha explicado anteriormente la trama 5 es un tipo de trama compuesta por lo que la dificultad que presenta a la hora de ser tratada es que siempre serán dos tramas enlazadas distintas con campos dependientes, como resultado se debe idear alguna manera para el tratamiento, por ello los payloads de cada una de las sentencias se suman para su tratamiento es decir la continuación del útlimo bit de la primera sentencia será el primer bit del payload de la segunda sentencia.

En la siguiente imagen (Figura 25) se observa como al igual que el resto de tramas se convierten los caracteres ASCII 6 bits en binario y al estar todo el payload sumado en una única tupla de caracteres se podrá realizar la decodificación directamente.

La trama 5 tendrá sus variables particulares puesto que es la que presenta una decodificación más diferente con respecto a las otras tramas, por ello la cadena de bits del payload tiene una longitud de 424 bits.

```
1841    print ("Es una trama de tipo 5")
1842    print
1843    print (carga_data5)
1844    print
1845    print (longdatar)
1846    print
1847
1848    for i in range(longdatar):
1849        caracter = carga_data5[i:i+1]  # En esta
1850        tbinario = caracter_abit(caracter)
1851        print (tbinario,)
1852        tramabinaria5 = tramabinaria5 + tbinario
1853
1854    print
1855    print
1856    print (tramabinaria5)
1857    print ("Vamos a formatear_trama5")
1858    formatear_trama5(tramabinaria5)
```

```
2163    campo_tipo = tramabinaria5[0:6]
2164    campo_repeticion = tramabinaria5[6:8]
2165    campo_MMSI = tramabinaria5[8:38]
2166    campo_AISversion = tramabinaria5[38:40]
2167    campo_numeroimo = tramabinaria5[40:70]
2168    campo_signollamada = tramabinaria5[70:112]
2169    campo_nombrenavio = tramabinaria5[112:232]
2170    campo_tiponavio = tramabinaria5[232:240]
2171    campo_dimensionproa = tramabinaria5[240:249]
2172    campo_dimensionpopa = tramabinaria5[249:258]
2173    campo_dimensionbabor = tramabinaria5[258:264]
2174    campo_dimensionestribor = tramabinaria5[264:270]
2175    campo_EPFD = tramabinaria5[270:274]
2176    campo_ETAmes = tramabinaria5[274:278]
2177    campo_ETAdia = tramabinaria5[278:283]
2178    campo_ETAhora = tramabinaria5[283:288]
2179    campo_ETAminuto = tramabinaria5[288:294]
2180    campo_calado = tramabinaria5[294:302]
2181    campo_destino = tramabinaria5[302:422]
2182    campo_DTE = tramabinaria5[422:423]
2183    campo_spare = tramabinaria5[423:424]
```

Figura 25. Código correspondiente a la selección de campos y su decodificación de la trama tipo 5.

En cuanto a la conversión a formato decimal es exactamente la misma que la realiza en las anteriores tramas sin embargo existen 3 parámetros que no se pueden representar en decimal de base 10 puesto que son caracteres ASCII , por ello se deberá realizar el paso contrario que se ha hecho hasta hora, es decir con la tupla binaria, se debe convertir la cadena de bits a su equivalente en ASCII.

```
2199    signollamada = debitsa_ascii(campo_signollamada)
2200    print (signollamada)
2201    nombrenavio = debitsa_ascii(campo_nombrenavio)
2202    print (nombrenavio)
2203    tiponavio = int(campo_tiponavio, 2)
2204    print (tiponavio)
2205    dimensionproa = int(campo_dimensionproa, 2)
2206    print (dimensionproa)
2207    dimensionpopa = int(campo_dimensionpopa, 2)
```

Figura 26. Código correspondiente a la traducción de los parámetros compuestos por ASCII.

En la figura 26 se observa que se llama al método debitsa_ascii donde se hará la conversión. Por ello se debe escribir otro diccionario pero en este caso será justo al contrario, cada tupla de 6 bits tendrá un carácter equivalente en ASCII.

```
ascii_6bits = {"000000": "@", "000001": "A", "000010": "B", "000011": "C", "000100": "D", "000101": "E", "000110": "F",
               "000111": "G", "001000": "H", "001001": "I", "001010": "J", "001011": "K", "001100": "L", "001101": "M",
               "001110": "N", "001111": "O", "010000": "P", "010001": "Q", "010010": "R", "010011": "S", "010100": "T",
               "010101": "U", "010110": "V", "010111": "W", "011000": "X", "011001": "Y", "011010": "Z", "011011": "[",
               "011100": "\\", "011101": "]", "011110": "^", "011111": "_", "100000": " ", "100001": "!", "100010": "\"",
               "100011": "#", "100100": "$", "100101": "%", "100110": "&", "100111": "'", "101000": "(", "101001": ")",
               "101010": "*", "101011": "+", "101100": ",", "101101": "-", "101110": ".", "101111": "/", "110000": "0",
               "110001": "1", "110010": "2", "110011": "3", "110100": "4", "110101": "5", "110110": "6", "110111": "7",
               "111000": "8", "111001": "9", "111010": ":", "111011": ";", "111100": "<", "111101": "=", "111110": ">",
               "111111": "?"}
```

Figura 27. Diccionario para la traducción de binario 6 bits a ASCII.

Para realizar el proceso contrario debemos dividir la longitud del campo entre 6 debido a que es la manera que independientemente la longitud del campo se traduzca correctamente, es decir agrupamos los bits de 6 en 6 y mediante un bucle recorremos la tupla binaria traduciendo cada 6 bits. Por lo que el bucle se repetirá tantas veces como tuplas de 6 bits haya (Figura 28).

```
def debitsa_ascii(campo):

    lcampo = len(campo)
    print ("este campo mide")
    print (lcampo)
    veces = lcampo / 6
    cadena = ""
    caracter = ""

    for i in range(0, int(veces)):

        j = i * 6
        h = j + 6
        bits = campo[j:h]
        caracter = decodsixbits_ascii(bits)
        cadena = cadena + caracter

    return cadena
```

Figura 28. Módulo dedicado a la traducción de binario a ASCII 6 bits

4.3 Interfaz Gráfica

Como se ha mencionado antes la librería utilizada para implementar la interfaz gráfica ha sido wxPython, en todo momento se ha buscado una interfaz sencilla, simple y que cumpliera la función de facilitar la interpretación de los datos. En un primer momento la ejecución y monitorización de los datos se realizaba a través de consola (Figura 29), lo cual supone una dificultad a la hora de interpretar los datos.

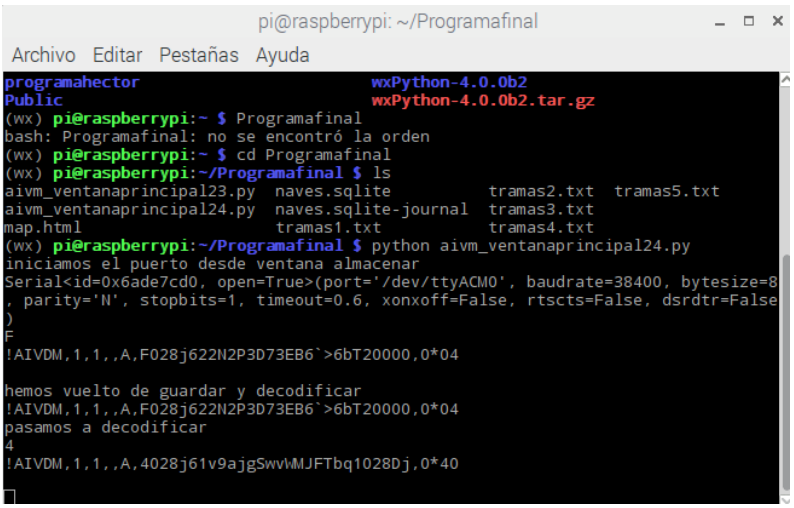

Figura 29.Ejecución de programa mediante consola.

La interfaz ha sido desarrollada en gran medida en un ordenador diferente a la raspberry debido a los requisitos exigidos y la necesidad de un programa más completo que los editores de texto instalados en la raspberry.

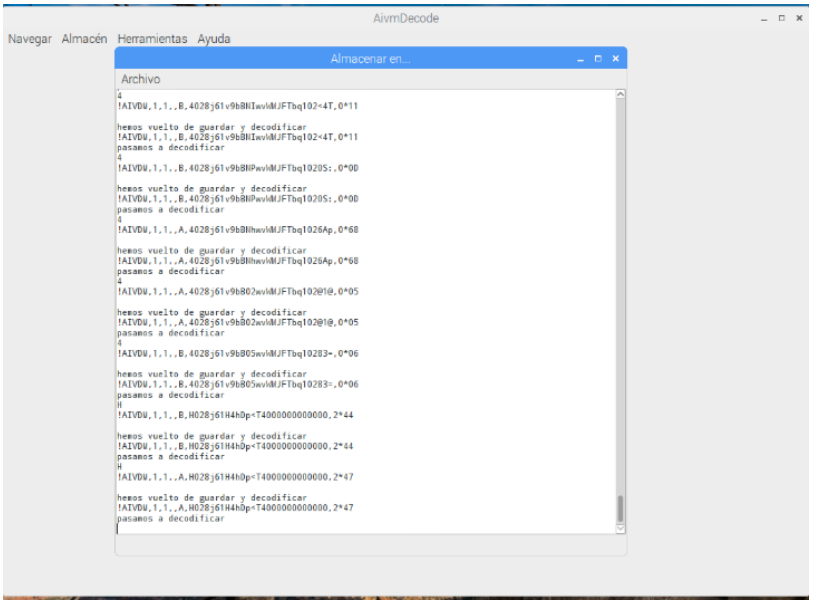

Figura 30.Menu principal del programa.

Un menú superior desde el que poder manejar cada elemento del programa, subdividido en 5 submenus diferentes, en el cual el primero serviría para la creación de mapas, el segundo serviría para una futura aplicación de uso del receptor a modo de GPS, el tercero nos permite la captura de tramas (Figura 31) y los dos últimos sirven para introducir tramas manualmente (Figura 30)

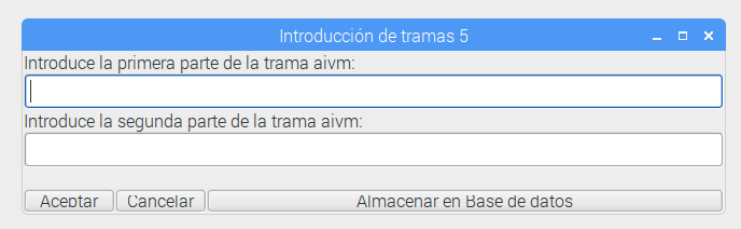

Figura 31.Menu de introducción de tramas.

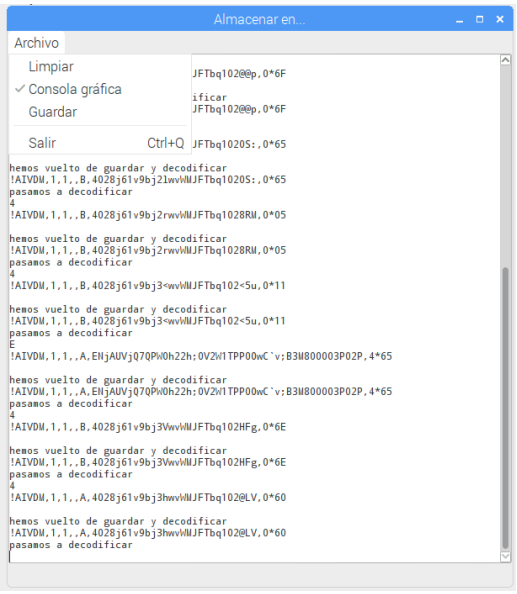

Figura 32.Menu de captura de datos.

Para la programación de estos módulos se ha hecho uso de las librerías de wxPyhon, donde se ha creado un menú principal, el cual nos permite saltar a diferentes frames creados para cada una de las funciones que se mostraran en el entorno virtual de wxpython.

Pese a que toda la ejecución pueda realizarse a través de esta interfaz gráfica, el programa se debe lanzar a través de consola, existe la posibilidad de crear un ejecutable para que lanzar el programa, lo que sugiere una línea futura a desarrollar en el programa.

4.4 Diseño modular de la base de datos

La siguiente parte del programa está dedicada a la creación de una base de datos donde poder almacenar el registro de los barcos y su gestión, para ello se pensó en diferentes formas de realizarla debido a la diversidad de datos existentes y a su dinamismo.

Finalmente se planteó el siguiente diagrama de tal manera que fuera efectivo y no tuviera una gran cantidad de datos.

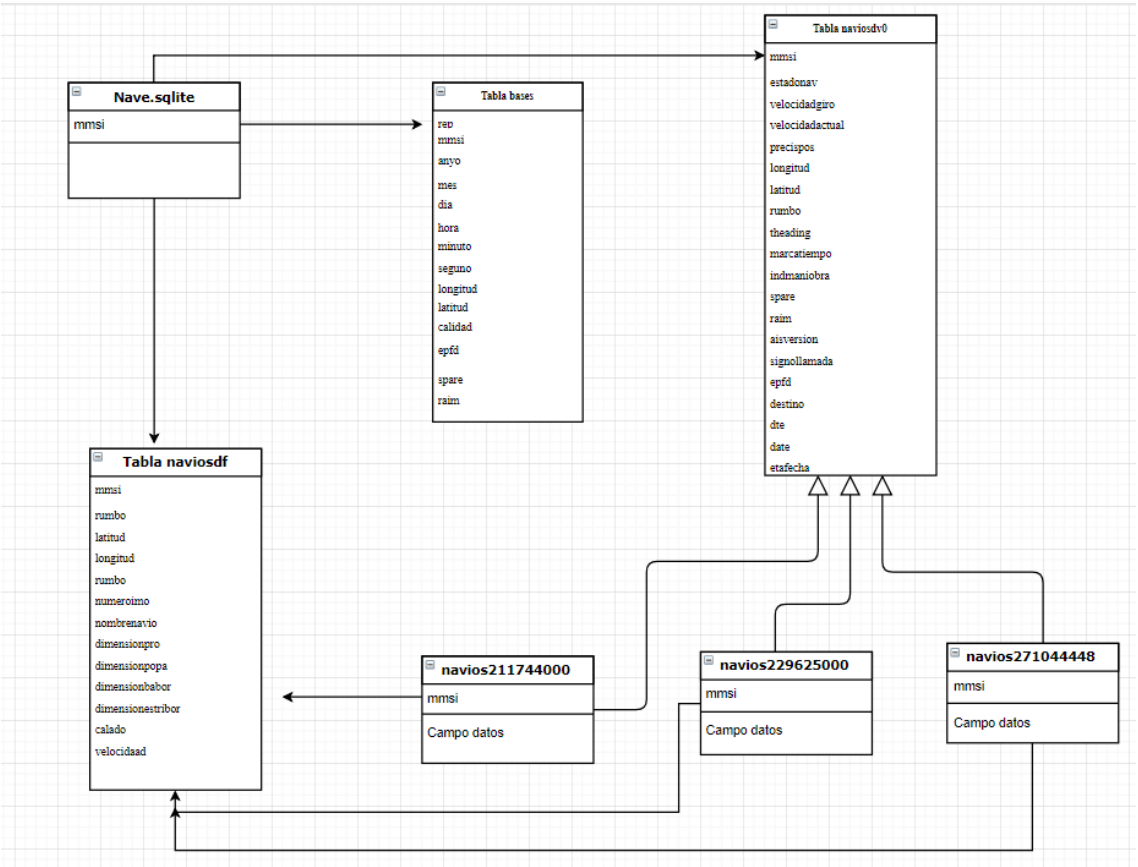

Figura 33.Esquema modular de la base de datos.

La base de datos se estructura en tres grandes tablas que son las principales “naviosdf”, “naviosdv0” y “bases” cómo podemos ver en la imagen(Figura 33) donde se genera una tabla para los datos de los barcos que sean variables sin embargo está será una tabla base sin datos que estará vacía, únicamente la utilizaremos para ser copiada, es decir debido a que el proceso de creación de una tabla mediante Python exigía un mayor procesamiento a la raspberry, si se

copiaba la tabla y se le cambiaba el nombre ahorraría procesamiento de creación innecesario cada vez que entrara un nuevo barco por lo que el procesamiento recae en la base de datos .

La siguiente tabla “naviosdf” sirve para almacenar los datos fijos de los barcos, es decir aquellos que son estáticos, además se pensó que para representar el mapa de todos los barcos, este obtuviera los datos de los diferentes barcos desde la base de datos por lo que está tabla se utiliza a modo de última posición válida para la representación de los barcos.

La tercera tabla es la correspondiente a las estaciones bases, la cual es independiente de las otras tablas, esta tabla se ha utilizado para guardar todos los datos que aportan las estaciones fijas que debido a su posición fija requieren un tratamiento más sencillo.

Además nos aporta la fecha y hora UTC por lo que ayuda a la geolocalización y en algunos casos principalmente en casos de emergencias es posible que se requiera de su posición.

El índice primario que servirá para enlazar y distinguir cada tabla será el mmsi puesto que todas las tramas deben llevar uno y es el número identificador de cada uno de los barcos, estaciones barcos etc.

Como se ha explicado anteriormente la tabla naviosdv0 sirve de tabla base para crear la tabla de cada navio, la idea de que cada navio tenga una tabla independiente sirve para poder manejar la gran cantidad de datos dinámicos que cada barco transmite, además estás tablas sirven para crear el mapa de la trayectoria de cada barco, debido a que en está tabla se guarda el registro de todos los datos que han sido recibidos correspondientes al mmsi de cada barco. Por lo tanto podemos determinar que trayectoria ha seguido simplemente realizando una consulta a la tabla de las 10 últimas posiciones de los barcos por ejemplo. Por ello manejaremos una gran cantidad de datos con respecto a cada barco(Figura 33).

4.4.1 Sistema de gestión de bases de datos.

Una vez se ha diseñado el diagrama que tendrá la base de datos, se debía buscar un formato de base de datos que cumpliera una serie de características debido a las condiciones en las que podría estar la raspberry y la capacidad de esta, por lo que dados estás especificaciones se requería un formato que cumpliera las siguientes características:

- Almacenamiento local, sin necesidad de servidores ni acceso a internet.
- Velocidad de acceso, latencia baja.
- Fácil manejo, interoperabilidad y portabilidad
- Gran capacidad de almacenaje y compresión de datos.
- Compatibilidad con lenguajes sql para realizar consultas externas.
- Suficientemente segura, fiable y robusta

Por ello se eligió sqlite, debido al sistema que tiene para gestionar la comunicación cliente servidor, el proceso llevado a cabo por SQLite no es independiente del programa. En lugar de eso, SQLite se enlaza directamente con esto por lo que pasa a formar parte de la aplicación. El programa hace llamadas a subrutinas y a las funciones propias de la librería para utilizar las funcionalidades de SQLite. Esto reduce la latencia en el acceso a la base de datos, básicamente debido a que la instanciación de funciones resulta más eficiente que la comunicación entre

procesos. Lo que servirá a la hora de guardar una gran cantidad de datos con un flujo ininterrumpido como es en el caso de zonas de tráfico denso de barcos.

El conjunto de la base de datos son guardados de manera local en un único archivo. Este diseño simple se logra comprimiendo toda la base de datos, nos permitirá cumplir el segundo requisito, ya que a diferencia de otros sistemas de gestión, como mysql se requiere una interfaz más compleja para poder gestionar los sql, sin embargo con sqlite es un simple archivo capaz de ser interpretado por casi cualquier diseñador y gestionador de bases de datos cumpliendo otro requisito. En definitiva SQLite tiene una pequeña memoria y una única biblioteca es necesaria para acceder a bases de datos, lo que lo hace ideal para aplicaciones de bases de datos incorporadas

En su última versión, SQLite permite crear bases de datos de una capacidad de hasta 2 Terabytes de tamaño. SQLite es un sistema completo de bases de datos que permite soportar diversas tablas, índices y vistas por lo que puede adaptarse perfectamente a nuestro proyecto. No se requiere un proceso independiente ejecutándose en un servidor ya que lee y escribe directamente sobre archivos que se encuentran en el disco duro. El formato de la base de datos es multiplataforma y el archivo es perfectamente compatible tanto en sistemas de 32 como de 64 bits con lo que la compatibilidad está asegurada.

En cuanto a la estabilidad SQLite es compatible con ACID, reunión de los cuatro criterios de Atomicidad, Consistencia, Aislamiento y Durabilidad. El costo de SQLite es de dominio público, y por tanto, es opensource por lo que puede utilizarse para cualquier aplicación o proyecto y se puede redistribuir libremente.

En resumen los requisitos especificados son superados por las principales características de sqlite lo cual lo hacía perfecto para la gestión de la base de datos del proyecto,

Se usaron dos herramientas diferentes para gestionar la creación y gestión de está, una de ellas era un programa para Windows llamado DB Browser for SQLite es una herramienta de alta calidad, visual y de código abierto para crear, diseñar y editar archivos de bases de datos compatibles con SQLite.

En cuanto al entorno de raspbian se eligión una herramienta de Firefox para poder gestionar la base de datos llamada SQlite manager.

4.4.1 Estructura del código de gestión de la base de datos

Una vez tenemos las herramientas y el motor que utilizaremos, pasamos a la parte de conexión de nuestro programa de decodificación con la base de datos creada, para ello desde Python únicamente se realizará la entrada de datos, la gestión de estos para evitar exceso de almacenaje y finalmente consulta de parámetros para la creación de los mapas, lo cual se explicará en el apartado siguiente.

El módulo dedicado a la gestión de la base de datos se ha dividido únicamente en tres funciones distintas almacenar_datos(t), almacenar_datos4(t) y almacenar_datos5(t). Como se puede observar está distribuido según la trama que deba almacenar, lo que se puede ver claramente en

el diagrama (Figura 33). Está estructura jerárquica nos permite una gestión totalmente independiente de las tramas.

En primer lugar se debe establecer la conexión con la base de datos creádas que en este caso es navios.sqlite.

```
db = sqlite3.connect('navios.db')
cur = db.cursor()
cur.execute("función sql adaptada a python")
```

Solamente se debe indicar la ruta donde se encuentra almacenada y mediante la función connect de la librería sqlite establecemos la conexión, previamente se deben de haber descargado e instalado las librerías sqlite3 e importarlas específicas para Python, a continuación se crea un cursor que nos permitirá apuntar a los parámetros de las tablas que hemos llamado cur.

El módulo de almacenar_datos() no solo se encarga de almacenar los datos de las tramas de tipo 1,2 y 3 también es utilizado para direccionar en caso de que alguno de los mmsi que entran sean repetidos. Por ello en caso de que en la tabla de datos fijos, naviosdf, exista ya un mmsi igual al que se ha recibido, este será redireccionado a su tabla correspondiente

Por ello lo primero que se discrimina es si la trama es de tipo 1 o de tipo 5 para ser tratada en su correspondiente módulo, a continuación en caso de ser una trama de tipo 1 se almacenan los campos que se han enviado desde el módulo de decodificación

Para a continuación comprobar si el mmsi que se ha recibido ha sido anteriormente almacenado o no, es decir

```
2030    cur.execute("SELECT * FROM naviosdf WHERE mmsi = ?", (d,))
2031    row = cur.fetchone()                    # Con esta sentencia vemos si la tabla naviosdf tiene el registro del navio (Está dado de alta)
2032    nombretabla = "naviosdv" + str(mmsi)
2033
2034
2035    if row:                                 # Si, hay registros
```

Figura 34.Código correspondiente a una consulta general a partir del mmsi en python.

En caso de que ese barco ya este registrado en nuestro barco, los nuevos datos que han sido recibidos deben ser almacenados en la tabla de datos variables correspondiente del barco y la tabla de datos fijos de todos los barcos que han sido registrados hasta ese momento será actualizada ya que únicamente se guarda el mismo barco una vez, es decir este es un ejemplo de la base de datos (Tabla 11) después de dos días capturando desde una posición alejada:

Tabla: naviosdv255805852 — Nuevo registro — Borrar registro — Nuevo registro — Borrar re

	mmsi	estadonav	velocidadgiro	velocidadactual	precispos	longitud	latitud	rumbo	theading	marcatiempo	inc	date	eta
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
1	255805852	0	0	10.0	0	-0.217481666...	39.42980833...	230.0	230	41	0	2018-05-12 11:51:43.198022	NULL
2	255805852	0	0	9.3	0	-0.219601666...	39.42845	230.0	230	29	0	2018-05-12 11:52:30.557874	NULL
3	255805852	0	0	7.5	0	-0.2256	39.424945	235.0	235	0	0	2018-05-12 11:55:01.333679	NULL
4	255805852	0	0	7.2	0	-0.226653333...	39.42440166...	235.7	235	29	0	2018-05-12 11:55:30.559763	NULL
5	255805852	0	0	7.0	0	-0.227731666...	39.42381333...	235.0	235	0	0	2018-05-12 11:56:01.337550	NULL

Tabla 11.Tabla generada de los datos variables recibidos de un barco.

En esta primera imagen observamos como los datos del mismo barco han sido actualizados, en este caso el almacenamiento se restringió a 7 registros, debido a la redundancia de algunos datos. La tabla ha sido creada exclusivamente para este barco y se comprende el diseño de la base de datos. El nombre de la tabla estará formado por naviosdv (navios datos variables) + "mmsi", que cambiará con cada barco añadido. Por ello como se ha explicado anteriormente la forma más conveniente de crear estas tablas no sería haciéndolas desde 0 sino que se deberá copiar de la tabla maestra naviosdv0 (Figura 35)la cual tiene creada ya la estructura y únicamente es necesario introducir nuevos registros.

```
cur.execute("INSERT INTO naviosdf (mmsi, longitud, latitud, rumbo,velocidad) VALUES (?,?,?,?,?)", (d,e,f,g,v,))

# Creamos la tabla del nuevo navio

cur.execute("SELECT sql FROM sqlite_master WHERE type='table' AND name='naviosdv0'")
txt=(cur.fetchone()[0])
txt=txt.replace("naviosdv0",nombretabla)                                    # Con este grupo de sentenc
cur.execute(txt)
cur.execute("""INSERT INTO """ + nombretabla + """ SELECT * FROM naviosdv0""")
```

Figura 35.Código correspondiente a la introducción de los campos y la copia de la tabla master naviosdv0.

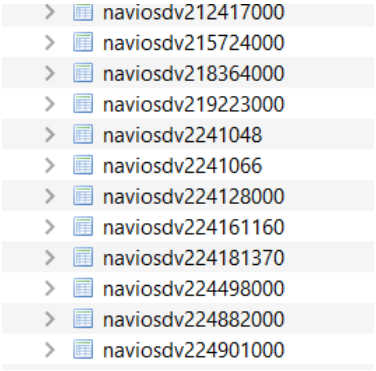

Tabla 12.Tabla de los diferentes navíos generadas en la recepción.

El código empleado para la introducción de datos es un tipo especial de sql puesto que ha sido adaptado para Python, por lo que para la introducción de variables provenientes de Python se debe sustituir como vemos en la fórmula siguiente, de tal manera que la tabla que cambiará de nombre dependiendo del mmsi recibido, en cuanto a las variables provenientes de Python existen dos sintaxis: la primera entre tres comillas y la segunda referenciándolas mediante interrogante como podemos observar:

```
cursor.execute("INSERT INTO "+nombretabla+"(variables a introducir
)VALUES(?,?,?,?)", variables_python)
```

En la siguiente imagen se aprecia como los datos de este barco están registrados en la tabla naviosdf, junto con los de este barco están cada uno de los que hayan sido recibidos y está tabla se actualiza con cada barco nuevo que entra o en caso de que el mmsi comparado exista se deben introducir los nuevos datos eliminando los anteriores(Tabla).

Para actualizar los registros se utilizará la función UPDATE (línea de código 2040) y para introducir los nuevos datos al igual que para la tabla naviosdv se utilizará INSERT (línea de código 2043)

Tabla: naviosdf

	mmsi	longitud	latitud	rumbo	numeroimo	nombrenavio	tipc
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
1	2241048	-0.30222	39.44913166...	-0.30222	NULL	NULL	NULL
2	373233000	-0.328376666...	39.43481833...	246.7	NULL	NULL	NULL
3	2241066	-0.324903333...	39.87223833...	-0.324903333...	NULL	NULL	NULL
4	247281400	-0.248116666...	39.42645	273.0	9498743	VISEMAR ONE	69
5	311055200	-0.319323333...	39.44446666...	81.8	NULL	NULL	NULL
6	373298000	-0.323236666...	39.44566166...	254.7	9524023	CORAL QUEEN	70
7	224498000	-0.32497	39.45473833...	272.9	NULL	NULL	NULL
8	247320400	115934.0	39.380215	276.0	NULL	NULL	NULL
9	277387000	-0.32795	39.4346	113.0	NULL	NULL	NULL
10	224128000	-0.305768333...	39.43344833...	303.0	NULL	NULL	NULL
11	256878000	-0.323428333...	39.45553666...	0.0	NULL	NULL	NULL
12	211335760	-0.316986666...	39.44947833...	331.0	NULL	NULL	NULL
13	374602000	-0.328745	39.43805666...	302.8	NULL	NULL	NULL
14	224901000	-0.301736666...	39.43162333...	86.4	NULL	NULL	NULL
15	224882000	-0.3245	39.45466666...	360.0	NULL	NULL	NULL
16	209115000	-0.323621666...	39.44915	41.2	NULL	NULL	NULL
17	304831000	-0.302783333...	39.43047	132.4	NULL	NULL	NULL
18	225437000	-0.3172	39.44236	359.9	NULL	NULL	NULL
19	538003317	-0.3264	39.43393333...	20.0	NULL	NULL	NULL
20	247086400	-0.302916666...	39.43101666...	138.0	NULL	NULL	NULL
21	210976000	-0.320333333...	39.4475	101.0	NULL	NULL	NULL
22	224161160	-0.3285	39.43616666...	13.9	NULL	NULL	NULL

Tabla 12.Estructura correspondiente a la tabla de los datos fijos de los barcos.

El siguiente módulo es almacenar_datos4(t), en este módulo funciona de manera muy similar al resto de tramas sin embargo esta parte del código solo debe gestionar una única tabla, bases.

El procedimiento será como se ha explicado para almacenar_datos(t) :

1. Conexión con la base de datos
2. Almacenamiento de los campos concatenados en variables diferentes
3. Comprobación de mmsi registrado
   3.1 En caso afirmativo se actualizarán los valores mediante una función UPDATE
   3.2 En caso negativo se introduce nuevamente un registro INSERT

Es destacable que está tabla tendrá un volumen mucho menor debido a que el número de estaciones bases no es comparable con el número de barcos.

Por último el modulo dedicado a las tramas de tipo 5 almacenar_datos5(t) tiene un esquema similar al que utilizan el resto de tramas cuando el registro existe y únicamente se debe acceder y actualizar con los nuevos datos, esto se entiende fácilmente ya que los registros de los diferentes barcos no están totalmente completados hasta que una trama de tipo 5 perteneciente a un barco llega, sin embargo estas son bastante poco frecuentes, principalmente los barcos de grandes dimensiones son los que aportan esta información principalmente a modo de aviso para el puerto y los barcos colindantes.

En conclusión este módulo ha sido ideado de forma que únicamente si existe un registro anterior del barco se almacenarán los campos correspondientes a esta trama y por consiguiente descartando aquellas que no hayan sido registradas previamente, el objetivo es principalmente que no hallan barcos no localizados dentro de la base de datos.

4.5 Representación y geolocalización de los barcos

Uno de los apartados más importantes a nivel gráfico para una aplicación posterior donde recrear a tiempo real el trafico entrante y saliente que el terminal AIS puede detectar sería el generador de mapas dinámicos, el cuál en primer lugar se pensó realizarse con una imagen estática mediante la librería matplotlib, es decir mediante mapas estáticos y una librería para representar puntos y líneas se crearía un mapa dimensionado manualmente, sin embargo el problema de generar el mapa de esta manera, sin embargo generar el mapa de esta manera presenta una serie de desventajas y dificultades que hizo que se descartara generar el mapa de esta manera:

- Mapa totalmente fijo, imposibilidad de representar puntos más allá de las dimensiones de la imagen o imágenes.
- Problemas con el cluster (grupo, en este caso la masiva agrupación de barcos), debido a la imposibilidad de acercar y alejar, al no existir una opción de zoom suficientemente nítida
- Dificultad, principalmente para dimensionar la imagen, debido a que sería necesario más imágenes para realizar una representación veraz, clara y nítida.
- Poca versatibilidad y compatibilidad, debido a que el diseño se debía hacer en Python esto dificultaba la interacción con otros lenguajes de programación a nivel de objetos.

  A pesar de todas las desventajas que presentaba, se llegó valorar está opción debido a que presenta dos ventajas muy importantes con respecto a otras librerías que permitan la representación y gestión de mapas
- Ausencia de necesidad de conexión a la red
- Librería más amplia para la representación de objetos de temática naval.

  El mapa estático se presentaba de esta manera, donde previamente se había guardado la imagen, adaptado y dimensionado (Figura 36).

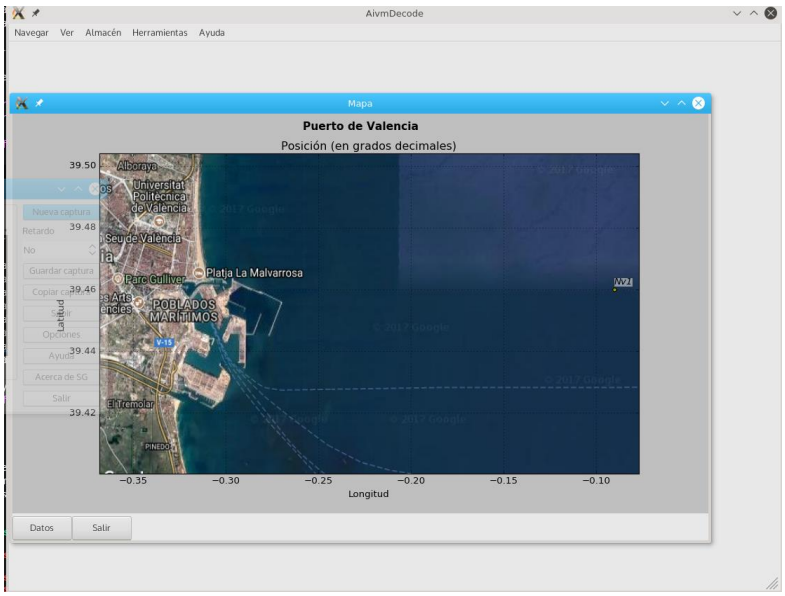

Figura 36.Mapa representado con Matplotlib

Por eso se buscó una opción que fuera más dinámica que permitierá la versatiblidad que matplolib no tenía y una gestión mucho más simple, finalmente se encontró una librería bastante novedosa que trabajaba con Python y generaba mapas dinámicos, la librería es Folium.

Folium se basa en los puntos fuertes del ecosistema Python y las fortalezas de mapeo de la biblioteca Leaflet.js.

Folium facilita la visualización de datos manipulados en Python en un mapa interactivo. Permite tanto el enlace de datos a un mapa para visualizaciones de coloridas como el paso de visualizaciones de Vincent / Vega como marcadores en el mapa.

Al basarse en Leaflet, está librería genera mediante sentencias en Python, código html y javascript, por lo que permite ser relacionado con más lenguajes de programación como php el propio html, css , java etc por lo que ofrece infinitas oportunidades a la hora de seguir desarrollando la aplicación software.

La librería folium requiere de la librería "Pandas" y "webbrowser", está última sirve para ejecutar en el navegador el archivo .html generado y que de otra manera queda guardado en la ruta especificada.

El mapa es generado mediante una sencillísima sentencia en Python la cual genera un mapa global totalmente interactivo y dinámico, como si de google maps se tratase tiene insertado las poblaciones, carreteras, rutas marítimas predeterminadas etc,

La sentencia que genera el mapa será la siguiente:

```
Nombre del mapa = folium.Map(location=[45.5236, -122.6750])
```

Unicamente es necesario especificar las coordenadas donde queremos centrar el mapa, consiguiendo un mapa global por el cual se puede desplazar con total libertad y perfectamente dimensionado.

Sin embargo folium presenta un gran inconveniente, ya que para la representación de objetos y de markers se ve un poco limitado debido a que es una librería en desarrollo y existe una gama insuficiente de markers para poder generar mapas de diferentes tipos y destacar los puntos que se deseen.

Existe uno ya creado que se asemeja bastante a la forma de un barco y nos permite conocer el rumbo y la dirección de los barcos, a este objeto debido a su representación mediante javascript nos permite añadirle características interesantes como la integrada en el proyecto donde se puede representar mediante un pop up los datos más importantes del barco como la posición, nombre o destino en una tabla sencilla hecha con html.

```
folium.Marker([lati, longi],popup=popup,icon=folium.Icon(color='red',icon='info-sign')).add_to(map)
```

La sentencia corresponde a un marcador para las estaciones base, donde se especifica sus coordenadas, que curiosamente deben ser introducidas al revés de lo que se haría en un mapa debido a la sintaxis de Folium, seguido del popup y su nombre y el icono que se desea en este caso para las estaciones base se seleccionó un icono de información, que como se ha explicado anteriormente debido a la escasez de marcadores de temática naval se optó por este.

En cuanto a la sentencia que genera los marcadores relativos a los barcos se diferencian principalmente en que estos están provistos de rumbo el cual es obtenido de la base de datos realizando previamente una consulta (Figura 37) y recorriendo al igual que con las bases de datos los registros existentes.

```
plugins.BoatMarker(location=(lati,longi),heading=rumbo,wind_heading=150,wind_s
                peed=45,color='#8f8',popup =popup ).add_to(map)
```

```
cursor.execute('''SELECT longitud,latitud,rumbo,velocidad,mmsi FROM naviosdf''' )

filas = cursor.fetchall()
for row in filas:

            longi =float(row[0])
            lati = float(row[1])
            rumbo=float(row[2])
            velo=str(row[3])
            mmsi=(row[4])
```

Figura 37.Código correspondiente a la consulta de los parámetros.

El pop up funciona a modo de página html totalmente en blanco. Donde cualquier sentencia permite actuar a este como enlace para representar el mapa de la trayectoria simplemente pulsando el botón como observamos en la (Figura 38).

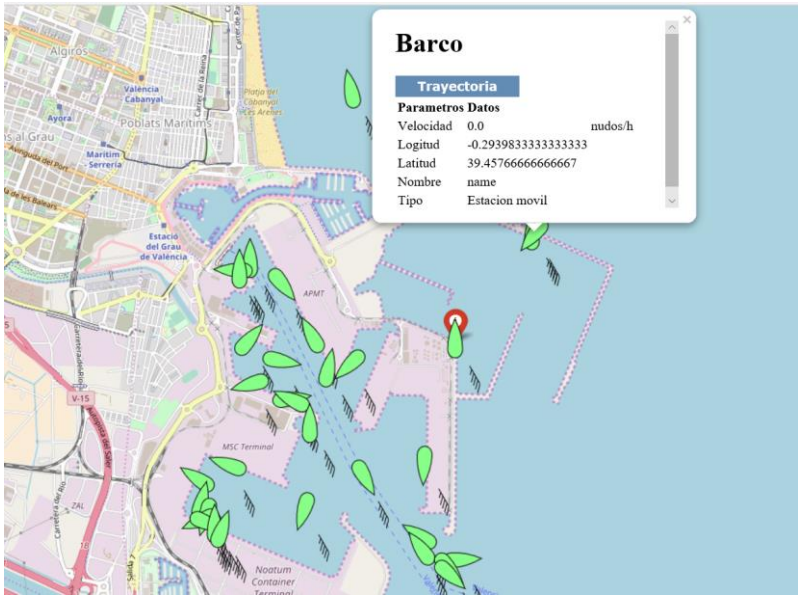

Figura 38. Mapa del puerto de valencia con los barcos almacenados en la base de datos.

En la imagen se puede observar la representación de un conjunto de barcos que han tenido esa posición en el pasado es decir, el terminal AIS no funciona a modo de radar puesto que el mapa no se ejecuta a tiempo real debido a que la entrada de tramas no es simultánea como se ha

explicado en capítulos anteriores, por lo que siempre existirá cierto retraso exceptuando los barcos de grandes dimensiones que tienen preferencia en la transmisión de slots.

Por ello requiere un mínimo de tiempo para poder tener toda la representación de un puerto con gran tráfico marino, sin embargo nos permite localizar a los diferentes barcos y la posibilidad de en un futuro realizar una sincronización casi instantánea de los barcos recibidos.

Sin embargo como se ha mencionado las ventajas que aporta el AIS no solo sirven para la geolocalización puesto que los barcos emiten muchos más datos que un simple radar y se puede saber en menos de dos minutos si un barco está a la espera para entrar a puerto, por lo que permite tener grandes oportunidades comerciales.

El código que genera el mapa con las posiciones de los diferentes barcos depende en su totalidad de las librerías Folium, por lo que debe tener conexión a internet para poder generar el mapa, lo cual se puede solucionar si se integran en la raspberry pi, y de la base de datos del programa.

El módulo dedicado a la base de datos no requiere de una estructura compleja, únicamente se requieren dos consultas una a la tabla bases para obtener las estaciones base existentes y posicionarlas en el mapa y la otra a la tabla naviosdf debido a que solo interesará la última posición enviada por el barco. Estas dos consultas deben estar controladas por un bucle de tal manera que se representen todos los puntos, por ello el mapa es previamente generado con los datos de la base de datos, es decir el mapa es generado de nuevo con cada barco añadido a la base de datos. Por lo que no existe una sincronización automática ni simultánea.

Por último es importante resaltar la combinación de tres lenguajes de programación totalmente diferentes, Python, html y javascript en la siguiente imagen(Figura se puede observar cómo se integran los tres lenguajes, donde Python sirve de generador y gestionador del resto, html permite la visualización de la interfaz y javascript logra dotar de dinamismo al mapa.

```
1338        cursor.execute('''SELECT longitud,latitud FROM bases ''' )
1339        filas = cursor.fetchall()
1340        for row in filas:
1341                longi =float(row[0])
1342                lati = float(row[1])
1343
1344                html ="""
1345                                    <html>
1346
1347                                    <head><title>Ejemplo de tabla sencilla</title></head>
1348                                    <body>
1349
1350                                    <h1>Barco</h1>
1351
1352                                    <table>
1353                                    <tr>
1354                                        <td><strong>Parametros</strong></td>
1355                                        <td><strong>Datos</strong></td>
1356                                        <td><strong></strong></td>
1357                                    </tr>
1358
1359                                    <tr>
1360                                        <td>Logitud</td>
1361                                        <td>"""+str(longi)+"""</td>
1362                                        <td></td>
1363                                    </tr>
1364
1365                                    <tr>
1366                                        <td>Latitud0</td>
1367                                        <td>"""+str(lati)+"""</td>
1368                                        <td></td>
1369
1370                                        <td>Tipo</td>
1371                                        <td>Estacion base</td>
1372                                        <td></td>
1373                                    </tr>
1374                                    </table>
1375
1376                                    </body>
1377                                    </html> """
1378                iframe = folium.IFrame(html=html, width=350, height=225)
1379                popup = folium.Popup(iframe, max_width=2650)
```

Figura 39.Código correspondiente a la creación del pop up.

La imagen corresponde a la generación de las estaciones bases que están guardadas en la base de datos, mediante un bucle recorremos cada una de las filass donde exista registro de estaciones base, a continuación se implementa mediante html y css para maquetarlo y embellecerlo el pop up que nos servirá como interfaz para mostrar los datos más importantes de cada uno de los barcos a continuación mediante un iframe y el código javascript generado por Python permitirá generar un pop up cada vez que se presione un barco.

4.5.1 Observaciones

Sin embargo es posible implementar la automatización del mapa con la actualización o entrada de un barco mediante java script, además de poder realizarlo a una velocidad casi simultánea, sin embargo se requiere de bastante velocidad de procesamiento para realizar estas dos funciones a la vez, aunque la raspberry Pi da la suficiente fluideza para poder observar los cambios de posición y de movimiento de los barcos.

Por otra parte debido a que la entrada de tramas viene determinada por factores agenos al terminal por lo que a pesar de tener toda la capacidad de procesamiento deseada seguiría habiendo barcos que no tendrían una actualización simultánea, sin embargo es un sistema muy fiable y se acerca lo suficiente para detectar todos los barcos que existen alrededor. Además el AIS no tiene una ráfaga constante de emisiones debido a que en las zonas de abundante tráfico marino debe esperar para que las rendijas de tiempo estén libres.

Por último cabe destacar el posible cluster (Figura 40) que pueda existir en zonas con demasiado tráfico marino, como se puede observar en esta imagen del puerto de valencia en un día normal.

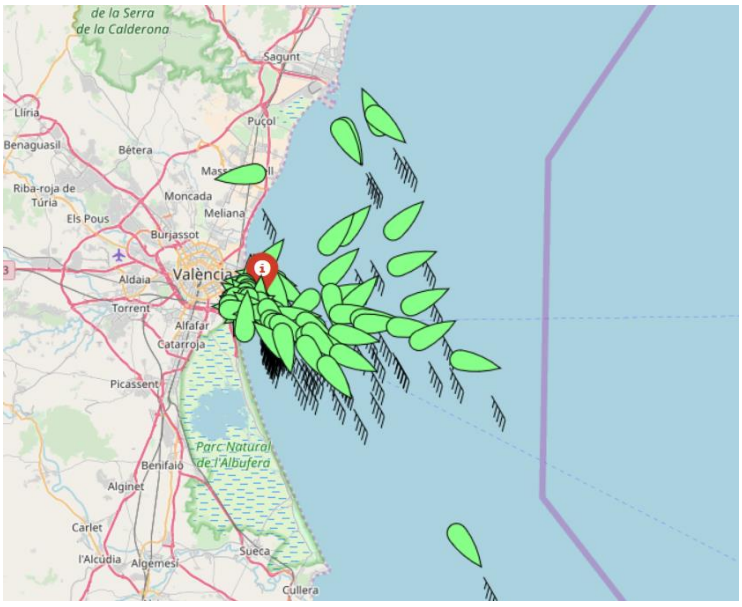

Figura 40. Mapa del cluster generado debido al tráfico existente en el puerto.

Esto se debe a la alta densidad de tráfico marino sin embargo es posible evitarlo mediante el zoom en el mapa, sin embargo debido a la continua actualización de esta novedosa librería, posiblemente en unos meses existirá una función que permite evitar el cluster.

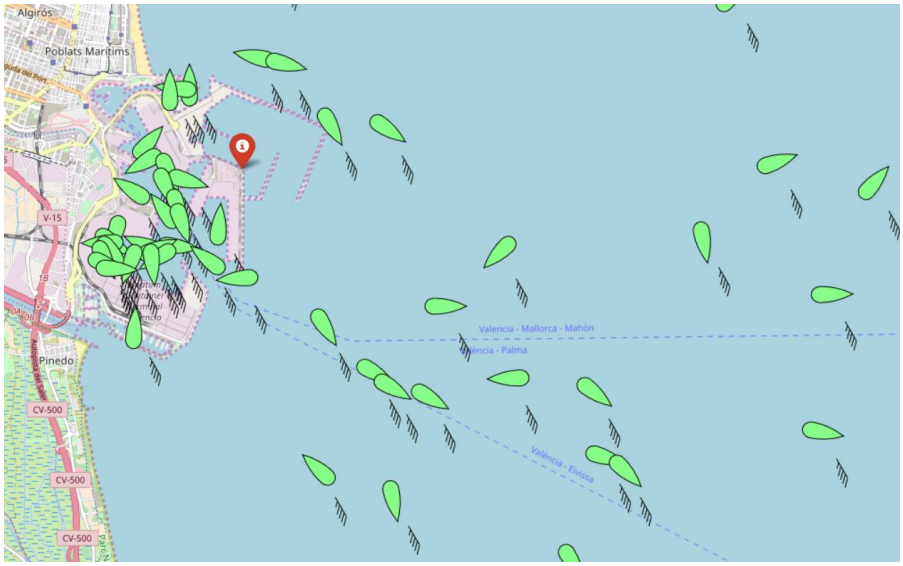

Figura 41. Mapa con mayor zoom para visualizar cada barco.

4.6 Threads

El último punto del programa está dedicado a la distribución en paralelo de procesos en la raspberry pi gestionado desde Python, esto fue implementado cuando surgió la necesidad de ejecutar varios procesos a la vez y explotar el máximo potencial que ofrece la raspberry, debido a que mientras realiza el proceso de escucha y recepción pueda decodificar, almacenar y representar.

Como se ha explicado en las especificaciones del hardware la raspberry dispone de cuatro núcleos, los cuales nos permitirán subdividir las tareas que realiza el programa, es decir dedicar un nucleo para cada una de las principales tareas. Los Threads o hilos permiten a la aplicación ejecutar multiples funciones, operaciones y ejecutar de forma paralela los subprocesos que puedan existir en una aplicación en un mismo espacio a la hora de procesar. El módulo utilizado para ello es el módulo threading.

Ejecutar varios hilos es similar a ejecutar varios programas diferentes al mismo tiempo, pero con algunas ventajas añadidas:

- Los hilos en ejecución de ocupan el mismo espacio que el hilo que se ejecuta como principal y por ello pueden compartir información mutuamente y no tener problemas a la hora de comunicarse entre sí.
- LA ejecución de un proceso mediante varios hilos requiere una menor capacidad de procesamiento y de memoria para ejecutar el mismo proceso pero de manera lineal.
- Permite simplificar el procesamiento requerido por las aplicaciones que necesitan ejecutar varias operaciones a la vez debido a que reduces en tareas más pequeñas las funciones que un programa debe hacer.

Por cada hilo existente se le asigna un puntero que sirve para realizar el seguimiento del proceso y de las instrucciones que se están ejecutando en cada momento. Además, la ejecución de un hilo se puede detener temporalmente o de manera indefinida. Normalmente, un proceso continua en ejecución cuando uno de los hilos lo sigue también, por lo tanto hasta cuando el

último hilo activo termina su función, el proceso también termina, liberando todos los recursos utilizados por ese hilo.

Un hilo o Thread (Figura 42) representa una operación que se ejecuta como subproceso independiente, es decir representa un hilo. Para crear un hilo se puede hacer de dos maneras: la primera, consiste en crear una función que se podrá invocar posteriormente, la cual ha sido la forma que se ha utilizado mayoritariamente en el programa, la segunda, consiste en crear mediante una subclase de Thread en la que se encontraran los métodos run e init().

Los objetos Thread (los hilos) tienen un parámetro target que permite definir que función se va a ejecutar. Una vez que los hilos están creados se inician con el método start(). Para asignar un nombre a los hilos se debe utilizar el método getName(), de esta manera se podrán referenciar posteriormente.

```
2250  def thread():
2251
2252      thread1 = threading.Thread(target = capturar_data)
2253      thread1.start()
```

Figura 42. Hilo creado para la captura de datos.

A los threads se les puede pasar parámetros que posteriormente pueden ser usados por las diferentes funciones del programa. Cualquier tipo de objeto puede ser pasado como parámetro a un thread.

En el ejemplo anterior, la aplicación espera antes de acabar a que todos sus threads se hayan completado. Algunas veces queremos lanzar un thread como un daemon lo que permite que una función secundaria se ejecute sin bloquear el hilo principal, por lo que en cualquier momento se podrá salir de la aplicación.

Esto es utilizado en el programa por su utilidad, debido a que en caso de que uno de los procesos falle o se desee cerrarlo, es decir permitirlo morir en mitad del proceso no supondrá una corrupción de datos. Por ejemplo, un proceso puede iniciar un hilo que haga algún tipo de subrutina que pueda ser utilizada para monitorizar los procesos.

Sin embargo para este proyecto es necesario que la recepción pare en el momento que el usuario desee para que tenga el control total, por lo que el hilo principal del programa puede finalizar aunque uno o más hilos hijos no hayan terminado su tarea; teniendo en cuenta que cuando finalice el hilo principal también lo harán estos hilos especiales llamados demonios

Para levantar un thread como daemon solo tenemos que invocar a su método setDaemon() pasándole el argumento True.

```
431      def StartThread(self):
432
433          """Arrancamos el hilo receptor"""
434
435          self.thread = threading.Thread(target=self.CapturaData)
436          self.thread.daemon = True
437          self.alive.set()
438          self.thread.start()
439
440      def StopThread(self):
441
442          """ Paramos el hilo receptor, esperamos hasta que termine"""
443
444          if self.thread is not None:
445              self.alive.clear()              # Limpiamos el evento alive  para el hilo
                 receptor
446              self.thread.join()             # Esperamos hasta que el hilo haya terminado
447              self.thread = None
448
449
450      def CapturaData(self):
```

Figura 43.Hilo creado para que siga a la escucha y recibiendo

Si ejecutáramos el ejemplo anterior (Figura 43), no recibiríamos el mensaje de salida del thread puesto que la aplicación saldría sin esperarlo. Por ello si un proceso se finaliza todos los demás lo harán también para ello se utilizan los daemon . Para esperar a que un thread daemon termine, debemos invocar explícitamente al método join().

Para saber si un thread está vivo, podemos utilizar el método isAlive() en él, que devolverá True o False según su estado.

```
self. thread = None
self.alive =threading.Event()
```

En conclusión los hilos permiten al programa ejecutar sus principales tareas paralelamente y de forma independiente por lo que se consigue que mientras el receptor sigue recibiendo y se decodifican las tramas, a su vez se están almacenando y representar el mapa sin tener que parar uno de los procesos. El problema realmente surge de cara al usuario puesto que en caso de que fuera una aplicación simplemente por consola, únicamente requeriríamos de una funcionalidad más lineal y el usuario por lo contrario necesita opciones que no tienen por qué ejecutarse con un patrón determinado.

Capítulo 5 Conclusiones y futuros desarrollos

La línea de trabajo de este TFG está orientada a la implementación de un receptor mediante el estándar AIS a partir de un prototipo hardware modular basado en raspberry pi..

Este trabajo se compone de una etapa previa de documentación sobre el estándar AIS estrictamente necesaria, donde se entra en detalle acerca de las características que debe cumplir este estándar y posteriormente una etapa de diseño hardware y software que satisfaga la necesidad de recabar los datos de los transmisores AIS.

En este proyecto se ha abordado el diseño de un receptor tipo AIS de bajo coste, podemos resumir

- Se ha conseguido integrar en un sistema de un coste inferior a los 70 euros, un receptor que en el mercado tiene un rango de precios que abarca desde los 300 el más simple y barato a más de 1000 euros.
- Se ha potenciado uno de los tantos usos que los nuevos ordenadores de placa y coste reducido ofrecen para desarrollar aplicaciones, en este caso a modo de receptor y decodificador con componentes eficientes y económicos
- Se ha desarrollado una aplicación sencilla y simple para la gestión de las tramas AIS en el entorno de Python con posibilidad de ampliación a aplicación web y desarrollo para un terminal completo de recepción de tramas AIS.
- Creación de mapas de tipo dinámico y representación gráfica de los datos recibidos por los barcos.
- Almacenamiento para un posterior tratamiento en una base de datos dinámica. Donde todos los barcos quedan registrados. Por lo tanto completamos el objetivo de la geolocalización y por consiguiente registro de los barcos, que en este caso por cercanía, en el puerto de Valencia, escalable a cualquier otra zona de tráfico marítimo denso.

5.2 Líneas futuras

La línea más clara que se abre sería la de continuar desarrollando la aplicación software puesto que existen hasta 27 tramas, por el momento, de las cuales solo se han tratado las cinco más importantes ya que el objetivo era la geolocalización de barcos, además se puede llegar a desarrollar una aplicación que funcione de forma casi simultánea a modo de GPS.

Otra posible línea de trabajo sería el análisis de todos los datos aportados por los barcos para optimizar la entrada de buques y barcos a un puerto, debido a la gran afluencia de tráfico marítimo, muchos barcos deben esperar antes de que se les puede dejar atracar debido a sus dimensiones, dársenas libres etc, lo que supone cuantiosos gastos mientras permanecen parados, por lo que sabiendo todos los datos de los barcos mediante el terminal se podría optimizar las entradas hasta evitar que cualquier barco de grandes dimensiones tuviera que esperar.

Se podría implementar un sistema de sincronización casi automática, es decir un mapa a tiempo real, aunque la decodificación y la recepción de tramas no se realiza de manera paralela, existe la posibilidad de una sincronización casi simultánea después de 1 minuto recibiendo tramas.

Por último existe la posibilidad de utilizar el terminal para emergencias marítimas debido a que entre las tramas que no se han tratado, existen varías dedicadas para estas situaciones, por lo que ofrece la posibilidad de desarrollar incluso una aplicación que exclusivamente sirva para notificar las emergencias marítimas.

5.3 Bibliografía

[1] Raspberry Pi 3 o Pi Zero : explote todo el potencial de su nano-ordenador Mocq, François. | Sánchez Conejo, Ángel María; Maestre Escalante, Antonio Jose.

[2] Learning PythonLutz, Mark.

[3] AIS : Automatic Identification System = Sistema de Identificación Automática : teoría y práctica : el sistema de navegación y seguridad del futuroHirche, Rüdiger | Madrid : Tutor, D.L. 2010.

[4] Python accelerators for high-performance computing,Marowka, Ami. The Journal of Supercomputing, 2018, Vol.74(4), pp.1449-1460

[5] Rec. ITU-R M.1371-1 1 RECOMMENDATION ITU-R M.1371-1 *Technical characteristics for a universal shipborne automatic identification system using time division multiple access in the VHF maritime mobile bandhttps://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.1371-1-200108-S!!PDF-E.pdf [Online]

[6] M.1371 : Características técnicas de un sistema de identificación automático mediante acceso múltiple por división en el tiempo en la banda de ondas métricas del servicio móvil marítimo. https://www.itu.int/rec/R-REC-M.1371-5-201402-I/es. [Online]

[7] https://docs.python.org/3/contents.html [Online]

[8] sqlite-amalgamation-3240000.C source code as an amalgamation, version 3.24.0. https://www.sqlite.org/ [Online].

[9] Análisis, diseño, desarrollo y promoción del sitio PROISER.COM y de las áreas de cliente y administración con PHP sobre Linux y SQLite [Recurso electrónico-CD-ROM]. Renovell Villar, Jaime | Pardo Gimilio, David; Universidad Politécnica de Valencia. Facultad de Informática | Valencia : Universidad Politécnica de Valencia, 2008.

[10] API reference documentation for the 4.0.2 release of wxPython Phoenix, built on 14 June 2018. https://wxpython.org.[Online].

[11] SQLite, Newman, Chris Author | Safari Tech Books Online.

[12] Python Data. Leaflet.js Maps. http://folium.readthedocs.io/en/latest/quickstart.html [Online].

[11] RASPBIAN STRETCH WITH DESKTOP.https://www.raspberrypi.org/downloads/raspbian/ [Online]

[11] AIS data sharing and vessel tracking by AISHub. http://www.aishub.net/[Online].