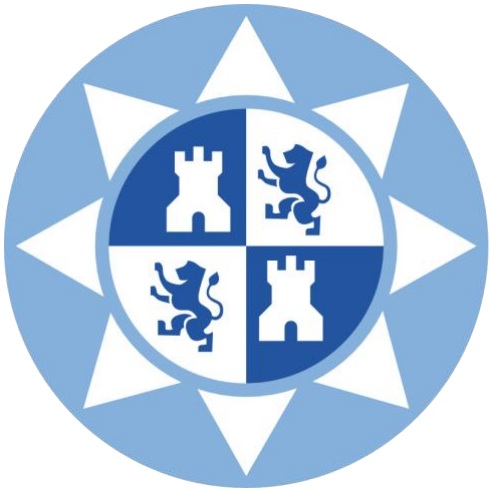
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE CARTAGENA



TRABAJO FINAL DE GRADO

IMPLEMENTACIÓN DE UNA HERRAMIENTA SOFTWARE PARA EL CONTROL Y SEGUIMIENTO REMOTO DE UN BARCO AUTÓNOMO

Autor: Mariano Fernández Ferrer

Directores: José María García-Pardo Molina

Antonio Mateo Aroca

Tipo de Proyecto:	Específico	**Curso Académico:**	2020-21
Director de Proyecto:	Molina García-Pardo, José María		
Departamento:	Tecnologías de la Inform. y las Comunic.		
Área conocimiento:	Teoría de la Señal y Comunicaciones		
Título en castellano:	Implementación de una herramienta software para el control y seguimiento remoto de un barco autónomo.		
Título en inglés:	Implementation of a software tool for remote control and monitoring of an autonomous boat		
Requisitos:	Estudiante de Telecomunicaciones		
Objetivos:	El objeto de este Trabajo de Fin de Grado es el desarrollo de un software de control que permitirá la comunicación entre una plataforma en tierra y un barco autónomo sin tripulantes, que permita tanto su control como su monitorización. Desde tierra se recibirán los datos a tiempo real captados por la embarcación, como por ejemplo su posición, velocidad, y otros proporcionados por los sensores. Dentro de este proyecto cabe diferenciar dos fases, una primera donde recibimos los datos y los representamos en una aplicación web programada con Python, y otra fase en la cual utilizamos el plotter de OpenCPN para visualizar remotamente donde se encuentra nuestro barco y además también poder darle órdenes para que se dirija a un lugar concreto.		
Resumen:	El proyecto consiste por tanto en el diseño y desarrollo de una aplicación para el control de un barco autónomo propulsado a motor eléctrico y/o con vela rígida, el cual deberá superar unas pruebas que se realizarán en el puerto de Cartagena para posteriormente en un futuro poder competir en The microtransat challenge. Este proyecto se enmarca dentro de un trabajo multidisciplinar donde participan las escuelas de Industriales, Teleco, Navales y Minas de la UPCT. En particular, la parte que aborda es la recopilación de los datos de la telemetría en una placa situada en el barco autónomo, que se transmiten vía radio y que posteriormente, se muestran a través de un software desarrollada específicamente para este propósito. Con objetivo de mejorar el diseño y funcionamiento del barco autónomo se va a llevar a cabo el proyecto de un sistema de telemetría. Se realizarán pruebas reales en el mar, mejorando todos los aspectos posibles como diseño de la aplicación, representaciones, portabilidad		
Fases:	1. Búsqueda de información e introducción al lenguaje de programación Python 2. Desarrollo de aplicación auxiliar con la cual vamos a realizar pruebas de comunicación. 3.Una vez establecida la comunicación, completamos el software con la parte que extrae de la trama de NMEA la información deseada. 4.Desarrollo de aplicación para la representación de los datos recibidos. 5. Implementación de mejoras visuales y funcionalidades en la aplicación para la versión final. 6. Pruebas y diagnóstico de nuestro sistema.		
Bibliografía:	Python: https://www.python.org/ IDLE: https://docs.python.org/3/library/idle.html OpenCPN: https://opencpn.org/		

Codirección del TFE	
[22990408]	MATEO AROCA, ANTONIO

AGRADECIMIENTOS

Con este trabajo fin de grado pongo un punto y aparte en mi formación universitaria, el cual me gustaría concluir agradeciendo en primer lugar a mis padres, por el apoyo tanto emocional como económico sin el que hubiera sido imposible cerrar esta etapa.

Por otra parte, me gustaría agradecer a mi tutor José María Molina por la ayuda, a mis compañeros José Carlos Urrea Celdrán y Miguel Ángel Escobar con los que he trabajado durante estos meses de proyecto, y al grupo VNAS en general.

ÍNDICE

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

1.1 Motivación

Con este trabajo lo que se pretende es colaborar en el proyecto con el cual se pretende participar en un futuro en “The Microtransat Challenge”. El Mircrotransat Challenge es una competición para diseñar y construir un velero autónomo de menos de 4 metros de eslora, capaz de cruzar el océano Atlántico sin interacción humana [1].

Previamente a dicha competición, se deberán superar ciertas pruebas en el puerto de Cartagena, para comprobar la flotabilidad de la embarcación en entornos reales y ver si es capaz de superar las adversidades que podemos encontrar en mar abierto. Además, comprobar si los motores son capaces de mover con la suficiente solvencia la embarcación y probar los sensores que nos informan de los parámetros del barco cuando esté navegando.

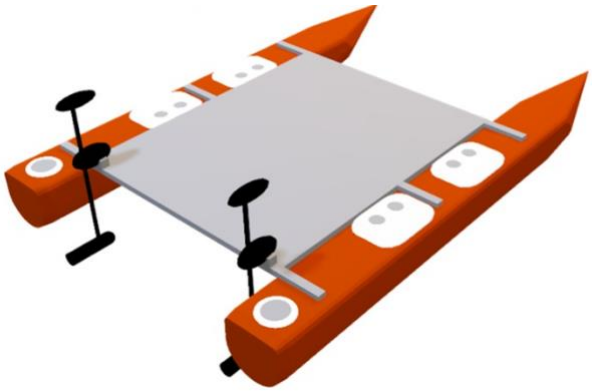


Figura 1 Estructura Catamarán [2]

El proyecto VNAS, Vehículo de Navegación Autónoma en Superficie, consiste en un trabajo multidisciplinar en el que se desarrollan diferentes tareas tales como la creación del casco del catamarán, el control de los motores, el montaje de los sensores y la parte en la que se encuentra este proyecto, la recepción y control de parámetros de la embarcación en superficie.

Figura 2 Logo VNAS

1.2 Objetivos

El objetivo de este trabajo final de grado consiste en desarrollar un software en el lenguaje de programación Python que sea capaz de leer un puerto y recibir los datos que se envían a dicho puerto desde los sensores y poder representarlos en tiempo real para así poder examinar el comportamiento de la embarcación.

Como punto de partida, realizamos un estudio profundo del proyecto VAMAR con el fin de adaptarlo a nuestras necesidades, así como familiarizarnos con la tecnología que se utiliza en el proyecto MotoStudent. El punto de partida de este proyecto viene dado por el proyecto realizado por dos compañeros en 2014 y continuado por otros compañeros sucesivamente.

1.3 Fases del proyecto

Este proyecto se ha llevado a cabo durante aproximadamente 10 meses, desde que se empezaron las primeras reuniones con el resto de compañeros y profesores que conforman el equipo de proyecto.

La primera fase de este proyecto fue la lectura de los trabajos de años anteriores para empezar a conocer las diferentes tecnologías empleadas y conoces un poco más de lo que trata el proyecto VNAS.

La segunda fase se llevó a cabo junto a mi compañero Miguel Ángel Escobar, donde estuvimos en contacto con los sensores que se iban a establecer en el barco, viendo para que servía cada uno y entendiendo el esquema de conexiones. Además, también nos desplazamos a la escuela de navales donde se encontraba la embarcación, para ver la disposición de los sensores y de las baterías en la misma, y la colocación de la caja estanca donde se iba a albergar la Raspberry para que no sufriera el desgaste de la humedad.

En la tercera fase se llevó a cabo el desarrollo del software encargado de la recepción de la información en tiempo real, así como la aplicación encargada de representar dichos datos. Para ello, se tuvo que realizar un estudio de los lenguajes de programación necesarios para dicho objeto.

La última fase del proyecto consistió en la unión de cada una de las anteriores partes para realizar las pruebas oportunas: correcto funcionamiento de los componentes en la embarcación, comunicación bidireccional, transmisión y recepción de información entre el barco y el cliente que solicita la información, y la representación de la información recibida en la aplicación diseñada para dicho fin.

1.4 Estructura de la memoria

La memoria de este proyecto se puede estructurar de la siguiente forma:

1) En primer lugar, se habla de la motivación que ha llevado a realizar este proyecto y los objetivos que se quieren conseguir.

2) En segundo lugar se desarrolla el concepto de telemetría y su evolución y la importancia que tiene en este proyecto.

3) En esta tercera parte, se describen los instrumentos instalados en la embarcación y los cuales proporcionan la información que obtenemos en la trama recibida por el puerto y la cual se representa en nuestra aplicación.

4) La cuarta parte describe los protocolos de comunicación y estándares en los que se transmiten los datos de la telemetría.

5) En el quinto apartado introducimos el lenguaje de programación Python con el que hemos desarrollado nuestra aplicación y la aplicación en concreto.

6) En esta última parte se sacan conclusiones acerca de los resultados obtenidos y sobre el proyecto en general, además de realizar proposiciones para líneas futuras.

2. ESTADO DEL ARTE

2.1 Introducción a la telemetría

La telemetría es el método de acceder de manera remota a variables o magnitudes físicas. Este concepto está muy ligado al de telecontrol por lo que no es extraño que los equipos de telemetría incorporen control a distancia. La palabra telemetría procede de las palabras griegas tele, que quiere decir a distancia, y la palabra "metron", que quiere decir medida. [3]

Las principales tecnologías usadas para la telemetría son Radio Frecuencia, redes GPRS/GSM e Internet. La preferencia por alguna de estas va a depender de las condiciones propias del proyecto: ¿hay conexión a Internet en la zona?, ¿hay una recepción de red celular suficiente para la comunicación? En el caso que ninguna de las opciones anteriores se pueda implementar se puede pensar en una solución con radiofrecuencia.[3]

Un sistema de telemetría está formado normalmente por tres partes, un dispositivo de entrada, el medio de transmisión y dispositivos de procesamiento y visualización de la señal.

Dispositivo de entrada: lo conforman sensores que son capaces de recoger de la información de la magnitudes que se encuentran bajo estudio y un transductor que se encarga de adaptar dichas señales generadas por los sensores al medio de transmisión utilizado.

Medio de transmisión: puede ser cableado o inalámbrico, dentro de los cableados podemos encontrar desde un cable de pares utilizado para la telefonía hasta fibra óptica. Dentro de la comunicaciones inalámbricas, de las cuales haremos uso en nuestro proyecto debido a que, por las condiciones en las que se desarrolla el proyecto hemos decidido que es lo más óptimo, destacamos la tecnología UMTS (3G) o WI-FI. En proyectos realizados por otros compañeros, se llegó a la conclusión que lo que aporta mayor fiabilidad es usar 3G.

Dispositivo de visualización de datos: son la parte principal de este proyecto. Son los encargados de la recepción y procesado de los datos generados por los sensores para su posterior almacenamiento o en nuestro caso visualización en tiempo real.
Para esta parte hemos desarrollado un software en Python que lee la información de un puerto y la representa en tiempo real.

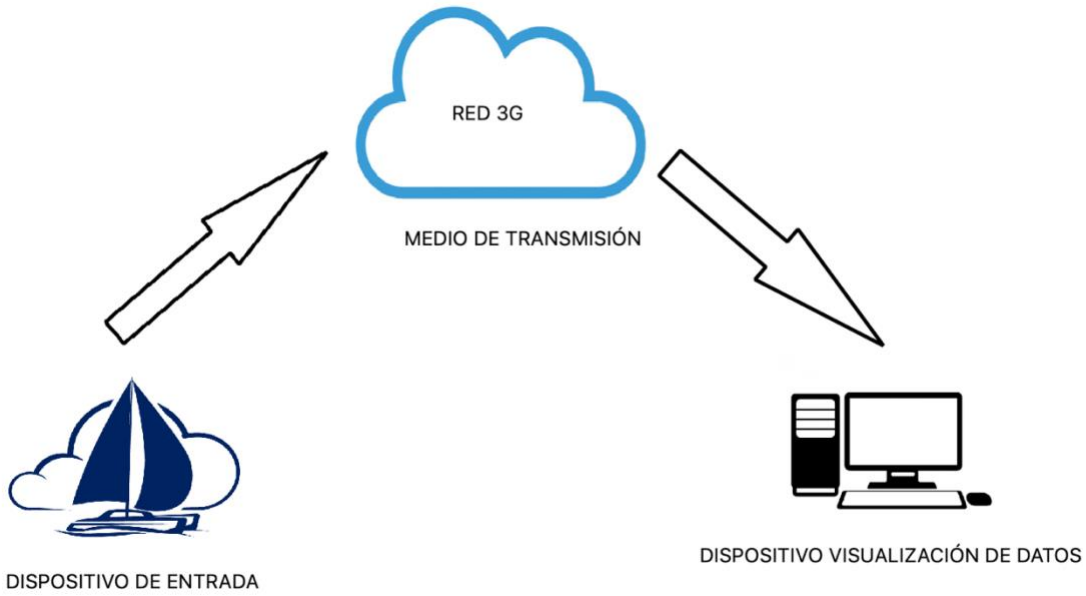


Figura 3 Sistema de telemetría

2.2 Comunicaciones Marítimas

Cuando un barco navega por mares u océanos, a varios kilómetros de tierra, se encuentra totalmente fuera de alcance de poder hacer uso de las redes terrenales. Por esto, es necesario hacer uso de sistemas de comunicaciones alternativas que proporcionen la seguridad y soporte para desarrollar con fiabilidad sus travesías.

2.2.1 Sistema de identificación automática

El sistema de identificación automática (AIS) es un sistema de información que emite datos captados a bordo y que, además, permite obtener los datos de otras embarcaciones que dispongan de él y que lo estén utilizando. [4]

El objetivo fundamental del **sistema AIS** es permitir a los buques comunicar su posición y otras informaciones relevantes para que otros buques o estaciones puedan conocerla y evitar colisiones. [4]

El sistema AIS opera a través de la banda marítima de VHF. Asimismo, permite ver en una pantalla la posición de los barcos de nuestro entorno y obtener información como su velocidad, rumbo o estado actual de navegación, pero siempre y cuando los otros barcos también transmitan información vía AIS. [4]

En consecuencia, el AIS permite ver datos instantáneos sobre los barcos que se muestran en la pantalla, pero no permite detectar objetos como boyas o la costa, por lo que no sustituye al radar. Además, el radar detecta un blanco aunque no emita, sin embargo, como acabamos de ver, esto no ocurre con el AIS. [4]

En definitiva, el sistema AIS nos ayuda a tener un mayor control sobre la seguridad del **tráfico marítimo**, a la par que facilita a los equipos de rescate la localización de una embarcación en peligro. Podemos decir que el AIS es una caja negra en cuya memoria se almacenan datos relevantes sobre la **navegación marítima** que contribuyen a clarificar la causa de determinados problemas. [4]

Figura 4 Sistema de identificación automática [3]

2.2.2 Radar

El radar es un sistema que usa ondas electromagnéticas para medir distancias, altitudes, direcciones y velocidades de objetos estáticos o móviles como aeronaves, barcos, vehículos motorizados, formaciones meteorológicas y el propio terreno.[5]

Su funcionamiento se basa en emitir un impulso de radio, que se refleja en el objetivo y se recibe típicamente en la misma posición del emisor. A partir de este «eco» se puede extraer gran cantidad de información. El uso de ondas electromagnéticas con diversas longitudes de onda permite detectar objetos más allá del rango de otro tipo de emisiones (luz visible, sonido, etc.)[5]

El **radar náutico** es uno de los equipos más importantes de un barco, considerado por algunos marinos como la ayuda más valiosa para la navegación. El radar náutico se utiliza generalmente para localizar objetos situados en nuestros alrededores, ya sean otros barcos, contenedores, o cualquier otro tipo de objeto contra el que pudiéramos colisionar o impactar.[5]

Figura 5 Radar Náutico [5]

Los componentes básicos de un radar son: un transmisor de radio de alta frecuencia, un receptor, una antena de radar y un monitor o pantalla.

El radar náutico funciona emitiendo una frecuencia de microondas o de radio de alta intensidad y frecuencia que detecta la energía que regresa tras chocar y rebotar contra un objeto. A partir de este “eco” que se nos muestra reflejado en una pantalla o monitor, se puede extraer mucha información y saber dónde está el objeto en cuestión y la distancia a la que se encuentra. La mayoría de los radares marinos operan en la banda X, que es la que proporciona mayor resolución de detección.

Además, el radar de una embarcación también puede detectar y rastrear tormentas durante la noche.[5]

En definitiva, el radar náutico es una gran herramienta para la navegación, así como para la prevención de colisiones. Asimismo, la instalación de un radar en la embarcación te proporcionará tranquilidad cuando se navegue bajo condiciones adversas, y en los momentos de buena visibilidad, podemos aprovechar para ampliar nuestra competencia en su utilización para estar preparados en momentos de dificultad. [5]

2.3 Embarcaciones autónomas

Durante toda la historia, numerosas embarcaciones han navegado a lo largo de los océanos del planeta con diversos fines, desde expediciones que han descubierto partes insólitas de nuestro planeta y que han enriquecido nuestra cultura, hasta otras que no pudieron llegar a su destino y se quedaron en el intento. Por esto, los barcos han ido evolucionando hasta llegar a las naves sin tripulación o con motores eléctricos, lo que hace que se puedan ahorrar costes al no contar con tripulación, o se reduzca la contaminación de los océanos al prescindir de combustibles fósiles.

2.3.1 Wave Glider

El Wave Glider es una embarcación diseñada para capturar el poder e información de la que está lleno el mundo marino. Con los últimos avances en recolección de energía y propulsión, combinados con una arquitectura de carga útil y sensores, Wave Glider es una plataforma de recolección de datos persistente.[6]

Este barco realiza una conversión del movimiento ondulatorio en propulsión, la energía de las olas es mayor en la superficie del agua, disminuyendo rápidamente al aumentar la profundidad. La arquitectura única de dos partes del Wave Glider aprovecha esta diferencia de energía para proporcionar propulsión hacia adelante.[6]

También ofrece un sistema de propulsión adicional que utiliza energía solar almacenada. El empuje direccional adicional aumenta la movilidad y la precisión y ayuda a navegar por las condiciones desafiantes del océano (depresión, corrientes altas y huracanes / ciclones), o adaptarse a los cambios de misión. El sistema de energía solar también recarga las baterías que alimentan los sensores.[6]

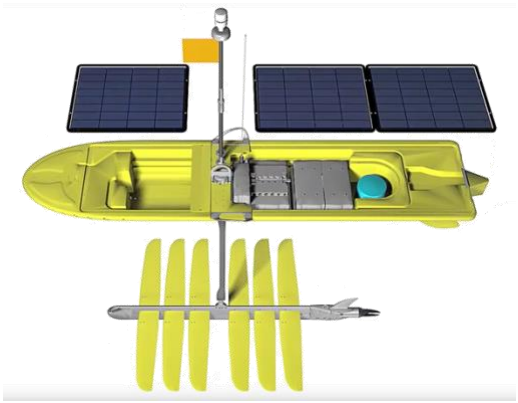


Figura 6 estructura Wave Glider [6]

Algunas de las aplicaciones de los Wave Gliders son:

Guerra antisubmarina: albergan cargas útiles y sensores de remolque para proporcionar conectividad submarina, abordar los desafíos de detección inicial y permitir pistas de los activos aéreos.[6]

En el ejercicio Unmanned Warrior de la Royal Navy británica, Liquid Robotics y Boeing utilizaron una red de USV persistentes para detectar, informar y rastrear un submarino en vivo.[6]

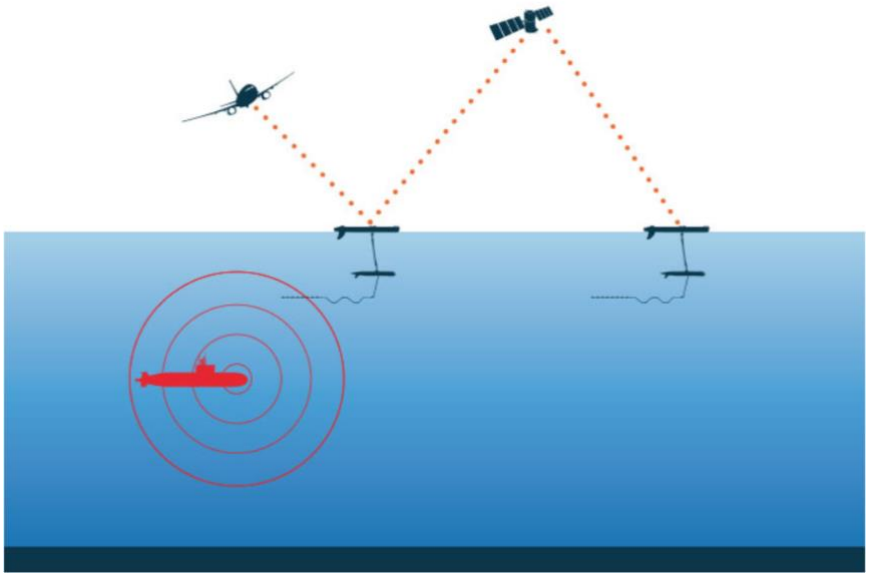


Figura 7 Guerra Antisubmarina Wave Glider [5]

Detección de embarcaciones en superficie: son una plataforma móvil de bajo costo para vigilancia acústica persistente 24 × 7, por lo que puede detectar embarcaciones que ingresan u operan en un área específica.[6]

Durante una misión para patrullar las aguas alrededor de las islas Pitcairn, el Wave Glider interceptó y recopiló con éxito datos sobre tres embarcaciones cuyas firmas AIS no estaban disponibles.[6]

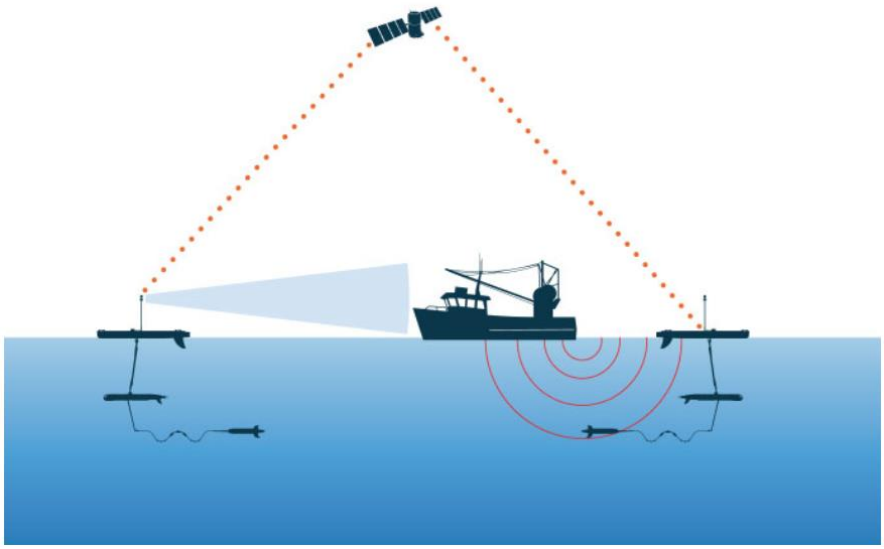


Figura 8 Detección embarcaciones en superficie [5]

Vigilancia y reconocimiento de inteligencia: son plataformas móviles de baja observación que permiten la vigilancia sobre el horizonte con cargas útiles tanto superficiales como subterráneas.[6]

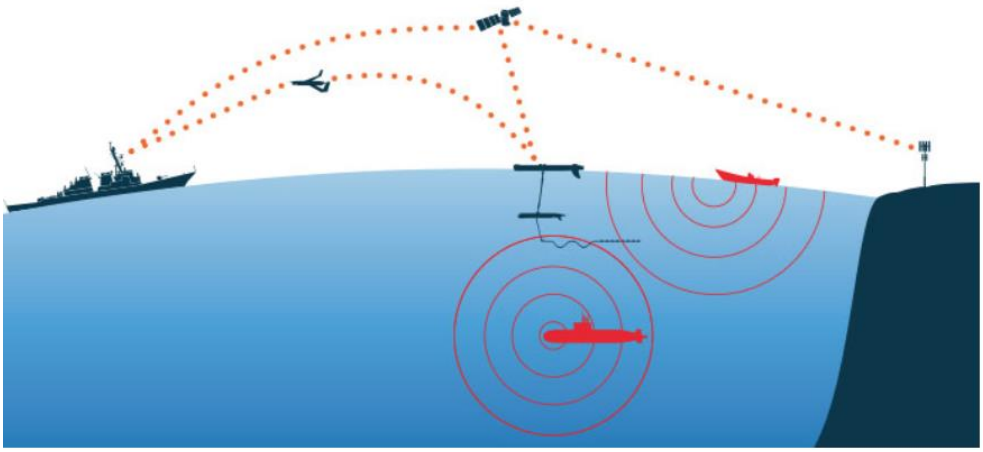


Figura 9 Vigilancia y reconocimiento de inteligencia [6]

Gateway de comunicaciones: las ondas de radio no viajan bien bajo el agua. Y la acústica subacuática requiere un relé en la superficie. Los Wave Gliders proporcionan un enlace esencial para conectar el lecho marino con el espacio en casi cualquier lugar del océano. [6]

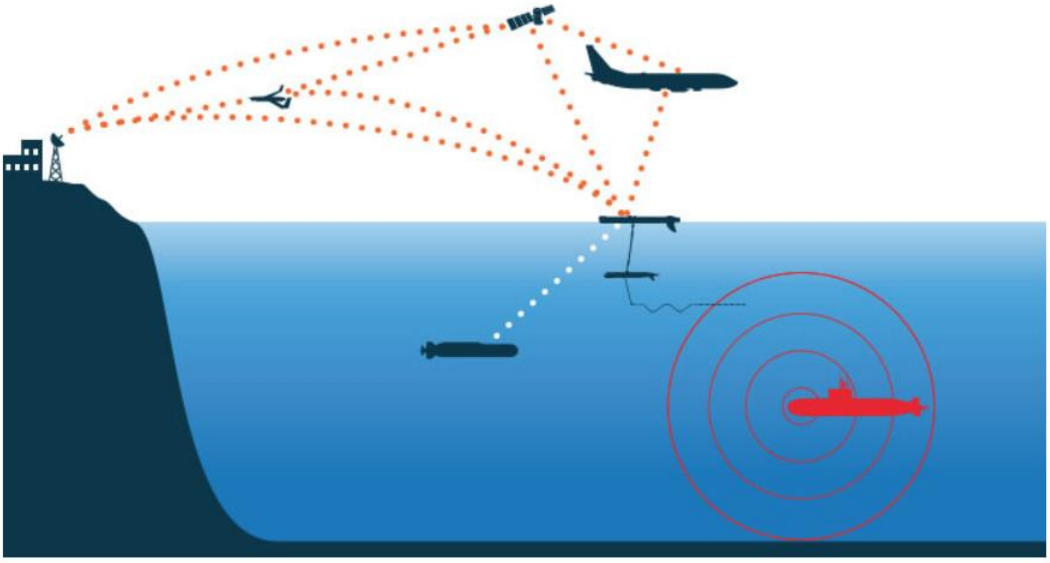


Figura 10 Gateway de comunicaciones [6]

2.3.2 Saildrone

Saildrone es una embarcación autónoma de tamaño pequeño, capaz de recopilar datos in situ relacionados con el océano a través de vehículos sin tripulación, por encima y por debajo de la superficie del mar. Las organizaciones gubernamentales, científicas y comerciales de todo el mundo confían en Saildrone para proporcionar la información crítica que necesitan, cuando la necesitan. [7]

El sistema de propulsión patentado de Saildrone consta de un ala alta y dura, un larguero longitudinal y una cola vertical. Una pestaña de ajuste en la cola ajusta el ángulo del ala al viento, de manera similar a la forma en que una pestaña de ajuste de ascensor controla el cabeceo de un avión. Este sistema de propulsión permite una duración de la misión de hasta 12 meses, sin necesidad de regresar a tierra para mantenimiento o repostaje, con el único requisito de realizar un mantenimiento anual que garantice una calidad de datos óptima. [7]

Los Saildrone están bajo la supervisión constante de un piloto humano a través del satélite y navegan de forma autónoma de un punto de referencia a otro prescrito, teniendo en cuenta el viento y las corrientes, mientras se mantienen dentro de un corredor de seguridad definido por el usuario. [7]

Para garantizar aún más una operación segura, cada USV está equipado con un transceptor del sistema de identificación automática (AIS), luces de navegación, reflector de radar, colores de ala de alta visibilidad y cuatro cámaras a bordo.[7]

Figura 11 Saildrone [7]

Los saildrones han participado en varios misiones con resultados satisfactorios:

Tiburones blancos

Se trata de una misión multidisciplinaria para comprender por qué los tiburones blancos se congregan regularmente en una parte aparentemente inhóspita del océano [7].

Los satélites han revelado que los tiburones blancos marcados en las aguas costeras de California migran anualmente a una región llamada White Shark Café, un área prácticamente desconocida situada en el giro subtropical del Pacífico norte (NPSG). En 2018, dos saildrones rastrearon tiburones blancos marcados en la región con el objetivo de arrojar algo de luz sobre una pregunta de larga data sobre sus hábitos migratorios. La misión se llevó a cabo en asociación con la Dra. Barbara Block de la Universidad de Stanford y el Schmidt Ocean Institute e involucró a un equipo multidisciplinario de oceanógrafos, ingenieros, ecólogos marinos y biólogos moleculares de cuatro instituciones [7].

Por primera vez, los datos de oceanografía física y química recopilados por saildrone se combinaron con datos biológicos y de comportamiento para ayudar a los científicos a comprender el papel que juega el White Shark Café en el ciclo de vida de esta especie [7].

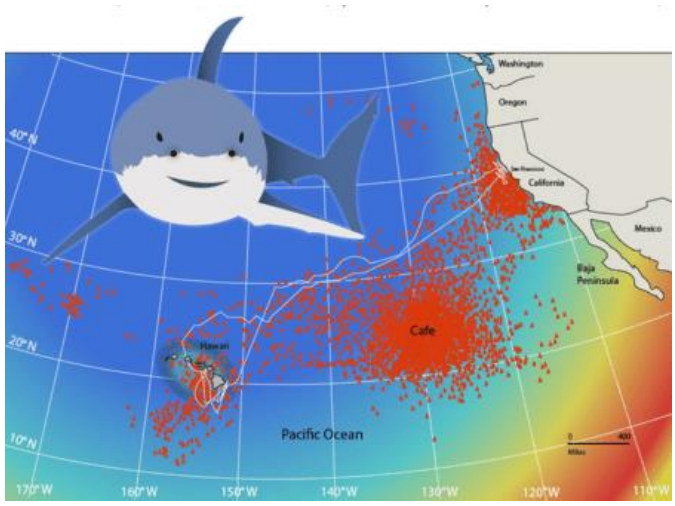


Figura 12 Datos obtenidos por el saildrone sobre el tiburón blanco [7]

Círculo Polar Ártico

La quinta misión anual del Ártico llevó una flota de saildrones a una nueva frontera, el borde del hielo del Ártico, para mejorar la predicción del hielo marino y el desarrollo de algoritmos satelitales [7].

Una flota de saildrones pasó 147 días en el mar de Chukchi navegando intencionalmente hacia y hacia el hielo marino del Ártico para proporcionar observaciones in situ para el desarrollo de algoritmos satelitales como parte del estudio de temperatura de la superficie del mar mejorada con múltiples sensores del Ártico. Un sexto vehículo realizó una encuesta de pesca en el mar de Bering. La flota viajó más de 36.000 millas náuticas y estableció un nuevo récord de latitud norte de Saildrone de 75,49 ° N [7].

Las comparaciones en tiempo real de los datos de Saildrone con modelos numéricos revelaron que los flujos de calor neto de superficie en el Ártico varían sustancialmente, las predicciones numéricas de corto alcance pueden ser muy buenas para ciertas variables pero no para otras, y se desvían rápidamente de las observaciones in situ de Saildrone como el aumenta el rango de predicción [7].

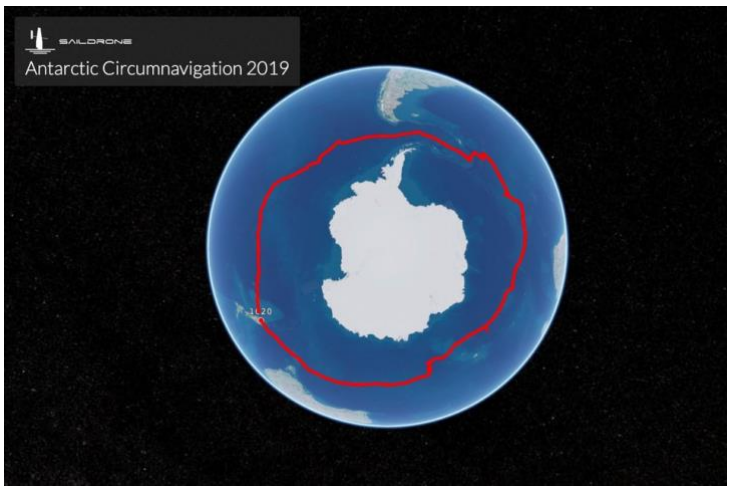


Figura 13 Circunvalación Saildrone Antártida [7].

Temperatura baja California

Un crucero de 60 días para estudiar la temperatura de la superficie del mar y los flujos de calor aire-mar a lo largo de las regiones frontales dinámicas frente a la costa de California [7].

El crucero de 60 días de Baja Campaign fue la primera vez que se utilizó la plataforma Saildrone USV en estudios de comparación con un satélite. El saildrone navegó desde San Francisco a lo largo de la costa de Estados Unidos / México hacia la isla Guadalupe a través del sistema de corrientes de California, que es muy variable. El objetivo de esta misión era realizar una comparación de tres vías entre los datos recopilados de un Saildrone USV, el satélite y las boyas amarradas y los buques de investigación a lo largo de la vía en una región conocida por su compleja dinámica aire-tierra-mar [7].

Los datos recopilados de esta misión sirven para avanzar en la investigación sobre la dinámica de las afloramientos, los flujos de aire y mar y los frentes oceánicos en una parte económicamente importante del océano [7].

Figura 14 Saildrone cruzando la bahía de San Francisco [7]

3. Proyecto VNAS

3.1 Descripción proyecto VNAS

VNAS es un proyecto llevado a cabo conjuntamente por alumnos y profesores de la Universidad Politécnica de Cartagena y de la Universidad de Murcia. Dentro de la Universidad politécnica de Cartagena, han participado diferentes departamentos de la Escuela Técnica Superior de Ingeniería Naval y Oceánica, Escuela Técnica superior de Ingeniería industrial y la Escuela Técnica Superior de Ingeniería de Telecomunicación, dentro de la cual se engloba este proyecto.

El objetivo de los diferentes componentes del proyecto es realizar una parte del mismo aportando los conocimientos de nuestro sector, para poder poner en conjunto todas las partes y satisfacer las necesidades del proyecto y que salga adelante de la mejor manea posible. En este caso, como miembros de la Escuela de Telecomunicación, estamos encargados de tantos la configuración de la Raspberry pi encargada de la gestión de las rutas y de los motores, como de la monitorización y el control del barco a distancia.

Las diferentes tareas que se han llevado a cabo o que están por terminarse son las siguientes:

-Construcción del catamarán: ha sido realizada por la escuela de navales. El objetivo de esta es conseguir una embarcación capaz de transportar los diferentes equipos e instrumentación necesarios con la suficiente flotabilidad y aerodinámica.

Figura 15 Construcción del catamarán

-Desarrollo de un sistema de control de motores: esta tarea ha sido asignada a mi compañero Miguel Ángel Escobar. El objetivo de esta tarea es el desarrollo de un sistema de control de los motores por medio de Arduino con objeto de poder garantizar su movimiento a partir de la información recibida de manera remota y manual por el usuario.

Figura 16 Arduino [8]

Software para el piloto automático: con dicho software, lo que se pretende es que la Raspberry sea capaz de dar ordenes a los motores para que se llegue al destino de manera totalmente autónoma. Ha sido realizado por José Carlos Urrea.

Instalación de los instrumentos necesarios para la telemetría: en esta parte lo que se pretende es instalar todos los componentes y la circuitería asociada que cada uno de ellos llevan, para poder proporcionar los datos del estado del barco, como el compás, el GPS, anemómetro etc.

Software para el seguimiento remoto del barco: en esta parte de encuentra el proyecto. El objetivo es desarrollar una herramienta que sea capaz de recibir los datos en tiempo real de la embarcación para así poder realizar un seguimiento remoto de ésta. Dicho software ha sido implementado en Python lo que hace que sea totalmente portable y fiable en los diferentes sistemas operativos que se quiera ejecutar.

3.2 ESQUEMA GLOBAL DEL SISTEMA

En la figura 17 podemos ver un esquema un plano completo del catamarán. Por un lado, se aprecia que las baterías que alimentarán el sistema, están colocadas en cada uno de los cascos, a los que se puede acceder con unas trampillas herméticas para que no entre agua. Por otra parte, en el centro de la superficie del barco se colocará una caja estanca para colocar en su interior los instrumentos de medida. Por último, en la popa del barco, anclados a la superficie de éste, irán los dos motores eléctricos que deben propulsar el barco.

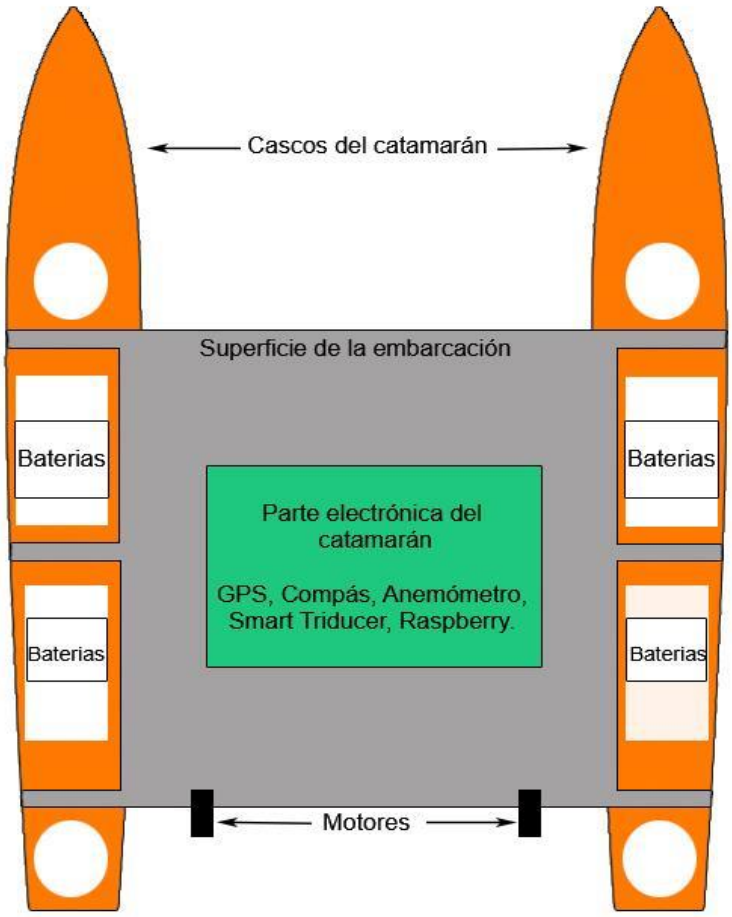


Figura 17 Esquema global del sistema [9]

Por otro lado, en la figura 18 tenemos el esquema del sistema electrónico del barco, que fue diseñado por dos compañeros años anteriores, Hassan Bahari y Alejandro González. Inicialmente, se instaló una raspberry con un software que posteriormente fue sustituida por otra raspberry y otro software que mejoraba las prestaciones del sistema.

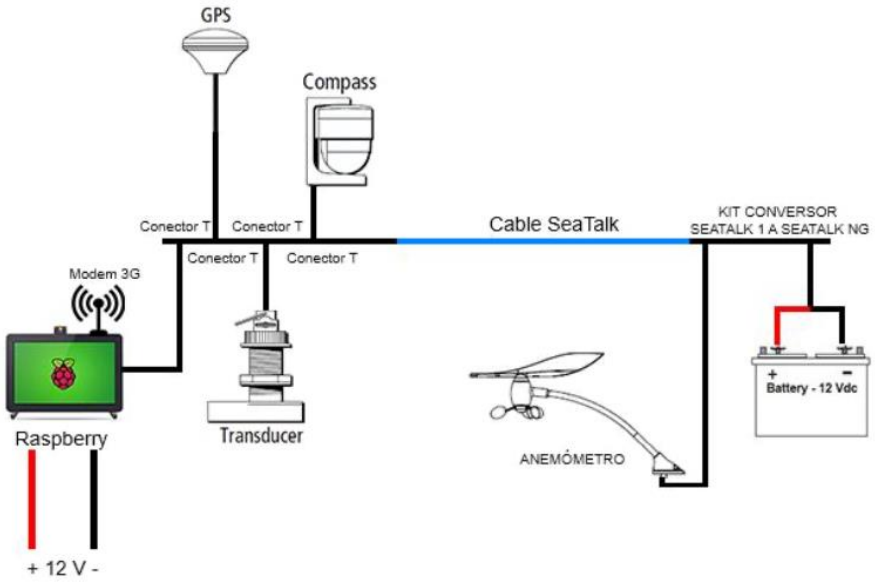


Figura 18 Esquema parte electrónica [9]

Se puede apreciar que el sistema consta de 3 sensores: sensor de posicionamiento GPS, un compás para saber la orientación e inclinación, y un transducer que nos proporciona la velocidad, temperatura y profundidad de la embarcación. Estos sensores se conectan mediante conectores tipo T a la raspberry, por un lado, y a un cable SeaTalk, cuya función es conectarse a las baterías y al anemómetro.

3.3 SENSORES, CABLES E INSTRUMENTACIÓN

- Puerta de enlace NGW-1 NMEA 2000: proporciona una forma sencilla de vincular las redes de datos de un barco y convierte los datos NMEA 0183 en datos NMEA 2000 y viceversa. [10]

Figura 19 Puerta de enlace NGW-1 NMEA 2000

- Raymarine SeaTalk 1 to SeaTalk NG converter: permite que determinados dispositivos SeaTalk 1 interactúen y se comuniquen en la red de datos Raymarine SeaTalk NG. [11]

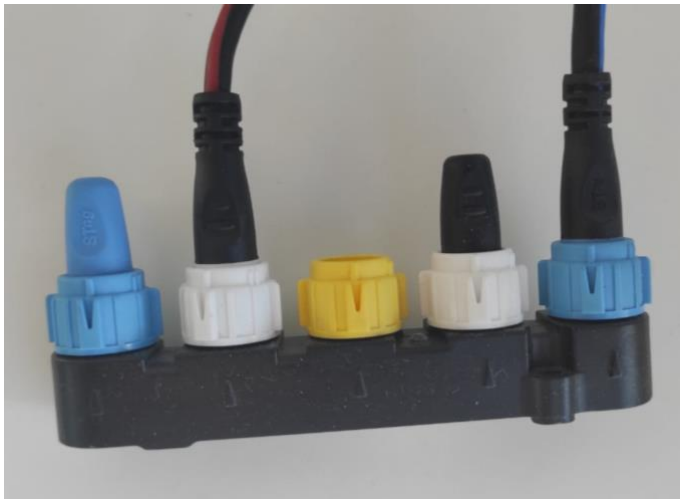


Figura 20 Raymarine SeaTalk 1 to SeaTalk NG converter

- Antena GPS GARMIN 19X 10Hz NMEA 2000: La antena/receptor de posición de alta precisión GPS 19x NMEA 2000 proporciona frecuencias de actualizaciones de 10 Hz para los datos de posición, velocidad y tiempo. Ofrece una recepción de alta sensibilidad y adquisición de la posición mejorada para la familia de Garmin de pantallas multifunción (MFD), pantallas de instrumentos náuticos y pilotos automáticos. [12]

- Compás AIRMAR H2183 NMEA 2000: es una brújula de estado solido de tres ejes combinada, que nos proporciona datos de rumbo e inclinación del barco.

Figura 22 Compás AIRMAR (vista lateral)

Figura 21 Compás AIRMAR (vista superior)

- Conector tipo T multipuerto NMEA2000:

Figura 23 Conector tipo T multipuerto NMEA2000

- Smart TRIDUCER AIRMAR DTS800: es un Smart TRIDUCER Multisensor que ofrece funciones de profundidad, velocidad y temperatura en un montaje pasacascos compacto. El sensor retráctil de bajo perfil calcula datos precisos de profundidad, velocidad y temperatura y los envía a cualquier pantalla o red NMEA 0183 o NMEA 2000 [13].

Figura 24 Smart Triducer Airmar

3.4 Software OPENCPN

OpenCPN es un software trazador de gráficos, que permite crear rutas de navegación, y ver la trayectoria que siguen los barcos en tiempo real. Para el correcto funcionamiento del sistema, realizamos las siguientes configuraciones en el mismo:

3.4.1 Creación de la ruta

Creamos la ruta que queramos que siga la embarcación. Al hacer click en nueva ruta, aparece un lápiz con el que podemos dibujar el trazado deseado. Cada click que hagamos es un nuevo waypoint. Es importante que coloquemos el primer waypoint sobre la posición inicial proporcionada por el GPS.

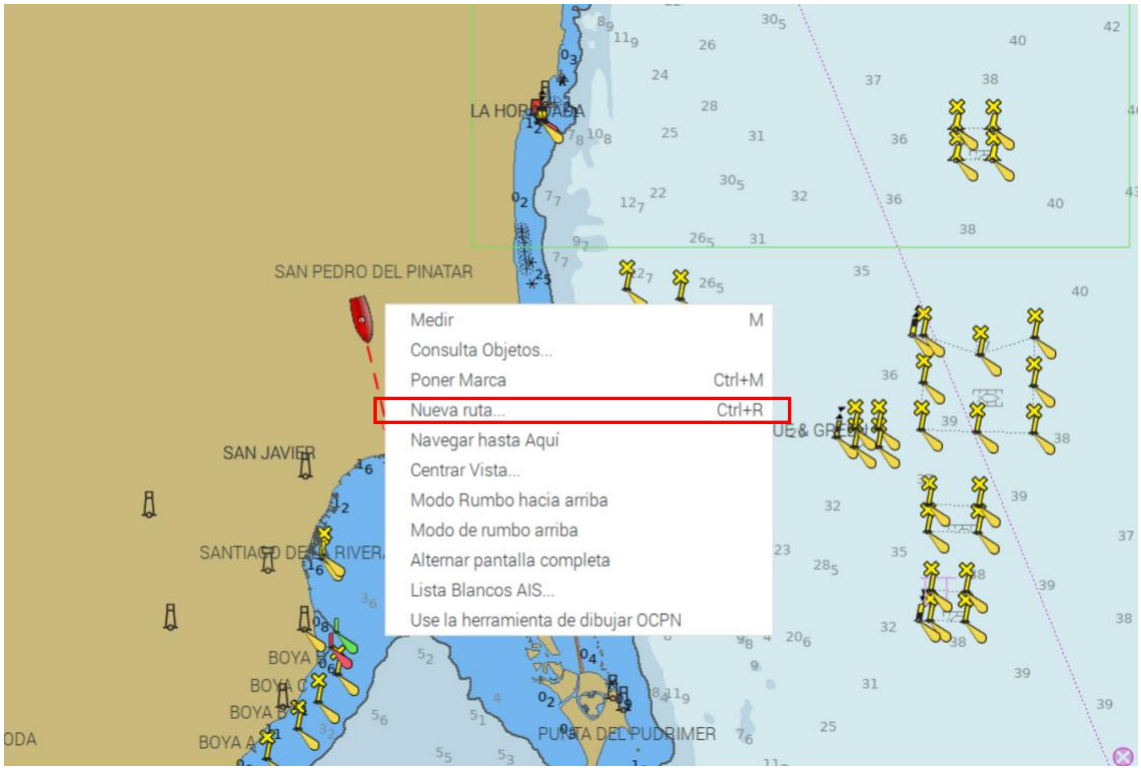


Figura 25 Creación ruta

3.4.2 Activación de la ruta

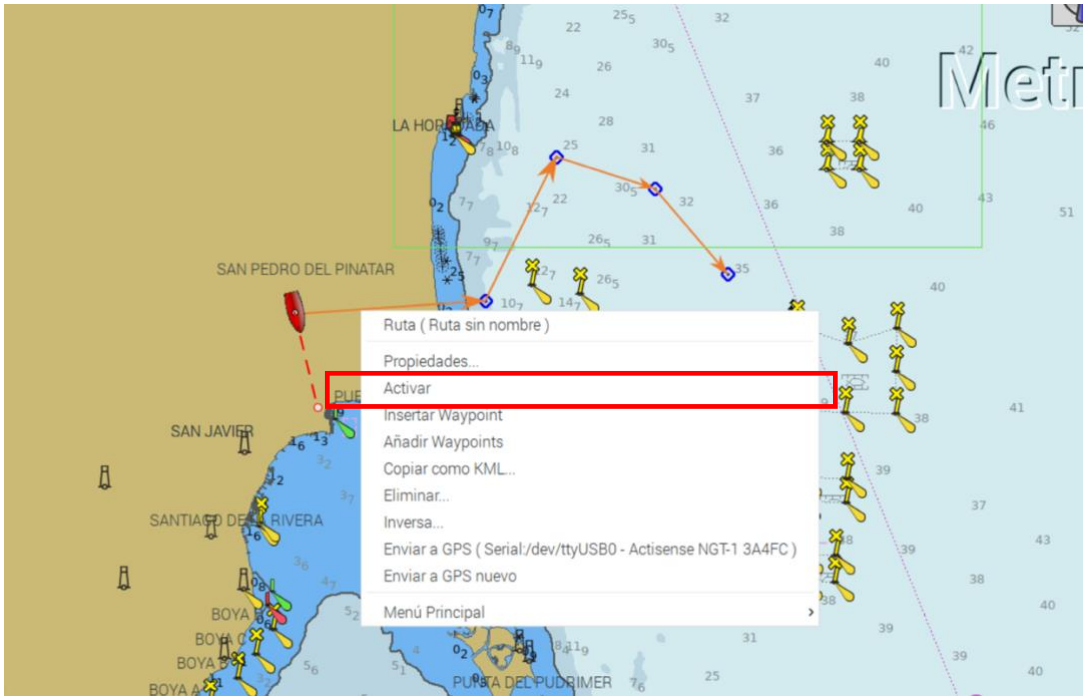


Figura 26 Activación ruta

3.4.3 Conexión de salida de datos

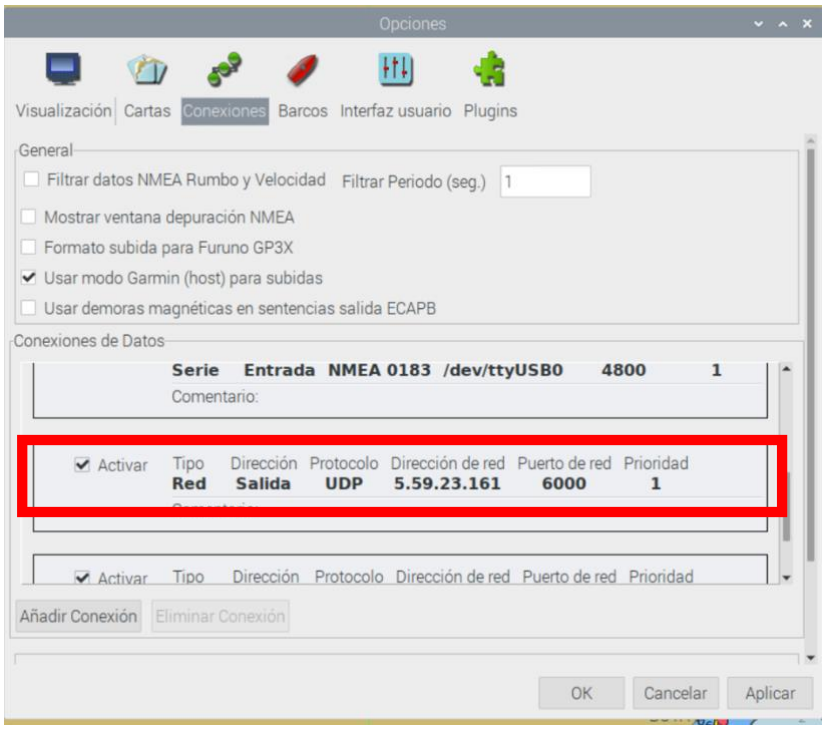


Figura 27 Conexión salida de datos

En el apartado de conexiones, creamos una conexión de tipo salida con la dirección IP y el puerto del equipo que vamos a utilizar para enviar los datos.

3.4.4 Filtro de sentencias NMEA

En el filtro de salida, seleccionamos la sentencia RMB, que es la que nos proporciona la información deseada como, la velocidad, distancia restante, posición a la que se dirige el barco etc.

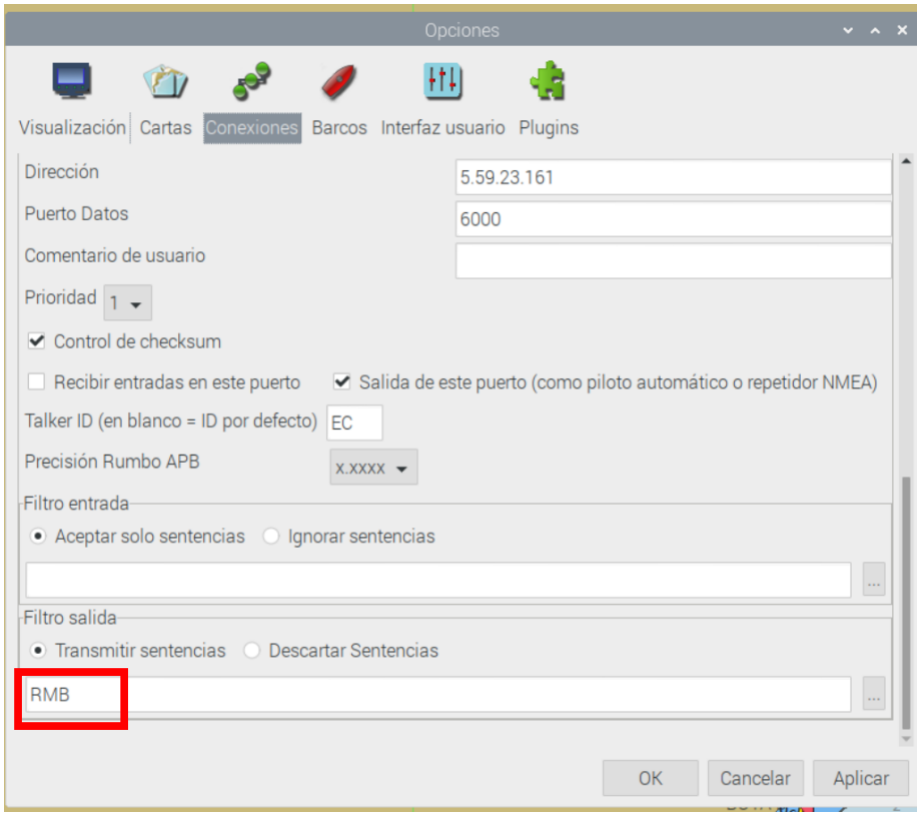


Figura 28 Filtro de salida NMEA

4.Desarrollo aplicación

4.1 Lenguaje Python

Como punto de partida para el desarrollo de la aplicación, había que seleccionar el lenguaje para conseguir dicho fin. La decisión fue optar por Python debido a que es un lenguaje de tipado dinámico, o lo que es lo mismo, que la comprobación de tipicación puede realizarse en tiempo de ejecución, además de ser uno de los lenguajes más portables, ya que no se necesitan grandes cambios en los equipos que lo ejecutan [14].

Guido Van Rossum es el creador y responsable de que Python exista. Se trata de un informático de origen holandés que fue el encargado de diseñar Python y de pensar y definir todas las vías posibles de evolución de este popular lenguaje de programación [14].

En las navidades de 1989 Van Rossum, mientras trabajaba en un centro de investigación holandés (CWI), decidió empezar un nuevo proyecto como pasatiempo personal. Pensó en darle continuidad a ABC, un lenguaje de programación que se desarrollo en el mismo centro en el que estaba trabajando [14].

Sin embargo, el proyecto no llegó mucho más lejos por las limitaciones del hardware de la época, así que Van Rossum decidió darle una segunda vida a su idea y partiendo de la base que tenía, empezó a trabajar en Python [14].

La última gran actualización de la historia de Python se produjo en el año 2008 con el lanzamiento de la versión 3.0, que venía a solucionar los principales fallos en el diseño de este lenguaje de programación.
Aunque Python mantiene su filosofía en esta última versión, como lenguaje de programación ha ido acumulando formas nuevas y redundantes de programar un mismo elemento. De ahí la necesidad de nuevas versiones que eliminen estos constructores duplicados [14].

Figura 29 Logo Python [14]

4.2 Sockets

Los sockets son mecanismos de comunicación que permiten establecer una comunicación entre dos procesos, incluso estando en máquinas distintas, lo que proporciona una gran interconectividad. Este mecanismo surge a principio de los años 80 con el sistema Unix de Berkeley. Son una forma sencilla de que dos programas se transmitan datos.

La comunicación queda definida por un par de direcciones IP, un protocolo de transporte y un par de puertos. Los sockets permiten establecer una arquitectura cliente-servidor. La comunicación es iniciada por uno de los programas, el cliente, mientras que el otro programa, el servidor, espera la solicitud de conexión por parte del cliente para iniciar la transmisión de datos.

En cuanto a las propiedades de los sockets, dependen de los protocolos que lo implementan. Los más conocidos son El TCP (Protocolo de control de transmisión) y el UDP (protocolo de datagrama de usuario).

El protocolo TCP está orientado a la conexión, se garantiza que toda la información llegue bien y en el orden en el que se envió. En cambio en el protocolo UDP solo se garantiza que si un mensaje llega, que este llegue bien, pero no garantiza que lleguen todos los mensajes o en el orden en el que se mandaron.

Para la comunicación con la Raspberry, hemos utilizado el protocolo UDP, ya que los mensajes que se reciben contienen la información de los sensores que cambiaran cada cierto tiempo, y en el receptor se leerá con una frecuencia mucho menor que la frecuencia de envío. Por tanto, no es necesario que se reciban todos los mensajes que se envían, y si se pierden algunos será irrelevante. En cambio, se gana sencillez y facilidad en la comunicación al elegir este protocolo.

4.3 Comunicación 3G

Como hemos dicho con anterioridad, la comunicación mediante red 3G es la más adecuada para el proyecto. La comunicación es unidireccional, o lo que es lo mismo, que solo uno de los dispositivos transmite información y el otro recibe. En nuestro caso es la raspberry la que transmite y nuestro pc el que escucha esa información.

Los dos dispositivos que intervienen en la comunicación, pertenecen a redes diferentes, ya que el pc estará conectado a la red WIFI y el raspberry a la red 3G. Esto hace que la complejidad de configuración aumente con respecto a si los dos equipos pertenecieran a la misma red.

Al pertenecer que trabajar en redes diferentes, hay que usar IPs públicas para la comunicación, pero éstas pueden cambiar en cualquier momento lo que mermaría la comunicación. Para solucionar esto lo que hemos hecho es crear dos dominios mediante la herramienta NO-IP, la cual da la posibilidad de hacerlo de forma gratuita. Trabajando con dominios nos aseguramos que siempre se establezca la comunicación ya que aunque la IP de alguno de ellos cambie, la propia red será la encargada de preguntar al servidor DNS, en el momento en el que se realice la conexión, a que IP corresponde el dominio determinado.

Los dos dominios creados son los siguientes:

-**“clientevnas.ddns.net”:** hace referencia a la raspberry.

-“vnasproyecto.ddns.net”: se corresponde con el PC situado en tierra.

Para el futuro hay que realizar las siguientes consideraciones:

Estos dominios creados tienen carácter temporal, es decir, expiran en cierto tiempo por lo que para poder seguir utilizándolos habría que renovarlos o también sustituirlos por dos nuevos. Esto queda a disposición de quien continúe el proyecto.

También hay que tener en cuenta que hay que abrir ciertos puertos del router de la red wifi a la que se conecta el pc de tierra, ya que hemos utilizado ciertos puertos que por defecto suelen estar cerrados.

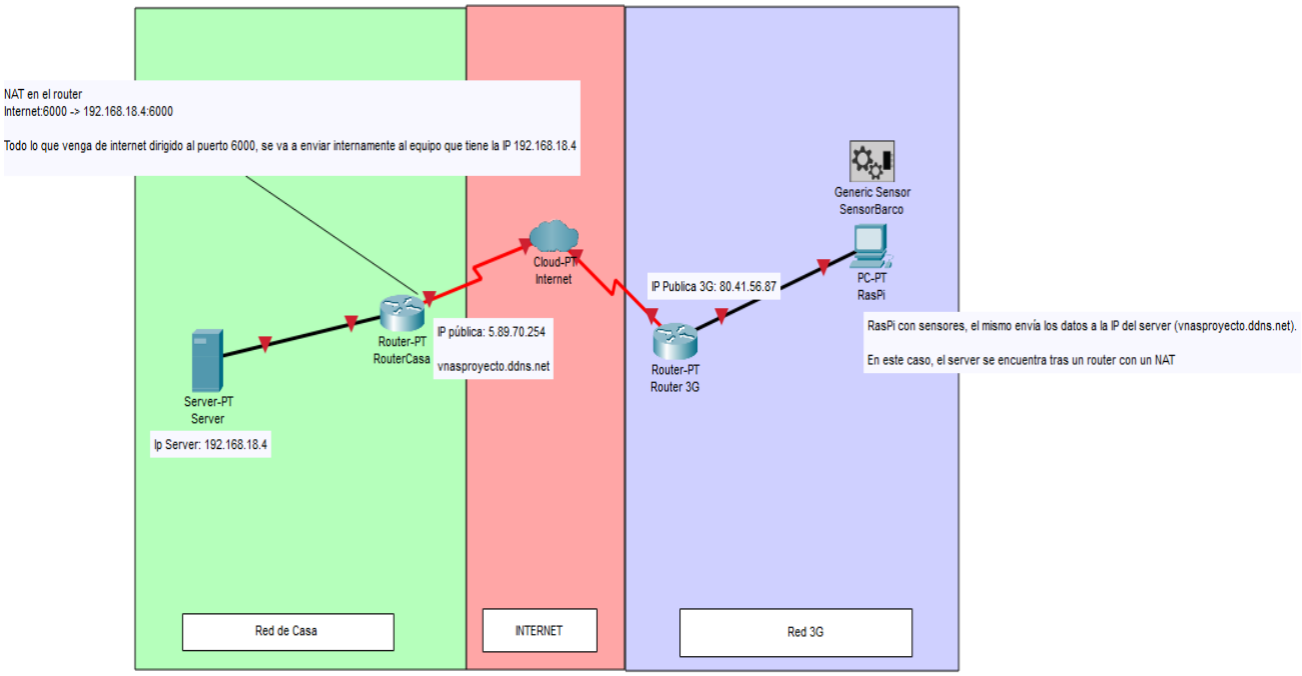


Figura 30 Esquema Global de red

Dentro de la red wifi al dispositivo que va a recibir la información, se le asigna una IP privada a la cual va a ser direccionado el tráfico que llegue al puerto 6000. Esta IP en nuestro caso es la 192.168.18.4, pero puede cambiar en el futuro y darnos problemas. Para evitar esto, en la configuración del PC podemos fijar esta ip, y en el router y protocolo DHCP, hacer que la ip fijada en el pc no sea asignada a ningún otro dispositivo que se conecte a la red en el futuro.

El NAT es un mecanismo con el que los routers IP cambian paquetes entre dos redes que asignan mutuamente direcciones ip incompatibles, en nuestro caso ips públicas y privadas.
Para ello, en la configuración del router, tenemos que ajustar la tabla NAT, con la que haremos que la información que llegue al puerto 6000, sea redireccionada a la interfaz de red 192.168.18.4. La interfaz gráfica de la configuración del router puede cambiar dependiendo del modelo, pero suele ser similar.

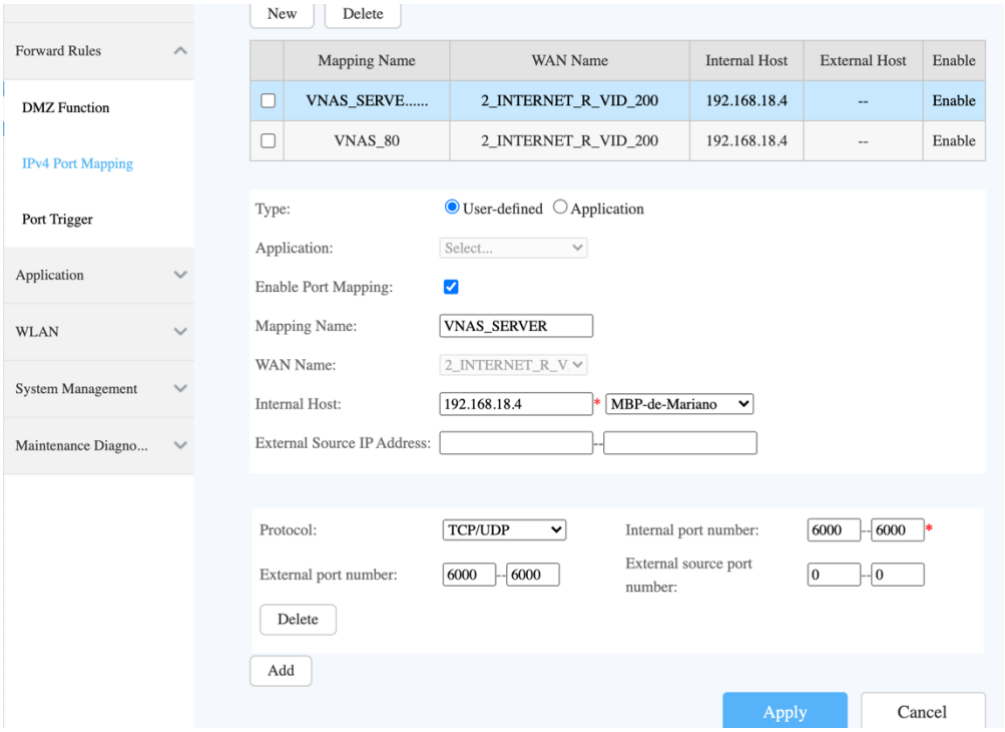


Figura 31 NAT de puerto

4.4 Programación Multithreading

Un hilo o Thread en Python, es un flujo independiente de procesamiento de información. Hay veces, que debido a las necesidades de la aplicación que queremos desarrollar es necesario que se ejecuten varios subprocesos de manera simultanea al proceso principal.

Ejecutar varios flujos independientes de manera simultánea puede parecer similar a ejecutar varios programas diferentes a la misma vez, pero con ciertos beneficios como los que exponemos a continuación:

- Dos o mas hilos o subprocesos comparten el mismo espacio de datos que el programa principal, lo que hace que puedan compartir información o comunicarse de manera más sencilla que si fueran completamente independientes.

- Los hilos se denominan procesos ligeros, ya que no necesitan mucha carga de memoria, en comparación con la ejecución de un programa principal.

Los procesos que pasan la mayoría del tiempo de ejecución esperando eventos externos son generalmente los mejores candidatos para el subproceso. Es muy probable que los programas que necesiten mas recursos de la CPU y que pasen poco tiempo esperando eventos externos no se ejecuten más rápido y no nos sea rentable utilizar los hilos para estos casos.

La aplicación desarrollada en este proyecto, consta de dos procesos principalmente, uno que se encarga de la comunicación con la Raspberry, para la recepción de las tramas de datos que contienen la información deseada, y otro proceso en el que dichos datos, se muestran en tiempo real por medio de la herramienta Tkinter de Python, que se desarrollará mas adelante.

La necesidad de crear dos hilos para los procesos radica en que, cada uno de los procesos utiliza un bucle infinito “While True”, uno para la actualización del puerto de datos y otro es el mainloop que utiliza Tkinter para escuchar los eventos que ocurren en la interfaz gráfica.

```
ip="192.168.18.4"
port=6000
SIZE=1024


def conectar():

    while True:
        time.sleep(1)
        data, addr=sock.recvfrom(SIZE)
        decoded=data.decode('utf-8')
        info=decoded.split(",")[1:15]
        Ilatitud=info[5]+' '+info[6]
        latitud.set(Ilatitud)
        Ilongitud=info[7]+' '+info[8]
        longitud.set(Ilongitud)
        Irumbo=info[10]
        rumbo.set(Irumbo)
        velocidad.set(info[11])
        distancia_restante.set(info[9])
        if info[12]=="V":
            estado.set("EN PROCESO")
        else:
            estado.set("FINALIZADO")


def crear_hilo():
    hilo=threading.Thread(target=conectar)
    hilo.start()
```

Lo primero que necesitamos hacer es crear el hilo en el momento que queramos iniciar la comunicación. Por esto asociamos la función crear_hilo, al botón “conectar”, para que cuando se accione, se ejecute la función, se cree el hilo, y se inicie la comunicación.

Para la comunicación con la Raspberry y la recepción de los datos, definimos la función “conectar”, en la que primero hacer uso de la librería timer, y hacemos un time.sleep(1) para no saturar el procesador. En el bucle infinito, ejecutamos la función “recvfrom” para leer la información recibida al puerto, cuya comunicación se crea antes de invocar la función.

La trama NMEA, como describiremos en capítulos posteriores, tiene una forma preestablecida, en la que cada campo representa un dato diferente. Cada uno de estos campos está separado por comas, por tanto, con la función Split(‘,’), podemos extraer cada uno de los campos e introducirlos en una posición de un array para su posterior tratamiento. En nuestro caso, ese array lo hemos denominado “info”, por eso asignamos a cada etiqueta una posición de este, según cual sea.

4.5 Interfaz gráfica de usuario

Para el desarrollo de la interfaz gráfica he utilizado la librería Tkinter, la cual es una adaptación de Tcl/Tk orientada a objetos, que proporciona un conjunto de herramientas para administrar ventanas.
Dentro de una interfaz gráfica podemos identificar 3 elementos principales:

-**Ventana:** es el espacio visual principal de la interfaz gráfica, donde se colocan en ella los frames y widgets que conforman la interfaz en su totalidad.
-**Frame:** es un widget que sirve para aglutinar el resto de widget y dividir el espacio de la ventana de manera ordenada. Se podría añadir otros widgets directamente sobre la ventana sin necesidad de un frame pero no es lo recomendable.
-**Widget:** son los elementos secundarios con los que el usuario interactúa con el programa, como los botones, etiquetas, campos de texto etc.

Lo primero que hacemos para empezar a desarrollar la interfaz es crear la ventana, donde iremos colocando el resto de elementos. Para establecer un tamaño a esta utilizamos la función geometry, a la que pasamos como argumento un string con el tamaño en píxeles de ésta.

```
ventana=tk.Tk()
ventana.title("VNAS")
ventana.geometry("900x450")
ventana.configure(bg=color_fondo)
ventana.iconbitmap('/Users/MarianoFenandezFerrer/Documents/proyecto_codigo/icono_vnas.ico')

#resto de componentes
.
.
.
.
.
.
.

ventana.mainloop()
```

Es importante colocar el método mainloop al final del código, ya que es el bucle infinito que se utiliza para esperar cualquier interacción por parte del usuario en la interfaz. Como ya se explicó anteriormente, es la razón por la cual se tuvo que utilizar programación basada en hilos.

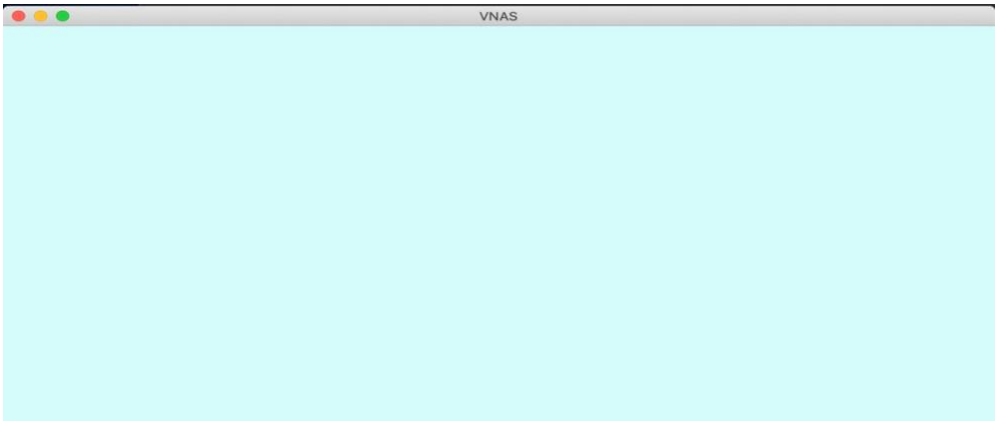


Figura 32 Ventana VNAS

Una vez creada nuestra ventana, lo que hacemos es instanciar el socket, o lo que es lo mismo la comunicación que vamos a usar para recibir los datos de la Raspberry. Con la función socket establecemos la comunicación, y con bind, asociamos esa comunicación con una IP y un puerto.

```
ip="192.168.18.4"
port=6000

sock = socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)
sock.bind((ip,int(port)))
```

Para terminar, creamos las etiquetas que van a contener los valores de la información a representar, y las etiquetas que indican a que corresponde dicho valor. Para que una etiqueta pueda cambiar su valor conforme se actualiza éste, tenemos que hacer uso de Stringvar, y asociar posteriormente ese Stringvar a la etiqueta.

Para crear el logo que aparece en la parte superior de la interfaz, lo hacemos con una label a la que le pasamos un objeto de la clase PhotoImage creado previamente. Al constructor de la clase PhotoImage le pasamos la ruta del archivo de imagen que queremos que forme el logo.

```
my_img= PhotoImage(file="logo_vnas.gif")
label_imagen=tk.Label(ventana,
image=my_img,bg=color_fondo).place(x=350, y=0)

label_latitud=tk.Label(ventana, text = "LATITUD DESTINO:
",font='Helvetica 16 bold',bg=color_fondo).place(x=100,y=180)
label_longitud=tk.Label(ventana, text = "LONGITUD DESTINO:
",font='Helvetica 16 bold',bg=color_fondo).place(x=100,y=240)
label_valor_rumbo=tk.Label(ventana, text = "RUMBO: ",font='Helvetica
16 bold',bg=color_fondo).place(x=550,y=180)
label_distancia_restante=tk.Label(ventana, text = "DISTANCIA
RESTANTE: ",font='Helvetica 16
bold',bg=color_fondo).place(x=550,y=240)
label_velocidad=tk.Label(ventana, text = "VELOCIDAD:
",font='Helvetica 16 bold',bg=color_fondo).place(x=100,y=300)
label_estado=tk.Label(ventana, text = "ESTADO VIAJE:
",font='Helvetica 16 bold',bg=color_fondo).place(x=550,y=300)

btn_inicio=tk.Button(ventana,text="CONECTAR",command=crear_hilo,widt
h=10,bg=color_fondo).place(x=380,y=350)
btn_salir=tk.Button(ventana,
text="SALIR",command=salida,width=10,bg=color_fondo).place(x=380,y=3
90)

#valores

latitud = tk.StringVar()
label_valor_latitud=tk.Label(ventana, textvariable = latitud,
fg='blue', font='Helvetica 16
bold',bg=color_fondo).place(x=290,y=180)

longitud = tk.StringVar()
label_valor_longitud=tk.Label(ventana, textvariable = longitud,
fg='blue', font='Helvetica 16 bold'
,bg=color_fondo).place(x=290,y=240)

rumbo=tk.StringVar()
label_valor_rumbo=tk.Label(ventana, textvariable = rumbo, fg='blue',
font='Helvetica 16 bold',bg=color_fondo).place(x=760,y=180)
```

```
distancia_restante=tk.StringVar()
label_valor_distancia_restante=tk.Label(ventana, textvariable =
distancia_restante, fg='blue', font='Helvetica 16
bold',bg=color_fondo).place(x=760,y=240)

velocidad=tk.StringVar()
label_valor_velocidad=tk.Label(ventana, textvariable = velocidad,
fg='blue', font='Helvetica 16 bold',bg=color_fondo
).place(x=290,y=300)

estado = tk.StringVar()
label_valor_estado=tk.Label(ventana, textvariable = estado,
fg='blue', font='Helvetica 16 bold',bg=color_fondo
).place(x=760,y=300)
```

Al ejecutar la app tendrá el siguiente aspecto. Podemos ver que tenemos dos botones, con el botón conectar, el usuario podrá iniciar la comunicación. Para que esto sea posible, al definir el botón, en el campo command tenemos que poner la referencia a la función que queremos llamar. En este caso, será la función “crear_hilo”, en la que crearemos el hilo de datos, para que en él se produzca la comunicación con la función conectar.

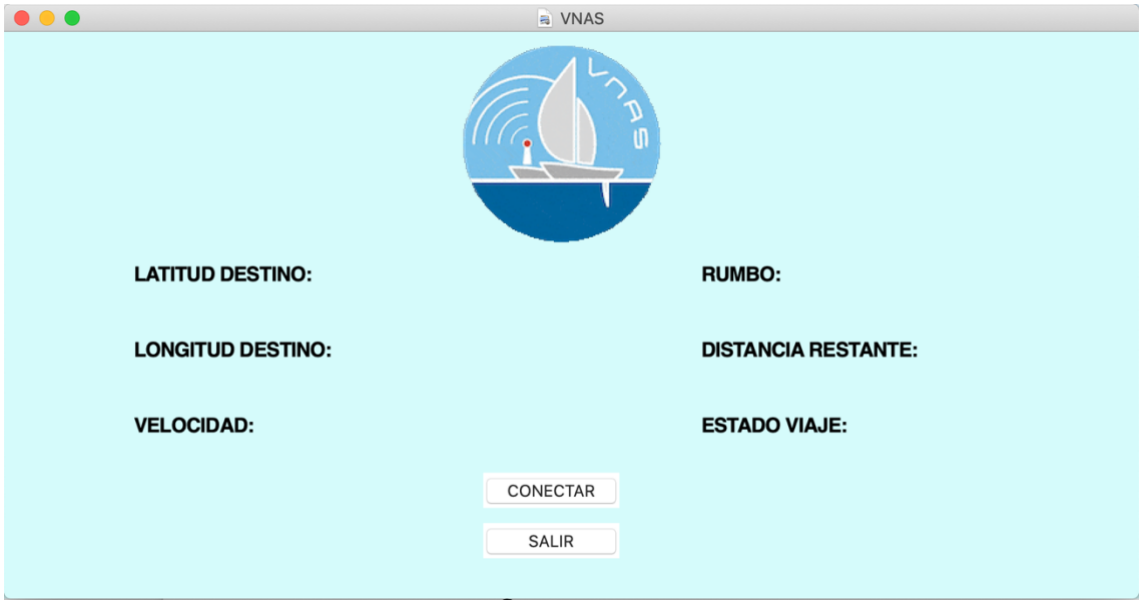


Figura 33 Interfaz Gráfica de Usuario

El segundo botón es el de “salir”, con el que podemos cerrar la app y finalizar la comunicación. Para esto, al definir el botón, en el apartado command ponemos la referencia a la función salida, en la que llamamos a la función sys.exit para cerrar finalizar la ejecución del programa.

```
def salida():
    response=messagebox.askokcancel("AVISO","¿DESEA SALIR DE LA
TELEMETRÍA?")
    if response == 1:
        sys.exit()
```

La ventana emergente que se genera al pulsar el botón salir, se crea con la clase messagebox, la cual tiene un atributo, response, con el que podemos saber el botón de la ventana emergente que ha sido pulsado y actuar en consecuencia.

Figura 34 Aviso fin conexión

4.6 Estándar NMEA

NMEA es la abreviatura de **National Marine Electronics Association**. Es una asociación fundada en1957 por un grupo de fabricantes de electrónica para obtener un sistema común de comunicación entre las diferentes marcas de electrónica naval. Poco a poco se fueron sumando todos los fabricantes a este estándar, además de organizaciones oficiales y gubernamentales.[13]

NMEA se creó para el intercambio de información digital entre productos electrónicos marinos. El primer protocolo estándar se llamó NMEA 0183, y es el que todavía utilizan y aceptan la mayoría de los equipos electrónicos que llevamos a bordo. Es un protocolo que define los requerimientos de datos y tiempo de transmisión en el formato serial a una velocidad de 4800 baudios (bits por segundo). Define también la norma que cada equipo sea emisor de NMEA y pueda ser escuchado por muchos receptores.[13]

La nueva versión NMEA 2000 mejora fundamentalmente en la velocidad de transmisión, pero no cambia en el concepto de conectividad.[13]

Con NMEA0183 teníamos un instrumento emitiendo datos hacia uno o varios instrumentos que recibían información. El GPS le da la señal al plotter, al radar y al piloto automático. Éste último está escuchando continuamente la señal que le envía el GPS y aprovecha su señal para mostrarla en su display. ¿Y qué ocurre si tenemos dos dispositivos que emiten la posición GPS del barco? Este protocolo sólo estaba pensado para emitir o recibir por el mismo cable. Sabe “hablar” o “escuchar” y no es capaz de “escuchar” dos señales que le hablan a la vez. Para ello necesitábamos un aparato que hace de distribuidor NMEA, que se encarga de recibir toda la información que emite cada componente, la procesa, la suma y emite una nueva señal con toda la información encadenada.[13]

En muchos casos acabamos por conectar todos los equipos que tienen salida NMEA a un distribuidor o adaptador que reúne la información que emite cada uno y la entrega a un cable RS232 que conectamos en el PC.[13]

Ya vimos que la desaparición del puerto serie (interface RS232) nos obliga a incorporar una adaptador “SERIAL TO USB”, para recibir la señal por los puertos USB que tienen todos los equipos actuales. Hay muchos adaptadores disponibles en el mercado. La marca SWEEX dispone de un modelo llamado “USB to SERIAL cable” cuyo funcionamiento he podido verificar con diferentes programas y equipos.
La aparición de este adaptador ha salvado la supervivencia de todo el software que limitaba su recepción de datos a NMEA 0183, ya que en su estándar se define la entrega en el PC a través de un interface serie. [13]

Una de las ventajas del nuevo estándar NMEA 2000 es que soporta conexión de equipos que envían y reciben información de forma simultánea, por lo que no requiere el distribuidor de información que tenemos en el estándar 0183. [13]

Para nuestra aplicación, nos interesa con especial interés la sentencia RMB, que tiene la siguiente estructura:

```
$GPRMB,A,0.66,L,001,002,3717.24,N,12379.57,W,041.3,032.5,000.5,V*0B
```

A	Estado de los datos: A= OK, V=warning
0.66,L	Error por desviación, viraje: L= izquierda, R= Derecha
001	ID waypoint origen
002	ID waypoint destino
3717.24,N	Latitud destino
12379.57,W	Longitude destino
041.3	Distancia restante hacia el destino
032.5	Rumbo hacia el destino
000.5	Velocidad en nudos hacia el destino
V	Estado del viaje: V= en proceso, A= finalizado

Figura 35 Tabla datos NMEA RMB

4.7 Simulación y resultados.

En la figura 35 podemos ver la aplicación durante el proceso de recepción y visualización de datos. Hemos decidido representar los parámetros que se muestran en la figura, pero hay disponibles muchos más. Para cambiar éstos solo habría que cambiar el filtro de salida en el OpenCPN para que se transmitan las sentencias NMEA que queramos y ajustar el código para manejar correctamente las sentencias recibidas.

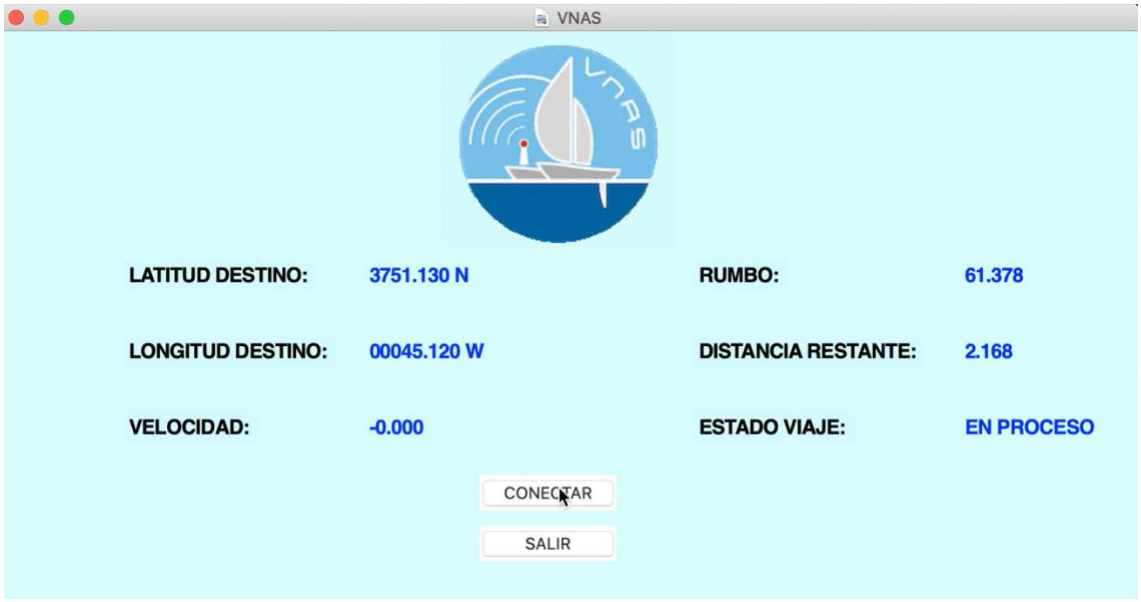


Figura 36 Aplicación en ejecución

5. Conclusiones y líneas futuras.

5.1 Conclusiones

Podemos sacar varias conclusiones de este proyecto. En primer lugar, tener la oportunidad de trabajar con instrumentación como los sensores GARMIN o los cables SeaTalk, nos ha permitido situarnos en un entorno real, ya que son utilizados no solo para aplicaciones didácticas como es este proyecto sino también en entornos reales con barcos comerciales y privados. Esto hace que podamos poner a prueba al sistema en condiciones de alta exigencia.

Por otra parte, en lo personal, el hecho de desarrollar la aplicación en un lenguaje que era totalmente desconocido para mi, como es Python me ha permitido ampliar los conocimientos sobre los lenguajes de programación, lo que puede servirme para mi vida profesional en el futuro.

5.2 Líneas futuras

Como líneas futuras, se podría establecer conexión entre la raspberry y la red 4G o 5G lo que mejoraría el rendimiento del sistema.

Por otra pate, se podrían utilizar los datos recibidos de la posición y destino del barco para representarlo en tiempo real en un mapa. Además, también se podría implementar una comunicación bidireccional entre el equipo de tierra y la raspberry situada a bordo para poder darle órdenes y cambiar tu posición una vez haya iniciado su trayecto.

Otro aspecto que se podría mejorar sería poder introducir más sensores e instrumentación para así poder tener mejor monitorización de la embarcación, además de montarlo todo en el barco para hacer pruebas reales.

6. ANEXOS

6.1 Anexo 1: Conexión modem 3G

Debido a las circunstancias en las que se va a desarrollar el proyecto, es totalmente necesario disponer de acceso a la red 3G, ya que el barco podrá estar a varias millar de tierra y no es viable cualquier otro tipo de comunicación como puede ser mediante una red de área local o Wi-Fi. Para esto, hemos utilizado un modem usb Huaweii E1752Cu. Para poder configurarlo correctamente, hemos tenido que realizar los siguientes pasos [16]:

6.1.1 Instalación del software necesario

Con la raspberry conectada con wifi, abrimos la consola de comando e ingresamos lo siguiente:

```
Sudo apt-getupdate
Sudo apt-get install ppp usb-modeswitch wvdial
```

6.1.2 Obtención de los códigos de conmutación USB

Reiniciamos el equipo SIN CONEXIÓN A INTERNET e ingresamos el siguiente comando:

```
Lsusb
```

Después de introducir el comando nos saldrá una lista con información, de la que nos interesa el código de producto y el de proveedor del dispositivo. En nuestro caso sería: 19d1:1433.

6.1.3 Generación de un archivo de configuración personalizado usb_modeswitch

Necesitamos crear un archivo de configuración personalizado para usb_modeswitch en Raspberry PI porque en un arranque en frío el dispositivo no siempre está activo todavía cuando usb_modeswitch se ejecuta al inicio y el dispositivo se deja en modo de almacenamiento USB.[12] Escribimos los siguiente:

```
cd / tmp
tar -xzvf /usr/share/usb_modeswitch/configPack.tar.gz 19d1 \: 1433
```

Escribimos en la parte en negrita, los números de producto y proveedor que anotamos en el paso anterior.

Ahora abra el archivo extraído con un editor de texto como leafpad reemplazando los códigos anotados en el paso 2.

hoja 19d1:1433

El contenido del archivo que nos interesa es el siguiente:

Proveedor predeterminado = 0x **19d2 Producto predeterminado** = 0x **2000**

TargetVendor = 0x **19d2**
TargetProduct = 0x **2002**

MessageContent = "55534243123456780000000000000061e000000000000000000000000000000"
MessageContent2 = "55534243123456790000000000000061b000000020000000000000000000000"
MessageContent3 = "55534243123456702000000080000c85010101180101010101010000000000"

El siguiente paso es abrir el archivo **/etc/usb_modeswitch.conf** y agregar la información anterior. Ingresamos los comandos siguientes:

sudo leafpad /etc/usb_modeswitch.conf

6.1.4 Generación archivo de configuración wvdial

El siguiente paso es crear un archivo de configuración para wvdial para que pueda conectarse con su proveedor de servicios.

Abra una ventana de terminal e ingrese:

sudo leafpad /etc/wvdial.conf

Reemplazamos el contenido del archivo por lo siguiente:

[Dialer 3gconnect]
Init1 = ZTA
Init2 = ATQ0 V1 E1 S0 = 0 y C1 y D2 + FCLASS = 0
Init3 = AT + CGDCONT = 1, "IP", "internet"
Modo estúpido = 1

Tipo de módem = Módem analógico
ISDN = 0
Teléfono = * 99 #
Módem = / dev / gsmmodem
Nombre de usuario = {}
Contraseña = {}
Baudios = 460800

Reemplazamos lo siguiente del archivo anterior:

1. Internet con el **APN de** su proveedor de servicios

2. El número de **teléfono** si necesita marcar un código diferente para conectarse.

3. El nombre de **usuario** y la **contraseña** si es necesario. Para dejar el nombre de usuario y la contraseña en blanco, use {}

6.1.5 Conexión a internet

Para conectar, debemos asegurarnos de que el dispositivo esté en modo módem.
Abra una terminal e ingrese:

```
sudo usb_modeswitch -c /etc/usb_modeswitch.conf
```

Luego, por fin, nos conectamos a internet:

```
wvdial 3gconnect
```

6.2 Anexo 2: Código completo de la aplicación

```
# -*- coding: utf-8 -*-


import tkinter as tk
import threading,socket,os,time,sys
from tkinter import *
from tkinter import messagebox

#[u'$ECRMB', u'A', u'0.001', u'L', u'001', u'002', u'3749.790', u'N',
u'00044.138', u'W', u'2.697', u'96.480', u'0.041', u'V', u'A*5B\r\n']

color_fondo="#CCFBFA"
ip="192.168.18.4"
port=6000
SIZE=1024


def conectar():

    while True:
        time.sleep(1)
        data, addr=sock.recvfrom(SIZE)
        decoded=data.decode('utf-8')
        info=decoded.split(",")[1:15]
        Ilatitud=info[5]+' '+info[6]
        latitud.set(Ilatitud)
        Ilongitud=info[7]+' '+info[8]
        longitud.set(Ilongitud)
        Irumbo=info[10]
        rumbo.set(Irumbo)
        velocidad.set(info[11])
        distancia_restante.set(info[9])
        if info[12]=="V":
            estado.set("EN PROCESO")
        else:
            estado.set("FINALIZADO")




def salida():
    response=messagebox.askokcancel("AVISO","¿DESEA SALIR DE LA
TELEMETRÍA?")
    if response == 1:
        sys.exit()

def crear_hilo():
    hilo=threading.Thread(target=conectar)
    hilo.start()
```

```
ventana=tk.Tk()
ventana.title("VNAS")
ventana.geometry("900x450")
ventana.configure(bg=color_fondo)
ventana.iconbitmap('/Users/MarianoFenandezFerrer/Documents/proyecto_c
odigo/icono_vnas.ico')


my_img= PhotoImage(file="logo_vnas.gif")
label_imagen=tk.Label(ventana,
image=my_img,bg=color_fondo).place(x=350, y=0)

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((ip,int(port)))

label_latitud=tk.Label(ventana, text = "LATITUD DESTINO:
",font='Helvetica 16 bold',bg=color_fondo).place(x=100,y=180)
label_longitud=tk.Label(ventana, text = "LONGITUD DESTINO:
",font='Helvetica 16 bold',bg=color_fondo).place(x=100,y=240)
label_valor_rumbo=tk.Label(ventana, text = "RUMBO: ",font='Helvetica
16 bold',bg=color_fondo).place(x=550,y=180)
label_distancia_restante=tk.Label(ventana, text = "DISTANCIA
RESTANTE: ",font='Helvetica 16
bold',bg=color_fondo).place(x=550,y=240)
label_velocidad=tk.Label(ventana, text = "VELOCIDAD:
",font='Helvetica 16 bold',bg=color_fondo).place(x=100,y=300)
label_estado=tk.Label(ventana, text = "ESTADO VIAJE:
",font='Helvetica 16 bold',bg=color_fondo).place(x=550,y=300)

btn_inicio=tk.Button(ventana,text="CONECTAR",command=crear_hilo,width
=10,bg=color_fondo).place(x=380,y=350)
btn_salir=tk.Button(ventana,
text="SALIR",command=salida,width=10,bg=color_fondo).place(x=380,y=39
0)

#valores

latitud = tk.StringVar()
label_valor_latitud=tk.Label(ventana, textvariable = latitud,
fg='blue', font='Helvetica 16
bold',bg=color_fondo).place(x=290,y=180)

longitud = tk.StringVar()
label_valor_longitud=tk.Label(ventana, textvariable = longitud,
fg='blue', font='Helvetica 16 bold'
,bg=color_fondo).place(x=290,y=240)

rumbo=tk.StringVar()
label_valor_rumbo=tk.Label(ventana, textvariable = rumbo, fg='blue',
font='Helvetica 16 bold',bg=color_fondo).place(x=760,y=180)
```

```
distancia_restante=tk.StringVar()
label_valor_distancia_restante=tk.Label(ventana, textvariable =
distancia_restante, fg='blue', font='Helvetica 16
bold',bg=color_fondo).place(x=760,y=240)

velocidad=tk.StringVar()
label_valor_velocidad=tk.Label(ventana, textvariable = velocidad,
fg='blue', font='Helvetica 16 bold',bg=color_fondo
).place(x=290,y=300)

estado = tk.StringVar()
label_valor_estado=tk.Label(ventana, textvariable = estado,
fg='blue', font='Helvetica 16 bold',bg=color_fondo
).place(x=760,y=300)




ventana.mainloop()
```

6.3 Anexo 3: Presupuesto

Antena GPS GARMIN 19X 10Hz NMEA 2000	239,00 €
Compás ciego AIRMAR H2183 NMEA 2000	810,00 €
Smart TRIDUCER AIRMAR DTS800	393,00 €
Anemómetro con veleta RAYMARINE	519,00 €
NGW-1 NMEA 2000 GATEWAY	158,95 €
Raymarine SeaTalk 1 to SeaTalk ng converter	118,49 €
Baterías 12 V	100,00 €
Conector tipo T varios puertos NMEA 2000	58,90 €
Raspberry pi	66,55 €
TOTAL	2463,89 €

*Precio con IVA incluido

Bibliografía

[1] https://www.microtransat.org/

[2] Avances en el sistema de control para la navegación de un catamarán de forma autónoma. José Carlos Urrea Celdrán.

[3] https://es.wikipedia.org/wiki/Telemetr%C3%ADa

[4] https://onnautic.com/blog/que-es-el-sistema-de-identificacion-automatica-o-ais/

[5] https://onnautic.com/blog/como-funciona-un-radar-de-barco/

[6] https://www.liquid-robotics.com/

[7] https://www.saildrone.com/

[8] https://www.arduino.cc/

[9] Desarrollo de una aplicación web de telemetría para el control de un barco autónomo. Hassan Bahari.

[10] https://actisense.com/products/nmea-2000-gateway-ngw-1/

[11] https://www.raymarine.es/view/index-id=1597.html

[12] https://www.garmin.com/es-ES/

[13] https://www.airmar.com/productdescription.html?id=110

[14] https://www.tokioschool.com/noticias/historia-python/

[15] https://www.informaticaabordo.com/2011/10/%C2%BFque-es-el-standar-nmea/

[16] https://tchekbo.wordpress.com/2018/03/31/how-to-setup-a-usb-3g-modem-using-usb_modeswitch-and-wvdial/