ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

TRABAJO FIN DE GRADO

Desarrollo de una aplicación web de telemetría para el control de un barco autónomo.

AUTOR: Hassan Bahari Rhetassi

DIRECTOR: José María Molina García-Pardo

CODIRECTOR: Antonio Mateo Aroca

Julio/2019

Autor	Hassan Bahari Rhetassi
E-mail del autor	bhassan2@live.com
Director	José María Molina García-Pardo
E-mail del director	josemaria.molina@upct.es
Título del TFG	Desarrollo de una aplicación web de telemetría para el control de un barco autónomo.

Resumen:

VNAS (Vehículo de Navegación Autónoma en Superficie) es un proyecto que consiste en el diseño e implementación de un barco autónomo desarrollado con las tecnologías más eficientes y menos costosas, y sobre todo con el objetivo de participar en un futuro en competiciones trasatlánticas como The Microtransat Challenge.

Se trata de un proyecto multidisciplinar en el cual participan varias escuelas que se encargan de diferentes tareas como la creación del casco del catamarán, la implementación de un sistema de control para la navegación del barco, la implementación de la telemetría para la comunicación y el desarrollo del software necesario para la representación y el control del barco.

Este proyecto consiste principalmente en el desarrollo de un software capaz de representar en tiempo real los datos recogidos por los componentes situados en el barco como los sensores, así como el control de dicho prototipo.

Titulación	Grado en Ingeniería de Sistemas de Telecomunicación
Departamento	Tecnologías de la Información y las Comunicaciones
Fecha de Presentación	Julio/2019

Agradecimientos:

Bismillahi-r-Rahmani-r-Rahim:

En primer lugar, me gustaría agradecer a Allah y a mis padres por apoyarme y dármelo todo. A mis dos hermanos, mi familia en general, y grandes amigos.

Y en especial a profesores, personal de la universidad, Humberto Martínez Barberá y sobre todo a José María Molina García Pardo por trabajar juntos en muchos proyectos.

Alhamdulilah

Índice

Índice de figuras

1. Introducción

1.1 Motivación

La finalidad de este proyecto es diseñar y desarrollar un prototipo de una embarcación autónoma con la idea de participar en un futuro en “The Microtransat Challenge”. “The Microtransat Challenge” es una competición en la que participan barcos con el objetivo de que compitan de manera autónoma. [1]

El proyecto, cuyo nombre es Vehículo de Navegación Autónoma en Superficie (VNAS), consiste en un trabajo multidisciplinar en el que se realizan diferentes tareas tales como la creación del casco del catamarán, el control de los motores, el montaje de los componentes del barco y la realización de un sistema de telemetría que se basará en una comunicación bidireccional a través del canal radio entre un servidor y un dispositivo electrónico situado en el barco para transmitir y recibir información.

Figura 1 Logo de VNAS

Como resultado final, el barco autónomo deberá superar unas pruebas llevadas a cabo con la misión de poner en práctica dicho vehículo, así como mejorar y obtener un dispositivo final altamente capacitado para competir en Microtransat Challenge. En dichas pruebas se comprobará la flotabilidad de la embarcación, el funcionamiento de los motores instalados y la adquisición y representación de los datos proporcionados por los sensores tales como la posición, la velocidad, la temperatura y la predicción del tiempo.

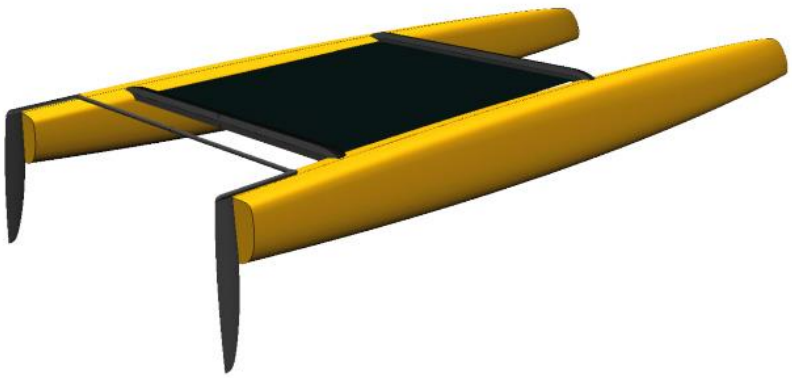

Figura 2 Estructura base de VNAS

1.2 Objetivos

Los objetivos marcados para este trabajo fin de grado son varios:

- Estudio profundo del proyecto VAMAR [2] con el fin de replicarlo, así como la familiarización con las tecnologías de telemetría que se usan en el proyecto de Motostudent [3].
- Estudio de la tecnología NMEA 0183 [4]. La embarcación se diseña con unos sensores conectados a una Raspberry que hace de cerebro para que ésta a su vez pueda enviar y recibir información con el servidor. La interconexión entre la Raspberry y los sensores se realiza a través del protocolo NMEA 0183 el cual permite la comunicación entre los instrumentos marítimos.
- Desarrollo de la comunicación entre servidor y cliente. Realizar una comunicación bidireccional a través de la tecnología 3G UMTS, WIFI y la red privada mediante el instrumento CMW500, entre una estación base que consistirá en un ordenador situado en el laboratorio SICOMO y una estación móvil que será la embarcación diseñada.
- Desarrollo de un software para representar la información del barco autónomo. El objetivo principal de este trabajo fin de grado es el de diseñar una aplicación web en la cual poder representar en tiempo real la información enviada por los sensores de la embarcación. Para ello, hay que estudiar el lenguaje de la información enviada por el prototipo con el fin de que se pueda interpretarla y traducirla en información accesible al usuario. Además, es necesario estudiar los lenguajes de programación que se van a usar para la implementación de la aplicación web.
- Desarrollo de un software que se encargue de enviar información tal como señales de órdenes para el prototipo con la finalidad de que realice una acción como el desplazamiento de un punto a otro a una velocidad determinada u otra función por el estilo.

1.3 Fases del proyecto

El proyecto se ha llevado a cabo a lo largo de un período de 12 meses desde que se empezaron a realizar los primeros pedidos, hasta la realización y la entrega de la memoria en julio de 2019.

La primera fase del proyecto se inició con la lectura de los PFCs [5] de un proyecto similar, MotoStudent, en el cual se utilizan tecnologías similares a las que vamos a usar en VNAS. Durante esta fase, se ha puesto en marcha el sistema de telemetría que se usa en MotoStudent con la idea de familiarizarnos y usarlo en nuestro proyecto.

La segunda fase del proyecto consistió en la replicación del sistema náutico del proyecto VAMAR, situado en el laboratorio de investigación de Ingeniería de la Información y las Comunicaciones en Fuente Álamo cuyo autor es Humberto Martínez Barberá. Para ello se realizaron varias reuniones junto a este profesor en los cuales nos ha explicado de

forma detallada su proyecto como las tecnologías usadas tales como NMEA 0183, la aplicación programada en Java usada para la transmisión y recepción de los datos de los sensores, el formato de la información de los sensores, etc. Posteriormente, realizamos los pedidos de los componentes necesarios como los sensores, cables NMEA 0183, Raspberry, baterías, Modem 3G, conversores SeaTalk 1, etc.

Durante la tercera fase y una vez recibido los pedidos, empezamos a realizar el montaje de nuestro proyecto, replicando el proyecto VAMAR y a su vez, añadiendo algunas que otras funcionalidades en la Raspberry. Una de las funciones más importantes ha sido la configuración del modem 3G para que éste pudiera conectarse a la red pública y pueda comunicar la Raspberry con nuestro servidor.

La cuarta fase se basó en el desarrollo del software en el servidor encargado de la recepción de información, así como el desarrollo de una aplicación web encargada de la interpretación y representación en tiempo real de los datos de los sensores. Para ello, se realizó un estudio profundo de los lenguajes necesarios en cada etapa para conseguir el objetivo.

La última fase del proyecto consistió en la unión de cada una de las anteriores partes para realizar una prueba: correcto funcionamiento de cada uno de los componentes en el catamarán, comunicación bidireccional, transmisión y recepción de información entre el barco y el servidor vía red 3G, interpretación y representación de la información en la aplicación web diseñada para tal fin.

1.4 Estructura de la memoria

La memoria de este proyecto está distribuida de la siguiente forma:

- En primer lugar, se realiza una introducción en la que se habla sobre el objetivo principal del proyecto VNAS y los motivos por los que se lleva a cabo.

- Posteriormente, se realiza un estudio sobre la telemetría para saber en que consiste y se explica su uso en el ámbito de las embarcaciones mediante ejemplos de prototipos reales.

- En el tercer apartado, se centra más en el proyecto VNAS. Se explica de manera detallada en que consiste, se menciona las tecnologías que usa y se fijan las diferentes tareas necesarias para conseguir el resultado buscado.

- En la cuarta parte, se mencionan las herramientas y los lenguajes de programación necesarios para llevar a cabo la aplicación web en la que se van a visualizar los resultados obtenidos.

- En la quinta fase, se explica paso a paso como se desarrolla la aplicación web con los diferentes lenguajes de programación.

- Durante la sexta fase, se muestra el resultado final de la aplicación web en la que se va a visualizar de manera exitosa la información en tiempo real del sensor GPS y sensor de Velocidad montado en el catamarán.

- Finalmente, en la última parte, se van a realizar conclusiones acerca de los resultados obtenidos y el proyecto en general, y también se van a realizar proposiciones como líneas futuras. Además, se adjuntarán una serie de anexos en los que vienen una serie de explicaciones de diferentes tecnologías usadas en este proyecto.

2. Estado del Arte

2.1 La telemetría

La telemetría es un sistema de comunicación cuyo objetivo es la de recoger datos desde un punto de difícil acceso físico para llevarlos a un equipo central donde pueden ser evaluados. El uso de la telemetría tiene lugar en varios campos como el espacio, medio terrestre, medio marítimo, domótica o robótica. La palabra telemetría tiene su origen en la terminología griega donde "tele" significa remoto, y "metron", quiere decir medida. La principal ventaja de la telemetría es la capacidad de la monitorización de información desde un receptor situado a una distancia variable desde el emisor. [6]

En general, el sistema de telemetría consta de tres partes fundamentales:

- Un equipo central el cual se encarga del procesamiento de datos recibidos por los sensores, así como el almacenamiento y la gestión de dichos datos con el fin de usarlos para otros fines. También se encarga de otras funciones como la codificación o la decodificación, corrección de errores, modulación o demodulación, etc.
- El medio de transmisión mediante el cual viaja la información entre los dos puntos puede ser por medio guiado, mediante cableado como la fibra óptica o cable coaxial, o por medio no guiado como radiofrecuencia, infrarrojo, ultrasonido, GSM, UMTS, LTE.
- Unos sensores o transductores como dispositivos remotos diseñados con las tecnologías requeridas para llevar a cabo tareas programadas por el equipo central.

Particularizando las anteriores definiciones en nuestro proyecto, el equipo central se trataría de nuestro servidor que hace la tarea de estación base. La red pública UMTS, sería el medio de propagación mediante el cual vamos a comunicarnos con el sensor que hace de estación móvil. Éste último, seria nuestra Raspberry, colocada en el catamarán donde recogerá información como posicionamiento, velocidad del barco, velocidad del viento, etc. y la enviará al servidor.

Figura 3 Ejemplo de las tres partes de un sistema de telemetría [7]

2.2 Historia de la telemetría

Los primeros pasos de la telemetría tienen sus orígenes durante el siglo XIX. Como medio de transmisión utilizaban cable. Uno de los primeros circuitos de transmisión de datos fue desarrollado en 1845 entre el "Winter Palace" del emperador de Rusia y el cuartel general del ejército.

En 1874, ingenieros franceses construyeron un sistema de telemetría capaz de enviar datos en tiempo real sobre la meteorología y los sensores profundidad de la nieve en la cima del Mont Blanc hasta Paris.

Más tarde, en 1901 fue patentado por el inventor americano C. Michalke un circuito de envío sincronizado de parámetros de rotación a distancia cuya principal utilidad era medir el ángulo de máquinas rotativa como por ejemplo la plataforma de una antena.

En 1906 fueron construidas en el observatorio Pulkovo en Rusia, una serie estaciones sísmicas con telemetría para recoger datos sobre terremotos y movimientos sísmicos.

Por otro lado, en el Canal de Panamá, entre 1913 y 1914 también fueron implementados sistemas de telemetría para monitorizar los niveles de agua y las esclusas.

Hasta ahora, las aplicaciones mencionadas eran a través de circuitos y cableados. Los primeros sistemas telemétricos "Wireless" o "sin cable" fueron desarrollados en 1930 por Robert Bureau en Francia y Pavel Molchanov en Rusia para aplicaciones en radio sondas. El sistema de Molchanov consistía en modular las medidas de temperatura y presión y transmitiéndolas por código Morse vía Wireless. También fue implementado un sistema llamado "Messina" de multiplexado de señales de radio en la nave V-2 alemana, pero era poco fiable y tanto en EE. UU. como en la URSS fueron rápidamente reemplazadas por sistemas mejores basados en la modulación por posición de pulso. [8]

2.3 Aplicaciones de la telemetría

Actualmente, la telemetría se da en muchas industrias como la espacial, la química y la eléctrica donde el proceso de monitorización y registro de mediciones constantes es eficiente. En el caso de la industria química o eléctrica es importante disponer de un sistema de seguridad que permita controlar las plantas para ser informados mediante la telemetría de cualquier incidente. En el medio espacial, las agencias como la NASA, la UK Space Agency, y la ESA, utilizan la telemetría para operar con satélites y con las estaciones base en la tierra.

Por otra parte, podemos encontrar el uso de la telemetría con mucha más precisión e importancia en el sector de las competiciones automovilísticas como la Formula 1, MotoGP y competiciones por el estilo. En este sector, la telemetría es altamente importante puesto que los equipos constan de una multitud de sensores que recogen datos en tiempo real que permiten al usuario trabajar con esa información con bastante eficacia.

2.4 Telemetría en embarcaciones

La evolución tecnológica en el sector marítimo cada vez está teniendo más protagonismo. En estos últimos años, cada vez hay más empresas que invierten en la robotización y automatización de vehículos marítimos. Los principales objetivos son la reducción de costes como el uso de energías renovables como la energía eléctrica en vez del petróleo o el coste del diseño puesto que se requiere de material más barato. También se busca disponer de un vehículo capacitado para trabajar bajo su autonomía sin necesidad de la intervención de una tripulación a bordo entre otros objetivos.

Rolls-Royace

A finales de 2018, gracias a la colaboración que empezó en 2015 entre Rolls Royce y Finferries, se realizó la primera demostración en el mundo de un ferry completamente autónomo, bautizado como SVAN (Safer Vessel with Autonomous Navigation). La demostración se realizó cubriendo la existente ruta de ferry que conecta el archipiélago de Turku, en Finlandia, realizando el viaje tanto de forma autónoma como por control remoto desde la estación base, a 45 km del archipiélago. [9]

Figura 4 Ferry SVAN

Energy Observer

El Energy Observer es el primer buque de energía autosuficiente con cero emisiones de gas de efecto invernadero o partículas finas que funciona con hidrógeno y fuentes renovables, gracias al acoplamiento de energía. Se trata de un desafío tanto humano como tecnológico, ya que esta embarcación necesita sol, viento y, por supuesto, las baterías o el hidrógeno. Las energías y los sistemas de almacenamiento se complementan entre sí. [10]

Figura 5 Energy Observer

Sea Hunter

Sea Hunter es el primer buque no tripulado de la US Navy, un cazador robótico de submarinos con capacidad para navegar durante 90 días autónomamente en cualquier condición climática sin necesidad de que ningún humano lo controle a bordo o remotamente, mediante una serie de sensores que le permiten evitar otros buques o cualquier tipo de accidente geográfico. [11]

Figura 6 Sea Hunter

3. Proyecto VNAS

En la Universidad Politécnica de Cartagena siempre se han ideado proyectos de investigación con el fin de desarrollar e implementar futuros dispositivos para una variedad de fines. Uno de esos proyectos se trata de VNAS el cual se propone por el departamento de Tecnologías de la Información y las Comunicaciones con la colaboración del centro de investigación y desarrollo de la Universidad de Murcia.

3.1 Descripción del proyecto VNAS

Siguiendo la misma línea del proyecto de MotoStudent el cual ha sido llevado a cabo por varios estudiantes y profesores y, además, ha tenido bastante éxito, nace la idea de desarrollar un vehículo marítimo como VNAS. Como se trata de un proyecto bastante importante, se busca la cooperación de varias escuelas de la universidad con el fin de repartir el trabajo, así como fomentar el trabajo en equipo y hacer partícipe de varios estudiantes de diferentes carreras. Por esa razón, VNAS es un proyecto multidisciplinar en el cual participan la Escuela Técnica Superior de Telecomunicación, la Escuela Técnica Superior de Naval y Oceánica y la Escuela de Industriales.

El objetivo de cada escuela es llevar a cabo una parte del proyecto que tenga relación con su sector. En este caso, como representantes de la escuela de Telecomunicación, nuestro objetivo es el desarrollo de la comunicación entre el barco y la estación base, así como de la implementación del software que se encarga de la representación de los datos de los sensores en tiempo real.

Después de varias reuniones, las primeras líneas que se han propuesto son las siguientes:

- Construcción del catamarán: esta tarea ha sido asignada para la escuela de Naval y Oceánica. Se busca el diseño de una estructura capacitada para el soporte de los componentes que irán montados y que dicha estructura presente una flotabilidad eficiente.

Figura 7 Catamarán de VNAS

- Desarrollo de un sistema de control de motores: esta tarea ha sido asignada para la escuela de Industriales. El objetivo es la interactuación con los motores del barco para poder controlarlos a distancia a través de un software.
- Desarrollo de la comunicación entre el servidor y el barco: para conseguir un vehículo autónomo se necesita desarrollar un enlace mediante el cual exista comunicación entre el barco y el equipo central. Esta tarea ha sido asignada para la escuela de Telecomunicación con el objetivo de implementar un sistema de comunicación a través de la red 3G.
- Desarrollo de un software para la adquisición de datos del barco y representarlos en tiempo real: hace falta disponer de un software que permita al usuario evaluar los datos recogidos por los sensores en el barco, procesar y dar órdenes al propio barco para que pueda realizar una función. Para ello, se ha asignado esta tarea también para la escuela de Telecomunicación puesto que requiere el diseño de una plataforma a través de lenguajes de programación como Java, PHP, JavaScript, etc.
- Compra de los componentes necesarios que irán montados en el catamarán como los sensores GPS, Compás, anemómetro, sensor Smart Triducer, conversores SeaTalk, Raspberry PI, motores, baterías, tarjeta MicroSD, tarjeta SIM y Modem 3G. La descripción de cada componente se muestra en el Anexo 1.

3.2 Esquema global del sistema

En la Figura 8 podemos visualizar un plano completo del prototipo. En un principio, se ha propuesto la estructuración de cada componente como el que aparece en el plano. Por un lado, observamos las cuatro zonas en los dos cascos donde irán las baterías para la alimentación de la embarcación. A su vez, sobre una superficie, se montará la parte electrónica que consiste en el sistema de sensores junto a otros componentes. También se montarán los dos motores que se van a conectar con la parte electrónica para recibir órdenes y realizar acciones de propulsiones. La superficie tiene que estar diseñada tal que pueda soportar un peso considerable capaz de aguantar tanto el sistema electrónico como la presencia de una persona.

La tarea de la creación de los dos cascos y la superficie que irá encima de ellos ha sido asignada para la escuela de Navales. De la implementación de un sistema de control sobre los dos motores situados en la superficie, se encarga la escuela de Industriales. Finalmente, la tarea de la parte electrónica es para la escuela de Telecomunicación.

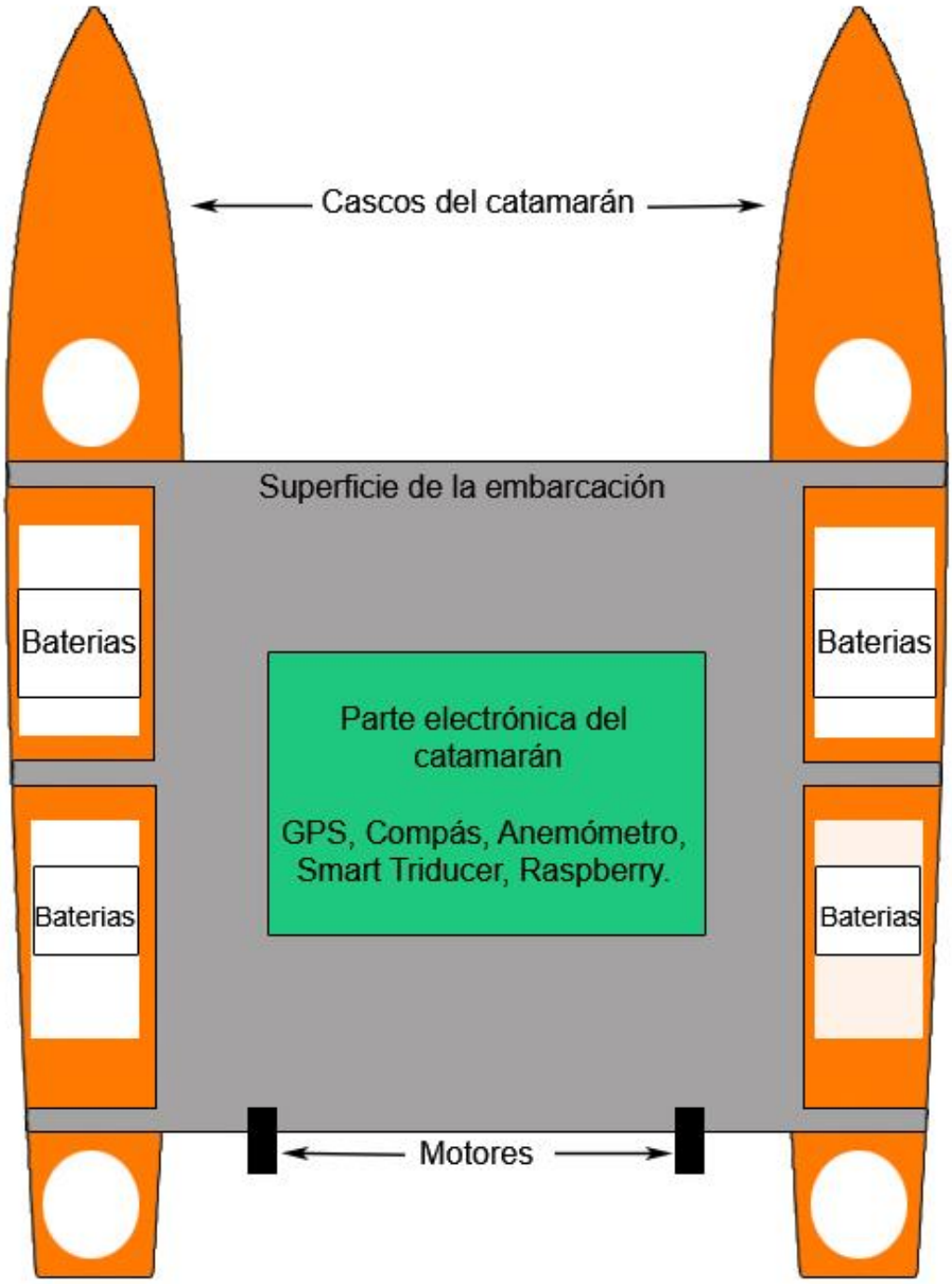

Figura 8 Esquema completo de la embarcación

Por otra parte, en la Figura 9, tenemos el esquema de sensores y la conexión que tienen con los demás componentes. Esta tarea fue llevada a cabo por los representantes de la escuela de Telecomunicación, los alumnos Alejandro González Redel y yo. Para ello, procedimos a realizar el montaje del sistema de sensores en el laboratorio SICOMO siguiendo paso a paso las explicaciones dadas por el director del proyecto VAMAR, Humberto.

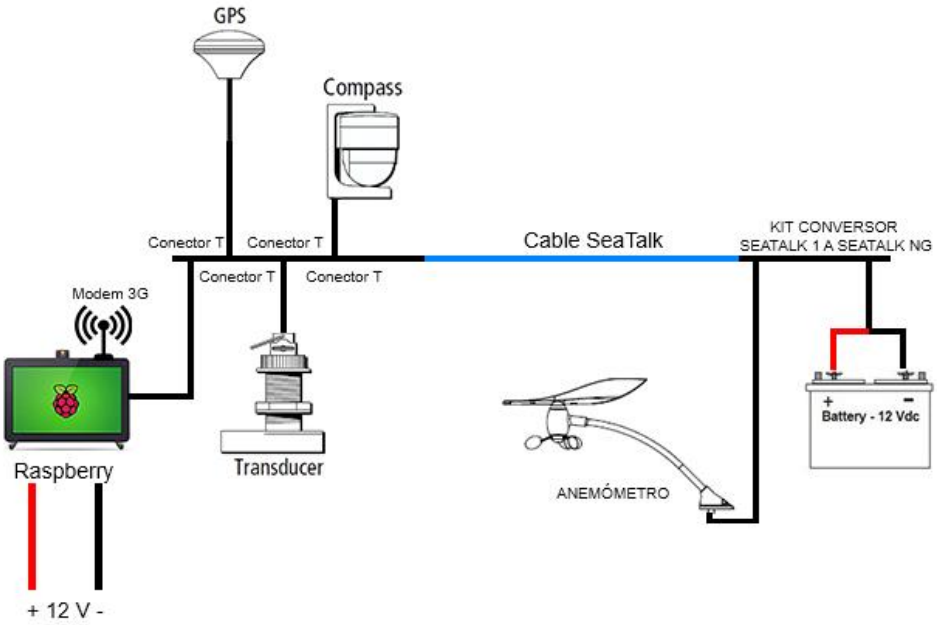

Figura 9 Esquema de la parte electrónica

De la figura anterior, podemos observar que tenemos tres sensores que son: sensor de posicionamiento GPS, sensor Compás encargado de indicarnos la orientación e inclinación del barco y el sensor Smart Triducer encargado de una multitud de funciones como la de medir la velocidad, la profundidad y la temperatura del catamarán. Estos sensores se conectan mediante conectores NMEA 2000 [12] a un conector en forma de T, LOWRANCE N2K-T-RD. A su vez, este conector se conecta por un lado a la Raspberry la cual hace de cerebro y, por otro lado, a un conversor SeaTalk mediante un cable SeaTalk. La función de éste es por un lado conectarse a la fuente de alimentación, baterías, que se encargan de alimentar todo el equipo y por otro, conectarse al anemómetro.

Dado que aún estamos en pleno desarrollo, el catamarán aún está en proceso de diseño. Por esa razón, realizamos una primera prueba de montaje en el laboratorio con el fin de desarrollar nuestra parte del proyecto.

3.3 Desarrollo de la comunicación entre servidor y Raspberry PI

Una vez realizado el montaje tal y como lo hemos explicado en el apartado anterior, pasamos a estudiar con profundidad sobre el cómo realizar la comunicación entre una estación base y la Raspberry. En principio teníamos varias alternativas, aunque el principal objetivo era conseguir establecer la comunicación mediante la red 3G. Por ello

estudiamos sobre como programar en una Raspberry puesto que no teníamos conocimientos previos sobre ello.

Para ello, el primer paso ha sido configurar la red WIFI en la Raspberry para que ésta pueda conectarse a Internet ya que se necesita el acceso a Internet para poder descargar e instalar una serie de paquetes. Esta tarea está explicada de manera detallada en Anexo 2.

Una vez conectados a Internet a través de WIFI, empezamos a investigar sobre el cómo configurar el Modem 3G para que la Raspberry pueda acceder a Internet a través de la red 3G. Para ello seguimos varias explicaciones en las cuales explican de manera detallada como se hace esta tarea. Los pasos se explican en el Anexo 3.

Después de conseguir desarrollar un enlace de comunicación a través de la red 3G entre el ordenador y la Raspberry el siguiente paso fue realizar la transmisión y recepción de datos.

3.4 Transmisión y recepción de datos entre estación base y Raspberry

A nivel de software, cabe destacar que en la Raspberry tenemos una aplicación llamada MyPlotter, programada en Java, cuya función es la de recoger los datos de los sensores y enviarlos por el puerto 1703 a través del protocolo UDP. [13]

Figura 10 Aplicación MyPlotter

También se encarga de mostrar por la pantalla de la Raspberry la información en tiempo real de los sensores. Podemos observar en la Figura 11, un mapa en el cual se representa la posición del barco, la velocidad de éste y la de viento, la profundidad y la orientación entre otros datos.

Figura 11 Aplicación MyPlotter con todas sus funciones

Puesto que la aplicación ya se encarga de enviar de forma automática los datos de los sensores por el puerto 1703, nosotros hemos realizado un pequeño programa en la Raspberry tal que su función es la de redireccionar todo el tráfico transmitido por el puerto comentado anterior a la dirección pública del router de laboratorio SICOMO.

El programa que se encarga de la anterior función que hemos programado en Java. En el Anexo 4, viene tanto el código del programa como la explicación de éste.

4. Lenguajes de programación y herramientas usadas.

En este apartado nos vamos a centrar en el desarrollo de la aplicación web en la cual se representarán los datos de los sensores en tiempo real. Para ello, se ha necesitado el estudio de varios lenguajes de programación que nos van a ser útiles para llevar a cabo dicho objetivo. A continuación, vamos a explicar el uso de cada lenguaje.

4.1 Lenguajes de programación

4.1.1 JAVA

Java es un lenguaje de programación orientado a objetos que permite el desarrollo de una variedad de aplicaciones. En la actualidad, es un lenguaje muy extendido puesto que presenta muchas ventajas con respecto a otros lenguajes de programación:

Una de las ventajas que tiene es la orientación a objetos. Java es un lenguaje que se basa en la interacción de unas estructuras llamadas objetos para la implementación de las aplicaciones. Por otra parte, presenta la ventaja de la concurrencia. Es una propiedad que admite la ejecución e integración de diversas tareas de forma simultánea.

En este proyecto se ha utilizado Java puesto que tenemos conocimientos previos de este lenguaje de programación. El uso que se le ha dado ha sido sobre todo para desarrollar en el servidor la aplicación encargada de recibir los datos de los sensores enviados por la Raspberry. [16]

Figura 12 Logo de Java

4.1.2 MySQL, SQL y PHP

MySQL es la herramienta que sirve para gestionar bases de datos. SQL es el lenguaje que nos permite crear, modificar y hacer cualquier operación en una base de datos.

PHP es un lenguaje de programación de lado del servidor diseñado principalmente para el desarrollo de la web. La relación entre PHP y SQL es que ambos lenguajes se pueden combinar con MySQL para trabajar con bases de datos. Esta relación nos permite interactuar con las bases de datos para la gestión y la extracción de los elementos con facilidad a nivel de desarrollo web. [17]

Figura 13 PHP + MySQL

4.1.3 JavaScript, HTML y CSS

JavaScript es un lenguaje de programación de lado del cliente orientado a objetos que permite a los programadores crear aplicaciones dinámicas. El uso de este lenguaje es por el hecho de que es mejor en comparación con otros lenguajes de programación. Principalmente, se utiliza este lenguaje para la implementación de la API de Google Maps en nuestra página web. Esto es porque es necesario un mapa en nuestra aplicación donde podemos visualizar la posición del catamarán de forma precisa.

HTML (HyperText Markup Language, es decir, Lenguaje de Marcas de Hipertexto) es un lenguaje de marcado que se utiliza para el desarrollo de páginas de Internet. Es necesario es lenguaje de programación dado que vamos a crear una web que necesita de unas estructuras y marcos donde tendremos contenidos.

CSS es un lenguaje de reglas en cascada que usamos para aplicar un estilo a nuestro contenido en HTML, por ejemplo, colocando colores de fondo, fuentes y marginando nuestro contenido en múltiples columnas. [18]

Figura 14 HTML, CSS y JavaScript

4.2 Herramientas de desarrollo

Para la aplicación de los anteriores lenguajes de programación se ha hecho uso de varios programas.

4.2.1 NetBeans

NetBeans es un entorno de desarrollo integrado (IDE) de código abierto que permite programar y ejecutar cualquier tipo de aplicación. Es un programa estándar desarrollado en Java que se adapta a cualquier lenguaje de programación como C/C++, HTML o PHP. [19]

Se ha utilizado este programa por el mero hecho de que presenta herramientas que simplifican el proceso de la creación de aplicaciones. La versión que se ha usado es NetBeans IDE 8.2. En la Figura 15, tenemos un ejemplo del entorno de NetBeans.

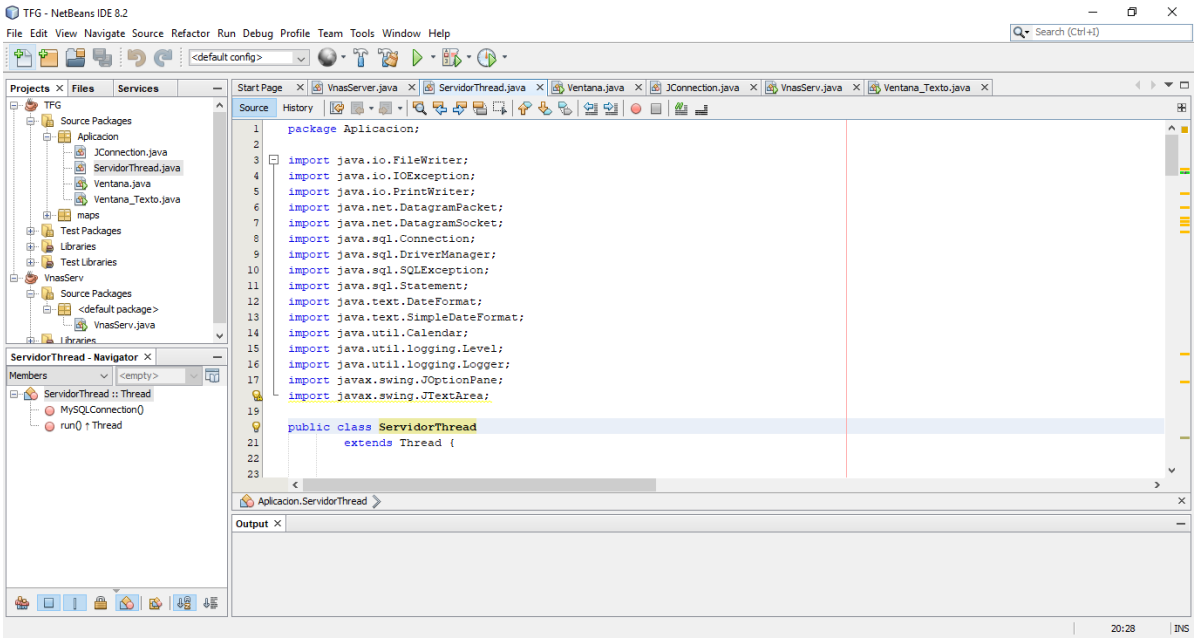

Figura 15 Entorno de NetBeans

4.2.2 XAMPP

XAMPP es un paquete de software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP y Perl. [20]

Figura 16 Apache, MySQL y PHP

Su principal ventaja es la facilidad que proporciona a la hora de su instalación y configuración. Además, permite tener un servidor de Apache donde se puede ejecutar scripts de PHP y almacenar una base de datos en MySQL. Esto se hace gracias a que XAMPP crea el entorno phpMyAdmin donde se realizan todas las funcionalidades como las comentadas anteriormente. En la Figura 17 podemos visualizar el entorno de XAMPP.

En este proyecto se ha usado XAMPP puesto que es la herramienta adecuada para crear un servidor web, a parte que nos proporciona la posibilidad de usar los lenguajes de programación como PHP en este caso, para la interactuación entre base de datos, y la aplicación web.

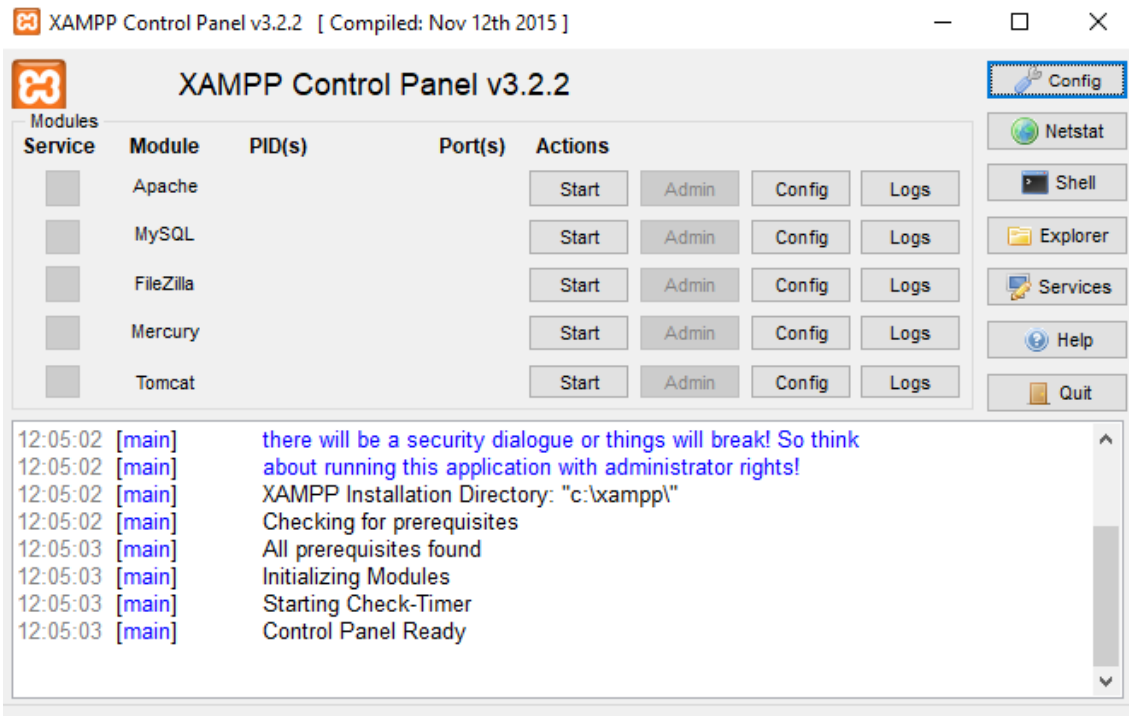

Figura 17 Entorno de XAMPP

4.2.3 Brackets

Brackets es un editor de texto moderno, gratuito y de código abierto, desarrollado por Adobe pensando principalmente en los programadores web que utilizan este tipo de editores de texto para programas en lenguajes como HTML, CSS y JavaScript. Este IDE cuenta con una serie de funciones y características que facilitan la tarea de programar. Entre otras, algunas de estas características son el resalte de sintaxis, la ayuda en la programación y, algo muy interesante en la programación web, el poder ver en tiempo real nuestra página web en una ventana a parte. [21]

Figura 18 Logo del programa Brackets

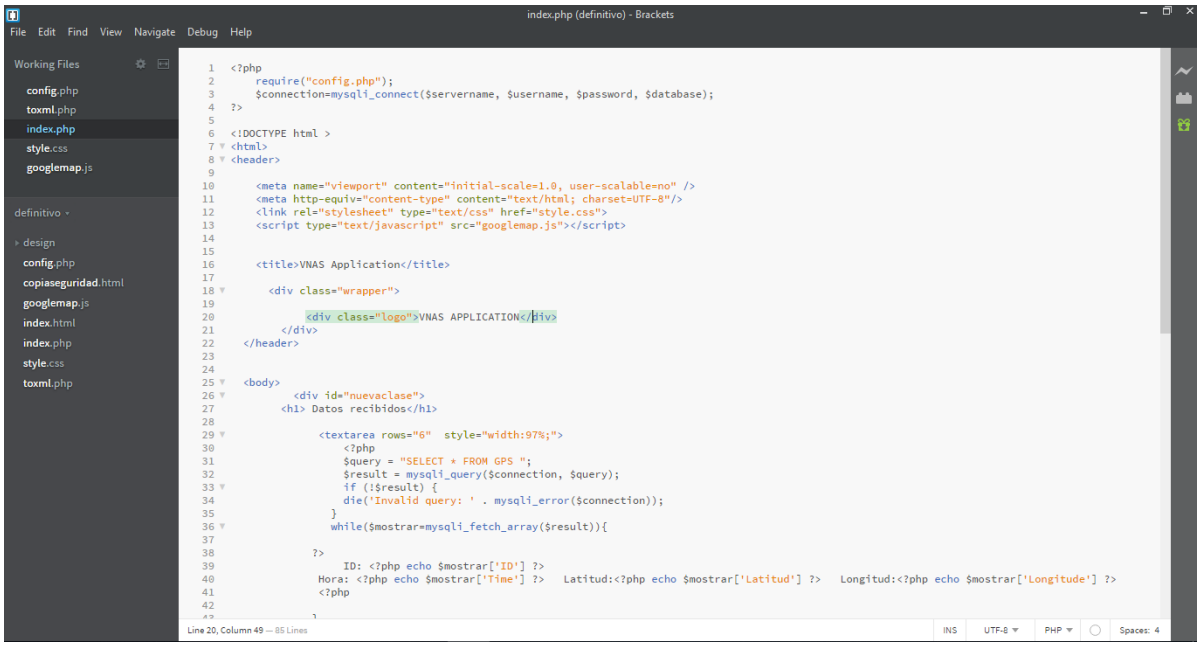

Figura 19 Entorno de Brackets

5. Desarrollo de la aplicación web

A lo largo de este apartado, vamos a explicar paso a paso lo que hemos hecho para conseguir el desarrollo de la aplicación web en la cual vamos a poder visualizar en tiempo real los datos de los sensores situados en el catamarán. Esto se ha conseguido gracias al uso de cada una de las herramientas y lenguajes de programación mencionados en el apartado anterior. A continuación, podemos visualizar un diagrama de bloques completo, Figura 20 y Figura 21, que explica de forma gráfica los pasos para conseguir el objetivo principal de este TFE.

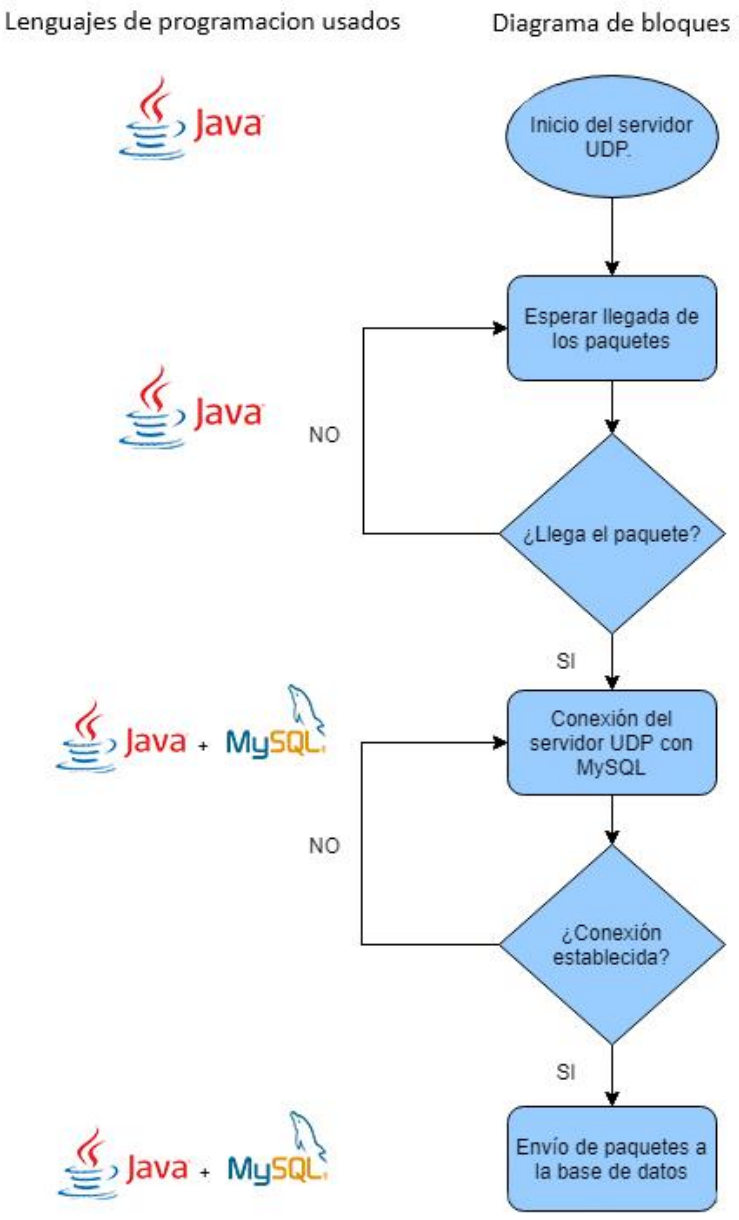

Figura 20 Diagrama de bloques I

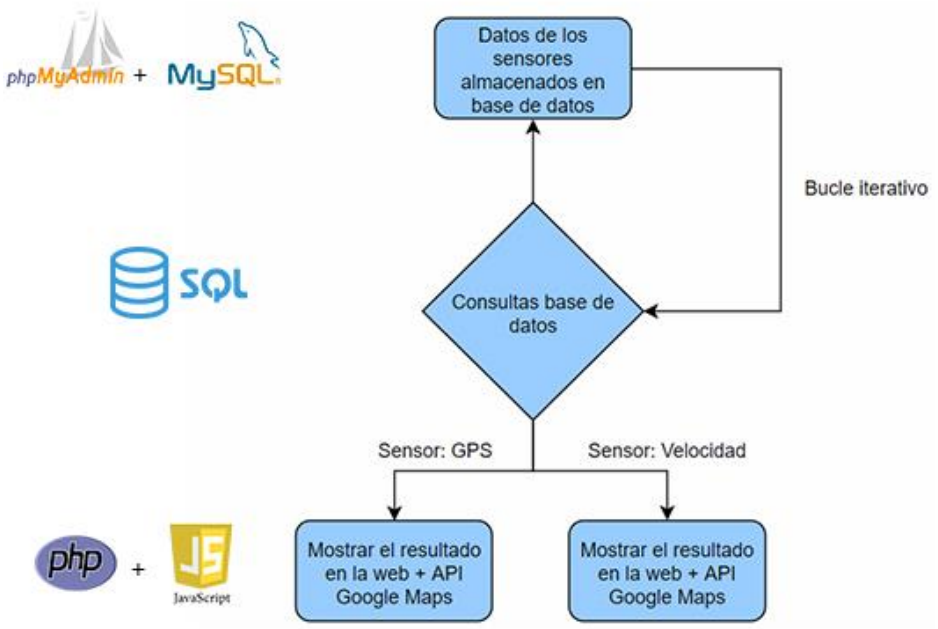

Figura 21 Diagrama de Bloques II

5.1 Servidor UDP para la escucha del puerto

En primer lugar, necesitamos una herramienta en el servidor que nos permita recibir los paquetes enviados por el cliente. Ante esta exigencia, se ha reusado la aplicación desarrollada en el proyecto MotoStudent la cual realiza la misma función que la que queremos llevar a cabo. [22]

La aplicación Servidor UDP escucha constantemente el puerto que se le introduce. El protocolo TCP es un protocolo síncrono el cual se basa en una serie de fases como establecimiento y finalización de enlace, además de control y confirmación de la recepción de paquetes al emisor. En cambio, el protocolo UDP es un protocolo asíncrono que no requiere de establecimiento de enlace ni de control sobre el envío de paquetes. Como primamos la velocidad y el tiempo menor posible, optamos por el protocolo UDP.

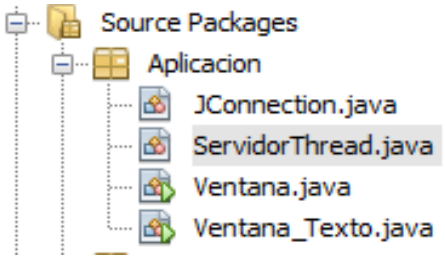

Figura 22 Clases del Servidor UDP

En la Figura 22, podemos visualizar la composición de la aplicación Servidor UDP programada en Java.

5.2 Creación de la base de datos

El siguiente paso es la creación de la base de datos para almacenar los paquetes recibidos mediante el protocolo UDP. El proceso de la creación de la base de datos viene explicado en el Anexo 5. En principio vamos a visualizar la información de los sensores de GPS y velocidad, aunque el proceso es el mismo para cada sensor.

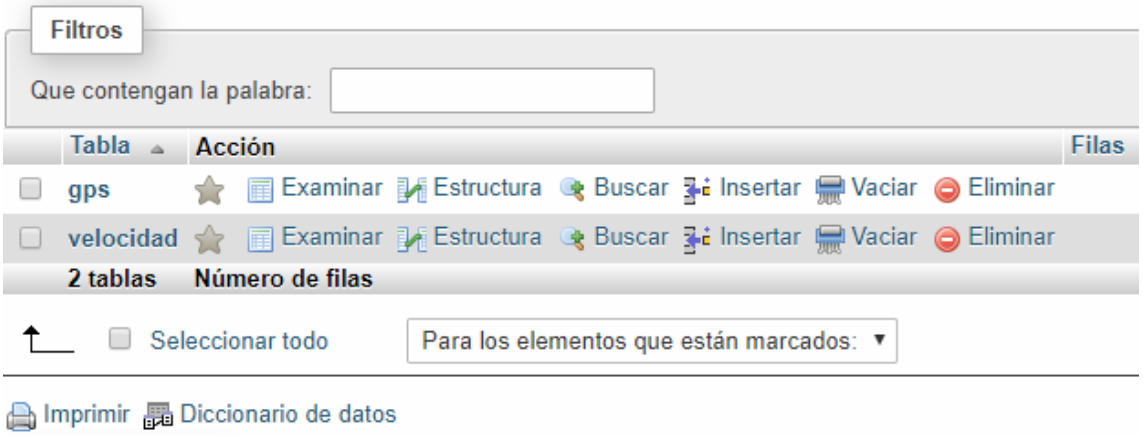

Figura 23 Tablas creadas para los sensores GPS y Velocidad

5.3 Conexión del Servidor UDP con Base de datos MySQL

El objetivo de este apartado es la conexión del Servidor UDP con MySQL para enlazarlo con la base de datos que hemos creado. Primeramente, tenemos que instalar el driver para JAVA de MySQL ya que mediante el cual conseguimos vincular la base de datos con Servidor UDP. Posteriormente, abrimos el paquete de la aplicación Servidor UDP en Java y añadimos la clase JConnection. [23]

```
public class JConnection {

        public static Connection ConnectDB(){
            try{
                Class.forName("com.mysql.jdbc.Driver");
                Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost/barco", "root", "");
                System.out.println("Conectado");
                Statement myStmt = conn.createStatement();
                return conn;
            } catch(Exception e){

                JOptionPane.showMessageDialog(null,e);
                return null;
            }
        }
```

Códigos 1 Clase JConnection

5.4 Envío de los paquetes del Servidor UDP a la base de datos

Cabe destacar que los datos recibidos vienen en un formato denominado NMEA0183. [15] En el siguiente cuadro, podemos visualizar el ejemplo de una línea enviada por el sensor.

```
$NKGLL,3851.39695,N,09447.98799,W,161943,V,N*50
```

Según el protocolo NMEA0183, esta línea corresponde al sensor de GPS.

GLL Geographic Position – Latitude/Longitude

```
       1       2 3       4 5         6 7
       |       | |       | |         | |
$--GLL,llll.ll,a,yyyyy.yy,a,hhmmss.ss,A*hh

1) Latitude
2) N or S (North or South)
3) Longitude
4) E or W (East or West)
5) Time (UTC)
6) Status A - Data Valid, V - Data Invalid
7) Checksum
```

Figura 24 Interpretación de la línea de GPS en NMEA0183

Con la gráfica anterior extraída del manual del protocolo NMEA0183, podemos traducir la línea en la siguiente información:

- Las dos primeras letras “NK” no se tienen en cuenta.
- “GLL” indica que se trata del sensor de GPS.

1) Latitud: “3851.39695”.
2) Norte.
3) Longitud: “09447.98799”.
4) W de oeste.
5) La hora a la que se ha hecho la medida: 16:19:43.
6) Dato valido o inválido.
7) Checksum.

Una vez entendido la información de las líneas NMEA0183, pasamos a la tarea más importante del apartado. Hay que tener en cuenta que la clase más importante es ServidorThread. Ésta se encarga de la gestión de la llegada de los paquetes UDP, extraer la información, encapsularla y enviarla a la base de datos. Para ello el objetivo es crear variables tal que a cada variable se le asigna un fragmento que pertenece a un dato del sensor, Figura 24.

En Códigos 2, ServidorThread establece la comunicación con la base de datos que hemos creado llamada “barco”. También se declaran las variables necesarias.

```
try {
            DatagramSocket serverSocket = new
DatagramSocket(Ventana_Texto.hostPort);
            byte[] receiveData = new byte['?'];
            String str = null;
            FileWriter fichero = null;
            PrintWriter pw = null;


            do {
                str = null;
                DatagramPacket receivePacket = new
DatagramPacket(receiveData, receiveData.length);
                serverSocket.receive(receivePacket);
                str = new String(receiveData, 0,
receivePacket.getLength());
                char[] cadena = str.toCharArray();
                //Variables para GPS
                String latitud;
                char NorS;
                String longitude;
                String Time;
                String latitud_dd;
                String latitud_mm;
                String longitude_ddd;
                String longitude_mm;
                int lat_dd;
                int long_ddd;
                float lat_mm;
                float long_mm;
                float lat_resultado;
                float long_resultado;
                String lat;
                String lng;
                char EorW;
                char Status;
                //Variables para velocidad
                String degreestrue;
                char TorF;
                String degreesmagnetic;
                char Magnetic;
                String Knots;
                char Nknots;
                String Kilometers;
                char K;
```

Códigos 2 Establecimiento de conexión con MySQL y declaración de Variables

Las variables anteriores creadas son para los sensores GPS y Velocidad. En **Códigos 3**, el programa se encarga de almacenar los datos de la línea NMEA0183 recibida correspondiente al sensor GPS en las variables creadas para dicho sensor. En **Códigos 4**, realiza la misma función, almacena datos de la línea NMEA0183 correspondiente al sensor Velocidad en las variables creadas para dicho sensor.

En las dos figuras de abajo, se visualiza la información que se puede extraer de las líneas NMEA0183 correspondientes a los sensores de GPS y Velocidad. Analizando las posiciones [3], [4] y [5] de la línea NMEA0183 recibida, podemos deducir de que sensor se trata. Para ello la variable **cadena** en la que se almacena constantemente líneas NMEA0183, se comprueba en las posiciones mencionadas anteriormente. Una vez identificado el tipo de sensor, en cada variable se almacena un fragmento de la línea NMEA0183.

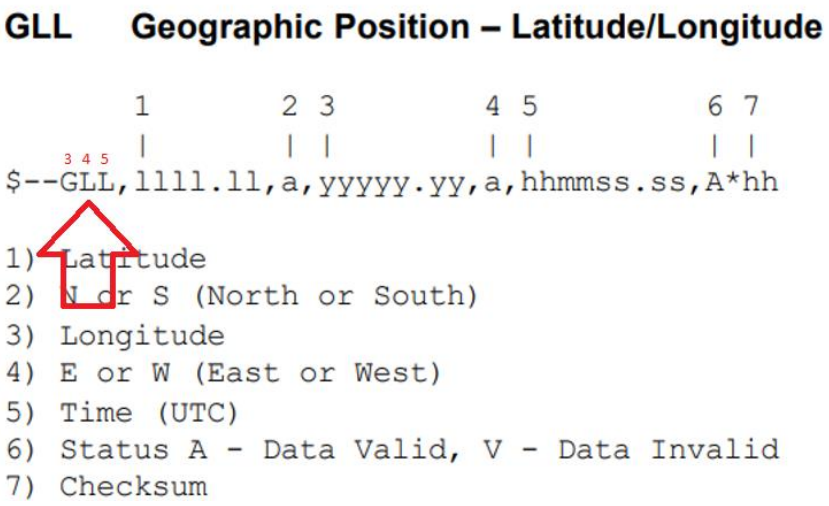

Figura 25 GLL corresponde al sensor GPS

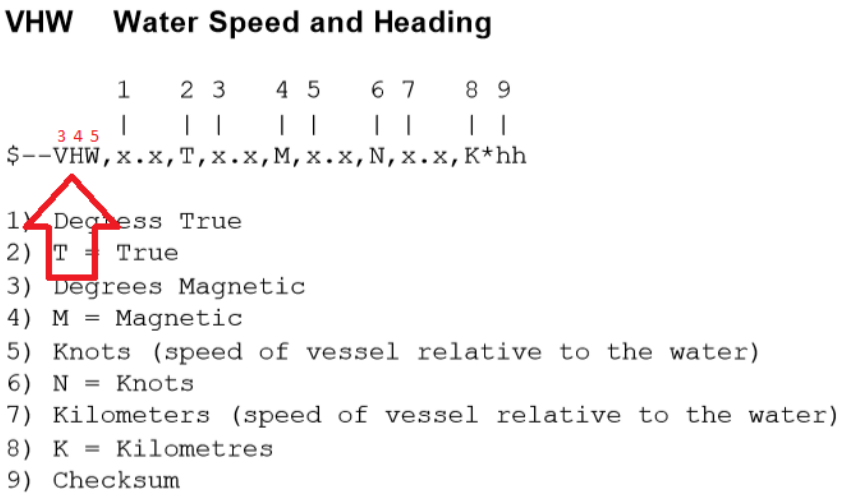

Figura 26 VHW corresponde al sensor Velocidad

```
if (cadena[3] == 'G' & cadena[4] == 'L' & cadena[5] == 'L') {
                    latitud = str.substring(7,17);
                    NorS = str.charAt(18);
                    longitude = str.substring(20,31);
                    EorW = str.charAt(32);
                    Time =
str.substring(34,36)+":"+str.substring(36,38)+":"+str.substring(38,40);
                    Status = str.charAt(41);
                    // Latitud
                    latitud_dd = latitud.substring(0,2);
                    latitud_mm = latitud.substring(3,9);
                    lat_dd = Integer.parseInt(latitud_dd);
                    lat_mm = Float.parseFloat(latitud_mm);
                    lat_resultado = (float)lat_dd + (float)lat_mm/60;
                    lat = String.valueOf(lat_resultado);
                     // Longitude
                    longitude_ddd = longitude.substring(0,3);
                    longitude_mm = longitude.substring(4,11);
                    long_ddd = Integer.parseInt(longitude_ddd);
                    long_mm = Float.parseFloat(longitude_mm);
                    long_resultado = long_ddd + (float)long_mm/60;
                    lng = String.valueOf(long_resultado);
                    System.out.println(lat+" "+NorS+" "+lng+" "+EorW+"
"+Time+" "+Status);
                    String sentencia = "Insert INTO GPS
(Latitud,NorS,Longitude,EorW,Time,Status) VALUES
('"+lat+"','"+NorS+"','"+lng+"','"+EorW+"','"+Time+"','"+Status+"')";
                    Statement st = Conexion.createStatement();
                    st.executeUpdate(sentencia);
                     }
```

Códigos 3 Programa para asignar a las variables creadas para el sensor GPS, los datos de la línea NMEA0183 correspondiente a dicho sensor

```
else{
                    if (cadena[3] == 'V' & cadena[4] == 'H' & cadena[5] ==
'W'){
                    degreestrue = str.substring(7,12);
                    TorF = str.charAt(13);
                    degreesmagnetic = str.substring(15,20);
                    Magnetic = str.charAt(21);
                    Knots = str.substring(23,29);
                    Nknots = str.charAt(30);
                    Kilometers = str.substring(32,36);
                    K = str.charAt(37);
                    System.out.println("Degrees True: " + degreestrue + " "
+ "True/False: " + TorF + " " + "Degrees Magnetic: " + degreesmagnetic +" "
+ "Magnetic: " + Magnetic + " " + "Knots: " + Knots + " " + "N_knots: " +
Nknots + " " + "Kilometers: " + Kilometers + " " + "K_kilometers: " + K);
                    String sentencia2 = "Insert INTO velocidad
(degreestrue,TorF,degreesmagnetic,Magnetic,Knots,Nknots,Kilometers,K)
VALUES
('"+degreestrue+"','"+TorF+"','"+degreesmagnetic+"','"+Magnetic+"','"+Knots
+"','"+Nknots+"','"+Kilometers+"','"+K+"')";
                    Statement st2 = Conexion.createStatement();
                    st2.executeUpdate(sentencia2);
                    }
                }
```

Códigos 4 Programa para asignar a las variables creadas para el sensor Velocidad, los datos de la línea NMEA0183 correspondiente a dicho sensor

Finalmente cabe destacar del código anterior una línea fundamental:

```
String sentencia = "Insert INTO GPS
(Latitud,NorS,Longitude,EorW,Time,Status) VALUES
('"+lat+"','"+NorS+"','"+lng+"','"+EorW+"','"+Time+"','"+Status+"')"
;

Statement st = Conexion.createStatement();
st.executeUpdate(sentencia);
```

Códigos 5 Programa para asignar a las variables creadas para el sensor GPS, los datos de la línea NMEA0183 correspondiente a dicho sensor

La variable **sentencia** corresponde a la sentencia **INSERT INTO** que utiliza SQL para insertar nuevas filas en las tablas en la base de datos. Esto nos interesa porque buscamos enviar los datos recibidos a las tablas de las bases de datos. En Java podemos realizar consultas con SQL, así como enviar o leer datos de la base de datos.

Como prueba, entramos a la base de datos "barco", y visualizamos la tabla GPS. Podemos observar en la Figura 27, como la tabla GPS se llena automáticamente con los datos de las líneas NMEA0183 que corresponden al sensor de GPS.

+ Opciones

	ID	Latitud	NorS	Longitude	EorW	Time	Status
Editar Copiar Borrar	1	37.000000	N	94.133133	W	10:04:02	V
Editar Copiar Borrar	2	37.000000	N	94.133133	W	10:04:02	V
Editar Copiar Borrar	3	37.000000	N	94.133133	W	10:04:02	V
Editar Copiar Borrar	4	37.000000	N	94.133133	W	10:04:02	V
Editar Copiar Borrar	5	37.000000	N	94.133133	W	10:04:02	V
Editar Copiar Borrar	6	37.000000	N	94.133133	W	10:04:02	V
Editar Copiar Borrar	7	38.023281	N	94.133133	W	10:04:02	V
Editar Copiar Borrar	8	38.023281	N	94.133133	W	10:04:02	V

Figura 27 Tabla GPS con los datos de las líneas NMEA0183 correspondientes a ésta

Como segunda prueba, abrimos la tabla Velocidad y exitosamente conseguimos ver los datos de las líneas NMEA0183 que corresponden al sensor de la velocidad.

+ Opciones

degreestrue	TorF	degreesmagnetic	Magnetic	Knots	Nknots	Kilometers	K
175.8	T	175.8	M	0.0	N	0.0	K
175.8	T	175.8	M	0.0	N	0.0	K
195.4	T	195.4	M	0.0	N	0.0	K
195.4	T	195.4	M	0.0	N	0.0	K
195.4	T	195.4	M	0.0	N	0.0	K
195.4	T	195.4	M	0.0	N	0.0	K
195.4	T	195.4	M	0.0	N	0.0	K

Figura 28 Tabla Velocidad con los datos de las líneas NMEA0183 correspondientes a ésta

5.5 Representación de los datos en una página web

Después de conseguir el almacenamiento de las líneas NMEA0183 enviadas por la Raspberry a nuestro servidor en la base de datos, podemos pasar a la parte del diseño de la plataforma web que represente tanto de forma numérica como gráfica los datos de los sensores GPS y Velocidad. Para ello vamos a usar el programa Brackets que nos permite programar en PHP, HTML, CSS y JavaScript, una página web.

Hasta ahora estamos trabajando con los datos de los sensores GPS y Velocidad para diseñar una primera aplicación web con el fin de hacer una prueba real. Los datos del sensor de Velocidad se van a representar de forma básica mientras el sensor GPS, vamos a usar una API de Google Maps en la cual vamos a poder visualizar el posicionamiento del barco en un mapa. [24]

Esta API combina MySQL y PHP con Google Maps de tal manera que nos permite hacer consultas a la tabla de GPS para que extraer los datos de latitud y longitud. A través de estos dos datos, podemos visualizar en el mapa la posición concreta del barco.

Los ficheros necesarios que hemos creado para diseñar la página web son los siguientes:

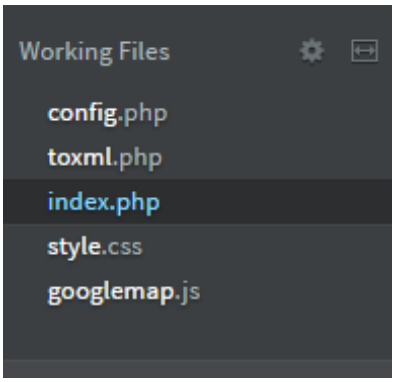

Figura 29 Clases creadas para el diseño de la web

- La clase **config.php** contiene los datos que se necesitan para enlazar nuestra página web con la base de datos MySQL. El nombre del servidor es "localhost", la base de datos que hemos creado es "barco", el usuario es "root" por defecto y no tiene contraseña por lo que no se pone nada

```php
<?php
        $servername = "localhost";
        $database = "barco";
        $username = "root";
        $password = "";
?>
```

Códigos 6 Variables para acceso a BBDD

Las clases **toxml.php** y **googlemaps.js** son las que vienen con la API de Google Maps. En la clase **toxml.php** podemos observar cómo se hacen las consultas a la tabla GPS de nuestra base de datos. Además, la función principal de **toxml.php** es realizar la conversión de los datos consultados en la tabla de la base de datos en formato XML.

```php
<?php
        require("config.php");
        // Start XML file, create parent node
        $dom = new DOMDocument("1.0");
        $node = $dom->createElement("markers");
        $parnode = $dom->appendChild($node);

        // Opens a connection to a MySQL server

        $connection=mysqli_connect ($servername, $username, $password);
        if (!$connection) {  die('Not connected : ' . mysql_error());}
        $db_selected = mysqli_select_db($connection, $database);
        if (!$db_selected) {
          die ('Can\'t use db : ' . mysql_error());
        }
        // Select all the rows in the markers table
        $query = "SELECT * FROM gps WHERE 1";
        $result = mysqli_query($connection, $query);
        if (!$result) {
          die('Invalid query: ' . mysql_error());
        }
        header("Content-type: text/xml");
        // Iterate through the rows, adding XML nodes for each
        while ($row = @mysqli_fetch_assoc($result)){
          // Add to XML document node
          $node = $dom->createElement("marker");
          $newnode = $parnode->appendChild($node);
          $newnode->setAttribute("latitud", $row['Latitud']);
          $newnode->setAttribute("longitude", $row['Longitude']);
        }
        echo $dom->saveXML();

?>
```

Códigos 7 Fichero toxml.php de la API de Google Maps

El código de abajo corresponde a la clase **googlemaps.js** programado en JavaScript. El programa hace la función de la lectura de los datos convertidos a XML para usarlos en el posicionamiento del barco en el mapa.

```
function initMap() {
            var cartagena = {lat: 37.595182, lng:  -0.982121};
              map = new google.maps.Map(document.getElementById('map'), {
            zoom: 17,
            center: cartagena
    });
            var marker = new google.maps.Marker({
            position: cartagena,
            map: map
    });
          // Change this depending on the name of your PHP or XML file
          downloadUrl('toxml.php', function(data) {
            var xml = data.responseXML;
            var markers =
xml.documentElement.getElementsByTagName('marker');
            Array.prototype.forEach.call(markers, function(markerElem) {
              var point = new google.maps.LatLng(
                 parseFloat(markerElem.getAttribute('latitud')),
                 parseFloat(markerElem.getAttribute('longitude')));
              var icono = {
                 path: google.maps.SymbolPath.CIRCLE,
                 strokeColor: '#da1823',
                 fillColor: '#da1823',
                 fillOpacity: .9,
                 strokeWeight: 1,
                 scale: 6,
              };
            var marker = new google.maps.Marker({
            position: point,
            icon: icono,
            map: map,
                });
            map.setCenter(point);
            });
          });
        }
      function downloadUrl(url, callback) {
        var request = window.ActiveXObject ?
            new ActiveXObject('Microsoft.XMLHTTP') :
            new XMLHttpRequest;

        request.onreadystatechange = function() {
          if (request.readyState == 4) {
            request.onreadystatechange = doNothing;
            callback(request, request.status);
          }
        };
        request.open('GET', url, true);
        request.send(null);
      }
      function doNothing() {}
```

Códigos 8 Clase de googlemaps.js de la API de Google Maps

- La clase **style.css** está programada en CSS y contiene el código que sirve para darle diseño a la web como definir el tipo de fuente, tamaño de fuente, el aspecto de los marcos o cuadros, etc.

```
html, body {
    height: 100%;
    margin: 0;
    font-family: sans-serif;
      }
header {
    width:100%;
    overflow:hidden;
    background-color: #0a78d5;
    height: 80px;
}
.wrapper {
    background: url("design/icon.png") no-repeat;
    width:100%;
    max-width:1000px;
    margin: auto;
    font-size: 50px;
    text-align: center;
    font-family: sans-serif;
    color: white;
    padding: 8px;
}
#map {
    float: left;
    height: 70%;
    width: 47%;
    border: 1px solid black;

      }
h1{
    margin-left: auto;
    margin-right: 25px;
    text-align: center;
}
h4{
    text-align: center;
}
#nuevaclase{
    margin:auto;
    float: left;
    height:500px;
    width: 50%;
    }
```

Códigos 9 La clase style.css que da el diseño a la web

- Finalmente, la clase **index.php** es la clase principal que programamos en HTML para crear la página web y en la que se reúnen todos los resultados que nos proporcionan las otras clases.

```
<?php
    require("config.php");
    $connection=mysqli_connect($servername, $username, $password,
$database);
?>
<!DOCTYPE html >
<html>
<header>
    <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
    <meta http-equiv="content-type" content="text/html; charset=UTF-8"/>
    <link rel="stylesheet" type="text/css" href="style.css">
    <script type="text/javascript" src="googlemap.js"></script>
    <title>VNAS Application</title>
      <div class="wrapper">
                          <div class="logo">VNAS APPLICATION</div>
                 </div>
  </header>
  <body>
          <div id="nuevaclase">
        <h1> Datos recibidos</h1>
              <textarea rows="6"  style="width:97%;">
                  <?php
                  $query = "SELECT * FROM GPS ";
                  $result = mysqli_query($connection, $query);
                  if (!$result) {
                  die('Invalid query: ' . mysqli_error($connection));
                }
                while($mostrar=mysqli_fetch_array($result)){
             ?>
                  ID: <?php echo $mostrar['ID'] ?>
              Hora: <?php echo $mostrar['Time'] ?>   Latitud:<?php echo
$mostrar['Latitud'] ?>   Longitud:<?php echo $mostrar['Longitude'] ?>
              <?php
             }
```

Códigos 10 Clase principal de la web I

```
                ?>
                    </textarea >
            <textarea rows="6" style="width:97%;">
            <?php
                $query = "SELECT * FROM Velocidad ";
                $result = mysqli_query($connection, $query);
                if (!$result) {
                die('Invalid query: ' . mysqli_error($connection));
              }
              while($mostrar=mysqli_fetch_array($result)){
           ?>
            Degrees True: <?php echo $mostrar['degreestrue'] ?>
            <?php
           }
                ?>
            </textarea>
       </div>

      <h1> Posicionamiento del barco</h1>
    <div id="map">

         <script async defer
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyDuJKn10qEA8h
i7E61F6f7-Oe2k352WnGc&callback=initMap">
   </script>
        </div>
     </body>
</html>

<h1> Posicionamiento del barco</h1>
    <div id="map">

         <script async defer
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyDuJKn10qEA8hi7E61F6f
7-Oe2k352WnGc&callback=initMap">
   </script>
        </div>
     </body>
</html>
```

Códigos 11 Clase principal de la web II

Con las clases definidas anteriormente, lo que buscamos es crear una página web con PHP, HTML y CSS, en la cual podemos representar datos de los sensores almacenados en la base de datos mediante consultas SQL. A su vez, buscamos representar la posición del barco a través de la API de Google Maps con JavaScript, XML y PHP.

6. Simulación y Resultados

Después de explicar minuciosamente cada paso que hemos hecho para conseguir una aplicación web como resultado final, en este apartado vamos a explicar las diferentes simulaciones que se han hecho y los resultados obtenidos. Vamos a comprobar el correcto funcionamiento de cada etapa desde el montaje electrónico de los sensores y sus conexiones entre sí mismos, hasta la visualización de la información de algún sensor en concreto en la aplicación web desarrollada. También realizaremos una prueba de comunicación a través de la red 3G y la red WIFI como alternativa para comprobar el correcto envío de los paquetes desde el cliente, la Raspberry, hasta el servidor, nuestro ordenador.

6.1 Comprobación de las conexiones de los sensores con la Raspberry

En primer lugar, vamos a comprobar el funcionamiento de las conexiones de los sensores con los cables SeaTalk y los conectores T entre otros. Para eso realizamos el montaje como el de la Figura 30 en el laboratorio SICOMO.

Figura 30 Montaje del sistema en laboratorio SICOMO

Cuando montamos el sistema, para comprobar que todos los sensores están perfectamente conectados, analizamos la aplicación que se visualiza en la pantalla de la Raspberry, MyPlotter. Uno de los modos para verificar el correcto funcionamiento de los sensores es realizando acciones sobre dichos sensores. Por ejemplo, si cogemos el sensor de velocidad y movemos la rueda que contiene, podemos observar en la pantalla la variación que se produce en la velocidad. Lo mismo ocurre con el sensor del compás, GPS o el anemómetro.

Figura 31 MyPlotter

6.2 Comprobación de la conexión vía WIFI y 3G

El enlace a través de la radiofrecuencia es importante puesto que nos permite enlazar cliente y servidor. Para comprobar esta comunicación entre ambos dispositivos, vamos a realizar una petición mediante un ping.

En primer lugar, realizamos la verificación a través de la red WIFI. Después de habilitar la red WIFI, explicada en el Anexo 2, realizamos un ping a la Raspberry desde nuestro ordenador en un programa de comandos como CMD.

```
C:\Users\Hassan>ping 192.168.0.135

Haciendo ping a 192.168.0.135 con 32 bytes de datos:
Respuesta desde 192.168.0.135: bytes=32 tiempo=10ms TTL=64
Respuesta desde 192.168.0.135: bytes=32 tiempo=2ms TTL=64
Respuesta desde 192.168.0.135: bytes=32 tiempo=6ms TTL=64
Respuesta desde 192.168.0.135: bytes=32 tiempo=6ms TTL=64

Estadísticas de ping para 192.168.0.135:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 2ms, Máximo = 10ms, Media = 6ms

C:\Users\Hassan>
```

Figura 32 Ping desde nuestro ordenador a Raspberry vía WIFI

Observamos que el ping se realiza de manera correcta por lo que existe comunicación entre ambos dispositivos.

El siguiente paso es comprobar el funcionamiento de la comunicación entre el barco y el servidor a través de la red 3G. Por lo tanto, tenemos que deshabilitar el adaptador WIFI

de la Raspberry ya que por defecto está configurado como predeterminado. Es decir, si no deshabilitamos el controlador del WIFI, la comunicación siempre va a realizarse a través del WIFI. Para deshabilitarlo, la explicación se encuentra en el último párrafo en el Anexo 2. A su vez es necesario haber configurado el modem 3G tal y como se explica en el Anexo 3.

Una vez realizado el anterior paso, procedemos a realizar un ping a la Raspberry desde nuestro ordenador a través de la red 3G.

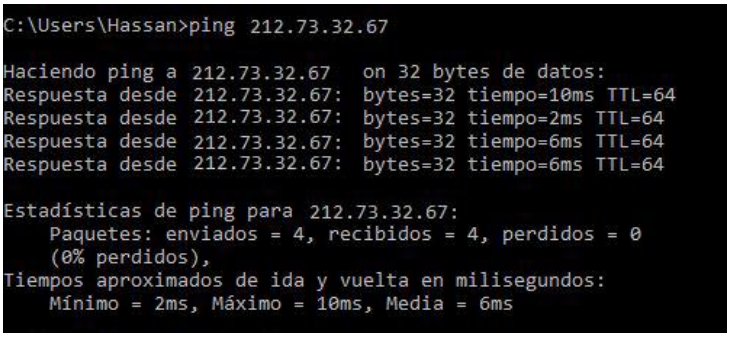

Figura 33 Ping desde nuestro ordenador a la Raspberry vía 3G

6.3 Transmisión y recepción de datos

Una vez comprobado el correcto funcionamiento de la comunicación inalámbrica tanto vía Wireless como vía 3G, procedemos a comprobar el funcionamiento de la transmisión y recepción de datos entre la Raspberry y el servidor.

Para ello desde el servidor, nuestro ordenador, usamos una aplicación para la recepción programada en Java. Esta aplicación es la misma usada en el proyecto de MotoStudent. [14]

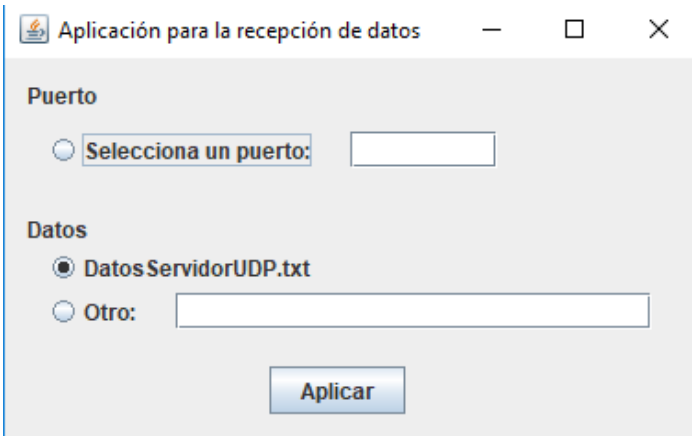

Figura 34 ServidorUDP para recepción de paquetes

Introduciendo el puerto 1704, tal y como lo hemos explicado en el Anexo 4, y seleccionando un nombre para el fichero .txt donde se almacenarán los datos, le damos a Aplicar. Observamos que la aplicación establece la conexión con la base de datos.

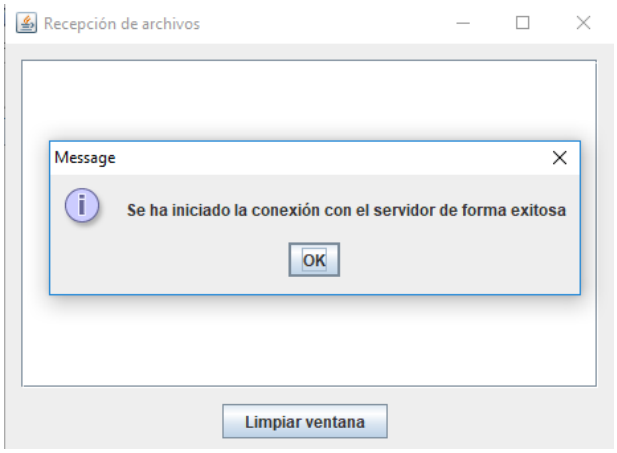

Figura 35 Conexión de ServidorUDP con BBDD

Cuando le damos a "OK", en la Figura 35, observamos que se reciben de manera exitosa los datos enviados por la Raspberry.

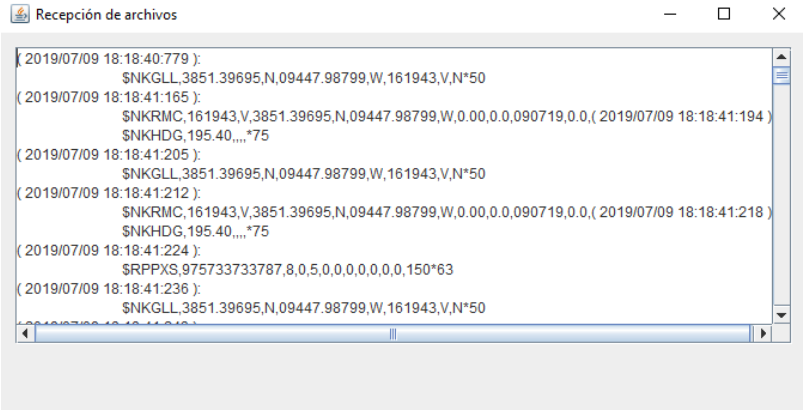

Figura 36 Recepción de datos

Para comprobar que los datos se almacenan en la base de datos, accedemos a phpMyAdmin y lo comprobamos. Y efectivamente, observamos que los datos se almacenan correctamente en la BBDD.

Figura 37 Paquetes almacenados en BBDD

6.4 Representación de los datos de los sensores en la aplicación web

El resultado final de la aplicación web implementada y desarrollada tiene el siguiente aspecto:

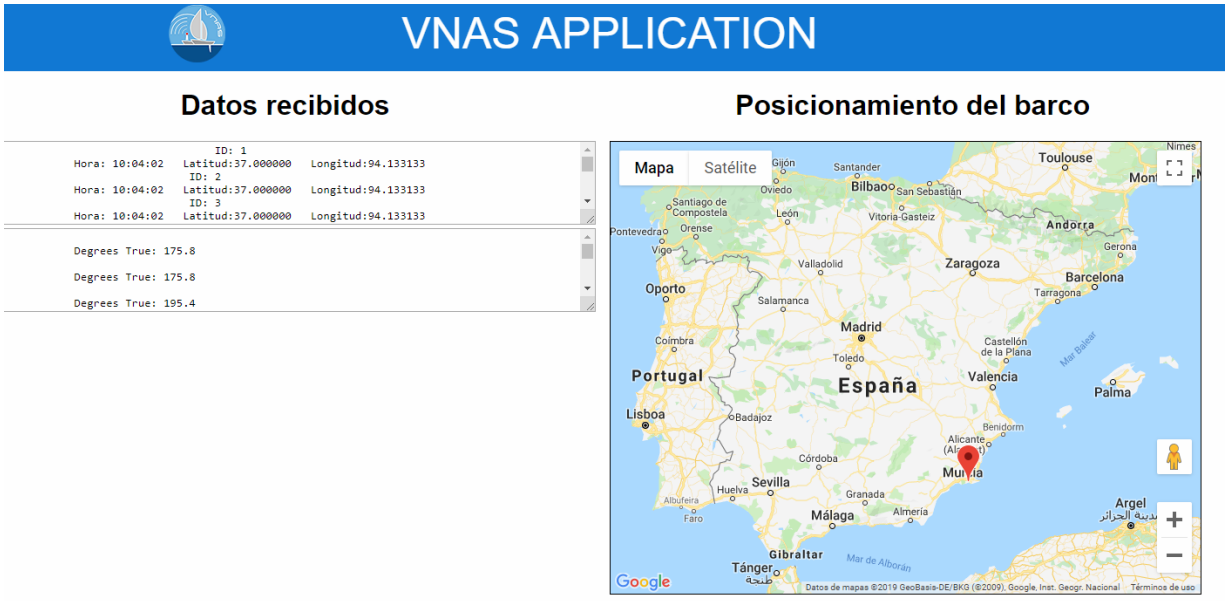

Figura 38 Aplicación web de VNAS

En la columna de la izquierda podemos visualizar dos tablas. La primera tabla recoge e imprime los datos del GPS concretamente el dato de latitud y longitud del barco medido por el sensor a una hora exacta. En la segunda tabla se visualizan los datos del sensor Velocidad del barco medidos en tiempo real.

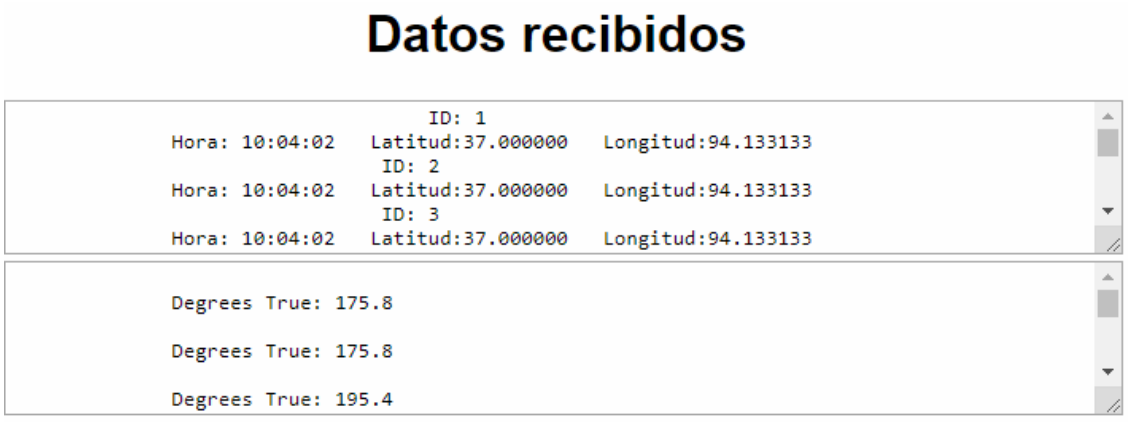

Figura 39 Columna izquierda de la aplicación WEB

Por ejemplo, la interpretación del sensor GPS sería la siguiente:

Sensor	Hora de medición	Latitud	Longitud
GPS	10:04:02	37º	94º13'31''

En la columna de la derecha podemos observar el mapa de la API de Google Maps en la cual nos marca con un marcador la posición concreta del barco.

Figura 40 Posicionamiento del barco en la API de Google Maps

En principio, surgieron problemas con la medición de la posición del GPS puesto que nos encontrábamos en el laboratorio situado en el sótano en el cual no teníamos comunicación con satélite. Para solucionar esto, probamos sacar el material al patio de la Escuela Técnica Superior de Telecomunicación y exitosamente obtuvimos los resultados que buscábamos.

Figura 41 Prueba de medición en la superficie terrestre

7. Conclusiones y líneas futuras

Las conclusiones que podemos sacar del proyecto VNAS son varias. En primer lugar, el hecho de trabajar con unos materiales altamente efectivos como los sensores de GARMIN o los cables SeaTalk, nos permiten diseñar prototipos en los cuales podemos desarrollar una inmensa cantidad de funcionalidades. Estas tecnologías ofrecen al usuario la capacidad de diseñar barcos de manera profesional con el propósito de ponerlos a prueba ante condiciones exigentes.

Por otra parte, el desarrollo de la telemetría entre el servidor y la Raspberry, y el diseño de la aplicación web para la representación de los datos proporcionados por los sensores, nos han permitido trabajar con una diversidad de tecnologías las cuales algunas de ellas desconocíamos. Esto nos ha blindado la posibilidad de aprender nuevas herramientas para el desarrollo de proyectos como este. El resultado ha sido la obtención de una aplicación web básica en la que hemos podido visualizar en tiempo real, datos de los sensores.

Como líneas futuras, es necesario la continuación con el desarrollo e implementación de la aplicación web con el fin de recoger la información de todos los sensores disponibles. A su vez, es necesario desarrollar en la aplicación funciones que permiten al usuario enviar información como ordenes al barco para que éste pueda realizar una acción.

También se puede profundizar en el desarrollo de la comunicación entre el servidor y el cliente, realizando ésta a través de la red 4G o 5G, redes que ofrecen más recursos al cliente

Finalmente, como línea futura importante, es necesario la instalación de los componentes en la superficie diseñada en el catamarán con el objetivo de ponerlo a prueba en el mar.

BIBLIOGRAFÍA

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

[2] Humberto Martínez Barberá "Vehículo Autónomo para Aplicaciones Marinas"

[3] http://www.dimf.upct.es/motoupct/index.html

[4] https://es.wikipedia.org/wiki/NMEA_0183

[5] PFCs de otros proyectos

Alejandro González Redel "Implementación de un sistema de radiocomunicaciones para la transmisión de la telemetría de un barco autónomo."

María Belén Pérez Muñoz "Implementación de un sistema de radiocomunicaciones para la transmisión de la telemetría de una moto de carreras"

Víctor Huéscar López "Desarrollo de una aplicación web de telemetría para el control de una moto de competición"

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

[7] https://www.slideshare.net/raquelbiolog/22-telemetria

[8] https://core.ac.uk/download/pdf/29405305.pdf

[9] https://www.xataka.com/robotica-e-ia/rolls-royce-presenta-finlandia-su-primer-ferry-autonomo

[10] https://www.larazon.es/lifestyle/tendencias/el-barco-del-futuro-ya-es-una-realidad-ND19929242

[11] https://vadebarcos.net/2018/09/15/sea-hunter-el-cazador-de-submarinos-autonomo-de-la-us-navy/

[12] https://es.wikipedia.org/wiki/NMEA_2000

[13] MyPlotter: aplicación usada en el proyecto VAMAR y programada por Humberto Martínez Barbéra.

[14] Víctor Huéscar López "Desarrollo de una aplicación web de telemetría para el control de una moto de competición"

[15] https://www.tronico.fi/OH6NT/docs/NMEA0183.pdf

[16]https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)#Orientado_a_objetos

[17] https://www.fdi.ucm.es/profesor/jpavon/web/35-PHP-MySQL.pdf

[18]https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/Qu%C3%A9_es_JavaScript

[19] https://netbeans.org/

[20] https://www.apachefriends.org/es/index.html

[21] http://brackets.io/

[22] Aplicación desarrollada en el proyecto “Desarrollo de una aplicación web de telemetría para el control de una moto de competición” de Víctor Huéscar López .
[23] http://codigoxules.org/conectar-mysql-utilizando-driver-jdbc-java-mysql-jdbc/

[24] https://developers.google.com/maps/documentation/javascript/mysql-to-maps

[25] https://www.amazon.es/Jago-ETBM03-1-fueraborda-el%C3%A9ctrico-propulsora/dp/B00JKL7512

[26] https://buy.garmin.com/es-ES/ES/p/100702

[27] https://www.azimutmarine.es/nautica/compas-h2183-de-estado-solido-para-nmea0183-2000.html

[28] https://www.waveinn.com/nautica-pesca/raymarine-dst800-analogic-triducer/578313/p?utm_source=google_products&utm_medium=merchant&id_product

[29] https://www.waveinn.com/nautica-pesca/raymarine-tacktick-t101/578711/p?utm_source=google_products&utm_medium=merchant&id_producte=665201&country=es&gclid=Cj0KCQjwyLDpBRCxARIsAEENsrKYoZMc1mg4hnaaXwELpYa423PiorcdW_mek1lIbW6qozVhTMmYwXgaArb8EALw_wcB&gclsrc=aw.ds

[30] https://www.amazon.es/Raymarine-E22158-convertidor-color-negro/dp/B003E1WVRW/ref=asc_df_B003E1WVRW/?tag=googshopes-21&linkCode=df0&hvadid=82841691490&hvpos=1o3&hvnetw=g&hvrand=17290569546008646983&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=1005499&hvtargid=pla-173821837930&psc=1

[31] https://www.amazon.es/Nueva-Pican2-CAN-Bus-tarjeta-Raspberry/dp/B01HBWI8BW

[32] https://www.amazon.es/Raspberry-Pi-Modelo-Placa-Color/dp/B07BFH96M3/ref=asc_df_B07BFH96M3/?tag=googshopes-21&linkCode=df0&hvadid=199473611233&hvpos=1o1&hvnetw=g&hvrand=17921277243435682915&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=1005499&hvtargid=pla-433654637008&psc=1

[33] https://www.azimutmarine.es/agm-12-55-ah.html

[34] https://www.amazon.es/Huawei-Technology-Ltd-E3531i-2-21-6Mbps/dp/B011YZZ6Q2/ref=asc_df_B011YZZ6Q2/?tag=googshopes-21&linkCode=df0&hvadid=82844303170&hvpos=1o2&hvnetw=g&hvrand=14838233591226850523&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=1005499&hvtargid=pla-187800433943&psc=1

[35] https://www.amazon.es/Kingston-SDC10-32GB-microSDHC-adaptador/dp/B00519BEQY/ref=asc_df_B00519BEQY/?tag=googshopes-21&linkCode=df0&hvadid=54367567275&hvpos=1o1&hvnetw=g&hvrand=10770316160726658083&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=1005499&hvtargid=pla-78307977206&psc=1

[36] https://www.luisllamas.es/raspberry-pi-wifi/

[37] https://www.thefanclub.co.za/how-to/how-setup-usb-3g-modem-raspberry-pi-using-usbmodeswitch-and-wvdial

ANEXOS

ANEXO 1

Descripción de los componentes que estarán montados en el prototipo

- JAGO ETBM02-1 MOTOR FUERABORDA ELÉCTRICO 50 LB

Los motores JAGO eléctricos son los que han sido seleccionados para ir montados en el catamarán. Se tratan de motores de 55 libras de empuje y 636 W de potencia los cuales funcionarán bajo una alimentación de 12 V en continua. Se instalarán dos motores que recibirán órdenes por parte de un microcontrolador. [25]

Figura 42 MOTOR FUERABORDA ELÉCTRICO 50 LB

- ANTENA GPS GARMIN 19X 10HZ NMEA 2000

La antena GPS es el sensor encargado de proporcionar la posición en tiempo real del catamarán. [26]

Figura 43 ANTENA GPS GARMIN

- COMPÁS CIEGO AIRMAR H2183 NMEA2000

El compás es un sensor que se encarga de la medición de la orientación del barco, así como su inclinación. [27]

Figura 44 COMPÁS CIEGO AIRMAR

- SMART TRIDUCER AIRMAR DST800

El sensor SMART TRIDUCER tiene una variedad de funciones. Nos proporciona tres parámetros importantes como son la velocidad a la que se desplaza el barco, la profundidad y la temperatura de éste. [28]

Figura 45 SMART TRIDUCER AIRMAR DST800

- ANEMÓMETRO CON VELETA RAYMARINE

El anemómetro junto a la veleta hacen la función de la medición de la velocidad y la dirección del viento. [29]

Figura 46 ANEMÓMETRO

- CONVERSOR SEATALK 1 A SEATALK NG

El conversor SeaTalk 1 a SeaTalk NG permite la conexión y comunicación de equipos que trabajan con SeaTalk 1 en redes de SeaTalk NG de Raymarine. [30]

Figura 47 CONVERSOR SEATALK 1 A SEATALK NG

- RASPBERRY PI3 CON EL ADAPTADOR PICAN2 Y RASPBERRYPI-DISPLAY

La Raspberry es el elemento encargado de recoger la información de los sensores, procesarlos y enviarlos al equipo central para estudiarlos. Para ello se le conecta un módulo PICAN2 con el fin de conectar la Raspberry al bus CAN al cual están conectados los componentes de Raymarine. [31]

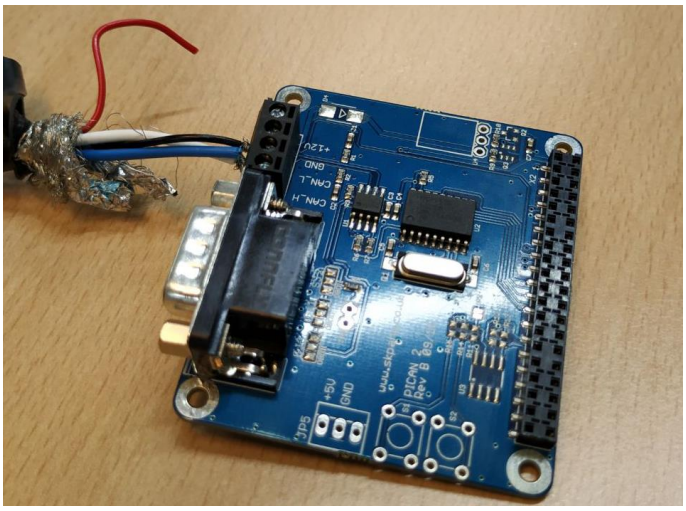

Figura 48 PICAN2

A su vez, se le conecta a la Raspberry PI una pantalla en la cual podremos visualizar los datos recogidos por los sensores mediante una aplicación programa en Java. [32]

Figura 49 RASPBERRY PI3 CON EL ADAPTADOR PICAN2

- BATERÍA ACIDO PLOMO 12/55AH

Las baterías que vamos a usar para la alimentación de toda la electrónica del catamarán serán las de tipo acido plomo 12/55AH y 12/100AH. [33]

Figura 50 BATERÍA ACIDO PLOMO

- MODEM 3G

Para poder comunicarse con la Raspberry, hace falta establecer una comunicación a través de la red 3G. Para eso necesitamos instalar en la Raspberry un modem 3G con una tarjeta SIM. El modem 3G es el elemento mediante el cual permite a la SIM conectarse a la red pública. [34]

Figura 51 MODEM 3G

- KINGSTON SDC10/8GB CLASE 10

Dentro de la Raspberry, tendremos una aplicación programada en Java, la misma que se utiliza en el proyecto VAMAR, que hace la función de recopilar los datos de los sensores y visualizarlos en la pantalla de la Raspberry PI. Para ello hace falta una tarjeta MicroSD que en este caso hemos seleccionado una de 8GB en la que se vuelca la aplicación. [35]

Figura 52 KINGSTON SDC10/8GB CLASE 10

ANEXO 2

Como activar/desactivar el adaptador WIFI de la Raspberry y conectarla a la red

- Conectamos la Raspberry con nuestro ordenador a través del cable Ethernet y asignamos al PC una IP estática como por ejemplo 192.168.100.2. Esto nos permite acceder a la Raspberry a través del protocolo SSH puesto que presenta por defecto una IP estática 192.168.100.1.
- Accedemos a la Raspberry a través del CMD, herramienta mediante la cual nos va a permitir programar la Raspberry. Para ello abrimos CMD y escribimos pi@192.168.100.1. Posteriormente nos pregunta por la contraseña de la Raspberry, que es **raspberry**
- Una vez dentro de la Raspberry, procedemos a activar la WIFI para poder conectarse a Internet. Escribimos para ello **sudo nano/etc/wpa_supplicant/wpa_suplicant.conf** que es la línea que nos permite entrar al fichero **wpa_supplicant** y configurar la red WIFI. [36]
- Dentro del archivo **wpa_supplicant.conf** escribimos el nombre de la red WIFI de laboratorio, así como la contraseña. Posteriormente salimos del archivo pulsando Control + O. Esta acción nos devuelve al panel en el que estábamos antes.

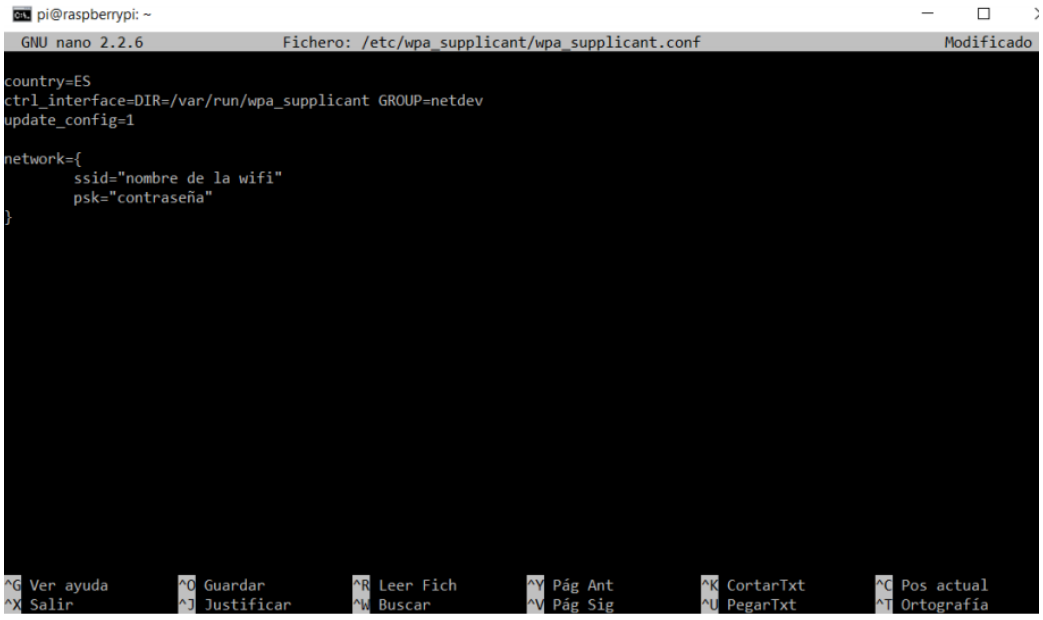

Figura 53 Fichero Wpa_supplicant

- Escribimos **sudo reboot** con el fin de reiniciar la Raspberry ya que es una acción necesaria para observar cualquier cambio que se realiza. Una vez reiniciada, desconectamos el cable Ethernet conectado a la Raspberry ya que ya no es necesario.
- Verificamos que la Raspberry está conectada al router del laboratorio. De varias formas se puede comprobar este paso. En nuestro caso, accedemos al panel de control del router para ver que dispositivos están conectados. En la Figura 54 observamos que la Raspberry está conectada de forma correcta. Observamos que la IP local que asigna el router a la Raspberry es **192.168.0.135**.

Clientes Inalámbricos

ID	Nombre	Dirección IP	Dirección MAC
1	DESKTOP-DGD2PE5	192.168.0.103	20-68-9D-E3-7F-1B
2	raspberrypi	192.168.0.135	B8-27-EB-08-96-53

Figura 54 Dispositivos conectados por WIFI al Router de SICOMO

- En futuras ocasiones si queremos acceder a la Raspberry y programarla a través de comandos podemos hacerlo sin necesidad de volver a conectar el cable Ethernet a nuestro ordenador ya que podemos acceder a través de la IP 192.168.0.135 que ha asignado el router a la Raspberry. Esto último se puede hacer siempre y cuando el ordenador desde donde accedemos a la Raspberry también esté conectado al mismo router que la Raspberry.

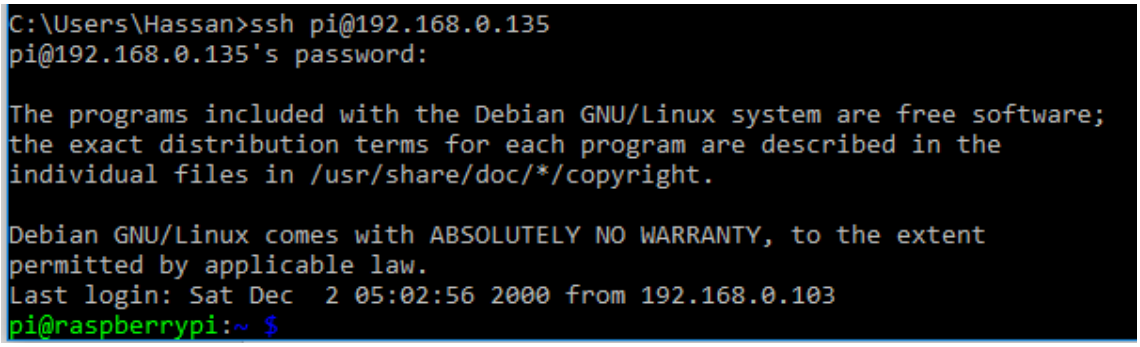

Figura 55 Conexión vía SSH con Raspberry

- Para desactivar el adaptador WIFI, basta con acceder al archivo **wpa_supplicant** y borrar el usuario y la contraseña del router. Una vez realizado esto, basta con reiniciar la Raspberry para conseguir deshabilitar el controlador del WIFI.

ANEXO 3

Como configurar el Modem 3G en la Raspberry

- En primer lugar, abrimos el comando CMD, por si no lo teníamos abierto, y nos conectamos a la Rasberry a través del protocolo SSH. Mediante el comando pi@192.168.0.135. Recordamos que 192.168.0.135 es la IP que asigna el router de laboratorio a la Raspberry cuando habilitamos el WIFI de ésta.
- Ahora procedemos a escribir el comando: **sudo apt-get update**. Este comando sirve para descargar e instalar las ultimas actualizaciones en la Raspberry. [37]
- Cuando conectamos el Modem 3G a la Raspberry, primero verificamos que está conectado correctamente. Para ello, escribimos el siguiente comando: **lsusb**. En Figura 56 observamos que el dispositivo está conectado.

```
pi@raspberrypi:~ $ lsusb
Bus 001 Device 005: ID 12d1:1433 Huawei Technologies Co., Ltd.
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp. SMSC9512/9514 Fast Ethernet Adapter
Bus 001 Device 002: ID 0424:9514 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Figura 56 Dispositivo MODEM 3G conectado a la Raspberry

- Por defecto, al conectar el Modem 3G, la Raspberry lo reconoce como dispositivo almacenamiento. Necesitamos descargar e instalar los paquetes necesarios para que el Modem pueda ser reconocido como un dispositivo 3G. Por lo tanto, escribimos: **sudo apt-get install ppp usb-modeswitch wvdial**
- Reiniciamos la Raspberry para refrescar. Escribimos **sudo reboot** y volvemos a acceder una vez reiniciada.
- El siguiente paso es el más importante. Insertamos en el Modem una tarjeta SIM que es el medio mediante el cual vamos a conectarnos a la red 3G. En nuestro caso, hemos usado una tarjeta SIM de la compañía Vodafone. Posteriormente, en la Raspberry escribimos el siguiente comando para acceder al archivo **leafpad** donde vamos a configurar el APN de nuestra SIM para tener acceder a Internet a través de la red 3G: **sudo leafpad /etc/usb_modeswitch.conf**
- Una vez dentro de leafpad, lo configuramos tal y como se muestra en la Figura 57.

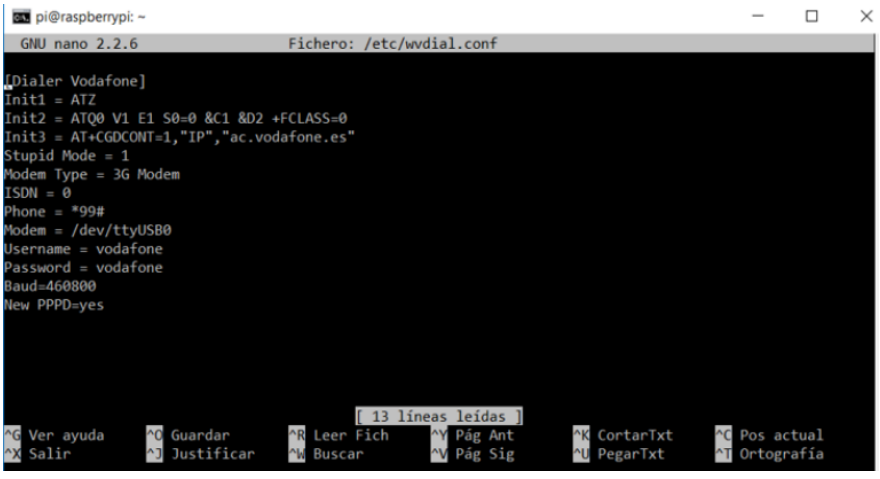

Figura 57 Configuración del MODEM 3G en la Raspberry a través de Wvdial

- Finalmente, pulsamos control + O para guardar y reiniciamos la Raspberry.

- Una vez reiniciada, procedemos a comprobar el funcionamiento del Modem y comprobar si finalmente se conecta a Internet. Para ello, entramos en la Raspberry y escribimos el siguiente comando: sudo wvdial Vodafone
- Observamos que el Modem funciona correctamente. En la siguiente figura podemos observar que la Raspberry tiene una IP local asignada por la red 3G, así como DNS primario y secundario.

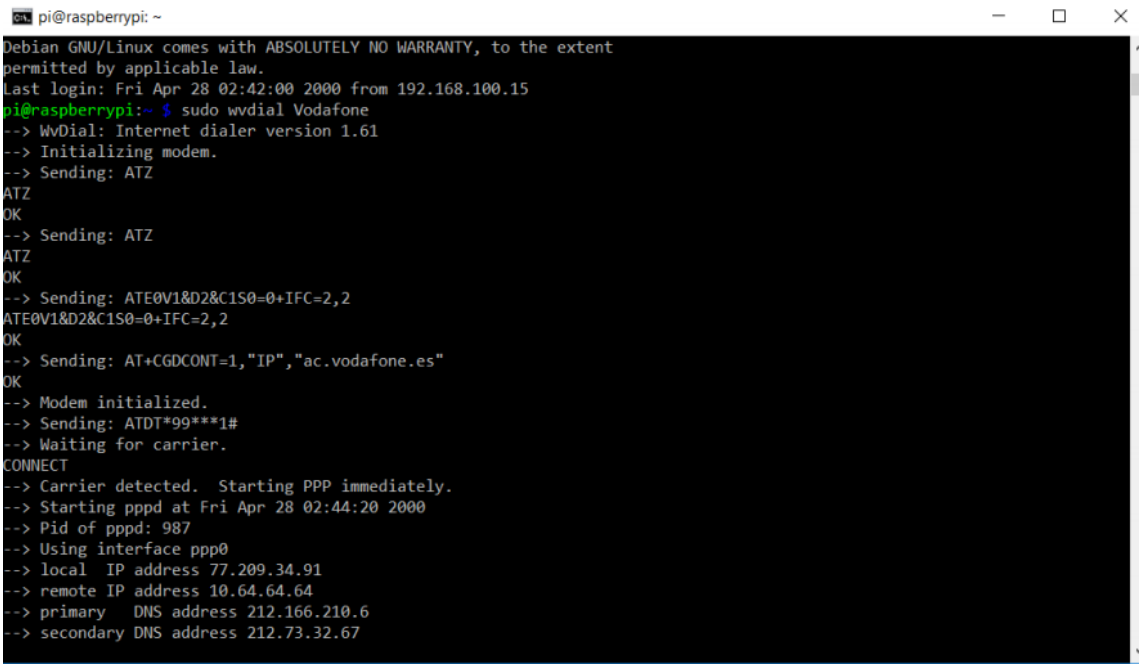

Figura 58 MODEM 3G activado

ANEXO 4

Código del programa encargado de redireccionar el tráfico desde el puerto 1703 a la dirección IP pública del router de laboratorio

```
package VnasServ;
import java.net.*;
import java.io.*;
import java.util.*;

public class VnasServ {
   public static void main(String[] args) throws IOException {
    int port=1703; //Puerto de escucha
    byte []datos = new byte[1024]; //tamaño del datagrama
    DatagramSocket socket; //Creamos el socket de recepcion
        socket = new DatagramSocket(port);
    DatagramSocket enviador;
        enviador = new DatagramSocket();
    System.out.println("Servidor en escucha.");

    while(true){
    DatagramPacket dgp = new DatagramPacket(datos, datos.length);
    socket.receive(dgp);
    byte[] recibido = dgp.getData();
    System.out.println(recibido);
    System.out.println("Mensaje recibido");
    DatagramPacket dgp_2 = new DatagramPacket(recibido,
recibido.length,
            InetAddress.getByName("212.128.44.153"), 1704);
    enviador.send(dgp_2);

    }
}}
```

Códigos 12 Programa que redirecciona el tráfico del puerto 1703 a otro puerto

- En la primera parte del código, creamos un DatagramaSocket que se encarga de escuchar el puerto 1703.
- En la segunda parte del código, creamos un DatagramaSocket que se encarga de redireccionar la información, escuchada por el puerto 1703, a la dirección pública del router que es 212.128.44.153, mediante el puerto 1704. Por lo tanto, en el router es necesario acceder a su panel de control y abrir el puerto 1704.

Servidores Virtuales

Añadir Eliminar

	ID	Tipo de servicio	Puerto externo	IP interna	Puerto interno	Protocolo	Estado	Modificar
☐	1	---	1704	192.168.0.103	1704	UDP		
☐	2	---	1703	192.168.0.146	1703	UDP		
☐	3	HTTP	8081	192.168.0.146	8081	TCP		

Figura 59 Puertos abiertos en el Router de SICOMO

ANEXO 5

Como crear una base de datos con la herramienta XAMPP

Iniciamos el programa XAMPP para que nos permita el acceso a phpMyAdmin. Una vez dentro de XAMPP, iniciamos los módulos Apache y MySQL como se muestra en la Figura 60.

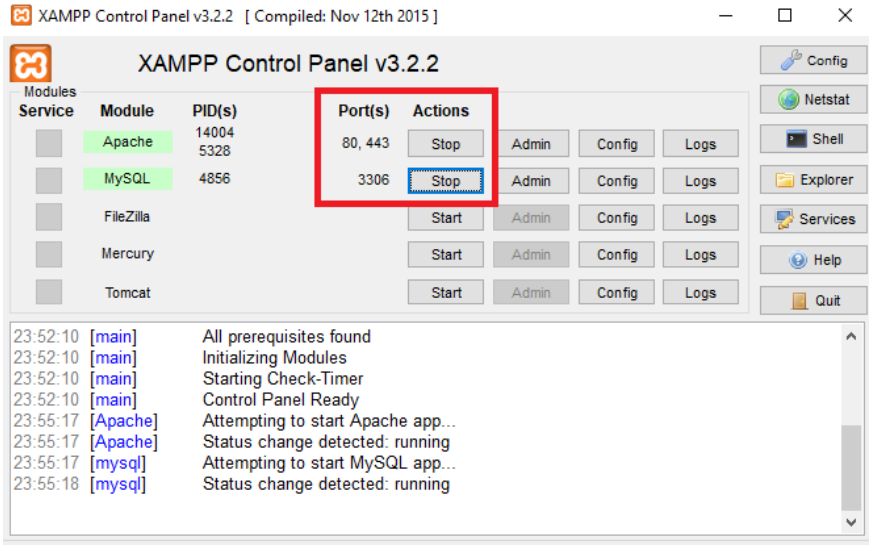

Figura 60 Panel de control de XAMPP

Ahora accedemos a phpMyAdmin. Para eso accedemos a cualquier navegador y escribimos http://localhost. Se nos muestra un panel como la Figura 61, y le damos a phpMyAdmin.

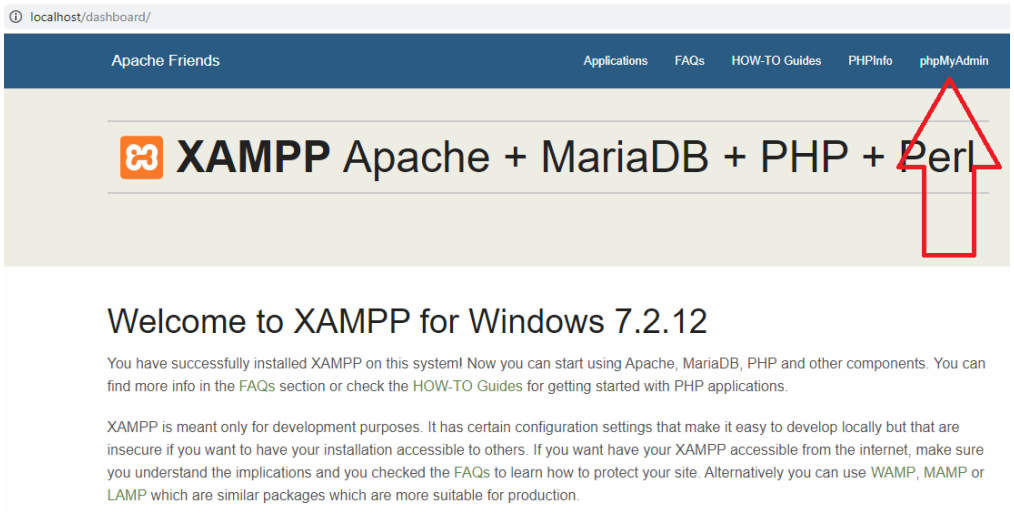

Figura 61 Localhost de XAMPP

Una vez dentro de phpMyAdmin, nos visualizará un panel como el de la Figura 62. En la columna de la izquierda es donde se encuentra la herramienta para crear una base de datos.

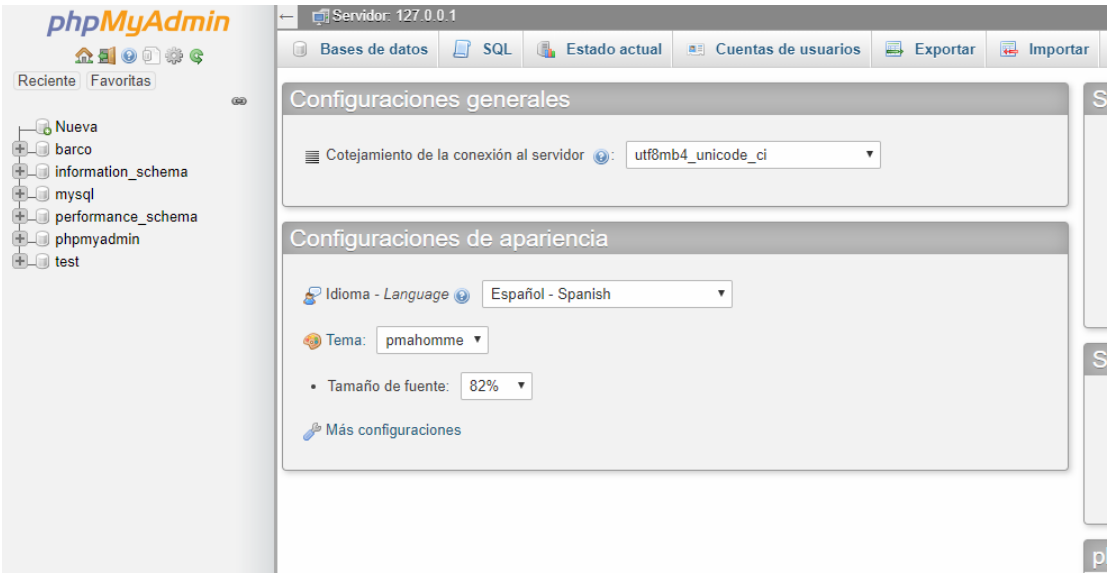

Figura 62 phpMyAdmin

Pulsamos el botón Nueva para crear una base de datos y le asignamos un nombre, así como el tipo de cotejamiento (normalmente suele ser utf8_general_ci).

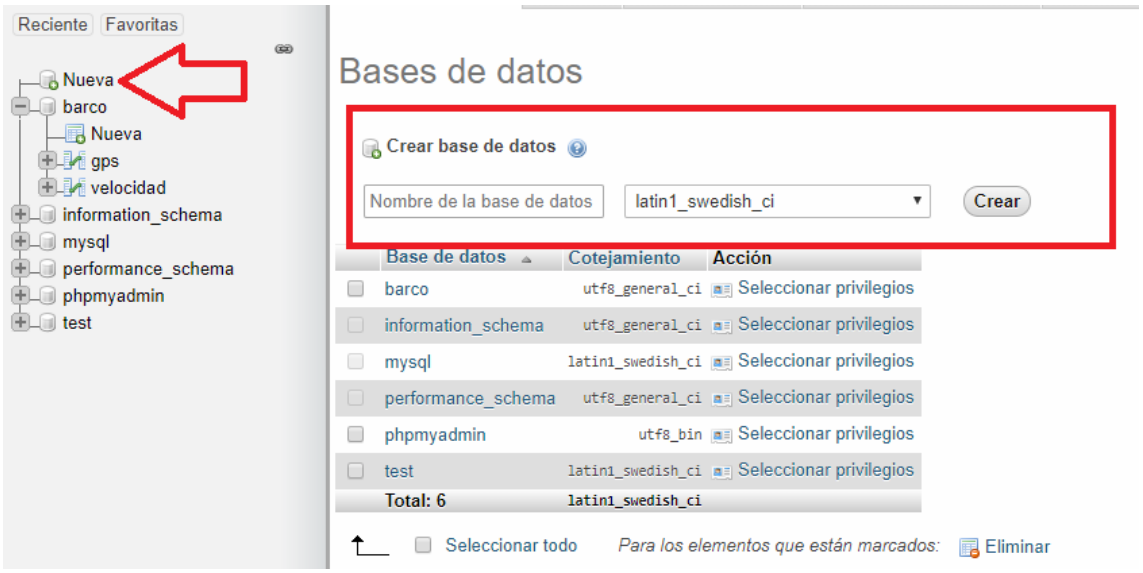

Figura 63 Como crear una BBDD

Una vez creada la base de datos, procedemos a crear tablas en ella. Cada tabla corresponderá a un sensor. En principio vamos a crear dos tablas que corresponden al sensor de la velocidad y el sensor de posicionamiento GPS. En la tabla, nos aparecen campos en los cuales vamos a declarar variables. Cada variable corresponde a un dato de la línea. En Figura 64 y 65, se visualizan las estructuras de las tablas creadas para GPS y Velocidad.

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo
☐	1	Latitud	float(10,6)			No
☐	2	NorS	varchar(1)	utf8_general_ci		No
☐	3	Longitude	float(10,6)			No
☐	4	EorW	varchar(1)	utf8_general_ci		No
☐	5	Time	varchar(8)	utf8_general_ci		No
☐	6	Status	varchar(1)	utf8_general_ci		No
☐	7	ID 🔑	int(11)			No

Figura 64 Variables creadas en la tabla GPS

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
☐	1	degreestrue	float(5,1)			No	*Ninguna*
☐	2	TorF	varchar(1)	utf8_general_ci		No	*Ninguna*
☐	3	degreesmagnetic	float(5,1)			No	*Ninguna*
☐	4	Magnetic	varchar(1)	utf8_general_ci		No	*Ninguna*
☐	5	Knots	float(6,1)			No	*Ninguna*
☐	6	Nknots	varchar(1)	utf8_general_ci		No	*Ninguna*
☐	7	Kilometers	float(4,1)			No	*Ninguna*
☐	8	K	varchar(1)	utf8_general_ci		No	*Ninguna*

Figura 65 Variables creadas en la tabla Velocidad