UNIVERSIDAD RICARDO PALMA

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA

SISTEMA SIMULADOR DEL SONAR DE LOS SUBMARINOS. UNA APLICACIÓN PARA LA MARINA DE GUERRA DEL PERÚ

PROYECTO DE TESIS PARA OPTAR EL TITULO PROFESIONAL DE INGENIERO INFORMÁTICO

Presentado por: Arias Bailly, Bruno

Asesor de Tesis Interno : Cabrera Diaz, Javier

Asesor de Tesis Externo : Del Carpio Azálgara, Luis

Lima , Perú – 2008

A mi madre
Libia Bailly Narváez

INDICE

Agradecimientos

Por darme la oportunidad de estudiar, agradezco a mi país, y le hago esta pequeña contribución.

Por la confianza depositada, agradezco a la Marina de Guerra del Perú, es especial a la Comandancia de la Fuerza de Submarinos.

Por el empuje en los días difíciles, agradezco el apoyo de mis padres y familiares.

Por permitirme crecer con los conocimientos adquiridos durante el Pre-grado, agradezco a la Universidad Ricardo Palma.

Por guiarme a través de la metodología de la investigación, agradezco a mis asesores el Capitán de Corbeta Luis DEL CARPIO Azálgara 2do Comandante de la Escuela de Submarinos y al Dr. Javier CABRERA Díaz quienes siempre estuvieron presente para apoyarme y aconsejarme en forma oportuna durante el desarrollo de la tesis.

Por colaborar, agradezco al Contra Almirante Fergán HERRERA Cuntti Comandante de la Fuerza de Submarinos, al Capitán de Navío Jorge SAZ Fernández, al Capitán de Fragata Luis SACO-VERTIZ Portal 1er Comandante de la Escuela de Submarinos, al Capitán de Corbeta Marco MONTERO Gallegos, al Capitán de Corbeta Fernando CASTILLO Heredia, al Teniente Primero Alberto ARRESE Schenone Jefe de Ingeniería del B.A.P Arica, al Teniente Primero Alfieri BUCCICARDI Duezada, al Técnico de Primera Motorista Juan ULLOA Osorio, al Técnico de Primera Electrónico Walter REGALADO Rosas, al Técnico de Tercera Controlista Leoncio BARRIGA Trujillo y a todo el personal sub alterno que forma parte de la dotación de la Escuela de Submarinos que con sus consejos específicos y experiencias me ayudaron en el desarrollo del software de simulación.

A todo ellos mis más sincero agradecimiento, por que no hubiese sido posible realizar este proyecto sin su ayuda.

Sistema Simulador del Sonar de los Submarinos – Una aplicación para la Marina de Guerra del Perú

Bruno ARIAS Bailly
Candidato a Ingeniero
Callao, Perú, 032
brunoariasbailly@yahoo.es

Javier CABRERA Díaz
Asesor de Tesis Interno

Luis DEL CARPIO Azálgara
Asesor de Tesis Externo

Resumen

En este trabajo de Tesis se propone el desarrollo de un Software de Simulación que permita brindar un entrenamiento más cercano a la realidad al personal de sonaristas en formación y calificado; con el objetivo de dar a conocer la operabilidad del sonar integrado a bordo de las unidades submarinas, así como la interacción con los módulos de simulación existentes en simulador de ataque de la Fuerza de Submarinos de la Marina de Guerra del Perú, ubicado en la Escuela de Submarinos.

La Tesis se divide en cuatro partes: la Introducción al tema propuesto y su problemática; el Marco Teórico que corresponde a las variables que engloban el alcance de este trabajo; el Estado del Arte donde se expone un caso en particular de un software de simulación de sonar desarrollado bajo otro lenguaje de programación, el cual no tiene similitud con el sonar existente a bordo de las unidades submarinas; finalmente el planteamiento de una propuesta de desarrollo, mejoramiento y ejecución de un sistema simulador de sonar que cumpla con las funcionalidades del equipo de a bordo y se integre al sistema de simulación existente.

La hipótesis está en demostrar que es posible desarrollar un software de simulación del sonar que permita realizar las funcionalidades que contiene el equipo original.

Probado esto a lo largo de la Tesis, se logra el objetivo: demostrar que es posible construir un software simulador de sonar que muestra los contactos, genere el ruido sintético de las hélices, envié los datos necesarios a los otros módulos de simulación, y simule las pruebas Test que contiene el equipo original.

Palabras Clave: Sistema Simulador, Sonar, unidades submarinas.

Sonar System Simulator of Submarines – An application for the Navy military of Peru

Bruno ARIAS Bailly
Candidate to engineer
Callao, Perú, 032
brunoariasbailly@yahoo.es

Javier CABRERA Díaz
Internal Thesis adviser

Luis DEL CARPIO Azálgara
External Thesis adviser

Abstract

In this Thesis work the development of a Software Simulation that allows closest offer at training to the reality for sonar operator's personal in formation and described System sets out; with the objective to present the operability on board to Sonar device of the submarine units, as well as interaction with existing simulation modules inside the simulator Attack of Submarine Force of the Navy military of Peru, Located at the Submarine School.

The Thesis is divided in four parts: the Introduction to the proposed subject and its problematic; Theoretical Frame that corresponds to variables that include the reach of this work; the State of the Art where a case in individual of a Sonar Simulator Software constructed under another programming language is exposed, which does not have similarity with electronic equipment sonar on board of the submarine units; finally the exposition of a development proposal, improvement and execution of a sonar simulator software that fulfills the functionality of the original electronic equipment on board.

The hypothesis is in demonstrating that it's possible to develop a sonar simulator software that allows to carry out functionalities of the original electronic equipment.

Proven this throughout the Thesis, the objective is obtained: to demonstrate that it's possible to construct a sonar simulator software that shows the contacts, generate the noise corresponding, sent the data necessary to other simulation modules, and simulating the Test containing in the original equipment.

Key words: Simulator System, Sonar, submarine units.

INTRODUCCIÓN

La Simulación es una imitación de algo real, de algún estado o proceso; es por ello que esta ha sido utilizada para optimizar funcionamientos o performance, seguridad, pruebas, entrenamiento y educación. La Simulación permite presentar los efectos reales a condiciones variantes y permite probar diferentes cursos de acción.

Desde hace varios años se han utilizado dispositivos, de acuerdo a la disponibilidad tecnológica de la época, para desarrollar máquinas que permitan simular algo real, habiendo sido desarrollados artificios mecánicos, eléctricos y después electrónicos para este fin.

Desde el surgimiento de las computadoras, la simulación ha avanzado enormemente recreando el comportamiento de equipos o sistemas con el fin de entrenamiento de personas en diversos campos a nivel mundial. Los sistemas simuladores constituyen una rama de la informática que cuenta con un gran éxito en el mundo, siendo estas aplicaciones de utilidad en temas tan variados que pueden ir desde la medicina hasta la utilización en áreas militares, como se demostrará en este trabajo de tesis. Los sistemas simuladores tienen cada vez mas un mayor auge debido a que van apareciendo técnicas como las de 3D y equipos más potentes que permiten un mayor realismo. Es muy común apreciar a muchos especialistas afirmar que el limite de los simuladores esta en la imaginación humana, siendo siempre de utilidad en toda aquella área donde se necesite operar un equipo o sistema disminuyendo riesgos y costos.

En los simuladores es necesario aplicar fórmulas matemáticas y físicas que permitan imitar el comportamiento de los equipos o sistemas que son objeto de la simulación, esto con el objetivo de obtener el mayor realismo posible, que por consecuencia brinde mejores resultados.

En la actualidad los sistemas simuladores no son mas parte de un solo computador, sino que comprenden un área física cada vez mayor y mas compleja, compuesta de

varios computadores conectados en una red cumpliendo cada uno una funcionalidad distinta, como la generación de datos a través de tramas en la red, visualización en tercera dimensión, así como uso de consolas o controles idénticos a los reales.

El objeto general de la realización de esta tesis es el estudio y aplicación del tema de un simulador en el área naval, que por su naturaleza no dispone de una amplia bibliografía con respecto al equipo objeto de este estudio, pero se dispone de una gran disponibilidad bibliográfica en las herramientas y metodologías para el desarrollo del simulador. En nuestro país se vienen desarrollando, cada vez mas, aplicaciones de este tipo en el entorno militar como es el caso.

Uno de los objetivos específicos que presenta esta tesis es el desarrollar un Software Simulador utilizando como herramienta el lenguaje de programación Java, que a pesar que no se utiliza con frecuencia en el ámbito de simuladores, es posible obtener una solución óptima para el requerimiento.

Otro de los objetivos específicos de desarrollo de esta tesis es el elaborar un Software de Simulación del Sonar de los Submarinos, que permita interactuar con los demás módulos del Simulador de Ataque de la Escuela de Submarinos, con el fin de brindar un entrenamiento al personal sonaristas en formación y calificados, haciendo uso de un equipo igual al que existen a bordo de las unidades submarinas mencionadas.

El interés por el desarrollo de esta tesis nació debido a que quien es partícipe de este tema de investigación, al haber pertenecido en un corto periodo a la Marina de Guerra del Perú, habiendo tenido el interés de ser un Oficial Submarinista y contando con el conocimiento necesario, aceptó el reto de desarrollar un software que permitiera contribuir en la formación del los futuros sonaritas, así como en el entrenamiento de aquellos que poseen esta calificación en conjunto con todos los módulos del Simulador de Ataque de la Escuela de Submarinos de nuestra armada.

De esta manera esta tesis tratará en el ***Capitulo 1: Presentación del Problema*** lo que concierne al título del trabajo en cuestión, así una definición mas detallada del

problema, los objetivos, la justificación, el alcance y por último la organización hacia la cual está dirigido el trabajo.

En el ***Capitulo 2: Marco Teórico*** se dará a conocer el significado de Simulador, una breve explicación acerca del Sonar de un Submarino, así como el lenguaje de programación usado, la generación de sonido sintetizado y los protocolos de comunicación estándares.

En el ***Capitulo 3: Estado del Arte*** se presenta un software simulador de sonar antiguo, con el propósito de conocer como se encuentra este Sistema, encontrar sus logros y falencias, lo cual me permitiera obtener una clara idea y entendimiento del mismo, así mismo se presentan las tendencias en el desarrollo de simuladores tanto en nuestro país como en el extranjero.

En el ***Capitulo 4: Aporte*** explica la aplicación realizada, los casos de uso involucrados, las clases utilizadas para resolución del problema.

Por último, con la experiencia obtenida en el estudio e implementación de Simuladores en un caso práctico, puedo, como una de las conclusiones definir que durante el ciclo de desarrollo la etapa más compleja es la creación o puesta en practica de las operaciones matemáticas y físicas que simulan la realidad de las operaciones a bordo, otra etapa que toma tiempo es la construcción de los circuitos y hardware que permiten la interacción con el software. Los Sistemas de Simulación son herramientas cada vez más necesarias en la actualidad y que nos sirven de apoyo en la realización de diversas actividades, como en este caso disminuyendo el costo generado por un ejercicio real para realizar un entrenamiento.

CAPÍTULO 1
PRESENTACIÓN DEL PROBLEMA

1.1 Titulo del trabajo

Sistema Simulador del Sonar de los Submarinos – Una aplicación para la Marina de Guerra del Perú.

1.2 Definición del problema

El Simulador de Ataque de la Escuela de Submarinos de la Marina de Guerra del Perú cuenta con un software de sonar limitado, que no cuenta con la funcionalidad del equipo real, como una interfaz electrónica y gráfica, además no genera un ruido sintetizado que cumpla con los parámetros de los contactos tales como distancia y velocidad, por lo contrario utiliza grabaciones que no corresponden a dichos parámetros.

1.3 Objetivo

1.3.1 Objetivo Principal

- Desarrollar un software que permita simular un sistema de sonar de un submarino, que interactúe con el simulador de ataque de la Escuela de Submarinos utilizado

para el entrenamiento de las dotaciones de las unidades submarinas, con el mayor realismo posible, con la funcionalidad del equipo original y genere ruido sintético de las hélices de los contactos simulados.

1.3.2 Objetivos Secundarios

- Mejorar el simulador de ataque que se encuentra dentro de las instalaciones de la Escuela de Submarinos.

- Medir la capacidad de puesta en práctica de los conocimientos que los alumnos hayan adquirido, así como los del personal abordo de las unidades submarinas. (como medida de entrenamiento)

1.4 Justificación

- Debido a que el Simulador de Ataque de la Escuela de Submarinos no cuenta con un software apropiado que simule correctamente las funcionalidades del equipo de sonar abordo de las unidades submarinas y sobre todo por que carece de la capacidad de generar ruido sintetizado que permita imitar los sonidos de los buques en el mar de acuerdo a sus características, entre otras bondades, surge la necesidad de investigar y desarrollar un software que cumpla con los requerimientos de la escuela antes mencionada.

- Mediante un software nuevo que simule el equipo de sonar original, los sonaristas estarán en capacidad de entrenarse en el ejercicio de reconocer algunas características requeridas de los buques que se encuentren en la mar mediante el sonido correspondiente que estos generan y realizar el seguimiento necesario a cada uno de los buques encontrados para luego enviar y comunicar dicha información hacia los otros puestos de comando dentro del submarino simulado (Simulador de Ataque).

- Costo de Desarrollo

2.7	Una (01) computadora Pentium IV	$800.00
3.7	Un (01) IDE de Programación NetBeans	$00.00
4.7	Un (01) S.O Windows 2000 Prof.	$150.00
5.7	Total Costo de Desarrollo	$950.00 ó S/.2660.00

- Costo de Implantación

6.7	Una (01) computadora Pentium IV	$ 800.00
7.7	Tres (03) planchas galvanizadas 1/8	$ 375.00
8.7	Cinco (05) kilos de soldadura cellocord 6011 3/3	$ 25.00
9.7	Dos (02) platinas de 1” x 1/8	$15.71
10.7	Dos (02) correderas telescópicas	$20.00
11.7	Un (01) disco de corte de 9”	$4.28
12.7	Un (01) disco de desbaste de 9”	$5.00
13.7	Dos (02) sierras sanflex	$3.57
14.7	Cuarenta (40) pernos bristol de 1” x 1/4” de acero inoxidable	$21.42
15.7	Sesenta (60) pernos zincados 1” x 1/4”	$4.28
16.7	Tres (03) cortes y dobles planchas	$42.85
17.7	Un (01) riel industrial	$2.85
18.7	Una (01) llave térmica	$12.5
19.7	Tres (03) lijas de agua de 240	$1.60
20.7	Tres (03) lijas de fierro #80	$1.60
21.7	Pintura al horno	$89.30
22.7	Mano de obra para consola	$267.90
23.7	Total de costo de implantación	$1693.86 ó S/.4742.80

1.5 Alcance

Desarrollar el software de simulación en el lenguaje de programación Java bajo la plataforma J2SE (Java 2 Standard Edition).

Validar las versiones del sistema, corregir las fallas que este presenta, y realizar los cambios que sean necesarios.

Implementar el software de simulación en la Escuela de Submarinos, a la cual está dirigido el proyecto.

1.6 Organización

Marina de Guerra del Perú, Comandancia de la Fuerza de Submarinos. Armada peruana, cuya misión es mantener la soberanía territorial y del mar peruano (Mar de Grau). La MGP cuenta con una flota de submarinos repotenciados con última tecnología. Los oficiales con la especialidad de submarinista llevan un entrenamiento riguroso y complejo, ya que en ellos recae la gran responsabilidad de asumir el mando de un submarino. Es por ello que la formación que brinda la Escuela de Submarinos durante la calificación es crucial de manera que mientras el Simulador de Ataque sea lo más real posible permitirá que los oficiales incrementen aún más la gran eficiencia que poseen.

CAPITULO 2

MARCO TEORICO

2.1 Introducción

En este capítulo se presenta las definiciones que abarcan el concepto de Simuladores el cual es necesario conocer antes de explicar la razón de ser del Sistema, así como una introducción al Sonar de un Submarino y cómo interactúa con otros equipos que forman parte del control de operaciones.

Posteriormente se explica brevemente los conceptos del lenguaje de programación, sonido sintetizado y protocolos estándares de comunicación utilizados para el desarrollo del Software.

2.2 Simulador

Un simulador es un aparato que permite la simulación de un sistema, reproduciendo su comportamiento. Los simuladores reproducen sensaciones que en realidad no están sucediendo.

Para simular el comportamiento de los equipos de la máquina simulada se pueden recurrir a varias técnicas. Se puede elaborar el modelo de cada equipo, se puede utilizar el equipo real o bien se puede utilizar el mismo software que se ejecuta en el equipo real pero haciéndolo correr en un computador más convencional.

2.2.1 Modelo Conceptual de Simulación

Dale K. Pace [9] define que muchos de los problemas potenciales de simulación pueden ser evitados mediante calidad explícita de modelo conceptual de simulación.

Toda simulación es desarrollada con un propósito. Normalmente el propósito de simulación es enunciado en objetivos y requerimientos de simulación. Desdichadamente, esos requerimientos pueden estar incompletos, no claros, inconsistentes y algunas veces equivocados. Un modelo conceptual de simulación de calidad minimiza ese tipo de problemas.

El modelo conceptual de simulación describe cómo el desarrollador de simulación espera transformar objetivos y requerimientos de simulación a especificaciones, las cuales contengan detalles adecuados para que la simulación sea diseñada e implementada satisfactoriamente. Una clara y comprensiva comunicación entre expertos de aplicación de dominio y desarrolladores de simulación es esencial para obtener los requerimientos correctos.

> "...sin un modelo conceptual explícito, es mucho más difícil determinar si una simulación es apropiada para reutilizar o para usar en combinación con otras simulaciones en una simulación distribuida"
>
> (Dale K. Pace 2002: 9)

Existen tres componentes básicos en el modelo conceptual: (1) el contexto de simulación el cual identifica las limitaciones que la simulación debe acomodar, (2) el espacio de misión (con los elementos de simulación) que constituyen el aspecto de representación de la simulación, y (3) el espacio de simulación que define las características de control de la simulación.

2.3 Sonar

El principal sensor de un submarino es el sonar pasivo, el cual cuenta con un arreglo de hidrófonos que puede ser cilíndrico, de herradura o esférico, el cual es escaneado digitalmente por sistema electrónico de escucha que presenta en una pantalla gráficamente los ruidos captados con respecto al norte verdadero. Con ese sensor los ruidos de las hélices y maquinarias de los buques son detectados y analizados, determinando sus parámetros característicos permitiendo de esta manera, la clasificación de los mismos. Una vez clasificado el contacto es necesario poner al submarino en posición de ataque dentro del alcance de sus armas, por lo cual la información de la marcación de los ruidos recibidos es transferida al sistema de control de tiro para la obtención de los parámetros del blanco (rumbo, distancia y velocidad).

2.3.1 Componentes del Sonar

Un sistema de sonar consta básicamente de lo siguiente:

- Arreglo de hidrófonos
- Electrónica de scanning
- Interfaz de datos propios (rumbo y velocidad)
- Interfaz con el sistema de control de tiro.

2.3.1.1 Arreglo de Hidrófonos

El arreglo de hidrófonos recibe las señales para la evaluación pasiva y activa, consiste de un cilindro de acero inoxidable donde están instalados los hidrófonos. El interior y exterior del cilindro están revestidos de “absorbente” de sonido de material plástico. La figura 2.1 muestra un arreglo de hidrófonos similar al usado abordo.

Figura 2.1: Arreglo de Hidrófonos.

2.3.1.2 Electrónica de Scanning

Actualmente se utiliza como electrónica de scanning el modo pasivo que se describe a continuación:

2.3.1.2.1 Modo Pasivo

Este modo está siempre presente al conectarse el equipo. Este es el modo normal de operación, recibiendo y evaluando señales de un emisor de ruido.

Las señales obtenidas por los grupos de hidrófonos son amplificadas, obteniéndose así una salida de ganancia en la señal amplificada, según el nivel general de ruidos sea alto o bajo, respectivamente.

Las señales de voltaje son digitalizadas y procesadas para el análisis de la marcación.

2.3.1.3 Interfaz de datos propios

La interfaz de datos propios sirve para recibir la información de rumbo, que es la orientación de la proa del submarino con respecto al norte verdadero. Esta información es necesaria para la determinación de las marcaciones verdaderas (con respecto al norte) de los ruidos recibidos.

2.3.1.4 Interfaz con el sistema de control de tiro

Una vez determinadas las marcaciones verdaderas de los contactos recibidos por el sistema de sonar, es necesario transferir esta información de marcaciones al sistema de control de tiro, con el fin de determinar los parámetros del contacto.

2.4 Lenguaje de programación Java

En los viejos días de los lenguajes de computadoras (15 años atrás aproximadamente), los programas eran diseñados para correr mas o menos sobre un solo sistema operativo, y el nombre del juego estaba en crear aplicaciones que corran tan rápido como sea posible. Hace algún tiempo, la World Wide Web y Java cambiaron esa noción de sistemas operativos basados en entornos de lenguaje a plataformas independientes de los lenguajes de control de redes y sistemas. Java representa el mar de cambio de la programación distribuida y desarrollo de aplicaciones que están tomando lugar en la industria de hoy en día. Lenguajes como Java cambian radicalmente el puesto de la carga de computadores desde una computadora local de escritorio hacia servidores que entregan el contenido ejecutable a los usuarios.

Los programadores han aceptado a Java rápidamente debido a que provee todo lo que se necesita en los lenguajes modernos, incluyendo:

- Características de un lenguaje Orientado a Objetos.
- Multihilos.
- Recolector de basura (Administración automática de memoria).
- Independiente de la plataforma.
- Trabajo con redes y características de seguridad.
- Características de desarrollo Web / Internet.

Las aplicaciones Java, por otra parte, se parecen sospechosamente un programa C++. El único problema al escribir aplicaciones Java en este momento es que Java es un lenguaje interpretado y esos programas escritos en Java requieren la Maquina Virtual de Java para ejecutarse. Afortunadamente, esta en curso el trabajo de desarrollo de compiladores que permitan a las aplicaciones Java correr más rápidamente y más eficientemente.

2.4.1 Arquitectura Natural:

Una definición es:

> **"**Este es un termino que los diseñadores de lenguajes usan para describir lenguajes como Java que son verdaderamente portables a través de diferentes sistemas operativos. Los programas escritos en lenguajes de arquitectura natural corren típicamente bajo interpretes de código de bytes que son capaces de correr en cualquier tipo de computadora." (David H. Friedel Jr. and Anthony Potts 1996: 5)

2.4.2 Clases:

Una definición es:

> **"**Java utiliza la básica tecnología de objetos encontrada en C++, el cual fue influenciado por Smalltalk. Características orientadas al objeto son implementadas en esos lenguajes usando bloques básicos construidos llamados clases. Esencialmente, una clase es una estructura que permite combinar datos y el código. Una vez que una clase ha sido definida, puede ser usada fácilmente para derivar otras clases." (David H. Friedel Jr. and Anthony Potts 1996: 5)

2.4.3 Programación Distribuida:

Una definición es:

> **“**Este campo emergente del desarrollo de software envuelve las técnicas de escritura de programas que pueden ser ejecutadas a través de redes como la Internet. En la programación tradicional, las aplicaciones corren sobre una computadora en singular y solo los datos son distribuidos a través de la red. En la programación distribuida los programas pueden ser descargados desde la red y ejecutados al mismo tiempo”. (David H. Friedel Jr. and Anthony Potts 1996: 6)

2.4.4 Recolector de Basura:

Definición:

> **“**Esta es la técnica de administración de memoria que los programas Java utilizan para manejar la asignación dinámica de memoria, En los tradicionales lenguajes de programación como C y C++, la memoria para variables dinámicas y objetos (punteros), debe ser asignadas y des asignadas manualmente por el programador. En Java, la asignación de memoria es manejada por el entorno de ejecución de Java (Java runtime environment), Eliminando así la necesidad de asignación explicita de memoria y los punteros. El recolector de basura es una de las características por las cuales muchos programadores hacen referencia a que Java es un mejor y más seguro C++”. (David H. Friedel Jr. and Anthony Potts 1996: 6)

2.4.5 Maquina Virtual de Java:

Una definición:

> **“**Este es el código que sirve como el motor o él interprete para los programas Java. Cualquier aplicación que sea capaz de correr programas Java requiere el uso de la Maquina Virtual de Java (VM)”. (David H. Friedel Jr. and Anthony Potts 1996: 6)

2.5 Sonido Sintetizado en Java

Una definición de sonido:

> “El sonido es creado cuando algo vibra a través de un medio; donde este ultimo puede ser el aire, las vibraciones pueden venir de los parlantes de un computador, es así que los tímpanos recepcionan las vibraciones y mandan una señal al cerebro, el cual lo interpreta como sonido.” (David Brackeen, Bret Backer y Laurence Vanhelswé 2003: Cap. 4: 2)

Las vibraciones a través del aire crean fluctuaciones de presión. Fluctuaciones rápidas crean ondas sonido de alta frecuencia, permitiendo escuchar un sonido agudo. La cantidad de presión en cada fluctuación es conocida como amplitud. Alta amplitud causa que se escuche un sonido fuerte. La figura 2.2 muestra la composición de las ondas de sonido.

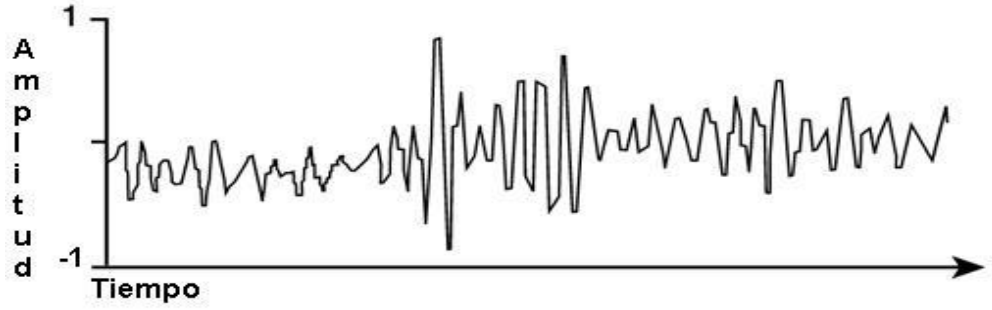


Figura 2.2: Las Ondas de sonido están compuestas de amplitudes cambiantes en el tiempo.

Un audio es codificado como una serie de sencillos (muestra de sonido) en un arreglo de bytes, los cuales son enviados a una clase Java hacia el mezclador. El contenido del arreglo de byte viene de un archivo de audio, así los efectos de sonido pueden ser modificados e incluso agregarlos al arreglo. Sin embargo en un audio sintetizado, las cosas no suceden así, para ello la siguiente definición:

> "En el audio sintetizado, las aplicaciones generan el arreglo datos byte sin requerir entrada de audio alguna. Potencialmente, cualquier sonido puede ser generado en tiempo de corrida". (Andrew Davison, 2005: Cap 10: 2)

Un audio es una mezcla de ondas sinusoidales, cada uno representando una nota o un tono. Una nota pura es una onda sinusoidal singular con una determinada amplitud y frecuencia.

La frecuencia puede ser definida como el número de ondas sinusoidales que pasan por un punto dado en un segundo. Una nota pura es una onda sinusoidal individual con una amplitud y frecuencia definida, y esa onda sinusoidal puede estar representada por una serie de muestras almacenadas en un arreglo de byte. La figura 2.3 muestra una conversión simple de análogo a digital.

Figura 2.3: Conversión simple de análogo a digital.

Una clase de fuente de datos línea en Java se configura para aceptar un formato de audio específico, la cual incluye una tasa de muestra. Por ejemplo, una tasa de muestra (sample

rate) de 21,000 causa 21,000 muestras para llegar al mezclador cada segundo. La frecuencia de una nota de 300 Hz, significa que 300 copias de esa nota llegaran al mezclador por segundo.

El número de muestras requeridas para representar una nota individual es una de las siguientes:

muestras / nota = (muestras / segundos) / (notas / segundo)

muestras / nota = tasa de muestras / frecuencia

Colocando como ejemplo una nota que necesite de 21,000 / 300 = 70 muestras. En otras palabras, la onda sinusoidal debe consistir de 70 muestras.

2.6 Datagramas UDP y Sockets

2.6.1 El Protocolo UDP

Una definición del protocolo de datagramas de usuario (UDP):

> "Es un protocolo alternativo para enviar datos sobre IP que es muy rápido, pero no confiable. Esto es por que cuando se envía data UDP, no existe forma de saber sí llegó, y mucho menos conocer sí los diferentes fragmentos de datos llegaron en el orden en el cual se envió. Sin embargo los fragmentos que llegan lo hacen mucho más rápido".
> (Elliotte Rusty Harold 2000: 38)

No es correcto usar el protocolo UDP para aplicaciones como FTP que requieren una transmisión de datos confiable sobre una red

potencialmente no confiable. Sin embargo hay una gran variedad de aplicaciones que requiere una mayor rapidez en comparación a la cantidad de bits correctos. Por ejemplo, en audio o video de tiempo real, la perdida o intercambio de paquetes simplemente aparece como estática, la cual es tolerable. Pero la espera de una petición TCP de retransmisión o la espera de un paquete son inaceptables.

La diferencia entre UDP y TCP a menudo se explican mediante una analogía con el teléfono y el sistema de correo postal. TCP es como el sistema telefónico. Cuando se llama a un número, el teléfono es preguntado y se establece una conexión entre las dos partes. Así como uno va hablando, se sabe que la otra parte escucha en el mismo orden. Sí el teléfono esta ocupado o no responde, entonces se vuelve a intentar. En contraste, UDP es como sistema de correo postal. Se envían paquetes de un correo hacia una dirección. Muchas de las cartas arriban, pero talvez muchas se pierdan en el camino. Las cartas probablemente arriben en el orden que se enviaron, pero es no es garantizado.

Ambos sistemas pueden ser usados. Cualquiera puede ser usado para muchos tipos de comunicaciones, en algunos casos definitivamente una es superior a otra. Lo mismo es verdad para UDP y TCP.

2.6.2 Datagramas UDP

La implementación de Java acerca UDP esta inmersa en dos clases: DatagramPacket y DatagramSocket. La clase DatagramPacket controla los bytes de datos dentro de paquetes UDP llamados datagramas y le deja el control de los datagramas que usted recibe. Un DatagramSocket envía tal y como recibe los datagramas UDP. Para enviar los datos, se colocan en un DatagramPacket y se envían usando un DatagramSocket. Para recibir los datos, se recibe un objeto DatagramPacket desde un DatagramSocket y entonces se lee el contenido del paquete. Los sockets mismos son muy sencillos de crear. En UDP, todo acerca de un datagrama, incluyendo la dirección a la cual se dirigen, esta incluida en el

mismo paquete; los socket solo necesitan saber el puerto local sobre el cual envían o escuchan.

UDP no tiene noción de un servidor de socket como TCP con las clases Socket y ServerSocket. En UDP se usa el mismo tipo de socket para enviar o recibir datos. Otra diferencia es que un solo DatagramSocket puede enviar o recibir datos desde muchos Host independientes. El socket no se dedica a una sola conexión como en TCP. De hecho, UDP no tiene conocimiento de algún concepto de conexión entre dos host; solo conoce de datagramas individuales. La figura 2.4 muestra la estructura de un data grama UDP.

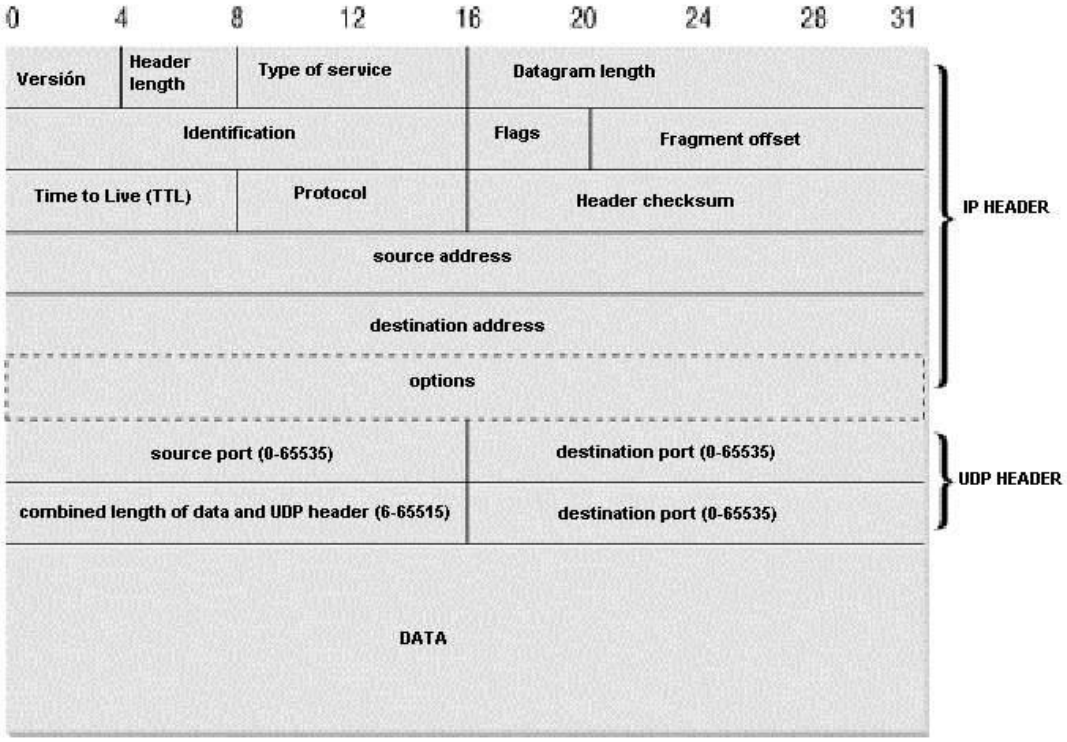


Figura 2.4: Estructura Data grama UDP.

2.6.3 Socket

Una definición de Socket es:

> “Un socket es una conexión entre 2 hosts (máquinas). Este puede realizar siete operaciones básicas:

- Conectarse a otra máquina
- Enviar datos
- Recibir datos
- Cerrar conexiones
- Hacer uso de un puerto
- Escuchar por datos entrantes
- Aceptar conexiones de maquinas remotas sobre el puerto usado" (Elliotte Rusty Harold 2000: 271)

La clase *Socket* de Java, la cual es usada por clientes y servidores, tiene métodos que corresponden a las primeras cuatro operaciones de la definición anterior. Las ultimas tres operaciones son necesitadas solo por el servidor, el cual espera por clientes que se conecten a el. Ellos son implementados por la clase *ServerSocket.*

2.7 Multihilos

Una definición de Hilo es:

> "Es simplemente la corrida de un proceso en ejecución. La forma más simple de explicar un hilo es decir que cuando una aplicación empieza por la invocación del método main, el código que es ejecutado esta corriendo en un hilo." (Mulholland y Murphy 2003: Cap 2: 2)

Deiteil [11] presenta la siguiente analogía:

> Así como el cuerpo humano realiza una gran variedad de operaciones en paralelo o de *forma concurrente o simultanea*. Por ejemplo, la respiración, la circulación de la sangre y la digestión pueden ocurrir simultáneamente. También las computadoras realizan operaciones concurrentes. Por ejemplo una computadora puede estar compilando

un programa, imprimiendo un archivo y recibiendo un correo electrónico por una red en forma concurrente.

2.7.1 Ciclo de vida de un hilo en Java

En cualquier momento se dice que un hilo esta en uno de varios estados de hilos que se muestran en la figura 2.5.

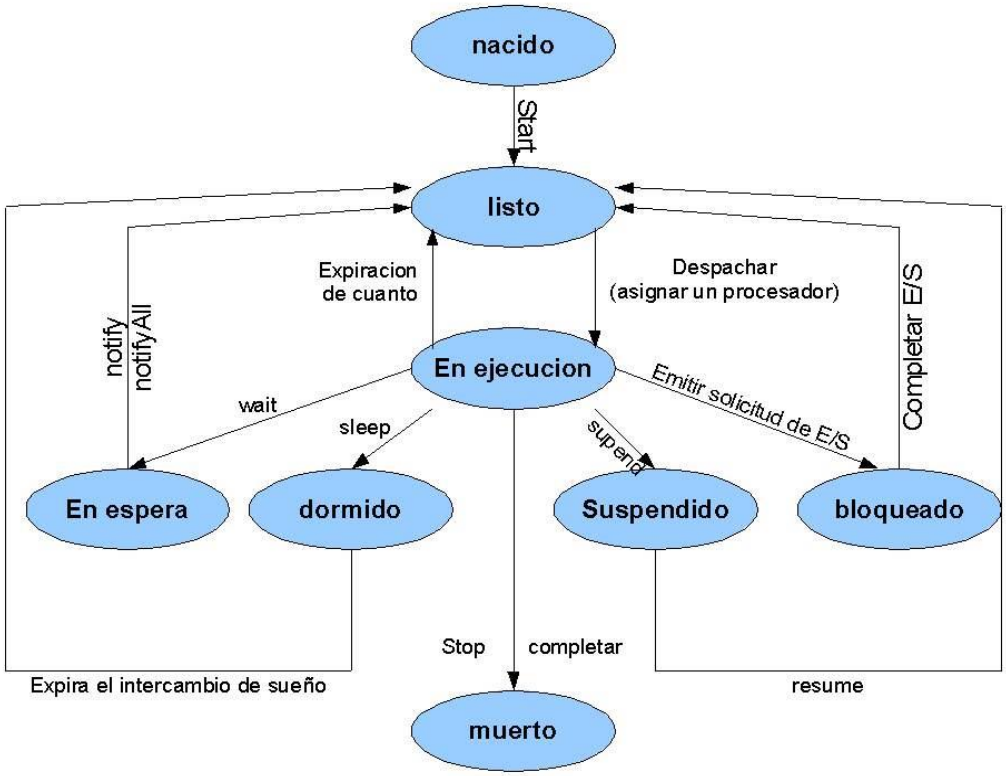


Figura 2.5: Ciclo de vida de un hilo.

Para entender el gráfico anterior, cuando creamos un hilo este se encuentra en el estado de *nacido*. El hilo permanece en ese estado hasta que se invoca al método *start* del hilo, esto hace que el hilo pase al estado de *listo*. El hilo listo de más alta prioridad pasa al estado de *en ejecución* cuando el sistema asigna procesador al hilo (el hilo empieza su ejecución). Un hilo pasa al estado *muerto* cuando termina su método *run* o cuando se invoca *stop* en algún momento entonces el sistema dispondrá del hilo muerto.

Un hilo en ejecución pasa al estado de *bloqueado* cuando el hilo emite una solicitud de entrada o salida. Un hilo bloqueado queda listo cuando la entrada o salida que estaba esperando termina. Un hilo bloqueado no puede usar procesador aunque haya uno disponible.

Cuando se invoca el método *sleep* de un hilo en ejecución, ese hilo pasa al estado de dormido. Un hilo dormido queda listo cuando expira el tiempo de sueño designado. Un hilo dormido no puede usar un procesador aunque haya uno disponible.
Cuando se invoca al método *suspend* de un hilo en ejecución, el hilo pasa al estado de suspendido. Un hilo suspendido queda listo cuando otro hilo invoca su método *resume* (reanudar). Un hilo suspendido no puede usar un procesador aunque haya uno disponible.

Cuando un hilo en ejecución llama al método *wait*, el hilo pasa al estado de espera, donde -*valga la redundancia*- espera en una cola asociada al objeto especifico para el cual invoco al *wait*. El primer hilo de la cola espera para un objeto en particular queda listo cuando otro hilo asociado a ese objeto emite una llamada al método *notify*. Todos los hilos de la cola de espera para un objeto dado quedan listos cuando un hilo asociado a esos objetos emite una llamada al método *notifyAll*.

2.8 El Protocolo NMEA 0183

Las siglas NMEA hacen referencia a la Asociación Nacional de Electrónica de la Marina (National Marine Electronics Association) que una asociación no lucrativa de manufactura, distribuidores, instituciones educacionales y otros interesados en ocupación periférica de electrónica de la marina.

Una definición es:

> “El estándar NMEA define una interfaz eléctrica y un protocolo de datos para comunicaciones entre instrumentación de marina”. (Klaus Betke 2001: 1).

NMEA es un estándar industrial voluntario, liberado por primera vez en marzo de 1983. Siendo actualizado cada cierto tiempo; la ultima liberación fue hecha en agosto de 2001 y esta disponible en las oficinas de NMEA.

NMEA también ha establecido un grupo de trabajo para el desarrollar un nuevo estándar para comunicaciones de datos entre dispositivos electrónicos en buques. El nuevo estándar, NMEA 2000, es un bidireccional, multi transmisor y multi receptor serial de datos en red.

Este estándar ha permitido la interfaz de equipos electrónicos marinos de diferentes marcas con la compatibilidad operacional entre ellos, tales como GPS, Radares, Sondas, Cartas Electrónicas, etc.

2.8.1 Interfaz Eléctrica

Los dispositivos NMEA son designados como cualquier transmisor o receptor, empleado una interfaz serial asíncrono con los siguientes parámetros:

2.1	Velocidad (Baud rate):	4800
3.1	Numero de bits de datos:	8 (bit 7 es 0)
4.1	Bit detenido:	1 (o más)
5.1	Paridad:	ningún
6.1	handshake:	ningún

NMEA 0183 permite un transmisor singular y varios receptores sobre un circuito. El cable recomendado para interconectar es el par trenzado blindado (shielded twister pair), con el blindaje a tierra solo en el transmisor.

2.8.2 Formato General de Sentencias

Todos los datos son transmitidos en forma de sentencias. Solo los caracteres ASCII son permitidos en la impresión, junto con CR (retorno de carro) y LF (salto de línea). Cada sentencia empieza con un “$” y termina con <CR><LF>. Existen tres tipos básicos de sentencias: sentencias del transmisor, sentencias de propietario y sentencias de consulta.

2.8.2.1 Sentencias del Transmisor

El formato general para la sentencia del Transmisor es:

$ttsss,d1,d2,.....<CR><LF>

Las primeras dos letras seguidas del “$” son el identificador del transmisor. Las siguientes tres (sss) son identificadores de sentencia, seguido por una coma que separa los números de campos de datos, opcionalmente puede seguir una suma de comprobación, termina con un retorno de carro y salto de línea. Los datos de los campos son únicamente definidos por cada tipo de sentencias. Un ejemplo de transmisor es:

$HCHDM,238,M<CR><LF>

Donde “HC” especifica al transmisor como si fuera el compás magnético, el “HDM” especifica él titulo del mensaje magnético seguido. El “238” es el valor de titulo, y “M” se designa como el valor de titulo como magnético.

2.8.2.2 Sentencias de Propietario

El estándar permite a fabricantes individuales definir formatos de sentencias de propietario. Estas sentencias inician con “$P”, luego tres letras de identificación del fabricante, seguido de cualquier dato que el fabricante desee, por ultimo el formato general del estándar de sentencias.

2.8.2.3 Sentencias de Consulta

Una sentencia de consulta es un significado para los receptores cuando realizan una petición en particular a un transmisor. El formato general es:

$ttllQ,sss,[CR],[LF]

Los primeros dos caracteres de los campos de la dirección son los identificadores del transmisor y los dos siguientes caracteres son lo identificadores del dispositivo que requiere la consulta. El quinto caracter es siempre una “Q” que define el mensaje como una consulta. El siguiente campo (sss) contiene las tres letras mnemónicas de la sentencia solicitada. Un ejemplo de sentencia de consulta es:

$CCGPQ,GGA<CR><LF>

donde el dispositivo “CC” (computadora) esta requiriendo al dispositivo GP (una unidad GPS) la sentencia “GGA”. El GPS entonces va a transmitir la sentencia una vez por segundo hasta que una consulta diferente sea solicitada.

2.8.3 Identificadores de Transmisor

Klaus Betke [4] muestra los siguientes identificadores:

AG: Piloto Automático – General
AP: Piloto Automático – Magnético
CD: Comunicaciones – Llamada Selectiva Digital
CR: Comunicaciones – Receptor
CS: Comunicaciones – Satélite
CT: Comunicaciones – Radio – Teléfono (MF/HF)
CV: Comunicaciones – Radio – Teléfono (VHF)
CX: Comunicaciones – escaneando receptor
DF: Direction Finder
EC: Electronic Chart Display & Information System (ECDIS)
EP: Emergency Position Indicating Beacom (EPIRB)
ER: Engine Room Monitoring Systems

GP: Sistema de Posicionamiento Global (GPS)
HC: Titulo – compás magnético
HE: Titulo – North Seeking Gyro
HN: Titulo – Non North Seeking Gyro
II: Instrumentación Integrada
IN: Navegación integrada
LC: Loran C
P: Código Propietario
RA: RADAR and/or ARPA
SD: Sondeo, profundidad
SN: Sistema de Posicionamiento Electrónico, otros/General
SS: Sondeo, escaneando
TI: Turn Rate Indicator
VD: Sensor de velocidad, Doppler, other/General
DM: Sensor de velocidad, log de rapidez, agua, magnético
VW: Sensor de velocidad, log de rapidez, agua, mecánica
WI: Instrumentos de clima
YX: Transductor
ZA: TimeKeeper – Atomic Clock
ZC: TimeKeeper – Chronometer
ZQ: TimeKeeper – Quartz
ZV: TimeKeeper – Radio Update, WWV o WWVH

2.9 El Protocolo IEEE 754

Existen 2 estándares IEEE diferentes para el punto flotante. IEEE 754 es un estándar binario que requiere $\beta=2$, $p=24$ para una precisión simple y $p=53$ para una precisión doble. También especifica la disposición de los bits en una simple y doble precisión. IEEE854 permite también $\beta=2$ o $\beta=10$ y no semejante al 754, no se especifica como los números de punto flotante son codificados dentro de bits. No se requiere un valor particular para “p”, pero en vez de eso especifica apremios sobre los valores permitidos de “p” para simple y doble precisión.

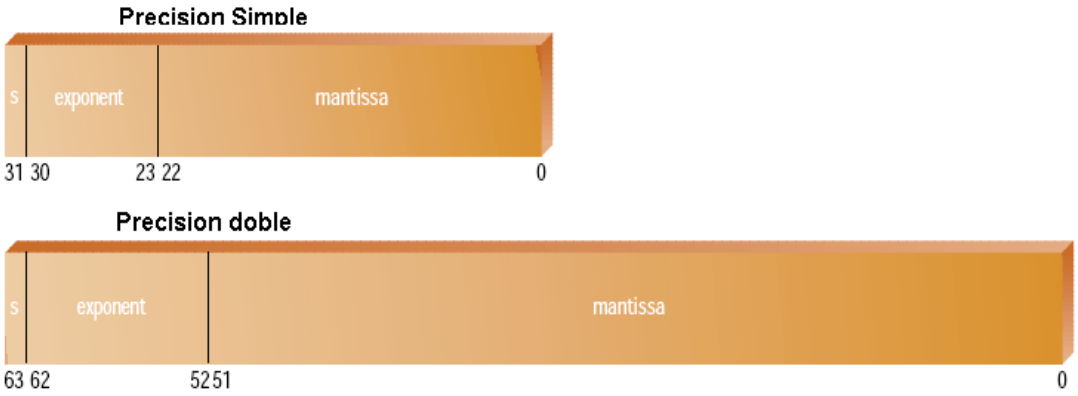


Figura 2.6: disposiciones de IEEE 754

2.9.1 Precisión

Según David Goldberg [6] el estándar IEEE define cuatro precisiones diferentes: simple, doble, simple extendida y doble extendida. En 754, simple y doble precisión corresponde a que hardware provee más punto flotante. Precisión simple ocupa una sola palabra de 32 bits, precisión doble dos palabras consecutivas de 32 bits. Precisión extendida es un formato que ofrece por lo menos una pequeña precisión y un rango extra.

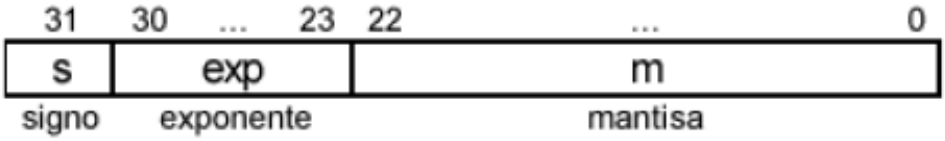
Parameter	Format			
	Single	Single-Extended	Double	Double-Extended
p	24	32	53	64
e_{max}	+127	1023	+1023	> 16383

Parameter	Format			
	Single	Single-Extended	Double	Double-Extended
e_{min}	-126	≤ -1022	-1022	≤ -16382
Exponent width in bits	8	≤ 11	11	15
Format width in bits	32	43	64	79

Tabla 2.1: IEEE 754 Formato de parámetros.

2.9.1.1 Precisión Simple

En precisión simple, para escribir un número flotante se usan 32 bits (4 bytes); 1 bit para el signo (s) del número, 23 bits para mantisa (m) y 8 bits para el exponente que se distribuyen de la siguiente forma:



El exponente se suele representar en exceso a 2^{n-1}-1, mientras que para la mantisa normalmente se utiliza signo magnitud. Adicionalmente la mantisa se suele normalizar colocando la coma decimal a la derecha del bit más significativo. A continuación se muestran dos ejemplos:

Ejemplo1: Para escribir el número $101110,01010111010000111110000111110001001 1_2$ en el estándar IEEE 754 con precisión simple, exponente de exceso 2^{n-1}-1 y mantisa en signo magnitud, para ello primero debemos normalizarlo:

$1,01110010101110100001111100001111100010011_2 \times 2^5$

el exponente en exceso 2^{n-1}-1 será:

$5_{10}+(2^{8-1}-1)_{10}=5_{10}+(2^7-1)_{10}=5_{10}+(128-1)_{10}=132_{10}=10000100_{EX.\ A\ 127}$

de la mantisa se cogen los 23 bits más significativos:

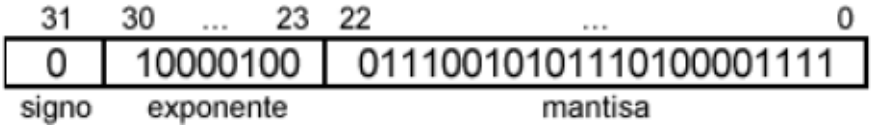
1,01110010101110000000111

el resto de los bits no se pueden representar debido a que no caben en la mantisa. Sin embargo, cuando la mantisa se normaliza situando la coma decimal a la derecha del bit más significativo, dicho bit siempre tiene el valor de 1. Por lo tanto, se puede prescindir de el, y coger en su lugar un bit de la

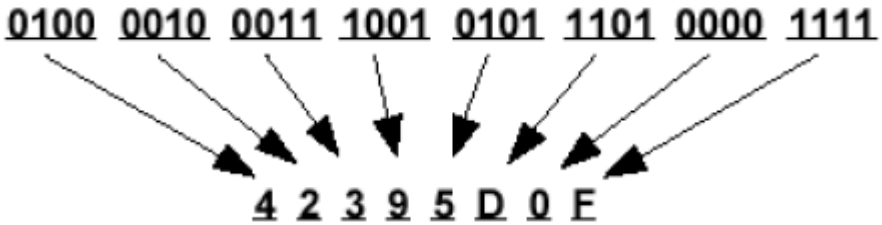
mantisa. De esta forma, la precisión del número representado es mayor. Así los bits de la mantisa son:

01110010101110100001111

Al bit omitido se le denomina bit implícito. Por otra parte, el bit de signo tiene el valor 0, ya que él numero es positivo. En consecuencia, el número puede ser representado como:



Para un mejor entendimiento los programadores representamos a los números flotantes en este formato, solemos usar el sistema hexadecimal quedando el número de la siguiente forma:



Por lo tanto:

$101110{,}01010111010000111110000111110001001{1}_{2} =$

$42395D0F_{CLF(PRECISION\ SIMPLE)}$

En este ejemplo los números no son exactamente iguales debido a que se realiza en precisión simple en donde no se representaron todos los bits de la mantisa.

2.9.2 Actuales Implementaciones de IEEE 754

Implementaciones actuales de la aritmética IEEE 754 pueden ser divididas dentro de dos grupos distinguidos por grado al cual ellos soportan diferentes formatos de punto flotante en hardware.

La poderosa arquitectura de IBM provee solo parcial soporte para precisión simple sin embargo se definen como sistemas simples /

dobles, incluyendo procesadores RISC, proveen total soporte para precisiones simples y dobles pero no soportan double precisión extendida.

Para ver la diferencia entre el comportamiento sobre un sistema basado en extendido sobre un sistema simple / doble, se considera un fragmento de código en versión C que se muestra a continuación:

```
int main() {
    double  q;

    q = 3.0/7.0;
    if (q == 3.0/7.0) printf("Equal\n");
    else printf("Not Equal\n");
    return 0;
}
```

Aquí las constantes 3.0 y 7.0 son interpretadas como números flotantes de precisión doble, y la expresión 3.0/7.0 hereda el tipo de dato “double”. Sobre un sistema simple / doble, la expresión será evaluada en precisión doble debido a que es el formato más eficiente para usar. Así, “q” se le asignara el valor 3.0/7.0 redondeado correctamente a la precisión doble. En la siguiente línea, la expresión 3.0/7.0 será evaluada nuevamente en precisión doble, y por supuesto el resultado será igual al asignado a “q”, entonces el programa imprimirá “Equal” como se esperaba.

Sobre un sistema extendido, incluso aunque la expresión 3.0/7.0 sea tipo “double”, el cociente será computado en un registro en formato “double” extendido, y así en el modo por defecto, será redondeado a precisión extendida “double”. Cuando el valor resultante es asignado a la variable “q”, entonces es almacenado en memoria y desde que “q” es declarada como “double” el valor será redondeado a precisión doble. En la siguiente línea, la expresión 3.0/7.0, el valor una vez mas será evaluado en precisión extendida obteniendo un resultado que difiere del valor almacenado en “q” con precisión doble causando que el programa imprima “Not equal”.

CAPITULO 3

ESTADO DEL ARTE

3.1 Introducción

En este capitulo se desarrolla el estado del arte en un tema que ha sido usado dentro del Simulador de Ataque de la Escuela de Submarinos, así como las tendencias que existen dentro de área de simulación a nivel mundial y de nuestra Marina de Guerra . El tema del estado del arte es:

1. Estado de arte relacionado con el entrenamiento de los sonaristas.

3.2 Entrenamiento de los Sonaristas

El problema tiene como núcleo que el simulador de ataque cuenta con un software limitado en funcionalidad y performance que permita el entrenamiento del personal en formación y el personal calificado como sonarista con realismo.

El sonarista no solo debe conocer la operación de reconocer los contactos que figuran en el sonar, sino también de enviar la información necesaria en el momento requerido a la computadora de control de tiro, así como informar al comandante las características del contacto escuchado.

El software de simulación de sonar debe recibir datos propios del software de control en tiempo real y transmitir las marcaciones al simulador del sistema de control de tiro.

Con el fin de brindar un marco conceptual del simulador en cuestión se presenta la Figura 3.1, que muestra la navegación de los distintos mundos que participan en el Simulador de Ataque, sobre este ultimo se generan ejercicios contactos

simulados dentro de una carta de navegación a fin de poner a prueba el conocimiento del personal de ataque dentro de una unidad submarina.

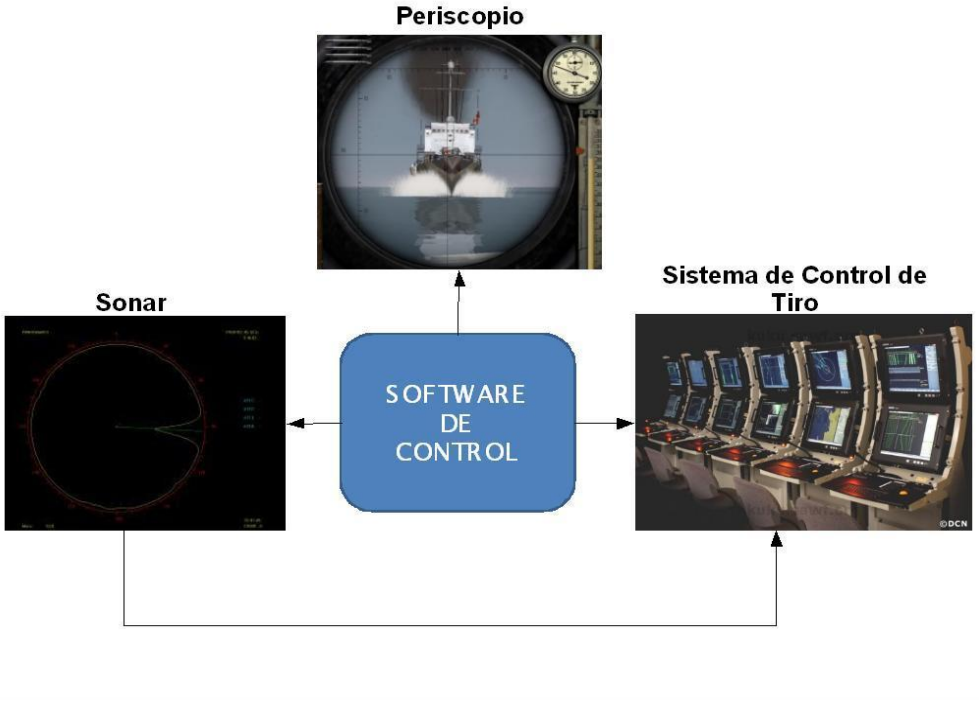


Figura 3.1: Interacción de los mundos

En la figura anterior se observa conceptualmente como esta constituido el simulador de ataque. Para entenderlo mejor una breve explicación.

El software de control es el que genera los contactos con sus características y distribuye la dinámica de movimiento a través de la red hacia los demás modulo. Se ingresan los datos propios (rumbo y velocidad), así como los datos de los contactos (velocidad, rumbo, distancia y nombre del buque); una vez ingresados los datos comienza el ejercicio.

El periscopio recoge la información que necesita para mostrar el buque contacto (3D) en la marcación correcta, de la misma manera el sonar toma la información necesaria para presentar el buque contacto en la pantalla y enviar las marcaciones en automático o manual a la computadora de ploteos.

La computadora de ploteos recibe información tanto del simulador de contactos como del sonar para realizar los cálculos matemáticos para determinar una

solución de ataque optima. Por otro lado la computadora de control de tiro establece la solución determinada para efectuar un lanzamiento de torpedo hacia el buque enemigo.

3.3 Desarrollo de un caso ejemplo de operación del Sonar

El software de simulación de sonar se carga automáticamente al iniciar el sistema operativo y se presenta la pantalla de un sonar que se muestra en la figura 3.2. El rumbo propio es visualizado por una pequeña línea de color verde, que parte del centro de la circunferencia y se extiende en este caso hacia el rumbo 045 aproximadamente. También se puede visualizar el cursor representado por una línea de color verde que parte desde el centro de la circunferencia hacia el borde de al misma, que permite al sonarista desplazarse hacia un ruido en particular

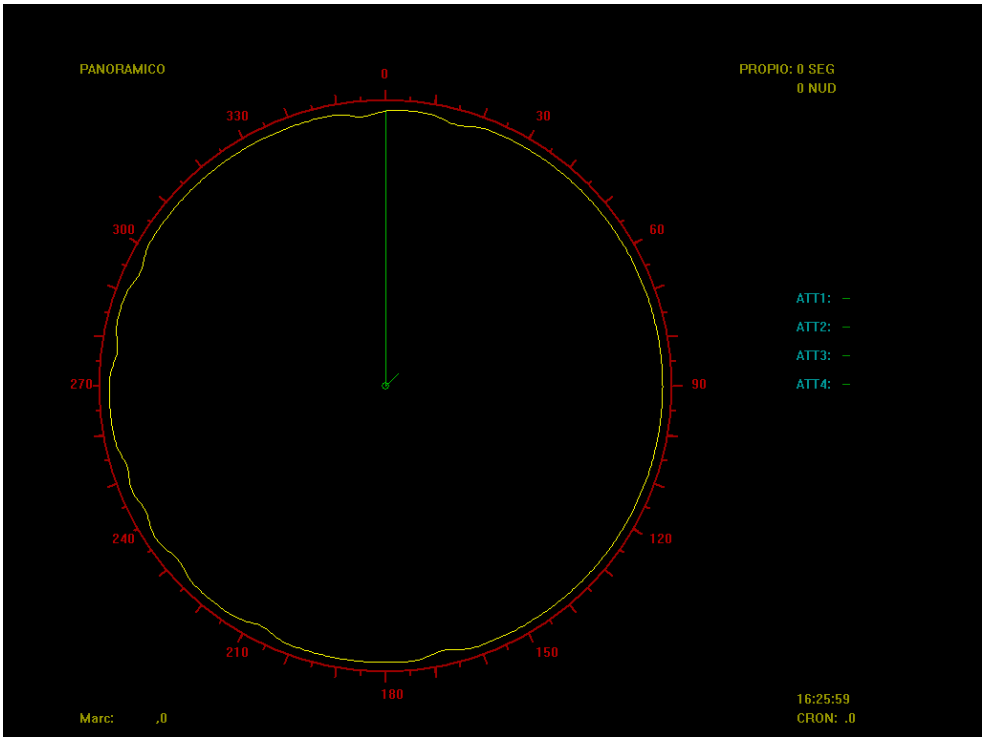


Figura 3.2: Sonar en Standby.

Cuando el sonar detecta un buque contacto lo visualiza en pantalla como muestra la figura 3.3, en donde se observa un contacto en una marcación

cercana al 090. Para escuchar al contacto el sonarista debe desplazar y colocar el puntero sobre la campana generada, acto seguido el sonarista puede enviar en manual o automático las marcaciones a la computadora de ploteo por medio de un canal de transmisión de datos, así como también puede determinar la velocidad aproximada de buque contacto de acuerdo a las revoluciones generadas por el batido de hélices efectuados en un tiempo determinado.

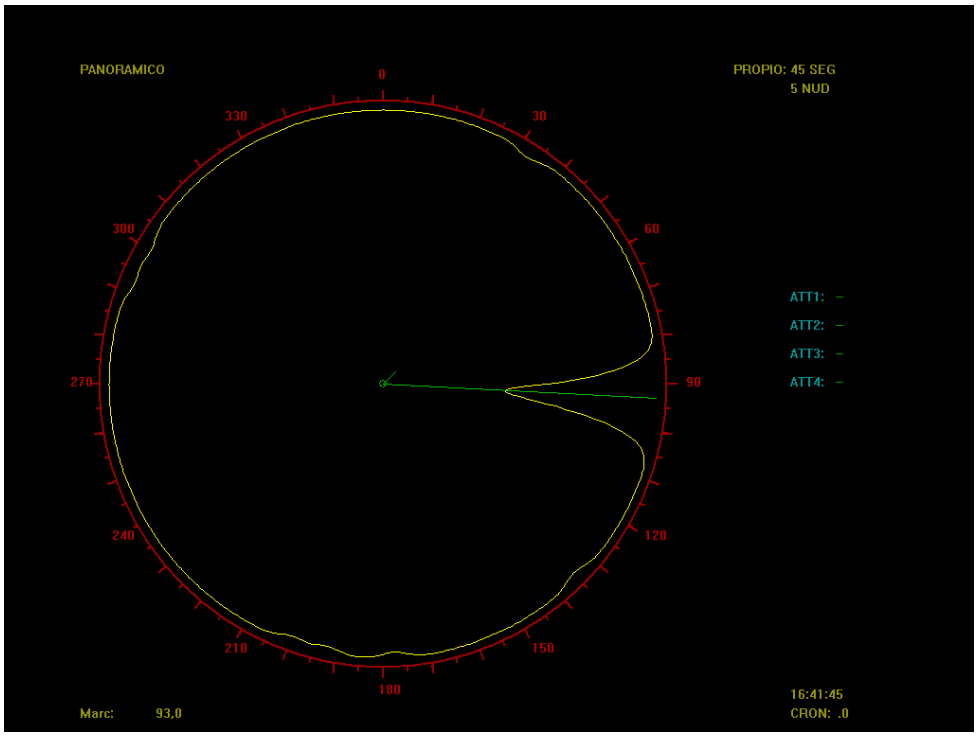


Figura 3.3: Sonar con un contacto en pantalla.

Siendo este tipo de software, herramientas de uso exclusivo para la milicia al rededor del mundo y teniendo en cuenta que la información acerca de cómo están compuestos, no es accesible; efectuar un benchmarking no es posible.

3.4 Tendencias

Según el Instituto Danés de Tecnologías, a medida que las tecnologías de simulación se hacen más sofisticadas y más rentables a la hora de desarrollar, y con el aumento del énfasis en la evaluación de su eficacia, es probable que siga

creciendo el desarrollo de simuladores en cada vez mas áreas de investigación y desarrollo, sobre todo para el entrenamiento de personal militar.

En nuestro país, especialmente en nuestra Marina de Guerra el desarrollo de simuladores para el entrenamiento de su personal también va en aumento, esto debido a los resultados obtenidos que demuestran su eficiencia, así como por los esfuerzos que viene desarrollando esta institución por dar a conocer la necesidad de apoyo por parte del gobierno. Es por ello que es muy probable que todas las instituciones militares y otras de nuestro país comiencen a desarrollar herramientas de simulación para entrenamiento, así como hacer uso de herramientas de código abierto como lo vienen haciendo países vecinos.

CAPITULO 4

APORTE

4.1 Introducción

De la revisión del software de simulación de sonar presentado en el capitulo anterior y la observación del equipo original de sonar a bordo de las unidades submarinas he podido concluir que, el sistema para entrenamiento de sonaristas tiene algunas complicaciones, siendo una estas la total diferencia entre el equipo de sonar a bordo de las unidades submarinas y la interfaz gráfica usada para el software, carece de una rutina de actualización de ruido generado por los parámetros de un buque contacto (como la velocidad), no permite efectuar las otras dos pruebas que realiza en sonar original y por ultimo no cuenta con una interfaz electrónica adecuada para la simulación.

4.2 Aporte Teórico

Por estos motivos, este trabajo está orientado a mejorar el simulador de sonar para que éste tenga todas las características y la mayor similitud del sonar a bordo de las unidades submarinas, con el fin de entrenar a los sonaristas en la operación del equipo.

Con el fin de brindar un marco conceptual del sistema en cuestión se presenta la figura 4.1 , que muestra la interacción de los distintos mundos que participan en el sistema que genera un ejercicio de ataque con el fin de mantener entrenado al personal submarinista, sin exponerlos a riesgo alguno y la figura 4.2 que muestra un diagrama de bloques con la secuencia y flujo de datos entre el sonar, el software de control y el sistema de control de tiro.

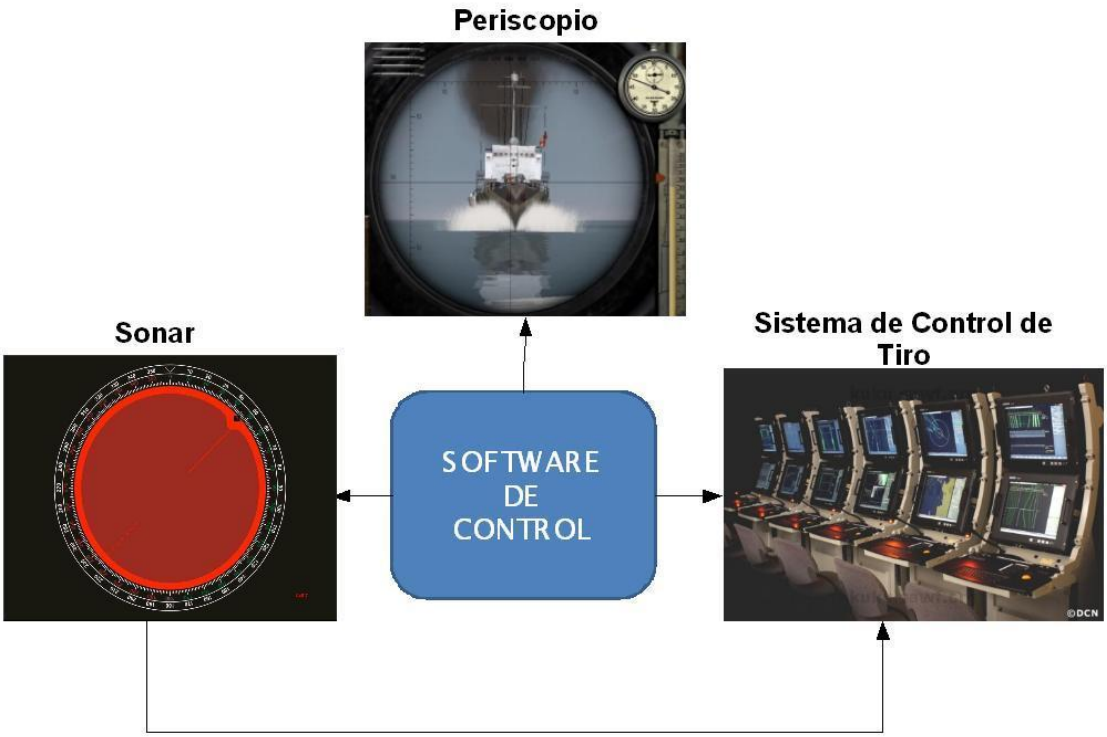


Figura 4.1: Interacción de los mundo con el Simulador de Sonar actual.

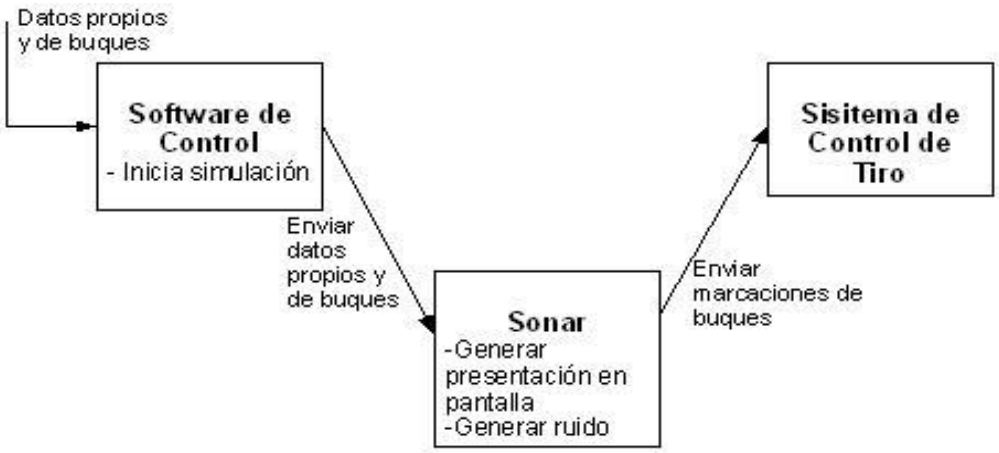


Figura 4.2: Secuencia de datos – SC, Sonar y SCT

En este trabajo se buscó subsanar todas las falencias que presenta el software de simulación de sonar anterior. A continuación una breve descripción del contenido de este software de simulación.

Una implementación nueva realizada en esta tesis es el uso de Hilos (procesos) basados en la clase “Thread” de Java para aquellas clases que heredan de esta clase y la Interfaz “Runnable” para aquellas clases que heredan de otras, pero necesitan usar hilos, todo esto con el objetivo de manejar cálculos, dibujos, envió y recepción de datos.

Existen cinco hilos o proceso dentro de la clase principal; un primer Hilo que maneja el dibujo de la pantalla del sonar y a demás esta sincronizado con otro Hilo de una clase “DatagramSocket” que maneja la recepción del rumbo propio y velocidad propia en el estándar de comunicaciones NMEA. El segundo Hilo, sincronizado con otro hilo de una clase “DatagramSocket”, que maneja los datos del buque contacto bajo el estándar IEEE 754 y a demás realiza el cálculo de la campana de Gauss, así como el cursor cuando se encuentra enganchado con un canal de transmisión de datos de contacto.

El tercer Hilo realiza los cálculos y dibujo de las campanas de Gauss cuando se ejecuta la prueba digital y dibuja el cursor cuando se engancha a cualquiera de estas campanas.

El cuarto Hilo maneja los sonidos de acuerdo al estado del sonar y al modo de operación. El quinto y último Hilo tiene como función enviar las marcaciones del buque contacto hacia la computadora de ploteos por medio de un DatagramSocket, haciendo uso del estándar IEEE 754 y envía los datos propios por el puerto de comunicaciones COM1 usando el estándar de comunicaciones NMEA.

El desarrollo del software se dividió en varias fases representadas en la figura 4.3 y descritas a continuación:

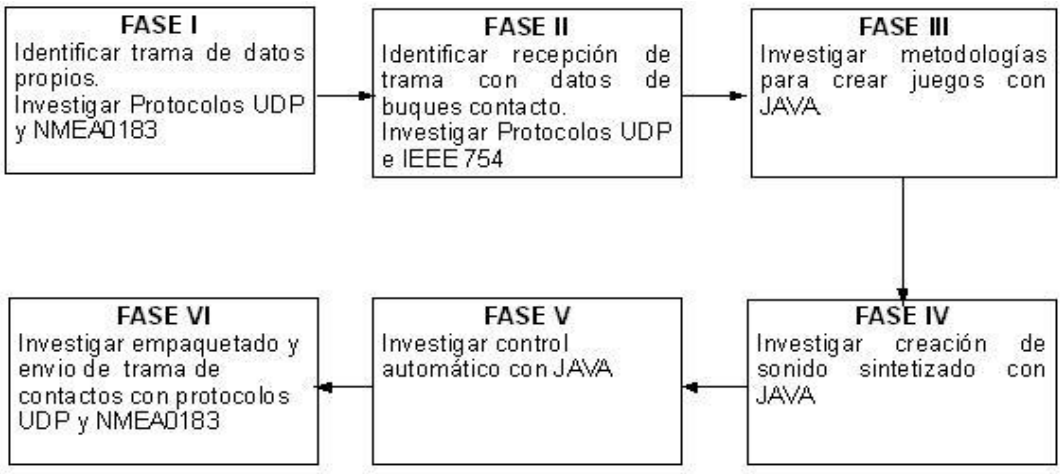


Figura 4.3: Fases de Desarrollo e Investigación.

La primera fase del desarrollo fue identificar la trama enviada por el software de control que contiene los datos propios para establecer la comunicación básica. Para ello fue necesario investigar el protocolo de comunicaciones NMEA 0183, así como verificar el puerto de comunicaciones por el cual el software de control emite dicha trama. A continuación se detalla el procedimiento mediante el cual se obtienen los datos propios que viajan por Ethernet haciendo uso de los protocolos UDP y NMEA.

La clase encargada de este proceso lleva como nombre “Server” y hereda de la clase Thread que le permite la ejecución de un hilo exclusivo para la recepción de los datos antes mencionados. Instanciando un objeto de la clase DatagramSocket se obtiene una conexión para recibir paquetes mediante un puerto de comunicaciones específico; luego es necesario declarar un arreglo de bytes en donde se va a almacenar la trama deseada, cada vez que se reciben datos. Acto seguido se instancia un objeto de la clase DatagramPacket pasándole como parámetros el arreglo de bytes antes mencionado y su longitud. El siguiente paso es ubicar los datos propios en las posiciones correctas del arreglo de bytes, extraerlos para luego ser almacenados en las variables respectivas, con el fin de enviarlas a la clase correspondientes.

Como segunda fase de desarrollo fue necesario identificar y conocer la trama con los datos de los buques contacto que transmite el software de control, con el fin de obtener la comunicación completa. Para ello fue necesario investigar el

uso del protocolo IEEE 754 para comunicaciones industriales, así como verificar una vez mas el puerto de comunicaciones correcto por el cual el software de control emite dicha trama. A continuación el detalle del procedimiento mediante el cual se obtiene los datos de los buques contacto que viajan por Ethernet haciendo uso de los protocolos UDP e IEEE 754.

La clase encargada de este proceso lleva el nombre de “ClienteBlanco” y hereda de la clase Thread. Instanciando un objeto de la clase DatagramSocket se obtiene una conexión para recibir paquetes mediante un puerto de comunicaciones específico; luego es necesario crear una matriz de tipo “float” con la capacidad de almacenar todos los contactos recibidos y sus parámetros. A diferencia de la clase “Server” esta clase cuando recibe los paquetes enviados por la red, debe crear e instancia un objeto de la clase ByteBuffer en donde se ordenan los datos en punto flotante. Acto seguido se extraen los datos requeridos de cada contacto en las posiciones correctas del Bytebuffer y se colocan en la matriz de contactos. Por ultimo se envía toda la matriz a la clase correspondiente.

El objetivo de la tercera fase fue investigar las metodologías para crear juegos en 2D haciendo uso del lenguaje Java, con el fin de realizar las operaciones y técnicas necesarias para simular visualmente el sonar original. Existen tres imágenes de fondos diseñadas a la medida original, una cuando el sonar se encuentra en OFF, otra para trabajar con rumbo verdadero y la ultima para cuando se encuentra con rumbo relativo. El siguiente paso fue colocar en la pantalla el cursor que permite posicionarse sobre los contactos, el cual esta representado por una línea que parte casi del centro de la pantalla hacia el borde de la circunferencia; para ello se hace uso del sistema de coordenadas polares. A demás el cursor obedece a una orden de la palanca en el panel de control del sonar para moverse ya sea hacia la derecha o izquierda.

Para obtener las coordenadas polares de los puntos que representan los píxeles desde los cuales se traza la línea que representa el cursor, están dadas por la siguiente formula:

x2 = 512 + (radio) * SEN(ángulo))

y2 = 384 - (radio) * COS(ángulo))

x3 = 512 + (radio/3.5) * (SIN(ángulo))

y3 = 384 - (radio/3.5) * COS(ángulo)

Donde “radio” es el radio de la circunferencia de la pantalla y “ángulo” representa en alguno de la marcación del cursor. Este ultimo empieza en el norte (000) y cada vez que se gira la palanca en el panel de control del sonar, este ángulo incrementa o decrementa en 0.1 dependiendo de si el movimiento de la palanca es hacia la izquierda o derecha. La figura 4.4 muestra una referencia a cómo queda el dibujo en la pantalla.

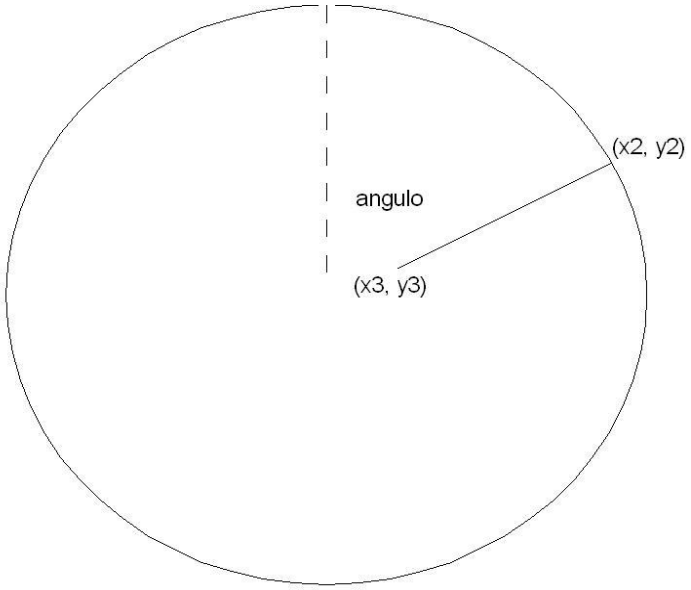


Figura 4.4: Dibujo del cursor en pantalla.

Para obtener las coordenadas polares de los puntos que representan los píxeles desde los cuales se traza la línea segmentada que representa el complemento del rumbo propio, están dadas por la siguiente formula:

x = 512 + radio * SEN(rumbo)

y = 384 - radio * COS(rumbo)

x1 = 512 + (radio/2) * SEN(rumbo)

y1 = 384 - (radio/2) * COS(rumbo)

Donde “radio” es el radio de la circunferencia de la pantalla y “rumbo” representa el ángulo de la proa de buque propio. Este ultimo varia con respecto a los datos

de la dinámica de movimiento del buque propio enviados a través de la red por el software de control. La figura 4.5 muestra una referencia a cómo queda dibujado el rumbo propio en la pantalla.

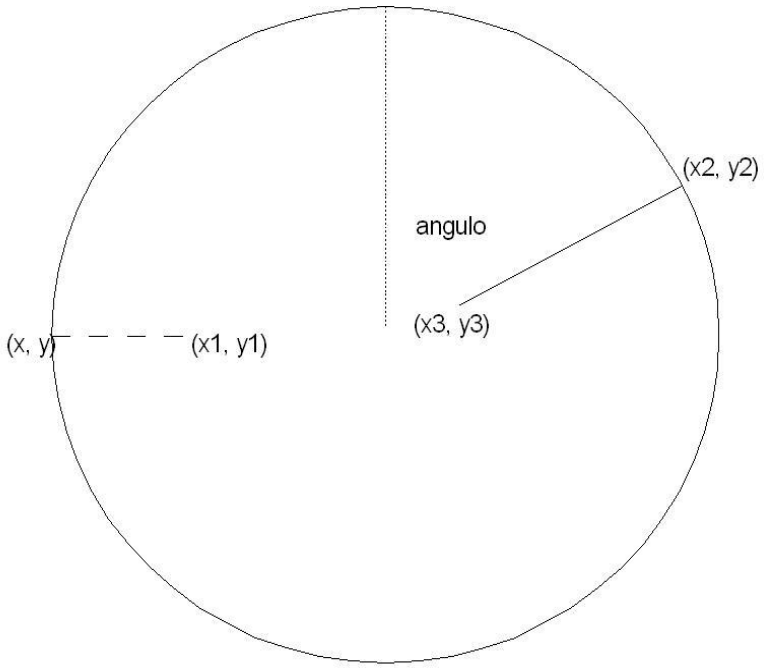


Figura 4.5: Rumbo propio al 090.

Con respecto a la visualización de buques contacto, se realizó una formula que permite dibujar una campana de Gauss con una altura variable, proporcional a la distancia y velocidad del buque contacto. Esta campana se dibuja línea por línea, variando el largo de cada línea exponencialmente, multiplicando por un factor de amplitud correspondiente a la altura de la campana hasta obtener la forma esperada. Existe un método dentro de la clase SimuSonar que recibe el largo de cada línea y el lugar donde debe colocar dicha línea, todo esto para guardar cada coordenada dentro de dos arreglos (un arreglo que guarda las (x, y), y el otro que guarda las (x1, y1)). El proceso repetitivo esta dado por las siguientes formulas:

FactAmp = Altura calculada en función de la velocidad y distancia

Largo = FactAmp * EXPONENCIAL(-xlargo * xlargo) donde xlargo se incrementa

para obtener las coordenadas de cada línea para dibujar la campana se utilizan las siguientes formulas:

xa = 512 + (radio) * (SEN(punto))

ya = 384 - (radio) * (COS(punto))

xb = 512 + (radio-largo) * (SEN(punto))

yb = 384 - (radio-largo) * (COS(punto))

Donde “punto” representa el lugar exacto donde se debe colocar la coordenada. La figura 4.6 muestra el dibujo de un contacto sobre la marcación 180. Para brindar una idea mas clara del dibujo, las líneas que conforman la campana se han colocado más anchas y en menor cantidad.

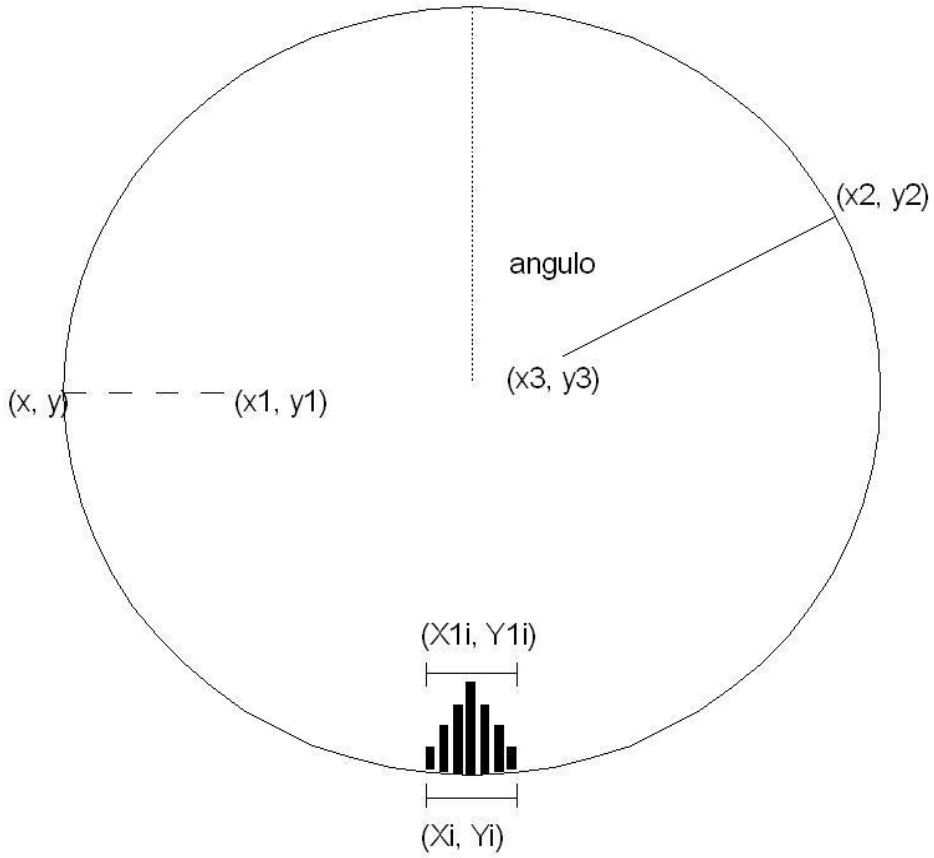


Figura 4.6: Dibujo ejemplo de un contacto en marcación 180.

La cuarta fase fue la mas difícil de todas, pues en esta caía la tarea de genera los ruidos o batido de hélices de los buques contacto generados por las revoluciones, de acuerdo a la distancia en que se encuentra con respecto al buque propio y la velocidad que contengan. El primer paso fue investigar audio sintetizado en el lenguaje de programación Java. Cumplida esta tarea se procedió al análisis de ruidos reales grabados en navegaciones a bordo. Para ello se utiliza un software que permite visualizar el espectro generado por los

sonidos reales, reflejando un conjunto de ondas sinusoidales con variaciones de amplitud. Una vez reconocido esto, la idea para construir las formulas que permitieran generar ruido para cada contacto se aclararon.

Cada sonido que se genera es almacenado en un objeto de la clase ShortBuffer para luego será reproducido en la clase correspondiente. Las siguientes formulas generan el ruido del batido de hélices para un buque contacto de acuerdo a sus parámetros.

VALOR=RAIZ[-2*LOG(ALEATORIO)*COS(2*PI*ALEATORIO)]+RAIZ[2*LOG(ALEATORIO)*SIN(2*PI*ALEATORIO]

VALOR2 = VALOR

sound = sound + velocidad

factor = 16000.0*0.12

u = 200+frecuencia*(SIN(sound*PI/180))

VALOR = u*VALOR

SONIDO=VALOR*(distancia/Rate)+(VALOR2*factor)

La variable “VALOR” almacena la formula que permite generar el ruido blanco (ruido del mar), luego la variable “VALOR2” es una copia de “VALOR”, debido a que valor será modificado, pero es necesario guardar su valor original. La variable “sound” es un acumulador para la propagación del sonido, mientras que la variable “factor” es la cantidad o tasa de muestras. Existe un condicional para la variable “u” que permite variar la frecuencia de acuerdo al numero de palas del buque contacto, la función de esta variable es para modificar una onda sinusoidal en particular. Ahora se modifica el valor de la variable “VALOR” multiplicándola por la variable “u” con el fin de establecer el valor de una pala en particular de la hélice de un buque. Por ultimo la variable “sonido” guarda el valor de una onda sinusoidal aplicándole la amplitud, la distancia y sumándole el ruido blanco para ser concatenado en el objeto de la clase ShortBuffer.

La quinta fase estuvo compuesta por la investigación de control automático en Java a través del puerto paralelo o también conocido como LPT1, para el manejo de las luces del panel de control del simulador de sonar. Para hacer un control automática con Java es necesario conocer el JNI (Java Native Interface), el cual es un marco de programación que permite utilizar librerías de lenguajes de programación tales como C, C++ y Assembler. La clase de nivel mas bajo utilizada para este propósito lleva como nombre “ParallelPort” y es proporcionada por la API de comunicaciones de Java (JavaComm). Esta clase contiene dos métodos nativos que tiene interacción con una DLL en el sistema operativo, así como un método de lectura y otro de escritura, cuya función es llevar a un nivel superior los método nativos de la misma. La clase “ControlLuces” contiene dos métodos que permiten encender y apagar las luces del panel de control del sonar; para ello contiene un objeto de la clase “ParallelPort”, el cual se instancia pasándole como parámetro la dirección de memoria del puerto LPT1, luego se lee un byte desde el pin del estado del puerto por medio del método “read” de la clase “ParallelPort” y finalmente se escribe un byte con el pin de datos respectivo para encender el foco correcto por medio del método “write” de la clase “ParallelPort”. La clase “SimuSonar” contiene un objeto de la clase “ControlLuces” que permite enviar la orden para encender o apagar las luces del panel de control del simulador de sonar cada vez que se genera un evento en las teclas correspondientes. La figura 4.7 muestra un esquema de control de luces similar al usado.

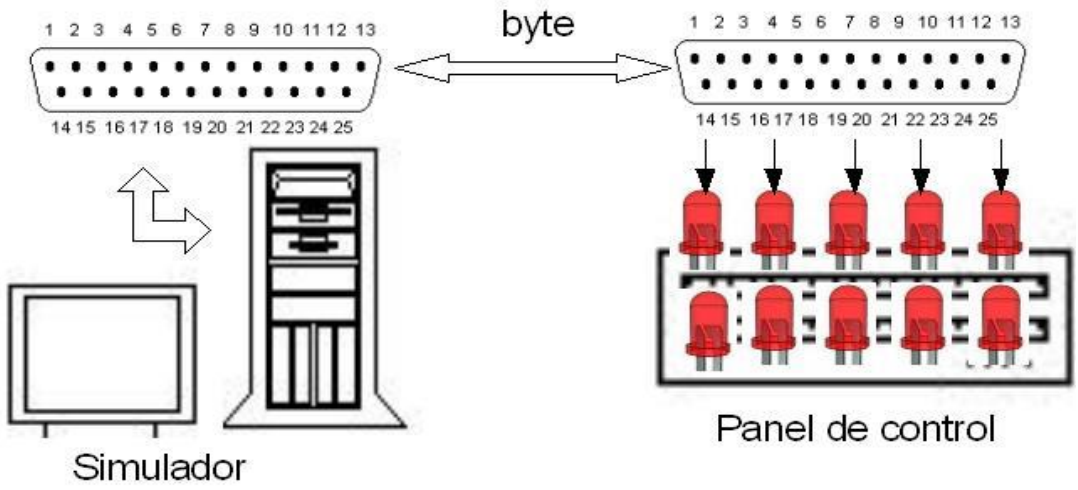


Figura 4.7: Control de luces vía LPT1.

Como se puede observar en la figura 4.6 el simulador lee un byte desde el pin del estado del puerto, para luego enviar un byte con el número y orden en el que se deben encender o apagar los LED en el panel, todo esto vía puerto paralelo (LPT1).

La sexta fase esta comprendida por el envío de datos (marcaciones de los contactos) hacia la computadora de ploteos. Para ello solo hacia falta conocer el puerto de comunicaciones, el lugar exacto en donde colocar los datos en la trama basada en el protocolo de comunicaciones IEEE 754 para luego ser enviada hacia la computadora de ploteos y que ésta la reciba correctamente. Los datos pueden ser enviados en manual mediante un evento en el panel de control del simulador de sonar o en automático cada quantum de tiempo.

La clase encargada de enviar los datos hacia la computadora de ploteos lleva como nombre “SonarServer”; esta clase crea e instancia un objeto de clase “DatagramSocket”, con el fin de establecer una comunicación mediante el protocolo UDP. El método “enviarTrama” recibe la marcación en tipo de dato real (double) y el número de canal de transferencia mediante un byte, luego crea un arreglo de byte. Este último es enviado como parámetro a través del método “wrap” de un objeto de la clase “ByteBuffer”, en donde se colocan los datos a enviar haciendo uso del protocolo IEEE 754 para puntos flotantes. El siguiente paso es crear un objeto de la clase “DatagramPacket” pasándole como parámetros el arreglo de byte, la longitud de dicho arreglo, la dirección IP para broadcast y el puerto de comunicaciones por el cual escucha la computadora de ploteos. Por ultimo se invoca al método “send” enviándole como parámetro el objeto de la clase “DatagramPacket” el cual contiene la trama con los datos requeridos.

4.3 Aporte Practico

El software de simulación del Sonar fue desarrollado con el lenguaje de programación Java bajo la plataforma J2SE (Java 2 Standard Edition) el cual es un lenguaje orientado a objetos.

Al software simulador se le asigno el nombre de SimuSonar (Sistema Simulador del Sonar de Submarinos).

La primera etapa en el desarrollo de este Sistema esta compuesta por un breve análisis que marca el alcance del mismo.

4.3.1 Diagrama de casos de uso

La figura 4.8 muestra el diagrama de casos de uso con el actor representado por el personal en capacitación o calificado como sonarista y los casos de uso del sistema correspondientes.

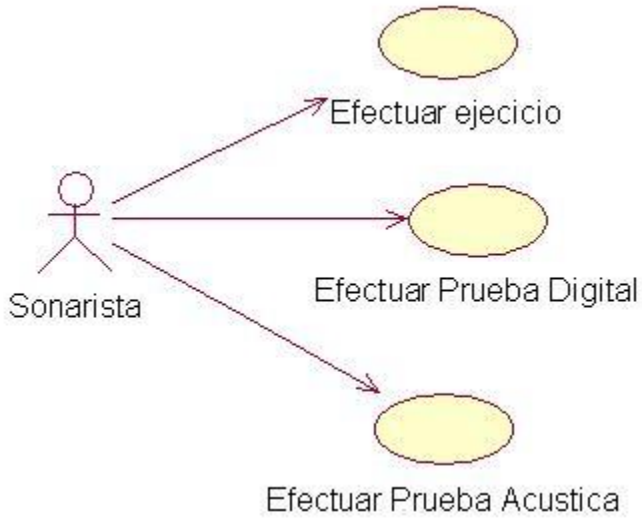


Figura 4.8: Diagrama de Casos de Uso.

El caso de uso “Realizar ejercicio” hace referencia al uso del sonar en su forma pasiva en donde se visualiza, escucha, determina la velocidad aproximada y envía datos del buque contacto. El caso de uso “Realizar Prueba Digital” hace referencia a las pruebas de funcionalidad del sistema digital y finalmente en el caso de uso “Realizar Prueba Acústica” hace referencia a la prueba que se realiza al arreglo de Hidrófonos, siendo esta la manera de determinar si funcionan correctamente.

4.3.2 Diagrama de clases

La figura 4.9 muestra la representación del sistema bajo un diagrama de clases en UML.

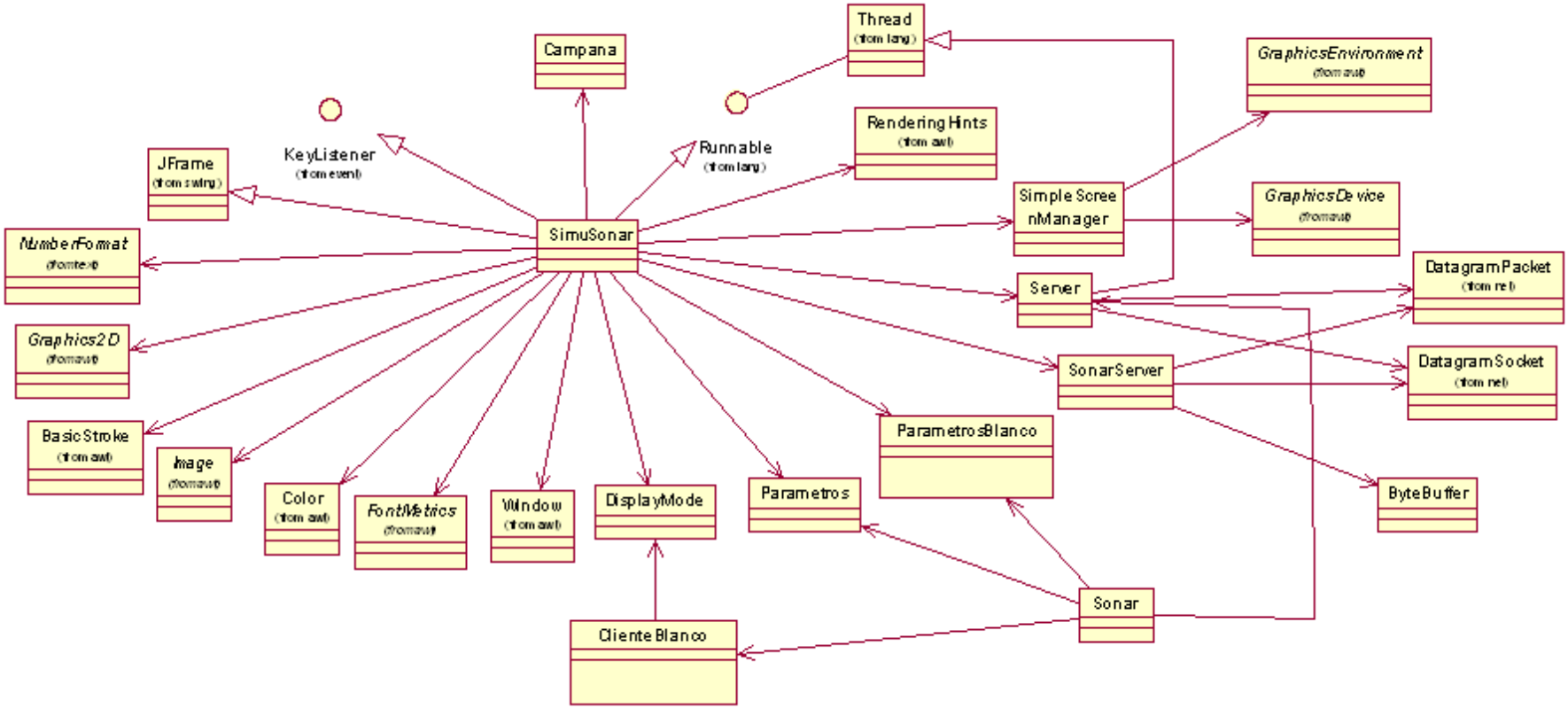


Figura 4.9: Diagrama de Clases.

A continuación se relata la descripción del software de simulación.

4.3.3 Pantalla Principal

El software de simulación de sonar se pone en funcionamiento cuando se inicia el sistema operativo de la computadora. La figura 4.10, muestra la interfaz de usuario principal en estado de apagado (OFF). Este estado permanece así, hasta que el usuario cambie de lugar la perilla correspondiente en el panel de control del sonar.

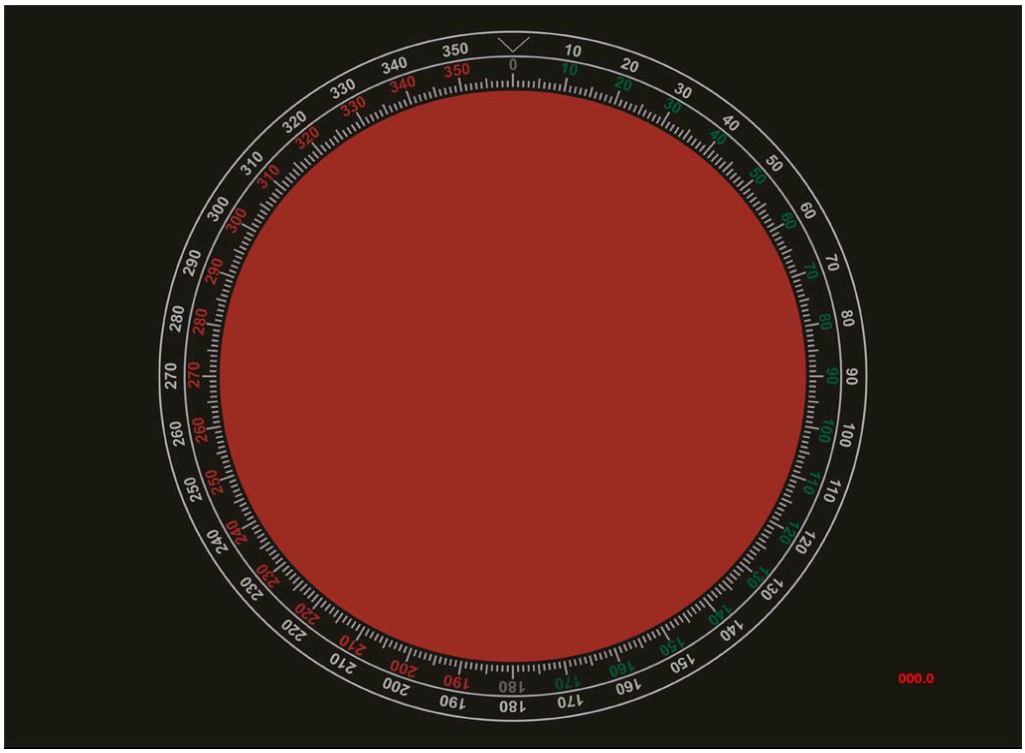


Figura 4.10: Sonar en estado apagado (OFF).

4.3.4 Equipo en operación pasiva sin contacto

Si el software de control no ha iniciado un ejercicio de simulación y el sonarista gira la perilla a modo de operación pasiva sin contacto, la interfaz de usuario cambia, mostrando solo el cursor mediante una línea en color naranja y el ruido propio al borde de la circunferencia con una barra circular tal y como lo muestra la figura 4.11. Durante este estado el usuario solo puedo mover el cursor haciendo uso de la palanca de movimiento ubicada

en el panel de control de sonar. Adicionalmente se puede visualizar en la parte inferior derecha, la marcación en la que se encuentra situado el cursor del sonar mediante números en color rojo.

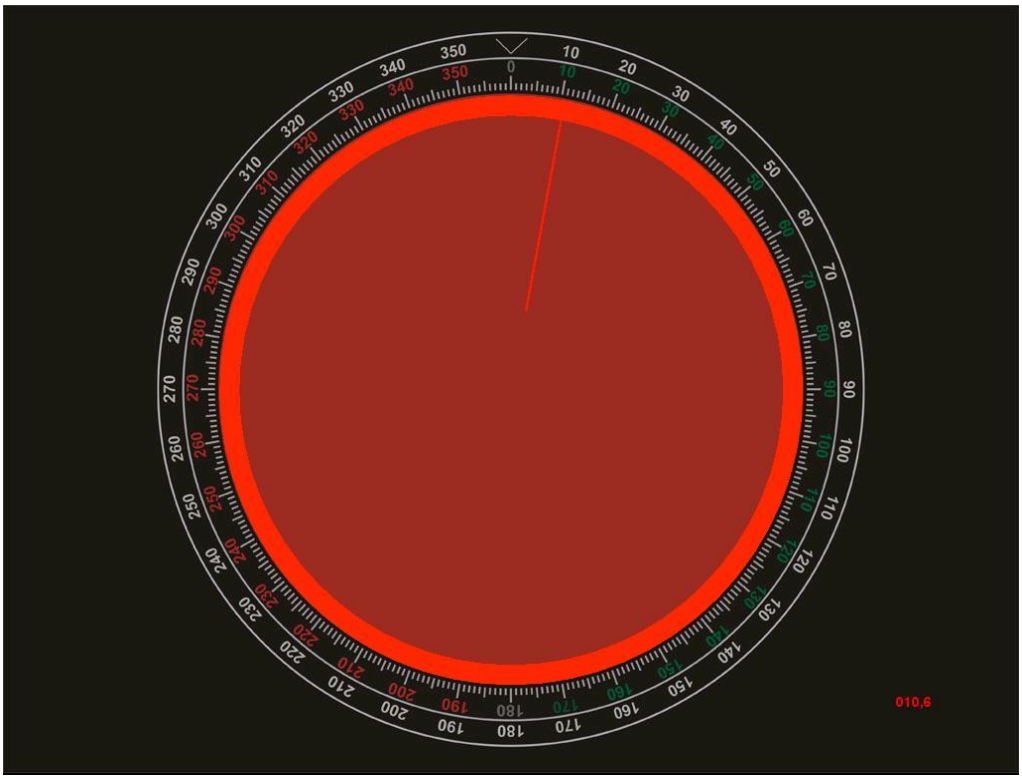


Figura 4.11: Sonar en estado de operación pasiva sin contacto.

4.3.5 Equipo en operación pasiva con contacto

Una vez que el software de control ingresa datos para el buque propio y para los contactos, inmediatamente pueden ser visualizados en la pantalla del sonar mediante el modo de operación pasiva. En la figura 4.12 se observa un contacto en la marcación 045 representado por una campana de GAUSS y el suplemento del rumbo propio mediante una línea segmentada de color naranja que parte de borde de la circunferencia hasta casi el centro de la misma. Una vez que se coloca el cursor sobre la campana, se puede escuchar el ruido generado por el batido de hélices, lo cual permite al sonarista calcular la velocidad aproximada a la que se encuentra el buque contacto. Para enviar la información del contacto, que permita plotearlo en el software de control de tiro, es necesario asignarle un canal de traqueo al contacto haciendo uso del botón adecuado en el panel

de control del sonar. Una vez hecho esto, se visualiza un pequeño cuadrado negro en cuyo interior se muestra el número del canal en color naranja. Solo entonces pueden ser enviadas las marcaciones en manual o en automático, según sea requerido.

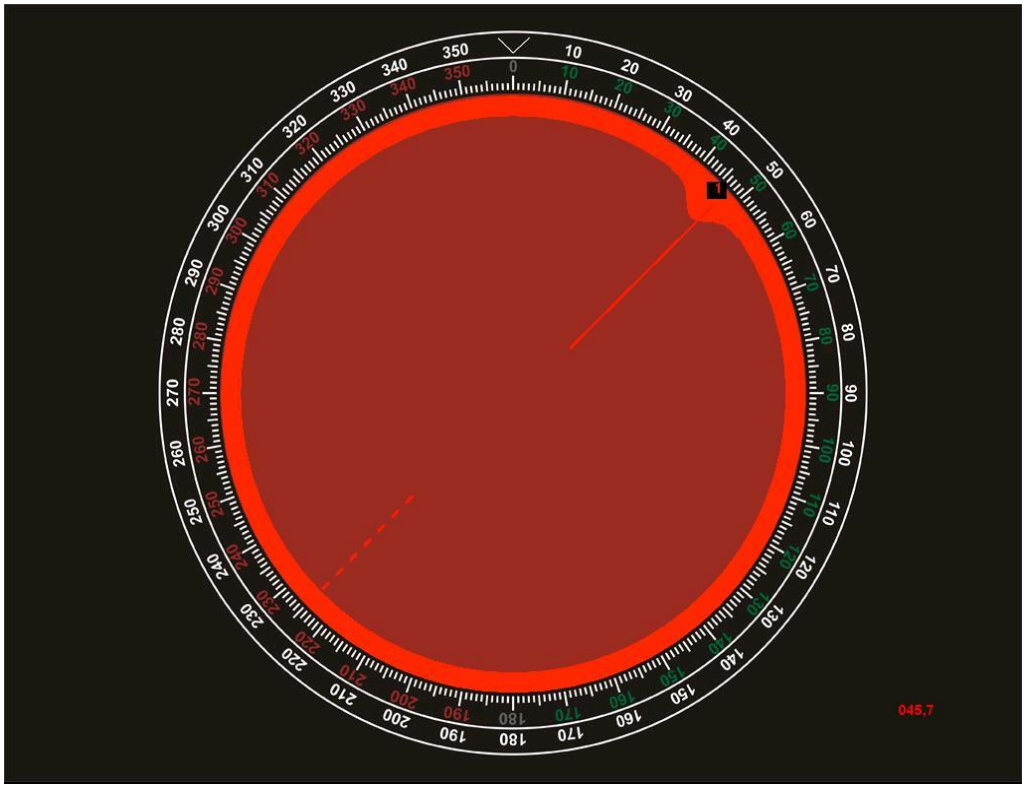


Figura 4.12: Sonar en estado de operación pasiva con un contacto en marcación 045.7.

4.3.6 Prueba Digital

La figura 4.13, muestra el sonar en estado de prueba digital (DIGITAL TEST) dibujando el ruido propio al borde de la circunferencia con una barra circular, el cursor, el suplemento del rumbo propio mediante una barra segmentada y dos campanas de GAUSS las cuales simulan dos contactos en el sonar. Ambas campanas generan un ruido particular permite al cursor engancharse a ellas y van avanzando en sentido horario, es por ello que el sonarista debe llevar el cursor del sonar hacia donde cada campana y presionar el botón de enganche, para luego asignarle una canal de transferencia. Si el cursor no engancha y/o no se puede asignar un canal de transferencia de datos, entonces el sonar no se encuentra 100% operativo.

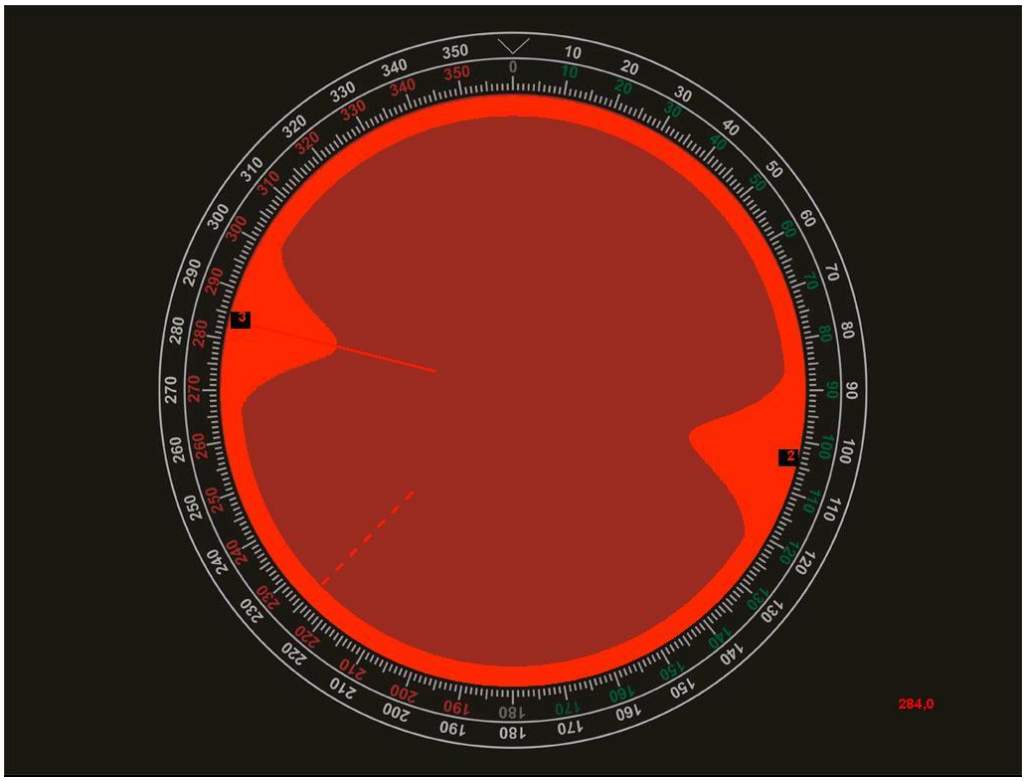


Figura 4.13: Sonar en estado de Digital Test.

4.3.7 Prueba Acústica

La figura 4.14, muestra el sonar en estado de prueba acústica (ACOUSTIC TEST); en donde aparece en un inicio la pantalla como en estado pasivo sin contacto alguna, pero con el transcurrir de unos segundos el anillo se cierra hasta casi llegar al centro de la pantalla, generando un ruido relativamente agudo y segmentado que incrementa su frecuencia a medida que el anillo se cierra. Si el anillo comienza a deformarse por algún ángulo, se puede llegar a la conclusión, que uno o más hidrófonos de arreglo se encuentran dañados.

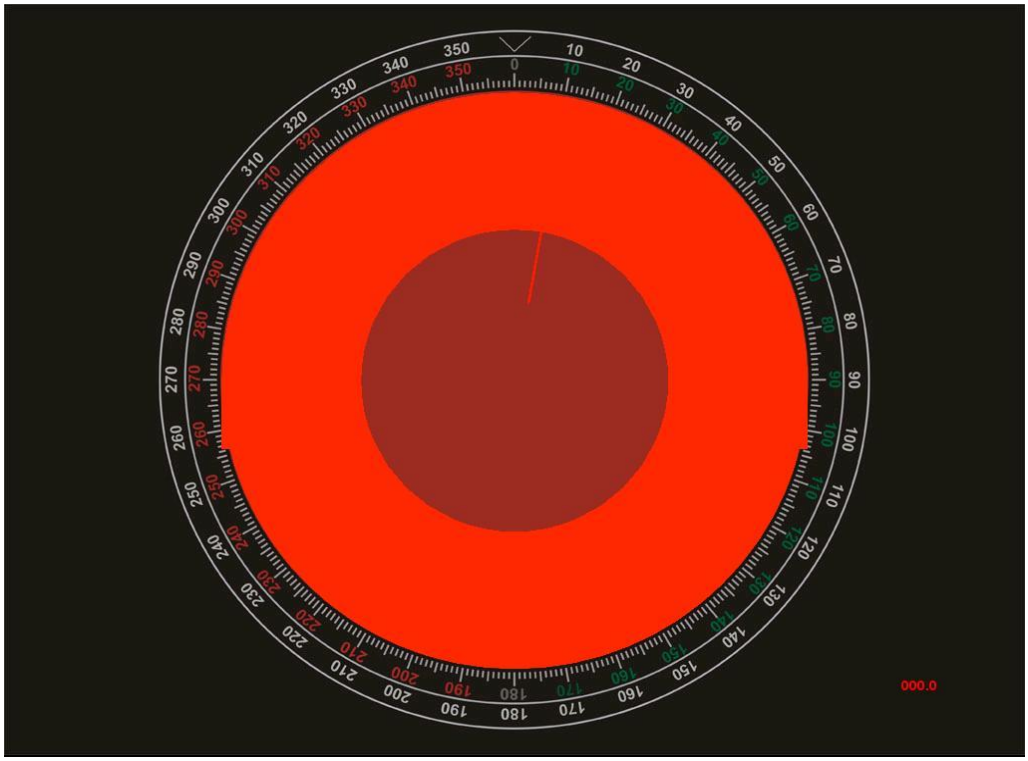


Figura 4.14: Sonar en estado de Acoustic Test.

4.3.8 Plataforma de Hardware

El software fue desarrollado y probado bajo la siguiente configuración de Hardware:

- Procesador Pentium IV de 1.2Ghz
- 512 MB de memoria RAM
- Tarjeta de Video ATI RADEON de 128MB de video
- Monitor de 19” en resolución de 1024 x 768

4.3.9 Configuración del Sistema

El software simulador de sonar fue desarrollado bajo las siguientes características de software:

- Lenguaje de programación JAVA bajo la plataforma estándar (J2SE – Java 2 standard edition).
- IDE de programación NetBeans 4.0
- Sistema Operativo Windows 2000 Profesional SP4

4.3.10 Métricas de Sonido

Con el fin de medir la similitud de los sonidos generado por el software simulador de sonar y los sonidos reales grabados abordo, fue necesario utilizar una herramienta que permita realizar un análisis espectral de los sonidos. La figura 4.15 muestra gráficamente un sonido real grabado con su respectiva amplitud y frecuencia.

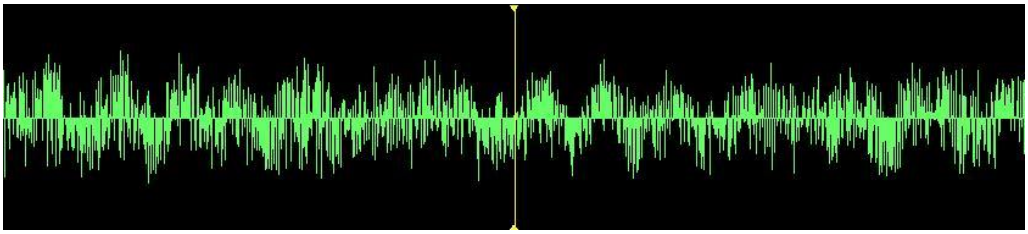


Figura 4.15: Sonido Real.

A continuación la figura 4.16 muestra gráficamente un sonido generado por el software simulador de sonar con su respectiva amplitud y frecuencia.

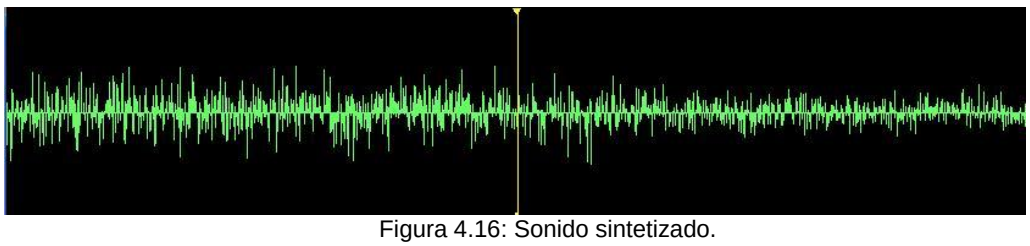


Figura 4.16: Sonido sintetizado.

Ambos sonidos corresponde a dos buques con distintas características y parámetros, sin embargo es posible observar cómo es que se llevo a cabo la comparación de sonidos con el fin obtener mayor semejanza durante la simulación ajustando las formulas que permiten incrementar la onda sinusoidal que varia la amplitud.

4.3.11 Situación Actual

En septiembre de 2006 llegue a tocar la escotilla de la Comandancia de la Fuerza de Submarinos para exponer mi intención de desarrollar un Sistema que permitiera solucionar alguna necesidad y que a la vez me sirviera para desarrollar mi tesis, tres meses después se me autorizó y sugirió el desarrollo de éste trabajo de tesis. A pesar de no tener experiencia en el desarrollo de aplicaciones como ésta, tomé el reto y comencé con la investigación de cada uno de los temas que se requería.

Después de haber pasado por cada una de las fases descritas en este capitulo, el 19 de agosto de 2007 se realizó la implantación de este trabajo en la Escuela de Submarinos sito en la Comandancia de la Fuerza de Submarinos en la Base Naval del Callao. A partir de ese momento comenzó la etapa de pruebas en acción.

> Para ello se programó una serie de ejercicios dentro del simulador de ataque, los cuales permitieron ubicar falencias existente en la primera versión del mismo. Hasta fines de noviembre de 2007 se continuó realizando pruebas y correcciones de las versiones, hasta llegar a la versión 1.7, siendo esta la ultima versión generada.

Glosario de terminos

1. **Byte**

Equivalente a octeto, es decir a ocho bits, para fines correctos, un byte debe ser considerado como una secuencia de bits contiguos, cuyo tamaño depende del código de información o código de caracteres en que sea definido. La unidad byte se representa con el símbolo B.

2. **C**

Es un lenguaje de programación creado en 1972 por Ken Thompson y Dennis M. Ritchie en los Laboratorios Bell como evolución del anterior lenguaje B, a su vez basado en BCPL.

3. **Campana de GAUSS**

Es una función que tiene la forma:

$f(x) = ae^{-(x-b)2/c2}$

donde a, b y c son constantes reales (a > 0).

4. **Clase ByteBuffer**

Clase Java que permite leer o escribir uno o más bytes de una manera más eficiente y organizada.

5. **Clase DatagramPacket**

Clase Java que permite recibir los mensajes la longitud, la dirección de Internet y el puerto en el que se envía por la clase DatagramSocket, estos datos son recepcionado en un arreglo de bytes.

6. **Clase DatagramSocket**

Clase Java que da soporte al uso de Socket para envío y recepción de datagramas UDP.

7. **Clase Thread**

Clase de Java que permite crear Hilos dentro un programa para efectuar distintas tareas en forma concurrente.

8. **Clase ParallelPort**

Clase Java que permite realizar tareas de lectura y escritura a través del Puerto paralelo de una computadora, haciendo uso de Interfaces Nativa de Java.

9. **Computadora de Ploteos**

Computador que forma parte del sistema de control de tiro, en donde se realizan cálculos matemáticos para determinar el rumbo de contacto ploteado.

10. **C++**

Es un lenguaje de programación, diseñado a mediados de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C. Se puede decir que C++ es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos.

11. **Datagramas**

Es un fragmento de paquete que es enviado con la suficiente información como para que la red pueda simplemente encaminar el fragmento hacia el equipo terminal de datos receptor, de manera independiente a los fragmentos restantes. Esto puede provocar una recomposición desordenada o incompleta del paquete en el destino.

12. Ethernet

Es el nombre de una tecnología de redes de computadoras de área local basada en tramas de datos. El nombre viene del concepto físico de ether. Ethernet define las características de cableado y señalización de nivel físico y los formatos de trama del nivel de enlace de datos del modelo OSI.

13. Float

El tipo de dato real define un conjunto de números que pueden ser representados con la notación de coma flotante.

14. Hidrófono

Un hidrófono es un transductor de sonido a electricidad para ser usado en agua o en otro líquido, de forma análoga al uso de un micrófono en el aire.

15. Hilo

Un Hilo es equivalente a un proceso del Sistema Operativo con la diferencia que los Hilos corren dentro de la Maquina Virtual de Java, la cual es un proceso del Sistema Operativo.

16. Internet

Es un método de interconexión descentralizada de redes de computadoras implementado en un conjunto de protocolos denominado TCP/IP y garantiza que redes físicas heterogéneas funcionen como una red lógica única, de alcance mundial.

17. Java

Una tecnología desarrollada por Sun Microsystems para aplicaciones software independiente de la plataforma orientada a objetos.

18. **Java Runtime Environment**

Corresponde a un conjunto de utilidades que permite la ejecución de programas java sobre todas las plataformas soportadas.

19. **Maquina virtual de Java**

Es un programa nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial, el cual es generado por el compilador del lenguaje Java.

20. **Marcación**

Se denomina marcación a un objeto al ángulo horizontal formado entre el norte y la visual desde el observador en el buque hasta el objeto. Se mide con origen en el norte de 0° a 360° en el sentido de las agujas del reloj.

21. **Periscopio**

Es un instrumento para la observación desde una posición oculta. En su forma sencilla es un tubo con un juego de espejos en los extremos, paralelos y en un ángulo de 45º respecto a la línea que los une.

22. **Proa**

Se llama proa a la parte delantera de un barco que va cortando las aguas del mar. También se denomina proa al tercio anterior del buque. Esta extremidad del buque es afinada para disminuir al máximo posible su resistencia al movimiento.

23. **Punto flotante**

Es un método de representación de números reales que se puede adaptar al orden de magnitud del valor a representar, usualmente trasladando la coma decimal —mediante un exponente— hacia la posición de la primera cifra significativa del valor.

24. Ruido sintetizado

O también conocido como sonido sintetizado, son los sonidos que se crean tras la ejecución de un algoritmo matemático.

25. Rumbo

Dirección en la que nos movemos o navegamos, o en la cual nos dirigimos o miramos y suele expresarse en forma del ángulo que forma esta dirección con otra tomada como referencia.

26. Simulador

Un simulador es un aparato que permite la simulación de un sistema, reproduciendo su comportamiento. Los simuladores reproducen sensaciones que en realidad no están sucediendo.

27. Simulador de Ataque

Simulador del puesto de comando de una unidad submarina.

28. Sistema de Control de Tiro

Conjunto de computadores a bordo de un submarino que generan la solución para efectuar un disparo.

29. Socket

Designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiarse cualquier flujo de datos, generalmente de manera fiable y ordenada.

30. Software:

Se refiere al equipamiento lógico o soporte lógico de un computador digital, comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de una tarea específica, en contraposición a los componentes físicos del sistema (hardware). Tales componentes lógicos incluyen, entre otras,

aplicaciones informáticas tales como procesador de textos, que permite al usuario realizar todas las tareas concernientes a edición de textos; software de sistema, tal como un sistema operativo, el que, básicamente, permite al resto de los programas funcionar adecuadamente, facilitando la interacción con los componentes físicos y el resto de las aplicaciones, también provee una interfase grafica ante el usuario.

31. Software de Control

Programa de computadora donde se generan e inician los datos necesarios para un ejercicio de ataque.

32. Sonar

Es el principal sensor de un submarino, esta compuesto de un arreglo de hidrófonos con el cual puede escuchar todos los sonidos del mar y graficarlos en una pantalla con el fin de discriminarlos y analizarlos.

33. Sonarista

Personal calificado en reconocimiento de los distintos tipos de sonidos generados en el mar, así como de operar el equipo de sonar a bordo de las unidades submarinas.

34. Smalltalk

Es un sistema informático que permite realizar tareas de computación mediante la interacción con un entorno de objetos virtuales. Metafóricamente, se puede considerar que un Smalltalk es un mundo virtual donde viven objetos que se comunican mediante el envío de mensajes.

35. Submarino

Es un tipo especial de buque capaz de navegar bajo el agua además de por la superficie, gracias a un sistema de flotabilidad variable.

36. World Wide Web (Red Global Mundial)

Es un sistema de documentos de hipertexto y/o hipermedios enlazados y accesibles a través de la Internet.

SIGLARIO

1. **3D:**
   Tridimensional

2. **J2SE:**
   Java 2 Standard Edition (Java 2 Edición Estándar)

3. **MGP:**
   Marina de Guerra del Perú

4. **VM:**
   Virtual Machine (Maquina Virtual)

5. **Hz:**
   Hertz, unidad de frecuencia del Sistema Internacional de Unidades.

6. **UDP:**
   User Datagram Protocol (Protocolo de data grama de usuario)

7. **IP:**
   Internet Protocol, número que identifica a cada dispositivo dentro de una red con protocolo IP.

8. **FTP:**
   File Transfer Protocol, protocolo para transferencia de archivos.

9. **TCP:**
   Transmission Control Protocol (Protocolo de Control de Transmisión).

10. **NMEA:**

    National Marine Electronics Association (Asociación Nacional de Electrónica de la Marina).

11. **ASCII:**

    American Standard Code for Information Interchange (Estadounidense Estándar para el Intercambio de Información).

12. **CR:**

    Carriage Return (retorno de carro).

13. **LF:**

    Line Feed (Salto de línea).

14. **GPS:**

    Global Positioning System (Sistema de Posicionamiento Global).

15. **IEEE:**

    The Institute of Electrical and Electronics Engineers (Instituto de Ingenieros Eléctricos y Electrónicos).

16. **IBM:**

    International Business Machines

17. **RISC:**

    Reduced Instruction Set Computer (Computadora con Conjunto de Instrucciones Reducido).

18. **COM1:**

    Puerto serie RS-232, utiliza cableado simple desde 3 hilos hasta 25 y que conecta ordenadores o microcontroladores a todo tipo de periféricos.

19. **JNI:**

    Java Native Interface (Interfase Nativa de Java), framework de programación que permite que un programa escrito en Java ejecutado en la máquina virtual java (JVM) pueda interactuar con programas escritos en otros lenguajes como C, C++ y ensamblador.

20. **LPT1:**

    Puerto Paralelo de impresión basado en el estándar IEEE 1284 (Standard Signaling Method for a Bi-directional Parallel Peripheral Interface for Personal Computers, en español, Estándar del Método de Señalización para una Interfase Paralela Bidireccional Periférica para Computadoras Personales).

21. **API:**

    Application Programming Interface (Interfaz de Programación de Aplicaciones).

22. **DLL:**

    Dynamic Linking Library (Bibliotecas de Enlace Dinámico).

23. **LED:**

    Light-Emitting Diode (Diodo emisor de luz).

24. **MB:**

    Megabyte, unidad de medida de cantidad de datos informáticos.

25. **SP4:**

    Service Pack 4 (Paquete de servicios 4).

26. **IDE:**

    Integrated Development Environment (Entorno de desarrollo integrado).

BIBLIOGRAFÍA

[1] David H. Friedel, Jr. and Anthony Potts
JAVA Programming Language HandBook.
The Corolis Group – USA. 1996.

[2] Elliotte Rusty Harold
Java Network Programming Second Edition.
O' Reilly – USA. 2000.

[3] Andrew Mulholland and Glen Murphy.
Java 1.4 Game Programming
Wordware Publishing - 2320 Los Rios Boulevar Plano Texas 75074 USA. 2003

[4] Klaus Betke
The NMEA 0183 Protocol
NMEA Office - New Bern NC 28564-3435 USA. 2000.
http://www.tronico.fi/OH6NT/docs/NMEA0183.pdf y http://www.nmea.org, Marzo, 2007.

[5] Wendy and Michael Boggs
UML with Rational Rose
Sybex – USA. 2002.

[6] David Goldberg
What Every Computer Scientist Should Know About Floating Point Arithmetic
Association for Computing Machinery INC – USA. 1991.

[7] Andrew Davison
Killer Game Programming in Java
O'Reilly – USA. 2005

[8] David Brackeen, Bret Backer y Laurence Vanhelswé
Developing Games in Java
New Riders Publishing – USA. 2003.

[9] Dale K. Pace
Modeling Simulation - Volumen 1
The Society form Modeling and Simulation International – USA. 2002.
http://www.modelingandsimulation.org/MandS0101/pdfs/MandSMag0101.pdf, Abril, 2007.

[10] Escuela de Submarinos
Manual de Electrónica de Equipos
Marina de Guerra del Perú, Callao. 1988.

[11] H.M Deitel y P.J Deitel
Cómo Programar en Java
Prentince Hall Hispano Americana S.A – México. 1998.