UNIVERSIDAD DE ZARAGOZA

ESCUELA UNIVERSITARIA DE INGENIERÍATÉCNICA INDUSTRIAL

Simulador Web de Circuitos con Amplificadores Operacionales

PROYECTO FIN DE CARRERA DE INGENIERÍA TÉCNICA INDUSTRIAL

Especialidad Electrónica

Septiembre de 2013

AUTOR: Oscar Fustero Zapata

DIRECTOR DEL PROYECTO:

Jesús Navarro Artigas

Área de Tecnología Electrónica

ÍNDICE DE CONTENIDOS

1 INTRODUCCIÓN

1.1 VISIÓN GENERAL

Hoy en día, en la sociedad española en la que vivimos, cada vez es mas frecuente, ver ir niños al colegio con un móvil inteligente en la mano. Juegan, se divierten, se comunican e interactúan con otros niños que al igual que ellos poseen otro móvil, con sistemas cada vez más inteligentes. Conformen crecen se convierten en adolescentes y aparecen en su vida otros dispositivos, PDA, Consolas de juegos, el ordenador de sobremesa en casa o la pizarra Electrónica en el colegio. Sus progenitores hacen uso de ellos, sus profesores hacen uso de ellos, la sociedad hace uso de ellos. Se acostumbran a vivir rodeados de todos estos dispositivos, e interactúan y los comprenden, muchas veces incluso mejor que sus mayores.

Posteriormente llegan a la Universidad, en este caso a la de Zaragoza y aparece el portátil. Con él pueden coger apuntes digitalmente en clase, tienen acceso al Anillo Docente Digital de la Universidad de Zaragoza, donde pueden ver apuntes de cualquier asignatura que esté disponible. Y tienen acceso, mediante la wifi, a toda la basta información que significa acceder a Internet.

Actualmente, cuando entras en una biblioteca de la universidad de Zaragoza, gran cantidad de estudiantes lo hacen delante de un portátil. Y el uso de papel y lápiz, que es fundamental para ciertos aprendizajes, disminuye cada año que pasa. Lo sustituyen programas como Word para textos, Excel para gráficas y cálculos, PowerPoint para presentaciones y por su puesto los Navegadores, las webs e Internet. Estas nuevas herramientas traen grandes ventajas en cuanto a la cantidad de información que es accesible, pero también algunos inconvenientes, pues la facilidad con la que se obtiene todo, les hace a veces, de acuerdo con mi experiencia personal, profundizar poco en esos conocimientos y despreciar en algunos casos aquellos magníficos libros y autores con los que yo aprendí durante la carrera.

Sin embargo, el barco ya zarpó y es imparable. Y yo he querido subirme a él, ponerme en el lugar del alumno de hoy en día, y hacer mi aportación aceptando la propuesta del director de este proyecto fin de carrera. Una aportación totalmente orientada a la docencia, teniendo en cuenta los medios actuales en uso, con un enfoque de aprendizaje rápido y directo, libre de profundidades conceptuales.

1.2 BASES, DESARROLLO Y OBJETIVOS DEL PROYECTO.

Se trataba de desarrollar un simulador de circuitos electrónicos con amplificadores operacionales, que sirviera como complemento docente, en la asignatura de Electrónica Analógica, impartida en el grado de Tecnologías y Servicios de Telecomunicación de la Universidad de Zaragoza por el Director del Proyecto, Don Jesús Navarro Artigas.

Con este proyecto se pretende dotar, por parte del profesor al alumno, de una herramienta de aprendizaje dinámica y visual con la que se ilustran los comportamientos de diferentes etapas y circuitos vistos en dicha asignatura.

De esta manera, ciertos conceptos de la asignatura que podrían resultar un tanto abstractos, pueden entenderse de una manera más sencilla mediante las aplicaciones gráficas que se han desarrollado.

La idea principal de este proyecto es que el alumno, una vez presentados los desarrollos conceptuales en clase, pueda asimilarlos de una manera más rápida mediante las diferentes simulaciones que proporcionan los applets desarrollados.

El simulador debía estar adaptado a la asignatura, pero también al nuevo perfil del alumno. Sencillo y de aprendizaje rápido. Además debía ser posible utilizarlo desde cualquier de los dispositivos electrónicos que he mencionado. Había que buscar el elemento más común en todos ellos. Y lo hay: todos pueden acceder a internet, todos tienen un navegador y casi todos pueden correr aplicaciones basadas en java desde el navegador.

Por tanto el simulador debía ser un simulador web basado en Java.

Sin embargo, crear un simulador desde cero era tarea ingente para un proyecto fin de carrera llevado a cabo por un solo alumno. Por lo que se partió de dos softwares ya desarrollados, y a los cuales debo mi gratitud:

- El primero es un proyecto ya desarrollado para esta asignatura, por Jorge Mañas Gregorio, exalumno de la universidad de Zaragoza, en el que se emulaban de forma estática las diferentes etapas con amplificadores analógicos, y que debido a su antigüedad, o mejor dicho a la velocidad con que se dan los cambios en informática, había dejado de funcionar, principalmente por la gran cantidad de código obsoleto. Este proyecto estuvo dirigido por Don Jesús Navarro Artigas y la propiedad intelectual es compartida por ambos. Habiendo sido entregado como proyecto final de carrera en la Universidad de Zaragoza.

- El segundo es un proyecto desarrollado inicialmente por Paul Falstad, matemático de carrera y programador por afición. Y consiste en un simulador de circuitos dinámico, con una matemática bastante compleja, basado en java, con una capacidad limitada de simulación, modelos de componentes electrónicos simples, y un código bastante farragoso y caótico, debido a las numerosas incursiones y aportaciones que otros colaboradores y estudiantes han hecho. El proyecto esta publicado en su página web, y los requisitos para su uso comprenden la posible modificación y redistribución del código siempre y cuando no sea para propósito comercial, como lo es en este caso.

Partiendo pues de estos dos softwares y estando los dos realizados bajo Java y con el uso de Applets. El lenguaje y el medio sobre el cual trabajar quedó definido.

Así pues, el simulador web se ha construido agrupando cuatro premisas: estas dos partes de programación, sincronizadas con los apuntes de la mencionada asignatura de Electrónica Analógica, y por último enfocando el simulador al alumno actual. Trabajo sin ninguna duda amplio y muy dificultoso, debido principalmente a la amplitud del lenguaje java y la extrema diferencia de forma de programar de uno y otro autor. La mayor dificultad y más de la mitad del tiempo y esfuerzo empleado ha venido de entender y comprender las aproximadamente 20.000 líneas de código programadas por Jorge Mañas y las 12.000 de Paul Falstad, que por otro lado son de un nivel matemático muy grande.

En la parte de Mañas, se ha realizado un gran trabajo actualizando gran número de líneas y código que ya no funcionaban, por estar obsoletas. Se han quitado las imágenes estáticas y se han sustituido por la simulación dinámica del circuito. Para ello se han añadido unas 3000 líneas más necesarias para unificar el simulador.

En la parte de Falstad, se han recortado unas 5000 líneas de código no necesarias para este simulador y se han cambiado la mayoría de las restantes, excepto las que se refieren al núcleo matemático de computación del simulador. Se ha traducido al español y comentado la mayoría del código restante, para que pueda ser rehusado fácilmente. Y se han cambiado varias librerías correspondientes a modelos de componentes electrónicos: Diodo, fuente de voltaje, resistencia, terminal de voltaje, amplificador operacional, transistor, etc., además de algunas librerías para la carga y descarga de los circuitos. Y en especial se ha regenerado y cambiado la representación gráfica de todo el simulador, para cumplir con esa cuarta premisa de ser sencilla al alumno de hoy.

Otro objetivo de este proyecto era que se pudiera ejecutar en servidor, para evitar la dependencia de la capacidad de computación del ordenador cliente. Esto se ha conseguido

por el hecho de estar programado en java pero con el código en forma de applets. En capítulos siguientes me extenderé más sobre las cualidades de un applet.

Y, más que un objetivo, un deseo mutuo del autor y el director del proyecto, era que fuera gratuito y de fácil alcance para el alumno de la Universidad de Zaragoza. En especial, quería ayudar al alumno con insuficientes recursos como para poder comprar un simulador comercial. El simulador aquí desarrollado es gratuito y se colgará en el anillo docente digital de forma que los alumnos puedan a acceder a el fácilmente.

Además podrá ser cambiado y mejorado por otros alumnos que lo deseen, con sólo solicitarlo al autor o el director de este proyecto, ya que **los derechos de autor sobre los resultados del mismo pertenecen, indistintamente, tanto al autor como al director del mismo.**

Por otro lado el simulador no permite una emulación compleja como la de otros simuladores comerciales como Pspice, Ni Multasen o EletroWin. Sin embargo no era un objetivo de este proyecto hacer una simulación exhaustiva.

1.3 JAVA Y COMPATIBILIDAD

Java se originó cuando algunas personas de Sun Microsystems trataban de desarrollar un lenguaje nuevo adecuado para programar aparatos caseros orientados a información. Después se reorientó el lenguaje hacia la *World Wide Web*. Aunque Java toma prestadas muchas ideas y algo de la sintaxis de C y C++, es un lenguaje nuevo orientado a objetos, incompatible con ambos.

La idea principal de usar Java para páginas *Web* interactivas es que una página *Web* puede apuntar a un programa Java pequeño, llamado applet (que podría traducirse como "aplicacioncita"). Cuando el visualizador llega a ella, el *applet* se “baja” a la máquina cliente y se ejecuta ahí de una manera segura. Debe ser estructuralmente imposible que el *applet* lea o escriba archivos que no está autorizado a acceder. Debe también ser imposible que el *applet* introduzca virus o cause algún otro daño. Por estas razones, y para lograr transportabilidad entre máquinas, los *applets* se compilan para crear un código de bytes después de escribirse y depurarse. Son estos programas en código de bytes a los que apuntan las páginas *Web*.

Antes de entrar en detalles del lenguaje Java, vale la pena decir algunas palabras sobre la utilidad del sistema Java y las razones por las que la gente quiere incluir *applets* Java en sus páginas Web.

Por una parte, los *applets* permiten que las páginas *Web* se vuelvan interactivas. Otra razón para ejecutarlos en la máquina cliente es que hace posible la adición de animación y sonido a las páginas Web sin tener que llamar a visualizadores externos. Como el *applet* se ejecuta localmente, aun si se está interpretando, puede escribir en cualquier parte de la pantalla de la manera que quiera y a muy alta velocidad.

El sistema Java tiene tres partes:

1. Un compilador de Java a código de bytes.
2. Un visualizador que entiende *applets*.
3. Un intérprete de código de bytes.

Un programa en Java consiste en uno o más paquetes, cada uno de los cuales contiene algunas definiciones de clases. Los paquetes pueden accederse remotamente a través de una red, por lo que aquellos destinados para ser usados por mucha gente deben tener nombres únicos.

Una definición de clase es una plantilla para producir ejemplares de objetos, cada uno de los cuales contiene las mismas variables de estado y los mismos métodos que todos los otros ejemplares de objetos de su clase. Sin embargo, los valores de las variables de estado de los diferentes objetos son independientes. Las clases, por tanto, son como moldes para hacer piezas: no son las piezas realmente, pero sirven para producir piezas estructuralmente idénticas, produciendo cada molde de piezas una forma diferente de pieza. Una vez producidas, las piezas (objetos) son independientes entre sí.

Además del lenguaje básico, los diseñadores de Java definieron e implementaron unas 200 clases con la versión inicial, que han ido ampliando con el paso del tiempo. Los métodos contenidos en estas clases forman una especie de entorno estándar para los creadores de programas Java. Las clases están escritas en Java, por lo que son transportables a todas las plataformas y sistemas operativos. Estas clases se agrupan en paquetes de tamaño desigual, cada uno de los cuales se enfoca hacia algún tema central. Los *applets* que necesiten un paquete en particular pueden incluirlo usando el enunciado *import* de Java. Los métodos contenidos pueden usarse según se requiera. Este mecanismo reemplaza la necesidad de incluir archivos de cabecera propia del lenguaje C. También reemplaza la necesidad de bibliotecas, puesto que los paquetes se cargan dinámicamente durante la ejecución al ser invocados.

Se enuncian brevemente estos paquetes:

- *java.lang* -> para funciones del lenguaje.
- *java.util* -> para utilidades adicionales.
- *java.io* -> para manejo de ficheros.
- *java.text* -> para formato de texto especializado.
- *java.awt* -> para diseño gráfico e interfaz de usuario.
- *java.awt.event* -> para gestionar eventos
- *javax.swing* -> nuevo diseño de GUI (*Graphical User Interface*).
- *java.net* -> para comunicaciones.
- *java.applet* -> contiene parte de la maquinaria básica de los *applets*.

En el presente proyecto, principalmente se hace uso de los paquetes *java.applet, java.awt, java.awt.event* y *javax.swing*.

A pesar de usar librerías standard de java, y de la política que tanto la compañía Sun Microsystems, fundadora del lenguaje Java, como de Oracle nueva propietaria, consistente en maximizar la compatibilidad entre todas las versiones de java, he tenido que tener especial cuidado de no usar propiedades métodos y objetos de dichas librerías que aun están disponibles pero que están obsoletos o van a ser obsoletos, tal como declara el IDE NetBeans en su ayuda contextual al programar con Java y la página web de Oracle sobre Java.

De hecho, como ya he comentado, el proyecto realizado por Jorge Mañas, utilizado en parte para realizar la parte estática de este simulador, había dejado de funcionar, y la razón principal es que algunos de los métodos y eventos de los objetos utilizados estaban obsoletos. La parte de código usado proveniente del simulador de Paul Falstad, funcionaba correctamente, sin embargo también existían deficiencias y métodos obsolescentes que se han corregido.

Así pues usando los foros de programación, la ayuda de Oracle sobre el lenguaje Java y la ayuda contextual del IDE NetBeans se han eliminado y sustituido estos métodos, eventos y propiedades. Con el fin de mantener la compatibilidad y la vida útil de este proyecto durante el máximo tiempo posible.

Algunos de los métodos y eventos que con más frecuencia han aparecido y han sido actualizados son:

java.awt.Dialog.show() -> setVisible(boolean)

java.awt.Component.resize(int, int) -> *setSize(int, int)*

java.awt.Dialog.hide() -> setVisible(boolean)

javax.swing.AbstractButton.setLabel(String) -> *setText(text)*

cast the intenger y doubles redundantes -> se eliminaron

java.awt.Component.move(int, int) -> *setLocation(int, int).*

java.awt.Component.handleEvent(Event) -> *processEvent(AWTEvent).*

java.awt.Component.show() -> setVisible(boolean)

java.awt.Scrollbar.setLineIncrement(int) -> *setUnitIncrement(int).*

java.awt.CheckboxGroup.setCurrent(Checkbox)->*setSelectedCheckbox(Checkbox)*

java.awt.Component.reshape(int, int, int, int) -> *setBounds(int, int, int, int).*

java.awt.TextArea.replaceText(String, int, int) -> *replaceRange(String, int, int).*

java.awt.Container.preferredSize() -> *getPreferredSize().*

java.awt.Container.insets() -> *getInsets().*

java.awt.Component.bounds() -> *getBounds().*

java.awt.Container.countComponents() -> *getComponentCount().*

java.awt.Menu.countItems() -> *getItemCount().*

java.awt.MenuItem.disable() -> *setEnabled(boolean).*

java.awt.Component.disable() -> *setEnabled(boolean).*

java.awt.MenuItem.enable() -> *setEnabled(boolean).*

java.awt.Component.enable() -> *setEnabled(boolean).*

1.4 ENTORNO ELEGIDO

Dentro de un tema concreto, un *applet* puede estar limitado a un solo concepto, o puede tratar varios conceptos relacionados. La selección de un tema apropiado para un *applet* y una buena práctica son extremadamente importantes.

Al ser una herramienta visual para el aprendizaje, el aspecto del interfaz de usuario de la aplicación es importante. Un mal diseño del GUI (*Graphical User Interface*) puede llevarnos a un *applet* que no sea muy útil a pesar del gran esfuerzo realizado en su desarrollo. El GUI del *applet* se puede desarrollar por separado del resto del programa, haciendo uso de las ventajas de la programación orientada a objetos de Java. Para el GUI también podemos utilizar una herramienta de desarrollo visual.

Para este proyecto se pensó en dos alternativas como posibles entornos de desarrollo de los applets. Estas dos posibilidades fueron “Eclipse Standard 4.3 + el I.D.E for Java Developers” y “NetBeans 7.2.1 I.D.E”. Los dos I.D.Es son de distribución libre, y permiten programar en multitud de lenguajes, PHP, Java, C/C++. Y las diferencias técnicas entre uno y otro no eran significativas. Sin embargo buscando ventajas y desventajas en los foros de internet, encontré que gran cantidad de foreros argumentaban que así como Eclipse era el mejor I.D.E. para PHP, pues fue el primer lenguaje con el que empezaron a desarrollar este I.D.E., la gran cantidad de plugins que diferentes programadores han añadido para NetBeans y que funcionan correctamente y la periodicidad de sus actualizaciones en paralelo con las diferentes versiones del Java Runtime, lo hacían mas útil para programar Java. Así pues y siendo poca la diferencia entre los dos, se eligió el I.D.E. NetBeans.

Esta utilidad permite desarrollar rápidamente un prototipo de interfaz gráfico. Los otros contenidos del *applet* (es decir, los contenidos técnicos y los objetos) se desarrollan aparte del GUI. Un diseño disciplinado y sustentado en la programación orientada a objetos va a ser beneficioso para futuros mantenimientos del programa y para incrementar la productividad en la programación, pues el desarrollo de clases de Java reutilizables puede llevarnos a recompilar con facilidad una librería de clases de Java en el área de etapas de Electrónica Analógica.

2 APPLETS, JAVA Y REQUISITOS

2.1 ¿POR QUÉ USAR APPLETS?

La gran ventaja de los *applets* es su universalidad. Un applet insertado en una página web, en principio, puede utilizarse en cualquier otra computadora con tal que tenga un navegador de Internet (Chrome 6.0, Internet Explorer 4.0, Firefox Mozilla 5.0) o versiones superiores) que entienda Java.

Una persona puede desarrollar un *applet* usando un PC con Windows, insertar su applet en una página web, ponerla en un servidor de Internet e instantáneamente su applet puede ser cargado y ejecutado en cualquier parte del mundo desde cualquier computadora conectada a Internet, independientemente del sistema operativo del que disponga el usuario. Esta independencia de la plataforma significa que códigos idénticos, tanto el código fuente como los compilados, se pueden ejecutar en sistemas Windows, Unix, etc. Estos *applets* de Java, al descargarse de Internet, podrán verse en cualquier sistema si su

explorador tiene habilitado el Java.

Otra gran ventaja es la seguridad que ofrecen al usuario. Un *applet*, por diseño, no puede acceder a los recursos del sistema del cliente en donde se está ejecutando, por lo cual no puede leer la información del cliente, ni modificarla, ni destruirla.

La disponibilidad para los estudiantes de los materiales del curso desde su casa o desde un terminal de trabajo en la Universidad sin necesidad de instalar ningún software o la posibilidad de realizar una clase práctica si se dispone de un aula con conexión a Internet es otro atractivo que ofrece el desarrollo de *applets* con fines docentes.

Java es un lenguaje de programación para Internet, y los *applets* están integrados naturalmente en el entorno de documentos con HTML (*HyperText Markup Language*).

Debido a la creciente expansión de la *World Wide Web* (WWW) y al uso de los exploradores que utilizan los documentos HTML por defecto, cada vez es mayor el desarrollo y la distribución de utilidades interactivas docentes por Internet.

A parte de las ventajas expuestas anteriormente en cuanto a la utilización de Java como entorno de desarrollo, comentar que los applets pueden ser fácilmente incluidos en una pagina *HTML* conjuntamente con otros elementos multimedia (imágenes, videos y sonidos), permitiendo una sencilla configuración del material multimedia dinámico destinado al aprendizaje. Esto se debe al hecho de que los *applets* se ejecutan embebidos en una página *Web.*

El objetivo de estos *applets* es producir una visualización dinámica de ciertos conceptos, e ilustrar visualmente el funcionamiento de las correspondientes configuraciones, facilitando la familiarización con dichos conceptos, lo que, por otra parte suele resultar más arduo contando únicamente con el material tradicional de aprendizaje. Es lo que podríamos denominar una facilitación visual de la teoría.

Un *applet* se sustenta en la interfaz gráfica de usuario para la entrada de datos y en la representación visual de los datos de la simulación. Por eso es una herramienta excelente para la enseñanza y el aprendizaje de conceptos abstractos y principios científicos o/y tecnológicos. Los procesos de simulación y los resultados pueden ser presentados visualmente en tiempo real. Una variación dinámica del *applet* mantendrá un alto nivel de atención del alumno.

Por otra parte, los sistemas de software destinados a la enseñanza se pueden clasificar de acuerdo con el modo en que los estudiantes acceden (locales y distribuidos), si es expansible por la comunidad docente a los usuarios o no (cerrado y abierto), y el entorno de ejecución del software (ejecutable y embebido).

Una aplicación web de estas características (applets) debería permitir un acceso

distribuido y una fácil expansión del sistema por parte del profesorado. Esto se puede llevar a cabo efectivamente si el software se ejecuta embebido en una página *HTML*.

Por el contrario, una aplicación web desarrollada en una plataforma específica con un lenguaje como C sería necesariamente cerrada.

Lo mismo podemos decir de una herramienta como Pspice. Los estudiantes tendrán acceso a la misma bien mediante un terminal en el laboratorio o con un CD-ROM distribuido individualmente (por ejemplo con un libro de texto). Este sistema es rígido. Una ampliación o reorganización no es fácil.

La herramienta informática a desarrollar en este proyecto (applets específicamente orientados a la Electrónica Analógica) es un ejemplo de sistema abierto, puesto que es una colección de material con tecnología *Web* ampliable y modificable en cualquier momento, abierto a todo tipo de cambios.

Por otro lado, los *applets* de Java permiten al alumno interactuar más estrechamente con el material de aprendizaje, dándole un mayor control sobre éste. Los resultados de la simulación se obtienen en tiempo real. En este sentido, los *applets* de Java añaden una nueva dimensión a los cursos multimedia basados en tecnologías *Web*. Tienen excepcionales características como herramientas explicativas y demostrativas.
De modo que, como elemento aislado, un *applet* permite mucha más interactividad y potencia de cálculo que otros elementos existentes. Los *applets* de Java por consiguiente son muy adecuados para integrar muchos principios y conceptos relacionados en un mismo elemento.

2.2 ¿QUÉ ES UN APPLET?

Tradicionalmente, el término *applet* se ha utilizado con los pequeños programas escritos en Java que se ejecutan desde un documento *Web*, es decir, desde un archivo con formato HTML. Las aplicaciones en Java son programas ejecutables desde la línea de comandos, independientemente del explorador *Web*. El tamaño o la complejidad de los *applets* en Java no tienen límites. De hecho, en cierto modo son más potentes que las aplicaciones escritas con ese lenguaje de programación. De todas formas, dentro de Internet, donde la velocidad de las comunicaciones está limitada y los tiempos de carga son realmente grandes, el tamaño de los *applets* ha de ser pequeño.

Las diferencias técnicas entre los *applets* y las aplicaciones estriban en el contexto donde se ejecutan. Una aplicación Java se ejecuta en un entorno extremadamente sencillo. Por otro lado, los *applets* de Java reciben la información directamente del explorador *Web*.

Ha de saber cuándo se inicia, cuándo se debe dibujar en la pantalla y cuándo se activa o desactiva. Como consecuencia de estos dos entornos tan dispares, los requisitos mínimos de los *applets* y las aplicaciones son distintos.

2.3 ESTRUCTURA DE LOS APPLETS

Para escribir *applets* de Java hay que utilizar una serie de métodos. Incluso para el *applet* más sencillo. Los métodos son códigos que "entienden" y manipulan el estado de un objeto. Pueden ser llamados dentro de una misma clase o por otras clases. Además estos métodos siguen un "ciclo de vida" del applet.

Cuando un applet se inicia, se llaman en este orden a los siguientes métodos:

- **init():** suele contener instrucciones para inicializar el applet, entre otras se declaran las variables y objetos que contendrá el applet, y si es necesario se inicializan.

- start(): no es un método obligatorio de programar, pero si conveniente. Al igual que init(), se suele usar para inicializar variables y objetos. Sin embargo init() sólo se ejecuta una vez en el ciclo de vida del applet y start() será llamado además cada vez que ser reinicie el applet.

- paint(): se encarga de mostrar el contenido del applet. Es el método que suele contener la mayor parte del código y se ejecuta cada vez que se tenga que redibujar el applet.

- stop(): suspende la ejecución del applet. Se llama cuando el applet se vuelve temporalmente invisible. Por ejemplo al cambiar de pestaña en un navegador. No es obligatorio y está en desuso ya que muchos de los navegadores bloquean esta función.

- destroy(): se ejecuta cuando no se va a necesitar más el applet, por ejemplo al cerrar una ventana que contiene el applet. Se debe de usar para liberar recursos de la memoria y cerrar cualquier proceso.

2.4 APPLETS Y LIBRERIAS JAVA

Como ya hemos apuntado, dentro de init() se suelen definir las variables y objetos a utilizar y dentro de paint(), se define de qué forma se dibujarán en nuestro entorno y cómo se comportarán. Sin embargo en java tenemos una cantidad de objetos bastante grande para elegir introducir en estos métodos y varias elecciones posibles.

En el simulador hay dos partes claramente diferenciadas, la parte estática y la parte dinámica. Cada una de las dos partes, se ha programado en un applet diferente. Siendo la parte estática la que, cuando obtiene los datos necesarios para su funcionamiento y para el funcionamiento del applet dinámico, llama, carga el applet dinámico y lo pone en marcha. Por tanto tenemos dos applets distintos con propiedades y características distintas. A saber:

-El applet dinámico, necesita velocidad de cálculo (especialmente cuando se le piden frecuencias altas), requiere un esfuerzo alto del computador en cálculo ya que mientras permanece encendido se están actualizando constantemente tanto el dibujo como todos los valores de tensiones y corrientes de cada uno de los nodos y componentes del circuito. Así mismo su diseño es sencillo, un poco plano, sin grandes complicaciones. Por ejemplo los botones no tienen bordes, las barras de desplazamiento no marcan su posición al pasar el ratón por encima, etc.

-El applet estático es más lento, no necesita más que un pequeño pico de gasto computacional al apretar el botón de simulación y después no consume. Por otro lado es el applet base del simulador, y por esta razón se ha hecho más atractivo.

Así pues, hay dos razones básicas por las que en cada applet, de forma mayoritaria, se han usado dos grandes bibliotecas, sus objetos, clases y métodos:

-Para el applet dinámico se ha elegido la biblioteca AWT, por que accede directamente al API de Windows, y por tanto carga con menos Kb el código y actúa a un nivel mas bajo. Aplicaciones en AWT pueden correr a velocidades parecidas a las que puede funcionar una aplicación en C/C++. Al elegir esta opción, sin embargo, perdemos un poco de presentación, algo de aspecto. Pero considero justo pagar algo de aspecto por conseguir una mayor eficiencia y respuesta.

-Para el applet estático se ha elegido la biblioteca SWING, que tras pasar por varios filtros termina accediendo a la AWT. Y por lo tanto es más lenta. Sin embargo sus objetos son más abstractos y añaden mayor peso en Kb al código. Por el contrario la apariencia de botones y controles es mucho mas atractiva y su usabilidad mayor y por estas razones se ha elegido esta biblioteca para diseñar esta applet.

2.5 PROCESO DE EJECUCIÓN DE UN APPLET POR UN EXPLORADOR WEB

Un explorador Web compatible con Java, cuando se encuentra con una etiqueta <APPLET> dentro de un documento HTML, dará una serie de pasos:

1. El explorador reservará espacio en el documento para mostrar el *applet*. Se utilizarán los parámetros WIDTH y HEIGHT para determinar las dimensiones de dicho espacio.
2. El explorador leerá los parámetros que se encuentren dentro de las etiquetas <PARAM><\PARAM>.
3. Se inicia la Máquina Virtual de Java y se pide que cargue e inicie el parámetro. El *applet* accede a los nombres y valores que se encuentran dentro de las etiquetas <PARAM><\PARAM>.
4. La Máquina Virtual crea una copia del *applet* y la ejecuta. Para ello se basa en el contenido del archivo *class* cuyo nombre hay que especificar en el código HTML.
5. El explorador llama al método *init()* del *applet* para que éste se inicie por sí solo.
6. La Máquina Virtual llama al método *start()* del *applet* en el momento en que está listo para iniciar el proceso. También llama al método *paint()* para dibujar el *applet* dentro de la ventana del explorador.
7. Siempre que se tenga que volver a dibujar el *applet*, el explorador volverá a llamar al método *paint()*.
8. El explorador llamará al método *stop()* cuando el usuario acceda a otro documento o ventana.
9. El explorador llamará al método *destroy()* cuando quiera eliminar al *applet* de la memoria.

2.6 REQUISITOS MÍNIMOS DEL SISTEMA

Los requisitos técnicos mínimos para visualizar correctamente los applets en un equipo son los siguientes:

1-Disponer de un **navegador web**. El simulador ha sido probado con las versiones mínimas de los siguientes navegadores:

-Internet Explorer 4.0
-Mozilla Firefox 5.0
-Google Chrome 6.0

Y también ha sido verificado y funciona con las últimas versiones de:

- Google Chrome versión 29.0.1547.62. Puede descargarlo aquí:
https://www.google.com/intl/es/chrome/browser/
Puede descargar otras versiones de Google Chrome en:
http://es.oldversion.com/windows/google-chrome/

- Internet Explorer versión 10.0.9200.16660. Puede descargarlo aquí:
http://windows.microsoft.com/es-ES/internet-explorer/download-ie
Puede descargar otras versiones de Internet Explorer en:
http://es.oldversion.com/windows/internet-explorer/

- Mozilla Firefox versión 22.0. Puede descargarlo aquí:
http://www.mozilla.org/es-ES/firefox/new/#download-fx
Puede descargar otras versiones de Mozilla Firefox:
https://ftp.mozilla.org/pub/mozilla.org/firefox/releases/

2- Un **monitor** que permita una resolución mínima de 600x800, aunque la aplicación está optimizada para la resolución 1024x768. Y puede funcionar a mayores resoluciones.

3- El **procesador** mínimo con el que ha sido probado y funciona correctamente fue un procesador Intel Pentium 500Mhz con 1GBytes de RAM. También ha sido probado con portátiles de gama media. Y con un AMD A10-5800 APU con 8GBytes de RAM. Por supuesto, la parte dinámica depende del procesador para poder dar una respuesta correcta,

pero principalmente esta dependencia se relaciona con la frecuencia de las ondas simuladas. Por lo que en caso de tener un procesador pequeño se recomienda simular a bajas frecuencias, y jugar con la barra de deslizamiento Simulación. Que permite ajustar la velocidad de cálculo y representación de la parte dinámica de la simulación.

4- Se ha trabajado principalmente con el **sistema operativo** Windows 7, pero se ha probado en Windows xp y en Windows Vista, y funciona. Durante el desarrollo y prueba del simulador no se han experimentado conflictos que hagan colgarse al sistema operativo, pero sí a algunos navegadores en ciertas condiciones extremas:

-Si abre la aplicación en ordenadores con poca capacidad de computación, o ejecuta el simulador introduciendo valores de frecuencia muy altos, puede encontrar desde que el redibujo del applet dinámico parpadea por que no se puede dibujar a tanta velocidad, pasando por que su navegador se ralentiza y llegando hasta tal punto que deberá reiniciar el navegador.

-Si abre varias ventanas del mismo simulador, simulando a la vez varios circuitos, dependiendo de la capacidad de su procesador y del navegador, o se ralentizará, o llegará un momento en que el circuito simulado no se cargará. Reinicie su navegador y cargue el circuito que desee.

5- Por ultimo, pero lo más importante, es disponer de un ordenador con una máquina virtual de Java instalada. Se ha verificado el simulador con la versión de Java 6.0, y se ha optimizado el simulador con la última versión disponible Java 7.0.25.

2.7 CONFIGURACIÓN DE JAVA Y NAVEGADORES EN WINDOWS.

No pasa un segundo en el mundo sin que internet crezca y se expanda. Conforme se produce esta expansión, siempre hay personas o agrupaciones de hackers que desean aprovecharse. Crean amenazas como virus y phishing. Por esto es conveniente tener tanto la versión de java como los navegadores actualizados. Sin embargo a veces a las compañías les resulta difícil mantener la compatibilidad entre proteger su ordenador de estas amenazas y permitirles total flexibilidad para la ejecución de cualquier programa en su propio ordenador.

Por esto, al abrir la página web con alguna de las etapas de este simulador, esté atento a las diferentes señales que su navegador le dará:

-Si no aparece el simulador, y ni siquiera un cuadro delimitador, lo más probable es que tenga usted un conflicto con anteriores versiones de java. Le recomiendo desinstalar todas las versiones de java. Puede hacerlo desde el panel de control de su ordenador, vaya al administrador de aplicaciones o programas y desinstale java. Reinicie su ordenador e instale únicamente la última versión.

-Si en su navegador aparece una barra amarilla en la parte superior o un aviso sobre vulnerabilidad de seguridad. Lea atentamente estos avisos, haga click sobre ellos, y elija permitir a la aplicación ejecutarse. Pues está ejecutando una aplicación segura.

-Si su navegador intenta cargar el simulador y aparece un cuadro con un error en rojo, la configuración de su navegador y de java está establecida en un nivel demasiado restrictivo. Vaya a panel de control, haga click sobre el icono JAVA, y en la pestaña seguridad baje el nivel de seguridad a medio. Haga click en aplicar y reinicie su navegador.

También es importante, especialmente en Mozilla Firefox, que tenga usted instalado el plugin de Java y que además lo seleccione como activo. Puede explorar las opciones de seguridad de su navegador, y busque en especial los permisos sobre plugins como JAVA. Actívelos.

Por último le recomiendo que, una vez que haya terminado de aprender con el simulador, reestablezca su nivel de seguridad al nivel alto, para estar a salvo de posibles ataques de otras aplicaciones que funcione sobre Java y que vengan de internet.

3 ETAPAS, INTERFAZ Y MANUAL DE USUARIO.

3.1 ETAPAS IMPLEMENTADAS

He dividido las 28 etapas implementadas con amplificadores operacionales en cuatro grupos:

1) Etapas en las cuales los amplificadores operacionales están configurados con realimentación negativa:

a) Etapa amplificadora inversora.
b) Etapa amplificadora no inversora.

c) Etapa amplificadora sumadora.
d) Etapa amplificadora restadora.
e) Etapa integradora.
f) Etapa derivadora.
g) Amplificador de instrumentación.
h) Amplificador en puente.
i) Conversor V/I.
j) Conversor I/V.
k) Etapa logarítmica.
l) Etapa anti logarítmica.
m) Rectificador de precisión.

2) Etapas en las cuales los amplificadores operacionales están configurados sin realimentación:

a) Comparador de Nivel.
b) Comparador de Ventana.

3) Etapas en las cuales los amplificadores operacionales están configurados con realimentación positiva:

a) Comparador con Histéresis.
b) Comparador con Histéresis Inversor.
c) Astable.
d) Monoestable
e) Generador de Onda Triangular
f) Generador de Barrido
g) Conversor Tensión-Frecuencia.
h) Oscilador en Puente de Wien.
i) Oscilador de Retardo de Fase.

4) Etapas didácticas en las cuales los amplificadores operacionales se presentan en diferentes configuraciones y el alumno debe, analizarlas y averiguar qué tipo de comportamiento tendrán. Pudiendo posteriormente simularlas comprobando si dedujo correctamente su respuesta.

a) Circuito 1.
b) Circuito 2.
c) Circuito 3.
d) Circuito 4.

Al abrir cada una de las etapas, éstas aparecen dibujadas en la ventana del simulador dinámico, con todos sus componentes electrónicos, por lo que podemos aprender sus componentes y conexiones en dicha pantalla. Así mismo se dibujan mediante etiquetas de texto las fórmulas mas importantes para el análisis de cada etapa.

Sin embargo, el simulador está creado para ser un una herramienta complementaria a la docencia, por lo que se recomienda para profundizar en el conocimiento de cada una de las etapas y de sus peculiaridades, leer el libro “Electrónica Analógica” de Don Jesús Navarro Artigas, que se encuentra disponible en la biblioteca Hypatia, u otros similares.

3.2 INTERFAZ, SIMULACIÓN DINÁMICA COMPARATIVA.

El objetivo de todos las simulaciones es ilustrar de manera gráfica el funcionamiento de las diferentes etapas, de manera que el usuario, mediante la introducción de una serie de datos, valores de tensión/es de entrada, valores de componentes (resistencias, condensadores, etc…), puede observar en una pantalla, a modo de osciloscopio, la evolución estática de entradas y salidas, y la evolución dinámica de entradas y salidas de manera gráfica y no analítica. De esta forma se están dando dos informaciones de cada etapa, que son complementarias, y que sustituyen a la información total dinámica y analítica de la etapa, que normalmente incluyen todos los simuladores complejos y comerciales. Es decir, se está dividiendo la tarea de estudio y aprendizaje en dos partes más simples.

Además se está añadiendo un concepto nuevo en la forma de aprendizaje de estas etapas, que proviene tanto de la contrastada experiencia docente del director del proyecto como de la propia experiencia docente del autor, y de los cursillos realizados por este último sobre métodos de estudios, memoria, audiencia y concentración, siguiendo el “Método Ilvem” cuyos autores son los hermanos Arnaldo y Horacio Alberto Krell.

Una de las bases de este método es aumentar el número de conexiones neuronales que nuestro cerebro realiza al aprender, y para hacer esto tratan de que el cerebro cree más relaciones comparativas de todo tipo.

Así pues, no se ha realizado una representación dinámica analítica profunda como es

común en los simuladores tradicionales y comerciales, sino que se ha realizado una representación dinámica comparativa, mediante códigos de colores y movimiento y pocos datos analíticos.

Este método puede parecer un concepto muy novedoso o simple como para funcionar, pero no lo es, ya que se creó en 1974 y es tremendamente eficaz a pesar de su sencillez, existiendo escuelas de dicho método en 24 países del mundo y 3 continentes.

Para que nos hagamos una idea de cómo funciona pondré un ejemplo:

Un estudiante realiza un resumen, o esquema de cualquier materia. Aunque tenga poca idea, lo subdivirá en partes, pero al separar en partes sin darse cuenta esta creando relaciones comparativas entre ellas, necesarias para separar las partes.

Si alguien con experiencia le enseña, seguro que añade diferentes estilos de subrayados, crea zonas y probablemente señala diferencias, creando más relaciones entre las diferentes partes del esquema, que así resulta más eficaz.

Por último, un buen profesor, introduciendo el uso de colores y pequeños dibujos, lo convertirá en un esquema más fácil de recordar y que perdurará en la memoria más tiempo. Al establecer relaciones adicionales, usando esta vez el color como herramienta.

Por fin, gracias a las nuevas tecnologías, hoy en día es posible hacer un esquema, que además de separaciones, diferentes tipos de letra, subrayados y colores, incluya animaciones, es decir movimiento. Y éste es un recurso definitivo para el aprendizaje rápido, como en mi experiencia docente he podido comprobar. El alumno de hoy en comparación con el de hace quince o veinte años, nace en un mundo rodeado de tecnología, y los programadores de ésta saben de la gran importancia del color y del movimiento, su capacidad de atraer la atención del consumidor, ya que están creando relaciones en su mente que perdurarán más que un simple texto llano o un dibujo estático, dada la dominante visual en el procesado de la información, propia de los humanos.

He querido explayarme en este concepto tan simple pero tan importante, pues para mi tiene suma importancia, en realidad es “mi granito de arena” de contribución a la mejora del aprendizaje del alumno. Y además es la actualización real de uno de los objetivos del proyecto, la sencillez y el aprendizaje rápido, que los estudiantes demandan en este momento.

También he creado una interfaz gráfica semejante para todas las etapas, se trata de repetir conceptos de una a otra, es decir, la distribución de sus componentes como, por ejemplo, campos de texto para la introducción de valores, etiquetas, botones simples y de selección, pantalla de salida de simulación, es similar en todas ellas. Es decir se trata de que una vez observada una, sea fácil intuir el funcionamiento de las restantes. Y por lo

tanto el simulador resulte sencillo.

3.3 EXPLICACION DE LA INTERFAZ, MANUAL DE USO.

En la interfaz se observan tres partes claramente diferenciadas, que son comunes a todos los applets que simulan las etapas: (Figura 1).

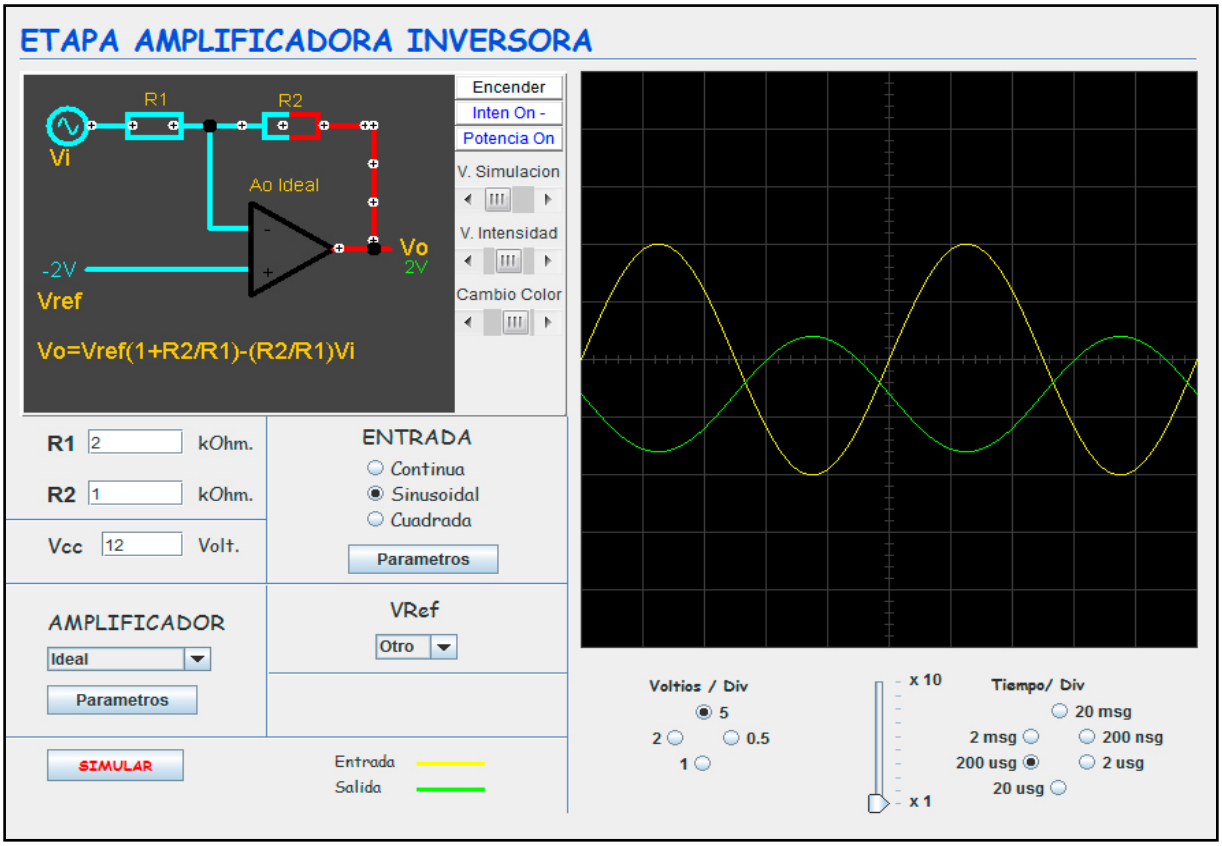

Figura 1.

-Una **ventana de simulación dinámica**, arriba y a la izquierda. Y a su derecha 3 botones y 3 deslizadores, para manejar la simulación dinámica (figura 2).

-Una **ventana de simulación estática** (osciloscopio) arriba a la derecha. Y debajo botones y deslizadores para regularla (figura 3).

-Un **Área de introducción de datos**: se sitúa debajo de la ventana de simulación dinámica, con campos de texto, botones y deslizadores, que sirven para introducir y seleccionar los valores de diferentes características del circuito, como resistencia, capacidad, tensiones de entrada y alimentación, amplitudes y frecuencias (figura 4).

3.3.1 Ventana de simulación dinámica:

Para controlar la representación de la simulación llevada a cabo, se cuentas con los siguientes botones/deslizadores (figura 2):

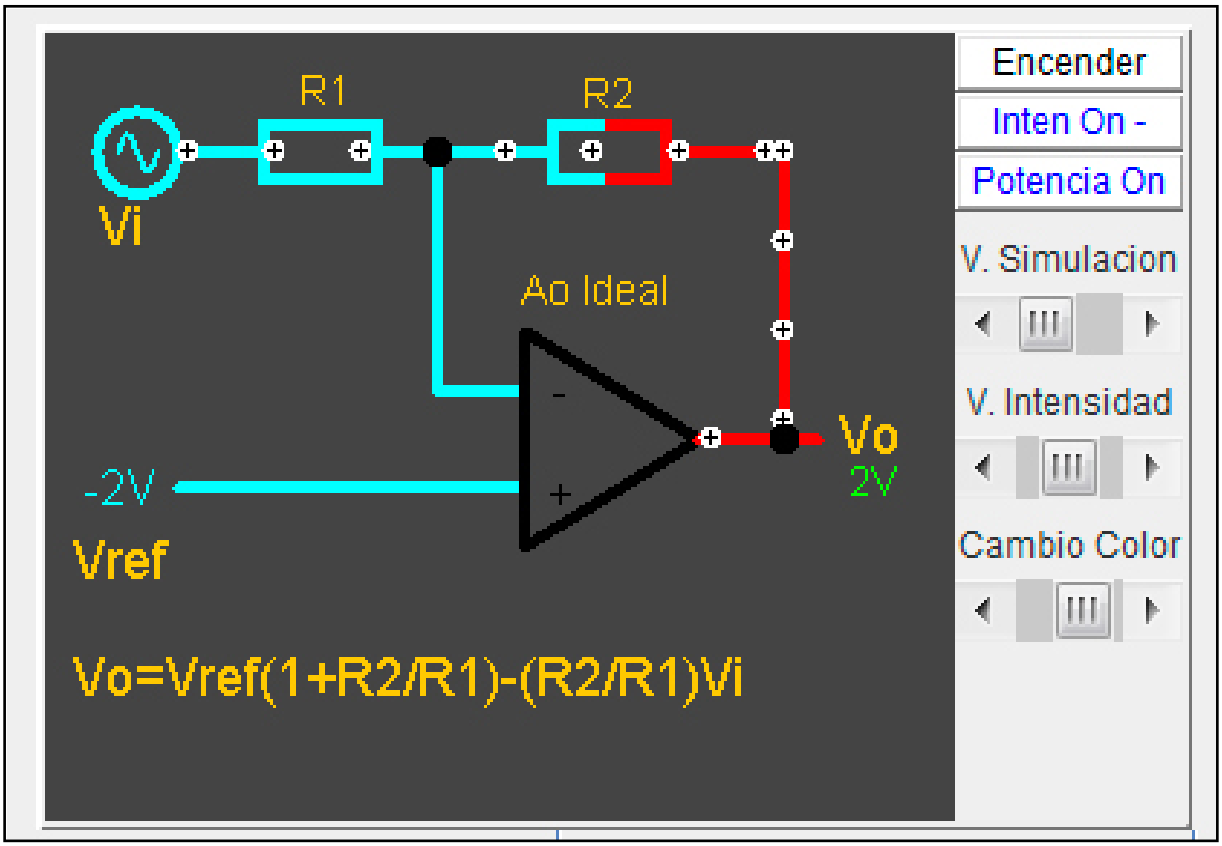

Figura 2.

-Botón "Encender/Apagar": Activa o desactiva el cálculo de corrientes y tensiones en el circuito.

-Botón "Intensidad Off/On+/On-": Permite ver o no las intensidades del circuito, representadas como protones que avanzan (+) en el sentido de la corriente de forma convencional, o como electrones (-) que avanzan en sentido no convencional.

-Botón "Voltaje/Potencia/Off": Permite ver o no la representación del Voltaje o la Potencia, sobre los elementos. El color rojo significa un voltaje positivo, el negro voltaje nulo y el color cian un voltaje negativo. Si hablamos de potencias, el rojo significa potencia consumida, el negro potencia nula y el cian potencia entregada.

-Deslizador "V.Simulación": Permite incrementar o decrementar la velocidad de cálculo y representación del simulador. Si las frecuencias introducidas son muy altas, será

difícil apreciar los cambios en el circuito; moviendo hacia la izquierda el deslizador se podrán apreciar mejor.

-Deslizador "V.Intensidad": Cuando se ha seleccionado la representación de las intensidades permite incrementar o decrementar la velocidad con que los protones o electrones se mueven. Si se mueven demasiado rápido, desplazando el deslizador hacia la izquierda, disminuye su velocidad.

-Deslizador "Cambio de Color": Al representar el voltaje o la potencia, los diferentes valores son representados con una escala de 250 colores que va del rojo al cian, pasando por el negro. Si no se ve ningún color es necesario mover el deslizador hacia la derecha, y si sólo se ve rojo negro y cian hay que mover el deslizador hacia la izquierda. De esta forma se podrán apreciar variaciones en voltaje y potencia y comparar los diferentes elementos.

3.3.2 Ventana de simulación estática:

Como podrás ver, se parece mucho a un osciloscopio.(Figura 3)

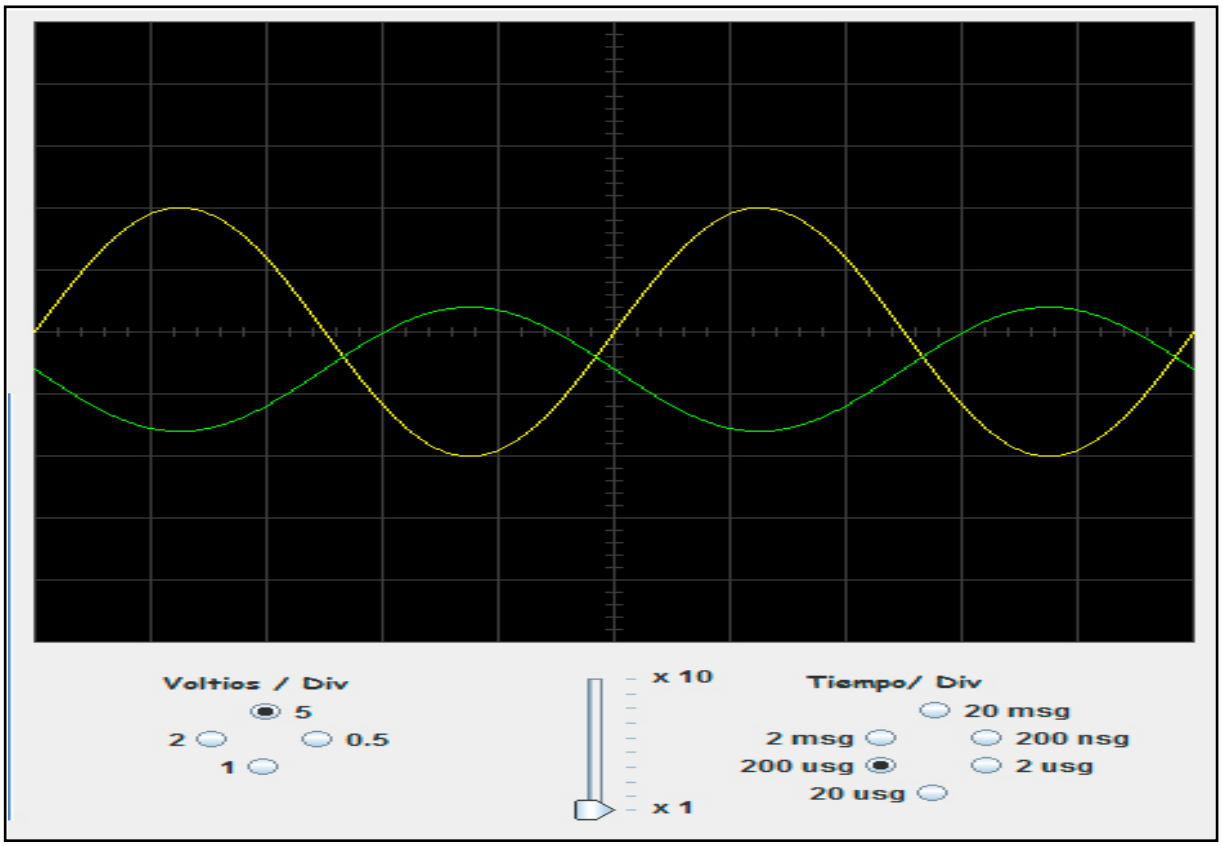

Figura 3.

Tiene dos ejes, en el eje horizontal se representa el tiempo y en el eje vertical la tensión o intensidad. Debajo de el, hay 2 tipos de botones. Unos están dedicados a cambiar la escala de voltaje de representación del osciloscopio y otros a cambiar la escala de tiempo del osciloscopio. También hay un deslizador que permite ajustar la escala de tiempo con mayor sensibilidad.

En todos los casos se ha pretendido que las simulaciones estáticas sean un reflejo fiel o una buena aproximación de lo que ocurre en un laboratorio cuando se observa el funcionamiento de estas etapas en el osciloscopio.

3.3.3 Área de introducción de datos:

Está área se adapta a cada circuito. Sin embargo hay un botón, el más importante del simulador, que permanece fijo, es el botón "Simular" (Figura 4). Al apretarlo, y si todos los valores han sido correctamente introducidos, aparecerán las simulación dinámica y estática del circuito. Cada vez que se cambien los valores de los campos de texto, deberá apretarse el botón "Simular" para introducir los cambios tanto en el simulador estático como en el dinámico.

Figura 4.

Se ha tenido especial cuidado, de que en los campos de textos sólo puedan introducir por teclado valores coherentes con las características de la etapa, asegurándose así un buen funcionamiento del programa. Por ejemplo, si para valores de resistencias o condensadores se introducen números negativos o de valor `0´, o si en cualquier campo de texto se escriben caracteres no numéricos, el programa abre una ventana de información donde nos advierte que son incorrectos (figura 5).

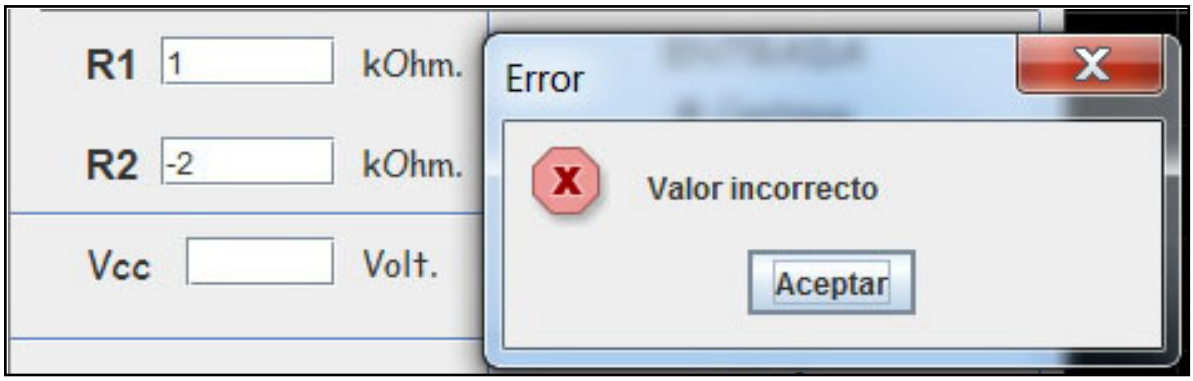

Figura5

También es posible introducir manejar señales de entrada de diferentes tipos: continua, triangular, sinusoidal o cuadrada, dependiendo de la etapa (Figura 6).

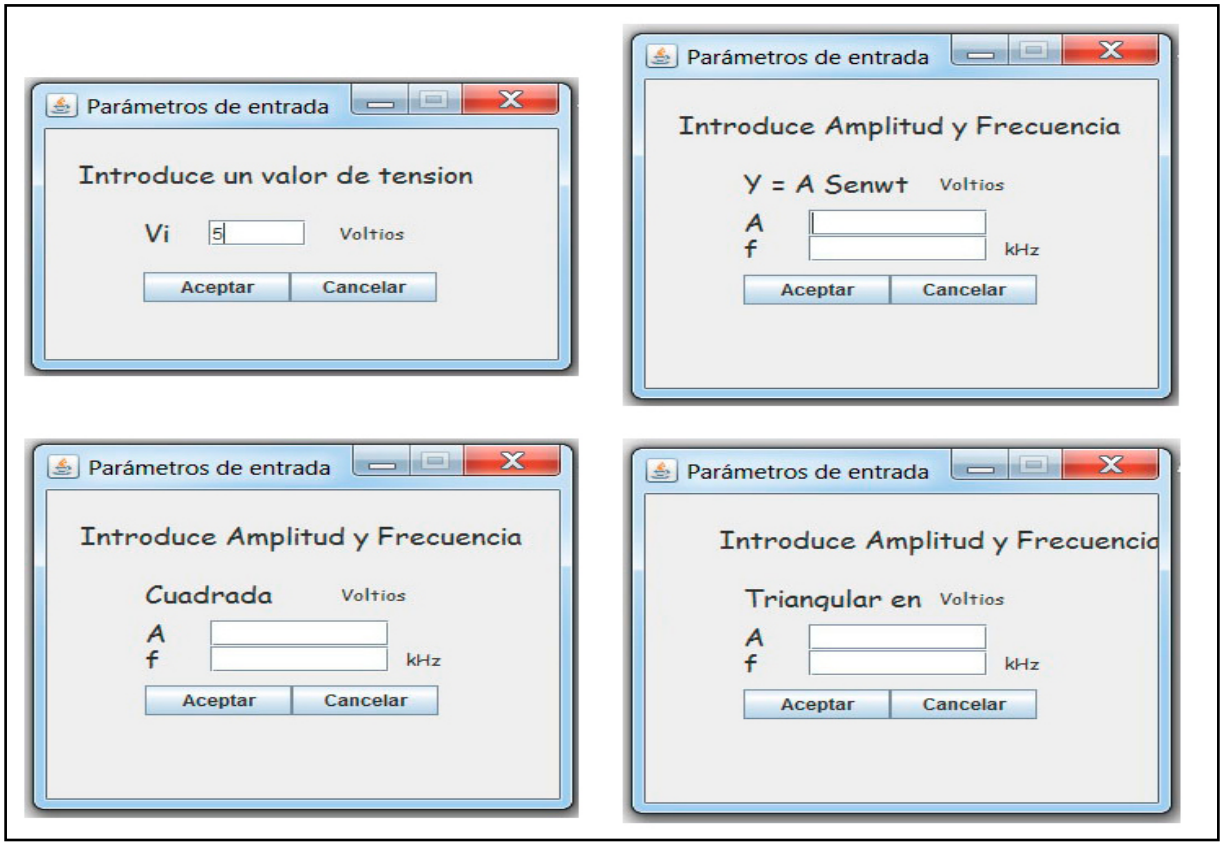

Figura 6.

Hay etapas en las que aparece un cuadro de texto para introducir la tensión de alimentación de los amplificadores operacionales, como son las etapas amplificadora inversora y no inversora y algunas con realimentación positiva. En esos casos los applets se han programado para que, a partir del valor de entrada introducido por el usuario, las alimentaciones sean simétricas. Y en el resto de applets el amplificador operacional está alimentado simétricamente a ±12V por defecto.

Una opción muy interesante desarrollada en algunas etapas es la posibilidad de elegir un amplificador operacional real como es el OP741 o el OP07. En este caso aparecerá también un botón "Parámetros" (Figura 7).

Dependiendo del amplificador elegido, el programa realiza la simulación con unos valores u otros. Concretamente, en las etapas amplificadora inversora y no inversora, si se elige uno de los dos amplificadores reales (741 o OP07), resulta una simulación sensible a la ganancia, ancho de banda, impedancia de entrada y limitación de corriente.

De esta manera el valor de la salida y su correspondiente forma de onda quedan modificados respecto al caso ideal, para determinados valores de entrada introducidos por el usuario.

En el caso de las etapas integradora y derivadora sucede exactamente lo mismo, pero en este caso la limitación de corriente no se ha tenido en cuenta.

Así, por ejemplo, trabajando a frecuencias elevadas (1 MHz) el amplificador no puede proporcionar la misma salida (puesto que la ganancia disminuye drásticamente) que trabajando a frecuencias más bajas. O, si la etapa alimenta una carga cuyo valor demanda una corriente que supera, la que puede dar el operacional real, la salida ya no será la que daría en el caso ideal, sino que quedará limitada a $I_{max} \cdot R_{carga}$.

Pulsando el botón de Parámetros que se observa en la interfaz, se abre una ventana donde se pueden observar los parámetros más característicos del amplificador operacional que en ese momento esté seleccionado, tales como Resistencia de entrada (Ri), Resistencia de salida (Ro), Ganancia en lazo abierto (Ao), Ancho de banda en lazo abierto, Frecuencia de transición, Rechazo en modo común (CMRR), Slew rate, Limitación de corriente.

Figura 7.

En el amplificador de instrumentación y el amplificador en puente se ha dispuesto un control deslizante con el que podremos variar el valor del factor K (amplificador de instrumentación) o de ΔR (amplificador en puente). Estos parámetros son característicos de las resistencias variables (NTC, PTC, LDR, etc…) pertenecientes a este tipo de etapas, como muestra la figura 8, y cuyo valor varía con cambios de magnitudes externas, como pueden ser TEMPERATURA, INTENSIDAD LUMÍNICA, etc.

Así pues, mediante este control deslizante, podemos variar la resistencia como si de la incidencia de un fenómeno externo se tratara. Esta variación se traducirá en un cambio en la salida que se podrá observar en tiempo real. Mediante un campo de texto, el usuario tiene la posibilidad de introducir, además, el porcentaje de variación de ΔR. Éste se ha limitado para que su rango sea de entre 0 y 20 %. Si introducimos un valor fuera de este rango, el programa abre una ventana de información y nos dice que estamos fuera del rango (Figura 8).

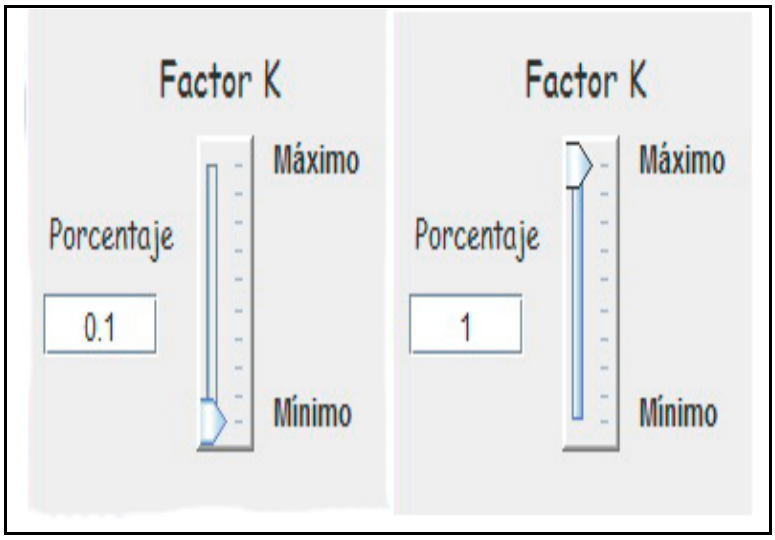

Figura 8.

En los ejercicios planteados Circuito 1 al 4, aparecerán pantallas de texto indicándo los enunciados del ejercicio a resolver y también botones que al apretarlos mostrarán la solución al ejercicio (Figura 9).

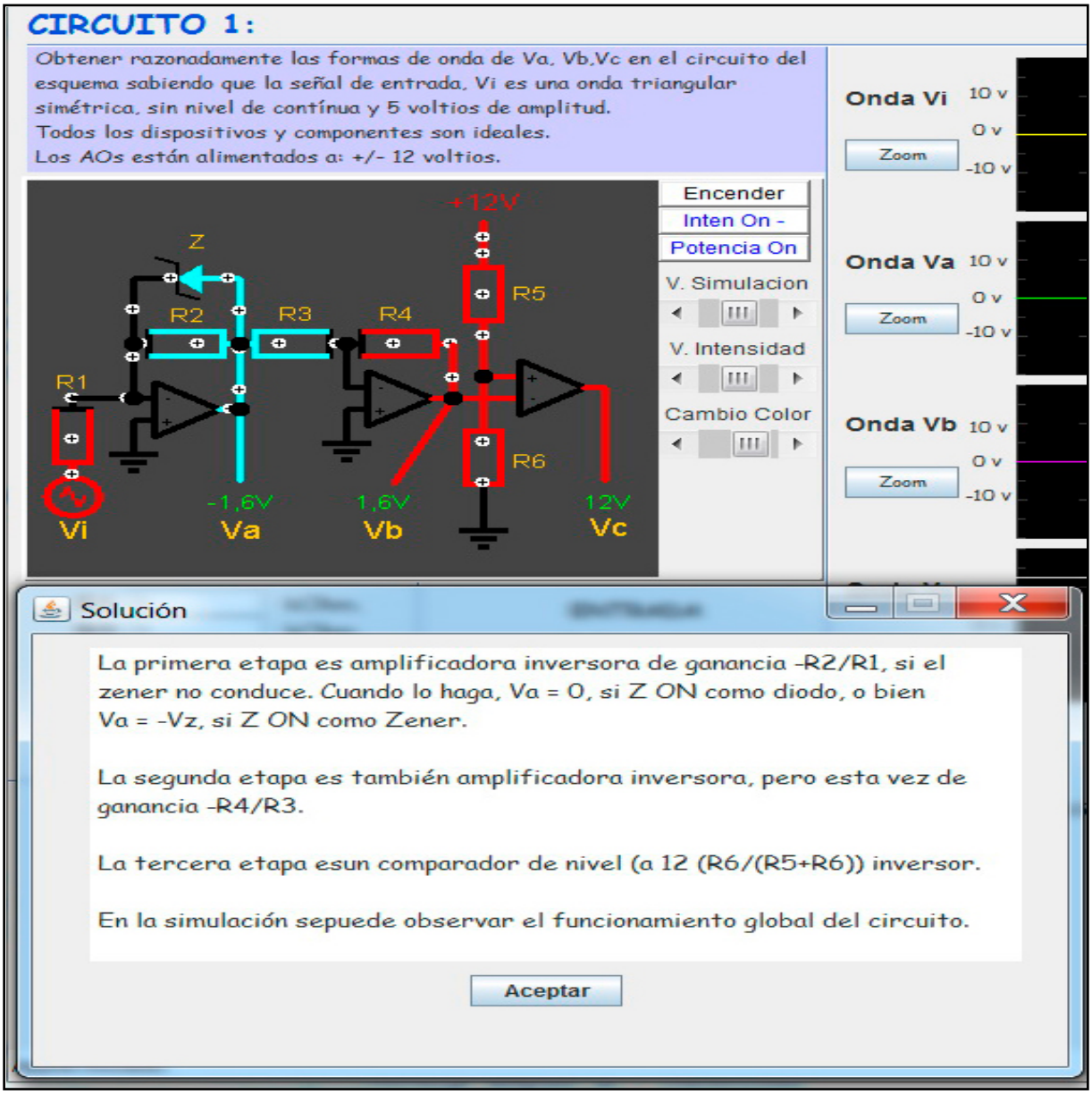

Figura 9.

3.4 ETAPAS CON INTERFAZ ESPECIAL

En algunas de las etapas, para poder apreciar los resultados mejor, se han inplementado botones de control diferentes o paneles de simulación estática adaptados al resultado de la etapa.

3.4.1 ETAPAS LOGARÍTMICA Y ANTILOGARÍTMICA

En ambas etapas el rango de valores que puede alcanzar la salida es muy diferente al que tiene la entrada, pues en la etapa logarítmica la salida, para valores convencionales de la tensión de entrada y resistencia R1, es inapreciable y, por el contrario, en la etapa

antilogarítmica se sale del rango en el que se muestra la entrada.

Por este motivo, en el caso de estas etapas se han dispuesto selectores independientes de fondo de escala, es decir uno para la entrada y otro para la salida, con valores de rango más pequeños o más grandes según sea necesario (figura 10).

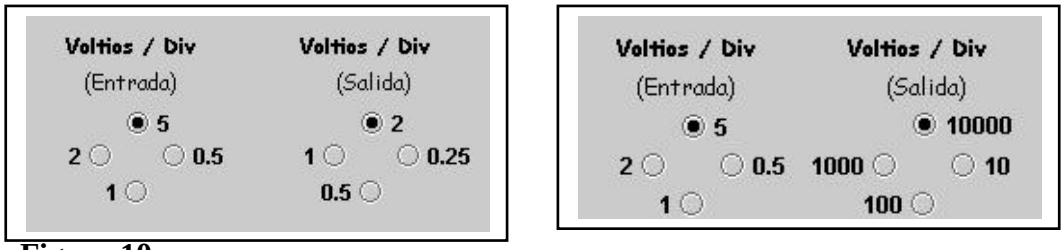

Figura 10

3.4.2 OSCILADORES EN PUENTE DE WIEN Y DE RETARDO DE FASE

En estas dos etapas se ha tenido en cuenta, a la hora de simular, las dos situaciones que se pueden dar en el momento inicial del funcionamiento de la etapa, esto es que, según los valores que introduzca el usuario, el oscilador pueda arrancar o no. En el primero de los casos, la señal se representa evolucionando desde 0 hasta el régimen permanente por medio de una oscilación de amplitud creciente, mientras que, en el segundo caso, la señal aparece como una onda senoidal amortiguada cuya amplitud decreciente tiende a 0.

Además se han dotado a estos dos applets de dos campos de texto donde el programa mostrará al usuario la frecuencia de la oscilación y la amplitud a la que tendería si se actuase sobre la ganancia del amplificador oportunamente (condición de mantenimiento)

Al igual que en casos anteriores, el valor de R3 aparece por defecto dependiendo de los valores de R1 y R2, y siempre y cuando se cumpla la condición de arranque del oscilador. En caso contrario, el campo de texto de R3 queda sin valor.

3.4.3 CIRCUITO 3

En esta etapa se pide obtener Vo en función de las entradas V1 y V2 y de los estados de conducción y corte de los diodos D1 y D2 del circuito. Para la adecuado representación del comportamiento de esta etapa se ha diseñado una pantalla más grande que las habitualmente utilizadas, donde se representan las zonas de conducción y no conducción de los diodos, así como las funciones de transferencia para cada región, de acuerdo con los

valores de R1, R2, R3, R4, V1 y V2, dado que estas regiones son cambiantes según los valores de las resistencias introducidas (Figura 11).

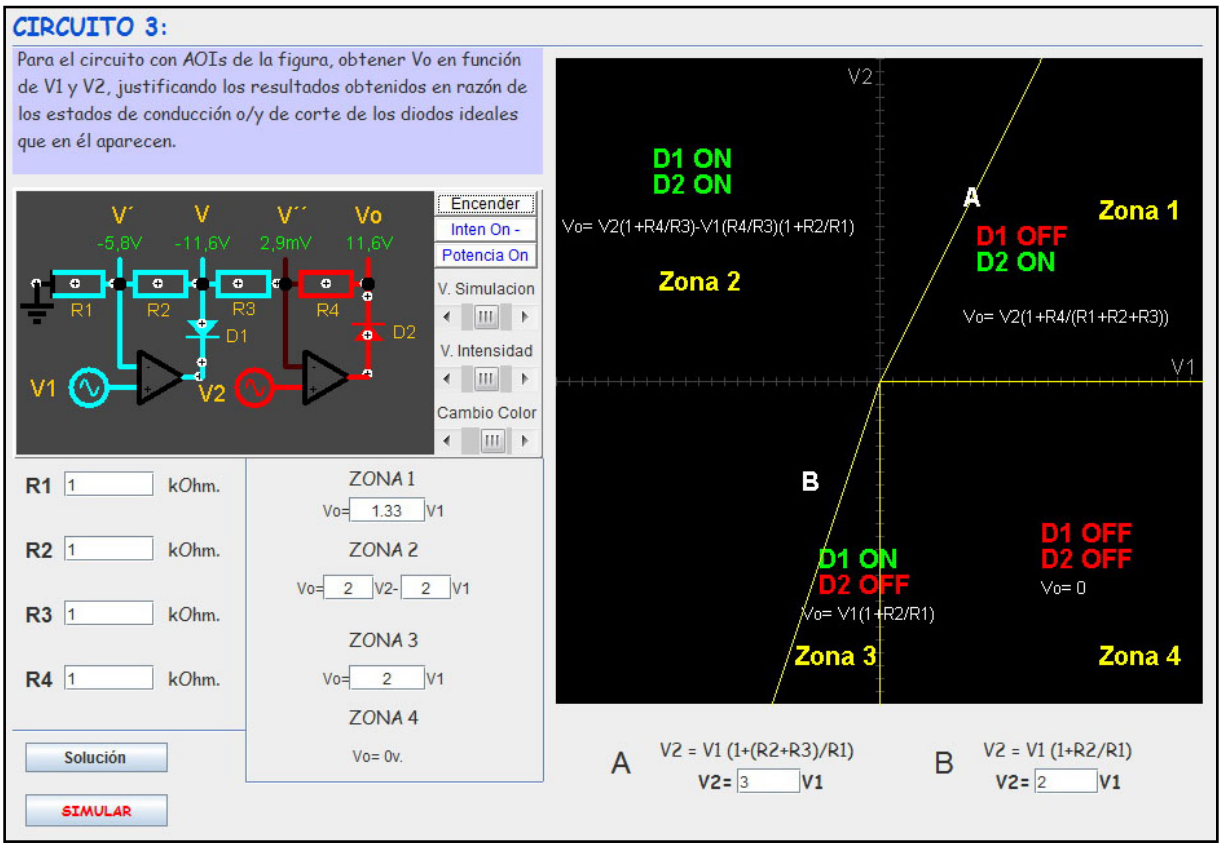

Figura 11

Se han dispuesto varios campos de texto donde se muestra, cuando se pulsa el botón simular, el valor numérico de la función de transferencia para cada zona en la que puede trabajar el circuito. También en otros campos de texto se muestra la relación entre V1 y V2 sobre las fronteras entre regiones.

Como en todos los circuitos de ejercicio, pulsando el botón de “Solución” se accederá a la ventana donde se da una explicación del comportamiento del circuito.

4 CONCLUSIONES

El diseñar e implementar una herramienta informática destinada a poner en práctica los conocimientos teóricos adquiridos en la asignatura de Electrónica Analógica por parte de sus alumnos, ha resultado un claro reto para el autor de este proyecto. No sólo porque su realización ha supuesto poner en práctica de una manera novedosa los conocimientos adquiridos durante la carrera cursada, sino porque que además le ha obligado a situarse en la perspectiva del usuario final, es decir del alumno que va a aprender con la herramienta desarrollada. Por ello desde la primera etapa, en la que se estudió el problema planteado, se pensó en agilizar y facilitar el proceso de aprendizaje del alumno en dicha asignatura.

Tras la finalización del proyecto aquí presentado, es factible comprobar que los objetivos inicialmente formulados, convenientemente tenidos en cuenta durante todo el desarrollo del mismo, se han cumplido de manera favorable, y mas que satisfactoria, consiguiendo de esta forma una aplicación amena, de uso fácil y totalmente práctica que enfrenta al alumno con los problemas planteados en la asignatura, permitiendo su interacción con los mismos y su rápida asimilación.

5 CONTENIDOS DEL CD ADJUNTO Y ORGANIZACIÓN DEL PROYECTO

Dado que el propósito de la memoria no es la de dar una descripción profunda del código fuente, ni se pretende realizar un curso de programación, no se incluye código fuente dentro del presente texto.

Seria de proceder dejar el código fuente para un Anexo, sin embargo, la aplicación ha terminado conteniendo 166.000 palabras y 37.000 líneas de código que en la letra Times New Roman tamaño 12, y con un tamaño A4 de hoja, ocuparían aproximadamente 925 páginas. Así pues, para reducir gasto en papel y el volumen físico del proyecto, se entrega todo el código de la , en este cd, dentro de una carpeta nombrada “Código Fuente” que a su vez contiene archivos en formato pdf de cada una de las clases Java implementadas.

El manual de Usuario se ha incluido en la memoria, pero además, desde el enlace “Instrucciones de manejo” de la pagina web principal de arranque del proyecto, se puede acceder a una versión reducida, para que el alumno no necesite recurrir a la lectura completa de la memoria.

También se ha incluido en la página web principal, abajo a la izquierda, un link a una pequeña guía para resolver problemas con java y los navegadores, que ha sido descrita con mayor amplitud en el punto 2.7 de esta memoria.

Se entregan dos versiones del proyecto, una versión para servidor de menor peso en KBytes, dentro de la carpeta “Versión Servidor del Simulador”, y otra versión con mayor peso en KBytes para ejecutar la aplicación en cliente, dentro de la carpeta “Versión Cliente del Simulador”.

En ambas versiones, la forma de arrancar el proyecto es haciendo click sobre el archivo “SimuladorAO.html”, que contiene enlaces a todas las etapas simuladas.

También se puede arrancar cada una de las etapas individualmente, desde dentro de la carpeta “Proyecto Applets”, haciendo click sobre cada uno de los archivos HTML que contiene.

Este cd incluye también una copia de la memoria del proyecto.

6 BIBLIOGRAFÍA

Además de las direcciones web que han sido principales para la ejecución de este proyecto, quiero mencionar también el apoyo recibido de diferentes foros de programación en java, gracias a los cuales me he ahorrado gran cantidad de tiempo y esfuerzo al resolverme problemas que las guías standard no aclaran.

[1] Java SE 7 API and developer guide Documentation.
http://docs.oracle.com/javase/search.html

[2] Información sobre Java.
http://es.wikipedia.org/wiki/Applet_Java

[3] Electrónica Analógica
Jesús Navarro Artigas, Textos docentes de la universidad de Zaragoza.

[4] Proyecto final de carrera de la universidad de Zaragoza "Herramienta informática de apoyo a la docencia en Electrónica Analógica basada en aplicaciones web. De Jorge Mañas Gregorio.

[5] Página web de Paul Falstad
http://www.falstad.com

[6] Método de estudio, memoria, audiencia y concentración.
Instituto de técnicas de estudio ILVEM. ISBN 129.740

[7] El Entorno de Programación de Fuente Abierto NetBeans.
http://es.scribd.com/doc/967380/Tutorial-Netbeans

[8] La clase Graphics, Métodos Generales
http://tabasco.torreingenieria.unam.mx/gch/Curso%20de%20Java%20CD/Documentos/froufe/parte15/cap15-5.html

[9] Apuntes de amplificadores operacionales de Pablo Morcello:
http://www.ing.unlp.edu.ar/electrotecnia/tcieye/Apuntes/Amplificadores%20operacionales%20_%20Rev2011.pdf

[10] Como programar un applet. Guía básica.
http://www.sc.ehu.es/sbweb/fisica/cursoJava/applets/intro/primero.htm

[11] Painting in AWT and SWING.
http://www.oracle.com/technetwork/java/painting

[12] Book: Electronic Circuit and System Simulation Methods
Lawrence T.Pillage, Ronald A. Rohrer, Chandramouli Visveshwariah