ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

SIMUTEMP: Simulador de un Conmutador Temporal

AUTOR: Francisco Javier Martínez Mercader
DIRECTOR: Francesc Burrull i Mestres

Julio / 2004

Autor	Francisco Javier Martínez Mercader
E-mail del Autor	fjmmercader@hotmail.com
Director	Francesc Burrull i Mestres
E-mail del Director	francesc.burrull@upct.es
Título del PFC	SIMUTEMP: Simulador de un Conmutador Temporal
Descriptores	Simulador, Conmutación Temporal

Resumen

Un elemento esencial de las redes de conmutación de circuitos son los conmutadores propiamente dichos. Como complemento a su estudio se pueden utilizar simuladores didácticos, que permiten una mejor comprensión de los conmutadores, así como la obtención de datos estadísticos.

En este proyecto se ha realizado un simulador de un conmutador temporal, herramienta que servirá para la docencia en la Escuela Técnica Superior de Ingeniería de Telecomunicación (E.T.S.I.T.) de la Universidad Politécnica de Cartagena (U.P.C.T.), disponiendo además del código fuente de la herramienta.

Titulación	Ingeniero Técnico de Telecomunicación
Intensificación	Telemática
Departamento	Departamento de Tecnologías de la Información y las Comunicaciones
Fecha de Presentación	Julio - 2004

Índice General

Capítulo 1
Introducción

Un elemento esencial en las redes de conmutación de circuitos son los conmutadores propiamente dichos. Como complemento a su estudio se pueden utilizar simuladores didácticos, que permiten una mejor comprensión de los conmutadores, así como la obtención de datos estadísticos.

En este proyecto se ha relizado un simulador de un conmutador temporal, herramienta que servirá para la docencia en la Escuela Técnica Superior de Ingeniería de Telecomunicación (E.T.S.I.T.) de la Universidad Politécnica de Cartagena (U.P.C.T.), disponiendo además del código fuente de la herramienta.

Capítulo 2
Objetivos

El objetivo de este Proyecto Fin de Carrera es implementar una aplicación didáctica consistente en un simulador de conmutación en el tiempo, partiendo como base el simulador comercial de código cerrado ya realizado *TIMSWIT*. Con ello se conseguirá una aplicación de la que se dispondrá el código fuente y las siguientes mejoras respecto al *timswit*:

- La marcha atrás en todos los posibles modos de transmisión (ciclo, slot y trama).

- Facilitar la comprensión del programa mediante ventanas emergentes de ayuda que el alumno podrá usar cuando la necesite, viendo así la utilidad y la función de todos los componentes del programa.

- En la opción de multiplexación por división en el tiempo, además de la opción conocida de 4 canales, se incluyen 3 apartados más: simulación con 8 canales, con 16, y con 32, adaptando así al simulador aún más a la realidad de lo que es TDM.

- La portabilidad del programa; éste consistirá en un archivo tipo *".jar"*, que se podrá ejecutar en cualquier sistema operativo, como Windows, LINUX, etc.

- La interfaz gráfica se mejora considerablemente al pasar de un programa en MS-DOS a una ventana tipo *JFrame*, donde el entorno visual es mucho más agradable, permitiendo la interacción del alumno con el ratón, y facilitando la comprensión del algoritmo mediante elementos gráficos y ventanas de ayuda emergentes.

- El software será desarrollado en la propia Universidad Politécnica de Cartagena, sin tener que pagar ningún tipo de licencia o tener problemas con ésta, y al ser software de la propia Universidad, estará en castellano, facilitando así su comprensión al alumno (al menos al "no erasmus").

- El código se dejará abierto para posibles mejoras futuras del programa.

Capítulo 3
Análisis y diseño

3.1. Ingeniería Inversa

En el programa a desarrollar, a pesar de partir de cero en cuanto a código se refiere, el comportamiento y los algoritmos se sacan partiendo de otro programa ya hecho, por lo que para su implementación y diseño se recurrirá a la ingeniería inversa.

La ingeniería inversa consiste en desmontar un objeto para ver cómo funciona y, de ese modo, duplicar o mejorar el mismo. En el ámbito informático, existen 2 maneras de llevarla a cabo:

1. Empleando métodos de “caja negra”, que consiste en observar desde fuera el comportamiento de un programa al que se somete a una serie de casos de uso. Esto es factible en programas pequeños.

2. Descompilando un programa de modo que se obtenga el código fuente, legible por un programador, del mismo a partir del código que ejecutamos en nuestro ordenador. Es la única manera con programas grandes, como sistemas operativos.

En el ámbito del software, esta práctica puede estar prohibida, tanto por las licencias con las que se vende como por leyes (aunque en este último caso la primera opción se permite).

Es por esta última razón y por la relativa simplicidad del programa por lo que se ha decidido que se usará el método de “caja negra”, viendo cómo actúa el algoritmo y plasmándolo en el código empezando desde cero.

A continuación se comentarán brevemente las características del programa de partida, el *timswit*, características que se verán en profundidad más adelante en el simulador implementado en este proyecto fin de carrera:

Figura 1: Timswit – Menú principal

En el simulador hay 3 opciones, Opción A y Opción B que serán las que tendremos en cuenta a la hora de la implementación, y una tercera Opción C que no se ha incluido en este proyecto, que es una mezcla de conmutación espacial y temporal usando TDM, que se dejará para posibles futuros proyectos.

3.2. Opción A

La Opción A es la demostración de una multiplexación por división en el tiempo de 4 canales. En la parte superior se encuentra el emisor, el receptor, y la línea de transmisión. En la parte inferior vemos el mecanismo de control de la ejecución, que se controla mediante F1, F2 y F3. Para ir al punto final de la simulación se pulsa F4, y para reiniciar le damos a F5. El F8 es para volver al menú principal:

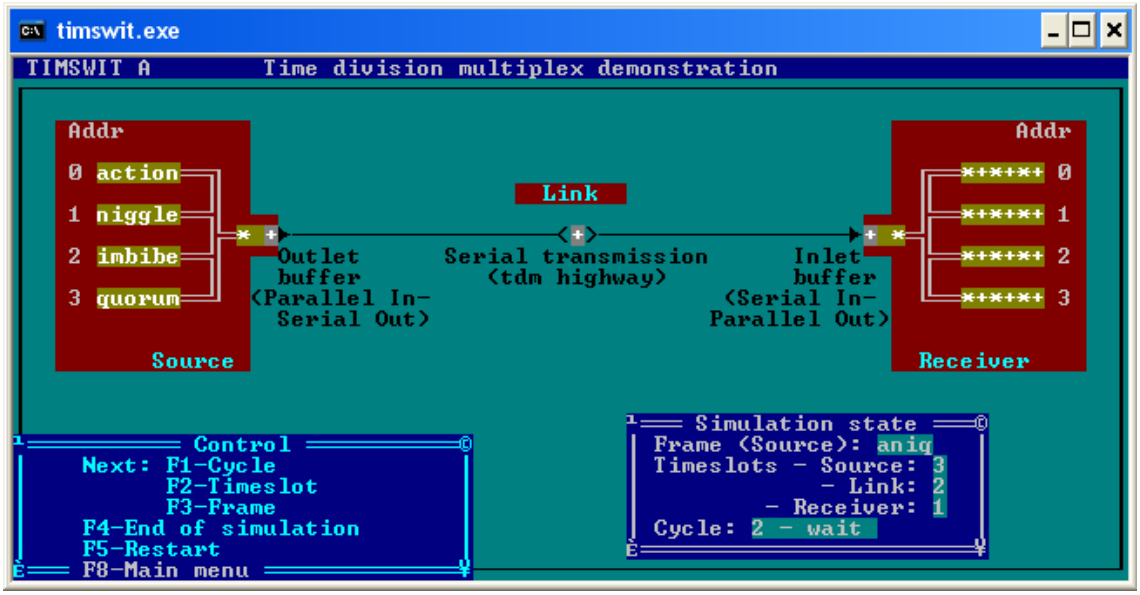


Figura 2: Timswit – Opción A

Tal como se ha descrito con anterioridad, las teclas F1, F2, y F3 controlan la ejecución paso a paso del programa, pero antes de explicar qué hace cada una, es preciso comprender los 3 diferentes ciclos en los que se desarrolla la transmisión en un slot completo de tiempo:

1. **“0 – Output”**

   Es el momento en el que una letra de la fuente pasa al buffer de salida de la misma, y en la línea se desplazan las letras, donde el buffer de entrada pasa a estar ocupado por la letra que estaba en la línea. Está representado como “0 – Output”.

2. **“1 – Input”**

   La letra que estaba en el buffer de entrada del receptor pasa a formar parte del canal que le corresponde en el mismo receptor, mientras que en la fuente desaparece la letra que queda almacenada en el buffer de salida. Este ciclo se representa por “1 – Input”.

3. **“2 – Wait”**

   Es un tiempo de espera necesario para la correcta simulación, dado que en el canal existe un retardo temporal que es el que simula este ciclo (tiempo de transmisión). Se representa mediante “2 – Wait”.

Dicho ya en qué consiste cada uno de los 3 ciclos, se explicará para qué funcionan las distintas teclas:

1. **F1**

   “Cycle”, simula la ejecución de un ciclo, pulsándolo reiteradamente se ve cada uno de los 3 ciclos de los que se compone una transmisión en un slot de tiempo.

2. **F2**

   “Timeslot”, se transmite el slot completo, es decir, se produce simultáneamente la ejecución de los 3 ciclos.

3. **F3**

   Se transmite una trama entera, es decir, 4 letras, que equivale a 4 slots y a 12 ciclos.

En cuanto a la línea, lo que inicialmente aparece también se transmite, aunque sea basura. Estos residuos están representados por los caracteres “*” y “+”, que van directamente a parar a los canales 2 y 3 (porque estos residuos estaban finalizando ya su transmisión, es como si se hubiera hecho previamente otra TDM).

3.3. Opción B

La opción B nos ofrece 3 posibilidades:

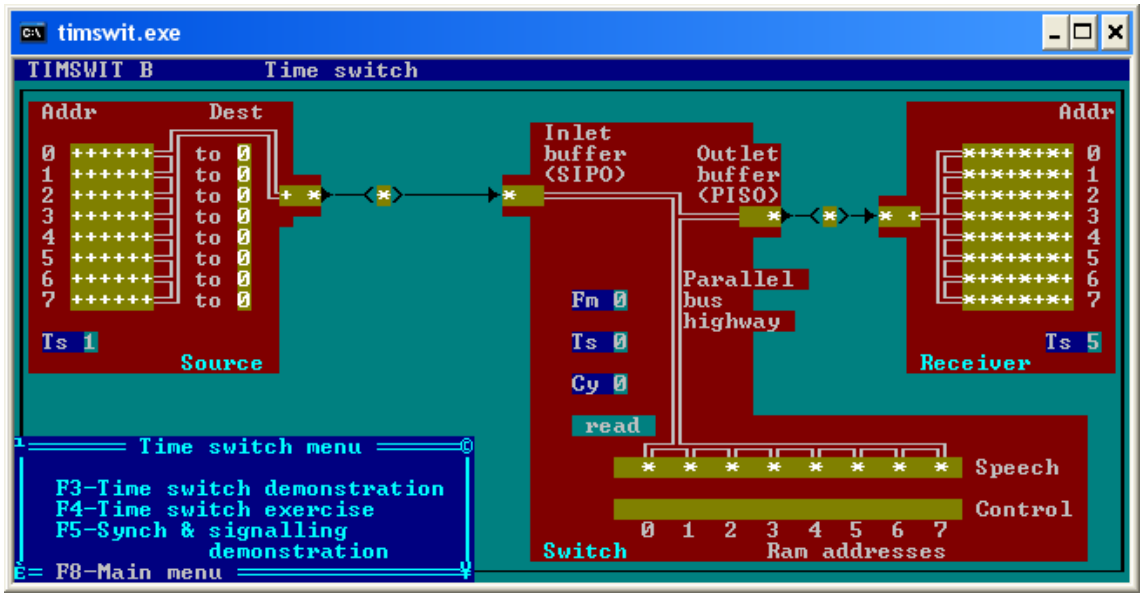


Figura 3: Timswit – Opción B

La opción F3 es una demostración de conmutación en el tiempo, es decir, una transmisión de emisor a receptor, ambos de 8 canales, y con un Switch encaminando las letras, guiándose éste por los destinos a donde debe ir cada canal.

La opción F4 es un ejercicio de lo anteriormente dicho, que para este proyecto no se va a implementar.

La opción F5 es una demostración de cómo funciona el sincronismo y la señalización, es decir, se dispone de un emisor con 8 canales, 1 de ellos destinado a la sincronización y otro para la señalización.

3.3.1. Time switch demonstration

Es la opción de la etapa T simple, donde las letras de la fuente serán encaminadas hacia el receptor a través del switch. Las letras se almacenan en speech, mientras que en control se decide cuándo se mandan al *outlet buffer*.

En la etapa T hay tres registros: *Fm, Ts* y *Cy*, que sirven para indicar cuántas tramas están pasando por la memoria de voz, qué slot de dicha trama se está almacenando en ese momento, y en qué ciclo se encuentra, respectivamente.

También nos encontramos con un módulo cuya única función es la de señalización, que está conectado de la fuente al switch.

Los ciclos ahora son *read, write, wait*:

1. **Read**

   Es el ciclo de lectura de la memoria de voz. La memoria de control es la que decide qué posición de la memoria de voz pasa al outlet buffer.

2. **Write**

   Es el ciclo de escritura en la memoria de voz. La dirección RAM que toque en ese momento es la que decide en qué posición de la memoria de voz se escribe.

3. **Wait**

   Es el ciclo de espera que hay que realizar para la correcta simulación.

La fuente y el receptor tienen un registro cada uno, el registro Ts, que indica de qué canal es la letra que se está transmitiendo (fuente) o en qué canal se está recibiendo la letra (receptor).

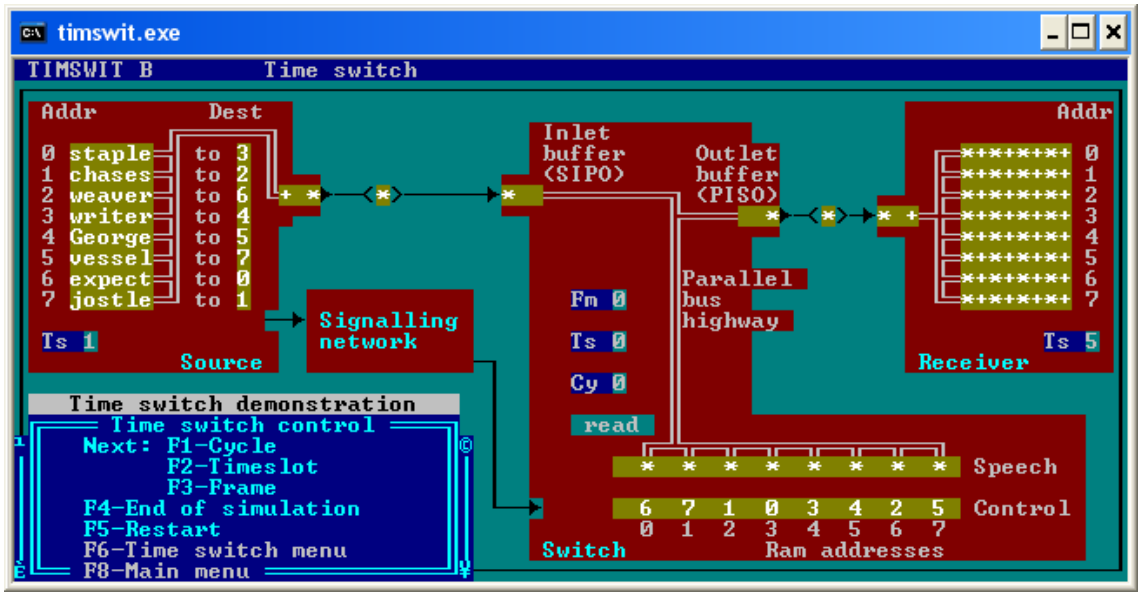


Figura 4: Opción B (F3)

3.3.2. Synch & Signalling demonstration

Tiene básicamente el mismo aspecto que la opción anterior, pero la funcionalidad cambia considerablemente.

Ya no hay 8 canales de datos, sino 6, los 2 restantes son para sincronismo (canal 0) y señalización (canal 4), y la memoria de control del switch inicialmente no está definida, se irá completando paso a paso con la ayuda del canal de señalización.

Otra novedad es el módulo de sincronismo y señalización, que por un lado se encarga de sincronizar los registros Fm, Ts y Cy, y por otro se encarga de rellenar la memoria de control para así poder mandar al outlet buffer el contenido de la memoria de voz. Si en una memoria de control está la letra X, quiere decir que ésta aún no ha sido definida, y lo que se enviará al outlet buffer será ruido.

El canal de señalización queda definido por los destinos que tomarán las letras que hay en la fuente, destinos que se forman aleatoriamente, con la excepción de los destinos 0 y 4, que están dirigidos, y sólo los tomarán los canales de sincronismo y señalización, respectivamente.

El sincronismo de multitrama se representa mediante “!!”, y el sincronismo de trama es “!”. El módulo de sincronismo y señalización se resetea cuando llega por primera vez “!!”, lo mismo pasa con el receptor.

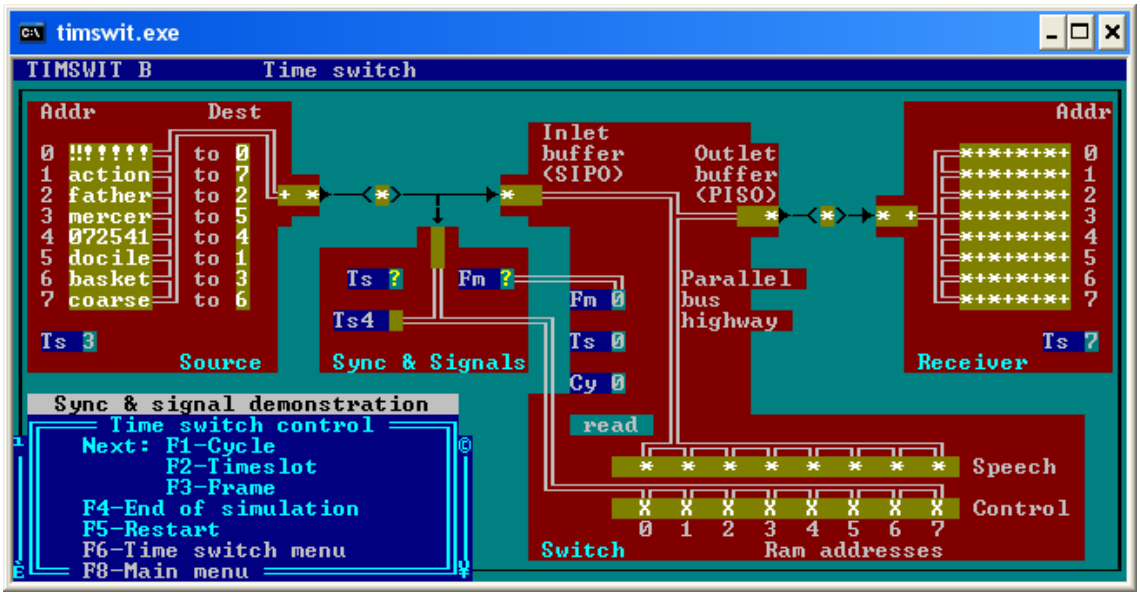


Figura 5: Opción B (F5)

Capítulo 4
Ámbito de aplicación

4.1. Elección del lenguaje de desarrollo

Se han barajado los siguientes lenguajes para desarrollar la implementación del programa:

C

El lenguaje por excelencia de programación de sistemas. Es un lenguaje de nivel medio (no se puede decir que sea de alto nivel, por incoporar muchos elementos propios del ensamblador), tremendamente ligado a UNIX (aunque es portable y hay compiladores para casi cualquier sistema operativo). Casi podríamos considerarlo como un ensamblador estructurado y portable. Difícil de aprender (no es recomendable como primer lenguaje, como sí lo podría ser *Pascal*), aunque tremendamente flexible. Bastante dado a errores, sobre todo entre programadores novatos.

C++

Evolución sobre el C. C++ sí se puede considerar de alto nivel. Es orientado a objetos, relativamente difícil de aprender (muchas características, muy complicado), pero combina la potencia y flexibilidad de C con orientación a objetos. Bastante utilizado. Dado a errores, aunque no tanto como C.

Java

Un lenguaje "de moda", de sintaxis parecido al C. Orientado a objetos (Java te obliga; C++ sólo te da la posibilidad), mucho menos flexible que C++, pensado para hacer aplicaciones interactivas más que controladores de dispositivos y sistemas operativos. Mucha aceptación popular, muchos recursos, y posibilidad de incluir programas en Java en páginas HTML (los llamados "applets").

Es un lenguaje multiplataforma, los programas Java se pueden ejecutar en cualquier plataforma soportada (Windows, UNIX, Mac, etc.), además es un lenguaje compilado e interpretado, ya que el compilador produce un código intermedio independiente del sistema llamado *bytecode*. Se necesita instalar en el ordenador la JVM (Java Virtual Machine), que es el intérprete que convierte el bytecode en código máquina. *Sun Microsystems* distribuye de forma gratuita el producto base, el llamado JDK (Java Depelopment Kit), también llamado J2SE (Java 2 Standard Edition), y se puede encontrar en la siguiente dirección : http://java.sun.com/

Elección final

Finalemente se ha elegido el lenguaje **Java** para la implementación del programa por las siguientes razones:

1. Se ha elegido en detrimento de C al ser un lenguaje orientado a objetos y por su portabilidad.

2. Frente a C++ tiene la ventaja de tener más portabilidad, se pueden usar las clases compiladas en cualquier plataforma (Windows, UNIX, etc.).

3. Una razón muy importante fue el entorno gráfico resultante, ya que Java proporciona elementos gráficos muy atractivos, sobre todo usando los componentes *swing*.

4.2. Elección del entorno de desarrollo

Para el lenguaje Java hay múltiples entornos de desarrollo, entre los que se han barajado:

JBuilder

Dirección: (Borland) http://www.borland.com/jbuilder/
Versión actual: 6.0
Plataformas: Windows, Linux, Solaris.
Licencia: La versión de evaluación, la Personal, es gratis, las avanzadas, Profesional y Enterprise son de pago.

JBuilder Foundation está diseñado para desarrolladores Java que quieran una alta productividad IDE (Entorno de Desarrollo Integrado) para crear más facilmente aplicaciones multiplataforma para Linux, Solaris y Windows.

JBuilder Foundation permite desarrollar rápidamente, compilar, ejecutar, y encontrar errores, usando las aplicaciones visuales de JBuilder o con métodos tradicionales de código.

Adicionalmente, JBuilder permite que los usuarios retoquen a su gusto y extiendan el entorno según sus necesidades de desarrollo usando Open Tools API, el cual facilita la integración de otros componentes adicionales.

Tiene la gran utilidad de visualizar los diagramas UML, además de poder desarrollar aplicaciones web con JSP y servlets.

Kawa

Versión actual: 5.0
Plataforma: Windows
Licencia: Como es habitual, versiones Profesional y Enterprise. Disponen de versión de evaluación.

NetBeans

Versión actual: 3.6
Plataforma: Todas con JVM
Licencia: Opensource
Dirección: http://www.netbeans.org/

NetBeans es una aplicación *open source* (el código del entorno está abierto a posibles modificaciones) de desarrollo escrita en java, esto quiere decir que se puede modificar el entorno de acuerdo a ciertos parámetros de licencia. Soporta, aparte de java, otros lenguajes, como C++.

Algunas de sus funciones y características son:

- Completado de código
- Soporte para escritura de servlets
- Ayudas con el código
- Ayuda on-line

Es un entorno muy cómodo en cuanto a interfaz gráfica se refiere, ya que ahorra al usuario escribir código tan sólo con arrastrar los componentes gráficos que soporta java y que están representados mediante iconos. Al añadir estos componentes, en la aplicación se genera el código correspondiente a dicha inclusión del elemento gráfico. Está recomendado por *Sun Microsystems*, aunque una de sus desventajas es que ocupa muchos recursos de memoria.

Elección final

El entorno principal elegido es el Kawa. Por su facilidad de uso y potencia, es el ideal para implementar los algoritmos, además, al ser un entorno de ventanas pero no muy sobrecargado, la programación se hace llevadera.

Para diseñar la interfaz gráfica, se ha decidido por el NetBeans, también por su gran facilidad de uso, aunque en este caso, el código que se genera al añadir componentes gráficos se ha copiado en el NetBeans y posteriormente pegado en el Kawa, ya que programar todo desde el NetBeans conllevaría mucho tiempo, por lo dicho anteriormente: consume muchos recursos de memoria, mientras que el Kawa es mucho más rápido y más accesible.

Finalmente, se ha escogido el JBuilder única y exclusivamente para realizar los diagramas UML de las clases que componen el programa.

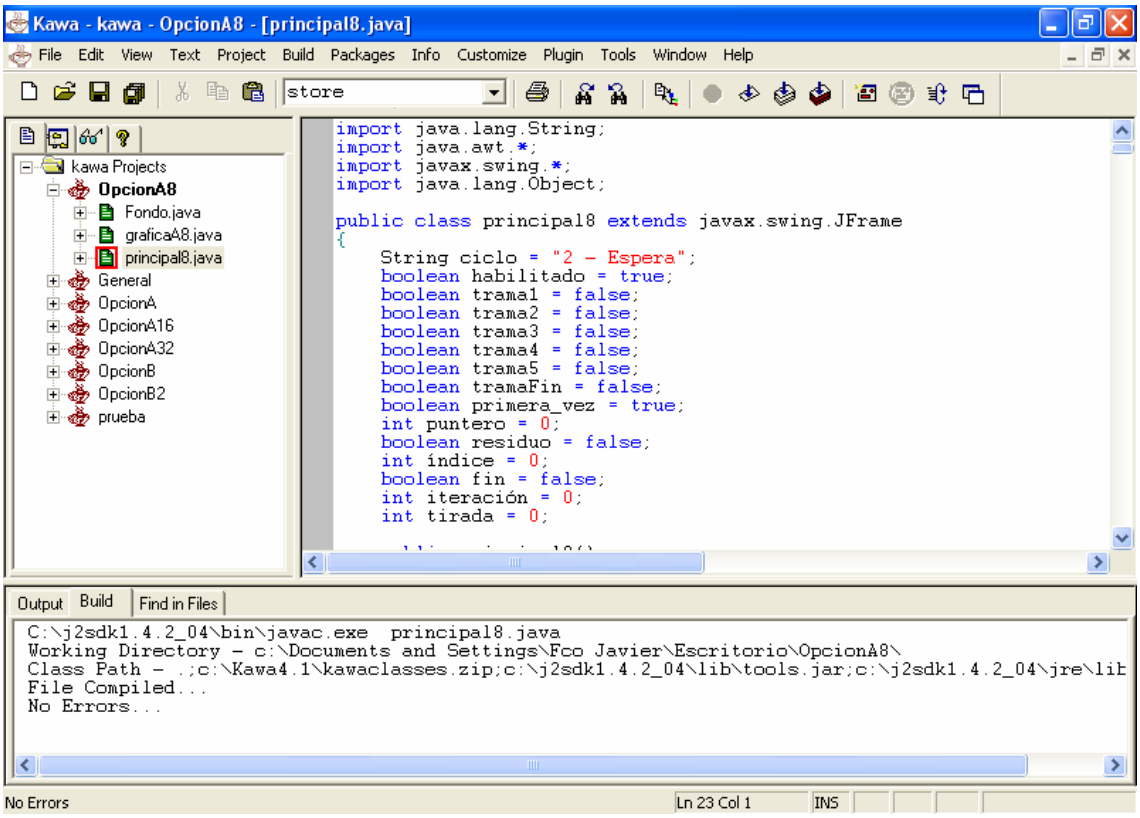


Figura 6: Ventana de Kawa

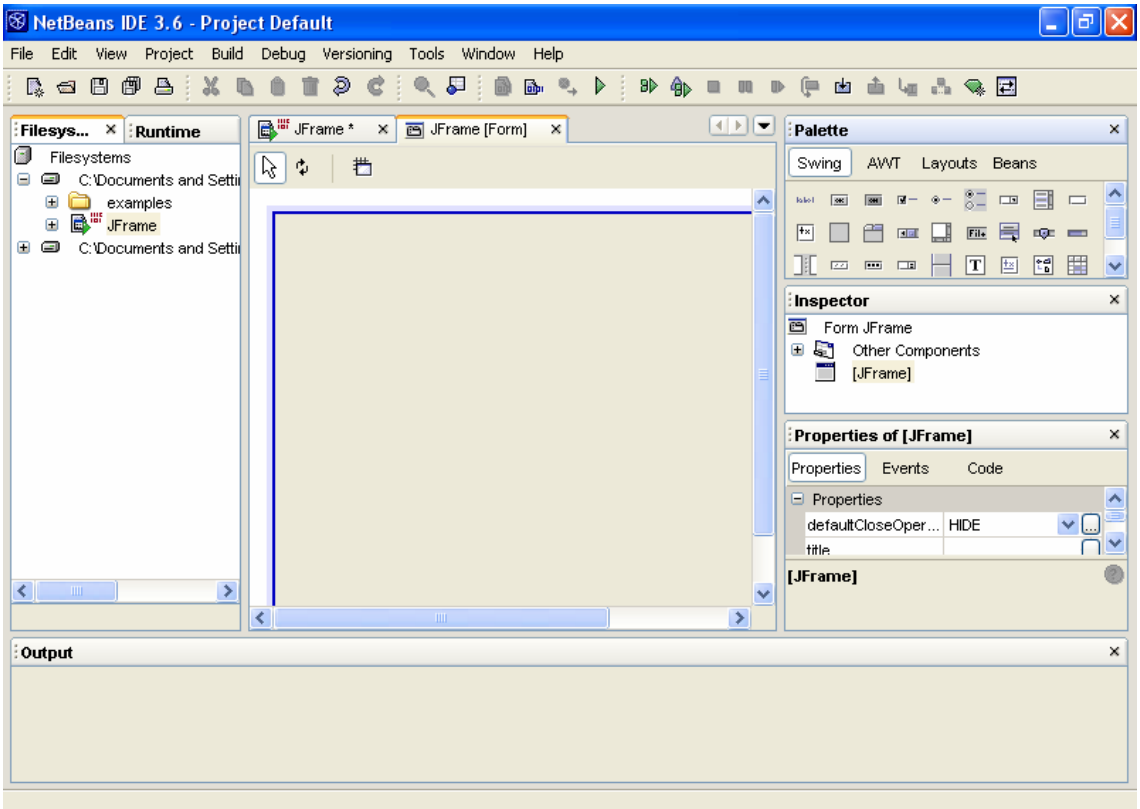


Figura 7: Ventana de NetBeans

Capítulo 5
Implementación de la aplicación

5.1. Menú principal

La aplicación comienza mediante un menú de inicio donde se pueden elegir 2 opciones: Opción A: Multiplexación por división en el tiempo, y Opción B: Conmutación temporal con TDM. Si se elige la Opción A, aparecen a la izquierda del menú 4 botones, donde se seleccionará con cuántos canales se desea hacer la simulación, y si se elige la Opción B, a la derecha de la pantalla aparecen 2 botones, donde se podrá seleccionar una conmutación con TDM simple, o con sincronismo y señalización.

Figura 8: Menú inicial del SIMUTEMP

La clase que implementa el menú es la clase principal del programa `General`, que previamente se ha señalado como 'Main Class' mediante el archivo MANIFEST.MF. Con posterioridad se explicará con todo detalle cómo se construye el archivo *.jar* final. Ésta es una clase que extiende de JFrame, donde según el botón pulsado, se mostrarán los distintos Jframes que componen el programa.

Por ejemplo, si se elige la opción de TDM de 4 canales, el botón pulsado tiene asociado un método manejador de eventos que hará lo que se pida cuando en dicho botón ocurra un evento, en este caso, de tipo ActionEvent, es decir, una pulsación. En este método se hará lo siguiente:

```
private   void   opcionA4Handler(java.awt.event.ActionEvent
evt)
{
      a4 = new OpcionA4();

      a4.show();
      (this).hide();
}
```

En el manejador ocurren 3 cosas:

1. Se crea un objeto de tipo OpcionA4, es decir, un Jframe con toda la funcionalidad del TDM de 4 canales.

2. Dicho Jframe creado, se muestra en pantalla por medio del método `show()`

3. Mediante la referencia `(this)` nos referimos al Jframe del propio menú, de manera que como ya no nos hará falta, ocultamos el menú con el método `hide()`.

De esta manera, todos los botones del menú tienen asociado un manejador que creará y mostrará Jframes del tipo que corresponda según el tipo de simulación elegida, y finalmente, cerrará el menú.

Con esta porción de código, el aspecto de la ventana cambia, sobre todo en los botones, es una forma de adaptarla gráficamente a Windows XP:

```
try
{
      UIManager.setLookAndFeel("com.sun.java.swing.plaf.window
      s.WindowsLookAndFeel");
       SwingUtilities.updateComponentTreeUI(this);
}
catch (Exception e){}
```

A continuación, pasaremos a explicar la implementación de todas las opciones del programa.

5.2. Opción A

En la opción A se presentan 4 formas de hacer un multiplexado por división en el tiempo, con 4 canales, con 8, con 16, y con 32. Explicaremos la de 4 canales, ya que es la más simple, y por no redundar en las otras, ya que su funcionamiento consiste en un simple escalado.

Lo primero que se encuentra es el módulo fuente, donde se hallan las letras a transmitir, las cuales se han separado en paneles de distinto color para hacer hincapié en el tratamiento individualizado de cada letra, y así facilitar la comprensión del algoritmo.

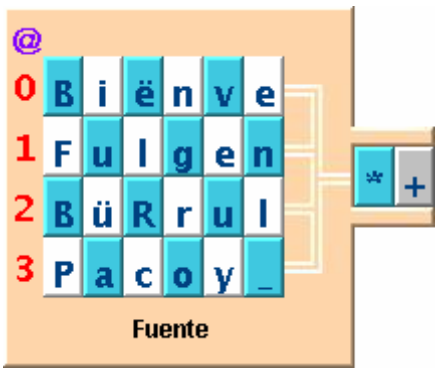


Figura 9: Fuente transmisora con 4 canales

En el momento en que en la fuente se produce una transmisión, aparece una flecha roja en las líneas que van desde el canal que esté transmitiendo hasta el buffer de salida, para indicar que la letra está saliendo de la fuente.

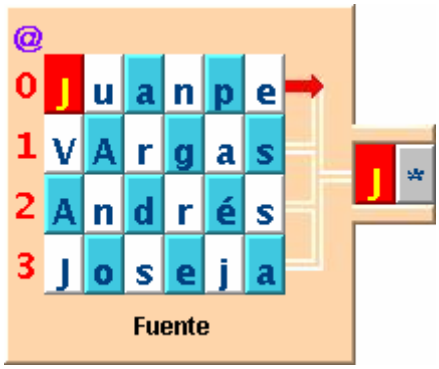


Figura 10: Fuente indicando qué slot sale al buffer

El módulo receptor tiene básicamente el mismo aspecto, y aquí se visualizan las flechas rojas en el instante en que el receptor esté recibiendo la letra. Éstas flechas se pueden ver en las líneas que van desde el buffer de entrada hasta el principio del canal que está recibiendo.

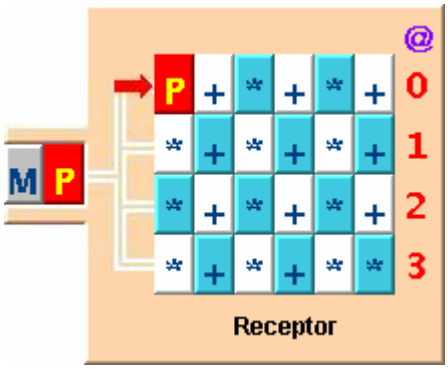


Figura 11: Módulo receptor

En la línea de transmisión se encuentran los buffers de salida de la fuente y de entrada del receptor, y con 3 visualizaciones en distintos lugares de la letra que se encuentra en el enlace. En el buffer de salida, los datos llegan a él en paralelo, es decir, primero un dato de un canal, luego de otro, etc., y salen de él para viajar como una trama en serie por la línea. A este proceso se le llama PISO (*Parallel In, Serial Out*), y cuando los datos llegan al receptor, se produce lo contrario, una conversión SIPO (*Serial In, Parallel Out*).

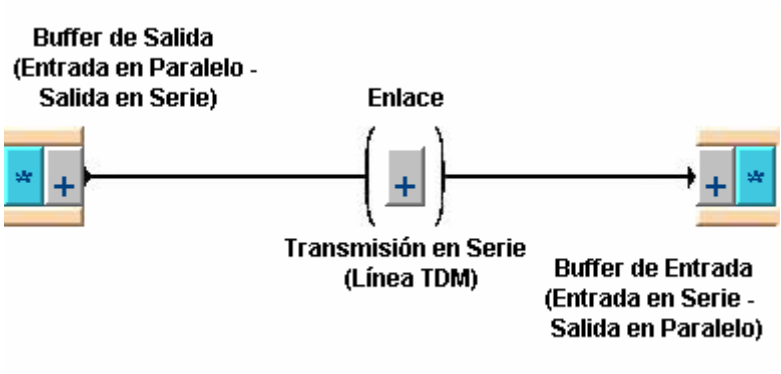


Figura 12: Línea de transmisión

Estado de la simulación

Estos indicadores muestran la trama que en ese instante se está transmitiendo desde la fuente, a qué canales corresponden las letras que se encuentran en el buffer de entrada, línea de transmisión y línea de salida, y finalmente, en qué ciclo se encuentra la transmisión de un slot temporal, es decir, 0 – Salida, 1 – Entrada, y 2 – Espera.

Figura 13: Cuadro de estado de la simulación

Como puede observarse, primero se visualiza la trama que en ese instante se está transmitiendo, que se sustituirá por la siguiente cuando empiece a transmitirse ésta, no cuando haya terminado la transmisión de la anterior. En slots, se observan los canales a los que en ese momento pertenecen las letras que ocupan los buffers de salida y entrada, y el enlace. Finalmente, vemos en cuál de los 3 ciclos se encuentra la transmisión en un slot de tiempo.

Botones de control

Con estos botones, el usuario interactuará con el simulador. Cuando estén deshabilitados, querrá decir que no tiene sentido pulsar el botón en cuestión, por ejemplo, al iniciar la aplicación todos los botones de marcha atrás están deshabilitados, incluido el de reinicio, porque no tiene sentido ir hacia atrás cuando todavía ni se ha empezado. Lo mismo ocurre con la marcha adelante y el botón fin de simulación, cuando ésta llega al final.

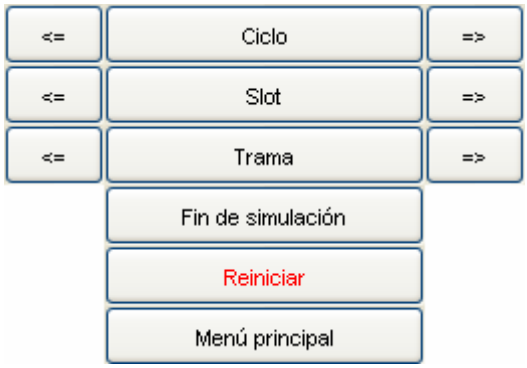


Figura 14: Cuadro de botones de control

El botón **ciclo adelante** (=>) transmite 1/3 de slot, es decir, hace, o un ciclo de salida, o un ciclo de entrada, o un ciclo de espera. Si el ciclo es de salida, la letra que se vaya a coger de la fuente se ilumina, y al mismo tiempo pasa a ocupar el buffer de salida, la letra que estaba en el buffer de salida pasa a transmitirse por la línea TDM, y la que estaba en la línea TDM, pasa al buffer de entrada del receptor. Si el ciclo es de entrada, la letra que se encuentra en el buffer de entrada, sigue ahí pero además se posiciona en su lugar correspondiente en el receptor, iluminándose dicho lugar, y si el ciclo es de espera, no se hace nada.

El botón **slot adelante** transmite un slot completo, es decir, 3 ciclos. Si la aplicación se encuentra en un ciclo de salida o de entrada porque previamente se ha estado utilizando el botón de ciclo adelante, ésta no adelanta 3 ciclos, sino que adelanta 2 en el caso de que esté en 0 – Salida o 1 ciclo en el caso de que esté en 1 – Entrada, esto se hace para dar por terminada la transmisión de ese slot para en las siguientes pulsaciones avanzar 3 ciclos con total normalidad, siempre empezando y finalizando en el ciclo de espera. Esto no sucede así con el botón **slot atrás**, ya que la finalidad de dicho botón no es ya la correcta simulación hacia atrás, sino la rectificación de un paso dado, o para volver a un estado concreto.

El botón **trama adelante** transmite una trama entera de 4 letras (en esta opción de 4 canales, porque en la de 8 canales serían 8 letras, en 16 canales 16 letras, etc.), lo que supone unos 12 ciclos. Aquí pasa lo mismo que con el botón slot adelante, se esté en el estado que se esté y en el ciclo que se esté, al pulsar trama adelante se llegará al punto en el que deja de enviarse la trama que se está transmitiendo, la cual se podrá apreciar en el estado de la simulación, y así acabar también en el ciclo de espera, para luego reanudar la marcha de forma normal, es decir, de 4 en 4 slots. Esto tampoco es aplicable al botón **trama atrás** por lo dicho en el párrafo anterior.

El botón **Fin de Simulación** transmite los 4 canales enteros, y se puede contemplar en el receptor cómo están ya todos situados en sus correspondientes direcciones. Es como si recorriéramos 78 ciclos, 26 slots, o 6 tramas (son 6 las que se

transmiten totalmente, aunque en realidad la simulación termina con la trama número 7 transmitiéndose).

El botón **Reiniciar** devuelve todo a su estado inicial.

El botón **Menú Principal** nos permite acceder a la ventana del menú que aparece al iniciarse el programa.

5.3. Opción B1

Esta opción conisiste en 8 canales a transmitir, pero la diferencia con el TDM simple radica en que cada canal tiene un destino aleatorio predefinido que no tiene por qué ser su misma posición en el receptor. Se hace necesaria así la presencia de un conmutador o Switch, que se encargue de almacenar las letras hasta que les llegue su turno de salir del switch, mirando la memoria de control.

A continuación se describirán las partes de un TDM con una etapa T:

En la fuente se pueden distinguir 8 canales, como en la opción del TDM de 8 canales simple, pero cada uno tiene su propio destino, cosa que también ocurría en el modo simple, pero de forma que cada canal tenía el destino en el receptor en la misma posición que ocupaba en la fuente, aquí también puede darse ese caso, pero lo más usual es que vaya a otra posición distinta:

Figura 15: Módulo fuente de 8 canales con sus destinos

El receptor tiene 8 canales, cada uno con 8 posiciones para guardar voz, en vez de 6, esto se debe a que hay que guardar los caracteres que ya había en la memoria de voz justo al principio de la simulación, representados mediante “*”, y también a la transmisión del carácter “+”, que se produce cuando en la fuente ya no

hay información útil que transmitir, y se transmite “+” de relleno, para terminar la simulación.

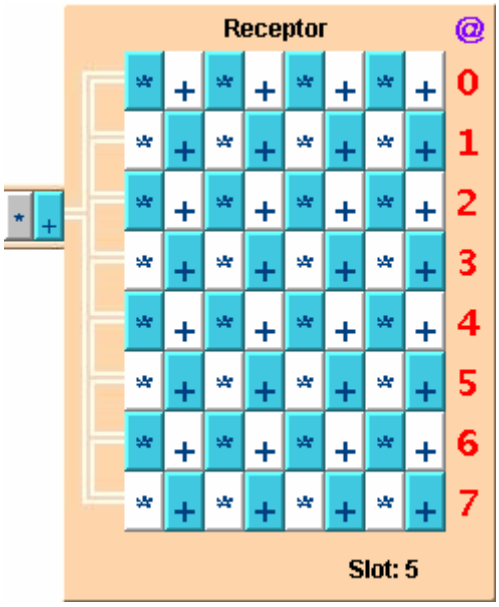


Figura 16: Módulo receptor de 8 canales y con espacio para 8 letras en cada uno

Tanto en la fuente como en el receptor, hay dos etiquetas que indican cuál es el canal que está transmitiéndose (caso de la fuente), y qué canal está recibiendo (caso del receptor). Se puede saber el canal porque, en realidad este registro indica qué slot de la trama que se está transmitiendo se está enviando en ese momento, y si es el slot 5, este slot pertenece al canal 5.

La etapa T, es decir, el switch, empieza en el buffer de entrada, donde se realiza una conversión SIPO (*Serial In, Parallel Out*), donde las letras entran en serie y se depositan en paralelo en la memoria de voz. Hay un panel donde se indica el número de trama **que se está despositando en la memoria de voz**, el slot de dicha trama que en ese momento se está almacenando, y el tipo de ciclo que se está produciendo, donde el 0 representa “Lectura”, el 1 “Escritura” y el 2 “Espera”.

En la parte inferior se encuentran las direcciones RAM, que se seguirán secuencialmente según se vaya iterando, y son fijas. Encima de las direcciones RAM se halla la memoria de control, cuyo contenido queda fijado por los destinos a los que irán las letras de los distintos canales de la fuente. Por ejemplo, si el canal **1** de la fuente tiene como destino el canal **7** del receptor, la memoria de control que está justo encima de la dirección RAM número **7** del switch, tendrá como contenido un **1**. Arriba de la memoria de control está la memoria de voz, donde se irán almacenando las letras que lleguen del buffer de entrada, hasta que les toque ir al buffer de salida y posteriormente ser reemplazadas.

Lo primero que ocurre en el switch es un ciclo de Lectura. En él, se producen cuatro sucesos:

1. Hay un desplazamiento tanto en la primera línea TDM como en la segunda.

2. Se atiende a la dirección RAM que toque, se empieza en la 0, y después de un slot, se continúa en la 1, y luego en la 2, y así sucesivamente.

3. En la memoria de control que corresponda con la dirección RAM que toque, se lee su contenido, que es un número.

4. Éste número leído de la memoria de control será la posición en la memoria de voz donde se **leerá** la letra que en este mismo ciclo, pasa al buffer de salida del switch, que hace una conversión PISO (parallel in, serial out), donde los datos llegan en paralelo desde la memoria de voz, y se transmiten en serie por la segunda línea TDM.

El ciclo de lectura gráficamente se puede apreciar por una flecha verde en cualquiera de las 8 posiciones de la memoria de voz.

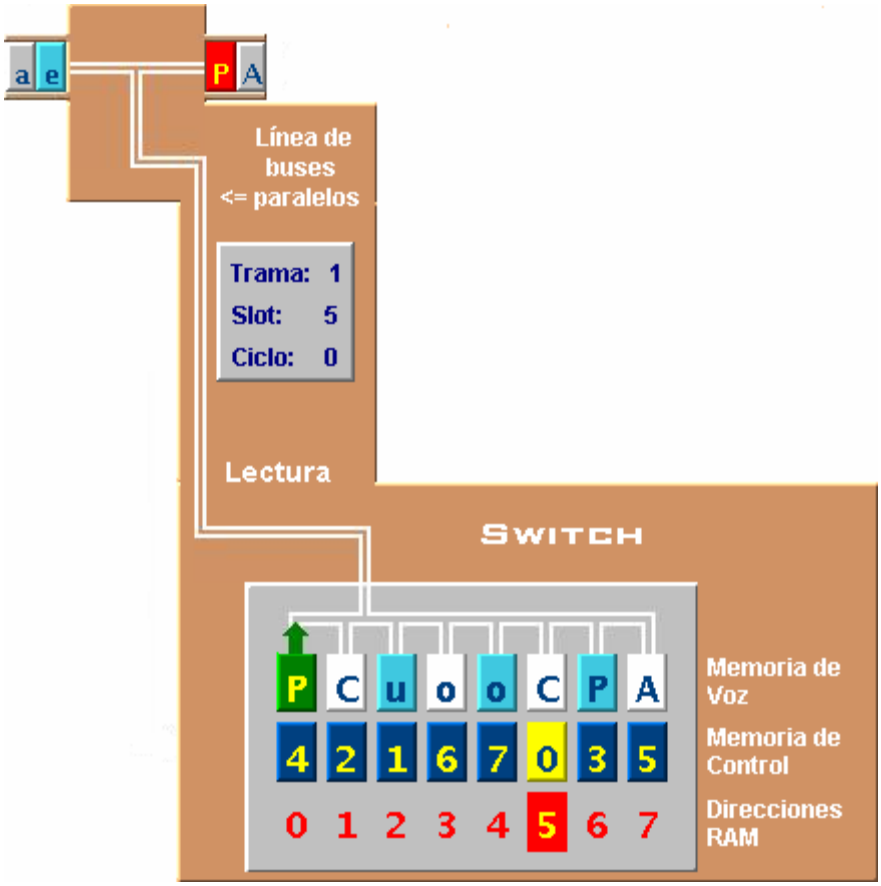


Figura 17: Etapa T o Switch

En la siguiente iteración se produce un ciclo de escritura, que tiene las siguientes fases:

1. Se atiende a la dirección RAM que toque.

2. En la memoria de voz que corresponda con la dirección RAM seleccionada, se sustituirá la letra que había antes por la que está en el buffer de entrada.

El ciclo de escritura se puede apreciar gráficamente por una flecha roja en cualquiera de las 8 posiciones de la memoria de voz.

En la siguiente iteración, en el ciclo de espera, no se hace nada, para simular el retardo.

5.4. Opción B2

Esta opción es similar a la anterior, pero con diferencias sustanciales. Hay 6 canales de datos en vez de 8, los otros 2 restantes son para sincronismo y señalización, para los que se les ha reservado las posiciones 0 y 4 respectivamente. La memoria de control no está definida inicialmente, se irá completando a lo largo de la simulación mediante la señalización.

El módulo fuente tiene el siguiente aspecto:

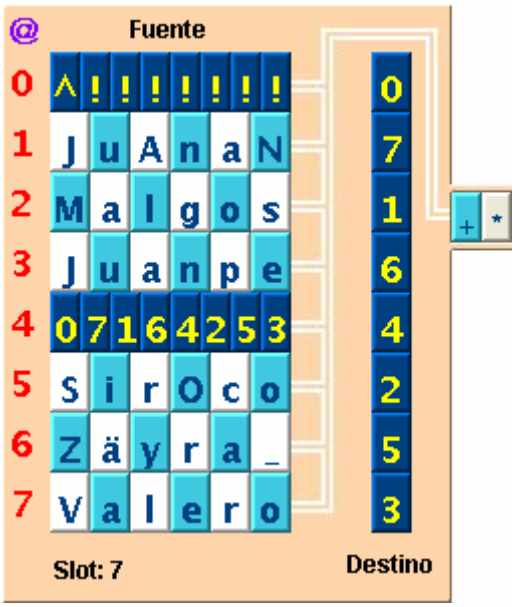


Figura 18: Módulo fuente con canales de sincronismo y señalización

Como puede verse, en el canal 0 se encuentra el canal de sincronismo, donde el sincronismo de multitrama viene representado por el carácter “^”, y el sincronismo de trama es el carácter “!”. El canal 4 es el de señalización, vemos que coincide con los destinos a los que irán a parar las letras de la fuente. Los destinos se han tomado de forma aleatoria, a excepción de los destinos 0 y 4, que están reservados para los canales de sincronismo y señalización (0 y 4 respectivamente).

La simulación empieza con fuente y receptor desincronizados, cosa que no ocurría en la anterior opción, donde receptor y fuente tenían un desfase de unos 4 slots.

El funcionamiento en la fuente es el siguiente:

Al estar fuente y receptor desincronizados, en la etiqueta “slot” puede aparecer cualquier número, empezando en un canal aleatorio. Por mucho que se itere no se mandan datos útiles hasta que el registro slot marque 0, es entonces cuando se envía

por primera vez el sincronismo de multitrama. Se va iterando y no se mandan datos, hasta que se llega al canal 4, donde se manda la señalización del canal 0, de tal forma que cuando se vuelve a la transmisión del canal 0, se manda el sincronismo de trama. Se sigue sin mandarse datos hasta que nuevamente se llega al canal 4. Tomando el ejemplo del dibujo, se manda señalización del canal 7, luego el canal de la fuente que tiene por destino el canal 7 del receptor, es decir, el canal 1, ya puede transmitir con normalidad sus datos. De esta forma, los canales de voz de la fuente tendrán permiso para transmitir en cuanto se haya mandado la señalización del canal que tienen por destino en la fuente. De no tener permiso, lo que se mandará será relleno, representado por el carácter "*". Cuando se terminan los datos útiles en los canales de la fuente, pero aún así hay que seguir transmitiendo para terminar la simulación, se manda el carácter "+", que también es relleno.

A continuación, puede observarse el módulo receptor, una vez finalizada la simulación, para que se vea con más claridad:

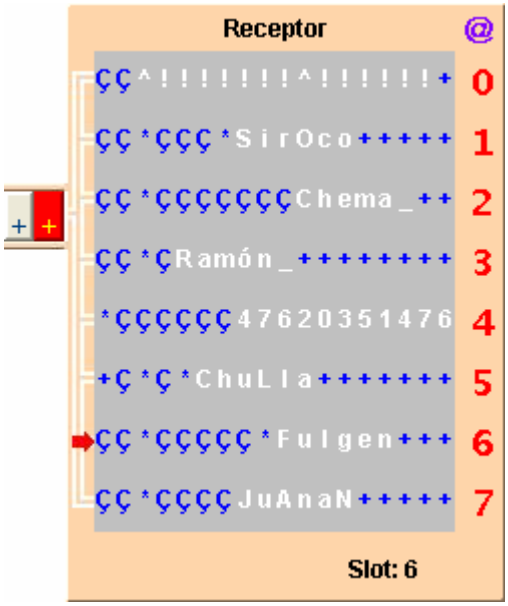


Figura 19: Módulo receptor de la opción de demostración de sincronismo y señalización

Cada canal tiene muchos más espacios que en la fuente para ver toda la progresión. Se puede ver cómo los primeros caracteres de cada canal son ruido, representados por el carácter "Ç", esto se debe a que aproximadamente durante el primer tercio de la simulación la memoria de control se está completando. A continuación tenemos ya la información útil (resaltada en blanco), y los demás caracteres representados por "+" son la consecuencia de transmitir relleno desde la fuente cuando ya no hay información útil que transmitir, pero que son necesarios para terminar correctamente la simulación.

El receptor inicialmente está desincronizado con la fuente, ya que los registros Slot de cada uno toman un valor aleatorio. El receptor se resetea, es decir, se sincroniza con la fuente cuando a éste le llega el sincronismo de multitrama, de forma que el desfase entre fuente y receptor es de 4 slots, como en la opción anterior.

En esta opción, el switch tiene el siguiente aspecto:

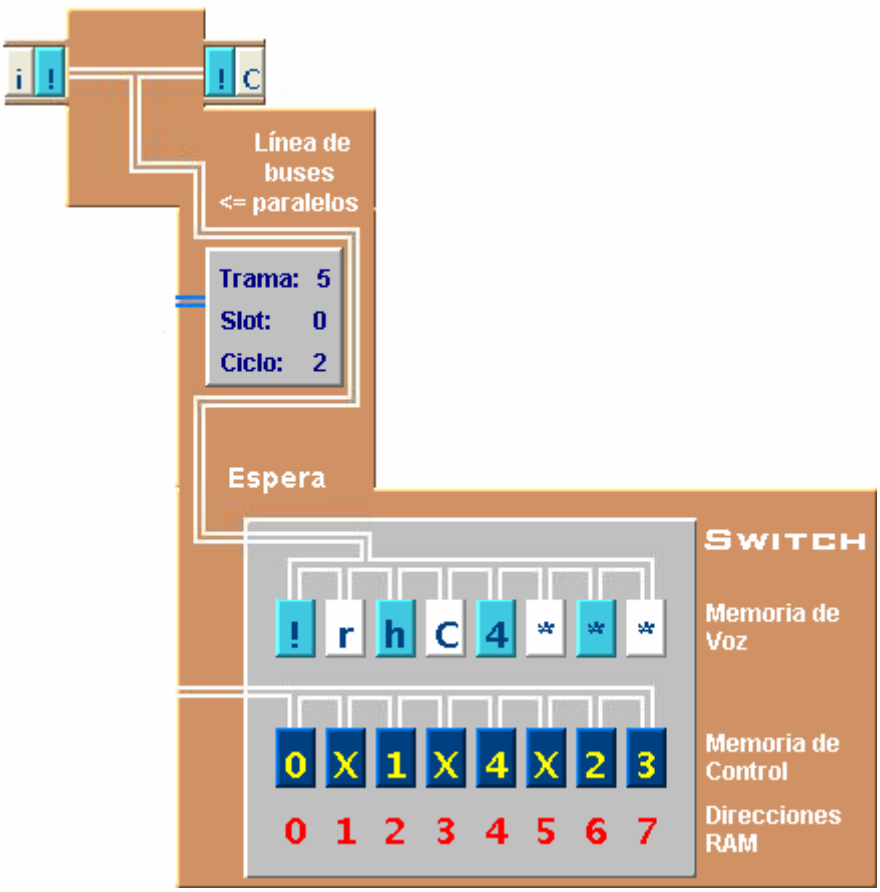


Figura 20: Etapa T con la memoria de control definiéndose

La memoria de voz y las direcciones RAM son iguales que en la Opción B1. El cambio más significativo se encuentra en la memoria de control, donde inicialmente están todos los valores a X. Eso quiere decir que la memoria de control no está definida inmediatamente después de que se generen los destinos, sino que se debe ir completando con la información que llegue del canal de señalización. Se puede observar que la memoria de control está conectada a unos buses que vienen desde el módulo de sincronismo y señalización que se verán posteriormente, módulo que también controla los registros contadores de tramas, de slots y de ciclos del switch.

En el ejemplo del dibujo, si llegara el turno de la dirección RAM 1, puede observarse que la memoria de control a la que corresponde esta dirección aún no está definida, luego a la segunda línea TDM lo que se envía es ruido.

De rellenar la memoria de control y de las labores de sincronismo se encarga el módulo de sincronismo y señalización:

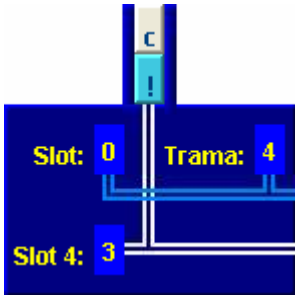


Figura 21: Módulo de sincronismo y señalización

A este módulo le llegan los datos desde la primera línea TDM (la que va del buffer de salida de la fuente al buffer de entrada del switch). En principio, los registros Slot y Trama aparecen con signo de interrogación, mientras que los registros contadores del switch van independientes. En el momento en que a este módulo le llega el sincronismo de multitrama, se produce un reset en dicho módulo y también en los registros contadores del switch, siendo los registros Slot y Trama del switch y del módulo idénticos a partir de ese momento.

El proceso de señalización empieza cuando al módulo le llega por primera vez información de señalización, es decir, un 0. Este 0 pasa a ocupar el registro Slot 4, inicialmente vacío, a la misma vez que se rellena la posición 0 de la memoria de control con el número 0. Este último número 0 no tiene nada que ver con el 0 que llega del canal de señalización, esta posición se rellena con dicho valor porque se atiende al registro Trama del módulo de sincronismo y señalización. Por ejemplo, si el siguiente dato de señalización fuera un 5, en la memoria de control que tenga la posición 5 se pondrá como contenido un 1, que es el número de trama que se está transmitiendo.

Todo esto ocurre en el ciclo de lectura, en el ciclo de escritura lo que se hace es escribir dicho dato de señalización en la memoria de voz, y la posición la decide el resgistro Slot del módulo de sincronismo y señalización, que será siempre la posición 4.

A continuación se procederá a explicar detalladamente las clases que componen el programa, los diagramas UML de cada clase van a ser de gran utilidad, con los que se podrán ver las variables utilizadas, los métodos, y las clases que interactúan con dicha clase, ya sea por herencia o composición. Se empezará explicando la clase General.

General

Ésta es la clase principal desde donde se llaman a las otras clases mediante composición, es decir, la clase General está compuesta de objetos del tipo OpcionA4, OpcionA8, OpcionA16, OpcionA32, OpcionB1 y OpcionB2. La clase General, al igual que todas las demás, es un Jframe (hereda de Jframe).

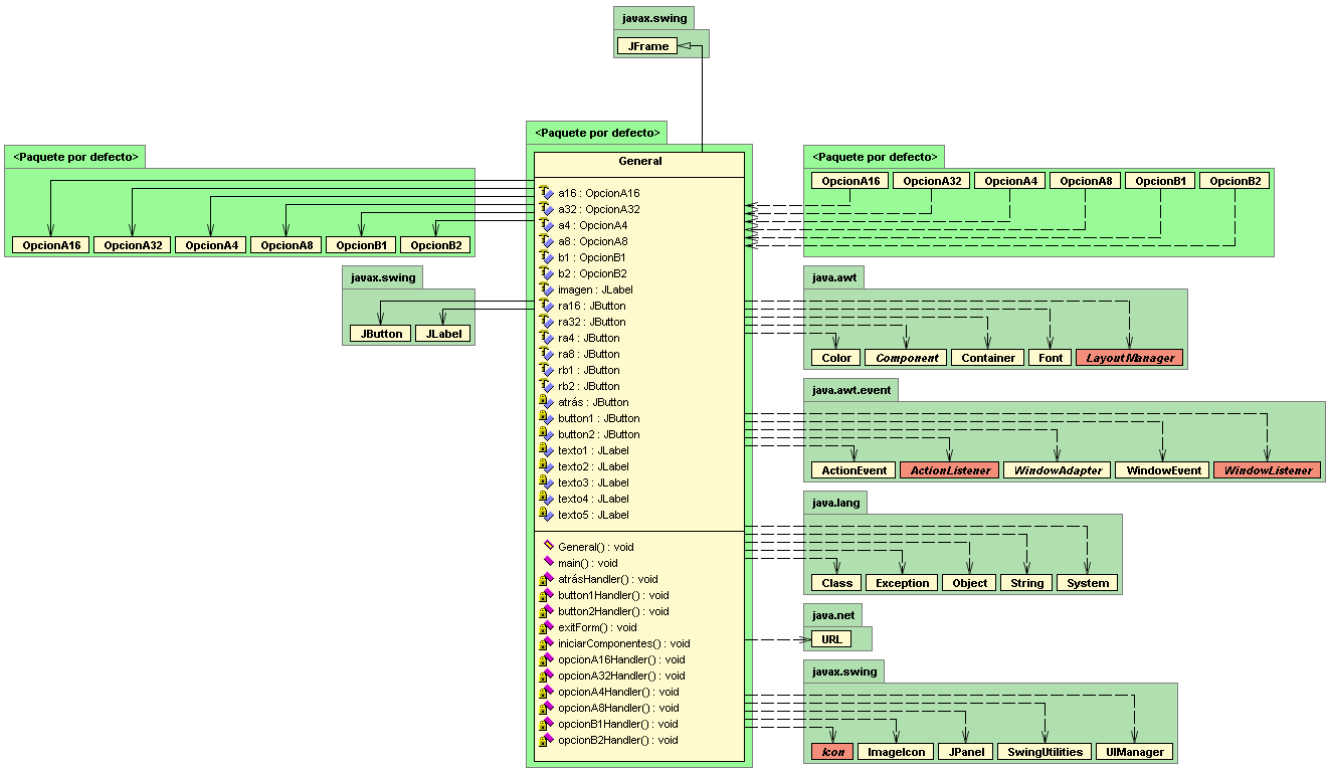


Figura 22: Diagrama UML de la clase General

Los métodos de la clase General son:

- `General():`

  Es el método constructor, que además de configurar el tamaño de la ventana, llama al método iniciarComponentes().

- `iniciarComponentes():`

  Crea todos los botones y las etiquetas de texto.

- `opcionA4Handler(java.awt.event.ActionEvent evt)`

Es el manejador que se ejecuta cuando pulsamos el botón del que escucha. En este manejador se crea el objeto del tipo OpcionA4, se muestra por pantalla, y cierra la ventana del menú principal. Los manejadores de eventos de los otros botones funcionan de manera análoga a éste con sus correspondientes objetos del resto de clases.

- `button1Handler(java.awt.event.ActionEvent evt)`

Este manejador sustituye los botones que aparecen nada más abrirse el menú principal por los del correspondiente submenú, en este caso, por las 4

opciones del TDM. En el otro manejador ocurre lo mismo con sus correspondientes opciones.

- `exitForm(java.awt.event.WindowEvent evt)`

  Este método sirve para salir de la aplicación.

- `main(String args[])`

  Éste es el método main de todo programa, aquí se crea un objeto de tipo General, y se muestra por pantalla.

OpcionA4

Es una clase que extiende de JFrame, y donde están tanto el algoritmo de TDM de 4 canales como la interfaz gráfica. Las clases OpcionA8, OpcionA16 y OpcionA32 tienen unas características análogas a la clase OpcionA4, de modo que sólo se comentará esta última.

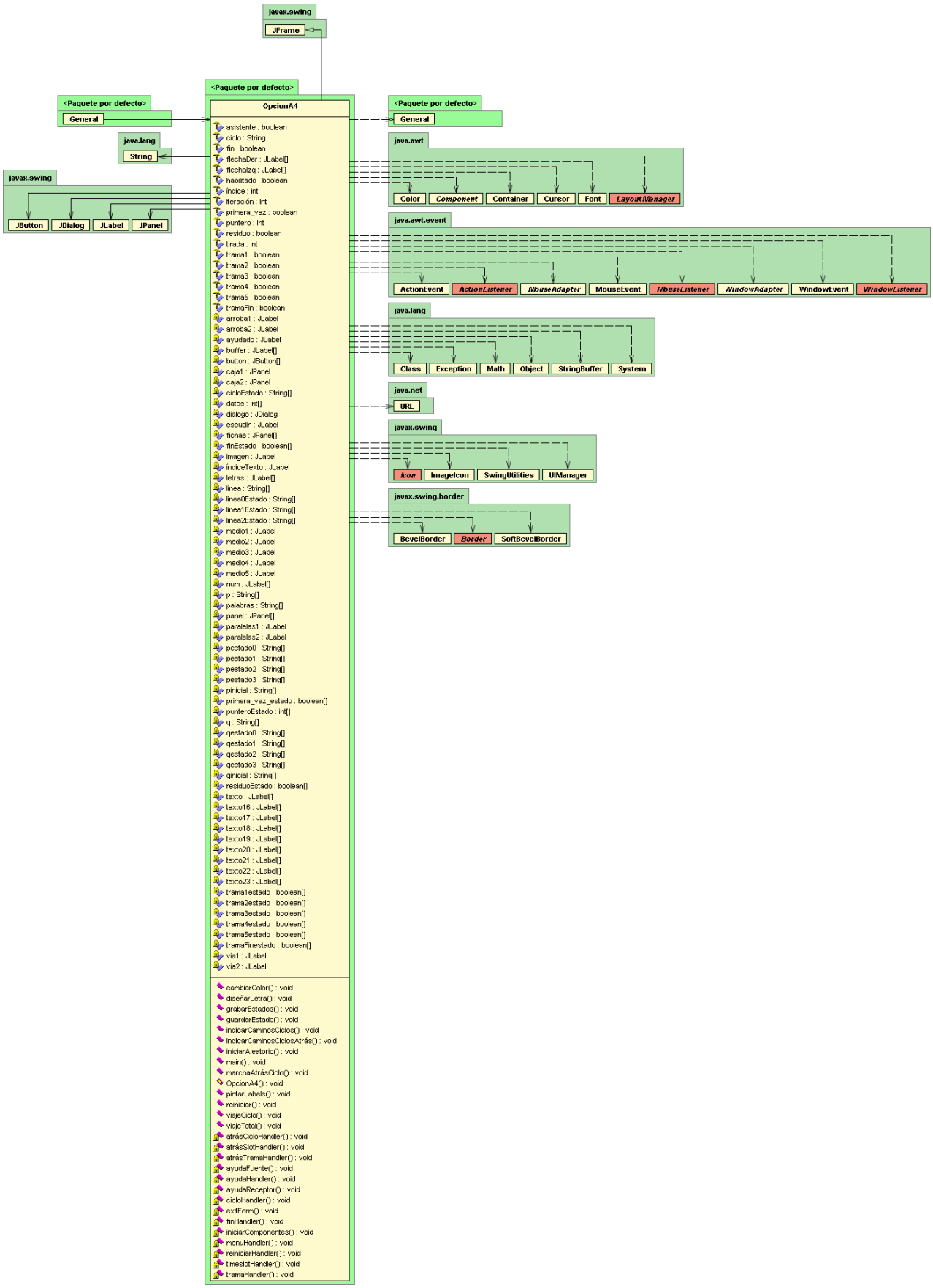


Figura 23: Diagrama UML de la clase OpcionA4

En esta clase cabe destacar los siguientes métodos:

- `public void cambiarColor(JPanel p, JLabel l)`

Ilumina una letra, es decir, pone el panel donde está situada en rojo y el color de la letra se vuelve amarillo.

- `public void diseñarLetra(JPanel p[], JLabel l[], int x, int y, int n, int f, int c)`

Se encarga de diseñar cada letra según las circunstancias, es decir, su colocación, su color, y a qué JPanel le añadimos el JLabel donde está ubicada la letra. Este método nos ahorra muchísimo código, ya que cada letra puede requerir 8 líneas o más; con este método tan sólo basta una sola línea.

- `public void guardarEstado()`

Este método guarda todos los posibles estados de todas las variables en su array correspondiente, es decir, todos los posibles estados de q[0] se guardarán en el array qestado0[], pero este método sólo lo hace una vez, para hacerlo todas las posibles iteraciones se ayudará del método grabarEstados().

- `public void grabarEstados()`

Este método se llama nada más iniciarse la aplicación, y hace un viaje completo hasta el final de la simulación, guardando los estados de todas variables en cada iteración, y reiniciando posteriormente. Primero llama al método guardarEstado(), con lo que se graban todas las variables en su estado inicial, luego llama al método viajeCiclo() para pasar al siguiente estado, y el contador de los arrays de los estados se incrementa en uno, y vuelve a hacer el proceso hasta el final de la simulación. Finalmente reinicia el sistema para que se pueda simular, pero las variables de estado no quedan reiniciadas, quedarán ahí grabadas para cuando se necesite utilizar la marcha atrás.

- `public void marchaAtrásCiclo()`

El valor actual de las variables pasará a ser el valor que tienen todas sus variables de estado en el instante anterior, es decir, en la posición contador – 1 del array (índice – 1). Posteriormente, se decrementa en uno este contador.

- `public void indicarCaminosCiclos()`

Sirve para ver de qué canales son las letras que están en el buffer de la fuente, en la línea, y en el buffer del receptor, y esto se mostrará en las etiquetas del estado de la simulación.

- `public void iniciarAleatorio()`

Aquí se generan de forma aleatoria las 4 palabras de entre las 32 posibles y se sitúan en la fuente. En java, para obtener un número aleatorio se hace uso de la función **Math.random(),** que devuelve un número aleatorio entre 0 (inclusive) y 1(exclusive), de tal forma que si queremos un número aleatorio de 0 a 32, el resultado de la función Math.random() lo multiplicamos por 32, y al resultado le hacemos un casting a entero. Por ejemplo:

```
int num = (int)(Math.random()*32);
```

- `public void reiniciar()`

Devuelve todas las variables a su estado inicial, excepto a las de estado, que seguirán con sus valores hasta que el usuario cierre la ventana o vuelva al menú principal.

- `public void viajeTotal()`

Llama al método viajeCiclo() tantas veces como falte para llegar al estado final.

- `public void pintarLabels()`

En cada iteración, las variables tienen un estado, luego necesitan ser también refrescadas en pantalla cada vez que se produce dicha iteración, de hacerlo sólo una vez, el usuario no podría ver los cambios. En cualquier acción que hagamos que suponga un cambio en las variables, se llamará al método pintarLabels(), que diseñará la ventana según el estado de la simulación.

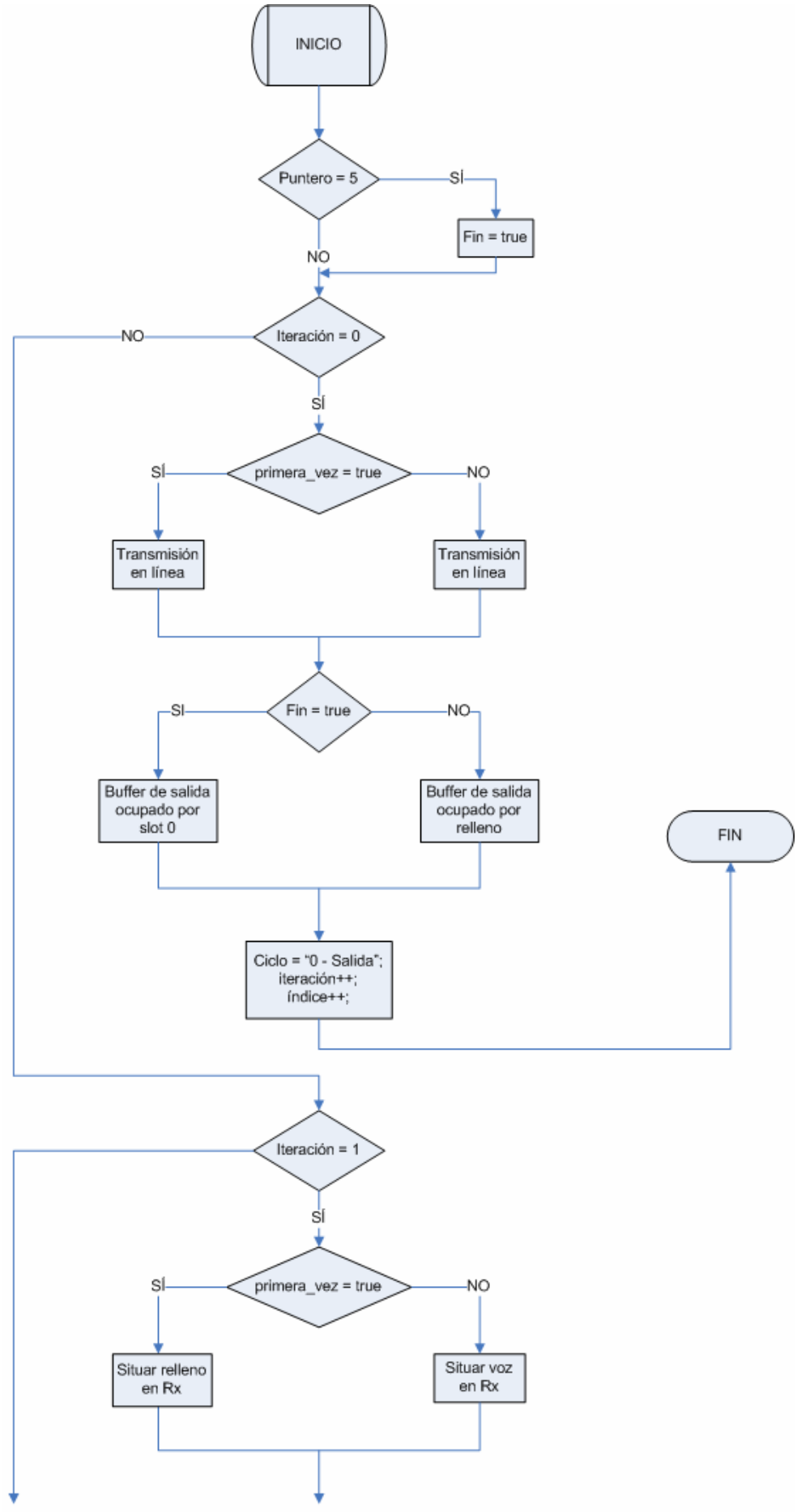
- `private void ayudaHandler(java.awt.event.ActionEvent evt)`

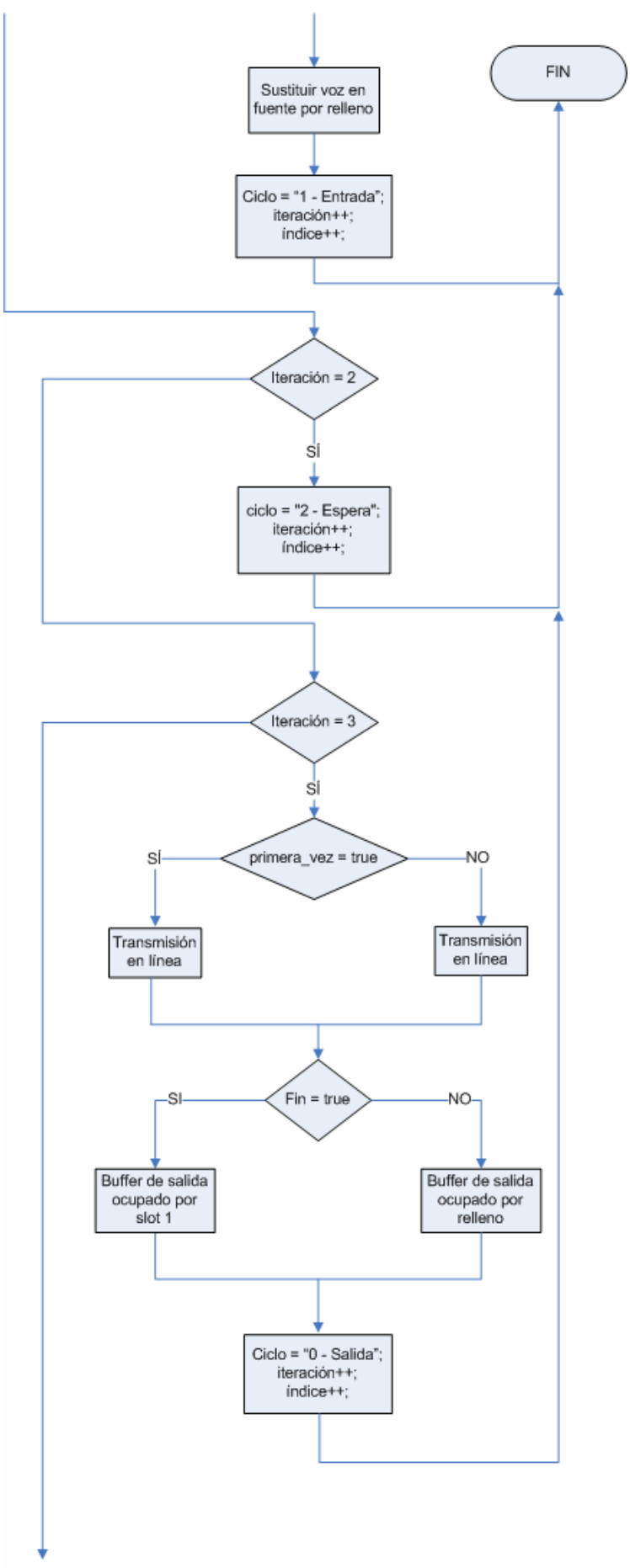
Activa la ayuda del programa poniendo a “true” la variable *asistente*, de tal forma que después de pulsar el botón de ayuda, se pinche el elemento que se pinche en el simulador, se abrirá una ventana tipo *Jdialog*, donde se mostrará al usuario para qué sirve dicho elemento.

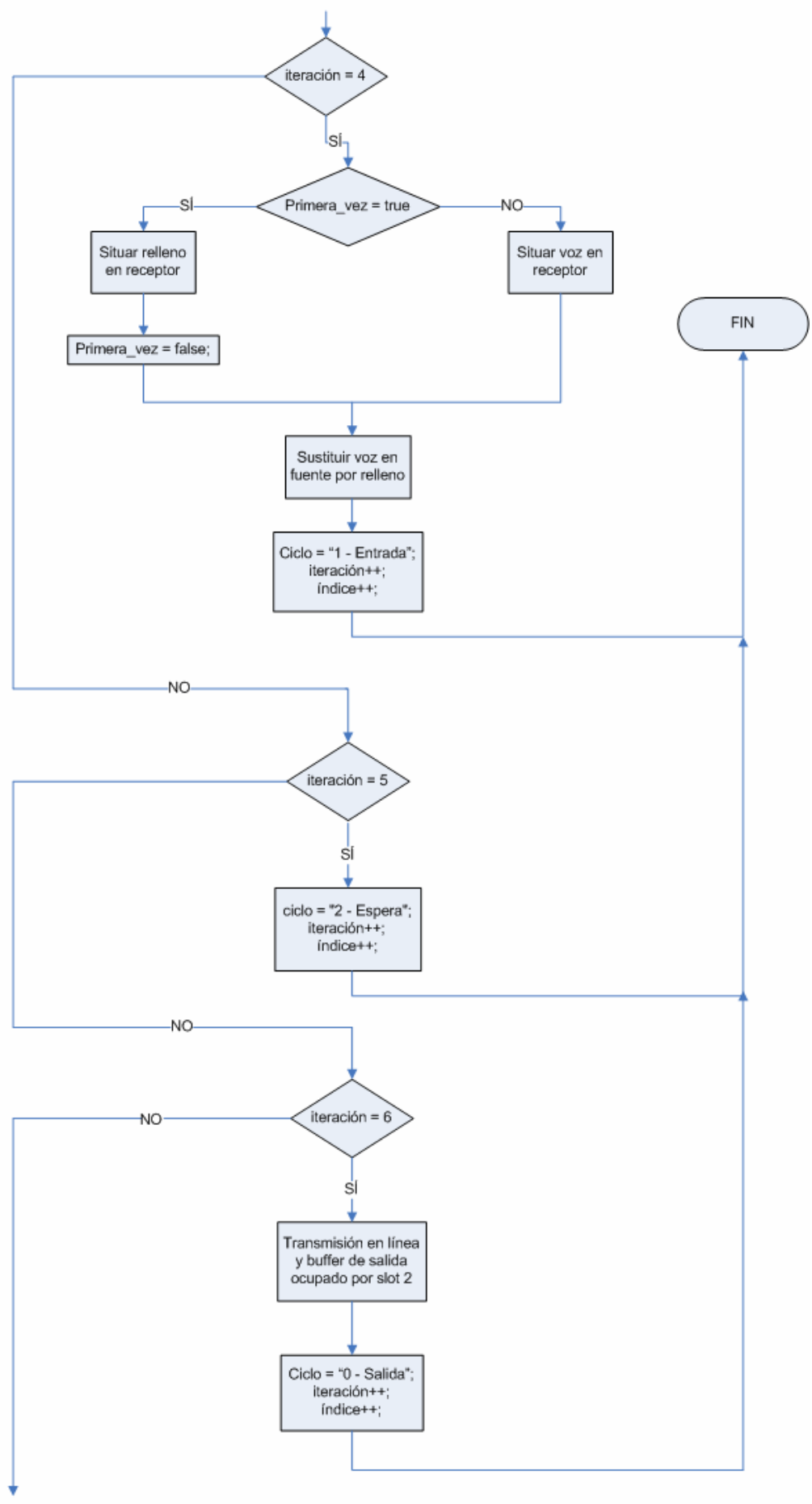
- `public void viajeCiclo(String pe, String qe)`

Es el método clave del programa, el que hace la transmisión ciclo a ciclo. El ciclo que debe realizar lo determina una variable fundamental en este programa, la variable *iteración*, que alcanza su máximo valor en 11. La variable *puntero* se encarga de seleccionar el número de la letra de cada canal que se va a transmitir, y también el que va a recibir.

A continuación, se verá el diagrama de flujo de dicho método en la OpcionA4:


INICIO
Puntero = 5
SÍ
Fin = true
NO
Iteración = 0
NO
SÍ
primera_vez = true
SÍ
NO
Transmisión en línea
Transmisión en línea
Fin = true
SI
NO
Buffer de salida ocupado por slot 0
Buffer de salida ocupado por relleno
FIN
Ciclo = "0 - Salida";
iteración++;
índice++;
Iteración = 1
SÍ
primera_vez = true
SÍ
NO
Situar relleno en Rx
Situar voz en Rx


FIN
Sustituir voz en fuente por relleno
Ciclo = "1 - Entrada";
iteración++;
índice++;
Iteración = 2
SÍ
ciclo = "2 - Espera";
iteración++;
índice++;
Iteración = 3
SÍ
SÍ
primera_vez = true
NO
Transmisión en línea
Transmisión en línea
SI
Fin = true
NO
Buffer de salida ocupado por slot 1
Buffer de salida ocupado por relleno
Ciclo = "0 - Salida";
iteración++;
índice++;


iteración = 4
SÍ
Primera_vez = true
SÍ
NO
Situar relleno en receptor
Situar voz en receptor
Primera_vez = false;
FIN
Sustituir voz en fuente por relleno
Ciclo = "1 - Entrada";
iteración++;
índice++;
NO
iteración = 5
SÍ
ciclo = "2 - Espera";
iteración++;
índice++;
NO
iteración = 6
NO
SÍ
Transmisión en línea y buffer de salida ocupado por slot 2
Ciclo = "0 - Salida";
iteración++;
índice++;

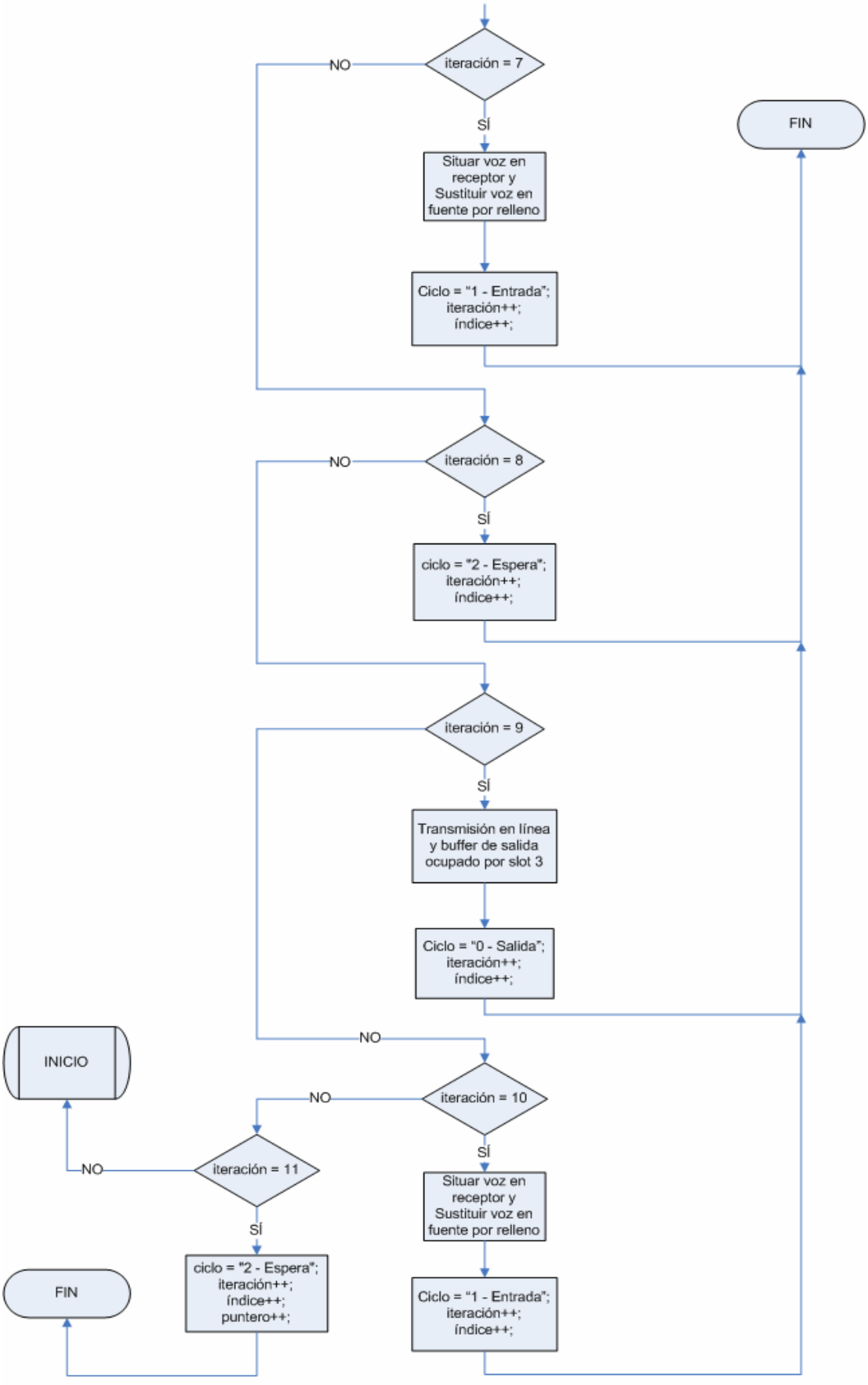


Figura 24: Diagrama de flujo del método viajeCiclo() de la OpcionA4

A continuación se expondrán los diagramas UML de las clases OpcionA8, OpcionA16, OpcionA32:

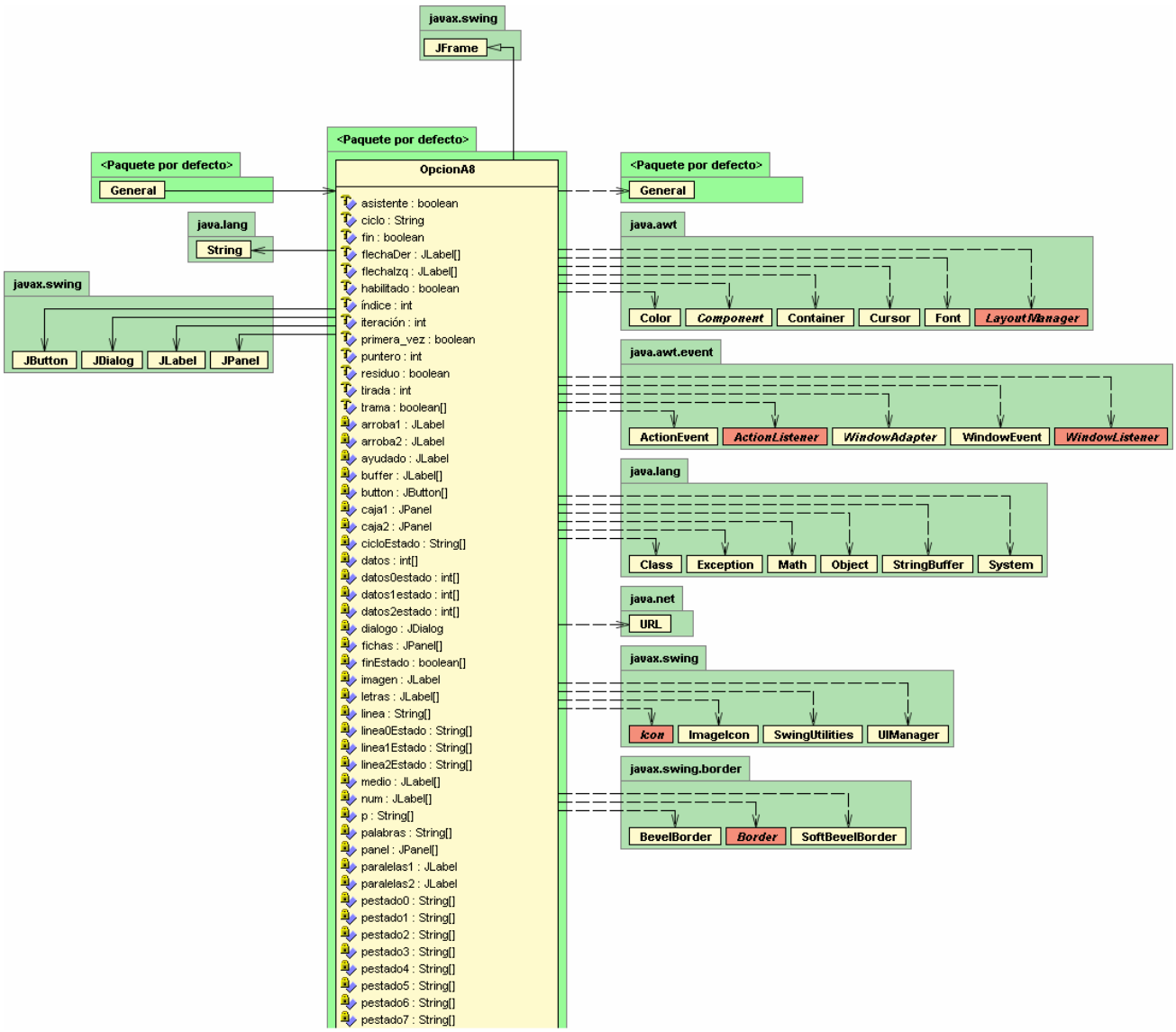


Figura 25: Primera mitad del diagrama UML de la OpcionA8

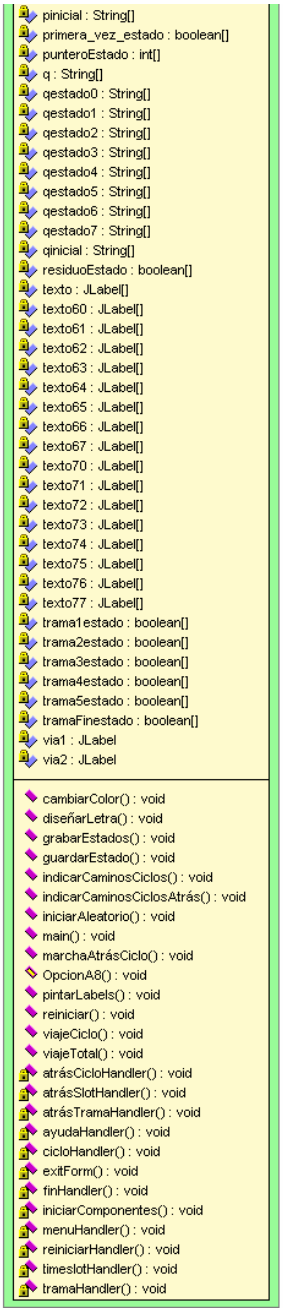


Figura 26: Segunda mitad del diagrama UML de la OpcionA8

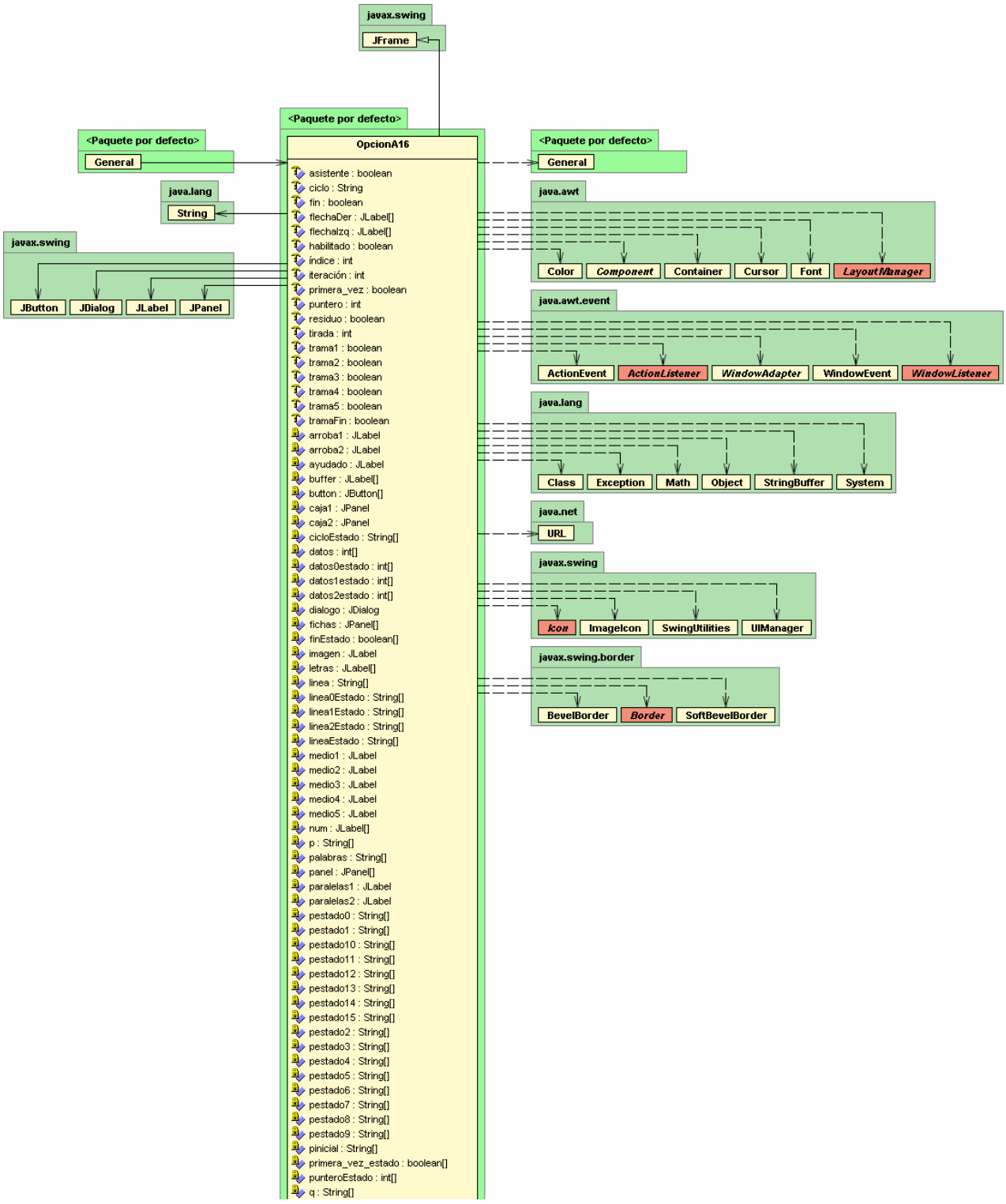


Figura 27: Primera mitad del diagrama UML de la OpcionA16

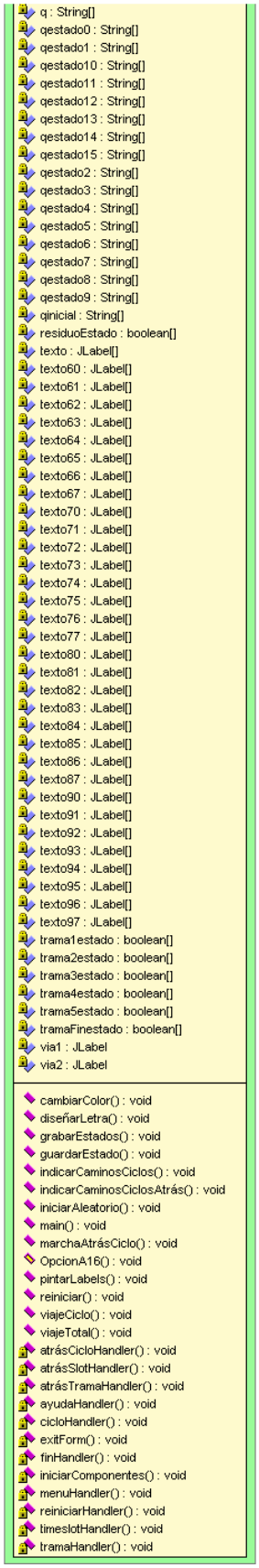


Figura 28: Segunda mitad del diagrama UML de la OpcionA16

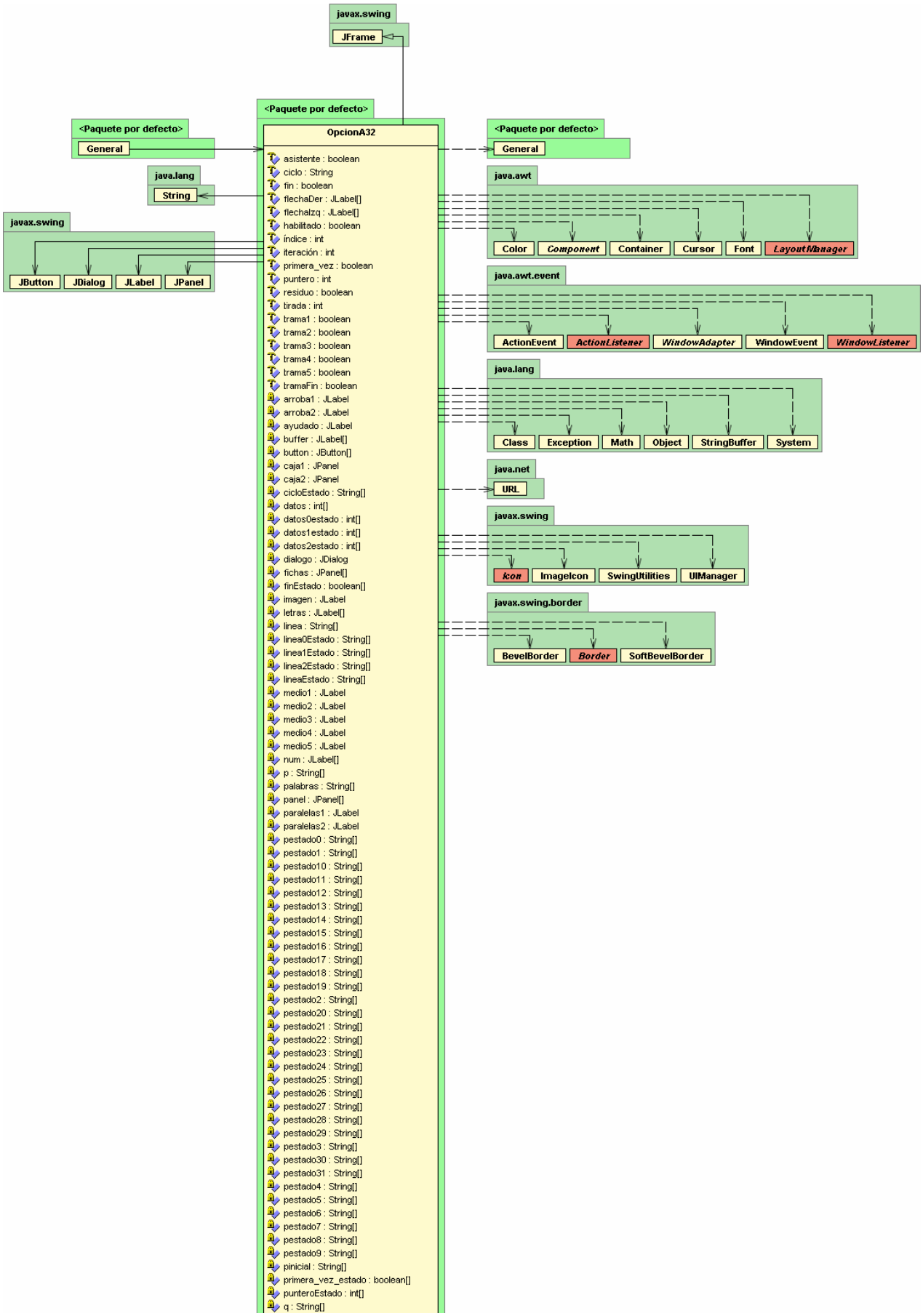


Figura 29: Primera mitad del diagrama UML de la OpcionA32

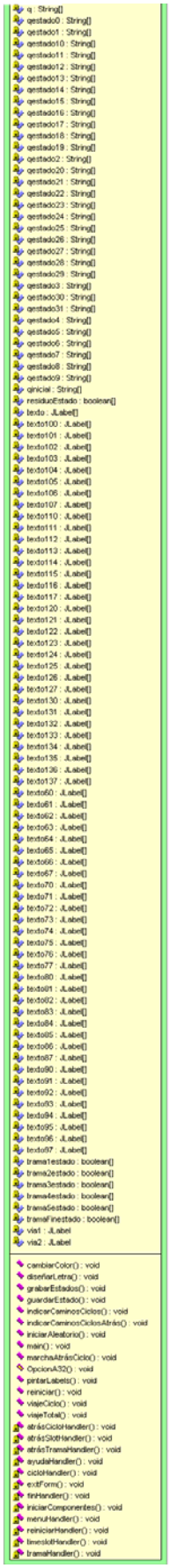


Figura 30: Segunda mitad del ddiagrama UML de la OpcionA32

OpcionB1

Es la clase que implementa la funcionalidad y la interfaz gráfica de la demostración de conmutación en el tiempo. Tiene bastantes métodos iguales que las anteriores opciones, de modo que se obviarán y se mostrarán los nuevos métodos:

- `public void contarSlots()`

  Hace la cuenta en los registros Slot de la fuente y del receptor.

- `public void diseñarLetraReceptor(Jpanel p[], Jlabel l[], int x, int y, int n, int f, int c)`

  Tiene la misma funcionalidad que el método diseñarLetra(), pero éste está adaptado a un receptor con 2 espacios más para datos.

- `public void situarControles()`

  Aquí se fija el contenido de la memoria de control según los destinos elegidos

- `public void viajeCiclo(String pe, String qe)`

  Como en la anterior opción, éste es el método fundamental donde está incluido el algoritmo de transmisión.

  Una porción de código que represente al primer ciclo de lectura sería:

```
posS = control[0];
linea[3] = speech[posS];
```

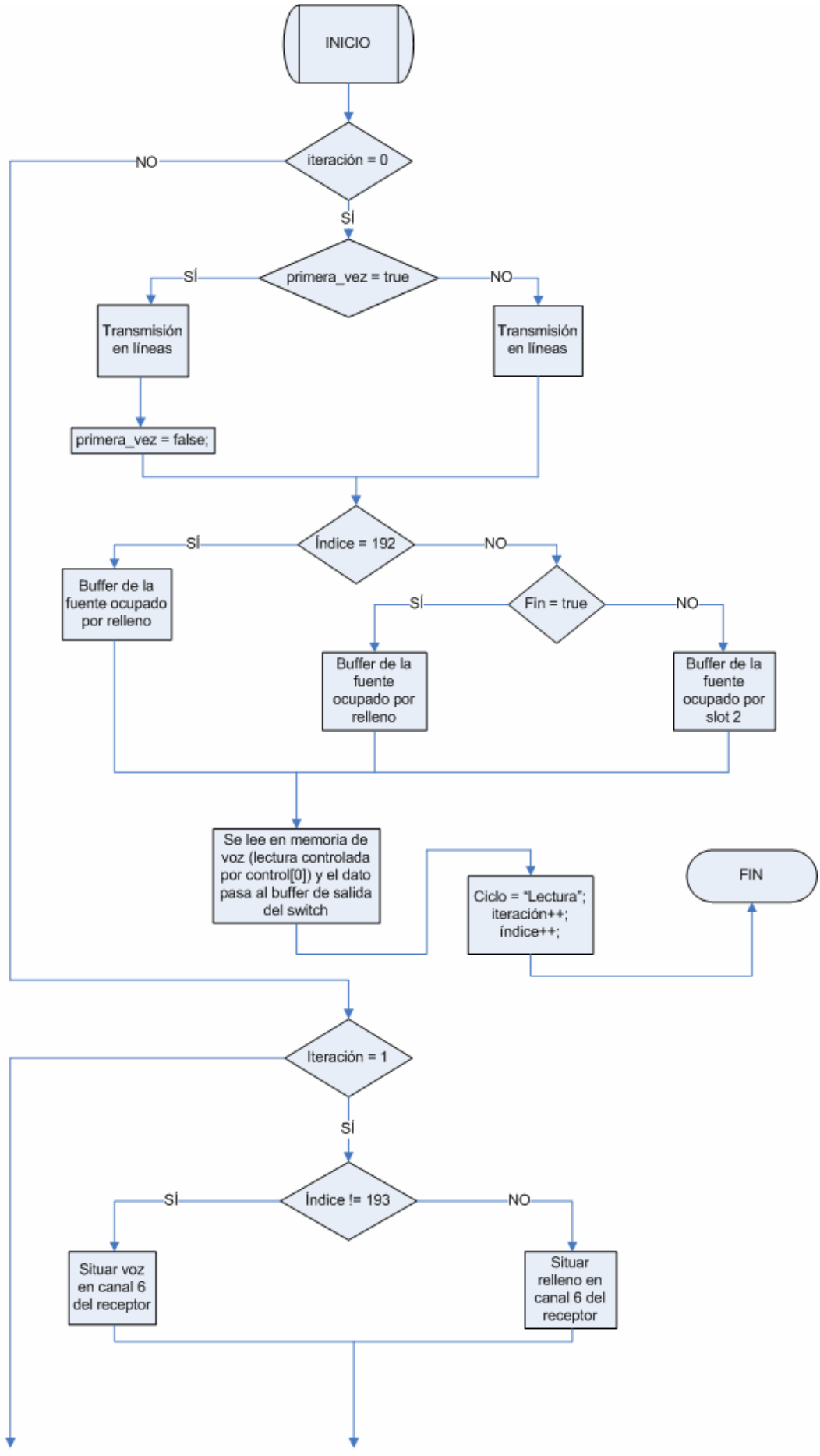
  Se puede observar cómo se guarda el valor (representado por *posS*) de la memoria de control de la posición 0, y ese mismo valor será la dirección de la memoria de voz (representada por *speech[]*), que se leerá para ocupar el buffer de salida del switch (representado por *linea[3]*).

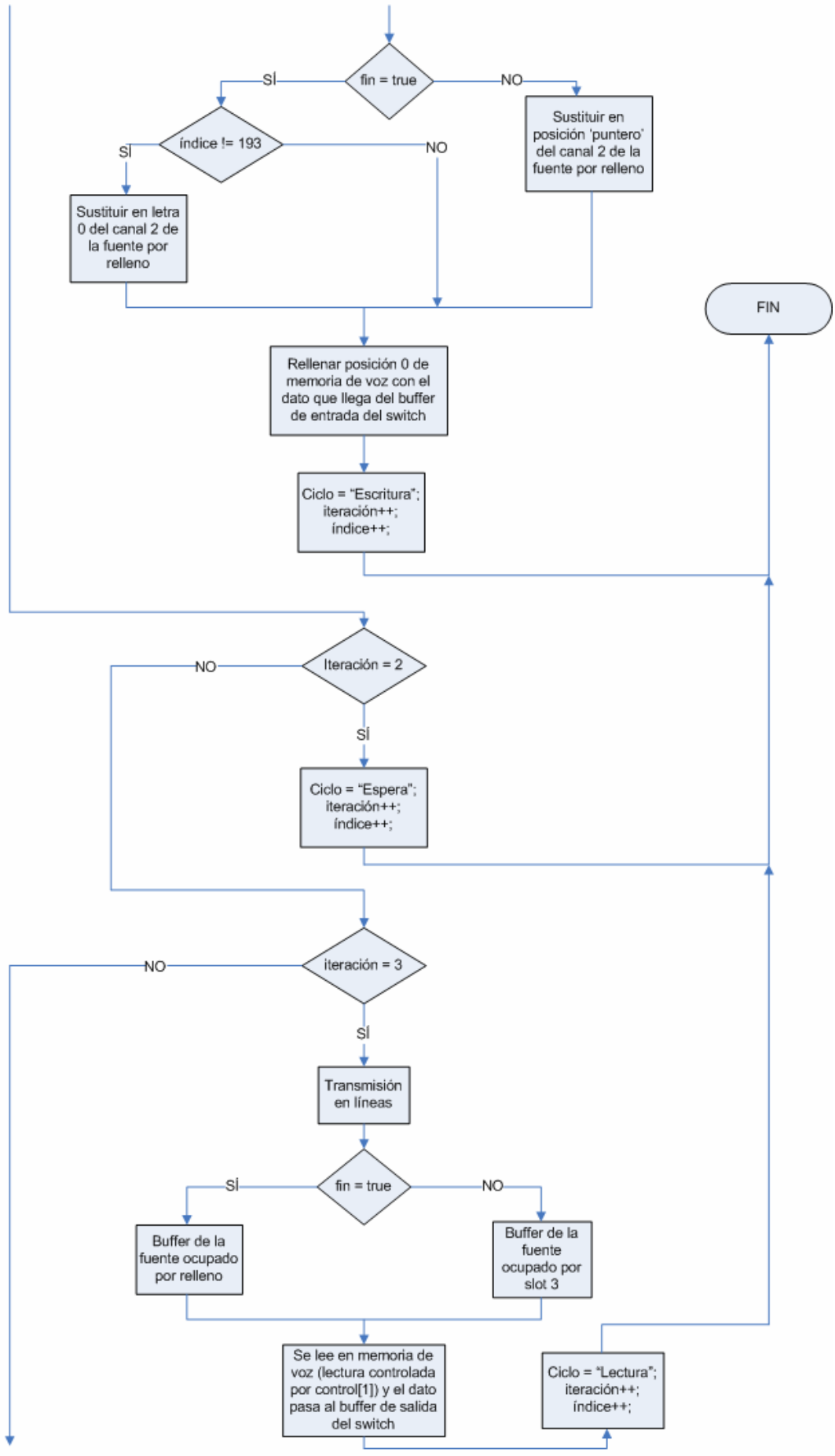
  Y la porción de código que implementa al primer ciclo de escritura es:

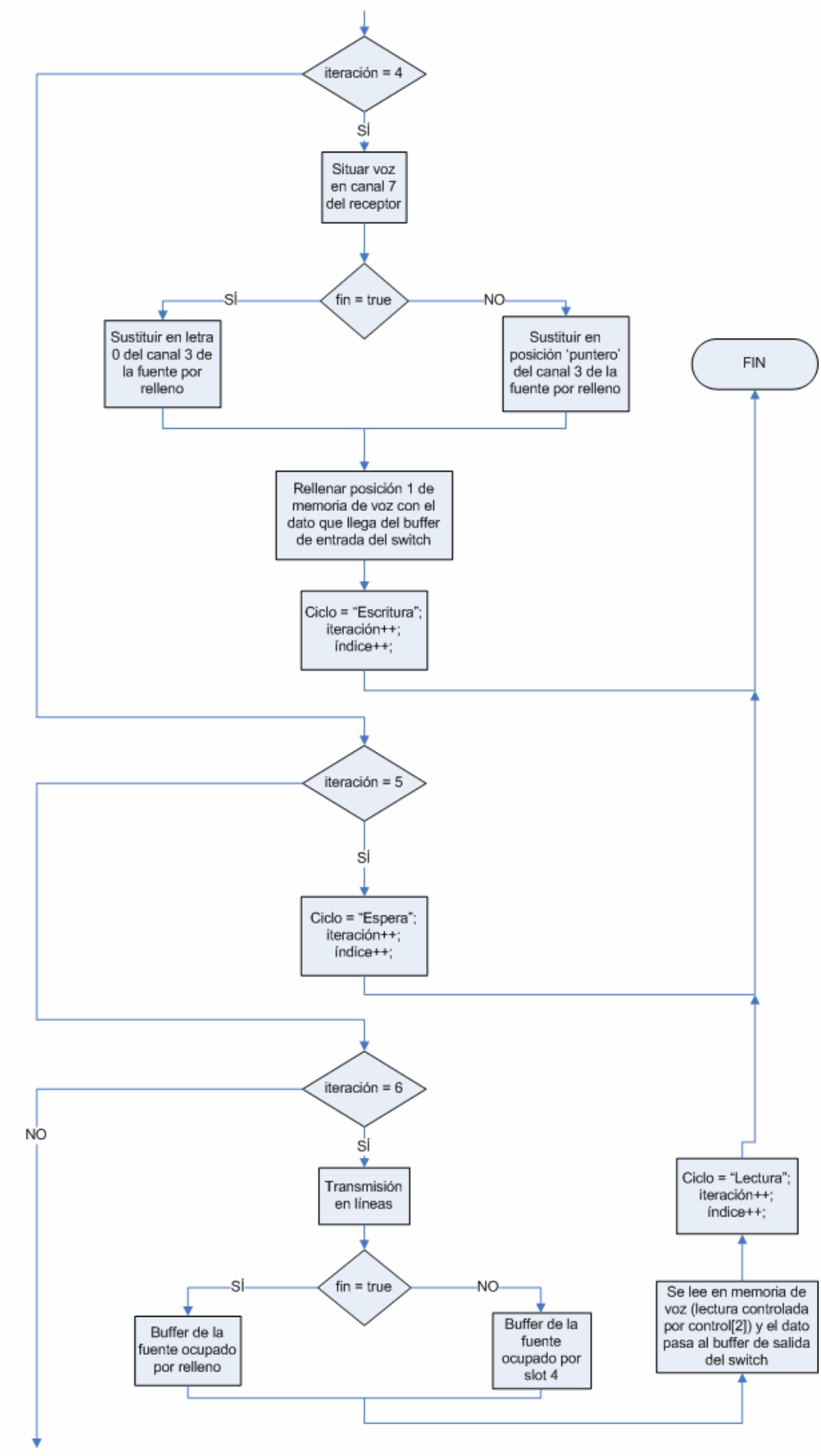
```
speech[0] = linea[2];
```

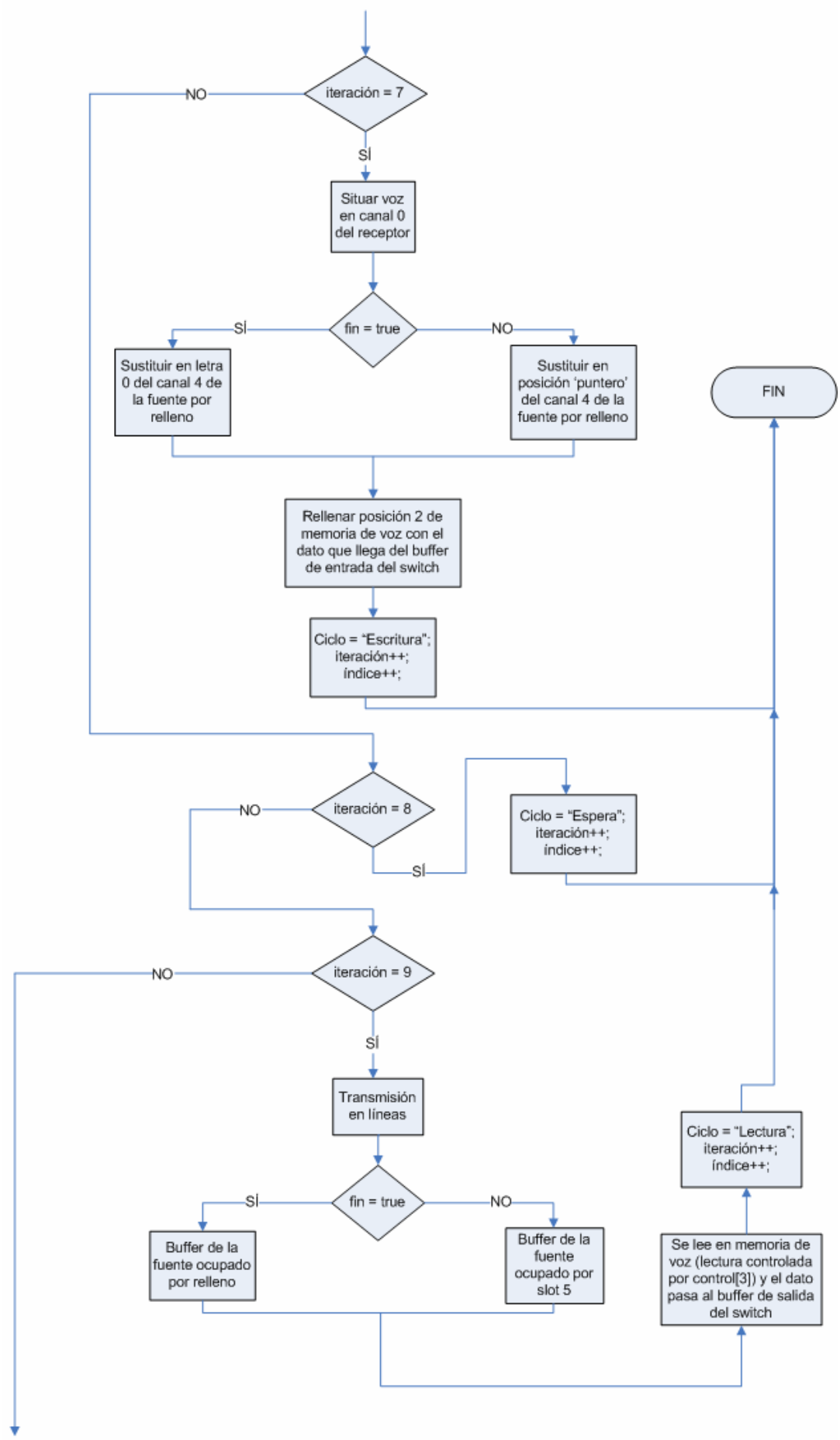
  A la memoria de voz de la posición 0 llega el dato que estaba en el buffer de entrada del switch.

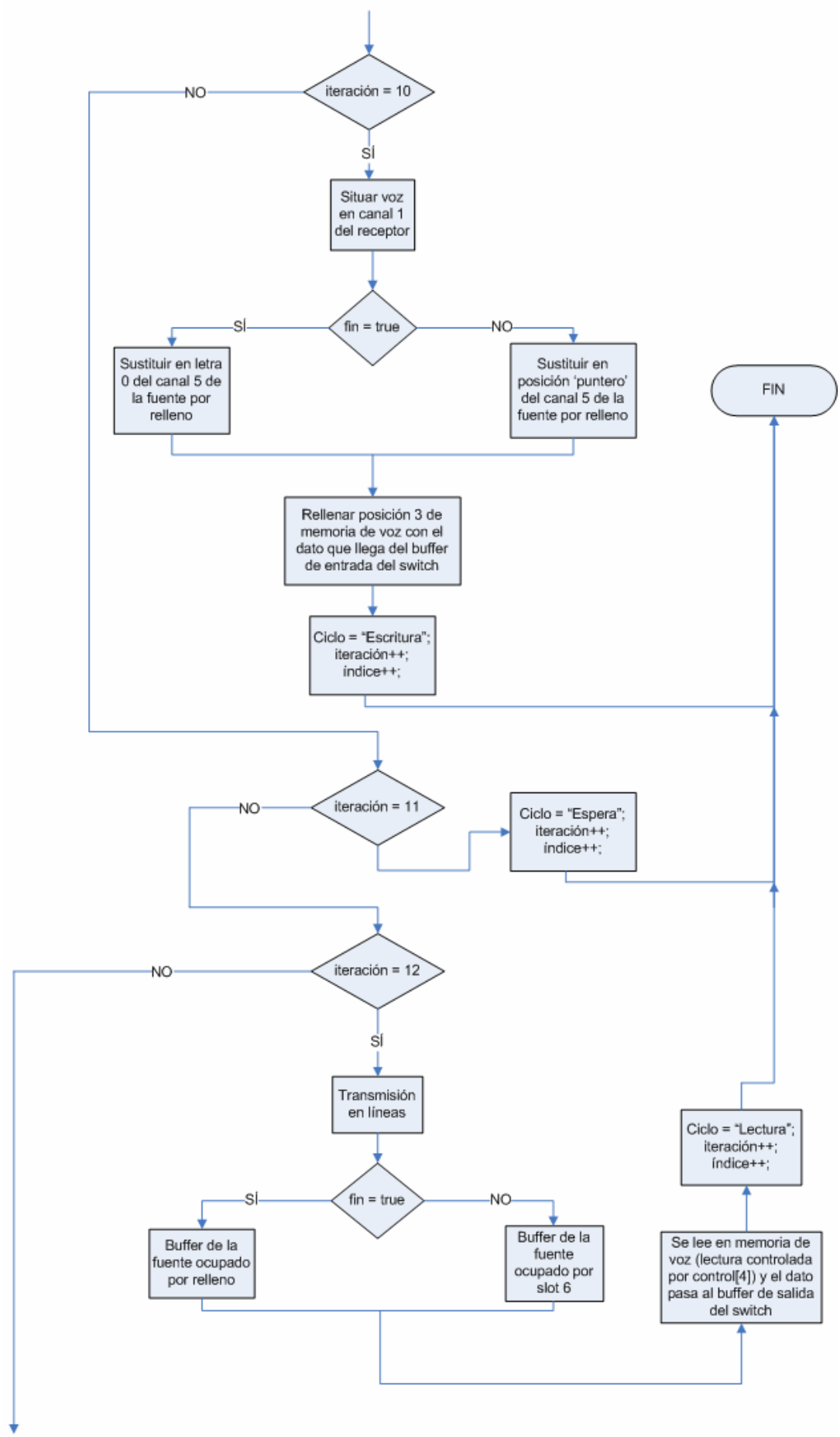
  A continuación se verá el diagrama de flujo del algoritmo de conmutación en el tiempo usando TDM:

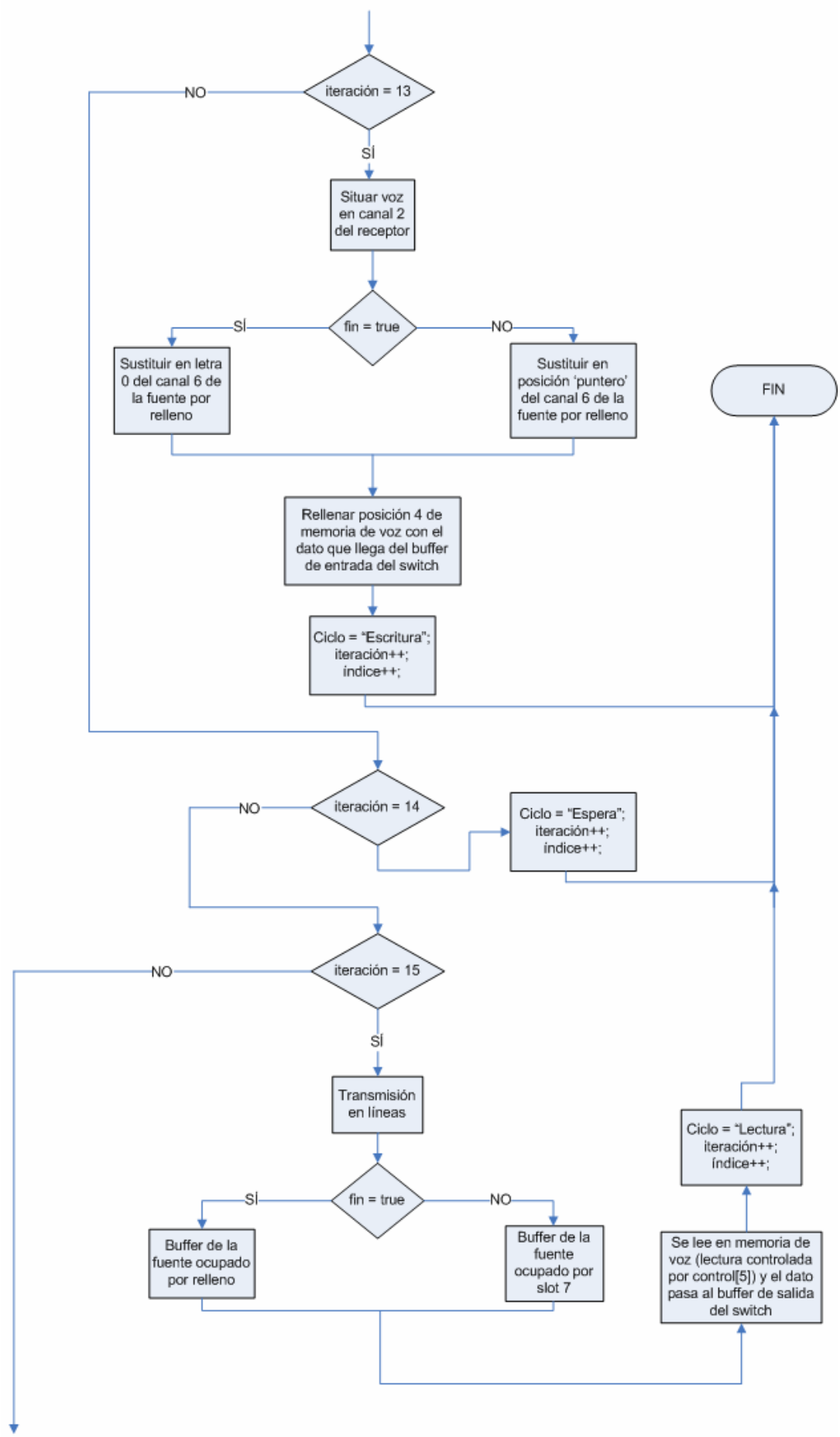

INICIO
iteración = 0
NO
SÍ
primera_vez = true
SÍ
NO
Transmisión en líneas
Transmisión en líneas
primera_vez = false;
Índice = 192
SÍ
NO
Buffer de la fuente ocupado por relleno
Fin = true
SÍ
NO
Buffer de la fuente ocupado por relleno
Buffer de la fuente ocupado por slot 2
Se lee en memoria de voz (lectura controlada por control[0]) y el dato pasa al buffer de salida del switch
Ciclo = "Lectura";
iteración++;
índice++;
FIN
Iteración = 1
SÍ
Índice != 193
SÍ
NO
Situar voz en canal 6 del receptor
Situar relleno en canal 6 del receptor

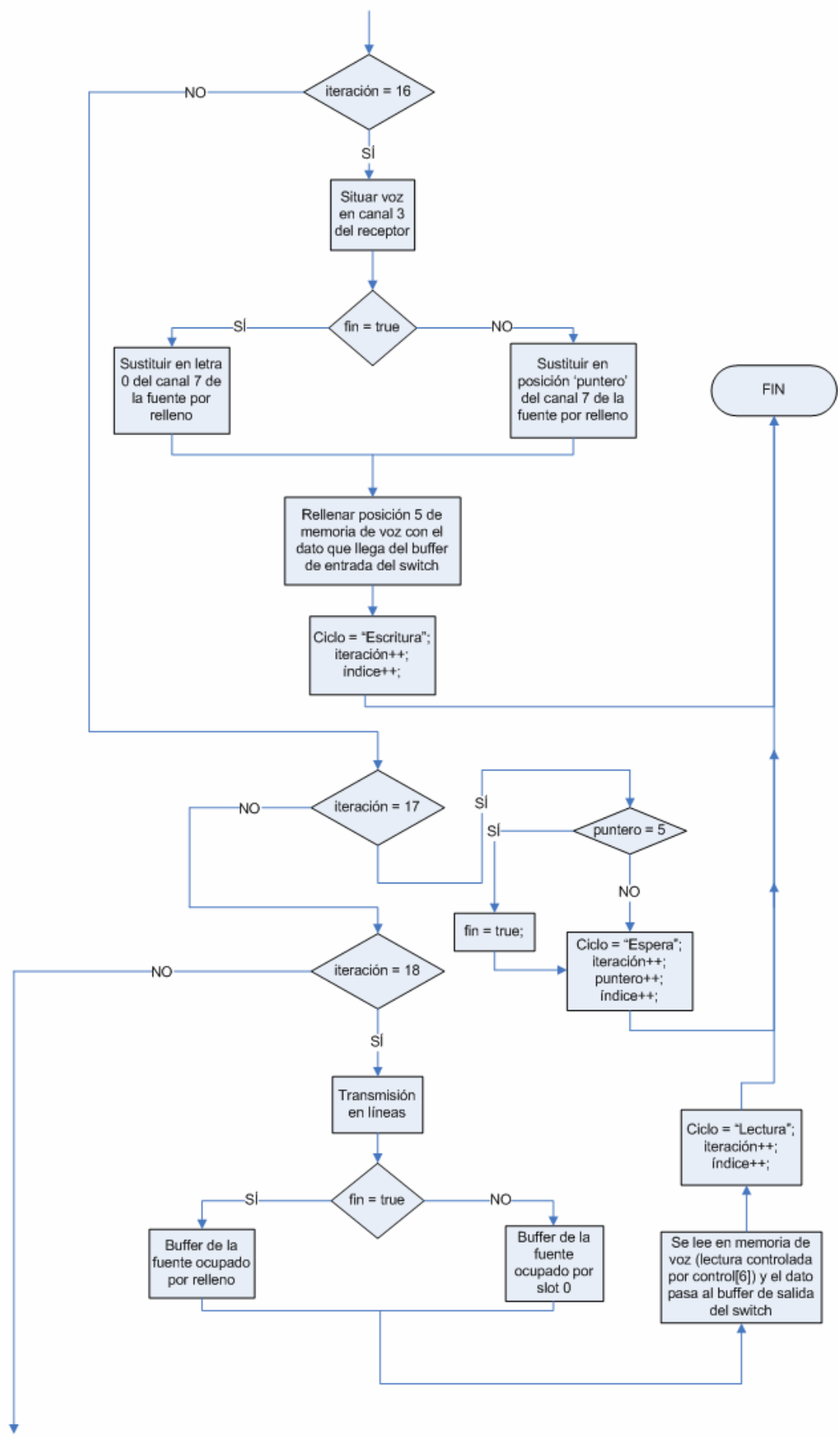

fin = true
SÍ
NO
índice != 193
SÍ
NO
Sustituir en posición 'puntero' del canal 2 de la fuente por relleno
Sustituir en letra 0 del canal 2 de la fuente por relleno
FIN
Rellenar posición 0 de memoria de voz con el dato que llega del buffer de entrada del switch
Ciclo = "Escritura"; iteración++; índice++;
Iteración = 2
NO
SÍ
Ciclo = "Espera"; iteración++; índice++;
iteración = 3
NO
SÍ
Transmisión en líneas
fin = true
SÍ
NO
Buffer de la fuente ocupado por relleno
Buffer de la fuente ocupado por slot 3
Se lee en memoria de voz (lectura controlada por control[1]) y el dato pasa al buffer de salida del switch
Ciclo = "Lectura"; iteración++; índice++;

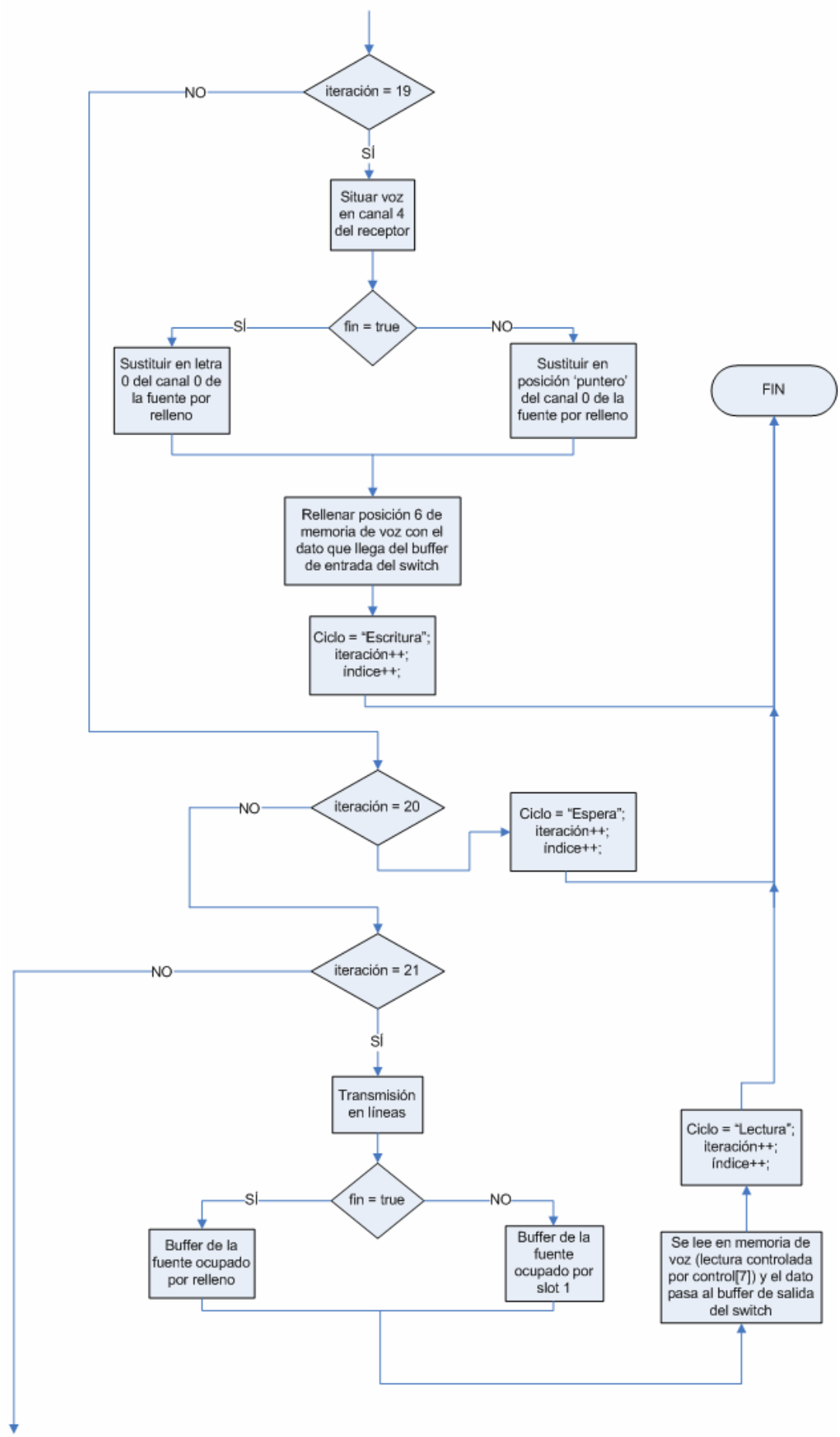

iteración = 4
SÍ
Situar voz en canal 7 del receptor
fin = true
SÍ
NO
Sustituir en letra 0 del canal 3 de la fuente por relleno
Sustituir en posición 'puntero' del canal 3 de la fuente por relleno
FIN
Rellenar posición 1 de memoria de voz con el dato que llega del buffer de entrada del switch
Ciclo = "Escritura"; iteración++; índice++;
iteración = 5
SÍ
Ciclo = "Espera"; iteración++; índice++;
iteración = 6
NO
SÍ
Transmisión en líneas
Ciclo = "Lectura"; iteración++; índice++;
fin = true
SÍ
NO
Buffer de la fuente ocupado por relleno
Buffer de la fuente ocupado por slot 4
Se lee en memoria de voz (lectura controlada por control[2]) y el dato pasa al buffer de salida del switch


iteración = 7
NO
SÍ
Situar voz en canal 0 del receptor
fin = true
SÍ
NO
Sustituir en letra 0 del canal 4 de la fuente por relleno
Sustituir en posición 'puntero' del canal 4 de la fuente por relleno
FIN
Rellenar posición 2 de memoria de voz con el dato que llega del buffer de entrada del switch
Ciclo = "Escritura"; iteración++; índice++;
iteración = 8
NO
SÍ
Ciclo = "Espera"; iteración++; índice++;
iteración = 9
NO
SÍ
Transmisión en líneas
fin = true
SÍ
NO
Buffer de la fuente ocupado por relleno
Buffer de la fuente ocupado por slot 5
Ciclo = "Lectura"; iteración++; índice++;
Se lee en memoria de voz (lectura controlada por control[3]) y el dato pasa al buffer de salida del switch


iteración = 10
NO
SÍ
Situar voz en canal 1 del receptor
fin = true
SÍ
NO
Sustituir en letra 0 del canal 5 de la fuente por relleno
Sustituir en posición 'puntero' del canal 5 de la fuente por relleno
FIN
Rellenar posición 3 de memoria de voz con el dato que llega del buffer de entrada del switch
Ciclo = "Escritura"; iteración++; índice++;
iteración = 11
NO
Ciclo = "Espera"; iteración++; índice++;
iteración = 12
NO
SÍ
Transmisión en líneas
fin = true
SÍ
NO
Buffer de la fuente ocupado por relleno
Buffer de la fuente ocupado por slot 6
Ciclo = "Lectura"; iteración++; índice++;
Se lee en memoria de voz (lectura controlada por control[4]) y el dato pasa al buffer de salida del switch


iteración = 13
NO
SÍ
Situar voz en canal 2 del receptor
fin = true
SÍ
NO
Sustituir en letra 0 del canal 6 de la fuente por relleno
Sustituir en posición 'puntero' del canal 6 de la fuente por relleno
FIN
Rellenar posición 4 de memoria de voz con el dato que llega del buffer de entrada del switch
Ciclo = "Escritura"; iteración++; índice++;
iteración = 14
NO
Ciclo = "Espera"; iteración++; índice++;
iteración = 15
NO
SÍ
Transmisión en líneas
fin = true
SÍ
NO
Buffer de la fuente ocupado por relleno
Buffer de la fuente ocupado por slot 7
Ciclo = "Lectura"; iteración++; índice++;
Se lee en memoria de voz (lectura controlada por control[5]) y el dato pasa al buffer de salida del switch


iteración = 16
NO
SÍ
Situar voz en canal 3 del receptor
fin = true
SÍ
NO
Sustituir en letra 0 del canal 7 de la fuente por relleno
Sustituir en posición 'puntero' del canal 7 de la fuente por relleno
FIN
Rellenar posición 5 de memoria de voz con el dato que llega del buffer de entrada del switch
Ciclo = "Escritura"; iteración++; índice++;
iteración = 17
NO
SÍ
SÍ
puntero = 5
NO
fin = true;
Ciclo = "Espera"; iteración++; puntero++; índice++;
iteración = 18
NO
SÍ
Transmisión en líneas
fin = true
SÍ
NO
Buffer de la fuente ocupado por relleno
Buffer de la fuente ocupado por slot 0
Ciclo = "Lectura"; iteración++; índice++;
Se lee en memoria de voz (lectura controlada por control[6]) y el dato pasa al buffer de salida del switch


iteración = 19
NO
SÍ
Situar voz en canal 4 del receptor
fin = true
SÍ
NO
Sustituir en letra 0 del canal 0 de la fuente por relleno
Sustituir en posición 'puntero' del canal 0 de la fuente por relleno
FIN
Rellenar posición 6 de memoria de voz con el dato que llega del buffer de entrada del switch
Ciclo = "Escritura"; iteración++; índice++;
iteración = 20
NO
Ciclo = "Espera"; iteración++; índice++;
iteración = 21
NO
SÍ
Transmisión en líneas
fin = true
SÍ
NO
Buffer de la fuente ocupado por relleno
Buffer de la fuente ocupado por slot 1
Ciclo = "Lectura"; iteración++; índice++;
Se lee en memoria de voz (lectura controlada por control[7]) y el dato pasa al buffer de salida del switch

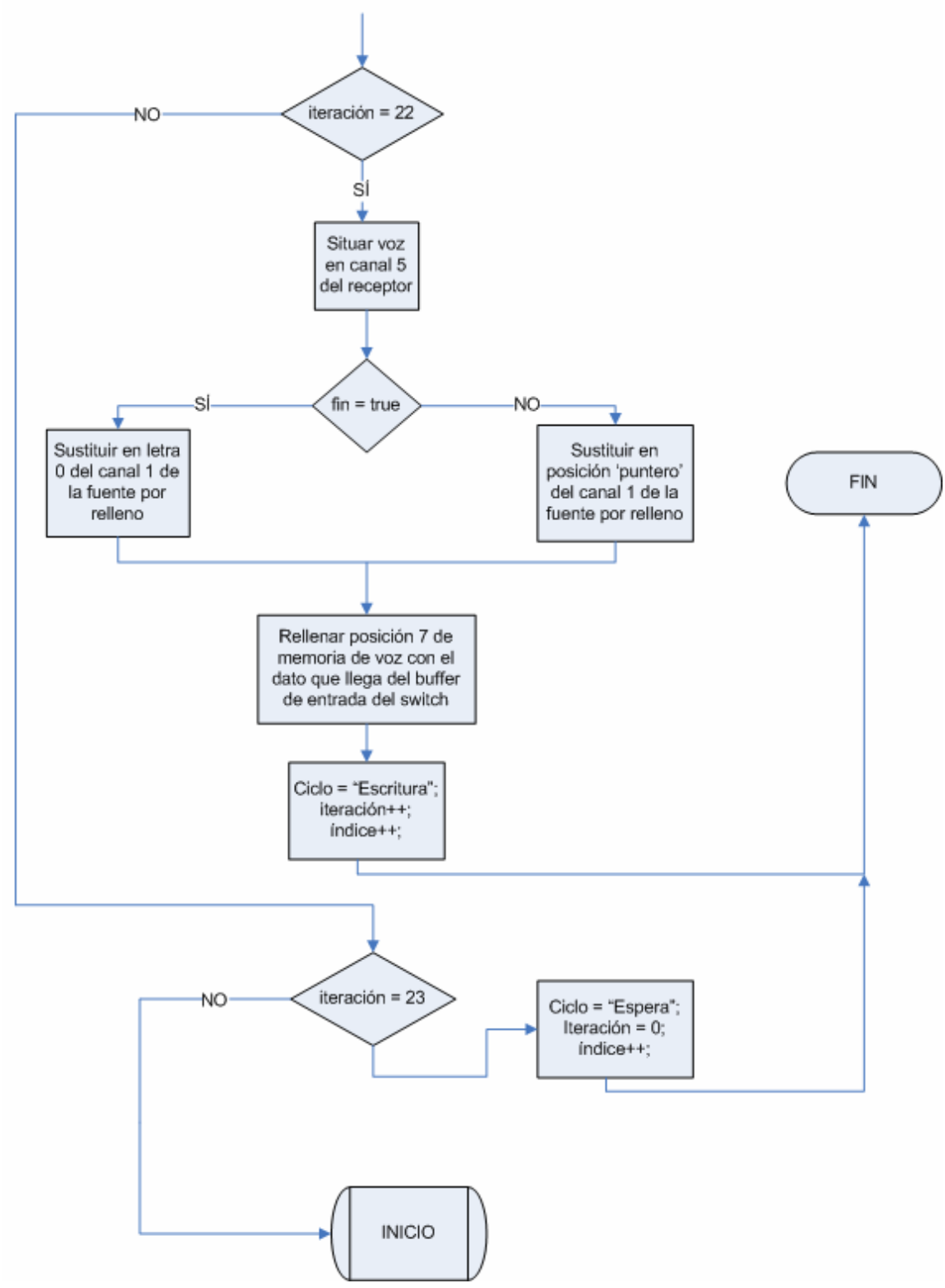


Figura 31: Diagrama de flujo del método viajeCiclo() de la OpcionB1

OpcionB2

Contiene funcionalidad (algoritmo) e interfaz gráfica. Los métodos más destacados son:

- `public void esLetra(String s, Jlabel l[])`

  En el receptor hay ruido, información útil e información de relleno, este método resalta la información útil en blanco.

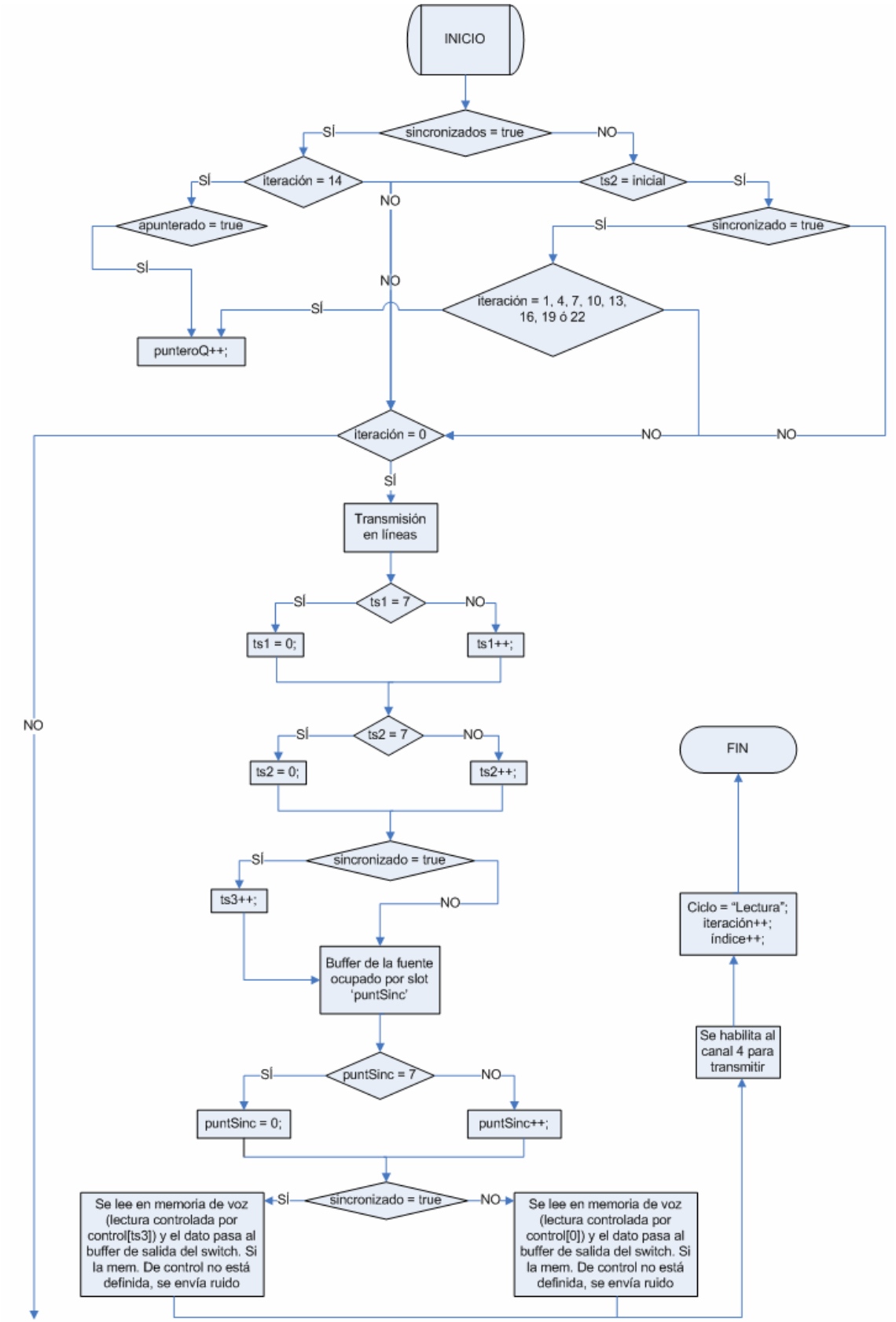
- `public void iniciarContadores()`

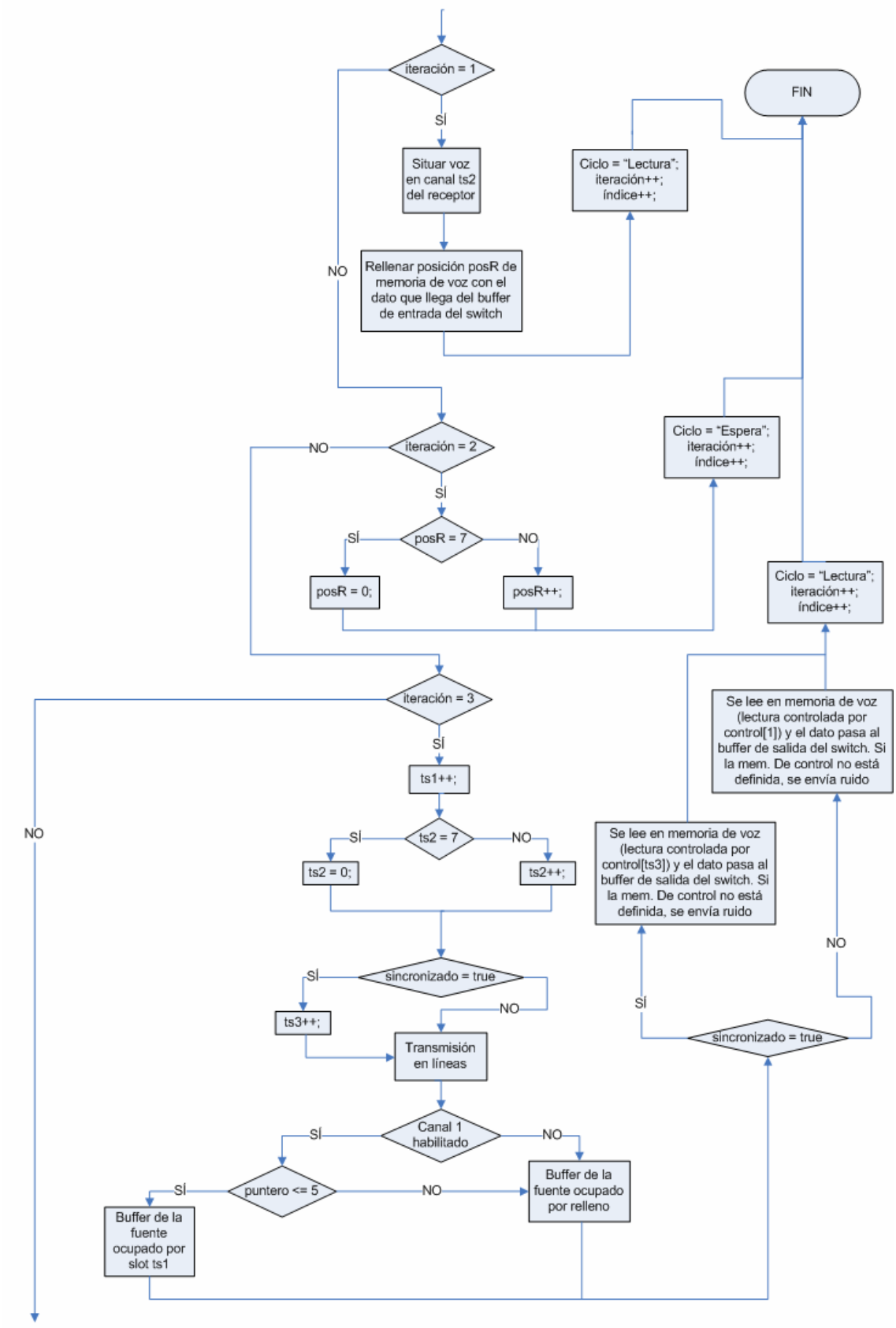
  Genera los valores aleatorios de los registros Slot de fuente y receptor.

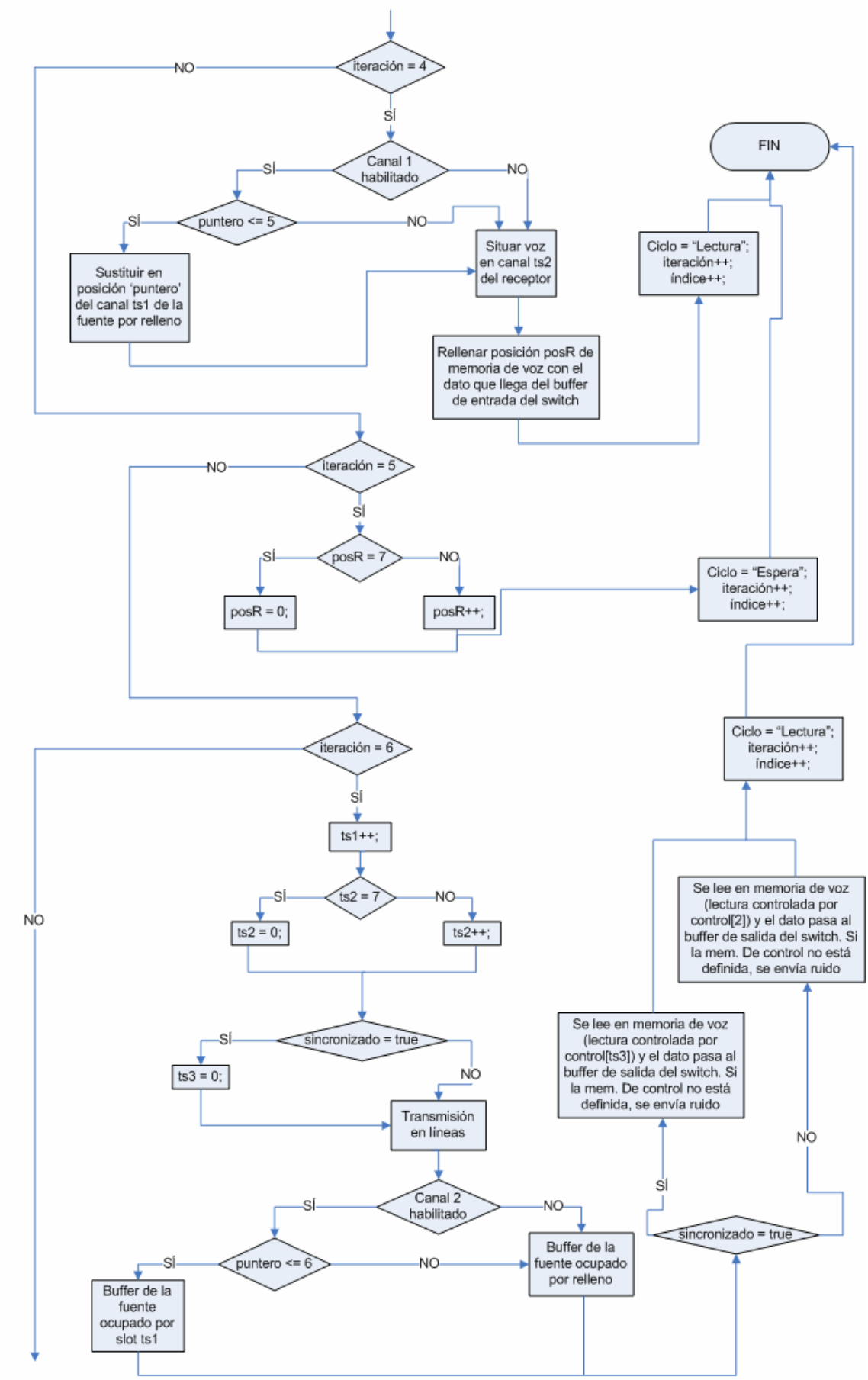
- `public void viajeCiclo(String pe, String qe)`

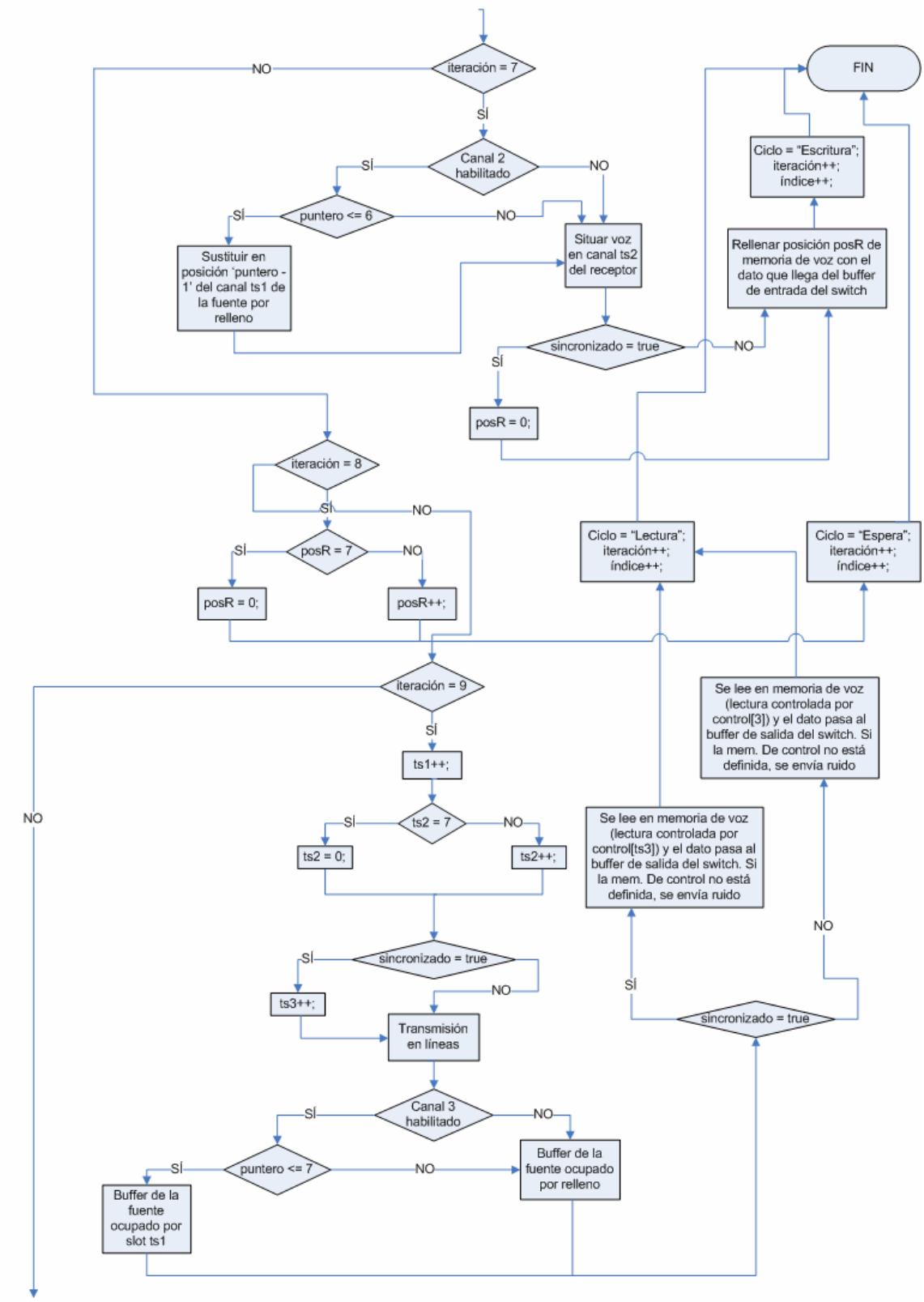
  Como en la anterior opción, éste es el método fundamental donde está incluido el algoritmo de transmisión. Hay 4 tipos de puntero, el que cuenta las posiciones del canal de sincronismo (*puntSinc*), el que cuenta las posiciones del canal de señalización (*puntSeñ*), el que cuenta las posiciones de los canales vocales en la fuente (*puntero*) y el que cuenta las posiciones de los canales en el receptor (*punteroQ*).

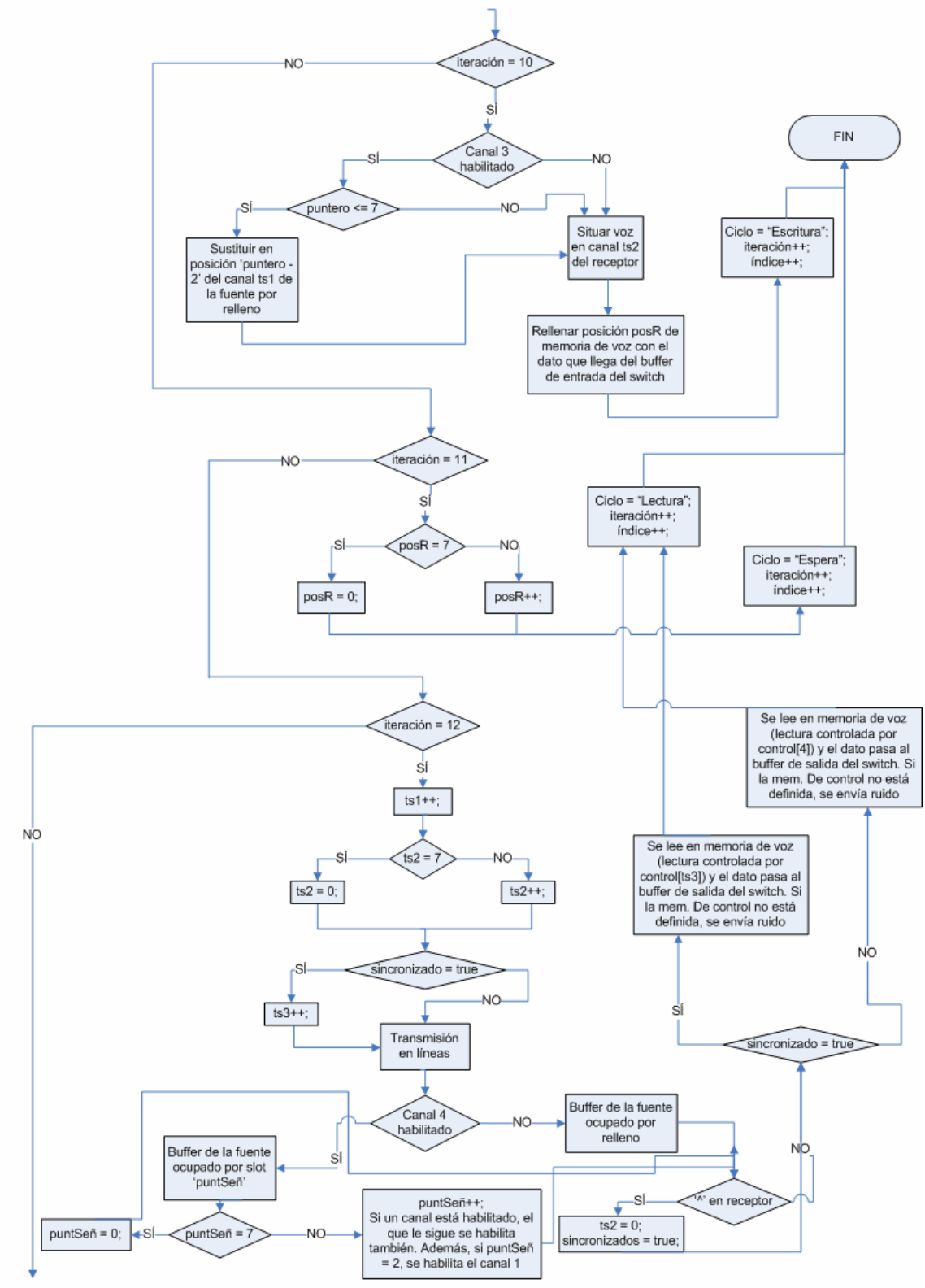
  El diagrama de flujo de este método en la opción de sincronismo y señalización es el siguiente:

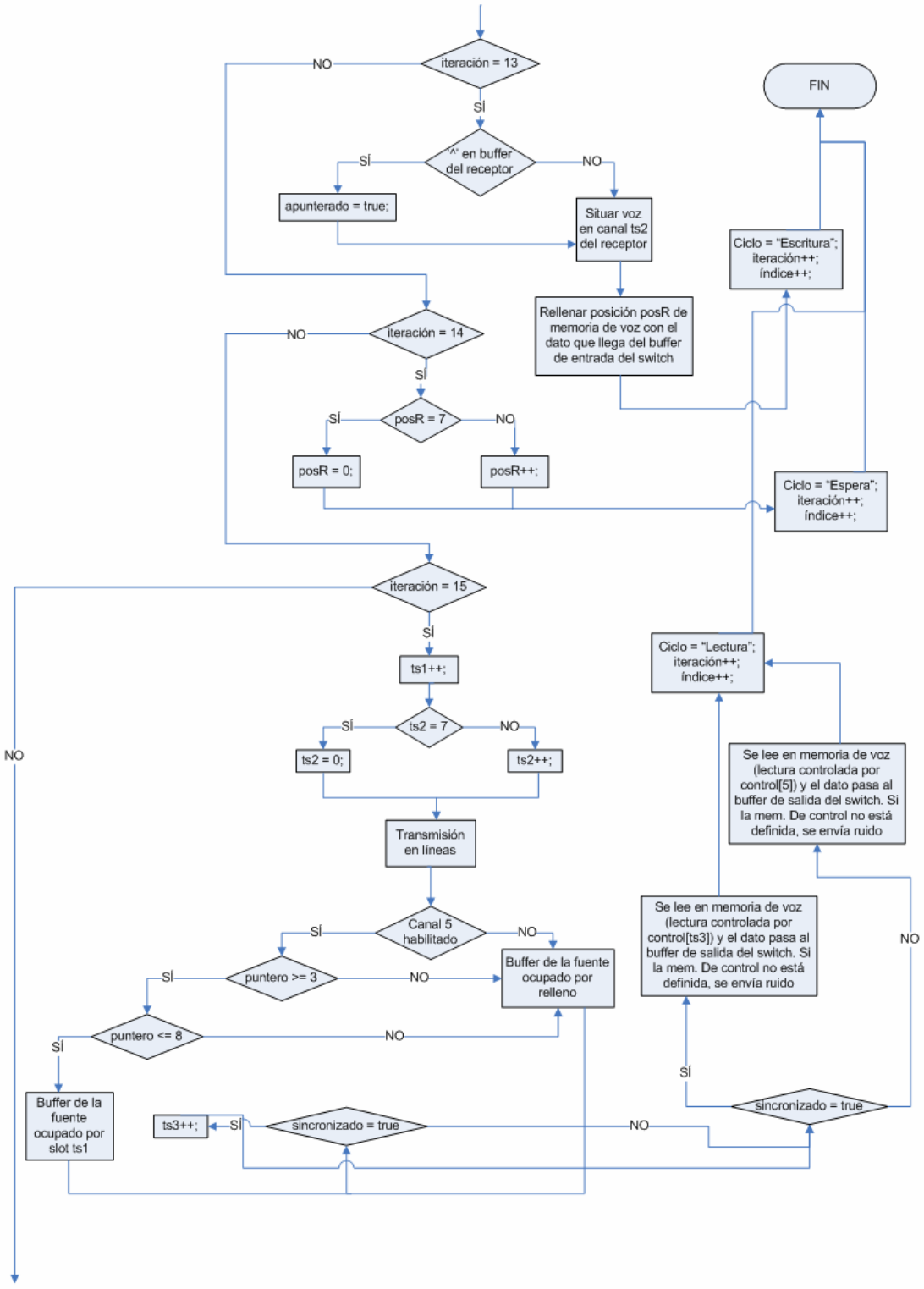

INICIO
sincronizados = true
SÍ
NO
iteración = 14
ts2 = inicial
apunterado = true
sincronizado = true
iteración = 1, 4, 7, 10, 13, 16, 19 ó 22
punteroQ++;
iteración = 0
Transmisión en líneas
ts1 = 7
ts1 = 0;
ts1++;
ts2 = 7
ts2 = 0;
ts2++;
sincronizado = true
ts3++;
Buffer de la fuente ocupado por slot 'puntSinc'
puntSinc = 7
puntSinc = 0;
puntSinc++;
Se lee en memoria de voz (lectura controlada por control[ts3]) y el dato pasa al buffer de salida del switch. Si la mem. De control no está definida, se envía ruido
Se lee en memoria de voz (lectura controlada por control[0]) y el dato pasa al buffer de salida del switch. Si la mem. De control no está definida, se envía ruido
Se habilita al canal 4 para transmitir
Ciclo = "Lectura"; iteración++; índice++;
FIN

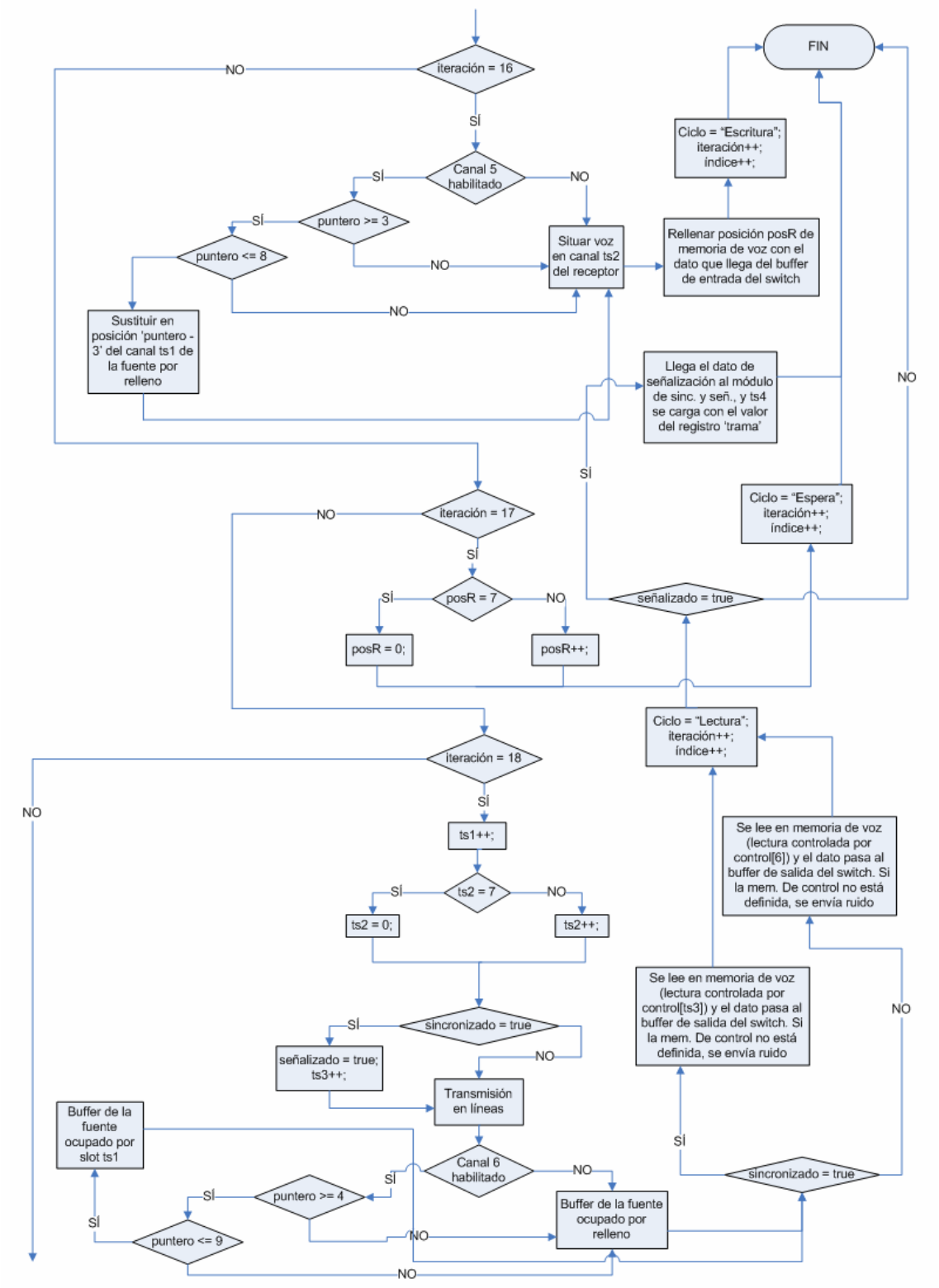

iteración = 1
SÍ
NO
Situar voz en canal ts2 del receptor
Rellenar posición posR de memoria de voz con el dato que llega del buffer de entrada del switch
Ciclo = "Lectura"; iteración++; índice++;
FIN
iteración = 2
NO
SÍ
posR = 7
SÍ
NO
posR = 0;
posR++;
Ciclo = "Espera"; iteración++; índice++;
Ciclo = "Lectura"; iteración++; índice++;
iteración = 3
NO
SÍ
ts1++;
ts2 = 7
SÍ
NO
ts2 = 0;
ts2++;
Se lee en memoria de voz (lectura controlada por control[1]) y el dato pasa al buffer de salida del switch. Si la mem. De control no está definida, se envía ruido
Se lee en memoria de voz (lectura controlada por control[ts3]) y el dato pasa al buffer de salida del switch. Si la mem. De control no está definida, se envía ruido
sincronizado = true
SÍ
NO
ts3++;
Transmisión en líneas
sincronizado = true
SÍ
NO
Canal 1 habilitado
SÍ
NO
puntero <= 5
SÍ
NO
Buffer de la fuente ocupado por slot ts1
Buffer de la fuente ocupado por relleno

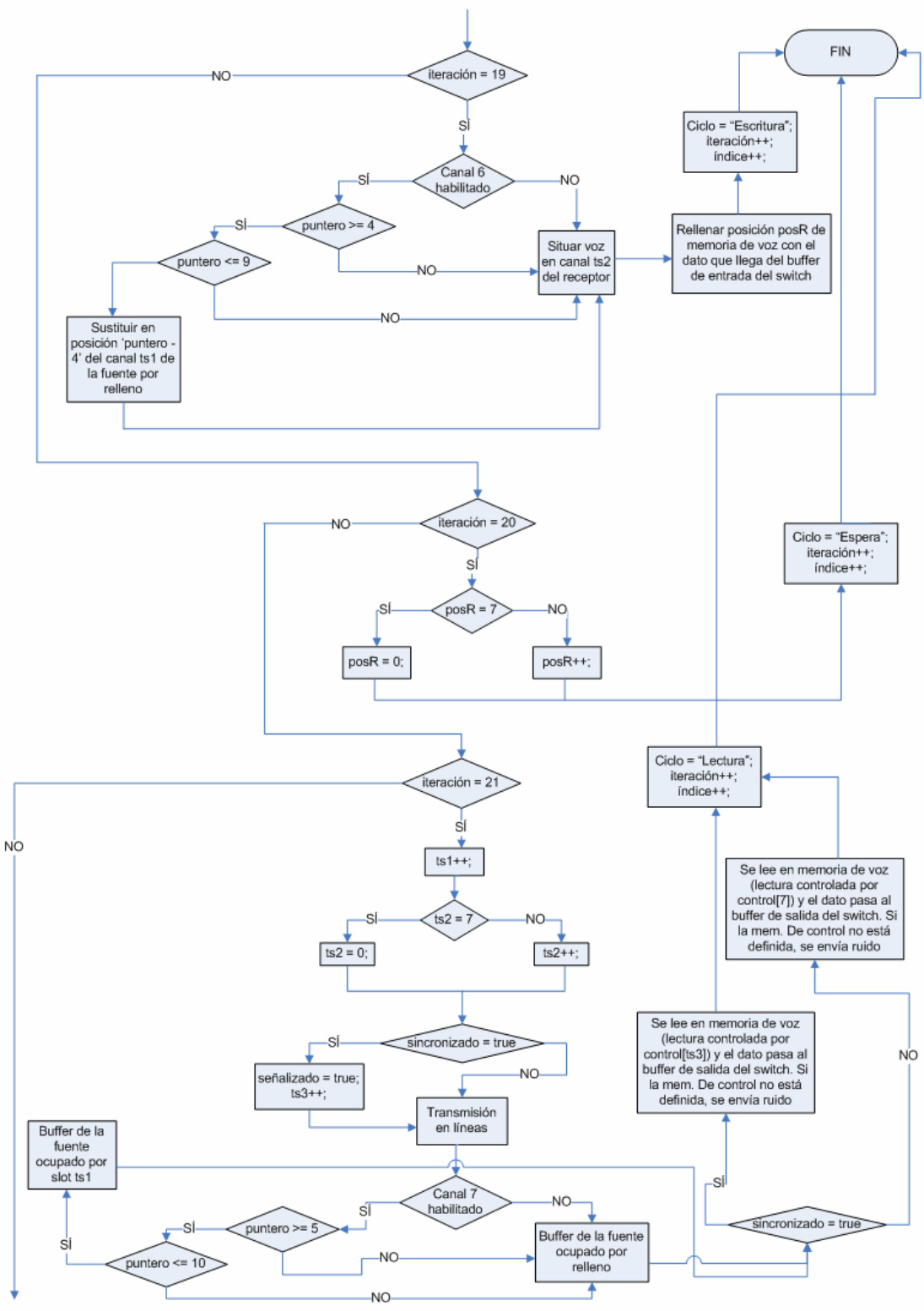

iteración = 4
NO
SÍ
Canal 1 habilitado
SÍ
NO
puntero <= 5
SÍ
NO
Sustituir en posición 'puntero' del canal ts1 de la fuente por relleno
Situar voz en canal ts2 del receptor
Rellenar posición posR de memoria de voz con el dato que llega del buffer de entrada del switch
Ciclo = "Lectura"; iteración++; índice++;
FIN
iteración = 5
NO
SÍ
posR = 7
SÍ
NO
posR = 0;
posR++;
Ciclo = "Espera"; iteración++; índice++;
iteración = 6
NO
SÍ
ts1++;
ts2 = 7
SÍ
NO
ts2 = 0;
ts2++;
sincronizado = true
SÍ
NO
ts3 = 0;
Transmisión en líneas
Canal 2 habilitado
SÍ
NO
puntero <= 6
SÍ
NO
Buffer de la fuente ocupado por slot ts1
Buffer de la fuente ocupado por relleno
Ciclo = "Lectura"; iteración++; índice++;
Se lee en memoria de voz (lectura controlada por control[2]) y el dato pasa al buffer de salida del switch. Si la mem. De control no está definida, se envía ruido
Se lee en memoria de voz (lectura controlada por control[ts3]) y el dato pasa al buffer de salida del switch. Si la mem. De control no está definida, se envía ruido
sincronizado = true
SÍ
NO


iteración = 7
NO
SÍ
Canal 2 habilitado
SÍ
NO
puntero <= 6
SÍ
NO
Sustituir en posición 'puntero - 1' del canal ts1 de la fuente por relleno
Situar voz en canal ts2 del receptor
sincronizado = true
NO
SÍ
posR = 0;
Rellenar posición posR de memoria de voz con el dato que llega del buffer de entrada del switch
Ciclo = "Escritura"; iteración++; índice++;
FIN
iteración = 8
SÍ
NO
posR = 7
SÍ
NO
posR = 0;
posR++;
Ciclo = "Lectura"; iteración++; índice++;
Ciclo = "Espera"; iteración++; índice++;
iteración = 9
NO
SÍ
ts1++;
ts2 = 7
SÍ
NO
ts2 = 0;
ts2++;
Se lee en memoria de voz (lectura controlada por control[3]) y el dato pasa al buffer de salida del switch. Si la mem. De control no está definida, se envía ruido
Se lee en memoria de voz (lectura controlada por control[ts3]) y el dato pasa al buffer de salida del switch. Si la mem. De control no está definida, se envía ruido
sincronizado = true
SÍ
NO
sincronizado = true
SÍ
NO
ts3++;
Transmisión en líneas
Canal 3 habilitado
SÍ
NO
puntero <= 7
SÍ
NO
Buffer de la fuente ocupado por slot ts1
Buffer de la fuente ocupado por relleno


iteración = 10
NO
SÍ
Canal 3 habilitado
SÍ
NO
puntero <= 7
SÍ
NO
Sustituir en posición 'puntero - 2' del canal ts1 de la fuente por relleno
Situar voz en canal ts2 del receptor
Rellenar posición posR de memoria de voz con el dato que llega del buffer de entrada del switch
FIN
Ciclo = "Escritura"; iteración++; índice++;
iteración = 11
NO
SÍ
posR = 7
SÍ
NO
posR = 0;
posR++;
Ciclo = "Lectura"; iteración++; índice++;
Ciclo = "Espera"; iteración++; índice++;
iteración = 12
NO
SÍ
ts1++;
ts2 = 7
SÍ
NO
ts2 = 0;
ts2++;
sincronizado = true
SÍ
NO
ts3++;
Transmisión en líneas
Se lee en memoria de voz (lectura controlada por control[4]) y el dato pasa al buffer de salida del switch. Si la mem. De control no está definida, se envía ruido
Se lee en memoria de voz (lectura controlada por control[ts3]) y el dato pasa al buffer de salida del switch. Si la mem. De control no está definida, se envía ruido
NO
SÍ
sincronizado = true
Canal 4 habilitado
NO
SÍ
Buffer de la fuente ocupado por relleno
Buffer de la fuente ocupado por slot 'puntSeñ'
puntSeñ = 0;
SÍ
puntSeñ = 7
NO
puntSeñ++;
Si un canal está habilitado, el que le sigue se habilita también. Además, si puntSeñ = 2, se habilita el canal 1
'^' en receptor
SÍ
NO
ts2 = 0;
sincronizados = true;


iteración = 13
NO
SÍ
'A' en buffer del receptor
SÍ
NO
apunterado = true;
Situar voz en canal ts2 del receptor
FIN
Ciclo = "Escritura"; iteración++; índice++;
Rellenar posición posR de memoria de voz con el dato que llega del buffer de entrada del switch
iteración = 14
NO
SÍ
posR = 7
SÍ
NO
posR = 0;
posR++;
Ciclo = "Espera"; iteración++; índice++;
iteración = 15
NO
SÍ
ts1++;
Ciclo = "Lectura"; iteración++; índice++;
ts2 = 7
SÍ
NO
ts2 = 0;
ts2++;
Se lee en memoria de voz (lectura controlada por control[5]) y el dato pasa al buffer de salida del switch. Si la mem. De control no está definida, se envía ruido
Transmisión en líneas
Canal 5 habilitado
SÍ
NO
Se lee en memoria de voz (lectura controlada por control[ts3]) y el dato pasa al buffer de salida del switch. Si la mem. De control no está definida, se envía ruido
NO
puntero >= 3
SÍ
NO
Buffer de la fuente ocupado por relleno
puntero <= 8
SÍ
NO
Buffer de la fuente ocupado por slot ts1
ts3++;
SÍ
sincronizado = true
NO
SÍ
sincronizado = true


FIN
iteración = 16
NO
SÍ
Canal 5 habilitado
SÍ
NO
puntero >= 3
SÍ
puntero <= 8
NO
NO
Situar voz en canal ts2 del receptor
Rellenar posición posR de memoria de voz con el dato que llega del buffer de entrada del switch
Ciclo = "Escritura"; iteración++; índice++;
Sustituir en posición 'puntero - 3' del canal ts1 de la fuente por relleno
Llega el dato de señalización al módulo de sinc. y señ., y ts4 se carga con el valor del registro 'trama'
NO
SÍ
iteración = 17
NO
SÍ
posR = 7
SÍ
NO
posR = 0;
posR++;
señalizado = true
Ciclo = "Espera"; iteración++; índice++;
Ciclo = "Lectura"; iteración++; índice++;
iteración = 18
NO
SÍ
ts1++;
ts2 = 7
SÍ
NO
ts2 = 0;
ts2++;
sincronizado = true
SÍ
NO
señalizado = true; ts3++;
Transmisión en líneas
Se lee en memoria de voz (lectura controlada por control[6]) y el dato pasa al buffer de salida del switch. Si la mem. De control no está definida, se envía ruido
Se lee en memoria de voz (lectura controlada por control[ts3]) y el dato pasa al buffer de salida del switch. Si la mem. De control no está definida, se envía ruido
NO
SÍ
sincronizado = true
Buffer de la fuente ocupado por slot ts1
Canal 6 habilitado
SÍ
NO
puntero >= 4
SÍ
SÍ
puntero <= 9
NO
NO
Buffer de la fuente ocupado por relleno


iteración = 19
NO
SÍ
Canal 6 habilitado
SÍ
NO
puntero >= 4
SÍ
puntero <= 9
NO
NO
Situar voz en canal ts2 del receptor
Sustituir en posición 'puntero - 4' del canal ts1 de la fuente por relleno
Rellenar posición posR de memoria de voz con el dato que llega del buffer de entrada del switch
Ciclo = "Escritura"; iteración++; índice++;
FIN
iteración = 20
NO
SÍ
posR = 7
SÍ
NO
posR = 0;
posR++;
Ciclo = "Espera"; iteración++; índice++;
iteración = 21
NO
SÍ
ts1++;
ts2 = 7
SÍ
NO
ts2 = 0;
ts2++;
sincronizado = true
SÍ
NO
señalizado = true; ts3++;
Transmisión en líneas
Buffer de la fuente ocupado por slot ts1
Canal 7 habilitado
SÍ
NO
puntero >= 5
SÍ
puntero <= 10
SÍ
NO
NO
Buffer de la fuente ocupado por relleno
Ciclo = "Lectura"; iteración++; índice++;
Se lee en memoria de voz (lectura controlada por control[7]) y el dato pasa al buffer de salida del switch. Si la mem. De control no está definida, se envía ruido
Se lee en memoria de voz (lectura controlada por control[ts3]) y el dato pasa al buffer de salida del switch. Si la mem. De control no está definida, se envía ruido
SÍ
sincronizado = true
NO

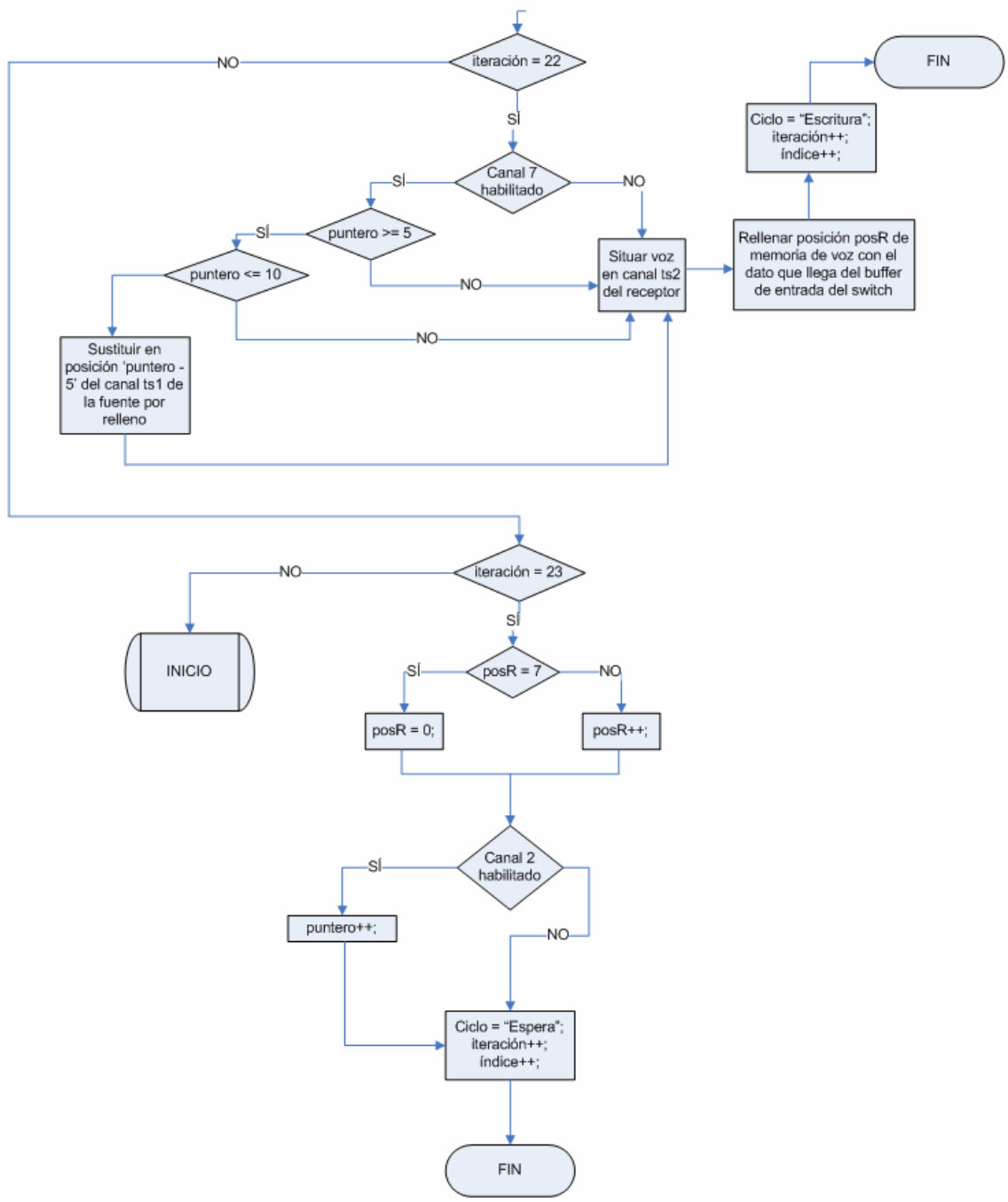


Figura 32: Diagrama de flujo del método viajeCiclo() de la OpcionB2

A continuación se pueden ver los diagramas UML de las clases OpcionB1 y OpcionB2:

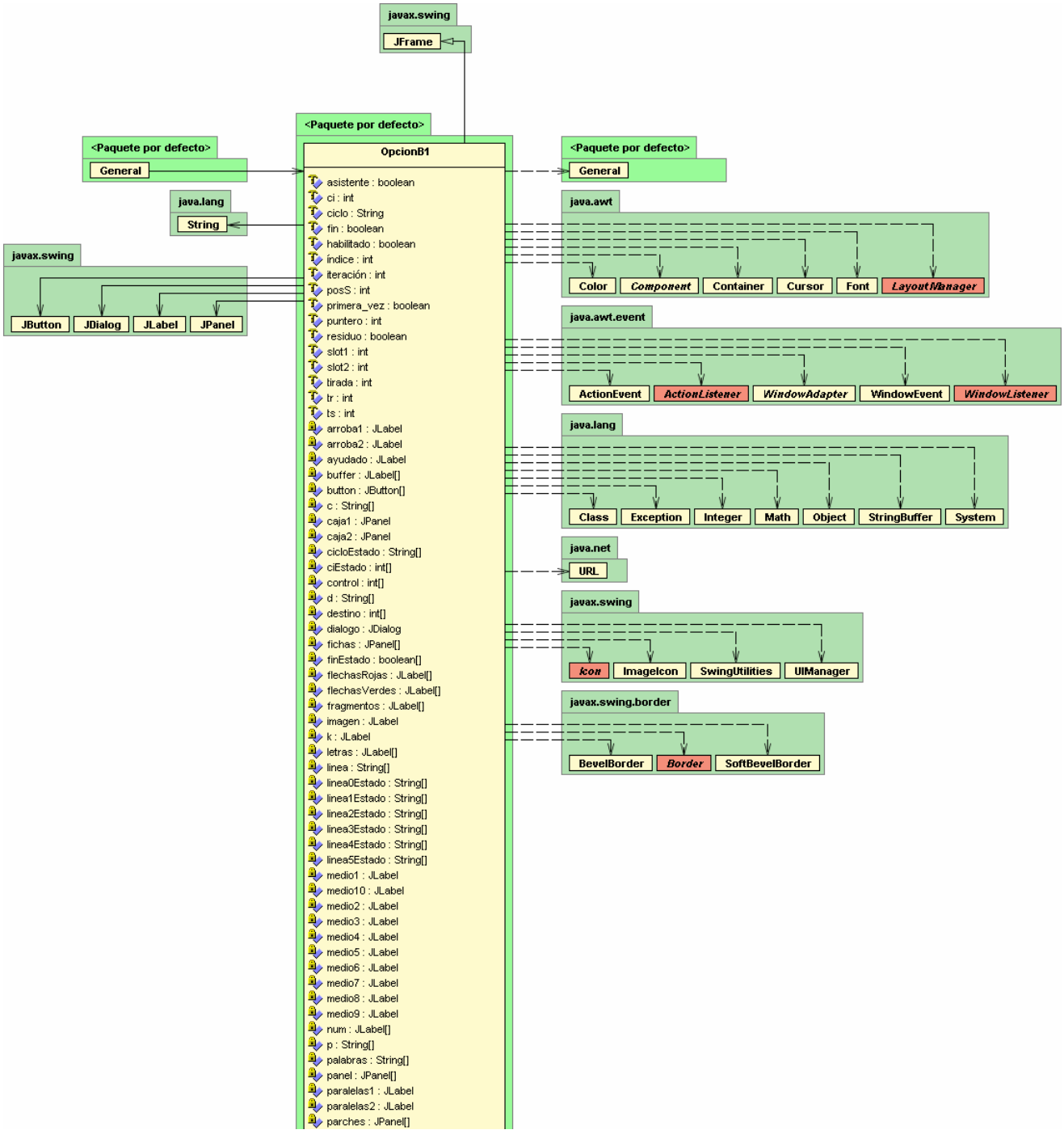


Figura 33: Primera mitad del diagrama UML de la OpcionB1

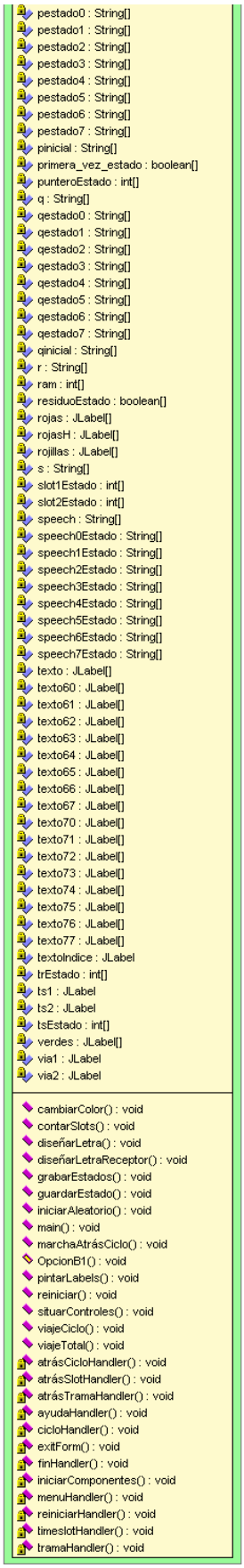


Figura 33: Segunda mitad del diagrama UML de la OpcionB1

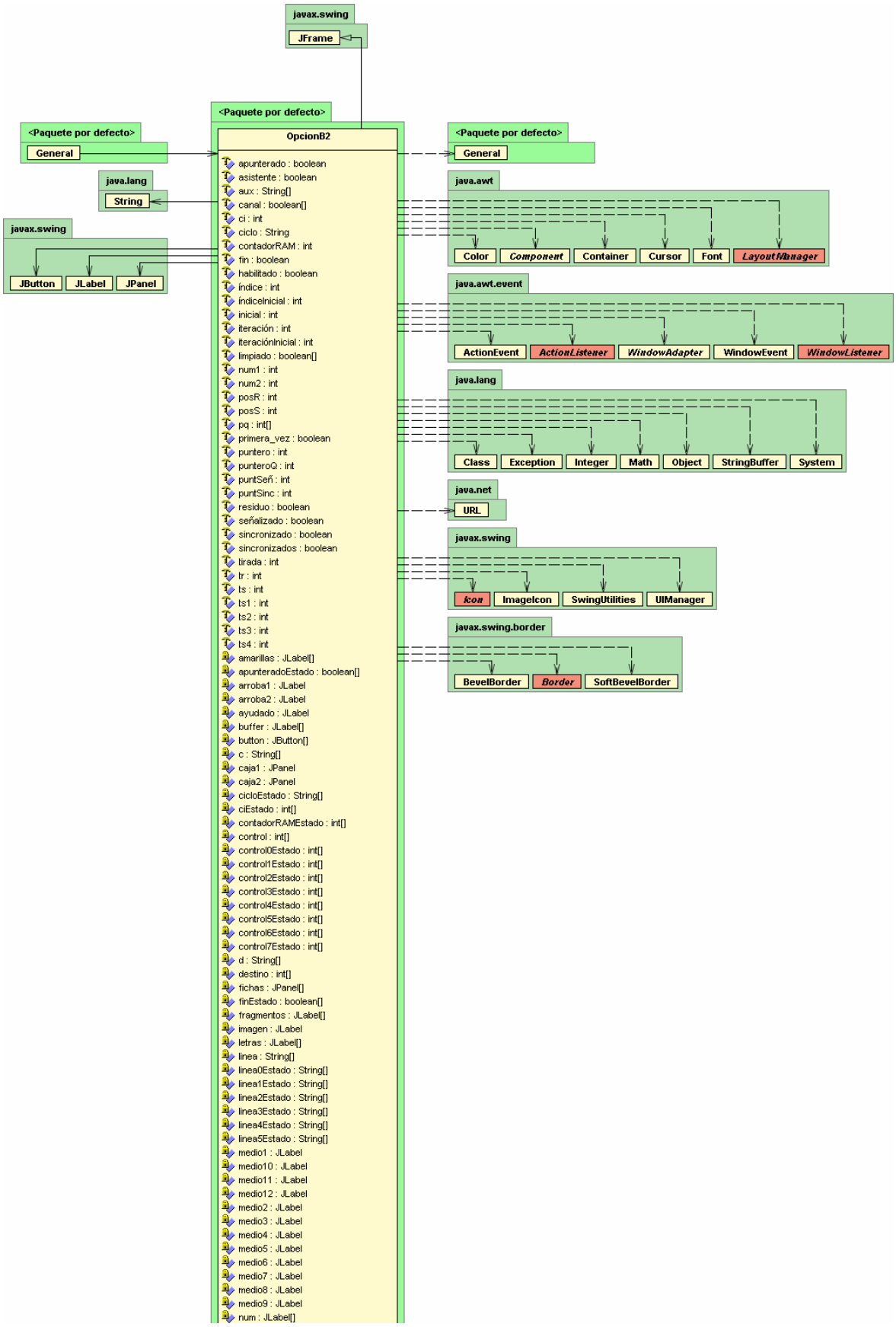


Figura 34: Primera mitad del diagrama UML de la OpcionB2

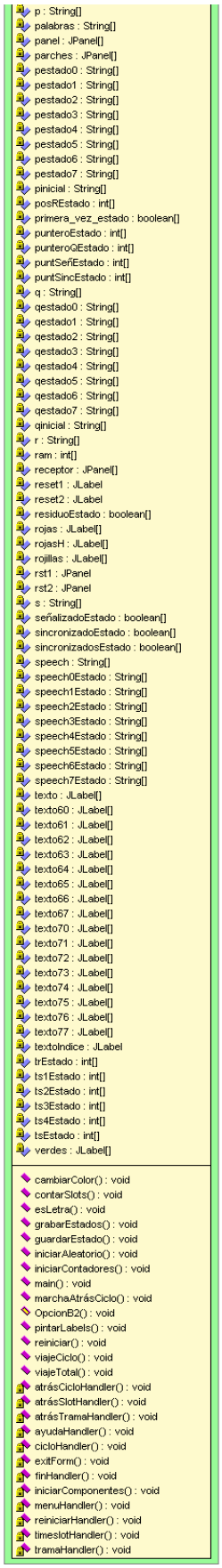


Figura 35: Segunda mitad del diagrama UML de la OpcionB2

Capítulo 6
Uso académico de la aplicación

6.1. Teoría de la conmutación

Antes de empezar a comentar el uso académico del programa, se comentarán brevemente conceptos básicos teóricos de conmutación.

Conmutación

La conmutación de circuitos es una técnica que permite que dos terminales, emisor y receptor, se comuniquen a través de un circuito establecido para tal propósito antes del inicio de la misma y liberando una vez que ha terminado, quedando en este caso a disposición de otros usuarios para su utilización.

La conmutación de circuitos es orientada a la conexión y por tanto consta de las 3 fases siguientes:

- Fase de establecimiento de la llamada.
- Fase de transferencia de la información.
- Fase de desconexión de la llamada.

Durante el establecimiento de la llamada se reservan recursos físicamente (canales dentro de la trama TDM) de forma que los bits que entran por un puerto son conmutados “instantáneamente” a un canal de un puerto de salida.

Este tipo de redes realizan la conmutación a nivel físico, transportando información, fundamentalmente voz, de forma digital. La transmisión de información digital como fax y datos, necesita de una transformación digital/analógica y analógica/digital en sus extremos.

Tipos de conmutación

Según su operación:

- Manual y automática
- Analógica y digital
- Espacial y temporal

Según el tipo de servicio:

- De circuitos
- De paquetes

Según su utilización:

- Privadas

- Públicas

En el caso del simulador didáctico implementado en este proyecto fin de carrera, los tipos de conmutación a utilizar son la conmutación de circuitos y la conmutación temporal.

Conmutación de circuitos

Es una técnica en la que los equipos que se comunican entre sí utilizan un canal dedicado extremo a extremo que se mantiene durante el tiempo de duración de la llamada o por el periodo de contratación.

Conmutación temporal

Método de conmutación de circuitos en que el tiempo está dividido en ranuras temporales y a cada flujo de información se le asocia una determinada ranura temporal, de tal forma que la información se encapsula en dichas ranuras temporales para viajar por dicho canal y se reconstruye cuando llega a su destino.

6.2. ¿Qué es el multiplexado?

El multiplexado es la técnica de combinar varias señales diferentes y transmitirlas simultáneamente sobre un solo canal o medio sin que se interfieran. El principal objetivo de la multicanalización es la economía.

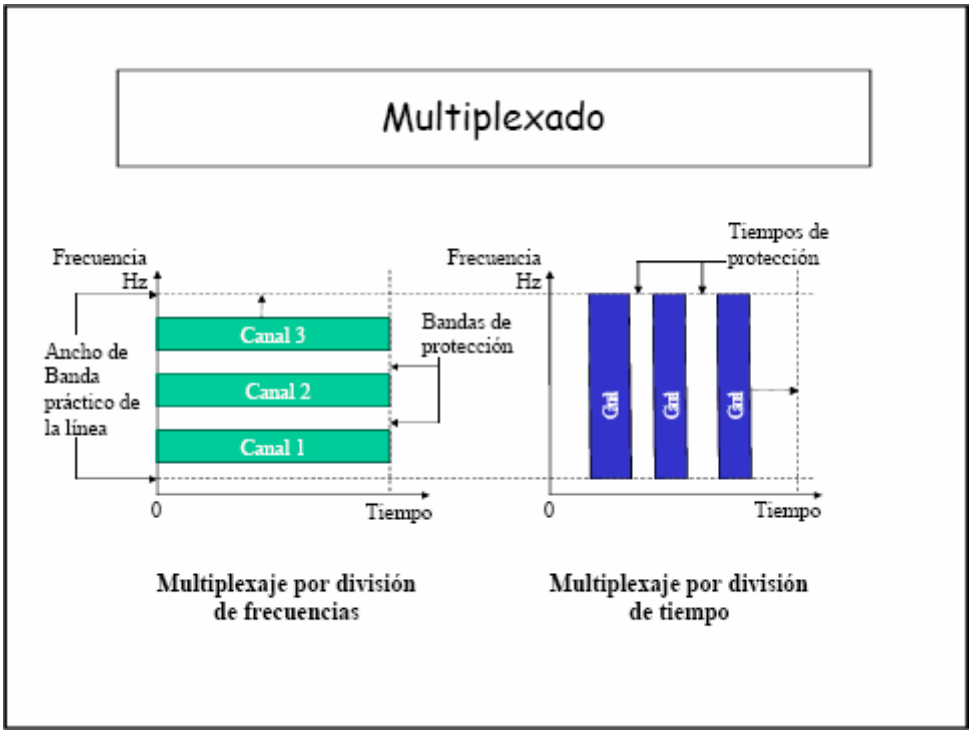


Figura 36: Multiplexado por división en frecuencia y por división en el tiempo

El multiplexado consiste en compartir una línea por muchos canales y puede realizarse con base en la frecuencia, el llamado **multiplexado por división de frecuencia**, o con base en el tiempo, llamado **multiplexado por división en el tiempo.**

Multiplexado por división de frecuencia (FDM), cada canal de voz se asigna a una porción única del ancho de banda disponible en la línea, y todo el tiempo utiliza en forma exclusiva el ancho de banda asignado.

Multiplexado por división en el tiempo (TDM), a cada canal se le asigna la totalidad del ancho de banda de la línea por períodos de tiempo regulares específicos; los intervalos de tiempo dependen, obviamente, del número de canales que comparten la línea. En cada caso hay bandas de protección que separan los canales adyacentes para evitar interferencia mutua (que en el simulador desarrollado en este proyecto fin de carrera se refleja mediante el ciclo de espera).

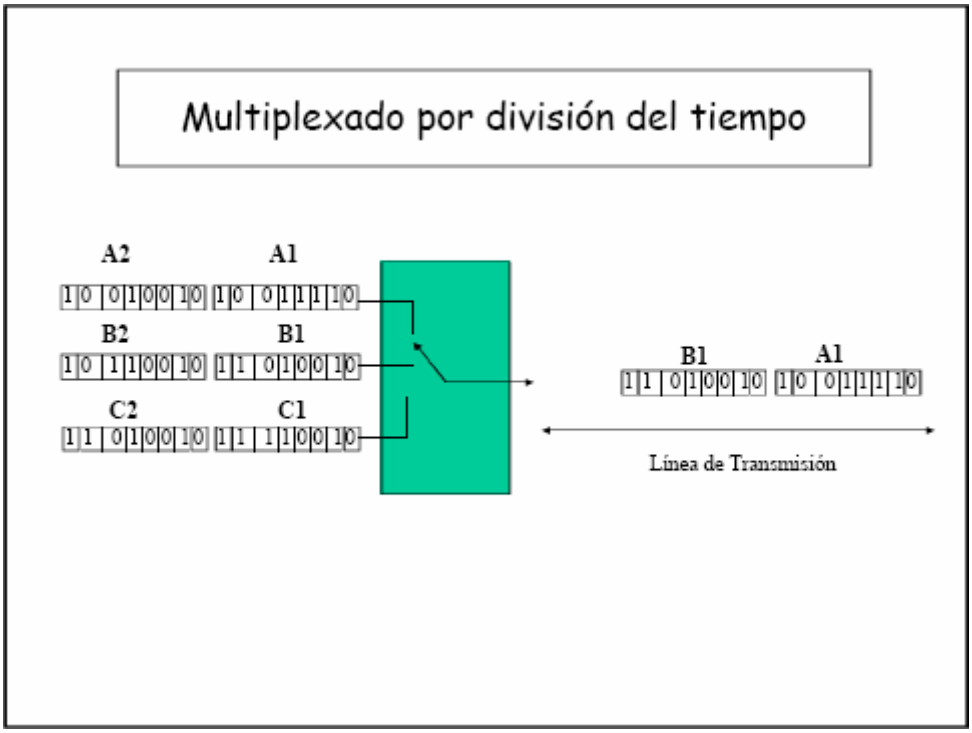


Figura 37: Multiplexado por división en el tiempo

La transmisión digital es a base de pulsos discretos y no de señales continuas, que en la práctica se lleva a cabo intercalando los pulsos de los diferentes canales de tal manera que la secuencia de 8 bits procedente del primer canal sea seguida de la secuencia de ocho pulsos que procede del segundo canal y así sucesivamente. El equipo multiplexor se puede considerar como un interruptor giratorio que capta por vez 8 bits de cada uno de los canales de entrada A, B, C. Así, el tren de bits de salida del equipo Multiplexor comprenderá, a su vez, el byte A1, el byte B1, el byte C1, espués, reiniciando el ciclo, el byte A2, el byte B2, el byte C2, etc.

Los diferentes canales comparten en tiempo la trayectoria de salida de transmisión.

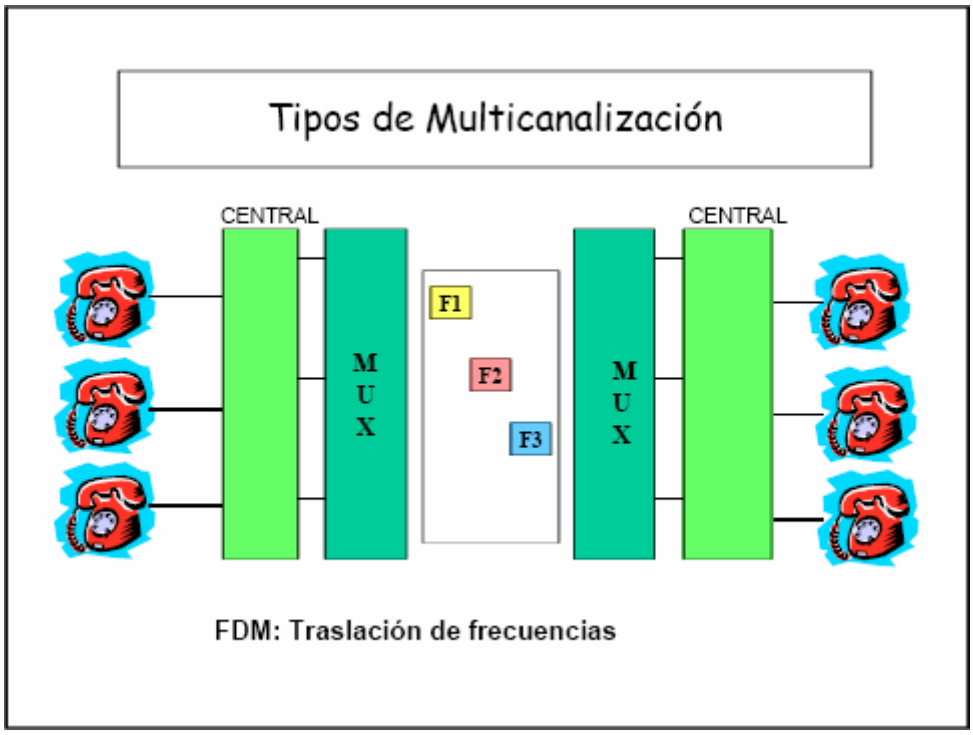


Figura 38: Traslación de frecuencias

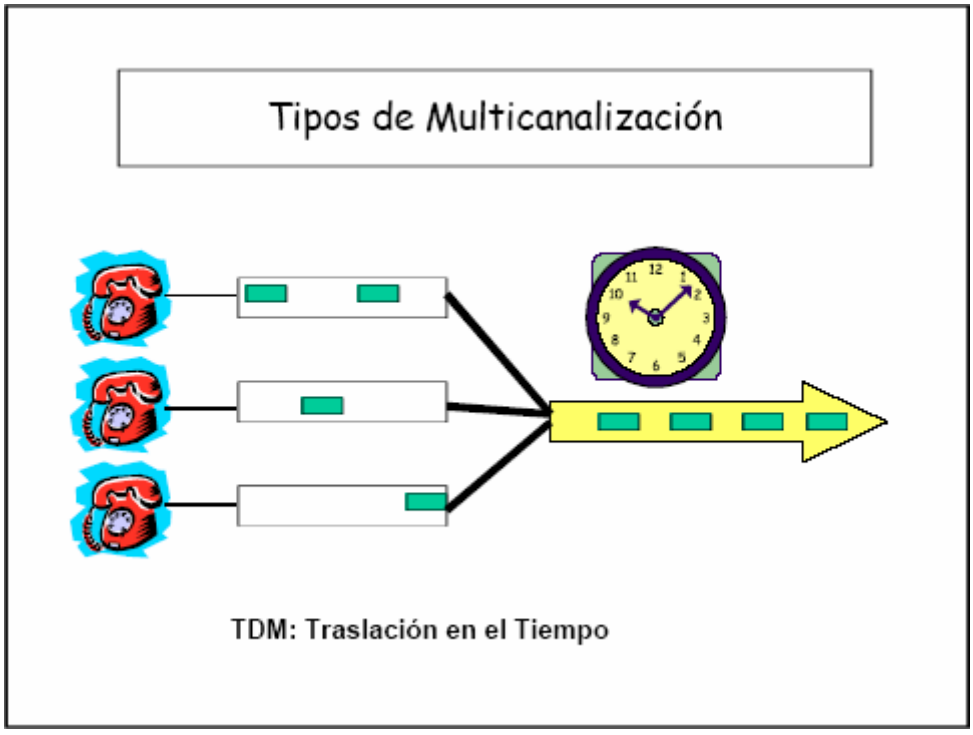


Figura 39: Traslación en el tiempo

Diferencias entre FDM y TDM

FDM

Divide el enlace en varios canales por asignación en ranuras de frecuencias separadas.

TDM

Divide el enlace en varios canales por asignación en ranuras de tiempo separadas.

6.3. Digitalización de señales

Se realiza mediante dispositivos electrónicos denominados conversores analógico/digitales, fundamental para transmitir información de voz por un sistema digital.

La conversión de una señal analógica en otra digitales “equivalente” se realiza en dos partes:

- Muestreo
- Cuantificación

Conversión analógico/digital

El proceso de **muestreo** consiste en tomar “muestras” del valor de la amplitud de la señal a intervalos regulares de tiempo.

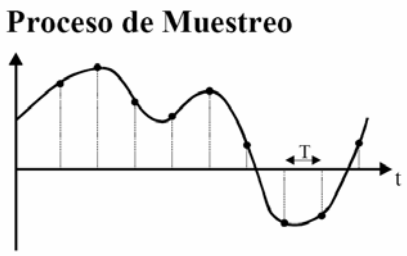


Figura 40: proceso de muestreo representado gráficamente

El proceso de **cuantificación** convierte las muestras en palabras binarias de n bits. Para ello se divide el margen dinámico en 2^n intervalos iguales.

En el ejemplo, n =3 y las palabras “0XX” y “1XX” corresponden a muestras positivas y negativas respectivamente:

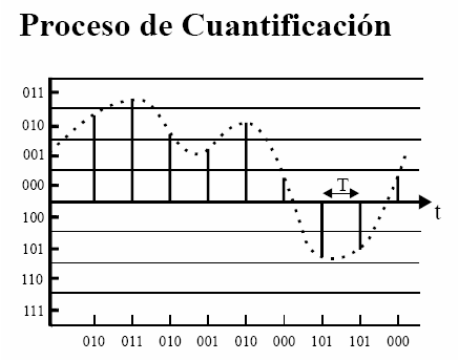


Figura 41: Proceso de cuantificación representado gráficamente

Conversión digital/analógico

Se realiza mediante un conversor digital/analógico en recepción. Convierte la secuencia de palabras binarias en una señal analógica con un perfil de tipo escalón, que al ser filtrada con un filtro paso bajo, pasa a ser aproximadamente la señal analógica original.

La calidad de la señal recuperada depende de la frecuencia de muestreo y del número de bits por muestra.

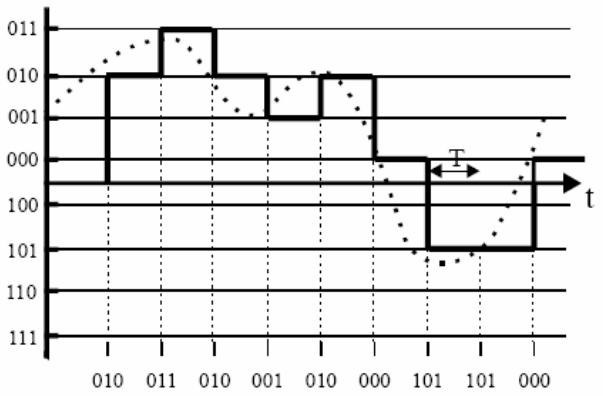


Figura 42: Conversión digital/analógico

Teorema del muestreo (Nyquist)

Para poder recuperar una señal a partir de sus muestras, es necesario que la señal analógica original haya sido muestreada a una frecuencia mayor o igual al doble de su frecuencia máxima.

La mayor parte de información de la señal de voz está por debajo de los 3400 Hz → la frecuencia de muestreo es 8000 Hz (muestras/s), lo que se traduce en una muestra cada T = 125 µs.

En general, se realiza un filtrado paso bajo de la señal antes de pasarla al conversor analógico/digital.

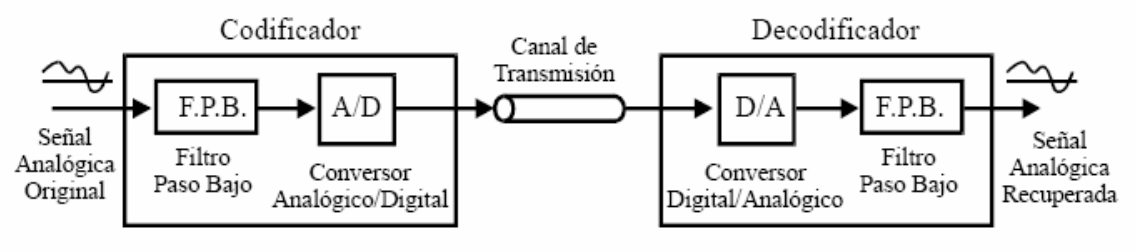


Figura 43: conversión analógico/digital y vicerversa

6.4. Multiplexación por división en el tiempo

Con la multiplexación por división en el tiempo (TDM) se permite compartir el medio de transmisión entre el conjunto de señales digitalizadas que se desean transmitir.

Se define un intervalo de tiempo T, éste se divide en N partes iguales y cada uno de los subintervalos o ranuras temporales (slots) se asignan a cada una de las N señales a transmitir.

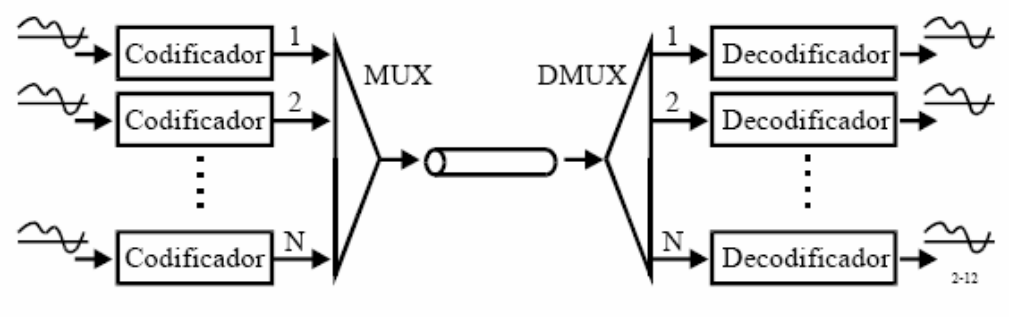


Figura 44: Multiplexación por división en el tiempo. Multiplexores y Demultiplexores

Se denomina canal a la repetición periódica (cada 125 μs) de ranuras temporales asociadas a una misma pareja codificador-decodificador, es decir, a una misma señal de voz → 8 bits/ 125 μs = 64 Kbit/s.

6.5. Sistema MIC 30+2

La tecnología digital está constantemente disminuyendo el coste de las soluciones a los problemas de telecomunicaciones. Por razones económicas, se introdujo ampliamente la transmisión MIC (Modulación por impulsos codificados, PCM) durante los años 60. La transmisión de conversaciones entre las centrales, se logró poniendo voz analógica dentro de paquetes digitales o intervalos de tiempo, siendo necesaria la conversión nuevamente a analógica con propósitos de conmutación. El desarrollo ha hecho ahora posible y económico conmutar directamente intervalos de

tiempo de entrada al intervalo de tiempo de salida requerido. Con propósitos de encaminamiento, una conversación entrante en un cierto intervalo de tiempo de un sistema MIC necesita conectarse a un circuito de salida, otro determinado intervalo de tiempo en otro sistema MIC. El conmutador digital conmuta una muestra de voz digital en un intervalo de tiempo de entrada al intervalo de tiempo de salida escogido, con un retardo por canal constante (por lo tanto, acotado), hacia el siguiente punto en la red telefónica.

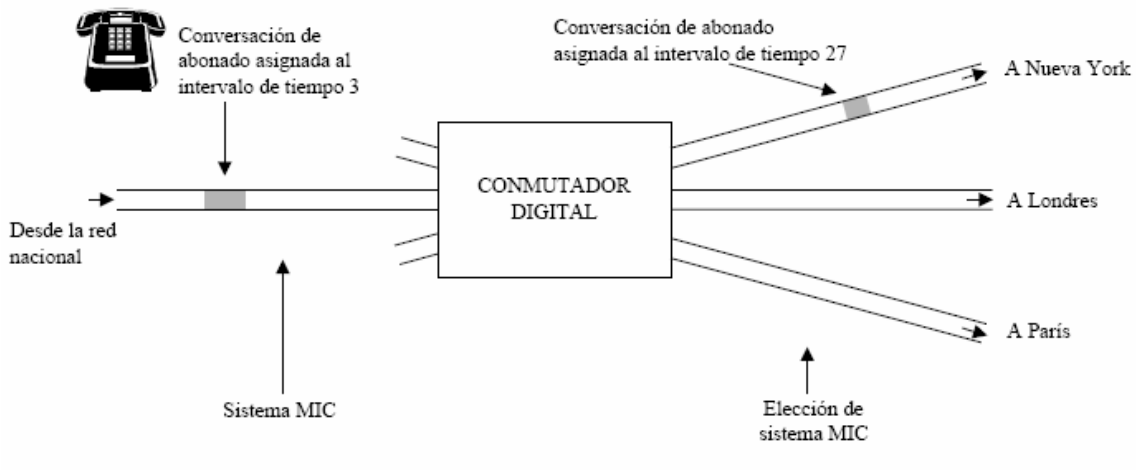


Figura 45: Conmutación digital usando sistema MIC

El abonado se va a conectar a Nueva York; un intervalo de tiempo libre ha de ser seleccionado en el sistema MIC de salida hacia Nueva York (Por ejemplo, el intervalo de tiempo 27).

Para hacer posible el interfuncionamiento de equipos de transmisión de equipos de transmisión diseñados por diferentes fabricantes, se estandarizó un conjunto de recomendaciones de transmisión que se denominan **Jerarquía Digital Plesiócrona (PDH)**.

Europa			Norte América		
Nivel Jerárquico	Número Canales	Tasa Binaria (Mbit/s)	Nivel Jerárquico	Número Canales	Tasa Binaria (Mbit/s)
E1	30	2.048	DS-1	24	1.544
E2	120	8.448	DS-1C	48	3.152
E3	480	34.368	DS-2	96	6.312
E4	1920	139.264	DS-3	672	44.736
E5	7680	565.148	DS-4	4032	374.176

En Europa cada nivel jerárquico se forma agregando la carga de cuatro enlaces del nivel jerárquico inferior.

Se diseñaron inicialmente para interconectar centrales telefónicas, por lo que se crea un canal de señalización.

También se añadió un canal de control para soportar diferentes funciones, como la de sincronismo de trama y múltiples señales de alarma.

El sistema E1 está compuesto por 30 canales de voz más 1 de sincronismo más 1 de señalización, lo que supone:

32 canales x 64 Kbit/s de cada canal = **2.048 Mbit/s**

Al sistema E1 también se le llama **MIC 30+2**, y la trama de este sistema es la siguiente:

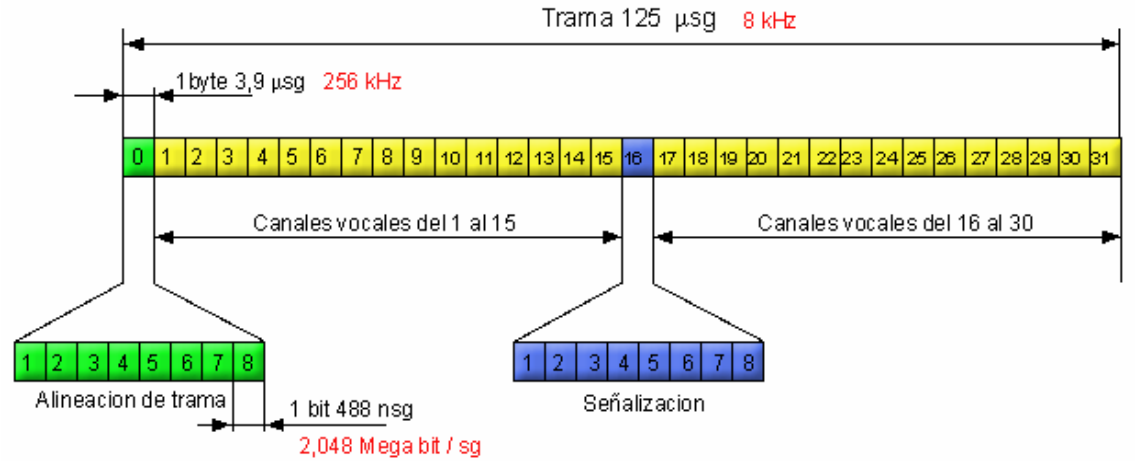


Figura 46: Trama del MIC 30+2

Se puede observar que el primer slot sirve para la alineación o sincronismo de trama (todos los slots 0 pertenecen al canal 0), y que el slot 16 sirve para la señalización (todos los slots 16 pertenecen al canal 16). El resto de slots (del 1 al 15 y del 17 al 31) pertenecen todos a canales de tráfico de abonado (voz o datos).

En el simulador desarrollado en este proyecto fin de carrera, el MIC 30+2 se puede simular en la opción B: Demostración de sincronismo y señalización, sólo que en vez de haber 30 canales vocales, sólo hay 6. Se hace de esta forma para ahorrar espacio y tiempo en la comprensión del algoritmo. Podría decirse que el sistema del simulador es una especie de “MIC 6+2”

Multiplexor MIC

La conversión de señales analógicas a transmisión MIC se lleva a cabo en el multiplexor MIC. El muestreo, la cuantificación y la codificación, se llevan a cabo de la manera normal y la salida hacia el terminal de central es un flujo de bits digital con, en el caso de la CEPT, una velocidad de bit de 2.048 Mb/s dividida en 32 intervalos de tiempo. Esto se aplica a las señales de voz de entrada; para las señales de salida la secuencia es al revés.

Terminal de central

El terminal de central tiene como propósito ordenar los intervalos de tiempo, provenientes de distintos multiplexores MIC, en fase con los intervalos de tiempo de la central. Esto se realiza mediante la amortiguación y colocación de nuevo reloj. Con el fin de optimizar la red de conmutación, se realiza frecuentemente conversión serial/en paralelo y multiplexación de varios sistemas MIC. Si, por ejemplo, ocho sistemas MIC son multiplexados y transmitidos en paralelo en un bus de 8 hilos, la frecuencia original, 2.048 Mb/s, se conserva en cada hilo. El bus transmite sin embargo, 256 los puntos cruzados divididos en el tiempo en la red de conmutación son usados más eficientemente.

6.6. Conmutación temporal

El conmutador de tiempo consiste en una memoria de voz, donde las palabras MIC son retrasadas en un número arbitrario de intervalos de tiempo (menos que una trama). La memoria de voz es controlada por una memoria de control. La escritura de la información de los intervalos de tiempo de entrada en la memoria de voz, puede ser secuencial controlada por un simple contador; intervalo de tiempo No. 1 en la celda No. 1, el No. 2 en la celda No. 2, etc., siendo la lectura de la memoria de voz realizada de manera complementaria. Esta memoria tiene tantas celdas como intervalos de tiempo haya y durante cada intervalo de tiempo ordena la lectura de una celda específica en la memoria de voz. El retardo efectivo, conmutación en el tiempo, es obviamente la diferencia de tiempo entre la escritura dentro de la memoria de voz y la lectura de la memoria, que será fijo por cada canal.

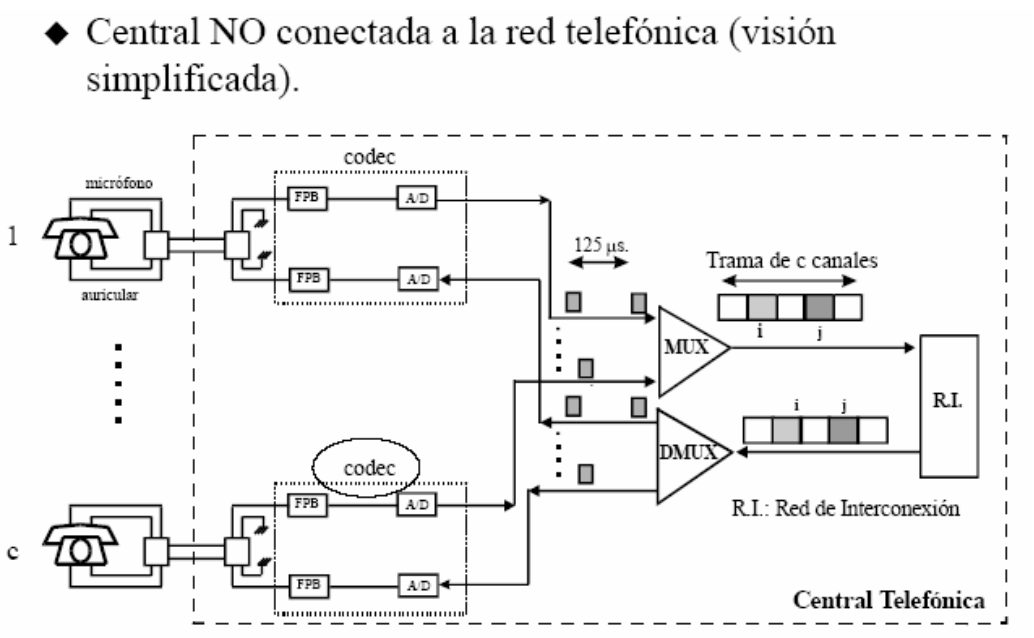


Figura 47: Ejemplo de central local digital

El intercambiador de canales (TSI) o Switch es el que realiza la conmutación propiamente dicha. Sus elementos principales son los siguientes:

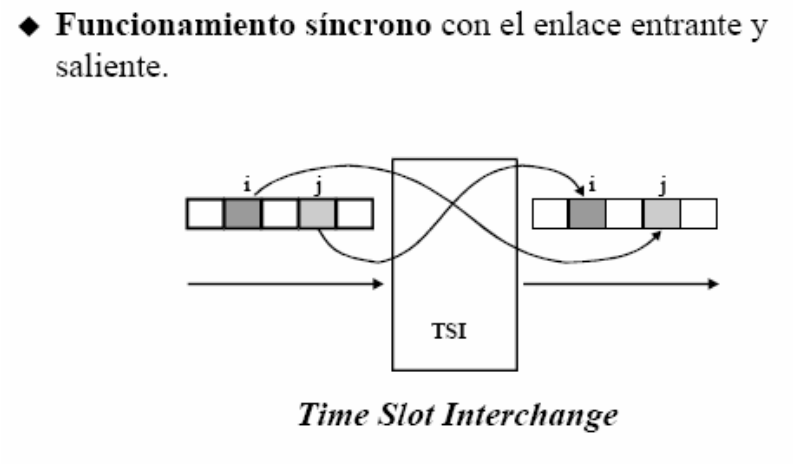


Figura 48: Intercambiador de canales o Switch

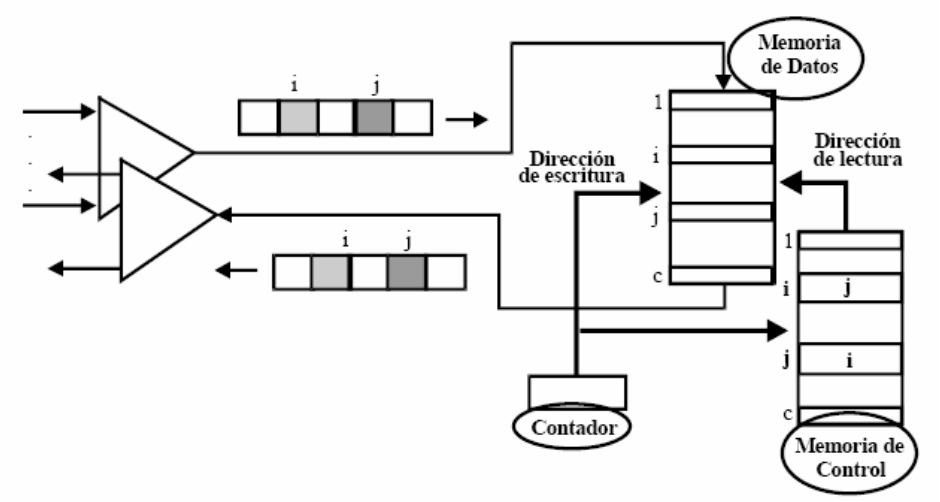


Figura 49: Elementos que componen el Switch

Hay 2 modos de funcionamiento: con Escritura Secuencial y Lectura Controlada (que es el modo de funcionamiento que se aplica en el simulador), o con Escritura Controlada y Lectura Secuencial.

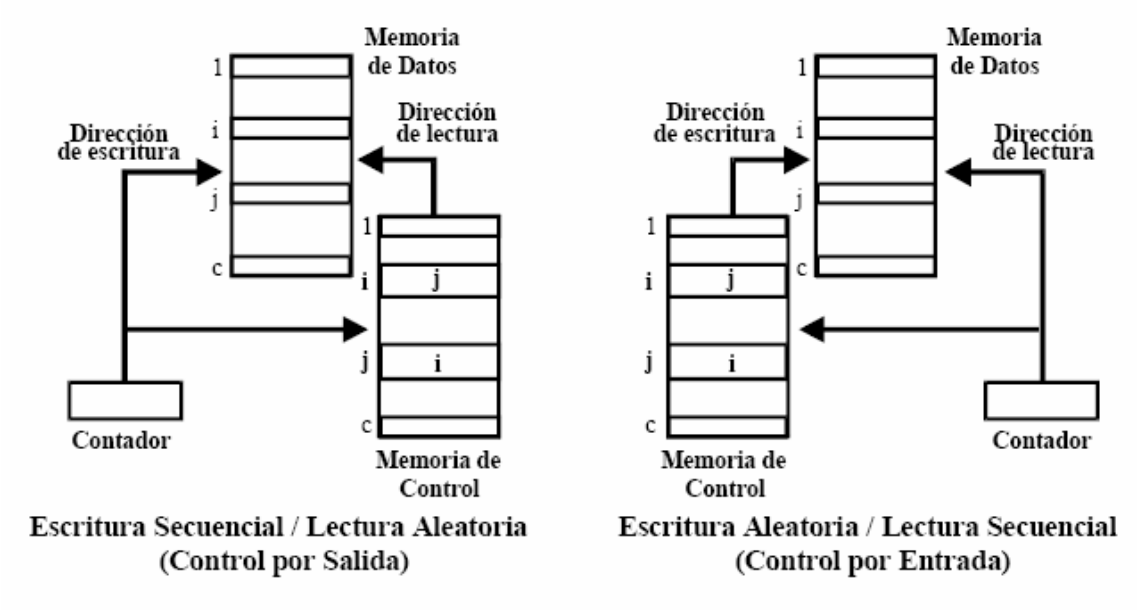


Figura 50: Distintos tipos de lectura y escritura en el switch

En el primer caso, la memoria se calcula:

$M = C \times 8 + L \times [Log_2 C]$ bits

Donde C es la longitud del MIC de entrada, 8 son los 8 bits por muestra en telefonía, y L es la longitud del MIC de salida.

En el segundo caso, la memoria se calcula:

$M = L \times 8 + C \times [Log_2 L]$ bits

6.7. Red de señalización

La red de señalización es la red de telecomunicación que da servicio a un sistema de señalización por canal común. Constituida por nodos de conmutación y proceso y por los enlaces que los interconectan.

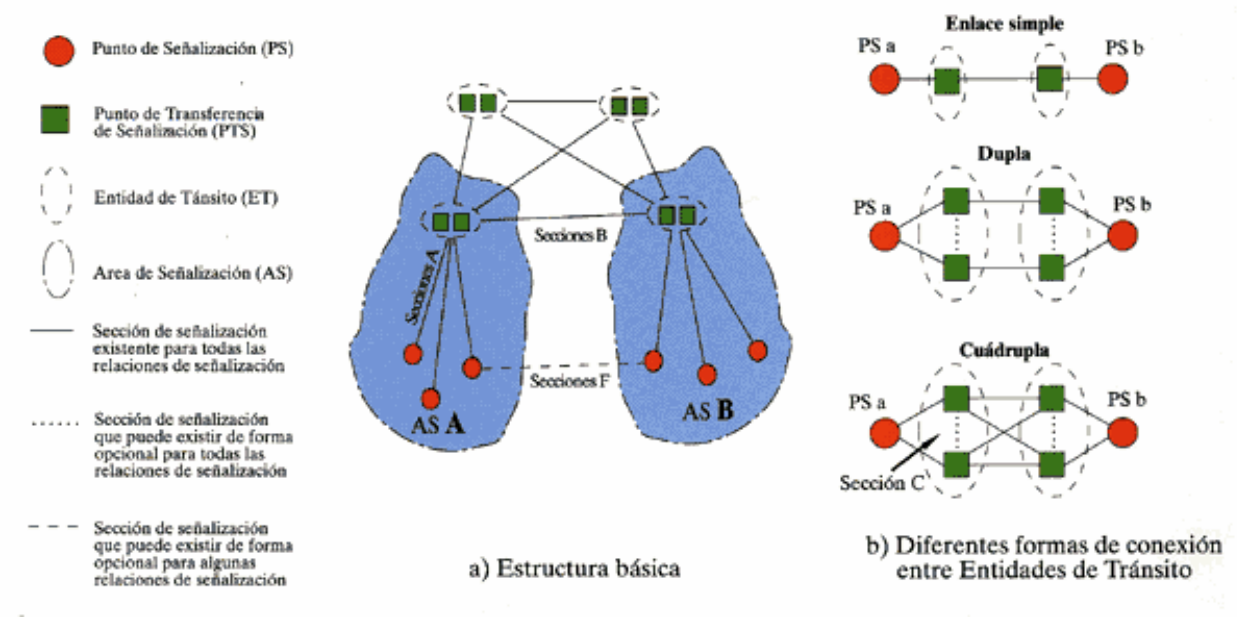


Figura 51: Estructura de una red de señalización

La señalización es todo aquello que sirve para dirigir, ordenar, monitorizar o informar. Se realiza entre abonado y central, y es interna a dicha central.

Entre elementos del sistema de transmisión:

- Repetidores, estaciones de radioenlace
- Controlar el funcionamiento y mantener la operatividad de los equipos

El señalizador sirve para transformar la naturaleza de las señales para que sean adaptadas al medio de transmisión.

Señalización por canal asociado

Un canal de voz está asociado a un canal de señalización que le es propio. La información de señalización y la de voz se transmiten por el mismo canal aunque a veces con unas bandas de frecuencia reservadas para la señalización.

En los sistemas de señalización por canal asociado el equipo de transmisión está activo por una mínima parte del tiempo total de utilización del canal (< 1%).

El equipo receptor esta funcionando permanentemente, pero su actividad es mínima.

Se propuso otro sistema de señalización para rentabilizar y tener un sistema mas eficaz que se basa en la separación entre canal de voz y la información de señalización.

Señalización por canal común

Un conjunto de canales comparten un canal para realizar la señalización del conjunto. El mensaje de señalización debe identificar al canal señalado

Señalización en un sistema MIC 30+2

Señalización asociada al canal:

- El canal 16 dispone de 8 bits separados en bloques de 4 que permiten la señalización de dos canales por trama.
- La asignación es estática

Señalización por canal común:

- El canal 16 se utiliza íntegramente para señalización y su tasa es de 64 kbps.
- Se utiliza independientemente del resto de los canales

Ventajas de la señalización por canal común

- Economía en los enlaces por supresión de los terminales de señalización.
- Aumento del vocabulario de señalización, lo que permite mas servicios y aplicaciones.
- Ganancia de velocidad en el establecimiento de llamadas.
- Aumento de la fiabilidad mediante el empleo de métodos mas eficaces de detección y corrección de errores.
- El SS7 ha sido concebido para constituir una red de señalización de multiservicio (RDSI).

El SS7 contempla dos modos: asociado y cuasi-asociado. El modo cuasi-asociado es un caso particular del no asociado en el que el camino de los mensajes de señalización esta predeterminado y es único.

SS7 contempla el caso de múltiples trayectos de señalización. Siempre que el reparto de carga esté totalmente definido en función de los enlaces de conversación utilizados.

Este nuevo concepto de señalización da lugar a la existencia de redes de señalización superpuestas a las de información del abonado, cuya estructura puede coincidir o no.

Capítulo 7
Conclusiones y líneas futuras

La principal conclusión es que en este proyecto fin de carrera se han logrado todos los objetivos que inicialmente estaban marcados respecto al programa timswit.

A modo de resumen, los objetivos alcanzados son:

- Se ha construido un simulador de conmutación temporal, ya sea con multiplexación por división en el tiempo simple, o con una etapa T de por medio.

- En la opción TDM simple se han añadido 3 apartados más, para poder hacer un TDM con 8, 16 y 32 canales, de forma que se acerca al simulador aún más a la realidad.

- La interfaz gráfica se ha mejorado, ya sea con paneles propios de java (mediante código), o con imágenes JPG, con lo que se ha hecho el entorno visual más agradable y más didáctico.

- En todas las opciones se les ha añadido la marcha atrás, muy útil para la docencia, ya que durante la simulación es necesario rectificar muchas veces en ciertos pasos para darse cuenta del funcionamiento de los algoritmos. Antes se tenía que volver a reiniciar todo de nuevo para volver a un paso dado.

- El código fuente se deja abierto disponible, para dejar el programa abierto a posibles mejoras, y a la depuración de posibles errores.

- Se ha conseguido el objetivo de la portabilidad, ya que el programa al estar hecho en java, se puede ejecutar en cualquier plataforma (Windows, LINUX, Macintosh, Solaris, etc).

- Un objetivo adicional ha sido crear una ayuda en el programa, para que cuando el alumno no sepa para qué sirven ciertos componentes del simulador, pueda recurrir a ella con sólo pinchar en dicho componente (tras activar el botón de la ayuda). Con esto se aumenta el valor didáctico del simulador.

Las líneas futuras para este programa dependerán de las nuevas necesidades que se encuentren a lo largo de las prácticas donde se utilice este simulador. En las prácticas de Conmutación de 2º curso de Telemática, profesores y alumnos verán qué mejoras se podrían hacer en función de las necesidades didácticas, completando el ciclo de vida del software.

Capítulo 8
Pasos para realizar un archivo “.jar”

Los archivos “.java” que contienen el código, al compilarlos generan los archivos “.class” que son los hacen funcionar la aplicación. Para tener juntas estas clases y las imágenes del programa, lo mejor es empaquetarlo todo en un archivo “.jar”, en el cual sólo hay que hacer doble click (como ejecutable que es) para que se inicie la aplicación.

Los pasos para crear dicho archivo son los siguientes:

1. Entrar en propiedades de “Mi PC”, en “Opciones Avanzadas → Variables de entorno”, y crear una variable llamada Path, donde su valor será el directorio bin del JDK que tengamos instalado.

2. Editar el archivo MANIFEST.MF (adjunto con el CD del proyecto) e indicar cuál será la clase principal (main class) del programa. Por ejemplo:

   ```
   Manifest-Version: 1.0
   Main-Class: General
   ```

3. Editar el archivo Comandos-jar.bat (también adjunto en el CD) e indicar cómo se llamará el archivo “.jar” que se desea construir, los archivos “.class” y las imágenes que se incluirán. La forma de editarlo es la siguiente:

   ```
   jar cmvf manifest.mf Simutemp.jar *.class *.JPG
   ```

   El asterisco en .class y .JPG quiere decir que se incluirán todos los archivos que tengan esta extensión.

4. Finalmente, hacer doble click en el archivo Comandos-jar.bat. Debe estar absolutamente todo en una misma carpeta, sino el archivo .bat no lo incluirá.

   Después de hacer doble click, el archivo “.jar” se encuentra construido en la misma carpeta donde estamos, listo para funcionar.

Recomendaciones:

- Tener instalado el JDK 1.4.2 o superior

- Si en el sistema está instalado el Winrar, desasociarlo con las extensiones “.jar”, ya que si no lo hacemos, el archivo “.jar” en vez de ejecutarse, se abre para explorar lo que hay dentro, es decir, el sistema lo trata como un archivo comprimido a explorar, no como un ejecutable.

Anexos

Anexo A
Manual de usuario

A.1. Introducción

El programa ha sido creado mediante java, por lo que en los ordenadores donde se utilice se deberá tener instalado el JRE (Java Runtime Environment) para que pueda ejecutarse, y éste incluye la Java Virtual Machine (JVM) y la API. Todo está incluido en el J2SE (Java 2 Standard Edition), donde puede instalarse cada cosa por separado.

Si lo que se desea es modificar el código fuente del programa, no es suficiente tener instalado el JRE, sino que se necesita el J2SE.

El software está disponible gratuitamente en http://java.sun.com/

La aplicación consiste en un único archivo con extensión ".jar" (Simutemp.jar), donde se encuentran las clases y las imágenes ".JPG" que componen el programa.

Se recomienda utilizarlo en Windows XP, ya que el diseño del simulador se ha hecho pensando en este sistema operativo, en cuanto a botones y aspecto gráfico. Utilizándolo en otros sistemas operativos como Linux varía un poco la forma de los botones y las etiquetas, que puede que no aparezcan completas, pero la funcionalidad y casi la totalidad de la presentación gráfica están también presentes en Linux.

A.2. Objetivos

El objetivo final de esta práctica es que el alumno entienda el funcionamiento de los conmutadores digitales basados en la multiplexación por división en el tiempo, en particular el multiplexado por división en el tiempo síncrono (STDM, *Synchronous Time Division Multiplex*), y el funcionamiento de las etapas de conmutación temporales (etapas T). El método utilizado será progresar en la comprensión de las distintas partes implicadas hasta alcanzar un nivel de complejidad similar al sistema MIC 30+2 utilizado hoy día en conmutación de circuitos digitales.

A.3. Introducción

El hecho de disponer de un canal con un cierto ancho de banda y una serie de comunicaciones a llevar a cabo obliga a la repartición del recurso canal. Existen diferentes técnicas para la repartición de dicho recurso, entre ellas la multiplexación por división en el tiempo (TDM, *Time Division Multiplex*). Ésta consiste en dividir el tiempo en distintas ranuras (*slots*) de tiempo, que pueden ser utilizadas por las distintas comunicaciones. Si además se habla de comunicación síncrona, dichas ranuras estarán confinadas dentro de intervalos regulares de tiempo en una trama que se repite periódicamente.

En este entorno, que se analizará en detalle en la Opción A, aparece la conmutación temporal, consistente en una reordenación temporal de las ranuras. Así, las tramas (agrupaciones de ranuras) a la salida de un conmutador temporal aparecen con sus ranuras reordenadas en el tiempo, quedando la conmutación realizada.

Finalmente, para que un conmutador síncrono funcione correctamente, éste debe saber en cada momento hacia dónde conmutar una ranura. Es decir, aparece la necesidad de sincronismo y señalización, conceptos que se trasladan a ranuras temporales y que se verán en detalle en la Opción B: Demostración de sincronismo y señalización.

A.4. Menú inicial

El programa se inicia con un menú del tipo JFrame, igual que todas las ventanas del programa, que son de tamaño fijo, y se pueden minimizar y cerrar sin ningún problema.

El menú tiene el siguiente aspecto:

Figura 52: Menú principal de SIMUTEMP

Pulsando la Opción A: Multiplexación por división en el Tiempo (TDM) se abre un submenú dentro del menú inicial, desapareciendo los botones iniciales y apareciendo a la izquierda los 4 botones de las 4 opciones del TDM simple:

Figura 53: Botones de la Opción A

Pulsando la Opción B: Conmutación temporal con TDM, el mismo caso, desaparecen los botones iniciales y aparecen a la derecha los botones de las 2 posibles alternativas dentro de la Opción B:

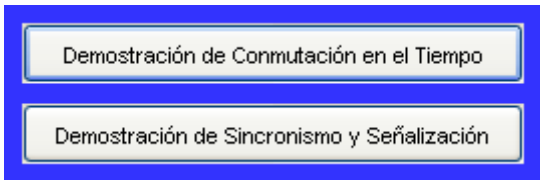


Figura 54: Botones de la Opción B

A.5. Opción A: Multiplexación por división en el tiempo

A lo largo de esta opción iremos viendo cómo se realiza una multiplexación por división en el tiempo. Cuatro son las opciones de las que se dispone, todas funcionan igual, con la salvedad de que todas tienen distinto número de canales.

El aspecto de un TDM con 4 canales es el siguiente:

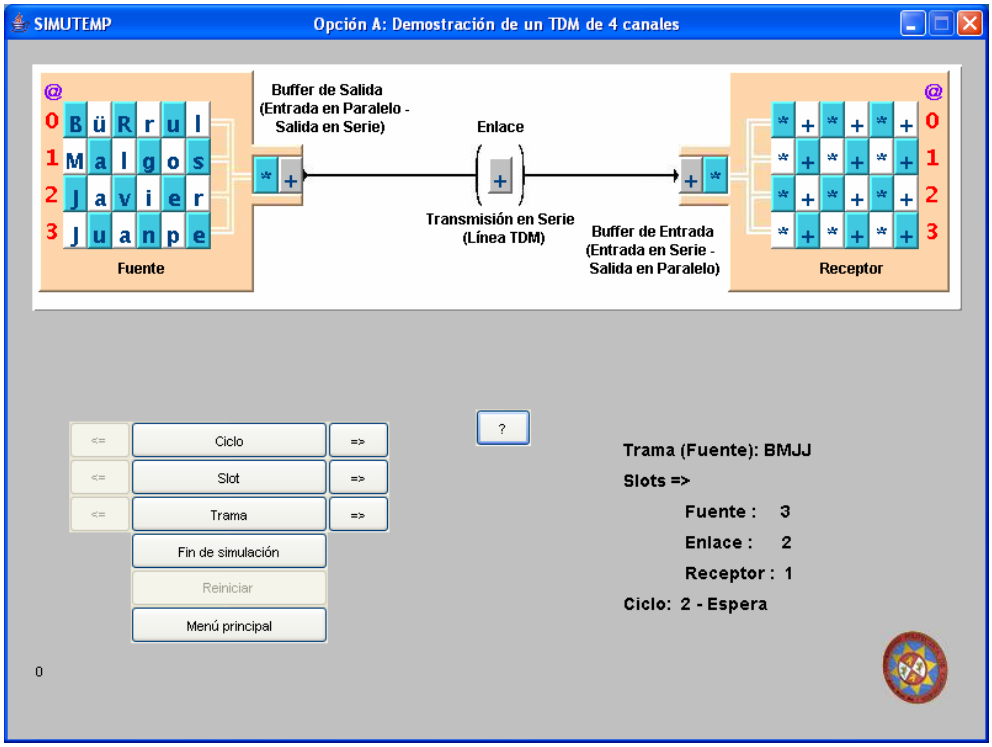


Figura 55: Opción A: TDM de 4 canales

- **Dibujo del sistema:** Podemos observar el módulo fuente, la línea de transmisión y el módulo receptor.

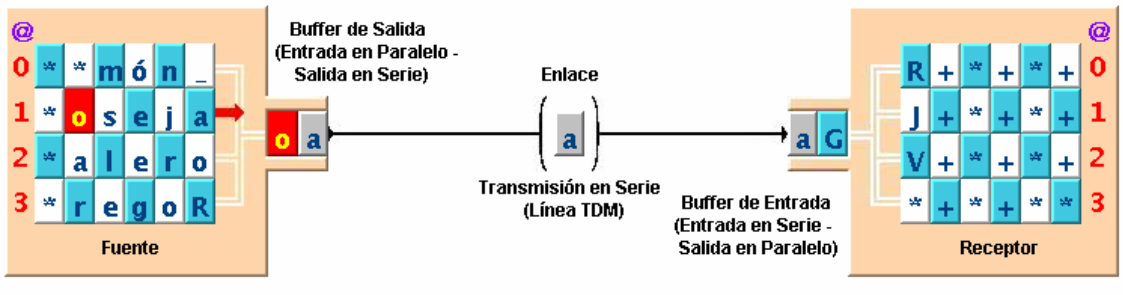


Figura 56: Módulos fuente y receptor, y línea de transmisión

- **Estado de la simulación:** Aquí se puede visualizar la trama que en ese momento se está transmitiendo, y también se puede saber qué slots de dicha trama están ocupando el buffer de salida de la fuente, el enlace, y el buffer de entrada del receptor, que es al fin y al cabo, saber de qué canales son dichas letras. En el registro "Ciclo" sirve para ver si nos encontramos en un ciclo de Salida (donde el dato de la fuente se deposita en el buffer de salida y la línea se desplaza), un ciclo de Entrada (donde el dato que hay en el buffer de entrada pasa a ocupar su lugar correspondiente en el receptor), o un cilo de Espera (Donde no se hace nada, sólo simula el retardo que hay en este tipo de transmisión).

Figura 57: Cuadro de estado de la simulación

- **Cuadro de botones:** Aquí es donde el alumno interactuará con el simulador, pudiendo avanzar en la transmisión un ciclo con el primer botón " => ", o bien con "Ciclo", que tienen la misma funcionalidad, esto también es aplicable al avanzar un slot y una trama, botones que se encuentran debajo. Los botones de la izquierda sirven para rectificar un paso en la simulación. Se puede dar un ciclo atrás, un slot atrás y una trama atrás. El botón "Fin de simulación" sirve para llegar al estado final de la simulación con toda la información de voz transmitida. Con el botón "Reiniciar" todo volverá al estado inicial, y también hay un botón para volver al menú principal.

Cuando no tiene sentido dar cierto paso, los botones que permitirían ese paso están deshabilitados, como los de marcha atrás, que al principio de la simulación están deshabilitados porque no tiene sentido dar marcha atrás cuando aún no se ha empezado.

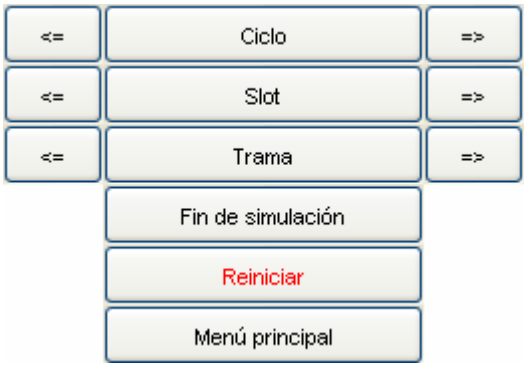


Figura 58: Cuadro de botones

- **Botón de ayuda:** Cuando se pulsa, podemos hacer click en cualquier elemento del simulador, y aparecerá un cuadro de diálogo explicando para qué sirve dicho elemento. El programa tiene una ayuda para utilizar cuando se necesite.

Figura 58: Botón de ayuda

A.6. Opción B: Conmutación temporal con TDM

Aquí veremos cómo los datos de voz son encaminados según sus destinos por una etapa T, es decir, por un switch conmutador, utilizando la ya conocida técnica de TDM.

A.6.1. Demostración de conmutación en el tiempo

Éste es el aspecto de una conmutación sencilla con TDM:

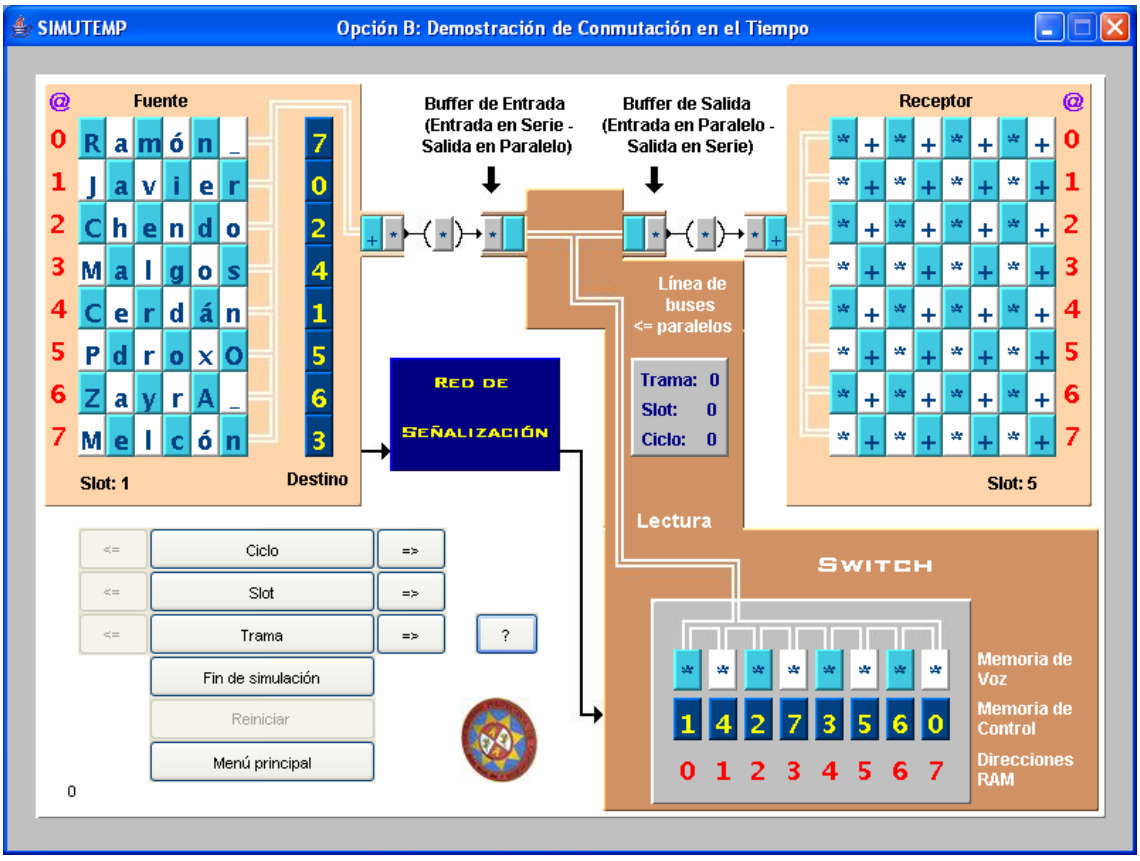


Figura 59: Opción B: Demostración de conmutación en el tiempo

El cuadro de botones hace exactamente lo mismo que en las anteriores opciones.

- **Indicador de ciclo:** En esta opción, los ciclos son Lectura, Escritura y Espera, se puede ver en la parte media del switch.

- **Memorias:** En este panel del switch se muestran de arriba abajo la memoria de voz, la memoria de control y las direcciones RAM:

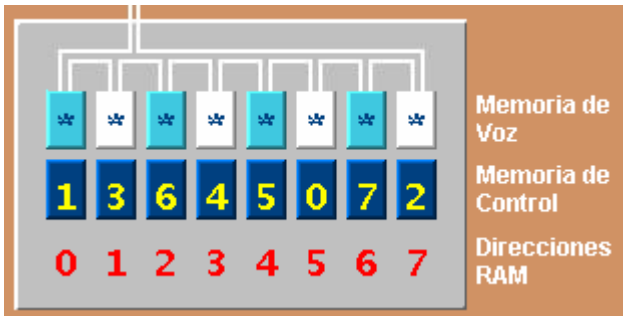


Figura 60: Panel de memorias del Switch

A.6.2. Demostración de sincronismo y señalización

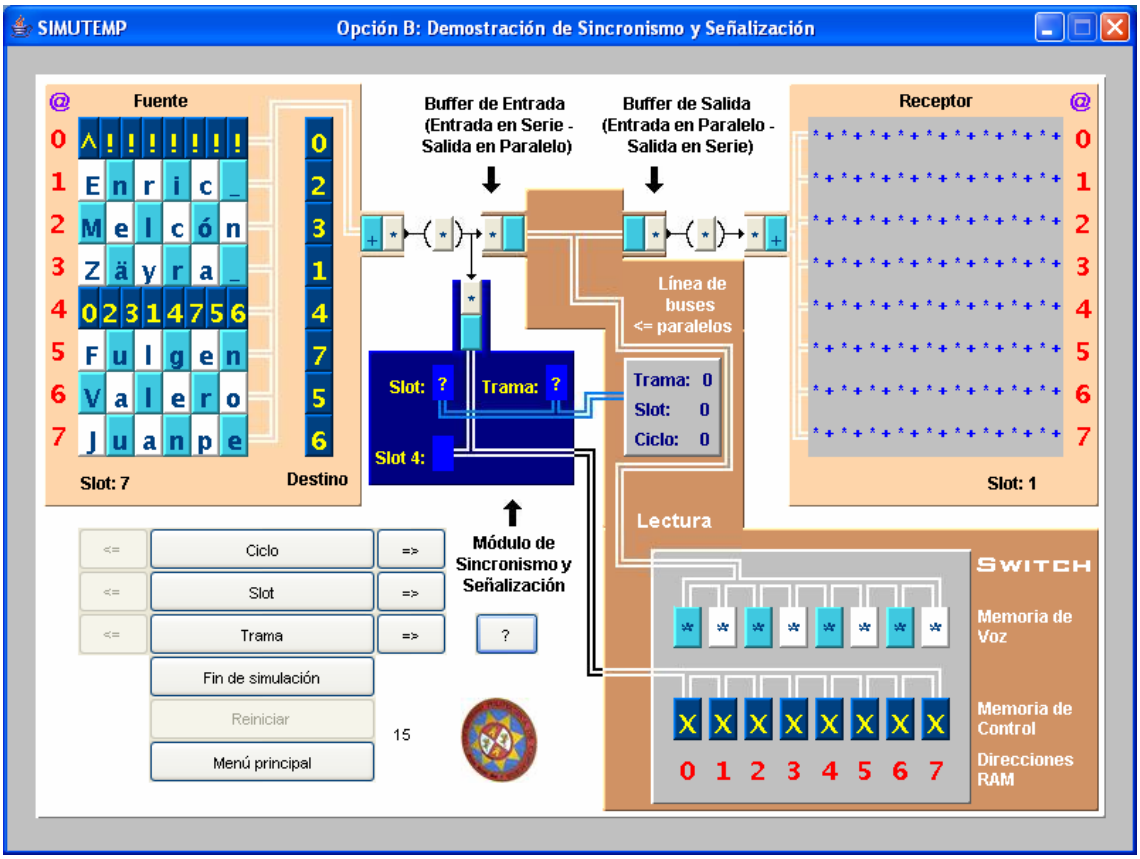


Figura 61: Opción B: Demostración de sincronismo y señalización

- **Memorias:** En este panel del switch se muestran de arriba abajo la memoria de voz, la memoria de control y las direcciones RAM, pero tiene distinto aspecto del switch de la opción anterior, ya que la memoria de

control aún no está definida, y está conectada al módulo de sincronismo y señalización.

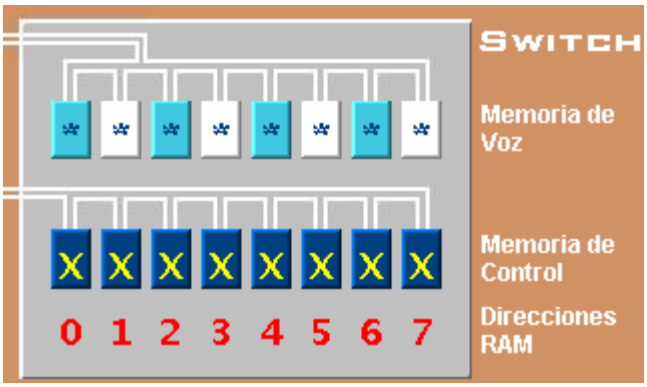


Figura 62: Panel de memorias del Switch con la memoria de control sin definir

- **Módulo de sincronismo y señalización:** El alumno tendrá que guiarse por él durante la simulación, ya que es el responsable de que el sistema se sincronice y de rellenar la memoria de control:

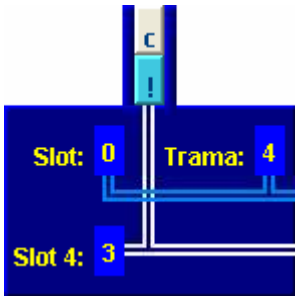


Figura 63: Módulo de sincronismo y señalización

- **Registros contadores de slots:** Para saber si el sistema está sincronizado o no, hay que mirar dichos registros en fuente y receptor:

Figura 64: Registro contador de slots del receptor

Bibliografía

J.E. Flood, “Telecommunications Switching, Traffic and Networks”, Editorial Prentice Hall, 1995, ISBN: 0-13-033309-3.

J.Bellamy, “Digital Telephony”, Editorial John Wiley & Sons, 1991 (2ª Edición), ISBN: 84-311-0576-3.

Transparencias de D. Jorge Martínez Bauset, Departamento de Comunicaciones de la Universidad Politécnica de Valencia.

Apuntes de clase de la asignatura de Conmutación, de 2º curso de Ingeniería Telemática de la Universidad Politécnica de Cartagena.

Apuntes de clase de la asignatura de Redes y Servicios de Comunicaciones, de 2º curso de Ingeniería Telemática de la Universidad Politécnica de Cartagena.

Apuntes de clase de la asignatura de Fundamentos de la Programación, de 2º curso de Ingeniería Telemática de la Universidad Politécnica de Cartagena.

Anexo B
Código fuente más significativo

El código fuente de la aplicación está disponible en el CD-ROM del proyecto. Sin embargo, aquí se escribe parte del código fuente de la aplicación que puede resultar más interesante.

B.1. OpcionA4

```
	import java.lang.String;
	import java.awt.*;
	import javax.swing.*;

	public class OpcionA4 extends javax.swing.JFrame
	{
		String ciclo = "2 - Espera"; //esto es lo que pondré en la
etiqueta
					//ciclo, o sea, 0 - Salida, 1 - Entrada, 2
- Espera

		boolean habilitado = true; //para que cuando se llegue al final
									//de la simulación y se
siga dándole
									//al botón, no haga
nada, evitando
									//así la aparición de
excepciones
		boolean trama1 = false;
		boolean trama2 = false;		//se ponen a true en el momento en
el
		boolean trama3 = false;		//que se transmite la primera letra
		boolean trama4 = false;		//de la trama, ésto me será útil
para
		boolean trama5 = false;		//visualizar en el momento justo la
		boolean tramaFin = false;  //trama que estoy transmitiendo

		boolean primera_vez = true;

		boolean asistente = false;

		int puntero = 0; //cuenta las posiciones dentro de
					//la misma palabra, y va
incrementándose
					//cada vez que se transmiten las 4
letras
					//de la posición puntero de cada
palabra,
					//es la encargada de elegir la letra a
					//transmitir, cualquiera que sea el
canal

		boolean residuo = false; //indicador de que pasamos a la
		//siguiente iteración de puntero pero aún quedan letras
		//colgando del anterior valor de puntero

		boolean fin = false; //indicador de que puntero ha alcanzado
		//su valor máximo, luego no seguiré tomando letras de p y
		//sólo tendré que terminar la transmisión de las letras que
		//se han quedado en la línea

		int iteración = 0; //será la variable principal del programa
					//es la encargada de decidir en qué
canal
					//busco la letra a transmitir, y en
qué
					//canal dejo la letra transmitida,
de cómo
					//cambia la línea, etc. El programa
en
					//general, se guía por esta
variable, que
					//se pone a 0 cuando cambia el valor
de puntero


		int índice = 0; //será la referencia que seguirá el método
```

```
                                        //guardarEstado()   y   asignarEstado(),
para
                                        //los arrays
            int tirada = 0;

            public OpcionA4()
            {
                  iniciarAleatorio();
             iniciarComponentes();
                  setSize(800, 600);
                  setResizable(false);
                  grabarEstados();

                  try
                  {

      UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAnd
Feel");
                        SwingUtilities.updateComponentTreeUI(this);
                  }
                  catch (Exception e){}
            }

            private       javax.swing.JButton       button[]       =       new
javax.swing.JButton[13];
            private JLabel buffer[] = new JLabel[4];
            private javax.swing.JLabel texto[] = new javax.swing.JLabel[15];

            private   javax.swing.JLabel   medio1,   medio2,   medio3,   medio4,
medio5;
            private javax.swing.JLabel ayudado, escudin, arroba1, arroba2;

            private javax.swing.JLabel índiceTexto;
            private javax.swing.JLabel texto16[] = new javax.swing.JLabel[6];
            private javax.swing.JLabel texto17[] = new javax.swing.JLabel[6];
            private javax.swing.JLabel texto18[] = new javax.swing.JLabel[6];
            private javax.swing.JLabel texto19[] = new javax.swing.JLabel[6];
            private javax.swing.JLabel texto20[] = new javax.swing.JLabel[6];
            private javax.swing.JLabel texto21[] = new javax.swing.JLabel[6];
            private javax.swing.JLabel texto22[] = new javax.swing.JLabel[6];
            private javax.swing.JLabel texto23[] = new javax.swing.JLabel[6];
            JLabel flechaIzq[] = new JLabel[4];
            JLabel flechaDer[] = new JLabel[4];

            private javax.swing.JLabel num[] = new javax.swing.JLabel[8];
            private javax.swing.JLabel letras[] = new javax.swing.JLabel[14];

            private javax.swing.JPanel panel[] = new javax.swing.JPanel[14];
            private javax.swing.JPanel caja1, caja2;
            private javax.swing.JPanel fichas[] = new javax.swing.JPanel[54];

          private void iniciarComponentes() {

                  for(int i=0;i<13;i++)
                  {
                        button[i] = new javax.swing.JButton();
                  }


                  for(int i=0;i<15;i++)
                  {
                        texto[i] = new javax.swing.JLabel();
                  }

                  ayudado = new javax.swing.JLabel();
                  arroba1 = new javax.swing.JLabel();
                  arroba2 = new javax.swing.JLabel();
```

```
                    índiceTexto = new javax.swing.JLabel();

                    for(int i=0;i<14;i++)
                    {
                          panel[i] = new javax.swing.JPanel();
                    }


                    caja1 = new javax.swing.JPanel();
                    caja2 = new javax.swing.JPanel();

                    for(int i=0;i<54;i++)
                    {
                          fichas[i] = new javax.swing.JPanel();
                    }



                    //Estos JLabels() visualizarán los valores de cada una
                    //de las letras en los canales de la fuente y de recepción:

                    for(int        i=0;i<6;i++){texto16[i]        =        new
javax.swing.JLabel();}
                    for(int        i=0;i<6;i++){texto17[i]        =        new
javax.swing.JLabel();}
                    for(int        i=0;i<6;i++){texto18[i]        =        new
javax.swing.JLabel();}
                    for(int        i=0;i<6;i++){texto19[i]        =        new
javax.swing.JLabel();}
                    for(int        i=0;i<6;i++){texto20[i]        =        new
javax.swing.JLabel();}
                    for(int        i=0;i<6;i++){texto21[i]        =        new
javax.swing.JLabel();}
                    for(int        i=0;i<6;i++){texto22[i]        =        new
javax.swing.JLabel();}
                    for(int        i=0;i<6;i++){texto23[i]        =        new
javax.swing.JLabel();}


                    for(int i=0;i<8;i++){num[i] = new javax.swing.JLabel();}
                    for(int        i=0;i<14;i++){letras[i]        =        new
javax.swing.JLabel();}

                    //Estos JLabels son los que visualizarán los valores que
                    //tome la línea de transmisión:

                    medio1 = new javax.swing.JLabel();
                    medio2 = new javax.swing.JLabel();
                    medio3 = new javax.swing.JLabel();
                    medio4 = new javax.swing.JLabel();
                    medio5 = new javax.swing.JLabel();

               getContentPane().setLayout(null);
                    getContentPane().setBackground(new java.awt.Color(192, 192,
192));
                    setTitle("SIMUTEMP
Opción A: Demostración de un TDM de 4 canales");
               addWindowListener(new java.awt.event.WindowAdapter() {
                 public void windowClosing(java.awt.event.WindowEvent evt) {
                       exitForm(evt);
                   }
               });


               button[0].setText("Ciclo");
                    button[0].addActionListener(new
java.awt.event.ActionListener() {
```

```
                    public void actionPerformed(java.awt.event.ActionEvent evt)
{
                        cicloHandler(evt);
                    }
                });


                getContentPane().add(button[0]);
                button[0].setBounds(100, 310, 160, 30);

                button[1].setText("Slot");
                     button[1].addActionListener(new
java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt)
{
                        timeslotHandler(evt);
                    }
                });

                getContentPane().add(button[1]);
                button[1].setBounds(100, 340, 160, 30);

                button[2].setText("Trama");
                     button[2].addActionListener(new
java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt)
{
                        tramaHandler(evt);
                    }
                });

                getContentPane().add(button[2]);
                button[2].setBounds(100, 370, 160, 30);

                button[3].setText("Fin de simulación");
                     button[3].addActionListener(new
java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt)
{
                        finHandler(evt);
                    }
                });

                getContentPane().add(button[3]);
                button[3].setBounds(100, 400, 160, 30);

                button[4].setText("Reiniciar");
                     button[4].setForeground(Color.red);
                     button[4].addActionListener(new
java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt)
{
                        reiniciarHandler(evt);
                    }
                });

                getContentPane().add(button[4]);
                button[4].setBounds(100, 430, 160, 30);

                button[5].setText("Menú principal");
                     button[5].addActionListener(new
java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt)
{
                        menuHandler(evt);
                    }
                });
```

```
                getContentPane().add(button[5]);
                button[5].setBounds(100, 460, 160, 30);


                     button[6].setText("<=");
                     button[6].addActionListener(new
java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt)
{
                        atrásCicloHandler(evt);
                    }
                });

                getContentPane().add(button[6]);
                button[6].setBounds(50, 310, 50, 30);

                     button[7].setText("<=");
                     button[7].addActionListener(new
java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt)
{
                        atrásSlotHandler(evt);
                    }
                });

                getContentPane().add(button[7]);
                button[7].setBounds(50, 340, 50, 30);


                     button[8].setText("<=");
                     button[8].addActionListener(new
java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt)
{
                        atrásTramaHandler(evt);
                    }
                });

                getContentPane().add(button[8]);
                button[8].setBounds(50, 370, 50, 30);

                     button[9].setText("=>");
                     button[9].addActionListener(new
java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt)
{
                        cicloHandler(evt);
                    }
                });

                getContentPane().add(button[9]);
                button[9].setBounds(260, 310, 50, 30);

                     button[10].setText("=>");
                     button[10].addActionListener(new
java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt)
{
                        timeslotHandler(evt);
                    }
                });

                getContentPane().add(button[10]);
                button[10].setBounds(260, 340, 50, 30);

                     button[11].setText("=>");
                     button[11].addActionListener(new
java.awt.event.ActionListener() {
```

```
                    public void actionPerformed(java.awt.event.ActionEvent evt)
{
                        tramaHandler(evt);
                    }
                });

                getContentPane().add(button[11]);
                button[11].setBounds(260, 370, 50, 30);

                     button[12].setText("?");
                     button[12].addActionListener(new
java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt)
{
                        ayudaHandler(evt);
                    }
                });

                     getContentPane().add(button[12]);
                     button[12].setBounds(380, 300, 45, 30);

                     caja1.addMouseListener(new java.awt.event.MouseAdapter() {
                    public void mouseClicked(java.awt.event.MouseEvent evt) {
                        ayudaFuente(evt);
                    }
                });

                     caja2.addMouseListener(new java.awt.event.MouseAdapter() {
                    public void mouseClicked(java.awt.event.MouseEvent evt) {
                        ayudaReceptor(evt);
                    }
                });

                     for(int i=0;i<24;i++)
                     {
                            fichas[i].addMouseListener(new
java.awt.event.MouseAdapter() {
                             public  void  mouseClicked(java.awt.event.MouseEvent
evt) {
                                    canalVoz(evt);
                             }
                     });
                     }

                     for(int i=24;i<48;i++)
                     {
                            fichas[i].addMouseListener(new
java.awt.event.MouseAdapter() {
                             public  void  mouseClicked(java.awt.event.MouseEvent
evt) {
                                    canalVozReceptor(evt);
                             }
                     });
                     }

                     for(int i=8;i<14;i++)
                     {
                            texto[i].addMouseListener(new
java.awt.event.MouseAdapter() {
                             public  void  mouseClicked(java.awt.event.MouseEvent
evt) {
                                    ayudaEstado(evt);
                             }
                     });
                     }

                     num[0].setText("0");
                     num[0].reshape(10, 30, 30, 30);
```

```
num[1].setText("1");
num[1].reshape(10, 60, 30, 30);

num[2].setText("2");
num[2].reshape(10, 90, 30, 30);

num[3].setText("3");
num[3].reshape(10, 120, 30, 30);

num[4].setText("0");
num[4].reshape(725, 30, 30, 30);

num[5].setText("1");
num[5].reshape(725, 60, 30, 30);

num[6].setText("2");
num[6].reshape(725, 90, 30, 30);

num[7].setText("3");
num[7].reshape(725, 120, 30, 30);

for(int i=0;i<8;i++)
{
	num[i].setFont(new  java.awt.Font("Lucida  Sans",  1,
18));
	num[i].setForeground(Color.red);
	panel[0].add(num[i]);
}


for(int i=0;i<14;i++)
{
	letras[i].setFont(new   java.awt.Font("Dialog",   1,
12));
	panel[0].add(letras[i]);
}



letras[0].setText("Buffer de Salida");
letras[0].reshape(195, 5, 100, 30);

letras[1].setText("(Entrada en Paralelo -");
letras[1].reshape(185, 20, 150, 30);

letras[2].setText(" Salida en Serie)");
letras[2].reshape(195, 35, 100, 30);

letras[3].setText("Buffer de Entrada");
letras[3].reshape(455, 120, 100, 30);

letras[4].setText("(Entrada en Serie -");
letras[4].reshape(450, 135, 150, 30);

letras[5].setText(" Salida en Paralelo)");
letras[5].reshape(450, 150, 150, 30);

letras[6].setText("Enlace");
letras[6].reshape(362, 35, 100, 30);

letras[7].setText("Transmisión en Serie");
letras[7].reshape(320, 110, 150, 30);

letras[8].setText("(Línea TDM)");
letras[8].reshape(350, 125, 100, 30);
```

```
                    letras[9].setText("Fuente");
                    letras[9].reshape(70, 150, 100, 30);

                    letras[10].setText("Receptor");
                    letras[10].reshape(640, 150, 100, 30);

                    arroba1.setText("@");
                    arroba1.setFont(new java.awt.Font("Comic Sans", 1, 14));
                    arroba1.setForeground(new java.awt.Color(128, 0, 255));
                    arroba1.setBounds(10, 5, 30, 30);
                    panel[0].add(arroba1);

                    arroba2.setText("@");
                    arroba2.setFont(new java.awt.Font("Comic Sans", 1, 14));
                    arroba2.setForeground(new java.awt.Color(128, 0, 255));
                    arroba2.setBounds(725, 5, 30, 30);
                    panel[0].add(arroba2);

                    getContentPane().add(panel[0],
java.awt.BorderLayout.CENTER);
                    panel[0].setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));


                    via1             =             new             JLabel(new
ImageIcon(getClass().getResource("/A4via.JPG")));
                    via1.setBounds(220, 65, 145, 50);
                    panel[0].add(via1);

                    via2             =             new             JLabel(new
ImageIcon(getClass().getResource("/A4via2.JPG")));
                    via2.setBounds(395, 65, 130, 50);
                    panel[0].add(via2);

                    ayudado.setText("Señala  con  el  cursor  donde  necesites
ayuda");
                    ayudado.setFont(new java.awt.Font("Dialog", 1, 14));
                    ayudado.setForeground(Color.red);
                    ayudado.setBounds(245, 250, 350, 30);
                    getContentPane().add(ayudado);
                    ayudado.setVisible(false);


                    for(int i=1;i<3;i++)
                    {
                          flechaIzq[i]        =        new        JLabel(new
ImageIcon(getClass().getResource("/A4flecharojahorizontal.JPG")));
                          panel[0].add(flechaIzq[i]);
                          flechaDer[i]        =        new        JLabel(new
ImageIcon(getClass().getResource("/A4flechader.JPG")));
                          panel[0].add(flechaDer[i]);
                    }

                    flechaIzq[0]          =          new          JLabel(new
ImageIcon(getClass().getResource("/A4flecharojahorizontalarriba.JPG")));
                    flechaIzq[3]          =          new          JLabel(new
ImageIcon(getClass().getResource("/A4flecharojahorizontalabajo.JPG")));
                    panel[0].add(flechaIzq[0]);
                    panel[0].add(flechaIzq[3]);

                    flechaIzq[1].setBounds(145, 70, 19, 12);
                    flechaIzq[2].setBounds(145, 100, 19, 12);
                    flechaIzq[0].setBounds(145, 40, 19, 12);
                    flechaIzq[3].setBounds(145, 130, 19, 12);

                    flechaDer[0]          =          new          JLabel(new
ImageIcon(getClass().getResource("/A4flechaderarriba.JPG")));
```

```
                    flechaDer[3]           =           new           JLabel(new
ImageIcon(getClass().getResource("/A4flechaderabajo.JPG")));
                    panel[0].add(flechaDer[0]);
                    panel[0].add(flechaDer[3]);

                    flechaDer[1].setBounds(580, 70, 19, 12);
                    flechaDer[2].setBounds(580, 100, 19, 12);
                    flechaDer[0].setBounds(580, 40, 19, 12);
                    flechaDer[3].setBounds(580, 130, 19, 12);

                    for(int i=0;i<4;i++)
                    {
                          flechaIzq[i].setVisible(false);
                          flechaDer[i].setVisible(false);
                    }

                    paralelas1           =           new           JLabel(new
ImageIcon(getClass().getResource("/A4lineas1.JPG")));
                    paralelas1.setBounds(145, 44, 33, 94);
                    panel[0].add(paralelas1);

                    paralelas2           =           new           JLabel(new
ImageIcon(getClass().getResource("/A4lineas2.JPG")));
                    paralelas2.setBounds(567, 44, 33, 94);
                    panel[0].add(paralelas2);

                    imagen            =            new           JLabel(new
ImageIcon(getClass().getResource("/upctgris.JPG")));
                    imagen.setBounds(710, 480, 54, 60);
                    getContentPane().add(imagen);



                    panel[0].reshape(20, 20, 755, 200);
                    panel[0].setBackground(Color.white);
                    panel[0].setLayout(null);


                    for(int i=0;i<53;i++)
                    {
                          panel[0].add(fichas[i]);
                    }

                    buffer[0]            =            new           JLabel(new
ImageIcon(getClass().getResource("/buffer1.JPG")));
                    buffer[1]            =            new           JLabel(new
ImageIcon(getClass().getResource("/buffer2.JPG")));
                    buffer[2]            =            new           JLabel(new
ImageIcon(getClass().getResource("/buffer3.JPG")));
                    buffer[3]            =            new           JLabel(new
ImageIcon(getClass().getResource("/buffer4.JPG")));

                    buffer[0].setBounds(174, 65, 46, 10);
                    buffer[1].setBounds(174, 105, 46, 10);
                    buffer[2].setBounds(525, 65, 46, 10);
                    buffer[3].setBounds(525, 105, 46, 10);

                    for(int i=0;i<4;i++)
                    {
                          panel[0].add(buffer[i]);
                    }

                    panel[0].add(caja1);
                    panel[0].add(caja2);
```

```
                    caja1.setBounds(5, 5, 175, 180);
                    caja1.setBackground(new java.awt.Color(255, 213, 170));
                    caja1.setBorder(new
javax.swing.border.BevelBorder(javax.swing.border.BevelBorder.RAISED));

                    caja2.setBounds(565, 5, 180, 180);
                    caja2.setBackground(new java.awt.Color(255, 213, 170));
                    caja2.setBorder(new
javax.swing.border.BevelBorder(javax.swing.border.BevelBorder.RAISED));









                    for(int i=0;i<53;i++)
                    {
                          fichas[i].setBorder(new
javax.swing.border.BevelBorder(javax.swing.border.BevelBorder.RAISED));
                    }

                pintarLabels();

                pack();
          }

             public void pintarLabels()
             {
                    for(int i=0;i<4;i++)
                    {
                          flechaIzq[i].setVisible(false);
                          flechaDer[i].setVisible(false);
                    }

                    if (ciclo.equals("0 - Salida"))
                    {
                          flechaIzq[datos[0]].setVisible(true);
                    }
                    if (ciclo.equals("1 - Entrada"))
                    {
                          flechaDer[datos[2]].setVisible(true);
                    }

                    for(int i=0;i<6;i++)
                    {
                          texto16[i].setText(p[0].valueOf(p[0].charAt(i)));
                    }
                    for(int i=0;i<6;i++)
                    {
                          texto17[i].setText(p[1].valueOf(p[1].charAt(i)));
                    }
                    for(int i=0;i<6;i++)
                    {
                          texto18[i].setText(p[2].valueOf(p[2].charAt(i)));
                    }
                    for(int i=0;i<6;i++)
                    {
                          texto19[i].setText(p[3].valueOf(p[3].charAt(i)));
                    }


                    if (q[0].charAt(0) == '¿'){texto20[0].setText("*");}
                    else{texto20[0].setText(q[0].valueOf(q[0].charAt(0)));}


                    if (q[0].charAt(1) == '&'){texto20[1].setText("+");}
                    else{texto20[1].setText(q[0].valueOf(q[0].charAt(1)));}
```

```
if (q[0].charAt(2) == '@'){texto20[2].setText("*");}
else{texto20[2].setText(q[0].valueOf(q[0].charAt(2)));}


if (q[0].charAt(3) == '#'){texto20[3].setText("+");}
else{texto20[3].setText(q[0].valueOf(q[0].charAt(3)));}


if (q[0].charAt(4) == '$'){texto20[4].setText("*");}
else{texto20[4].setText(q[0].valueOf(q[0].charAt(4)));}


if (q[0].charAt(5) == '?'){texto20[5].setText("+");}
else{texto20[5].setText(q[0].valueOf(q[0].charAt(5)));}


if (q[1].charAt(0) == '¿'){texto21[0].setText("*");}
else{texto21[0].setText(q[1].valueOf(q[1].charAt(0)));}


if (q[1].charAt(1) == '&'){texto21[1].setText("+");}
else{texto21[1].setText(q[1].valueOf(q[1].charAt(1)));}


if (q[1].charAt(2) == '@'){texto21[2].setText("*");}
else{texto21[2].setText(q[1].valueOf(q[1].charAt(2)));}


if (q[1].charAt(3) == '#'){texto21[3].setText("+");}
else{texto21[3].setText(q[1].valueOf(q[1].charAt(3)));}


if (q[1].charAt(4) == '$'){texto21[4].setText("*");}
else{texto21[4].setText(q[1].valueOf(q[1].charAt(4)));}


if (q[1].charAt(5) == '?'){texto21[5].setText("+");}
else{texto21[5].setText(q[1].valueOf(q[1].charAt(5)));}


if (q[2].charAt(0) == '¿'){texto22[0].setText("*");}
else{texto22[0].setText(q[2].valueOf(q[2].charAt(0)));}


if (q[2].charAt(1) == '&'){texto22[1].setText("+");}
else{texto22[1].setText(q[2].valueOf(q[2].charAt(1)));}


if (q[2].charAt(2) == '@'){texto22[2].setText("*");}
else{texto22[2].setText(q[2].valueOf(q[2].charAt(2)));}


if (q[2].charAt(3) == '#'){texto22[3].setText("+");}
else{texto22[3].setText(q[2].valueOf(q[2].charAt(3)));}


if (q[2].charAt(4) == '$'){texto22[4].setText("*");}
else{texto22[4].setText(q[2].valueOf(q[2].charAt(4)));}


if (q[2].charAt(5) == '?'){texto22[5].setText("+");}
else{texto22[5].setText(q[2].valueOf(q[2].charAt(5)));}


if (q[3].charAt(0) == '¿'){texto23[0].setText("*");}
else{texto23[0].setText(q[3].valueOf(q[3].charAt(0)));}
```

```
                    if (q[3].charAt(1) == '&'){texto23[1].setText("+");}
                    else{texto23[1].setText(q[3].valueOf(q[3].charAt(1)));}


                    if (q[3].charAt(2) == '@'){texto23[2].setText("*");}
                    else{texto23[2].setText(q[3].valueOf(q[3].charAt(2)));}


                    if (q[3].charAt(3) == '#'){texto23[3].setText("+");}
                    else{texto23[3].setText(q[3].valueOf(q[3].charAt(3)));}


                    if (q[3].charAt(4) == '$'){texto23[4].setText("*");}
                    else{texto23[4].setText(q[3].valueOf(q[3].charAt(4)));}


                    if (q[3].charAt(5) == '?'){texto23[5].setText("+");}
                    else{texto23[5].setText(q[3].valueOf(q[3].charAt(5)));}


                    for(int i=0;i<6;i++)
                    {
                          texto16[i].setFont(new  java.awt.Font("Lucida  Sans",
1, 18));
                          texto17[i].setFont(new  java.awt.Font("Lucida  Sans",
1, 18));
                          texto18[i].setFont(new  java.awt.Font("Lucida  Sans",
1, 18));
                          texto19[i].setFont(new  java.awt.Font("Lucida  Sans",
1, 18));
                          texto20[i].setFont(new  java.awt.Font("Lucida  Sans",
1, 18));
                          texto21[i].setFont(new  java.awt.Font("Lucida  Sans",
1, 18));
                          texto22[i].setFont(new  java.awt.Font("Lucida  Sans",
1, 18));
                          texto23[i].setFont(new  java.awt.Font("Lucida  Sans",
1, 18));
                    }


                          diseñarLetra(fichas, texto16, 25, 30, 1, 0, 1);

                          diseñarLetra(fichas, texto17, 25, 60, 4, 6, 2);
                          diseñarLetra(fichas, texto18, 25, 90, 7, 12, 1);
                          diseñarLetra(fichas, texto19, 25, 120, 10, 18, 2);
                          diseñarLetra(fichas, texto20, 600, 30, 8, 24, 1);
                          diseñarLetra(fichas, texto21, 600, 60, 11, 30, 2);
                          diseñarLetra(fichas, texto22, 600, 90, 14, 36, 1);
                          diseñarLetra(fichas, texto23, 600, 120, 17, 42, 2);

                          if     (índice     ==     2){cambiarColor(fichas[41],
texto22[5]);}
                          if     (índice     ==     5){cambiarColor(fichas[47],
texto23[5]);}
                          if(índice       ==       73){cambiarColor(fichas[0],
texto16[0]);}
                          if(índice       ==       76){cambiarColor(fichas[6],
texto17[0]);}




                    if (puntero == 0)
```

```
                        {
                        texto[8].setText("Trama      (Fuente):      "      +
pinicial[0].charAt(puntero) +
                        pinicial[1].charAt(puntero)                         +
pinicial[2].charAt(puntero) + pinicial[3].charAt(puntero));
                        }

                        else if (puntero<=5 && trama1 == true)
                        {
                        texto[8].setText("Trama      (Fuente):      "      +
pinicial[0].charAt(1) +
                        pinicial[1].charAt(1)   +   pinicial[2].charAt(1)   +
pinicial[3].charAt(1));
                        }

                        else if (puntero<=5 && trama2 == true)
                        {
                        texto[8].setText("Trama      (Fuente):      "      +
pinicial[0].charAt(2) +
                        pinicial[1].charAt(2)   +   pinicial[2].charAt(2)   +
pinicial[3].charAt(2));
                        }

                        else if (puntero<=5 && trama3 == true)
                        {
                        texto[8].setText("Trama      (Fuente):      "      +
pinicial[0].charAt(3) +
                        pinicial[1].charAt(3)   +   pinicial[2].charAt(3)   +
pinicial[3].charAt(3));
                        }

                        else if (puntero<=5 && trama4 == true)
                        {
                        texto[8].setText("Trama      (Fuente):      "      +
pinicial[0].charAt(4) +
                        pinicial[1].charAt(4)   +   pinicial[2].charAt(4)   +
pinicial[3].charAt(4));
                        }

                        else if (puntero<=5 && trama5 == true)
                        {
                              texto[8].setText("Trama    (Fuente):    "    +
pinicial[0].charAt(5) +
                        pinicial[1].charAt(5)   +   pinicial[2].charAt(5)   +
pinicial[3].charAt(5));
                        }

                        else if (puntero>5 && tramaFin == false)
                        {
                              texto[8].setText("Trama    (Fuente):    "    +
pinicial[0].charAt(5) +
                        pinicial[1].charAt(5)   +   pinicial[2].charAt(5)   +
pinicial[3].charAt(5));
                        }


                        else if (puntero>5 && tramaFin == true)
                        {
                              texto[8].setText("Trama (Fuente): ****");

                        }

                        //Para el caso en que cambia puntero a 1 y hay que
seguir
                        //mostrando la trama para puntero=0:

                        else
                        {
```

```
                              texto[8].setText("Trama    (Fuente):    "    +
pinicial[0].charAt(0) +
                        pinicial[1].charAt(0)   +   pinicial[2].charAt(0)   +
pinicial[3].charAt(0));
                        }
                        texto[8].reshape(500, 320, 250, 25);
                        texto[8].setFont(new    java.awt.Font("Dialog",    1,
14));
                        getContentPane().add(texto[8]);

                  texto[9].setText("Slots =>");
                  texto[9].reshape(500, 345, 200, 25);
                        texto[9].setFont(new    java.awt.Font("Dialog",    1,
14));
                  getContentPane().add(texto[9]);

                  texto[10].setText("Fuente :     " + datos[0]);
                        texto[10].setFont(new   java.awt.Font("Dialog",   1,
14));
                  texto[10].reshape(550, 370, 200, 25);
                  getContentPane().add(texto[10]);

                  texto[11].setText("Enlace :      " + datos[1]);
                        texto[11].setFont(new   java.awt.Font("Dialog",   1,
14));
                  texto[11].reshape(550, 395, 200, 25);
                  getContentPane().add(texto[11]);

                  texto[12].setText("Receptor :  " + datos[2]);
                        texto[12].setFont(new   java.awt.Font("Dialog",   1,
14));
                  texto[12].reshape(550, 420, 200, 25);
                  getContentPane().add(texto[12]);

                  texto[13].setText("Ciclo:  " + ciclo);
                        texto[13].setFont(new   java.awt.Font("Dialog",   1,
14));
                  texto[13].reshape(500, 445, 200, 25);
                  getContentPane().add(texto[13]);

                        índiceTexto.setText(" " + índice);
                  índiceTexto.reshape(20, 500, 200, 25);
                  getContentPane().add(índiceTexto);

                        medio1.setText(linea[0]);
                  medio1.setFont(new java.awt.Font("Lucida Sans", 1, 18));
                        if                (ciclo.equals("0                -
Salida")){cambiarColor(fichas[48], medio1);}
                        else{medio1.setForeground(new  java.awt.Color(0,  64,
128));
                        fichas[48].setBackground(new java.awt.Color(63, 200,
224));}
                        fichas[48].add(medio1);
                        fichas[48].setBounds(180, 75, 20, 30);

                        medio2.setText(linea[1]);
                  medio2.setFont(new java.awt.Font("Lucida Sans", 1, 18));
                  medio2.setForeground(new java.awt.Color(0, 64, 128));
                        fichas[49].setBackground(new     java.awt.Color(192,
192, 192));
                        fichas[49].add(medio2);
                        fichas[49].setBounds(200, 75, 20, 30);

                        medio3.setText(linea[1]);
                  medio3.setFont(new java.awt.Font("Lucida Sans", 1, 18));
                  medio3.setForeground(new java.awt.Color(0, 64, 128));
                        fichas[50].setBackground(new     java.awt.Color(192,
192, 192));
```

```
                        fichas[50].add(medio3);
                        fichas[50].setBounds(370, 75, 20, 30);

                        medio4.setText(linea[1]);
                  medio4.setFont(new java.awt.Font("Lucida Sans", 1, 18));
                  medio4.setForeground(new java.awt.Color(0, 64, 128));
                        fichas[51].setBackground(new     java.awt.Color(192,
192, 192));
                        fichas[51].add(medio4);
                        fichas[51].setBounds(525, 75, 20, 30);

                        medio5.setText(linea[2]);
                  medio5.setFont(new java.awt.Font("Lucida Sans", 1, 18));
                        if                (ciclo.equals("1                -
Entrada")){cambiarColor(fichas[52], medio5);}
                        else{medio5.setForeground(new  java.awt.Color(0,  64,
128));
                        fichas[52].setBackground(new java.awt.Color(63, 200,
224));}
                        fichas[52].add(medio5);
                        fichas[52].setBounds(545, 75, 20, 30);


                  for(int i=0;i<12;i++)
                  {
                        button[i].setEnabled(true);
                  }



                  if (índice == 0)
                  {
                        button[6].setEnabled(false);
                        button[7].setEnabled(false);
                        button[8].setEnabled(false);
                        button[4].setEnabled(false);
                  }
                  if (índice == 78)
                  {
                        button[9].setEnabled(false);
                        button[10].setEnabled(false);
                        button[11].setEnabled(false);
                        button[3].setEnabled(false);
                  }

            }//llave de pintarLabels()



            public void diseñarLetra(JPanel p[], JLabel l[], int x, int y,
int n, int f, int c)
            {
                  for(int i=0;i<6;i++)
                  {
                        p[i+f].add(l[i]);
                        p[i+f].setBounds(x, y, 20, 30);
                        if (índice == n){cambiarColor(p[i+f], l[i]);}
                        else
                        {
                              l[i].setForeground(new  java.awt.Color(0,  64,
128));

                              if (c == 1)
                              {
                                    if (x == 25||x == 65||x == 105||x ==
600||x == 640||x == 680)
                                    {
```

```
                                        p[i+f].setBackground(new
java.awt.Color(63, 200, 224));
                                }
                                else if (x == 45||x == 85||x == 125||x
== 620||x == 660||x == 700)
                                {

      p[i+f].setBackground(Color.white);
                                }
                          }
                          else if (c == 2)
                          {
                                if (x == 25||x == 65||x == 105||x ==
600||x == 640||x == 680)
                                {

      p[i+f].setBackground(Color.white);
                                }
                                else if (x == 45||x == 85||x == 125||x
== 620||x == 660||x == 700)
                                {
                                        p[i+f].setBackground(new
java.awt.Color(63, 200, 224));
                                }
                          }
                    }
                    x = x + 20;
                    n = n + 12;
              }
        }



        private void cicloHandler(java.awt.event.ActionEvent evt)
        {
              if (asistente == true)
              {
                    ayudado.setVisible(false);
                    asistente = false;
                    setCursor(new
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
                    JLabel frase[] = new JLabel[9];
                    for(int i=0;i<9;i++)
                    {
                          frase[i] = new JLabel();
                          frase[i].setFont(new   java.awt.Font("Dialog",
1, 12));
                    }
                    JDialog dialogo = new JDialog();
                    dialogo.getContentPane().setBackground(Color.white);
                    dialogo.setTitle("Ayuda Botón Ciclo");
                    dialogo.setSize(450, 250);
                    dialogo.setResizable(false);
                    dialogo.getContentPane().setLayout(null);
                    frase[0].setText("Mediante su ejecución paso a paso,
vemos los 3 ciclos en los que se");
                    frase[1].setText("divide  el  desplazamiento  de  1
slot, el primer ciclo corresponde a la");
                    frase[2].setText("entrada de la letra seleccionada
en el buffer de salida, con el");
                    frase[3].setText("consiguiente desplazamiento de la
línea, el segundo ciclo corresponde");
                    frase[4].setText("a la entrada en el receptor de la
letra que había quedado en el buffer");
                    frase[5].setText("de entrada tras el primer ciclo, y
el tercero es un ciclo de espera");
```

```
                        frase[6].setText("necesario   para   la   correcta
simulación dado que en el canal existe un");
                        frase[7].setText("retardo  temporal  que  es  el  que
simula este tercer ciclo (Tiempo de");
                        frase[8].setText("transmisión).");
                        frase[0].reshape(20, 20, 400, 20);
                        frase[1].reshape(20, 40, 400, 20);
                        frase[2].reshape(20, 60, 400, 20);
                        frase[3].reshape(20, 80, 400, 20);
                        frase[4].reshape(20, 100, 400, 20);
                        frase[5].reshape(20, 120, 400, 20);
                        frase[6].reshape(20, 140, 400, 20);
                        frase[7].reshape(20, 160, 400, 20);
                        frase[8].reshape(20, 180, 400, 20);
                        dialogo.setModal(true); //así hasta que no se cierre
                                                                  //la  ventana
de ayuda, no se podrá seguir
                        for(int i=0;i<9;i++)
                        {
                               dialogo.getContentPane().add(frase[i]);
                        }

                        dialogo.show();
                  }
                  else
                  {
                        if (índice == 78)   //si  ha  llegado  al  final,  lo
inhabilitamos
                        {
                               return;
                        }
                        else if (habilitado == true)
                        {
                               indicarCaminosCiclos();
                               viajeCiclo(p[tirada], q[tirada]);
                               pintarLabels();
                        }
                  }
            }

            private void timeslotHandler(java.awt.event.ActionEvent evt)
            {
                  if (asistente == true)
                  {
                        ayudado.setVisible(false);
                        asistente = false;
                        setCursor(new
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
                        JLabel frase[] = new JLabel[5];
                        for(int i=0;i<5;i++)
                        {
                               frase[i] = new JLabel();
                               frase[i].setFont(new   java.awt.Font("Dialog",
1, 12));
                        }
                        JDialog dialogo = new JDialog();
                        dialogo.getContentPane().setBackground(Color.white);
                        dialogo.setTitle("Ayuda Botón Slot");
                        dialogo.setSize(450, 180);
                        dialogo.setResizable(false);
                        dialogo.getContentPane().setLayout(null);
                        frase[0].setText("Simula  el  desplazamiento  de  las
letras en un slot de tiempo, donde");
                        frase[1].setText("se sitúa la letra a transmitir de
la fuente en el buffer de salida,");
                        frase[2].setText("las   letras   de   la   línea   se
desplazan, y la que llega al buffer de");
```

```
                        frase[3].setText("entrada    llega    a    su    lugar
correspondiente en el receptor. En definitiva,");
                        frase[4].setText("se    ejecutan    los    3    ciclos
simultáneamente.");

                        frase[0].reshape(20, 20, 400, 20);
                        frase[1].reshape(20, 40, 400, 20);
                        frase[2].reshape(20, 60, 400, 20);
                        frase[3].reshape(20, 80, 400, 20);
                        frase[4].reshape(20, 100, 400, 20);
                        dialogo.setModal(true); //así hasta que no se cierre
                                                                  //la  ventana
de ayuda, no se podrá seguir
                        for(int i=0;i<5;i++)
                        {
                               dialogo.getContentPane().add(frase[i]);
                        }

                        dialogo.show();
                  }
                  else
                  {
                  if (índice == 78)
                  {
                        return;
                  }

                  if (habilitado == true)
                  {

                        if  (iteración  ==  0||iteración  ==  3||iteración  ==
6||iteración == 9)
                        {
                               indicarCaminosCiclos();
                               viajeCiclo(p[tirada], q[tirada]);
                               indicarCaminosCiclos();
                               viajeCiclo(p[tirada], q[tirada]);
                               indicarCaminosCiclos();
                               viajeCiclo(p[tirada], q[tirada]);
                        }
                        else  if  (iteración  ==  1||iteración  ==  4||iteración
== 7||iteración == 10)
                        {
                               indicarCaminosCiclos();
                               viajeCiclo(p[tirada], q[tirada]);
                               indicarCaminosCiclos();
                               viajeCiclo(p[tirada], q[tirada]);
                        }
                        else  if  (iteración  ==  2||iteración  ==  5||iteración
== 8||iteración == 11)
                        {
                               indicarCaminosCiclos();
                               viajeCiclo(p[tirada], q[tirada]);
                        }

                        pintarLabels();
                  }
                  }
           }

              private void tramaHandler(java.awt.event.ActionEvent evt)
              {
                     if (asistente == true)
                     {
                        ayudado.setVisible(false);
                        asistente = false;
                        setCursor(new
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
```

```
                        JLabel frase[] = new JLabel[2];
                        for(int i=0;i<2;i++)
                        {
                              frase[i] = new JLabel();
                              frase[i].setFont(new   java.awt.Font("Dialog",
1, 12));
                        }
                        JDialog dialogo = new JDialog();
                        dialogo.getContentPane().setBackground(Color.white);
                        dialogo.setTitle("Ayuda Botón Trama");
                        dialogo.setSize(400, 120);
                        dialogo.setResizable(false);
                        dialogo.getContentPane().setLayout(null);
                        frase[0].setText("Se transmite una trama entera, en
este caso unas 4 letras,");
                        frase[1].setText("equivale a pulsar 4 veces el botón
'Slot'.");

                        frase[0].reshape(20, 20, 400, 20);
                        frase[1].reshape(20, 40, 400, 20);

                        dialogo.setModal(true); //así hasta que no se cierre
                                                                  //la  ventana
de ayuda, no se podrá seguir
                        for(int i=0;i<2;i++)
                        {
                              dialogo.getContentPane().add(frase[i]);
                        }

                        dialogo.show();
                  }
                  else
                  {
                  if (índice == 78){return;}
                  else if (habilitado == true)
                  {
                        if (índice >= 72)
                        {
                              viajeTotal();
                        }
                        else
                        {
                              if (índice < 12)
                              {
                                    for(int i=índice;i<12;i++)
                                    {
                                          indicarCaminosCiclos();
                                          viajeCiclo(p[tirada],
q[tirada]);
                                    }
                              }
                              else if (índice >= 12 && índice <24)
                              {
                                    for(int i=índice;i<24;i++)
                                    {
                                          indicarCaminosCiclos();
                                          viajeCiclo(p[tirada],
q[tirada]);
                                    }
                              }
                              else if (índice >= 24 && índice <36)
                              {
                                    for(int i=índice;i<36;i++)
                                    {
                                          indicarCaminosCiclos();
                                          viajeCiclo(p[tirada],
q[tirada]);
                                    }
```

```
                                }
                                else if (índice >= 36 && índice <48)
                                {
                                       for(int i=índice;i<48;i++)
                                       {
                                              indicarCaminosCiclos();
                                              viajeCiclo(p[tirada],
q[tirada]);
                                       }
                                }
                                else if (índice >= 36 && índice <48)
                                {
                                       for(int i=índice;i<48;i++)
                                       {
                                              indicarCaminosCiclos();
                                              viajeCiclo(p[tirada],
q[tirada]);
                                       }
                                }
                                else if (índice >= 48 && índice <60)
                                {
                                       for(int i=índice;i<60;i++)
                                       {
                                              indicarCaminosCiclos();
                                              viajeCiclo(p[tirada],
q[tirada]);
                                       }
                                }
                                else if (índice >= 60 && índice <72)
                                {
                                       for(int i=índice;i<72;i++)
                                       {
                                              indicarCaminosCiclos();
                                              viajeCiclo(p[tirada],
q[tirada]);
                                       }
                                }
                         }
                         pintarLabels();
                  }
                  }
           }

              private void finHandler(java.awt.event.ActionEvent evt)
              {
                     if (asistente == true)
                     {
                            ayudado.setVisible(false);
                            asistente = false;
                            setCursor(new
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
                            JLabel frase[] = new JLabel[3];
                            for(int i=0;i<3;i++)
                            {
                                   frase[i] = new JLabel();
                                   frase[i].setFont(new   java.awt.Font("Dialog",
1, 12));
                            }
                            JDialog dialogo = new JDialog();
                            dialogo.getContentPane().setBackground(Color.white);
                            dialogo.setTitle("Ayuda Botón Fin de Simulación");
                            dialogo.setSize(400, 140);
                            dialogo.setResizable(false);
                            dialogo.getContentPane().setLayout(null);
                            frase[0].setText("Se transmiten todas las letras, el
receptor tiene ya todos");
                            frase[1].setText("los  canales  ocupados  y  en  el
buffer de entrada queda la");
```

```
                        frase[2].setText("última   letra   situada   en   el
receptor.");
                        frase[0].reshape(20, 20, 400, 20);
                        frase[1].reshape(20, 40, 400, 20);
                        frase[2].reshape(20, 60, 400, 20);

                        dialogo.setModal(true); //así hasta que no se cierre
                                                                  //la  ventana
de ayuda, no se podrá seguir
                        for(int i=0;i<3;i++)
                        {
                               dialogo.getContentPane().add(frase[i]);
                        }

                        dialogo.show();
                  }
                  else
                  {
                  if (índice>=78)
                  {
                        return;
                  }
                  else if (habilitado == true)
                  {
                        indicarCaminosCiclos();
                        viajeTotal();
                        pintarLabels();
                  }
                  }
            }

            private void reiniciarHandler(java.awt.event.ActionEvent evt)
            {
                  if (asistente == true)
                  {
                        ayudado.setVisible(false);
                        asistente = false;
                        setCursor(new
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
                        JLabel frase = new JLabel();
                        JDialog dialogo = new JDialog();
                        dialogo.getContentPane().setBackground(Color.white);
                        dialogo.setTitle("Ayuda Botón Reiniciar");
                        dialogo.setSize(230, 100);
                        dialogo.setResizable(false);
                        dialogo.getContentPane().setLayout(null);
                        frase.setText("Vuelve todo a su estado inicial");
                        frase.setFont(new java.awt.Font("Dialog", 1, 12));
                        frase.reshape(20, 20, 400, 20);

                        dialogo.setModal(true); //así hasta que no se cierre
                                                                  //la  ventana
de ayuda, no se podrá seguir
                        dialogo.getContentPane().add(frase);

                        dialogo.show();
                  }
                  else
                  {
                        reiniciar();
                        pintarLabels();
                  }
            }
            private void menuHandler(java.awt.event.ActionEvent evt) {
              if (asistente == true)
                  {
                        ayudado.setVisible(false);
                        asistente = false;
```

```
                        setCursor(new
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
                        JLabel frase[] = new JLabel[2];
                        for(int i=0;i<2;i++)
                        {
                              frase[i] = new JLabel();
                              frase[i].setFont(new   java.awt.Font("Dialog",
1, 12));
                        }
                        JDialog dialogo = new JDialog();
                        dialogo.getContentPane().setBackground(Color.white);
                        dialogo.setTitle("Ayuda Botón Menú Principal");
                        dialogo.setSize(340, 120);
                        dialogo.setResizable(false);
                        dialogo.getContentPane().setLayout(null);
                        frase[0].setText("Cierra  esta  ventana  y  vuelve  al
menú principal del");
                        frase[1].setText("simulador.");

                        frase[0].reshape(20, 20, 400, 20);
                        frase[1].reshape(20, 40, 400, 20);

                        dialogo.setModal(true); //así hasta que no se cierre
                                                                  //la  ventana
de ayuda, no se podrá seguir
                        for(int i=0;i<2;i++)
                        {
                              dialogo.getContentPane().add(frase[i]);
                        }

                        dialogo.show();
                  }
                  else
                  {
                        General g1 = new General();
                        g1.show();
                        (this).hide();
                  }
            }

               private void atrásCicloHandler(java.awt.event.ActionEvent evt)
               {
                     if (asistente == true)
                     {
                        asistente = false;
                        setCursor(new
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
                        ayudado.setVisible(false);

                        JLabel frase = new JLabel();
                        frase.setFont(new java.awt.Font("Dialog", 1, 12));

                        JDialog dialogo = new JDialog();
                        dialogo.getContentPane().setBackground(Color.white);
                        dialogo.setTitle("Ayuda Marcha Atrás Ciclo");
                        dialogo.setSize(220, 100);
                        dialogo.setResizable(false);
                        dialogo.getContentPane().setLayout(null);
                        frase.setText("Retrocede un ciclo.");
                        frase.reshape(20, 20, 400, 20);

                        dialogo.setModal(true); //así hasta que no se cierre
                                                                  //la  ventana
de ayuda, no se podrá seguir

                        dialogo.getContentPane().add(frase);

                        dialogo.show();
```

```
                }
                else
                {
                        if(índice == 0){return;}
                        else
                        {
                                marchaAtrásCiclo();
                                pintarLabels();
                        }
                }

        }

           private void atrásSlotHandler(java.awt.event.ActionEvent evt)
           {

                if (asistente == true)
                {
                        asistente = false;
                        setCursor(new
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
                        ayudado.setVisible(false);

                        JLabel frase = new JLabel();
                        frase.setFont(new java.awt.Font("Dialog", 1, 12));

                        JDialog dialogo = new JDialog();
                        dialogo.getContentPane().setBackground(Color.white);
                        dialogo.setTitle("Ayuda Marcha Atrás Slot");
                        dialogo.setSize(220, 100);
                        dialogo.setResizable(false);
                        dialogo.getContentPane().setLayout(null);
                        frase.setText("Retrocede un slot.");
                        frase.reshape(20, 20, 400, 20);

                        dialogo.setModal(true); //así hasta que no se cierre
                                                                //la  ventana
de ayuda, no se podrá seguir

                        dialogo.getContentPane().add(frase);

                        dialogo.show();
                }

                else
                {
                if(índice == 0){return;}
                else if (índice<3)
                {
                        reiniciar();
                        pintarLabels();
                }
                else
                {
                        marchaAtrásCiclo();
                        marchaAtrásCiclo();
                        marchaAtrásCiclo();
                        pintarLabels();
           }
                }
           }

           private void atrásTramaHandler(java.awt.event.ActionEvent evt)
           {
                if (asistente == true)
                {
                        asistente = false;
```

```
                        setCursor(new
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
                        ayudado.setVisible(false);

                        JLabel frase = new JLabel();
                        frase.setFont(new java.awt.Font("Dialog", 1, 12));

                        JDialog dialogo = new JDialog();
                        dialogo.getContentPane().setBackground(Color.white);
                        dialogo.setTitle("Ayuda Marcha Atrás Trama");
                        dialogo.setSize(220, 100);
                        dialogo.setResizable(false);
                        dialogo.getContentPane().setLayout(null);
                        frase.setText("Retrocede una trama.");
                        frase.reshape(20, 20, 400, 20);

                        dialogo.setModal(true); //así hasta que no se cierre
                                                                    //la  ventana
de ayuda, no se podrá seguir

                        dialogo.getContentPane().add(frase);

                        dialogo.show();
                  }
                  else
                  {
                  if (índice == 0)
                  {
                        return;
                  }
                  else if (índice<12)
                  {
                        reiniciar();
                        pintarLabels();
                  }
                  else
                  {
                        for(int i=0;i<12;i++)
                        {
                              marchaAtrásCiclo();
                        }
                        pintarLabels();
                  }
                  }
           }

              private void ayudaHandler(java.awt.event.ActionEvent evt)
              {
                    if (asistente == true)
                    {
                        asistente = false;
                        setCursor(new
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
                        ayudado.setVisible(false);
                    }
                    else
                    {
                        asistente = true;
                        setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
                        ayudado.setVisible(true);

                    }
              }


              private void ayudaFuente(java.awt.event.MouseEvent evt)
              {
```

```
                    if (asistente == true)
                    {
                           asistente = false;
                           setCursor(new
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
                           ayudado.setVisible(false);
                           JLabel frase[] = new JLabel[3];
                           for(int i=0;i<3;i++)
                           {
                                  frase[i] = new JLabel();
                                  frase[i].setFont(new   java.awt.Font("Dialog",
1, 12));
                           }
                           JDialog dialogo = new JDialog();
                           dialogo.getContentPane().setBackground(Color.white);
                           dialogo.setTitle("Ayuda Módulo Emisor");
                           dialogo.setSize(380, 140);
                           dialogo.setResizable(false);
                           dialogo.getContentPane().setLayout(null);
                           frase[0].setText("Éste  es  el  módulo  fuente,  donde
hay 4 canales con las");
                           frase[1].setText("palabras a transmitir,  las cuales
van identificadas");
                           frase[2].setText("por  las  direcciones  especificadas
a la izquierda.");

                           frase[0].reshape(20, 20, 400, 20);
                           frase[1].reshape(20, 40, 400, 20);
                           frase[2].reshape(20, 60, 400, 20);

                           dialogo.setModal(true); //así hasta que no se cierre
                                                                  //la  ventana
de ayuda, no se podrá seguir
                           for(int i=0;i<3;i++)
                           {
                                  dialogo.getContentPane().add(frase[i]);
                           }

                           dialogo.show();

                    }
             }

             private void ayudaReceptor(java.awt.event.MouseEvent evt)
             {
                    if (asistente == true)
                    {
                           asistente = false;
                           setCursor(new
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
                           ayudado.setVisible(false);
                           JLabel frase[] = new JLabel[3];
                           for(int i=0;i<3;i++)
                           {
                                  frase[i] = new JLabel();
                                  frase[i].setFont(new   java.awt.Font("Dialog",
1, 12));
                           }
                           JDialog dialogo = new JDialog();
                           dialogo.getContentPane().setBackground(Color.white);
                           dialogo.setTitle("Ayuda Módulo Receptor");
                           dialogo.setSize(380, 140);
                           dialogo.setResizable(false);
                           dialogo.getContentPane().setLayout(null);
                           frase[0].setText("Éste es el módulo receptor, donde
hay 4 canales esperando");
                           frase[1].setText("las  letras  que  vayan  entrando  en
el buffer, cada canal");
```

```
                        frase[2].setText("se identifica con las direcciones
mostradas a la derecha.");

                        frase[0].reshape(20, 20, 400, 20);
                        frase[1].reshape(20, 40, 400, 20);
                        frase[2].reshape(20, 60, 400, 20);

                        dialogo.setModal(true); //así hasta que no se cierre
                                                                //la  ventana
de ayuda, no se podrá seguir
                        for(int i=0;i<3;i++)
                        {
                               dialogo.getContentPane().add(frase[i]);
                        }

                        dialogo.show();

                }
        }


        private void canalVoz(java.awt.event.MouseEvent evt)
        {
               if (asistente == true)
               {
                        asistente = false;
                        setCursor(new
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
                        ayudado.setVisible(false);
                        JLabel frase[] = new JLabel[4];
                        for(int i=0;i<4;i++)
                        {
                               frase[i] = new JLabel();
                               frase[i].setFont(new   java.awt.Font("Dialog",
1, 12));
                        }
                        JDialog dialogo = new JDialog();
                        dialogo.getContentPane().setBackground(Color.white);
                        dialogo.setTitle("Ayuda Canal Vocal");
                        dialogo.setSize(380, 160);
                        dialogo.setResizable(false);
                        dialogo.getContentPane().setLayout(null);
                        frase[0].setText("Éste es uno de los 4 canales de
voz que se transmitirán");
                        frase[1].setText("al receptor. Cada canal tiene 6
letras a transmitir. En");
                        frase[2].setText("el momento en que una letra esté
pasando el buffer de ");
                        frase[3].setText("salida, ésta se iluminará.");

                        frase[0].reshape(20, 20, 400, 20);
                        frase[1].reshape(20, 40, 400, 20);
                        frase[2].reshape(20, 60, 400, 20);
                        frase[3].reshape(20, 80, 400, 20);

                        dialogo.setModal(true); //así hasta que no se cierre
                                                                //la  ventana
de ayuda, no se podrá seguir
                        for(int i=0;i<4;i++)
                        {
                               dialogo.getContentPane().add(frase[i]);
                        }

                        dialogo.show();

                }
        }
```

```
            private void canalVozReceptor(java.awt.event.MouseEvent evt)
            {
                  if (asistente == true)
                  {
                        asistente = false;
                        setCursor(new
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
                        ayudado.setVisible(false);
                        JLabel frase[] = new JLabel[4];
                        for(int i=0;i<4;i++)
                        {
                              frase[i] = new JLabel();
                              frase[i].setFont(new   java.awt.Font("Dialog",
1, 12));
                        }
                        JDialog dialogo = new JDialog();
                        dialogo.getContentPane().setBackground(Color.white);
                        dialogo.setTitle("Ayuda Canal Vocal");
                        dialogo.setSize(380, 160);
                        dialogo.setResizable(false);
                        dialogo.getContentPane().setLayout(null);
                        frase[0].setText("Éste es uno de los 4 canales de
voz que se encuentra en");
                        frase[1].setText("el receptor. Cada canal tiene 6
letras a recibir. En el");
                        frase[2].setText("momento en que una letra esté
pasando del buffer de ");
                        frase[3].setText("entrada al receptor, ésta se
iluminará.");

                        frase[0].reshape(20, 20, 400, 20);
                        frase[1].reshape(20, 40, 400, 20);
                        frase[2].reshape(20, 60, 400, 20);
                        frase[3].reshape(20, 80, 400, 20);

                        dialogo.setModal(true); //así hasta que no se cierre
                                                                  //la  ventana
de ayuda, no se podrá seguir
                        for(int i=0;i<4;i++)
                        {
                              dialogo.getContentPane().add(frase[i]);
                        }

                        dialogo.show();

                  }
            }

            private void ayudaEstado(java.awt.event.MouseEvent evt)
            {
                  if (asistente == true)
                  {
                        asistente = false;
                        setCursor(new
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
                        ayudado.setVisible(false);
                        JLabel frase[] = new JLabel[10];
                        for(int i=0;i<10;i++)
                        {
                              frase[i] = new JLabel();
                              frase[i].setFont(new   java.awt.Font("Dialog",
1, 12));
                        }
                        JDialog dialogo = new JDialog();
                        dialogo.getContentPane().setBackground(Color.white);
                        dialogo.setTitle("Ayuda Indicadores de Estado");
                        dialogo.setSize(380, 280);
```

```
                        dialogo.setResizable(false);
                        dialogo.getContentPane().setLayout(null);
                        frase[0].setText("Son indicadores del estado de la
simulación, donde se");
                        frase[1].setText("puede observar en este orden: la
trama que en ese instante");
                        frase[2].setText("se está transmitiendo, los canales
a los que pertenecen");
                        frase[3].setText("las letras que están en la fuente,
en el enlace y en el");
                        frase[4].setText("receptor, y finalmente, el ciclo
donde estamos, donde");
                        frase[5].setText("0 - Salida es el ciclo donde
información de voz de la ");
                        frase[6].setText("fuente llega al buffer de salida,
1 - Entrada es el ciclo");
                        frase[7].setText("donde la letra del buffer de
entrada pasa a ocupar su sitio");
                        frase[8].setText("en el receptor, y 2 - Espera es el
ciclo de espera necesario");
                        frase[9].setText("para la correcta simulación.");

                        frase[0].reshape(20, 20, 400, 20);
                        frase[1].reshape(20, 40, 400, 20);
                        frase[2].reshape(20, 60, 400, 20);
                        frase[3].reshape(20, 80, 400, 20);
                        frase[4].reshape(20, 100, 400, 20);
                        frase[5].reshape(20, 120, 400, 20);
                        frase[6].reshape(20, 140, 400, 20);
                        frase[7].reshape(20, 160, 400, 20);
                        frase[8].reshape(20, 180, 400, 20);
                        frase[9].reshape(20, 200, 400, 20);



                        dialogo.setModal(true); //así hasta que no se cierre
                                                                //la  ventana
de ayuda, no se podrá seguir
                        for(int i=0;i<10;i++)
                        {
                               dialogo.getContentPane().add(frase[i]);
                        }

                        dialogo.show();

                  }
            }

         // Para salir de la aplicación:

         private void exitForm(java.awt.event.WindowEvent evt)
            {
              System.exit(0);
         }


            //Con viajeCiclo() se hará el mínimo desplazamiento posible,
            //si queremos transmitir de golpe una trama, pues llamaremos
            //a este método tantas veces como canales tengamos, es el
            //método clave del programa:

            public void viajeCiclo(String pe, String qe)
            {
                  String pe1, pe2, pe3, qe1, qe2, qe3;
                  pe = p[tirada];
                  qe = q[tirada];
                  pe1 = p[tirada + 1];
                  qe1 = q[tirada + 1];
```

```
                    pe2 = p[tirada + 2];
                    qe2 = q[tirada + 2];
                    pe3 = p[tirada + 3];
                    qe3 = q[tirada + 3];

                    if (puntero > 5)
                    {
                          fin = true;
                    }

                    if(iteración == 0)//1ª pulsación
                    {
                          if (residuo == true)
                          {
                                linea[2]                                        =
linea[1].valueOf(linea[1].charAt(0));
                                linea[1]                                        =
linea[0].valueOf(linea[0].charAt(0));
                          }

                          if (primera_vez == true)
                          {
                                linea[2]                                        =
linea[1].valueOf(linea[1].charAt(0));
                                linea[1]                                        =
linea[0].valueOf(linea[0].charAt(0));
                          }
                          if (fin == false)
                          {
                                linea[0] = pe.valueOf(pe.charAt(puntero));

                          }

                          if (fin == true)
                          {
                                linea[0]                                        =
linea[0].replace(linea[0].charAt(0), '*');
                          }

                          ciclo = "0 - Salida";
                          if (puntero == 1){trama1 = true;}
                          else  if  (puntero  ==  2){trama1  =  false;trama2  =
true;}
                          else  if  (puntero  ==  3){trama2  =  false;trama3  =
true;}
                          else  if  (puntero  ==  4){trama3  =  false;trama4  =
true;}
                          else  if  (puntero  ==  5){trama4  =  false;trama5  =
true;}
                          else  if  (puntero  >  5){trama5  =  false;tramaFin  =
true;}
                          iteración++;
                          índice++;
                    }

                    else if(iteración == 1)//2ª pulsación
                    {

                          if (residuo == true)
                          {
                                qe2  =  qe2.replace(qe2.charAt(puntero  -  1),
linea[2].charAt(0));
                                q[tirada + 2] = qe2;
                          }
                          if (primera_vez == true)
                          {
                                qe2        =        qe2.replace(qe2.charAt(5),
linea[2].charAt(0));
```

```
                              q[tirada + 2] = qe2;
                        }

                        if (fin == false)
                        {
                              pe = pe.replace(pe.charAt(puntero), '*');
                              p[tirada] = pe;
                        }
                        ciclo = "1 - Entrada";
                        iteración++;
                        índice++;
                  }

                  else if (iteración == 2)//3ª pulsación
                  {
                        ciclo = "2 - Espera";
                        iteración++;
                        índice++;
                  }

                  else if (iteración == 3)//4ª pulsación
                  {
                        if (residuo == true)
                        {
                              linea[2]                                          =
linea[1].valueOf(linea[1].charAt(0));
                        }

                        if (primera_vez == true)
                        {
                              linea[2]                                          =
linea[1].valueOf(linea[1].charAt(0));
                        }
                        if (fin == false)
                        {
                              linea[1]                                          =
linea[0].valueOf(linea[0].charAt(0));
                              linea[0] = pe1.valueOf(pe1.charAt(puntero));
                        }
                        if (fin == true)
                        {
                              linea[1]                                          =
linea[1].replace(linea[1].charAt(0), '*');
                        }
                        ciclo = "0 - Salida";
                        iteración++;
                        índice++;
                  }

                  else if (iteración == 4)//5ª pulsación
                  {
                        if (residuo == true)
                        {
                              qe3  =  qe3.replace(qe3.charAt(puntero  -  1),
linea[2].charAt(0));
                              q[tirada + 3] = qe3;
                              residuo = false;
                        }
                        if (primera_vez == true)
                        {
                              qe3        =        qe3.replace(qe3.charAt(5),
linea[2].charAt(0));
                              q[tirada + 3] = qe3;
                              primera_vez = false;
                        }
                        if (fin == false)
                        {
                              pe1 = pe1.replace(pe1.charAt(puntero), '*');
```

```
                              p[tirada + 1] = pe1;
                        }
                        ciclo = "1 - Entrada";

                        iteración++;
                        índice++;
                  }

                  else if (iteración == 5)//6ª pulsación
                  {
                        ciclo = "2 - Espera";
                        iteración++;
                        índice++;
                  }

                  else if (iteración == 6)//7ª pulsación
                  {
                        if (fin == false)
                        {
                              linea[2]                                    =
linea[1].valueOf(linea[1].charAt(0));
                              linea[1]                                    =
linea[0].valueOf(linea[0].charAt(0));
                              linea[0] = pe2.valueOf(pe2.charAt(puntero));
                        }
                        ciclo = "0 - Salida";
                        iteración++;
                        índice++;

                        if (fin == true)
                        {
                              habilitado = false;
                        }
                  }


                  else if (iteración == 7)//8ª pulsación
                  {
                        if (fin == false)
                        {
                              qe                                          =
qe.replace(qe.charAt(puntero),linea[2].charAt(0));
                              q[tirada] = qe;
                              pe2 = pe2.replace(pe2.charAt(puntero), '*');
                              p[tirada + 2] = pe2;
                        }
                        ciclo = "1 - Entrada";
                        iteración++;
                        índice++;
                  }

                  else if (iteración == 8)//9ª
                  {
                        ciclo = "2 - Espera";
                        iteración++;
                        índice++;
                  }

                  else if (iteración == 9)  //10ª pulsación
                  {
                        if (fin == false)
                        {
                              linea[2]                                    =
linea[1].valueOf(linea[1].charAt(0));
                              linea[1]                                    =
linea[0].valueOf(linea[0].charAt(0));
                              linea[0] = pe3.valueOf(pe3.charAt(puntero));
                        }
```

```
                        ciclo = "0 - Salida";
                        iteración++;
                        índice++;
                }
                else if (iteración == 10) //11ª pulsación
                {
                        if (fin == false)
                        {
                                qe1     =     qe1.replace(qe1.charAt(puntero),
linea[2].charAt(0));
                                q[tirada + 1] = qe1;
                                pe3 = pe3.replace(pe3.charAt(puntero), '*');
                                p[tirada + 3] = pe3;
                        }
                        ciclo = "1 - Entrada";
                        iteración++;
                        índice++;
                }
                else if (iteración == 11) //12ª pulsación
                {
                        ciclo = "2 - Espera";
                        iteración = 0;
                        residuo = true;
                        puntero++;
                        índice++;
                }

        }//llave del método

        public void reiniciar()
        {
                for(int i=0;i<p.length;i++)
                {
                        p[i] = pinicial[i];
                        q[i] = qinicial[i];
                }
                linea[2] = "*";
                linea[1] = "+";
                linea[0] = "*";
                iteración = 0;
                puntero = 0;
                fin = false;
                residuo = false;
                ciclo = "2 - Espera";
                datos[0] = 3;
                datos[1] = 2;
                datos[2] = 1;
                primera_vez = true;
                habilitado = true;
                trama1 = false;
                trama2 = false;
                trama3 = false;
                trama4 = false;
                trama5 = false;
                tramaFin = false;
                índice = 0;
                asistente = false;
        }



        public void indicarCaminosCiclos()
        {
                if (iteración == 0)
                {
                        datos[0] = 0;
                        datos[1] = 3;
                        datos[2] = 2;
```

```
	}

	else if (iteración == 1){return;}
	else if (iteración == 2){return;}

	else if (iteración == 3)
	{
		datos[0] = 1;
		datos[1] = 0;
		datos[2] = 3;
	}

	else if (iteración == 4){return;}
	else if (iteración == 5){return;}

	else if (iteración == 6)
	{
		datos[0] = 2;
		datos[1] = 1;
		datos[2] = 0;
	}

	else if (iteración == 7){return;}
	else if (iteración == 8){return;}

	else if (iteración == 9)
	{
		datos[0] = 3;
		datos[1] = 2;
		datos[2] = 1;
	}
	else if (iteración == 10){return;}
	else if (iteración == 11){return;}
}

public void indicarCaminosCiclosAtrás()
{
	if (iteración == 0){return;}
	else if (iteración == 1)
	{
		datos[0] = 3;
		datos[1] = 2;
		datos[2] = 1;
	}
	else if (iteración == 2) {return;}
	else if (iteración == 3) {return;}
	else if (iteración == 4)
	{
		datos[0] = 0;
		datos[1] = 3;
		datos[2] = 2;
	}
	else if (iteración == 5) {return;}
	else if (iteración == 6) {return;}
	else if (iteración == 7)
	{
		datos[0] = 1;
		datos[1] = 0;
		datos[2] = 3;
	}
	else if (iteración == 8) {return;}
	else if (iteración == 9) {return;}
	else if (iteración == 10)
	{
		datos[0] = 2;
		datos[1] = 1;
		datos[2] = 0;
	}
```

```
        else if (iteración == 11) {return;}
}


public void guardarEstado()
{
        pestado0[índice] = p[0];
        pestado1[índice] = p[1];
        pestado2[índice] = p[2];
        pestado3[índice] = p[3];

        qestado0[índice] = q[0];
        qestado1[índice] = q[1];
        qestado2[índice] = q[2];
        qestado3[índice] = q[3];

        linea0Estado[índice] = linea[0];
        linea1Estado[índice] = linea[1];
        linea2Estado[índice] = linea[2];

        primera_vez_estado[índice] = primera_vez;
        residuoEstado[índice] = residuo;
        finEstado[índice] = fin;
        punteroEstado[índice] = puntero;
        cicloEstado[índice] = ciclo;

        trama1estado[índice] = trama1;
        trama2estado[índice] = trama2;
        trama3estado[índice] = trama3;
        trama4estado[índice] = trama4;
        trama5estado[índice] = trama5;
        tramaFinestado[índice] = tramaFin;
}

public void marchaAtrásCiclo()
{
        p[0] = pestado0[índice - 1];
        p[1] = pestado1[índice - 1];
        p[2] = pestado2[índice - 1];
        p[3] = pestado3[índice - 1];

        q[0] = qestado0[índice - 1];
        q[1] = qestado1[índice - 1];
        q[2] = qestado2[índice - 1];
        q[3] = qestado3[índice - 1];

        linea[0] = linea0Estado[índice - 1];
        linea[1] = linea1Estado[índice - 1];
        linea[2] = linea2Estado[índice - 1];

        primera_vez = primera_vez_estado[índice - 1];
        residuo = residuoEstado[índice - 1];
        fin = finEstado[índice - 1];
        puntero = punteroEstado[índice - 1];
        ciclo = cicloEstado[índice - 1];

        trama1 = trama1estado[índice - 1];
        trama2 = trama2estado[índice - 1];
        trama3 = trama3estado[índice - 1];
        trama4 = trama4estado[índice - 1];
        trama5 = trama5estado[índice - 1];
        tramaFin = tramaFinestado[índice - 1];

        indicarCaminosCiclosAtrás();

        if (iteración == 0){iteración = 11;}
        else{iteración--;}
        índice--;
```

```
                habilitado = true;
        }

        //Con grabarEstados() lo que hago es toda la conmutación
        //hasta el final, y en cada iteración, guardar el estado,
        //luego reinicio para que todo esté en su estado inicial
        //listo para su simulación paso a paso, lo único que no
        //se reinicia son los arrays donde se encuentran los estados
        //en cada iteración de todas las variables, así, sea cual sea
        //el estado en el que me encuentre, si doy marcha atrás, el
        //programa ya sabrá a qué estado acudir guiándose por el valor
        //de índice:

        public void grabarEstados()
        {
                for(int i=0;i<78;i++)
                {
                        guardarEstado();
                        viajeCiclo(p[tirada], q[tirada]);
                }
                reiniciar();
        }

        public void viajeTotal()
        {
                for(int i=índice;i<78;i++)
                {
                        indicarCaminosCiclos();
                        viajeCiclo(p[tirada], q[tirada]);
                }
        }


        public void cambiarColor(JPanel p, JLabel l)
        {
                p.setBackground(Color.red);
                l.setForeground(Color.yellow);
                p.setBorder(new
javax.swing.border.BevelBorder(javax.swing.border.BevelBorder.RAISED));
        }




        //el array datos[] servirá para indicar qué canales están
        //haciendo uso de la salida, el enlace y la entrada:

        private static int datos[] = new int[3];

        //Los pestado[] y qestado[] recogerán todos los valores
        //que presenten las variables p[] y q[] cuando usemos
        //grabarEstados(). A dichas variables acudiremos a la
        //hora de dar marcha atrás. Los arrays tienen tamaño de
        //79, que es el número máximo de posibles estados que
        //pueden presentarse en esta simulación para 4 canales,
        //estos tamaños lógicamente aumentarán conforme aumentemos
        //el número de canales en las otras opciones del simulador.
        //Lo dicho de pestado[] respecto a p[] es ampliable a todas
        //las variables que tienes su propio nombreseguido de "estado":



        private static String pestado0[] = new String[79];
        private static String pestado1[] = new String[79];
        private static String pestado2[] = new String[79];
        private static String pestado3[] = new String[79];
```

```
private  static String qestado0[] = new String[79];
   private      static String qestado1[] = new String[79];
   private      static String qestado2[] = new String[79];
   private      static String qestado3[] = new String[79];
   private      static String linea0Estado[] = new String[79];
   private      static String linea1Estado[] = new String[79];
   private      static String linea2Estado[] = new String[79];
   private static boolean primera_vez_estado[] = new boolean[79];
   private static boolean residuoEstado[] = new boolean[79];
   private static boolean finEstado[] = new boolean[79];
   private static int punteroEstado[] = new int[79];
   private static String cicloEstado[] = new String[79];
   private static boolean trama1estado[] = new boolean[79];
   private static boolean trama2estado[] = new boolean[79];
   private static boolean trama3estado[] = new boolean[79];
   private static boolean trama4estado[] = new boolean[79];
   private static boolean trama5estado[] = new boolean[79];
   private static boolean tramaFinestado[] = new boolean[79];
   private static JLabel imagen, via1, via2, paralelas1, paralelas2;
   private static JDialog dialogo;

   private static String pinicial[] = new String[4];
   private static String qinicial[] = new String[4];

private static String linea[] = new String[3];
   private static String p[] = new String[4]; //emisor
   private static String q[] = new String[4]; //receptor

   //A continuación se exponen las 32 posibles palabras a
   //transmitir, de las cuales se tomarán aleatoriamente en
   //esta opción 4, y lo mismo para otras opciones, se toma
   //un número aleatorio de palabras igual al número de canales
   //existente en el emisor:

   private static String palabras[] = {"Javier", "GregoR",
          "Biënve", "Juanpe", "Cerdán", "ChuLla", "Abogaö", "Andrés",
          "Piñero", "CañitA", "JuAnaN", "Sirôco", "Litros", "PdroxO",
          "Joseja", "ZayrA_", "Tëleco", "Malgos", "Chendo",
          "Chema_", "Enric_", "Melcón", "Fulgen", "ToñitO",
          "VArgas", "BüRrul", "Verdú_", "Lolica",
          "Mägañâ", "Pacoy_", "Ramón_", "Valero" };

   public void iniciarAleatorio()
   {

          boolean pos[] = new boolean[32];

          linea[0] = "*";
          linea[1] = "+";
          linea[2] = "*";

          //a q[] le asigno estos valores que no son los que
          //aparecerán para el usuario, la razón de que no se
          //ponga directamente *+*+*+ en q[] es por problemas
          //a la hora de recibir la letra, que si se sustituye
          //un *, esa letra aparece en todos los * de ese q[],
          //y lo mismo ocurre con los +, por ello pongo todos
          //los caracteres distintos, y ya en pintarLabels()
          //los "enmascaro":

          for(int k=0; k<q.length; k++)
          {
                 q[k] = "¿&@#$?";
          }

          for(int j=0; j<p.length; j++)
          {
                 pos[j] = false;
```

```
                    }

                    for(int i=0; i<p.length; i++)
                    {
                           int num = (int)(Math.random()*32);
                           if (pos[num] == false) //si es la primera vez que
accedemos a la palabra
                           {
                                  p[i] = palabras[num];
                                  pos[num] = true;
                           }
                           else{i--;}

                    }//llave del tercer for

                    datos[0] = 3;
                    datos[1] = 2;
                    datos[2] = 1;

                    //conviene tener unos valores iniciales invariables, para
                    //a la hora de usar el método reiniciar(), colocar dichos
                    //valores:

                    for(int i=0; i<p.length;i++)
                    {
                           pinicial[i] = p[i];
                           qinicial[i] = q[i];
                    }

             }//llave de iniciarAleatorio()


             public static void main(String []args)
             {
                    new    OpcionA4().show();
             }

      }
```

B.2. OpcionB1

De la OpcionB1 no se pondrá todo, se expondrá el método *viajeCiclo()*, el más importante.

```
public void viajeCiclo(String pe, String qe)
```

```
		{
			String pe1, pe2, pe3, qe1, qe2, qe3;
			String pe4, pe5, pe6, pe7, qe4, qe5, qe6, qe7;
			pe = p[tirada];
			qe = q[tirada];
			pe1 = p[tirada + 1];
			qe1 = q[tirada + 1];
			pe2 = p[tirada + 2];
			qe2 = q[tirada + 2];
			pe3 = p[tirada + 3];
			qe3 = q[tirada + 3];
			pe4 = p[tirada + 4];
			qe4 = q[tirada + 4];
			pe5 = p[tirada + 5];
			qe5 = q[tirada + 5];
			pe6 = p[tirada + 6];
			qe6 = q[tirada + 6];
			pe7 = p[tirada + 7];
			qe7 = q[tirada + 7];


			if (iteración == 0) //1ª pulsación
			{
				if (primera_vez == false)
				{
					linea[5]                                        =
linea[4].valueOf(linea[4].charAt(0));
					linea[4]                                        =
linea[3].valueOf(linea[3].charAt(0));
					linea[2]                                        =
linea[1].valueOf(linea[1].charAt(0));
					linea[1]                                        =
linea[0].valueOf(linea[0].charAt(0));
				}
				if (primera_vez == true)
				{
					linea[5]                                        =
linea[4].valueOf(linea[4].charAt(0));
					linea[4] = "+";
					linea[2]                                        =
linea[1].valueOf(linea[1].charAt(0));
					linea[1]                                        =
linea[0].valueOf(linea[0].charAt(0));
					primera_vez = false;
				}

				if (índice == 192)
				{
					linea[0] = "+";
				}

				else if (fin == false)
				{
					linea[0] = pe2.valueOf(pe2.charAt(puntero));
				}

				else if (fin == true)
				{
					linea[0] = pe2.valueOf(pe2.charAt(0));
				}
				posS = control[0];
				linea[3] = speech[posS];
				ciclo = "Lectura";
				iteración++;
				índice++;
			}
			else if (iteración == 1)
			{
```

```
                        if (índice != 193)
                        {
                              qe6     =     qe6.replace(qe6.charAt(puntero),
linea[5].charAt(0));
                              q[tirada + 6] = qe6;
                        }
                        else
                        {
                              qe6        =        qe6.replace(qe6.charAt(0),
linea[5].charAt(0));
                              q[tirada + 6] = qe6;
                        }
                        if (fin == false)
                        {
                              pe2 = pe2.replace(pe2.charAt(puntero), '+');
                              p[tirada + 2] = pe2;
                        }
                        if (fin == true && índice != 193)
                        {
                              pe2 = pe2.replace(pe2.charAt(0), '+');
                              p[tirada + 2] = pe2;
                        }
                        if (índice == 193)
                        {
                              p[tirada + 2] = "++++++";
                        }
                        speech[0] = linea[2];

                        ciclo = "Escritura";
                        iteración++;
                        índice++;
                  }
                  else if (iteración == 2)
                  {
                        ciclo = "Espera";
                        iteración++;
                        índice++;
                  }
                  else if (iteración == 3)
                  {

                        linea[5] = linea[4].valueOf(linea[4].charAt(0));
                        linea[4] = linea[3].valueOf(linea[3].charAt(0));
                        linea[2] = linea[1].valueOf(linea[1].charAt(0));
                        linea[1] = linea[0].valueOf(linea[0].charAt(0));

                        if (fin == false)
                        {
                              linea[0] = pe3.valueOf(pe3.charAt(puntero));

                        }
                        if (fin == true)
                        {
                              linea[0] = pe3.valueOf(pe3.charAt(0));
                        }


                        posS = control[1];
                        linea[3] = speech[posS];


                        ciclo = "Lectura";
                        iteración++;
                        índice++;
                  }
                  else if (iteración == 4)
                  {
```

```
                        qe7        =        qe7.replace(qe7.charAt(puntero),
linea[5].charAt(0));
                        q[tirada + 7] = qe7;

                        if (fin == false)
                        {
                              pe3 = pe3.replace(pe3.charAt(puntero), '+');
                              p[tirada + 3] = pe3;
                        }
                        if (fin == true)
                        {
                              pe3 = pe3.replace(pe3.charAt(0), '+');
                              p[tirada + 3] = pe3;
                        }

                        speech[1] = linea[2];
                        ciclo = "Escritura";
                        iteración++;
                        índice++;
                  }
                  else if (iteración == 5)
                  {
                        ciclo = "Espera";
                        iteración++;
                        índice++;
                  }
                  else if (iteración == 6)
                  {
                        linea[5] = linea[4].valueOf(linea[4].charAt(0));
                        linea[4] = linea[3].valueOf(linea[3].charAt(0));
                        linea[2] = linea[1].valueOf(linea[1].charAt(0));
                        linea[1] = linea[0].valueOf(linea[0].charAt(0));

                        if (fin == false)
                        {
                              linea[0] = pe4.valueOf(pe4.charAt(puntero));
                        }
                        if (fin == true)
                        {
                              linea[0] = pe4.valueOf(pe4.charAt(0));
                        }
                        posS = control[2];
                        linea[3] = speech[posS];


                        ciclo = "Lectura";
                        iteración++;
                        índice++;
                  }
                  else if (iteración == 7)
                  {
                        qe         =         qe.replace(qe.charAt(puntero),
linea[5].charAt(0));
                        q[tirada] = qe;

                        if (fin == false)
                        {
                              pe4 = pe4.replace(pe4.charAt(puntero), '+');
                              p[tirada + 4] = pe4;
                        }
                        if (fin == true)
                        {
                              pe4 = pe4.replace(pe4.charAt(0), '+');
                              p[tirada + 4] = pe4;
                        }
                        speech[2] = linea[2];

                        ciclo = "Escritura";
```

```
                        iteración++;
                        índice++;
                }
                else if (iteración == 8)
                {
                        ciclo = "Espera";
                        iteración++;
                        índice++;
                }
                else if (iteración == 9)
                {
                        linea[5] = linea[4].valueOf(linea[4].charAt(0));
                        linea[4] = linea[3].valueOf(linea[3].charAt(0));
                        linea[2] = linea[1].valueOf(linea[1].charAt(0));
                        linea[1] = linea[0].valueOf(linea[0].charAt(0));

                        if (fin == false)
                        {
                                linea[0] = pe5.valueOf(pe5.charAt(puntero));
                        }
                        if (fin == true)
                        {
                                linea[0] = pe5.valueOf(pe5.charAt(0));
                        }

                        posS = control[3];
                        linea[3] = speech[posS];

                        ciclo = "Lectura";
                        iteración++;
                        índice++;
                }
                else if (iteración == 10)
                {
                        qe1        =        qe1.replace(qe1.charAt(puntero),
linea[5].charAt(0));
                        q[tirada + 1] = qe1;

                        if (fin == false)
                        {
                                pe5 = pe5.replace(pe5.charAt(puntero), '+');
                                p[tirada + 5] = pe5;
                        }
                        if (fin == true)
                        {
                                pe5 = pe5.replace(pe5.charAt(0), '+');
                                p[tirada + 5] = pe5;
                        }

                        speech[3] = linea[2];
                        ciclo = "Escritura";
                        iteración++;
                        índice++;
                }
                else if (iteración == 11)
                {
                        ciclo = "Espera";
                        iteración++;
                        índice++;
                }
                else if (iteración == 12)
                {
                        linea[5] = linea[4].valueOf(linea[4].charAt(0));
                        linea[4] = linea[3].valueOf(linea[3].charAt(0));
                        linea[2] = linea[1].valueOf(linea[1].charAt(0));
                        linea[1] = linea[0].valueOf(linea[0].charAt(0));

                        if (fin == false)
```

```
                        {
                                linea[0] = pe6.valueOf(pe6.charAt(puntero));
                        }
                        if (fin == true)
                        {
                                linea[0] = pe6.valueOf(pe6.charAt(0));
                        }
                        posS = control[4];
                        linea[3] = speech[posS];

                        ciclo = "Lectura";
                        iteración++;
                        índice++;
                }
                else if (iteración == 13)
                {
                        qe2        =        qe2.replace(qe2.charAt(puntero),
linea[5].charAt(0));
                        q[tirada + 2] = qe2;

                        if (fin == false)
                        {
                                pe6 = pe6.replace(pe6.charAt(puntero), '+');
                                p[tirada + 6] = pe6;
                        }
                        if (fin == true)
                        {
                                pe6 = pe6.replace(pe6.charAt(0), '+');
                                p[tirada + 6] = pe6;
                        }
                        speech[4] = linea[2];

                        ciclo = "Escritura";
                        iteración++;
                        índice++;
                }
                else if (iteración == 14)
                {
                        ciclo = "Espera";
                        iteración++;
                        índice++;
                }
                else if (iteración == 15)
                {
                        linea[5] = linea[4].valueOf(linea[4].charAt(0));
                        linea[4] = linea[3].valueOf(linea[3].charAt(0));
                        linea[2] = linea[1].valueOf(linea[1].charAt(0));
                        linea[1] = linea[0].valueOf(linea[0].charAt(0));

                        if (fin == false)
                        {
                                linea[0] = pe7.valueOf(pe7.charAt(puntero));
                        }
                        if (fin == true)
                        {
                                linea[0] = pe7.valueOf(pe7.charAt(0));
                        }

                        posS = control[5];
                        linea[3] = speech[posS];

                        ciclo = "Lectura";
                        iteración++;
                        índice++;
                }
                else if (iteración == 16)
                {
```

```
                        qe3        =        qe3.replace(qe3.charAt(puntero),
linea[5].charAt(0));
                        q[tirada + 3] = qe3;

                        if (fin == false)
                        {
                              pe7 = pe7.replace(pe7.charAt(puntero), '+');
                              p[tirada + 7] = pe7;
                        }
                        if (fin == true)
                        {
                              pe7 = pe7.replace(pe7.charAt(0), '+');
                              p[tirada + 7] = pe7;
                        }

                        speech[5] = linea[2];
                        ciclo = "Escritura";
                        iteración++;
                        índice++;
                  }
                  else if (iteración == 17)
                  {
                        ciclo = "Espera";
                        iteración++;
                        if (puntero == 5){fin = true;}
                        puntero++;
                        residuo = true;
                        índice++;
                  }
                  else if (iteración == 18)
                  {
                        linea[5] = linea[4].valueOf(linea[4].charAt(0));
                        linea[4] = linea[3].valueOf(linea[3].charAt(0));
                        linea[2] = linea[1].valueOf(linea[1].charAt(0));
                        linea[1] = linea[0].valueOf(linea[0].charAt(0));


                        if (fin == false)
                        {
                              linea[0]   =   pe.valueOf(pe.charAt(puntero   -
1));
                        }
                        if (fin == true)
                        {
                              linea[0] = pe.valueOf(pe.charAt(5));
                        }

                        posS = control[6];
                        linea[3] = speech[posS];
                        ciclo = "Lectura";
                        iteración++;
                        índice++;
                  }
                  else if (iteración == 19)
                  {

                        qe4    =    qe4.replace(qe4.charAt(puntero    -    1),
linea[5].charAt(0));
                        q[tirada + 4] = qe4;

                        if (fin == false)
                        {
                              pe = pe.replace(pe.charAt(puntero - 1), '+');
                              p[tirada] = pe;
                        }
                        if (fin == true)
                        {
                              pe = pe.replace(pe.charAt(5), '+');
```

```
                        p[tirada] = pe;
                }
                speech[6] = linea[2];

                ciclo = "Escritura";
                iteración++;
                índice++;
        }
        else if (iteración == 20)
        {
                ciclo = "Espera";
                iteración++;
                índice++;
        }
        else if (iteración == 21)
        {
                linea[5] = linea[4].valueOf(linea[4].charAt(0));
                linea[4] = linea[3].valueOf(linea[3].charAt(0));
                linea[2] = linea[1].valueOf(linea[1].charAt(0));
                linea[1] = linea[0].valueOf(linea[0].charAt(0));


                if (fin == false)
                {
                        linea[0]  =  pe1.valueOf(pe1.charAt(puntero  -
1));
                }
                if (fin == true)
                {
                        linea[0] = pe1.valueOf(pe1.charAt(5));
                }


                posS = control[7];
                linea[3] = speech[posS];
                ciclo = "Lectura";
                iteración++;
                índice++;
        }
        else if (iteración == 22)
        {
                qe5    =    qe5.replace(qe5.charAt(puntero    -    1),
linea[5].charAt(0));
                q[tirada + 5] = qe5;

                if (fin == false)
                {
                        pe1  =  pe1.replace(pe1.charAt(puntero  -  1),
'+');
                        p[tirada + 1] = pe1;
                }
                if (fin == true)
                {
                        pe1 = pe1.replace(pe1.charAt(5), '+');
                        p[tirada + 1] = pe1;
                }
                speech[7] = linea[2];

                ciclo = "Escritura";
                iteración++;
                índice++;
        }
        else if (iteración == 23)
        {
                ciclo = "Espera";
                if (índice == 191)
                {
                        for(int i=0;i<p.length;i++)
```

```
                        {
                                p[i] = pinicial[i];
                        }
                }
                iteración = 0;
                índice++;
        }
}
```

B.3. OpcionB2

Al igual que en la anterior opción, sólo se explicará el método *viajeCiclo().*

```
	public void viajeCiclo(String pe, String qe)
		{
			if (sincronizados)
			{
				if (iteración == 14 && apunterado)
				{
					punteroQ++;
				}
			}


			else if (ts2 == inicial && sincronizado)
			{
				if (iteración == 1||iteración == 4||iteración == 7
				||iteración == 10||iteración == 13||iteración == 16
				||iteración == 19||iteración == 22)
				{
					punteroQ++;
				}
			}



			if (iteración == 0)
			{

				if (primera_vez == false)
				{
					linea[5]                                        =
linea[4].valueOf(linea[4].charAt(0));
					linea[4]                                        =
linea[3].valueOf(linea[3].charAt(0));
					linea[2]                                        =
linea[1].valueOf(linea[1].charAt(0));
					linea[1]                                        =
linea[0].valueOf(linea[0].charAt(0));
				}
				if (primera_vez == true)
				{
					linea[5]                                        =
linea[4].valueOf(linea[4].charAt(0));
					linea[4] = "+";
					linea[2]                                        =
linea[1].valueOf(linea[1].charAt(0));
					linea[1]                                        =
linea[0].valueOf(linea[0].charAt(0));
					primera_vez = false;
				}

				if (ts1 == 7){ts1 = 0;}
				else{ts1++;}

				if (ts2 == 7){ts2 = 0;}
				else{ts2++;}

				if (sincronizado == true)
				{
					ts3++;
				}


				linea[0] = p[ts1].valueOf(p[ts1].charAt(puntSinc));
				if (puntSinc == 7){puntSinc = 0;}
				else{puntSinc++;}

				if (sincronizado == true)
				{
					posS = control[ts3];
```

```
                              if (control[ts3] == 8){linea[3] = "Ç";}
                              else{linea[3] = speech[posS];
                              aux[6] = speech[posS];}
                        }
                        else
                        {
                              posS = control[0];
                              if (control[0] == 8){linea[3] = "Ç";}
                              else{linea[3] = speech[posS];
                              aux[6] = speech[posS];}
                        }



                        canal[4] = true;
                        ciclo = "Lectura";
                        iteración++;
                        índice++;

                  }
                  else if (iteración == 1)
                  {

                        q[ts2]   =   q[ts2].replace(q[ts2].charAt(punteroQ),
linea[5].charAt(0));

                        speech[posR] = linea[2];
                        ciclo = "Escritura";
                        iteración++;
                        índice++;
                  }
                  else if (iteración == 2)
                  {
                        if (posR == 7){posR = 0;}
                        else{posR++;}
                        ciclo = "Espera";
                        iteración++;
                        índice++;
                  }
                  else if (iteración == 3)
                  {
                        ts1++;
                        if (ts2 == 7){ts2 = 0;}
                        else{ts2++;}

                        if (sincronizado == true)
                        {
                              ts3++;
                        }

                        if (primera_vez == false)
                        {
                        linea[5] = linea[4].valueOf(linea[4].charAt(0));
                        linea[4] = linea[3].valueOf(linea[3].charAt(0));
                        linea[2] = linea[1].valueOf(linea[1].charAt(0));
                        linea[1] = linea[0].valueOf(linea[0].charAt(0));
                        }
                        if (primera_vez == true)
                        {
                              linea[5]                                    =
linea[4].valueOf(linea[4].charAt(0));
                              linea[4] = "+";
                              linea[2]                                    =
linea[1].valueOf(linea[1].charAt(0));
                              linea[1]                                    =
linea[0].valueOf(linea[0].charAt(0));
                              primera_vez = false;
                        }
```

```
                        if (canal[1] == true)
                        {
                              if (puntero <= 5)
                              {
                                    linea[0]                                    =
p[ts1].valueOf(p[ts1].charAt(puntero));
                              }
                              else
                              {
                                    linea[0] = "+";
                              }
                        }

                        if (canal[1] == false)
                        {
                              linea[0] = "*";
                        }

                        if (sincronizado == true)
                        {
                              posS = control[ts3];
                              if (control[ts3] == 8){linea[3] = "Ç";}
                              else{linea[3] = speech[posS];}
                        }
                        else
                        {
                              posS = control[1];
                              if (control[1] == 8){linea[3] = "Ç";}
                              else{linea[3] = speech[posS];}
                        }
                        ciclo = "Lectura";
                        iteración++;
                        índice++;

                  }
                  else if (iteración == 4)
                  {
                        if (canal[1] == true)
                        {
                              if (puntero <= 5)
                              {
                                    p[ts1]                                      =
p[ts1].replace(p[ts1].charAt(puntero), '+');
                              }

                        }

                        q[ts2]   =   q[ts2].replace(q[ts2].charAt(punteroQ),
linea[5].charAt(0));

                        speech[posR] = linea[2];
                        ciclo = "Escritura";
                        iteración++;
                        índice++;
                  }
                  else if (iteración == 5)
                  {
                        if (posR == 7){posR = 0;}
                        else{posR++;}

                        ciclo = "Espera";
                        iteración++;
                        índice++;
                  }
                  else if (iteración == 6)
                  {
                        ts1++;
                        if (ts2 == 7){ts2 = 0;}
```

```
                        else{ts2++;}

                        if (ts1 == 2) //
                        {
                              sincronizado = true;
                        }

                        if (sincronizado == true)
                        {
                              ts3 = 0;
                        }

                        if (primera_vez == false)
                        {
                        linea[5] = linea[4].valueOf(linea[4].charAt(0));
                        linea[4] = linea[3].valueOf(linea[3].charAt(0));
                        linea[2] = linea[1].valueOf(linea[1].charAt(0));
                        linea[1] = linea[0].valueOf(linea[0].charAt(0));
                        }
                        if (primera_vez == true)
                        {
                              linea[5]                                    =
linea[4].valueOf(linea[4].charAt(0));
                              linea[4] = "+";
                              linea[2]                                    =
linea[1].valueOf(linea[1].charAt(0));
                              linea[1]                                    =
linea[0].valueOf(linea[0].charAt(0));
                              primera_vez = false;
                        }

                        if (canal[2] == true)
                        {
                              if (puntero <= 6)
                              {
                                    linea[0]                              =
p[ts1].valueOf(p[ts1].charAt(puntero - 1));
                              }
                              else
                              {
                                    linea[0] = "+";
                              }
                        }
                        if (canal[2] == false)
                        {
                              linea[0] = "*";
                        }

                        if (sincronizado == true)
                        {
                              posS = control[ts3];
                              if (control[ts3] == 8){linea[3] = "Ç";}
                              else{linea[3] = speech[posS];}
                        }
                        else
                        {
                              posS = control[2];
                              if (control[2] == 8){linea[3] = "Ç";}
                              else{linea[3] = speech[posS];}
                        }
                        ciclo = "Lectura";
                        iteración++;
                        índice++;
                  }

                  else if (iteración == 7)
                  {
                        if (canal[2] == true)
```

```
                        {
                                if (puntero <= 6)
                                {
                                        p[ts1]                                          =
p[ts1].replace(p[ts1].charAt(puntero - 1), '+');
                                }
                        }


                        q[ts2]   =   q[ts2].replace(q[ts2].charAt(punteroQ),
linea[5].charAt(0));


                        if (sincronizado == true)
                        {
                                posR = 0;
                        }

                        speech[posR] = linea[2];
                        ciclo = "Escritura";
                        iteración++;
                        índice++;
                }
                else if (iteración == 8)
                {
                        if (posR == 7){posR = 0;}
                        else{posR++;}
                        ciclo = "Espera";
                        iteración++;
                        índice++;
                }
                else if (iteración == 9)
                {
                        ts1++;
                        if (ts2 == 7){ts2 = 0;}
                        else{ts2++;}

                        if (sincronizado == true)
                        {
                                ts3++;
                        }

                        if (primera_vez == false)
                        {
                        linea[5] = linea[4].valueOf(linea[4].charAt(0));
                        linea[4] = linea[3].valueOf(linea[3].charAt(0));
                        linea[2] = linea[1].valueOf(linea[1].charAt(0));
                        linea[1] = linea[0].valueOf(linea[0].charAt(0));
                        }
                        if (primera_vez == true)
                        {
                                linea[5]                                        =
linea[4].valueOf(linea[4].charAt(0));
                                linea[4] = "+";
                                linea[2]                                        =
linea[1].valueOf(linea[1].charAt(0));
                                linea[1]                                        =
linea[0].valueOf(linea[0].charAt(0));
                                primera_vez = false;
                        }

                        if (canal[3] == true)
                        {
                                if (puntero <= 7)
                                {
                                        linea[0]                                =
p[ts1].valueOf(p[ts1].charAt(puntero - 2));
                                }
```

```
                        else
                        {
                              linea[0] = "+";
                        }
                  }
                  if (canal[3] == false)
                  {
                        linea[0] = "*";
                  }

                  if (sincronizado == true)
                  {
                        posS = control[ts3];
                        if (control[ts3] == 8){linea[3] = "Ç";}
                        else{linea[3] = speech[posS];}
                  }
                  else
                  {
                        posS = control[3];
                        if (control[3] == 8){linea[3] = "Ç";}
                        else{linea[3] = speech[posS];}
                  }
                  ciclo = "Lectura";
                  iteración++;
                  índice++;
            }
            else if (iteración == 10)
            {
                  if (canal[3] == true)
                  {
                        if (puntero <= 7)
                        {
                              p[ts1]                                    =
p[ts1].replace(p[ts1].charAt(puntero - 2), '+');
                        }
                  }


                  q[ts2]   =   q[ts2].replace(q[ts2].charAt(punteroQ),
linea[5].charAt(0));

                  speech[posR] = linea[2];
                  ciclo = "Escritura";
                  iteración++;
                  índice++;
            }
            else if (iteración == 11)
            {
                  if (posR == 7){posR = 0;}
                  else{posR++;}

                  ciclo = "Espera";
                  iteración++;
                  índice++;
            }
            else if (iteración == 12)
            {

                  if (ts2 == 7){ts2 = 0;}
                  else{ts2++;}
                  ts1++;

                  if (sincronizado == true)
                  {
                        ts3++;
                  }

                  if (primera_vez == false)
```

```
                        {
                                linea[5]                                        =
linea[4].valueOf(linea[4].charAt(0));
                                linea[4]                                        =
linea[3].valueOf(linea[3].charAt(0));
                                linea[2]                                        =
linea[1].valueOf(linea[1].charAt(0));
                                linea[1]                                        =
linea[0].valueOf(linea[0].charAt(0));
                        }
                        if (primera_vez == true)
                        {
                                linea[5]                                        =
linea[4].valueOf(linea[4].charAt(0));
                                linea[4] = "+";
                                linea[2]                                        =
linea[1].valueOf(linea[1].charAt(0));
                                linea[1]                                        =
linea[0].valueOf(linea[0].charAt(0));
                                primera_vez = false;
                        }

                        if (canal[4] == true)
                        {
                                linea[0]                                        =
p[ts1].valueOf(p[ts1].charAt(puntSeñ));
                                if (puntSeñ == 7){puntSeñ = 0;}
                                else
                                {
                                        puntSeñ++;

                                        if (canal[6] == true){canal[7] = true;}
                                        if (canal[5] == true){canal[6] = true;}
                                        if (canal[3] == true){canal[5] = true;}
                                        if (canal[2] == true){canal[3] = true;}
                                        if (canal[1] == true){canal[2] = true;}
                                        if (puntSeñ == 2){canal[1] = true;}
                                }
                        }
                        if (canal[4] == false)
                        {
                                linea[0] = "*";
                        }


                        //aquí es donde el contador del receptor se
                        //sincroniza con la fuente:

                        if (linea[5].equals("^"))
                        {
                                ts2 = 0;
                                sincronizados = true;
                        }

                        if (sincronizado == true)
                        {
                                posS = control[ts3];
                                if (control[ts3] == 8){linea[3] = "Ç";}
                                else{linea[3] = speech[posS];}
                        }
                        else
                        {
                                posS = control[4];
                                if (control[4] == 8){linea[3] = "Ç";}
                                else{linea[3] = speech[posS];}
                        }

                        ciclo = "Lectura";
```

```
                        iteración++;
                        índice++;

                }
                else if (iteración == 13)
                {
                        if (linea[5].equals("^"))
                        {
                                apunterado = true;
                        }

                        q[ts2]   =   q[ts2].replace(q[ts2].charAt(punteroQ),
linea[5].charAt(0));



                        speech[posR] = linea[2];
                        ciclo = "Escritura";
                        iteración++;
                        índice++;
                }
                else if (iteración == 14)
                {
                        if (posR == 7){posR = 0;}
                        else{posR++;}

                        ciclo = "Espera";
                        iteración++;
                        índice++;
                }
                else if (iteración == 15)
                {
                        ts1++;
                        if (ts2 == 7){ts2 = 0;}
                        else{ts2++;}



                        if (primera_vez == false)
                        {
                                linea[5]                                        =
linea[4].valueOf(linea[4].charAt(0));
                                linea[4]                                        =
linea[3].valueOf(linea[3].charAt(0));
                                linea[2]                                        =
linea[1].valueOf(linea[1].charAt(0));
                                linea[1]                                        =
linea[0].valueOf(linea[0].charAt(0));
                        }
                        if (primera_vez == true)
                        {
                                linea[5]                                        =
linea[4].valueOf(linea[4].charAt(0));
                                linea[4] = "+";
                                linea[2]                                        =
linea[1].valueOf(linea[1].charAt(0));
                                linea[1]                                        =
linea[0].valueOf(linea[0].charAt(0));
                                primera_vez = false;
                        }

                        if (canal[5] == true && puntero >= 3)
                        {
                                if (puntero <= 8)
                                {
                                        linea[0]                                =
p[ts1].valueOf(p[ts1].charAt(puntero - 3));
                                }
```

```
                        else
                        {
                              linea[0] = "+";
                        }
                  }
                  else if(canal[5] == true && puntero < 3)
                  {
                        linea[0] = "*";
                  }

                  if (canal[5] == false)
                  {
                        linea[0] = "*";
                  }

                  if (sincronizado == true)
                  {
                        ts3++;
                  }

                  if (sincronizado == true)
                  {
                        posS = control[ts3];
                        if (control[ts3] == 8){linea[3] = "Ç";}
                        else{linea[3] = speech[posS];}
                  }
                  else
                  {
                        posS = control[5];
                        if (control[5] == 8){linea[3] = "Ç";}
                        else{linea[3] = speech[posS];}
                  }
                  ciclo = "Lectura";
                  iteración++;
                  índice++;



            }
            else if (iteración == 16)
            {
                  if (canal[5] == true && puntero >= 3)
                  {
                        if (puntero <= 8)
                        {
                              p[ts1]                                    =
p[ts1].replace(p[ts1].charAt(puntero - 3), '+');
                        }
                  }


                  q[ts2]   =   q[ts2].replace(q[ts2].charAt(punteroQ),
linea[5].charAt(0));

                  speech[posR] = linea[2];
                  ciclo = "Escritura";
                  iteración++;
                  índice++;
            }
            else if (iteración == 17)
            {
                  if (posR == 7){posR = 0;}
                  else{posR++;}

                  ciclo = "Espera";
                  iteración++;
                  índice++;
```

```
                    }
                    else if (iteración == 18)
                    {
                           ts1++;
                           if (ts2 == 7){ts2 = 0;}
                           else{ts2++;}

                           if (sincronizado == true)
                           {
                                  señalizado = true;
                                  ts3++;
                           }

                           if (primera_vez == false)
                           {
                                  linea[5]                                        =
linea[4].valueOf(linea[4].charAt(0));
                                  linea[4]                                        =
linea[3].valueOf(linea[3].charAt(0));
                                  linea[2]                                        =
linea[1].valueOf(linea[1].charAt(0));
                                  linea[1]                                        =
linea[0].valueOf(linea[0].charAt(0));
                           }
                           if (primera_vez == true)
                           {
                                  linea[5]                                        =
linea[4].valueOf(linea[4].charAt(0));
                                  linea[4] = "+";
                                  linea[2]                                        =
linea[1].valueOf(linea[1].charAt(0));
                                  linea[1]                                        =
linea[0].valueOf(linea[0].charAt(0));
                                  primera_vez = false;
                           }

                           if (canal[6] == true && puntero >= 4)
                           {
                                  if (puntero <= 9)
                                  {
                                         linea[0]                                 =
p[ts1].valueOf(p[ts1].charAt(puntero - 4));
                                  }
                                  else
                                  {
                                         linea[0] = "+";
                                  }
                           }
                           else if (canal[6] == true && puntero < 4)
                           {
                                  linea[0] = "*";
                           }
                           if (canal[6] == false)
                           {
                                  linea[0] = "*";
                           }

                           if (sincronizado == true)
                           {
                                  posS = control[ts3];
                                  if (control[ts3] == 8){linea[3] = "Ç";}
                                  else{linea[3] = speech[posS];}
                           }
                           else
                           {
                                  posS = control[6];
                                  if (control[6] == 8){linea[3] = "Ç";}
                                  else{linea[3] = speech[posS];}
```

```
                        }

                        ciclo = "Lectura";
                        iteración++;
                        índice++;



                        if (señalizado == true)
                        {
                              ts4 = Integer.parseInt(linea[2]);
                              control[ts4] = contadorRAM;
                              if (contadorRAM == 7)
                              {
                                    contadorRAM = 0;
                              }
                              else
                              {
                                    contadorRAM++;
                              }
                        }

                  }

                  else if (iteración == 19)
                  {
                        if (canal[6] == true && puntero >= 4)
                        {
                              if (puntero <= 9)
                              {
                                    p[ts1]                                    =
p[ts1].replace(p[ts1].charAt(puntero - 4), '+');
                              }
                        }


                        q[ts2]   =   q[ts2].replace(q[ts2].charAt(punteroQ),
linea[5].charAt(0));


                        speech[posR] = linea[2];
                        ciclo = "Escritura";
                        iteración++;
                        índice++;
                  }
                  else if (iteración == 20)
                  {
                        if (posR == 7){posR = 0;}
                        else{posR++;}

                        ciclo = "Espera";
                        iteración++;
                        índice++;
                  }
                  else if (iteración == 21)
                  {
                        ts1++;
                        if (ts2 == 7){ts2 = 0;}
                        else{ts2++;}

                        if (sincronizado == true)
                        {
                              ts3++;
                        }

                        if (primera_vez == false)
                        {
```

```
                        linea[5] = linea[4].valueOf(linea[4].charAt(0));
                        linea[4] = linea[3].valueOf(linea[3].charAt(0));
                        linea[2] = linea[1].valueOf(linea[1].charAt(0));
                        linea[1] = linea[0].valueOf(linea[0].charAt(0));
                        }
                        if (primera_vez == true)
                        {
                              linea[5]                                          =
linea[4].valueOf(linea[4].charAt(0));
                              linea[4] = "+";
                              linea[2]                                          =
linea[1].valueOf(linea[1].charAt(0));
                              linea[1]                                          =
linea[0].valueOf(linea[0].charAt(0));
                              primera_vez = false;
                        }

                        if (canal[7] == true && puntero >= 5)
                        {
                              if (puntero <= 10)
                              {
                                    linea[0]                                    =
p[ts1].valueOf(p[ts1].charAt(puntero - 5));
                              }
                              else
                              {
                                    linea[0] = "+";
                              }
                        }
                        else if (canal[7] == true && puntero < 5)
                        {
                              linea[0] = "*";
                        }
                        if (canal[7] == false)
                        {
                              linea[0] = "*";
                        }

                        if (sincronizado == true)
                        {
                              posS = control[ts3];
                              if (control[ts3] == 8){linea[3] = "Ç";}
                              else{linea[3] = speech[posS];}
                        }
                        else
                        {
                              posS = control[7];
                              if (control[7] == 8){linea[3] = "Ç";}
                              else{linea[3] = speech[posS];}
                        }
                        ciclo = "Lectura";
                        iteración++;
                        índice++;

                  }
                  else if (iteración == 22)
                  {
                        if (canal[7] == true && puntero >= 5)
                        {
                              if (puntero <= 10)
                              {
                                    p[ts1]                                      =
p[ts1].replace(p[ts1].charAt(puntero - 5), '+');
                              }
                        }

                        q[ts2]   =   q[ts2].replace(q[ts2].charAt(punteroQ),
linea[5].charAt(0));
```

```
                speech[posR] = linea[2];
                ciclo = "Escritura";
                iteración++;
                índice++;
        }
        else if (iteración == 23)
        {
                if (posR == 7){posR = 0;}
                else{posR++;}

                ciclo = "Espera";
                iteración = 0;
                índice++;
                if (canal[2] == true)
                {
                        puntero++;
                }
        }
}
```

B.4. General

La clase General se expondrá en su totalidad.

```
import java.lang.String;
import java.awt.*;
import javax.swing.*;
import java.io.*;

public class General extends javax.swing.JFrame
{
       JLabel imagen;

       OpcionA4 a4;
       OpcionA8 a8;
       OpcionA16 a16;
       OpcionA32 a32;
       OpcionB1 b1;
       OpcionB2 b2;

       JButton ra4, ra8, ra16, ra32, rb1, rb2;

       JLabel l1, l2;

    public General()
       {
        iniciarComponentes();
             setSize(800, 600);
             setResizable(false);

             try
             {

UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAnd
Feel");
                    SwingUtilities.updateComponentTreeUI(this);
             }
             catch (Exception e){}
    }


       private javax.swing.JButton button1, button2, atrás;
       private  javax.swing.JLabel  texto1,  texto2,  texto3,  texto4,
texto5;

    private void iniciarComponentes()
       {
             button1 = new javax.swing.JButton();
             button2 = new javax.swing.JButton();
             atrás = new javax.swing.JButton();
             texto1 = new javax.swing.JLabel();
             texto2 = new javax.swing.JLabel();
             texto3 = new javax.swing.JLabel();
             texto4 = new javax.swing.JLabel();
             texto5 = new javax.swing.JLabel();
             ra4 = new javax.swing.JButton();
             ra8 = new javax.swing.JButton();
             ra16 = new javax.swing.JButton();
             ra32 = new javax.swing.JButton();
             rb1 = new javax.swing.JButton();
             rb2 = new javax.swing.JButton();

             l1 = new JLabel();
             l2 = new JLabel();

             l1.setText("SIMUTEMP");
             l1.setFont(new java.awt.Font("Arial Black", 1, 36));
             l1.setForeground(Color.yellow);
             l1.setBounds(280, 180, 250, 100);
             getContentPane().add(l1);
```

```
                    l2.setText("DEMOSTRACIÓN Y EJERCICIOS DE UN CONMUTADOR
TEMPORAL");
                    l2.setFont(new java.awt.Font("Dialog", 1, 18));
                    l2.setForeground(Color.cyan);
                    l2.setBounds(110, 250, 600, 100);
                    getContentPane().add(l2);

                    imagen            =            new            JLabel(new
ImageIcon(getClass().getResource("/definitivo.JPG")));
                    imagen.setBounds(290, 150, 219, 242);
                    getContentPane().add(imagen);


                    ra4.setText("TDM de 4 canales");
               ra4.addActionListener(new java.awt.event.ActionListener() {
                   public void actionPerformed(java.awt.event.ActionEvent evt)
{
                       opcionA4Handler(evt);
                   }
               });

                    ra4.setBounds(30, 200, 150, 30);
                    getContentPane().add(ra4);

                    ra8.setText("TDM de 8 canales");
               ra8.addActionListener(new java.awt.event.ActionListener() {
                   public void actionPerformed(java.awt.event.ActionEvent evt)
{
                       opcionA8Handler(evt);
                   }
               });

                    ra8.setBounds(50, 240, 150, 30);
                    getContentPane().add(ra8);

                    ra16.setText("TDM de 16 canales");
               ra16.addActionListener(new java.awt.event.ActionListener() {
                   public void actionPerformed(java.awt.event.ActionEvent evt)
{
                       opcionA16Handler(evt);
                   }
               });

                    ra16.setBounds(70, 280, 150, 30);
                    getContentPane().add(ra16);

                    ra32.setText("TDM de 32 canales");
               ra32.addActionListener(new java.awt.event.ActionListener() {
                   public void actionPerformed(java.awt.event.ActionEvent evt)
{
                       opcionA32Handler(evt);
                   }
               });

                    ra32.setBounds(90, 320, 150, 30);
                    getContentPane().add(ra32);

                    rb1.setText("Demostración de Conmutación en el Tiempo");
               rb1.addActionListener(new java.awt.event.ActionListener() {
                   public void actionPerformed(java.awt.event.ActionEvent evt)
{
                       opcionB1Handler(evt);
                   }
               });

                    rb1.setBounds(520, 230, 250, 30);
                    getContentPane().add(rb1);
```

```
                rb2.setText("Demostración de Sincronismo y Señalización");
            rb2.addActionListener(new java.awt.event.ActionListener() {
                public void actionPerformed(java.awt.event.ActionEvent evt)
{
                    opcionB2Handler(evt);
                }
            });

                rb2.setBounds(520, 270, 250, 30);
                getContentPane().add(rb2);



                button1.setText("Opción A: Multiplexación por División en
el Tiempo (TDM)");
                button1.addActionListener(new
java.awt.event.ActionListener() {
                public void actionPerformed(java.awt.event.ActionEvent evt)
{
                    button1Handler(evt);
                }
            });

                getContentPane().add(button1);
            button1.setBounds(200, 410, 400, 30);

                button2.setText("Opción B: Conmutación Temporal con TDM");
                button2.addActionListener(new
java.awt.event.ActionListener() {
                public void actionPerformed(java.awt.event.ActionEvent evt)
{
                    button2Handler(evt);
                }
            });

                getContentPane().add(button2);
            button2.setBounds(200, 450, 400, 30);

                atrás.setText("<= Atrás");
                atrás.addActionListener(new java.awt.event.ActionListener()
{
                public void actionPerformed(java.awt.event.ActionEvent evt)
{
                    atrásHandler(evt);
                }
            });

                getContentPane().add(atrás);
            atrás.setBounds(360, 430, 80, 30);



            addWindowListener(new java.awt.event.WindowAdapter() {
                public void windowClosing(java.awt.event.WindowEvent evt) {
                    exitForm(evt);
                }
            });

                getContentPane().setLayout(null);
            getContentPane().setBackground(new java.awt.Color(51, 51, 255));
                setTitle("SIMUTEMP");
            addWindowListener(new java.awt.event.WindowAdapter() {
              public void windowClosing(java.awt.event.WindowEvent evt) {
                    exitForm(evt);
                }
            });

                texto1.setText("UNIVERSIDAD POLITÉCNICA DE CARTAGENA");
```

```
                    texto1.setFont(new java.awt.Font("Dialog", 1, 18));
                    texto1.setForeground(Color.white);
                    texto1.reshape(195, 20, 440, 30);
                    getContentPane().add(texto1);

                    texto2.setText("Escuela Técnica Superior de Ingeniería de
Telecomunicación");
                    texto2.setFont(new java.awt.Font("Dialog", 1, 14));
                    texto2.setForeground(Color.white);
                    texto2.reshape(185, 50, 450, 30);
                    getContentPane().add(texto2);

                    texto3.setText("Área de Ingeniería Telemática");
                    texto3.setFont(new java.awt.Font("Dialog", 1, 14));
                    texto3.setForeground(Color.white);
                    texto3.reshape(290, 80, 250, 30);
                    getContentPane().add(texto3);

                    texto4.setText("DEMOSTRACIONES Y EJERCICIOS DE CONMUTACIÓN
EN EL TIEMPO");

                    getContentPane().add(texto4);

                    JPanel panelNombre = new JPanel();
                    panelNombre.setBounds(0, 510, 800, 60);
                    panelNombre.setBackground(new java.awt.Color(0, 0, 128));
                    panelNombre.setLayout(null);
                    getContentPane().add(panelNombre);

                    texto5.setText("Autor:    Francisco    Javier    Martínez
Mercader");
                    texto5.setForeground(Color.white);
                    texto5.setFont(new java.awt.Font("Dialog", 3, 14));
                    texto5.setBounds(254, 14, 300, 30);
                    panelNombre.add(texto5);









                    ra4.setVisible(false);
                    ra8.setVisible(false);
                    ra16.setVisible(false);
                    ra32.setVisible(false);
                    rb1.setVisible(false);
                    rb2.setVisible(false);
                    atrás.setVisible(false);

                pack();
            }


               private void opcionA4Handler(java.awt.event.ActionEvent evt)
               {
               a4 = new OpcionA4();

                     a4.show();
                     (this).hide();
            }
               private void opcionA8Handler(java.awt.event.ActionEvent evt)
               {
                a8 = new OpcionA8();

                     a8.show();
                     (this).hide();
```

```
}
   private void opcionA16Handler(java.awt.event.ActionEvent evt)
   {
    a16 = new OpcionA16();

          a16.show();
          (this).hide();
}
   private void opcionA32Handler(java.awt.event.ActionEvent evt)
   {
    a32 = new OpcionA32();

          a32.show();
          (this).hide();
}
   private void opcionB1Handler(java.awt.event.ActionEvent evt)
   {
    b1 = new OpcionB1();

          b1.show();
          (this).hide();
}
   private void opcionB2Handler(java.awt.event.ActionEvent evt)
   {
    b2 = new OpcionB2();

          b2.show();
          (this).hide();
}
   private void atrásHandler(java.awt.event.ActionEvent evt)
   {
   ra4.setVisible(false);
          ra8.setVisible(false);
          ra16.setVisible(false);
          ra32.setVisible(false);
          rb1.setVisible(false);
          rb2.setVisible(false);
          atrás.setVisible(false);
          button1.setVisible(true);
          button2.setVisible(true);
          l1.setVisible(true);
          l2.setVisible(true);
}


   private void button1Handler(java.awt.event.ActionEvent evt)
   {
          ra4.setVisible(true);
          ra8.setVisible(true);
          ra16.setVisible(true);
          ra32.setVisible(true);
          atrás.setVisible(true);

          button1.setVisible(false);
          button2.setVisible(false);
          l1.setVisible(false);
          l2.setVisible(false);
   }

   private void button2Handler(java.awt.event.ActionEvent evt)
   {
          rb1.setVisible(true);
          rb2.setVisible(true);
          atrás.setVisible(true);

          button1.setVisible(false);
          button2.setVisible(false);
          l1.setVisible(false);
```

```
            l2.setVisible(false);
      }

    private void exitForm(java.awt.event.WindowEvent evt)
      {
       System.exit(0);
    }



    public static void main(String args[])
      {
       new General().show();
    }
}
```

Anexo C
Trabajo a realizar por el alumno

1. Principios del múltiplex síncrono por división en el tiempo (STDM)

Seleccionando la opción A del simulador:

1. Dibujar y describir los módulos del esquema STDM que se muestra en pantalla. ¿Por qué se realizan conversiones paralelo-serie y serie-paralelo en la fuente y en el receptor respectivamente?

2. ¿Cuántos ciclos hay en cada slot y qué operaciones se hacen en cada ciclo?. ¿Por qué es necesario el ciclo de espera?

3. ¿Cuántos slots hay en cada trama?¿Por qué?

4. ¿Cuál es el desfase en slots entre fuente y receptor?. ¿A qué se debe dicho desfase?

5. ¿Cuántas tramas se requieren para enviar una palabra de las usadas en el simulador entre fuente y receptor?. ¿Por qué?

2. Etapa de conmutación T

Seleccionando la opción B del simulador y la demostración del conmutador temporal:

1. Dibujar la etapa T y describir todos los elementos que la integran, indicando la función que realizan así como el contenido de la memoria de control. ¿Qué relación existe entre el origen y el destino de una conexión y la posición y el contenido de la memoria de control?. ¿Cuál es el equivalente analógico de esta etapa T?

Mediante una ejecución ciclo a ciclo, slot s lot, o trama a trama, según convenga, responder a las siguientes cuestiones:

2. Si el conmutador está en el slot 4, en qué slot se encuentra cada una de las partes del sistema: la fuente, el primer enlace, el segundo enlace y el receptor?. ¿A qué se deben tales desfases?

3. En un ciclo de escritura, ¿cómo se determina la dirección de la memoria de voz donde se debe escribir?. ¿Es la escritura secuencial o controlada?

4. En un ciclo de lectura, ¿cómo se determina la dirección de la memoria de voz de donde se debe leer?. ¿Es la lectura secuencial o controlada?

5. En esta demostración, ¿cuál es el retardo que introduce el conmutador en cada canal?. Justificar dicho retardo en relación al contenido de la memoria de control.

6. Tomando todos los casos posibles, ¿cuáles son el máximo y el mínimo retardo que puede introducir el conmutador en un canal?. Poner un ejemplo donde se alcancen ambas situaciones.

3. Sincronismo y señalización en un sistema TDM

Seleccionando la opción B del simulador y la demostración de sincronismo y señaliación, y usando ejecución ciclo a ciclo, slot a slot, o trama a trama según convenga, responder a las siguientes cuestiones:

1. ¿Qué tipo de señalización se utiliza, señalización en banda o señalización por canal común?. ¿Por qué?. ¿En qué canal va ubicada?. ¿Cuál es el canal de sincronismo?

2. ¿Qué función tiene el módulo de sincronismo y señaliación?

3. Con el sistema desincronizado, ¿qué operaciones reliza el módulo de sincronismo y señalización y el receptor al recibir el sincronismo de trama (representado en el simulador por el carácter “!”) y el sincronismo de multitrama (representado en el simulador por el carácter “^”)?. Tras recibir el receptor el sincronismo de multitrama, ¿cuál es el desfase entre fuente, conmutador y receptor?, ¿coincide con el indicado en el apartado 2, punto 2?

4. Para cargar la memoria de control de utilizan dos registros: el registro *Tr* y el registo *Slot 4*. ¿Qué contiene cada uno de ellos?. ¿Cuál de ellos determina la dirección de la memoria de control, el contenido de la memoria de control, la dirección de la fuente y la dirección del receptor?. ¿Por qué?

5. ¿Cuántas tramas hay en una multitrama?

6. Describir la estructura de la trama y multitrama del sistema MIC 30+2 explicado y compararla con la que se presenta en la práctica. ¿Qué analogías y diferencias existen entre ambas?