UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

Proyecto Fin de Carrera

Escuela superior técnica de Ingenieros de Telecomunicaciónes

Servidor para uniformizar el manejo de distintos analizadores de redes

Manual del Programador

Autor: Rubén Del Olmo Esteban

Director: Vicent Miquel Rodrigo Peñarrocha

Fecha: Septiembre 2013

Índice de contenido

1. Introducción

En este manual de usuario se va a explicar los aspectos relacionados con el programación del sistema. Explicando variables, clases y métodos que se han utilizado. Así como detalles de funcionamiento como los ciclos seguidos por los programas principales.

Se parte del hecho que se ha leído la memoria del proyecto. También el usuario de este manual ha de tener conocimientos sobre el funcionamiento de los analizadores A8714 y sus comandos SCPI. Puesto que no se va explicar ninguno de los 2 y los comandos SCPI utilizados están dentro del manual de los analizadores.

También se recomiendan conocimientos de C++, PHP y javascript para poder acompañar las explicaciones con el código escrito del programa.

El proyecto ejecuta en 3 sitios distintos:

1º. El servidor con las librerías Visa.

2º. El servidor Web.

3º. El navegador del cliente.

El código del primero se encuentra en C++ y utilizar las librerias Visa, Pthread y librerias del sistema. Las 2 primeras se deben de añadir al compilador correctamente para poder compilar el proyecto.

El código del segundo y tercero se encuentran ambos en el servidor Web. En PHP y Javascript respectivamente.

Se va a proceder a explicar paso a paso el funcionamiento de cada parte de ejecución del proyecto. Se va a suponer que el sistema está configurado de la manera adecuada, y en algunos casos, como las IP's de los servidores, se explicará donde se pueden cambiar.

2. Servidor Visa

El código del Servidor Visa se explicará en el orden en el que está escrito. Para así poder facilitar su comprensión.

2.I Cabeceras

Las cabeceras incluidas en el proyecto son:

#include <visa.h>	Librerías Visa
#include <stdio.h>	Cabecera de entrada/salida en C.
#include <winsock.h>	Librería para envío de sockect TCP. Se utiliza para manejar las comunicaciones TCP/IP y crear los sockect que se comunicarán con el Servidor Visa.
#include <string.h>	Librería para el manejo de cadenas en C.
#include <math.h>	Librería de funciones matemáticas adicionales.
#include <pthread.h>	Librería utilizada para crear hilos de ejecución. Ver manual sobre su funcionamiento.
#if defined (_WIN32) #include <windows.h> /* for Sleep() */ #define YIELD Sleep(10000) #elif defined (_WINDOWS) #include <io.h> /* for _wyield */ #define YIELD_wyield() #else #include <unistd.h> #define YIELD sleep (1) #endif	Librerías temporales de windows utilizadas para dejar algún hilo de ejecución en estado de espera en el caso de que sea necesario.

En la cabecera también se incluye la siguiente línea:

"#define _CRT_SECURE_NO_DEPRECATE".

Esta línea se ha incluido porque se utilizan funciones de C obsoletas como son strcopy y sprintf. Aunque estas funciones no son seguras, porque pueden provocar errores en memoria al no estar limitado el número de caracteres que escriben. Se ha decidido utilizarlas porque las librerias Visa trabajan directamente con cadenas de este tipo. Una futura ampliación sería la optimización de estas cadenas y añadir mayor seguridad a su gestión.

2.II Variables globales

El proyecto utiliza varias variables globales para su funcionamiento:

```
pthread_mutex_t *semaforosana; //Creamos un vector de semaforos.

char ** volatile nombres; //Le estoy diciendo que lo volatile son los caracteres finales.

volatile unsigned int ninstrumentos;

pthread_mutex_t semaforo;

ViSession defaultRM;

Analizador **Vectorana;
```

- **pthread_mutex_t *semaforosana:**

Es el vector de semáforos que rige el acceso a las variables de analizador. Su tamaño es igual al número de analizadores que ha detectado el proyecto. Cada semáforo está relacionado con la variable analizador que ocupa su misma posición dentro del vector de analizadores. Se utiliza cuando se quiere acceder a un analizador, ya bien sea para leer o para escribir.

- **char ** volatile nombres:**

En este vector se guardan los IDN de todos los analizadores detectados. Se utiliza para reconectarse con el analizador correspondiente en caso de que éste se caiga. Está organizado de la misma manera que el vector de semaforos, correspondiéndose cada posición con la del vector de analizadores. Es volatile ya que no queremos que el compilador almacene su valor en caché y la guarde directamente en memoria. Al especificar volatile al final se indica que lo volatile son los valores de memoria, donde se guardan los caracteres de la cadena. En caso contrario, lo que se guardaría en memoria serían los punteros a vector, provocando que no se guardaran en memoria los datos tipo char.

- **volatile unsigned int ninstrumentos:**

En esta variable se guardan el número de analizadores detectados.

- **ViSession defaultRM:**

Es la sesión Visa por defecto. Se utiliza para encontrar los instrumentos conectados al servidor y abrir sesiones con ellos. Para más información sobre estás sesiones consultar con el manual de las librerías Visa.

- **Analizador **Vectorana:**

Es el vector que almacena todas las variables analizador detectadas. Su tamaño es igual al número de instrumentos que se han detectado previamente en el “main” del Servidor. La posición que ocupa cada analizador está relacionada con el vector de semáforos y la propiedad “puesto” de cada variable tipo analizador.

2.III Clases o estructuras definidas

La siguiente parte del código define las clases utilizadas y una estructura creada para un analizador en concreto.

2.IV Estructura Parametros

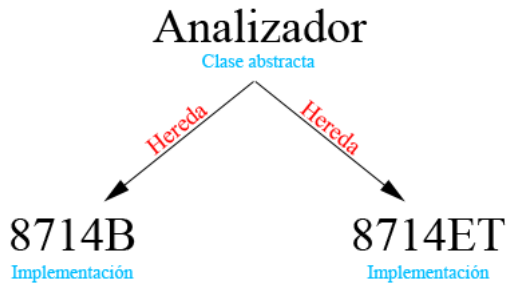
Esta estructura es:

```
struct Parametros{

        volatile double fini,ffinal,IFBandwidth;

        volatile unsigned int npuntos,Z0;

};
```

Se podría haber escogido crear una clase para almacenar estos datos; pero al no necesitar ningún método esta clase, se optó por una estructura.

Esta estructura se utiliza para almacenar los parámetros de calibrado de un analizador. Para que en casos como el analizador 8714B, que no se dispone de manera remota, la forma de averiguar su calibración, se pueda calcular su estado comparándolos con los valores actuales del analizador.

En cuanto a las clases, existen 3 clases creadas en el proyecto. La clase analizador y 2 clases hijas por cada analizador soportado: 8714B y 8714ET.



La clase analizador es una clase abstracta, utilizada para definir la idea que se tiene de el funcionamiento y variables de un analizador. El resto de analizadores heredan de ella e implementan los métodos definidos en esta clase. Excepto en algunos casos que dichos métodos pueden ser utilizados en cualquier analizador ya que son órdenes genéricas. Si existiera algún problema se podría modificar estos métodos genéricos para adaptarlos a nuevos analizadores en caso de que hubiera incompatibilidades.

El resto de métodos recogidos en la clase analizador deben de ser implementados y realizados para el buen funcionamiento del proyecto. Se corresponden cada uno con un tipo de menú incluido en la página Web. A continuación se explicará para que sirve cada uno, y más tarde, en la implementación de cada analizador se explicará brevemente los posibles pormenores de cada uno. No se incluirá ninguna explicación sobre los comandos utilizados, ya que estos se encuentran explicados en el

manual del analizador.

2.V Analizador

```
class Analizador{

public:

    volatile bool Averaging, Automatico, Calibrando, Disparounico, EstadoTraza, PortExtensions, RF, SweepMode;

    volatile char Haytraza;

    char *Direccion, * volatile Estadocalibra, * volatile Estadomenu;

    volatile unsigned char Datamath, Formato, Memoria, Parametro;

    volatile double Fancho, Fcentral, Ffinal, Fini, IFBandwidth;

    volatile float Port1Extension, Port2Extension, Puntos[3202], SweepTime, VelocityFactor;

    volatile unsigned int Avgfactor, CalKit, Npuntos, Puesto, SweepType, Z0;

    volatile unsigned short int Tipo;

    volatile ViSession vi;

    pthread_t Enviacomandosid,Enviadatosid;

    virtual void calibracion(char *datos,char *cadena) = 0;

    virtual void cambiarformato(char *datos,char *cadena) = 0;

    virtual void cambiarparametros(char *datos) = 0;

    virtual void deshabilitarinterrupciones () = 0;

    virtual void display(char *datos,char *cadena) = 0;

    virtual void frecuencias(char *datos,char valor[10],char *cadena) = 0;

    virtual void habilitarinterrupciones() = 0;

    virtual void habilitarnuevatraza() = 0;

    virtual void obtenerdatos(char *datos,char *cadena) = 0;

    virtual void obtenerformato() = 0;
```

```
        virtual void obtenermath() = 0;

        virtual void obtenertraza() = 0;

        virtual void pendiente(char *datos,char *cadena) = 0;

        ViStatus preguntar(char* mensaje,char* pregunta,int maximo)

        virtual void preset() = 0;

        virtual void promediado(char *datos,char *cadena) = 0;

        virtual ViStatus reconectar() = 0;

}
```

2.V.1 Propiedades

Todas las variables son volatile para garantizar su acceso desde distintos hilos de ejecución del programa. En caso contrario se explicará porque se ha decidido cambiar esta propiedad. Existen valores como IF Bandwidth o Z0. Que en analizadores como los de las serie 8714 tienen valores predefinidos. A pesar de eso se guardan como valores con mayor precisión ya que está definidos de manera general para todos los analizadores. Cuando se interactua con estos valores se tiene presente este hecho y se ofrecen las opciones limitadas a ese analizador.

Las variables booleanes son las siguientes y indican si están activos que:

- ***Averaging:***

el analizador está realizando un promediado de las trazas.

- ***Automatico:***

nos encontramos en modo continuo. Su consecuencia es que en la función obtener traza se pide una traza nueva al analizador (Modo continuo). Si está desactivado este disparo se tendrá que producir internamente.

- ***Calibrando:***

el analizador se encuentra calibrando. De está manera se protege al analizador, contra el envío de comandos que no tengan que ver con la calibración. Comprobando esta variable antes de cada comando realizado.

- ***Disparounico:***

nos encontramos en los modos “hold” o “simple”. Se utiliza para no producir disparos internamente cada vez que se produce un cambio.

- ***EstadoTraza:***

los datos de la traza actual que tenemos son válidos. Está variable se utiliza para informar cuando se envía al Servidor Web, que la traza que poseemos actualmente no es válida, debido algún cambio en los parámetros de configuración y que ya se ha ordenado realizar una nueva al analizador.

- ***PortExtensions:***

el parámetro PortExtensions del analizador se encuentra activo.

- ***RF:***

la señal de radiofrecuencia del analizador se encuentra activa.

- ***SweepMode:***

el tiempo de barrido de la traza es calculado automáticamente por el analizador. Esta variable no se corresponde con el valor real del analizador, ya que si el tiempo de barrido es menor que 0,7 segundos, el tiempo de barrido lo seleccionaremos en modo manual y fijaremos el tiempo a 0,7 segundos después de tomar la primera traza. Aunque de cara al usuario este valor esté en automático.

Las variables de tipo carácter se utilizan para 2 usos:

- Almacenar cadenas de caracteres.

- Almacenar valores que no requieren un rango mayor de 256. Facilitando así su conversión y su envío al Servidor Web al ser un carácter.

El nombre de las variables y almacenan:

- ***Haytraza:***

el identificador de la traza actual. Se compara con el valor actual del Servidor Web y si posee un identificador de traza distinto se envía la traza actual. El valor “\n” está exento de los valores posibles de haytraza, porque generaba problemas de reconocimiento en el Servidor Web. Asi que se utiliza para control de datos, cuando accedemos a una analizador por primera vez se envía este valor para indicar que obligatoriamente hay que enviar la traza.

- ***Direccion:***

la dirección Visa asociada al analizador. Cuando se quiere reconectar con el analizador el Servidor, por que se ha caído, se utiliza este valor, sin tener que volver a identificar, ni buscar el analizador

entre los instrumentos conectados. Este valor no es volatile, porque es de solo lectura. Solo se escribe cuando se inicializa el analizador y luego ya no se cambia.

- ***Estadocalibra:***

el estado de la calibración del analizador. Pudiendo ser: “ “, “C”, ”C?”, “Cx”. Esta variable se actualiza cada vez que se cambia un parámetro que puede afectar a la calibración. Como son: el número de puntos, las frecuencias en las que se toma la traza, el ancho de banda del filtro de resolución, la impedancia del sistema, parámetro de medida y un cambio en la propia calibración.

- ***Estadomenu:***

el menú en el que se encuentra calibrando en este momento el analizador. Es una medida de protección adicional, para evitar comandos de calibración que no son adecuados a la calibración actual. A pesar de ser volatile, a veces como acaba de funcionar bien del todo, llegando a no guardarse el valor en memoria cuando es necesario. Así que el Servidor Web está protegido frente a posibles valores inválidos en este campo.

Esta variable su valor siempre empieza por 6, el valor de menú de calibración y el resto de la cadena indica el valor del parámetro “opcion” de la petición Web tipo GET al servidor.

- ***Datamath:***

el estado de la operación matemática que el analizador realiza sobre la traza. Estás operaciones las realiza el analizador, el Servidor Visa no cambia los datos de traza para nada. Sus valores tienen un rango del 1 al 5. Aunque en los analizadores 8714 solo se pueden utilizar los 2 primeros valores.

  - 1: OFF.
  - 2: Data/Mem.
  - 3: Data*Mem.
  - 4: Data−Mem.
  - 5: Data+Mem.

- ***Formato:***

el formato en que está realizada la traza. Se utilizan los numeros 1al 9 y las letra “a” y “b”. Se corresponden con los siguientes valores:

  - 1: Magnitud en formato logarítmico.
  - 2: Fase.
  - 3: Retardo de grupo.

- 4: Carta de Smith.
- 5: Formato polar.
- 6: Magnitud en formato linear.
- 7: Relación de onda estacionaria.
- 8: Parte real de la medida.
- 9: Parte imaginaria de la medida.
- a: Fase expandida. (No presente en los analizador 8714B y 8714ET)
- b: Fase positiva.

- ***Memoria:***

la traza que se representa por pantalla. Se utilizan valores del 1 al 3 para referirse a los distintas fuentes de donde puede obtener la traza el analizador.

- 1 Data: Obtiene los datos directamente de la traza tomada, o de una operación matemática entre la memoria y la traza tomada si Data Math está activo.
- 2 Mem: Obtiene los datos almacenados en la memoria del analizador.
- 3 Data & Mem (No implementado): Saca por pantalla ambas trazas.

- ***Parametro:***

el parámetro de medida de donde obtenemos la traza. Su rango de valores se implementa del 1 al 4 significando cada uno:

- 1: S11.
- 2: S21.
- 3: S12.
- 4: S22

Según la precisión que se requiera, las variables enteras podrán ser tipo double o tipo float.

Las primeras se reservan para frecuencias de medida. Las frecuencias se guardan con su valor real, tomando como unidad de medida el segundo. Sus valores se utilizan para guardar:

- ***Fancho:***

el valor del ancho de banda de la medida.

- ***Fcentral:***

el valor de la frecuencia central de la medida.

- ***Ffinal:***

la frecuencia final de medida.

- ***Fini:***

la frecuencia de inicio de medida.

- ***IFBandwidth:***

la frecuencia del filtro de resolución con el que se toma la traza.

Las variables tipo float tienen como misión preservar el valor de:

- ***Port1Extension:***

la extensión en segundos del primer puerto. Ya que los analizadores 8714 solo nos dejan introducir el valor en estás unidades.

- **Port2Extension:**

la extensión en segundos del segundo puerto.

- ***Puntos:***

de la traza actual. El tamaño máximo de este vector está ajustado al número máximo de puntos que podemos tener, que es en formato polar/carta de Smith de 1601 puntos. Se debe a que en este modo se envía el doble de puntos, la parte real y parte imaginaria.

- ***SweepTime:***

el tiempo de barrido de traza. Al igual que SweepMode este valor que se almacena en ocasiones no se corresponde con el valor interno del analizador. Debido a que si el tiempo es menor que 0.7 segundos se fija a este valor. Este valor almacena el valor que tendría en el caso de que no limitáramos este tiempo.

- ***VelocityFactor:***

con el parámetro velocity factor del analizador.

Las variables tipo int se corresponden con los siguientes parámetros del analizador y del proyecto:

- ***Avgfactor:***

el número de trazas que toma el analizador para realizar el promediado.

- ***CalKit:***

el kit de usado para la calibración del analizador. Como cada analizador tiene sus propios kit de calibración registrados, se utiliza este número para referirse a cada uno de ellos en el orden especificado en el manual. En la sección dedica al menú de calibración de cada analizador y concretamente al kit de calibración se explicará esta correspondencia.

- ***Npuntos:***

el número de puntos sobre el que se ha tomado la traza actual activa.

- ***Puesto:***

la posición que ocupa el analizador, en el vector interno de analizadores detectados por el proyecto.

- ***SweepType:***

el valor sweep type del analizador. Al igual que los kit de calibración cada analizador posee sus valores. Y se corresponde internamente cada valor entero con cada tipo de valor de este parámetro.

- ***Z0:***

el valor de la impedancia del sistema. En el caso de los analizadores 8714 está fijado a 2 valores 50 y 75 Ohmios. Pero se ha escogido una variable de tipo entero porque en el analizador más nuevo, este parámetro se puede configurar entre un rango de valores.

- ***Tipo:***

el tipo de analizador que se está utilizando. Se aplica para que cuando estamos recibiendo una variables a través del vector de analizadores, conocer concretamente el modelo del analizador. Los Valores actuales son los siguientes, aunque se pueden ampliar en un futuro:

    - 1: HP 8714B.
    - 2: HP 8714ET.
    - 3: Agilent E5062A (No implementado)
    - 4: Analizador por defecto (No implementado).

  Esta variable se ha acortado el rango, ya que no se ve necesario que sea un entero normal.

Para guardar la sesión con la que se envían los comandos necesarios al analizador se utiliza la variable de tipo ViSession:

- ***vi:***

está variable se utiliza para comunicarse con el analizador. Es de tipo "public" para que se pueda utilizar desde el exterior, sin utilizar ninguna de los métodos incluidos en la clase. El tipo de sesión abierta será distinta según el analizador.

Las últimas propiedades de la variable analizador son los ID de los hilos de programación asociados a dicho analizador. Esta variable no se ha utilizado más allá de su inicialización en el constructor de la clase. Se incluyó entre las variables de analizador, para poder controlar los hilos de ejecución en caso de que hiciera falta. Como por ejemplo, por un error de conexión del analizador.

2.V.2 Métodos:

todos los métodos, excepto disparo y preguntar, son abstractos. En está sección se definirán la idea que se tiene que haga cada uno de ellos, así como su correspondencia con los parámetros que se le envían a cada método. Basándonos en la comunicación existente entre el Servidor Web y enviadatos. Más adelante en cada analizador se especificará como se ha implementado concretamente cada una de las funciones.

Los métodos relacionados con el menú del analizador, deben incluir todas las funciones de dicho menú. Si se quieren ampliar funciones que faltan a algunos menús, por ejemplo el menú power dentro de Sweep Setup, se tendrá que incluir dentro del método pendiente. También en dichos métodos el primer parámetro que se recibe es una variable de tipo cadena de nombre datos. En la que se incluye la información enviada por el Servidor Web. Incluyendo: Menú, opción y datos adicionales.

Los métodos se relacionan con los menús del analizador Agilent E5062A de la siguiente manera, también se incluye que métodos habilitan una nueva traza:

Nombre del menú	**Nº**	**Nombre del método**	**Descripción**	**¿ Produce disparo en el modo "con cambios" ?**
Measurement	1	cambiarparametros	Se utiliza para cambiar el parámetro de medida. Ya sea en reflexión (S11) o en transmisión (S12).	Sí.
Format	2	cambiarformato	Cambia el formato en el que está representada la traza.	Sí.
Display	4	display	Cambia las opciones de representación de la traza. En el proyecto se realiza en su mayoría en local, en el servidor web. En este caso se utiliza para almacenar	Sí, excepto en Data→Mem.

			la traza en la memoria del analizador, representar a esta o una combinación con la traza actual mediante una operación matemática.	
Average	5	promediado	Controla el promediado y filtro de resolución del analizador.	Sí.
Calibration	6	calibracion	Es el menú que permite calibrar el analizador y sus opciones.	Sí, incluyendo cada paso en la calibración, excepto si se modifica el Kit de calibración.
Stimilus	7	frecuencias	Se utiliza para cambiar las frecuencias.	Sí.
Sweep Setup	8	pendiente	Permite cambiar las opciones bajo las que mide la traza (número de puntos, tiempo de traza...).	No, sólo si se cambia el número de puntos o el modo de medida.
Trigger	9	disparo	Cambia las opciones de disparo del analizador.	No. (Mirar tabla de disparo)
Preset	b	preset	Inicializa el analizador a un estado conocido. Así como todos los parámetros de configuración.	Sí.
Actualizar traza, guardar datos	0	obtenerdatos	Opciones especiales implementadas para el proyecto no incluidas en el analizador.	Sólo en actualizar traza.

Los métodos en los que se incluye la variable “cadena”, son aquellos en los que se espera la respuesta del analizador. Esta cadena no se cambia, a no ser que ocurra un error, en cuyo caso se incluirá el mensaje de error.

Aquellos que cambian una variable del analizador incluyen 3 partes:

- Envio de comandos para cambiar el parámetro del analizador indicado, enviando la cadena correspondiente según la opción indicada por el usuario.

- En caso de que se prevea un posible error, debido a que el usuario haya introducido un valor no válido o no realizable, se procede a la lectura de errores en el analizador a través del comando "SYST:ERR?\n", en caso de que haya error, se almacena de este error en la variable “cadena”.

- Lectura del parámetro cambiado, así como otro posibles parámetros a los que pueda afectar.

- ***calibracion:***

las opciones que incluye este menu son:

  - Opción 1 Calibrate: Es el menú utilizado para calibrar el analizador. Debe proporcionar la comunicación necesaria para calibrar los parámetros S. Como en cada analizador estas opciones de configuración son distintas, se deberán de adaptar al analizador que tengamos.

    También debe de ocuparse del manejo de las propiedades Calibrando y Estadomenu. Activandolos cuando sea necesario para indicar que nos encontramos en una calibración y en que estado se encuentra la calibración. También las utiliza para comprobar en cada paso, para saber si el comando que se ha enviado corresponde al estado de calibración en el que nos encontramos, en el caso de que la calibración sea secuencial. Concretamente Calibrando para el primer paso y Estadomenu para los demás.

  - Opción 2 Clear All: Borra la calibración actual del analizador.

  - Opción 3 Property: ?????

  - Opción 4 Cal Kit: Permite escoger el Kit con el calibramos el analizador. Cada analizador tienes sus propios valores.

  - Opción 5 Modify Cal Kit (No implementado): Permite configurar los Kit de calibración de usuario.

  - Opción 6 Port Extensions: habilita y especifica la longitud de los puertos de medida.

    - Opción 1 Extensions: Habilita o deshabilita la longitud de los puertos de medida a través de la variable PortExtensions.

    - Opción 2 Extension Port1: Especifica la longitud en segundos del primer puerto. Se almacena en la variable Port1Extension.

    - Opción 3 Extension Port2: Especifica la longitud en segundos del primer puerto. Se almacena en la variable Port2Extension.

  - Opción 7 Velocity Factor: Modifica el parámetro velocity factor.

  - Opción 8 Set Z0: Cambia la impedancia del sistema.

- ***cambiarformato:***

Los valores de este formato son los mismos para todos los analizadores, se incluyen en la descripción de la variable Formato y se corresponden con los valores del campo opción. Como los analizadores contestan con el nombre del formato que se está utilizando, el método obtenerformato

es el encargado de preguntar al analizador y guardar los valores en la variable Formato.

- ***cambiarparametros:***

Los valores de este formato son los mismos para todos los analizadores y se incluyen en la descripción de la variable Parametro.

- **deshabilitarinterrupciones:**

configura el analizador para que no notifique al hilo principal que se ha producido una interrupción.

- **disparo:**

su funcionamiento, se basa en modificar las variables Automatico y Disparounico de manera que coincidan con cada modo de disparo. También se habilita una nueva traza en el caso de que esta se necesaria.

Opción	**Modo de disparo**	**Automatico**	**Disparounico**	**¿ Habilitar nueva traza ?**
1	Hold	No	Sí	No
2	Single	No	Sí	Sí
3	Con cambios	No	No	Sí
4	Continuo	Si	No	Sí

- ***display:***

A causa de que los analizadores, si se pregunta por este parámetro, devuelven una cadena, se utiliza la función obtener obtenermath para guardar el valor correcto en la variable correspondiente. Cada opción implementa:

  - Opción 1 al 3 (No implementadas): Se reservan para Allocate Channels, Number of Traces y Allocate Traces respectivamente, que se utilizan para la representación por pantalla de varias trazas a la vez.

  - Opción 4 Display: se elige la fuente de donde se obtiene la traza en base a los valores de la variable Memoria.

  - Opción 5 Data→Mem: guarda la traza actual en la memoria del analizador.

  - Opción 6 Data Math: escoge si se realiza una operación matemática entre la traza actual y la guardada en memoria. Sigue los valores de Datamath.

- ***frecuencias:***

debe de cambiar también el analizador al modo de frecuencias escogido:

- Frecuencia final y/o frecuencia inicial.
- Frecuencia central y/o ancho de banda.
- Si el ancho de banda igual 0, se escoge el modo de una única frecuencia.

Afecta a las variables: Fancho, Fcentral, Ffinal y Fini.

En la variable valor se envía la frecuencia que queremos enviar. Para poder incluirla con sprintf en la cadena a enviar en al analizador.

- **habilitarinterrupciones:**

prepara el hilo de ejecución para ser interrumpido cuando sea necesario por el analizador. También configura las interrupciones en el analizador. Pero no las activa. La activación de las interrupciones se hace cada vez que se habilita una nueva traza.

- **habilitarnuevatraza:**

ordena al analizador que realice una nueva traza e interrumpa al programa cuando este lista.

- **obtenerdatos:**

las opciones del 1 al 3 se utilizan para preguntar los datos actuales al analizador y reenviarlos al Servidor Web. La opción 4 se encarga de notificar al analizador que el usuario desea una nueva traza. Las opciones son:

- Opción 1: Formato de datos csv.
- Opción 2: Formato .s1p.
- Opción 3: Formato .s2p, 4 parámetros. (No implementado)
- Opción 4: Pedir nueva traza.

- ***obtenerformato:***

pregunta al analizador por el formato en el que se envía la traza actual y lo guarda en la variable Formato con los valores especificados en esta, en base a las respuesta que se le da.

- ***obtenermath:***

pregunta al analizador si se realiza alguna operación matemática sobre la traza. También analiza la respuesta recibida y la compara internamente para guardar el estado en la variable Datamath.

- ***obtenertraza:***

guarda la traza actual en la variable Puntos, aumenta el valor de ID de traza Haytraza y activa las interrupciones para la siguiente traza en el caso de estemos en modo automático. Cuando se realiza esto último se comprueba si el valor del barrido de traza es menor de 0,7, para que en caso de que así sea, subirlo hasta ese límite.

Cuando se interrumpe al servidor, una vez comprobado a que analizador corresponde dicha interrupción, se llama a este método para obtener la traza y activar de nuevo la interrupción en caso de que sea necesario.

- ***pendiente:***

cada opción se corresponde a uno de estos valores:

  - Opción 1 Power (No implementado): menú que controla la potencia con la que se toma la medida a lo largo del eje de frecuencias.

  - Opción 2 Sweep Time: el tiempo de barrido de traza, registrado por SweepTime.

  - Opción 3 Sweep Mode: si el tiempo anterior utiliza de modo automático o lo introduce el usuario. Se guarda en la variable SweepMode.

  - Opción 4 Points: elige el número de puntos que se toma la traza, cuyo valor se recoge en Npuntos.

  - Opción 5 Sweep Type: x.

- ***preguntar:***

realiza una pregunta al analizador y guarda su respuesta en la variable mensaje.

  - En mensaje guardamos las respuesta cuando se realiza la lectura del buffer de las librerías visa, en caso de que haya habido un error se transmitirá dicho error.

  - Con pregunta se especifica la orden que se ha de enviar al analizador.

  - Máximo es el número de intentos de lectura que se han de realizar, antes de que se considere que el analizador está desconectado. Se ha incluido este parámetro porque dependiendo el tipo de orden, se puede esperar, que haya cierto retraso al responder.

En el caso de que el número máximo de intentos se supere. Se realizará una llamada a la función reconectar. Una vez acabada esta función, si el resultado a sido positivo, se volverá a intentar realizar la pregunta. Sino se consigue otra vez respuesta se mostrará por pantalla un mensaje de error indicando que hay algún problema con la pregunta realizada.

Este método se utiliza para comprobar el estado de analizador, antes de enviar cualquier comando. Utilizando para ellos el comando "*OPC?". Así nos aseguramos de que el analizador está conectado y listo para recibir las ordenes y en el caso de que no lo esté, que se realiza una reconexión para

poder solucionar este problema.

- ***preset:***

Esta función es la primera que se utiliza el usuario antes de utilizar el analizador, porque anteriormente ha podido ser utilizado en local, o por otro programa, desincronizándose con el Servidor Visa. A través de está función se asegura que los valores que dispone el servidor son los correctos. El método ha de reactivar y configurar las interrupciones, por si estas, hubieran quedado anuladas.

- ***promediado:***

Cada valor del parámetro opción es:

  - Opción 1Averaging Restart: reinicia el contador de promediado a 1.
  - Opción 2 Avg Factor: relacionado con la propiedad Avgfactor. Guarda el número de trazas de las que se toma la media.
  - Opción 3 Averaging: utiliza la propiedad Averaging. Activa o desactiva el promedidado de la traza.
  - Opción 4 y 5, Smo Aperture y Smoothing (No implementado): Controla las funciones de suavizado. Los analizadores 8714 no lo soportan.
  - Opción 6 IF Bandwidth: es el ancho de banda del filtro de resolución. Este valor se guarda como en la variable IF Bandwidth.

- ***reconectar:***

reinicia la conexión que se tenía previamente con el analizador. Si no se puede conectar devolverá el estado de tipo ViStatus recogido del error que de cuando el programa se intente conectar. Para iniciar la conexión se utiliza el valor guardado previamente en Direccion.

También habilita y deshabilita las interrupciones. Pero no las vuelve a configurar, ya que se supone que se ha perdido la conexión. No que otro programa haya modificado el analizador.

2.V.3 Constructor de la clase

Al ser una clase abstracta no se implementa constructor. También hay que tener en cuenta que cada analizador puede tener valores de inicialización distintos. El constructor de cada clase hija deberá de realizar:

- realizar un preset sobre el analizador, para llevarlo a un estado conocido y así estar sincronizado con el servidor.
- inicializar todas las propiedades de la clase analizador, más las suyas propias con los valores obtenidos del preset anterior. También inicializar propiedades como las que indican la

propiedad “puesto”, el tipo de analizador, la conexión abierta Visa...

- la creación de los hilos enviadatos y enviacomandos. Asociando su ID con sus respectivas variables.

2.VI Características comunes

Los analizadores de la clase A8714 poseen una serie de características comunes. Obteniendo el mismo código de manejo para ambos.

2.VI.1 Obtención de traza

El primer paso es elegir de donde queremos obtener la traza en función del valor de Memoria. También se desactivará el modo de lectura, para que deje de leer cuando se encuentra con un carácter de fin de cadena. Debido a que en el envío de la traza el programa se puede encontrar con este tipo de caracteres.

Para enviar la traza, utilizamos el formato binario de 32 bits, que es el más rápido y no requerimos tanta precisión como para utilizar el de 64. El formato en el que envían los datos los analizadores es:

#<num_digits><num_bytes><data_bytes>

Siguiendo este esquema, primero leeremos los 2 primero bytes. Para conocer el número de dígitos del campo siguiente. Y luego leeremos este campo para conocer la longitud exacta del último campo y leerlos todos.

Todos los datos leídos se almacenaran en la variable puntos, realizando un conversión a tipo float de 4 en 4 bytes.

Por último si se encuentra en modo de disparo continuo se activaran las interrupciones y comprobaremos que el valor de barrido de traza es mayor que 0,7 segundos.

2.VI.2 Interrupciones

La explicación de las interrupciones, y el uso de los registros utilizados en el proyecto se explica en la memoria del proyecto. Aquí se explicará brevemente que comandos se utilizan y se explicará su utilización a nivel de código.

La habilitación de interrupciones se realiza de forma normal en las librerías visa, asignando a la interrupción una rutina la cual se activará cuando se produzca una interrupción. Está rutina ha sido llamada viinterrumpir.

Para activar los registros se utilizan los comandos:

- “SRE 32”: Con esto se activa el bit 5 de las interrupciones generales (Standard Event Summary).

- ESE 1: Con esto se activa el bit 1 del Standard Event Summary habilitando que se interrumpa cuando se han completado todas las instrucciones anteriores al ultimo “OPC” enviado.

Durante el código cada vez que se quiera enviar un comando, se deshabilitarán previamente las interrupciones. Para luego llamar a la función habilitarnuevatraza si se requiere una nueva traza, y por tanto el uso de interrupciones. En estos analizadores, al no tener registro de interrupciones para avisar de una nueva traza. Se utilizará el comando “INIT1” para iniciar una nueva traza y “OPC” para que se interrumpa cuando el comando anterior ha finalizado

2.VII A8714ET

Este analizador tiene la particularidad que se ha implementado un modo de funcionamiento basado en 2 sesiones Visa. Una para las interrupciones y otra para obtener datos. Esto se debe a que este analizador no posee interrupciones mediante el protocolo TCP/IP pero si podemos comunicarnos con él por este medio. Por ello las implementaciones extra en el funcionamiento están orientadas a este doble control. Para que sigue cumpliendo las especificaciones previas de las clase analizador padre, la sesión de comunicación, se deberá de guardar sobre la propiedad “vi”.

También el constructor de la clase, los métodos basados en interrupciones y que modifiquen la variable de sesion VISA deberán tener en cuenta este cambio.

2.VII.1 Calibración

La calibración en este analizador es secuencial. Es decir no se puede escoger el tipo de corrección que queremos aplicar. Sino que se pedirá el de conexión adecuada para calibrar en base al kit de calibración usado.

Por eso en el programa se ha implementado un menú secuencial utilizando la variable Estadomenu para indicar en que parte de la calibración nos encontramos.

Para abortar una calibración, a parte del envío del comando correspondiente al analizador, se desactiva Calibrando. Estadomenu no se modifica, ya que la próxima activación de Calibrando inicializará este valor y no se puede acceder a él Calibrando sin estar activado.

2.VII.1.i Menú de calibración

Posee 2 menús de calibrado, uno por cada parámetro de medida (S11,S21). Antes de entrar a la opción de cada menú comprobamos que el parámetro es el adecuado, en caso contrario devolvemos un error al servidor Web.

Parámetro S11 (1- Port Cal):

0. Open activo (menú 6 y opción 15): Estadomenu aún no tiene un valor asignado, con lo cual no se comprueba que se encuentre en el menú adecuado, al estar al inicio de la calibración. Como medida adicional de seguridad el programa si que utiliza el valor de Calibrando para cerciorarse de que no se encuentra en una calibración. Ya que puede ocurrir que se envíe este comando, por múltiples pulsaciones en la página web por ejemplo, cuando ya nos encontremos en una calibración activa. Se inicializa Estadomenu al menú en el que nos

encontramos, “615”.

1. Short activo (menú 6 y opción 151): El valor de Estadomenu es “615”, la comprobación se realiza en la 4ª posición de este vector ya que en este caso será “\0”. Si todo es correcto se actualizará Estadomenu a 6151.

2. Load activo (menú 6 y opción 152): Estadomenu tiene como valor “6151”. Con lo cual la 4ª posición de este vector valdrá “1”. Sino se producen errores se actualizará Estadomenu al siguiente valor.

3. Fin de la calibración, el usuario a pulsado Load. (menú 6 y opción 153): Se comprueba que la posición 4ª su valor es “2” y se inicializa el fin de la calibración. Como en este analizador el cálculo de la calibración se realiza de manera casi inmediata, no se introduce ningún tiempo de espera para acabar de calibrar. Se indica el fin de la calibración desactivando Calibrando y no se modifica Estadomenu ya que en la próxima vez que active Calibrando se inicializará su valor. Y no se puede acceder a este último menú si Calibrando no está activo.

Parámetro S21 (Response & Isolation):

0. Open activo (menú 6 y opción 14): Estadomenu aún no tiene un valor asignado, con lo cual no se comprueba que se encuentre en el menú adecuado, al estar al inicio de la calibración. Como medida adicional de seguridad el programa si que utiliza el valor de Calibrando para cerciorarse de que no se encuentra en una calibración. Ya que puede ocurrir que se envíe este comando por múltiples pulsaciones en la página web por ejemplo, cuando ya nos encontremos en una calibración activa. Se inicializa Estadomenu al menú en el que nos encontramos, “614”.

1. Short activo (menú 6 y opción 141): El valor de Estadomenu es “614”, la comprobación se realiza en la 4ª posición de este vector ya que en este caso será “\0”. Si todo es correcto se actualizará Estadomenu a 6141.

2. Load activo (menú 6 y opción 142): Estadomenu tiene como valor “6141”. Con lo cual la 4ª posición de este vector valdrá “1”. Sino se producen errores se actualizará Estadomenu al siguiente valor.

3. Through (menú 6 y opción 143): Estadomenu tiene como valor “6142”. Con lo cual la 4ª posición de este vector valdrá “2”. Sino se producen errores se actualizará Estadomenu al siguiente valor.

4. Fin de la calibración, el usuario a pulsado Load. (menú 6 y opción 144): Se comprueba que la posición 4ª su valor es “3” y se inicializa el fin de la calibración. Como en este analizador el cálculo de la calibración se realiza de manera casi inmediata, no se introduce ningún tiempo de espera para acabar de calibrar. Se indica el fin de la calibración desactivando Calibrando y no se modifica Estadomenu ya que en la próxima vez que active Calibrando se inicializará su valor. Y no se puede acceder a este último menú si Calibrando no está activo.

2.VII.1.ii Kit's de calibración

La correspondencia entre los kits de calibración y sus valores para este analizador son:

1. Type-N(f).
2. Type-N(m).
3. 3.5 mm.
4. Type-F.
5. APC-7.
6. 7-16.
7. User Cal Kit A.
8. User Cal Kit B.
9. User Cal Kit C.
10. User Cal Kit D.
11. User Cal Kit E.
12. User Cal Kit F.
13. User Cal Kit G.
14. User Cal Kit H.
15. User Cal Kit I.
16. User Cal Kit J.

2.VII.2 IF Bandwidth discreto

En el analizador 8714ET no se puede seleccionar un filtro de resolución a medida del usuario, sino que se debe escoger una serie de valores de una batería de filtros. A cada valor se le ha asignado un número, aun en el programa, el filtro se almacene como una variable de tipo float para tener compatibilidad con analizadores más nuevos. Los valores de las opciones que se reciben son los siguientes:

1. Wide (6500 Hz).
2. Med Wide (4000 Hz).
3. Medium (3700 Hz)
4. Med Narrow (1200 Hz).
5. Narrow (250 Hz).

6. Fine (15 Hz).

2.VII.3 Propiedades adicionales

- ***IP:***

guarda la dirección IP del analizador. Esta variable no se ha hecho de tipo volátil porque se podría considerar de solo lectura. Su valor solo se modifica al inicio y cuando se realiza una reconexión con el analizador. A través de esta variable se inicia la conexión de comunicación con el analizador y se guarda por si fuera necesario usarla.

- ***viinterrumpir:***

es un enlace a la sesión Visa abierta por GPIB para manejar las interrupciones del analizador. Cuando se llaman a métodos que trabajan con las interrupciones, se utiliza esta sesión.

2.VIII A8714B

Este analizador no se podía averiguar el estado de la calibración de forma remota. Imposibilitando al servidor VISA conocerlo y notificarlo al servidor web. La solución fue crear una estructura, que cada vez que se calibrará el analizador, guardar los parámetros de configuración. Y comparar esta estructura cuando fuera necesario para saber si está calibrado.

Sin en un futuro se implementa la calibración mediante ficheros, será necesario que en este analizador cuando se cargue el fichero, se analicen y guarden los valores de configuración.

2.VIII.1 Calibración

La calibración en este analizador es secuencial. Es decir no se puede escoger el tipo de corrección que queremos aplicar. Sino que se pedirá el de conexión adecuada para calibrar en base al kit de calibración usado.

Por eso en el programa se ha implementado un menú secuencial utilizando la variable Estadomenu para indicar en que parte de la calibración nos encontramos.

Para abortar una calibración, a parte del envío del comando correspondiente al analizador, se desactiva Calibrando. Estadomenu no se modifica, ya que la proxima activación de Calibrando inicilizará este valor y no se puede acceder a él Calibrando sin estar activado.

2.VIII.1.i Menú de calibración

Posee 2 menús de calibrado, uno por cada parámetro de medida (S11,S21). Antes de entrar a la opción de cada menú comprobamos que el parámetro es el adecuado, en caso contrario devolvemos un error al servidor Web.

Parámetro S11 (1- Port Cal):

5. Open activo (menú 6 y opción 15): Estadomenu aún no tiene un valor asignado, con lo cual

no se comprueba que se encuentre en el menú adecuado, al estar al inicio de la calibración. Como medida adicional de seguridad el programa si que utiliza el valor de Calibrando para cerciorarse de que no se encuentra en una calibración. Ya que puede ocurrir que se envíe este comando (por múltiples pulsaciones en la página web por ejemplo) cuando ya nos encontremos en una calibración activa. Se inicializa Estadomenu al menú en el que nos encontramos, "615".

6. Short activo (menú 6 y opción 151): el valor de Estadomenu es "615", la comprobación se realiza en la 4ª posición de este vector ya que en este caso será "\0". Si todo es correcto se actualizará Estadomenu a 6151.

7. Load activo (menú 6 y opción 152): Estadomenu tiene como valor "6151". Con lo cual la 4ª posición de este vector valdrá "1". Sino se producen errores se actualizará Estadomenu al siguiente valor.

8. Fin de la calibración, el usuario a pulsado Load (menú 6 y opción 153): se comprueba que la posición 4ª su valor es "2" y se inicializa el fin de la calibración. Como en este analizador el cálculo de la calibración se realiza de manera casi inmediata, no se introduce ningún tiempo de espera para acabar de calibrar. Se indica el fin de la calibración desactivando Calibrando y no se modifica Estadomenu ya que en la próxima vez que active Calibrando se inicializará su valor. Y no se puede acceder a este último menú si Calibrando no está activo.

Parámetro S21 (Enhanced Response):

0. Loads activo (menú 6 y opción 14): estadomenu aún no tiene un valor asignado, con lo cual no se comprueba que se encuentre en el menú adecuado, al estar al inicio de la calibración. Como medida adicional de seguridad el programa si que utiliza el valor de Calibrando para cerciorarse de que no se encuentra en una calibración. Ya que puede ocurrir que se envíe este comando, por múltiples pulsaciones en la página web por ejemplo, cuando ya nos encontremos en una calibración activa. Se inicializa Estadomenu al menú en el que nos encontramos, "614".

1. Through activo (menú 6 y opción 141): el valor de Estadomenu es "614", la comprobación se realiza en la 4ª posición de este vector ya que en este caso será "\0". Si todo es correcto se actualizará Estadomenu a 6141.

2. Fin de la calibración, el usuario a pulsado Load (menú 6 y opción 142): se comprueba que la posición 4ª su valor es "1" y se inicializa el fin de la calibración. Como en este analizador el cálculo de la calibración se realiza de manera casi inmediata, no se introduce ningún tiempo de espera para acabar de calibrar. Se indica el fin de la calibración desactivando Calibrando y no se modifica Estadomenu, ya que en la próxima vez que active Calibrando se inicializará su valor. Y no se puede acceder a este último menú si Calibrando no está activo.

2.VIII.1.ii Kit's de calibración

La correspondencia entre los kits de calibración y sus valores para este analizador son:

1. Type-N(f).

2. Type-N(m).
3. User Defined.
4. 3.5 mm.
5. Type-F.

2.VIII.2 IF Bandwidth discreto

En el analizador 8714ET no se puede seleccionar un filtro de resolución a medida del usuario, sino que se debe escoger una serie de valores de una batería de filtros. A cada valor se le ha asignado un número, aun en el programa, el filtro se almacena como una variable de tipo float para tener compatibilidad con analizadores más nuevos. Los valores de las opciones que se reciben son los siguientes:

1. Wide (6500 Hz).
2. Medium (3700 Hz)
3. Narrow (250 Hz).
4. Fine (15 Hz).

2.VIII.3 Propiedades y métodos adicionales

- ***Calibrado:***

Almacena los parámetros sobre los que se ha calibrado el analizador. Para así poder tener un referente cuando queramos saber si el analizador se encuentra calibrado.

- ***estadocalibracion:***

Devuelve el estado en el que se encuentra el analizador con respecto a la calibración siendo:

  - "C\n": Si todos los parámetros coinciden. Indicando que el analizador está correctamente calibrado.
  - "C?\n": Si las frecuencias en las que nos encontramos estás dentro del rango de frecuencias en el que se ha calibrado y alguno de los parámetros de calibración es distinto del original. Indicando que no está calibrado pero es posible interpolar la calibración actual con cierto margen de error.
  - " \n": En el caso de que las frecuencias no se encuentren dentro del margen, con lo cual al no disponer de datos en esas frecuencias, el analizador no puede interpolar y no aplica ningún tipo de corrección a la medida.

La comparación se realiza:

- Primero en el rango de frecuencias, si nos encontramos fuera del rango de frecuencias el analizador está descalibrado.

- Comparamos los parámetros: Fini, Ffinal, IFBandwidth, Npuntos y Z0 si son iguales que cuando se realizó la calibración. Si no lo son se notificará con "C?" y si lo son con "C".

2.IX main

El método main realiza el siguiente esquema:

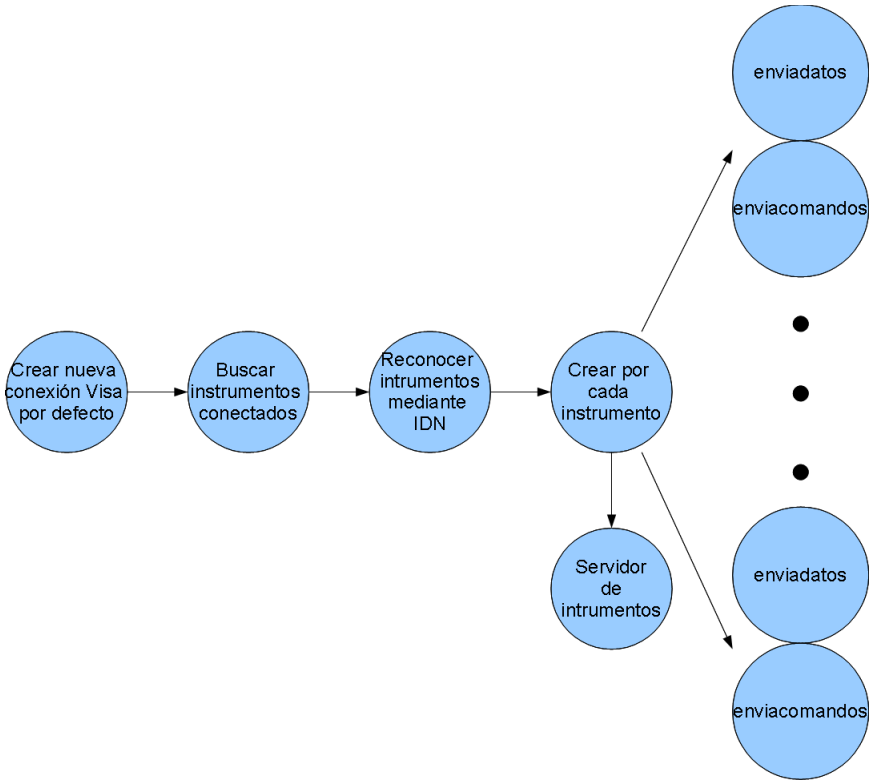


Ilustración 1: Esquema de ejecución

Podemos distinguir 2 partes separadas en esta función: la inicialización y el bucle principal encargado de controlar las opciones del servidor.

2.IX.1 Inicialización del servidor

Lo primero que se comprueba es que las librerias de windows para realizar conexiones de sockect estén instaladas y soporten las funciones que utilizamos. En caso contrario se mostrará un error por pantalla y se saldrá del programa.

El segundo paso es abrir una conexión VISA por defecto. Esta se almacenará en la variable defaultRM. Se utilizará a lo largo del servidor para iniciar y cerrar conexiones con los analizadores.

La siguiente tarea que realiza el servidor, es escanear todos los posibles instrumentos que hay conectados a él. Una vez estén todos registrados por el servidor, inicia una conexión con cada uno

de ellos, para saber si son analizadores soportados por el servidor. Con el fin de reconocerlos se utiliza el comando "*IDN?", en cuya cadena de respuesta, se incluye el tipo de modelo al que estamos accediendo.

Como ya se tiene el número de analizadores detectados en la variable ninstrumentos. Se pueden empezar a inicializar las variables cuyo tamaño está relacionado con este valor.

La primera variable en crearse es un vector de semáforos, que regulará el acceso a las variables compartidas de cada analizador. Se almacena bajo la variable semaforosana. El semáforo utilizado es con bloqueo. Es decir cuando un hilo quiera acceder a la variable analizador, activará el semáforo correspondiente a ese analizador. Si el semáforo ya ha sido activado y abierto por otro hilo, este se quedará esperando a que se cierre. Cuando se acaba de utilizar la variable analizador se cierra el semáforo para que otro hilo pueda acceder.

Igualmente se reservará espacio en memoria para el vector de cadenas nombres y el vector de analizadores Vectorana. Ambos vectores se rellenaran cuando se inicialicen cada uno de los analizadores.

Se le asignará a cada analizador una posición, denominada puesto a nivel de código, que indica la posición de cada analizador en distintos vectores, siendo estos:

- Vector de control de todos analizadores.
- Vector de semáforos.
- Vector de almacenamiento de nombres.

La posición sirve también para asignar el número de puertos a los que se conectará el Servidor Web, cuando se comunique con el Servidor VISA.

El siguiente paso es inicializar los vectores creados anteriormente. En función del tipo de analizador recogido en la variable Tipo se creará un objeto de la clase adecuada al tipo. Igualmente se recogerá el nombre del analizador para sacarlo por pantalla y se inicializará el semáforo correspondiente.

Después de liberar toda la memoria utilizada para encontrar los analizadores, se creará el hilo que atiende por primera vez al usuario, se ha indicado al servidor web el numero de analizadores y sus puertos: menuinstrumentos.

2.IX.2 Bucle principal

En esté bucle se incluirán las funciones avanzadas del servidor. Como pueden ser: volver a escanear y todos los analizadores, reiniciar las conexiones, eliminar analizadores de la lista, cerrar el programa...

De momento solo incluye la opción 0 que se utiliza para salir del programa.

2.X Funciones

2.X.1 convertirnumero

Almacena en cadena el valor incluido en la variable numero. Añadiendo las unidades en hercios correspondientes al valor indicado.

2.X.2 convertircadena

Analiza los valores multiplicativos (kilo, mega y giga) de las unidades de la variable cadena, para devolverlo en la función junto con el número indicado.

2.X.3 recortarcadena

Devuelve la cadena de origen en la cadena destino recortando por los valores de inicio y fin indicados.

2.X.4 err_handler

Imprime por pantalla el error de tipo ViStatus asociado a la sesión indicada.

2.X.5 preguntarsesion

Realiza la misma función que preguntar. La diferencia es que se especifica la sesión visa sobre la que se realiza la pregunta. Está función se utiliza cuando no tenemos ninguna variable de la clase analizador instanciada y queremos preguntar con seguridad a un instrumento.

2.XI Rutinas de atención a interrupciones

2.XI.1 mySrqHdlr

Es la rutina que atiende a las interrupciones producidas por analizadores cuando tenemos una nueva traza. Si no encontramos por GPIB se interrumpe igualmente en todos los hilos, aunque el analizador no haya activado la linea de interrupciones, ya que esta es compartida.

Como consecuencia el primer paso es asegurarnos que el analizador ha sido el que activado la linea. A continuación se borra este estado y se obtiene la traza.

2.XI.2 myExceptionHandler

Se utiliza para depurar posibles fallos en el código. Es una rutina que se activa junto con las librerias VISA, para que cada vez que haya un error en estas, se detecte y se imprima por pantalla.

2.XII Hilos de ejecución

2.XII.1 enviadatos

Su función es enviar los datos del analizador que le pida el Servidor Web. El Servidor Web le envía el menú, la opción en los que se encuentra el usuario y la identificación de traza que posee el servidor.

En el primer paso se inicia el servidor para responder a las peticiones del servidor web. Utilizando el valor 55551 + 1 * puesto. Si el inicio de conexión resulta fallido o no se puede reservar el puerto, se imprimirá un error por pantalla y se cerrará el hilo.

A continuación se entrará en un bucle en el que se atenderá a las peticiones de los usuarios que vayan llegando. Según el menú en el que nos encontremos se enviará información actual de las variables que estén asociadas a ese menú.

La respuesta contiene los siguientes campos, separados entre ellos cuando se envía por el carácter "\n" (Espacio en blanco):

- Identificador de traza (Obligatorio): número único asociado a cada traza. Su tamaño es de un byte. Se utiliza para saber si los servidores poseen la misma traza y en caso de que no sea así enviarla.
- Calibración en marcha (Obligatorio): si es cierto, indica al servidor que el analizador se encuentra actualmente en una calibración.
- Estado del menú de calibrado (Opcional): si el campo anterior es cierto, este campo se enviará. Especifica en que parte del proceso de calibración se encuentra el servidor.
- Parámetros de configuración asociados al menú elegido (Opcional): en caso de que el menú en el que nos encontremos requiera un parámetro para reflejar el estado del analizador, éste campo se enviará.
- Estado de la traza (Obligatorio): indica la validez de la traza, siguiendo el siguiente código numérico:
  - 0: la traza actual no es valida y se actualizará.
  - 1: traza valida.
  - 2: traza valida y se ha de actualizar dentro de un segundo.
- Frecuencias, número de puntos, formato de la traza y estado de la calibración (Opcional): variables asociadas a una traza. Se actualizan y envían cada vez que tenemos una traza nueva.
- Traza (Opcional): conjunto de valores que indican la medida actual del analizador.

El valor del identificador de traza se lleva mediante una variable tipo carácter. Se eligió porque tenía el rango suficiente para diferenciar una traza de otra, y era la unidad mínima que se podía enviar con la conexión empleada. Cuando se realiza una petición por parte del Servidor Web, se envía el valor que posee éste. El Servidor VISA lo compara con el valor de la traza actual y envía la traza en función de lo recibido. También se tiene en cuenta el estado de la traza, si su valor es 0, no se enviará, así se realiza un uso eficiente de recursos, ya que la traza que se posee no es válida.

El valor de traza perteneciente al carácter "\n", es un carácter especial que se utiliza única y exclusivamente para indicar al Servidor VISA de que no se dispone de ninguna traza y que ha de enviar la traza sea cual sea el estado. Se utiliza este valor porque la separación entre campos está

indicada por este carácter, y como valor de traza normal podría generar problemas.

El formato de datos de este envío se realiza en ASCII. Esto es porque idealmente el Servidor VISA y Servidor Web están ambos funcionando en el mismo ordenador. Por lo tanto no hay que preocuparse de optimizar el ancho de banda de los datos enviados. Si no fuera así, ésto tampoco plantea un retardo excesivo en las comunicaciones entre los servidores.

2.XII.2 enviacomandos

Su cometido es la comunicación con el analizador. Es el hilo al que se interrumpe cuando se tiene una traza nueva.

El Servidor Web se pone en contacto con este hilo para comunicar los comandos que se quieren enviar. Este le devuelve si el comando realizado ha producido o no algún error y en caso de que lo haya producido el error devuelto.

La codificación de los comandos que se quieren realizar está ordenado por la interfaz del analizador. Indicando el número de menú escogido y la opción realizada.

Al igual que en el caso de enviadatos lo primero es iniciar las conexiones en el puerto 55551 + 2 * puesto. Y si hubiera algún error, ejecutar el cierre del hilo de ejecución.

Lo siguiente es habilitar las interrupciones, para asociarlas a este hilo y obtener la primera traza para enviarla al cliente.

Cuando llega una petición del Servidor lo primero que se realiza es un borrado y deshabilitado de las interrupciones. Impidiendo que mientras enviamos un comando se pueda interrumpir la ejecución del programa principal y por tanto solapar el envío de un comando.

También se cierra el semáforo que evita el acceso por parte del otro hilo a la variable creada del analizador correspondiente.

Después se procesa y realiza el comando indicado. Siguiendo el número de menú y el método asociado correspondiente a la clase analizador.

Dependiendo del modo de disparo en el que nos encontremos se ejecutará uno de los siguientes casos:

- Modo single/hold: en este caso las interrupciones se dejarán deshabilitadas y no se realizará una nueva traza. Excepto en el caso de que el comando recibido sea el de actualizar traza o cambiar a modo single.

- Modo con cambios: si el comando realizado cambia algún parámetro de la traza (número de puntos, frecuencias, filtro de resolución...), se activaran las interrupciones y se iniciará una nueva traza.

- Modo continuo: Se activarán las interrupciones y se iniciará de una nueva traza en caso de que algún parámetro haya cambiado.

A continuación se enviará la respuesta del analizador al Servidor Web para mostrarla por pantalla en

caso de que sea necesario.

Una vez obtenida la traza, en caso de que estemos en modo automático activaremos las interrupciones y se pedirá una nueva traza. Sino se dejarán deshabilitadas.

2.XII.3 menuinstrumentos

Este hilo atiende a las peticiones en el puerto 55550. Se utiliza para conocer los analizadores que hay conectados, su nombre y su tipo.

Una vez asegurada la conexión se entra en un bucle donde se atiende a las peticiones que recibamos. Enviando al servidor web el nombre de cada analizador y su tipo. Separando cada uno de ellos con “\n”.

3. Servidor Web

La ejecución del código web se realiza en PHP. También se incluye código javascript que se ejecutará en el lado del cliente y se comentará más adelante. Se pueden distinguir 3 partes en el código, funciones auxiliares, la obtención de los datos necesarios y el envío de la página web. Al igual que antes se explicará a la par que el código escrito aclarandolo y comentándolo.

3.I Funciones

• Botones

Se utilizan para no tener que escribir el mismo código cada vez que se quiere mostrar por pantalla un botón. Existen 2 versiones, la versión normal cuando se pincha, es un link hacia la misma página Web. La versión Java cuando se pincha se redirige a una función Javascript. Todos los botones, excepto los deshabilitados, reciben 2 argumentos en su función:

1. "$nombre": Indica el texto que hay encima o se incluye en el botón.

2. "$link" o "$función": Indica la acción que se realizará al pinchar en el botón.

Existen 2 tipo de estructuras de botones:

Ilustración 2: Botón normal

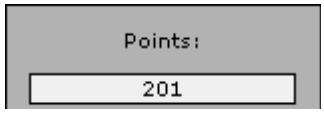


Ilustración 3: Botón con valor

Los tipos de botones con funciones adicionales creados son:

- ***botonjavaparametro***

  Tiene la misma función que botón java normal, solo que el argumento $numero se utiliza para pasar argumentos a la funcion adecuada.

- ***botonmenu***

  Botón con valor que se utiliza en el menú principal. "$Valor" indica el campo que se incluirá en la parte más clara del botón.

- ***formboton***

  Botón con valor que se utiliza en los menús intermedios. Obtiene el valor de menú en el que nos encontramos y lo añade a los parámetros a enviar en la petición de tipo get. "$Valor" indica el campo que se incluirá en la parte más clara del botón y "$opcion" la opción que ha sido escogida para enviarla mediante el formulario.

- ***formbotonjavamenu y formbotonjavanumerica***

  Botón con valor. Ambos se utilizan para mostrar el menú de introducción de valores. La versión numérica incluye también los valores máximos y mínimos que vamos a permitir.

- **Comandos**

Ilustración 4: Comandos

Es el menú que su utiliza para introducir datos, existen 3 versiones: para números enteros, para números decimales y números enteros llamando a una función java. Sus argumentos son:

- $nombre: Indica el nombre del comando. En el caso de la Ilustración 4 sería "Puntos".
- $valor: Es el nombre del input text. También se utiliza para darle valor, ya que su nombre será el valor enviado en la petición get. En la versión java se utiliza también para definir la función java, a la que se llamará al aceptar los datos.
- $opcion: La opción del menú en la que nos encontramos.
- $patron: indica el patrón HTML5 que se aceptará en la introducción de datos.
- $Titulo: será el texto que aparecerá cuando se sitúe el ratón encima del cuadro de texto.
- $min y $max (Solo en Comandos): indica el máximo y el mínimo que se puede introducir como número entero.

- **convertir**

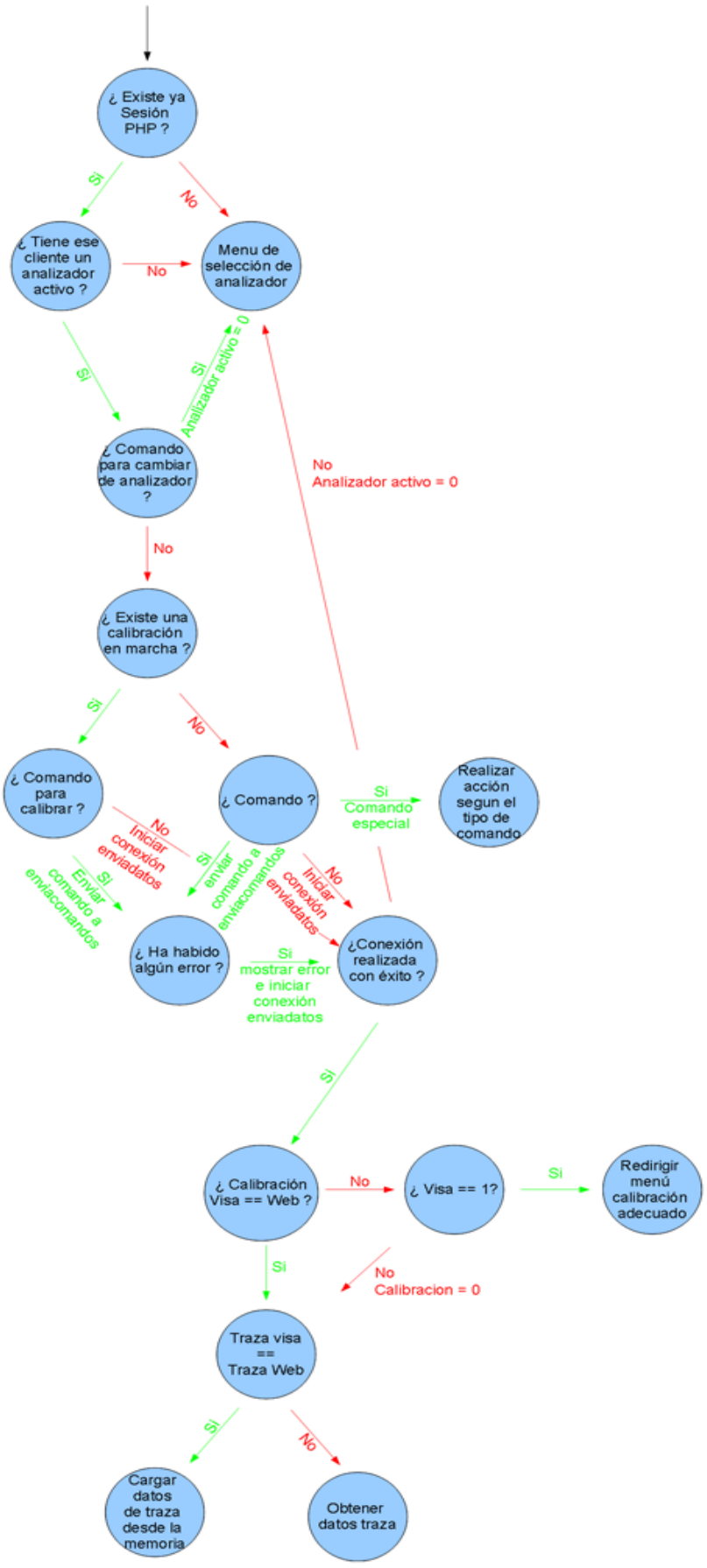
Función que convierte los valores con unidades multiplicativas a un valor entero en la unidad base.

3.II Cuerpo del código

El protocolo HTTP es un protocolo sin estado, con lo cual se mantienen la conexión TCP abierta entre el Servidor Web y el Servidor VISA, hasta que la reacción pedida se haya realizado; para así poder enviar algún error al usuario de la página, en caso de que haya habido algún comando fallido.

También influye en el envío de errores al usuario. Ya que enviamos solo los errores que haya habido asociados al comando que haya sido realizado actualmente. Si el usuario no realiza ningún acceso y hay un error, no tenemos manera de comunicarlo al instante. Igualmente tampoco se puede notificar si hay una nueva traza, o de algún cambio en los parámetros del analizador.

El esquema seguido para atender a cada petición es el siguiente:


¿ Existe ya Sesión PHP ?
Si
No
¿ Tiene ese cliente un analizador activo ?
No
Menu de selección de analizador
Si
Si
Analizador activo = 0
¿ Comando para cambiar de analizador ?
No
Analizador activo = 0
No
¿ Existe una calibración en marcha ?
Si
No
¿ Comando para calibrar ?
¿ Comando ?
Si
Comando especial
Realizar acción segun el tipo de comando
No
Iniciar conexión enviadatos
Si
enviar comando a enviacomandos
No
Iniciar conexión enviadatos
Si
Enviar comando a enviacomandos
¿ Ha habido algún error ?
Si
mostrar error e iniciar conexión enviadatos
¿Conexión realizada con éxito ?
Si
¿ Calibración Visa == Web ?
No
¿ Visa == 1?
Si
Redirigir menú calibración adecuado
Si
No
Calibracion = 0
Traza visa == Traza Web
Si
No
Cargar datos de traza desde la memoria
Obtener datos traza

Cuando recibimos una petición Web, comprobamos si ese usuario ya existe en las sesiones PHP. Si no existe se creará una sesión nueva indicando que el analizador que ha seleccionado el usuario es el “0”.

Cada analizador está indicado por su número de puesto que se equipara con el valor de la variable puesto del servidor VISA. El encargado de transimitir este número es el hilo menuinstrumentos que opera en el puerto 55550.

3.II.1 No hay nada seleccionado

Si no hay nada seleccionado, la variable de sesión “Instrumento” es 0, se envía el menu de selección de analizador.

Si el usuario ya se encontraba en este menú y ya ha escogido una opción se le redirigirá al analizador que haya seleccionado. Para saberlo, se utiliza el número de parámetros que haya enviado el usuario en la petición get. Si ha escogido una opción será mayor que 0.

Cada analizado conectado enviado, se separa por “\n”. Así que con cada fgets se obtendrá un dato.

3.II.2 Analizador activo

Lo siguiente que se comprueba es si se quiere cambiar de analizador, Opción de menú “e”. En caso afirmativo se borrará la variable de sesión “Instrumento” a 0, se igualará el identificador de traza a “\n” para indicar que se ha cambiado de analizador y se redirigirá al menú.

La calibración es un parámetro que es común a todos los navegadores y especifico de cada analizador. Para ello en vez de almacenarlo en variable de sesión, se ha utilizado memoria compartida con el comando shm. La llave para que se utiliza es el valor de puesto del analizador seleccionado. Como es una variable de memoria compartida, se utiliza un sistema de semáforos para regular su acceso. Con $shm_id = shmop_open($shm_key, "c", 0644, 1) abrimos el acceso al semáforo, en caso de que esté ocupado se quedará esperando y con shmop_close($shm_id) cerraremos el semáforo para que puedan acceder otros usuarios.

Si el analizador se encuentra en una calibración activa. Solo se enviaran los comandos de calibración, y comandos especiales. Por el contrario si está desactivada no se pueden enviar los comandos de calibración. Una vez preparado si hay que enviar o no algo a enviacomandos se realizará la conexión. Para detectar cuando enviar algo, se cuenta el número de parámetros get que tenemos y el menú en el que nos encontramos.

Si vamos a enviar un comando de calibración se activará en memoria compartida el valor asignado a “calibración” para ese analizador.

En ambos casos, si se encuentra calibrando o si no, se ha de enviar el comando a enviacomandos en caso de que se precise, y mostrar si hubiera un error por pantalla. En el caso de calibración se mostrará la carga que debemos conectar por el mismo sistema para enviar errores por pantalla.

Por otra parte se encuentran los comandos especiales, que son comandos que no son enviados para el analizador, sino para el Servidor VISA. Estos comandos son:

- Actualizar traza: con este comando indicamos al Servidor VISA que ha de iniciar un nuevo barrido de traza.

- Guardar datos: se envían en formato polar, al Servidor Web los datos de la traza actual que ha recibido el Servidor VISA. Éste los procesa y los envía en forma de archivo al usuario con el formato que haya indicado. Existen 2 formatos:

  - CSV: es un tipo de fichero con formato abierto pensado para representar datos en forma de tabla de manera sencilla. Los valores se representan, se imprimen en formato ASCII, separados por comas entre ellos (punto y coma en países donde la coma se utiliza para separar los decimales). Cada fila de la tabla se especifica mediante un salto de linea. Si algún campo tiene coma, punto y coma, salto de linea o comillas dobles se encierra entre comillas dobles. A parte de eso no se especifica ninguna otra cosa, ni el juego de caracteres usados, ni el orden de los bytes... Cuando se abre el fichero en muchos casos hay que decirle al programa estos datos.

  - Touchstone: en un principio fue un formato creado para el programa homónimo de EEsof, actualmente propiedad de Agilent. En la actualidad es un estándar de facto, que se utiliza para representar parámetros de radiofrecuencia de un circuito.

    Los comentarios se escriben con el signo "!" delante.

    Consta de una primera línea en la que se indica la frecuencia, el tipo de parámetro, los valores representados (Parte real, Magnitud y fase) y el valor de la impedancia.

    A continuación se pueden añadir otros datos como número de puertos, frecuencias... En el caso del proyecto no se incluye nada.

    Y luego se insertan los datos en formato ASCII, siguiendo el siguiente orden:

    frecuencias, valor11,valor12,valor13 .... valorxx-3, valorxx-2, valorxx-1, valorxx

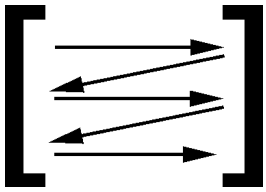
    

    Ilustración 5: Recorrido en la matriz de parámetros S

    Cada valor se compone de 2 partes indicadas en la primera línea del documento.

- Obtener solo datos de traza: este comando es utilizado por script que ejecuta el navegador web. Sirve para que el Servidor Web envíe si tenemos una nueva traza, los datos. Sin estar estos datos en formato de página Web. De esta forma se actualiza la gráfica que tenemos en pantalla, aprovechando la conexión TCP por el puerto 80, que se inicia con el servidor web cuando realizamos una petición Web.

- Cambiar de analizador: se inicializa a 0 el analizador activo y se redirige al menú de

selección del analizador. También se resetea el valor de identificador de la traza actual.

El siguiente paso que se detalla es crear una conexión con el hilo enviadatos. Si no se puede iniciar esta conexión significa que por alguna razón el servidor del analizador se ha caído, o que el valor del analizador actual que poseemos es incorrecto. Por esta razón se redirige al usuario al menú de selección de analizador.

Si la conexión se ha realizado con éxito, se le enviará al servidor el identifiador actual de traza y el menú en el que se encuentra el usuario. Sino existe aún un identificador de traza, se le dará el valor "/n" para indicar al servidor que es la primera vez que se entra al analizador. Después se obtendrá de enviadatos el identificador de traza.

Lo siguiente es comprobar si el valor de calibración está sincronizado. En caso de haber una desincronización hay 2 casos:

- Servidor VISA tiene la calibración activa: en este caso activamos el valor de calibración del Servidor Web para ese analizador (que es común para todas las sesiones) y se redirige al menú de calibración que nos indique el Servidor VISA.

- Servidor VISA no tiene una calibración activa: desactivamos el valor de calibración y proseguimos con el proceso como si ambos valores hubieran sido iguales.

Después de esta comprobación se obtienen los datos del analizador relacionados con el menú que nos encontremos. Por ejemplo si nos encontramos en el menú de calibración, obtendríamos datos de la impedancia del sistema y el kit de calibración usado.

Para saber si hemos o no de recibir una traza se utiliza un acuerdo común entre ambos servidores. Como ambos tienen el valor del otro de identificador de traza pueden decidir si se ha de enviar una nueva traza o no. La condición para recibir y enviar es la misma en ambos. Junto a la traza se envían parámetros que están asociados a ella como son: frecuencia de inicio, frecuencia final, número de puntos, parámetro medido, ancho de banda del filtro de resolución y formato de la traza.

Si tenemos nueva traza nos la descargaremos y sino indicaremos al script en java, que debe de cargar la traza almacenada en el navegador web en el disco duro del usuario. Así nos evitamos enviar la traza con la página web, a no ser que tengamos una nueva.

El último envío se realiza para actualizar la traza mediante javascript. Si la el menú que hemos recibido es el 0 con opción 4. Indicará que hemos de enviar la traza, en vez de la pagina web. Se enviarán los datos necesarios en ASCII en caso de que tengamos una traza nueva, precedida por "1" . Y sino se enviará "0" y se indicará el estado de la traza actual, para que en caso de que se esté realizando una traza, notificar al navegador web que pregunte dentro de un segundo de nuevo.

Los valores asociados a la traza se guardan como valores de sesión PHP. El Servidor Web se descargará una traza por cada cliente activo que tengamos, cuando este cliente acceda, aunque ésta sea la misma.

Ya antes de enviar la página web al usuario, si se está calibrando, se comprobará si la opción y el menú en el que se encuentra es el adecuado, para en caso contrario reenviarle al menú a la fase de calibración en la que se encuentre.

3.III Envío de la página web

La parte fija de la página web es el menú superior. Así como la tabla que separa los menús de la parte gráfica.

Lo primero que se escribe es el titúlo de la gráfica y los menús de introducción de datos. Ambos ocultos y solo están activos en caso de que se requiera activarlos.

Lo siguiente es incluir el espacio en el que se representa la gráfica, al cual se le ha dado el nombre de “chart” y la parte de abajo donde se representan las frecuencias.

La última parte de la página web son los menús de comandos, que se ajustaran al estado en el que se encuentre el usuario gobernado por los parámetros enviados en la petición get. También se tiene en cuenta el tipo de analizador que se está usando para modificar los menús, en caso de que se requiera, para añadir o quitar opciones extra.

4. Navegador del cliente

Los códigos javascript se dividen en 3 partes: la incluida en el código de la página web, las funciones globales de funcionamiento y las funciones especificas de cada menú. Está división está hecha para ahorrar el envió de datos en la página web y no enviar más códigos repetidos de los que sean necesarios.

También almacenaremos en el navegador datos sobre la traza y su representación. En almacenamiento local y de sesión.

4.I Almacenamiento HTML5

4.I.1 Almacenamiento de sesión

Cookies | Almacenaje local | Almacenaje de sesión | Opciones de widgets

Clave	Valor
http://158.42.188.87/Servidor/index.php	
Anchodebanda	10
Div	10
Marcadoractivo	0
Marcadores	[[false,0],[false,0],[false,
Pendiente	3
Posicionref	5
ScaleDivisions	10
Tipodepico	1
Valorref	0
ejeyactivo	true
invertircolor	false
mostrarfrecuencia	true
titulo	""
tituloact	false

Ilustración 6: Almacenamiento de sesión

- ***Anchodebanda***

  Indica el ancho de banda en porcentaje que se utiliza para la función buscar picos.

- ***Div***

  Número de divisiones del eje "Y" de la gráfica. Relacionada con la variable global homónima.

- ***Marcadoractivo***

  Almacena el marcador que actualmente está seleccionado. Si su valor es 0 no hay marcador activo.

- ***Marcadores***

  Matriz que almacena los marcadores de la gráfica. El primer valor indica si está activo y el segundo el punto donde se encuentra el marcador.

- ***Pendiente***

  Almacena la pendiente con la que se buscaran los picos en la función buscarpicos.

- ***Posicionref***

  Indica la división sobre la que se sitúa el eje "X". Relacionada con la variable global homónima.

- ***ScaleDivisions***

  Tamaño de cada división de la gráfica. Relacionada con la variable global homónima.

- ***Tipodepico***

  Tipo de pico que buscará funcion buscarpico. Su posibles valores son:

  - 0: Se busca un pico negativo.
  - 1: Se busca de ambos tipos.
  - 2: Se busca pico positivo.

- ***Valorref***

  Valor numérico sobre el que sitúa el eje de las "X". Relacionada con la variable global homónima.

- ***ejeyactivo***

  Almacena si el eje y es visible. Relacionada con la variable global homónima.

- ***invertircolor***

  Cuando esta activo los colores son blanco y azul. Mientras que si es falso son negro y amarillo.

- ***mostrarfrecuencia***

  Indica si se a de ver la barra de frecuencias que se emplaza debajo de la gráfica.

- ***titulo***

  Cadena que contiene el título de la gráfica. Relacionada con la variable global homónima.

- ***tituloact***

  Almacena se muestra el título de la gráfica o no. Relacionada con la variable global homónima.

4.I.2 Almacenamiento local

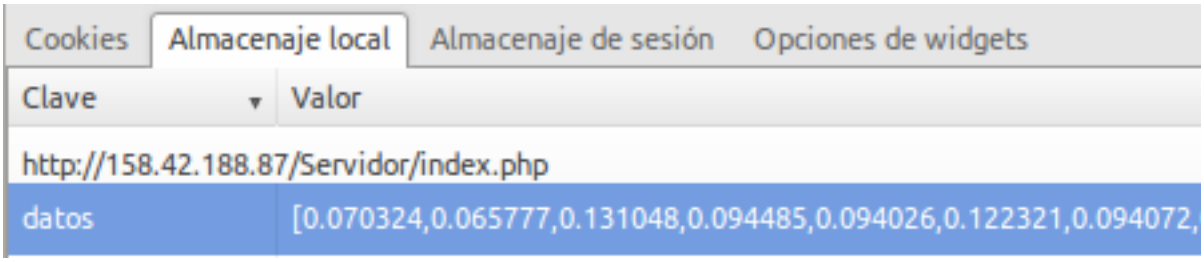


Ilustración 7: Almacenamiento local

La única variable almacenada es la variable que almacena los datos de la traza. Como actualmente el proyecto solo soporta una traza, esta variable es un vector unidimensional.

4.II Incluido en la página

4.II.1 Variables globales del script

- ***chart***

  Variable que almacena el enlace a la gráfica.

- ***Div***

  Número de divisiones de la gráfica, está relacionada con el almacenamiento de sesión homónimo.

- ***fini***

  Frecuencia de inicio de la traza.

- ***ffin***

  Frecuencia de final de la traza.

- ***datos***

  Los valores medidos de la traza.

- ***formato***

  El formato de datos que tiene la traza. Tiene los mismos valores que la variable Formato del servidor VISA.

- ***parametro***

  El parámetro medido por el analizador. Tiene los mismos valores que la variable Parametro del servidor VISA.

- ***Posicionref***

  Indica la división sobre la que se sitúa el eje x, está relacionada con el almacenamiento de sesión homónimo.

- ***puntos***

  Número de puntos de la traza obtenida.

- ***ScaleDivisions***

  Tamaño de cada división de la gráfica, está relacionada con el almacenamiento de sesión homónimo.

- ***titulo***

  Cadena que contiene el título de la gráfica, está relacionada con el almacenamiento de sesión homónimo.

- ***tituloact***

  Almacena se muestra el título de la gráfica o no, está relacionada con el almacenamiento de sesión homónimo.

- ***ejeyactivo***

  Almacena si el eje y es visible, está relacionada con el almacenamiento de sesión homónimo.

- ***Valorref***

  Valor numérico sobre el que sitúa el eje de las x, está relacionada con el almacenamiento de sesión homónimo.

- ***globo***

  Indica si el cursor está sobre la traza, con lo cual hay que dibujar un globo de texto con los datos de ese punto.

- ***actualizando***

  Variable global que se utiliza de semáforo, para que cuando esté activa, la gráfica no de error al intentar borrar el globo de datos si el ratón se sale de ella. Se utiliza mientras se actualizan los datos de la tráfica.

- ***punto***

  El punto sobre el que se sitúa el globo de datos.

- ***y***

  El valor "y" del punto en referencia al cual se situará el globo de datos. Se obtiene a partir del valor del datos sobre el que está situado el ratón.

- ***estadotraza***

  El estado de la traza significando según su valor:

  - 0: La traza actual no es válida y se actualizará.
  - 1: La traza actual es válida y no se va actualizar.
  - 2: La traza actual es válida y se actualizará.

4.II.2 Función principal

Este código está reducido al mínimo posible. Lo primero que se realiza es llamar a la función Inicializar, para inicializar las variables cuyos valores son constantes o están almacenados en la memoria del navegador.

Lo siguiente antes de crear las gráfica es inicializar las variables que dependen de php, estás variables se actualizan y se envían cada vez que el usuario refresca la página. Son las que dependen directamente de la traza y por tanto las que se necesita para representarla.

El siguiente trozo de código está escrito en php, porque dependiendo si se tiene traza nueva o no se realizará una acción:

- Con traza nueva: se enviarán los datos nuevos para que los almacene el servidor web.
- El navegador web cargará los datos de la traza del almacenamiento local.

Ya para finalizar se llamará a Creagrafica para inicializar la gráfica y se configurará el script de java para que cada segundo se llame a la función Actualizargrafica.

4.III Funciones globales

- **Actualizartraza**

Es la función que se utiliza cuando el usuario pulsa el botón Actualizar traza. Está función avisa al servidor Visa que quiere realizar un barrido de una nueva traza, sitúa estadotraza a 0 y redibuja la gráfica con este estado para avisar al usuario que la traza actual no es válida.

- **Actualizargrafica**

Función que se ejecuta cada segundo para actualizar la gráfica en caso de que sea necesario.

Primero se comprueba el estado de la traza, si es 1 quiere decir que no se va actualizar.

Si es así, se realiza una petición web con el menú 0 y la opción 4 indicando que no queremos que se nos envíe la página, solo la traza en caso de que sea necesario. Si el servidor tiene una traza nueva se enviará.

En caso de que haya una nueva traza, despues de recibirla, guardarla y dibujarla. Se llamará a la función para dibujar un globo con los datos, si el usuario tiene el ratón sobre la traza. También se actualizaran los datos nuevos que hemos recibido en los menús y la barra de frecuencias.

- **Convertir**

Convierte un punto de la gráfica a su frecuencia asociada utilizando las variables globales de frecuencia de inicio y final.

- **Guardarpng**

Guarda la gráfica actual en formato “png”. Para evitar problemas con las imágenes de los marcadores y las divisiones de la gráfica. Se redibuja la gráfica siguiendo opciones especiales para este fin.

- **GuardarSVG**

Guarda la gráfica actual en formato de datos de gráfico vectorial. Para evitar problemas con las imágenes de los marcadores y las divisiones de la gráfica. Se redibuja la gráfica siguiendo opciones especiales para este fin.

- **Frecuenciar**

Convierte una variable que contenga una frecuencia en herzios, en una cadena con la unidad multiplicativa más cercana. También acepta números negativos. Se utiliza para representar frecuencias en la gráfica.

- **hacerdiv**

Dibuja las divisiones de la gráfica.

Para dibujarlos se utiliza la gráfica como si fuera un elemento canvas de HTML5. Para ello obtenemos el “render” gracias a la propiedad “renderer” que incluye la clase “charts” del plugin gráfico.

Con número se calcula el valor en la escala de la gráfica de cada división. A la hora de dibujar se dibujan 2 cosas en la gráfica:

- La línea de cada división: Si la división actual tiene el mismo valor que la división sobre la que el usuario quiere dibujar el eje x. La línea se dibujará con un ancho de 2 pixel. Sino se dibujará con un ancho de 1 pixel. Se incluyen también una condición para dibujar el eje “x” negro o blanco en función de los colores escogidos.

- Los número que acompañan a cada división: con ejeyactivo se controla que se muestren los números. Lo primero que se hace es redondearlos, con 2 decimales. Para obtener la posición de la gráfica donde se dibuja, se utiliza la formula 567 – i * 540. Se ha obtenido por la formula de la recta. Sabiendo que la ultima división se sitúa sobre el valor 567, y la primera sobre el valor 17.

- **hacerglobo**

Función que dibuja un globo con los datos del punto de la traza sobre el que sitúa el usuario el ratón.

Para dibujarlos se utiliza la gráfica como si fuera un elemento canvas de HTML5. Para ello obtenemos el “render” gracias a la propiedad “renderer” que incluye la clase “charts” del plugin gráfico.

Lo primero es saber el color del que vamos a dibujar el globo. Una vez lo tenemos el globo se compone de 4 elementos:

- Rectangulo de frencuencias: se dibuja sobre la parte de abajo de la gráfica. Su id es “cuadrado”.

- Texto de frecuencias: texto interior del rectángulo de frecuencias. Su id es “letras”.

- Rectangulo de punto: se dibuja justo sobre la traza donde está situado el ratón. Su id es “cuadrado2”.

- Texto de punto: se dibuja dentro del rectángulo de punto. Su id es “letras2”.

La id se utiliza para borrar de la gráfica estás figuras cuando sea necesario.

- ## hacermarcadores

Función que se encarga de dibujar los marcadores en la gráfica.

Los valores de los marcadores dependen de si el usuario ha activado el marcador diferencial o no. En función de este valor, los valores estarán referidos con respecto a este ultimo valor.

Está función dibuja 2 partes separadas en la gráfica:

- El dibujo del marcador sobre el punto que se encuentra. Para ello se utiliza una propiedad de “highcharts” para cambiar el dibujo de un punto. Poniéndole el del marcador sobre el que nos encontramos.

- Los valores de los marcadores. Para ello se utiliza la propiedad titulo de “highcharts”. Lo haremos sobre una cadena a la que poco a poco se le va añadiendo la información de cada marcador, si este se encuentra activo.

Para los dibujos de los marcadores, se utilizan imágenes en png con la forma del marcador. Para escoger cada una, su nombre es el valor de la variable invertircolor y el número del marcado. Por ejemplo false1.png.

Para dibujar cada marcador se hará un bucle que recorrerá cada uno de ellos y en caso de que se encuentren activos, se dibujaran sobre la gráfica y se añadirán a la variable cadena. También el script tiene en cuenta el caso de que el marcador diferencial este activo, con lo cual los valores se restaran a los de este marcador.

- ## hacermarcadorescaptura

Dibuja los marcadores para crear una captura de pantalla. Su funcionamiento es exactamente igual al de hacermarcadores, con la diferencia que se cambian las imágenes de los marcadores, por el símbolo de un triangulo para evitar que la captura no se pueda realizar.

- **Creargrafica**

Función que se utiliza para dibujar una nueva gráfica.

La variable que se obtiene como argumento de la función sirve para indicar si la gráfica que estamos creando se va a utilizar para realizar una captura de pantalla y así poder aplicarle opciones especiales.

Gran parte del código son las opciones con las que se va a dibujar la gráfica. Indicadas todas ellas en la documentación de "highcharts". También se incluyen las funciones que se aplicaran en caso de que se realicen ciertos eventos sobre la gráfica, se asocían todas a point.

Ocurren cuando:

- ***events: 'click'***

  Se pulsa sobre un punto de la gráfica.

  Se comprueba si hay algún marcador en el punto donde el usuario ha pulsado. Si es así ese marcador pasará a ser el activo.

  Si no hay ningún marcador en ese punto y existe un marcador activo. Se trasladará el marcador activo al punto pulsado.

- ***'mouseOver'***

  Se sitúa el ratón sobre la gráfica.

  Se elimina el posible globo de datos que hubiera podido haber antes y se genera uno nuevo. Limitando la x del punto para no salirse de la gráfica.

- ***' mouseOut'***

  El ratón sale de la gráfica.

  Se borra el globo de datos, solo en el caso de que no estemos actualizando la traza. Porque al actualizar, se activa también está función.

Después de configurar las opciones, el siguiente paso es asociar una traza a la gráfica. Para ello se utiliza un vector nuevo auxiliar en el que se copiarán los datos. Ya que "highcharts" utiliza el puntero al vector que se le da. Modificando el vector de datos sino tenemos la precaución de hacer una copia.

En función del formato será un vector de datos, o una matriz, en el caso de carta de Smith o números polares. También se vigilará que los datos se representen dentro de los límites de la gráfica, y en caso contrario se dibujarán en el borde.

Con la variable cadena modificaremos el título de la traza, incluyendo sus formato, parámetro de medida y configuración gráfica.

Lo siguiente son opciones especificas que se cambian en función de la configuración que haya exigido el usuario o circunstancias de configuración. Como puede ser el cambio de colores, dibujar una carta de smith o adaptar la gráfica para realiza una captura.

Una vez está todo configurado se crea la nueva gráfica y si no estamos en una carta de smith se dibujarán las divisiones y los marcadores llamando a las funciones hacerdiv y hacermarcadores. En el caso de estar realizando una captura solo se llamará a hacermacadorescaptura.

- **Inicializar**

Inicializa los valores de todas las variables globales. También carga los valores almacenados en memoria y modifica el valor del titúlo HTML de la gráfica al elegido por el usuario.

4.IV Función especificas de cada menu

4.IV.1 Menú Scale

A este menú esta asociado el archivo Funcionesgraficas.js.

- ***AutoScale***

  Sitúa los límites de la gráfica entre los límites de la traza. Para ello se calcula el valor máximo y mínimo de los datos, redondeandolos con 3 cifras decimales. Una vez se tienen estos valores se deja el valor de referencia de manera, que deje la gráfica centrada entre estos 2 valores. Y se calcula una escala por división adecuada en base a las divisiones que ya se tenían guardadas.

  Después de cambiar los valores se volverá a dibujar la gráfica.

- ***Divisions***

  A esta función se la llama cuando el usuario a introducido el número de divisiones que quiere en la gráfica.

  Lo primero que se hace es comprobar que los valores introducidos son válidos. Si el número es válido, se guardará y se procederá cambiar el número de divisiones.

  En caso de que Posicionref sea mayor que las divisiones escogidas, lo igualaremos a estas para evitar incongruencias en la representación.

- ***Funcionver***

  Muestra la barra de introducción de comandos, con el texto leyenda y dando al valor inputbox la variable nombre.

  Leyenda se utiliza de esta manera porque no se dejaba pasar como argumento una cadena con espacios.

- ***Funcionvernumerica***

  Muestra la barra de introducción de datos, con el texto leyenda y dando al valor inputbox la variable nombre. Mini y maxi incluyen el máximo y mínimo del número entero que se espera.

  Leyenda se utiliza de esta manera porque no se dejaba pasar como argumento una cadena con espacios.

- ***Funcionocultar***

  Oculta la barra de introducción de datos.

- ***MarkerReference***

  Sitúa el valor de la posición de referencia al marcador actual seleccionado, si está seleccionado.

- ***ReferencePosition***

  A esta función se la llama cuando el usuario ha introducido el valor de la división en la que quiere situar el eje "x".

  Lo primero que se hace es comprobar que los valores introducidos son válidos. Si el número es válido, se guardará y se procederá a cambiar el valor de la división sobre el que se sitúa el eje "x". Modificando todos los valores que están relacionados con este.

- ***ReferenceValue***

  A esta función se la llama cuando el usuario a introducido el valor de referencia del eje "x".

  Lo primero que se hace es comprobar que los valores introducidos son válidos. Si el número es válido, se guardará y se procederá a cambiar el valor de referencia sobre el que se sitúa el eje "x". Modificando todos los valores que están relacionados con este.

- ***ScaleDiv***

  Cambia el tamaño de cada división de la gráfica.

  Una vez se comprueba que se ha introducido un valor numérico valido se cambiará la escala y se dibujará de nuevo la gráfica.

- ***window.onload***

  Es la función que se ejecuta al cargar la página. Da valores a los botones del menú Scale, puesto que estos valores dependen de los valores almacenados, la memoria del navegador y PHP no tiene constancia de ellos.

4.IV.2 Menú Display

A este menú esta asociado el archivo Funciondisplay.js.

- ***EditTitleLabelverfuncion***

  Modifica el titulo HTML de la gráfica.

- ***EditTitleLabelver***

  Muestra el menú para introducir el título de la gráfica en la parte superior.

- ***Funcionocultar***

  Función que oculta el menú de introducción de título.

- ***Frequency***

  Oculta o muestra la barra de frecuencias, dependiendo del valor de mostrarfrecuencia.

- ***GraticuleLabel***

  Oculta o muestra los valores numéricos del eje “y”.

- ***InvertColor***

  Invierte los colores de la gráfica.

- ***TitleLabel***

  Oculta o muestra el titulo HTML de la gráfica.

- ***window.onload***

  Es la función que se ejecuta al cargar la página. Da valores a los botones del menú Display, puesto que estos valores dependen de los valores almacenados, la memoria del navegador y PHP no tiene constancia de ellos.

4.IV.3 Menú Markers

Esté menú tiene 2 archivos de script, ya que tiene el menú principal el cual carga el archivo Funcionactivarmar.js y el submenú “Clear Marker Menu” que carga el archivo Funciondesactivarmar.js.

4.IV.3.i Funcionactivarmar:

- ***Marker***

  Activa el marcador indicado por el argumento i.

  Se comprueba si el punto en el que se encontraba el marcador anteriormente es mayor que el número actual. Para que en caso de que esto ocurra, situarlo al final de la gráfica.

- ***RefMarkerMode***

  Activa o desactiva el marcador referencial, según se encuentre o no activo.

4.IV.3.ii Funciondesactivarmar:

- ***AllOFF***

  Desactiva todos los marcadores activos.

- ***Marker***

  Desactiva el marcador indicado por i.

4.IV.4 Menú search

A este menú esta asociado el archivo Funcionessearch.js.

- ***Anchodebanda***

  Modifica el valor anchodebanda, que rige la función para buscar picos.

  Se comprueba si el valor que se ha introducido es válido. Este debe de encontrarse entre 0 y 100 porque es un porcentaje.

- ***BuscarPico***

  Busca el primer pico con las características introducidas.

  Si no tenemos un marcador activado, se activará el primer marcador y se empezará a buscar un pico a partir del primer punto de la gráfica.

  Si por el contrario se tiene un marcador activo, se buscará a partir de el marcador activo que el usuario haya seleccionado.

  Para buscar el pico se utiliza la función SiguientePico creada para este fin.

  Si la posición obtenida es distinta del número de puntos, el final de la gráfica, quiere decir que se ha encontrado un pico. Con lo cual se trasladará el marcador activo a dicho punto.

- ***Funcionvernumerica***

  Activa el menú para introducir datos.

- ***Funcionocultar***

  Oculta el menú de introducción de datos.

- ***Maxmarcador***

  Función que busca el máximo de la gráfica.

  Si no se tiene ningún marcador activo, se activará el primer marcador para colocarlo sobre el máximo de la gráfica.

  Para buscar el máximo se utiliza un bucle que recorre todo el vector de datos y encuentra la posición del punto cuyo valor es más alto.

- ***Minmarcador***

  Función que busca el mínimo de la gráfica.

  Si no se tiene ningún marcador activo, se activará el primer marcador para colocarlo sobre el máximo de la gráfica.

  Para buscar el mínimo se utiliza un bucle que recorre todo el vector de datos y encuentra la posición del punto cuyo valor es más bajo.

- ***MultiPico***

  Busca todos los picos de la gráfica que tengan las características incluidas por el usuario. Hasta un máximo de 9 picos.

  Para marcarlos, se utilizan secuencialmente todos los marcadores del 1 al 9.

  La forma de operar es utilizando un bucle cuyo límite es el número de marcadores y las posición del punto actual de la traza en la que nos encontramos.

  Con SiguientePico se recorre cada punto de la traza parandose la función en cada punto encontrado. Cada vez que se encuentra un punto se borra el marcador actual y se inicia una nueva pasada al bucle.

  Si se ha encontrado algún pico se vuelven a dibujar los marcadores.

- ***Pendiente***

  Modifica el valor Pendiente que rige la función para buscar picos.

  Se comprueba si el valor que se ha introducido es válido. Este debe de encontrarse entre 0 y

1000 por poner un límite a la pendiente escogida.

- ***SiguientePico***

  El funcionamiento del bucle de SiguientePico se detalla en la memoria del proyecto.

  Son 3 bucles que se ejecutan dependiendo del punto en el que nos encontremos. El primer bucle es para los puntos del inicio, el segundo para los puntos intermedio y el último para los puntos que se encuentren al final de la traza.

  La condición de finalización de cada bucle es que se haya acabado ese tramo o que hayamos encontrado lo que buscamos.

  Para cada trozo de datos escogido en función del ancho de banda, se haya el máximo y el mínimo y lo comparamos con el punto actual. Si cumple las condiciones habremos encontrado un pico.

  Para elegir la siguiente iteración del bucle, se escogerá el máximo o mínimo local de cada trozo o el punto final de cada tramo.

- ***window.onload***

  Es la función que se ejecuta al cargar la página. Da valores a los botones del menú Display puesto que estos valores dependen de los valores almacenados la memoria del navegador y PHP no tiene constancia de ellos.

4.IV.5 Menú Marker Function

Modifica la frecuencia indicada por i al valor del marcador que se encuentra activo.

Lo que hace es redirigir al usuario a una página indicando por el menu “c” que nos encontramos en Marker Function y se le añade el valor de la frecuencia.