Proyecto Fin de Carrera

Escuela superior técnica de Ingenieros de Telecomunicaciónes

Servidor para uniformizar el manejo de distintos analizadores de redes

Autor: Rubén Del Olmo Esteban

Director: Vicent Miquel Rodrigo Peñarrocha

Fecha: Septiembre 2013

Índice de contenido

1. Presentación

1.I Introducción

El analizador de redes es un sistema de medida que, en base a la detección de una serie de señales de entrada y de salida sobre sus puertos, caracteriza lo que haya conectado a éstos mediante sus parámetros de dispersión.

Su alto coste no permite, para un laboratorio de prácticas comprar más de uno de estos a la vez. Con el paso del tiempo, en el laboratorio de microondas de la escuela técnica superior de Ingenieros de Telecomunicacion de la Universidad Politécnica de Valencia, se ha ido adquiriendo nuevos analizadores de redes, contando en la actualidad con 3 equipos. Por motivos del mercado, no se pudo comprar analizadores exactamente iguales que el primero. Con lo cual la situación del laboratorio es de 3 analizadores distintos que son:

- HP 8714B
- HP 8714ET
- Agilent E5062A

En el caso de los dos primeros el modo de funcionamiento, menús y opciones son parecidos. Mientras que el 3º cambia por completo siendo más nuevo y utilizando una interfaz totalmente distinta.

Este hecho, a la hora de impartir prácticas, plantea el siguiente problema: es necesario enseñar a los alumnos la utilización de 2 interfaces distintos, con el incremento de tiempo y la confusión que acarrea para su enseñanza. Este es el punto de partida del proyecto: dotar de una interfaz común a todos los analizadores, que permita emplear eficazmente el tiempo de laboratorio en medir y no en localizar opciones de distintos menús.

1.II Planteamiento del problema a solucionar

En un principio, la interfaz de los analizadores no se puede cambiar. Como consecuencia se debe crear alguna forma de presentar al usuario un acceso uniforme a todos los analizadores. Para ello, en esté proyecto, nos conectaremos a los analizadores mediante instrumentación remota y se creará un programa que se encargará de dotar de dicha interfaz común a los distintos analizadores.

La instrumentación remota es un tema ampliamente estudiado, para el cual actualmente la mayoría de instrumentos están preparados. También existen múltiples soluciones comerciales para la programación del control remoto. Más adelante se explicará cual de estás soluciones se ha decidido escoger.

Limitaciones del proyecto

1.II Planteamiento del problema a solucionar

Como interfaz de ejemplo para el proyecto se ha escogido la del analizador más nuevo: Agilent E5062A.

Si se intentaran implementar todas las funcionalidades que nos ofrece este analizador, el proyecto sería demasiado grande y no se podría realizar. También el uso de instrumentación remota permite nuevas funcionalidades (como por ejemplo acceso multiusuario), creando también un grado de complejidad muy alto. Por ello vamos a limitar sus funciones basándonos en el uso que se les estaba dando en clase a los analizadores, para así poder obtener un programa funcional, que se pueda utilizar en clase e ir ampliandolo convenientemente en un futuro.

Del uso del laboratorio podemos partir de:

- Solo un grupo de alumnos accede al analizador. Nadie más modifica parámetros, ni toca nada que no sean ellos.

- Los alumnos se encuentran frente al analizador, con lo cual pueden realizar el calibrado y cambiar el dispositivo que se está midiendo.

Teniendo en cuenta estas 2 situaciones, podemos limitar el programa de la siguiente manera:

1. Solo un programa podrá acceder al analizador: se deberá de utilizar dicho programa como medio para hacer llegar las ordenes y comandos requeridas por el usuario. Su consecuencia más directa es que no se podrá manejar el analizador ni en local (manejar el panel frontal), ni desde 2 programas distintos a la vez.

2. Solo habrá un "usuario" accediendo al analizador: definiendo usuario, como un cliente o conjunto de clientes, con su propios parámetros de configuración del analizador. Esto no impide el acceso de varios clientes al mismo analizador. La consecuencia es que los parámetros serán compartidos entre todos y las modificaciones realizadas en la configuración del analizador, afectará a todos.

3. El calibrado se realizará "in situ": se excluye la opción de cargar un fichero en remoto con una calibración predefinida anteriormente.

A nivel técnico estas limitaciones, simplifican el proyecto. El programa que se realice las siguientes propiedades:

1. El programa que controle el analizador, no tendrá que preguntar cada vez que acceda, por el estado de todos los parámetros; puesto que al ser el único que accede a él, los valores que guarda serán los mismos que los del analizador. Solo deberá preguntar por aquellas opciones que hayan cambiado.

2. Cada analizador tendrá solo un grupo de variables (Parámetros de configuración). En caso de que hubiera más de un usuario, dependiendo del tipo de funcionamiento elegido obtendría más. Con esto evitamos realizar un control de usuarios, y lo dejamos abierto para futuros proyectos.

3. Evitamos el tener que crear algún tipo de conexión con el cliente para subir el archivo de calibración. También en caso de que fuera necesario, mantener algún servidor de ficheros para almacenar estos datos.

1.III Posibles estructuras

Ahora que se tienen los requisitos técnicos fundamentales del proyecto presentados, se puede pasar a presentar la división de funciones que se ha realizado en el conjunto de programas. Así como la explicación de dicha división, sus ventajas y desventajas. Las estructuras están presentadas en orden cronológico hasta llegar a la estructura final que es la utilizada actualmente.

1.III.1 Estructura con un nivel

La primera estructura que se podría pensar, por ser la más sencilla, para solucionar el problema que nos atañe sería:

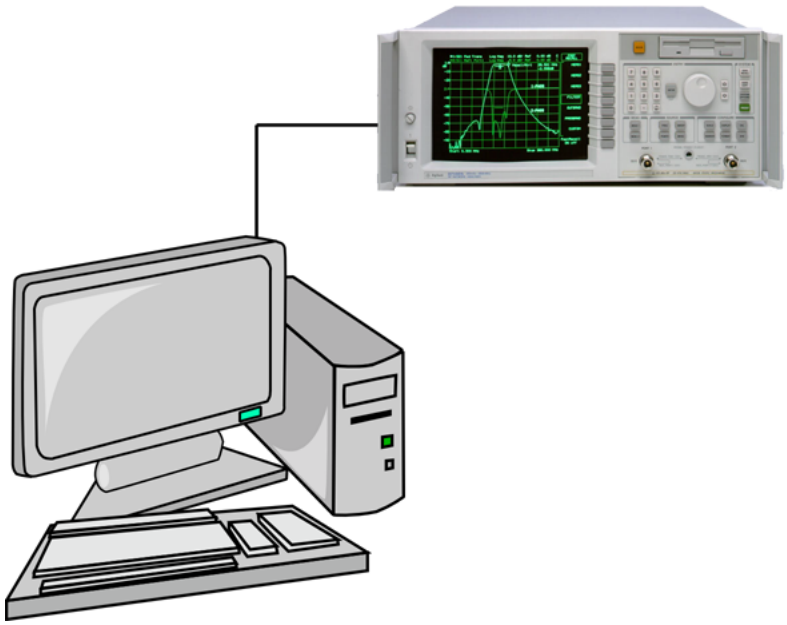


Ilustración 1: Estructura de un nivel.

Esta estructura es muy sencilla de implementar. El programa se instalaría en cada ordenador del laboratorio que se quisiera que pudiera acceder al analizador.

Como problema presenta que en caso de querer ver desde varios ordenadores y/o dispositivos la información del analizador, habría que implementar algún tipo de comunicación entre ambos programas, complicando el código innecesariamente.

Además de esto tendría otra serie de ventajas e inconvenientes:

Ventajas:

1.III Posibles estructuras

- Sería la solución a coste mínimo en dinero. Puesto que los ordenadores del laboratorio ya están instalados y solo habría que invertir en las conexiones.

- Es la solución más simple.

- Es la opción que a nivel de seguridad, menos problemas crea.

- En caso de que fallara un analizador no afectaría a los demás.

Desventajas:

- No da ninguna versatilidad, solo se puede acceder desde el ordenador del puesto.

- No se tiene ningún registro externo de las ordenes que envía el usuario al analizador.

- Las opciones están limitadas a el uso de un único ordenador. Sería como tener una réplica del analizador en el ordenador.

- Aunque sea la más simple, requiere el uso de un lenguaje con interfaz gráfica, así como capacidad para realizar gráficas a tiempo real.

- En caso de que se quisiera cambiar alguna parte del funcionamiento del programa, como está unido todo en un bloque, un cambio puede provocar errores en todos los niveles de funcionamiento del programa.

A pesar de sus ventajas, no se decidió escoger esta estructura, porque es preferible una mayor versatilidad en el proyecto. Así como que 2 usuarios puedan ver desde distintos dispositivos el mismo analizador.

1.III.2 Estructura con 2 niveles

Para solucionar este problema se aplicó la técnica “divide y vencerás”. Consiste en dividir el funcionamiento del proyecto en distintos programas, cada uno con su función. Se dividió en 2 partes, que llamaremos servidor central y cliente.

- Servidor central: es el encargado de comunicarse con los analizadores, proporcionar las datos necesarios al cliente para su funcionamiento, transmitir las ordenes que le indique el cliente y detectar los analizadores conectados, su modelo y adaptarse para obtener la información que requiera el cliente.

- Cliente: su función es la de presentar por pantalla los datos obtenidos del analizador por medio del servidor central, así como dar una interfaz gráfica al usuario para poder manejar éste.

1.III Posibles estructuras

La estructura del proyecto quedaría así:

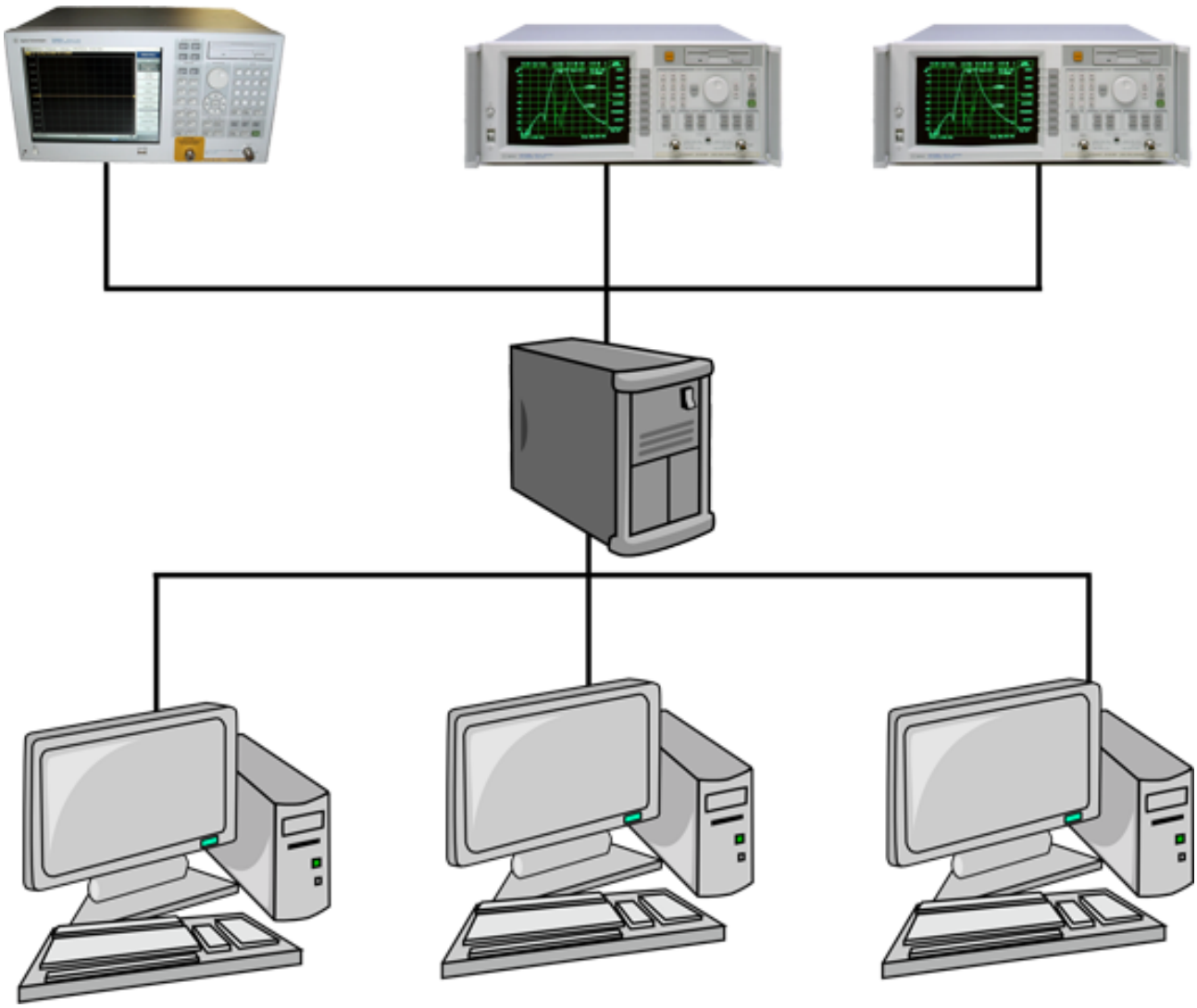


Ilustración 2: Estructura de dos niveles.

Las ventajas y desventajas de esta estructura son:

Ventajas:

- Aísla el acceso directo al analizador del mundo exterior, sin repercutir en el acceso al servidor.
- Se puede implementar un control de acceso por usuarios. También se puede se puede controlar a que funciones del analizador pueden acceder.
- Facilidad de acceso desde cualquier ordenador que se esté conectado con el servidor.

Desventajas:

- Más coste que el anterior, puesto que requiere un servidor dedicado para los analizadores.
- La estructura es más complicada.
- Si falla el servidor, falla todo y no se puede acceder a ninguno de los analizadores.

Se decidió tomar esta estructura a pesar de sus desventajas, debido a su capacidad de adaptación a distintos escenarios de uso, para añadir compatibilidad al proyecto, y así poder conectar otros dispositivos a parte de ordenadores. Se decidió que el servidor de cara al usuario fuera una página web.

También se decidió que por la parte del servidor, para la comunicación con los analizadores, se utilizarían las librerías VISA. Las librerías VISA implementan funciones de comunicación, control y detección de instrumentos, facilitando la programación y detección de errores.

Ambas decisiones están tomadas bajo el mismo criterio por el que se tomó la estructura: intentar que el proyecto sea lo más universal posible.

A cambio esto plantea un problema. Al utilizar la librerías VISA, el lenguaje de programación es C bajo Windows. Se planteó la solución de usar un servidor web integrado con el programa principal, pero esto generaba problemas de seguridad. Así que se optó por separar el servidor web, del servidor con las librerías VISA.

1.III.3 Estructura final

Con esto ya tenemos la estructura final del proyecto. En este momento se aplicó una estrategia de tipo “Top-down” para empezar a desarrollar el proyecto. Utilizando está estrategia cada parte del sistema de definió como una “caja negra” con una serie de funcionalidades basadas en las especificaciones desarrolladas al inicio.

- Analizador: Su función es la de medir los dispositivos y proporcionar los datos necesarios sobre medidas y parámetros de funcionamiento al servidor VISA. También debe de avisar cuando tiene un datos nuevos al servidor para que este los recoja. No debe comunicarse con ningún otro programa, ni utilizarse en local. La comunicación con el Servidor VISA se realiza mediante comandos SCPI.
- Servidor VISA: Es el encargado de detectar los analizadores conectados en todos los medios disponibles e iniciar y mantener una comunicación con ellos. Debe obtener todos los parámetros de configuración del analizador y responder con los datos y ordenes necesarias hacia el analizador cuando el Servidor web lo requiera. Utiliza librerías VISA para comunicarse con los analizadores y TPC/IP para comunicarse con el servidor web.
- Servidor Web: Es la pasarela entre el usuario final (Navegador web) y el Servidor VISA.

1.III Posibles estructuras

Aísla al Servidor VISA del exterior protegiéndolo frente ataques informáticos. También debe dar formato a la información recibida del Servidor VISA y convertirla en forma de página web para que el navegador Web pueda representarla.

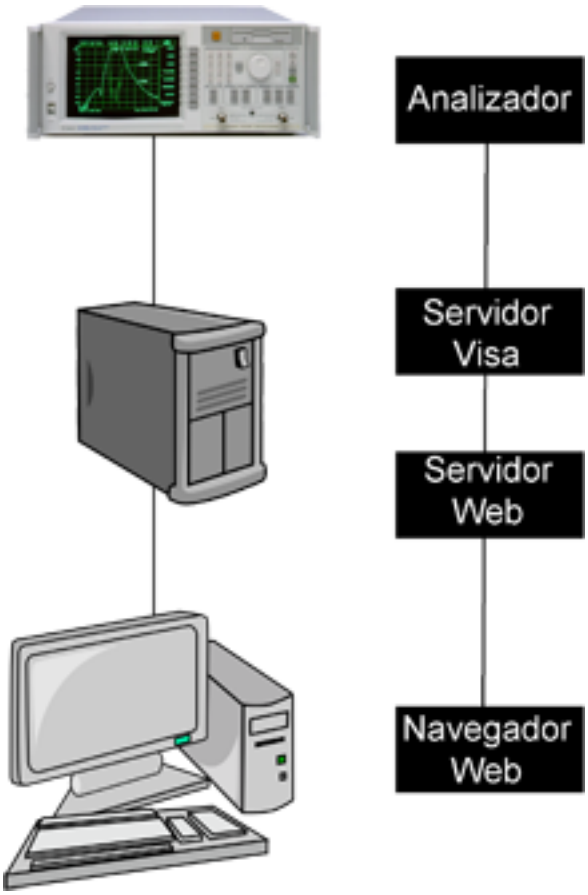


Ilustración 3: Estructura final.

- Navegador Web: Representa por pantalla al usuario la información recibida del servidor. Tanto a nivel gráfico como a nivel de texto. Para poder funcionar necesita tener instalado Javascript.

2. Partes del proyecto

En este apartado se desarrollará cada una de las "cajas negras" descritas anteriormente, tratándolas como elementos independientes entre si que han de realizar las funciones mencionadas.

2.I Analizador

Como ya se ha mencionado antes, en el laboratorio existen 3 tipos de analizadores. El contenido de este proyecto se basa en los 2 más viejos, siendo los modelos HP 8714B y HP 8714ET. El analizador más nuevo se deja como línea futura para otros proyectos. Ya que de momento no es necesario implementarlo, porque la interfaz del proyecto es la misma que la que presenta este de cara al usuario.

A efectos prácticos, excepto por algunos cambios, los 2 analizadores soportados por el proyecto tienen los mismos comandos y características casi en su totalidad. Así que trataremos a ambos de la misma manera. Para simplificar términos, a partir de ahora, cuando hable de analizadores me referiré solo a la serie HP 8714.

En este caso los analizadores poseen 2 canales para tomar las medidas. Un canal es una determinada configuración del analizador para tomar medidas. Incluyendo datos como las frecuencias, el puerto del analizador por el que se toman las medidas, número de puntos de medida, etc...

En los analizadores 8714, aunque por defecto no es así, los canales están desacoplados, es decir, se puede definir un rango distinto de frecuencias para cada canal. En el caso del proyecto, solo se utiliza el primer canal para realizar medidas. No se consideró implementar más de un canal, debido a que durante las prácticas esta característica no se utiliza y requería un tiempo de desarrollo mayor, así como más cambios en el código del programa.

Cada canal posee una serie de parámetros de configuración, que junto a los generales del analizador, debemos explicar para entender el funcionamiento del analizador. Parte de estos parámetros más adelante se verán reflejados como variables asociadas a cada analizador en el Servidor VISA. Los parámetros más importantes son los siguientes:

- Start Frequency: frecuencia de inicio en la que se toma la medida.
- Stop Frequency: frecuencia en la que se para de obtener la medida.
- Sweep Time: tiempo que se tarda en realizar un barrido desde la frecuencia inicial a la frecuencia final. Puede ser:
    - Automático: en cuyo caso el instrumento lo fijará solo.

- Manual: el usuario lo introducirá. Este tiempo nunca debe ser menor que un cierto valor que depende del número de puntos de traza y ancho del filtro de resolución utilizado.

- Number of Points: número de puntos equiespaciados en la frecuencia, en los que toma muestras el analizador.

- Average Factor: factor de promediado. Con este valor se indica si se quiere realizar la media de varias trazas y el número de trazas que se toman para realizar el promedio.

- System Bandwidth: su valor es el ancho de banda del filtro de resolución del sistema.

Además de los parámetros de cada canal, el analizador también tiene parámetros globales. El más destacable es: Smith Chart Z0 que especifica la impedancia de entrada del sistema.

2.I.1 Comandos SCPI

Para comunicarnos con los analizadores utilizaremos comandos SCPI (Standard Commands for Programmable Instruments). SCPI es un estándar creado en 1990 para ser usado en el control de instrumentación. Esta definido por la norma IEEE 488.2 que especifica las sintaxis y formato de datos de las ordenes. Es un protocolo de nivel de aplicación, con lo cual la capa física no está definida. Tampoco incluye más control de errores que el que incluya los protocolos que se utilicen por debajo. Los comandos SCPI pueden servir para 2 tipos de operaciones: para preguntar por un valor o para ordenar una acción. Algunos comandos pueden servir para ambas cosas.

Los comandos se agrupan en forma de árbol, separando cada una de sus ramas por ":". Por ejemplo el comando "SENSe1:FREQuency:STARt 300 KHZ". Pertenecería a la rama SENSe1(Refiriéndose al canal 1 del analizador), FREQ (Refiriéndose a la frecuencias de analizador) y STARt (Refiriéndose a la frecuencia de inicio del analizador).

En el mismo ejemplo podemos ver la sintaxis de un comando para ordenar una acción, en el ejemplo para cambiar la frecuencia de inicio. La rama de comandos que hemos visto antes sirven para definir la acción que queremos realizar. Mientras que con "300 KHZ" expresamos las opciones de la acción realizada. En este caso fijando a 300 kilohercios la frecuencia.

Si se hubiera querido preguntar por la frecuencia de inicio. El comando se habría escrito de forma similar al anterior, pero con un cambio: "SENSe1:FREQuency:STARt?". El interrogante final indica que es un pregunta y esperamos respuesta por parte del dispositivo.

El significado de las mayúsculas es que siempre debemos escribirlas para comunicarnos con el dispositivo. Mientras que las minúsculas no son necesarias, así bien el comando inicial se podría usar de la siguiente manera: "SENS1:FREQ:STAR 300 KHZ"

En la norma IEEE 488.2 se especifican varios comandos que deben de ser implementados por cualquier instrumento que acepte comandos SCPI. Estos comandos son:

2.I Analizador

*CLS	Borra los registros de estado.
*ESE	Habilita el registro de eventos.
*ESE?	Pregunta que eventos están habilitados.
*ESR?	Pregunta por el estado del registro de eventos.
*IDN?	Identificación del instrumento.
*OPC	Comando finalizado.
*OPC?	Pregunta si el comando ha finalizado.
*RST	Comando de reset.
*SRE	Habilita el servicio de interrupciones.
*SRE?	Pregunta por el estado del servicio de interrupciones.
*STB?	Pregunta por el byte de estado.
*TST?	Cadena de testeo.
*WAI	Cadena de espera.

Todos han sido usados en el proyecto para lograr el funcionamiento óptimo del analizador. Más adelante se hablará del uso específico que se le ha dado a “OPC”. Este comando sirve para preguntar cuando el comando anterior a OPC ha finalizado. La diferencia entre “OPC” y “OPC?”, es que el segundo se espera a recibir la respuesta por parte del instrumento, mientras que el primero se generará una interrrupción cuando haya acabado el comando anterior.

Otro comando importante es *WAI. Se utiliza para sincronizar comandos. Implica que el comando que vendrá a continuación del *WAI no se podrá realizar, hasta que el comando precedente a *WAI no haya sido realizado. Junto con OPC, juega un papel importante para evitar solapamiento entre comandos. Esto significa que si un comando no ha finalizado, y se recibe otro comando nuevo, no ejecutará el nuevo para evitar que el comando anterior se cancele o dé error.

2.I.2 Interrupciones

Una interrupción es una señal recibida en un programa. Cuando está señal es recibida, el programa para la ejecución actual y ejecuta un código especifico para atender a dicha interrupción. En el caso del analizador las interrupciones las realiza él sobre el Servidor VISA, para que este pueda leer la traza y guardarla.

El método natural para obtener trazas a tiempo real de los analizadores es éste. Puesto que, nos permite no tener un programa sondeando constantemente al analizador hasta que acabe de medir una traza. Evitando así un gasto de CPU innecesario.

El control de las interrupciones se realiza desde los registros de estado del analizador.

2.I.2.i Registros de estado

Los registros de estado de los analizadores contienen información sobre si mismo y las medidas que realiza. Estos registros están basados en el siguiente modelo:

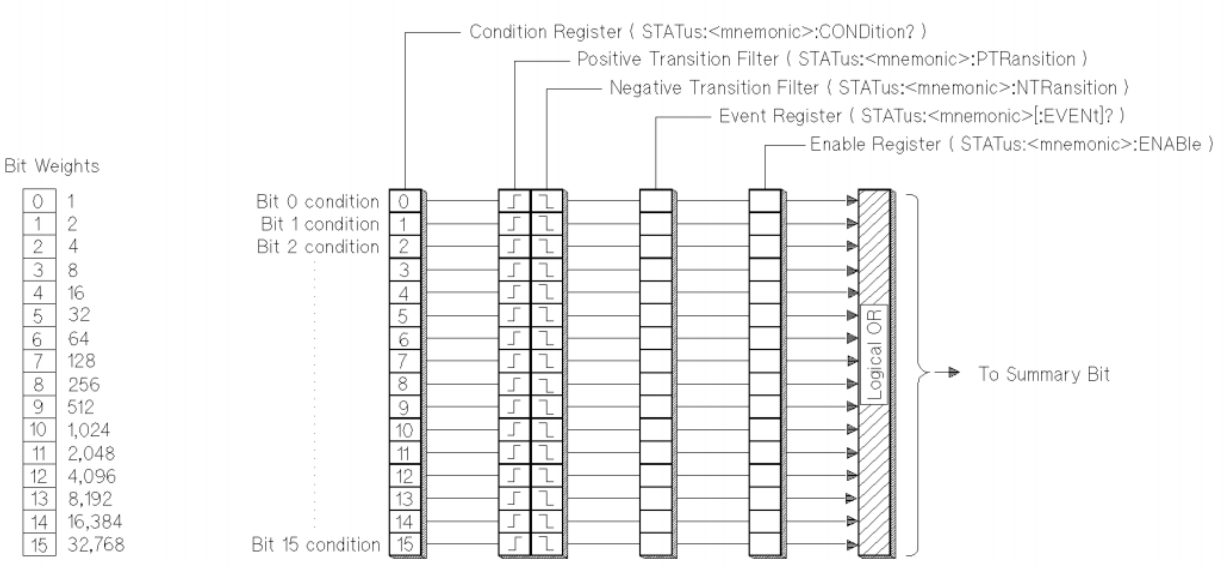


Ilustración 4: Modelo de registros de estado.

Condition Register: el registro de condición, monitoriza el estado del analizador. El valor del registro de condición se actualiza a tiempo real. Cuando la condición a la que es referida ese bit es verdadera, su valor es 1 y 0 para falso. Este registro es de solo lectura.

Transition Registers: los registros de transición se activan cuando hay un cambio en el bit de condición. De 0 a 1 para los positivos y 1 a 0 para los negativos. Se pueden activar ambos para que los 2 cambios sean notificados. Estos registros no son afectados por el comando "*CLS" o al ser leídos. Se resetea su estado después de realizar "*RST2" o después del comando SYSTem:PRESet.

Event Register: el registro de eventos guarda cualquier cambio ocurrido en los registros de

transición. Se activa a 1 el bit correspondiente y guardándose este estado hasta que es leído o es utilizado el comando "*CLS". Es un registro de solo lectura.

Enable Register: este registro activa si los cambios producidos en el registro de eventos son transmitidos a la OR lógica que controla el bit de estado. Este registro es de escritura/lectura y se pone a 0 cuando se realiza el comando "*CLS".

Los analizadores utilizan esta estructura, pero siendo los registros de tamaño de 8 bits. Un pequeño esquema de la estructura que utilizan es el siguiente:

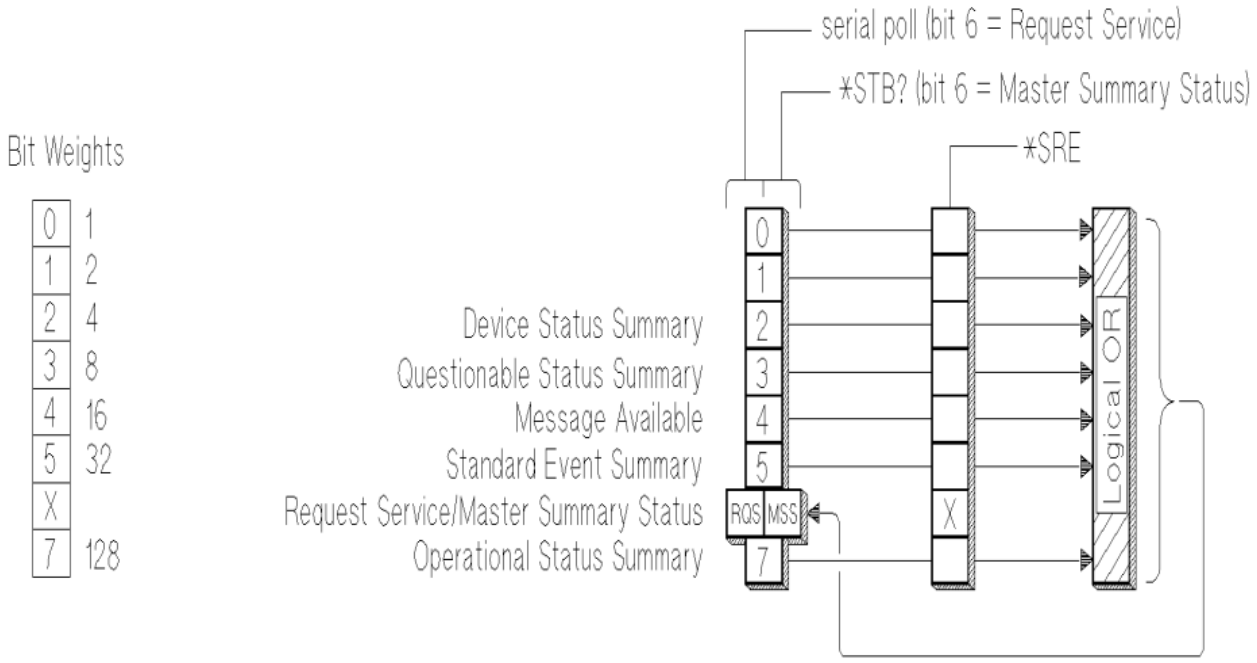


Ilustración 5: Registro de estado principal.

Cada uno de los bits está ligado a su vez a otros registros, que permiten especificar un poco más que opciones queremos. Así para activar una interrupción por parte del analizador, deberemos activar los bits correspondientes que queramos monitorizar en cada una de sus secciones. Activandolos en el registro general.

2.I.2.ii Generación de una interrupción

Una vez el analizador esta preparado para generar interrupciones, el proceso que sigue es el siguiente: cuando hay un cambio en el bit de estado, el analizador inicia una petición de respuesta, "SQR" (Service request), si se cumplen las siguientes condiciones:

- El bit correspondiente en Service Request enable resgister está a 1.

- El analizador no tiene otra petición de respuesta pendiente. Es decir el controlador no ha leído aún el byte de estado del analizador.

Cuando estas condiciones se cumplen, si se está trabajando por GPIB, se activa la linea de interrupción. Y ya es el controlador, en este caso el Servidor VISA, el encargado de averiguar que instrumento ha activado la linea de interrupción y el motivo de esta.

2.I.2.iii Registros utilizados en el proyecto

El primer problema se encontró con el analizador 8714ET. Debido a que a pesar de poderse utilizar remotamente mediante una conexión TPC/IP, no lleva implementado el estándar para interrupciones por TPC/IP, imposibilitando el utilizar interrupciones vía este medio. Esto fuerza a que ambos analizadores estén conectados por GPIB al Servidor VISA.

El segundo problema encontrado es que ninguno de los analizadores poseen en sus registros de estado un bit que indique cuando se ha terminado de realizar una traza. La solución implementada pasa por utilizar el comando “OPC”.

Los analizadores poseen un comando SCPI, “INIT1”, cuya actuación depende del barrido interno de traza del analizador.

- En traza continua: Cuando el comando haya finalizado indicará que se ha acabado un barrido de traza y que la traza actual es válida para leerla.

- En modo hold: Inicia un barrido de traza y cuando haya acabado el comando indicará que se puede leer.

Hay que destacar en ambos casos que la ejecución del comando acaba cuando se ha realizado una traza, para ello se puede utilizar los comandos OPC en sus 2 versiones para saberlo.

En este proyecto el analizador se utiliza internamente en modo hold, indicando el Servidor VISA cuando se ha de realizar la siguiente traza. Para que el analizador nos interrumpa al utilizar el comando OPC debemos activar los registros de estado consecuentes a dicha función. En esté caso habría que activar el bit 5 (Standard Event Summary) como se puede ver en la ilustración5 .

También como se ha visto anteriormente hay que activar el bit necesario de la sección Standard Event Summary. Los bits de está sección son los que muestra la ilustración6.

El bit uno es el correspondiente a operación completada y por tanto al comando OPC.

Así se consigue que el analizador interrumpa al Servidor VISA cuando una nueva traza está lista.

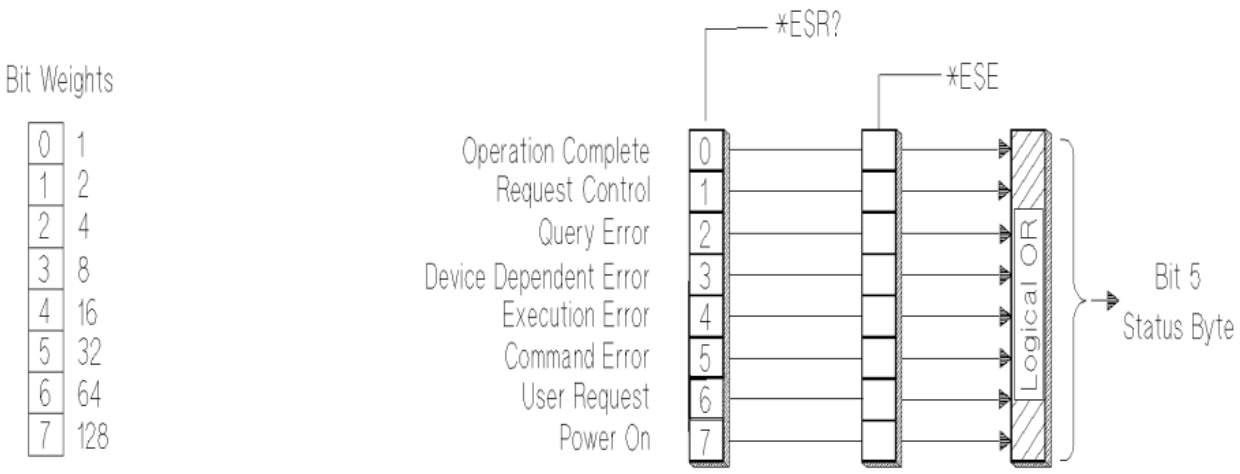


Ilustración 6: Registro Standard Event Summary

2.I.3 Modos de disparo

Para entender bien el uso que realiza el proyecto de los analizadores, deberemos explicar y comprender los modos de disparo principales que están implementados. El Servidor VISA utiliza estos modos, que están relacionados con los modos de disparos de los analizadores, pero los analizadores funcionarán con modo que el servidor crea necesario.

2.I.3.i Modo hold

Mantiene la traza actual por pantalla hasta que se inicie una nueva orden de barrido.

2.I.3.ii Modo Single

Inicia un barrido en cuanto se pulsa. Tras obtener una traza, pasa a modo hold.

2.I.3.iii Modo con cambios

El primer modo de disparo que se introdujo en el programa para los analizadores es el que se llamó “con cambios”. En este modo el analizador solo realiza un barrido cuando se ha introducido algún cambio de parámetros que afecte la traza. También se puede pedir una nueva traza al analizador pulsando el botón de “nueva traza”.

Este modo está pensado para utilizar el canal de comunicaciones lo menos posible y no hacer un uso excesivo de él. También evita que el analizador esté midiendo continuamente, cuando no es necesario hacerlo. Sería el modo ideal para una conexión por Internet.

Al principio del proyecto este modo se esperaba a tener la traza, quedando la página web en

estado de actualizando, para enviar los datos. Con la inclusión de las rutinas de atención a interrupciones, se cambió y el analizador avisa al servidor VISA cuando la traza está realizada.

2.I.3.iv Modo continuo

Es el modo de funcionamiento continuo normal que se utiliza en los analizadores. El analizador realiza un barrido tras otro de forma continua. Para realizar con el programa este modo, se ha decidido emplear interrupciones. Genéricamente las interrupciones se utilizan para que el analizador avise de cuando ha ocurrido algo al programa principal y así mientras tanto, el analizador y el servidor puedan realizar otra tareas. Ya que mientras se está tomando la traza, el analizador es capaz de recibir ordenes y procesarlas.

En nuestro caso, el analizador nos ha de avisar cuando tiene disponible una nueva traza. Para que así el servidor VISA pueda pedirla, recibirla, almacenarla y enviarla al servidor web en caso de que este la necesite.

Como ya veremos más adelante, el tiempo de refresco de traza en el navegador web está fijado en 1 segundo. Sería un gasto de recursos innecesario que en ese intervalo de tiempo el analizador realizara más de una traza, ya que nadie la aprovecharía. Durante el modo continuo se utiliza como tiempo Sweep Time un valor mínimo de 0.7 segundos. Dejando un margen de 0.3 segundos, para que el programa puede preguntar y obtener la traza.

En todos los modos de disparo, el servidor no queda bloqueado esperando a que el analizador tenga lista la traza.

2.II Servidor VISA

Este programa es el encargado de realizar la gestión de los analizadores, para que luego el servidor web pueda disponer de los datos necesarios para presentárselos al usuario. Para que pueda realizar esta tarea debe de estar instalado en una máquina que disponga de conexión con los analizadores. Su función es la de hacer de puente entre los analizadores y el servidor web, evitando que el servidor web tenga acceso directo a los analizadores. Este hecho también permite más versatilidad ya que el Servidor Web puede alojar otras páginas web en otra máquina distinta del Servidor VISA si así se requiere.

La primera decisión a tomar es la forma de comunicación con los analizadores. Con el analizador 8714ET se podría implementar un comunicación por TCP/IP directamente. Aunque como hemos comentado anteriormente, para utilizar interrupciones, hemos de utilizar GPIB. También aunque hiciéramos la conexión por TCP, anteriormente hemos visto que las interrupciones se han de manejar obligatoriamente por GPIB.

2.II.1 Librerías VISA

Para implementar esta comunicación se decidió utilizar VISA (Virtual Instrument Software Architecture). VISA es un estándar para la configuración, programación y detección de errores en

sistemas de instrumentación que utilicen como medio de comunicación GPIB, VXI, PXI, Serial, Ethernet, y/o USB. Los fabricantes utilizan como base este estándar para ofrecer distintas soluciones de programación tales como Agilent-VEE o LabVIEW. También ofrecen soluciones en forma de librerías para utilizar en distintos lenguajes de programación. Para este proyecto se decidió utilizar las librerías bajo C. Para así poder obtener mayor rendimiento y control del servidor.

Una vez ya se tiene definido el lenguaje de programación del Servidor, falta definir el sistema operativo que utilizará. Esta decisión viene intrínsecamente ligada a la de utilizar las librerías VISA. El principal problema que nos encontramos, es que de momento, ninguna de las implementaciones de las librerías VISA existentes es código abierto. Aunque existe un proyecto en desarrollo para tal fin. La consecuencia de este hecho, es que no existan casi alternativas para Linux y todas las soluciones pasen por utilizar Windows.

Esto conlleva a que la implementación actual de este proyecto utilice Windows. Aunque en un futuro se podría migrar a Linux.

Otra de las decisiones fue escoger la implementación de las librerías. Ya que cada fabricante posee las suyas propias. En este caso se optó por utilizar la implementación de Agilent, puesto que, los analizadores y el interfaz GPIB-USB son de dicho fabricante.

2.II.2 Multitarea

El servidor tendrá que hacer frente a la conexión de varios analizadores a la vez. También a la de varias conexiones por parte del servidor web. Por tanto, se plantea la necesidad de que el servidor sea multitarea. Se podría pensar la utilización de interrupciones, atendiendo con una interrupción a cada petición del servidor. Pero sería una mala opción, ya que no permitiría la ejecución de código que atendiera a varios analizadores y/o clientes a la vez.

Como ya hemos visto antes, el Servidor VISA funciona bajo Windows. Con lo cual para solucionar el problema de procesar varias conexiones a la vez solo existe el recurso de utilizar varios hilos de ejecución a la vez.

2.II.2.i Hilos de ejecución

Para explicar que es un hilo de ejecución, primero se ha de entender que es un proceso y cuales son sus atributos. Un proceso es una ejecución de código independiente en un procesador con las siguientes características:

- Código: es el conjunto de instrucciones que debe de ejecutar el procesador.
- Memoria: es el espacio en memoria asignada a dicho proceso, donde se almacenan las variables que pueda tener.
- Recursos del sistema operativo.

- Privilegios: operaciones que están permitidas que realice ese proceso.

- Registros de estado: contienen el estado del procesador, como por ejemplo, siguiente instrucción a ejecutar (contador de programa), dirección de memoria asignada...

- Número de proceso.

Un proceso puede tener uno o varios hilos de ejecución, dependiendo del sistema operativo en el que se encuentre. Definiendo hilo como el conjunto de instrucciones a realizar con ciertas características.

Cada hilo tiene sus propios: contador de programa, pila de ejecución y estado de la CPU. Comparte con los demás hilos: espacio en memoria, ficheros abiertos, privilegios, etc... Al poder compartir espacio en memoria permite bajo ciertas condiciones, que si modificamos un valor global de memoria, este pueda ser leído desde otros hilos inmediatamente.

Que un programa tenga varios hilos no quiere decir que se realicen a la vez en el tiempo. Esto es debido a que un procesador, solo puede realizar una instrucción a la vez, aunque de cara al usuario dé la sensación de simultaneidad en el tiempo. La ejecución de hilos distintos de un programa, se ejecuta, como si fueran distintos procesos en un sistema monoprocesador.

Es de la siguiente forma:

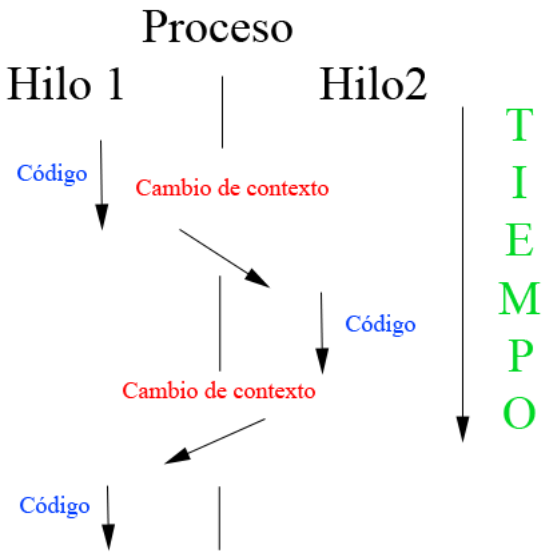


Ilustración 7: Esquema de ejecución de distintos hilos.

El cambio de contexto no se puede predecir cuando va a llegar. Si un hilo se queda esperando un señal de entrada/salida, o recibe un señal de espera, se producirá un cambio de contexto, para aprovechar el tiempo y que pueda ser utilizado por otros hilos. De la misma forma que en un

sistema operativo se reparte la carga entre procesos, en función de las necesidades de cada proceso.

El hecho de que en cualquier momento se pueda pasar el control a otro proceso, ha de estar presente siempre a la hora de acceder a las variables compartidas entre varios hilos. Por lo tanto, hemos de implementar algún sistema, que evite que 2 hilos puedan acceder simultáneamente a la misma variable provocando errores. Para el proyecto se ha escogido la opción de implementar un sistema de semáforos con bloqueo, pero esta elección depende de las librerías utilizadas para implementar el ejecución multihilo en el proyecto.

Dependiendo del lenguaje de programación utilizado la implementación multihilo viene ya integrada con el propio lenguaje (Java, .NET, Delphi). O se ha de implementar dependiendo de librerías exteriores, como en C o C++, que es el caso que nos ocupa. Estas librerías dependen del sistema operativo sobre el que se esté ejecutando el programa.

Por si en un futuro se quisiera migrar el Servidor VISA, a Linux, y así facilitar la transición de código de un sistema a otro, se decidió utilizar el estándar "POSIX Threads", presentes en ambos sistemas operativos. También ese API (Application programming interface) fue escogido porque permitía todas las opciones requeridas para el funcionamiento del servidor.

2.II.2.ii Pthreads

El estándar, POSIX.1c, Threads extensions (IEEE Std 1003.1c-1995), define un API para la creación y utilización de hilos. Está definido como un conjunto de funciones y constantes a implementar en C a través de la cabecera "pthread.h".

La implementación del estándar escogida para el proyecto es el conjunto de librerías Pthreads-w32. Es software libre bajo licencia LGPL (GNU Lesser General Public License).

2.II.2.iii Otras librerías

También se han utilizado otras librerías de uso general. Para la entrada/salida por pantalla de mensajes, manejo de cadenas y medida de tiempos. Así como la librería del sistema Windows "windows.h" para dejar suspendido un hilo por un tiempo cuando sea necesario.

2.II.3 Funcionamiento

Ahora que ya se han descrito los medios utilizados en el Servidor VISA, se puede explicar su funcionamiento.

Cuando se ejecuta el Servidor VISA, primero se realiza un proceso de inicialización.

La inicialización del servidor se realiza siguiendo el siguiente esquema:

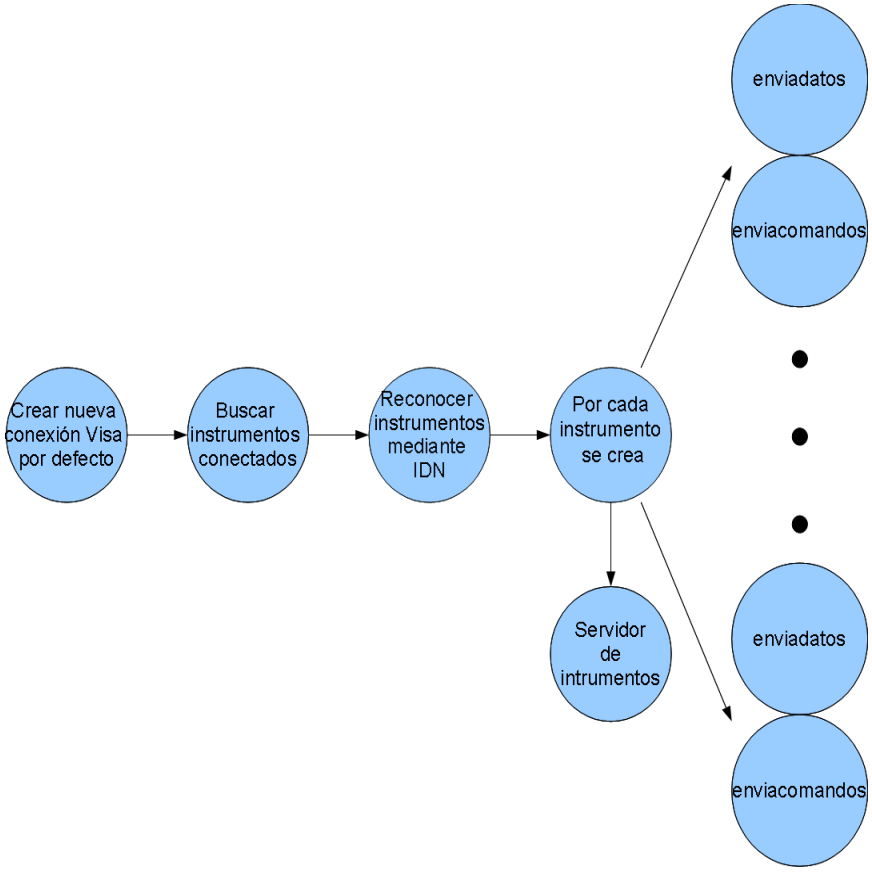


Ilustración 8: Esquema de inicialización.

2.II.3.i Inicialización del servidor

El primer 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 rastrear todos los posibles instrumentos que hay conectados a él. Una vez están 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.

Se almacenará en el vector de cadenas "nombres".

También se guardará un número de tipo entero que a nivel de código se llama "tipo" que indica el modelo de analizador conectado, siendo:

1. 8714B
2. 8714ET
3. Agilent E5062A

4. Analizador/instrumento desconocido

Las opciones 3 y 4 no están implementadas en el proyecto y se dejan para lineas futuras. También se podría ampliar este código con nuevos analizadores. El valor de este entero lo comprenden y lo utilizan el Servidor VISA y el Servidor Web.

El siguiente paso es crear un vector de semáforos, que regulará el acceso a las variables compartidas de cada analizador. 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.

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 último paso es inicializar las conexiones y variables adecuadas para cada tipo de analizador. También creamos 2 hilos de ejecución, enviadatos y enviacomando.

2.II.3.ii La clase analizador

Como el lenguaje utilizado por el proyecto es C++, se utiliza las propiedades del lenguaje orientado a objetos para definir 3 objetos de la siguiente forma:

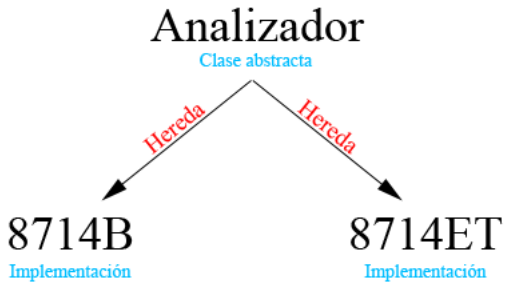


Ilustración 9: Herencia en la clase Analizador.

2.II Servidor VISA

La clase analizador es la clase padre de la que heredarán todos los analizadores. Analizador es una idea, una clase abstracta, una definición a nivel de código de las características (variables y métodos) que debe de tener un analizador y las cuales están obligadas a implementar sus hijas. Con lo cual no puede existir ningún objeto analizador, pero si que puede recibir variables de sus hijas. Esto es utilizado en el proyecto dentro del vector de analizadores, y así, poder acceder a cualquier analizador y comunicarse con él, sin necesidad de saber que tipo es. Esto también facilita el añadir nuevos analizadores al proyecto, ya que solo hay que crear un clase hija de analizador, que realice las funciones exigidas. Sin tener que cambiar el código principal del programa, más allá de los analizadores reconocidos.

Veamos ahora los métodos de la clase analizador, la mayoría de los métodos incluidos en la clases, se relacionan con los menús incluidos en la página web. Indicando las opciones escogidas en dicho menú:

Nombre del menú	Nombre del método	Descripción
Measurement	cambiarparametros	Se utiliza para cambiar el parámetro de medida. Ya sea en reflexión (S11) o en transmisión (S12).
Format	cambiarformato	Cambia el formato en el que está representada la traza.
Display	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 la traza en la memoria del analizador, representar a esta o una combinación con la traza actual mediante una operación matemática.
Average	promediado	Controla el promediado y filtro de resolución del analizador.
Calibration	calibracion	Es el menú que permite calibrar el analizador y sus opciones.
Stimilus	frecuencias	Se utiliza para cambiar las frecuencias.
Sweep Setup	pendiente	Permite cambiar las opciones bajo las que mide la traza (número de puntos, tiempo de traza...).
Trigger	disparo	Cambia las opciones de disparo del analizador

Actualizar traza, guardar datos	obtenerdatos	Opciones especiales implementadas para el proyecto no incluidas en el analizador.

El método preguntar es el único método que implementa la clase Analizador. Con él, se puede preguntar a un analizador un valor, incluyendo el número de intentos fallidos que permita cuando realizamos esa pregunta. En caso de que ese número se exceda se procederá a realizar una reconexión con el analizador. Esta reconexión se realiza a través del método reconectar que es distinto en cada analizador.

También se han creado métodos para el control de interrupciones. Estos métodos se han de encargar de habilitar y deshabilitar las interrupciones en el analizador. La implementación que se haga de dichos métodos, depende del analizador, y habilitará lo necesario en cada uno para obtener una interrupción en cada nueva traza. Desde fuera para el programa, no tiene que preocuparse de lo que hay que hacer, facilitando el añadir nuevos analizadores con distintas formas de interrupción.

El método habilitar traza ordena al analizador el disparo de una traza nueva. Y obtener traza se utiliza para preguntar la traza actual.

Cada analizador tiene sus propias características, para poder implementar pequeños cambios en el funcionamiento de ambos.

2.II.3.iii 8714B

Este analizador tiene la particularidad de que no tiene ningún comando para averiguar si está calibrado y su estado. Como esta información se considera vital para las prácticas, se ha creado un sistema que permite a través del servidor VISA conocer esta circunstancia. Se basa en que cuando se calibra desde el servidor, se almacena la configuración del analizador, para más tarde cuando queramos saber su estado, compararlos y dar un resultado.

2.II.3.iv 8714ET

Como ya se explicó anteriormente, las interrupciones en este analizador solo se pueden detectar por GPIB. Más adelante se realizará un estudio de tiempos y de los problemas que esta conexión crea. Para solventar este problema se decidió por la solución más sencilla y eficaz: el control de interrupciones realizarlo por GPIB, mientras que, el envió de comandos y recepción de datos llevarlo a cabo por TCP/IP.

Cuando se crea un objeto de este tipo, preguntamos por GPIB la IP del analizador en la red y tratamos de crear una conexión IP. Hemos supuesto previamente que el servidor y analizador pueden comunicarse a través de la red.

Iniciando 2 conexiones:

1. La conexión que se almacena en la variable “vi”, que sirve para comunicarse con el analizador y es por TCP/IP.

2. La conexión “viinterrumpir” sobre la cual se activan las interrupciones y es por GPIB.

En la cadena IP, se almacena la IP del servidor que hemos obtenido a través de GPIB.

2.II.3.v Creación del servidor de instrumentos

Una vez se han inicializado todos los analizadores, se crea un servidor al que pueda acceder en un inicio el Servidor Web para poder conocer los analizadores conectados y su tipo. También indica el puerto al que a de acceder el Servidor Web para conectarse con los distintos analizadores cuando lo requiera el usuario.

2.II.3.vi Entrada de una nueva conexión

Cuando se ha acabado el proceso de inicialización tenemos:

- Un vector con todas las variables de tipo analizador, su longitud es igual al número de analizadores detectados.

- Un vector con semáforos para regular el acceso a la variable de cada analizador.

- 2 hilos de ejecución por cada analizador “enviadatos” y enviacomandos”.

- Un hilo de ejecución encargado de comunicar el número de instrumentos, tipo, nombre y dirección.

Cuando llega una petición del Servidor Web, en primera instancia se conecta al hilo-servidor de instrumentos, para averiguar el número de analizadores conectados y sus características. Cuando el Servidor Web, ya tiene que analizador quiere conectarse el usuario, entran en juego los 2 hilos de ejecución de cada analizador:

2.II.3.vii 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.

Este servidor también es el encargado de actualizar los valores de los parámetros del analizador en caso de que hayan sido cambiados, así como de controlar, que el tiempo de traza es mayor que 0.7 segundos para así adaptarlo al ritmo de actualización del Servidor Web.

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 hilos a la variable creada del analizador correspondiente.

Después se procesa y realiza el comando indicado. 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.

Cuando se recibe la interrupción de una traza, lo primero que se realiza, es deshabilitar las interrupciones para evitar recibir una interrupción por parte de otros analizadores. Se debe a razón de que en GPIB las interrupciones son compartidas. Cuando se recibe una interrupción se ha de preguntar a cada instrumento cual de ellos ha activado la linea de interrupción. Para poder actuar unicamente sobre el analizador que ha interrumpido.

Después se procede a preguntar al analizador la traza y a obtenerla.

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.II.3.viii 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.

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.

Una vez se ha visto las características de los analizadores y el Servidor VISA, se puede pasar a detallar la conexión entre ambos y su comportamiento. Así como explicar la selección de dichas conexiones mediante medidas de tiempos.

2.II.4 Control de errores

Servidor VISA lleva implementado control de errores para varias anomalías posibles:

2.II.4.i Apagado o desconexión de un analizador

Si el Servidor detecta que no puede conectarse con un analizador. Este trata de realizar una reconexión con él. En caso de que la reconexión sea positiva se sigue trabajando de manera normal. Esta detección solo se realiza cuando se recibe un comando. A consecuencia de esto, los posibles comandos anteriores recibidos puede que no se realicen. Antes de recibir una traza siempre se comprueba que el analizador está conectado y en caso contrario se realiza una reconexión.

2.II.4.ii Desincronización general

Si el analizador se maneja en modo manual, se realiza un preset, o se apaga y se enciende, puede provocar un error de desincronización con los datos que contiene el servidor, así como, la eliminación de las interrupciones. Para ello de manera manual se ha de realizar un preset en la página web, dicho preset devuelve al analizador al estado óptimo para su funcionamiento con el servidor. Este también se realiza si se reinicia el Servidor VISA.

2.II.4.iii Acceso a comandos de calibración

El menú de calibración está protegido para funcionar únicamente en caso de que haya una calibración en curso. También cuando se realiza una calibración, se comprueba que ésta sea la adecuada para el parámetro actualmente seleccionado.

2.II.4.iv Envío de comandos durante la calibración

De igual forma, cuando se calibra, el resto de comandos están bloqueados y en caso de recibir uno, no se realiza y se redirige a la pagina web de calibración con el estado actual. También se ha implementado un sincronismo adicional entre el Servidor Web y el Servidor VISA para evitar estados de calibración incorrectos.

2.III Comunicación Analizador – Servidor VISA

Para poder entender las repercusiones de esta conexión en el proyecto, primero se debe describir las características de estas:

2.III.1 Conexiones empleadas

2.III.1.i Ethernet

Como ya hemos visto el analizador 8714ET se puede conectar por GPIB o TCP/IP. La conexión IP se realiza mediante TCP al puerto 5025. La conexión se abre al inicio del proyecto, cuando se abre sesión con las librerías VISA. Cuando se quiere transmitir algo, se envía un paquete TCP/IP que incluye en el campo de datos las ordenes SCPI que queremos realizar. De la misma manera se recibirá en el campo de datos la respuesta a dichas ordenes si la hubiera. Un paquete TCP sobre ethernet tiene la siguiente forma:

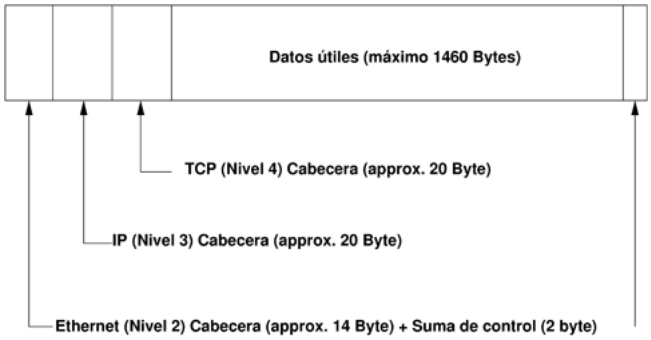


Ilustración 10: Paquete ethernet.

La respuesta del analizador se envía en la parte de datos. Se envía en código ASCII sin ningún control de errores, ya que TCP contiene seguridad para poder comprobar que lo que se ha recibido es correcto y sino pedir su reenvío. Aunque la protección de la integridad de datos de TCP es muy débil, es más que suficiente ya que nos el servidor y el analizador suelen encontrarse en red local. Los 1460 bytes del tamaño puede verse reducido según las condiciones de la red, por ejemplo si la conexión se realiza mediante una VPN este valor disminuirá.

La conexión del analizador 8714ET es 10BASE-T. Con el cual tenemos una velocidad máxima de 10 Mb/s, aunque lo conectemos a una red más rápida.

2.III.1.ii GPIB

GPIB es un bus paralelo. Tiene 8 lineas por las que se envía cada bit, 5 líneas de control del bus, 3 lineas para controlar el envío y validación de los datos y 8 lineas de tierra.

Entre las líneas de control del bus se encuentra la linea de petición de interrupciones de la que ya se ha hablado anteriormente.

La validación de un byte y envío del siguiente se realiza mediante 3 lineas de control de envío, realizando un proceso de “handshake”. Estas 3 líneas representan las disponibilidad para recibir traza, la validez de los datos enviados y la confirmación de los datos leídos.

Con estos datos podemos entender los distintos escenarios de tiempos presentados a continuación:

2.III.2 Análisis de tiempos

Como tenemos 2 tipos de conexiones, se deberá de medir los tiempos en ambos escenarios. GPIB, puede utilizar el mismo bus para comunicarse con varios analizadores. Al ser un bus compartido, se producen colisiones y no se puede comunicar el servidor con más de un analizador a la vez. Crea un problema de tiempos, porque el tiempo que tarde en responder y enviar un analizador los datos, es un tiempo que no podemos recibir datos de otros analizadores.

2.III Comunicación Analizador – Servidor VISA

Las colisiones en GPIB generan bastantes problemas de tiempos y de programación. El tiempo para obtener una traza, será el tiempo que estemos esperando porque el bus esté ocupado por otro analizador más el tiempo que tardemos en obtener la traza.

También debemos de considerar el caso en el que la conexión con un analizador se caiga. En este caso tendremos problemas, porque el tiempo máximo de respuesta de las librerías VISA es de 2 segundos. Provocando un bloqueo del sistema durante 2 segundos. De manera adicional crea problemas porque hay que implementar un control de errores mayor para el envío de comandos, realizándose tanto en recepción como en envío.

Este último problema, hizo que se buscara una solución alternativa a conectar los 2 analizadores por GPIB. Porque la limitación de tiempos, aun permitía cierto margen para realizar otras tareas, pero una caída de un analizador provocaba un fallo en cadena que hacía que el servidor dejara de funcionar durante 10 segundos. Añadiendo un bloqueo de 10 segundos cada vez que el usuario iniciará un nuevo intento de comunicación con el analizador caído.

El servidor VISA regula el acceso a cada analizador mediante semáforos. Cada vez que una parte del programa quiere acceder cierra el semáforo si se encuentra abierto y sino se espera a que se habrá para continuar y cerrar el acceso. El acceso se bloque desde 3 partes del código:

- Desde el enviadatos: este se produce cuando accedemos a las variables para leerlas y poder enviarlas al servidor. Como estos valores no tienen nada que ver con la capacidad de respuesta del analizador, se medirán más adelante.

- Desde enviacomandos cuando se envía una orden al analizador: se espera a que éste la procese para poder cambiar el parámetro de configuración asociado a esa orden si lo hubiera. Este valor depende del comando que utilicemos, la conexión empleada y del analizador.

- Desde enviacomandos cuando hay una traza disponible: al ser mediante interrupciones, se debería de bloquear enviacomandos cuando se realiza una interrupción. En lugar de eso, con las librerías VISA, se ejecuta un nuevo hilo que atiende a la rutina de atención a la interrupción. Si nos encontramos en modo continuo, este tiempo se ejecuta cada 0,7 segundos en el peor de los casos. Y mientras se está realizando la traza es tiempo que no podemos atender a las peticiones de enviadatos y enviacomandos. Con lo cual es un tiempo vital, que si es excesivo, puede dar lugar a problemas. Este tiempo depende del analizador, la conexión, empleada y el formato de datos utilizado.

De los 3 bloqueos presentados anteriormente el último es el que debemos de tener en cuenta, medir y optimizar. El segundo afectará, pero no se puede optimizar ya que depende del comando enviado y lo que tarde en respondernos el analizador. No se puede configurar ningún parámetro para que este sea más rápido.

2.III Comunicación Analizador – Servidor VISA

Primero se va a analizar el envío de una traza empleando los diferentes tipos de conexiones. La conexión GPIB se realizará de modo directo y con un único instrumento. El retardo producido si hay más de un instrumento conectado lo veremos más tarde.

Como ya hemos visto, los analizadores, avisan al Servidor VISA de cuando tienen una traza lista. Después se pregunta por esa traza y se recibe según un formato específico.

En GPIB los datos se envían byte a byte. Con un mecanismo mediante distintas lineas de control que asegura que el byte ha sido recibido. Se enviaría desde el servidor la orden de obtener traza y luego contestaría el analizador al que ha sido enviada esa orden.

El esquema que se sigue TCP/IP para enviar una traza es el siguiente.

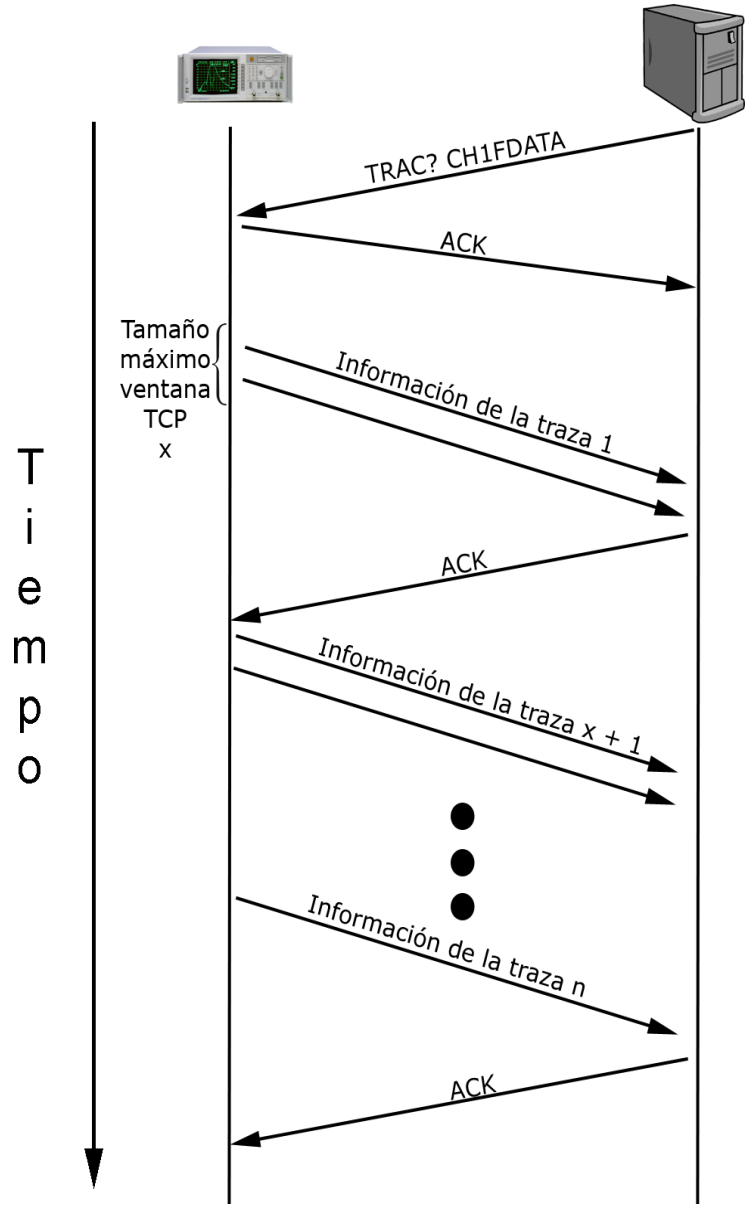


Ilustración 11: Envío por TCP/IP

2.III Comunicación Analizador – Servidor VISA

El analizador tarda un tiempo en responder a la pregunta, porque tiene que preparar los datos. Dependiendo del formato en el que se envíen los datos (32 bits, 64 bits, ASCII) y el formato de la traza (logarítmico, carta de Smith, lineal...), este tiempo de respuesta será distinto. Este tiempo no se puede medir con las utilidades que nos proporciona Agilent. Así que las comparaciones de velocidad entre las 2 conexiones, las debemos realizar en función de los puntos y con el mismo formato de envió y de traza.

El número de bytes que se envían con cada formato y para cada número de puntos es:

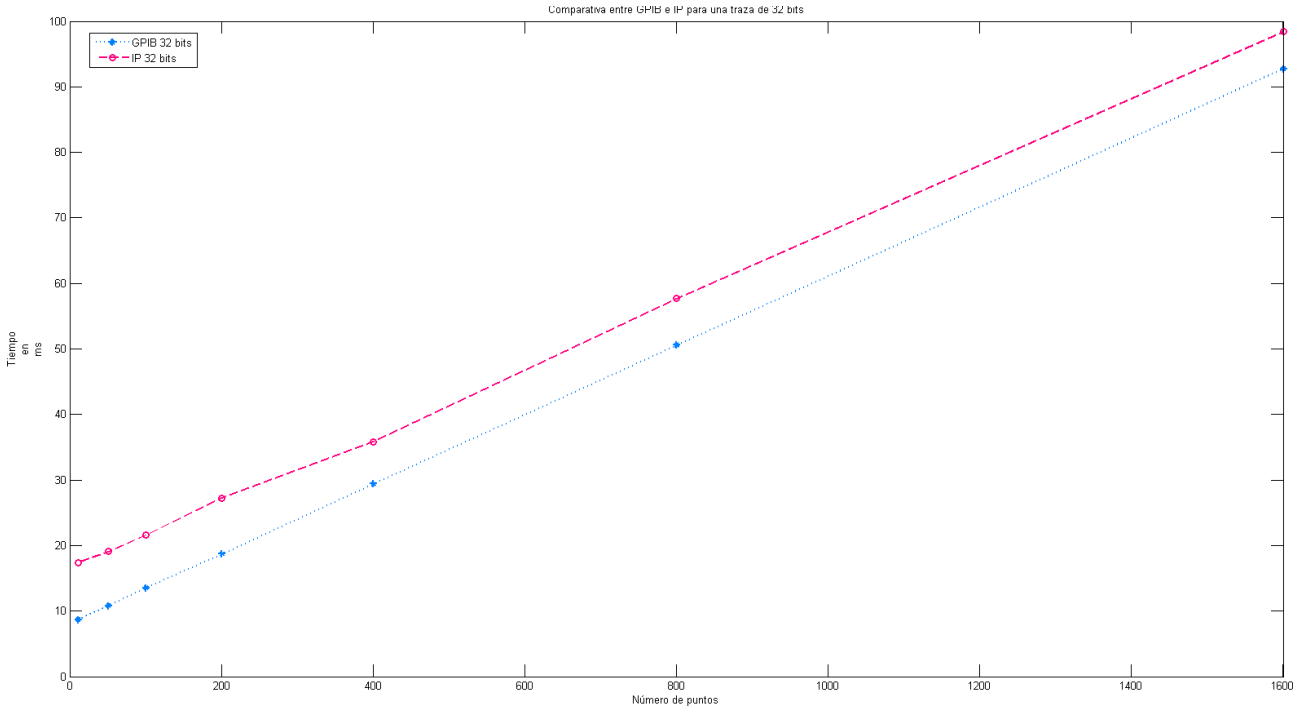
Nº de puntos	32 bits	64 bits	ASCII
11	49	93	154
51	210	414	714
101	410	814	1414
201	810	1615	2814
401	1611	3215	5614
801	3211	6415	11214
1601	6411	12816	22414

La primera comparativa se ha hecho con datos de la traza en formato logarítmico para 3 formatos de envío: 32 bits, 64 bits y ASCII.

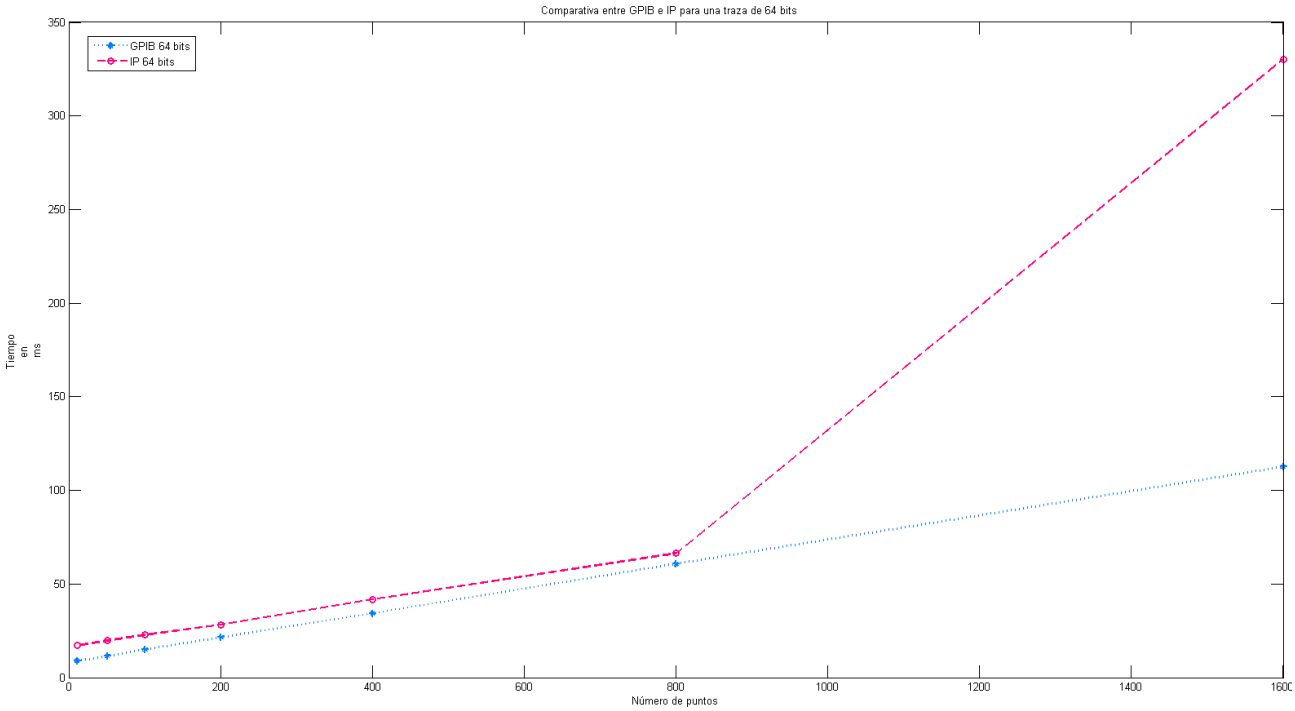
Como podemos ver en las gráfica 1 y gráfica 2, para trazas de 32 bits y 64 bits GPIB e IP son casi igual de rápidos. Esto ocurre porque la validación de los datos en GPIB se realiza en paralelo, mientras que en TCP/IP es en serie. Añadiendo un tiempo extra. También juegan un papel importante las cabeceras TCP/IP, sobre todo para pocos puntos, donde su tamaño es del mismo orden que los datos enviados. Y ambos tienen velocidades de envío parecidas: 8 Mb/s para GPIB y 10 Mb/s para ethernet.

También se pueden observar, que el envío de IP en 64 bits está compuesto por varias rectas. Este cambio de pendientes se debe al tamaño de la ventana de transmisión TCP/IP, cuando superamos el tamaño de la ventana de transmisión se producen 2 tandas de envió de paquetes. Aumentando el tiempo de envió esperando al reconocimiento de que se ha recibido el primer paquete como se indica en el gráfico explicativo de envío sobre TCP/IP.

2.III Comunicación Analizador – Servidor VISA

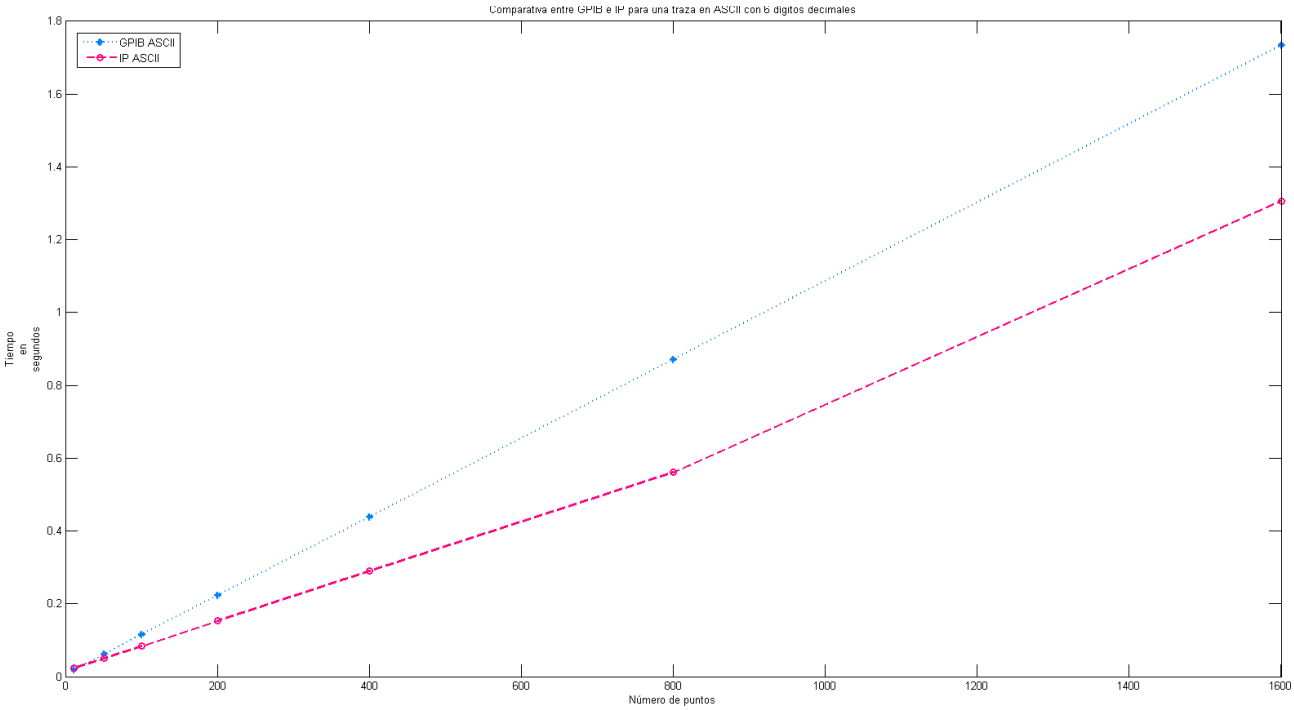


Gráfica 1: Comparativa entre GPIB e IP para una traza de 32 bits.



Gráfica 2: Comparativa entre GPIB e IP para una traza de 64 bits.

2.III Comunicación Analizador – Servidor VISA



Grafica 3: Comparativa entre GPIB e IP para una traza en ASCII con 6 dígitos decimales.

Mientras que para envío de trazas en ASCII, como se puede ver en la gráfica 3 es más rápido IP. Esto se debe a la forma en la que envía los datos el analizador:

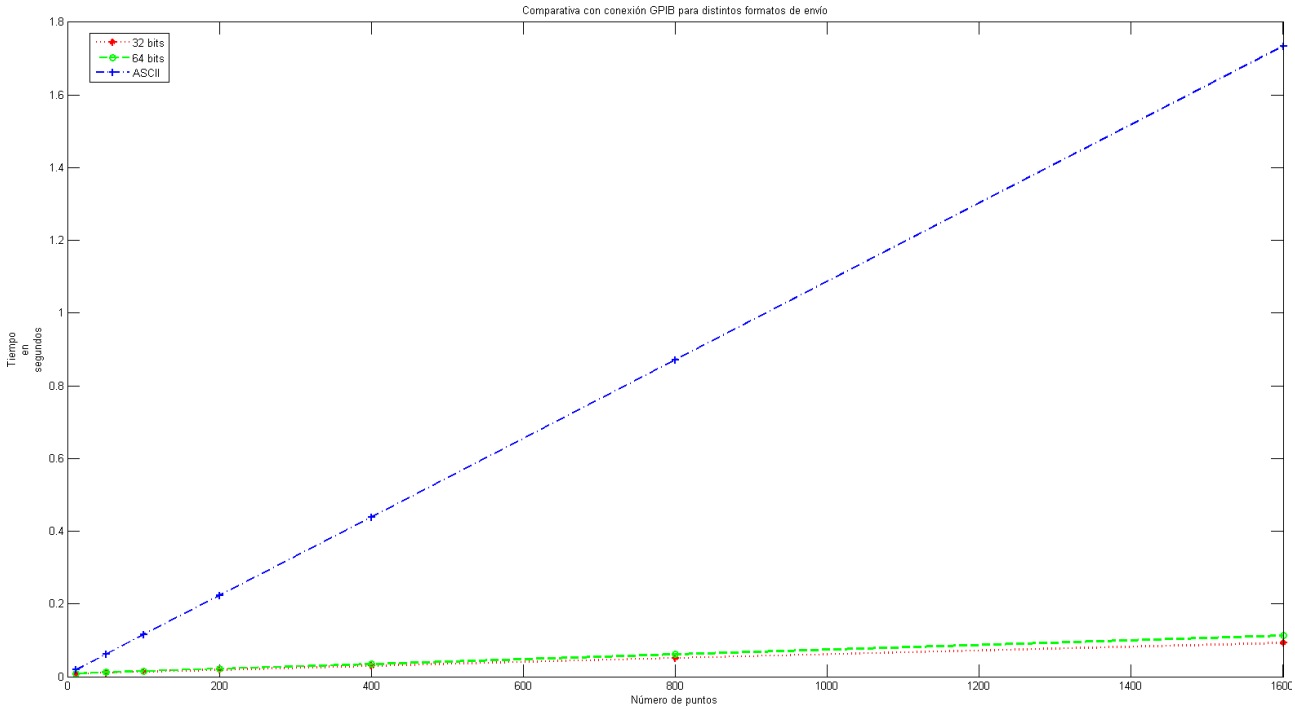
- Antes se ha hablado del tiempo que tarda en formatear los datos el analizador para enviarlos. En el caso de GPIB no envía los datos hasta que estos no han sido almacenados todos los puntos en el formato correcto. En TCP/IP como se validan los datos paquete a paquete. El analizador envía los 2 primeros paquetes cuando tiene el número de puntos con el formato necesario para enviarlos.

Para enviar los datos en formato ASCII, el analizador tarda la mayor parte del envío. Como podemos ver en las gráficas, en GPIB para enviar 1601 puntos con 64 bits se tarda 112 milisegundos. Mientras que para un tamaño parecido de datos en ASCII se tardan 871 ms. Suponiendo muy pequeño el tiempo para formatear los datos en 64 bits, el tiempo de formateo de ASCII sería de 759 ms aproximadamente. Aunque no se puede medir exactamente.

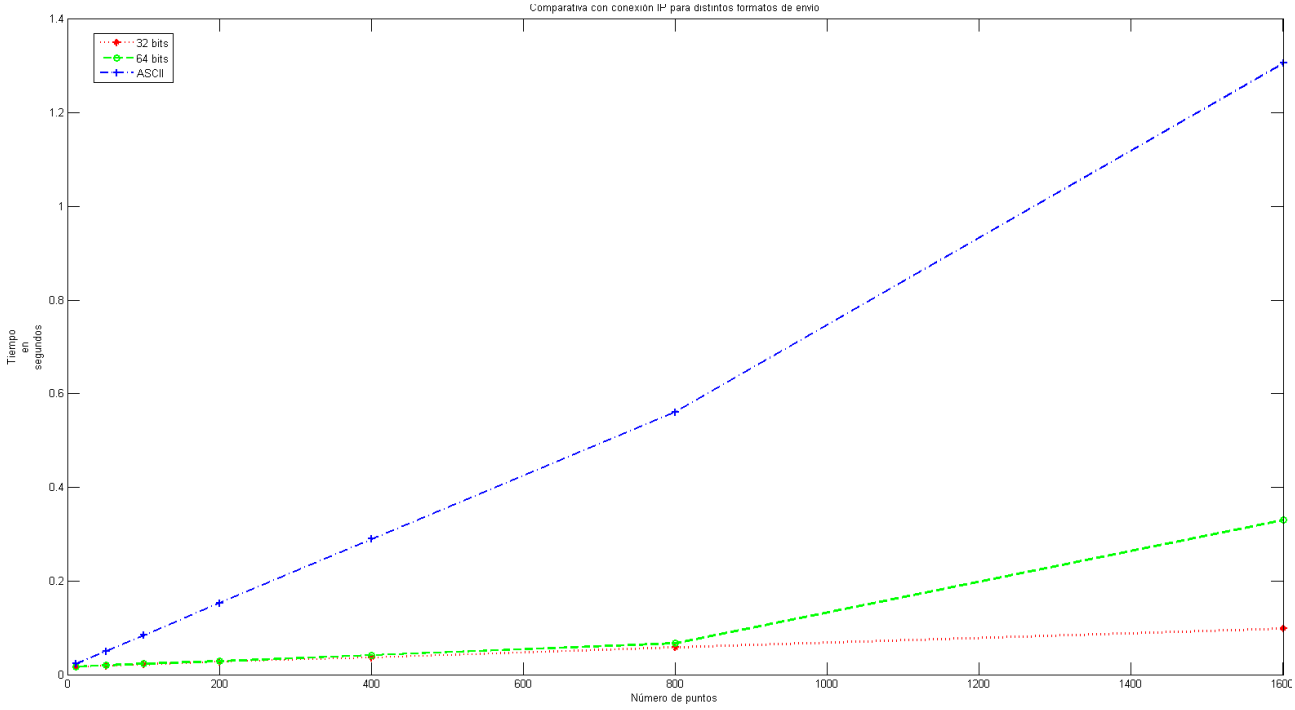
En esos 759 ms en GPIB no ha enviado nada, mientras que en TCP/IP se habrán podido enviar la mayoría de datos, quedando un paquete con los últimos datos por enviar. La misma comprobación se puede hacer para el resto de datos obteniendo resultados parecidos.

Ahora se compararan los valores anteriores, pero incluyendo en la misma gráfica los diferentes formatos de envío para la misma conexión. En estos datos ya se incluye el tiempo que tarda en preparar la traza el analizador; ya que lo que realmente nos interesa, es el tiempo que permanece el servidor bloqueado por el acceso a las variables compartidas. Y con la programación actual del proyecto, este tiempo está incluido.

2.III Comunicación Analizador – Servidor VISA



Grafica 4: Comparativa con conexión GPIB para distintos formatos de envío.



Grafica 5: Comparativa con conexión IP para distintos formatos de envío.

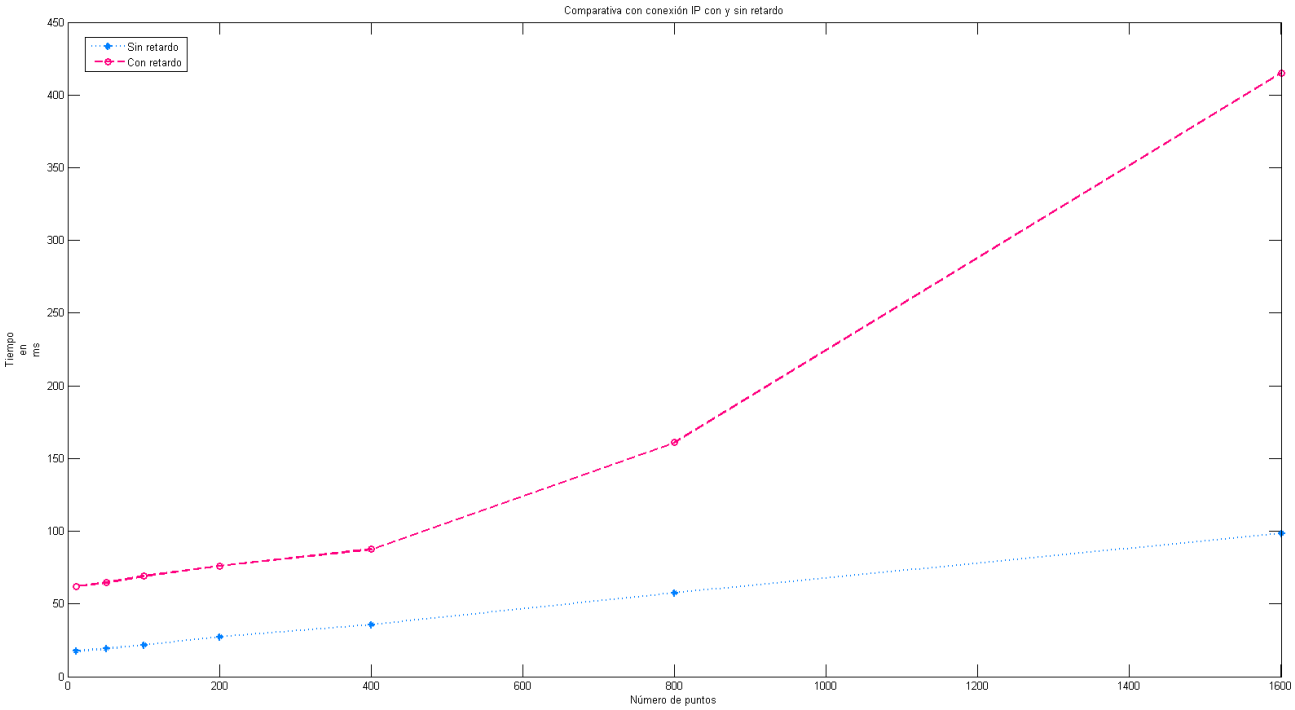
2.III Comunicación Analizador – Servidor VISA

Como se puede ver en las gráficas 4 y 5. El formato de trazas más rápida, se puede ver que el formato de envío de traza es el de 32 bits sino requerimos tanta precisión. Como los datos utilizados en prácticas, no la requieren, se decidió utilizar este formato.

La principal diferencia entre GPIB e IP es la implementación. IP es barato, además de tener la ventaja de poderse utilizar estas redes , que generalmente hoy en día están muy extendidas. También se puede acceder de forma remota desde cualquier parte de Internet, si el analizador está conectado.

Aunque el analizador 8714ET no dispone de interrupciones por TPC/IP, se van a presentar medidas suponiendo un acceso remoto al analizador mediante distintos escenarios. Esta configuración crea un acceso más lejano al analizador, del que permite el cable GPIB. Esto se realiza porque se podría implementar un método de interrupciones alternativo al realizado, pudiendo estar el Servidor VISA en un lugar distinto del analizador. Así veremos las repercusiones que esto podría tener sobre los tiempos de acceso a las trazas y datos del analizador.

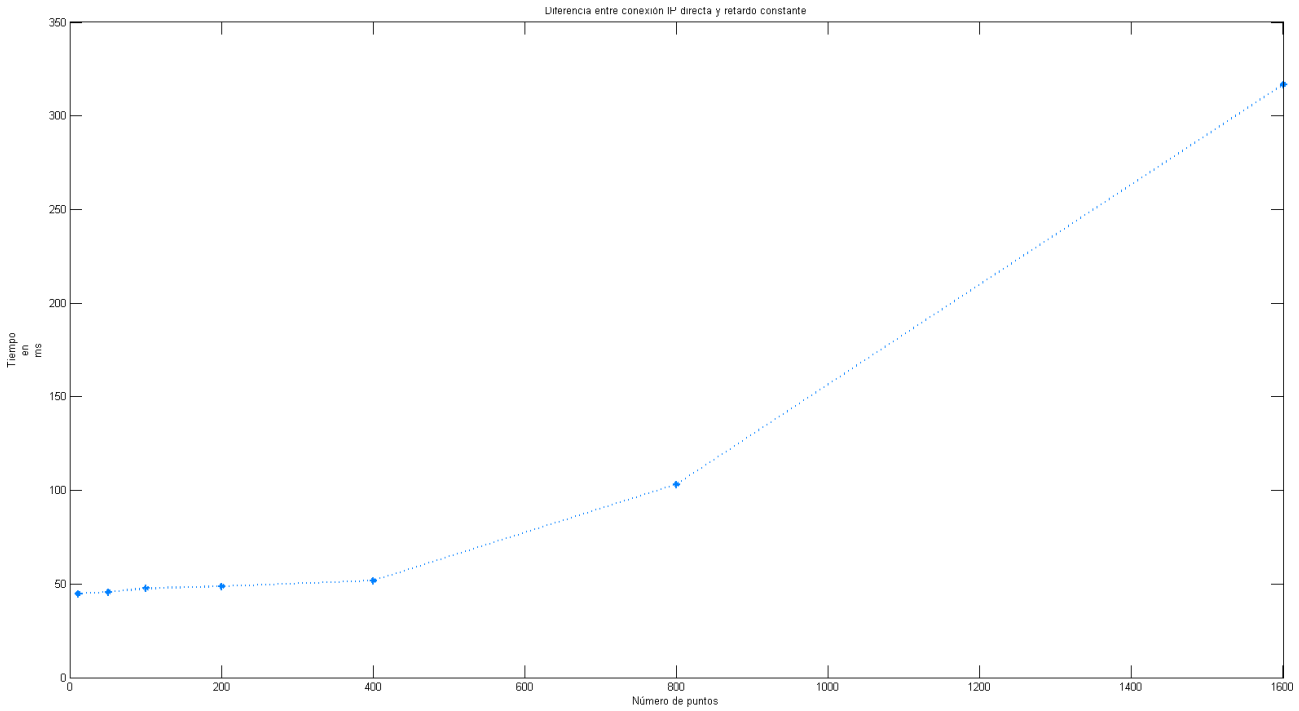
En el primer caso, mostrado en la gráfica 6 nos encontramos con una red con retardo constante. Los datos han sido medidos a través de una VPN con la universidad desde una conexión a Internet de 3 Mb/s. El ping medio entre el analizador y el servidor que pide la traza es de 45,584 ms, con una desviación típica de 0,3573 ms.



Grafica 6: Comparativa con conexión IP con y sin retardo.

Para poder ver el efecto del retardo, se han restado ambas gráficas:

2.III Comunicación Analizador – Servidor VISA



Grafica 7: Diferencia entre conexión IP directa y retardo constante.

Podemos ver 3 tramos claramente diferenciados. De 11 a 401 puntos, de 401 a 801 y de 1601. Como estamos haciendo una VPN, el tamaño es de los datos en 1389 bytes. Si miramos de datos enviados, cada tramo se corresponde con una tanda de paquetes TCP/IP. Aumentando el tiempo en el retardo que hay entre los 2 extremos.

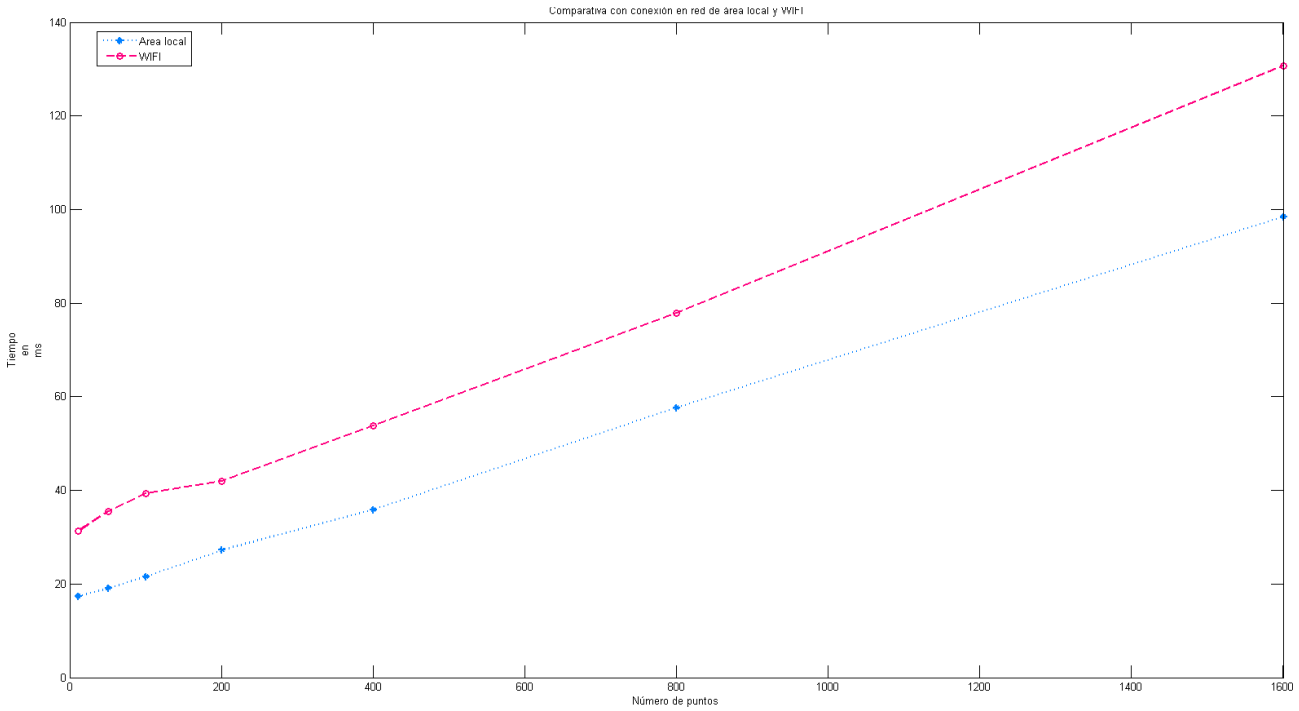
En el tramo de 11 a 401 el valor es exactamente el mismo que el ping entre ambos extremos. La pequeña pendiente que hay en este tramo, se debe a la diferencia de velocidad entre ambas conexiones 3 Mb/s frente a 10 Mb/s.

En el segundo escenario, se ha conectado el servidor que se comunica con el analizador a través de la red WIFI de la universidad, con la señal que llega desde el laboratorio. Como se muestra en la gráfica 8, podemos ver un efecto producido por el retardo parecido al anterior. Que en este caso tiene un valor medio de 64.7544 ms.

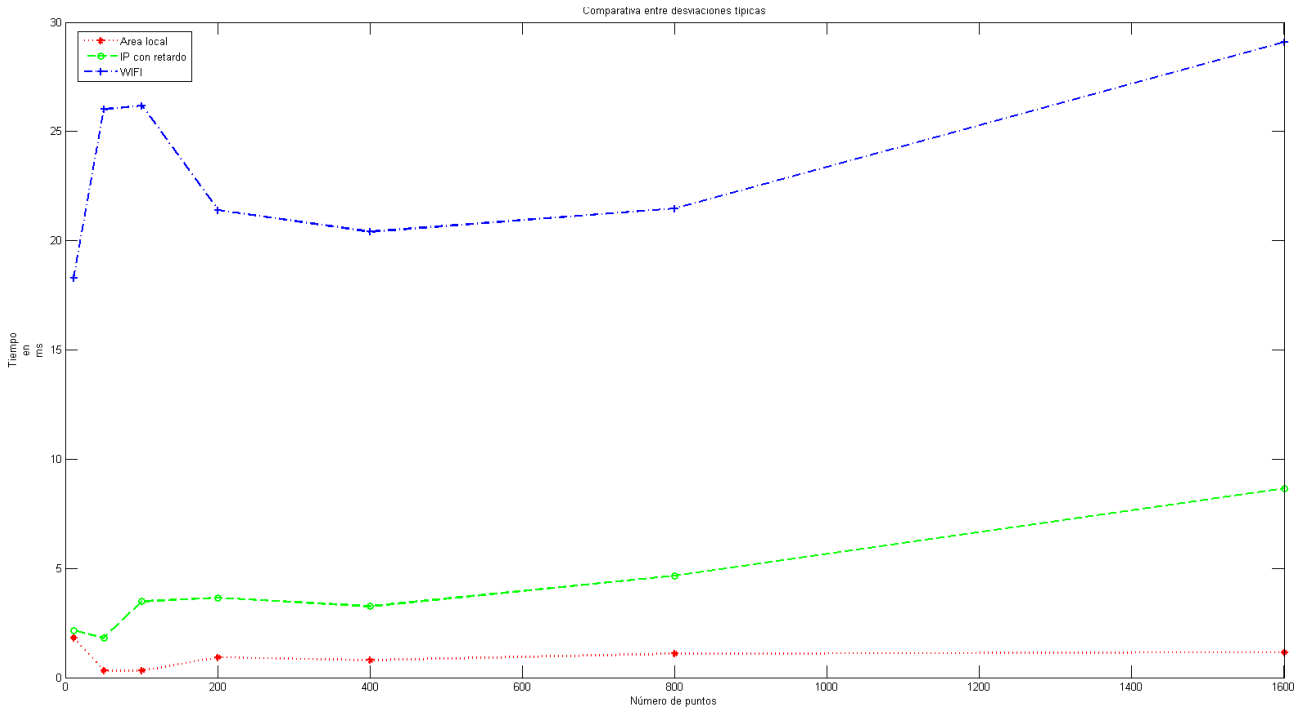
La diferencia con el caso anterior, es cuando observamos la desviación típica de las medidas mostrada en la gráfica 9. En este caso la variación de los datos sobre la media es mayor. Esto es debido a que WIFI es un medio compartido. Las medidas están realizadas en una hora de poco tráfico. En otros momentos se pueden llegar a producir retardos de 1 segundo entre el envío. Esto podría provocar que el usuario no percibiera el refresco de los datos cada segundo en su navegador. Al margen de esto el sistema no daría ningún problema.

El último escenario medido es la repercusión que tiene en los tiempos el tener un analizador más conectado al bus GPIB. Mostrado en la gráfica 10.

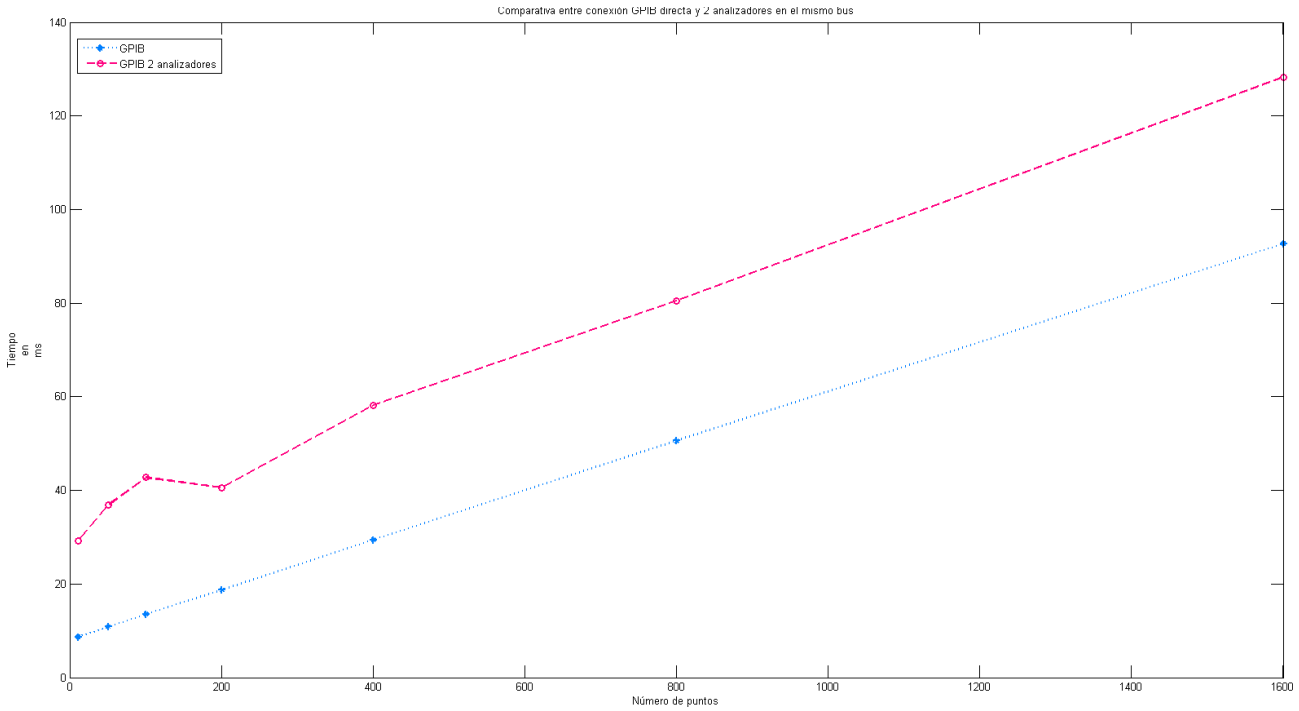
2.III Comunicación Analizador – Servidor VISA



Grafica 8: Comparativa con conexión en red de área local y WIFI.



Grafica 9: Comparativa entre desviaciones típicas.



Grafica 10: Comparativa entre conexión GPIB directa y 2 analizadores en el mismo bus

Como podemos ver en esta última gráfica los tiempos aumentan de manera casi constante. Ya que el otro analizador tiene también que intervenir en la recepción de datos, provocando una pérdida de velocidad. Estas restricciones de velocidad solo ocurrirán con el analizador B, ya que el ET los datos de la traza se obtienen por TCP/IP. En ningún caso llegaría a sobrepasar los 0.3 segundos de guarda que se han incluido para obtener la traza.

Como conclusión a todas estas medidas podemos obtener que el envío optimo es 32 bits conectando directamente el analizador al servidor. Y si alguna circunstancia no permite este hecho, tendremos que sumar un retardo a los tiempos de envío, que mientras no sobrepase el límite de 0,3 segundos impuesto, no tendrá ninguna consecuencia.

2.IV Servidor Web

Esta parte del proyecto es la que se accede desde el exterior. Su cometido es recibir las ordenes que quiera ejecutar el usuario y representar lo datos recibidos por el analizador. Puede estar funcionando en un ordenador aparte o en el mismo que el Servidor VISA, ya que la comunicación entre ambos se realiza mediante TCP/IP.

Como es una página web utiliza HTML para la parte de la comunicación con el navegador web. Para recibir datos, conectarse con el Servidor VISA y cambiar la página web en función de las acciones del usuario se utiliza el lenguaje de programación PHP. También se envía código javascript para que el navegador lo interprete y represente los datos de la traza gráficamente y realice actualizaciones de ésta, periódicas si así lo requiere.

En la implementación realizada en el proyecto se ha utilizado como Servidor Web lighttpd utilizando como sistema operativo Ubuntu Server.

2.IV.1 Lighttpd

Para el proyecto se requería un servidor web, que fuera rápido y consumiera el mínimo de recursos posibles. Estos requisitos se deben a que se planteaba que era deseable que tanto el Servidor Web, como el Servidor VISA, se encontraran funcionando bajo el mismo ordenador. También al deseo de que el proyecto entero, consuma los mínimos recursos posibles para poder ser utilizado sin limitaciones.

De entre el amplio abanico existente de Servidores Web, los que más se ajustaban a las características deseadas eran los llamados “servidores ligeros”. Servidores pensados para webs con gran volumen de carga o que requieran una velocidad de respuesta rápida. De este tipo de servidores web se escogió “lighttpd”.

Lighttpd es un servidor rápido y seguro. Es software libre y su licencia permite ser utilizado en proyecto sin problemas. La licencia bajo la que se distribuye es BSD modificada.

Tiene soporte para varios lenguajes de programación entre los que se encuentra PHP, teniendo en éste mejoras especificas para mejorar su funcionamiento. También permite utilizar cifrado SSL y métodos de autenticación. Cumpliendo con las características requeridas y añadiendo nuevas para lineas futuras del proyecto como es la implementanción seguridad.

Ha sido utilizado por páginas web como youtube o wikipedia. Actualmente está presente en algunos sitios, muchos de ellos con gran volumen de visitas: http://redmine.lighttpd.net/projects/lighttpd/wiki/PoweredByLighttpd

Esto demuestra que lighttpd es seguro y una buena opción para el proyecto. También si lo comparamos con otros servidores conocidos como apache o thttpd demuestra ser más rápido en la mayoría de pruebas realizadas.

Esto no implica que en un futuro no se pueda utilizar otro servidor web, el código está escrito en PHP y cualquier servidor que soporte PHP podría utilizarlo. Es fue una de las razones para utilizar PHP como lenguaje de programación, la otra fue su versatilidad y variedad de soluciones que presenta ante un problema. El mismo criterio nos llevó a utilizar javascript en la parte del navegador. Así como elementos de HTML5.

2.IV.2 Código de la página web

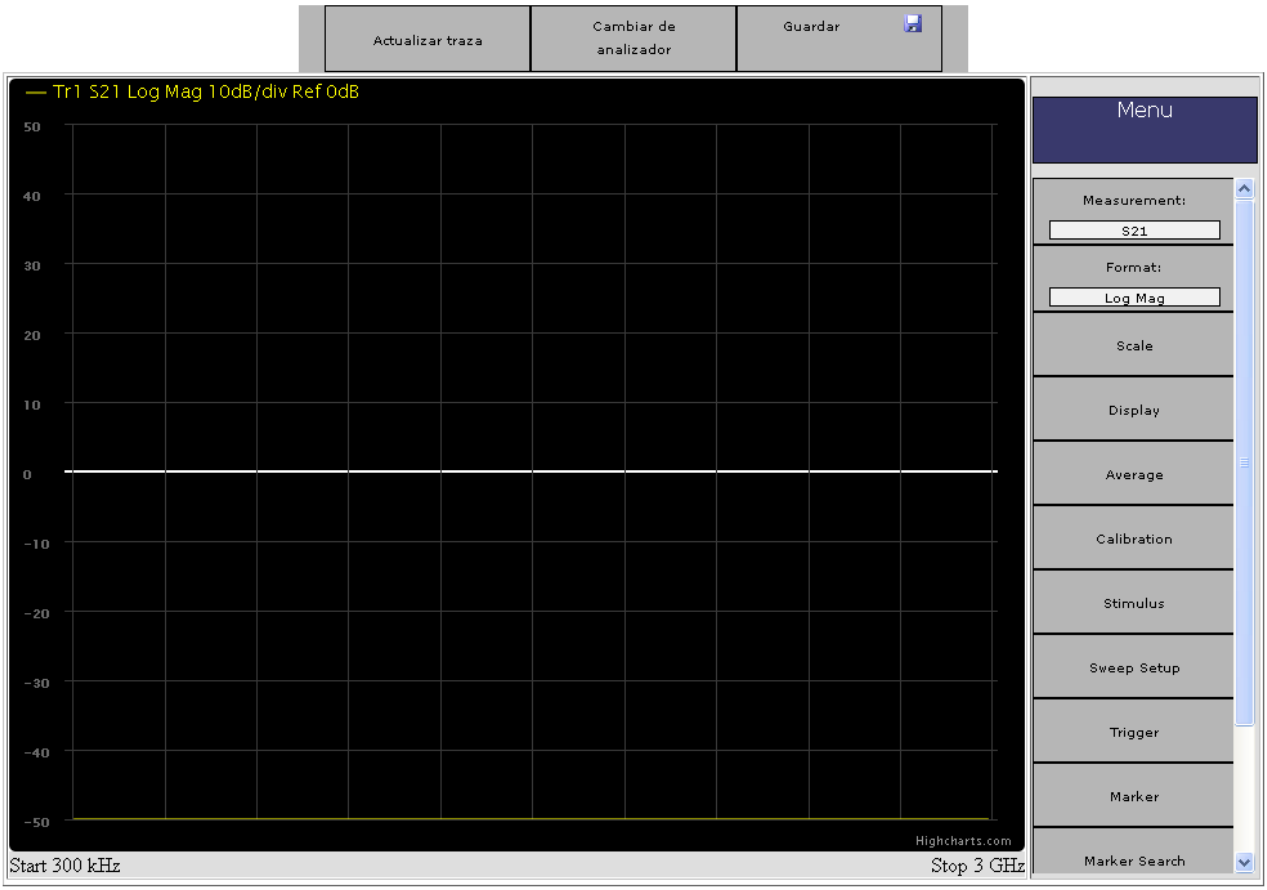
El código utilizado en la página web se ha intentado que sea lo más fiel a los estándares. No cumple con todos por la utilización de una llamada a funciones javascript con parámetros que no está reconocida. Aunque todos los navegadores actuales la entienden.

Para realizar las funcionalidades de la página se ha optado por minimizar el uso de javascript. Si

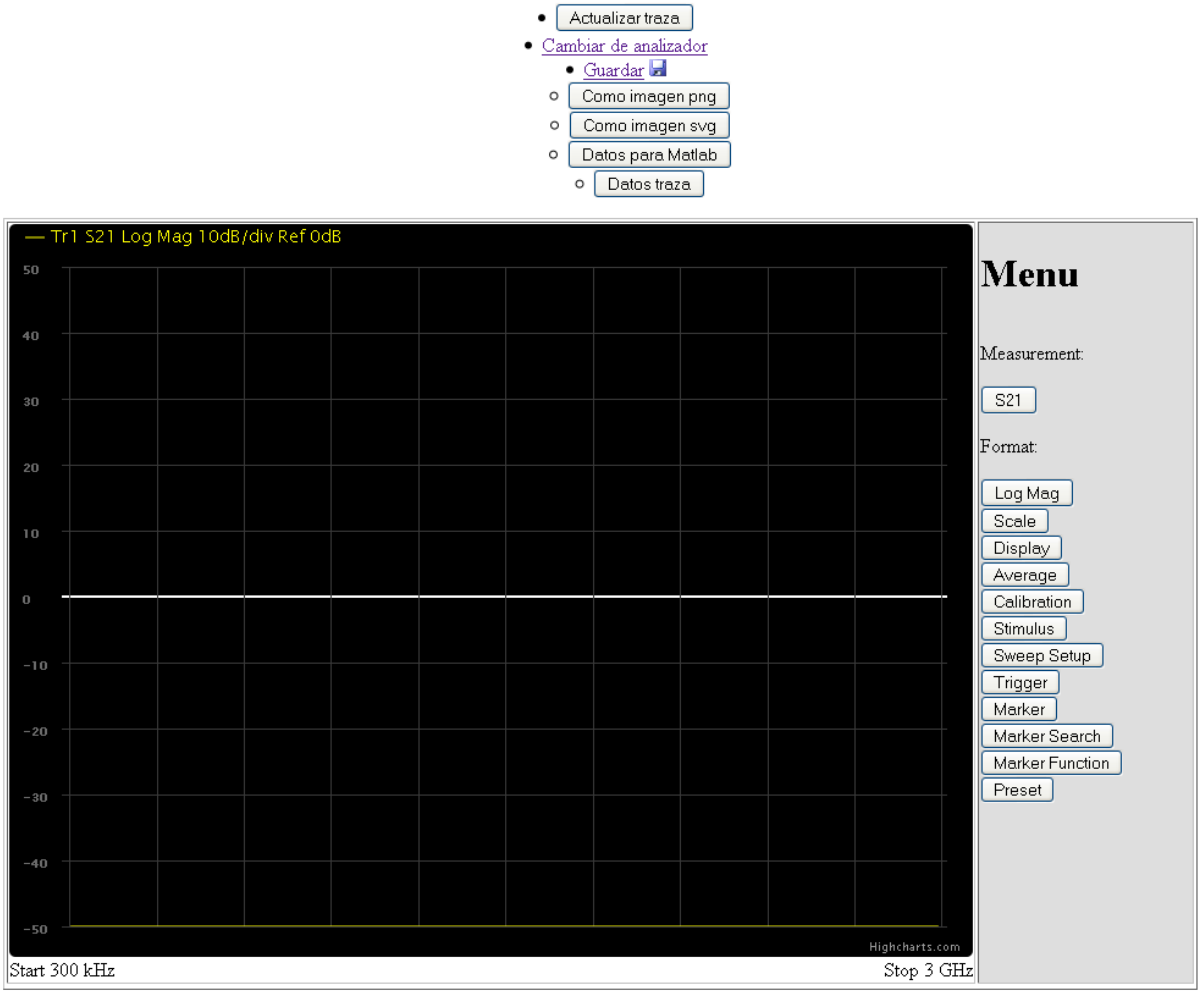
bien ha habido veces que ha sido necesario. Un ejemplo de ello sería la introducción de datos, donde se ha optado por la validación de los datos introducidos mediante la propiedad "pattern" de HTML5. Esta característica solo se ha podido usar en casos como la introducción de datos que modificaban la gráfica y se ha tenido que implementar una comprobación adicional con javascript, en la función que permite cambiar aspectos de esta.

CSS es un lenguaje de hojas de estilos usado para definir el aspecto de páginas webs escritas en HTML y XHTML. Define unas normas y características que luego serán aplicadas a cada elemento de la página web.

El código CSS ha sido validado al 100%. Y se ha procurado que la página web sea visible aunque éste no esté presente. Preservando que en caso de algún error pueda ser utilizado. De la misma manera se ha tratado no hacer un uso excesivo de las características de CSS para colocar los elementos dentro de la página web.



Gráfica 11: Página web con código CSS activado



Gráfica 12: Pagina web sin código CSS.

La utilización de HTML5 y javascript permite que la página web pueda ser utilizada desde cualquier navegador actualizado. Incluyendo los navegadores soportados por dispositivos móviles. Para la entrada de datos, se utilizan etiquetas HTML5 específicas para números decimales.

De las librerías gráficas utilizadas para la representación de la traza, se hablará más adelante, ya que utilizan código javascript que lo ejecuta el navegador web.

2.IV.3 HTTP

El protocolo HTTP es un protocolo sin estado. Es decir, no se almacena en el servidor información sobre conexiones anteriores producidas en el servidor. Es un protocolo de petición-respuesta, en el que el cliente envía datos y el servidor le da la respuesta en función de los datos enviados. Después de eso se cierra la conexión, y si el cliente quiere obtener datos de nuevo del servidor tiene que volver a enviar de nuevo una petición.

Para enviar datos al servidor web existen 2 métodos:

- Get: Los datos se envían mediante la URL, es decir la dirección de acceso al servidor. Por ejemplo si buscamos la palabra "busqueda" en google enviamos al servidor lo siguiente: "https://www.google.es/search?q=busqueda&ie=utf-8&oe=utf-8&aq=t&rls=org.mozilla:es-ES:official&client=firefox-a&channel=fflb". Como podemos ver se incluye busqueda en la URL así como otro datos.

- Post: Los datos se envían junto con la petición web al servidor.

La principal diferencia entre estos 2 métodos, es que con el método Post no hay limite de datos adjuntos. Estos 2 métodos están pensados para:

- Get: Está pensado para enviar ordenes o datos al servidor. Los datos enviados pueden ser almacenados en el historial y compartidos con otra persona si queremos que accedan a la misma orden.

- Post: Su función es la de subir archivos o datos que deben ser almacenados en el servidor.

En la práctica estos 2 métodos se utilizan indistintamente. Ambos son igual de inseguros para enviar información sensible al servidor porque http no es un protocolo cifrado.

En el proyecto se utiliza el método GET, ya que queremos enviar ordenes al servidor web.

Esto crea un problema, que se ha de forzar al navegador mediante código HTML para que envié ciertos datos cada vez que se acceda. Como puede ser el nombre de usuario, idioma en el que queremos la página web u opciones de configuración.

Para solucionarlo se crearon las cookies. Una cookie es un fragmento de información que se guarda en el ordenador del usuario, que se guarda a través navegador con el que visitó la página web y está asociada al dominio de cada página web. Con lo cual 2 navegadores no contiene las mismas cookies. Dependiendo del navegador la información se guardará de una forma o de otra.

Cuando el navegador accede a una página web comprueba si tiene alguna cookie asociada y si la hay la envía.

2.IV.4 Sesiones en PHP

Esto no sirve para identificarse de manera segura en una página web. La razón es porque la cookies son modificables, además de enviarse sin cifrar. Si guardamos el nombre de usuario, cualquier usuario podría modificar las cookies y autenticarse con otro nombre. Por otra parte si se guardara la contraseña en las cookies y se enviara sería inseguro pues viajaría en cada conexión y podrías ser fácilmente descubierta.

La solución a esto son las sesiones PHP. Que permiten almacenar datos sobre el usuario en el servidor web. El servidor web almacena por cada usuario y navegador un id. Así el usuario cada vez que accede se identifica en la página web.

2.IV.5 Funcionamiento

Como se ha visto antes, 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.

De la misma forma, se ha mantenido esta premisa a lo largo de todo el proyecto, excepto en el caso de los analizadores, el protocolo de conexión creado entre el Servidor VISA y el Servidor Web es sin estado. El Servidor VISA no almacena datos sobre las conexiones recibidas, el Servidor Web es el encargado de almacenar la información sobre donde debe acceder para conectarse a cada analizador.

Cuando se recibe una petición Web, se sigue un diagrama de decisiones, según el estado en el que se encuentre el usuario y lo que haya enviado, se le dirigirá al menú correspodiente de la página.

2.V Comunicación Servidor VISA – Servidor Web

Tenemos 3 tipos de conexiones que se realizan entre estos 2 servidores. Cada una con una funcionalidad distinta. En común tienen todas, que son realizadas bajo protocolo TCP/IP y los datos se envían en formato ASCII.

Los puertos han sido escogido desde el 55550 en adelante. Son puertos que no están destinados para ningún servicio.

2.V.1 Servidor de intrumentos

Se realiza mediante una conexión al puerto 55550. Una vez establecida la conexión se envía al servidor web los analizadores que hay conectados siguiendo el siguiente formato:

<nombre del analizador 0(Cadena IDN)><Tipo de analizador 0><nombre del analizador 1(Cadena IDN)><Tipo de analizador 1>...

Para diferenciar cada campo del envío se utiliza “\n” como carácter de separación.

También se indica el puerto al que deberemos conectarnos para acceder a cada analizador mediante la posición que se envía. Siendo el 55550 + 2*i para enviadatos y 55552 + 2*i para

enviacomandos. Así el analizador que ocupa la posición 2 sería el puerto 55554 y 55555.

2.V.2 Enviadatos

El puerto en el que se establece la conexión, depende del orden en el que haya sido detectado el analizador. Siendo el puerto a partir del 55551 y calculándose de la manera anteriormente vista.

El envío de datos se realiza en ASCII. Si se enviara en otro formato se perdería el tiempo convirtiendo los datos a un formato más eficiente para ambos extremos. Para valores enteros se enviaran tantos decimales como tengamos con un máximo de 10 dígitos, para variables booleanas se enviará 0 para indicar falso y 1 para indicar verdadero.

Como el rango de frecuencias es muy amplio y mediante código ASCII supondría el envío de un alto número de ceros, el formato de envío para las frecuencias varía un poco y se incluyen las unidades en las que se encuentran. Por ejemplo un valor de 3 500 000 Hz, se enviaría con la siguiente cadena: “3.5 MHz”, utilizando “.” para separar la parte entera de la parte decimal y así poder ser igualada en PHP directamente a un valor entero.

Esto se aplica a todos los valores que estén medidos en frecuencia, incluyendo el ancho de banda del filtro de resolución. Aunque como los analizadores 8714ET no nos dejan utilizar valores discretos del filtro de resolución, cuando se envían los datos desde el Servidor Web al Servidor VISA, se envía un valor que indica el filtro que ha sido escogido de entre los disponibles. Aunque se almacena en formato entero para que sea compatible con analizadores más nuevos.

Valores discretos como son el formato de la traza o el parámetro medido, que no van a pasar de 256 opciones, se utilizan unsigned char para referirse a la opción escogida.

Para conectarse, el servidor espera en el puerto que le ha sido asignado para recibir una conexión entrante. Cuando la recibe, el cliente le envía:

<Id de traza><Menú en el que se encuentra (Opcional)><Opción realizada (Opcional)>

No existe ninguna separación entre los campos ya que tienen una longitud definida excepto la opción realizada. Siendo Id de traza y el Menú un byte. Para averiguar internamente si se ha enviado los valores de menú y opción se compara con '\0' que es el carácter que indica final de cadena.

Después se elabora la respuesta con los datos proporcionados por el cliente siguiendo el esquema visto anteriormente en la explicación de enviadatos.

2.V.3 Enviacomandos

El puerto en el que se establece la conexión depende del orden en el que haya sido detectado el analizador. Siendo el puerto a partir del 55552 y calculándose de la manera anteriormente vista.

La conexión con enviacomandos solo se realiza, cuando el cliente desea comunicar con el analizador, para ello una vez inicia una conexión con el servidor, el cliente envía:

<Menú en el que se encuentra><Opción realizada>

En base al menú en el que se encuentre el cliente se llamará a la opción adecuada para cada menú.

Después de realizar la acción elegida en el analizador, y de recibir la respuesta de este se enviará al servidor el resultado, indicando con el primer dígito de la cadena que se envía si se ha de mostrar por pantalla al usuario algún mensaje o no. Si se envía un 0 se indicará que el mensaje que viene a continuación se ha de mostrar, y con 1 que no se debe de mostrar nada.

2.V.4 Medida de tiempos

Para realizar la medida de tiempos, para el posterior modelado del sistema, vamos a realizar las pruebas siguiendo 2 esquemas: con conexión de ambos servidores dentro del mismo ordenador, conectados por una máquina virtual y conexión entre distintos ordenadores.

El proceso seguido por el servidor PHP al conectarse al Servidor VISA es el mostrador por la ilustración 12 .

Así pues se podría pensar que tenemos 2 tiempos para calcular. El tiempo de respuesta de enviacomandos y el tiempo de respuesta de enviadatos.

En enviacomandos, no se puede optimizar ningún tiempo, puesto que el tiempo que tarde en responder, es el tiempo que tarde el analizador en realizar el comando sea cual sea. Mientras tanto si el servidor recibe otras peticiones estás quedaran en cola hasta que el comando sea resuelto.

Existe la posibilidad de que si recibimos muchos comandos simultáneamente y durante mucho tiempo seguido, se provoque un bloqueo en el servidor. Pero como una de las primeras premisas en las que se basó este proyecto, es el uso único del analizador para un usuario, la única manera de que ocurriera esto, es que el usuario enviara repetidamente durante mucho tiempo la misma orden de manera periódica. Con lo cual se entiende un buen uso del proyecto del usuario y no va a producir este hecho. Por si acaso se ha puesto a prueba este bloqueo, poniendo en cola más de 10 ordenes seguidas, con posibilidad de repetición por parte del usuario, sin perjudicar esta prueba a su funcionamiento. Así que no se estudiaran los tiempos de respuesta de enviacomandos.

2.V Comunicación Servidor VISA – Servidor Web

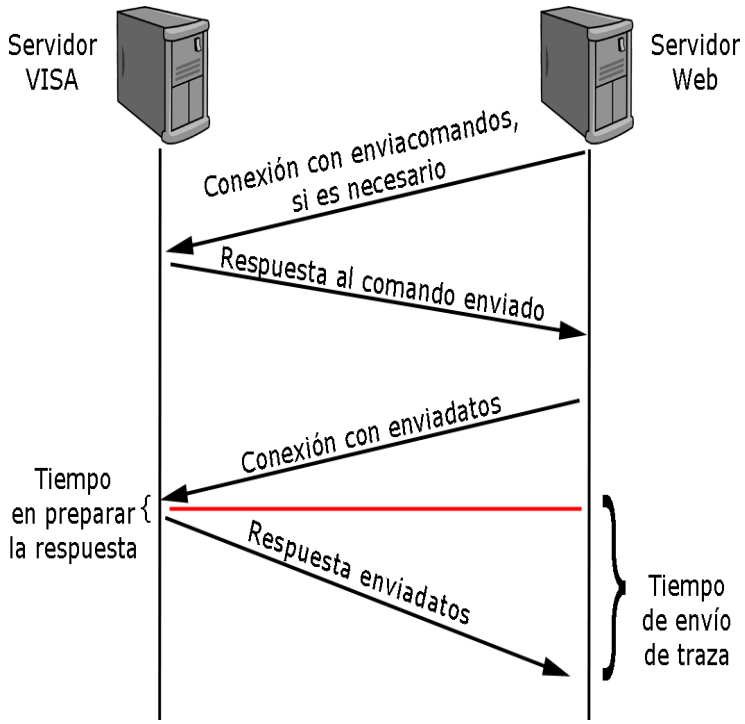


Ilustración 12: Comunicación entre los 2 servidores.

El otro envío es más interesante, ya que depende intrínsecamente del Servidor VISA. No se realiza ninguna comunicación con el servidor y puede ser mejorable. También es conveniente medir este tiempo, para más tarde cuando analicemos el sistema, tenerlo en cuenta. Ya que se podría producir un bloqueo en el modo de traza automático, incumpliendo el refresco de traza de 1 segundo o incluso llegando a colapsar el Servidor VISA.

Anteriormente ya se ha visto tanto la petición del servidor, como la respuesta por parte de este. Así podemos ver 2 tiempos claramente diferenciados, el tiempo que se tarda en preparar la respuesta y el tiempo que tardamos en enviar dicha respuesta.

2.V.4.i Tiempo en preparar la respuesta

El tiempo que se tarda en preparar la respuesta, depende del ordenador sobre el que esté funcionando el Servidor VISA. Además de poseer el inconveniente, de que este tiempo bloqueará también la CPU del Servidor. Este tiempo puede ser mayor o menos si tenemos una traza nueva disponible.

Si ambos Servidores se ejecutan en el mismo sistema operativo, éste será el único tiempo que tendremos. Se puede considerar 0 el tiempo de envío en esta circunstancia.

Se podría suponer que este tiempo es cercano a 0; ya que es un acceso a las variables internas del servidor. El problema radica en que hemos de convertir estos datos, a datos de tipo cadena para enviárselos al Servidor Web. Si hay que enviar una traza, el caso peor, aparece cuando se quiere enviar una traza con carta de Smith de 1601. Teniendo que recorrer un vector de 3202 puntos y

convirtiendo cada uno a formato ASCII.

El valor medio medido es de 71,6 ms, teniendo una desviación típica de 2,6 ms. Estos valores han sido medidos,con el Servidor VISA atendiendo a un analizador y el otro en reposo. Estos valores podrían mejorar o empeorar si:

- El Servidor VISA se encuentre en el mismo ordenador que el Servidor Web; ya que ambos compartirán CPU y podrán ralentizar este proceso. Del mismo modo que una carga excesiva de CPU por otro programa puede empeorar.

- El Servidor Web se ejecute en una máquina virtual. Produciendo efectos no deseados en este tiempo. Al depender del sistema que se ejecute.

- La potencia de la que disponga el ordenador donde se ejecuta.

2.V.4.ii Tiempo de envío de traza

En este caso, el tiempo varía con los 2 casos que se han presentado anteriormente. Así pues tenemos 2 tiempos que estudiar y tener en cuenta.

La diferencia entre valores se debe a que en uno tenemos una velocidad de transmisión de Gigabit ethernet para el caso de que ambos se encuentre en el mismo ordenador y de 100 Mb/s si se encuentran en diferentes ordenadores. Si ambos tuvieran la misma velocidad probablemente no habría mucha diferencia entre ambos:

- Para el mismo ordenador se ha obtenido un tiempo de 23 ms de media y una desviación típica 3,9 ms.

- Para el distintos ordenadores se ha obtenido un tiempo de 51,2 ms de media y una desviación típica 4,2 ms.

2.V.4.iii Conclusiones

Los valores medidos nos dejan con un valor medio de bloqueo de Servidor VISA de: 94,6 ms, no permitiendo dejar acceder a otros navegadores al mismo analizador, con un valor máximo de 100,4 ms. Para comprobar que este valor es correcto se ha medido desde el otro lado del servidor VISA, concretamente desde el Servidor Web. Este tiempo ha dado como resultado: 108,7 ms, un valor parecido si tenemos en cuenta, que el aumento con respecto al Servidor VISA, se debe al posterior tratamiento de los datos recibidos para poder representarlos en la gráfica.

Estos valores junto con el valor medido para guardar la traza, más tarde serán importantes para el análisis de tiempos del sistema, ya que nos marcarán los límites de funcionamiento del proyecto.

2.VI Navegador Web

En el usuario final se ejecuta un navegador web. La ventaja que permite es que actualmente los dispositivos que soportan HTML5 es un área bastante variada. La principal característica que utiliza el proyecto es el añadido en HTML5 del almacenamiento mediante javascript de datos en el cliente. La diferencia con las cookies, es que no viajan en cada petición web que se realiza a la página.

2.VI.1 Local storage y sesion storage

Esta opción es soportada por la mayoría de los navegadores a partir de Internet explorer 8, basados en Mozilla 3.5, Safari 4, Google Chrome 5 y Opera 10.5. El tamaño máximo de los datos no puede exceder de los 5 Mbytes para cada tipo de almacenamiento como mínimo. Algunos navegadores poseen mayor capacidad.

Existen 2 tipos de almacenamiento: local y de sesión. La diferencia entre un modo de almacenamiento y otro es el tiempo de vida y el ámbito en el que se utiliza.

- Para sesión, su tiempo de vida está limitado hasta que se cierre la pestaña que ha creado los datos y su ámbito es única y exclusivamente para esa pestaña.

- Para local, los datos permanecen aunque el navegador sea cerrado y se puede utilizar por cualquier pestaña que tenga el mismo dominio web que quien la creó.

Solo permite almacenar cadenas. Para almacenar otro valores utilizaremos el método de javascript "*JSON.stringify*" para convertir variables a cadenas y "*JSON.parse*" para devolverlas a su estado anterior.

En el proyecto se utilizan ambos tipos de almacenamiento. Pasamos a comentar que variables son de cada tipo:

Almacenamiento local:

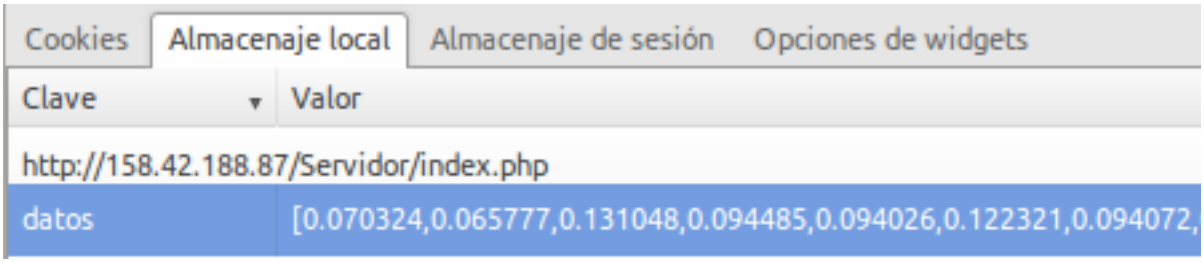


Ilustración 13: Variables de almacenamiento local.

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. Se utiliza como variable de sesión local porque, con los supuesto hechos en prácticas anteriormente, no había ninguna razón para que un usuario accediera a más de un analizador a la vez. También siendo PHP

el que almacena los parámetros del analizador y el valor de identificación de traza.

Ilustración 14: Almacenamiento de sesión.

Las variables de sesión de PHP no distinguen entre pestañas de un navegador, pero si entre navegadores; así pues no tenía sentido que cada pestaña tuviera su propia traza. Esto tiene 2 consecuencias:

- Bajo el mismo navegador no se puede acceder a la vez a 2 analizadores. Porque comparten el mismo valor almacenado de la traza.

- Si tenemos más de 2 pestañas abiertas no hace falta que las 2 pidan la traza al servidor, ahorrando recursos.

Si se deseara implementar un acceso a 2 analizadores podría ser posible moviendo está variable a almacenamiento de sesión y almacenando en las cookies el valor del identificador de traza y no en PHP.

En la ilustración 14 se muestra lo que se almacena en sesión. Se guardan todos los valores que configuran el estado de la gráfica. Así como las opciones de configuración de la función añadida "buscar picos".

Todos estos valores, se almacenan en sesión para que cada pestaña pueda tener unos valores independientes de la misma gráfica y así poder tener 2 pestañas abiertas, cada una por ejemplo, con una parte de la gráfica o unos valores marcados.

Una vez ya tenemos todos los valores almacenados, debemos representar los datos por pantalla.

2.VI.2 Plugin gráfico: Highcharts

En un principio se enviaban imágenes desde el servidor PHP con la traza representada. Pero este método no dejaba interactividad al usuario y no permitía representar los datos de la manera deseada, que es lo más cercana a los analizadores. Con lo cual se buscaba algo que permitiera al usuario interactuar con la gráfica.

Con la llegada de HTML5 se pusieron a disposición de los programadores una serie de métodos en javascript que permitían dibujar gráficas. Teniendo en cuenta esto, se empezó una búsqueda de plugins gráficos que funcionaran bajo javascript y permitieran el efecto deseado.

Al buscar plugins gráficos se barajaron varias opciones, lo primero que se deseaba, es que no se necesitara acceso a Internet para crear las gráficas. Ésto descartó la utilización de Google Charts, un plugin gráfico bastante completo, pero que requería el acceso a Internet para poder utilizar sus capacidades gráficas.

El segundo plugin que se probó fue: "amCharts". Aunque era bastante interesante, porque permitía opciones como: hacer zoom, visualización de valores de traza con solo pasar el ratón por encima de ella y reconocimiento de clics del usuario sobre la traza, no permitía ajustar los valores de los ejes según el esquema de divisiones y Unidades/div que utilizan los analizadores. Así como tampoco permitía la representación de la carta de Smith o formato polar.

Finalmente observando las estadísticas usadas en Internet, y en el proceso de investigación, se descubrió (en el periódico 20 minutos): un plugin gráfico que se ajustaba a los requisitos del proyecto: "Highcharts JS".

Entre los clientes de este plugin gráfico se encuentran: IBM, NASA, Siemens, HP, EMC, CBS, Hitachi, Ericsson, BMW, Nissan, Sony, Fujitsu, Citi, Rabobank, RIM, BBC, Financial Times and MasterCard.

Para fines no comerciales se distribuye bajo licencia creative commons 3.0. Permitiendo ser

usado por el proyecto con la inclusión de un pequeño enlace en la gráfica a la página web del creador.

Highcharts nos permite realizar zoom en la gráfica, mostrar los valores de la gráfica cuando ponemos el ratón encima de un punto, detectar cuando el usuario hace clic en un punto de la gráfica y realizar capturas de pantalla de la gráfica, en forma de gráficos vectoriales. También permite establecer los límites de los ejes, de forma que encajen con el esquema de los analizadores y realizar gráficas en formato polar (aunque en estas no funcionan los marcadores).

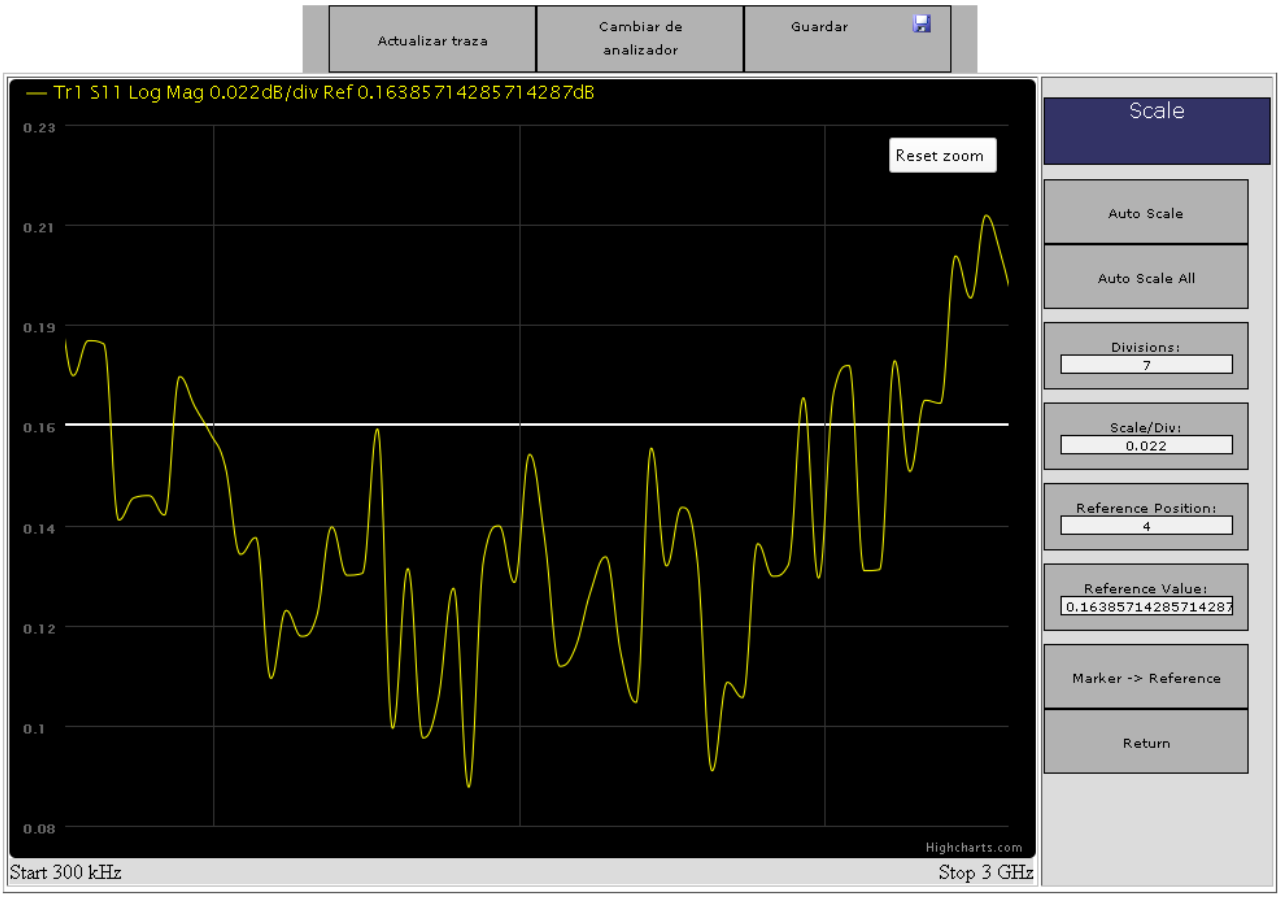


Ilustración 15: Ejemplo de zoom en la gráfica.

A cambio si utilizamos el modo continuo, al actualizar los datos de la gráfica, cada segundo tenemos que crear una gráfica nueva. La consecuencia de esto, es que no nos deja mantener las funciones de zoom, ni en la marca que aparece para ver los datos. Solo se ha podido arreglar el problema con la marca de los datos. El zoom no se pudo arreglar debido a un bug en el plugin gráfico que actualiza los valores de la gráfica. Para arreglar las marcas, se realizó un nuevo dibujo del texto de los datos, aprovechando las funciones de la gráfica. Al controlarse el dibujo del punto resaltado, se pueden guardar los datos, y si hay actualización, volver a mostrarlos.

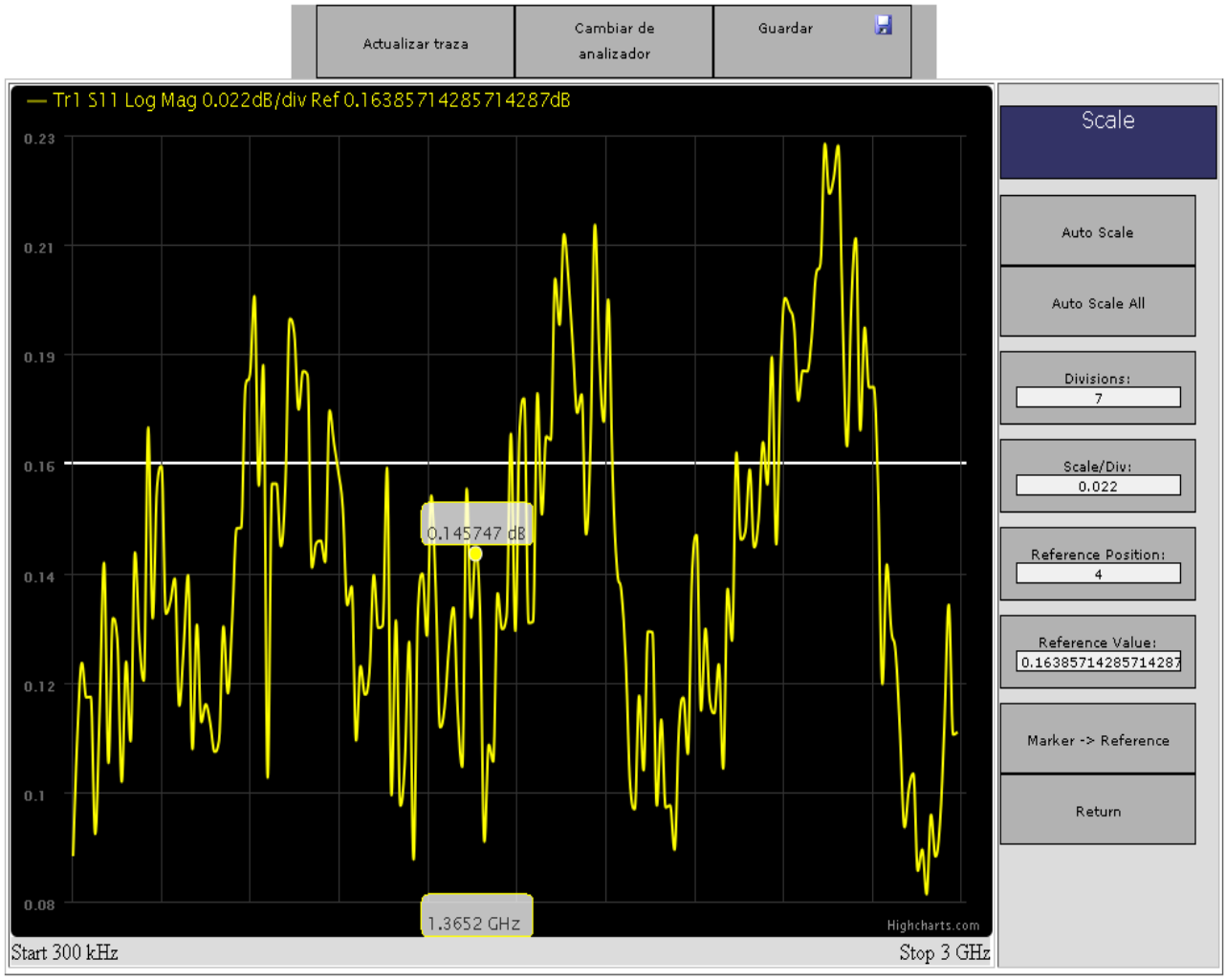


Ilustración 16: Ejemplo de globo de datos que aparece al pasar el ratón sobre la gráfica.

La función de realizar capturas de pantalla de la gráfica también tiene 2 inconvenientes que fueron resueltos.

1. Las divisiones de la gráfica, se realizan manualmente dibujándolas sobre los valores que queremos, que han sido previamente calculados. Esto provoca que no aparezcan al realizar una captura de pantalla. La solución está en que cuando se realiza una captura, la gráfica incluye automáticamente sus propias divisiones, que no corresponden con los valores incluidos en la configuración. Los valores límites de los ejes escogidos para la gráfica se redondean al realizar la captura.

2. Al estar en formato vectorial, no deja incluir imágenes, a pesar de que, éstas también estén en dicho formato. Como consecuencia al hacer una captura no aparecen los números de cada marcador y al realizar capturas en formato polar y carta de smith no aparece su fondo correspondiente. Aunque al ser sobre fondo negro y estar en formato vectorial se podría insertar de forma sencilla la carta después si se necesitara. La diferencia se puede observar a continuación en las ilustraciones 17 y 18.

2.VI Navegador Web

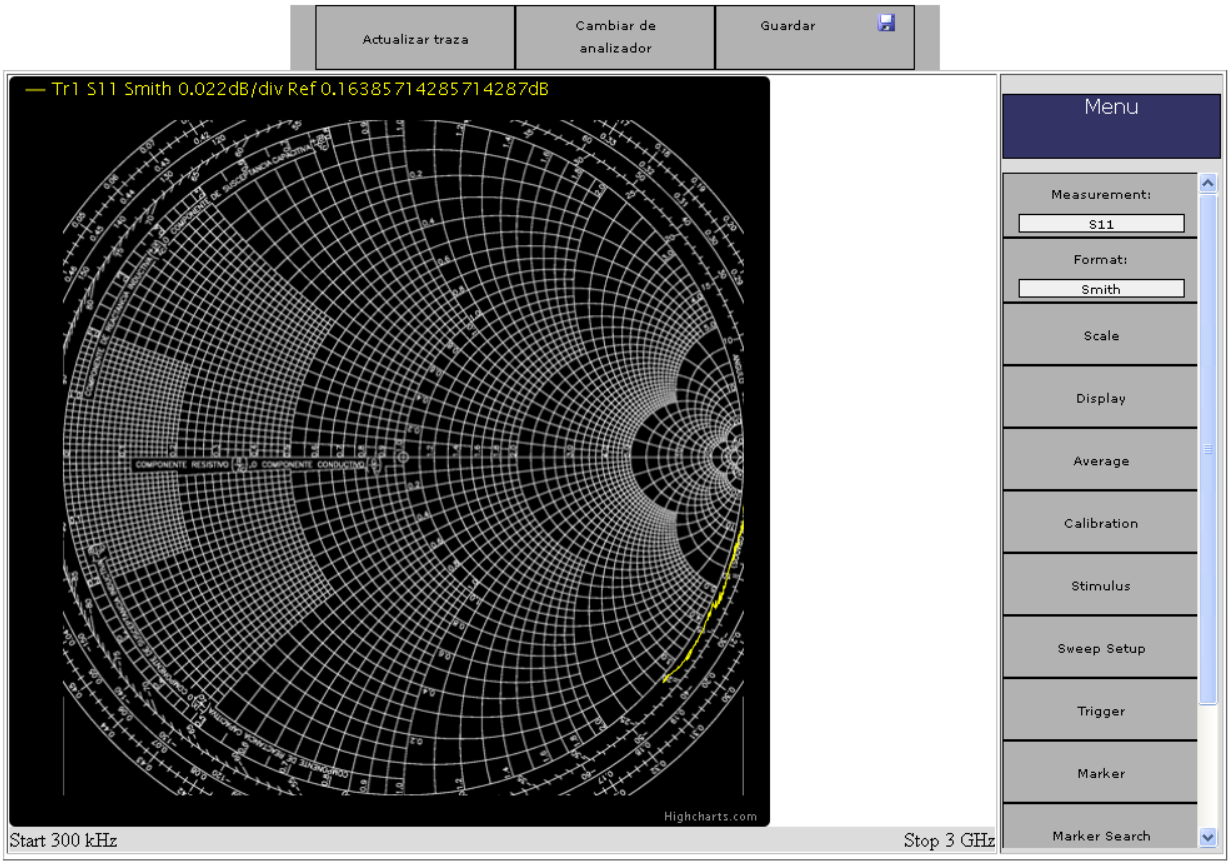


Ilustración 17: Captura de una carta de Smith desde la página web.

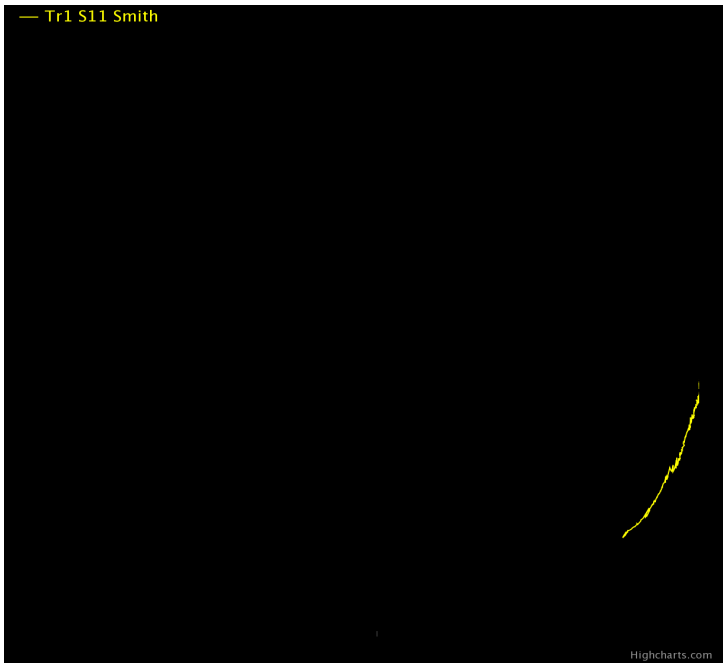


Ilustración 18: Captura de una carta de Smith desde la opción de guardar captura de pantalla.

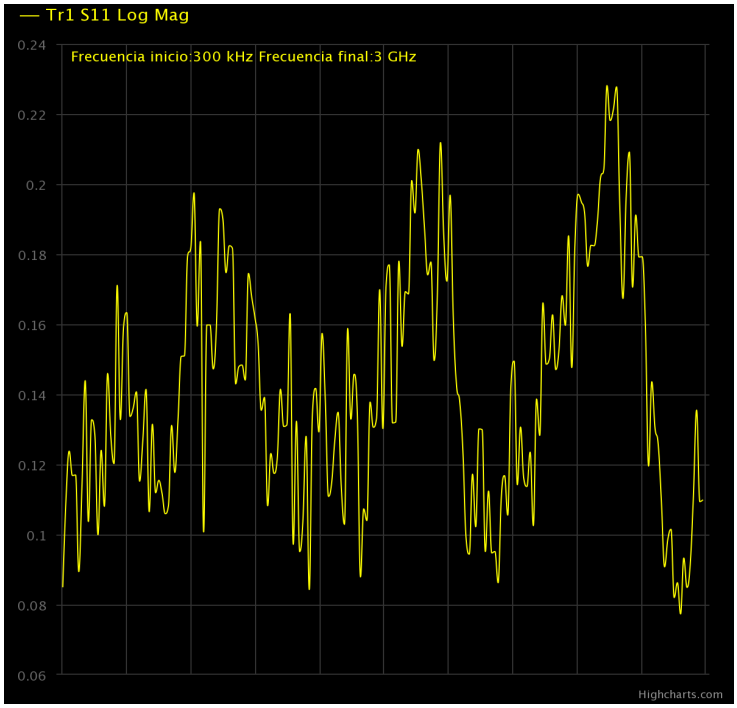
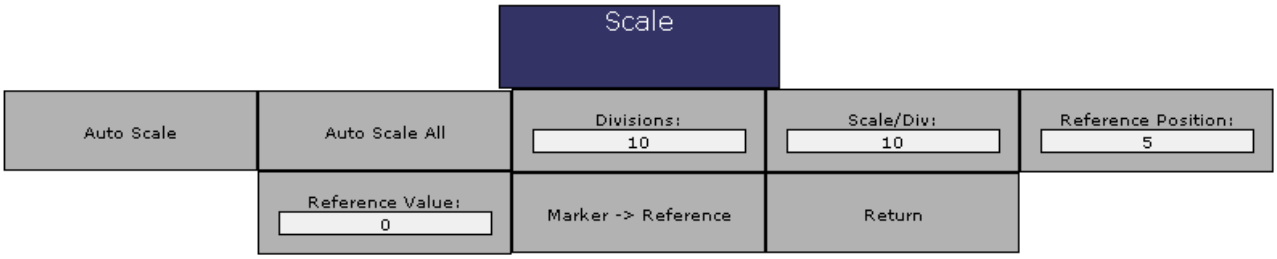


Ilustración 19: Captura de pantalla normal.

2.VI.3 Menú gráficos

Existen varios menús del proyecto que se procesan y dibujan sobre la gráfica. Todos ellos se realizan con javascript desde el lado del cliente.

2.VI.3.i Scale



Dibujo 1: Menú Scale.

El menú Scale se utiliza para controlar los valores de los ejes de la gráfica. Con “Divisions” y “Scale/Div” se controla el tamaño de cada división y el número de divisiones de la gráfica. Con “Reference position” se indica en que división queremos situar el valor del eje x de la gráfica, y con “Reference value” en que valor numérico.

La funciona “Auto Scale” sitúa estos valores automáticamente maximizando y ajustando los

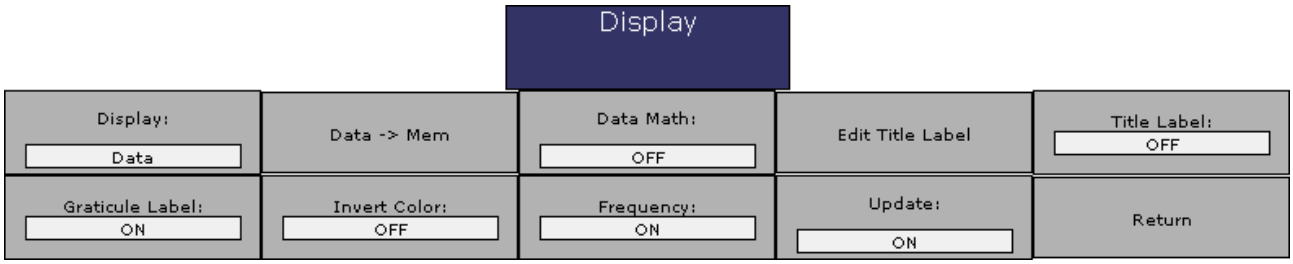
limites de la gráfica al máximo y mínimo de la función y centrando el valor “Reference Value”.

Con “Marker → Reference” se sitúa el valor de la posición de referencia al marcador actual seleccionado, si está seleccionado.

Las variables locales utilizadas en estos menús son:

- “Div”: Almacena el número de divisiones que quiere el usuario.
- “Posicionref”: Almacena la división sobre la que se encuentra el eje x.
- “ScaleDivisions”: Almacena el tamaño que queremos para cada división.
- “Valorref”: Almacena el valor donde se situá el eje x.

2.VI.3.ii Display



Dibujo 2: Menú Display.

Estas son las partes del menú display que funcionan. La opciones referidas a varias trazas y canales han sido desactivadas, porque no han sido implementadas.

Las 3 primeras opciones se utilizan para el manejo de trazas en la memoria del analizador. Con el primero podemos elegir entre ver la traza actual, la traza almacenada o una operación matemática entre ambos. Con el segundo se guarda la traza actual en memoria y el tercero activa la operación matemática deseada.

Las siguientes 2 opciones sirven para mostrar y editar el título que se mostrará en la parte superior de la gráfica como se muestra en la ilustración 20.

Con “Graticule Label” se muestran o no, los valores del eje y. Estos valores están dibujados sobre la gráfica para coincidir con las divisiones creadas.

“Invert color” cambia los colores de amarillo y negro a azul y blanco. También invierte los colores del fondo de la carta de Smith.

"Frequency" muestra las frecuencias de inicio y fin debajo de la gráfica.

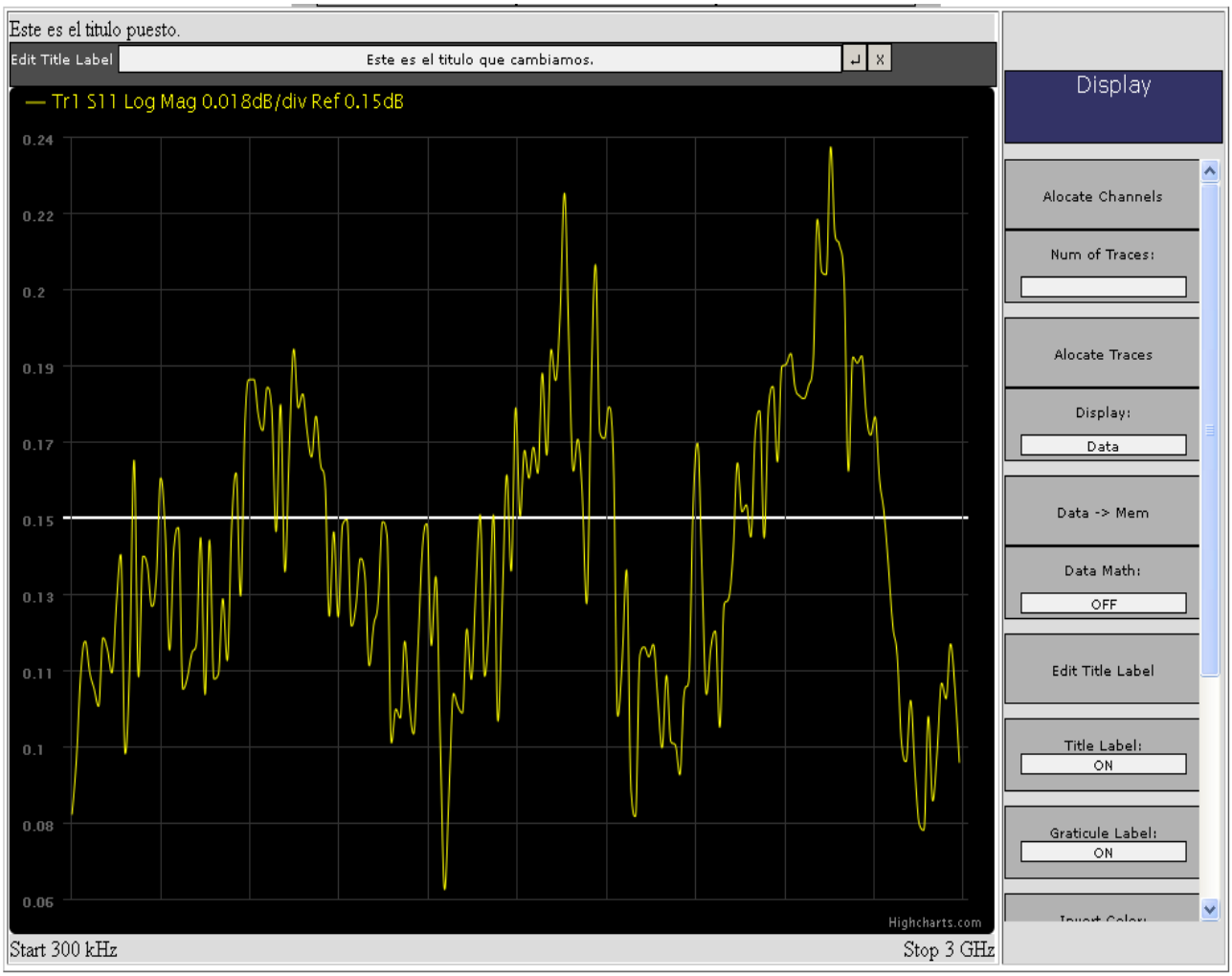


Ilustración 20: Menú para cambiar título.

La ilustración 21 muestra una gráfica con los colores invertidos, las frecuencias ocultas y los valores del eje y.

Si "Update" está desactivado, hace que el analizador no realice nuevos barridos.

Las variables locales que utiliza este menú son:

- "ejeyactivo": Indica si se muestran los valores del eje Y.
- "invertircolor": Si es cierto, los colores serán blanco y azul.
- "mostrarfrecuencia": Almacena si se muestran los valores de frecuencias debajo de la gráfica.

- “titulo”: Almacena el valor del nombre que el usuario quiere darle a la gráfica.
- “tituloact”: Muestra o no el valor anterior en la parte superior de la gráfica.

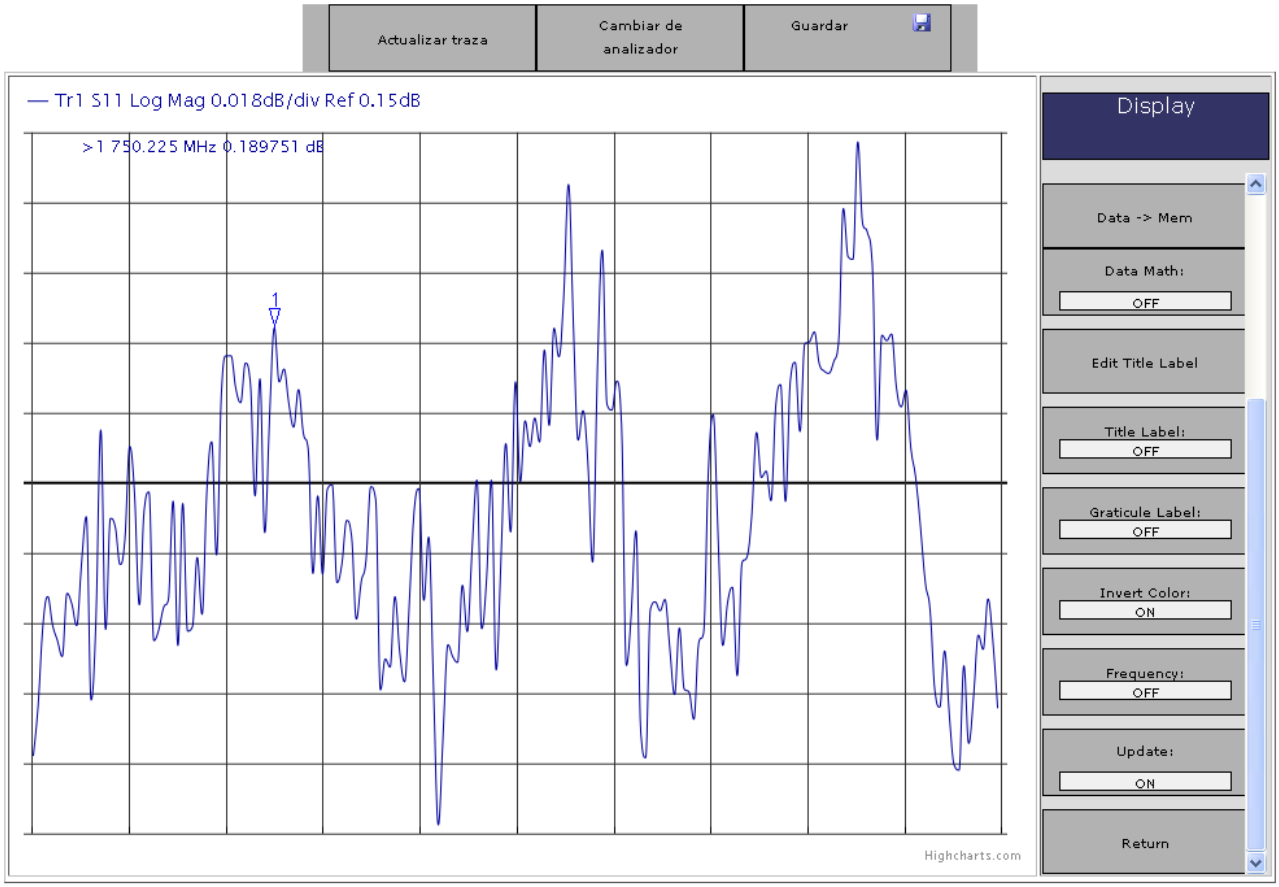
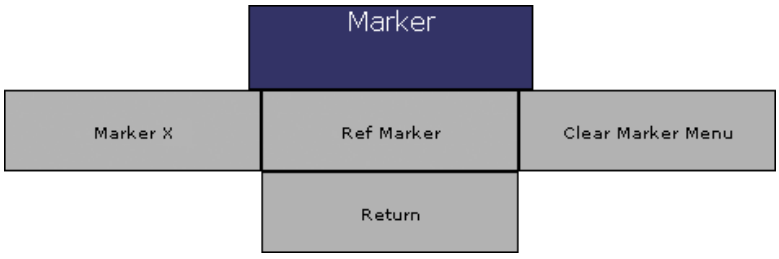


Ilustración 21: Gráfica con varios opciones activadas.

2.VI.3.iii Markers



Dibujo 3: Menu Markers.

Con este menú se manejan los marcadores de la gráfica. Los marcadores indican la frecuencia y valor del punto sobre el que se les sitúa.

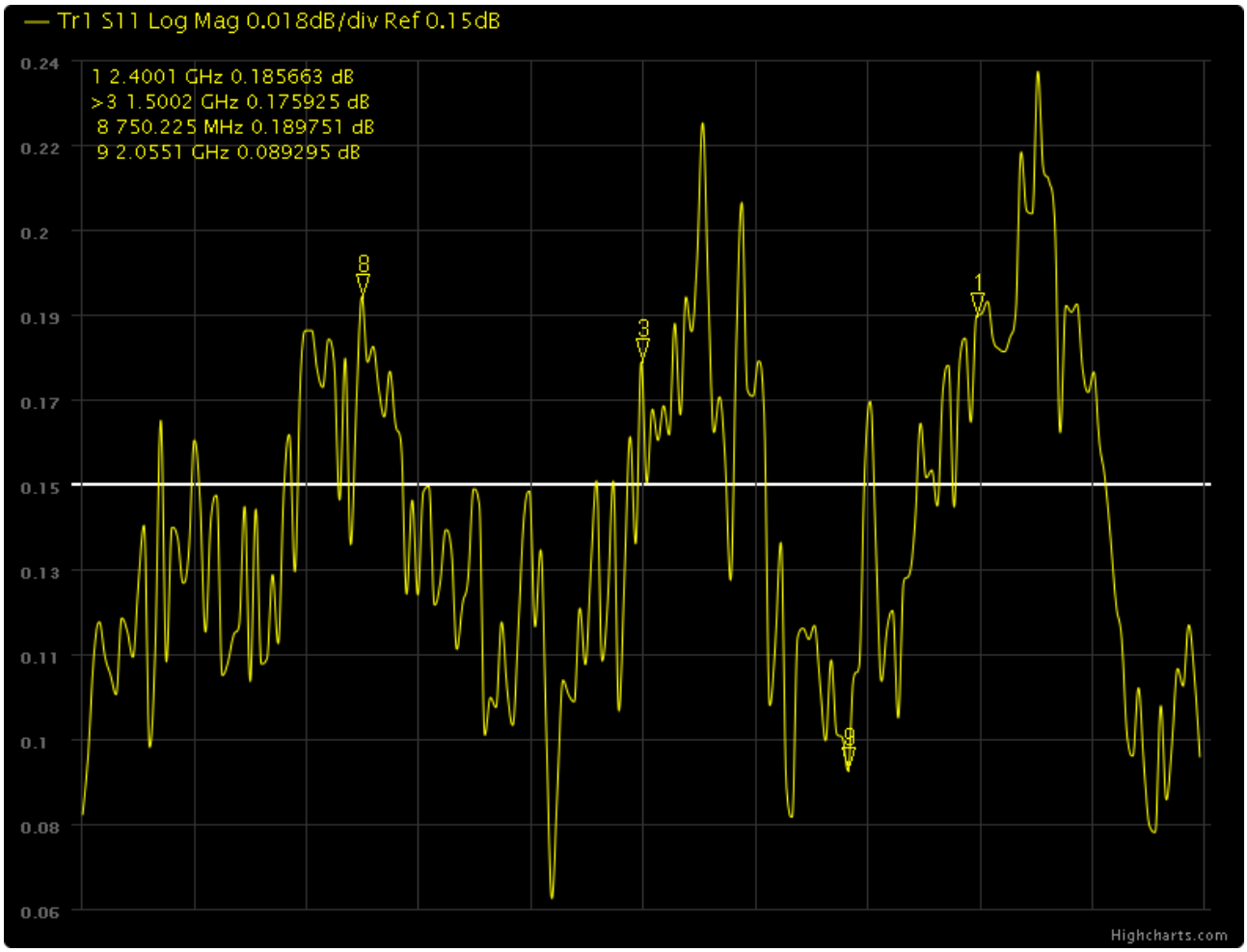


Ilustración 22: Ejemplo de marcadores.

Para utilizar un marcador: primero se debe activar en su botón correspondiente, por ejemplo Marker 1 para el marcador número 1 y luego con hacer clic en un punto de la gráfica, el marcador que hayamos activado se colocará en dicho punto. Para desactivarlo se realiza en "Clear Marker Menu".

Para cambiar de marcador activo, o bien se pulsa sobre el marcador que queremos activar en la gráfica, o se pulsa su botón correspondiente en el menú.

"Ref Marker" es el marcador diferencial. Cuando se activa este marcador, el valor del resto de marcadores se imprimen con la diferencia a donde este marcador diferencial esté situado. Excepto por ésto, su funcionamiento es el mismo que los demás marcadores.

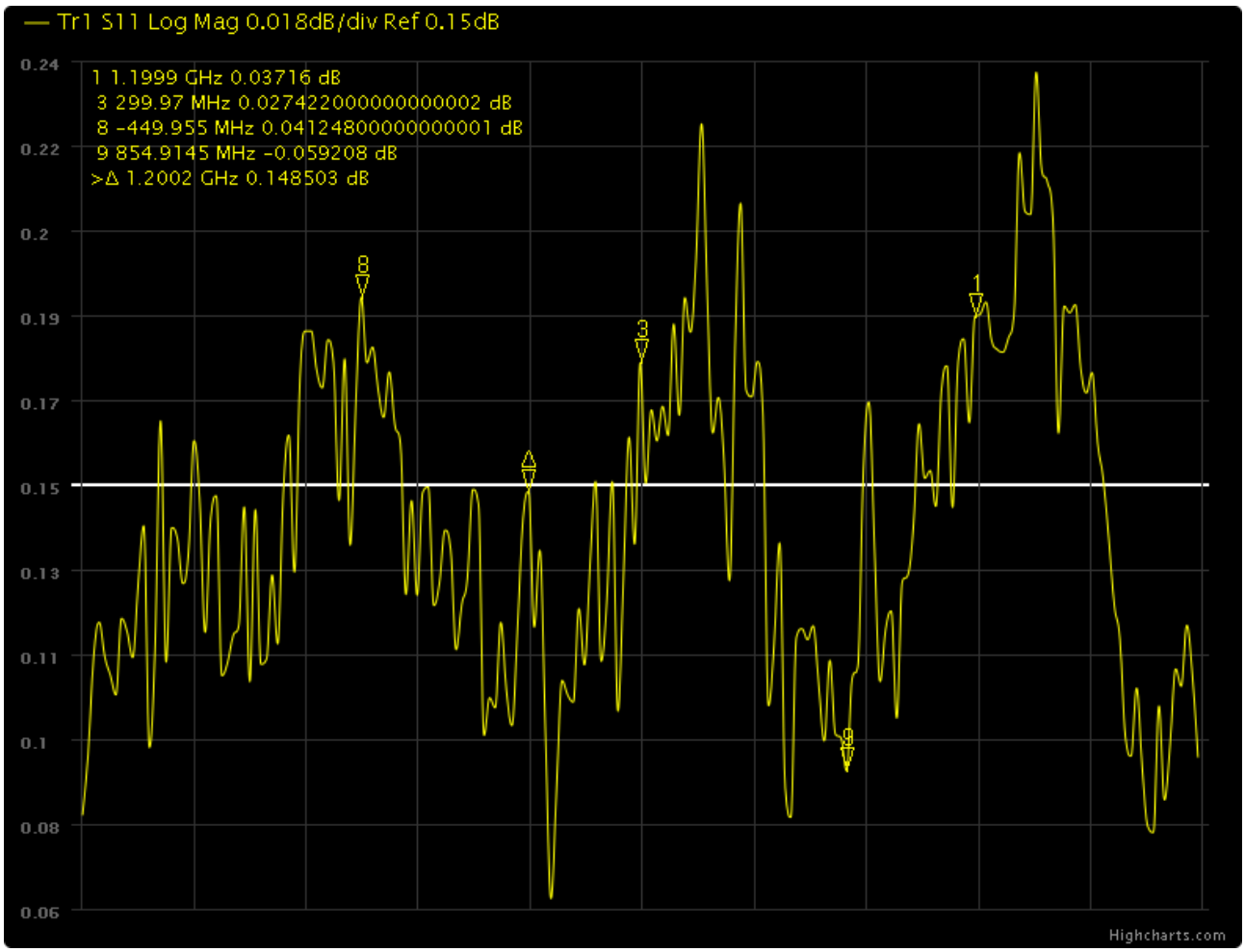


Ilustración 23: Ejemplo de funcionamiento de "Ref Marker"

Los marcadores se almacenan internamente en el navegador en las variables locales “Marcador” y “Marcadoractivo”. La primera es una matriz de tamaño 10 x 2. Donde la primera fila almacena, si el marcador correspondiente a esa posición (0 para el 1, 1 para el 2 y así sucesivamente) está activado. El segundo valor indica el punto sobre el que se encuentra el marcador.

Como los marcadores están referidos internamente a un punto en la gráfica, si cambiamos las frecuencias o el número de puntos de la traza, los marcadores que hayamos posicionado se mantendrán en el punto que estaban, pero no en la misma frecuencia.

2.VI.3.iv Marker Search

Dibujo 4: Menú Marker Search.

En este menú se utilizan los marcadores para buscar puntos destacados de la gráfica. Con "Max" y "Min" se busca el punto máximo y el mínimo de la gráfica y se lleva el marcador actual allí.

Con "Buscar Pico" se colocará el marcador sobre el siguiente pico que encontremos con las características definidas a partir de donde se ha situado el marcador. Sino encuentra nada se dejará el marcador en el punto donde estaba.

Con "Multipico" se buscaran todos los picos que respondan a las características definidas.

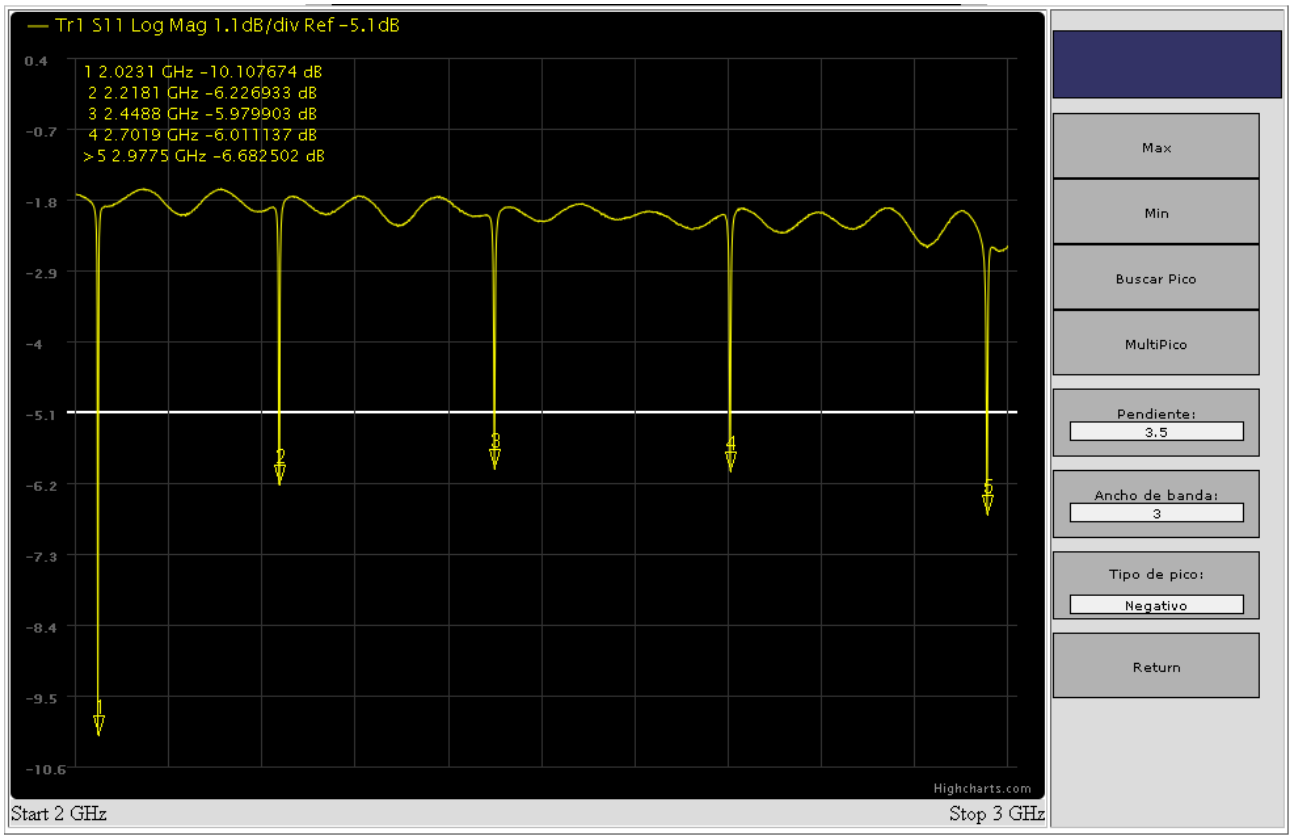


Ilustración 24: Resultado de aplicar la función Multipico.

Para configurar “Buscar Pico” y “Multipico” tenemos 3 factores:

- “Pendiente”: indica cuanto estimamos, o queremos, que caiga el pico en el ancho de banda dado.
- “Ancho de banda”: es un porcentaje que especifica el porcentaje de pantalla que ocupa el pico.
- “Tipo de pico”: si el pico que queremos que se busque es negativo, positivo o ambas opciones.

El parámetro “Ancho de banda” y “Pendiente” están relacionados entre si. Si incluimos un valor de ancho de banda muy pequeño con respecto al pico que queremos detectar, la pendiente tendrá que ser menor para poder detectarlo. Si incluimos un valor de ancho de banda muy grande, puede que 2 picos nos los detecte como uno solo. Igualmente si el valor de pendiente es muy pequeño, puede que nos detecte como pico partes de la gráfica que nosotros no consideramos así. En la ilustración siguiente se puede observar cada uno de estos parámetros para un pico negativo.

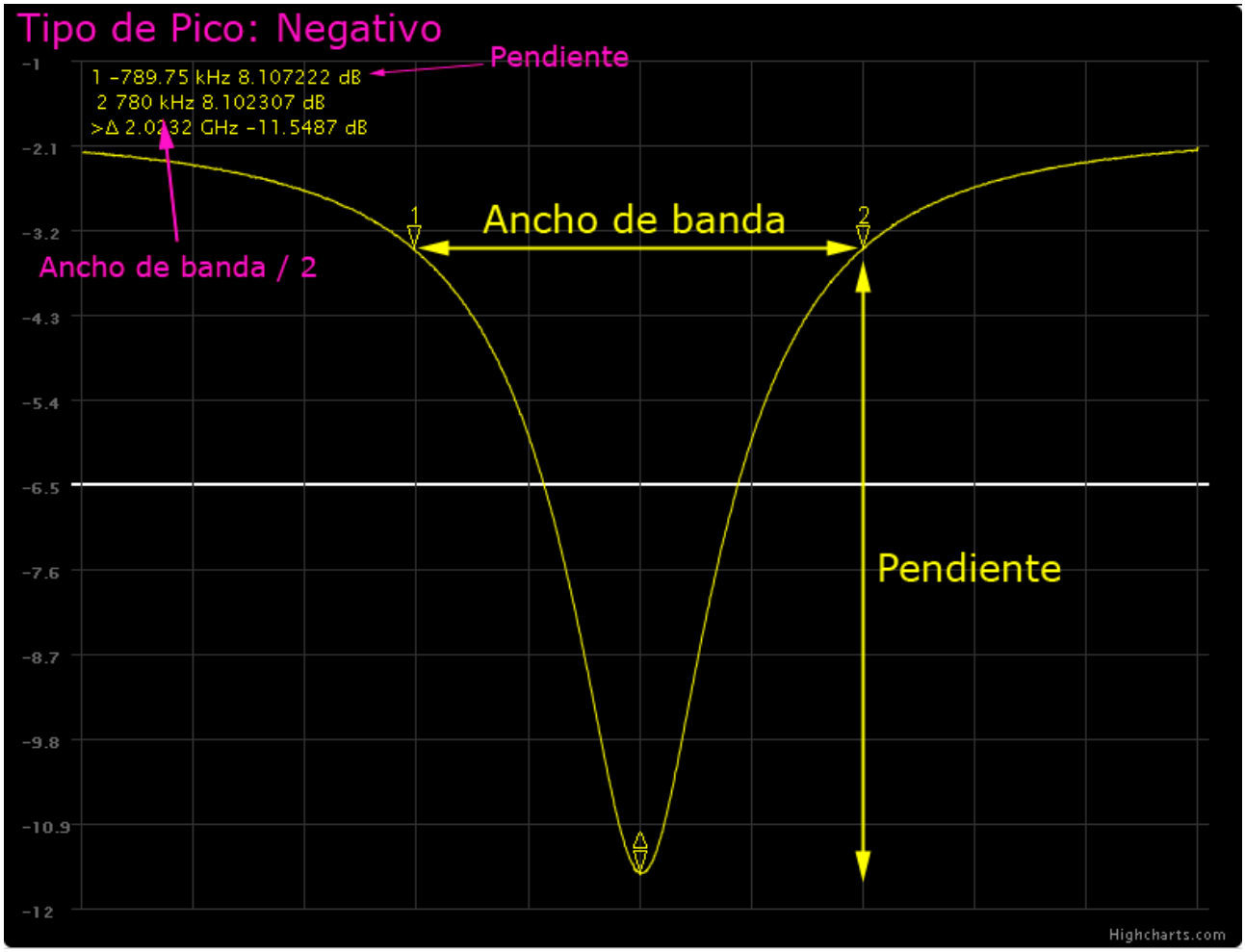


Ilustración 25: Pico negativo con los parámetros utilizados para buscar picos.

El funcionamiento de la función interna para buscar un pico es el siguiente:

Si estamos en la función “Multipico” el punto inicial será 0, si estamos en “Buscar pico” será el punto donde esté situado el marcador.

Se tomará un margen de puntos basado en el ancho de banda. La mitad del ancho de banda se tomará en los valores negativos con respecto al punto seleccionado y la otra mitad en los valores positivos. En caso de que se llegue al límite de puntos o a valores negativos, se escogerán tantos puntos como sea posible sin sobrepasar estos límites. En la ilustración 26 se muestra el caso normal, y en las ilustraciones 27 y 28 se representan los casos límites.

Se analizará en el trozo de traza en el que nos encontramos, localizando su máximo y su mínimo. Si el punto seleccionado es un máximo o un mínimo, se comprobará si cumple con la especificación como se muestra en la ilustración 29.

Si es un mínimo y está la opción para buscar picos negativos activada, se buscarán los máximos locales a ambos lados (izquierdo y derecho). Y como podemos comprobar reflejado ilustración 29. se restará cada valor al del punto, si es mayor que la pendiente. En ambos casos se habrá encontrado un pico que se corresponde con las especificaciones.

Si es un máximo y se han requerido picos positivos en la búsqueda. Se realizará la misma operación que para el pico anterior, pero utilizando mínimos locales en vez de máximos.

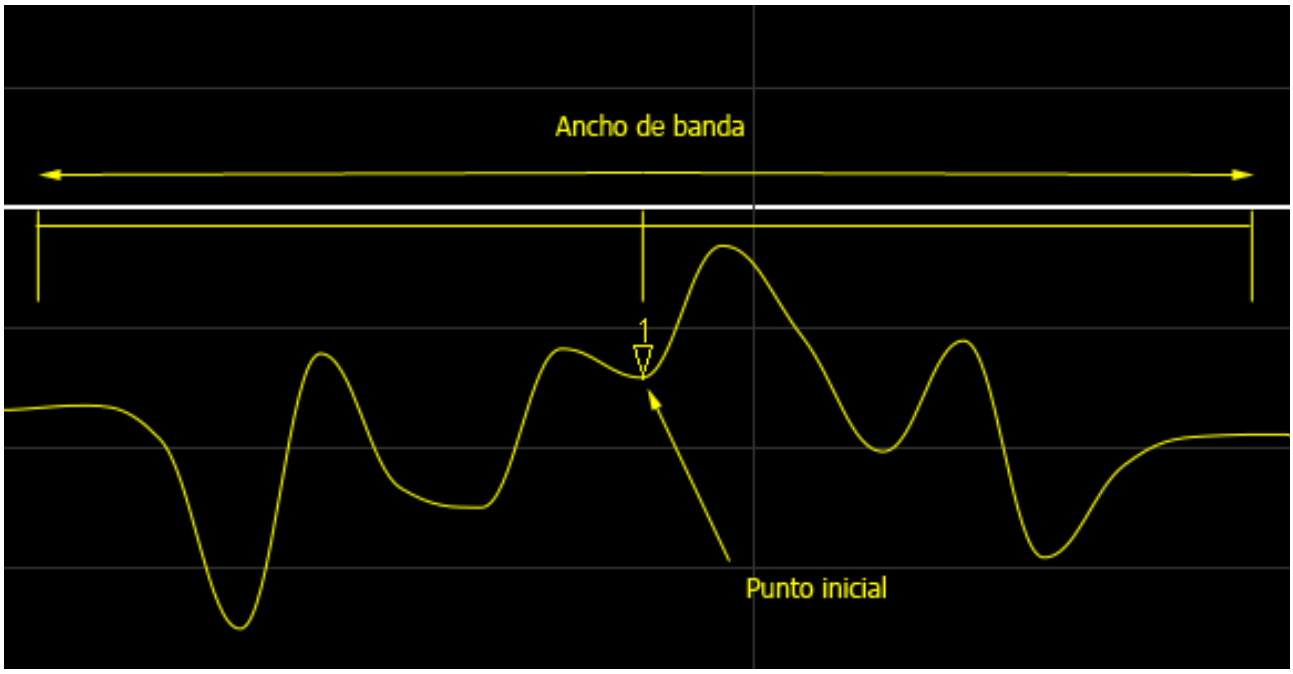


Ilustración 26: Caso normal.

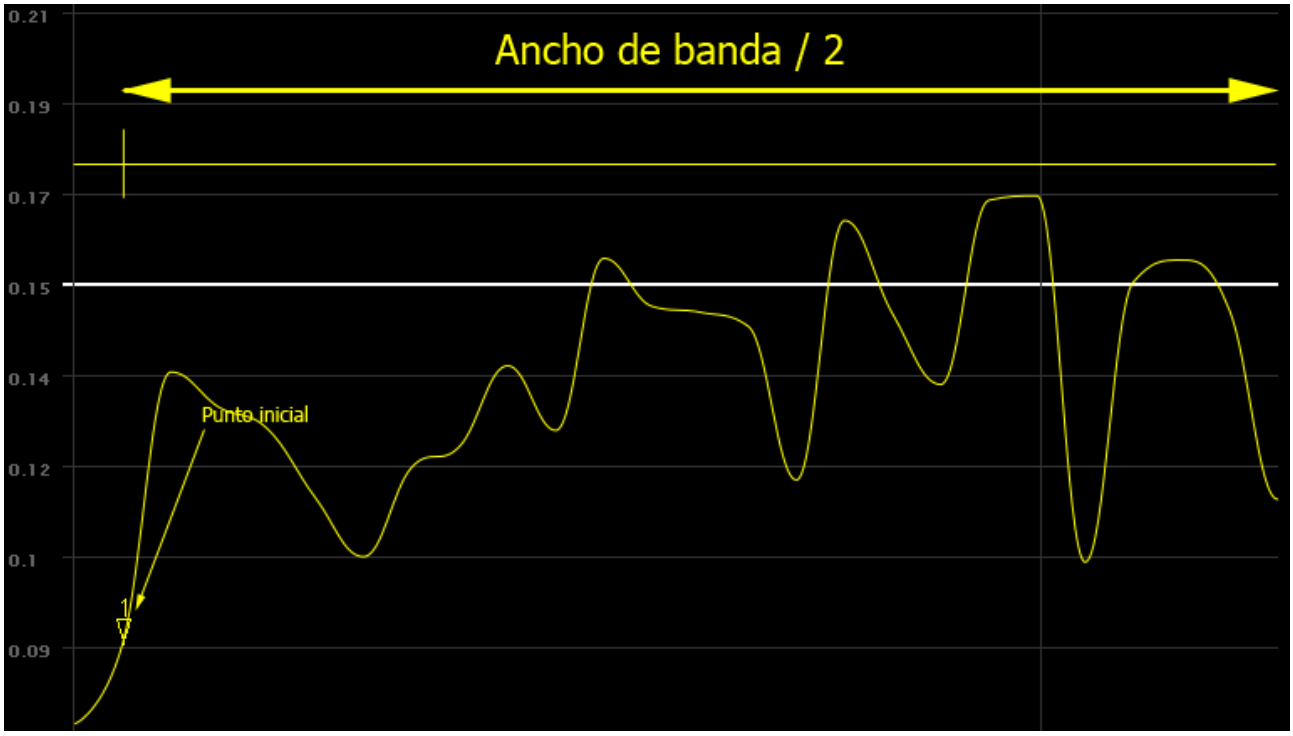


Ilustración 27: Caso limitado por la izquierda.

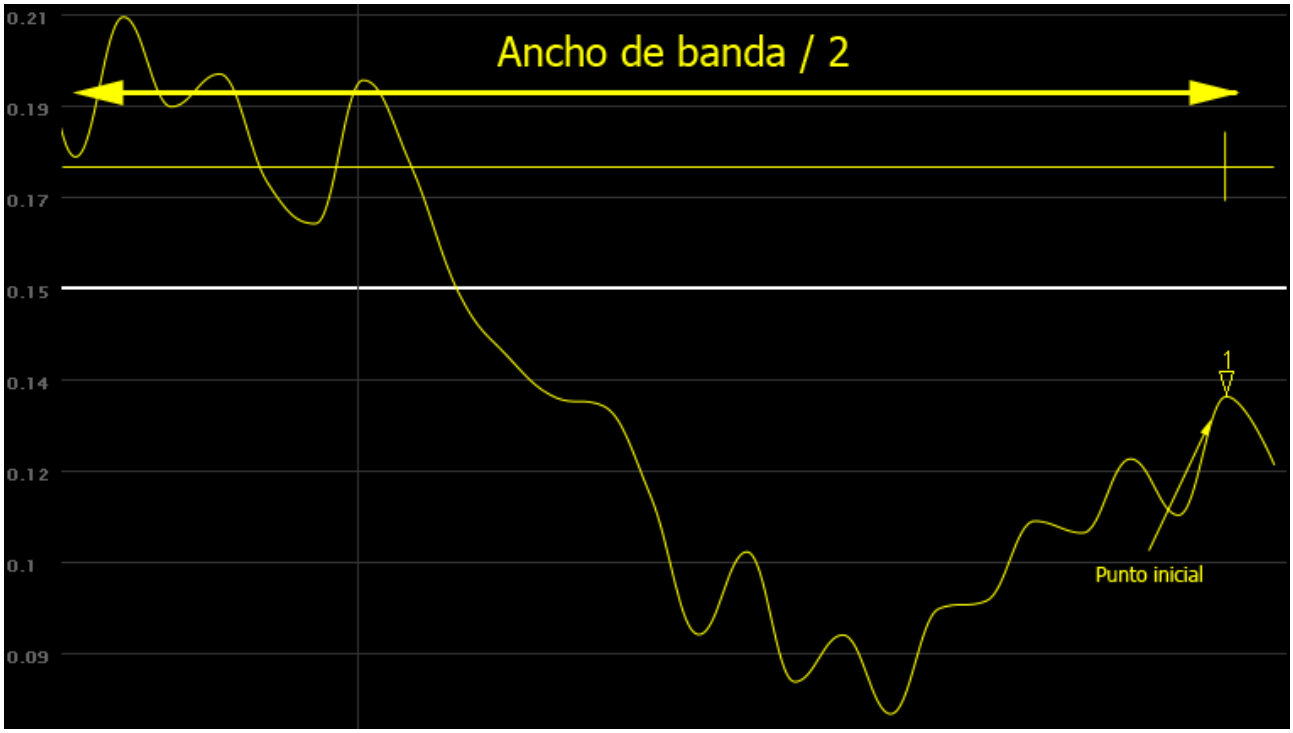


Ilustración 28: Caso limitado por la derecha.

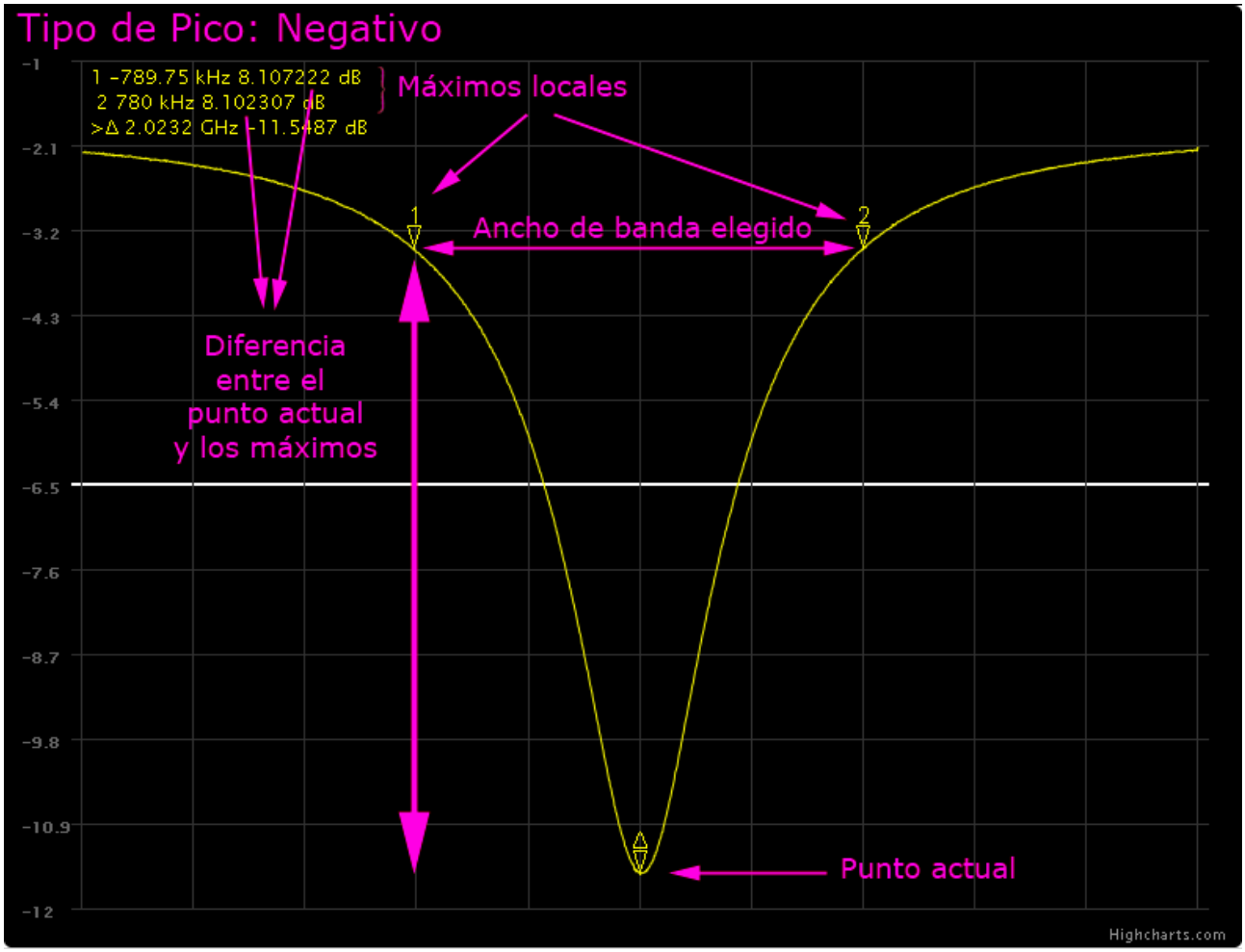


Ilustración 29: Especificaciones del pico.

Si no se ha encontrado un pico, o estamos en la función "Multipico" pasaremos al siguiente punto. El siguiente punto se escogerá entre los puntos de valor superior dentro del ancho de banda, con el siguiente criterio:

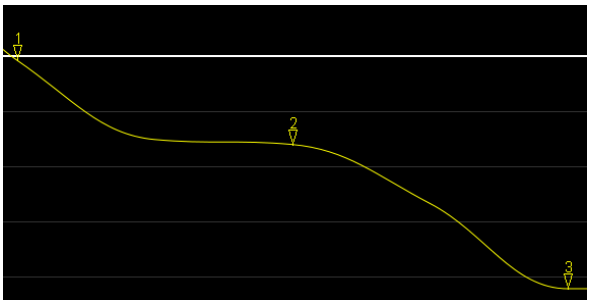


Ilustración 30: Minimo local a la izquierda.

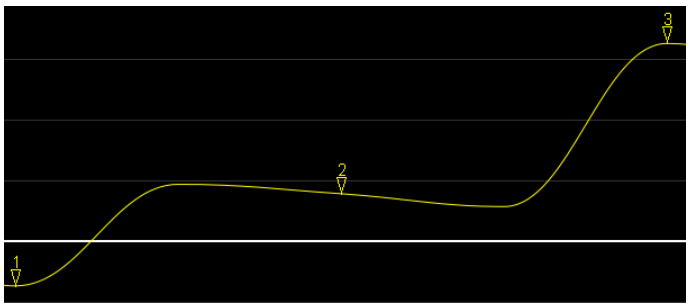
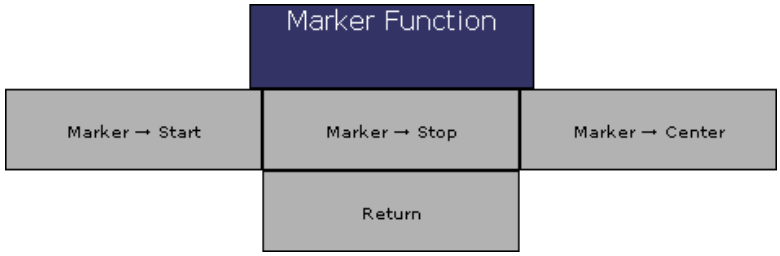


Ilustración 31: Minimo local a la derecha.

- Si el máximo o el mínimo local de ese trozo se encuentra en los valores positivos de la derecha, saltamos a ese punto. Marcador 3 de las ilustraciones 30 y 31 lo indica.

- Si ambos se encuentran a la izquierda saltamos la mitad del ancho de banda.

2.VI.3.v Marker Function



Dibujo 5: Menú Function.

Esté menú utiliza el marcador activo, para cambiar las frecuencias de inicio, fin o central.

2.VI.4 Actualización de la gráfica

A la hora de plantearse una actualización continua de la traza desde el navegador web, aparece el problema de que HTTP es un protocolo sin estado. A causa de esto, el servidor web no puede notificar al navegador que existe una nueva traza. En un inicio, se esperaba a que la traza estuviera lista después de cada orden, ya que aún no estaban implementadas las interrupciones. Y tampoco tenía forma el analizador de notificar al servidor VISA de que se había acabado la traza, sin bloquear a este último. Para los modos de disparo “hold” , “simple” y “con cambios” esto no supone ningún problema. Ya que si se produce una traza, es porque el usuario la ha ordenado. El único inconveniente era el retraso para tiempos de traza grandes. Pero en el modo continuo se requiere que la traza se actualice constantemente, sin ninguna orden. Este modo era irrealizable de una manera eficiente.

Con la llegada de las interrupciones, el modo traza continua se podía implementar, sin evitar un consumo mayor de recursos; pero seguía sin poderse notificar al navegador si existía una nueva traza.

Para solucionarlo existen 2 tipos de soluciones:

1. Conexión permanente: Cada navegador inicializaría un conexión TCP con el servidor, y al recibir una traza nueva se le notificaría por ese canal. El problema que crea, es que no podríamos utilizar el Servidor Web para establecer esta conexión. Tendríamos que tener un servidor adicional, o permitir el acceso del navegador a “Enviadatos”. La primera opción implica menos recursos en el cliente y la segunda crea una manera de saltarse la barrera de seguridad creada, para que no se pueda acceder al servidor de los analizadores. También genera problemas, porque evita que el Servidor PHP actualice su valor de “haytraza”, provocando que la próxima conexión que realice el cliente al Servidor PHP, obligue a que se tenga que descargar la traza aunque ya se tenga almacenada en el navegador.

2. Conexión a intervalos de tiempo regulares: Cada cierto tiempo fijado, el navegador

utilizando código “Javascript” le pregunta a la página web si existe nueva traza. Y si la hay se la descarga. A cambio por cada cliente que tengamos en activo con traza continua, tendremos un acceso cada segundo al Servidor Web. Podría provocar problemas de carga si hubiera muchos clientes conectados.

Se ha escogido en este proyecto la segunda solución. El intervalo de actualización es de 1 segundo. El problema de sobrecarga se estudiará más adelante, pero es difícil que se llegue a sobrecargar el servidor web con todos los ordenadores que se disponen en el laboratorio. “Lighttpd” está preparado para soportar un carga de clientes mayor.

Para preguntar al Servidor Web se utiliza el valor de menú especial 0 y de opción 4. Con esto se le indica al servidor que debe de enviar los de la traza. El formato de los datos es:

<¿ Nueva traza ?><Estado de la traza>(Campos opcionales)<frecuencia de inicio><frecuencia final><Parametro S medido><Estado de la calibración><formato de la traza><número de puntos><Traza>

Los campos están separados con el carácter “\n”. Los datos están escritos en ASCII, utilizando para las frecuencias, el formato especificado anteriormente para el envío de frecuencias entre el Servidor VISA y el Servidor Web. Como ya no estamos utilizando HTML, sino solo aprovechando la conexión abierta en el puerto 80 por el servidor, podríamos enviar los datos en un formato más adecuado para ser enviado, si se viera que el servidor tiene problemas de carga.

El primer valor es un parámetro “boolean” que indica si la traza es nueva. Ésto se puede hacer porque el identificador de traza está guardado en una variable de sesión PHP, y el almacenamiento de la traza en el navegador, se guarda en el tipo de almacenamiento que se corresponde con las variables de sesión PHP. Con lo cual si el servidor PHP tiene activado el valor de que esa traza es nueva, quiere decir que no ha sido descargada por el navegador correspondiente a esa sesión. Una vez descargada esta variable su valor será falso y ya habrá sido recibido por el navegador. Estaremos seguros de que se ha recibido porque todo se realiza bajo protocolo TCP que garantiza que los datos han sido enviados y recibidos.

El siguiente valor es estado de traza, que se corresponde con los valores que indican 0, si la traza no es válida, 1 si es valida y no va a ser actualizada y 2 si es válida y nos encontramos en modo continuo. Esté valor se utiliza para establecer la actualización de traza en los casos 0 y 2. Por eso siempre se envía, para actualizar a inválido el estado de la traza en modo continuo y que el resto de usuarios sepan que la traza actual que poseen va a ser cambiada.

El resto de campos se envían solo si la traza de la que se dispone es nueva. Están todos asociados a la traza. “Estado de la calibración” indica si la traza actual procede de un analizador calibrado.

2.VII Comunicación Servidor Web – Navegador web

El navegador se comunica con el Servidor Web mediante Http. Se realiza mediante un proceso de petición-respuesta al puerto TCP 80 del Servidor Web.

Temporalmente lo único que importa, es que el envío de datos de la traza entre el código Javascript y el Servidor Web sea los más rápido posible, para así en modo traza continua dar al usuario sensación de continuidad. La carga de la página también es deseable que no sea excesivamente lenta, porque sino en apariencia, parecerá que el proyecto no responde.

La respuesta de ambos tiempos, depende de lo valores de conexión entre los 2 servidores visto anteriormente, el tiempo que tarde el script en PHP en preparar la respuesta, el volumen de datos de ésta y los archivos adicionales que se descarguen, tales como imágenes, código css, extensiones de javascript...

Al igual que antes, el peor tiempo se dará cuando los datos que hemos de transmitir sean más grandes. Los tiempos medidos para una carta de Smith de 1601 puntos. Oscilan entre los 200 ms y 140 ms.

Mientras que para la carga de la página su valor será de alrededor de 1,1 segundos.

3. Implementación actual del proyecto

3.I Estudio de tiempos del sistema actual

No podemos realizar un estudio completo por teoría de colas del sistema; pero si que podemos aplicar algunos conceptos para poder predecir los casos en los que el sistema se saturará provocando un mal funcionamiento.

El diagrama del bloques del sistema sería el siguiente:

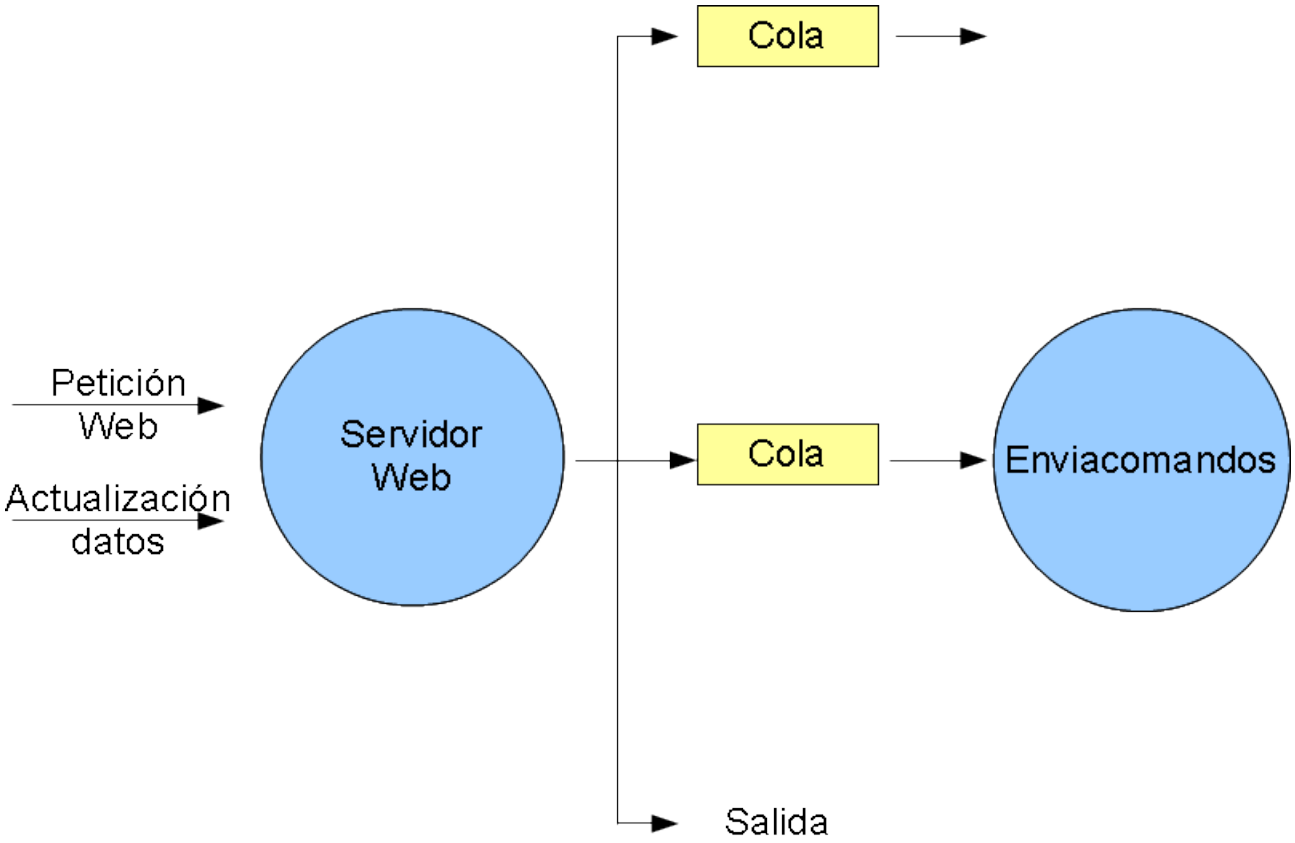


Ilustración 32: Simplificación del sistema por teoría de colas.

La llegada de clientes se puede producir en un principio cuando se quiere realizar una acción sobre el analizador u obtener información. Pero en modo continuo tenemos una llegada constante de clientes, al igual que una comunicación constante con el analizador, pudiendo ambos llegar a provocar un bloqueo en el servidor.

Esto añadido a que no todos los clientes que acceden al servidor web pasan a las colas y que cada analizador tiene sus propios hilos con sus propias colas, imposibilita el estudio por teorías de colas. En vez de esto analizaremos los casos limites de 2 partes, la llegada al servidor Web y la

comunicación del Servidor Web con los hilos correspondientes.

Podríamos separar una tercera parte que incluyera la comunicación del analizador con el servidor. Pero como ya vimos en el apartado primero de tiempos, se ha aplicado como solución el utilizar para el 8714ET el GPIB solo para interrumpir. Si se necesitara implementar más de un analizador por el mismo GPIB habría que estudiar los tiempos para que la transmisión de todas las trazas consecutivas no demorara más de un segundo.

3.I.1 Servidor Web

El servidor Web es multitarea y atiende a la vez a la petición de todos los clientes. Se podría producir un bloqueo del servidor si recibiera un exceso de peticiones a la vez, este límite depende del ordenador en el que esté instalado y las características del servidor usado. En el caso del proyecto esta circunstancia tiene un valor muy alto, ya que llegar a este límite sería muy alto.

Pero este límite se puede reducir si tenemos en cuenta 2 circunstancias, en las que el servidor se quedará bloqueado con usuario activo bloqueado:

- Acceso a la variable compartida de calibración.
- Acceso a uno de los 2 hilos de cada analizador.

En ambos casos se puede producir la llegada al límite si el usuario pulsara seguido en el botón del menú correspondiente al ver que el servidor no da respuesta. Sin embargo como se verá más adelante en líneas futuras, este posible error se puede arreglar, con un control de los comandos enviados por cada usuario.

Si aplicamos la corrección pertinente explicada arriba, obtendríamos que el límite de usuarios activo del servidor en el peor de los casos, es igual al límite de usuarios soportado por el servidor. Teniendo en cuenta que se ha producido un bloqueo y todos están esperando.

Temporalmente este bloqueo no se puede estudiar, ya que dependiendo del analizador y el comando enviado, los bloqueos citados anteriormente pueden llegar a ser alrededor de 0,2 segundos para un comando rápido hasta 1 minuto o mayor para guardar una calibración en el analizador 8714B. Solo se puede suponer el caso peor de todos, que sería un tiempo de bloqueo lo suficientemente alto como para saturar el servidor, llegando al límite expuesto.

3.I.2 Servidor Visa

En el Servidor Visa es más fácil que se produzca un bloqueo. Como ya se vió en el apartado de tiempos de dicho servidor el único caso de bloqueo se encuentra si nos encontramos en modo continuo. En el resto de modos de disparo, se saturaría el servidor Web, puesto que enviacomandos cierra el acceso a más usuarios mientras se produce un comando.

En el modo de disparo continuo, el colapso llega porque cada segundo tenemos una petición por

cada usuario activo. En 0,7 segundos, dejando un margen de 0,3 para obtener y almacenar la traza del analizador, debemos de haber dado servicio a todos los usuarios activos.

No tenemos en cuenta el posible envío de comandos, ya que cierra la actualización de traza y por tanto los usuarios que lleguen o bien se quedaran en cola esperando a que acabe el comando actual, o si ha acabado el comando actual se notificará que se está realizando una traza y no se enviará nada.

Con lo cual la condición límite que tenemos es:

$$U \cdot t < 0{,}7$$

U: número de usuario (1)

t: tiempo de envío de una traza

El tiempo de envío de traza lo hemos obtenido en el apartado de tiempos. Así que podemos despejar el número de usuarios.

Con los tiempos medidos en la implementación actual del proyecto tenemos que en el peor de los casos, con una traza de 1601 puntos en formato polar es de: 94,6 ms con un valor máximo de 100,4 ms.

Despejando en la fórmula obtendremos que el número máximo de usuarios límite son de 7 por analizador.

Cumple con las prestaciones indicadas en un principio del sistema, puesto que, una de ellas era que solo un grupo de alumnos iba a manejar el analizador. Siendo los grupos de 3 o 4 alumnos, 3 ordenadores por puesto, con lo cual el numero máximo de usuarios es 3, en el caso de cada uno quiera ver el analizador desde un ordenador.

Sin embargo es una característica muy baja, para aumentar este número se podrían aplicar correcciones como almacenar la traza en formato carácter, y realizar su conversión en el servidor PHP, reduciendo el tiempo de envío y de preparación de la respuesta.

Por parte de la preparación de la respuesta, este tiempo se reduciría cercano a 0, en el peor de los casos 1 ms. Con lo cual rebajaríamos el tiempo de manera cercana a los 20 ms. Si tenemos en cuenta que también se reduciría el volumen de datos al ser enviados en formato carácter, el tiempo de envío se reduciría notablemente también.

El tiempo de envío aumentará según los parámetros adicionales que se añadan a la traza, vamos a seguir suponiendo que daría el mismo resultado que antes, aunque sería ligeramente menor debido a que se envía menos volumen de datos. Al ser este tiempo de 20 ms, con esta mejora el número de

usuarios se multiplicaría por 5, alcanzando los 35 usuarios.

Otra mejora que se podría realizar es la de almacenar la traza en el servidor PHP, y que fuera común para todos los usuarios. En este caso el límite de usuarios sería el que tuviera PHP, teniendo en cuenta que habría un acceso compartido y generaría problemas de bloqueo. Teniendo que buscar un modo de que el acceso de PHP a esa variable compartida sea mínimo. O se encontraría con un problema similar actual.

La realización de ambas soluciones están detalladas en el punto de líneas futuras.

4. Conclusiones y líneas futuras

4.I Líneas futuras

Al ser este proyecto amplio y tener varios servidores, existen 2 líneas separadas una de otra que consiste en: la mejora de cada uno de los servidores, y otras 2 más consistentes en mejorar la comunicación entre ambos y aumentar las funciones.

Con esto se quiere indicar que la primera y la segunda de estas ramas se podrían realizar por separado sin afectar la una a la otra, puesto que no modifican la comunicación entre ambos servidores, tal y como está planteada en el proyecto.

Las líneas futuras presentes en este proyecto son:

4.I.1 Aumento en la seguridad

Actualmente el uso del proyecto está restringido al laboratorio y a sus ordenadores por razones de seguridad.

Al principio de esta memoria se hablaba de una serie de premisas sobre las que se había creado el proyecto. De esas premisas, la segunda supuesta, eliminaba el control de usuarios. Al quitar el control de usuarios, no se controlaba el acceso a los servidores y por tanto se obviaba la seguridad.

Esta línea trataría de proporcionar el control de usuario que falta, añadiendo seguridad también frente a posibles ataques. El proyecto se ha construido pensando que los usuarios van a ser responsables y nadie va a atacar el servidor. En el mundo real esta premisa no es verdad y por tanto debe de ser evitada.

Para arreglarlo se recomendaría, a parte de las medidas de seguridad normales como tener actualizado todo, crear varios usuarios en el analizador, teniendo cada uno distintos accesos a las funciones de este. Se deja al criterio del alumno que continúe los permisos que tenga cada uno, y cuantos de éstos se crearan.

Sería interesante poder utilizar autenticación utilizada en la red de la universidad. Y en un futuro poder integrar la página web con poliformat haciendo más cómodo a los profesores el manejo de los alumnos que pueden o no acceder y cuando.

Los controles que serían deseables establecer son:

- Se debería de controlar que usuarios pueden acceder y cuando, para así por ejemplo, poder evitar que en mitad de un examen, un alumno pueda manejar el analizador de otro desde fuera.

- Si hiciera falta restringir el acceso a algunas funciones peligrosas del analizador, como

puede ser: la potencia de señal, configuraciónes especiales, etc...

- Controlar el acceso de los comandos enviados por el usuario, pudiendo el usuario administrador del sistema (Los profesores) comprobar los comandos, que ha utilizado el alumno y facilitar la enseñanza del manejo del analizador.

- Permitir que un usuario solo pueda enviar un comando a la vez. La solución más simple y sencilla sería utilizar las variables de sesión PHP para indicar si se está enviando un comando. Recordemos que hasta que no se completa el comando, no se envía la página, y por tanto no se cierra la conexión. Así si el mismo usuario accede por darle a refrescar en la página por ejemplo, y no se ha completado el comando anterior, se evita enviarlo.

- Cifrar la comunicación, para evitar suplantación de la identidad de los usuarios.

4.I.2 Añadir nuevos analizadores

En el laboratorio queda un analizador que no está soportado por el proyecto. Es el analizador sobre el que está basado el menú principal, con lo cual se decidió no implementarlo de momento ya que su uso era el mismo.

De igual forma, se pueden implementar otro analizador nuevo en el caso de que se adquiriera. El proyecto está ideado de forma que solo haya que añadir la clase necesaria al código para implementarlo. Y dentro de dicha clase crear cada una de las funciones de comunicación indicadas en el proyecto.

También convendría separar el código fuente en un archivo nuevo e importarlo en la función principal del programa, esto organizaría el código y facilitaría su depuración. No se ha realizado tal separación porque solo se han implementado 2 analizadores. Pero para un número mayor es aconsejable realizarlo.

4.I.3 Aumento de funciones

Durante el proyecto se han quedado funciones de los analizadores sin utilizar, puesto que la intención del proyecto era crear un servidor con las funciones básicas y normales utilizadas durante las clases en el laboratorio. Por ello han quedado funciones sin resolver como pueden son:

- Menú de Power.

- Uso de varias trazas a la vez.

- Modificación de los kit's de calibración.

- Etc...

Su implementación requeriría de la activación del menú correspondiente en el servidor PHP y la creación del método o función necesaria para mandar las ordenes al analizador. En casos como el menú Power que se encuentra dentro de otro menú, la función ya ha sido creada y sería necesario implementar los comandos adicionales.

También se podría aprovechar las librerías gráficas para añadir alguna función de representación que no estuviera en el analizador. Como ya se ha hecho con el zoom, o la visualización de puntos al pasar el ratón sobre la traza. También se pueden mejorar funciones que no vayan bien en el analizador, como se ha hecho con la función buscar picos.

Otra parte a tener en cuenta es que actualmente no se tiene ningún control sobre el bus GPIB. Si se conectara más de un dispositivo por el mismo bus, el programa fallaría y se crearían errores. Para solucionarlo habría que implementar un sistema de detección del número de bus e instrumentos conectados, puesto que este control no lo establecen las librerías visa.

4.I.4 Posibles optimizaciones

Estás mejoras estarían encaminadas a aumentar las respuesta del servidor, su velocidad y el número de usuarios que pueden utilizarlo a la vez. En el apartado de tiempos ya se ha hablado de 2 mejoras que se podrían realizar:

- Cambiar el formato de envío de traza entre los servidores al original del analizador. Dejando la traza en formato IEEE 754, utilizando para almacenarla un vector de caracteres, se optimizaría el envío de la traza consiguiendo una velocidad mayor. También se evitaría el tiempo que se tarda en preparar la traza para enviar, consiguiendo ahorrar 50 ms por cada petición enviada. La conversión de la traza a un formato que entienda el plugin gráfico utilizado sería recomendable hacerla en el navegador del cliente. Aunque aumentaría un poco el uso de recursos en el navegador, se conseguiría que la página web fuera más ligera de enviar.

- Hacer que la traza de cada analizador, se almacenara en el servidor PHP para todos los usuarios de ese analizador. Evitando así el envío de la traza una vez por cada usuario.

4.II Conclusiones

Recordemos que el objetivo de este proyecto era: “dotar de una interfaz común a todos los analizadores, que permita emplear eficazmente el tiempo de laboratorio en medir y no en localizar opciones de distintos menús”.

Este objetivo se ha cumplido, ya que, ya sea de manera local o remota todos los analizadores disponen de la misma interfaz. También se ha reducido el tiempo de las prácticas para guardar trazas. En los analizadores 8714 para guardar una traza, se necesitaba hacer uso de la disquetera, y luego utilizarlo en un ordenador que tuviera un lector de disquete. Actualmente de manera remota las trazas se guardan a través del navegador del dispositivo que se esté usando. Permitiendo mayor agilidad para realizar este tipo de medidas.

4.II Conclusiones

También se ha logrado implementar un acceso a los analizadores desde cualquier dispositivo que soporte un navegador web. Esta característica amplia el uso de los analizadores e incluso abre la puerta a un futura implementación, en la que se pueda utilizar el analizador desde cualquier lugar.

Por supuesto el programa no está finalizado, aunque se hayan cumplido las metas propuestas por este proyecto. Por ello en su día se limitó el uso y funciones que eran deseables que realizara. En un futuro como ya se ha visto en el apartado de líneas futuras se le podría dotar de nuevas funciones. También mejorar y optimizarlo para obtener un comportamiento idéntico al que tendríamos delante del analizador e incluso mejor al permitir realizar nuevas opciones no implementadas.

Un ejemplo de esto es la función buscar picos o el estado de calibración en el 8714B. El primero no funciona como debe en los analizadores y el segundo no estaba implementado de manera remota. Pero ambas opciones se han podido implementar sin problemas.

Otra de las conclusiones que podemos ver es, que gracias a este proyecto, se podrá hacer un mejor uso de los analizadores. Siempre ha habido problemas con ellos, ya que si se rompe una parte del analizador, como pueden ser los botones, cambiarlos requiere un coste elevado. De igual manera como ya ha pasado, si se rompe la disquetera del analizador, imposibilita su uso en clase. Con este proyecto se han evitado ambos incidentes. Debido a que el analizador ya no se maneja en local y las trazas se obtienen de a través del navegador.

Por último resaltar que ya no es necesario estar delante del analizador para obtener las medidas. Esto ha demostrado ser especialmente eficaz en clases, en la que varios grupos de alumnos han de realizar un medida breve. De otro modo hubiera implicado una complicación mayor, al tener cada analizador sus propios menús. De igual forma habría habido problemas para guardar las trazas en un disquete.

5. Bibliografía

Las fuentes bibliográficas utilizadas para realizar esta memoria y este proyecto han sido:

[1] "Librería pthreads información y descarga", http://www.sourceware.org/pthreads-win32/

[2] "Thread (computing)", https://en.wikipedia.org/wiki/Thread_%28computing%29

[3] Blaise Barney, Lawrence Livermore National Laboratory. "POSIX Threads Programming" https://computing.llnl.gov/tutorials/pthreads/

[4] "Servidor Web lighttpd", http://www.lighttpd.net/

[5] "Plugin gráfico highcharts", http://www.highcharts.com/

[6] "Licencia BSD", http://es.wikipedia.org/wiki/Licencia_BSD

[7] "PHP", http://php.net/

[8] IBIS Open Forum April 24, 2009. "Touchstone® File Format Specification", http://www.eda.org/ibis/touchstone_ver2.0/touchstone_ver2_0.pdf

[9] Agilent SICL User's Guide for IO Libraries Suite 16.3, http://cp.literature.agilent.com/litweb/pdf/5991-1389EN.pdf

[10] Documentación sobre C++, http://www.cplusplus.com/reference/

6. Anexo

6.I Manual del programador

El manual del programador ha sido concebido como un documento entero pensado para ser manejado de manera independiente a esta memoria. Se incluye como anexo a esta memoria en otro documento porque está relacionado con el proyecto. En el se profundiza en el código de los servidores, su funcionamiento y se explica para que se ha creado y utilizado cada aspecto de éste.