INSTITUTO POLITECNICO NACIONAL

CENTRO DE INVESTIGACION EN COMPUTACION
MAESTRIA EN CIENCIAS DE LA COMPUTACION

INSTRUMENTO VIRTUAL PARA LA MEDICION Y EVALUACION DE UN PATRON DE RADIACION DE ANTENA

T E S I S

QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A :
ING. VICTOR MANUEL MEDINA MENESES

DIRECTOR DE TESIS: DR. SERGIO SUAREZ GUERRA

MEXICO, D.F. JUNIO 2004

DEDICATORIAS

Para mi amada esposa Abigail, gracias por todo el apoyo que medio para poder lograr este proyecto y por seguir siendo...

Gracias.

Para mis hijas Nayibi, Andrea e Ingrid por todo el tiempo que les robe, y por la motivación que me hicieron sentir al querer darle ejemplo. Gracias.

A mis padres Samuel y Esperanza por haberme creado con el afán de aprender

A mis Hermanos Alejandro y Gilberto por la apuesta t…

AGRADECIMIENTOS

Gracias Dios por haberme dado la fuerza para concluir este trabajo.

Al Dr. Sergio Suárez Guerra, quien no sólo me dirigió en este trabajo de tesis sino también me brindo su apoyo y amistad.

A la comisión revisora por sus comentarios que contribuyeron a enriquecer este trabajo.

A todos los profesores del CIC por la transmisión de sus conocimientos.

Al IPN, al CIC y sus trabajadores por darme la oportunidad de ser parte de la institución.

A los compañeros del centro de control de satélites por el apoyo que me dieron tanto material como moralmente en la elaboración de este trabajo.

INDICE

RESUMEN

Este trabajo se da una solución al problema de la adquisición y evaluación del patrón de radiación de una antena de manera automática por medio del desarrollo de un instrumento virtual.

Se desarrolló un sistema cliente servidor donde la comunicación se hace con paquetes “TCP” por lo que el cliente puede ubicarse en sitios remotos a través de una “WAN” o “INTERNET”.

Se desarrolló un cliente que encapsula la “GUI” y proporciona las herramientas para evaluar el patrón de radiación y generar reportes. El cliente se desarrolló con lenguaje “JAVA” por lo que puede correr en cualquier estación de trabajo que tenga una máquina virtual de “JAVA”. Además se incluyó la capacidad de evaluar el porcentaje de área del patrón de radiación que sobrepasa a la curva de evaluación.

Se desarrolló un servidor que encapsula el control del analizador de espectros y puede adquirir el patrón desde un trazo del analizador (forma tradicional) o puede hacer un muestreo del valor de potencia que tiene el analizador para generar el trazo del patrón.

El servidor se ejecuta en el sistema operativo RT-Linux y hace uso de las funciones de tiempo real duro para garantizar que las muestras de potencia, tomadas para generar la gráfica del patrón de radiación, estén igualmente espaciadas y el patrón adquirido no se deforme.

Se generó un sencillo protocolo de comunicación entre cliente y servidor basado en el “Standard Commands for Programming Instrumentation (SCPI)”. Por lo que si se desea controlar otro analizador de espectros se puede generar otro programa servidor. Sin modificar el cliente que es la parte más grande del software.

Con este trabajo se da cumplimiento a una necesidad de la empresa SATMEX para realizar la evaluación de los patrones de radiación de las antenas satelitales y garantizar que los usuarios no se interfieran en sus comunicaciones.

ABSTRACT

This work gives a solution to the automatic antenna pattern acquisition and evaluation, through developing a virtual instrument

A Client-Server system was development where the communications is made with TCP packets, for this reason the client could be in remote site through a WAN or Internet.

A client that encapsulates GUI was development, and the client gives the tools to evaluate the antenna pattern and produce print reports. The client was developed in Java language, that's because it can be run in any work station with a Java virtual machine. The capability to evaluate the percentage of area above the evaluation curve was add.

A server was development. This server encapsulates the spectrum analyzer control and could acquire the pattern from a spectrum analyzer trace (traditional way) o sampling the power from the spectrum analyzer to generate the trace.

The server is running in an RT-Linux operating system and makes use of the hard real time functions to ensure that the power samplings, taking to generate the antenna pattern, where equally distant in time pattern where not deform.

A simple communications protocol was design between the client and server; it was based in the Standard Commands for Programming Instrumentation (SCPI). That's make possible make another server program if you wish to control another spectrum analyzer with out have to modify the client, witch is the bigger part of the software system.

This work gives a satisfaction to a SATMEX necessity to evaluate antenna patterns to guarantee that the users don't interference with each others in theirs communications.

INDICE DE FIGURAS

INDICE DE TABLAS

GLOSARIO

ANSI	"American National Standards Institute". Instituto de Estándares Nacionales Americanos.
AP	"Antenna Pattern". Patrón de radiación de antena.
API	"Application Interface". Interfaz de aplicación, conjunto de funciones y la descripción de cómo usarlas que se ofrece a los programadores.
CALLBACK	Es el mecanismo mediante el cual a algún objeto se le provee de cierta información que le permite llamar al objeto original.
CO-POL	"Co-Polarization", Polarización deseable.
CR	"Carry Return". Retorno de carro.
CROSS-POL	"CROOS-Polarization". Polarización cruzada, no deseable ya que puede causar interferencia.
CW	"Continuous Wave", Portadora limpia.
DMA	"Direct Memory Access". Acceso directo a memoria. Modo en que algunos dispositivos colocan sus datos directamente en la memoria RAM sin pasar por el microprocesador.
GB	"Giga Byte". 1024*1024*1024 bytes.
GPIB	"General Purpose Interface Bus". Bus de interfaz de propósito general, inicialmente desarrollado por HP como HPIB, también conocido como el estándar IEEE Standard 488.1.
GUI	"Graphical User Interface". Interfaz gráfica de usuario, en esta interfaz la forma de interactuar es a través de iconos, ventanas y dispositivos apuntadores como el ratón.
HOST	Computadora anfitriona.

HP	"Hewlett Packard".
HP-UX	Unix de Hewlett Packard.
IEEE	"Institute of Electrical and Electronics Engineers". Instituto de Ingenieros Eléctricos y Electrónicos.
IP	"Internet Protocols". Conjunto de protocolos responsables de mover los paquetes de datos de un nodo a otro nodo.
ISA	“Industry Standard Architecture”. Bus estándar de 16 bits para la transferencia de datos entre el microprocesador y sus dispositivos.
ITU	"International Telecommunications Union".
LF	"Line Feed", Alimentación de línea.
MB	"Mega Byte". 1024*1024 bytes.
MODAL	Tipo de ventana de dialogo que bloquea la entrada en todas las ventanas superiores en el contexto de una aplicación. El tipo contrario es "modeless".
PC	"Personal Computer". Computadora personal.
PCI	“Peripheral Component Interconnect”. Bus estándar de 32 o 64 bits para la transferencia de datos entre el microprocesador y sus dispositivos.
RAM	"Random Access Memory". Memoria de acceso aleatorio.
ROM	"Read Only Memory". Memoria de solo lectura.
RS232	Interfaz serial de comunicación de datos.
SCPI	“Standard Commands for Programming Instrumentation". Comandos estándar para la programación de instrumentación, es una adición a la especificación IEEE 488.2. La especificación IEEE 488.2 define comandos generales mientras que SCPI proporciona comandos requeridos para la operación de ciertos tipos de instrumentos.

SL	Sistema en línea.
SO	Sistema operativo.
SOTR	Sistema operativo de tiempo real
STR	Sistema de tiempo real.
TCP	"Transmition Control Protocol". Protocolo de Control de Transmisión, es responsable de verificar la entrega correcta de datos entre el cliente y el servidor, agrega soporte para detección de errores y pérdida de datos y para iniciar retransmisiones hasta que los datos sean entregados correctamente.
THREAD	Hilo de ejecución de un proceso.
UIT	“Unión Internacional de Telecomunicaciones”.
UML	"Unified Modeling Language". Lenguaje unificado de modelado.
US FCC	"United States Federal Communications Commission".
WAN	"Wide Area Network". Red de área amplia.
WEB	Usado para referirse a WWW.
WWW	"World Wide Web". Red de área mundial, red de computadoras consistentes en una colección de sitios de Internet que ofrecen textos, sonido, gráficos, animaciones y video a través del protocolo de transferencia de hipertexto.

INTRODUCCION

Las compañías que prestan el servició de capacidad satelital tienen la necesidad de medir el patrón de radiación de las antenas que hacen uso de su segmento espacial, para asegurarse que cumplen con las recomendaciones ITU-R S.465-5 y S.580-5 de la UIT ("Unión Internacional de telecomunicaciones")[UIT465, 1993][UIT580, 1993]. También se deben asegurar cumplir la regla 25.209 de la US FCC ("Federal Communications Commission")[Andrew, 1994].

La gráfica de la variación espacial de la potencia recibida/radiada a un cierto radio constante desde la antena es llamado **patrón de potencia**. La gráfica de la variación espacial del campo eléctrico (magnético) a un cierto radio constante desde la antena es llamado **patrón de amplitud de campo**.

Usualmente, el **patrón de campo** describe los valores del campo normalizado (potencia normalizada) con respecto al valor máximo. El patrón de potencia y el patrón de amplitud de campo son iguales cuando se calculan y grafican en decibeles y simplemente es llamado **patrón de radiación**.

Necesidad de evaluar el patrón de radiación

Los satélites geoestacionarios por su naturaleza solo pueden ser colocados sobre el ecuador por lo que el número de posiciones orbitales para colocarlos es limitado. En un principio se colocaban satélites geoestacionarios con una separación de 3 grados. Pero con la escasez de posiciones orbitales útiles se han colocado satélites con una separación de 1.5 grados.

Debido a lo cerca que conviven los satélites geoestacionarios es necesario tomar precauciones para que una estación terrena que suba señales a un satélite no interfiera con las señales de un satélite contiguo.

Para evitar las interferencias la UIT y la US FCC recomiendan que el patrón de radiación no sobrepase un envolvente de $29-25\log(\varphi)$[Andrew, 1994][UIT465, 1993].

En la siguiente figura se muestra un patrón de radiación con la envolvente $29-25\log(\varphi)$ sobrepuesta.

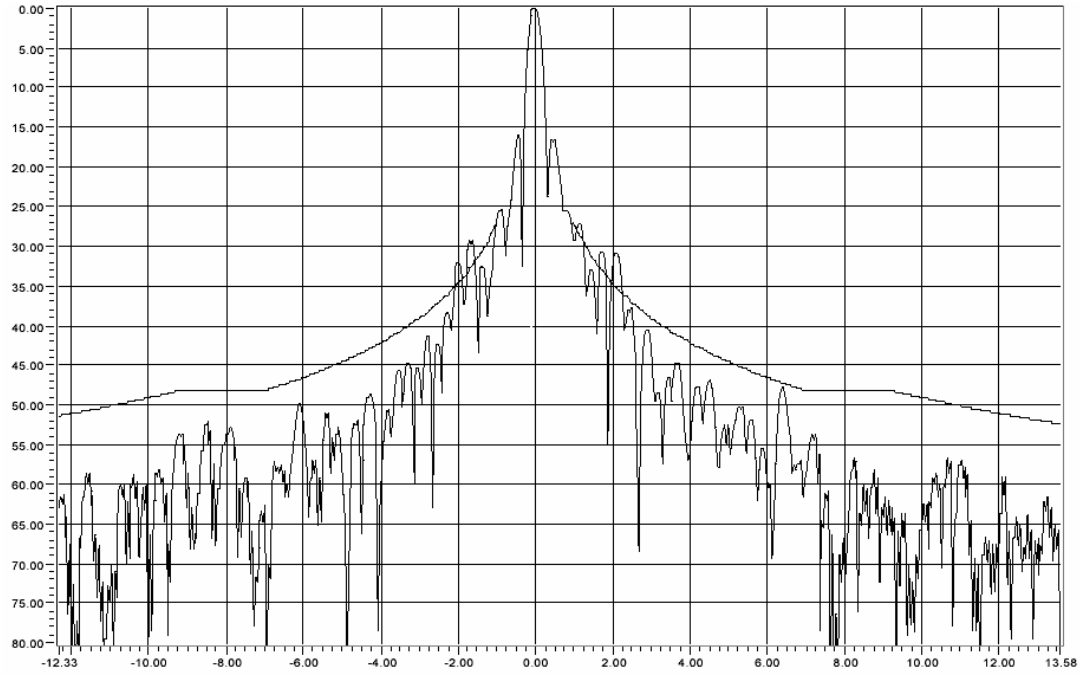


Figura 0-1 Evaluación de un Patrón de Radiación.[1]

Organización del documento

Esta tesis se divide en 4 capítulos.

En el **capítulo 1** se aborda el estado del arte, proporcionando una descripción de lo que es el patrón de radiación como se adquiere y evalúa. También se proporciona una descripción de los instrumentos virtuales de los que son los sistemas en tiempo real tratando las tareas periódicas. Además se comenta sobre la creación de manejadores de dispositivos y lo que son los sistemas en línea y su interacción con los sistemas de tiempo real.

Así mismo se plantea una descripción del problema, los objetivos a alcanzar y la solución propuesta.

[1] Cortesía de la compañía Vertex Corporation.

En el **capitulo 2** se hace un análisis de los requerimientos que la empresa SATMEX plantea y se describe como se pretende satisfacer estos requerimientos. Se utiliza UML, con el enfoque que da Douglas[Douglas B., 2003][Douglas B., 2004] para su aplicación al análisis y diseño de sistemas de tiempo real, en la creación de un diagrama de casos de uso general del sistema y la creación de diagramas de secuencia.

También se justifica el porque se necesita hacer un muestreo en tiempo real y porque se selecciono RT-Linux como plataforma.

En el **capítulo 3** se presenta el diseño y la implementación del software cubriendo los diagramas de clase, el protocolo de comunicación entre el cliente y el servidor haciendo énfasis en la manera en que se evito el bloqueo del STR en la comunicación con el SL.

También se muestran los detalles de las funciones manejadoras de la tarjeta GPIB que se tuvieron que implementar para el muestreo en tiempo real.

Así mismo se documentan los parámetros de la tarea de tiempo real que realiza el muestreo.

En el **capítulo 4** se muestra el sistema obtenido para la adquisición y evaluación del patrón de radiación de una antena, así como la medición y evaluación de un patrón de radiación de una de las antenas de SATMEX. También se describe la metodología obtenida para tal medición y evaluación. Se finaliza el capítulo con las conclusiones, recomendaciones y siguientes pasos.

CAPITULO 1 ESTADO DEL ARTE

En este capítulo se muestra qué es el patrón de radiación de una antenas satelital cómo se resuelve en la actualidad la medición y caracterización de los diagramas de radiación de las antenas para comunicaciones satelitales, también se describen los instrumentos virtuales, STR, SR, SL, tareas periódicas, manejadores de dispositivos y la comunicación entre STR y SL.

1.1 Antenas reflectoras

Un reflector es usado para concentrar la energía electromagnética en un punto focal donde es localizado el receptor/alimentador. Los astrónomos ópticos tienen un amplio conocimiento de que un espejo cilíndrico parabólico transforma los rayos desde una fuente lineal en su línea focal en un conjunto de rayos paralelos. Los reflectores son usualmente parabólicos (paraboloides). De hecho, el primer uso de un reflector parabólico (cilindro) fue usado para ondas de radio por Heinrich Hertz en 1888.

Raramente los reflectores de esquina son utilizados. Las antenas reflectoras tienen muy alta ganancia y directividad. Aplicaciones típicas son en radiotelescopios y en telecomunicaciones por satélite.

1.2 Patrón de radiación

Un patrón de radiación es un gráfica que representa las intensidades de los campos o las densidades de potencia en varias posiciones angulares en relación con una antena. Si el patrón de radiación se traza en términos de la intensidad del campo eléctrico (E) o de la densidad de potencia (P), se llama patrón de radiación absoluto. Si se traza la intensidad del campo o la densidad de potencia en relación al valor en un punto de referencia, se llama patrón de radiación relativo.

Algunas veces no nos interesa el patrón de radiación en tres dimensiones, al no poder hacerse mediciones exactas sobre el. Lo que se suele hacer es un corte en el diagrama de radiación en tres dimensiones para pasarlo a dos dimensiones. Este tipo de diagrama es el más habitual ya que es más fácil de medir y de interpretar.

La figura siguiente muestra un patrón de radiación absoluto para una antena no especificada. El patrón se traza sobre papel en coordenadas polares con la línea gruesa sólida representando los puntos igual densidad de potencia (10 uW/m2). Los gradientes circulares indican la distancia en pasos de dos kilómetros. Puede verse que la radiación máxima está en una dirección de 90º de la referencia. La densidad de potencia a 10 kilómetros de la antena en una dirección de 90º es 10uW/,2. En una

dirección de 45º, el punto de igual densidad de potencia es cinco kilómetros de la antena; a 180º, esta a solamente 4 kilómetros; y en una dirección de -90º, en esencia no hay radiación.

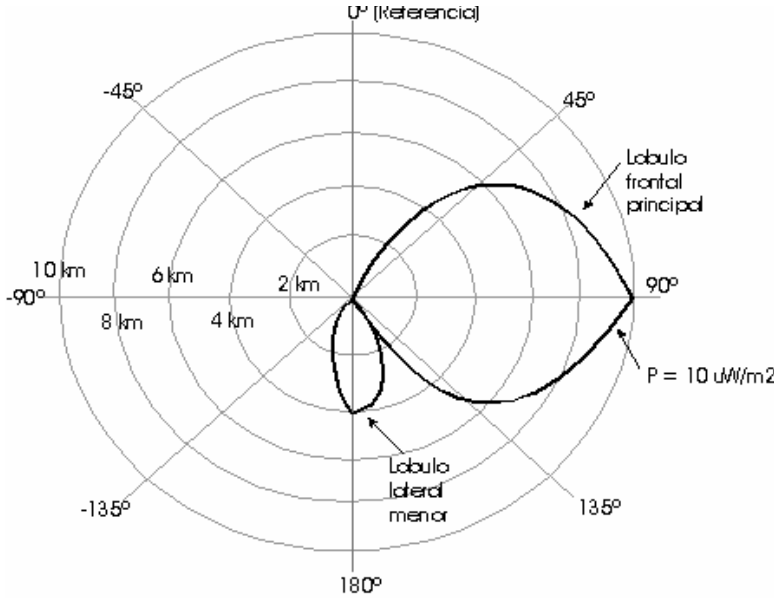


Figura 1-1 Patrón de radiación absoluto.

En la Figura 2-6 el haz principal se encuentra en una dirección de 90º y se llama lóbulo principal. Puede existir más de un lóbulo principal. También hay un haz secundario o lóbulo menor en una dirección de +180º. Normalmente, los lóbulos menores representan radiación o recepción indeseada. Debido a que el lóbulo principal propaga y recibe la mayor parte de energía, este lóbulo se llama lóbulo frontal (la parte frontal de la antena). Los lóbulos adyacentes al lóbulo frontal se llaman lóbulos laterales (el lóbulo menor de 180º es el lóbulo lateral), y los lóbulos que están en dirección exactamente opuesta al lóbulo frontal se llaman lóbulos traseros (en este patrón no se muestra ningún lóbulo trasero). La línea que divide el lóbulo principal desde el centro de la antena en la dirección de máxima radiación se llama línea de tiro.

En la siguiente figura se muestra un patrón de radiación relativo para la densidad de potencia en decibeles. En una dirección de ± 45º de la referencia, la densidad de

potencia es -3dB (media potencia) relativa a la densidad de potencia de dirección de máxima radiación (0º).

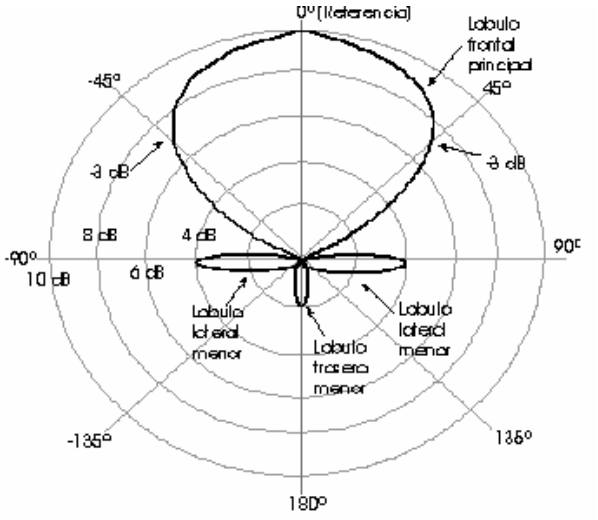


Figura 1-2 Patrón de radiación relativo con densidad de potencia en decibeles.

Como mencionamos anteriormente los patrones de radiación mostrados en las figuras anteriores están en dos dimensiones. Sin embargo, la radiación proveniente de una antena real es tridimensional. Por consiguiente, los patrones de radiación se toman en ambos planos, el horizontal (azimut), y el vertical (elevación). El patrón de radiación también puede ser representado en coordenadas rectangulares como se muestra en las siguientes figuras.

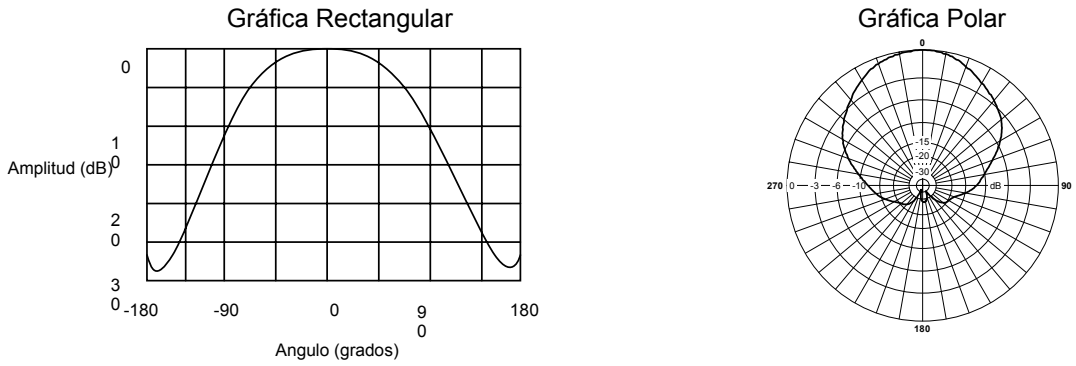
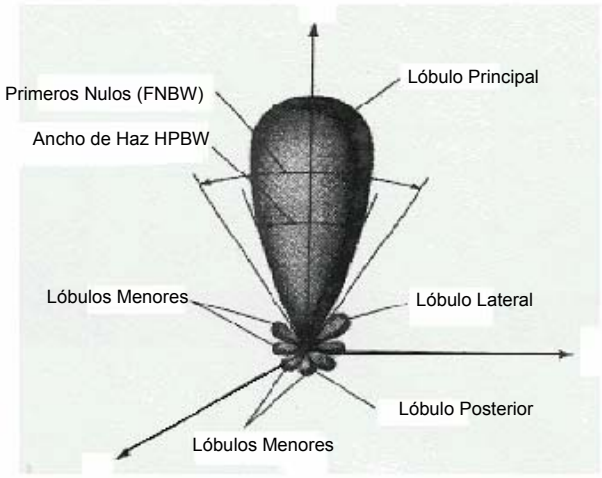


Figura 1-3 Gráfica Rectangular y Polar de un Patrón de Radiación.

1.2.1 Lóbulos del patrón de radiación

Un patrón de radiación consiste en un lóbulo mayor (referido como el lóbulo principal) y varios lóbulos menores. Generalmente los lóbulos menores a cada lado del lóbulo principal son referidos como los lóbulos laterales y el lóbulo directamente opuesto al lóbulo principal se referencia como el lóbulo posterior. Las regiones de relativa baja intensidad de radiación entre los lóbulos son llamados nulos.



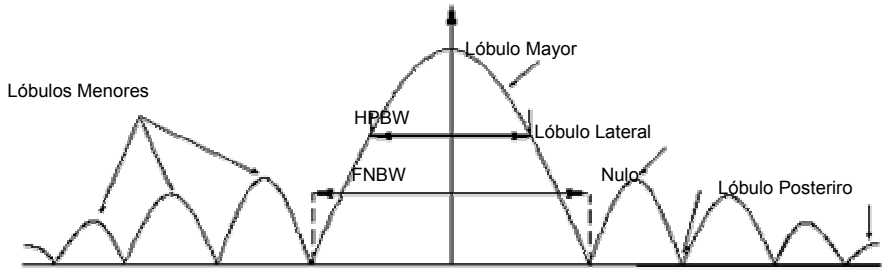


Figura 1-4 Lóbulos de un Patrón de Radiación.

1.2.2 Patrón a la recepción y a la transmisión.

Los patrones de radiación pueden ser medidos a la transmisión o a la recepción. Los patrones que interesan a los operados satelitales son los que se miden a la transmisión por que son los que tiene efectos sobre el satélite y sobre las señales de otros clientes.

1.2.3 Adquisición y evaluación del patrón

En el artículo "Field Testing An Earth Station for Two Degree Compliance"[Morgan A. et al, 1988] los autores, Michael A. Morgan y Dennis Burt, sientan las bases para la medición automatizada del sistema de patrón de radiación utilizando un satélite intermedio entre la antena transmisora y receptora. En éste artículo también se menciona la necesidad de la corrección del patrón de radiación de azimut debido a la forma de la tierra.

En el artículo mencionan que hacen uso de un analizador 8566, de una computadora HP9816S para obtener el trazo del analizador y montarle la curva de evaluación. Desde entonces las compañías fabricantes de antenas han desarrollado sistemas propios para medir el patrón de radiación de una antena, estos sistemas no los comercializan por lo que si una compañía requiere de un software para medir el patrón de radiación lo debe adquirir sobre pedido a las compañías fabricantes de software y no siempre resulta lo que se requiere.

1.2.3.1 Detalle de la adquisición

Como antecedente la antena a evaluar debe ser capaz de moverse con velocidad constante, si no se mueve con velocidad constante la medición del patrón se deforma y por lo tanto no será real.

Se realizan los siguientes pasos:

- Medir cuanto tiempo tarda la antena a evaluar en recorrer los grados que se moverá para adquirir el patrón, (típicamente de -5 a +5 grados de su posición de máximo apuntamiento).
- En la antena a evaluar teniendo máximo apuntamiento enviar una portadora sin modular (CW) en una frecuencia acordada.
- Utilizando una antena receptora colocar su bajada al analizador de espectros.
- Sintonizar el analizador de espectros ajustando la frecuencia central, el nivel de referencia, la banda de frecuencia el tiempo de barrido, el filtro de resolución y el filtro de video. Para obtener en el despliegue la portadora sin modular (CW).
- Mover la antena a evaluar a la posición inicial del movimiento (ángulo inicial).
- En el analizador:
    - Ajustar que realice un solo trazo.
    - Ajustar la banda de frecuencia a 0 (“Frequency Spawn 0”). Para que funcione como osciloscopio y solo mida la potencia recibida.
    - Ajustar el tiempo de barrido al tiempo que tarda la antena en moverse los grados a medir.
- Iniciar el barrido en el analizador de espectros al mismo tiempo que se empieza el movimiento de la antena. En la figura 2.5 se muestran posiciones sucesivas del patrón de radiación al mover una antena desde -5 grados a 5 grados de su posición de máximo apuntamiento, un circulo rojo muestra el valor de la potencia que el analizador de espectros esta recibiendo para diferentes instantes en el tiempo, se observa como en los extremos del movimiento es posible que se este midiendo los lóbulos laterales.

Al final del procedimiento anterior el despliegue del analizador de espectros tendrá el patrón de radiación de la antena.

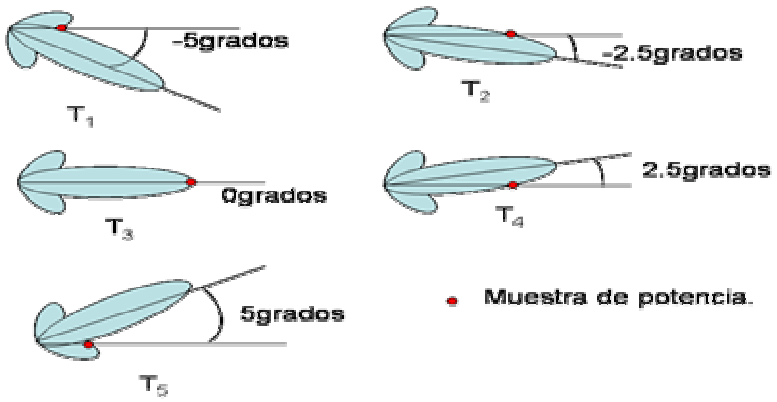


Figura 1-5 Posiciones sucesivas del patrón de radiación al mover la antena.

1.2.3.2Detalle de la evaluación.

- Para evaluar el patrón de radiación se le monta a este la siguiente curva de evaluación:

Gain-[29- 25LOG (ø)dB	1 ≤ ø ≤7	-1 ≤ ø ≤-7
Gain-[+8]dB	7 < ø ≤ 9.2	-7 < ø ≤ -9.2
Gain-[32- 25LOG (ø)] dB	9.2 < ø ≤ 48	-9.2 < ø ≤ -48
Gain-[-10]dB	48 < ø ≤ 180	-48 < ø ≤ -180

En la figura 3-9 se puede observar la envolvente para una antena cuya ganancia es de 50 dB.

Para montar la curva:

- Se puede fotografiar el despliegue del analizador y dibujar a mano la curva como en un principio se hacía.
- Si el analizador cuenta con una salida para “plotter” o impresora se puede imprimir el patrón y posteriormente dibujar a mano la cuerva.
- Si al analizador se le conecta una computadora, ésta puede encargarse de montarle la curva de evaluación después de haber adquirido el patrón del analizador.

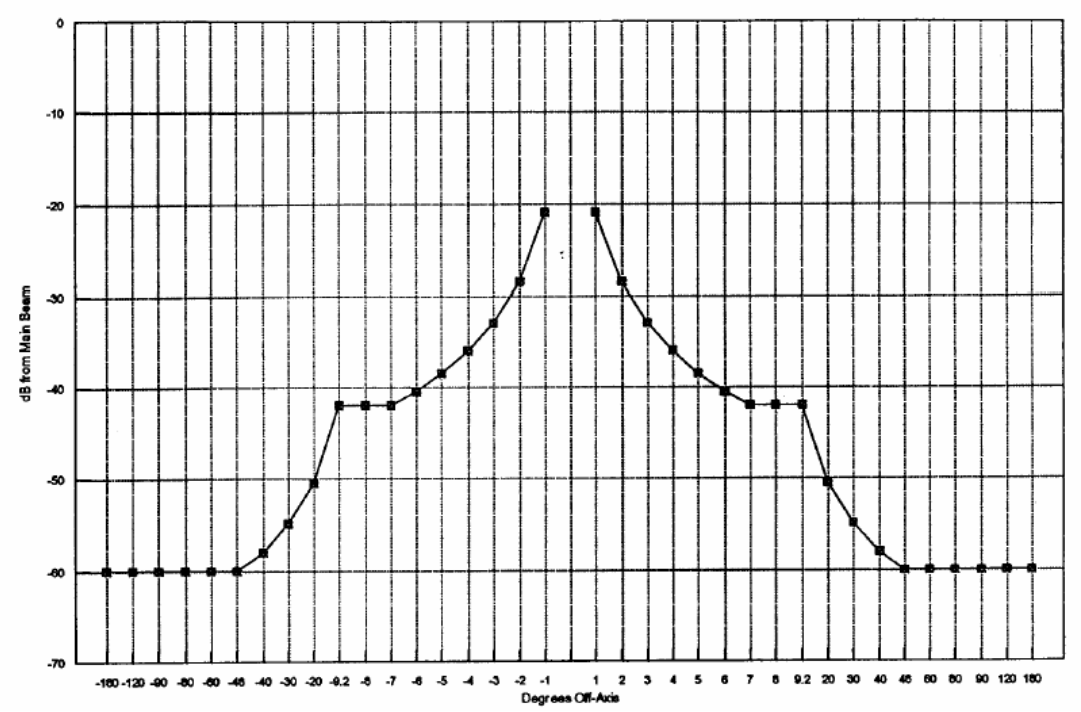


Figura 1-6 Curva de evaluación para una ganancia de 50dB.[2]

- Una vez que se tiene la curva montada se pueden seguir recomendaciones de la UIT[UIT465, 1993][UIT732, 1993][UIT580, 1993] o de la US FCC[Andrew, 1994], algunas de estas recomendaciones son las siguientes.
    - Que el número total de lóbulos que sobrepasan la curva de evaluación sea menor que el 10% de los lóbulos totales.
    - Si el número total de lóbulos no es mayor de 5 entonces la suma de los ángulos de los lóbulos que sobrepasan la curva de evaluación debe de ser menor del 10% del ángulo total que se evaluó.

[2] Cortesía de Andrew Corporation.

- Ningún lóbulo debe de sobrepasar en más de 3 dB la curva de evaluación.
- No se debe sobrepasar la curva de evaluación en x=-1.5grados ó en x=+1.5grados.

1.3 Instrumento virtual.

Los avances en la tecnología de computo han creado una nueva cara a la instrumentación, agregando a la medición de variables físicas la capacidad del procesamiento, análisis, almacenamiento, distribución y despliegue de datos e información relacionas con la medición.

La instrumentación virtual también involucra la interfaz Hombre-Máquina, las funciones de análisis y procesamiento de las mediciones, las rutinas de almacenamiento y recuperación y la comunicación con otros equipos.

El instrumento virtual es definido entonces como una capa de software y posiblemente hardware que se le agrega a una PC, de tal forma que permite a los usuarios interactuar con la computadora como si estuviesen utilizando su propio instrumento electrónico "hecho a la medida"[Vicuña R., 2001].

La naturaleza de cualquier instrumento es que posea características de tiempo real,…Por lo tanto para que una herramienta de medición virtual sea considerada como tal, resulta importante que la plataforma de hardware y software que la soporta contenga propiedades de tiempo real[Vicuña R., 2001].

1.4 Sistemas de tiempo real

Se define un sistema de tiempo real (STR) como un *sistema en el cuál la corrección del mismo depende no sólo de los resultado lógicos de sus cálculos, sino también del tiempo en el cuál se producen*[Vicuña R., 2001].

Se pueden clasificar los sistemas de tiempo real de acuerdo a los plazos de vencimiento de los procesos que se ejecutan en tres tipos:

Duro ("Hard"). Es de tipo duro si la terminación de los procesos después de sus "deadlines" puede resultar catastrófica para el sistema[Butazzo G., 1997].

Suave ("Soft"). Es de tipo suave si la terminación de los procesos después de sus "deadlines" decrementa el funcionamiento del sistema pero no atenta contra su funcionamiento correcto[Butazzo G., 1997].

"Firm". Es de tipo "firm" si el "deadline" no se cumple ocasionalmente , pero ya no hay beneficio en un cumplimiento tardío[Burns. et al, 1997].

1.4.1 Tareas periódicas

La tareas periódicas consisten en una secuencia de actividades idénticas llamadas instancias[Butazzo G., 1997].

Una tarea periódica puede ser determinada por su tiempo de ejecución C_i y su deadline relativo D_i que muchas veces es coincidente con su periodo T_i. Además estos parámetros son considerados como constantes para cada instancia.

1.4.2 Sistemas operativos de tiempo real

Los sistemas operativos de tiempo real básicamente deben de satisfacer:

- Calendarización de tareas bajo restricciones de tiempo y de prioridades.
- Previsibilidad.
- Tolerancia a fallos.
- Manejo de los deadlocks de procesos.
- Garantías de ejecución de tareas en tiempos límites.

Deben implementar algunas funciones básicas:

- El cambio de contexto entre tareas debe de ser lo más rápido posible, por lo que el código encargado de ello es normalmente muy pequeño.
- Manejan peticiones de memoria real, no virtual. El manejo de memoria virtual exige esfuerzos adicionales.
- Proveen archivos secuenciales adicionales para acumulación rápida de datos.
- Implementan algoritmos sofisticados de planificación de tareas.

Los sistemas operativos de tiempo real comerciales se pueden clasificar en dos categorías:

- Sistemas operativos de tiempo real.
- Sistemas operativos con extensiones de tiempo real[Vicuña R., 2001].

1.4.2.1RT-Linux

Las primeras versiones de RT-Linux ofrecían un API muy reducido sin tener en cuenta ninguno de los estándares de tiempo real: POSIX Real-Time extensions, PThreads, etc.

A partir de la versión 2.0 Victor Yodaiken decide reconvertir el API original a otro que fuera "compatible" con el API de POSIX Threads.

Para implementar RT-Linux en lugar de modificar el núcleo de Linux para ofrecer nuevas llamadas al sistema y que sea predecible, lo que se hizo fue construir directamente sobre el procesador (i386) un pequeño núcleo --que no tiene nada que ver con el núcleo de Linux-- con su propio planificador. Sobre este núcleo (o máquina virtual) el SO Linux se ejecuta como una tarea más. Linux se ejecuta en background, cuando no hay otras tareas que ejecutar.

El planificador por defecto que viene con RT-Linux es un planificador basado en prioridades estáticas y trata a la tarea Linux como la tarea de menor prioridad. Si las tareas de tiempo real consumen todo el tiempo del procesador, entonces la tarea Linux no recibe tiempo de procesador y da la impresión de que el sistema se ha "colgado".

1.4.2.1.1 Características de RT-Linux:

- Sistema operativo de tiempo real estricto.
- Extensiones para entorno multiprocesador SMP (x86).
- API "próximo" al de POSIX threads. Planificador expulsivo por prioridades fijas, señales, sistema de archivos POSIX (open, close, etc.) semáforos y variables condición.
- Depuración de código mediante GDB (GNU Debuger).
- Soporte para arquitecturas x86 y PPC.
- Acceso directo al hardware (puertos e interrupciones).
- Comunicación con procesos Linux mediante memoria compartida y tuberías.
- Estructura modular para crear sistemas embedidos.
- Eficiente gestión de tiempos. En el peor caso se dispone de una resolución próxima al microsegundo (para un i486).

- Facilidades para incorporar nuevos componentes: relojes, dispositivos de E/S y planificadores. ·

1.5 Manejadores de dispositivos

Un manejador de dispositivos es código específico para un dispositivo físico y que permite comunicarse con él[Rubini A. et al, 2001].

El kernel de un sistema operativo tiene interconstruidos varios manejadores de dispositivos para comunicarse por ejemplo con el teclado, el disco duro, unidades de cinta, etc.

Generalmente cada fabricante ofrece su dispositivo con manejadores de dispositivos específicos para diferentes sistemas operativos y la información para construir estos manejadores no es pública por lo que la construcción de un manejador de dispositivos para algún sistema operativo no es sencilla debido a la carencia de información.

1.5.1 Manejadores de dispositivos en RT-Linux.

En RT-Linux como en Linux solo existe una manera de implementar un “device driver” (manejador de dispositivos) pero la forma de cargarlo puede ser de dos tipos:

- Interconstruido en el kernel.
- Como módulo, el cual se carga y descarga dinámicamente.

Otra forma de acceder al dispositivo es que el código esté inmerso en la aplicación para lo cual no se tiene que crear un manejador de dispositivo sino solo las rutinas necesarias para ocupar solo las funciones que se necesiten del dispositivo. Este enfoque tiene la desventaja que estas rutinas no se puedan usar en otra aplicación por ser muy específicas, pero tiene la ventaja de que si se enfocan en solo la funcionalidad deseada pueden ser más rápidas de crear.

1.5.2 Sincronización del manejador de dispositivo

La sincronización entre el hardware y el procesador se logra esencialmente de dos maneras:

- Por poleo, en esta manera el manejador constantemente interroga al hardware. Esto resulta tiempo perdido del procesador, pero algunas veces es la forma más rápida de comunicarse con el hardware.

- Por interrupciones, esta manera sólo es posible si el hardware la soporta. En esta manera el dispositivo informa al procesador, vía una interrupción, que se ha completado una operación entonces el procesador suspende su actual proceso y carga un a rutina de servicio de interrupción.

1.5.3 Modo de DMA

Se usa cuando grandes cantidades de datos son continuamente transportados desde o para el dispositivo. En este modo el controlador de DMA se encarga del transporte del dispositivo a la memoria o viceversa sin que intervenga el procesador.

El dispositivo generalmente dispara una interrupción cuando se ha completado la transferencia. Aunque existen algunos dispositivos que no soportan el manejo de interrupciones y para los cuales se tiene que usar el poleo.

1.6 Sistemas en línea.

A diferencia de un STR un SL no tiene que cumplir con las siguientes condiciones:

- Contacto con el mundo físico.
- Emisión de respuestas correctas.
- Sincronía con el mundo físico.

El SL solo tiene que procesar datos de acuerdo a su orden de llegada sin importar el tiempo de duración , sin sincronía.

Un SL es aquel sistema que está encendido, conectado y listo para recibir datos bajo el control de una computadora central, una red de cómputo o una conexión directa[Medel J. et al, 2002].

1.6.1 Comunicación entre un STR y un SL

Existen dos técnicas básicas para la comunicación entre STR y SL[Medel J. et al, 2002]*:*

- *Comunicación a través de colas de espera, es usado a través de sockets o colas de mensajes (queues) implementados en el SL … lo recomendable es tener un buffer lo suficientemente grande para recibir sus datos sin bloquear su funcionamiento.*
- *Comunicación por mensajes no bloqueantes,*

- *Los sistemas POSIX … soportan el paso de mensajes no bloqueantes y sockets no bloqueantes… de tal manera que el STR no entre en el estado de bloqueo por espera de respuesta; la mayor desventaja de éste ocurre cuando hay una gran pérdida de datos por problemas en la red o por alta de capacidad del cliente (SL)*

1.7 UML para tiempo real

En este trabajo se utilizó la metodología de Douglas[Douglas B., 2003][Douglas B., 2004] para el modelado, análisis y diseño del sistema. La mayoría de las metodologías del uso de UML se enfocan en el dominio de los negocios o bases de datos y existen muy pocas que se enfoquen a los sistemas de tiempo real. La metodología de Douglas utiliza la notación y semántica de objetos que tiene UML en el desarrollo de sistemas de tiempo real.

Algunas de las características que se encontraron de esta metodología son las siguientes:

- Debido a que los sistemas de tiempo real interactúan con el mundo físico, hay un mayor uso de los actores en los diferentes tipos de diagramas, estos actores representan sensores, actuadotes o algún otro dispositivo.

- Como frecuentemente la transmisión de mensajes no es dada en la misma máquina y muchas de las veces estos mensajes son transmitidos a dispositivos físicos, los mensajes en los diagramas de secuencia no siempre se mapean a métodos en las clases sino, que son mensajes que se deben de transmitir por alguna interfaz física con el protocolo adecuado.

- Los diagramas de secuencia contiene marcas de restricciones de tiempo y marcas de estado.

1.8 Descripción del Problema

Existen pocos desarrolladores que ofrezcan un producto que realice mediciones del Patrón de Radiación de Antenas. En Internet no se ha podido encontrar referencia alguna de un proveedor comercial.

Las compañías que necesitan medir el Patrón de Radiación de una Antena tienen que llevar a cabo un desarrollo propio, tal es el caso de Vertex , Andrew y Comsat.

Otras compañías han comprado un software a la medida de sus necesidades para medir el patrón de radiación de una antena, es decir han tenido que hacer un pedido especial, con lo que el costo por este software ha sido alto.

La compañía SATMEX cuenta con un software para la medición del patrón de radiación que le fue comprado a la compañía Calian de Canadá (actualmente desaparecida). Este software utiliza un analizador de espectro HP8566B que actualmente está por ser descontinuado y la compañía HP ya no dará servicios para este equipo. Además la máquina sobre la que corre este sistema es una HP9000 con sistema operativo HP-UX, lo que hace que el costo de mantenimiento de esta máquina sea alto.

Por lo anterior SATMEX ha pedido a la compañía norteamericana "SAT Corporation" que desarrolle un software a la medida para medir y evaluar el patrón de radiación, pero el desarrollo ha tardado mucho y el software no ha funcionado como SATMEX lo requiere.

Al tener un desarrollo propio del software para medición del patrón de radiación la compañía ganara flexibilidad al poder realizar las modificaciones que desee, podrá resistir la obsolescencia de sistemas atados al hardware y se ahorrará el costo de las modificaciones.

1.9 Objetivo General:

Tener un instrumento virtual que sirva para la adquisición y evaluación del patrón de radiación de antenas de los clientes de SATMEX y de sus propias antenas.

1.10 Objetivos Específicos:

- Desarrollar una metodología para medir el patrón de radiación de una antena utilizando el equipo de la empresa SATMEX (Una antena receptora, una analizador de espectros y la red y computadoras).
- Disponer de un programa que entrega los resultados de la evaluación del patrón de radiación y elabora un reporte para el usuario.
- Aplicar las facilidades del sistema operativo RT-Linux en la adquisición de datos de instrumentos.
- Desarrollar rutinas para manejar la tarjeta GPIB PCIIA de "National Instruments" para el sistema operativo RT-Linux.
- Utilizar el estándar IEEE-488.
- Controlar un analizador de espectros por medio de la interfaz IEEE-488.
- Crear un protocolo entre el cliente y el servidor basado en el estándar SCPI.

1.11 Solución propuesta

Para poder adquirir y manipular los datos de un patrón de radiación se propone el uso de un instrumento virtual es decir una capa de software y hardware que les permita a

los usuarios interactuar con su propio instrumento hecho a la medida [Perales A., 2001] para adquirir los datos del patrón de radiación, almacenarlos, procesarlos, evaluarlos y emitir un reporte todo esto dentro de una interfaz gráfica.

En la figura 1-2 se muestra un diagrama del equipamiento usado y sus conexiones:

El instrumento virtual tendrá dos piezas de software:

El programa servidor que se encargará de colectar los valores de la señal el cuál estará desarrollado en C y utilizará las facilidades de RT-Linux, este programa correrá en una PC compatible la cuál tendrá el sistema operativo RT-Linux y contará con una interfaz GPIB para conectarse a un analizador de espectros.

El segundo programa será el cliente que contará con la “GUI”, estará desarrollado en Java y se conectará al servidor a través de la red. Este programa a parte de ser la interfaz gráfica con el usuario se encargará de enviar las peticiones de medición al servidor y de procesar los resultados mostrándolos en forma gráfica, así mismo montará sobre la gráfica las funciones que recomienda la UIT para evaluar el patrón y emitir un reporte.

Se cuenta con una tarjeta GPIB PCIIA de “National Instruments” para la cual ya existen controladores para Linux. Con esta tarjeta se pretende controlar analizadores de espectro HP8566B, se utilizará este analizador por su disponibilidad.

Para medir el patrón de radiación de la antena de un cliente se le pedirá que apunte al satélite y suba una portadora sin modular (CW), se le pedirá que haga un barrido en azimut y otro en elevación. Utilizando las antenas disponibles en SATMEX se encaminará la señal al sistema de medición.

El diagrama de equipos y la solución propuesta se muestra en la Figura 1-7.

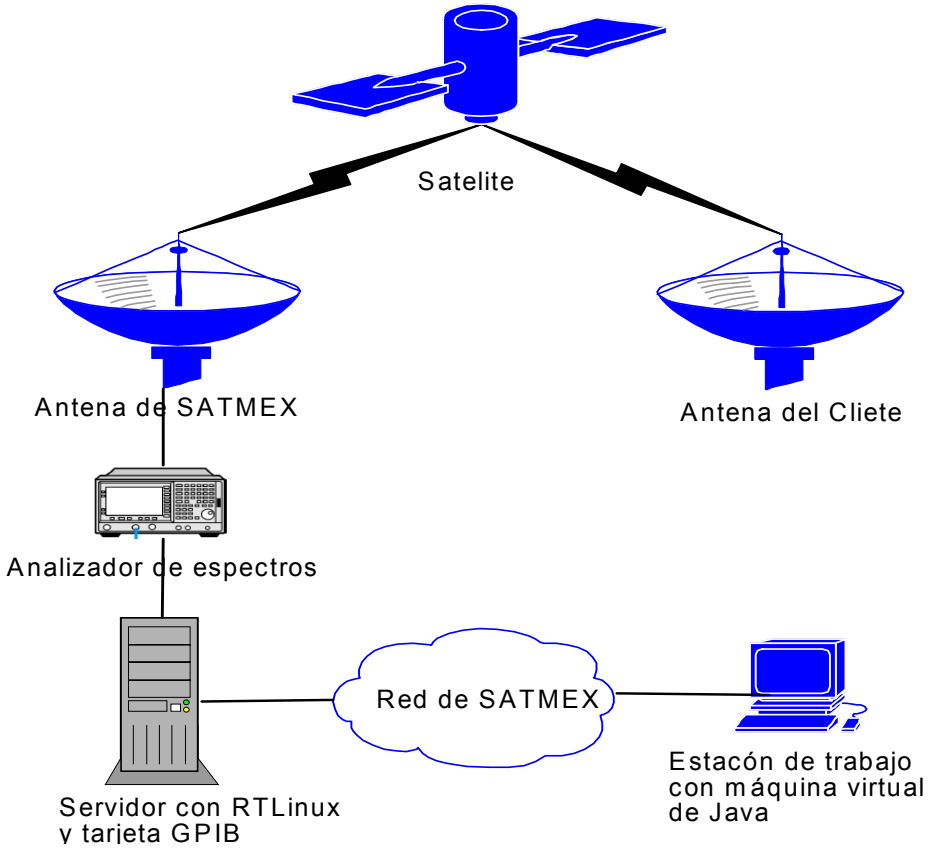


Figura 1-7 Diagrama de equipamiento y conexiones.

CAPITULO 2 ANALISIS DEL PROBLEMA

En este capítulo se exponen los requerimientos que se exigieron a la solución del trabajo, también se hace una descripción de las herramientas de software elegidas para desarrollar el sistema, de igual forma se presenta el equipo existente en la compañía que se usó para la implementación del sistema.

Así mismo se presenta el diagrama de casos de uso general y un diagrama de secuencia para cada caso de uso.

2.1 Requerimientos

Los requerimientos de la empresa que debe satisfacer el sistema se enlistan a continuación:

- Sistema económico sin gastos de inversión.
- Se debe de utilizar el equipamiento existente.
- Se debe de poder medir el patrón de radiación desde cualquier sitio de la empresa a través de la red.

El requerimiento que indica que el sistema debe de ser económico y sin gastos de inversión nos obliga a elegir herramientas de desarrollo de software libre.

Las herramientas de software que se eligieron para el servidor son las siguientes:

1. *Sistema operativo Linux*: es un sistema operativo libre que es estable y se ha utilizado ampliamente para el control de equipos de medición. Se utilizará la distribución RedHat Linux versión 7.0[RedHat, 2003] por contar con el disco de instalación.

2. *Sistema operativo RT-Linux*: El sistema operativo Linux puede ser complementado con los módulos de RT-Linux para convertirlo en un sistema operativo de tiempo real duro. También se pudo haber elegido QNX debido a que en ambos sistemas se ha trabajado, pero la decisión se inclinó por RT-Linux, debido a que para este SOTR se encontró un manejador de dispositivo para la tarjeta GPIB con la que se contaba y como no se cuenta con información sobre la tarjeta GPIB sería difícil hacer uno para QNX[Vicuña R. et al, 2001], además el tiempo de desarrollo de un manejador de dispositivos es largo.

3. *"Device driver" de GPIB*: Se encontró un paquete para Linux de manejadores de tarjetas GPIB en "sourceforge[Sourceforge, 2003] pero estos manejadores fueron

hechos para versiones del Kernel de Linux superiores a la 2.4 (la que se está usando es la 2.2 porque para ésta se tienen los parches de RT-Linux), se probó bajar estos manejadores y tratar de compilarlos pero no se tuvo éxito. Se siguió buscando en la red un manejador y finalmente se encontró uno en la página del Dr. Jens Thoms Törring[Törring J., 2003] el cuál se pudo compilar correctamente.

4. *Lenguaje C*: Se utilizará el lenguaje C que viene en la distribución de Linux elegida en este caso es la versión 2.96 de gcc.

Las herramientas de software que se eligieron para el cliente son las siguientes:

1. *Java:* Se eligió a Java como el lenguaje de desarrollo de la interfaz gráfica con el usuario, no sólo por la práctica que se tiene en el uso de este lenguaje, sino además por sus características de orientación a objetos y la gran cantidad de APIs existentes, con lo anterior se acelera la creación de aplicaciones, además proporciona independencia de plataforma. Se utilizó el paquete "j2sdk1.4.1_06".

2. *NetBeans IDE:* Este es un ambiente de integrado de desarrollo para Java el cuál proporciona entre otras cosas facilidades para el diseño gráfico de la interfaz y la autogeneración de código para ésta. Se uso la versión 3.5.1.

3. *Graph*: Es un paquete libre para el desplegado y manipulación de gráficas desarrollado por Leigh Brookshaw. Se utilizó la versión 2.4 obtenida de Internet[Brookshaw L., 2003].

Para satisfacer el requerimiento que pide que se use el equipo existente, se utilizó el que a continuación se indica, todo este equipo es con el que la empresa SATMEX ya contaba:

1. *Computadora Servidora:* PC Pentium III 600 KHz con disco duro de 6GB Memoria RAM de 64 MB CD-ROM 52x y con Sistema Operativo RT-Linux.

Figura 2-1 Servidor.

2. *Tarjeta GPIB:* Tarjeta “National Instruments” PCIIA de bus ISA.

Figura 2-2 Tarjeta GPIB.

3. *Analizador de Espectros:* modelo HP8566B: Opera en el rango de 100 Hz a 2.5 GHz en la banda de baja frecuencia y de 2 a 22 GHz en la banda de microondas preseleccionada. Contiene una microcomputadora interna para el control automático[HewlettPackard, 1984].

Figura 2-3 Analizador de Espectros HP8566B.

4. *Computadora Cliente:* PC Pentium IV 1.4 MHz, memoria de 128 MB, disco duro de 20 MB. Con máquina virtual de Java. Puede ser cualquiera que tenga una máquina virtual Java.

Para satisfacer el requerimiento que indica que se debe de poder medir el patrón desde cualquier sitio de la empresa se implementó lo siguiente:

1. *Sistema cliente-servidor:* El cliente puede estar en cualquier sitio de la empresa.
2. *El cliente corre en una máquina virtual Java:* Cualquier estación de trabajo con máquina Java puede ser el cliente.
3. *Protocolo de comunicación:* Se implementó un protocolo de comunicación basado en el “Standard Commands for Programming Instrumentation (SCPI)”.

2.2 Diagrama de Casos de Uso

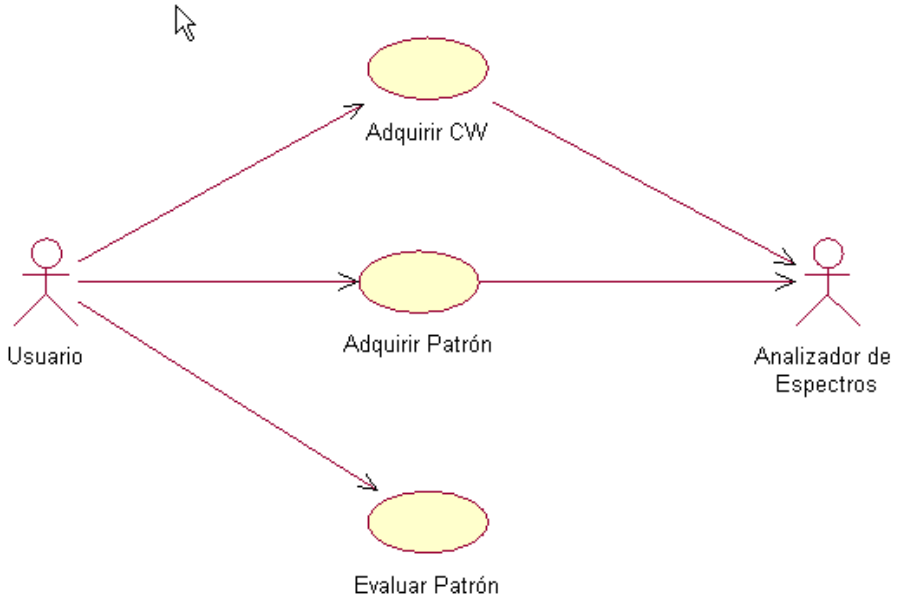


Figura 2-4 Diagrama de General de Casos de Uso del Sistema.

En el diagrama anterior se muestra el diagrama casos de uso general del sistema. Debido a que es un sistema remoto, el usuario debe de adquirir un trazo de la portadora sin modular (CW) para verificar que el analizador esté sintonizado en la portadora correcta. En seguida procede a adquirir el patrón ya sea que lo haga el analizador o el proceso servidor, una vez que se tiene el patrón se procede a introducir datos de la antena y de la estación para posteriormente centrarlo al máximo, normalizarlo y corregirlo, si se trata de un patrón en azimut, posteriormente se evalúa el patrón y se genera un reporte.

2.3 Diagramas de secuencias

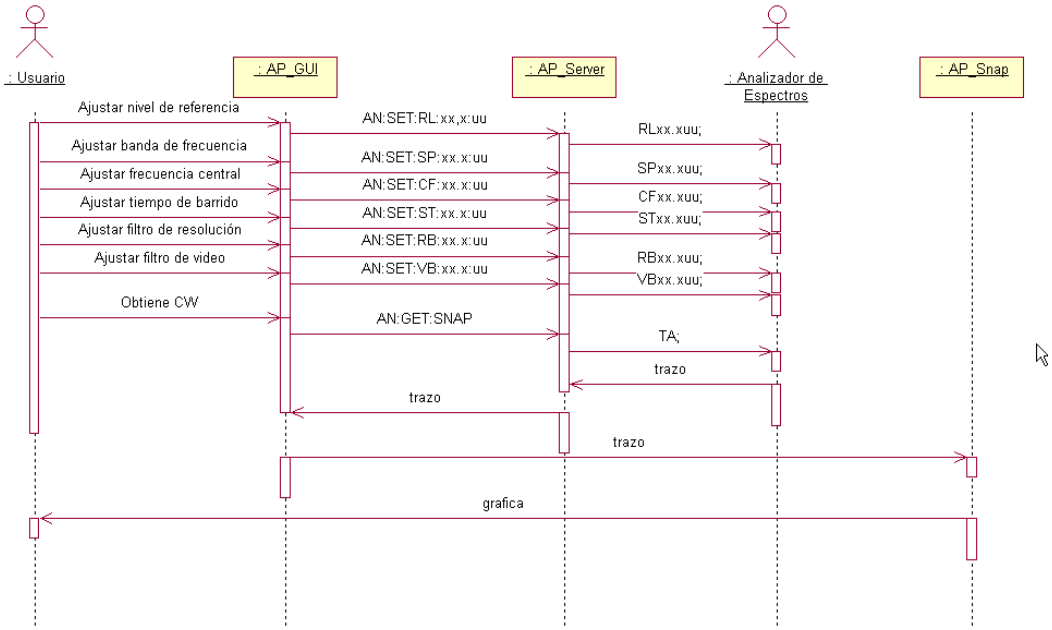


Figura 2-5 Diagrama de secuencia para el caso de uso de la adquisición de "CW".

En la figura 3-5 se muestra el diagrama de secuencias para el caso de uso de adquisición de la portadora sin modular (CW). En el diagrama se aprecian los mensajes que van del cliente (A_GUI) al servidor (AP_Server), estos mensajes son comandos SCPI enviados a través de TCP/IP. Los mensajes que van del servidor (AP_Server) al analizador son instrucciones del analizador enviados a través de la interfaz GPIB. El mensaje de trazo es la secuencia de valores de potencia que en su conjunto forman el patrón de radiación.

En la figura 3-6 se muestra el diagrama de la adquisición del patrón de radiación para el escenario cuando la adquisición la hace el analizador de espectros. Para el caso de uso de adquisición del patrón de radiación se tiene dos escenarios:

- La adquisición la hace el analizador de espectros.
- La adquisición la hace el servidor.

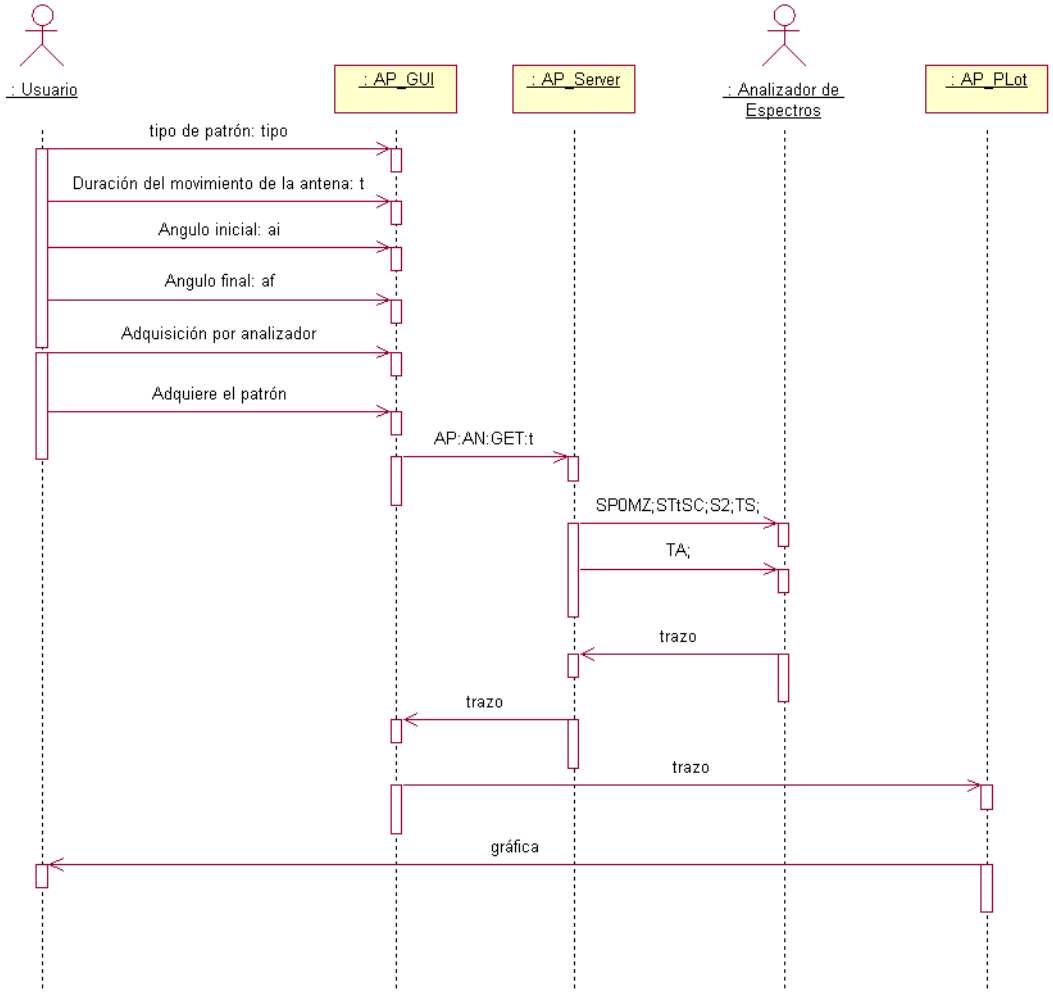


Figura 2-6 Diagrama de secuencia de la adquisición del patrón de radiación con adquisición por el analizador de espectros.

En la figura 3-7 se muestra el diagrama de secuencias de la adquisición del patrón de radiación para el escenario cuando la adquisición es hecha por el servidor. Se observa que se tienen restricciones de tiempo las cuales se documentan según [Douglas B., 2003][Douglas B., 2004].

Cabe mencionar que la comunicación entre AP_GUI y AP_Server es mediante comandos SCPI a través de TCP/IP, y la comunicación de AP_Muestreo con el analizador de espectros es a través de instrucciones del analizador a través de GPIB.

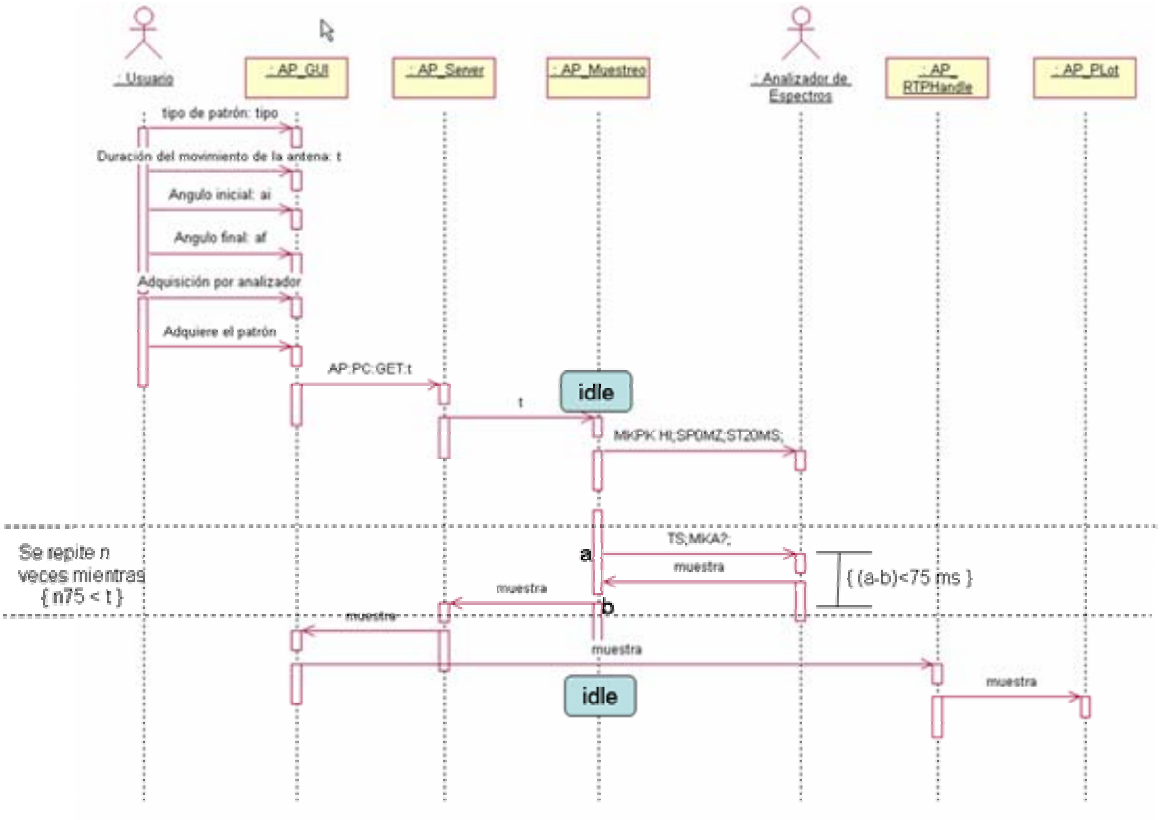


Figura 2-7 Diagrama de secuencia de la adquisición del patrón de radiación con adquisición por el servidor.

En la figura 3-8 se muestra un diagrama de secuencias de la evaluación del patrón de radiación. Se observa que toda la evaluación es hecha en el objeto AP_Plot el cuál debe de contener todas las funciones para la manipulación del patrón de radiación así como las funciones de evaluación y de generar reportes.

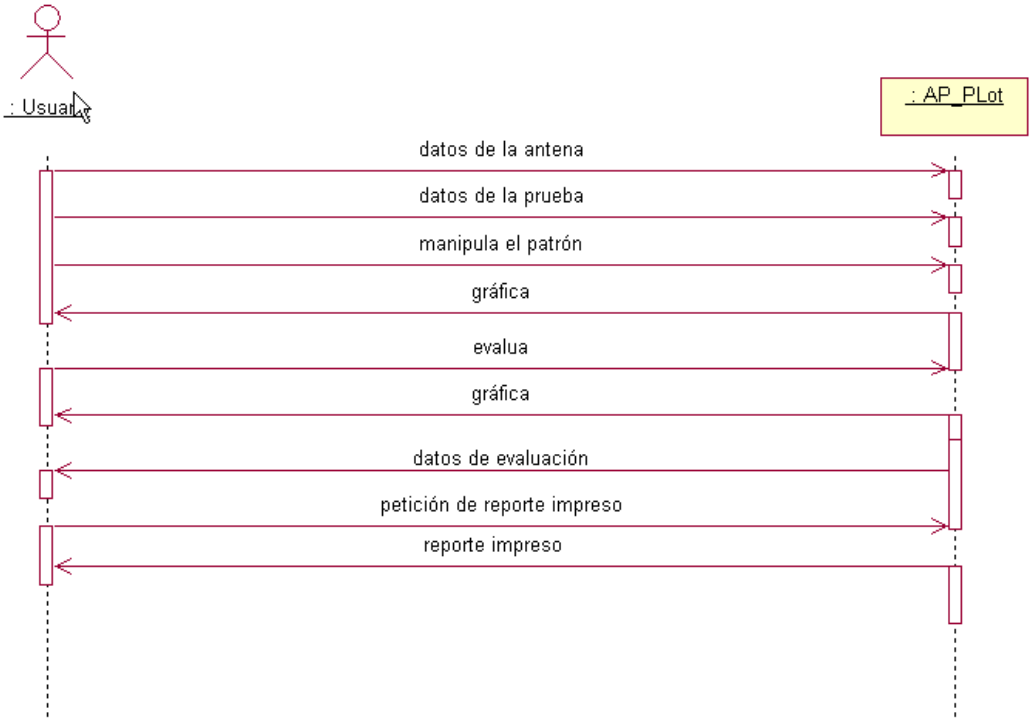


Figura 2-8 Diagrama de secuencias de la evaluación del patrón de radiación.

2.4 Muestreo en tiempo real

Para que los operadores del sistema (usuarios) pudieran ver una graficación en línea del patrón de radiación y no tuvieran que estar esperando a que el analizador termine de adquirir el patrón, para que se pudiera desplegar en una computadora remota, se decidió implementar un muestreo en el servidor.

Este muestreo debe de estar sincronizado con el movimiento de la antena para hacer corresponder la toma de muestras con incrementos del ángulo de la antena. Como la antena es una antena remota y dado que el movimiento de la antena es a velocidad constante una forma de saber la posición de la antena es asegurando que las muestras sean tomadas a intervalos de tiempo constantes e ir calculando el incremento del ángulo. Si no se asegura que la toma de las muestras se haga a intervalos constantes

el patrón de radiación se deformaría, dado que se asignaría un valor de potencia a un ángulo para el cuál no correspondería, debido a la correspondencia directa entre tiempo y movimiento.

Por lo anterior se determina que el muestreo debe ser una tarea de tiempo real periódica para asegurar que las muestras sean tomadas a intervalos de tiempo iguales.

2.5 Selección de SOTR

Lo comentado en el punto anterior sólo se garantiza con un SOTR. Dentro de las opciones de sistemas operativos de tiempo real sólo se habían utilizado dos, RT-Linux y en QNX. Pero la selección de RT-Linux fue debido a la existencia de un “device driver” para la tarjeta que se ocuparía y dado que hacer un “device driver” para QNX implicaría tiempo y tal vez no se contaría con la información suficiente de la tarjeta para poder tener éxito.

Eliminado: DISEÑO E IMPLEMENTACION

CAPITULO 3 DISEÑO E IMPLEMENTACION

En este capítulo se describen los diagramas de clase de los componentes del sistema, se presenta primero los diagramas de clase del cliente y posteriormente se muestran los diagramas de clase del servidor. Se hace énfasis en el protocolo de comunicación SCPI que se diseñó para comunicar al cliente y el servidor. También se expone con más detalle como se diseñaron las funciones manejadoras de dispositivo para el control de la tarjeta GPIB y de la forma en que se comunica el STR y la graficación en línea.

Además se comenta la decisión de porque usar un sistema de tiempo real crítico y de por que se escogió RT-Linux.

El código fuente completo del sistema se encuentra en un CD-ROM que acompaña este trabajo.

3.1 Diagramas de clases del cliente

En el diagrama de la figura 4.1 se muestran las principales clases del cliente.

Una instancia de la clase AP_GUI es la encargada de presentar la interfaz gráfica con el usuario, la cual usa una instancia de la clase AP_Snap para mostrar un trazo adquirido del analizador de espectros, además también usa una instancia de la clase AP_Plot para mostrar un patrón de radiación adquirido por el analizador.

Para mostrar un patrón adquirido por muestreo en tiempo real la instancia de clase de AP_GUI le envía las muestras de potencias a la clase de AP_RTPHandle. Estos dos objetos fueron instanciados en procesos diferentes, incluso como se implementaron en lenguaje Java se encuentran en máquinas virtuales diferentes, por lo que el mecanismo por el que se comunican es mediante sockets.

AP_RTPHandle de la misma manera que AP_GUI usa una instancia de la clase AP_Plot para graficar el patrón de radiación, sólo que en esta ocasión los puntos se van graficando en línea en lugar de todos juntos.

La clase coClient es la clase encargada de las comunicaciones TCPI/IP como se aprecia en el apéndice B, la misma contiene una función para enviar mensajes “sendCmd()” y otra para recibir los mensajes “getMsg()”.

La clase AP_Plot es la encargada de mostrar gráficamente el patrón de radiación para lo cual usa el paquete “graph”, que es un paquete “freeware” de manipulación de gráficas se utiliza la versión 2.4 de éste paquete. También AP_Plot es encargado de

evaluar el patrón de radiación y de generar un reporte de esta evaluación a petición del usuario, por lo que también presenta una interfaz gráfica.

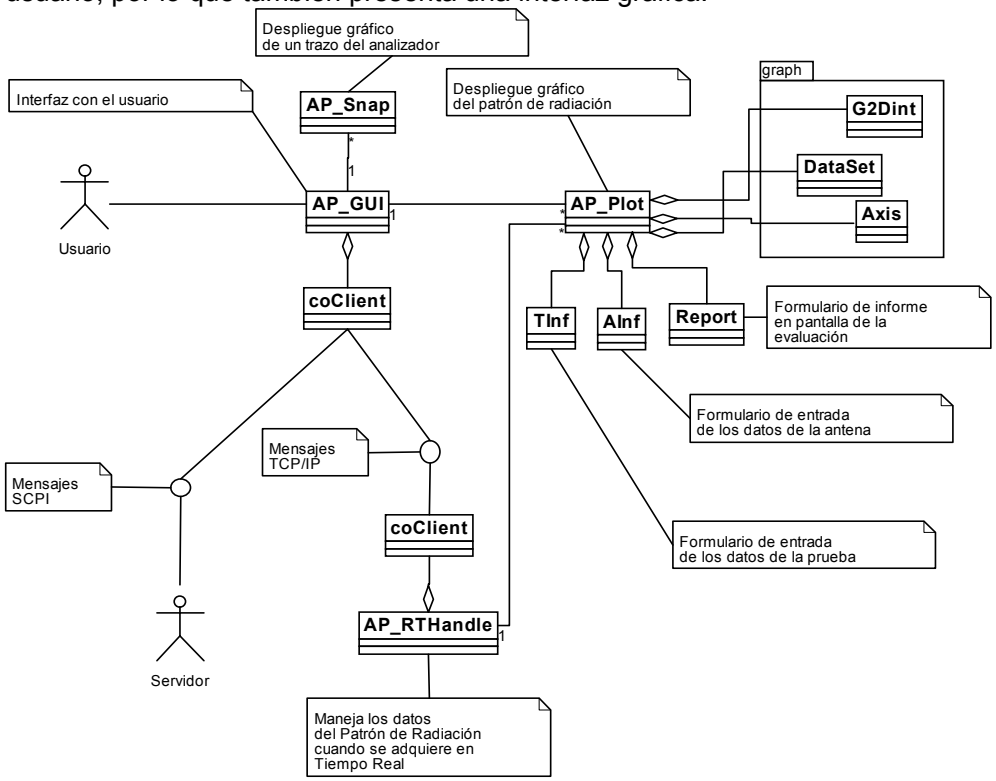


Figura 3-1 Clases del cliente.

3.2 Comunicación entre el cliente y el servidor

La comunicación entre cliente servidor se hace a través de sockets TCP/IP, se diseñaron comandos del tipo SCPI para establecer la comunicación. En la figura 4-2 se muestra un diagrama de la comunicación señalando que la interfaz es a través de TCP/IP con comandos SCPI, esta interfaz es una interfaz física.

En el apéndice A se muestran los comandos tipo SCPI que fueron diseñados y sus respuestas.

Eliminado: DISEÑO E IMPLEMENTACION

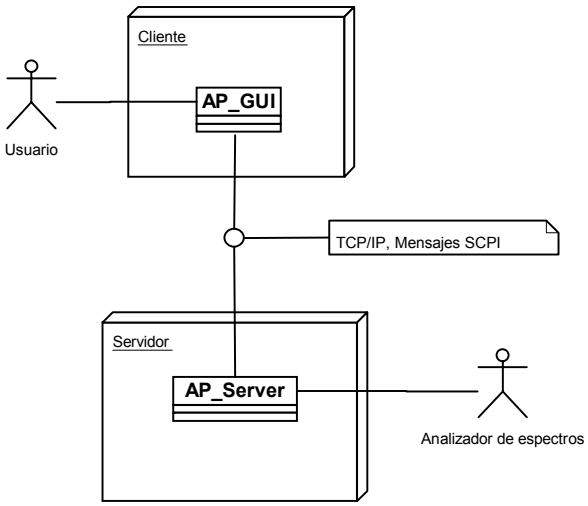


Figura 3-2 Comunicación cliente-servidor.

3.2.1 Forma en que se evito el bloqueo entre el STR y el cliente para la graficación en línea (SL)

Para evitar el bloqueo entre la interacción entre el STR y el sistema de graficación en línea se utilizó una técnica diferente a las descritas en el apartado 2.6.1.

En la máquina servidora están corriendo dos procesos AP_Muestreo y AP_Server, el proceso AP_Server es un proceso que no es de tiempo real por lo que tiene la prioridad más baja y AP_Muestreo que tiene la prioridad más alta por ser un proceso de tiempo real.

AP_Server permanece inactivo hasta que AP_Muestreo coloca en una tubería de tiempo real la duración del muestreo. Entonces AP_Muestreo empieza a tomar de manera periódica una muestra de potencia del analizador de espectros y la coloca en un área de memoria compartida que hace las veces de una cola “queue”. Debido a que la escritura en memoria es no bloqueante no corre riesgos de que se AP_Muestreo se bloquee.

Durante la diferencia de tiempo entre el plazo “deadline” D y el tiempo de ejecución C de AP_Muestreo la tarea AP_Server que no es de tiempo real va sacando de la memoria compartida “queue” los datos y los envía por medio de sockets TCPIP al cliente (SL) para su graficación, si esta tarea se bloquea debido a cargas en la red o al desempeño del SL, no importa, porque debido a que tiene prioridad baja la tarea de tiempo real continuará su ejecución, así que AP_Server seguirá enviando datos al SL aún después que el muestreo hecho por AP_Muestreo haya terminado.

Este sistema de comunicación entre un STR y un SL se puede decir que es basado en una cola “queue” en el STR y mensajes bloqueantes con ayuda de una tarea de baja prioridad.

3.3 Diagramas de clases del servidor

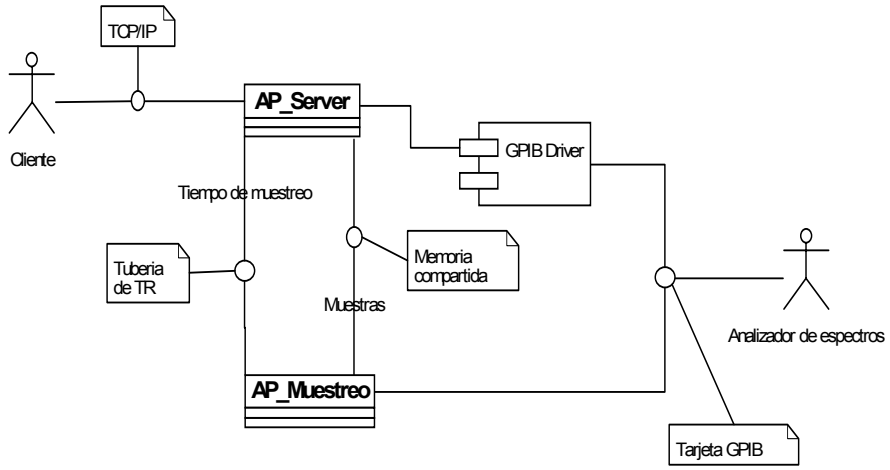


Figura 3-3 Clases del servidor.

En la figura 4-3 se observan las clases del servidor las cuales se mapean a procesos en el servidor, aunque estas clases no fueron escritas en un lenguaje orientado a objetos se tratarán como objetos para fines de diseño.

La clase AP_Server se encarga de recibir las peticiones que AP_GUI realiza a través de la red y traducirlas a peticiones hacia el analizador. Por lo que la clase AP_Server también se encarga que la comunicación con el analizador de espectros, para lo cuál utiliza un “device driver” libre, el cual se indica en el diagrama de clases como un componente, creado por Dr. Jens Thoms Törring[Törring J., 2003]. Este “device driver” es un

Eliminado: DISEÑO E IMPLEMENTACION

módulo de Linux y la forma en que se utiliza es creando un dispositivo, “/dev/Analizador”, el cuál es mapeado a la hora de que el módulo es cargado. Una vez cargado el módulo se puede acceder a este dispositivo con las funciones ANSI estándar “open(), write(), read() y close()”.

Por ejemplo:

- Se abre el dispositivo para lectura y escritura de la siguiente manera:

```
fd0=open("/dev/Analizador1", O_RDWR)
```

- Para configurarlo a un ancho de muestreo en el analizador de 0 y un tiempo de barrido de 20 milisegundos se utiliza el siguiente comando:

```
write(fd0,";MKPK HI;MKA?;\n", 15);
```

- Y para leer un dato enviado por el analizador se utilizaría la llamada “read” de la siguiente manera:

```
n =read(fd0, buf, BUFSIZE);
```

La clase AP_Muestreo es la encargada de realizar el muestreo en tiempo real del analizador por lo que está programada usando las funciones de tiempo real que proporciona RT-Linux. Para que esta clase se pueda comunicar con el analizador de espectros se implementaron funciones para manejar la tarjeta GPIB, lo anterior se debió a que el “device driver” no puede ser manejado en programación de tiempo real con RT-Linux.

Las causas por la que no puede ser usado fueron las siguientes:

- Hace uso de printf, el cuál no es soportado por RT-Linux.
- Hace uso de archivos de bitácora. Lo cuál introduce tiempos de acceso que no son predecibles.

Además la integración de algunas rutinas del “device driver” con el módulo de RT-Linux se complicó por lo que se decidió crear funciones para manejar la tarjeta GPIB que sólo incluyeran la funcionalidad mínima necesaria para tomar muestras de potencia del analizador desde la clase AP_Muestreo.

Como se observa en el diagrama de la figura 4-3 AP_Server se comunica con la tarjeta GPIB a través del “device driver” y AP_Muestreo se comunica con la tarjeta directamente a través sus funciones para manejar la tarjeta GPIB.

El uso del “device driver” bajado de Internet y las funciones para manejar la tarjeta GPIB se detallan a continuación:

Manejador	Funciones
“Device driver”	• Inicializar la tarjeta GPIB. • Enviar y recibir parámetros del analizador de espectro (Frecuencia central, nivel de referencia, ancho de frecuencia, frecuencia central, tiempo de barrido, filtro de resolución, filtro de video). • Copiar el desplegado del analizador de espectros (“snap”). • Copiar el patrón de radiación del analizador de espectros cuando éste lo adquirió.
Funciones manejadoras	• Se utilizan sólo para que AP_Muestreo adquiera cada muestra de potencia del analizador de espectros que en su conjunto forman el patrón de radiación.

Tabla 3-1 Funciones de los manejadores de la tarjeta GPIB.

Algunas de las características de las dos manejadores de la tarjeta GPIB se detallan a continuación:

Manejador	Características
“Device driver”	• Se integra al sistema operativo. Puede ser direccionado con una ruta en /dev. • Es fácilmente usado por cualquier aplicación. • Trata de implementar la

	funcionalidad de la tarjeta GPIB en su totalidad. • Se utiliza con llamados estándar. • No puede ser usado en STR.
Funciones manejadoras	• No se integra al sistema operativo sino que forman parte de la aplicación. • No se puede usar por otra aplicación. • Implementa funcionalidad mínima, sólo la que necesita la aplicación. • Se utiliza con funciones no estándar. • Pueden ser usadas en STR.

Tabla 3-2 Características de los manejadores de la tarjeta GPIB.

3.3.1 Detalle del diseño de las funciones para manejar la tarjeta GPIB

Para utilizar un dispositivo aparte de las opciones comentadas en la sección 1.5 existe otra opción que es la de acceso directo al dispositivo. Esta opción da la ventaja de que no se tiene que implementar todas las interfaces para el manejo de toda la funcionalidad del dispositivo por lo que el tiempo de desarrollo es menor y puede ser la opción más recomendada cuando no se piensa utilizar el mismo dispositivo para otras aplicaciones.

Se decidió crear funciones para manejar la tarjeta GPIB y tener el control directamente desde AP_Muestreo.

Debido a que no se contaba con información del manejo de la tarjeta a bajo nivel para poder desarrollar las funciones manejadoras se procedió a analizar el código fuente del "device driver" que proporciona el Dr. Jens Thoms Törring [Törring J., 2003] , en el driver se observó que la tarjeta se podía manejar de dos formas por DMA o por poleo, se decidió implementar las funciones utilizando poleo dado que era el más sencillo por que no involucraba técnicas más complejas como el manejo de interrupciones y el control del DMA.

En el diseño se asume que la tarjeta GPIB ya ha sido inicializada y sólo enviará y recibirá datos del dispositivo número 18 (el analizador de espectros). Por lo que dentro de las funciones está contenido el número de dispositivo a direccionar.

El driver creado es un drive mínimo que se compone de las siguientes funciones básicas: GPIBopen(), GPIBclose(), GPIBread(), GPIBwrite() y las siguientes funciones de apoyo: GPIBwaitOUT(),GPIBwaitIN(),GPIBwaitCMD(),GPIBsetATN().

La función “GPIBopen()” llama la atención del analizador llamando a la función “GPIBsetATN()” la cuál activa la línea ATN del bus de control GPIB. Después “GPIBopen()” escribiendo el comando 0x1f en el puerto 0x2bd coloca al analizador de espectros y a cualquier otro dispositivo que esté en el bus GPIB a modo remoto. En seguida se muestra el código de la función “GPIBopen()”, cabe mencionar que “GPIBsetATN()” hace un poleo a un puerto para ver si su acción tuvo éxito. En seguida se muestra el código de “GPIBopen()”:

```
000129: int GPIBopen()
000130: {
000131:         int timeout;
000132:         timeout=GPIBsetATN();
000133:         if (timeout)
000134:         {
000135:                 //rtl_printf("\n[GPIBopen]:%d",-1);
000136:                 return (-1);
000137:         }
000138:
000139:         outb_p(0x1f,0x2bd); //set REN
000140:         rtl_delay((long) 100000); //wait 100 us
000141:         return(0);
000142: }
```

Las funciones “GPIBread(), GPIBwrite()” envían una secuencia de comandos a la tarjeta GPIB para configurar al dispositivo 0, controlador, como oyente o hablante y al 18, analizador, como hablante u oyente respectivamente. Estas funciones utilizan a “CPIBwaitCMD()” para hacer un poleo y saber cuando la tarjeta está lista para enviar el siguiente byte de la secuencia de comandos. También utilizan a las funciones “GPIBwaitIN() y GPIBwaitOUT()” para hacer poleo y saber cuando la tarjeta está lista para enviar o recibir datos respectivamente.

En seguida se muestra el código fuente de la función “GPIBwaitOUT()” se observa en la línea 111 que hace un poleo del puerto 0x2b9 para saber si la tarjeta está lista para enviar otro dato:

```
000102: int GPIBwaitOUT()
000103: {
000104:         unsigned char status;
000105:         hrtime_t inicio, fin, dif, valido;
000106:
000107:         valido = (hrtime_t) 500000000;
000108:         inicio = gethrtime();
```

Eliminado: DISEÑO E IMPLEMENTACION

```
000109:         do
000110:         {
000111:                 status=inb_p(0x2b9);
000112:                 if ((status&0x04)==0x04)
000113:                 {
000114:                         //rtl_printf("\n[GPIBwaitOUT] return: -2");
000115:                         return -2;
000116:                 }
000117:                 fin = gethrtime();
000118:                 dif=fin-inicio;
000119:                 if (dif>valido)
000120:                 {
000121:                         //rtl_printf("\n[GPIBwaitOUT] return: -1");
000122:                         return -1;
000123:                 }
000124:         }while ((status & 0x02)!=0x02);
000125:         return 0;
000126: }
```

Las funciones manejadoras de la tarjeta GPIB quedan ocultas en AP_Muestreo, y son diseñadas para ser usadas sólo en esta aplicación. Por lo tanto éstas no pueden servir para otro propósito más que muestrear cada 75 milisegundos valores de potencia del analizador de espectros HP8566B con dirección GPIB 18.

Mecanismo	Identificación	Función
Tubería de tiempo real	/dev/rtf0	Recibir el tiempo en segundos que durará el muestreo (cadena ASCII). Una vez recibido el muestreo comienza.
Memoria compartida de tiempo real	muestras	Buffer en donde se colocan las muestras de potencia. Cada muestra está en código ASCII y separadas por <CR> y >LF>.

Tabla 3-3 Mecanismos de comunicación de AP_Muestreo

La comunicación con el proceso de AP_Muestreo, se hace a través de los mecanismos de comunicación entre procesos que se detallan en la tabla 3-3.

Eliminado: DISEÑO E IMPLEMENTACION

3.3.2 Determinación del periodo del muestreo

La única restricción que tenemos para el proceso de tiempo real es que las muestras sean tomadas en espacios de tiempo iguales, no se tiene ninguna restricción en cuanto al tamaño de este periodo, por lo que se midió varias veces el tiempo que se tardaba en tomar una muestra y se ajustó al más crítico. La gráfica de la figura 4-4 muestra las muestras tomadas.

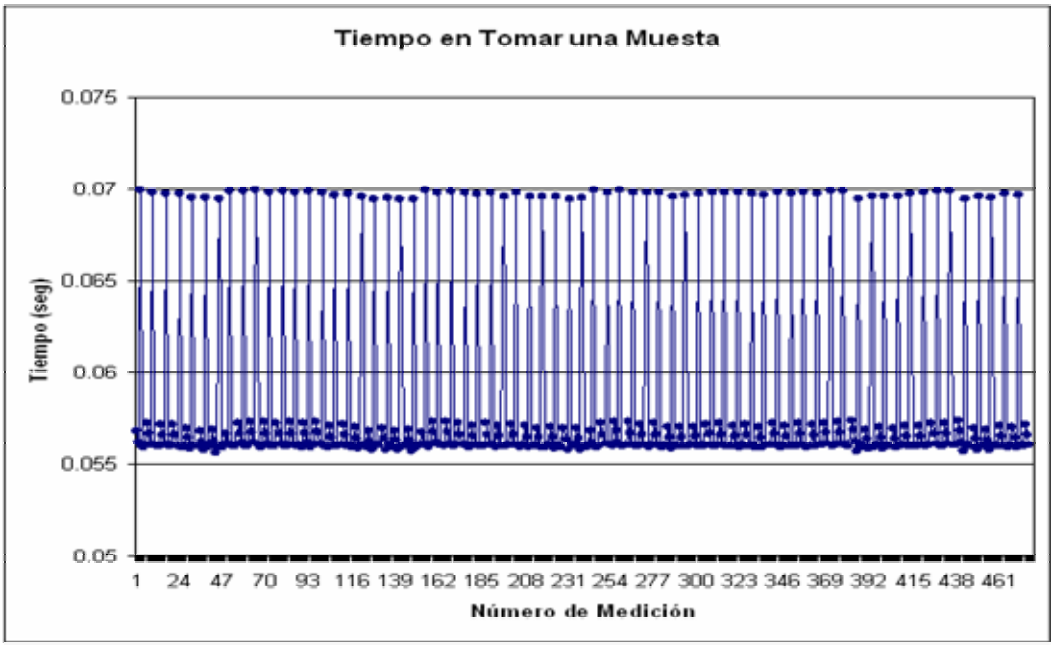


Figura 3-4 Tiempo en Tomar una Muestra.

El tiempo máximo de 477 mediciones fue de 0.06997725 segundos. Y el tiempo promedio de 0.0581927 segundos. Debido a que en el sistema es necesario utilizar un tiempo real duro se utilizó el tiempo máximo agregándole un margen, por lo que para el período del muestreo se utilizó 75 milisegundos.

En casos prácticos una tarea periódica puede ser caracterizada por su tiempo de ejecución Ci y su plazo “deadline” Di[Butazzo G., 1997]. Por lo que con los datos anteriores es posible trazar el cronograma de la tarea AP_Muestreo como se muestra en la figura 4-5.

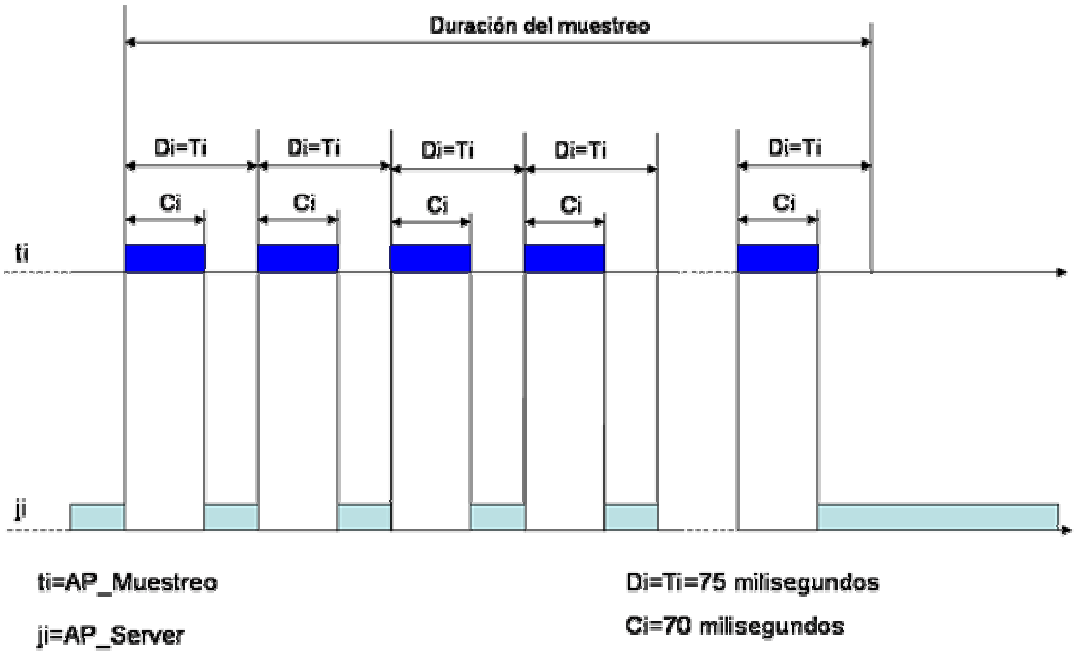


Figura 3-5 Cronograma de AP_Muestreo.

3.4 Diagrama de distribución

El sistema se diseñó para que se ejecute en dos partes: una corriendo en la máquina servidora, encargada de comunicarse con el analizador de espectros y de la obtención del patrón de radiación, así como del muestreo en tiempo real.

La otra parte está corriendo en la máquina cliente encargada de la comunicación con el usuario por medio de una “GUI”. Además esta parte también se encarga de la manipulación gráfica del patrón de radiación, de su evaluación y de la generación de reportes. También se encarga de la graficación en línea del patrón de radiación obtenido en tiempo real.

Eliminado: DISEÑO E IMPLEMENTACION

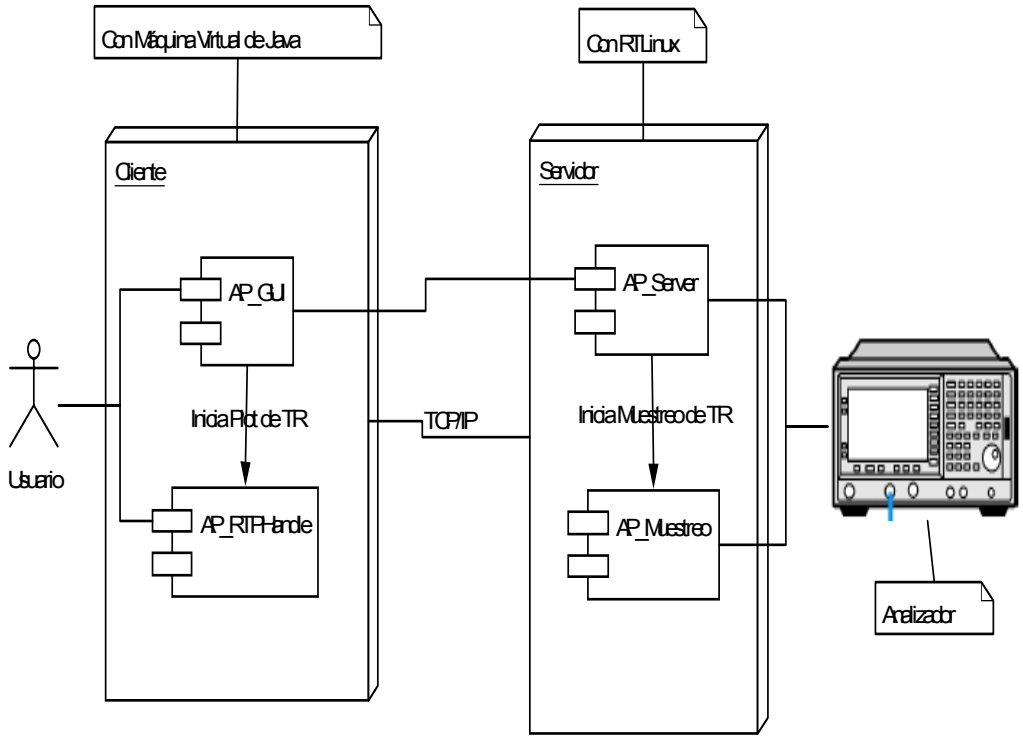


Figura 3-6 Diagrama de distribución.

En el diagrama de la figura 4-7 se observa que el sistema que se ejecuta en el cliente se divide en dos componentes (procesos) uno (AP_GUI) encargado de la interfaz gráfica de usuario y del desplegado y evaluación del patrón y otro (AP_RTPHandle) encargado de la graficación en línea del patrón adquirido por muestreo en tiempo real.

También el subsistema que se ejecuta en el servidor está dividido en dos componentes (procesos) uno, que no es programado en tiempo real (AP_Server), que se encarga de la comunicación con el analizador de espectros para su configuración, y del control de la adquisición del patrón por parte del analizador de espectros; y otro, que está programado con las facilidades de tiempo real de RT-Linux (AP_Muestreo), que se encarga de hacer un muestreo en tiempo real de los valores de potencia del analizador de espectros para generar el patrón de radiación.

CAPITULO 4 RESULTADOS

En este capítulo se exponen los resultados obtenidos con el sistema desarrollado, además se explica la forma que trabaja el sistema para adquirir un patrón de radiación. Así mismo, se expone la metodología encontrada para medir y evaluar el patrón de radiación de una antena y se finaliza el capítulo con las conclusiones obtenidas, recomendaciones y siguientes pasos.

4.1 Inicialización del analizador de espectros

La GUI se compone de dos secciones, la primera llamada “Initialization” consiste en controles para actualizar o recuperar los parámetros que son necesarios configurar en el analizador de espectros para medir el patrón de radiación. En seguida se muestra esta interfaz:

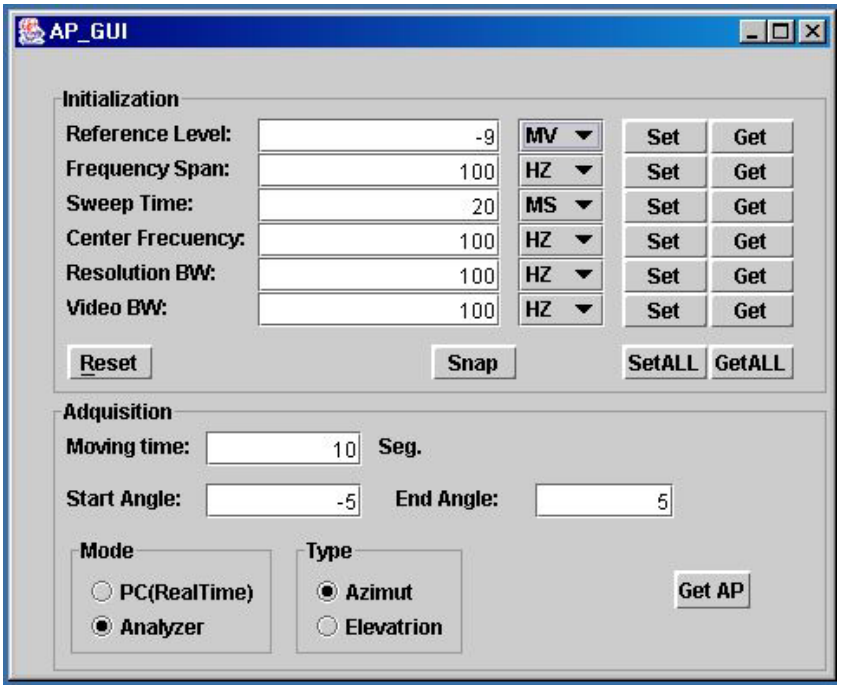


Figura 4-1 AP_GUI en ejecución.

Se tiene además los botones: [Reset] que sirve para inicializar el analizador de espectros, [Snap] para obtener una copia del despliegue del analizador de espectros, [SetALL] para enviar todos los parámetros al analizador de espectros y [GetALL] para obtener todos los parámetros del analizador de espectros.

Una vez que se envían todos los parámetros al analizador de espectros se procede a obtener una copia de su despliegue con el botón [Snap], para comprobar que la portadora sin modular está sintonizada. En seguida se muestra la ventana obtenida al pulsar [Snap] después de haber sintonizado el analizador de espectros a la portadora sin modular, donde se ve la respuesta a la portadora, Figura 4-2:

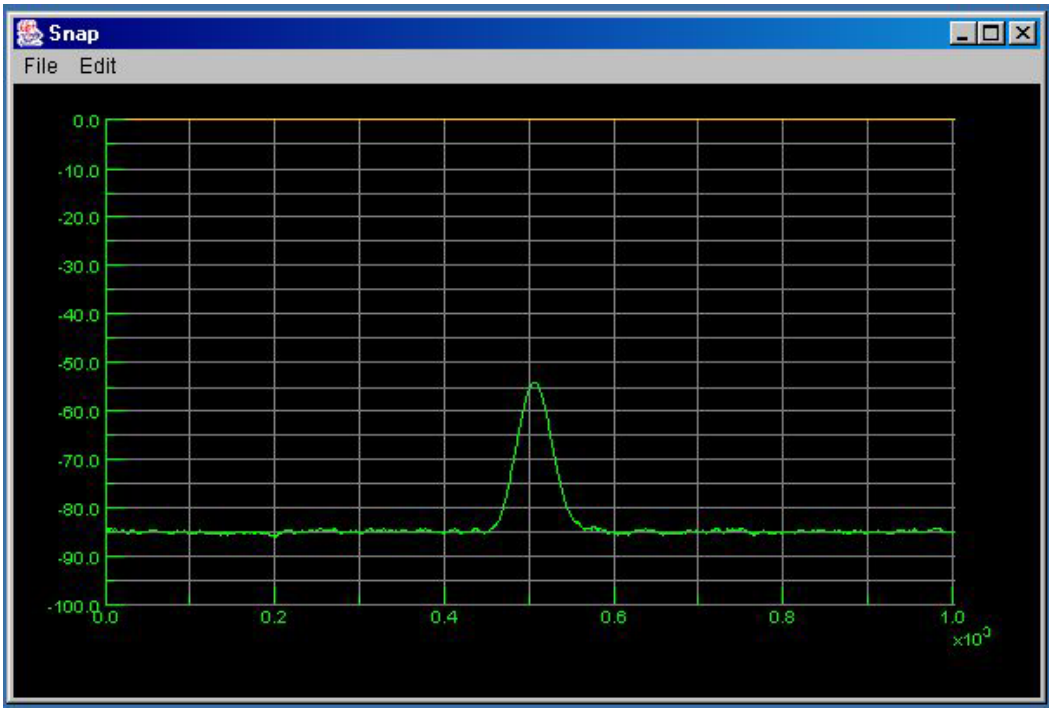


Figura 4-2 Resultado de Snap.

Hay que recordar que el analizador de espectros no está físicamente en el mismo sitio en el que se encuentra el operador por lo que es importante tener el botón de [Snap] para que se pueda verificar que el analizador está sintonizado.

4.2 Adquisición del patrón de radiación

Una vez sintonizado el analizador de espectros se procede a utilizar la sección de adquisición, en ésta se coloca el tiempo que dura la antena para recorrer el ángulo para el cual se medirá el patrón de radiación. Se selecciona quien hará el muestreo si el analizador de espectros o el servidor (PC). También se selecciona que tipo de patrón de radiación será si de azimut o de elevación. Una vez que se configura esta información se procede a obtener el patrón de radiación pulsando el botón de [GetAP]. En seguida se observa el resultado obtenido al medir un patrón de radiación de una antena marca Vertex de 7 m de diámetro:

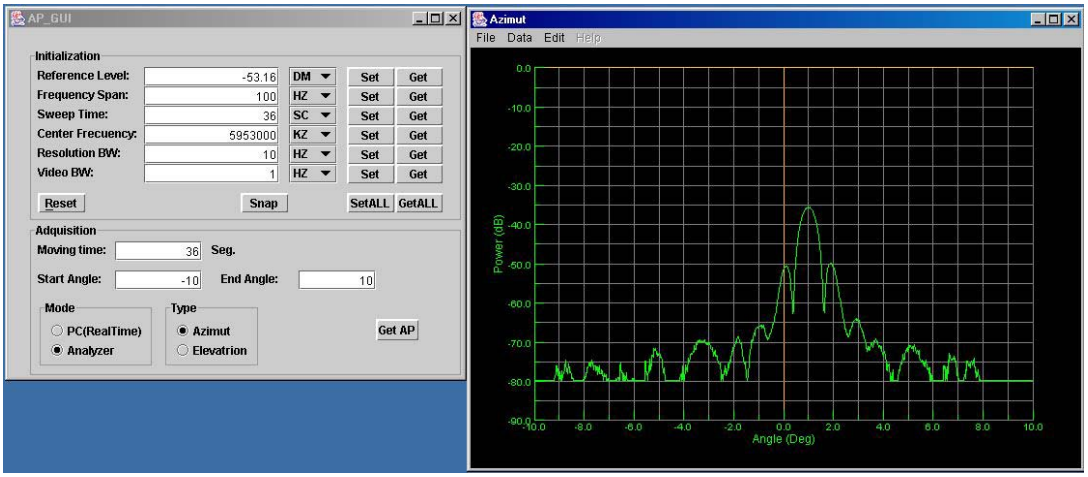


Figura 4-3 Patrón de Radiación obtenido con el sistema.

4.3 Manipulación del patrón de radiación

Una vez que se obtuvo el patrón de radiación se procede a normalizarlo y a centrarlo en el máximo, esto se logra con las opciones de [Next Max] [Previus Max] y [Normalize] del menú [Edit].

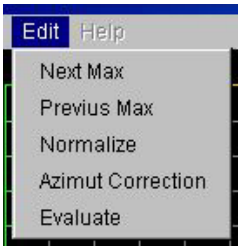


Figura 4-4 Menú Edit de AP_Plot.

En seguida se muestra el patrón centrado y normalizado:

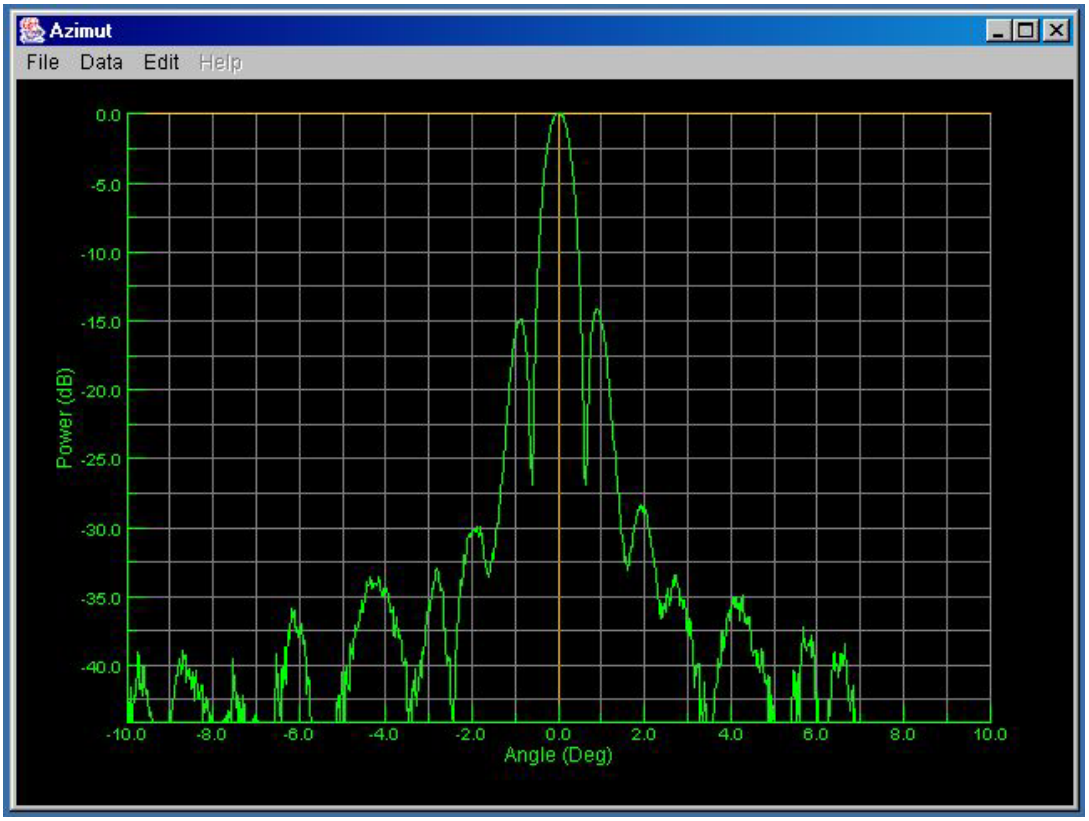


Figura 4-5 Patrón centrado y normalizado.

Una vez que el patrón de radiación se ha centrado y si el patrón es de azimut se debe corregir mediante la opción [Azimut correction] del menú [Edit], pero antes de esto se

debe de haber introducido la elevación del sitio donde se encuentra la antena para la cual se está llevando a cabo la medición, en nuestro caso 63.3 grados, este dato se introduce en el formulario que aparece al seleccionar la opción [Antenna Information] del menú [Data] como se muestra a continuación:

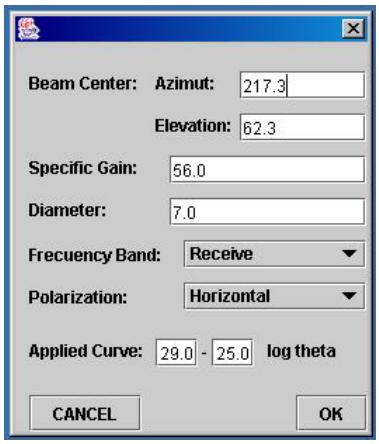


Figura 4-6 Formulario de la información de la antena de AP_Plot.

En seguida se muestra el patrón de radiación corregido:

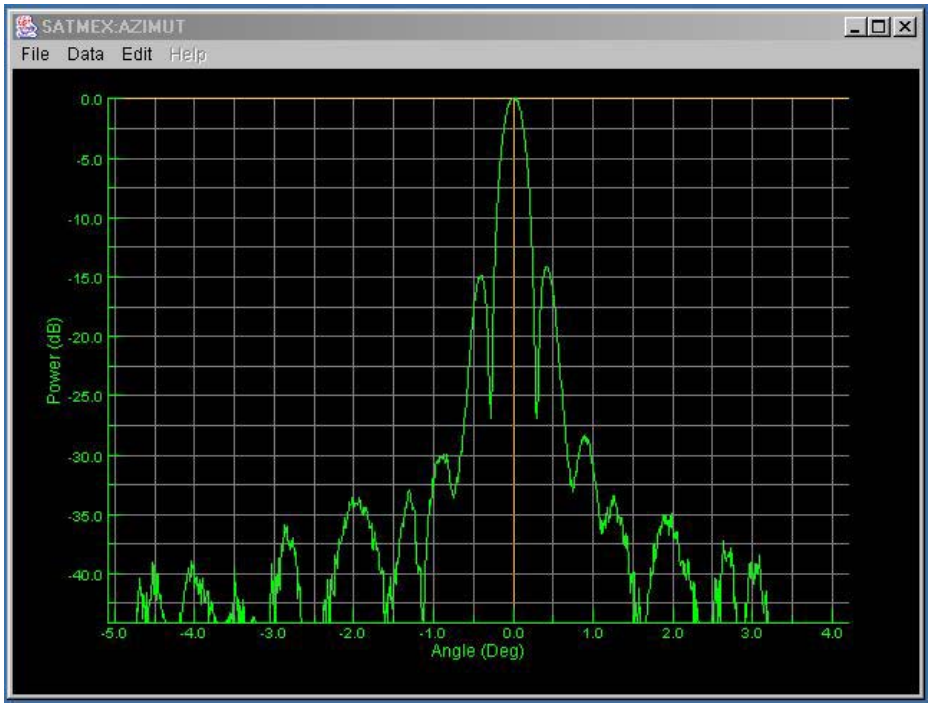


Figura 4-7 Patrón de radiación de azimut corregido.

4.4 Evaluación del patrón de radiación

Una vez corregido el patrón (sólo necesario para patrones de azimut) se procede a evaluarlo, previó a la evaluación se debe de haber llenado el campo de ganancia de la opción [Antenna Information] del menú [Data] (Figura 5-6), para nuestro caso 56.0. En seguida se utiliza la opción [Evaluate] del menú [Edit] (Figura 5-4) para evaluar al patrón de radiación, en seguida se muestra la evaluación obtenida:

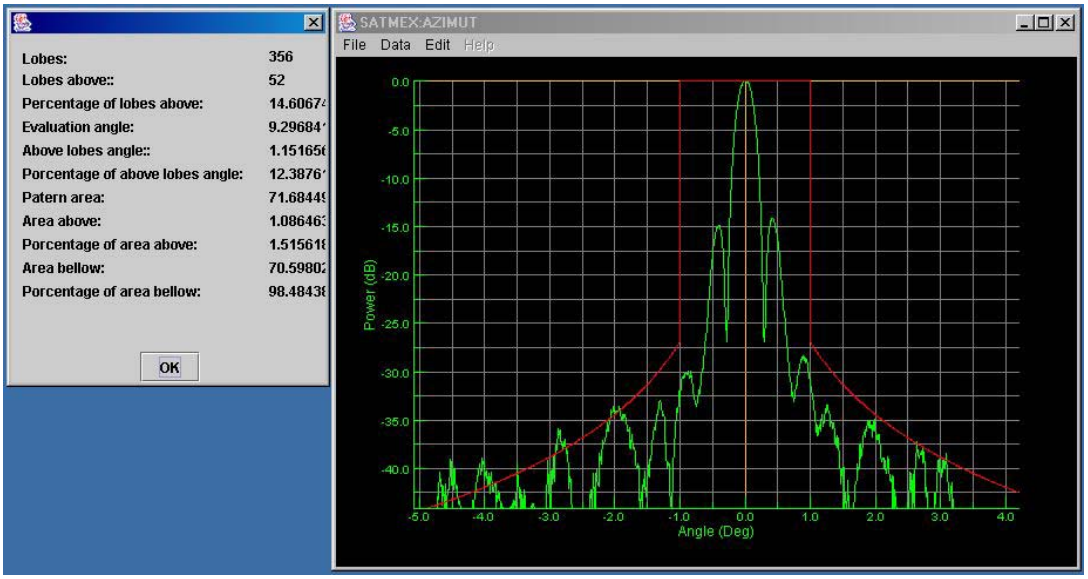


Figura 4-8 Resultado de la evaluación del patrón de radiación.

4.5 Reporte

Una vez que se tiene la evaluación del patrón de radiación se procede a imprimir un reporte para lo cual se deben de haber llenado los datos informativos de la opción [Antenna Information] (Figura 4-6) y [Test Information] (Figura 4-9) del menú [Data].

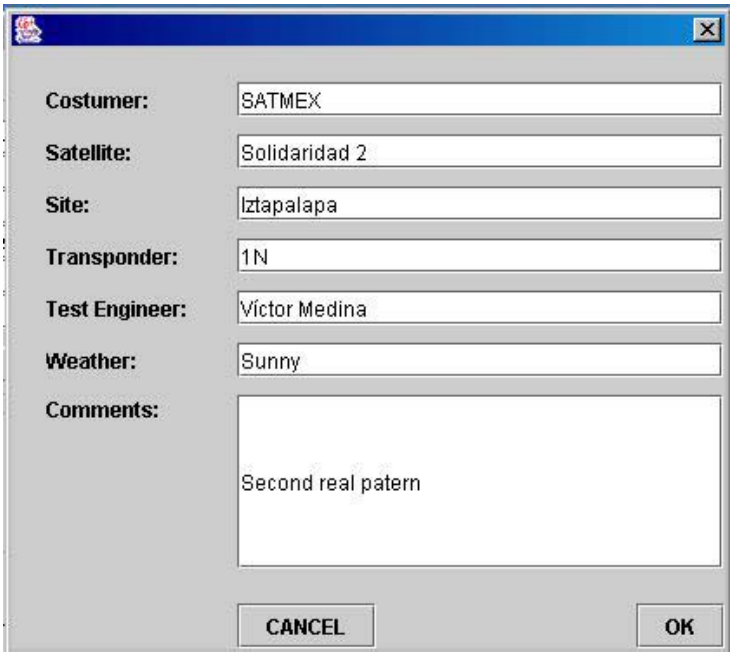


Figura 4-9 Formulario de la información de la prueba de AP_Plot.

Para generar el reporte se utiliza la opción [Print Report] del menú [File]. En seguida se muestra un reporte generado de la medición del patrón de radiación:

Figura 4-10 Reporte de la medición del patrón de radiación.

4.6 Metodología para medir el patrón de radiación.

Se obtuvo la siguiente metodología para medir el patrón de radiación:

1. Medir cuanto tiempo tarda la antena a evaluar en recorrer los grados que se moverá para adquirir el patrón, (típicamente de -5 a +5 grados de su posición de máximo apuntamiento).
2. En la antena a evaluar teniendo máximo apuntamiento enviar una portadora sin modular (CW) en una frecuencia acordada.
3. Utilizando una antena receptora colocar su bajada al analizador de espectros.
4. Utilizando el sistema sintonizar el analizador de espectros ajustando la frecuencia central, el nivel de referencia, la banda de frecuencia el tiempo de barrido, el filtro de resolución y el filtro de video.
5. Pedir un “SNAP” para obtener en el despliegue la portadora sin modular (CW) y verificar que el analizador esta sintonizado.
6. Mover la antena a evaluar a la posición inicial del movimiento (ángulo inicial).
7. En el sistema indicar si el tipo de patrón (azimut o elevación), el ángulo inicial y final, la duración del movimiento de la antena y el tipo de adquisición (Analizador o servidor).
8. Pulsar en el sistema el botón de inicio de adquisición [Get AP] al mismo tiempo que se empieza el movimiento de la antena.
9. Una vez adquirido el patrón se debe manipular para que se normalice, centre y si es un patrón de azimut se haga la corrección.
10. Se procede a evaluar el patrón con la opción del menú.
11. Se imprime un reporte.
12. FIN.

4.7 Conclusiones

Se cumplieron los objetivos planteados y que a continuación se describen:

- Se desarrolló un instrumento virtual para la medición, evaluación y generación de reportes del patrón de radiación.
- Se desarrolló una metodología utilizando el instrumento virtual y el equipamiento de SATMEX (analizador de espectro y una antena) para medir el patrón de radiación.
- Se utilizaron las facilidades de RT-Linux en la adquisición de datos de un analizador de espectros.
- Se desarrollaron funciones para manejar la tarjeta GPIB PCIIA de “National Instruments” para el sistema operativo RT-Linux.
- Se controló un analizador de espectros por medio de la interfaz IEEE-488 (GPIB).
- Se desarrolló un protocolo de comunicación entre el cliente y el servidor del instrumento virtual basado en el estándar SCPI. Con este protocolo se obtuvo una API que no está basada en llamadas a funciones sino en envío de mensajes a través de TCP/IP.

Adicionalmente se cumplieron con los siguientes requerimientos que marcaba la empresa:

- Se construyó un sistema económico basado en software libre (Java, RT-Linux, etc.) y utilizando la infraestructura y el equipamiento con los que cuenta la empresa.
- Se construyó un sistema cliente-servidor para hacer posible que el patrón de radiación pueda ser medido desde cualquier sitio de la empresa a través de la su red. Debido que el cliente está desarrollado en Java puede ejecutarse en casi cualquier máquina por lo que para poder medir el patrón remotamente basta que se tenga acceso a una estación de trabajo que pueda ejecutar la máquina virtual de Java.

4.8 Recomendaciones

Utilizar la adquisición del patrón de radiación de tipo “PC”, para este sistema en particular, se recomienda para una duración de muestreo mayor a 1 minuto ya que se obtienen más muestras de las que puede dar un muestreo del analizador de espectros 8566B que son 1001 datos.

Se recomienda desarrollar un servidor por tipo de analizador de espectros. Este servidor debe de ser pequeño y encapsular el manejo de la interfaz física y lógica con el analizador de espectros. Es decir debe de manejar y encapsular la interfaz física ya sea GPIB, RS232, ETHERNET, o cualquier otra; y la interfaz lógica, los comandos que se transmitirán al analizador de espectros, sean estos propietarios o algún estándar como el SCPI; y comunicarse con el cliente por medio del protocolo de comandos SCPI desarrollados en este trabajo.

Para obtener mejores resultados en el muestreo en tiempo real se recomiendan las siguientes acciones:

- Cambiar el servidor por una máquina más veloz.
- Cambiar el analizador de espectros por uno más moderno y veloz.
- Usar una tarjeta GPIB de bus PCI.
- Crear funciones manejadoras de la tarjeta GPIB para RT-Linux que utilice DMA.

4.9 Siguientes pasos

Desarrollar servidores para cada tipo de analizador de espectros con los que se cuenta.

Agregar más funcionalidad al cliente como:

- Cálculo de la ganancia de la antena.
- Medición del cross-pol y co-pol.
- Graficación en coordenadas polares.
- Graficación en tercera dimensión.
- Suavizado del patrón de radiación.

- Corrección automática de los ángulos inicial y final del movimiento de la antena para el patrón de radiación de azimut. (Dividiendo entre el coseno de la elevación de la antena a evaluar).

Eliminar el costo del analizador de espectros, tal vez, utilizando una tarjeta analizadora de espectros con adquisición de datos y filtrado digital. Hay que analizar costos.

BIBLIOGRAFIA

[ACM, 2004] ACM, (2004). "UML Fundamentals" Curso Virtual. Digital Think.

[Advanced, 2003] Advanced Horizons, Inc., (2003). http://www.ahinc.com/hhbus.htm

[Andrew, 1994] Andew Corporation, (1994). "Summary of FCC Rules for Testing Satellite Earth Station Transmit Antennas", Andrew Corporation, pp. 1-4.

[Belotserkovski, 1983] Belotserkovski, (1983). "Fundamentos de Antenas". Marcombo, pp. 11-20.

[Booch. et al, 1998] Booch, Grady, (1998). "Unified Modeling Language User Guide", Addison-Wesley.

[Brookshaw L., 2003] Brookshaw Leigh, (2003). "http://www.sci.usq.edu.au/staff/leighb/graph/". Graph

[Bruce E., 2002] Bruce Eckel, (2002). "Thinking in Java 3rd edition". Prentice Hall.

[Butazzo G., 1997] Butazzo Guorgio, (1997). "Hard Real-Time Computing Systems Predictable Scheduling Algorithms and Applications". Kluwer Academic Publisers.

[Burns. et al, 1997] Burns Alan and Wellings Andy. (1997) "Real-Time Systems and Programing Languages". Adison Wesley Longman.

[Cambridge, 1992] Cambridge University, (1992). "Numerical Recipies in C: The Art of Scientific Computing". Cambridge University Press, pp 133

[Dominguez A., 1995] Dominguez Garcia A., (1995). "Cálculo de Antenas". Alfaomega, pp 36-40.

[Douglas B., 2003] Douglass, Bruce Powel, (2003). "Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems". Addison Wesley, pp 528

[Douglas B., 2004] Douglass, Bruce Powel, (2004). "Real Time UML: Advances in The UML for Real-Time Systems, Third Edition", Addison Wesley, pp 752

[Guevara P., 2003] Guevara López Pedro, (2003). "Notas del curso: Laboratorio de tiempo real". CIC.

[Gutiérrez A. 2002] Gutiérrez Tornés Agustin, (2002). "Notas del curso de ingeniería de software". CIC

[Harbison S.et al, 1991] Harbison Samuel P. & Steele Guy L. Jr., (1991) "C a Reference Manual". Prentice Hall.

[HewlettPackard, 1982] Hewlett Packard, (1982). "Tutorial Description of the Hewlett-Packard Interface Bus", Hewlett Packard.

[HewlettPackard, 1984] Hewlett-Packard, (1984). "8566B SPECTRUM ANALYZER OPERATING AND PROGRAMMING MANUAL". Hewlett-Packard, pp. 537.

[Holler M., 2000] Holler Mirko , (2000). "Real Time Linux 2.0 Instalation & Examples". http://midas.psi.ch/rtlinux , pp 1-10.

[Kernighan B. et al, 1987] Kernighan Brian W. & Pike Rob, (1987). "El Entorno de Programación UNIX". Prentice Hall, pp 209-237.

[Kernighan B. et al, 1978] Kernighan Brian W. & Ritchie Dennis M., (1978). "El Lenguaje de Programación C". Prentice Hall.

[Kraus J., 1984] Kraus John D., (1984). "Electromagnetismo". McGraw Hill, pp.405-412.

[Kraus J., 1988] Kraus John D., (1988). "Antennas". McGraw Hill, pp 20-23, 762-767, 807-809.

[Lemay L. et al, 1996] Lemay Laura & Perkins Charles L.., (1996). "Teach Yourself Java in 21

days". SAMS.

[Manjares R. 2000] Manjares S. Jorge Robeto, 2000 "Notas del curso programación orientada a objetos". CIC

[Márquez F., 2001] Márquez García Francisco Manuel, (2001). "UNIX Programación avanzada (segunda edición)". Alfaomega, pp 191-238, 333-439.

[Medel J. et al, 2002] Medel Juárez José de Jesús, Guevara López Pedro y Barrón Fernández Ricardo, (2002). "Interacción entre Sistemas de Tiempo Real y Sistemas en Línea" CIC, pp 1-6.

[Mitchell M. et al, 2001] Mitchell Mark, Oldman Jeffrey and Samuel Alex, (2001). "Advanced Linux Programing". Pearson Education.

[Morgan A. et al, 1988] Morgan Michael A. and Burt Dennis, (1988). "Field Testing An Earth Station for Two Degree Compliance", Andrew Corporation, pp. 1-7.

[Perales A., 2001] Perales Briseño Alejandro, (2001). "Diseño y Desarrollo de un Osciloscopio Virtual Sobre un Sistema Operativo En Tiempo Real" Tesis de Maestria. CIC.

[RedHat, 2003] Read Hat Linux, (2003). "http://www.readhat.com/".

[Ripoll I., 1998] Ripoll Ismael, (1998). "Tutorial de Real Time Linux", pp 1-58.

[Rubini A. et al, 2001] Rubini Alessandro & Corbet Jonathan, (2001). "Linux Device Drivers, 2nd Edition". O'Reilly.

[Salmerón M., 1981] Salmerón María José, (1981). "Radiación, propagación y antenas". Trillas, pp 26.

[Sourceforge, 2003] Sourceforge, (2003) "http://linux-gpib.sourceforge.net/".

[Suárez S., 2001] Suárez Guerra, Sergio, (2001). "Notas del curso: Sistemas de tiempo real". CIC.

[Törring J., 2003] Törring Dr. Jeans Thoms, (2003). "http://www.physik.fu-berlin.de/~toerring/".

[UIT465, 1993] UIT, (1993).RECOMENDACIÓN UIT-R S.465-5 "Diagrama de radiación de referencia de estación terrena para utilizar en la coordinación y evaluación de las interferencias, en la gama de frecuencias comprendidas entre 2 y unos 30 GHz", pp. 1-2.

[UIT732, 1993] UIT, (1993).RECOMENDACIÓN 732 "Método para el tratamiento estadístico de las crestas de los lóbutlos laterales de las antenas de estación terrena", pp. 1-2.

[UIT580, 1993] UIT, (1993).RECOMENDACIÓN UIT-R S.580-5 "Diagramas de radiación que han de utilizarse como objetivos de diseño para las antenas de las estaciones terrenas que funcionan con satélites geoestacionarios", pp. 1-3.

[Vicuña R. et al, 2001] Vicuña Heredia Juan Roberto y Guevara López Pedro, (2001) "Elaboración de Administradores de Recursos en QNX para Aplicaciones de Sistemas de Tiempo Real", CIC

[Vicuña R., 2001] Vicuña Heredia Juan Roberto, (2001). "Guia para el Desarrollo del Administrador de Recursos para una Terjeta de Entrada-Salida De Señales en QNX ty Su Implementación en un Sistema de Control en Tiempo Real". CIC

[Wilshire P., 2000] Wilshire Phil, (2000). "Instaling RT-Linux", RT-Linux Documentation Group, pp 1-9.

[Yodaiken V. et al, 1996] Yodaiken Victor and Barabanov Michael, (1996), "Real Time Linux". RT-Linux Documentation Group, pp-1-9.

[Yodaiken V. et al, 1999] Yodaiken Victor and Barabanov Michael, (1999), "RTLinux Version Two", VJY Associates LLC, pp-1-14.

[Zsolt P., 2003] Zsolt Dr. Pápay , (2003). "http://www.hit.bme.hu/people/papay/edu/GPIB/tutor.htm".

ANEXO A INTRODUCCION A SCPI

La especificación SCPI define un lenguaje de programación usado para controlar instrumentos de medición y pruebas como osciloscopios, generadores de funciones, fuentes de poder y analizadores de espectro. Estos instrumentos implementan SCPI en su firmware.

SCPI es en algún sentido una adición a la especificación IEEE 488.2. La especificación IEEE 488.2 define comandos generales mientras que SCPI proporciona comandos requeridos para la operación de ciertos tipos de instrumentos.

El consorcio inicial de SCPI fue conformado por HP, Tektronix, Fluke, Philips, Wavetek, Ramal-Dana, Keithley, Cruel & Kjaer y Nacional Instruments. El consorcio de SCPI ahora mantiene la definición de SCPI y documento formal que la describe.

Los beneficios de SCPI es compatibilidad, esto es interoperabilidad entre diferentes instrumentos. El mismo comando que realiza una cierta función en un instrumento realizará exactamente la misma función en un instrumento completamente diferente, siempre y cuando ambos compartan esta capacidad. Un programa diseñado para controlar cierto tipo de instrumento, como un generador de funciones, funcionará con un generador de funciones de un diferente vendedor con pocos o ningún cambio.

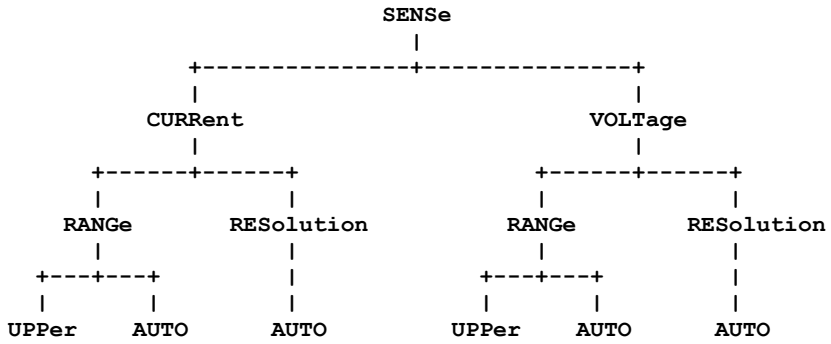
SCPI esta diseñado con comandos en diferentes niveles de generalidad para ayudar a proporcionar esta compatibilidad. Un comando de alto nivel como MEASURE:VOLTAGE:AC? (“lee el voltaje de AC”) funcionará en osciloscopio o un voltímetro. Al mismo tiempo SCPI proporciona comandos para controlar instrumentos a nivel bajo que permiten la programación de instrumentos de precisión, pero que talvez n o trabajen en otro instrumento que no sea de precisión.

SINTAXIS DE LOS COMANDOS SCPI

Uno de los mejores principios en los que se basa SCPI es que si existe un comando que pone un valor existe una petición similar que lee el valor.

SCPI organiza los comandos en varios conjuntos que se mapean a subsistemas en el instrumento. Los comandos en cada subsistema están definidos en un orden jerárquico similar a la estructura jerárquica del sistema de archivos encontrado en las computadoras. En SCPI esta estructura de comandos es llamada “árbol de comandos”.

Un ejemplo simplificado, para el comando SENSe que se implementas en los multímetros digitales, se muestra en seguida:



El árbol de comandos se describe con nomenclatura similar a la que se usa para los sistemas de archivos. El comando inicial del árbol es llamado comando raíz, y el comando del subsistema esta en tutas de ramas del árbol. Por ejemplo, una ruta se define por la siguiente secuencia de comandos:

```
:SENSe:Voltaje:RANGe:AUTO
```

La secuencia anterior coloca el multímetro para que lea voltaje y use un rango automático. Nótese que los dos puntos (“:”) se usan como separadores en la ruta. Otra ruta es:

```
:SENSe:CURRent:RANGe:UPPer
```

La ruta anterior coloca el multímetro para que lee corriente y usa el rango superior de corriente. Nótese que no se necesita enviar la ruta completa de un comando cada vez.

Cuando se decodifica un comando de un subsistema el parser mantiene la ruta de comandos actual, algo parecido a como al directorio actual en los sistemas de archivos jerárquicos, esto especifica el bloque del subsistema del instrumento para el cual se están decodificando comandos.

El parser navega en el árbol de acuerdo a las siguientes reglas:

- Después de un encendido o de el comando *RST el apuntador se pone a la raíz.

- Un terminador de mensaje, usualmente un <newline> o un carácter (line-feed), también pone el apuntador a la raíz.
- Un carácter ":" mueve el apuntador un nivel abajo en el árbol. Si es el primer carácter en la secuencia especifica la raíz.
- Un carácter ";" separa dos comandos en el mismo subsistema de comandos no mueve el apuntador.
- Espacios en blanco y tabuladores son ignorados. Sin embargo espacios en blanco son requeridos para separar parámetros del comando. Por ejemplo `SOURce:VOLTage6.2` es incorrecto y se debe usar `SOURce:Voltaje 6.2`
- Las comas son usadas para separar parámetros en un mismo comando.
- Comandos comunes como *RST, no son comandos del subsistema y no son interpretados como parte de la ruta.

Por ejemplo:

```
:SENSe:VOLTage ; RANGe:AUTO ; RESolution:AUTO
```

Es lo mismo que:

```
:SENSe:VOLTage:RANGe:AUTO
:SENSe:VOLTage:RESolution:AUTO
```

EL árbol de comandos es especificado concisamente por medio de una tabla e comandos del subsistema que define el comando y sus parámetros. Por ejemplo el árbol de comandos de SENSE se evalúa por medio de la siguiente tabla

Command	**Parameters**
[:SENSe]	
:CURRent	
:RANGe	
:AUTO	Boolean or ONCE
[:UPPer]	numeric
:RESolution	numeric
:AUTO	Boolean or ONCE
:VOLTage	
:RANGe	

```
          :AUTO          Boolean or ONCE
        [:UPPer]         numeric
      :RESolution        numeric
          :AUTO          Boolean or ONCE
  ____________________________________
```

La jerarquía del comando está dada por el nivel de sangría en la columna del comando.

La mayoría de los comandos se muestran como cadenas con letras en mayúscula seguidas de letras en minúscula. Esta mezcla no es parte de la definición SCPI. SCPI no es sensitiva a mayúsculas. Lo que las letras en minúscula significan es que estos caracteres son opcionales y se pueden descartar si se desea. Por ejemplo:

```
:SENSe:CURRent:RANGe:AUTO ON
```

Es lo mismo que:

```
:SENS:CURR:RANG:AUTO ON
```

Los comandos dentro de paréntesis cuadrados son comandos implícitos, significa que si un comando en ese nivel de la ruta no es especificado se asume. Por ejemplo:

```
:SENSe:VOLTage:RANGe:UPPer 6.5
```

Es lo mismo que:

```
:VOLTage:RANGe 6.5
```

Para la mayoría de los comandos que ponen un valor existe su equivalente comando de petición que lee el valor. El comando de petición es el mismo que el comando que coloca el valor pero termina con el carácter “?”. Por ejemplo:

```
:SENSe:VOLTage:RANGe
```

Tiene su comando de petición equivalente:

```
:SENSe:VOLTage:RANGe?
```

Si la tabla de comandos contiene un comando que termina con el carácter “?” significa que solo existe el comando de petición.

APENDICE A. COMANDOS SCPI DISEÑADOS Y SUS RESPUESTAS

COMANDO	**DESCRIPCION**
INIT	Inicializa el analizador.
AP:GET:AN:*valor*	Adquiere el patrón de radiación por medio del analizador de espectros. El parámetro *valor* es el tiempo en segundos que durará el muestreo.
AP:GET:PC:*valor*	Adquiere el patrón de radiación por medio de muestreo en tiempo real. El parámetro *valor* es el tiempo en segundos que durará el muestreo.
AN:SET:RL:*valor*:[DM,MV,UV]	Ajusta el nivel de referencia (Referens Level) en el analizador de espectros
AN:SET:SP:*valor*:[HZ,KZ,MZ,GZ]	Ajusta el ancho de barrido (Frecuency Span) en el analizador de espectros.
AN:SET:ST:*valor*:[SC,MS,US]	Ajusta el tiempo de barrido (Sweep Time) en el analizador de espectros.
AN:SET:CF:*valor*:[HZ,KZ,MZ,GZ]	Ajusta la frecuencia central (Center Frequency) en el analizador de espectros.
AN:SET:RB:*valor*:[HZ,KZ,MZ,GZ]	Ajusta el ancho de banda de resolución (Resolution Band Width) en el analizador de espectros.
AN:SET:VB:*valor*:[HZ,KZ,MZ,GZ]	Ajusta el ancho de banda de video (Video Band Width) en el analizador de espectros.
AN:GET:RL	Obtiene el nivel de referencia (Referens Level) del analizador de espectros
AN:GET:SP	Obtiene el ancho de barrido (Frecuency Span) del analizador de espectros.
AN:GET:ST	Obtiene el tiempo de barrido (Sweep Time) del

	analizador de espectros.
AN:GET:CF	Obtiene la frecuencia central (Center Frequency) del analizador de espectros.
AN:GET:RB	Obtiene el ancho de banda de resolución (Resolution Band Width) del analizador de espectros.
AN:GET:VB	Obtiene el ancho de banda de video (Video Band Width) del analizador de espectros.
AN:GET:SNAP	Obtiene un trazo del analizador de espectros.

Tabla A-1 Comandos SCPI para la comunicación cliente-servidor.

Al final de cada comando se envía un retorno de carro (<CR>) y un salto de línea (<LF>).

COMANDO	**RESPUESTA**
INIT	OK
AP:GET:AN:*valor*	*-77.00*<CR><LF> *-95.00*<CR><LF> . . . *-76.10*<CR><LF> *-94.70*<CR><LF> END<CR><LF>
AP:GET:PC:*valor*	*-77.00*<CR><LF> *-95.00*<CR><LF> . . . *-76.10*<CR><LF> *-94.70*<CR><LF> END<CR><LF>

AN:SET:RL:*valor*:[DM,MV,UV]	Ninguna respuesta
AN:SET:SP:*valor*:[HZ,KZ,MZ,GZ]	Ninguna respuesta
AN:SET:ST:*valor*:[SC,MS,US]	Ninguna respuesta
AN:SET:CF:*valor*:[HZ,KZ,MZ,GZ]	Ninguna respuesta
AN:SET:RB:*valor*:[HZ,KZ,MZ,GZ]	Ninguna respuesta
AN:SET:VB:*valor*:[HZ,KZ,MZ,GZ]	Ninguna respuesta
AN:GET:RL	0:DM<CR><LF>
AN:GET:SP	20000000000:HZ<CR><LF>
AN:GET:ST	.500000:SC<CR><LF>
AN:GET:CF	12000000000:HZ<CR><LF>
AN:GET:RB	3000000:HZ<CR><LF>
AN:GET:VB	3000000:HZ<CR><LF>
AN:GET:SNAP	*-77.00*<CR><LF> *-95.00*<CR><LF> . . . *-76.10*<CR><LF> *-94.70*<CR><LF> END<CR><LF>

Tabla A-2 Respuesta a los Comandos SCPI.

AL final de cada línea de la respuesta se recibe un retorno de carro <CR> y un salto de línea <LF>.

APENDICE B. DETALLE DE LAS CLASES

Detalle de la clase AP_GUI

En la figura B-1 se muestra el diagrama de la clase AP_GUI en este diagrama se muestran sus atributos y sus funciones, para simplificar el diagrama solo se muestran las funciones y clases que no tiene que ver con el manejo de ventanas:

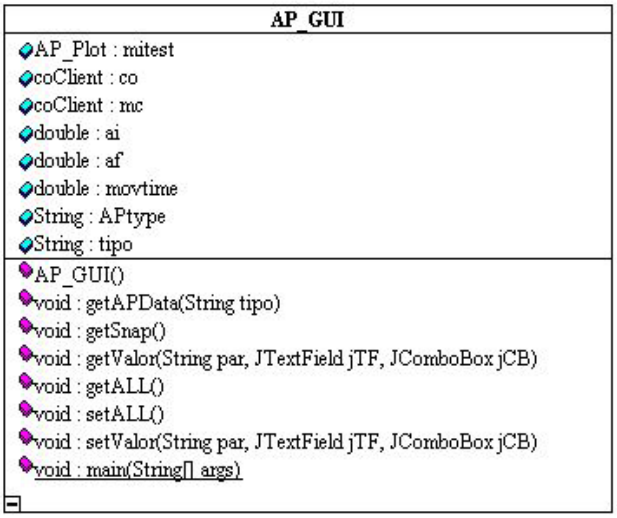


Figura B-1 Variables y funciones miembros de la clase AP_GUI.

En la tabala B-1 son descritas las variables miembro de la clase y en la tabla B-2 se describen las funcionesmiembreo.

Variable	**Descripción**
mitest	Un objeto del tipo **AP_Plot** que se usará para mostrar el Patrón de radiación.

co	Objeto de tipo **coClient** usado para establecer la comunicación TCP/IP con el servidor.
mic	Objeto de tipo **coClient** usado para establecer la comunicación con el proceso que se encarga de dibujar el patrón en tiempo real (**AP_RTPHandle**).
ai, af	Contiene el ángulo inicial y final del movimiento de la antena.
movetime	Contiene el tiempo que tarda la antena en llegar de **ai** a **af.**
APtype	Contiene el tipo de patrón de radiación que se obtendrá puede ser de azimut o de elevación.
tipo	Es quien hará el muestreo el analizador de espectros (AN) o el servidor (PC).

Tabla B-1 Variables miembro de la clase AP_GUI.

Función	**Descripción**
AP_GUI()	Constructor
getAP_Data(String tipo)	Obtiene las muestras del patrón de radiación, es decir procesa la respuesta a los comandos "**AP:GET:AN**" y "**AP:GET:PC**" par lo cuál crea una instancia de la clase **AP_Plot** y le envía el muestreo para el primer comando o le envía el muestreo a **AP_RTPHandle** en el caso del segundo comando. El argumento es el tipo de muestreo "AN" o "PC".
getSnap()	Procesa la respuesta la comando "**AN:GET:SNAP"** instanciando la clase **AP_GetSnap** y enviándole el trazo para

	que se muestre.
getValor(String par, JTextField jTF, JComboBox jCB)	Obtiene los valores de los parámetros del analizador y los coloca en las campos de texto de la "GUI" . Es decir procesa la respuesta al comando **"AN:GET:*XX"*** y lo conforma**.** También muestra en la "GUI" el resultado obtenido. El parámetro *par* es el parámetro del que se quiere el valor (RL, SP, ST, CF, RB o VB); *jTF* es el campo de texto de la "GUI" en donde se mostrará y *jCB* es la caja de elecciones donde se muestran las unidades.
getALL()	Hace llamados a getValor() para obtener todos los parámetros.
setAll()	Hace llamados a setValor() para ajustar todos los parámetros en el analizador.
setValor(String par, JTextField jTF, JComboBox jCB)	Conforma el comando "AN:SET:*XX"* para enviarselo al servidor y que se ajuste el parámetro en el analizador. El parámetro *par* es el parámetro que se quiere ajustar (RL, SP, ST, CF, RB o VB); *jTF* es el campo de texto de la "GUI" de donde se toma el valor al que se ajustará el parámetro y *jCB* es la caja de elecciones de donde se toma las unidades.
main()	Es la función principal de entrada.

Tabla B-2 Funciones de la clase AP_GUI.

En la figura B-2 se muestran las variables y funciones de la clase AP_GUI que conforman la GUI estas fueron generadas automáticamente por el programa NETBeans.

Figura B-2 Variables y funciones miembros para la GUI de la clase AP_GUI.

Las variables se refieren a los componentes de la GUI y las funciones son acciones que se ejecutan cuando algún botón de la GUI es presionado o algún objeto es seleccionado. La función **initComponents()** es llamada para iniciar los componentes de la GUI. En la figura B-3 se muestra el diseño de la GUI:

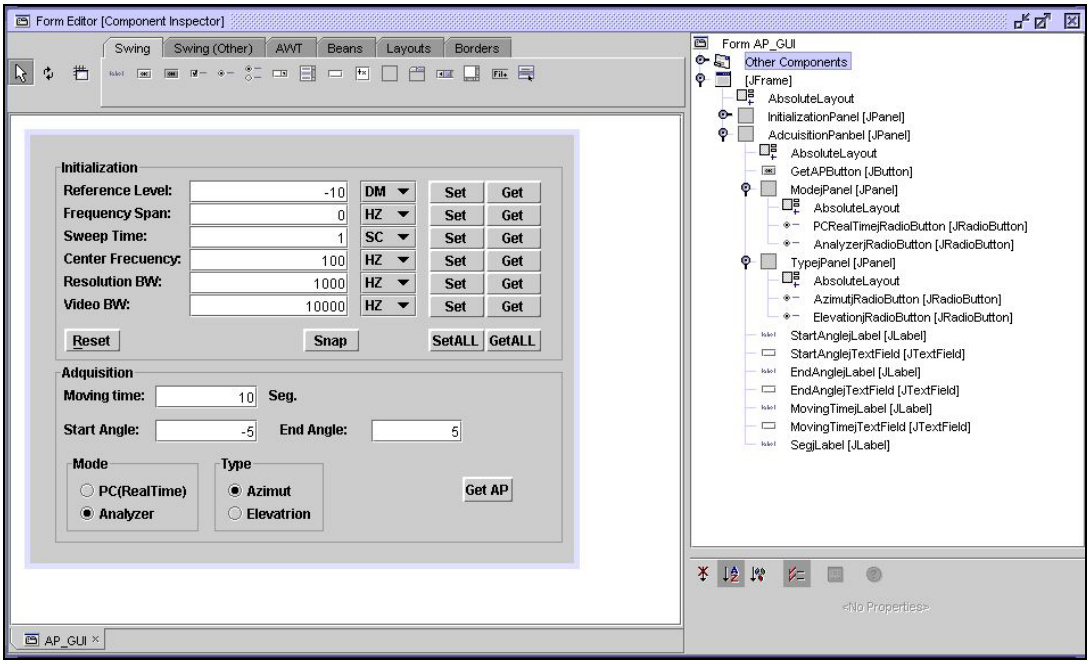


Figura B-3 GUI de la clase AP_GUI.

Detalle de la clase AP_Plot

En la figura B-4 se muestra el diagrama de la clase AP_Plot, para simplificar el diagrama solo se muestran las funciones y clases que no tiene que ver con el manejo de ventanas:

Figura B-4 Variables y funciones miembros de la clase AP_Plot.

En seguida son descritas las variables miembro de la clase:

Variable	**Descripción**
graph	Es el objeto que sirve para manipular la gráfica del patrón de radiación es tomado del paquete “graph”.
APatern	Contiene los puntos muestrados.
Envolvente	Contiene los puntos de la envolvente usada para la evaluación del patrón de radiación.
xayix, yaxis	Definen los ejes de la gráfica.
changed	Lógico usado para saber si hay algún cambio en la gráfica.
name	Nombre del archivo a abrir o salvar.
elementos	Lógico para saber si hay puntos muestreados.
mitipo	Tipo de patrón de radiación (azimut o elevación).
azimut, elevation, gain, diameter, frecuancyband, polarization, A, B	Parámetros de la antena a la que se le esta midiendo el patrón e radiación. *A* y *B* son los coeficientes de la curva de evaluación A-BLog(ángulo).
costumer, satellite, site, transponder, enginer, weather, comments	Parámetros de la medición, *enginer* es el nombre del ingeniero que esta realizando la medición.
TAP	Contiene una copia de los puntos muestreados cuando se esta manipulando el patrón de radiación.
lmax	El último máximo encontrado.
yamax	Lógico que dice si ya fue encontrado un máximo.
firstmax	Lógico que dice si es el primer máximo.
inc	Incremento en los valores de x.

mixmax	X máximo que se dibujará en la gráfica.
mixmin	X mínimo que se dibujará en la gráfica.
miymax	Y máximo que se dibujará en la gráfica.
miymin	Y mínimo que se dibujará en la gráfica.
nlobes	Número de lóbulos que tiene el patrón.
nlobesabove	Número de lóbulos arriba de la curva de evaluación.
plobes	Porcentaje de lóbulos arriba de la curva de evaluación.
Tangle	Angulo total de la medición.
AngleAbove	Suma de los ángulos en la que el patrón de radiación sobrepasa la curva de evaluación.
pangle	Porcentaje de ángulo en el que el patrón de radiación sobrepasa la curva de evaluación.
area	Area del patrón de radiación.
areaabove	Area del patrón de radiaciónque sobrepasa la curva de evaluación.
areabellow	Area del patrón de radiación que no sobrepasa la curva de evaluación.
pareaabove	Porcentaje del área del patrón de radiación que sobrepasa la curva de evaluación.
pareabellow	Porcentaje del área del patrón de radiación que no sobrepasa la curva de evaluación.

Tabla B-3 Variables miembro de la clase AP_Plot.

En seguida se describen las funciones de la clase:

Función	**Descripción**
AP_Plot(String tipo)	Constructor. El argumento *tipo* es el tipo de

	patrón a adquirir (**azimut** o **elevation**).
exitf()	Función para salir de la ventana.
addPuntos(double datos[], int np)	Adiciona un conjunto de puntos al patrón de radiación, *datos* es un arreglo que contiene los pares (x,y) de puntos y *np* es el número de puntos.
addPunto(double x, double y)	Adiciona un punto al patrón de radiación.
envolvente(double x)	Regresa el valor de y de la envolvente para un determinado *x*.
drawEnvolvente()	Dibuja la envolvente en la gráfica. La envolvente depende de la ganancia de la antena.
evaluate(Report mirep)	Evalúa el patrón de radiación. Llama a **drawEnvolvente()**,**getLobes()**,**getAngles()** y **getAreas().** Se utiliza el un objeto **Report** para a él informen las demás rutinas de sus resultados.
getLobes(Report mirep)	Obtiene los lóbulos que sobrepasan la curva de evaluación y lo informa a **mirep.**
getAngles(Report mirep)	Obtiene la suma de los ángulos en los que el patrón de radiación sobrepasa la curva de evaluación y lo informa a **mirep.**
getAreas(Report mirep)	Obtiene el área del patrón de radiación que sobrepasa la curva de evaluación y lo informa a **mirep.**
saveAP(String Nombre)	Salva el patrón de radiación con el nombre contenido en **Nombre**. Se salva también los datos de la antena y los datos de la medición.
loadAP(String Nombre)	Recupera un patrón de radiación con el nombre contenido en **Nombre**.
checkChange()	Verifica si el patrón de radiación a sufrido

	cambios y pregunta si se desea salvarlo antes de realizar alguna manipulación.
main(String args[])	Cuando **AP_Plot** es llamada sola esta es la rutina de entrada.
printReport (PrintJob pj)	Imprime un reporte.
printPlot (PrintJob pj)	Imprime la gráfica actual.
centerNext()	Centra el patrón de radiación al siguiente máximo.
centerPrevius()	Centra el patrón de radiación al máximo anterior.
Normalize()	Hace que el máximo coincida con el 0 del eje y.
azimutCorrection()	Corrige un patrón de radiación de azimut multiplicando los valores del eje x por el coseno de la elevación de la antena para la cuál se hizo la medición.
ajustaEscala()	Ajusta la escala de la gráfica para que se muestre es cuadro comprendido entre (**mixmin,miymin,mixmas,mimas).**

Tabla B-4 Funciones de la clase AP_Plot.

La función getAreas utiliza la extensión de la formula trapezoidal tomada de “The Art Of Scientific Computing”[Cambridge, 1992].

En la figura B-5 se muestran las variables y funciones de la clase AP_Plot que conforman la GUI estas fueron generadas automáticamente por el programa NETBeans.

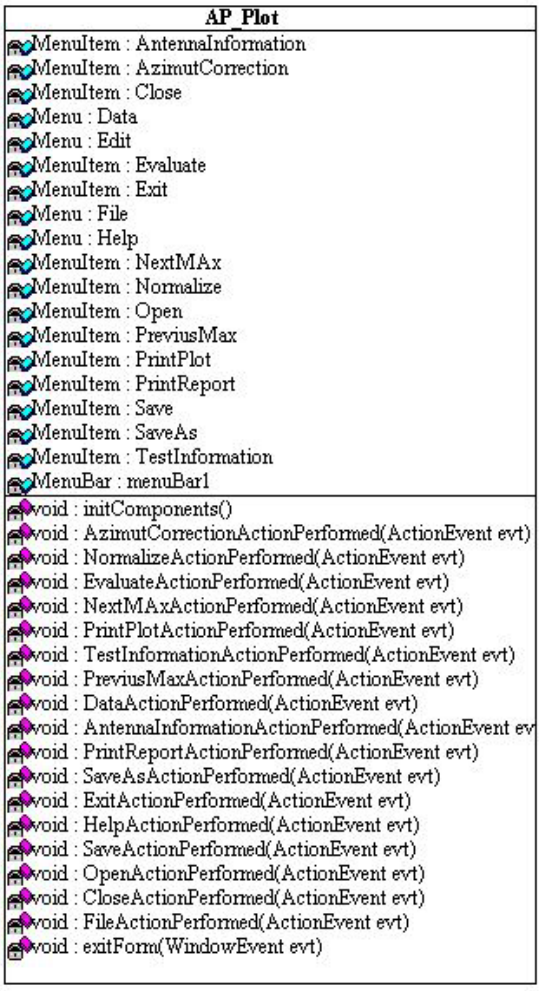


Figura B-5 Variables y funciones miembros para la GUI de la clase AP_Plot.

Las variables se refieren a los componentes de la GUI y las funciones son acciones que se ejecutan cuando algún menú u objeto de la GUI es seleccionado. En la figura B-6 se muestra el diseño de la GUI:

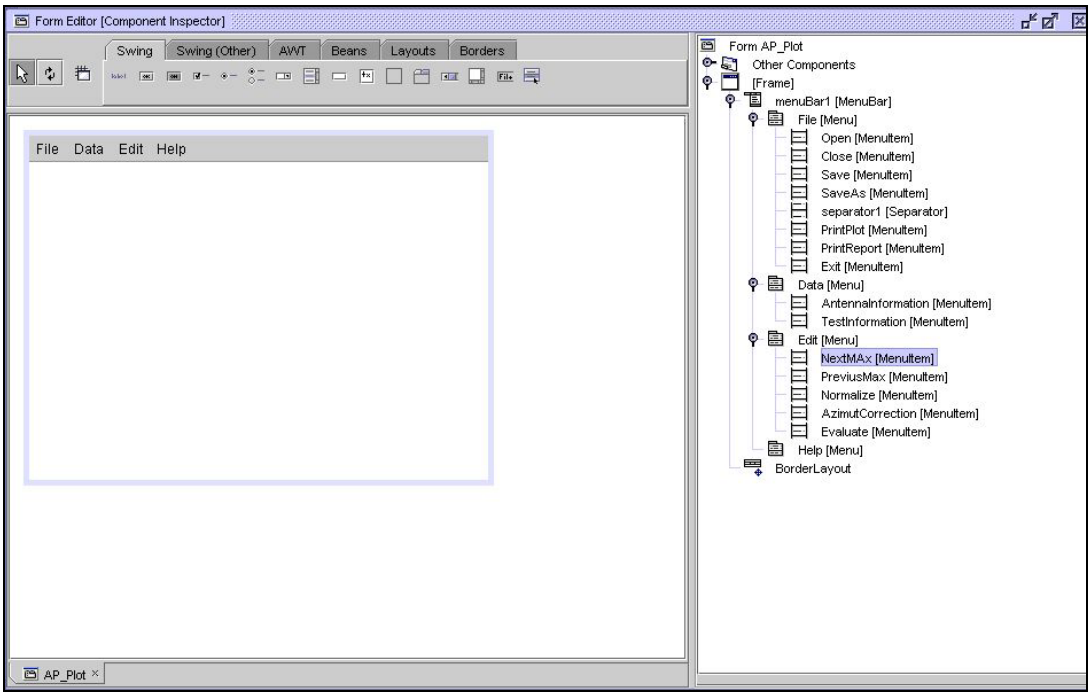


Figura B-6 GUI de la clase AP_Plot

Detalle de la clase AP_Snap

En la figura B-7 se muestra el diagrama de la clase AP_Snap en la que se muestran sus atributos y sus funciones, para simplificar el diagrama solo se muestran las funciones y clases que no tiene que ver con el manejo de ventanas:

Figura B-7 Variables y funciones miembros de la clase AP_Snap.

En seguida son descritas las variables miembro de la clase:

Variable	**Descripción**
Graph	Es el objeto que sirve para manipular la gráfica del trazo del analizador es tomado del paquete “graph”.
APatern	Contiene los puntos del trazo.
xayix, yaxis	Definen los ejes de la gráfica.

Changed	Lógico usado para saber si hay algún cambio en la gráfica.
Name	Nombre del archivo a abrir o salvar.
Elementos	Lógico para saber si hay puntos en la gráfica.
TAP	Contiene una copia del trazo del analizador cuando éste se esta manipulando.
Lmax	El último máximo encontrado.
Yamax	Lógico que dice si ya fue encontrado un máximo.
Firstmax	Lógico que dice si es el primer máximo.
Inc	Incremento en los valores de x.
Mixmax	X máximo que se dibujará en la gráfica.
Mixmin	X mínimo que se dibujará en la gráfica.
Miymax	Y máximo que se dibujará en la gráfica.
Miymin	Y mínimo que se dibujará en la gráfica.

Tabla B-5 Variables miembro de la clase AP_Snap.

En seguida se describen las funciones de la clase:

Función	**Descripción**
AP_Snap(String tipo)	Constructor. El argumento *tipo* es el título de la ventana.
exitf()	Función para salir de la ventana.
addPuntos(double datos[], int np)	Adiciona un conjunto de puntos a la gráfica, *datos* es un arreglo que contiene los pares (x,y) de puntos y *np* es el número de puntos.

addPunto(double x, double y)	Adiciona un punto a la gráfica.
saveAP(String Nombre)	Salva la gráfica con el nombre contenido en **Nombre**.
loadAP(String Nombre)	Recupera una gráfica con el nombre contenido en **Nombre**.
checkChange()	Verifica si la gráfica ha sufrido cambios y pregunta si se desea salvarla antes de realizar alguna manipulación.
main(String args[])	Es la rutina de entrada, cuando **AP_Snap** es llamada sola.
printPlot (PrintJob pj)	Imprime la gráfica actual.
centerNext()	Centra la gráfica al siguiente máximo.
centerPrevius()	Centra la gráfica al máximo anterior.
Normalize()	Hace que el máximo coincida con el 0 del eje y.
ajustaEscala()	Ajusta la escala de la gráfica para que se muestre es cuadro comprendido entre (**mixmin,miymin,mixmas,mimas).**

Tabla B-6 Funciones de la clase AP_Snap.

En la figura B-8 se muestran las variables y funciones de la clase AP_Snap que conforman la GUI estas fueron generadas automáticamente por el programa NETBeans.

Las variables se refieren a los componentes de la GUI y las funciones son acciones que se ejecutan cuando algún menú u objeto de la GUI es seleccionado. En la figura B-9 se muestra el diseño de la GUI.

Figura B-8 Variables y funciones miembros para la GUI de la clase AP_Snap.

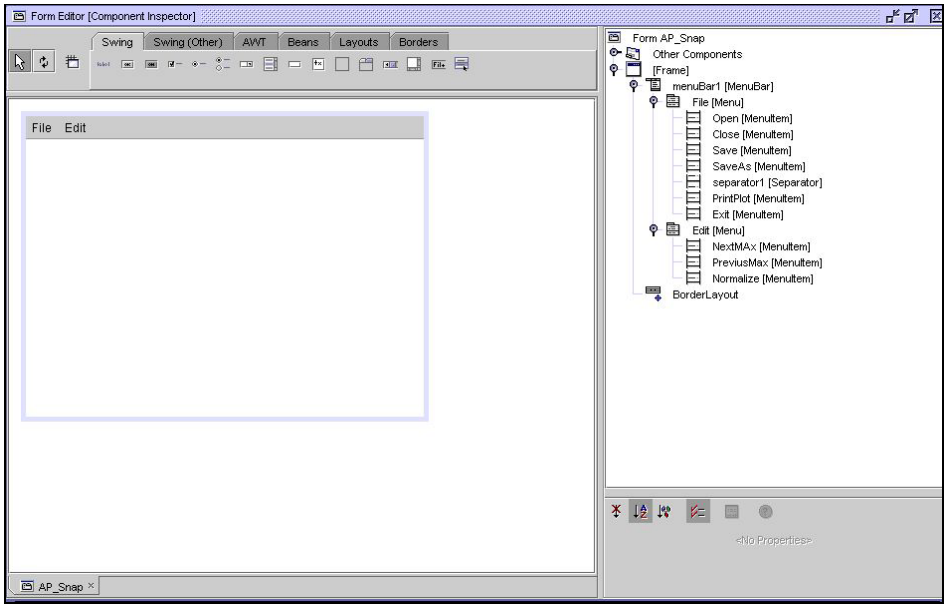


Figura B-9 GUI de la clase AP_Snap

Detalle de la clase AInf

En la figura B-10 se muestra el diagrama de la clase AP_AInf en la que se muestran sus atributos y sus funciones. Esta clase es un formulario que recolecta datos de la antena y los informa a la clase AP_Plot de la que es parte. Solo se muestran las funciones y clases que no tiene que ver con el manejo de ventanas:

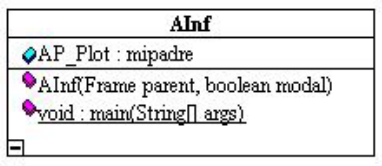


Figura B-10 Variables y funciones miembros de la clase AP_GUI.

En seguida son descritas las variables miembro de la clase:

Variable	**Descripción**
mipadre	Un objeto de tipo AP_Plot por medio de el se hace un “callback” a la clase AP_Plot, que lo instanció, para informarle de los datos de este formulario.

Tabla B-7 Variables miembro de la clase AInf .

En seguida se describen las funciones de la clase:

Función	**Descripción**
AInf(Frame parent, bolean modal)	Constructor. El argumento **parent** es la clase que lo instancia y **modal** indica si es un dialogo “modal” o no.
Main(String[] args)	Función de entrada cuando se llama a **Ainf** sola.

Tabla B-8 Funciones de la clase AInf.

En la figura B-11 se muestran las variables y funciones de la clase **AInf** que conforman la GUI estas fueron generadas automáticamente por el programa NETBeans.

Figura B-11 Variables y funciones miembros para la GUI de la clase AInf.

Las variables se refieren a los componentes de la GUI y las funciones son acciones que se ejecutan cuando se presiona algún botón. Si se presiona el botón OK se hace un “CALLBACK” a mipadre para informarle de los datos de la antena.

En la figura B-12 muestra el diseño de la GUI:

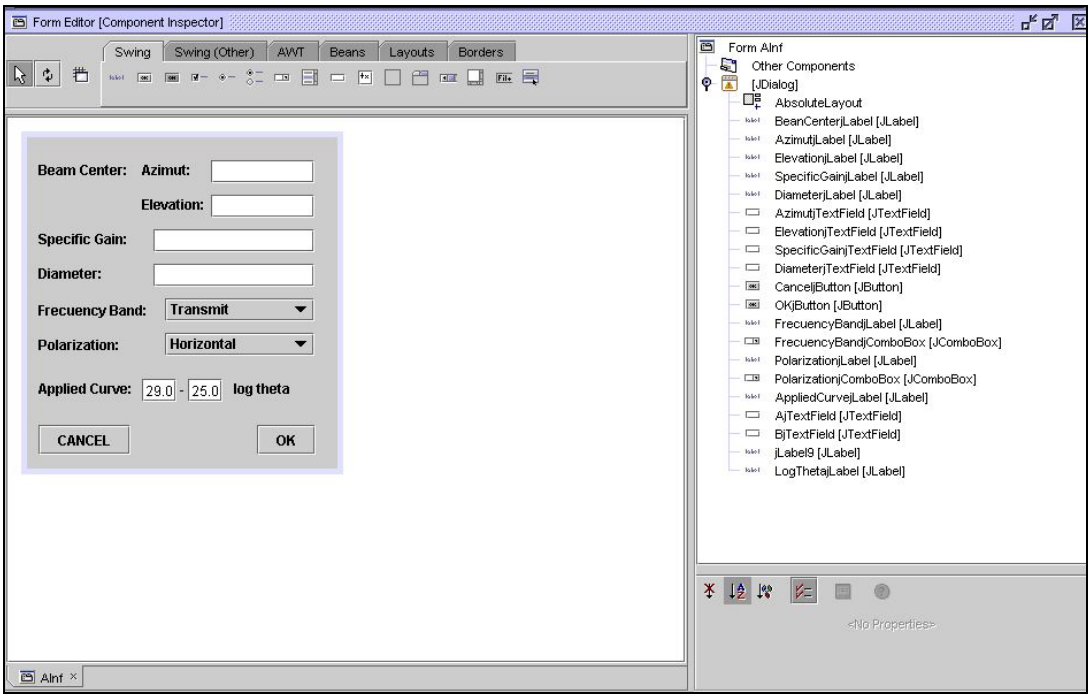


Figura B-12 GUI de la clase AInf.

Detalle de la clase TInf

En la figura B-13 se muestra el diagrama de la clase TInf en la que se muestran sus atributos y sus funciones Esta clase es un formulario que recolecta datos de la medición los informa a la clase AP_Plot de la que es parte. Solo se muestran las funciones y clases que no tiene que ver con el manejo de ventanas:

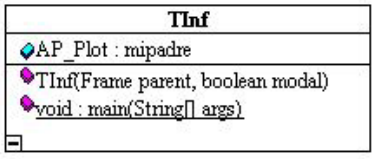


Figura B-13 Variables y funciones miembros de la clase TInf .

En seguida son descritas las variables miembro de la clase:

Variable	Descripción
mipadre	Un objeto de tipo AP_Plot por medio de el se hace un "CALLBACK" a la clase AP_Plot, que lo instanció, para informarle de los datos de este formulario.

Tabla B-9 Variables miembro de la clase TInf.

En seguida se describen las funciones de la clase:

Función	Descripción
TInf(Frame parent, bolean modal)	Constructor. El argumento **parent** es la clase que lo instancia y **modal** indica si es un dialogo "modal" o no.
Main(String[] args)	Función de entrada cuando se llama a **TInf** sola.

Tabla B-10 Funciones de la clase TInf.

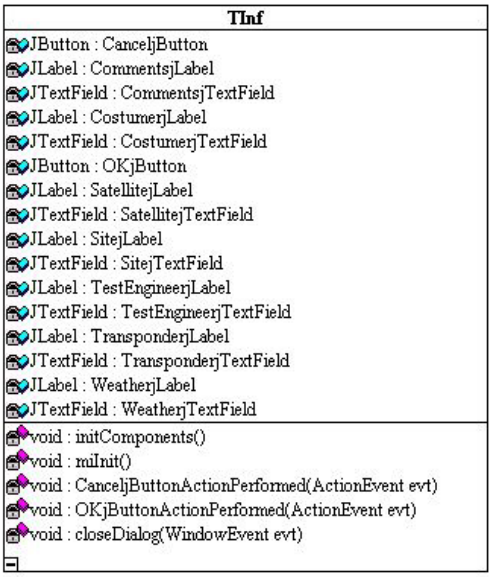


Figura B-14 Variables y funciones miembros para la GUI de la clase TInf.

En la figura B-14 se muestran las variables y funciones de la clase **TInf** que conforman la GUI estas fueron generadas automáticamente por el programa NETBeans.

Las variables se refieren a los componentes de la GUI y las funciones son acciones que se ejecutan cuando algún botón de la GUI es presionado. Si se presiona el botón OK se hace un "CALLBACK" a **mipadre** para informarle de los datos de la antena.

En la figura B-15 se muestra el diseño de la GUI:

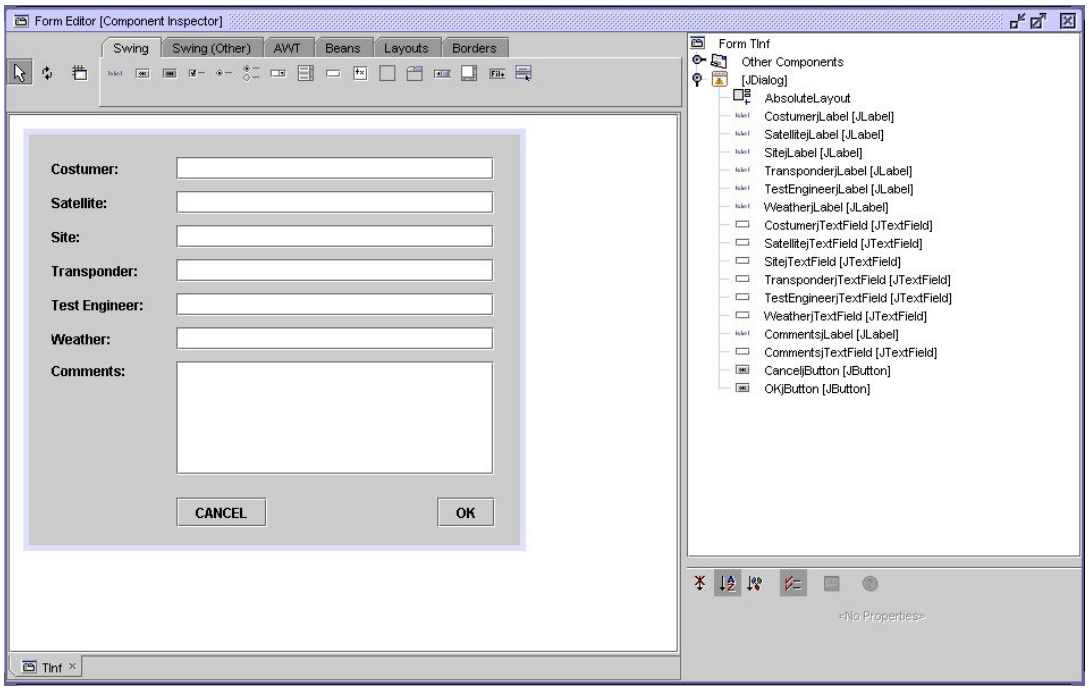


Figura B-15 GUI de la clase TInf.

Detalle de la clase Report

En la figura B-16 se muestra el diagrama de la clase Report en la que se muestran sus funciones, no se muestran las funciones y atributos que conforman la GUI:

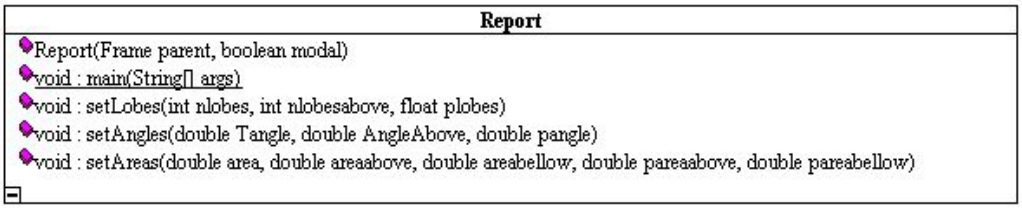


Figura B-16 Funciones miembros de la clase Report.

En seguida se describen las funciones de la clase:

Función	**Descripción**
Report(Frame parent, bolean modal)	Constructor. El argumento **parent** es la clase que lo instancia y **modal** indica si es un dialogo “modal” o no.
Main(String[] args)	Función de entrada cuando se llama a **Report** sola.
setLobes(int nlobes, int nlobesabove, float plobes)	Función que coloca los valores del número de lóbulos, lóbulos por arriba de la curva de evaluación y el porcentaje de lóbulos por arriba de la evaluación en el formulario de salida.
setAngles(double Tangle, double AngleAbove, double pangle)	Función que coloca los valores del ángulo de evaluación, la suma de los ángulos de los intervalos que el patrón de radiación esta por arriba de la curva de evaluación y el porcentaje que representa esta suma en el formulario de salida.
setAreas(double area, double areaabove, double areabellow, double pareaabove, double pareabellow)	Función que coloca los valores correspondientes a el área total, el área por arriba de la curva de evaluación, el área por debajo y los porcentajes de estas áreas.

Tabla B-11 Funciones de la clase Report.

En la figura B-17 se muestran las variables y funciones de la clase **Report** que conforman la GUI estas fueron generadas automáticamente por el programa NETBeans.

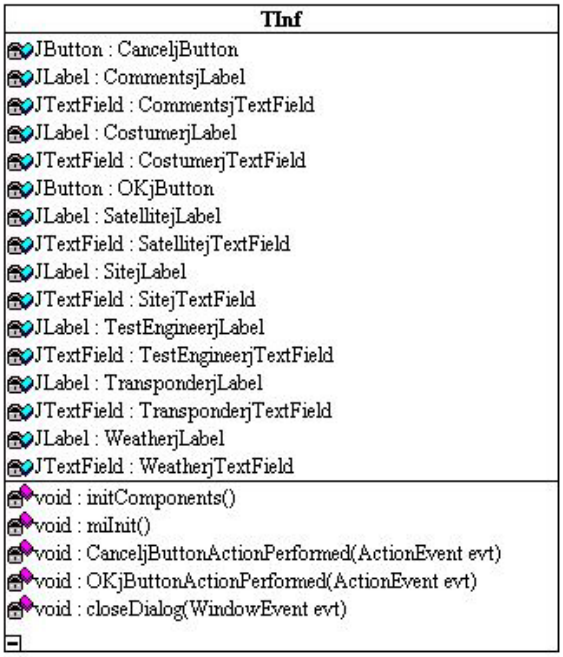


Figura B-17 Variables y funciones miembros para la GUI de la clase Report.

Las variables se refieren a los componentes de la GUI y las funciones son acciones que se ejecutan cuando algún botón de la GUI es presionado. En la figura B-18 se muestra el diseño de la GUI:

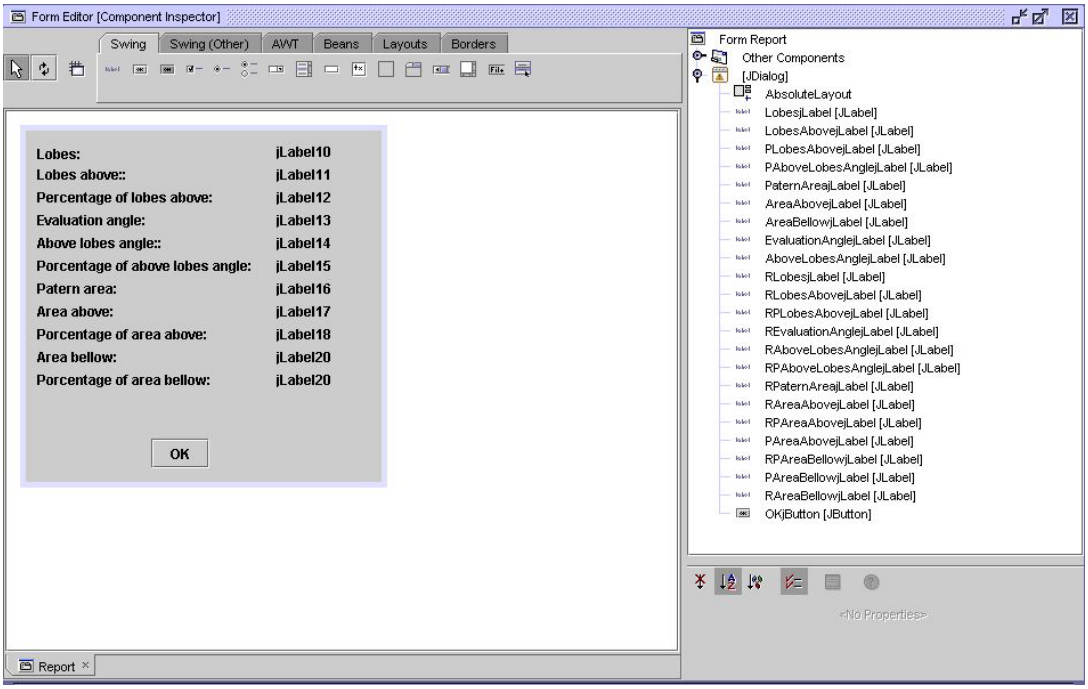


Figura B-18 GUI de la clase Report.

Detalle de la clase coClient

En figura B-19 se muestra el diagrama de la clase **coClient** en la que se muestran sus atributos y sus funciones:

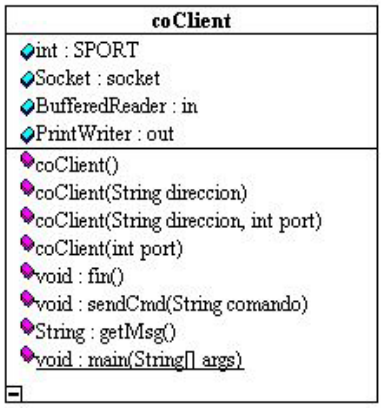


Figura B-19 Variables y funciones miembros de la clase coClient.

En seguida son descritas las variables miembro de la clase:

Variable	**Descripción**
SPORT	Número de puerto al que se conectara. El puerto por omisión es el1717.
socket	Objeto tipo socket para establecer la conexión.
in	Flujo de entrada de la conexión.
out	Flujo de salida de la conexión.

Tabla B-12 Variables miembro de la clase coClient.

En seguida se describen las funciones de la clase:

Función	**Descripción**
coClient()	Constructor. El host es “localhost” y el puerto es 1717.
coClient(String direccion)	Constructor. El host es **direccion** y el puerto es 1717.
coClient(String direccion, int port)	Constructor. El host es **direccion** y el puerto es **port**.

coClient(int port)	Constructor. El host es “localhost” y el puerto es **port**.
fin()	Función que cierra la conexión.
sendCmd(String comando)	Función que envía el mensaje **comando** al host.
getMsg()	Función que lee un mensaje del host.
main(String[] args)	Función de entrada cuando se llama soloa a coClient.

Tabla B-13 Funciones de la clase coClient.

Detalle de la clase AP_RTPHandle

En la figura B-20 se muestra el diagrama de la clase **AP_RTPHandle** en la que se muestran sus atributos y sus funciones:

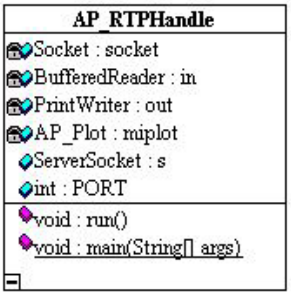


Figura B-20 Variables y funciones miembros de la clase AP_RTPHandle.

En seguida son descritas las variables miembro de la clase:

Variable	**Descripción**
socket	Objeto tipo socket para establecer la conexión.
in	Flujo de entrada de la conexión.
out	Flujo de salida de la conexión.

miplot	Objeto tipo AP_Plot para mostrar los datos muestreados en tiempo real.
s	Objeto tipo ServerSocket para esperar las peticiones del cliente.
PORT	Puerto TCP. 1718

Tabla B-14 Variables miembro de la clase AP_RTPHandle.

En seguida se describen las funciones de la clase:

Función	**Descripción**
run()	Ejecución del Thread.
main(String[] args)	Función de entrada al proceso.

Tabla B-15 Funciones de la clase AP_RTPHandle.

Detalle de la clase AP_Server

En la figura B-21 se muestra el diagrama de la clase **AP_Server** en la que se muestran sus atributos y sus funciones:

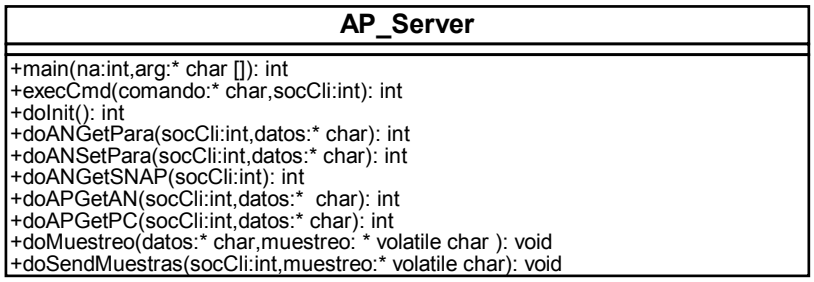


Figura B-21 Variables y funciones miembros de la clase AP_Server.

En seguida se describen las funciones de la clase:

Función	**Descripción**
main(int na,char *arg[])	Función principal en la que se espera una conexión una vez establecida se entra a un ciclo en el que se recibe un comando y se llama a la función **execCmd()** para procesarlo.
execCmd(char *comando,int socCli)	Función que interpreta el comando y llama a alguna de las siguientes funciones para ejecutarlo: **doInit(),doANGetPara(), doANSetPara(), doANGetSNAP()**,**doAPGetAN(), doAPGetPC().**
doInit()	Procesa el comando **INIT** . Manda un **reset** al analizador de espectros.
doANGetPara(int socCli, char *par, char *unidades)	Procesa un comando del tipo "**AN:GET:*par***". Obtiene del analizador de espectros el valor del parámetro en ***par***.
doANSetPara(int socCli,char *datos,char *par,char *unidades)	Procesa un comando del tipo "**AN:SET:*par:valor:unidades***". Ajusta en el analizador de espectros el parámetro en ***par*** al valor de ***valor.***
doANGetSnap(int socCli)	Procesa un comando del tipo "**AN:GET:SNAP**". Consigue un trazo del analizador de espectros.
doAPGetAN(int socCli, char *datos)	Procesa un comando del tipo "**AP:GET:AN**". Consigue un patrón de radiación muestreado por el analizador de espectros.
doAPGetPC(int socCli,char *datos)	Procesa un comando del tipo "**AP:GET:PC**". Consigue un patrón de radiación muestreado por el servidor. Se utiliza la función **doMuestreo()** para muestrear los valores del analizador de espectros y la función **doSendMuestras()**

	para enviarlos al cliente.
doMuestreo(char *datos, volatile char *muestreo)	Coloca en la tubería **/dev/rtf0** la duración del muestreo para indicarle al modulo AP_Muestreo que inicie el muestreo.
doSendMuestras(int socCli, volatile char *muestreo)	Monitorea la memoria compartida "**muestras**" y cuando hay datos nuevos los envía al cliente. Cuando enla memoría compartida contiene la cadena "FIN" inicializa la memoria compartida y termina.

Tabla B-16 Funciones de la clase AP_Server.

Detalle de la clase AP_Muestreo

En la figura B-22 se muestra el diagrama de la clase **AP_Muestreo** en la que se muestran sus atributos y sus funciones:

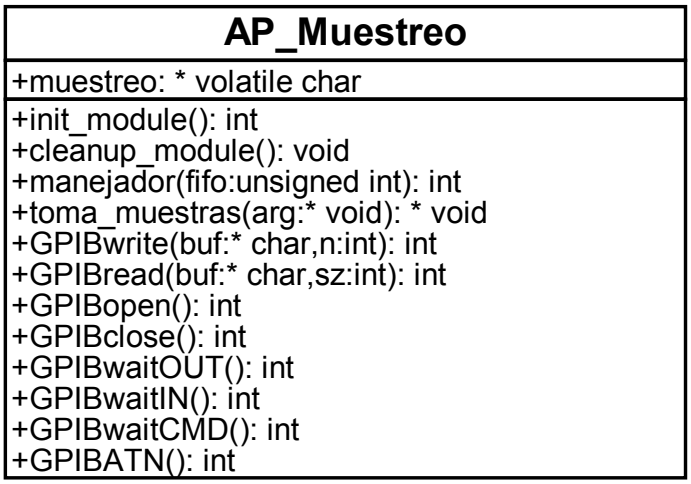


Figura B-22 Variables y funciones miembros de la clase AP_Muestreo.

En seguida son descritas las variables miembro de la clase:

Variable	Descripción
muestreo	Es la memoria compartida donde se colocarán los datos muestreados.

Tabla B-17 Variables de la clase AP_Muestreo.

En seguida se describen las funciones de la clase:

Función	Descripción
init_module()	Función para inicializar el móndulo en ella se asigna la función **manajador** para que sea llamada cuando exista algún dato en la tubería **/dev/rtf0**. Adfemás de crea ésta tubería y la memoria compartida **muestras**.
cleanup_module()	Es llamada cuando el módulo se elimina de la memoria. Libera la memoria compartida y la tubería.
manejador(unsigned int fifo)	Esta rutina es llamada cuando **AP_Server** coloca el tiempo de muestreo en la tubería sin nombre. Inicia el “thread” **toma_muestras()** y le pasa como argumento el tiempo de muestreo.
toma_muestras(void *arg)	Inicia la conexión GPIB con **GPIBopen()**. Después configura el analizador de espectros, a que el marcador se posicione en el máximo luego cambia a “spawn” 0 y un retrazo de cada 20 milisegundos, enviándole el comando "MKPK HI;SP0MZ;ST20MS;" por medio de la función **GPIBwrite()**. En seguida empieza el ciclo de muestreo por el tiempo recibido como argumento. Toma una muestra cada 60 milisegundos pidiéndole al analizador de espectros un valor con el comando "TS;MKA?;” utilizando

	GPIBwrite() y **GPIBread()** para leer el valor. Al final cierra la conexión GPIB con **GPIBclose()**.
GPIBopen()	Inicializa el bus llamando a **GPIBATN()** y colocando la línea Remote ENable.
GPIBclose()	Deshabilita la línea Remote ENable.
GPIBread(char *buf, int sz)	Envía ***sz*** bytes de ***buf*** al analizador de espectros. Primero configura el analizador de espectros como oyente y el Servidor como hablante mediante los siguientes bytes de comando {0x5f,0x3f,20,53}.
GPIBwrite(char *buf, int n)	Lee la respuesta del analizador de espectros y la coloca en ***buf*** en ***n*** se coloca cuantos bytes fueron leídos. Primero configura el analizador de espectros como hablante y al Servidor como oyente mediante los siguientes bytes de comando {0x5f,0x3f,32,40}.
GPIBwaitOUT()	Espera que se pueda enviar otro dato al bus de datos o timeout.
GPIBwaitIN()	Espera que se pueda leer otro dato del bus, ocurra un timeout o sea el último dato.
GPIBwaitCMD()	Espera que se pueda enviar el siguiente byte de comando por el bus.
GPIBsetATN()	Informa que el servidor esta en control del bus GPIB.

Tabla B-18 Funciones de la clase AP_Muestreo.