_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE UPV INGENIEROS DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DISEÑO Y AUTOMATIZACIÓN DE UN SISTEMA DE MEDIDA PARA LA CARACTERIZACIÓN DEL CANAL RADIO EN INTERIORES

Santiago Martínez Toval

Tutor: Lorenzo Rubio Arjona

Cotutor: Vicent Miquel Rodrigo Peñarrocha

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2015-16

Valencia, 29 de junio de 2016

Resumen

En este Trabajo Final de Grado (TFG) se ha diseñado e implementado una sonda de canal basada en un analizador de redes vectorial (ARV) y un sistema de posicionamiento XY. Se ha empleado la herramienta GUIDE de Matlab para crear un programa a partir del cual de manera remota y automatizada se puedan configurar todos los parámetros del ARV y controlar el movimiento del posicionador.

La sonda implementada permite caracterizar el comportamiento selectivo en frecuencia del canal a partir de la medida del parámetro de dispersión S_{21}. Se ha añadido también la posibilidad de medir el resto de parámetros de dispersión, así como los parámetros *b* y *B*. En cuanto al posicionador se ha implementado tanto la posibilidad de realizar medidas por mallado rectangular como por barrido circular, para la estimación de algoritmos en distintos ángulos de llegada de las contribuciones.

Resum

En aquest Treball Final de Grau (TFG) s'ha dissenyat i implementat una sonda de canal basada en un analitzador de xarxes vectorial (ARV) i un sistema de posicionament XY. S'ha emprat l'eina GUIDE de Matlab per a crear un programa a partir del com de manera remota i automatitzada es puguen configurar tots els paràmetres del ARV i controlar el moviment del posicionador.

La sonda implementada permet caracteritzar el comportament selectiu en freqüència del canal a partir de la mesura del paràmetre de dispersió S21. S'ha afegit també la possibilitat de mesurar la resta de paràmetres de dispersió, així com els paràmetres b i B. Quant al posicionador s'ha implementat tant la possibilitat de realitzar mesures per mallado rectangular com per escombratge circular, per a l'estimació d'algorismes en diferents angles d'arribada de les contribucions.

Abstract

In this final thesis, it has been designed and implemented a channel sounder based on a Vector Network Analyzer (VNA) and a XY positioning system. It has been used GUIDE tool of Matlab to create a program, from which in a remote and automated way could be configured all VNA parameters as well as the positioner movement.

The sounder implemented allows to characterize the frequency selective behavior channel from the measurement of the dispersion parameter S_{21}. We have also added the chance to measure the rest of dispersion parameters, and other parameters as *b and B*. As to the positioner, it has been implemented both the possibility of measuring by meshing rectangular such as circular sweep, for algorithms estimating at different angles arrival of the contributions.

Índice

Capítulo 1. Introducción, objetivos y estructura de la memoria

1.1 Introducción

El campo de las comunicaciones inalámbricas estudia un amplio rango de posibilidades y recursos de las ondas electromagnéticas. Es importante conocer las características del canal radio en el que nos vamos a desenvolver en cada caso, ya que como es sabido en las comunicaciones inalámbricas siempre vamos a encontrar un canal muy hostil en términos de dispersión temporal al efecto multicamino, en el cual la distancia entre transmisor y receptor junto con la frecuencia de trabajo van a ser determinantes en el diseño final del sistema.

El canal radio presenta ciertas ventajas en transmisión, como puede ser el fácil despliegue o la movilidad que nos permite. Sin embargo, también presenta ciertos inconvenientes como el multicamino o el efecto doppler, lo cual obliga a realizar la caracterización del canal radio para obtener un diseño óptimo del sistema.

El analizador nos proporciona la respuesta en frecuencia mediante el *'Power Delay Profile (PDP)'* en base a las contribuciones multicamino que recibe. Es en este punto donde entra en juego toda la teoría de Fourier aplicada al estudio de la función de transferencia del canal (*H(f)*).

1.2 Objetivos

El objetivo del TFG es el diseño, implementación y automatización de una sonda de canal. La sonda permite medir el comportamiento selectivo en frecuencia del canal a través de la función de transferencia cronovariable.

La sonda está formada por un ARV, un posicionador XY, antenas y cables. El proceso de automatización se va a realizar desde un PC empleando la herramienta GUIDE de Matlab, desde la cual crearemos un programa que nos permita controlar tanto el ARV como el posicionador.

Aunque en primer lugar se pretende medir el parámetro S_{21}, la sonda implementada permite pedir cualquier otro parámetro. En este sentido, el programa implementado permite el control automático del ARV, ya que ofrece la posibilidad de medir los parámetros b1 y b2, como ejemplo, haciendo que el equipo funcione en modo analizador de espectros.

A continuación, se presentan los componentes empleados en nuestra sonda de medida.

Analizador de redes (Keysight N5227A)

Un Analizador de Redes es un instrumento capaz de analizar las propiedades delas redes eléctricas, especialmente aquellas propiedades asociadas con la reflexión y la transmisión de señales eléctricas, conocidas como parámetros de dispersión. Los analizadores de redes son frecuentemente usados en altas frecuencias, que operan hasta 110 GHz. Disponen de un número determinado de puertos (mínimo 2) bidireccionales, permitiendo así la medida de los diferentes parámetros de dispersión.

Existen dos tipos de Analizadores de redes:

- SNA (Scalar Network Analyzer): Analizador de redes escalar, mide propiedades de amplitud solamente
- VNA (Vector Network Analyzer): Analizador de redes vectoriales, mide propiedades de amplitud y fase

El caso del analizador escalar es similar al de un analizador de espectros combinado con un generador de barrido. Por el contrario un analizador vectorial nos permite medir también la fase, y es el tipo de analizador de redes en el que se centran en desarrollar los fabricantes, por lo que será el que emplearemos en el trabajo.

El analizador de redes que vamos a emplear es el modelo de N5227A de *Keysight Technologies*. Este dispone de 2 puertos (existe el modelo de 4 puertos) y nos permite trabajar hasta la frecuencia de 70GHz, por lo que cubre perfectamente la banda de frecuencias sobre la que tenemos especial interés. En el capítulo de teoría trataremos la componente teórica involucrada en el analizador y en el análisis de los parámetros S.

En el caso del modelo de analizador que empleamos además de permitirnos medir los parámetros de dispersión, también podemos medir otros parámetros: A1, A2, B1, B2, a1, a2, b1, b2, R1, R2…

En la Figura 1 se puede apreciar una foto con el detalle del panel frontal del ARV con el vamos a trabajar.

Figura 1: Analizador de redes N5227A

Sistema de posicionado XY

Para la caracterización del canal, y en concreto la medida del PDP, es necesario realizar medidas en una zona pequeña, lo que se denomina small-local área, de modo que se realice un promediado de los PDPs medidos. Lo habitual es realizar un muestreo espacial en forma de rejilla, aunque en ocasiones interesa un muestreo circular. A partir de N mediciones del PDP, el PDP medio se estima de la siguiente forma:

$$PDP(\tau) = \frac{1}{N}\sum_{i=1}^{N} PDP_i(\tau) \quad (1)$$

Para esto emplearemos una mesa que consta de una plataforma móvil controlada por dos motores, uno desplaza en el eje X y el otro en el eje Y. El usuario configurará la forma y orden en que se moverá la antena y el espacio que cubrirá.

En la Figura 2 se muestra un detalle del panel frontal y trasero del controlador de Arrick Robotics. En la Figura 3 se muestra una foto de la mesa posicionadora XY.

Figura 2. Panel frontal y trasero del sistema MD-2

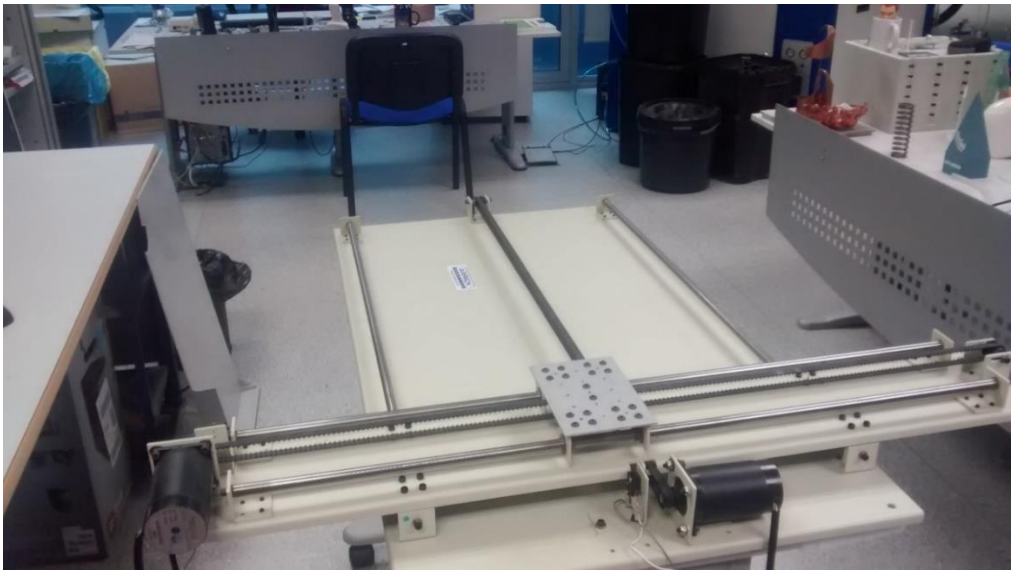


Figura 3. Mesa posicionadora XY

Antenas

El sistema está concebido para operar en la banda de 60 GHz, utilizando antenas omnidireccionales. No obstante, la sonda de canal implementada permite medir en cualquier rango de frecuencias por encima de 10MHz y por debajo de 70 GHz, que es hasta donde el analizador es capaz de llegar

Interfaz controlador (PC)

Finalmente la tercera parte fundamental del trabajo es un ordenador desde el que controlemos la automatización de las medidas mediante el software *MatLab*. Desde el pc se controlará tanto el analizador como la mesa de posicionado con el uso de una interfaz gráfica que diseñaremos con la herramienta de diseño gráfico de usuarío *GUIDE* contenida dentro de *MatLab*.

MatLab es una herramienta de software matemático que ofrece un entorno de desarrollo integrado con un lenguaje propio de programación, que comparte similitudes con otros lenguajes como C o JAVA. Permite la creación de interfaces de usuario y la comunicación con programas que trabajan con otros lenguajes así como con otros dispositivos hardware.

Por una parte desarrollaremos una interfaz encargada de controlar todo lo concerniente al analizador y de forma totalmente independiente otra para manejar el posicionador. Posteriormente se unirán ambas opciones a fin de disponer de un sistema de medidas completo.

También se hará uso de otras herramientas propias de Matlab como Midedit (Matlab Instrument Driver Editor) para crear y editar drivers, o Tmtool (Test & Measurement Tool) que nos permitirá trabajar con los drivers y crear objetos para interactuar con el analizador.

En la Figura 4 se muestra un detalle de cómo se han conectado las distintas partes de la sonda.

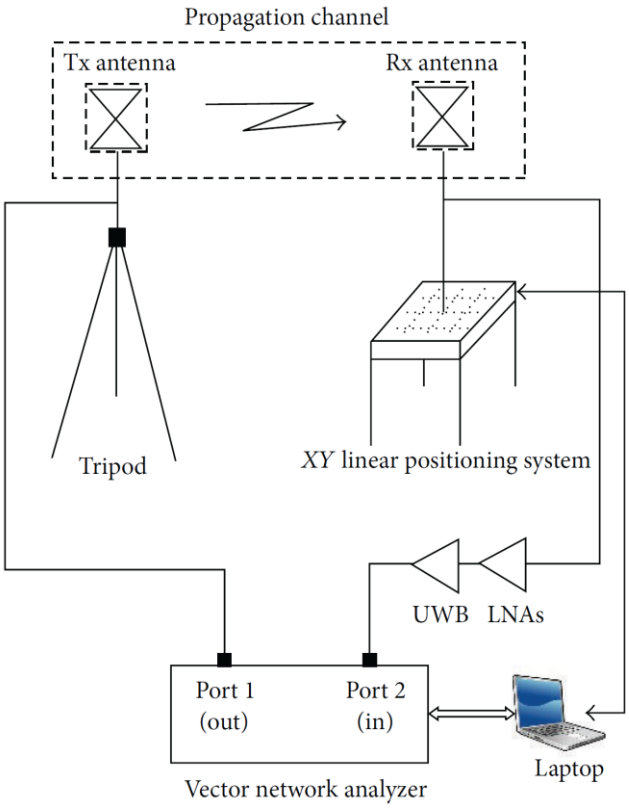


Figura 4. Sonda de medida [1]

1.3 Estructura de la memoria

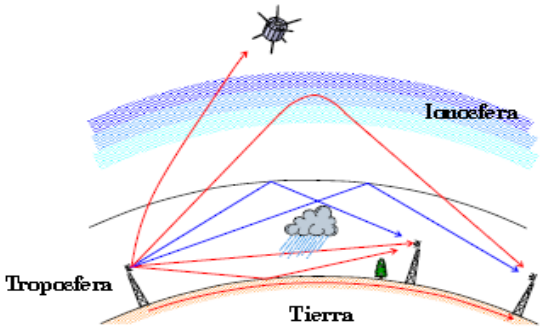
La memoria se encuentra estructurada en varios capítulos bien diferenciados. En el Capítulo 2 se describen los principales aspectos teóricos relacionados con el canal radio. Seguidamente en el Capítulo 3 se habla de la metodología seguida para el desarrollo del TFG. Posteriormente, en el Capítulo 4 se presenta y detalla el programa realizado con la herramienta GUIDE de Matlab. En éste capítulo se desarrolla el grueso del trabajo realizado. Finalmente, en el Capítulo 5 se presentan las conclusiones y también se indican ciertas líneas de trabajo futuras.

Capítulo 2. Aspectos teóricos

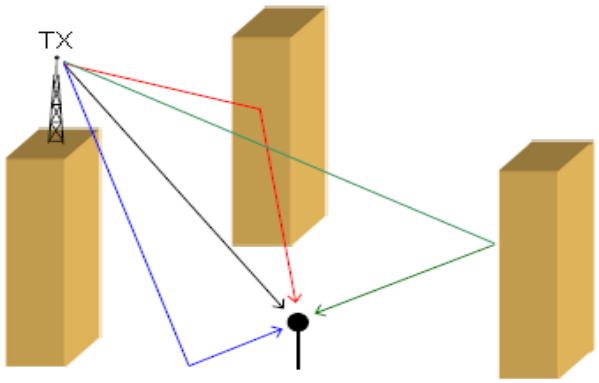
2.1 Canal radio

El canal radio suele ser siempre un medio hostil en el cual propagar una onda electromagnética puede llegar a complicarse bastante. Cuando queremos establecer una comunicación entre transmisor y receptor son innumerables la cantidad de objetos interferentes que hay por medio (personas, coches, edificios, árboles, montañas, etc.). Es por esto que a la antena receptora la señal que le llega proviene de múltiples contribuciones, ya que por un lado le podrá llegar una componente directa de la señal que se estaba transmitiendo, pero a su vez estarán llegando replicas procedentes de reflexiones, difracciones, scattering… y con distintos niveles de potencia, es lo que se llama efecto multicamino

En la Figura 5 se muestran los tipos de propagación y el efecto multicamino.



(a)



(b)

Figura 5: (a)Principales mecanismos de propagación y (b) efecto multicamino [2]

En nuestro caso vamos a trabajar en un entorno de interior, con lo cual evitamos el efecto de gran parte de los objetos interferentes que comentábamos con anterioridad, sin embargo, hay otros como sillas, mesas o paredes, que provocan distintas contribuciones en el receptor, contribuyendo a un efecto de dispersión temporal (selectividad en frecuencia).

Del mismo modo, si la antena transmisora, la receptora o los objetos interferentes se desplazan, la señal sufrirá dispersión en frecuencia (selectividad en el tiempo).

Con el fin de intentar corregir todos estos procesos interferentes en la propagación existe el modelado/caracterización del canal. Dado que los métodos de propagación son totalmente arbitrarios no es fácil realizar dicho modelado. Se emplean aproximaciones y simplificaciones a la hora de aplicarlo a casos prácticos. Por tanto, cuando se realiza el modelado del canal radio tan solo se trata de una aproximación en la que se han tenido en cuenta los aspectos del canal que afectan en mayor medida a la propagación de la señal.

En recepción el parámetro más importante es la intensidad de campo o potencia recibida en función del servicio que estemos prestando.

Generalmente se manejan varios conceptos en lo referente a la recepción en el canal radio:

1. Intensidad de campo mínima utilizable: valor mínimo que permite obtener cierta calidad en recepción. Está condicionado a la sensibilidad del receptor, al rendimiento de la antena y al ruido.
2. Intensidad de campo utilizable: también se tienen en cuenta los efectos interferentes de otros transmisores.
3. Relación de protección en RF: valor mínimo de la relación entre la señal deseada y la no deseada a la entrada del receptor.

En la Figura 6 se muestra un detalle de los distintos canales de transmisión que existen.

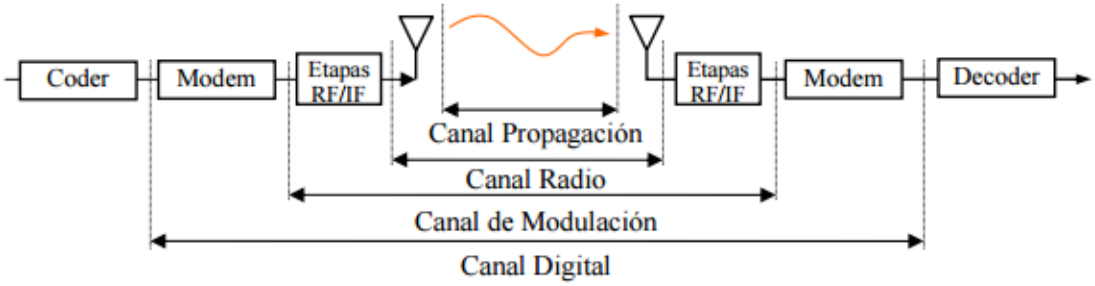


Figura 6. Diferentes definiciones del canal de transmisión [3]

Antenas:

La antena es un elemento que permite radiar, de forma eficiente, una energía en forma de onda electromagnética. Las propiedades de una antena en transmisión son las mismas que las de una antena en recepción. Por lo que, en vez de radiar, la antena puede recibir esa radiación que, una vez guiada hasta el receptor, se traducirá en energía eléctrica. Esto permite el diseño de variados sistemas de radiocomunicación, con la característica de movilidad como máximo exponente de este tipo de comunicaciones.

Las características de las antenas dependen de la relación entre sus dimensiones y la longitud de onda de la señal enviada o recibida. Si las dimensiones de la antena son mucho más pequeñas que la longitud de onda las antenas se denominan *elementales*, si tienen dimensiones del orden de media longitud de onda se llaman *resonantes*, y si su tamaño es mucho mayor que la longitud de onda son *directivas*.

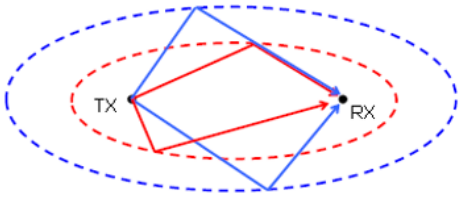
2.1.1 Dispersión Temporal

Dentro de la caracterización del canal radio, la parte que atañe a la dispersión temporal es parte crítica en la transmisión de la señal. Se produce como consecuencia del efecto multicamino y puede darse bien en el dominio del tiempo (dispersión) o bien en el dominio de la frecuencia (selectividad e interferencia entre símbolos).

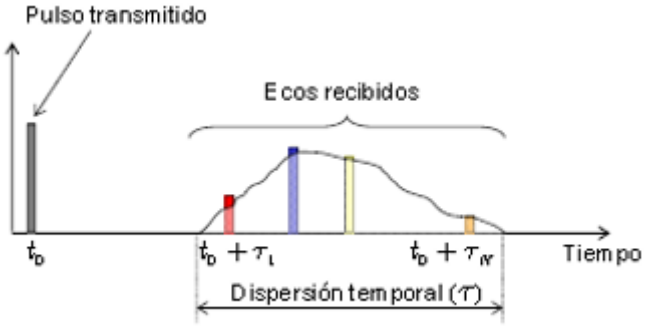
- Interferencia entre símbolos (ISI)

  Es más pronunciado cuando mayor sea el retardo entre los caminos. Como la dispersión del retardo viene dada, esto nos limita el tiempo de símbolo, es decir, la velocidad de transmisión con la que podemos operar.

  En la Figura 7 podemos ver un detalle de la Interferencia entre Símbolos.



(a)



(b)

Figura 7. (a) y (b) Interferencia entre símbolos (ISI) [2]

- Desvanecimiento selectivo en frecuencia. Posibilidad de fading plano o bien de fading selectivo.
  - Fading plano: conserva el espectro de la señal. No se pueden discriminar componentes en RX.
  - Fading selectivo: no conserva el espectro de la señal; aparece ISI. Se pueden discriminar las componentes en RX.

2.1.2 Parámetros para la caracterización del canal radio[1]

Los parámetros más importantes para definir el canal radio, pueden obtenerse de la respuesta al impulso del canal. En base a explicar el comportamiento dispersivo en tiempo y frecuencia del canal, haremos uso de la respuesta al impulso de un canal radio variable en el tiempo.

Si estamos en un entorno variable en el cual existe multicamino, se puede interpretar la señal recibida como una superposición temporal de todas las contribuciones que se van recibiendo,

[1] Ecuaciones extraídas de las fuentes [4], [5] y [6]

teniendo en cuenta en cada caso tanto la atenuación como el retraso específico para dicho camino de propagación. Si escribimos la señal pasobanda transmitida como:

$$s(t) = Re\{s_p(t)e^{j2\pi f_c t} \tag{2.1}$$

Siendo s_p(t) la envolvente compleja y f_c la frecuencia de la portadora, la señal pasobanda que se recibe será:

$$y(t) = Re\left\{\sum_{k=1}^{\infty}\sum_{i=1}^{\infty} a_{k,i}s_p(t-\tau_i)e^{j2\pi(f_c+f_d\cos(\theta_{k,i}))t+\phi_{k,i}}\right\} = \tag{2.2}$$

$$= Re\left\{\left[\sum_{i=1}^{\infty} h(t,\tau_i)\delta(t-\tau_i)\otimes s_p(t)\right]e^{j2\pi f_c t}\right\} =$$

$$= Re\{[h(t,\tau_i)\otimes s_p(t)]e^{j2\pi f_c t}\}$$

Donde

$$h(t,\tau_i) = \sum_{k=0}^{\infty} a_{k,i}\, e^{j2\pi(f_c+f_d\cos(\theta_{k,i}))t+\phi_{k,i}} \tag{2.3}$$

$$h(t,\tau) = \sum_{i=0}^{\infty} h(t,\tau_i)\,\delta(t-\tau_i) \tag{2.4}$$

Con $a_{k,i}$ y $\varphi_{k,i}$ siendo la amplitud y la fase de la i-ésima contribución que llega al receptor con un ángulo $\theta_{k,i}$ con respecto a la dirección de movimiento, y retardo τ_i. El término f_d es la máxima frecuencia Doppler, $f_d = v/\lambda_c$, donde v hace referencia a la velocidad del receptor, $\lambda_c = c_0/f_c$ es la longitud de onda asociada a la frecuencia portadora f_c, y c_0 es la velocidad de la luz. La función δ(.) es una delta de Dirac, y $\otimes$ significa la operación de convolución. La ecuación (X) corresponde a la respuesta al impulso variante en el tiempo del canal radio. Específicamente $h(t,\tau)$ es la respuesta del canal equivalente pasobajo en el instante t a un impulso generado transcurridos τ segundos. $h(t,\tau)$ es conocido como la *función delay-spread de entrada*.. El término $h(t,\tau_i)$ es el coeficiente complejo dependiente del tiempo asociado al retardo τ_i, y puede expresarse como:

$$h(t,\tau_i) = h_{iR}(t) + jh_{iQ}(t) \tag{2.5}$$

Donde

$$h_{iR}(t,\tau_i) = \sum_{k=0}^{\infty} a_{k,i}\cos(2\pi f_d cos\theta_{k,i}t + \phi_{k,i}) \tag{2.6}$$

$$h_{iQ}(t,\tau_i) = \sum_{k=0}^{\infty} a_{k,i}\,\text{sen}(2\pi f_d cos\theta_{k,i}t + \phi_{k,i}) \tag{2.7}$$

Etas son las componentes en fase y cuadratura de la señal.

En sentido práctica, cuando tenemos muchos canales físicos, estos pueden ser considerados estacionarios en cortos periodos de tiempo (o pequeños tramos espaciales) aunque en sentido estricto no lo sean, pero si en un sentido amplio (Wide Sense Stacionary Channels, WSS). De igual forma un canal puede conllevar scaterring incorrelado (Uncorrelated Scattering, US) en su variable temporal. Si se combina WSS con US aparece WSSUS, que es de uso frecuente en modelado de sistemas celulares. En este caso el canal puede representarse como una línea de fragmentos con retardo (Tapped Delay Line, TDL) que tiene una respuesta al impulso como sigue:

$$h(t,\tau) = \sum_{i=0}^{\infty} h_i(t)\,\delta(t-\tau_i) \tag{2.8}$$

Haciendo uso de esta expresión, cada uno de los fragmentos se corresponderá con un grupo de componentes multicamino separadas por muy poco en el tiempo/espacio. Esta forma de representarlo es usualmente empleada en la teoría de caracterización de canales porque la resolución temporal del receptor no es suficiente para resolver todas las componentes multicamino en gran parte de los casos prácticos. No tiene sentido que el número de fragmentos permanezca constante durante un periodo de tiempo, donde la suposición de WSS se da por válida. En la siguiente figura (Figura 8) se puede observar un modelo de TDL con elementos de retardo.

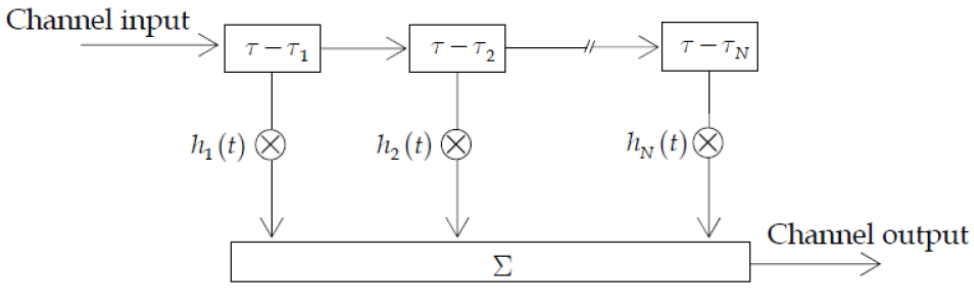


Figura 8. Modelo TDL

La variación temporal de la amplitud compleja de los fragmentos es debida a las fases relativas de las componentes multicamino que cambian con los desplazamientos, en términos de longitud de onda, del transmisor/receptor y objetos interferentes. Así, $h_i(t)$ está referida en la literatura como *el desvanecimiento de la envolvente compleja* del camino i-ésimo. La variación temporal de $h_i(t)$ es el modelo a través las distribuciones Rayleigh, Rice o Nakagami-m entre otros.

El efecto dispersivo tanto en el dominio del tiempo con en el de la frecuencia en el canal radio nos condiciona las técnicas empleadas en transmisión para mitigar los impedimentos y límites del sistema. Sirva de ejemplo en dispersión temporal tener que implementar técnicas de ecualización, o con dispersión en frecuencia usar diversidad y ecualización adaptativa.

Parámetros de dispersión en el tiempo

Sea un canal variante en el tiempo con multicamino, se pueden expresar sus características de dispersión temporal si se expresa la función de autocorrelación a la salida del canal en función de la autocorrelación de la función delay-spread de entrada, denotada como:

$$R_h(t,s,\tau,\eta) \triangleq E\{h(t,\tau)h^*(s,\eta)\} \tag{2.9}$$

Donde E es la esperanza matemática.

Si consideramos que trabajamos con WSSUS, la autocorrelación del canal de salida se puede describir por el delay cross-power spectral density. El suponer WSSUS implica caracterizar el canal a pequeña escala donde se puede observar el scattering local. De aquí se obtiene lo que se conoce como *Power Delay Profile* PDP.

En base a esto el parámetro más importante para caracterizar el comportamiento en dispersión temporal del canal es el rms (root mean square) delay spreed, que se corresponde con el segundo momento central del PDP:

$$\tau_{rms} \triangleq \sqrt{\frac{\int_0^\infty (\tau-\bar{\tau})^2 P_h(\tau)d\tau}{\int_0^\infty P_h(\tau)d\tau}} \tag{2.10}$$

Siendo $\bar{\tau}$ el primer momento central del PDP definido como:

$$\bar{\tau} \triangleq \sqrt{\frac{\int_0^\infty \tau P_h(\tau)d\tau}{\int_0^\infty P_h(\tau)d\tau}} \tag{2.11}$$

El comportamiento selectivo en frecuencia de los canales radio se describe haciendo uso de la *función de autocorrelación tiempo-frecuencia*, representada como $R_T(\Omega,\zeta)$, el cual es una transformada de Fourier de $P_h(\xi,\tau)$ sobre la variable retardo, es decir.

$$R_T(\Omega,\xi) = \int_{-\infty}^{\infty} P_h(\xi,\tau)e^{-j2\pi\Omega\tau}d\tau \tag{2.12}$$

Donde la variable Ω viene a representar un intervalo en frecuencia, de modo que cuando ξ=0, la expresión se reduce a:

$$R_T(\Omega) = \int_{-\infty}^{\infty} P_h(\tau)e^{-j2\pi\Omega\tau}d\tau \tag{2.13}$$

Donde $R_T(\Omega)$ es conocida como la *función de correlación en frecuencia*. Una referencia para medir la selectividad en frecuencia del canal es el ancho de banda de coherencia, que se denota por B_c. Físicamente representa el ancho de banda del canal en el cual el mismo experimenta un comportamiento aproximadamente plano en frecuencia. Como $P_h(\tau)$ está relacionada con $R_T(\Omega)$ por una transformada de Fourier, existe la relación inversa entre el rms delay spread y el ancho de banda de coherencia, es decir $B_c \propto 1/\tau_{rms}$

Parámetros de dispersión en frecuencia

Las características de la dispersión en frecuencia pueden ser derivadas del *Doppler-Cross-Power Spectral Density* $P_H(\Omega,\nu)$, que en un canal WSSUS se encuentra relacionado con la función *Doppler Spread* de salida $R_H(\Omega;\nu,\mu)$.

$$R_H(f, f+\Omega; \nu, \mu) = \delta(\nu - \mu) P_H(\Omega, \nu) \tag{2.14}$$

Donde la función de autocorrelación viene definida como

$$R_H(f, m; \nu, \mu) \triangleq E\{H(f, \nu) H^*(m, \mu)\} \tag{2.15}$$

Siendo $H(f,\nu)$ la *función Doppler-spread* de salida. En la ecuación f y m son variables de frecuencia, mientras que ν y μ corresponden a variables de frecuencias Doppler. De las relaciones entre las funciones de autocorrelación en un canal WSSUS, la función $P_H(\Omega,\nu)$ puede también entenderse como la transformada de Fourier de la función de scattering con recto de la variable τ. Cuando $\Omega \to 0$, $P_H(\Omega,\nu)$ se simplifica como $P_H(\nu)$, el cual adquiere el nombre en la literatura de *Doppler power density spectrum* (PDS). De forma similar al PDP, el PDS Doppler también se puede obtener a partir de la integral de la función de scattering sobre el variable retardo.

$$\nu_{rms} \triangleq \sqrt{\frac{\int_0^\infty (\nu - \bar{\nu})^2 P_H(\nu) d\nu}{\int_0^\infty P_H(\nu) d\nu}} \tag{2.16}$$

Donde $\bar{\nu}$ es la media del Doppler spread, o el primer momento central del PDS Doppler, dado por la siguiente expresión.

$$\bar{\nu} \triangleq \sqrt{\frac{\int_0^\infty \nu P_H(\nu) d\nu}{\int_0^\infty P_H(\nu) d\nu}} \tag{2.17}$$

Los desplazamientos causan interferencias destructivas de las componentes multicamino en el receptor, las cuales llegan al receptor con distintos retardos que cambian en el tiempo/espacio. Este tipo de fading se observa en escalas espaciales en términos de longitud de onda, y está reflejado en la literatura como *fading* a pequeña escala en contraposición con el *fading* a gran escala (también llamado *shadowing*) debido a la obstrucción o bloqueo de los caminos de propagación. La variación de las envolventes de señal recibida también puede modelarse de forma estadística usando comunes distribuciones Rayleigh, Rice o Nakagami-m.

De forma similar al ancho de banda de coherencia, para un canal variante en el tiempo es posible definir un parámetro llamado *tiempo de coherencia*, representado como T_c, para referirse al intervalo de tiempo en el cual el canal puede ser considerado estacionario. Hay una relación inversa entre el *rms* Doppler spread y el tiempo de coherencia, es decir, $T_c \propto 1/\tau_{rms}$.

Capítulo 3. Metodología

3.1 Descripción del entorno de trabajo

Este proyecto guarda similitudes con proyectos realizados en años anteriores, en los cuales también se habían realizado sistemas automatizados de medida en interiores para otras frecuencias y empleando otro tipo de analizador (de la familia *Rohde & Schwarz)*. Sin embargo aunque el sistema de posicionado es el mismo y su implementación es bastante similar a las ya realizadas previamente, la parte de programación del control del analizador es completamente nueva, ya que en todos los proyectos anteriores se había empleado SCPI y en este por primera vez se utiliza COM. Por tanto se ha empleado un driver que nos proporciona el fabricante en su página web a partir del cual se construye un objeto que nos permitirá interactuar con el analizador (ver Capítulo 4 Desarrollo del trabajo). Destacar que el driver solo estaba disponible para sistemas de 32 bits, por lo que habría que emplear una versión de Matlab también de 32 bits, lo cual junto con otros condicionantes que comentaremos a continuación supuso un pequeño problema de inicio.

La idea inicial era trabajar en el ordenador personal, ya que suponía una mayor comodidad en todos los aspectos, sin embargo éste era de 64 bits, por tanto se planteaban dos posibilidades: instalar bien una partición, bien una máquina virtual que trabajase con 32bits, o por el contrario emplear un ordenador distinto. Sin embargo posteriormente se vio que el sistema MD-2 garantizaba el correcto funcionamiento en Windows XP con 32bits. Teniendo en cuenta esto, además del problema comentado anteriormente respecto al driver de Matlab, se optó por trabajar directamente en un ordenador con sistema operativo Windows XP 32 bits.

Por cuestiones de compatibilidades se optó por desarrollar todo el trabajo en la versión de Matlab 2009, ya que si se realiza en una versión antigua y después el mismo programa se emplea en una más moderna, sería muy raro que apareciesen problemas de algún tipo, pero si por el contrario realizamos el trabajo en una versión actual y posteriormente precisamos ejecutarlo en versión previas, es bastante fácil que aparezcan problemas de librerías y compatibilidad.

3.1.1 COM vs SCPI[2]

Como hemos dicho, el analizador de redes empleado nos permitía trabajar con dos tipos de programación: SCPI y COM.

[2] Información extraída de la fuente [7]

SCPI (Estándar Commands for Programmable Instruments): define un estándar para la sintaxis y comandos a emplear en el control de dispositivos de prueba y de medición programables.

El hecho de tratarse un sistema estándar para todos los dispositivos es lo que hace muy sencillo su uso y facilita encontrar todos los comandos necesarios para realizar cualquier tipo de acción. Además, aunque originalmente fue creado para GPIB (IEEE-488), permite otros tipos de comunicaciones físicas: RS-232, Ehternet, USB, VXIbus, HiSLIP.

COM (Component Object Model): plataforma de Microsoft para componentes software. Permite la comunicación en cualquier lenguaje de programación que permita la creación y empleo de objetos. Emplea protocolo binario en la conexión, lo cual permite invocar directamente cualquier característica del analizador.

En trabajos anteriores siempre se había empleado SCPI, ya que los fabricantes facilitaban las librerías para sus dispositivos y era mucho más sencillo trabajar directamente con los comandos SCPI.

Para nuestro trabajo se planteó el estudio de ambos métodos para comprobar si alguno de los dos nos proporcionaba una ventaja sustancial respecto al otro o si por el contrario era totalmente indiferente emplear uno u otro. Si no fuese porque COM emplea protocolo binario, nos sería indiferente emplear cualquiera de los métodos, sin embargo el hecho de utilizar este protocolo hace que COM sea mucho más eficiente que SCPI y los tiempos de espera de instrucciones se acorten. Es por esta razón que se optó por emplear este tipo de programación. La interacción con el analizador se realiza por medio de un objeto creado a partir del driver, lo cual como veremos posteriormente supuso ciertos problemas a la hora de programar (ver Capítulo 4 Desarrollo del trabajo).

3.2 Estructuración de las tareas

El desarrollo del trabajo se encuentra dividido en varias secciones bien diferencias, independientes e interconectadas al mismo tiempo entre sí.

1. Búsqueda de información y documentación necesaria
   1.1 Documentación acerca del uso de la herramienta de diseño gráfico GUIDE. (Diciembre 2015)
   1.2 Documentación acerca del uso y programación del analizador N5227A. (Enero 2016)

2. Realización interfaz gráfica encargada de controlar el analizador de redes
   2.1 Método de creación del objeto a partir del driver para programación COM. (Febrero 2016)
   2.2 Comunicación del PC y del analizador. (Febrero 2016)
   2.3 Diseño interfaz gráfica del analizador. (Marzo 2016/Mayo 2016)

3. Realización interfaz gráfica encargada de control el posicionador XY. (Junio 2016)
4. Unión de ambas interfaces para el control de la sonda de medida. (Junio 2016)
5. Desarrollo y redacción de la memoria documentada del trabajo. (Mayo 2016/ Junio 2016)

3.3 Diagrama temporal

Tarea	DICIEMBRE	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO
1	X	X					
2			X	X	X	X	
3						X	X
4							X
5						X	X

Tabla 1. Diagrama Temporal

3.4 Presupuesto

Material:

- Analizador de redes N5227A: 184.397,00€
- Antenas: 1.794,79€ (cada una)
- Mesa posicionador Arrick Robotics: 1.483.3€
- Sistema MD-2 Arrick Robotics: 1.200€
- Sopores de antenas, transiciones, cables de red: 250€
- PC: 600€

TOTAL: 191.519,88€

Capítulo 4. Desarrollo del trabajo

En este capítulo de la memoria se expone lo que ha sido el desarrollo en sí del trabajo. En primer lugar, explicamos detalladamente el proceso de creación del objeto de Matlab necesario para emplear la programación COM. Seguidamente se comenta el método para establecer conexión tanto con el analizador como con el sistema encargado de controlar el movimiento de los motores de la mesa.

Posteriormente antes de comenzar a detallar lo que sería el desarrollo de la interfaz, se comentan los fundamentos básicos que hay que conocer para saber desenvolverse con la herramienta GUIDE de Matlab. Hecho esto ya pasamos con las interfaces por separado y luego con ambas anexionadas en una única.

Para finalizar comentamos los problemas que nos han surgido al ir avanzando en nuestro trabajo y como los hemos solucionado.

4.1 Creación del objeto en Matlab

Como ya se ha comentado antes, para el desarrollo del trabajo se optó por el tipo de programación COM, para lo cual el fabricante facilita un driver que contiene una serie de librerías para poder interactuar con el analizador. Matlab facilita herramientas de creación de objetos a partir de drivers importados, a continuación explicamos los pasos a seguir.

4.1.1 Importar driver con Midedit

La herramienta *Midedit* (Matlab Instrument Driver Editor) nos permite importar, modificar y crear drivers, así como definir sus propiedades y funciones.

En las Figuras 9-13 se muestran detalles de cómo crear el objeto en Matlab a partir del driver facilitado por el fabricante.

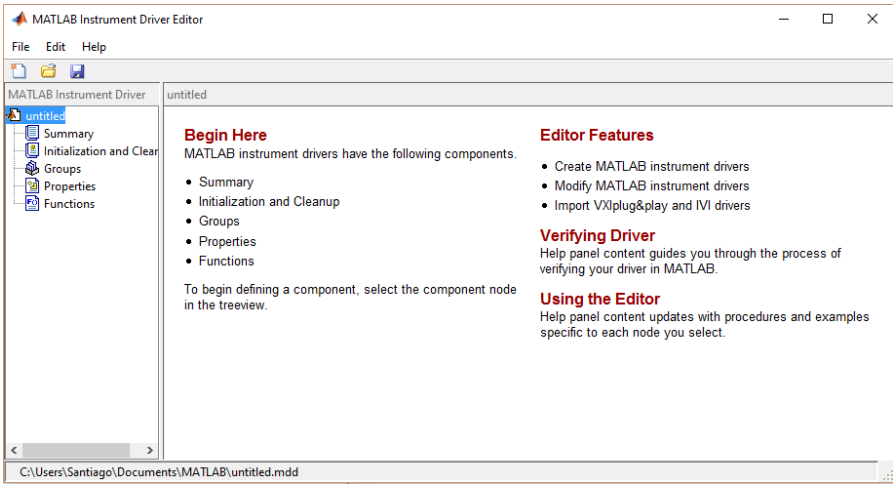


Figura 9. Midedit

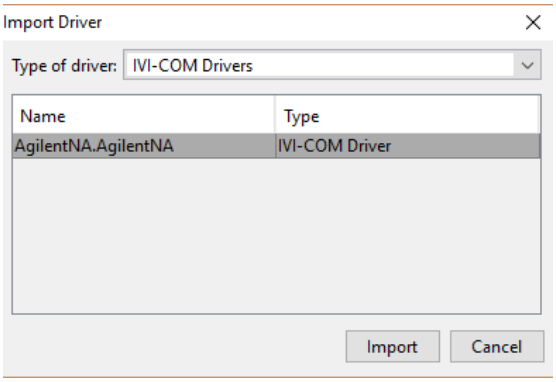


Figura 10. Importar Driver

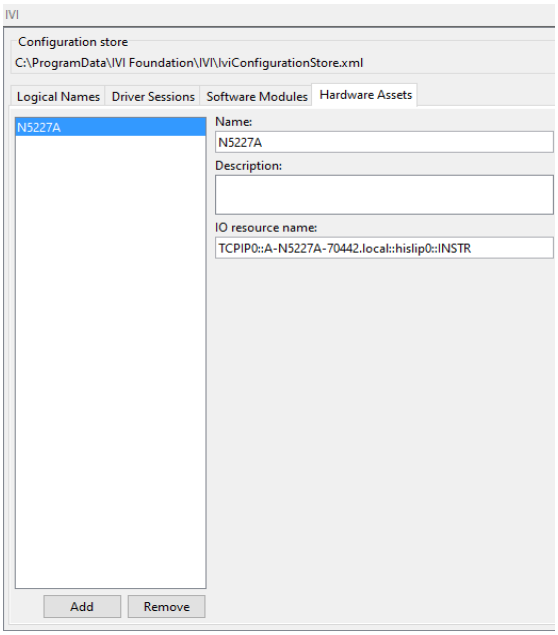
Una vez importamos el driver lo guardamos con un nombre característico que nos permita identificarlo posteriormente: AgilentNAmdd

4.1.2 Creación del IVI driver

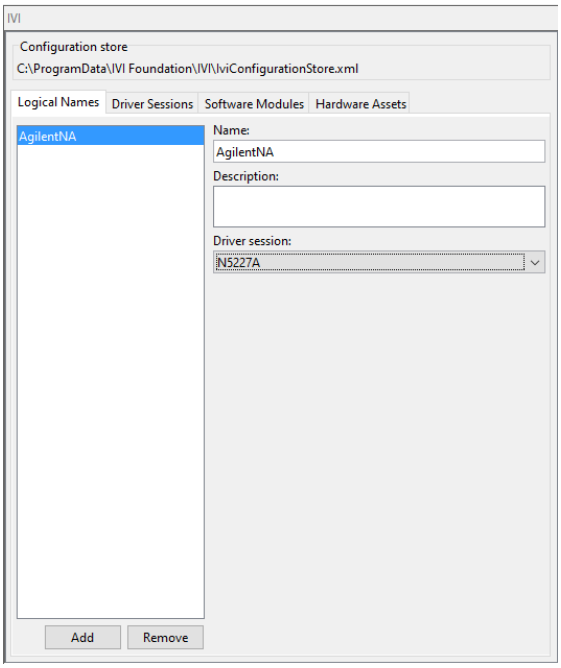
Una vez hemos importado nuestro driver, empleamos otra herramienta que nos facilita Matlab, *tmtool*. Desde esta ventana podemos crear un IVI driver, que utilizará como referencia el driver que hemos importado anteriormente. Necesitamos especificar el nombre IO del hardware empleado. En nuestro caso al estar realizando conexión vía LAN con protocolo TCP/IP tiene el siguiente aspecto: TCPIP0::A-N5227A-70442.local::hislip0::INSTR.

En la pestaña de *Software Modules* nos aparece directamente el correspondiente a nuestro analizador. En *Driver Sessions* nos aparecen dos pestañas con el nombre de las dos ventanas anteriores, por lo que aquí deberíamos seleccionar lo que hemos configurado anteriormente. Resaltar que el driver nos permite seleccionar una opción de simulación, mediante la cual aunque no dispongamos físicamente del analizador, Matlab "crea" un analizador virtual y podemos probar a enviar y recibir instrucciones en caso de que no dispongamos en ese momento del analizador.

Por último en la pestaña de *Logical NAmes* lo único que tendremos que hacer es seleccionar el Driver Session que hemos configurado en la pestaña anterior y ponerle un nombre identificativo, que será a través del cual llamemos a nuestro IVI driver.



(a)

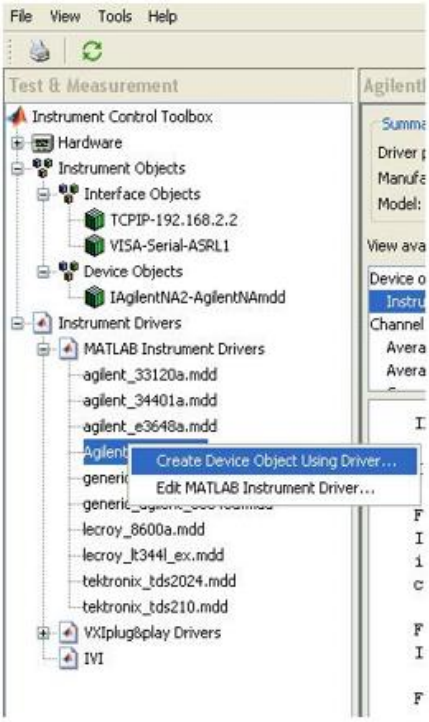


(b)

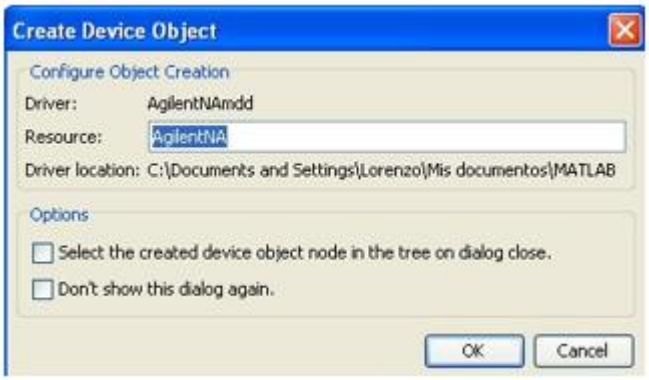
Figura 11. (a) y (b) Creación IVI driver

4.1.3 Creación del objeto

En el último paso simplemente tenemos que combinar los dos apartados anteriores. Sobre el driver importado en el primer apartado seleccionamos la opción de crear un objeto, y como recurso definimos el IVI driver creado en el segundo apartado.



(a)



(b)

Figura 12. (a) y (b) Creación del objeto

De este modo ya tenemos el objeto a través del cual nos comunicaremos con el analizador mediante protocolo TCP/IP a través de cable LAN.

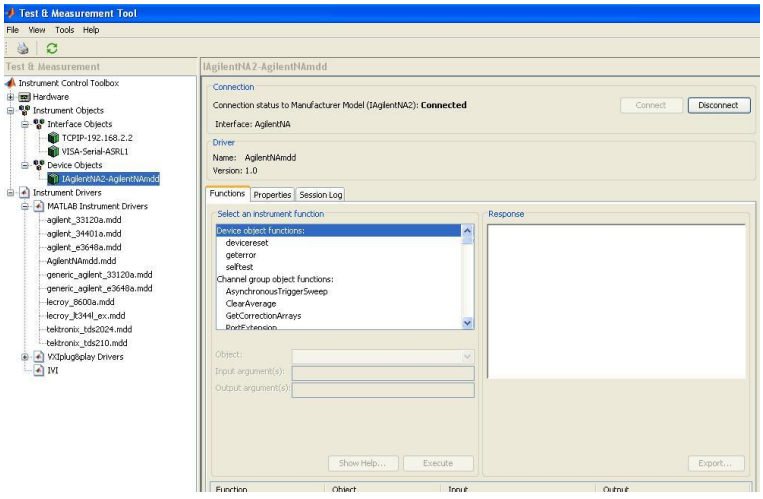


Figura 13. Conexión con el analizador mediante el objeto

4.2 Establecer conexión con el analizador

En orden a poder controlar remotamente el analizador desde el PC es preciso que configuremos la conexión vía LAN a través de un cable Ethernet creando una Red de Área Local. El proceso no supone una gran dificultad, ya que básicamente consiste en asignar unas IPs determinadas a ambos dispositivos, de manera que sea inmediata la conexión.

Se va a explicar el proceso en el sistema operativo Windows XP (para PC) y en Windows 7 (para analizador), y en ambos casos los pasos a seguir son idénticos. Sin embargo es posible que si se utilizase algún sistema operativo posterior como Windows 8 o Windows 10 el aspecto de algunas de las ventanas podría cambiar.

La forma más sencilla de acceder a la configuración de red es mediante: *Inicio->Panel de Control->Conexiones de Red*. Una vez aquí seleccionamos la opción de propiedades del icono de *Conexión de área local*.

En las Figuras 14-16 se muestra el detalle de cómo se establece la conexión del PC con el analizador.

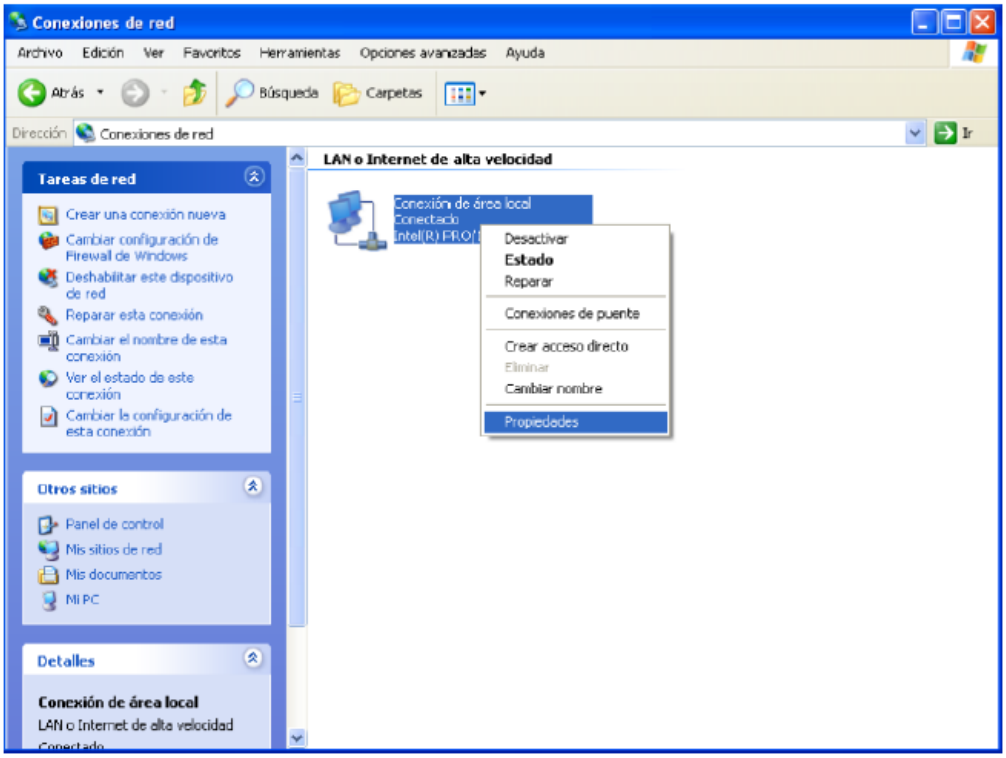
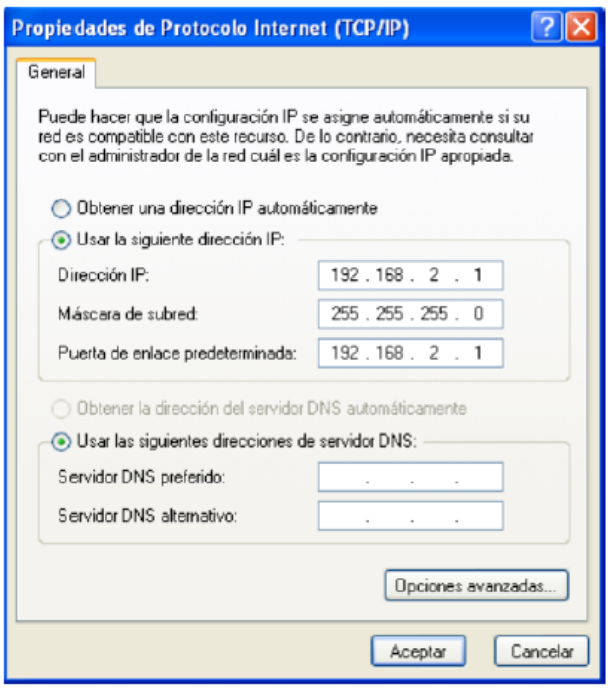


Figura 14. Configurar red de área local 1

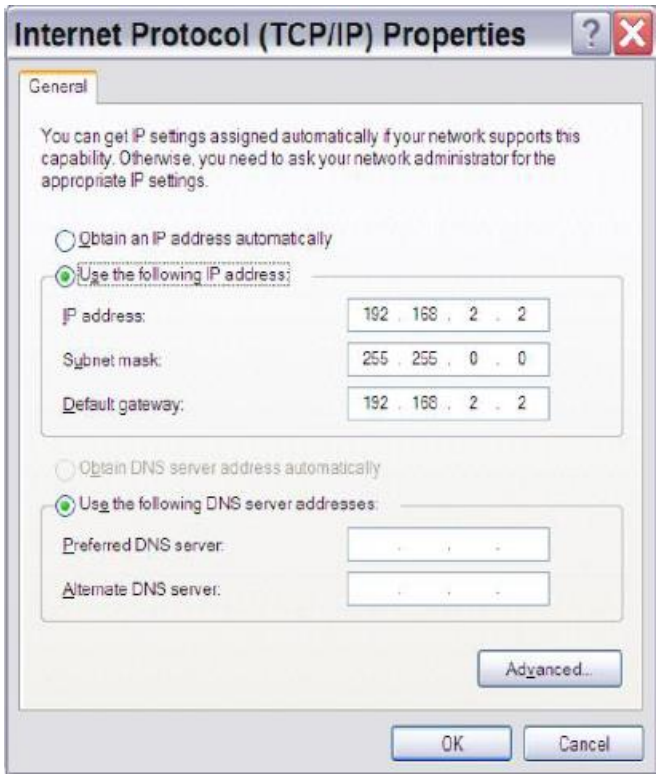
Una vez dentro de propiedades buscamos el protocolo TCP/IP y volvemos a pinchar en propiedades.

Figura 15. Configurar red de área local 2

Una vez que estamos dentro de esta ventana ya podemos asignar manualmente la IP que deseemos, así como la máscara de subred, que deberá ser común en ambos dispositivos. En nuestro caso hemos optado por asignar al PC la IP 192.168.2.1 y al analizador la inmediatamente posterior 192.168.2.2. La máscara de subred será 255.255.255.0.



(a)



(b)

Figura 16. (a) y (b) Configurar red de área local 3 y 4

4.3 Establecer conexión con el posicionador

NOTA: Hemos decidido no extendernos en exceso en este apartado, ya que en trabajos anteriores se ha hecho uso del mismo sistema y se encuentra todo perfectamente documentado por lo que sería repetir una vez más toda la información.

El sistema con el que estamos trabajando se conecta a nuestro PC por vía puerto paralelo.

En las dos figuras siguientes se muestra la conexión del PC con el sistema de posicionamiento y un detalle de los motores empleados.

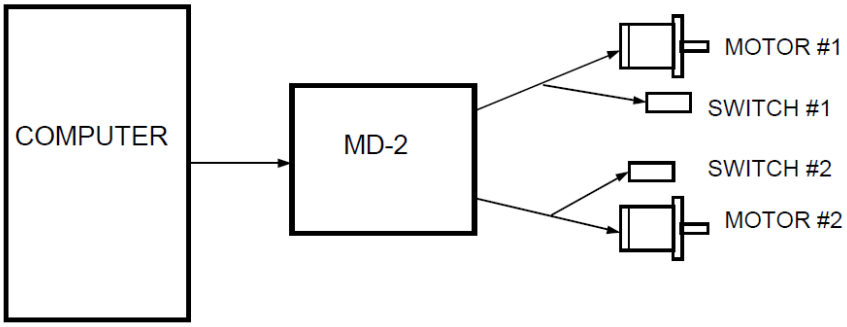


Figura 17. Conexión sistema MD-2

Figura 18. Detalle motor

Una vez conexionado todo lo único que se debe hacer es comprobar que el puerto paralelo de nuestro PC se encuentra con la configuración adecuada. Esto lo podemos comprobar desde el *Administrador de Dispositivos* de nuestro PC.

En las Figuras 19-22 se muestra el proceso para establecer conexión entre PC y posicionador.

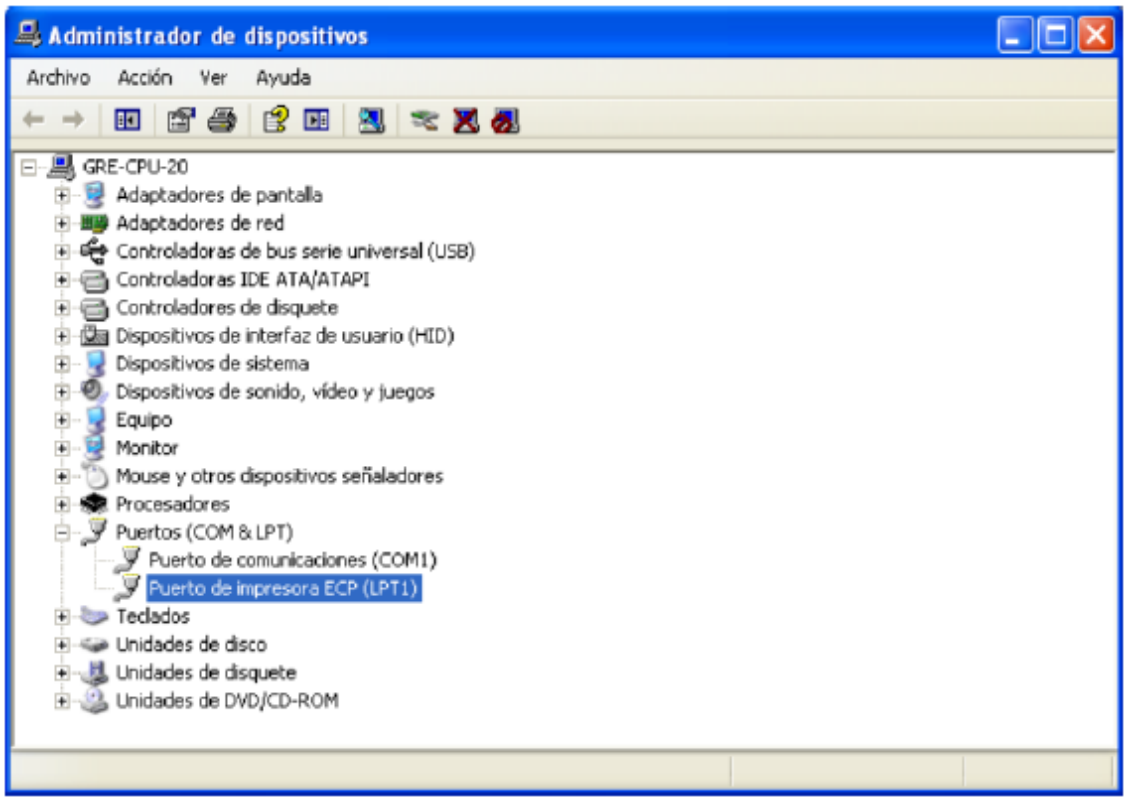


Figura 19. Configuración puerto paralelo 1

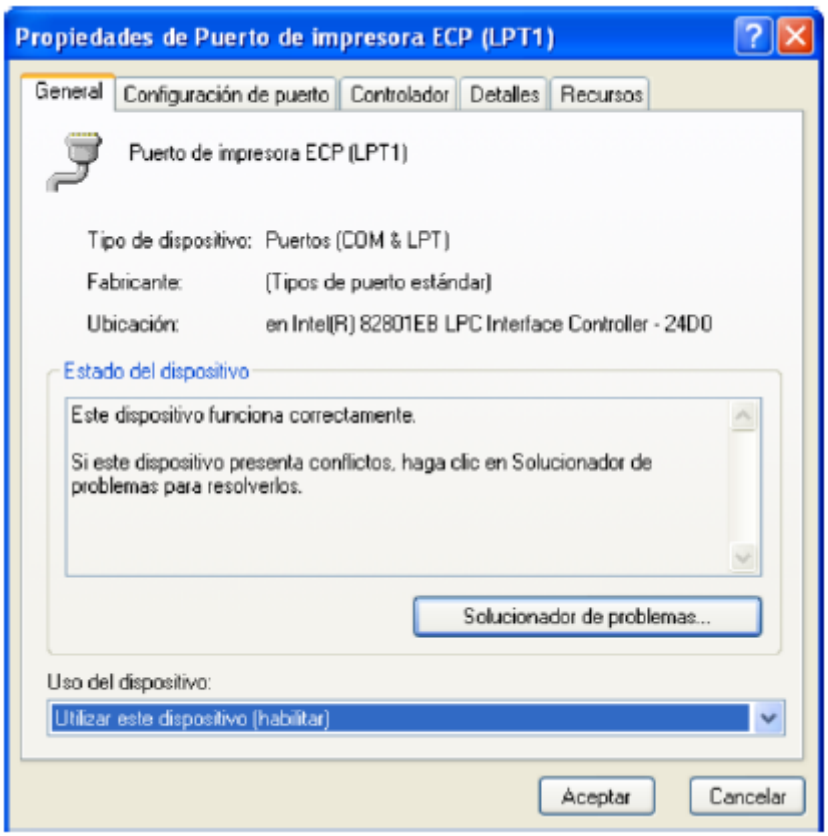


Figura 20. Configuración puerto paralelo 2

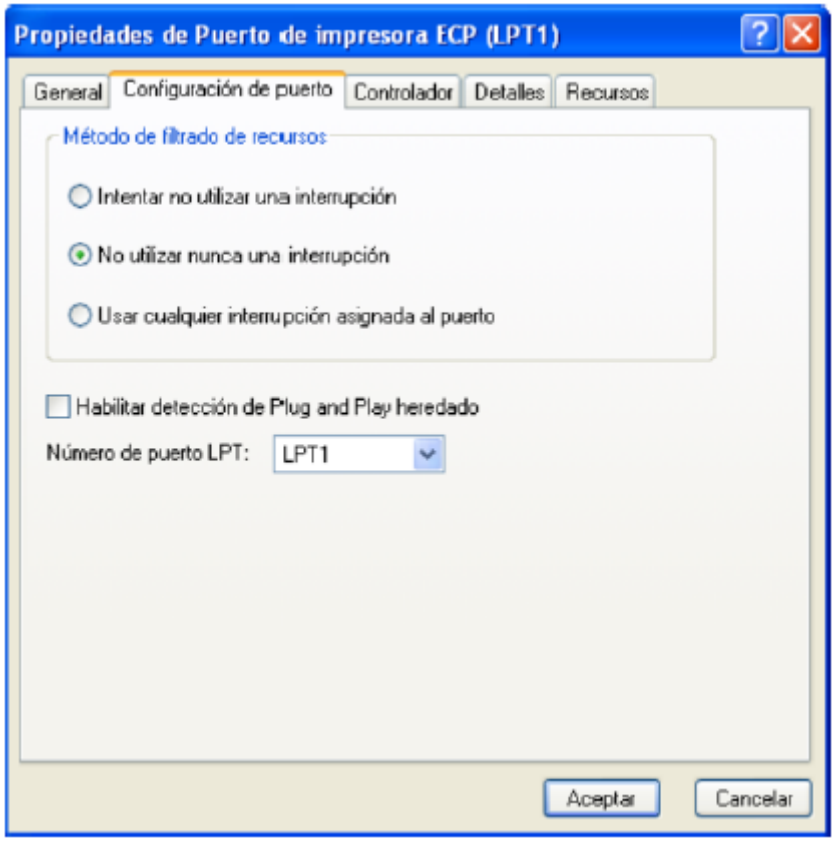


Figura 21. Configuración puerto paralelo 3

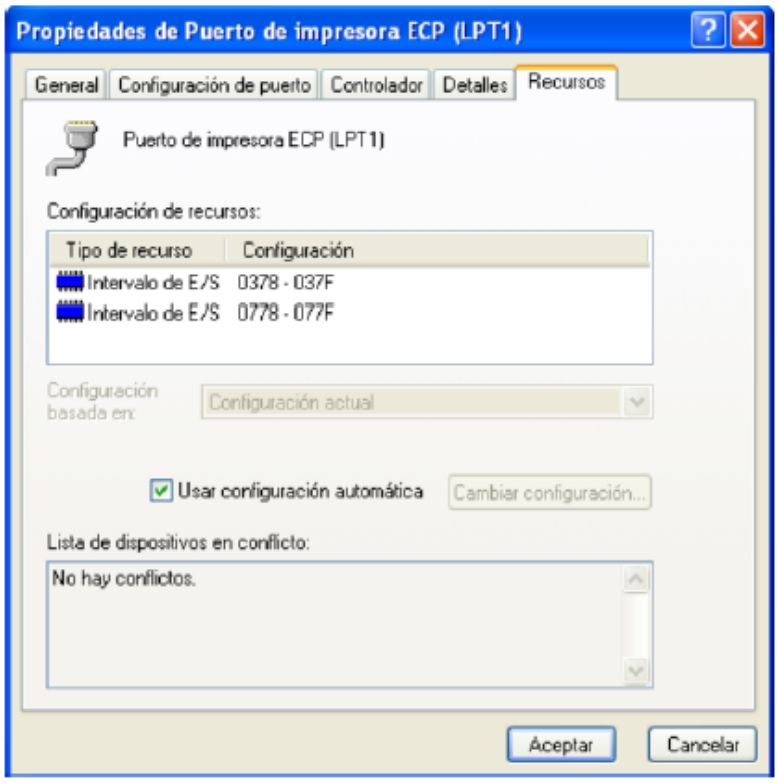


Figura 22. Configuración puerto paralelo 4

Posteriormente lo único que tenemos que hacer es comprobar que efectivamente hemos configurado todo de manera apropiada. Para esto el fabricante nos facilite el software a partir del cual podemos indicar que se mueva cualquiera de los dos motores tanto como queramos o que vuelvan a la posición inicial. En la figura inferior se muestra la pantalla principal del software de Arrick Robotis.

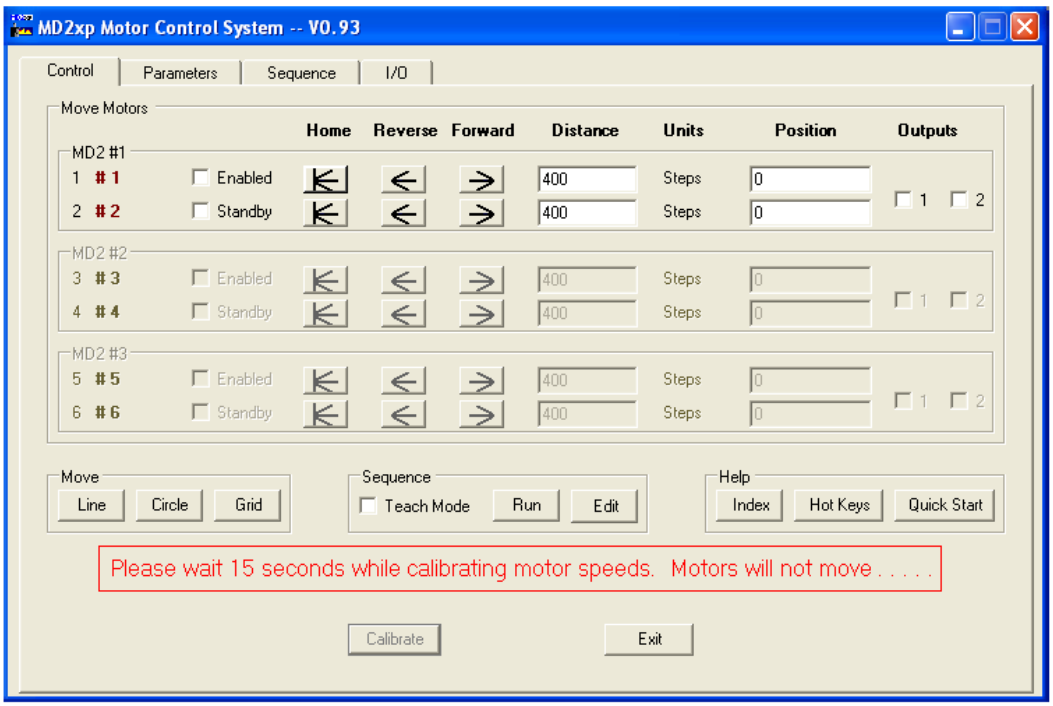


Figura 23. Software de Arrick Robotics

Cada vez que ejecutamos el programa este realiza de manera automática la calibración, y genera un archivo con dicha calibración. Es importante que antes de ejecutar el programa comprobemos que no existe ningún archivo de calibración previo, y si es así debemos eliminarlo, ya que si no podría darse algún problema al calibrar.

4.4 Diseño interfaz gráfica

4.4.1 Herramienta GUIDE[3]

GUIDE es un entorno de programación visual de Matlab que permite realizar y ejecutar programas mediante una interfaz gráfica e ingresando datos de forma continuada. Posee las características básicas de otros programas visuales como puedan ser Visual Basic o Visual C++.

Para ejecutar la herramienta podemos optar por escribir en la ventana de comandos la instrucción >>guide o por el contrario seleccionar el icono resaltado en la siguiente imagen.

En las Figuras 24-25 se muestra el detalle de como ejecutar la herramienta GUIDE y que aspecto tiene.

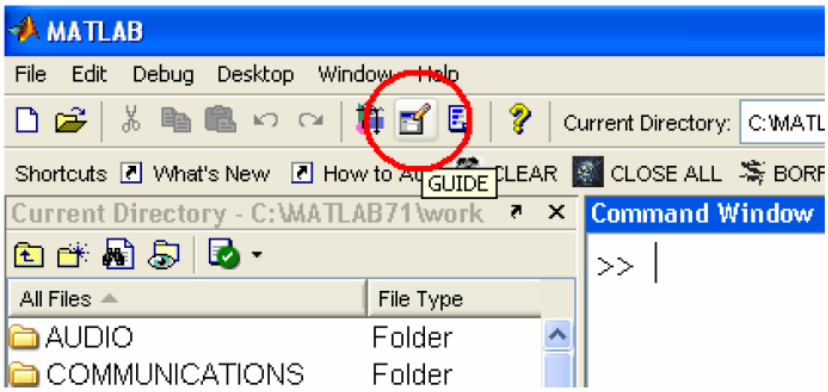


Figura 24. Inicializar herramienta GUIDE

Posteriormente se nos pregunta si queremos escoger alguna plantilla predefinida para nuestro diseño o queremos un diseño en blanco. También se nos ofrece la posibilidad de abrir plantillas previas que tengamos diseñadas.

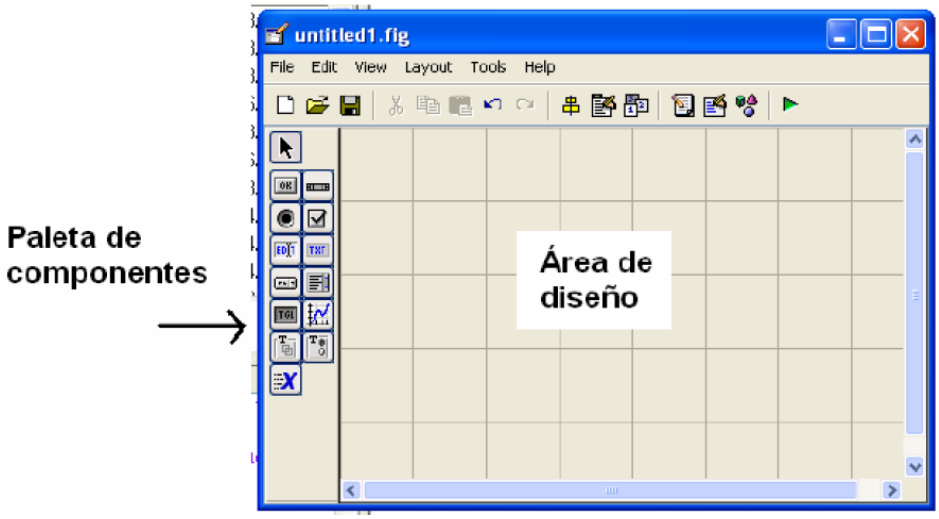


Figura 25. Aspecto ventana de diseño GUIDE

[3] Información extraída de la fuente [8]

Como se puede ver en la imagen anterior, la herramienta nos brinda una serie de elementos con los que trabajar:

Nombre del Elemento	Descripción
Check box	Indica el estado de una opción o atributo
Editable Text	Caja para editar texto
Pop-up menu	Provee una lista de opciones
List Box	Muestra una lista deslizable
Push Button	Invoca un evento inmediatamente
Radio Button	Indica una opción que puede ser seleccionada
Toggle Button	Solo dos estados, “on” o “off”, ”forward” o “reverse”, etc.
Slider	Usado para representar un rango de valores
Static Text	Muestra un string de texto en una caja
Panel button	Agrupa botones como un grupo
Button Group	Permite exclusividad de selección con los radio button

Tabla 2. Elementos de diseño GUIDE

Estas interfaces se guardan con el formato .fig, formato propio de Matlab que se emplea para mostrar por pantalla la interfaz diseñada y los resultados requeridos. Sin embargo paralelamente se genera un archivo .m con el mismo nombre que el anterior, en el cual se encontrará todo el código relativo a nuestro diseño. Por cada elemento que colocamos en el área de diseño Matlab genera una o dos funciones en base al tipo de elemento del que se trate.

- *CreateFcn:* función encargada de definir las condiciones iniciales del objeto al que representa (por lo general estas funciones no se modifican en ningún momento de modo manual).
- *Callback:* función que es llamada cada vez que modificamos cualquier elemento de los que hemos colocado en el diseño. En estas funciones será en las cuales implementemos todo el código correspondiente a cada uno de los elementos para que se ejecute cada vez que los empleemos.

Los valores de todas las propiedades de los elementos, así como las variables transitorias, son almacenados en una estructura, a la cual se accederá por medio de un único identificador común para todos ellos:

```
handles.output = hObject;
```

Tenemos una estructura (handles) en la cual vamos almacenando todas las variables que creamos y sus valores. De este modo podemos usar las variables que queramos en cualquier función de nuestro programa. Sin embargo, si queremos que se apliquen los cambios que hemos hecho, al final de la función tenemos que ejecutar la siguiente instrucción:

```
guidata(hObject, handles);
```

A la hora de modificar las propiedades de los elementos no es necesario estar llamando a la estructura que hemos creado y tener que estar escribiendo tanto código, sino que se puede emplear la ventana del *Property Inspector* a través de la cual tenemos acceso directo a todas las propiedades y podemos cambiarlas a nuestro gusto.

Por último es importante conocer el modo en el que realizamos la asignación y obtención de valores de los componentes. Para esto emplearemos las sentencias *set* y *get* respectivamente.

- `set(handles.TagDelElemento,'NombrePropiedad','Valor');`
  Establecemos el valor de la propiedad del elemento indicado.

- `get(handles.TagDelElemento,'NombrePropiedad');`

Obtenemos el valor de la propiedad del elemento indicado.

4.4.2 Interfaz para controlar el analizador

NOTA: Tanto en este apartado como en el siguiente se integran y se aplican todos los conceptos explicados en los apartados anteriores de la memoria, por lo que antes de continuar se recomienda leer todo si no se ha hecho previamente.

Aunque el trabajo originalmente estaba pensado para automatizar las medidas usando una mesa como posicionador, a la hora de realizarlo se optó por diseñar por un lado la parte correspondiente a la interfaz gráfica que se iba a encargar de controlar el analizador y por otro la parte gráfica pertinente para manejar la mesa. De este modo se puede dotar al sistema de una mayor versatilidad, ya que se puede optar por tomar las medidas usando la mesa o sin usarla y así ampliar las funcionalidades iniciales para las que estaba concebido el proyecto.

En primer lugar, vamos a empezar explicando cómo se ha desarrollado la interfaz encargada de controlar las funciones del analizador, ya que esta es la base de nuestro trabajo.

En las Figuras 26-42 las imágenes que se muestran se corresponden con las distintas funciones configuradas para controlar el analizador.

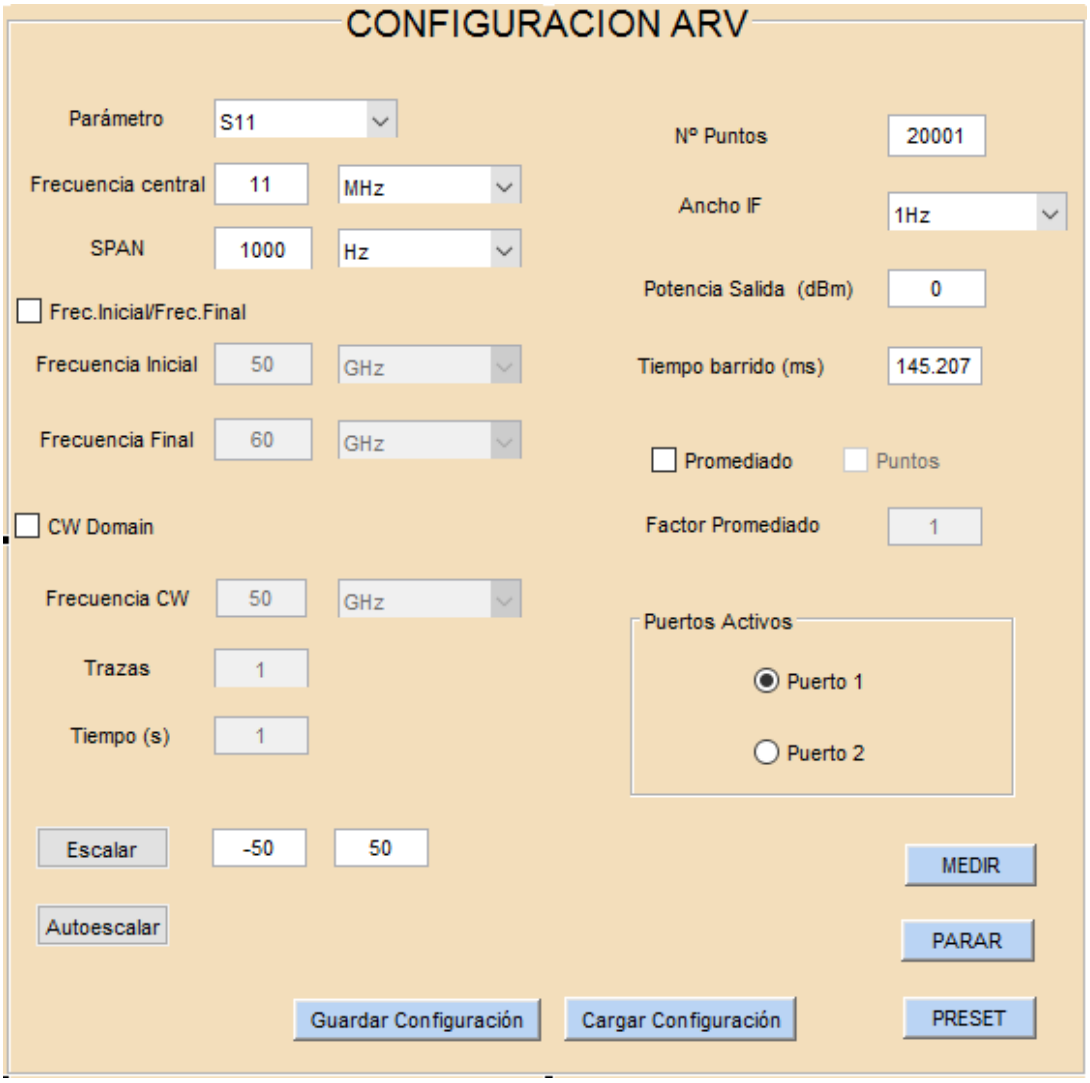


Figura 26. Aspecto interfaz gráfica del analizador

SELECCIÓN DE PARÁMETRO:

El primer elemento que nos encontramos a la hora de configurar un analizador de redes, es el parámetro que queremos medir. En el caso del N5227A se nos permite medir una gran variedad de parámetros.

Hemos optado por facilitar la posibilidad de mostrar en primer lugar los parámetros de dispersión: S11, S12, S21 y S22. Seguidamente también se permite seleccionar los parámetros: A1, A2, B1, B2, a1, a2, b1, b2. De este modo, aunque el diseño inicial de la sonda de medidas en nuestro trabajo se basaría en estudiar el comportamiento de los parámetros de transmisión S12 y S21, hemos habilitado también la medida de otros parámetros en caso de que se quiera utilizar el sistema de medida para otras aplicaciones.

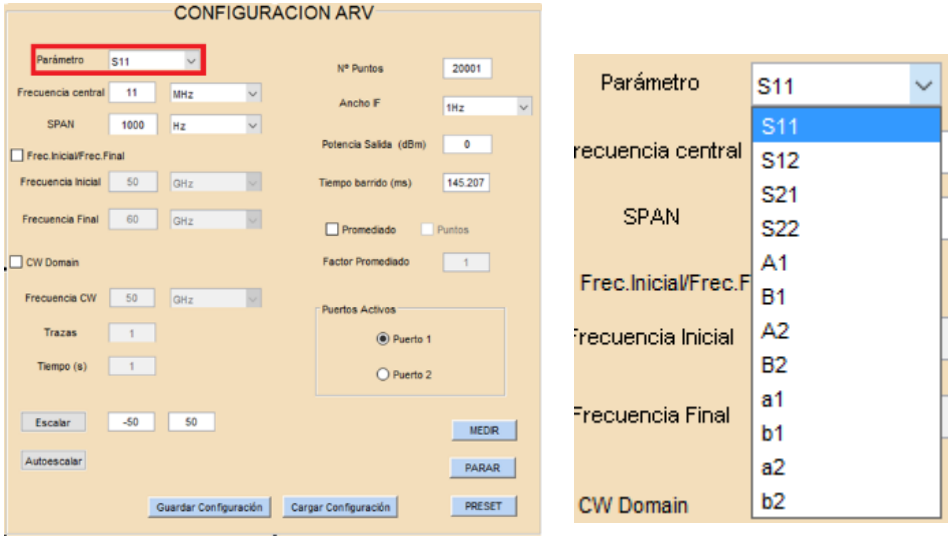


Figura 27. Selección de parámetro

SELECCIÓN DE FRECUENCIAS:

Para la selección de frecuencias a medir se plantean dos posibilidades igualmente válidas:

1. Configurar una frecuencia central y un SPAN que nos determinará las frecuencias inicial y final. (fini=fc-SPAN/2, ffin=fc+SPAN/2).

2. Configurar directamente frecuencia inicial y frecuencia final.

Aunque en trabajos anteriores no se había hecho, en el nuestro hemos optado por facilitar al usuario el poder trabajar con ambas opciones, ya que dependiendo de la medida que se quiera hacer puede interesar más seleccionar frecuencias de un modo u otro.

Ejemplo: queremos realizar medidas alrededor de una frecuencia determinada pero no tenemos muy claro dónde empezar y dónde terminar: Se emplearía en este caso **fc y SPAN.**

Tenemos una frecuencia inicial y final definida y queremos medir todo lo que pasa en ese ancho de banda: Se emplearía en este caso **fini y ffin.**

De tal modo que mediante un *Checkbox* podemos seleccionar si trabajamos de uno u otro modo. En el momento en que una de las dos opciones está habilitad automáticamente la otra se deshabilita y todos los botones correspondientes quedan no disponibles (ver figura inferior).

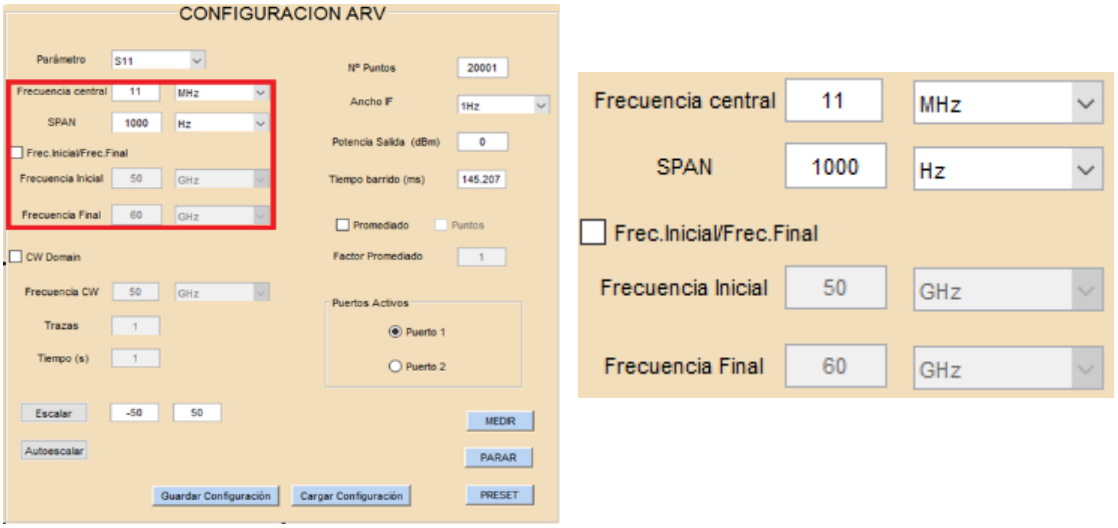
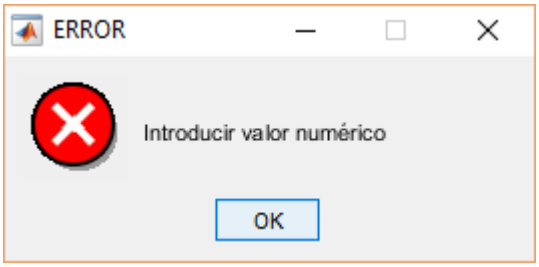
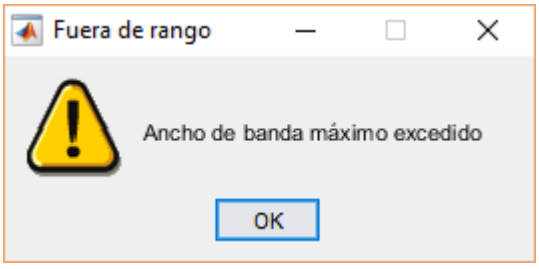


Figura 28. Selección de frecuencias

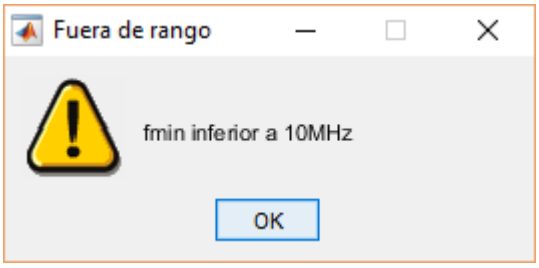
En ambos casos se han aplicado las restricciones pertinentes en cuanto a los valores que nos permite introducir el analizador. Éste es capaz de trabajar desde los 10MHz hasta los 70GHz, por lo que teniendo en cuenta esto se han configurado ventanas emergentes de aviso en caso de incumplir alguna de las condiciones. Se muestran a continuación.



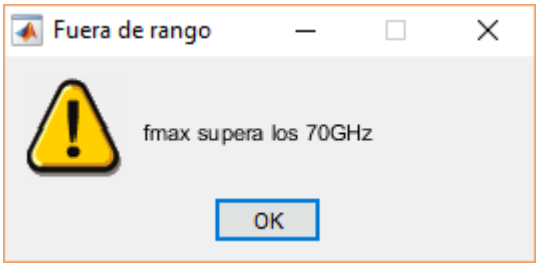
(a)



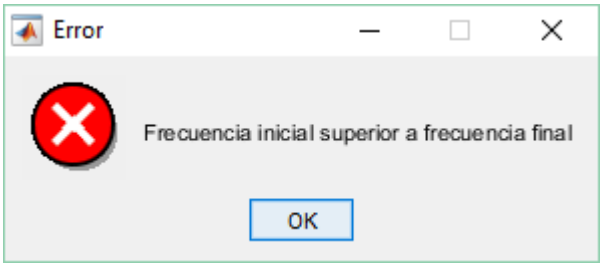
(b)


Fuera de rango
fmin inferior a 10MHz
OK

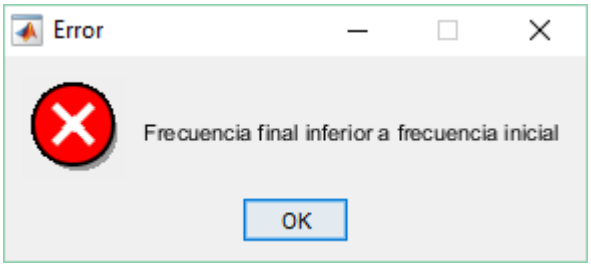
(c)


Fuera de rango
fmax supera los 70GHz
OK

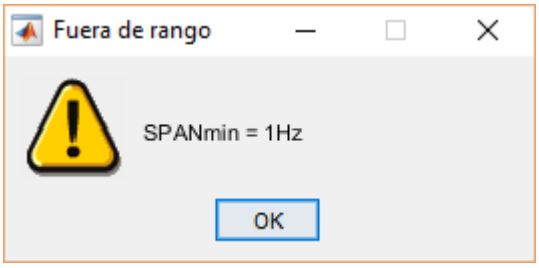
(d)


Error
Frecuencia inicial superior a frecuencia final
OK

(e)


Error
Frecuencia final inferior a frecuencia inicial
OK

(f)


Fuera de rango
SPANmin = 1Hz
OK

(g)

Figura 29. (a-g) Mensajes de error para control de frecuencias

Cada vez que se genera uno de estos avisos el programa automáticamente se encarga de establecer un valor por defecto valido que solvente el error.

MODO ONDA CONTINUA (CW):

Visto como se encuentra desarrollado el contro y selección de frecuencias, pasamos a explicar el modo de onda continua (CW). El analizador nos brinda la opción de entrar en este modo, en el cual se mide durante un tiempo determinado una única frecuencia. Es importante implementar el control de este modo ya que puede ser de interés estudiar cómo se comporta una antena, una cavidad… trabajando a una frecuencia determinada con el paso del tiempo.

En este caso también hemos hecho uso de un *Checkbox* a través del cual optaremos por trabajar en frecuencia o en onda continua. De igual manera se han creado ventanas de diálogo emergente en caso de infringir algún parámetro, similares a las empleadas para seleccionar frecuencias.

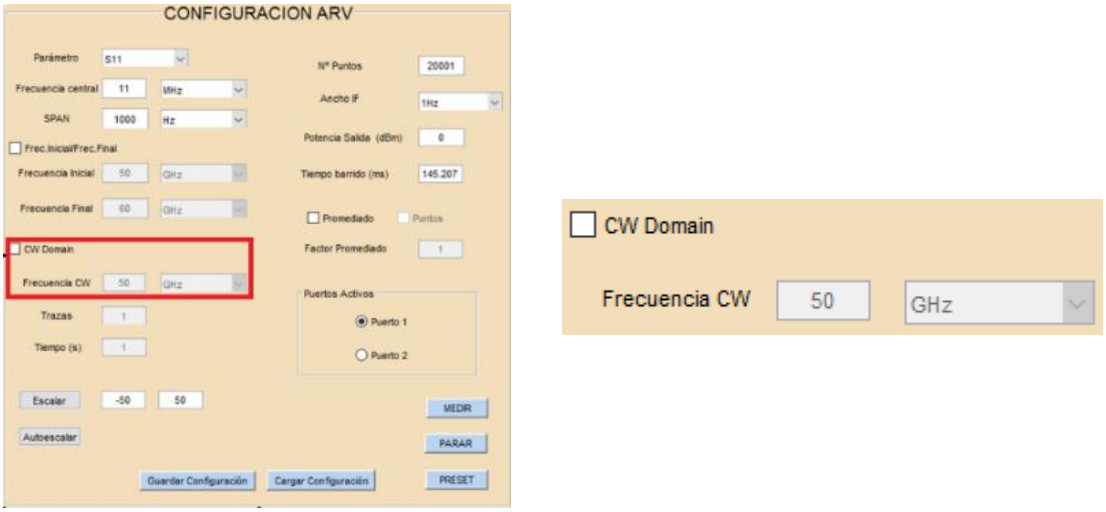


Figura 30. Selección modo onda continua (CW)

Una vez descritas las maneras de trabajar con el analizador, bien en frecuencia, bien en onda continua, pasamos a hablar de parámetros generales.

NÚMERO DE PUNTOS:

El número de puntos sirve para conseguir una mejor o peor resolución de la traza, sin embargo el hecho de aumentar los puntos hace que el barrido sea más lento. Existe la posibilidad de encontrar el número óptimo de puntos para la medida que estamos realizando. Esto es cuando aunque sigamos aumentando el número de puntos no supone un cambio significante en la medida.

En el caso del N5227A el número máximo de puntos que permite por traza es de 100001. Se han establecido las restricciones necesarias para que como mínimo el número de puntos sea 1 y como máximo 100001, de modo que en caso de error el programa lo solventa por sí mismo.

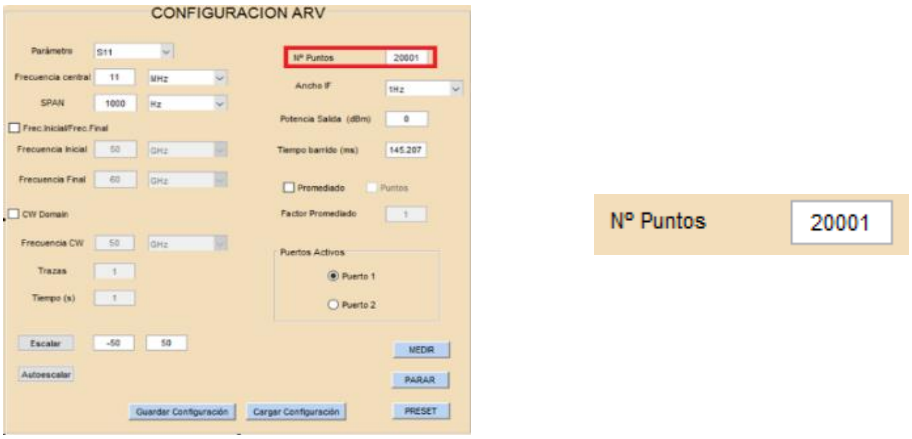


Figura 31. Número de puntos

TIEMPO DE BARRIDO:

Este parámetro se encuentra estrechamente ligado con el anterior, ya que el número de puntos nos va a determinar el tiempo de barrido mínimo necesario y en caso de intentar establecer un valor inferior el analizador no nos lo permitirá. Por tanto, el valor que aparece en el *Edit_Text* correspondiente al tiempo de barrido se verá modificado cada vez que modifiquemos el nº de puntos, ya que el programa leerá ese tiempo mínimo del analizador y será el que nos sitúe en forma de *String* en la casilla.

Una vez que tenemos ese tiempo mínimo, si intentamos asignar un tiempo inferior nos saltará un aviso indicando que se ha introducido un valor inferior al mínimo permitido. De tal modo que en todo momento se podrá establecer un tiempo de barrido superior al mínimo para el número de puntos indicado, pero en ningún caso ese tiempo podrá ser inferior.

El valor del tiempo de barrido también se verá modificado como consecuencia de variar el ancho IF, lo cual tendremos en cuenta.

Figura 32. Tiempo de barrido

POTENCIA DE SALIDA:

En cuanto a la potencia de salida por puerto, el analizador admite cualquier valor en el rango de -30dBm a 30dBm. Sin embargo, realizando pruebas a distintas frecuencias se comprobó que en ningún caso se podía llegar a ese valor obteniendo una respuesta balanceada.

Desde 10MHz hasta 38GHz el analizador admitía una potencia de salida máxima por puerto de 13dBm, mientras que de 38GHz hasta 70GHz el valor máximo se reducía a 10dBm. En caso de seleccionar potencias superiores a estos valores el analizador nos muestra por pantalla un aviso de que se está superando el nivel de potencia recomendado y la respuesta que se obtiene no está siendo balanceada.

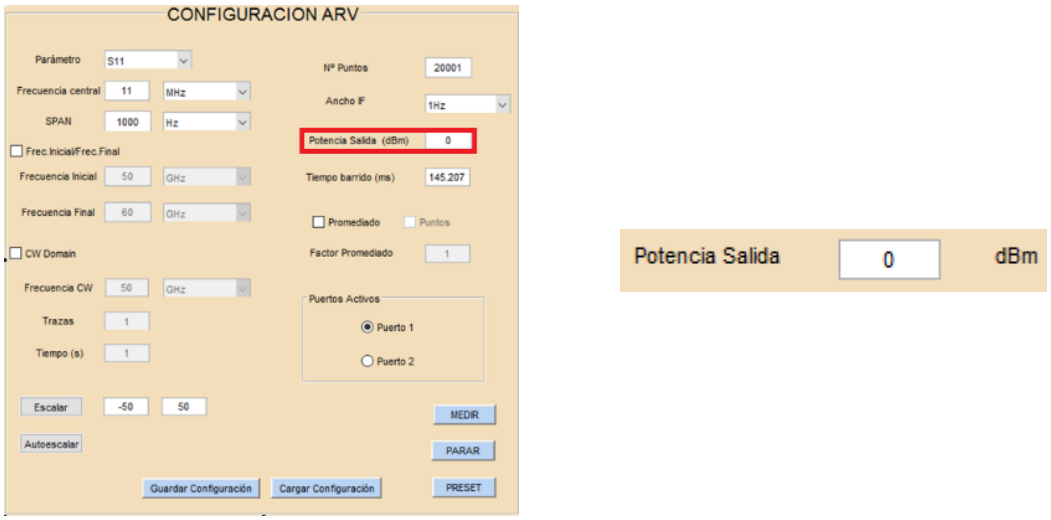


Figura 33. Potencia de salida

PUERTOS ACTIVOS:

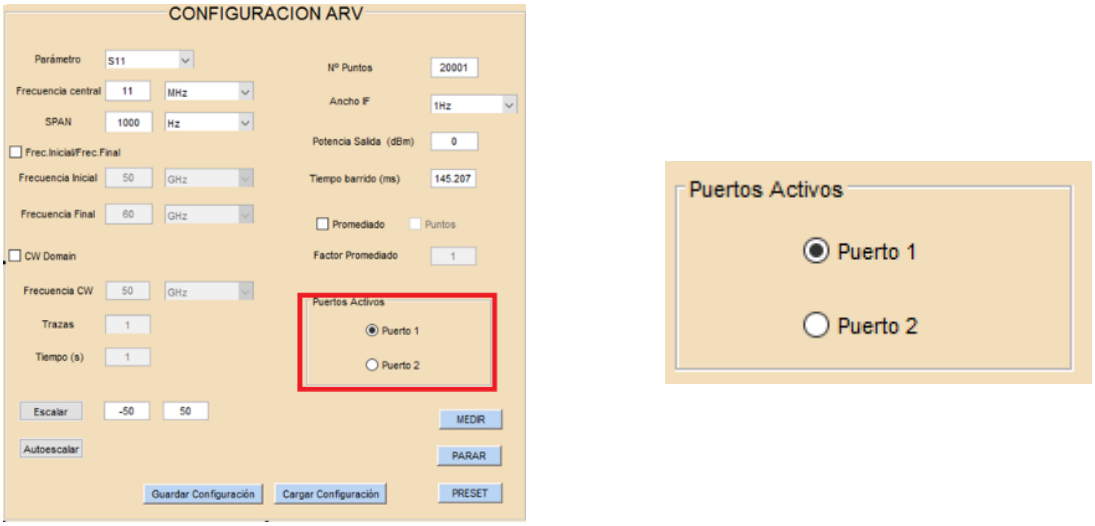


Figura 34. Puertos Activos

Se han habilitado dos botones mediante los cuales podemos activar o desactivar los dos puertos de los que dispone nuestro analizador, de manera que podemos escoger sacar potencia por ambos a la vez, por sólo uno de ellos o por ninguno. Por defecto se ha establecido como activo el puerto 1 al inicializar el analizador ya que la medida que aparece por pantalla se corresponde con el parámetro S11.

ANCHO IF:

Con este parámetro se consigue modificar el ancho de banda del filtro paso banda de frecuencia intermedia. Reducir este ancho de banda resulta en una reducción del efecto del ruido aleatorio. Sin embargo, esto también implica que el tiempo de barrido aumentará según se estrecha el IF bandwith.

Cada analizador permite establecer determinados valores. En el caso del N5227A el valor mínimo es de 1Hz y el máximo llega hasta los 15MHz.

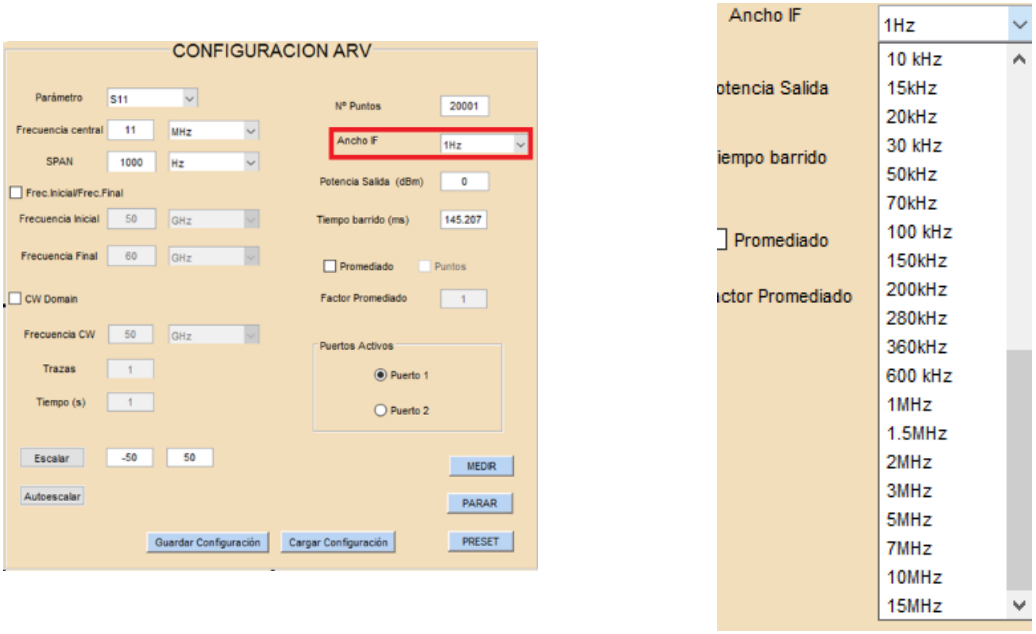


Figura 35. Ancho de banda IF

Aunque es cierto que cuando más pequeño es el valor ganamos en prestaciones en cuanto a ruido y también a resolución, puede haber casos en los que resulte interesante el trabajar con valores elevados, como podría ser medir la potencia total asociada a una banda de frecuencias o medir el ruido.

PROMEDIADO:

La función de promediado del analizador nos permite reducir el ruido en nuestras medidas. Por defecto viene desactivado, y el factor de promediado es 1. Sin embargo, si lo activamos nos permite un factor de promediado de hasta 65536 (2^{16}).

Nos permite realizar promediado por barrido o punto a punto, por lo que hemos habilitado la posibilidad de trabajar con ambos mediante un *CheckBox*.

- Barrido: cada punto se forma a partir de la media del mismo punto a lo largo de los barridos.
- Punto: cada punto se mide el número de veces indicado en el factor de promediado y se hace la media, y entonces ya se puede pasar a medir el siguiente punto.

La fórmula de promedio por barrido es la siguiente:

$$NuevaMedia = \left(\frac{NuevoDato}{n}\right) + \left(\frac{AntiguaMedia}{n} * (n-1)\right) \quad (4.1)\ [9]$$

Donde n es el factor de promediado.

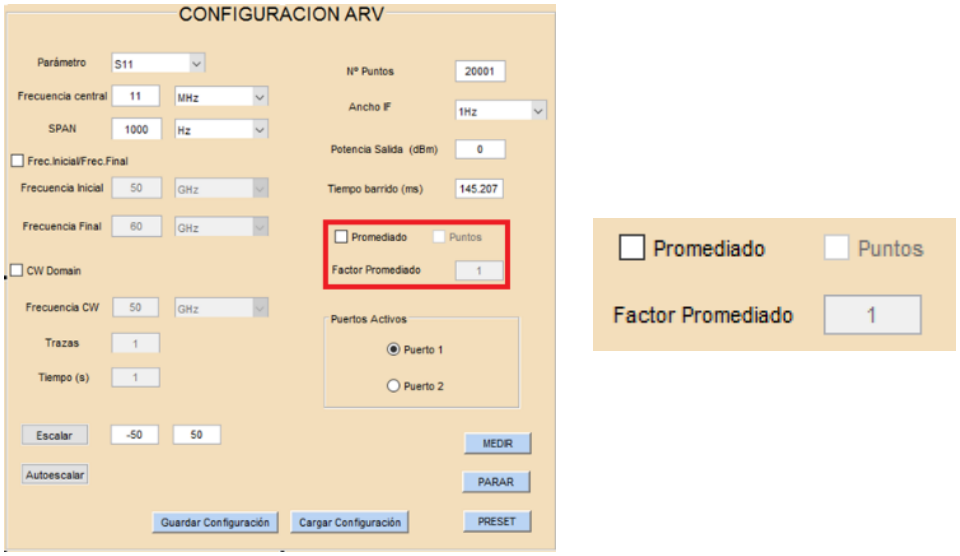


Figura 36. Promediado de medida

TRAZAS:

En el caso de que estemos trabajando en el modo de onda continua, a la hora de realizar las mediciones nos va a interesar realizar múltiples de ellas, de tal modo que debemos habilitar una casilla para seleccionar las veces que queramos que se dispare el *Trigger* y así realizar las medidas que consideremos oportunas.

El analizador nos permite realizar hasta 65536 (2^{16}) trazas. El número de trazas que queramos realizar será lo que nos determine el tiempo durante el cual estaremos midiendo. Mediante esta casilla únicamente vamos a controlar que el número introducido este dentro de los márgenes permitidos (1 a 65536), ya que las pertinentes instrucciones para ejecutar los X disparos del *Trigger* se encontrarán en la función encargada de control el botón *Medir* (explicación más adelante).

Al tratarse de una función que queremos utilizar únicamente para las mediciones en onda continua, siempre que no estemos en dicho modo la casilla permanecerá deshabilitada.

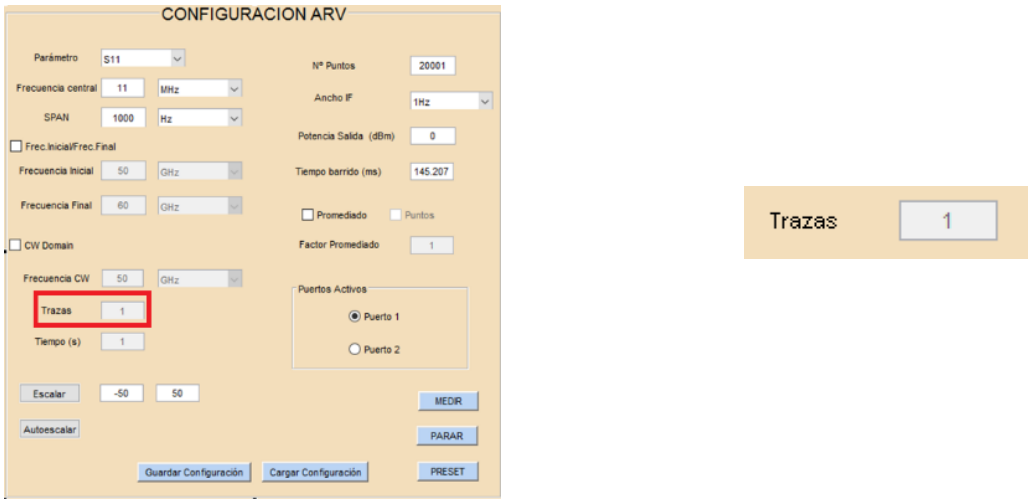


Figura 37. Número de trazas

TIEMPO:

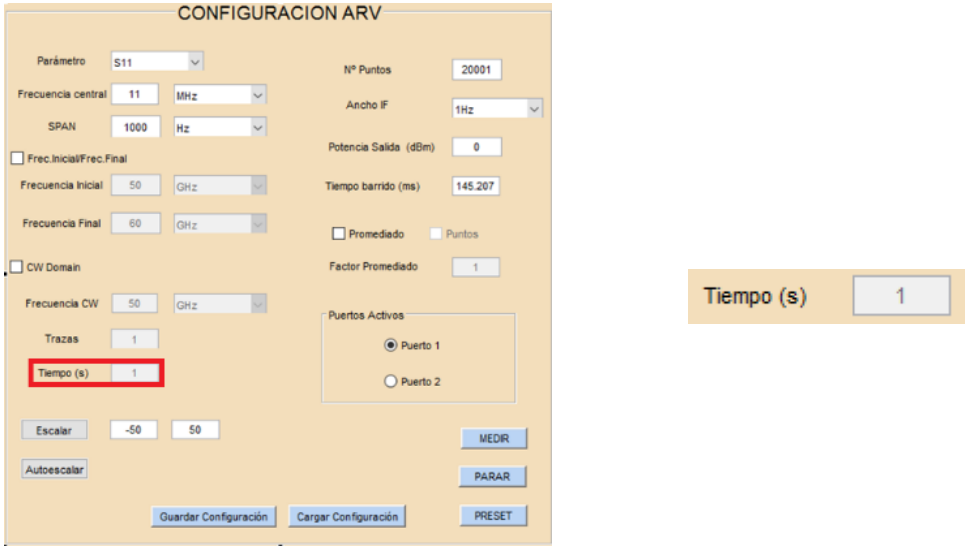


Figura 38. Tiempo medida en CW

Del mismo modo que disponemos de la opción de establecer un número determinado de trazas a medir cuando estamos trabajando en el modo de onda continua, también es interesante el que podamos definir un tiempo determinado para la medida. Teniendo en cuenta este tiempo, así como el tiempo de barrido podemos pasar a determinar el número de trazas que se medirán con una simple división, por lo tanto, conseguimos la misma funcionalidad a partir de un parámetro que puede ser más interesante emplear en función del tipo de medición que estemos realizando.

Las únicas dos limitaciones que se tienen en cuenta en este caso es que el tiempo sea al menos suficiente como para realizar una traza (tiempo de barrido) y que tampoco se supere el número de trazas máximo permitido por el analizador (65536).

ESCALAR/AUTOESCALAR:

Figura 39. Definir escala

Al cambiar el parámetro que estamos midiendo, así como sus propiedades es habitual que la medida se nos descentre en la pantalla. Es por esto que resulta interesante implementar un botón que le ordene al analizador que realice un autoescalado.

De igual forma también se permite al usuario introducir por sí mismo entre que valores quiere que se muestre la medida por pantalla. El analizador permite mostrar un rango de 6000dB, desde -3000dB a 3000dB, con un máximo de 500dB por división. Es importante señalar que el analizador internamente introduce una limitación extra: la diferencia en valor absoluto entre el valor máximo de la parte positiva de la escala y el valor mínimo de la parte negativa de la misma, no puede exceder los 1000dB, de tal modo que en la programación se han considerado estas limitaciones y en caso de que el usuario trate de quebrantar uno de estos límites será informado mediante un mensaje emergente y el programa establecerá de forma autómata un valor válido.

GUARDAR/CARGAR CONFIGURACIÓN:

- Guardado:

En lo referente a guardar la configuración de parámetros con la que estamos trabajando se nos plantearon dos posibilidades: guardarla en el PC, o guardarla en el propio analizador.
A nivel de programación resultaba mucho más sencillo realizar un volcado de datos directamente en el propio analizador, ya que con una simple instrucción que indicase que queríamos guardar el estado actual y la ruta en la que guardar el archivo era más que suficiente. Este archivo se guarda en un formato propio del analizador (.sta).

```
groupObj = get(handles.VNAObj, 'System');

groupObj = groupObj(1);

invoke(groupObj, 'SaveState', 'C:\Users\Public\Documents\Network
Analyzer\guardado1.sta');
```

Sin embargo, se optó por guardarla en el pc, para una mayor comodidad de transporte y visualización. Esto es así porque se escogió crear una variable de tipo estructura (*Struct*) en Matlab en la cual se irían almacenando todos los parámetros dentro de dicha estructura, y de

este modo desde el propio Matlab sin necesidad de tener el analizador presente se podría cargar esta variable y ver su contenido.

```
parametros.NombreParametro=get(handles.TagParametro,'Value');
```

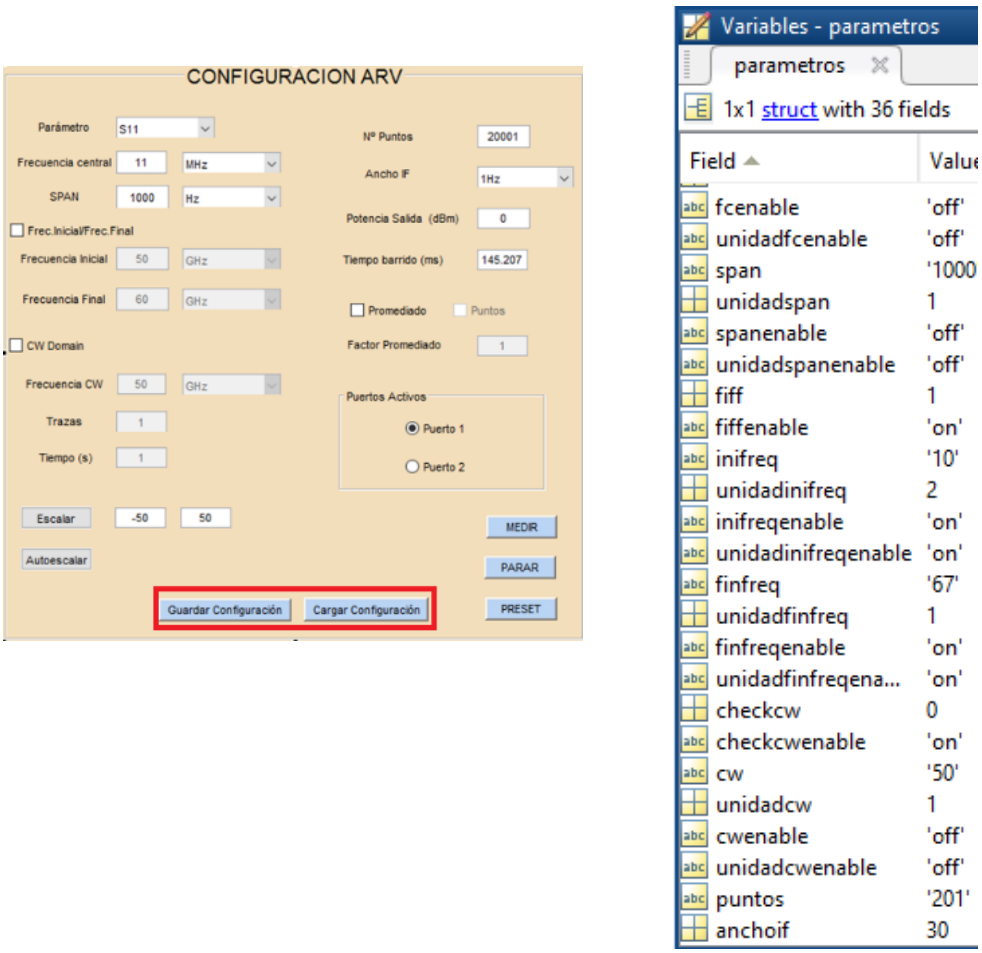


Figura 40. Ejemplo de estructura con los parámetros guardados

Una vez que tenemos todos los parámetros dentro de la estructura basta con especificar la ruta del PC para guardar la estructura y ejecutar la instrucción *Save*.

```
%Especificamos el nombre del archivo y la ruta en que queremos
%guardarlo.

[parametros.nombreArchivo parametros.ruta]=uiputfile('C:\Documents and
Settings\Lorenzo\Mis documentos\MATLAB\GuardarConfig\*.mat','Guardar
Configuración');

newfilename=fullfile(parametros.ruta, parametros.nombreArchivo);

save(newfilename, 'parametros'); %Guardamos la configuracion del
analizador en el PC.
```

- Cargar:

Del mismo modo que cuando estábamos guardando la configuración, en el momento de cargarla dependerá de donde la hemos guardado. Una vez más si hubiésemos optado por guardar en el analizador para cargar todos los parámetros bastaría con llamar a la función *RecallState* e indicarle el directorio.

```
groupObj = get(handles.VNAObj, 'System'); %

groupObj = groupObj(1);

invoke(groupObj, 'RecallState', 'C:\Users\Public\Documents\Network Analyzer\guardado1.sta');
```

En nuestro caso el proceso de carga se ejecutará en dos pasos. En el primero cargamos la estructura que tenemos guardada en un directorio X de nuestro pc.

```
[parametros.nombreArchivo parametros.ruta]=uigetfile('C:\Documents and Settings\Lorenzo\Mis documentos\MATLAB\GuardarConfig\*.mat','Cargar Configuración');

newfilename=fullfile(parametros.ruta, parametros.nombreArchivo);

load(newfilename, 'parametros');
```

En segundo lugar, una vez que tenemos la estructura cargada en Matlab podemos proceder a cargar los datos, proceso que a su vez también se hará en dos pasos: carga de parámetros en nuestra interfaz y carga de parámetros en el analizador.

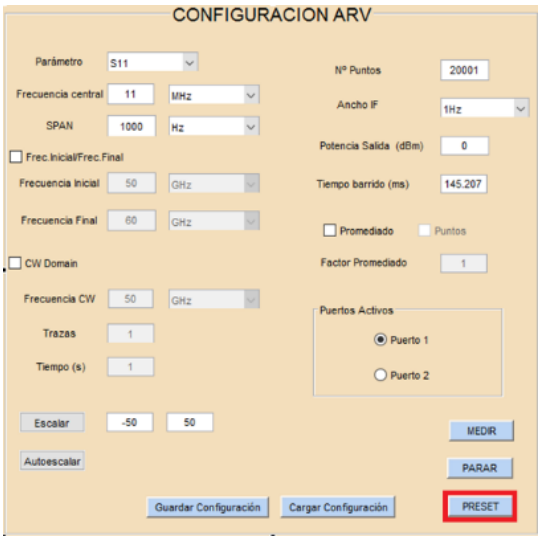
En ambos casos empleando el comando (mirar subapartado 4.4.1) dado por la instrucción *Set* podemos asignar a cada elemento el valor que le corresponde.

```
set(handles.TagDelElemento,'NombrePropiedad','Valor');
```

PRESET:

El analizador nos ofrece la posibilidad de realizar un preset y regresar a una configuración predeterminada. Se trata de una simple instrucción que se envía al pulsar el botón de preset. Además de esto, cuando pulsamos el botón desde nuestro programa, es necesario que los valores que en éste nos aparecen por pantalla en referencia a todos los parámetros del analizador se actualicen con los valores por defecto que tiene asignado el analizador cuando se realiza el preset.

Esto se hará de nuevo mediante la línea de comando dada por la instrucción *Set*.



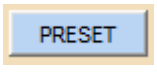


Figura 41. Botón de preset

MEDIR:

El desarrollo de esta función es posiblemente el más complicado de nuestro diseño. Esto es así porque desde ella se controlan todos los tipos de mediciones que podemos realizar en los modos que nos permite el analizador anteriormente descritos y básicamente porque hay que llevar un control exhaustivo de los tiempos de ejecución de instrucciones, ya que si no se pueden dar casos de solapamientos de medidas y sobreescritura de ficheros.

Lo primero que habría que indicar es que se ha optado por guardar todas las mediciones en el propio analizador. Se hubiese podido hacer el volcado de datos igualmente en el PC, sin embargo, teniendo en cuenta que el analizador tiene entradas de puertos USB y al fin y al cabo trabaja con un sistema operativo de Windows no se consideró necesario. Todos los archivos se guardan en el formato .csv, puesto que de este modo además de poder ser interpretados por el propio analizador, cuando exportemos los datos a nuestro PC para trabajar con ellos, estos podrán ser abiertos tanto desde Matlab como desde Excel, facilitándonos así el posterior procesamiento de las medidas.

Si pasamos a hablar de los tipos de mediciones que podemos guardar conforme hemos programado nuestra interfaz podemos diferenciar 4:

1. Medida en frecuencia sin promediado: Es la opción más fácil de medida ya que no hay que tener en cuenta ni tiempos de promediado ni de disparo de *Trigger*. Simplemente se guarda lo que en ese momento tengamos en pantalla.

2. Medida en frecuencia con promediado de barrido: Podemos realizar una medida en una determinada banda de frecuencias utilizando promediado por barrido. En este caso hay que tener en cuenta el factor de promediado empleado para saber cuánto debe esperar Matlab hasta ejecutar la instrucción de guardar.

   Mediante comandos SCPI podemos saber en qué estado de barrido se encuentra el analizador, por lo que manera más sencilla resultó ser configurar el promediado con un factor X y en base a éste efectuar tal número de disparos con el *Trigger* de modo que cuando terminase pasaría a estado *Hold* entonces con un simple bucle que comprobase si hemos llegado o no a tal estado se puede saber en qué momento hemos terminado de promediar y podemos guardar la medida.

3. Medida en frecuencia con promediado punto a punto: Función similar a la anterior con la diferencia de que en este caso el promediado se realiza punto a punto. De igual modo hay que considerar el factor de promediado para el tiempo de espera.

   En este caso la forma de saber cuánto hemos de esperar reside en el tiempo de barrido, ya que en este tipo de promediado, cuando establecemos un factor de promediado determinado, el tiempo de barrido cambia, por lo tanto conocemos exactamente el tiempo que va a tardar en realizar la traza completa (podríamos establecer un valor de tiempo mayor si quisiésemos, pero en ningún caso menor), por lo que simplemente hay que indicarle a Matlab que espere ese tiempo antes de guardar la medida.

4. Medida en onda continua (CW): Si estamos trabajando en onda continua, tenemos un número determinado de trazas que guardar. Por tanto, teniendo en cuenta el tiempo de barrido y el número de trazas, se le puede indicar a Matlab que espere un tiempo determinado a que se genere una traza completa y poder guardarla antes de que se ejecute el siguiente disparo y así evitar solapamiento de medidas.

Nota: *a la hora de determinar el tiempo de espera entre disparo y disparo también hay que tener en cuenta lo que tarda Matlab en ejecutar la instrucción encargada de guardar la medida. Tras realizar pruebas con la función tic/toc se comprobó que este tiempo era de aproximadamente 8ms, de tal modo que se optó por establecer 1 centésima de segundo para dejar así algo de margen.*

De tal modo que se ha creado una estructura de directorios en el analizador con diferentes carpetas para almacenar cada tipo de mediciones.

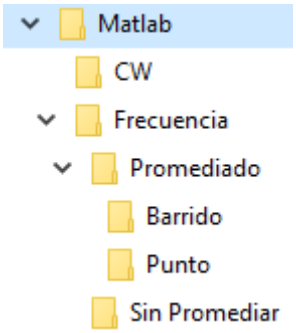


Figura 42. Directorios

Además del proceso de medida y guardado, en la programación de este botón también se tiene en cuenta el estado en que se encuentra el botón *PARAR* (explicado a continuación), ya que en caso de ser pulsado habrá que interrumpir la medida que estemos haciendo, lo cual haremos mediante un if que comprobará si ha sido pulsado o no y en caso de haberlo sido mediante un *Break* se interrumpirá la medida.

```
if get(handles.pushbutton_parar,'UserData')==1
    break
end
```

PARAR:

La función de este botón es exclusivamente parar la medida que se esté realizando independientemente del estado en el que se encuentre. Su programación se limita a una única línea de código.

```
set(handles.pushbutton_parar,'UserData',1);
```

Cuando pulsamos el botón configuramos el parámetros *UserData* a nivel alto, de tal modo que desde la función medir explicada anteriormente, podremos hacer una llamada a esta función para comprobar si el botón ha sido pulsado, caso en el cual se ejecuta la instrucción *Break* interrumpiéndose de este modo el proceso de medida.

4.4.3 Interfaz para controlar la mesa

La segunda parte de nuestro trabajo consistía en desarrollar otra interfaz con la herramienta GUIDE de Matlab a partir de la cual pudiésemos controlar una mesa.

En trabajos anteriores ya se había realizado algo similar, sin embargo, en nuestro trabajo hemos añadido una nueva forma de medida que comentaremos a continuación.

En las Figuras 43.55 se muestran las distintas funciones configuradas para controlar el posicionador.

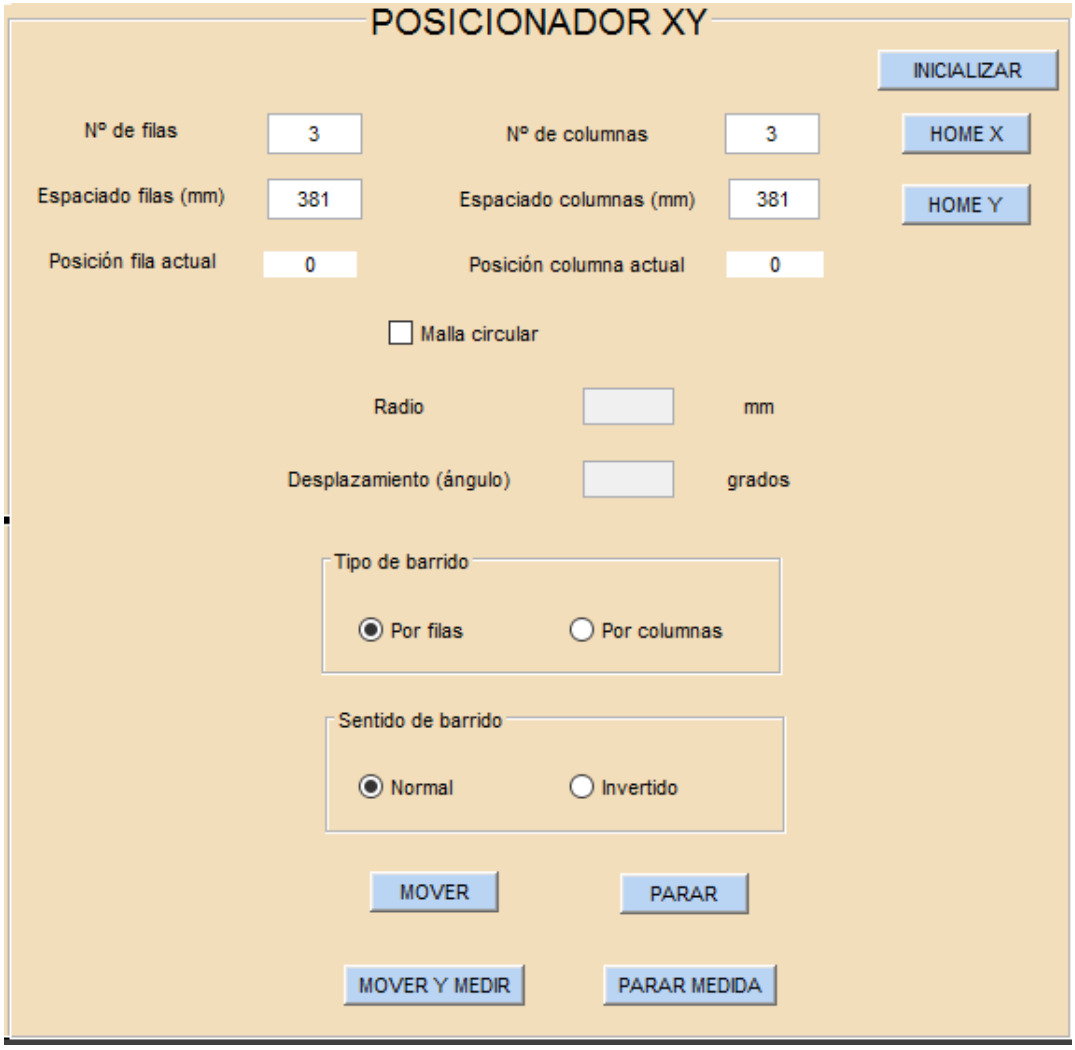
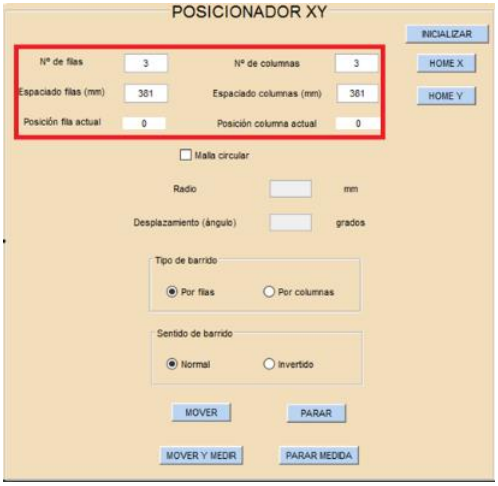


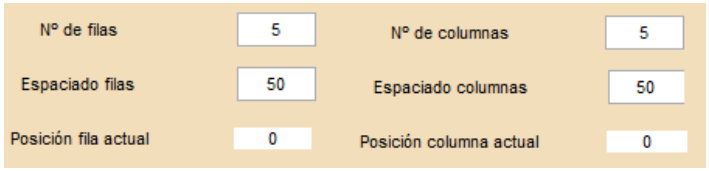
Figura 43. Aspecto interfaz gráfica posicionador

En éste caso, la programación de la interfaz es mucho más sencilla en líneas generales de lo que es la concerniente al control del analizador, ya que básicamente es aplicar las restricciones de movimiento correspondientes a las dimensiones de la mesa e indicarle a cada motor que se desplace los pasos correspondientes mediante un bucle.

REJILLA RECTÁNGULAR:



(a)



(b)

Figura 44. (a) y (b) Parámetros para moverse en una zona rectangular

En primer lugar, hemos decidido implementar las opciones correspondientes a realizar las medidas en una rejilla rectangular o lo que sería lo mismo, en una matriz. Los motores pueden moverse hasta 760 pasos desde la posición inicial en cada eje hasta el final del rail. Por lo tanto, podemos configurar una matriz de las dimensiones que queramos siempre y cuando el producto del número de filas o columnas por su correspondiente espaciado no supere dicho valor.

Como hemos dicho, el programa nos permite indicar el número de filas y columnas a medir, es decir, las dimensiones de nuestra matriz (cuantos puntos de medida habrá), y de igual modo por cuanto están espaciados. En caso de no cumplir los limites indicados anteriormente, automáticamente aparecerá un aviso de que estos están siendo rebasados, y se pondrá el entero mayor que permita cumplir las limitaciones.

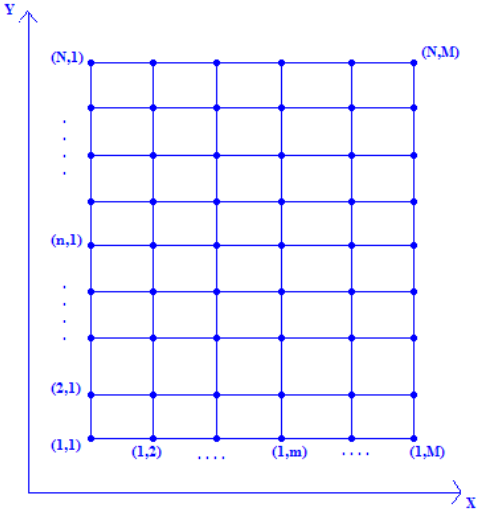


Figura 45. Aspecto de la matriz

En este caso nuestra matriz no tiene situado el punto (1,1) en la esquina superior izquierda, sino que se encuentra la inferior izquierda, sin embargo, a nivel de cálculo y programación esto es transparente, ya que lo único que tendremos que hacer es indicar que el motor se mueva en una dirección u otra.

Es importante señalar que de la manera que hemos programado la mesa, definamos la matriz que definamos, ésta siempre se encontrará centrada con respecto al centro de la mesa, de modo que el punto central de la matriz siempre coincidirá con el punto central de la mesa.

Los dos últimos cuadros de texto que aparecen en la imagen de la interfaz, son los que nos indican en qué posición de nuestra matriz nos encontramos en cada momento, ya que cada vez que la mesa cambia de posición estos valores se actualizan, de este modo podemos confirmar que el movimiento se está realizando de modo correcto.

CÁLCULO DE PARÁMETROS DE LA MATRIZ:

En nuestro programa, además de las funciones respectivas a los botones que aparecen en nuestra interfaz, también se han añadido algunas funciones de cálculo interno que faciliten la estructuración del código y agilicen su ejecución.

Éste es el caso de la función encargada de calcular los parámetros de la matriz que hemos definido con los valores explicados anteriormente. Esta función calcula los 4 parámetros básicos para saber cómo movernos: las posiciones iniciales y finales en ambos ejes.

El usuario indica los valores de espaciado en milímetros (teniendo en cuenta que la distancia máxima es de 762 mm en cada eje) y al principio de esta función se realiza la conversión a los pasos correspondientes, realizando un redondeo, ya que únicamente podemos desplazarnos valores enteros de pasos. En base a esto se obtienen los 4 parámetros indicados previamente.

$$x_{inicial} = \frac{760-(M-1)\Delta x}{2} \tag{4.2}$$

$$y_{inicial} = \frac{760-(N-1)\Delta y}{2} \tag{4.3}$$

$$x_{final} = x_{inicial} + (M-1)\Delta x \tag{4.4}$$

$$y_{final} = y_{inicial} + (N-1)\Delta y \tag{4.5}$$

Conocidos estos 4 valores ya conocemos la posición inicial (1, 1) de nuestra matriz y la posición final (N, M), de tal modo que, con los datos de espaciado facilitados también por el usuario, ya tenemos todo lo necesario para saber en qué posiciones movernos.

TIPO DE BARRIDO:

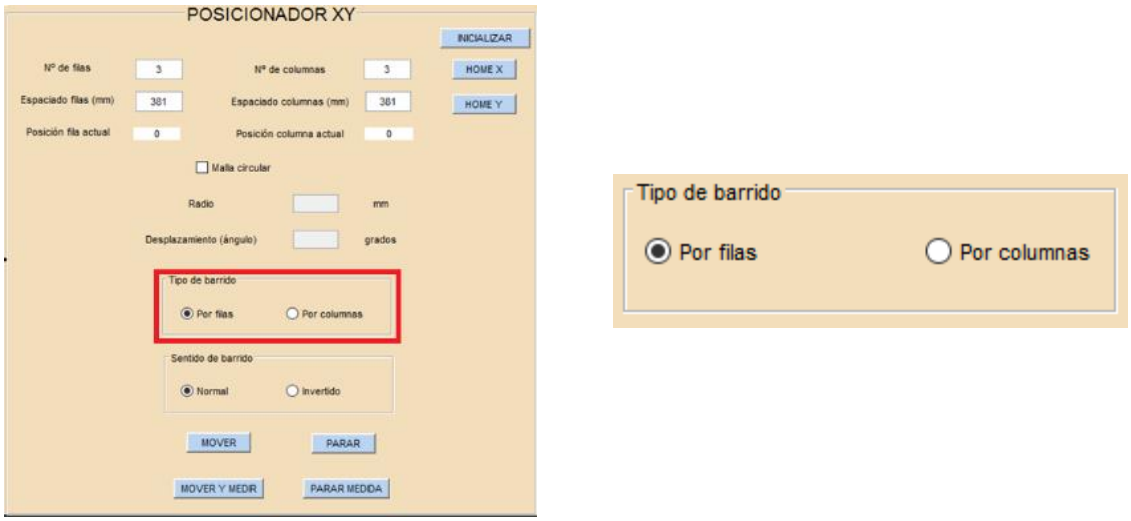


Figura 46. Tipo de barrido

Una vez definida nuestra rejilla, la siguiente opción de la que dispone nuestro programa es como queremos efectuar el barrido. Cabe la posibilidad de realizarlo por filas (eje X) o por columnas (eje Y).

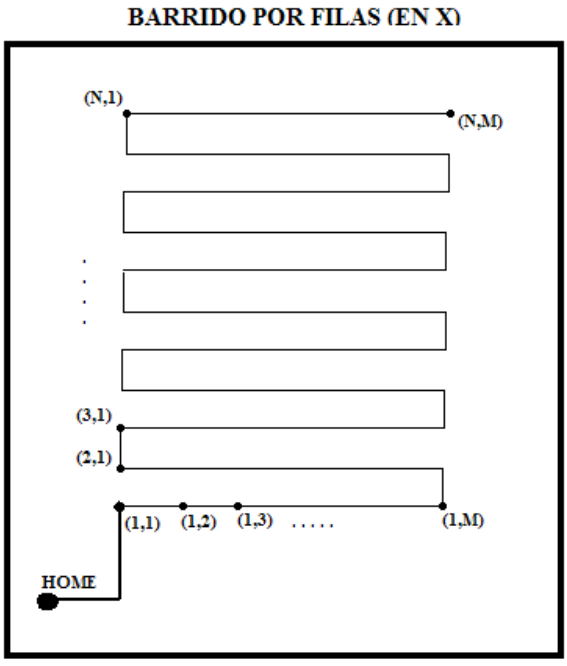


Figura 47. Barrido por filas

En este primer caso, empezamos el recorrido en el punto (1,1) y avanzamos hasta el (1, M), es decir completamos la primera fila. Posteriormente pasamos al punto (2, M) y retrocedemos en el eje X hasta llegar al (2, 1,) y de esta manera vamos avanzando hasta llegar a la última fila. En caso de que el número de filas sea impar, la posición final será la (N, M), por el contrario, si tuviésemos filas pares acabaríamos en el (N, 1).

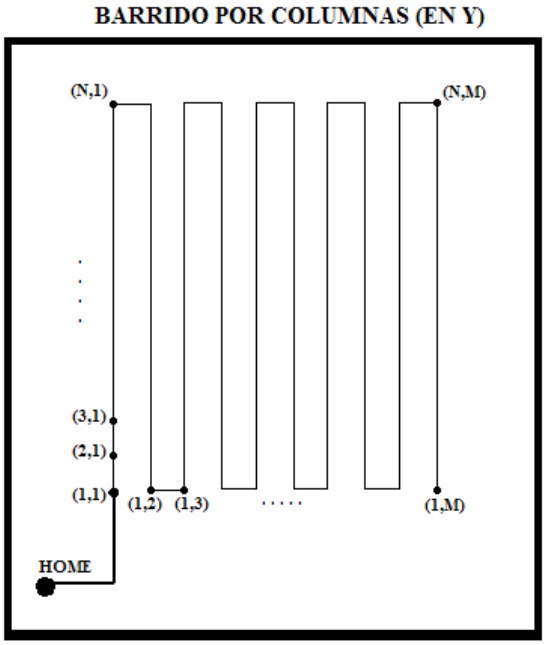


Figura 48. Barrido por columnas

Para el caso de realizar el barrido en columnas, la posición inicial continúa siendo la (1,1) pero en este caso avanzamos hasta la (N,1), completando así la primera columna. Se pasa seguidamente a la (N, 2) y se avanza hasta la (1, 2), y así sucesivamente. En este caso si el número de columnas es impar acabaremos en la posición (N, M) y en caso de ser par en (1, M).

Como vemos, ya se emplee un tipo de barrido u otro, realizamos el desplazamiento en modo de zigzag, ya que de este modo conseguimos ahorrar tiempo a la hora de realizar las medidas oportunas.

SENTIDO DEL BARRIDO:

Figura 49. Sentido del barrido

También se optó por implementar una opción a nuestro programa que, aunque no sea de vital importancia, puede resultar útil en ciertos casos. Se trata de indicar que sentido queremos que siga el barrido. Básicamente mediante esta función permitimos al usuario la posibilidad de realizar el camino inverso al realizado una vez que se ha recorrido la matriz entera, ya sea por

filas o por columnas. Por tanto, en un principio la opción de realizar el movimiento en sentido inverso se encuentra deshabilitada, pero una vez se ha completado el desplazamiento en la dirección normal, esta opción aparece como disponible por si quisiésemos volver a medir el canal en el sentido opuesto.

A nivel de programación se han empleado unos flags como indicadores del sentido de movimiento que estamos siguiendo, ya que esta es la manera más sencilla de trabajar e indicar a los motores si deben moverse en una u otra dirección.

REJILLA CIRCULAR:

Figura 50. Parámetros para moverse en círculo

Esta opción es la gran novedad en nuestro trabajo en comparativa con los anteriores en lo que respecta a la parte de la mesa. Hasta el momento siempre se había planteado el desplazamiento en rejillas rectangulares, pero a la hora de organizar nuestro trabajo se propuso añadir la funcionalidad de desplazarnos también en círculos. Se habilitan dos casillas al usuario para que introduzca los dos únicos datos necesarios, el radio de la circunferencia y cada cuantos grados queremos realizar una medida, con estos datos ya podremos realizar internamente mediante una función (explicada a continuación) la conversión al movimiento que habría que realizar tanto en el eje X como el Y.

CÁLCULO DE PARÁMETROS DE LA CIRCUNFERENCIA:

De igual modo que tenemos una función para calcular los parámetros necesarios para desplazarnos en una matriz, tenemos otra para desplazarnos en una circunferencia.

Como hemos dicho el usuario nos proporciona radio (r) y desplazamiento en grados ($\Delta\alpha$). Con esto podemos obtener un vector con todos los ángulos en los que se realizarán medidas.

Una vez más el usuario, al igual que sucedía en la rejilla, nos da el valor del radio en milímetros, por lo que realizamos la conversión a pasos con el correspondiente redondeo. (El radio máximo será de 381 mm, ya que como se ha dicho las dimensiones de la mesa son 762x762 mm).

```
angulo=angulo*2*pi/360;
alpha=0:angulo:2*pi;
```

Si tomamos como referencia la división por cuadrantes de la circunferencia que se suele hacer habitualmente, podemos saber cuál será el incremente o decremento tanto en X como en Y entre cada posición.

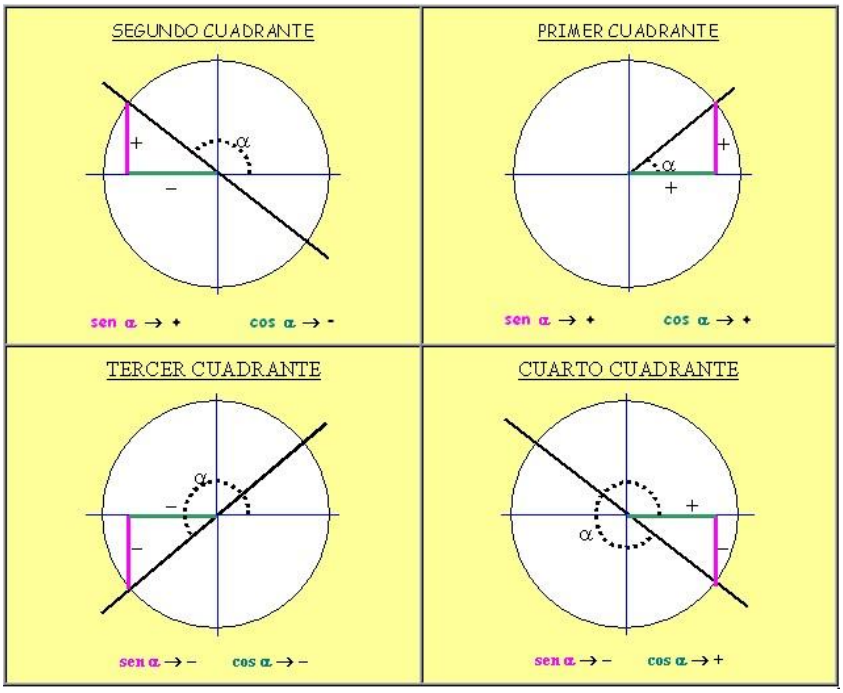


Figura 51. Cuadrantes circunferencia

$$\Delta x = r * \cos(\alpha) \tag{4.6}$$

$$\Delta y = r * \text{sen}(\alpha) \tag{4.7}$$

Conseguimos así otros dos vectores con los correspondientes valores para las variaciones en cada eje en cada uno de los movimientos realizados.

Hecho esto es preciso que tomemos algún punto de referente en la mesa para situar el centro de nuestra circunferencia. Por el simple hecho de poder realizar circunferencias en el mayor rango posible de valores para el radio, se optó por situar el centro de la misma en el centro de la mesa, es decir, en el punto (380, 380). Por lo tanto, la posición inicial a partir de la cual empezaremos a construir nuestra circunferencia será:

$$x = 380 + r * \cos(0) = 380 + r \tag{4.8}$$

$$y = 380 * \text{sen}(0) = 380 \tag{4.9}$$

De tal modo que la posición relativa en cada momento será:

$$x_i = x_{i-1} + \Delta x_i \tag{4.10}$$

$$y_i = y_{i-1} + \Delta y_i \tag{4.11}$$

INICIALIZAR:

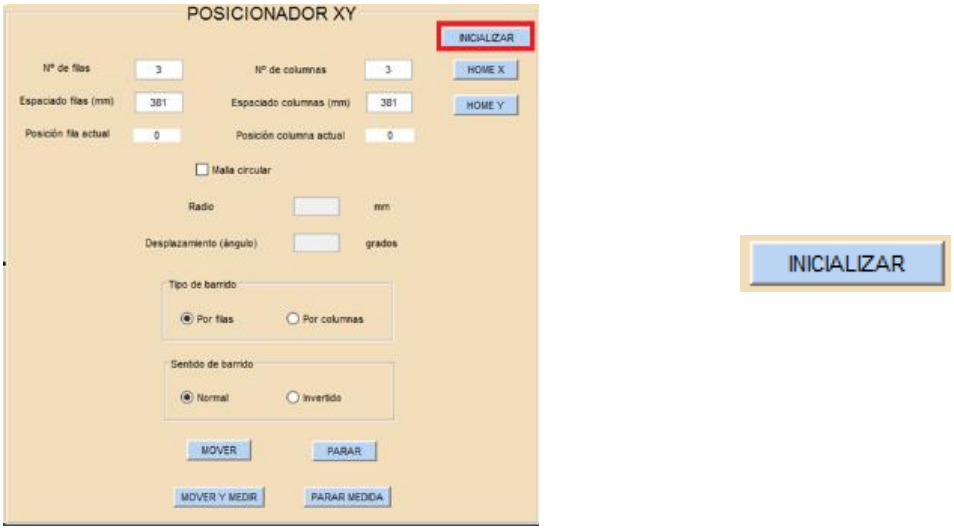


Figura 52. Botón para inicializar el sistema MD-2

Para poder mover los motores del sistema MD-2 es preciso haber ejecutado el software de Arrick Robotics para que realice la calibración. Debido a esto se ha configurado este botón, a partir del cual se hace una llamada al archivo .exe que ejecuta el software y realiza la calibración. Previamente se ejecuta una instrucción que comprueba si en el directorio existía ya un archivo con una calibración previa, y de ser así lo elimina antes de ejecutar el programa para que no haya problemas posteriores.

Una vez que el programa ya ha realizado la calibración, es necesario seleccionar la casilla *enable* del propio software, ya que sino no será posible mover los motores.

HOME:

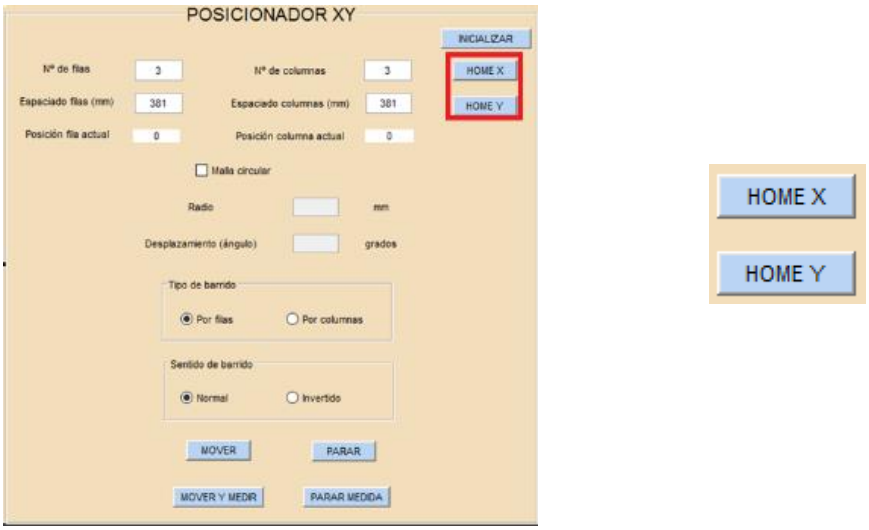


Figura 53. Botones para ir a posición inicial

Como ya se ha comentado, todos los movimientos se ejecutan tomando como referencia última la posición (0, 0) de la mesa, de manera que se consideró importante configurar los botones correspondientes que nos permitiesen volver a esta posición.

En dicha posición hay colocados en cada uno de los ejes unos interruptores, que al ser pulsados nos permiten saber que ya hemos llegado a la posición de partida. Por tanto, mediante la lectura de los pines 12 y 13 para los ejes X e Y respectivamente podemos saber cuándo se han pulsado los interruptores.

MOVER:

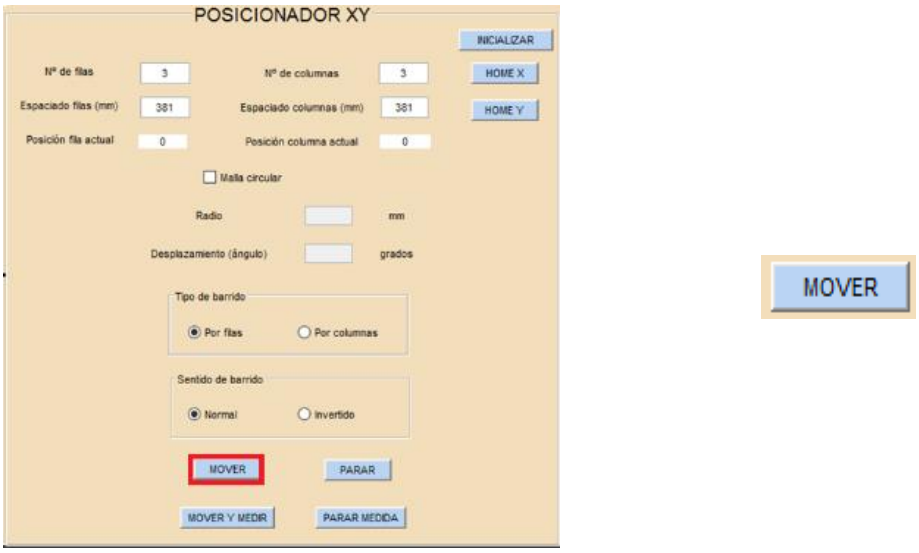


Figura 54. Botón para mover los motores en la rejilla indicada

La programación de este botón es la parte más tediosa de todos los que forman la interfaz que controla el movimiento de la mesa, ya que en él se integran todos los explicados anteriormente. Su funcionalidad se encuentra divida en diferentes pasos:

1. Se calculan los parámetros correspondientes a la rejilla con la que vayamos a trabajar: rectangular o circular.

2. Si no nos encontramos en la posición (0, 0) nos desplazamos a esta llamando a las funciones encargadas de controlar los botones *Home X y Home Y*.

3. Nos desplazamos a la posición inicial de medida con los parámetros calculados en el punto 1.

4. Nos movemos por toda la rejilla que ha sido definida por el usuario.

En el último punto, en el caso de que se trate de un desplazamiento en un matriz, hay que considerar además de lo anteriormente dicho el hecho de que nos desplacemos en filas o en columnas. Además, una vez terminada el movimiento, si seleccionamos la opción de desplazarnos en sentido inverso, al pulsar sobre el botón *Mover*, pasamos directamente al punto 4 de los anteriores, despreciando los 3 primeros. Como explicamos en *Sentido de Barrido*, mediante el uso de flags podemos saber en qué situación nos encontramos y así ejecutar el paso que corresponda.

Si estamos realizando un movimiento circular la programación del movimiento se simplifica bastante, ya que como hemos dicho antes, simplemente multiplicando por seno y coseno ya tenemos prácticamente todo el trabajo hecho.

PARAR:

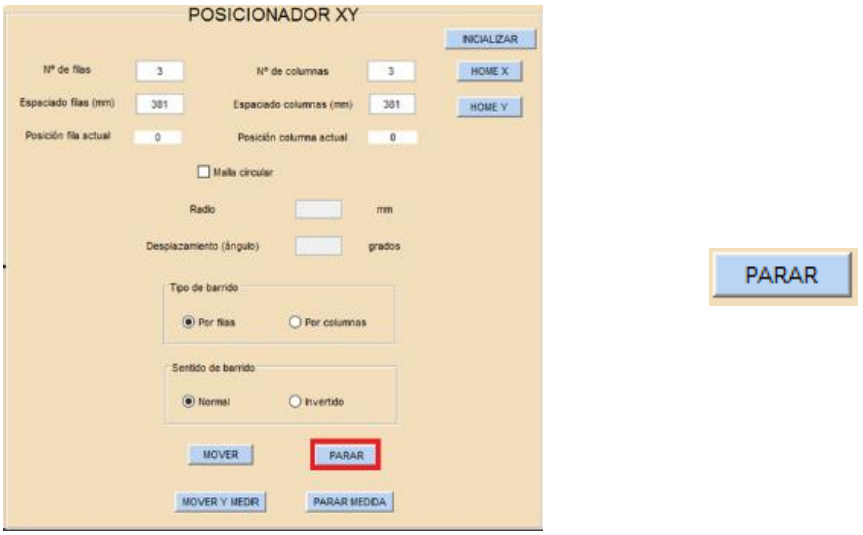


Figura 55. Botón que detiene los motores

Desde la función encargada de controlar este botón lo que se hace es simplemente enviar los comandos apropiados para parar el movimiento de los motores.

```
dio=digitalio('parallel','LPT1');
data1=addline(dio,0:7,0,'out');
putvalue(data1,255);
```

Desde la función encargada de controlar el botón de Mover se realizan comprobaciones en cada paso por los bucles que la componen y en caso de haber pulsado este botón se hace la llamada la función y se para el movimiento.

4.4.4 Anexionado de ambas interfaces

En las Figuras 56-60 se muestra el detalle de los botones que controlar la sonda completa y mensajes de información para el usuario.

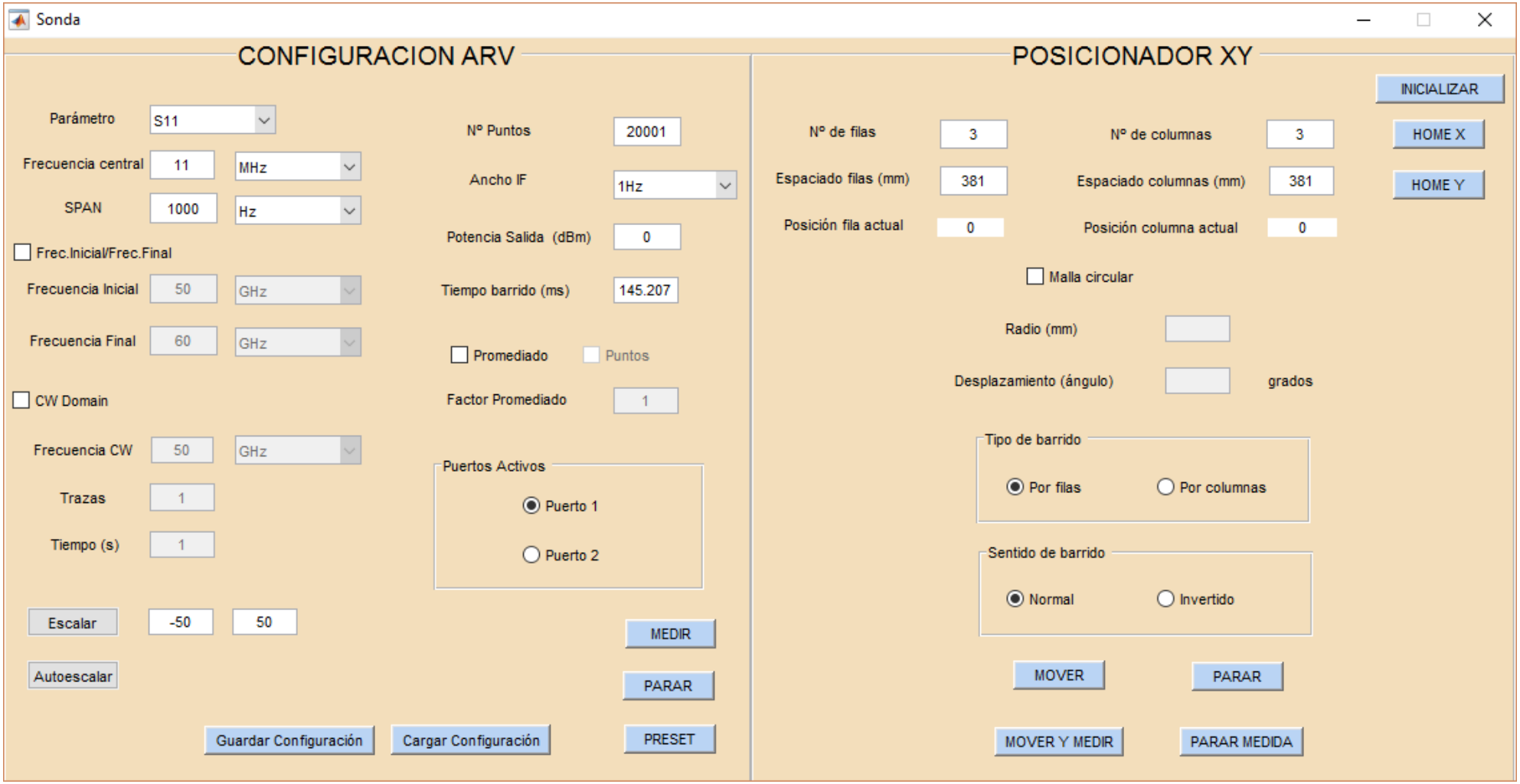


Figura 56. Interfaz gráfica del programa completo

Finalmente se procede a unir las dos interfaces en una sola para tener así un control total del sistema de medida. Casi la totalidad del código de ambas se ha respetado de manera integra a la hora de juntarlo, sin embargo, se han realizado algunas modificaciones necesarias en ciertas funciones para que no existan posibles confrontaciones entre determinadas funciones.

En esta parte únicamente se han añadido dos nuevos botones a todos los explicados con anterioridad, siendo estos los más relevantes de todos.

MOVER Y MEDIR:

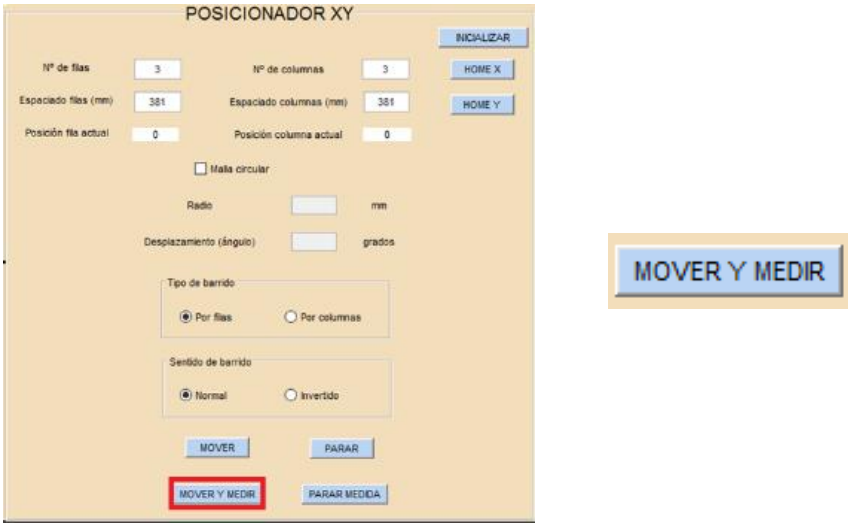


Figura 57. Botón encargado de mover el posicionador y realizar las medidas indicadas

Se trata del botón encargado de controlar todo: movimiento de la mesa, parámetros del analizador, medidas a realizar, donde y como guardarlas. En consecuencia, la función que lo controla es la más extensa y compleja de todo el código que compone el programa.

El código se compone en gran parte del mismo código que conformaba el botón de *Mover*, solo que en este caso cada vez que la mesa se desplaza y llega a uno de los puntos de medida, se para y en se momento se llama a la función *Medir*.

Dicha función a su vez ha sido modificada, ya que se han añadido ciertas condiciones para discriminar si estamos o no empleando la mesa, ya que usamos el mismo código tanto con la mesa cómo sin ella.

La mayor diferencia en la función *Medir* reside a la hora de establecer el nombre con el que queremos que se guarden los ficheros. En el caso de hacerlo sin mesa se realiza una concatenación de strings del siguiente modo:

NombreIntroducidoPorUsuario_ParámetroMedido_Fecha_Hora

En el caso de emplear la mesa de medidas, este modelo de nombre de fichero no nos es útil, ya que necesitamos saber en qué posición estamos midiendo. Por tanto, en el caso de que estemos midiendo en forma de matriz, el nombre quedaría como sigue:

NombreIntroducidoPorUsuario_ParámetroMedido_Fecha_NxM

Donde *NxM* indica en qué posición de la matriz se ha realizado dicha medida. El valor de N y M lo podemos leer de las dos casillas de texto que tenemos en nuestro programa, las cuales se van actualizando cada vez que se cambia de posición.

Por otro lado, si estuviésemos realizando un barrido circular la información que nos interesaría conocer es con que ángulo se ha realizado cada medida.

NombreIntroducidoPorUsuario_ParámetroMedido_Fecha_Angulo

Donde el ángulo lo conocemos gracias al vector que tenemos calculado con todos los ángulos de la circunferencia en los que se van a realizar medidas.

NOTA: *En todos los casos, si la medida que se está realizando es en onda continua (CW) al final del nombre se añadirá un subíndice que nos indique el número de traza que se está midiendo.*

PARAR MEDIDA:

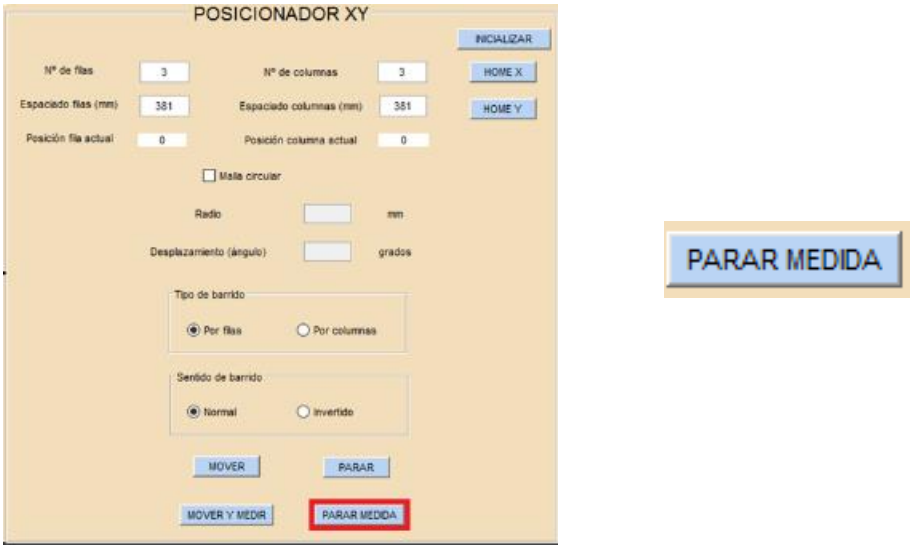


Figura 58. Botón encargado de parar el movimiento de los motores y las medidas que se estén realizando

Este botón en sí es una implementación de otros dos botones. Desde él se realiza la llamada en primer lugar a la función encargada de parar la medida y seguidamente a la que para los motores, de tal modo que al ser el único botón que dejamos habilitado una vez que pulsamos el de *Mover y Medir*, cuando lo presionamos interrumpe inmediatamente la ejecución de éste y para el proceso de medida.

MENSAJES DE INFORMACIÓN:

Por último, se han añadido dos mensajes emergentes a nivel informativo para el usuario. El primero de ellos se muestra por pantalla al inicializar el programa y le da al usuario ciertas pautas a seguir para la correcta utilización del mismo.

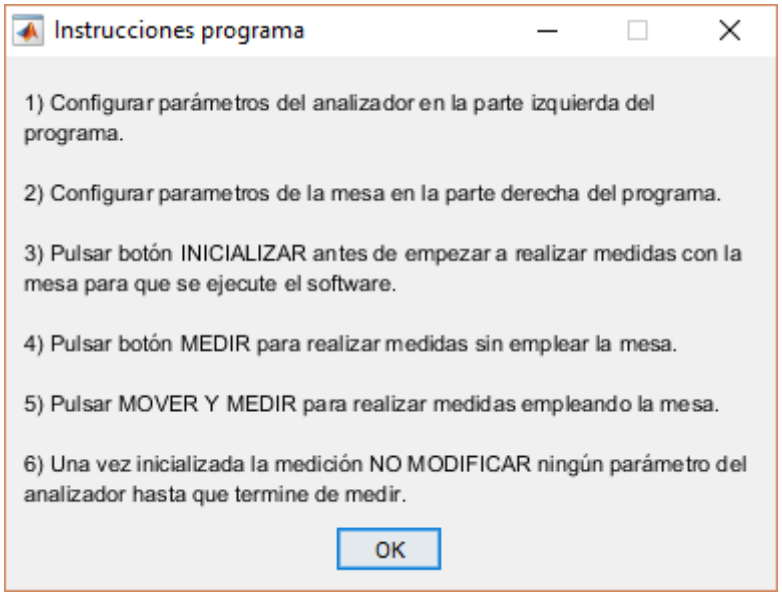


Figura 59. Instrucciones inicio programa

El otro mensaje que se emite es una ayuda a la hora de ejecutar el botón *Inicializar*, en el cual se le indica al usuario que se ejecutará el software de Arrick Robotics y ciertas pautas a seguir.

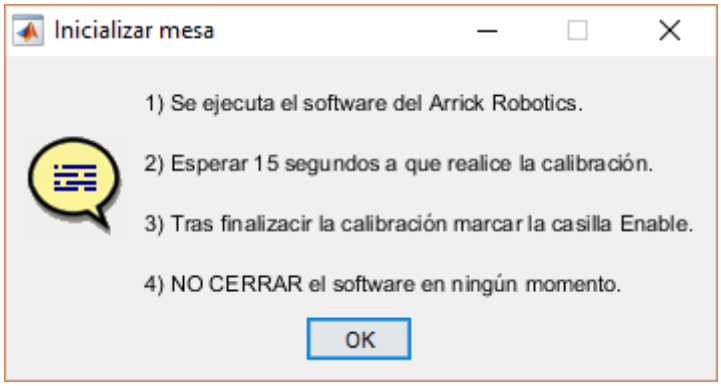


Figura 60. Mensaje de ayuda al pulsar *Inicializar*

4.4.5 Problemas y soluciones

A lo largo del desarrollo de toda la interfaz encargada de controlar nuestro sistema de medidas han aparecido múltiples inconvenientes, los cuales se han podido solventar.

Analizador:

El primer y principal problema que encontramos fue el de la limitación de instrucciones que nos ofrecía la programación COM. Como ya se ha explicado, las instrucciones que se envían al analizador se ejecutaban a través de un objeto que creábamos a partir del driver que el fabricante nos facilitaba. El problema residía en que este driver no había sido específicamente diseñado para trabajar en Matlab, si no que estaba pensado para emplearlo en *Visual Basic o Visual C++*. Es por esto que el número de funciones que nos facilitaba para trabajar desde Matlab se encontraba muy reducido.

Sirva de ejemplo que mediante la programación COM no es posible desde Matlab emplear un comando que nos permita cambiar el parámetro que estamos midiendo, o modificar la potencia de salida. Del mismo modo algunas otras funciones, aunque si estaban disponibles, como el control de *Trigger*, no nos permitían emplear todas sus funcionalidades, sino que solo podíamos usar algunas de ellas.

Cabían varias soluciones a la hora de resolver este problema. La más complicada, aunque seguramente la más óptima en cuanto a nivel de depuración y optimización en la programación, sería editar el driver e incorporarle esas funciones extra de las que no disponía. Por otro lado, la opción más sencilla consistía en emplear programación mediante comandos SCPI en todas aquellas funciones en las que COM no estuviese disponible.

Por cuestiones de tiempo y de no complicar en exceso el trabajo (ya que sería salirse del objetivo original del mismo) se optó por la solución sencilla, emplear comandos SCPI.

Tal y como explicamos en apartados anteriores, SCPI está estandarizado y esto hace que la lista de comandos disponibles sea completa y sean comunes empleemos el entorno de programación que empleemos. Para trabajar con estos comandos es preciso la creación de otro objeto independiente del que teníamos creado con el driver. Se creó un objeto VISA (Visual Instrument Software Architecture) sobre protocolo TCP/IP, a través del cual se envían las instrucciones al analizador.

```
vendor='agilent';

rsrcname='TCPIP0::A-N5227A-70442.local::hislip0::INSTR';

handles.objvisa=visa(vendor,rsrcname);
```

Para crear este objeto únicamente tenemos que facilitarle a Matlab el fabricante y el I/O name de nuestro analizador. Creado el objeto, la instrucción empleada para enviar órdenes es *Fprintf* y para obtenerlas *Fscanf()*.

```
fprintf(handles.objvisa,'COMANDO SCPI');

fscanf(handles.objvisa);
```

Con esto conseguimos solventar el problema de ausencia de comandos en COM, lo cual nos permitía poder trabajar ya libremente con cualquier función del analizador.

Otro problema importante que nos surgió fue a la hora de guardar las mediciones de las trazas en onda continua. La función *Trigger* del analizador puede estar en varios estados.

1. Hold: La medición se queda capturada en la pantalla y el analizador no dispara ningún pulso.

2. Single: El analizador dispara un único pulso e inmediatamente después pasa a estado Hold.

3. Group: El analizador dispara un número X de pulsos que le indiquemos y después pasa a estado Hold.

4. Continuous: El analizador está constantemente enviando pulsos.

La idea era pues, dado un número X de trazas que queremos medir, realizar un número X de disparos. Si le damos esta instrucción al analizador, efectivamente realiza los X disparos indicados, pero a la hora de guardar tan solo almacena los datos correspondientes a la traza generada con el último y nosotros buscamos que se guarden todas las trazas que se generan.

La única solución que encontramos para este inconveniente fue enviar los pulsos de uno en uno (estado Single), encerrados dentro de un bucle for con tantas iteraciones como trazas queramos y por cada una de estas iteraciones realizar un volcado de datos en el analizador. Como en este caso se crean múltiples ficheros con la misma medida en breves instantes de tiempo, a la hora de proporcionarle un nombre se hace uso de la variable encargada de llevar la cuenta de iteraciones para diferenciar así los ficheros.

Ejemplo:

```
y=sprintf('MMEM:STOR:DATA "D:/Medidas/Santi/Matlab/CW/%s.csv","CSV
Formatted DATA","Trace","DB",%d',s1,param);

fprintf(handles.objvisa,y);
```

Otra opción sería establecer como nombre de los ficheros la hora y fecha exacta en que se ha realizado la medición, sin embargo, al poder tratarse de mediciones de milisegundos, muchos archivos estarían guardados en el mismo segundo lo cual acabaría provocando una sobreescritura de datos.

Siguiendo con el proceso de guardado nos apareció otro problema. En este caso era al querer guardar medidas promediadas en frecuencia, ya fuese con promedio en barrido o punto a punto. Había que determinar el tiempo de espera para que acabase de realizarse el promediado antes de que Matlab ejecutase la instrucción para guardar los datos.

En el caso de promediar por barrido el problema surge porque cuando medimos, aunque tengamos determinado un tiempo de barrido, el analizador necesita un tiempo para realizar el cálculo de promediado y desde Matlab no hay forma de saber el tiempo interno que tarda el analizador en efectuar estos cálculos. Matlab implementa algunas funciones como *Pause()* mediante la cual podemos indicarle que espere un determinado tiempo a ejecutar la siguiente instrucción, pero en este caso al desconocer el tiempo del analizador no nos es de utilidad.

De igual forma el analizador permite una solución que devuelve un valor de bit determinado por traza que nos indica si se encuentra o no promediando, sin embargo, la combinación de valores resultaba demasiado alta como para controlarla sin complicar en exceso la programación. Por tanto, como se ha comentado en el apartado 4.4.2, la solución más sencilla que se encontró para esto consistía en una vez determinado el factor de promediado que queríamos, efectuar exactamente ese mismo número de pulsos con el *Trigger*, de este modo cuando se enviasen todos los pulsos el analizar pasaría a estado *Hold* y así sería fácil saber en qué momento podemos proceder a guardar nuestra medida.

```
fprintf(handles.objvisa,':SENS:SWE:MODE GRO');

fprintf(handles.objvisa,'SENS:SWE:MODE?'); %preguntamos al  analizador
en qué estado está

s2=fscanf(handles.objvisa);
z=strcmp(s1,s2);

while  (z==0)
    fprintf(handles.objvisa,'SENS:SWE:MODE?');
    s2=fscanf(handles.objvisa);
    z=strcmp(s1,s2);
end
```

En s1 tenemos almacenado la palabra 'HOLD', de modo que cuando hagamos la lectura con la instrucción *Fscanf()* y la palabra obtenida en s2 sea también 'HOLD', se saldrá del bucle.

En la otra situación, punto a punto, el problema era el mismo pero la solución no. En este caso el factor de promediado nos determinaba el tiempo de barrido, por tanto, la solución era similar al proceso aplicado para guardar en onda continua: conocemos el tiempo de barrido, por tanto, le indicamos a Matlab que espere un determinado tiempo antes de guardar para dar tiempo al analizador de completar la medida.

Finalmente, para concluir con los problemas respectivos al funcionamiento del analizador, nos apareció uno en lo respectivo a modificar el parámetro que queríamos medir y posibles fallos que esto acarreaba en el tiempo de barrido.

El analizador no contiene ninguna instrucción que nos permita cambiar o seleccionar directamente que parámetro es el que queremos que se muestre por pantalla y medir. Por el contrario, lo que tenemos que hacer es crear una medida, a la que daremos el nombre que queramos y que se corresponda con el parámetro que queremos medir en cada caso. En nuestro caso se han habilitado hasta 12 parámetros para poder ser medidos, esto conllevaba crear en un inicio 12 medidas diferentes, de modo que así podríamos seleccionar cuál de ellas es la que se mostraría por pantalla y mediríamos. Sin embargo, el hecho de crear estas medidas suponía que, aunque no las estuviésemos usando ni mostrando por pantalla, el analizador internamente las estaba considerando, por tanto, al tener medidas que utilizan unas un puerto como transmisor y las otras el otro (aunque los estableciésemos como desactivados) el analizador realizaba un doble barrido, empleando el doble del tiempo del necesario, lo cual no tenía ningún sentido.

Para solucionar este inconveniente la única opción que encontramos fue el crear únicamente una medida, en lugar de las 12. Por tanto, lo que se hizo fue cada vez que se cambiaba de parámetro llamar a una función que eliminase todas las medidas existentes en el canal e inmediatamente después crease una única correspondiente al parámetro concreto que estuviésemos seleccionando para medir.

Posicionador:

En el caso del sistema de Arrick Robotics partíamos de la base de trabajos anteriores, en los cuales estaba claramente explicado cómo hacer para mandarle instrucciones para mover los motores, por tanto, en ese sentido no ha aparecido ningún tipo de problema.

En el caso de la funcionalidad nueva para describir un movimiento circular, en un principio surgió la duda de cómo realizar un movimiento circular ya que los motores únicamente nos permitían desplazarnos en los ejes X e Y una determinada cantidad de pasos. Sin embargo, se optó por realizar el cálculo de dos vectores en los que estarían contenidos las posiciones exactas en cada eje para cada posición de medida y una vez conocidos estos vectores se podría calcular la diferencia en pasos entre una posición y la siguiente, de modo que ya sabríamos exactamente cuánto tendría que moverse cada motor, bien en una dirección u en la otra, lo cual nos vendrá delimitado precisamente por el signo que tengan esos valores de diferencias calculados a partir de los vectores de posición.

Sonda:

A la hora de unir las dos interfaces, surgió un único problema y no resultó ser algo difícil de solventar. Cuando pulsamos el botón *Mover y Medir* cada vez que se nos movemos a una

posición, se realiza una llamada a la función *Medir* para que el analizador realice las medidas pertinentes y guarde los datos en el PC.

Comprobamos que, a la hora de ejecutar la instrucción de guardar, como estábamos guardando archivos en múltiples posiciones diferentes, pero como el mismo nombre se producían sobreescritura de datos. Por tanto, se buscó una notación específica para cada caso de modo que no tuviésemos este inconveniente, de modo que se optó por seguir el modelo explicado en el apartado 4.4.4 cuando hablamos del botón *Mover y Medir*. Mediante una variable que utilizamos como flag resulta fácil indicar si estamos realizando las medidas empleando o no la mesa, de tal manera que así el programa sabe si tiene que guardar con una u otra notación.

Capítulo 5. Conclusión y líneas de trabajo futuras

5.1 Conclusión

En este TFG se ha diseñado e implementado una sonda de canal basada en el dominio de la frecuencia. La sonda está formada por un ARV que mide el parámetro de dispersión S_{21}, un sistema de posicionamiento XY que nos permite desplazar la antena receptora para caracterizar el PDP medio a partir de medidas en una zona pequeña (en términos de longitud de onda) y un PC desde el que se controla remotamente mediante Matlab los elementos anteriores, permitiendo la automatización del proceso de medidas, que se guardan en el disco duro del ARV por cuestión de tiempos.

Se ha desarrollado una interfaz gráfica basada en el entorno de programación GUIDE de Matlab. Que permite un fácil manejo de los dispositivos. Ésta permite medir con el ARV y el posicionador XY, pero también permite medir únicamente haciendo uso del ARV, como por ejemplo en aplicaciones de onda continua.

La comunicación con el ARV se realiza vía LAN con programaciones COM y SCPI. La mesa XY utiliza un controlador basado en el puerto LPT1, lo que obliga a incorporar un driver para su control vía Matlab.

El fabricante del ARV provee de un driver para su control desde Matlab empleando programación COM. Sin embargo, el control no es posible en todas sus funciones empleando este driver, por lo que se ha tenido que emplear programación SCPI mediante protocolo TCP/IP para completar las funciones necesarias.

De igual modo, aunque el trabajo inicial estaba concebido para medir la transmisión en el canal radio en interiores a partir del estudio del parámetro S_{21}, se ha ido más allá de este objetivo y se ha implementado la posibilidad de medir otros parámetros, completando así una sonda de medida más completa.

Por último, en el diseño del control del posicionador. Considerando trabajos anteriores, en los cuales nos podíamos desplazar describiendo movimientos en forma de matriz, se ha optado por añadir también la posibilidad de movernos en movimientos circulares ya que, en determinados tipos de medidas, como puede ser observar las contribuciones obtenidas desde los distintos ángulos de llegada.

5.2 Líneas de trabajo futuras

A lo largo del desarrollo del trabajo, han ido surgiendo ideas que podrían ser ejecutadas como futuros trabajos. Estas surgen más como posibles mejoras que como continuación de lo ya hecho, pero que, aun siendo mejoras, son los suficientemente complejas como para formar por si solas lo que sería un trabajo completo y extenso.

En primer lugar, planteamos la posibilidad de desarrollar un driver para sustituir al que hemos empleado para la parte de programación COM en nuestro trabajo. Como ya hemos dicho el driver que nos facilita el fabricante se encuentra muy limitado en las funcionalidades que permite implementar, por lo que resultaría muy interesante desarrollar un driver desde el cual se tuviesen todas las mismas funcionalidades que poseemos al realizar la programación mediante comandos SCPI, ya que de este modo nos podríamos ahorrar este tipo de programación y limitarnos a emplear la COM, que como se ha explicado, es más eficiente en cuanto al tiempo de ejecución de instrucciones.

También planteamos la posibilidad de desarrollar una sonda similar a ésta empleando otros posicionadores que permitan establecer la conexión por otro método: USB, LAN… comprobando si es más conveniente un método u otro.

Bibliografía

[1] L. Rubio, J. Reig, H. Fernández, V.M. Rodrigo, "Experimantal UWB Propagation Channel Path Loss and Time-Dispersion Characterization in a Laboratory Environment". 2013 (Online)

http://www.hindawi.com/journals/ijap/2013/350167/

[2] Lorenzo Rubio Arjona, "Transparencias asignaturas Radiocomunicaciones, Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación, Universidad Politécnica de Valencia".2015

[3] J. José Murillo Fuentes, "Fundamentos de Radiación y Radiocomunicación, 2007 (Online)

http://personal.us.es/murillo/docente/radio/documentos/Temas1a5.pdf

[4] José María Hernando Rábanos "Transmisión por radio",Editorial centro de estudios Ramón Areces S.A. 1995

[5] Robert E. Collin "Foundations for Microwave Engineering", McGraw-Hill International Editions. 199

[6] L. Rubio, J. Reig, H. Hernandez, "Propagation Aspects in Vehicular Networks, Vehicular Technologies", Miguel Almeida(Ed.), InTech, 2011

[7] Keysight Technologies: PNA Series Network Analyzers, Help (Online)

http://na.support.keysight.com/pna/help/latest/Programming/Learning_about_COM/COM_versus_SCPI.htm

[8] D.Orlando Barragán Guerrero, "Manual de Interfaz Gráfica de Usuario en Matlab Parte 1". 2008

[9] Keysight Technologies: PNA Series Network Analyzers, Help (Online)

http://na.support.keysight.com/pna/help/WebHelp9_9/help.htm

[10] F.L. Peñaranda, M. Baquero Escudero, V.M. Rodrigo, V.E. Boria, "*Apuntes asignatura Microondas, Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación"*.2015

[11] Keysight Technologies: COM vs SCPI, Mayo 2016 (Online)

http://na.support.keysight.com/pna/help/latest/Programming/Learning_about_COM/COM_versus_SCPI.htm#lang

[12] Matías Mateu, *"Caracterización del canal radio, Propagación en Entornos Urbanos"* (Online)

https://eva.fing.edu.uy/pluginfile.php/63614/mod_resource/content/1/Canal_Radio_parte1.pdf