UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

DISEÑO Y AUTOMATIZACIÓN DE UNA SONDA DE MEDIDA DEL CANAL RADIO EN EL DOMINIO DE LA FRECUENCIA

Cristina Català Lahoz

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 2019-20

Valencia, 03 de julio de 2020

Agradezco a mis dos tutores, Lorenzo y Vicent Miquel,
por ayudarme y guiarme durante todo el proceso de la realización de este proyecto.

Gracias a mi familia y amigos
por todo su apoyo moral durante estos cuatro años.

Resumen

En este Trabajo Fin de Grado (TFG) se ha diseñado e indicado el procedimiento de implementación de una sonda de canal para la caracterización del canal de radiocomunicaciones en interiores. La sonda está formada por un analizador de redes vectorial y dos sistemas de posicionamiento de antenas. Para su diseño se ha empleado *App Designer* de Matlab, un entorno de desarrollo para diseñar y programar aplicaciones.
Esta aplicación permite caracterizar el comportamiento selectivo en frecuencia del canal a partir del parámetro S_{21} de *scattering*, además de poder medir los demás parámetros de dispersión y los parámetros de potencia absoluta disponibles en el analizador. También permite controlar dos posicionadores, uno rectangular y otro lineal. El programa da opción a elegir las distintas configuraciones de transmisión y recepción, para realizar medidas SISO, SIMO, MISO y MIMO. Por último, permite seleccionar los parámetros que marcarán el barrido de las antenas a lo largo de los posicionadores.

Resum

En aquest Treball Fi de Grau (TFG) s'ha dissenyat i indicat el procediment d'implementació d'una sonda de canal per a la caracterització del canal de radiocomunicacions en interiors. La sonda està formada per un analitzador de xarxes vectorial i dos sistemes de posicionament d'antenes. Per al seu disseny s'ha emprat *App Designer* de Matlab, un entorn de desenvolupament per a dissenyar i programar aplicacions.
Aquesta aplicació permet caracteritzar el comportament selectiu en freqüència a partir del paràmetre S_{21} de *scattering*, a més de poder mesurar els altres paràmetres de dispersió y els paràmetres de potència absoluta disponibles a l'analitzador. També permet controlar dos posicionadors, un rectangular y un altre lineal. El programa dóna opció a triar les diferents configuracions de transmissió i recepció, per a realitzar mesures SISO, SIMO, MISO i MIMO. Finalment, permet seleccionar els paràmetres que marcaran el escombrat de les antenes al llarg dels posicionadors.

Abstract

In this Final Degree Project (TFG), the implementation procedure of a channel sounder for the characterization of the indoor radio communication channel has been designed and indicated. The sounder is made up of a vector network analyzer and two antenna positioning systems. For its design has been used *App Designer* from Matlab, a development environment for designing and programming applications.
This application allows us to characterize the selective frequency behavior of the channel from the parameter S_{21} of scattering, in addition to being able to measure the other dispersion parameters and the absolute power parameters available in the analyzer. It also allows controlling two positioners, one rectangular and other linear. The program gives the option to choose the different transmission and reception configurations, to perform SISO, SIMO, MISO and MIMO measurements. Finally, it allows selecting the parameters that will mark the sweep of the antennas along the positioners.

Índice general

Índice de figuras

Índice de tablas

Acrónimos

2D Two Dimensions. 33

3D Three Dimensions. 33

5G Fifth Generation. 1, 20

ARV Analizador de Redes Vectorial. 2, 6, 7, 43, 46, 52, 54, 68, 77

ASCII American Standard Code for Information Interchange. 6

CLI Command-Line Interface. 21

COM Component Object Model. 6, 48

CW Continuous Wave. 63

DDSF Doppler Delay Spread Function. 11, 12

DFT Discrete Fourier Transform. 11–13

ESPRIT Estimation of Signal Parameters via Rotation Invariance Techniques. 20

GPS Global Positioning System. 9

GRE Grupo de Radiación Electromagnética. 1

GTK+ Gimp Toolkit. 21

GUI Grafical User Interface. 21, 22

GUIDE Grafical User Interface Development Environment. 2, 4, 22–24

HTML HyperText Markup Language. 24

IDE Integrated Drive Electronics. 22

IDSF Input Delay Spread Function. 11, 12

IEEE Institute of Electrical and Electronics Engineers. 6

IF Intermediate Frequency. 7, 43, 60, 61

IFFT Inverse Fast Fourier Transform. 20

IFT Inverse Fourier Transform. 11–13

IP Internet Protocol. 5, 45–47

ISI Intersymbol Interference. 11

iTEAM Instituto de Telecomunicaciones y Aplicaciones Multimedia. 1

IVA Impuesto al Valor Agregado. 76

LabVIEW Laboratory Virtual Instrument Engineering Workbench. 21

LAN Local Area Network. 45, 46

MATLAB MATrix LABoratory. 2, 4–6, 19–22, 24, 30, 39–41, 45, 46, 55, 65, 77

MIMO Multiple Input Multiple Output. 2, 3, 7, 20, 43, 66, 74, 75, 77

MISO Multiple Input Single Output. 3, 7, 43, 66, 72–74

MUSIC MUltiple SIgnal Classification. 20

NF Noise Floor. 14, 60, 61

NUI Natural User Interface. 21

ODSF Output Doppler Spread Function. 11, 12

PDP Power Delay Profile. 1, 4, 6, 13–17, 19, 77

PHP Hypertext Preprocessor. 21

RADAR RAdio Detecting And Raging. 9

RGB Red, Green, Blue. 37, 39

RiMAX RItcher MAXimum likelihood. 20

RMS Root Mean Square. 14

RS-232 Recommended Standard 232. 4, 47

SAGE Space-Alternating Generalized Expectation-maximization. 20

SCPI Standard Commands for Programmable Instruments. 6, 7, 77

SIMO Single Input Multiple Output. 3, 7, 43, 66, 68–70, 72, 73

SISO Single Input Single Output. 3, 7, 43, 66–68

SNA Scalar Network Analyzer. 3

TCP Transmission Control Protocol. 5, 45–47

TFG Trabajo Fin de Grado. 1–4, 77

TFM Trabajo Fin de Máster. 77

TVTF Time Variable Transfer Function. 11, 12

UI User Interface. 6, 29, 33

UPV Universitat Politècnica de València. 1

US Uncorrelated scattering. 12, 13

USB Universal Serial Bus. 3, 4, 47, 48, 63

VISA Visual Instrument Software Architecture. 5, 46, 47

VNA Vector Network Analyzer. 3

WSS Wide Sense Stationary. 12, 13

WSSUS Wide Sense Stationary Uncorrelated Scattering. 6, 12, 13

ZUI Zooming User Interface. 21

Capítulo 1

Introducción y objetivos

1.1. Introducción

Los sistemas de comunicaciones móviles e inalámbricas utilizan el canal radio para el intercambio de información entre dos puntos. El canal radio puede ser un medio hostil que impida la correcta comunicación en un sistema de comunicaciones. Tanto en un medio exterior como en un medio interior, existen obstáculos con los que se encontrará la onda que transporta la información y que afectan a la propagación de ésta. Además de que se trata de un canal variante en el tiempo, y la distancia entre transmisor y receptor también varía con éste. Todo esto causa que al receptor lleguen distintas contribuciones de la señal en distintos intervalos de tiempo y con fase distinta, suceso conocido como 'efecto multicamino'. Así como el 'efecto Doppler' producido por el movimiento relativo del transmisor respecto al receptor o *scatterers* (elementos en el canal), que afecta a la frecuencia de la onda.

Por todo lo anterior, es importante caracterizar el canal radio para un diseño óptimo del sistema a implementar. Mediante la función de transferencia del canal $T(f,t)$ podremos ver como se comportará el sistema de comunicaciones cuando se transmita a través de él. Con un Analizador de Redes podemos medir la respuesta en frecuencia del canal y transformarla para obtener el perfil de retardo (PDP, *Power Delay Profile*) que mostrará las contribuciones multicamino recibidas en cada instante.

1.2. Objetivos

El objetivo de este TFG es el diseño e implementación de una sonda de canal que permita medir el comportamiento selectivo en frecuencia del canal, empleando el equipamiento disponible en el laboratorio del GRE (Grupo de Radiación Electromagnética) del iTEAM (Instituto de Telecomunicaciones y Aplicaciones Multimedia) de la UPV (Universitat Politècnica de València). Esta sonda se utilizará posteriormente para estudiar el canal en la banda milimétrica para futuras comunicaciones 5G.

Este trabajo amplía y mejora la sonda previa existente [1] y se diseña y programa en un entorno diferente, donde se implementan nuevas e interesantes funcionalidades. La sonda controla un Analizador de Redes Vectorial y dos posicionadores de antenas. El diseño de la interfaz gráfica se

realiza desde un ordenador empleando App Designer de MATLAB.

El programa diseñado introduce la funcionalidad de poder realizar medidas MIMO (*Multiple Input Multiple Output*), gracias a los dos posicionadores de antenas. También permite utilizar el ARV de forma individual (sin hacer uso de los posicionadores), ya que no solo está preparado para medir el parámetro S_{21}, si no que se pueden medir los demás parámetros disponibles en el analizador.

Más adelante se explican las funcionalidades del entorno de programación utilizado, además del funcionamiento de la sonda diseñada y el procedimiento seguido para su programación y automatización.

1.3. Organización del TFG

En el Capítulo 2 se incluye una descripción del entorno de trabajo, los equipos empleados, la configuración de la sonda y el tipo de programación utilizado para la programación del interfaz controlador. También se añade un apartado con la distribución de las tareas que se han seguido para realizar el proyecto.

En el Capítulo 3 se abordan los aspectos teóricos básicos relacionados con los parámetros que se utilizan normalmente a la hora de caracterizar el canal en un escenario de interior, además de una serie técnicas de medida y equipamiento utilizados.

En el Capítulo 4 se ve una explicación del funcionamiento de la herramienta *App Designer* de MATLAB utilizada para realizar la interfaz gráfica que controla los distintos instrumentos. También se exponen algunas diferencias con la antigua herramienta de MATLAB, GUIDE.

En el Caítulo 5 se incluye todo el desarrollo del trabajo, desde la conexión con los instrumentos hasta la programación y el diseño de la interfaz y la descripción de sus componentes.

En el Capítulo 6 se incluye el pliego de condiciones, con un despliegue del presupuesto total para la realización del proyecto.

Por último, en el Capítulo 7 se encuentran las conclusiones de este Trabajo Fin de Grado y las propuestas de trabajo futuro.

Capítulo 2

Metodología

2.1. Gestión del proyecto

En esta sección se indican los distintos materiales y equipos empleados para la gestión de este proyecto. Respecto a los años anteriores se ha añadido un nuevo posicionador lineal (X), además del posicionador rectangular (XY) del que ya se disponía. También se ha añadido un nuevo complemento al controlador de estos posicionadores, que nos permite controlarlos con un cable USB (tipo A) desde cualquier ordenador con esta conexión. Finalmente, se explican los distintos tipos de programación que permite el analizador de redes empleado y cual se ha considerado la mejor opción para ello.

2.1.1. Entorno de trabajo

En este apartado se explican los equipos empleados para el desarrollo del proyecto y sus características más importantes.

2.1.1.1. Analizador de redes

Un analizador de redes nos permite analizar las propiedades de las redes eléctricas, aunque se centran en las propiedades relacionadas con la reflexión y la transmisión de señales eléctricas, comúnmente llamados parámetros S o de dispersión.

Existen dos tipos de analizadores de redes, SNA (*Scalar Network Analyzer*) y VNA (*Vector Network Analyzer*). El primer tipo de analizador de redes únicamente mide propiedades de amplitud, mientras que el segundo es capaz de medir propiedades de amplitud y fase. Normalmente, cuando hablamos de un "Analizador de Redes", nos referimos a un VNA. Los fabricantes más comunes de estos son *Keysight Technologies* (*Agilent* antiguamente), *Anritsu* y *Rhode & Schwarz*.

El analizador de redes del que disponemos en el laboratorio es el modelo N5227A de *Keysight Technologies* de dos puertos. Su rango de frecuencias va desde los 10 MHz hasta los 67 GHz y permite medir parámetros S y otros parámetros de potencia (A, B, R1, R2, a1, a2, b1, b2) en cualquiera de los dos puertos disponibles. En la Figura 2.1 podemos ver el modelo del analizador utilizado.

Figura 2.1: Analizador de Redes *Keysight* N5227A (de 2 puertos)

2.1.1.2. Sistema de posicionadores

Para la medida del PDP utilizaremos los posicionadores mencionados anteriormente. Estos harán un barrido (en forma de rejilla o circular) según las instrucciones que se ordenarán desde la interfaz programada. Para cada posición de los posicionadores (según el tipo de medida intervendrán los dos posicionadores o solo uno de ellos) se realizará la medida de la función de transferencia del canal $T(f,t)$ y una vez haya finalizado el barrido, los datos estarán disponibles para su procesado.

Los dos posicionadores mencionados están formados por una plataforma con ejes que se desplazan a lo largo de estos con ayuda de motores. El posicionador X es el más simple, está formado por un solo eje de 59 pulgadas (150 cm) y un motor. El posicionador XY está compuesto por dos ejes, de 30 pulgadas, montados en perpendicular para barrer una superficie de unos 5800 cm^2, y dos motores (uno para cada eje). En las Figuras 2.2 y 2.3 se pueden ver los dos tipos de posicionadores empleados. Estos pertenecen a la empresa de *Arrick Robotics*.

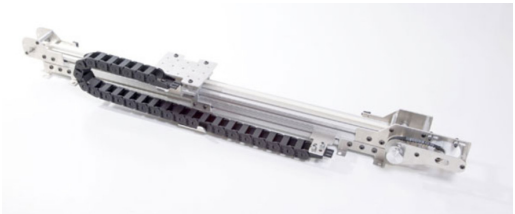


Figura 2.2: Posicionador X

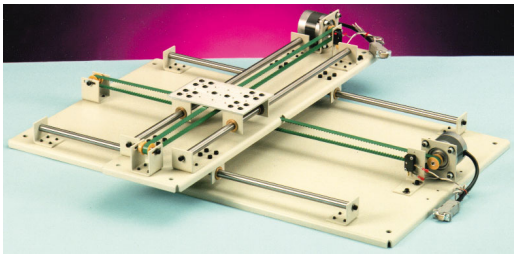


Figura 2.3: Posicionador XY

Para el control de los motores de los dos posicionadores se dispone del sistema C4/MD-2, también de *Arrick Robotics*, que permite programar el control del movimiento de la dirección, posición y velocidad de cada motor de los posicionadores. Se conecta a un puerto RS-232 o con USB tipo A al ordenador. Por tanto, desde nuestra interfaz indicaremos todos estos movimientos que debe hacer, y en cada posición el analizador deberá medir la función de transferencia del canal. En la Figura 2.4 se puede ver una imagen de estos dos dispositivos junto con los dos tamaños de motores.

Figura 2.4: Sistema controlador MD-2 y C4 de *Arrick Robotics*

2.1.1.3. Interfaz controlador

El control del analizador de redes y los dos posicionadores se realiza desde un ordenador que puede ser portátil. Para ello se usa la herramienta *App Designer* de MATLAB. Este entorno nos servirá para crear un programa desde el que podremos cambiar los parámetros del analizador por una parte y por otra automatizar los movimientos de los posicionadores. También usaremos la herramienta `tmtool` (*Test & Measurement Tool*) para crear un objeto VISA (*Visual Instrument Software Architecture*) sobre protocolo TCP/IP, a través del cual enviaremos las intrucciones al analizador desde MATLAB.

La configuración del sistema sería la siguiente: el ordenador estaría conectado al analizador y a los dos posicionadores (al controlador de estos), y el analizador, a su vez, se conectaría desde los puertos 1 y 2 a las antenas transmisora y receptora para ver la respuesta del canal. Un posible esquema de la sonda sería el de la Figura 2.5.

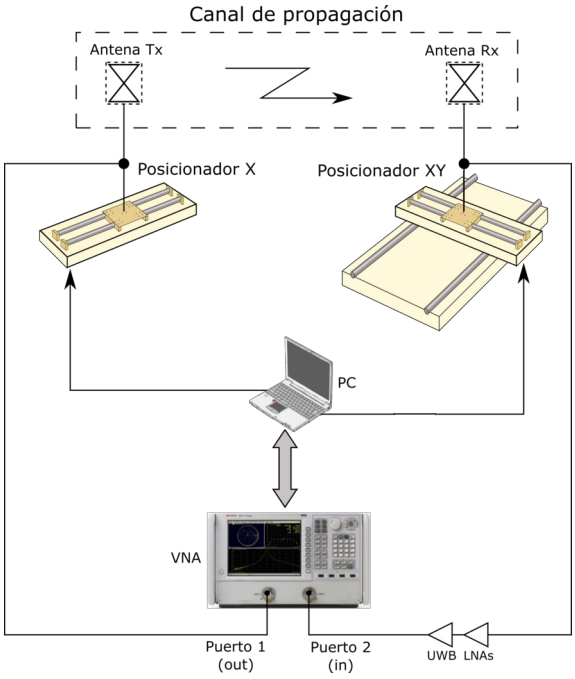


Figura 2.5: Ejemplo de configuración de la sonda de medida

2.1.2. Tipo de programación

El analizador de redes de *Keysight* nos permite programar de dos maneras: SCPI y COM [2]

- Los comandos SCPI (*Standard Commands for Programmable Instruments*) definen un estándar para la sintaxis de los comandos que se utilizan para controlar dispositivos de prueba y medida programables, como el analizador de redes.
  Los comandos SCPI son cadenas de texto ASCII, que se envían al instrumento a través de la capa física (por ejemplo, IEEE-488.1). Los comandos son una serie de una o más palabras clave seguidas de los parámetros de configuración adecuados.
- COM (*Component Object Model*) es un estándar de interfaz binaria para componentes software introducido por Microsoft en 1993. Permite la creación de objetos de comunicación en una amplia gama de lenguajes de programación.
  COM utiliza un protocolo binario, lo que le permite invocar directamente una función PNA. Esto lo hace más eficiente que SCPI.

Para nuestro objetivo, COM introduce la ventaja de que los tiempos de espera al enviar instrucciones al analizador sean menores que en SCPI, ya que en este último debe decodificar antes la secuencia de texto ASCII. La sonda anterior se programó con COM, pero surgieron algunos problemas. Uno de ellos, y quizás el principal, fue la limitación de instrucciones que ofrecía COM, ya que se creaba un objeto a partir del driver que facilita el fabricante *Keysight*, y este driver no estaba diseñado para trabajar en MATLAB sino en *Visual Basic* o *Visual C++*. Por tanto al final se tubo que utilizar SCPI para muchas de las instrucciones que enviaba la sonda, ya que no permitía cambiar el parámetro de la medida, modificar la potencia de salida o utilizar algunas funciones del *Trigger* entre algunas de las muchas restricciones. Actualmente existe un driver para MATLAB pero sólo tiene una versión de 32 bits, que lo hace incompatible con el ordenador.

Por todo ello, se ha decidido utilizar SCPI para todo el programa, ya que aunque en algunas instrucciones los tiempos de espera sean mayores, evitamos tener que crear dos objetos para el mismo analizador y quitamos una complejidad innecesaria a la programación. SCPI está estandarizado y el fabricante nos ofrece una lista completa de los comandos para el analizador, lo que lo hace sencillo y eficaz [3].

2.2. Distribución en tareas

El desarrollo del trabajo se descompone en las siguientes tareas:

1. Estudios previos y búsqueda de información y documentación
    1.1. Lectura y búsqueda de información acerca de la teoría del canal radio.
    1.2. Documentación sobre programación de interfaces en App Designer.
    1.3. Documentación sobre el uso de la programación SCPI para el analizador Keysight N5227A.
2. Definición y diseño de la sonda.
    2.1. Establecer la comunicación entre el ARV y el ordenador.
    2.2. Diseño y programación de la interfaz gráfica para la configuración del ARV.
    2.3. Diseño de la interfaz gráfica para la configuración de los posicionadores.
3. Desarrollo y redacción de la memoria del trabajo.

En la siguiente página se muestra un diagrama de Gantt con todas las fechas y duración de cada tarea.

2.2.1. Diagrama temporal

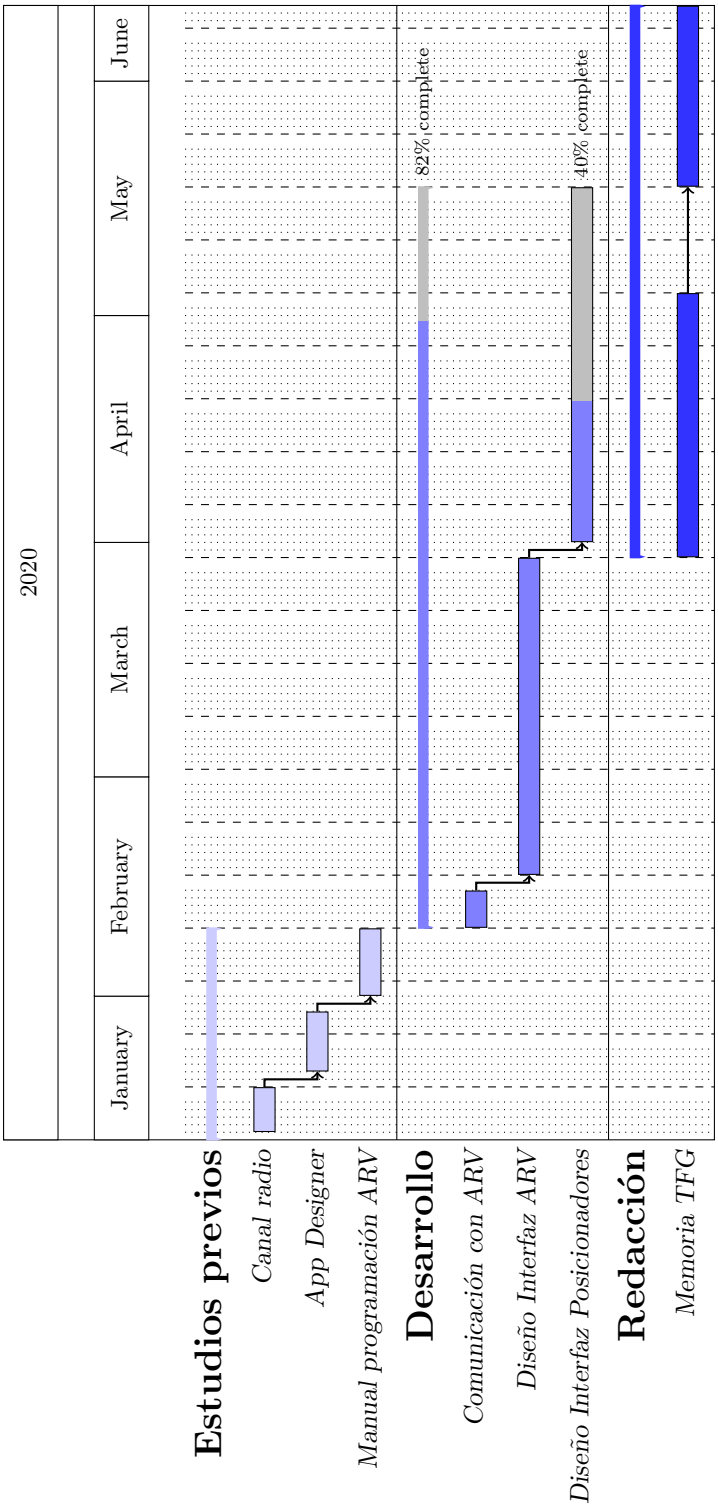


Figura 2.6: Diagrama de Gantt

Capítulo 3

Canal radio

3.1. Introducción

La propagación de ondas de radio se rige por la teoría del electromagnetismo establecida por el físico y matemático escocés James Clerk Maxwell, quien demostró que la electricidad, el magnetismo y la luz son manifestaciones del mismo fenómeno.

Una comprensión completa de la propagación de las ondas de radio es fundamental para la correcta implantación de muchos sistemas, incluidas las comunicaciones móviles, la detección y localización por radio (RADAR) y la navegación del sistema de posicionamiento global (GPS), por poner algunos ejemplos. Es esencial en estos y otros sistemas poder medir el canal y ver la respuesta al impulso del canal o la respuesta en frecuencia. Esto puede usarse en el transmisor y/o en el receptor para garantizar que los datos se transfieran de la manera más efectiva con distorsión, interferencia y pérdida de señal mínimas.

Una serie de factores contribuyen a la pérdida de transmisión de las ondas de radio, como los efectos atmosféricos, el terreno y la propagación a través de los edificios. La estimación de la pérdida de transmisión es importante para garantizar una calidad de servicio aceptable mediante la ubicación de los transmisores en ubicaciones apropiadas y con niveles de potencia adecuados para proporcionar cobertura sobre el área geográfica deseada. Asimismo, al receptor le llegarán diferentes contribuciones de la señal enviada debido a distintos obstáculos (edificios, árboles, coches, montañas, personas, etc.), por lo que aparecerán replicas de la señal procedentes de reflexiones, difracciones, *scattering*, etc. y con distintos niveles de potencia. A este fenómeno se le denomina efecto multicamino.

3.2. Mecanismos de propagación y efecto multicamino

Por mecanismos de propagación se entienden los procesos físicos que intervienen en la propagación de las ondas electromagnéticas: principalmente atenuación, reflexión especular, reflexión difusa, difracción, refracción y *scattering* (dispersión) [4].

El estudio de la propagación electromagnética se divide en cuatro áreas, designadas como onda de superficie, onda ionosférica, onda de dispersión troposférica y onda espacial. En la Figura 3.1

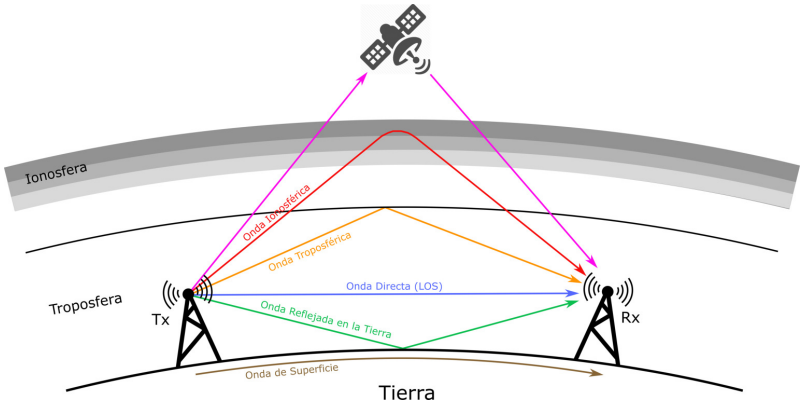


Figura 3.1: Mecanismos de propagación de las ondas de radio

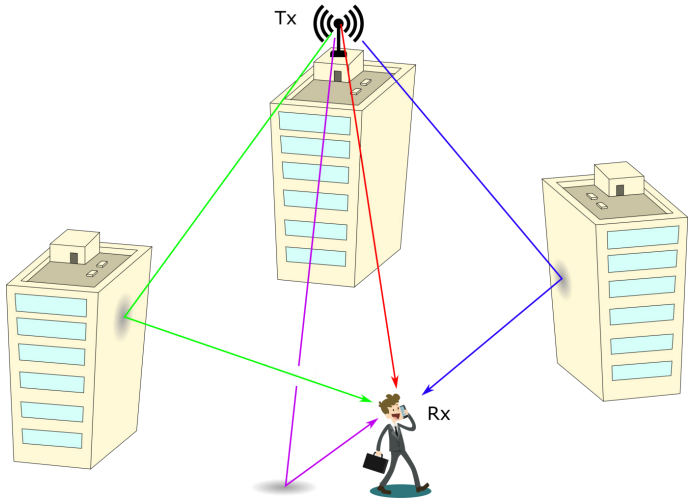


Figura 3.2: Ejemplo del efecto multicamino

podemos ver una representación de los distintos tipos de propagación.

Debido a los distintos obstáculos que se encuentra una onda en su trayecto tienen lugar estos mecanismos de propagación y, por tanto, la señal que llega al receptor resulta deteriorada. Los obstáculos reducen el nivel de la señal captada por el receptor. Esta atenuación es conocida como desvanecimiento lento o ensombrecimiento (*shadow fading* o *shadowing*).
La señal radio también puede verse sometida a reflexiones en los obstáculos presentes en el camino (no necesariamente en la línea de visión directa). Este fenómeno, conocido como propagación multicamino, ocasiona la llegada al receptor de distintas señales con diferente fase y distinto nivel de potencia. Un ejemplo de propagación multicamino es el que se muestra en la Figura 3.2. La propagación multicamino da lugar a desvanecimientos rápidos.

Para poder desplegar un sistema de comunicaciones es necesario conocer como afectarán todos estos efectos del canal y corregirlos. Para ello hacemos uso de la caracterización del canal radio, que mediante aproximaciones, puede describir el comportamiento del canal tanto en el dominio del tiempo como en el de la frecuencia.

3.3. Caracterización del canal radio

En aplicaciones de banda ancha correspondientes a los sistemas radioeléctricos digitales modernos, es necesario conocer las consecuencias de la propagación multicamino y la variabilidad con el tiempo del canal radio. Hablamos, por tanto, de la caracterización en banda ancha de canales multitrayecto.

Una consecuencia de la propagación multicamino es la llegada al receptor de múltiples ecos con amplitudes, fases y tiempos de llegada distintos. A este efecto se le llama dispersión temporal. Ésta produce, en el dominio del tiempo, interferencia entre símbolos (ISI) y en el dominio de la frecuencia, en sistemas de banda ancha, desvanecimiento selectivo en frecuencia (*Frequency Selective Fading*). Por todo esto, la señal recibida tendrá cierta distorsión.

En cuanto a los efectos debidos a la variabilidad del canal al desplazarse un terminal, la amplitud de la tensión recibida varía en función del tiempo. Esto producirá dos efectos. El primero, el desvanecimiento espacial se transforma en desvanecimiento selectivo en el tiempo (*Time Selective Fading*). El segundo, las variaciones temporales de la tensión dan lugar a una dispersión en frecuencia, asociada al desplazamiento Doppler (*Doppler shift*), que produce variaciones de las frecuencias espectrales de la señal.

3.3.1. Funciones del sistema

Los canales multitrayecto variables con el tiempo pueden caracterizarse mediante funciones de transferencia y de respuesta impulsiva. Debido a los fenómenos de variación temporal y desplazamiento Doppler, se manejan cuatro variables: tiempo (t), retardo o dispersión temporal (τ), frecuencia (f) y el desplazamiento Doppler (ϑ).

Tenemos por tanto cuatro ecuaciones, llamadas funciones del sistema o funciones de *Bello* [5], que son las siguientes:

1. Función de respuesta impulsiva variable con el tiempo (IDSF: *Input Delay Spread Function*): $h(t, \tau)$
2. Función de transferencia variable con el tiempo (TVTF: *Time Variable Transfer Function*): $T(f, t)$
3. Función desplazamiento Doppler-retardo (DDSF: *Doppler Delay Spread Function*): $S(\tau, \vartheta)$
4. Función de transferencia frecuencia-desplazamiento Doppler (ODSF: *Output Doppler Spread Function*): $H(f, \vartheta)$

Estas cuatro funciones están relacionadas entre si mediante transformadas de Fourier directas (DFT) e inversas (IFT) tal como se indica en la Figura 3.3.

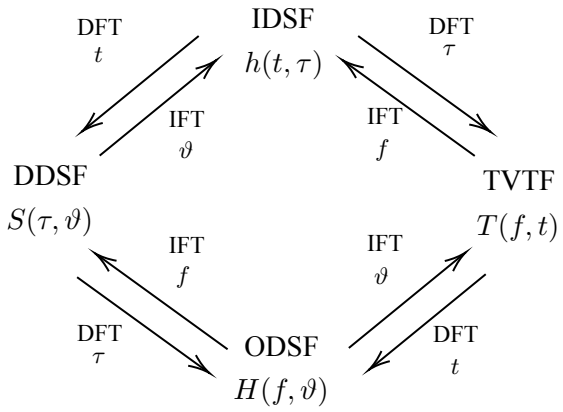


Figura 3.3: Relaciones de Fourier entre las funciones del sistema

La caracterización de un canal variable mediante estas funciones es puramente teórica. En la realidad, los canales varían aleatoriamente con el tiempo y las funciones del sistema se convierten en procesos estocásticos. Por tanto, se realiza un estudio aproximado utilizando las funciones de correlación, donde $\mathrm{E}[\cdot]$ es el cálculo de la esperanza matemática de una variable aleatoria:

$$\begin{aligned} R_{\mathrm{h}}(t_1,t_2;\tau_1,\tau_2) &= \mathrm{E}[h^*(t_1,\tau_1)h(t_2,\tau_2)] \\ R_{\mathrm{T}}(f_1,f_2;t_1,t_2) &= \mathrm{E}[T^*(f_1,t_1)T(f_2,t_2)] \\ R_{\mathrm{H}}(f_1,f_2;\vartheta_1,\vartheta_2) &= \mathrm{E}[H^*(f_1,\vartheta_1)H(f_2,\vartheta_2)] \\ R_{\mathrm{S}}(\tau_1,\tau_2;\vartheta_1,\vartheta_2) &= \mathrm{E}[S^*(\tau_1,\vartheta_1)S(\tau_2,\vartheta_2)] \end{aligned} \tag{3.1}$$

Pese a esta simplificación, sigue siendo complicada la caracterización de los canales utilizándolas. Por ello, aprovechando el comportamiento de los canales variables multitrayecto que se observan en la práctica, se realizan otras simplificaciones y consideramos los siguientes canales prácticos:

1. WSS, *Wide Sense Stationary*. Canal estacionario en sentido amplio, para recorridos pequeños de los terminales, t_1 y t_2 difieren poco, por que las funciones de correlación temporal dependen solo de la diferencia de tiempos.

2. US, *Uncorrelated scattering*. Se dice que el canal tiene dispersión Doppler incorrelada.

3. WSSUS, *Wide Sense Stationary Uncorrelated Scattering*. Combina los dos anteriores, ya que los canales reales tienen, con buena aproximación, las propiedades de WSS y US. Consideraremos que los canales cumplirán estas características.

Finalmente, las funciones de correlación son las siguientes, donde $u = t_2 - t_1$ y $v = f_2 - f_1$.

$$\begin{aligned} &\text{IDSF: } R_{\mathrm{h}}(u;\tau_1-\tau_2) = \delta(\tau_2-\tau_1)P_{\mathrm{h}}(u,\tau) \\ &\text{TVTF: } R_{\mathrm{T}}(v;u) \\ &\text{ODSF: } R_{\mathrm{H}}(v;\vartheta_1-\vartheta_2) = \delta(\vartheta_2-\vartheta_1)P_{\mathrm{H}}(v,\vartheta) \\ &\text{DDSF: } R_{\mathrm{S}}(\tau_1,\tau_2;\vartheta_1,\vartheta_2) = \delta(\vartheta_2-\vartheta_1)\delta(\tau_2-\tau_1)P_{\mathrm{S}}(\tau,\vartheta) \end{aligned} \tag{3.2}$$

A continuación se explican estas funciones:

1. $P_{\mathrm{h}}(u,\tau)$: Función de correlación en la variable de tiempo u (WSS) y función densidad de potencia en la variable τ (US). Si hacemos $u = 0$, la función resultante es $P_{\mathrm{h}}(\tau)$, y aporta información de la dispersión de la potencia (en dB) en función del retardo τ. A esta se le denomina perfil de retardo de potencia (PDP, *Power Delay Profile*) y de ella se extraen muchos parámetros usados para la caracterización de los canales multitrayecto.

2. $R_{\mathrm{T}}(v;u)$: Función de correlación en la variable de frecuencia v (WSS) y función de correlación en la variable de tiempo u (US). A partir de $R_{\mathrm{T}}(v)$ ($u = 0$) obtenemos el ancho de banda de coherencia B_{c} y análogamente, a partir de $R_{\mathrm{T}}(u)$ ($v = 0$) podemos obtener el tiempo de coherencia T_{c}.

3. $P_{\mathrm{H}}(v,\vartheta)$: Función de correlación en la variable de frecuencia v (WSS) y función densidad de potencia en la variable dispersión Doppler ϑ (US).

4. $P_{\mathrm{S}}(\tau,\vartheta)$: Función densidad de potencia en la variable de dispersión τ y función densidad de potencia en la variable dispersión Doppler ϑ. La función $P_{\mathrm{S}}(\theta)$ proporciona la variación de la potencia en función de la dispersión Doppler y se le denomina perfil potencia Doppler, la cual se obtiene a partir de la siguiente integral:

$$P_{\mathrm{S}}(\theta) = \int_0^{\infty} P_{\mathrm{S}}(\tau,\theta)d\tau \tag{3.3}$$

La función $P_{\mathrm{S}}(\tau,\vartheta)$ es importarte para la caracterización del canal radio ya que nos da información sobre la dispersión temporal y la frecuencial, al igual que el PDP, del que se pueden extraer varios parámetros.

Al igual que las funciones del sistema, estas funciones de correlación están relacionadas mediante transformadas de Fourier, tal como se representa en la Figura 3.4.

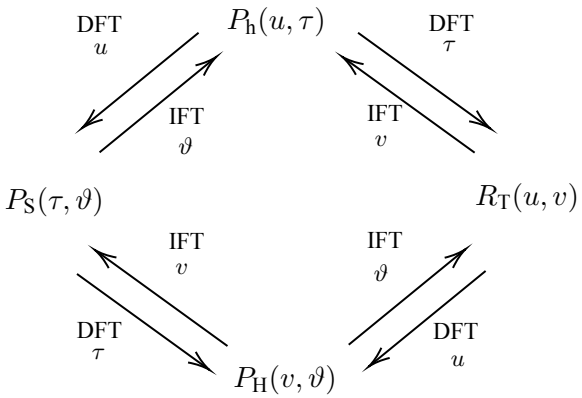


Figura 3.4: Funciones características de canales WSSUS

3.4. Parámetros para la caracterización del canal radio

Matemáticamente el PDP se define como:

$$P(\tau) = \sum_{k=0}^{N-1} a_k^2 \delta(t - \tau_k) \tag{3.4}$$

Donde a_k y τ_k son las amplitudes reales y los retardos, respectivamente, del k-ésimo componente multitrayecto para N componentes.

Para la caracterización del canal se utilizan algunos parámetros básicos [5] , los cuales se extraen del PDP y de la función de *scattering*:

a. Para describir la dispersión temporal: *Mean Excess Delay*, *RMS Delay Spread* y Ancho de banda de coherencia.

b. Para describir la dispersión en frecuencia: Dispersión Doppler y Tiempo de coherencia.

3.4.1. Parámetros de dispersión temporal

Mean Excess Delay

Define el retardo medio de las contribuciones y tiene la siguiente expresión:

$$\overline{\tau} = \frac{\int_{\tau_{\text{min}}}^{\tau_{\text{máx}}} \tau P(\tau) d\tau}{\int_{\tau_{\text{min}}}^{\tau_{\text{máx}}} P(\tau) d\tau} \tag{3.5}$$

En las medidas siempre hay un umbral de ruido (NF, *Noise Floor*) por debajo del cual se considera que lo que se mide es ruido térmico, por lo que no contribuye al PDP. En la expresión, τ_{min} es el retardo mínimo y corresponde al valor del primer cruce con NF y $\tau_{\text{máx}}$ corresponde al valor del último cruce con NF.

RMS (*Root Mean Square*) *Delay Spread*

Es el más importante de los parámetros que definen la dispersión temporal del canal. Los retardos de las contribuciones se miden respecto a la primera señal que se recibe. Su expresión es la siguiente:

$$\sigma_\tau = \sqrt{\frac{\int_{\tau_{\text{min}}}^{\tau_{\text{máx}}} (\tau - \overline{\tau})^2 P(\tau) d\tau}{\int_{\tau_{\text{min}}}^{\tau_{\text{máx}}} P(\tau) d\tau}} \tag{3.6}$$

Ancho de banda de coherencia

B_c es una medida del rango de frecuencias sobre el cual el canal puede ser considerado plano (atenuación y el desplazamiento en fase permanecen aproximadamente constantes) y tiene una relación inversa con σ_τ.

Cuando el ancho de banda de la señal a transmitir es mayor que el ancho de banda de coherencia del canal, se produce desvanecimiento selectivo en frecuencia e interferencia entre símbolos. Por el contrario, si el ancho de banda de la señal es menor que B_c, se dice que estamos transmitiendo con *flat fading*.

Si el ancho de banda de coherencia se define como el ancho de banda sobre el cual la función de correlación de frecuencia está por encima de 0.9, entonces el ancho de banda de coherencia es

aproximadamente:

$$B_{\mathrm{c}} \approx \frac{1}{50\sigma_\tau} \tag{3.7}$$

Si la definición se relaja para que la función de correlación de frecuencia sea superior a 0.5, entonces el ancho de banda de coherencia es aproximadamente:

$$B_{\mathrm{c}} \approx \frac{1}{5\sigma_\tau} \tag{3.8}$$

3.4.2. Parámetros de dispersión en frecuencia

Dispersión Doppler

Se denomina dispersión Doppler, o *Doppler Spread*, al momento de segundo orden de $P_S(\vartheta)$, un concepto similar al del PDP. Su expresión es la siguiente:

$$B_{\mathrm{D}} = \sqrt{\frac{\int_{-f_{\mathrm{d}}}^{f_{\mathrm{d}}} (\vartheta - \overline{\vartheta})^2 P_{\mathrm{S}}(\vartheta) d\vartheta}{\int_{-f_{\mathrm{d}}}^{f_{\mathrm{d}}} P_{\mathrm{S}}(\vartheta) d\vartheta}} \tag{3.9}$$

siendo $\overline{\vartheta}$ su valor medio:

$$\overline{\vartheta} = \frac{\int_{-f_{\mathrm{d}}}^{f_{\mathrm{d}}} \vartheta P_{\mathrm{S}}(\vartheta) d\vartheta}{\int_{-f_{\mathrm{d}}}^{f_{\mathrm{d}}} P_{\mathrm{S}}(\vartheta) d\vartheta} \tag{3.10}$$

En la expresión, f_{d} corresponde al valor máximo de desplazamiento Doppler y toma el siguiente valor, donde f_{c} es la frecuencia central de la señal transmitida y v, la velocidad del receptor respecto a la fuente:

$$f_{\mathrm{d}} = \frac{v f_{\mathrm{c}}}{c_0} \tag{3.11}$$

Tiempo de coherencia

De forma similar al ancho de banda de coherencia para la dispersión temporal, el tiempo de coherencia T_c es la longitud del intervalo de tiempo en el cual podemos considerar que la respuesta del canal no cambia. En este caso si el tiempo de coherencia es mayor que el de la señal, hablaremos de desvanecimiento lento o no selectividad en el tiempo. Al contrario, tendremos desvanecimiento rápido o selectividad en el tiempo. En este caso, T_c es aproximadamente la inversa del valor máximo de desplazamiento Doppler:

$$T_{\mathrm{C}} \approx \frac{1}{f_{\mathrm{d}}} \tag{3.12}$$

Para un canal con estadística Rayleigh se tiene:

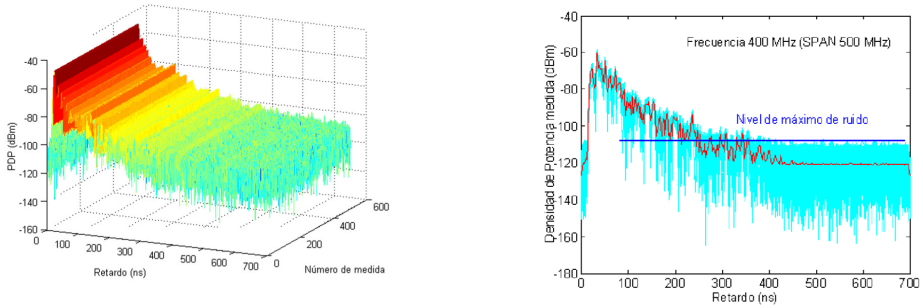
$$T_C = \sqrt{\frac{9}{16\pi f_d^2}} \tag{3.13}$$

3.5. Técnicas de medida y equipamiento.

Existen una serie de técnicas de medida y equipamiento que podemos utilizar para realizar una caracterización experimental del canal radio en base a medidas. En este apartado se explica qué es lo que nos interesa medir para analizar el canal radio desde un punto de vista experimental, qué técnicas de medida podemos utilizar y por último cuál es la instrumentación de medida que podemos utilizar en base a la técnica de medida [6].

3.5.1. Medida para la caracterización del canal radio.

Uno de los parámetros que nos va a aportar la información que estamos buscando a la hora de analizar el comportamiento de ese canal radio es el PDP. Este nos permite describir el comportamiento selectivo del canal en términos de frecuencia a partir de la función de correlación en frecuencia y nos permite también evaluar el comportamiento dispersivo a través del parámetro de Delay Spread.



(a) **(b)**

Figura 3.5: Ejemplo de *Power Delay Profile* (Fuente [6])

En la Figura 3.5a se muestra la medida de un PDP medido de forma continua en el tiempo. Los niveles altos corresponderían a las contribuciones multicamino que están llegando a la señal y por tanto en esas 500 medidas que se han realizado apenas hay una variación de la potencia de esas contribuciones. Pero en los valores de ruido se puede ver una mayor variación. Si superponemos los 500 PDPs medidos obtendríamos la representación que aparece en la Figura 3.5b. En las zonas donde tenemos señal, el margen dinámico de esas variaciones es más pequeño que en la zona donde tenemos ruido.

Para mejorar la relación señal a ruido en el estudio de ese PDP podemos hacer el promediado de esos 500 perfiles que se han medido, nos quedaría entonces el perfil que aparece en rojo en la imagen de la derecha. Al hacer esto disminuimos la varianza del ruido. A modo de ejemplo, podemos establecer un nivel de ruido como el marcado en la imagen, de modo que recortamos el

PDP a ese nivel de ruido (también denominado nivel de *Threshold*) y analizamos los parámetros de dispersión en el tiempo y los parámetros de correlación en base a este PDP.

En cualquier instrumento de medida el ruido siempre va a existir debido al carácter resistivo de los elementos del receptor y lo que interesará es realizar varias medidas para hacer el promediado y por tanto reducir el efecto del ruido, mejorando por tanto la relación señal a ruido.

3.5.2. Técnicas de medida.

En este apartado se explica cómo podemos medir el parámetro en el que estamos interesados. Podemos utilizar técnicas en el dominio del tiempo o en el dominio de la frecuencia. Las técnicas que se utilizan en el dominio del tiempo nos llevan a medir la respuesta impulsional del canal $h(t, \tau)$. Por otro lado, las técnicas que se utilizan en el dominio de la frecuencia miden la función de transferencia del canal. En ambos casos, mediante transformadas directas o inversas de Fourier podemos obtener el resto de funciones del sistema y por tanto parámetros como el Power Delay Profile.

Dominio del tiempo

- Caracterización por impulsos.
- Caracterización por espectro ensanchado.

Dominio de la frecuencia

- Estimación de la función de transferencia cronovariable.

3.5.2.1. Caracterización por impulsos.

Mediante la técnica de caracterización por impulsos transmitimos un pulso y medimos los ecos en recepción y así podemos estimar la respuesta impulsional del canal radio.

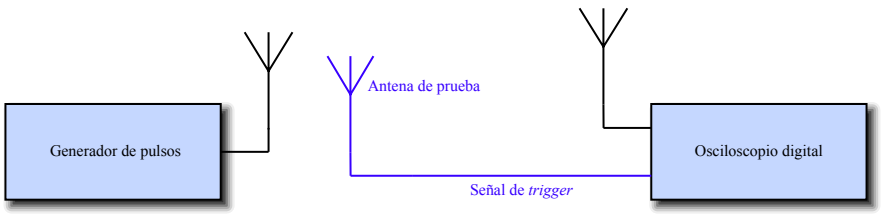


Figura 3.6: Esquema para la caracterización por impulsos

El ancho de banda empleado en la medida depende de la forma y de la duración del pulso, lo ideal es utilizar pulsos muy estrechos (típicamente del orden de nanosegundos) para conseguir una buena resolución temporal en la respuesta impulsional.

Como instrumento para generar los pulsos podemos utilizar un generador de pulsos convencional y como instrumento en la parte del receptor para detectar los distintos ecos podemos utilizar un osciloscopio digital. El osciloscopio digital iría tomando trazas, muestreado la amplitud de la señal a lo

largo del tiempo. Como se ha comentado antes, mediríamos varias trazas para después promediar y así eliminar el efecto indeseado del ruido. Es necesario inicialmente transmitir un pulso y hacer una captura de una traza lo suficientemente larga como para detectar el retardo máximo de propagación y a partir de ahí automatizar el sistema de medida. Esto es necesario para evitar desechar ecos que nos llegan con retardos más o menos grandes. Esta técnica de medida también necesita una antena adicional para el disparo de la captura. Esta antena captará la primera contribución multicamino y eso activará una señal de trigger para indicar al osciloscopio que empiece a realizar las capturas de las tramas. En ocasiones podemos prescindir de esta antena y hacer una captura o varias consecutivas en el tiempo y posteriormente mediante técnicas de procesado offline, sincronizar las distintas tramas para realizar el promediado.

Podemos encontrar distintos equipos generadores de pulsos que nos permiten modificar la frecuencia, el ancho y la forma del pulso. Si este generador de pulsos es del orden de nanosegundos necesitamos también osciloscopios digitales con una frecuencia de muestreo muy elevada para poder tener una buena resolución en esa respuesta impulsional.

3.5.2.2. Caracterización por espectro ensanchado

La técnica de caracterización por espectro ensanchado es la otra técnica que podemos utilizar en el dominio del tiempo. Con esta técnica transmitimos una secuencia pseudo aleatoria, una secuencia de pseudo ruido. Esa secuencia se propaga por el canal y llega al receptor. En recepción correlamos la señal recibida con la secuencia de pseudo ruido que hayamos transmitido y lo que tendremos a la salida es la respuesta impulsional del canal radio. Esta es la técnica que se suele utilizar en la mayor parte de los terminales radio para hacer una estimación del canal.

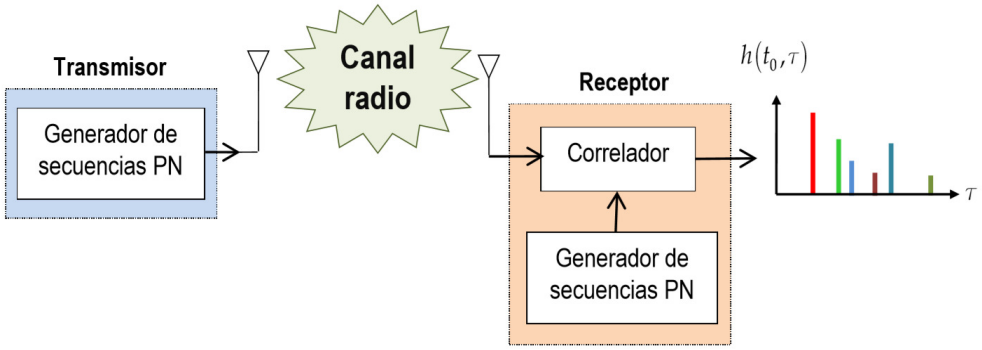


Figura 3.7: Esquema para la caracterización por espectro ensanchado (Fuente [6])

La secuencia de pseudo ruido que transmitidos es una secuencia de longitud de N símbolos en el chip y en recepción haremos la correlación. La salida del correlador la muestreamos a la tasa de chip que estamos utilizando y cada valor representará el retardo introducido por el canal (*taps*).
La resolución de esa respuesta impulsional será inversamente proporcional a la tasa de chip, por lo tanto cuando el chip tenga una duración temporal muy pequeña la resolución temporal será mayor. El máximo retardo detectable sin alias será N veces el tiempo de chip (NT_C), siendo T_C el tiempo de chip. Esto es importante a la hora de seleccionar la longitud de la secuencia de pseudo ruido ya que para canales muy dispersivos esta secuencia deberá ser muy larga para tener una buena resolución.

Podemos utilizar como equipos transmisores y equipos receptores generadores y analizadores vectoriales (I-Q) de señal. Al generador le habremos grabado una secuencia de pseudo ruido que estará transmitiendo y en recepción utilizaremos un analizador vectorial (I-Q). Después mediante un procesado *offline* (por ejemplo con MATLAB) podemos estimar la respuesta impulsional del canal haciendo la correlación de la señal que hemos detectado con la secuencia que estamos transmitiendo.

3.5.2.3. Estimación de la función de transferencia cronovariable.

En el dominio de la frecuencia lo que haríamos sería estimar la función de transferencia. Podemos utilizar un analizador de redes vectorial para medir la respuesta del canal. La antena transmisora se conectaría al puerto 1 del analizador de redes y la antena receptora se conectaría al puerto 2, de esta manera podemos medir el parámetro S_{21}. Ese parámetro es proporcional a la función de transferencia del canal que queremos medir en el instante t_0, ya que lo que estamos midiendo no es la función de transferencia del canal sino que medimos la función de transferencia enventanada en una ventana rectangular. Esto es así porque medimos desde una frecuencia inicial hasta una frecuencia final, por tanto hay un efecto de enventanado.

$$S_{21} \propto T(f,t_0) \equiv \frac{Y(f,t_0)}{X(f,t_0)} \tag{3.14}$$

$$T_w(f,t_0) = T(f,t_0)W(f) \tag{3.15}$$

Cuando hacemos la transformada inversa de Fourier a esta medida para calcular la respuesta impulsional vemos que tenemos la respuesta impulsional del canal convolucionada con la respuesta de esa ventana rectangular, que será una *sinc*:

$$h_w(t_0,\tau) = \mathcal{F}^{-1}T_w(f,t_0) = h(t_0,\tau) \otimes w(t) \tag{3.16}$$

A partir de la medida de la respuesta impulsional podemos obtener el PDP en ese instante de tiempo calculando el módulo de esta respuesta impulsional y elevándolo al cuadrado.

$$PDP(t_0,\tau) = |h_w(t_0,\tau)|^2 \tag{3.17}$$

Si lo que hacemos son varias medidas en el tiempo o en posiciones muy próximas en términos de longitud de onda, lo que podemos calcular es el PDP promedio como el promediado de los PDP individuales.

$$PDP(\tau) = \frac{1}{N}\sum_{i=1}^{N} PDP(t_i,\tau) \tag{3.18}$$

El barrido en frecuencia que realiza el analizador de redes lo hace a través de una sinusoide desde una frecuencia inicial hasta una frecuencia final y esa diferencia se denomina SPAN. Por tanto en el

analizador de redes debemos indicar el SPAN que estamos utilizando en la medida y la frecuencia central f_0:

$$f_0 = \frac{f_2 - f_1}{2} \tag{3.19}$$

Si utilizamos un SPAN muy grande la resolución en el tiempo es muy buena y a medida que el SPAN se va reduciendo la resolución temporal empeora y los distintos ecos empiezan a solaparse unos con otros.

A partir de la respuesta que ha medido el analizador de redes (parámetro S_{21} en frecuencia) hay equipos que permiten hacer directamente la IFFT y por tanto tendríamos la respuesta impulsional del canal enventanada. Si el equipo no lo tiene, lo que podemos hacer es grabar esas trazas que hemos medido y, por ejemplo con MATLAB realizar esta IFFT.

Durante el tiempo de medida tenemos que garantizar condiciones de estacionariedad en el canal, es decir, todo tiene que permanecer estático (sin gente moviéndose o elementos que puedan alterar la medidas durante ese tiempo). También tenemos que tener en cuenta que los cables utilizados en el analizador de redes introducen pérdidas adicionales, además de evitar usar amplificadores porque pueden ocasionar reflexiones que podríamos confundir con contribuciones multicamino. En bandas de milimétricas por encima de 30 GHz, donde las atenuaciones son muy grandes, una alternativa es sustituir el coaxial por enlaces de radio sobre fibra.

3.5.3. Uso de posicionadores

Las campañas de medida que se realizan para caracterizar el comportamiento del canal involucran el uso de posicionadores lineales o rectangulares con el fin de hacer una caracterización a pequeña escala.

El sistema transmisor o receptor se puede montar sobre un posicionador lineal o rectangular con el fin de emular un array lineal virtual o en el caso del posicionador XY, podemos mover la antena en un plano horizontal. La antena se va desplazando vamos midiendo la respuesta impulsional del canal.

El poder medir en un array rectangular donde los elementos del array o las distintas posiciones de la antena están espaciadas una la longitud menor que ½ lo que nos permite es, mediante distintas técnicas y algoritmos, poder estimar la dirección de llegada de las contribuciones. El conocer la dirección de llegada de las contribuciones multicamino es importante sobre todo para el análisis de sistemas MIMO y para los sistemas *massive* MIMO en 5G. Entre los algoritmos que podríamos utilizar destacan el algoritmo MUSIC, ESPRIT, SAGE y RiMAX siendo estos dos últimos los más comúnmente utilizados por su precisión.

Capítulo 4

Interfaz Gráfica de Usuario

4.1. Introducción

La Interfaz Gráfica de Usuario, abreviada como GUI (*Graphical User Interface*), es un programa informático que actúa de intermediario entre usuario y máquina. Para ello utiliza un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. La finalidad es proporcionar un entorno visual sencillo en el que un usuario pueda interactuar con una máquina sin necesidad de tener conocimientos de informática.

Surge como evolución de las interfaces de línea de comandos (CLI) que se empleaban para los primeros sistemas operativos y es una parte fundamental en cualquier entorno gráfico de los dispositivos actuales. Existen de diversos tipos, como ZUI (*Zooming User Interface*), un entorno gráfico donde los usuarios pueden cambiar la escala de la vista para ver más detalles o menos, y navegar a través de diferentes documentos. También tenemos las NUI (*Natural User Interface*), que son las interfaces de las pantallas táctiles y permiten una interacción directa con el usuario mediante sus manos.

Figura 4.1: Evolución de las interfaces de usuario (Fuente [7])

Existen diversos lenguajes de programación que permiten el desarrollo de interfaces gráficas de usuario. Entre los más utilizados se encuentran C# (*C Sharp*), C++, Vala, GTK+, PHP, Glade, Python, JavaScript y Java.

Destinados a ingeniería de sistemas tenemos LabVIEW, pensado para sistemas hardware y software de pruebas, control y diseño, simulado o real y embebido. También tenemos MATLAB, el cual hemos utilizado en este proyecto ya que nos permite crear interfaces gráficas de usuario y comunicarnos con programas y dispositivos hardware de forma sencilla e intuitiva.

4.2. Interfaz gráfica de usuario en MATLAB

MATLAB (proveniente de *MATrix LABoratory*) es una aplicación de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M).

Este software ofrece la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware.

Para expander sus prestaciones, MATLAB dispone de herramientas adicionales como Simulink (plataforma de simulación multidominio), GUIDE y *App Designer* (editores de interfaces de usuario). Además, se pueden ampliar las funcionalidades de MATLAB con las *Toolboxes* y las de Simulink con *Blocksets*.

En definitiva, con MATLAB podemos crear interfaces de usuario de tres formas distintas:

1. Manualmente con funciones.
2. Con GUIDE.
3. Con App Designer.

Mediante el primer método, se codifica el diseño de la interfaz y el comportamiento de esta con funciones de MATLAB exclusivamente. Esto hace que tengamos que indicar manualmente la posición y características de cada objeto con los que se van a interactuar, lo que demora y dificulta la programación de la interfaz.

Por el contrario, GUIDE es un entorno gráfico que permite el diseño de interfaces gráficas de usuario. Las ventajas que ofrece GUIDE respecto a la programación con funciones son varias. Para empezar, se diferencian por una parte el diseño de la interfaz y por otra el código que define el comportamiento. Para el diseño de la interfaz existen una serie de componentes predefinidos en GUIDE que podemos arrastrar al área de trabajo y colocar en la posición deseada, además de cambiar las propiedades de estos objetos. Estos objetos aparecen automáticamente en el código junto con las propiedades que les hemos dado, por lo que acelera la programación. También tenemos algunas funciones predefinidas que podemos añadir al código, como la función *StartupFcn* o las retrollamadas (*callbacks*) disponibles para cada componente, que permiten hacer cambios cuando el usuario interactúa con estos.

App Designer se introduce en la versión R2016a de MATLAB. A diferencia de GUIDE, ésta ofrece una mayor integración entre el *Layout* de la aplicación y el código, además de ser una plataforma más renovada y más atractiva para el usuario. También contiene una gama de componentes más grande y una programación del código más fácil e intuitiva.

4.3. Diferencia entre GUIDE y *App Designer*

En este apartado se van a exponer algunas diferencias entre *App Designer* y su predecesor, GUIDE. Estas son algunas de las razones por las que se ha decidido programar con *App Designer*:

1. **Editor de código**. En App Designer se edita el código dentro de un editor que posee la aplicación, además este contiene indentación automática e indicaciones de error o advertencias de *Code Analyzer*. Este editor no permite editar todas las partes del código, como en GUIDE, si no que para evitar cometer errores solo podemos editar propiedades, funciones y retrollamadas.

2. **Retrollamadas innecesarias**. App Designer no crea retrollamadas de los componentes a no ser que el programador desee añadirlas.

3. **Propiedad *Value* en las retrollamadas**. Cuando añadimos una retrollamada, en la mayoría de los componentes se crea automáticamente una línea de código dentro de la función donde se accede a la propiedad *Value* de dicho objeto.

4. **Archivo único**. *App Designer* solo crea un archivo .mlapp que contiene todo lo referido a la aplicación. En GUIDE teníamos un .fig además.

5. **Propiedades de los componentes**. GUIDE utiliza las funciones get y set para acceder y cambiar los valores de las propiedades de los componentes. En cambio *App Designer* introduce el acceso a las propiedades a través de puntos. Por ejemplo, para un Edit Field numérico llamado Velocidad, cambiamos su valor de la siguiente manera:

   GUIDE
   ```
   set(handles.Velocidad,'Value') = 80;
   ```
   App Designer
   ```
   app.Velocidad.Value = 80;
   ```

6. **Argumentos de entrada de retrollamadas**. En GUIDE, las retrollamadas tienen como argumentos de entrada hObject, eventdata y handle. El argumento hObject contiene el objecto que está siendo utilizado, eventdata contiene los eventos que suceden y handle es una estructura con el identificador de los objetos. En cambio en *App Designer* tenemos app y event. App hace referencia a la aplicación en si y event contiene información sobre el evento que ocurre. Por ejemplo para un Edit Field que ha sido editado:

   GUIDE
   ```
   function editfield_Callback(hObject, eventdata, handle)
        value = get(hObject,'Value');
   end
   ```
   App Designer
   ```
   function EditFieldValueChanged(app, event)
        value = app.EditField.Value;
   end
   ```

Además de las anteriores características, hay algunas más de las que GUIDE no disponía y se han añadido en *App Designer*, son las siguientes [8]:

- Compartir como MATLAB Web App.
- Contenedores con barra deslizante.
- Administrador de diseño de cuadrícula.
- Selector de fechas.
- *Edit Field* numérico.
- Añadir imágenes.
- Spinner.
- Botón de estado.
- Listas de jerarquía en árbol.
- Componentes de Instrumentación, Aeroespacial y HTML.

Por todos los motivos anteriormente mencionados, hemos elegido programar la sonda con *App Designer*, ya que da un mejor resultado. A continuación se explican los distintos componentes que ofrece y una breve descripción de como funciona *App Designer*.

Para este proyecto, se ha usado la versión R2020a de MATLAB. En esta nueva versión ya no está disponible GUIDE, cosa que en las versiones anteriores sí. También es posible migrar aplicaciones antiguas de GUIDE a *App Designer*.

4.4. *App Designer*

4.4.1. Inicio

Podemos iniciar *App Designer* desde dos sitios distintos dentro de MATLAB. El primero es desde ***HOME→New→App***, y la segunda forma de acceder es seleccionando ***APPS→Design App***. Podemos observarlo en las Figuras 4.2 y 4.3.

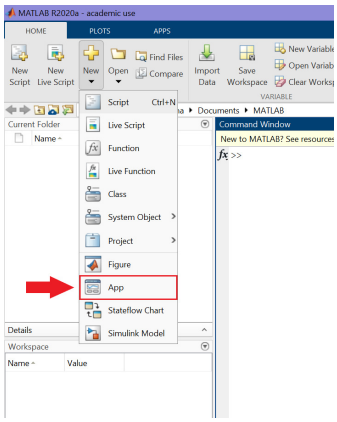


Figura 4.2: Acceso a *App Designer* (1)

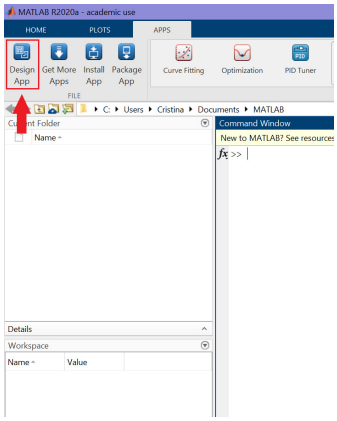


Figura 4.3: Acceso a *App Designer* (2)

Una vez abierto, aparece la siguiente página de inicio en la que podemos crear una aplicación en blanco, cargar un ejemplo de los que ofrece o abrir una aplicación (extensión `.mlapp`) que tengamos creada en nuestro ordenador.

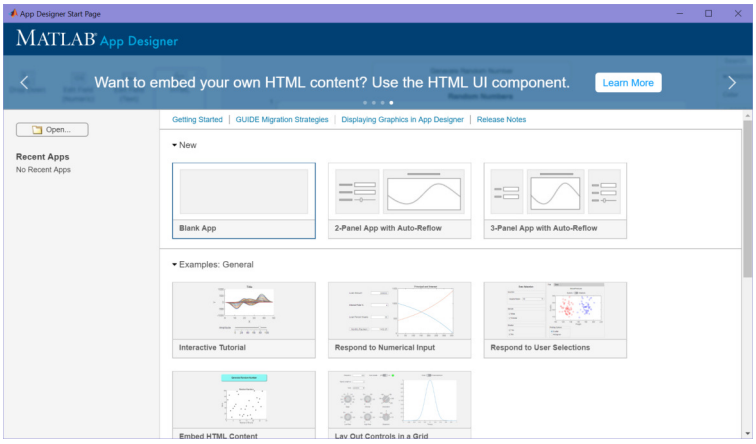


Figura 4.4: *App Designer Start Page*

Para ver las características de *App Designer* iniciamos una aplicación en blanco. Aquí podemos distinguir entre dos partes muy importantes de la creación de una aplicación: la parte visual y el código. Para editar el diseño de la aplicación tendremos que posicionarnos en la pestaña de *Design View*, y para programar las acciones de los componentes cambiaremos a *Code View*.

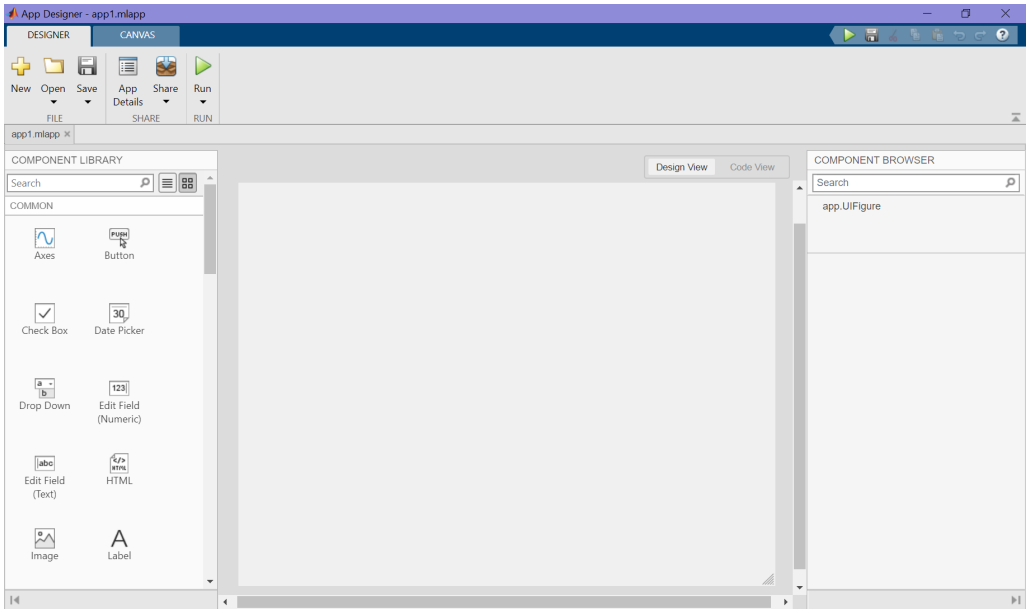


Figura 4.5: Pestaña *Design View*

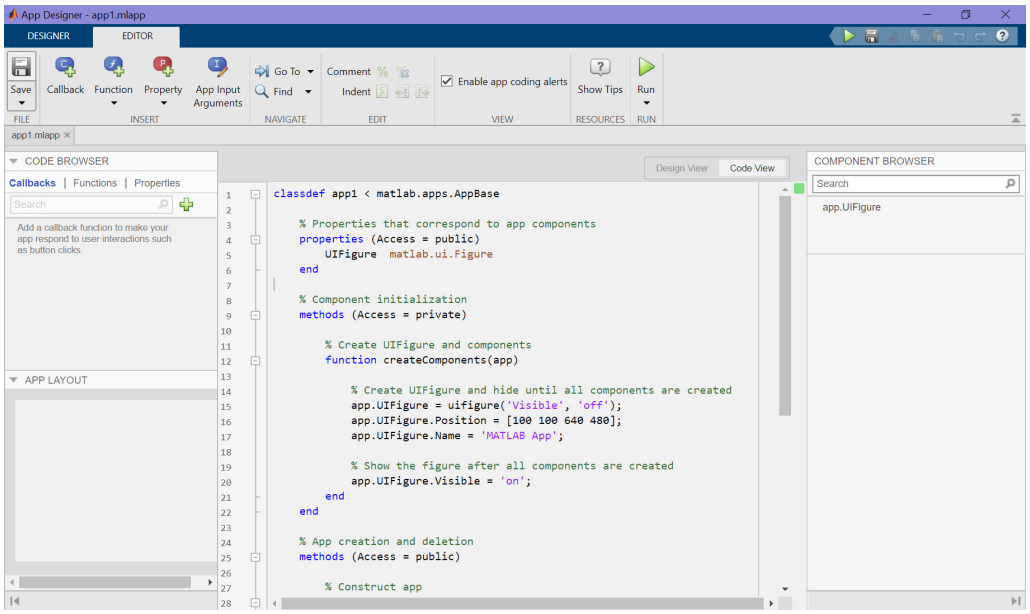


Figura 4.6: Pestaña *Code View*

4.4.2. *Design View*

Si nos situamos sobre la pestaña *Design View* podemos editar la apariencia de la aplicación arrastrando los distintos componentes que tenemos en el panel de la izquierda sobre el área en blanco y cambiar algunas de sus propiedades. Los distintos paneles que tenemos para el diseño de la apariencia son los siguientes:

1. ***Component Library***. En este panel situado a la izquierda se encuentran los distintos componentes que podemos utilizar para el diseño. Aparte de los componentes que aparecen aquí también existen otros que podemos añadir programáticamente. En el Apartado 4.4.4 se detallan los distintos componentes que se pueden utilizar.

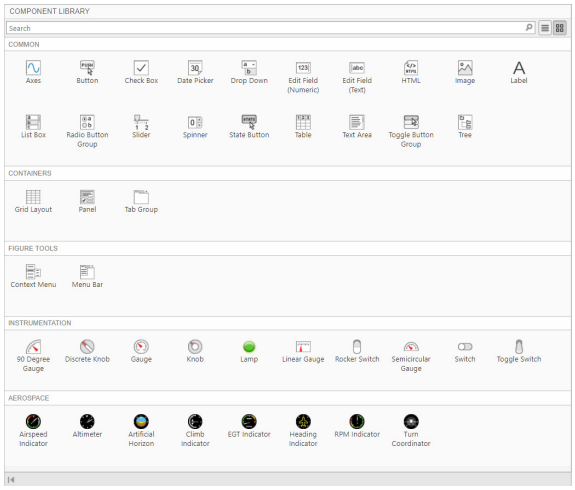


Figura 4.7: Panel *Component Library*

2. ***Component Browser***. En esta pestaña se sitúa a la derecha y es común para *Design View* y *Code View*. Aquí se puede ver en la parte superior una lista jerarquizada de los distintos componentes que se hayan colocado en el área de edición. En la Figura 4.8 se ha colocado un *Edit Field* (numérico) en el área en blanco como ejemplo. Al seleccionar uno de los componentes de la lista podemos editar en la parte inferior sus características como en este caso la etiqueta, el valor, los límites superior e inferior, el alineamiento, la fuente y el color, la posición, etc.

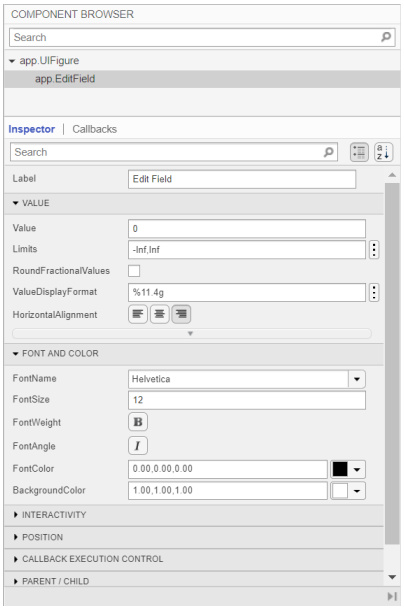


Figura 4.8: Panel *Component Browser*

3. **Barra de Herramientas**. En la barra superior podemos ver dos pestañas, *Designer* (común también para *Design View* y *Code View*) y *Canvas*. En la primera tenemos las opciones básicas de abrir, crear, o guardar, además de poder editar algunos detalles de la aplicación, compilarla para crear una aplicación de escritorio o web y la opción de correr la app. En la pestaña *Canvas* tenemos algunas herramientas para el diseño visual de la aplicación como alinear objetos, ajustar el tamaño de estos, ver líneas guía, etc.

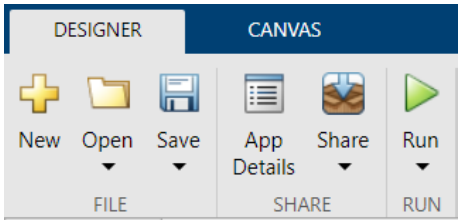


Figura 4.9: Pestaña *Designer*

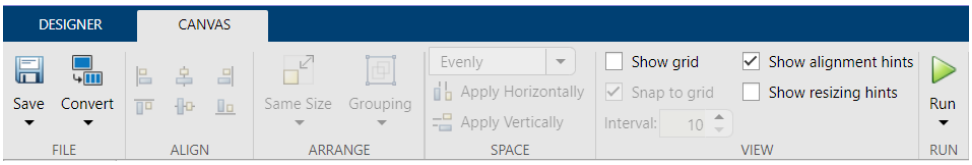


Figura 4.10: Pestaña *Canvas*

4.4.3. *Code View*

Cuando vemos la parte del código, aparecen algunos paneles distintos a los del *Design View*, además de los comunes mencionados anteriormente:

1. ***Code Browser***. En este panel se pueden administrar y crear tres partes fundamentales del código: retrollamadas, funciones y propiedades.

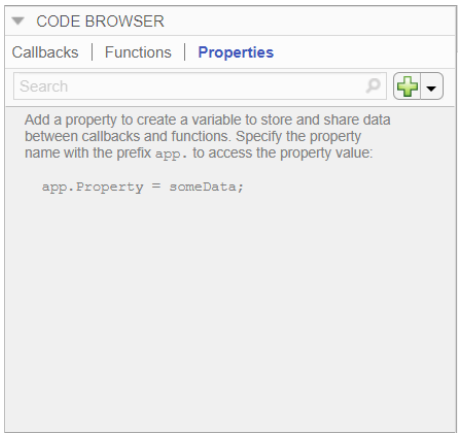


Figura 4.11: Panel *Code Browser*

a. **Retrollamadas (*callbacks*)**. Son funciones que se ejecutan en respuesta a las interacciones que tiene el usuario con los distintos componentes, como por ejemplo escribir un número por teclado o presionar un botón.
Cada componente tiene una serie de retrollamadas predefinidas, que podemos añadir desde el panel *Code Browser* presionando el signo de añadir (+). También existen retrollamadas para el UI Figure (la aplicación en si) definidas, como la función *StartupFcn* donde pondríamos el código que queramos que se ejecute nada más iniciar la aplicación, entre otras funciones.

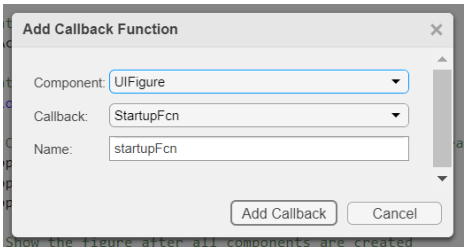


Figura 4.12: Crear retrollamada

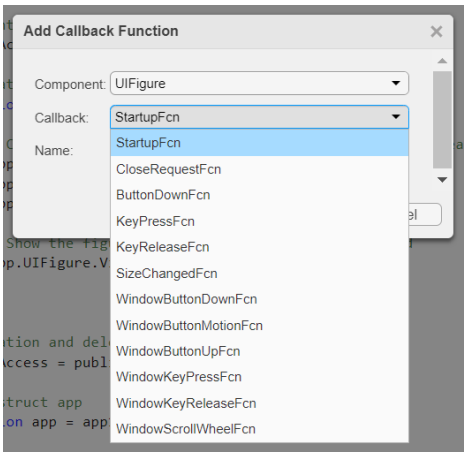


Figura 4.13: Retrollamadas para UI Figure

Otra forma de añadir una retrollamada es haciendo click derecho sobre el propio componente en el *Design View* o en la lista de componentes del panel *Component Browser* y seleccionar la retrollamada que queremos añadir (*Callbacks→Add callback*). Esta se creará automáticamente en el código donde podremos escribir las acciones que queremos que ocurran cuando interactuemos con alguno de los componentes.

b. **Funciones**. Estas funciones se crean para evitar tener que repetir las mismas acciones varias veces dentro del código. Existen funciones privadas y públicas. Estas últimas se utilizan para ser llamadas entre aplicaciones distintas.

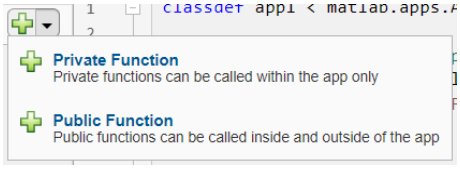


Figura 4.14: Crear función

Una vez seleccionado el tipo de función que queremos crear, se crea automáticamente en el código el marco para la función, al que podemos añadirle más argumentos de entrada o salida:

```
methods (Access = private)
      function results = func(app)
      end
end
```

```
methods (Access = private)
      function [a b] = func(app, c)
      end
end
```

c. **Propiedades**. Se usan para compartir datos entre funciones o retrollamadas. Accedemos a ellas a través del prefijo `app`. Por ejemplo si tenemos una propiedad llamada `temperatura` y queremos llamarla dentro de una retrollamada o una función para cambiarle el valor la llamaremos de la manera `app.temperatura = 30`, por ejemplo. Al igual que las funciones las propiedades pueden ser también públicas o privadas. Los componentes también tienen algunas propiedades definidas por MATLAB, como por ejemplo si queremos que en alguna ocasión un botón no esté disponible y no se pueda presionar (o al contrario), pondremos el siguiente código:

```
app.Button.Enable = 'off'
```

```
app.Button.Enable = 'on'
```

2. **Pestaña *Editor***. Se encuentra en la barra de herramientas superior y tenemos opciones para navegar dentro del código, comentarlo o editarlo, dejar que nos muestre *tips* de ayuda, y también podemos añadir retrollamadas, funciones y propiedades.

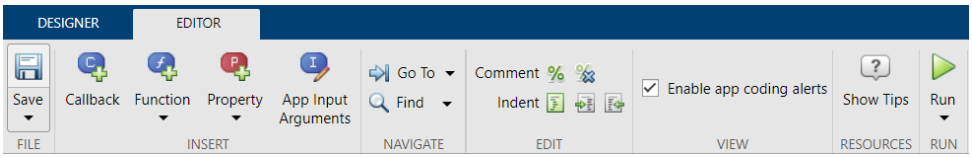


Figura 4.15: Pestaña *Editor*

Para finalizar, podemos dividir la estructura del código en tres partes tal como se muestra abajo. La **parte A** se crea automáticamente al posicionar los componentes en la aplicación y no podemos editar el código directamente. Esta parte define las propiedades correspondientes a los componentes, en el código de ejemplo esta el fondo de la aplicación y un botón. La **parte B** corresponde a todo lo que sí podemos editar en el código, aquí añadimos por orden primero las propiedades, luego las funciones y luego las retrollamadas. La **parte C** inicializa los componentes y la aplicación con todas sus características. Esta parte tampoco es editable manualmente desde el código, para editarla debemos cambiar las propiedades de los componentes desde el panel *Component Browser* (pestaña *Inspector*).

Parte A

```
classdef app1 < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure  matlab.ui.Figure
        Button    matlab.ui.control.Button
    end
```

Parte B

```
    properties (Access = private)
        Property % Description
    end

    methods (Access = private)

        function results = func(app)

        end
    end

    % Callbacks that handle component events
    methods (Access = private)

        % Button pushed function: Button
        function ButtonPushed(app, event)

        end
    end
```

Parte C

```
    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [100 100 640 480];
            app.UIFigure.Name = '\acrshort{matlab} App';

            % Create Button
            app.Button = uibutton(app.UIFigure, 'push');
            app.Button.ButtonPushedFcn = createCallbackFcn(app,
            ↪  @ButtonPushed, true);
            app.Button.Position = [271 296 100 22];

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = app1

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

4.4.4. Librería de componentes

En este apartado se van a exponer y detallar brevemente los distintos componentes y objetos que se pueden utilizar para crear una aplicación. [9]

4.4.4.1. Representación gráfica

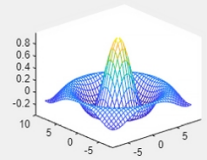
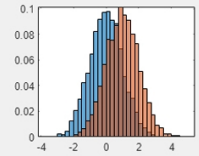
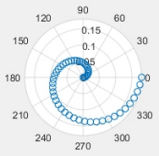
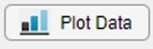
REPRESENTACIÓN GRÁFICA	
	**UI Axes** Objeto que se usa para representar datos 2D o 3D, integrado en *app designer*. Los modos de zoom, panorámica y rotación solo admiten un subconjunto de opciones para los ejes de la interfaz de usuario.
	**Axes**[1] Los clásicos ejes para la representación de datos. Estos admiten todo el conjunto de opciones.
	**Geo Axes**[1] Muestra datos en coordenadas geográficas (latitud/longitud) en un mapa. El mapa está en vivo, se puede desplazar para ver otras ubicaciones geográficas y acercar y alejar el mapa para ver con más detalle.
	**Polar Axes**[1] Crea una figura con ejes polares para representar datos.

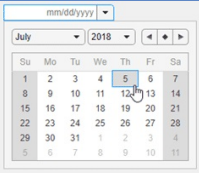
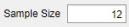
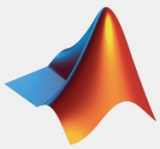
Tabla 4.1: Componentes para la representación de datos

4.4.4.2. Componentes comunes

COMPONENTES COMUNES	
	**Button** Los botones son componentes de la interfaz de usuario que responden cuando el usuario los presiona y los suelta. Estos pueden tener texto e icono además de un texto de ayuda cuando pasas el ratón por encima (la mayoría de componentes permiten añadir uno).

Continúa en la siguiente página

[1]Disponible solo mediante programación

COMPONENTES COMUNES	
	Check Box Una casilla de verificación es un componente para indicar el estado de una preferencia u opción. El valor puede ser 0 o 1. También permite tener un texto de ayuda.
	**Date Picker** Los selectores de fechas permiten a los usuarios seleccionar fechas de un calendario interactivo. El calendario permite establecer limites de selección, inhabilitar fechas concretas o un rango de días de la semana.
	**Drop Down** Las listas desplegables son componentes de la interfaz de usuario que permiten al usuario seleccionar una opción o escribir texto (si está la opción *Editable* activada). Los elementos pueden tener datos asociados.
	**Edit Field (Number)** Son campos de edición numéricos y permiten introducir valores numéricos al usuario en la aplicación. El objeto permite introducir límite superior e inferior, redondeo de números fraccionarios o mostrar las unidades especificadas.
	**Edit Field (Text)** Permite al usuario introducir texto por teclado, y permite seleccionar el alineamiento de este.
	**Image** Inserta una imagen a partir de un directorio existente en el ordenador, como un logotipo o un icono. Se pueden cambiar las propiedades de esta como el alineamiento y la escala.
	Label Muestra un texto estático para etiquetar partes de la aplicación.
	**List Box** Muestra una lista de ítems personalizados, los cuales pueden tener datos asociados. Esta lista puede ser multiselección, es decir, el usuario puede seleccionar varios ítems a la vez si esta opción está activada.

Continúa en la siguiente página

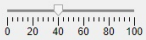
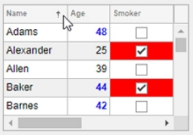
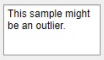
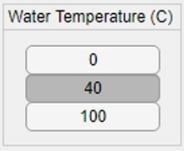
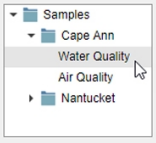
COMPONENTES COMUNES	
	**Radio Button Group** Permite crear un conjunto de opciones de selección, dentro de estas solo puede estar seleccionada una, los estados de estos son 0 o 1.
	**Slider** Control deslizante que permite al usuario seleccionar un valor dentro de un rango de valores definido, no son valores exactos. Permite orientación horizontal y vertical.
	**Spinner** Tiene la misma funcionalidad que los campos de edición numéricos, pero en este caso se puede añadir un paso al incrementar o decrementar el valor.
	**State Button** Botón de estado, tiene los estados 0 o 1, es decir, el botón puede estar seleccionado o no. Permite añadir texto e icono.
	**Table** Crea una tabla para mostrar datos en la aplicación. La tabla permite datos numéricos y de texto, además de elementos lógicos como un *check box*.
	**Text Area** Permite al usuario introducir varias líneas de texto o editarlas. El alineamiento horizontal es editable.
	**Toggle Button Group** Presentan un conjunto de opciones dentro de un grupo de botones. El usuario puede seleccionar uno de todos ellos (son parecidos a los botones de estado).
	**Tree** Los árboles son componentes para presentar listas de elementos en una jerarquía dentro de una aplicación. Se les puede asignar iconos y pueden ser editables y también multiselección.

Tabla 4.2: Componentes comunes

4.4.4.3. Contenedores y herramientas de figuras

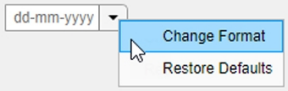
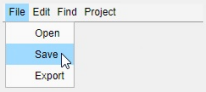
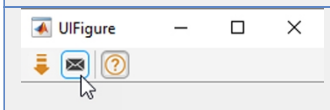
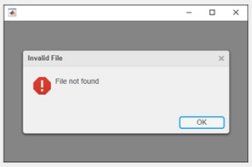
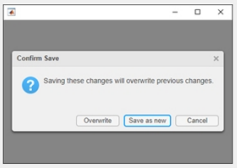
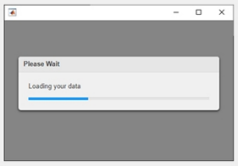
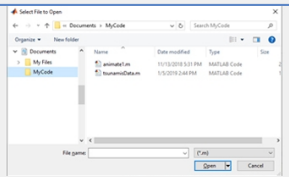
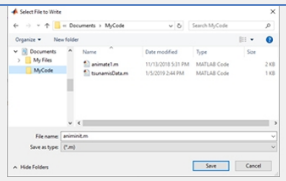
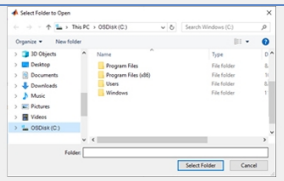
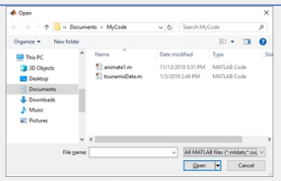
CONTENEDORES Y HERRAMIENTAS DE FIGURAS	
	**Context Menu** Es un menú de opciones que aparece cuando hacemos click derecho sobre un objeto gráfico o un componente. El menú puede tener submenús y tantas opciones como queramos. Todos los componentes pueden tener uno.
	**Menu Bar** Muestran listas desplegables de opciones en la parte superior de una ventana de la aplicación. Las opciones del menú pueden tener atajos desde el teclado tipo ***Ctrl+X*** además de opciones con indicadores o *checks* con valor 0 o 1.
	**Grid Layout** Colocan los componentes de la IU a lo largo de las filas y columnas de una cuadrícula invisible que abarca toda la figura o un contenedor dentro de la figura.
	Panel Crean un panel dentro del área de diseño para agrupar varios componentes juntos. Pueden tener título y ser desplazables con el *scroll* del ratón.
	**Tab Group** Crean contenedores con pestañas, con opción de colocarlas arriba o abajo o en los lados.
	**Toolbar**[1] Permite crear una barra de herramientas horizontal en la parte superior de la aplicación.

Tabla 4.3: Contenedores y herramientas de figuras

[1] Disponible solo mediante programación

4.4.4.4. Diálogos y notificaciones

DIÁLOGOS Y NOTIFICACIONES	
	**Alert**[1] Crea una ventana de diálogo con la función `uialert` que muestra un mensaje. Por defecto lleva el icono de error y el botón OK.
	**Confirmation**[1] Crea una ventana de diálogo de confirmación con la función `uiconfirm` que muestra un mensaje y varios botones con opciones personalizables.
	**Progress**[1] Crea un cuadro de diálogo con un determinado progreso con la función `uiprogressdlg`.
	**Color Picker**[1] Muestra un selector de color modal y devuelve el color como un vector de valores RGB entre 0 y 1.
	**File Selection**[1] `uigetfile` abre un cuadro de diálogo modal que enumera los archivos en la carpeta actual. Permite a un usuario seleccionar o ingresar el nombre de un archivo. Devuelve el nombre del archivo cuando el usuario hace clic en Abrir.
	**Save File**[1] `uiputfile` abre un cuadro de diálogo modal para seleccionar o especificar un archivo.
	**Folder Selection**[1] `uigetdir` abre un cuadro de diálogo modal que muestra las carpetas en el directorio de trabajo actual y devuelve la ruta que el usuario selecciona en el cuadro de diálogo.
	**Load Variable**[1] `uiopen` abre un cuadro de diálogo modal que permite abrir un archivo de matlab y cargarlo en el *workspace*.

Continúa en la siguiente página

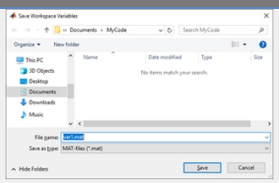
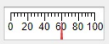
DIÁLOGOS Y NOTIFICACIONES	
	**Save Variable**[1] `uisave` abre un cuadro de diálogo modal que permite guardar las variables del *workspace* especificadas en una ruta.

Tabla 4.4: Diálogos y Notificaciones

4.4.4.5. Instrumentación

INSTRUMENTACIÓN	
	**Knob** Al igual que el componente Slider, este tampoco permite indicar números exactos. Se desliza la rueda para indicar un valor, los límites se indican en las propiedades del componente.
	**Discrete Knob** Este mando permite seleccionar entre varios estados limitados, y no un rango de números. Cada ítem puede tener un dato asociado.
	**Gauge** Los medidores son componentes que representan instrumentos de medición. Existen de varios modelos, tal como se muestran en los siguientes componentes.
	**90 Degree Gauge** Tal como expresa su nombre, este medidor tiene forma de ángulo de 90 grados. Los límites son personalizables como en el anterior.
	**Linear Gauge** Medidor de instrumentación en forma lineal, tiene el mismo funcionamiento que los anteriores.

Continúa en la siguiente página

[1] Disponible solo mediante programación

INSTRUMENTACIÓN	
	**Semicircular Gauge** Medidor de instrumentación de 180 grados de ángulo o semicircular. Mismo funcionamiento que los anteriores.
	**Lamp** Indican un estado usando colores. Los colores se especifican como un triplete RGB, un código de color hexadecimal o los colores ya definidos en MATLAB.
	**Switch** Es un interruptor que indica un estado lógico (0 o 1). Está como predefinido los estados 'On' y 'Off' pero se pueden personalizar. Estos dos ítems pueden tener datos asociados.
	**Rocker Switch** Al igual que el anterior, se trata de interruptor con dos estados. A este modelo se le llama interruptor basculante.
	**Toggle Switch** Interruptor de palanca, otra forma de interruptor con la misma funcionalidad.

Tabla 4.5: Instrumentación

4.4.5. Compartir aplicación

Para el posterior uso de la aplicación por otras personas, en otros ordenadores, tenemos dos opciones: compartir la aplicación entre otros usuarios de MATLAB o compilar la aplicación mediante MATLAB Compiler para que usuarios sin MATLAB puedan utilizarla.

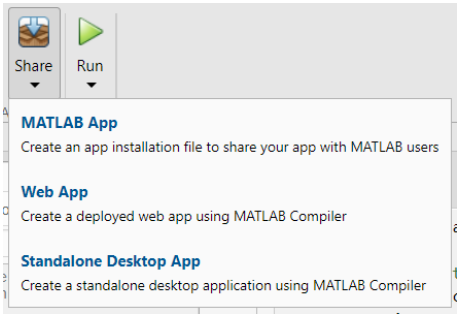


Figura 4.16: Opciones de compartir una aplicación de *App Designer*

4.4.5.1. Uso compartido de apps con otros usuarios de MATLAB

Se puede empaquetar cualquier app de MATLAB en un único archivo que se puede compartir con otros usuarios mediante MATLAB Desktop y MATLAB Online.

Cuando se empaqueta una app, MATLAB crea un único archivo de instalación (.mlappinstall). Con este archivo, otros usuarios pueden instalar la aplicación y acceder a ella desde la galería de apps. Si se comparte la aplicación con otros usuarios, se les puede permitir colaborar en el diseño, por tanto, se condecen permisos para editarla.

Para ello, hemos de dirigirnos a la pestaña ***DESIGNER→Share→MATLAB App***. Desde aquí podremos introducir la descripción de la aplicación y posteriormente empaquetarla.

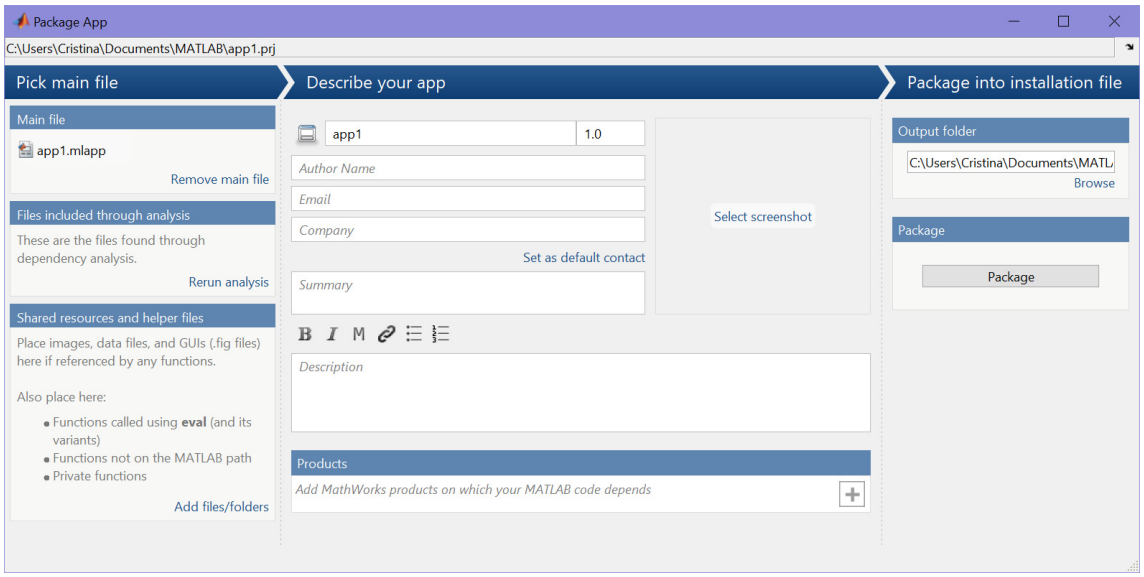


Figura 4.17: Empaquetar aplicación

4.4.5.2. Creación de apps de escritorio y web independientes

MATLAB Compiler [10] permite compartir programas de MATLAB como aplicaciones independientes y apps web. Con esto se puede crear un programa que pueda instalarse y abrirse en otro ordenador tenga instalado MATLAB o no.

Para crear una aplicación de escritorio nos dirigiremos a ***Designer→Share→Standalone Desktop App***.

En la parte superior seleccionamos las opciones para descargar Runtime, que son las bibliotecas que permitirán ejecutar el programa en los ordenadores que no tengan MATLAB instalado. En la primera opción este conjunto de bibliotecas se descargan desde la web mientras se está instalando la aplicación y en la segunda opción se almacenan en el instalador.

En esta ventana podemos también introducir una imagen y un nombre para el instalador, o una pantalla mientras se está instalando. Introducimos también la información del creador y el correo electrónico, además de una descripción del programa y la versión de este.

Una vez seleccionamos *Package* se crearán varias carpetas, en dos de ellas podemos encontrar el ejecutable para instalar el programa:

1. **Carpeta for_redistribution**. En esta carpeta encontraremos el archivo .exe de instalación que contiene además el Runtime para aquellos ordenador que no tengan MATLAB ni Runtime instalado previamente. En el ejecutable se descargarán todos los archivos necesarios.

2. **Carpeta for_redistribution_files_only**. En esta carpeta encontraremos también un ejecutable .exe pero en este caso no se descargará ni instalará Runtime. Este instalador solo sirve para los ordenadores que ya tengan MATLAB o Runtime instalados, ya que solo instalará la aplicación.

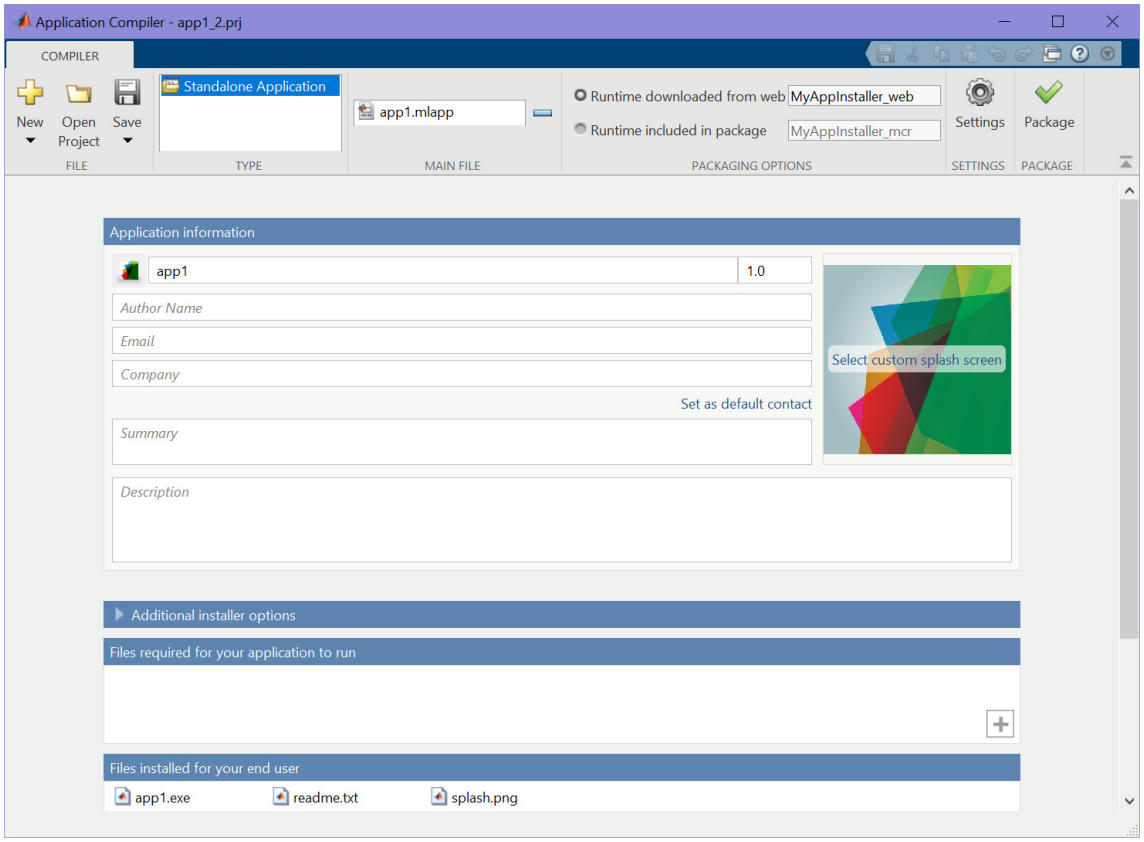


Figura 4.18: Aplicación de escritorio independiente de MATLAB

También podemos crear aplicaciones web mediante MATLAB Web App Server [11], este permite alojar aplicaciones de MATLAB como apps web interactivas. Creamos la aplicación con App Designer, se empaqueta con MATLAB Compiler y se aloja con MATLAB Web App Server. Los usuarios finales pueden acceder a las apps web y ejecutarlas utilizando un navegador, sin necesidad de instalar nada.

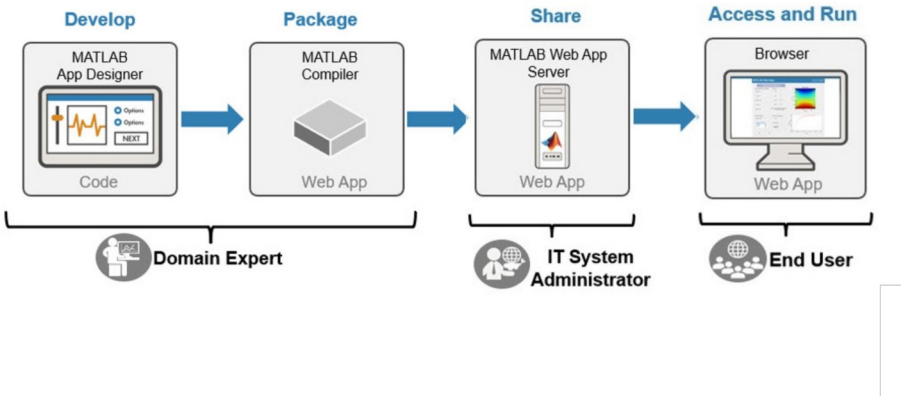


Figura 4.19: Proceso para la creación de una aplicación web (Fuente [11])

Capítulo 5

Desarrollo y resultados del trabajo

5.1. Introducción

En la Figura 5.1 se presenta un diagrama de flujo del funcionamiento de la aplicación, desde que se inicia hasta las funciones que se pueden realizar a grandes rasgos.

Una vez abrimos el programa, éste ejecuta su funcionamiento con la función *Start-Up*. Esta función comprueba si la conexión con el analizador de redes es correcta. Si no estamos conectados al ARV nos aparecerá un mensaje de error en el que podemos decidir si queremos volver a intentar hacer la conexión con el analizador o simplemente cerrar la aplicación. La aplicación no permite el acceso a los menús de configuración si el analizador de redes no está conectado, ya que no tiene ningún sentido y daría errores.

Si la conexión con el ARV ha sido correcta, entonces pasaremos a una pantalla donde podemos elegir entre configurar el analizador de redes o los posicionadores. Estas dos configuraciones son independientes una de la otra, por lo que se pueden configurar en el orden que se quiera.

En la pantalla de configuración del ARV tenemos dos opciones: cargar una configuración que tengamos guardada en un archivo `.mat` o podemos ir cambiando la configuración en la aplicación. Por defecto, los valores de los parámetros que aparecen al iniciar el programa son con los que también se inicia el analizador. En esta pantalla podemos cambiar el parámetro a medir en el analizador y seleccionar el puerto si este corresponde a un parámetro de potencia. También podemos cambiar la selección de las frecuencias a medir y otros parámetros como la potencia de salida, el ancho de banda IF, etc. Tenemos también la opción de realizar una medida con promediado, y por último cambiar la escala de visualización en la pantalla del ARV. Una vez configurado podemos guardar esta configuración para poder cargarla en otro momento si es que queremos la misma para posteriores medidas. Por último podremos realizar una medida con esta configuración o resetear todos los parámetros.

En la pantalla de configuración de los posicionadores podremos elegir el tipo de medida que queremos hacer: SISO, SIMO, MISO o MIMO. En las tres últimas seleccionaremos qué posicionador queremos que sea el transmisor o receptor, si el X o el XY. Una vez hecho esto podremos configurar la rejilla de estos según corresponda y realizar las medidas.

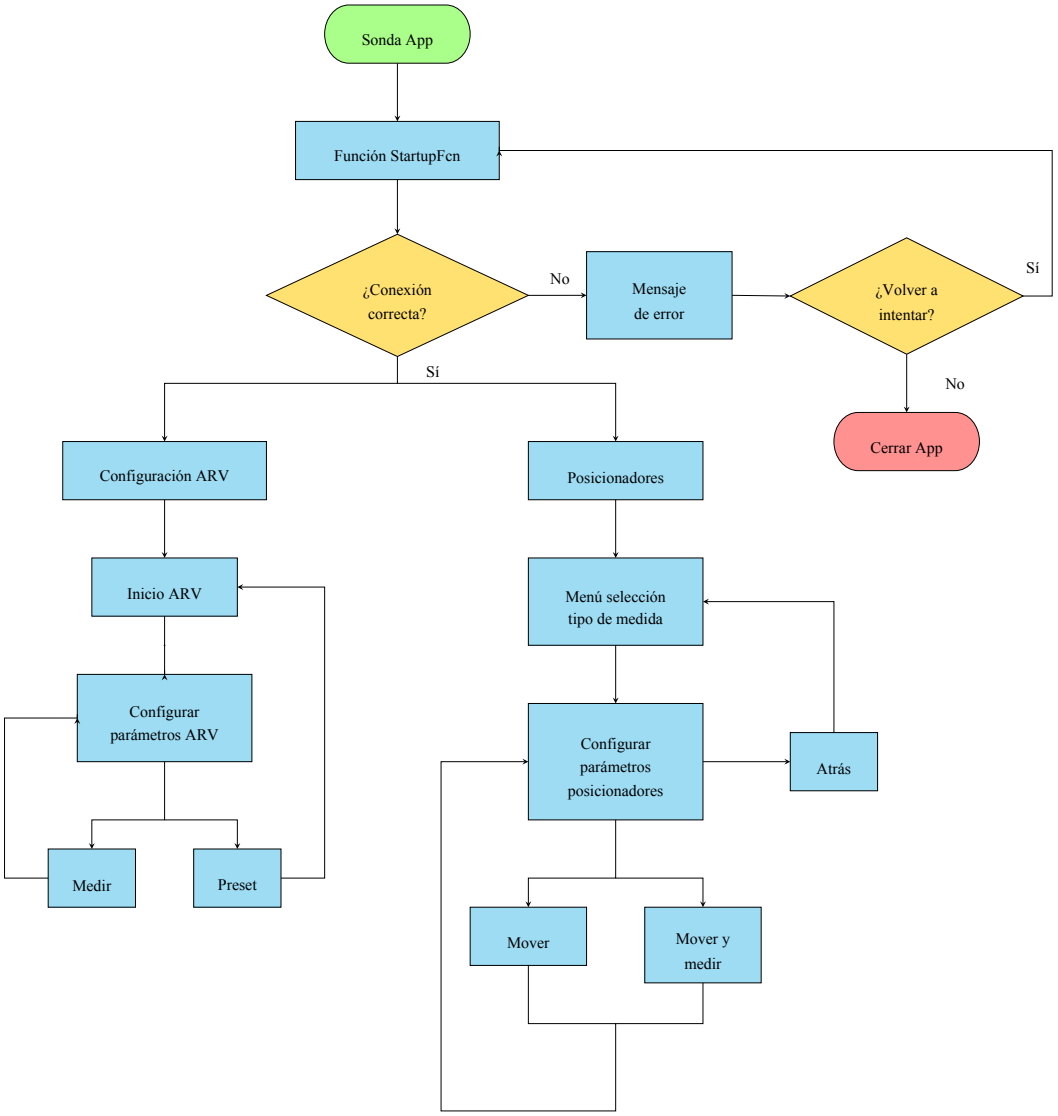


Figura 5.1: Diagrama de flujo de la aplicación *Sonda App*.

5.2. Conexión con el analizador de redes

Para crear una conexión con el analizador de redes, utilizaremos la herramienta `tmtool` (*Test & Measurement Tool*), que nos permite controlar instrumentos desde MATLAB.

Antes de realizar la conexión desde MATLAB, debemos configurar la conexión vía LAN a través de un cable Ethernet creando una Red de Área Local. Para ello seguiremos los siguientes pasos.

1. Conectamos el analizador a nuestro ordenador mediante el cable Ethernet.
2. En nuestro ordenador personal (sistema operativo Windows 10) nos dirigimos a Panel de control → Redes e Internet → Conexiones de red.
3. Hacemos clic derecho sobre la Conexión de área local y seleccionamos Propiedades.
4. Dentro de Propiedades buscamos el elemento Protocolo de Internet versión 4 (TCP/IPv4) y seleccionamos Propiedades.
5. Dentro de las propiedades de Protocolo de Internet introduciremos la IP manualmente, tal como se muestra en la Figura 5.2.

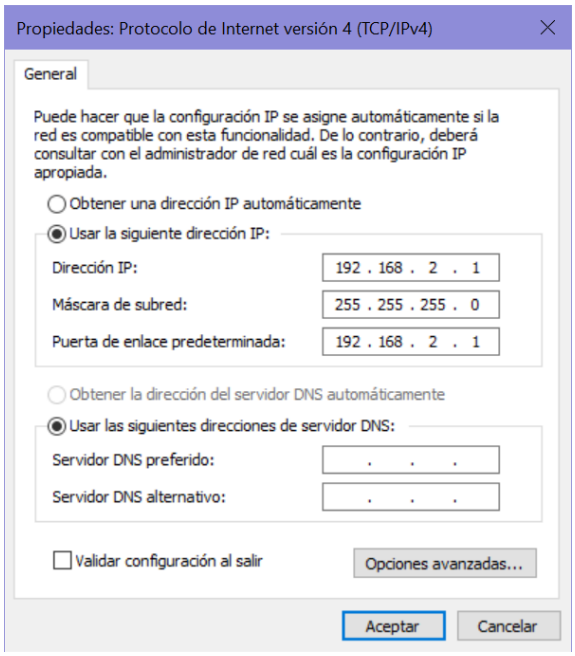


Figura 5.2: Configuración Red de Área Local en ordenador personal

6. Para el analizador seguiremos todo el proceso anterior y le asignamos la dirección IP inmediatamente posterior a la que habíamos puesto en nuestro ordenador, como se indica en la Figura 5.3.

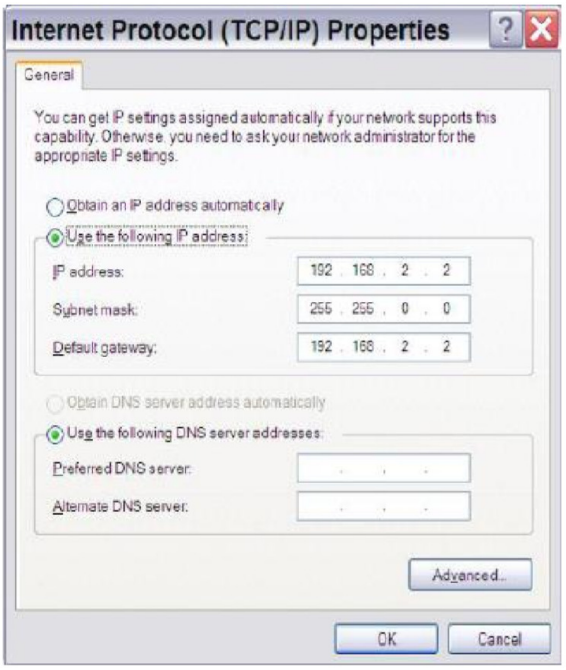


Figura 5.3: Configuración Red de Área Local en ARV

Una vez hecho esto abriremos la herramienta `tmtool` y nos posicionaremos sobre el menú de la izquierda y seleccionaremos Hardware → VISA → TCPIP(VXI-11) como se puede ver en la Figura 5.4, ya que realizaremos la conexión al analizador vía LAN con protocolo TCP/IP a través de un cable *Ethernet*.

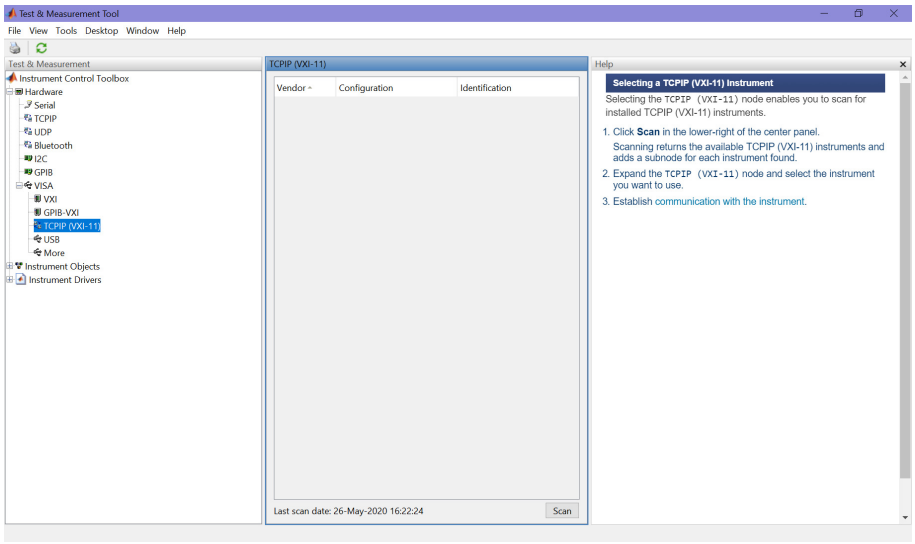


Figura 5.4: Herramienta `tmtool`

Con el analizador conectado al ordenador, pulsaremos el botón *Scan* situado en la parte inferior del panel central para que MATLAB encuentre nuestro instrumento. Una vez aparezca en pantalla lo seleccionaremos y le daremos a *Connect*. Automáticamente se crea un objeto para esta interfaz

donde podemos enviar instrucciones al analizador desde aquí y también ver el código con el que podemos crear un objeto con el que controlarlo desde *App designer* y conectarlo. El código de creación y conexión del analizador realiza lo siguiente:

1. Busca un instrumento de tipo VISA TCP/IP con *Resource Name*: TCPIP0::192.168.2.2::hislip0::INSTR.
2. Si el objeto ya estaba creado previamente no se vuelve a crear y se utiliza el anterior. Si por el contrario el objeto encontrado está vacío, se crea el objeto VISA-TCPIP.
3. Se crea la conexión con el instrumento.

```
% Find a VISA-TCPIP object.
app.ARV = instrfind('Type', 'visa-tcpip', 'RsrcName',
↪  'TCPIP0::192.168.2.2::hislip0::INSTR', 'Tag', '');

% Create the VISA-TCPIP object if it does not exist
% otherwise use the object that was found.
if isempty(app.ARV)
    app.ARV = visa('KEYSIGHT', 'TCPIP0::192.168.2.2::hislip0::INSTR');
else
    fclose(app.ARV);
    app.ARV = app.ARV(1);
end

% Connect to instrument object, ARV.
fopen(app.ARV);
```

5.3. Conexión con los posicionadores

El sistema de posicionadores está formado por un Controlador C4 [12] que se conecta al PC mediante un cable USB a puerto serie RS-232 de 9 pines. Este controlador C4 puede controlar dos sistemas MD-2 [13], mediante un cable de 40 pines hembra (salida de C4) a 36 pines macho (entrada de cada MD-2). Habrá un MD-2 por cada posicionador, por lo que uno de ellos controlará los dos motores del posicionador XY, y el otro el motor del posicionador X, tal como se ve en el esquema de la Figura 5.5:

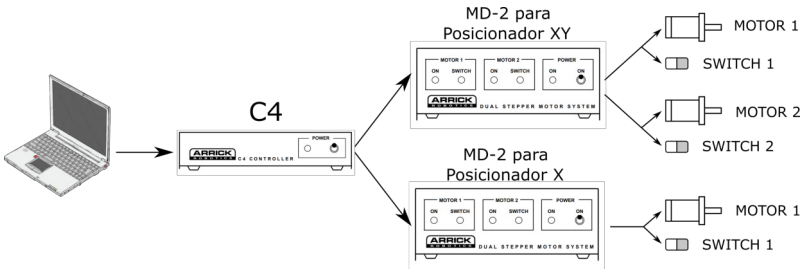


Figura 5.5: Conexión del sistema de control de los posicionadores

Una vez conectado todo el sistema, nos ayudaremos del sistema que proporciona el fabricante *Arrick Robotics* para comprobar que todo funciona correctamente y que podemos pasar a la programación de su funcionamiento.

Este software se llama *MD2xp Motor Control System*, y necesitaremos la versión 2.0 o posterior para poder operar con los MD2 a través del controlador C4. Este programa nos permite mover los motores en cualquier dirección y hacer que vuelban a su posición inicial. (Veáse la Figura 5.6)

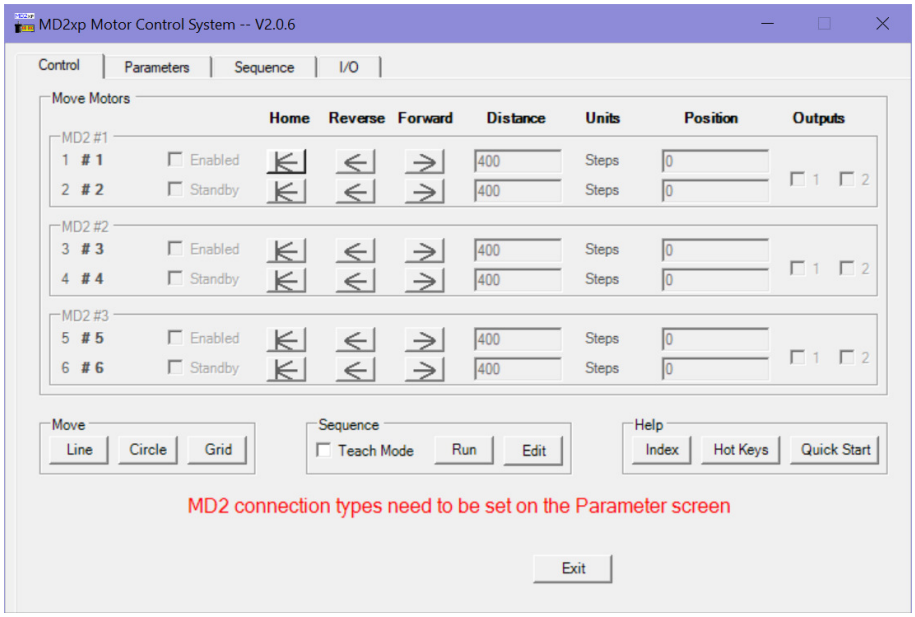


Figura 5.6: Software MD2xp Motor Control System

Antes de mover los motores, debemos indicar al programa que vamos a utilizar el controlador C4, para ello nos dirigiremos a la pestaña *Parameters* y pulsaremos sobre el campo de *Connection*, en *Click to set port*.

En la nueva ventana, llamada *MD2 Connection*, seleccionaremos que vamos a utilizar MD2#1 Y MD2#2 con la opción de conexión *Serial Port via C4* tal como se ve en la Figura 5.8 e indicaremos el puerto COM (que será el puerto USB que hayamos conectado), un identificador y el puerto de los motores.

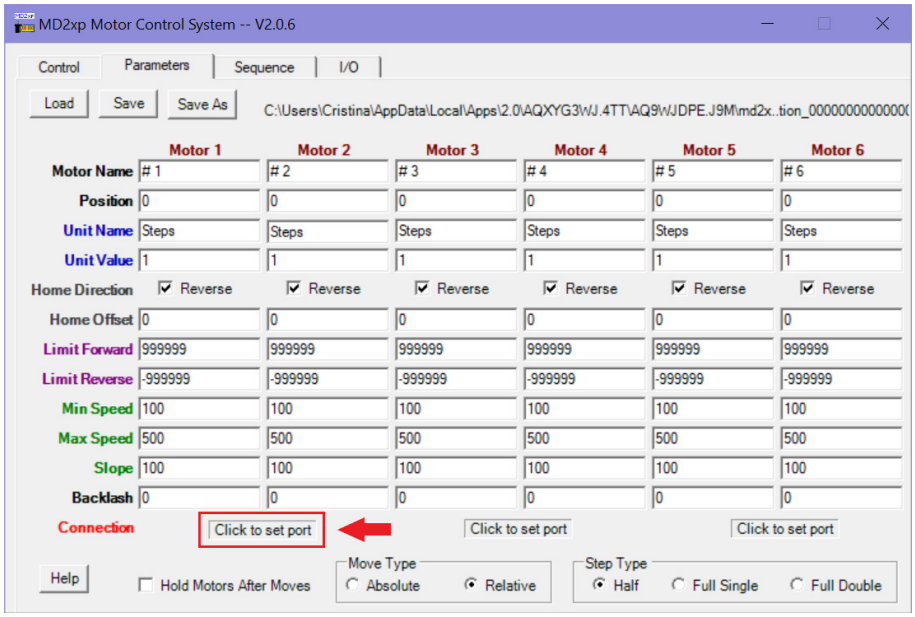


Figura 5.7: Configuración del sofware MD2xp

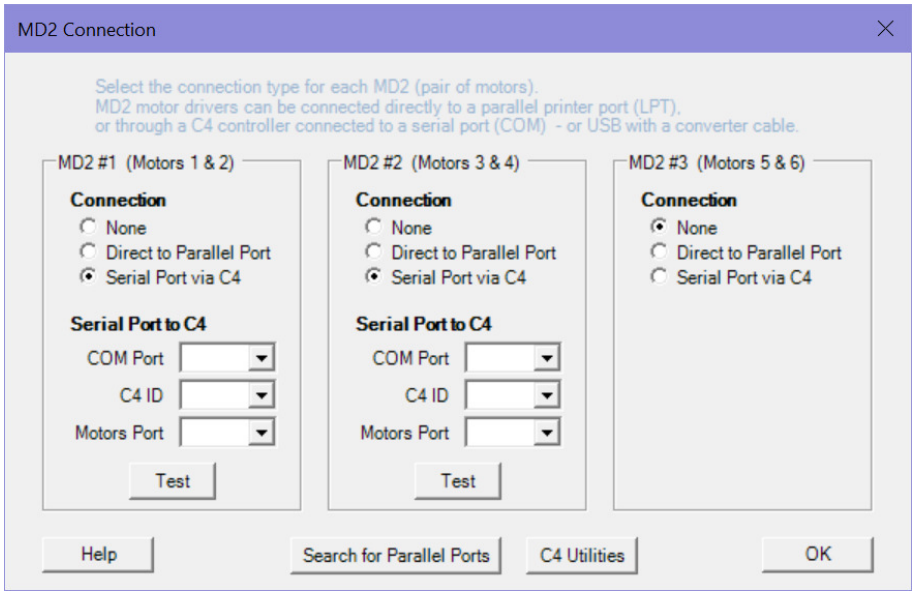


Figura 5.8: Ventana *MD2 Connection*

Si pulsamos el botón *C4 Utilities*, nos aparecerá una nueva venta donde podemos comprobar el funcionamiento del controlador C4, enviar comandos manualmente desde el cuadro de texto, cambiar el identificador y actualizar el *firmware* del C4.

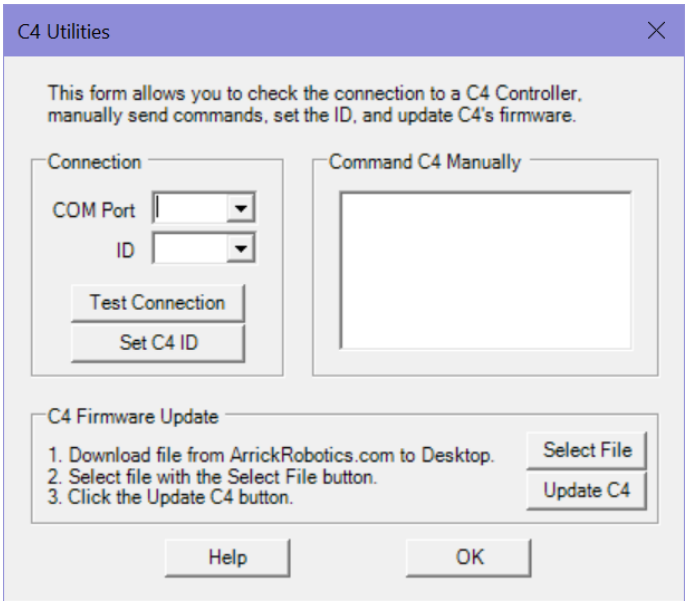


Figura 5.9: Ventana *C4 Utilities*

5.4. Diseño de la interfaz gráfica

5.4.1. Función Start-Up

Una vez abrimos el programa, lo primero que se ejecuta en el código es la función *Start-Up*. Está función inicializa algunas variables del programa además de realizar la conexión con el analizador. La conexión con este es vital en el programa ya que sin él no sería posible su uso. Por ello, se ha diseñado de tal manera que sólo se pueda acceder al programa si el analizador de redes está correctamente conectado al ordenador que estemos utilizando.

Para ello, se ha creado un bloque *try-catch* en el que se intente realizar la conexión con el analizador tal como se ha visto en el código del apartado 5.2. El código sería el siguiente:

```
% Connect to instrument object, ARV.
try
    fopen(app.ARV);
    % Reseteamos el analizador y ponemos todas las variables
    % predeterminadas
    Preset(app)
catch
    ErrorConexion(app)
end
```

Podemos observar que en el código se realiza la llamada a dos funciones: `Preset` y `ErrorConexion`. La primera función realiza varias cosas, ya que se encarga de inicializar la aplicación para su uso en lo referido al analizador. Para comunicarse con el analizador utilizamos los siguientes comandos:

```
fprintf(app.ARV,'instrucción')
```

```
dato = query(app.ARV,'pregunta')
```

El primero para enviar una instrucción o un dato al analizador y el segundo para preguntar al analizador sobre su estado o algún dato, como por ejemplo la potencia. Esta función `Preset`, por tanto, realiza lo siguiente:

1. Realiza un preset al analizador para que, cada vez que iniciemos el programa, vuelvan todos los parámetros a los que vienen por defecto. Esto se puede hacer con la siguiente instrucción:

```
fprintf(app.ARV, 'SYST:PRES');
```

2. Lo segundo que realiza es la creación de una nueva medida, en este caso del parámetro S_{11} que más tarde se podrá cambiar por el que desee el usuario. Para ello primero borramos las medidas existentes en el analizador y creamos una con el nombre que queramos. Después la seleccionamos y la mostramos por la pantalla del analizador.

```
fprintf(app.ARV, 'CALC:PAR:DEL:ALL');
fprintf(app.ARV, 'CALC:PAR:DEF:EXT "MyMeas", "S1_1"');
fprintf(app.ARV, 'CALC:PAR:SEL "MyMeas"');
fprintf(app.ARV, 'DISP:WIND:TRAC:FEED "MyMeas"');
```

3. Por último, está función recoge todos los datos del analizador que podemos editar en nuestra aplicación y los actualiza en esta. Así, cuando abramos el programa, podremos ver los datos reales que tiene en ese momento el analizador, y editar los que queramos. Por ejemplo para el *Edit Field* de la potencia de salida, haríamos lo siguiente:

```
s_pot = query(app.ARV, 'SOUR:POW?');
Potencia = str2double(s_pot);
app.PotenciadeSalidaEditField.Value = Potencia;
```

Por otro lado, tenemos la función `ErrorConexion`, que es llamada cuando el analizador no está conectado. Esta función crea un cuadro de diálogo de error en el que nos avisa que el analizador no está conectado, y nos da la opción de volver a intentar realizar la conexión o la opción Cancelar, que cerraría el programa por completo. Hasta que la conexión con el analizador no sea correcta, no podremos acceder a su configuración.

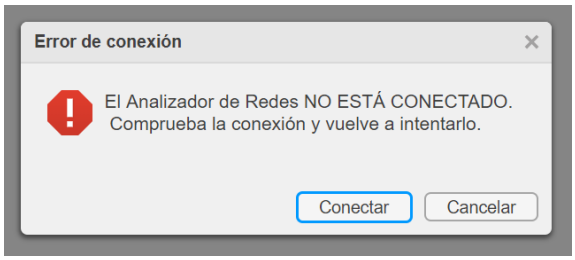


Figura 5.10: Cuadro de diálogo Error de Conexión

5.4.2. Control del analizador

Una vez el analizador está correctamente conectado, se nos mostraría la configuración del analizador de redes, que es la primera pestaña que encontramos en el programa. Este apartado de la interfaz permite controlar el analizador de manera independiente a los posicionadores, ya que de esta manera podremos realizar medidas con y sin ellos y dar una función extra al programa.

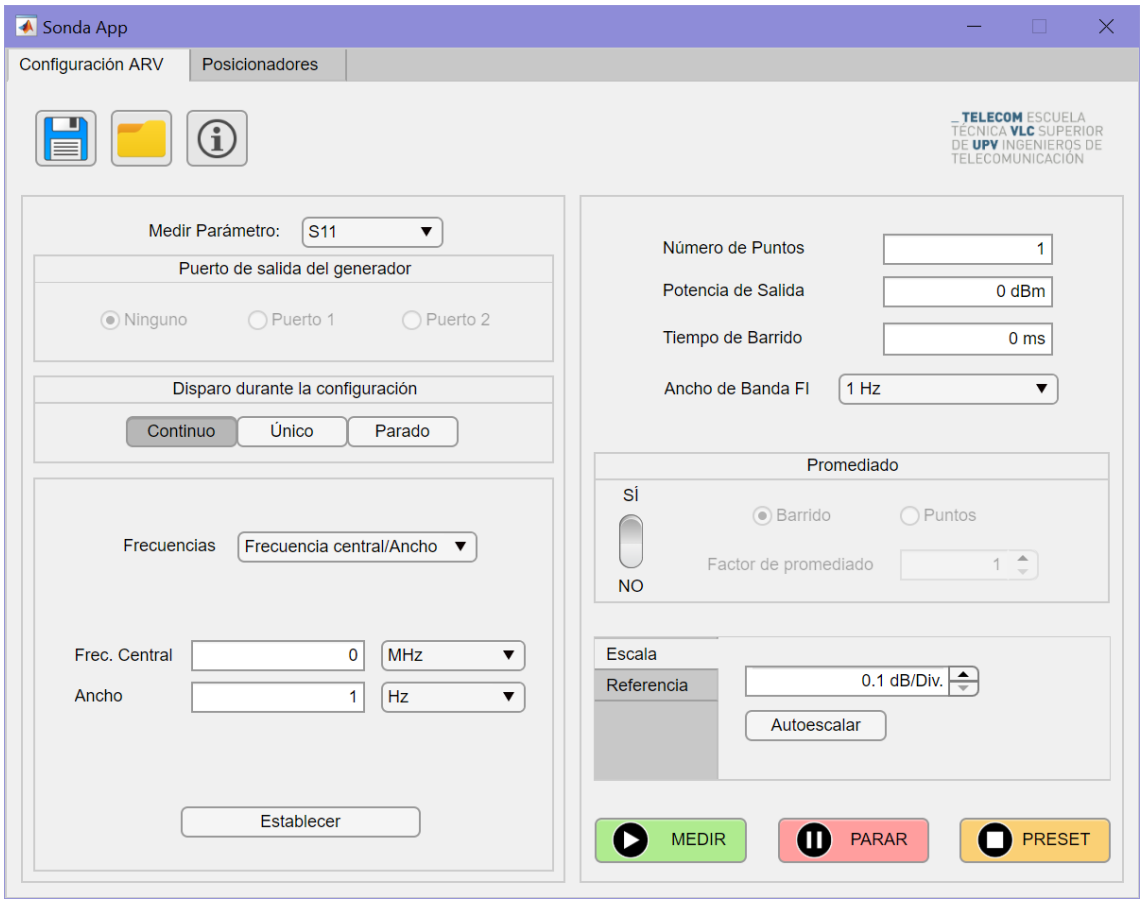


Figura 5.11: Configuración del ARV

5.4.2.1. Componentes de la interfaz

En este apartado se van a describir los componentes que forman el apartado de la configuración del ARV de la interfaz de manera detallada, su funcionamiento y algunos ejemplos de implementación de estos con código.

Botones superiores

En la parte superior a la izquierda tenemos tres botones, el primero para guardar la configuración, el segundo para cargar una configuración existente en el ordenador y el tercero para visualizar información de uso.

Figura 5.12: Botones superiores

1. **Guardar configuración.** Esta opción nos permite guardar una configuración de los parámetros del analizador, en nuestro ordenador, en un archivo `.mat`. Lo que permite esto es ver en cualquier momento los parámetros de esta configuración, ya que están guardados en una estructura.
   Para realizar esto, primero guardamos todos los parámetros en una estructura cada vez que pulsemos el botón de guardar, por lo que esto irá implementado en la retrollamada de este botón. Por ejemplo, si queremos guardar el parámetro seleccionado, usamos el siguiente código para guardarlo en una estructura llamada `configuracion`:

```
configuracion.ParametroDropDown = app.ParametroDropDown.Value;
```

   Lo segundo es abrir el explorador de archivos para guardar esta estructura dentro de una ruta y especificar un nombre (elegido por el usuario) con la extensión `.mat` y seguidamente guardarlo. Un ejemplo del código sería el siguiente:

```
% Creamos ventana de exploración de archivos para guardar la
% configuracion como .mat

[configuracion.filename,configuracion.path] =
↪  uiputfile('*.mat','Guardar Configuración ARV');

% Guardamos el archivo en la ruta especificada
newfile = fullfile(configuracion.path, configuracion.filename);
save(newfile,'configuracion');
```

2. **Cargar configuración.** De la misma manera que en el punto anterior, para cargar una configuración que tengamos guardada en nuestro ordenador, tendremos que abrir un explorador de archivos para seleccionar el archivo y la ruta donde se encuentra.

```
% Creamos ventana de exploración de archivos para cargar el
% archivo
[configuracion.filename,configuracion.path] =
↪  uigetfile('*.mat','Cargar Configuración ARV');

% Ruta del archivo
newfile = fullfile(configuracion.path, configuracion.filename);

% Cargamos el archivo
load(newfile,'configuracion');
```

   Una vez cargada la estructura, asignaremos los parámetros en sus correspondientes componentes, igual que antes.

```
app.ParametroDropDown.Value=configuracion.ParametroDropDown;
```

3. **Información de uso.** Si pulsamos este botón, simplemente aparecerá un cuadro de diálogo con varios puntos que indican cierta información o pautas de uso de la aplicación, tanto de la parte de la configuración del ARV como de los posicionadores.

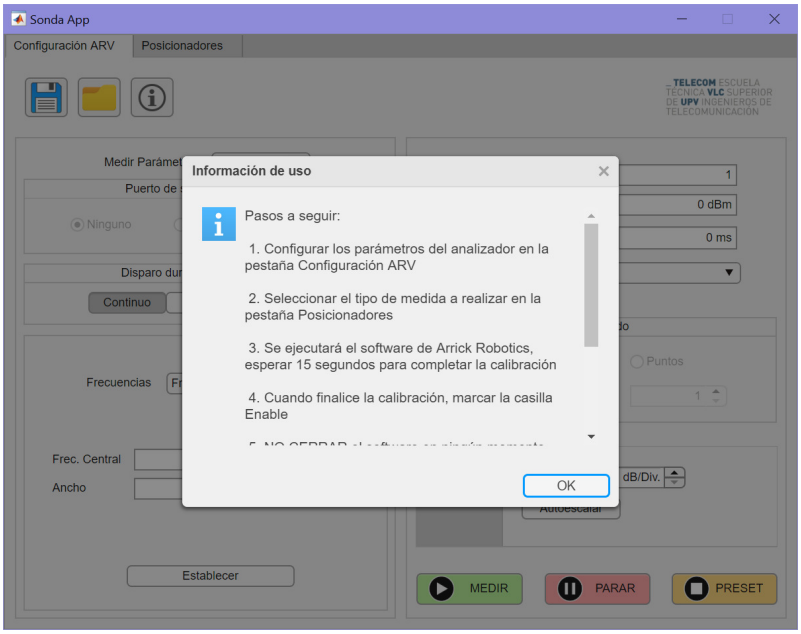


Figura 5.13: Ventana de información de uso

Selección de parámetro

Lo primero a la hora de configurar el analizador de redes, es elegir el parámetro que se desea medir. En este caso se han implementado todos los parámetros que permite pedir el analizador que disponemos en el laboratorio, que se muestran en la Figura 5.15.

Además, si elegimos un parámetro que no sea uno de los parámetros S, tendremos la opción de seleccionar el puerto de salida. Esta opción solo estará disponible al seleccionar uno de estos parámetros.

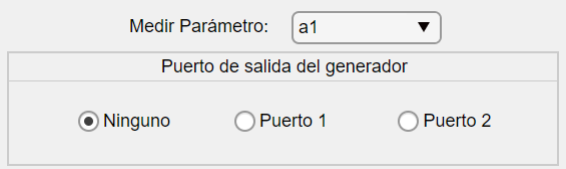


Figura 5.14: Puerto de salida del generador

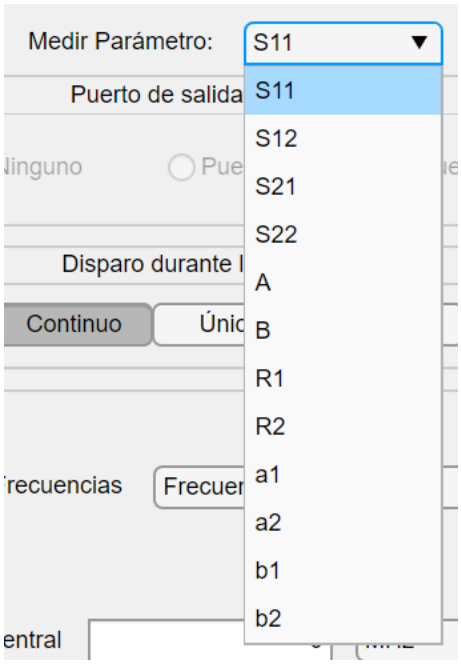


Figura 5.15: Selección de parámetro

Para cambiar de parámetro, por ejemplo al S_{21}, enviaremos la siguiente instrucción al analizador:

```
fprintf(app.ARV, 'CALC:PAR:MOD:EXT "S1_2"');
```

Disparo durante la configuración

Está opción se ha considerado implementarla debido a que esta aplicación se ha programado de manera que cuando uno de los parámetros sea cambiado por el usuario, el nuevo valor se envíe inmediatamente al analizador. Por tanto, podemos ver los cambios en el analizador a la vez que vamos cambiando la configuración desde el programa. Debido a esto, la idea de poder cambiar el disparo desde aquí nos resulta útil por si queremos cambiar de continuo a único, o a parado, mientras vamos configurando el analizador.

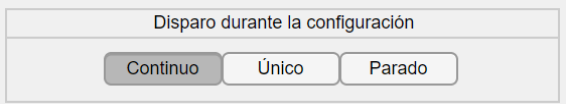


Figura 5.16: *Trigger* durante la configuración

Por ejemplo, para realizar un disparo único en el analizador, enviamos la siguiente instrucción desde MATLAB:

```
fprintf(app.ARV,'SENS:SWE:MODE SING');
```

Selección de frecuencias

Para la selección de frecuencias hemos habilitado tres opciones. Las dos primeras son para seleccionar un rango de frecuencias determinado y medir ahí. La tercera opción se trata de medir en modo de onda continua, en la cual mediríamos durante un tiempo determinado en una única frecuencia.

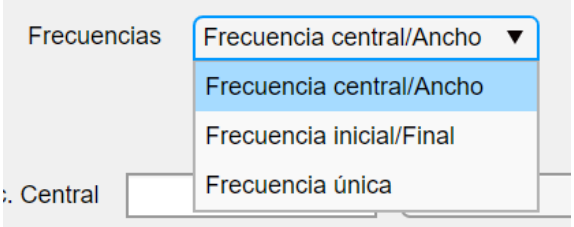


Figura 5.17: Selector de frecuencias

La primera opción se trata de indicar la frecuencia central y el ancho (*span*). Esta opción es útil si no tenemos claro la frecuencia inicial y la final donde queremos medir.

$$\begin{aligned} f_{\text{inicial}} &= \frac{f_c - \text{SPAN}}{2} \\ f_{\text{final}} &= \frac{f_c + \text{SPAN}}{2} \end{aligned} \tag{5.1}$$

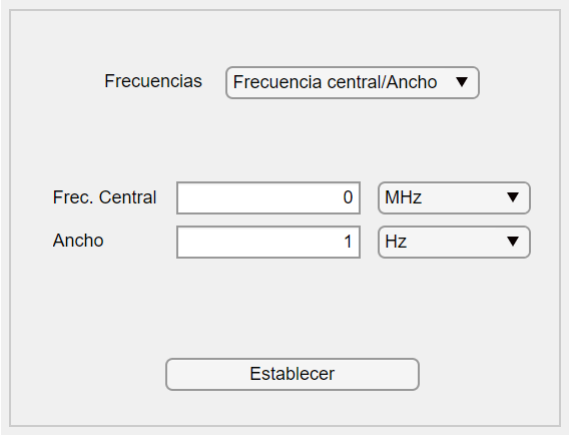


Figura 5.18: Frecuencia central y ancho

La segunda opción sería útil si ya tenemos claro el ancho de banda que queremos medir, por lo que solo tendríamos que introducir la frecuencia inicial y la final.

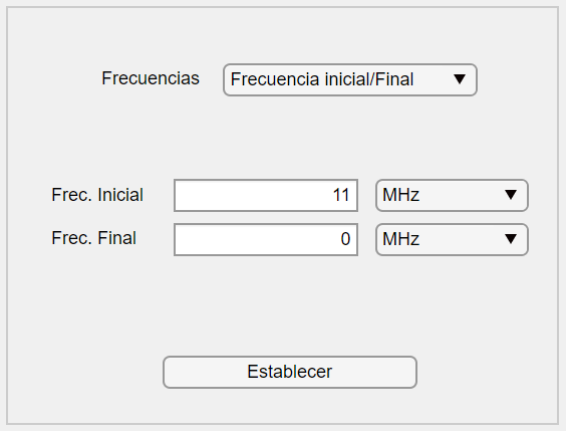


Figura 5.19: Frecuencia inicial y final

En estas dos opciones las frecuencias deben de estar entre los 10 MHz y los 67 GHz. Por lo que se han creado algunos cuadros de advertencia y error que aparecerán si introducimos algún dato incorrecto.

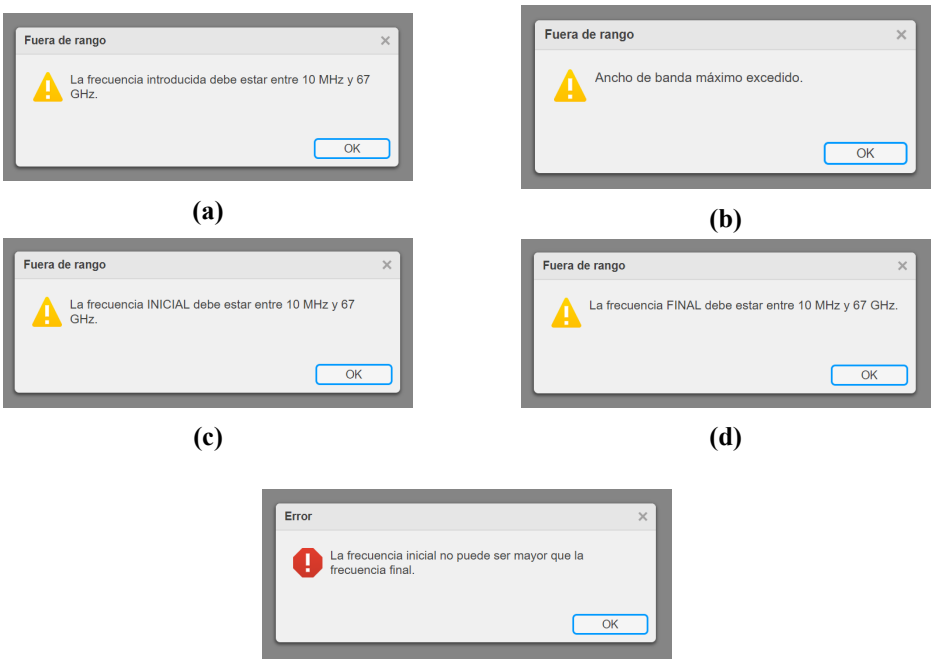


Figura 5.20: Cuadros de advertencia y error para la selección de frecuencias

Por último tenemos la opción de modo de onda continua, que nos permite medir a una única frecuencia durante un tiempo determinado. Aquí deberemos configurar tres parámetros: la frecuencia a la que queremos medir, el número de trazas, y el tiempo de medida.

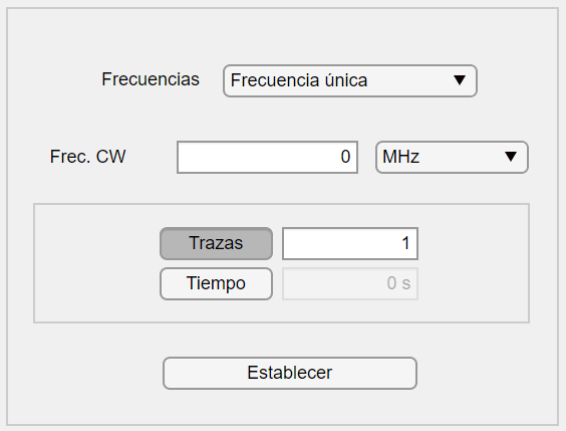


Figura 5.21: Modo onda continua

Hemos introducido un grupo de botones de selección en el que hay que elegir si queremos introducir el número de trazas o el tiempo de la medida, ya que uno depende del otro. Por tanto si cambiamos el valor de uno de ellos, el otro automáticamente cambiará de valor y se actualizará. En concreto, el número de trazas es igual al tiempo de la medida entre el tiempo de barrido. Así mismo, si cambiamos el tiempo de barrido, estos dos valores también se actualizarán si estamos realizando una medida en modo de onda continua.

El número de trazas es equivalente a cuantas veces iniciará un barrido o medida única (*Single*). El analizador nos permite realizar entre 1 y 65536 (2^{16}) trazas. Estos límites se han definido para el *Edit Field* númerico, por lo que si intentamos poner un número que no esté comprendido entre esos valores no lo aceptará como válido. Estos disparos se ejecutarán cuando pulsemos el botón Medir, que explicaremos más adelante. Otro límite es el referido al tiempo, este no debe ser menor que el tiempo de barrido, ya que si no no sería suficiente para realizar una traza, que dura el tiempo de barrido.

Una vez tenemos cualquiera de las tres opciones configuradas, pulsaremos el botón Establecer para enviar las instrucciones y los datos al analizador, que cambiarán automáticamente. La retrollamada de este botón es la función que se encarga de comprobar que los valores introducidos son los correctos, por lo que los cuadros de advertencia y error aparecerán una vez pulsemos este botón. Una vez los datos sean correctos, se establecerán las frecuencias elegidas en el analizador.

Número de puntos

El número de puntos ayuda a conseguir una mejor resolución de la traza que estemos midiendo. Pero por el contrario, el tiempo de barrido será mayor, y por lo tanto, irá más lento. Cuando llegamos a un punto en el que aumentamos este número y no conseguimos una diferencia importante o significativa, estaremos ante el número de puntos óptimo. El analizador permite como máximo 100001 puntos, por lo que se ha establecido el límite entre 1 y 100001.

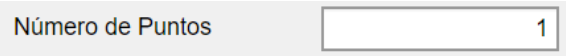


Figura 5.22: Número de puntos

Por ejemplo, si quisiéramos establecer 1000 puntos en el analizador, enviaríamos la siguiente instrucción:

```
fprintf(app.ARV, 'SENS:SWE:POIN 1000');
```

Una vez cambiamos el número de puntos, el tiempo de barrido cambiaría, por lo que hay que enviar una pregunta al analizar sobre su nuevo valor de tiempo de barrido, y actualizarlo en la aplicación.

Potencia de salida

El analizador admite una potencia de salida de entre -30 dBm y 30 dBm (La potencia de salida predeterminada en el analizador es de 0 dBm). Por lo que en este *Edit Field* numérico tiene estos dos límites y solo admite valores dentro de este rango.

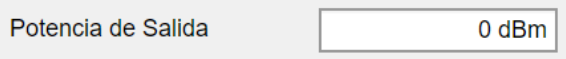


Figura 5.23: Potencia de salida

Por otra parte, ya en proyectos anteriores se ha determinado que no se obtenía una respuesta balanceada con algunos valores de potencia de salida muy grandes. Desde 10 MHz hasta 38 GHz el analizador admite una potencia de salida máxima de 13 dBm, y desde 38 GHz hasta 67 GHz, máximo 10 dBm. Estos valores son simplemente una recomendación para el correcto funcionamiento, por lo que la aplicación no nos prohibirá ponerlos mayores, pero nos aparecerá un cuadro de información donde nos indicará que se ha superado el valor de potencia recomendado y nos preguntará si queremos cambiarlo a este valor o dejar el que hemos introducido.

(a) **(b)**

Figura 5.24: Cuadros de información para la potencia de salida

Todas estas comprobaciones las hace la retrollamada de este objeto, además para cada selección de frecuencias del apartado visto anteriormente. Como ejemplo, si queremos enviar al analizador que tenga una potencia de salida de 10 dBm, enviaremos la siguiente instrucción:

```
fprintf(app.ARV,'SOUR:POW 10');
```

Tiempo de barrido

Como hemos dicho antes, el tiempo de barrido cambia automáticamente al introducir el número de puntos, y por tanto ese valor del tiempo de barrido será el mínimo necesario para funcionar con ese número de puntos. Podremos introducir un número mayor al que establece el analizador, pero nunca un número menor. Si intentamos introducir un número menor, no enviará ninguna instrucción y volverá a aparecer el número que había anteriormente. El valor que introducimos debe estar en milisegundos.

El ancho de banda IF también afectará al tiempo de barrido. Según el ancho de banda IF se estreche, el tiempo de barrido aumentará, y hay que tener esto en cuenta.

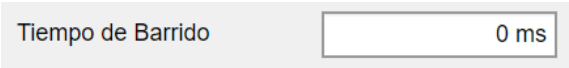


Figura 5.25: Potencia de salida

Por ejemplo, si queremos introducir un tiempo de barrido de 10 ms, enviaremos la siguiente instrucción:

```
fprintf(app.ARV,'SENS:SWE:TIME 10ms']);
```

Ancho de banda IF

Este parámetro cambia el ancho de banda del filtro paso banda de frecuencia intermedia. Si reducimos este ancho de banda reduciremos también el efecto del ruido (NF), aunque para otros casos puede interesar tener un valor alto, como el de medir potencia en una banda de frecuencias o el ruido.

El analizador del que disponemos permite introducir ciertos valores determinados de ancho de banda, desde 1 Hz hasta 15 MHz. Todos ellos se han establecido en el desplegable que se puede ver a continuación.

Si seleccionamos un ancho de 150 Hz por ejemplo, se enviará la siguiente instrucción al analizador:

```
fprintf(app.ARV, 'SENS:BWID 150HZ');
```

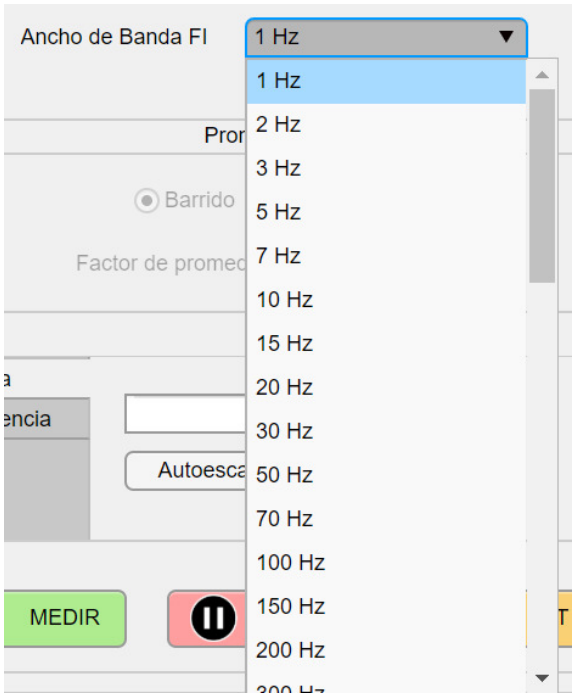


Figura 5.26: Ancho de banda IF

Promediado

El promediado nos ayuda a reducir el ruido (NF) en las medidas. El analizador calcula cada punto de datos en función del promedio de varias medidas. Se determina el número de medidas configurando el factor de promediado. Cuanto mayor sea este, mayor será la reducción de ruido. Si queremos activarlo a la hora de medir solo hay que presionar en el interruptor que hay a la izquierda, ya que por defecto viene desactivado. Si el interruptor está desactivado, no podremos acceder a cambiar ningún valor. Como máximo, el analizador nos permite introducir un factor de promediado de 65536 (2^{16}).

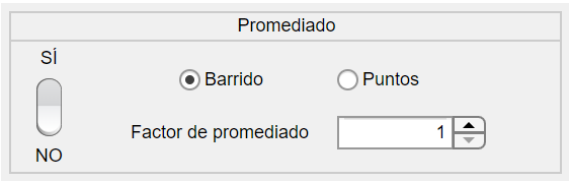


Figura 5.27: Promediado

Dentro de este tenemos dos opciones, realizar un promediado por barrido o por puntos. En el promediado por barrido, cada punto se forma a partir de la media del mismo punto de los barridos anteriores. Cuando el número de barridos es igual al factor de promediado, el promedio sigue la siguiente fórmula, donde n es el factor de promediado [14]:

$$\text{Nueva media} = \frac{\text{Nuevo dato}}{n} + \text{Antigua media} \cdot \left(\frac{n-1}{n}\right) \tag{5.2}$$

En el promediado por puntos, primero se mide cada punto tantas veces como indique el factor de promediado y se realiza el promedio de estos. Una vez hecho esto, pasa al siguiente punto y así hasta finalizar.

Para activar o desactivar el promediado, se envían las siguientes instrucciones:

```
fprintf(app.ARV,'SENS:AVER OFF');
fprintf(app.ARV,'SENS:AVER ON');
```

En el caso de desactivarlo, está instrucción se enviará cuando desactivemos el interruptor. Pero en el caso de activar el promediado, se enviará cuando pulsemos el botón de MEDIR, ya que al enviar esta instrucción el analizador empieza inmediatamente a medir.

También, para indicar el tipo de promediado, por barrido o por puntos, habrá que enviar la selección al analizador:

```
fprintf(app.ARV,'SENS:AVER:MODE SWEEP');
fprintf(app.ARV,'SENS:AVER:MODE POINT');
```

Y si por ejemplo, indicamos que queremos un factor de promediado de un valor 30, se enviaría lo siguiente:

```
fprintf(app.ARV,'SENS:AVER:COUN 10');
```

Escala

Muchas veces la medida se descentra de la pantalla, o por el contrario queremos indicar otra escala al analizador para situarla bien en la pantalla. Por eso se ha habilitado un apartado de la aplicación a la escala.

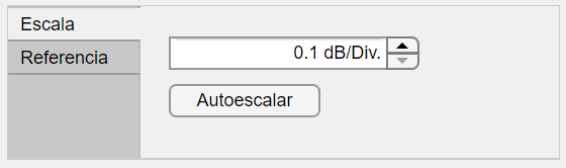


Figura 5.28: Escala

Tenemos varias funciones, al igual que tiene el analizador, una función de autoescalar. Para ello solo tenemos que pulsar el botón y se enviará la siguiente instrucción:

```
fprintf(app.ARV,'DISP:WIND:Y:AUTO');
```

También podemos cambiar manualmente los valores de la escala. El analizador puede mostrar hasta un rango de 6000 dB, desde -3000 dB hasta 3000 dB, con un máximo de 500 dB por división. Entre estos valores, podemos cambiar el número de dB por división, con un límite de entre 0.1 dB/Div.

y 500 dB/Div., implementados ya en el *spinner* asociado a este valor.

Si cambiamos la pestaña a Referencia, aquí podemos cambiar el nivel de referencia, que puede ir desde -3000 dB hasta 3000 dB, y la posición de referencia, que va desde 0 a 10 divisiones.

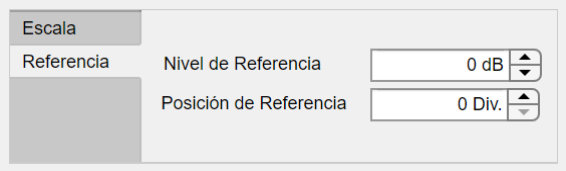


Figura 5.29: Referencia

Estos tres valores se pueden establecer en el analizador con las siguientes instrucciones, por ejemplo para 10 dB/Div., un nivel de referencia de 0 dB y una posición de referencia de 2 divisiones.

```
fprintf(app.ARV,'DISP:WIND:TRAC:Y:PDIV 10');
fprintf(app.ARV,'DISP:WIND:TRAC:Y:RLEV 0');
fprintf(app.ARV,'DISP:WIND:TRAC:Y:RPOS 2');
```

Medir, Parar y Preset

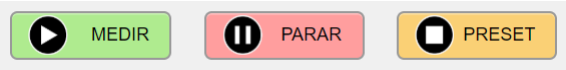


Figura 5.30: Botones MEDIR, PARAR y PRESET

El botón MEDIR tiene la función más extensa de todas, ya que este botón tiene que comprobar que tipo de medida estamos haciendo, comprobar si hay promediado o no, y a partir de ahí ejecutar un código distinto.

Para guardar las medidas se han creado una serie de directorios en el analizador que se dividen según el tipo de medida que se trate, ya que guardarlas todas en el propio ordenador llevaría tiempo innecesario de volcado de datos. El analizador tiene un puerto USB desde el que se pueden pasar las medidas al ordenador, por lo que ya tenemos esa opción si quisiéramos tener algunas de ellas.

Además, se ha creado un cuadro que aparece cuando pulsamos el botón medir. Desde aquí podemos poner el nombre que deseemos a la medida o indicar que se asigne un nombre automáticamente. El nombre automático sería de la siguiente forma, donde aparecería CW al final (acompañado del número de la traza) si se tratase de este tipo de medida:

Parámetro_Fecha_Hora(_CW_trazaX)

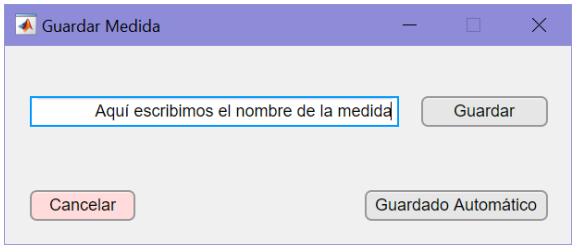


Figura 5.31: Guardar medida

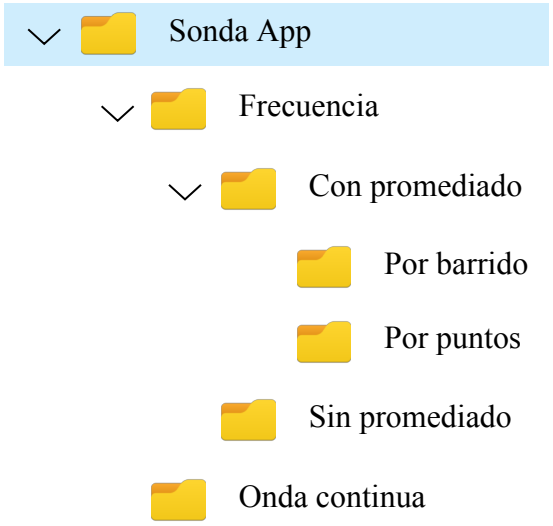


Figura 5.32: Directorios del analizador

Por tanto, consideramos 4 tipos de medidas, donde cada una se ha programado de la siguiente manera:

1. **Medida en frecuencia sin promediado.** En este tipo de medida únicamente se guarda lo que se esté mostrando en ese momento por pantalla.

2. **Medida en frecuencia con promediado por barrido.** Aquí tendríamos que tener en cuenta cuanto tarda en realizarse el promediado para poder guardar la medida definitiva. Para ello utilizamos el factor de promediado. Antes de comenzar a medir debemos hacer un *clear* de los promediados anteriores de otra medida (sólo para el caso de promediado por barrido). Si enviamos un *query* al analizador preguntando el estado del barrido (*sweep*), podremos saber si esta en modo *hold*. Esto nos interesa ya que en base al factor de promediado haremos tantos disparos al *trigger* como indique este número, todo ello dentro de un bucle que compruebe que después del último disparo el barrido esté en modo *hold*. En ese momento podremos guardar la medida.

3. **Medida en frecuencia con promediado por puntos.** Igual que en el modo anterior, tenemos en cuenta el factor de promediado introducido por el usuario a la hora de guardar la medida.

En este tipo de promediado es más sencillo saber cuando se ha acabado de medir, ya que como va punto por punto, el tiempo de barrido cambia. Únicamente hay que volver a preguntar al analizador cual es el nuevo tiempo de barrido y indicar a MATLAB que hay que esperar dicho tiempo y después guardar la medida. Si el usuario cambiará este tiempo de barrido a un número mayor, esperaríamos este último.

4. **Medida en onda continua.** En este modo, hay que guardar cierto número de trazas medidas. Como sabemos el tiempo de barrido por traza, le indicamos a MATLAB que tiene que esperar dicho tiempo para que se genere una traza y guardarla antes de que se haga el siguiente disparo del *trigger* para no solapar las medidas.
   A todo esto hay que añadirle el tiempo que se tarda en ejecutar la instrucción de guardar la medida en MATLAB, que se ha comprobado que es unos 8 ms, pero se han establecido 10 ms para tener cierto margen.

A continuación tenemos el botón PARAR, que solo está disponible si el botón MEDIR ha sido pulsado y por lo tanto se está midiendo en ese momento. La retrollamada de este botón únicamente envía la instrucción al analizador de que se pase a estado *hold* y por tanto se pare la medida que se estaba efectuando.

Por último, el botón PRESET únicamente hace una llamada a la función `Preset(app)`, ya explicada con detenimiento en el apartado 5.4.1. Esta función resetea todo lo que hay en ese instante en el analizador y en la aplicación.

5.4.3. Control de los posicionadores

Debido a la situación actual con el COVID-19 no se ha podido asistir al laboratorio para realizar la programación de los posicionadores con la aplicación ni hacer pruebas con estos. Por ello se ha decidido hacer la parte visual de la interfaz (que no precisa de pruebas y es posible hacerla desde casa) y retomar la programación de esta una vez empiece el siguiente curso. La finalización del programa formará una parte del TFM, tal como se explica en el apartado de "Propuesta de trabajo futuro". A continuación se explica que elementos se han introducido para el control de los posicionadores y hacer medidas con ellos y cuál sería su funcionamiento.

La siguiente funcionalidad del programa es poder controlar unas mesas posicionadoras para hacer medidas de distintos tipos: SISO, SIMO, MISO y MIMO. Por ello, lo primero que vemos al acceder a la pestaña de Posicionadores es un menú con cuatro botones donde podemos elegir el tipo de medida que queremos realizar. Nada más elegir un tipo de medida, se ejecutaría el software de *Arrick Robotics*, que realizaría una calibración de los posicionadores.

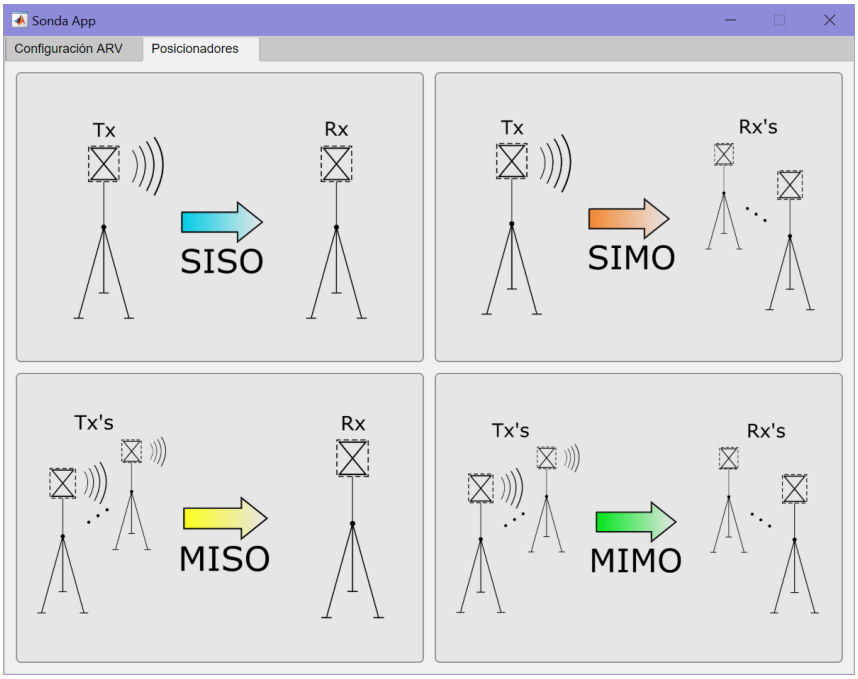


Figura 5.33: Menú principal Posicionadores

5.4.3.1. Componentes de la interfaz

En este apartado se va a proceder a exponer los distintos componentes que forman este apartado según el tipo de medida, ya que cada sub-menú dispone de elementos distintos.

Medida SISO

Una vez pulsamos sobre la opción de realizar medidas SISO, nos aparece directamente la pantalla donde podemos configurar los posicionadores y realizar medidas con ellos.

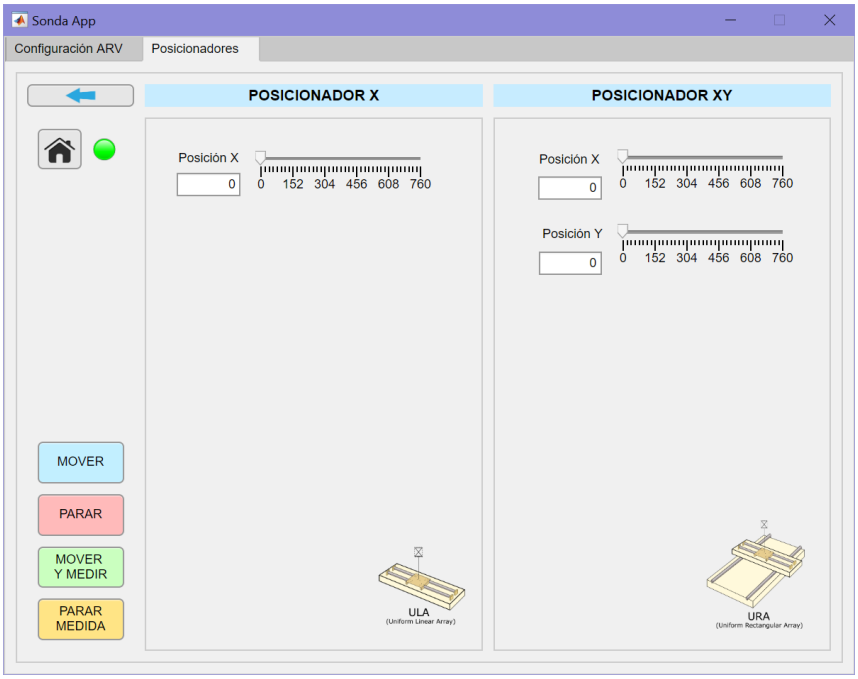


Figura 5.34: Pantalla para medidas SISO

En este tipo de medidas no hace falta el uso de los posicionadores, únicamente sería necesaria una antena sobre un trípode. Pero en este caso se ha querido dar la posibilidad de utilizar los posicionadores y poder situar la antena en cualquier posición de estos para realizar una medida SISO. Por tanto, el uso de la configuración de éstos posicionadores solo estaría accesible si se detecta que éstos están conectados, si no estas opciones no serían editables.

Como se puede ver, tenemos la configuración del posicionador X y la del posicionador XY tal como se aprecia en las imágenes inferiores a la vez.
En los dos, vemos unos deslizadores con los que se puede cambiar la posición de cada uno. Para el posicionador X únicamente necesitamos uno ya que solo se moverá en un eje, así como para el XY necesitamos dos, ya que este posicionador se mueve en el eje X y en el Y.

Los motores con los que movemos estos ejes pueden moverse hasta 760 pasos desde la posición inicial hasta el final del eje. Por ello, cada deslizador va desde la posición 0 hasta la 760. El valor del deslizador en el que estemos aparecerá en el cuadro numérico que hay al lado de ellos. También se podrá cambiar directamente el valor de la posición en el cuadro numérico.

Además de eso, vemos en la parte de la izquierda de la aplicación varios botones. Todos ellos son comunes a cualquier tipo de medida que hayamos seleccionado. El botón superior con una flecha nos permite volver al menú principal y cambiar el tipo de medida. El botón que tiene justo debajo con el icono de una casa, hace que todos los posicionadores conectados en ese instante vuelvan a su posición inicial. Una vez estén en la posición inicial, el indicador estará en color verde, y si están en otra posición estará en color rojo.

Los botones que vemos más abajo se encargarán del control de los posicionadores y de realizar las medidas. El botón MOVER desplazará los posicionadores a la posición indicada según el usuario o hará que el posicionador se vaya moviendo según la rejilla definida por el usuario (explicado más adelante). El botón PARAR únicamente parará el movimiento de los motores en el instante que

lo pulsemos. El botón MOVER Y MEDIR unirá el movimiento de los posicionadores con todas las funcionalidades del botón MEDIR de la pestaña de configuración del ARV. En este caso, si estamos utilizando las mesas, en el nombre con el que se guardaría automáticamente la medida vendría indicado el tipo de medida y la posición de la mesa o mesas. Por último, el botón PARAR MEDIDA pararía primero la medida, y después los motores.

Medida SIMO

A la hora de realizar una medida SIMO , nos pregunta antes que mesa queremos utilizar como receptor, tal y como vemos en la siguiente imagen. Podemos verlo fácilemente con los dibujos empleados.

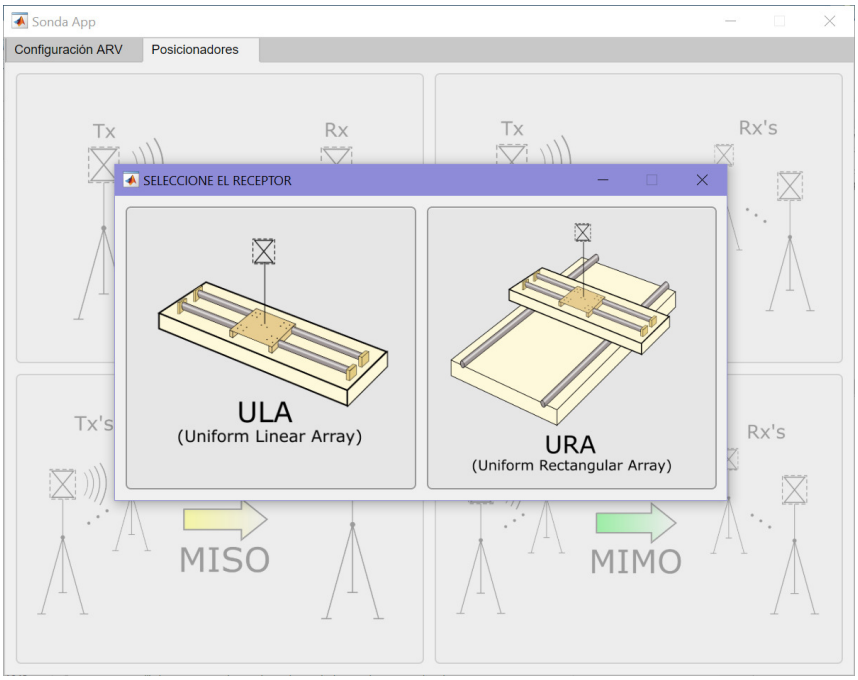


Figura 5.35: Selección del receptor

Si hemos seleccionado el posicionador X como receptor, tendremos la siguiente pantalla de configuración. En ella podemos ver varias diferencias. Lo primero es que como transmisor podremos usar el posicionador XY si es que lo hemos conectado, tal como con las medidas SISO. La novedad está en que para el posicionador X podemos definir una rejilla por la que moverse y por tanto crear varias medidas en cada posición del receptor. Para configurar está rejilla en la mesa lineal es más sencillo, ya que solo tiene un eje. Hay que indicar el número de filas que queramos que tenga el eje, y la separación entre cada fila en mm (máximo 762 mm). El deslizador se ajustará automáticamente a los valores que hayamos introducido (realizando un redondeo) y nos irá indicando por donde se encuentra el posicionador mientras se va realizando la medida. Con estos datos se le indicará al posicionador en que posiciones debe moverse para ajustarse a la rejilla que hayamos creado. También podemos indicar el sentido del barrido del posicionador, que podrá ser inverso si el posicionador ha llegado a la posición final y queremos medir en el sentido opuesto.

Para el cálculo de la posición inicial y final del posicionador x, se utilizan las siguientes ecuaciones,

donde M es el número de filas y Δx es la separación entre filas:

$$x_{inicial} = \frac{760 - (M-1)\Delta x}{2} \tag{5.3}$$

$$x_{final} = x_{inicial} + (M-1)\Delta x \tag{5.4}$$

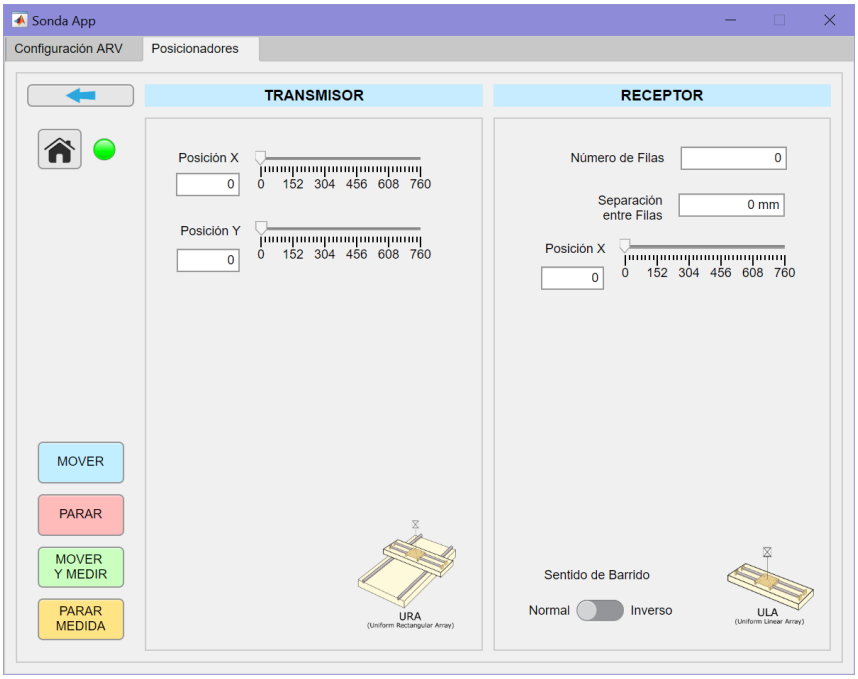


Figura 5.36: Configuración SIMO con posicionador X

Por el contrario, si hemos seleccionado el posicionador XY como receptor, nos encontraremos con una configuración distinta para este.

Igual que en los casos anteriores, podremos utilizar la otra mesa como transmisor si está conectada, pero sin moverse (ya que si no no sería una medida SIMO). Para el receptor hay que configurar una rejilla, esta podrá ser rectangular o circular.

Para una rejilla rectangular, tendremos que crear una matriz similar a la de la Figura 5.38. Para ello habrá que introducir el número de filas y columnas (ya que ahora tenemos eje X e Y en la mesa) y también la separación entre ellas en mm.

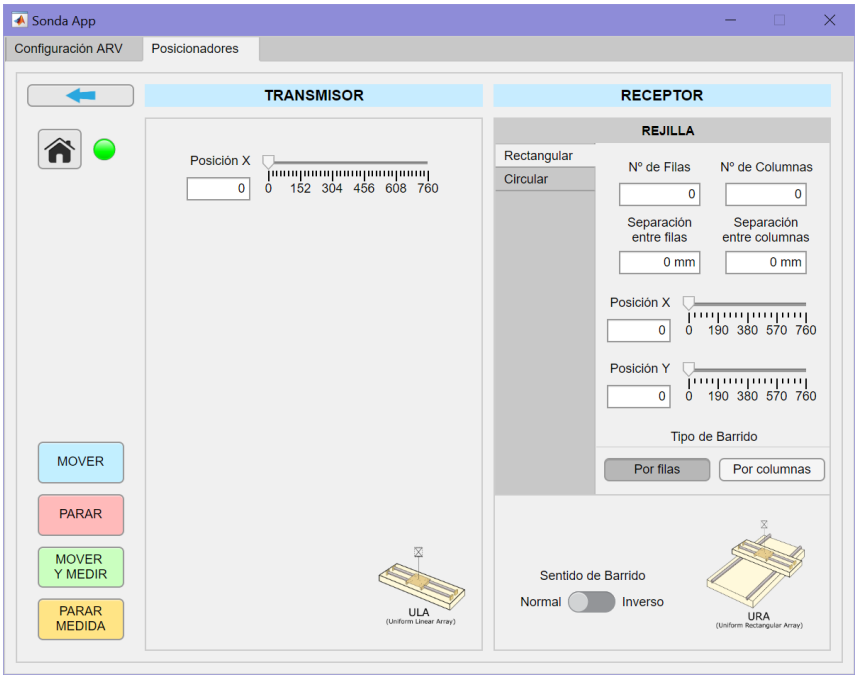


Figura 5.37: Configuración SIMO con posicionador XY

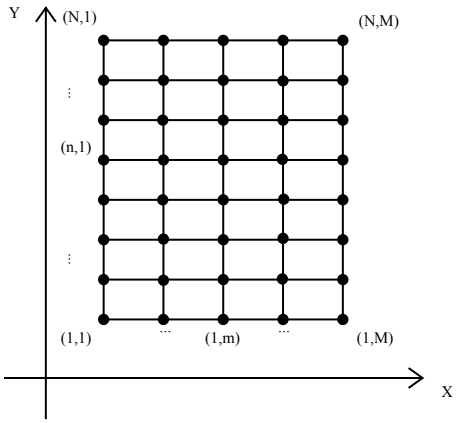


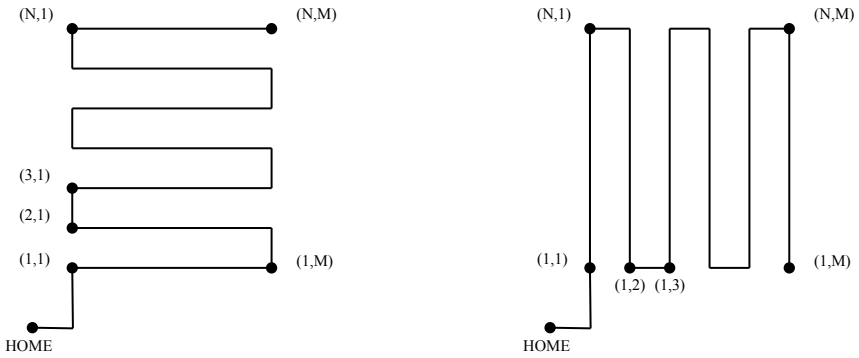
Figura 5.38: Matriz para el posicionador XY

Las ecuaciones para el cálculo de la posición inicial y final de cada eje son las siguientes:

$$\begin{aligned} x_{\text{inicial}} &= \frac{760 - (M-1)\Delta x}{2} \\ y_{\text{inicial}} &= \frac{760 - (N-1)\Delta y}{2} \end{aligned} \tag{5.5}$$

$$\begin{aligned} x_{\text{final}} &= x_{\text{inicial}} + (M-1)\Delta x \\ y_{\text{final}} &= y_{\text{inicial}} + (N-1)\Delta y \end{aligned} \tag{5.6}$$

Además, tenemos que indicar queremos un barrido por filas o por columnas. Una descripción gráfica de cada tipo de barrido a lo largo de la matriz sería la siguiente:



(a) Barrido por filas **(b) Barrido por columnas**

Figura 5.39: Tipos de barrido para el posicionador XY

Por último e igual que anteriormente, podemos indicar que queremos el barrido en sentido inverso, que se habilitará una vez el recorrido del posicionador haya llegado a la posición (N,M) que es la final, y podremos medir haciendo el recorrido inverso.

Otra opción es configurar una rejilla circular para el posicionador XY. Si seleccionamos la pestaña Circular, vemos lo siguiente.

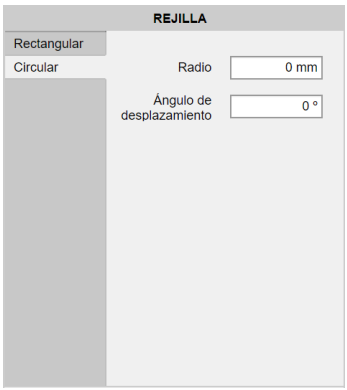


Figura 5.40: Configuración rejilla circular

El usuario únicamente debe de introducir el radio de la circunferencia (que no debe ser mayor a 381 mm) y el ángulo de desplazamiento en esa circunferencia en mm. Si consideramos que el centro de la mesa está en la posición (380,380) y que α es igual a un vector de 0 a 2π con un paso del ángulo introducido, los cálculos para cada posición X e Y son los siguientes (veáse la Figura 5.41):

$$\begin{aligned} \Delta x &= r \cdot cos(\alpha) \\ \Delta y &= r \cdot sen(\alpha) \end{aligned} \tag{5.7}$$

$$\begin{aligned} x_{\text{inicial}} &= 380 + r \cdot cos(0^o) = 380 + r \\ y_{\text{inicial}} &= 380 + r \cdot sen(0^o) = 380 \end{aligned} \tag{5.8}$$

Por tanto, la posición irá variando según el desplazamiento del ángulo:

$$\begin{aligned} x_i &= x_{i-1} + \Delta x_i \\ y_i &= x_{y-1} + \Delta y_i \end{aligned} \tag{5.9}$$

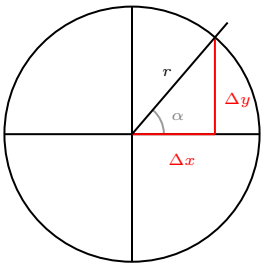


Figura 5.41: Representación del desplazamiento en una circunferencia de radio r

Medida MISO

Las medidas MISO se configurarían igual que las SIMO, pero en este caso elegiremos una de las mesas como transmisor, por lo que lo primero que nos pregunta es que mesa queremos utilizar para el transmisor.

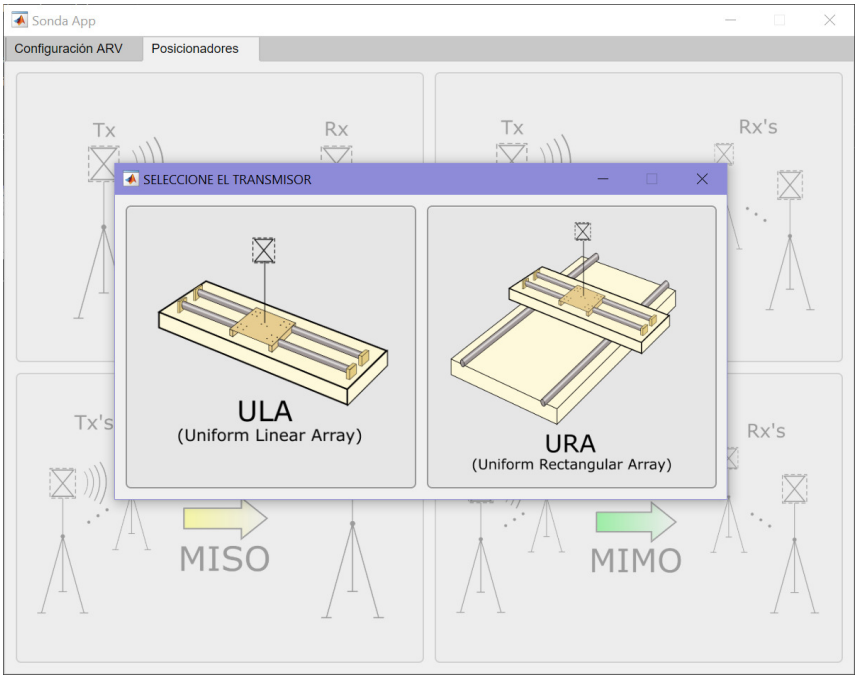


Figura 5.42: Selección del transmisor

Por lo demás, la configuración para los dos tipos de transmisores sería exactamente igual a la de SIMO.

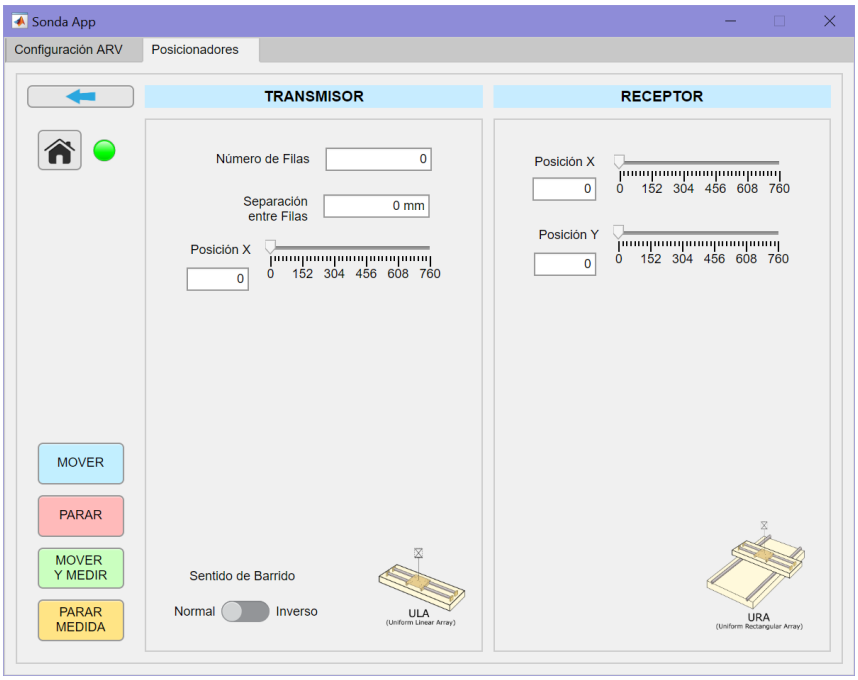


Figura 5.43: Configuración MISO con posicionador X

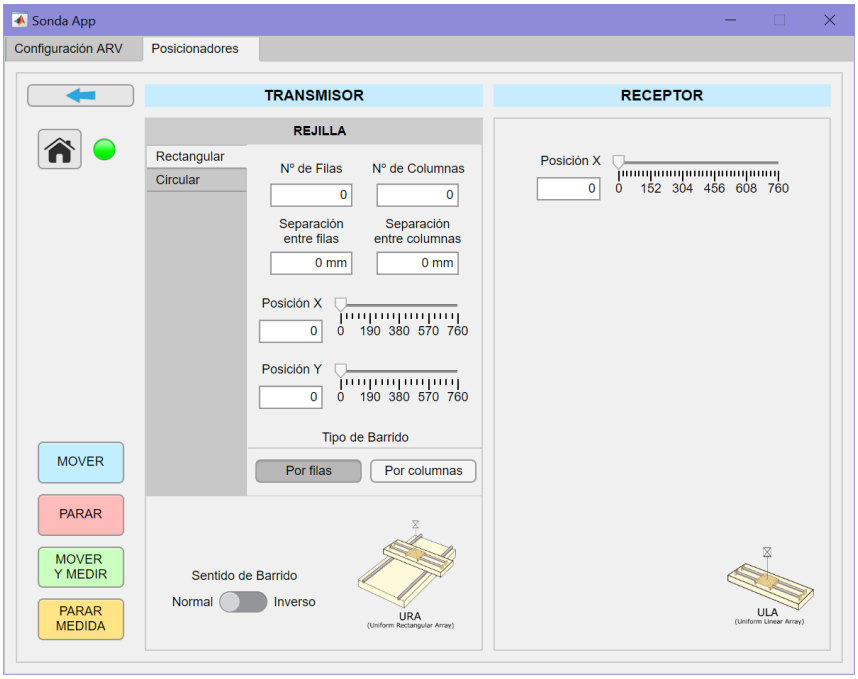


Figura 5.44: Configuración MISO con posicionador XY

Medida MIMO

Para las medidas MIMO, se unen las configuraciones de los dos posicionadores. Podemos elegir cual queremos que sea el transmisor o el receptor.

Figura 5.45: Selección transmisor-receptor

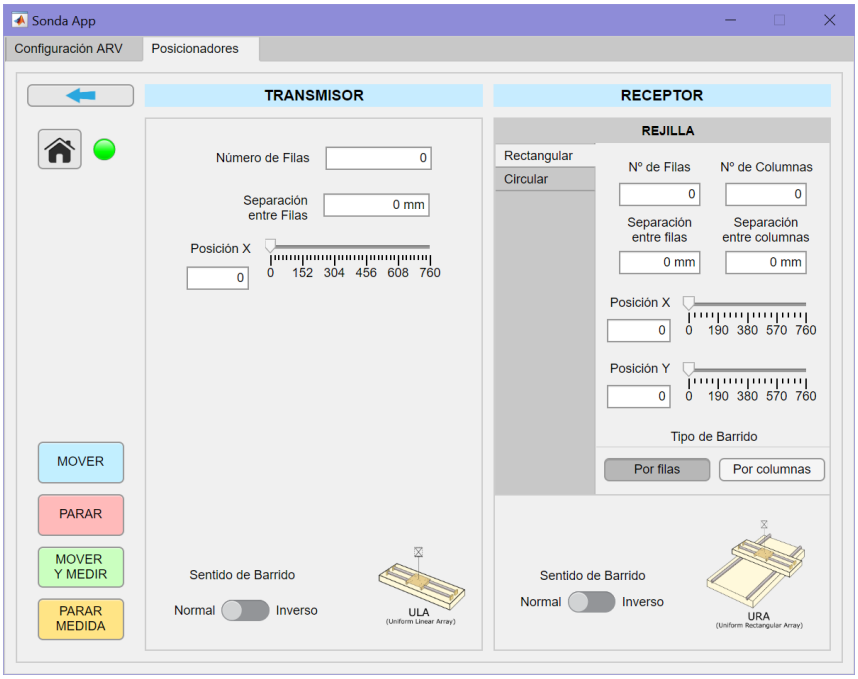


Figura 5.46: Primera configuración para MIMO

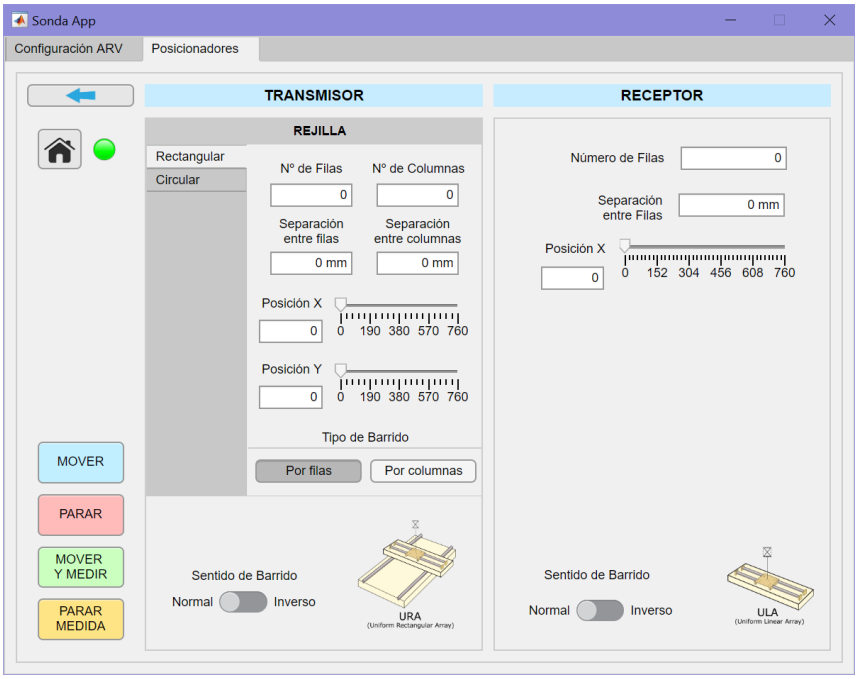


Figura 5.47: Segunda configuración para MIMO

Capítulo 6

Pliego de condiciones

6.1. Introducción

En este capítulo se realiza una valoración económica de la sonda y del trabajo realizado para su implementación. Todos los precios que se muestran a continuación incluyen el IVA (algunos de los equipos han sido comprados en España y otros en Estados Unidos de América). En la Tabla 6.1 se resume el presupuesto indicado.

6.2. Presupuesto

Material y equipos			
Elemento	Unidades	Coste por unidad	Coste total
Analizador de redes *Keysight* N5227A	1	184.397€	184.397€
Posicionador XY *Arrick Robotics*	1	1.483,30€	1.483,30€
Posicionador X modelo tGlide 1500 mm	1	999,87€	999,87€
Sistema MD-2 *Arrick Robotics*	1	1.200€	1.200€
Sistema MD-2 con controlador C4	1	1.062,75€	1.062,75€
Antenas	2	1.794,79€	3.589,58€
Soportes de antenas, transiciones, motores y cables de red	1	382,84€	382,84€
Reductores	3	177,12€	531,36€
Ordenador personal	1	1000€	1000€
Definición y programación de la sonda			
Elemento	Horas dedicadas	Coste por hora	Coste total
Coste de hora de implementación y diseño de la sonda	300	40€	14.520€
		Coste total	209.166,70€

Tabla 6.1: Resumen de presupuesto

Capítulo 7

Conclusiones y propuesta de trabajo futuro

7.1. Conclusiones

El objetivo de este TFG era diseñar y automatizar una sonda de canal en el dominio de la frecuencia. Se ha dotado a la sonda del control independiente del ARV, con el que podemos medir el parámetro S_{21} además de otros parámetros de interés, pudiendo cambiar la configuración del analizador. También contiene el control de dos posicionadores para realizar todo tipo de medidas MIMO. Con todo ello podemos caracterizar el PDP a partir de medidas que se hagan con la sonda.

Se ha diseñado la interfaz gráfica para el control de los elementos con la nueva herramienta *App Designer de MATLAB*. Ésta nos ha permitido crear una interfaz muy intuitiva y fácil de usar, además de dar al analizador un uso independiente con los posicionadores. La comunicación con el analizador se ha implementado a través de comandos SCPI exclusivamente.

Por último, se ha diseñado la interfaz gráfica para los dos posicionadores dando la opción de realizar medidas en diferentes combinaciones.

7.2. Propuesta de trabajo futuro

Como propuestas de trabajo futuro, en primer lugar la propuesta más importante es la implementación de la programación de la parte de los posicionadores y la integración del movimiento de posicionadores con la realización de las medidas, además de la realización de una campaña de medidas usando el programa completo y el análisis de éstas. Esto formará parte de la temática de mi TFM.

Además, también es interesante la idea de utilizar otro tipo de posicionador, con un desplazamiento angular, para añadir así otro tipo de medidas y aumentar las posibilidades de la sonda.

También puede implementarse una primera visualización de la medida que se realice con los posicionadores para decidir seguir midiendo en las demás posiciones o cambiar alguna parte de la configuración. Esto ayudaría a evitar tener que repetir algunas medidas después de un procesado *offline*.

Bibliografía

[1] Santiago Martínez Toval. *Diseño y automatización de un sistema de medida para la caracterización del canal radio en interiores*. Trabajo Fin de Grado, 2016.

[2] Keysight Technologies. *COM versus SCPI*. url: `http://na.support.keysight.com/pna/help/latest/Programming/Learning_about_COM/COM_versus_SCPI.htm` (visitado 14-02-2020).

[3] Keysight Technologies. *PNA Series Network Analyzers Help*. url: `http://na.support.keysight.com/pna/help/latest/help.htm` (visitado 14-03-2020).

[4] Lorenzo Rubio Arjona. *Mecanismos de propagación y efecto multicamino*. Transparencias de la asignatura Radiocomunicaciones, Grado en Ingeniería y Tecnologías y Servicios de Telecomunicación, Universitat Politècnica de València, 2019.

[5] Jose María Hernando Rábanos, Jose Manuel Riera Salís y Luis Mendo Tomás. *Transmisión por radio*. Séptima Ed. Editorial Universitaria Ramón Areces, Madrid, 2013. isbn: 978-84-9961-106-8.

[6] Lorenzo Rubio Arjona. *Caracterización del canal radioeléctrico*. Transparencias de la asignatura Sistemas y Servicios de Transmisión por Radio, Máster Universitario en Ingeniería de Telecomunicación, Universitat Politècnica de València, 2020.

[7] Wikipedia. *Interfaz gráfica de usuario*. url: `https://es.wikipedia.org/wiki/Interfaz_grafica_de_usuario#Tipos_de_interfaces_graficas_de_usuario` (visitado 16-05-2020).

[8] MATLAB. *Comparing GUIDE and App Designer*. url: `https://es.mathworks.com/products/matlab/app-designer/comparing-guide-and-app-designer.html` (visitado 16-04-2020).

[9] MATLAB. *MATLAB App Designer Component Gallery*. url: `https://es.mathworks.com/products/matlab/app-designer/component-gallery.html` (visitado 25-05-2020).

[10] MATLAB. *MATLAB Web App Server*. url: `https://es.mathworks.com/products/matlab-web-app-server.html` (visitado 25-05-2020).

[11] MATLAB. *MATLAB Compiler*. url: `https://es.mathworks.com/products/compiler.html` (visitado 25-05-2020).

[12] Arrick Robotics. *C4 Controller User's Guide*. url: `https://arrickrobotics.com/c4guide.pdf` (visitado 20-04-2020).

[13] Arrick Robotics. *MD-2 Dual Stepper Motor System User's Guide*. url: `https://arrickrobotics.com/md2guide.pdf` (visitado 20-04-2020).

[14] Keysight Technologies. *Noise Reduction Techniques*. url: `http://na.support.keysight.com/pna/help/latest/S2_Opt/Trce_Noise.htm` (visitado 10-06-2020).