Escuela Superior Politécnica del Litoral

Facultad de Ingeniería en Electricidad y Computación

Diseño e implementación de un sistema de control y medición para la caracterización de canales inalámbricos

Proyecto Integrador

Previo la obtención del Título de:

Ingeniero en Telecomunicaciones

Presentado por:

Dick Bryan Torres Martínez

Luis Anthony Chinga Cruz

Guayaquil - Ecuador

Año: 2023

Dedicatoria

Le dedico este proyecto a mis padres que me han apoyado a lo largo de mi vida, hermano y abuelita por siempre darme paz en días difíciles.

A mis tíos y tía por inculcarme el esfuerzo continuo y guiarme con sus consejos.

A mi compañera de vida, Jamileth Aristega, por su comprensión, cariño y por estar presente en cada momento de dificultad para brindarme su apoyo incondicional.

Dick Bryan Torres Martínez

Dedicatoria

Dedico este proyecto a mis padres, compañera de vida y amigos, cuyo amor incondicional y apoyo constante han sido mi mayor motivación para alcanzar mis objetivos.

Luis Chinga

Agradecimientos

Expreso mi agradecimiento principalmente a Dios por darme la oportunidad de cumplir una meta importante, por permitir que mi familia esté conmigo en cada momento de mi vida universitaria.

A mis padres y hermano que forman el pilar principal de mi vida.

A mi tía, tíos y abuelita que siempre estuvieron dispuestos ayudarme.

A todos los profesores que tuve desde el preuniversitario que me impulsaron a seguir mejorando.

A los docentes, M.Sc. Maricela Freire, M.Sc. Eduardo Chancay y Francisco Novillo, Ph.D por orientarme y aconsejarme en el ámbito académico y profesional.

Dick Bryan Torres Martínez

Agradecimientos

Quiero expresar mi profundo agradecimiento a los docentes que compartieron este camino conmigo. Mi gratitud a mi familia, novia y amigos por su apoyo emocional y sacrificios para que pudiera concentrarme en este proyecto.

Luis Chinga

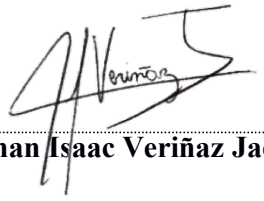
Declaración Expresa

“Los derechos de titularidad y explotación, nos corresponde conforme al reglamento de propiedad intelectual de la institución; Luis Anthony Chinga de la Cruz y Dick Bryan Torres Martinez damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Dick Bryan Torres Martínez

Luis Anthony Chinga de la Cruz

Evaluadores



Herman Isaac Veriñaz Jadán

PROFESOR DE LA MATERIA

Firmado electrónicamente por:
MARICELA ELISET
FREIRE BARBA

Maricela Eliset Freire Barba

PROFESOR TUTOR

Resumen

Este trabajo se centra en el desarrollo de un software en Python diseñado para el control preciso del mecanismo de orientación de una antena, cuyas partes son impresas en 3D. El software facilita la adecuada orientación de la antena, permitiendo tomar mediciones precisas en canales inalámbricos. Además, permite una manipulación manual y automática de los motores, así como la interconexión con un Analizador de Redes Vectoriales (VNA) para realizar mediciones en diversos ángulos. La solución proporciona una interfaz gráfica intuitiva para los usuarios, posibilitando el establecimiento de parámetros como la posición de los motores y la frecuencia en Hz de medición. Además, se implementa una función automática que permite explorar amplios rangos angulares como el procedimiento Channel Sounding. La integración de la antena en el soporte de movimiento y la conexión con el VNA confiere a este trabajo un enfoque innovador, abriendo posibilidades en aplicaciones como el modelado de enlaces eficientes de radiofrecuencia. El software resultante presenta una herramienta versátil y eficaz para el avance de la investigación en el diseño de sistemas inalámbricos y componentes electromagnéticos.

Palabras Clave: Control de motores, Impresión 3D, Antena, Python, Mediciones angulares

Abstract

This work addresses the development of comprehensive software in Python for precise control of the orientation mechanism of an antenna, parts of which are 3D-printed. The software facilitates the accurate positioning of the antenna, enabling precise measurements in wireless channels. It allows for both manual and automatic motor manipulation and includes integration with a Vector Network Analyzer (VNA) to perform measurements at different angles. An intuitive graphical interface is provided for users, enabling the setting of parameters like motor positions and the measurement frequency in Hz. Moreover, an automated function has been introduced to explore broad angular ranges, as in the Channel Sounding procedure. By integrating the antenna into the motion platform and its connection to the VNA, this work takes on an innovative approach, unlocking potential in applications such as efficient radiofrequency link modeling. The resulting software offers a versatile and effective tool to advance research in wireless system design and electromagnetic components.

Keywords: *Motor Control, 3D Printing, Antenna, Python, Angular Measurement*

Índice General

Índice de Figuras

Abreviaturas

ESPOL	Escuela Superior Politécnica del Litoral
VNA	Vectorial Network Analyzer
PDP	Power Delay Profile
GUI	Graphical User Interface
CSV	Comma Separated Values
FSPL	Free Space Path Loss,
CNC	Control Numérico Computarizado
TXT	Archivo de texto
RX	Antena Receptora
TX	Antena Transmisora
AZ	Azimuth
EL	Elevación

Simbologías

m	Metro
s	Segundo
A	Amperio
v	voltio
Hz	Hertz
°	Grados
GHz	Giga Hertz
log	Logarítmico
lin	Lineal
G	Giga
S21	Potencia recibida en el puerto 2 en comparación con la enviada por el puerto 1

Capítulo 1

1. Introducción

El rápido avance de las tecnologías inalámbricas que vive el mundo actualmente ha llevado a una creciente demanda de sistemas de comunicación confiables, sin dejar a un lado el rendimiento. Es de vital importancia reconocer la caracterización de un canal en los enlaces de radiofrecuencia para poder mitigar diversos problemas de interferencia o irregularidades en las comunicaciones inalámbricas. La caracterización del canal de radiofrecuencia implica el estudio de las propiedades y comportamientos del medio de transmisión, como la propagación de la señal, la atenuación, la interferencia y el ruido.

Comprender estas características es importante para certificar una comunicación efectiva y confiable. Un canal de radiofrecuencia puede estar sujeto a diferentes factores que perturban la calidad de la señal, como la atenuación debido a la distancia, la presencia de obstáculos físicos, la interferencia causada por otras fuentes de señales y el ruido electromagnético ambiental. Al caracterizar el canal, los ingenieros pueden determinar la influencia de estos factores y diseñar sistemas de comunicación que sean capaces de superar estos desafíos [1].

En el presente proyecto integrador se realiza un sistema que permita medir y analizar las características clave de los canales inalámbricos, como la atenuación, la interferencia y el ruido. El sistema propuesto utiliza una composición de hardware y software técnico para realizar mediciones precisas y obtener datos confiables. Se lleva a cabo el concepto de "Channel sounding" ya que se hace un medición y análisis de un canal inalámbrico de comunicación en tiempo real.

Una vez implementado, el sistema de control y medición permite llevar a cabo experimentos y mediciones en diferentes entornos inalámbricos, como áreas urbanas, rurales o interiores. Esto proporciona información valiosa sobre la variabilidad del canal y ayuda a los

ingenieros a optimizar los sistemas de comunicación inalámbrica, diseñar estrategias de mitigación de interferencias y mejorar la calidad de la transmisión.

1.1 Descripción del problema

En Ecuador, los diseñadores de enlaces de radiofrecuencia a menudo no llevan a cabo un estudio exhaustivo del canal de comunicación y del balance energético. Esto resulta en enlaces deficientes, la adquisición e instalación de equipos innecesarios y un impacto medioambiental negativo. Este problema no solo afecta a una sola empresa, sino que se extiende a todo el sector de las telecomunicaciones.

A medida que la demanda de servicios de comunicación inalámbrica continúa creciendo, es crucial comprender y optimizar la eficacia de las transmisiones en entornos inalámbricos. El problema que este proyecto aborda es la necesidad de presentar un sistema de control y medición que permite obtener una caracterización detallada de los canales inalámbricos. Este sistema proporciona mediciones precisas de factores clave como la atenuación, la interferencia y el ruido, lo que permite optimizar los métodos de comunicación inalámbrica, mejorar la calidad de la señal y proporcionar una experiencia de usuario superior.

1.2 Justificación del problema

La falta de un estudio adecuado del canal de comunicación conduce a la implementación de enlaces de radiofrecuencia ineficientes. Esto significa que la calidad de la comunicación puede verse comprometida, lo que resulta en conexiones lentas, interrupciones y una menor capacidad de transmisión de datos. Esta ineficiencia afecta negativamente a las empresas de telecomunicaciones y a los usuarios finales que dependen de una conexión confiable y de alta calidad. Al resolver este problema, se busca un avance significativo en la capacidad de diseñar y desplegar sistemas de comunicación inalámbrica más eficientes y confiables. Además, se abren oportunidades para el desarrollo de estrategias de mitigación de

interferencias, la optimización de recursos y el progreso continuo de la calidad de servicio en el sector de las telecomunicaciones.

La caracterización del canal ayuda en la planificación de redes inalámbricas al proporcionar información sobre la cobertura y la capacidad esperadas en diferentes ubicaciones y entornos. Esto permite una asignación más eficiente de recursos, como la ubicación de torres o estaciones base, la asignación de frecuencias y la configuración de la red, lo que conduce a una mejor calidad de servicio y una mayor capacidad de la red. Se puede llevar a cabo ubicaciones de torres y estaciones bases de una manera más estratégica reduciendo costos de construcción, ruido ambiental e inconvenientes a los ciudadanos.

1.3 Objetivo general

Mejorar la calidad, validez y sostenibilidad de los enlaces de radiofrecuencia en Ecuador a través de la implementación de un mecanismo de posicionamiento funcional y un software de asistencia, beneficiando tanto a las empresas de telecomunicaciones como al sector productivo en general.

1.3.1 Objetivos específicos

- Diseñar e implementar un sistema que permita el movimiento de la antena en las direcciones en azimut y elevación mediante motores de pasos.
- Posicionar y controlar la plataforma de medición del parámetro S21.
- Diseñar un esquema gráfico de usuario (GUI) que permita al usuario ingresar la posición en azimut y elevación deseada para la antena, el número de puntos de medición en el VNA, frecuencia inicial y final para las mediciones.
- Implementar el código para adquirir los datos de las mediciones del VNA y guardarlos en un archivo CSV de manera organizada para su posterior análisis.

1.4 Propuesta de solución

Se propone el desarrollo de un sistema que consta de una parte mecánica responsable del movimiento de la antena en azimut y horizontal controlado por computadora usando Python, con el objetivo de replicar eficientemente el concepto de "Channel Sounding". Esto permite una amplia caracterización de canal. Se cuenta también con una interfaz del Vectorial Network Analyzer (VNA) conectada a la antena y, a su vez, a la computadora para recopilar los resultados de la caracterización del canal utilizando Python.

1.5 Marco teórico

En esta sección, se detallan los temas que son esenciales para el entendimiento de la caracterización de canales inalámbricos y para plasmar un enfoque sólido y fundamentado para el diseño y ejecución del sistema de control y medición.

1.5.1 Características del canal inalámbrico

La caracterización del canal inalámbrico se refiere al estudio y análisis de las propiedades y comportamiento del canal de comunicación inalámbrico. El canal inalámbrico es el medio a través del cual las señales inalámbricas se propagan desde un transmisor hacia un receptor. Comprender las características del canal es esencial para el diseño y optimización de sistemas de comunicación inalámbrica. La caracterización del canal inalámbrico se lleva a cabo mediante mediciones empíricas y modelado matemático. Estos datos y modelos se utilizan para desarrollar algoritmos y técnicas de procesamiento de señales que mejoren la calidad de la comunicación inalámbrica, como técnicas de modulación, codificación, ecualización adaptativa, diversidad de antenas y cancelación de interferencias [3].

1.5.2 Perdida de propagación

La pérdida de propagación se refiere a la disminución de la potencia de la señal a medida que se propaga a través del espacio [4]. Esto puede deberse a factores como la distancia,

los obstáculos físicos, la absorción y la dispersión de la señal. La caracterización de la pérdida de propagación implica medir la atenuación de la señal en diferentes ubicaciones y frecuencias.

El modelo de pérdida de propagación más comúnmente utilizado es el modelo de pérdida de propagación en espacio libre (FSPL). Este prototipo se aplica en entornos donde no hay obstrucciones significativas y se enfoca en la ley del inverso del cuadrado de la distancia.

1.5.3 Fading

El fading es la variación aleatoria de la amplitud y la fase de la señal inalámbrica debido a la interferencia y los efectos de propagación. Puede ser causado por reflexiones, difracción, dispersión y multipath. El fading puede ser selectivo en frecuencia (afectando diferentes frecuencias de manera diferente) o no selectivo. La caracterización del fading implica medir y modelar la variabilidad de la señal en el tiempo y la frecuencia [6].

1.5.4 Retardo y dispersión de retardo

El retardo es el tiempo que tarda la señal en viajar desde el transmisor hasta el receptor. La dispersión de retardo se refiere a la propagación de la señal a través de múltiples rutas debido a reflexiones y dispersión [7]. La caracterización del retardo y la dispersión de retardo implica medir el retardo y el perfil de dispersión de la señal para comprender la interferencia intersimbólica y diseñar técnicas de ecualización y cancelación de interferencias [8].

1.5.5 Ancho de banda disponible

El ancho de banda disponible se refiere al rango de frecuencias disponibles para la transmisión inalámbrica. La caracterización del ancho de banda implica medir el espectro electromagnético y determinar las frecuencias utilizables para la comunicación inalámbrica sin interferencias perjudiciales.

El ancho de banda afecta directamente la calidad de la señal transmitida. Un ancho de banda insuficiente puede resultar en distorsiones, pérdida de información y degradación de la

señal. Al caracterizar el canal, es importante determinar si el ancho de banda disponible es suficiente para transmitir la señal deseada sin degradación significativa [9].

En entornos donde varios sistemas comparten el mismo rango de frecuencia, el ancho de banda disponible determina cuántos canales pueden utilizarse simultáneamente sin interferencia significativa. Al caracterizar el canal, es importante considerar la presencia de otras señales y sistemas que puedan interferir y evaluar cómo se distribuye el ancho de banda disponible entre ellos.

1.5.6 Técnicas de medición de canal inalámbrico

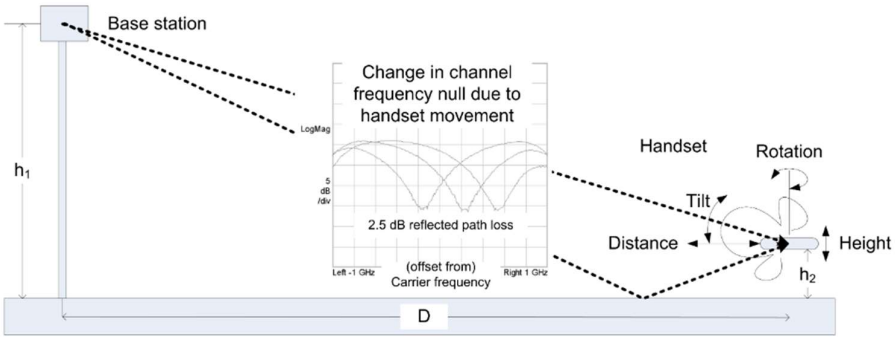
Las técnicas de medición de canales inalámbricos se utilizan para obtener información detallada sobre las características y propiedades de los canales de comunicación inalámbrica. Estas técnicas permiten estudiar cómo se propagan las señales a través del canal y cómo se ven afectadas por factores como la atenuación, el ruido y la interferencia. Estas técnicas de medición se aplican de acuerdo con los objetivos específicos del proyecto y las características del canal inalámbrico a caracterizar. Las mediciones proporcionan información valiosa para el diseño y la optimización de sistemas de comunicación inalámbrica, la mitigación de interferencias y la mejora de la calidad de la señal [10].

1.5.7 Channel sounding

Esta técnica implica enviar señales de prueba conocidas a través del canal y medir las respuestas recibidas. Se utilizan secuencias pseudoaleatorias, tonos o señales de prueba específicas para caracterizar las propiedades del canal. La información obtenida incluye la respuesta de frecuencia, la atenuación, el retardo de propagación y la dispersión temporal [12]. El objetivo principal del channel sounding es obtener una comprensión detallada de cómo el canal afecta la señal transmitida. Esto incluye la medición de parámetros clave del canal, como la respuesta de frecuencia, la atenuación, el retardo y el perfil de dispersión.

Figura 1.1

Canal de radio con rutas de señal en movimiento



Nota: La figura muestra el escenario de una farola que ilustra el problema de un canal de radio con rutas de señal en movimiento. Tomada de [20]

Para realizar el channel sounding, se utilizan equipos especializados que incluyen un transmisor y un receptor, y se llevan a cabo mediciones en tiempo real o se graban para su posterior análisis. Estos equipos generan señales de prueba y miden las características de la señal recibida después de su propagación a través del canal.

1.5.7.1 Sondeo de un solo tono (Single-tone sounding). En esta técnica, se transmite una única frecuencia o tono a través del canal y se mide la respuesta en frecuencia del canal mediante el análisis de la señal recibida [12].

1.5.7.2 Sondeo de espectro amplio (Wideband sounding). Esta técnica implica la transmisión de señales con un ancho de banda amplio y conocido. El objetivo es obtener información sobre la dispersión de la señal y los efectos de retardo en el canal [12].

1.5.8 Mediciones de potencia

Se realizan mediciones de potencia para determinar la potencia de la señal recibida en diferentes ubicaciones del canal. Esto proporciona información sobre la atenuación y las variaciones de potencia a lo largo del canal. Las mediciones de potencia también pueden utilizarse para calcular la relación señal a ruido (SNR) y evaluar la calidad de la señal [13].

1.5.9 Mediciones de interferencia

Estas mediciones se centran en cuantificar la interferencia presente en el canal. Se utilizan técnicas para identificar y medir las señales de interferencia, como señales de otros sistemas inalámbricos, dispositivos electrónicos cercanos o fuentes de ruido ambiental. Esto ayuda a evaluar la calidad de la señal y a diseñar estrategias de mitigación de interferencias [14].

1.5.10 Mapeo del canal

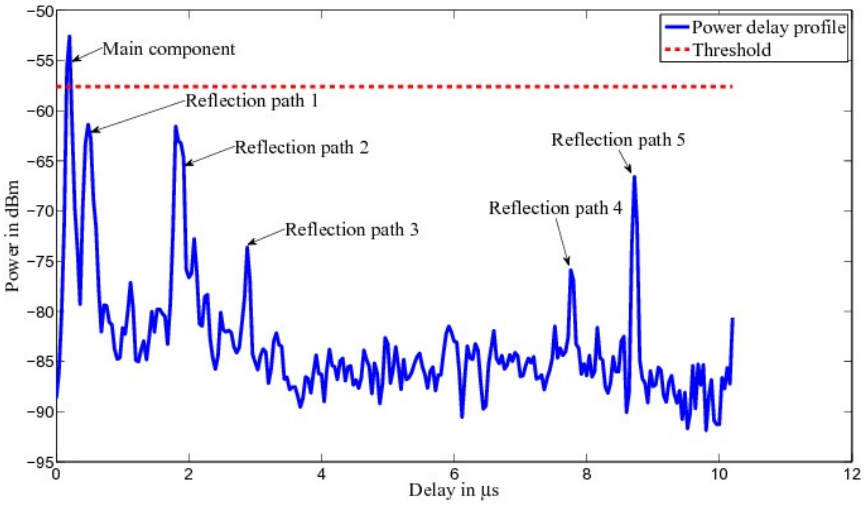
Esta técnica implica realizar mediciones en diferentes ubicaciones dentro de un área geográfica para obtener un mapeo detallado de las características del canal. Se miden la atenuación, la calidad de la señal y otros parámetros del canal en puntos específicos para evaluar la cobertura y la capacidad del sistema inalámbrico.

1.5.11 Power Delay Profile (PDP)

El PDP se utiliza comúnmente en el campo de las comunicaciones inalámbricas y las redes móviles para comprender mejor la propagación de la señal y el comportamiento del canal. En la mayoría de los entornos de propagación, una señal puede llegar al receptor a través de múltiples trayectorias debido a reflexiones, dispersión y refracción en el medio de propagación. Estas trayectorias tienen diferentes longitudes y, por lo tanto, diferentes retrasos de tiempo [16]. El PDP muestra cómo la energía de la señal varía en función del tiempo. Generalmente, presenta un patrón con varios picos, donde cada pico corresponde a una componente multipath con un retraso característico.

Figura 1.2

Potencia de la señal en función del retardo



Nota: Un perfil de retardo de potencia con el umbral. Tomado de [16].

Power Delay Profile es una herramienta clave para comprender la propagación de señales inalámbricas en canales con múltiples trayectorias. Proporciona información esencial para el diseño de sistemas de comunicación y la optimización de la calidad de la señal, especialmente en entornos complejos y desafiantes [17].

Capítulo 2

2. Desarrollo de la metodología

Inicialmente, se llevó a cabo una investigación para identificar un lenguaje de programación que permitiera controlar los movimientos en azimut y elevación de una antena, teniendo en cuenta los drivers y motores disponibles, con el objetivo de replicar el concepto de Channel Sounding. A partir de esto, se diseñó el software. En cuanto a la parte física, se fabricaron piezas mediante impresión 3D y se procedió a su ensamblaje.

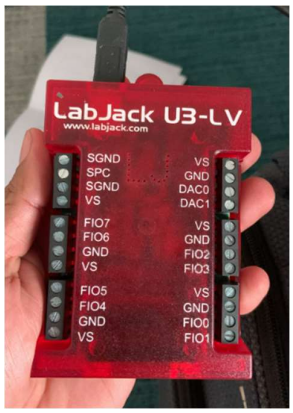
Una vez finalizada esta primera fase, se colocaron dos motores responsables de proporcionar el movimiento al sistema en azimut y elevación a la antena. Estos dos motores fueron conectados a dos drivers que tienen la función de adaptadores y, a su vez, los drivers se conectaron a un labjack U3 para poder controlar el movimiento de los motores por medio de código Python. Posteriormente, se desarrolló una interfaz para el VNA en código Python con el fin de presentar los resultados obtenidos por el Vectorial Network Analyzer.

Con todos los dispositivos físicos listos y definida la metodología que se usaría para controlar los movimientos, se procedió a la elaboración del software. Este software fue construido en el lenguaje de programación Python y se dividió en dos partes: la parte mecánica, encargada de todo el mecanismo de movimiento que controlaba los motores, y la parte de comunicación con el VNA. El objetivo era obtener los valores del parámetro S21 del canal inalámbrico y, de esta manera, caracterizar el canal inalámbrico.

2.1 Dispositivos utilizados

Figura 2.1

Controlador LabJack - LV U3

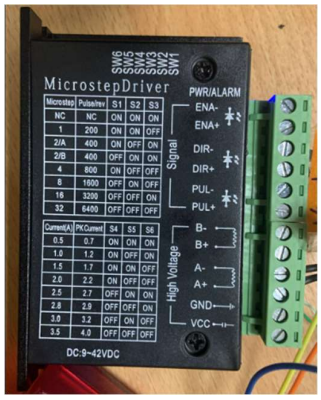


Nota: Controlador LabJack que envía pulsos discretos al driver.

El controlador LabJack es un dispositivo que permite la comunicación entre computadoras y sistemas de medición y control, además proporciona bibliotecas de software en varios lenguajes de programación. Una fotografía del LabJack se puede observar en la Figura 2.1.

Figura 2.2

Driver TB6600



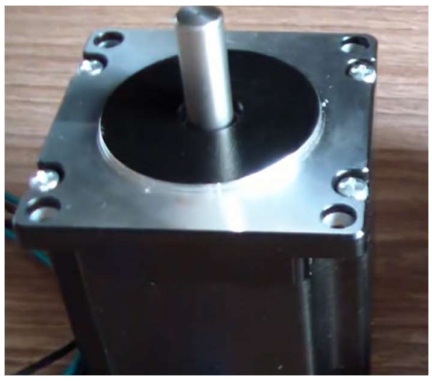
Nota: Driver que interpreta los pulsos enviados por el LabJack.

El TB6600 se utiliza en una variedad de aplicaciones que requieren control de motores paso a paso, como CNC, impresoras 3D, sistemas de posicionamiento. Se optó por este driver

debido a los niveles de corriente en sus pines de salida, ya que el motor opera con 1 A y el driver cuenta con este rango de amperaje. Una fotografía del driver se puede observar en la Figura 2.2.

Figura 2.3

Motor de pasos



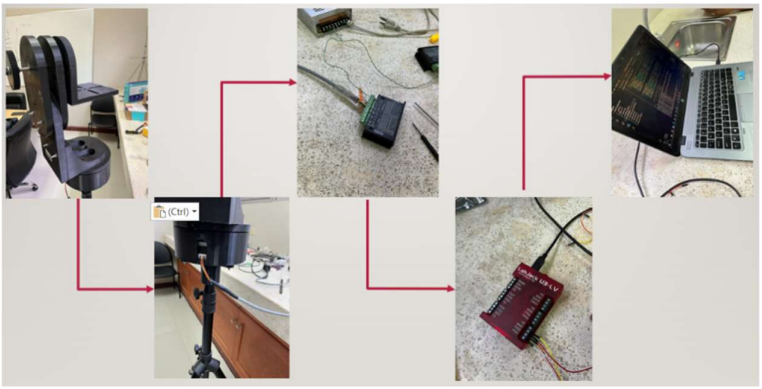
Nota: Motor de pasos para dar movimiento a la antena.

El motor de pasos de la marca moon tiene una configuración de 1.8° por paso, lo cual es configurable de acuerdo con el driver utilizado proporcionando precisión en el sistema de posicionamiento. Una fotografía del motor se puede observar en la Figura 2.3.

2.2 Conexión de los dispositivos

Figura 2.4

Conexión de los dispositivos

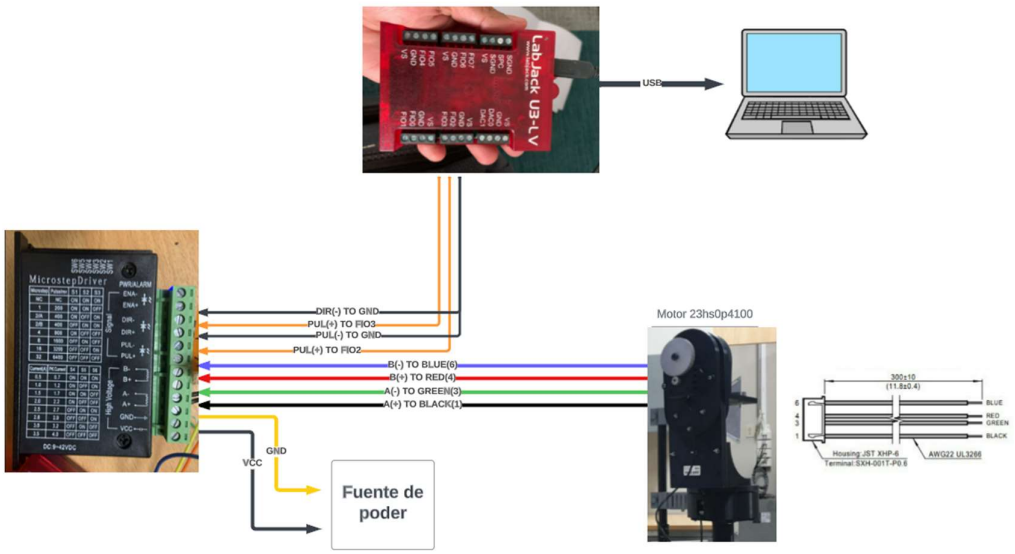


Nota: Conexión física de los dispositivos realizado en el laboratorio

Durante la implementación del sistema, se configuró el controlador LabJack para enviar regularmente pulsos de corriente discretos a través de sus pines. Estos pulsos fueron recibidos por el driver de pasos tb6600, que demostró tener la capacidad de interpretar los pulsos de corrientes y los gestionó a pasos y dirección del motor. Fue por esta razón que se conectó el motor al driver, permitiendo que el motor lograra moverse la cantidad deseada de pasos y con dirección, ya fuera hacia la derecha o hacia la izquierda. Para ilustrar de manera clara las conexiones del sistema, se incluyó la Figura 2.4 y la Figura 2.5 en el documento.

Figura 2.5

Diagrama de conexión de los dispositivos



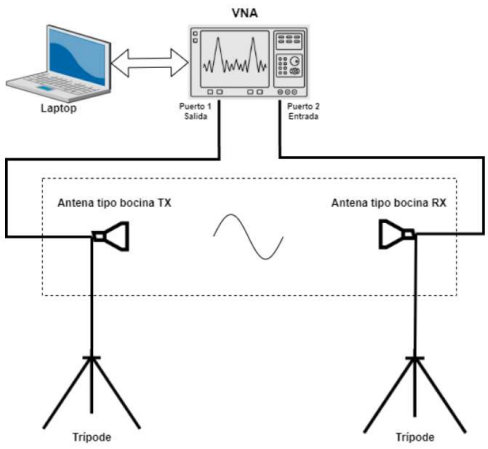
Nota: Visualización grafica de las conexiones de los pines entre el LabJack, el driver y el motor

2.3 Procedimiento de medición

La comunicación que se estableció entre la computadora y el VNA fue para poder controlar sus configuraciones de frecuencia, formato de valores y números de puntos como se puede observar en la Figura 2.6. Bajo estas configuraciones que ingresó el usuario, el VNA tomó las mediciones y se obtuvieron los valores del parámetro S21 conforme a estas configuraciones de la transmisión inalámbrica.

Figura 2.6

Conexión del VNA con las antenas y la PC

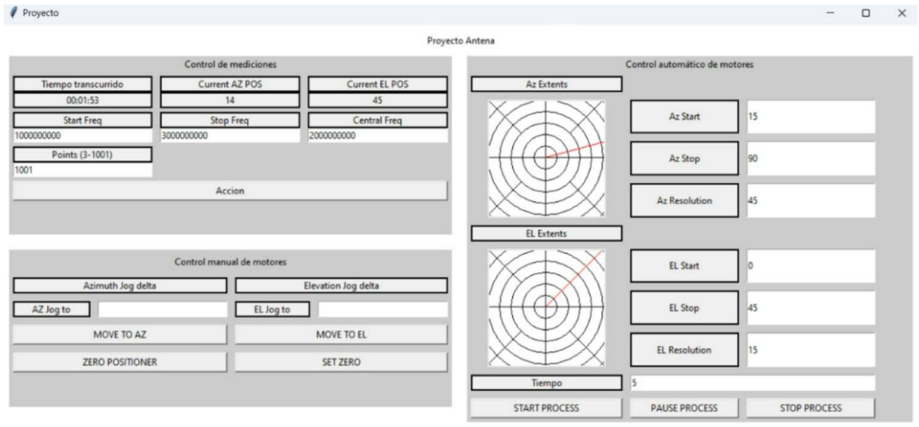


Nota: Representación ilustrativa de la conexión de las antenas pare realizar la medición. Tomado de [19]

La parte mecánica y la parte de la comunicación con el VNA se plasmó en una interfaz en la cual el usuario pueda indicar el ángulo que desea mover la antena en elevación y azimut. También se habilitó para que el usuario pudiera ingresar las configuraciones de la frecuencia inicial, frecuencia final y número de puntos para obtener los valores del parámetro S21. Se establecieron dos opciones para la toma de medidas: manual y automáticamente.

Figura 2.7

Interfaz de usuario



Nota: Interfaz gráfica de interacción del usuario

En la Figura 2.7 se presenta la interfaz en la que el usuario ingresa los valores de mediciones y las posiciones de la antena. En la parte superior izquierda se ingresa los valores de configuración del VNA para tomar la medición, en la parte inferior izquierda se ingresa la posición en grados de la antena para realizar una medición manual. Una vez ingresados los datos de configuración del VNA y posición de la antena, se presiona el botón “acción” y se obtiene un archivo .csv o .txt de los valores del parámetro S21 y de las frecuencias.

Para el modo automático, se configuró el sistema de manera que el usuario pudiera introducir las posiciones de inicio y fin de la antena con su respectiva resolución. Se programó la antena para hacer un barrido basado en las posiciones definidas por el usuario, registrando la medición del parámetro S21 en cada posición según la resolución determinada

2.4 Prueba de línea de vista directa

Figura 2.8

Prueba de medición

Nota: Transmisión inalámbrica con línea de vista sin obstáculos

Como se mencionó anteriormente, se conectó la antena TX al puerto 1, manteniéndola fija, mientras que la antena RX, que realizó un barrido de 180° en azimut y 45° en elevación, se conectó al puerto 2. En la Figura 2.8, las antenas tienen una distancia entre sí de 3 metros y están ubicadas en línea de vista.

2.5 Prueba de línea de vista indirecta

Figura 2.9

Prueba de medición

Nota: Transmisión inalámbrica sin línea de vista

Las mediciones se llevaron a cabo con una frecuencia inicial de 2GHz y frecuencia final de 3GHz, con 1001 puntos, lo que significó que se obtuvieron 1001 valores del parámetro S21. Como se observa en la Figura 2.9, la antena RX se fijó a 45° en azimuth hacia la izquierda y 45° en elevación hacia la izquierda.

2.6 Consideraciones éticas y legales

Durante el proceso de la toma de mediciones, no se causaron interferencias dañinas a otros dispositivos que estuvieron cerca del área de medición. No se incumplió con las regulaciones sobre el uso del espacio radioeléctrico y las normativas de radiocomunicaciones,

ya que fue una transmisión en rango de frecuencias de 2G Hz a 3G Hz, obteniendo solamente el parámetro S21 de la transmisión inalámbrica. Los datos recolectados no perjudicaron ni atacaron la privacidad de ninguna entidad; los valores del parámetro S21 fueron utilizados únicamente para temas profesionales en el análisis de una transmisión inalámbrica. El proyecto fue realizado en su totalidad con fines investigativos y para el avance tecnológico de las telecomunicaciones.

.

Capítulo 3

3. Resultados y análisis

En esta sección, se presentan y analizan los resultados obtenidos del parámetro S21 al mover la antena en las direcciones de azimut y elevación.

Se hicieron dos tipos de mediciones por lo tanto se obtuvieron dos tipos de resultados en la recolección de datos. Se realizó la medición de manera manual y de manera automática, en ambas mediciones el formato del archivo que se obtiene en formato .csv y fue exportado a Excel para su posterior procesamiento.

3.1 Resultados de manera manual

Conforme se describió en la metodología, existen dos modos de medición: manual y automática. En la Figura 3.1 se muestran los datos recopilados cuando se realiza una medición manual, consta de 4 columnas que son frecuencia, S21 magnitud, current AZ y current EL.

Figura 3.1

Visualización de resultados manuales en Excel

	Frecuencia (Hz)	S21 Magnitud (dB)	Current AZ	Current EL
2	+2.00000000000E+009,+2.33333333333E+009,+2.6666	+1.52949848362E-004,-6.35802983877E-006,-6.83368905291E-005,+3.6	35	50
3				

Nota: Visualización de los resultados obtenidos en la medición manual

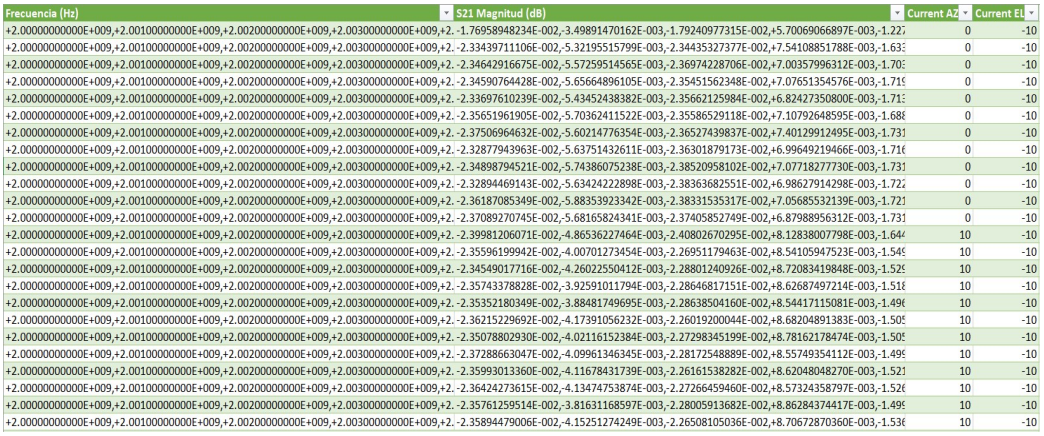
Solo existe una fila de datos ya que en el método manual se toma una sola medición en la posición que el usuario haya ingresado en azimuth y elevación. La cantidad de valores es de acuerdo con el número de puntos que el usuario haya ingresado.

3.2 Resultados de manera automática

La Figura 3.2 exhibe los datos de la medición automática. Al igual que con el método manual, se presentan cuatro columnas: frecuencia, magnitud, posición en azimuth y posición en elevación.

Figura 3.2

Visualización de resultados automáticos en Excel



Frecuencia (Hz)	S21 Magnitud (dB)	Current AZ	Current EL
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-1.76958948234E-002,-3.49891470162E-003,-1.79240977315E-002,+5.70069066897E-003,-1.22	0	-10
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-2.33439711106E-002,-5.32195515799E-003,-2.34435327377E-002,+7.54108851788E-003,-1.63	0	-10
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-2.34642916675E-002,-5.57259514565E-003,-2.36974228706E-002,+7.00357996312E-003,-1.70	0	-10
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-2.34590764428E-002,-5.65664896105E-003,-2.35451562348E-002,+7.07651354576E-003,-1.71	0	-10
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-2.33697610239E-002,-5.43452438382E-003,-2.35662125984E-002,+6.82427350800E-003,-1.71	0	-10
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-2.35651961905E-002,-5.70362411522E-003,-2.35586529118E-002,+7.10792648595E-003,-1.68	0	-10
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-2.37506964632E-002,-5.60214776354E-003,-2.36527439837E-002,+7.40129912495E-003,-1.73	0	-10
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-2.32877943963E-002,-5.63751432611E-003,-2.36301879173E-002,+6.99649219466E-003,-1.71	0	-10
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-2.34898794521E-002,-5.74386075238E-003,-2.38520958102E-002,+7.07718277730E-003,-1.73	0	-10
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-2.32894469143E-002,-5.63424222898E-003,-2.38363682551E-002,+6.98627914298E-003,-1.72	0	-10
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-2.36187085349E-002,-5.88353923342E-003,-2.38331535317E-002,+7.05685532139E-003,-1.72	0	-10
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-2.37089270745E-002,-5.68165824341E-003,-2.37405852749E-002,+6.87988956312E-003,-1.73	0	-10
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-2.39981206071E-002,-4.86536227464E-003,-2.40802670295E-002,+8.12838007798E-003,-1.64	10	-10
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-2.35596199942E-002,-4.00701273454E-003,-2.26951179463E-002,+8.54105947523E-003,-1.54	10	-10
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-2.34549017716E-002,-4.26022550412E-003,-2.28801240926E-002,+8.72083419848E-003,-1.52	10	-10
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-2.35743378828E-002,-3.92591011794E-003,-2.28646817151E-002,+8.62687497214E-003,-1.51	10	-10
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-2.35352180349E-002,-3.88481749695E-003,-2.28638504160E-002,+8.54417115081E-003,-1.49	10	-10
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-2.36215229692E-002,-4.17391056232E-003,-2.26019200044E-002,+8.68204891383E-003,-1.50	10	-10
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-2.35078802930E-002,-4.02116152384E-003,-2.27298345199E-002,+8.78162178474E-003,-1.50	10	-10
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-2.37288663047E-002,-4.09961346345E-003,-2.28172548889E-002,+8.55749354112E-003,-1.49	10	-10
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-2.35993013360E-002,-4.11678431739E-003,-2.26161538282E-002,+8.62048048270E-003,-1.52	10	-10
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-2.36424273615E-002,-4.13474753874E-003,-2.27266459460E-002,+8.57324358797E-003,-1.52	10	-10
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-2.35761259514E-002,-3.81631168597E-003,-2.28005913682E-002,+8.86284374417E-003,-1.49	10	-10
+2.00000000000E+009,+2.00100000000E+009,+2.00200000000E+009,+2.00300000000E+009,+2.	-2.35894479006E-002,-4.15251274249E-003,-2.26508105036E-002,+8.70672870360E-003,-1.53	10	-10

Nota: Visualización de los resultados obtenidos en la medición automática

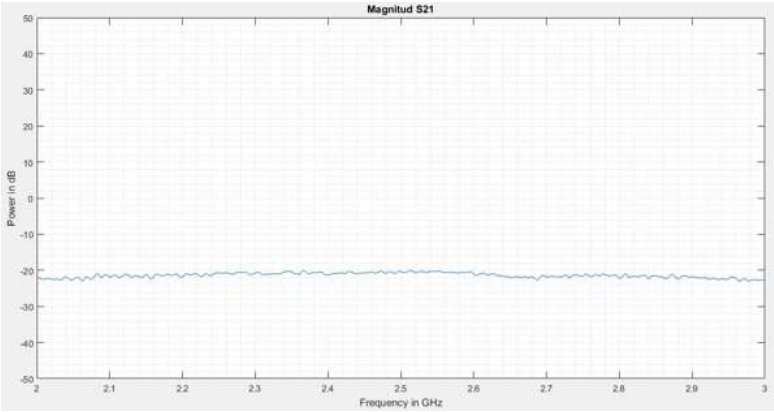
En el caso del método automático existen muchas más filas de datos. Esto es porque se obtienen los valores del parámetro S21 para cada posición en azimuth y elevación que el usuario haya ingresado con su respectiva resolución. Este método es ideal para realizar un barrido.

3.3 Resultados de mediciones del VNA

Los resultados de las mediciones realizadas con el dispositivo VNA se presentan en las Figuras 3.1 y 3.2. Los datos capturados fueron graficados en magnitud vs frecuencia y en potencia vs distancia (Power Delay Profile). Específicamente, se muestra la variación de las magnitudes S21 en relación con la frecuencia en la Figura 3.3. La caracterización de un canal inalámbrico a través del PDP mediante el análisis de parámetros S21 fue un enfoque complementario que proporcionó información valiosa para el diseño y la optimización de sistemas de comunicación inalámbrica.

Figura 3.3

Magnitud S21 vs Frecuencia en GHz

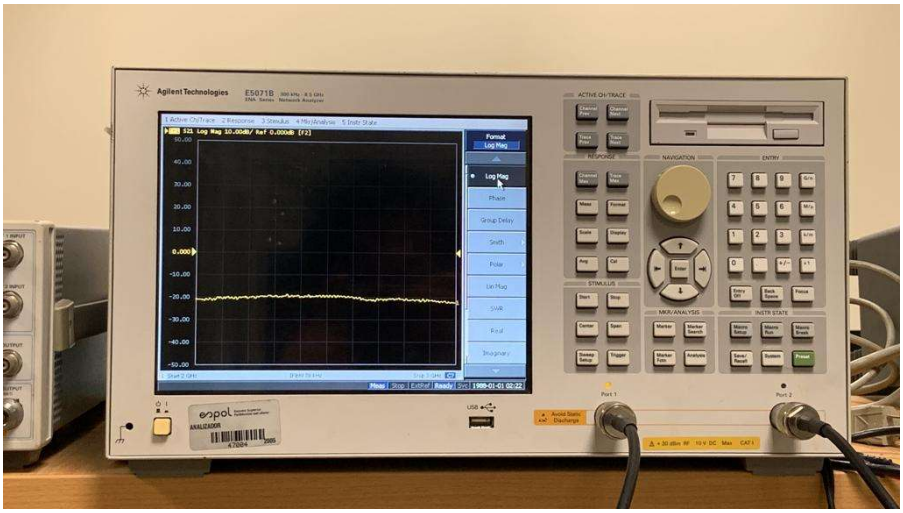


Nota: Visualización del gráfico la magnitud del parámetro S21 vs Frecuencia en GHz

Utilizando la plataforma de programación y cálculo "Matlab", se generaron las gráficas de magnitud versus frecuencia. Estas gráficas permitieron corroborar visualmente los valores obtenidos del parámetro S21. En la Figura 3.3 se presenta la gráfica generada con Matlab, mientras que la Figura 3.4 muestra la gráfica visualizada en el VNA durante la medición. Ambas gráficas son similares, lo que sugiere una coherencia en los resultados.

Figura 3.4

Magnitud S21 vs Frecuencia en el VNA

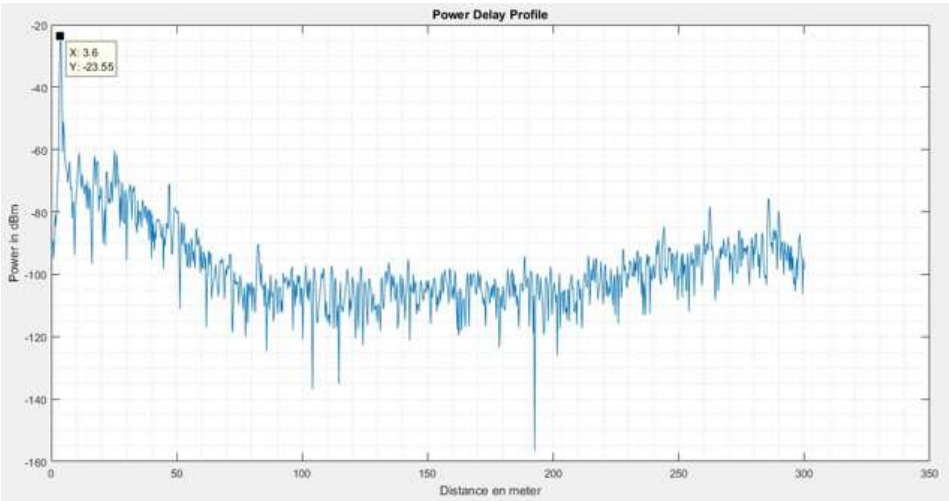


Nota: Gráfica del VNA cuando se realiza la medición del parámetro S21

Además de la representación magnitud vs frecuencia, la visualización Power Delay Profile agrega otra dimensión importante a nuestros resultados. Esta representación gráfica nos permite explorar cómo la potencia de las señales varía en función de la distancia recorrida como se muestra en la Figura 3.5, proporcionando información valiosa sobre las características de la propagación de señales en diferentes condiciones angulares. La combinación de ambas representaciones nos brinda una visión más completa y detallada de la respuesta de nuestro sistema en términos de magnitud y propagación de señales.

Figura 3.5

Potencias en dBm vs distancias



Nota: Visualización de la gráfica que define la distancia más corta en la que se encuentran las antenas

Por ejemplo, en la Figura 3.5 se puede observar que el pico de potencia más alto fue a los 3.6 m, es decir que esa fue la distancia más corta en la que llega la señal de una antena a otra. Al momento de realizar la medición, la distancia real entre las antenas era de 3 m, lo que indica una variación de solo 0.6 m.

Es importante resaltar que estas gráficas se obtuvieron utilizando únicamente los resultados obtenidos de los valores del parámetro S21, es decir, no se utilizaron los valores de frecuencia, ni posicionamiento en azimuth y elevación.

El VNA tiene la capacidad de entregar los valores del parámetro S21 en diferentes formatos, por ejemplo: log/fase, lin/fase, G+jB, etc. El formato escogido fue el “polar” (Real/Imaginario), en el que se obtuvieron los resultados del parámetro S21. Con el formato escogido se obtuvo el doble de valores de acuerdo con el número de puntos ingresado por el usuario, es decir, si el usuario ingresó 1001 puntos, al final se obtuvieron 2002 valores del parámetro S21, 1001 reales y 1001 imaginarios.

Figura 3.6

Resultado del VNA en formato polar

Nota: Visualización de la gráfica del VNA cuando se obtiene el parámetro S21 en forma polar

3.4 Correlación entre la posición de la antena y parámetros S21

Al analizar los datos presentados en las secciones anteriores, se observó una interesante correlación entre la posición angular de la antena y los valores de los parámetros S21. En particular, se identificó una tendencia de amplificación en ciertas direcciones angulares específicas.

Esta observación sugiere la existencia de puntos de orientación preferentes que podrían ser aprovechados en aplicaciones donde se requiere una mayor respuesta de señal. Por otro lado, también se aprecia una atenuación en otras posiciones angulares, lo que podría indicar zonas de menor cobertura o interferencias en esos puntos.

La correlación encontrada respalda la idea de que el rendimiento de la antena está intrínsecamente ligado a su orientación. Estos resultados también subrayan la importancia de considerar cuidadosamente la posición angular al diseñar sistemas de comunicación o aplicaciones que dependen de la captación de señales. En última instancia, estas tendencias ofrecen información valiosa para optimizar la dirección de la antena y mejorar la calidad de las conexiones y transmisiones en aplicaciones del mundo real.

3.5 Análisis de resultados

Como se mencionó en la sección de resultados, en la gráfica del Power Delay Profile se reflejó que a 3.6 metros está el pico de potencia. Durante el experimento, las antenas se situaron a una distancia de 3 m entre ellas, están directas y cuentan con una línea de vista sin obstáculos, como se puede observar en la Figura 3.7. Esto quiere decir que se presentaron atenuaciones en el canal inalámbrico porque la Figura 3.5 sugiere que la distancia más corta fue de 3.6 m, indicando que hay una pérdida de potencia.

Las antenas y el medio de propagación pueden introducir atenuaciones y ganancias que afectan la potencia recibida. Si la ganancia del canal supera las pérdidas, podría resultar en una potencia aparentemente mayor a la esperada.

Figura 3.7

Posiciones de las antenas en la medición

Nota: Medición del parámetro S21 con línea de vista en el laboratorio

No solamente es importante verificar que exista línea de vista libre de obstáculos físicos entre las dos antenas al momento de realizar una transmisión inalámbrica para brindar un servicio de comunicación, sino que también se deben evaluar otros parámetros como la capacidad de las antenas, interferencias externas y el entorno de propagación. En el diseño y optimización de canales inalámbricos son diversos los factores a tomar en cuenta para una tener una buena señal. La presencia de patrones de amplificación en ciertas direcciones de azimut y elevación sugiere la existencia de zonas preferenciales de radiación. Lo que hace que estas mediciones sean útiles en aplicaciones que requieran una cobertura enfocada en regiones específicas.

Estos patrones de amplificación podrían ser explotados en escenarios donde se necesite una comunicación o detección concentrada en determinadas áreas geográficas. Los resultados obtenidos proporcionan una base empírica sólida para la toma de decisiones en el diseño y la implementación de sistemas de antenas, permitiendo maximizar la eficiencia y efectividad de las comunicaciones.

La consistencia de estos resultados con las predicciones teóricas respalda la validez de la metodología implementada. Además, valida la precisión y la confiabilidad del software desarrollado para controlar tanto la antena como el dispositivo de medición VNA. Esta coherencia entre la teoría y la práctica refuerza la confianza en la rigurosidad de nuestro enfoque y la fiabilidad de los datos recopilados, cimentando la base para una interpretación sólida de los resultados.

En esta sección, se han presentado y analizado los resultados obtenidos a través de la metodología propuesta. Los patrones observados en las mediciones de antena y las tendencias en las mediciones VNA indican un potencial significativo para aplicaciones prácticas. Estos resultados contribuyen al entendimiento del comportamiento de la antena en relación con

diferentes parámetros y sentarán las bases para la discusión en el siguiente capítulo, donde se discutirán las conclusiones generales de la investigación.

3.6 Análisis de costos

Como se mencionó anteriormente en este documento, el lenguaje de programación escogido para la elaboración del software fue Python, el cual es un lenguaje de programación gratuito, y las librerías usadas también lo son. A su vez se utilizó el programa Matlab que también es gratuito, para la realización de la representación gráfica de los resultados obtenidos. El hardware que se utilizó para la elaboración del proyecto fue otorgado por el cliente del proyecto, por lo que no se compró ningún componente electrónico.

Capítulo 4

4. Conclusiones y recomendaciones

En el presente capítulo se presentan las conclusiones de acuerdo con los objetivos planteados y la evaluación de los resultados del proyecto. También se presentan las recomendaciones adecuadas a tomar en cuenta en la resolución del proyecto.

4.1 Conclusiones

- El desarrollo del sistema de orientación para la antena no solo representa un logro técnico, sino que también refuerza la importancia de la precisión en investigaciones que dependen de mediciones angulares. Su implementación potenciará futuros trabajos que requieran una orientación precisa y constante, brindando así una plataforma sólida para llevar a cabo mediciones detalladas y consistentes en una variedad de posiciones angulares.
- La implementación de la plataforma para medir el parámetro S21 ha asegurado mediciones precisas y consistentes, reduciendo potenciales errores y siendo esencial para la recolección de datos confiables. Esta capacidad no solo ha facilitado la obtención de resultados confiables, sino que también ha sentado las bases para futuras investigaciones en este campo.
- La creación de una interfaz gráfica de usuario (GUI) no solo resultó esencial para permitir a los usuarios definir la posición deseada en azimut y elevación para la antena, sino también para configurar otros parámetros relevantes para las mediciones. Esta interfaz intuitiva no solo mejoró la usabilidad del sistema, sino que también contribuyó significativamente a la precisión del proceso de toma de datos.
- La exitosa implementación del código para adquirir y organizar los datos de las mediciones del VNA en un formato de archivo CSV organizado representa un hito significativo. Esta implementación no solo ha permitido un almacenamiento estructurado de los datos capturados, sino que también ha facilitado un análisis eficiente

y sistemático en etapas posteriores, demostrando así su importancia en el proceso de investigación.

- La realización exitosa de los objetivos específicos no solo respalda la validez y efectividad de la metodología desarrollada, sino que también sienta las bases para futuras investigaciones y aplicaciones en el campo de las antenas y las comunicaciones. Los resultados obtenidos no solo enriquecen la comprensión del comportamiento de la antena en función de la orientación, sino que también abren oportunidades para la optimización de sistemas de comunicación y redes inalámbricas, subrayando así su importancia en el avance tecnológico y científico del área.

4.2 Recomendaciones

- Un área prometedora podría ser la expansión de este estudio para analizar sistemas de antenas múltiples en lugar de una única antena. Explorar cómo la orientación de múltiples antenas puede influir en el rendimiento y en la formación de haces direccionales podría tener implicaciones significativas en el diseño de redes inalámbricas y sistemas de comunicación más avanzados.
- Mejora continua de la interfaz gráfica de usuario (GUI) para hacerla más intuitiva y de fácil uso podría aumentar aún más la eficiencia del proceso de adquisición de datos. Además, la automatización de ciertas etapas, como la selección de posiciones angulares y la configuración del VNA, podría reducir aún más la posibilidad de errores humanos y acelerar el proceso global.
- Asegurarse de seleccionar los componentes adecuados, como motores de paso, controladores, VNA y otros elementos físicos necesarios. Investigar y elegir productos de calidad que sean compatibles y se adapten a las necesidades del Proyecto. Tener en cuenta factores como la corriente con la que operan los dispositivos, calentamiento, voltaje que resisten y el peso.

Referencias

[1] Xuefeng, Y., & Xiang, C. (2016). Propagation Channel Characterization, Parameter Estimation, and Modeling for Wireless Communications. Pekin: Wiley-IEEE Press.

[2] Barrios, U., & Alexis. (2021). Comparación de modelos de propagación de ondas de radio de un canal inalámbrico en un entorno urbano de la ciudad de Barranquilla. Corporación Universidad de la Costa, 31, 2745-0090. Accedido el 22 de junio, 2023, desde https://repositorio.cuc.edu.co/handle/11323/8739.

[3] G. C. Abner, "Medición con un sistema de posicionamiento para caracterizar un dipolo en la banda de frecuencia de 700mhz", Tesis de pregrado, Dept. Ing. Comunicaciones., Inst. Politécnico Nacional, México, 2020.

[4] P. M. Carlos, & L. P. Giovanny, "Caracterización de una antena patch de doble ranura para transmisión de energía inalámbrica", Tesis de pregrado, Dept. Ing. Telecomunicaciones., Inst. Tecnológico Metropolitano, Colombia, 2017.

[5] S. P. Hernán, & V. I. José, "Un simulador de canales inalámbricos", Tesis de pregrado, Dept. Ing. Telecomunicaciones., Univ. Pontificia Católica de Valparaíso, Chile, 2006.

[6] Galvis, Alexander., & Gómez, C., & Hincapié, R. (2007). Wireless Channel Models and their Application on Design. Universidad Pontificia Bolivariana. Accedido el 25 de junio, 2023, desde https://www.icesi.edu.co/revistas/index.php/sistemas_telematica/article/view/964

[7] Silvina. A., & Cabrera. M., & Bilbao. J & Ferreyra. (2012). Sistemas de comunicación inalámbricos con tecnología MIMO. Dpto. de Electricidad, Electrónica y Computación, 1668-9178. Accedido el 25 de junio, 2023, desde https://www.facet.unt.edu.ar/revistacet/wcontent/uploads/sites/28/2023/03/n34inv01.pdf

[8] S. S. Alfredo, “Pérdidas por propagación y dispersión temporal para el canal de radiocomunicaciones omnidireccional y direccional a frecuencias de milimétricas en entorno microcelular de plantación de cítricos”, Tesis de pregrado, Dept. Ing. Telecomunicaciones., Univ. Politécnica de Cartanega, Colombia, 2020.

[9] C. Correa, “Evaluación de las características de enlace de largo alcance carentes de línea de vista para aplicaciones WLL usando protocolo IEEE 802.11b”. Tesis de pregrado. Departamento de Electrónica. Universidad Técnica Federico Santa María. Valparaíso, Chile. 2004.

[10] Del Valle. D. M., & Caldera. R., & Ramirez. J., & Martínez. J. (2009). MODELADO DE CANAL INALÁMBRICO OFDM, HACIA LAS COMUNICACIONES 4G. Concibe, 1665-5745. Accedido el 25 de junio, 2023, desde http://e-gnosis.udg.mx/index.php/eg/article/view/111/92

[11] Channel sounding technique using MIMO software radio architecture. Artículo presentado en 5th European Conference on Antennas and Propagation (EUCAP), Roma, Italia.

[12] Weiner. A., & Dezfooliyan. A. (2012). Evaluation of Time Domain Propagation Measurements of UWB Systems Using Spread Spectrum Channel Sounding. IEEE Transactions on Antennas and Propagation, 1558-2221. Accedido el 25 de junio, 2023, desde https://ieeexplore.ieee.org/abstract/document/6236039

[13] Suárez, M. F. (2016). Estudio y recomendaciones para mitigar la interferencia en redes wifi en zonas no controladas no licenciadas. Recuperado de: http://hdl.handle.net/20.500.11912/2633.

[14] D. L. Hernandez., & R. Torres., & I. Flores, “Sistema inalámbrico de transmisión de audio con aplicación a domótica”, Tesis de pregrado, Dept. Ing. Telecomunicaciones., Inst. Politécnico Nacional, México, 2012.

[15] Milanés. D, “Demodulación y síntesis de señales ODFM, IEEE 802.11”, Tesis de pregrado, Dept. Ing. Telecomunicaciones., Univ. Nacional de Cuyo, Argentina, 2021.

[16] Yang, Kun & Roste, Terje & Bekkadal, Fritz & Ekman, Torbjörn. (2010). Channel characterization including path loss and Doppler effects with sea reflections for mobile radio propagation over sea at 2 GHz. 1-6. 10.1109/WCSP.2010.5633545.

[17] C. L. Holloway, M. G. Cotton and P. McKenna, "A model for predicting the power delay profile characteristics inside a room," in IEEE Transactions on Vehicular Technology, vol. 48, no. 4, pp. 1110-1120, July 1999, doi: 10.1109/25.775360.

[18] Ruíz, M. O. (2015). Matlab aplicado a telecomunicaciones. Madrid: Marcombo.

[19] Agilent E5070B/E5071B ENA Series RF Network Analyzers, Programmer’s Guide

[20] Cain, P. (2015). Channel Sounding Measurement of a Time Varying cm and mm Wave Channel. Accedido el 28 de junio, 2023, desde https://www.5gtechnologyworld.com/channel-sounding-measurement-of-a-time-varying-cm-and-mm-wave-channel/

Apéndice

Apéndice A. Código del control automático (automaticControl.py)

```
import threading
import tkinter as tk
from tkinter import messagebox

import math
import asyncio


class AutomaticControl(tk.Frame):
    def __init__(self, _, motores, temporizador, graficos, medicion):
        super().__init__()

        # Instanciar graficos
        self.graficos = graficos
        # Instanciar motores
        self.motores = motores
        # Instanciar medicion
        self.medicion = medicion

        # Instanciar temporizador
        self.temporizador = temporizador

        # Crear el Frame
        self.frame = tk.Frame(_, bg="#CCCCCC")
        self.frame.grid(row=0, column=1, rowspan=2, padx=10, sticky="nsew")
```

```
        self.azResolution = tk.StringVar(value="")
        self.elStart = tk.StringVar(value="")
        self.elStop = tk.StringVar(value="")
        self.elResolution = tk.StringVar(value="")
        self.setTime = tk.StringVar(value="")

        self.isPaused = tk.BooleanVar(value=False)

    def create_widgets(self):
        frame = self.frame

        validacion = frame.register(self.validar_campo)

        # Titulo
```

```
        # Titulo
        label_superior = tk.Label(
            frame, text="Control automático de motores", bg="#CCCCCC", fg="black"
        )
        label_superior.grid(row=0, columnspan=4, padx=5, sticky="nsew")

        # Azimuth Extents

        az_extent_label = tk.Label(frame, text="Az Extents", relief="solid")
        az_extent_label.grid(row=1, column=0, padx=5, pady=5, sticky="nsew")

        # grafico1_frame = tk.Label(frame, text="Grafico", relief="solid")
        grafico1_frame = tk.Canvas(frame, width=150, height=150, bg="white")
        grafico1_frame.grid(row=2, column=0, rowspan=3, padx=5, pady=5)

        az_start_label = tk.Label(frame, text="Az Start", relief="solid")
        az_start_label.grid(row=2, column=1, padx=5, pady=5, sticky="nsew")

        az_start_label_value = tk.Entry(
            frame,
            textvariable=self.azStart,
            validate="key",
            validatecommand=(validacion, "%P"),
        )
        az_start_label_value.grid(row=2, column=2, padx=5, pady=5, sticky="nsew")

        az_stop_label = tk.Label(frame, text="Az Stop", relief="solid")
        az_stop_label.grid(row=3, column=1, padx=5, pady=5, sticky="nsew")
```

```
        az_stop_label_value = tk.Entry(
            frame,
            textvariable=self.azStop,
            validate="key",
            validatecommand=(validacion, "%P"),
        )
        az_stop_label_value.grid(row=3, column=2, padx=5, pady=5, sticky="nsew")

        az_resolution_label = tk.Label(frame, text="Az Resolution", relief="solid")
        az_resolution_label.grid(row=4, column=1, padx=5, pady=5, sticky="nsew")

        az_resolution_label_value = tk.Entry(
            frame,
            textvariable=self.azResolution,
            validate="key",
            validatecommand=(validacion, "%P"),
        )

        az_resolution_label_value.grid(row=4, column=2, padx=5, pady=5, sticky="nsew")

        # EL Extents
```

```
    # EL Extents

    el_extent_label = tk.Label(frame, text="EL Extents", relief="solid")
    el_extent_label.grid(row=5, column=0, padx=5, pady=5, sticky="nsew")

    grafico2_frame = tk.Canvas(frame, width=150, height=150, bg="white")
    grafico2_frame.grid(row=6, column=0, rowspan=3, padx=5, pady=5)

    # Dibujar el gráfico de radar
    self.graficos.setLabel(grafico1_frame, grafico2_frame)

    # self.dibujar_grafico_radar(grafico1_frame, centro_x, centro_y, radios, valores)
    # self.dibujar_grafico_AZ()

    el_start_label = tk.Label(frame, text="EL Start", relief="solid")
    el_start_label.grid(row=6, column=1, padx=5, pady=5, sticky="nsew")

    el_start_label_value = tk.Entry(
        frame,
        textvariable=self.elStart,
        validate="key",
        validatecommand=(validacion, "%P"),
    )
                          validacion: str = frame.register(self.validar_campo)

    el_start_label_value.grid(row=6, column=2, padx=5, pady=5, sticky="nsew")

    el_stop_label = tk.Label(frame, text="EL Stop", relief="solid")
    el_stop_label.grid(row=7, column=1, padx=5, pady=5, sticky="nsew")
```

```
    el_stop_label_value = tk.Entry(
        frame,
        textvariable=self.elStop,
        validate="key",
        validatecommand=(validacion, "%P"),
    )

    el_stop_label_value.grid(row=7, column=2, padx=5, pady=5, sticky="nsew")

    el_resolution_label = tk.Label(frame, text="EL Resolution", relief="solid")
    el_resolution_label.grid(row=8, column=1, padx=5, pady=5, sticky="nsew")

    el_resolution_label_value = tk.Entry(
        frame,
        textvariable=self.elResolution,
        validate="key",
        validatecommand=(validacion, "%P"),
    )
```

```
    set_time_label = tk.Label(frame, text="Tiempo", relief="solid")
    set_time_label.grid(row=9, column=0, padx=5, pady=5, sticky="nsew")

    set_time_label_value = tk.Entry(
        frame,
        textvariable=self.setTime,
        validate="key",
        validatecommand=(validacion, "%P"),
    )

    set_time_label_value.grid(
        row=9, column=1, columnspan=2, padx=5, pady=5, sticky="nsew"
    )

    # botones

    start_process_button = tk.Button(
        frame, text="START PROCESS", command=self.start
    )
    start_process_button.grid(row=10, column=0, padx=5, pady=5, sticky="nsew")
    start_process_button = tk.Button(
        frame, text="PAUSE PROCESS", command=self.pause
    )
    start_process_button.grid(row=10, column=1, padx=5, pady=5, sticky="nsew")

    stop_process_button = tk.Button(frame, text="STOP PROCESS", command=self.stop)
    stop_process_button.grid(row=10, column=2, padx=5, pady=5, sticky="nsew")

    for columna in range(4):
        frame.grid_columnconfigure(columna, weight=1)
```

```
# Funcion para validar numero en Entry

def validar_campo(self, input_text):
    if input_text == "" or input_text == "-":
        return True
    else:
        return self.validar_entero(input_text)

def validar_entero(self, input_text):
    try:
        int(input_text)
        return True
    except ValueError:
        messagebox.showerror("Error", "Ingresa sólo números enteros.")
        return False
```

```
    def getIntNotNull(self, num):
        if num == "" or num == "0":
            return 0
        else:
            try:
                return int(num)
            except ValueError:
                return 0

    # Funciones para los botones

    def pausarMotores(self):
        if self.motores.movimiento:
            self.motores.pauseMotors()
            self.movimientoMotores.join()

    def moveAutomaticMotors(self):
        self.motores.movimiento_automatico(
            self.getIntNotNull(self.azStart.get()),
            self.getIntNotNull(self.azStop.get()),
            self.getIntNotNull(self.elStart.get()),
            self.getIntNotNull(self.elStop.get()),
            self.getIntNotNull(self.azResolution.get()),
            self.getIntNotNull(self.elResolution.get()),
            self.getIntNotNull(self.setTime.get()),
            self.medicion,
        )
```

```
    def start(self):
        if not self.temporizador.contando:
            self.contador = threading.Thread(
                target=self.temporizador.iniciar_temporizador
            )
            self.contador.start()

            azRes = self.getIntNotNull(self.azResolution.get())
            elRes = self.getIntNotNull(self.elResolution.get())

            if self.motores.movimiento:
                azStart = self.getIntNotNull(self.azStart.get())
                elStart = self.getIntNotNull(self.elStart.get())
            else:
                azStart = self.getIntNotNull(self.motores.currentAZ.get())
                elStart = self.getIntNotNull(self.motores.currentEL.get())

            self.pausarMotores()
```

```
        if azRes != 0 and elRes != 0:
            # Iniciamos temporizador

            if not self.motores.movimiento:
                self.movimientoMotores = threading.Thread(
                    target=self.moveAutomaticMotors
                )
                self.movimientoMotores.start()

            else:
                messagebox.showerror("Error", "Motores están en movimiento")

        else:
            messagebox.showerror("Error", "Las resoluciones no deben ser cero")

    def pause(self):
        # self.motores.isPaused.set(True)

        if self.temporizador.contando:
            self.temporizador.pausar_temporizador()
            self.contador.join()

        if self.motores.movimiento:
            self.motores.pauseMotors()
            # self.movimientoMotores.join()

        print("proceso pausado")
```

```
        print("proceso pausado")

    def stop(self):
        # self.motores.mover_motor(1, 0)
        # self.motores.mover_motor(1, 0)
        # self.motores.isPaused.set(True)

        self.temporizador.detener_temporizador()
        if self.temporizador.contando:
            self.contador.join()

        self.motores.stopMotors()
        # self.movimientoMotores.join()

        print("proceso detenido")
```

Apéndice B. Código del control manual (controlManual.py)

```
import threading
import tkinter as tk
from tkinter import messagebox


class ControlManual(tk.Frame):
    def __init__(self, _, motores, graficos):
        super().__init__()

        # Instanciar motores

        self.motores = motores
        # Instanciar graficos
        self.graficos = graficos

        # Crear el Frame
        self.frame = tk.Frame(_, bg="#CCCCCC")
        self.frame.grid(row=1, column=0, padx=10, pady=20, sticky="nsew")

        self.create_variables()
        self.create_widgets()

    def create_variables(self):
        # Crear una variable de tipo StringVar
        self.azJogTo = tk.StringVar(value="")
        self.elJogTo = tk.StringVar(value="")

    def create_widgets(self):
        frame = self.frame

        validacion = frame.register(self.validar_campo)
```

```
        # Titulo
        label_superior = tk.Label(
            frame, text="Control manual de motores", bg="#CCCCCC", fg="black"
        )
        label_superior.grid(row=0, columnspan=4, padx=5, pady=5, sticky="nsew")

        # Azimuth Jog delta
        azimuth_label = tk.Label(frame, text="Azimuth Jog delta", relief="solid")
        azimuth_label.grid(row=1, column=0, columnspan=2, padx=5, pady=5, sticky="nsew")

        az_jogto_label = tk.Label(frame, text="AZ Jog to", relief="solid")
        az_jogto_label.grid(row=2, column=0, padx=5, pady=5, sticky="nsew")
```

```
    az_jogto_entry = tk.Entry(
        frame,
        textvariable=self.azJogTo,
        validate="key",
        validatecommand=(validacion, "%P"),
    )
    az_jogto_entry.grid(row=2, column=1, padx=5, pady=5, sticky="nsew")

    # Elevation Jog delta

    elevation_label = tk.Label(frame, text="Elevation Jog delta", relief="solid")
    elevation_label.grid(
        row=1, column=2, columnspan=2, padx=5, pady=5, sticky="nsew"
    )

    az_jogto_label = tk.Label(frame, text="EL Jog to", relief="solid")
    az_jogto_label.grid(row=2, column=2, padx=5, pady=5, sticky="nsew")

    az_jogto_entry = tk.Entry(
        frame,
        textvariable=self.elJogTo,
        validate="key",
        validatecommand=(validacion, "%P"),
    )
    az_jogto_entry.grid(row=2, column=3, padx=5, pady=5, sticky="nsew")

    # botones

    az_button = tk.Button(frame, text="MOVE TO AZ", command=self.moveToAZPosition)
    az_button.grid(row=3, column=0, columnspan=2, padx=5, pady=5, sticky="nsew")
```

```
    el_button = tk.Button(frame, text="MOVE TO EL", command=self.moveToELPosition)
    el_button.grid(row=3, column=2, columnspan=2, padx=5, pady=5, sticky="nsew")

    zero_positioner_button = tk.Button(
        frame, text="ZERO POSITIONER", command=self.zeroPositioner
    )
    zero_positioner_button.grid(
        row=4, column=0, columnspan=2, padx=5, pady=5, sticky="nsew"
    )

    set_zero_positioner_button = tk.Button(
        frame, text="SET ZERO", command=self.setZeroPos
    )
    set_zero_positioner_button.grid(
        row=4, column=2, columnspan=2, padx=5, pady=5, sticky="nsew"
    )
```

```python
    # Funcion para validar numero en Entry

    def validar_campo(self, input_text):
        if input_text == "" or input_text == "-":
            return True
        else:
            return self.validar_entero(input_text)

    def validar_entero(self, input_text):
        try:
            int(input_text)
            return True
        except ValueError:
            messagebox.showerror("Error", "Ingresa sólo números enteros.")
            return False

    # Funciones para los botones

    def pausarMotores(self):
        if self.motores.movimiento:
            self.motores.pauseMotors()
            self.movimientoMotores.join()

    def moveAzMotor(self):
        self.motores.movimiento = True
        self.motores.mover_motor(1, int(self.azJogTo.get()))
        self.motores.movimiento = False

    def moveToAZPosition(self):
        self.pausarMotores()
        if self.validar_entero(self.azJogTo.get()):
```

```python
    def moveToELPosition(self):
        self.pausarMotores()
        if self.validar_entero(self.elJogTo.get()):
            # Mueve motores
            self.movimientoMotores = threading.Thread(target=self.moveELMotor)
            self.movimientoMotores.start()

    def zeroPositioner(self):
        self.movimientoMotores = threading.Thread(target=self.motores.stopMotors)
        self.movimientoMotores.start()

    def setZeroPos(self):
        self.motores.currentAZ.set(0)
        self.motores.currentEL.set(0)
        self.graficos.dibujar_grafico(1, 0)
        self.graficos.dibujar_grafico(2, 0)
```

Apéndice C. Código de los datos (datos.py)

```python
import tkinter as tk

import time


class Datos:
    def __init__(self, master):
        self.master = master
        self.contador = 0
        self.tiempo = 0

    def iniciar_contador(self, tiempo):
        self.contador = 0

        self.tiempo = tiempo
        self.actualizar_contador()

    def detener_contador(self):
        self.master.after_cancel(self.after_id)

    def actualizar_contador(self):
        self.contador += 1

        if self.contador < self.tiempo:  # Detener el contador después de 10 segundos
            print("obteniendo datos")
            self.after_id = self.master.after(1000, self.actualizar_contador)
```

Apéndice D. Código de los gráficos (gráficos.py)

```python
import math
import tkinter as tk


class Graficos:
    def __init__(self):
        # Coordenadas del centro del círculo
        global centro_x
        global centro_y
        centro_x = 75
        centro_y = 75

        global color_line
        color_line = "red"
        self.azRes = 0
        self.azStart = 0
```

```
        self.azStart = 0
        self.azStop = 0

        self.elRes = 0
        self.elStart = 0
        self.elStop = 0

        self.azPoints = []
        self.elPoints = []

    def setLabel(self, labelAZ, labelEL):
        self.labelAZ = labelAZ
        self.labelEL = labelEL

        self.dibujar_base_grafico(self.labelAZ)
        self.dibujar_base_grafico(self.labelEL)

    def dibujar_base_grafico(self, canvas):
        # Radio del círculo
        for i in range(10):
            radio = 15 * (i + 1)

            # Dibujar el círculo
            canvas.create_oval(
                centro_x - radio,
                centro_y - radio,
                centro_x + radio,
                centro_y + radio,
                outline="black",
            )
```

```
    def dibujar_base_grafico(self, canvas):
        # Radio del círculo
        for i in range(10):
            radio = 15 * (i + 1)

            # Dibujar el círculo
            canvas.create_oval(
                centro_x - radio,
                centro_y - radio,
                centro_x + radio,
                centro_y + radio,
                outline="black",
            )

        canvas.create_line(0, centro_y, 200, centro_y)
        canvas.create_line(centro_x, 0, centro_x, 200)
```

```
        # Dibujar los segmentos del gráfico
        for i in range(4):
            # Calcular las coordenadas de inicio y fin del segmento
            x1 = centro_x + (200 * math.cos(math.radians(i * angulo - 45)))
            y1 = centro_y + (200 * math.sin(math.radians(i * angulo - 45)))
            x2 = centro_x + (75 * 0.6 * math.cos(math.radians(i * angulo - 45)))
            y2 = centro_y + (75 * 0.6 * math.sin(math.radians(i * angulo - 45)))

            # Dibujar el segmento
            canvas.create_line(x1, y1, x2, y2)

    def dibujar_grafico(self, tipo, angulo):
        if tipo == 1:
            self.labelAZ.delete("all")
            self.dibujar_base_grafico(self.labelAZ)
            self.crear_linea(self.labelAZ, self.azPoints, angulo)
        else:
            self.labelEL.delete("all")
            self.dibujar_base_grafico(self.labelEL)
            self.crear_linea(self.labelEL, self.elPoints, angulo)

    def crear_linea(self, canvas, lista, angulo):
        lista = [x for x in lista if x < angulo]
        lista.append(angulo)
        for i in lista:
            x2 = centro_x + (150 * math.sin(math.radians(i + 90)))
            y2 = centro_x + (150 * math.cos(math.radians(i + 90)))
            canvas.create_line(centro_x, centro_y, x2, y2, fill=color_line)
```

Apéndice E. Código de la ventana principal (main.py)

```
import time
import tkinter as tk

from controlManual import *
from automaticControl import AutomaticControl
from graficos import Graficos
from temporizador import Temporizador
from motores import Motores
from medicion import Medicion

from measurementControl import *
```

```
# Asignamos el root window con un tamaño de 1200x500, asumiendo que serán 3 frames en una fila
ventana = tk.Tk()
ventana.geometry("1200x550")
ventana.title("Proyecto")
ventana.configure(bg="white")

# Crear el Label encima de la fila de los frames
label_superior = tk.Label(ventana, text="Proyecto Antena", bg="white", fg="black")
label_superior.pack(side=tk.TOP, fill=tk.X, pady=10)

# Crear un frame que contenga los tres frames
contenedor_frames = tk.Frame(ventana, bg="white")
contenedor_frames.pack(side=tk.TOP, fill=tk.BOTH, expand=True)


# Iniciamos graficos
graficos = Graficos()

# Iniciamos motores
motores = Motores(ventana, graficos)

# Iniciamos temporizador
temporizador = Temporizador()

# Iniciamos medicion
medicion = Medicion()

# Crear los tres frames dentro del contenedor_frames

# Crear instancia de MiWidget y pasarle el contenedor_frames
```

```
frame1 = ControlManual(contenedor_frames, motores, graficos)
frame2 = MeasurementControl(contenedor_frames, motores, temporizador, medicion)
frame3 = AutomaticControl(contenedor_frames, motores, temporizador, graficos, medicion)

# Configurar el peso de las columnas para la expansión equitativa
contenedor_frames.grid_columnconfigure(0, weight=1)
contenedor_frames.grid_columnconfigure(1, weight=1)

ventana.mainloop()

# Apagar los motores y cerrar la conexión con el LabJack U3
motores.apagado()
```

Apéndice F. Código de la medición manual (measurementControl.py)

```python
import tkinter as tk
from tkinter import messagebox


class MeasurementControl(tk.Frame):
    def __init__(self, _, motores, temporizador, medicion):
        super().__init__()

        # Instanciar motores
        self.motores = motores

        # Instanciar medicion
        self.medicion = medicion

        # Crear el Frame
        self.frame = tk.Frame(_, bg="#CCCCCC")
        self.frame.grid(row=0, column=0, padx=10, sticky="nsew")

        self.create_variables()
        self.create_widgets(temporizador)

    def create_variables(self):
        # Crear una variable de tipo StringVar

        self.currentTime = tk.StringVar(value="00:00:00")

    def create_widgets(self, temporizador):
        frame = self.frame

        validacion = frame.register(self.validar_numero)
```

```python
        validacion = frame.register(self.validar_numero)

        # Titulo
        label_superior = tk.Label(
            frame, text="Control de mediciones", bg="#CCCCCC", fg="black"
        )
        label_superior.grid(row=0, columnspan=3, padx=5, sticky="nsew")

        # Tiempo transcurrido

        current_time_label = tk.Label(frame, text="Tiempo transcurrido", relief="solid")
        current_time_label.grid(row=1, column=0, padx=5, pady=(5, 0), sticky="nsew")
```

```
        current_time_label_value = tk.Label(frame, text="00:00:00", relief="solid")
        current_time_label_value.grid(row=2, column=0, padx=5, sticky="nsew")
        temporizador.setLabel(current_time_label_value)

        # Current AZ POS

        current_az_pos_label = tk.Label(frame, text="Current AZ POS", relief="solid")
        current_az_pos_label.grid(row=1, column=1, padx=5, pady=(5, 0), sticky="nsew")

        current_az_pos_label_value = tk.Label(
            frame, textvariable=self.motores.currentAZ, relief="solid"
        )
        current_az_pos_label_value.grid(row=2, column=1, padx=5, sticky="nsew")

        # Tiempo transcurrido

        current_el_pos_label = tk.Label(frame, text="Current EL POS", relief="solid")
        current_el_pos_label.grid(row=1, column=2, padx=5, pady=(5, 0), sticky="nsew")

        current_el_pos_label_value = tk.Label(
            frame, textvariable=self.motores.currentEL, relief="solid"
        )
        current_el_pos_label_value.grid(row=2, column=2, padx=5, sticky="nsew")

        # Frecuencia inicial

        start_freq_label = tk.Label(frame, text="Start Freq", relief="solid")
        start_freq_label.grid(row=3, column=0, padx=5, pady=(5, 0), sticky="nsew")

        start_freq_entry = tk.Entry(
            frame,
```

```
            textvariable=self.medicion.startFreq,
            validate="key",
            validatecommand=(validacion, "%P"),
        )
        start_freq_entry.grid(row=4, column=0, padx=5, sticky="nsew")

        # Frecuencia final
        stop_freq_label = tk.Label(frame, text="Stop Freq", relief="solid")
        stop_freq_label.grid(row=3, column=1, padx=5, pady=(5, 0), sticky="nsew")

        stop_freq_entry = tk.Entry(
            frame,
            textvariable=self.medicion.stopFreq,
            validate="key",
            validatecommand=(validacion, "%P"),
        )
```

```
        # Frecuencia central
        central_freq_label = tk.Label(frame, text="Central Freq", relief="solid")
        central_freq_label.grid(row=3, column=2, padx=5, pady=(5, 0), sticky="nsew")

        central_freq_entry = tk.Entry(
            frame,
            textvariable=self.medicion.centralFreq,
            validate="key",
            validatecommand=(validacion, "%P"),
        )
        central_freq_entry.grid(row=4, column=2, padx=5, sticky="nsew")

        # No Puntos
        points_label = tk.Label(frame, text="Points (3-1001)", relief="solid")
        points_label.grid(row=5, column=0, padx=5, pady=(5, 0), sticky="nsew")

        points_entry = tk.Entry(
            frame,
            textvariable=self.medicion.points,
            validate="key",
            validatecommand=(validacion, "%P"),
        )
        points_entry.grid(row=6, column=0, padx=5, sticky="nsew")

        # botones

        direct_position_button = tk.Button(frame, text="Accion", command=self.accion)
        direct_position_button.grid(
            row=7, column=0, columnspan=3, padx=5, pady=5, sticky="nsew"
        )
```

```
        for columna in range(3):
            frame.grid_columnconfigure(columna, weight=1)

    # Funcion para validar numero en Entry
    def validar_numero(self, input_text):
        if input_text.isdigit() or input_text == "":
            return True
        else:
            messagebox.showerror("Error", "Ingresa solo números.")
            return False

    # Funciones para los botones
    def accion(self):
        print("accion")
        n_muestras = 10
        for i in range(n_muestras):
            self.medicion.obtenerDatos(
                    self.motores.currentAZ.get(), self.motores.currentEL.get(), False
                    )
        self.medicion.sonido()
```

Apéndice G. Código para la medición en el VNA (medición.py)

```python
import csv
import time
import tkinter as tk

import pyvisa as vi
import numpy as np
import winsound

class Medicion:
    def __init__(self):
        self.create_variables()
        self.initCSV()
        # Inicializa la comunicación VISA
        rm = vi.ResourceManager()
        print(rm.list_resources())
        self.vna = rm.open_resource("GPIB0::17::INSTR")

        # Imprime la identificacion del instrumento
        print("Datos del dispositivo conectado: " + str(self.vna.query("*IDN?")))

        print("Comenzando las configuraciones...")

    def create_variables(self):
        # Crear una variable de tipo StringVar

        self.startFreq = tk.IntVar(value=0)
        self.stopFreq = tk.IntVar(value=0)
        self.centralFreq = tk.IntVar(value=0)
        self.points = tk.IntVar(value=0)
```

```python
    def obtenerDatos(self, currentAz, currentEL, isGeneralReport):
        # Se crean las listas para guardar los valores del parametro S21 y frecuencia
        datos_magnitud = []
        datos_freq = []
        self.s21_data = []
        freqIni = self.getIntNotNull(self.startFreq.get())
        freqFin = self.getIntNotNull(self.stopFreq.get())
        points = self.getIntNotNull(self.points.get())
        # Configura el rango de frecuencia
        self.vna.write(f":SENS1:FREQ:STAR {freqIni}")
        self.vna.write(f":SENS1:FREQ:STOP {freqFin}")
        self.vna.write(f":SENS1:SWE:POIN {points}")  # NUMERO DE PUNTOS
        self.vna.write(":CALC1:FORM POL")  # Seleccionar el formato tipo polar
        self.vna.write(":CALC1:PAR1:DEF S21")  # Definir el parametro S21
```

```
    # Se recorre 6 veces para que el VNA de los valores correctos
    cantidad_muestras = 106

    for i in range(cantidad_muestras):

        self.vna.write(":CALC1:DATA:FDAT?")
        datos_magnitud.append(self.vna.read())

    self.vna.write(":SENS1:FREQ:DATA?")
    datos_freq.append(self.vna.read())

    # # Se toma los valores de la ultima medicion que da el VNA para asegurar los datos obtenidos
    del datos_freq[0:5]
    del datos_magnitud[0:5]
    n_muestras = 100

    freq = datos_freq
    mag = datos_magnitud

    self.s21_data.append(
        (freq, mag, currentAz, currentEL)
    )  # Guarda la frecuencia y el valor del parámetro S21
     # Guarda la frecuencia y el valor del parámetro S21

    if isGeneralReport:
        self.guardarDatos()
    else:
        self.guardarDatosSingles(currentAz, currentEL)
```

```
def getIntNotNull(self, num):
    if num == "" or num == "0":
        return 0
    else:
        try:
            return int(num)
        except ValueError:
            return 0

def initCSV(self):
    # Guardar los resultados en un archivo CSV
    with open(self.output_file_name, "w", newline="") as csvfile:
        writer = csv.writer(csvfile)
        writer.writerow(
            ["Frecuencia (Hz)", "S21 Magnitud (dB)", "Current AZ", "Current EL"]
        )
    print(f"Archivo sobreescrito: {self.output_file_name}")
```

```
    def guardarDatos(self):
        # Guardar los resultados en un archivo CSV o TXT

        with open(self.output_file_name, "a", newline="") as csvfile:
            writer = csv.writer(csvfile)
            for freq, mag, cAz, cEL in self.s21_data:
                writer.writerow([freq, mag, cAz, cEL])

        print(f"Datos guardados en el archivo: {self.output_file_name}")

    def guardarDatosSingles(self, currentAz, currentEL):
        # Guardar los resultados en un archivo CSV o TXT
        output_file = f"resultados_vna_az{currentAz}_el{currentEL}.csv"  # O cambia la extensión a "resu
        with open(output_file, "w", newline="") as csvfile:
            writer = csv.writer(csvfile)
            writer.writerow(
                ["Frecuencia (Hz)", "S21 Magnitud (dB)", "Current AZ", "Current EL"]
            )  # Escribir encabezados
            for freq, mag, cAz, cEL in self.s21_data:
                writer.writerow([freq, mag, cAz, cEL])

        print(f"Datos guardados en el archivo: {output_file}")


    def sonido(self):
        # Para saber cuando termina el programa de ejecutarse añadimos un sonido
        frecuencia = 1500
        duracion = 1000
        winsound.Beep(frecuencia, duracion)
```

Apéndice H. Código que inicia los motores (motores.py)

```
import time
import tkinter as tk
import math
import datos as dt
import u3
class Motores:
    def __init__(self, master, graficos):
        super().__init__()

        self.master = master
        self.graficos = graficos

        self.motorAZ = 1
        self.motorEL = 2
```

```
        self.movimiento = False
        self.wasPaused = False
        self.wasStoped = False
        print("initMotores")

        # Crear una instancia del objeto U3 para el LabJack U3-LV
        self.d = u3.U3()

        # Configurar los pines de salida digital para los motores
        self.d.getFeedback(
            u3.BitDirWrite(0, 0)
        )  # Configurar FIO0 como salida digital (Step+/-) motor 1
        self.d.getFeedback(
            u3.BitDirWrite(1, 0)
        )  # Configurar FIO1 como salida digital (Dir +/-) motor 1
        self.d.getFeedback(
            u3.BitDirWrite(2, 0)
        )  # Configurar FIO2 como salida digital (Step+/-) motor 2
        self.d.getFeedback(
            u3.BitDirWrite(3, 0)
        )  # Configurar FIO3 como salida digital (Dir +/-) motor 2
        self.currentAZ = tk.IntVar(value=0)
        self.currentEL = tk.IntVar(value=0)

        # self.isPaused = tk.BooleanVar(value=False)

    # Definir funciones para controlar los pasos del motor 1
    def paso_der_motor_1(self):
        if self.movimiento:

            self.d.getFeedback(u3.BitStateWrite(1, 1))  # Establecer FIO1 en alto (Dir+)
```

```
    def paso_der_motor_1(self):
        if self.movimiento:

            self.d.getFeedback(u3.BitStateWrite(1, 1))  # Establecer FIO1 en alto (Dir+)
            self.d.getFeedback(u3.BitStateWrite(1, 0))  # Establecer FIO1 en bajo (Dir-)
            self.d.getFeedback(u3.BitStateWrite(0, 1))  # Establecer FIO0 en alto (Step+)
            self.d.getFeedback(u3.BitStateWrite(0, 0))  # Establecer FIO0 en bajo (Step-)

            # print("AZ +")
            time.sleep(0.0005)  # Duración del pulso de paso

    def paso_izq_motor_1(self):
        if self.movimiento:
```

```
    def paso_izq_motor_1(self):
        if self.movimiento:

            self.d.getFeedback(u3.BitStateWrite(1, 0))  # Establecer FIO1 en bajo (Dir+)
            self.d.getFeedback(u3.BitStateWrite(1, 1))  # Establecer FIO1 en alto (Dir-)
            self.d.getFeedback(u3.BitStateWrite(0, 1))  # Establecer FIO0 en alto (Step+)
            self.d.getFeedback(u3.BitStateWrite(0, 0))  # Establecer FIO0 en bajo (Step-)

            # print("AZ -")
            time.sleep(0.0005)  # Duración del pulso de paso

    # Definir funciones para controlar los pasos del motor 2
    def paso_der_motor_2(self):
        if self.movimiento:

            self.d.getFeedback(u3.BitStateWrite(3, 1))  # Establecer FIO3 en alto (Dir+)
            self.d.getFeedback(u3.BitStateWrite(3, 0))  # Establecer FIO3 en bajo (Dir-)
            self.d.getFeedback(u3.BitStateWrite(2, 1))  # Establecer FIO2 en alto (Step+)
            self.d.getFeedback(u3.BitStateWrite(2, 0))  # Establecer FIO2 en bajo (Step-)

            # print("EL +")
            time.sleep(0.0005)  # Duración del pulso de paso

    def paso_izq_motor_2(self):
        if self.movimiento:

            self.d.getFeedback(u3.BitStateWrite(3, 0))  # Establecer FIO3 en bajo (Dir+)
            self.d.getFeedback(u3.BitStateWrite(3, 1))  # Establecer FIO3 en alto (Dir-)
            self.d.getFeedback(u3.BitStateWrite(2, 1))  # Establecer FIO2 en alto (Step+)
            self.d.getFeedback(u3.BitStateWrite(2, 0))  # Establecer FIO2 en bajo (Step-)
```

```
    # Función para apagar los motores
    def apagado(self):

        self.d.getFeedback(
            u3.BitDirWrite(0, 0)
        )  # Configurar FIO0 como salida digital (Step+/-) motor 1
        self.d.getFeedback(
            u3.BitDirWrite(1, 0)
        )  # Configurar FIO1 como salida digital (Dir +/-) motor 1
        self.d.getFeedback(
            u3.BitDirWrite(2, 0)
        )  # Configurar FIO2 como salida digital (Step+/-) motor 2
        self.d.getFeedback(
            u3.BitDirWrite(3, 0)
        )  # Configurar FIO3 como salida digital (Dir +/-) motor 2

        self.d.close()
```

```
    def mover_motor(self, motor_num, angulo):
        print(f"Moviendo el motor {motor_num} a {angulo} grados...")
        # Cálculo de pasos requeridos para el ángulo dado
        if self.movimiento:
            self.graficos.dibujar_grafico(motor_num, angulo)

        numero_pasos = 0

        #pasos = int((angulo / 0.45))  # Suponiendo que el motor tiene 1200 pasos por vuelta com
        if motor_num == 1:
            escalaAZ = 0.1
            oldAz = self.currentAZ.get()

            pasos = int(
                (angulo / escalaAZ)
            )  # Suponiendo que el motor tiene 800 pasos por vuelta completa

            numero_pasos = pasos - (oldAz / escalaAZ)

            # Determinar la dirección del movimiento
            #if (pasos <= 3600) and (pasos >= -3600):  # angulo >= 0 else 0
            if (pasos <= 3600) and (pasos >= -3600):  # angulo >= 0 else 0
                while numero_pasos > 0:
                    if self.movimiento:
                        self.paso_izq_motor_1()
                        numero_pasos -= 1
                        value = int((pasos - numero_pasos) * escalaAZ)
                        self.currentAZ.set(value)
                    else:
                        break
```

```
                while numero_pasos < 0:
                    if self.movimiento:
                        self.paso_der_motor_1()
                        numero_pasos += 1
                        value = int((pasos + numero_pasos) * escalaAZ)
                        self.currentAZ.set(value)
                    else:
                        break

        if motor_num == 2:
            escalaEL = 0.1
            oldEl = self.currentEL.get()
            pasos = int(
                (angulo / escalaEL)
            )  # Suponiendo que el motor tiene 1200 pasos por vuelta completa
```

```
            numero_pasos = pasos - (oldEl / escalaEL)
            # Determinar la dirección del movimiento
            if (pasos <= 3600) and (pasos >= -3600):  # angulo >= 0 else 0
                while numero_pasos > 0:
                #while numero_pasos > 0:
                    if self.movimiento:
                        self.paso_izq_motor_2()
                        numero_pasos -= 1
                        value = int((pasos - numero_pasos) * escalaEL)
                        self.currentEL.set(value)
                    else:
                        break

                while (numero_pasos < 0):
                #while numero_pasos < 0:

                    if self.movimiento:
                        self.paso_der_motor_2()
                        numero_pasos += 1
                        value = int((pasos + numero_pasos) * escalaEL)
                        self.currentEL.set(value)
                    else:
                        break

    def pauseMotors(self):
        self.movimiento = False
        self.wasPaused = True
        print("pausando motores")

    def stopMotors(self):
        self.movimiento = False
```

```
    def stopMotors(self):
        self.movimiento = False
        print("pausamos motores 1 segundos")
        time.sleep(3)
        print("comenzamos el encerado")
        self.movimiento = True
        self.mover_motor(1, 0)
        self.mover_motor(2, 0)
        self.wasStoped = True
        print(" encerado completado")

    # Función para mover un motor a un ángulo específico
```

```python
    # Función para mover un motor a un ángulo específico
    def movimiento_automatico(
        self,
        az_inicial,
        az_final,
        el_inicial,
        el_final,
        res_az,
        res_el,
        tiempo,
        medicion,
    ):
        self.movimiento = True
        self.medicion = medicion

        if not self.wasPaused or self.wasStoped:
            # Calibramos motores
            self.mover_motor(1, 0)
            self.mover_motor(2, 0)

            # Se setea EL al ángulo inicial
            if self.currentEL.get() != el_inicial:
                self.mover_motor(self.motorEL, el_inicial)

            # Se setea AZ al ángulo inicial
            if self.currentAZ.get() != az_inicial:
                self.graficos.dibujar_grafico(1, az_inicial)
                self.mover_motor(self.motorAZ, az_inicial)
            self.wasStoped = False
```

```python
        self.wasPaused = False

        while self.currentEL.get() != el_final and self.movimiento:
            # Se setea el motor AZ al siguiente ángulo
            self.mover_automatic_az(az_inicial, az_final, res_az, tiempo, el_inicial)
            angulo = self.get_automatic_angle(self.currentEL.get(), el_final, res_el)

            # Se setea el motor EL al siguiente ángulo
            self.mover_motor(self.motorEL, angulo)

        # Se setea el motor AZ al siguiente ángulo
        self.mover_automatic_az(az_inicial, az_final, res_az, tiempo, el_inicial)
```

```
    def mover_automatic_az(self, az_inicial, az_final, res_az, tiempo,el_inicial):
        while self.currentAZ.get() != az_final and self.movimiento:
            self.obtenerDatos(tiempo)
            angulo = self.get_automatic_angle(self.currentAZ.get(), az_final, res_az)
            # Se setea el motor al ángulo siguiente ángulo
            self.mover_motor(self.motorAZ, angulo)

        self.obtenerDatos(tiempo)
        # Se setea AZ y EL al ángulo inicial
        self.mover_motor(self.motorAZ, az_inicial)
        #self.mover_motor(self.motorEL, el_inicial)



    def get_automatic_angle(self, angulo, angulo_final, resolucion):
        if abs(angulo_final - angulo) >= resolucion:
            if angulo_final > angulo:
                angulo += resolucion
            if angulo_final < angulo:
                angulo -= resolucion
        else:
            angulo = angulo_final
        return angulo

    # Función para mover un motor a un ángulo específico
    def movimiento_autom(
        self, motor_num, angulo_inicial, angulo_final, resolucion, tiempo
    ):
        print(
            f"Moviendo el motor {motor_num} de {angulo_inicial} a {angulo_final} grados..."
```

```
        # Se setea el motor al ángulo inicial
        self.mover_motor(motor_num, angulo_inicial)

        # Simulación de tiempo de recolección de datos

        datos = dt.Datos(self.master)
        self.obtenerDatos(tiempo)
        # datos.iniciar_contador(tiempo)

        if motor_num == 1:
            angulo = self.currentAZ.get()
        else:
            angulo = self.currentEL.get()
        while angulo != angulo_final:
            if self.movimiento:
                if abs(angulo_final - angulo) >= resolucion:
                    if angulo_final > angulo:
                        angulo += resolucion
```

```
                if angulo_final < angulo:
                    angulo -= resolucion
            else:
                angulo = angulo_final

            # Se setea el motor al ángulo siguiente ángulo
            self.mover_motor(motor_num, angulo)

            self.obtenerDatos(tiempo)
        else:
            break

def obtenerDatos(self, tiempo):
    """TODO: Proceso para obtener datos"""

    for i in range(tiempo):
        if self.movimiento:
            print("obteniendo datos")

            self.medicion.obtenerDatos(
            self.currentAZ.get(), self.currentEL.get(), True
             )

        time.sleep(1)
    self.medicion.sonido()
    print("Fin del temporizador")
```