

Manual de usuario para el GPR de laboratorio



Universidad de los Andes

Proyecto:

Microwave Detection of Improvised Explosive Devices in Colombia (MEDICI)

Autores:

Daniel Julián González Ramírez - dj.gonzalez1203@uniandes.edu.co

Luis Eduardo Quibano Alarcón - le.quibano@uniandes.edu.co

Roberto Bustamante Miller - rbustama@uniandes.edu.co

Junio 2018

Índice

1 Pre-requisitos para el funcionamiento	2
1.1 Instalación de dependencias	2
2 Identificación de los equipos	3
2.1 Analizador Vectorial de Redes	3
2.2 Controlador de Posición	3
2.3 Antenas	4
3 Introducción al software controlador del GPR	5
3.1 Ejecución del programa	5
3.2 Conexión de los equipos	5
3.3 Procedimiento de calibración	8
3.4 Operaciones de pausar, reanudar y abortar	8
3.5 Datos de salida	9
4 Configuración y toma de mediciones	10
4.1 Configuración del VNA	10
4.2 Resolución espacial de las mediciones	10
4.3 Configuración de las trayectorias	11
4.3.1 Trayectoria serpiente	11
4.3.2 Trayectoria manual	13
5 Lectura de los datos	17
6 Bibliografía	21

1. Pre-requisitos para el funcionamiento

Este programa fue desarrollado en Python versión 3 utilizando un computador con sistema operativo Windows.

Los equipos a controlar son la tarjeta xPro que permite mover el posicionador y el analizador vectorial (VNA) Anritsu MS2026C para obtener las señales S21. Si desea controlar otro equipo diferente al mencionado en este manual, por favor revise el manual del programador donde encontrará las secciones de código que sea necesario cambiar.

Para el correcto funcionamiento de la herramienta GPR de laboratorio es necesario contar con los siguientes programas y dependencias:

- Python 3
- pip → Gestor de paquetes Python.
- pyserial → Paquete para el control por puerto serial.
- python-vxi11 → Paquete para el control por protocolo vxi11 hacia el equipo VNA.
- PyQt5 → Paquete para el control de interfaz gráfica en QT.
- numpy → Paquete de computación para Python.
- scipy → Paquete de herramientas y algoritmos matemáticos.
- matplotlib → Paquete para el control de gráficas.

1.1. Instalación de dependencias

Al contar con el gestor de paquetes **pip** es posible instalar la mayoría de dependencias, para ello es necesario que ejecute el siguiente comando desde el terminal de Windows o Linux.

```
pip3 install pyserial PyQt5 numpy scipy matplotlib
```

Este proceso tomará un tiempo, una vez finalizado, descargue la dependencia que permite controlar el equipo VNA, para ello realice los siguientes pasos:

1. Descargue el repositorio del protocolo VXI11 en <https://github.com/python-ivi/python-vxi11>
2. Descomprima los archivos
3. Con el terminal, ingrese a la carpeta del repositorio a través del comando **cd <dir>**
4. Ingrese el comando **python setup.py install**

Con la instalación de estas dependencias ya se podrá ejecutar el programa para el control del posicionador y del VNA.

2. Identificación de los equipos

2.1. Analizador Vectorial de Redes



Figura 1: Analizador vectorial de redes conectado a las antenas y posicionador

El analizador vectorial de redes (VNA) utilizado en el sistema del laboratorio GPR es de marca Anritsu, el número de modelo es MS2026C [2] y es utilizado para obtener los parámetro S12/S21; una de las grandes características de este equipo es la adquisición de datos a una velocidad de $350\mu\text{s}$ por cada punto de barrido; otras de sus características son:

- Rango de frecuencia: 5 KHz - 6 GHz
- Sensibilidad -96 dBm
- Puntos de barrido: 2 - 4001

Este equipo puede ser controlado mediante puerto USB o mediante conexión Ethernet, para el desarrollo de este sistema se utilizó la conexión a Ethernet ya que brinda más ancho de banda y mayor rapidez en la transmisión de datos.

La forma de configurar y controlar el equipo se da mediante el estándar SCPI (*Standard Commands for Programmable Instruments*) para ejecutar funciones del equipo de manera remota.

2.2. Controlador de Posición

El controlador de posición es el sistema que permite mover el sistema GPR y que reporta la posición donde se toman las medidas. El controlador utilizado para el GPR es el Spark-Concepts xPro V3.2 [5]. El controlador xPro permite controlar dos motores paso a paso para mover el GPR en el plano X-Y. Los motores pasos a paso utilizados son motores OpenBuilds NEMA-23 de referencia mt-2303hs280aw-ob [3], el desplazamiento sobre los rieles se genera gracias a los motores, rodamientos y correas GT3. El controlador de posición se alimenta con una fuente de 24 VDC, 21 A. Esta fuente se encarga de energizar la tarjeta xPro y dar corriente a los motores. Las partes que componen

el controlador de posición se muestran en las figura 2

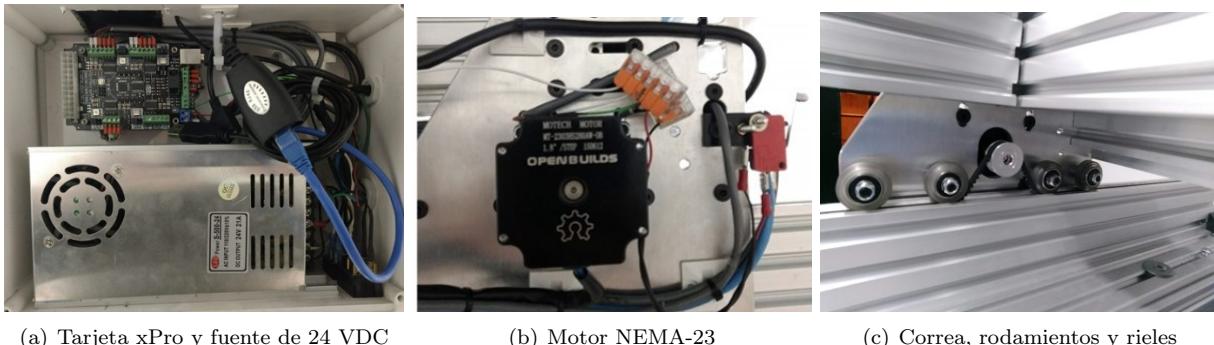


Figura 2: Hardware que compone el controlador de posición

El controlador xPro utiliza el software GRBL [1] para enviar y recibir las señales de los motores. El software GRBL, es utilizado por muchos fabricantes de máquinas de Control Numérico por Computadora (CNC), ya que por medio de comandos es posible controlar detalladamente el comportamiento de los motores paso a paso.

La velocidad máxima de movimiento del GPR es $2000 \text{ mm/min} = 3.33 \text{ cm/s}$. Este parámetro limita la resolución espacial, ya que las mediciones de VNA son tomadas consecutivamente, por lo tanto el espaciamiento máximo de las medidas está limitado por la velocidad máxima.

2.3. Antenas

Las antenas utilizadas para las mediciones son de la marca RF Space, son antenas de ranura cónica [4] lo que permite que sean antenas muy directivas, las ganancias de este tipo de antenas se encuentran entre los 7 dBi y los 11 dBi; a demás el rango de frecuencias es de 600 MHZ - 6 GHZ.

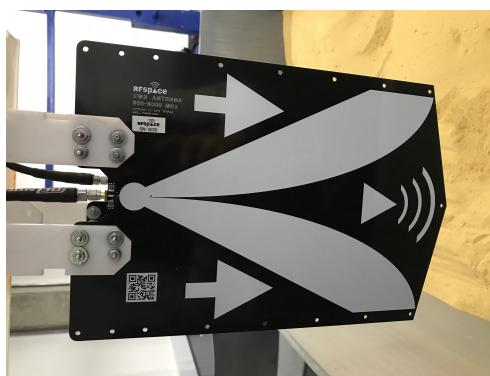


Figura 3: Antenas de Ranura Cónica RF Space

3. Introducción al software controlador del GPR

3.1. Ejecución del programa

El programa controlador del GPR se ejecuta con Python desde una ventana de comandos, utilizando la instrucción: `Python gpr.py`. La instrucción debe ejecutarse desde la carpeta que contiene el archivo `gpr.py`, la versión más actualizada del programa se encuentra adjunta a este documento. Para el correcto funcionamiento, se debe tener en la carpeta del programa, como mínimo, los archivos enmarcados en rojo en la figura 4.



Figura 4: Carpeta del software controlador del GPR

3.2. Conexión de los equipos

Los equipos que deben conectarse al computador son la tarjeta xPro, controladora de posición, y el analizador vectorial de redes, VNA.

La tarjeta xPro cuenta con un puerto USB tipo B. La conexión a la tarjeta debe hacerse entre un puerto USB del computador y el puerto USB de la tarjeta; existen diferentes configuraciones de cables para hacer esta conexión, por ejemplo con un cable UTP con terminales RJ-45 y adaptadores RJ-45 a USB. Una vez se realiza la conexión de la tarjeta xPro al computador, es necesario verificar que puerto de Windows tiene la conexión desde el administrador de dispositivos, como se muestra en la figura 5

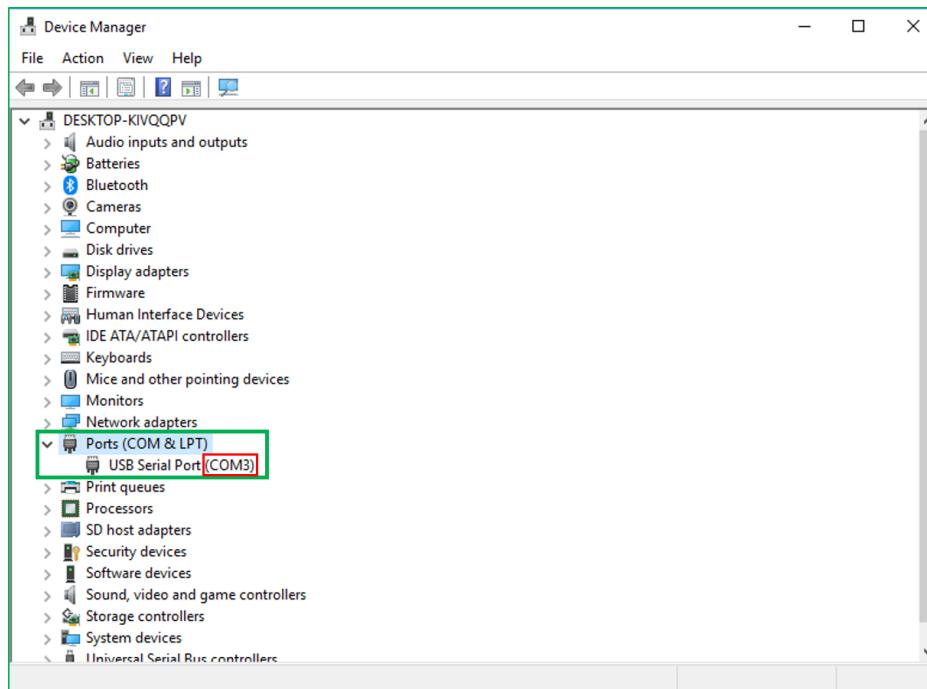


Figura 5: Identificación del número de puerto desde el administrador de dispositivos de Windows

El VNA cuenta con un puerto Ethernet y USB, para lograr obtener medidas con alta tasa de transmisión se sugiere utilizar el puerto Ethernet. La conexión al VNA debe hacerse entre el puerto Ethernet del computador y el puerto Ethernet del VNA mediante un cable UTP con terminales RJ-45. En el VNA y en el computador deben configurarse las direcciones IP tal que los dos equipos se encuentren en la misma red.

Una vez se tiene la conexión física entre los equipos, estos se deben conectar al software controlador del indicando la dirección IP del VNA y el puerto al cual está conectada la tarjeta xPro, como se muestra en la figura 6. Si la conexión es exitosa, los botones de conexión se deshabilitarán y se habilitará el botón de calibración de coordenadas, como se muestra en la figura 7.

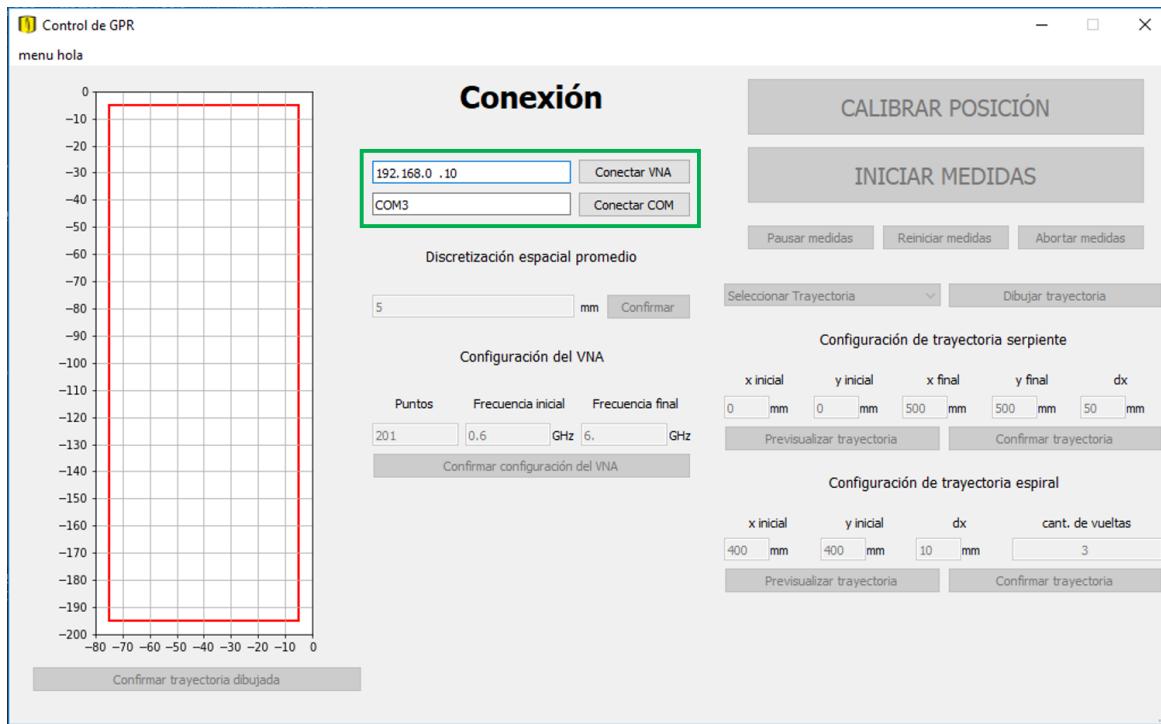


Figura 6: Conexión del VNA y el controlador de posición desde la interfaz gráfica

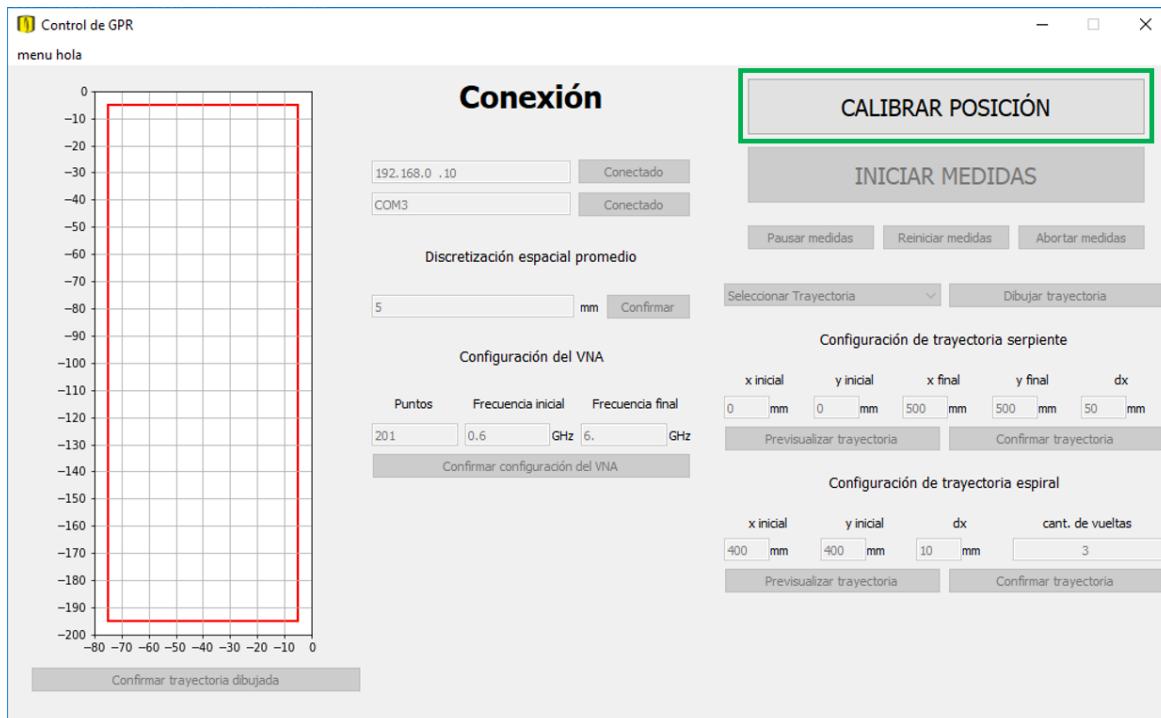


Figura 7: Interfaz gráfica posterior a la conexión del VNA y el controlador de posición

3.3. Procedimiento de calibración

Una vez se ha hecho la conexión del VNA y el controlador de posición, se procede a la calibración del sistema de coordenadas. Para calibrar las coordenadas del GPR, basta con pulsar el botón CALIBRAR POSICIÓN.

Este procedimiento de calibración consiste en llevar al GPR a la posición ($x = 0$, $y = 0$), como aún no se tiene una referencia del sistema de coordenadas, el controlador de posición mueve el GPR en la dirección positiva de ambos ejes hasta encontrar los interruptores de final de carrera. Para que este procedimiento sea exitoso, el movimiento debe ser a una velocidad baja, por lo tanto la calibración puede tardar varios minutos.

Una vez se tiene la calibración del GPR, se pueden tomar múltiples mediciones sin necesidad de calibrar las coordenadas nuevamente. El procedimiento para la configuración y la toma de mediciones se expone en el capítulo 4.

3.4. Operaciones de pausar, reanudar y abortar

En el caso que se esté realizando una medición y sea necesario ponerla en pausa o abortarla, el software del controlador de GPR cuenta con tres botones dispuestos para estas funciones, como se muestra en la figura 8.

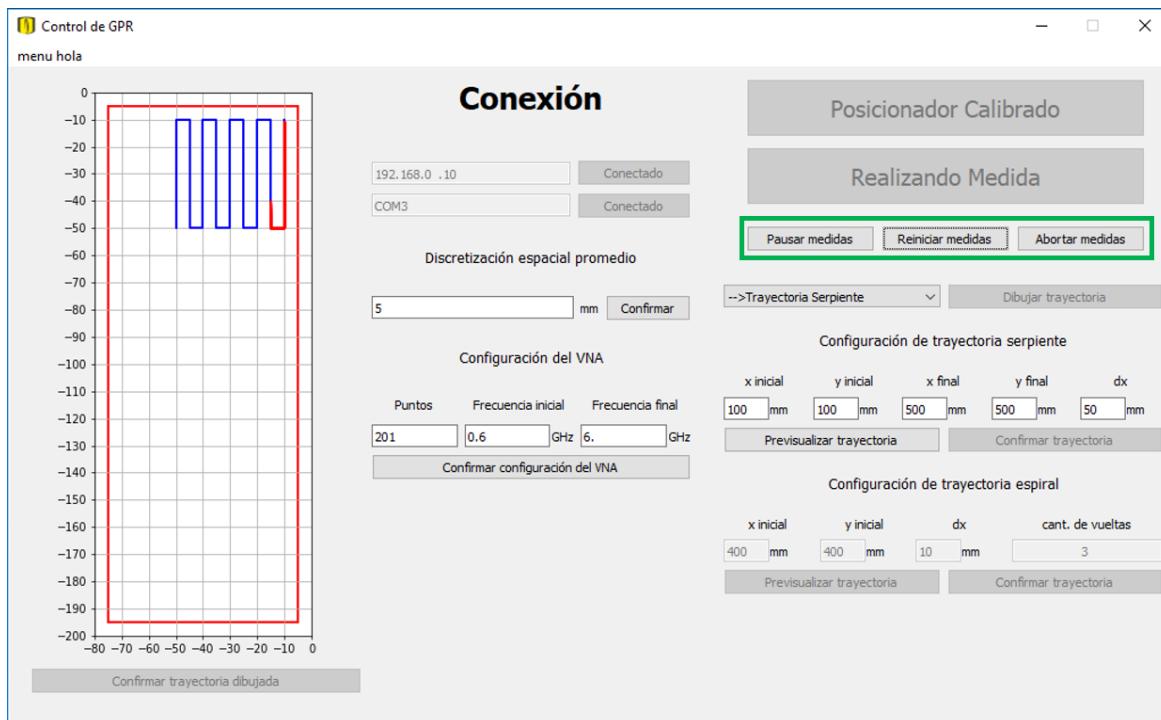


Figura 8: Funciones de pausar, reanudar y abortar en la interfaz gráfica

El botón **Pausar medidas**, detiene el movimiento del GPR sin borrar las instrucciones aún pendientes por

ejecutarse. Desde el estado de pausa, es posible reanudar la medición con la trayectoria ya enviada y también es posible abortar las mediciones. Con el botón **Reiniciar medidas**, se reanuda la trayectoria pausada y la medición de datos. Con el botón **Abortar medidas**, se cancela la trayectoria enviada y se genera un archivo con los datos ya tomados. Después de abortar una medición es posible enviar otra trayectoria al GPR, para la toma de datos.

3.5. Datos de salida

Una vez una trayectoria termina o es abortada, los datos de las mediciones realizadas en la trayectoria son almacenados en un archivo de formato **.mat**. Este archivo contiene la descripción de la trayectoria recorrida, la velocidad a la cual se desplazó el GPR, la configuración del VNA, las mediciones obtenidas del parámetro S21, la posición relacionada con cada medición, y el tiempo que tomó cada medición.

Todos los datos de salida pueden ser procesados con Matlab, gracias al formato utilizado. Desde Matlab se puede obtener la transformada inversa de Fourier de los datos obtenidos por el VNA para formar las trazas en el dominio del tiempo y formar imágenes del perfil del suelo. También se pueden procesar los datos para obtener la distancia promedio entre los puntos, visualizar el parámetro S21 y obtener el tiempo promedio de medición; en el capítulo 5 se da un código de ejemplo para obtener estos datos de las trayectorias tipo serpiente.

4. Configuración y toma de mediciones

4.1. Configuración del VNA

Es posible configurar el VNA para modificar la cantidad de puntos y el rango de frecuencias de medición para el parámetro S21. Estos parámetros pueden configurarse desde la interfaz gráfica, como se muestra en la figura 9.

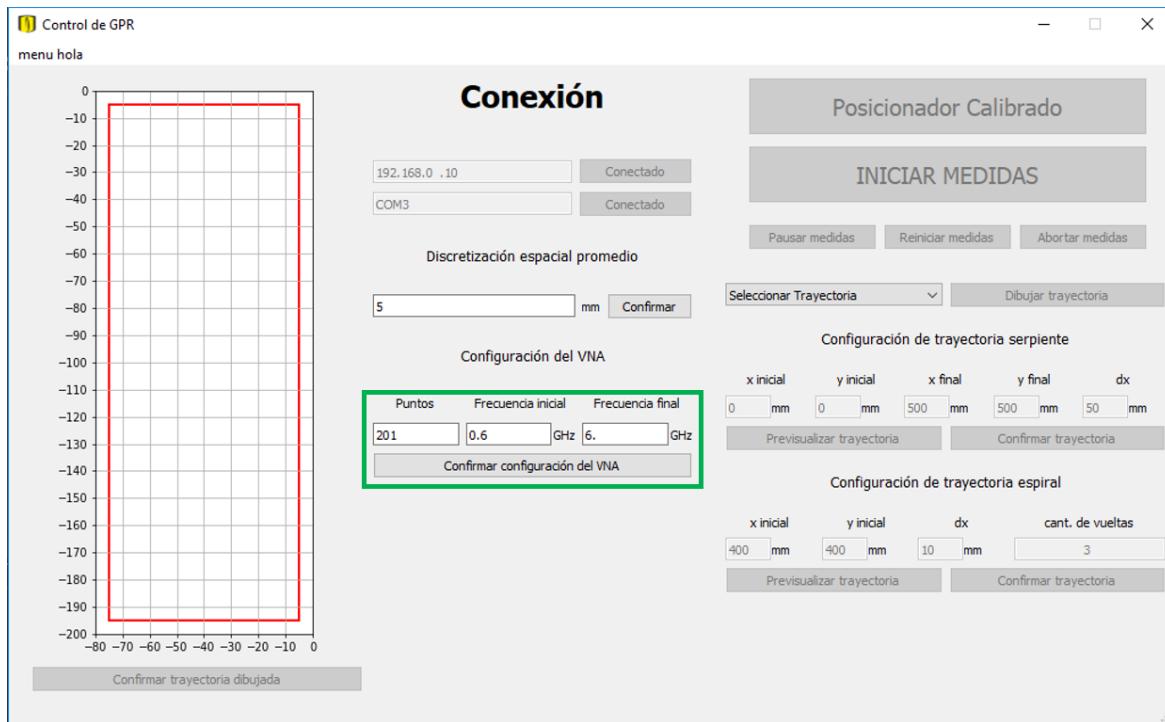


Figura 9: Cambio de la configuración del VNA desde la interfaz gráfica

Al cambiar la cantidad de puntos, también se modifica el tiempo de medición, cada punto se traduce en $350 \mu\text{s}$. Esto modifica la resolución espacial de las mediciones, por lo tanto se recomienda confirmar la discretización espacial deseada, después de modificar la cantidad de puntos.

4.2. Resolución espacial de las mediciones

La resolución espacial determina en qué intervalos de distancia se toma una medición con el VNA. La resolución espacial puede ajustarse desde la interfaz gráfica, como se muestra en la figura 10.

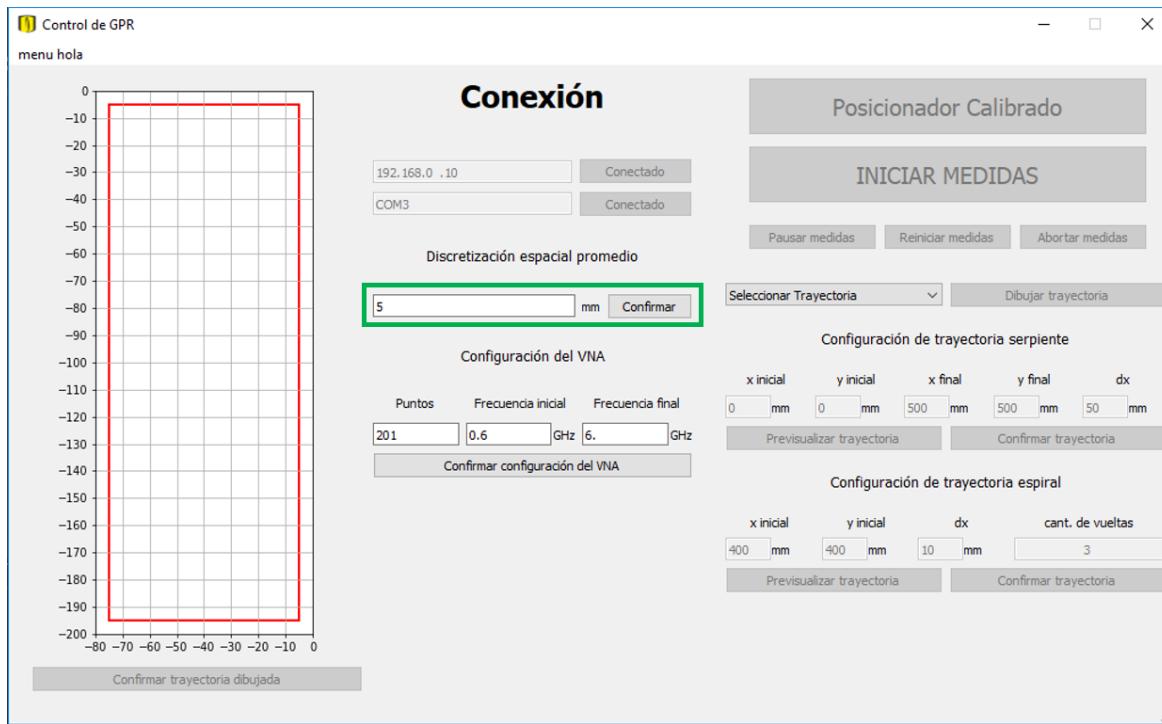


Figura 10: Cambio de la resolución espacial desde la interfaz gráfica

Al cambiar la resolución, se modifica la velocidad de desplazamiento del GPR. La velocidad se calcula a partir de la resolución deseada y el tiempo estimado para la medición del VNA. Como se menciona en el capítulo 2, la velocidad máxima de desplazamiento del GPR es 2000 mm/s, este parámetro limita la distancia máxima entre medidas. Si la resolución seleccionada resulta en una velocidad mayor a la máxima, el software configurará la velocidad como 2000 mm/s y mostrará al usuario la resolución espacial que se utilizará.

4.3. Configuración de las trayectorias

4.3.1. Trayectoria serpiente

En este modo de trayectoria el usuario podrá generar una trayectoria en forma de serpiente con configuraciones variables. Una vez seleccionado en el menú el tipo de trayectoria, se habilitarán las cinco opciones que puede configurar (Ver fig.11) que son: X inicial y final; Y inicial y final; y dx. Este ultimo valor corresponde a la distancia en el eje X que correrá el control de posición para continuar con el desplazamiento en el eje Y.

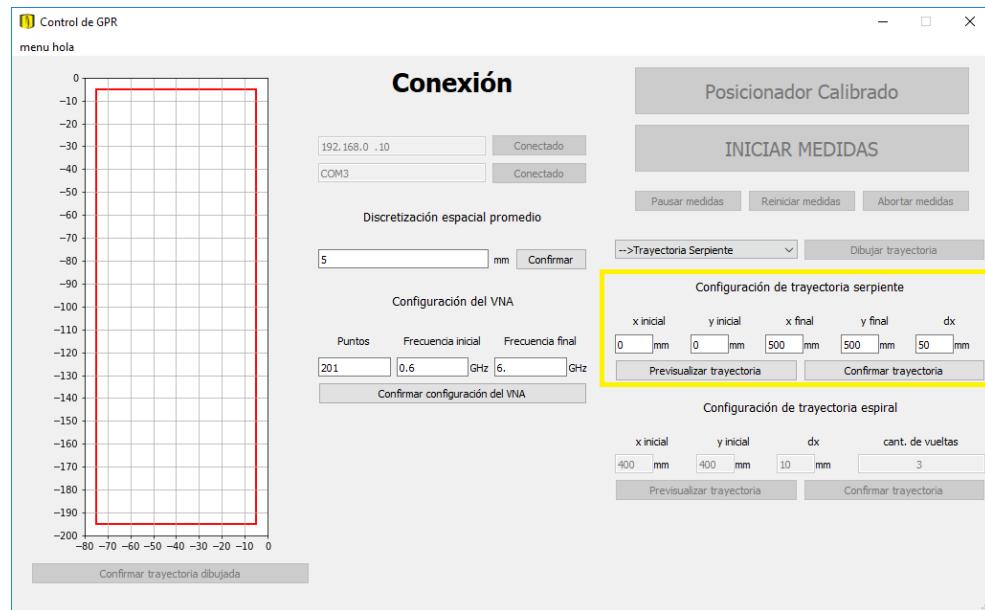


Figura 11: Selección de trayectoria tipo serpiente

Una vez asignados los diferentes valores, puede realizar una visualización de la trayectoria, recordar que el borde rojo de la pre-visualización corresponde a la distancia máxima que el controlador de posición puede recorrer. En la figura 12 vemos un ejemplo de una trayectoria definida en forma de serpiente.

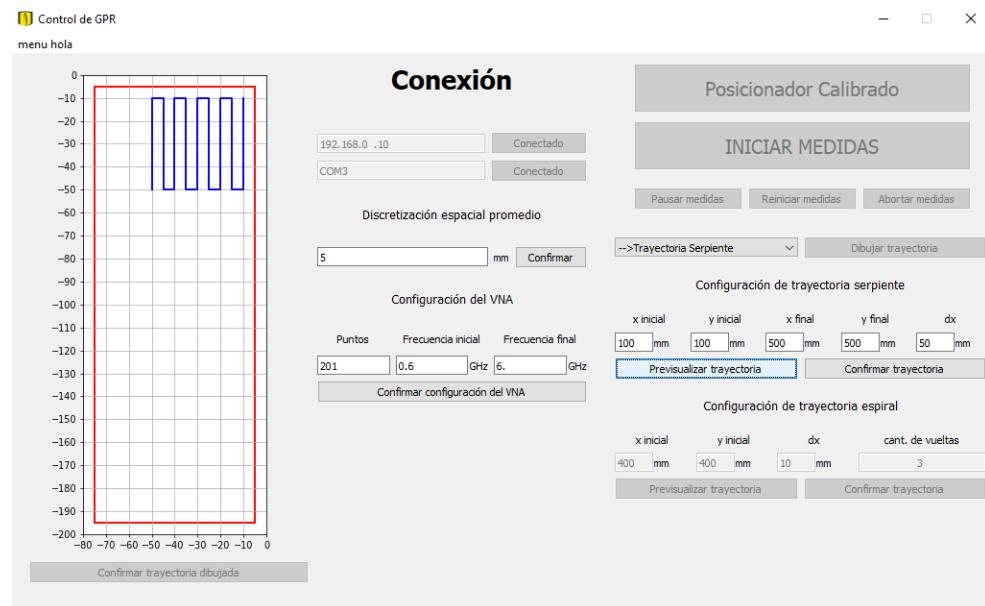


Figura 12: Visualización de la trayectoria tipo serpiente

Cuando se hace clic en el botón “Previsualizar Trayectoria” se habilita el botón que confirma la trayectoria

dibujada; al oprimir “Confirmar Trayectoria” el controlador de posición se dirigirá inmediatamente a la ubicación inicial del trayecto, cuando el controlador de posición se encuentra listo para iniciar las medidas se habilita el botón “Iniciar Medidas”; recuerde que mientras el sistema se encuentra realizando la medición, puede detenerlas, reanudarla o abortarla. Durante el recorrido del controlador de posición, la imagen de la trayectoria definida se irá llenando de color rojo; esto significa que se ha logrado almacenar las medidas en dichos durante el trayecto. En la figura 13 se puede observar el recorrido actual que ha realizado el controlador de posición.

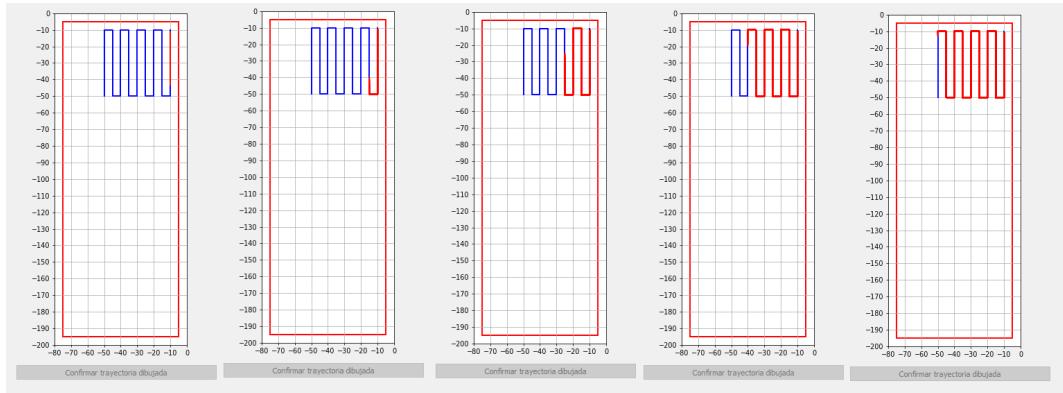


Figura 13: Medidas realizadas por el sistema GPR

4.3.2. Trayectoria manual

Para este modo de trayecto el usuario puede definir el movimiento que va a realizar el sistema GPR, dando la libertad de crear trayectorias complejas. Para crear este trayecto, debe seleccionar “->Trayectoria Manual” (Figura 14) , inmediatamente el sistema le pedirá seleccionar con el cursor dentro del área de dibujo la posición inicial (Ver figura 15).

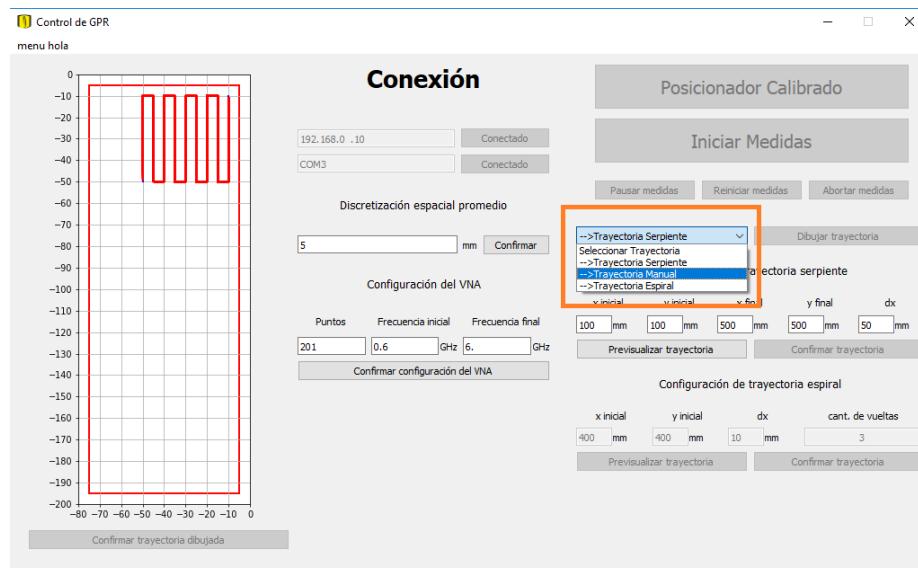


Figura 14: Selección de trayectoria manual en el sistema GPR

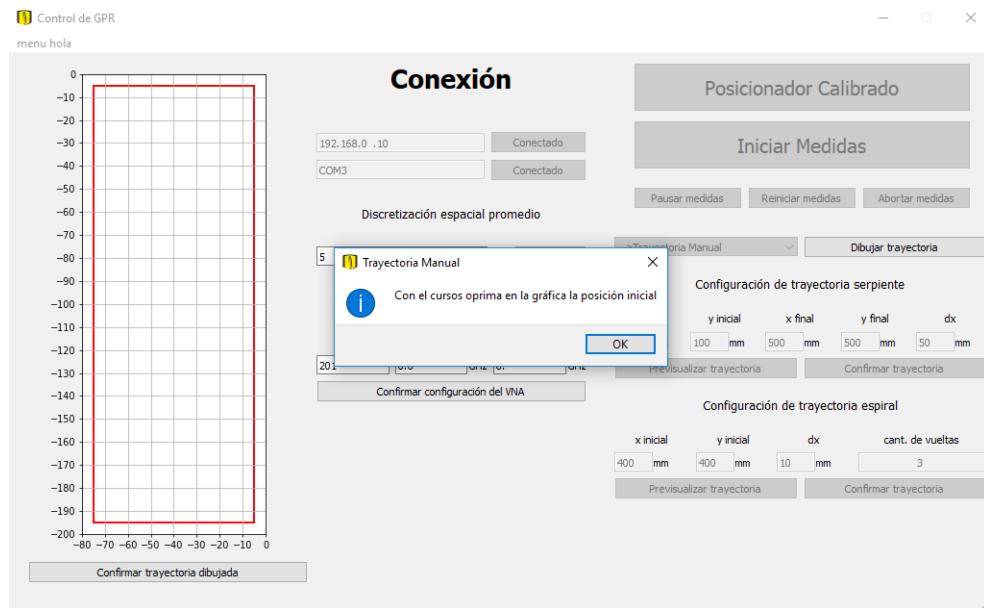


Figura 15: Aviso para iniciar dibujo

Cuando se selecciona el punto inicial y se confirma (Ver figura 16), el controlador de posición se moverá inmediatamente a la posición seleccionada, mientras se realiza el movimiento, el usuario puede oprimir el botón “Dibujar Trayectoria”, con esto puede dibujar el trayecto desde el punto inicial (Ver figura 17).

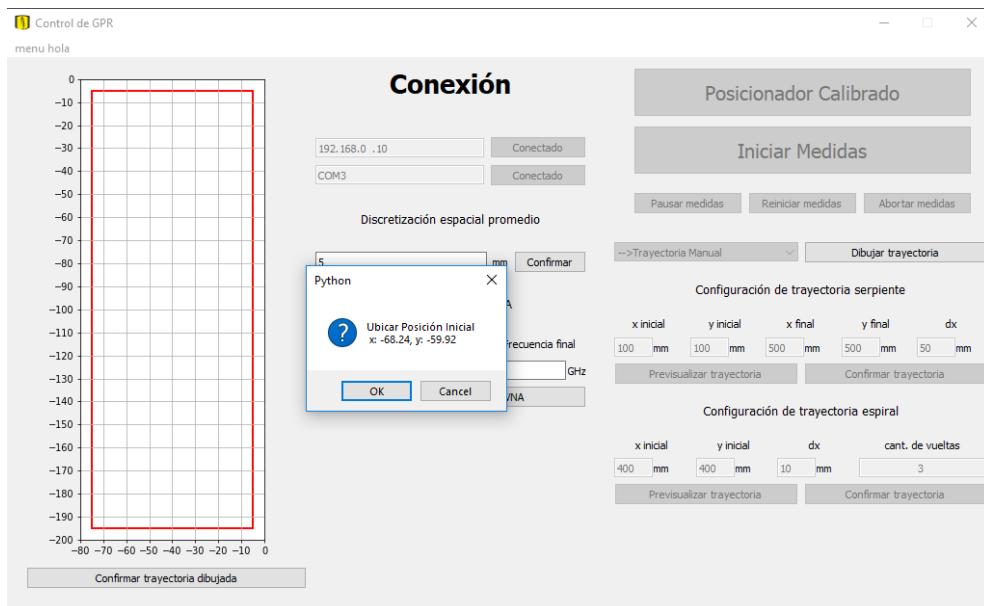


Figura 16: Aviso para confirmar punto de inicio

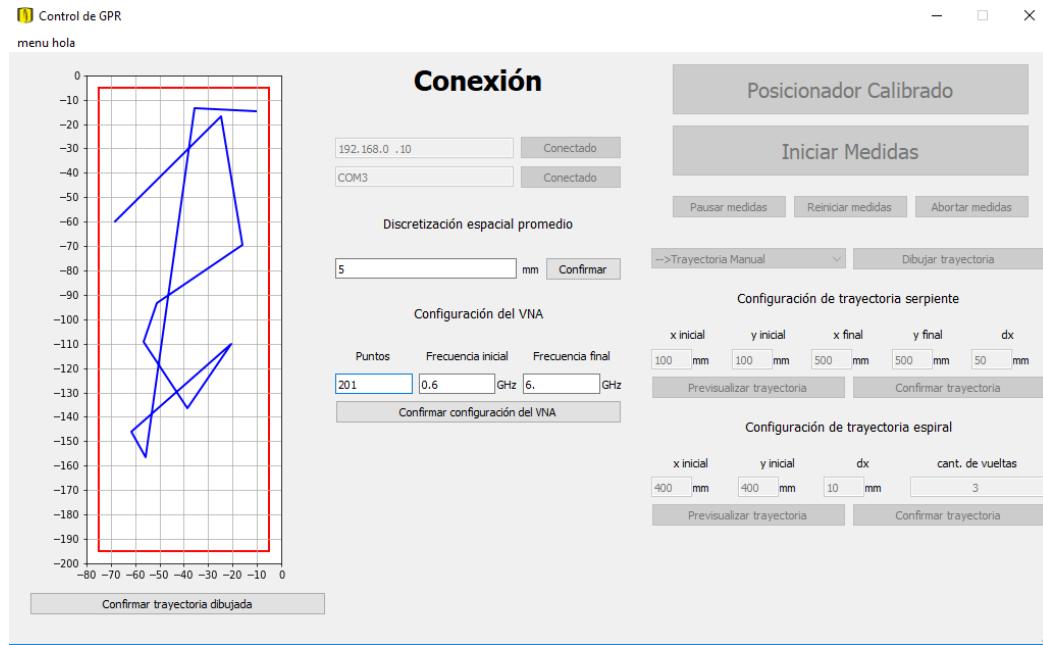


Figura 17: Dibujo de la trayectoria manual

Cuando el usuario termine de dibujar la trayectoria manual, en la parte inferior del dibujo se debe oprimir “Confirmar trayectoria dibujada” y con esto se habilita el botón “Iniciar medidas”

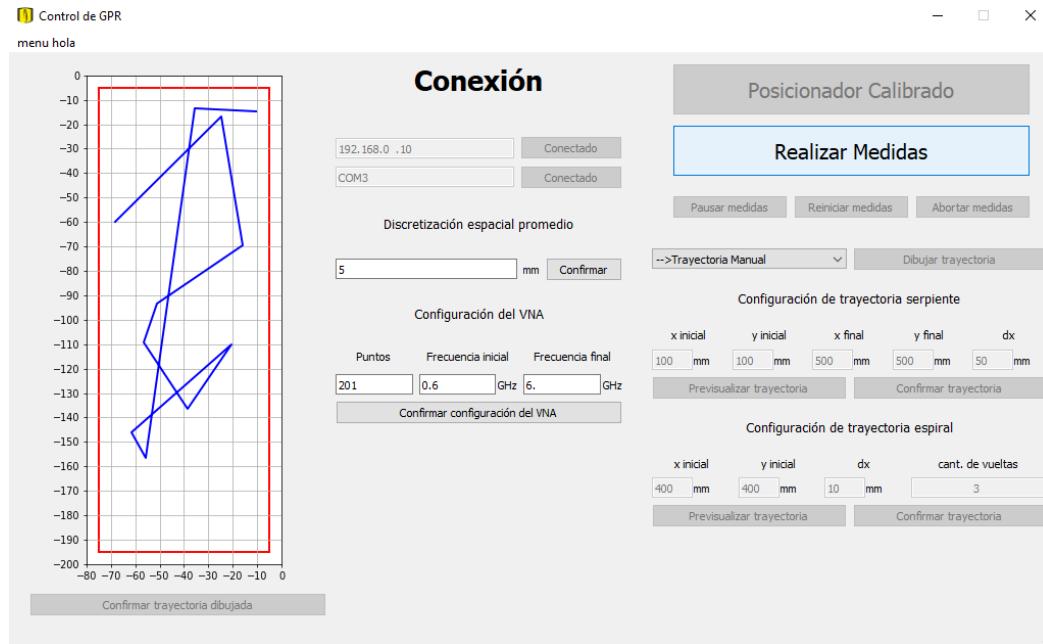


Figura 18: Confirmar Dibujo

Así como ocurre con el trayecto tipo serpiente, el usuario puede conocer el recorrido que ha realizado el sistema GPR porque el dibujo cambia a color rojo. En la figura 19 se observa el proceso del sistema GPR para realizar las mediciones en el trayecto definido por el usuario.

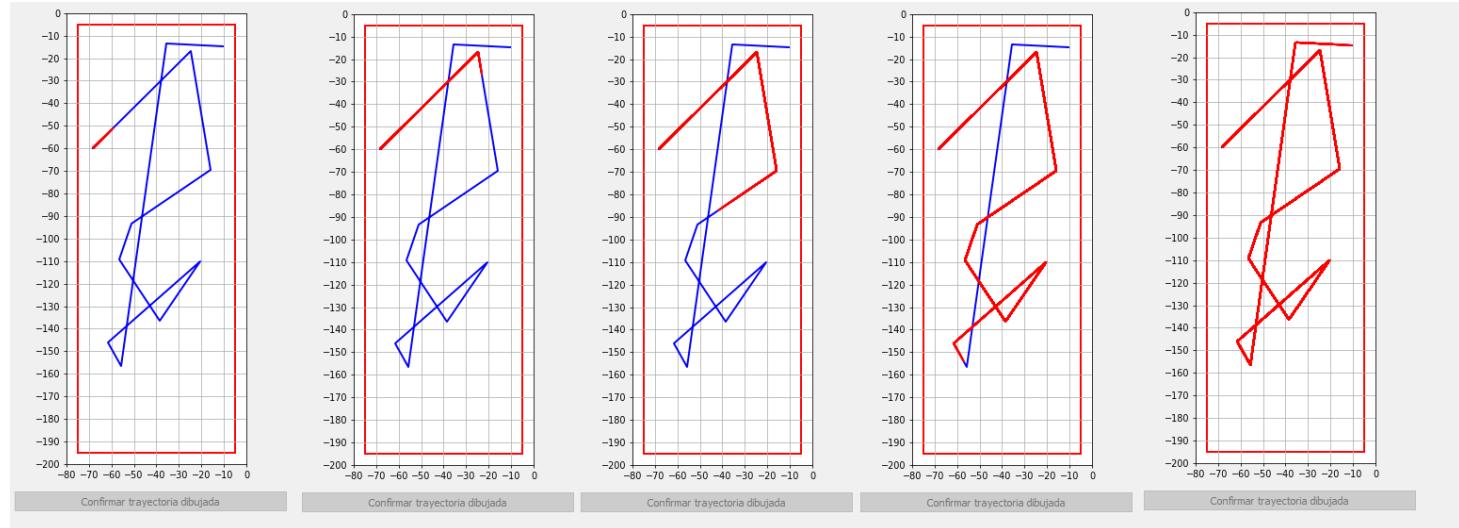


Figura 19: Trayecto realizado por el controlador de posición

5. Lectura de los datos

Los datos de salida, almacenados en un archivo .mat pueden ser leídos en Matlab fácilmente. Desde el software controlador del GPR se almacenan los datos de manera que al importarlos en Matlab, estén por defecto en forma de variables. A continuación se presenta un código donde se importan los datos, se genera la gráfica del parámetro S21 para cada medición, se genera la gráfica de la trayectoria recorrida, y se generan histogramas con las estadísticas de tiempos por medición y distancia entre puntos:

```
% Procesamiento_Ctrl_y_VNA.m
%
% Este programa procesa los datos obtenidos del control de posicion y el
% VNA en la rutina de Python.
%
% Humanitarian Microwave Detection of Improvised Devices in Colombia
% Universidad de los Andes
%
% Desarrolladores:
% Daniel Gonzalez Ramirez -dj.gonzalez1203@uniandes.edu.co
% Luis Eduardo Quibano Alarcon -le.quibano@uniandes.edu.co
% Julio 7 de 2018
clear, clc
close all
%% Carga del archivo con los datos
% Se realiza la carga de los datos a procesar. Ya que se espera
% eventualmente contar con muchos archivos de datos, se realiza una
% confirmacion de los datos seleccionados dando una descripcion general del
% archivo cargado.
answer = 'No';
salida = 'No';
while strcmp(answer,'No') && strcmp(salida,'No')
    [file, path] = uigetfile({'*.mat';'*.*'},'File Selector');
    load(strcat(path,file))

    % Los datos que se cargan del archivo son: Parametro S21 complejo
    % medido con el VNA (DatosVNAComplex), tiempo transcurrido entre las
    % mediciones (elapsed), descripcion general de la medicion hecha
    % (description), tiempo muerto para refrescar la pantalla del VNA
    % (offset), y posicion medida con el control de posicion (position)

    quest = ['Mostrar Datos de: ',description];
    answer = questdlg(quest,'Confirmar datos seleccionados','Si','No','Si');
    if strcmp(answer,'No')
        quest = 'Desea cancelar la ejecucion?';
        salida = questdlg(quest,'Confirmar salida','Si','No','Si');
        if strcmp(salida,'Si')
            return;
        end
    end
end
clear quest answer salida
% Almacenamiento de los datos en arreglos tipo celda
% Los datos cargados vienen como matrices de caracteres. Para procesarlos
% mas facilmente, estos pueden ser almacenados como arreglos tipo celda.

% Conversion de datos a arreglos tipo celda de cadenas de caracteres:
DatosVNAComplex = cellstr(DatosVNAComplex);
position = cellstr(position);

Q_meas = length(DatosVNAComplex); % Cantidad de mediciones

% Creacion de los arreglos para almacenar los datos procesados:
DatosVNAProcesados = cell(1,Q_meas);
Coor_x = zeros(1,Q_meas);
Coor_y = zeros(1,Q_meas);

% % Almacenamiento de los datos como numeros
% Con un ciclo se recorren los arreglos sin procesar y se almacenan los
% datos procesados en los arreglos creados para ese fin.
for i = 1:Q_meas

    display(i)

    DatoVNA = DatosVNAComplex{i}; % Aux. almacena el dato de la posicion i

    % El VNA reporta sus datos con el siguiente formato: #AX[]; # es el
    % caracter '#'. AX es la cabecera de los datos, A es un digito que
    % indica cuantos digitos ocupa la cabecera (A+X) y X indica cuantos
    % bytes hay despues de la cabecera AX. Posterior a #AX, vienen los
    % datos de parte real e imaginaria del parametro S21; el formato en que
    % se entregan estos datos es:
    % p_real_1,p_imag_1,p_real_2,p_imag_2,...,p_real_n,p_imag_n.
    % Cada fila del arreglo de celdas corresponde a una medicion del
    % parametro S21 para los puntos de frecuencia configurados en el
    % programa de adquisicion de datos.

    len_cabecera = str2double(DatoVNA(2)); % Dato A (cant. digitos en cab.)
    valores_S21 = DatoVNA(len_cabecera+3:end); % Valores S21 (string)
    valores_S21 = strsplit(valores_S21,''); % Separacion en cadenas indep.
    valores_S21 = str2double(valores_S21); % Conversion de S21 (double)
    valores_S21 = valores_S21(1:end-1); % Borra valor NaN generado al final

    Q_pun_VNA = length(valores_S21)/2; % Cantidad de puntos en frecuencia
```

```

compleja = zeros(1,Q_pun_VNA); % Vector para almacenar S21 complejo
% Ciclo para almacenar los valores en compleja
k = 1;
for j=1:2:length(valores_S21)
    compleja(k) = valores_S21(j) + li*valores_S21(j+1);
    k = k + 1;
end
% Guarda el valor complejo obtenido para la medicion
DatosVNAProcesados{i} = compleja;

DatoPOS = position{i}; % Aux. almacena el dato de la posicion
% Encuentra la posicion en la cadena de caracteres donde se encuentra
% el dato correspondiente a la coordenada x:
Pos_x = strfind(DatoPOS,'WPos:') + 5;

DatoPOS = DatoPOS(Pos_x:end); % Borra los datos de la cabecera
% Encuentra la posicion en la cadena de caracteres donde se encuentra
% el dato correspondiente a la coordenada y:
Pos_y = strfind(DatoPOS,',') + 1;

% La posicion de la coordenada y en la cadena de caracteres se utiliza
% para conocer el fin de la coordenada x:
Dato_x = DatoPOS(1:Pos_y-2); % Coordenada x de la medicion (string)

DatoPOS = DatoPOS(Pos_y:end); % Borra los datos ya almacenados

% Encuentra la posicion en la cadena de caracteres donde se encuentra
% el fin de la coordenada y:
Pos_fin = strfind(DatoPOS,',')-1;
Dato_y = DatoPOS(1:Pos_fin); % Coordenada y de la medicion (string)

Coor_x(i) = str2double(Dato_x); % Coordenada x de la medicion (double)
Coor_y(i) = str2double(Dato_y); % Coordenada y de la medicion (double)
end
clear DatoVNA len_cabecera_valores_S21 compleja DatoPOS Pos_x Pos_y ...
Dato_x Dato_y Pos_fin i j k
%% Visualizacion de los datos obtenidos
% Visualizacion de los datos del VNA
DatosVNAProcesados=DatosVNAProcesados'; % Transpone el arreglo del S21
freq = linspace(0,6,6,201);
figure('Name','Datos S21')
for i = 1:Q_meas
    subplot(2,1,1);plot(freq,real(DatosVNAProcesados{i}),'b','LineWidth',2)
    xlim([freq(1) freq(end)]); ylabel('Parte real S21');
    xlabel('Frecuencia [GHz]');
    title(strcat('Medicion #:',num2str(i)));
    subplot(2,1,2);plot(freq,imag(DatosVNAProcesados{i}),'r','LineWidth',2)
    xlim([freq(1) freq(end)]); ylabel('Parte imaginaria S21');
    xlabel('Frecuencia [GHz]');
    pause(0.01)
end
% Visualizacion de los datos de posicion
figure('Name','Posiciones de la trayectoria')
plot(Coor_x,Coor_y,'g','Marker','o')
xlabel('Eje x'); ylabel('Eje y')
title('Trayectoria del GPR')
clear i
%% Histogramas
%Histograma del tiempo transcurrido
figure('Name','Histograma tiempo transcurrido')
histogram(elapsed,20);
ylabel('Cantidad de datos'); xlabel('Tiempo transcurrido [s]')
title('Tiempo de una medicion')
dim = [.5 .5 .3 .3];
str = {char(strcat('Promedio: ','{',num2str(mean(elapsed)))),...
        char(strcat('Desviacion est.: ','{',num2str(std(elapsed))))};
annotation('textbox',dim,'String',str,'FitBoxToText','on');

% Histograma del cambio de posicion, entre mediciones, para toda la
%trayectoria
Cambio_pos = zeros(1,Q_meas);
for t = 1:Q_meas-1
    Cambio_pos(t) = sqrt(Coor_y(t+1)^2+Coor_x(t+1)^2)-sqrt(Coor_y(t)^2+Coor_x(t)^2);
end
Cambio_pos(1) = 0;
Cambio_pos = abs(Cambio_pos);
figure('Name','Histograma cambio de la posicion')
histogram(Cambio_pos,20);
ylabel('Cantidad de datos'); xlabel('Distancia de cambio [mm]')
title('Cambio de la posicion entre mediciones, para toda la trayectoria')
dim = [.2 .5 .3 .3];
str = {char(strcat('Promedio: ','{',num2str(mean(Cambio_pos)))),...
        char(strcat('Desviacion est.: ','{',num2str(std(Cambio_pos))))};
annotation('textbox',dim,'String',str,'FitBoxToText','on');

clear t r dim str

```

Los resultados que generan este código son gráficas del parámetros S21 para cada medición, una gráfica de la trayectoria recorrida, un histograma con los tiempos de medición y un histograma con el cambio de posición para cada medición. Un ejemplo se muestra en las figuras 20 a 22.

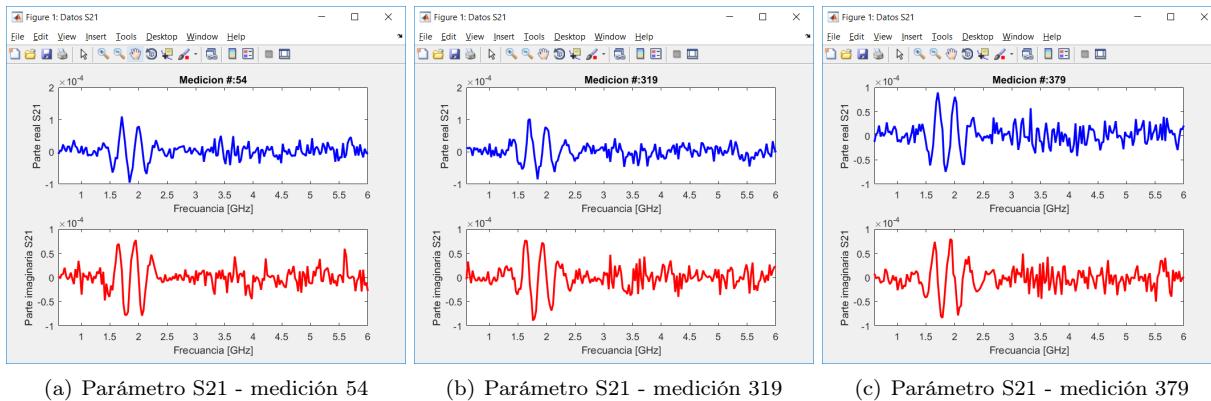
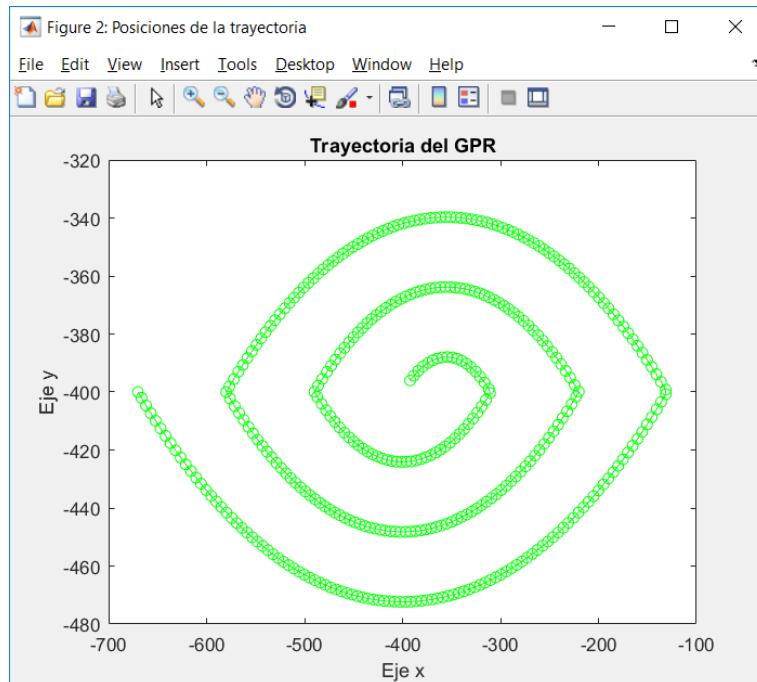
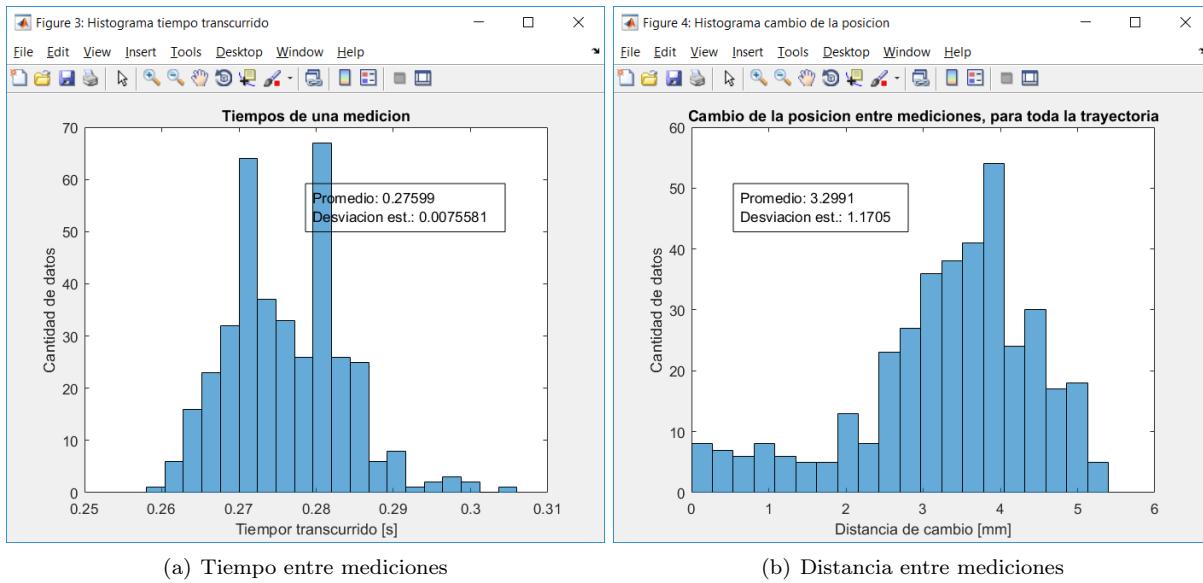
Figura 20: Gráficas del parámetro S_{21} medido

Figura 21: Gráfica de la trayectoria generada desde Matlab



(a) Tiempo entre mediciones

(b) Distancia entre mediciones

Figura 22: Estadísticas de las mediciones

Con estos resultados se puede evaluar fácilmente que tan constante es el cambio de posición y el tiempo entre las mediciones. A partir de este archivo, es posible obtener diferentes resultados y procesamiento sobre los datos recuperados.

6. Bibliografía

Referencias

- [1] Sonny Jeon aka @chamint. Grbl. <https://github.com/gnea/grbl>. Accessed: 2018-07-25.
- [2] Ansrtsu. Vna master ms2026c. <https://www.anritsu.com/en-US/test-measurement/products/ms2026c>. Accessed: 2018-07-25.
- [3] OpenBuilds. Nema-23 stepper motor. <https://openbuildspartstore.com/nema-23-stepper-motor/>. Accessed: 2018-07-25.
- [4] RF Space. Ultra-wideband pcb tapered slot antenna tsa600. http://rfospace.com/RFSPACE/Antennas_files/TSA600.pdf. Accessed: 2018-07-25.
- [5] Spark-Concepts. xpro v3.2. <https://github.com/Spark-Concepts/xPRO>. Accessed: 2018-07-25.