UNIVERSIDAD NACIONAL DE CHIMBORAZO
FACULTAD INGENIERÍA
CARRERA ELECTRÓNICA Y TELECOMUNICACIONES

Diseño e implementación de un radar de banda estrecha, utilizando SDR para la detección de objetos y medición de distancia.

Trabajo de Titulación para optar al título de Ingeniero en Electrónica y Telecomunicaciones

Autor:
Pino Enríquez, Jeffersson Pascal

Tutor:
PhD. Leonardo Fabián Rentería Bustamante

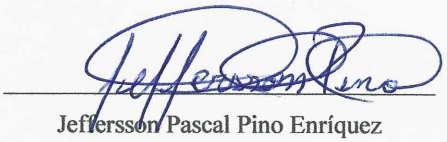
Riobamba, Ecuador. ***2023***

DERECHOS DE AUTORÍA

Yo, Jeffersson Pascal Pino Enríquez, con cédula de ciudadanía 060335083-6, autor del trabajo de investigación titulado: Diseño e implementación de un radar de banda estrecha, utilizando SDR para la detección de objetos y medición de distancia, certifico que la producción, ideas, opiniones, criterios, contenidos y conclusiones expuestas son de mí exclusiva responsabilidad.

Asimismo, cedo a la Universidad Nacional de Chimborazo, en forma no exclusiva, los derechos para su uso, comunicación pública, distribución, divulgación y/o reproducción total o parcial, por medio físico o digital; en esta cesión se entiende que el cesionario no podrá obtener beneficios económicos. La posible reclamación de terceros respecto de los derechos de autor (a) de la obra referida, será de mi entera responsabilidad; librando a la Universidad Nacional de Chimborazo de posibles obligaciones.

En Riobamba, 17 de abril de 2023.



Jeffersson Pascal Pino Enríquez

C.I: 060335083-6

DICTAMEN FAVORABLE DEL TUTOR Y MIEMBROS DE TRIBUNAL;

Quienes suscribimos, catedráticos designados Tutor y Miembros del Tribunal de Grado para la evaluación del trabajo de investigación Diseño e implementación de un radar de banda estrecha, utilizando SDR para la detección de objetos y medición de distancia, presentado por Jeffersson Pascal Pino Enríquez, con cédula de identidad número 060335083-6 certificamos que recomendamos la APROBACIÓN de este con fines de titulación. Previamente se ha asesorado durante el desarrollo, revisado y evaluado el trabajo de investigación escrito y escuchada la sustentación por parte de su autor; no teniendo más nada que observar.

De conformidad a la normativa aplicable firmamos, en Riobamba 17 de abril de 2023

José Luis Jinez Tapia, MsC.
PRESIDENTE DEL TRIBUNAL DE GRADO

Marlon Danilo Basantes Valverde, PhD
MIEMBRO DEL TRIBUNAL DE GRADO

Edgar Giovanny Cuzco Silva, MsC.
MIEMBRO DEL TRIBUNAL DE GRADO

Leonardo Fabián Rentería Bustamante, PhD.
TUTOR

CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL

Quienes suscribimos, catedráticos designados Miembros del Tribunal de Grado para la evaluación del trabajo de investigación Diseño e implementación de un radar de banda estrecha, utilizando SDR para la detección de objetos y medición de distancia, presentado por Jeffersson Pascal Pino Enríquez, con cédula de identidad número 0603350836, bajo la tutoría del PhD. Leonardo Fabián Rentería Bustamante; certificamos que recomendamos la APROBACIÓN de este con fines de titulación. Previamente se ha evaluado el trabajo de investigación y escuchada la sustentación por parte de su autor; no teniendo más nada que observar.

De conformidad a la normativa aplicable firmamos, en Riobamba 17 de abril de 2023.

Presidente del Tribunal de Grado
MsC. José Luis Jinez Tapia

Miembro del Tribunal de Grado
PhD. Marlon Danilo Basantes Valverde

Miembro del Tribunal de Grado
MsC. Edgar Giovanny Cuzco Silva

C E R T I F I C A C I Ó N

Que, **PINO ENRIQUEZ JEFFERSSON PASCAL** con CC: **0603350836**, estudiante de la Carrera **ELECTRONICA Y TELECOMUNICACIONES**, **NO VIGENTE,** Facultad de **INGENIERIA**; ha trabajado bajo mi tutoría el trabajo de investigación titulado "**DISEÑO E IMPLEMENTACIÓN DE UN RADAR DE BANDA ESTRECHA, UTILIZANDO SDR PARA LA DETECCIÓN DE OBJETOS Y MEDICIÓN DE DISTANCIA**", cumple con el 2.1%, de acuerdo al reporte del sistema Anti plagio **URKUND**, porcentaje aceptado de acuerdo a la reglamentación institucional, por consiguiente autorizo continuar con el proceso.

Riobamba, 10 de abril de 2023

Firmado electrónicamente por:
LEONARDO FABIAN
RENTERIA BUSTAMANTE

PhD. Leonardo Fabián Rentería Bustamante
TUTOR TRABAJO DE INVESTIGACIÓN

DEDICATORIA

El presente trabajo está dedicado a mi madre Silvia Jacqueline, quien me ha brindado su apoyo incondicional e incentivado en gran medida a cumplir mis objetivos, a mi hijo Jaylogan Aaron y su madre Ana Gabriela quienes han sido una motivación para culminar mis estudios, a mis hermanos Erika Joith, y Samantha, a mi familia, a mis amigos y a todos quienes me ayudaron y orientaron en esta etapa.

AGRADECIMIENTO

Extiendo mi gratitud eterna, a la mejor persona del mundo, mi madre Silvia Jacqueline quien es mi mayor fuente de inspiración y orgullo, mi motor para poder avanzar, quien ah echo más esfuerzo del necesario por mis hermanos y por mí. Espero algún día poder recompensar todo tu amor.

Agradezco a Ana Gabriela que siempre estuvo ahí conmigo apoyándome, brindándome cariño y corrigiéndome cuando lo he necesitado, quien me dio el mejor regalo del mundo, mi hijo, Jaylogan Aaron, mi nueva luz que con su cariño y amor me motiva día a día a continuar, seguir adelante y cumplir nuevas metas. Espero cumplir con sus expectativas y dar todo de mí.

A mis hermanos Erika, Joith y Samantha que han sido pacientes, comprensivos y tolerantes. Los mejores hermanos que se podría pedir.

A mis abuelos Alberto y Marta, a mis tíos y al resto de mi familia.

También extiendo mi más sincero agradecimiento a mis amigos, amigas y a todas las personas que han aportado en mi vida, tanto en mi formación académica como en mi crecimiento personal.

Muchas Gracias.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente proyecto de investigación tiene la finalidad de diseñar e implementar un radar de banda estrecha, utilizando Radio Definido por Software (SDR) para la detección de objetos y medición de distancia, el diseño se desarrolló en la plataforma de software libre GNURadio y se implementó en dos módulos de aprendizaje activo de radio definido por software "Adalm Pluto". El sistema está basado en el algoritmo de radar de Onda Continua con Múltiples Frecuencias (MFCW) y consta de dos transmisores de onda continua y dos receptores calibrados a diferente frecuencia con una separación entre ellas Δf la cual determinara el rango de medición Rmax, es posible determinar la distancia de un objeto detectado gracias al cambio de fase de la señal, cuando la señal es reflejada en un objeto a una distancia R regresa al radar con un desfase en cada portadora, las señales de transmisión son eliminadas después de ser recibidas obteniendo únicamente el módulo del desplazamiento de fase ($\Delta\Phi$) por lo que tenemos una distancia máxima sin ambigüedad cuando $\Delta\Phi=2\pi$, en el primer capítulo se hace una recopilación informativa de antecedentes, motivación y problemática y justificación del proyecto, a continuación, el capítulo, dos se recoge la fundamentación teórica y física de la que parte el radar, en el capítulo tres se establece el tipo de investigación, variables, herramientas necesarias y se lleva a cabo el diseño, implementación, y puesta en marcha del radar, posteriormente, en el capítulo cuatro se expone los resultados y el análisis de los mismos partiendo de herramientas estadísticas y de estimación, como son análisis de error relativo, comparación de medias, intervalos de confianza, y un control de calidad de las mediciones realizadas por el prototipo, finalmente, en el capítulo cinco se da las conclusiones y recomendaciones de la investigación.

Palabras claves: Portadora, SDR (Radio Definido por software), Banda estrecha, Ancho de banda, Cambio de fase, GNURadio, Error Relativo.

ABSTRACT

The aim of this research project is to design and implement a narrowband radar using Software Defined Radio (SDR) for object detection and distance measurement. The design was developed on the open-source software platform, GNURadio, and implemented on two active learning modules of Software Defined Radio, called "Adalm Pluto". The system is based on the Multiple Frequency Continuous Wave (MFCW) radar algorithm and consists of two continuous wave transmitters and two calibrated receivers operating at different frequencies with a separation of Δf, which determines the maximum range of measurement, $Rmax$. It is possible to determine the distance of a detected object thanks to the phase shift of the signal when it is reflected off an object at a distance R and returns to the radar with a phase shift on each carrier. The transmission signals are eliminated after being received, obtaining only the phase shift module (ΔΦ). Therefore, we have a maximum distance without ambiguity when $\Delta\Phi = 2\pi$, the chapter one provides background information, motivation, problem statement, and project justification, the chapter two discusses the theoretical and physical foundation of the radar system, the chapter three outlines the research methodology, variables, necessary tools, and carries out the design, implementation, and testing of the radar, the chapter four presents the results and their analysis using statistical and estimation tools such as relative error analysis, mean comparison, confidence intervals, and quality control of the measurements taken by the prototype. Finally, the chapter five provides the conclusions and recommendations of the research.

Keywords: Carrier, Software Defined Radio (SDR), Narrowband, Bandwidth, Phase Shift, GNURadio, Relative Error.

ALFONSO FABIAN MARTINEZ CHAVEZ

Firmado digitalmente por ALFONSO FABIAN MARTINEZ CHAVEZ
Fecha: 2023.04.13 09:49:49 -05'00'

Reviewed by:
Mgs. Alfonso Fabian Martínez Chávez.
ENGLISH PROFESSOR
c.c. 0602778268

CAPÍTULO I. INTRODUCCION.

La tecnología radar experimenta un resurgimiento gracias a la demanda de equipos compactos, rentables y de alta potencia, tanto en aplicaciones militares, comerciales y de control de tráfico. Lo que ha llevado a la una reevaluación de cómo desarrollar soluciones de radar robustas y económicas. Tecnologías innovadoras, tales como vehículos aéreos no tripulados, automóviles autónomos y una gran variedad de aplicaciones comerciales, se basan en un radar de estado sólido, en los que se ha rediseñado la programación y fabricación para satisfacer las necesidades del mercado. Para responder a formas y ondas de radar inteligentes y complejas están surgiendo la radio y los radares cognitivos, habilitados por sistemas de radio definido por software (SDR).[1]

Los radares tradicionales se desarrollaban para la detección de objetivos a grandes distancias en el orden de los kilómetros, con el uso de una gran potencia. Sin embargo, se han vuelto bastante útiles en un sin número de aplicaciones en la actualidad, en las que se requiere una resolución basándose en metros e incluso en centímetros.[2]

En el presente trabajo se llevará a cabo el diseño de un radar de banda estrecha de bajo costo utilizando Radio Definido por Software y el algoritmo de radar de onda continua en frecuencia múltiple (MFCW).

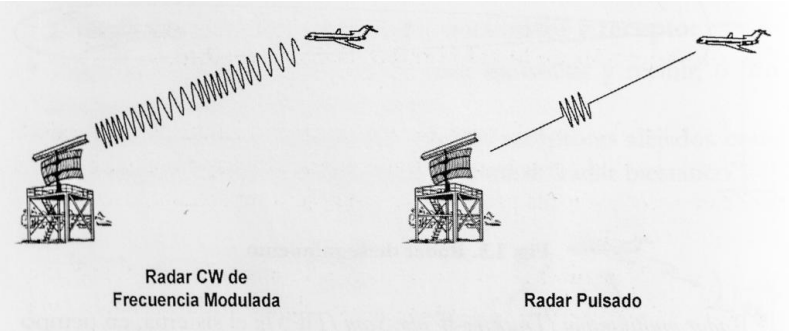


Figura 1: Radares tradicionales, donde la antena envía una señal FMCW o una señal pulsante para poder determinar la distancia.[3]

En los radares tradicionales se transmite una señal electromagnética, y las ondas de radio son reflejadas hacia el receptor. El radar emplea el tiempo de viaje de la señal para calcular la distancia en función de la velocidad de la onda. En el caso del radar de onda continua

con múltiples frecuencias (MFCW), se envían varias ondas continuas, una por cada portadora, con una pequeña variación de frecuencia, lo que permite ahorrar el ancho de banda; ya que no se emplea una modulación para obtener la señal de referencia como en el caso del radar de onda continua en frecuencia modulada FMCW.

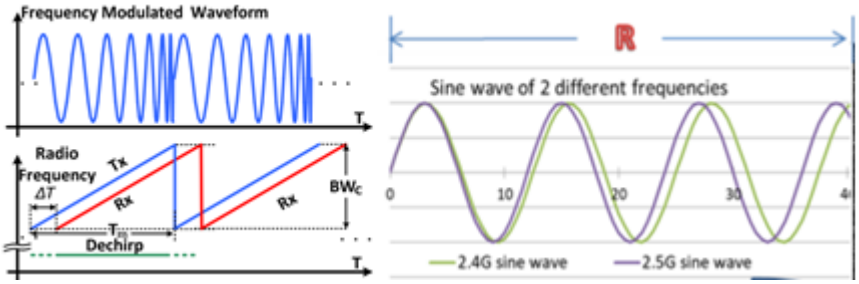


Figura 2:Forma de onda del Radar de onda continua en frecuencia modulada (FMCW) y del Radar de onda continua en múltiples frecuencias (MFCW).[2]

La investigación tiene como objetivo el diseño y construcción de un dispositivo radar que use un ancho de banda reducido, desarrollado sobre un Radio Definido por Software (SDR). Tendrá como finalidad la detección de objetos y la medición de distancia, teniendo en cuenta el uso de frecuencias de banda libre y un ancho de banda pequeño a comparación de otras tecnologías.

La contribución de esta investigación experimental es el estudio de resultados medidos por el radar y tratar de reducir al máximo el error en la medición, mediante la modificación directa de la variable independiente, tratando de tener una tolerancia mínima en este error.

El algoritmo de radar de onda continua con múltiples frecuencias es muy poco popular y casi nulo en el desarrollo de radares comerciales, a pesar de tener excelentes prestaciones, este dispositivo será de amplia utilidad y su alcance podrá tener múltiples aplicaciones enfocadas en su principal función tales como:
Educación sobre comunicaciones inalámbricas (Radares - Universidad).
Seguridad (Bancos, cooperativas, casas, etc.).
Sensor anticolisión (Vehículos Autónomos).

Su desarrollo es relativamente veloz, las interfaces de trasmisión y recepción se integran sobre el SDR a partir de software utilizando GNU Radio, entorno que nos permitirá el ajuste y calibración de las señales, y la manipulación directa de la variable independiente.

El proyecto está planteado a desarrollar en el lapso julio – septiembre 2022 en el laboratorio de electrónica de la facultad de ingeniería UNACH.

1.1 Planteamiento del Problema

En la actualidad, existen diversos dispositivos y prototipos de radar que nos ofrecen diferentes beneficios tales como cálculo de velocidad o distancia, monitoreo climático, detección de objetivos(targets), etc. Dichos dispositivos emplean diferentes tecnologías y requerimientos para un desempeño exitoso, y se encuentran aún en continuo desarrollo gracias a los niveles de integración logrados por la electrónica de alta frecuencia y los avances en las técnicas de procesamiento de señales.

El problema de los equipos convencionales como el radar de pulsos que genera como su nombre lo indica un pulso de señal de alta frecuencia y potencia, es que tiene que esperar un periodo prolongado de descanso en el que se recibe y procesa la señal antes de enviarse un nuevo pulso por lo que la medición de distancia no es continua. En el caso del radar de onda continua (CW) si bien ahora la señal se envía y procesa simultáneamente, ya que es continua y constante tanto en amplitud y frecuencia, únicamente nos entrega el desplazamiento de frecuencia para determinar la velocidad, pero no da información relevante para el cálculo de distancia. Por otro lado, en el caso del radar de onda continua en frecuencia modulada (FMCW), al modular la onda en frecuencia nos da la posibilidad de medir el tiempo de vuelo y por consecuencia la medición de distancias, pero tiene el inconveniente de necesitar grandes anchos de banda y alta potencia para generar la señal.[3]

La Distancia medida por el radar está directamente asociada a diferencia de fase de la señal reflejada, donde se medirá el tiempo de vuelo de la señal propagada en el vacío, y su resolución será afectada directamente con la frecuencia y por consecuencia con el ancho de banda de la señal. El desarrollo de este dispositivo se delimita a un espacio en el que se puedan llevar a cabo mediciones continuamente como un laboratorio, habitación, etc.

El objetivo fundamental del dispositivo es detectar objetos y medir la distancia desde el radar hasta ellos, La valoración de su desempeño estará muy marcada, con análisis

estadísticos que probaran su eficacia, a diferencia de elementos convencionales como sensores de distancia y/o movimientos como el infrarrojo, el ultrasónico, etc. Tienen un espectro de Radiofrecuencia mucho mayor(penetrabilidad), mejor tiempo de vida, totalmente directivo, de carácter industrial, mayor distancia de medición, mejor precisión, etc. Sin embargo, en principio la presente investigación puede ser utilizada, desde una perspectiva académica, este radar puede ser enfocado como un instrumento que, permita estimular el aprendizaje de conceptos teóricos que, en muchas ocasiones, pueden ser bastantes complejos de comprender como: el efecto Doppler, cambio de fase, los radares y sus aplicaciones en el campo de las telecomunicaciones, etc. Por ello, su desarrollo se planea en tarjetas SDR (basadas en FPGA) y a través de un software sencillo que permita la modificación de diferentes parámetros en el radar, considerando una mejora en la adquisición de competencias generales y específicas en los estudiantes, y posteriormente, puede aplicarse para la detección temprana de intrusos en instituciones financieras gracias a sus características.

La construcción del dispositivo cuenta con antenas que se montaran en una base fija para poder regular su altura, para las pruebas se limitara esta altura a 1 metro desde el suelo, y como objetivo se establecerá una pantalla metálica (0.65 m * 0.65 m), la cual se ubicara en una base móvil a 0.8 metros desde el suelo al panel dándonos una altura del objeto entre 0.8 y 1.40 metros por lo que fácilmente podrá captar objetos medianos, personas, vehículos, etc., con lo cual quedara en evidencia la intrusión de estos.

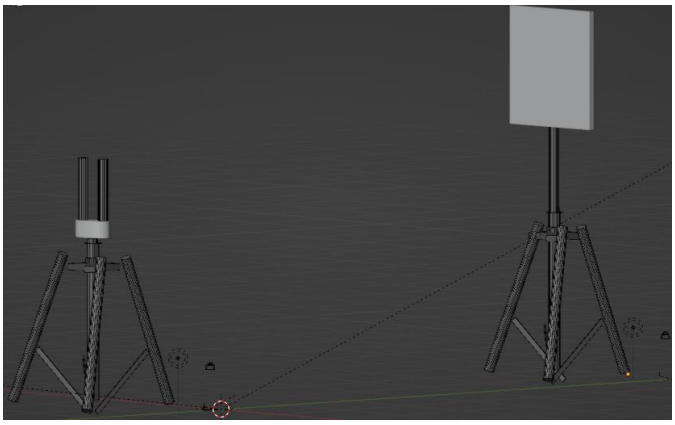


Figura 3: Simulación del escenario de Pruebas, las antenas medirán la distancia hasta el panel metálico.

1.2 Justificación

La utilización de un radio definido por software (SDR) para el desarrollo de un radar de onda continua en múltiples frecuencias (MFCW) ofrece la integración de todos los componentes necesarios como generación de señales, transmisor, receptor y procesamiento, en una sola plataforma, que puede ser utilizada, modificada o visualizada por el usuario mediante un sistema computacional o embebido. Lo que permite la construcción de prototipos potentes y compactos.

La implementación de un radar de onda continua en múltiples frecuencias (MFCW) solventa de manera adecuada el cálculo de distancias con el uso de un ancho de banda relativamente estrecho, ya que no se necesita una modulación en frecuencia la cual aumenta significativamente el ancho de banda para obtener una buena resolución, además de ser capaz de detectar objetivos que se encuentran detrás de paredes o paneles de distintos materiales que no sean totalmente reflectivos, tales como ladrillos, bloques, yeso, etc. por lo que es útil incluso si las antenas están siendo bloqueadas a diferencia de otro método para medir distancia como sensores ultrasónicos, infrarrojos, láser, etc.

Un sistema de radar de onda continua con múltiples frecuencias (MFCW), permite la detección de objetos estáticos y en movimiento, y la medición de distancia con una alta precisión en el orden de los centímetros, mediante el empleo de 2 o más canales de transmisión que generan y envían señales de diferentes frecuencias con ancho de banda estrecho en el orden de los KHz. La distancia se puede determinar gracias a la mezcla del desplazamiento de fase de las señales con la portadora, por lo que el post procesado se realizara únicamente multiplicando esta mezcla con una constante fácil de calcular y que permite la detección de objetos y una precisa determinación de distancia.[4]

1.3 Objetivos

1.3.1 General

- Diseñar e implementar un radar de banda estrecha, utilizando SDR para la detección de objetos y medición de distancia.

1.3.2 Específicos

- Estructurar los parámetros de Transmisión y diseño de las antenas.
- Establecer las interfaces del Transmisor (Tx) y Receptor (Rx) usando una plataforma de código abierto (GNURadio).
- Construir el Radar empleando el algoritmo de radar de onda continua con frecuencia múltiple (MFCW).
- Determinar el error existente entre la distancia medida del objeto detectado por el radar y la distancia real medida por un instrumento de longitud estándar

CAPÍTULO II. MARCO TEÓRICO.

2.1 Fundamentación teórica

2.1.1 El Radar

Se puede definir al radar como el proceso por el cual se puede detectarla presencia de un objeto en el espacio, determinando su posición, altitud y distancia respecto a un punto conocido, en el que generalmente es la instalación del sistema; haciendo el uso de la propiedad de reflexión que poseen las ondas electromagnéticas. [3]

El término Radar proviene de RAdio Detection And Ranging, donde para medir el alcance o distancia a la que se encuentra un eco o blanco, sea fijo o móvil, es necesario medir el tiempo que transcurre desde que el pulso es trasmitido, choca el blanco, se refleja y regresa al radar.[4]

2.1.2 Tipos de Radar

Los radares pueden clasificarse en dos categorías: Radar con generación de imágenes y Radar de medición. En los radares de generación de imágenes se trata de generar una imagen muy parecida a un mapa a partir de los datos que recibe. Sus aplicaciones más comunes son el radar meteorológico y el radar de vigilancia aérea.

En los radares de medición el resultado únicamente es un valor numérico puro, sin generación de imágenes y sus aplicaciones más características son altímetros y velocímetros.

2.1.2.1 Radar primario

Un radar primario es aquel que emite las señales de alta frecuencia al espacio y también reciben la señal después de ser reflejada en los objetivos, luego se procesan estos datos y se evalúan para generar imágenes o detectar objetivos.[3]

2.1.2.2 Radar secundario

Un radar secundario, por su parte, necesita que el blanco colabore para ser detectado, es decir, que dicho objetivo debe contar con un transpondedor, el cual recibirá una señal codificada a modo de interrogador, y este responderá con una señal generada en el mismo que también es codificada y enviada de regreso al radar secundario igual que el radar primario estos datos pueden ser usados para el cálculo de distancias, pero además esta señal contiene mucha más información útil por ejemplo la altitud, identificación, problemas técnicos, etc.[3]

2.1.2.3 Radares de pulso

Un radar de pulso es aquel que emite una señal de alta frecuencia de forma pulsante donde se envía un pulso de señal y luego una larga pausa en la que se podrá recibir los ecos antes de ser transmitida un nuevo pulso de señal. Tanto la distancia como la dirección y altura se pueden determinar si fuera necesario a partir de la posición de la antena y el tiempo de retorno de la señal.[3]

2.1.2.4 Radar de onda continua

Los radares de onda continua (CW) emiten una señal como su nombre lo indica de manera continua sin interrupción de ningún tipo, y las señales reflejadas son recibidas y procesadas de manera simultánea y continua.

En este tipo de radar, el receptor no tiene que estar estrictamente en el mismo lugar del trasmisor, por lo que permite que cualquier trasmisor potente, por ejemplo, uno de radiodifusión pueda usarse como transmisor de radar si un receptor remoto compara los tiempos de propagación directa y la señal que es reflejada, la ubicación de una aeronave puede determinarse evaluando las señales de 3 estaciones diferentes. El uso más común para este tipo de radar es la medición de velocidad gracias al desplazamiento de fase por efecto Doppler [3]

2.1.2.5 Radar FMCW

El Radar de onda continua en frecuencia modulada o FMCW (Frequency Modulated Continuous Wave radar). Es una evolución del radar de onda continua donde la señal trasmitida es constante en amplitud, pero modulada en frecuencia, esto permite además de medir la velocidad de un objeto, poder medir también la distancia de este en función a la estación gracias a la propiedad de vuelo de la señal al desplazamiento de frecuencia. La gran ventaja de este tipo de radar es que la evaluación de datos es de forma continua sin espera por lo que el resultado de la medición esta siempre a disposición.[3]

2.1.2.6 Radares con modulación intra-pulso

Este tipo de radar trasmiten pulsos con una potencia menor, pero con una duración mayor, el pulso es modulado en fase o frecuencia similar al FMCW utilizan la técnica de compresión de pulso para determinar la distancia al objetivo lo que permite poder recibir y procesar las señales incluso dentro de la duración de transmisión de pulso.[3]

2.1.2.7 Radares bi-estáticos

Se conoce como radar bi-estático al que consiste en múltiples sitios de transmisión y recepción separados por una distancia considerable.[3]

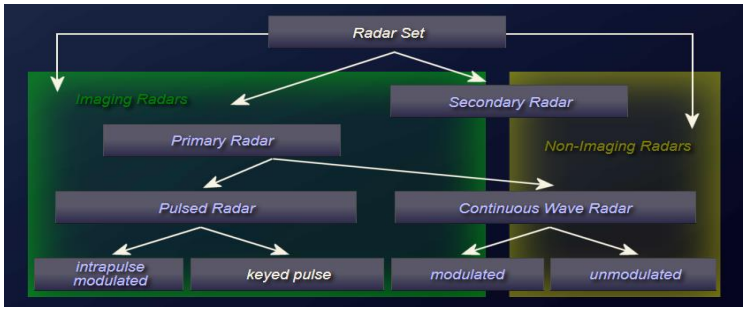


Figura 4:Tipos de Radar.[3]

2.1.3 Radar de onda continua con múltiples frecuencias MFCW

Estos radares transmiten simultáneamente varias ondas continuas en amplitud, generalmente sinusoidal, cada onda posee una portadora con una frecuencia única y diferente de las demás, por lo que el requisito de ancho de banda es muy bajo en el orden de los kilohercios y por consiguiente baja potencia.

La distancia se calcula a partir de la diferencia de fases entre las señales CW por lo que es un tipo extendido de radar de interferometría que se basa en medición de fase, la diferencia de cada fase en cada una de las frecuencias se determina de forma única.[2]

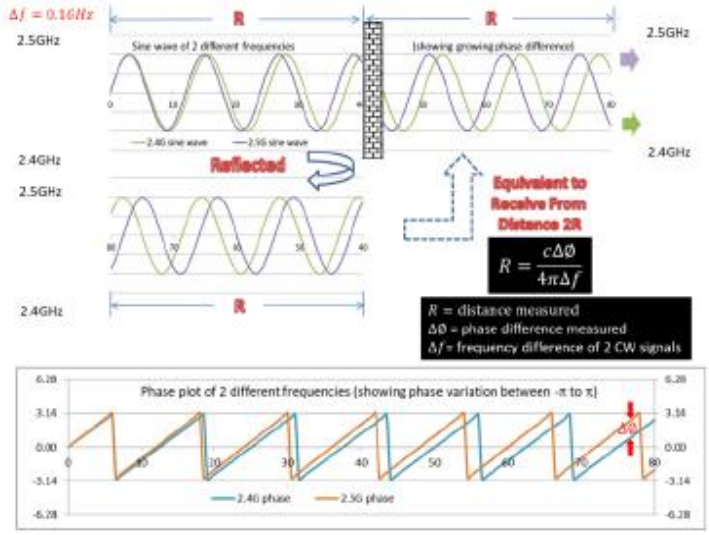


Figura 5: Radar MFCW operando con dos frecuencias diferentes de ondas CW con longitudes de onda ligeramente diferentes. [2]

2.1.4 Radio definido por software (SDR)

La radio definida por software es un sistema de radio comunicaciones donde los componentes que tradicionalmente eran implementados en hardware tales como mezcladores, filtros, moduladores, demoduladores, etc. Son ahora implementados mediante software gracias a las tecnologías FPGA utilizando dispositivos de computación o sistemas embebidos. Aunque el concepto de SDR no es nuevo, la continua evolución de sistemas digitales ha hecho posible la generación de sistemas prácticos potentes y rentables.

Los SDR en la actualidad se han convertido en un medio de desarrollo en múltiples áreas como civil, militar, académica, aérea, automovilística, etc. Generando tarjetas de evaluación suficientes para aprender y establecer prototipos propios en cualquier área que se quiera desempeñar, desde radio aficionado hasta prototipos que generan relevancia económica.

2.2 Fundamentación Matemática

Se parte del hecho que un radar de onda continua con múltiples frecuencias hace posible determinar de manera óptima los valores de distancia empleando 2 o más señales portadoras.

$$S_1(t) = A_1 \sin(2\pi f_1 t) \qquad (1)$$

$$S_2(t) = A_2 \sin(2\pi f_2 t) \qquad (2)$$

Cuando la señal es reflejada en un objeto a una distancia *R* regresa al radar con un desfase en cada señal enviada.

$$S_1(t) = A_{1r} \sin(2\pi f_1 t - \phi_1) \qquad (3)$$

$$S_2(t) = A_{2r} \sin(2\pi f_2 t - \phi_2) \qquad (4)$$

De acuerdo con el desplazamiento Doppler tenemos que el desplazamiento de fase es igual a:

$$\phi = \frac{4\pi R}{C} f \qquad (5)$$

Por lo que las señales reflejadas son:

$$S_1(t) = A_{1r} \sin\left(2\pi f_1 t - \frac{4\pi R}{C} f_1\right) \qquad (6)$$

$$S_2(t) = A_{2r} \sin\left(2\pi f_2 t - \frac{4\pi R}{C} f_2\right) \qquad (7)$$

La distancia medida está directamente ligada al desplazamiento de fase $\Delta\phi$, es decir, la diferencia entre las fases de cada señal. Las señales de transmisión son eliminadas después de ser recibidas al ser multiplicadas por su conjugada, obteniendo únicamente el módulo del desplazamiento de fase.

$$\Delta\phi = \left(\frac{4\pi R}{C} f_1\right) - \left(\frac{4\pi R}{C} f_2\right) \qquad (8)$$

$$\Delta\phi = \left(\frac{4\pi R}{C}\right)(f_1 - f_2) \qquad (9)$$

$$\mathrm{R} = \frac{C\Delta\phi}{4\pi(f_1 - f_2)} \qquad (10)$$

Por lo que tenemos que la Distancia es igual a:

$$R = \frac{C\Delta\phi}{4\pi\Delta_f} \qquad (11)$$

Donde:

$C =$ Rapidez de la luz

$\Delta\phi =$ Módulo del desplazamiento de fase

$\Delta f =$ Intervalo de la frecuencia (diferencia entre f_1 y f_2)

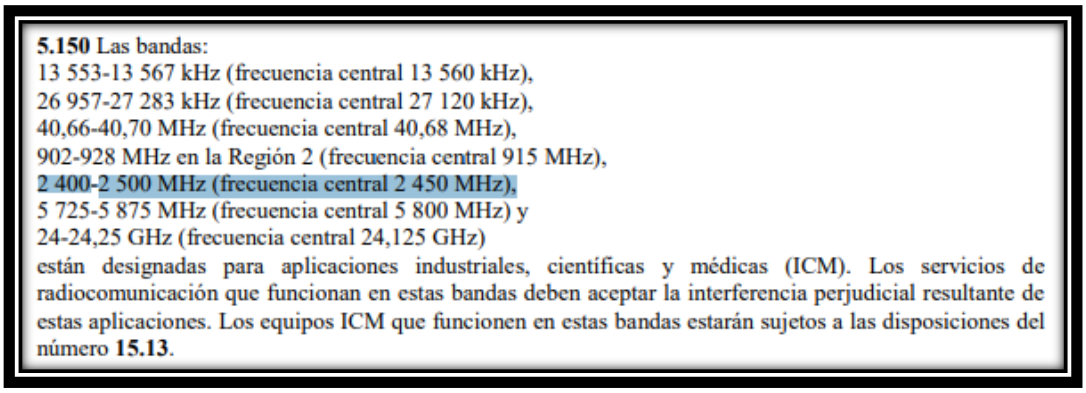
Para calcular la distancia máxima que puede ser medida se asume un cambio de fase máximo $\Delta\phi = 2\pi$ por lo que:

$$R_{max} = \frac{2\pi C}{4\pi\Delta_f} \qquad (12)$$

$$R_{max} = \frac{C}{2\Delta_f} \qquad (13)$$

2.3 Banda de Frecuencias

De acuerdo con la Agencia de Regulación y Control de las Telecomunicaciones (ARCOTEL), en su Plan nacional de frecuencias Ecuador 2021 con respecto al uso de las bandas de frecuencia denominadas no licenciadas o "aficionados" establece que dichas bandas están destinadas a la investigación, industria, ciencia o medicina. Y que los servicios de radio comunicación que las usen deben aceptar la interferencia perjudicial resultante.



5.150 Las bandas:
13 553-13 567 kHz (frecuencia central 13 560 kHz),
26 957-27 283 kHz (frecuencia central 27 120 kHz),
40,66-40,70 MHz (frecuencia central 40,68 MHz),
902-928 MHz en la Región 2 (frecuencia central 915 MHz),
2 400-2 500 MHz (frecuencia central 2 450 MHz),
5 725-5 875 MHz (frecuencia central 5 800 MHz) y
24-24,25 GHz (frecuencia central 24,125 GHz)
están designadas para aplicaciones industriales, científicas y médicas (ICM). Los servicios de radiocomunicación que funcionan en estas bandas deben aceptar la interferencia perjudicial resultante de estas aplicaciones. Los equipos ICM que funcionen en estas bandas estarán sujetos a las disposiciones del número **15.13**.

Figura 6: Bandas de frecuencia no licenciada - Plan nacional de frecuencias. ARCOTEL.

CAPÍTULO III. METODOLOGIA.

3.1 Tipo de investigación

La presente investigación fue de carácter experimental. Se desempeño el proceso de diseño y construcción de un dispositivo radar de banda estrecha utilizando radio definido por software (SDR) aplicando el algoritmo de radar de onda continua con múltiples frecuencias, lo que hizo posible el desempeño de su enfoque principal "la detección de objetos y medición de distancias"

3.2 Instrumentos de investigación

La elaboración de esta investigación utilizo diferentes técnicas e instrumentos se clasifico y jerarquizo los artículos, libros y revistas (Análisis documental), se registró los datos medidos (Observación), y se manipulo las variables para la evaluación del desempeño (recolección, análisis e interpretación).

Posteriormente, se procedió con la metodología descriptiva, en la que se relató las características fundamentales del tema, el análisis de los aportes de otras fuentes y finalmente se profundizo con el método experimental, manipulando las variables para observar las alteraciones en el funcionamiento.

3.3 Población y muestra

3.3.1 Población

Se estableció una población finita, donde se recolecto diariamente datos de la distancia medida por el radar hasta un objeto. Los objetos fueron ubicados en diferentes posiciones. Y los datos se tomaron a distintas horas del día, elegidas aleatoriamente.

	Mediciones diarias	**Periodo de medición(días)**	**Mediciones finales**
Posición1 = 100cm	10	30	300
Posición1 = 200cm	10	30	300
Posición1 = 100cm	10	30	300
Posición1 = 300cm	10	30	300
		Total	**1200**

Tabla 1: Periodo de medición y cantidad de datos.

Al cumplir el periodo de 30 días tendremos un total de 1200 datos.

3.3.2 Muestra

La muestra se obtuvo aplicando una fórmula de cálculo de tipo cuantitativo, donde los datos se comportan de una manera relativamente uniforme. Para una población donde se conoce la cantidad de unidades de observación finita que la integran se tiene:

$$n = \frac{N\,Z^2 S^2}{d^2(N-1) + Z^2 S^2}$$

Ecuación 1: Muestra población finita.[5]

Donde:

$n =$ tamaño de la muestra

$N =$ tamaño de la población

$Z =$ Nivel de confianza

$S^2 =$ varianza de la población en estudio (cuadrado de la desviación estándar y puede obtenerse de estudios similares o pruebas piloto)

$d =$ nivel de precisión absoluta. Referido a la amplitud del intervalo de confianza deseado en la determinación del valor promedio de la variable en estudio.

% Error	Nivel de Confianza	Valor de Z calculado en tablas
1	99 %	2.58
5	95 %	1.96
10	90 %	1.645

%	Valor d
90	0.1
95	0.05
99	0.001

Figura 7: Tablas para el cálculo de la muestra. [5]

$$n = \frac{N\,Z^2 S^2}{d^2(N-1) + Z^2 S^2} = \frac{(1200)\,(1.96^2)(7.37^2)}{(0.05^2)(1200-1) + (1.96^2)(7.37^2)} = 492.54$$

$$\approx \quad 493\ Muestras$$

Por facilidad, en la toma de los datos de la muestra, el número será de 500 es decir 125 datos por cada posición.

3.4 Hipótesis

3.4.1 Hipótesis nula

Los datos medidos con el Radar tienen un error de medida mayor o igual que del 5% ($\mu\_1 \geq \mu\_2$).

μ_1 = Error de los datos medidos por el radar con respecto a la medición por cinta métrica.

μ_2 = Rango de tolerancia del error, equivalente al 5%.

3.4.2 Hipótesis alternativa

Los datos medidos con el Radar tienen un error de medida menor que del 5% ($\mu\_1 < \mu\_2$)

3.5 Métodos de análisis

3.5.1 Error Relativo

El error relativo sirve como un indicador de calidad en las mediciones y se puede calcular al dividir el error absoluto y el valor considerado como exacto(media), de igual forma se puede representar a modo de porcentaje.

$$\varepsilon_r = {}^{\varepsilon_a}/_{\bar{X}} \qquad (14)$$

$$\varepsilon_r = {}^{\varepsilon_a}/_{\bar{X}} \,.100\% \qquad (15)$$

Donde:

$\bar{X}$ =Valor considerado real(media)

ε_a =Error absoluto

El error absoluto es un indicador de la imprecisión que tiene una determinada medida, y se obtiene de la diferencia entre un valor real y el valor medido.

$$\varepsilon_a = \bar{X} - X_i \qquad (16)$$

Donde:

$\bar{X}$ =Valor considerado real(media)

X_i =Valor medido

3.5.2 Comparación de medias (t-student)

La comparación de medias o t-student es un método estadístico que sirve para contrastar si existe una diferencia significativa en la medida de una variable, se basa en la comparación de las medias y su error estándar

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{S_1^2}{n_1} - \frac{S_2^2}{n_2}}} \qquad (17)$$

Donde:

$\bar{X}_1$ y $\bar{X}_2$ = Medias a comparar

S_1^2 y S_1^2 = Varianza

n_1 y n_2 = tamaño muestral

3.5.3 Intervalos de Confianza

Se conoce como intervalo de confianza a la estimación puntual del parámetro poblacional, nos da una idea aproximada de si va a servir y nos permite acotar entre dos valores en donde se encontrará la media de la población.

$$\frac{\bar{X} - \mu}{\delta / \sqrt{n}} \sim N(0,1) \qquad (18)$$

Donde:

$\bar{X} =$ Media muestral

$\mu =$ Media poblacional

$\delta =$ Desviación típica(conocida)

$\sqrt{n} =$ Raíz cuadrada del tamaño muestral

3.6 Estructuración los parámetros de transmisión y diseño de las antenas.

3.6.1 Parámetros de transmisión

Como se estableció en el fundamento matemático (punto 2.2), se seleccionaron dos frecuencias de portadora, teniendo en cuenta las bandas de frecuencia disponibles para el desarrollo del proyecto de investigación (punto 2.3) y las especificaciones técnicas del módulo de aprendizaje activo de radio definido por software Adalm Pluto (Anexo 1).

Se estableció que gracias a la cobertura de frecuencias del módulo Adalm Pluto de 325MHz a 3.8GHz, se pudo definir varias bandas, pero se decidió la utilización de la banda de frecuencias de 2400MHz a 2500MHz. Ya que, gracias a su longitud de onda, se pudieron

diseñar antenas más pequeñas, y eficaces. Además, se obtuvo un cambio de frecuencia Δf de hasta 100MHz.

Portadoras (GHz)	Δ_f (MHz)	$Rmax$ (cm)	Constante de calculo
2.4 2.5	100	150	0.23873
2.45 2.5	50	300	0.47746
2.465 2.5	35	420	0.68209
2.475 2.5	25	600	0.95492

Tabla 2: Parámetros de Portadoras seleccionadas.

El radar de onda continua con múltiples frecuencias a diferencia del Radar de onda continúa modulado en frecuencia o el radar de pulsos requiere una menor potencia y un ancho de banda relativamente estrecho en el orden de los Kilohercios, por lo que el ancho de banda máximo del módulo Adalm Pluto equivalente a 20MHz cubrió perfectamente los requerimientos de implementación.

Ancho de banda (Portadora)	Factor de incrementos	Frecuencia de muestreo
2 KHz	1000	2 MHz
5.21KHz	100	521KHz

Tabla 3: Ancho de banda de portadora

3.7 Construcción del prototipo

3.7.1 Diseño de antenas

Para las antenas de transmisión y recepción se seleccionó el diseño de un parche rectangular Microstrip con hendiduras. Para todos los diseños se utilizó un sustrato FR-4 con una constante dieléctrica de 4.3 modelados en CTS studio.

Las antenas fueron alimentadas a través de una terminal SMA hembra con una impedancia de 50Ω.

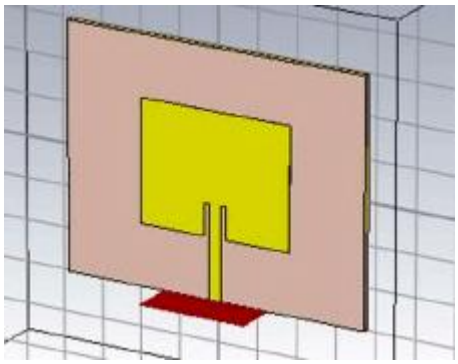


Figura 8: Parche microstrip diseñado y simulado para frecuencias de 2.4, 2.45 y 2.5GHz.

Parámetro/frecuencia	**2.4GHz**	**2.45GHz**	**2.5GHz**
Parche (mm)	37.96 * 29.77	37.96 * 29.14	36.44 * 28.57
Sustrato (mm)	75.92 * 59.54	75.92 *58.28	72.88 * 57.14
Línea de tx (mm)	2.8 * 22.695	2.8* 23.43	2.8 * 22.095
Hendiduras (mm)	1.4 * 7.81	1.4 * 7.81	1.4 * 7.81
Parámetro S11	-45.81	-32.751	-38.568
Ganancia (dB)	2.184	2.225	2.113
Directividad (dBi)	6.773	6.759	6.690
Impedancia real (Ω)	49.936	49.928	49.935

Tabla 4: Parámetros físicos del diseño de Antenas.

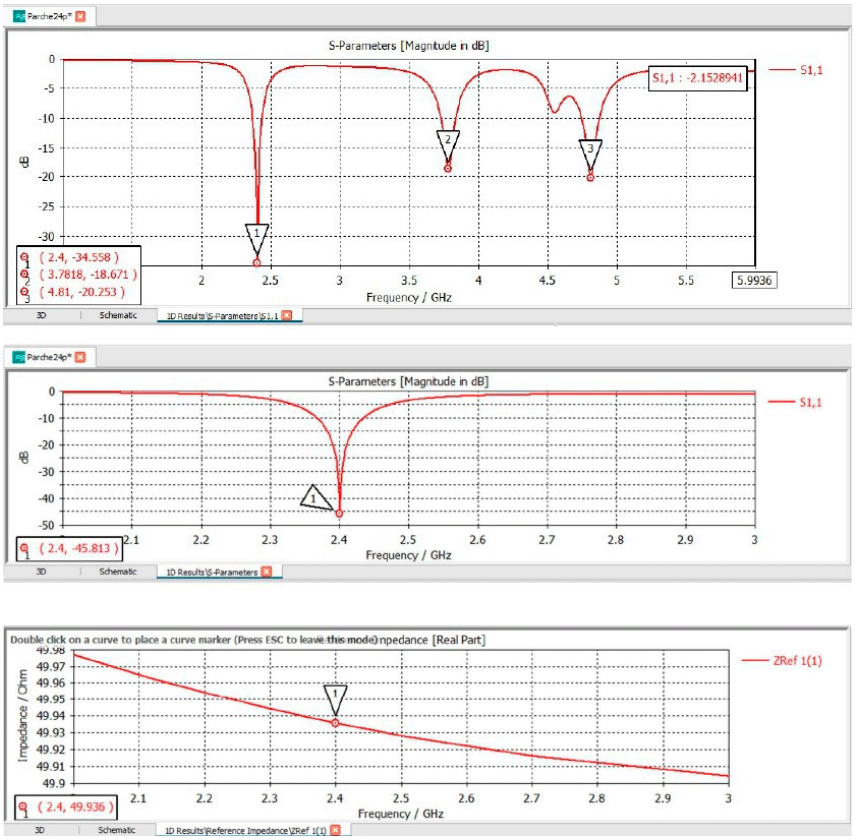


Figura 9: Simulación de la antena parche, de 2.4GHz, Parámetros s e impedancia.

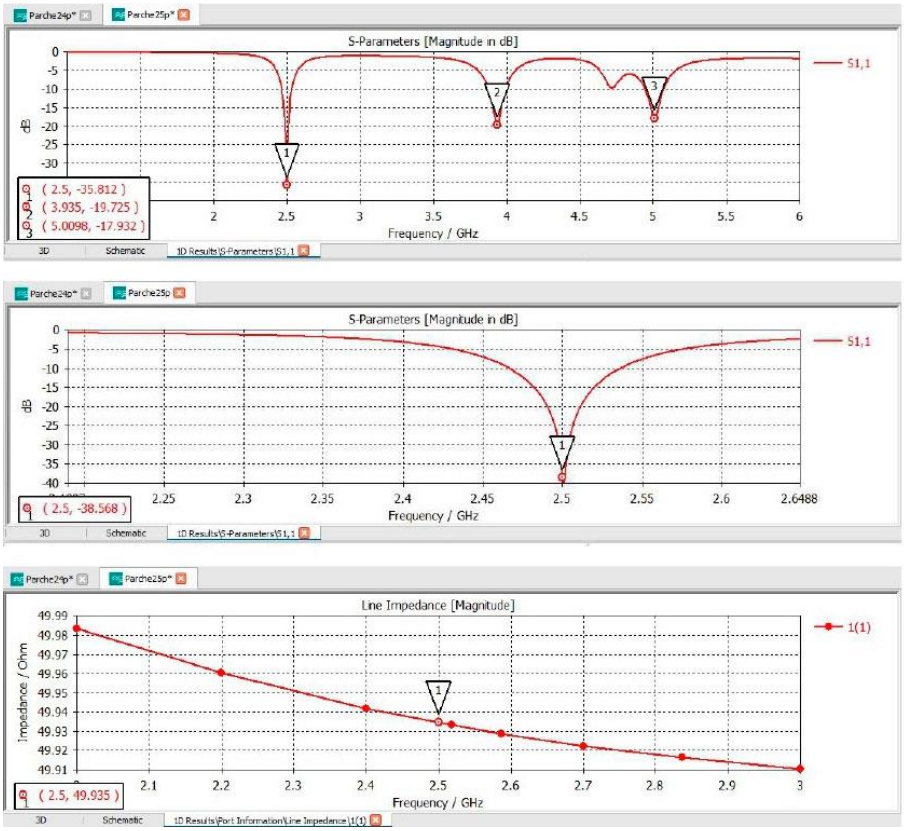


Figura 10: Simulación de la antena parche, de 2.5GHz, Parámetros s e impedancia.

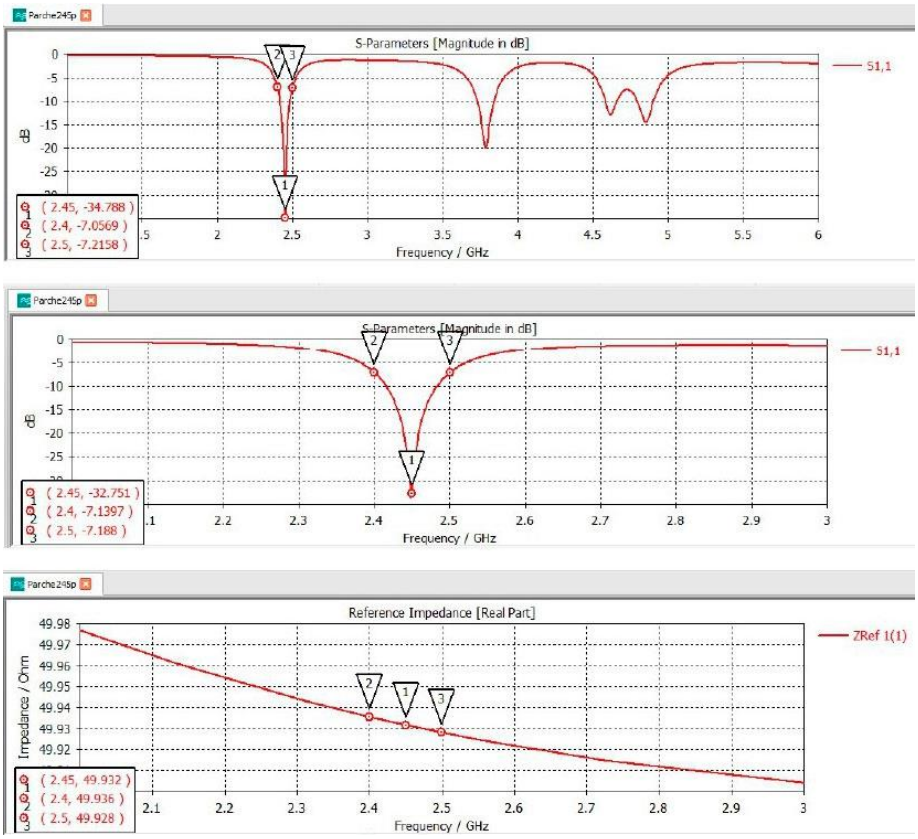


Figura 11: Simulación de la antena parche, de 2.45GHz, Parámetros s e impedancia.

Como se pudo observar en la figura 9-11 las simulaciones ofrecen un funcionamiento de las antenas aproximado por lo que posteriormente se realizó pruebas a las antenas en el analizador de espectros, para observar su desempeño real, como se puede observar en la figura 12.

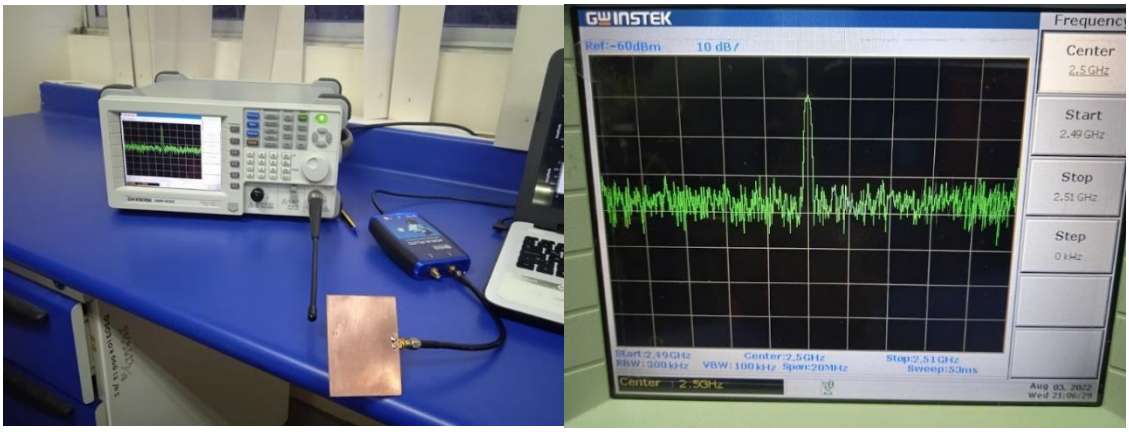


Figura 12: Visualización de una portadora de 2.5GHz generada por el SDR y transmitida a través de las antenas diseñadas, utilizando el analizador de espectros.

Los cables que llevaron las señales transmitidas y receptadas desde el SDR hasta las antenas y viceversa fueron construidos con base en cable coaxial estándar RG-58 de impedancia 50Ω con terminales SMA macho de impedancia 50Ω.

3.7.2 Interfaces de Transmisión y Recepción en GNURadio

En primer lugar, se desarrolló un Radar de onda continua con múltiples frecuencias (MFCW) con el uso de un solo módulo Adalm Pluto, una antena transmisora y una antena receptora, el cambio de portadoras se hace de forma manual, figura 13. Se detectó el inconveniente de que, se necesita una recalibración inicial con cada cambio de portadora.

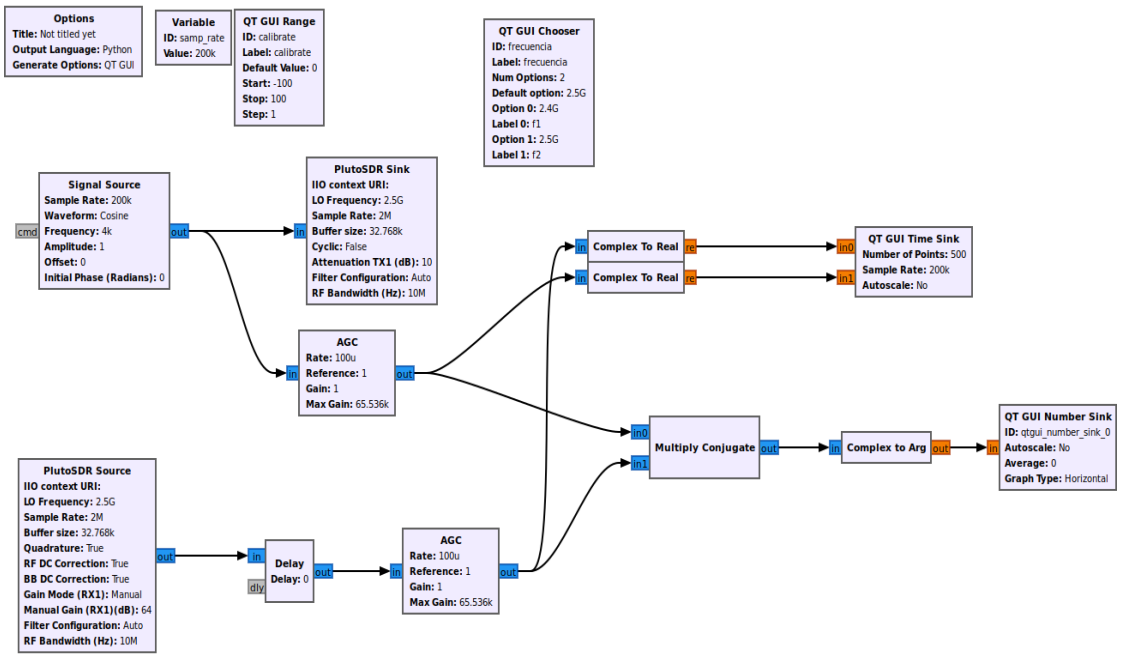
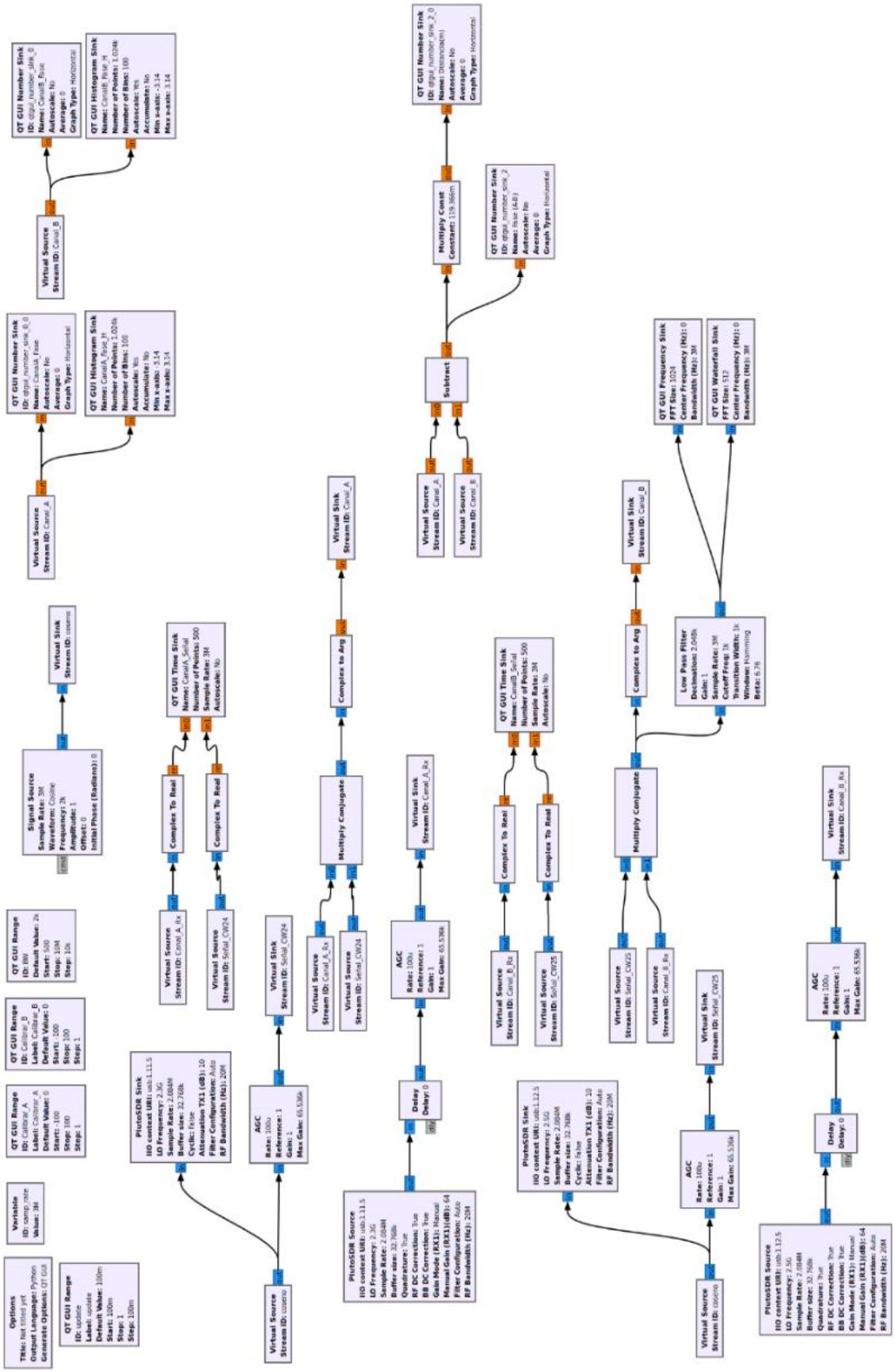


Figura 13: Diseño preliminar con cambio de portadora manual.

Para solventar este inconveniente se desarrolló un nuevo diseño del Radar, con un SDR por cada portadora, dos antenas transmisoras y 2 receptoras, figura 14.

Finalmente, se realizó una mejora significativa al diseño del radar, añadiendo un sistema para contrarrestar la interferencia por fuga del canal, eliminando así que la señal trasmitida, incida directamente sobre la antena receptora y así poder evitar errores en la lectura de datos, también se añadió un análisis paralelo en uno de los canales CW para poder obtener una lectura aproximada de velocidad, figura 15.



5Figura 14: Diseño del radar MFCW.

3.7.2.1 Etapas del diseño

Como se puede observar en la figura 15, el diseño cuenta de diferentes etapas descritas a continuación:

Variables: Son palabras a las que se les asigna un valor fijo o un rango de valores, ayudan en la asignación de parámetros en los bloques.

Generación de señal: Se compone del bloque *signal source* el cual construye una señal cosenoidal con una frecuencia definida y sin interrupciones.

Transmisores: Los bloques *PlutoSDR Sink,* corresponden a los transmisores SDR. Cada transmisor se alimenta de la señal coseno generada y la elevan a la frecuencia del oscilador 2.475GHz y 2.5GHz, esta señal es enviada a las antenas.

Receptores: Los bloques *PlutoSDR Source*, corresponden a los receptores SDR. Reciben señales desde las antenas, tienen el oscilador configurado para recibir las señales que contengan su frecuencia base.

Control de fugas: Se utiliza la misma señal CW generada, pero se desplaza en fase y amplitud con el objetivo de eliminar señales inducidas directamente del transmisor, esta etapa puede no ser necesaria, si se tiene un buen espaciamiento entre la antena Transmisora y receptora de cada portadora.

Cálculo de distancia: Una vez obtenida la señal de los receptores se elimina la portadora de cada frecuencia con el bloque *multiply conjugate*, posteriormente se realiza la fórmula matemática de atención de distancia, a modo de bloques restando las 2 componentes desplazando la frecuencia y multiplicándolo por una constante previamente calculada, tabla 1.

Cálculo de velocidad (Adicional): Como cada portadora es independiente de la otra se puede tomar su componente para hacer el análisis de velocidad, sin afectar el cálculo de distancia.

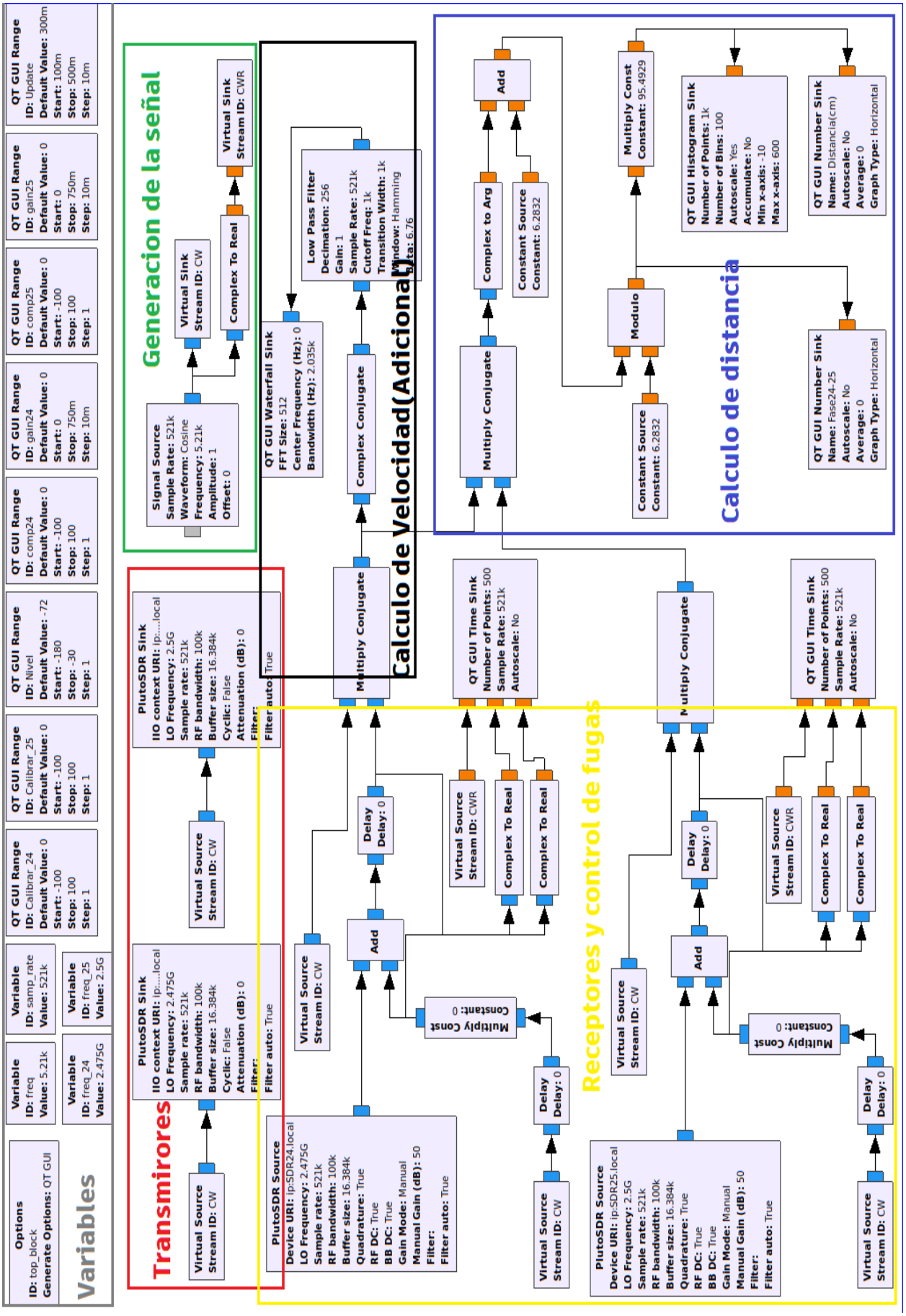


Figura 15: Diseño final separado por etapas.

3.7.3 Implementación física del radar y pruebas

Para la implementación física del radar, se construyó un soporte metálico en el que se montaron las antenas de transmisión y recepción de cada portadora, de tal forma que se evite una inducción directa de la señal, figura 18.

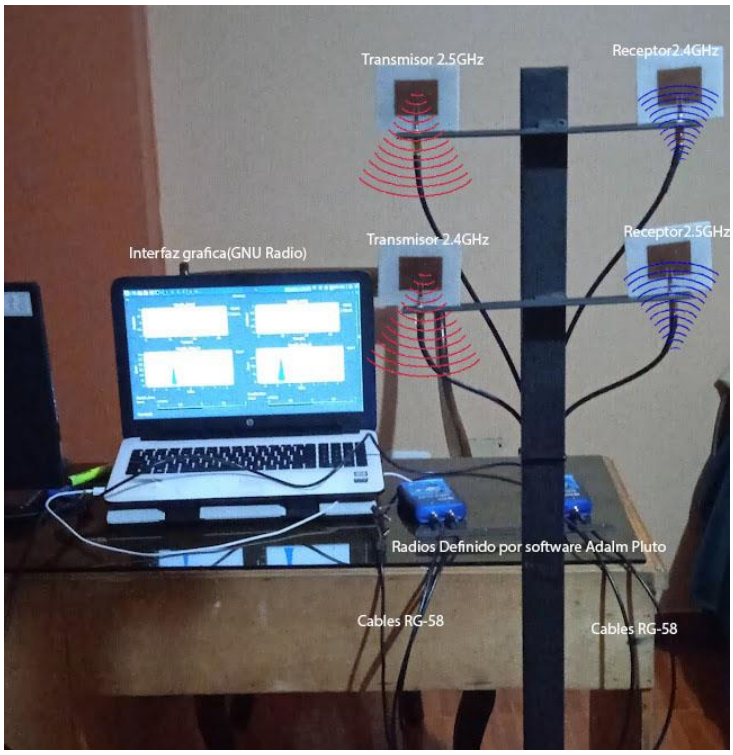


Figura 16: Prototipo de Radar de onda continua con múltiples frecuencias.

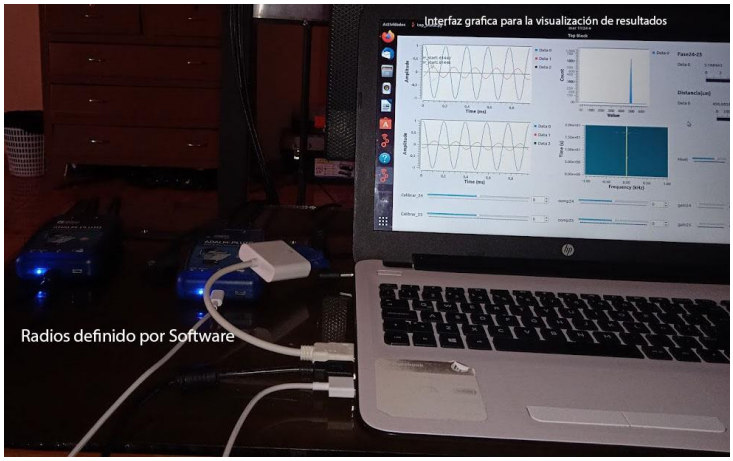


Figura 17: Interfaz gráfica del Radar, donde se observará el cambio de fase y la medición de distancia.

Una vez concluida la construcción del prototipo se ubicó en el laboratorio de electrónica de la universidad nacional de Chimborazo donde se efectuó las pruebas de ejecución como se puede observar en las figuras 18 - 20.

Figura 18: Ubicación del prototipo en el laboratorio de electrónica de la Universidad nacional de Chimborazo (2022), para la realización de pruebas.

Figura 19: Detección del objeto reflejante y medición de distancia.

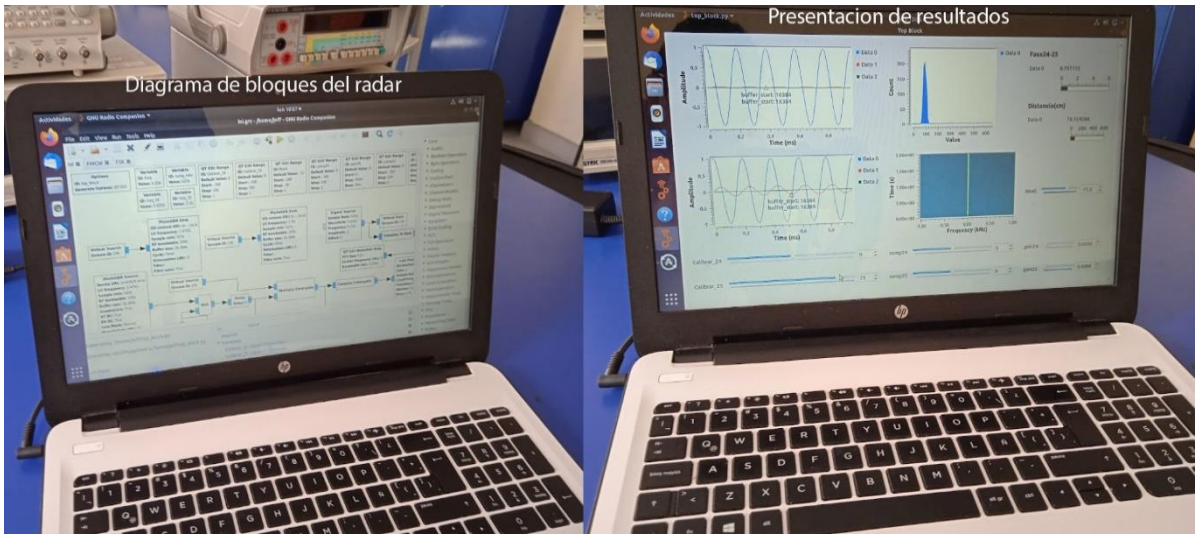


Figura 20: Interfaz gráfica donde se puede evidenciar el diagrama de bloques del radar y la presentación de resultados.

Después de haber realizado las primeras mediciones se detectó que el soporte metálico no se encontraba aislado correctamente y por consiguiente afectaba el patrón de radiación de las antenas e influía negativamente en las mediciones, para continuar con las pruebas se decidió cambiar a un soporte realizado en madera, aumentando también la separación vertical y horizontal de las antenas como se puede observar en las figuras 21 - 25.

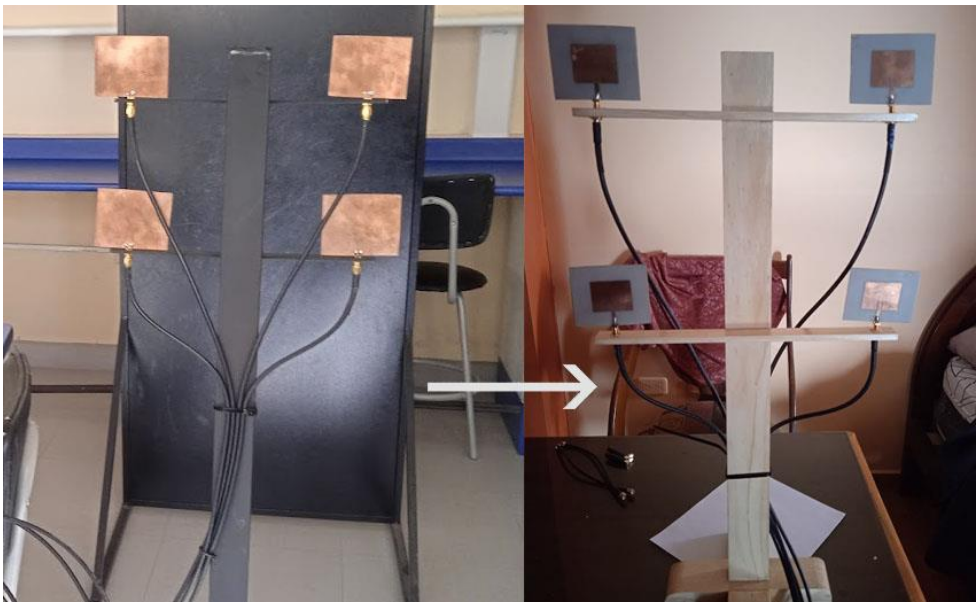


Figura 21: Cambio de soporte de antenas.

Figura 22: Ubicación del radar para nuevas pruebas.

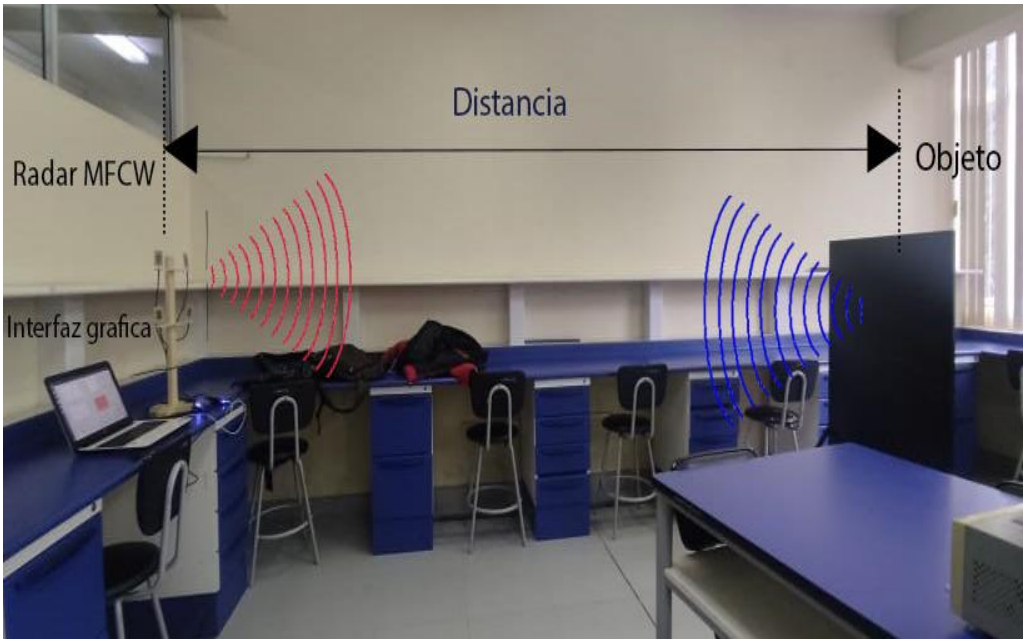


Figura 23: Representación visual del funcionamiento.

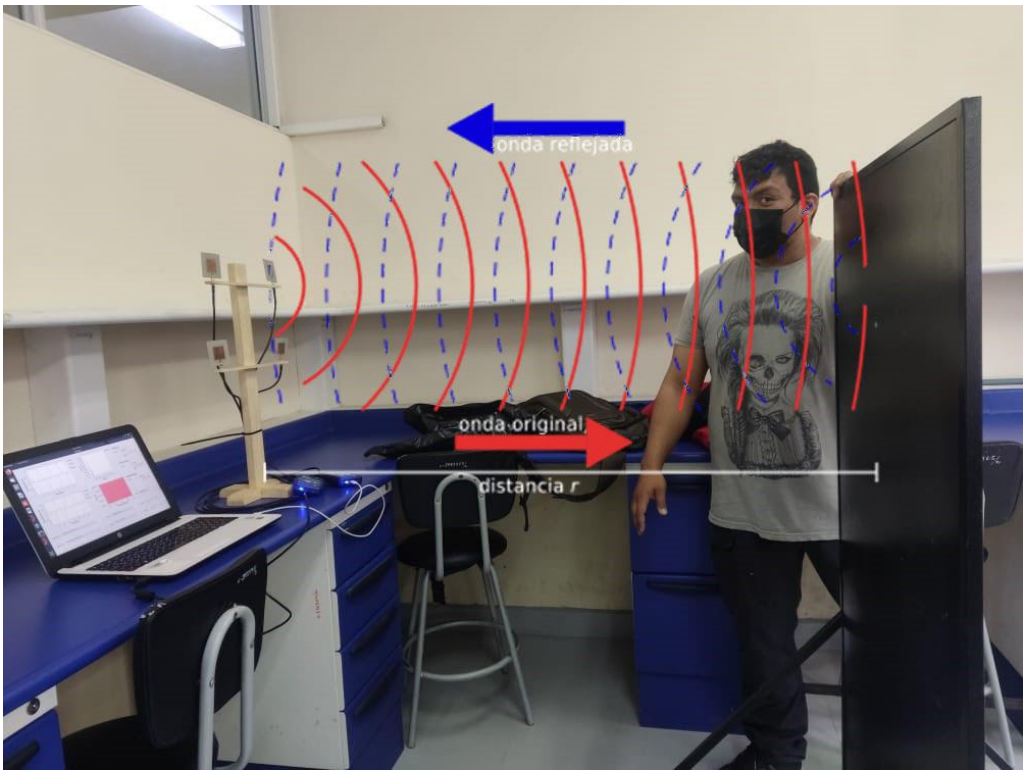


Figura 24: Representación visual del funcionamiento del radar.

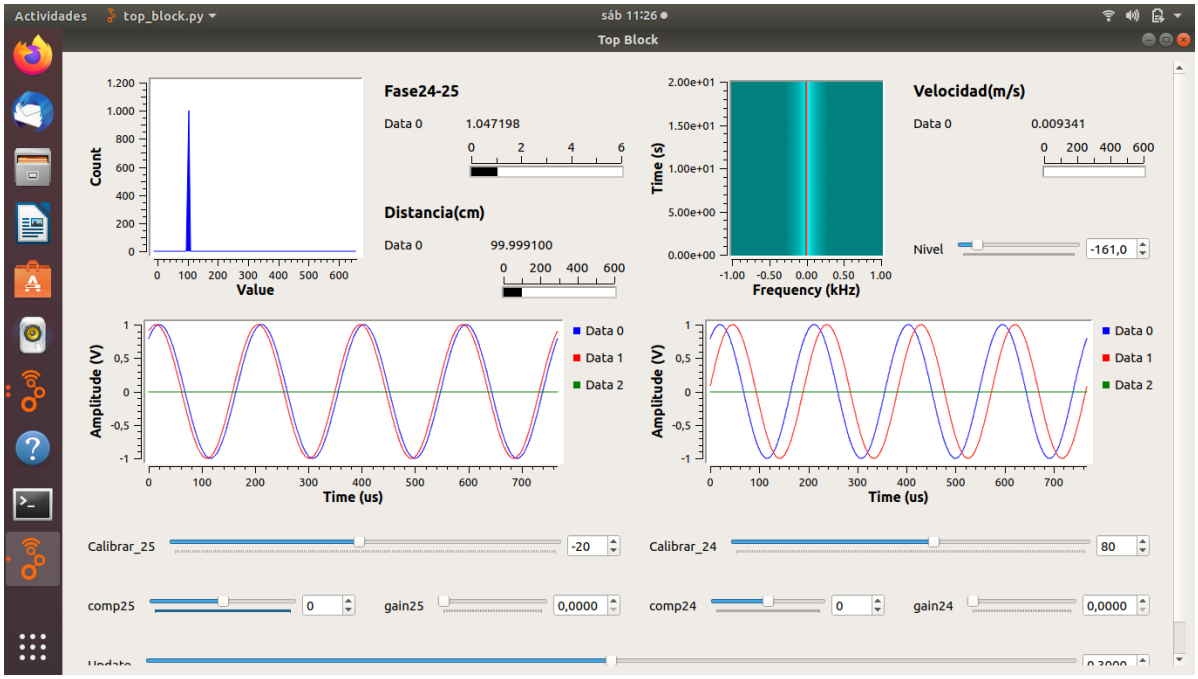


Figura 25: Presentación de los resultados de funcionamiento del radar.

CAPÍTULO IV. RESULTADOS Y DISCUSIÓN

4.1 Resultados

Como se propuso, al iniciar la investigación se fue tomando diariamente datos de distancia tomada por el radar por un periodo de 30 días, se ubicó como objetivo un panel metálico que se puede desplazar. Este panel se ubicó en las posiciones establecidas de 100, 200, 300 y 400 cm, que fueron medidos por un instrumento de medición estándar (Flexómetro).

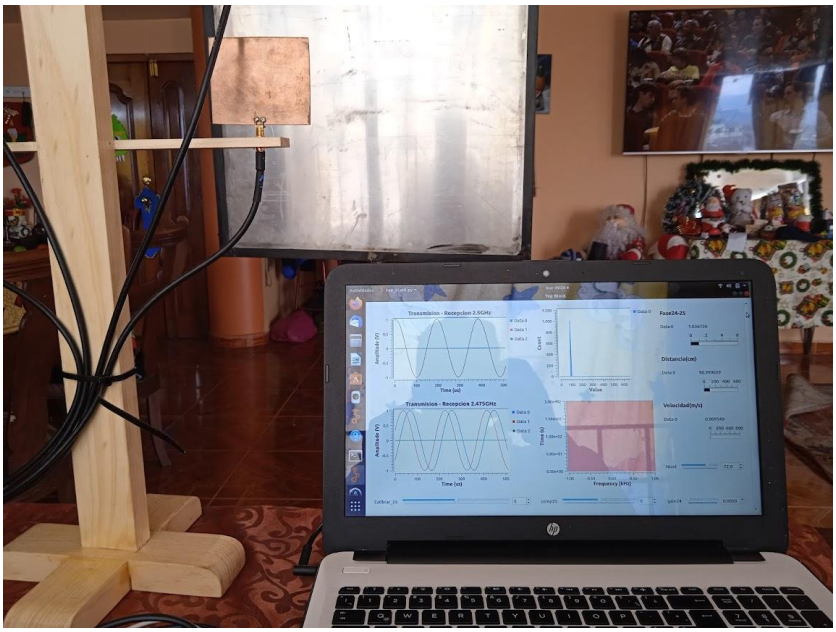


Figura 26: Ubicación del panel(objetivo) a 100cm, para la recolección de datos.

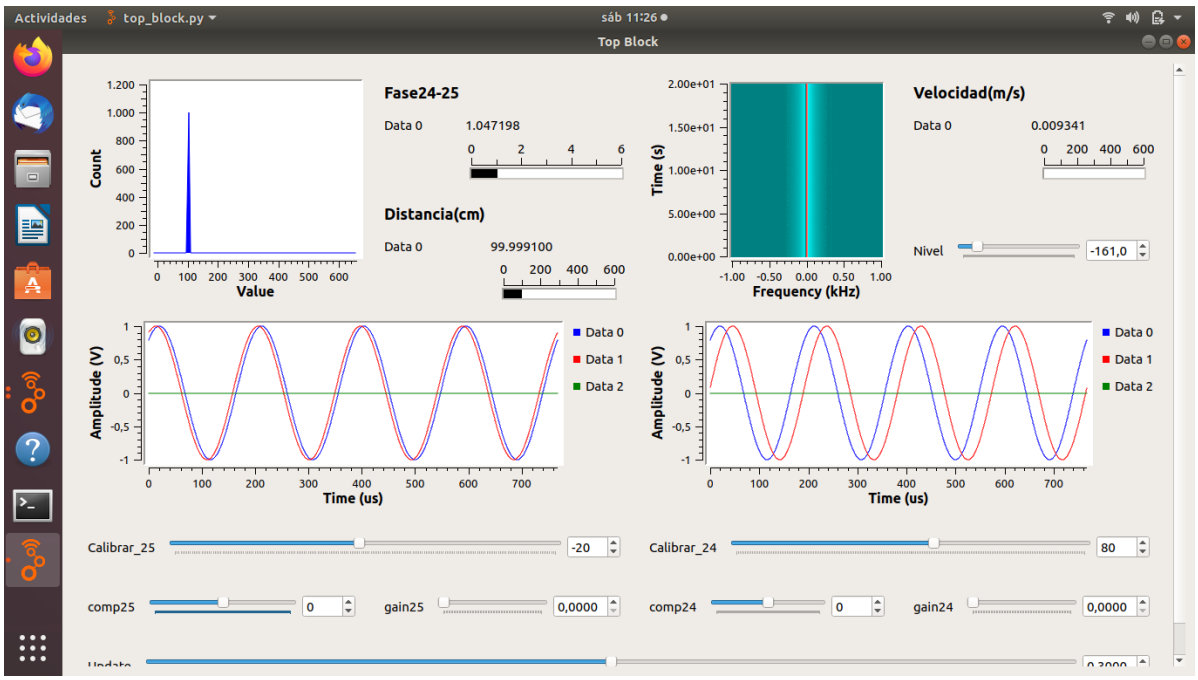


Figura 27: Ejemplo de medición de 100cm para la recolección de datos.

Figura 28: Ubicación del panel(objetivo) a 200cm, para la recolección de datos.

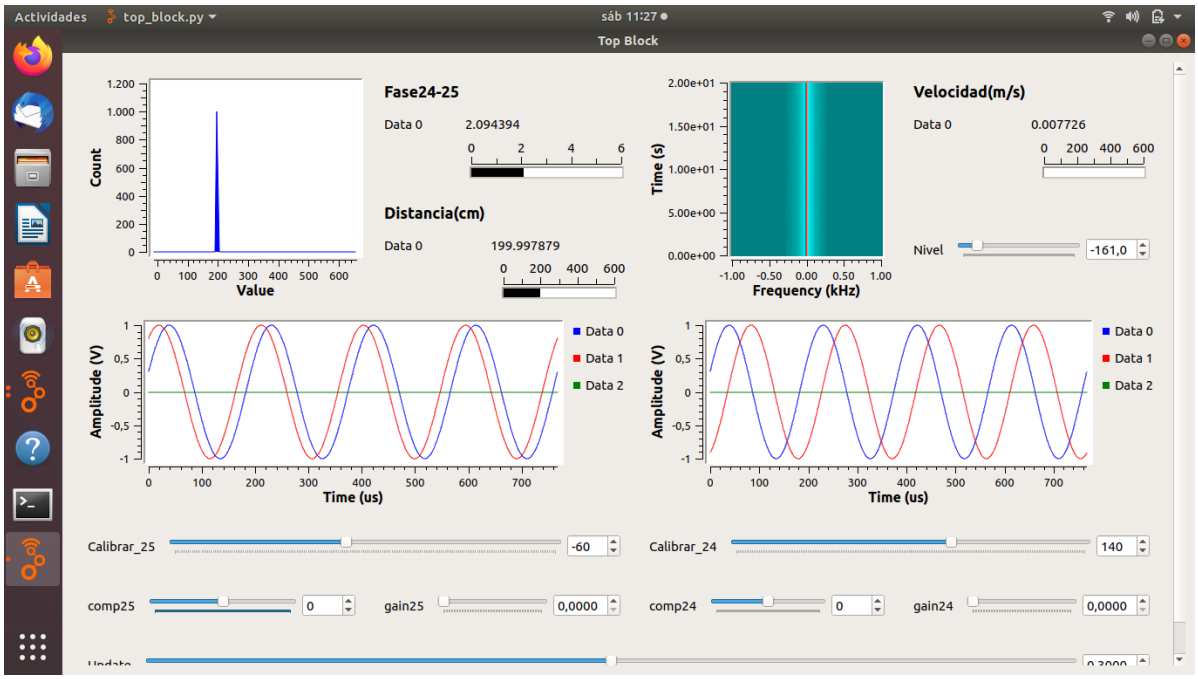


Figura 29: Ejemplo de medición de 200cm para la recolección de datos.

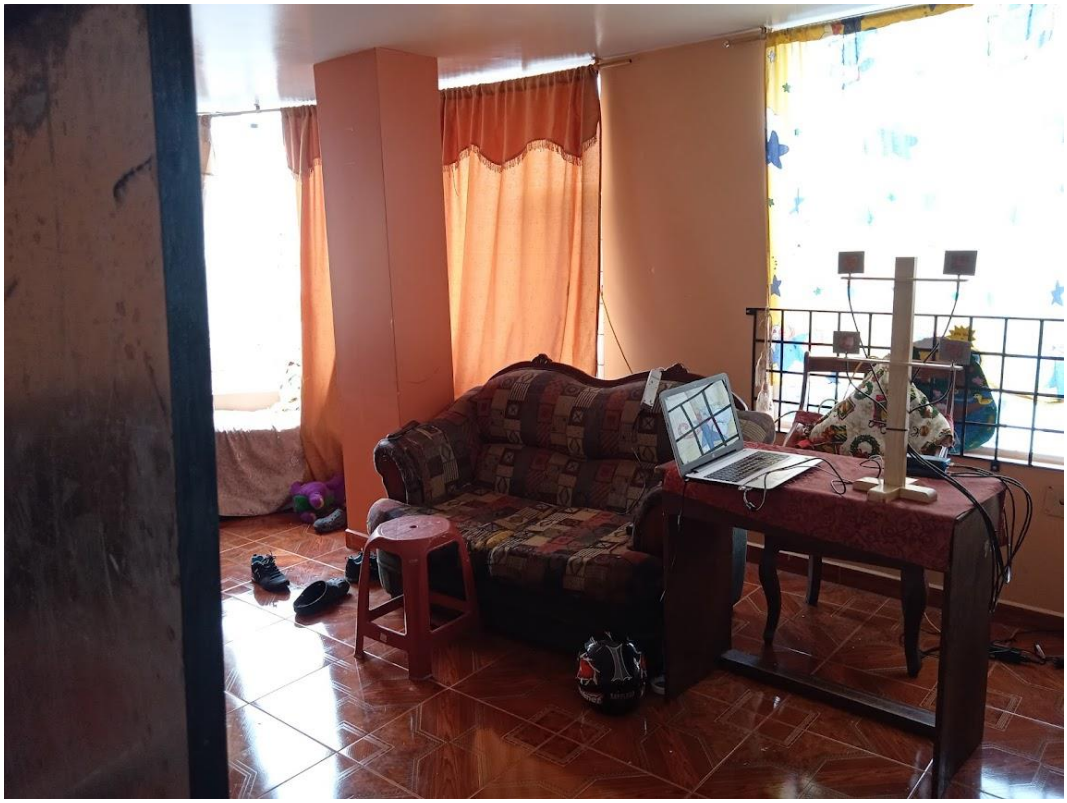


Figura 30: Ubicación del panel(objetivo) a 300cm, para la recolección de datos.

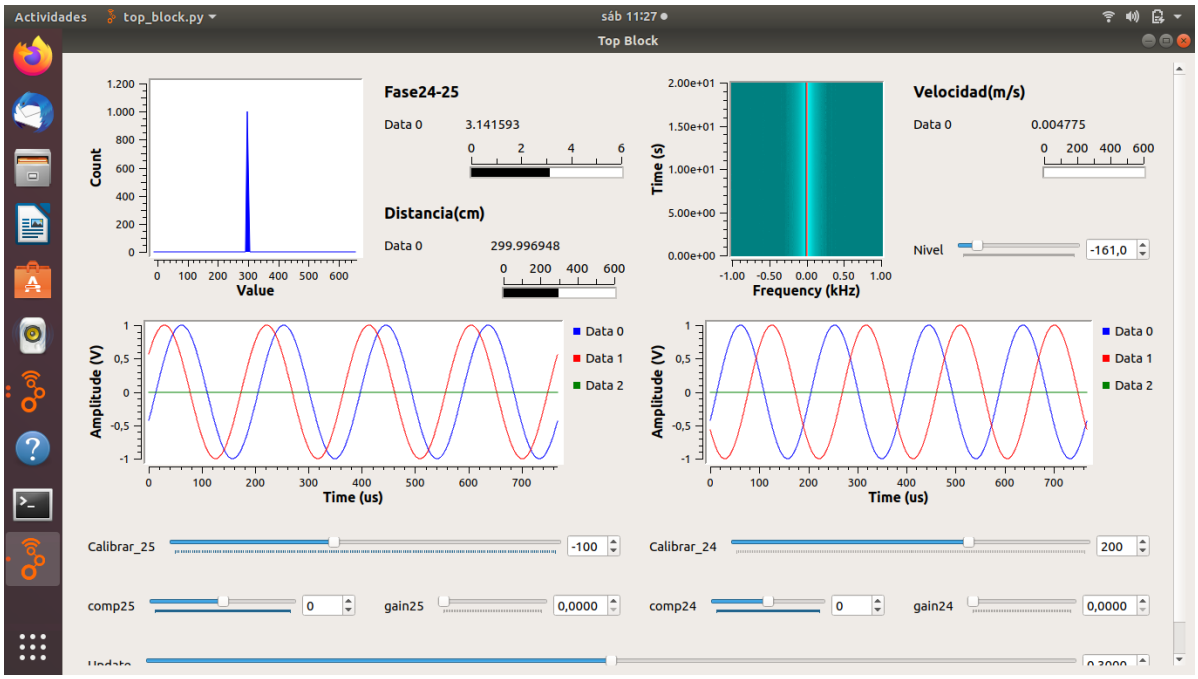


Figura 31: Ejemplo de medición de 300cm para la recolección de datos.

Figura 32:Ubicación del panel(objetivo) a 400cm, para la recolección de datos.

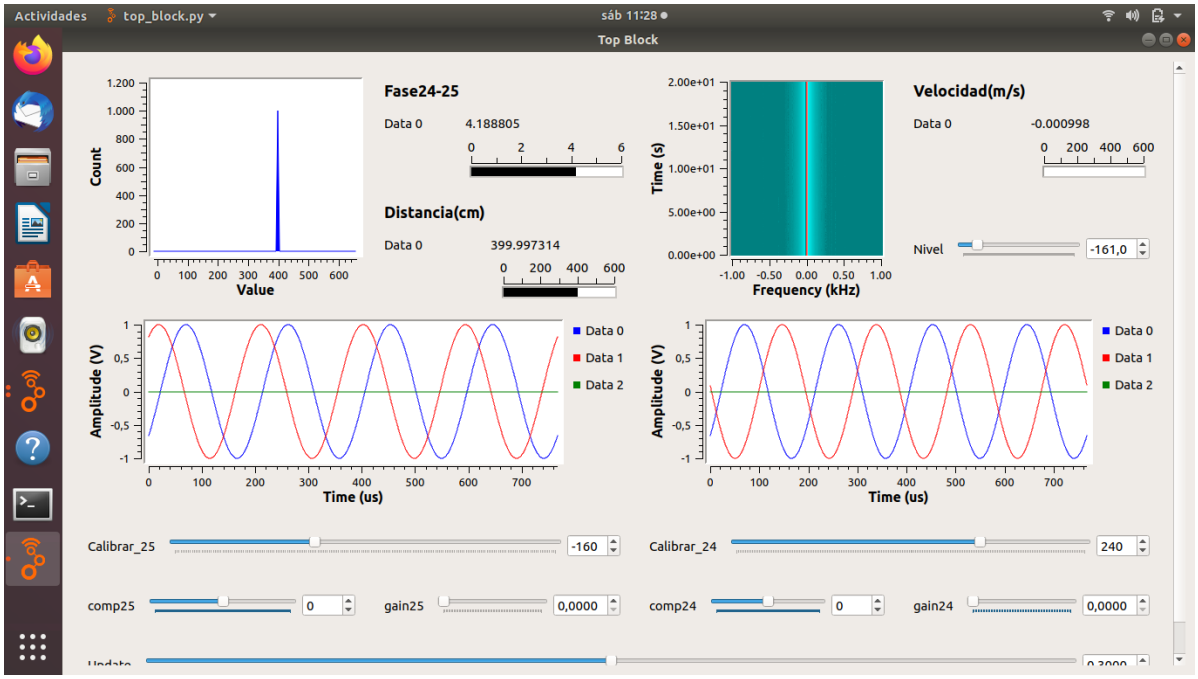


Figura 33: Ejemplo de medición de 400cm para la recolección de datos.

Una vez ubicado el panel metálico se tomó la lectura del radar recolectando diariamente 10 datos en cada posición, por lo que al final de cada día tenemos 40 datos, tabla 5, 6.

Medicion diaria de distancia usando el dispositivo Radar MFCW — Selección de portadoras sin ambigüedad

	Frecuencia 1 (GHz)	Frecuencia 2 (GHz)	Diferencia de frecuencia (GHz)	Distancia Maxima sin ambigüedad (cm)	Medida con flexometro (cm)	Medicion por Radar (cm)	Diferencia en la medicion(cm)	Observacion
1	2,465	2,5	0,035	428,5714286	100	98,999992	1,000008	
2	2,465	2,5	0,035	428,5714286	100	98,999962	1,000038	
3	2,465	2,5	0,035	428,5714286	100	98,999931	1,000069	
4	2,465	2,5	0,035	428,5714286	100	98,999931	1,000069	
5	2,465	2,5	0,035	428,5714286	100	98,999107	1,000893	
6	2,475	2,5	0,025	600	100	97,99897	2,00103	
7	2,475	2,5	0,025	600	100	97,99894	2,00106	
8	2,475	2,5	0,025	600	100	97,999962	2,000038	
9	2,475	2,5	0,025	600	100	97,999039	2,000961	
10	2,475	2,5	0,025	600	100	98,999107	1,000893	
11	2,465	2,5	0,035	428,5714286	200	198,999954	1,000046	
12	2,465	2,5	0,035	428,5714286	200	198,999974	1,000026	
13	2,465	2,5	0,035	428,5714286	200	198,000458	1,999542	
14	2,465	2,5	0,035	428,5714286	200	198,000443	1,999557	
15	2,465	2,5	0,035	428,5714286	200	198,000688	1,999312	
16	2,475	2,5	0,025	600	200	197,997864	2,002136	
17	2,475	2,5	0,025	600	200	197,997879	2,002121	
18	2,475	2,5	0,025	600	200	197,997818	2,002182	
19	2,475	2,5	0,025	600	200	197,998703	2,001297	
20	2,475	2,5	0,025	600	200	197,998672	2,001328	
21	2,465	2,5	0,035	428,5714286	300	299,000824	0,999176	
22	2,465	2,5	0,035	428,5714286	300	299,000885	0,999115	
23	2,465	2,5	0,035	428,5714286	300	299,000824	0,999176	
24	2,465	2,5	0,035	428,5714286	300	299,000916	0,999084	
25	2,465	2,5	0,035	428,5714286	300	299,000884	0,999116	
26	2,475	2,5	0,025	600	300	298,996918	1,003082	
27	2,475	2,5	0,025	600	300	298,996948	1,003052	
28	2,475	2,5	0,025	600	300	298,998291	1,001709	
29	2,475	2,5	0,025	600	300	298,99884	1,00116	
30	2,475	2,5	0,025	600	300	298,99974	1,00026	
31	2,465	2,5	0,035	428,5714286	400	400,000793	0,000793	
32	2,465	2,5	0,035	428,5714286	400	400,000844	0,000844	
33	2,465	2,5	0,035	428,5714286	400	400,000793	0,000793	
34	2,465	2,5	0,035	428,5714286	400	399,99998	2E-05	
35	2,465	2,5	0,035	428,5714286	400	399,99995	5E-05	
36	2,475	2,5	0,025	600	400	400,000275	0,000275	
37	2,475	2,5	0,025	600	400	400,000214	0,000214	
38	2,475	2,5	0,025	600	400	400,000916	0,000916	
39	2,475	2,5	0,025	600	400	399,99955	0,00045	
40	2,475	2,5	0,025	600	400	400,000793	0,000793	

Tabla 5: Tabla de mediciones diarias (ejemplo día 4).

Medicion diaria de distancia usando el dispositivo Radar MFCW — Delimitacion a ambigüedad 600cm

	Frecuencia 1 (GHz)	Frecuencia 2 (GHz)	Diferencia de frecuencia (GHz)	Distancia Maxima sin ambigüedad (cm)	Medida con flexometro (cm)	Medicion por Radar (cm)	Diferencia en la medicion(cm)	Observacion
1	2,475	2,5	0,025	600	100	98,999892	1,000108	
2	2,475	2,5	0,025	600	100	98,990072	1,009928	
3	2,475	2,5	0,025	600	100	98,998847	1,001153	
4	2,475	2,5	0,025	600	100	98,999477	1,000523	
5	2,475	2,5	0,025	600	100	98,999474	1,000526	
6	2,475	2,5	0,025	600	100	98,990054	1,009946	
7	2,475	2,5	0,025	600	100	98,999072	1,000928	
8	2,475	2,5	0,025	600	100	98,998882	1,001118	
9	2,475	2,5	0,025	600	100	98,999892	1,000108	
10	2,475	2,5	0,025	600	100	99,000057	0,999943	
11	2,475	2,5	0,025	600	200	199,000054	0,999946	
12	2,475	2,5	0,025	600	200	199,000747	0,999253	
13	2,475	2,5	0,025	600	200	198,999854	1,000146	
14	2,475	2,5	0,025	600	200	198,99982	1,00018	
15	2,475	2,5	0,025	600	200	198,998192	1,001808	
16	2,475	2,5	0,025	600	200	198,999879	1,000121	
17	2,475	2,5	0,025	600	200	198,997864	1,002136	
18	2,475	2,5	0,025	600	200	198,998291	1,001709	
19	2,475	2,5	0,025	600	200	198,998077	1,001923	
20	2,475	2,5	0,025	600	200	199,000854	0,999146	
21	2,475	2,5	0,025	600	300	299,000443	0,999557	
22	2,475	2,5	0,025	600	300	299,000824	0,999176	
23	2,475	2,5	0,025	600	300	298,997854	1,002146	
24	2,475	2,5	0,025	600	300	298,99992	1,00008	
25	2,475	2,5	0,025	600	300	299,14143	0,85857	
26	2,475	2,5	0,025	600	300	299,74625	0,25375	
27	2,475	2,5	0,025	600	300	299,000824	0,999176	
28	2,475	2,5	0,025	600	300	298,999918	1,000082	
29	2,475	2,5	0,025	600	300	298,998291	1,001709	
30	2,475	2,5	0,025	600	300	298,999866	1,000134	
31	2,475	2,5	0,025	600	400	400,00824	0,00824	
32	2,475	2,5	0,025	600	400	399,000443	0,999557	
33	2,475	2,5	0,025	600	400	399,000824	0,999176	
34	2,475	2,5	0,025	600	400	399,000824	0,999176	
35	2,475	2,5	0,025	600	400	399,000793	0,999207	
36	2,475	2,5	0,025	600	400	398,999352	1,000648	
37	2,475	2,5	0,025	600	400	398,99947	1,00053	
38	2,475	2,5	0,025	600	400	398,998825	1,001175	
39	2,475	2,5	0,025	600	400	398,99894	1,00106	
40	2,475	2,5	0,025	600	400	398,999844	1,000156	

Tabla 6: Tabla de mediciones diarias (ejemplo día 18).

Una vez concluido el periodo de recolección de datos (30 días), se clasificó y separo las mediciones, se obtuvo una población total de 1200 datos, 300 por cada posición, y se procedió a tomar una muestra aleatoria. Donde obtuvimos 125 datos de cada posición para proceder con la evaluación de los resultados.

	Medida100cm	Medida200cm	Medida300cm	Medida400cm
276	98,999474	198,999970	298,997340	399,999107
277	98,999470	199,000820	298,967940	400,006680
278	98,999892	198,999900	298,997860	399,999779
279	98,999477	198,999790	298,967870	399,998917
280	98,999866	198,999800	298,999950	399,998170
281	98,998866	198,999950	299,000820	400,008240
282	98,998772	198,999970	299,000240	399,999039
283	98,999866	198,999990	299,000270	399,999100
284	98,999784	199,000980	299,007650	400,009160
285	98,999892	198,998910	298,997370	399,999077
286	98,999474	198,999970	298,997340	399,999107
287	98,999470	199,000820	298,967940	400,006680
288	98,999892	198,999900	298,997860	399,999779
289	98,999477	198,999790	298,967870	399,998917
290	98,999866	198,999800	298,999950	399,998170
291	98,998854	198,999950	299,000820	400,008240
292	98,999774	198,999970	299,000240	399,999039
293	98,999866	198,999990	299,000270	399,999100
294	98,999917	199,000980	299,007650	400,009160
295	98,999892	198,998910	298,997370	399,999077
296	98,999474	198,999970	298,997340	399,999107
297	98,999774	199,000820	298,967940	400,006680
298	98,999892	198,999900	298,997860	399,999779
299	98,999425	198,999790	298,967870	399,998917
300	98,999794	198,999800	298,999950	399,998170

Figura 34: Clasificación de datos para obtener la población.

	Medida100cm	Medida200cm	Medida300cm	Medida400cm
101	98,999866	198,999800	298,999950	399,998170
102	98,998772	198,999970	299,000240	399,999039
103	98,999892	198,998910	298,997370	399,999077
104	98,999892	198,999900	298,997860	399,999779
105	98,998772	198,999970	299,000240	399,999039
106	98,999866	198,999990	299,000270	399,999100
107	98,999784	199,000980	299,007650	400,009160
108	98,999892	198,998910	298,997370	399,999077
109	98,999474	198,999970	298,997340	399,999107
110	98,999866	198,999800	298,999950	399,998170
111	98,998772	198,999970	299,000240	399,999039
112	98,999866	198,999990	299,000270	399,999100
113	98,999784	199,000980	299,007650	400,009160
114	98,999474	198,999970	298,997340	399,999107
115	98,999470	199,000820	298,967940	400,006680
116	98,998866	198,999950	299,000820	400,008240
117	98,999784	199,000980	299,007650	400,009160
118	98,999892	198,999900	298,997860	399,999779
119	98,998854	198,999950	299,000820	400,008240
120	98,999866	198,999990	299,000270	399,999100
121	98,999892	198,998910	298,997370	399,999077
122	98,999474	198,999970	298,997340	399,999107
123	98,999774	199,000820	298,967940	400,006680
124	98,999892	198,999900	298,997860	399,999779
125	98,999425	198,999790	298,967870	399,998917

Figura 35: Muestra tomada aleatoriamente de la población, 125 datos por cada posición.

Al concluir el proyecto, el radar fue capaz de detectar los objetos y obtener la de distancia del radar a dicho objeto (panel metálico), datos muy cercanos a la lectura real(flexómetro). Además de capturar la presencia de objetos en movimiento al observar cambios bruscos en la fase o en la lectura de distancia. Gracias a dichas propiedades, el radar tendría aplicaciones como medición de distancia hasta un obstáculo, objetos en movimiento (vehículos autónomos, personas sin visión), medidor de distancia o velocidad (radares en carretera), e incluso educativos al ser de código abierto.

4.2 Interpretación y análisis de Resultados

Una vez concluida las pruebas y la recolección de datos de la población, se procedió a evaluar la muestra y con ello analizar los resultados, se realizó pruebas estadísticas, test de normalidad, comparación de medias en intervalos de confianza al 95% y un control de calidad de las mediciones para cada una de las posiciones. A continuación, se presenta los resultados a 100cm.

Medida con el Radar a 100 cm

Gráfico de Dispersión

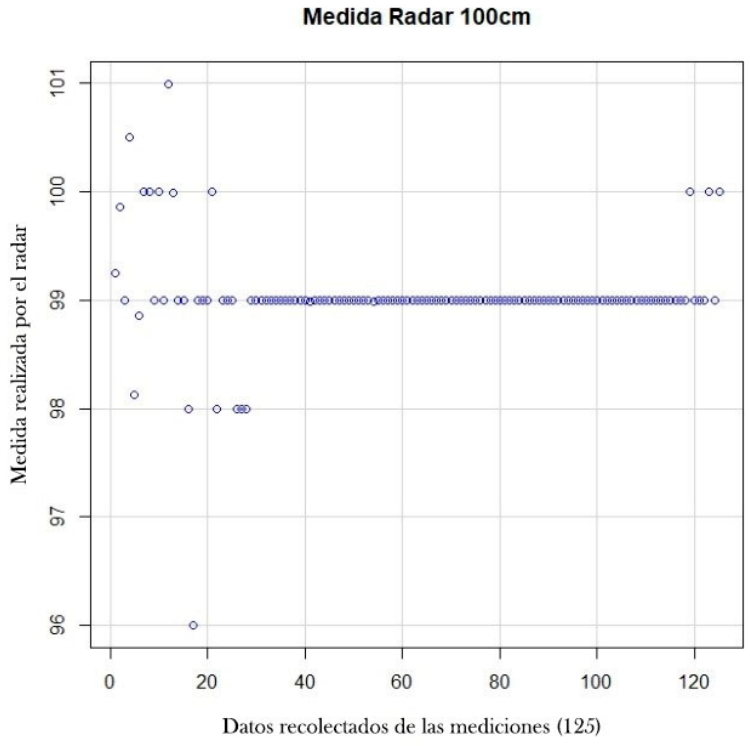


Figura 36: Grafico de frecuencias de medidas a 100cm.

Estadísticos

mean	Sd	IQR	cv	0%	25%	50%	75%	100%	n
99.02813	0.4894622	0.00082	0.004942659	95.991	98.99907	98.99978	99.99989	100.9991	125

Tabla 7: Datos de los resultados estadísticos.

Histograma de Frecuencias

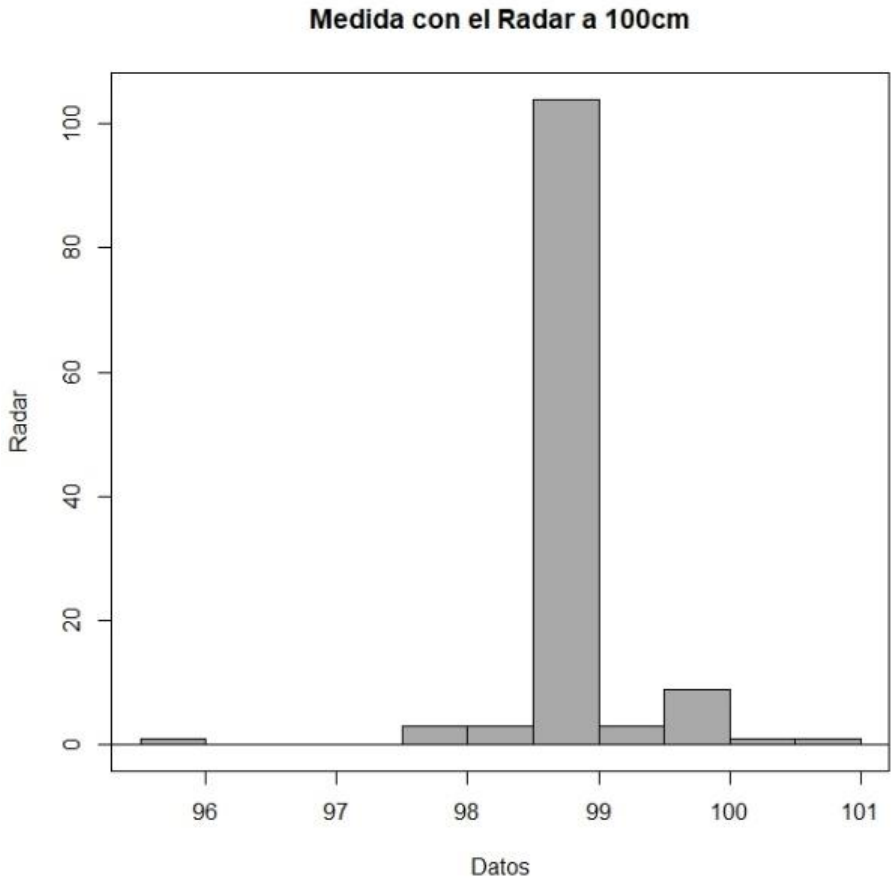


Figura 37: Histograma de frecuencias de medidas a 100cm.

Test de Normalidad

Shapiro-Wilk normality test

data: Medida.por.Radar

W = 0.51718, p-value < 2.2e-16

Puesto que el p-valor es menor que 0.05 (95% de significancia), los datos proceden de una Distribución de Probabilidades Normalizada.

Regresión lineal (ajuste de los datos a un modelo lineal)

```
Residuals:
     Min       1Q   Median       3Q      Max
-3.01299 -0.04087 -0.03101 -0.01952  1.98873

Coefficients:
             Estimate Std. Error  t value Pr(>|t|)
(Intercept) 99.0061637  0.0884135 1119.808   <2e-16 ***
Datos        0.0003487  0.0012178    0.286    0.775
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4913 on 123 degrees of freedom
Multiple R-squared:  0.0006661, Adjusted R-squared:  -0.007459
F-statistic: 0.08199 on 1 and 123 DF,  p-value: 0.7751
```

$$Medida = 99.006 + 0.0003\ (datos) + \epsilon$$

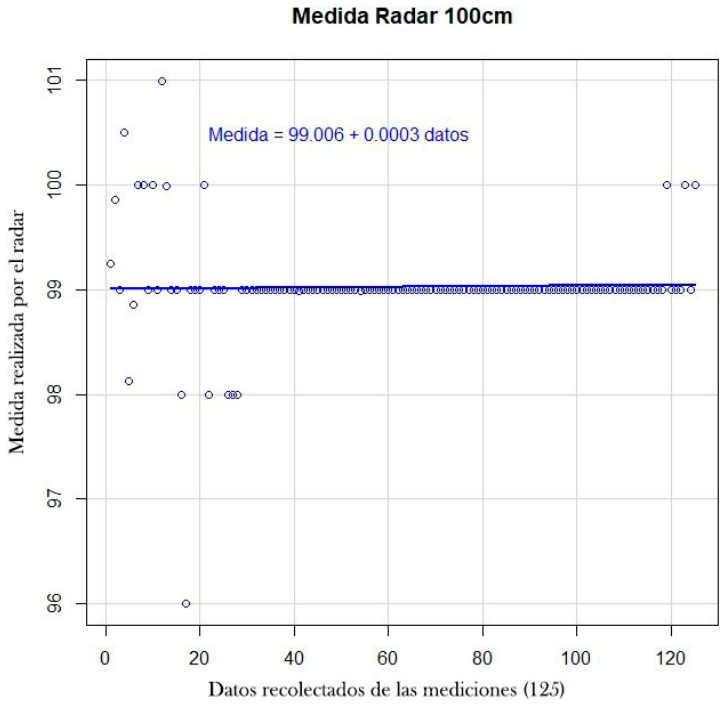


Figura 38: Ajuste de los datos a un modelo lineal.

Modelo de Intervalos de Confianza

	Estimate	2.5%	97.5%
Intercept	99.0061637267	98.831154538	99.181172915
Datos	0.0003486946	-0.002061845	0.002759235

Tabla 8: Datos del modelo de Intervalos de confianza, prueba de normalidad.

Con una prueba de Normalidad afirmativa para los datos del Radar. Se demuestra que los datos obtenidos para una distancia fijada de 100 cm (medida con la cinta métrica), están dentro del 99.18% de confianza. Dentro de un nivel del 97.5% de confiabilidad.

Prueba de Hipótesis

Ho: Los datos medidos con el Radar tienen un error de medida mayor o igual que del 5% ($\mu_1 \geq \mu_2$)

Ha: Los datos medidos con el Radar tienen un error de medida menor que del 5% ($\mu_1 < \mu_2$)

μ1 = Error de los datos medidos por el radar con respecto a la medición por cinta métrica.

μ2 = Rango de tolerancia del error, equivalente al 5%.

Test de varianza para una muestra

```
        One sample Chi-squared test for variance

data:  Medida.por.Radar[!is.na(Medida.por.Radar)]
X-squared = 29.707, df = 124, p-value < 2.2e-16
alternative hypothesis: true variance is not equal to 1
95 percent confidence interval:
 0.1895623 0.3124756
sample estimates:
var of Medida.por.Radar[!is.na(Medida.por.Radar)]
                                        0.2395733
```

Con un p-valor de p-value < 2.2e-16, la hipótesis alternativa determina que la varianza no es igual a 1, con un 95% de confiabilidad y se procede al test de comparación de medias.

Comparación de Medias entre la distancia fija 100 cm y las medidas dadas por el Radar

Aplicando un test t-student para muestras Independientes, se tiene

```
        Paired t-test

data:  Medida.por.Radar and Medida.en.cinta.metrica
t = -22.2, df = 124, p-value < 2.2e-16
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
       -Inf -0.8993167
sample estimates:
mean of the differences
             -0.9718685
```

Con un valor del estadístico t = -22.2, grados de libertad df = 124, y un p-valor de p-value < 2.2e-16, se rechaza la hipótesis nula y se acepta la hipótesis alternativa; esto es, los datos medidos con el radar tienen un error de medida menor que el 5% con respecto a la medida fijada de 100 cm, con un nivel de confianza del 95%.

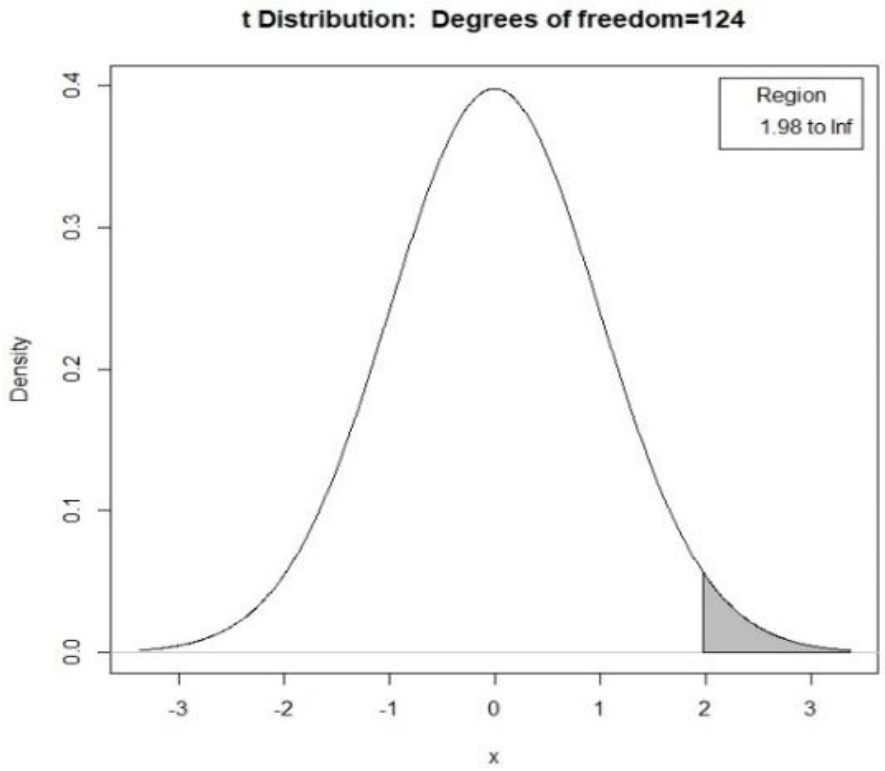


Figura 39: Distribución t de los valores medidos a 100cm.

Control de Calidad Medias Continuas Fase I Métrica 3 Sigma

Con los datos obtenidos por el Radar para una distancia fijada de 100 cm

```
List of 11
 $ call      : language qcc(data = cbind(Medida.en.cinta.metrica, Medida.por.Radar), type = "xbar", title = "xbar Chart for (Medida.en.ci| __truncated__
 $ type      : chr "xbar"
 $ data.name : chr "cbind(Medida.en.cinta.metrica, Medida.por.Radar)"
 $ data      : num [1:125, 1:2] 100 100 100 100 100 100 100 100 100 100 ...
  ..- attr(*, "dimnames")=List of 2
 $ statistics: Named num [1:125] 99.6 99.9 99.5 100.2 99.1 ...
  ..- attr(*, "names")= chr [1:125] "1" "2" "3" "4" ...
 $ sizes     : int [1:125] 2 2 2 2 2 2 2 2 2 2 ...
 $ center    : num 99.5
 $ std.dev   : num 0.883
 $ nsigmas   : num 3
 $ limits    : num [1, 1:2] 97.6 101.4
  ..- attr(*, "dimnames")=List of 2
 $ violations:List of 2
 - attr(*, "class")= chr "qcc"
```

Centrada al 99.5 y con una métrica de 3 sigma, se tiene una desviación estándar 0.883, lo que establece que para un 73% de las mediciones el Radar arroja medidas confiables al 99.5%. Se evidencia un mayor error en las medidas por el radar en algunas de las primeras y últimas mediciones (27% de las mediciones totales).

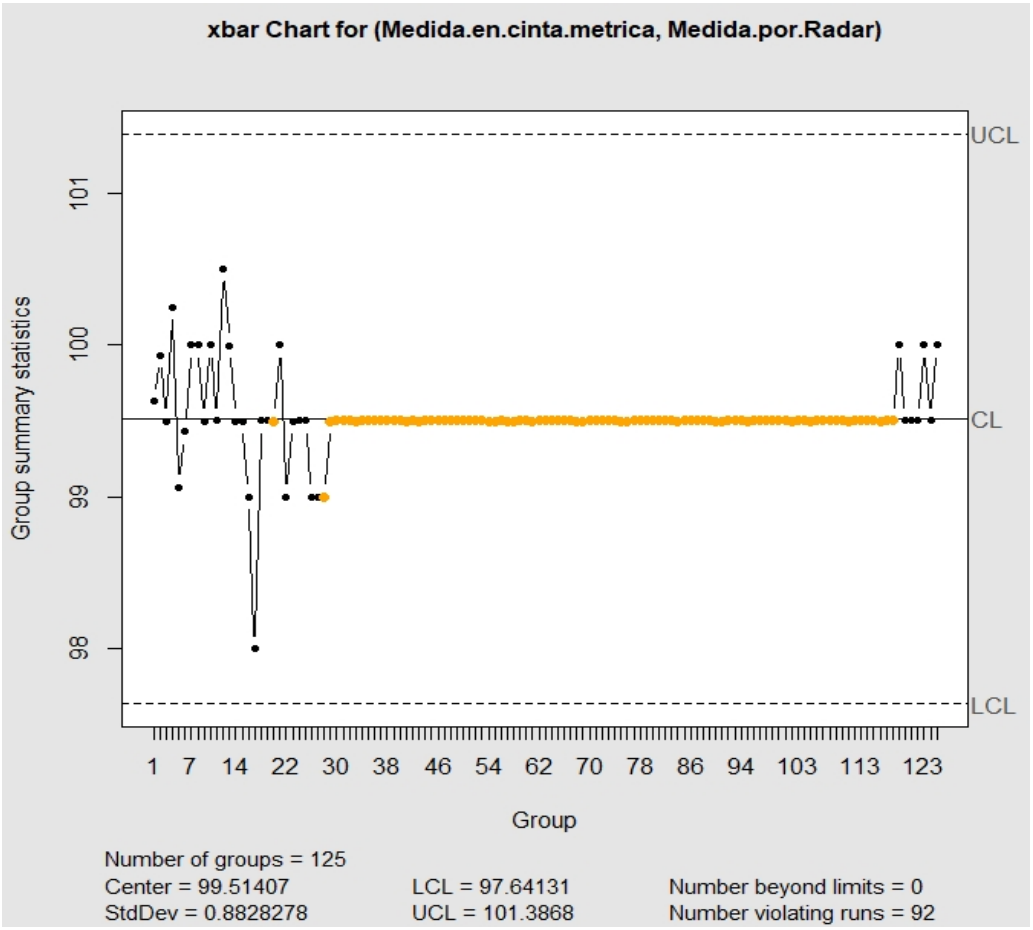


Figura 40: Resultados del control de calidad aplicado a las mediciones a 100cm.

Las medidas presentan anormalidades significativas en las primeras mediciones que se corrigieron con el cambio del soporte de las antenas. Una vez corregida la falla se puede observar una lectura estable con mínimas variaciones, con un error relativo inferior al 5%. Posteriormente se pueden observar unas anomalías en las mediciones finales causadas por interferencia humana durante la medición.

	Trayecto Ida				Trayecto Regreso			
Número de Cono	1	2	3	4	4	3	2	1
Distancia Teórica [m]	6	12	18	25	25	18	12	6
Tiempo Experimental [s]	43,9	50,1	55,8	62	67	73,03	78,06	82,9
Distancia Experimental [m]	6,12	11,86	17,76	24,71	24,67	18,11	12,93	5,92
% Error	2 %	1,1 %	2 %	1,16 %	1,32 %	0,6 %	7,75 %	1,33 %

Tabla 9: Tabla de medición piloto para el cálculo de la varianza. [5]

La tabla 7 expone los resultados de un trabajo de otra fuente, que desarrollo un radar MFCW. Se pudo observar un error de lectura variante, con valores entre 0,6% hasta 7.75%, igualmente se puede notar un incremento del error según aumente la distancia, haciendo una comparativa con la presente investigación se puedo notar un valor de error con mayores variaciones, pero también que posee un rango de medición sin ambigüedad mucho mayor de hasta 25m. Por lo que se puede concluir que ambos modelos funcionan técnicamente iguales con un error inferior al 5% en las lecturas de hasta 6 metros.

CAPÍTULO V. CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- Las mediciones realizadas por el radar onda continua con múltiples frecuencias (MFCW), ofrecen resultados con un error de medición muy bajo, inferiores al 5% y se ubican dentro de los intervalos de confianza al 95%

- La desviación estándar de las medidas indica que para un 73% de las mediciones el Radar arroja medidas confiables al 99.5%. Sin embargo, el error estándar del radar aumenta a medida que la distancia crece.

- El radar utiliza un ancho de banda relativamente bajo para el cálculo de distancias, y nos ofrece múltiples aplicaciones con un bajo consumo de energía. El radar ofrece muchas aplicaciones dentro de las cuales podríamos destacar, su uso en vehículos autónomos, para la detección temprana de intrusos en entidades financieras, detección de movimiento dentro de paredes que permitan la entrada de señales electromagnéticas (yeso, madera, etc.).

5.2 Recomendaciones

- Se debe establecer un espaciamiento físico adecuado entre la antena de transmisión y recepción de cada portadora para evitar la inducción de frecuencia directa.

- Para la obtención de mejores resultados se recomienda que la base donde se montan las antenas esté aislada de estas y que no afecte o modifique en el patrón de radiación de estas.

- Se recomienda aumentar el valor de frecuencia de actualización (update) en las gráficas de GNURadio para poder observar de mejor manera los valores medidos.

- Se recomienda para trabajos futuros la implementación de una sincronización física entre los SDR Adalm Pluto, para una mejor toma de distancias. Además, de el desarrollo de un control automático para eliminar la inducción directa de señal al receptor de los SDR.

BIBLIOGRÁFIA

[1] J. Cartagena and M. Silva, “Diseño e implementación de un prototipo de radar de objetivos móviles con radio definido por software y gnuradio,” Undergraduate Thesis, Universidad Politécnica Salesiana del Ecuador, 2021. Accessed: 2022. [Online]. Available: http://dspace.ups.edu.ec/handle/123456789/20961

[2] V. Cai, “Designing a narrowband radar using GNU radio and software defined radio for tomography and indoor sensing,” Proceedings of the GNU Radio Conference, vol. 5, no. 1, Art. no. 1, 2020, [Online]. Available: https://pubs.gnuradio.org/index.php/grcon/article/view/67

[3] C. Wolff, “Radar Basics,” Radartutorial. http://radartutorial.eu/ (accessed Dec. 08, 2021).

[4] J. Rugeles, M. Quintero, and L. Quibano, “Multiple frequency continuous wave SDR radar implementation using GNU Radio,” Ingeniería Y Desarrollo, vol. 38, no. 10, pp. 224–242, Jan. 2021, doi: 10.14482/inde.38.1.621.38.

[5] S. Aguilar-Barojas, “Fórmulas para el cálculo de la muestra en investigaciones de salud,” Salud en tabasco, vol. 11, no. 1–2, pp. 333–338, 2005, Accessed: Dec. 08, 2021. [Online]. Available: http://www.redalyc.org/articulo.oa?id=48711206

[6] T. Fepeussi, N. Testi, Y. Xu, and Y. Jin, “High-accuracy narrowband software-defined radar using successive multiple-frequency continuous-wave modulation for sensing applications,” IEEE Transactions on Microwave Theory and Techniques, vol. 67, no. 9, pp. 3917–3927, 2019, doi: 10.1109/TMTT.2019.2910056.

[7] Y.-K. Kwag, J.-S. Jung, I.-S. Woo, and M.-S. Park, “Modern software defined radar (SDR) technology and its trends,” Journal of electromagnetic engineering and science, vol. 14, no. 1, pp. 321–328, Dec. 2014, doi: 10.5515/JKIEES.2014.14.4.321.

[8] F. A. Farinha, “Implementação de um sistema de RADAR numa plataforma de rádio definido por software,” Undergraduate Thesis, Universidade De Coimbra, 2018. Accessed: Dec. 08, 2021. [Online]. Available: http://hdl.handle.net/10316/86775

[9] Q. Zhu and Y. Wang, “FMCW Radar Implemented with GNU Radio Companion,” MSc Thesis, Linnaeus University, 2016.

[10] J. Marimuthu, K. S. Bialkowski, and A. Abbosh, “Software-defined radar for medical imaging,” IEEE Transactions on Microwave Theory and Techniques, vol. 64, no. 2, pp. 643–652, 2016, doi: 10.1109/TMTT.2015.2511013.

[11] Y. Ma, Y. Zeng, and S. Sun, “A Software Defined Radio Based multi-function Radar for IoT Applications,” in Asia-Pacific Conference on Communications, 2018, vol. 24, no. 10, pp. 239–244. doi: 10.1109/APCC.2018.8633541.

[12] Z. Fang, L. Lou, C. Yang, K. Tang, and Y. Zheng, “A ku-band FMCW Radar on Chip for Wireless Micro Physiological Signal Monitoring by Interferometry Phase Analysis,” in IEEE International Symposium on Circuits and Systems, May 2018, pp. 1–4. doi: 10.1109/ISCAS.2018.8351226.

[13] J. Macacero, O. Gerasta, D. Pongcol, V. Ylaya, and A. Caberos, “Underground Target Objects Detection Simulation Using FMCW Radar with SDR Platform,” in IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology,Communication and Control, Environment and Management, 2018, pp. 1–7. doi: 10.1109/HNICEM.2018.8666248.

[14] S. Aulia, S. Andriyan, and A. Munir, “Stationary and moving targets detection on FMCW radar using GNU radio-based software defined radio,” in International Symposium on Intelligent Signal Processing and Communication Systems, 2015, pp. 468–473. doi: 10.1109/ISPACS.2015.7432817.

}

ANEXOS

ANEXO 1. Datos Técnicos del módulo de aprendizaje activo de radio definido por software “Adalm Pluto”

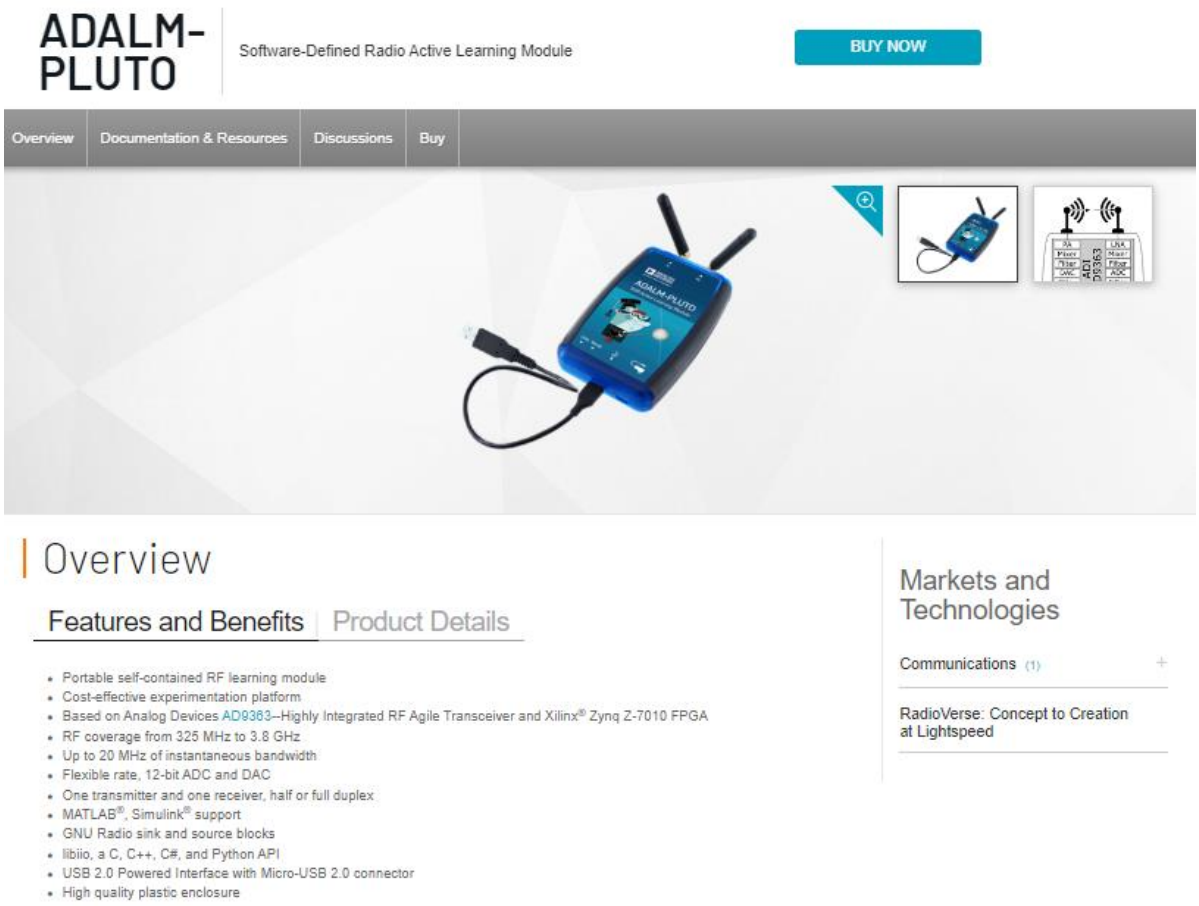


Figura 41: Detalles técnicos del módulo SDR Adalm Pluto.

ANEXO 2. Diseño en CTS de una antena parche a 2.4 GHz

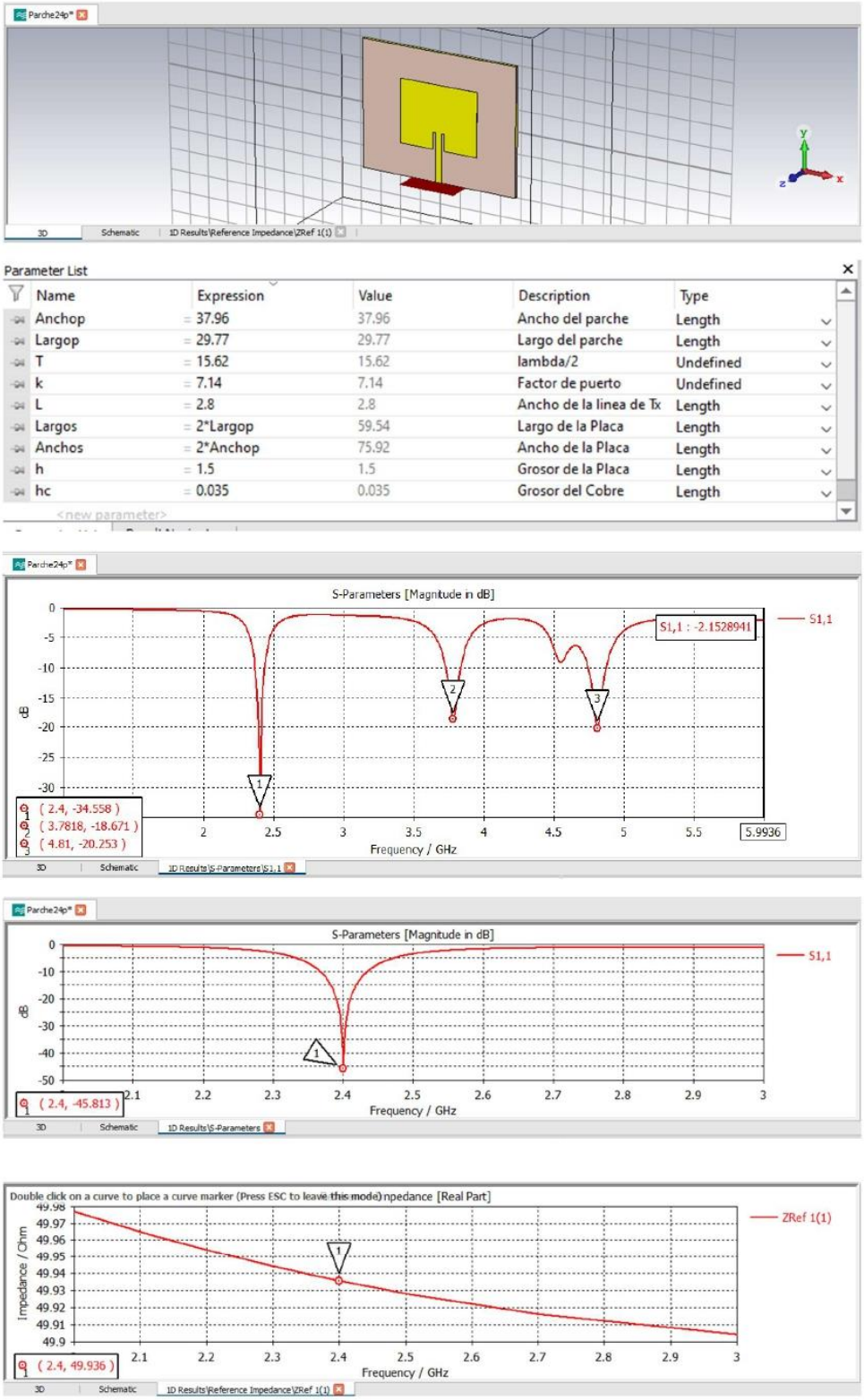


Figura 42: Parámetros físicos, parámetros s e impedancia de la antena de 2.4 GHz

ANEXO 3. Diseño en CTS de una antena parche a 2.45 GHz

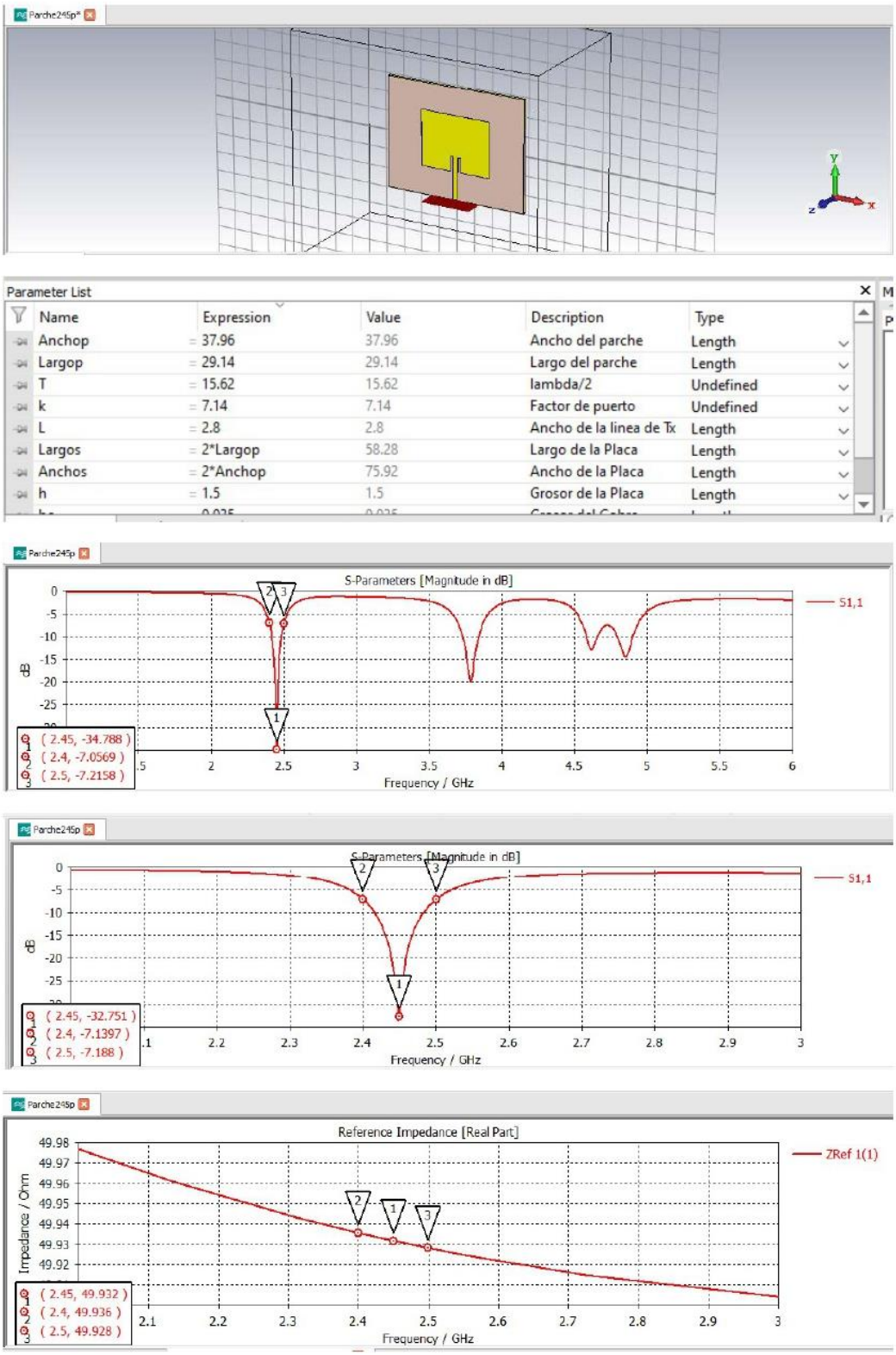
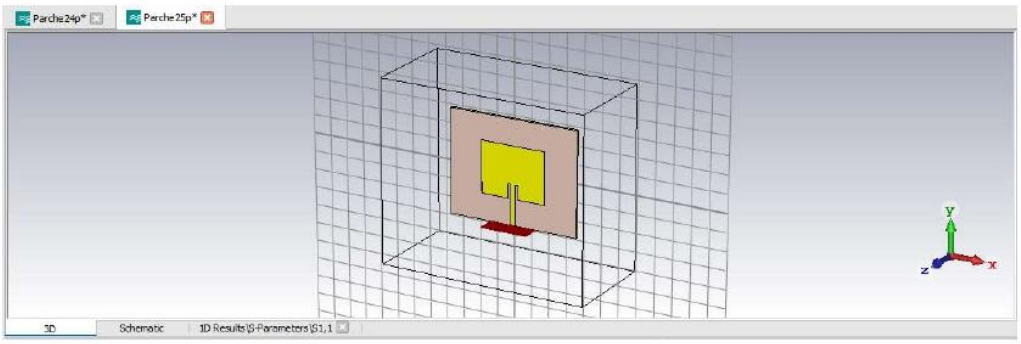


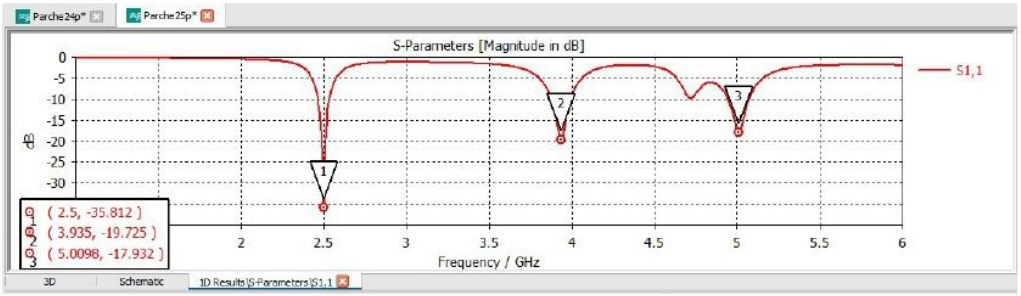
Figura 43: Parámetros físicos, parámetros s e impedancia de la antena de 2.45 GHz

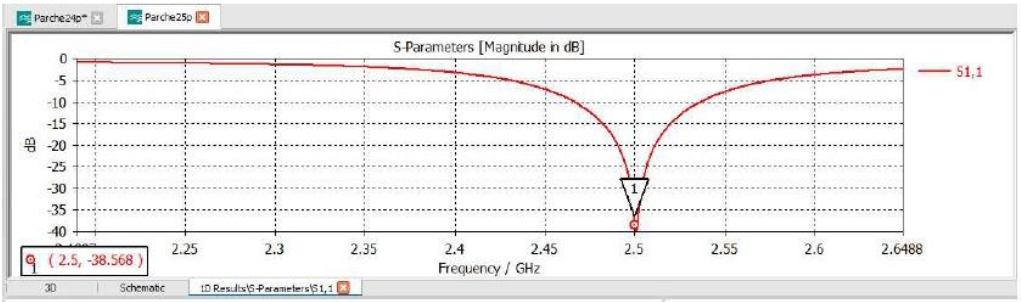
ANEXO 4. Diseño en CTS de una antena parche a 2.5 GHz



Parameter List

Name	Expression	Value	Description	Type
Anchop	36.44	36.44	Ancho del parche	Length
Largop	28.57	28.57	Largo del parche	Length
T	15.62	15.62	lambda/2	Undefined
k	7.14	7.14	Factor de puerto	Undefined
L	2.8	2.8	Ancho de la linea de Tx	Length
Largos	2*Largop	57.14	Largo de la Placa	Length
Anchos	2*Anchop	72.88	Ancho de la Placa	Length
h	1.5	1.5	Grosor de la Placa	Length
hc	0.035	0.035	Grosor del Cobre	Length





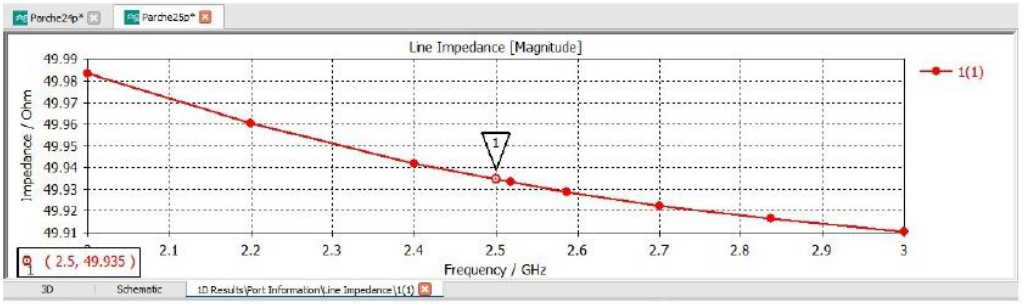


Figura 44: : Parámetros físicos, parámetros s e impedancia de la antena de 2.5 GHz

ANEXO 5. Pruebas en el laboratorio

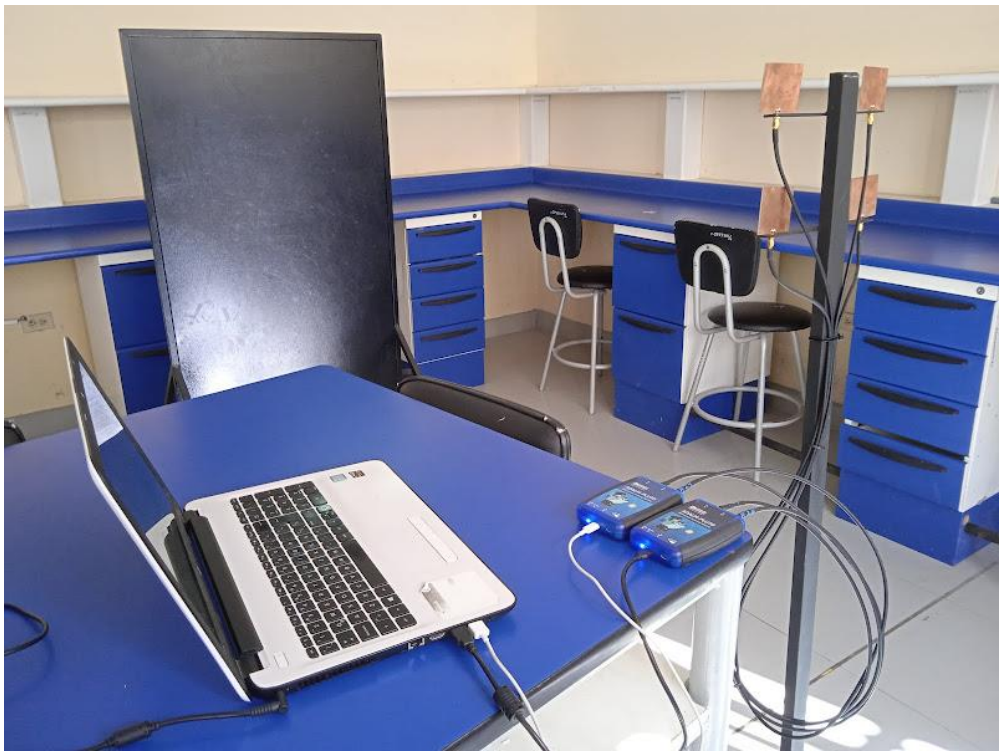


Figura 45: Mediciones iniciales en el laboratorio de electrónica de la Universidad Nacional de Chimborazo (2022).

Figura 46: Mediciones iniciales en el laboratorio de electrónica de la Universidad Nacional de Chimborazo (2022).

ANEXO 6. Pruebas a diferentes distancias

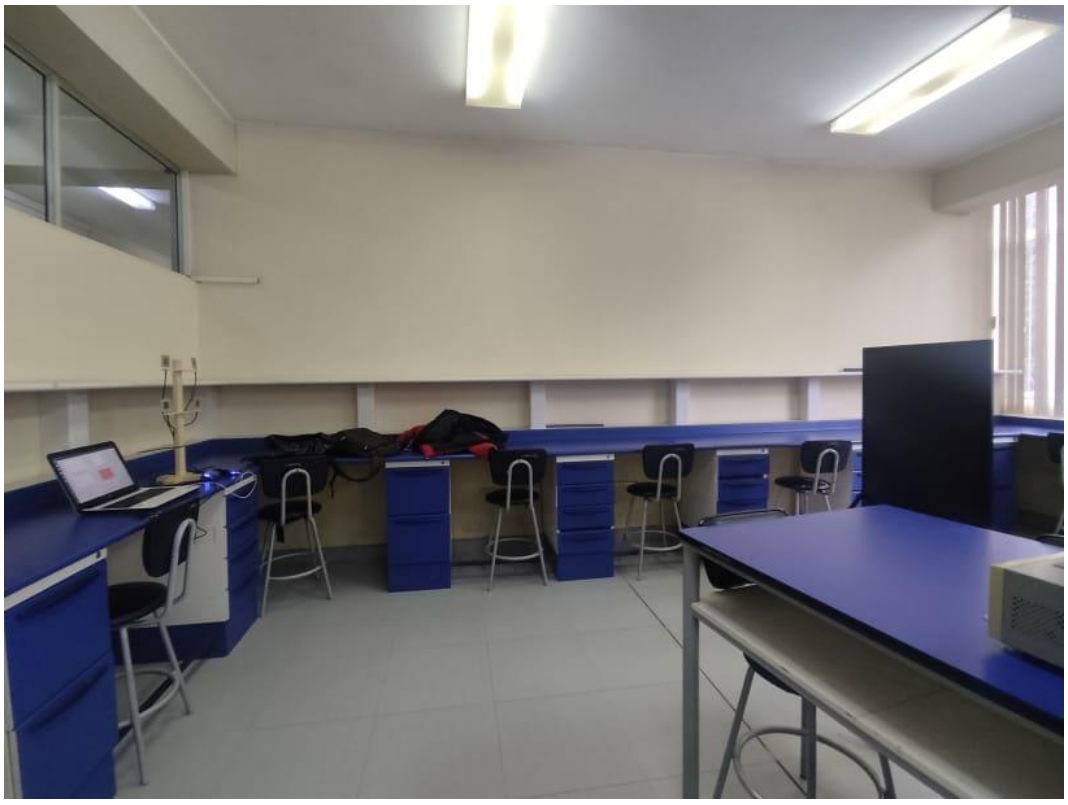


Figura 47: Pruebas de operación a diferentes distancias.

Figura 48: Pruebas de operación a diferentes distancias.

Figura 49: Pruebas de operación a diferentes distancias.

Figura 50: Pruebas de operación a diferentes distancias.

ANEXO 7. Tablas de mediciones al azar

Medicion diaria de distancia usando el dispositivo Radar MFCW — Delimitacion a ambigüedad 600cm

	Frecuencia 1 (GHz)	Frecuencia 2 (GHz)	Diferencia de frecuencia (GHz)	Distancia Maxima sin ambigüedad (cm)	Medida con flexometro (cm)	Medicion por Radar (cm)	Diferencia en la medicion(cm)	Observacion
1	2,475	2,5	0,025	600	100	98,998866	1,001134	
2	2,475	2,5	0,025	600	100	98,998772	1,001228	
3	2,475	2,5	0,025	600	100	98,999866	1,000134	
4	2,475	2,5	0,025	600	100	98,999784	1,000216	
5	2,475	2,5	0,025	600	100	98,999892	1,000108	
6	2,475	2,5	0,025	600	100	98,999474	1,000526	
7	2,475	2,5	0,025	600	100	98,99947	1,00053	
8	2,475	2,5	0,025	600	100	98,999892	1,000108	
9	2,475	2,5	0,025	600	100	98,999477	1,000523	
10	2,475	2,5	0,025	600	100	98,999866	1,000134	
11	2,475	2,5	0,025	600	200	198,999954	1,000046	
12	2,475	2,5	0,025	600	200	198,999974	1,000026	
13	2,475	2,5	0,025	600	200	198,999992	1,000008	
14	2,475	2,5	0,025	600	200	199,00098	0,99902	
15	2,475	2,5	0,025	600	200	198,998917	1,001083	
16	2,475	2,5	0,025	600	200	198,999974	1,000026	
17	2,475	2,5	0,025	600	200	199,000825	0,999175	
18	2,475	2,5	0,025	600	200	198,9999	1,0001	
19	2,475	2,5	0,025	600	200	198,999794	1,000206	
20	2,475	2,5	0,025	600	200	198,999802	1,000198	
21	2,475	2,5	0,025	600	300	299,000824	0,999176	
22	2,475	2,5	0,025	600	300	299,000244	0,999756	
23	2,475	2,5	0,025	600	300	299,000275	0,999725	
24	2,475	2,5	0,025	600	300	299,00765	0,99235	
25	2,475	2,5	0,025	600	300	298,997375	1,002625	
26	2,475	2,5	0,025	600	300	298,997345	1,002655	
27	2,475	2,5	0,025	600	300	298,96794	1,03206	
28	2,475	2,5	0,025	600	300	298,997864	1,002136	
29	2,475	2,5	0,025	600	300	298,967879	1,032121	
30	2,475	2,5	0,025	600	300	298,999954	1,000046	
31	2,475	2,5	0,025	600	400	400,00824	0,00824	
32	2,475	2,5	0,025	600	400	399,999039	0,000961	
33	2,475	2,5	0,025	600	400	399,9991	0,0009	
34	2,475	2,5	0,025	600	400	400,00916	0,00916	
35	2,475	2,5	0,025	600	400	399,999077	0,000923	
36	2,475	2,5	0,025	600	400	399,999107	0,000893	
37	2,475	2,5	0,025	600	400	400,00668	0,00668	
38	2,475	2,5	0,025	600	400	399,999779	0,000221	
39	2,475	2,5	0,025	600	400	399,998917	0,001083	
40	2,475	2,5	0,025	600	400	399,99817	0,00183	

Tabla 10: Tabla de mediciones del radar en un día(azar).

4

ANEXO 8. Cálculo del error relativo de una tabla de mediciones al azar.

	Medicion diaria de distancia usando el dispositivo Radar MFCW						98,96085838	199,0473935	299,1549287	399,8203947
	Frecuencia 1 (GHz)	Frecuencia 2 (GHz)	Diferencia de frecuencia (GHz)	Distancia Maxima sin ambigüedad (cm)	Medida con flexometro (cm)	Medicion por Radar (cm)	Diferencia en la medicion(cm)	Error absoluto	Error relativo	Error relativo porcentual
1	2,475	2,5	0,025	600	100	98,999892	1,000108	0,03903362	0,010106097	1,01%
2	2,475	2,5	0,025	600	100	98,990072	1,009928	0,02921362	0,010205328	1,02%
3	2,475	2,5	0,025	600	100	98,998847	1,001153	0,03798862	0,010116656	1,01%
4	2,475	2,5	0,025	600	100	98,999477	1,000523	0,03861862	0,01011029	1,01%
5	2,475	2,5	0,025	600	100	98,999474	1,000526	0,03861562	0,010110321	1,01%
6	2,475	2,5	0,025	600	100	98,990054	1,009946	0,02919562	0,01020551	1,02%
7	2,475	2,5	0,025	600	100	98,999072	1,000928	0,03821362	0,010114383	1,01%
8	2,475	2,5	0,025	600	100	98,998882	1,001118	0,03802362	0,010116303	1,01%
9	2,475	2,5	0,025	600	100	98,999892	1,000108	0,03903362	0,010106097	1,01%
10	2,475	2,5	0,025	600	100	99,000057	0,999943	0,03919862	0,010104429	1,01%
11	2,475	2,5	0,025	600	200	199,000054	0,999946	0,0473395	0,005023658	0,50%
12	2,475	2,5	0,025	600	200	199,000747	0,999253	0,0466465	0,005020176	0,50%
13	2,475	2,5	0,025	600	200	198,999854	1,000146	0,0475395	0,005024663	0,50%
14	2,475	2,5	0,025	600	200	198,99982	1,00018	0,0475735	0,005024833	0,50%
15	2,475	2,5	0,025	600	200	198,998192	1,001808	0,0492015	0,005033012	0,50%
16	2,475	2,5	0,025	600	200	198,999879	1,000121	0,0475145	0,005024537	0,50%
17	2,475	2,5	0,025	600	200	198,997864	1,002136	0,0495295	0,00503466	0,50%
18	2,475	2,5	0,025	600	200	198,998291	1,001709	0,0491025	0,005032515	0,50%
19	2,475	2,5	0,025	600	200	198,998077	1,001923	0,0493165	0,00503359	0,50%
20	2,475	2,5	0,025	600	200	199,000854	0,999146	0,0465395	0,005019639	0,50%
21	2,475	2,5	0,025	600	300	299,000443	0,999557	0,1544857	0,003341269	0,33%
22	2,475	2,5	0,025	600	300	299,000824	0,999176	0,1541047	0,003339995	0,33%
23	2,475	2,5	0,025	600	300	298,997854	1,002146	0,1570747	0,003349923	0,33%
24	2,475	2,5	0,025	600	300	298,99992	1,00008	0,1550087	0,003343017	0,33%
25	2,475	2,5	0,025	600	300	299,14143	0,85857	0,0134987	0,002869984	0,29%
26	2,475	2,5	0,025	600	300	299,74625	0,25375	0,5913213	0,000848223	0,08%
27	2,475	2,5	0,025	600	300	299,000824	0,999176	0,1541047	0,003339995	0,33%
28	2,475	2,5	0,025	600	300	298,999918	1,000082	0,1550107	0,003343024	0,33%
29	2,475	2,5	0,025	600	300	298,998291	1,001709	0,1566377	0,003348462	0,33%
30	2,475	2,5	0,025	600	300	298,999866	1,000134	0,1550627	0,003343197	0,33%
31	2,475	2,5	0,025	600	400	400,00824	0,00824	0,1878453	2,06093E-05	0,00%
32	2,475	2,5	0,025	600	400	399,000443	0,999557	0,8199517	0,002500015	0,25%
33	2,475	2,5	0,025	600	400	399,000824	0,999176	0,8195707	0,002499062	0,25%
34	2,475	2,5	0,025	600	400	399,000824	0,999176	0,8195707	0,002499062	0,25%
35	2,475	2,5	0,025	600	400	399,000793	0,999207	0,8196017	0,00249914	0,25%
36	2,475	2,5	0,025	600	400	398,999352	1,000648	0,8210427	0,002502744	0,25%
37	2,475	2,5	0,025	600	400	398,99947	1,00053	0,8209247	0,002502449	0,25%
38	2,475	2,5	0,025	600	400	398,998825	1,001175	0,8215697	0,002504062	0,25%
39	2,475	2,5	0,025	600	400	398,99894	1,00106	0,8214547	0,002503774	0,25%
40	2,475	2,5	0,025	600	400	398,999844	1,000156	0,8205507	0,002501513	0,25%

Tabla 11: Calculo de error absoluto y relativo en Excel.

ANEXO 9. Código del proyecto en Python

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
##################################################
# GNU Radio Python Flow Graph
# Title: Top Block
# Generated: Tue Nov 22 09:34:54 2022
##################################################

from distutils.version import StrictVersion

if __name__ == '__main__':
    import ctypes
    import sys
    if sys.platform.startswith('linux'):
        try:
            x11 = ctypes.cdll.LoadLibrary('libX11.so')
            x11.XInitThreads()
        except:
            print "Warning: failed to XInitThreads()"

from PyQt5 import Qt
from PyQt5 import Qt, QtCore
from gnuradio import analog
from gnuradio import blocks
from gnuradio import eng_notation
from gnuradio import filter
from gnuradio import gr
from gnuradio import iio
from gnuradio import qtgui
from gnuradio.eng_option import eng_option
from gnuradio.filter import firdes
from gnuradio.qtgui import Range, RangeWidget
from optparse import OptionParser
import sip
import sys
from gnuradio import qtgui


class top_block(gr.top_block, Qt.QWidget):
```

```
def __init__(self):
    gr.top_block.__init__(self, "Top Block")
    Qt.QWidget.__init__(self)
    self.setWindowTitle("Top Block")
    qtgui.util.check_set_qss()
    try:
        self.setWindowIcon(Qt.QIcon.fromTheme('gnuradio-grc'))
    except:
        pass
    self.top_scroll_layout = Qt.QVBoxLayout()
    self.setLayout(self.top_scroll_layout)
    self.top_scroll = Qt.QScrollArea()
    self.top_scroll.setFrameStyle(Qt.QFrame.NoFrame)
    self.top_scroll_layout.addWidget(self.top_scroll)
    self.top_scroll.setWidgetResizable(True)
    self.top_widget = Qt.QWidget()
    self.top_scroll.setWidget(self.top_widget)
    self.top_layout = Qt.QVBoxLayout(self.top_widget)
    self.top_grid_layout = Qt.QGridLayout()
    self.top_layout.addLayout(self.top_grid_layout)

    self.settings = Qt.QSettings("GNU Radio", "top_block")

    if StrictVersion(Qt.qVersion()) < StrictVersion("5.0.0"):
        self.restoreGeometry(self.settings.value("geometry").toByteArray())
    else:
        self.restoreGeometry(self.settings.value("geometry", type=QtCore.QByteArray))

    ##################################################
    # Variables
    ##################################################
    self.freq = freq = 5210
    self.samp_rate = samp_rate = freq*100
    self.puntos = puntos = 100*4
    self.memoria = memoria = 0x8000
    self.gain25 = gain25 = 0
    self.gain24 = gain24 = 0
    self.freq_25 = freq_25 = 2500000000
    self.freq_24 = freq_24 = 2475000000
    self.comp25 = comp25 = 0
    self.comp24 = comp24 = 0
    self.Update = Update = 0.3
    self.Nivel = Nivel = -72
```

```
        self.Calibrar_25 = Calibrar_25 = 0
        self.Calibrar_24 = Calibrar_24 = 0

        ##################################################
        # Blocks
        ##################################################
        self._gain25_range = Range(0, 0.750, 0.01, 0, 200)
        self._gain25_win = RangeWidget(self._gain25_range, self.set_gain25, "gain25",
"counter_slider", float)
        self.top_grid_layout.addWidget(self._gain25_win, 5, 3, 1, 1)
        [self.top_grid_layout.setRowStretch(r,1) for r in range(5,6)]
        [self.top_grid_layout.setColumnStretch(c,1) for c in range(3,4)]
        self._gain24_range = Range(0, 0.750, 0.01, 0, 200)
        self._gain24_win = RangeWidget(self._gain24_range, self.set_gain24, "gain24",
"counter_slider", float)
        self.top_grid_layout.addWidget(self._gain24_win, 4, 3, 1, 1)
        [self.top_grid_layout.setRowStretch(r,1) for r in range(4,5)]
        [self.top_grid_layout.setColumnStretch(c,1) for c in range(3,4)]
        self._comp25_range = Range(-100, 100, 1, 0, 200)
        self._comp25_win = RangeWidget(self._comp25_range, self.set_comp25,
"comp25", "counter_slider", int)
        self.top_grid_layout.addWidget(self._comp25_win, 4, 2, 1, 1)
        [self.top_grid_layout.setRowStretch(r,1) for r in range(4,5)]
        [self.top_grid_layout.setColumnStretch(c,1) for c in range(2,3)]
        self._comp24_range = Range(-100, 100, 1, 0, 200)
        self._comp24_win = RangeWidget(self._comp24_range, self.set_comp24,
"comp24", "counter_slider", int)
        self.top_grid_layout.addWidget(self._comp24_win, 5, 2, 1, 1)
        [self.top_grid_layout.setRowStretch(r,1) for r in range(5,6)]
        [self.top_grid_layout.setColumnStretch(c,1) for c in range(2,3)]
        self._Update_range = Range(0.1, 0.5, 0.01, 0.3, 200)
        self._Update_win = RangeWidget(self._Update_range, self.set_Update, "Update",
"counter_slider", float)
        self.top_layout.addWidget(self._Update_win)
        self._Nivel_range = Range(-180, -30, 1, -72, 200)
        self._Nivel_win = RangeWidget(self._Nivel_range, self.set_Nivel, "Nivel",
"counter_slider", float)
        self.top_grid_layout.addWidget(self._Nivel_win, 3, 3, 1, 1)
        [self.top_grid_layout.setRowStretch(r,1) for r in range(3,4)]
        [self.top_grid_layout.setColumnStretch(c,1) for c in range(3,4)]
        self._Calibrar_25_range = Range(-100, 100, 1, 0, 200)
        self._Calibrar_25_win = RangeWidget(self._Calibrar_25_range,
self.set_Calibrar_25, "Calibrar_25", "counter_slider", int)
```

```
        self.top_grid_layout.addWidget(self._Calibrar_25_win, 4, 0, 1, 2)
        [self.top_grid_layout.setRowStretch(r,1) for r in range(4,5)]
        [self.top_grid_layout.setColumnStretch(c,1) for c in range(0,2)]
        self._Calibrar_24_range = Range(-100, 100, 1, 0, 200)
        self._Calibrar_24_win = RangeWidget(self._Calibrar_24_range,
self.set_Calibrar_24, "Calibrar_24", "counter_slider", int)
        self.top_grid_layout.addWidget(self._Calibrar_24_win, 5, 0, 1, 2)
        [self.top_grid_layout.setRowStretch(r,1) for r in range(5,6)]
        [self.top_grid_layout.setColumnStretch(c,1) for c in range(0,2)]
        self.qtgui_waterfall_sink_x_0 = qtgui.waterfall_sink_c(
                512, #size
                firdes.WIN_BLACKMAN_hARRIS, #wintype
                0, #fc
                2035, #bw
                "", #name
               1 #number of inputs
        )
        self.qtgui_waterfall_sink_x_0.set_update_time(0.10)
        self.qtgui_waterfall_sink_x_0.enable_grid(False)
        self.qtgui_waterfall_sink_x_0.enable_axis_labels(True)

        if not True:
          self.qtgui_waterfall_sink_x_0.disable_legend()

        if "complex" == "float" or "complex" == "msg_float":
          self.qtgui_waterfall_sink_x_0.set_plot_pos_half(not True)

        labels = ['', '', '', '', '',
                  '', '', '', '', '']
        colors = [0, 0, 0, 0, 0,
                  0, 0, 0, 0, 0]
        alphas = [1.0, 1.0, 1.0, 1.0, 1.0,
                  1.0, 1.0, 1.0, 1.0, 1.0]
        for i in xrange(1):
           if len(labels[i]) == 0:
              self.qtgui_waterfall_sink_x_0.set_line_label(i, "Data {0}".format(i))
           else:
              self.qtgui_waterfall_sink_x_0.set_line_label(i, labels[i])
           self.qtgui_waterfall_sink_x_0.set_color_map(i, colors[i])
           self.qtgui_waterfall_sink_x_0.set_line_alpha(i, alphas[i])

        self.qtgui_waterfall_sink_x_0.set_intensity_range(Nivel, 10)
```

```
        self._qtgui_waterfall_sink_x_0_win =
sip.wrapinstance(self.qtgui_waterfall_sink_x_0.pyqwidget(), Qt.QWidget)
        self.top_grid_layout.addWidget(self._qtgui_waterfall_sink_x_0_win, 2, 2, 2, 1)
        [self.top_grid_layout.setRowStretch(r,1) for r in range(2,4)]
        [self.top_grid_layout.setColumnStretch(c,1) for c in range(2,3)]
        self.qtgui_time_sink_x_0_0 = qtgui.time_sink_f(
                puntos, #size
                samp_rate, #samp_rate
                "", #name
                3 #number of inputs
        )
        self.qtgui_time_sink_x_0_0.set_update_time(Update)
        self.qtgui_time_sink_x_0_0.set_y_axis(-1, 1)

        self.qtgui_time_sink_x_0_0.set_y_label('Amplitude (V)', "")

        self.qtgui_time_sink_x_0_0.enable_tags(-1, False)
        self.qtgui_time_sink_x_0_0.set_trigger_mode(qtgui.TRIG_MODE_FREE,
qtgui.TRIG_SLOPE_POS, 0.0, 0, 0, "")
        self.qtgui_time_sink_x_0_0.enable_autoscale(False)
        self.qtgui_time_sink_x_0_0.enable_grid(False)
        self.qtgui_time_sink_x_0_0.enable_axis_labels(True)
        self.qtgui_time_sink_x_0_0.enable_control_panel(False)
        self.qtgui_time_sink_x_0_0.enable_stem_plot(False)

        if not True:
          self.qtgui_time_sink_x_0_0.disable_legend()

        labels = ['', '', '', '', '',
                '', '', '', '', '']
        widths = [1, 1, 1, 1, 1,
                1, 1, 1, 1, 1]
        colors = ["blue", "red", "green", "black", "cyan",
                "magenta", "yellow", "dark red", "dark green", "blue"]
        styles = [1, 1, 1, 1, 1,
                1, 1, 1, 1, 1]
        markers = [-1, -1, -1, -1, -1,
                -1, -1, -1, -1, -1]
        alphas = [1.0, 1.0, 1.0, 1.0, 1.0,
                1.0, 1.0, 1.0, 1.0, 1.0]

        for i in xrange(3):
            if len(labels[i]) == 0:
```

```
            self.qtgui_time_sink_x_0_0.set_line_label(i, "Data {0}".format(i))
        else:
            self.qtgui_time_sink_x_0_0.set_line_label(i, labels[i])
        self.qtgui_time_sink_x_0_0.set_line_width(i, widths[i])
        self.qtgui_time_sink_x_0_0.set_line_color(i, colors[i])
        self.qtgui_time_sink_x_0_0.set_line_style(i, styles[i])
        self.qtgui_time_sink_x_0_0.set_line_marker(i, markers[i])
        self.qtgui_time_sink_x_0_0.set_line_alpha(i, alphas[i])

    self._qtgui_time_sink_x_0_0_win =
sip.wrapinstance(self.qtgui_time_sink_x_0_0.pyqwidget(), Qt.QWidget)
    self.top_grid_layout.addWidget(self._qtgui_time_sink_x_0_0_win, 2, 0, 2, 2)
    [self.top_grid_layout.setRowStretch(r,1) for r in range(2,4)]
    [self.top_grid_layout.setColumnStretch(c,1) for c in range(0,2)]
    self.qtgui_time_sink_x_0 = qtgui.time_sink_f(
            puntos, #size
            samp_rate, #samp_rate
            "", #name
            3 #number of inputs
    )
    self.qtgui_time_sink_x_0.set_update_time(Update)
    self.qtgui_time_sink_x_0.set_y_axis(-1, 1)

    self.qtgui_time_sink_x_0.set_y_label('Amplitude (V)', "")

    self.qtgui_time_sink_x_0.enable_tags(-1, False)
    self.qtgui_time_sink_x_0.set_trigger_mode(qtgui.TRIG_MODE_FREE,
qtgui.TRIG_SLOPE_POS, 0.0, 0, 0, "")
    self.qtgui_time_sink_x_0.enable_autoscale(False)
    self.qtgui_time_sink_x_0.enable_grid(False)
    self.qtgui_time_sink_x_0.enable_axis_labels(True)
    self.qtgui_time_sink_x_0.enable_control_panel(False)
    self.qtgui_time_sink_x_0.enable_stem_plot(False)

    if not True:
      self.qtgui_time_sink_x_0.disable_legend()

    labels = ['', '', '', '', '',
            '', '', '', '', '']
    widths = [1, 1, 1, 1, 1,
            1, 1, 1, 1, 1]
    colors = ["blue", "red", "green", "black", "cyan",
            "magenta", "yellow", "dark red", "dark green", "blue"]
```

```
        styles = [1, 1, 1, 1, 1,
                1, 1, 1, 1, 1]
        markers = [-1, -1, -1, -1, -1,
                 -1, -1, -1, -1, -1]
        alphas = [1.0, 1.0, 1.0, 1.0, 1.0,
                1.0, 1.0, 1.0, 1.0, 1.0]

        for i in xrange(3):
           if len(labels[i]) == 0:
              self.qtgui_time_sink_x_0.set_line_label(i, "Data {0}".format(i))
           else:
              self.qtgui_time_sink_x_0.set_line_label(i, labels[i])
           self.qtgui_time_sink_x_0.set_line_width(i, widths[i])
           self.qtgui_time_sink_x_0.set_line_color(i, colors[i])
           self.qtgui_time_sink_x_0.set_line_style(i, styles[i])
           self.qtgui_time_sink_x_0.set_line_marker(i, markers[i])
           self.qtgui_time_sink_x_0.set_line_alpha(i, alphas[i])

        self._qtgui_time_sink_x_0_win =
sip.wrapinstance(self.qtgui_time_sink_x_0.pyqwidget(), Qt.QWidget)
        self.top_grid_layout.addWidget(self._qtgui_time_sink_x_0_win, 0, 0, 2, 2)
        [self.top_grid_layout.setRowStretch(r,1) for r in range(0,2)]
        [self.top_grid_layout.setColumnStretch(c,1) for c in range(0,2)]
        self.qtgui_number_sink_0_0_0 = qtgui.number_sink(
           gr.sizeof_float,
           0,
           qtgui.NUM_GRAPH_HORIZ,
           1
        )
        self.qtgui_number_sink_0_0_0.set_update_time(Update)
        self.qtgui_number_sink_0_0_0.set_title('Velocidad(m/s)')

        labels = ['', '', '', '', '',
                 '', '', '', '', '']
        units = ['', '', '', '', '',
                '', '', '', '', '']
        colors = [("black", "black"), ("black", "black"), ("black", "black"), ("black",
"black"), ("black", "black"),
                 ("black", "black"), ("black", "black"), ("black", "black"), ("black", "black"),
("black", "black")]
        factor = [1, 1, 1, 1, 1,
                1, 1, 1, 1, 1]
        for i in xrange(1):
```

```
            self.qtgui_number_sink_0_0_0.set_min(i, -0.5)
            self.qtgui_number_sink_0_0_0.set_max(i, 600)
            self.qtgui_number_sink_0_0_0.set_color(i, colors[i][0], colors[i][1])
            if len(labels[i]) == 0:
                self.qtgui_number_sink_0_0_0.set_label(i, "Data {0}".format(i))
            else:
                self.qtgui_number_sink_0_0_0.set_label(i, labels[i])
            self.qtgui_number_sink_0_0_0.set_unit(i, units[i])
            self.qtgui_number_sink_0_0_0.set_factor(i, factor[i])

        self.qtgui_number_sink_0_0_0.enable_autoscale(False)
        self._qtgui_number_sink_0_0_0_win =
sip.wrapinstance(self.qtgui_number_sink_0_0_0.pyqwidget(), Qt.QWidget)
        self.top_grid_layout.addWidget(self._qtgui_number_sink_0_0_0_win, 2, 3, 1, 1)
        [self.top_grid_layout.setRowStretch(r,1) for r in range(2,3)]
        [self.top_grid_layout.setColumnStretch(c,1) for c in range(3,4)]
        self.qtgui_number_sink_0_0 = qtgui.number_sink(
            gr.sizeof_float,
            0,
            qtgui.NUM_GRAPH_HORIZ,
            1
        )
        self.qtgui_number_sink_0_0.set_update_time(Update)
        self.qtgui_number_sink_0_0.set_title('Distancia(cm)')

        labels = ['', '', '', '', '',
                  '', '', '', '', '']
        units = ['', '', '', '', '',
                 '', '', '', '', '']
        colors = [("black", "black"), ("black", "black"), ("black", "black"), ("black",
"black"), ("black", "black"),
                  ("black", "black"), ("black", "black"), ("black", "black"), ("black", "black"),
("black", "black")]
        factor = [1, 1, 1, 1, 1,
                  1, 1, 1, 1, 1]
        for i in xrange(1):
            self.qtgui_number_sink_0_0.set_min(i, -0.5)
            self.qtgui_number_sink_0_0.set_max(i, 600)
            self.qtgui_number_sink_0_0.set_color(i, colors[i][0], colors[i][1])
            if len(labels[i]) == 0:
                self.qtgui_number_sink_0_0.set_label(i, "Data {0}".format(i))
            else:
                self.qtgui_number_sink_0_0.set_label(i, labels[i])
```

```
            self.qtgui_number_sink_0_0.set_unit(i, units[i])
            self.qtgui_number_sink_0_0.set_factor(i, factor[i])

        self.qtgui_number_sink_0_0.enable_autoscale(False)
        self._qtgui_number_sink_0_0_win =
sip.wrapinstance(self.qtgui_number_sink_0_0.pyqwidget(), Qt.QWidget)
        self.top_grid_layout.addWidget(self._qtgui_number_sink_0_0_win, 1, 3, 1, 1)
        [self.top_grid_layout.setRowStretch(r,1) for r in range(1,2)]
        [self.top_grid_layout.setColumnStretch(c,1) for c in range(3,4)]
        self.qtgui_number_sink_0 = qtgui.number_sink(
            gr.sizeof_float,
            0,
            qtgui.NUM_GRAPH_HORIZ,
            1
        )
        self.qtgui_number_sink_0.set_update_time(Update)
        self.qtgui_number_sink_0.set_title("Fase24-25")

        labels = ['', '', '', '', '',
                  '', '', '', '', '']
        units = ['', '', '', '', '',
                 '', '', '', '', '']
        colors = [("black", "black"), ("black", "black"), ("black", "black"), ("black",
"black"), ("black", "black"),
                  ("black", "black"), ("black", "black"), ("black", "black"), ("black", "black"),
("black", "black")]
        factor = [1, 1, 1, 1, 1,
                  1, 1, 1, 1, 1]
        for i in xrange(1):
            self.qtgui_number_sink_0.set_min(i, 0)
            self.qtgui_number_sink_0.set_max(i, 2*3.1416)
            self.qtgui_number_sink_0.set_color(i, colors[i][0], colors[i][1])
            if len(labels[i]) == 0:
                self.qtgui_number_sink_0.set_label(i, "Data {0}".format(i))
            else:
                self.qtgui_number_sink_0.set_label(i, labels[i])
            self.qtgui_number_sink_0.set_unit(i, units[i])
            self.qtgui_number_sink_0.set_factor(i, factor[i])

        self.qtgui_number_sink_0.enable_autoscale(False)
        self._qtgui_number_sink_0_win =
sip.wrapinstance(self.qtgui_number_sink_0.pyqwidget(), Qt.QWidget)
        self.top_grid_layout.addWidget(self._qtgui_number_sink_0_win, 0, 3, 1, 1)
```

```
[self.top_grid_layout.setRowStretch(r,1) for r in range(0,1)]
[self.top_grid_layout.setColumnStretch(c,1) for c in range(3,4)]
self.qtgui_histogram_sink_x_0 = qtgui.histogram_sink_f(
          1000,
          100,
        -10,
        600,
          "",
          1
)

self.qtgui_histogram_sink_x_0.set_update_time(Update)
self.qtgui_histogram_sink_x_0.enable_autoscale(True)
self.qtgui_histogram_sink_x_0.enable_accumulate(False)
self.qtgui_histogram_sink_x_0.enable_grid(False)
self.qtgui_histogram_sink_x_0.enable_axis_labels(True)

if not True:
  self.qtgui_histogram_sink_x_0.disable_legend()

labels = ['', '', '', '', '',
          '', '', '', '', '']
widths = [1, 1, 1, 1, 1,
          1, 1, 1, 1, 1]
colors = ["blue", "red", "green", "black", "cyan",
          "magenta", "yellow", "dark red", "dark green", "dark blue"]
styles = [1, 1, 1, 1, 1,
          1, 1, 1, 1, 1]
markers = [-1, -1, -1, -1, -1,
           -1, -1, -1, -1, -1]
alphas = [1.0, 1.0, 1.0, 1.0, 1.0,
          1.0, 1.0, 1.0, 1.0, 1.0]
for i in xrange(1):
    if len(labels[i]) == 0:
        self.qtgui_histogram_sink_x_0.set_line_label(i, "Data {0}".format(i))
    else:
        self.qtgui_histogram_sink_x_0.set_line_label(i, labels[i])
    self.qtgui_histogram_sink_x_0.set_line_width(i, widths[i])
    self.qtgui_histogram_sink_x_0.set_line_color(i, colors[i])
    self.qtgui_histogram_sink_x_0.set_line_style(i, styles[i])
    self.qtgui_histogram_sink_x_0.set_line_marker(i, markers[i])
    self.qtgui_histogram_sink_x_0.set_line_alpha(i, alphas[i])
```

```
        self._qtgui_histogram_sink_x_0_win =
sip.wrapinstance(self.qtgui_histogram_sink_x_0.pyqwidget(), Qt.QWidget)
        self.top_grid_layout.addWidget(self._qtgui_histogram_sink_x_0_win, 0, 2, 2, 1)
        [self.top_grid_layout.setRowStretch(r,1) for r in range(0,2)]
        [self.top_grid_layout.setColumnStretch(c,1) for c in range(2,3)]
        self.pluto_source_0_0 = iio.pluto_source('ip:SDR25.local', int(freq_25),
int(samp_rate), int(100000), memoria, True, True, True, "manual", 50, '', True)
        self.pluto_source_0 = iio.pluto_source('ip:SDR24.local', int(freq_24),
int(samp_rate), int(100000), memoria, True, True, True, "manual", 48, '', True)
        self.pluto_sink_0_0 = iio.pluto_sink('ip:SDR25.local', int(freq_25), int(samp_rate),
int(100000), memoria, False, 0, '', True)
        self.pluto_sink_0 = iio.pluto_sink('ip:SDR24.local', int(freq_24), int(samp_rate),
int(100000), memoria, False, 0, '', True)
        self.low_pass_filter_0 = filter.fir_filter_ccf(256, firdes.low_pass(
                1, samp_rate, 1000, 1000, firdes.WIN_HAMMING, 6.76))
        self.iio_modulo_ff_0 = iio.modulo_ff(1)
        self.blocks_multiply_const_vxx_1_0 = blocks.multiply_const_vcc((gain24, ))
        self.blocks_multiply_const_vxx_1 = blocks.multiply_const_vcc((gain25, ))
        self.blocks_multiply_const_vxx_0_0 = blocks.multiply_const_vff((0.009549274, ))
        self.blocks_multiply_const_vxx_0 = blocks.multiply_const_vff((95.492, ))
        self.blocks_multiply_conjugate_cc_1 = blocks.multiply_conjugate_cc(1)
        self.blocks_multiply_conjugate_cc_0_0 = blocks.multiply_conjugate_cc(1)
        self.blocks_multiply_conjugate_cc_0 = blocks.multiply_conjugate_cc(1)
        self.blocks_delay_1_0 = blocks.delay(gr.sizeof_gr_complex*1, comp24)
        self.blocks_delay_1 = blocks.delay(gr.sizeof_gr_complex*1, comp25)
        self.blocks_delay_0_0 = blocks.delay(gr.sizeof_gr_complex*1, Calibrar_24)
        self.blocks_delay_0 = blocks.delay(gr.sizeof_gr_complex*1, Calibrar_25)
        self.blocks_conjugate_cc_0 = blocks.conjugate_cc()
        self.blocks_complex_to_real_0_2 = blocks.complex_to_real(1)
        self.blocks_complex_to_real_0_1_0 = blocks.complex_to_real(1)
        self.blocks_complex_to_real_0_1 = blocks.complex_to_real(1)
        self.blocks_complex_to_real_0_0_0 = blocks.complex_to_real(1)
        self.blocks_complex_to_real_0_0 = blocks.complex_to_real(1)
        self.blocks_complex_to_real_0 = blocks.complex_to_real(1)
        self.blocks_complex_to_arg_0 = blocks.complex_to_arg(1)
        self.blocks_add_xx_1_0 = blocks.add_vcc(1)
        self.blocks_add_xx_1 = blocks.add_vcc(1)
        self.blocks_add_xx_0 = blocks.add_vff(1)
        self.analog_sig_source_x_0 = analog.sig_source_c(samp_rate,
analog.GR_COS_WAVE, freq, 1, 0)
        self.analog_const_source_x_0_0 = analog.sig_source_f(0,
analog.GR_CONST_WAVE, 0, 0, 2*3.1416)
```

```
        self.analog_const_source_x_0 = analog.sig_source_f(0,
analog.GR_CONST_WAVE, 0, 0, 2*3.1416)

        ##################################################
        # Connections
        ##################################################
        self.connect((self.analog_const_source_x_0, 0), (self.blocks_add_xx_0, 1))
        self.connect((self.analog_const_source_x_0_0, 0), (self.iio_modulo_ff_0, 1))
        self.connect((self.analog_sig_source_x_0, 0), (self.blocks_complex_to_real_0_0, 0))
        self.connect((self.analog_sig_source_x_0, 0), (self.blocks_delay_1, 0))
        self.connect((self.analog_sig_source_x_0, 0), (self.blocks_delay_1_0, 0))
        self.connect((self.analog_sig_source_x_0, 0), (self.blocks_multiply_conjugate_cc_0,
0))
        self.connect((self.analog_sig_source_x_0, 0),
(self.blocks_multiply_conjugate_cc_0_0, 0))
        self.connect((self.analog_sig_source_x_0, 0), (self.pluto_sink_0, 0))
        self.connect((self.analog_sig_source_x_0, 0), (self.pluto_sink_0_0, 0))
        self.connect((self.blocks_add_xx_0, 0), (self.iio_modulo_ff_0, 0))
        self.connect((self.blocks_add_xx_1, 0), (self.blocks_delay_0, 0))
        self.connect((self.blocks_add_xx_1_0, 0), (self.blocks_delay_0_0, 0))
        self.connect((self.blocks_complex_to_arg_0, 0), (self.blocks_add_xx_0, 0))
        self.connect((self.blocks_complex_to_real_0, 0), (self.qtgui_time_sink_x_0, 1))
        self.connect((self.blocks_complex_to_real_0_0, 0), (self.qtgui_time_sink_x_0, 0))
        self.connect((self.blocks_complex_to_real_0_0, 0), (self.qtgui_time_sink_x_0_0, 0))
        self.connect((self.blocks_complex_to_real_0_0_0, 0),
(self.blocks_multiply_const_vxx_0_0, 0))
        self.connect((self.blocks_complex_to_real_0_1, 0), (self.qtgui_time_sink_x_0, 2))
        self.connect((self.blocks_complex_to_real_0_1_0, 0), (self.qtgui_time_sink_x_0_0,
2))
        self.connect((self.blocks_complex_to_real_0_2, 0), (self.qtgui_time_sink_x_0_0, 1))
        self.connect((self.blocks_conjugate_cc_0, 0), (self.low_pass_filter_0, 0))
        self.connect((self.blocks_delay_0, 0), (self.blocks_complex_to_real_0, 0))
        self.connect((self.blocks_delay_0, 0), (self.blocks_multiply_conjugate_cc_0, 1))
        self.connect((self.blocks_delay_0_0, 0), (self.blocks_complex_to_real_0_2, 0))
        self.connect((self.blocks_delay_0_0, 0), (self.blocks_multiply_conjugate_cc_0_0,
1))
        self.connect((self.blocks_delay_1, 0), (self.blocks_multiply_const_vxx_1, 0))
        self.connect((self.blocks_delay_1_0, 0), (self.blocks_multiply_const_vxx_1_0, 0))
        self.connect((self.blocks_multiply_conjugate_cc_0, 0), (self.blocks_conjugate_cc_0,
0))
        self.connect((self.blocks_multiply_conjugate_cc_0, 0),
(self.blocks_multiply_conjugate_cc_1, 0))
```

```
        self.connect((self.blocks_multiply_conjugate_cc_0_0, 0),
(self.blocks_multiply_conjugate_cc_1, 1))
        self.connect((self.blocks_multiply_conjugate_cc_1, 0),
(self.blocks_complex_to_arg_0, 0))
        self.connect((self.blocks_multiply_const_vxx_0, 0),
(self.qtgui_histogram_sink_x_0, 0))
        self.connect((self.blocks_multiply_const_vxx_0, 0), (self.qtgui_number_sink_0_0,
0))
        self.connect((self.blocks_multiply_const_vxx_0_0, 0),
(self.qtgui_number_sink_0_0_0, 0))
        self.connect((self.blocks_multiply_const_vxx_1, 0), (self.blocks_add_xx_1, 1))
        self.connect((self.blocks_multiply_const_vxx_1, 0),
(self.blocks_complex_to_real_0_1, 0))
        self.connect((self.blocks_multiply_const_vxx_1_0, 0), (self.blocks_add_xx_1_0, 1))
        self.connect((self.blocks_multiply_const_vxx_1_0, 0),
(self.blocks_complex_to_real_0_1_0, 0))
        self.connect((self.iio_modulo_ff_0, 0), (self.blocks_multiply_const_vxx_0, 0))
        self.connect((self.iio_modulo_ff_0, 0), (self.qtgui_number_sink_0, 0))
        self.connect((self.low_pass_filter_0, 0), (self.blocks_complex_to_real_0_0_0, 0))
        self.connect((self.pluto_source_0, 0), (self.blocks_add_xx_1_0, 0))
        self.connect((self.pluto_source_0_0, 0), (self.blocks_add_xx_1, 0))

    def closeEvent(self, event):
        self.settings = Qt.QSettings("GNU Radio", "top_block")
        self.settings.setValue("geometry", self.saveGeometry())
        event.accept()

    def get_freq(self):
        return self.freq

    def set_freq(self, freq):
        self.freq = freq
        self.set_samp_rate(self.freq*100)
        self.analog_sig_source_x_0.set_frequency(self.freq)

    def get_samp_rate(self):
        return self.samp_rate

    def set_samp_rate(self, samp_rate):
        self.samp_rate = samp_rate
        self.qtgui_time_sink_x_0_0.set_samp_rate(self.samp_rate)
        self.qtgui_time_sink_x_0.set_samp_rate(self.samp_rate)
```

```
        self.pluto_source_0_0.set_params(int(self.freq_25), int(self.samp_rate), int(100000),
True, True, True, "manual", 50, '', True)
        self.pluto_source_0.set_params(int(self.freq_24), int(self.samp_rate), int(100000),
True, True, True, "manual", 48, '', True)
        self.pluto_sink_0_0.set_params(int(self.freq_25), int(self.samp_rate), int(100000), 0,
'', True)
        self.pluto_sink_0.set_params(int(self.freq_24), int(self.samp_rate), int(100000), 0, '',
True)
        self.low_pass_filter_0.set_taps(firdes.low_pass(1, self.samp_rate, 1000, 1000,
firdes.WIN_HAMMING, 6.76))
        self.analog_sig_source_x_0.set_sampling_freq(self.samp_rate)

    def get_puntos(self):
        return self.puntos

    def set_puntos(self, puntos):
        self.puntos = puntos

    def get_memoria(self):
        return self.memoria

    def set_memoria(self, memoria):
        self.memoria = memoria

    def get_gain25(self):
        return self.gain25

    def set_gain25(self, gain25):
        self.gain25 = gain25
        self.blocks_multiply_const_vxx_1.set_k((self.gain25, ))

    def get_gain24(self):
        return self.gain24

    def set_gain24(self, gain24):
        self.gain24 = gain24
        self.blocks_multiply_const_vxx_1_0.set_k((self.gain24, ))

    def get_freq_25(self):
        return self.freq_25

    def set_freq_25(self, freq_25):
        self.freq_25 = freq_25
```

```
        self.pluto_source_0_0.set_params(int(self.freq_25), int(self.samp_rate), int(100000), True, True, True, "manual", 50, '', True)
        self.pluto_sink_0_0.set_params(int(self.freq_25), int(self.samp_rate), int(100000), 0, '', True)

    def get_freq_24(self):
        return self.freq_24

    def set_freq_24(self, freq_24):
        self.freq_24 = freq_24
        self.pluto_source_0.set_params(int(self.freq_24), int(self.samp_rate), int(100000), True, True, True, "manual", 48, '', True)
        self.pluto_sink_0.set_params(int(self.freq_24), int(self.samp_rate), int(100000), 0, '', True)

    def get_comp25(self):
        return self.comp25

    def set_comp25(self, comp25):
        self.comp25 = comp25
        self.blocks_delay_1.set_dly(self.comp25)

    def get_comp24(self):
        return self.comp24

    def set_comp24(self, comp24):
        self.comp24 = comp24
        self.blocks_delay_1_0.set_dly(self.comp24)

    def get_Update(self):
        return self.Update

    def set_Update(self, Update):
        self.Update = Update
        self.qtgui_time_sink_x_0_0.set_update_time(self.Update)
        self.qtgui_time_sink_x_0.set_update_time(self.Update)
        self.qtgui_number_sink_0_0_0.set_update_time(self.Update)
        self.qtgui_number_sink_0_0.set_update_time(self.Update)
        self.qtgui_number_sink_0.set_update_time(self.Update)
        self.qtgui_histogram_sink_x_0.set_update_time(self.Update)

    def get_Nivel(self):
        return self.Nivel
```

```
    def set_Nivel(self, Nivel):
        self.Nivel = Nivel
        self.qtgui_waterfall_sink_x_0.set_intensity_range(self.Nivel, 10)

    def get_Calibrar_25(self):
        return self.Calibrar_25

    def set_Calibrar_25(self, Calibrar_25):
        self.Calibrar_25 = Calibrar_25
        self.blocks_delay_0.set_dly(self.Calibrar_25)

    def get_Calibrar_24(self):
        return self.Calibrar_24

    def set_Calibrar_24(self, Calibrar_24):
        self.Calibrar_24 = Calibrar_24
        self.blocks_delay_0_0.set_dly(self.Calibrar_24)


def main(top_block_cls=top_block, options=None):

    if StrictVersion("4.5.0") <= StrictVersion(Qt.qVersion()) < StrictVersion("5.0.0"):
        style = gr.prefs().get_string('qtgui', 'style', 'raster')
        Qt.QApplication.setGraphicsSystem(style)
    qapp = Qt.QApplication(sys.argv)

    tb = top_block_cls()
    tb.start()
    tb.show()

    def quitting():
        tb.stop()
        tb.wait()
    qapp.aboutToQuit.connect(quitting)
    qapp.exec_()


if __name__ == '__main__':
    main()
```