Republic of Iraq
Ministry of Higher Education
and Scientific Research
University of Babylon
College of Engineering
Electrical Engineering Departement

Implementation of Radio Frequency Signal Tracking System Based on Direction of Arrival Technique

A Thesis

Submitted to the College of Engineering
of the University of Babylon in Partial Fulfillment
of the Requirements for the Degree of Doctor of Philosophy
in Engineering/ Electrical Engineering/ Electronics and Communications

by

Ali Najim Abdullah Sabti

Supervised by

Prof. Dr. Laith Ali Abdul-Rahaim

2021 A.D ***1443 A.H***

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

﴿ قَالُوا سُبْحَانَكَ لَا عِلْمَ لَنَا إِلَّا مَا

عَلَّمْتَنَا إِنَّكَ أَنتَ الْعَلِيمُ الْحَكِيمُ ﴾

صَدَقَ اللَّهُ الْعَلِيّ الْعَظِيم

البقرة (23)

To
The Memory of My Father

To

My Family, My Wife
and Friends with Love and
Respect

Acknowledgements

In the name of Allah, the Most Gracious and the Most Merciful

First and above all, I praise ALLAH, the almighty, for providing me with this opportunity and granting me the capability to proceed with this thesis.

I want to express my thanks, respect, and gratitude to my supervisor Prof. Dr. Laith Ali Abdul-Rahaim, who stood with me throughout the research and supported me in difficult times. I ask ALLAH always to keep him and help him in his life.

I would like to warmly thank my parents, mother, and brothers for their material and spiritual support in all aspects of my life.

I want to thank my wife for supporting and encouraging me during the study period.

I would also like to thank the Department of Electrical Engineering and the College of Engineering in general to provide all the supplies and devices needed to complete this thesis.

Ali Najim Abdullah

2021

Abstract

Radio Frequency Signal Tracking (RFST) system is one of the essential technologies considered a tool for monitoring and retrieving radio frequency signals. Therefore, an RFST system based on Direction of Arrival (DOA) has been proposed through which radio frequency spectrum usage is monitored to radio environment checks to scan for unauthorized Radio Emission Sources (RES).

The proposed system is a network of connected devices, computers and RF Tracking Stations. RF Tracking Stations are vehicles that carry signals tracker equipment and are connected to the operations center. It monitors the frequency spectrum, tracks RES angles and interrupts RES directions using the triangulation process to calculate the location of the RES.

This thesis deals with two main parts: the first is the simulation of the work of the proposed system, and the second is the practical implementation of the system from the hardware and software components of the proposed system. The simulation part implemented by Matlab software offers different methods such as simulating the work of seven algorithms: Beamscan, MVDR, MUSIC, ESPRIT, Beamspace-ESPRIT ROOT-MUSIC, ROOT-WSF. These algorithms track the RES angles by using an antenna array to obtain the tracking process with the best accuracy.

Three shapes of antenna arrays, linear, rectangular and circular, are tested with tracking algorithms to extract the best accuracy of DOA from the RF tracking station. Furthermore, the RFST system is designed with different techniques from the geographically distributed RF Tracking Stations to extract the coordinates of wireless sources operating at 990 MHz.

The practical side includes hardware and software components carried on the vehicle or located in the operations center. The vehicle contains a Raspberry Pi3 microcomputer with two Software Defined Radios (SDR). The first SDR operates with a four-antenna array and is used to track the RES angles and the second for monitoring the spectrum. The operations center contains an SDR spectrum analyzer and operations center GUI software designed by MATLAB, linked to the internet. These parts work together to achieve the system's primary goals, including periodic or permanent real-time monitoring of radio signals in the frequency range from 25 MHz to 1.75 GHz, effective tracking and accurate monitoring of radio sources.

The system work starts from the operations center that displays all the tracking tours for specific RES by sending the RF Tracking Stations a set of data for the tracking tour shared by the internet. In addition, the operations center has a spectrum analyzer associated with the operation center GUI to track RES. The data is continuously updated with the operations center to synchronize with the last tracking rounds of specific RES. The RF Tracking Station operator adjusts the tracking operation parameters to locate the RES using triangulation based on tracking tours and data from different locations.

The simulation results show that the best tracking algorithms are ROOT-MUSIC and ROOT-WSF with linear array and ROOT-MUSIC with a circular antenna array. The RFST system was also tested with different configurations starting from one station node to four stations. The results showed that the use of four stations provides the best RES estimation with an error distance of up to 2m with the ROOT-WSF algorithm. In The practical results, the minimum error distances are 4.23m and 4m when using the MUSIC tracking algorithm with ULA and UCA. The comparison of simulation and practical results show the proposed system has been built accurately on both sides.

Table of Contents

List of Symbols

A	The Steering Vectors Matrix
$\overline{\mathrm{A}}$	Multiplication of The Noise Subspace
$\mathrm{a}(\theta_0)$	The Steering Vector Related with The Angle θ_0
$\mathrm{a}^H(\theta)$	Transpose of A Complex Conjugate of The Steering Vector
$\bar{a}$	The Entries Sum of $\overline{\mathrm{A}}$
$a_m(\theta)$	The Steering Vector
c	The Speed of Light
D	The Diagonal Matrix
f_c	The Carrier Frequency
$G(z)$	Z-Transformation of $\overline{\mathrm{A}}$
I	Zero Mean
ℓ	Index of The Source Signal Components
K_1	The First Selection Matrix
K_2	The Second Selection Matrix
m	The Antenna Elements
n	Time Index
$n_m[n]$	The Sampled Noise Vector
$\mathrm{n}_{\mathrm{x}_1}$	The Noise Vector in The First Subarray
$\mathrm{n}_{\mathrm{x}_2}$	The Noise Vector in The Second Subarray
n_n	Gaussian Noise
q_n	Eigenvectors of The Spatial Correlation Matrix
Q	Matrix Subspace
Q_n	Noise Subspace Matrix
Q_s	Signal Subspace
R	Hermitian Matrix
$s(t)$	Incoming Signal at The Array Elements
s_ℓ	Components of Source Signal
s_n	The Signal Vector
w^H	Transpose of A Complex Conjugate of Antenna Element Weight
x_1	The First Subarray Vector
x_2	The Second Subarray Vector
x_n^H	Transpose of Complex Conjugate The Received Samples

$x_m[n]$	The Sampled Signals
(x, y)	The Coordinates of Source Location
y(n)	The Incident Signal
Π_p	The Exchanging Matrix
$P_{\mathrm{BS}}(\theta)$	The Output Power of The Beamformer
$P_{\mathrm{BS}}(\theta_0)$	The Average Received Power
$P_{\mathrm{MUSIC}}(\theta)$	MUSIC Received Power
$P_{\mathrm{MVDR}}(\theta)$	MVDR Received Power
U_n	Noise Subspaces Matrix
U_s	Signal Subspaces Matrix
θ_l	The Incoming Signal Angle
σ^2	Covariance Matrix
σ_n^2	The Noise Power
σ_s^2	The Signal Covariance Matrix
ϕ_ℓ	The Eigenvalues of Φ
Δt_m	The Time Delay
Λ	The Eigenvalue of The Signal Subspaces
Φ	Diagonal Elements
Ψ	Nonsingular Matrix
ω	The Weight of Antenna Elements
λ	Wavelength

List of Abbreviations

ADC	Analog to Digital Converter
AOA	Angle of Arrival
CDF	Crossing Direction Finding
CPU	Central Processing Unit
CSD	Cross-Spectral Density
DCS	Digital Cellular System
DEUCE	Direction Estimation For The Uncorrelated Emitters
DF	Direction Finding
DOA	Direction of Arrival
DSP	Digital Signal Processing
ESPRIT	Estimation of Signal Parameters via Rotational Invariance Technique
FDFIB	Frequency-Domine-Frequency Inverint Beamformer
FFT	Fast Fourier Transform
IOT	Internet of Things
JADE	Joint AOA and DOA Estimation
LOB	Line of Bearing
MSWF	Multi-Stage Wiener Filter
MUSIC	Multiple Signal Classification
MVDR	Capon's Minimum Variance Distortionless Response
RDF	Radio Direction Finder
RES	Radio Emission Sources
RFST	Radio Frequency Signal Tracking
RSS	Receiving Signal Strength
RSS	Receiving Signal Strength
SDR	Software Defined Radio
SNR	Signal to Noise Ratio
SOI	Signal of Interest
TDF	ESPRIT-time-delay frequency technique
TDOA	Time Difference of Arrival
TLS	Total Least Square
TOA	Time of Arrival
UCA	Uniform Circle Array
ULA	Uniform Linear Array
URA	Uniform Rectangular Array
USRP	Universal Software Radio Peripheral
VULA	Virtual Uniform Linear Array
WSF	Weighted Subspace Fitting
WSN	Wireless Sensors Networks

List of Figures

List of Tables

List of Publications

The following papers were published during the preparation of this thesis.

1. Ali Najim Abdullah, Laith Ali Abdul-Rahaim, "**Comparative Study of Super- Performance DOA Algorithms based for RF Source Direction Finding and Tracking**," Technology Reports of Kansai University, vol. 63, No. 2, 2021.

2. Ali Najim Abdullah, Laith Ali Abdul-Rahaim, "**Enhancing the Performance of Localization System for Radio Frequency Transmitters Based on DOA and Triangulation algorithms**," International Journal of Engineering Trends and Technology, vol. 69, August 2021.

3. Ali Najim Abdullah, Laith Ali Abdul-Rahaim, "**Performance Enhancement of Array Configurations with DF Algorithms Based for RDF**," 2021 1st Babylon International Conference on Information Technology and Science (BICITS), 2021.

4. Ali Najim Abdullah, Laith Ali Abdul-Rahaim, "**Design and Enhancement of RF Emitters Localization System Based on DF Techniques**," 2021 international conference on Advanced Computer Applications (ACA2O21).

Chapter One

General Introduction

1.1 Background

Radio Frequency Signal Tracking (RFST) is an important system widely used in various areas and was developed to provide information about transmission systems through radio channels. RFST based on Direction of Arrival (DOA) can effectively counteract many threats by identifying, tracking and localizing dangerous Radio Emission Sources (RES). Commercial and public radio systems services, and state agencies as a whole, is under the threat of radio attacks and must take reciprocal measures to employ RFST as an effective counteraction to these threats [1].

The system consists of fixed and mobile RF Tracking stations, distributed as required within a specific area. These stations explore the frequency spectrum and track the directions of the RES through the tracking algorithms at each station. When the direction of the RES is detected through more than one station, the system interrupts these directions to determine the location of the RES [1].

When the RFST system is integrated with DOA and Internet, it is considered a new great development that generates a new era of RFST technological that is determined to achieve good performance under several tracking scenarios. Therefore, Internet enables RFST to coverage the required area [2].

The RFST system is a system of RF Tracking Stations with an operations center linked with Internet to achieve RES Tracking and location estimation. RFST prove instantaneous tracking data and location estimation of the signal received from the RES. It is also synchronous with distributed RF Tracking Stations for RES tracking and location estimations and mapping representation through Google map [2].

1.2 Problem Definition

Convolutional RFST system suffers from various problems such as complicated manual tracking, scanning area within the small coverage area, instantaneous common information are not available about specific RF source, monitoring the spectrum with the limited band and limited tracking algorithms. In addition, the installation of these systems is difficult, expensive, and requires specific communication infrastructure, which requires periodic maintenance. All these problems appear the compelling need for the development of RFST technologies and equipment, and the creation, by the researchers. The RFST system with DOA and Internet becomes more effective by addressing all its problems, the unified approach to its structures, and the multifunctionality and universality of its solutions.

1.3 Literature Review

RFST system is attractive for tracking applications in modern communication infrastructure. Therefore, many research papers on this topic have been presented, cited and these papers represent the base research of the RFST.

(R. Schmidt and R. Franks, 1986) [3], presented the experimentally confirms the MUSIC set of rules which became carried out to the overall performance of the sign subspace method to DF below very fashionable eventualities and conditions. Their small tool consisted of an eight-element antenna array thirteen wavelengths in diameter, an eight-channel receiver and digitizer, and a minicomputer with disk storage to method the digitized data.

(Y. T. Chan and K. C. Ho, 1994) [4], shows a method for locating a source using several hyperbolic curves included by Time Difference of Arrival (TDOA) data. The nonlinear equations related to TDOA estimates and source location can be turned into linear equations in the unknown factors and the intermediate variable by introducing an intermediate variable. It is as simple as spherical interpolation but

performs substantially better, especially when minimal sensor numbers than previous localization methods.

(Muhamed, Rias, 1996) [5], His thesis aims to design, develop, and test a DOA measurement system employing a 2050 MHz antenna array. The MUSIC and ESPRIT algorithms are two DOA techniques discussed. These algorithms have simulated and compared it under various scenarios. The practical part is a six-element evenly spaced linear array receiver and three Ariel DSP-96 boards based on the Motorola DSP96002 to estimate DOA techniques.

(Hu, Zhong, 1998) [6], implemented an eight-element antenna array system for evaluating various DOA and localization algorithms, such as MUSIC and ESPRIT algorithms. Although more complicated, joint AOA and DOA Estimation (JADE) algorithms were proposed to have higher resolutions in both space and time domains potentially. Using JADE algorithms, DOA is resolved through a fraction of the sample period in the case of bandlimited known signals.

(S. Shahbazpanahi, S. Valaee and M. H. Bastani, 2001) [7], proposed algorithm based on ESPRIT to estimate the central angle and scattered source. TLS-ESPRIT was developed to calculate the central angles for incoherent and coherent sources. The extension widths of coherent sources are calculated by creating a one-dimensional source parameter estimator (DSPE) spectrum for each source. The extension widths of incoherently dispersed sources are computed using the distribution's core moments.

(L. A. Cirillo, A. M. Zoubir and A. B. Gershman, 2002) [8], proposed a DF technique for FM signals based on an averaged spatial time-frequency distribution. The direction estimation for the uncorrelated emitters (DEUCE) technique improves the performance. This method requires the knowledge of the sources auto-term locations on the time-frequency plane. When no prior knowledge of the sources'

time-frequency plan is available, an automated point is chosen. The suggested technique produces results that are pretty close to known auto-term locations.

(Qin, Hongfeng, Jianguo Huang, and Qunfei Zhang, 2003) [9], used ESPRIT-TDF method and a high-resolution time-delay frequency technique for multiple source localization. The generalized eigenvalues and eigenvectors automatically pair DOA and time-delay and estimate them simultaneously. The ESPRIT-TDF offers good joint parameter estimation performance and does not require a phase for emitting signals. It is also simple to build due to its small calculation.

(Do-Hong, Tuan, Franz Demmel, and Peter Russer, 2004) [10], based on the FDFIBs and the MUSIC estimator, it gives an analytic investigation of the RMSE and the resolution of the threshold for wideband DOA estimation (FDFIB-MUSIC). By increasing the number of time samples, the SNR, and the number of FDFIBs, the resolution threshold was enhanced while computing complexity was reduced.

(Huang, Lei, Shunjun Wu, and Linrang Zhang, 2005) [11], proposed using the multi-stage wiener filter to construct the MUSIC algorithm (MSWF). MSWF was used with forwarding recursions to estimate the noise subspace, even incoherent signals. His method outperforms the conventional MUSIC algorithm, which calculates the sample covariance matrix and eigenvectors.

(V. Y. Vu and A. B. Delai, 2006) [12], presented the DOA measurement in an azimuth plan utilizing ULA and an all-digital receiver array in which ADC immediately digitizes the Radio Frequency signal. His study aimed to monitor DOA in real-time for an inter-vehicle in intelligent transportation systems. DOA is calculated using the MUSIC high-resolution approach. The measuring system consists of four quasi-Yagi elements ULA and four parallel all-digital receivers controlled by a digital oscilloscope.

(Al Jabr, Kareem, 2007) [13], presented the UCA-ESPRIT and UCA-ROOT-MUSIC algorithms. The mode excitation approach was utilized in element space to convert the UCA into a virtual uniform linear array (VULA) in mode space. The Hermitian Toeplitz matrix was recreated from the observed data vector using a decor-related technique to reorganize the data and increase the dimensionality of the noise space. As a result, signal and noise spaces might be calculated more precisely.

(Goossens, Roald, Hendrik Rogier, and Steven Werbrouck, 2008) [14], proposed sparse UCAs were used to estimate DOA. The sparse UCA-ROOT-MUSIC, derived from the classic UCA-ROOT-MUSIC. Even in practical arrays where the antenna components are not equal and there was no perfect circular symmetry, his approach enhanced the accuracy of DOA estimates with a limited number of elements.

(Reed, Jesse, 2009) [15], focused on the challenge of source localization using a network of passive wireless sensors. At each sensor, localization is accomplished using DF or range estimation. His research includes single-target and multiple-target scenarios and two signal-selective localization techniques. The first uses the well-known cyclic MUSIC technique, while the second employs modulation and beamforming categorization. UCA is used as an RF front end, and the USRP is used as a processor.

(W Tidd, Will, 2010) [16], used 8-element UCA to configure source localization and tracking system. His research aims to accomplish exact source localization, fast spectral sensing, quick target discrimination and recognition, jamming and multipath signal suppression, and real-time location tracking. The studies used two targets moving at different directions and speeds to test and verify the MUSIC with Kalman tracking.

(Kulaib, Ahmed Rashed, 2011) [17], presented the comparison of linear and circular array shapes in DOA estimation. A comprehensive analytical model for each geometry was created under various sensor array and signal conditions, the comparative numerical research illustrates the benefits and downsides of each technique. His comparison focuses on numerous factors, including noise level, beamwidth, and accuracy for WSN localization.

(Ling, Wang, and Shuzhi Sam Ge, 2011) [18], showed a Crossing Direction Finding (CDF) method for locating numerous targets. If the directions to the other two determined sites are known ahead of time, the position of a target is unique. The target's two 2D DOAs are collected utilizing two cooperating radars, respectively. The L-shape array and conjugate-ESPRIT technique automatically match the 2D DOAs. Using a single reference sensor in different coordinate systems, the DOAs of multiple targets are automatically matched.

(Malanowski, Mateusz, 2012) [19], presented a target tracking algorithm for passive radar. The technique has been implemented in real-world applications. The precision of localization is improved by combining bistatic and Cartesian trackers. On both simulated and real-world data, the method performed admirably. Future studies should incorporate DOA and the use of more than three transmitters and numerous receivers.

(Baig, Nauman Anwar, and Mohammad Bilal Malik, 2013) [20], deal with increasing the DOA estimate's resolution allows for separating closely spaced targets in space. MVDR has a good resolution due to the output power minimization subject to the limitation. The resolution of the MUSIC algorithm is higher. The beamwidth is reduced when directed sensors are used instead of omnidirectional sensors, and the gain increases. This research concludes that ESPRIT is more computationally efficient than other methods since it does not require a scan of all possible angles.

(Susaritha, U. S., and Ashita Priya Thomas, 2014) [21], presented a comparison of different DOA estimate methods and adaptive beamforming applications for target detection and tracking. With initial data obtained from RSS measurement in a Wi-Fi environment, DOA algorithms such as Beamscan, MVDR, MUSIC, ESPRIT, and ROOT-MUSIC are used. The ROOT-MUSIC and ESPRIT algorithms used a moving object to estimate its location. The ideal combination for the best performance is discovered. The simulations were done out using the Matlab.

(PĂUN, Mirel, Răzvan TAMAS, and Ion Marghescu, 2015) [22], proposed a practical implementation of a Software-Defined Radio (SDR) localization system that is passive. A USRP N200 platform with a TVRX2 RF daughterboard is used in the proposed system. MUSIC and Cross-Spectral Density (CSD) are two techniques that have been tested for implementing the DOA estimate. Both approaches, MUSIC and CSD, show good overall accuracy in the single source experiment. However, in the case of several target sources, the MUSIC algorithm is the sole option.

(R. K . B., 2015) [23], suggested an orthogonally polarized linear array structure relative to the commonly used two-dimensional array. An elegant one-dimensional search technique is used to calculate 2D-DOA for a single source scenario. 2D-DOA estimation of one and two wideband sources by a two-sub-band filter technique is given. The two-sub-band filter method, combined with orthogonally polarized array topologies, enhanced estimation accuracy while requiring little computation time in this simulation investigation.

(Abdessamad, Wissam, 2016) [24], proposed a Software Defined Radio (SDR) tool for detecting and tracking illicit broadcasts in 3G and 4G networks. His work developed a DF tool on USRPs and supported it with statistical analysis equipment of the 3G. The created platform offers high precision at a low cost.

(Kristiyana, Samuel, 2017) [25], proposed three different RDF stations are used. The direction data is taken from each RDF station without using a mobile human operator, and the triangle centroid algorithm is used to estimate it. As a result, the position of the source is automatically calculated. The triangle centroid algorithm is used to estimate the source imaginary. This experiment found that the imaginary RF source is roughly 3.3m for longitude and 3.2m for latitude compared to the genuine source.

(Btissam B., 2018) [26], presented a performance evaluation of two estimation methods, ROOT-MUSIC and ROOT-WSF, under extreme signal degradation conditions. The purpose of this study is to compare and contrast the two DOA estimate techniques using four criteria: closely spaced sources, noise detection high frequency, and correlated signal. Under these challenging conditions, the ROOT-WSF method enables more accurate DOA detection.

(Miyamoto, Kazuki, Nobuyoshi Kikuma, and Kunio Sakakibara, 2019) [27], discussed the method of using compressed sensing to estimate DOA. This work introduces FFT into compressed sensing to use numerous snapshots of the array-received signal. The operation of compressed sensing by FFT is examined using computer simulation, demonstrating that many snapshots are successfully employed for DOA estimation using compressed sensing.

(Btissam B., 2019) [28], studied ESPRIT, MUSIC, WSF, MVDR, and beamspace are some of the DOA techniques. These algorithms are compared and assessed against a set of requirements, including the number of elements, sample size, and the number of received signals. The primary goal of this research is to find the optimum DOA estimation that can be used in an intelligent antenna system. In each instance, ROOT-WSF gives a more accurate identification of DOA.

(Zuokun Li, 2020) [29], used a four-element DOA estimation system of KerberosSDRs with 4-omnidirectional antennas with a Raspberry Pi. The four antennas are positioned in a ULA and UCA pattern, respectively. MUSIC, Bartlett, MEM, and MVDR are among the algorithms employed. The DOA for the stationary target signal source is estimated using the MUSIC algorithm, and the source position is displayed on the map.

(Araszkiewicz, Piotr, Norbert Niderla, and Kacper Murat, 2021) [30], focuses on the issue of finding IOT devices in an open environment. For identifying devices with their low-power radio signal, passive approaches may be more efficient in calculation and hardware. Two passive notions are the measurement of amplitude and arrival time disparities of signals acquired from three receivers. These techniques were combined to get findings with a modest measurement error.

Many previous works focused only on the DOA algorithm function, an essential factor in the RFST system. They did not touch on the collection center of the tracking data, instantaneous data sharing, tracking area range instantaneous radio source tracking. Mainly, RFST needs to have higher accuracy, small physical size, flexibility, versatility, played installation and so on, due to reliable estimation algorithms, dynamic spectrum range, and wide operating frequency range. In literature, researchers have reported different techniques to enhance algorithms, techniques, and accuracy, which are devoted to a literature review.

In addition, several past research works are presented to understand current challenges in this field. Our proposed system is building an RFST system based on Internet. Various techniques for collecting data and tracking sources are studied within a specific area. Then, RF tracking units mounted on vehicles are deployed to compile tracking information and share it instantly with the operations center. Therefore, the radio sources are constantly tracked.

1.4 Thesis Contribution

The main contribution of this thesis is developing different simulations and implementation of RFST and the use of many other techniques to enhance the accuracy of RES location determination to satisfy the best performance requirements. It can be summarized as follows:

1. Comprehensive simulation and analysis of all tracking algorithms were achieved under harsh conditions to evaluate the performance of each of them in terms of accuracy. In addition, various antenna array configurations were also simulated using tracking algorithms to extract the best possible accuracy for tracking operation.

2. Simulation of RFST system was achieved to track and localize the RES based on tracking algorithms and triangulation process with different scenarios and analysis it in terms of accuracy.

3. Internet integration with the RFST system and Google Maps to monitor and track the RES anytime and anywhere. Therefore, the resolution of the tracking operation is gained by achieving a 2m error distance with four tracking stations.

4. The RFST system was implemented, consisting of vehicle-mounted devices SDR that measures the spectrum and tracks the RES with sharing data with the operations center instantaneously, tracking the location of RES with an error distance of 4.23 meters from the original location.

5. The comparison was implemented between the RFST simulation system and the practical RFST system, an error distance of 6m in the experimental results over simulation results using four tracking points. Therefore, the proposed system provides approximate performance on both sides.

1.5 Thesis goals

This thesis aims to simulate and design a real-time RFST that can track and localize the RF emissions in the scanning area. The first approach of this thesis seeks to develop and simulate the RFST system. The second approach is experimental verification of the RFST system.

The following are the primary goals of this thesis:

i. Simulating the tracking algorithms, which is used in the final proposed system.
ii. Simulating array configuration with the tracking algorithms for getting the best tracking operation.
iii. Simulating an RFST system with different distributions of the RF tracking Stations to determine RES with measuring error distance between estimated and actual positions.
iv. Integrating the RFST with Internet to achieve remote monitoring.
v. Implementing RF tracking vehicle connected with Internet to measure the spectrum and track the new RES.
vi. Implementing the RFST system, evaluation and discussion of the results obtained.
vii. Investigating the obtained results in both practical and simulation aspects.

1.5 Thesis Organization

The work in the thesis is divided into five chapters as follows:

- **Chapter One** presents the Introduction and some of the literature survey.
- **Chapter Two** presents the concepts and architecture of the RFST and discusses some of the principles of operation and theories for each part of the proposed work.
- **Chapter Three** presents the simulation of the RFST System with various tracking algorithms and the practical implementation of RFST, the practical design, and its components.
- **Chapter Four** presents the system's results and discussions.
- **Chapter Five** includes the conclusions and the suggestions for future works.

The thesis ends with a list of references for papers and books referred to this thesis.

Chapter Two

Theoretical Background

2.1 Introduction

This chapter will discuss the basic concepts of the RFST system. Radio Direction Finding (RDF) is the main principle of the RFST system. The basic knowledge of this system is the using a network of RF tracking stations, which used for measuring the Receiving Signal Strength (RSS), Time-of-Arrival (TOA), and Time-Difference -of- Arrival (TDOA), which is called Line of Bearing (LOB) or Angle of Arrival (AOA).

Moreover, many tracking algorithms are discussed, such as Beamscan, MVDR, Esprit, Beamspace-Esprit, MUSIC, ROOT-MUSIC, and ROOT-WSF. In addition, this chapter considers RF tracking concepts and techniques using internet for RFST systems, which is a new way of combining tracking data to improve the performance of tracking technology over classic techniques that use the methods with limited operations.

2.2 Radio Direction Finding Fundamentals

A radio direction finder measures a radio wave's angle of arrival and allows the determination of its source direction. By using a single direction finder, we can determine the bearing of the RES. To estimate the source location, more of one RDF must be distributed in the scanning area. The location of unknown RES is calculated by the bearing intersection points [31].

A multiple-antenna array with one or more receivers is used in radio direction finding systems to estimate the bearing angle or geographical coordinates of an intercepted Signal of Interest (SOI). The system has a primary goal to measure a

DOA. However, it has many uses, from amateur radio contests to emergency services to military defence and intelligence activities [31].

Radio Direction Finding is traditionally categorized as a goniometric radio navigation system. The primary function of the navigations system was to locate a mobile item, such as ships or airplanes, by the Earth's coordination system. If the coordination of two or more of the radio-emitters is known, the own object coordinating can be computed and having determined its direction [32].

The signal-bearing can only be represented in a straight line for a very short distance mapped by a geodesic line (great circle) because the surface of Earth is not flat. The signal-bearing lines over vast distances, connecting the supplied sites in the shortest path possible along the surface of the Earth [32].

The used cartographic projection determines the geodesic line view. Recently, RDF for radio navigation purposes was lost significance due to the deployment of satellite navigation systems. At the same time, the need for computed RES are as relative as before, in various significant areas, which intending [32]:

- Radio-tracking issues to locate radio transmissions.
- Terrorism-related issues.
- Military uses.
- A multiple-access radio system is used in a modern radio system.
- Scientific research in the field of radio astronomy.

2.3 History of RDF Technique

In 1888, RDF approaches were discovered as Heinrich Hertz invented directional antenna properties and executed his experiment within radio waves' decimeter range. The RDF with a rotating antenna was the most popular RDF system type [32].

During World War I, the use of RDF with the frame rotating antennas. RDF system performed well in the frequency range from 200 to 1500 kHz, with a high signal to noise ratio. It permits measurements to be made to one degree of accuracy. However, RDF produced significant errors in the frequency range (3–30 MHz) [32].

In 1917, Frank Adcock discovered antennas with a vertical spike that spaced less than (1/10) of a wavelength. As a result, a directional diagram (pattern) achieves in the direction of antennas and cannot be achieved with horizontal components. In practice, Adcock's antennas have been widely used since around 1931. The space between spike antennas is much less than the wavelength. The antennas have a small stand [32].

In 1925–1926 Watson-Watt used two fixed antennas frames with a combination of a non-directional spike instead of the rotating antenna frame, as well as the use of an electron-beam tube was used to display the bearings. The approaches of Adcock and Watson-Watt are termed amplitude-phase techniques because the signal amplitude can be synchronized measurement [31, 33, 34].

During World War II, direction finders with spaced frame antennas were manufactured in the U.K. and the U.S. and used for direction finding in hectometer (short) waves. The need to develop such radio direction finders was related to the fact that direction finders with the Adcock antenna system provided significant errors at source distances of 50–350 km due to the sharp decrease of signal level upon increasing the incident angle of radio waves [32].

In 1941, the Doppler effect was discovered. Doppler and pseudo-Doppler systems use a single non-directional antenna and are phase systems. The maximum Doppler shift is detected. When the direction radio signal coincides during the rotation of the antenna at that point within the circle, this antenna passes the radio signal to the FM receiver. The output of the frequency detector provides to the phase detector, which

has a reference input linked with a sinusoidal source to drive the antenna system. The DOA is considered when the output of the phase detector achieves the maximum voltage and is properly phased of the reference signal. Using a fasting antenna switcher in the circular array, pseudo Doppler devices replicate the single antenna rotation of the Doppler RDF [31, 32, 33].

In 1943, Wullenweber RDF appeared. It consists of two directional antennas and one non-directional antenna. The system is developed, which have forty vertically positioned wideband antenna arrays arranged in a circle configuration with a radius of 120m. A metal grid is placed behind the interior circle of antennas to ensure the reception of radio signals. The RDF frequency band was 6-20 MHz antenna elements passed signals to the phase shifters and collected them to find the best pattern in the specific sector [32].

In the early 1970s, digital methods and devices were implemented into radio direction-finding systems. In the 1980s and 1990s, digital signal processing (DSP) began to be used for direction finding more often. Digital Signal Processing (DSP) became widely employed for the RDF special in 1980 and 1990. The need to detect the RDF of the signals with an extended frequency spectrum. Using DSP, a wide-band RDF is developed with utilized adaptive and programmable tuning device, merging the searching and RDF operations. The use of DSP provides the implementation of spectral analysis theory approaches, such as determining the parameters of the incident wave using signals from field sensors. Furthermore, DSP enables DOA algorithms, such as the MUSIC algorithms [31, 32].

2.4 Structure of RDF System

As shown in Figure (2.1), the main component of the RDF system is the antenna array system, radio receiver, DSP unit, and indicator for RDF results. In addition, other units can be added to the RDF diagram depending on the requirements, such as

a navigation system for estimating the RDF and correct location, remote control modules via cable lines or communication systems, and so on [32, 35].

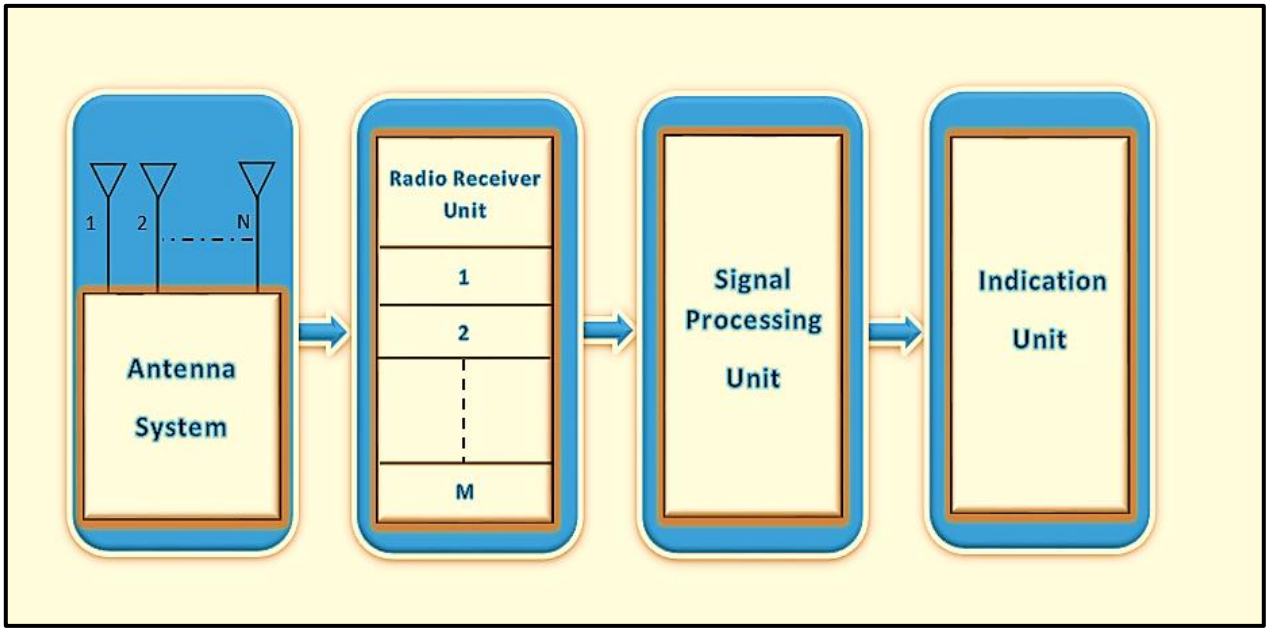


Figure (2.1) General Block Diagram of RDF System [32].

The radio receiving part is used to filter, amplify and the frequency conversion process of the input signals. The signals in the intermediate frequency move from the radio receiver output to the DSP unit, and they are subjected to Analog to Digital Converter (ADC). The DOA azimuth is estimated based on the obtained samples, according to the RDF method used [32, 35].

The digital signal-processing unit allows the avoidance of some shortcomings peculiar to analog direction finders. For example, digital processing provides the synchronization of receiving channels, correction of the phase and amplitude values for antennas, cables, etc [32, 35].

The indication unit's purpose is to show the RDF results. The RDF computing system manages the operation, saves databases with RDF and signals parameters for technical analysis, displays the RDF results and spectrum analysis, and makes reports about the overall process [32, 37].

2.4.1 Antenna Array System

Antenna systems are swiftly appearing as essential technologies to provide an improved function to develop the overall RDF system performance. Besides the researches in DOA algorithms, which allow the optimal performance of existing array geometries, it can be discovered array geometries with the best specifications [38, 39].

Multiple antennas are suitable for the RDF systems since they provide the DOA estimation of numerous almost impossible signals with a single antenna. Furthermore, using digital signal processing techniques, a smart antenna can estimate accurate DOA, improve immunity to noise and interference, enhance the RDF system behaviour, and minimize overall infrastructure costs [40, 41].

This section presents the most common antenna array geometries, each one with its configuration. The antenna array system has different parameters, such as the number of elements and inter-element spacing, which depends on the signal's wavelength. The analysis of the array geometries specifications is considered to significantly impact the DOA performance, such as radiation pattern and the time delay between elements [42].

Furthermore, instead of a single rotating antenna element, an antenna array is used because it is neither practical nor desirable from a mechanical aspect. The antenna array is configured with a specific geometry that includes crossed loop, monopole, and dipole elements. They are sampled electronically using electronic switches, a process is known as cyclic scanning. The separation between individual antenna elements must be less than half of the wavelength to produce accurate DOA estimates. The wavelength of the incoming signal should be used to adjust the spacing elements [42, 43].

It has the most basic common types of antennas arrays geometries are Uniform Linear Arrays (ULA), Uniform Rectangular Arrays (URA), Uniform Circular Arrays (UCA), and, which are presented in the following [44]:

2.4.1.1 Uniform Linear Array (ULA)

ULA is a set of antenna elements configured as a straight line on the axis, as shown in Figure (2.2). Each sub-element has a specific sub-spacing according to the frequency of the received signal. The first sub-element at the origin position becomes a reference position to the other sub-element arranged beside it [39, 42].

ULA is a conventional antenna array geometry considered the antenna arrays' base configuration. Furthermore, it is essential to study the impact of ULA on the performance of DOA algorithm estimation because of its simple structure and widely use. Some super-resolution DOA estimation algorithms currently operate only with ULA [39, 44].

However, it has many drawbacks: it can estimate angles only in the range from 0 ° to 180 ° and receive information with one-dimensional angular only . Furthermore, Figure (2.2) shows that the array elements are arranged in the x-axis to receive the signal from the reference element to the last element [39, 44].

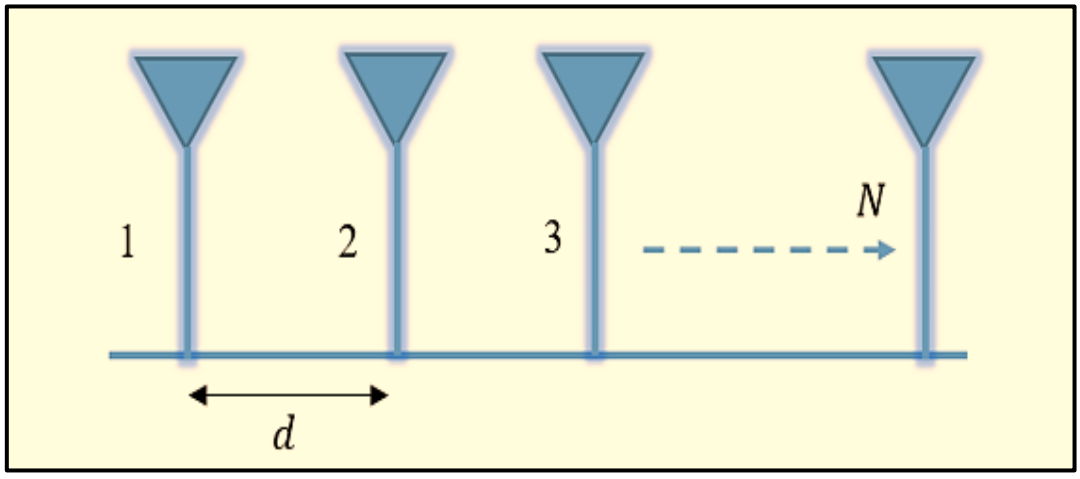


Figure (2.2) Uniform Linear Array [39].

2.4.1.2 Uniform Rectangular Array (URA)

URA is a set of antenna elements configured from two or more uniform linear sub-arrays put from the x-axis in the x-y plane, as shown in Figure (2.3). Each sub-element is equally spaced according to the wavelength of the incoming signal. The first element is placed at a specific point, known as a reference point, and other elements are arranged in a rectangular form [38, 42].

URA configuration needs a large number of elements to work correctly. For example, in Figure (2.3), where N is the number of elements on the x-axis, M is the number of elements on the y-axis, d is spacing between any two elements, the antenna array's rectangular configuration has a set of two ULAs located in parallel to the x–y plane [39]. URA is a widely used array geometry because it is a concentrated displacement type with many array elements placed in small plane space. Similarly, the smart antenna's performance can be enhanced by increasing the number of elements through a limited space [42,44].

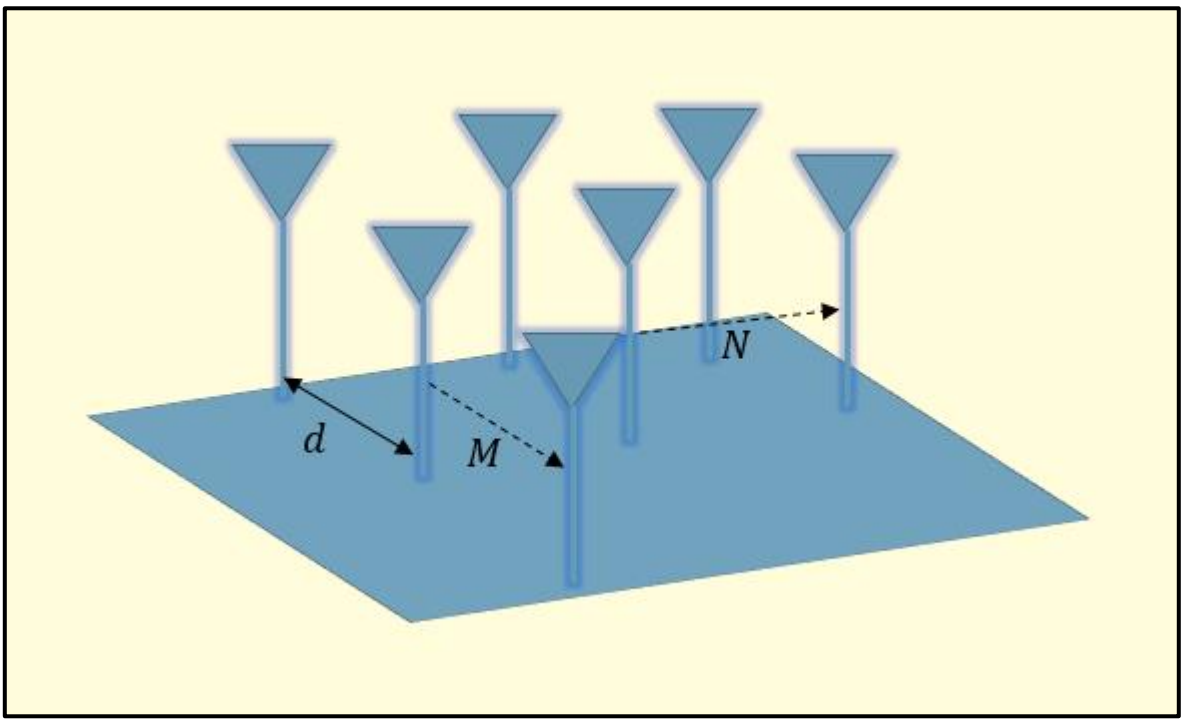


Figure (2.3) Uniform Rectangular Array [39].

2.4.1.3 Uniform Circle Array (UCA)

UCA is a set of antenna elements configured as a circle form on the y-x axis. Each sub-element has a specific sub-spacing according to the frequency of the received signal. The first sub-element is placed at a distance from the original position, representing a radius and becoming a reference to the other sub-elements arranged beside it [42, 44].

Figure (2.4) shows UCA elements forming a circle in the x-y plane and 3D directivity. It is profusely used because it has a good performance that can operate in two dimensions state and can prove a constant DOA estimation in the range from 0 ° to 360 ° to estimate the two-dimensional angular of RF source. However, UCA needs more number array elements for improving performance [44].

However, the calculation amount will be more significant, and structures will become more complex. Certainly, UCA is one of the most popular geometric shapes of arrays in studies because of the eclectic specifications. Furthermore, along the z-axis, UCA is isotropic, which implies that the azimuth cannot influence DOA estimation performance [39, 44].

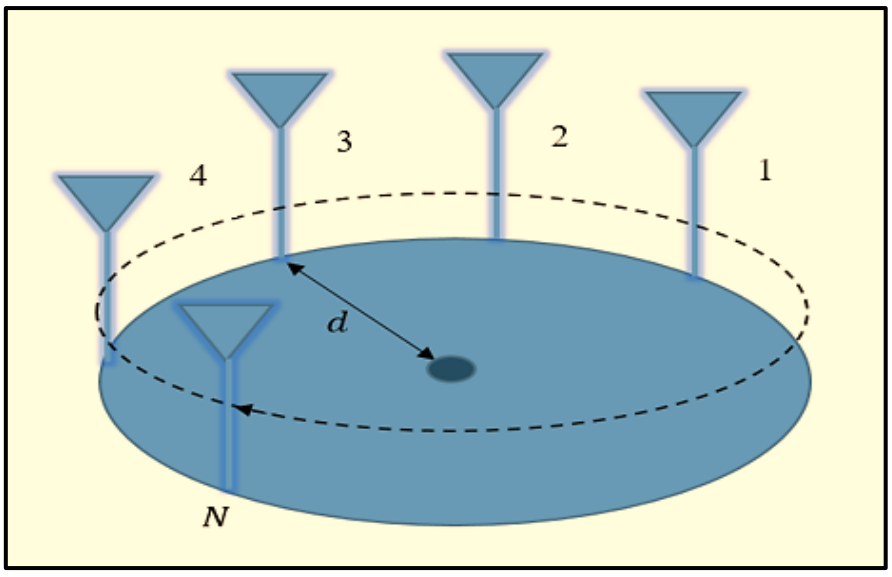


Figure (2.4) Uniform Circle Array [39].

2.4.2 Radio Receiver Unit

Software-Defined Radio (SDR) is a term in which the hardware components of a radio communication system such as modulators, mixers, filters, demodulators, detectors, amplifiers, as shown in Figure (2.5), are implemented by software on computer system [45]. A basic SDR system can include a computer with a sound card or ADC and an antenna. A design SDR can transmit and receive a variety of radio systems based on the utilized software [46].

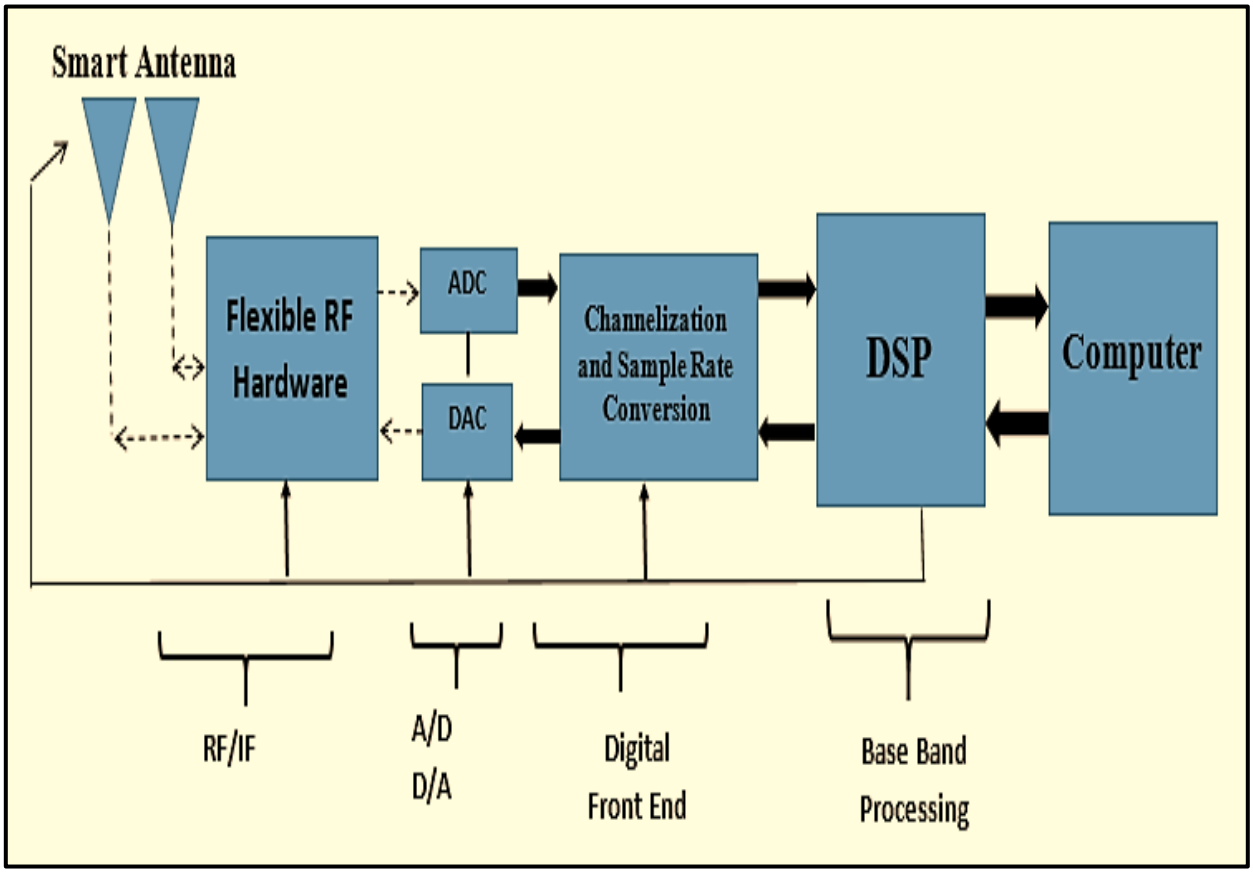


Figure (2.5) General Block Diagram of SDR [45].

The military and mobile phone applications can benefit significantly from SDR, which must handle a large varying of radio protocols. According to proponents such as the Wireless Innovation Forum, SDR is projected to become the most popular technology in wireless communications [47].

In ways, SDR can be flexible enough to analyze the "limiting spectrum" hypotheses made by designers for traditional radio systems. Moreover, SDR can be used for spread spectrum, and ultra-wideband allows transmitting on the different frequency at the various locations with minimal interference and applying error detection and correction techniques with merged it to rectify all errors caused by that diffident factors [47, 48].

SDR receivers provide immunity to interference by using programmable antennas that adaptively lock to a directional signal, neglecting other directions allowing them to identify low signals [45, 48].

SDR offers a digital solution for many radio systems because its platform allows for many uses: a signal receiver. Therefore, it is used as RDF through operating with multiple element arrays. In this thesis, SDR is used as an RDF receiver to determine the angle of arrival. Low-cost SDRs were used with some modifications to operate as a coherent receiver. RDF is created using a switching technique and multiple antenna arrays. The tracking algorithm is used to determine the DOA [46, 48].

SDR is considered an essential element in the RFST system because it allows the process of the RF signal by software and can operate with different frequency bands according to user requests and others. Therefore, all these specifications impact the construction of the RFST system [47, 48].

2.5 Direction of Arrival Estimation Techniques

As shown in Figure (2.6), DOA methods can design and adapt the directivity of array antennas to track the desired signal. For example, an antenna array can be designed to monitor the number of incident signals and take signals from specific directions only while rejecting other signals [49].

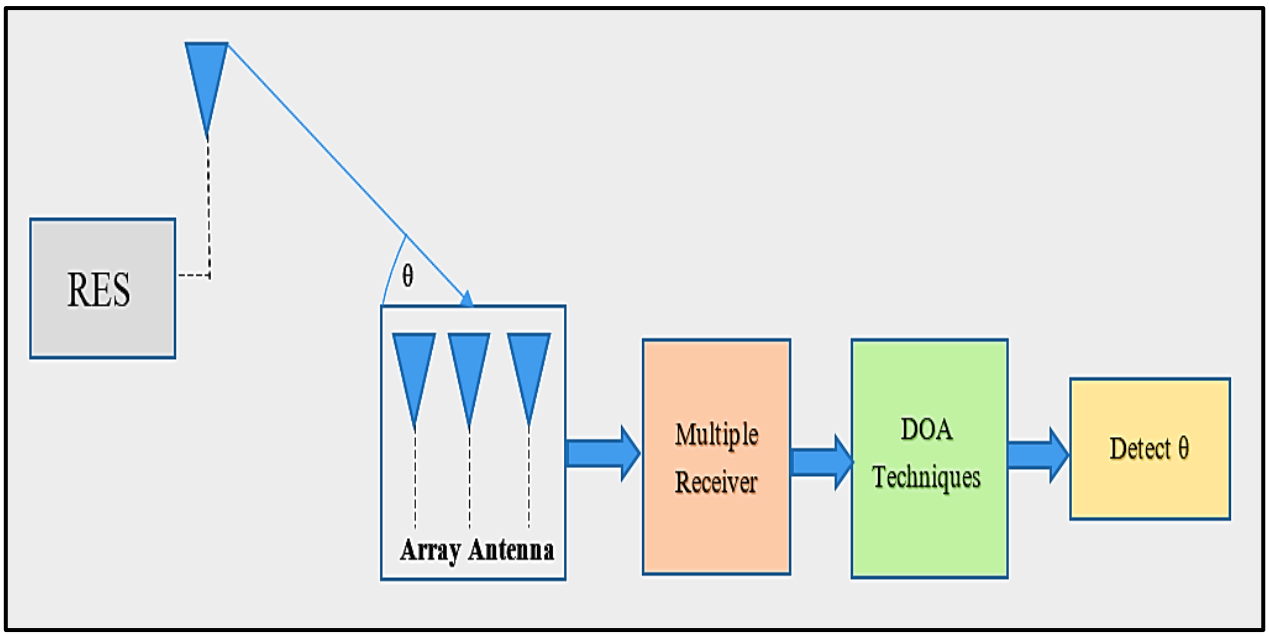


Figure (2.6) General Block of DOA Estimation Process [38].

DOA estimation methods are classified into two categories. The first group depends on the calculation of the spatial spectrum for the incoming signals and determining the DOA methods by specifying the spectrum peaks, which is called spectral-based algorithms [50]. It is also known as conventional methods, which have a powerful resolution because it is based on the physical size of the array aperture [50, 51].

The second group depends on the Eigen decomposition that directly estimates DOA methods without computing the spectrum. Therefore, it is known as parametric array processing techniques that as the performance increases, it becomes higher complexity and more computations with providing higher accuracy and resolution as compared with the first group [52].

2.5.1 The Received Signal Model

This model considers ULA with N sensors arranged as 1 to N and uniformly distance d between two elements. When the incoming signal is at the array elements, the steering with time can be defined by [53]:

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

s_ℓ represents a source signal component that is required to estimate its DOA, and f_c represents the carrier frequency [54]. The time delay is needed to arrive at the signal can be computed as:

$$\Delta t_m = \frac{md}{c}\sin\theta \tag{2.2}$$

c represents the speed of light that equal to $c = \lambda f_c$, λ represents the wavelength. The distance between array elements should be less than or equal to $d \geq \frac{\lambda}{2}$ for best performance [53, 54]:. Therefore, the signal received by the m th antenna element can be represented by:

$$x_m(t) = \mathrm{Re}\{s_\ell(t - \Delta t_m)e^{j2\pi f_c(t-\Delta t_m)}\} \tag{2.3}$$

Now, the signal is at down converted to become the baseband signal, which is:

$$x_m(t) = s_\ell(t - \Delta t_m)e^{-j2\pi f_c \Delta t_m} \tag{2.4}$$

The baseband signal is by T period, to obtain:

$$x_m(nT) = s_\ell(nT - \Delta t_m)e^{-j2\pi f_c \Delta t_m} \tag{2.5}$$

T also represents the symbol-period must be greater than the time-delay in one element of the array:

$$T >> \Delta t_m,\ m = 0, \ldots, N-1 \tag{2.6}$$

The approximation can be made on Eq. (2.5) to become:

$$x_m(nT) \approx s_\ell(nT)e^{-j2\pi f_c \Delta t_m} \tag{2.7}$$

Eq. (2.8) can be rewritten in the form of time index n to become:

$$x_m[n] \approx s_\ell[n]e^{-j\frac{2\pi}{\lambda}md\sin\theta} = s_\ell[n]a_m(\theta) \tag{2.8}$$

$$a_m(\theta) = e^{-j\frac{2\pi}{\lambda}Nd\sin\theta}, \text{ for } m = 0, \dots, N-1 \tag{2.9}$$

The n -th symbols of the ℓ -th signals can be represented by $s_\ell[n]$ for $\ell = 0,1, \dots, \mathcal{L}-1$. Hence, the sampled signals at the mth from antenna array can be written as:

$$x_m[n] \approx \sum_{\ell=0}^{\mathcal{L}-1} s_\ell[n]a(\theta_\ell) \tag{2.10}$$

2.5.2 Matrix Representation for Array Data

The scaling factors are collected in a vector of $N \times 1$, called the array steering vector in contexts of beamforming [54]. A matrix is configured by the column-vectors $\mathrm{a}(\theta_\ell)$, steering vectors of the signals $s_\ell(t)$:

$$\mathrm{A} = [\mathrm{a}(\theta_0) \dots \mathrm{a}(\theta_\ell) \dots \dots \mathrm{a}(\theta_{\mathcal{L}-1})] \tag{2.11}$$

$$\mathrm{a}(\theta_\ell) = \begin{bmatrix} a_0(\theta_\ell) \\ \vdots \\ a_{N-1}(\theta_\ell) \end{bmatrix} \tag{2.12}$$

The array output can be expressed written as:

$$\mathrm{x}_n = [\mathrm{a}(\theta_0)\mathrm{a}(\theta_1) \dots \mathrm{a}(\theta_{\mathcal{L}-1})]\mathrm{s}_n + \mathrm{n}_n = \mathrm{A}\mathrm{s}_n + \mathrm{n}_n \tag{2.13}$$

X_n is a vector that represents the sampled signal with N-dimensional, A has $N \times \mathcal{L}$ dimensional, s_n is a vector of signal, $n_m[n]$ is the vector of the noise defined at each element[54].

2.5.3 Eigen Structure of the Spatial Covariance Matrix

Assume that s_n and n_n are uncorrelated vectors, n_n is Gaussian noise that sampled and obtained zero mean with covariance matrix $\sigma^2\mathrm{I}$ [54]. Therefore, the spatial correlation matrix can be expressed as:

$$R = \mathbb{E}[\mathrm{x}_n\mathrm{x}_n^H] = \mathbb{E}[(\mathrm{A}\mathrm{s}_n + \mathrm{n}_n)(\mathrm{A}\mathrm{s}_n + \mathrm{n}_n)^H]$$

$$= \mathrm{A}\mathbb{E}[\mathrm{s}_n\mathrm{s}_n^H]\mathrm{A}^H + \mathbb{E}[\mathrm{n}_n\mathrm{n}_n^H]$$

$$= \mathrm{A}\mathrm{R}_{\mathrm{ss}}\mathrm{A}^H + \sigma^2\mathrm{I}_{N\times N} \tag{2.14}$$

R represents the Hermitian matrix that can be unitarily decomposed with real eigenvalues. For examining the R matrix eigenvectors with assuming N is large enough $N > \mathcal{L}$.

Any vector,q_n that is orthogonal with the columns of A, are eigenvectors of the spatial correlation matrix that can be expressed by modification the characteristic equation.

$$\mathrm{R}\mathrm{q}_n = (\mathrm{A}\mathrm{R}_{\mathrm{ss}}\mathrm{A}^H + \sigma^2\mathrm{I})\mathrm{q}_n = 0 + \sigma^2\mathrm{I}\mathrm{q}_n = \sigma^2\mathrm{q}_n \tag{2.15}$$

σ^2 is equal to the corresponding eigenvalue of q_n. The dimensions of A is $M \times \mathcal{L}$. Therefore, it will be $N - \mathcal{L}$ linearly independent vectors have eigenvalues that are equal to σ^2[54]. The noise subspace is the space spanned by the $N - \mathcal{L}$ eigenvectors. If $\mathrm{A}\mathrm{R}\mathrm{A}^H$ has q_s eigenvector, the characteristic equation can be expressed:

$$\mathrm{R}\mathrm{q}_s = (\mathrm{A}\mathrm{R}_{\mathrm{ss}}\mathrm{A}^H + \sigma^2\mathrm{I})\mathrm{q}_s = \sigma_s^2\mathrm{q}_s + \sigma^2\mathrm{I}\mathrm{q}_s = (\sigma_s^2 + \sigma^2)\mathrm{q}_s \tag{2.16}$$

Where R has q_s eigenvector with $(\sigma_s^2 + \sigma^2)$ eigenvalue, where $\mathrm{A}\mathrm{R}_{\mathrm{ss}}\mathrm{A}^H$ has σ_s^2 eigenvalue, the linear combination of A columns is $\mathrm{A}\mathrm{R}_{\mathrm{ss}}\mathrm{A}^H\mathrm{q}_s$, q_s available in the A column-space. R has $\mathcal{L}$ of linearly independent eigenvectors. Hence, the signal subspace is the space spanned by $\mathcal{L}$ vectors. Two subspaces are orthogonal to each

other: signal and noise subspaces [54]. Therefore, R can be written with the eigendecomposition as:

$$\mathrm{R} = \mathrm{QDQ}^{H} = [\mathrm{Q}_s \quad \mathrm{Q}_n] \begin{bmatrix} \mathrm{D}_s & 0 \\ 0 & \sigma^2 \mathrm{I} \end{bmatrix} [\mathrm{Q}_s \quad \mathrm{Q}_n]^{H} \tag{2.17}$$

The Q matrix is divided to Q_s matrix with $(N \times \mathcal{L})$, Q_s has columns that are configured by the $\mathcal{L}$ eigenvector represents the signal subspace, and Q_n matrix with $N \times (N - \mathcal{L})$ has columns representing the noise eigenvectors. The diagonal D matrix has the diagonal elements of the eigenvalues of R and is divided to $\mathrm{D}_s$matrix with $\mathcal{L} \times \mathcal{L}$ diagonal elements of the signal eigenvalues and $\sigma^2 \mathbf{I}_{N \times N}$ matrix with $(M - \mathcal{L}) \times (M - \mathcal{L})$ diagonal elements that are configured by noise eigenvalues of $N - \mathcal{L}$ [54].

2.5.4 Extrema-Searching Technique

The principle of this technique is based on the scan of the beam in the particular dimension with the measurement of the received power by the antenna array. Therefore, the DOA is estimated by determining the highest power peaks with a specific direction. This technique considers three different methods: Beamscan MVDR and MUSIC methods. Beamscan and MVDR are primarily beamforming-based, but MUSIC employs noise subspace to achieve high-resolution estimates. The methods of this technique are presented with ULA, where the incident signal y(n) is given in terms of the weighted sum that is collected by the antennas array [54]:

$$y[n] = w^{H} x[n] \tag{2.18}$$

2.5.4.1 Beamscan

The DOA is calculated using this method, which involves measuring signal intensity at each possible angle and picking the arrival angles at the power peak.

According to (19), the power peak in the beam will occur when weights w is equal to the steering vector. Therefore, the DOA has been estimated at the highest power point obtained [54]. The output power of the beamformer can be expressed by:

$$\begin{aligned} P_{\mathrm{BS}}(\theta) &= \mathbb{E}[|y(n)|^2] \\ &= \omega^H \mathbb{E}[\mathrm{x}(n)\mathrm{x}^H(n)]\omega \\ &= \omega^H \mathrm{R}\omega \end{aligned} \tag{2.19}$$

Based on Eq. (2.10), the average received power can be defined as:

$$\begin{aligned} P_{\mathrm{BS}}(\theta_0) &= \mathbb{E}[|\omega^H \mathrm{x}(n)|^2] \\ &= \mathbb{E}[|\omega^H [\mathrm{a}(\theta_0)\mathrm{s}(k) + \mathrm{n}(n)]|^2] \\ &= \left(|\omega^H \mathrm{a}(\theta_0)|^2 (\sigma_s^2 + \sigma_n^2)\right) \end{aligned} \tag{2.20}$$

$\mathbf{a}(\theta_0)$ represents the steering vector related to the angle θ_0, $\mathbf{n}(n)$ represents the noise vector, σ_s^2 is the signal power, σ_n^2 is noise power. When $w = a(\theta_0)$, the maximum average received power is obtained. Therefore, the antenna array will have the peak gain in the direction of θ_0 from all the possible weight vectors. As well as, the phases of the signal components in the sensors are aligned. In this technique, the searching process is implemented on all possible angles and the power is measured on all of them. Therefore, the power of Beamscan can be expressed as [54]:

$$P_{\mathrm{BS}}(\theta) = \omega^H \, \mathrm{R} \, \omega = \mathrm{a}^H(\theta) \mathrm{R} \, \mathrm{a}(\theta) \tag{2.21}$$

DOA is estimated by capturing the power peaks. Although a simpler method, the minimal effectiveness and performance can occur due to the width and height of the side lobes and signals from various sources are considered, a poor resolution will occur [54].

The improvement of this method resolution is achieved by enhancing the antenna array configuration. This will increase the $\mathrm{a}(\theta)$ elements and increase computational processing and complexity [54].

This method steps are presented in the following, where N represents the number of the elements of antennas, L represents the number of the sources, S represents the samples number, and P is the scan steps, where $\theta \in \{-90^{\circ} : 90^{\circ}\}$. Beamscan is presented in Algorithm 1 [54].

Algorithm 1: Beamscan Procedure	
1: $\hat{\mathrm{R}} = \frac{1}{S}\sum_{n=0}^{S-1} \mathrm{x}_n \mathrm{x}_n^H$	▹ Autocor. Matrix $-SM^2 + M^2$
2: $P_{\mathrm{BS}}(\theta) = \mathrm{a}^H(\theta)\mathrm{Ra}(\theta)$	▹ scan step $-M^2 + M$
3: Findpeaks	▹ estimating DOA $-4\mathcal{L}P$
	▹ Complexity: $M^2(S+2) + M + 4\mathcal{L}P$

2.5.4.2 MVDR

MVDR is standard for Capon's Minimum Variance Distortionless Response, which has a technique similar to the Beamscan approach by computing the received power of the incoming signals in all available directions. From the DOA, The power with the angle is estimated by limiting the actual direction beamformer gain to one and using the remaining freedom degrees to reduce the output power collective of incoming signals arriving from all available directions. The basic idea is that the cost function of the power must be minimal for each possible angle θ [54].

$$\min_{\mathrm{w}} \quad \mathrm{w}^H \mathrm{Rw} \qquad \mathrm{w}^H \mathrm{a}(\theta) = 1 \tag{2.22}$$

The received power of MVDR can be expressed:

$$P_{\mathrm{MVDR}}(\theta) = \frac{1}{\mathrm{a}^H(\theta)\mathrm{R}^{-1}\mathrm{a}(\theta)} \tag{2.23}$$

The disadvantage of the MVDR algorithm is the need to calculate the inverse matrix that may fail in case the highly correlated signals are coming. However, this method gives a higher resolution in the DOA estimation than the Beamscan method. MVDR is presented in Algorithm 2 [54].

Algorithm 2: MVDR Procedure	
1: $\hat{\mathbf{R}} = \frac{1}{S}\sum_{n=0}^{S-1}\mathbf{x}_n\mathbf{x}_n^H$	▹ Autocorr. Matrix - $SM^2 + M^2$
2: $\hat{\mathbf{R}}^{-1}$	▹ Matrix inverse $-4M^2$
3: $P_{\text{MVDR}}(\theta) = \frac{1}{\mathbf{a}^H(\theta)\hat{\mathbf{R}}_{\text{XX}}^{-1}\mathbf{a}(\theta)}$	▹ Angles scan $-M^2 + M$
4: Findpeaks	▹ Determine estimated DOA $-4\mathcal{L}P$
	▹ Total complexity: $M^2(S+6) + M + 4\mathcal{L}P$

2.5.4.3 MUSIC

MUSIC is an abbreviation for Multiple Signal Classification. It is a DOA technique classified as one of the subspace methods that was recently proposed and a widespread method. It is a high-precision resolution method for DOA estimation, which indicates the incoming signals' number and directions of arrival. MUSIC is the DOA mothed that depends on Eigen analysis, which is exploited to analyze the covariance matrix into two orthogonal subspaces: signals subspace and noise subspaces [54, 55].

Therefore, the arriving signals have the signal subspace that contains the steering vector orthogonal with the noise subspace. Eigenvectors can be produced by the variance matrix analysis or the Singular Value decomposition (SVD) analysis in the covariance matrix. One of the MUSIC method features calculates the powers and

cross-correlations among the incoming signals. The set of DOA of multiple incoming signals is estimated by determining the peaks of the MUSIC algorithm [54].

$\mathrm{a}(\theta)$ represents the steering vector of the one incident signal, so, $\mathrm{a}^H(\theta)\mathrm{Q}_n = 0$, Q_n matrix represents the noise subspace. a (θ) are not precisely orthogonal with the noise subspace because of estimating errors of Q_n [49, 54].

$$P_{\mathrm{MUSIC}}(\theta) = \frac{1}{\mathrm{a}^H(\theta)\mathrm{Q}_n\mathrm{Q}_n^H\mathrm{a}(\theta)} \tag{2.24}$$

Nevertheless, the function proved the most significant value in the case of θ is equal to the angle of arrival associated with one of the signals. In the implementation of MUSIC, it firstly estimates a based on Q_nAnd then finds the L peaks in (25); the related angles prove the angle of arrival estimation [54].

The MUSIC algorithm is widely used due to its high resolution. It provides a higher DOAs selection for multiple signals with a good resolution better than 2° and low complexity. Nevertheless, MUSIC sometimes has obstacles such as it wants an initial detection of the system responses, knowledge of noise and interference and the number of signal sources. MUSIC is presented in Algorithm 3 [54].

Algorithm 3: MUSIC Procedure	
1: $\hat{\mathbf{R}} = \frac{1}{S}\sum_{n=0}^{S-1}\mathbf{x}_n\mathbf{x}_n^H$	▹ Autocorr. Matrix - $SM^2 + M^2$
2: $\hat{\mathbf{R}} = \mathbf{QDQ}^H$	▹ Eigendecomposition - $\frac{2M^3}{3}$
3: $\mathbf{Q}_n\mathbf{Q}_n^H$	▹ Eigenvectors multiplication - $M^3 + M^2\mathcal{L}$
4: $P_{\mathrm{MUSIC}}(\theta) = \frac{1}{\mathbf{a}^H(\theta)\mathbf{Q}_n\mathbf{Q}_n^H\mathbf{a}(\theta)}$	▹ Angles scan- M^2M
5: Findpeaks	▹ Determine estimated DOA $-4\mathcal{L}P$

2.5.5 Matrix Shifting Techniques

The primary method based on the matrix shifting technique is ESPRIT, one of the most extensively used methods for estimating DOA. As mentioned, the noise subspace is used in the MUSIC approach, while the signal subspace is used in conjunction with a rotational variance technique in ESPRIT. Estimations are obtained using neither nonlinear optimization nor spectral measure search so that a complexity lower than the extrema searching techniques, scanning for all possible DOA [54].

2.5.5.1 Conventional ESPRIT

ESPRIT is defined as the Estimation of Signal Parameters via the Rotational Invariance Technique. It can be classified as one of the subspace techniques used to estimate the DOA.

Roy and Kailath first proposed the ESPRIT method to estimate DOA. The advantage of ESPRIT is reducing the calculating power and requiring less memory storage. As well as, it does not need a heavily search over all possible steering vectors. The main idea of the ESPRIT method is to research the function of rotational invariance in the signal subspaces formed by two sub-arrays with a translation invariance structure derived from the original array [54, 55, 56].

The ESPRIT operates on the antenna array with N elements, separated into subarray as shown in Figure (2.7). The doublets can be separated to form two subarrays with N elements each. The distance d may be different from Δ [54, 55, 56]. The x_1 and x_2 sub-arrays outputs are expressed as:

$$x_1[n] = \sum_{\ell=0}^{\mathcal{L}-1} s_\ell[n] a(\theta_\ell) + n_{x_1}[n] \tag{2.25}$$

$$x_2[n] = \sum_{\ell=0}^{\mathcal{L}-1} s_\ell[n] e^{j\frac{2\pi}{\lambda}\Delta\sin(\theta_\ell)} a(\theta_\ell) + n_{x_2}[n] \tag{2.26}$$

Where x_1 and x_1 vectors with N × 1, n_{x_1} and n_{x_2} represent the noise vector at the two subarrays. The subarrays output of x_1 and x_2 can be written as:

$$x_1 = As + n_{x_1} \tag{2.27}$$

$$x_2 = A\Phi s + n_{x_2} \tag{2.28}$$

The output vector of the entire array is written as:

$$x[n] = \begin{bmatrix} x_1[n] \\ x_2[n] \end{bmatrix} = \begin{bmatrix} A \\ A\Phi \end{bmatrix} s[n] + \begin{bmatrix} n_1[n] \\ n_2[n] \end{bmatrix} = Q_s s[n] + n[n] \tag{2.29}$$

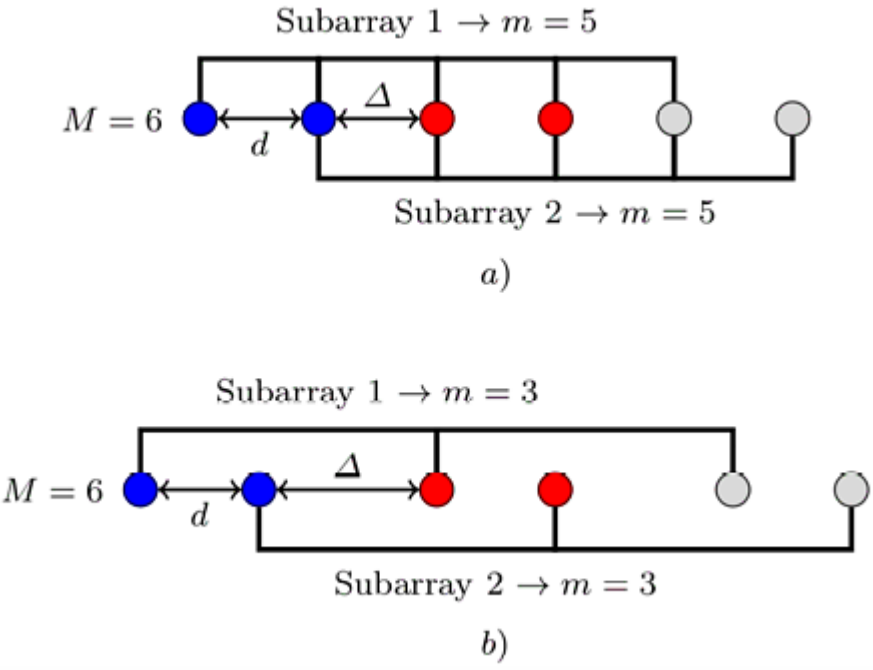


Figure (2.7) Two Of ESPRIT Subarrays Formation Using M = 6 Antenna Elements: a) Equidistant Array with 3 Equidistant Identical Doublets and $d = \Delta$. b) Array with 3 Non-Equidistant Identical Doublets and $d \neq \Delta$ [54].

The Q_s is used to estimate the Φ diagonal elements without A. Hence, $Q = \{Q_s \quad Q_n\}$ is determined from the R Eigen-decomposition [54]. When

E_s matrix has columns based on the signal subspace that corresponds to the x vector, so E_s and Q_s are associated by T transformation with $L \times L$ can be expressed by:

$$E_s = Q_s T = \begin{bmatrix} AT \\ A\Phi T \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \end{bmatrix} \tag{2.30}$$

The subspace of E_1 and E_2 are similar to A. Hence, E_1 and E_2 with A have the same bonds. a nonsingular matrix Ψ with L × L can be expressed as:

$$E_1 \Psi = E_2 \tag{2.31}$$

Ψ can be expressed by:

$$\begin{aligned} AT\Psi &= A\Phi T \\ AT\Psi T^{-1} &= A\Phi \\ \Psi &= T^{-1}\Phi T. \end{aligned} \tag{2.32}$$

The Ψ eigenvalues should equal the Φ diagonal elements, and T columns are the Ψ eigenvectors. This is the most critical relationship in the evolution of ESPRIT and its features [54].

Hence, the ϕ_ℓ eigenvalues of Φ are computed, DOA can be calculated as:

$$\phi_\ell = e^{j\frac{2\pi}{\lambda}\Delta\theta_\ell},\ \ell = 1 \dots \mathcal{L} \tag{2.33}$$

$$\theta_\ell = \arcsin\left(\frac{\lambda \cdot \arg(\phi_\ell)}{2\pi\Delta}\right) \tag{2.34}$$

$$\arg(\phi) = arctan\left(\frac{Im(\phi)}{Re(\phi)}\right) \tag{2.35}$$

The ESPRIT- based on E_1 and E_2 signal subspace, determine Ψ, calculate the Φ eigenvalues, and finally estimate the DOA. ESPRIT-DOA is presented in Algorithm 4 [54].

Algorithm 4: ESPRIT Procedure	
1: $\hat{\mathrm{R}} = \frac{1}{S}\sum_{n=0}^{S-1} \mathrm{x}_n \mathrm{x}_n^H$	▹ Autocorr. Matrix - $SM^2 + M^2$
2: $\hat{\mathrm{R}} = \mathrm{QDQ}^H$	▹ Eigendecomposition - $\frac{2M^3}{3}$
3: $\Psi = \mathrm{T}\Phi\mathrm{T}^H$	▹ Eigendecomposition - $\frac{2M^3}{3}$
4: $\mathrm{E}_1\Psi = \mathrm{E}_2$	▹ Matrix multiplication - $\frac{\mathcal{L}^2}{2} + \frac{\mathcal{L}}{2}$
5: Solve ϕ	▹ Eigenvalues $- \frac{2M^3}{3}$
6: $\theta_k = \arcsin\left(\frac{\lambda \cdot \arg(\phi_k)}{2\pi\Delta}\right)$	▹ Find DOA - $\frac{\mathcal{L}^2}{2} + \frac{\mathcal{L}}{2}$
	▹ Total complexity: $M^2\left(S + \frac{6M}{3} + 1\right) + \mathcal{L}(\mathcal{L}+1)$

2.5.5.2 Beamspace-ESPRIT

Beamspace-ESPRIT is a technique used for the reduction of computational complexity. It originates from the ESPRIT algorithm to solve the problem of subspace DOA estimation by reducing dimensions in beamspace. The Beamspace-ESPRIT algorithm drawback is that it always needs prior knowledge of the sector where the signals are lying to the center position of the output beam fan. Nevertheless, this technique provides excellent performance and is less computational [54 ,55, 56].

Π_p Exchanging matrix $p \times p$ with antidiagonal of ones and zeros other where:

$$\Pi_p = \begin{bmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{bmatrix} \in \mathbb{R}^{p \times p} \tag{2.36}$$

Furthermore, M complex matrix is known as centro-Hermitian if

$$\Pi_p \mathrm{M}^* \Pi_q = \mathrm{M} \tag{2.37}$$

Q matrix can be written:

$$\Pi_p \mathrm{Q}^* = \mathrm{Q} \tag{2.38}$$

The unitary sparse is specially set left -real matrices are shown as Q_p.

$$\mathrm{Q}_{2n} = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathrm{I}_n & j\mathrm{I}_n \\ \Pi_n & -j\Pi_n \end{bmatrix} \tag{2.39}$$

$$\mathrm{Q}_{2n+1} = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathrm{I}_n & 0_{n\times 1} & j\mathrm{I}_n \\ 0_{n\times 1}^T & \sqrt{2} & 0_{n\times 1}^T \\ \Pi_n & 0_{n\times 1} & -j\Pi_n \end{bmatrix} \tag{2.40}$$

Q_{2n} even and Q_{2n+1} odd order. The selection matrix can write the shift-invariance property associated with the Beamspace-ESPRIT:

$$\mathrm{J}_2 = [0_{m\times 1} \quad \mathrm{I}_m] \tag{2.41}$$

N is the element number of the subarrays. Therefore, the real transformations of the selection matrix can be expressed as:

$$\mathrm{K}_1 = 2\mathrm{Re}\{\mathrm{Q}_m^H \mathrm{J}_2 \mathrm{Q}_M\} \tag{2.42}$$

$$\mathrm{K}_2 = 2\,\mathrm{Im}\ \{\mathrm{Q}_m^H \mathrm{J}_2 \mathrm{Q}_M\} \tag{2.43}$$

Beamspace-ESPRIT is composed of three steps:

1) Estimating of the real subspace.

2) Least-squares solution of the problem.

3) Final decomposition of eigenvalues (EVD).

Beamspace-ESPRIT is presented in Algorithm 5 [54].

Algorithm 5: Beamspace- ESPRIT Procedure	
Input: X and $\mathcal{L}$.	
1: Compute $\mathrm{X}_{\mathrm{fba}}$	▹ $2M^3 + MS^2$
2: $\widehat{\mathbf{R}} = \frac{1}{S}\sum_{n=0}^{S-1}\mathbf{X}_{\mathrm{fba}} \cdot \mathbf{X}_{\mathrm{fba}}^{H}$	▹ $SM^2 + M^2$
3: $\mathcal{T} = \mathrm{Re}\{\mathbf{Q}_M^H \widehat{\mathbf{R}} \mathbf{Q}_M\}$	▹ $2M^2$
4: Compute $\widehat{\mathrm{E}}_{\mathrm{s}}$ from SVD of $\mathcal{T}$	▹ $\frac{2M^3}{3}$
5: Compute K_1 and K_2	▹ $4M^2$
6: $\mathbf{K}_1\widehat{\mathrm{E}}_{\mathbf{s}}\Gamma \approx \mathbf{K}_2\widehat{\mathrm{E}}_{\mathrm{s}}$	▹ $2M^2$
7: $\hat{\Gamma}_{\mathrm{LS}} = \left(\mathrm{K}_1\widehat{\mathrm{E}}_{\mathrm{s}}\right)^{\dagger}\left(\mathrm{K}_2\widehat{\mathrm{E}}_{\mathrm{s}}\right)$	▹ $M^3 + 2M^2 + M$
8: Solve the eigenvalues ϕ of $\hat{\Gamma}_{\mathrm{LS}}$	▹ $\frac{2M^3}{3}$
9: $\theta_k = \arcsin\left(\frac{2\cdot\arctan(\phi_k)}{\frac{2\pi\Delta}{\lambda}}\right)$	$\frac{\mathcal{L}^2}{2} + \frac{\mathcal{L}}{2}$
	▹ Total complexity: $\frac{13M^3}{3} + 11M^2 + M(S^2 + S + M + 1) + \frac{\mathcal{L}^2}{2} + \frac{\mathcal{L}}{2}$

2.5.6 Polynomial-Rooting Techniques

Many enhancements and adjustments to DOA algorithms have been proposed to increase resolution while reducing complexity. Polynomial-rooting approaches, which use a polynomial parameterization to estimate DOA, are advancements where the estimated angles are the roots of a polynomial. ROOT-MUSIC and ROOT-WSF use this technique [54].

2.5.6.1 ROOT-MUSIC:

MUSIC algorithm has developed by adding several modifications and many improvements to become its performance more accurate and less the calculation complexity. Therefore, it is called Root-Music, because it utilizes Polynomial-rooting techniques and polynomial parameterization to estimate DOA, developed by Barbell [54].

Root-Music uses the roots and properties of the signal Eigenvector inside the unit circle to estimate the DOA of the sources and know the rational spectrum function with enhanced resolution possibility. In comparison cases, the root music is selected over many DOA techniques because it is immediately derived from MUSIC techniques, which provides a higher performance-complexity tradeoff. However, it is used only for a uniform spaced linear array [54].

The ROOT-MUSIC method is based on a polynomial rooting approach and offers better resolution than the traditional MUSIC. As previously expressed in Eq. (23), the conventional MUSIC is expressed [54, 56, 57]:

$$\mathrm{P}_{\mathrm{MUSIC}}(\theta) = \frac{1}{\mathrm{a}^{H}(\theta)\bar{\mathrm{A}}\,\mathrm{a}(\theta)} \tag{2.44}$$

Where $\overline{\mathrm{A}} = \mathrm{Q}_n \mathrm{Q}_n^H$. Rewriting (2.40), we can obtain

$$\mathrm{P}_{\text{R-MUSIC}}^{-1} = \sum_{m=1}^{M} \sum_{n=1}^{M} e^{-j\frac{2\pi}{\lambda} m d \sin\theta} \overline{\mathrm{A}}_{mn} e^{j\frac{2\pi}{\lambda} n d \sin\theta} \tag{2.45}$$

$$\mathrm{P}_{\text{R-MUSIC}}^{-1} = \sum_{l=-M+1}^{M-1} \bar{a}_l e^{-j\frac{2\pi}{\lambda} l d \sin\theta} \tag{2.46}$$

$\bar{a}$ is the entries sum of $\overline{\mathrm{A}}$ with l th diagonal:

$$\bar{a}_l \triangleq \sum_{m-n=l} \overline{\mathrm{A}}_{mn} \tag{2.47}$$

$G(z)$ polynomial is defined with $2(M+1)$ order :

$$G(z) = \sum_{l=-M+1}^{M+1} \bar{a}_l z^{-l} \tag{2.48}$$

Root-MUSIC is described in Algorithm 6. The $G(z)$ is resolved by a unit circle. MUSIC can be evaluated [54, 57]. The $G(z)$ roots close to the unit circle are corresponding to peaks of MUSIC, the $G(z)$ pole at $z = z_l = |z_l| e^{j\arg(z_l)}$:

$$\theta_l = \arcsin\left(\frac{\lambda \cdot \arg(z_l)}{2\pi d}\right) \tag{2.49}$$

Algorithm 6: Root-MUSIC Procedure	
1: $\hat{\mathbf{R}} = \frac{1}{S}\sum_{n=0}^{S-1} \mathbf{x}_n \mathbf{x}_n^H$	▹ Autocorr. Matrix $-SM^2 + M^2$
2: $\hat{\mathbf{R}} = \mathrm{QDQ}^H$	▹ Eigendecomposition - $\frac{2M^3}{3}$
3: $\overline{\mathbf{A}} = \mathrm{Q}_n \mathbf{Q}_n^H$	▹ Eigenvectors multiplication - $M^3 + M^2\mathcal{L}$
4: Solve $G(z)$	▹ Find polynomial roots - $2M^2(M-1)$
5: $\theta_l = \arcsin\left(\frac{\arg(z_l)}{\frac{2\pi d}{\lambda}}\right)$	▹ Find angles $-2(M-1)$
	▹ Total complexity: $\frac{11}{3}M^3 + M^2(S+\mathcal{L}-1) + 2(M-1)$

2.5.6.2 ROOT-WSF

ROOT-WSF is the abbreviation for Weighted Subspace Fitting (WSF) algorithm that is one important of array signal processing methods. The simple idea of WSF depends on the orthogonality of signal subspaces and noise subspaces to estimate the DOA of signals. It is considered an asymptotically adapted parametric technique. WSF has one important advantage that removes coherency among the signal sources so that it is effective for incoherence signals and coherent signals with increasing computation. In WSF, the decomposition technique is used for eigenvalues as the best eigenvectors through a diagonal matrix and the associated eigenvectors through the signal subspaces matrix.WSF algorithm can be expressed in the following [59]. The matrix of covariance R is recalled:

$$R = APA^{\mathrm{H}} + \sigma^2 I = U_s \Lambda_s U_s^H + \sigma^2 U_n U_n^H \tag{2.50}$$

The steering matrix is $A(\theta)$, P is the signal statistical expectation and Λ eigenvalues matrix. By making $I = U_s U_s^H + U_n U_n^H$ and $\sigma^2 U_n U_n^H$ is removed from both sides by multiplying the equation byU_s:

$$APA^{\mathrm{H}} U_s + \sigma^2 I = U_s \Lambda_s I \tag{2.51}$$

Rearranging the equation:

$$U_s = AT \tag{2.52}$$

The angles are calculated by solving the nonlinear equation below. :

$$\hat{\theta} = \arg\left\{\mathrm{m}\ \ \mathrm{Tr}\left\{\left((I - A(A^H A)^{-1} A^H)\hat{U}_S W \hat{U}_S^H\right)\right\}\right\} \tag{2.53}$$

$$W = \left(\widehat{\Lambda}_s - \hat{\sigma}^2 I\right)^2 \widehat{\Lambda}_s^{-1} \tag{2.54}$$

Where m is the minimum value of real transformation matrix Tr. ROOT-WSF is a Root weighted subspace fitting algorithm that is another mothed for DOA estimation [59].

2.6 RFST System Principle

RFST system has an essential function; it will be impossible to tackle radio environment tracking issues without it. RFST procedures can be described in the following points [60]:

- Real time monitoring and localizing of RES.
- Fasting scan for any new RES and measurement of its properties.
- Databases are created, replenished, and adjusted flexibly.
- Radio channel tracking, demodulated recording and listening, and technical radio signal analysis.
- Measurement of standard radio parameters and RSS of new RES.
- RDF and locating RES.
- Monitoring the licensed broadcasting with its assigned frequency band.
- Monitoring the bands of frequency and calculating the channel occupancy.
- Monitoring the interference of the radio sources.
- Suppression of illegal RES activity.

The above tasks are required to solve radio-electronic intelligence problems critical to law enforcement agencies and the military. The issues related to radio tracking are typically solved by federal and civil agencies charging with regulating the radio frequency spectrum usage. However, the fundamental purpose of radio intelligence is to detect radio emissions, determine the geographic locations of their

sources, estimate the type of modulation and assess its parameters, and code radio transmissions [60].

In reality, RFST can serve the interests of the citizens and government agencies by recognizing RES, determining its location, and determining its significant features. The following feature of the RFST [32, 60]:

- Frequency analysis
- Bandwidth analysis
- Identifying the type of modulation and measuring its parameters.
- Spectrum occupation analysis.
- RDF and coordinate estimation.
- Measurements of RSS and power density.

The differences between RF spectrum regulation and radio-electronic intelligence according to the task's aims and depth. If the interception of the civil services is dangerous, monitoring is the essential requirement for RES recognition [32, 60].

2.6.1 RFST Systems Concepts

The required number of the radio monitoring stations is computed by area size, the zone topography, affiliations and the finance possibility. In the perfect case, any points of territory monitoring should be in the operations area, at least two RDF, to find RES coordinates. The operational frequency range must encompass 9 kHz to 3 GHz, with exceptions up to 18 or even 40 GHz. However, the number of posts and system costs are unacceptably raised in this circumstance [60].

As a result, the more favorite approaches to the RFST which is when there are permanent stations for action zones in densely populated areas, mobile stations mounted on the ground, airplane or the water transportation carriers, and mobile stations, which can be deployed quickly in the desired areas, including in difficult-to-reach locations [32].

For RES coordinates estimation, the systems that meet the conditions mentioned above can be used on one of two well-known approaches [32, 60] :

- The method of the goniometrical (with the help of the RDF).
- The way of the Difference distance measuring (systems of the hyperbolic).

2.6.1.1 Goniometrical Method

RDF is a part of the construction of the goniometric systems for the RES locating estimation, as shown in Figure (2.8). Because the exact location of the RES is uncertain, two or more RDF are utilized, each in a separate part of the systems activity zone. None of the observation stations requires exact synchronizing in time with other stations. Determining the direction of quasi-continuous RES is an undeniable advantage of goniometric systems.

Other advantages of the systems are the minimal amount of data transferred from station to station to calculate RES location coordinates. On the other hand, the efficiency is decreased to a minimal three monitoring stations, whereas the partial efficiency was maintained until two observation stations [60]. The major shortcoming for these systems links with the suitable RDF shortcomings [60]:

- The estimation of RES coordinates Errors depends on the RES in the limitation of the coverage area of the system, RDF mutual locations.

- The cost of RDF was relatively high, especially in antennas with wide ranges with a coverage factor frequency of more than ten.

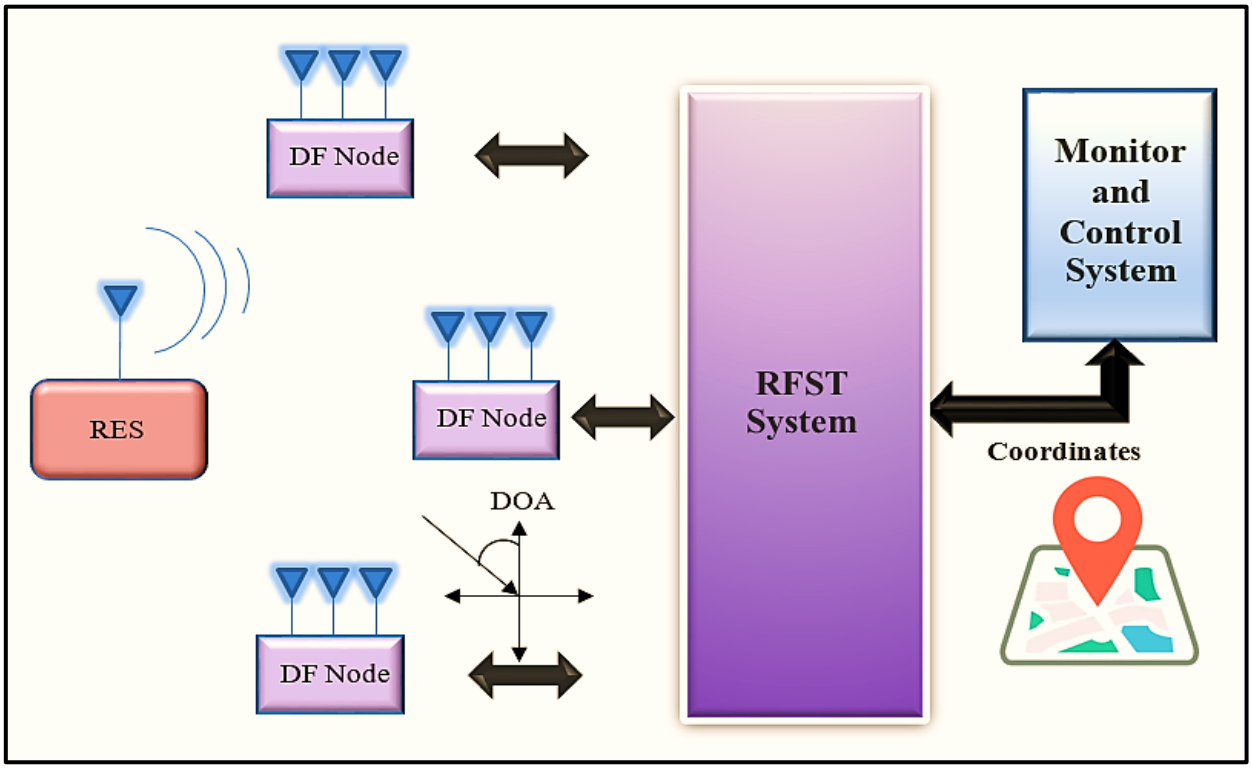


Figure (2.8) Block Diagram of RFST System [32].

The type of modulation and the width of the RES spectrum can essentially impact the accuracy of the RDF. This effect is almost non-existent in the multiple-channel interferometers, but it is significant in quasi-Doppler RDF [60].

The DOA Techniques is based on vectors ranging and uses a principle of triangulation to compute the RF location target. To determine the received radio signals, this approach needs at least two nodes RDF direction. [60]. The intersection of drawn lines from nodes of the RDF to the RF emitter represents the RF transmitter.

When two vectors meet in a point, they form a single angle. The RES position can be calculated using the estimated angle of arrival and other relevant data. RDF stations with known coordinates must be utilized as a reference to get precise computations of the essential information in this technique. [61].

Simple trigonometric formulas are used in the triangulation algorithm, as shown in Figure (2.8). Assume the sensor arrays are in 2-D coordinates, with the distance L between unknown RES location (x, y) the RDF station. Then, knowledge of the location of RDF stations and the two directions of arrival at θ1, θ2 can be found based on the coordinates (x, y) [61].

$$x = \frac{L}{(\tan \theta 1 + \tan \theta 2)} \tag{2.55}$$

Then for *y*:

$$y = x \tan \theta 1 \tag{2.56}$$

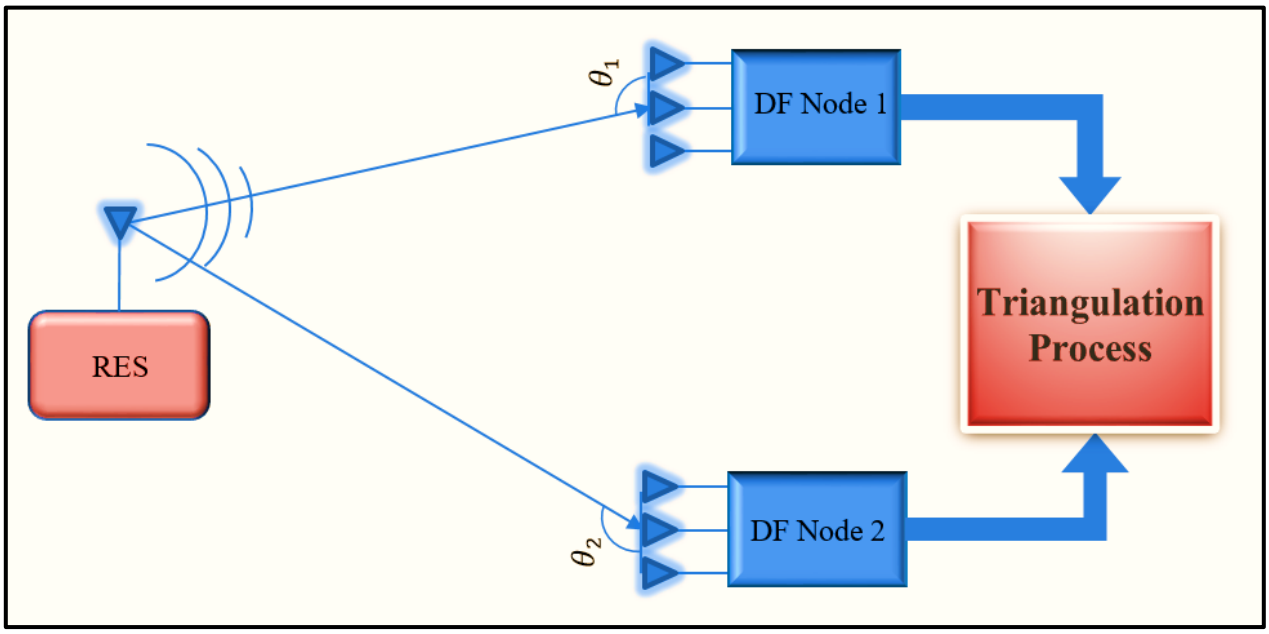


Figure (2.9) Block Diagram of Triangulation Process [61].

2.6.1.2 Time Difference of Arrival (TDOA)

RES coordinate was computed by measuring the time delay during the radio signal propagations from the RES to observations stations, as shown in Figure (2.10). The independence of outcomes from antennas types employed and the polarization of the radio wave type is characteristic for such systems. The best results are usually obtained by using a non-directional antenna. The fundamental advantage of goniometric systems is their TDOA feature. The difference in delay time between the two observations determines the RES location. The hyperbola depends on focusing on observation stations locations [60].

As a result, TDOA systems are sometimes known as hyperbolic systems. Thus, the TDOA station number must be at least three, however in this case, two (or more) hyperbolic equation system solutions may emerge at RES locations in some geographical area. Therefore, the addition of the fourth observation station just inserts the system's needed adequacy and removes the uncertainty [60].

The main disadvantage of TDOA is it’s impossible to compute the non-modulated wave source locations. The RES locations accuracy is based on its modulation. The better results for the RES can be accomplished with the function of fast-falling autocorrelation of the modulating wave. Unlike the goniometric systems for the RES location estimation, TDOA requires time synchronization between all stations with a precision of 10^{-8}s. The equipment becomes more complicated because of the synchronization [60].

Unlike an RDF, whose operation results may be the azimuth value, a signal sample is the outcome of a single observation station in the TDOA system. The signal samples are sent from all observation points to the total station for the coordinate’s calculations, determining the proper delays and the RES location. As a result, the

volume of information's conveyed from the observation station to the coordinates computation station in the TDOA system can be more than in a goniometric system.

Therefore, it is preferred to make goniometric systems based on RDF are favorite for implementing geographically distributed systems for the RES location estimation with the arbitrary modulation type [60].

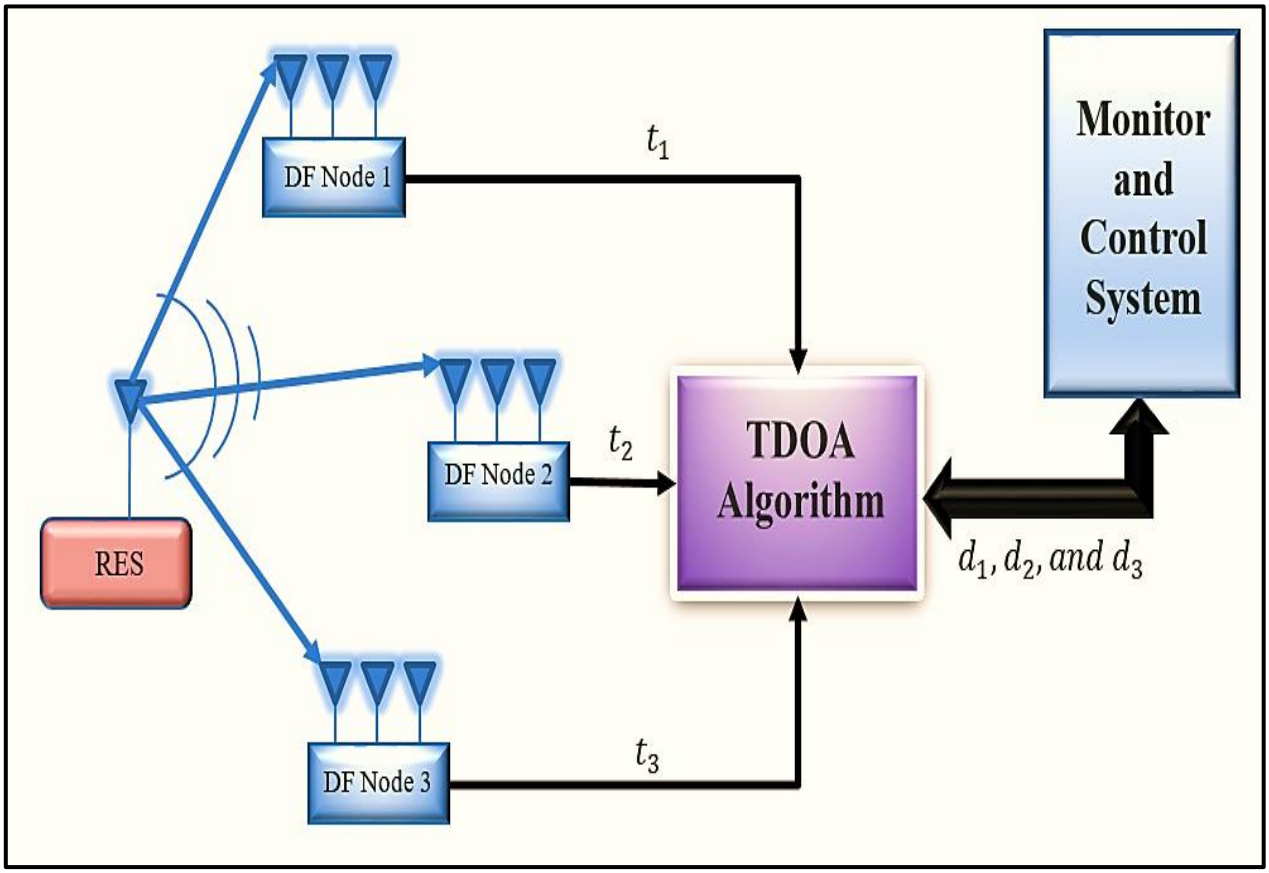


Figure (2.10) Block Diagram of TDOA [32].

2.6.2 RFST Techniques

This section discusses the organization of the RFST using the following system as an example: stationary RFST, mobile RFST, portable RFST, and combined RFST.

2.6.2.1 Stationary RFST

This technique is designed for prominent cities and industrial areas, as shown in Figure (2.11). It is made up of permanent stations with antenna systems installed at high altitudes and on the roofs of tall structures.

The control center communicates with the periphery stations through wires or radio channels. Permanent peripheral stations are typically remotely managed and do not need a Stationary operator presence [60].

Figure (2.11) Stationary RFST [60].

2.6.2.2 Mobile RFST System

The movable central and peripheral stations are installed on mobile carriers in this arrangement, as shown in Figure (2.12). A radio channel controls the mobile

peripheral stations from the central mobile station. The RFST system can swiftly modify its operation zone due to the radio equipment's location on the mobile carrier, ensuring the timely completion of RFST actions and the identification of RES location [60].

Mobile equipment effectively expands the tasks ranges for the RES locations estimation and the radio monitoring, making tasks that would be difficult or impossible to do with fixed equipment possible. RES detection and parameters are monitored using transmitting antennas and estimation of the operation zone of the mobile radio communications systems, the precise RES location estimating under conditions of range overloading, and parameter monitoring and RES signal detection using antennas directed transmitting [60].

Figure (2.12) Mobile RFST System [60].

2.6.2.3 Geographically Distributed RFST System

This system, which consists of easily deployed posts, is designed to automatically estimate the RES location and RFST, as shown in Figure (2.13). This station radio equipment belongs to the portable multifunctional equipment (PMFE) family, designed for many operators to travel by hand.

Static or temporary stations with electrical power sources and at an open site, similar equipment can be employed. This method is intended for RFST, simultaneous RDF, and locating the RES. A portable RDF comprises a central station and one or more peripheral stations [60].

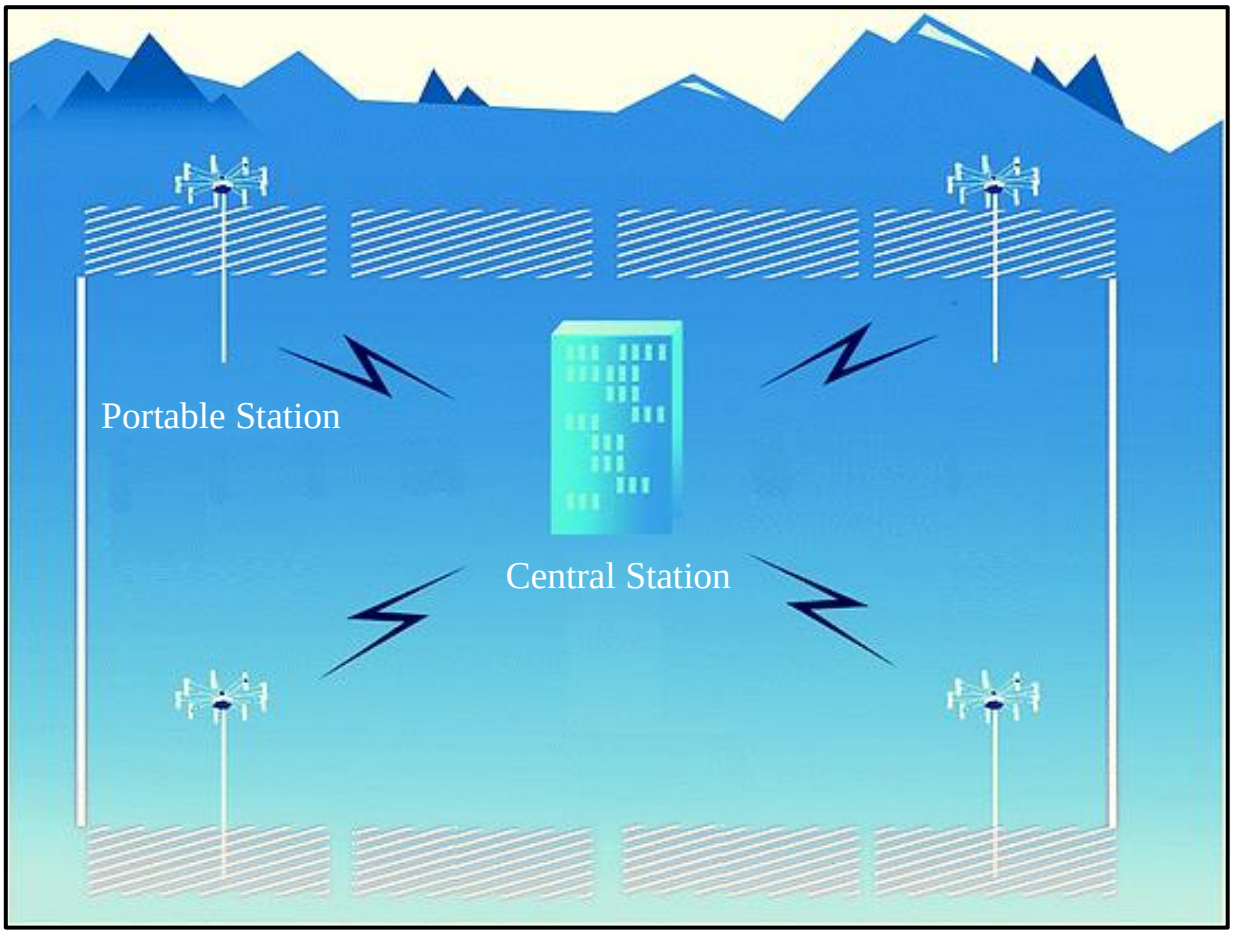


Figure (2.13) Geographically Distributed RFST System [60].

2.6.2.3 RFST System Combinations

The three systems discussed above can be combined into one system. The integrated system may have fixed stations [60], deployable stations and mobile stations, as shown in Figure (2.14).

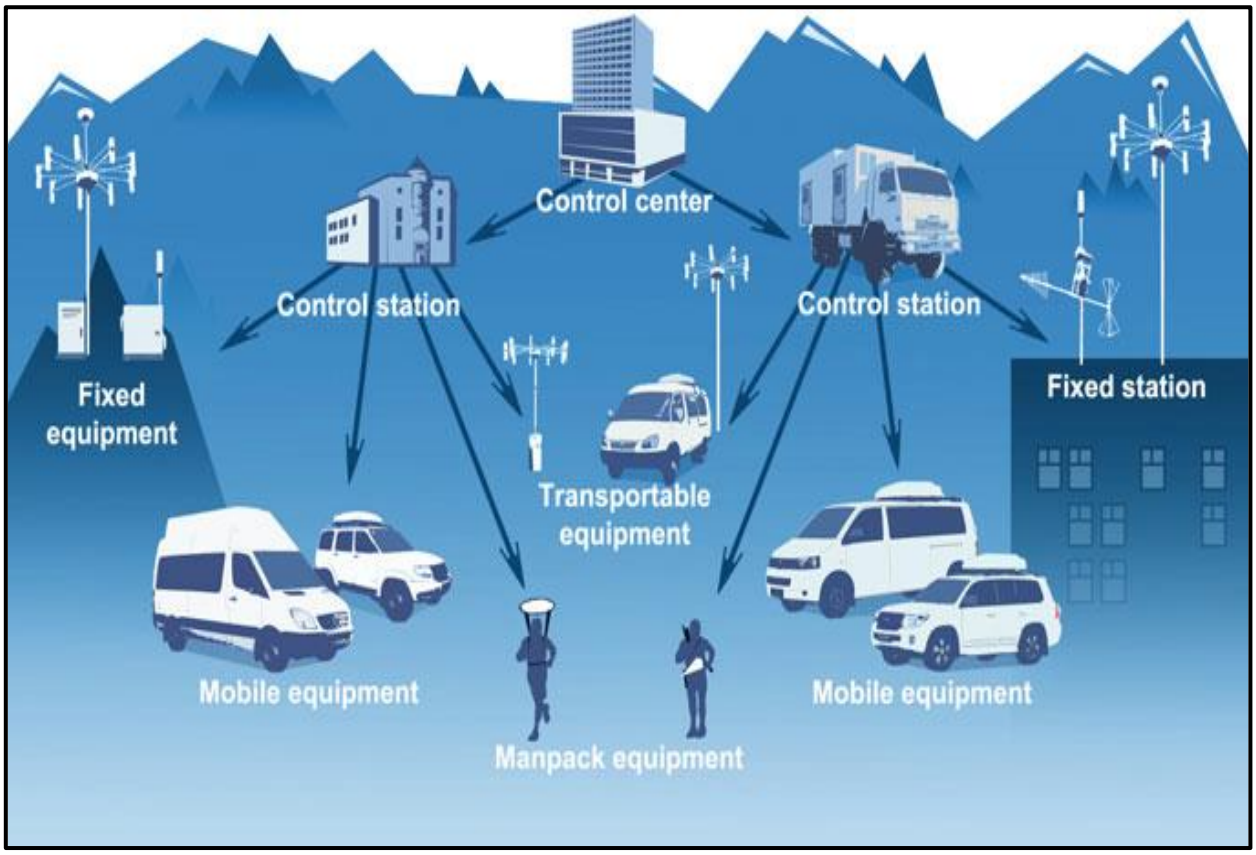


Figure (2.14) RFST System Combinations [60].

Chapter Three

The Proposed System Design

3.1 Introduction

This chapter presents the proposed RFST system design and simulation in two faces: the first part is the simulation of the system using Matlab, and the second part is the practical implementation of the proposed system.

The first part presents the simulation of the essential stages of building an RFST system, starting from simulating the performance of the RF Tracking Station, which has algorithms mentioned in the second chapter, to examining the works of these tracking algorithms under certain conditions, and ending with stimulating their work with different shapes of arrays to get the best performance from tracking algorithms. It also offers other cases for tracking and calculating the location of RES. The tracking scenario starts from a single station to multiple stations to identify the accuracy of the proposed system.

The second part introduces the physical architecture of the RFST system, including hardware and software platforms and communication and interaction with internet. The proposed system construction is presented in its essential components, requirements, and methodologies for construction and implementation. The features and operation of the system are also discussed.

3.2 Proposed System Design

The proposed system design of the RFST system must be defined in detail to understand its function. In this section, the block diagram of the proposed system is presented as shown in Figure (3.1), which contains two main parts: the operations center and radio signal tracking stations.

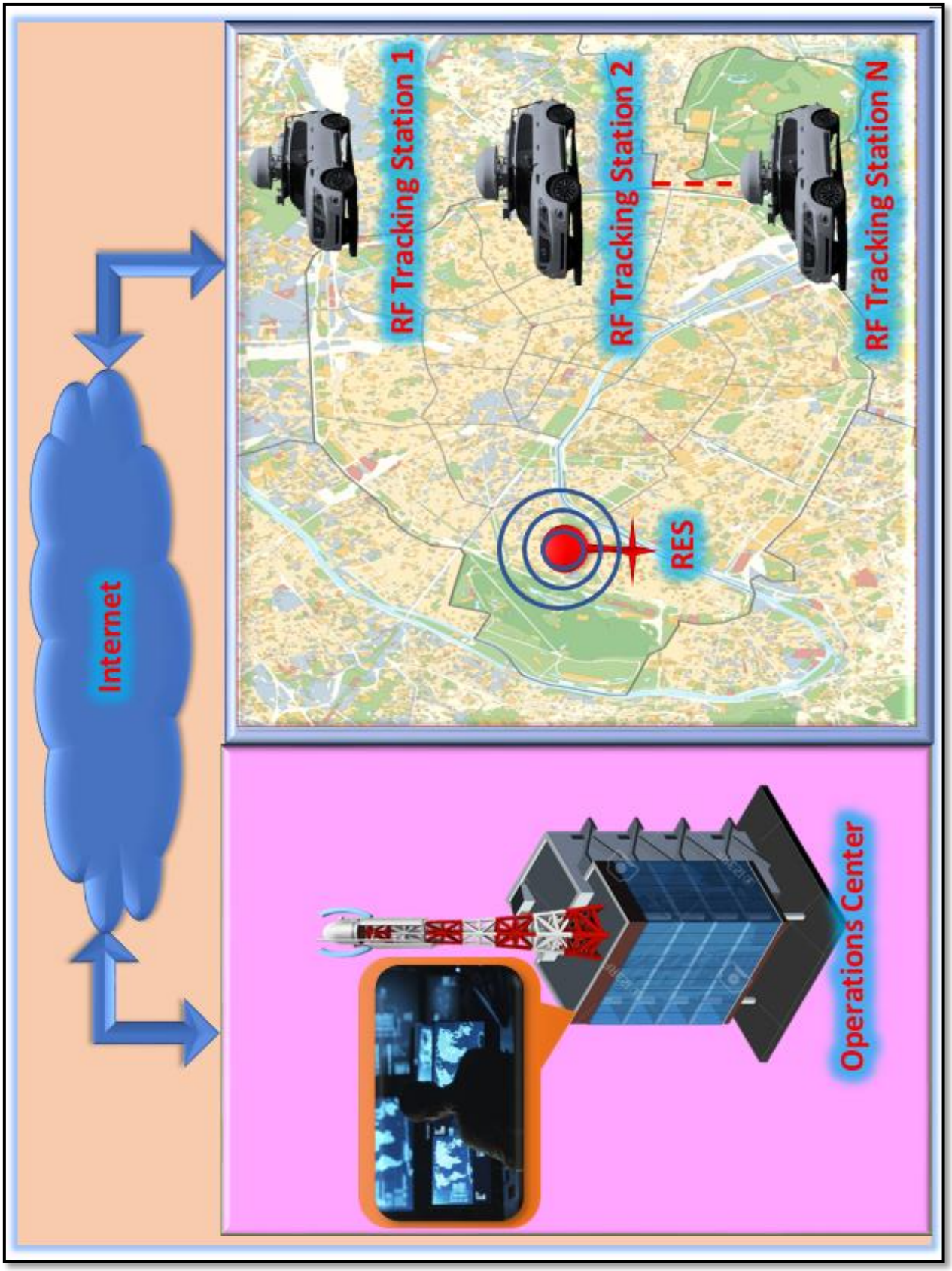


Figure (3.1) Block Diagram of the Proposed System.

3.2.1 Operations Center

Radio Frequency Signal will be tracked centrally in the operations center with geographically distributed stations equipped with trackers for spectrum measurement and direction-finding. They have connected to operations center rooms via internet. As a result, operators and supervisors in the operations center can monitor and track with a continuous look at multiple information flows to monitor the state of the frequency spectrum and check the presence of radio sources in a particular area.

The operations center contains a central computer that collects data from the tracking stations via internet. The data will be processed using the MATLAB program, which represents the locations of the tracking stations on the map with the tracking data. It calculates the locations of the receiving sources through the intersection of the directions of the RES, then applies the triangulation process. These operations are periodically used to track the location of the RES on the map.

The operations center can access all vehicles via IP address tracking devices and adjust operating parameters such as center frequency and bandwidth. On the other hand, the vehicle sends the required information about the tracking tours from the source.

3.2.2 RF Tracking Station

The primary function of these stations is to track the RES by monitoring the spectrum and tracking of RES, as shown in Figure (3.2). These collected data is shared with the operations center via internet. In addition, the proposed system contains geographically distributed stations within the specified area to collect data about a specific source. In this thesis, the proposed system will be built in several stages as follows:

1. Single tracking station to track and estimate RES location.
2. Two tracking stations to track the RES with calculating the tracking accuracy.
3. Three tracking stations to track the RES and obtain more tracking accuracy.
4. Four tracking stations to track the RES and obtain accurate tracking.

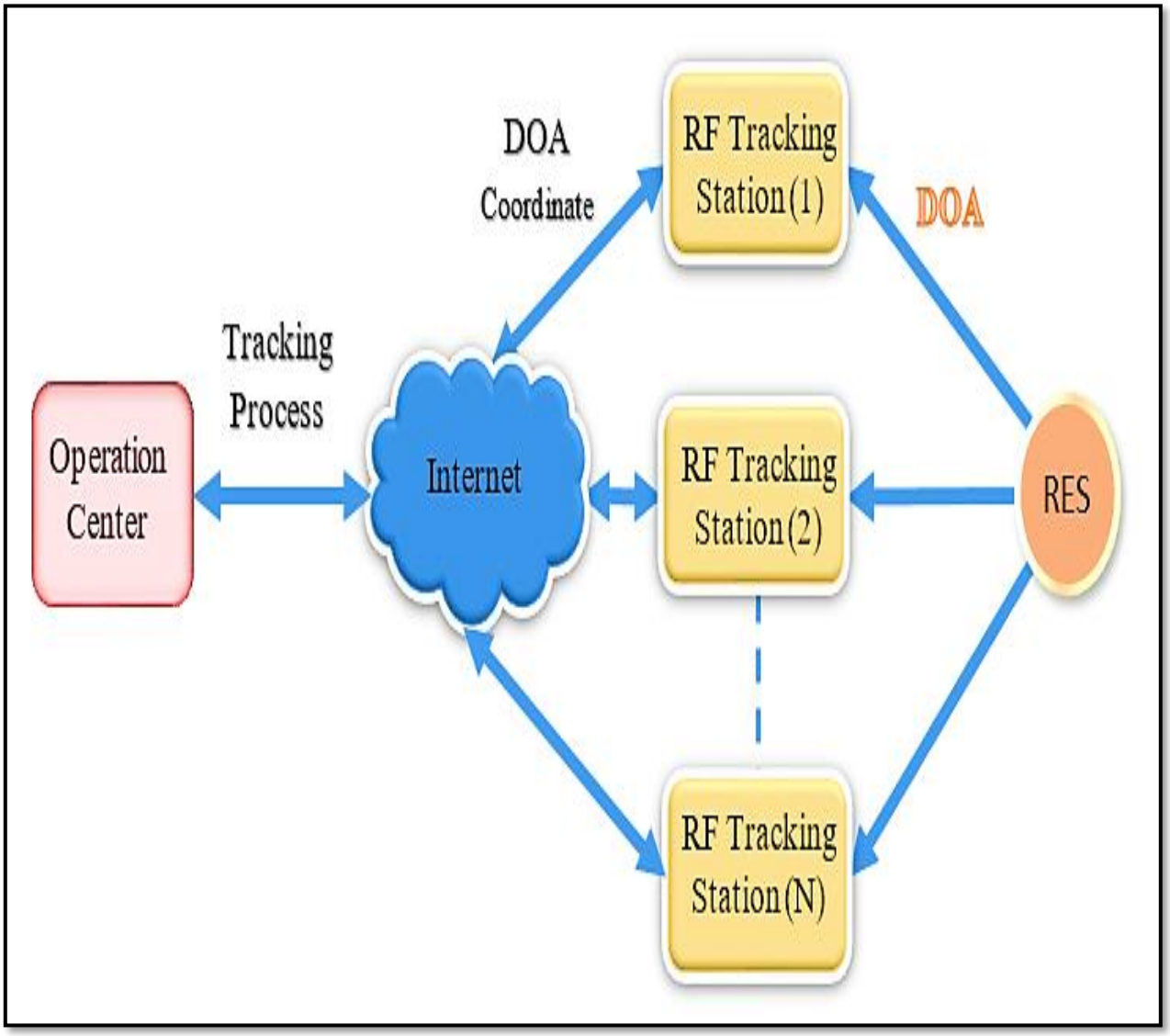


Figure (3.2) Functions of the Proposed System.

3.3 System Operation

The primary function of the proposed system is tracking the RES by scanning spectrum, identifying the direction of RES and sharing data with the operations center by internet and MATLAB for analysis and representations on Google Map. Let us begin with the operation center to illustrate the operation stages. The steps below provide a thorough understanding of how the system works:

- The operations center monitors the RF tracking stations location and can send information to guide stations to a specific place to get tracking data
- Once the operations center selects tracking information (frequency and field), it sends it to mobile RF tracking stations to start the tracking process.
- When the RF tracking stations receive the tracking information, it adjusts parameters to start the operation of mobile RF scanning to estimate the direction of RES.
- When the tracking process starts, the computer in the RF tracking station monitors the RF spectrum and implements tracking algorithms with connected receivers and array antennas.
- The computer connects internet and sends instantaneous tracking data for the specified field.
- All the tracking data is displayed numerically and graphically in the operations center.
- Operations center analyses the collected data for the specific field to track the desired RES and share the collected information with other RF tracking stations.

3.4 System Simulation

The system simulation can also help to provide a primary idea for the system performance by analyzing the approximate working conditions, which is challenging to implement in practice directly. Therefore, simulation, data collection, and analysis of the RFST system through a combination of functions and scripts are used.

RFST system simulation allows us to avoid implementing the reparation of many hardware prototypes to analyze the design for existing or new additions. Before including the hardware prototype, users can check many digital models.

The section presents the performance differences among RFST versions with different tracking algorithms, array configurations, and elements. The simulated data is then saved in data files organized by factors like antenna elements count, antenna spacing, and frequency changes.

The simulation of the RF Tracking Station operation and evaluating the accuracy of the tracking algorithms under certain conditions is significant. For example, the accuracy of the tracking operation depends on the resolution of algorithms. Therefore, to simulate various antenna arrays with extracting the best performance of tracking algorithms with the best antenna arrays type for tracking operations.

The simulation of RFST system configurations may provide a comprehensive view of the system's work by evaluating the accuracy of the RFST system with various tracking algorithms to ensure the system works properly. Therefore, many simulation cases will be considered, such as the tracking system having a single or multiple tracking stations. Furthermore, the tracking process will be taken for static and moving RES, and station status will be considered in case of fixed or moving. The RFST system is also integrated with internet to assess the obtained results for the system.

3.4.1 Single RF Tracking Station Performance

This section presents the RF Tracking Station's performance simulation, which is considered essential to the RFST system. Therefore, this simulation aims to show the results of the best resolution of tracking station algorithms that can be achieved because this will affect RFST system accuracy.

In this simulation experiment, a uniform linear array has been considered with $\lambda/2$ the spacing of sub-antenna, where λ is the wavelength of incoming signals. The investigation is presented for various tests with changing different affected parameters, which are

- The number of incoming signals (M),
- The number of antennas (N).
- The number of snapshots (S).

This test considers the seven kinds of DOA tracking algorithms used in the tracking station presented in the following.

1. Beamscan.
2. MVDR.
3. MUSIC.
4. ESPRIT.
5. Beamspace-ESPRIT.
6. ROOT-MUSIC.
7. ROOT-WSF.

The results are considered to evaluate the tracking algorithms performance in the moving source and multipath Environment, regarded as an affected test for the RFST system.

In RF Tracking Station simulation, the results are measured the input-output relationship of the algorithms types concerning parameters variation associated with the accuracy of tracking. As shown in Figure (3.3), GUI was designed using MATLAB App Designer, through which simulations of the tracking algorithms are managed. This GUI contains several windows for entering data and displaying results.

In the GPS of the RF Tracking Station window, the coordinates of the station's location are entered. Also, in the RF tracking station parameters window, signal information and the DOA parameters are set. The frequency spectrum of incoming signals is displayed in the Spectrum window at the bottom of the GUI. In addition, the results are displayed for all algorithms in the estimation window by selecting one of the estimation techniques.

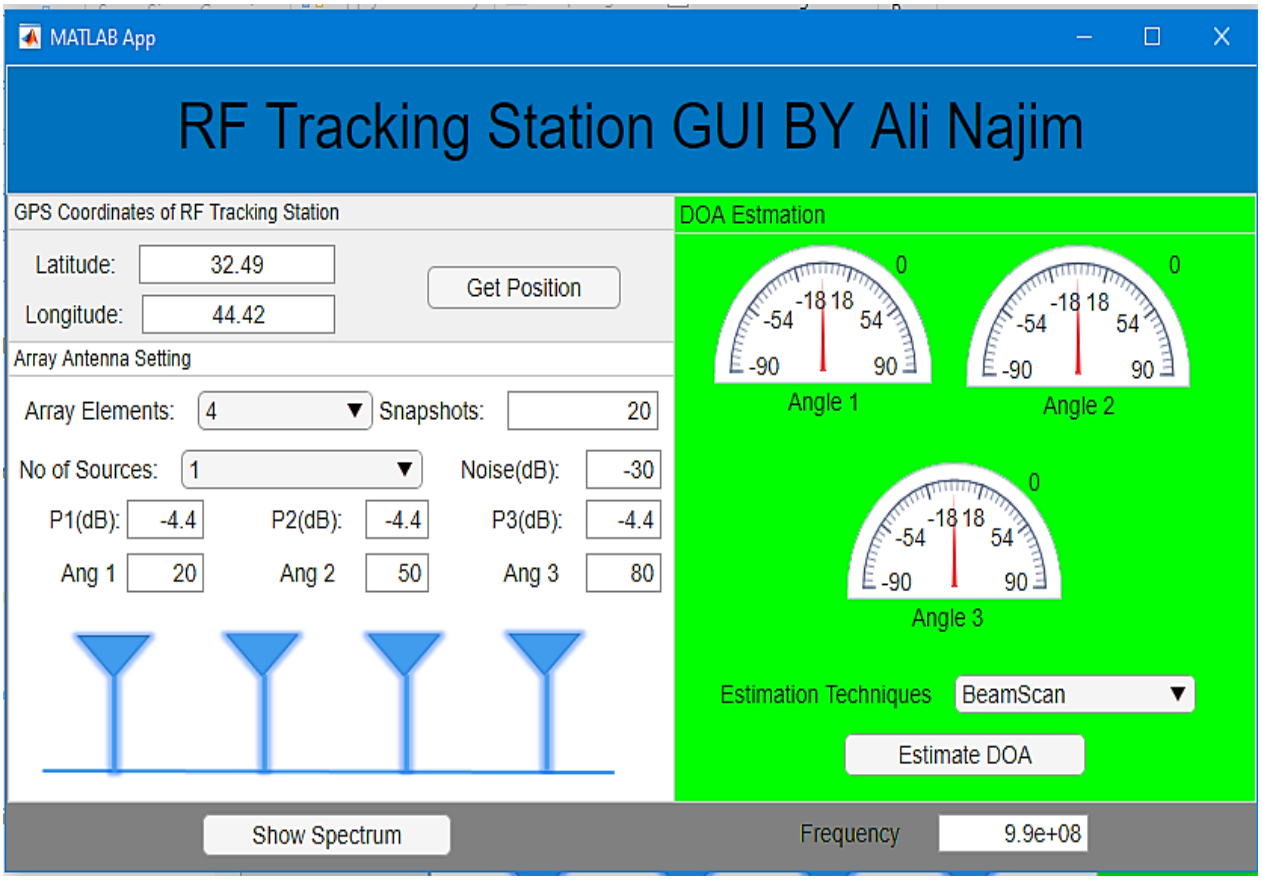


Figure (3.3) GUI of Tracking Station Simulation.

In Figure (3.4), the RF Tracking Station simulation flowchart was illustrated to clarify the programming steps of tracking operation.

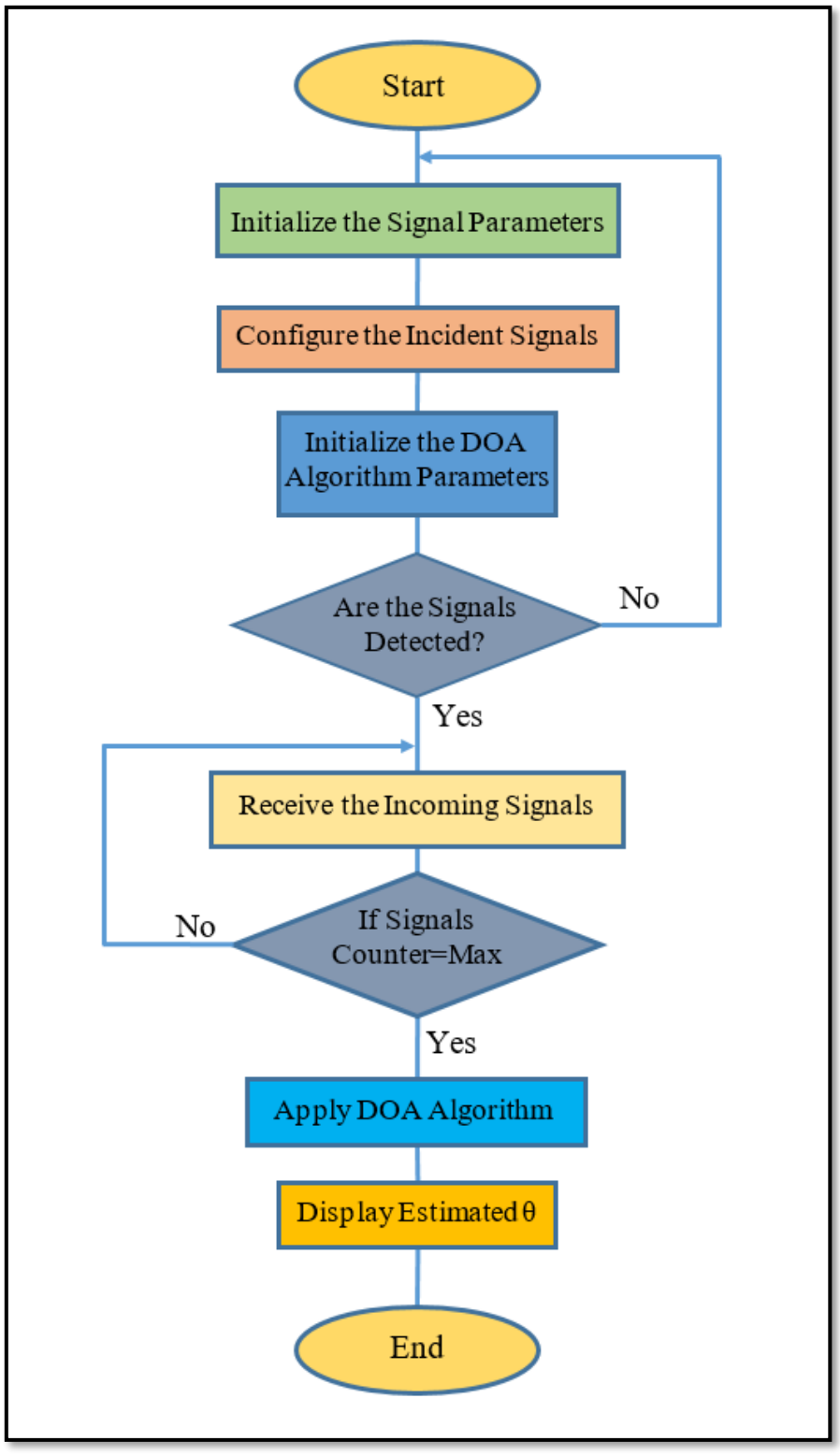


Figure (3.4) Flowchart of RF Tracking Station Operation.

3.4.2 Tracking Algorithms with Conditions

The section presents the investigation of the tracking algorithm with certain conditions to determine the best resolution between ROOT-MUSIC and ROOT-WSF. It is tested under complex criteria: the number of incoming signals, Correlated and uncorrelated signals, Converging sources, and the different frequency signals.

This case shows the effect of increasing the number of incoming signals on RF Tracking stations. Therefore, the Tracking algorithm provides good estimation results when the small number of signals. However, tracking techniques will have less accurate estimation when more signals are incoming.

In correlated and uncorrelated signals, Correlated signals are presented as signals that share the same properties as another signal; they can also be given as signals that exhibit a statistical correlation between samples or have a non-flat spectrum. Conversely, uncorrelated signals lack intersample correlation or have a flat range.

In space converging sources, this section includes converging sources in space, which is a delicate situation observed repeatedly in communication systems. Therefore, a super-resolution of the tracking algorithm is needed. The simulation section details the evaluation conducted to accurately identify the superior tracking technique to estimate the tracking data of space converging sources accurately.

In the different frequency signals, this impact changes when high or low-frequency signals come across in the other frequency signals. It is provided that low-frequency signals make good coherence and are good-correlated. However, high-frequency signals achieve a low degree of coherence and are uncorrelated. Therefore, the variance between low and high frequencies impacts the estimation of the DOA. In Figure (3.5), the flowchart of the RF Tracking Station simulation was illustrated to clarify the programming steps.

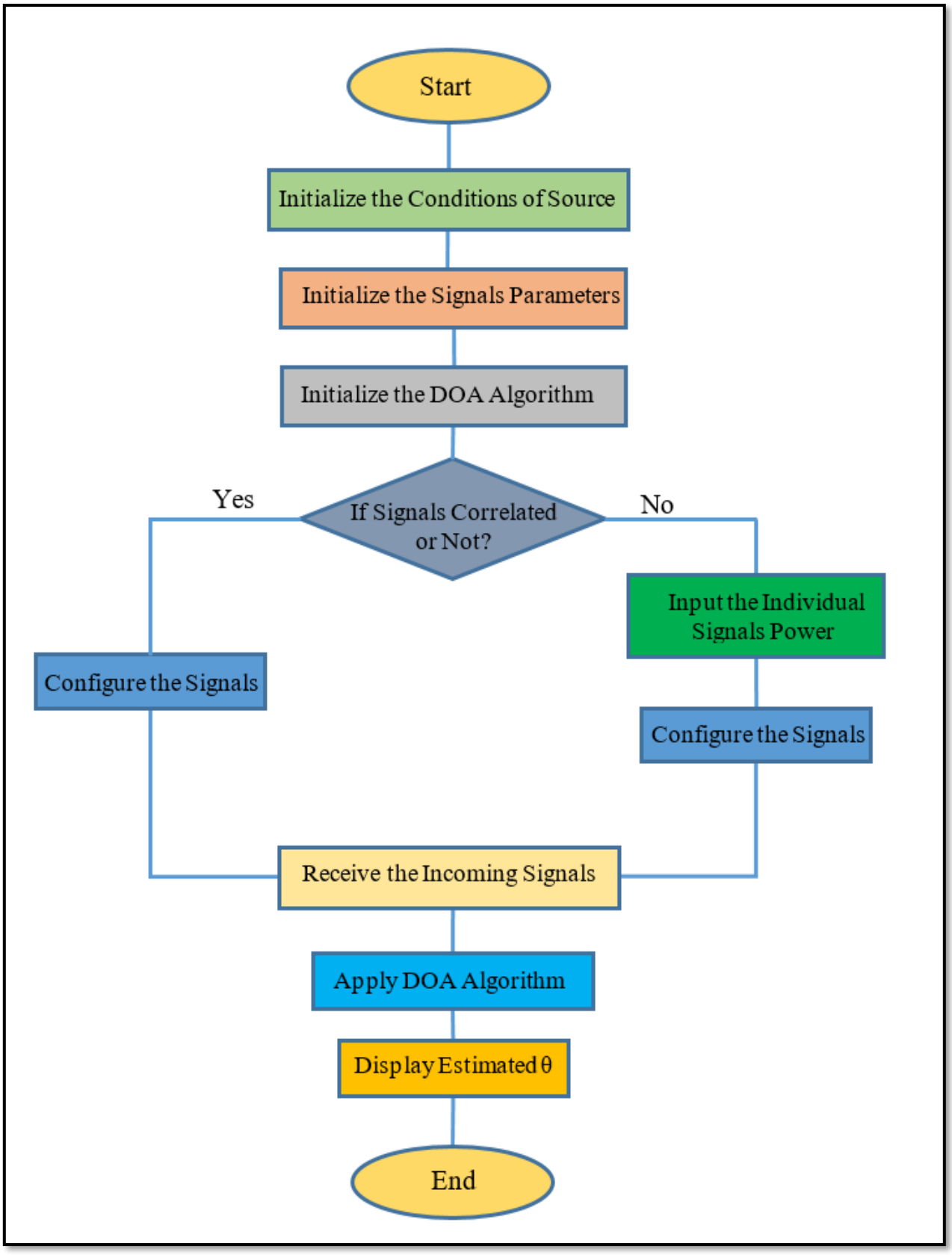


Figure (3.5) Flowchart of RF Tracking Station Operation with Conditions.

3.4.3 RF Tracking Station with Various Arrays Configuration

This simulation study the impact of various array geometries with different tracking algorithms. The performance of each geometry is compared to several tracking algorithms to achieve an accuracy criterion for the RFST system. Therefore, the best geometry can be selected for each tracking algorithm. Different types of array geometry, ULA, URA, UCA geometries, are experimented with spatial spectrum DOA algorithms such as Beamscan, MVDR, and MUSIC.

The array geometry is constructed as ULA, URA, and UCA, consisting of a number of antenna elements defined in simulation. The array element spacing is also illustrated in the simulation with each array geometry. ULA only needs one variable for element spacing. URA and UCA need two variables of element spacing for configuration.

The incident wavefield is contracted by initializing its parameters such as a number of sources, physical angles, the carrier frequency of the incoming sources. The critical point in this simulation is the implementation of a tracking algorithm. The Scan angles of the algorithm must be initials depending on array configuration. The output of the simulation is the plot of the spatial spectrum. The locations of the largest peaks of the spectrum identify the bearing of the signals. Therefore, this process will be continuously repeated to track the RES.

Array geometries are of eclectic interest by their consistent performance over the different angles and simple concept geometry. It can quickly be developed to get a new geometry by changing spacing elements, inserting new features into the structure, or varying the geometry to better array performance in response to the RFST system. In Figure (3.6), the flowchart of the RF Tracking Station simulation with array configurations was illustrated to identify the simulation steps.

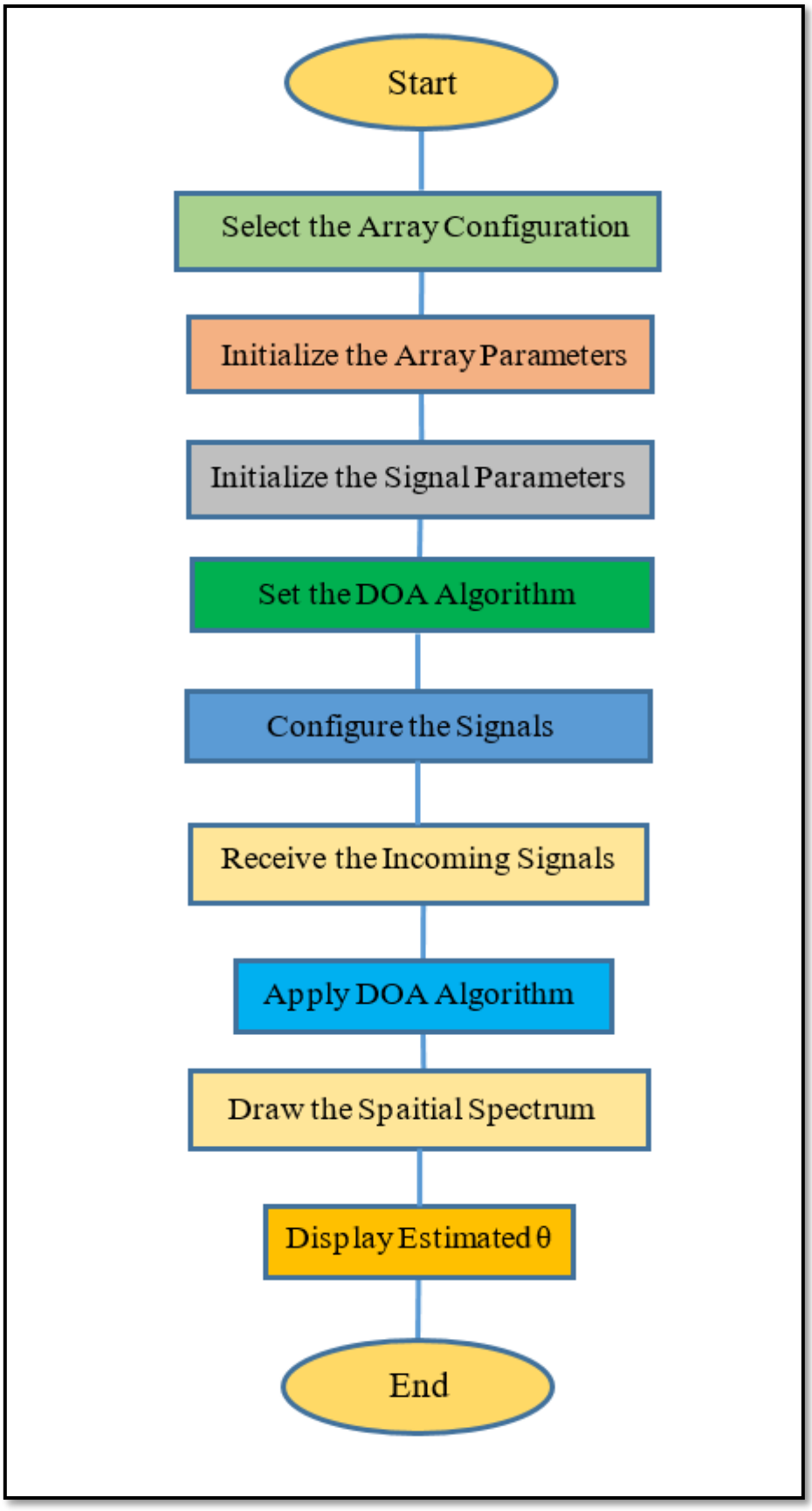


Figure (3.6) RF Tracking Station Performance with Array Configurations.

3.4.4 RFST System

This section presents the design of an RFST system that utilized connected RF tracking stations with internet. Figure (3.7) shows that the RF tracking station measured the direction data. It is then collected by internet to the operations center, which is applied the triangulation process to estimate the RES position and perform the tracking operation. Therefore, the RES position is automatically calculated. The system model is based on tracking algorithms and internet. The operations center uses the obtained results of the tracking stations for applying the triangulation process, which depends on the intersection point of triangle equations as the RES position.

The simulation proves the performance of the RFST system was carried out in stages, which include building the system in the following:

- One RF Tracking Station to track the direction of RES.
- Two RF Tracking Stations to track the RES.
- Three RF Tracking Stations to track the RES.
- Four RF Tracking Stations to track the RES.

Many cases are considered in the simulation that is presented in the following:

- RF Tracking Station and RES have static positions.
- RF Tracking Stations has fixed positions and RES in continuous moving.
- RF Tracking Station is constantly moving, and RES have fixed positions.

Triangulation has been achieved when two directions of RES are detected from two RF Tracking Stations with different positions. Therefore, a more accurate estimation of the RES location is obtained when tracking stations increase. The locations of stations are selected with uniform distribution in the scanning area. Stations are organized with specific configurations to provide better performance. The coordinates of the RES can be determined by the tracking data and GPS

coordinates on each station. Each station has coordinates and placed them in separate locations to achieve a tracking process.

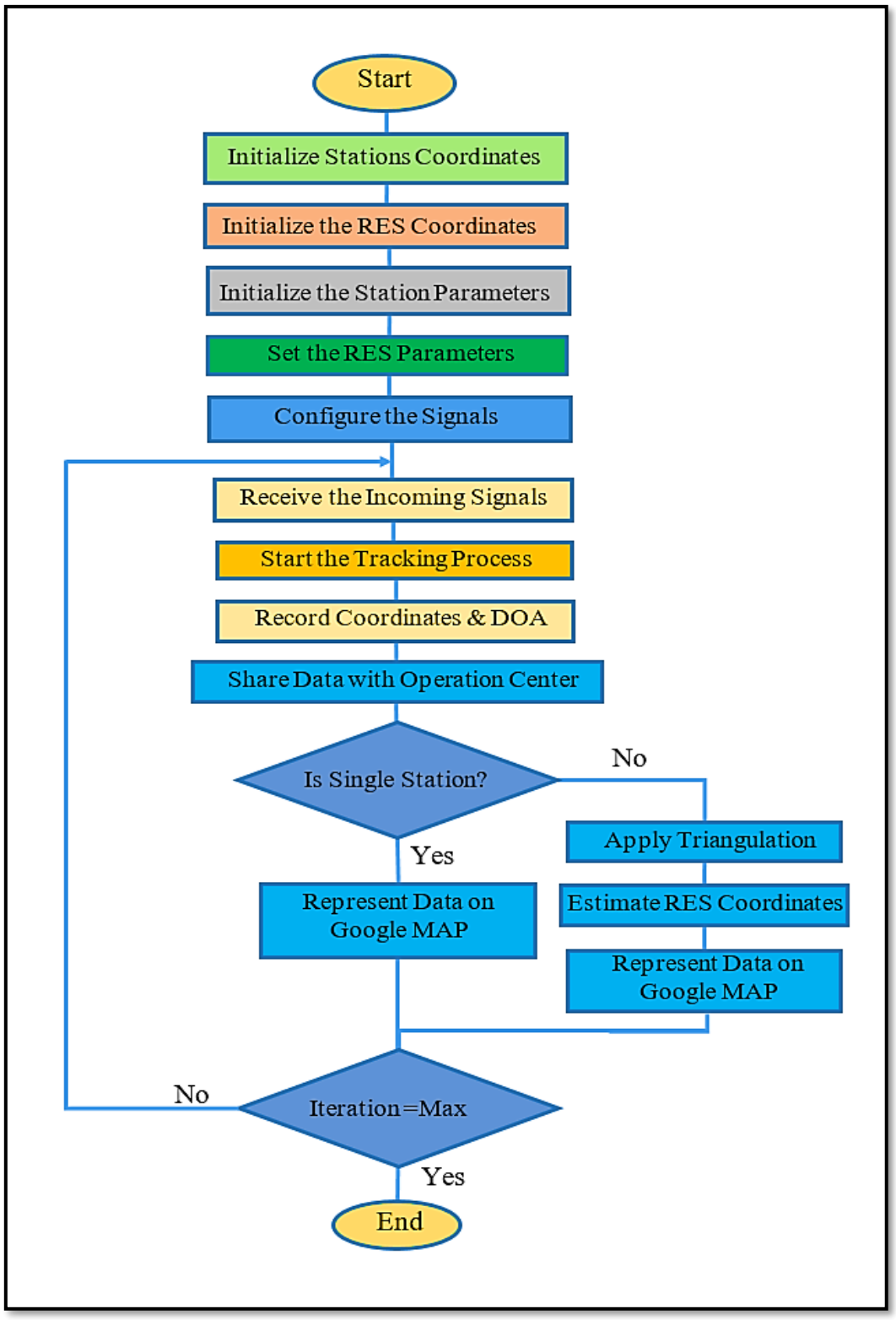


Figure (3.7) RFST System Simulation Steps.

3.5 System Implementation

The final part of this work concerned the proof-of-concept implementation of the RFST proposed system. Initial work on this system yielded working versions, as presented in the simulation section. The design of this system proves that there is the ability to build an RFST system based on open source systems interfaced with the internet.

This section presents the implementation of the RFST system as shown in Figure (3.8), including both the hardware and software platforms and the connection and interfacing with the internet. The system is also presented with its essential components, requirements, and construction and implementation methodologies. The system's features and operation are also discussed.

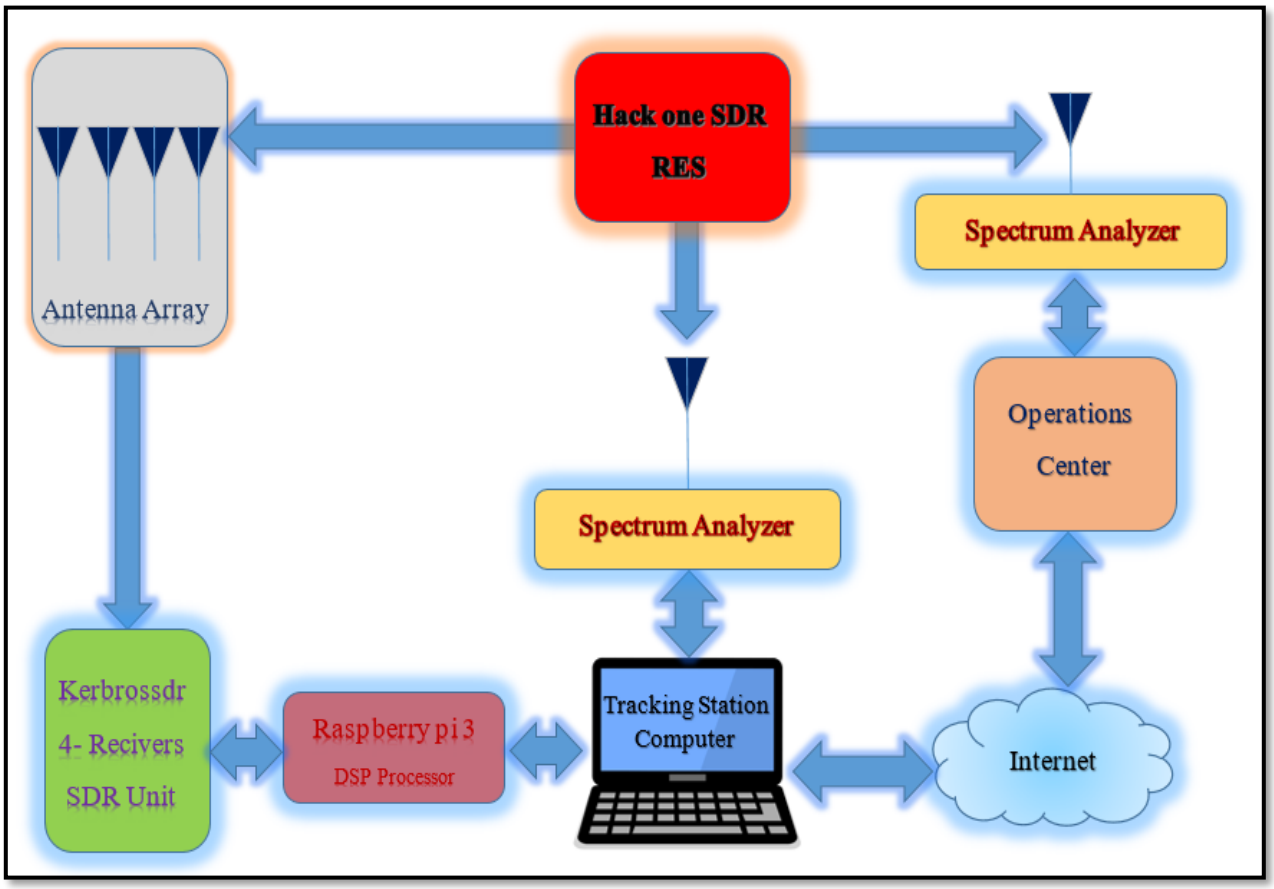


Figure (3.8) Block Diagram of RFST System Implementation.

3.5.1 Antennas Array

This system antenna array was constructed from four whip antenna, an ultra-wideband antenna that provides consistent high omnidirectional gain and efficiency from 25MHz to 1.75GHz. Figure (3.9) shows that this antenna was made to be mounted on metal surfaces for best performance because it is a magnetic mount. When mounted on a metal plate. Antennas array can be configured in two formations that are ULA and UCA.

Figure (3.9) Whip Antenna.

3.5.1.1 ULA

ULA is configured of four sensor array elements that are evenly spaced in a straight line. The antenna array elements are configured from the element origin point set as the reference element of the array, and the inter-element spacing between any two elements is d. ULA model. ULA can realize the one-dimensional DOA

estimation. As well, it estimates the heading angle of the RF signal source. For example, ULA can measure the angle in a field of 180 degrees that only has a 180 degrees resolution, as shown in Figure (3.10). This is simply a straight line of omnidirectional antennas (e.g. magnetic whips or dipoles) spaced out at d.

$$d=\lambda * s \tag{3.1}$$

Where λ is the frequency wavelength and s is the inter-element spacing factor.

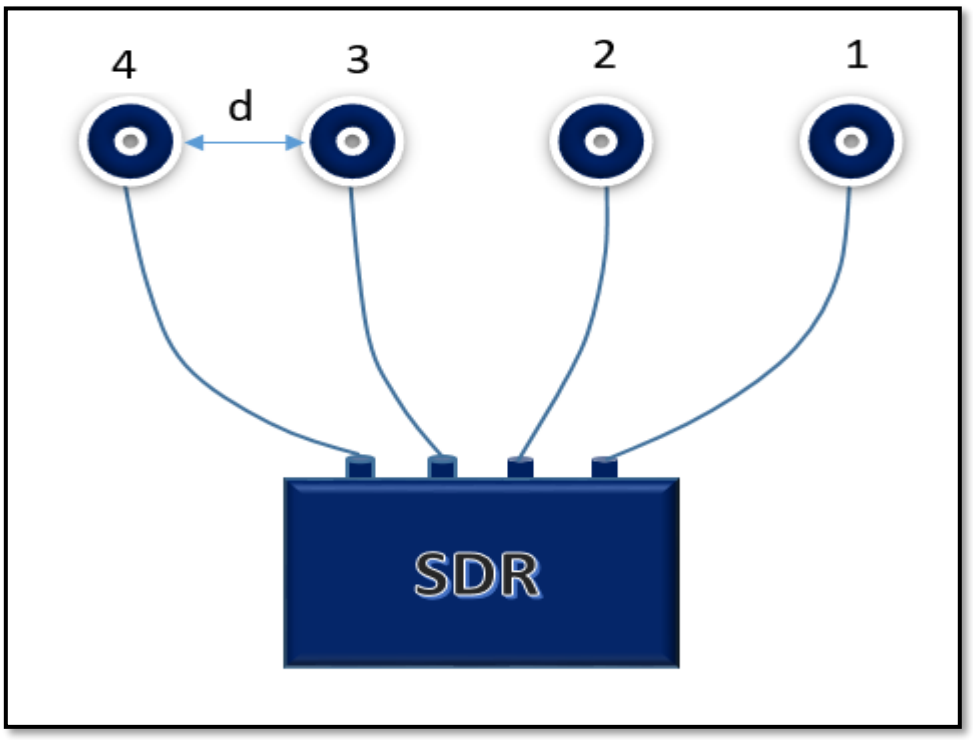


Figure (3.10) ULA Connection with SDR.

3.5.1.2 UCA

UCA is the configuration in which the antennas are arranged in a circle and analyze the three-dimensional coordinate system in the plane formed by the X-Y axis. The origin point is at the center of the circle, the radius is R, and the sensor array elements are evenly and uniformly dispersed around the circumference to configure a circular antenna. In UCA, as shown in Figure (3.11), the linear distance between any two array elements is d, as mentioned in ULA.

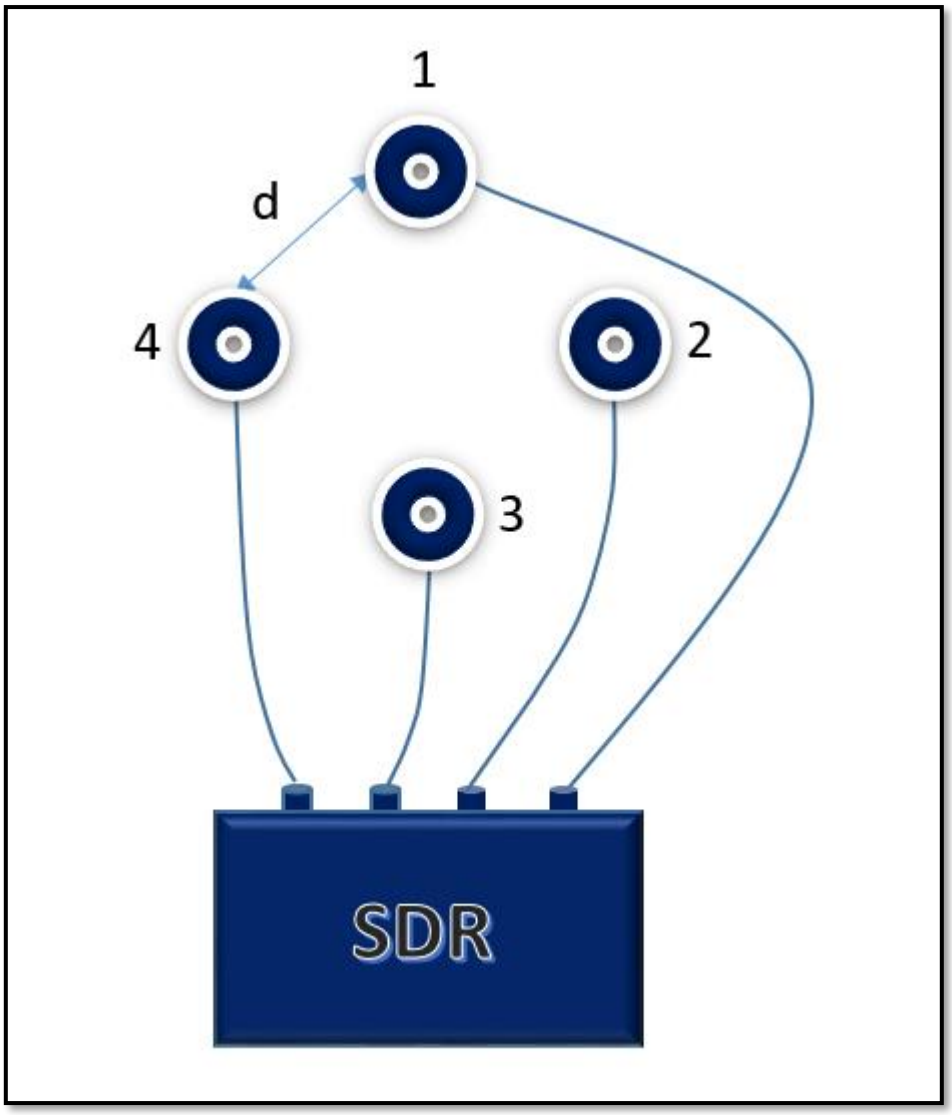


Figure (3.11) UCA Connection with SDR.

3.5.2 KerbersSDR Receiver

The KerbersSDR is open-source radio that can be developed to operate as RDF. It is a low-cost four-tuner phase and coherent SDR. This SDR can be used to create to prove a graphically measuring display that monitors the collected bearing from the transmitter location. KerbersSDR board datasheet is presented in appendix (B). It consists of four RTL-SDR (R820T2) as shown in Figure (3.12), a noise source with a wide frequency range controlled in software, a USB Hub, and a calibration board

to synchronize samples with the noise source. Computer devices with an operating system based on A Linux such as a PC or Raspberry Pi as shown in Figure (3.13).

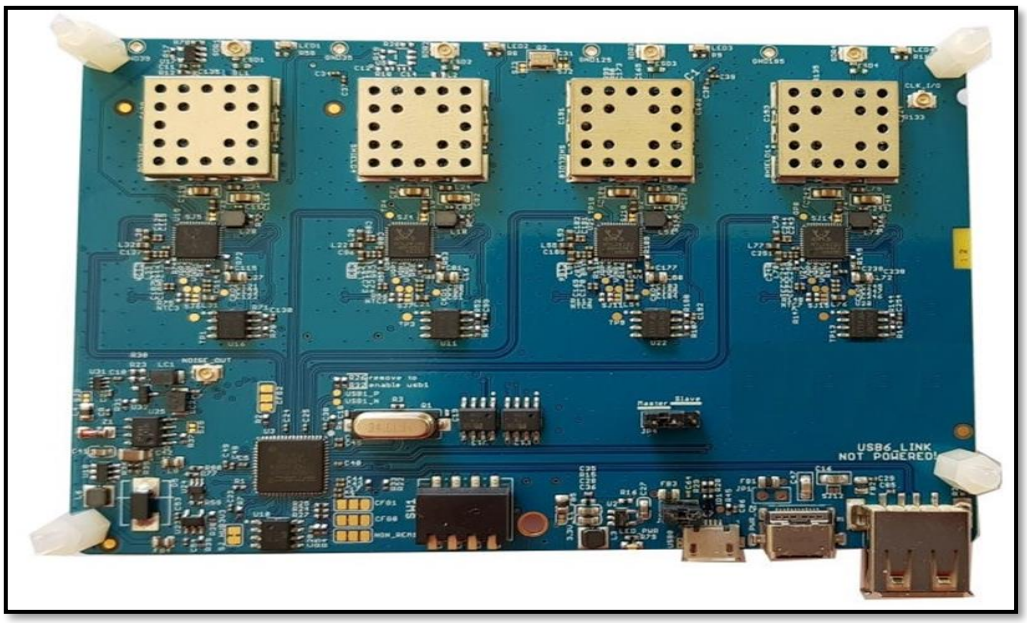


Figure (3.12) Top View of 4-SDR (R820T2).

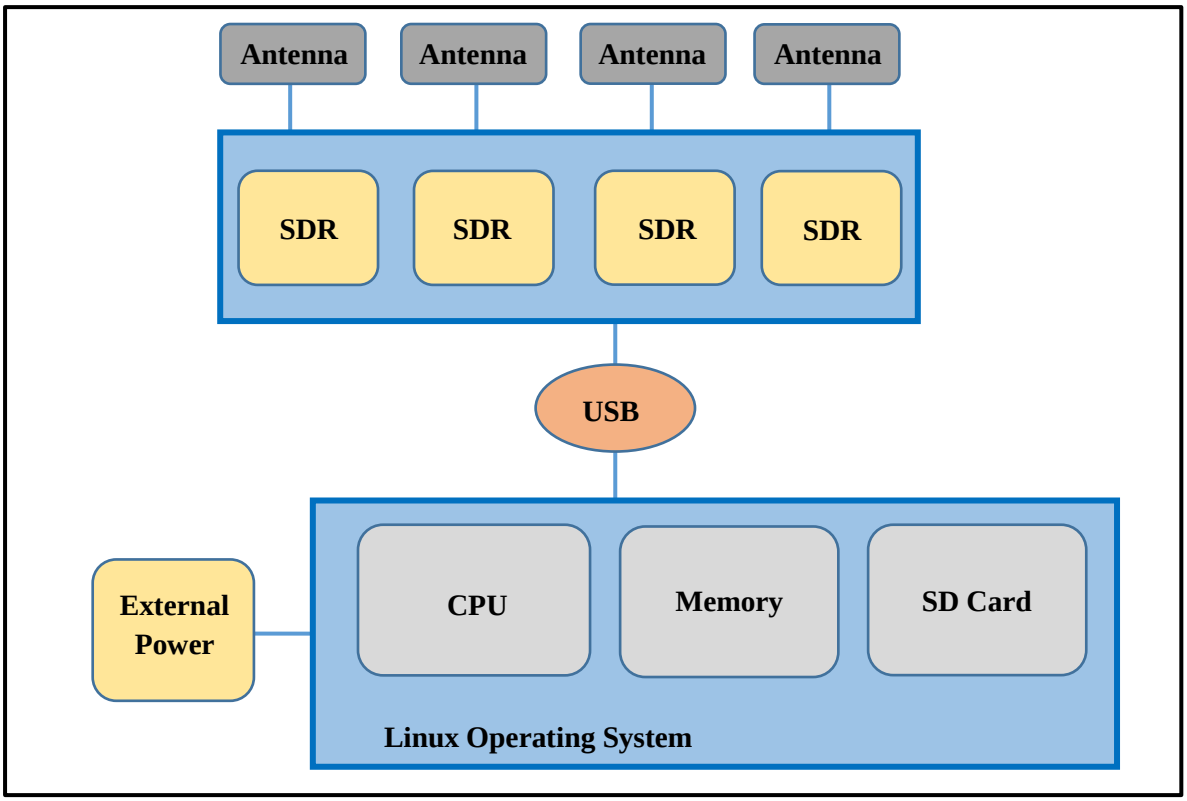


Figure (3.13) Kerbrossdr (R820T2) Connected to Raspberry Pi.

Each SDR receiver is constructed from the RTL2832U and R820T2. The micro-USB port is linked to the USB port of the computer, and the right micro-USB of the RTL-SDR dongle is only for extra power. General specifications of the Kerbrossdr are listed below in Table 3.1.

Table 3.1. General specifications of the KerbrosSDR.

Characteristic	Specification
Operating frequency range	24 MHz – 1.75 GHz
Bit Depth	8 bits
ADC sample rate	2.4 MSPS
Power supply	5V USB powered (micro USB)
Compatibility	Linux

In RFST applications, this SDR is used in RF Tracking Station to collect the bearings of the RES is related to the antenna array. Therefore, the direction of RES is calculated by taking the difference between the user direction (found approximately by GPS and Google Earth) and the measured bearing provided by the KerbrosSDR dongle. The RES position is calculated based on the triangulation process. The obtained readings of RF Tracking Station at various locations attempt to determine the intersection of line crossing. Where the intersection occurs is represented as the estimated location of the RES.

3.5.3 Raspberry Pi3

The Raspberry Pi3 is one of the essential parts in installing an RF Tracking Station. It connects with SDR to manipulate the data received for the incoming signals on the array. By using Raspberry Pi, SDR operation parameters are entered,

instantaneous operations are managed, and the results of the tracking algorithms are shared through IP.

Raspberry Pi3 is chosen for this system because it uses less energy, is small, and has strong processors. The Raspberry Pi3 datasheet is presented in appendix (D), has a memory chip and a graphics processor. In addition, it consists of connectors and interfaces that allow it to be linked to external devices. A Linux operating system is installed in Raspberry Pi. It contains a python programming language used to build tracking algorithms. The Raspberry Pi3, as shown in Figure (3.14), is programmed to process the incoming signals by SDR and then implements the algorithms to share the results over Wi-Fi.

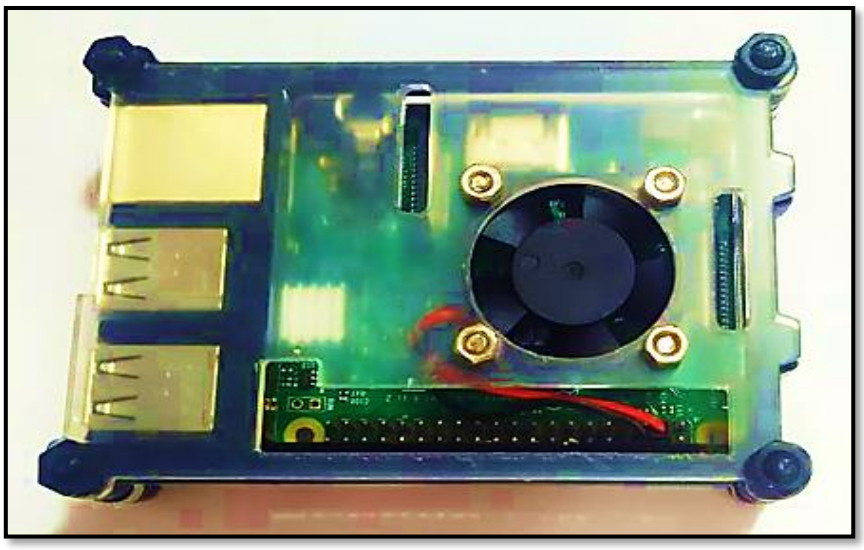


Figure (3.14) Raspberry Pi 3 Connection with SDR.

3.5.4 Operations Centers GUI

Operations center GUI is considered an essential software part of the RFST system. This model of applications was programmed to interface with internet and Google maps, as shown in Figure (3.15). In addition, this GUI connects with RF Tracking Stations for collecting the bearings and GPS data for specified scanning areas.

Figure (3.15) Operation Center GUI.

This GUI enables the operations center supervisor to collect the bearings information for a specific RES from different locations by multiple RF Tracking Stations. In a single tracking station, it is possible for a measured bearing of signal to be skewed or incorrect as it can be reflected by multiple surfaces and appear to be arriving from the wrong direction. Therefore, the multipath issue can be overcome by collecting various data of many RF Tracking stations distributed in different positions. Furthermore, if the tracking data are collected from many locations, the RES location can be calculated by specifying the intersection between bearing lines collected from different points.

This GUI is implemented to gain RFST capabilities with interfacing RTL-SDR dongle and internet. It can record and plot bearing data based on Google maps. In addition, it can determine the estimated location of RES. Therefore, the RES can be evaluated by providing sufficient data collected. With operations center GUI, the supervisor enables to analyze the spectrum, monitors RF Tracking Station measurement and GPS coordinates, and represents collected data on Google map.

3.5.5 Spectrum Analyzer

Advanced RF Spectrum Analyzer is used in RFST to monitor the spectrum in RF Tracking Station and operations center. It is based on RTL-SDR hardware with MATLAB software to implement RF spectrum analyzer systems and digital receivers. This technology is selected to gain the function of the RFST system, such as monitoring signals in the range of 25MHz to 1.75GHz, digitizing it with hardware, and then extracting the signal's information in the software.

This RTL-SDR is linked to the computer as shown in Figure (3.16), the RTL-SDR antenna becomes an RF input port. Spectrum Analyzer GUI is used to sketch the SDR operation, such as receiving samples, demodulating, decoding, and obtaining the output signal. This SDR datasheet is presented in appendix (D). The

RTL-SDR receives RF signals, monitored using designed GUI for spectrum analyzers. The Spectrum Analyzer FFT, which has a bandwidth of MHz, displays the spectrum of a received complex signal in GUI. For example, if SDR is set to be 990 MHz with a 3MHz sampling rate, the device will capture, downconvert, and complex demodulate signals in the 988.5MHz -991.5MHz range.

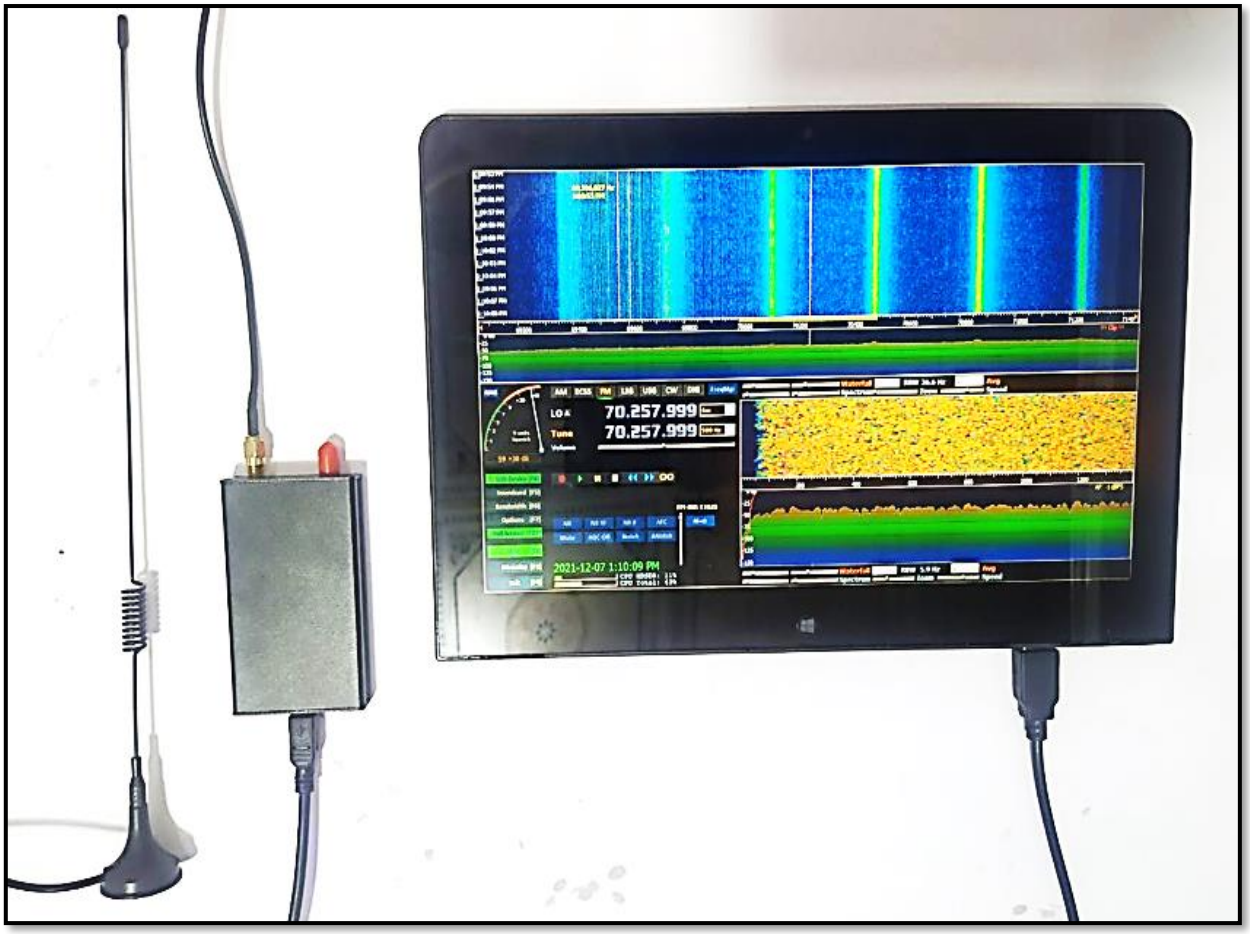


Figure (3.16) RTL-SDR Connected with Computer.

In spectrum analyzer GUI, the green line depicts this range as shown. As seen by the vertical orange line, the power scale of the FFT is represented in dBm. The scope displays the instantaneous power levels of spectrum components in received bands, making it simple to spot signals above the noise floor. It can also be used to manually tune the RTL-SDR to any RF carrier frequency required to receive.

Spectrum Waterfall displays the same frequency range of FFT Spectrum Analyzer, but instead of showing instantaneous power levels, it displays activity as an intensity plot over time (the last 50ms).

The colour scale displays activity power against the frequency components of RF signals, as shown in Figure (3.17). Spectrogram represents the signals in pattern from dark blue to dark red (the noise floor) (high power signal). It allows tracking activity bursts across time, as shown in the figure of the purple line.

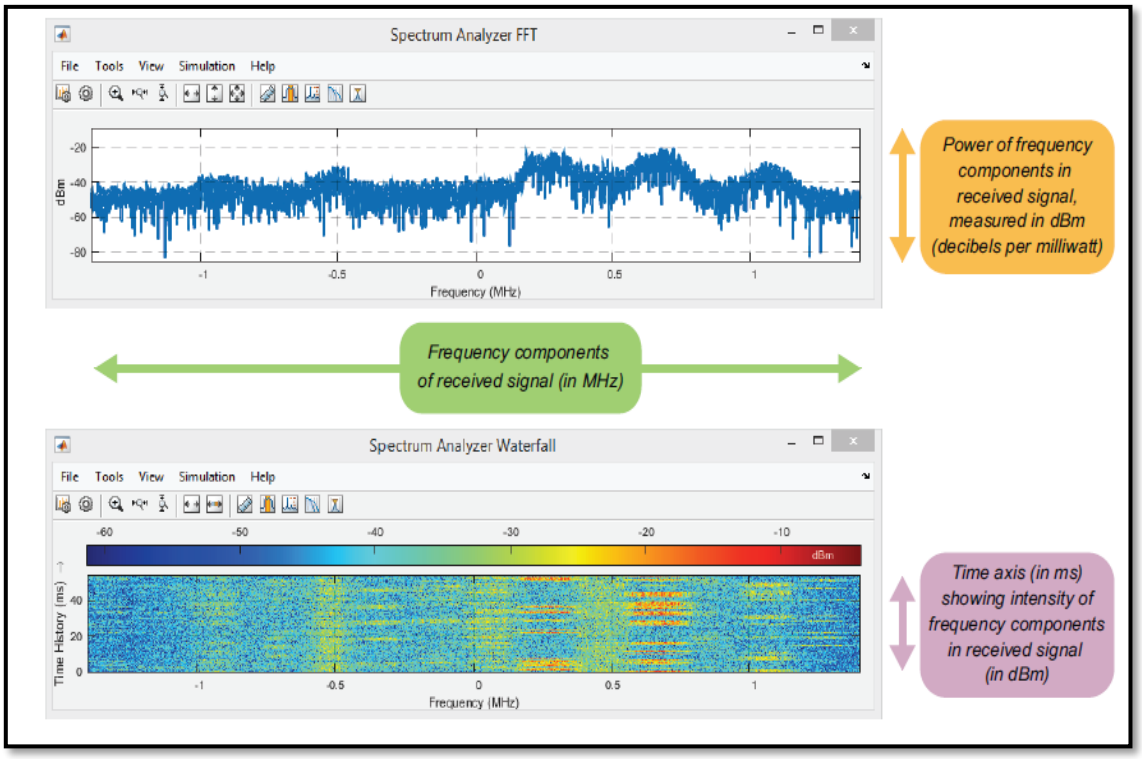


Figure (3.17) The Spectrum Representation of RTL-SDR.

It's worth noting that data shown in the scope appears (and travels) from bottom to top, which corresponds to the window's time on the left. Spectrum analyzer GUI tunes the RTL-SDR to navigate the RF spectrum, GUI as shown in Figure (3.18), provides easy control of the tuner parameters and frequency and control the designed model.

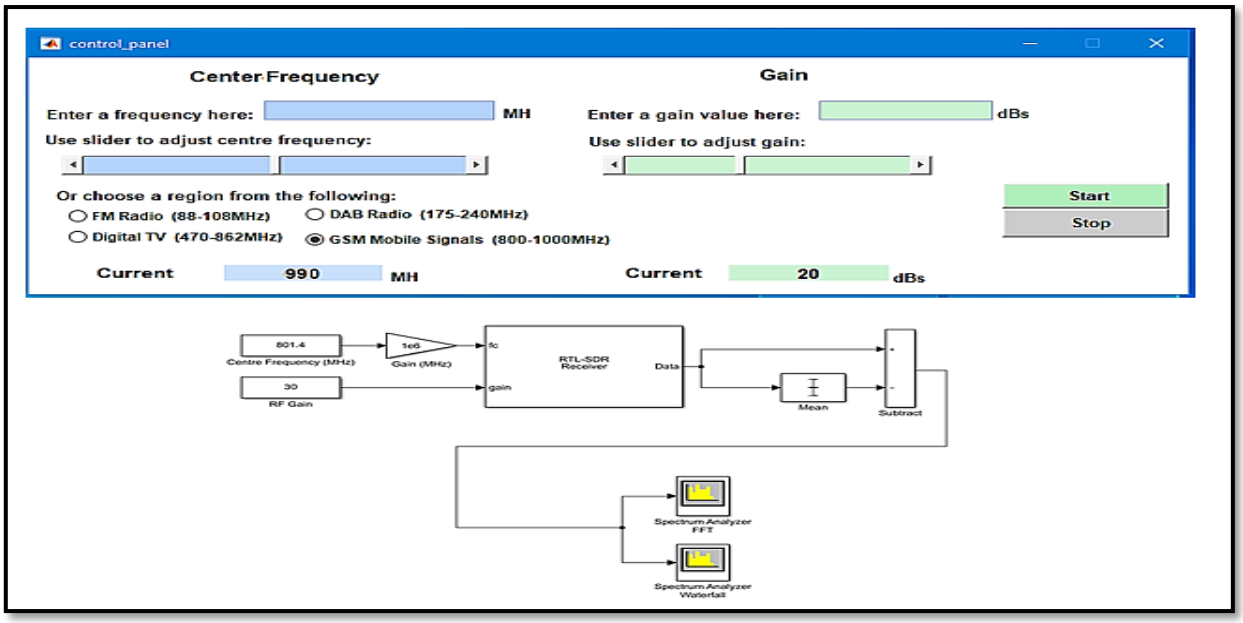


Figure (3.18) GUI Control and Simulink of RTL-SDR.

3.5.6 HackRF-One SDR

HackRF-One SDR can work as a transmitter or even receiver of any radio signal and needs a single peripheral connected to a computer. Figure (3.19) shows that this SDR is connected through USB and can transmit or receive signals with operating frequency from 1 MHz to 6 GHz and output power of 30 mW to 1 mW depending on the band.

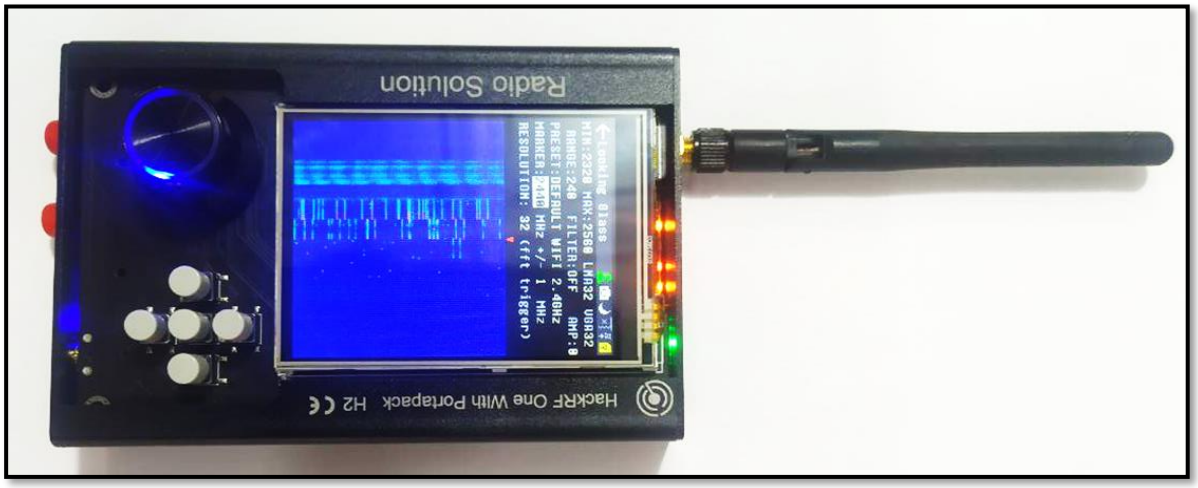


Figure (3.19) HacKRF-One SDR.

3.5.7 RF Tracking Station Configuration

This section expresses the implementation of the RF Tracking Station, starting from installing the Linux Operating system. This OS is designed for the Raspberry Pi 3, which can be flashed to an SD card. First, Linux can be installed using the Etcher program installer, as shown in figure (3.20), which handles the flashing. Then, SD Card is burned and inserted into Raspberry Pi3.

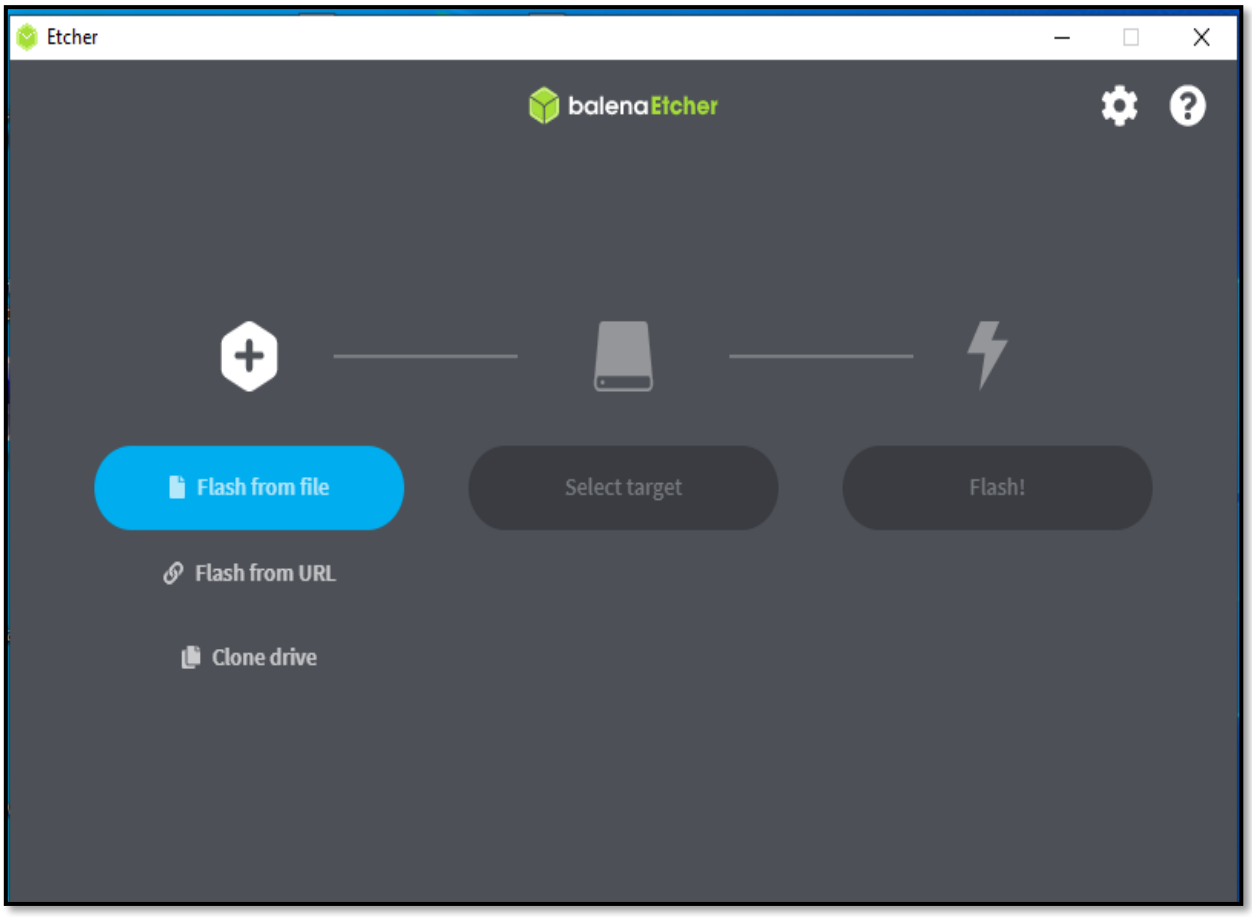


Figure (3.20) Etcher Program.

KerbersSDR dongle is plugged into Raspberry. The power cable is also plugged, Wi-Fi connects to Raspberry Pi3, which operates as a server and connects with devices through the IP address. KerbersSDR is programming by python code, as shown in appendix (F), so the web interface is already configuring parameters.

The setting of KerbersSDR software is receiver configuration and spectrum to set the frequency, sample rate and gain settings, calibration with the noise source, antenna array connection, and algorithm estimation to show the signal bearings SDR setting steps are shown in appendix (B).

As shown in Figure (3.21), the antennas are configured for KerbersSDR and spectrum analyzer on the vehicle's top. The antennas configuration depends on the center frequency of RES, which is needed to be estimated its locations.

Figure (3.21) Antenna Configuration on the Vehicle.

The hardware connection of the RF Tracking station is shown in Figure (3.22), where the component of this station are connected and powered by a 5 V and 3 A . Raspberry Pi3 is connected to the vehicle monitor through Wi-Fi to sets the tracking parameters of the station. SDR spectrum analyzer is interfaced with vehicle monitor to display the spectrum as shown in Figure (3.23).

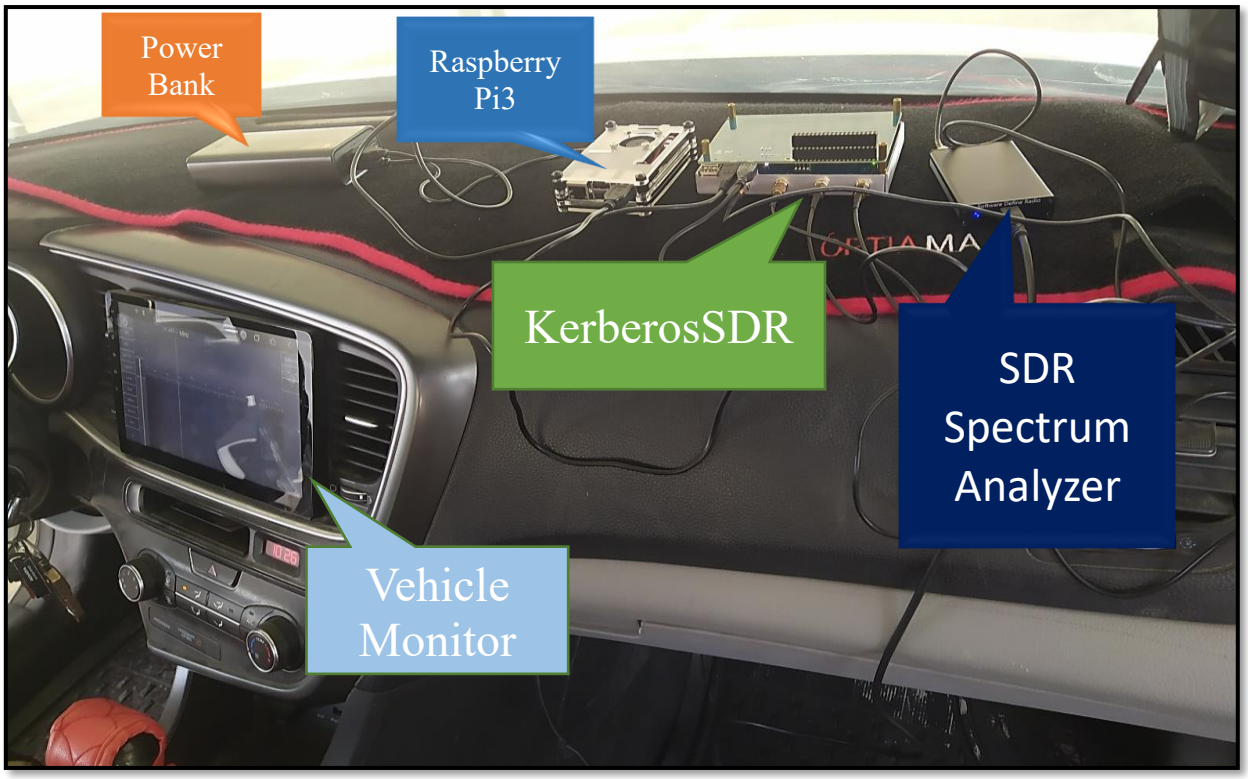


Figure (3.22) Hardware Connection of RF Tracking Station.

Figure (3.23) RF Tracking Station Monitor.

3.6 System Features

The system features can be summarized as follows:

- Real-time analysis with the highest possible rate and resolution.
- Search for any "new" emission quickly, measure its characteristics, and track its positions.
- Radio channel monitoring, demodulated signal listening and recording, and technical study of radio transmissions
- Measurement of RSS new RES and represents it on the MAP.
- Tracking and estimation of RES location.
- Because the internet is used to share data worldwide, data from the scanning region can be monitored at any time and from anywhere.
- A database to store the collected bearings about the scanning area.
- Even if there is no Internet connection, the system runs according to the most recent setting.
- The system can be turned in its parameters by Wi-Fi.
- This system accommodates an unlimited number of RF Tracking Stations.
- This system provides an unlimited number of GUI monitors.

Chapter Four

The Results and Discussions

4.1 Introduction

This chapter presents RFST system results in two phases: simulation results and practical results to comprehensively analyze the system.

The simulation results present the serial stages of RFST system configuration such as RF tracking station operation, tracking algorithms with different conditions, RF tracking station with various array shapes, and multiple RFST system designs. The practical results of implementing the proposed system are presented and discussed. It also is shown descriptions of the experimental cases, and obtained results are based on measurements done in the wireless facilities.

4.2 Simulation Results of the RFST System

This section presents the results of the simulation of the RFST system from a single RF Tracking Station to the RFST System.

4.2.1 Performance Results of Single RF Tracking Station

In this part, the tracking algorithms are evaluated by many tests, presented in the following sections.

4.2.1.1 Comparison of DOA Spatial Spectrum

The spatial spectrum method is utilized to evaluate the DOA algorithms' performance. Therefore, The DOA techniques are tested with simulation parameters f=990 MHz, N=4, N=8 and S=20. The spectrum results are shown in the Figures below.

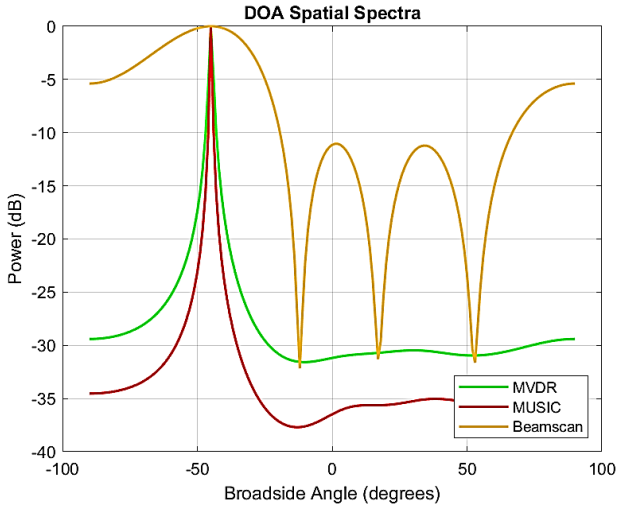
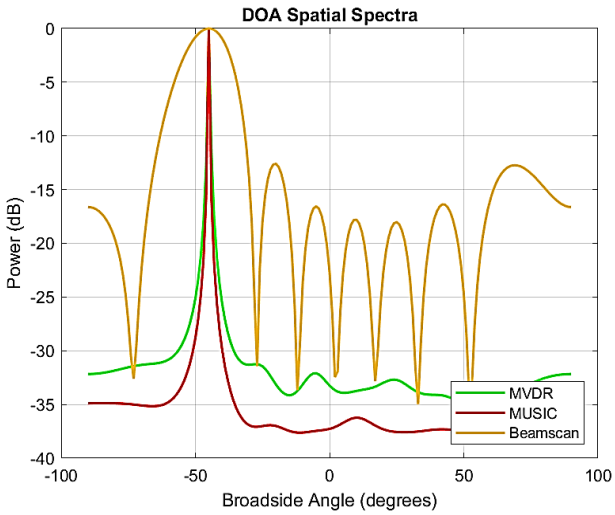


Figure (4.1). (a) One Signal with N=4.



(b) One Signal with N=8.

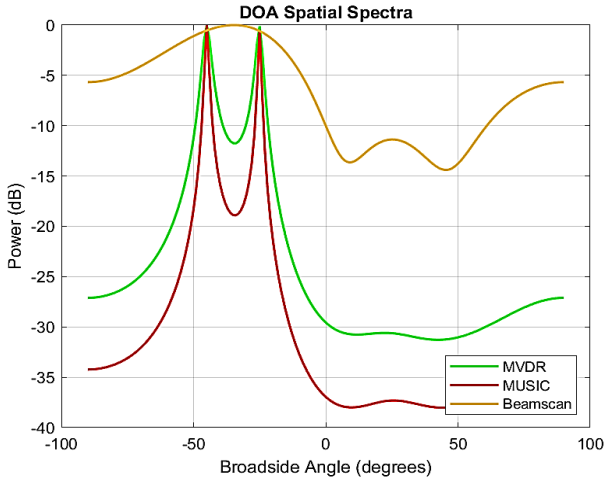
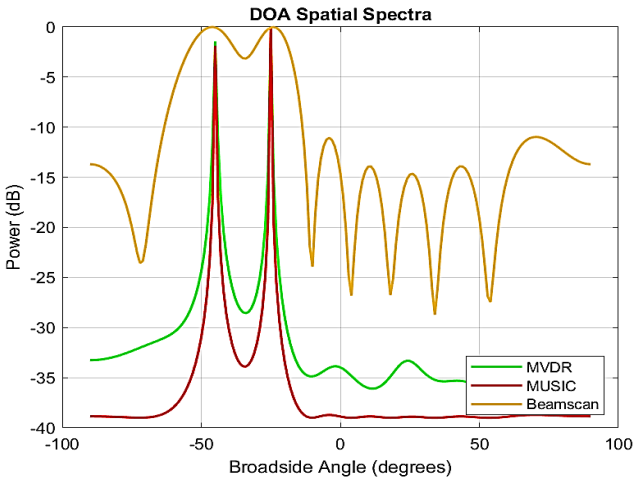


Figure (4.2). (a) Two Signals with N=4.



(b) Two Signals with N=8.

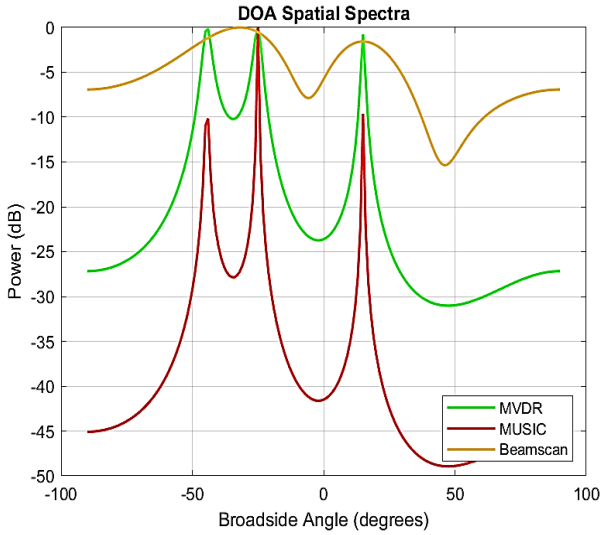
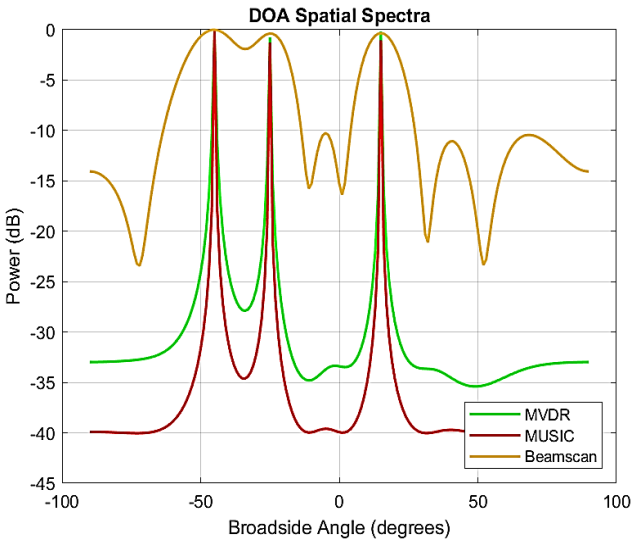


Figure (4.3). (a) Three Signal with N=4.



(b) Three Signal with N=8.

The results in the figures are obtained from three DOA techniques. In Figure (4.1).(a), Figure (4.2).(a) and Figure (4.3).(a), DOA algorithms with N=4 monitor the incoming signal in cases of 1, 2 and 3 signals, as well as in Figure (4.1). (b), Figure (4.2). (b), and Figure (4.3). (b), but the DOA algorithms operated with N=8. The Beamscan gives unreliable results in the figures above as the incoming signals increase.

However, the MVDR algorithm presents an acceptable response with some deviation, sometimes at angle convergence. However, the MUSIC algorithm provides the most robust and accurate result over the three DOA techniques. Therefore, the figures above show the best result obtained from the MUSIC algorithm in the case of using the spatial spectrum.

4.4.1.2 Comparison of DOA Algorithms

In the following, we will perform several tests on the performance of the different techniques of DOA to see the specific properties for every algorithm and decide the best method that provides a super-resolution of the tracking station.

The obtained results can be seen in several tables with several parameters at different frequency bands. The implemented tests are divided into the subsequent subsections, as shown in the following.

A-Frist Test: DOA Vs The Number of Incoming Signals

This test aims to show increasing the incoming signals on each estimation technique. Table 4.1 results are considered with 2 and 3 incoming signals with simulation parameters f=990MHz, S=20 and N=4. Assuming the signals are equal power= -4.4dB, Noise power equal =-40dB.

If the two signals are incoming, two directions must be estimated, and the same matter in three signals is significant to show the effect in each of the DOA algorithms. The estimated angles are approximately equal to imposed angles, which is required. All DOA techniques provide approximate estimation results. However, Root-MUSIC and Root-WSF give a high accuracy estimation**.**

Table 4.1. DOA vs The Number Of Incoming Signals.

No. of signals	**M=2**		**M=3**		
True Angle	**-45.2**	**26.51**	**70**	**10.7**	**-20.45**
Beamscan	-46	27	69	13	-24
MVDR	-46	26	70	12	-23
MUSIC	-46	26	70	11	-21
ESPRIT	-45.5	26.18	70.08	10.51	-20.8
Beamspace- ESPRIT	-45.6	26.19	70.23	10.52	-20.7
Root-MUSIC	-45.25	26.52	70	10.6	-20.6
Root-WSF	-45.19	26.51	70	10.65	-20.65

B-Second Test: DOA Vs The Number Of Snapshots

This test shows the varying number of snapshots on the DOA estimation. The comparison is shown in Table 4.2 presents the variation of snapshots numbers against estimated angles in each DOA method with simulation parameters f=990MHz, M=2 and N=4.

The variation is taken from 20, 50 and 100 to show, which gives high resolution in all possible states. The following results show that the accuracy of DOA estimation

is proportional to the number of snapshots. The best results are given from Root-MUSIC and Root-WSF algorithms, which prove the most accurate DOA estimation for all proposed snapshots.

Table 4.2. DOA vs The Number Of Snapshots.

No. of Snapshots	**S=20**		**S=50**		**S=100**	
True Angle	**-22**	**10.83**	**-22**	**10.83**	**-22**	**10.83**
Beamscan	-22	10	-23	12	-23	11
MVDR	-22	10	-22	11	-22	11
MUSIC	-22	10	-22	11	-22	11
ESPRIT	-22.046	10.38	-21.87	10.98	-21.95	10.89
Beamspace- ESPRIT	-22.9	10.36	-21.91	10.96	-21.95	10.91
ROOT-MUSIC	-22.81	10.49	-21.97	10.89	-21.98	10.85
ROOT-WSF	-22.91	10.51	-21.96	10.85	-22	10.86

C-Third Test: DOA Vs The Number Of Antenna Elements

This test aims to evaluate each DOA estimation algorithm's performance depending on the number of the antenna in arrays. As shown Table 4.3 presents the changes in antenna numbers against estimated angles in each tracking techniques with simulation parameters f=990MHz, S=20 and M=2.

The changing is taken from 4, 8 and 16 in antenna elements. It is shown that the higher-resolution DOAs estimation is obtained by increasing the number of antenna elements in the array. Therefore, better results are shown from Root-WSF and Root-

MUSIC with changing the number of antenna elements. However, ROOT-WSF is the more accurate method.

Table 4.3. DOA vs The Number Of Antenna.

No. of Antenna	**N=4**		**N=8**		**N=16**	
True Angle	**-34.46**	**19.9**	**-34.46**	**19.9**	**-34.46**	**19.9**
Beamscan	-33	19	-35	20	-34	20
MVDR	-35	20	-35	20	-34	20
MUSIC	-34	20	-35	20	-34	20
ESPRIT	-33.88	19.94	-34.76	19.8	-34.5	19.79
Beamspace- ESPRIT	-33.81	20.1	-34.76	19.8	-34.42	19.8
ROOT-MUSIC	-34.3	20	-34.63	19.88	-34.46	19.9
ROOT-WSF	-34.50	19.9	-34.63	19.89	-34.46	19.9

4.2.1.3 Comparison of DOA with Various Application

This study is aimed to evaluate the performance of DOA techniques with different applications such as GSM and Wi-Fi. As shown in Table 4.4, the comparison study presents the variation of frequency bands against estimated angles in each DOA technique with simulation parameters S=20, M=2 and N=8.

All algorithms give acceptable results with all applications with this condition, but it shows sensitivity to the operating frequency seen in Table 4.4. However, Root-WSF and Root-MUSIC methods prove better results with a slight error.

Table 4.4. Comparison of DOA with Various Applications.

Application	**GSM**		**Wi-Fi**	
Frequency	**990MHz**	**1800MHz**	**2.4GHz**	**5GHz**
True Angle	**-37**	**-12.67**	**25.75**	**29.23**
Beamscan	-38	-13	25	29
MVDR	-38	-13	25	28
MUSIC	-38	-13	25	29
ESPRIT	-36.89	-12.18	25.30	28.49
Beamspace- ESPRIT	-37.94	-12.39	24.92	28.04
ROOT-MUSIC	-36.99	-12.70	25.32	28.57
ROOT-WSF	-36.99	-12.67	25.37	28.74

4.2.1.4 Comparison of DOA in Multipath Environment

This test aims to try the high-resolution DOA algorithms to estimate coherent sources in multipath situations where various sources are coherent or correlated. Unfortunately, many DOA estimation algorithms give wrong results or fail estimation.

This test takes two incoming signals models with six multipath reflections of the first and second source with simulation parameters f=990 MHz, N=8 and S=20. As a result, it has magnitudes equal to 0.7, 0.33 and 0.4 from the first source, and 0.5, 0.25, and 0.22 from the second source.

Table 4.5 shows that ROOT-MUSIC give fail estimation and unacceptable results. But, the Corrected ROOT-MUSIC used spatial smoothing to prove good

results. On the other hand, Root-WSF is particularly desirable in the case of coherent sources because it does not need spatial smoothing for properly estimating the DOAs when the number of sources is known compared to the other methods.

Table 4.5. Comparison of DOA in Multipath Environment.

Sources	Magnitude	True Angles	ROOT-WSF	ROOT-MUSIC	ROOT-MUSIC With Spatial Smoothing
Frist Source	1	-17	-17	-17.834	-17
	0.7	35	34.93	**23.44**	34.94
	0.33	10	10	**-69.18**	9.998
	0.4	22	22.02	22	22.02
Second Source	1	25	25.064	20.33	25.06
	0.5	11	11.13	**44.04**	11.14
	0.25	-40	-39.979	**-71.48**	-39.98
	0.22	0	0.0004	0.0015	0.0125

4.2.1.5 Tracking the Moving Source

From this test, it is shown that the ability to use the DOA algorithm as a tracking algorithm for moving sources. For example, in Table 4.6 when the source gives a signal at a specific frequency and angle of arrival with simulation parameters f=990MHz, N=8 and S=20.

The DOA algorithm can estimate and track its angle of arrival to get the new angle where the source moves from one point to other. The best results are given from Root-MUSIC and Root-WSF algorithms, which prove the most accurate DOA tracking for all possible directions.

Table 4.6. Tracking the Angle of the Moving Source.

Moving Source Angles	ROOT-WSF	ROOT-MUSIC
0	-0.06	-0.12
15	15.04	15.17
30	29.87	29.77
45	45.13	45.13
60	60.16	60.24
75	74.94	74.82
90	87.85	86.69

4.2.1.6 Results Discussion of RF Tracking Station Simulation

All techniques are essential for the RFST system, which prove useful performance and reasonable calculation complication. Root-WSF and Root-MUSIC methods give the strongest resolution estimation over the various effects taken in this study. Therefore, these two techniques will be the most desirable in tracking operations. In terms of super regulation, the ROOT-WSF method proves the higher resolution of DOA estimation.

4.2.2 Tracking Algorithms with Conditions Simulation Results

In this simulation experiment, a uniform linear array has been considered with the number of snapshots (20), eight array elements, and λ/2 the spacing of sub antenna.

4.2.2.1 The Number of Incoming Signals

This part evaluates the two DOA algorithms in cases 3 and 5 incoming signals, considering the operating frequency of 990MHz. The remaining parameters remain unchanged. Table 4.7 and Table 4.8 show that the simulation results show that ROOT-WSF estimation accuracy surpasses the ROOT-MUSIC estimation accuracy. As well, ROOT-MUSIC proves good results in both cases.

Table 4.7. DOA with 3 Incoming Signals.

True Angle	**-45.2**	**26.51**	**63**
ROOT-MUSIC	-45.101	26.302	62.9707
ROOT-WSF	-45.20	26.5177	62.9809

Table 4.8. DOA with 5 Incoming Signals.

True Angle	**-50**	**-17.22**	**0**	**20.81**	**40**
ROOT-MUSIC	-49. 81	-17.212	0.0008	20.5004	39.9504
ROOT-WSF	-49.992	-17.226	0.0001	20.8104	39.9902

4.2.2.2 Correlated and Uncorrelated Signals

This test evaluates the DOA estimation methods for both correlated and uncorrelated signals at a 5GHz. The remaining parameters remain unchanged.

Table 4.9. Correlated and Uncorrelated Signals.

Frequency	True Angle	-50	-17.22	0	20.81	40
Correlated	ROOT-MUSIC	-46.35	-12.91	0.006	17.3	30
	ROOT-WSF	-49.992	-17.226	0.0001	20.81	39.99
Uncorrelated	ROOT-MUSIC	-49.981	-15.49	0.008	20.80	39.95
	ROOT-WSF	-49.998	-17.22	0.0001	20.81	39.99

The simulation results in Table 4.9 demonstrate that the ROOT-WSF algorithm is more accurate than the ROOT-MUSIC algorithm.

4.2.2.3 Space Converging Sources

Tables 4.10 and Table 4.11 discuss two difficult conditions involving very close sources to converge in space. It is considered two spacing of 5° and 1° with an angle of arrival (AOA) of Θ1=-10°, Θ2= -5°, Θ3=0°, Θ4=5°, Θ5=10°, and 5° with an angle of arrival (AOA) of Θ1=0°, Θ2= 1°, Θ3=2°, Θ4=3°, and Θ5=4°. The remaining parameters remain unchanged. The frequency is 5GHz, with signals are correlated.

Table 4.10. Converging Sources with 5° Space.

True Angle	-10	-5	0	5	10
ROOT-MUSIC	-12.01	-5.5	0.7	7	11
ROOT-WSF	-10.2	-5.4	0.03	5.01	10.03

Table 4.11. Converging Sources with 1° Space.

True Angle	0	1	2	3	4
ROOT-MUSIC	0.07	-25	1.25	2.69	3.98
ROOT-WSF	0.04	0	1.6	2.8	4

The simulation results show that ROOT-MUSIC cannot estimate the DOA in both proposed cases. However, ROOT-WSF can accurately estimate all angles in 15° to 5° spaced angle differences.

4.2.2.4 The Different Frequency Signals

In this section, the evaluation of two DOA depends on the variation of the operating frequency. In this context. It is considered as having two frequencies, 990 MHz and 5GHz, and five incoming signals as shown in Table (4.11).

Table 4.12. The Different Frequency Signals.

Frequency	True Angle	-50	-17.22	0	20.81	40
900 MHz	ROOT-MUSIC	-43.25	-17.2	0.3	20.80	39.97
	ROOT-WSF	-49.992	-17.22	0.0001	20.81	39.99
5 GHz	ROOT-MUSIC	-49.981	-17.23	0.001	20.80	38
	ROOT-WSF	-49.992	-17.212	0.0001	20.81	39.99

In Table 4.12, ROOT-MUSIC algorithms demonstrated increased sensitivity as a function of operating frequency, which is evident for lower (990 MHz) and higher operating frequencies (5GHz). ROOT-MUSIC detected three AOA but not the fourth or fifth. However, the ROOT-WSF algorithm accurately estimated all the DOA in both cases.

4.2.2.5 The Accuracy of Tracking Algorithms

Based on the obtained results, the accuracy can be defined for the algorithms, as shown in Table 4.13.

$$Accuracy = 100\% - \left|\left(\frac{\theta e - \theta a}{\theta a}\right) * 100\%\right| \quad (4.1)$$

Where θe is the estimated angle and θa is the actual angle.

Table 4.13. The Accuracy of Tracking Algorithms under Conditions.

Conditions		ROOT-MUSIC	ROOT-WSF
Correlated and Uncorrelated	**Correlated**	75%	100%
	Uncorrelated	90%	100%
Space Converging Sources	**5° Space**	80%	95%
	1° Space	70%	75%
Different Frequency	**990 MHz**	87%	100%
	5 GHz	95%	100%

From the accuracy side, ROOT-WSF estimation has the highest estimation accuracy. ROOT-WSF provided an accuracy of about 95% in all possible cases.

4.2.2.6 Comparison of Tracking Algorithms with Conditions

From this test, it can be evaluated the considered tracking Algorithms depend on specific criteria such as the number of incoming signals, Correlated and uncorrelated signals, Converging sources, and the different frequency signals. ROOT-WSF algorithm proves the super-resolution for tracking the DOA under conditions. This analysis is very important for the RFST system because of the selection of tracking algorithms related to the resolution of the RFST system.

4.2.3 RF Tracking Station with Various Arrays Configuration Results

The study considered several tests by changing different affected array geometry and taking the following parameter for all tests such as frequency 990MHz, the number of incoming signals (3), the number of antennas (8), and the number of the snapshots (20). All array geometries have been considered in these simulations with the sub-antenna spacing of $\lambda/2$.

4.2.3.1 DOA Spatial Spectrum

Arrays geometries performance is evaluated using DOA spatial spectrum techniques. Therefore, the test results are considered based on the power graphs against the scanned angles. The simulation results are shown for the following three types of arrays geometries. The simulation deals with a one-dimensional array and two-dimensional array, respectively.

A- ULA - DOA Spatial Spectrum:

This part analyzes the performance of ULA with three DOA estimation algorithms, where the simulation was implemented with Beamscan, MVDR, and MUSIC algorithms. Using ULA, the DOA algorithms scan all the angles defined in the simulation. Then, as shown in Figure (4.4), it is the graph of power against the scanned angles.

It has three curves for three DOA estimations to show the peak graph of DOAs mentioned above and configure a general idea of the chart, which can be used for data analysis and estimate the correct angle.

In ULA-DOA estimation, the graph peaks are shown in Figure (4.4), where the incident angles are 25, 40, and 55 degrees.

Three power peaks against incident angles are distinguished easily in the MUSIC graph because peaks are near 0 dB to lower than -30dB. However, incident angles are difficult to differentiate from Beamscan and MVDR curves because all power values are relatively close to the peak. Therefore, Figure (4.4) shows the best result obtained from the MUSIC algorithm in using ULA.

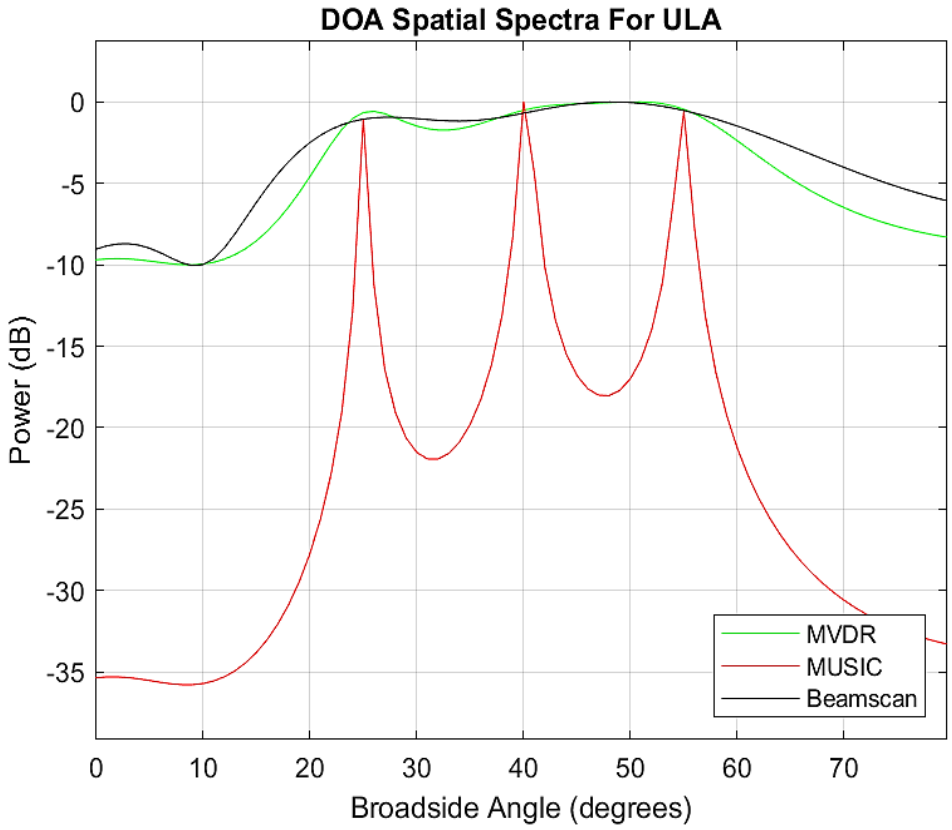


Figure (4.4) ULA Spatial Spectrum.

B- URA -DOA Spatial Spectrum:

This part analyzes the performance of URA for three DOA estimation algorithms: Beamscan, MVDR, and MUSIC algorithms. Using URA, the DOA algorithms scan two-dimensional azimuth and elevation angles. Figure (4.5) are the graph of power against the scanned angles, which has three curves for the DOA estimation to show data analysis and estimate the correct angle.

In URA-DOA estimation, the peak graph of two dimensions and three dimensions is shown in figures mentioned above, where the incident angles are 25, 40, and 55 degrees. There are three power peaks against incident angles, which can be organized only in MUSIC curve.

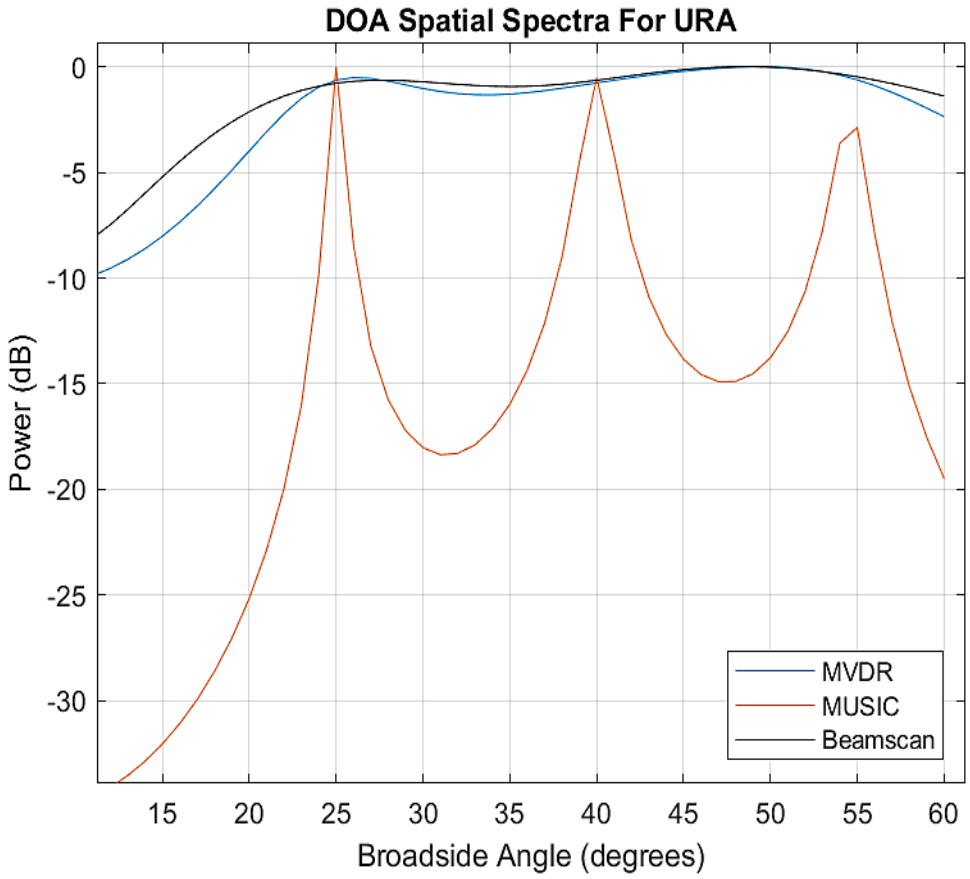


Figure (4.5) URA Spatial Spectrum.

C-UCA DOA spatial spectrum

This part analyzes the performance of UCA for three DOA estimation algorithms: Beamscan, MVDR, and MUSIC algorithms. Using UCA, the DOA algorithms scan the angles in two-dimensional azimuth and elevation angle. Figure (4.6) has three curves for DOA estimation to show data analysis and estimate the correct angle.

In UCA- DOA estimation, the peak graph of two dimensions and three dimensions is shown in Figure (4.6), where the incident angles are 25, 40, and 55 degrees. There are three power peaks against incident angles, which can be organized directly in the DOA curve.

In Figure(4.6), Beamscan peaks are near 0dB to lower than -5 dB, MVDR peaks are near 0dB to lower than -8 dB, and MUSIC peaks are near 0dB to drop than -35dB. Therefore, the three DOA algorithms mentioned above proved a good estimation resolution with URA. However, Beamscan and MVDR curves are more difficult to distinguish incoming signals' angles than MUSIC.

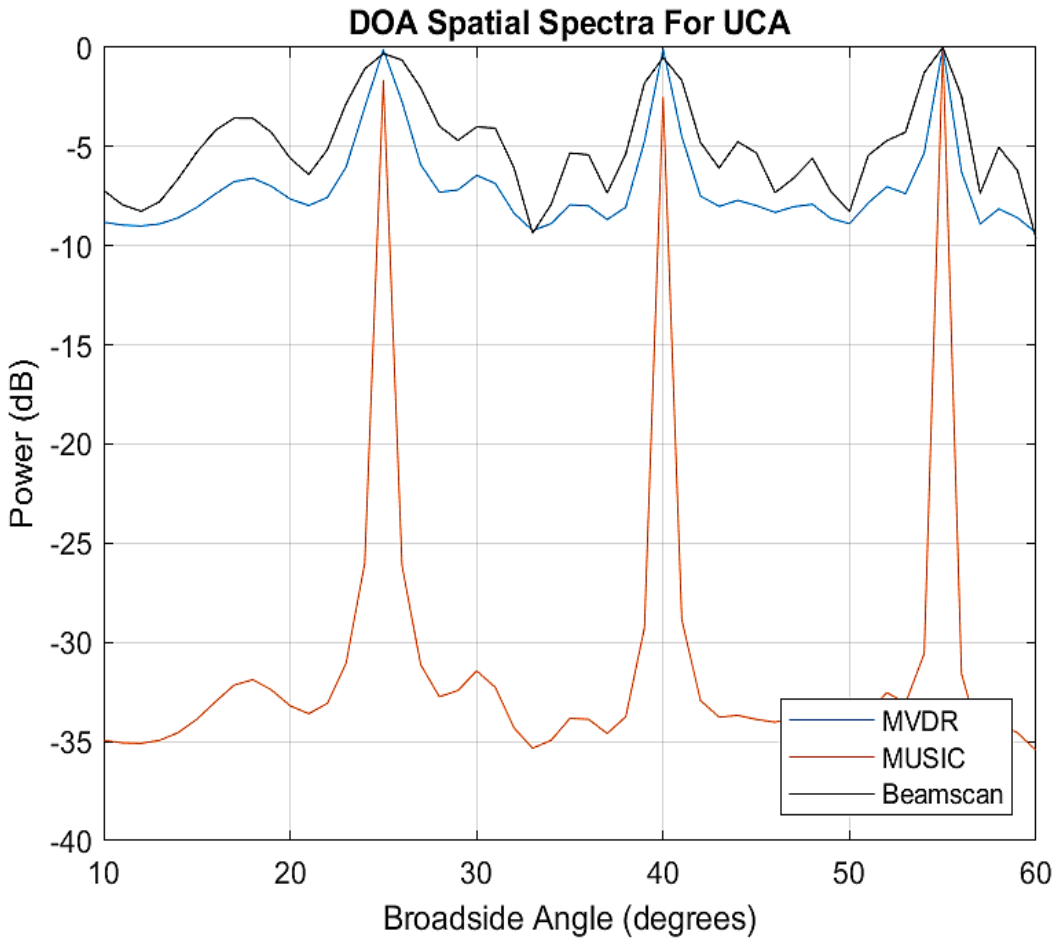


Figure (4.6) UCA Spatial Spectrum.

.

4.2.3.2 Evaluation of Array with Tracking Algorithms

This section shows the effect of array antenna geometries on DOA estimation techniques. The simulation parameters considered are frequency (990GHz), the number of snapshots (20), and the number of elements (8). It was thought that three signals were incoming. Therefore, three angles must be estimated.

The obtained results are shown in Table 4.14, Table 4.15, and Table 4.16 considering three incoming signals with angles 25, 40, and 55 degrees must be estimated with each DOA technique.

Table 4.14, MUSIC, ROOT-MUSIC, and ROOT-WSF proved a correct estimation compared with Beamscan and MVDR, which fail in estimation.

Table 4.14. Tracking Algorithm and ULA.

No.	**Array Configuration**	**ULA**		
	True Angle	**25**	**40**	**55**
1	Beamscan	27	3	48
2	MVDR	26	-51	51
3	MUSIC	25	40	55
4	ROOT-MUSIC	24.75	39.82	53.77
5	ROOT-WSF	24.89	40	54.92

In Table 4.15, MUSIC demonstrated a good estimation resolution compared with Beamscan and MVDR, which fail in the estimation. ROOT-MUSIC and ROOT-WSF don’t work with URA configuration.

Table 4.15. Tracking Algorithm and URA.

No.	Array Configuration	URA		
	True Angle	**25**	**40**	**55**
1	Beamscan	28	-	48
2	MVDR	26	-	49
3	MUSIC	25	40	55
4	ROOT-MUSIC	-	-	-
5	ROOT-WSF	-	-	-

In Table 4.16, MVDR and MUSIC give a correct estimation compared with Beamscan and ROOT-MUSIC, which are incorrect. Therefore, MUSIC provides a suitable performance and has a high accuracy estimation with all array antenna geometries.

Table 4.16. Tracking Algorithm and UCA.

No.	Array Configuration	UCA		
	True Angle	**25**	**40**	**55**
1	Beamscan	12	32	55
2	MVDR	25	40	55
3	MUSIC	25	40	55
4	ROOT-MUSIC	25.69	175	52.62
5	ROOT-WSF	-	-	-

4.2.3.3 Comparison of Array Configurations Performance

This test presented a detailed analysis of array geometries performance effected on tracking algorithms estimation. According to array configuration, it is sure that the direction of the incident angle is affected by signal detection. ULA operates in one dimension state such the incident angle comes close to 0 degrees or 180 degrees, the ability of ULA detection will decrease significantly. But, URA and UCA can operate in two dimensions state. They can prove a constant DOA estimation in the range from 0 ° to 360 ° to estimate the RF source of two-dimensional angular.

Therefore, from the analysis made above, It is concluded that ULA had a good performance in accuracy and detection ability with MUSIC, ROOT-MUSIC, and ROOT-WSF over the various affected is taken in this study failed with Beamcans and MVDR methods.

However, URA has relatively poor properties in simulation and gives correct estimation only with the MUSIC method. On the other hand, UCA proved a good estimation resolution with MVDR and MUSIC methods. Therefore, it can be declared that ULA and UCA provide a better performance as compared with URA, which needs more elements for correct estimation. MUSIC estimation also shows suitable performance in all simulation cases with array geometries types.

4.2.4 Results of the RFST System Configurations

This test presents the results of the designed RFST system. ULA was considered in each RF Tracking Station with N=4 and $\lambda/2$ the sub- spacing of each element, where λ is the wavelength of incoming signals. The signals propagated at a frequency 990MHz and the speed of light $3*10^8$m/s. Three tracking algorithms are used: MUSIC, ROOT-MUSIC, and ROOT-WSF to get the accurate tracking process.

4.2.4.1 One RF Tracking Station

The results of a single RF Tracking Station were considered in this test. The RES to be tracked is available in this station's detection area, which collects the DOA information regarding the received signal from the RES to the operations center by internet. The tracking scenarios considered for the system are illustrated in the figures below, where the tracking station is available in the detection area and the RES with the actual distance 780m on angle 26.5 degrees from the tracking station. The station's location is fixed as a reference station of the system. The RES is available in the detection area of the tracking station. For RES signals impinging at the position as shown in Table 4.17.

Table 4.17. Actual Coordinates of One Station and RES.

NO.	Test Items	Latitude	Longitude
1	RES	32.505578	44.420545
2	Station 1	32.499283	44.416813

RF Tracking estimations were obtained in Table 4.18, where three tracking algorithms are used for tracking the angle of RES.

Table 4.18. One RF Tracking Station Estimation.

NO.	Tracking Algorithms	Estimated Angles
1	MUSIC	27
2	ROOT-MUSIC	26.56
3	ROOT-WSF	26.56

The RFST system represents the tracking data on the Google Map, as shown in Figure (4.7).

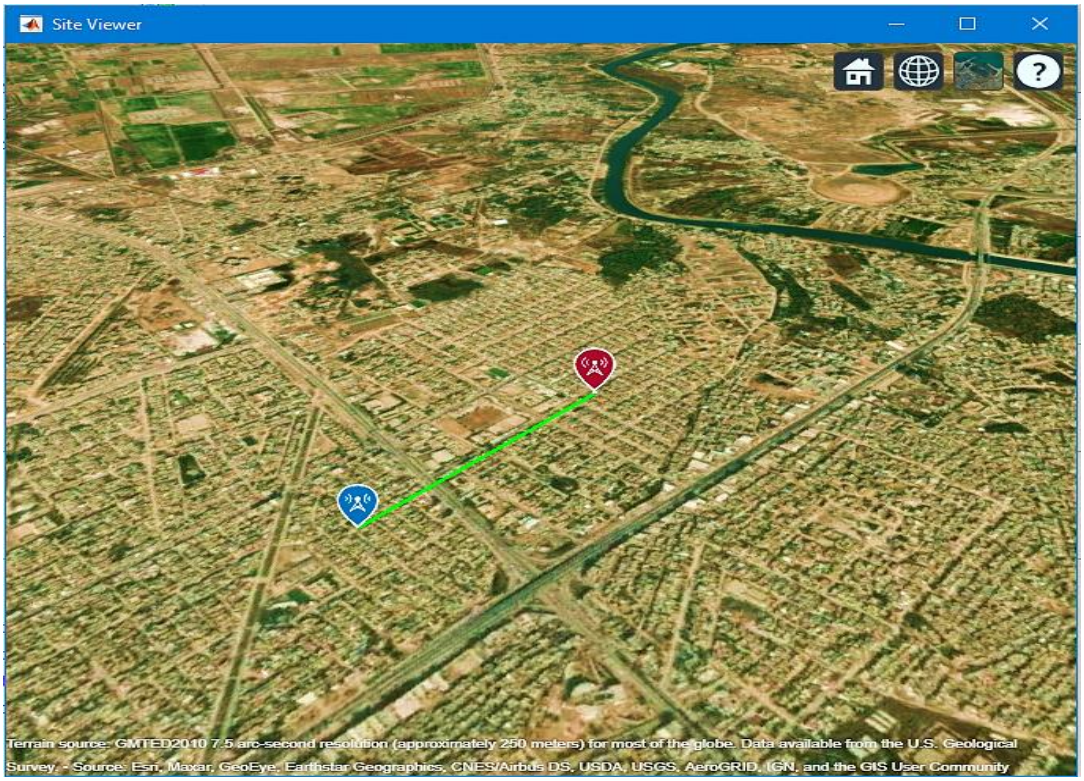


Figure (4.7) Single Tracking Station Map Representation.

4.2.4.2 Two RF Tracking Stations

Two RF Tracking stations were considered in this test. There are two tracking stations available in the detection area and the RES with the actual distance of 780m on angle 26.5 degrees from the first tracking station. The first station is considered a reference station for the second station. The tracking data comes from two stations. Therefore, it can be tracking the RES position. The RES is available in the detection area of the tracking station. For RES signals impinging at the position as shown in Table 4.19.

Table 4.19. Actual Coordinates of Two Stations and RES.

NO.	Test Items	Latitude	Longitude
1	RES	32.505578	44.420545
2	Station 1	32.499283	44.416813
3	Station 2	32.508276	44.416813

Two RF Tracking station with RES is represented by the RFST system on the Google Map, as shown in Figure (4.8).

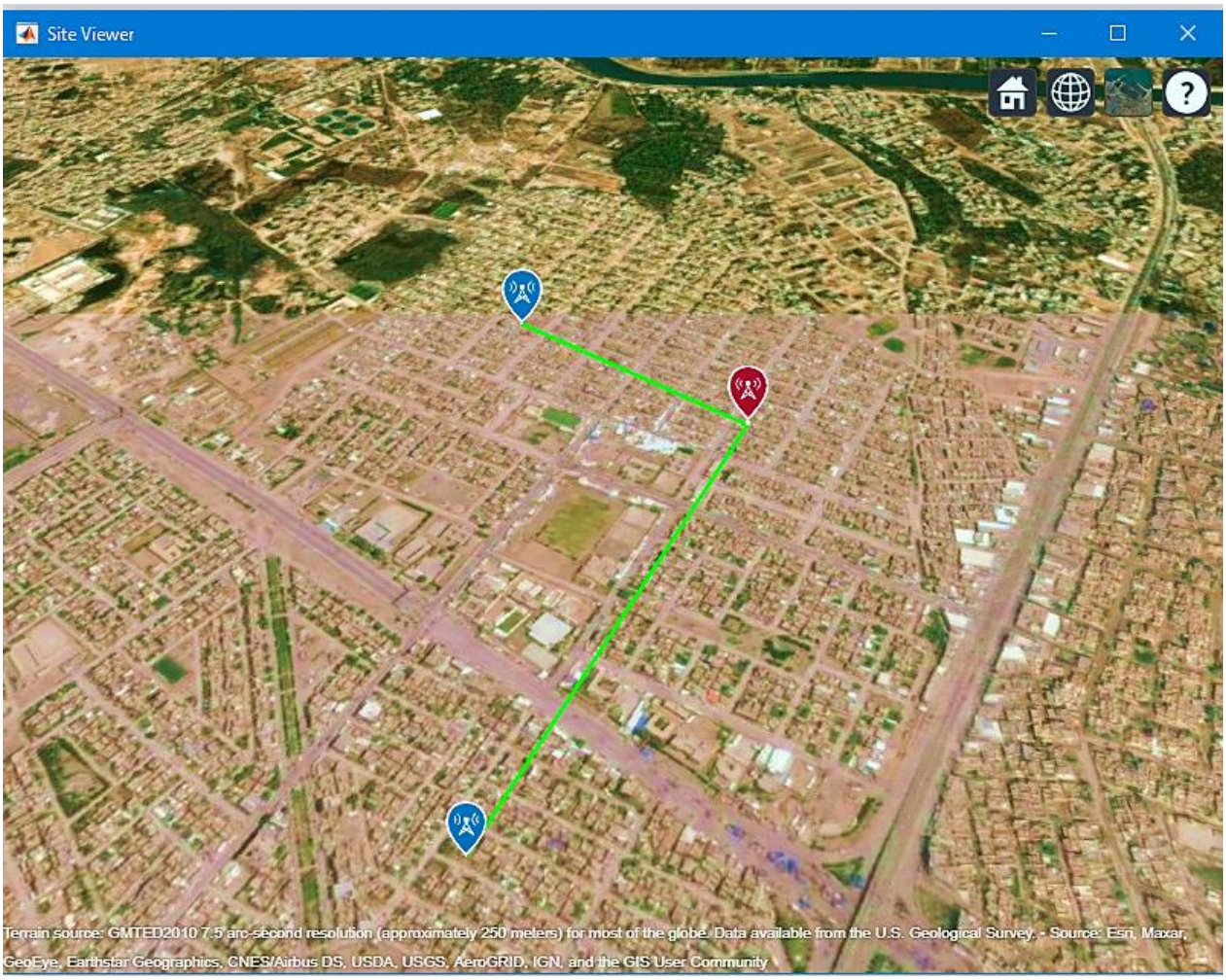


Figure (4.8) Two Tracking Stations Map Representation.

Two RF Tracking stations estimations were obtained in Table 4.20, where three tracking algorithms are used for tracking the angle of RES. Error distance is calculated between actual and estimated positions of the RES.

$$Error\ distance = \sqrt{(xa - xe)^2 + (ya - ye)^2} \quad (4.2)$$

Where (xa, ya) is the actual position and (xe, ye) is the estimated position.

Table 4.20. Two RF Tracking Stations Estimation.

NO.	Tracking Algorithms	Estimated Position		Error Distance (m)
		Latitude	Longitude	
1	MUSIC	32.505712	44.420545	14.8
2	ROOT-MUSIC	32.505631	44.420655	12.1
3	ROOT-WSF	32.505622	44.420637	10.3

4.2.4.3 Three RF Tracking Stations

Three RF Tracking stations were considered in this test. Three tracking stations are available in the detection area and the RES with an actual distance of 780m on angle 26.5 degrees from the first tracking station. The tracking data comes from three stations. Therefore, it can be tracking the RES position from three stations. The actual position of the three stations and RES are shown in Table 4.21.

Table 4.21. Actual Coordinates of Three Stations and RES.

NO.	Test Items	Latitude	Longitude
1	RES	32.505578	44.420545
2	Station 1	32.499283	44.416813
3	Station 2	32.508276	44.416813
4	Station 3	32.499283	44.427477

The tracking data of the three RF Tracking stations with RES is represented by the RFST system on the Google Map, as shown in Figure (4.9).

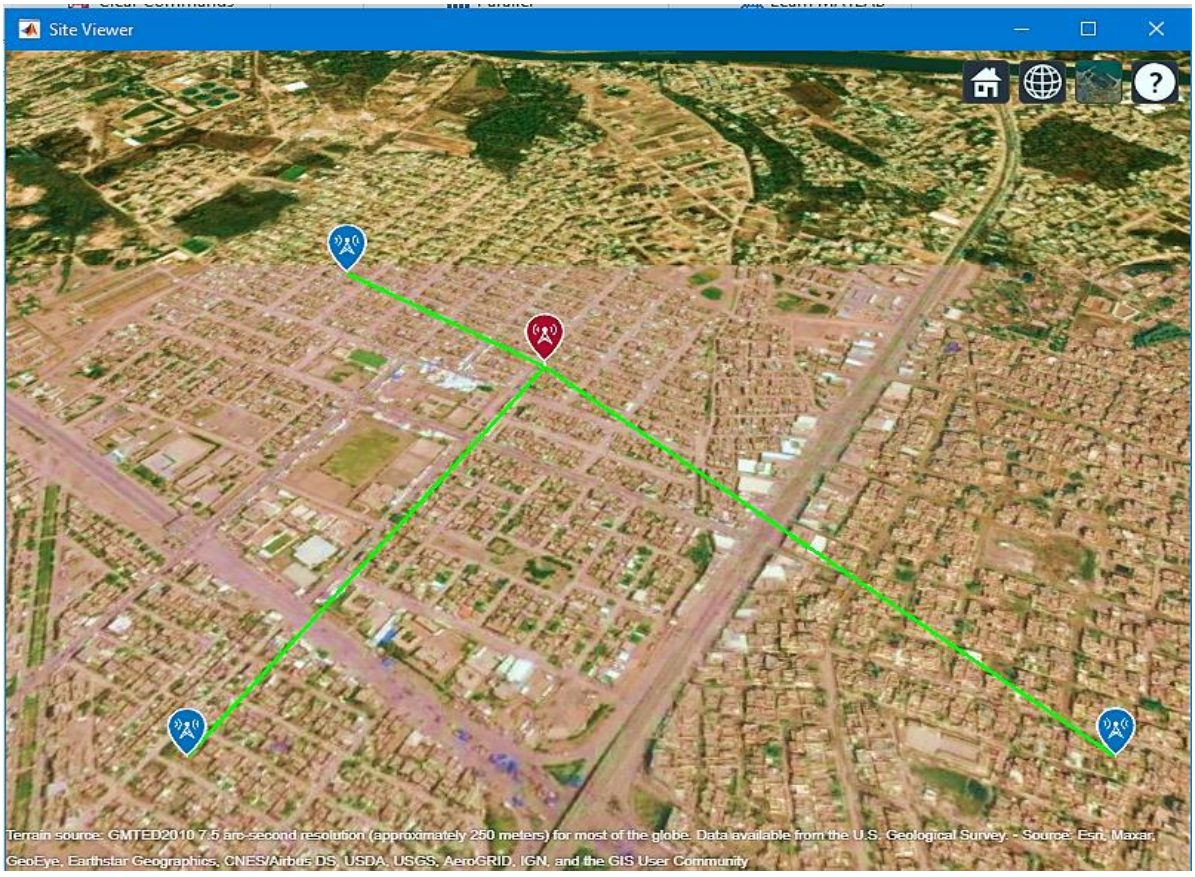


Figure (4.9) Three Tracking Stations Map Representation.

Three RF Tracking stations estimations are shown in Table 4.22. Three algorithms are used for tracking. The error distance is calculated for each algorithm.

Table 4.22. Three RF Tracking Stations Estimation.

NO.	Tracking Algorithms	Estimated Position		Error Distance (m)
		Latitude	Longitude	
1	MUSIC	32.505660	44.420643	12.65
2	ROOT-MUSIC	32.505604	44.420612	7.3
3	ROOT-WSF	32.505603	44.420588	5.1

4.2.4.4 Four RF Tracking Stations

Four RF Tracking stations are defined in this test. These four tracking stations surround the detection area and the RES with an actual distance of 780m on the angle 26.5 degrees from the first tracking station. The RES position is tracked from four stations. The actual position of the four stations and RES are shown in Table 4.23.

Table 4.23. Actual Coordinates of Four Stations and RES.

NO.	Test Items	Latitude	Longitude
1	RES	32.505578	44.420545
2	Station 1	32.499283	44.416813
3	Station 2	32.508276	44.416813
4	Station 3	32.499283	44.427477
5	Station 4	32.508276	44.427477

Four RF Tracking stations estimations are presented in Table (4.24), with three algorithms are used with calculating the error distance for each algorithm.

Table 4.24. Four RF Tracking Stations Estimation.

NO.	Tracking Algorithms	Estimated Position		Error Distance (m)
		Latitude	Longitude	
1	MUSIC	32.505641	44.420620	10.13
2	ROOT-MUSIC	32.505602	44.420559	3
3	ROOT-WSF	32.505594	44.420554	2

Four RF Tracking stations with their tracking data and RES is shown by the RFST system on the Google Map, as shown in Figure (4.10).

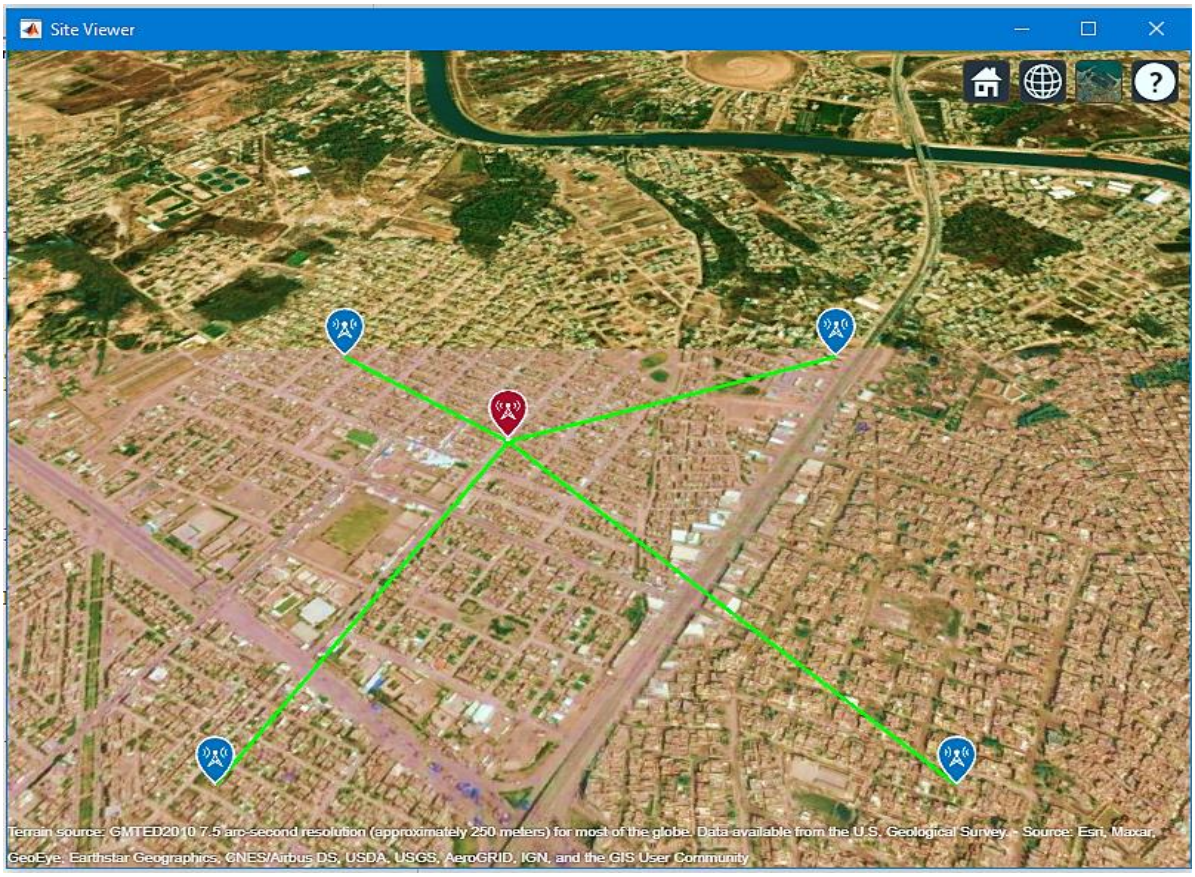


Figure (4.10) Four Tracking Stations Map Representation.

4.2.4.5 Fixed RF Tracking Stations with Moving the RES

This simulation scenario presents the proposed system's performance that uses multiple connected RF Tracking Stations with continuous monitoring for moving the RES. The tracking data are instantaneously collected from each station by the operations center to illustrate the RES's moving path. Four RF Tracking stations are defined in this test. These four tracking stations surround the detection area, and the RES has continuous moving at the path, which is drawn in the simulation. ROOT-WSF is only considered the tracking algorithm with ULA.

The RFST system on the Google Map represents four RF Tracking stations with Moving RES, as shown in Figure (4.11). This simulation result calculates the RES speed by speed equal distance between two estimated positions over processing time between two positions evaluated by the RFST system.

$$speed = 60m/1s = 60m/s = 216Km/h$$

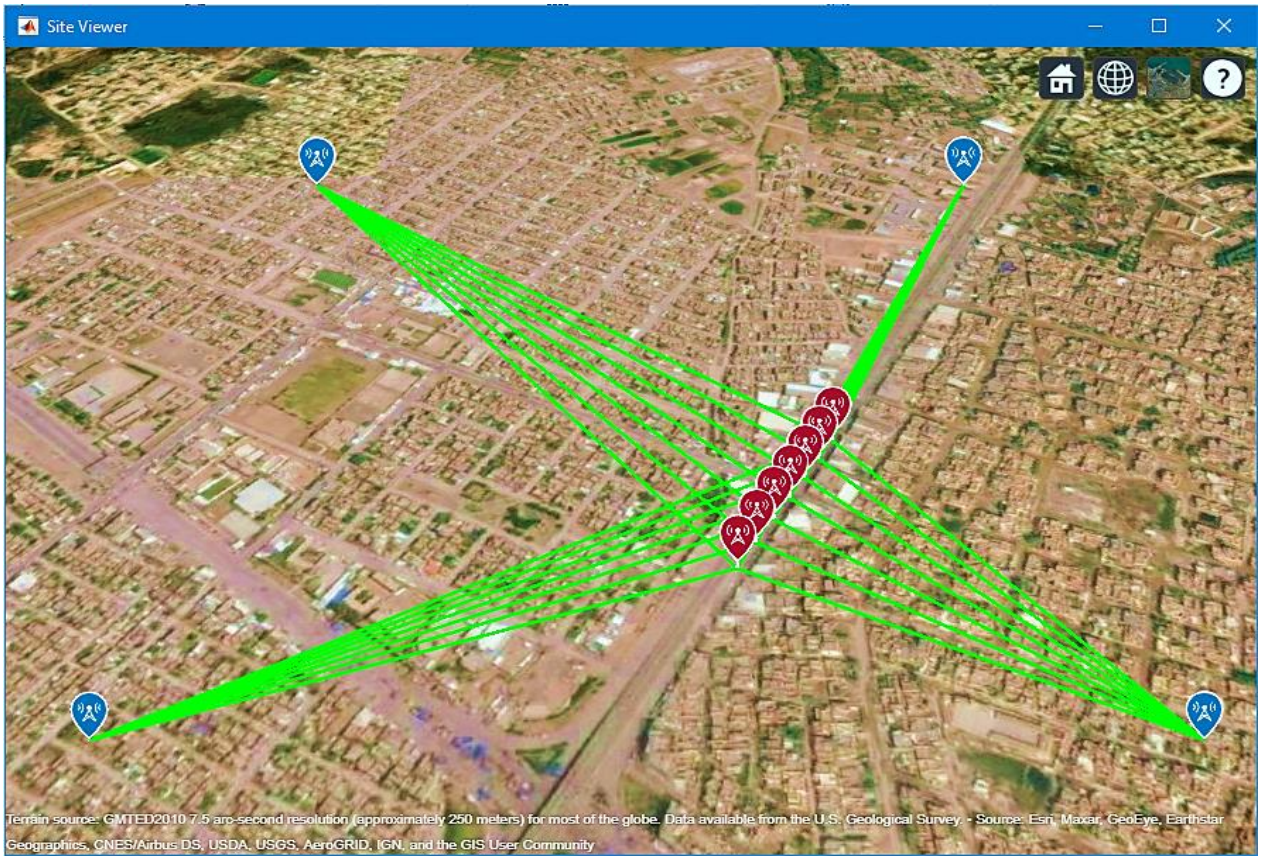


Figure (4.11) Fixed Four Tracking Stations with Moving RES.

4.2.4.6 Moving RF Tracking Stations with Fixed the RES

This simulation scenario presents another case of the proposed system, which uses moving RF Tracking Stations with continuous monitoring to fix the RES. Each tracking station with constant moving around the detection area can instantaneously collect the tracking data from different locations to the operations center. The moving path of four RF Tracking stations is defined in this test. Again, ROOT-WSF has only considered the tracking algorithm with ULA.

Four RF Tracking stations with their tracking data and RES is shown by the RFST system on the Google Map, as shown in Figure (4.12).

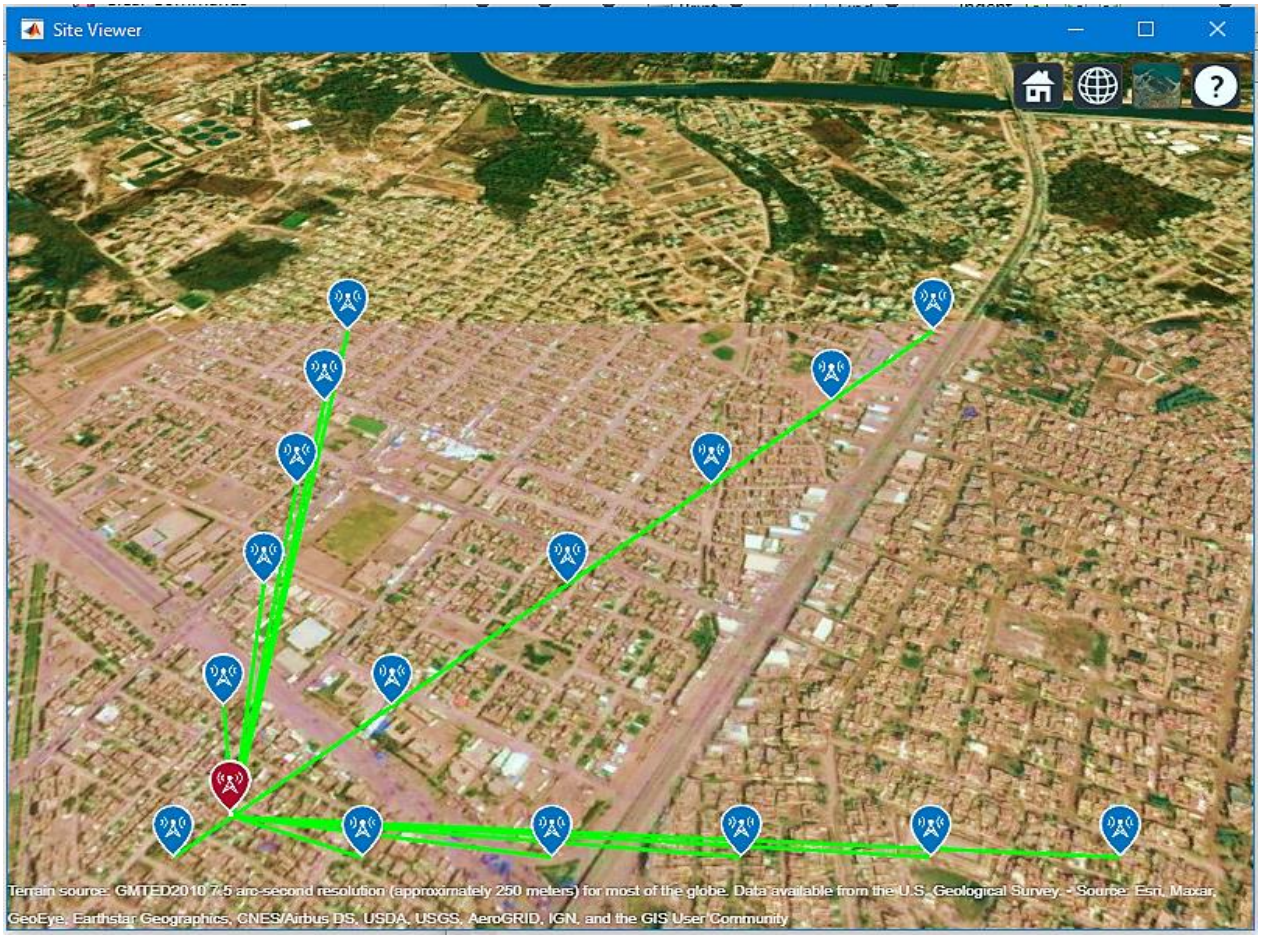


Figure (4.12) Moving RF Tracking Stations with Fixed RES.

4.2.4.7 The Discussion of RFST Simulation Results

This RFST system simulation presents the performance of the RFST system with various tracking scenarios and a different number of connected RF Tracking stations based on GPS coordinates.

The system results are evaluated by error distance, the difference between the latitude and longitude between actual and estimated RES positions. The minimum error distance is 2m when using the WSF tracking algorithm.

4.3 Practical Results of the RFST System

This section presents the results of the practical implementation of the RFST system to the configuration stages and operations of the RFST System.

4.3.1 The Results of Implemented RF Tracking Station

In this part, the results of implemented RF Tracking Station tracking algorithms are evaluated by many tests, presented in the following sections. The operating frequency of RTL-SDR is set to 990MHz, the sampling rate is to 1024KHz, the four receiving gain is 28dB, the FIR filter bandwidth is set 30kHz with 100 taps of the FIR, and the Decimation is set to 1. It can be display-processing results with operations center GUI.

The experiment utilizes a miniature open-source transceiver that can transmit various signals with different frequency bands to evaluate the system's overall performance. Tracking information is determined that the HackRF-One SDR emits the signal with a frequency of 990MHz. An antenna array receives the signal, and each receiver of the KerberosSDR measures the signal strength. As a result, the spectrum of the RES can be displayed together for the four channels, as shown in Figure (4.13).

The channel is received the source signal with an operating frequency of 990MHz, which is a GSM band. This frequency band is essential, and this band should always be safe. Furthermore, the practical test should be done with no interference with the other sources, which operated in the same frequency band close to the experimental position. Therefore, this experiment environment is preferable to track the RES to estimate the accuracy of the RFST system.

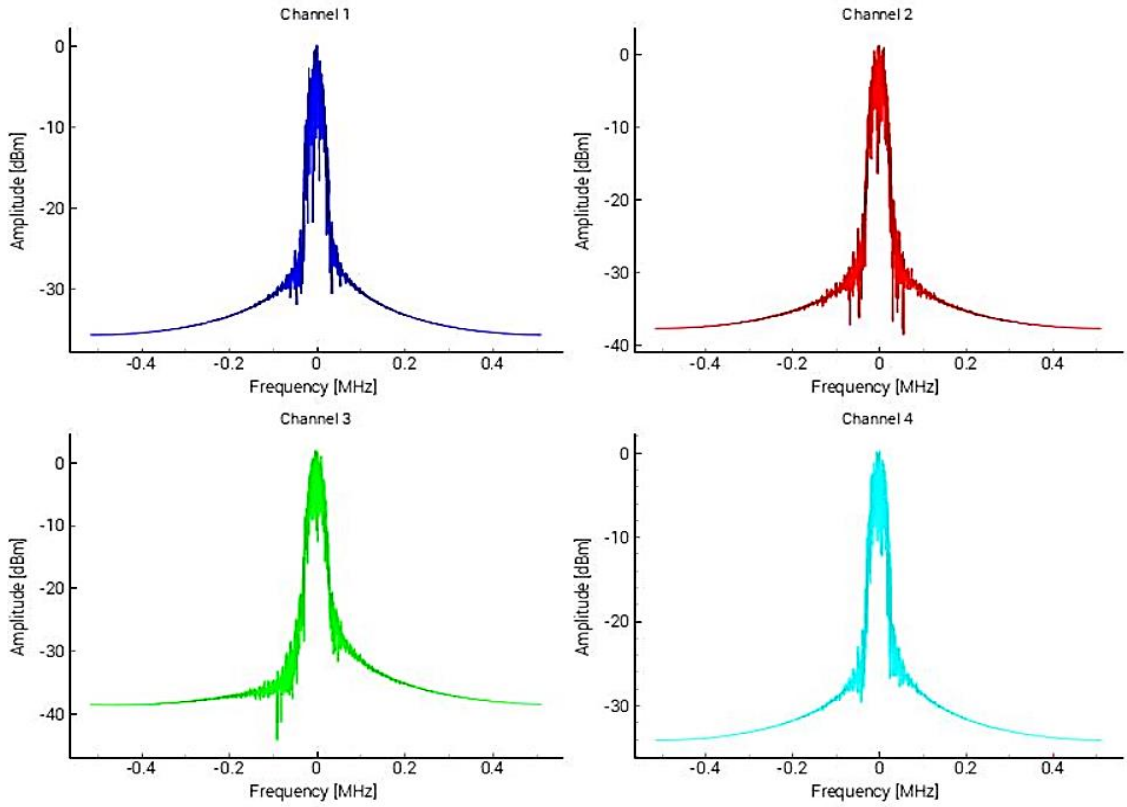


Figure (4.13) Spectrum of Four Channels of KerberosSDR.

4.3.1.1 Frist Configuration of RF Tracking Station

Frist configuration of the RFST system is used a ULA antenna array with a MUSIC algorithm. ULA elements are configured straight with d separation distance between adjacent elements. The spacing factor is 0.33, which is set in the software of KerberosSDR.

$$d = 0.33 * \lambda * 100 = 0.33 * 0.303 * 100 = 10cm$$

The KerberosSDR is configured on the vehicle with a ULA antenna array to collect the tracking data, as shown in Figure (4.14). Then, the vehicle starts moving for scanning the area. After driving for a few minutes, the tracking data file must be checked to ensure the bearing lines point in one direction. The tracking data file is shared with the operations center by internet.

Figure (4.14) The Practical Test of First Configuration.

A-One Tracking Point

In this test, the RF Tracking vehicle tracks the RES from a single position where RES is available in the detection area. RF tracking vehicle collects the tracking data with information regarding the received signal from the RES to the operations center GUI by internet. The RES with the actual distance 128m on the angle 45 degrees from the tracking station. The vehicle location is fixed as a reference station of the system. The positions of the vehicle and RES are shown in Table 4.25.

Table 4.25. One Point Tracking Data.

NO.	Test Items	Latitude	Longitude	Tracking Angle (deg)
1	RES	32.544668	44.435566	38.2
2	Station 1	32.543250	44.435465	

B-Two Tracking Points

The RF Tracking vehicle tracks the RES from two positions in this test. The positions of the vehicle and RES are shown in Table 4.26.

Table 4.26. Two Points Tracking Data.

NO.	Test Items	Latitude	Longitude	Error Distance (m)
1	RES	32.544149	44.436184	7.51
2	Station 1	32.543250	44.435465	
3	Station 2	32.545498	44.435465	

The operations center GUI of the RFST system represents the tracking data on the Google Map with the estimation of RES coordinates, as shown in Figure (4.15).

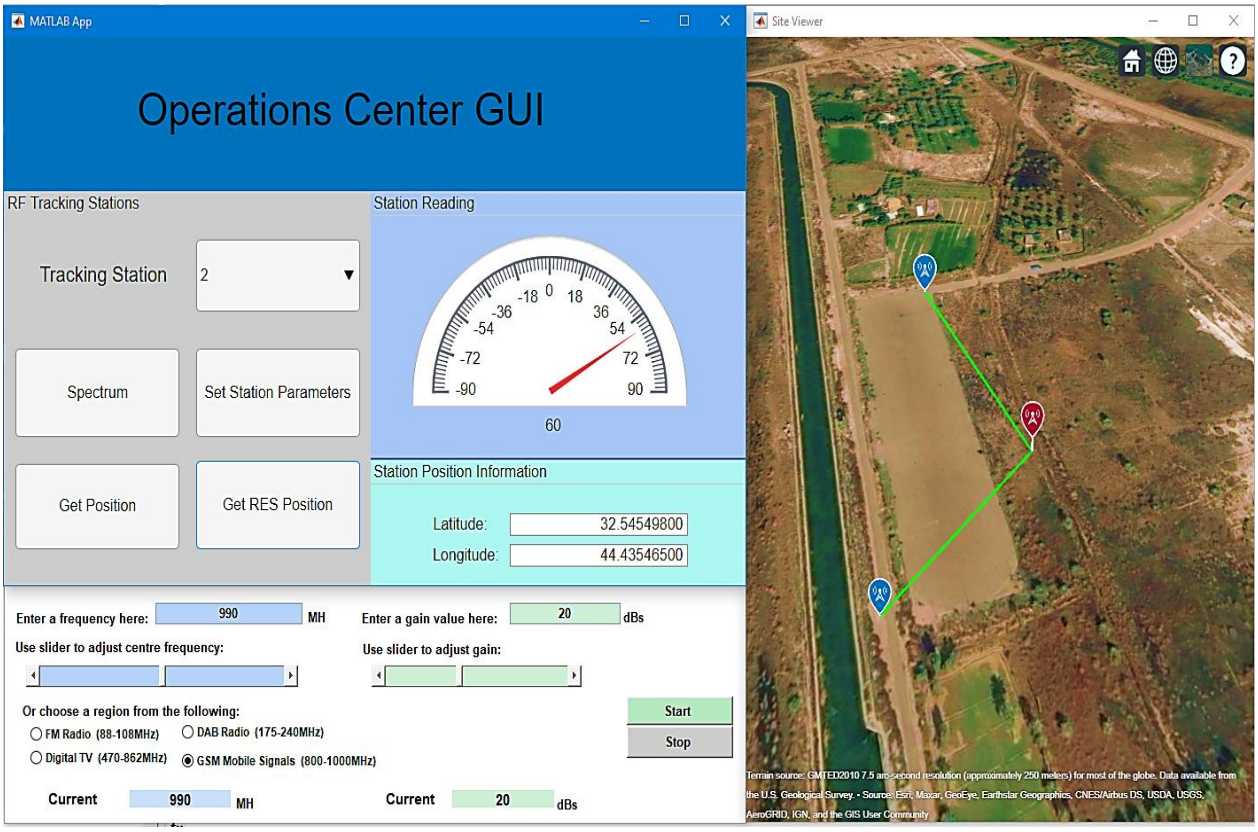


Figure (4.15) Operations Center GUI with Two Tracking Points.

C-Three Tracking Points

The RF Tracking vehicle tracks the RES from two positions in this test. The positions of the vehicle and RES are shown in Table 4.27.

Table 4.27. Three Points Tracking Data.

NO.	Test Items	Latitude	Longitude	Error Distance (m)
1	RES	32.544149	44.436184	5.01
2	Station 1	32.543250	44.435465	
3	Station 2	32.545498	44.435465	
4	Station 3	32.543250	44.438132	

The operations center GUI represents the tracking data collected from three positions with the estimation of RES coordinates, as shown in Figure (4.16).

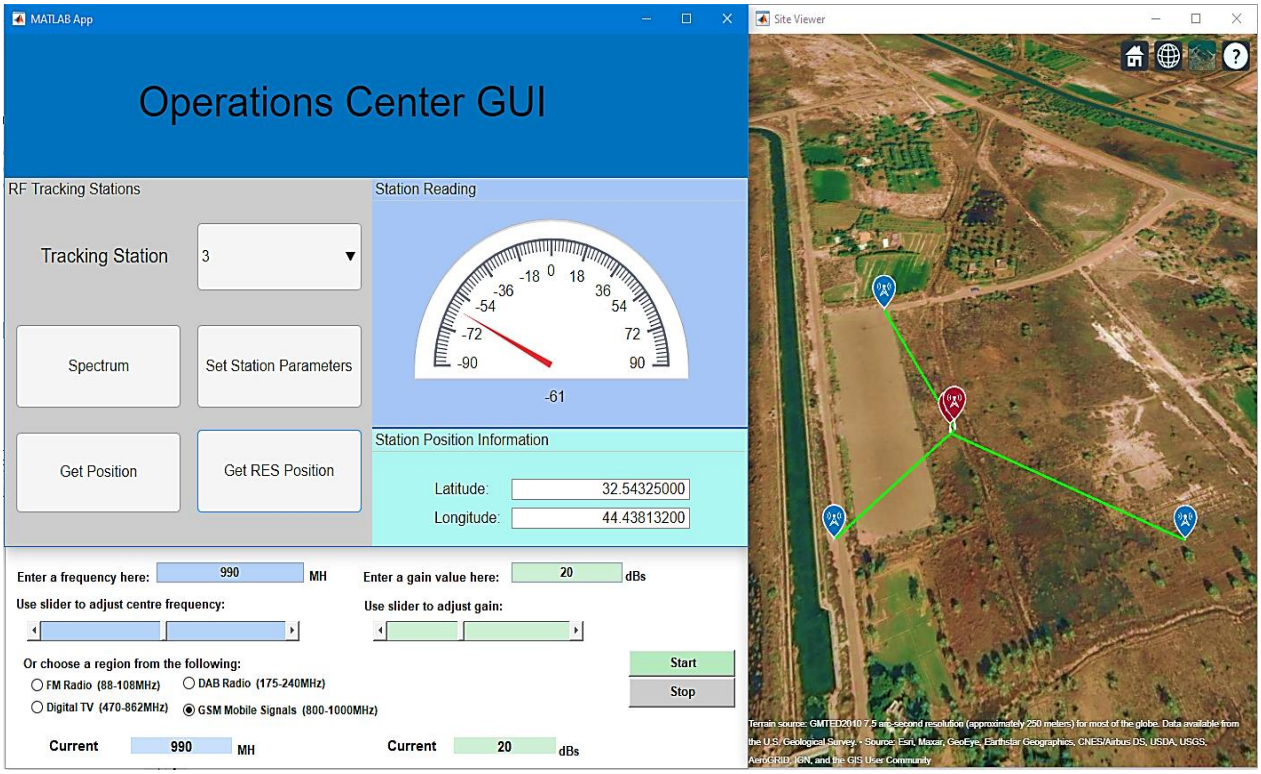


Figure (4.16) Operations Center GUI with Three Tracking Points.

C-Four Tracking Points

In this test, Four points are used to track the RES. The positions of the vehicle and RES are shown in Table 4.28.

Table 4.28. Four Points Tracking Data.

NO.	Test Items	Latitude	Longitude	Error Distance (m)
1	RES	32.544149	44.436184	4.23
2	Point 1	32.543250	44.435465	
3	Point 2	32.545498	44.435465	
4	Point 3	32.543250	44.438132	
5	Point 4	32.545498	44.438132	

Four tracking points with the estimation of RES coordinates can be represented by The operations center GUI, as shown in Figure (4.17).

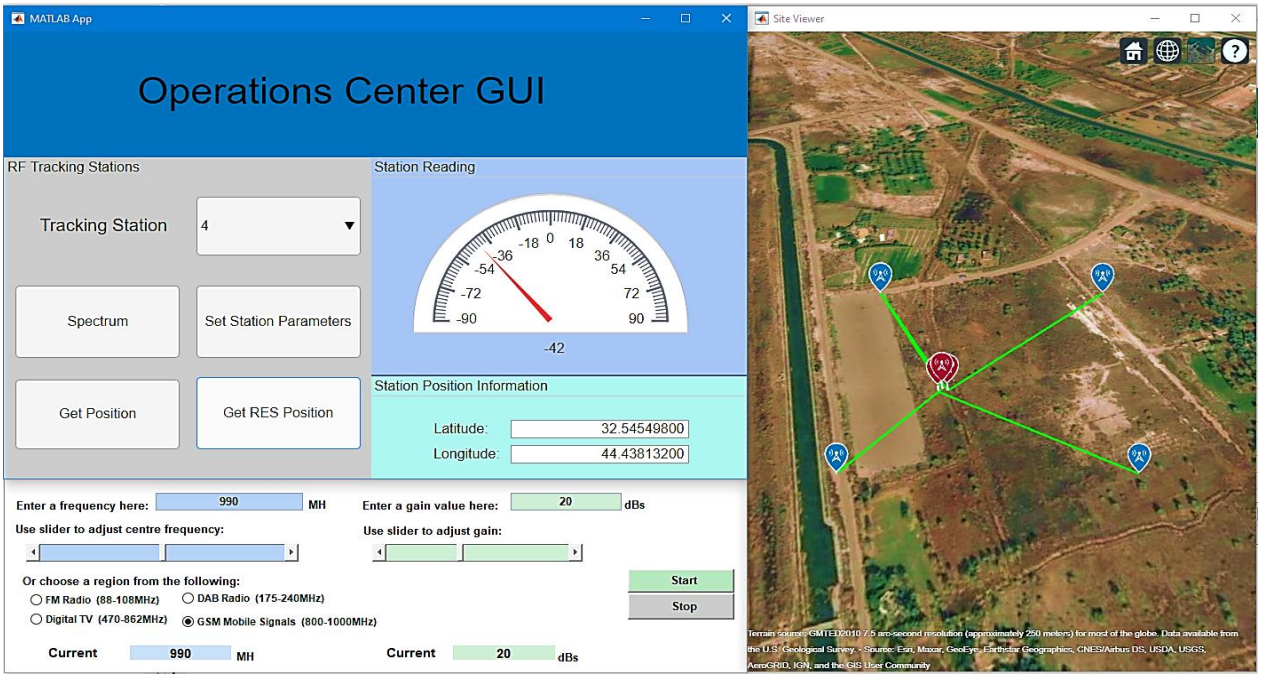


Figure (4.17) Operations Center GUI with Four Tracking Points.

4.3.1.2 Second Configuration of RF Tracking Station

The second configuration of the RFST system is used a UCA with a MUSIC algorithm. ULA elements are configured in a circle form with a d separation distance between adjacent elements in the circle's diameter, and r is the radius of UCA. The spacing factor is 0.33, which is set in the software of KerberosSDR.

$$d = 0.33 * \lambda * 100 = 0.33 * 0.303 * 100 = 10cm$$

$$R = d/1.4 = 7.1cm$$

The KerberosSDR is configured on the vehicle with a UCA antenna array to collect the tracking data, as shown in Figure (4.18). Then, the vehicle scans the area about RES.

Figure (4.18) The Practical Test of Second Configuration.

In this test, Four points are used to track the RES. The positions of the vehicle and RES are shown in Table 4.29.

Table 4.29. Four Points Tracking Data of the Second Configuration.

NO.	Test Items	Latitude	Longitude	Error Distance (m)
1	RES	32.544149	44.436184	4
2	Point 1	32.543250	44.435465	
3	Point 2	32.545498	44.435465	
4	Point 3	32.543250	44.438132	
5	Point 4	32.545498	44.438132	

Four tracking points with the estimation of RES coordinates can be represented by The operations center GUI, as shown in Figure (4.19).

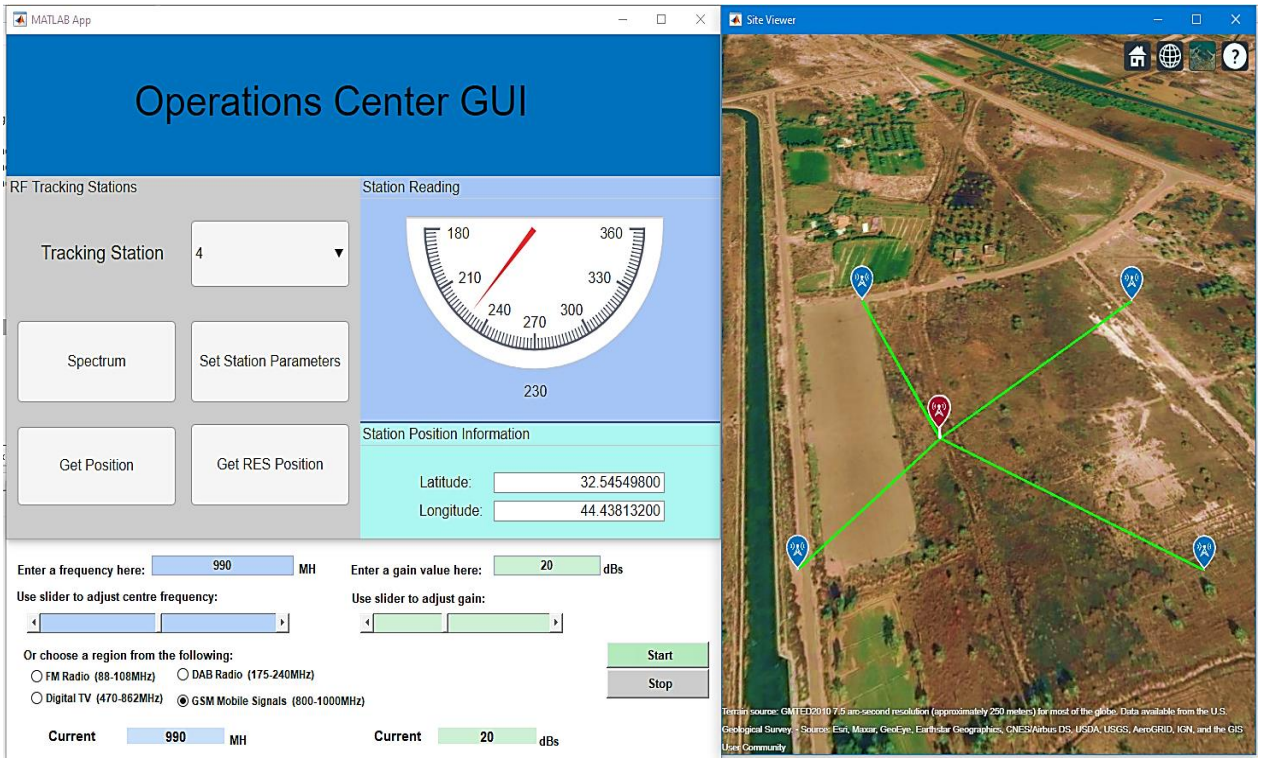


Figure (4.19) Operations Center GUI with UCA.

4.3.1.3 The Results of Spectrum Analyzer

The RTL-SDR receives RF signals that can be monitored using MATAB Simulink spectrum analyzers when the Matlab is running. For example, the Spectrum Analyzer FFT shows the signal spectrum at 990 MHz, as shown in Figure (4.20), in which the operator displays the range of RES.

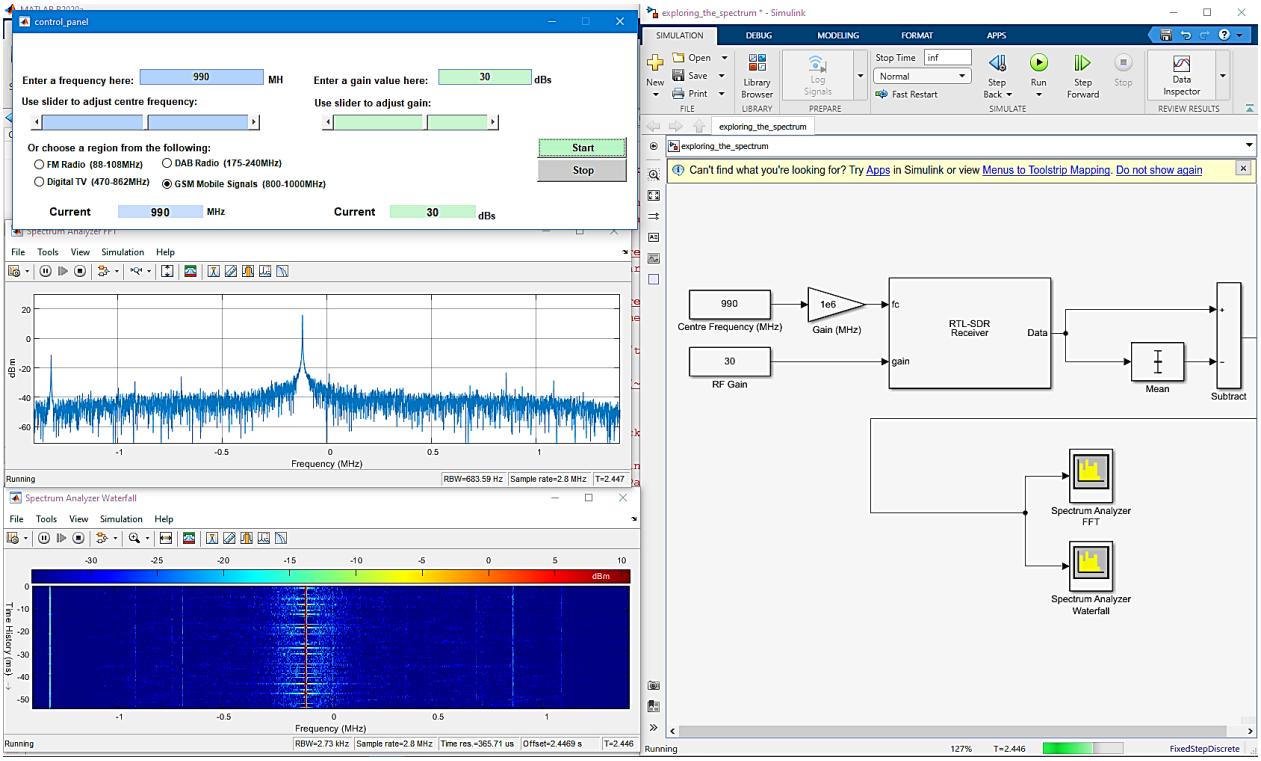


Figure (4.20) Software Spectrum Analyzer.

4.3.2 The Discussion of Practical Results

The practical results present the performance of the RFST system with MUSIC tracking algorithm and different tracking points for scanning RES based on GPS coordinates.

The system results are obtained by collecting tracking data from different points, from a single to four points. The best tracking resolution is determined by the minimum error distance, which are 4.23m and 5m when using the MUSIC tracking algorithm with UCA and ULA.

4.4 Comparison of Simulation and Practical Sides

Simulation and practical results of the proposed system are presented. Thus, the performance of the proposed system is proven in real terms. Furthermore, the system's work in both simulated and practical aspects is compared in the Table 4.30 through the obtained results.

Table 4.30. Comparison of Simulation and Practical Results.

NO.	Evaluation Factor	Error Distance (m)	
	Scanning Area	1000*1000m^2	250*250m^2
	Tracking Points	Simulation	Practical
1	Two Points	14.8	7.51
2	Three Points	12.65	5.01
3	Four Points	10.13	4.23

The MUSIC algorithm is used in both faces because KerbrosSDR provides spatial spectrum tracking algorithms. From Table (4.30), the comparison becomes very clear, and the results on both sides are very close, taking into account the difference in the tracking area. For example, the four-point case is taken, which has the highest tracking accuracy. There is a difference of 6m between the two sides, indicating that the proposed system has been built accurately in both cases.

Furthermore, Table 4.31 shows the comparison between our work and past works. Simulation, practical algorithms, and accuracy study were considered the most important aspects of radio signal tracking analysis. Therefore, it can be compared among works in the same field. Our work has studied most of the specific aspects. Thus, our work has overcome the most recent works mentioned in this thesis.

Table 4.31. Comparison of Works.

No	Works	Simulation Side	Practical Side	Algorithms Study	Accuracy Study
1	**Our work**	Many simulations cases for tracking operation	Used two SDR for spectrum analyzer and estimate DOA	Tested Seven algorithms for tracking	Study tracking accuracy with error 2m
2	**Piotr 2021, [30]**	Did not study	Used SDR for estimate DOA	Tested one algorithm for tracking	Did not study
3	**Zuokun 2020, [29]**	Did not study	Used SDR for estimate DOA	Only MUSIC algorithm	Did not study
4	**Btissam B. 2019, [28]**	Did not study	Did not deal with the practical side	Tested Seven algorithms	Did not study
5	**Kristiyana 2017, [25]**	A single case of three RDF stations are used for tracking	Did not deal with the practical side	Did not deal with algorithms	Study the tracking operation with error 4.6m

Chapter Five

Conclusions and Future Work

5.1 Conclusions

The following are the main points of the thesis conclusions:

- The DOA techniques plays a significant role in constructing the RFST system.
- The operations center has an instantaneous synchronization with the RF Tracking Stations because of the efficient internet sharing.
- In the tracking algorithms test, the higher-resolution estimation is obtained using 8 or 16 elements in the array.
- The tracking algorithms test shows that the accuracy of DOA estimation is proportional to the number of snapshots.
- Root-WSF and Root-MUSIC methods proved the strongest resolution estimation in the tracking algorithms test.
- ROOT-WSF algorithm proves the super-resolution for tracking the DOA under hard conditions.
- ULA and UCA provide better performance than URA in the array configurations test, which needs more elements for correct estimation.
- MUSIC estimation shows suitable performance in the simulation of array geometries types.
- In the simulation of the RFST system, the minimum error distance is 2m when using the WSF tracking with the intersection of four RF Tracking Stations.

- In The practical results, the minimum error distances are 4.23m and 4m when using the MUSIC tracking algorithm with ULA and UCA.

- The comparison of simulation and practical results show an error distance around 6m in the practical tracking results with the intersection of four tracking points, indicating that the proposed system has been built accurately on both sides.

- Commercially available systems is costly, making them cost-prohibitive for most individuals and organizations. However, advancements in Software Defined Radios (SDRs) and simplistic software programs standard in hobbyist and professional systems have driven the cost down and increased overall availability.

5.2 Future works

Some of the following points can be used to improve the current system. :

- By using the Artificial Neural Network (ANN) models to the azimuth and elevation arrival angles estimation of the signal source. The ANN model can be used to compute the signal source's angular positions, which yields more precise results.
- Develops tracking systems to eight SDR receives by using two synchronized SDR dongles.
- Improving antenna arrays configuration and extracting the best performance of tracking algorithms with the best structure of antenna arrays.
- By improving technologies of AI and SDR. Cognitive radio applications are on the horizon. The integration of these two fields of study will create new capabilities for communication infrastructure.

References

[1] Rehmani, Mubashir Husain, and Riadh Dhaou, eds. *Cognitive radio, mobile communications and wireless networks*. Springer International Publishing, 2019.

[2] Hosmer, Chet, Hosmer, and McDermott. *Defending IoT Infrastructures with the Raspberry Pi*. Apress, 2018.

[3] R. Schmidt and R. Franks, "Multiple source DF signal processing: An experimental system," in IEEE Transactions on Antennas and Propagation, vol. 34, no. 3, pp. 281-290, March 1986, doi: 10.1109/TAP.1986.1143815.

[4] Y. T. Chan and K. C. Ho, "A simple and efficient estimator for hyperbolic location," *IEEE Transactions on Signal Processing*, vol. 42, no. 8, pp. 1905-1915, Aug. 1994, doi: 10.1109/78.301830.

[5] Muhamed, Rias. *Direction of arrival estimation using antenna arrays*. Diss. Virginia Tech, 1996.

[6] Hu, Zhong. *Evaluation of joint aoa and doa estimation algorithms using the antenna array systems*. Diss. Virginia Tech, 1999.

[7] S. Shahbazpanahi, S. Valaee and M. H. Bastani, "Distributed source localization using ESPRIT algorithm," *IEEE Transactions on Signal Processing*, vol. 49, no. 10, pp. 2169-2178, Oct. 2001, doi: 10.1109/78.950773.

[8] L. A. Cirillo, A. M. Zoubir and A. B. Gershman, "Direction-of-arrival estimation for uncorrelated FM signals," *Sensor Array and Multichannel Signal Processing Workshop Proceedings*, 2002, pp. 318-322, doi: 10.1109/SAM.2002.1191052.

[9] Qin, Hongfeng, Jianguo Huang, and Qunfei Zhang. "A novel joint estimator of direction-of-arrival and time-delay for multiple source localization." *International Conference on Neural Networks and Signal Processing*, 2003. Proceedings of the 2003. Vol. 2. IEEE, 2003.

[10] Do-Hong, Tuan, Franz Demmel, and Peter Russer. "Wideband direction-of-arrival estimation using frequency-domain frequency-invariant beamformers: an analysis of performance." *IEEE microwave and wireless components letters* 14.8 (2004): 383-385.

[11] Huang, Lei, Shunjun Wu, and Linrang Zhang. "A novel MUSIC algorithm for direction-of-arrival estimation without the estimate of covariance matrix and its eigendecomposition." *2005 IEEE 61st Vehicular Technology Conference.* Vol. 1. IEEE, 2005.

[12] V. Y. Vu and A. B. Delai, "Digital Solution for inter-vehicle localisation system by means of Direction-Of-Arrival," *2006 International Symposium on Intelligent Signal Processing and Communications*, 2006, pp. 875-878, doi: 10.1109/ISPACS.2006.364781.

[13] Al Jabr, Kareem A. *Modified UCA-ESPRIT and modified UCA-ROOT-MUSIC for estimating DOA of coherent signals using one snapshot*. Diss. Wichita State University, 2007.

[14] Goossens, Roald, Hendrik Rogier, and Steven Werbrouck. "UCA Root-MUSIC with sparse uniform circular arrays." *IEEE transactions on signal processing* 56.8 (2008): 4095-4099.

[15] Reed, Jesse. *Approaches to multiple-source localization and signal classification*. Diss. Virginia Tech, 2009.

[16] W Tidd, Will, et al. "A RF source localization and tracking system." *2010-MILCOM 2010 MILITARY COMMUNICATIONS CONFERENCE*. IEEE, 2010.

[17] Kulaib, Ahmed Rashed, et al. "Performance evaluation of linear and circular arrays in wireless sensor network localization." *2011 18th IEEE International Conference on Electronics, Circuits, and Systems*. IEEE, 2011.

[18] Ling, Wang, and Shuzhi Sam Ge. "Localization method of multiple targets by crossing direction finding." *2011 IEEE 5th International Conference on Cybernetics and Intelligent Systems (CIS)*. IEEE, 2011.

[19] Malanowski, Mateusz. "Algorithm for target tracking using passive radar." *International Journal of Electronics and Telecommunications* 58 (2012): 345-350.

[20] Baig, Nauman Anwar, and Mohammad Bilal Malik. "Comparison of direction of arrival (DOA) estimation techniques for closely spaced targets." *International journal of future computer and communication* 2.6 (2013): 654.

[21] Susaritha, U. S., and Ashita Priya Thomas. "A Comparative Study of Different DOA Estimation Schemes and Adaptive Beam Forming Techniques for Target Detection and Tracking." *International Journal of Computer Applications* 975 (2015): 8887.

[22] PĂUN, Mirel, Răzvan TAMAS, and Ion Marghescu. "A Software-Defined Radio Approach for Direction Finding." *UPB Sci. Bull.*, Series C 77.4 (2015).

[23] Balasubramanian, Ramaswamy Karthikeyan. *Development of Novel Approaches for High Resolution Direction of Arrival Estimation Techniques*. Diss. Coventry University, 2016.

[24] Abdessamad, Wissam, et al. "An SDR platform using direction finding and statistical analysis for the detection of interferers." *2016 8th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. IEEE, 2016.

[25] Kristiyana, Samuel, et al. "The Radio Frequency Source Position Finder Based on The Triangle-Centroid-Algorithm." *IJITEE (International Journal of Information Technology and Electrical Engineering)* 1.1 (2017): 13-18.

[26] Boustani, Btissam, et al. "Performance analysis of direction of arrival estimation under hard condition." *2018 4th International Conference on Optimization and Applications (ICOA)*. IEEE, 2018.

[27] Miyamoto, Kazuki, Nobuyoshi Kikuma, and Kunio Sakakibara. "Performance Evaluation of DOA Estimation Using Compressed Sensing with Fast Fourier

Transform." *2019 International Workshop on Electromagnetics: Applications and Student Innovation Competition (iWEM)*. IEEE, 2019.

[28] Boustani, Btissam, et al. "Performance analysis of direction of arrival algorithms for smart antenna." *International Journal of Electrical and Computer Engineering* 9.6 (2019): 4873.

[29] Li, Zuokun, et al. "Application Research on DOA Estimation Based on Software-Defined Radio Receiver." *Journal of Physics: Conference Series*. Vol. 1617. No. 1. IOP Publishing, 2020.

[30] Araszkiewicz, Piotr, Norbert Niderla, and Kacper Murat. "Low-budget passive locating system for IoT applications: analysis and implementation." *Photonics Applications in Astronomy, Communications, Industry, and High Energy Physics Experiments* 2020. Vol. 11581. International Society for Optics and Photonics, 2020.

[31] Harter, Nathan M. *Development of a single-channel direction finding algorithm*. Diss. Virginia Tech, 2007.

[32] Rembovsky, Anatoly, et al. "*Radio monitoring." Problems, methods and equipment. Lecture notes in electrical engineering. Springer* (2009).

[33] Moell, Joseph D., and Thomas N. Curlee. *Transmitter Hunting: Radio Direction Finding Simplified*. Vol. 2701. McGraw Hill Professional, 1987.

[34] H. G. Schantz, "On the origins of RF-based location," *2011 IEEE Topical Conference on Wireless Sensors and Sensor Networks*, 2011, pp. 21-24, doi: 10.1109/WISNET.2011.5725029.

[35] Sánchez, Tibisay, et al. "Radio direction finding system for spectrum management activities in developing countries." *2016 IEEE International Symposium on Antennas and Propagation (APSURSI)*. IEEE, 2016.

[36] Schell, Stephan V., and William A. Gardner. "18 High-resolution direction finding." *Handbook of statistics* 10 (1993): 755-817.

[37] O'Donoughue, Nicholas. *Emitter Detection and Geolocation for Electronic Warfare*. 1st ed, Artech House, 2019.

[38] Teutsch, Heinz. *Modal array signal processing: principles and applications of acoustic wavefield decomposition*. Vol. 348. Springer, 2007.

[39] Gross, Frank B. *Smart Antennas with MATLAB®*. 1st ed, McGraw-Hill Education, 2015.

[40] Luo, Qi, et al. *Low-cost smart antennas*. John Wiley & Sons, 2019.

[41] Vijay, Mane Sunita, and U. L. Bombale. "An overview of smart antenna and a survey on direction of arrival estimation algorithms for smart antenna." *Journal of Electronics and Communication Engineering (IOSR-JECE)*. 2014.

[42] Xiong, Hao. "Antenna array geometries and algorithms for direction of arrival estimation." *MRes thesis, University of Nottingham* (2013).

[43] Manikas, A., A. Alexiou, and H. R. Karimi. "Comparison of the ultimate direction-finding capabilities of a number of planar array geometries." *IEE Proceedings-Radar, Sonar and Navigation* 144.6 (1997): 321-329.

[44] Xia, Maohui, and Ruiyan Du. "New method of effective array for 2-D direction-of-arrival estimation." *International Journal of Innovative Computing, Information and Control* 2.6 (2006): 1391-1397.

[45] Wyglinski, Alexander M., et al. *Software-defined radio for engineers.* Artech House, 2018.

[46] Rouphael, Tony J. *RF and digital signal processing for software-defined radio: a multi-standard multi-mode approach*. Newnes, 2009.

[47] Chávez-Santiago, Raúl, et al. "Applications of software-defined radio (SDR) technology in hospital environments." *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2013.

[48] Danymol, R., T. Ajitha, and R. Gandhiraj. "Real-time communication system design using RTL-SDR and Raspberry Pi." *2013 International Conference on Advanced Computing and Communication Systems*. IEEE, 2013.

[49] Foutz, Jeffrey, Andreas Spanias, and Mahesh K. Banavar. "Narrowband direction of arrival estimation for antenna arrays." *Synthesis Lectures on Antennas* 3.1 (2008): 1-76.

[50] Van Trees, Harry L. *Optimum array processing: Part IV of detection, estimation, and modulation theory*. John Wiley & Sons, 2004.

[51] Chen, Zhizhang, Gopal Gokeda, and Yiqiang Yu. *Introduction to Direction-of-arrival Estimation*. Artech House, 2010.

[52] Mishra, Amit Kumar, and Ryno Strauss Verster. *Compressive sensing based algorithms for electronic defence*. Springer International Publishing, 2017.

[53] Tuncer, T. Engin, and Benjamin Friedlander, eds. *Classical and modern direction-of-arrival estimation*. Academic Press, 2009.

[54] Gentilho, Edno, Paulo Rogerio Scalassara, and Taufik Abrão. "Direction-of-Arrival Estimation Methods: A Performance-Complexity Tradeoff Perspective." *Journal of Signal Processing Systems* 92.2 (2020): 239-256.

[55] Khmou, Youssef, Said Safi, and Miloud Frikel. "Comparative study between several direction of arrival estimation methods." Journal of Telecommunications and Information Technology (2014).

[56] Paulraj, R. Roy,"Kailath, Estimation of signal parameters via rotational invariance techniques – ESPRIT," *In Proc. Nineteenth Asdomar Conf. on Circuits, Systems and Comp., Asilomar, CA.*, 1985.

[57] Lavate, T. B., Kokate, V. K., & Sapkal, A. M., "Performance Analysis of Music and esprit DOA estimation algorithms for adaptive array smart antenna in Mobile Communication," *2010 Second International Conference on Computer and Network Technology*. https://doi.org/10.1109/iccnt.2010.45, 2010.

[58] Osman, Lotfi, Imen Sfar, and Ali Gharsallah. "Comparative study of high-resolution direction-of-arrival estimation algorithms for array antenna system." *International Journal of Research and Reviews in Wireless Communications (IJRRWC)* Vol 2 (2012).

[59] Viberg, Mats, Bjorn Ottersten, and Thomas Kailath. "Detection and estimation in sensor arrays using weighted subspace fitting." *IEEE transactions on Signal Processing* 39.11 (1991): 2436-2449.

[60] Rembovsky, Anatoly M., et al. *Radio monitoring: automated systems and their components*. Springer International Publishing, 2018.

[61] MATLAB, "The MathWorks, Source Localization Using Generalized Cross Correlation," Inc, Natick, United States, 2020.

Appendix A

DOA Algorithms (Matlab Code)

A-1 Beamscan

```
c = 900e6;
lambda = physconst('LightSpeed')/fc;
ula = phased.ULA('NumElements',8,'ElementSpacing',lambda/2);
ula.Element.FrequencyRange = [8e8 1.2e9];
l=length(angs);
c = physconst('LightSpeed');
pos = getElementPosition(ula)/lambda;
Nsamp = 2000;
nPower = 0.01;
fs=8e3;
rs = rng(2007);
signal = sensorsig(pos,Nsamp,angs,nPower);
properties (Nontunable, Dependent = true)
   NumPhaseShifterBits = 0
end
properties (Nontunable, PositiveInteger)
   NumSignals = 1;
end
methods
   function obj = BeamscanEstimator(varargin)
      obj = obj@phased.internal.AbstractULASpectralDOA(varargin{:});
   end
end
methods
   function set.NumPhaseShifterBits(obj,value)
      validateattributes( value, { 'double','single' }, ...
        { 'scalar', 'integer', 'nonnegative', 'finite' }, '', 'NumPhaseShifterBits');
      obj.pNumPhaseShifterBits = value;
```

```
        end
        function val = get.NumPhaseShifterBits(obj)
            val = obj.pNumPhaseShifterBits;
        end
    end
    methods (Access = protected)
        function privDOASpectrum(obj,Cx)
            sv = obj.pSteeringVector;
            obj.pPattern = real(sum((sv'*Cx).*sv.',2));
        end
        function titlestr = getSpectrumPlotTitle(obj)
coder.extrinsic('phased.BeamscanEstimator.getSpectrumPlotTitleExtrinsic');
            titlestr = obj.getSpectrumPlotTitleExtrinsic();
        end
        function str = getIconImpl(obj)
            str = sprintf('ULA Beamscan\nSpectrum ');
        end
    end
    methods (Static, Hidden)
        function titlestr = getSpectrumPlotTitleExtrinsic()
            titlestr = sprintf('%s %s',...
                getString(message('phased:doa:Beamscan')),...
                getString(message('phased:doa:SpatialSpectrum')));
        end
    end
    methods (Static,Hidden,Access=protected)
        function groups = getPropertyGroupsImpl
            groups = getPropertyGroupsImpl@phased.internal.AbstractULASpectralDOA;
            props = {...
              'NumPhaseShifterBits'};
            iNew = find(strcmp(groups(1).PropertyList,'ForwardBackwardAveraging'));
```

```
            groups(1).PropertyList = [groups(1).PropertyList(1:iNew-1) props
groups(1).PropertyList(iNew:end)];
            groups(1).DependOnPrivatePropertyList =  [groups(1).DependOnPrivatePropertyList
{'NumPhaseShifterBits'}];
        end
        function header = getHeaderImpl
            header = matlab.system.display.Header(...
                'Title',getString(message('phased:library:block:BeamscanEstimatorTitle')),...
                'Text',getString(message('phased:library:block:BeamscanEstimatorDesc')));
        end
    end
    end

    spatialspectrum =
phased.BeamscanEstimator('SensorArray',ula,'OperatingFrequency',fc,'ScanAngles',-90:90);
    spatialspectrum.DOAOutputPort = true;
    spatialspectrum.NumSignals = l;
    [~,e1] = spatialspectrum(signal);
```

A-2 MVDR

```
    roperties (Nontunable, Dependent=true)
        %NumPhaseShifterBits    Number of bits in phase shifters
        NumPhaseShifterBits = 0
    end
    properties (Nontunable, PositiveInteger)
        NumSignals = 1;
    end
    methods
        function obj = MVDREstimator(varargin)
            obj = obj@phased.internal.AbstractULASpectralDOA(varargin{:});
        end
    end
```

```
methods
   function set.NumPhaseShifterBits(obj,value)
      validateattributes( value, { 'double','single' }, ...
        { 'scalar', 'integer', 'nonnegative', 'finite' }, '', 'NumPhaseShifterBits');
      obj.pNumPhaseShifterBits = value;
   end
   function val = get.NumPhaseShifterBits(obj)
      val = obj.pNumPhaseShifterBits;
   end
end
methods (Access = protected)
   function privDOASpectrum(obj,Cx)
      sv = obj.pSteeringVector;
      obj.pPattern = 1./real(sum(sv'.*(Cx\sv).',2));  % equation 9.13 [1]
   end
   function titlestr = getSpectrumPlotTitle(obj)
      coder.extrinsic('phased.MVDREstimator.getSpectrumPlotTitleExtrinsic');
      titlestr = obj.getSpectrumPlotTitleExtrinsic();
   end
   function str = getIconImpl(obj) %#ok<MANU>
      str = sprintf('ULA MVDR\nSpectrum');
   end
end
methods (Static, Hidden)
   function titlestr = getSpectrumPlotTitleExtrinsic()
      titlestr = sprintf('%s %s',...
         getString(message('phased:doa:MVDR')),...
         getString(message('phased:doa:SpatialSpectrum')));
   end
end
methods (Static,Hidden,Access=protected)
   function groups = getPropertyGroupsImpl
```

```
            groups = getPropertyGroupsImpl@phased.internal.AbstractULASpectralDOA;
            props = {...
              'NumPhaseShifterBits'};
            iNew = find(strcmp(groups(1).PropertyList,'ForwardBackwardAveraging'));
            groups(1).PropertyList = [groups(1).PropertyList(1:iNew-1) props
groups(1).PropertyList(iNew:end)];
            groups(1).DependOnPrivatePropertyList =  [groups(1).DependOnPrivatePropertyList
{'NumPhaseShifterBits'}];
        end
        function header = getHeaderImpl
            header = matlab.system.display.Header(...
                'Title',getString(message('phased:library:block:MVDREstimatorTitle')),...
                'Text',getString(message('phased:library:block:MVDREstimatorDesc')));
        end
    end
end
    mvdrspatialspect = phased.MVDREstimator('SensorArray',ula,...
        'OperatingFrequency',fc,'ScanAngles',-90:90,...
        'DOAOutputPort',true,'NumSignals',l);
[~,e2] = mvdrspatialspect(signal);
```

A-3 MUSIC

```
        % Estimate the covariance matrix Cx and compute eigenvalues and
        % eigenvectors of Cx.
        % Estimate the covariance matrix
        Cx = step(obj.cCovEstimator,X);

        % Compute eigenvectors of the covariance matrix
        [eigenvals, eigenvects] = privEig(obj,Cx);
        eigenvals = cast(eigenvals,classtouse);
        eigenvects = cast(eigenvects,classtouse);
```

```
    % Obtain signal subspace dimension
    Nsig = getNumSignals(obj,eigenvals,obj.pNumSnapshots,fb);

    if Nsig==0
        if isempty(coder.target)
            warning(message('phased:phased:doa:ZeroSourceNumber'));
        end
        return;
    end

    % Form MUSIC denominator matrix from noise subspace
    % eigenvectors
    noise_eigenvects = eigenvects(:, Nsig + 1:end);

    % Compute spatial spectrum. Add a small positive constant to prevent
    % division by zero.
    sv = obj.pSteeringVector;
    D = sum(abs((sv'*noise_eigenvects)).^2,2)+eps(1); % 9.44 in [1]
    obj.pPattern = 1./D;

end

function titlestr = getSpectrumPlotTitle(obj) % #ok<MANU>
    coder.extrinsic('phased.MUSICEstimator.getSpectrumPlotTitleExtrinsic');
    titlestr = obj.getSpectrumPlotTitleExtrinsic();
end

function D = getNumSignals(obj,eigenvals,K,fb)
    % Obtain signal subspace dimension
    if strcmp(obj.NumSignalsSource,'Auto')
        if strcmp(obj.NumSignalsMethod,'AIC')
```

```
            D = phased.internal.aictest(eigenvals,K,fb);
          else
            D = phased.internal.mdltest(eigenvals,K,fb);
          end
        else
          D = obj.NumSignals;
        end
      end

      function validatePropertiesImpl(obj)
        validatePropertiesImpl@phased.internal.AbstractULASpectralDOA(obj);
        if strcmp(obj.NumSignalsSource,'Property')
          % Verify that requested number of signals is not greater than
          % the number allowed based on the array size and smoothing.
          maxNumSig = getMaxNumSignal(obj);
          cond = obj.NumSignals > maxNumSig;
          if cond

coder.internal.errorIf(cond,'phased:phased:internal:AbstractULASpectralDOA:InvalidNumSourc
es',maxNumSig);
          end
        end
      end

      function maxNumSig = getMaxNumSignal(obj)
        % NumSignals should be no greater than possible signal subspace dim - 1
        maxNumSig = getEffectiveChannel(obj) - 1;
      end

      function [eigenvals, eigenvects] = privEig(obj,Sx) %#ok<INUSL> % #ok<MANU>
        [eigenvects, eigenvalsC] = eig(Sx);
        eigenvals = real(eigenvalsC);
```

```
            [eigenvals,indx] = sort(diag(eigenvals),'descend');
            eigenvects= eigenvects(:,indx);
            eigenvals(eigenvals<0) = 0;
        end

        function flag = isInactivePropertyImpl(obj, prop)
            flag = false;
            SourceAuto = strcmp(obj.NumSignalsSource,'Auto');
            if SourceAuto && strcmp(prop,'NumSignals')
                flag = true;
            end
            if ~SourceAuto && strcmp(prop,'NumSignalsMethod')
                flag = true;
            end
        end
    end

end
```

A-4 ESPRITE

```
        function doasOut = stepImpl(obj, X)
        % Element-space ESPRIT

            classtouse = class(X);
            ds = obj.pDS;
            fb = obj.ForwardBackwardAveraging;

            Cx = step(obj.cCovEstimator,X);
            [eigenvals, eigenvects] = privEig(obj,Cx);
            eigenvals = cast(eigenvals,classtouse);%g1812952
            eigenvects = cast(eigenvects,classtouse);
```

```
        % Signal subspace dimension
        D = getNumSignals(obj,eigenvals,obj.pNumSnapshots,fb);
        if D==0
           if isempty(coder.target)
              warning(message('phased:phased:doa:ZeroSourceNumber'));
           end
           doasOut = zeros(1,0,classtouse);
           return;
        end

        % check eigenvalues against source dimension. ESPRIT does not work
        % when there are less then D non-zero eigenvalues.
        D_act = sum(eigenvals>eps(max(abs(eigenvals))));
        cond = D_act < D;
        if cond
           coder.internal.errorIf(cond,'phased:phased:ESPRITEstimator:NotEnoughRank', D,
D_act);
        end

        % Selection matrices
        [Js1, Js2] = local_selection_matrices(cast(obj.pEffChannels,classtouse)+1,...
           cast(obj.RowWeighting,classtouse),ds,fb);

        % Core ESPRIT algorithm
        psieig = privCoreESPRIT(obj,eigenvects,D,Js1,Js2,obj.Method);

        % Extract angle
        doas = local_extract_angles(psieig,ds,fb);

        % Convert angles
        doasOut = privConvertAngles(obj,doas);
```

```
end
methods (Static,Hidden,Access=protected)
function [Js1, Js2] = local_selection_matrices(M,ms,ds,fb)
   Ns = M-ds;
   w = min(ceil(ms),Ns-ceil(ms)+1;
   weights = diag(sqrt([1:w-1 w*ones(1,Ns-2*(w-1)) w-1:-1:1]));
   O = zeros(Ns,ds);
   if fb
      % Constants
      Qn = phased.internal.unitarymat(M);
      Qns = phased.internal.unitarymat(Ns);
      Js2 = [O weights];
      K = Qns'*Js2*Qn;
      Js1 = real(K);     % Eq 9.147 in [1]
      Js2 = imag(K);     % Eq 9.148 in [1]
   else
      % Selection Matrices
      Js1 = [weights O]; % Eq 9.134 in [1]
      Js2 = [O weights]; % Eq 9.135 in [1]
   end
end
function doas = local_extract_angles(psieig,ds,fb)
   if fb
      if isreal(psieig)
         psieigReal = psieig;
      else
         %reliability test for unitary ESPRIT
         %failed, return real part only
         psieigReal = real(psieig);
      end
      doas = 2*atan(1/ds*psieigReal);
   else
```

```
        doas = 1/ds*angle(psieig);
    end
end
```

A-5 Beamspace ESPRIT

```
    % Estimate the forward-backward averaged spatial covariance matrix
    % in element space.
    classtouse=class(X);
    Sx = step(obj.cCovEstimator,X);
    Nbs = obj.pEffChannels;

    M = size(Sx,1); % effective array element number

    uc = round(sin(obj.BeamFanCenter*pi/180)*M/2);

    % Compute the real beamspace covariance matrix
    [W, beamfan] = privDFTMatrix(M,Nbs,uc);
    Sxbs = real(W'*Sx*W);
    [eigenvals, eigenvects] = privEig(obj,Sxbs);
    eigenvals = cast(eigenvals,classtouse);% g1812952
    eigenvects = cast(eigenvects,classtouse);

    % Obtain signal subspace dimension
    D = getNumSignals(obj,eigenvals,obj.pNumSnapshots,true);
    if D==0
       if isempty(coder.target)
          warning(message('phased:phased:doa:ZeroSourceNumber'));
       end
       doasOut = zeros(1,0,class(X));
       return;
    end
```

```
            % Selection matrices
            [Gamma1, Gamma2] =
local_selection_matrices(M,cast(Nbs,classtouse),cast(beamfan,classtouse));

            % Core ESPRIT algorithm
            psieig = privCoreESPRIT(obj,eigenvects,D,Gamma1,Gamma2,obj.Method);
            psieigReal = real(psieig);
            % Convert angles
            doasOut = privConvertAngles(obj,2*atan(psieigReal));

        end

    function [W, beamfan] = privDFTMatrix(M,Nbs,uc)
    %privDFTMatrix DFT (a.k.a. Butler) Matrix
    %   M=N-L+1 number of effective sensors
    %   Nbs number of beams
    %   uc center beam (for reduced dimension beamspace i.e. Nbs<M)

        % Define Nbs beam indices
        m = floor(-Nbs/2)+1:floor(Nbs/2); % Eq 3.322 & 3.323 in [1]
        beamfan = m+uc; % uc*2/M increments

        % Wrap if beamfan>floor(M/2)
        idx = beamfan>floor(M/2);
        if any(idx),
            beamfan(idx) = beamfan(idx)-M;
        end

        % Wrap if beamfan<floor(-M/2)+1
        idx = beamfan<floor(-M/2)+1;
        if any(idx),
            beamfan(idx) = beamfan(idx)+M;
```

```
    end

    n = (-(M-1)/2:(M-1)/2).';
    W = 1/M*exp(1j*2*pi/M*n*beamfan);

  end

  function [Gamma1, Gamma2] = local_selection_matrices(M,Nbs,beamfan)
  % Generate selection matrix

    aux = cos(pi/M*beamfan);
    lastCol1 = [zeros(Nbs-2,1);aux(Nbs)];
    Gamma1temp = diag(aux(1:Nbs-1))+diag(aux(2:Nbs-1),1); % Eq 9.313
    Gamma1 = [Gamma1temp lastCol1];

    aux = sin(pi/M*beamfan);
    lastCol2 = [zeros(Nbs-2,1);aux(Nbs)];
    Gamma2temp = diag(aux(1:Nbs-1))+diag(aux(2:Nbs-1),1); % Eq 9.314
Gamma2 = [Gamma2temp lastCol2];
```

A-6 ROOT-MUSIC

```
  properties (Dependent, Nontunable)
    SpatialSmoothing
  end

  properties (Nontunable, Logical)
    ForwardBackwardAveraging = false;
  end

  % Following properties are only applicable to UCA
  properties (Access = protected, Nontunable)
```

```
 pPhasedModeTransform = false;
 % K * Radius
 pKR;
 %beamspace or ula space
 pIsBeamspace = true;
end
properties (Access = protected)
 % phase mode beamformer
 pFeH;
 % Holds either phase mode beamformer
 % or the elevation dependent ULA space
 % beamformer weights
 pConjugatedBFWeights
 % Bessel function diagonal matrix
 pJ
 % last used elevation angle
 pLastElAng;
end

methods
      sensorArray = obj.SensorArray;
      if isa(sensorArray,'phased.UCA')
         obj.pPhasedModeTransform = true;
         wavelength = obj.PropagationSpeed/obj.OperatingFrequency;
         k = 2*pi/wavelength;
         obj.pKR = k*sensorArray.Radius;
         Nuca = getNumElements(obj.SensorArray);
         maxPhaseMode = getMaxPhaseMode(obj);
         V = getPhasedModeW(Nuca,-maxPhaseMode:maxPhaseMode);%Eq 11 in [2]
         Cj = diag(1i.^[-maxPhaseMode:0 -1:-1:-maxPhaseMode]); %Eq 10 in [2]
         %phase mode beamformer
         obj.pFeH = (Cj*V').'; %Eq 9 in [2]
```

```
      obj.pConjugatedBFWeights = obj.pFeH;
      if obj.SpatialSmoothing
         obj.pIsBeamspace = false;
      end
      %Initialize to an invalid value to indicate nothing cached yet
      obj.pLastElAng = 91;
      obj.pJ = zeros(2*maxPhaseMode+1,2*maxPhaseMode+1);
   end
end

function doas = stepImpl(obj, Xarg,varargin)

   classtouse=class(Xarg);
   fb = obj.ForwardBackwardAveraging;
   phasedModeTransform = obj.pPhasedModeTransform;
   if phasedModeTransform
      elAng = varargin{1};
      updateBeamformer(obj,elAng);
      % Transform input from element space to beamspace
      X=  Xarg*obj.pConjugatedBFWeights;
   else
      X = Xarg;
   end
   Cx = cast(step(obj.cCovEstimator,X),classtouse);
   [eigenvals, eigenvects] = privEig(obj,Cx);
   eigenvals = cast(eigenvals,'double');
   eigenvects = cast(eigenvects,'double');

   % Note when ForwardBackwardAveraging is true, the returned
   % eigenvects are the eigenvectors of Sq instead of the covariance
   % matrix. We have to further deduct eigenvects of the covariance
   % matrix.
```

```
if fb
   eigenvects = local_convert(eigenvects);
end

% Obtain signal subspace dimension
Nsig = getNumSignals(obj,eigenvals,obj.pNumSnapshots,fb);
if Nsig==0
   if isempty(coder.target)
      warning(message('phased:phased:doa:ZeroSourceNumber'));
   end
   doas = zeros(1,0,classtouse);
   return;
end

% Core Root-MUSIC algorithm
N = obj.pEffChannels;
% Separate the noise eigenvectors
noise_eigenvects = eigenvects(:,Nsig+1:end); %Qn
if phasedModeTransform && obj.pIsBeamspace % when in beamspace
   % polynomial D = Sbeamspace'*Qn*Qn'*Sbeamspace
   %              Sula'*J'*Qn*Qn'*J*Sula
    Qn = obj.pJ*noise_eigenvects;
else %when in ULA or ULA like space
   % polynomial D = Sula'QnQn'Sula
    Qn = noise_eigenvects;
end
doasCM = cast(privCoreRootMUSIC(Qn,Nsig,N),classtouse);
%doasCM = privCoreRootMUSIC(noise_eigenvects,Nsig,N);
if phasedModeTransform
   % only need to convert from rad to degrees here since
   % Z = e^j*theta, angle(Z) = theta (root music polynomial in
   % terms of z) Eq 13 in [2]
```

```
            doasRad = doasCM*180/pi;
            % convert to row vector
            doasCol = doasRad(:);
            doas = doasCol.';
        else
            % Convert angles when ULA
            doas = privConvertAngles(obj,doasCM);
        end
    end
    function privValidateSensorArray(~,val)
        cond = ~(isa(val,'phased.ULA') ...
            || isa(val,'phased.HeterogeneousULA') ...
            || isa(val,'phased.UCA'));
        if cond
            coder.internal.errorIf(cond,'phased:phased:RootMUSICEstimator:InvalidArray');
        end

        if isa(val,'phased.UCA')
            elem = val.Element;
            cond = ~(isa(elem,'phased.IsotropicAntennaElement') || ...
                isa(elem,'phased.OmnidirectionalMicrophoneElement') || ...
                isa(elem,'phased.ShortDipoleAntennaElement') || ...
                isa(elem,'phased.CustomAntennaElement') || ...
                isa(elem,'phased.CustomMicrophoneElement'));
            if cond

coder.internal.errorIf(cond,'phased:phased:RootMUSICEstimator:InvalidElement',class(elem));
            end
        end

    end
    function N = getNumEffectiveElements(obj)
```

```
        if isa(obj.SensorArray,'phased.UCA')
            maxPhaseMode = getMaxPhaseMode(obj);
            N = 2*maxPhaseMode+1;
        else
            N = getNumElements(obj.SensorArray);
        end
    end
     function num = getNumInputsImpl(obj)
        if isa(obj.SensorArray,'phased.UCA')
            num = 2;
        else
            num = 1;
        end
     end
end


function doas = privCoreRootMUSIC(Qn,Nsig,N)
    roots_D = roots(D);
    roots_D1 = roots_D(abs(roots_D) < 1);
    [~,indx] = sort(abs(abs(roots_D1)-1));
    sorted_roots = roots_D1(indx);
    doas = angle(sorted_roots(1:Nsig)); %get phase of the roots (z)

end

function eigenvects = local_convert(eigenvects)
% Deduct eigenvects of the covariance matrix from eigenvects of Sq

    M = size(eigenvects,1);

    half = floor(M/2);
```

```
        Uc1 = eigenvects(1:half,:);

        if rem(M,2)
            Uc2 = eigenvects(half+2:M,:);
            Umid = sqrt(2)*eigenvects(half+1,:);
        else
            Uc2 = eigenvects(half+1:M,:);
            Umid = [];
        end
        % Note: J*X2 with J = fliplr(eye(half)) is equivalent to flipud(X2)
        % eigenvects = 1/sqrt(2)*[Uc1+i*Uc2;Umid;J*(Uc1-i*Uc2)];   % Eq 7.68 in [1]
        eigenvects = 1/sqrt(2)*[Uc1+1i*Uc2;Umid;flipud(Uc1-1i*Uc2)];

    end

    function w = getPhasedModeW(N,phaseModes)

    w = complex(zeros(N,numel(phaseModes)));
    colIdx = 1;
    elAzimuth = (-(N-1)/2:(N-1)/2)*2*pi/N; %in radians
    for phaseMode  = phaseModes
        %Eq 4 in [2], note that the element positions on the circle are ordered
        %differently in our toolbox than in [2]
        w(:,colIdx) = (exp(1i*phaseMode*elAzimuth)/sqrt(N))';
        colIdx = colIdx+1;
    end
end
```

A-7 ROOT-WSF

```
function loadObjectImpl(obj,s,wasLocked)
   s = loadSubObjects(obj,s);
   fn = fieldnames(s);
   for m = 1:numel(fn)
      obj.(fn{m}) = s.(fn{m});
   end
end

function doas = stepImpl(obj, X)

   K = obj.pNumSnapshots;
   N = obj.pEffChannels;

   % Compute the eigenvectors/eigenvalues of the spatial
   % covariance matrix
   Cx = step(obj.cCovEstimator,X);
   [eigenvals eigenvects] = privEig(obj,Cx);

   % Obtain Signal subspace dimension
   D = getNumSignals(obj,eigenvals,K,false);
   if D==0 && isempty(coder.target)
     warning(message('phased:phased:doa:ZeroSourceNumber'));
     doas = zeros(1,0,class(X));
     return;
   end

   % compute constants
   if (obj.Method(2) == 'M') % 'IMODE'
      % Define handle to the function that will compute Qd
      Gamasv = eigenvals(1:D);
      sigmaw2 = sum(eigenvals(D+1:N))/(N-D);  % Eq (8.525) in [1]
```

```
    Waov = (Gamasv-sigmaw2).^2./Gamasv;  % Eq (8.523) in [1]
    Us = bsxfun(@times,eigenvects(:,1:D),sqrt(Waov(:).'));
    IN = Us.';
    K = D;
else
    IN = X;
end
T = transformation_matrix(D); % Transformation matrix

% Step 1: Initialization
[m,test,B,b] = local_init(N,D);

maxiter = obj.MaximumIterationCount;
while test && m<maxiter
    % Step 2: Compute Q
    f = local_compute_Qx(B,IN,N,D,K);
    Q = T'*f*T; % Eq (8.539) IMODE or Eq (8.514) IQML in [1]

    % Step 3: Compute the Eigenvectors of Real(Q)
    [V,E] = eig(real(Q));
    [~,idx] = min(diag(E));
    c = V(:,idx);                    % Eq (8.540) in [1]

    % Step 4: Find roots of polynomial b
    aux = b;
    b = T*c;                         % Eq (8.541) in [1]
    b_conj = conj(flipud(b));
    B = toeplitz([b_conj;zeros(N-D-1,1)],[b_conj(1) zeros(1,N-D-1)]);
    m=m+1;                           % Number of iterations
    test = (norm(aux-b)>sqrt(eps));      % Eq (8.543) in [1]
end
```

```
    doas = angle(roots(cast(b,class(X))));              % Eq (8.542) in [1]

    % Convert angles
    doas = privConvertAngles(obj,doas);


function [m,test,B,b] = local_init(N,D)
% Initialization

  m = 0;                  % Counter
  test = true;
  B = complex([eye(N-D);zeros(D,N-D)]); % Eq (8.538) in [1]
  b = complex([zeros(D,1);1]);

end

function Qx = local_compute_Qx(B,X,N,D,K)
% X,N,D are constants

  R = B'*B;
  Qx = complex(zeros(D+1,D+1));
  for k = 1:K, % Loop over the number of snapshots
    Ak = complex(zeros(N-D,D+1));
    for n = 1:N-D,
      idx = n+D:-1:n;
      Ak(n,:) = X(k,idx);      % Eq (8.498) in [1]
    end
    Qx = Qx + Ak'/R*Ak;
  end

end
```

```
function T = transformation_matrix(D)
% Transformation Matrix T is (D+1) x (D+1)

   if rem(D,2),
      % D odd
      I = eye((D+1)/2);
      J = fliplr(I);   % Exchange matrix
      T = 1/sqrt(2)*[kron(I,[1 1i]);kron(J,[1 -1i])]; % Eq (8.506) in [1]
   else
      % D even
      I = eye(D/2);
      J = fliplr(I);
      T = 1/sqrt(2)*[...
         [kron(I,[1 1i]) zeros(D/2,1)];...
         [zeros(1,D) 1];...
         [kron(J,[1 -1i]) zeros(D/2,1)]];
   end

end
```

Appendix B
KerberoSDR

Powering and Connecting the Hardware

Your KerberosSDR should already come fully assembled.

As there are four on board RTL-SDRs, KerberosSDR has fairly high power requirements. A single USB 3.0 port on a computer that can handle up to 3A is enough to power the device from one high quality USB cable. Note that only the microUSB port is connected to the computers USB port and that the USB-C port (Batch 1), or the right USB micro (Batch 2 or newer) is only for additional power.

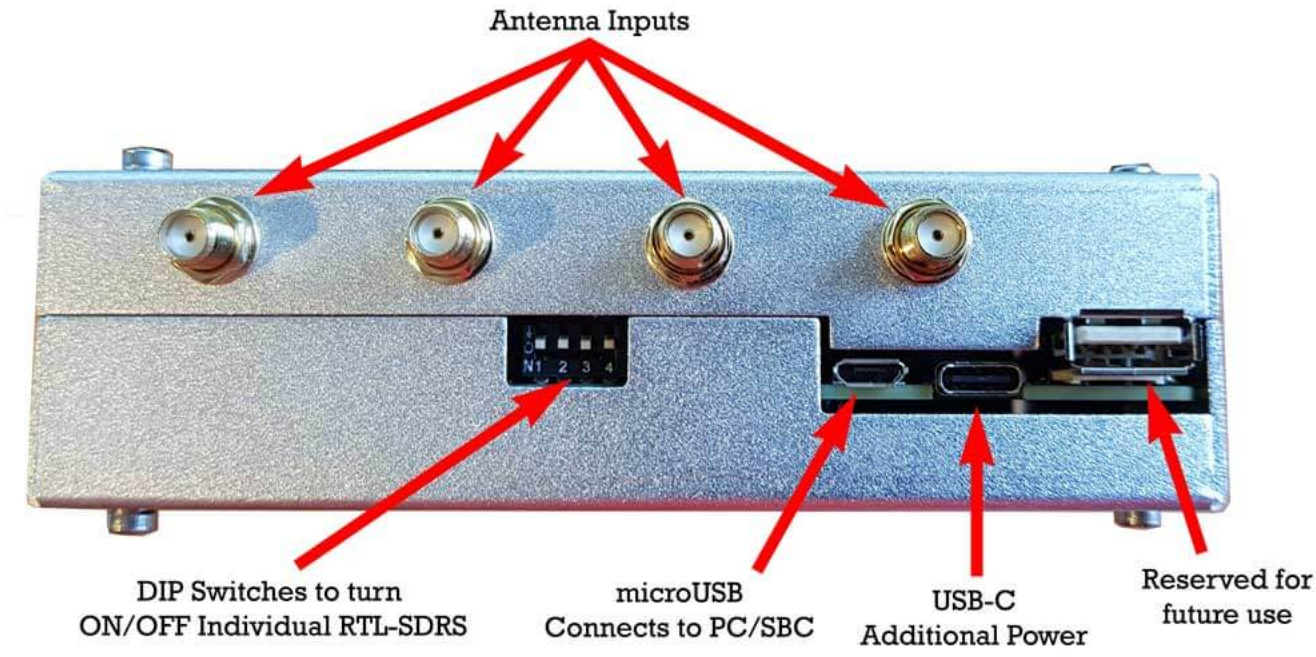
If you only have USB 2.0 hardware, or cannot provide 3A from the USB port (e.g. single board PCs cannot provide this much current), then you'll need to plug a second power source into the USB-C port (Batch 1)/right USB micro (Batch 2 or newer), on the KerberosSDR. This power source should be an plugged into a wall 5V USB supply or external 5V battery if going portable. The supply should be able to deliver 2.4A.

Note that the USB-C port on batch 1 units will not activate for high voltage "power delivery" (PD) USB-C power supplies, such as those used with laptops. It will only work on normal 5V power supplies.

BATCH 2 USB CHANGES (BATCH 2 SHIPPING SINCE FEB 1 2019):

Please note the USB-C port shown in the image below has been replaced with a microUSB port on batch 2 and newer productions.

Due to too many people having issues finding USB-C supplies that work, batch 2 reverts to microUSB on the power port. If you purchased after FEB1 2019, you will have a batch 2 unit.



KerberosSDR Labelled Ports

Antenna 1 is the port to the left. You can confirm this by looking at the DIP switches. The writing on the DIP switches indicates the antenna order.

40 Pin Header

Please note that this header is experimental only and we are not supporting use of this feature at the moment.

It is designed for powering a Raspberry Pi. If you connect a Raspberry Pi to the header, and power the KerberosSDR you can power it this way. But you must have a very good power supply for the KerberosSDR.

Raspberry Pi/Single Board Computer (SBC) Ready to Use Images

Download the appropriate .img file below, unzip it, and burn it to an 8GB or larger SD Card for your SBC using Win32DiskImager or Etcher.

On your SBCs, it is important to make sure thermal throttling is not occurring (via appropriate heatsinking/fans on the SBC if needed) otherwise sample loss, and hence coherent lock failure may occur.

Raspberry Pi 3 / 4

External WiFi AutoConnect Hotspot: KerberosAndroid
External WiFi AutoConnect Password: KerberosAndroid
External WiFi IP Address: Check WiFi Hotspot Device for assigned IP

Internal WiFi Hotspot: KerberosPi
Internal WiFi Password: KerberosPi
Internal WiFi IP Address: 192.168.4.1 (for VNC, web interface and Android App)

VNC & Linux Username: pi
VNC & Linux Password: raspberry

Internal Hotspot Address for Web Control: IP_ADDR:8080/init
Internal Hotspot Address for Android App: IP_ADDR:8081

Update Notes: Only the latest IMG V1.5 & 1.6 works on the Pi 4. IMG1.6 fixes long term sync stability on the Pi 4, upgrades the web UI, add PR peak hold & improves update rates significantly. 1.6.1 Fixes a HTML bug on the passive radar web display.

Download IMG 1.6.1: https://drive.google.com/open?id=1DhCLYv99QHwpJRM2gvG6UtrIM1RO8_Ua

Download IMG V1.6: https://drive.google.com/open?id=1MK9OiGCznLAv8hENDZoOMKl_F0_GYCZs
Download IMG V1.5: https://drive.google.com/file/d/14AVWcNOKr6wyBtfkwIBFqNrN3d9xAweA/view?usp=sharing
Download IMG V1.4: https://drive.google.com/file/d/1H77bUt9AXVZC8NXvcl0cBubuh9vuMTti/view?usp=sharing
Download IMG V1.3: https://drive.google.com/file/d/1Vwt-bcQNDP6KEWtqdadOdZOT-LBGQXhj/view?usp=sharing

Tinkerboard

External WiFi AutoConnect Hotspot: KerberosAndroid
External WiFi AutoConnect Password: KerberosAndroid
External WiFi IP Address: Check WiFi Hotspot Device for assigned IP

Internal WiFi Hotspot: Kerberos_Tinker
Internal WiFi Password: kerberos
Internal WiFi IP Address: 192.168.4.1 (for VNC, web interface and Android App)
Linux Password: kerberos

Default Internal Hotspot IP address for VNC: IP_ADDR:5900 (no password)

Address for web control: IP_ADDR:8080/init
Address for Android App: IP_ADDR:8081

Download IMG V1.4: https://drive.google.com/open?id=1shO-9xD2prrqGBSg9Zjj5tVCighqPg3p

Download IMG V1.3: https://drive.google.com/file/d/1Btter1AWNWV0VCnXqfosu5E0V6E3RTXO/view?usp=sharing

---

Note please see our current issues forum thread for any currently known issues.

Other SBCs

We have decided to release images only for the Pi 3 and Tinkerboard at the moment. Other SBCs like the Odroid XU4 do not have built in WiFi and the Rock Pi 4 appear to have show stopping bugs with the WiFi Hotspot which we use for connections. However, the software does work on the XU4 and Rockpi4, so feel free to

2. Plug KerberosSDR into SBC, and connected the KerberosSDR's power cable.
3. On your Android device create a Mobile Hotspot with username/password "KerberosAndroid" on your phone.
4. Turn on your SBC, and wait a couple of minutes for it to boot and connect.
5. Check your Android Mobile Hotspot settings for the connected devices, and get the IP address.
6. Enter the address for web control (IP_ADDR:8080/init) in a browser or in the app.
7. Go through configure and sync steps described further below.

Manually Installing the software on a PC / Other SBC

NOTE: There has been some confusion around this, if you are using the ready to use images shown above you DO NOT need to follow this manual installation guide as the software is already pre-installed on the ready to use images. These manual installation steps are only for installing the software yourself on a PC or your own SBC.

The KerberosSDR software has been tested on Ubuntu Debian systems.

Install Dependencies via apt:

```
sudo apt update
sudo apt install python3-pip python3-pyqt4 build-essential gfortran libatlas3-base libatlas-base-dev python3-dev python3-setuptools libffi6 libffi-dev py
```

Just in case, try to uninstall any previously installed numpy packages as we want to install with pip3 to get optimized BLAS routines.

```
sudo apt remove python3-numpy
```

Install Dependencies via pip3:

```
pip3 install numpy
pip3 install matplotlib
pip3 install scipy
pip3 install cairocffi
pip3 install pyapril
pip3 install pyargus
pip3 install pyqtgraph
pip3 install peakutils
pip3 install bottle
pip3 install paste
```

Install RTL-SDR-Kerberos Drivers

Our Kerberos RTL-SDR driver branch contains code for slightly modified Osmocom RTL-SDR drivers that enable GPIO, disable dithering, and disables zerocopy buffers which seems to cause trouble on some ARM devices.

```
sudo apt-get install libusb-1.0-0-dev git cmake

git clone https://github.com/rtlsdrblog/rtl-sdr-kerberos

cd rtl-sdr-kerberos
mkdir build
cd build
cmake ../ -DINSTALL_UDEV_RULES=ON
make
sudo make install
sudo ldconfig

echo 'blacklist dvb_usb_rtl28xxu' | sudo tee – append /etc/modprobe.d/blacklist-dvb_usb_rtl28xxu.conf
```

Now you may need to reboot, but on most systems you should be able to run rtl_test as described below already.

Test 4x Tuners (Optional)

At this stage we recommend first testing your four tuners with rtl_test. Open four terminal windows, or tabs, and in each window run "rtl_test -d 0", "rtl_test -d 1", "rtl_test -d 2" and "rtl_test -d 3". Ensure that each unit connects and displays no errors.

If you are running on an SBC, you will probably want to first edit the C code Makefile compiler flags to be optimized for your particular architecture. The Makefile is stored in _receiver/C. Otherwise if running on a PC/laptop, simply run the following script to compile the code.

```
cd kerberossdr
sh setup_init.sh
```

(Optional if you are using an SBC and/or intend to use the web interface features). Using a text editor like gedit or nano, edit the IP address parameter in run.sh, and set it to your machines IP address. On SBCs you may also wish to enable the lines in run.sh that enable the "performance" CPU mode to reduce the likelihood of CPU throttling. On a Pi 3 or other SBC that shares Ethernet/WiFi and USB bandwidth please also enable the wondershaper lines, to ensure the USB bandwidth is not suffocated by network use.

Now you can run the software by typing:

```
./run.sh
```

Synchronizing and Calibrating KerberosSDR in the Demo Software

These steps assume that you've connected your KerberosSDR to the PC and power. The steps to the software are summarized as follows:

1. Set the frequency, sample rate and gain settings.
2. Calibrate the SDR with the noise source.
3. Connect antennas.
4. Use the DOA or Passive Radar tabs.

Setting The Frequency and Gain

1. Click on the Configuration and Spectrum screen

2. Click on start processing. Confirm that an update rate is showing.

3. Under receiver configuration enter the center frequency of your signal of interest, and a sampling rate. For sampling rates we recommend 2.56 MSPS for passive radar, and 1.0 MSPS for direction finding.

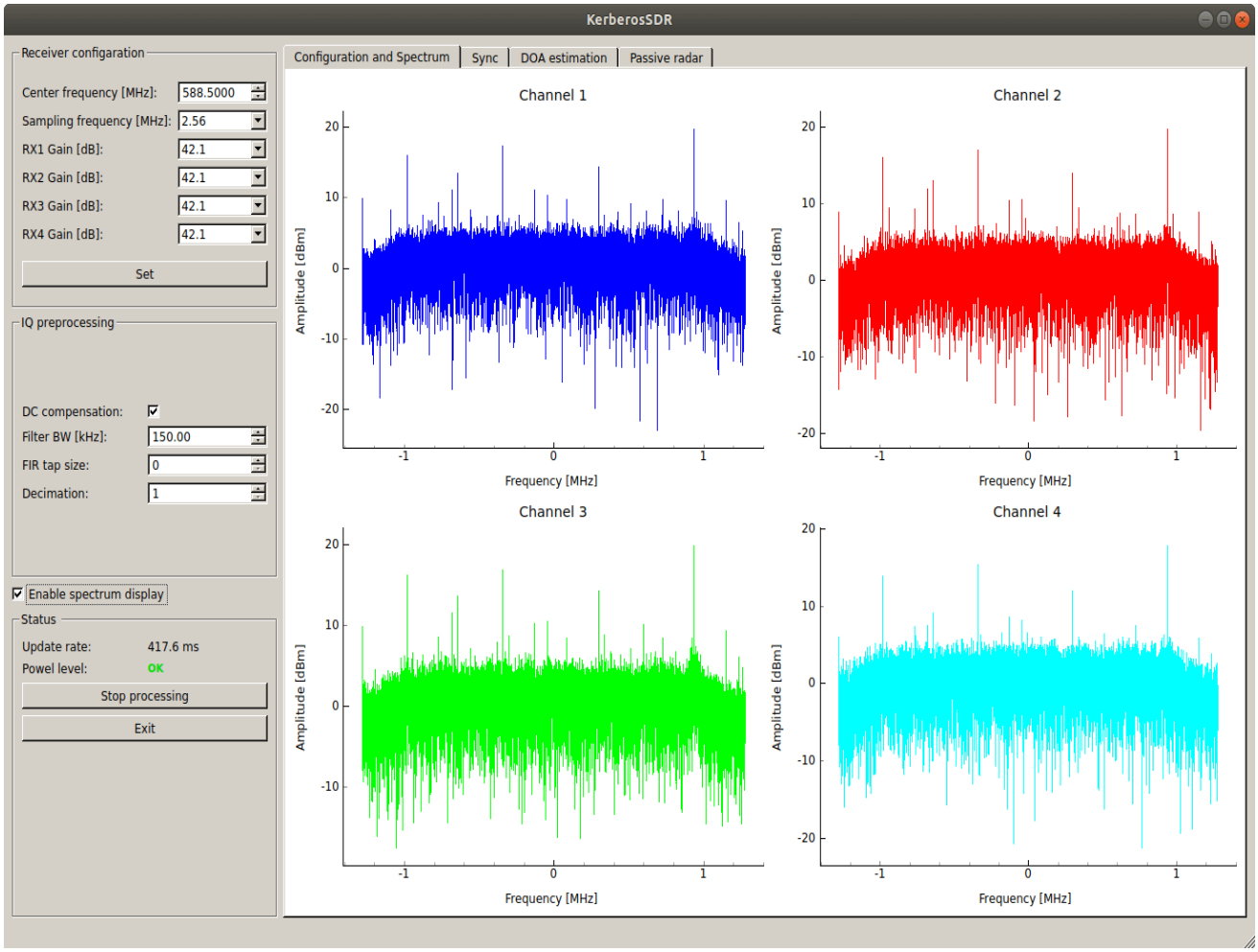
4. Set the gains for each receiver as required to optimize the signal SNR. The gains of each receiver must be set at 15.7 or higher in order to successfully sync KerberosSDR. (Note that Passive Radar only needs the first two receivers, but you still need to set the gains on all receivers above 15.7 for sync)

5. Click on the "Set" button to write the frequency, sampling rate and gain settings to the KerberosSDR.

6. Check DC compensation on for slightly better results (optional).

7. Click the "Enable spectrum display" checkbox if desired to check the spectrum. This will show you the spectrum visually and you can check that your signals are being received.

8. Disable the spectrum display when finished to reduce the CPU load.



Configuration and Spectrum Screen

Synchronization and Calibration Process

Initial Step: When synchronizing, we recommend disconnecting your antennas first.

- **Direction Finding:** Disconnect antennas and optionally connect the included 4x 50 Ohm terminators. Connecting the terminators is optional, but doing so may give you *slightly* more accurate results. You must perform both sample and phase calibration.
- **Passive Radar:** If you are doing passive radar, you do not need to connect the terminators. Only sample calibration is required.

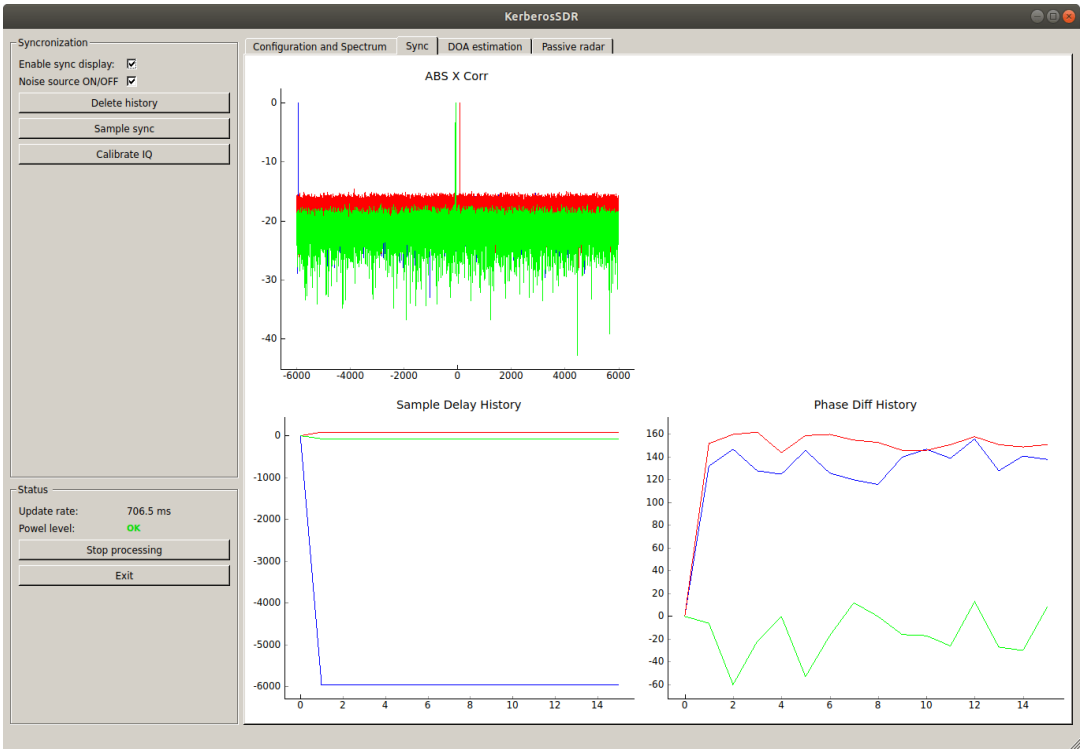
Alternatively, you can also synchronize your KerberosSDR with the antennas connected. BUT, if there are signals within the bandwidth being currently received then this can distort the phase calibration. So we recommend checking for signals first in the spectrum graphs.

IMPORTANT: Always calibrate with the FIR filter set to zero, and the decimation set to 1.

1. Calibration assumes that you have already set the desired frequency, sample rate and gains. Every time you change those settings by clicking on the "Set" button, you will need to re-calibrate.

2. Click on the Sync tab.

3. Enable the Noise source and sync display check boxes. Notice that the Abs X corr graph has three peaks, and that the Sample delay history and phase diff history is non-zero. This indicates that the KerberosSDR is not synced.

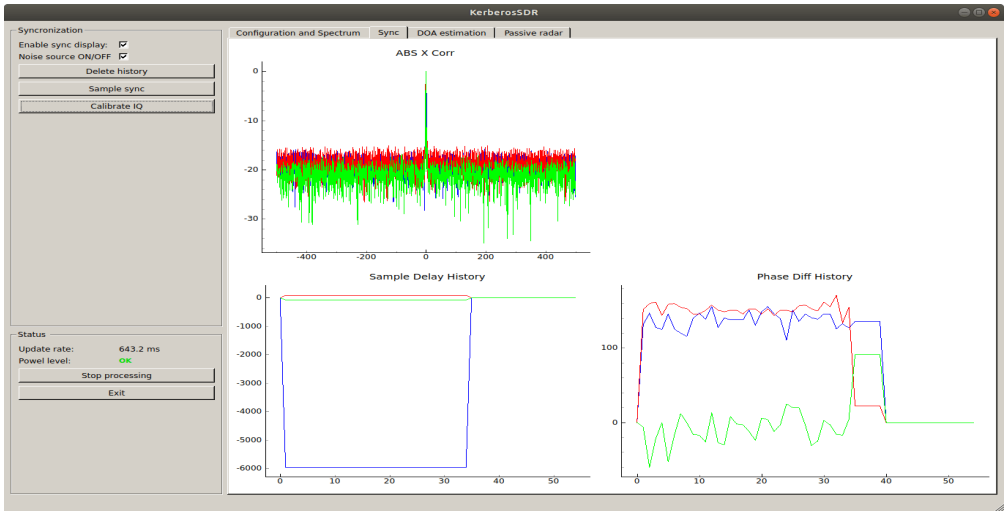


Unsynced KerberosSDR Graphs

4. Click on "Sample Sync". Wait until the sample delay history graph and Abs X Corr graphs become zero'd.

5. If doing direction finding, click on Correct IQ. Wait until the Phase diff history graph is zero'd.

6. At this point KerberosSDR is in sample and phase sync. Please note that you must perform these steps every time you start the application, or every time that you tune to a different frequency or change the gain settings.



7. You can now uncheck the noise source and graph check boxes.

Troubleshooting

If for some reason you are having issues follow the troubleshooting guide below

1. Ensure that you are using a high quality microUSB data cable. To date, the most common issue has been people with cheap microUSB cables. Cheap low quality cables won't be able to handle the data transfer rates required by the 4x RTL-SDR dongles.

2. Double check that your KerberosSDR is powered correctly via the USB-C port. When plugged in there should be a white light glowing inside the enclosure (you can peek through where the DIP switches are).

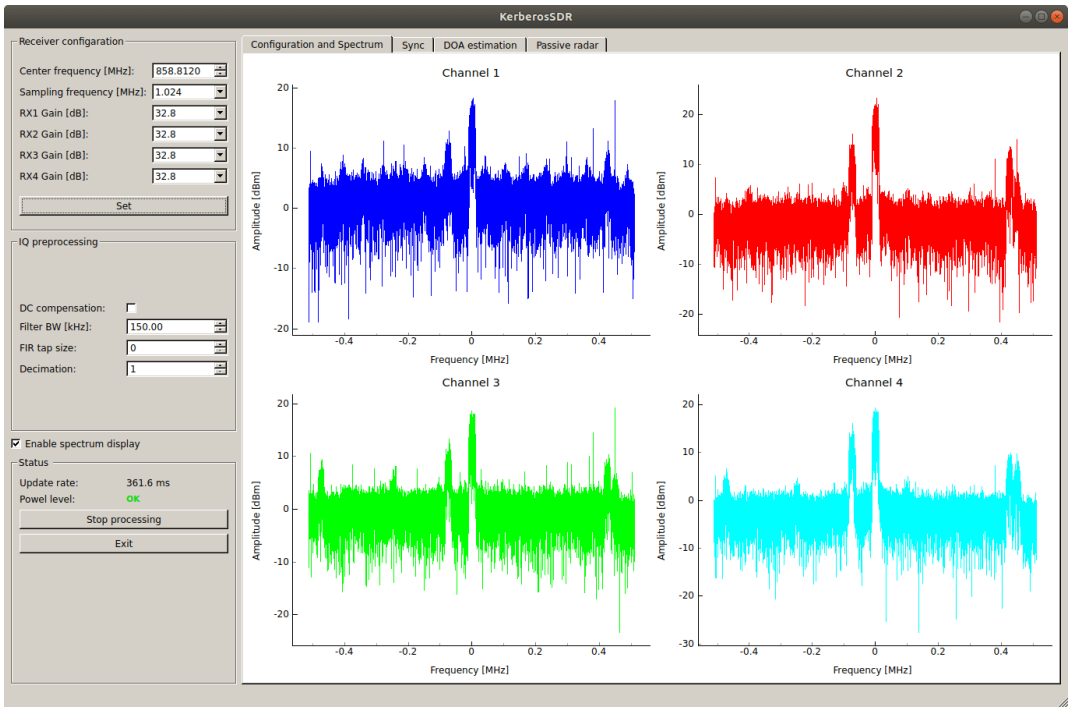
3. You can enable logging by editing the execution line in the run.sh file. Comment out the execution line, and uncomment the commented one that has logging. Check the log files after trying to run the code. (NOTE, older versions and some current images may not have the logging line. Simply edit the execution line and remove the output redirection to /dev/null.

Direction Finding

These steps assume that the synchronization and phase calibration steps have been completed.

1. Connect your direction finding antenna array (more details on that below), and go back to the Configuration and Spectrum tab.

2. Enable the spectrum display and check that your signal of interest is centered in the spectrum display.

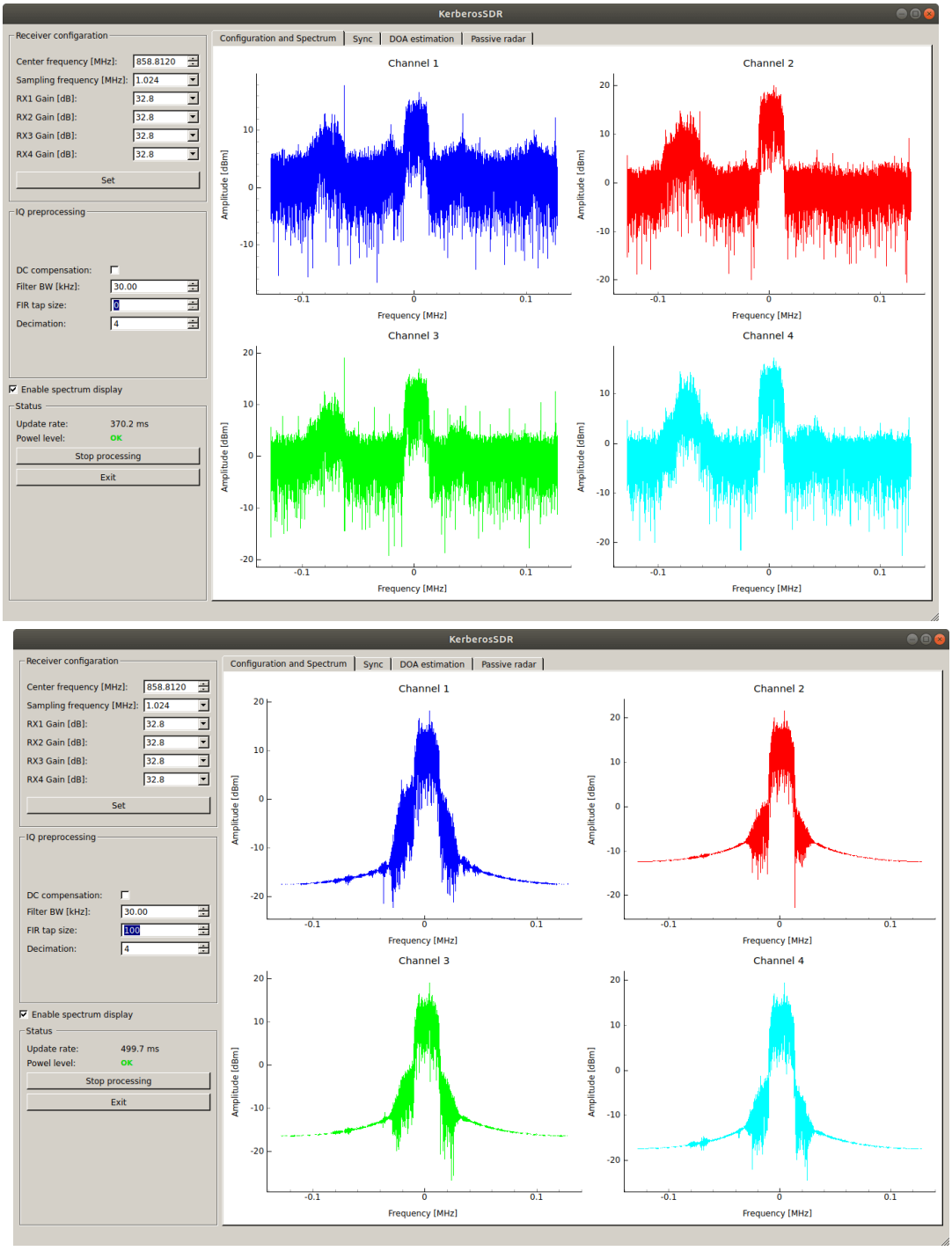


Tuned to a signal of interest

3. Using the decimation, FIR tap size and filter BW settings, reduce the bandwidth so that only the centered signal of interest is being received. You will want to use these options to filter out other signals, so that only your signal of interest can be seen in the spectrum.

   For example, for a 25 kHz signal, we might set the filter bandwidth to 25 kHz, the FIR tap size to 100, and the decimation to 4.

   IMPORTANT: The decimation value must be a power of two with a maximum value of 8, for example: 1, 2, 4, 8. A decimation value of 2 will halve the bandwidth, a value of 4 will halve it once again.



FIR filter enabled. Nearby signals removed from bandwidth. Only the signal of interest remains.

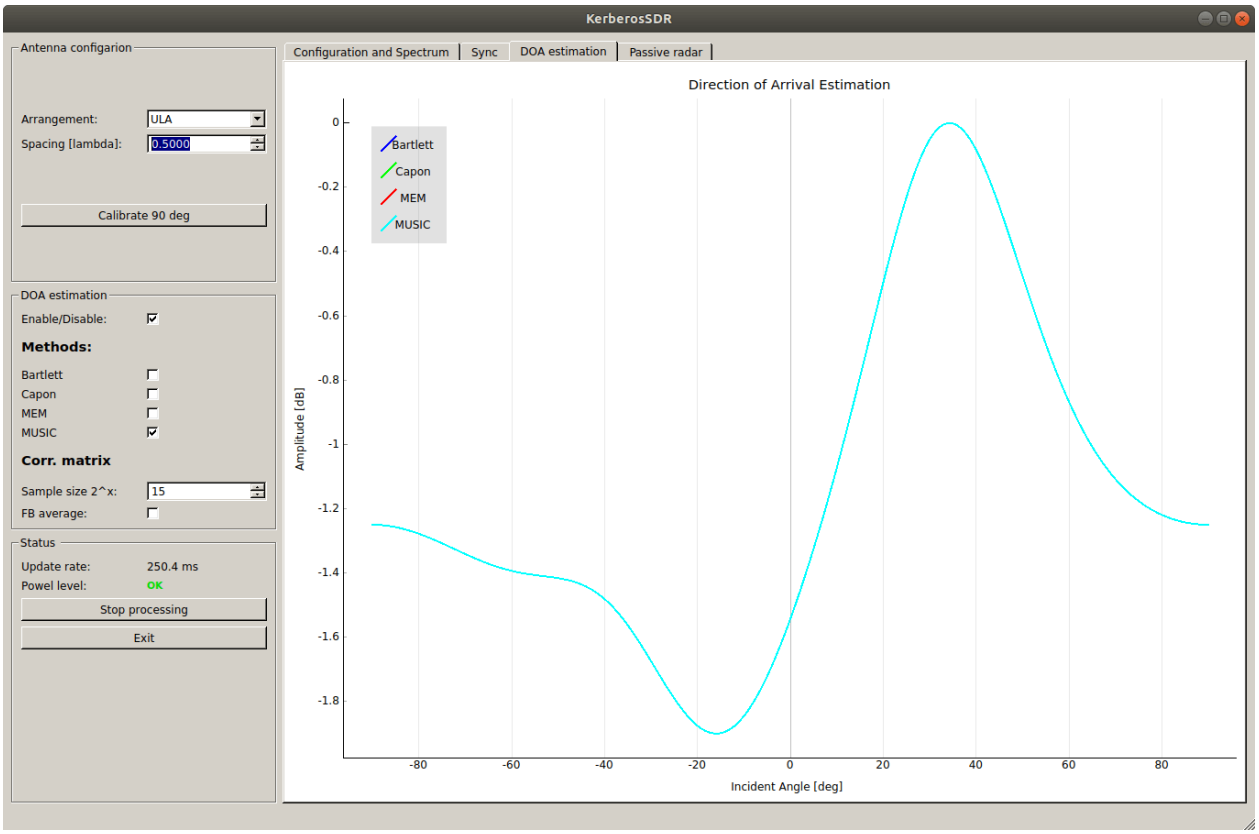
4. Disable the spectrum display.

5. Click on the DOA Estimation tab.

6. Use the Arrangement drop down box to choose whether your antennas are arranged in a uniform linear array (ULA) or uniform circular array (UCA) (see further down for antenna array details).

7. Set the spacing factor of your antennas (see further for details).

8. Check the Enable box, and choose **ONE** DOA algorithm. The best algorithm for most applications would be MUSIC.



Antenna Interelement Spacing Factor

The antennas need to be spaced out from one another ideally at a spacing factor of somewhere between 0.1 - 0.5 multiplied by the frequency wavelength. The interelement spacing factor **must be below 0.5 to avoid ambiguities**. **And in practice, you should not have the spacing factor come close to 0.5.**

Larger spacing factors result in higher resolution, but require more space. Higher resolution means thinner peaks on the graph, and the ability to differentiate between multiple angles including multipath. But often this results in a multipath signal sometimes showing as more dominant. Smaller spacing factors and thus smaller resolutions can in effect average multipath as the peaks will combine into one wider peak that could be slightly skewed. Feel free to experiment with different spacing factors, but we've found that **1/3 or 0.33 tends to be a good compromise and works the best in most cases, but we have had success all the way down to 0.1.**

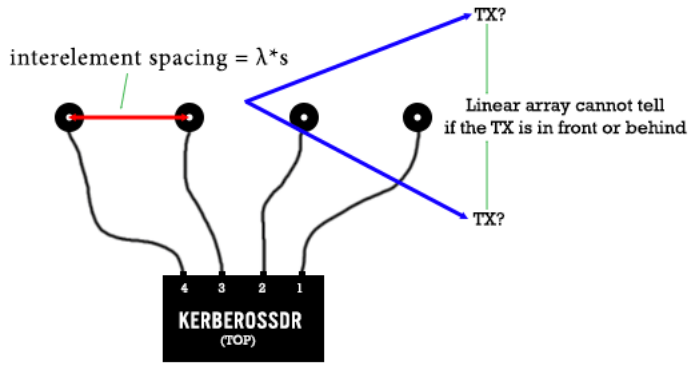
ULA - Uniform Linear Array

This is simply a straight line of omnidirectional antennas (e.g. magnetic whips or dipoles) spaced out at λ * s, where λ is the frequency wavelength, and s is the interelement spacing factor. As discussed previously, in order to avoid ambiguities, the interelement spacing factor must be less than 0.5.

Ideally the interelement spacing factor is somewhere between 0.1 to 0.5, and we normally recommend 1/3 or 0.33.

We recommend starting your direction finding experiments with a uniform linear array as this is the easiest antenna arrangement to test with. The circular array could produce more false results due to the greater risk of multipath. However, note that a linear array cannot tell if a transmitter is in front or behind it.

If going hand held, it can be a good idea to place the antennas on a stick that can be rotated. The most accurate results come from when the antennas are at 0 degrees (facing the signal). So one technique is to rotate the stick until the KerberosSDR software reads 0 degrees. A compass can be placed in the middle of the stick, and the direction perpendicular to the antennas is the direction of the transmitter.



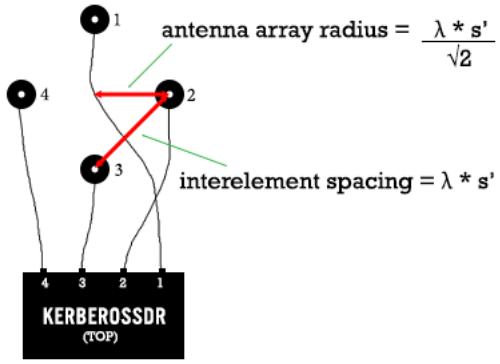
λ = frequency wavelength
s = interelement spacing factor (0.1 - 0.5 typ.) (~0.33 recommended)

Uniform Linear Array

UCA - Uniform Circular Array

This has the antennas arranged in a circle. Like the linear array, the maximum element spacing must be smaller than half of the frequency wavelength (0.5 * λ) to avoid ambiguities. Again for most cases we'd using about recommend 1/3 the wavelength.

The KerberosSDR software expects the interelement spacing factor to be converted into a radius. So for example, if we chose an interelement spacing of s' = 0.33, the radial spacing factor to enter into the software would be s = 0.33 / (sqrt(2)) = 0.233.



λ = frequency wavelength
s' = interelement spacing factor (0.1 - 0.5) (~0.33 recomended)
s = radius adjusted spacing factor = s' / √2
ENTER s as spacing value in KerberosSDR software

Uniform Circular Array

LATEST CODE VERSION NOTE: In the latest version (graphs have a black background), we've merged community code which replaces the spacing value with just the antenna inter-element spacing. For a circular array, this is just the distance between antennas, there is no need to adjust for the radius.

Physical Limitations of Direction Finding and Things to Know

Multipath: Multipath occurs when signals bounce of walls and other objects. This can cause a signal to appear to be coming from another direction. If the line of sight to the signal is completely blocked, you are likely to receive a reflection more strongly.

If you're using a linear antenna array, you can enable Forward-Backward averaging which can help, but could also make things worse. A directional linear antenna array (like an array of patch antennas) could work even better to reduce the effect of multipath. To get around multi-path issues, we recommend using a vehicle and driving around with our Android app (see below). By moving around you can get multiple readings and average out and ignore false multipath readings.

To reduce multipath it's also important to ensure that the antennas have identical radiation patterns. So if using whip antennas ensure good coupling to the ground plane, and identical lengths of coax etc.

Other Direction Finding Tips

Watch out for signal reflections from the KerberosSDR enclosure itself, your laptop, and any nearby objects. These can skew the results. If possible have these objects underneath the antenna array.

We recommend using short high quality shielded coax cables on all your antennas. If signals enter through the coax cable, this can distort the direction finding system.

If using magnetic mount whips, ensure that you use whips with good coupling to the ground plane (e.g. car roof). Some further advice can be found on the PA8W RDF website.

DOA Algorithms

In the demo software there is a choice from the Bartlett, Capon, MEM and MUSIC direction finding algorithms. Generally we find that MUSIC is the most reliable one, with Capon coming a close second. Unless you know what you are doing, **only activate one algorithm at a time**. Activating more than one will combine the results into an average.

When using a linear array you can activate FB Averaging (Forward-Backward), which averages signals coming in from behind and in front of the antenna array. This reduces the effect of multipath reflections, but can also cause the angle to become skewed. Do not turn it on when using a circular array.

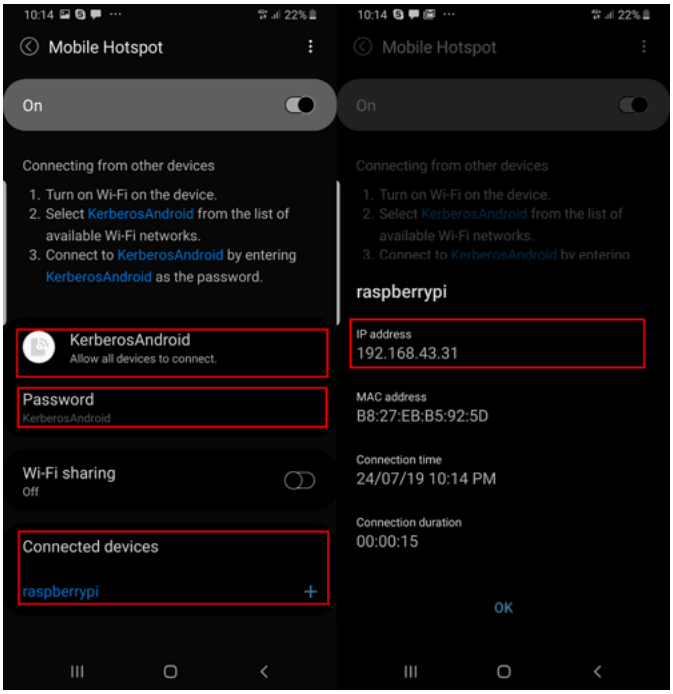
Direction Finding Android App and Web Compass Display

App Download: https://play.google.com/store/apps/details?id=com.rtlsdr.realtimebearing

To use the Android app you will first need to set up a "Mobile Hotspot" on your phone. This will allow you to share the 3G/4G connection on your phone with the KerberosSDR server.

To do this, go into your Android settings and find the Mobile Hotspot setting. It is usually under Connection or WiFi settings. Create a new hotspot with name "KerberosAndroid", and password "KerberosAndroid". Activate the hotspot.

The KerberosSDR software images for the Pi 3 and Tinkerboard (v1.4+) will automatically connect to this hotspot after booting (please wait 2-3 minutes for boot). After it has booted, look at the Mobile Hotspot settings in your Android device and find the "Connected Devices" heading. Use this to determine the IP Address that it has given the server.



If you are using a laptop or your own SBC, you will need to either manually connect, or set up your own autoconnection.

If Mobile Hotspot on your phone is not activated when the KerberosSDR server on the Pi3/Tinkerboard boots, the server will automatically create it's own hotspot. You can instead connect your phone to this hotspot, but be aware that your phone will not have an internet connection, and so features like map download and navigation will not function.

Compass Display

PLEASE NOTE: The compass is currently broken in the Tinkerboard Image. The Android App is working normally.

The compass display runs via a webpage. Browse to http://IP_ADDR:8081/compass.html

Android App Usage

1. Create a Mobile Hotspot on your phone with the username/password KerberosAndroid/KerberosAndroid.
2. Boot the Pi3/Tinkerboard KerberosSDR server and wait 2-3 minutes for it to connect to the hotspot. Find the IP address of the connected KerberosSDR server from your phone.
3. In a web browser, browse to IP_ADDR:8080/init, OR set the IP Address in the Kerberos SDR App settings, then go to the Kerberos SDR Server Settings window via the hamburger menu to access the web interface.
4. Next using the instructions previously described in this guide, set the frequency & gain, phase calibrate the unit, and set the bandwidth, decimation and FIR filtering for your signal of interest.
5. Still on the KerberosSDR web interface enable direction finding, and enable a direction finding algorithm (MUSIC is the most common).
6. Go back to the main screen of the app, then using the menu hamburger (three dots on the top right) open the settings.
7. If you didn't already, change the server address to IP_ADDR:8081, where IP_ADDR should be replaced with the IP address of the machine running KerberosSDR.
8. Choose whether or not you are using a linear or circular antenna array under the "Antenna Array Settings"->"Antenna Array Type" options. If using a linear array, we suggest setting the "Linear Search Direction" setting to both. This will show you both possible direction solutions (back and forward) for a linear array.
9. Choose the bearing mode that you are using. If you're walking or driving around in a car, GPS mode will be the most accurate. If you are stationary and have calibrated your phones compass, you can try the compass mode, although phone compasses can be quite inaccurate. Otherwise if you are stationary you can use the manual mode.
10. Change the log file name to a unique file name with .csv extension.
11. Leave everything else as their default, and go back to the map by using the back arrow.
12. Press the red timer icon on the bottom right to begin direction finding. Once pressed you should see lines indicating the user bearing, and the RF bearing.
13. If you are logging, you can now walk/drive around with your KerberosSDR.
14. You can enable a live direction heatmap grid by going to the hamburger menu and enabling "Plot Grid and Estimate". This grid will automatically update over time, gathering confidence as more data at more locations is collected. Not that this grid only works properly with Circular Antenna Arrays.

Sample Size: The number of samples to use. Higher may result in cleaner results, but slower updates.

Time Domain Clutter Suppression: Suppresses signals reflections from static objects (clutter). Will most likely need to be always on, otherwise the clutter can cause the actual objects to be lost in the noise.

Method: Clutter Suppression method, currently only Wiener - SSMI is available.

Filter Dimension: How far on the PR radar display do you want clutter suppression to be active. Increasing this value enables clutter suppression to be active at further distances. Typically set equal to the max range, but could be reduced to reduce CPU usage.

Max Range: ***Must be a power of 2 (e.g. 64, 128, 256, 512).*** Up to what range of the radar graph do you want to display.

Max Doppler: The max speed on the radar graph that you want to display.

Windowing: The windowing function used in the algorithm. Minor effect, but experiment with this.

Dynamic: Essentially the contrast of the passive radar display. Adjust this so that actual detected objects are displayed strongly on your graph.

We have set up the settings so that the defaults should be fine for most people. Simply enable time domain clutter cancelling, and passive radar processing to start. You can then try tweaking the settings.

Passive Radar Demo Posts

Radar Timelapses

Measuring traffic in a Neighbourhood with KerberosSDR and Passive Radar

Peak Hold Demo

Web Interface

If you have set your IP address in the run.sh file, the KerberosSDR demo software will host a web interface which can be used to control it on a headless system. This is useful for when running on a single board PC, without a monitor as you won't have access to the actual GUI. As an alternative to the web interface, you could also VNC into the machine.

To access the web interface ensure that both devices are connected to the same network, and then browse to (replacing ID_ADDR with the server IP Address):

http://IP_ADDR:8080/init

On the Raspberry Pi and Tinkerboard you can connect to the WiFi hotpost and connect to the default IP address of 192.168.4.1.

On these pages you can set all the settings as in the GUI, and view the graphs. Settings will only be updated once you click on the "Update" or "Start" buttons.

[URIS id=29207]

KeberosSDR Notes and Troubleshooting

I don't want to use the hotspot, and want to connect to my own home WiFi network

For the Raspberry Pi edit "sudo nano /etc/wpa_supplicant/wpa_supplicant.conf", and change the SSID and key to your own home network. After you reboot it should connect to your WiFi network automatically.

For the Tinkerboard, simply enter the network settings GUI via the desktop start menu, and edit the WiFi details there.

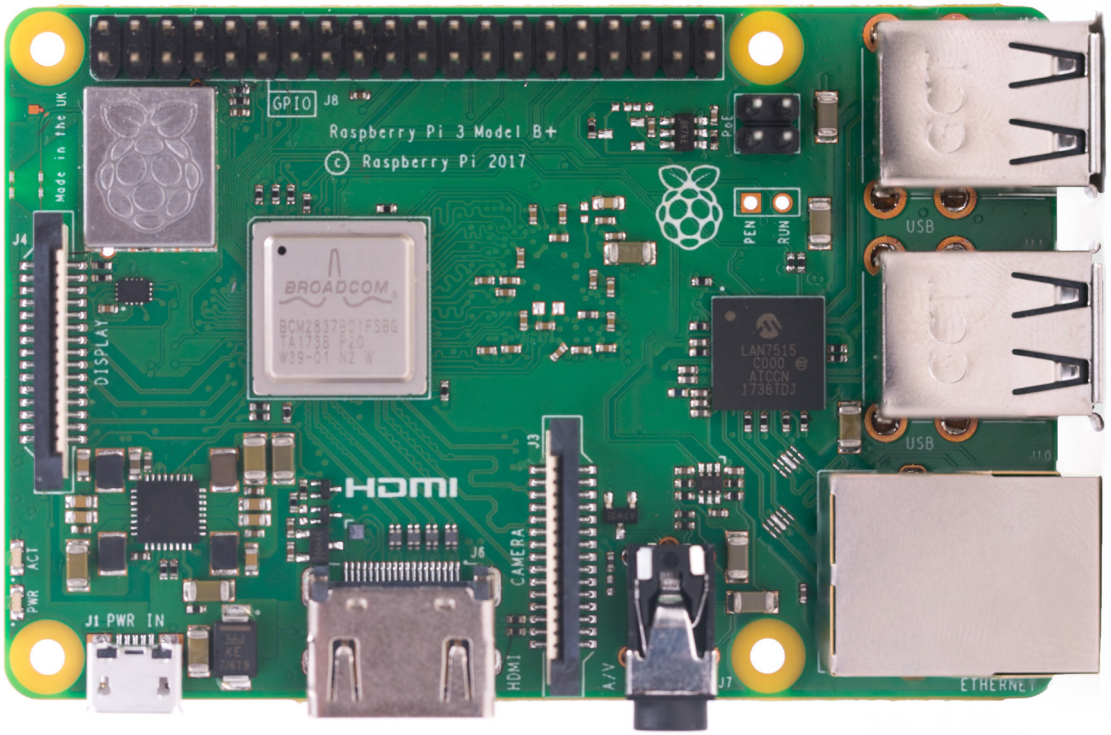
Heat Management / Warm Up - KerberosSDR loses phase sync after 5-10 minutes the first time it's booted

Large temperature swings can cause the KerberosSDR to lose phase sync. If you're starting up the unit and it has been sitting in cold temperatures, let it run for 10 minutes to warm up. During this time watch the sync screen to ensure it stays synced.

In hot environments with stale air, KerberosSDR may lose sample sync if the device gets too hot. In this case keep the device cool with a small amount of airflow. The bottom side of the box acts as a heatsink, so another tip is to place the box upside down or on its side, so that heat can dissipate better.

Appendix C
Raspberry Pi 3 Model B+

Overview



The Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range, boasting a 64-bit quad core processor running at 1.4 GHz, dual-band 2.4 GHz and 5 GHz wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and PoE capability via a separate PoE HAT

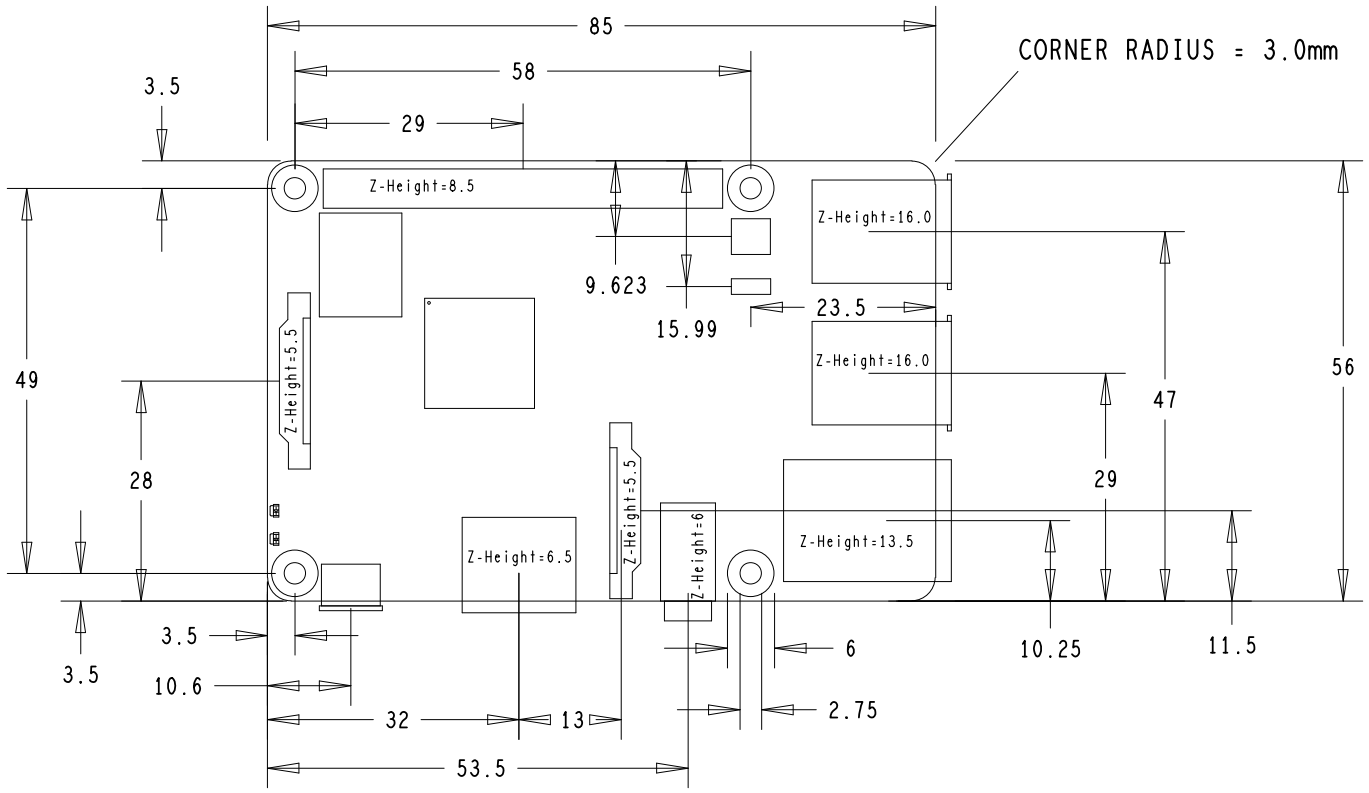
The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products with significantly reduced wireless LAN compliance testing, improving both cost and time to market.

The Raspberry Pi 3 Model B+ maintains the same mechanical footprint as both the Raspberry Pi 2 Model B and the Raspberry Pi 3 Model B.

Specifications

Processor:	Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4 GHz
Memory:	1GB LPDDR2 SDRAM
Connectivity:	■ 2.4 GHz and 5 GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE ■ Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps) ■ 4 × USB 2.0 ports
Access:	Extended 40-pin GPIO header
Video & sound:	■ 1 × full size HDMI ■ MIPI DSI display port ■ MIPI CSI camera port ■ 4 pole stereo output and composite video port
Multimedia:	H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics
SD card support:	Micro SD format for loading operating system and data storage
Input power:	■ 5 V/2.5 A DC via micro USB connector ■ 5 V DC via GPIO header ■ Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
Environment:	Operating temperature, 0–50 °C
Compliance:	For a full list of local and regional product approvals, please visit www.raspberrypi.org/products/raspberry-pi-3-model-b+
Production lifetime:	The Raspberry Pi 3 Model B+ will remain in production until at least January 2023.

Physical specifications



Warnings

- This product should only be connected to an external power supply rated at 5 V/2.5 A DC. Any external power supply used with the Raspberry Pi 3 Model B+ shall comply with relevant regulations and standards applicable in the country of intended use.
- This product should be operated in a well-ventilated environment and, if used inside a case, the case should not be covered.
- Whilst in use, this product should be placed on a stable, flat, non-conductive surface and should not be contacted by conductive items.
- The connection of incompatible devices to the GPIO connection may affect compliance, result in damage to the unit, and invalidate the warranty.
- All peripherals used with this product should comply with relevant standards for the country of use and be marked accordingly to ensure that safety and performance requirements are met. These articles include but are not limited to keyboards, monitors, and mice when used in conjunction with the Raspberry Pi.
- The cables and connectors of all peripherals used with this product must have adequate insulation so that relevant safety requirements are met.

Safety instructions

To avoid malfunction of or damage to this product, please observe the following:

- Do not expose to water or moisture, or place on a conductive surface whilst in operation.
- Do not expose to heat from any source; the Raspberry Pi 3 Model B+ is designed for reliable operation at normal ambient temperatures.
- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.
- Whilst it is powered, avoid handling the printed circuit board, or only handle it by the edges to minimise the risk of electrostatic discharge damage.

Appendix D

RTL-SDR Blog V3 Datasheet

The RTL-SDR Blog V3 is an improved RTL-SDR dongle. RTL-SDR dongles were originally designed for DVB-T HDTV reception, but they were found by hardware hackers to be useful as a general purpose SDR. The standard dongles are okay for DVB-T reception, but are just barely suitable for SDR users/experimenters. The RTL-SDR Blog V3 was redesigned with SDR user needs in mind, instead of DVB-T HDTV users who typically have more relaxed requirements.

Purchase at: www.rtl-sdr.com/store

Quickstart setup guide available at: www.rtl-sdr.com/qsg

Basic Information

- **Bandwidth:** Up to 2.4 MHz stable.
- **ADC:** RTL2832U 8-bits
- **Frequency Range:** 500 kHz – 1766 MHz (500 kHz – 24 MHz in direct sampling mode)
- **Typical Input Impedance:** 50 Ohms
- **Typical Current Draw:** 270 – 280 mA

Required Computing Hardware

Same requirements as a regular RTL-SDR. Compatible with Windows XP and above (SDR# requires Win 7 or newer), Linux, MacOS and Android. A dual core machine is recommended.

Single board PCs like the Raspberry Pi, Odroid, C.H.I.P are also supported with most command line apps.

RTL-SDR V3 Improvements over generic models

TCXO

The V3 uses a 1PPM TCXO for excellent frequency stability. The temperature drift is around 0.5 – 1 PPM, and the initial offset is 0 – 2 PPM. This means that the signal will not drift on the spectrum as the dongle or ambient temperature changes. Also, the frequency offset will be close to zero. Standard dongles have a PPM offset of up to 100PPM, and tend to drift a lot. Using a TCXO solves these problems.

SMA Connector

Typical RTL-SDR dongles use a relatively obscure MCX RF connector. The V3 uses commonly used SMA connectors, so it is easy to obtain adapters, connectors and antennas for the unit. SMA connectors also last longer.

Aluminium Enclosure

Unlike standard RTL-SDR's, the V3 comes standard with an aluminium enclosure. The enclosure has two purposes. The first is to help block any RF interference from entering through the PCB. The second is to act as a heatsink to the PCB.

Improved Heat Dissipation

Typical R820T/2 RTL-SDR dongles tend to lose PLL lock in L-band at around 1.5 GHz and above, causing a loss of reception to those frequencies. The reason is due to the high heat generated by the R820T2 chip. The V3 uses a thin thermal pad to thermally bond the PCB and metal enclosure together. This allows the metal case to work as a heat sink, which solves the PLL lock problem. Ideally the thermal pad should be as thin as possible to enhance maximum heat transfer, and we have designed the enclosure so that the thermal pad only needs to be 3mm thick.

The V3 also uses a larger ground plane on the middle layers of the PCB which also helps with heat dissipation.

R820T2 Chip

Older RTL-SDR units used the R820T chip. There is a newer R820T2 which has slightly better manufacturing tolerances. The R820T2 is produced in a factory with higher quality silicon which allows for more reliable chips. A side effect of the better silicon is overall slightly better and more stable sensitivity across manufacturing runs compared to the R820T, and less PLL lock problems at L-band frequencies.

Improved ESD protection on the RF front end

The BAV99 diode which is used on most RTL-SDR dongles is not a true ESD rated diode. We have added a real ESD rated diode for better protection. The BAV99 remains in the circuit as it works a strong signal clipper, which prevents damage to the R820T2 from overly strong signals. Please remember that not even this will save your radio from a lightning strike or huge ESD impulse, and any permanently outdoor mounted antenna system must have its own lightning and ESD protection. To help avoid lightning damage unplug your antenna during a storm and when the dongle is not in use.

Improved front end circuit

The standard matching circuit on the RTL-SDR was designed for DVB-T use, and tends to attenuate signals above ~1 GHz. The new matching circuit has less attenuation above 1 GHz and similar performance below. We have used high quality, high SRF, high Q inductors in this circuit.

Software switchable 4.5v bias tee.

The V3 makes use of a low noise LDO and one of the GPIO pins on the RTL2832U to provide a 4.5V bias tee that can be activated in software. The bias tee can pull about 180 mA continuously so is suitable for the majority of 3-5V powered LNAs that are popular with RTL-SDR devices. The bias tee is protected against accidental short circuits at the LDO level, and with a thermal auto-resetting PTC fuse. See 'Activating the Bias Tee' for more information on software for activating the bias tee.

This bias tee is great for powering a remote LNA (like Adams PSA5043+ based LNA4ALL) or something like the SpyVerter upconverter.

Bias Tee Warning: The bias tee thermal fuse or LDO could be damaged if you short circuit the bias tee for long periods of time. Before turning on the bias tee, ensure the circuit to be powered is not shorted, or that the RTL-SDR is not connected to a DC shorted antenna!

Lower Voltage Operation

The V3 uses an LDO that has a much lower 'dropout' voltage compared to the typical AMS1117 LDO used on most dongles. Hence the V3 should run better on long USB extension cables.

Long USB cables tend to drop the 5V USB voltage down to lower levels. Below about 4V the AMS1117 stops working. The LDO used in the V3 works almost down to 3.3V.

Of course, with low voltages from long USB cable, the bias tee will be unable to put out 4.5V. At low voltages the bias tee LDO will revert to a non-filtered voltage slightly under the supply.

Reduced noise with a modified PCB design

Typical RTL-SDR dongles use 2-layer PCB designs and route signal lines improperly. The V3 uses a modified 4-layer PCB design which helps to significantly reduces clock spurs and noise pickup.

The V3 also adds a USB common mode choke on the USB data lines to reduce USB noise, adds SMD ferrite chokes on the PCB power lines, and uses a lower noise LDO.

HF direct sampling circuit, diplexed out from the SMA connector

The idea behind direct sampling mode is that an antenna can be connected directly to the ADC pins of the RTL2832U, and this can enable HF reception. This is useful because the R820T/2 tuner can only tune down to about 24 MHz at the lowest. On typical R820T RTL-SDR dongles one can enable direct sampling mode by soldering a wire to the Q-branch pins of the RTL2832U. The RTL2832U samples at 28.8 MHz, so 0 – 14.4 MHz, and 14.4 MHz – 28.8 MHz can be listened to.

The V3 has direct sampling mode implemented in hardware already, so no hardware mods are required to listen to HF via direct sampling.

To split the HF signal out at the SMA connector, a diplexer tuned to 25 MHz is used. A 10dB buffer preamp sits after the diplexer which helps to boost the signal and overcome losses in the subsequent filter and impedance transformer. After the preamp is a 24 MHz low pass filter and then an impedance matching and single to double ended transformer. The addition of the preamp, filter and transformer ensures good direct sampling performance.

The result is that 500 kHz to about 24 MHz can be received in direct sampling mode.

Direct sampling could be more sensitive than using an upconverter, but dynamic won't be as good as with an upconverter. It can overload easily if you have strong signals since there is no gain control. And you will see aliasing of signals mirrored around 14.4 MHz due to the Nyquist theorm. But direct sampling mode should at least give the majority of users a decent taste of what's on HF. If you then find HF interesting, then you can consider upgrading to an upconverter like the SpyVerter (the SpyVerter is the only upconverter we know of that is compatible with our bias tee for easy operation, other upconverters require external power).

If you search on YouTube for "RTL-SDR V3", you will find several videos showing what you can get in direct sampling mode. Most people are surprised at how good it can be, but also many users will need a broadcast AM filter to reduce overloading. We sell a suitable broadcast AM filter on our store www.rtl-sdr.com/store.

Expansion pads on the PCB

Access pads for the unused GPIO pins, CLK in/out, 3.3V, GND and I2C pins have been added. The CLK input/output is disconnected by default. Access pads for the I branch have also been added as some users and industrial customers are using these in special projects. These pads are only for advanced users who need them for special projects. Take care as these pins are not ESD protected.

Clock selector jumper

By soldering in a 4 pin 1.27mm pitch jumper header and removing the default 0 Ohm resistor, one can now easily select between the onboard clock, an external clock, or having the on board clock be the output for another dongle. This is for advanced users only who want to experiment with things like passive radar, and coherent receivers.

Corner mounting holes for those who want to stack PCBs.

Some customers have been building devices that require multiple RTL-SDR dongles, and these standoff holes should aid in stacking.

Feature Information

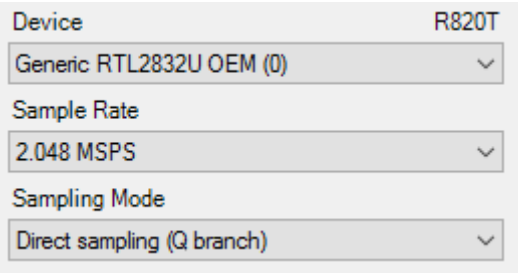
Feature 1: Direct Sampling HF Mode

This feature allows you to listen to HF signals between about 500 kHz to 28.8 MHz.

To use direct sampling mode first connect an appropriate HF antenna to the SMA antenna port (this is the same port where you connect your VHF/UHF antenna).

In SDR# select the Q-branch in the configure menu (the cog icon next to the play button). (If it is greyed out make sure you stop the SDR first, by clicking the stop button in SDR#)

Press Play and tune to 500 kHz – 28.8 MHz.



VHF antennas like small discones or short whip antennas will probably not pick up HF signals very well, if at all. If you have no such antenna you *might* get something with the large telescopic antenna extended to its maximum length of 1.5m, but really this is still not long enough for HF. You can instead use the screw nut provided with the antenna base to clamp on a long wire antenna that is 5 meters or more in length. Ideally you should use a 9:1 unun with the long wire antenna for optimal reception but it is not totally necessary. Even more ideally you'd use an antenna tuner, though this is expensive.

Other software like HDSDR and GQRX can also support direct sampling. It may entail setting a device string, and for the Q-branch, the value should be 2. In GQRX the device string would be "rtl=0,direct_samp=2" (without the quotes). Make sure that there is no space after the comma.

To go back to listening to frequencies above 28.8 MHz remember to change the sampling mode back to "Quadrature Sampling".

Note that this feature makes use of *direct sampling* and so aliasing will occur. The RTL-SDR samples at 28.8 MHz, thus you may see mirrors of strong signals from 0 – 14.4 MHz while tuning to 14.4 – 28.8 MHz and the other way around as well. If these images cause problems, then to remove them you will need to use a low pass filter for 0 – 14.4 MHz, and a high pass filter for 14.4 – 28.8 MHz. Either that or you can simply filter your exact band of interest.

Feature 2: Software Selectable Bias Tee

The V3 RTL-SDR introduces a bias tee which can be enabled easily in software.

WARNING: Before using the bias tee please ensure that you understand that you should not use this option when the dongle is connected *directly* to a DC short circuited antenna. Although the bias tee circuit is dual protected against accidental shorts with a PTC automatically resetting fuse and overcurrent protection on the LDO, short circuiting the bias tee for an extended period (hours) could damage the LDO or fuse permanently. Only use it while connected to an actual powered device, like an LNA, active antenna or the SpyVerter.

To make things clearer: DC Short Antenna -> LNA -> Coax -> V3(bias tee on) is fine. What's not good and makes no sense anyway is DC Short Antenna -> Coax -> V3(bias tee on). DC Short Antenna -> Coax -> V3(bias tee off) is fine.

To enable the bias tee in Windows:

1. Download and extract all the files in the zip file downloadable at https://github.com/rtlsdrblog/rtl-sdr/releases/tag/v1.1 into a folder on your PC. It contains two batch files that can be run.

2. Next make sure that all SDR software like SDR#/HDSDR/SDR-Console etc is fully closed. If there is another program accessing the RTL-SDR the bias tee software will not run.
3. Run the biastee_on.bat file to turn the bias tee on. It will run and open a CMD prompt that will briefly say "Found Rafael Micro R820T Tuner". The CMD prompt will close soon after upon success.

The bias tee is now on. To turn it off repeat steps 2 & 3, but instead run the biastee_off.bat batch file. Alternatively, simply disconnect and then reconnect the SDR to turn the bias tee off.

If you have multiple dongles connected you'll need to edit the batch file to specify what dongle's bias tee you want to activate. Open the bat file with any text editor, like Notepad, and add the dongle selector "-d" flag. For example, to activate the bias tee on the dongle that was plugged in second you'd need to change it to "rtl_biast -b 1 -d 1".

If you get a Smart Screen message, click on More Info, and then on Run Anyway. Also note that some versions of Windows may fail to run batch files due to misconfiguration or aggressive antivirus software. If you cannot fix these problems with Windows or your antivirus, run the command manually on the CMD line.

To run it manually on the CMD line first browse to the directory where the bias tee software is stored using "cd" (e.g. cd C:\SDR\bias_tee_folder), and then run:

ON: rtl_biast -b 1

OFF: rtl_biast -b 0

If needed select a particular RTL-SDR device with the -d flag.

In Linux or MacOS download the source from git, compile it the same way you do the regular RTL-SDR drivers, and then run ./rtl_biast -b 1 to turn the bias tee on and ./rtl_biast -b 0 to turn the bias tee off. The procedure is:

```
git clone https://github.com/rtlsdrblog/rtl_biast
cd rtl_biast
mkdir build
cd build
cmake ..
make
cd src
./rtl_biast -b 1
```

If you want to be able to run the bias tee program from anywhere on the command line you can also run "sudo make install".

If you have trouble running the bias tee use a multimeter to check if there is 4.5V at the SMA port, and that your powered device is actually capable of receiving power. Remember that not all LNA's can accept bias tee power. We recommend Adam 9A4QV's LNA4ALL, as you can order this from his store with the bias tee power option enabled.

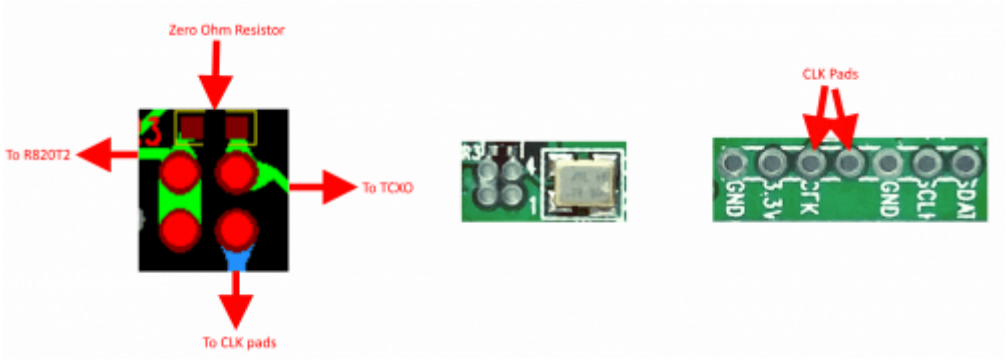
Feature 3: Selectable Clock & Expansion Headers

This is for advanced users who need to daisy chain clocks together for coherent experiments, or need to access other ports. You can either bridge the clock selector the directly with a solder bridge, or solder on a 1.27mm 2×2 header pin jumper.

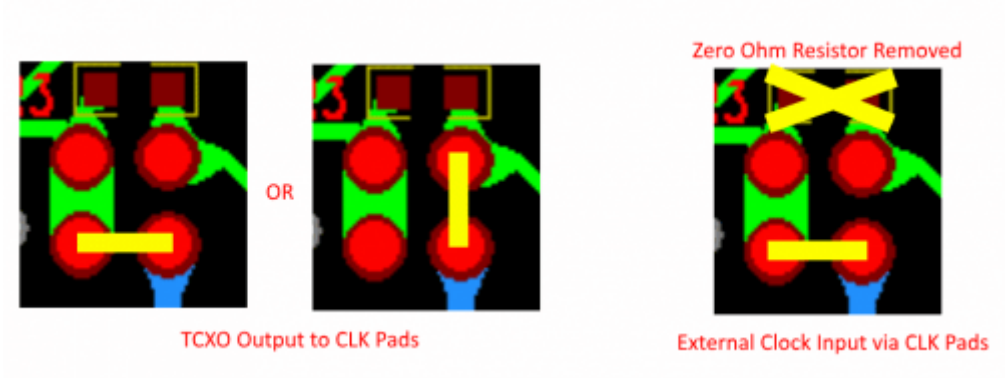
1. To add a jumper to the CLK selector header.
2. Carefully remove the 0 Ohm resistor.
3. Very carefully solder a 1.27mm 2×2 header onto the clock selector pads.

You can now select your clock input.

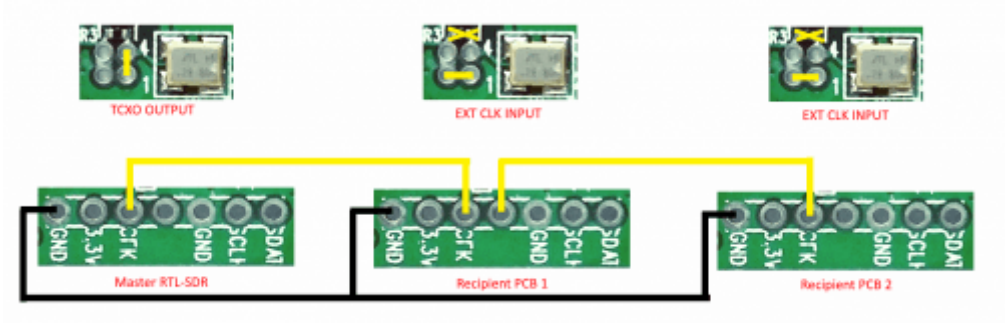
How to connect the CLK jumpers:



The first position allows you to output the dongles clock to the CLK pads. The second position allows you to input an external clock.



An example of CLK daisy chaining is shown below. One dongles TCXO is connected to two other dongles who have disconnected clocks.



Appendix E

HackRF One

Description

HackRF One, from Great Scott Gadgets, is a Software Defined Radio (SDR) peripheral capable of transmission or reception of radio signals from 1 MHz to 6 GHz. It covers many licensed and unlicensed ham radio bands. It is designed to enable test and development of modern and next generation radio technologies. HackRF One is an open source hardware platform that can be used as a USB peripheral or programmed for stand-alone operation.

HackRF One works like a sound card of computer. It processes Digital Signals to Radio waveforms allowing integration of large-scale communication networks. It is designed to test, develop, improvise and modify the contemporary Radio Frequency systems.

HackRF One has an injection molded plastic enclosure and ships with a micro USB cable. An antenna is not included. ANT500 is recommended as a starter antenna for HackRF One.

HackRF One is test equipment for RF systems. It has not been tested for compliance with regulations governing transmission of radio signals. You are responsible for using your HackRF One legally.

Features

•1 MHz to 6 GHz operating frequency

•half-duplex transceiver

•up to 20 million samples per second

•8-bit quadrature samples (8-bit I and 8-bit Q)

•compatible with GNU Radio, SDR#, and more

•software-configurable RX and TX gain and baseband filter

•software-controlled antenna port power (50 mA at 3.3 V)

•Open source hardware

•SMA female antenna connector

•SMA female clock input and output for synchronization

•convenient buttons for programming

•internal pin headers for expansion

•Hi-Speed USB 2.0

•USB-powered

Part List

1 x HackRF One

1 x Micro USB Cable

https://www.seeedstudio.com/HackRF-One-p-2870.html 6-6-17

Appendix F

KerberosSDR (Python)

```
import sys
import os
import time
import numpy as np
from scipy import fft,ifft
from scipy import signal
from scipy.signal import correlate
from PyQt4 import QtGui, QtCore
from pyapril import channelPreparation as cp
from pyapril import clutterCancellation as cc
from pyapril import detector as det
from pyapril.hitProcessor import CA_CFAR
class SignalProcessor(QtCore.QThread):
    signal_spectrum_ready = QtCore.pyqtSignal()
    signal_sync_ready = QtCore.pyqtSignal()
    signal_DOA_ready = QtCore.pyqtSignal()
    signal_overdrive = QtCore.pyqtSignal(int)
    signal_period    = QtCore.pyqtSignal(float)
    signal_PR_ready = QtCore.pyqtSignal()
    def __init__(self, parent=None, module_receiver=None):
        super(SignalProcessor, self).__init__(parent)
        self.module_receiver = module_receiver
        self.en_spectrum = True
        self.en_sync = True
        self.en_sample_offset_sync = False
        self.en_record = False
        self.en_calib_iq = False
        self.en_calib_DOA_90 = False
        self.en_DOA_estimation = False
```

```
        self.en_PR_processing = False
        self.en_PR_autodet = False
        self.en_DOA_Bartlett = False
        self.en_DOA_Capon = False
        self.en_DOA_MEM = False
        self.en_DOA_MUSIC = False
        self.en_DOA_FB_avg = False
        self.DOA_inter_elem_space = 0.5
        self.DOA_ant_alignment = "ULA"
        self.ref_ch_id = 0
        self.surv_ch_id = 1
        self.en_td_filtering = False
        self.td_filter_dimension = 1
        self.max_Doppler = 500  # [Hz]
        self.windowing_mode = 0
        self.max_range = 128  # [range cell]
        self.cfar_win_params = [10,10,4,4] # [Est. win length, Est. win width, Guard win length,
Guard win width]
        self.cfar_threshold = 13
        self.RD_matrix = np.ones((10,10))
        self.hit_matrix = np.ones((10,10))
        self.RD_matrix_last = np.ones((10,10))
        self.RD_matrix_last_2 = np.ones((10,10))
        self.RD_matrix_last_3 = np.ones((10,10)
        self.center_freq = 0  # TODO: Initialize this [Hz]
        self.fs = 1.024 * 10**6  # Decimated sampling frequncy - Update from GUI
        #self.sample_size = 2**15
        self.channel_number = 4
        # Processing parameters
        self.test = None
        self.spectrum_sample_size = 2**14 #2**14
        self.DOA_sample_size = 2**15 # Connect to GUI value??
```

```
            self.xcorr_sample_size = 2**18 #2**18
            self.spectrum = np.ones((self.channel_number+1,self.spectrum_sample_size),
dtype=np.float32)
            self.xcorr = np.ones((self.channel_number-1,self.xcorr_sample_size*2),
dtype=np.complex64)
            self.phasor_win = 2**10 # Phasor plot window
            self.phasors = np.ones((self.channel_number-1, self.phasor_win), dtype=np.complex64)
            self.run_processing = False
            self.delay_log= np.array([[0],[0],[0]])
            self.phase_log= np.array([[0],[0],[0]])
            self.DOA_Bartlett_res = np.ones(181)
            self.DOA_Capon_res = np.ones(181)
            self.DOA_MEM_res = np.ones(181)
            self.DOA_MUSIC_res = np.ones(181)
            self.DOA_theta = np.arange(0,181,1)
            self.lastTime = 0
            self.runningSync = 0
            self.timed_sync = False
            self.noise_checked = False
            self.resync_time = -1
        def run(self):
            self.run_pocessing = True
            while self.run_processing:
                start_time = time.time()
                self.module_receiver.download_iq_samples(
                self.DOA_sample_size = self.module_receiver.iq_samples[0,:].size
                self.xcorr_sample_size = self.module_receiver.iq_samples[0,:].size
                self.xcorr = np.ones((self.channel_number-1,self.xcorr_sample_size*2),
dtype=np.complex64)
                if self.module_receiver.overdrive_detect_flag:
                    self.signal_overdrive.emit(1)
                else:
```

```
                self.signal_overdrive.emit(0)
            # Display spectrum
            if self.en_spectrum:
                self.spectrum[0, :] = np.fft.fftshift(np.fft.fftfreq(self.spectrum_sample_size,
1/self.fs))/10**6
                m = self.channel_number
                #self.spectrum[1:m+1,:] =
10*np.log10(np.fft.fftshift(np.abs(np.fft.fft(self.module_receiver.iq_samples[0:m,
0:self.spectrum_sample_size]))))
            for m in range(self.channel_number):
                    self.spectrum[m+1,:] =
10*np.log10(np.fft.fftshift(np.abs(np.fft.fft(self.module_receiver.iq_samples[m,
0:self.spectrum_sample_size]))))
                self.signal_spectrum_ready.emit()
            if self.en_sync or self.timed_sync:
                self.sample_delay()
                self.signal_sync_ready.emit()
            if self.en_sample_offset_sync:
                self.module_receiver.set_sample_offsets(self.delay_log[:,-1])
                self.en_sample_offset_sync = False
            # IQ calibration request
            if self.en_calib_iq:
                # IQ correction
                for m in range(self.channel_number):
                    self.module_receiver.iq_corrections[m] *=
np.size(self.module_receiver.iq_samples[0, :])/(np.dot(self.module_receiver.iq_samples[m,
:],self.module_receiver.iq_samples[0, :].conj()))
                c = np.sqrt(np.sum(np.abs(self.module_receiver.iq_corrections)**2))
                self.module_receiver.iq_corrections =
np.divide(self.module_receiver.iq_corrections, c)
                #print("Corrections: ",self.module_receiver.iq_corrections)
                self.en_calib_iq = False
```

```
            if self.en_calib_DOA_90:
                x = self.DOA_inter_elem_space * np.cos(2*np.pi/4 * np.arange(4))
                y = self.DOA_inter_elem_space * np.sin(-2*np.pi/4 * np.arange(4)) # For this
specific array only
                ref_vector = de.gen_scanning_vectors(4, x, y, np.zeros(1))[:, 0]
                #ref_vector = np.exp(1j*2*np.pi*0.5*np.cos(np.radians(0-
np.arange(self.channel_number)*(360)/self.channel_number))) # UCA
                N= np.size(self.module_receiver.iq_samples[0, :])
                for m in range(self.channel_number):
                    self.module_receiver.iq_corrections[m] *=
ref_vector[m]*N/(np.dot(self.module_receiver.iq_samples[m,
:],self.module_receiver.iq_samples[0, :].conj()))
                #print("Corrections: ",self.module_receiver.iq_corrections)
                self.en_calib_DOA_90 = False
            if self.en_DOA_estimation:
                # Get FFT for squelch
                self.spectrum[1,:] =
10*np.log10(np.fft.fftshift(np.abs(np.fft.fft(self.module_receiver.iq_samples[0,
0:self.spectrum_sample_size])))


                self.estimate_DOA()
                self.signal_DOA_ready.emit()
            if self.en_record:
                np.save('hydra_samples.npy', self.module_receiver.iq_samples)
            if self.timed_sync and not self.en_sync:
                if not self.noise_checked:
                    self.module_receiver.switch_noise_source(0)
                self.timed_sync = False
                self.en_sample_offset_sync=True
                self.runningSync = 0
            resync_on = True
            if(self.resync_time < 10):
```

```
                resync_on = False
            if(((start_time - self.lastTime) > self.resync_time) and not self.en_sync and
resync_on):
                self.lastTime = start_time
                self.module_receiver.switch_noise_source(1)
                time.sleep(0.1)
                self.runningSync = 1
                self.timed_sync = True
            stop_time = time.time()
            self.signal_period.emit(stop_time - start_time)
        def sample_delay(self):
            N = self.xcorr_sample_size
            iq_samples = self.module_receiver.iq_samples[:, 0:N]
            delays = np.array([[0],[0],[0]])
            phases = np.array([[0],[0],[0]])
            np_zeros = np.zeros(N, dtype=np.complex64)
            x_padd = np.concatenate([iq_samples[0, :], np_zeros])
            x_fft = np.fft.fft(x_padd)
            for m in np.arange(1, self.channel_number):
                y_padd = np.concatenate([np_zeros, iq_samples[m, :]])
                y_fft = np.fft.fft(y_padd)
                self.xcorr[m-1] = np.fft.ifft(x_fft.conj() * y_fft)
                delay = np.argmax(np.abs(self.xcorr[m-1])) - N
                #phase = np.rad2deg(np.angle(self.xcorr[m-1, delay + N]))
                phase = np.rad2deg(np.angle(self.xcorr[m-1, N]))
                #self.phasors[m-1, :] = (iq_samples[0, offset: self.phasor_win+offset] *
iq_samples[m, offset+delay: self.phasor_win+offset+delay].conj())
                #self.phasors[m-1, :] = (iq_samples[0, 0: self.phasor_win] * iq_samples[m, 0:
self.phasor_win].conj())
                self.IQSamples[1, :] = np.roll(self.IQSamples[1, :], delay * -1)
                if delay > 0:
                    self.IQSamples[1, -delay::] = np.zeros(delay, dtype=np.complex64)
```

```
            if delay < 0:
                self.IQSamples[1, 0: np.abs(delay)] = np.zeros(np.abs(delay),
dtype=np.complex64)
            #msg = "[ INFO ] delay: " + str(delay)
            #print(msg)
            delays[m-1,0] = delay
            phases[m-1,0] = phase
        self.delay_log = np.concatenate((self.delay_log, delays),axis=1)
        self.phase_log = np.concatenate((self.phase_log, phases),axis=1)
    def delete_sync_history(self):
        self.delay_log= np.array([[0],[0],[0]])
        self.phase_log= np.array([[0],[0],[0]])


    def estimate_DOA(self):
        #print("[ INFO ] Python DSP: Estimating DOA")

        iq_samples = self.module_receiver.iq_samples[:, 0:self.DOA_sample_size]
        # Calculating spatial correlation matrix
        R = de.corr_matrix_estimate(iq_samples.T, imp="fast")

        if self.en_DOA_FB_avg:
            R=de.forward_backward_avg(R)

        M = np.size(iq_samples, 0)

        if self.DOA_ant_alignment == "UCA":
            self.DOA_theta =  np.linspace(0,360,361)
            #scanning_vectors = de.gen_uca_scanning_vectors(M, self.DOA_inter_elem_space,
self.DOA_theta)
            x = self.DOA_inter_elem_space * np.cos(2*np.pi/M * np.arange(M))
```

```
            y = self.DOA_inter_elem_space * np.sin(-2*np.pi/M * np.arange(M)) # For this
specific array only
            scanning_vectors = de.gen_scanning_vectors(M, x, y, self.DOA_theta)

             # DOA estimation
            if self.en_DOA_Bartlett:
               self.DOA_Bartlett_res = de.DOA_Bartlett(R, scanning_vectors)
            if self.en_DOA_Capon:
               self.DOA_Capon_res = de.DOA_Capon(R, scanning_vectors)
            if self.en_DOA_MEM:
               self.DOA_MEM_res = de.DOA_MEM(R, scanning_vectors,  column_select = 0)
            if self.en_DOA_MUSIC:
               self.DOA_MUSIC_res = de.DOA_MUSIC(R, scanning_vectors, signal_dimension
= 1)

         elif self.DOA_ant_alignment == "ULA":
            self.DOA_theta =  np.linspace(-90,90,181)
            x = np.zeros(M)
            y = np.arange(M) * self.DOA_inter_elem_space
            scanning_vectors = de.gen_scanning_vectors(M, x, y, self.DOA_theta)
            # DOA estimation
            if self.en_DOA_Bartlett:
               self.DOA_Bartlett_res = de.DOA_Bartlett(R, scanning_vectors)
            if self.en_DOA_Capon:
               self.DOA_Capon_res = de.DOA_Capon(R, scanning_vectors)
            if self.en_DOA_MEM:
               self.DOA_MEM_res = de.DOA_MEM(R, scanning_vectors,  column_select = 0)
            if self.en_DOA_MUSIC:
               self.DOA_MUSIC_res = de.DOA_MUSIC(R, scanning_vectors, signal_dimension
= 1)

         #print(self.DOA_MUSIC_res)
```

الحقيقي لإشارات الراديو في نطاق التردد من25 ميجاهرتز إلى 1.75 جيجا هرتز ، والتتبع الفعال والمراقبة الدقيقة لمصادر الراديو.

يبدأ عمل النظام من مركز العمليات الذي يعرض جميع جولات التتبع الخاصة بمصدر راديو معين عن طريق إرسال محطات تتبع التردد اللاسلكي مجموعة من البيانات لجولة التتبع عبر الانترنت. بالإضافة إلى ذلك، يحتوي مركز العمليات على محلل طيف مرتبط بواجهة المستخدم الرسومية لمركز التشغيل لتتبع المصدر اللاسلكية. يتم تحديث البيانات باستمرار مع مركز العمليات لمزامنتها مع جولات التتبع الأخيرة لمصادر الانبعاث المحددة. يقوم مشغل محطة تتبع التردد الراديوي بضبط معلمات عملية التعقب لتحديد موقع المصادر باستخدام التثليث بناءً على جولات التتبع وبيانات التتبع من مواقع مختلفة.

تظهر نتائج المحاكاة أن أفضل خوارزميات التتبع هي ROOT-MUSIC وROOT-WSF مع مصفوفة الخطية وROOT-MUSIC مع المصفوفة الدائرية. تم أيضًا اختبار نظام (RFST) بتشكيلات مختلفة تبدأ من عقدة محطة واحدة إلى أربع محطات. أظهرت النتائج أن استخدام أربع محطات يوفر أفضل تقدير موقع مصادر الانبعاث الراديوية مع مسافة خطأ تصل إلى 2متر باستخدام خوارزمية في ROOT-WSF. اما بالنسبة لنتائج العملية، تكون مسافات الخطأ الدنيا 4.23 متر و4 متر عند استخدام خوارزمية تتبع الموسيقى مع ULA وUCA. تظهر مقارنة المحاكاة والنتائج العملية أن النظام المقترح قد تم بناؤه بدقة على كلا الجانبين.

الخلاصـــــــــــة

يعد نظام تتبع إشارات الترددات الراديوية (RFST) أحد التقنيات الأساسية التي تعتبر أداة لمراقبة وتعقب إشارات الترددات اللاسلكية. لذلك، تم اقتراح نظام (RFST) الذي يتم من خلاله مراقبة استخدام طيف التردد اللاسلكي لفحوصات البيئة الراديوية للبحث عن مصادر انبعاث الراديو غير المصرح بها. ان نظام المقترح القائم على تقنية اتجاه الوصول مكون من شبكة من الأجهزة المتصلة وأجهزة الكمبيوتر ومحطات تتبع الترددات الراديوية. حيث ان محطات تتبع الترددات الراديوية هي عبارة عن مركبات تحمل معدات تعقب الإشارات ومتصلة بمركز العمليات الذي يتم خلاله مراقبة الطيف الترددي، وتتبع ومقاطعة زوايا واتجاهات مصادر الانبعاث الراديوية وحساب موقعها باستخدام عملية التثليث.

تتناول هذه الرسالة قسمين رئيسيين: الأول هو محاكاة عمل النظام المقترح، والثاني هو التطبيق العملي للنظام من مكونات الأجهزة والبرمجيات للنظام المقترح. يقدم جزء المحاكاة الذي ينفذه برنامج Matlab طرقًا مختلفة مثل محاكاة عمل سبع خوارزميات: Beamscan، MVDR، MUSIC، ESPRIT، Beamspace-ESPRIT ROOT-MUSIC و ROOT-WSF. تتبع هذه الخوارزميات زوايا مصادر الانبعاث الراديوية باستخدام مصفوفات الهوائيات للحصول على عملية التعقب بأفضل دقة.

يتم اختبار ثلاثة أشكال من مصفوفات الهوائيات والتي هي خطية ومربعة ودائرية تعمل مع خوارزميات التعقب لاستخراج أفضل دقة لاتجاه وصول (DOA) من محطة تتبع الترددات الراديوية. علاوة على ذلك، تم تصميم نظام (RFST) بتقنيات مختلفة من محطات تتبع الترددات الراديوية الموزعة جغرافيًا لاستخراج إحداثيات المصادر اللاسلكية التي تعمل عند 990 ميجاهرتز.

يشمل الجانب العملي مكونات الأجهزة والبرامج المحمولة على السيارة أو الموجودة في مركز العمليات. تحتوي السيارة على حاسوب صغير (Raspberry Pi3) مع جهازي راديو معرف بالبرمجيات (SDR). يعمل أول (SDR) بمصفوفة مكونة من أربعة هوائيات ويستخدم لتتبع زوايا المصادر اللاسلكية والثاني لمراقبة الطيف. يحتوي مركز العمليات على محلل طيف (SDR) وبرنامج واجهة المستخدم الرسومية لمركز العمليات المصمم بواسطة MATLAB، المرتبط بالانترنت. تعمل هذه الأجزاء معًا لتحقيق الأهداف الأساسية للنظام ، بما في ذلك المراقبة الدورية أو الدائمة في الوقت

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل / كلية الهندسة
قسم الهندسة الكهربائية

تنفيذ منظومة تعقب الترددات الراديوية

بالاعتماد على تقنية اتجاه الوصول

اطروحة

مقدمة إلى كلية الهندسة في جامعة بابل

وهي جزء من متطلبات الحصول على درجة الدكتوراه

فلسفة في الهندسة/ الهندسة الكهربائية/ الالكترونيك والاتصالات

من قبل

علي نجم عبدالله سبتي

بأشراف

الأستاذ الدكتور

ليث علي عبد الرحيم

1443 هـ **2021 م**