ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδίαση και ανάπτυξη ενός συστήματος παρακολούθησης του φάσματος με χρήση SDR συσκευών.

Διπλωματική Εργασία

ΚΛΕΙΤΣΟΓΙΑΝΝΗΣ ΒΗΣΣΑΡΙΩΝ

Επιβλέπων: ΚΟΡΑΚΗΣ ΑΘΑΝΑΣΙΟΣ

ΙΟΥΛΙΟΣ 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

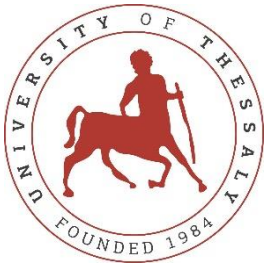
Σχεδίαση και ανάπτυξη ενός συστήματος παρακολούθησης του φάσματος με χρήση SDR συσκευών.

Διπλωματική Εργασία

ΚΛΕΙΤΣΟΓΙΑΝΝΗΣ ΒΗΣΣΑΡΙΩΝ

Επιβλέπων: ΚΟΡΑΚΗΣ ΑΘΑΝΑΣΙΟΣ

ΙΟΥΛΙΟΣ 2025



UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Design and implementation of spectrum sensing software using SDR devices.

Diploma Thesis

KLEITSOGIANNIS VISSARION

Supervisor: KORAKIS ATHANASIOS

JULY 2025

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων **Κοράκης Αθανάσιος**
Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μέλος **Αργυρίου Αντώνιος**
Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μέλος **Φλέγκας Παρίσης**
Επίκουρος Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελούν αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλουν οποιασδήποτε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχουν έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Δηλώνω επίσης ότι τα αποτελέσματα της εργασίας δεν έχουν χρησιμοποιηθεί για την απόκτηση άλλου πτυχίου. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο Δηλών

Κλειτσογιάννης Βησσαρίων

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism.

The Declarant

Kleitsogiannis Vissarion

Διπλωματική Εργασία

Σχεδίαση και ανάπτυξη ενός συστήματος παρακολούθησης του φάσματος με χρήση SDR συσκευών.

ΚΛΕΙΤΣΟΓΙΑΝΝΗΣ ΒΗΣΣΑΡΙΩΝ

Περίληψη

Αυτή η διπλωματική εργασία πραγματεύεται την σχεδίαση και την ανάπτυξη ενός συστήματος παρακολούθησης του φάσματος με ανίχνευση ενέργειας με τη χρήση μιας SDR συσκευής και πιο συγκεκριμένα του Pluto SDR. Η φασματική ανάλυση είναι μια διαδικασία πολύ σημαντική στα cognitive radio networks, καθώς επιτρέπει να εντοπίσουμε τις μπάντες συχνοτήτων οι οποίες δεν χρησιμοποιούνται από κάποιο άλλο χρήστη και έτσι να κάνουμε εκεί τη μετάδοση μας χωρίς να έχουμε ξένες παρεμβολές. Επικεντρωνόμαστε στην ανάπτυξη λογισμικού το οποίο σε συνδυασμό με το Pluto SDR θα μπορεί να μας πληροφορήσει για την ύπαρξη ή μη σημάτων στην περιοχή συχνοτήτων που θα του δοθεί ως είσοδος μετρώντας τα επίπεδα ενέργειας των σημάτων που έχουμε λάβει. Γίνεται αναλυτική παρουσίαση όλων των εννοιών που θα μας απασχολήσουν ώστε να φτάσουμε στον τελικό στόχο μας. Τα αποτελέσματα δείχνουν ότι το λογισμικό που έχουμε αναπτύξει μπορεί σίγουρα να ανιχνεύει με επιτυχία σήματα στην περιοχή συχνοτήτων που θα του ζητηθεί σύμφωνα πάντα με τις δυνατότητες και τους περιορισμούς που εισάγει το Pluto SDR.

Diploma Thesis

Design and implementation of spectrum sensing software using SDR devices.

KLEITSOGIANNIS VISSARION

Abstract

This thesis deals with the design and development of a spectrum monitoring system with energy detection using an SDR device and more specifically the Pluto SDR. Spectrum analysis is a very important process in cognitive radio networks, as it allows us to identify frequency bands that are not used by any other user and thus make our transmission there without having foreign interference. We are focusing on the development of software which, in combination with the Pluto SDR, will be able to inform us about the presence or absence of signals in the frequency range that will be given to it as input by measuring the energy levels of the signals we have received. There is a detailed presentation of all the concepts that will concern us so that we can reach our final goal. The results show that the software we have developed can certainly successfully detect signals in the frequency range that will be requested from user according to the limitations introduced by Pluto SDR.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Συντομογραφίες

EETT	Εθνική Επιτροπή Τηλεπικοινωνιών και Ταχυδρομείων
ADC	Analog To Digital Converter
dBs	Decibels
DDC	Digital Downconvercion
EHF	Extremely High Frequency
FFT	Fast Fourier Transform
Fs	Sampling frequency
IEEE	Institute of Electrical and Electronics Engineers
IF	Intermediate Frequency
PLL	Phase-Locked Loop
RF	Radio Frequency
SDR	Software Defined Radios
SHF	Super High Frequency
SNR	Signal To Noise Ratio
UHF	Ultra High Frequency
VCO	Voltage Controlled Oscillator

Κεφάλαιο 1 Εισαγωγή

Οι ασύρματες επικοινωνίες αποτελούν ένα πολύ σημαντικό και αναπόσπαστο κομμάτι της καθημερινότητας μας. Όλοι μας καθημερινά χρησιμοποιούμε τις δυνατότητες και τις ευκολίες που μας προσφέρουν οι ασύρματες επικοινωνίες. Προσφέρουν πολλούς τρόπους επικοινωνίας μεταξύ ανθρώπων, συσκευών και υπηρεσιών. Χρησιμοποιούν ραδιοκύματα με σκοπό τη μετάδοση της πληροφορίας χωρίς να χρησιμοποιούμε καλώδια. Κινητή τηλεφωνία, δορυφορικές επικοινωνίες, Wi-Fi είναι μόνο μερικές μορφές ασύρματων τηλεπικοινωνιών. Το κανάλι όμως της επικοινωνίας αυτών των τεχνολογιών είναι αρκετά ευαίσθητο και υπάρχουν αρκετά προβλήματα τα οποία πολλές φορές επηρεάζουν την ποιότητα και την αξιοπιστία της επικοινωνίας. Ξένες παρεμβολές, θόρυβος, πολλαπλές οδεύσεις, πολλαπλές μεταδόσεις στο ίδιο κανάλι, hidden terminal και exposed terminal είναι μερικά από τα προβλήματα που υπάρχουν στις ασύρματες τεχνολογίες.

1.1 Αντικείμενο της διπλωματικής

Ένας πολύ σημαντικός παράγοντας για να έχουμε μια καλή και αξιόπιστη επικοινωνία είναι το κανάλι μας. Αναφέραμε παραπάνω ότι στις ασύρματες επικοινωνίες το κανάλι μας είναι ο αέρας και για να μεταδώσουμε κάτι χρησιμοποιούμε τα ραδιοκύματα δηλαδή κάποιες συχνότητες. Υπάρχουν ελεύθερες συχνότητες και συχνότητες τις οποίες για να τις χρησιμοποιήσει κανείς θα πρέπει να πληρώσει. Στις ελεύθερες συχνότητες μπορεί να μεταδώσει ή να λάβει πληροφορία ο οποιοσδήποτε. Καταλαβαίνει εύκολα κανείς ότι υπάρχει πλήθος μεταδόσεων. Επίσης πολύ σημαντικό είναι να κάνουμε τη μετάδοση μας σε μια περιοχή συχνοτήτων όπου δεν υπάρχει κάποια άλλη μετάδοση. Πολύ απλά γιατί αν υπάρχει κι άλλη ή άλλες μεταδόσεις στο κανάλι μου τότε η ποιότητα της

επικοινωνίας μας θα είναι πολύ κακή. Η πληροφορία μας θα αλλοιωθεί και θα αναγκαστούμε να δοκιμάσουμε ξανά. Ο σκοπός λοιπόν αυτής της εργασίας ήταν να φτιάξουμε ένα λογισμικό το οποίο σε συνεργασία με μια SDR συσκευή θα είναι ικανό να κάνει ανίχνευση στο φάσμα των συχνοτήτων για πιθανές μεταδόσεις εκείνη τη στιγμή ούτως ώστε να αποφύγουμε την ταυτόχρονη χρήση του καναλιού. Έτσι μπορούμε να επιλέξουμε για τις μεταδόσεις μας μία περιοχή συχνοτήτων η οποία δεν χρησιμοποιείται από κάποιον άλλο και να έχουμε μια επικοινωνία καλής ποιότητας και αξιόπιστη.

Ρυθμίζοντας κατάλληλα την SDR συσκευή στην επιθυμητή κεντρική συχνότητα αυτή θα μας επιστρέψει μιγαδικά δείγματα στο πεδίο του χρόνου σε μία περιοχή συχνοτήτων γύρω από τη συγκεκριμένη κεντρική συχνότητα. Μέσω μετασχηματισμού Fourier θα πάρουμε δείγματα στο πεδίο της συχνότητας. Εκεί θα υπολογίσουμε την ισχύ και την ενέργεια των δειγμάτων αυτών και θα τα μετατρέψουμε σε λογαριθμική κλίμακα. Τα αποτελέσματα τα εμφανίζουμε σε μία γραφική παράσταση από την οποία μπορεί κανείς να συμπεράνει αν στο εύρος συχνοτήτων που τον ενδιαφέρει υπάρχει κάποια μετάδοση σε εξέλιξη ή όχι. Με αυτό τον τρόπο μπορούμε να αποφύγουμε σε μεγάλο βαθμό ταυτόχρονες μεταδόσεις στις ίδιες συχνότητες καθώς και να επιλέξουμε ένα ελεύθερο κανάλι για να μεταδώσουμε ή να λάβουμε πληροφορία.

1.2 Οργάνωση του τόμου

Στο κεφάλαιο 1 γίνεται μια εισαγωγή στις ασύρματες επικοινωνίες, τα προβλήματα που αντιμετωπίζει ο χώρος και γίνεται αναφορά στο αντικείμενο αυτής της εργασίας.

Στο κεφάλαιο 2 θα γνωρίσουμε τα ηλεκτρομαγνητικά κύματα, το ηλεκτρομαγνητικό φάσμα συχνοτήτων και θα μιλήσουμε και για τη σωστή διαχείριση του.

Στο κεφάλαιο 3 παρουσιάζονται τα Software Defined Radios (SDR), σε συνεργασία με τα οποία έχουμε φτιάξει το λογισμικό μας για την ανάλυση του φάσματος συχνοτήτων μέσω ανίχνευσης ενέργειας.

Στο κεφάλαιο 4 μιλάμε για τα σήματα, τη δειγματοληψία, τη διαμόρφωση και αποδιαμόρφωση σημάτων.

Στο κεφάλαιο 5 παρουσιάζονται οι έννοιες των πεδίων του χρόνου – συχνότητας και στη συνέχεια μιλάμε για τον μετασχηματισμό Fourier.

Στο κεφάλαιο 6 και αφού έχουμε εξηγήσει στα προηγούμενα κεφάλαια τις έννοιες που απαιτούνταν, παρουσιάζεται το λογισμικό που αναπτύξαμε για την ανάλυση του φάσματος.

Τέλος, στο κεφάλαιο 7 παρουσιάζονται συγκεντρωτικά τα κυριότερα αποτελέσματα της παρούσας εργασίας, η συμβολή της στη διερεύνηση του θέματος και πιθανές προτάσεις για περαιτέρω διερεύνηση του θέματος στο μέλλον.

ΚΕΦΑΛΑΙΟ 2 Το ηλεκτρομαγνητικό φάσμα

2.1 Εισαγωγή

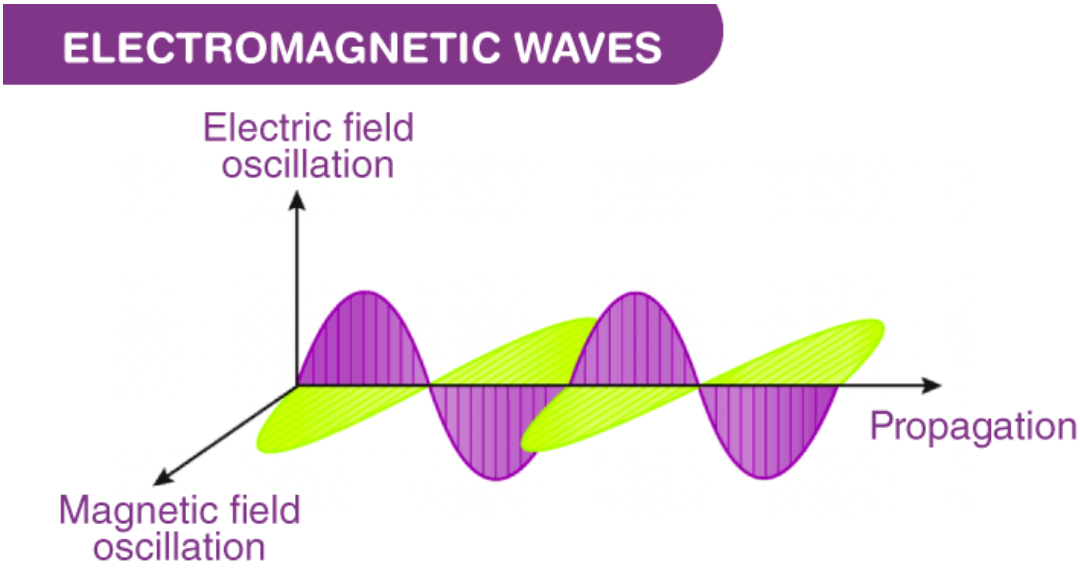
Οι ασύρματες επικοινωνίες έχουν φέρει επανάσταση στον τρόπο με τον οποίο ανταλλάσουμε καθημερινά πληροφορίες. Κινητά τηλέφωνα, ηλεκτρονικοί υπολογιστές, έξυπνες συσκευές, Wi-Fi, ραδιοεπικοινωνίες, δορυφορικές επικοινωνίες και άλλες ακόμα μορφές βασίζονται σε ηλεκτρομαγνητικά κύματα για να λάβουν και να στείλουν δεδομένα. Η πληροφορία που θέλουμε να μεταδώσουμε μέσω κατάλληλης μετατροπής πχ με μια βιντεοκάμερα ή ένα μικρόφωνο γίνεται ηλεκτρικό σήμα. Το ηλεκτρικό αυτό σήμα μετατρέπεται στην κεραία του πομπού σε ένα ηλεκτρομαγνητικό κύμα το οποίο θα ταξιδέψει ως την κεραία του δέκτη με την ταχύτητα του φωτός. Εκεί θα πάρει τη μορφή ηλεκτρικού σήματος και στη συνέχεια θα μετατραπεί σε εικόνα ή ήχο αντιστρόφως ανάλογα με πριν. Αυτά τα ηλεκτρομαγνητικά κύματα αποτελούν τη βάση των σύγχρονων ασύρματων τηλεπικοινωνιών. Σε αυτό το κεφάλαιο θα αρχίσουμε να εξηγούμε κάποιες από τις βασικές έννοιες γύρω από τις οποίες περιστρέφεται η παρούσα εργασία.

2.2 Ηλεκτρομαγνητικά κύματα

Τι είναι τα ηλεκτρομαγνητικά κύματα;

Τα ηλεκτρομαγνητικά κύματα είναι συγχρονισμένα ηλεκτρικά και μαγνητικά πεδία τα οποία ταλαντώνονται σε κάθετα επίπεδα μεταξύ τους και κάθετα προς τη διεύθυνση διάδοσης. Στο κενό διαδίδονται με την ταχύτητα του φωτός (c = 3* 10^8 m/s). Σε όλα τα υπόλοιπα υλικά διαδίδονται με μικρότερες ταχύτητες. Χαρακτηριστικά μεγέθη ενός ηλεκτρομαγνητικού κύματος είναι η συχνότητα, η περίοδος και το μήκος κύματος.

Στο παρακάτω σχήμα βλέπουμε ένα ηλεκτρομαγνητικό κύμα :



Εικόνα 1: Ηλεκτρομαγνητικό κύμα

Η συχνότητα του ηλεκτρομαγνητικού κύματος είναι ο αριθμός των κύκλων ή ταλαντώσεων που κάνει το κύμα σε χρόνο ίσο με ένα δευτερόλεπτο. Συμβολίζεται με f και μετριέται σε Hertz (Hz). Η ονομασία προέρχεται από το Γερμανό φυσικό Χάινριχ Χέρτζ. Συχνότητα ενός Hz ισοδυναμεί με μία ταλάντωση ανά δευτερόλεπτο.

Η περίοδος του ηλεκτρομαγνητικού κύματος είναι ο ελάχιστος χρόνος που απαιτείται για μια πλήρη ταλάντωση (ή ένα κύκλο) του ηλεκτρομαγνητικού κύματος. Είναι το αντίστροφο της συχνότητας και συμβολίζεται με T (T=1/f). Μετριέται σε δευτερόλεπτα.

Μήκος κύματος είναι η απόσταση που διανύει διαδιδόμενο το κύμα στο χρονικό διάστημα μιας περιόδου T του ηλεκτρομαγνητικού κύματος. Συμβολίζεται με λ και μετριέται σε μέτρα (m).

Θεμελιώδης εξίσωση της κυματικής

Μέσα σε κάποιο υλικό η ταχύτητα της ηλεκτρομαγνητικής ακτινοβολίας στο μέσο αυτό ισούται με το γινόμενο του μήκους κύματος στο συγκεκριμένο μέσο και της συχνότητας της ακτινοβολίας αυτής. Σε διαφορετικό υλικό αλλάζει το μήκος κύματος της ακτινοβολίας, η συχνότητα μένει ίδια.
Αν το υλικό μας είναι το κενό τότε έχουμε:
$c = \lambda * f \Rightarrow \lambda = c / f$
Από εδώ συμπεραίνουμε ότι το μήκος κύματος και η συχνότητα είναι αντιστρόφως ανάλογα μεγέθη, αυτό σημαίνει ότι όταν αυξάνει η συχνότητα το μήκος κύματος θα μειώνεται και αντίστροφα.

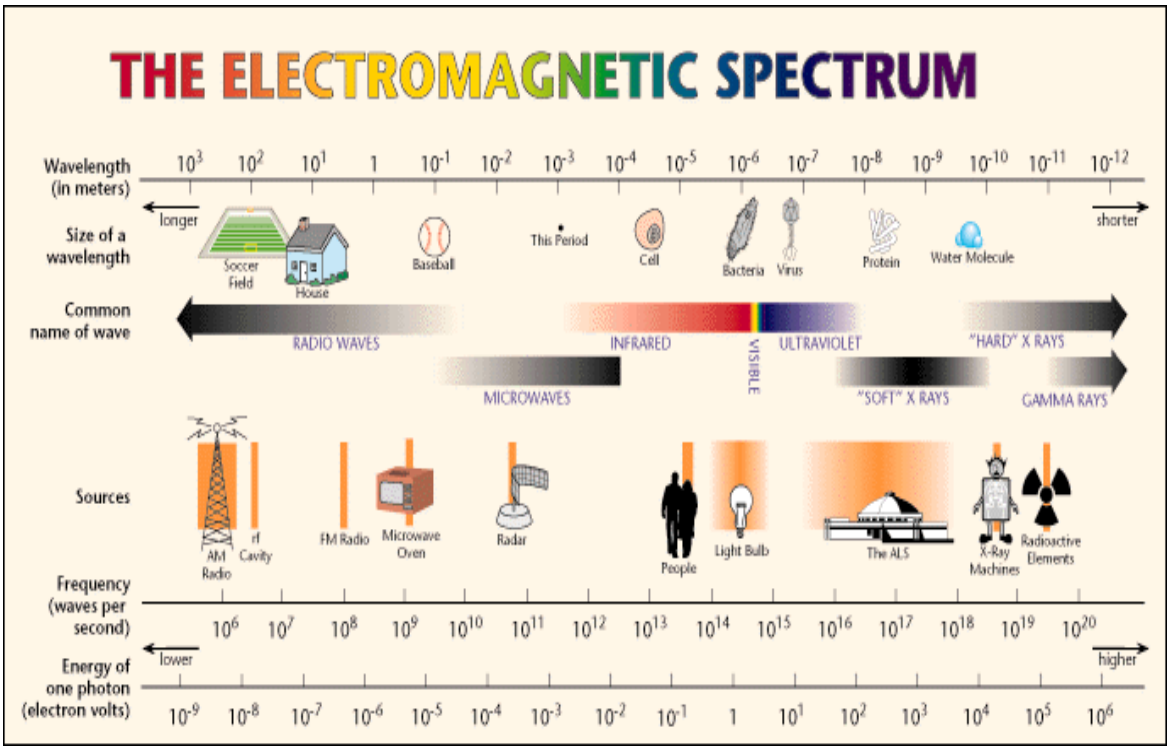
2.3 Το Ηλεκτρομαγνητικό φάσμα

Τι είναι η ηλεκτρομαγνητική ακτινοβολία;

Ηλεκτρομαγνητική ακτινοβολία είναι η εκπομπή στο χώρο ηλεκτρομαγνητικής ενέργειας υπό τη μορφή ηλεκτρομαγνητικών κυμάτων.

Ηλεκτρομαγνητικό φάσμα

Το ηλεκτρομαγνητικό φάσμα είναι όλη η περιοχή συχνοτήτων που καλύπτουν τα ηλεκτρομαγνητικά κύματα. Χωρίζεται σε περιοχές βάσει των συχνοτήτων ή ισοδύναμα των μηκών κύματος τους , καθεμιά έχει τα δικά της χαρακτηριστικά και εφαρμογές. Τέτοιες περιοχές είναι οι ακτίνες γ, οι ακτίνες Χ, η υπεριώδης ακτινοβολία, το ορατό φως, η υπέρυθρη ακτινοβολία, τα μικροκύματα και τα ραδιοκύματα ξεκινώντας από αυτή με το μικρότερο μήκος κύματος και καταλήγοντας σε αυτή με το μεγαλύτερο. Στην εικόνα 2 φαίνονται οι συχνότητες και τα μήκη κύματος των διάφορων κατηγοριών των ηλεκτρομαγνητικών κυμάτων.



Εικόνα 2: Το ηλεκτρομαγνητικό φάσμα

Πάμε να δούμε λίγα πράγματα για την κάθε κατηγορία ξεχωριστά..

Ραδιοκύματα – Μικροκύματα

Είναι τύποι ακτινοβολίας που ταξιδεύουν εύκολα στην ατμόσφαιρα και τα σύννεφα. Παράγονται μέσω της επιτάχυνσης των ηλεκτρονίων στις κεραίες.

Τα ραδιοκύματα είναι ηλεκτρομαγνητικά κύματα με συχνότητες από 3kHz έως και 300GHz (μήκη κύματος 1 χιλιοστό έως 100 χιλιόμετρα). Έχουν μήκη κύματος μεγαλύτερα από το ορατό φως. Χρησιμοποιούνται πολύ στην ραδιοφωνία, την τηλεόραση, τις τηλεπικοινωνίες, τα ραντάρ.

Τα μικροκύματα θεωρούνται ένα κομμάτι των ραδιοκυμάτων καλύπτουν συχνότητες από 300 MHz έως 300GHz (μήκη κύματος από 1 χιλιοστό έως 30 εκατοστά). Χωρίζονται σε τρεις υποκατηγορίες:

Ultra High Frequency : 0.3 -3 GHz (UHF)

Super High Frequency : 3 – 30 GHz (SHF)

Extremely High Frequency : 30 – 300 GHz (EHF)

Χρησιμοποιούνται σε φούρνους μικροκυμάτων, ραντάρ, δορυφορικές επικοινωνίες.

Υπέρυθρες ακτινοβολίες

Καλύπτουν συχνότητες από 300 GHz έως και 400 THz (μήκη κύματος από 750 nm έως και 1 mm). Έχουν μήκη κύματος μεγαλύτερα από το ορατό φως και μικρότερα από τα μικροκύματα. Είναι αόρατες στο μάτι μπορούμε όμως να τις νιώσουμε στο δέρμα μας σαν ζέστη. Μπορούν να γίνουν αντιληπτές από ζώα όπως πχ τα σκυλιά και τεχνητά με κάμερες. Μερικές υπέρυθρες ακτινοβολίες έχουν την ικανότητα να ταξιδεύουν μεγάλες αποστάσεις στην ατμόσφαιρα ενώ άλλες απορροφώνται σε μερικά μόνο μέτρα. Χρησιμοποιούνται στα τηλεχειριστήρια, ενώ παλιότερα για μεταφορά δεδομένων μεταξύ κινητών τηλεφώνων.

Ορατή ακτινοβολία

Είναι η ηλεκτρομαγνητική ακτινοβολία που ανιχνεύεται από το ανθρώπινο μάτι. Τα μάτια μας έχουν την ικανότητα να βλέπουν ηλεκτρομαγνητικά κύματα με μήκος κύματος 380 nm με 750 nm το οποίο σε συχνότητες μεταφράζεται σε ένα εύρος από 400 THz έως 790 THz. Βέβαια αυτό αλλάζει από άνθρωπο σε άνθρωπο. Εκπέμπεται από τον ήλιο και τις λυχνίες πυρακτώσεως.

Υπεριώδης ακτινοβολία

Επίσης εκπέμπεται από τον ήλιο και από άλλες πηγές υψηλής ενέργειας και θερμοκρασίας. Το μεγαλύτερο μέρος της απορροφάται από τα ανώτερα τμήματα της ατμόσφαιρας. Δεν μπορούμε να δούμε αυτή τη μορφή ακτινοβολίας με τα μάτια μας Μπορεί να εισχωρήσει σε μικρές αποστάσεις μέσα σε στερεά αντικείμενα. Γι αυτό προκαλεί εγκαύματα στο δέρμα μας.

Ακτίνες Χ

Έχουν μήκη κύματος από 10 nm έως και 10 pm, αυτό σε συχνότητες αντιστοιχεί σε ένα εύρος από 30 PHz έως 30 EHz.

Εκπέμπονται από πηγές πολύ υψηλής ενέργειας όπως supernova, ραδιενεργά υλικά ή μηχανήματα ακτίνων Χ. Δεν επηρεάζονται από την ατμόσφαιρα επίσης έχουν την ικανότητα να εισχωρούν σε στερεά σώματα, αντικείμενα κάτι το οποίο τα κάνει πολύ χρήσιμα στην καταπολέμηση του λαθρεμπορίου, στη βιομηχανία και στην ιατρική. Ωστόσο λόγω του ότι μπορούν να προκαλέσουν βλάβη στους οργανισμούς πρέπει να χρησιμοποιούνται με σύνεση και προσοχή.

Ακτίνες γ

Είναι οι ακτίνες με τη μεγαλύτερη συχνότητα με μήκος κύματος από 10^-10 m έως και 10^-14 m.

Προκύπτουν από πυρηνικές αντιδράσεις, όπως η διάσπαση ραδιενεργών πυρήνων ή στοιχειωδών σωματιδίων. Έχουν την ικανότητα να εισχωρούν σε στερεά σώματα, αντικείμενα σε σπουδαίες αποστάσεις. Είναι πολύ επικίνδυνες, μεταλλάσσουν το DNA και προκαλούν θάνατο στους οργανισμούς που εκτίθενται σε αυτές.[1,2]

2.4 Η διαχείριση του ηλεκτρομαγνητικού φάσματος και η σημασία της

Το ηλεκτρομαγνητικό φάσμα είναι αγαθό περιορισμένης φύσεως και πρέπει να διαχειρίζεται σωστά καθώς οι ανάγκες ολοένα και αυξάνονται. Η διαχείριση του είναι πολύ σημαντική και αφορά στο διαμοιρασμό των συχνοτήτων με τέτοιο τρόπο ώστε να πετύχουμε την καλύτερη δυνατή χρήση τους ελαχιστοποιώντας τις όποιες παρεμβολές μεταξύ των διαφορετικών υπηρεσιών που το χρησιμοποιούν. Η διεθνής ένωση τηλεπικοινωνιών (ITU) είναι αυτή που θέτει τους γενικούς κανόνες που διέπουν τη χρήση του φάσματος κι από εκεί και πέρα η κάθε χώρα είναι υποχρεωμένη να θεσπίσει νόμους για τη σωστή και αποδοτική διαχείρισή του. Κύριος σκοπός της διαχείρισης αυτής είναι να αποφευχθούν παρεμβολές που μπορούν να οδηγήσουν σε υποβάθμιση της ποιότητας υπηρεσιών ραδιοεπικοινωνίας ή ακόμα και σε διακοπή τους. Επίσης σκοπός είναι και η εκχώρηση των ραδιοσυχνοτήτων σύμφωνα με καθορισμένες προϋποθέσεις για σωστή χρήση του φάσματος.

Για να γίνουμε πιο συγκεκριμένοι θα δούμε πως μοιράζεται στην πράξη το φάσμα των συχνοτήτων. Θα αναφέρουμε μερικές ζώνες συχνοτήτων και τι μεταδίδεται εκεί.

Σε συχνότητες μικρότερες των 100 MHz έχουμε Cb radios, pagers και αναλογικά ασύρματα τηλέφωνα.

Σε συχνότητες μεταξύ 100 MHz και 800 MHz έχουμε broadcast μεταδόσεις (τηλεφωνία, ραδιοφωνία).
Σε συχνότητες μεταξύ 800 MHz και 1000 MHz έχουμε επικοινωνίες έκτακτης ανάγκης και κυψελωτά συστήματα.
Στα 1.8 GHz με 2 GHz έχουμε την κύρια μπάντα συχνοτήτων για τα κυψελωτά συστήματα και τα ασύρματα.
Στα 2.4 GHz με 2.5 GHz έχουμε ασύρματη τηλεφωνία, Wi-Fi, φούρνους μικροκυμάτων, Bluetooth.
Στα 3.3 GHz με 3.8 GHz έχουμε τα Fixed Wireless Access Systems (FWA).
Στα 4.8 GHz με 5.8 GHz έχουμε το Wi-Fi.
Στα 11 GHz με 15 GHz έχουμε την δορυφορική τηλεόραση.
Υπάρχουν ελεύθερες ζώνες (2.4 GHz και 5 GHz) και ζώνες στις οποίες πρέπει κανείς να πληρώσει για να έχει το δικαίωμα να τις χρησιμοποιήσει. Στην Ελλάδα υπεύθυνη για τη διαχείριση του φάσματος είναι η Εθνική Επιτροπή Τηλεπικοινωνιών και Ταχυδρομείων(ΕΕΤΤ).

Η διαχείριση του φάσματος είναι κρίσιμη για την εύρυθμη λειτουργία των σύγχρονων τηλεπικοινωνιών και απαιτεί συνεργασία σε διεθνές, εθνικό και τοπικό επίπεδο.

ΚΕΦΑΛΑΙΟ 3 Software Defined Radios (SDR)

3.1 Εισαγωγή

Ο τομέας των τηλεπικοινωνιακών συστημάτων και των δικτύων είναι από τους πιο αναπτυσσόμενους στη σύγχρονη εποχή. Οι άνθρωποι αναζητούν συνεχώς νέους τρόπους και μέσα για να επικοινωνήσουν. Έτσι έχουμε μεγάλη εξέλιξη και πολλές και συχνές αλλαγές. Τα παραδοσιακά συστήματα που είναι βασισμένα στο hardware “δυσκολεύονται” να ακολουθήσουν αυτούς τους ρυθμούς των αλλαγών λόγω της γρήγορης απαρχαίωσης του hardware. Χρειαζόμασταν λοιπόν κάτι το οποίο να μπορεί να αλλάζει πολύ εύκολα. Αυτά είναι τα Software Defined Radios (SDRs) και αποτελούν επανάσταση στο χώρο των ασύρματων τηλεπικοινωνιών καθώς όχι μόνο μας απαλλάσσουν από πολλά προβλήματα που είχαμε εώς τώρα αλλά επιπρόσθετα μας δίνουν νέες δυνατότητες.

3.2 Τι είναι ένα ραδιοσύστημα;

Ως ραδιοσύστημα θα μπορούσαμε να χαρακτηρίσουμε οποιαδήποτε συσκευή μπορεί να χρησιμοποιηθεί για την ανταλλαγή πληροφορίας μεταξύ δύο σημείων μέσω ενσύρματης ή ασύρματης ζεύξης.

3.3 Τι είναι τα SDR συστήματα;

Τα SDRs είναι συστήματα ραδιοεπικοινωνίας των οποίων τα συστατικά στοιχεία που παραδοσιακά ήταν υλοποιημένα με υλικό (hardware) (μίκτες, φίλτρα, ενισχυτές, διαμορφωτές/αποδιαμορφωτές κτλ.) πλέον υλοποιούνται με λογισμικό (software) σε κάποιο προσωπικό υπολογιστή ή σε κάποιο ενσωματωμένο σύστημα.
Όλες οι λειτουργίες μιας SDR συσκευής υλοποιούνται μέσω λογισμικού χωρίς να απαιτείται καμία αλλαγή σε επίπεδο υλικού.

Χοντρικά μία SDR συσκευή λαμβάνει ηλεκτρομαγνητικά κύματα και κάνοντας τις κατάλληλες μετατροπές τα παραδίδει στον υπολογιστή μας ώστε να τα επεξεργαστούμε ή να τα απεικονίσουμε. Πέρα από την λήψη υπάρχει και η δυνατότητα εκπομπής σημάτων.[3,4]

3.4 Ιστορικά στοιχεία

Το 1970 συναντάμε για πρώτη φορά τον όρο ψηφιακός δέκτης, σε ένα εργαστήριο στην Καλιφόρνια. Εκεί δημιουργήθηκε το σύστημα “Μίδας” το οποίο είναι ένα εργαλείο το οποίο έκανε ανάλυση σημάτων βασικής ζώνης (baseband) με χρήση λογισμικού (software).
Το 1982 ένας Γερμανόαμερικανός ηλεκτρολόγος μηχανικός, ο κύριος Ulrich Lothar Albert Rohde μαζί με την ομάδα του ανέπτυξαν την πρώτη SDR συσκευή. Δούλευαν σε μια μεγάλη εταιρία ηλεκτρονικών στην Αμερική η οποία συνεργαζόταν εκείνο τον καιρό με το υπουργείο Άμυνας των ΗΠΑ.
Το 1984 στο Τέξας σε μία εταιρία που λέγεται Raytheon κατασκευάσαν έναν ψηφιακό δέκτη σημάτων βασικής ζώνης οποίος με χρήση πολλών φίλτρων μπορούσε να αποφύγει τις παρεμβολές και να αποδιαμορφώσει ευρυζωνικά (broadband) σήματα.
Το 1991 ξεκίνησε ένα πρόγραμμα της Αμερικανικής αεροπορίας, με ονομασία SPEAKeasy. Στόχος τους να αναπτύξουν ένα ραδιόφωνο με δυνατότητα επικοινωνίας σε πολλαπλά πρωτόκολλα.
Το 1992 ο Joseph Mitola κάνει τη δημοσίευση του πρώτου άρθρου για τα software radios στην IEEE.
Το 2001 δημιουργείται το GNU - Radio το οποίο είναι ένα open source framework που παρέχει έτοιμα blocks για τη δημιουργία software-defined radios και συστήματα επεξεργασίας σημάτων. Αυτό γίνεται είτε με χρήση κάποιου RF hardware για δημιουργία software-defined radio είτε χωρίς hardware σε κάποιο περιβάλλον προσομοίωσης.[4]

3.5 Πλεονεκτήματα των SDR συσκευών

Πάμε να δούμε τι είναι αυτό που κάνει τα SDR συστήματα ξεχωριστά. Μερικά από τα πλεονεκτήματα των SDR συσκευών αναφέρονται παρακάτω.

- Με τα SDRs μπορούμε να χρησιμοποιήσουμε το ίδιο hardware για διάφορες εφαρμογές, έτσι μειώνεται η ανάγκη μας για χρήση εξειδικευμένων κάθε φορά συσκευών. Αυτό οδηγεί σε μεγάλη εξοικονόμηση χρημάτων καθώς το hardware δεν μας είναι άχρηστο όταν παλιώσει πχ μια τεχνολογία και αντικατασταθεί από μία νεότερη ή προγραμματίσουμε την SDR συσκευή μας για να τρέξει ένα άλλο πρωτόκολλο.
- Τα SDRs μπορούν με κάποιες αλλαγές στον κώδικά τους να υποστηρίζουν διαφορετικές συχνότητες, διαμορφώσεις χωρίς να χρειάζεται να αλλάξουμε hardware. Αυτό επιτρέπει εύκολες αναβαθμίσεις και προσαρμογή σε νέα πρωτόκολλα.
- Μπορούν να εκπέμψουν και να λάβουν σε πολλαπλά κανάλια την ίδια χρονική στιγμή.
- Μας δίνουν τη δυνατότητα να επαναπρογραμματίσουμε το σύστημα μας την ώρα που αυτό λειτουργεί. Έτσι μπορούμε να διορθώνουμε λάθη άμεσα και συνεχώς.
- Μπορούμε να κάνουμε τα πάντα σε software.
- Μπορούμε να χρησιμοποιήσουμε το software μεταξύ διαφορετικών προιόντων, κι έτσι αποφεύγουμε κόστος για την ανάπτυξη του software.
- Στην έρευνα δίνουν τη δυνατότητα σε ερευνητές να πειραματιστούν να αναπτύξουν και να δοκιμάσουν νέες τεχνολογίες που δεν υπήρχαν ως σήμερα ή που υπάρχουν αλλά δεν έχουν ακόμα βγει προς εμπορική χρήση.

3.6 Μειονεκτήματα των SDR συσκευών

Τίποτα στη ζωή δεν έχει μόνο θετικά, έτσι και οι SDR συσκευές έχουν τα αρνητικά τους. Παρακάτω αναφέρουμε μερικά μειονεκτήματα τους.

- Τα SDRs συχνά απαιτούν υψηλή επεξεργαστκή ισχύ, ειδικά σε real-time επικοινωνία. Αυτό οδηγεί σε μεγάλη κατανάλωση ενέργειας και είναι πρόβλημα για συσκευές που λειτουργούν με μπαταρίες.
- Για να αναπτύξει κανείς SDR συστήματα απαιτεί γνώση τόσο του software όσο και του hardware.
- Η software based προσέγγιση εισάγει κάποια καθυστέρηση λόγω της επεξεργασίας των σημάτων στους επεξεργαστές η οποία σε real-time εφαρμογές είναι κρίσιμη.
- Μπορεί τα SDR να προσφέρουν ευελιξία, όμως περιορίζονται από τους περιορισμούς του hardware της συσκευής στην οποία τρέχουν. Μπορεί να έχουμε περιορισμούς στο εύρος συχνοτήτων, στο bandwidth, στην ποιότητα των σημάτων και όχι μόνο.
- Τα SDR απαιτούν υψηλή επεξεργαστική ισχύ κάτι που αυξάνει το κόστος σε πρώτη φάση, και σε δεύτερη φάση αυτό δεν είναι και ότι καλύτερο για συστήματα με περιορισμένους πόρους όπως ενσωματωμένα συστήματα και κινητές συσκευές.
- Τα SDR συστήματα είναι πιο ακριβά από τα ραδιοσυστήματα απλής λειτουργίας καθώς έχουν πιο πολύπλοκη κατασκευή.
- Δεν συμφέρει να χρησιμοποιηθούν σε όλες τις συσκευές όπως πχ στα κινητά τηλέφωνα καθώς η διάρκεια ζωής ενός κινητού τηλεφώνου είναι γενικά μικρή.
- Το hardware γενικά είναι πιο γρήγορο από το software.

3.7 RF Hardware

Ένα σύστημα SDR αποτελείται από δύο μέρη, το εμπρόσθιο (RF front-end) και το ψηφιακό (digital back-end) τμήμα. Το εμπρόσθιο ασχολείται με την λήψη και την εκπομπή των αναλογικών σημάτων, ενώ το ψηφιακό ασχολείται με την επεξεργασία των ψηφιακών σημάτων.[5]

Το RF hardware υλοποιεί τις εξής λειτουργίες για τη λήψη σημάτων:

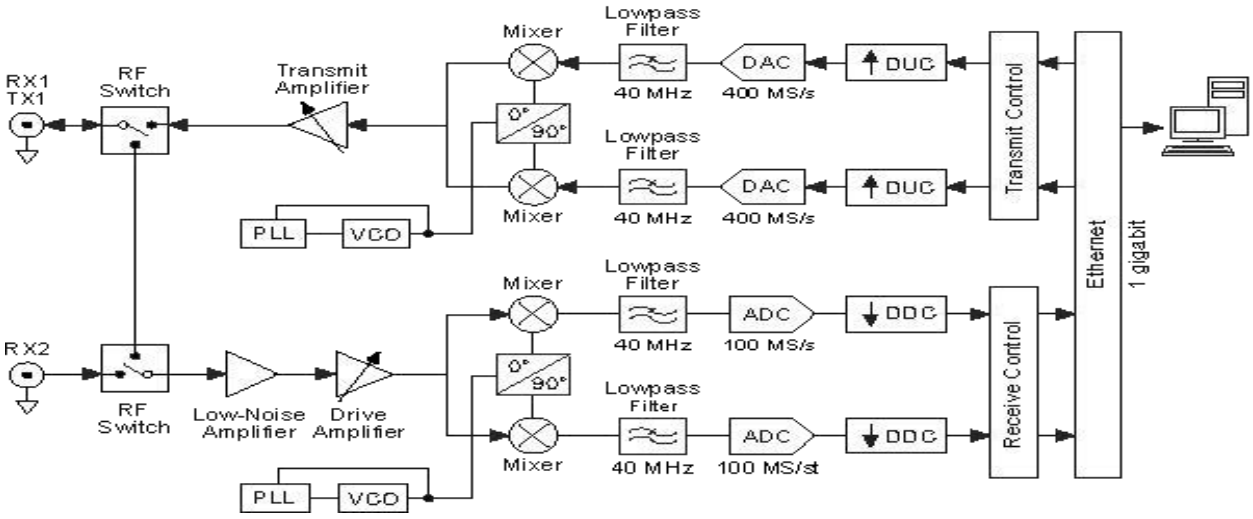
- Λαμβάνοντας το σήμα στην κεραία, σπάμε ουσιαστικά το σήμα μας σε δύο διαφορετικά μονοπάτια. Έτσι έχουμε δύο streams, τα οποία θα συνενωθούν ξανά λίγο πριν περάσουμε τα δείγματα στη μεριά του υπολογιστή. Γίνεται μετατροπή δηλαδή των πραγματικών αριθμών σε μιγαδικούς (complex numbers). Το ένα stream έχει το πραγματικό μέρος των μιγαδικών και το άλλο το φανταστικό μέρος των μιγαδικών. Θα περάσουν χώρια κάποια μέρη του SDR και στη συνέχεια θα ενωθούν ξανά λίγο πριν φτάσουν στον ηλεκτρονικό υπολογιστή.
- Φιλτράρισμα (Filtering)
  Κόβουμε κάποιες από τις συχνότητες που δε μας χρειάζονται, φυσικά και μπορούμε να το κάνουμε και σε δεύτερη φάση αυτό με χρήση software.
- Μετατροπή του σήματος από αναλογικό σε ψηφιακό.
  Ο υπολογιστής δεν αναγνωρίζει κάτι πέρα από αριθμούς, οπότε αφού λάβαμε τα δείγματα πρέπει να του τα δώσουμε σε μία κατανοητή γι αυτόν μορφή ώστε να τα επεξεργαστούμε με τη βοήθεια του.
- Downconversion
  Κατεβάζουμε το σήμα μας στην συχνότητα 0 Hz, το κάνουμε δηλαδή baseband γιατί εκεί είναι πιο εύκολη η επεξεργασία του.
- Decimation
  Καθώς το σήμα περνάει από τα διάφορα μέρη του SDR μπορεί να αλλάξει ο ρυθμός μετάδοσής του. Πριν φτάσει στον υπολογιστή μας θα πρέπει να έχει το ρυθμό που είχαμε προκαθορίσει αρχικά.
- Αποστολή στον ηλεκτρονικό υπολογιστή μέσω USB, Ethernet.

Όλες αυτές ήταν οι λειτουργίες που υλοποιεί το RF hardware για λήψη σημάτων. Υπάρχουν αντίστοιχες λειτουργίες που υλοποιεί το RF hardware για την εκπομπή σημάτων και ουσιαστικά είναι οι αντίστροφες από αυτές που είδαμε. Στην περίπτωση εκπομπής ο ηλεκτρονικός υπολογιστής στέλνει τα μιγαδικά δείγματα (samples) στην SDR συσκευή, γίνονται οι αντίστροφες λειτουργίες από αυτές που είδαμε και το σήμα εκπέμπεται.

Αφού είδαμε ποιες λειτουργίες υλοποιούνται από το RF hardware καλό θα ήταν να δούμε και αυτές που δεν γίνονται από το RF hardware.

Συγκεκριμένα δεν γίνεται ανίχνευση σήματος, διαμόρφωση, αποδιαμόρφωση, συχρονισμός, κωδικοποίηση και αποκωδικοποίηση. Στα κλασσικά RF συστήματα όλες αυτές οι λειτουργίες υλοποιούνταν από το hardware. Όταν χρησιμοποιούμε όμως μία SDR συσκευή τότε πρέπει όταν φτάσουν τα δείγματα στον υπολογιστή να κάνουμε εμείς κάποιες από αυτές τις λειτουργίες για να πάρουμε την πληροφορία που μεταδόθηκε πάνω από τον “αέρα”.

Στην παρακάτω εικόνα βλέπουμε την αρχιτεκτονική λήψης και εκπομπής μιας SDR συσκευής.



Εικόνα 3: Αρχιτεκτονική SDR

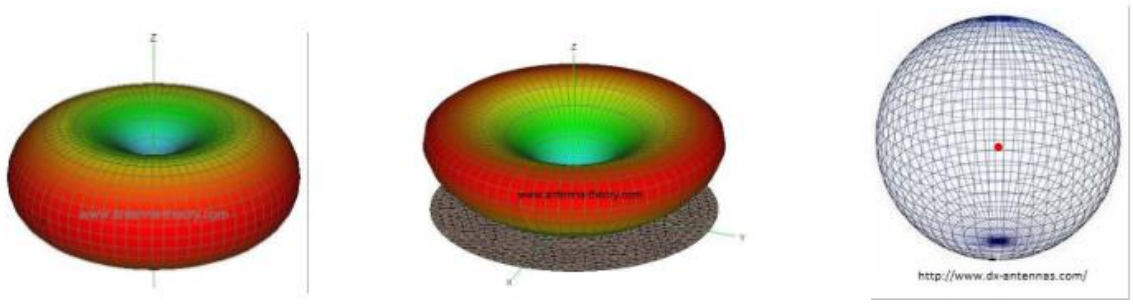
Ας δούμε όμως ένα ένα τα μέρη αυτών των διαδρομών..

RF Διακόπτης

Επειδή εδώ υπάρχουν δύο κεραίες ο διακόπτης αυτός χρειάζεται για να μπορούμε να κάνουμε επιλογή κεραίας. Δεν είναι όλες οι κεραίες το ίδιο, ούτε δουλεύουν καλά σε όλες τις συχνότητες η καθεμιά.

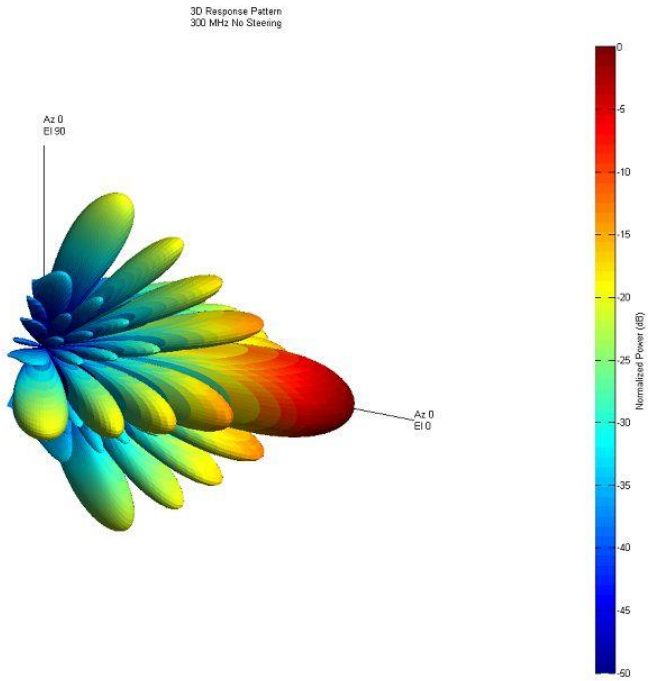
Κεραία

Η λειτουργία μιας κεραίας είναι να μετατρέπει την ηλεκτρομαγνητική ακτινοβολία σε ηλεκτρικό σήμα και να το ενισχύει λίγο. Αυτό όταν μιλάμε για λήψη σήματος, όταν έχουμε μετάδοση παίρνει ένα ηλεκτρικό σήμα το μετατρέπει σε ηλεκτρομαγνητική ακτινοβολία και το στέλνει.
Κάθε κεραία έχει ένα εύρος συχνοτήτων στο οποίο δουλεύει σωστά, δεν ενισχύει το ίδιο το σήμα παντού, ρόλο επίσης στη λειτουργία της παίζει η κατευθυντικότητα της. Βάσει αυτής έχουμε 3 τύπους κεραιών, τις μονοπολικές, τις διπολικές, τις ισοτροπικές, οι οποίες δεν υπάρχουν αλλά είναι οι ιδανικές που θα θέλαμε να υπάρχουν.



Εικόνα 4: Διπολική κεραία (αριστερά), Μονοπολική κεραία (κέντρο), Ισοτροπική κεραία (δεξιά).

Στην πραγματικότητα μια κατευθυντική κεραία είναι κάπως έτσι:



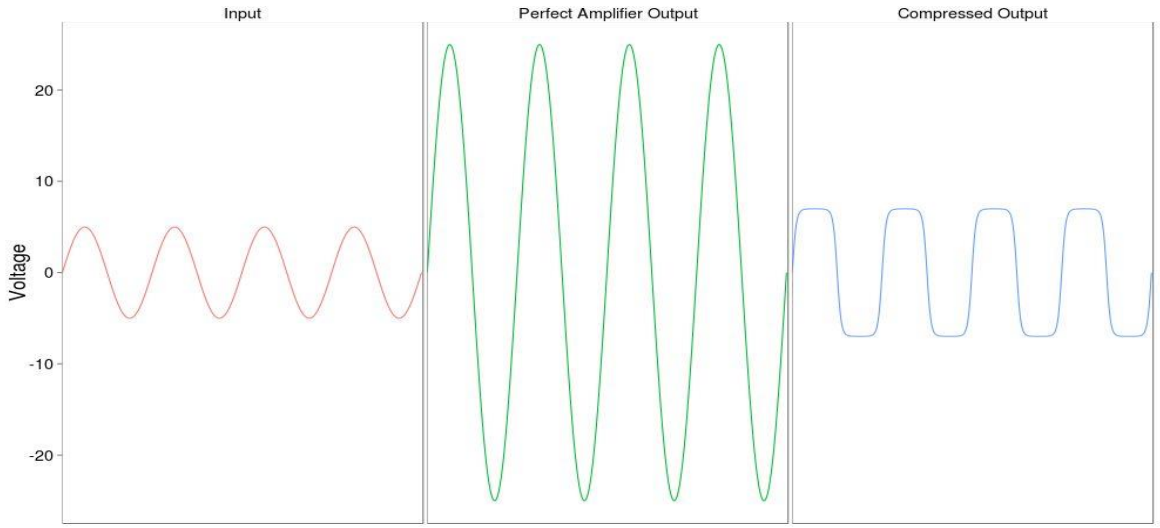
Εικόνα 5: Κατευθυντική κεραία

Ενισχυτής χαμηλού θορύβου (Low noise amplifier)

Γενικά, η δουλειά ενός ενισχυτή είναι το σήμα που θα πάρει στην είσοδο του να το πολλαπλασιάσει με ένα gain και να το δώσει στην έξοδο του. Ο συγκεκριμένος ενισχυτής προσπαθεί να ενισχύσει το σήμα μας κρατώντας χαμηλά το θόρυβο με σκοπό να έχουμε ένα καλό SNR στην έξοδο του. Δηλαδή να το σήμα μας να είναι αρκετά ισχυρό σε σχέση με το θόρυβο ώστε να έχουμε μια καλής ποιότητας επικοινωνία.

Υπάρχουν συγκεκριμένες συχνότητες στις οποίες μπορεί να ανταποκριθεί καλά ένας ενισχυτής.

Το εύρος τιμών στο οποίο συμπεριφέρεται γραμμικά ένας ενισχυτής έχει να κάνει με την ποιότητα κατασκευής του, οι πιο φτηνοί ενισχυτές έχουν μικρότερο πεδίο τιμών, ενώ οι ακριβότεροι έχουν μεγαλύτερο πεδίο τιμών.
Αν το εύρος είναι μικρό τότε έχουμε gain compression δηλαδή δεν έχουμε καλή ενίσχυση και αλλάζει η μορφή του σήματος μας. Αυτό φαίνεται στην παρακάτω εικόνα όπου βλέπουμε το σήμα στην είσοδο του ενισχυτή και αντίστοιχα βλέπουμε 2 εξόδους, την τέλεια περίπτωση και την compressed. Η διαφορά αυτή οφείλεται στο εύρος τιμών των ενισχυτών.



Εικόνα 6: Η τέλεια και η compressed έξοδος του ενισχυτή

Σημαντικό είναι και το noise figure του ενισχυτή, δηλαδή ο λόγος του πραγματικού θορύβου του ενισχυτή προς τον θερμικό θόρυβο (σε λογαριθμική αναπαράσταση). Όλα τα ηλεκτρονικά μέρη εισάγουν κάποιο θόρυβο ο οποίος είναι αναπόφευκτος. Αυτός ο θόρυβος λέγεται θερμικός θόρυβος (thermal noise). Υπάρχει και θόρυβος επιπλέον του θερμικού θορύβου σε κάθε ένα από αυτά τα μέρη. Το noise figure που ανέφεραμε προηγουμένως είναι ο λόγος του θορύβου και του θερμικού θορύβου σε κάποιο συγκεκριμένο εύρος συχνοτήτων, δηλαδή Noise / Thermal Noise σε dBs.

Μίκτης - VCO - PLL - Αλλαγή φάσης (Phase Shift)

Ο ρόλος του μίκτη είναι να προσθέσει δύο σήματα στην είσοδό του και το αποτέλεσμα να το δώσει στην έξοδο. Στις δύο εισόδους του μίκτη έχουμε τη συχνότητα την οποία έχουμε λάβει από τον ενισχυτή και τη συχνότητα που έχουμε λάβει από τον τοπικό ταλαντωτή. Στην έξοδο του μίκτη θα πάρουμε ένα σήμα με μία ενδιάμεση συχνότητα IF η οποία είναι μια προσθαφαίρεση των σημάτων που έχουμε στην είσοδο.

Κάθε RF συσκευή διαθέτει έναν ταλαντωτή (Voltage-controlled oscillator) ο οποίος είναι υπεύθυνος ώστε να παράξει τη συχνότητα στην οποία είτε θέλουμε να μεταδώσουμε είτε να λάβουμε ένα σήμα. Ο ταλαντωτής αυτός ανάλογα με την τάση που έχει στην είσοδο αλλάζει και η συχνότητα που παράγει.

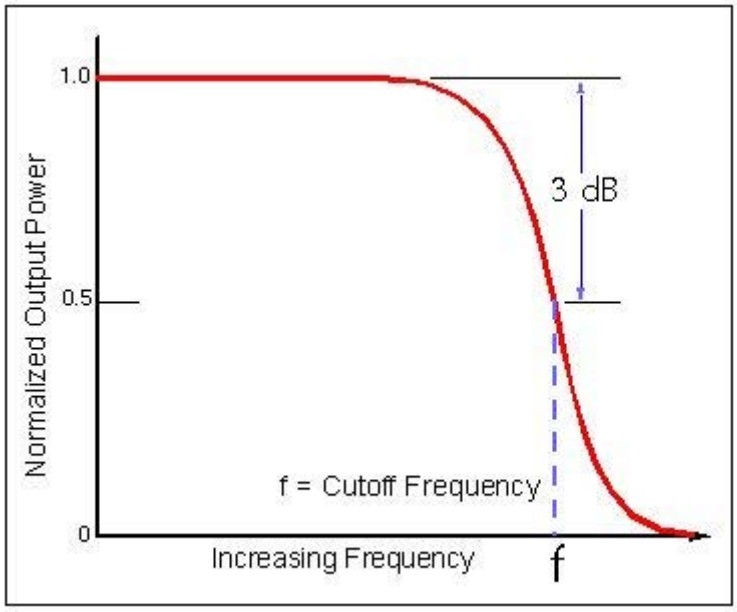
Το PLL (phase locked loop) έχει σαν σκοπό του να συγχρονίσουμε τη συχνότητα και τη φάση του τοπικού ταλαντωτή με τη συχνότητα και τη φάση του σήματος που λαμβάνουμε για να μπορέσουμε να κάνουμε σωστά τη μίξη των δύο σημάτων και να συνεχίσουμε την επεξεργασία του σήματος μας.

Το Phase shift αλλάζει τη φάση του σήματος μας κατά 90 μοίρες ώστε να έχουμε το “in phase” και το “quadrature” σήμα.

Χαμηλοπερατά φίλτρα (Low pass filters)

Τα χαμηλοπερατά φίλτρα τα χρησιμοποιούμε για να μπορούμε να κόψουμε κάποιες συχνότητες οι οποίες δεν είναι επιθυμητές, στη συγκεκριμένη περίπτωση τις υψηλές συχνότητες. Στην έξοδο του χαμηλοπερατού φίλτρου θα πάρουμε το σήμα μας έχοντας κατά πολύ μειωμένες τις υψηλές συχνότητες. Χαρακτηρίζονται από την συχνότητα αποκοπής (cutoff frequency). Συχνότητες μικρότερες της συχνότητας αποκοπής περνάνε κανονικά στην έξοδο ενώ όσες είναι μεγαλύτερες της εξασθενούν.

Όσο καλύτερο το φίλτρο τόσο πιο κατακόρυφη θα είναι η γραμμή στη συχνότητα αποκοπής.



Εικόνα 7: Χαμηλοπερατό φίλτρο

ADC – Μετατροπέας αναλογικού σε ψηφιακό

Το σήμα μας πρέπει πλέον να μετατραπεί σε ψηφιακό ούτως ώστε να μπορεί να γίνει κατανοητό από τον υπολογιστή. Στην είσοδο του ADC έχουμε το αναλογικό σήμα μας και στην έξοδο παίρνουμε τα ψηφιακά δείγματα (samples).
Εδώ γίνεται η δειγματοληψία του σήματος μας, κάτι το οποίο θα δούμε παρακάτω. Όσο πιο ακριβός είναι ο συγκεκριμένος ADC τόσο καλύτερη ανάλυση έχει. Όσο πιο πολλά bits ανάλυση έχει τόσο πιο πολλούς αριθμούς μπορεί να αναπαραστήσει.

DDC – Digital downconversion

Σε αυτό το στάδιο το σήμα μας γίνεται baseband φέρνοντας τη συχνότητα των δειγμάτων μας στα 0Hz και γίνεται και το decimation ώστε να έχει το σήμα μας τον επιθυμητό ρυθμό μετάδοσης χωρίς να χαθεί πληροφορία.

RX Control

Σε αυτό το σημείο η πληροφορία που έχουμε λάβει είναι έτοιμη να σταλεί στον υπολογιστή και ανάλογα με τον τρόπο που θα σταλεί υπάρχει και μια διαφορετική

επεξεργασία. Πχ άλλη διαδικασία για Ethernet, άλλη διαδικασία για USB 2.0, άλλη για PCI Express κλπ.

Στην εργασία αυτή θα μας απασχολήσει το να λαμβάνουμε σήματα γι αυτό και έγινε αναλυτική επεξήγηση των σταδίων της λήψης. Για τη διαδικασία της εκπομπής σήματος η διαδικασία είναι η ακριβώς αντίστροφη ξεκινώντας από τον ηλεκτρονικό υπολογιστή και καταλήγοντας στην κεραία της SDR συσκευής.

3.8 PLUTO SDR

Το PlutoSDR είναι ένα σχετικά φτηνό SDR το οποίο μπορεί να χρησιμοποιηθεί για εκπαιδευτικούς σκοπούς και όχι μόνο. Είναι ικανό να λαμβάνει και να εκπέμπει σήματα από 70 MHz μέχρι και 6 GHz με εώς και 61.44 Mega Samples per second (MSPS) σε 20 MHz Bandwidth. Έχει USB 2.0 κάτι το οποίο σημαίνει ότι μας περιορίζει το sample rate στα 5 MHz αν θέλουμε να λαμβάνουμε το 100% των δειγμάτων στο χρόνο. Υποστηρίζει βέβαια sample rate εώς και 61 MHz (με απώλεια δειγμάτων λόγω του USB 2.0). Έχει ένα κανάλι για λήψη και ένα κανάλι για εκπομπή σήματος, full duplex. ‘Εχει μικρό μέγεθος, χωράει σε μια τσέπη και ζυγίζει περίπου 300 γραμμάρια.
Έχει μειωμένα χαρακτηριστικά λόγω του χαμηλότερου κόστους, πχ αυτή η συσκευή έχει USB 2.0, άλλες ακριβότερες έχουν PCI express, GigaBit Ethernet, οπτική ίνα. Δεν έχει embedded επεξεργαστή κι από μόνο του δεν μπορεί να τρέξει.[6]

Εικόνα 8: Pluto SDR, εξωτερικά(αριστερά), εσωτερικά και σε επίπεδο components(κέντρο και δεξιά).

ΕΓΚΑΤΑΣΤΑΣΗ ΚΑΙ ΣΥΝΔΕΣΗ PLUTO SDR

Για λειτουργικό σύστημα WINDOWS θα πρέπει να εγκαταστήσουμε τους drivers από το σύνδεσμο που βρίσκεται στην βιβλιογραφία – αναφορές.[7]

Για λειτουργικό σύστημα LINUX δε χρειάζεται κάτι, απλά το συνδέουμε σε μία διαθέσιμη θύρα USB και θα εμφανιστεί σαν Ethernet USB adapter.
Μπορούμε να κάνουμε ένα ping στη διεύθυνση 192.168.2.1 για να δούμε αν όλα είναι καλά. Αν το ping δουλεύει αφαιρούμε το Pluto SDR από τη USB θύρα και κοιτάμε αν εξακολουθεί να δουλεύει το ping. Αν συνεχίζει να δουλεύει τότε αυτό σημαίνει οτι κάποια άλλη συσκευή χρησιμοποιεί αυτή τη διεύθυνση και θα πρέπει να αλλάξουμε διεύθυνση στο Pluto SDR. Το ίδιο θα κάνουμε κι αν το ping δεν δούλευε αρχικά. Η διεύθυνση μας χρειάζεται και στο πρόγραμμα μας οπότε καλό θα είναι να την σημειώσουμε κάπου.

ΑΛΛΑΓΗ IP ΔΙΕΥΘΥΝΣΗΣ PLUTO SDR

Αναφέραμε προηγουμένως οτι το Pluto SDR εμφανίζεται σαν USB. Μέσα υπάρχει ένα αρχείο config.txt, το ανοίγουμε, βάζουμε την επιθυμητή διεύθυνση και κάνουμε eject το Pluto SDR (προσοχή, δε το αποσυνδέουμε). Περιμένουμε λίγα

δευτερόλεπτα και μετά το αποσυνδέουμε και το ξανασυνδέουμε. Ελέγχουμε το αρχείο config.txt να δούμε αν οι αλλαγές μας έχουν αποθηκευτεί. Αν ναι, είμαστε έτοιμοι.

ΕΓΚΑΤΑΣΤΑΣΗ PLUTO SDR

Οι παρακάτω εντολές θα μας δώσουν τη δυνατότητα να μπορούμε να χρησιμοποιούμε το Pluto SDR στον υπολογιστή μας. Συγκεκριμένα εγκαθιστούν τις παρακάτω βιβλιοθήκες:

1. **libiio**
2. **libad9361-iio**
3. **pyadi-iio**

```
sudo apt-get update
sudo apt-get install build-essential git libxml2-dev bison flex libcdk5-dev cmake python3-pip libusb-1.0-0-dev libavahi-client-dev libavahi-common-dev libaio-dev
cd ~
git clone --branch v0.23 https://github.com/analogdevicesinc/libiio.git
cd libiio
mkdir build
cd build
cmake -DPYTHON_BINDINGS=ON ..
make -j$(nproc)
sudo make install
sudo ldconfig

cd ~
git clone https://github.com/analogdevicesinc/libad9361-iio.git
cd libad9361-iio
mkdir build
cd build
cmake ..
make -j$(nproc)
sudo make install
```

```
cd ~
git clone --branch v0.0.14 https://github.com/analogdevicesinc/pyadi-iio.git
cd pyadi-iio
pip3 install --upgrade pip
pip3 install -r requirements.txt
sudo python3 setup.py install
```

ΠΑΡΑΜΕΤΡΟΠΟΙΗΣΗ PLUTO SDR ΓΙΑ ΝΑ ΑΥΞΗΣΟΥΜΕ ΤΟ ΕΥΡΟΣ RF

Το Pluto SDR έχει περιορισμένο εύρος για τις κεντρικές συχνότητες και για τα sample rates. Το τσιπάκι του όμως είναι ικανό για μεγαλύτερες συχνότητες.
Μάλιστα τα παρακάτω βήματα τα προτείνει η ίδια η Analog Devices.
Ανοίγουμε ένα terminal και γράφουμε:
ssh root@192.168.2.1
ο κωδικός που θα μας ζητηθεί είναι “analog” .
Τώρα θα πρέπει να είμαστε στην αρχική οθόνη του Pluto SDR.
Αν έχουμε από την έκδοση 0.31 και κάτω τότε τρέχουμε τις εξής εντολές:

```
fw_setenv attr_name compatible
fw_setenv attr_val ad9364
reboot
```

Αν έχουμε από 0.32 και μετά τότε τρέχουμε αυτές τις εντολές:

```
fw_setenv compatible ad9364
reboot
```

Αυτό ήταν! Πλέον μπορούμε να έχουμε κεντρικές συχνότητες στο εύρος από τα 70 MHz εώς και τα 6 GHz και sample rates εως 56 MHz.[8]

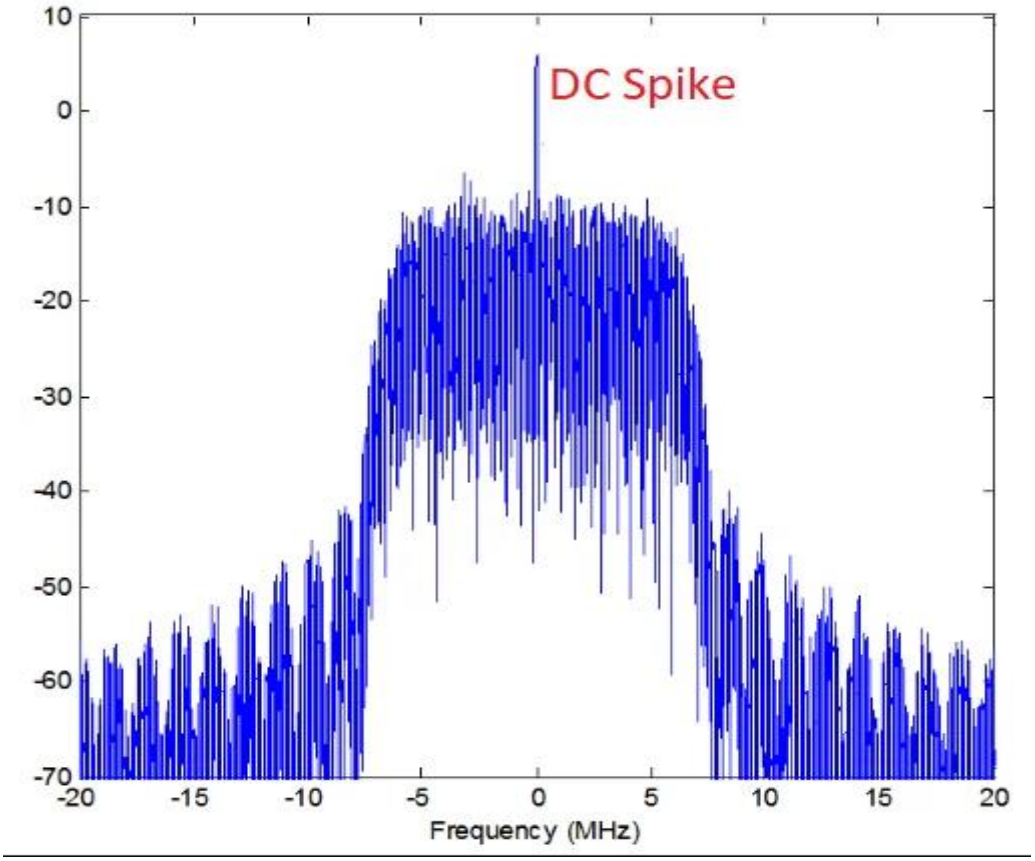
3.9 ΠΡΟΒΛΗΜΑΤΑ ΤΩΝ SDR ΣΥΣΚΕΥΩΝ

FREQUENCY OFFSET

Υπάρχει περίπτωση να συντονίσουμε το SDR μας σε κάποια κεντρική συχνότητα και να παρατηρήσουμε πως το σήμα μας δεν είναι ακριβώς στο κέντρο. Για παράδειγμα να βάλουμε το SDR μας στα 2.5 GHz και το σήμα να φαίνεται σε κάποια συχνότητα λίγο μικρότερη από τα 2.5 GHz. Αυτό συμβαίνει γιατί το SDR μας έχει ένα τοπικό ταλαντωτή ο οποίος δεν είναι τέλειος και έχει κάποια ανοχή σε λάθη. Υπάρχει διαφοροποίηση από την μία SDR συσκευή στην άλλη και παίζουν κι άλλοι παράγοντες ρόλο γι αυτό το φαινόμενο όπως η θερμοκρασία, η συχνότητα, οι δονήσεις.

DC OFFSET

Ένα δεύτερο πρόβλημα που αντιμετωπίζουμε όταν χρησιμοποιούμε τις SDR συσκευές είναι το DC Offset. Αν έχουμε DC Offset θα δούμε στην κεντρική συχνότητα που έχουμε συντονίσει το SDR μας να υπάρχει μετάδοση η οποία όμως δεν είναι πραγματική. Είναι κάτι που εμφανίστηκε κατά την επεξεργασία του σήματος με τη συσκευή μας. Αυτό το φαινόμενο προκαλείται επειδή ο μίκτης της SDR συσκευής μας έχει κάποιες διαρροές. Πιο συγκεκριμένα εκτός από την συχνότητα που θα έπρεπε να έχουμε στην έξοδο του μίκτη παίρνουμε και κάποιες εξασθενημένες εκδόσεις των σημάτων που είχαμε στην είσοδο του μίκτη.
Στην παρακάτω εικόνα μπορούμε να δούμε πως μοιάζει το DC Offset στην πράξη. Λέγεται αλλιώς και DC Spike.



Εικόνα 9: DC spike

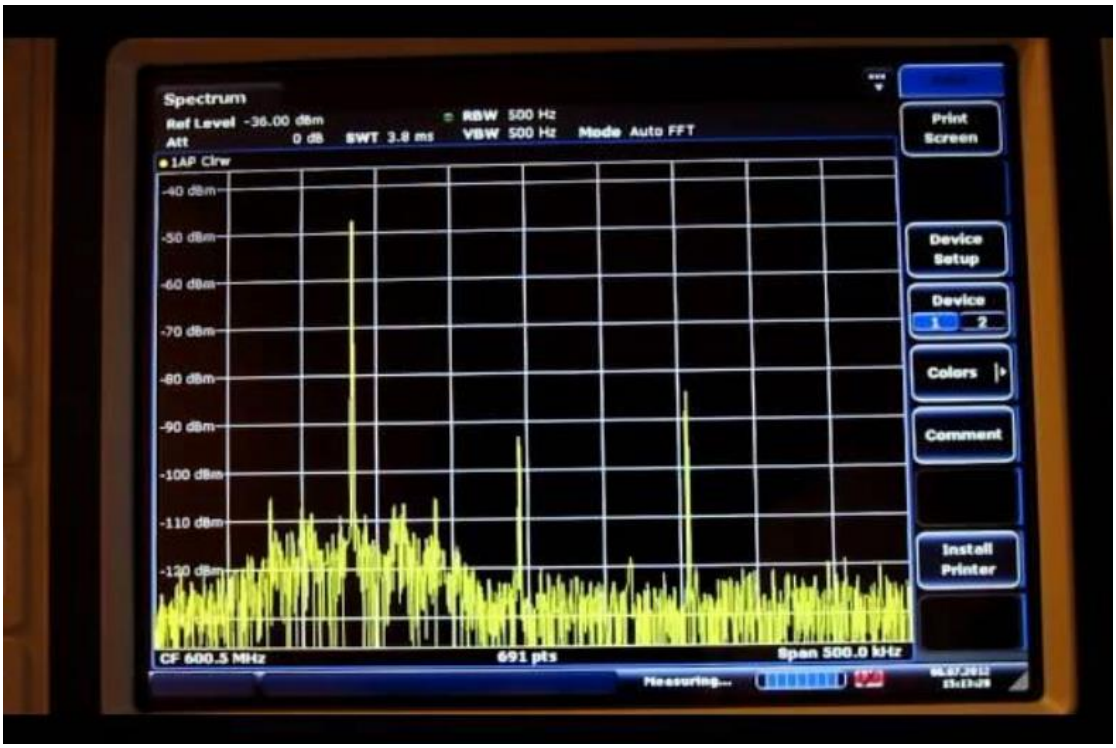
MIXER SPURS

Ένα τρίτο πρόβλημα που έχουμε όταν χρησιμοποιούμε τις SDR συσκευές είναι τα Mixer Spurs. Μοιάζει λίγο με το προηγούμενο υπό την έννοια ότι έχουμε κι εδώ την εμφάνιση μη πραγματικών μεταδόσεων. Εδώ ο μίκτης βγάζει στην έξοδο του αρμονικές της συχνότητας του τοπικού ταλαντωτή και της RF συχνότητας. Έτσι η έξοδος μας περιλαμβάνει σήματα σε αυτές τις συχνότητες

$n * F_{LO} \pm m * F_{RF}$ όπου n, m = 0, 1, 2, 3…

Για παράδειγμα αν είχαμε έναν τοπικό ταλαντωτή στα 100 MHz τότε θα είχαμε spurs στις συχνότητες 200 MHz, 300 MHz, 400 MHz …

Στην παρακάτω εικόνα βλέπουμε Mixer Spurs αλλά και DC Spike.



Εικόνα 10: DC spike και mixer spurs

ΚΕΦΑΛΑΙΟ 4 Σήματα και Δειγματοληψία

4.1 Σήματα

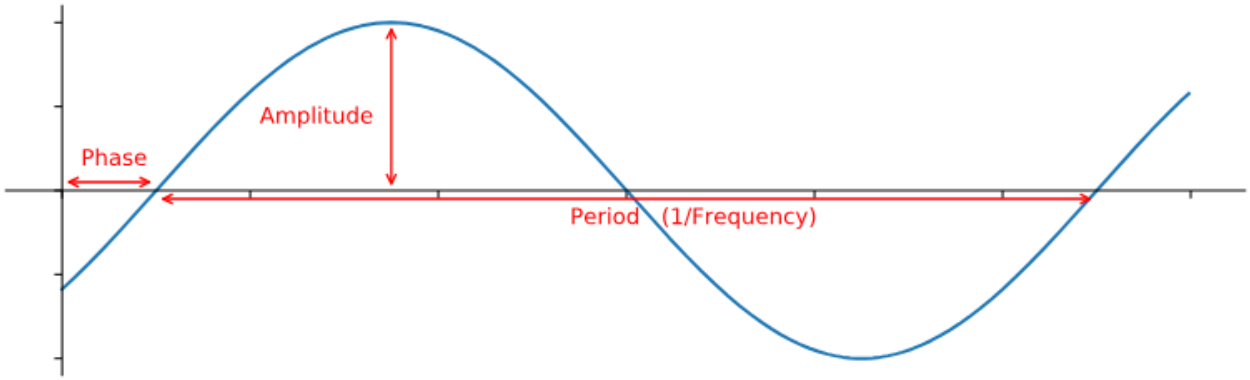
Τι είναι σήμα;

Ως σήμα ορίζουμε τη μεταβολή ενός φυσικού μεγέθους x = x(t) όπως πχ τάση, ρεύμα, ισχύς ως προς το χρόνο t.

Τι είναι ένα ηλεκτρικό σήμα;
Είναι μία μορφή μετάδοσης ενέργειας, πληροφορίας μέσω ηλεκτρικών φορτίων.

Περιοδικά σήματα
Ένα σήμα x(t) λέγεται περιοδικό σήμα αν υπάρχει θετικός αριθμός T τέτοιος ώστε να ισχύει x(t) = x(t + T) για κάθε t. Η ποσότητα T ονομάζεται περίοδος του σήματος και μετριέται σε δευτερόλεπτα (seconds). Η ελάχιστη περίοδος λέγεται θεμελιώδης περίοδος (T_o) και ορίζει την μικρότερη χρονική διάρκεια μετά την οποία το περιοδικό σήμα θα αρχίσει να επαναλαμβάνεται.
Η αντίστροφη ποσότητα της περιόδου T είναι η συχνότητα f του σήματος f = 1/T . Μας δίνει το πλήθος των επαναλήψεων του σήματος στην μονάδα του χρόνου και μετριέται σε Hz (Hertz).
Η κυκλική συχνότητα είναι η ποσότητα ω = 2π/T = 2πf, μετριέται σε ακτίνια ανά δευτερόλεπτο (rad/s).

Ένα παράδειγμα περιοδικών σημάτων είναι τα ημιτονοειδή σήματα με τα οποία θα ασχοληθούμε στα πλαίσια αυτής της εργασίας.
Αυτά είναι σήματα που περιγράφονται από μια σχέση της μορφής
x(t) = A cos(ωt + θ).
Το A στην παραπάνω σχέση είναι το πλάτος του σήματος, δηλαδή η ένταση του σήματος.
Το ω είναι η κυκλική συχνότητα που αναφέραμε λίγο παραπάνω και το θ είναι η φάση του σήματος. Η φάση αναφέρεται στη θέση του σήματος σε σχέση με το χρόνο. Στην εικόνα 11 βλέπουμε ένα ημιτονοειδές σήμα με τα χαρακτηριστικά του μεγέθη.



Εικόνα 11: Ημιτονοειδές σήμα

Απεριοδικά σήματα

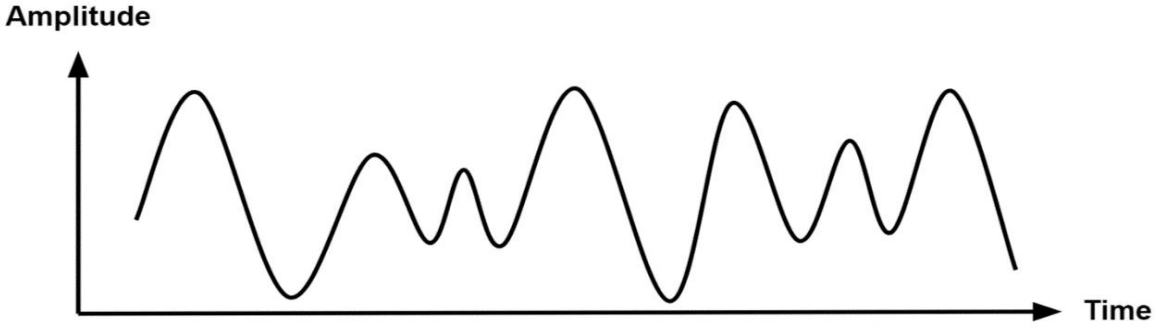
Ένα σήμα x(t) λέγεται απεριοδικό αν δεν υπάρχει θετικός αριθμός T τέτοιος ώστε να ισχύει x(t) = x(t + T) για κάθε t.

Με βάση τη μεταβλητή του χρόνου τα σήματα διακρίνονται σε **σήματα συνεχούς χρόνου** και **σήματα διακριτού χρόνου**.
Σήμα συνεχούς χρόνου είναι ένα σήμα x(t) το οποίο ορίζεται για κάθε τιμή του t στο διάστημα χρόνου (α,β).
Σήμα διακριτού χρόνου είναι ένα σήμα που ορίζεται μόνο για κάποιες (συγκεκριμένες) στιγμές του χρόνου.

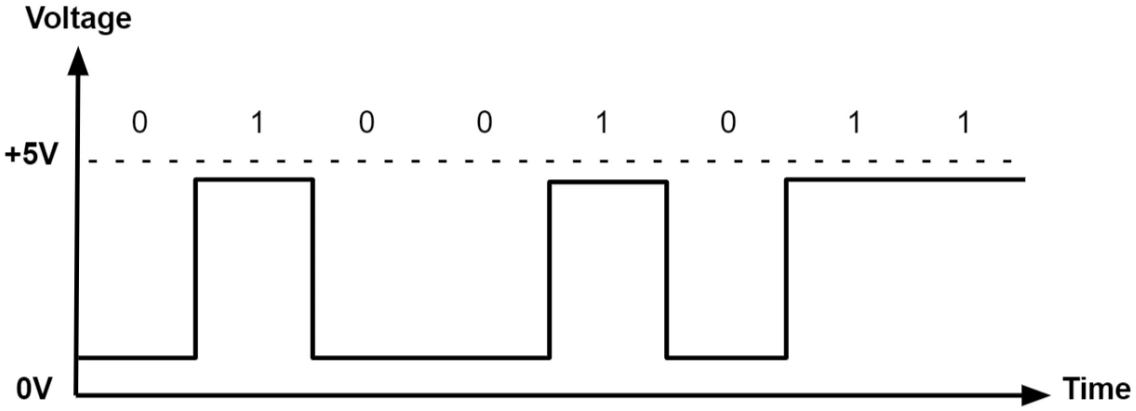
Με βάση την μεταβλητή του πλάτους τα σήματα διακρίνονται σε **σήματα συνεχούς τιμής** και σε **σήματα διακριτής τιμής**.
Σήμα συνεχούς τιμής είναι ένα σήμα που παίρνει όλες τις δυνατές τιμές σε ένα διάστημα τιμών.
Σήμα διακριτής τιμής είναι ένα σήμα που παίρνει τιμές από ένα πεπερασμένο σύνολο τιμών.

Αναλογικά και ψηφιακά σήματα

Τα αναλογικά σήματα είναι σήματα τα οποία λαμβάνουν δυνητικά οποιαδήποτε τιμή μέσα σε ένα διάστημα τιμών. Τα ψηφιακά σήματα από την άλλη μπορούν να πάρουν μόνο διακριτές τιμές.



Εικόνα 12: Αναλογικό σήμα



Εικόνα 13: Ψηφιακό σήμα

Μέση τιμή σήματος

Για περιοδικό σήμα συνεχούς χρόνου x(t) με περίοδο T η μέση τιμή του ισούται με:

$$x_{av} = \frac{1}{T}\int_{0}^{T} x(t)dt$$

Για περιοδικά σήματα διακριτού χρόνου με περίοδο N, η μέση ανά περίοδο τιμή είναι:

$$x_{av} = \frac{1}{N+1}\sum_{n=0}^{N} x[n]$$

Ενεργός τιμή σήματος

Για σήματα συνεχούς χρόνου η ενεργός τιμή ορίζεται σε ένα διάστημα [t1,t2] από την σχέση:

$$x_{rms} = \left[\frac{1}{t_2 - t_1} \int_{t_1}^{t_2} |x(t)|^2 \, dt \right]^{\frac{1}{2}}$$

Για σήματα διακριτού χρόνου η ενεργός τιμή σε ένα χρονικό διάστημα [n1,n2] ορίζεται από τη σχέση:

$$x_{rms} = \left[\frac{1}{n_2 - n_1 + 1} \sum_{n=n_1}^{n_2} |x[n]|^2 \right]^{\frac{1}{2}}$$

Ενέργεια σήματος

Για σήματα συνεχούς χρόνου η ενέργεια ορίζεται σε ένα διάστημα [t1, t2] από την σχέση:

$$W_{[t_1,t_2]} = \int_{t_1}^{t} |x(t)|^2 \, dt$$

Για σήματα διακριτού χρόνου η ενέργεια ορίζεται σε ένα διάστημα [n1,n2] από τη σχέση:

$$W_{[n_1,n_2]} = \sum_{n=n_1}^{n_2} x^2[n]$$

Στιγμιαία ισχύς σήματος

Για σήματα συνεχούς χρόνου η στιγμιαία ισχύς ορίζεται από τη σχέση:

$$p(t)=|x(t)|^2$$

Για σήματα διακριτού χρόνου η στιγμιαία ισχύς ορίζεται από τη σχέση:

$$p[n] = |x[n]|^2$$

Μέση ισχύς σήματος

Για σήματα συνεχούς χρόνου σε ένα διάστημα [t1, t2] η μέση ισχύς του σήματος ορίζεται από την σχέση:

$$p_{av} = \frac{1}{t_2 - t_1} \int |x(t)|^2 dt$$

Για σήματα διακριτού χρόνου σε ένα διάστημα [n1,n2] η μέση ισχύς του σήματος ορίζεται από την σχέση:

$$p_{av} = \frac{1}{n_2 - n_1 + 1} \sum_{n=n_1}^{n_2} x^2[n]$$

Και κάπου εδώ ολοκληρώσαμε την αναφορά μας σε βασικές έννοιες των σημάτων. Από αυτή την ενότητα πολύ σημαντική για την εργασία είναι η ισχύς σήματος.[8,9,10,13]

4.2 Γραμμικές και λογαριθμικές αναπαραστάσεις

Στα σήματα μας απασχολεί η ισχύς με την οποία γίνεται μία εκπομπή ή αντίστοιχα η ισχύς ενός σήματος που λαμβάνουμε. Υπάρχουν δύο αναπαραστάσεις της

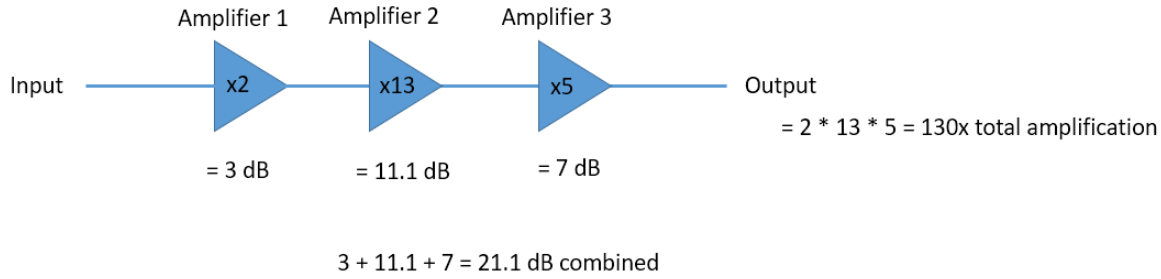
ισχύος του σήματος. Η γραμμική αναπαράσταση και η λογαριθμική αναπαράσταση. Στην γραμμική αναπαράσταση όπου είναι και η πιο συνηθισμένη η ισχύς μετριέται σε Watt. Στη λογαριθμική αναπαράσταση μετριέται σε dBs (dBW, dBm). Όταν έχουμε να αντιμετωπίσουμε μικρά νούμερα και μεγάλα ταυτόχρονα ή μόνο πολύ μεγάλους αριθμούς τότε μας βολεύει πολύ να χρησιμοποιούμε τη λογαριθμική κλίμακα. Ο τρόπος για να πάμε από τη γραμμική στη λογαριθμική αναπαράσταση είναι ο παρακάτω τύπος:

$$x_{dB} = 10 \log_{10} x$$

Η αντίστροφη διαδρομή επιτυγχάνεται μέσω του παρακάτω τύπου:

$$x = 10^{x_{dB}/10}$$

Στην επεξεργασία σημάτων έχουμε να κάνουμε με μεγάλους και μικρούς αριθμούς ταυτόχρονα. Ένας επιπλέον λόγος που μας βολεύει η λογαριθμική αναπαράσταση είναι ότι χρησιμοποιώντας την κάνουμε προσθέσεις εκεί που στη γραμμική αναπαράσταση κάνουμε πολλαπλασιασμούς. Για παράδειγμα μπορούμε να δούμε το παρακάτω σχήμα, δεξιά φαίνονται οι πράξεις που θα κάναμε για να βρούμε το output σε γραμμική αναπαράσταση, ενώ από κάτω φαίνονται οι πράξεις για τον υπολογισμό του output σε λογαριθμική αναπαράσταση.[8]



Εικόνα 14: Παράδειγμα υπολογισμού με γραμμική και λογαριθμική αναπαράσταση

Να ξεκαθαρίσουμε όμως ότι τα dBs δεν είναι μονάδα, μία τιμή μετρημένη σε dB δεν έχει μονάδα. Είναι σχετικό με κάτι άλλο. Για παράδειγμα η ισχύς ενός σήματος σε dBW είναι η ισχύς του σήματος σε σχέση με το 1 Watt. Αντίστοιχα η ισχύς ενός σήματος σε dBm είναι η ισχύς του σήματος σε σχέση με το 1 mWatt.

Ενδεικτικά παραθέτουμε μερικά παραδείγματα παρακάτω:

1x → 0 dB

2x → 3 dB

10x → 10 dB

0.5x → -3 dB

0.1x → -10 dB

100x → 20 dB

1000x → 30 dB

10000x → 40 dB

Για να μετατρέψουμε τις λογαριθμικές αναπαραστάσεις της ισχύος σε γραμμική αναπαράσταση, δηλαδή σε Watt ή mWatt, χρησιμοποιούμε έναν από τους δύο παρακάτω τύπους ανάλογα με το αν θέλουμε να μετατρέψουμε τα dBW ή τα dBm:

$P_{(W)} = 10 \text{ ^ } (P_{(dBw)} / 10dBW)$

$P_{(mW)} = 10 \text{ ^ } (P_{(dBm)} / 10dBm)$

Για τη μετατροπή από γραμμική αναπαράσταση δηλαδή από Watt ή mWatt σε λογαριθμική αναπαράσταση δηλαδή dBW ή dBm χρησιμοποιούμε έναν από τους δύο παρακάτω τύπους ανάλογα με το αν θέλουμε να μετατρέψουμε τα Watt ή mWatt:

$P_{(dBW)} = 10 * \log10(P_{(W)} / 1W)$

$P_{(dBm)} = 10 * \log10(P_{(mW)} / 1mW)$

4.3 SNR

Με τον όρο θόρυβος αναφερόμαστε σε τυχαία, απρόβλεπτα και ανεπιθύμητα σήματα τα οποία δε φέρουν κάποια χρήσιμη πληροφορία. Αυτά τα σήματα δημιουργούν προβλήματα στις μεταδόσεις μας και υποβαθμίζουν την ποιότητα της επικοινωνίας μας. Όπως όλα τα σήματα έτσι και ο θόρυβος έχει κάποια ισχύ.

Ο λόγος της ισχύος του σήματος προς την ισχύ του θορύβου ονομάζεται SNR (Signal to noise ratio).

$$\mathrm{SNR} = \frac{P_{signal}}{P_{noise}}$$

Το SNR επειδή είναι λόγος, δεν έχει μονάδα και είναι καθαρός αριθμός. Επιλέγουμε και πάλι λογαριθμική αναπαράσταση και ο τύπος υπολογισμού του είναι ο παρακάτω:

$$\mathrm{SNR_{dB}} = P_{signal\_dB} - P_{noise\_dB}$$

Η ερμηνεία του SNR είναι αρκετά απλή, σε περίπτωση που έχουμε μηδενικό SNR (0 dB) τότε η ισχύς σήματος και θορύβου είναι ίδια. Σε περίπτωση που το SNR είναι αρνητικό σημαίνει ότι η ισχύς του θορύβου είναι μεγαλύτερη από την ισχύ του σήματος μας. Τέλος σε περίπτωση που το SNR είναι θετικό αυτό σημαίνει ότι η ισχύς του σήματος μας είναι μεγαλύτερη από την ισχύ του θορύβου.

Χρειαζόμαστε το SNR να είναι όσο το δυνατόν μεγαλύτερο, τιμές κάτω από 20 dB είναι προβληματικές και η επικοινωνία μας θα παρουσιάζει προβλήματα. Απώλεια πακέτων, αδυναμία του δέκτη να ξεχωρίσει τα σύμβολα μεταξύ τους λόγω της υψηλής ισχύος του θορύβου σε σχέση με την ισχύ του σήματος μας είναι κάποιοι από τους λόγους για τους οποίους η επικοινωνία μας δεν είναι αυτή που θα θέλαμε.

Για να αυξήσουμε τις τιμές του SNR μπορούμε να αυξήσουμε την ισχύ του σήματος μας κρατώντας το θόρυβο χαμηλά, να φιλτράρουμε τον θόρυβο και να χρησιμοποιήσουμε τεχνικές διόρθωσης λαθών.[11]

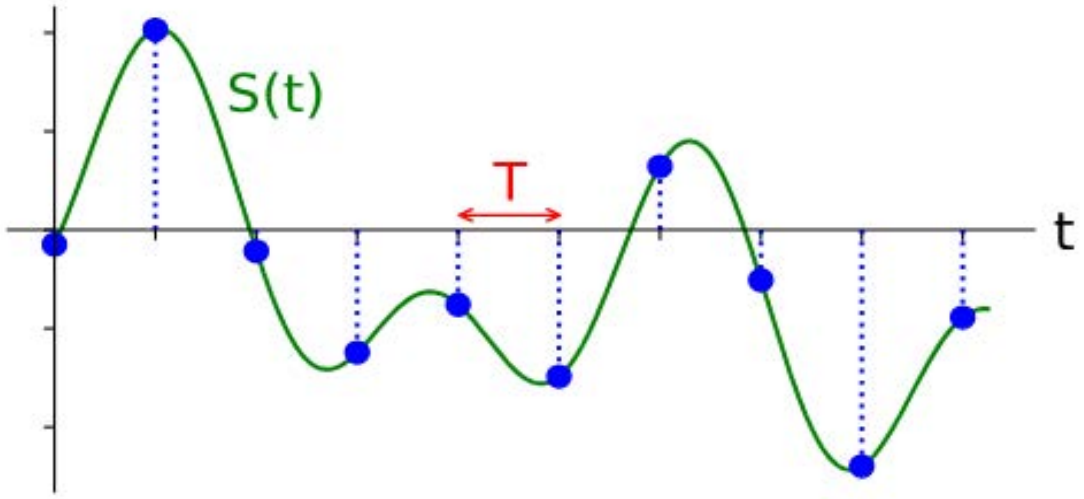
4.4 Δειγματοληψία

Οι SDR συσκευές κατά την λειτουργία του δέκτη λαμβάνουν ένα ηλεκτρομαγνητικό κύμα στην κεραία τους, εκεί αυτό μετατρέπεται σε ηλεκτρικό σήμα και μετά από επεξεργασία μας παραδίδουν αριθμούς, πιο συγκεκριμένα μιγαδικούς αριθμούς (complex numbers). Αυτά είναι τα δείγματα (samples). Προκύπτουν από την διαδικασία της δειγματοληψίας.

Έστω ότι έχουμε ένα σήμα S(t), η καταγραφή των τιμών του σήματος σε κάποιες χρονικές στιγμές ονομάζεται δειγματοληψία του σήματος S. Αν καταγράφουμε τιμές του σήματος ανά χρονικό διάστημα T τότε το T λέγεται περίοδος δειγματοληψίας (sampling period).

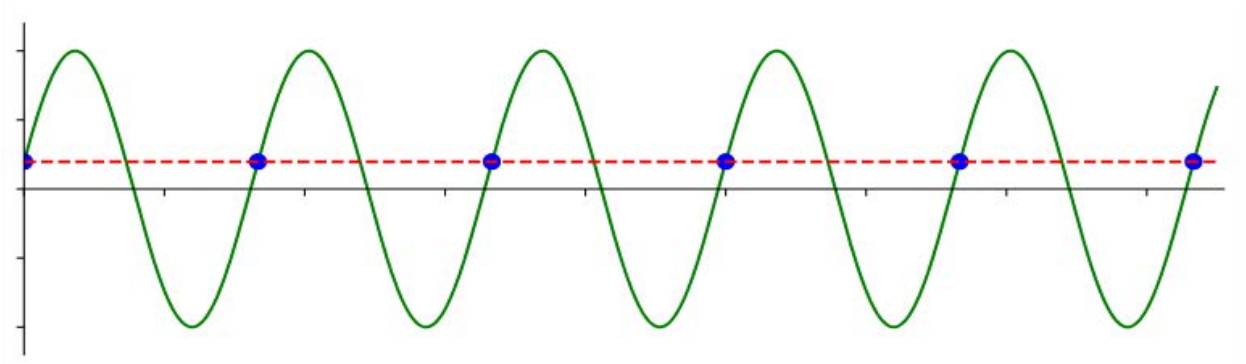
Ο αριθμός των δειγμάτων που παίρνουμε σε κάθε δευτερόλεπτο λέγεται συχνότητα δειγματοληψίας (sampling frequency ή αλλιώς sample rate).

Πχ αν έχουμε sample rate 2 Hz αυτό σημαίνει ότι σε κάθε δευτερόλεπτο εμείς παίρνουμε 2 δείγματα (samples).



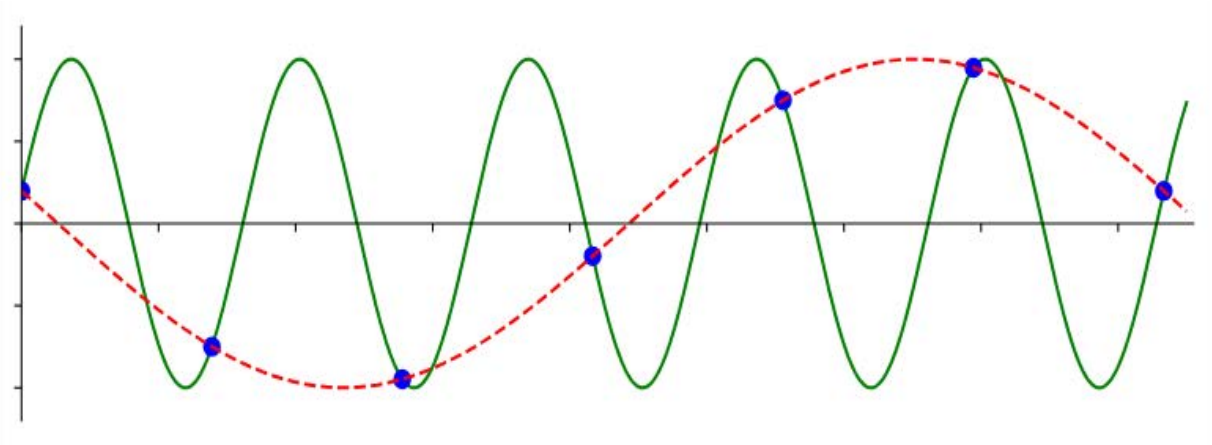
Εικόνα 15: Δειγματοληψία

Το πόσο συχνά θα παίρνουμε δείγματα από ένα σήμα είναι κάτι αρκετά σημαντικό. Έστω f η συχνότητα του S(t) και Fs η συχνότητα δειγματοληψίας.
Μπλε είναι τα δείγματα που πήραμε σε κάθε περίπτωση και με κόκκινη διακεκομμένη γραμμή το σήμα που καταφέρνουμε να φτιάξουμε (επιτυχώς ή ανεπιτυχώς) σε κάθε περίπτωση.
Αν έχουμε Fs = f τότε θα πάρουμε κάτι τέτοιο:



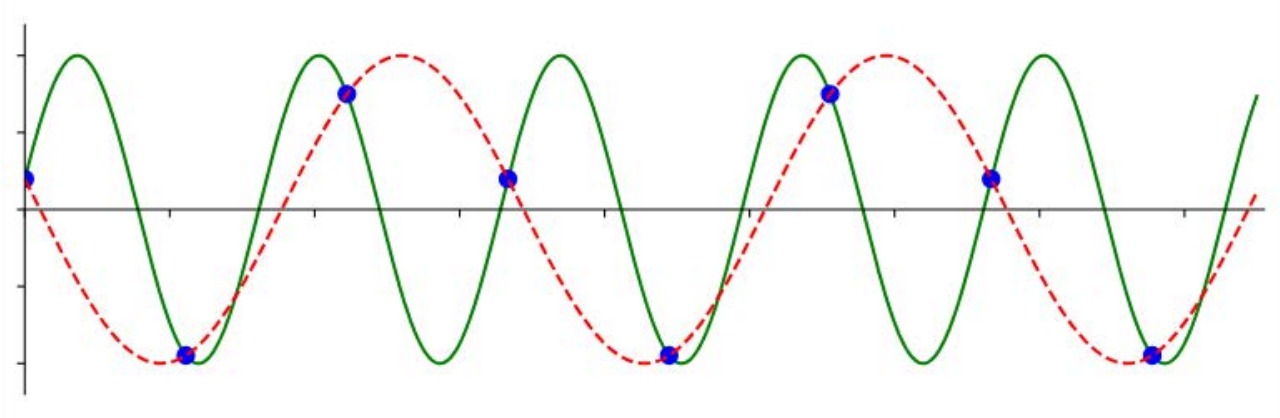
Εικόνα 16: Undersampling

Αν έχουμε Fs = 1.2f τότε πάλι δεν μπορούμε δυστυχώς με τις τιμές που έχουμε πάρει να αναπαραστήσουμε το σήμα σωστά. Αυτό φαίνεται παρακάτω:



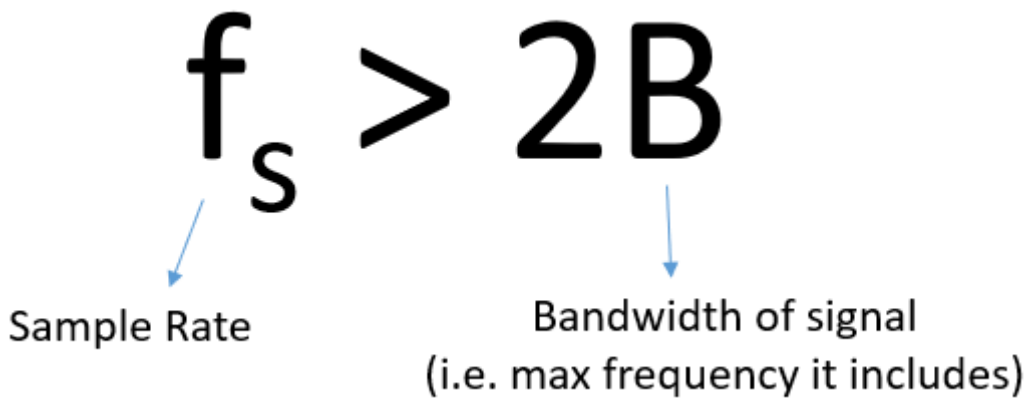
Εικόνα 17: Undersampling

Για Fs = 1.5f κάπως καλύτερα αλλά και πάλι όχι…



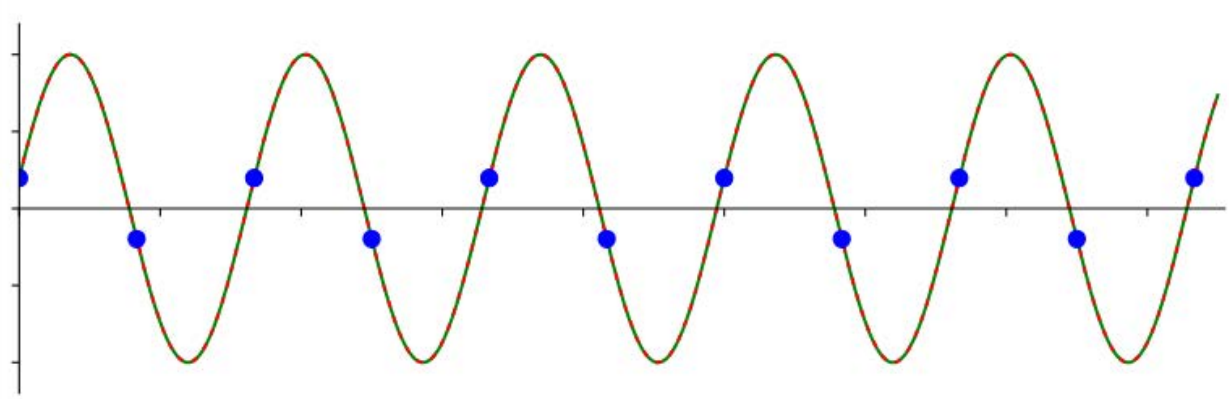
Εικόνα 18: Undersampling

Και κάπου εδώ έρχεται το θεώρημα δειγματοληψίας Nyquist-Shannon[3,8] το οποίο μας λέει ότι η συχνότητα δειγματοληψίας πρέπει να είναι τουλάχιστον διπλάσια από τη μέγιστη συχνότητα του σήματος.



Εικόνα 19: Θεώρημα δειγματοληψίας Nyquist-Shannon

Μετά την εφαρμογή του όλα δείχνουν μια χαρά..

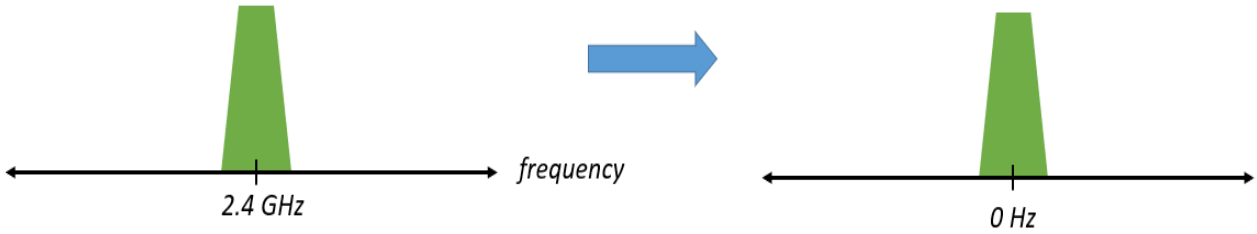


Σχήμα 20 : Δειγματοληψία που υπακούει στο θεώρημα δειγματοληψίας Nyquist - Shannon

Τα περισσότερα σήματα αποτελούνται από πολλές διαφορετικές συχνότητες, εμείς θα πρέπει να κάνουμε δειγματοληψία με τουλάχιστον διπλάσια συχνότητα της μέγιστης όλων αυτών των συχνοτήτων του σήματος.

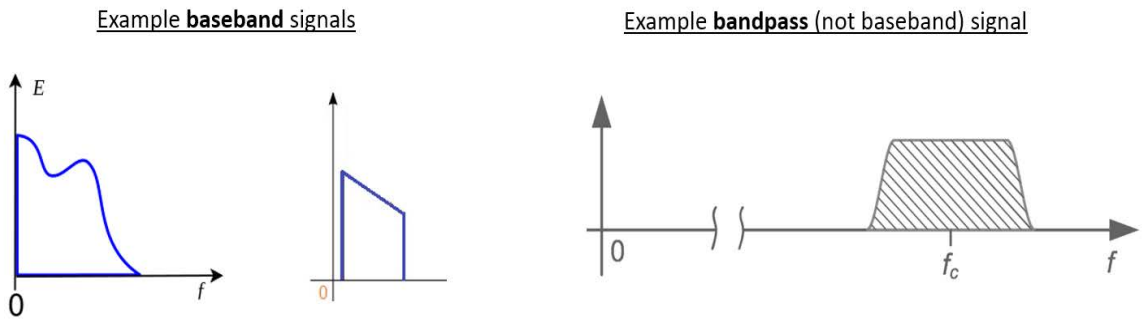
Είχαμε πει προηγουμένως ότι πρέπει να κάνουμε δειγματοληψία με sample rate τουλάχιστον διπλάσιο από τη μέγιστη συχνότητα του σήματος, όμως αν όντως το κάνουμε αυτό πχ για ένα σήμα Wi-Fi στο οποίο η συχνότητα του φέροντος θα είναι

στα 2.4 GHz τότε θα πρέπει να κάνουμε δειγματοληψία το λιγότερο στα 4.8 GHz. Για να γίνει όμως δειγματοληψία σε τόσο γρήγορο ρυθμό θα πρέπει να έχουμε στη διάθεση μας ένα πανάκριβο μηχάνημα. Το Pluto SDR για παράδειγμα υποστηρίζει sample rates εώς και 56 MHz. Αυτό που κάνουμε για να το αποφύγουμε είναι να μεταφέρουμε το σήμα μας από τα 2.4 GHz στα 0 Hz. Δηλαδή εκεί που είχαμε κεντρική συχνότητα τα 2.4 GHz τώρα πλέον θα έχουμε κεντρική συχνότητα τα 0Hz. Με δεδομένο ότι τα περισσότερα σήματα έχουν ένα εύρος συχνοτήτων (bandwidth) που κυμαίνεται από 100 kHz εώς και 40 MHz μπορούμε να κάνουμε τη δειγματοληψία μας σε πολύ χαμηλότερα sample rates. Αυτή η διαδικασία ονομάζεται downconversion και είναι κάτι το οποίο το κάνει το SDR για μας.



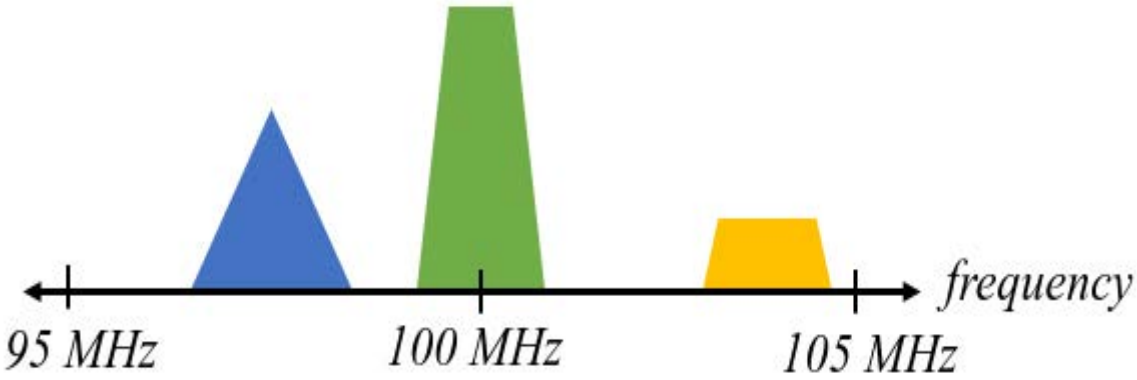
Εικόνα 21: Downconversion

Baseband σήματα είναι αυτά τα οποία έχουν κεντρική συχνότητα τα 0 Hz ή γενικά πολύ κοντά στα 0 Hz.
Bandpass σήματα είναι αυτά των οποίων η κεντρική συχνότητα βρίσκεται αρκετά μακριά από το 0Hz.



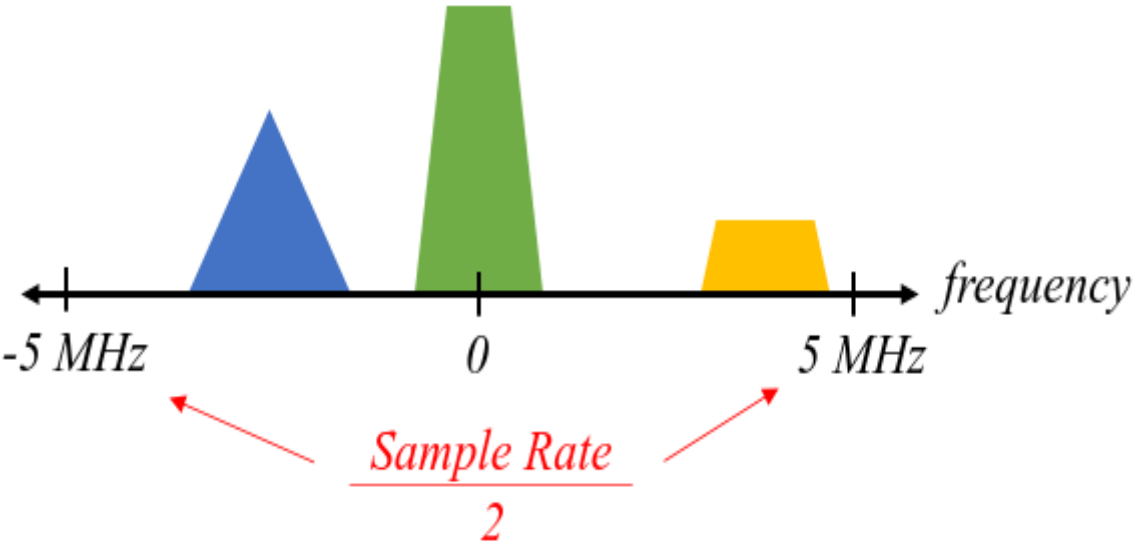
Εικόνα 22: Baseband σήματα (αριστερά), Bandpass σήματα (δεξιά)

Η δημιουργία, η επεξεργασία, η ανάλυση των σημάτων μας μας βολεύει πολύ να γίνεται στο baseband καθώς εκεί χρειαζόμαστε πολύ μικρότερα sample rates τα οποία υποστηρίζονται και από τις συσκευές μας.
Αντίστοιχα η μετάδοση θέλουμε να γίνει σε υψηλή συχνότητα για να έχουμε όσο το δυνατόν μικρότερο μήκος κύματος και κατ επέκταση μικρότερη κεραία σε μήκος για να λάβουμε το σήμα.
Το σήμα μας όμως στο πεδίο της συχνότητας αποτελείται από πολλές συχνότητες οι οποίες βρίσκονται μέσα σε κάποιο εύρος, το λεγόμενο bandwidth το οποίο ισούται με τη διαφορά της μεγαλύτερης συχνότητας του σήματος και της μικρότερης συχνότητας του σήματος. Στο κέντρο βρίσκεται η κεντρική συχνότητα και όταν το σήμα μας μεταφερθεί για επεξεργασία στην κεντρική συχνότητα 0Hz τότε όλες οι συχνότητες του σήματος που ήταν μικρότερες της κεντρικής συχνότητας τώρα θα γίνουν αρνητικές συχνότητες. Φυσικά αυτό είναι καθαρά θέμα αναπαράστασης καθώς δεν έχουν νόημα οι αρνητικές συχνότητες.
Στο παράδειγμα που ακολουθεί βλέπουμε αρχικά ένα bandpass σήμα με κεντρική συχνότητα 100 MHz και bandwidth 10 MHz με μέγιστη συχνότητα σήματος τα 105 MHz και ελάχιστη συχνότητα σήματος τα 95 MHz.



Εικόνα 23: Πριν το downconversion

Για λόγους επεξεργασίας όμως το SDR μας κάνει downconversion στο σήμα αυτό και το σήμα πλέον έχει κεντρική συχνότητα 0 Hz, το bandwidth παραμένει ίδιο, η μέγιστη συχνότητα του σήματος είναι 5 MHz και η ελάχιστη συχνότητα του σήματος είναι τα -5 MHz.



Εικόνα 24: Μετά το downconversion

Δεν υπάρχει συχνότητα -5 MHz και γενικά αρνητικές συχνότητες απλά αυτό το -5 MHz δείχνει ότι είναι μια συχνότητα μικρότερη της κεντρικής συχνότητας κατά 5 MHz.

Γενικά η διαδικασία έχει ως εξής, για να εκπέμψουμε ένα σήμα παρέχουμε στην SDR συσκευή μας τα samples τα οποία είναι μιγαδικοί αριθμοί. Ρυθμίζουμε επίσης κάποιες παραμέτρους της συσκευής όπως η κεντρική συχνότητα της μετάδοσης, το sample rate και άλλα και στη συνέχεια η SDR συσκευή μας μετατρέπει το σήμα σε bandpass. Μεταφέρει το σήμα σε μια υψηλότερη συχνότητα για να μεταδοθεί. Μόνο η μετάδοση πραγματικών αριθμών έχει νόημα, δεν υφίσταται μετάδοση μιγαδικών αριθμών.

Στην περίπτωση του δέκτη ρυθμίζουμε κάποιες παραμέτρους του SDR όπως την κεντρική συχνότητα στην οποία θα “ακούει”, το sample rate και μερικές ακόμα και αυτό λαμβάνει ένα bandpass σήμα το οποίο το κάνει downconversion και το μετατρέπει έτσι σε baseband σήμα για να το επεξεργαστούμε.[8]

4.5 ΔΙΑΜΟΡΦΩΣΗ – ΑΠΟΔΙΑΜΟΡΦΩΣΗ ΣΗΜΑΤΟΣ

Διαμόρφωση είναι η συστηματική μεταβολή κάποιου χαρακτηριστικού της (υψίσυχνης) φέρουσας κυματομορφής (φέρον) όπως πχ το πλάτος, η συχνότητα, η φάση σε συνάρτηση με το πληροφοριακό σήμα μας (το οποίο είναι σήμα χαμηλής συχνότητας).

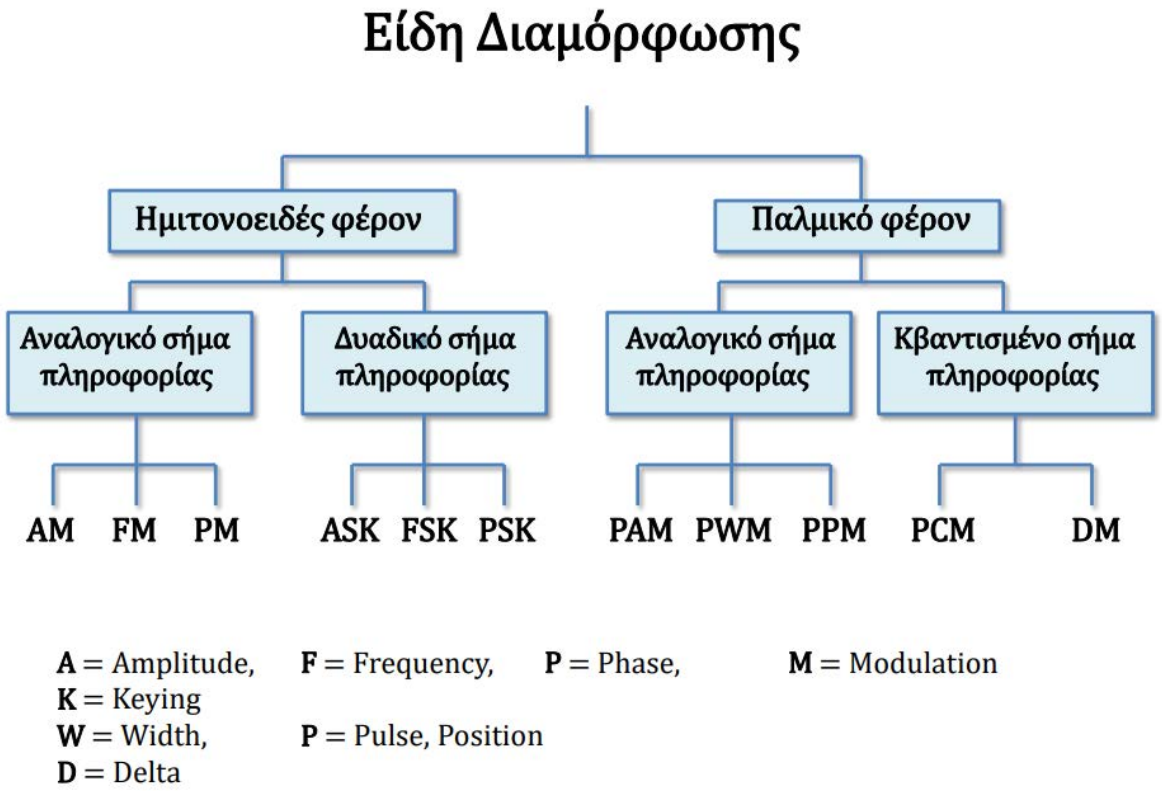
Μερικοί λόγοι για τους οποίους κάνουμε διαμόρφωση είναι οι εξής:

- Για να λάβουμε/στείλουμε ένα σήμα μέσω μιας κεραίας το μήκος της κεραίας και το μήκος κύματος του σήματος που θέλουμε να λάβουμε/εκπέμψουμε σχετίζονται καθώς η κεραία θα πρέπει να έχει μήκος λ / 2 – λ / 4 όπου λ είναι το μήκος κύματος του σήματος. Ισχύει c = λ * f και άρα όσο πιο μεγάλη είναι η συχνότητα τόσο πιο μικρό είναι το μήκος κύματος. Αν έχουμε σήματα χαμηλής συχνότητας αυτά έχουν πολύ μεγάλο μήκος κύματος συνεπώς χρειάζονται και κεραία χιλιομέτρων. Με τη διαμόρφωση μεταφέρουμε το σήμα μας σε μια περιοχή συχνοτήτων για να μπορεί να μεταδοθεί/ληφθεί πιο εύκολα.

- Πολυπλεξία
  Μεταφέρουμε το φάσμα πολλών σημάτων σε διαφορετικές φασματικές περιοχές και μπορούμε να τα μεταδώσουμε ταυτόχρονα μέσα από το ίδιο κανάλι.

- Μέσω της διαμόρφωσης μπορεί να έχει νόημα η εκχώρηση συχνοτήτων, έχουμε αδειοδοτημένες συχνότητες και ελεύθερες. Στις αδειοδοτημένες πρέπει να πληρώσεις για να πάρεις ένα κομμάτι του φάσματος και να το χρησιμοποιήσεις. Για να εκπέμψεις σε αυτό το κομμάτι που πλήρωσες και μόνο σε αυτό θα πρέπει να διαμορφώσεις το σήμα σου κατάλληλα.[8,12]

Υπάρχουν αναλογικές και ψηφιακές διαμορφώσεις και όλα τα είδη φαίνονται στην παρακάτω εικόνα:



Εικόνα 25: Είδη διαμόρφωσης

QUADRATURE SAMPLING

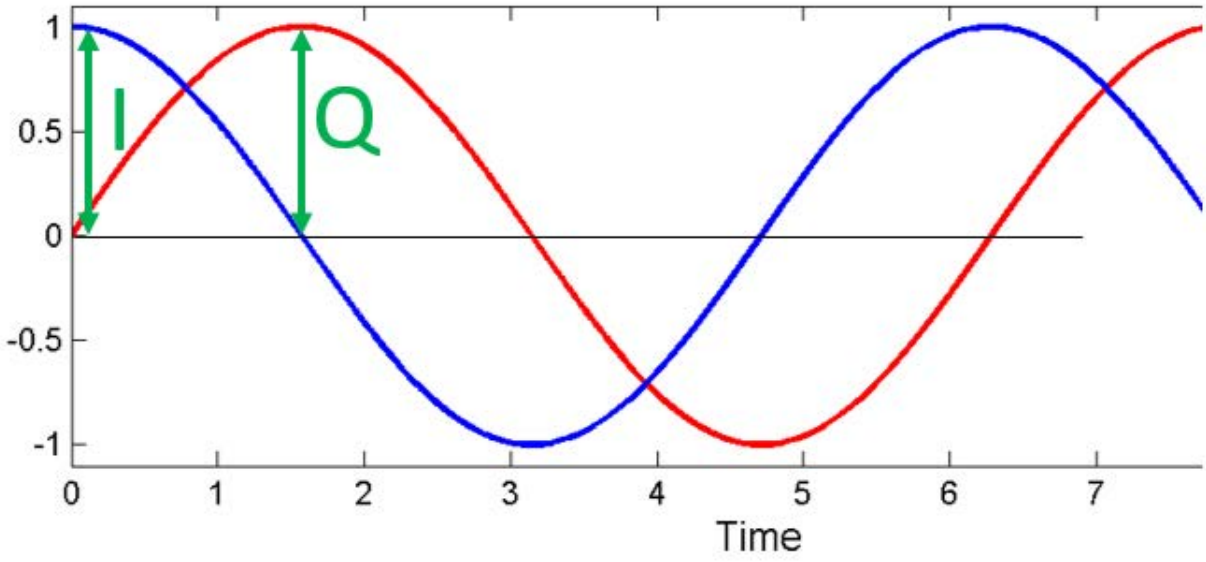
Επειδή αντιμετωπίζουμε πρόβλημα να διακρίνουμε τη φάση των σημάτων που λαμβάνουμε (πχ cos(x) ή cos(-x)) αλλά και τις κορυφές τους χρησιμοποιούμε τα quadrature signals.

Κάθε σήμα που θα φτάνει ή θα φεύγει από την SDR συσκευή μας στην πραγματικότητα θα είναι άθροισμα δύο σημάτων. Αυτά τα σήματα θα έχουν μεταξύ τους διαφορά φάσης 90 μοιρών γι αυτό και λέγονται quadrature σήματα. Στην περίπτωση μας θα έχουμε ένα ημίτονο κι ένα συνημίτονο.

Icos(2πft)

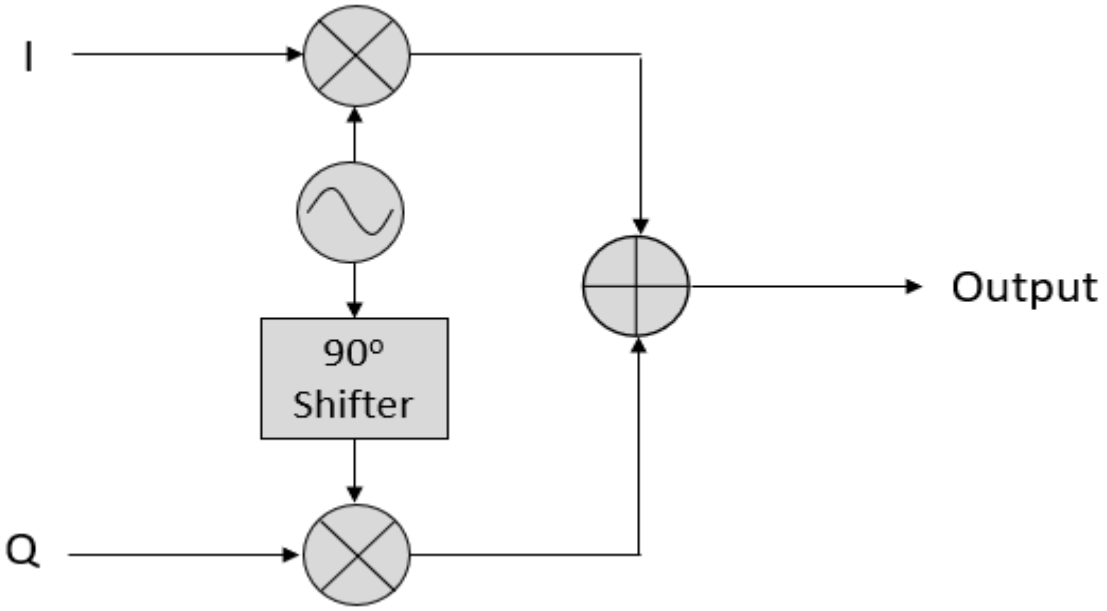
Qsin(2πft)

όπου I είναι το πλάτος του συνημιτόνου (προκύπτει από το “in phase”) και Q είναι το πλάτος του ημιτόνου (προκύπτει από το “quadrature”).



Εικόνα 26: Quadrature signals

Έστω ότι θέλουμε να κάνουμε transmit αυτό το σήμα: $x(t) = A\cos(2\pi ft + \varphi)$.
Λόγω της τριγωνομετρικής ιδιότητας $\cos(a+b) = \cos(a)\cos(b) - \sin(a)\sin(b)$
το σήμα μας γράφεται $x(t) = A\cos(\varphi)\cos(2\pi ft) – A\sin(\varphi)\sin(2\pi ft)$.
Το $I = A\cos(\varphi)$ και το $Q = A\sin(\varphi)$ άρα το σήμα μας γίνεται κάπως έτσι :
$x(t) = I\cos(2\pi ft) - Q\sin(2\pi ft)$

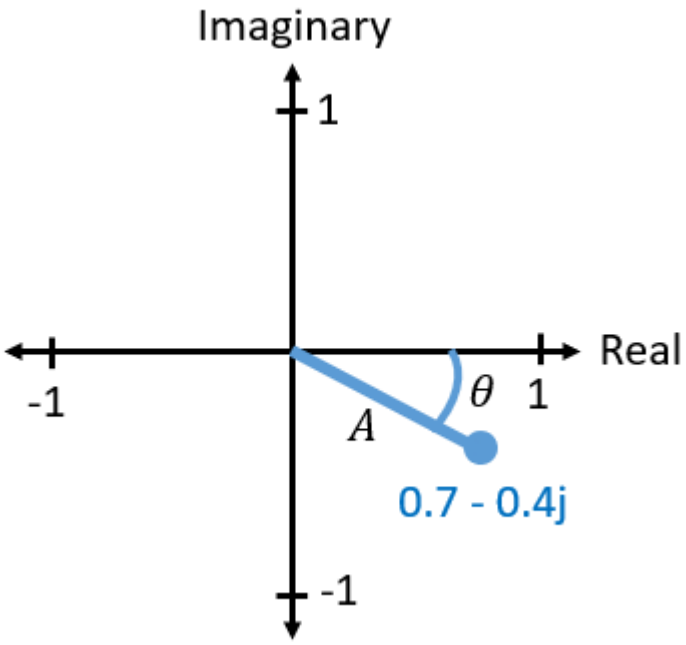


Εικόνα 27: Το άθροισμα των I και Q σημάτων μας δίνει το output

Προσθέτοντας ένα ημίτονο και ένα συνημίτονο παίρνουμε ένα άλλο ημιτονοειδές σήμα x(t) με διαφορετικό πλάτος και φάση, και μάλιστα απλά αλλάζοντας τα πλάτη I και Q μπορούμε να ελέγχουμε το πλάτος και τη φάση του x(t).

Και ακριβώς εδώ κολλάνε οι μιγαδικοί αριθμοί (complex numbers).

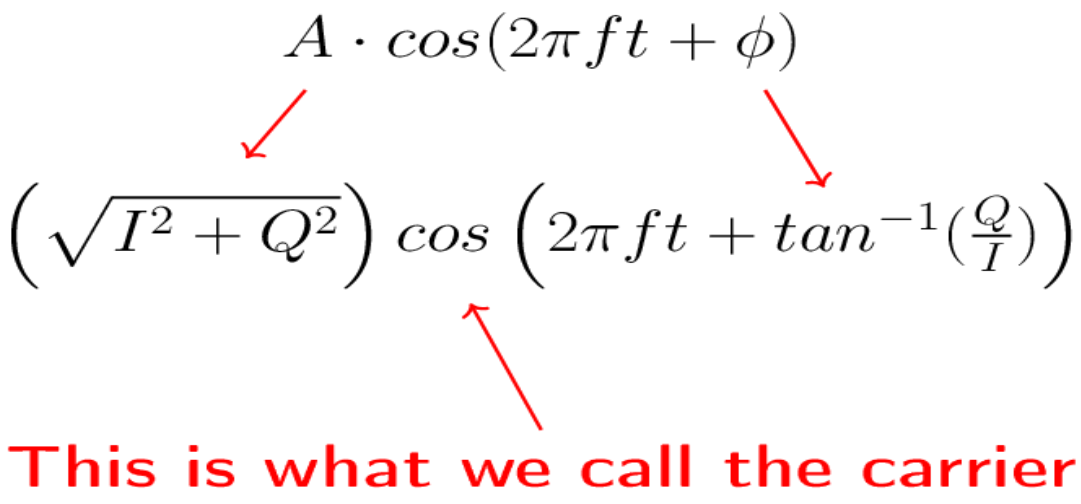
Οι αριθμοί αυτοί είναι της μορφής a + bj για παράδειγμα 0.7 – 0.4j .
Είναι δύο αριθμοί μαζί, το πραγματικό μέρος και το φανταστικό μέρος, στο παράδειγμα μας το πραγματικό μέρος είναι το 0.7 ενώ το φανταστικό μέρος είναι το -0.4 .
Πλάτος είναι η απόσταση του σημείου A(a,b) από το σημείο O(0,0), ενώ φάση είναι η γωνία που σχηματίζει το διάνυσμα OA με τον άξονα των πραγματικών αριθμών.
Στις παρακάτω εικόνες φαίνεται το διάγραμμα του μιγαδικού αριθμού μας και ο τρόπος υπολογισμού του πλάτους και της φάσης για το παράδειγμα μας.



Εικόνα 28: Πλάτος και φάση μιγαδικών αριθμών

$$\text{magnitude} = \sqrt{a^2 + b^2} = 0.806$$
$$\text{phase} = \tan^{-1}\left(\frac{b}{a}\right) = -29.7^\circ = -0.519 \quad \text{radians}$$

Στο quadrature sampling το I είναι το πραγματικό μέρος ενώ το Q είναι το φανταστικό μέρος. Έτσι αν θέλαμε να κάνουμε transmit αυτό το sample (0. 7 - 0.4j) τότε το σήμα που θα στέλναμε θα ήταν το x(t) = 0.7cos(2πft) – 0.4sin(2πft).
Από την ιδιότητα acos(x) + bsin(x) = Acos(x – φ) με A το πλάτος και φ την φάση η εξίσωση τελικά γίνεται x(t) = 0.806cos(2πft + 0.519).
Η συχνότητα f είναι η κεντρική συχνότητα του σήματος που στέλνουμε στον αέρα δηλαδή η συχνότητα του ηλεκτρομαγνητικού κύματος.
Το σήμα x(t) = 0.806cos(2πft + 0.519) καλείται φέρον σήμα γιατί είναι αυτό που ουσιαστικά “κουβαλάει” το σήμα μας σε κάποια RF συχνότητα.



Εικόνα 29: Carrier signal

Αλλάζοντας εμείς τιμές στα I και Q αλλάζουμε το πλάτος και τη φάση του φέροντος σήματος. Μπορούμε επίσης να αλλάξουμε και τη συχνότητα του όπως γίνεται στο FM.

Διαμορφώνουμε δηλαδή το φέρον σήμα.

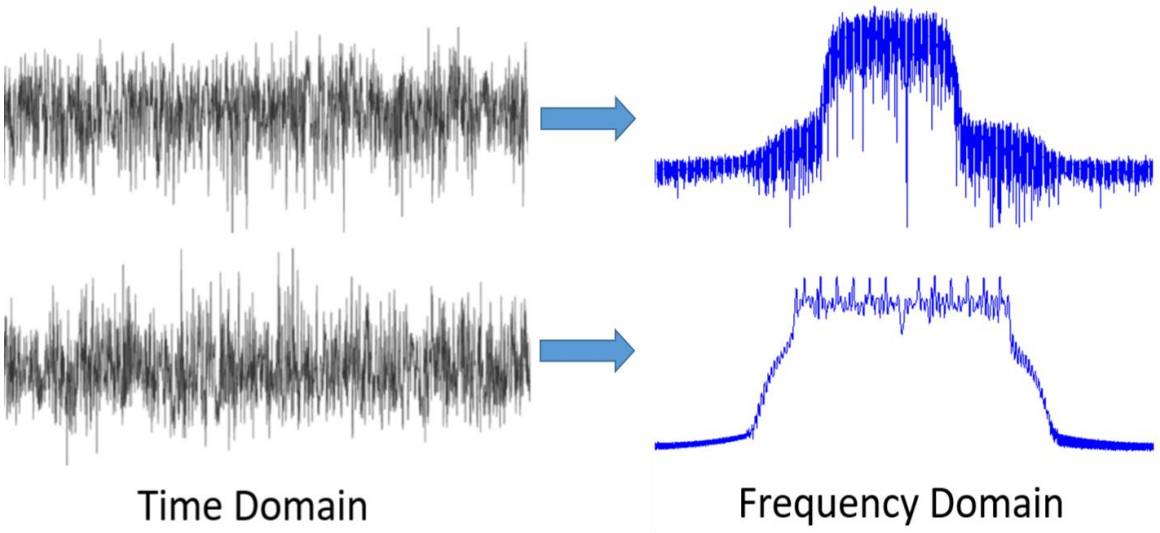
Για παράδειγμα αν θέλαμε να μεταδώσουμε το sample 1 + 0j και το sample 0 +1j θα στέλναμε cos(2πft) και sin(2πft) άρα θα είχαμε αλλαγή της φάσης του φέροντος κατά 90 μοίρες μεταξύ αυτών των δύο δειγμάτων.[8,13]

ΚΕΦΑΛΑΙΟ 5 Μετασχηματισμός Fourier

5.1 Πεδίο χρόνου και πεδίο συχνότητας

Τώρα θα δούμε με δύο πολύ σημαντικές έννοιες στην ανάλυση σημάτων και συστημάτων, το πεδίο του χρόνου και το πεδίο της συχνότητας.

Το πεδίο του χρόνου αναπαριστά πως αλλάζει ένα σήμα στο χρόνο, βλέπουμε έτσι το πλάτος του σήματος σαν συνάρτηση του χρόνου και αυτό μας επιτρέπει να κάνουμε τις παρατηρήσεις μας για τα χρονικά διαστήματα που μας ενδιαφέρουν. Το πεδίο της συχνότητας από την άλλη αναλύει το σήμα στις συχνότητες από τις οποίες αποτελείται και μας δείχνει πόση ένταση έχει το σήμα σε κάθε μία από αυτές. Κάθε σήμα μπορεί να γραφτεί σαν άθροισμα πολλών ημιτονοειδών σημάτων, κάποια από αυτά χρειάζονται πιο πολλά ημιτονοειδή σήματα για να αναπαρασταθούν, άλλα λιγότερα, κι άλλα άπειρα. Καθένα από αυτά έχει το δικό του πλάτος, φάση και συχνότητα.
Ο τρόπος με τον οποίο πηγαίνουμε από το πεδίο του χρόνου στο πεδίο της συχνότητας είναι ο μετασχηματισμός Fourier. Ο αντίστροφος μετασχηματισμός Fourier μας πηγαίνει από το πεδίο της συχνότητας στο πεδίο του χρόνου. [8]



Εικόνα 30: Πεδίο χρόνου και πεδίο συχνότητας

5.2 Μετασχηματισμός Fourier

Για ένα σήμα x(t) μπορούμε να πάμε στο πεδίο της συχνότητας μέσω του παρακάτω τύπου:

$$X(f) = \int x(t) e^{-j2\pi ft} dt$$

Το σήμα μας στο πεδίο του χρόνου είναι το x(t) και στο πεδίο της συχνότητας είναι το X(f), t ο χρόνος και f η συχνότητα αντίστοιχα. Το j είναι η φανταστική μονάδα.

Ο αντίστροφος μετασχηματισμός Fourier δίνεται από τον παρακάτω τύπο:

$$x(t) = \frac{1}{2\pi} \int X(f) e^{j2\pi ft} df$$

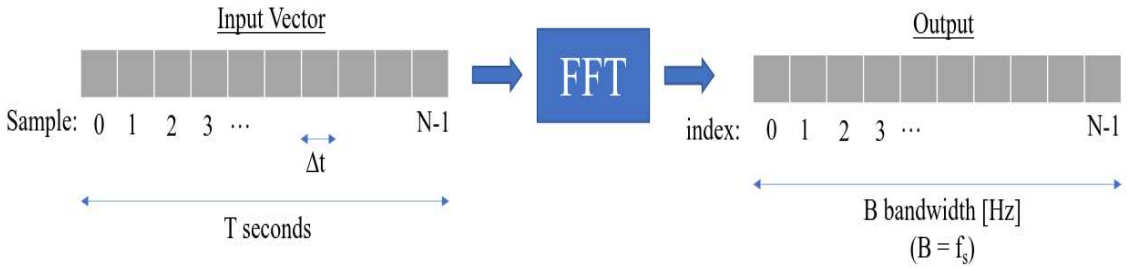
Αυτοί οι τύποι είναι για τη συνεχή μορφή του μετασχηματισμού Fourier, εμείς θα χρειαστούμε τη διακριτή του μορφή η οποία είναι η εξής :

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{j2\pi}{N} kn}$$

Η κύρια διαφορά είναι πως εδώ αντί για ολοκλήρωμα έχουμε άθροισμα.
Στα πλαίσια αυτής της εργασίας θα χρησιμοποιήσουμε τον αλγόριθμο του Fast Fourier Transform (FFT) ο οποίος είναι μια υλοποίηση του διακριτού μετασχηματισμού Fourier και θα μας υπολογίσει το μετασχηματισμό Fourier των σημάτων μας.
Ο FFT είναι μια συνάρτηση με μία είσοδο και μία έξοδο που μετατρέπει το σήμα από το πεδίο του χρόνου στο πεδίο της συχνότητας.

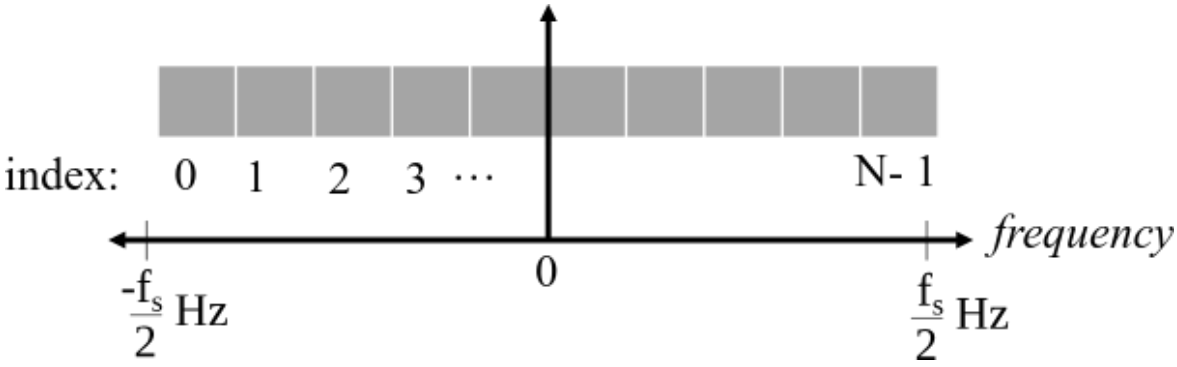
Εικόνα 31: Από το πεδίο του χρόνου στο πεδίο της συχνότητας μέσω FFT

Εμείς δίνουμε ως είσοδο στη συνάρτηση FFT ένα διάνυσμα από δείγματα (samples) και θα πάρουμε ως έξοδο το ίδιο διάνυσμα από δείγματα στο πεδίο της συχνότητας όμως. Η είσοδος και η έξοδος έχουν ακριβώς το ίδιο μέγεθος πχ αν δώσουμε στην είσοδο 1024 δείγματα τότε θα πάρουμε στην έξοδο 1024 δείγματα.



Εικόνα 32: Είσοδος→ FFT→ Έξοδος

Τώρα που έχουμε εξηγήσει πια τι είναι το sample rate είναι η κατάλληλη στιγμή να αναφερθούμε στο τι ακριβώς μας επιστρέφει ένα σκανάρισμα με το Pluto SDR σε κάποια συγκεκριμένη κεντρική συχνότητα με κάποια συχνότητα δειγματοληψίας (f_s). Το output που θα πάρουμε έχει άμεση σχέση με το sample rate καθώς το διάστημα συχνοτήτων στο οποίο αναφέρεται εκτείνεται πάντα από [- f_s / 2, f_s / 2]. Αυτό φαίνεται και στο παρακάτω σχήμα.



Εικόνα 33: Συχνότητες που θα πάρουμε ως output από σκανάρισμα με συχνότητα δειγματοληψίας f_s

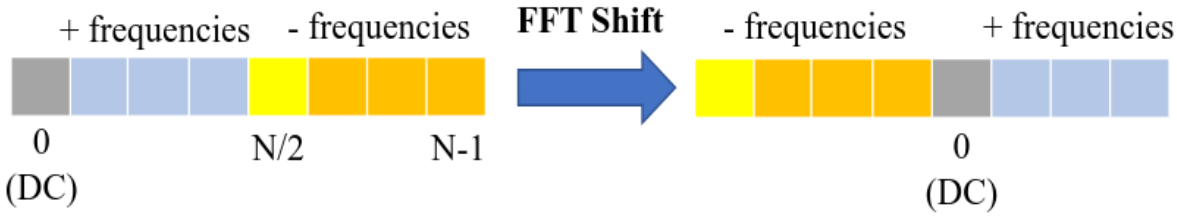
Αυτά τα δείγματα που πήραμε μόλις από το σκανάρισμα θα τα δώσουμε σαν είσοδο στο μετασχηματισμό Fourier . Δώσαμε ως είσοδο λοιπόν έναν αριθμό από

samples στο πεδίο του χρόνου και αυτός θα μας επιστρέψει τον ίδιο αριθμό από samples αλλά στο πεδίο της συχνότητας. Όλα αυτά τα samples όμως αναφέρονται στο διάστημα συχνοτήτων [- f_s / 2, f_s / 2].
Αν N ο αριθμός των δειγμάτων στην είσοδο τότε όσο κι αν αυξήσουμε το N πάλι τις ίδιες συχνότητες θα βλέπουμε στην έξοδο αφού το εύρος συχνοτήτων εξαρτάται μόνο από το sample rate. Αυτό που θα κερδίσουμε σε αυτή την περίπτωση είναι να πάρουμε μια καλύτερη αναπαράσταση του σήματος στο πεδίο της συχνότητας αφού το κάθε bin εκεί θα περιέχει f_s / N συχνότητες.

Πιο συγκεκριμένα η έξοδος του FFT λοιπόν θα μας δώσει κάτι σαν αυτό :

Εικόνα 34 : Έξοδος FFT

Αυτό που θα θέλαμε εμείς όμως είναι να έχουμε στο κέντρο την κεντρική συχνότητα (0 Hz), αριστερά της τις αρνητικές συχνότητες και δεξιά της τις θετικές συχνότητες. Έτσι θα πρέπει να χρησιμοποιήσουμε μία άλλη συνάρτηση την FFT Shift για να γίνει αυτή η ανακατάταξη των συχνοτήτων.



Εικόνα 35: FFTshift

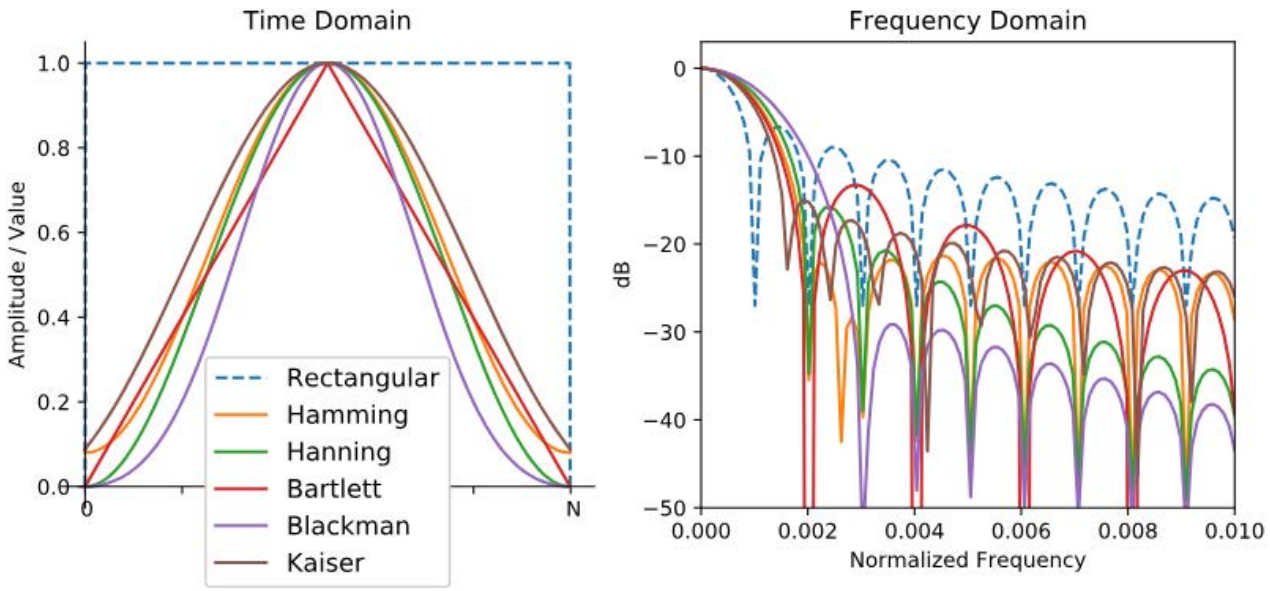
Στην Python η συνάρτηση για την FFT Shift είναι η numpy.fft.fftshift() και βρίσκεται ορισμένη στη βιβλιοθήκη NumPy. [8,13]

5.3 FFT SIZE

Το μέγεθος του FFT είναι ο αριθμός των bins που χρησιμοποιούμε για να διαιρέσουμε το διαθέσιμο φάσμα σε ίσα μέρη. Για παράδειγμα αν έχουμε μια περιοχή συχνοτήτων της τάξεως των 20MHz τότε αν χρησιμοποιήσουμε 1024 bins τότε θα έχουμε 20MHz / 1024 = 0.019MHz ανά bin, ενώ αν χρησιμοποιούσαμε 56 bins τότε θα είχαμε 20MHz / 56 = 0.35MHz ανά bin.
Γενικά προτιμάμε να είναι δύναμη του 2 το FFTSize, αν δεν είναι ίσως να είναι λίγο χειρότερα τα πράγματα από άποψη ταχύτητας στους υπολογισμούς. Δεν είναι απαραίτητο να κάνουμε μετασχηματισμό Fourier σε όλα τα samples που παίρνουμε, αρκεί ένα μέρος τους. Για παράδειγμα αν λαμβάνουμε 100 χιλιάδες δείγματα είναι μια χαρά να κάνουμε στα 1024 από αυτά FFT.
Επιλέγοντας ένα μεγάλο FFTSize μας δίνει καλύτερη αναπαράσταση του σήματος μας στο πεδίο της συχνότητας και τα επίπεδα του θορύβου είναι μειωμένα όμως χρειαζόμαστε μεγαλύτερο χρόνο επεξεργασίας των σημάτων.

5.4 WINDOWING

Όταν κάνουμε FFT για να πάρουμε την έκδοση του σήματος μας στο πεδίο της συχνότητας ο FFT περιμένει ότι του δώσαμε ένα περιοδικό σήμα, δηλαδή περιμένει ότι το πρώτο και το τελευταίο sample που του δώσαμε συνδέονται μεταξύ τους, έχουν δηλαδή ίδια ή περίπου ίδια τιμή. Στην περίπτωση μας αυτό δε συμβαίνει γι αυτό και χρησιμοποιούμε μία συνάρτηση window. Πριν λόιπόν κάνουμε FFT πολλαπλασιάζουμε τα samples που έχουμε πάρει με μία τέτοια συνάρτηση. Αυτό που κάνει αυτή η συνάρτηση είναι να δίνει την τιμή 0 στο πρώτο και στο τελευταίο δείγμα μας και έτσι λύνεται το συγκεκριμένο πρόβλημα. Υπάρχουν αρκετές τέτοιες συναρτήσεις μερικές από τις οποίες είναι οι Hamming, Hanning, Blackman, Kaiser. Οι διαφορές τους είναι πολύ μικρές, εμείς εδώ στην εργασία θα χρησιμοποιήσουμε τη Hamming window συνάρτηση στην Python numpy.hamming(N) όπου N το μέγεθος του πίνακα με τα samples μας άρα το FFTSize μας. [8]



Εικόνα 36: Συναρτήσεις Windowing

5.5 Power Spectral Density

Η Power Spectral Density είναι ένα πολύ σημαντικό εργαλείο για την οπτικοποίηση των σημάτων στο πεδίο της συχνότητας. Ουσιαστικά μας δείχνει πως κατανέμεται η ισχύς ενός σήματος σε όλες τις συχνότητες που το αποτελούν. Ο αντίστοιχος όρος στα ελληνικά είναι φασματική πυκνότητα ισχύος. Ο τρόπος υπολογισμού της Power Spectral Density παρουσιάζεται παρακάτω:

- Αρχικά πολλαπλασιάζουμε τα samples τα οποία λάβαμε με μία συνάρτηση παραθύρου (window function) για να έχουν το πρώτο και το τελευταίο από τα δείγματα μας περίπου ίδια ή ίδια τιμή.
- Στη συνέχεια, παίρνουμε τον μετασχηματισμο Fourier των samples που έχουμε. Αν έχουμε 1024 samples, τότε το FFTsize θα είναι 1024. Στην έξοδο θα λάβουμε 1024 complex floats.
- Μετά υπολογίζουμε το πλάτος της εξόδου του μετασχηματισμού Fourier, το οποίο θα μας δώσει 1024 real floats.
- Υπολογίζουμε το τετράγωνο του παραπάνω υπολογισμού το οποίο θα μας δώσει την ισχύ.
- Κανονικοποιούμε διαιρώντας με το γινόμενο του FFTsize και του sample rate.

- Μετατρέπουμε σε λογαριθμική κλίμακα (σε dBs) χρησιμοποιώντας το $10\log_{10}$.
- Τέλος, κάνουμε FFTshift για να διορθώσουμε την κατανομή των συχνοτήτων. Να θυμίσουμε ότι οι συχνότητες στην έξοδο του μετασχηματισμού Fourier βγαίνουν με την εξής σειρά: πρώτα η μηδενική συχνότητα, μετά οι θετικές συχνότητες και στο τέλος οι αρνητικές. Αυτό που θέλουμε και αυτό που πετυχαίνουμε με το fftshift είναι να έχουμε αρχικά τις αρνητικές, μετά το 0 και στο τέλος τις θετικές συχνότητες. [8]

ΚΕΦΑΛΑΙΟ 6 Φασματικός αναλυτής με χρήση SDR συσκευής

6.1 Τι είναι οι φασματικοί αναλυτές (spectrum analyzers);

Οι φασματικοί αναλυτές είναι εργαλεία που λειτουργούν στο πεδίο της συχνότητας και πιο συγκεκριμένα οπτικοποιούν την ισχύ των σημάτων σε σχέση με την συχνότητα. Με λίγα λόγια μας δείχνουν την εικόνα του φάσματος σε μία επιλεγμένη περιοχή συχνοτήτων για κάποια συγκεκριμένη χρονική στιγμή.

6.2 Χρησιμότητα

Οι φασματικοί αναλυτές είναι πάρα πολύ χρήσιμοι στις ασύρματες επικοινωνίες. Μας δείχνουν την ισχύ των σημάτων που υπάρχουν σε μία περιοχή συχνοτήτων που ζητήσαμε για κάποια συγκεκριμένη χρονική στιγμή. Μπορούμε να διαπιστώσουμε λοιπόν εύκολα αν υπάρχει κάποια άλλη μετάδοση σε αυτή την περιοχή συχνοτήτων. Έτσι μπορούμε να γλυτώσουμε από παρεμβολές ή ταυτόχρονες μεταδόσεις. Όλα αυτά θα είχαν ως αποτέλεσμα προβλήματα στην επικοινωνία και γενικότερα κακή ποιότητα επικοινωνίας.

6.3 Είσοδοι του προγράμματος

Το πρόγραμμα που έχουμε ετοιμάσει για τα πλαίσια αυτής της εργασίας είναι γραμμένο σε γλώσσα Python. Αξιοποιεί τις δυνατότητες που μας δίνει η SDR συσκευή μας, το Pluto SDR.
Το πρόγραμμα αποτελείται από 3 modules, το κύριο module ονομάζεται pluto_spectrum_analyzer.py , και τα άλλα δύο create_table.py και connection_manager.py έχουν να κάνουν με την σύνδεση του προγράμματος με μία βάση δεδομένων και την αποθήκευση δεδομένων στη βάση αυτή.
Οι είσοδοι του προγράμματος δίνονται από την γραμμή εντολών μέσω ορισμάτων κάνοντας χρήση του ArgParser.

Συγκεκριμένα ο χρήστης υποχρεωτικά πρέπει να δώσει ως είσοδο μία αρχική συχνότητα και ένα εύρος συχνοτήτων στο οποίο θα γίνει η ανάλυση του φάσματος. Η αρχική συχνότητα είναι σε MHz και έχει ένα εύρος αποδεκτών τιμών από 70 MHz εώς και 5995 MHz. Αυτός ο περιορισμός έχει να κάνει με την SDR συσκευή η οποία μπορεί να λάβει / μεταδώσει σήματα από 70 MHz εώς 6 GHz.
Για παράδειγμα μία είσοδος θα μπορούσε να είναι η εξής αν θέλαμε να σκανάρουμε από τα 2400 MHz έως τα 2430 MHz θα πληκτρολογούσαμε στην γραμμή εντολών python3 pluto_spectrum_analyzer.py -f 2400 -b 30 .
Από εκεί και πέρα έχουμε και προαιρετικά ορίσματα, το μέγεθος καναλιού (channel bandwidth), ο χρόνος σκαναρίσματος, ο αριθμός των περασμάτων και το μέγεθος του μετασχηματισμού Fourier.
Πιο αναλυτικά:

Μέγεθος καναλιού (channel bandwidth)

Το εύρος συχνοτήτων που ζητάει ο χρήστης από το πρόγραμμα μας να ελεγχθεί θα το μοιράσουμε σε ίσα κομμάτια, τα οποία είναι τα κανάλια μας. Αυτό είναι το μέγεθος καναλιού. Κι εδώ λόγω περιορισμών του Pluto SDR λόγω του USB 2.0 για τα πλαίσια αυτής της εργασίας θα χρησιμοποιήσουμε sample rates εώς 10 MHz.
Επίσης το χαμηλότερο sample rate που μπορούμε να χρησιμοποιήσουμε στο Pluto SDR είναι τα 521 kHz. Έτσι τα αποδεκτά κανάλια που μπορούμε να χρησιμοποιήσουμε θα είναι από 0.53 MHz εώς 10 MHz.
Η επιλογή μεγέθους για το κανάλι μας είναι προαιρετική, αν ο χρήστης δε ζητήσει από το πρόγραμμα κάτι συγκεκριμένο τότε επιλέγεται το μεγαλύτερο δυνατό κανάλι που εξυπηρετεί την κάθε περίσταση.
Αν θέλαμε στο προηγούμενο παράδειγμα εισόδου να προσθέσουμε μέγεθος καναλιού 5 MHz τότε θα γράφαμε στην γραμμή εντολών
python3 pluto_spectrum_analyzer.py -f 2400 -b 30 -c 5 .
Αν το αφήναμε όπως πιο πάνω χωρίς να επιλέξουμε κανάλι (python3 pluto_spectrum_analyzer.py -f 2400 -b 30) τότε θα γινόταν το σκανάρισμα μας με κανάλι 10 MHz.

Χρόνος σκαναρίσματος (καθαρός χρόνος sensing)

Ο χρήστης αν το θέλει έχει τη δυνατότητα να ορίσει για πόσο χρόνο θέλει να σκανάρουμε το εύρος συχνοτήτων το οποίο θέλει να ελέγξουμε. Είναι και αυτό ένα προαιρετικό όρισμα, η τιμή που θα δώσει ο χρήστης είναι σε δευτερόλεπτα. Η συγκεκριμένη χρονική τιμή αφορά χρόνο sensing.

Αν δεν δώσει κάποιο χρόνο ο χρήστης στην γραμμή εντολών τότε το ζητούμενο εύρος συχνοτήτων θα σκαναριστεί τέσσερις φορές ανεξάρτητα από το πόσο χρόνο θα πάρει, καθώς δεν έχουμε κάποιο χρονικό περιορισμό.

Στο παράδειγμα μας αν θέλαμε να σκανάρουμε για 10 δευτερόλεπτα το ζητούμενο εύρος συχνοτήτων πολύ απλά θα γράφαμε στην γραμμή εντολών python3 pluto_spectrum_analyzer.py -f 2400 -b 30 -t 10 .

Μέγιστος χρόνος σκαναρίσματος είναι τα 60 δευτερόλεπτα. Το μέγιστο όριο είναι τυπικό, βάλαμε απλώς μία αρκετά μεγάλη τιμή. Ενδεικτικά σε ένα σκανάρισμα σε 2-3 δευτερόλεπτα μπορεί να έχουμε 30-40 περάσματα. Είναι ήδη πολλά για αν έχουμε ένα καλό αποτέλεσμα. Σε ενδεχόμενο σκανάρισμα με περιορισμό χρόνου και επαναλήψεων τα όρια αυτά ενδέχεται να ξεπεραστούν μέχρι να ικανοποιηθεί ένα από τα δύο κριτήρια τερματισμού.

Αριθμός περασμάτων

Μία άλλη δυνατότητα που μας δίνει το πρόγραμμα είναι να επιλέξουμε τον αριθμό των περασμάτων, δηλαδή πόσες φορές θα σκανάρουμε το ζητούμενο εύρος συχνοτήτων. Συγκεκριμένα όταν ο χρήστης ζητάει να σκανάρει το πρόγραμμα τις συχνότητες από 2400 MHz έως 2430 MHz ο έλεγχος αυτού του εύρους συχνοτήτων θα γίνει τέσσερις φορές, θα βγει ένας μέσος όρος και αυτό θα είναι και το αποτέλεσμα που θα δει στην οθόνη. Αν ο χρήστης δεν επιθυμεί κάτι άλλο τότε το σκανάρισμα θα αποτελείται από τέσσερα περάσματα. Αν επιθυμεί περισσότερα ή λιγότερα τότε θα πρέπει να δώσει μία είσοδο της μορφής

python3 pluto_spectrum_analyzer.py -f 2400 -b 30 -i 10 .

Σε αυτή την περίπτωση θα κάνει δέκα φορές τον έλεγχο του ζητούμενου εύρους συχνοτήτων.

Μέγιστος αριθμός περασμάτων στο πρόγραμμα είναι τα 300 περάσματα και ελάχιστος το 1 πέρασμα. Το άνω όριο είναι τυπικό, βάλαμε απλώς μία μεγάλη τιμή, 15 περάσματα είναι ήδη αρκετά πόσο μάλλον τα 300. Σε ενδεχόμενο

σκανάρισμα με περιορισμό χρόνου και επαναλήψεων τα όρια αυτά ενδέχεται να ξεπεραστούν μέχρι να ικανοποιηθεί ένα από τα δύο κριτήρια τερματισμού.

FFTsize

Το πρόγραμμα υποστηρίζει και διαφορετικά FFTsize (μεγέθη μετασχηματισμού Fourier). Το FFTsize είναι ουσιαστικά ο αριθμός των bins που παράγονται από τον μετασχηματισμό Fourier. Για κάποια δεδομένη τιμή καναλιού αύξηση του FFTsize θα μας δώσει καλύτερη αναπαράσταση του σήματος αλλά χρειάζεται περισσότερο χρόνο για επεξεργασία. Το πρόγραμμα μας υποστηρίζει FFTsize από 32 εώς 32768.

Αν το FFTsize είναι δύναμη του 2 τότε αυτό είναι πιο γρήγορο σε σχέση με κάποιο αριθμό που δεν είναι. Σε αυτό το πρόγραμμα λοιπόν στην περίπτωση που ο χρήστης θελήσει κάποιο συγκεκριμένο FFTsize και το οποίο δεν είναι δύναμη του 2 τότε το μετατρέπουμε στην αμέσως μεγαλύτερη δύναμη του 2. Default FFTsize για το πρόγραμμά μας είναι τα 8192 bins άρα αν ο χρήστης δεν δώσει κάτι διαφορετικό από την γραμμή εντολών με αυτό θα γίνουν οι υπολογισμοί μας.

Στο παράδειγμα μας αν θέλαμε να έχουμε FFTsize 1024 τότε θα τρέχαμε το πρόγραμμα με αυτόν τον τρόπο python3 pluto_spectrum_analyzer.py -f 2400 -b 30 -s 1024 .

Συγκεντρωτικά τα παρακάτω είναι τα ορίσματα που μπορεί κανείς να χρησιμοποιήσει για είσοδο στο πρόγραμμα:

-f τιμή αρχική συχνότητα (υποχρεωτικό)
-b τιμή εύρος συχνοτήτων (υποχρεωτικό)
-c τιμή μέγεθος καναλιού (προαιρετικό)
-t τιμή καθαρός χρόνος sensing (προαιρετικό)
-i τιμή αριθμός περασμάτων (προαιρετικό)
-s τιμή FFTsize (προαιρετικό)

Φυσικά όλα αυτά τα ορίσματα μπορούν να συνδυαστούν ταυτόχρονα σε κάποια πιθανή είσοδο του προγράμματος.

Ένα παράδειγμα είναι το εξής:

python3 pluto_spectrum_analyzer.py -f 2400 -b 30 -c 5 -i 15 -t 40 -s 1024

Εδώ ο χρήστης ζητάει από το πρόγραμμα να σκανάρει το διάστημα 2400 - 2430 MHz χωρίζοντας το σε κανάλια των 5 MHz με FFTsize 1024. Ο έλεγχος θέλουμε να γίνει 15 φορές ενώ έχουμε και χρονικό περιορισμό 40 δευτερολέπτων. Στην περίπτωση αυτή ο έλεγχος θα γίνει μέχρι να συμπληρώσουμε είτε τις επαναλήψεις είτε τον χρόνο. Όποιο από τα δύο πραγματοποιηθεί πρώτο θα σταματήσει το σκανάρισμα και θα μας οδηγήσει στην επεξεργασία των δειγμάτων που συλλέξαμε ως τότε.

Όλα τα δεδομένα της εισόδου ελέγχονται από ειδικά σχεδιασμένες συναρτήσεις έτσι ώστε το πρόγραμμα να έχει σωστές τιμές για να κάνει τους απαιτούμενους υπολογισμούς.

6.4 Κύριο μέρος του προγράμματος

Αφού ο χρήστης έχει περάσει στο πρόγραμμα όλες τις τιμές με τις οποίες θέλει να γίνει το σκανάρισμα τότε τοποθετούμε σε μία λίστα όλες τις κεντρικές συχνότητες των καναλιών στα οποία έχουμε σπάσει το ζητούμενο εύρος συχνοτήτων. Το πρόγραμμα χρησιμοποιεί μία επικάλυψη 5% μεταξύ των καναλιών. Όπως καταλαβαίνει κανείς με αυτή την επικάλυψη κάποια bins έχουν υπολογιστεί 2 φορές στην περίπτωση που έχουμε από 2 κανάλια και πάνω. Υπολογίζουμε αρχικά πόσα είναι αυτά τα bins και αργότερα θα χρησιμοποιήσουμε αυτήν την πληροφορία για να τα επεξεργαστούμε σωστά και να βγάλουμε σωστά αποτελέσματα.
Στη συνέχεια υλοποιούμε τις 4 διαφορετικές λειτουργίες σκαναρίσματος που μπορεί να επιλέξει ο χρήστης όταν χρησιμοποιεί το πρόγραμμα.
α) Σκανάρισμα με συγκεκριμένο αριθμό περασμάτων και χρονικό περιορισμό.
β) Σκανάρισμα με συγκεκριμένο αριθμό περασμάτων.
γ) Σκανάρισμα με χρονικό περιορισμό.
δ) Σκανάρισμα χωρίς περιορισμούς.

Για κάθε μία από αυτές τις λειτουργίες καλείται μία συνάρτηση που υλοποιεί κάθε φορά ότι απαιτείται για να μας δώσει το πρόγραμμα τα αποτελέσματα που θέλει ο χρήστης.
’Ολες όμως καλούν εσωτερικά την συνάρτηση scan_bandwidth(sdr, args) άρα ας ξεκινήσουμε από αυτή τη συνάρτηση.
Η συνάρτηση αυτή όταν καλείται κάνει ένα πέρασμα από το σκανάρισμα. Αν ο χρήστης θέλει περισσότερα περάσματα τότε καλείται περισσότερες από μία φορές. Δέχεται δύο ορίσματα sdr και args. Το πρώτο είναι για να έχουμε πρόσβαση στο SDR μας και να μπορούμε να το ρυθμίσουμε σωστά και το δεύτερο είναι για να έχουμε πρόσβαση στις εισόδους του χρήστη.
Αρχικά ρυθμίζουμε την SDR συσκευή μας. Πιο συγκεκριμένα στη μεταβλητή num_samps έχουμε δώσει την τιμή 35000. Έτσι όταν τρέχουμε την εντολή sdr.rx_buffer_size = num_samps λέμε στην SDR συσκευή μας ότι κάθε φορά που θα καλούμε την sdr.rx() θα συλλέγει για εμάς 35000 δείγματα.
Με την εντολή sdr.gain_control_mode_chan0 = “manual” καταργούμε την αυτόματη ρύθμιση του gain στην SDR συσκευή μας και με τις εντολές gain = 50 και sdr.rx_hardwaregain_chan0 = gain θέτουμε το gain = 50 dB.
Από εκεί και πέρα χρησιμοποιούμε την λίστα με τις κεντρικές συχνότητες όλων των καναλιών του ζητούμενου εύρους συχνοτήτων για να σκανάρουμε και να πάρουμε δείγματα σε κάθε κανάλι. Αυτό γίνεται πολύ απλά με την εξής διαδικασία.
Με την εντολή sdr.rx_lo = int(freq) λέμε στο SDR να πάει στην συχνότητα freq. Το SDR εκεί θα ακούσει ένα εύρος συχνοτήτων ίσο με sample rate. Πιο συγκεκριμένα θα ακούσει από συχνότητες από freq - (sample_rate / 2) εώς και freq + (sample_rate / 2) θα μας επιστρέψει 35000 δείγματα.[14]
Η συνάρτηση μέσω της οποίας γίνεται το σκανάρισμα κάθε καναλιού είναι η sdr.rx() και στην μεταβλητή rx_samples αποθηκεύονται τα 35000 δείγματα που μας επιστρέφει. Αυτά τα δείγματα που πήραμε από ένα κανάλι αποθηκεύονται σε μία λίστα που λέγεται iteration_samples.
Στη συνέχεια γίνεται η ίδια διαδικασία έως ότου μας τελειώσουν οι κεντρικές συχνότητες, δηλαδή όλα μας τα κανάλια. Όταν αυτό συμβεί η λίστα iteration_samples θα έχει όλα τα δείγματα για ένα πλήρες πέρασμα του σκαναρίσματος. Η scan_bandwidth() τότε επιστρέφει στην συνάρτηση που την κάλεσε τα δείγματα του περάσματος.

Τώρα που έχουμε δει την scan_bandwidth() μπορούμε να μιλήσουμε για τις βασικές συναρτήσεις των επιμέρους λειτουργιών του κώδικα.

- Για την α λειτουργία καλείται η συνάρτηση iterations_time_restricted_scan(sdr, args). Έχει δύο ορίσματα το sdr για να μπορούμε να ρυθμίζουμε την SDR συσκευή και το args για πρόσβαση στις εισόδους του χρήστη.
  Η συνάρτηση αυτή είναι σχεδιασμένη να τρέξει όσα περάσματα μπορεί χωρίς να ξεπεράσουμε είτε το χρονικό όριο που έθεσε ο χρήστης είτε τον αριθμό περασμάτων που επιθυμούσε. Σε κάθε πέρασμα μετράμε τον χρόνο που πήρε για να γίνει και τον συγκρίνουμε με τον υπόλοιπο χρόνο που έχει απομείνει για το σκανάρισμα. Αν προλαβαίνουμε τότε κάνουμε κανονικά το πέρασμα, αν όχι σταματάμε. Εσωτερικά καλούμε την scan_bandwidth() για να μας επιστρέψει τα δείγματα όλου του περάσματος. Αυτά τα δείγματα τα αποθηκεύουμε στη λίστα all_samples. Όταν τελειώσει το σκανάρισμα η all_samples θα έχει όλα τα δείγματα του σκαναρίσματος, δηλαδή όλα τα δεδομένα μας. Η all_samples είναι και αυτό που επιστρέφουμε στην main() για να γίνει η επεξεργασία των δεδομένων πλέον.

- Για την β λειτουργία καλείται η συνάρτηση iteration_restricted_scan(sdr, args). Έχει δύο ορίσματα το sdr για να μπορούμε να ρυθμίζουμε την SDR συσκευή και το args για πρόσβαση στις εισόδους του χρήστη.
  Η συνάρτηση αυτή είναι σχεδιασμένη να τρέξει όσα περάσματα επιθυμεί ο χρήστης. Πιο συγκεριμένα καλεί τη scan_bandwidth() όσες φορές ζήτησε ο χρήστης και αποθηκεύει στην all_samples τα δείγματα όλων των περασμάτων που ζητήθηκαν. Στο τέλος επιστρέφει στην main() όλα τα δείγματα του σκαναρίσματος.

- Για την γ λειτουργία καλείται η συνάρτηση time_restricted_scan(sdr, args). Έχει δύο ορίσματα το sdr για να μπορούμε να ρυθμίζουμε την SDR συσκευή και το args για πρόσβαση στις εισόδους του χρήστη.
  Η συνάρτηση αυτή είναι σχεδιασμένη να τρέξει όσα περάσματα προλαβαίνει στο χρονικό διάστημα που επιθυμεί ο χρήστης. Καλό θα είναι να διευκρινήσουμε ότι το χρονικό αυτό διάστημα αφορά χρόνο sensing. Πιο

συγκεριμένα καλεί τη scan_bandwidth() μία φορά μετρώντας πόσο χρόνο έκανε για ένα πέρασμα. Αφαιρεί αυτό τον χρόνο από το χρονικό όριο του χρήστη για να βρει πόσος χρόνος απομένει κι αν αυτός επαρκεί για καινούριο πέρασμα τότε το κάνει χρονομετρώντας το ξανά. Αυτό συνεχίζεται όσο έχουμε χρόνο και για νέο πέρασμα, αν ωστόσο κάποια στιγμή ξεμείνουμε από χρόνο τότε επιστρέφουμε την λίστα all_samples στην main() στην οποία καθ όλη τη διάρκεια του σκαναρίσματος αποθηκεύουμε τα δεδομένα των περασμάτων.

- Για την δ λειτουργία καλείται η συνάρτηση no_restricted_scan(sdr, args). Έχει δύο ορίσματα το sdr για να μπορούμε να ρυθμίζουμε την SDR συσκευή και το args για πρόσβαση στις εισόδους του χρήστη.
  Η συνάρτηση αυτή αποτελεί την βασική λειτουργία σκαναρίσματος χωρίς κανένα περιορισμό, ούτε χρονικό ούτε με κάποιο συγκεκριμένο αριθμό περασμάτων. Για την ακρίβεια είναι σχεδιασμένη να τρέξει τέσσερα περάσματα στο ζητούμενο εύρος συχνοτήτων. Καλεί τη scan_bandwidth() τέσσερις φορές και αποθηκεύει στην all_samples τα δείγματα των τεσσάρων περασμάτων που έκανε. Στο τέλος επιστρέφει στην main() όλα τα δείγματα του σκαναρίσματος.

Όποια από τις παραπάνω συναρτήσεις και να κλήθηκε, όποιο μονοπάτι και να διαλέξαμε είναι βέβαιο πλέον ότι στην main έχουμε στην λίστα all_samples όλα μας τα δεδομένα του σκαναρίσματος. Σειρά τώρα έχει η επεξεργασία και η αποθήκευση των δεδομένων μας στη βάση δεδομένων μας.

Επόμενη βασική συνάρτηση που θα χρησιμοποιήσουμε για την επεξεργασία των δειγμάτων μας είναι η psd(rx_samples, args). Όπως βλέπουμε έχει δύο ορίσματα, το πρώτο είναι τα δείγματα που θα της δώσουμε να επεξεργαστεί και το δεύτερο είναι για την πρόσβαση στα δεδομένα τις εισόδου από το χρήστη.
Αρχικά, κρατάμε FFTsize από τα συνολικά 35000 δείγματα που έχουμε. Στη συνέχεια εφαρμόζουμε μία window function, πιο συγκεκριμένα την Hamming, και κάνουμε μετασχηματισμό Fourier στα δείγματα μας. Υπολογίζουμε το πλάτος τους, υψώνουμε στο τετράγωνο και κανονικοποιούμε διαιρώντας με το γινόμενο sample rate * FFTsize . Μετατρέπουμε σε dBs και με τη συνάρτηση FFTshift κάνουμε μία

αναδιάταξη στα bins ώστε να έχουμε αριστερά της κεντρικής συχνότητας τις μικρότερες συχνότητες και δεξιά της τις μεγαλύτερες της. Αυτό το κάνουμε γιατί μετά το μετασχηματισμό Fourier οι συχνότητες βγαίνουν λίγο “ανακατεμένες”. Τέλος η συνάρτηση επιστρέφει στην main στην λίστα samples_dbs τα επεξεργασμένα πλέον δείγματα σε μορφή λίστας με FFTsize σε αριθμό στοιχεία.

Τα δεδομένα μας θα αποθηκευτούν σε βάση δεδομένων μέσω της συνάρτησης insert_in_database(samples_dbs, iteration_no).
Πλέον έχουμε ένα έως και αρκετά περάσματα αποθηκευμένα στη βάση δεδομένων μας. Επειδή όμως για τα τελικά αποτελέσματα χρειαζόμαστε ένα μόνο πέρασμα θα βγάλουμε ένα μέσο όρο για το κάθε κανάλι. Αυτό υλοποιούμε στη συνέχεια όπου με τρόπο ζητάμε από τη βάση δεδομένων τα δεδομένα που υπάρχουν αποθηκευμένα σε αυτή και αφορούν το ίδιο κανάλι. Τα δεδομένα τα παίρνουμε από τη βάση δεδομένων μέσω της συνάρτησης get_channel_data_from_database(channel_id). Ο μέσος όρος των δειγμάτων που έχουμε για το κάθε κανάλι αποθηκεύεται στη λίστα db_list[]. Έτσι η db_list έχει στη θέση 0 τα δεδομένα του 1ου καναλιού, στη θέση 2 τα δεδομένα του 2ου καναλιού και πάει λέγοντας..
Το μόνο που μένει πλέον είναι να συνενώσουμε πλέον τα δείγματα των καναλιών μεταξύ τους ώστε να έχουμε όλο το ζητούμενο bandwidth.
Λόγω της επικάλυψης είπαμε και προηγουμένως ότι μεταξύ γειτονικών καναλιών υπάρχουν στα σκαναρίσματα μας bins τα οποία αναφέρονται στις ίδιες περιοχές συχνοτήτων και αυτό αποτελεί πρόβλημα για εμάς. Πράγμα που σημαίνει ότι μεταξύ γειτονικών καναλιών έχουμε υπολογίσει κάποια bins δύο φορές. Έτσι αν απλώς συνενώναμε τα δείγματα των καναλιών θα είχαμε οδηγηθεί με βεβαιότητα σε λάθος υπολογισμούς. Ο τρόπος με τον οποίο αντιμετωπίσαμε το ζήτημα είναι να υπολογίζουμε αρχικά πόσα bins είναι αυτά στη μεταβλητή overlap_bins. Στη συνέχεια υπολογίσαμε τον μέσο όρο των τιμών των δειγμάτων που είχαμε γι αυτά τα bins και τα κρατάμε μία φορά αντί για δύο. Η διαδικασία της συνένωσης έχει ως εξής. Κρατάμε από το πρώτο κανάλι εκείνα τα δείγματα που δεν γίνονται overlap. Αφου κρατήσουμε αυτά τα δείγματα τότε θα υπάρχουν κάποια στο τέλος του πρώτου καναλιού και στις αρχές του δεύτερου καναλιού τα οποία αναφέρονται στις ίδιες συχνότητες. Εκεί υπολογίζουμε τον μέσο όρο τους και τα κολλάμε στα δείγματα του πρώτου καναλιού που έχουμε κρατήσει. Το δεύτερο κανάλι έχει

overlapped δείγματα και στην αρχή του και στο τέλος του. Τα δείγματα που είναι στην αρχή του ήδη τα έχουμε συνενώσει καθώς ο μέσος όρος τους έχει μπει στο τέλος του πρώτου καναλιού. Επομένως δε τα χρειαζόμαστε, συνενώνουμε αυτά που είναι στη μέση από το δεύτερο κανάλι τα οποία αναφέρονται στις επόμενες συχνότητες. Το τέλος του δευτέρου καναλιού θα προστεθεί με την αρχή του τρίτου καναλιού για να βγάλουμε ένα μέσο όρο τους τον οποίο θα συνενώσουμε στα δείγματα μας. Η ίδια διαδικασία συνεχίζεται μέχρι τέλος ώστε να μην έχουμε overlapped bins και να έχουμε δείγματα για όλες τις συχνότητες από μία φορά.
Έτσι συνενώνοντας τα δείγματα μας μπορούμε πλέον να έχουμε την γραφική παράσταση μας η οποία θα μας δείξει τα αποτελέσματα του σκαναρίσματος μας.
Φυσικά υπάρχει και η περίπτωση να έχουμε μόνο ένα κανάλι στο σκανάρισμα μας. Σε αυτή την περίπτωση προχωράμε κατευθείαν στη γραφική παράσταση αφού δεν υφίσταται το συγκεκριμένο πρόβλημα.
Η γραφική παράσταση είναι μία γραφική παράσταση ισχύος ως προς τη συχνότητα. Μπορούμε να δούμε λοιπόν την ενέργεια που υπάρχει στην περιοχή συχνοτήτων που ζήτησε ο χρήστης και να συμπεράνουμε αν υπάρχει ή όχι κάποια μετάδοση στις συγκεκριμένες συχνότητες.

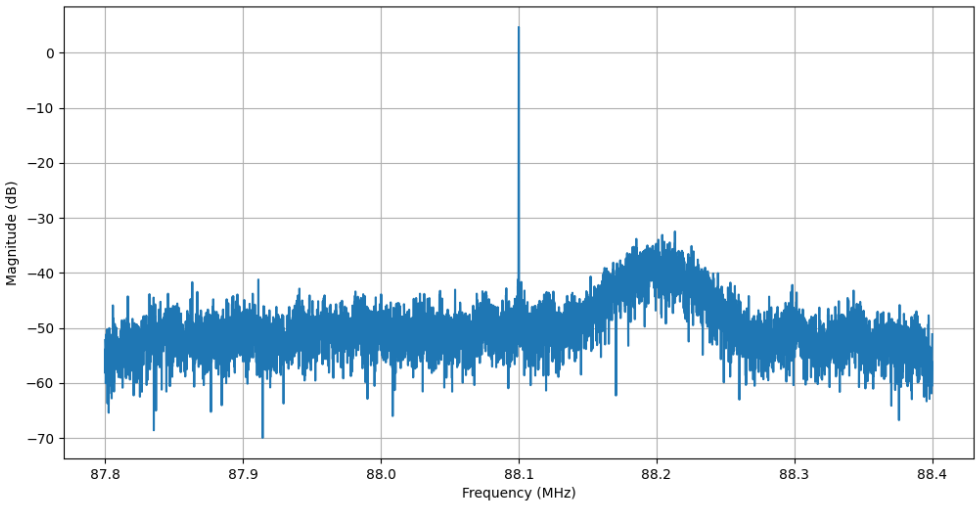
6.5 ΠΑΡΟΥΣΙΑΣΗ ΦΑΣΜΑΤΙΚΟΥ ΑΝΑΛΥΤΗ ΜΕ ΧΡΗΣΗ PLUTO SDR

Ήρθε η ώρα να δούμε και το πρόγραμμα σε δράση. Ο κώδικας βρίσκεται στο παράρτημα για να μπορείτε να δείτε αναλυτικά όλες τις εντολές του. Τώρα εδώ έχουμε τρέξει το πρόγραμμα σε διάφορες λειτουργίες του και σε διαφορετικές συχνότητες και εύρη συχνοτήτων ώστε να σας το παρουσιάσουμε. Πιο συγκεκριμένα έχουμε κάνει πειράματα στα FM, Wi-Fi, LTE. Πάμε λοιπόν να δούμε ένα ένα τα πειράματα σε κάθε επιμέρους εύρος συχνοτήτων.

FM

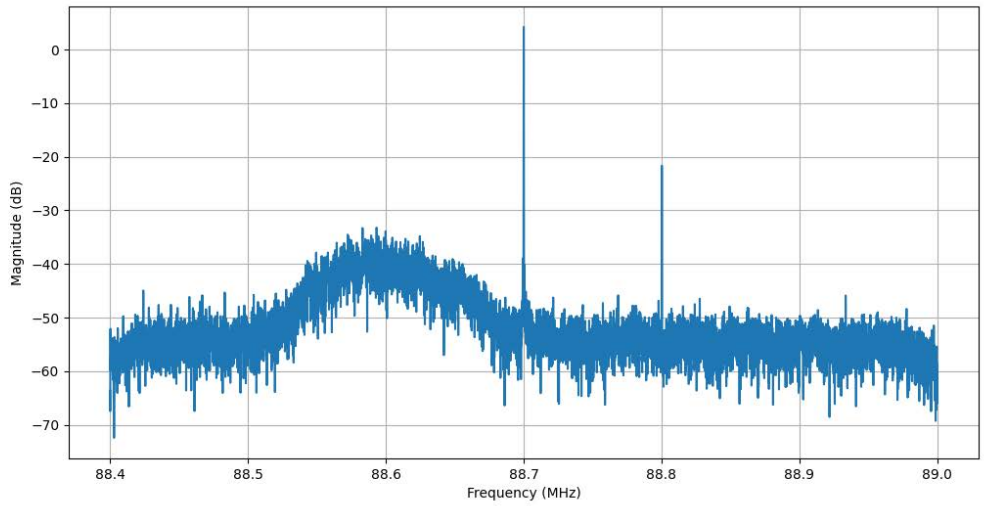
Η μπάντα συχνοτήτων των FM εκτείνεται από τα 87.0 MHz εώς τα 108.0 MHz. Συντονίσαμε λοιπόν το Pluto SDR σε αυτή την περιοχή συχνοτήτων και ιδού τα αποτελέσματα.

Δοκιμάσαμε να κάνουμε ανίχνευση μεταδόσεων σε όλη την μπάντα των FM. Έχουμε κάνει στοχευμένες δοκιμές σε ένα μεμονωμένο ραδιοφωνικό σταθμό, σε δύο ραδιοφωνικούς σταθμούς μαζί, σε μια γειτονιά τεσσάρων ραδιοφωνικών σταθμών αλλά και σε όλη την περιοχή συχνοτήτων από τα 87 εώς τα 108 MHz.
Ας ξεκινήσουμε με δύο παραδείγματα μεμονωμένων ραδιοφωνικών σταθμών της περιοχής του Βόλου.
Ξεκινάμε με ένα σκανάρισμα στο εύρος συχνοτήτων 87.8 – 88.4 MHz, κανάλι στα 0.6 MHz, 4 περάσματα με FFTsize 8192 και τα αποτελέσματα φαίνονται στην παρακάτω εικόνα.



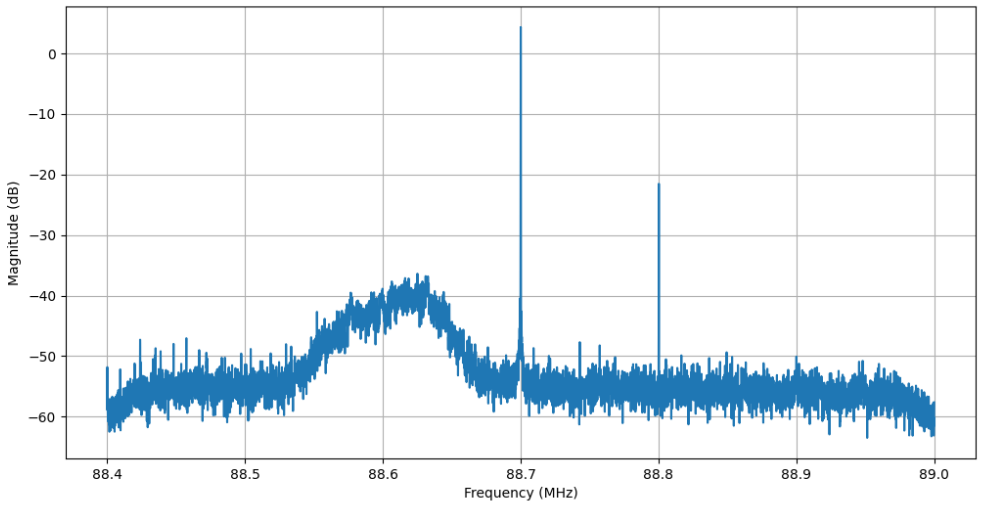
Εικόνα 37: -f 87.8 -b 0.6 -s 8192

Στη συνέχεια έχουμε τον αμέσως επόμενο ραδιοφωνικό σταθμό της περιοχής στα 88.6 MHz. Εδώ κάναμε τον έλεγχό μας στο εύρος συχνοτήτων 88.4 – 89.0 MHz, κανάλι και πάλι στα 0.6 MHz, 4 περάσματα με FFTsize 8192.



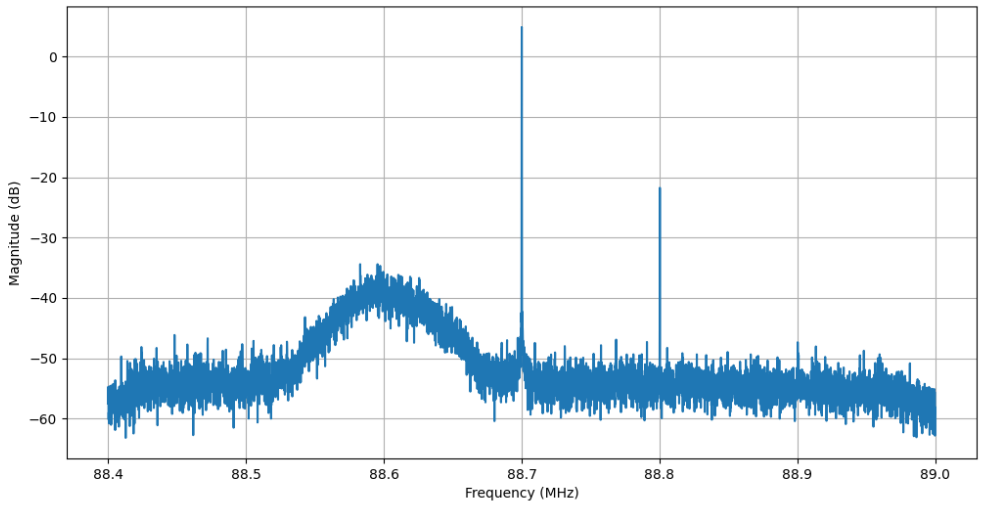
Εικόνα 38 : -f 88.4 -b 0.6 -s 8192

Εδώ επαναλάβαμε το σκανάρισμα με χρονικό περιορισμό 1 δευτερολέπτου κρατώντας όλες τις υπόλοιπες παραμέτρους σταθερές. Έγιναν συνολικά 13 επαναλήψεις με συνολικό χρόνο sensing 0.9356 δευτερόλεπτα.



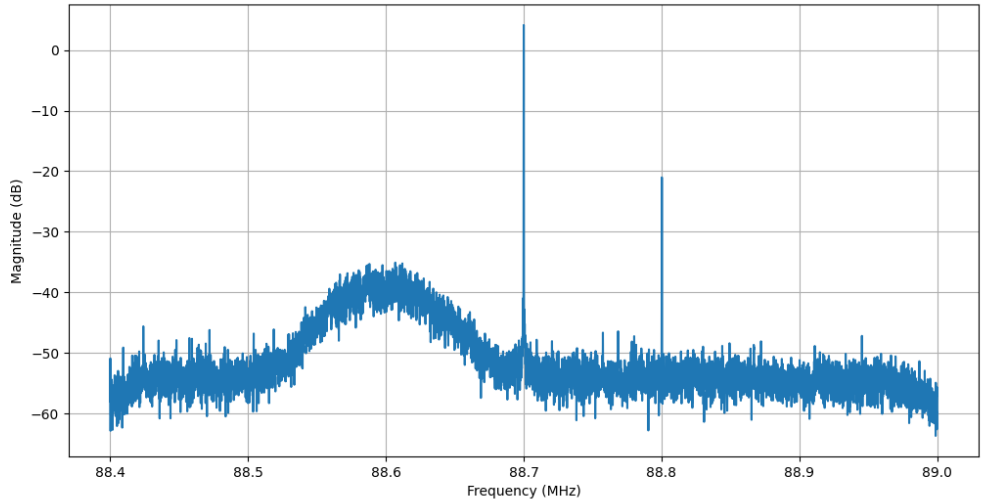
Εικόνα 39: -f 88.4 -b 0.6 -t 1 -s 8192

Στη συνέχεια πραγματοποιήσαμε το ίδιο σκανάρισμα χωρίς χρονικό περιορισμό αλλά με περιορισμό επαναλήψεων. Συγκεκριμένα ζητήσαμε 10 επαναλήψεις με χρόνο sensing 0.7128 δευτερόλεπτα.



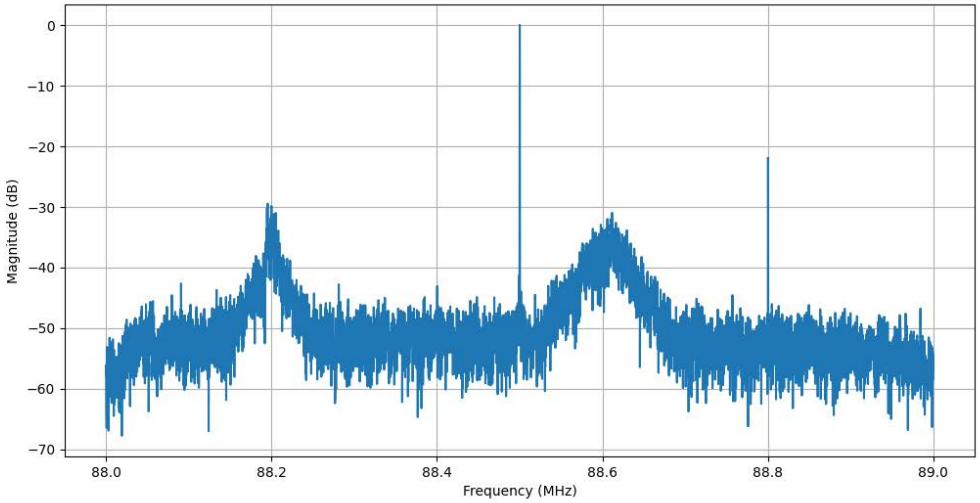
Εικόνα 40: -f 88.4 -b 0.6 -i 10 -s 8192

Τέλος κάναμε ένα ίδιο σκανάρισμα με χρονικό περιορισμό 1 δευτερολέπτου και περιορισμό επαναλήψεων ίσο με 10. Τελικά το σκανάρισμα μας σταμάτησε στα 10 περάσματα και σε χρόνο sensing 0.7203 δευτερολέπτων.



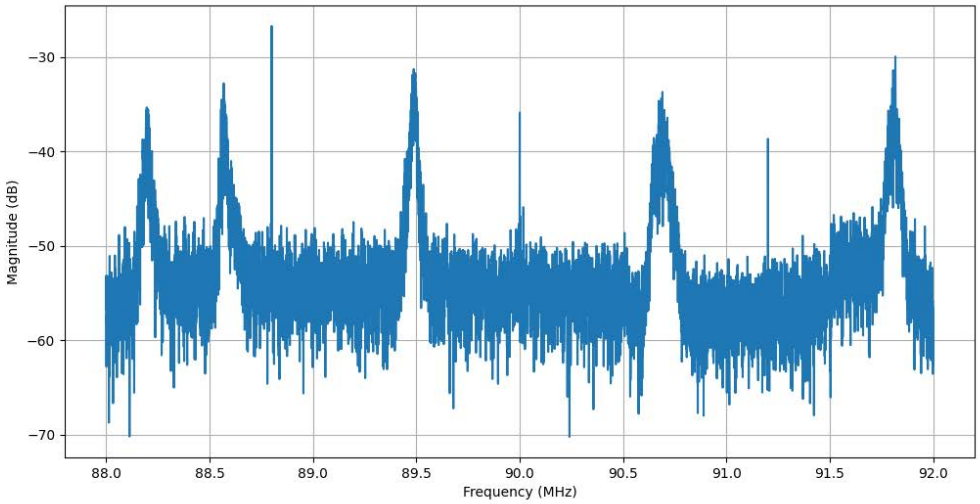
Εικόνα 41: -f 88.4 -b 0.6 -t 1 -i 10 -s 8192

Τώρα έχουμε ένα σκανάρισμα με τους 2 παραπάνω ραδιοφωνικούς σταθμούς μαζί αυτή τη φορά. Εύρος σκαναρίσματος 88 – 89 MHz, κανάλι 1 MHz αυτή τη φορά, 4 περάσματα και FFTsize 8192.



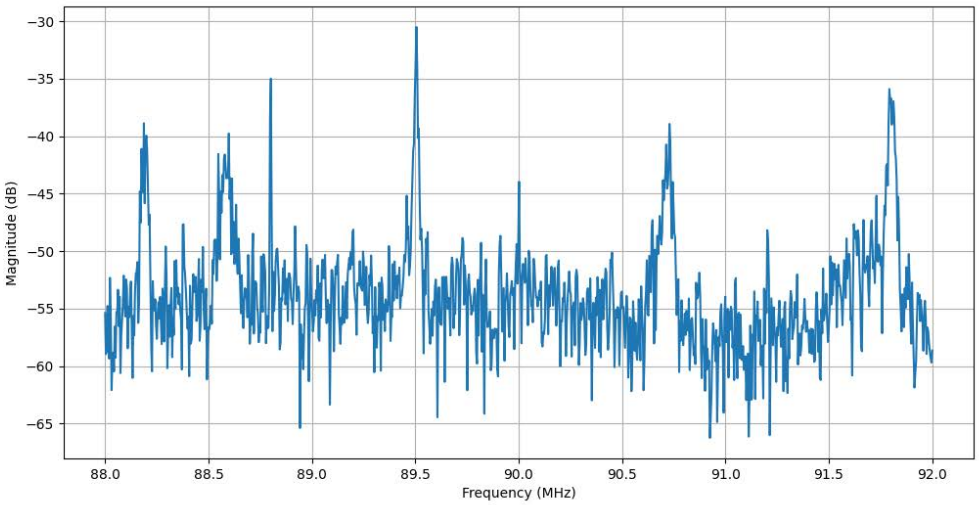
Εικόνα 42: -f 88 -b 1 -s 8192

Ας περάσουμε τώρα να δούμε μία πεντάδα ραδιοφωνικών σταθμών στο εύρος συχνοτήτων 88 – 92 MHz, κανάλι 4 MHz, 4 περάσματα και FFTsize 8192.



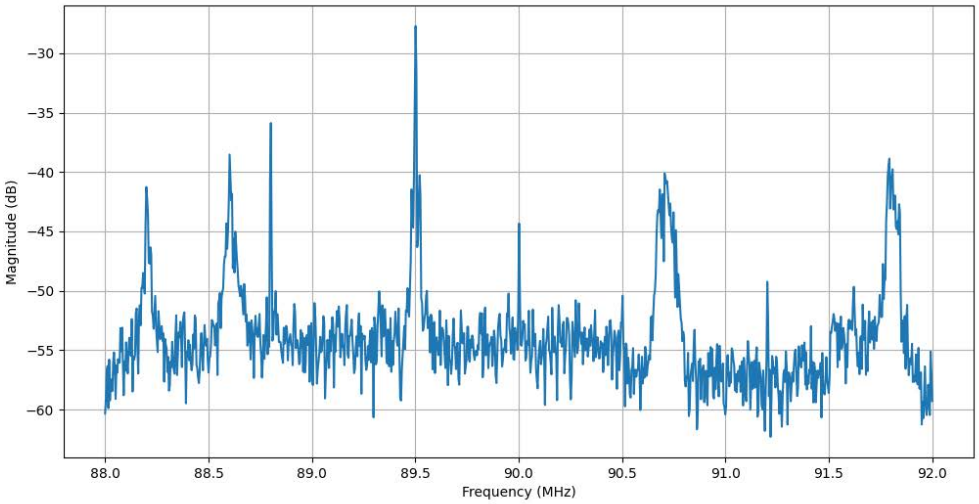
Εικόνα 43: -f 88 -b 4 -s 8192

Εκτελέσαμε ακριβώς τον ίδιο έλεγχο αλλά με FFTsize 1024 αντί για 8192. Τα αποτελέσματα στην παρακάτω εικόνα.



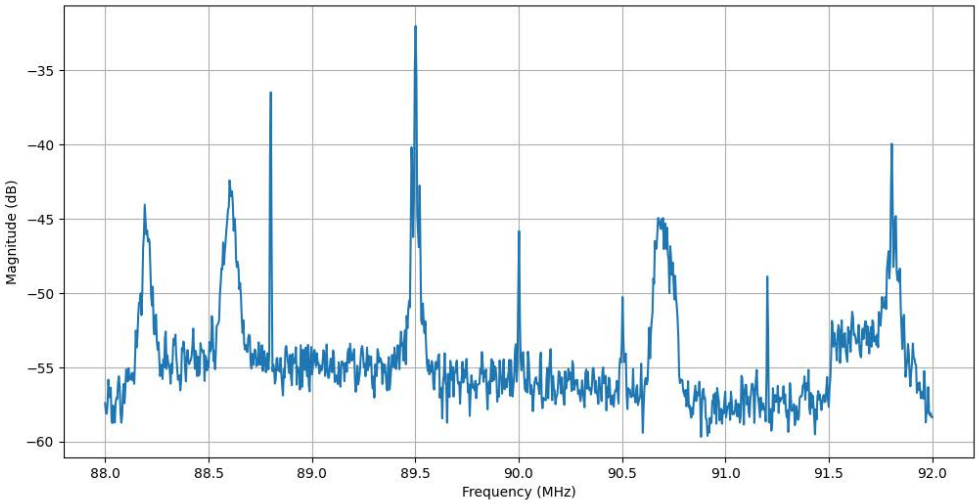
Εικόνα 44: -f 88 -b 4 -s 1024

Στη συνέχεια δοκιμάσαμε να βάλουμε περιορισμό επαναλήψεων, 10 συγκεκριμένα.



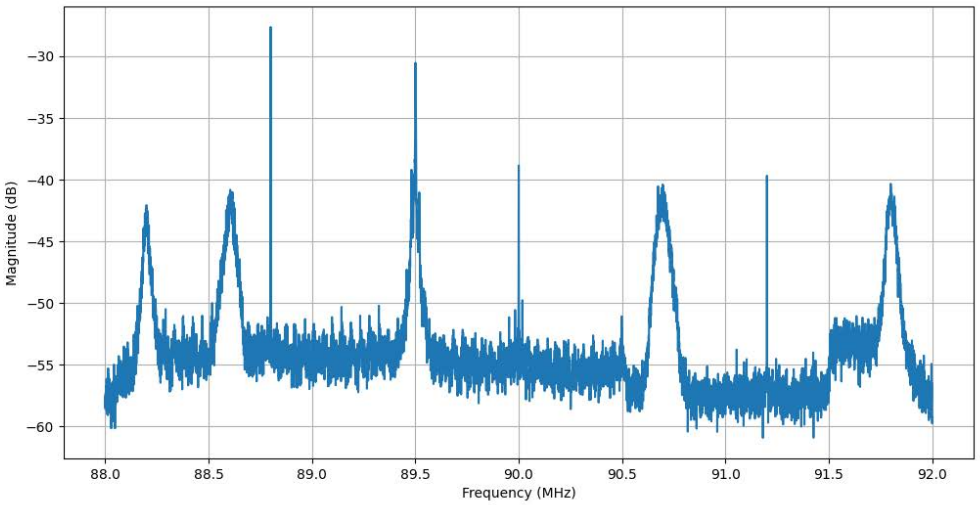
Εικόνα 45: -f 88 -b 4 -i 10 -s 1024

Το ίδιο σκανάρισμα με χρονικό περιορισμό 1 δευτερολέπτου χωρίς περιορισμό επαναληψεων.



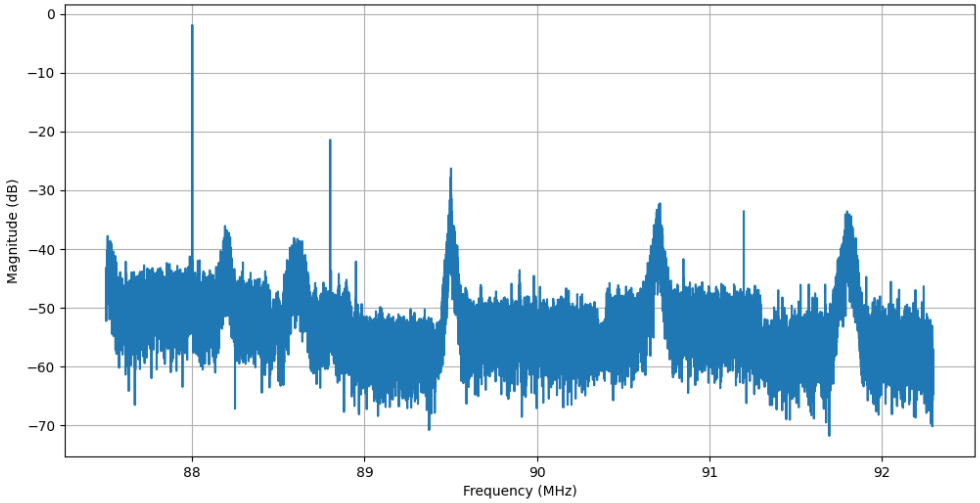
Εικόνα 46: -f 88 -b 4 -t 1 -s 1024

Επαναφέραμε το FFTsize στο 8192 και τρέχουμε το παραπάνω σκανάρισμα ξανά.



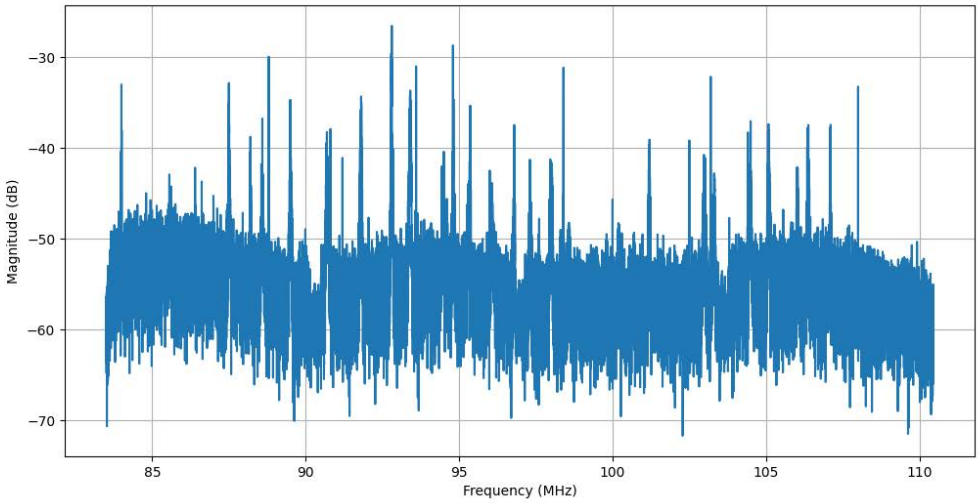
Εικόνα 47: -f 88 -b 4 -t 1 -s 8192

Τέλος, τρέχουμε το σκανάρισμα 88-92 MHz, με κανάλι 1 MHz, χρονικό περιορισμό 1 δευτερόλεπτο, FFTsize 8192.



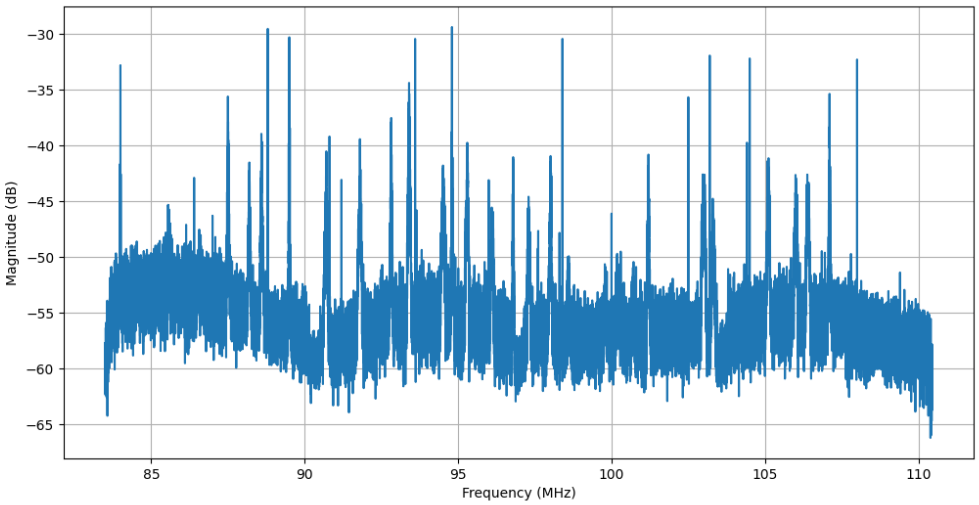
Εικόνα 48: -f 88 -b 4 -c 1 -t 1 -s 8192

Τώρα ακολουθούν κάποια τρεξίματα σε όλο το φάσμα των FM, θα ξεκινήσουμε με ένα σκανάρισμα από 87 – 108 MHz, με κανάλια 10 MHz, 4 περάσματα με FFTsize 8192.



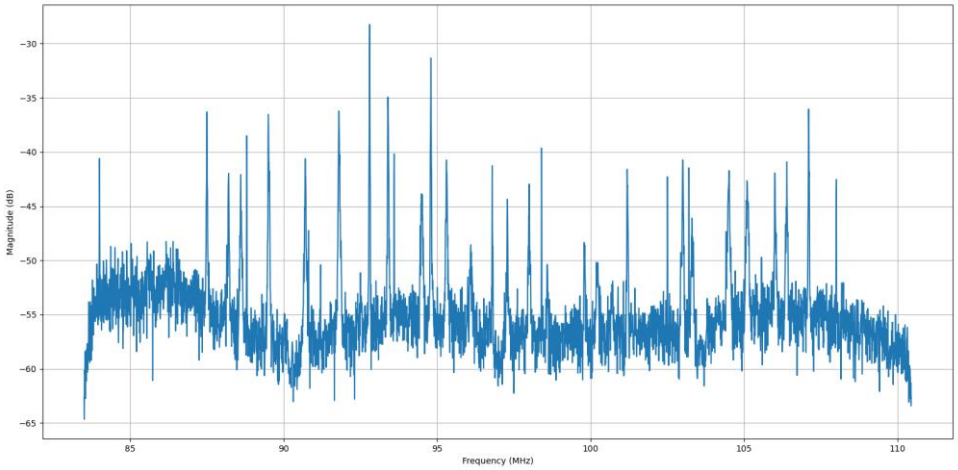
Εικόνα 49: -f 87 -b 21 -s 8192

Ακολουθεί το ίδιο σκανάρισμα με χρονικό περιορισμό 1 δευτερόλεπτο. Έγιναν συνολικά 13 περάσματα με χρόνο sensing 0.9499 δευτερόλεπτα.



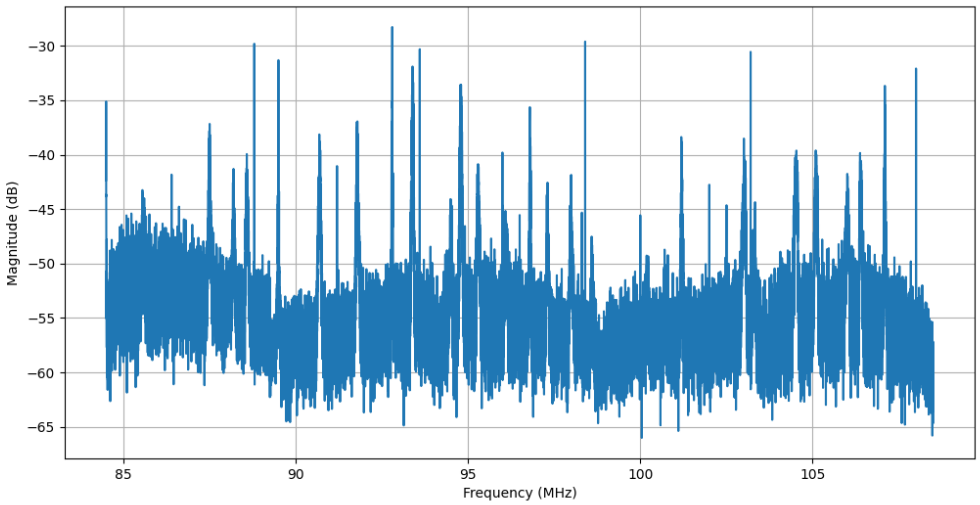
Εικόνα 50: -f 87 -b 21 -t 1 -s 8192

Έχουμε επίσης το ίδιο με το προηγούμενο σκανάρισμα αλλά με FFTsize 1024.



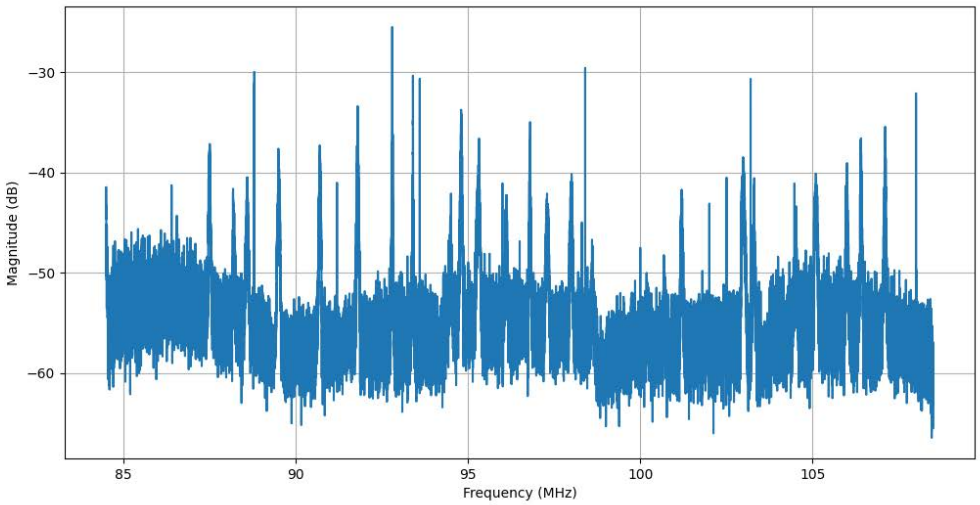
Εικόνα 51: -f 87 -b 21 -t 1 -s 1024

Το προτελευταίο σκανάρισμα στη μπάντα των FM αφορά το εύρος συχνοτήτων από 87 – 108 MHz, με κανάλι 5 MHz, χρονικό περιορισμό 1 δευτερολέπτου και FFTsize 8192. Έγιναν συνολικά 9 περάσματα σε χρόνο sensing 0.9211 δευτερολέπτων.



Εικόνα 52: -f 87 -b 21 -c 5 -t 1 -s 8192

Τελευταίο είναι το ίδιο με το προηγούμενο αλλά αντί για χρονικό περιορισμό ζητήσαμε 10 περάσματα. Συνολικός χρόνος sensing 1.0095 δευτερόλεπτα.



Εικόνα 53: -f 87 -b 21 -c 5 -i 10 -s 8192

Αυτά ήταν όλα τα σκαναρίσματα που αφορούσαν το FM, ακολουθεί πίνακας που παρουσιάζει τα χρονικά δεδομένα του κάθε σκαναρίσματος. Έχουμε χρόνο για sensing, χρόνο για την επεξεργασία των δειγμάτων μας και χρόνο για τη μεταφορά και αποθήκευση των δεδομένων μας στη βάση δεδομένων. Παρουσιάζουμε σε αυτόν τον πίνακα τις τρεις αυτές χρονικές παραμέτρους για ένα κανάλι (channel),

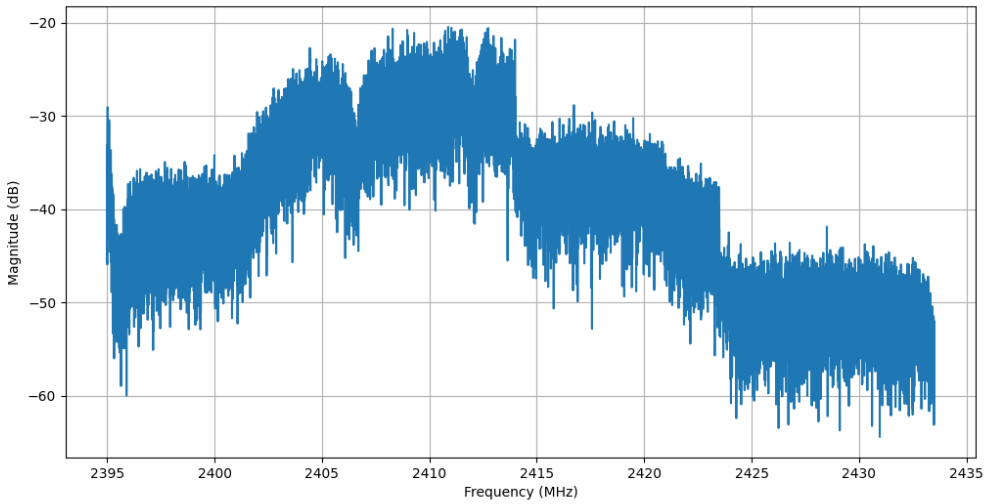
για ένα πέρασμα (iteration) και για όλο το σκανάρισμα (scan). Φυσικά για όλα τα παραπάνω τρεξίματα.

Παράμετροι Scan	Sense channel	Sense iteration	Sense scan	Proc. channel	Proc. iteration	Proc. scan	DB channel	DB iteration	DB scan	Total time
-f 87.8 -b 0.6 -s 8192	0.0484	0.0484	0.1917	0.0003	0.0003	0.0013	0.0110	0.0110	0.0597	0.2527 (4 iter)
-f 88.4 -b 0.6 -s 8192	0.0715	0.0715	0.2845	0.0004	0.0004	0.0014	0.0109	0.0109	0.0559	0.3418 (4 iter)
-f 88.4 -b 0.6 -t 1 -s 8192	0.0714	0.0714	0.9356	0.0003	0.0003	0.0044	0.0113	0.0113	0.1593	1.0993 (13iter)
-f 88.4 -b 0.6 -i 10 -s 8192	0.0714	0.0714	0.7128	0.0003	0.0003	0.0031	0.0112	0.0112	0.1260	0.8419 (10iter)
-f 88.4 -b 0.6 -t 1 -i 10 -s 8192	0.0719	0.0719	0.7203	0.0003	0.0003	0.0037	0.0110	0.0110	0.1221	0.8461 (10iter)
-f 88 -b 1 -s 8192	0.0484	0.0484	0.1917	0.0003	0.0003	0.0013	0.0110	0.0110	0.0597	0.2527 (4 iter)
-f 88 -b 4 -s 8192	0.0221	0.0221	0.0865	0.0003	0.0003	0.0013	0.0111	0.0111	0.0570	0.1448 (4 iter)
-f 88 -b 4 -s 1024	0.0221	0.0221	0.0871	0.0002	0.0002	0.0006	0.0103	0.0103	0.0500	0.1376 (4 iter)
-f 88 -b 4 -i 10 -s 1024	0.0219	0.0219	0.2187	0.0001	0.0001	0.0014	0.0095	0.0095	0.1077	0.3279 (10iter)
-f 88 -b 4 -t 1 -s 1024	0.0219	0.0219	0.9910	0.0001	0.0001	0.0068	0.0101	0.0101	0.4780	1.4758 (45iter)
-f 88 -b 4 -t 1 -s 8192	0.0219	0.0219	0.9923	0.0003	0.0003	0.0144	0.0111	0.0111	0.5245	1.5311 (45iter)
-f 88 -b 4 -c 1 -t 1 -s 8192	0.0479	0.2402	0.9680	0.0003	0.0017	0.0066	0.0111	0.0599	0.2395	1.2141 (4 iter)
-f 87 -b 21 -s 8192 (c=7)	0.0237	0.0846	0.3081	0.0003	0.0013	0.0054	0.0114	0.0464	0.2280	0.5414 (4 iter)
-f 87 -b 21 –t 1 -s 8192 (c=7)	0.0177	0.0716	0.9499	0.0003	0.0012	0.0175	0.0120	0.0457	0.6251	1.5925 (13iter)
-f 87 -b 21 -t 1 -s 1024 (c=7)	0.0180	0.0725	0.9672	0.0002	0.0006	0.0080	0.0103	0.0403	0.5246	1.4998 (13iter)
-f 87 -b 21 -c 5 -t 1 -s 8192 (c=7)	0.0200	0.0995	0.9211	0.0003	0.0017	0.0160	0.0114	0.0638	0.5703	1.5073 (9iter)
-f 87 -b 21 -c 5 -i 10 -s 8192 (c=7)	0.0200	0.1005	1.0095	0.0003	0.0018	0.0170	0.0107	0.0568	0.6016	1.6281 (10iter)

2.4 – 5 GHz

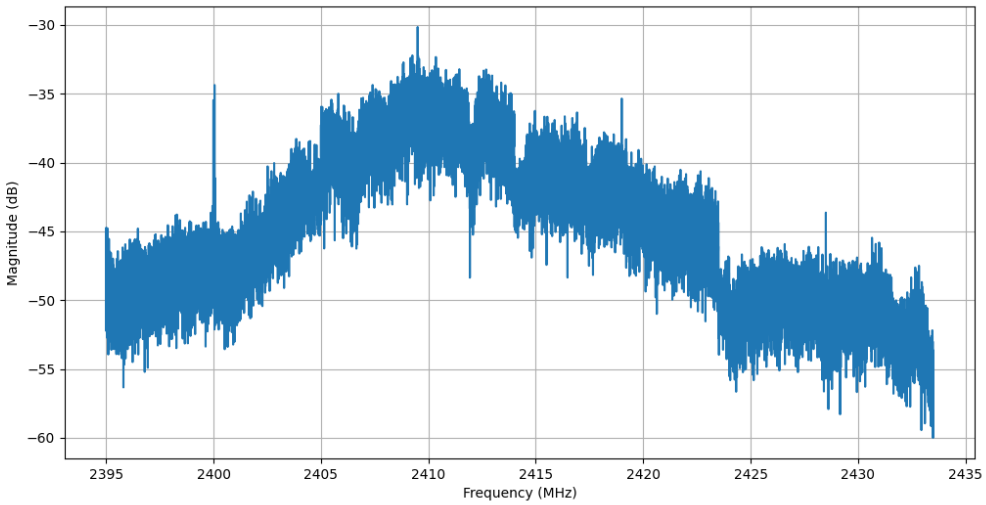
Τώρα θα περάσουμε στα σκαναρίσματα που τρέξαμε στα 2.4 GHz και στα 5 GHz.

Αρχικά κάναμε ένα σκανάρισμα μεταξύ 2400 – 2430 MHz με κανάλι 10 MHz και FFTsize 8192. Τα 4 iterations μας δίνουν το παρακάτω αποτέλεσμα. Εδώ φαίνεται η μετάδοση που υπάρχει στο κανάλι 1 (συχνότητες 2401 – 2423 MHz με κεντρική συχνότητα τα 2412 MHz).



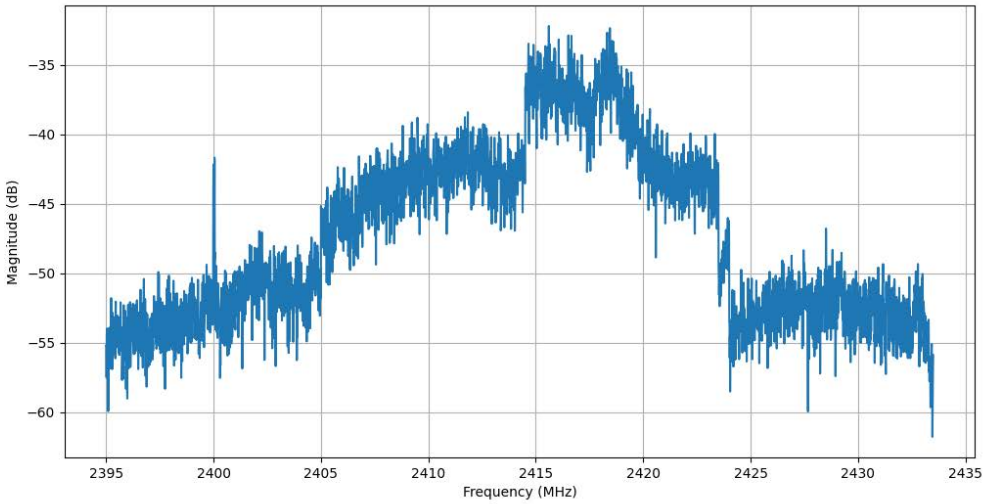
Εικόνα 54: -f 2400 -b 30 -s 8192

Το επόμενο είναι ακριβώς το ίδιο σκανάρισμα όμως αντί για 4 iterations που είναι το default τώρα κάναμε 13 iterations.Το αποτέλεσμα έχει βελτιωθεί λόγω των περισσότερων επαναλήψεων.



Εικόνα 55: -f 2400 -b 30 -i 13 -s 8192

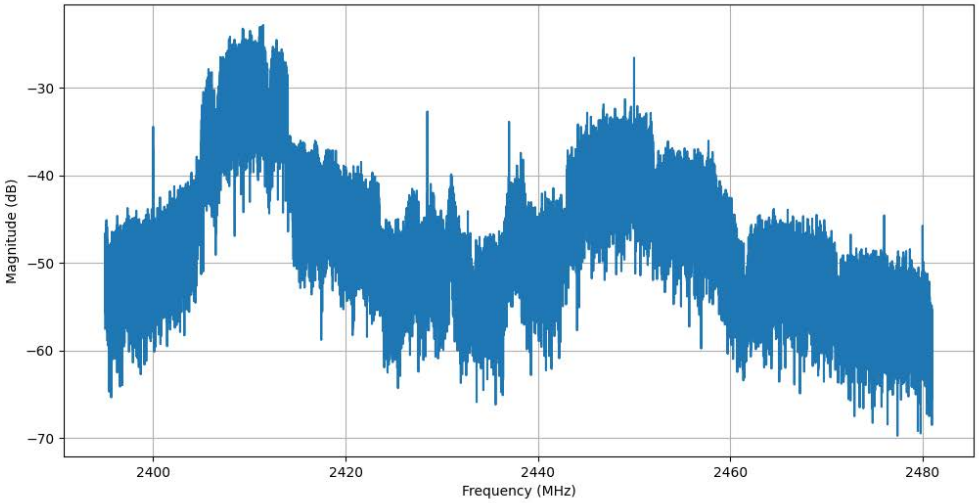
Αυτό που αλλάζει στο επόμενο σκανάρισμα είναι το FFTsize που από 8192 γίνεται 1024. Έχουμε πάλι 13 iterations στο ίδιο εύρος συχνοτήτων.



Εικόνα 56: -f 2400 -b 30 -i 13 -s 1024

Το τελευταίο σκανάρισμα που κάναμε στα 2.4 GHz είναι από τα 2400 MHz εώς τα 2480 MHz, κανάλι 10 MHz, 4 περάσματα και FFTsize 8192. Εδώ εκτός από τη

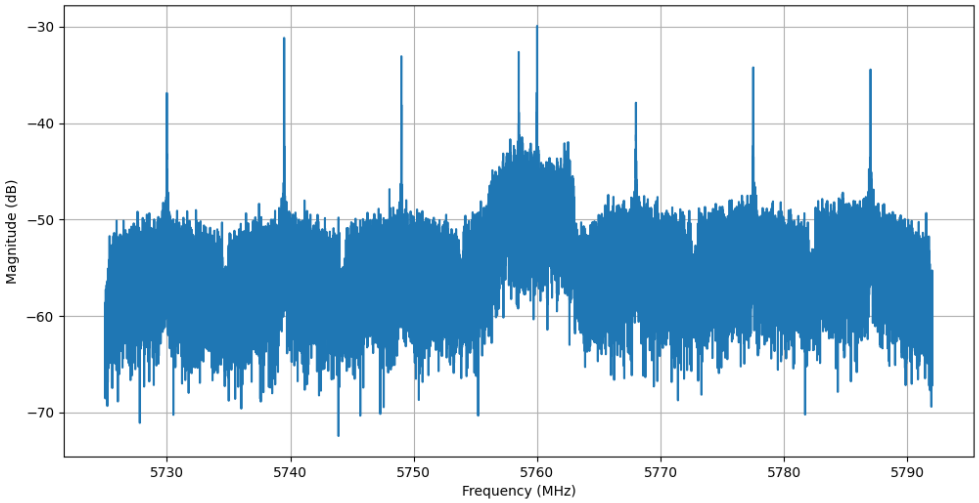
μετάδοση στο κανάλι 1 βλέπουμε και μία δεύτερη μεταξύ των συχνοτήτων 2440 - 2460 MHz.



Εικόνα 57: -f 2400 -b 80 -s 8192

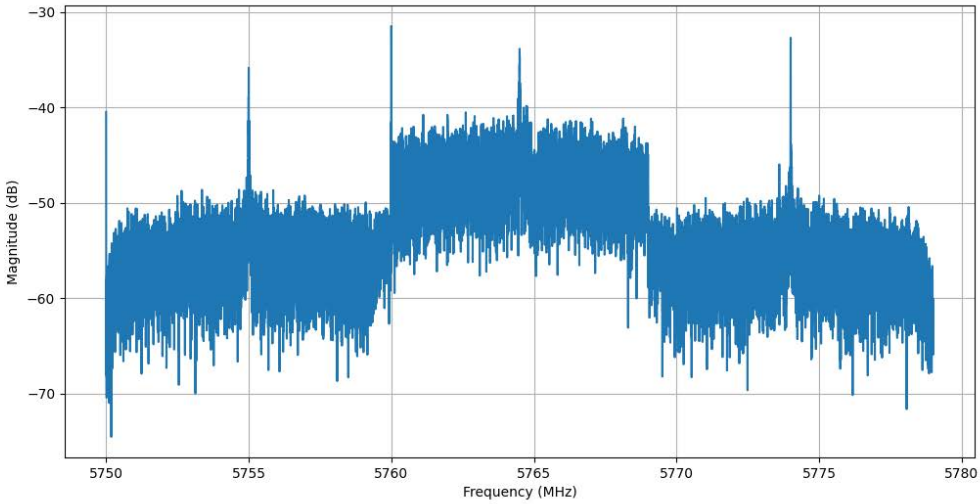
Πλέον μεταφερόμαστε στα 5 GHz για να δούμε κι εδώ άλλα δύο σκαναρίσματα που δοκιμάσαμε.

Το πρώτο μας δείχνει τις συχνότητες από τα 5730 MHz εώς τα 5790 MHz με κανάλι 10 MHz, 4 περάσματα και FFTsize 8192. Εδώ φαίνεται μετάδοση στο κανάλι 52 (κεντρική συχνότητα 5260 MHz, 5250 – 5270 MHz), επίσης πολύ εμφανές είναι και το DC spike στην κεντρική συχνότητα κάθε καναλιού από τα 7 που έχουμε συνολικά στο σκανάρισμα μας.



Εικόνα 58: -f 5730 -b 60 -s 8192

Το επόμενο και τελευταίο σκανάρισμα στα 5 GHz αφορά τις συχνότητες 5755 – 5775 MHz, με κανάλι 10 MHz, 4 περάσματα και FFTsize 8192. Εδώ βλέπουμε καλύτερα την μετάδοση στο κανάλι 52 που είδαμε προηγουμένως.



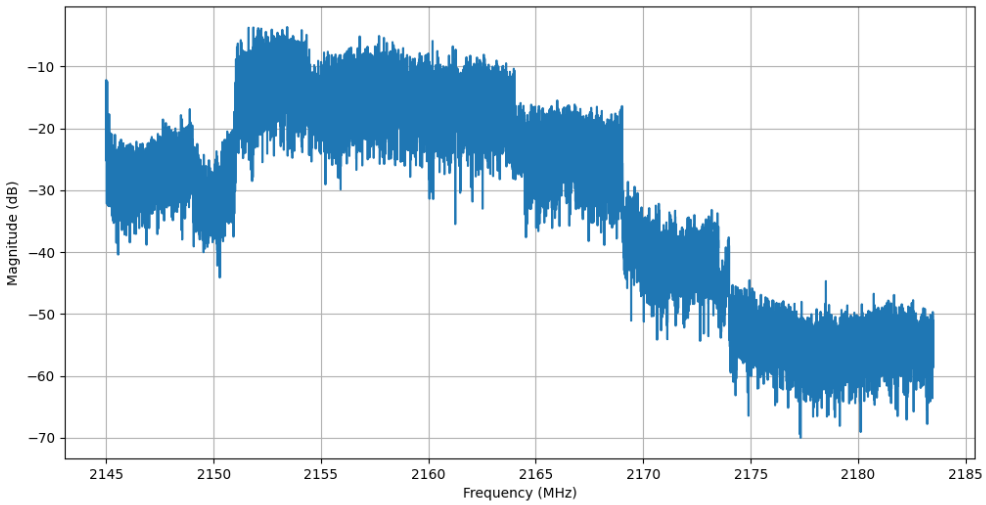
Εικόνα 59: -f 5755 -b 20 -s 8192

Στον παρακάτω πίνακα παρουσιάζουμε αναλυτικά τις χρονικές παραμέτρους του κάθε σκαναρίσματος που παρουσιάσαμε παραπάνω και αφορά τα 2.4 και 5 GHz.

Παράμετροι Scan	Sense channel	Sense iteration	Sense scan	Proc. channel	Proc. iteration	Proc. scan	DB channel	DB iteration	DB scan	Total time
-f 2400 -b 30 -s 8192	0.0165	0.0657	0.2756	0.0003	0.0012	0.0156	0.0111	0.0429	0.1734	0.4647 (4 iter)
-f 2400 -b 30 -i 13 -s 8192	0.0162	0.0660	0.8554	0.0012	0.0046	0.0425	0.0494	0.1966	1.8221	2.7201 (13 iter)
-f 2400 -b 30 -i 13 -s 1024	0.0165	0.0659	0.8860	0.0004	0.0017	0.0158	0.0426	0.1678	1.5684	2.4702 (13 iter)
-f 2400 -b 80 -s 8192	0.0167	0.1788	0.6482	0.0012	0.0105	0.0457	0.0491	0.4533	1.4425	2.1364 (4 iter)
-f 5730 -b 60 -s 8192	0.0167	0.1159	0.4765	0.0011	0.0080	0.0314	0.0489	0.3407	1.3669	1.8748 (4 iter)
-f 5755 -b 20 -s 8192	0.0165	0.0502	0.2129	0.0011	0.0034	0.0125	0.0484	0.1453	0.5595	0.7849 (4 iter)

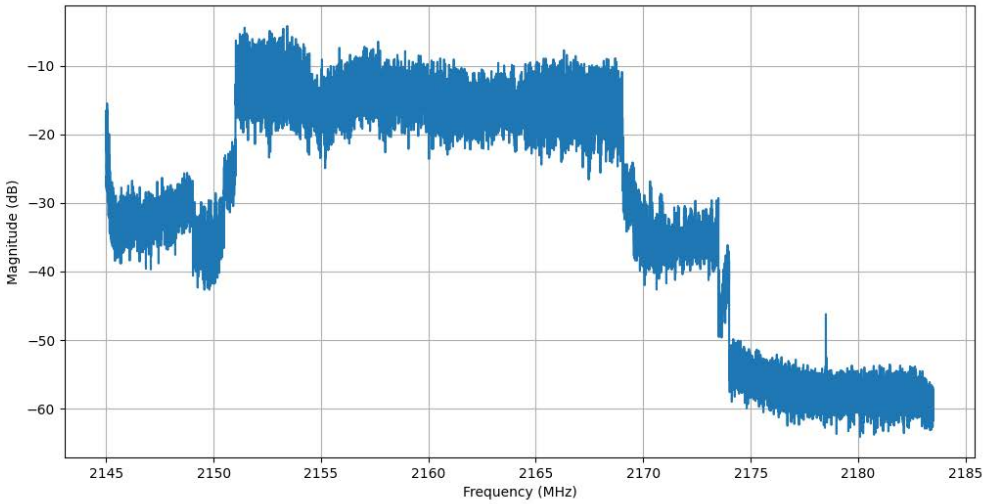
LTE

‘Εχουμε δοκιμάσει και τρία σκαναρίσματα στο LTE. Το πρώτο αφορά το εύρος 2150-2180 MHz με κανάλια 10 MHz, 4 περάσματα και FFTsize 8192. Εδώ χρησιμοποιήσαμε ένα κινητό τηλέφωνο το οποίο μέσω δεδομένων κινητής τηλεφωνίας κατέβαζε βίντεο από το διαδίκτυο. Εκείνη την στιγμή τρέξαμε το φασματικό αναλυτή μας και συλλάβαμε επ αυτοφόρω αυτό το κατέβασμα πληροφορίας όπως δείχνει και το output του προγράμματος μας στις συχνότητες 2150 – 2170 MHz.



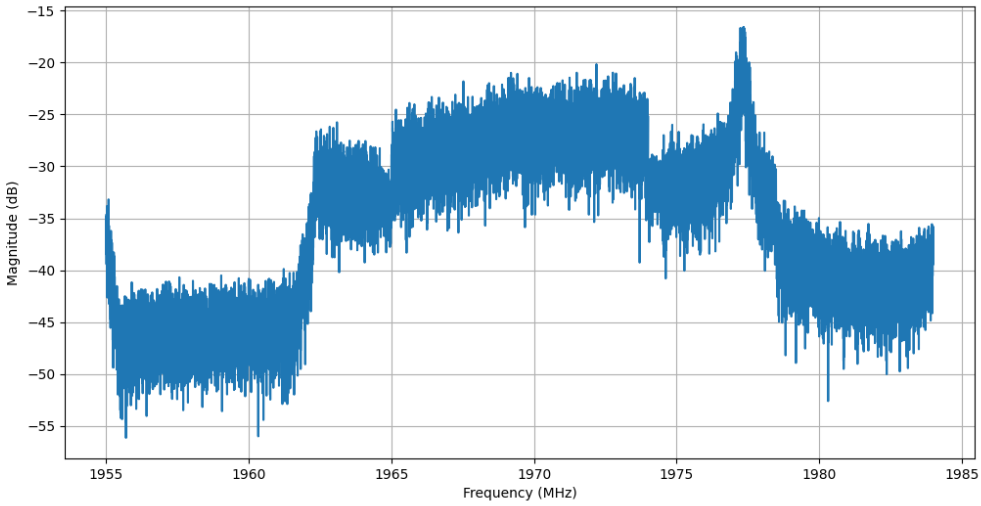
Εικόνα 60: -f 2150 -b 30 -s 8192

Το δεύτερο σκανάρισμα που έχουμε είναι το ίδιο σκανάρισμα με περιορισμό 13 επαναλήψεων το οποίο μας δίνει αυτό το plot.



Εικόνα 61: -f 2150 -b 30 -i 13 -s 8192

Τέλος έχουμε ένα σκανάρισμα σε upload στο εύρος συχνοτήτων 1960-1980 MHz με κανάλια των 10 MHz, FFTsize 8192 με περιορισμό 7 επαναλήψεων . Πιο συγκεκριμένα εδώ τρέξαμε τον φασματικό αναλυτή μας ενώ ταυτόχρονα βάλαμε ένα κινητό τηλέφωνο να κάνει upload πληροφορία μέσω δεδομένων κινητής τηλεφωνίας. Έτσι βλέπουμε κι αυτή τη μετάδοση στο παρακάτω plot.



Εικόνα 62: -f 1960 -b 20 -i 7 -s 8192

Στον παρακάτω πίνακα παρουσιάζουμε αναλυτικά τις χρονικές παραμέτρους του κάθε σκαναρίσματος που παρουσιάσαμε παραπάνω και αφορά το LTE.

Παράμετροι Scan	Sense channel	Sense iteration	Sense scan	Proc. channel	Proc. iteration	Proc. scan	DB channel	DB iteration	DB scan	Total time
-f 2150 -b 30 -s 8192	0.0163	0.0659	0.2777	0.0002	0.0023	0.0541	0.0103	0.0764	0.6629	0.9947 (4 iter)
-f 2150 -b 30 -i 13 -s 8192	0.0164	0.0692	0.9076	0.0002	0.0009	0.0116	0.0095	0.0386	0.5008	1.4200 (13 iter)
-f 1960 -b 20 -i 7 -s 8192	0.0229	0.0559	0.4045	0.0012	0.0035	0.0239	0.0536	0.1649	1.1522	1.5807 (7 iter)

6.6 Παρατηρήσεις

Κοιτάζοντας τις χρονικές παραμέτρους που συγκεντρώσαμε από τα σκαναρίσματα που δοκιμάσαμε παρατηρούμε οτι αύξηση του FFTsize σημαίνει και αύξηση του χρόνου επεξεργασίας καθώς έχουμε περισσότερα δείγματα να επεξεργαστούμε.
Η αποστολή των επεξεργασμένων δειγμάτων στη βάση δεδομένων μας είναι η πιο βαριά χρονικά διαδικασία, ακολουθεί το sensing και το λιγότερο χρόνο του συνολικού χρόνου σκαναρίσματος τον ξοδεύουμε για την επεξεργασία των δειγμάτων μας.

Γενικά είναι καλύτερο να έχουμε μεγάλα κανάλια και λιγότερα σε αριθμό παρά να έχουμε πολλά και μικρότερα. Με το μεγάλο κανάλι θα κάνουμε πολλά περισσότερα περάσματα και έτσι θα έχουμε καλύτερη εικόνα του φάσματος, με τα πολλά κανάλια και μικρότερα σε μέγεθος αυξάνεται και ο χρόνος επεξεργασίας και ο χρόνος που ξοδεύουμε για τη μεταφορά των δεδομένων στη βάση μας.

Επίσης όταν έχουμε πολλά κανάλια έχουμε και περισσότερο συνολικά χρόνο σκαναρίσματος αφού οι παραπάνω χρόνοι συμπεριλαμβάνονται στον υπολογισμό του. Κι όλα αυτά κάνοντας λιγότερα περάσματα σε σχέση με το αν είχαμε λιγότερα κανάλια.

ΚΕΦΑΛΑΙΟ 7 ΕΠΙΛΟΓΟΣ

Στα πλαίσια αυτής της εργασίας αναπτύξαμε έναν φασματικό αναλυτή με χρήση της γλώσσας προγραμματισμού Python και μιας SDR συσκευής, του ADALM PLUTO SDR. Δείξαμε πως με μία σχετικά φθηνή SDR συσκευή και με μία γλώσσα προγραμματισμού μπορούμε να αναλύσουμε σήματα σε ένα σχετικά μεγάλο εύρος συχνοτήτων από 70 MHz εώς 5995 MHz. Μπορέσαμε να οπτικοποιήσουμε το φάσμα συχνοτήτων που ζήτησε ο χρήστης και να ανιχνεύσουμε με ακρίβεια μεταδόσεις που συμβαίνουν σε αυτό το εύρος συχνοτήτων την χρονική περίοδο του σκαναρίσματος. Ανιχνεύσαμε μεταδόσεις FM, Wi-Fi, LTE και σίγουρα δεν περιοριζόμαστε μόνο σε αυτές. Οτιδήποτε μεταδίδεται μεταξύ 70 MHz και 5995 MHz θα μπορούσε να ανιχνευτεί από το πρόγραμμά μας. Αυτή η εργασία τονίζει την ευελιξία και την σημασία των SDR συσκευών στην ανάλυση και επεξεργασία σημάτων. Είδαμε στην πράξη πως με κάποιες αλλαγές στον κώδικα η SDR συσκευή μας μπορεί να κάνει τελείως διαφορετικά πράγματα κι αυτό είναι κάτι που γενικά προσπαθούμε να εκμεταλλευτούμε προς όφελος μας. Κι όλα αυτά με το ίδιο hardware.

7.1 ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Μελλοντικά θα μπορούσαμε να φτιάξουμε ένα Graphical User Interface (GUI) ούτως ώστε να είναι πιο φιλικό προς τον χρήστη και έτσι να μπορεί να χρησιμοποιηθεί κι από άτομα τα οποία δεν έχουν τόσο καλή επαφή με τους ηλεκτρονικούς υπολογιστές. Επίσης καλό θα ήταν να αφαιρέσουμε το DC Spike και άλλα τυχαία spikes που εμφανίζονται λόγω του hardware των SDR συσκευών. Είπαμε και προηγουμένως ότι όλα αυτά δεν αποτελούν πραγματικές μεταδόσεις αλλά είναι φαινόμενα που δημιουργούνται κατά την επεξεργασία των δεδομένων μας στα επιμέρους components. Έτσι καλό θα ήταν να τα αφαιρέσουμε καθώς μπορεί να μας μπερδέψουν. Επιπλεόν, θα μπορούσαμε να απευθύνουμε απευθείας ερωτήματα προς τη βάση δεδομένων και εκείνη να μας επιστρέφει στο πρόγραμμα τα δεδομένα που έχουμε ζητήσει. Αυτό θα μας έδινε πολλές επιπλέον

δυνατότητες εκτός από το κλασσικό σκανάρισμα που ζήτησε ο χρήστης. Θα μπορούσαμε να δούμε δεδομένα περασμάτων, καναλιών ξεχωριστά για να μας βοηθήσουν στην ανάλυση μας.

ΒΙΒΛΙΟΓΡΑΦΙΑ - ΑΝΑΦΟΡΕΣ

[1] https://www.met.nps.edu/~psguest/EMEO_online/module2/module_2_5.html

[2] https://en.wikipedia.org/wiki/Electromagnetic_spectrum

[3] Travis F. Collins, Robin Getz, Di Pu, Alexander M. Wyglinski Software-Defined Radio for Engineers 2018

[4] https://en.wikipedia.org/wiki/Software-defined_radio

[5] https://aerospacereviews.com/home/technology/software-defined-radio/

[6] https://wiki.analog.com/university/tools/pluto

[7] https://github.com/analogdevicesinc/plutosdr-m2k-drivers-win/releases/download/v0.7/PlutoSDR-M2k-USB-Drivers.exe

[8] PySDR: A Guide to SDR and DSP using Python https://pysdr.org/index.html

[9] Σημειώσεις Τηλεπικοινωνιακά Συστήματα με Python, Θωμάς Καμαλάκης https://eclass.hua.gr/modules/document/file.php/DIT149/%CE%A0%CE%B1%CE%BB%CE%B9%CE%AD%CF%82%20%CE%A3%CE%B7%CE%BC%CE%B5%CE%B9%CF%8E%CF%83%CE%B5%CE%B9%CF%82%20%CE%9C%CE%B1%CE%B8%CE%AE%CE%BC%CE%B1%CF%84%CE%BF%CF%82.pdf

[10] Διαφάνειες του μαθήματος Σήματα και Συστήματα του ΤΕΙ Δυτικής Ελλάδος, Μιχάλης Παρασκευάς http://opencourses.teiwest.gr/modules/document/index.php?course=CIED107&openDir=/52246099c8f0

[11] https://en.wikipedia.org/wiki/Signal-to-noise_ratio

[12] Διαφάνειες του μαθήματος Τηλεπικοινωνιακά Συστήματα Ι του ΤΕΙ Δυτικής Ελλάδος, Μιχάλης Παρασκευάς http://opencourses.teiwest.gr/modules/document/index.php?course=CIED106&openDir=/5607f084iUbK

[13] Richard G. Lyons Understanding Digital Signal Processing Third Edition

[14] https://wiki.gnuradio.org/index.php?title=PlutoSDR_Source

ΠΑΡΑΡΤΗΜΑ

Κώδικας

Σε αυτό το σημείο σας παραθέτω τον κώδικα pluto_spectrum_analyzer.py που σε συνδυασμό με το PLUTO SDR κάνει την ανάλυση του φάσματος. Ακολουθούν τα άλλα 2 modules.

pluto_spectrum_analyzer.py

```python
import adi # type: ignore
import argparse
import matplotlib.pyplot as plt
import numpy as np
import pickle
import sys
import time
from create_table import create_raw_table
from connection_manager import db_connection

# This class initializes our command line arguments and checks if they are proper
class Arguments:
    def __init__(self, f, b, c, i, t, s):   # Constructor
        self.frequency, self.bandwidth = self.frequency_bandwidth_check(f, b)
        self.channel = self.channel_check(c)
        self.iterations = self.iterations_check(i)
        self.time = self.time_check(t)
        self.fftsize = self.fftsize_check(s)
        self.num_samples = 35000
        self.final_iterations = 0   # Shows us the real iterations because we update its value during the execution


    # We check if frequency and bandwidth are proper
    def frequency_bandwidth_check(self, f, b):
        try:
            freq = float(f)
        except:
            print("Illegal value of frequency. Program terminates!!!")
            sys.exit()
        try:
            bw = float(b)
        except:
            print("Illegal value of bandwidth. Program terminates!!!'")

        if freq < 70.0:     # Pluto restriction-frequency is between 70 MHz and 6 GHz
            freq = 70.0
        if bw < 0.521:     # Pluto restriction-lowest sample rate is 521 kHz
            bw = 0.53
        if freq + bw > 5995.0:
            print("Sum of frequency and bandwidth must be lower than 5995 MHz. Program terminates!!!")
            sys.exit()
        return freq, bw


    # We check the given channel bandwidth if it is legal or not and we compute it if the user did not give us a value for channel bandwidth
    def channel_check(self, channel):
        if channel == None:     # If user does not provide channel bandwidth
            if self.bandwidth <= 10.0:
                channel = self.bandwidth
                return channel

            # If bandwidth > 10 MHz we divide the bandwidth with 2, 3, 4... until we have a channel <= 10 MHz
            div = 2.0
            channel = self.bandwidth / div
            while channel > 10.0:
                div += 1.0
                channel = self.bandwidth / div
            return int(channel)
        # User provided us channel bandwidth
        try:
            channel = abs(float(channel))   # If user provides us with negative number
        except:
            print("Illegal value of channel bandwidth. Program terminates!!!")
            sys.exit()
```

```
        # Pluto restriction-max sample rate for this project is 10 MHz. For bigger sample rates we lose samples because of USB 2.0
        if channel > 10.0:
            print("This program supports channel bandwidth from 0.53 MHz to 10 MHz. Program terminates!!!")
            sys.exit()
        elif channel < 0.53:
            print("This program supports channel bandwidth from 0.53 MHz to 10 MHz. Program terminates!!!")
            sys.exit()
        elif channel > self.bandwidth:
            print("Channel bandwidth cannot be greater than bandwidth. Program terminates!!!")
            sys.exit()
        return channel


    # We check if the number of iterations is proper
    def iterations_check(self, i):
        if i == None:    # If user did not provide number of iterations
            return -1    # i =-1 means that user does want specific number of iterations
        try:
            iterr = abs(int(i))
        except:
            print("Illegal value of iterations. Program terminates!!!")
            sys.exit()
        if iterr > 300:     # Max iterations = 300
            return 300
        elif iterr == 0:    # Min iterations = 1
            return 1
        else:
            return iterr


    # We check if the time in seconds is proper
    def time_check(self, t):
        if t == None:
            return -1    # time = -1 means there is no time restriction
        try:
            ttime = abs(int(t))
        except:
            print("Illegal value of seconds. Program terminates!!!")
            sys.exit()
        if ttime > 60:
            return 60    # Max scan time = 60 seconds
        elif ttime == 0:
            return -1
        else:
            return ttime


    # We check if the FFTsize is proper (is power of 2 , between 2^5 and 2^15 )
    def fftsize_check(self, s):
        if s == None:
            return 8192     # Default FFTsize value
        try:
            fftsize = abs(int(s))
        except:
            print("Illegal value of FFTsize. Program terminates!!!")
            sys.exit()
        # We check if FFTsize is power of 2
        fftsize_temp = 32
        while fftsize_temp < fftsize:
            fftsize_temp *= 2
        if fftsize_temp > 32768:
            return 32768   # Max FFTsize is 32768
        else:
            return fftsize_temp


# Psd function computes the FFT of our received samples and the Power Spectral Density
def psd(rx_samples, args):
    rx_samples = rx_samples[0:args.fftsize]     # We check only FFTsize samples out of the received samples
    rx_samples = rx_samples * np.hamming(len(rx_samples))   # Apply a Hamming window
    ffted = (np.fft.fft(rx_samples))    # FFT
    power = (abs(ffted))**2
    ps_density = power / ((args.channel * (10**6)) * args.fftsize)
    psd_dB = 10.0 * np.log10(ps_density)    # dBs
    psd_dB_shifted = np.fft.fftshift(psd_dB)    # FFTshift
    return(psd_dB_shifted)


# Saves specific data channel samples in database
def insert_in_database(samples_dbs, iteration_no):
    with db_connection() as conn:
        curr = conn.cursor()
        curr.execute("INSERT INTO samples_database.numpy_raw_data(scan_number, channel_data) VALUES (%s, %s)",
                     (iteration_no, pickle.dumps(samples_dbs))  # For serialization
                )
        conn.commit()


# Scan_bandwidth sets up Pluto SDR for reception and gets the samples in every center_frequency
def scan_bandwidth(sdr, args):
    iteration_samples = []
    num_samps = args.num_samples
    # Config Rx
    sdr.sample_rate = int(sample_rate)
```

```
    sdr.gain_control_mode_chan0 = "manual"
    gain = 50.0
    sdr.rx_hardwaregain_chan0 = gain    # Set receive gain
    sdr.rx_rf_bandwidth = int(sample_rate)   # Set the bandwidth of the reception filter (Hz)
    sdr.rx_buffer_size = num_samps   # Each call of rx() will return num_samps number of samples
    iteration_sense = 0    # Sense time for one iteration
    for freq in center_freqs:
        sdr.rx_lo = int(freq)   # The SDR will scan at this center_frequency
        channel_scan_time_start = time.time()   # Start time of channel sense
        sdr.rx_destroy_buffer()
        rx_samples = sdr.rx()
        channel_scan_time_finish = time.time()   # Finish time of channel sense
        channel_time_sense = channel_scan_time_finish - channel_scan_time_start   # Time of channel sense
        iteration_sense += channel_time_sense
        iteration_samples.append(rx_samples)
    return [iteration_samples, channel_time_sense, iteration_sense]


# For every scan iteration takes samples from all bandwidth and saves them in a list.
# Called when user does not provide iterations and time. Default scan iterations = 4
def no_restricted_scan(sdr, args):
    all_samples = []
    scan_sense = 0   # Scan sense time
    for i in range(1, args.iterations + 1):
        scan_b = scan_bandwidth(sdr, args)
        all_samples.append(scan_b[0])
        channel_time_sense = scan_b[1]   # Channel sense time
        iteration_sense = scan_b[2]     # Iteration sense time
        scan_sense += iteration_sense
    args.final_iterations = args.iterations   # We update the real iterations counter
    return [all_samples, channel_time_sense, iteration_sense, scan_sense]


# For every scan iteration takes samples from all bandwidth and saves them in a list.
# Called when user wants specific number of scan iterations
def iteration_restricted_scan(sdr, args):
    all_samples = []
    scan_sense = 0   # Scan sense time
    for i in range(1, args.iterations + 1):
        scan_b = scan_bandwidth(sdr, args)
        all_samples.append(scan_b[0])
        channel_time_sense = scan_b[1]   # Channel sense time
        iteration_sense = scan_b[2]  # Iteration sense time
        scan_sense += iteration_sense
    args.final_iterations = args.iterations   # We update the real iterations counter
    return [all_samples, channel_time_sense, iteration_sense, scan_sense]


# Scans bandwidth with time restriction
def time_restricted_scan(sdr, args):
    scan_sense = 0   # Scan sense time
    all_samples = []
    scan_b = scan_bandwidth(sdr, args)
    channel_time_sense = scan_b[1]   # Channel sense time
    iteration_sense = scan_b[2]     # Iteration sense time
    scan_sense += iteration_sense
    all_samples.append(scan_b[0])   # Scan once in order to measure time for one iteration
    user_time = args.time - iteration_sense   # Time left after one iteration
    count = 2   # Counts number of iterations
    while(user_time > iteration_sense):   # If we have time we continue scanning
        scan_b = scan_bandwidth(sdr, args)
        if scan_b[2] > user_time:
            break
        channel_time_sense = scan_b[1]
        iteration_sense = scan_b[2]
        scan_sense += iteration_sense
        all_samples.append(scan_b[0])   # Scan once in order to measure time for one iteration
        user_time -= iteration_sense    # Time left after this iteration
        count += 1
    args.final_iterations = count - 1   # We update the real iterations counter
    return [all_samples, channel_time_sense, iteration_sense, scan_sense]


# Scans bandwidth with time and iterations restriction
def iterations_time_restricted_scan(sdr, args):
    scan_sense = 0   # Scan sense time
    scan_b = scan_bandwidth(sdr, args)
    channel_time_sense = scan_b[1]   # Channel sense time
    iteration_sense = scan_b[2]     # Iteration sense time
    scan_sense += iteration_sense
    all_samples.append(scan_b[0])   # Scan once in order to measure time for one iteration
```

```
    user_time = args.time - iteration_sense   # Time left after one iteration
    args.final_iterations += 1   # We update the real iterations counter
    for i in range(2, args.iterations + 1):
        scan_b = scan_bandwidth(sdr, args)
        if scan_b[2] > user_time:
            args.final_iterations = i - 1
            break
        channel_time_sense = scan_b[1]
        iteration_sense = scan_b[2]
        scan_sense += iteration_sense
        all_samples.append(scan_b[0])   # Scan once in order to measure time for one iteration
        user_time -= iteration_sense    # Time left after this iteration
        args.final_iterations += 1   # We update the real iterations counter
    return [all_samples, channel_time_sense, iteration_sense, scan_sense]


# Returns channel data of a specific channel and a specific iteration
def get_channel_data_from_database(channel_id):
    with db_connection() as conn:
        curr = conn.cursor()
        curr.execute(
            """
            SELECT channel_data
            FROM samples_database.numpy_raw_data
            WHERE id=%s;
            """,
            (channel_id, )
        )
        return np.array(pickle.loads(curr.fetchone()[0]))   # Diserialization



if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('-f', help = 'lowest frequency')   # User provides low limit of frequency in MHz
    parser.add_argument('-b', help = 'bandwidth')   # User provides the bandwidth he wants to scan in MHz
    parser.add_argument('-c', help = 'channel')   # User provides channel bandwidth in MHz
    parser.add_argument('-i', help = 'iterations')   # User provides number of iterations of the scan
    parser.add_argument('-t', help = 'time')   # User provides time he wants to scan in seconds
    parser.add_argument('-s', help = 'FFT size')  # User provides specific FFTsize
    args = parser.parse_args()
    arguments = Arguments(args.f, args.b, args.c, args.i, args.t, args.s )

    sdr = adi.Pluto("ip:192.168.2.1")
    sample_rate = arguments.channel * (10**6)   # Hz
    if arguments.channel == arguments.bandwidth:   # If we have just one channel to scan
        f_start = arguments.frequency * (10**6) + (sample_rate / 2)  # First center frequency
    else:   # If we have more than one channel
        f_start = arguments.frequency * (10**6)   # First center frequency
    f_end = arguments.frequency * (10**6) + arguments.bandwidth * (10**6)   # User wants to scan until f_end frequency

    overlap = 0.05 * sample_rate    # 5% overlap between channels
    step_size = sample_rate - overlap   # New center frequency step
    bin = sample_rate / arguments.fftsize   # Bin size
    overlap_bins = int(overlap / bin)   # Number of bins that are overlapped
    center_freqs = np.arange(f_start, f_end, step_size)   # List with all the center frequencies of our channels
    coverage_start = center_freqs[0] - sample_rate / 2   # The real starting frequency of our scan
    coverage_end = center_freqs[-1] + sample_rate / 2   # Coverage end is the ending frequency of our scan
    # Case that we have some more bandwidth to scan, so we need one more channel due to the overlap
    if coverage_end < f_end :
        extra_center_freq = center_freqs[-1] + step_size
        center_freqs = np.append(center_freqs, extra_center_freq)
        coverage_end = center_freqs[-1] + sample_rate / 2   # In that case we update coverage_end

    create_raw_table()   # We then create the database array

    all_samples = []    # All samples from our scan
    # Four different operations
    if arguments.iterations != -1 and arguments.time != -1:  # User provides us with iterations number and time
        scan = iterations_time_restricted_scan(sdr, arguments)
        all_samples = scan[0].copy()
        channel_sense_time = scan[1]    # Time to sense one channel
        iteration_sense_time = scan[2]    # Time to sense one iteration
        scan_sense_time = scan[3]   # Time to sense all iterations of the scan
    elif arguments.iterations != -1 :   # User provides us with iterations number
        scan = iteration_restricted_scan(sdr, arguments)
        all_samples = scan[0].copy()
        channel_sense_time = scan[1]    # Time to sense one channel
        iteration_sense_time = scan[2]    # Time to sense one iteration
        scan_sense_time = scan[3]   # Time to sense all iterations of the scan
    elif arguments.time != -1 :   # User provides us with time
        scan = time_restricted_scan(sdr, arguments)
```

```
        all_samples = scan[0].copy()
        channel_sense_time = scan[1]    # Time to sense one channel
        iteration_sense_time = scan[2]    # Time to sense one iteration
        scan_sense_time = scan[3]   # Time to sense all iterations of the scan
    else :   # User does not provide us with iterations number and time, our scan will consist of 4 iterations
        arguments.iterations = 4    # Default value of iterations
        scan = no_restricted_scan(sdr, arguments)
        all_samples = scan[0].copy()
        channel_sense_time = scan[1]    # Time to sense one channel
        iteration_sense_time = scan[2]    # Time to sense one iteration
        scan_sense_time = scan[3]   # Time to sense all iterations of the scan


    # Here we do signal processing of our samples and transfer data to database
    scan_psd_time = 0
    scan_d_time = 0
    for i in range(0, len(all_samples)):
        iteration_psd_time = 0  # Time to process iteration data
        iteration_d_time = 0   # Time to transfer iteration data to database
        for j in range(0, len(center_freqs)):
            start_psd_channel = time.time()
            samples_dbs = psd(all_samples[i][j], arguments)
            finish_psd_channel = time.time()
            time_psd_channel = finish_psd_channel - start_psd_channel   # Channel processing time
            iteration_psd_time += time_psd_channel
            start_d_channel = time.time()
            insert_in_database(samples_dbs, i+1)
            finish_d_channel = time.time()
            time_d_channel = finish_d_channel - start_d_channel   # Time to transfer channel data to database
            iteration_d_time += time_d_channel
            samples_dbs = []
        scan_psd_time += iteration_psd_time     # Time to process data from all iterations of the scan
        scan_d_time += iteration_d_time     # Time to transfer all data of the scan to database

    channel_total_time = channel_sense_time + time_psd_channel + time_d_channel     # Total time to scan a channel
    iteration_total_time = iteration_sense_time + iteration_psd_time + iteration_d_time     # Time to scan an iteration
    scan_total_time = scan_sense_time + scan_psd_time + scan_d_time     # Total scan time

    # print(f"After {arguments.final_iterations} iterations the total scan time is : {scan_total_time:.4f} seconds.")
    # print(f"Time to sense one single channel is : {channel_sense_time:.4f} seconds.")
    # print(f"Time for processing one single channel is : {time_psd_channel:.4f} seconds.")
    # print(f"TIme to transfer to database data from one single channel is : {time_d_channel:.4f} seconds.")

    # print(f"Time to sense one iteration is : {iteration_sense_time:.4f} seconds.")
    # print(f"Time for processing one iteration is : {iteration_psd_time:.4f} seconds.")
    # print(f"Time for tranfering iteration data to database is : {iteration_d_time:.4f} seconds.")

    # print(f"Time for sensing all iterations of the scan is : {scan_sense_time:.4f} seconds.")
    # print(f"Time for processing of all iterations is : {scan_psd_time:.4f} seconds.")
    # print(f"Time for database transfer of all iteration data is : {scan_d_time:.4f} seconds.")

    # print(f"Time to scan one single channel(sense + processing + database) is : {channel_total_time:.4f} seconds.")
    # print(f"Time to scan one iteration(sense + processing + database) is : {iteration_total_time:.4f} seconds.")
    # print(f"Time to scan all iterations(sense + processing + database) is : {scan_total_time:.4f} seconds.")


    # Scan has finished and dBs have been stored to database. Now we retrieve data from the database and show plots.
    db_list = []   # Contains numpy arrays with average values of dBs for every channel
    for channel in range(1, len(center_freqs) + 1):   # For every channel create average numpy array in dBs
        dbs_temp = np.zeros(arguments.fftsize, dtype = float )
        channel_iter = channel
        for iter in range(1, arguments.final_iterations + 1):   # For every iteration of a specific channel retrieve the dBs
            dbs_temp += get_channel_data_from_database(channel_iter)
            channel_iter += len(center_freqs)
        db_list.append(dbs_temp / arguments.final_iterations)

     # We concatenate all channels together in final_dBs for the plot
    if len(center_freqs) == 1:  # If we have only one channel
        final_dBs = db_list[0]
        # Frequencies for the plot
        frequencies = np.linspace((coverage_start/(10**6)), coverage_end/(10**6), num=(arguments.fftsize * len(center_freqs)))
    else:   # If we have more than one channels we have to fix the overlap issue
        # We use start and end to specify the bins that need to be averaged together from neighbour channels
        start = arguments.fftsize - overlap_bins
        end = overlap_bins
        sum = []
        avg = []
        final_dBs = db_list[0][0:start]
        for i in range(1, len(db_list), 1):
            sum = db_list[i-1][start:] + db_list[i][0:end]
            avg = sum / 2
            if i > 1:
                final_dBs = np.concatenate((final_dBs, db_list[i-1][end:start]))
            final_dBs = np.concatenate((final_dBs, avg))
            sum = []
            avg = []
        final_dBs = np.concatenate((final_dBs,db_list[i][end:]))
```

```
409         # Frequencies for the plot
410         num_freq = (arguments.fftsize + ((len(center_freqs)-1) * (arguments.fftsize - overlap_bins)))
411         frequencies = np.linspace((coverage_start/(10**6)), coverage_end/(10**6), num= num_freq)
412
413
414     # Plotting the spectrum
415     plt.figure(figsize=(12, 6))
416     plt.plot(frequencies, final_dBs)
417     plt.xlabel("Frequency (MHz)")
418     plt.ylabel("Magnitude (dB)")
419     plt.grid(True)
420     plt.show()
```

connection_manager.py

```
1  import contextlib
2  import psycopg2
3
4  db_user = "postgres"
5  db_passwd = "postgres"
6  db_port = "5432"
7  db_name = "samples_database"
8
9  connection_string = f"postgresql://{db_user}:{db_passwd}@localhost:{db_port}/{db_name}"
10
11 @contextlib.contextmanager
12 def db_connection():
13     conn = psycopg2.connect(connection_string)
14
15     try:
16         yield conn
17     except Exception as e:
18         print(e)
19     finally:
20         conn.close()
```

create_table.py

```
1  from connection_manager import db_connection
2
3  sql_create = """
4      DROP TABLE IF EXISTS samples_database.{table_name};
5      CREATE TABLE IF NOT EXISTS samples_database.{table_name}
6      (
7          id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1),
8          scan_number integer,
9          channel_data BYTEA
10     );
11
12     ALTER TABLE IF EXISTS samples_database.{table_name}
13         OWNER TO {db_user};
14     """
15
16 def create_raw_table():
17     with db_connection() as conn:
18         curr = conn.cursor()
19         curr.execute(sql_create.format(table_name = "numpy_raw_data", db_user = "postgres"))
20         conn.commit()
```