KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

YAZILIM TANIMLI RADYO İLE SONOBOY ALICI GERÇEKLENMESİ

YÜKSEK LİSANS TEZİ

Elektronik Müh. Engin KARABULUT

Anabilim Dalı: Elektronik ve Haberleşme Müh.

Danışman: Prof. Dr. Sarp ERTÜRK

KOCAELİ, 2009

KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

YAZILIM TANIMLI RADYO İLE SONOBOY ALICI GERÇEKLENMESİ

YÜKSEK LİSANS TEZİ

Elektronik Müh. Engin KARABULUT

Tezin Enstitüye Verildiği Tarih: 22.05.2009

Tezin Savunulduğu Tarih: 31.07.2009

Tez Danışmanı	Üye	Üye
Prof. Dr. Sarp ERTÜRK	Doç. Dr. Tuna TUĞCU	Doç.Dr.Oğuzhan URHAN
(.....)	(.....)	(.....)

KOCAELİ, 2009

ÖNSÖZ VE TEŞEKKÜR

Bu yüksek lisans tezi kapsamında ülkemizde ilk defa Türkiye Bilimsel ve Teknolojik Araştırma Kurumu (T.Ü.B.İ.T.A.K) Marmara Araştırma Merkezi (M.A.M) Bilişim Teknolojileri Enstitüsü (B.T.E) tarafından geliştirilen sonoboy sistemine yazılım tanımlı bir alıcı geliştirilmiştir. Geleneksel sonoboy alıcılar demodülasyon ve kerteriz analizi işlemlerini analog olarak veya analog-sayısal hibrid yapılarla gerçekleştirdiklerinden dolayı hassas kerteriz değerleri elde edilememektedir. Kerteriz hatalarını azaltmak ve daha hassas analizler yapabilmek amacıyla bu tez çalışmasında yazılım tanımlı radyo ile sonoboy alıcı geliştirilmiştir. Yazılım tanımlı radyo ile demodülasyon ve kerteriz analizi işlemleri sayısal olarak gerçeklenmiştir. Geliştirilen yazılım ile istenilen sonoboy kanalında kayıt yapmak ve kerteriz açısının hesaplanması mümkün olmaktadır.

Bu çalışma, T.Ü.B.İ.T.A.K bünyesindeki B.T.E'de gerçekleştirilmiştir. B.T.E, bilgi teknolojileri alanında uygulamalı araştırmalar ve projeler geliştiren bir kurumdur.

Tezin her aşamasında bana desteğini eksik etmeyen tez danışmanım, çok değerli hocam Sn. Prof.Dr.Sarp ERTÜRK'e, teorik ve pratik olarak desteklerini esirgemeyen Sn. Y.Müh.Kd.Yzb. Özer EROĞLU'na, B.T.E Milli sonoboy projesi yöneticisi Sn. Serdar BİRECİK'e. Ayrıca hayatım boyunca beni destekleyen ve bugünlere getiren babam Semai KARABULUT'a ve annem Perihan KARABULUT'a sonsuz minnet duygularımı sunarım.

İÇİNDEKİLER

ÖNSÖZ VE TEŞEKKÜR i
İÇİNDEKİLER ii
ŞEKİLLER DİZİNİ iv
TABLOLAR DİZİNİ vi
SEMBOLLER vii
ÖZET ix
İNGİLİZCE ÖZET x
1. GİRİŞ 1
2. YAZILIM TANIMLI RADYO 3
2.1. Yazılım Tanımlı Radyo Donanım Mimarileri 4
2.2. Yazılım Tanımlı Radyo Yazılım Mimarileri 5
2.2.1. SCA yazılım tanımlı radyo mimarisi 6
2.2.2. GNU Radio yazılım tanımlı radyo mimarisi 8
3. DIFAR SONOBOY SİSTEMİ 11
3.1. Sonoboyların Tarihçesi 12
3.2. DIFAR Akustik İşaretlerinin Çoklanması ve İletimi 13
3.2.1. Manyetik kerteriz düzeltmesi ile yönlü hidrofonların dikgen genlik modülasyonu 15
3.2.2. DIFAR standartındaki çoklanmış birleşik işaretten hedefin manyetik kuzeye olan kerterizinin bulunması 19
3.2.3. DIFAR standartındaki çoklanmış birleşik işaretin FM modülasyonu 21
3.2.4. DIFAR FM modülasyonu iletim parametreleri 24
4. YAZILIM GELİŞTİRME ORTAMI 26
4.1. Python Programlama Dili 26
4.1.1. Python yorumlayıcısı 27
4.1.2. Python veri tipleri 28
4.1.2.1. Boolean değerler 28
4.1.2.2. Sayısal veri tipleri 28
4.1.2.3. String tipleri 29
4.1.2.4. Dizi Tipleri 29
4.1.3. Kodlama imlası 30
4.1.3.1. Kod blokları 30
4.1.3.2. Kontrol yapıları 30
4.1.3.3. Fonksiyon ve sınıf tanımlamaları 31
4.2. Scipy Python Bilimsel Kütüphanesi 32
4.2.1. Scipy organizasyonu 33
4.3. Matplotlib Python Grafik Kütüphanesi 34
4.3.1. Örnek bir matplotlib uygulaması 35
4.4. GNU Radio Yazılım Tanımlı Radyo Geliştirme Ortamı 36
4.4.1. GNU Radio akış diyagramları 37
4.4.2. GNU Radio veri formatları 38
4.4.3. GNU Radio yazılım geliştirme metotları 38
4.4.4. GNU Radio yazılım örneği 41

4.5. Universal Software Radio Peripheral 42
4.5.1. USRP mimarisi 43
4.5.2. Cypress FX2 mikrodenetleyici 43
4.5.3. ADC ve DAC 44
4.5.4. FPGA 45
4.5.5. TVRX harici genişleme kartı 46
5.YAZILIMIN GERÇEKLENMESİ 48
5.1. Yazılım Tanımlı Radyo İle Sonoboy Alıcının Gerçeklenmesi 49
5.1.1. Yazılım tasarımı 51
5.1.1.1. wfm_rx_block sınıfı 52
5.1.1.2. build_gui GUI fonksiyonu 55
5.2. Temelbant İşaretten Kerteriz Açısının Hesaplanması 58
5.2.1. Yönlü ve yönsüz hidrofonların akustik işaretlerinin elde edilmesi 59
5.2.2. Yönlü ve yönsüz hidrofon işaretlerinden hedefin manyetik kuzeye olan kerterizinin hesaplanması 65
5.2.2.1. Yönlü hidrofon işaretlerinden uygun olmayan örneklerin atılması 65
5.2.2.2. Seçilen örneklerden hedefin manyetik kuzeye olan kerterizinin hesaplanması 67
6. DENEYSEL SONUÇLAR 69
6.1. Saat 11:12:30 İtibariyle I. Sonoboy Ölçüm Sonuçları 70
6.2. Saat 11:16:31 İtibariyle II. Sonoboy Ölçüm Sonuçları 72
6.3. Saat 11:20:46 İtibariyle I. Sonoboy Ölçüm Sonuçları 73
6.4. Saat 11:23:06 İtibariyle II. Sonoboy Ölçüm Sonuçları 74
6.5. Saat 11:30:46 İtibariyle I. Sonoboy Ölçüm Sonuçları 75
6.6. Saat 12:09:23 İtibariyle I. Sonoboy Ölçüm Sonuçları 76
6.7. Saat 12:15:46 İtibariyle III. Sonoboy Ölçüm Sonuçları 77
6.8. Saat 12:25:32 İtibariyle I. Sonoboy Ölçüm Sonuçları 78
6.9. Saat 12:31:46 İtibariyle I. Sonoboy Ölçüm Sonuçları 79
6.10. Saat 12:35:46 İtibariyle II. Sonoboy Ölçüm Sonuçları 80
7. SONUÇLAR VE ÖNERİLER 81
KAYNAKLAR 82
EKLER 84
ÖZGEÇMİŞ 90

ŞEKİLLER DİZİNİ

Şekil 2.1: İdeal bir SDR alıcısı şekli 3
Şekil 2.2: Süperheterodin hibrid SDR mimarisi 4
Şekil 2.3: Sayısallaştırılmış radyo frekansı SDR mimarisi 4
Şekil 2.4: SCA mimarisinin katmanlı yapısı 6
Şekil 2.5: GNU Radio mimarisinin katmanlı gösterimi 8
Şekil 2.6: GNU Radio bloklarının çerçeve yapısı 10
Şekil 3.1: İlk sonoboy AN/CRT-1 ve günümüz AN/SSQ-53F sonoboyu 11
Şekil 3.2: AN/SSQ-53B DIFAR sonoboyu 13
Şekil 3.3: DIFAR yönlü ve yönsüz hidrofon cevabı 14
Şekil 3.4: Yönlü hidrofon-manyetik kuzey ilişkisi 15
Şekil 3.5: Manyetik kerteriz düzeltmesi ile hidrofon işaretlerinin çoklanması ve iletimi 16
Şekil 3.6: AR35 "flux gate" manyetik pusula 16
Şekil 3.7: Eşitlik 3.15'in frekans spektrumu görüntüsü 18
Şekil 3.8: Manyetik kuzey ile hedefin aynı eksende olduğu senaryo 20
Şekil 3.9: FM modülasyonu sembolik gösterimi 22
Şekil 3.10: İletim durumunda demodüle edilmiş sonoboy temelbant spektrumu 23
Şekil 4.1: Örnek bir interaktif python uygulaması 28
Şekil 4.2: Scipy ile bessel fonksiyonların hesaplanması 32
Şekil 4.3: Matplotlib kütüphanesi ile üç boyutlu grafik uygulaması 34
Şekil 4.4: Matplotlib kütüphanesi ile iki boyutlu uygulama örneği 36
Şekil 4.5: GNU Radio akış diyagramı yapısı 37
Şekil 4.6: GNU Radio yazılım katmanları 39
Şekil 4.7: Örnek GNU Radio projesi FFT penceresi 42
Şekil 4.8: USRP anakartı ve TVRX genişleme kartı 42
Şekil 4.9: USRP mimarisi blok diyagramı 43
Şekil 4.10: FPGA işaret işleme blokları 45
Şekil 4.11: DDC blok gösterimi 45
Şekil 4.12: DUC blok gösterimi 46
Şekil 4.13: TVRX genişleme kartı blok diyagramı 47
Şekil 5.1: Hazırlanan test düzeneği 48
Şekil 5.2: Yazılım grafik arayüzü 49
Şekil 5.3: Yazılım akış diyagramı 52
Şekil 5.4: wfm_rx_block sınıfı kurucu fonksiyonu blok gösterimi 53
Şekil 5.5: build_gui GUI fonksiyonu blok diyagramı 56
Şekil 5.6: Örnek bir akış diyagramı kontrolü 57
Şekil 5.7: "start_record_samples" fonksiyonu yapısı 58
Şekil 5.8: "stop_record_samples" fonksiyonu yapısı 58
Şekil 5.9: DIFAR çoklanmış işaretin demodülasyonu 59
Şekil 5.10: Zaman ekseninde gösterilmiş temelbant işareti 60
Şekil 5.11: İlk sonar darbesinin ayrıntılı gösterimi 60
Şekil 5.12: Faz pilotu faz açısının bulunması 62
Şekil 5.13: Çapraz korelasyon sonucu 63

Şekil 5.14: Dikgen taşıyıcıların üretilmesi 63
Şekil 5.15: Zaman ekseninde dikgen taşıyıcıların gösterilmesi 64
Şekil 5.16: QAM demodülasyonu blok diyagramı..... 64
Şekil 5.17: Demodüle edilmiş yönsüz ve yönlü hidrofon işaretleri 65
Şekil 5.18: Yönlü hidrofon işaretlerindeki arasındaki faz kayması 66
Şekil 5.19: (k) eşik değerinin bulunması..... 66
Şekil 5.20: Atılacak örnek indislerinin listesinin çıkartılması 66
Şekil 5.21: Hedefin manyetik kuzeye olan kerterizinin hesaplanması 67
Şekil 6.1: Tatbikat daldırma noktaları..... 69
Şekil 6.2: Zaman ekseninde temelbant işaret ve sonar darbelerinin gösterimi 70
Şekil 6.3: Hesaplanan kerteriz değerleri grafiği (11:12:30)..... 71
Şekil 6.4: Kerteriz hesabı hata yüzdeleri grafiği (11:12:30)..... 71
Şekil 6.5: Hesaplanan kerteriz değerleri grafiği (11:16:31)..... 72
Şekil 6.6: Kerteriz hesabı hata yüzdeleri grafiği (11:16:31)..... 72
Şekil 6.7: Hesaplanan kerteriz değerleri grafiği (11:20:46)..... 73
Şekil 6.8: Kerteriz hesabı hata yüzdeleri grafiği (11:20:46)..... 73
Şekil 6.9: Hesaplanan kerteriz değerleri grafiği (11:23:06)..... 74
Şekil 6.10: Kerteriz hesabı hata yüzdeleri grafiği (11:23:06)..... 74
Şekil 6.11: Hesaplanan kerteriz değerleri grafiği (11:30:46)..... 75
Şekil 6.12: Kerteriz hesabı hata yüzdeleri grafiği (11:30:46)..... 75
Şekil 6.13: Hesaplanan kerteriz değerleri grafiği (12:09:23)..... 76
Şekil 6.14: Kerteriz hesabı hata yüzdeleri grafiği (12:09:23) 76
Şekil 6.15: Hesaplanan kerteriz değerleri grafiği (12:15:46)..... 77
Şekil 6.16: Kerteriz hesabı hata yüzdeleri grafiği (12:15:46)..... 77
Şekil 6.17: Hesaplanan kerteriz değerleri grafiği (12:25:32)..... 78
Şekil 6.18: Kerteriz hesabı hata yüzdeleri grafiği (12:25:32)..... 78
Şekil 6.19: Hesaplanan kerteriz değerleri grafiği (12:31:46)..... 79
Şekil 6.19: Hesaplanan kerteriz değerleri grafiği (12:31:46)..... 79
Şekil 6.21: Hesaplanan kerteriz değerleri grafiği (12:35:46) 80
Şekil 6.22: Kerteriz hesabı hata yüzdeleri grafiği (12:35:46)..... 80

TABLOLAR DİZİNİ

Tablo 2.1: Amplifikatör bloğunun örnek API kümesi..... 7
Tablo 3.1: Sonoboy çalışma frekansları ve kanal numaraları 24
Tablo 4.1: Scipy organizasyon yapısı 33
Tablo 4.2: GNU Radio modülleri ve açıklamaları 39
Tablo 4.3: USRP ADC/DAC parametreleri 44
Tablo 5.1: Kerteriz hesaplanmasında örnek bir senaryo..... 68
Tablo 6.1: Sonoboy parametreleri..... 69
Tablo 6.2: Analiz edilen darbe örnek aralıkları (11:12:30)..... 71
Tablo 6.3: Analiz edilen darbe örnek aralıkları (11:16:31)..... 72
Tablo 6.4: Analiz edilen darbe örnek aralıkları (11:20:46)..... 73
Tablo 6.5: Analiz edilen darbe örnek aralıkları (11:23:06)…..... 74
Tablo 6.6: Analiz edilen darbe örnek aralıkları (11:30:46)..... 75
Tablo 6.7: Analiz edilen darbe örnek aralıkları (12:09:23)..... 76
Tablo 6.8: Analiz edilen darbe örnek aralıkları (12:15:46)..... 77
Tablo 6.9: Analiz edilen darbe örnek aralıkları (12:25:32)..... 78
Tablo 6.10: Analiz edilen darbe örnek aralıkları (12:31:46)..... 79
Tablo 6.11: Analiz edilen darbe örnek aralıkları (12:35:46)..... 80

SEMBOLLER

e_{comp} : DIFAR çoklanmış işaret
e_{cx} : x modüle edici taşıyıcı
e_{cy} : y modüle edici taşıyıcı
e_{f} : Frekans pilotu
e_{mx} : Genlik modüle edilmiş x hidrofon işareti
e_{my} : Genlik modüle edilmiş y hidrofon işareti
e_{O} : Yönsüz(omni) hidrofonun elektriksel cevabı
e_{o} : Yönsüz(omni) hidrofon amplifikatörü çıkışı
e_{p} : Faz pilotu
e_{s} : Akustik hedefin deniz suyuna yaptığı basınç
e_{X} : X eksenine yönlü hidrofonun elektriksel cevabı
e_{x} : X yönlü hidrofon amplifikatörü çıkışı
e_{Y} : Y eksenine yönlü hidrofonun elektriksel cevabı
e_{y} : Y yönlü hidrofon amplifikatörü çıkışı
k_{f} : Modülasyon sabiti
θ : Akustik hedef işaretinin yönlü hidrofonların y eksenine olan açısı
ϕ : Yönlü hidrofonların manyetik kuzeye olan açısı
γ : Akustik hedef işaretinin manyetik kuzeye olan kerterizi
α : Frekans pilotu ve faz pilotu arasındaki faz farkı
τ : Çapraz korelasyon sonucu bulunan τ faz farkı
ρ : Yönlü ve yönsüz hidrofon işaretlerinin açısal frekansı
β : FM modülasyon indeksi
λ : Dalga boyu
ω : Açısal frekans

Kısaltmalar

ADC : Analog Digital Converter (Analog Sayısal Dönüştürücü)
AGC : Automatic Gain Control (Otomatik Kazanç Kontrolü)
AM : Amplitude Modulation (Genlik Modülasyonu)
API : Application Programming Interface (Uygulama Programlama Arayüzü)
B.T.E : Bilişim Teknolojileri Enstitüsü
CCM : CORBA Components Model

CIC : Cascaded Integrator-Comb
CORBA : Common Object Request Broker Architecture
CORDIC : Coordinate Rotation Digital Computer
CPU : Central Processing Unit (Merkezi İşlem Birimi)
DAC : Digital Analog Converter (Sayısal Analog Dönüştürücü)
DDC : Digital Down Converter
DIFAR : Directional Frequency and Ranging Device
DUC : Digital Up Converter
FFT : Fast Fourier Transform (Hızlı Fourier Dönüşümü)
FIR : Finite Impulse Response
FM : Frequency modulation (Frekans Modülasyonu)
FPGA : Field-Programmable Gate Array
GHz : Giga Hertz
GMSK : Gaussian Minimum-Shift Keying
GPIF : General Purpose Interface
GUI : Graphical User Interface (Grafik Kullanıcı Arayüzü)
Hz : Hertz
IDL : Interface Description Language (Arayüz Tanımlama Dili)
IF : Intermediate frequency (Ara Frekans)
IIR : Infinite Impulse Response
JTRS : Joint Tactical Radio System
kHz : Kilo Hertz
LNA : Low Noise Amplifier (Düşük Gürültülü Amplifikatör)
LOFAR : Low Frequency Analysis and Recording
M.A.M : Marmara Araştırma Merkezi
MHz : Mega Hertz
M.I.T : Massachusetts Institute of Technology
NBFM : Narrow-Band Frequency modulation (Darbant Frekans Modülasyonu)
NYP : Nesne Yönelimli Programlama
OFDM : Orthogonal Frequency-Division Multiplexing
PA : Power Amplifier (Güç Amplifikatörü)
PGA : Programmable Gain Amplifiers (Programlanabilir Kazanç Yükselteci)
PSK : Phase Shift Keying
R.C.A : Radio Corporation of America
RF : Radio Frequency (Radyo Frekansı)
QAM : Quadrature Amplitude Modulation (Dikgen Genlik Modülasyonu)
SCA : Software Communications Architecture
SDR : Software Defined Radio (Yazılım Tanımlı Radyo)
SNR : Signal to Noise Ratio (İşaret Gürültü Oranı)
SWIG : Simplified Wrapper and Interface Generator
TCP/IP : Transmission Control Protocol/Internet Protocol
T/R : Transmit / Receive
T.Ü.B.İ.T.A.K : Türkiye Bilimsel ve Teknolojik Araştırma Kurumu
UHF : Ultra High Frequency (Ultra Yüksek Frekans)
USB : Universal Serial Bus
USRP : Universal Software Radio Peripheral
VHF : Very High Frequency (Çok Yüksek Frekans)

YAZILIM TANIMLI RADYO İLE SONOBOY ALICI GERÇEKLENMESİ

Engin KARABULUT

Anahtar kelimeler: Yazılım Tanımlı Radyo, Sonoboy, DIFAR, GNU Radio, Python, USRP

Özet: Günümüzde geleneksel radyo iletişim sistemleri yerlerini hızla yazılım tanımlı radyo sistemlerine bırakmaktadırlar, bunda yazılımın esnekliği ve güncellenebilirlik önemli bir rol oynamaktadır. Bu tez kapsamında yazılım tanımlı radyo ile DIFAR tipi sonoboy alıcı geliştirilmiştir. Geleneksel sonoboy alıcılar demodülasyon ve kerteriz analizi işlemlerini analog olarak veya analog-sayısal hibrid yapılarla gerçekleştirdiklerinden dolayı hassas kerteriz değerleri elde edilememektedir. Kerteriz hatalarını azaltmak ve daha hassas analizler yapabilmek amacıyla bu tez çalışmasında yazılım tanımlı radyo ile sonoboy alıcı geliştirilmiştir. Alıcı yazılımı geliştirilme sürecinde python ve GNU Radio gibi açık kaynak kodlu yazılımlar tercih edilmiştir.

Tez kapsamında geliştirilen yazılım tanımlı sonoboy alıcı ile hem yazılım tanımlı radyo konusunun bir uygulaması yapılmış hemde B.T.E bünyesinde geliştirilen sonoboy projesini tamamlayıcı bir prototip ürün ortaya konulmuştur. Geliştirilen yazılım ile istenilen sonoboy kanalında kayıt yapmak ve manyetik kerteriz açısının hesaplanması mümkün olmaktadır. Gerçek deniz ortamında elde edilen ölçüm sonuçları kabul edilebilir sınırlar içerisindedir.

THE IMPLEMENTATION OF SONOBUOY RECEIVER USING SOFTWARE DEFINED RADIO

Engin KARABULUT

Keywords : Software Defined Radio, Sonobuoy, DIFAR, GNU Radio, Python, USRP

Abstract: Today, software defined radio systems are superseding traditional radio communication systems rapidly. In this process flexibility and upgradeability of software plays an important role. In this thesis project DIFAR type sonobuoy receiver has been developed using software defined radio. Traditional sonobuoy receivers perform demodulation and bearing calculation by means of analog systems or hybrid systems, therefore precision bearing values cannot be obtained. In order to reduce bearing errors and perform precision analysis, a new sonobuoy receiver has been developed using software defined radio based on digital signal processing techniques. In software development process, open source softwares, such as python and GNU Radio are preferred.

With software defined sonobuoy receiver which is developed in this thesis project an application of software defined radio subject has been made and an integral part of sonobuoy project that is developed by B.T.E has been produced. Using this sonobuoy receiver, recording of any sonobuoy channel and calculating of magnetic bearing of unknown target is possible. System test has been made in real sea environment and results of test are in acceptable levels.

1.GİRİŞ

Bu tez kapsamında, SDR (Yazılım Tanımlı Radyo) ile DIFAR tipi sonoboy alıcı yazılımı geliştirilmiştir. Geleneksel sonoboy alıcılar demodülasyon ve kerteriz analizi işlemlerini analog olarak veya analog-sayısal hibrid yapılarla gerçekleştirdiklerinden dolayı hassas kerteriz değerleri elde edilememektedir. Kerteriz hatalarını azaltmak ve daha hassas analizler yapabilmek amacıyla bu tez çalışmasında SDR ile sonoboy alıcı geliştirilmiştir. SDR ile demodülasyon ve kerteriz analizi işlemleri sayısal olarak gerçeklenmiştir. SDR sistemleri, geleneksel olarak donanım bazında gerçeklenen mikser, filtre, modülatör-demodülatör gibi blokları bir kişisel bilgisayarda veya gömülü sistemde yazılım temelinde gerçekleyen bir radyo iletişim sistemidir. İşaret işleme bloklarının yazılım ile gerçeklenmesinden dolayı sistem oldukça esnektir. Tipik bir SDR sisteminin donanım katında, alıcı-verici (transceiver), kaynak işaretinin örneklenmesi için genişbantlı bir ADC (Analog Digital Converter), işaret işleme bloklarında sentezlenen işaretin tekrar analog biçime dönüştürülmesi için genişbantlı bir DAC (Digital Analog Converter) bulunur. Bu donanımları kullanan işaret işleme modülleri ise SDR'nin yazılım ayağını oluşturur.

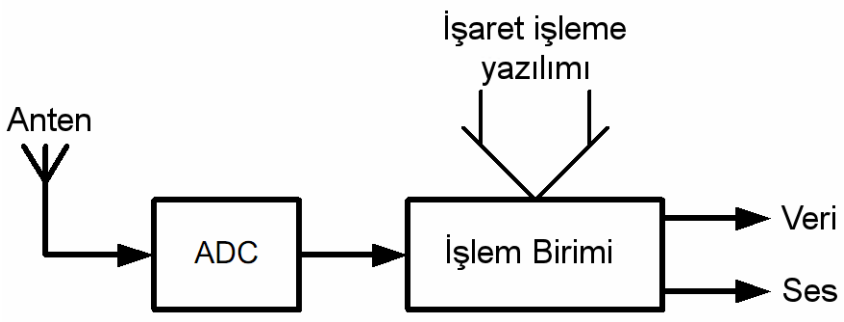
Geliştirilen yazılım için açık kaynak kodlu GNU Radio geliştirme ortamı tercih edilmiştir. Tez kapsamında Ettus firmasından USRP (Universal Software Radio Peripheral) yazılım tanımlı radyo kiti temin edilmiştir. GNU Radio açık kaynaklı kodlu yapısından dolayı geniş bir geliştirici kitlesine sahiptir. Yazılımın geliştirilmesi sırasında bu geliştiricilerin bulunduğu e-posta listelerinden destek alınmıştır.

İkinci bölümde SDR sistemleri tanıtılarak ve çeşitli konseptler ortaya koyulmuştur. Üçüncü bölümde gerçeklenmek istenen DIFAR tipi sonoboy alıcı ile ilgili bilgi ve matematiksel model verilmiştir. Dördüncü bölümde yazılım geliştirilirken kullanılan python programlama dili ve kütüphaneleri, GNU Radio ve USRP konuları incelenmiştir. Beşinci bölümde yazılımın gerçeklenmesi ayrıntısıyla verilmiştir. Altıncı bölümde ise deneysel sonuçlar verilmiştir.

Geliştirilen yazılım ile T.Ü.B.İ.T.A.K M.A.M B.T.E tarafından geliştirilen sonoboy sistemlerine açık kaynak kodlu bir yazılım tanımlı alıcı geliştirilmeye çalışılmıştır. Alınan sonuçlar, prototip aşamasında olan alıcının ileride profesyonel olarak kullanılması için ümit vermektedir.

2.YAZILIM TANIMLI RADYO

Yazılım tanımlı radyo, sayısallaştırılmış radyo işaretlerini yazılım modülleri ile işleyen bir tekniktir. Bu noktada temel amaç geleneksel radyo sistemlerindeki donanımsal bileşenleri işlem gücünün elverdiği kadar yazılım uygulamalarına dönüştürmektir. Yazılım tanımlı radyo kavramının nihai hedefi, her frekans aralığında, işaret bant genişliğinde, modülasyonda, değişen şartlarda adaptif olarak çalışabilecek esnek bir sistem yaratmaktır. Yazılımın esnek yapısı bu noktada anahtar roldedir.



Şekil 2.1: İdeal bir SDR alıcısı şekli

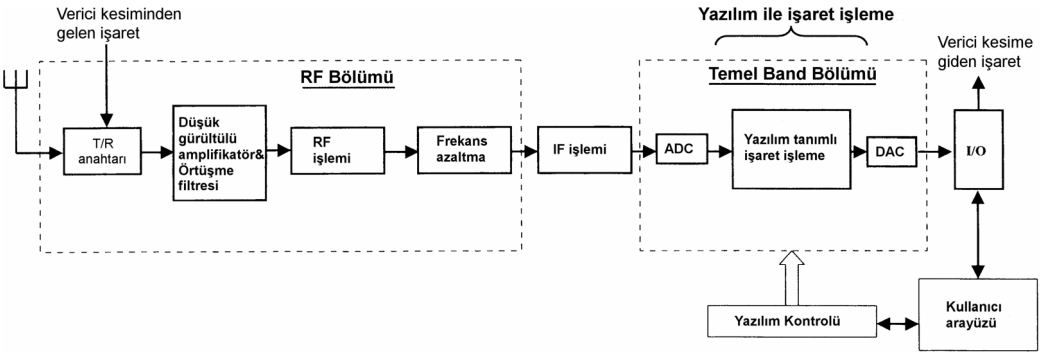
Şekil 2.1'de ideal bir SDR alıcısı verilmiştir. Günümüz teknolojisi ile böyle bir ideal alıcı tasarlamak verimli değildir. Bu yüzden ADC'den önce süperheterodin bir yapı kullanılması tercih edilen bir yöntemdir.

1991 yılında Joseph Mitola'nın bu kavramı ilk olarak ortaya atmasından beri bu hedefe ulaşmak için kapsamlı araştırmalar yapılmıştır. 1997 yılında Amerikan ordusu tarafından başlatılan JTRS (Joint Tactical Radio System) projesi bu konuda öncü durumdadır. Projeye göre 2010 yılında ordunun tüm ses ve veri iletişiminin SDR tabanlı olması amaçlanmıştır. JTRS'nin bir yan ürünü olan SCA (Software Communications Architecture) mimarisi SDR sistemlerine yazılım ve donanım bazında bir standart getirmiştir. SCA sayesinde kodların yeniden kullanılabilirliği ve sistem modellemesi oldukça kolaylaştırılmıştır.

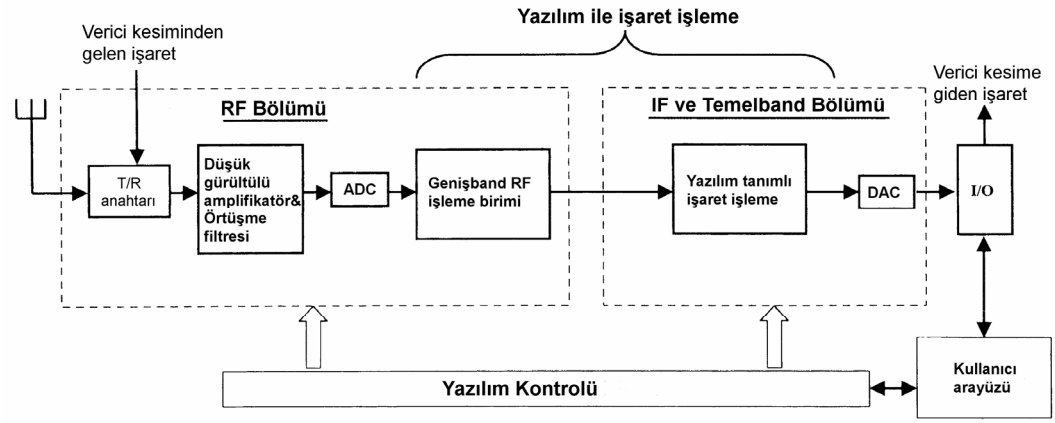
Günümüzde SCA'nın yanısıra GNU Radio gibi açık kaynaklı SDR platformları geliştirilmiştir. Akademik ve ticari projeler için oldukça uygun olan bu platform geniş kullanıcı kitlesi sayesinde hızlı bir ilerleme sergilemektedir.

2.1. Yazılım Tanımlı Radyo Donanım Mimarileri

Yazılım tanımlı radyo sistemleri yazılım/donanım oranına göre çeşitli şekillerde tasarlanabilir [1]. Şüphesiz tercihlerde maliyet ve sistem isterleri önemli rol oynar.



Şekil 2.2: Süperheterodin hibrid SDR mimarisi [1].



Şekil 2.3: Sayısallaştırılmış radyo frekansı SDR mimarisi [1].

Şekil 2.2 ve Şekil 2.3'te SDR sistemleri için iki tür donanım yaklaşımı verilmiştir. İki yaklaşım arasında yazılım/donanım oranı açısından fark vardır. Şekil 2.2'de verilen süperheterodin hibrid SDR mimarisinde ADC, IF (Intermediate Frequency) biriminden sonra gelmektedir. IF frekansı antenden alınan işaretin frekansından düşük olacağından dolayı düşük frekans örneklemeli bir ADC yeterli olacaktır. Bu ise maliyet açısından olumlu bir değişkendir.

RF (Radyo frekansı) işlemlerinin tümü analog olarak süperheterodin bir yapı ile gerçeklenmiştir. Temelbant bölümü ise işaret işleme modülleri kullanılarak yazılımda gerçeklenmiştir. Bu mimaride kısmi bir esneklik vardır. Antenden alınan işareti IF freaknsına indirgeyen RF bölümü donanımsal yapısından dolayı değiştirilemez, örnek olarak IF bant genişliği, çalışma frekans aralığı gibi parametrelerin esnek olarak değiştirilmesi mümkün değildir, bu ancak donanıma müdahale edilmesi ile olabilir. Sonuç olarak adaptif bir yapıdan söz edilemez.

Temelbant bölümü ise yazılım tanımlı olduğundan dolayı modülasyon gibi iletim parametreleri kolaylıkla değiştirilebilir. Bu mimaride işlem gücünden ve maliyetten tasarruf olmasına karşın esneklikten taviz verilmiştir.

Şekil 2.3'de verilen sayısallaştırılmış radyo frekansı SDR mimarisinde ADC antene daha yaklaşmıştır yani ideal yazılım tanımlı radyo alıcısına oldukça yaklaşılmıştır. ADC öncesi sadece düşük gürültülü amplifikatör ve örtüşmeyi önleyici bir filtre kullanılmıştır. Sayısallaştırma RF aşamasında gerçekleştiğinden dolayı IF işlemi genişbantlı RF işleme biriminde gerçeklenmiştir. Bu mimaride çalışma frekans aralığı , IF bant genişliği dinamik olarak yazılım ile değiştirilebilmektedir, bu durumda sistem oldukça esnek olmakla birlikte süperheterodin hibrid SDR mimarisine göre daha yüksek bir işlem gücü gerektirir.

RF işleme birimindeki esneklik bilişsel radyo (Cognitive Radio) için anahtar bir rol oynamaktadır. Genişbant RF işleme biriminde eklenecek yapay zeka desteği ile bilişsel radyonun ihtiyaç duyduğu adaptif yetenekler sağlanabilir. Yakın bir gelecekte ADC/DAC birimlerinde ve işlem gücünde meydana gelecek gelişmelerle birlikte bu mimarinin popüler olacağı öngörülmektedir.

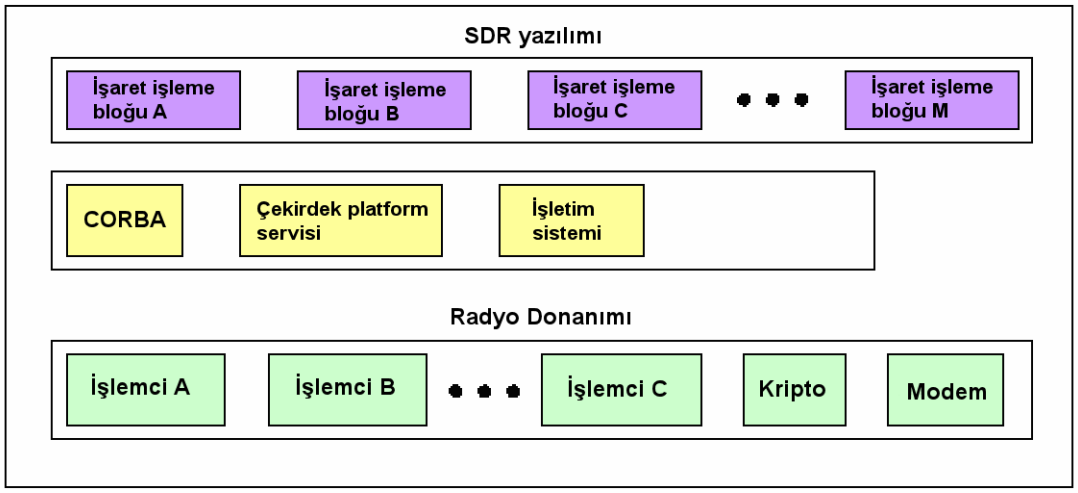
2.2. Yazılım Tanımlı Radyo Yazılım Mimarileri

Yazılım tanımlı radyolar genelde standardize edilmiş yazılım platformları ile tasarlanır.

Bunlara örnek olarak SCA, GNU Radio, OSSIE verilebilir. Bu bölümde kısaca SCA ve GNU Radio platformları verilecektir.

2.2.1 SCA yazılım tanımlı radyo mimarisi

SCA(Software Communications Architecture) [4], donanım ve yazılım modüllerinin yazılım tanımlı radyoda uyum içinde çalışmalarını sağlayan açık bir mimaridir. SCA, Amerikan ordusunun JTRS yazılım tanımlı radyosu için geliştirilmiştir. SCA destekleyen her donanım aynı standart çalışma ortamını sunduğu için yazılımın tekrar kullanılabilirliği oldukça kolaydır. Şekil 2.4'te SCA mimarisinin katmanlı yapısı verilmiştir.



Şekil 2.4: SCA mimarisinin katmanlı yapısı [4].

SCA, nesne yönelimli teknikler, CORBA (Common Object Request Broker Architecture) ve CCM (CORBA Components Model) ile inşa edilmiştir. SCA, SDR uygulaması yazılım bileşenleri arasındaki API (Uygulama programlama arayüzü) lerini tanımlamak için Şekil 2.4' te verilen blok yapısını sunar. API tanımlaması için anahtar bir rol oynayan bu blok yapısı bileşen seviyesinde yeniden kullanımı kolaylaştırır. Bunun yanında geliştiricilere bu arayüzleri tanımlarken oldukça esneklik sunar. SCA yazılım bileşenleri arasındaki bu API arayüzlerini standartlaştırmak için endüstri standartı IDL (Arayüz Tanımlama Dili) kullanır.

SCA'da tanımlanan bu arayüz tanımlama arayüzleri işaret işleme bloklarının bağımsız olarak geliştirilmesini ve daha önemlisi bu işaret işleme bloklarının SCA uyumlu radyo platformları arasında kolaylıkla taşınmasını sağlar.

Donanım tarafından bakıldığında bir çok SCA uyumlu radyonun RF ve modem altsistemleri için benzer donanımlar kullandığı görülür. Örnek olarak LNA (Düşük gürültülü amplifikatör) ve PA (güç amplifikatörü) vazgeçilmez donanımlardır. Bu varsayımdan hareketle donanım ve yazılım bileşenlerini belirli bloklara ayırabiliriz. Örnek olarak LNA ve PA 'yı tanımladığımız bir bloğun diğer işaret işleme blokları ile iletişim kurduğu IDL ile oluşturulmuş API'si tablo 2.1'de verilmiştir.

Tablo 2.1: Amplifikatör bloğunun örnek API kümesi

Tanımlayıcı	**Tipi**	**Açıklaması**
dutyCycle	değişken	belirli bir amplifikatör için "duty cycle" değeri
maxGain	değişken	belirli bir amplifikatör için maksimum operasyonel kazanç değeri
minGain	değişken	belirli bir amplifikatör için minimum operasyonel kazanç değeri
gain	değişken	mevcut kazanç değeri
setGain	fonksiyon	mevcut kazanca yeni değer veren fonksiyon
getGain	fonksiyon	mevcut kazancı okuyan fonksiyon
getDutyCycle	fonksiyon	mevcut "duty cycle" değerini okuyan fonksiyon
getMaxGain	fonksiyon	amplifikatörün desteklediği maksimum kazancı okuyan fonksiyon
getMinGain	fonksiyon	amplifikatörün desteklediği minimum kazancı okuyan fonksiyon

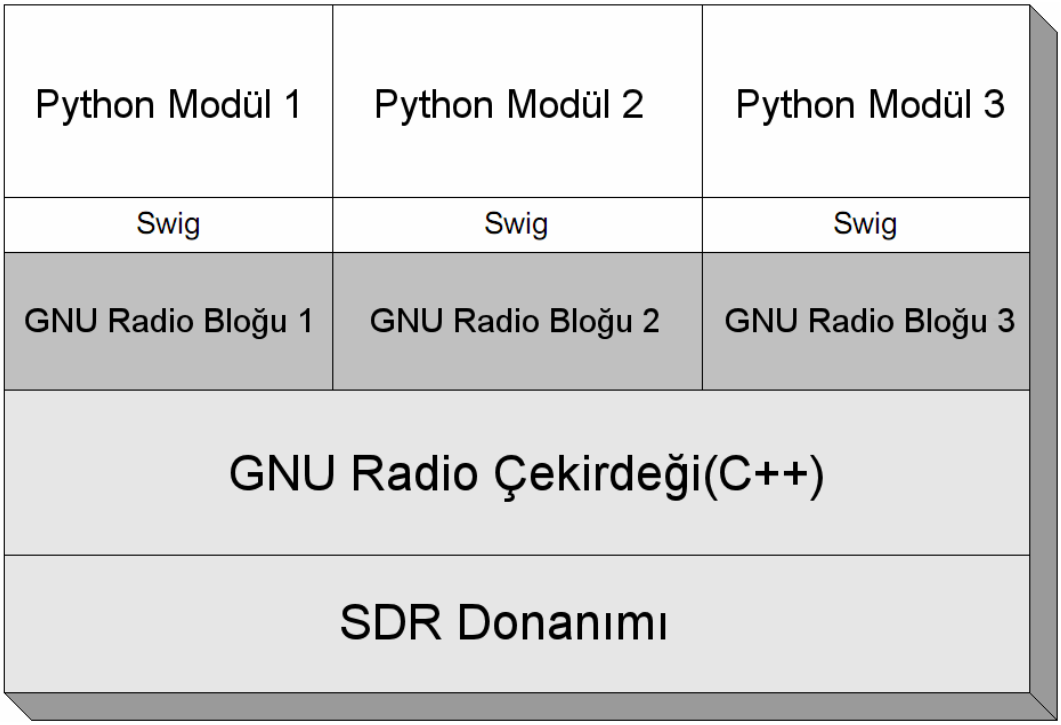
SCA'nın oluşturulmasında aşağıda verilen hedefler amaçlanmıştır.

- Yaygın açık mimari: Standartlaştırılmış açık bir mimarinin kullanılması, ortak çalışma, yazılımın tekrar kullanılabilirliği, yazılım güncelleme ve ölçeklenebilirlik avantajlarına sahiptir.
- Esnek yapı: SCA destekleyen donanımlar yer, hava, deniz radyo standartlarını geniş bir aralıkta destekleyebilir.
- Çoklu bant desteği: Bölüm 2.1'de değinilen sayısallaştırılmış radyo frekansı mimarisi ile istenilen RF frekansında ve IF bant genişliğinde SDR sistemleri tasarlanabilir.

- Uyumluluk: SCA radyoları esnek yapısı sayesinde donanım ile gerçeklenen geleneksel radyo sistemleri ile iletişim kurabilir.
- Yazılım güncellemesi: SCA destekli SDR cihazlarına yazılım tanımlı yapısı sayesinde yeni teknolojiler kolaylıkla yazılım güncellemesi ile uygulanabilir.
- Yazılımın yeniden kullanılabilirliği: SCA'nın SDR kavramına getirdiği en büyük yenilik yazılımın yeniden kullanılabilirliğidir. Bu sayede yazılım geliştirme süresi ve maliyetler azalmaktadır.

2.2.2 GNU Radio yazılım tanımlı radyo mimarisi

GNU Radio [5], 1998 yılında açık kaynak kodlu bir yazılım tanımlı radyo platformu oluşturmak amacıyla başlatılan bir projedir. John Gilmore ismindeki hayırsever Eric Blossom'a 320000$ bağışlayarak projenin başlamasını sağlamıştır. GNU Radio yazılım tanımlı radyo uygulamalarını kapsayan işaret işleme kütüphanesidir. Başlangıçta M.I.T'deki SpectrumWave projesi tarafından geliştirilen Pspectra yazılımının bir türevi olarak geliştirilmiştir.



Şekil 2.5: GNU Radio mimarisinin katmanlı gösterimi [6].

GNU Radio C++ ve python programlama dillerinde gerçeklenmiştir. Python ve C++ arasındaki köprü SWIG (Simplified Wrapper and Interface Generator) ile kurulmuştur. C++, düşük seviyeli işaret işleme bloklarında ve donanım destek bloklarında kullanılmaktadır. Yüksek seviyeli python programlama dili ise c++'da geliştirilen bu blokları birbirine bağlamak ve arayüz geliştirmek için kullanılmaktadır. Bu yaklaşım c++'ın yüksek hızı ile python'un kullanım kolaylığını birleştirmektedir. Şekil 2.5'te GNU Radio katmanlı yapısı verilmiştir. Bir kere bu bloklar birbirine istenilen şekilde bağlandığında GNU Radio'nun işlem yöneticisi kaynaktan alınan örnekleri bloklarda sırayla işletir ve sonucu dosyaya veya bir çıkış birimine aktarır. Bu çıkış birimi ses kartı veya USRP olabilir.

GNU Radio yazılım tanımlı radyonun ihtiyaç duyabileceği tüm kütüphaneleri içermektedir. Bunlardan bazıları,

Modülasyon:

- GMSK
- PSK
- OFDM
- FM, NBFM , AM …

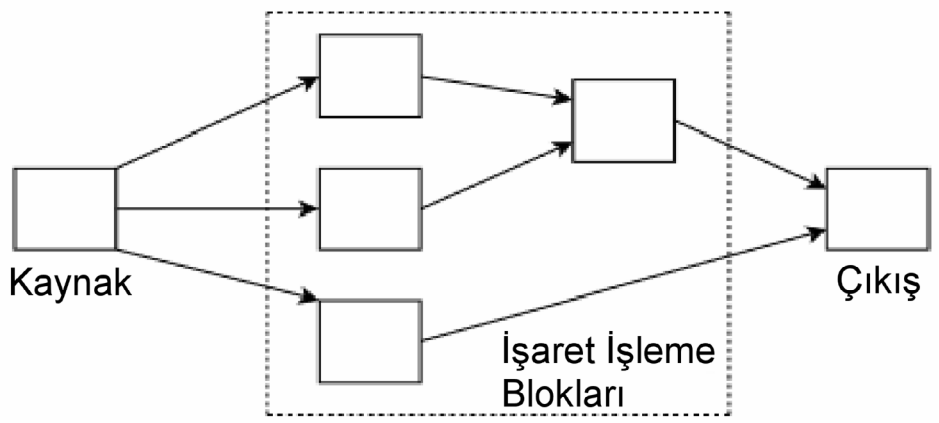
Hata düzeltme kodları:

- Reed-Solomon
- Viterbi
- Turbo kodlar

İşaret işleme yapıları:

- Çeşitli filtreler (FIR,IIR…)
- FFT
- Dengeleyiciler (equalizers)
- İşlem yöneticileri (scheduler)

GNU Radio bloklarını diyagram yapısı ile ifade edebiliriz. Bir diyagramı gerekli blokların eklendiği ve birleştirildiği bir çerçeve olarak düşünebiliriz. Şekil 2.6'da bu çerçeve yapısı sembolik olarak gösterilmiştir. Kaynak tarafından gönderilen veri akışı istenilen şekilde birbirine bağlanmış işaret işleme bloklarından geçerek çıkış



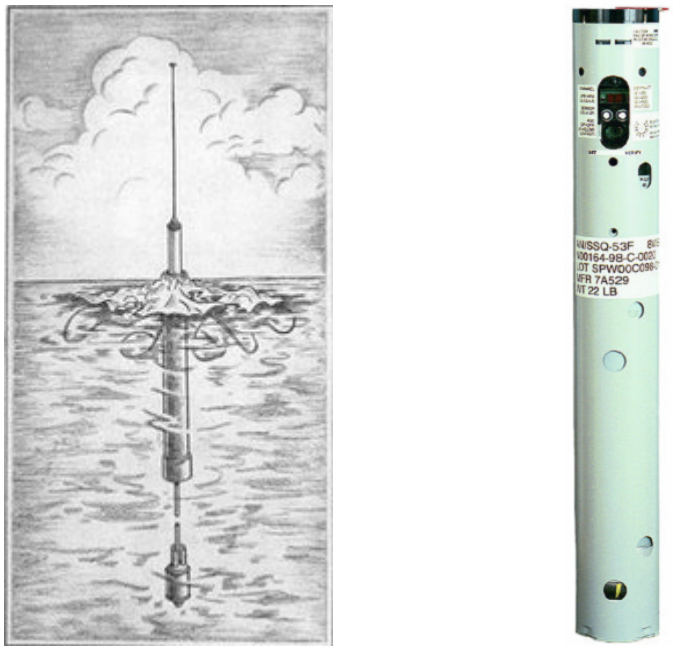
Şekil 2.6: GNU Radio bloklarının çerçeve yapısı

birimine ulaşmaktadır. Bu işlemlerden GNU Radio'nun işlem yöneticisi (scheduler) sorumludur.

Bu yüksek lisans tezinde ele alınan "yazılım tanımlı radyo ile sonoboy alıcı gerçeklenmesi" projesinde açık kaynaklı yapısı, ücretsiz olması ve e-posta listeleri ile ücretsiz yardım alınabildiği için GNU Radio platformu tercih edilmiştir.

3. DIFAR SONOBOY SİSTEMİ

Sonoboy uçak veya yüzen gemilerden denize atılarak denizaltıların yerini tespit etmek amacıyla kullanılan bir sonar sistemidir. Su altı savaşlarının yanısıra su altı akustik araştırmalar diğer bir uygulama alanıdır.



Şekil 3.1: İlk sonoboy AN/CRT-1 ve günümüz AN/SSQ-53F sonoboyu [8].

Sonoboylar üç kategoride sınıflanabilir. Bunlar:

- Pasif sonoboylar: Pasif sonoboylar su altında sadece dinleme yaparlar, bu yüzden tespit edilmeleri zordur. Su üstü gemileri ve su altında denizaltılardan kaynaklanan mekanik gürültüleri, mürettebat tarafından kaynaklanan sesleri, nükleer denizaltılardan kaynaklanan soğutma sistemi gürültülerini hidrofonlar ile dinleyerek bu bilgiyi VHF/UHF radyo vericisi ile uçak veya su üstü gemilerinde bulunan alıcılara iletirler.
- Aktif sonoboylar: Aktif sonoboylar su altında akustik işaretler yayarak bunların geri dönüş yönü ve zaman gecikmesinden kerteriz ve uzaklık bilgilerini hesaplayarak su üstü gemisi veya uçaktaki alıcıya VHF/UHF radyo vericisi ile

iletirler.

- Özel amaçlı sonoboylar: Özel amaçlı sonoboyların yer ve deniz bilimleri alanında bilimsel araştırma maksadıyla kullanılması yaygındır. Örnek olarak su altı sismik hareketleri dinlemede ve deniz suyunun niteliğinin belirlenmesinde kullanılar. Bir diğer kullanım alanı ise deniz kazalarında kaza alanına atılarak periyodik RF işaret göndererek kaza alanının işaretlenmesini sağlarlar.

3.1. Sonoboyların Tarihçesi

3 Eylül 1939'da İngiltere ve Fransa Almanya'ya karşı savaş açtıklarını ilan ederler ve Atlantik'teki büyük mücadele başlar. Savaşın başında Almanya sadece 57 U-boat adı verilen denizaltılarına sahipti, fakat bu sayı kısa sürede artmıştır. Buna güvenen Almanya her türlü gemiye karşı ani saldırılara başlamıştır. 1941 yılı boyunca müttefik kuvvetlere 1118 U-boat saldırısı gerçekleşmiştir, buna karşılık yakalanabilen U-Boat sayısı sadece 30 adettir.

11 Ocak 1942'de Amerikan deniz kuvvetlerine ait bir geminin U-boat'lar tarafından ilk kez vurulmasından sonra ABD başkanı Franklin Roosevelt'in danışmanı Vannevar Bush tarafından su altı tehditlerini tespit amacıyla bir radyo vericili su altı akustik algılayıcı geliştirilmesi fikri ortaya atılmıştır.

RCA (Radio Corporation of America) tarafından geliştirilen ilk sonoboy 7 Mart 1942 tarihinde New London açıklarında S-20 denizaltılarını tespit etmek için denenmiştir. Deney sonucu üç deniz miline kadar denizaltılar tespit edilmiş ve beş deniz miline kadar radyo yayını başarıyla alınmıştır. Bu prototiplerden sonra AN/CRT1 ilk seri üretim sonoboy geliştirilmiştir. Şekil 3.1'de AN/CRT1 verilmiştir. Bu ilk sonoboy yönsüz bir hidrofona ve 60-72 MHz aralığında 6 kanallı yarım watt gücünde radyo vericisine sahipti.

Savaş sonrası sonoboylar sovyet denizaltılarını tespit etmek amacıyla kullanılmaya başlandı. 1965 yılına kadar birçok sonoboy geliştirildi fakat bunlar hep LOFAR (Low Frequency Analysis and Recording) tipi yönsüz sonoboylardı. LOFAR sonoboylar yönsüz olduklarından sualtı tehditlerini algılıyor fakat yönünü tek başına

tespit edemiyorlardı. Üç veya daha fazlasını kombine ederek konum tespiti yapmak mümkün olabiliyordu. Bu zorlukları aşmak ve daha az sonoboy ile konum tespiti yapmak amacıyla 1965 yılında AN/SSQ-53 serisi DIFAR (Directional Frequency and Ranging) sonoboyu geliştirilmiştir.



Şekil 3.2: AN/SSQ-53B DIFAR sonoboyu [9].

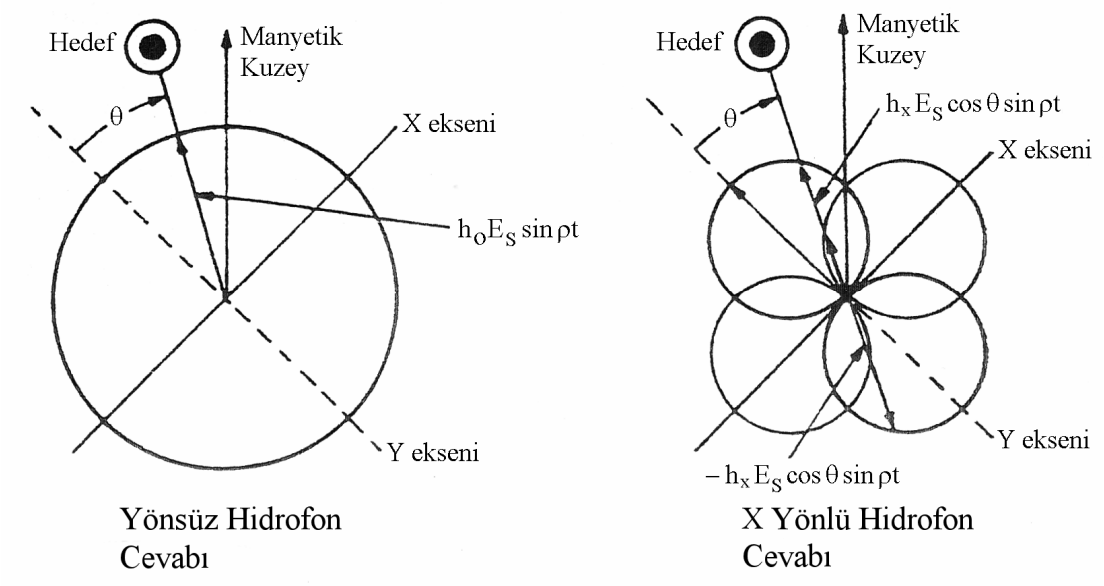
Bir adet yönsüz(omni) ve iki adet yönlü hidrofona sahip DIFAR sonoboyları ile konum tespiti yapmak mümkündür, ayrıca birden fazla DIFAR sonoboy ile hesaplanan kerterizin hata yüzdesi azaltılabilir.

3.2 DIFAR Akustik İşaretlerinin Çoklanması ve İletimi

Su altı akustik hedefinin manyetik kuzeye olan kerterizini hesaplamak için yönsüz ve yönlü hidrofonlardan gelen verilerin hatasız bir biçimde uçakta veya su üstü gemisinde bulunan sonoboy alıcıya iletilmesi gerekmektedir. Bu görev aşağıda verilen üç kıstası gerektirir.

- Yönlü akustik algılayıcı sisteminin geliştirilmesi.
- Bu algılayıcı sistemine manyetik kuzeyin referans olarak entegre edilmesi.,
- Sonoboy alıcıda akustik algılayıcı sistemi verisinin hatasız olarak tekrar elde edilmesi.

Şekil 3.3'te akustik hedefe olan DIFAR hidrofon cevabı verilmiştir. İki yönlü hidrofon x-y eksenine birbirinin sağına düşecek şekilde 180°'lik bir açıyla yerleştirilmiştir. X hidrofonu çıkışı, akustik hedef tam Y eksenine düştüğü zaman maksimum olur, hedef X eksenine düştüğü zamanda Y hidrofon çıkışı maksimum olur.



Şekil 3.3: DIFAR X-yönlü ve yönsüz hidrofon cevabı [9].

Yönlü ve yönsüz hidrofon cevaplarını matematiksel olarak aşağıdaki gibi ifade edebiliriz [9],

Akustik hedefin suya yaptığı akustik basınç,

$$e_S = E_S \sin \rho t \tag{3.1}$$

olarak ifade edilir, bu denklemde ρ, eşitlik (3.2)'deki gibi ifade edilebilir.

$$\rho = 2\pi f \tag{3.2}$$

f, yani hidrofonların çalışma frekansı DIFAR standartında 5-2400 Hz aralığıdır.

İki yönlü hidrofonların elektriksel cevapları,

$$e_X = h_X e_S \cos\theta = h_X E_S \sin \rho t \cos\theta \tag{3.3}$$

$$e_Y = h_Y e_S \sin\theta = h_Y E_S \sin \rho t \sin\theta \tag{3.4}$$

olarak verilir.

Benzer şekilde yönsüz (omni) hidrofonun hedeften gelen akustik su basıncına sabit bir cevabı vardır. Eşitlik (3.5)'te yönsüz hidrofonun elektriksel cevabı verilmiştir.

$$e_O = h_O E_S \sin \rho t \tag{3.5}$$

Yönlü ve yönsüz hidrofonların elektriksel cevap eşitliklerindeki h_X, h_Y, h_O

katsayıları hidrofonların hassasiyetinin ayarlandığı devre elemanlarının katkısıdır. Bu hassasiyet ayarının nedeni modülatör kazançlarının eşit olmadığı durumlarda, bunları dengelemektir.

Eğer dengeli ideal bir modülatör kullandığımızı kabul edersek ($h_X = h_Y = h_O = 1$), hidrofonların ön kuvvetlendirici çıkışları,

$$e_o = E_S \sin \rho t \tag{3.6}$$

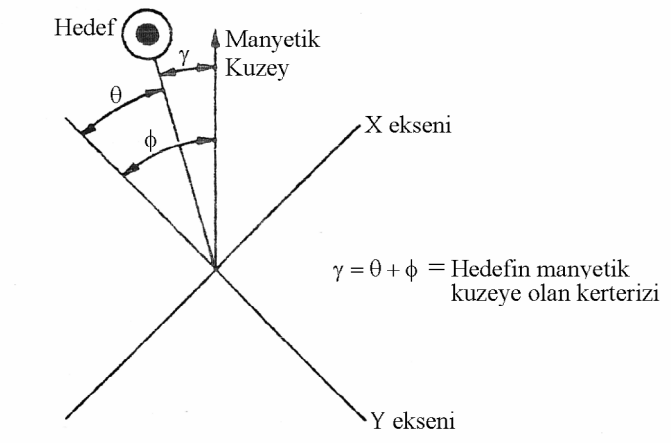
$$e_x = E_S \cos\theta \sin \rho t \tag{3.7}$$

$$e_y = E_S \sin\theta \sin \rho t \tag{3.8}$$

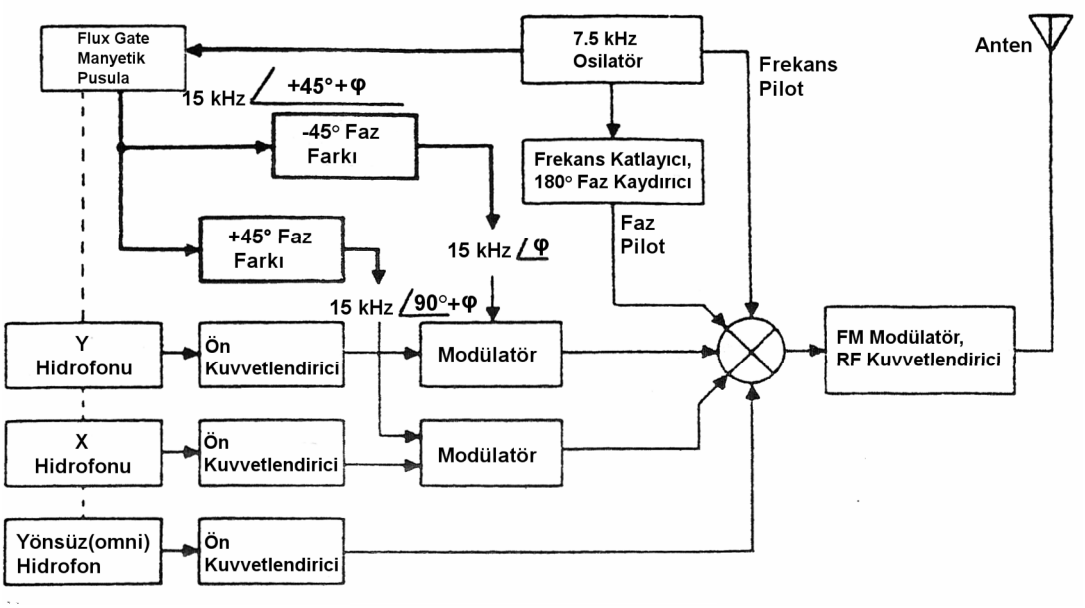
eşitliklerine dönüşür.

3.2.1 Manyetik kerteriz düzeltmesi ile yönlü hidrofonların dikgen genlik modülasyonu

Sonoboyun deniz suyunda dairesel hareketi kontrol edilemediği için yönlü hidrofonların X ve Y eksenleri için bir coğrafi referans tayin edilemez, bu ise su altı hedefin doğru konumunun bulunmasını imkansız kılar. Bu yüzden DIFAR sonoboya bir elektronik pusula eklenmiştir. Bu pusula ile manyetik kuzey noktası, X ve Y hidrofon koordinatlarına referans olarak tayin edilmiştir. Şekil 3.4'te yönlü hidrofonların manyetik kuzey ile ilişkisi verilmiştir.



Şekil 3.4: Yönlü hidrofon-manyetik kuzey ilişkisi [9].



Şekil 3.5: Manyetik kerteriz düzeltmesi ile hidrofon işaretlerinin çoklanması ve iletimi [9].

Şekil 3.5'te manyetik kerteriz düzeltmesi ile çoklanma blok diyagramı verilmiştir. Şekil 3.5'teki "flux gate" manyetik pusulaya 7.5 kHz osilatörden kare dalga formunda işaret uygulandığında, manyetik pusula manyetik kuzeye olan mekanik pozisyona göre giriş işaretinin frekansının iki katında φ kadar faz farkına sahip çıkış işareti üretmektedir. Bu çıkışa daha sonra +45° ve -45° faz farkı eklenerek yönlü hidrofonların dikgen genlik modülasyonu için iki taşıyıcı üretilir.



Şekil 3.6: AR35 "flux gate" manyetik pusula [10].

Sonoboyun üretim aşamasında yönlü hidrofonların Y ekseni ile manyetik kuzey çakıştırılmıştır. Buna göre Y ekseni manyetik kuzeyi gösterdiği zaman φ derecesi 0°

olacaktır. Bu kalibrasyondan sonra artık sonoboy su içinde dönse bile manyetik kuzey referansından dolayı kerteriz hatasız olarak hesaplanabilir.

Şekil 3.5'deki blok diyagramını matematiksel olarak aşağıdaki sıra ile ifade edebiliriz:

Eşitlik (3.7) ve (3.8) 'i

$$\begin{aligned} e_x &= E_S \cos\theta \sin\rho t \\ &= \frac{E_S}{2}\left[\sin(\rho t + \theta) + \sin(\rho t - \theta)\right] \\ e_y &= E_S \sin\theta \sin\rho t \\ &= \frac{E_S}{2}\left[\cos(\rho t - \theta) - \cos(\rho t + \theta)\right] \end{aligned} \tag{3.9}$$

şeklinde yazabiliriz.

Şekil 3.5'te ifade edilen frekans pilotu frekans modülasyonu ile iletim aşamasında temelbant'da oluşabilecek frekans kayıklığını alıcıda belirlemek için kullanılan referans bir pilot işaretidir ve

$$e_f = E_f \cos(\omega t - \alpha) \tag{3.10}$$

eşitliği ile verilir.

Faz pilotu ise QAM (dikgen genlik modülasyonu) ile modüle edilmiş yönlü hidrofonları demodüle etmek için alıcıda kullanılacak taşıyıcı pilot işaretidir ve

$$e_p = -E_c \cos(2\omega t) \tag{3.11}$$

eşitliği ile verilir.

QAM modülasyonu için kullanılacak birbirine dikgen taşıyıcılar,

$$\begin{aligned} e_{cx} &= \sin(2\omega t + \phi) \\ e_{cy} &= \cos(2\omega t + \phi) \end{aligned} \tag{3.12}$$

eşitliği ile verilir.

(3.9) ve (3.12) eşitlikleri kullanılarak x,y hidrofonlardan elde edilen işaretler QAM modülasyonuna tabi tutulur.

$$\begin{aligned} e_{my} &= e_y e_{cy} = \frac{E_S}{2}\cos(2\omega t+\phi)[\cos(\rho t-\theta)-\cos(\rho t+\theta)] \\ &= \frac{E_S}{4}\begin{bmatrix}\cos(\rho t-\theta+2\omega t+\phi)+\cos(\rho t-\theta-2\omega t-\phi)\\ -\cos(\rho t+\theta+2\omega t+\phi)-\cos(\rho t+\theta-2\omega t-\phi)\end{bmatrix} \\ e_{mx} &= e_x e_{cx} = \frac{E_S}{2}\sin(2\omega t+\phi)[\sin(\rho t+\theta)+\sin(\rho t-\theta)] \\ &= \frac{E_S}{4}\begin{bmatrix}\cos(\rho t+\theta-2\omega t-\phi)-\cos(\rho t+\theta+2\omega t+\phi)\\ +\cos(\rho t-\theta-2\omega t-\phi)-\cos(\rho t-\theta+2\omega t+\phi)\end{bmatrix} \end{aligned} \tag{3.13}$$

Birbirine dikgen taşıyıcılarla modüle edilmiş yönlü hidrofon işaretleri toplanırsa,

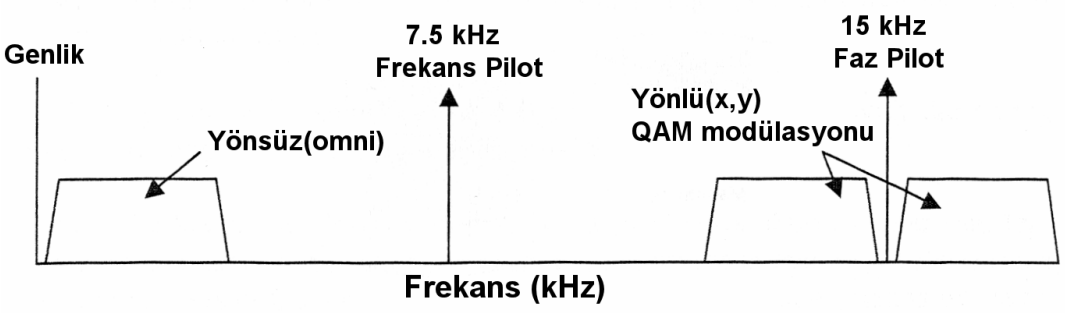
$$\begin{aligned} e_{my}+e_{mx} &= \frac{E_S}{2}[\cos(\rho t-\theta-2\omega t-\phi)-\cos(\rho t+\theta+2\omega t+\phi)] \\ &= E_S \sin\rho t\sin(2\omega t+\theta+\phi) \\ &= E_S \sin\rho t\sin(2\omega t+\gamma) \end{aligned} \tag{3.14}$$

olarak yazılır. Bu eşitlikte bulunan γ açısı Şekil3.4'teki hedefin manyetik kuzeye olan kerteriz açısıdır.

Eşitlik (3.6), (3.10), (3.11) ve (3.14)'ün toplanmasıyla DIFAR standartındaki çoklanmış birleşik işaret ,

$$e_{comp} = E_S \sin\rho t + E_f \cos(\omega t-\alpha) - E_p \cos 2\omega t + E_S \sin\rho t\sin(2\omega t+\gamma) \tag{3.15}$$

olarak elde edilir. Şekil 3.7'de eşitlik (3.15)'in spektrumdaki görüntüsü verilmiştir.



Şekil 3.7: Eşitlik 3.15'in frekans spektrumu görüntüsü [8].

3.2.2 DIFAR standartındaki çoklanmış birleşik işaretten hedefin manyetik kuzeye olan kerterizinin bulunması

QAM ile modüle edilmiş X,Y yönlü hidrofonlarının demodülasyonu için sonoboyda oluşturulan faz pilotun (eşitlik 3.11) alıcıda tekrar üretilmesi gerekmektedir. Bu işlem bölüm 5.2'de ayrıntılı açıklanacaktır. Faz pilotu elde ettikten sonra QAM demodülasyonunda kullanacağımız biribirine dikgen taşıyıcıları

$$\begin{aligned} e_{c(EW)} &= \cos(2\omega t) \\ e_{c(NS)} &= \sin(2\omega t) \end{aligned} \tag{3.16}$$

olarak ifade edebiliriz.

Demodülasyon için QAM işaretini (eşitlik 3.14) ayrı ayrı eşitlik (3.16)'daki taşıyıcılarla çarparsak,

$$\begin{aligned} e_{(EW)} &= \frac{E_S}{2}\cos(2\omega t)\left[\cos(\rho t - \theta - 2\omega t - \phi) - \cos(\rho t + \theta + 2\omega t + \phi)\right] \\ &= \frac{E_S}{4}\begin{bmatrix}\cos(\rho t - \theta - 2\omega t - \phi + 2\omega t) + \cos(\rho t - \theta - 2\omega t - \phi - 2\omega t) \\ -\cos(\rho t + \theta + 2\omega t + \phi + 2\omega t) - \cos(\rho t + \theta + 2\omega t + \phi - 2\omega t)\end{bmatrix} \\ &= \frac{E_S}{4}\left[\cos(\rho t - \gamma) + \cos(\rho t - \gamma - 4\omega t) - \cos(\rho t + \gamma + 4\omega t) - \cos(\rho t + \gamma)\right] \end{aligned} \tag{3.17}$$

$4\omega t$ = 30kHz'de oluşan bileşenleri alçak geçiren bir filtreden geçirip eledikten sonra,

$$\begin{aligned} e_{(EW)} &= \frac{E_S}{4}\left[\cos(\rho t - \gamma) - \cos(\rho t + \gamma)\right] \\ &= \frac{E_S}{2}\sin\rho t \sin\gamma \\ &= \frac{E_S}{2}\sin\rho t \sin(\theta + \phi) \end{aligned}$$

Aynı şekilde ikinci taşıyıcı için aynı işlemi yaparsak,

$$\begin{aligned} e_{(NS)} &= \frac{E_S}{2}\sin(2\omega t)\left[\cos(\rho t - \theta - 2\omega t - \phi) - \cos(\rho t + \theta + 2\omega t + \phi)\right] \\ &= \frac{E_S}{4}\begin{bmatrix}\sin(2\omega t + \rho t - \theta - 2\omega t - \phi) + \sin(2\omega t - \rho t + \theta + 2\omega t + \phi) \\ -\sin(2\omega t + \rho t + \theta + 2\omega t + \phi) - \sin(2\omega t - \rho t - \theta - 2\omega t - \phi)\end{bmatrix} \end{aligned} \tag{3.18}$$

$$= \frac{E_S}{4}\left[\sin(\rho t - \gamma) + \sin(4\omega t - \rho t + \gamma) - \sin(4\omega t + \rho t + \gamma) + \sin(\rho t + \gamma)\right]$$

$4\omega t$ = 30kHz'de oluşan bileşenleri alçak geçiren bir filtreden geçirip eledikten sonra,

$$e_{(NS)} = \frac{E_S}{4}\left[\sin(\rho t - \gamma) - \sin(\rho t + \gamma)\right] \tag{3.18}$$

$$= \frac{E_S}{2}\sin\rho t\cos\gamma$$

$$= \frac{E_S}{2}\sin\rho t\cos(\theta + \phi)$$

eşitliklerini elde ederiz. Hedefin manyetik kuzeye olan kerterizini(γ) ,

$$\gamma = \arctan\left(\frac{e_{(EW)}}{e_{(NS)}}\right)$$

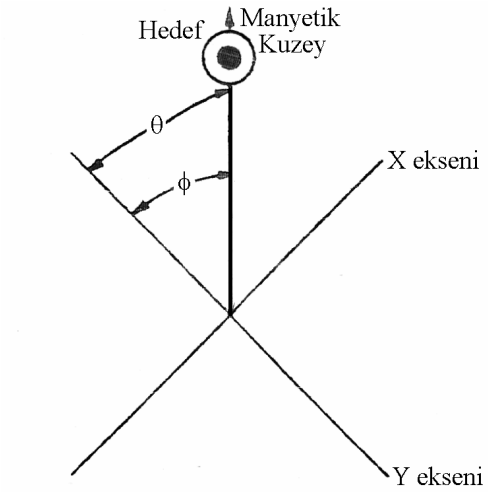
$$= \arctan\left(\frac{\frac{E_S}{2}\sin\rho t\sin(\theta + \phi)}{\frac{E_S}{2}\sin\rho t\cos(\theta + \phi)}\right) \tag{3.19}$$

$$= \arctan(\tan(\theta + \phi))$$

$$= \theta + \phi = \gamma$$

işlemi ile buluruz.

Örnek olarak Şekil 3.8'deki gibi hedef ve manyetik kuzeyin aynı eksende çakıştığını düşünelim.



Şekil 3.8: Manyetik kuzey ile hedefin aynı eksende olduğu senaryo

θ açısı saat yönünde alındığı için +45° kabul edersek ve φ açısı saat yönünün tersine alındığı için -45° kabul edersek,eşitlik 3.19'a göre,

$$\begin{aligned}\gamma &= \theta + \phi \\ &= (+45° - 45°) \\ &= 0°\end{aligned}$$

olarak bulunur, yani manyetik kuzey ile hedef aynı eksende çakıştığından γ yani hedefin manyetik kuzeye olan kerterizi 0° dir.

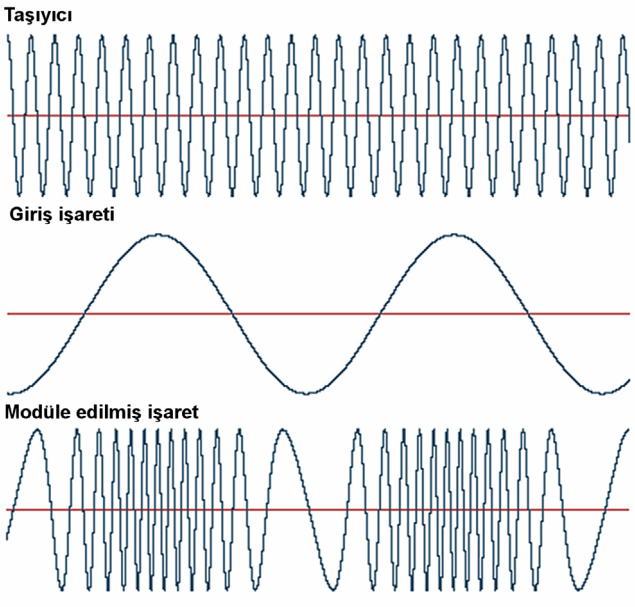
3.2.3. DIFAR standartındaki çoklanmış birleşik işaretin FM modülasyonu

Eğer 18 kHz civarı bant genişliğindeki temelbant çoklanmış işareti anten ile kablosuz olarak iletmek istersek 4167 m boyunda λ/4 bir antene ihtiyaç duyarız, bu ise pratikte imkansızdır. Ayrıca aynı ortamda çalışan sonoboyların birbirlerini etkilememesi gerekmektedir, yani elektromanyetik spektrumun ayrı bölgelerinde yayınım yapmaları gerekmektedir. Bu iki problemin üstesinden gelmek için temelbant işaretin daha yüksek bir taşıyıcı işaret ile modüle edilmesi gerekmektedir. Bu amaçla genişbant FM modülasyonu seçilmiştir. FM modülasyonunun seçilmesindeki sebepler aşağıda verilmiştir,

- FM vericisini düşük maliyetle gerçeklemek kolaydır.
- FM modülasyonunda tüm verici gücü yararlı olarak kullanılır, AM modülasyonunda ise verici gücünün %50'si bilgi taşımayan taşıyıcıda ziyan olmaktadır.
- FM işaretleri atmosferik gürültüden AM'e oranla daha az etkilenir ve yüksek sadakatlıdır.

Şekil 3.9'da görüldüğü üzere frekans modülasyonu, taşıyıcı işaretin frekansını modüle edici temelbant işaret ile değiştiren bir modülasyon sistemidir[11]. Eğer temelbant işareti sinüzoidal bir işaret olarak varsayarsak, frekans modülasyonu için bu işaretin genliği ve frekansı kritiktir. İletilmek istenen bilgi sinyali (mesela ses) değişken frekanslı ve genellikle çok harmonikli bir sinyaldir. Ancak bu sinyalin bir kosinüs dalgasıyla ifade edilebileceği varsayılırsa,

$$m(t) = a.\cos(2\pi f_m t) \tag{3.20}$$



Şekil 3.9: FM modülasyonu sembolik gösterimi

olarak yazılabilir.

Taşıyıcı işaret ise,

$$u(t) = A_c.\cos(2\pi f_c t) \tag{3.21}$$

olarak verilebilir.

Burada a ve A_c sırasıyla bilgi ve taşıyıcı genlikleri, f_m ve f_c ise modüle edici işaret ve taşıyıcı frekanslarıdır. Modüle olmamış taşıyıcının frekansına(f_c) merkez frekans denilir. Modülasyon sonrası taşıyıcı işaretin frekans sapması,

$$\phi(t) = 2\pi k_f \int_{-\infty}^{t} m(\tau)d\tau = \frac{k_f a}{f_m}\sin(2\pi f_m t) \tag{3.22}$$

olarak bulunur. Burada taşıyıcı frekans sapması($\phi(t)$), modülasyon sabiti(k_f) parametresine, bilgi işareti genliğine(a) ve frekansına(f_m) bağlıdır.

Buna göre modüle edilmiş FM işareti,

$$u(t) = A_c\cos(2\pi f_c t + \phi(t)) = A_c\cos\left(2\pi f_c t + \frac{k_f a}{f_m}\sin(2\pi f_m t)\right) \tag{3.23}$$

biçiminde yazılabilir.

Eşitlik (3.23)'deki $\frac{k_f a}{f_m}$ ifadesini,

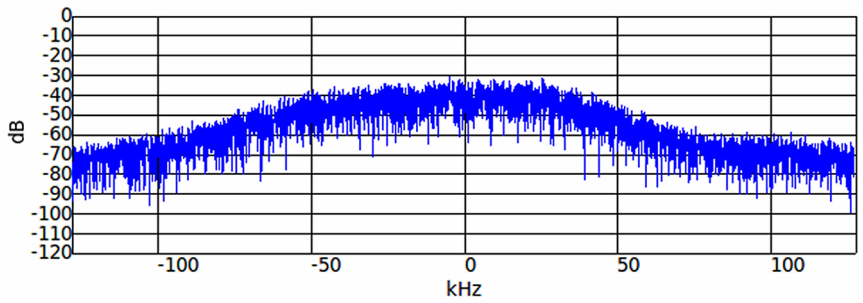
$$\beta = \frac{k_f a}{f_m} \tag{3.24}$$

şeklinde yazabiliriz. Bu eşitlikteki β, modülasyon indeksi olarak tanımlanır. Eşitlik 3.24'ü $m(t)$ için genelleştirirsek,

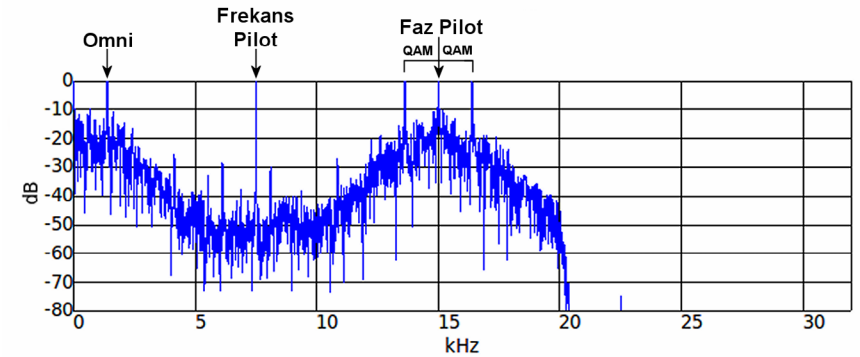
$$\beta = \frac{k_f \max[|m(t)|]}{W} = \frac{\Delta f_{max}}{W} \tag{3.25}$$

şekline dönüşür. Burada W, $m(t)$ bilgi işaretinin bant genişliği, Δf_{max} ise maksimum frekans sapmasıdır.

Şekil 3.9 ve 3.10'da iletim durumundaki FM spektrumu ve demodüle edilmiş sonoboy temelbant spektrumu verilmiştir.



Şekil 3.9: İletim durumunda sonoboy FM spektrumu



Şekil 3.10: İletim durumunda demodüle edilmiş sonoboy temelbant spektrumu

3.2.4. DIFAR FM modülasyonu iletim parametreleri

DIFAR standartı FM modülasyonu parametreleri aşağıdaki gibi verilmiştir.

- RF çıkış gücü: 1 watt RMS
- RF frekans bantı: 0.375 MHz aralıklarla 136-173,5 MHz bantında 99 kanal. (Tablo 3.1'de verilmiştir.)
- Modülasyon: Genişbant FM
- Maksimum salınım: ± 105 kHz
- Temelbant bant genişliği: 20 kHz
- Anten yönelticiliği: Yönsüz (omnidirectional)
- β (modülasyon indeksi): ~11

Tablo 3.1: Sonoboy çalışma frekansları ve kanal numaraları

Kanal No.	Frekans (MHz)	Kanal No.	Frekans (MHz)	Kanal No.	Frekans (MHz)
01	162.250	34	136.750	67	149.125
02	163.000	35	137.125	68	149.500
03	163.750	36	137.500	69	149.875
04	164.500	37	137.875	70	150.250
05	165.250	38	138.250	71	150.625
06	166.000	39	138.625	72	151.000
07	166.750	40	139.000	73	151.375
08	167.500	41	139.375	74	151.750
09	168.250	42	139.750	75	152.125
10	169.000	43	140.125	76	152.500
11	169.750	44	140.500	77	152.875
12	170.500	45	141.875	78	153.250
13	171.250	46	141.250	79	153.625
14	172.000	47	141.625	80	154.000
15	172.750	48	142.000	81	154.375
16	173.500	49	142.375	82	154.750
17	162.625	50	142.750	83	155.125
18	163.375	51	143.125	84	155.500
19	164.125	52	143.500	85	155.875
20	164.875	53	143.875	86	156.250
21	165.625	54	144.250	87	156.625
22	166.375	55	144.625	88	157.000
23	167.125	56	145.000	89	157.375
24	167.875	57	145.375	90	157.750
25	168.625	58	145.750	91	158.125
26	169.375	59	146.125	92	158.500
27	170.125	60	146.500	93	158.875
28	170.875	61	146.875	94	159.250
29	171.625	62	147.250	95	159.625
30	172.375	63	148.625	96	160.000
31	173.125	64	148.000	97	160.375

Tablo 3.1:(Devam) Sonoboy çalışma frekansları ve kanal numaraları

Kanal No.	Frekans (MHz)	Kanal No.	Frekans (MHz)	Kanal No.	Frekans (MHz)
32	136.000	65	148.375	98	160.750
33	136.375	66	148.750	99	161.125

4. YAZILIM GELİŞTİRME ORTAMI

Proje geliştirme sürecinde açık kaynak kodlu yazılım ve donanımlardan istifade edilmiştir. Açık kaynaklı yazılım, kaynak kodu isteyen herkese açık olan yazılımlardır. Bu tür yazılımların ayırt edici özelliği kullanıcıya yazılımı değiştirme özgürlüğü sağlamasıdır. Açık kaynak kodlu yazılımlar, uyarlanabilir, sağlam, hızlı ve güvenlidir. Açık kaynak kod dünyası, yeni bir yazılım üretme biçimi, yeni iş modelleri sunmaktadır. Dünyanın her tarafından bilişim uzmanlarınca imece yöntemi ile endüstri standartlarında geliştirilen açık kaynak kod yazılımları, insanlığın ortak malıdır. Projede kullanılan açık kaynak kodlu yazılımlar bu bölümde özet olarak anlatılacaktır.

4.1. Python Programlama Dili

Python [13], genel kullanımlı yüksek seviyeli bir programlama dilidir. python'un geliştirilmesi 1980'lerde Hollanda'lı programcı Guido Van Rossum tarafından başlatılmıştır. 16 Ekim 2000 tarihinde 2.0 sürümüne "garbage collector" ve unicode desteği ile ulaşmıştır. 3 Aralık 2008'de 3.0 sürümü yayınlanmış fakat 2.x sürümleri ile uyumluluğu tam olmadığından dolayı henüz yaygın değildir. Bu yüksek lisans tez projesinde 2.5.2 sürümü temel alınmıştır.

Python'un geliştirilmesindeki tasarım hedefi kod okunabilirliğidir. Kodlama imlası ve semantik kurallar esnek ve minimaldir. Python kendisi yalın ve yüksek seviyeli bir dil olmasına rağmen geniş kütüphane desteği ile çok amaçlı kullanılabilir. Yapısal programlamanın yanında NYP (Nesne Yönelimli Programlama) ve dinamik bellek yönetimi gibi gelişmiş özellikler sunar. NYP [15], özetle bir bilgisayar programlama yaklaşımıdır. Günümüzde pekçok çağdaş programlama dili tarafından desteklenmektedir.

1960'lı yılların sonuna doğru ortaya çıkan bu yaklaşım, o dönemin yazılım

dünyasında beliren bir bunalımın sonucudur. Yazılımların karmaşıklığı ve boyutları sürekli artıyor, ancak belli bir nitelik düzeyi korumak için gereken bakımın maliyeti (zaman ve çaba olarak) daha da hızlı artıyordu. NYP'yi bu soruna karşı bir çözüm haline getiren başlıca özelliği, yazılımda birimselliği (İngilizce - modularity) benimsemesidir.

NYP'nin altında yatan birimselliğin ana fikri, her bilgisayar programının, etkileşim içerisinde olan birimler veya nesneler kümesinden oluştuğu varsayımıdır. Bu nesnelerin her biri, kendi içerisinde veri işleyebilir, ve diğer nesneler ile çift yönlü veri alışverişinde bulunabilir. Halbuki NYP'dan önce var olan tek yaklaşımda (Yapısal programlama), programlar sadece bir komut dizisi veya birer fonksiyon kümesi olarak görülmekteydiler. Bilimsel çevreler tarafından NYP'nın geçmişe göre daha yüksek esneklik ve bakım kolaylığı sunduğu iddia edilmektedir. Bu sebepten dolayı, günümüzün geniş çaplı yazılım projelerinde yaygınca kullanılmaktadır.

4.1.1. Python yorumlayıcısı

Python bir yorumlayıcı dilidir, yani yazılan program kodları derlenmez, her zaman satır satır işletilir. Python bu yapısıyla açık kaynak kod mantığına uygundur, python'da geliştirilen kodlar daima derlenmemiş olduğundan açılıp incelenebilir, derlenmiş kodlarda ise bu mümkün olmayabilir. Bunun bir dezavantajı ise kodların derlenmemesinden dolayı mimariye özel optimizasyonların yapılamamasıdır, bunun sonucunda C++ gibi orta düzeyli dillere göre bir miktar performans kaybı meydana gelmektedir, günümüzün yüksek performanslı CPU (Merkezi İşlem Birimi)'larında bu kayıp ihmal edilebilir seviyededir.

Python'un yüksek seviyeli yapısı sayesinde geliştirme aşamasında önemli bir zaman kazanılabilir, bunun yanında yorumlayıcı interaktif olarakta çalıştırılabilir. Bu özellik hazır modüllerin test edilmesinde ve dilin öğrenilmesi aşamasında kullanışlıdır. Şekil 4.1'de örnek bir interaktif python uygulaması verilmiştir.

Python'un önemli bir özelliği program kodlarının modüllere ayrılarak yeniden kullanılmasını sağlamasıdır. Python yalın haliyle standart modüllerle gelmektedir,

yani yazılım geliştirilmeye hemen başlanabilir. Bunların yanında I/O, GUI ve TCP/IP soketi gibi popüler modüller kolaylıkla eklenebilir.

```
engink@engink-laptop:~$ python
Python 2.5.2 (r252:60911, Oct  5 2008, 19:24:49)
>>> print "2 + 2 is", 2 + 2
2 + 2 is 4
>>> print "3 * 4 is", 3 * 4
3 * 4 is 12
>>> print "100 - 1 is", 100 - 1
100 - 1 is 99
>>> print "(33 + 2) / 5 + 11.5 is", (33 + 2) / 5 + 11.5
(33 + 2) / 5 + 11.5 is 18.5
```

Şekil 4.1: Örnek bir etkileşimli python uygulaması

4.1.2 Python veri tipleri

Python'da da diğer dillerdeki temel veri tipleriyle birlikte kendine özgü bazı veri tipleri bulunur. Python'un en büyük avantajlarından biri öntanımlı veri tiplerinin kolayca programcı tarafından tanımlanacak sınıflar tarafından taklit edilebilmesidir. __getattr__ ve __str__ gibi özel metodların yaratılan sınıf tarafından taklit edilmesi ile tamamen isteğe göre hazırlanmış özel veri tipleri üretilip diğer tüm veri türleri gibi kullanılabilir.

4.1.2.1 Boolean değerler

Python'un geniş veri tipleri arasında en yeni olanı boolean değerlerdir. Python 2.2'den itibaren 0 ve 1 sayılarına denk olan True, False tanımlamaları getirilmiştir. Python aynı zamanda C'ye benzer bir şekilde her türlü veri tipini doğruluk testinde kullanabilir. Örneğin boş bir string, boş bir liste veya 0 sayısı yanlış, 0 dışındaki herhangi bir sayı, boyutu 0'dan büyük olan herhangi bir liste veya string ise doğru kabul edilecektir.

4.1.2.2 Sayısal veri tipleri

Python'da dört temel sayısal veri tipi bulunur. Bunları integer (tamsayı), long (uzun tamsayı), float (kayar noktalı sayı), complex (karmaşık sayı) olarak

sınıflandırabiliriz. Bu veri tiplerinin tamamı işaretli sayılardır. Zorunlu durumlarda Python yorumlayıcısı integer, long ve float tipleri arasında otomatik değişimler yapma özelliğine sahiptir. Örnek olarak Python yorumlayıcısına integer veri tipinin sınırlarını aşan bir değer verdiğimiz takdirde yorumlayıcı bunu otomatik olarak long tipine çevirecektir. Benzer şekilde bir işlemin operandlarından birinin float olması halinde sonuçta float şeklinde dönecektir.

4.1.2.3 String tipleri

Python'a unicode desteği eklendiğinden beri string ve unicode değişkenler ayrı veri tipleri olarak tanımlanmıştır. Unicode değişkenleri temel olarak string veri tipleri ile aynı özellikleri paylaşsa da, kendine özgü bazı ek fonksiyonlara sahiptir. Unicode nesnelerini kullanırken unutulmaması gereken bir detay eğer nesne ASCII karakterlerinin dışında veri içerecekse unicode nesnesinin encode() fonksiyonunu kullanarak uygun karakter setine dönüştürülmesini sağlamaktır.

4.1.2.4 Dizi Tipleri

Python diğer dillerdeki gibi tek bir dizi tipi yerine kendi içerisinde Tuple ve List olmak üzere iki farklı veri tipini barındırır. Tuple tipindeki dizilerin en büyük özelliği içerdiği verilerin değiştirilmesinin mümkün olmamasıdır. Buna karşılık List tipindeki diziler sonradan değiştirilebilirler. Python dizi tiplerinin diğer dillerdeki benzer tiplerden en önemli farkı negatif indeksler ve dizi dilimlemeleridir (slicing). Negatif indeksler sayesinde dizi elamanlarına alışıldık 0'dan <dizi uzunluğu> - 1'e kadar olan sayılarla erişim dışında dizinin sonundan başlayarak negatif sayılarla da erişebiliriz. Dizinin son elemanı -1 olmak üzere başa doğru gidildikçe küçülen negatif sayılarla dizinin diğer elemanlarına erişebiliriz. Buna göre dizi[-1] bize dizinin son elemanını verirken dizi[-2] sondan ikinci elemanı verecektir. Bu özellik Python'un dizi dilimleme yetenekleri ile birleştirildiğinde bize dizilerin belirli bölümlerini ayıklamak konusunda oldukça güçlü bir araç sunar.

4.1.3. Kodlama imlası

4.1.3.1 Kod blokları

Python'un diğer dillerle farkının en çok ortaya çıktığı nokta blok yapılarıdır. Diğer dillerde {} karakterleri veya begin, end gibi işaretlerle ile sağlanan blok yapıları Python'da girinti-çıkıntılar (tab) yardımıyla sağlanır. Kodun içerisinde aynı seviyede girintiye sahip olan alt alta bölümler aynı blok içerisinde kabul edilir. Python kodu yazarken girinti/çıkıntıları doğru ayarlanmadığı taktirde Python yorumlayıcısı hata verecektir.

4.1.3.2 Kontrol yapıları

Bir dilin kontrol yapıları en önemli parçasıdır. If-Else gibi denetleme yapıları ve for, while gibi döngüler her dilde bulunsa da dilden dile bazı farklılıklar içerirler. Python her programlama dili gibi bu yapıları içerir. Kontrol yapıların genel yapısı şu şekildedir:

<kontrol deyimi> <kontrol cümlesi>:
    <çalıştırılacak komut>

Python'daki denetleme deyimleri ve döngüler diğer dillerdekinden daha farklı olmadığından dolayı bu bölümde ayrıntıya girilmeyecektir. Fakat diğer dillerdeki benzerlerinden bazı farklar gösteren for ve while deyimlerine değinmekte fayda vardır.

C ve Pascal gibi dillerde for deyimi sayı sayma esasına dayanır. Sayı tipinde bir değişken artırılıp azaltılarak kontrol sistemi kurulur. Python'da ise for döngüsü bir liste üzerinde işletilerek listenin her elemanı listedeki sıralarına göre işleme alınır. For döngüsü her ne kadar bir liste üzerinde çalışsa da kimi zaman doğrudan sayılar üzerinde çalışılması gerekebilir. Python böyle durumlar için range() isimli fonksiyonunu sunmaktadır. Bu fonksiyon kendisine verdiğimiz parametrelere göre bize bir sayı listesi üretir. range() fonksiyonu üç farklı parametre alır,bunları taban sayı, tavan sayı ve artırma miktarı olarak sıralayabiliriz. Bu duruma aşağıdaki gibi örnek verebiliriz.

```
>>> range(10)  #0'dan 10'a kadar (10 hariç) olan sayıları döndür
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(1, 11, 2)  #11'den küçük pozitif tek tamsayıları döndür
[1, 3, 5, 7, 9]
```

Python'da diğer dillerden farklılık gösteren diğer bir deyim ise if deyimidir. Python'da case veya switch gibi ayrı bir kalıp bulunmadığından if deyimi bu görevi üstlenecek şekilde düzenlemiştir. Bir if deyiminin içerisinde kullanacağınız birden fazla elif (Else-if'in kısaltması) deyimi ile farklı koşulları test etmek mümkündür.

4.1.3.3 Fonksiyon ve sınıf tanımlamaları

Python'da fonksiyon tanımlamaları def anahtar kelimesi ile yapılır. genel yapı ,
def <fonksiyon_adi>(<parametreler>):
biçimindedir.

Fonksiyon, bu tanımın altındaki satırlara uygun girinti (tab) ile yazılır. Fonksiyon parametrelerini tanımlarken duruma göre bazı yaklaşımlar kullanmamız mümkündür. Örnek olarak, tanımladığımız fonksiyon parametrelerinin bazılarına öntanımlı değerler atamamız mümkündür. Bu sayede fonksiyonu çağırırken yalnızca öntanımlı değeri değiştirmemiz gereken durumlarda o parametreye müdahale etmemiz gerekecektir. Parametrelere öntanımlı değer atanması şu şekilde yapılır:

```
def fonksiyon_adı(gun ="salı"):
...
```

Bu durumda Python yorumlayıcısı fonksiyonu sadece fonksiyon_adı() şeklinde çağırdığımızda “gun” değişkenine otomatik olarak “salı” değerini atayacaktır. Kural olarak, öntanımlı değer atanan parametreler, atanmayan parametrelerden sonra gelir.

Sınıf tanımlamaları class anahtar sözcüğü ile yapılır. Python’da sınıf tanımlaması şu şekilde yapılır,
class (<temel_sinif1>, < temel_sinif2>,...):

Eğer sınıf bir temel sınıf'tan (base class) türetilmiyorsa parantezlerin kullanılmaması gerekir. Bir sınıfın tanımında birden çok temel sınıf kullanmak mümkündür, bu sayede Python'da nesne yönelimli programlamanın özelliklerinden çoklu miras almayı (multiple inheritance) kullanmamız mümkün hale gelir.

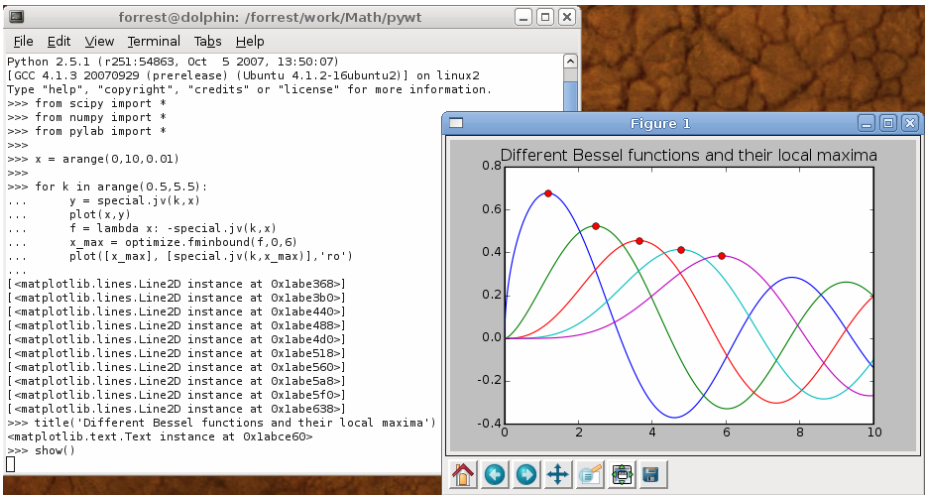
Bir sınıf içerisindeki fonksiyonları tanımlarken ilk parametre fonksiyonun çağrıldığı sınıf örneğine (instance) referans veren self parametresi olmalıdır. Parametresi olmayan fonksiyonları tanımlarken dahi bu parametrenin parametre listesinin ilk elemanı olarak eklenmesi gerekmektedir. Bu parametrenin adının self olması zorunlu olmamakla birlikte uygulamada bu tanımlama standart olmuştur. Hello adlı bir fonksiyona sahip bir sınıf aşağıdaki gibi gözükecektir:

```
class HelloWorld:
  def Hello(self):
    print "Hello Python!"
```

Python yorumlayıcısı bu fonksiyonu çağırırken self parametresini otomatik olarak yerleştirecektir.

4.2. Scipy Python Bilimsel Kütüphanesi

SciPy, python için numpy kütüphanesi üzerine kurulmuş matematiksel algoritmalar ve fonksiyonlar topluluğudur [16].



Şekil 4.2: Scipy ile bessel fonksiyonların hesaplanması.

Scipy, python'a sayısal verilerin yüksek seviyeli komutlarla işlenmesi ve gösterilmesi konusunda önemli bir güç sağlar. Scipy ile interaktif python ortamı Matlab, Octane, Scilab gibi bir veri işleme ortamına dönüşür. Şekil 4.2'de örnek bir scipy uygulaması verilmiştir.

Python'a scipy'nin entegre edilmesiyle gelişmiş programlar ve özelleşmiş uygulamalar yaratılması için güçlü bir programlama altyapısı kazanılmış olur. Scipy ile yazılan bilimsel uygulamalar dünya çapına yayılmış sayısız geliştiricinin geliştirdiği bilimsel modüllerden istifade ederler.

Matematiksel modüllere ek olarak paralel programlama ve veri tabanı alt rutinleri hazır olarak gelmektedir. Python yorumlayıcısının Linux dağıtımlarında varsayılan olarak kurulu olduğunu kabul edersek. Debian uyumlu dağıtımlarda scipy paketini,

sudo apt-get install python-scipy

komutu ile kurabiliriz. Bu işlemden sonra scipy kütüphanesini,

```
import scipy as sp
```

şeklinde kendi projemize ekleyebiliriz.

4.2.1. Scipy organizasyonu

Scipy farklı bilimsel hesaplama alanlarında alt paketlere ayrılmıştır. Bazı yaygın fonksiyonlar scipy ile varsayılan olarak kurulan scipy-base paketi ile gelmektedir. Scipy alt paketleri istenirse birbirinden bağımsız olarakta projelere eklenebilir, bu sayede kullanılmayan paketler bellekte yer işgal etmez. Tablo 4.1'de scipy alt paketleri ve açıklamaları verilmektedir.

Tablo 4.1: Scipy organizasyon yapısı

Alt paket	**Açıklaması**
cluster	Clustering algoritmaları
constants	Fiziksel ve matematiksel sabitler
fftpack	Hızlı fourier dönüşümü rutinleri
integrate	İntegral ve diferansiyel denklem çözümleme rutinleri

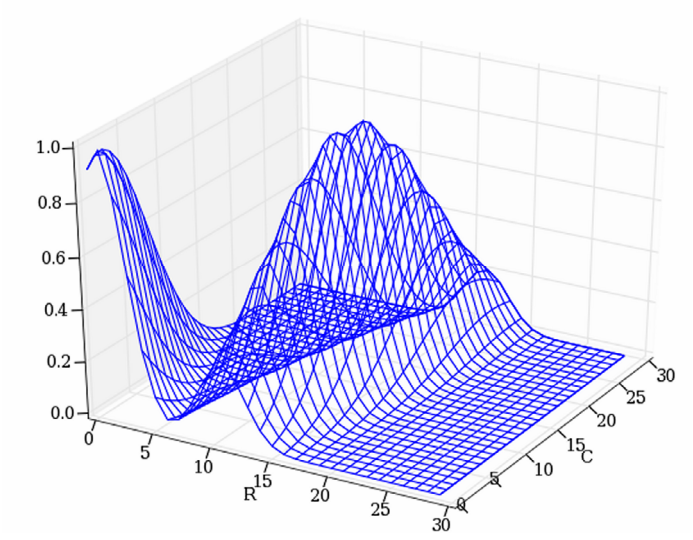
Tablo 4.1:(Devam) Scipy organizasyon yapısı

interpolate	İnterpolasyon ve yumuşatma rutinleri
i/o	Giriş/Çıkış rutinleri
linalg	Lineer cebir
maxentropy	Maksimum entropi metotları
ndimage	N-boyutlu görüntü işleme rutinleri
odr	Dikgen uzaklık regresyonu
optimize	Optimizasyon ve kök bulma rutinleri
signal	İşaret işleme rutinleri
sparse	Sparse matris rutinleri
spatial	Uzamsal veri yapıları ve algoritmaları
special	Özel fonksiyonlar
stats	İstatistiksel dağılımlar ve fonksiyonları
weave	C/C++ entegrasyonu

Bu yüksek lisans tezinde scipy kütüphanesinin constants, fftpack, I/O, signal altpaketleri kullanılmıştır.

4.3. Matplotlib Python Grafik Kütüphanesi

Matplotlib, python'da tanımlanan dizilerden iki ve üç boyutlu grafikler çizdirilmesi amacıyla kullanılan bir python kütüphanesidir [17]. Matplotlib kütüphanesinin geliştirilme amacı birkaç komut ile istenilen grafiklerin basitçe çizdirilmek istenmesidir.



Şekil 4.3: Matplotlib kütüphanesi ile üç boyutlu grafik uygulaması

Python yorumlayıcısının Linux dağıtımlarında varsayılan olarak kurulu olduğunu kabul edersek. Debian uyumlu dağıtımlarda matplotlib paketini,

sudo apt-get install python-matplotlib

komutu ile kurabiliriz. Bu işlemden sonra matplotlib kütüphanesini,

import matplotlib

şeklinde kendi projemize ekleyebiliriz. Şekil 4.3'te bir matplotlib grafik uygulaması örneği verilmiştir.

Bu yüksek lisans tezinde matplotlib kütüphanesinin mlab ve pylab alt paketleri kullanılmıştır

4.3.1. Örnek bir matplotlib uygulaması

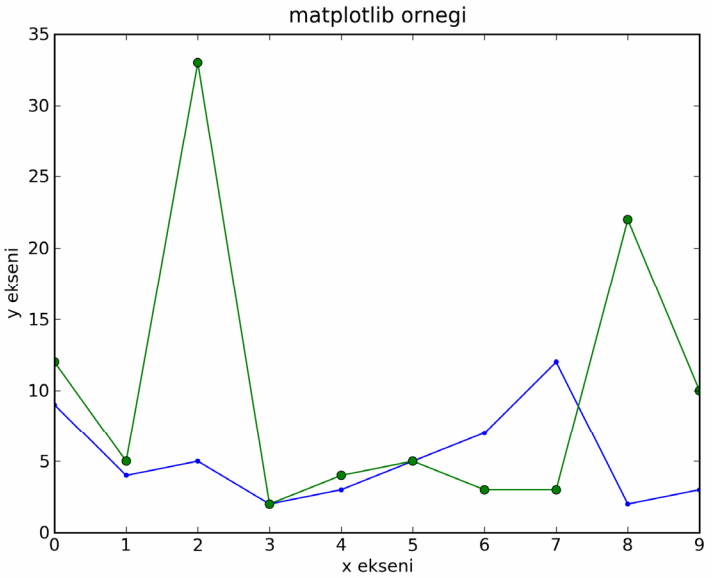
Örnek olarak iki boyutlu "ornek1" ve "ornek2" dizilerinin grafiğe aktarılmasını inceleyelim. Buna göre program kodu,

```
from pylab import *

s1 = [9,4,5,2,3,5,7,12,2,3]
s2 = [12,5,33,2,4,5,3,3,22,10]
t = arange(0,10)
plot ( t,s1,'.-',label='ornek1' )
plot ( t,s2,'o-',label='ornek2' )
xlabel('x ekseni')
ylabel('y ekseni')
title('matplotlib ornegi')
savefig("ornek2.png",dpi=(300))
```

şeklinde olacaktır. s1 ve s2'yi 1 Hz örneklenmiş sayısal işaret olarak düşünelim, t ise 0'dan 9'a kadar örnekleme aralıkları olsun. "plot" fonksiyonu pylab altpaketine dahil olan iki ve üç boyutlu grafikleri oluşturan fonksiyondur.

Plot fonksiyonuna x ekseni "t" ve y ekseni "s1,s2" örneklerini parametre olarak verirsek,



Şekil 4.4: Matplotlib kütüphanesi ile iki boyutlu uygulama örneği

Şekil 4.4 grafiğini elde ederiz.

4.4. GNU Radio Yazılım Tanımlı Radyo Geliştirme Ortamı

GNU Radio yazılım tanımlı radyoları gerçeklemek amacıyla geliştirilmiş ücretsiz bir yazılım kütüphanesidir. GNU Radio GNU GPL lisansı altında dağıtılmaktadır, buna göre kullanıcılar GNU Radio kodlarını inceleyebilir ve istedikleri gibi değiştirebilirler.

GNU Radio'nun hedefi mümkün olan tüm donanımı yazılım biçimine dönüştürmektir. GNU Radio bir çok işaret ve veri işleme bloğu sunar, bu bloklar arasındaki veri akışı GNU Radio çerçevesi ile sağlanır. Bunlardan istenilenler birbirine bağlanarak, kullanıcılar yazılım tanımlı radyolarını oluşturabilirler. GNU Radio bloklarının bir kısmını aşağıdaki listede tanımlayabiliriz,

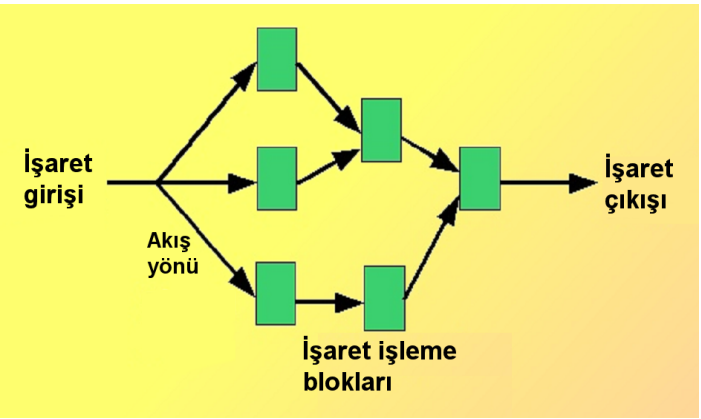
- Matematik işlemleri (toplama,çarpma,log...)
- Bit serpiştirme, geciktirme blokları
- Filtreler (FIR,IIR, Hilbert,...)
- FFT blokları
- Otomatik kazanç kontrolü (AGC)
- Modülasyon ve demodülasyon (FM, AM, PSK, QAM, GMSK, OFDM…)

- Ara değer kestirimi ve veri azaltma blokları
- Trellis ve Viterbi desteği
- İşaret Jeneratörleri
- Gürültü jeneratörleri (white, pink, brown…)
- Rastgele numara jeneratörleri
- USRP kaynak ve çıkış blokları
- Grafik çıktıları (Osiloskop, spektrum analizör,…)
- Ses kaynak ve çıkış blokları
- Dosya kaynak ve çıkış blokları (Dosyadan okuma ve dosyaya kaydetme)
- UDP kaynak ve çıkış blokları (örneklenmiş işaretleri ağdan transfer etmek için)

Bölüm 2.2.2'de ayrıntılı bir biçimde değinildiği üzere GNU Radio projesi Eric Blossom tarafından başlatılmış ve günümüzde dünya çapındaki gönüllü geliştiriciler tarafından desteklenmektedir. GNU Radio'nun gerçek çalışma ortamı Linux olmasına rağmen az bir çaba ile windows ortamında çalışabilmektedir.

4.4.1. GNU Radio akış diyagramları

Her GNU Radio uygulaması akış diyagramlarından oluşturulmaktadır. Bu akış diyagramının işaret girişi, çıkışı ve işaret işleme blokları vardır.



Şekil 4.5: GNU Radio akış diyagramı yapısı

Şekil 4.5'te GNU Radio akış diyagramı yapısı verilmiştir. İşaret işleme bloklarının bir veya birden fazla girişi veya çıkışı olabilir. Girişi olmayan bloklar "işaret

kaynağı" bloğu, çıkışı olmayan bloklar "işaret çıkışı" bloğu olarak isimlendirilir.

Blokların bir kısıtlayıcı özelliği döngü oluşturamamalarıdır, yani bir bloğun çıkışı tekrar kendi girişine bağlanamaz. Bu döngü içeren işaret işleme rutinlerinin mümkün olamayacağı anlamına gelmez, fakat bu gibi döngüler bir bloğun içinde gerçeklenmelidir.

4.4.2. GNU Radio veri formatları

GNU Radio bloklarında kullanılan en yaygın veri biçimleri float ve complex formatındadır. Complex biçimler reel ve imajiner olarak iki kısımdan oluşurlar. Buna göre her örnek 8 byte büyüklüğündedir. Diğer veri formatlarını,

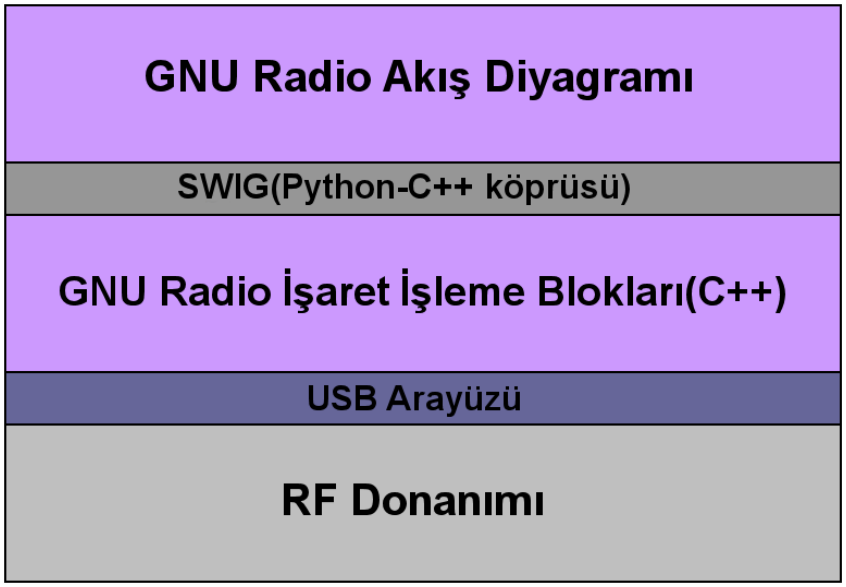
- f- float (4 byte)
- s-short integer (2 byte)
- b-byte integer (1 byte)

olarak sıralayabiliriz.
Bu değişkenler vektör biçimindede ifade edilebilir,bu özellik FFT bloğu gibi belli sayıda örnek gerektiren uygulamalarda yararlıdır.

İşaret işleme bloklarının giriş-çıkış tipleri şu notasyonla belirlenir. Örnek olarak,
ornek_vXY
isimli bloğu ele alırsak, v bu bloğun vektörel olduğunu, X hangi türde girişi olduğunu, Y ise hangi türde çıkışı olduğunu gösterir. Buna göre "fft_vcc" bloğunun vektörel, complex giriş ve çıkışı olduğunu ifade edebiliriz.

4.4.3. GNU Radio yazılım geliştirme metotları

GNU Radio C++ ve python yazılım dilleri ile geliştirilmiştir. Bu iki programlama dili arasındaki bağlantı yine açık kaynaklı bir proje olan SWIG ile sağlanmıştır. C++ düşük seviyeli işaret işleme bloklarında ve donanım desteği için kullanılırken, python ise işlem bloklarını birbirine bağlamak amacıyla kullanılmaktadır.



Şekil 4.6: GNU Radio yazılım katmanları [18].

Şekil 4.6'da GNU Radio yazılım ve donanım katmanları verilmiştir. Bu yaklaşım C++'ın yüksek hızı ile python'un kullanım kolaylığını birleştirmektedir. Bir uygulama geliştirmek için "gr" modülünün top_block sınıfından kendi sınıfını türetebilir ve ya stdgui2 modülünün std_top_block sınıfı ile GUI arayüzlü uygulama geliştirilebilir. Tablo 4.2'de GNU Radio modülleri ve açıklamaları verilmiştir.

Tablo 4.2 GNU Radio modülleri ve açıklamaları

Modül	**Açıklaması**
gr	Ana GNU Radio modülü
usrp	USRP kaynak, çıkış sürücüleri ve kontrol yazılımları
audio	Ses kartı kaynak, çıkış sürücüleri.(Harici bir RF tuner ile darbantlı bir alıcı olarak kullanılabilir.)
blks2	Modülatör/Demodülatör modülü
optfir	Optimal FIR filtre dizayn rutinleri
plot_data	GNU Radio matplotlib destek modülü.
wxgui	GNU Radio wxwidgets destek rutinleri, bu modül ile kolayca grafik arayüzler hazırlanabilir.
eng_notation	Mühendislik notasyonu desteği($10^6 = 100$M gibi).
eng_options	Python komut satırı parametrelerine mühendislik notasyonu desteği sağlar.
gru	Matematiksel ve çeşitli araçlar.

İşaret işleme blokları senkron veya asenkron olabilir. Senkron bloklar giriş veri oranının örnek hızıyla senkronize çalışırlar. Senkron bloklar gr_sync_block(1:1), gr_sync_interpolator(1:N) ve gr_sync_decimator(N:1) sınıflarından türetilirler.(Parantez içinde verilen oranlar çıkış/giriş örnek hızı oranlarıdır. Asenkron bloklar ise gr_block sınıfından türetilirler. İşaret işleme blokları genelde C++ ile yazılırlar, bazende birkaç blok bir araya getirilerek yeni bir blok oluşturulabilir. Bu yeni blok gr.hier_block2 sınıfından türetilmek zorundadır. Buna örnek olarak,

```
class HierBlock(gr.hier_block2):
    def __init__(self, audio_rate, if_rate):
        gr.hier_block2.__init__(self, "HierBlock",
        gr.io_signature(1, 1, gr.sizeof_float),
        gr.io_signature(1, 2, gr.sizeof_gr_complex))
            B1 = gr.block1(...)
            B2 = gr.block2(...)
            self.connect(self, B1, B2, self)
```

örneğini verebiliriz. Bu örnekte gr.hier_block2 sınıfından türetilen HierBlock sınıfına B1 ve B2 olarak iki harici blok eklenmiş ve connect metodu ile bunlar birbirine bağlanmıştır, artık "HierBlock" sınıfı bir modül gibi kullanılabilir.

İstenilen bloklar tanımlandıktan sonra top_block ve ya std_top_block sınıflarından türetilmiş ana sınıfa ait connect metodu ile birbirine bağlanırlar. Daha sonra start metodu ile işaret işleme başlatılır, start metodu ile başlatılan işlem yöneticisi "general_work" fonksiyonunu çağırmaktadır (senkron bloklar için "work"). Gerçek işaret işleme bu fonksiyonlar içinde icra edilmektedir. "general_work" fonksiyonu işlem bittikten sonra geriye kaç giriş çıkış örneğinin işlendiğini ve kaç çıkış örneğinin üretildiği bilgisini döndürür.

GNU Radio aynı zamanda işaret giriş bloklarından gelen örnekleri tamponlayabilir. Tamponlama özellikle CPU'nun giriş örnek hızına yetişemediği durumlarda yararlıdır, fakat tampon büyüklüğününde bir sınırı olduğundan eğer tampon taşarsa giriş örnekleri kaybedilebilir. Böyle bir problem yetersiz işlem güçlerinde meydana gelebilir, ayrıca PC'nin gerçek zamanlı bir işletim sistemi taşımamasıda ayrı bir dezavantajdır. Gerçek zamanlı işletim sistemlerinde kritik işlem bloklarına öncelik verilebilir.

4.4.4. GNU Radio yazılım örneği

Bu bölümde giriş seviyesinde bir gnuradio uygulaması verilecek ve açıklanacaktır. Buna göre örneğimizi,

```
#!/usr/bin/env python

from gnuradio import gr
from gnuradio import audio
from gnuradio.wxgui import stdgui2, fftsink2, form
import wx

class audio_fft_graph (stdgui2.std_top_block):
   def __init__(self,frame,panel,vbox,argv):
      stdgui2.std_top_block.__init__(self,frame,panel,vbox,argv)

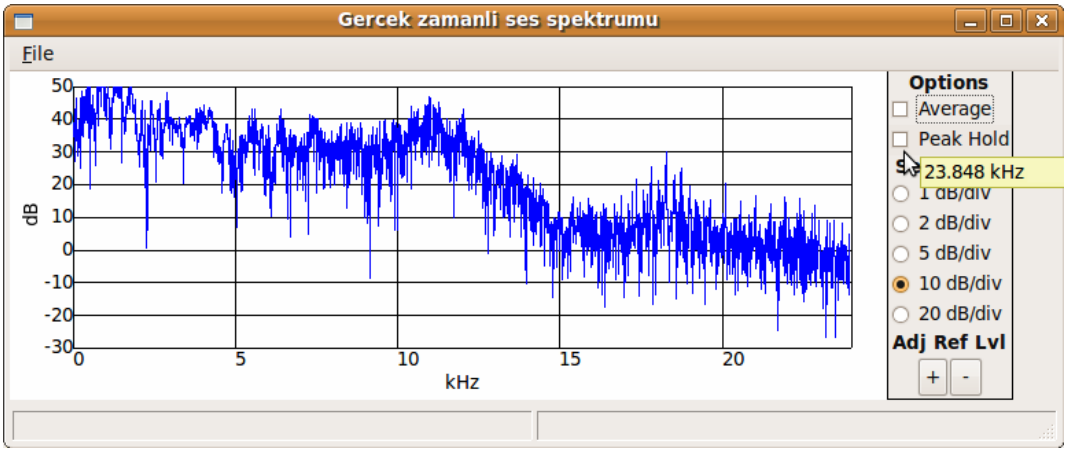
      audio_sample_rate = 48000
      src = audio.source(audio_sample_rate)
      amp = gr.multiply_const_ff(1000)
      sink = fftsink2.fft_sink_f(panel,fft_size =8192,sample_rate=audio_sample_rate)
      self.connect (src,amp,sink)

if __name__ == '__main__':
   app = stdgui2.stdapp (audio_fft_graph,"Gercek zamanli ses spektrumu")
   app.MainLoop ()
```

şeklinde verebiliriz. Örneğimizde ilk olarak Tablo 4.2'de ayrıntılı açıklaması verilen gr, audio ve wxgui modülleri projeye dahil edilmiştir. Daha sonra std_top_block sınıfından türetilmiş audio_fft_graph sınıfı ile gui oluşturulmuştur. audio_fft_graph sınıfının kurucu fonksiyonu __init__ içerisinde tanımlanan bloklar şunlardır,

- src: ses kartını giriş kaynağı olarak audio_sample_rate örnekleme hızına göre tanımlar.
- amp: giriş işaretinin genliğini 1000 kat arttırır
- sink: amp bloğu çıkışının audio_sample_rate örnekleme hızında frekans spektrumunu grafiksel olarak gösterir.

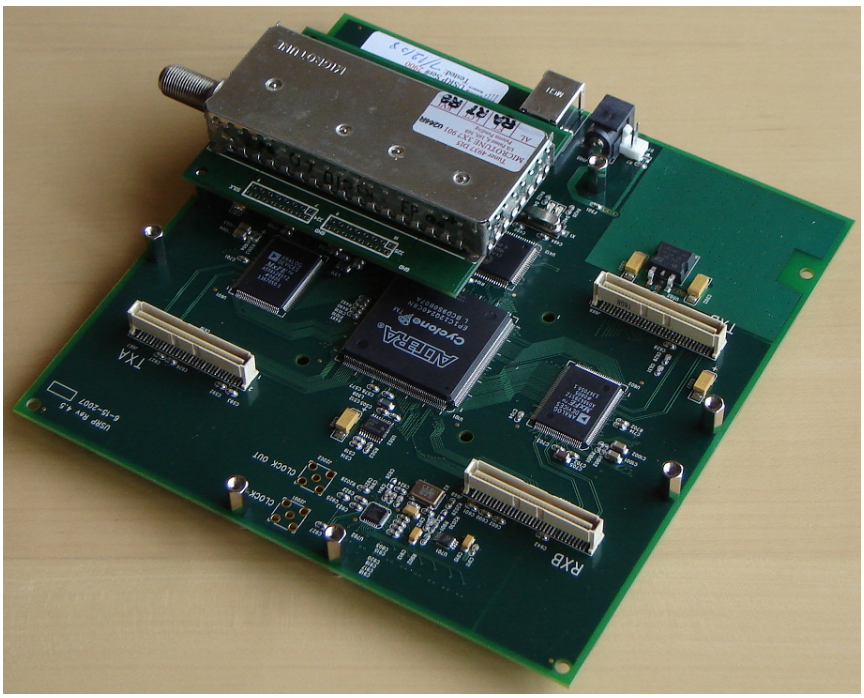
Daha sonra bu üç blok connect metodu ile birbirine bağlanır. Bu aşamada akış diyagramını başlatacak start metodu projede görülmemektedir, çünkü std_top_block sınıfı kurucu fonksiyonu işletirken start metodunu otomatik olarak çağırmaktadır. Proje işletildikten sonra görülecek olan frekas spektrumu penceresi Şekil 4.7'de verilmiştir.



Şekil 4.7: Örnek GNU Radio projesi FFT penceresi

4.5. Universal Software Radio Peripheral

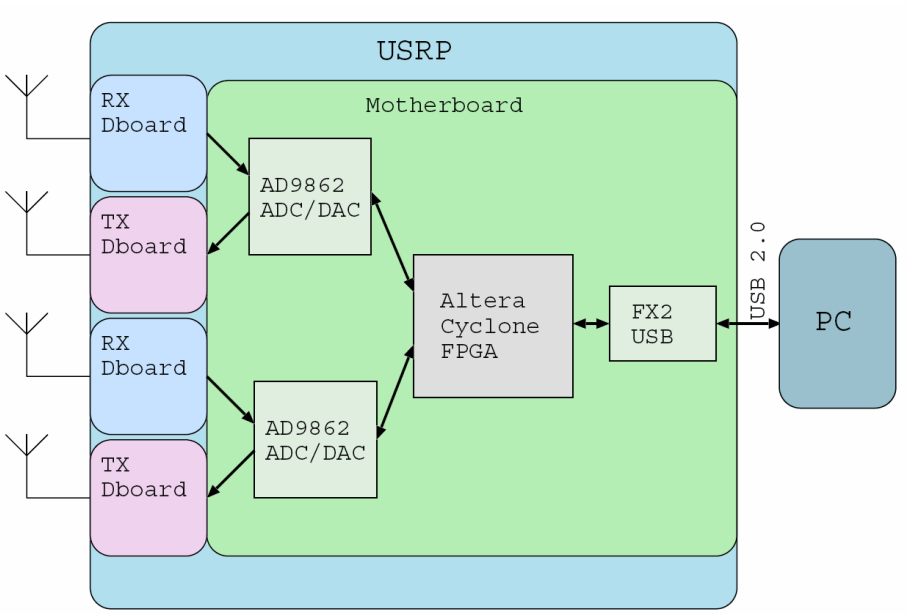
USRP, GNU Radio projesi kapsamında geliştirilmiş açık kaynak kodlu bir yazılım tanımlı radyo donanımıdır. USRP (Şekil 4.8), bir anakart ve 0-2.4 GHz aralığında yayın alan ve ileten çeşitli genişleme kartlarından oluşur. USRP'yi istenilen frekans bantına çekmek için bu genişleme kartları kolaylıkla değiştirilebilir. USRP'ye ait tüm devre şemaları, FPGA (Field Programmable Gate Array), Verilog kodları GNU GPL lisansı altında ücretsiz olarak dağıtılmaktadır.



Şekil 4.8: USRP anakartı ve TVRX genişleme kartı

4.5.1. USRP mimarisi

USRP anakartının ana bileşeni Altera Cyclone EP1C12 FPGA oluşturmaktadır, iki adet AD9862 karışık sinyal ön uç işlemcisi ve Cypress FX2 USB2.0 arayüz entegresi diğer bileşenleri oluşturmaktadır. Şekil 4.9'da USRP mimarisi blok olarak verilmiştir.



Şekil 4.9: USRP mimarisi blok diyagramı [20].

Harici genişleme kartları bant filtreleme, RF frekansından IF (ara frekans)'a dönüşüm ve ya IF'ten RF frekansına dönüşüm gibi analog işlemleri gerçekleştirirler. USRP IF'i alıcı kesimde örnekleyerek, PC'ye USB 2.0 veriyolu ile transfer etmektedir. Verici kesimde ise USB'den alınan IF işareti interpolasyon işlemine tabi tutularak DAC ile analog forma dönüştürülür. Harici genişleme kartları ile IF işareti istenilen RF bant aralığına çekilebilir.

4.5.2. Cypress FX2 mikrodenetleyici

FX2 gömülü USB 2.0 çekirdeği içeren 8051 uyumlu bir mikrodenetleyicidir. FPGA'e paralel bir veriyolu ile bağlıdır. Bu veriyolunun yanısıra GPIF (General Purpose Interface) adı verilen kontrol işaretlerine sahiptir. Bu kontrol işaretleri sayesinde FPGA, FX2'yi farklı amaçlar için kontrol edebilir. FX2 ile USRP tüm

USB 2.0 arayüzü işlemlerini gerçekleştirir.

4.5.3. ADC ve DAC

USRP'de iki adet AD9862 karışık sinyal işlemcisi vardır. Her AD9862 iki adet ADC ve iki adet DAC içerir. Tablo 4.3'te USRP ADC/DAC parametreleri verilmiştir.

Tablo 4.3 USRP ADC/DAC parametreleri

	DAC	**ADC**
Örnekleme hızı	128 Msps	64 Msps
Örnekleme çözünürlüğü	14-bit	12-bit
Adet	4	4

ADC'den hemen önce bir PGA (Programlanabilir Kazanç Yükselteci) bulunur, bu sayede 20 dB'ye kadar bir kazanç sağlanabilir. Buradaki amaç ADC'nin 72 dB dinamik aralığının tümünü kullanabilmektedir. ADC'nin tüm aralığı 2V Vpp ve giriş empedansı 200 Ω farksaldır. Örnekleme işleminde işaret bant genişliği ile SNR (Signal to Noise Ratio) arasındaki bağıntı [21],

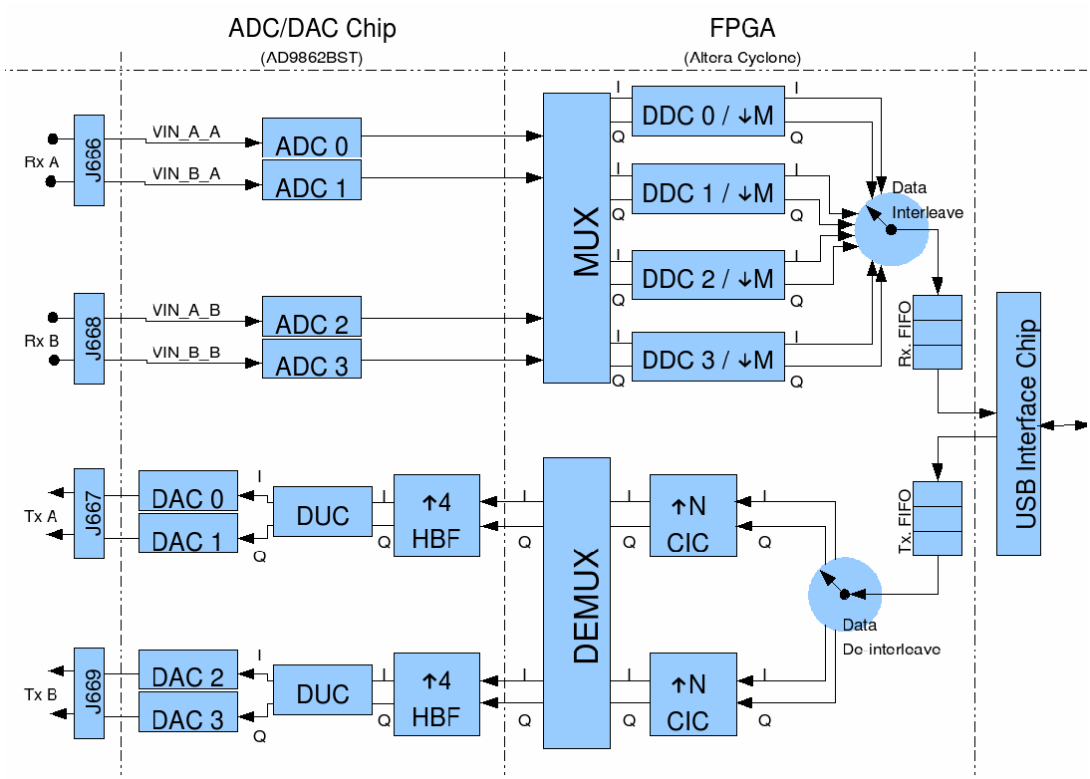
$$\mathrm{SNR} = 6{,}02\mathrm{N} + 1{,}76\mathrm{dB} + 10\log\left(\frac{f_s}{2B}\right) \tag{4.1}$$

olarak verilebilir. Bu eşitlikte,
N: ADC'nin örnekleme çözünürlüğü
f_s: ADC'nin örnekleme hızı
B: Alınmak istene işaretin bant genişliği
olarak tanımlanır.

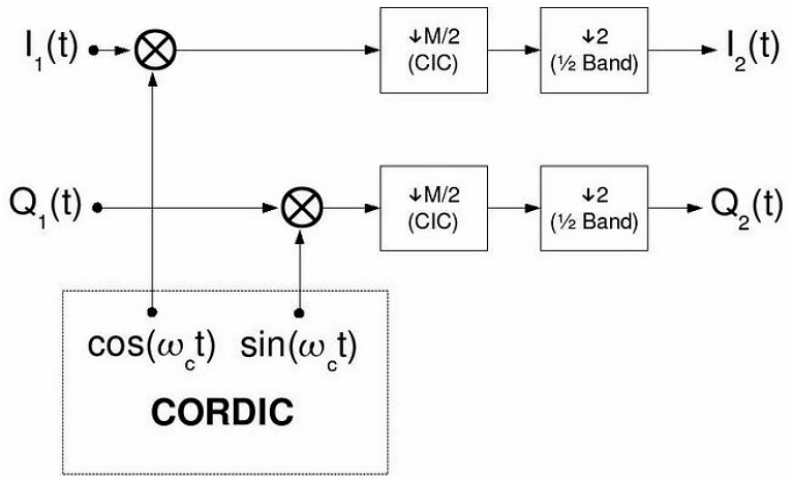
Verici kısmı ise çıkış örneklerini DAC örnekleme hızına dönüştürmek için bir yükselteç ve ara değer kestirimi işlemi sunar, daha sonra DAC tarafından analog IF işareti oluşturulur. 50 Ω'luk farksal çıkışı olan DAC'tan sonra 20 dB kazanca kadar programlanabilir bir PGA bulunur.

4.5.4. FPGA

USRP anakartı ile birlikte gelen Altera Cyclone FPGA yüksek hızlı işaret işleme işlemlerini gerçekleştirir, bunun yanında örneklenmiş IF işaretini istenilen bant genişliğine düşürerek PC işlem gücünden tasarruf sağlayabilir. Yüksek seviyeli FPGA işaret işleme blokları Şekil 4.10'da verilmiştir.



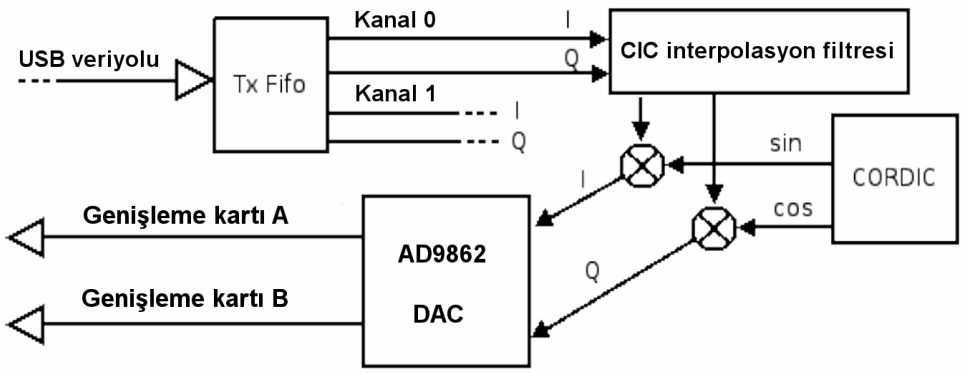
Şekil 4.10: FPGA işaret işleme blokları [20].



Şekil 4.11: DDC blok gösterimi [21].

FPGA, ADC'lerden gelen örneklenmiş giriş işaretlerini bir çoklayıcı ile DDC (Digital Down Converter)'lere iletir. DDC'ler geleneksel süperheterodin yapıya benzer bir teknik kullanırlar. DDC, CORDIC (Coordinate Rotation Digital Computer) [23] algoritması ile gerçeklenmiş programlanabilir bir taşıyıcı üreteci ile örneklenmiş işareti çarparak alınmak istenen işareti 0 Hz ve civarına çeker, daha sonra CIC (Cascaded Integrator Comb) filtresi ile istenilen örnekleme oranına düşürür. Şekil 4.11'da DDC'nin blok gösterimi verilmiştir.

Verici bölümü yayın alma bölümüne benzer bir yapıdadır. DAC 128 Msps örnekleme hızında olduğundan dolayı, USB'den alınan IF işaretinin örnekleme hızının bu orana çıkarılması gerekmektedir. Bu işlem interpolasyon olarak adlandırılır ve FPGA'de gerçeklenmiştir. Yayın alma bölümündeki DDC'nin tam ters işlemini yapan DUC(Digital Up Converter) AD9862'de dahili olarak mevcuttur. Şekil 4.12'de DUC'ın blok gösterimi verilmiştir.

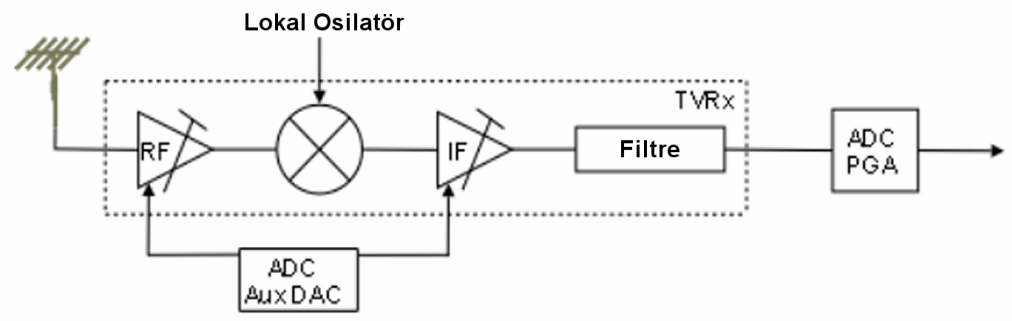


Şekil 4.12: DUC blok gösterimi [21].

4.5.5. TVRX harici genişleme kartı

TVRX, VHF ve UHF bantlarını kapsayan bir genişleme kartıdır. 50-860 MHz aralığındaki istenilen 6 MHz'lik bant aralığını IF frekansına dönüştürebilir. Tüm frekans ayarlama ve AGC (Auto Gain Control) fonksiyonları GNU Radio ile kontrol edilebilmektedir. TVRX 43,75 MHz ve 5,75 MHz olarak iki IF çıkışına sahiptir. USRP anakartı 5,75 MHz'lik ikinci IF çıkışını kullanmaktadır. TVRX, 50 dB

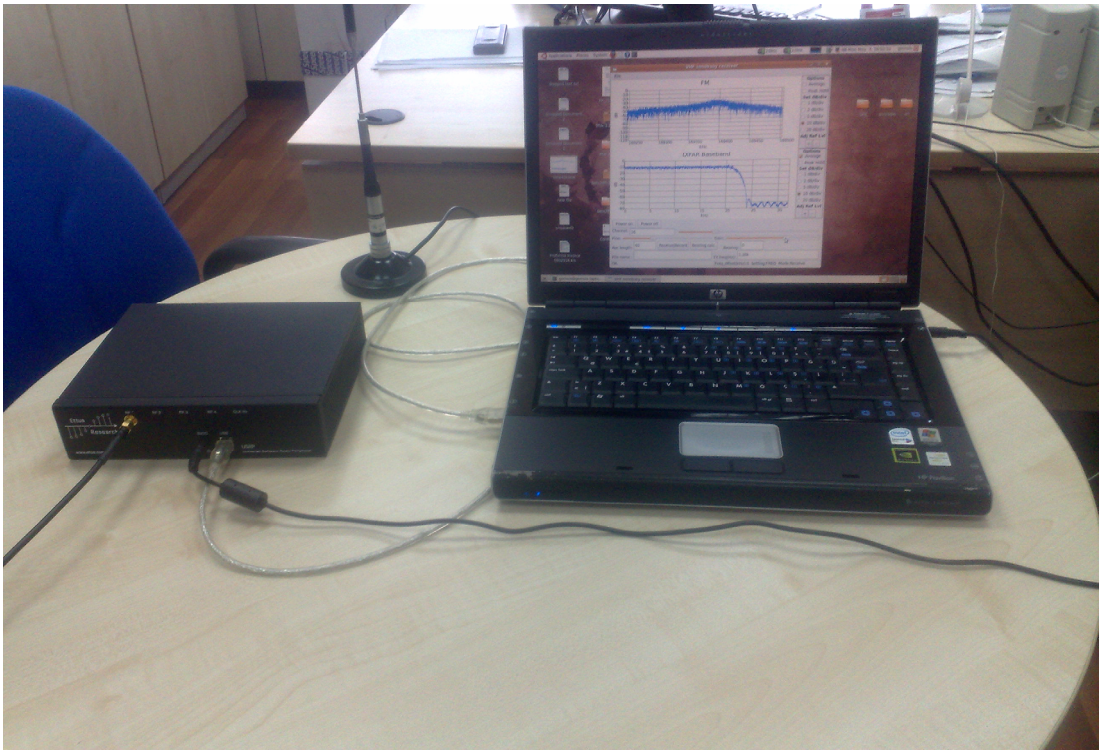
sahasında RF otomatik kazanç kontrolü ve 33 dB IF otomatik kazanç kontrolüne sahiptir. Şekil 4.13'te TVRX'in blok diyagramı verilmiştir



Şekil 4.13: TVRX genişleme kartı blok diyagramı [21].

5.YAZILIMIN GERÇEKLENMESİ

Tez kapsamında geliştirilen yazılım tanımlı radyo alıcısı yazılımı DIFAR tipindeki sonoboyları tam olarak desteklemekte olup python programlama dili ile geliştirilmiştir. Bu yazılım linux tabanlı 32 bitlik bir yazılım olup Ubuntu ve Pardus dahil çeşitli linux dağıtımlarında çalışabilmektedir. Şekil 5.1'de hazırlanan test düzeneği gösterilmiştir.



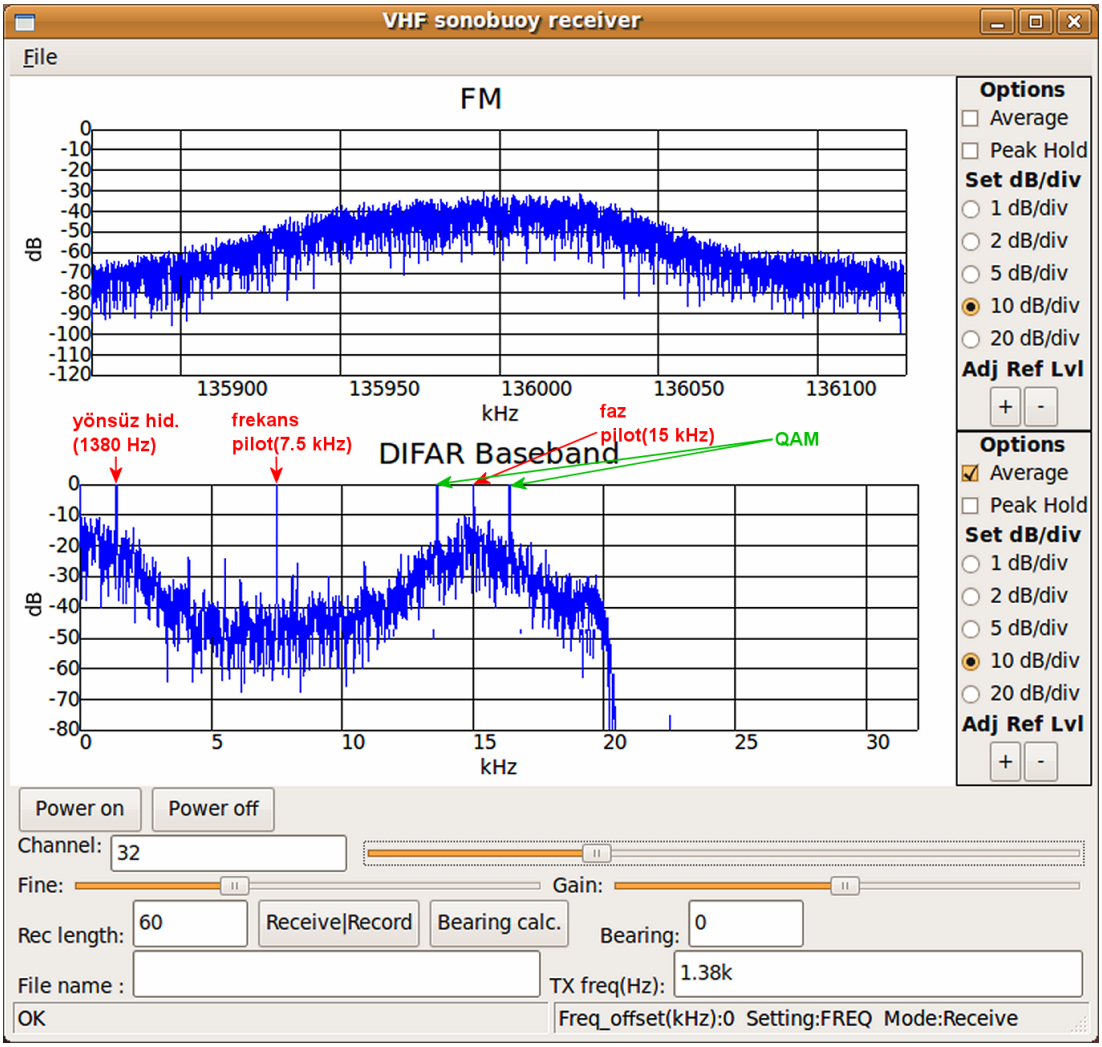
Şekil 5.1: Hazırlanan test düzeneği

Yazılım, bölüm 3'te değinilen matematiksel model temel alınarak bölüm 2 ve 4'te değinilen hususlar doğrultusunda geliştirilmiştir.

Bu bölümde yazılım tanımlı radyo alıcısı ve demodüle edilen akustik işaretlerin analizi kısımları iki ayrı bölümde incelenecektir.

5.1. Yazılım Tanımlı Radyo İle Sonoboy Alıcının Gerçeklenmesi

Tez kapsamındaki projenin FM alıcısı ve grafik arayüzü GNU Radio ve USRP ile gerçeklenmiştir.



Şekil 5.2: Yazılım grafik arayüzü

Geliştirilen program konsol ekranından,

sudo python usrp_snby_rcv.py

şeklinde başlatılmaktadır. Program açıldığında ekranda görülen grafik arayüzü Şekil 5.2'de verilmiştir.

Yazılım ekranı dört alt satırdan oluşmaktadır. İlk satırda antenden alınan işaretin FM

spektrumu gösterilmektedir. Bu işlem esasen giriş işaretinin FFT (Fast Fourier Transform)'sinin alınıp grafiksel olarak ekrana basılmasıdır. İkinci satırda ise demodüle edilmiş FM işaretinin temelbant spektrumu gösterilmektedir. Bu gösterim ile alınan işaretin kalitesi ile ilgili fikir edinilebilmektedir. Örneğin iyi olmayan bir alış performansında dip gürültüsü fazla olacağından dolayı sağlıklı bir analiz yapılamaz. Üçüncü satırda FM alış parametrelerinin, kayıt/analiz işlemlerinin ve çeşitli kontrollerin bulunduğu bir pencere bulunmaktadır. Bu penceredeki grafiksel kontrolleri sırasıyla aşağıdaki gibi verebiliriz.

- "Power on" butonu: Bu buton kontrolü, durdurulmuş bir GNU Radio akış diyagramını tekrar başlatmak amacıyla kullanılır.
- "Power off" butonu: Bu buton yürüyen bir akış denetimini durudurmak için kullanılır. Bu özellik önceden kaydedilmiş kayıt örneklerini analiz ederken yararlıdır, çünkü akış diyagramını durdurarak kazanılacak işlem gücü işlem yoğun bir görev olan akustik analizde kullanılabilir. İstenildiğinde "Power on" butonu ile akış diyagramı tekrar başlatılabilir.
- "Channel": Bu alana Tablo 3.1'de verilen sonoboy RF kanallarından istenilen biri girilebilir, enter tuşuna basıldığında istenilen kanal ayarları yapılmış olacaktır. Bu alandaki değer aynı zamanda sürüklenen bir çubuk ile ayarlanabilmektedir. Bu çubuk sürüklenerek 1-99 aralığında bir sonoboy kanalı kolayca seçilebilir.
- "Fine": Bu sürükleme çubuğu hassas kanal ayarı yapmak amacıyla kullanılır. DIFAR sonoboylar genelde 10-20 kHz civarı bir frekans hatasına sahiptir, bunun nedeni maliyeti düşürmek için toleransı yüksek osilatörlerin kullanılmasından kaynaklanmaktadır. Hassas ayar ile kanal frekansına -10,+20 kHz aralığında ofset eklenebilir. Hassas ayar sonucunda temelbant daki işaretin SNR oranı iyileşir.
- "Gain": Bu kaydırma çubuğu ile PGA'nın kazancı 0-20 dB aralığında ayarlanabilmektedir. Uygun değer FM ve temelbant spektrumuna bakılarak ayarlanabilir, FM spektrumundaki işaret ne doyuma gidecek, ne de çok zayıf olacak şekilde optimum bir değer belirlenebilmektedir.
- "Rec length": Bu alana temelbant işaretin ne kadar bir aralıkta kaydedileceği saniye olarak girilir.
- "Receive|Record": Bu buton ile alıcının kayıt moduna girmesi sağlanır. "Rec length" alanı ile belirlenen süre boyunca temelbant işaret kayıt edilir. Kayıt işlemi bittikten sonra akış diyagramı kendini dondurur, tekrar normal çalışma

moduna geri dönmek için bu butona ikinci kez basılması yeterlidir.

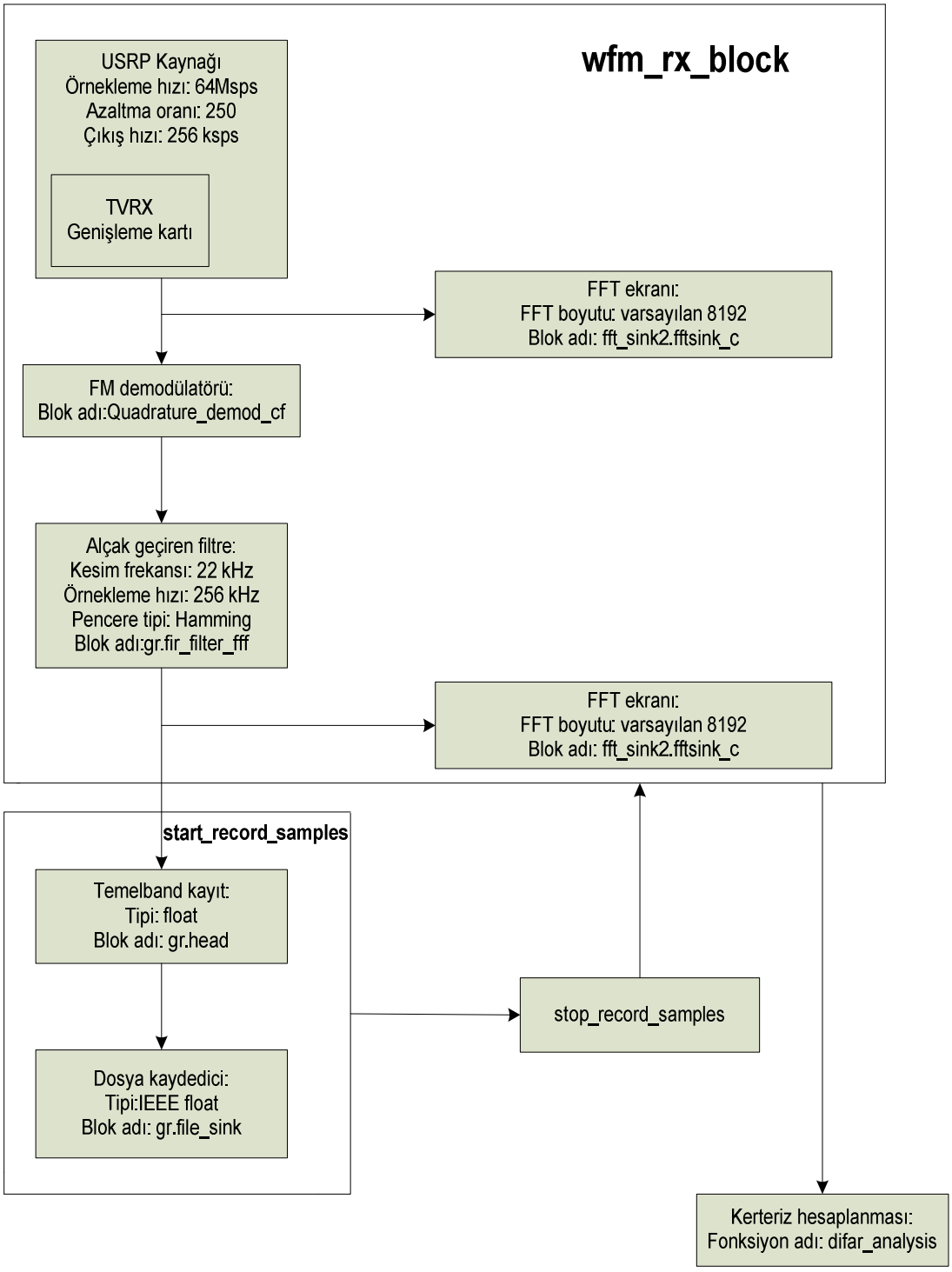
- "Bearing calc": Bu buton ile kaydedilmiş temelbant işaretlerinin akustik analizi yapılmaktadır.
- "Bearing": Bu alanda akustik analiz sonucu hesaplanan kerteriz değeri gösterilmektedir.
- "Filename": Bu alanda kaydedilen temelbant işaretin dosya ismi, kayıt tarihi, kayıt saati ve kanal numarası formatında gösterilir. İkinci bir amacıda istenilen zamanda daha önceden kaydedilmiş kayıtların dosya adının girilerek analiz edilmesidir.
- "TX freq(Hz)": Bu alana test senaryosunda analiz edilecek hedef olan aktif sonarın ses frekansı girilir.

Bu özelliklere ek olarak arayüzün dördüncü satırında bazı bilgi mesajları bulunmaktadır. Bunları şöyle sıralayabiliriz,

- Kanal kitlenme uyarısı: İstenilen kanal seçildiğinde eğer doğru frekansa kilitlenilirse "OK" mesajı, eğer kilitlenmez ise "Failed" mesajı görüntülenir.
- Freq_offset(kHz): Hassas ayar ile kanal frekansına eklenen frekans ofset değeri burada görüntülenir.
- Mode: Alıcının hangi çalışma modunda olduğunu gösterir. Normal çalışma modunda "Receive", kayıt modunda "Record" mesajı görüntülenir.

5.1.1. Yazılım tasarımı

Bu bölümde, bölüm 4.4.1'de verilen GNU Radio yazılım geliştirme metodları ile sonoboy alıcının VHF(Çok Yüksek Frekans) kısmının geliştirilmesi incelenecektir. Yazılımın geliştirilmesinde Python programlama dilinin sunduğu nesne yönelimli programlama ve yapısal programlama metodlarından yararlanılmıştır. Bölüm üç kısımdan oluşmaktadır. Birinci bölümde ana sınıfın türetilmesi ve kurucu fonksiyonun tanımlanması, ikinci bölümde arayüz tasarımı, üçüncü bölümde ise ana sınıfın sahip olduğu yardımcı fonksiyonlar işlenecektir. Basitlik olması açısından program kodları incelenirken bazı tanımlamalar atlanabilecektir, program kodlarının tamamı Ek A'da verilmiştir. Program kodlarının basitleştirilmiş bir akış diyagramı Şekil 5.3'te verilmiştir.



Şekil 5.3: Yazılım akış diyagramı

5.1.1.1. wfm_rx_block sınıfı

Tez kapsamındaki projede GUI arayüzünü ve akış diyagramını oluşturmak amacıyla stdgui2 modülünün "std_top_block" sınıfından "wfm_rx_block" sınıfı türetilmiştir.

Her NYP sınıfında olduğu gibi bir kurucu sınıfa sahip olan wfm_rx_block sınıfının kurucu "init" fonksiyonu sistemin ilklendirilmesi maksadıyla kullanılmaktadır.

Şekil 5.4'te init kurucu fonksiyonunun program kodlarının blok gösterimi verilmiştir.

```
class wfm_rx_block (stdgui2.std_top_block):
   def __init__(self,frame,panel,vbox,argv):
      stdgui2.std_top_block.__init__ (self,frame,panel,vbox,argv)

      konsoldan parametre alan rutinler
      global değişkenler

      self.u = usrp.source_c()                                          1. Blok
      adc_rate = self.u.adc_rate()
      usrp_decim = 250                                  #CIC azaltma oranı
      self.u.set_decim_rate(usrp_decim)
      usrp_rate = adc_rate / usrp_decim                 # 256 ksps
      chanfilt_decim = 4                                # filtre azaltma oranı
      self.demod_rate = usrp_rate / chanfilt_decim      # 64 ksps
      self.rx_subdev_spec = pick_subdevice(self.u)
      self.u.set_mux(usrp.determine_rx_mux_value(self.u,self.rx_subdev_spec))
      self.subdev = usrp.selected_subdev(self.u, self.rx_subdev_spec)

      fm_demod_gain = usrp_rate/(2*math.pi*75e3)                        2. Blok
      self.fm_demod = gr.quadrature_demod_cf (fm_demod_gain)

      chan_filt_coeffs = gr.firdes.low_pass (1,                         3. Blok
                                             usrp_rate,
                                             22e3,
                                             4e3,
                                             gr.firdes.WIN_HAMMING)
      self.chan_filt = gr.fir_filter_fff (chanfilt_decim, chan_filt_coeffs)

      self.connect (self.u, self.fm_demod, self.chan_filt)

      self._build_gui(vbox, usrp_rate, self.demod_rate, options)

      gain_range = self.subdev.gain_range()                             4. Blok
      self.gain = float(gain_range[0]+gain_range[1])/2
      self.freq_init = 136e6
      self.u.tune(0, self.subdev, self.freq_init)
      self.set_gain(self.gain)
```

Şekil 5.4: wfm_rx_block sınıfı kurucu fonksiyonu blok gösterimi

1. blokta USRP ve genişleme kartına ait ilklendirme ve GNU Radio blokları tanımlanmaktadır. Bu bloğu ayrıntılı incelersek,

- self.u, usrp modülünün source_c() sınıfını tutan bir nesnedir. source_c, USRP'yi kompleks formatta veri kaynağı olarak tanımlayan bir sınıftır. Bundan sonra USRP SDR platformu self.u olarak çağrılacaktır.
- self.u sınıfının adc_rate() fonksiyonu USRP'nin desteklediği maksimum örnekleme değerini döndürür. adc_rate değişkeninde tutulan bu değer CIC azaltma oranına bölünerek USRP çıkış örnekleme hızı belirlenir.
- self.u sınıfının set_decim_rate fonksiyonu FPGA'de gerçeklenen CIC'in azaltma oranını belirler. Projede bu değer 250 olarak seçilmiştir. Buna göre USRP çıkış örnekleme hızı 256 ksps olacaktır.
- chanfilt_decim oranı 3. blokta tanımlanan FIR filtrenin örnek azaltma oranını belirlemektedir. Buna göre self.demod_rate ile belirlenen değer 64 ksps olacaktır. Bu değer ile Nyquist örnekleme teorisine göre 32 kHz'e kadar olan temelbant işaret elde edilebilir. Bu ise Şekil 3.7'de verilen sonoboy çoklanmış birleşik işareti için yeterlidir.
- pick_subdevice fonksiyonu GNU Radio ile gelen bir fonksiyondur. Parametre olarak USRP sınıflarını alan fonksiyonunu amacı USRP'ye takılı olan genişleme kartlarını tespit etmektir, usrp modülüne ait selected_subdev fonksiyonu ile tespit edilen genişleme kartlarından kullanılmak istenen bir tanesi seçilebilir.
- usrp modülünün determine_rx_mux_value fonksiyonu seçilen genişleme kartına göre FPGA'de gerçeklenen çoklayıcı değerini hesaplar. Daha sonra hesaplanan bu değer self.u sınıfının set_mux fonksiyonu ile çoklayıcıya iletilir.

2. blokta genişbant FM demodülatörü tanımlanmaktadır. Demodülatöre ait kazanç değeri varsayılan olarak bırakılmıştır. Buna göre,

$$\text{fm_demod_gain} = \frac{256.10^3}{2.3{,}14.75.10^3} \cong 0{,}544$$

olarak hesaplanır. FM demodülasyon işlemini gerçekleştiren quadrature_demod_cf fonksiyonu fm_demod_gain değerini parametre olarak alır.

3. blokta demodüle edilmiş temelbant işaret 22 kHz kesim frekansı olan hamming pencereli FIR filtreden geçirilir. gr modülünün firdes_low_pass fonksiyonu FIR filtrenin katsayılarını hesaplayan bir fonksiyondur. Aldığı parametreler sırasıyla şöyledir,

- gain: filtre kazancı
- sampling_freq: filtre örnekleme frekansı (Hz)
- cutoff_freq: kesim frekansı (Hz)
- transition_width: filtrenin zayıflatmaya başladığı frekans değeri
- window: pencereleme fonksiyonu

Hesaplanan parametreler gr modülünün fir_filter_fff fonksiyonuna 1. blokta açıklanan chanfit_decim örnek azaltma oranı ile parametre olarak geçirilerek alçak geçiren self.chan_filt bloğu oluşturulur.

Oluşturulan bloklar daha sonra connect metodu ile birbirine bağlanır, artık akış diyagramı hazırdır.

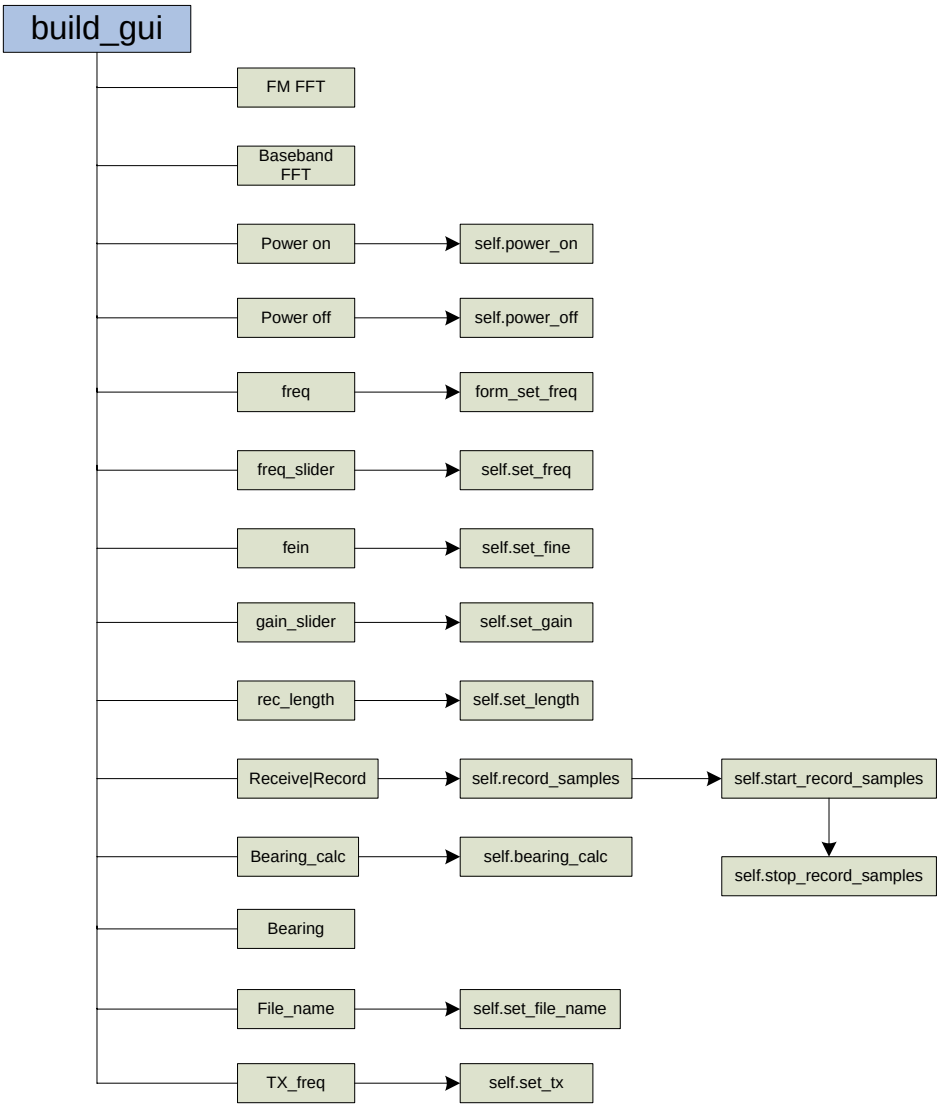
Son olarak 4. blokta bazı ilklendirmeler yapılmaktadır. Buna göre subdev sınıfının gain_range() fonksiyonuyla bağlı olan genişleme kartının kazanç aralığı elde edilmektedir. Daha sonra set_gain fonksiyonu ile ortalama bir kazanç değeri verilmektedir. Benzer şekilde self.u nesnesinin tune fonksiyonu ile 136 MHz değeri başlangıç frekansı olarak verilmektedir.

5.1.1.2. build_gui GUI fonksiyonu

build_gui fonksiyonu tez kapsamındaki projede kullanıcı grafik arayüzü işlemlerinden sorumludur. Bu bölümde kullanıcı grafik arayüzü nesnelerinin yaratılması ve bunların yardımcı fonksiyonlar ile olan etkileşimi incelenecektir.

GNU Radio, kullanıcı grafik arayüzlerinin kolayca yaratılması için wxgui alt modülünü sunar, bu GNU Radio modülü wxPython ile oluşturulmuş hazır sınıflardan oluşur. Bu sınıflardan bazıları şöyledir,

- stdgui2: Temel GUI sınıfı, buton, metin alanı gibi temel nesneler bu sınıf ile yaratılır.
- fftsink: Giriş veri akışını FFT'sini alarak ekrana basar.
- scopesink2: Giriş veri akışını zaman ekseninde ekrana basar.
- numbersink2: Giriş verisini nümerik olarak ekrana basar.



Şekil 5.5: build_gui GUI fonksiyonu blok diyagramı

Şekil 5.5'te tez kapsamındaki projede wxgui ile hazıarlanmış kullanıcı grafik arayüzü nesneleri ve bunların yardımcı fonksiyonlarının blok diyagramı verilmiştir. Birinci sütundaki nesneler Şekil 5.2'de görülen grafik nesneleridir. İkinci sütunda bu nesnelerin çağırdığı fonksiyonlar verilmiştir. Bu fonksiyonların detayları aşağıda verilmiştir.

- "self.power_on" fonksiyonu: Bu fonksiyon "Power on" butonu tarafından çağırılır. Amacı durdurulmuş bir akış diyagramını tekrar başlatmaktır. Buna basit bir örnek Şekil 5.6'da verilmiştir.
- "self.power_off" fonksiyonu: Bu fonksiyon "Power off" butonu tarafından çağırılır. Amacı çalışan bir akış diyagramını durdurmaktır. Buna basit bir örnek Şekil 5.6'da verilmiştir.

```
class my_top_block(gr.top_block):
    def __init__(self):
        gr.top_block.__init__(self)
        ... # Burada GNU Radio işlem blokları tanımlanır

if __name__ == '__main__':
    self.power_on ()    # akış diyagramı başlatılır
    sleep(5)            # 5 saniye gecikme
    self.power_off()    # akış diyagramı power_off fonksiyonu ile durdurulur
    self.power_on ()    # akış diyagramı tekrar başlatılır
```

Şekil 5.6: Örnek bir akış diyagramı kontrolü

- "self.set_freq" fonksiyonu: Bu fonksiyon "Channel" isimli bilgi girişi yapılan alan ve kaydırma çubuğu tarafından çağırılır, fonksiyona parametre olarak girilen 1-99 arası kanal numaraları burada Tablo 3.1'e göre frekans değerlerine çevirilerek alıcının frekans ayarı yapılır, eğer işlem başarılı olursa GUI'nin 4. satırında "OK" bilgi mesajı yazdırılır.
- "self.set_fine" fonksiyonu: Bu fonksiyon "Fine" kaydırma çubuğu tarafından çağırılır, fonksiyona parametre olarak girilen -10,+20 arası değerler, 1 kHz ve katları olarak frekans farkı değerlerine çevirilerek alıcının ayarlandığı frekans değerine ofset olarak eklenir, böylece ince ayar yapmak mümkün olur.
- "self.set_gain" fonksiyonu: Bu fonksiyon "Gain" kaydırma çubuğu tarafından çağırılır, fonksiyona parametre olarak girilen 0-115 arası değer ile PGA programlanarak IF seviyesi arttırılır.
- **"**self.set_length" fonksiyonu: Bu fonksiyon ile "Rec length" bilgi giriş alanına girilen saniye formatındaki değer "start_record_samples" fonksiyonuna geçirilerek kayıt yapılacak süre belirlenir.
- "self.record_samples" fonksiyonu: Bu fonksiyon "Receive|Record" butonu tarafından çağırılır, fonksiyon normal çalışma durumundaysa "self.start_record_samples" fonksiyonunu, eğer kayıt modundaysa "self.stop_record_samples" fonksiyonunu çağırır.
- "self.start_record_samples" fonksiyonu: Bu fonksiyon temelbant kayıt işlemlerini gerçekleştirir. Şekil 5.7'de ayrıntılı bir açıklama verilmiştir.
- "self.stop_record_samples" fonksiyonu: Bu fonksiyon alıcıyı kayıt modundan çıkararak normal moda geçirir. Şekil 5.8'de ayrıntılı bir açıklama verilmiştir.
- "self.bearing_calc" fonksiyonu: Bu fonksiyonu "Bearing_calc" butonu

tarafından çağırılır. Amacı kaydedilen temelbant örneklerinin akustik analizinin yapılarak kerteriz açısının hesaplanmasıdır.

```
def start_record_samples (self):
    …
    self.stop()       # akış diyagramı durdurulur
    self._head = gr.head(gr.sizeof_float, int(samples)) ()    # samples değişkeni kadar
                      #float örneği geçiren daha sonra akış diyagramını durduran blok
    …
    self.utfil = gr.file_sink(gr.sizeof_float,self.file_name)  # dosyaya float formatta
                                                               # yazan blok
    self.connect(self.chan_filt,self._head,self.utfil)  # bloklar birbirine bağlanır
    self.wait()  # akış diyagramının başlatılmasına hazırlık
    self.start()  # akış diyagramı başlatılır
    …
```

Şekil 5.7: "start_record_samples" fonksiyonu yapısı

```
def stop_record_samples (self):

    self.stop()          # akış diyagramı durdurulur
    self.disconnect(self.chan_filt,self._head,self.utfil)    # bağlı bloklar koparılır
    self.wait()          # akış diyagramının başlatılmasına hazırlık
    self.start()         # akış diyagramı başlatılıyor
    …
```

Şekil 5.8: "stop_record_samples" fonksiyonu yapısı

- "self.set_file_name" fonksiyonu: Bu fonksiyon "File_name" bilgi girişi alanına girilen dosya ismini "self.bearing_calc" fonksiyonuna geçirerek analiz edilecek dosya ismini belirler.
- "self.set_tx" fonksiyonu: Bu fonksiyon "TX_freq" bilgi girişi alanına girilen aktif sonar ses frekansını "self.bearing_calc" fonksiyonuna geçirir.

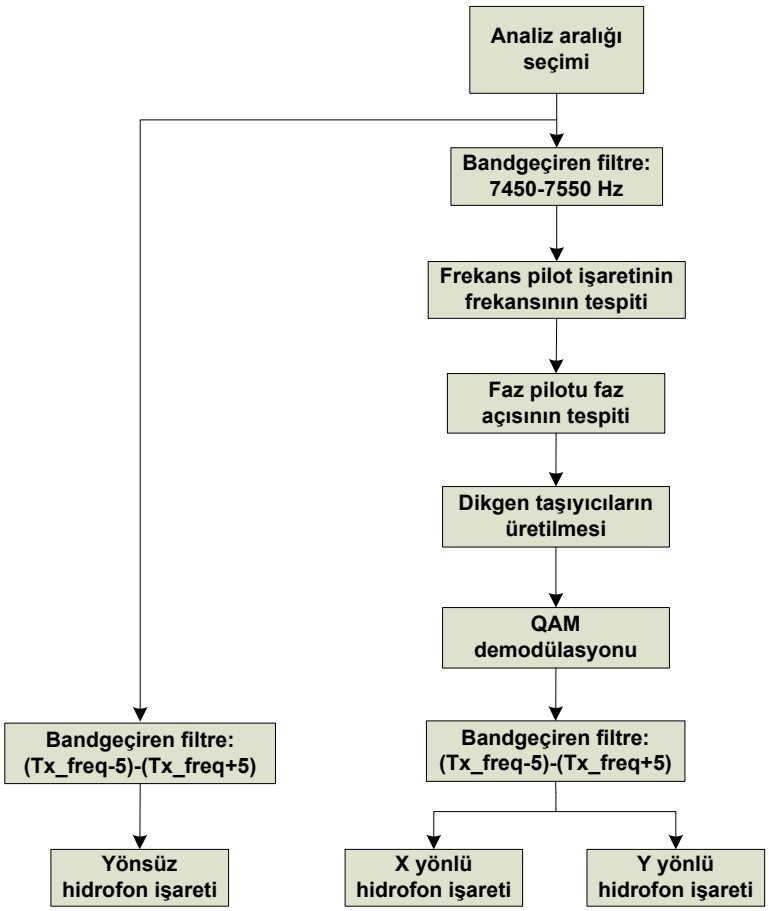
5.2. Temelbant İşaretten Kerteriz Açısının Hesaplanması

FM demodülasyonu sonucu elde edilen temelbant işaret eşitlik 3.15'te verilen bileşenlerden oluşmaktadır. Bu bölümde DIFAR standartındaki çoklanmış birleşik işaretten hedefin manyetik kuzeye olan kerterizinin hesaplanması incelenecektir. Bölümün ilk kısmında yönsüz ve yönlü hidrofon akustik işaretlerinin elde edilmesi, ikinci kısmında bu işaretlerden manyetik kerterizin elde edilmesi işlenecektir.

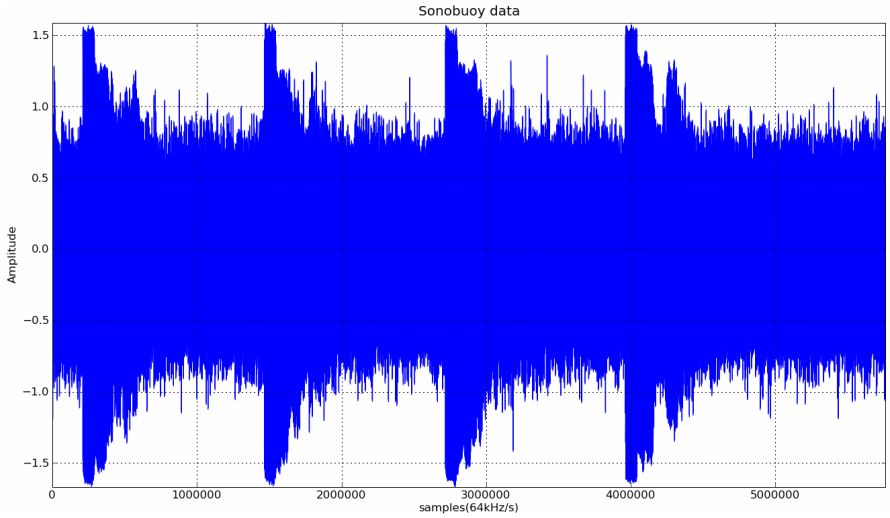
5.2.1. Yönlü ve yönsüz hidrofonların akustik işaretlerinin elde edilmesi

Bu bölümde Eşitlik 3.15'te verilen DIFAR çoklanmış işaretin Başlık 3.2.2.'de verilen matematiksel eşitliklere uygun olarak demodüle edilmesi incelenecektir. Bu işleme ait bir blok diyagram Şekil 5.9'da verilmiştir. Demodülasyon işlemini oluşturan blokları aşağıdaki gibi tanımlayabiliriz.

- "Analiz aralığı seçimi" : Tez kapsamındaki projenin saha çalışmaları sırasında deniz oratamındaki test ortamını akustik olarak aydınlatmak amacıyla 1380 Hz frekansında sonar kullanılmıştır. Bu sonar seçilebilen uzunluklarda darbe yayınımı yapmaktadır. Analiz edilecek darbe aralığı kullanıcı tarafından seçilmektedir. Buna göre temelbant işaret kayıtları bir diziye geçirilip daha sonra kullanıcıya kayıt sürecinde belirlenen örnekleme hızında zaman ekseninde grafiksel olarak gösterilmektedir.

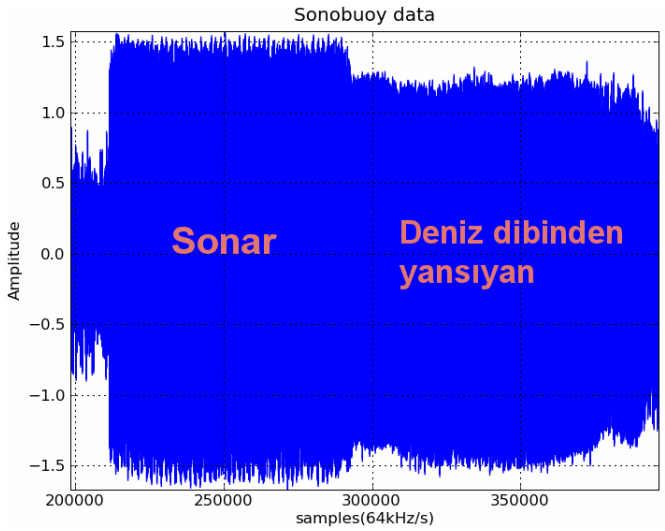


Şekil 5.9: DIFAR çoklanmış işaretin demodülasyonu



Şekil 5.10: Zaman ekseninde gösterilmiş temelbant işareti

Şekil 5.10'da saha çalışması sırasında kaydedilmiş bir temelbant kayıt örneği verilmiştir. Grafikte dört adet sonar darbe yayınımı görülmektedir. Örnek olarak ilk darbeyi Şekil 5.11'deki gibi gösterebiliriz. Şekilde sonar darbesi belirtilmiştir, bunun ardından deniz dibinden geri yansıyan akustik işaret görülmektedir. Bu çalışmada sadece sonar darbesi değerlendirmeye alınmıştır.



Şekil 5.11: İlk sonar darbesinin ayrıntılı gösterimi

- "Yönsüz hidrofon işaretinin elde edilmesi" : Şekil 3.7'de görüldüğü gibi yönsüz hidrofon işareti 0-2400 Hz aralığında modüle edilmemiş olarak iletilmektedir. Buna göre 1380 Hz'deki sonar için 1375-1385 Hz aralığında bir bant geçiren filtre kullanılarak yönsüz hidrofon çıkışı elde edilmektedir. Elde edilen yönsüz

hidrofon işareti Şekil 5.17'de gösterilmiştir.

- "Frekans pilotunun frekansının tespit edilmesi" : Eşitlik 3.10 ile verilen frekans pilotu, frekans modülasyonu ile iletim aşamasında temelbant'da oluşabilecek frekans kaymasını alıcıda belirlemek için kullanılan 7,5 kHz frekansında referans bir pilot işaretidir. Meydana gelen frekans kayıklığının iki katı kadar 15 kHz değerindeki faz pilotun saptığı kabul edilmektedir. Bu işlemden önce giriş işareti 7450-7550Hz aralığında bir bantgeçiren filtreden geçirilerek sadece frekans pilotun mevcut olduğu ara işaret üretilir. Frekans pilotunun hassas bir şekilde frekansını bulmak için matplotlib kütüphanesinin mlab modülünün "psd" [17] fonksiyonu kullanılmıştır. psd, "Welch ortalama periodogram" [23] metoduna göre giriş işaretinin spektral güç yoğunluğunu hesaplayan bir fonksiyondur. Yapısı,

  psd(x, NFFT, Fs, window)

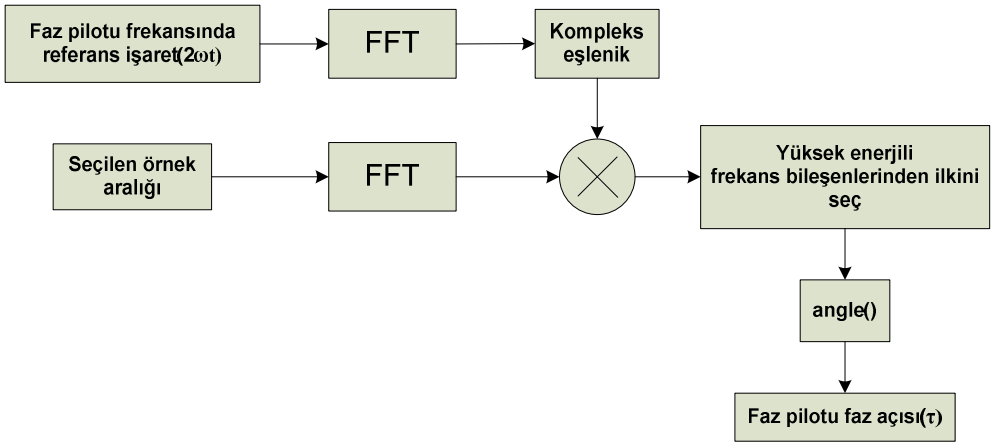
  şeklindedir. Parametrelerini aşağıdaki gibi sıralayabiliriz.

  - x: Örneklenmiş giriş dizisi
  - NFFT: Spektral güç yoğunluğu hesaplanırken uygulanacak FFT boyutu
  - Fs: Örnekleme hızı
  - window: Pencereleme fonksiyonu

örnek olarak,
x : Şekil 5.10'dan 25e4-26e4 aralığında
NFFT: 2^{22} (projede tercih edilen FFT boyutu)
Fs: 64000 Hz
window: mlab.window_none
seçilirse frekans pilotun gerçek frekansı ωt = 7500,15258789 Hz olarak bulunur. Frekans pilotun ideal şartlarda 7500 Hz'de bulunması gerektiği düşünülürse, iletim sürecindeki frekans kayması 0,15258789 Hz olarak bulunur.

- "Faz pilotu faz açısının tespiti ve dikgen taşıyıcıların üretilmesi": Eşitlik 3.11 ile verilen faz pilotu, QAM (dikgen genlik modülasyonu) ile modüle edilmiş yönlü hidrofonları demodüle etmek için alıcıda kullanılacak taşıyıcı pilot işaretidir. İdeal şartlarda 15 kHz frekansında olan faz pilot işareti iletim aşamasında temelbant'da oluşabilecek frekans kayması sonucunda kayabilmektedir, frekans

kayma değeri olarak frekans pilotta meydana gelen frekans kaymasının 2 katı alınmaktadır. Buna göre az önceki örnekteki faz pilotun frekansı 2ωt = 15000,30517578 Hz olarak bulunur.



Şekil 5.12: Faz pilotu faz açısının bulunması

Demodülasyon aşamasında kullanılan dikgen taşıyıcıları üretmek için frekans pilotunun (ωt) frekansının iki katının alınarak üretilen faz pilotu frekansındaki (2ωt) referans işaret ile gerçek faz pilotu arasındaki faz farkı (τ) bulunmalıdır. Bu işlemi gösteren blok diyagram Şekil 5.12'de verilmiştir. Çapraz korelasyon adı verilen bu yöntem iki işaretin benzerliğini ölçmede kullanılan yöntemlerden biridir ve eşitlik 5.1'deki gibi verilebilir.

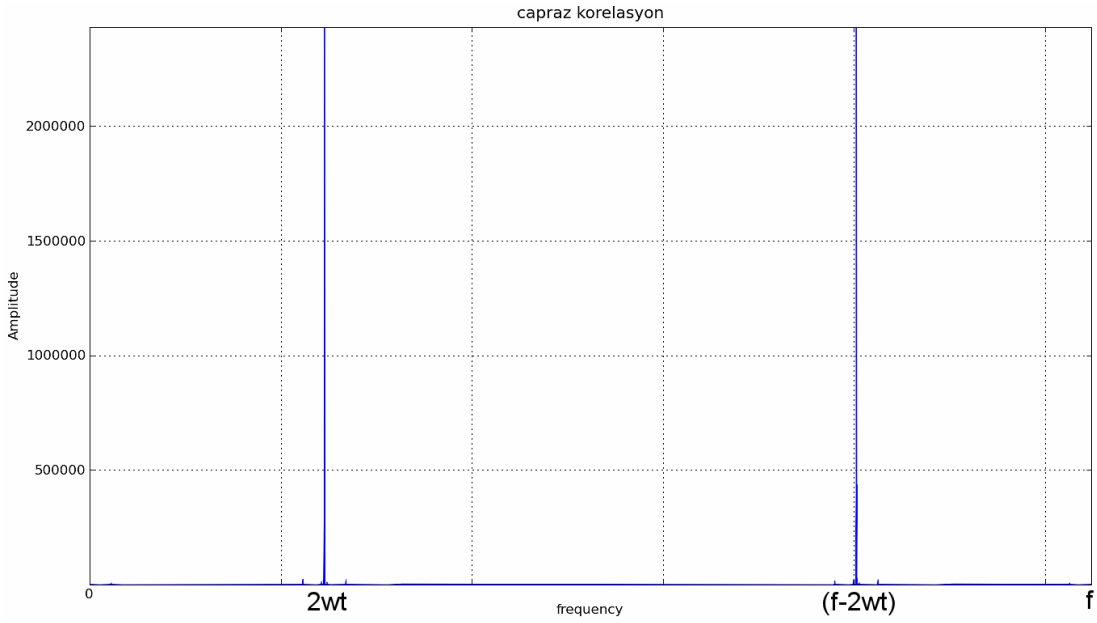
$$(f \star g)(x) = \int f^*(t)g(x+t)\,dt \tag{5.1}$$

Buna göre referans işaretin (2ωt) 2^{20} boyutlu FFT'si alınır, aynı şekilde seçilen örnek aralığınında aynı boyutta FFT'si alınır. Daha sonra FFT'si alınan referans işaretin kompleks eşleniği alınarak seçilen örnek aralığının FFT'si çarpılır. Bu işlemi,

τ = angle(FFT(A) * conj(FFT(B))) = angle(FFT(A))- angle(FFT(B))

şeklinde tanımlayabiliriz. Bu yöntemin kullanılmasının faydası belli bir bölgedeki dominant işareti bulmak için (örneğin burada faz pilotu) kullanışlı olması. Sonuçta faz farkını aradığımız frekans 2ωt ise, o frekansta bir sinüzoidal üretip kompleks eşleniğini alarak orijinal işaretimizle (seçilen örnek aralığı) çarparsak 2ωt ve onun frekans eksenindeki simetriği frekanslarında yüksek enerjili frekans bileşenleri

oluşur. Bu da aradığımız frekans'a (2ωt) ait faz farkını kolayca bulmamıza yarar. Şekil 5.13'te çarpım sonucu frekans ekseninde oluşacak görüntü verilmiştir.



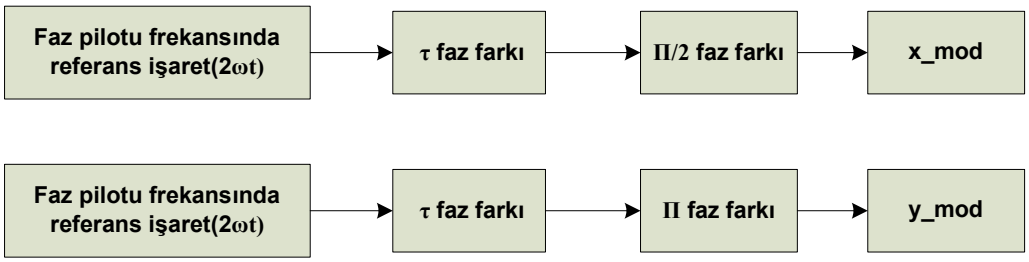
Şekil 5.13: Çapraz korelasyon sonucu

Çapraz korelasyon sonucu bulunan τ faz farkından yararlanarak dikgen taşıyıcıları (Şekil 5.15),

$$x_mod = \cos(2\omega t + \tau + \pi/2) \tag{5.2}$$

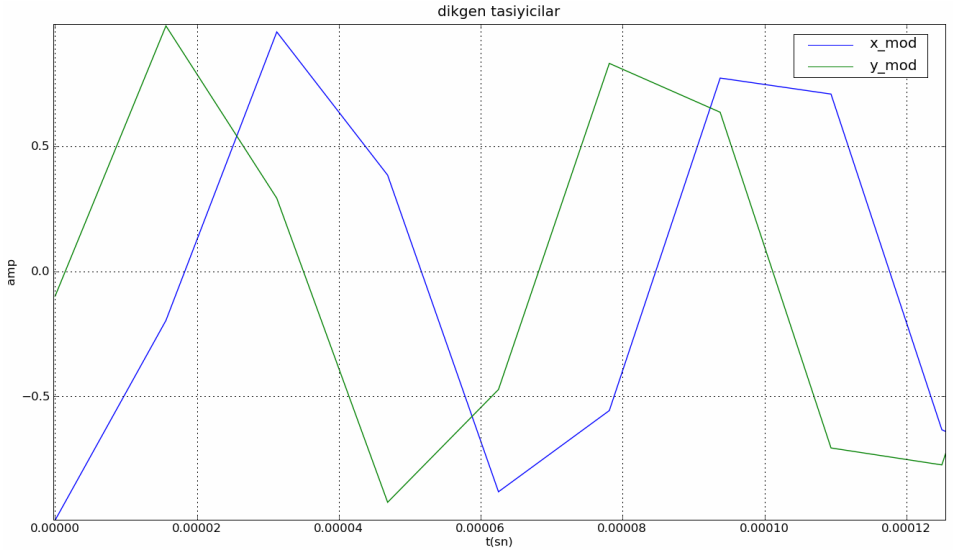
$$y_mod = \cos(2\omega t + \tau + \pi) \tag{5.3}$$

gibi üretebiliriz. Bu işleme ait blok diyagram Şekil 5.14'de verilmiştir.



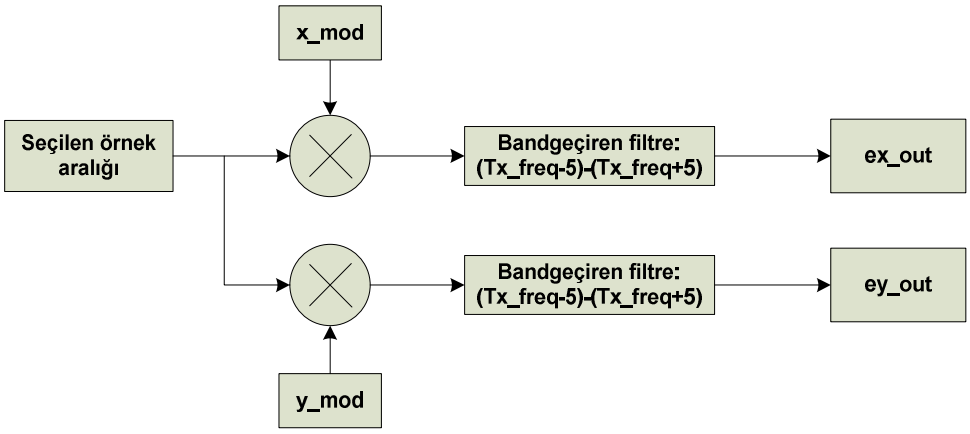
Şekil 5.14: Dikgen taşıyıcıların üretilmesi

- "QAM demodülasyonu" : x ve y yönlü hidrofon işaretlerinin elde edilmesi için Eşitlik 3.17 ve 3.18'deki denklemlerde belirtilen hususlara uygun olarak QAM

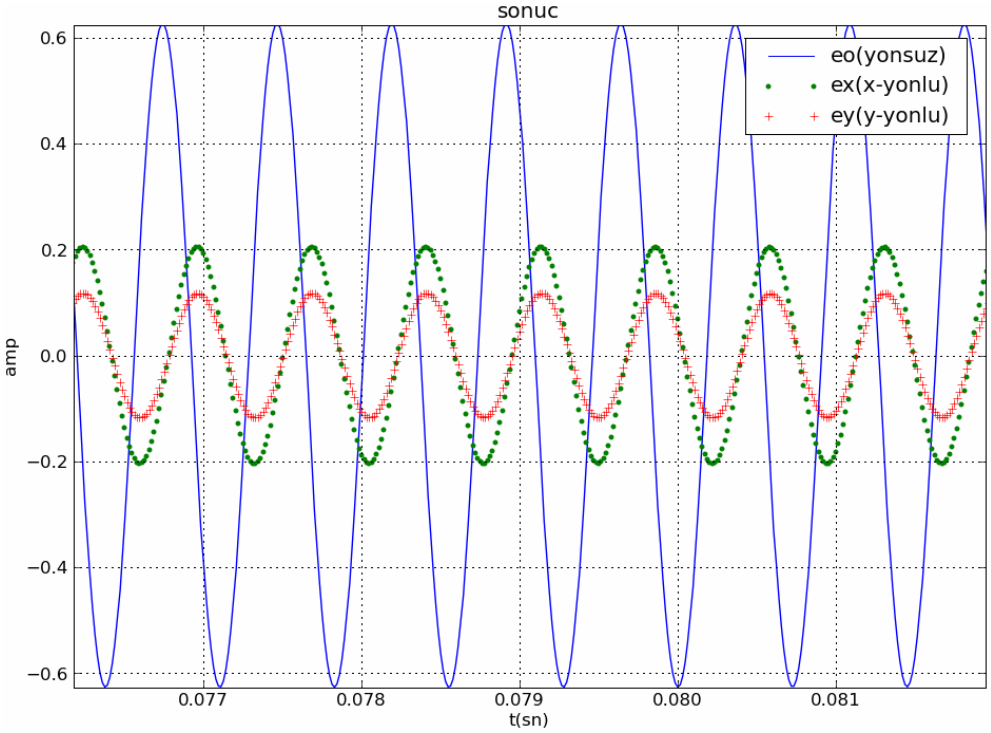


Şekil 5.15: Zaman ekseninde dikgen taşıyıcıların gösterilmesi

demodülasyonu yapılmalıdır. QAM demodülasyonuna ait blok diyagramı Şekil 5.16'da verilmiştir. Bir önceki konuda üretilen dikgen taşıyıcılar seçilen örnek aralığı ile çarpılırsa elde edilmek istenen yönlü hidrofon işaretleri ve 4ωt frekansında yüksek frekanslı bileşenler oluşur. Bu yüzden bant geçiren bir filtreden geçirilerek yönlü hidrofon işaretleri elde edilir. Demodüle edilmiş yönlü ve yönsüz hidrofon işaretlerinin bir görüntüsü Şekil 5.17'de verilmiştir. Şekildeki x ve y yönlü hidrofon işaretleri bölüm 5.2.2.1'de değinildiği üzere ideal durumdadır. Buna göre x ve y yönlü hidrofon işaretleri aynı fazda elde edildiğinden dolayı sıfır geçişleri aynı zaman aralıklarında olmaktadır. Böylece manyetik kuzeye olan kerteriz değeri doğru olarak bulunabilir.



Şekil 5.16: QAM demodülasyonu blok diyagramı



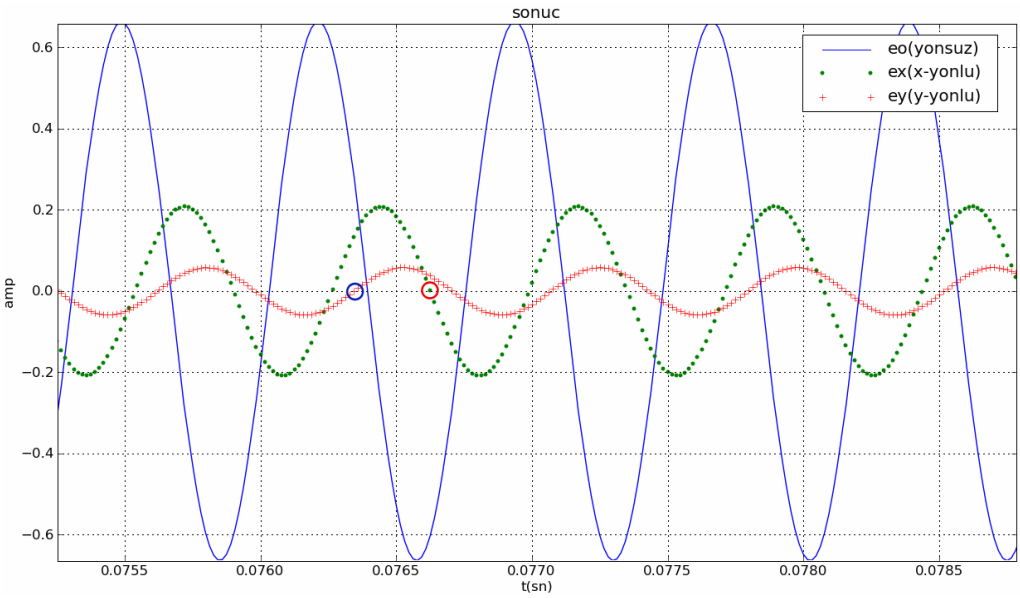
Şekil 5.17: Demodüle edilmiş yönsüz ve yönlü hidrofon işaretleri

5.2.2. Yönlü ve yönsüz hidrofon işaretlerinden hedefin manyetik kuzeye olan kerterizinin hesaplanması

Bu bölümde, Bölüm 5.2.1'de elde edilmiş yönlü ve yönsüz hidrofon işaretlerinden hedefin manyetik kuzeye olan kerterizinin hesaplanması incelenecektir. Bu işlem iki aşamadan oluşur. Buna göre ilk olarak hesaplanacak kerteriz değerinin hatasını aşırı saptırabilecek örnekler atılacaktır, ikinci aşamada ise seçilen örneklerden hedefin manyetik kuzeye olan kerterizi bulunacaktır.

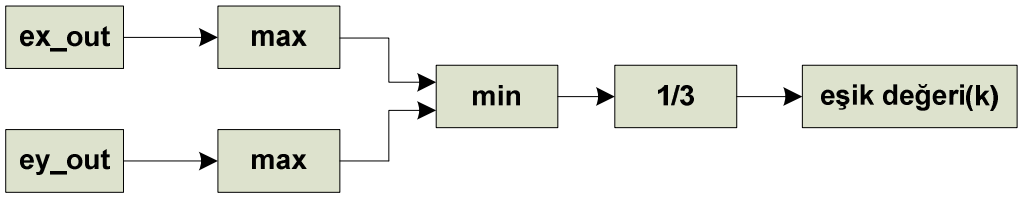
5.2.2.1. Yönlü hidrofon işaretlerinden uygun olmayan örneklerin atılması

İdeal bir durumda x ve y yönlü hidrofon işaretlerinin aynı fazda veya biribirlerine 180° faz farklı olması gerekir. Buna örnek olarak Şekil 5.17'i verebiliriz. Bu şekilde görüldüğü gibi x ve y hidrofonları aynı fazdadır. fakat iyi olmayan bir FM iletimi sırasında temelbant işaretinde meydana gelebilecek olan bozulmalar sonucu yönlü hidrofonlar arasında rastgele faz farkları oluşur, bunun en büyük nedeni frekans ve faz pilotlarında meydana gelen faz ve frekans kaymalandır. Böyle durumlarda yönlü hidrofon işaretleri Şekil 5.18'deki gibi olur.

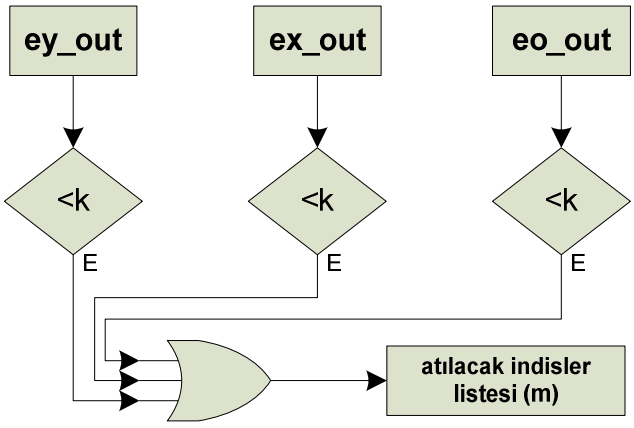


Şekil 5.18: Yönlü hidrofon işaretlerindeki arasındaki faz kayması

Daire içinde gösterilen x ve y hidrofon işaretlerinin sıfır geçişleri eş zamanlı olmadığından dolayı atan2 [24] işleminde anormal değerler oluşur, bu ise kerteriz açısında büyük hatalara sebep olur. Bu durumu önlemek için iki işaretinde sıfır geçişlerine yakın olan örnekler atılır. Şekil 5.19'da bu işleme ait bir blok diyagram verilmiştir.



Şekil 5.19: (k) eşik değerinin bulunması

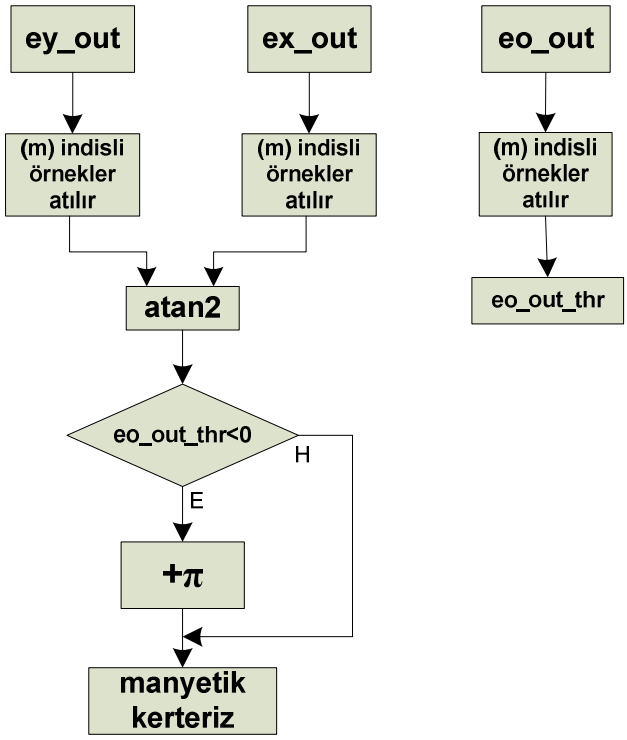


Şekil 5.20: Atılacak örnek indislerinin listesinin çıkartılması

Şekil 5.19'da öncelikle x ve y hidrofon işaretlerinin maksimum aldığı değer tespit edilir, daha sonra bu iki değerin minimum olanının 1/3'ü eşik değeri (k) olarak belirlenir. Bu eşik değerinin altındaki örnek indisleri Şekil 5.20'deki gibi OR'lanarak atılacak indislerin listesi (m) çıkarılır.

5.2.2.2. Seçilen örneklerden hedefin manyetik kuzeye olan kerterizinin hesaplanması

Kerteriz hesabına uygun olmayan örnek indislerin listesi (m) çıkarıldıktan sonra Şekil 5.21'de gösterilen sırada hedefin manyetik kuzeye olan kerterizi hesaplanır, ilk olarak (m) indisli örnekler yönlü ve yönsüz hidrofon işaretlerinden atılır.



Şekil 5.21: Hedefin manyetik kuzeye olan kerterizinin hesaplanması

İkinci olarak eşitlik 3.19'da gösterildiği gibi y hidrofonun x hidrofonuna oranının arctan'ı alınarak γ bulunur. Bu aşamada yönsüz hidrofonun fazı oldukça önemlidir, buna göre yönsüz hidrofonun fazı negatif ise yönlü hidrofonların fazları 180° ters gibi düşünülmelidir, özet olarak eğer yönsüz hidrofonun fazı negatif ise eşitlik

3.19'da bulunan γ değerine 180° faz farkı eklenmelidir. Buna örnek bir senaryo Tablo 5.1'de verilmiştir.

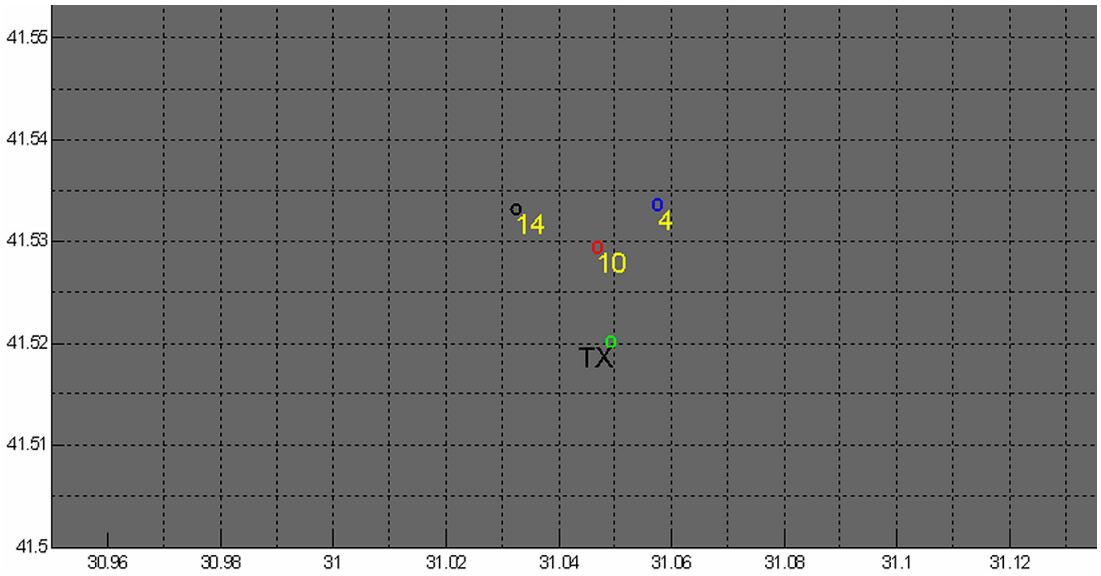
Tablo 5.1: Kerteriz hesaplanmasında örnek bir senaryo

Gerçek kerteriz açısı	**Yönsüz hid. fazı**	**x yönlü hid. fazı**	**y yönlü hid. fazı**	**atan2 sonucu bulunan kerteriz**
30°	+ -	+ -	+ -	30° -150°
150°	+ -	- +	+ -	150° -30°
210°	+ -	- +	- +	-150° (210°) 30°
330°	+ -	+ -	- +	-30° (330°) 150°

6.DENEYSEL SONUÇLAR

Bu bölümde tez kapsamında geliştirilen DIFAR tipi sonoboy alıcısının deniz testlerine ilişkin sonuçlar verilecektir.

31/03/2009 tarihinde Karadeniz Ereğlisi açıklarında icra edilen tatbikatta üç adet DIFAR tipi sonoboy atılmıştır. Sonoboy daldırma noktaları ve manyetik kuzeye olan kerterizi bulunmak istenen sonar daldırma noktası(TX) Şekil 6.1'de verilmiştir.



Şekil 6.1: Tatbikat daldırma noktaları

Şekil 6.1'de görülen sonoboy daldırma noktalarını temsil eden 4,10,14 numaraları seçilen sonoboy kanalını belirtmektedir. Tablo 6.1'de sonoboylara ait parametreler verilmiştir.

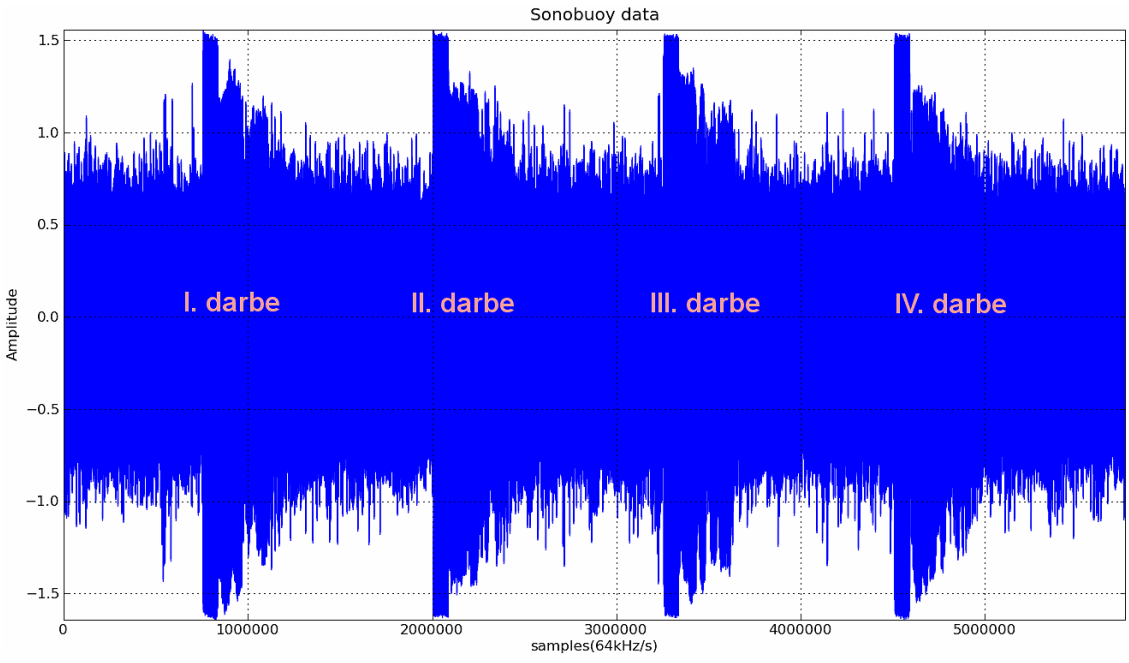
Tablo 6.1: Sonoboy parametreleri

Sonoboy numarası	**Kanal no**	**Haritadan hesaplanan kerteriz değeri**
I.	10	168,9482°
II.	4	205,067°
III.	14	136,1°

Testler sırasında USRP-TVRX kombinasyonu ve 6,5 dBi kazancında çok yönlü (omni) VHF anten kullanılmıştır. Her bir kayıt 30 sn uzunluğunda yapılmıştır. Sonoboyların kerteriz analizinde elde edilen kerteriz hatası değerleri performans ölçütü olarak kabul edilmiştir. Buna göre $\pm$10 hata aralığı en iyi performans, $\pm$20 aralığı ise kabul edilebilir performans olarak değerlendirilmektedir. Testler sırasında I. sonoboy en iyi performansı göstermiştir. II. sonoboy ise ortalama performans göstermiştir. III. sonoboydan ise çok iyi bir netice alınamamıştır. Buna sebep olarak FM iletimindeki SNR düşüklüğünü gösterebiliriz, bunun nedeni atılan sonoboydaki bir arıza veya anten konumundaki bir terslik olabilir. Devam eden başlıklarda iyi netice veren I. sonoboy ağırlıklı olmak üzere alınan kayıt örneklerine ait test sonuçları ve meydana gelen kerteriz hatalarına ait grafikler verilecektir.

6.1. Saat 11:12:30 İtibariyle I. Sonoboy Ölçüm Sonuçları

Toplam dört adet 1380 Hz frekansında sonar darbesi kaydedilen bu ölçümün, zaman eksenindeki görüntüsü Şekil 6.2'de verilmiştir.



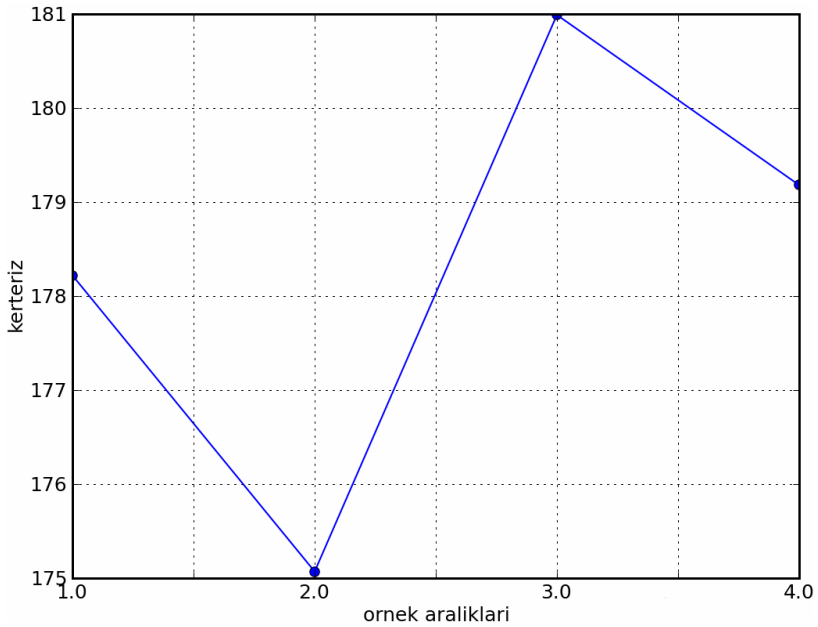
Şekil 6.2: Zaman ekseninde temelbant işaret ve sonar darbelerinin gösterimi

Tablo 6.2'de kerteriz analizi yapılan örnek aralıkları verilmiştir. Şekil 6.3'te darbe aralıklarının hesaplanan kerteriz değerleri grafiksel olarak verilmiştir. Şekil 6.4'te ise

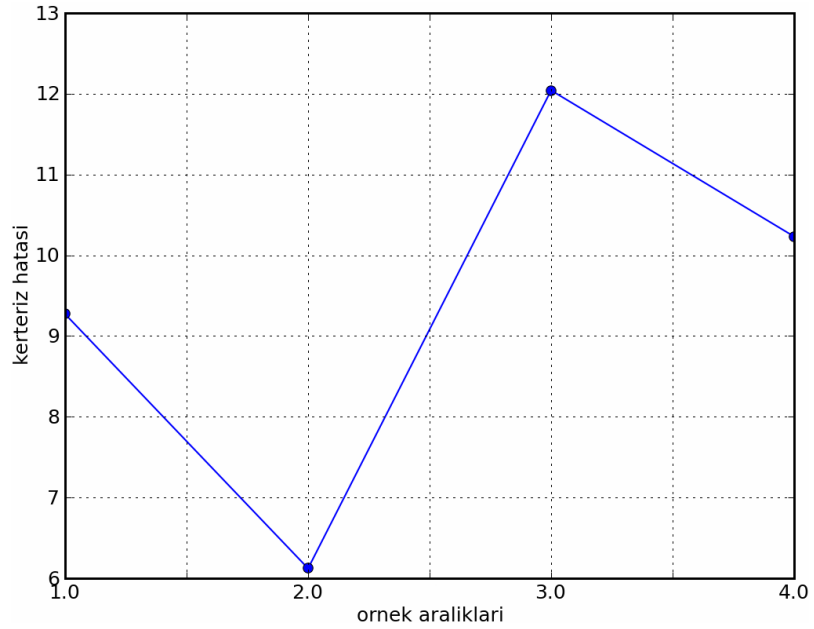
kerteriz analizinde hesaplanan kerteriz hatası değerleri grafiksel olarak verilmiştir.

Tablo 6.2: Analiz edilen darbe örnek aralıkları (11:12:30)

Darbe sırası	**Analiz edilen darbe örnekleri aralığı**
I	790000-800000
II	2030000-2040000
III	3290000-3300000
IV	4545000-4555000

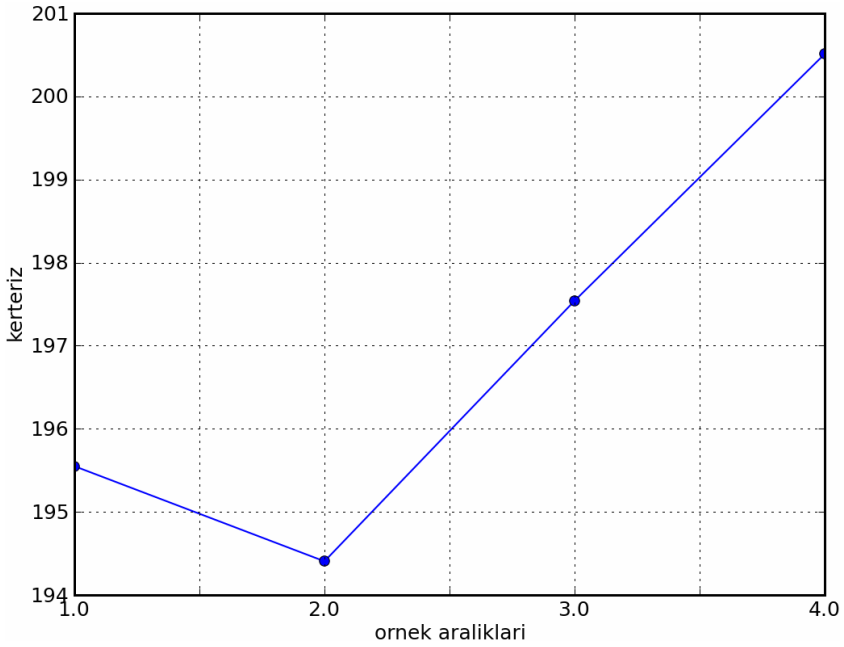


Şekil 6.3: Hesaplanan kerteriz değerleri grafiği(11:12:30)

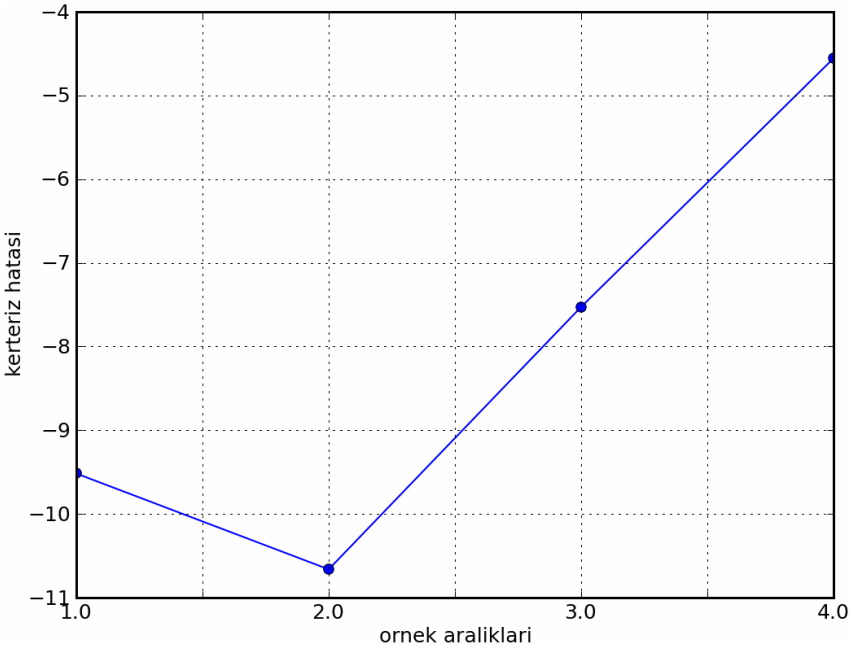


Şekil 6.4: Kerteriz hataları grafiği(11:12:30)

6.2. Saat 11:16:31 İtibariyle II. Sonoboy Ölçüm Sonuçları



Şekil 6.5: Hesaplanan kerteriz değerleri grafiği (11:16:31)

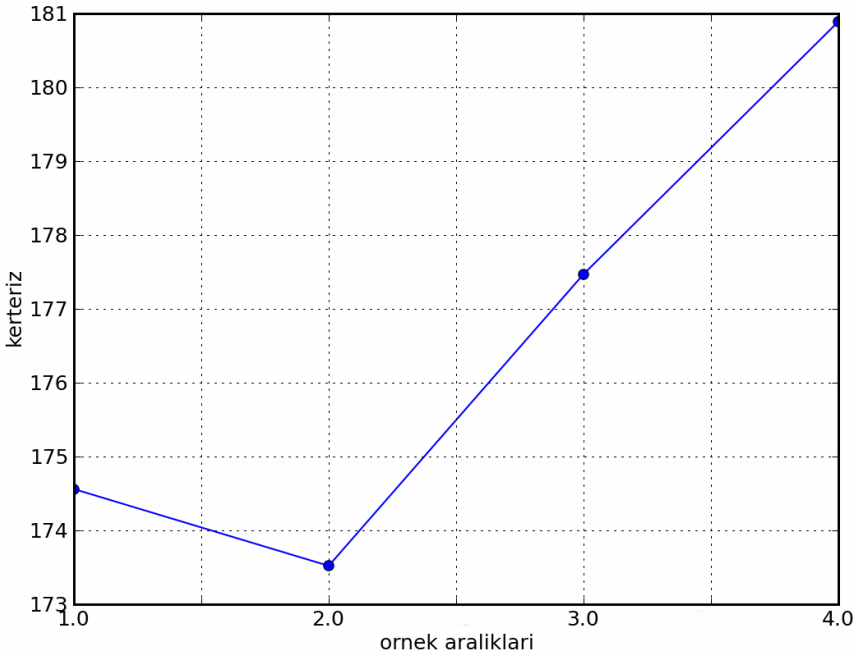


Şekil 6.6: Kerteriz hataları grafiği (11:16:31)

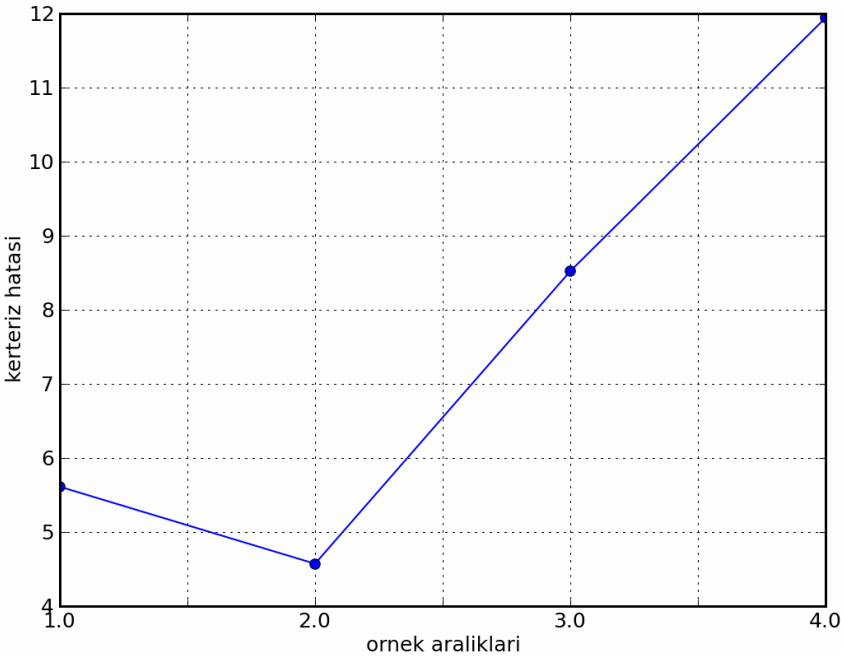
Tablo 6.3: Analiz edilen darbe örnek aralıkları (11:16:31)

Darbe sırası	**Analiz edilen darbe örnekleri aralığı**
I	480000-490000
II	2200000-2210000
III	3410000-3420000
IV	4660000-4670000

6.3. Saat 11:20:46 İtibariyle I. Sonoboy Ölçüm Sonuçları



Şekil 6.7: Hesaplanan kerteriz değerleri grafiği (11:20:46)

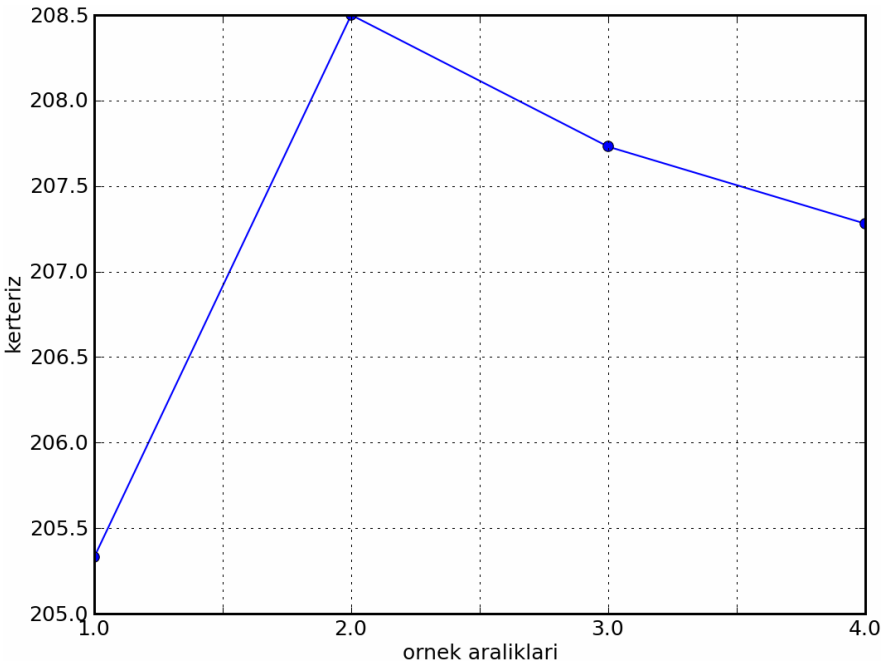


Şekil 6.8: Kerteriz hataları grafiği (11:20:46)

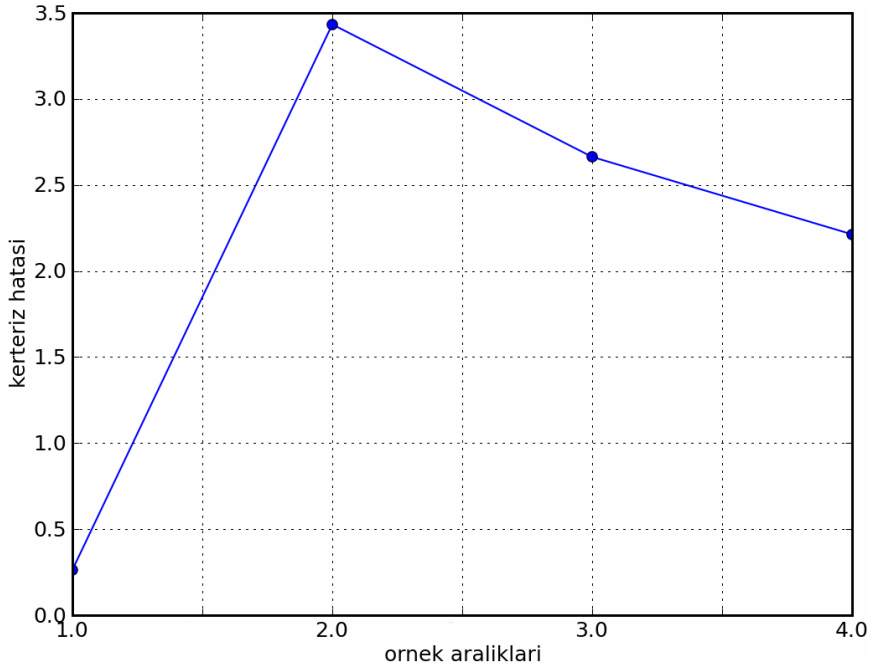
Tablo 6.4: Analiz edilen darbe örnek aralıkları (11:20:46)

Darbe sırası	**Analiz edilen darbe örnekleri aralığı**
I	770000-780000
II	2040000-2050000
III	3290000-3300000
IV	5110000-5120000

6.4. Saat 11:23:06 İtibariyle II. Sonoboy Ölçüm Sonuçları



Şekil 6.9: Hesaplanan kerteriz değerleri grafiği (11:23:06)

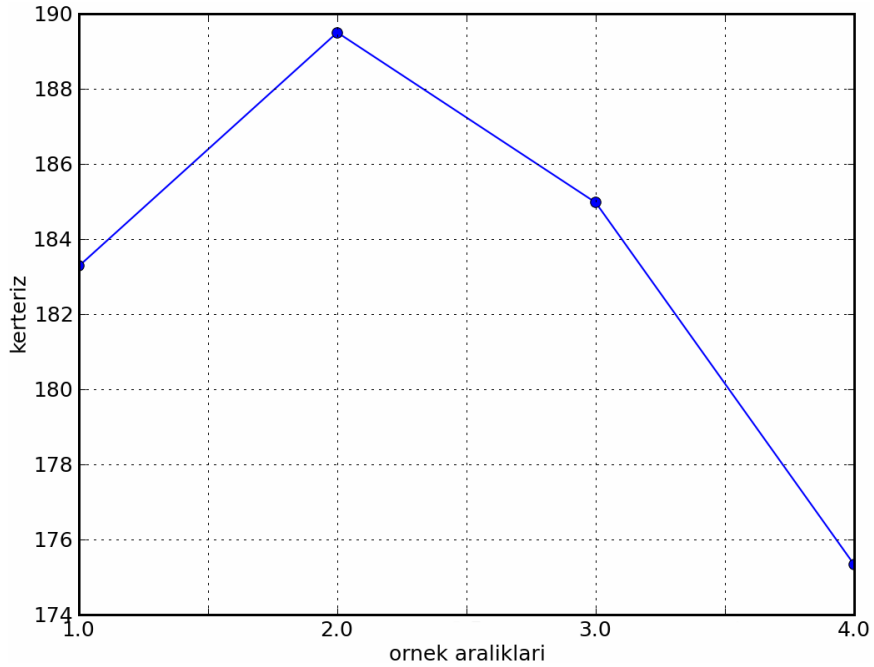


Şekil 6.10: Kerteriz hataları grafiği (11:23:06)

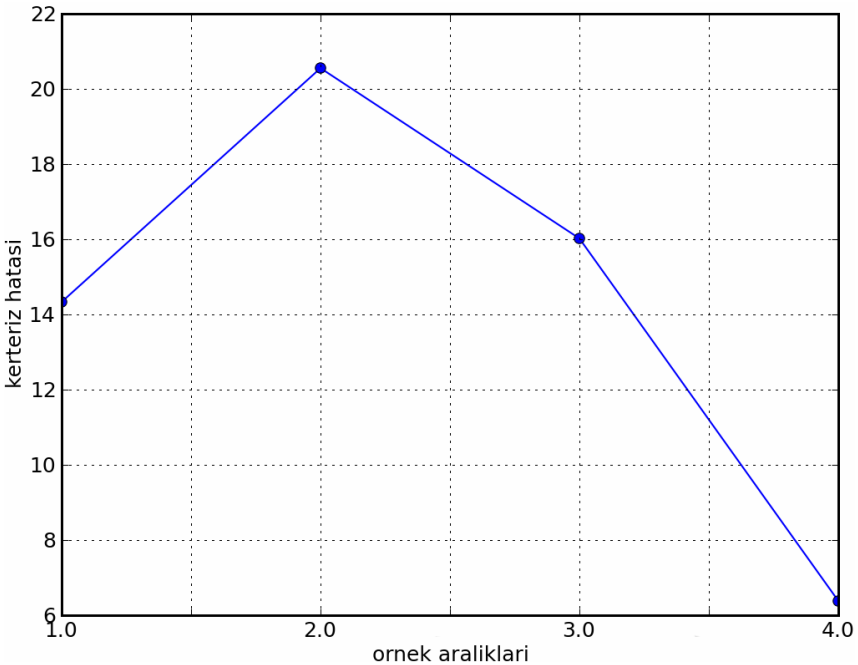
Tablo 6.5: Analiz edilen darbe örnek aralıkları (11:23:06)

Darbe sırası	**Analiz edilen darbe örnekleri aralığı**
I	1220000-1230000
II	2470000-2480000
III	3720000-3730000
IV	4960000-4970000

6.5. Saat 11:30:46 İtibariyle I. Sonoboy Ölçüm Sonuçları



Şekil 6.11: Hesaplanan kerteriz değerleri grafiği (11:30:46)

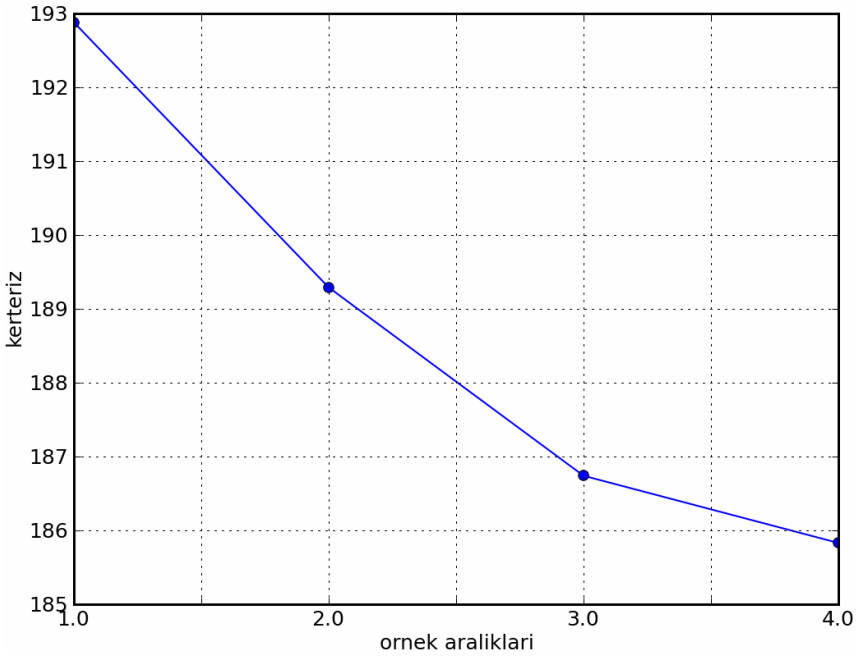


Şekil 6.12: Kerteriz hataları grafiği (11:30:46)

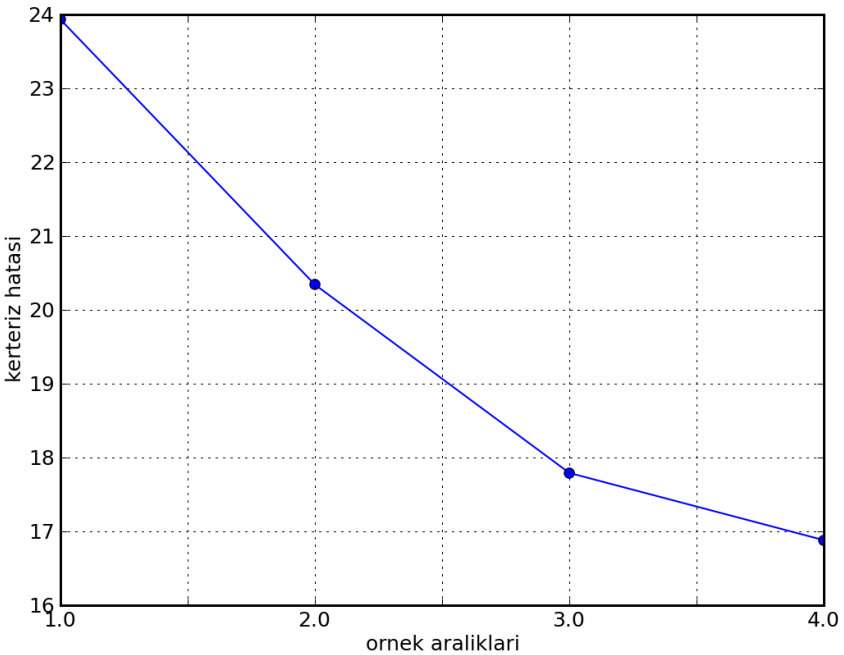
Tablo 6.6: Analiz edilen darbe örnek aralıkları (11:30:46)

Darbe sırası	**Analiz edilen darbe örnekleri aralığı**
I	930000-940000
II	2190000-2200000
III	3440000-3450000
IV	5665000-5675000

6.6. Saat 12:09:23 İtibariyle I. Sonoboy Ölçüm Sonuçları



Şekil 6.13: Hesaplanan kerteriz değerleri grafiği (12:09:23)

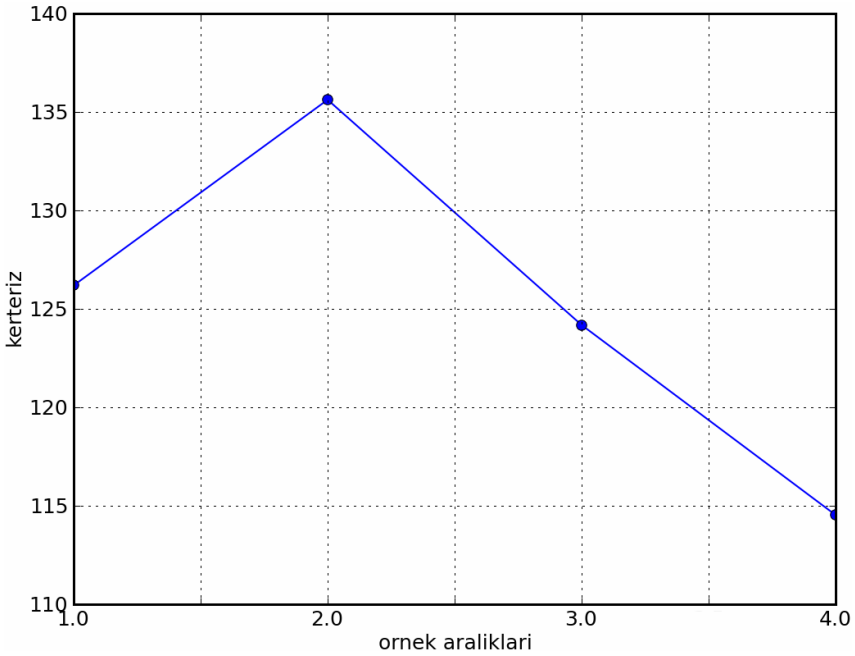


Şekil 6.14: Kerteriz hataları grafiği (12:09:23)

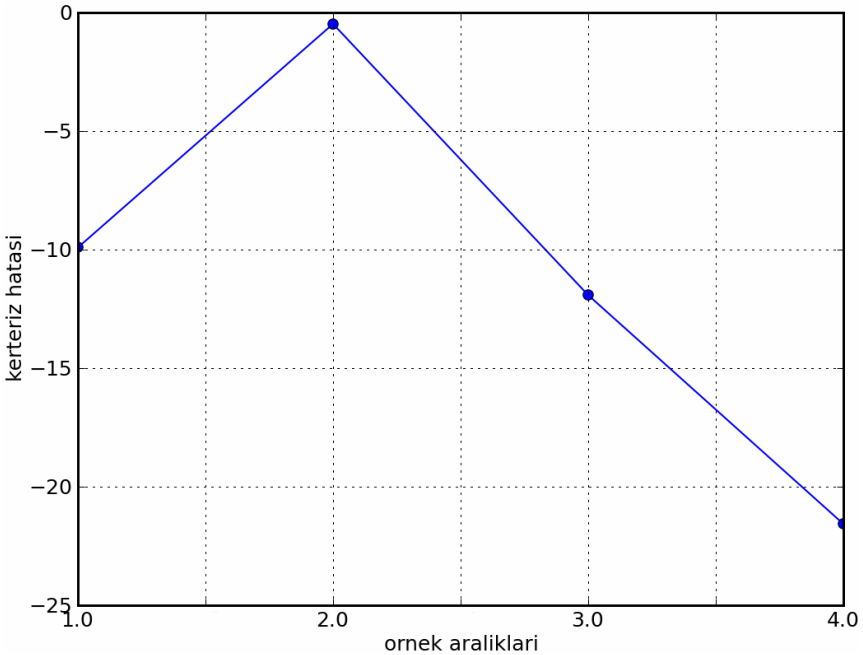
Tablo 6.7: Analiz edilen darbe örnek aralıkları (12:09:23)

Darbe sırası	**Analiz edilen darbe örnekleri aralığı**
I	860000-870000
II	2100000-2110000
III	3355000-3365000
IV	4610000-4620000

6.7. Saat 12:15:46 İtibariyle III. Sonoboy Ölçüm Sonuçları



Şekil 6.15: Hesaplanan kerteriz değerleri grafiği (12:15:46)

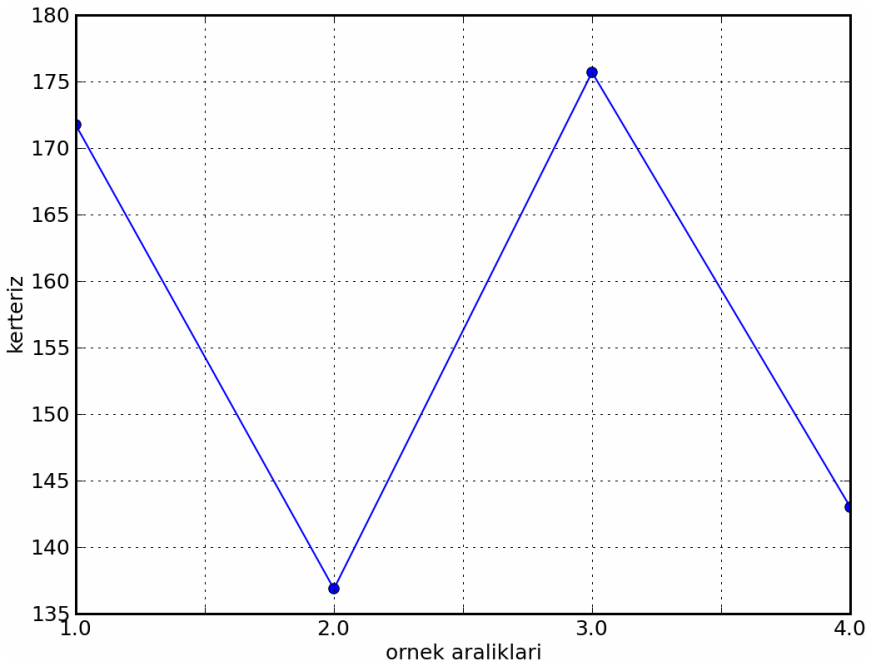


Şekil 6.16: Kerteriz hataları grafiği (12:15:46)

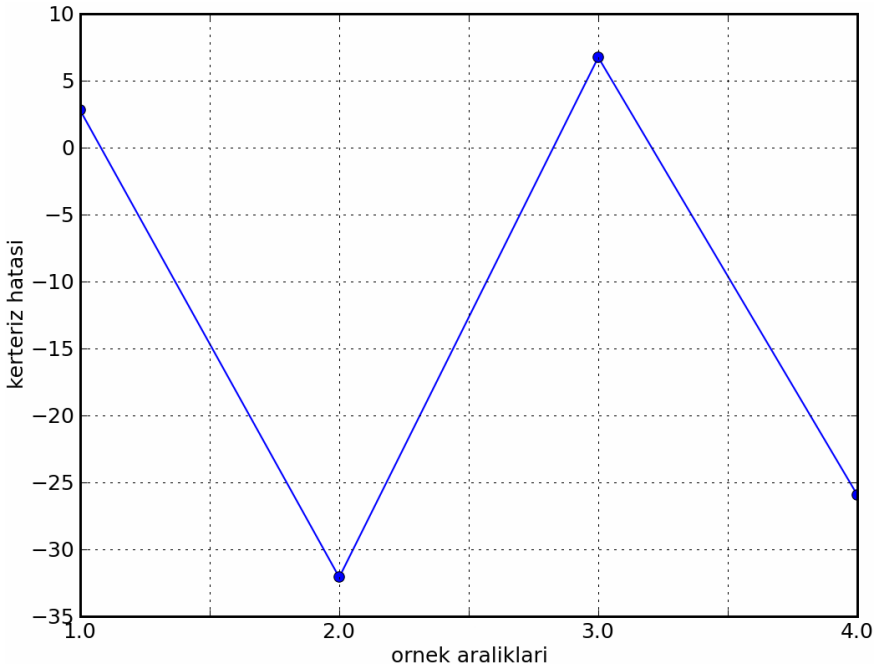
Tablo 6.8: Analiz edilen darbe örnek aralıkları (12:15:46)

Darbe sırası	**Analiz edilen darbe örnekleri aralığı**
I	1810000-1820000
II	3040000-3050000
III	4290000-4300000
IV	5540000-5550000

6.8. Saat 12:25:32 İtibariyle I. Sonoboy Ölçüm Sonuçları



Şekil 6.17: Hesaplanan kerteriz değerleri grafiği (12:25:32)

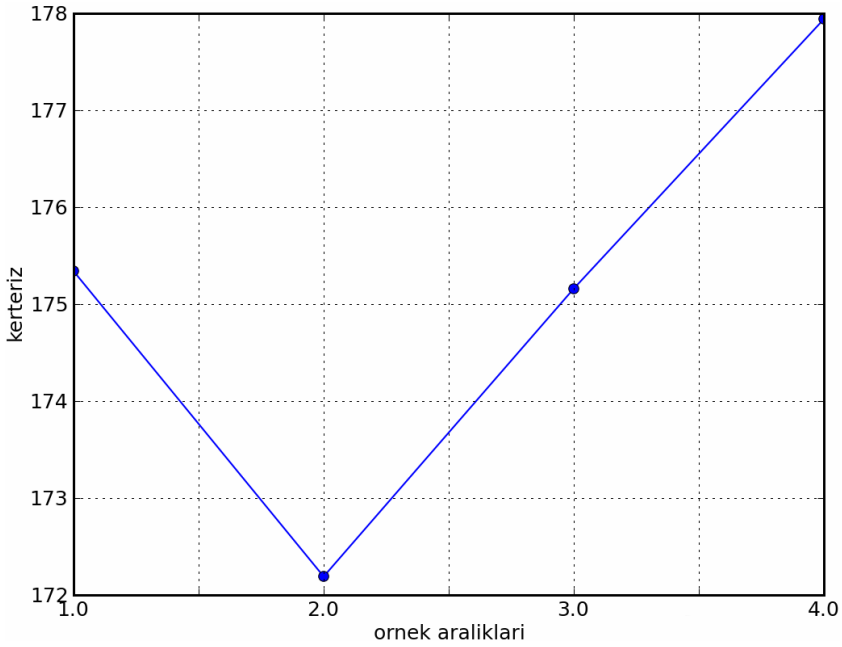


Şekil 6.18: Kerteriz hataları grafiği (12:25:32)

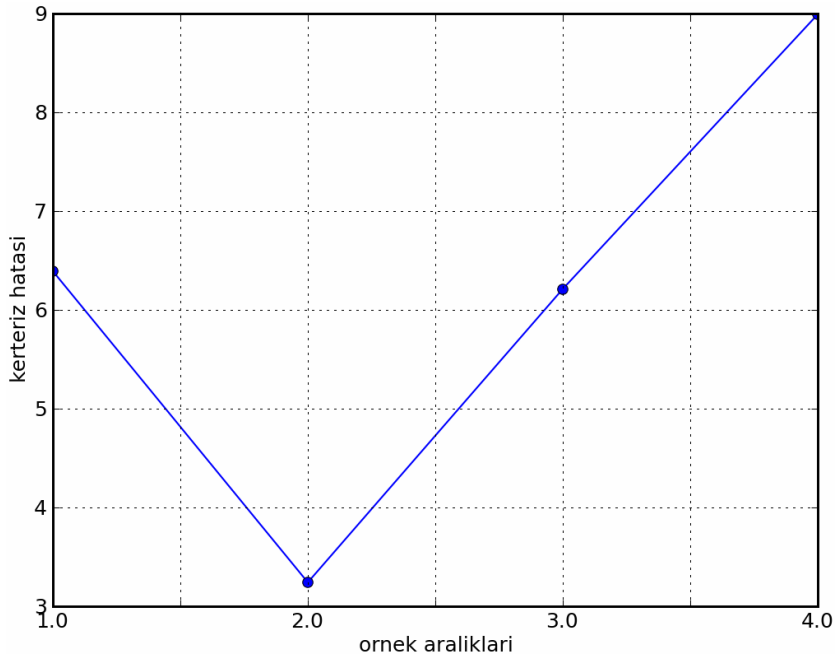
Tablo 6.9: Analiz edilen darbe örnek aralıkları (12:25:32)

Darbe sırası	**Analiz edilen darbe örnekleri aralığı**
I	200000-210000
II	1420000-1430000
III	3515000-3525000
IV	4730000-4740000

6.9. Saat 12:31:46 İtibariyle I. Sonoboy Ölçüm Sonuçları



Şekil 6.19: Hesaplanan kerteriz değerleri grafiği (12:31:46)

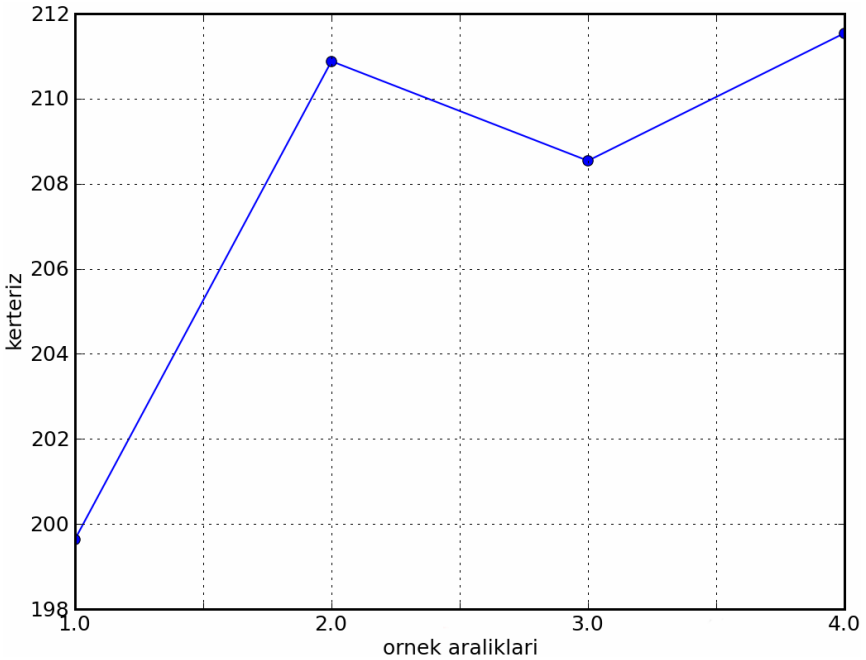


Şekil 6.20: Kerteriz hataları grafiği (12:31:46)

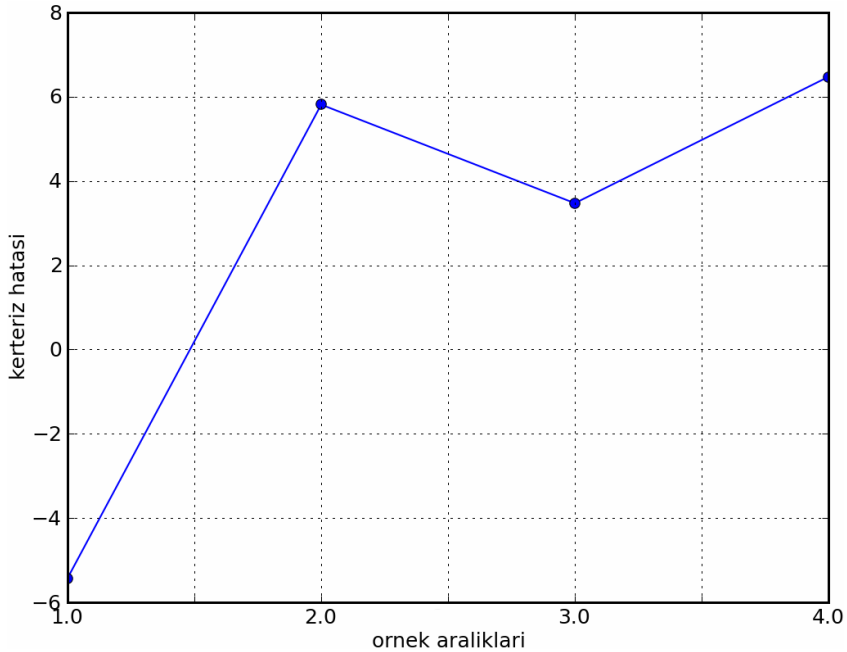
Tablo 6.10: Analiz edilen darbe örnek aralıkları (12:31:46)

Darbe sırası	**Analiz edilen darbe örnekleri aralığı**
I	250000-260000
II	1500000-1510000
III	2750000-2760000
IV	4000000-4010000

6.10. Saat 12:35:46 İtibariyle II. Sonoboy Ölçüm Sonuçları



Şekil 6.21: Hesaplanan kerteriz değerleri grafiği (12:35:46)



Şekil 6.22: Kerteriz hataları grafiği (12:35:46)

Tablo 6.11: Analiz edilen darbe örnek aralıkları (12:35:46)

Darbe sırası	**Analiz edilen darbe örnekleri aralığı**
I	660000-670000
II	1910000-1920000
III	4280000-4290000
IV	5530000-5540000

7. SONUÇLAR VE ÖNERİLER

Bu çalışmada, SDR ile DIFAR tipi sonoboy alıcı geliştirilmiştir. Radyo iletişim sistemlerinin popüler bir konusu olan SDR sayesinde demodülasyon ve kerteriz analizi sayısal olarak gerçeklenmiştir. SDR, radyo sinyallerini işlemek için dirençler, kondansatörler, geri besleme devreleri kullanmak yerine yazılım modülleri kullanılarak radyo fonksiyonlarının önemli bir kısmını yerine getiren bir radyo teknolojisidir. Geliştirilen yazılım ile yazılımın esnekliği ve sayısal işaret işlemenin hassaslığı kombine edilerek kerteriz hatası azaltılmış ve daha hassas kerteriz hesaplanması mümkün kılınmıştır.

Yazılımın geliştirilmesi sürecinde kullanılan GNU Radio, Python, Scipy, Matplotlib gibi açık kaynak kodlu yazılımlar konusunda önemli bir deneyim kazanılmıştır. Bu sayede gelecek SDR uygulamalarında bu araçların kullanılmasının mümkün olduğu gösterilmiştir.

Gerçek deniz ortamında yapılan sistem testleri sonucu özellikle I numaralı sonoboy'dan iyi netice alınmıştır. Buna göre hesaplanan kerteriz hatalarının %80'i kabul edilebilir performans sınırları içerisinde kalmıştır. Alınan sonuçlar, prototip aşamasında olan alıcının ileride profesyonel olarak kullanılması için ümit vermektedir.

Sistemin hassasiyetinin arttırılması ve aynı anda birden çok sonoboyun analiz edilebilmesi amacıyla projede kullanılan TVRX genişleme kartı yerine IF bant genişliği daha geniş ve giriş hassasiyeti daha yüksek yeni bir RF genişleme kartı geliştirilebilir.

KAYNAKLAR

[1] Tuttlebee, W., "Software Defined Radio: Enabling Technologies", First edition, ***John Wiley & Sons Ltd***, 3-22, (2002).

[2] Tuttlebee, W., "Software Defined Radio: Origins, Drivers and International Perspectives", First edition, ***John Wiley & Sons Ltd***, 3-18, (2002).

[3] Bard, J., Kovarik Jr., V. J., " Software defined radio : the software communications architecture", ***John Wiley & Sons Ltd***, 3-19, (2007).

[4] Wikipedia, 2009, Software Communications Architecture[online], Wikipedia-The Free Encyclopedia, http://en.wikipedia.org/wiki/Software_Communications_Archi tecture **(Ziyaret tarihi: 20 Mart 2009)**.

[5] Wikipedia, 2009, GNU Radio [online], Wikipedia-The Free Encyclopedia, http://en.wikipedia.org/wiki/Gnu_radio **(Ziyaret tarihi: 10 Mart 2009)**.

[6] Cooley, J., E., "A Day in the Life of the RF Spectrum", ***Massachusetts Institute of Technology*****,** Massachusetts**,** 30-35, (2005)

[7] Wikipedia, 2009, Sonobuoy [online], Wikipedia-The Free Encyclopedia, http://en.wikipedia.org/wiki/Sonobuoy **(Ziyaret tarihi: 15 Mart 2009)**.

[8] Holler, R., A., Horbach, A., W., McEachern, J., F., "The Ears of Air ASW - A History of U.S. Navy Sonobuoys", First edition, ***Navmar Applied Sciences Corporation***, 13-93, (2008)

[9] "AN/SSQ-53B user manual", ***Sparton Electronics***, 1-20, (1984)

[10] "AR35 Dual Axis Fluxgate Magnetometer Wıth Floating Core", ***Autonnic Research Ltd***, (2003-2006)

[11] Proakis, J., G., Salehi, M., "Communication Systems Engineering", Second Edition, ***Prentice-Hall Inc,*** 96-100, (2002)

[12] Verburgt, P., W., "NATO AIR-ASW Buoys Interoperability Specification Volume I", ***Naval Air Warfare Center Aircraft Division***, (I-1-I-15),(2-6-2-9), (2006)

[13] Rossum, G. v., Drake, F., L., "Python Tutorial", Release 2.0, ***BeOpen PythonLabs***, 7-66, (2000).

[14] Tonguç, Y., 2008, Eğlenceli bir programlama dili: Python [online], ileriseviye.org, http://www.ileriseviye.org/arasayfa.php?inode=python101.html **(Ziyaret tarihi: 20 Mart 2009)**.
[15] Schach, S., R., "Object-Oriented and Classical Software Engineering", 5th edition, ***McGraw-Hill***, 17-21, (2002).

[16] "SciPy Reference Guide", Release 0.7, ***SciPy Community***, (3-9)-(80-120), (2008).

[17] Dale, D., Droettboom, M., Firing, E., Hunter, J., "Matplotlib Reference Guide", "Release 0.98.5.1", ***Matplotlib Community***, (1-119)-(519-540), (2008).

[18] Manicka, N., "Gnu Radio Testbed", Yüksek Lisans Tezi, ***University of Delaware,*** Delaware, 25, (2007).

[19] Blossom, E., How to Write GNU Radio Python Applications [online], gnuradio.org , http://gnuradio.org/trac/wiki/Tutorials/WritePythonApplications **(Ziyaret tarihi: 12 Mart 2009)**.

[20] Muller, A., "DAB Software Receiver Implementation", Yüksek Lisans Tezi, ***Eidgenössische Technische Hochschule(ETH)***, Zurich, 15-25, (2008).

[21] Volkwin, A., "Suitability of a Commercial Software Defined Radio System for Passive Coherent Location", Yüksek Lisans Tezi, ***University of Cape Town Department of Electrical Engineering***, Cape Town, 41-56, (2008).

[23] Muller, J., M., "Elementary Functions Algorithms and Implementation", ***Birkhäuser***, 101-125, (1961).

[23] Proakis, J., G., Dimitris, G., M., "Digital Signal Processing", Third edition, ***Prentice-Hall Inc***, 908-920, (1996).

[24] Rossum, G. v., Drake, F., L., "Python Library Reference", Release 2.0, ***BeOpen PythonLabs***, 81-84, (2000).

EKLER

EK-A

Bu bölümde konu 5.1'de işlenilmiş olan yazılım tanımlı radyo ile gerçeklenen sonoboy alıcıya ait olan yazılım kodları verilmiştir.

Tablo A1: Yazılım tanımlı radyo ile gerçeklenen sonoboy alıcı kodları

```
    #!/usr/bin/env python

from gnuradio import gr,gru, eng_notation, optfir
from gnuradio import usrp
from gnuradio import blks2
from gnuradio.eng_option import eng_option
from usrpm import usrp_dbid
import sys
import math
from numpy import array
from scipy import *
from gnuradio.wxgui import stdgui2, fftsink2, form
import wx
from time import strftime,localtime
from optparse import OptionParser
import test

def pick_subdevice(u):
    """
    #
    #
    #
    #
    @return a subdev_spec
    """
    return usrp.pick_subdev(u, (usrp_dbid.TV_RX,
                                usrp_dbid.TV_RX_REV_2,
                                usrp_dbid.TV_RX_REV_3,
                                usrp_dbid.BASIC_RX))

class wfm_rx_block (stdgui2.std_top_block):
    def __init__(self,frame,panel,vbox,argv):
        stdgui2.std_top_block.__init__ (self,frame,panel,vbox,argv)

        parser=OptionParser(option_class=eng_option)
        parser.add_option("-f", "--fft_size", type="eng_float",
default=1024,
                          help="set fft size")
        parser.add_option("-p", "--buffer_path", type="string",
default="/home/engink/SPE/",
                          help="recording directory")

        (options, args) = parser.parse_args()
        if len(args) != 0:
            parser.print_help()
            sys.exit(1)
```

Tablo A1:(Devam) Yazılım tanımlı radyo ile gerçeklenen sonoboy alıcı kodları

```
    self.frame = frame
        self.panel = panel
        self.state = "FREQ"
        self.freq = 136e6
        self.time = 10
        self.chan = 0
        self.recording =0
        self.nsamples = 0
        self.mode = "Receive"
        self.file_name = "R"
        self.tx_freq =1380
        self.freq_offset=0
        self.bearing = 0
        self.power = True

        self.freq_table= arange(0,99)
        self.freq_table=
[162.250,163.000,163.750,164.500,165.250,166.000,166.750,167.500,168.250,
169.000,169.750,170.500,171.250,172.000,172.750,173.500,162.625,163.375,
164.125,164.875,165.625,166.375,167.125,167.875,168.625,169.375,170.125,

170.875,171.625,172.375,173.125,136.000,136.375,136.750,137.125,137.500,
137.875,138.250,138.625,139.000,139.375,139.750,140.125,140.500,141.875,
141.250,141.625,142.000,142.375,142.750,143.125,143.500,143.875,144.250,
144.625,145.000,145.375,145.750,146.125,146.500,146.875,147.250,148.625,
148.000,148.375,148.750,149.125,149.500,149.875,150.250,150.625,151.000,
151.375,151.750,152.125,152.500,152.875,153.250,153.625,154.000,154.375,
154.750,155.125,155.500,155.875,156.250,156.625,157.000,157.375,157.750,
158.125,158.500,158.875,159.250,159.625,160.000,160.375,160.750,161.125]
        # akış diyagramı oluşturuluyor

        self.u = usrp.source_c()                    # usrp kaynak

        adc_rate = self.u.adc_rate()                # 64 MS/s
        usrp_decim = 250
        self.u.set_decim_rate(usrp_decim)
        usrp_rate = adc_rate / usrp_decim           # 256 kS/s
        chanfilt_decim = 4
        self.demod_rate = usrp_rate / chanfilt_decim
        self.rx_subdev_spec = pick_subdevice(self.u)

self.u.set_mux(usrp.determine_rx_mux_value(self.u,self.rx_subdev_spec))
        self.subdev = usrp.selected_subdev(self.u, self.rx_subdev_spec)
        print "Using RX d'board %s" % (self.subdev.side_and_name(),)

        fm_demod_gain = usrp_rate/(2*math.pi*75e3)
        self.fm_demod = gr.quadrature_demod_cf (fm_demod_gain)

        chan_filt_coeffs = gr.firdes.low_pass (1,      # filtre kazancı
                                               usrp_rate, # örnekleme hızı
                                               22e3,      #kesim frekansÄ±
                                               4e3,       #zayıflatmaya
basladıgı frekans
                                               gr.firdes.WIN_HAMMING)
        self.chan_filt = gr.fir_filter_fff (chanfilt_decim,
chan_filt_coeffs)

        self.connect (self.u, self.fm_demod, self.chan_filt)
        self._build_gui(vbox, usrp_rate, self.demod_rate, options)
        g = self.subdev.gain_range()
        self.gain = float(g[0]+g[1])/2

        self.freq = 1e6
        self.u.tune(0, self.subdev, self.freq)
        self.set_gain(self.gain)
```

Tablo A1:(Devam) Yazılım tanımlı radyo ile gerçeklenen sonoboy alıcı kodları

```
    def _set_status_msg(self, msg, which=0):
        self.frame.GetStatusBar().SetStatusText(msg, which)

    # GUI nesneleri oluşturuluyor
    def _build_gui(self, vbox, usrp_rate, demod_rate, options):

        def _form_set_freq(kv):
            return self.set_freq(kv['freq'])

        if 1:
            self.src_fft = fftsink2.fft_sink_c(self.panel, title="FM",

fft_size=int(options.fft_size), sample_rate=usrp_rate,
                               ref_scale=32768.0, ref_level=0, y_divs=12)
            self.connect (self.u, self.src_fft)
            vbox.Add (self.src_fft.win, 4, wx.EXPAND)

        if 1:
            post_filt_fft = fftsink2.fft_sink_f(self.panel, title="DIFAR
Baseband", fft_size= int(options.fft_size), sample_rate=self.demod_rate,
y_per_div=10, ref_level=0, average=True)
        self.connect (self.chan_filt, post_filt_fft)
        vbox.Add (post_filt_fft.win, 4, wx.EXPAND)

        self.myform = myform = form.form()

        hbox = wx.BoxSizer(wx.HORIZONTAL)
        hbox.Add((5,0), 0)

        button_on =
form.button_with_callback(parent=self.panel,label="Power on",
callback=self.power_on)
        hbox.Add(button_on,0,wx.EXPAND)
        hbox.Add((5,0), 0)
        button_off =
orm.button_with_callback(parent=self.panel,label="Power off",
callback=self.power_off)
        hbox.Add(button_off,0,wx.EXPAND)
        hbox.Add((5,0), 0)
        vbox.Add(hbox, 0, wx.EXPAND)
                hbox = wx.BoxSizer(wx.HORIZONTAL)
        hbox.Add((5,0), 0)
        myform['freq'] = form.float_field(parent=self.panel, sizer=hbox,
label="Channel",value=32, weight=1,
callback=myform.check_input_and_call(_form_set_freq,
self._set_status_msg))

        hbox.Add((10,0), 0)
        myform['freq_slider'] =
form.quantized_slider_field(parent=self.panel, sizer=hbox,value=32,
weight=3, #range=(87.9e6, 108.1e6, 0.1e6),
           range=(1, 99, 1),
           callback=self.set_freq)
        hbox.Add((5,0), 0)
        vbox.Add(hbox, 0, wx.EXPAND)

        hbox = wx.BoxSizer(wx.HORIZONTAL)
        hbox.Add((5,0), 0)

        myform['fine_tune'] = \
form.quantized_slider_field(parent=self.panel,  sizer=hbox,value=0,
label="Fine",
                                        weight=3, range=(-20, 40, 1),
                                        callback=self.set_fine)
        hbox.Add((5,0), 0)
```

Tablo A1:(Devam) Yazılım tanımlı radyo ile gerçeklenen sonoboy alıcı kodları

```
    myform['gain'] = \
form.quantized_slider_field(parent=self.panel, sizer=hbox, label="Gain",
                                weight=3, range=self.subdev.gain_range(),
                                                    callback=self.set_gain)
        hbox.Add((5,0), 0)
        vbox.Add(hbox, 0, wx.EXPAND)

        hbox = wx.BoxSizer(wx.HORIZONTAL)
        hbox.Add((5,0), 0)

        myform['record_time'] = form.float_field(
        parent=self.panel, sizer=hbox, label="\nRec length",value=60,
weight=1,callback=myform.check_input_and_call(self.set_length))
        hbox.Add((5,0), 0)
        button1 =
form.button_with_callback(parent=self.panel,label="Receive|Record",
callback=self.record_samples)
        hbox.Add(button1,0,wx.EXPAND)
        hbox.Add((5,0), 0)
        button2 =
form.button_with_callback(parent=self.panel,label="Bearing calc.",
callback=self.bearing_calc)
        hbox.Add(button2,0,wx.EXPAND)
        hbox.Add((20,0), 0)
        myform['bearing'] = form.float_field(
        parent=self.panel, sizer=hbox, label="\nBearing",value=0,
weight=1)
        hbox.Add((5,0), 0)
        vbox.Add(hbox, 0, 0)
        hbox = wx.BoxSizer(wx.HORIZONTAL)
        hbox.Add((5,0), 0)

        myform['file_name'] = form.text_field(
            parent=self.panel, sizer=hbox, label="\nFile name ", weight=1,
            callback=myform.check_input_and_call(self.set_file_name))
        hbox.Add((5,0), 0)

        myform['tx_freq'] = form.float_field(
        parent=self.panel, sizer=hbox, label="\nTX freq(Hz)",value=1380,
weight=1, callback=myform.check_input_and_call(self.set_tx))
        hbox.Add((5,0), 0)
        vbox.Add(hbox, 0, wx.EXPAND)

    def power_on (self):
        if self.power == False:
            self.wait()
            self.start()
            self.mode = "receive"
            self.update_status_bar ()
            self.power = True

    def power_off (self):
        if self.power == True:
            self.stop()
            self.mode = "offline"
            self.update_status_bar ()
            self.power = False

    def set_fine (self, fine):
        khz = fine*1000
        self.freq_offset = khz
        fix_freq = self.freq + khz
        usrp.tune(self.u, 0, self.subdev, fix_freq)
        self.update_status_bar()
        print fix_freq
```

Tablo A1:(Devam) Yazılım tanımlı radyo ile gerçeklenen sonoboy alıcı kodları

```
    def set_freq(self, target_freq):
        self.chan = target_freq
        temp = int(target_freq-1)
        temp_mhz = (self.freq_table[temp])*1e6
        self.freq = temp_mhz
        print temp_mhz
        r = usrp.tune(self.u, 0, self.subdev, temp_mhz)
        if r:
            self.freq = temp_mhz
            self.myform['freq'].set_value(target_freq)
            self.myform['freq_slider'].set_value(target_freq)
            self.update_status_bar()
            self._set_status_msg("OK", 0)
            return True
        self._set_status_msg("Failed", 0)
        return False

    def set_gain(self, gain):
        self.myform['gain'].set_value(gain)
        self.subdev.set_gain(gain)

    def set_length(self,length):
        self.time=length['record_time']
        print self.time

    def set_file_name(self,filename):
        self.file_name = filename['file_name']
        print self.file_name

    def set_tx(self,txfreq):
        self.tx_freq = txfreq['tx_freq']
        print self.tx_freq

    def bearing_calc(self):
        self.bearing =
test.difar_analysis(self.file_name,"/home/engink/SPE/",self.tx_freq,
self.demod_rate)
        self.myform['bearing'].set_value(self.bearing)

    def update_status_bar (self):
        msg = "Freq_offset(kHz):%r  Setting:%s  Mode:%s" %
(int(self.freq_offset/1000), self.state, self.mode)
        self._set_status_msg(msg, 1)
        self.src_fft.set_baseband_freq(self.freq+self.freq_offset)

    def record_samples (self):
        if  self.recording:
            self.recording =0
            self.stop_record_samples()
        else:
            self.recording =1
            self.start_record_samples()

    def start_record_samples (self):
        samples = int(self.time)*64000
        time_rec = strftime("%d-%m-%Y|%H:%M:%S", localtime())
        self.stop()
        self._head = gr.head(gr.sizeof_float, int(samples))
        self.file_name = time_rec +"-ch"+str(int(self.chan))+".dat"
        self.myform['file_name'].set_value(self.file_name)
        self.utfil = gr.file_sink(gr.sizeof_float,self.file_name)
        self.connect(self.chan_filt,self._head,self.utfil)
        self.wait()
        self.start()
        self.mode = "Record"
```

Tablo A1:(Devam) Yazılım tanımlı radyo ile gerçeklenen sonoboy alıcı kodları

```
        self.update_status_bar ()
        self.recording = True

    def stop_record_samples (self):

        self.stop()

        self.disconnect(self.chan_filt,self._head,self.utfil)
        self.wait()
        self.start()
        self.mode = "Receive"
        self.update_status_bar ()
        self.recording = False

if __name__ == '__main__':
    app = stdgui2.stdapp (wfm_rx_block, "VHF sonobuoy receiver")
    app.MainLoop ()
```

ÖZGEÇMİŞ

1981 yılında İzmir-Bornova' da doğdu. İlk, orta ve lise öğrenimini Bornova' da tamamladı. 1999 yılında girdiği Erciyes Üniversitesi Mühendislik Fakültesi Elektronik Mühendisliği Bölümü'nden 2005 yılında Elektronik Mühendisi olarak mezun oldu. 2005 yılından itibaren TÜBİTAK MAM Bilişim Teknolojileri Enstitüsü'nde araştırmacı olarak görev yapmaktadır.