



**KABLOSUZ SENSÖR AĞLARI İÇİN RSSİ DEĞERİ İLE
YAPAY SİNİR AĞI YAKLAŞIMLI KONUM BULMA**

Bil. Müh. Resul DOĞAN

Yüksek Lisans Tezi

Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Ebubekir ERDEM

AĞUSTOS-2018

T.C
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

KABLOSUZ SENSÖR AĞLARI İÇİN RSSI DEĞERİ İLE YAPAY SİNİR AĞI YAKLAŞIMLI
KONUM BULMA

YÜKSEK LİSANS TEZİ

Resul DOĞAN

121129102

Tezin Enstitüye Verildiği Tarih : 17 Temmuz 2018

Tezin Savunulduğu Tarih : 2 Ağustos 2018

Tez Danışmanı : Dr. Öğr. Üyesi Ebubekir ERDEM (F.Ü.)

Diğer Jüri Üyeleri : Doç. Dr. Muhammed Fatih TALU (İ.Ü.)

Doç. Dr. Taner TUNCER (F.Ü.)

Ağustos-2018

ÖNSÖZ

Bu tez çalışmasında gerek yönlendirmeleriyle gerekse desteğiyle sürekli yanımda olan, sayın danışman hocam Dr. Öğr. Üyesi Ebubekir ERDEM'e, yüksek lisans yapmama vesile olan sayın hocam Prof. Dr. Ahmet Bedri ÖZER'e ve vermiş oldukları değerli derslerle, bu tez çalışmamı bitirmemde yardımcı olan başta Prof. Dr. Yetkin TATAR'a, Doç.Dr. Taner TUNCER'e ve Doç. Dr. Burhan ERGEN hocalarıma teşekkürü bir borç bilirim. Ayrıca Fırat Üniversitesi Bilimsel Araştırma Projeleri Yönetim Birimine (FÜBAB) MF.16.05 Projesi dâhilinde yüksek lisans tezime yaptığı katkılardan dolayı teşekkür ederim.

Resul DOĞAN
ELAZIĞ – 2018

İÇİNDEKİLER

Sayfa No

ÖNSÖZ	II
İÇİNDEKİLER.....	III
ÖZET	V
SUMMARY	VI
ŞEKİLLER LİSTESİ	VII
TABLolar LİSTESİ	IX
KISALTMALAR.....	X
1. GİRİŞ.....	1
1.1. Kablosuz Sensör Ağ Mimarileri Kullanılarak Yapılan Çalışmalar	1
2. KABLOSUZ SENSÖR AĞLAR	4
2.1. Genel Tanım	4
2.2. Kablosuz Algılayıcı Ağlarının Uygulama Alanları	5
2.2.1. Askeri Uygulamalar.....	5
2.2.2. Çevre Uygulamaları.....	5
2.2.3. Sağlık Uygulamaları	6
2.2.4. Ev Uygulamaları.....	6
2.2.5. Ticari Uygulamalar.....	6
2.3. Kablosuz Sensör Ağlarının Sıkıntıları	7
2.4. Kablosuz Sensör Ağlarda Kullanılan Sensör Düğümlerinin Yapısı	7
2.5. Memsic TELOSB TPR2420CA Sensör Düğümü.....	8
2.6. Kablosuz Sensör Ağlar İçin Gömülü İşletim Sistemleri.....	9
2.6.1. TINYOS	10
2.7. XBee RF Haberleşme Modülleri	10
2.7.1. XBee AT ve API Çalışma Modları	12
2.7.2. XBee XCTU Konfigürasyon Yazılımı	12
3. KABLOSUZ SENSÖR AĞLARDA KONUM BULMA YÖNTEMLERİ.....	14
3.1. Genel Bilgi	14
3.2. Mesafe Bağımlı Yöntemler (Range Based – Distance Based)	15

3.2.1.	Time of Arrival (TOA)	16
3.2.2.	Time Difference of Arrival (TDOA)	16
3.2.3.	RSSI.....	16
3.2.4.	Angel of Arrival (AOA)	17
3.3.	Mesafe Bağımsız Yöntemler (Range Free – Distance Free)	17
3.3.1.	DV-Hop	17
3.3.2.	APIT	18
3.3.3.	Centroid	18
3.3.4.	Bounding Box Yöntemi.....	18
4.	YAPAY SİNİR AĞLARI.....	20
4.1.	Genel Bilgi	20
4.2.	Matlab İle Yapay Sinir Ağları Kullanımı	21
5.	SİSTEM TASARIMI VE UYGULAMASI	23
5.1.	Genel Açıklama	23
5.2.	Xbee Modüllerin Parametre Ayarlarının Yapılması	24
5.3.	Lineer Hareket Eden Araç Tasarımı ve Programlanması	26
5.4.	Üç Adet Köşe Düğümün Hazırlanıp Programlanması.....	32
5.5.	4. Köşe Düğümün (Base Station) Hazırlanıp Bu Düğüm İle Bilgisayar Haberleşmesini Sağlayan Yazılımın Yazılması.....	33
5.6.	Deney Ortamının Seçilmesi ve Uygulamanın Gerçekleştirilmesi	34
5.7.	RSSI - Mesafe Kalibrasyonu ve Konum Bulma	36
6.	SONUÇ.....	41
	REFERANSLAR	42
	EKLER	44
	ÖZGEÇMİŞ	59

ÖZET

Kablosuz sensör ağlarda RSSI ile mesafe kalibrasyonu ve konum bulma üzerine birçok yöntem, algoritma ve çalışma mevcuttur. Bu tez çalışmasında RSSI ile mesafe kalibrasyonu ve konum bulmaya, yapay sinir ağları kullanarak farklı bir bakış açısı getirilmeye çalışılmıştır. Bu amaçla öncelikle kablosuz sensör düğüm ihtiva eden hareketli bir araç tasarlanmıştır. Bu araç 4 bir köşesinde kablosuz sensör düğüm bulunan ve her birimi 60 cm olan 16x14'lük bir deney ortamında, 60 cm'lik ilerlemelerle köşe düğümlere olan uzaklık – RSSI değerleri ve aracın x, y koordinatları bilgisayar ortamında kayıt altına alınmıştır. Bu işlemler her noktada her düğüm için, gürültüden etkilenen ölçümlerin sonuca etkisini azaltmak için, 100 kez tekrarlanmış, sonuç olarak toplamda 4x100x(16x14) adet RSSI mesafe ve konum bilgisi kayıt altına alınmıştır. Bu veriler sonraki aşamada, MATLAB ortamında tasarlanan yapay sinir ağlarında eğitim ve test verisi olarak kullanılmış ve ağırlıkları hesaplanan yapay sinir ağları fonksiyonları elde edilmiştir. Aynı şekilde yine MATLAB ortamında Bounding Box algoritması için bir fonksiyon yazılmış, hem Bounding Box algoritmasına göre hem de elde edilen YSA'lerden elde edilen sonuçlar karşılaştırılmıştır.

Uygulamanın gerçekleştirilmesinde Arduino IDE, MATLAB, Netbeans IDE yazılım gerçekleştirme ortamları yanında, uygulama sonucunda elde edilen verilerin saklanması amacıyla MySql veri tabanı kullanılmıştır. Kablosuz sensör düğümler Arduino ve XBee modüller kullanılarak oluşturulmuş, XBee modüllerin konfigürasyonu, XCTU arayüzü kullanılarak USB üzerinden yapılmıştır.

Anahtar Kelimeler: Yapay Sinir Ağları, YSA, Kablosuz Sensör Ağlar, WSN, MATLAB, Arduino, Java, XBee, Zigbee

SUMMARY

An Artificial Neural Network Approach For Wireless Sensor Networks Localization With RSSI Value

A great deal of method, algorithm and study on distance calibration and positioning through RSSI in wireless sensor networks are available. In this dissertation, it is intended to provide a different perspective using artificial neural networks to distance calibration and positioning through RSSI. To that end, a mobile tool carrying wireless sensor node has been designed. In an experiment environment where this tool has a wireless sensor node at its each four corners and each unit equals to 60 cm at 16x14, the distance to corner nodes with 60 cm advancement – RSSI values and x, y coordinates of the tool have been recorded in the computer. In order to mitigate the effects of measurements affected by the noise on the results, these processes have been repeated 100 times for each node at each point and consequently, a total of 4x100x(16x14) RSSI distance and positioning information have been recorded. In the next stage, these data have been used as training and test data in artificial neural networks designed in MATLAB environment and the functions of artificial neural networks, whose weight was counted, have been obtained. Similarly again in MATLAB environment, a function was coded for Bounding Box algorithm and both results obtained from Bounding Box algorithm and ANN's have been compared.

While realizing the application, in addition to Arduino IDE, MATLAB, Netbeans IDE software realization environments, MySql database has been used in order to store the data obtained from the application. Wireless sensor nodes have been produced using Arduino and XBee modules and the configuration of XBee modules has been made through USB using XCTU interface.

Keywords: Artificial Neural Networks, ANN, Wireless Sensor Networks, WSN, MATLAB, Arduino, Java, XBee, Zigbee

ŞEKİLLER LİSTESİ

Sayfa No

Şekil 2.1. Kablosuz Sensör Ağların Genel Mimarisi [4].....	4
Şekil 2.2. MacroMote Minyatür Sensörü	5
Şekil 2.3. Ev Kontrol Otomasyonu	6
Şekil 2.4. Endüstriyel Otomasyon Sistemi.....	7
Şekil 2.5. Sensör Düğümün Yapısı	8
Şekil 2.6. TelosB TPR2420CA Blok Diyagramı [13].....	9
Şekil 2.7. TelosB TPR2420CA Genel Görünümü [13].....	9
Şekil 2.8. TinyOS İşletim Sistemi Mimarisi [14].....	10
Şekil 2.9. XBee Modül Genel Görünümü	11
Şekil 2.10. Arduino ve XBee Bağlantı Şeması	12
Şekil 2.11. XCTU Yazılımın Genel Görünümü	13
Şekil 3.1. Temel Konum Bulma Yöntemleri.....	14
Şekil 3.2. Konum Bulma Yöntemlerinin Düğüm Sayısına Göre Maliyet Grafiği [16].....	15
Şekil 3.3. Konum Bulma Yöntemlerinin Düğüm Sayısına Göre Enerji Tüketimleri Grafiği [16]	15
Şekil 3.4. XBee ve CC2420 Yongasının RSSI - Mesafe Grafiği [18]	17
Şekil 3.5. APIT Yöntemi İle Konum Bulma	18
Şekil 3.6. Bounding Box Algoritması ile Konum Bulma [19].....	19
Şekil 4.1. İnsan Sinir Hücresi Yapısı [21].....	20
Şekil 4.2. Yapay Sinir Ağı Genel Topolojisi [22]	21
Şekil 4.3. Matlab Neural Fitting Aracı	22
Şekil 5.1. Uygulamanın Genel Algoritması	24
Şekil 5.2. XCTU Yazılımın Genel Görünümü	25
Şekil 5.3. AP ve AO Konfigürasyon Ayarları.....	25
Şekil 5.4. meOrion Kontrol Kartı.....	26
Şekil 5.5. Step Motor Sürücü Kartı	27
Şekil 5.6. mOrion Kontrol Kartı Genel Görünümü ve Pin Yapısı	27
Şekil 5.7. Çalışmada Kullanılan Stepper Motor.....	28

Şekil 5.8. Kontrol Kartı, Step Motor Sürücü Kartı ile Step Motor Bağlantı Şeması	29
Şekil 5.9. Aracımızın Genel Görünümü	29
Şekil 5.10. Aracımızın Kontrol Akış Diyagramı	31
Şekil 5.11. Aracımızdan Adresi Belirli Bir Düğüme Gönderilen Veri Paketi Çerçeve Yapısı	31
Şekil 5.12. Arduino Uno ve XBee Modül Bağlantısı	32
Şekil 5.13. Köşe Düğümün Genel Görünümü	33
Şekil 5.14. Çalışma Ortamı ve Köşe Düğümlerin Konumlarının Grafikselsel Görünümü	34
Şekil 5.15. Uygulamanın Gerçekleştirildiği Odaanın Genel Görünümü	35
Şekil 5.16. Mesafe Kalibrasyonu Amaçlı Oluşturulan Yapay Sinir Ağımızın Genel Görünümü	36
Şekil 5.17. Mesafe Kalibrasyonu Amaçlı Oluşturulan Yapay Sinir Ağımızın Genel Görünümü	37
Şekil 5.18. Matlab Ortamında Konum Bulma Komutumuz	38
Şekil 5.19. Yapay Sinir Ağı İle Bulunan Konumların Hata Oranlarını Bulan Fonksiyonun Çağırılması ve Sonucu	39
Şekil 5.20. Bounding Box algortiması ile Konum Bulan Fonksiyonun Çağırılması	39
Şekil 5.21. Bounding Box Algoritması İle Bulunan Konumların Hata Oranlarını Bulan Fonksiyonun Çağırılması ve Sonucu	39

TABLÖLAR LİSTESİ

Sayfa No

Tablo 5.1. Örnek XBee Veri Paketi	35
Tablo 5.2. Bulunan Tahmini Koordinat Tablosu	38
Tablo 5.3. Yapay Sinir Ağı ve Bounding Box Yöntemleri İle Elde Edilen Sonuçlar	40



KISALTMALAR

WSN	: Wireless Sensor Network
YSA	: Yapay Sinir Ağları
ANN	: Artificial Neural Networks
APIT	: Approximate Point in Triangulation
GPS	: Global Positioning System
TOA	: Time of Arrival
TDOA	: Time Difference of Arrival
MEMS	: Micro-Electro-Mechanical Systems
RSSI	: Received Signal Strength Indicator
LQI	: Link Quality Indicator

1. GİRİŞ

Birçok alanda uygulanabilir olması, araştırmaya ve geliştirmeye uygun yapısıyla son zamanlarda gündemde oldukça yer tutan Kablosuz Sensör Ağlar çevresindeki fiziksel büyüklükleri sensörler ile algılayıp birbirleri ile kablosuz olarak haberleşebilen ve sahip oldukları kısıtlı kaynakları (pil, hafıza vb.) en verimli şekilde kullanmayı amaçlayan yüzlerce hatta binlerce düğümden oluşan ağlardır. Teknolojinin düşük güç tüketen ve maliyeti düşük kablosuz sensör düğümlerin üretimine olanak sağlamasıyla kablosuz sensör ağlar klasik kablolu sensör sistemlerinin yerine geçmeye başlamıştır. Kablosuz sensör ağlar çevresel sıcaklık, nem kontrolü, askeri ortam izleme, düşman tespiti endüstriyel sensör değerlerinin takibi gibi değişik birçok uygulama alanına sahip bir teknolojidir. Kablosuz sensör ağlar 21. yüzyılın en önemli teknolojilerinden biri olarak tanımlanmaktadır [1].

Kablosuz sensör ağların temelini oluşturan kablosuz sensör düğümler 4 temel unsurdan oluşur. Bunlar algılama ünitesi, işlem ünitesi, haberleşme ve güç ünitesidir. Güç tüketimi sensör düğümlerin tasarımını kısıtladığından dikkate alınması gereken bir konudur. Çoğu durumda haberleşme devreleri enerjinin en çok tüketildiği en temel bölümdür [2,3].

Wi-Fi, Bluetooth, 3G, 4G gibi kablosuz haberleşme teknolojileri büyük boyutlu verilerin taşınmasına olanak sağlarlar ancak bununla birlikte enerji tüketimleri ve sistem gereksinimleri de fazladır. Kablosuz sensör ağlarda kaynakların yetersiz olması bu teknolojileri kullanmaya olanak sağlamamaktadır. IEEE 802.15.4 standardı ile sistem kaynaklarını en verimli kullanmayı amaçlar ve kablosuz sensör ağlarında en çok kullanılan ZigBee protokolünün temelini oluşturur.

Kablosuz sensör ağlarda enerji kaynaklarının kısıtlı olmasından dolayı konum bulmaya yardımcı donanımlar kablosuz sensör düğümlerde genellikle bulunmamaktadır. Ayrıca kablosuz sensör ağların genellikle dağıtık mimaride olması ağdaki herhangi bir düğümün konumunun hesaplanabilmesi önemli bir araştırma konusudur.

1.1. Kablosuz Sensör Ağ Mimarileri Kullanılarak Yapılan Çalışmalar

Kablosuz sensör ağlarla ilgili askeri, çevre, sağlık ve ticari birçok uygulamanın yanı sıra akademik olarak da birçok çalışma ve araştırma yapılmış ve yapılmaya devam edilmektedir. Askeri algılama, fiziksel güvenlik, hava trafik kontrol, endüstriyel

otomasyonlar, trafik gözetim, doğal sınır gözetleme, bina ve inşaat gözetleme Kablosuz Sensör Ağların, mevcut ve potansiyel uygulamalarıdır [1].

Şanlıurfa Bozova ilçesinde bulunan Atatürk Barajı gövdesinde bulunan tünellerde sıcaklık ve nemin kablosuz sensör düğümlerden oluşan bir ağla takibi ve klima sisteminin devreye girip çıkma zamanının belirlenmesi amaçlı bir uygulama yapılmıştır [4].

Kablosuz sensör ağlarda veri bütünlüğü, veri güvenliği ve enerji verimliliği konularında, geleneksel yöntemler yerine yapay zekâ algoritmaları kullanılarak iyileştirmeler yapılmıştır [5].

Biyoalgılayıcılar kullanılarak sağlık sektöründe hastaların, sağlık verilerinin uzaktan takibi ve erken uyarı amaçlı kablosuz sensör ağları kullanılmıştır [6].

Gaz, alev ve sıcaklık sensörleri kullanarak, ev içi yangın tespit ve uyarı sistemi kablosuz sensör ağlar kullanılarak gerçekleştirilmiştir [7].

Manyetik sensörler ihtiva eden kablosuz sensör düğümlerden oluşan bir ağ kullanılarak trafikteki araç yoğunluğu tespit edilmiştir [8].

Yeraltı su seviyesi ve boşluk suyu basıncını ölçmeye yarayan piyezometre sensörlerle baraj rezervuarının durumunu takip etmek ve erken uyarı verebilmek için kablosuz sensör ağlar kullanılarak bir çalışma yapılmıştır [9]. Benzer bir çalışma toprak dolgu baraj gövdelerindeki jeofiziksel değişimleri izlemek amaçlı, jeofiziksel sensör düğümler kullanılarak yapılmıştır [10].

1.2. Tezin Amacı ve Kapsamı

Bu tez çalışmasında dağıtık mimariye sahip ve GPS, GNSS gibi konum bulma donanımlarının enerji tüketimi, maliyet gibi konulardan dolayı kullanılamayacağı kablosuz sensör ağlarda, yapay sinir ağı kullanılarak, kablosuz sensör düğümlerin koordinatlarının en doğru şekilde bulunması amaçlanmıştır. Bu amaçla hareketli bir araç uygun donanımlar seçilerek tasarlanmış, kablosuz düğümler (Node), uygun donanımlar seçilerek oluşturulmuş ve tasarlanan yapay sinir ağlarını eğitip test etmek için gerekli olan veri setleri uygulama ortamında bu donanım ve araçlarla elde edilip bilgisayar ortamında kayıt altına alınmıştır.

Bu tez çalışması, kablosuz sensör ağlar (Wireless Sensor Network - WSN) hakkında bilgi verildiği birinci bölüm, kablosuz sensör ağlarda konum bulma yöntemleri ve bu yöntemlere ait algoritmaların açıklandığı ikinci bölüm, yapay sinir ağları ve MATLAB ortamında YSA araçları hakkında bilgilerin bulunduğu üçüncü bölüm, yapılan çalışmanın

yapım ve uygulama aşamalarının açıklandığı dördüncü bölüm ve çalışma sonucu elde edilen sonuçların paylaşıldığı beşinci bölüm olmak üzere beş bölümden oluşmaktadır.

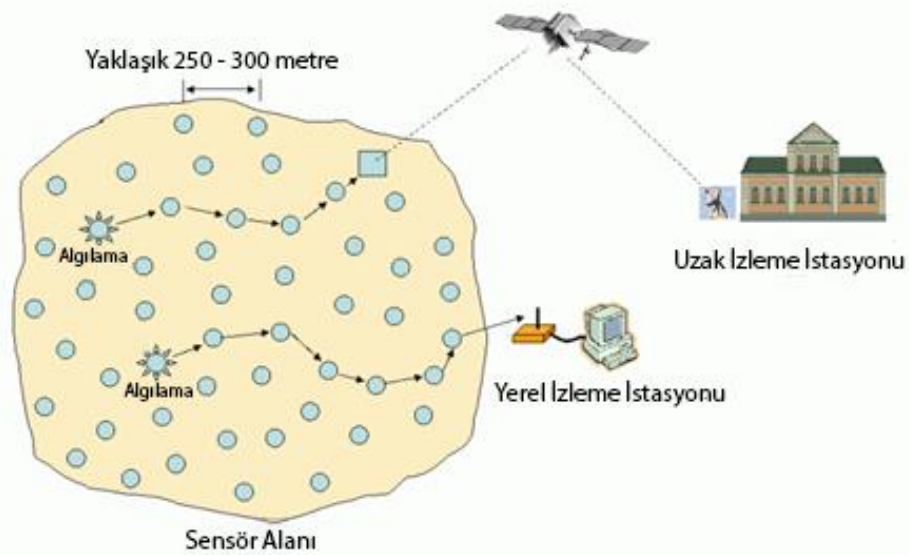
Bu tez çalışması, FÜBAP (Fırat Üniversitesi Bilimsel Araştırma Proje Birimi) tarafından MF.16.05 numaralı proje ile desteklenmiştir.



2. KABLOSUZ SENSÖR AĞLAR

2.1. Genel Tanım

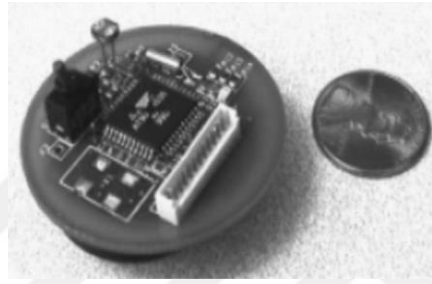
Çevreyi daha iyi algılayabilmek ve analiz edebilmek için düşük güç tüketen, kısıtlı işlemci ve hafızaya sahip kablosuz düğümlerden (node) oluşan büyük ölçekli sensör ağları, yapılan çalışmalar sonucunda araştırmacıların ilgi odağı olmuştur [11]. Askeri güvenlik, fiziksel algılama, hava trafik kontrolü, akıllı bina uygulamaları, endüstriyel otomasyonlar, trafik gözetim ve denetleme ve doğal sınır izleme, kablosuz sensör ağların ihtiva ettiği mevcut ve potansiyel uygulamalardan birkaçıdır [1]. Kablosuz sensör ağlar (wireless sensör network – WSN) kavramı ilk kez 1980'lerin başlarında karşımıza çıkmıştır. Mikro elektromekanik sistemlerdeki (MEMS) gelişmeler ve kablosuz haberleşme sistemlerindeki ilerlemelerle birlikte 1990'lı yıllarda önemli bir araştırma alanı haline gelmeye başlamıştır. İlk zamanlarda askeri alanda kullanılan kablosuz sensör ağları; zamanla maliyetlerin düşmesi ile çok yaygın olarak kullanılmaya başlanmıştır [12]. Şekil 2.1'de Kablosuz sensör ağların genel yapısı görülmektedir.



Şekil 2.1. Kablosuz Sensör Ağların Genel Mimarisi [4]

Birçok alanda uygulanabilir olması, araştırmaya ve geliştirmeye uygun yapısıyla son zamanlarda gündemde oldukça yer tutan kablosuz sensör ağlar, çevresindeki fiziksel büyüklükleri sensör düğümleriyle algılayıp birbirleri ile kablosuz olarak haberleşebilen ve

sahip oldukları kısıtlı kaynakları (pil, hafıza vb.) en verimli şekilde kullanmayı amaçlayan yüzlerce hatta binlerce düğümden oluşan ağlardır. Teknolojinin düşük güç tüketen ve maliyeti düşük kablosuz sensör düğümlerin üretimine olanak sağlamasıyla kablosuz sensör ağlar klasik kablolu sensör sistemlerinin yerine geçmeye başlamıştır. Kablosuz sensör ağlar, çevresel sıcaklık ve nem kontrolü, askeri ortam izleme, düşman tespiti, endüstriyel otomasyon takibi gibi değişik birçok uygulama alanına sahip bir teknolojidir. Kablosuz sensör ağlar 21. yüzyılın en önemli teknolojilerinden biri olarak tanımlanmaktadır [1]. Şekil 2.2’de MacroMote kablosuz sensör düğümü görülmektedir.



Şekil 2.2. MacroMote Minyatür Sensörü

2.2. Kablosuz Algılayıcı Ağlarının Uygulama Alanları

Sensör ağları, nem, sıcaklık, basınç, ses, ışık ve hareketlilik gibi durumsal değişiklikleri takip edebilecek yapıdaki termik, sismik, manyetik ve görsel birçok farklı tipte algılayıcı içerebilir. Bu ağların uygulama alanları askeri, çevre, sağlık, ev ve diğer ticari alanlar olmak üzere sınıflandırılabilir. Kablosuz sensör ağların, sensör çeşitliliğine göre kullanım alanları çok fazladır. Kablosuz sensör ağ uygulamalarının başlıcaları aşağıda listelenmiştir.

2.2.1. Askeri Uygulamalar

- Düşman kuvvetlerini izleme
- Tatbikat veya savaş alanı gözetleme
- Hedefleme
- Muharebe hasar tespiti
- Nükleer, biyolojik ve kimyasal saldırı tespiti

2.2.2. Çevre Uygulamaları

- Mikroklima
- Orman yangınlarının algılanması

- Taşkınların algılanması
- Hassas tarım

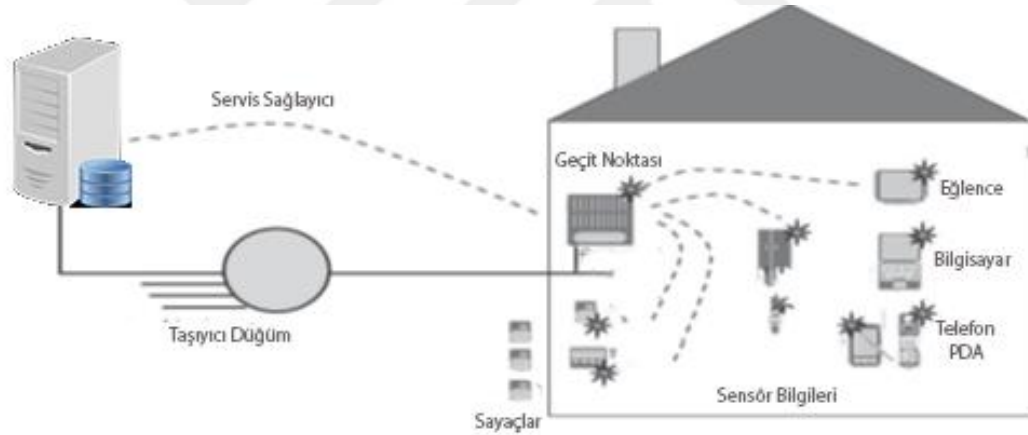
2.2.3. Sağlık Uygulamaları

- Fizyolojik dataların uzaktan takibi
- Doktorların ve hastaların hastane içinde takibi
- İlaç yönetimi
- Yaşlı yardımı

2.2.4. Ev Uygulamaları

- Ev otomasyonu
- Otomatik sayaç okuma

Şekil 2.3’de bir ev kontrol otomasyonunun genel yapısı görülmektedir.

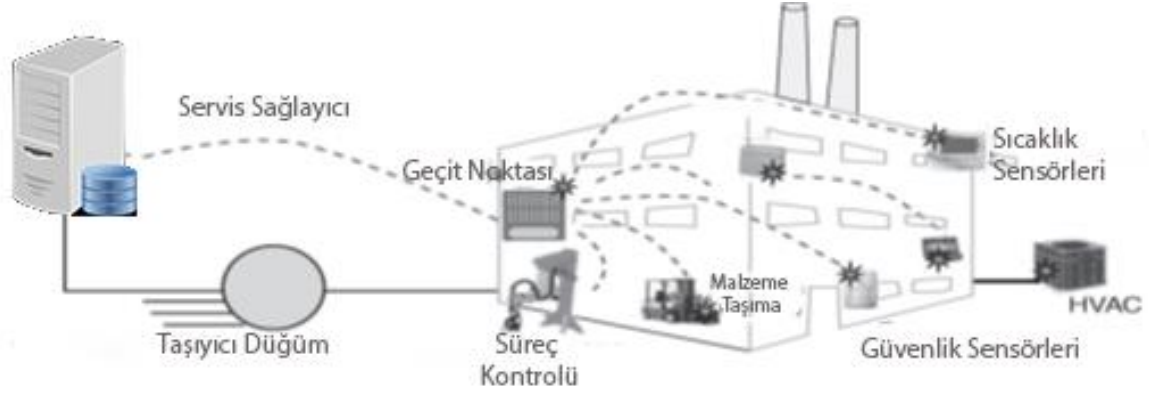


Şekil 2.3. Ev Kontrol Otomasyonu

2.2.5. Ticari Uygulamalar

- Endüstriyel ve ofis binalarının çevresel kontrolü
- Araç takibi ve tespiti
- Trafik akışının gözetimi
- Envanter kontrolü

Şekil 2.4’te kablosuz sensör ağlarla gerçekleştirilmiş endüstriyel kontrol otomasyonunun genel yapısı görülmektedir.



Şekil 2.4. Endüstriyel Otomasyon Sistemi

2.3. Kablosuz Sensör Ağlarının Sıkıntıları

Kablosuz sensör ağlarındaki belli başlı sıkıntılar şunlardır:

- Sensör düğümler yoğun olarak yerleştirildiğinden maliyet artar.
- Sensör düğümler hatalara eğilimlidirler
- Sık sık ağ topolojisi değişir.
- Hafıza ve hesaplama gücünden dolayı sınırlıdır.
- Çok sayıda sensör düğüm olduğundan sensör düğümler global haberleşme yapamazlar.

2.4. Kablosuz Sensör Ağlarda Kullanılan Sensör Düğümlerinin Yapısı

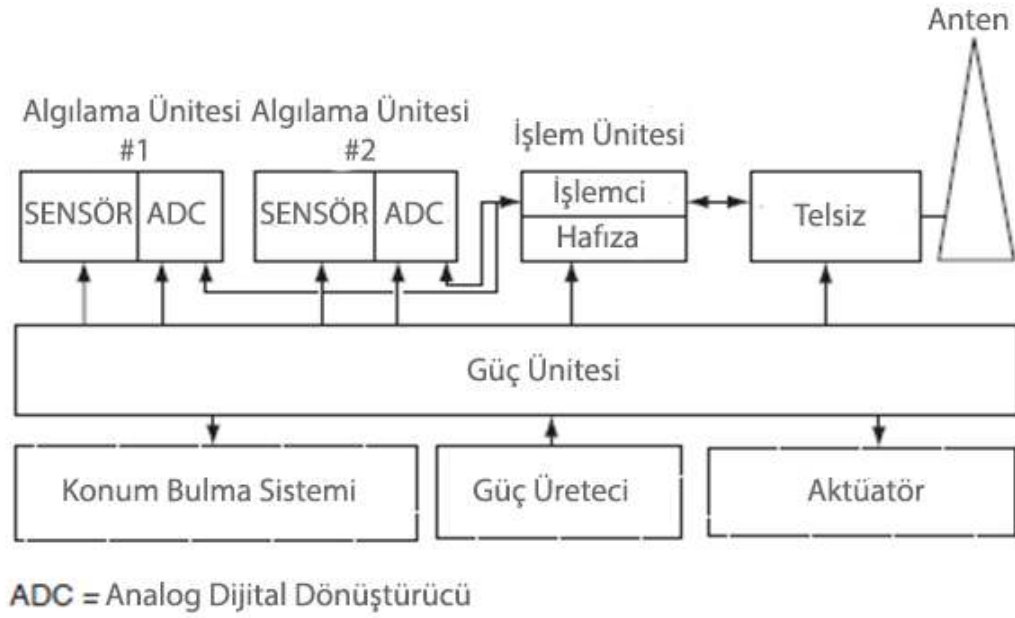
Kablosuz sensör ağların temelini oluşturan kablosuz sensör düğümler 4 temel unsurdan oluşur. Bunlar algılama ünitesi, işlem ünitesi, haberleşme ünitesi ve güç üniteleridir. Güç tüketimi sensör düğümlerin tasarımını kısıtladığından dikkate alınması gereken önemli bir konudur. Çoğu durumda haberleşme devresi, enerjinin en çok tüketildiği temel sensör bölümüdür [2,3]. Kaynakların verimli kullanılmasını amacıyla birçok gömülü işletim sistemi tasarlanmıştır. Ayrıca yine kaynakların kısıtlı olmasından dolayı, kullanılamayan her kaynağın eksikliğini gidermek için birçok araştırma konusu oluşmuştur.

Ayrıca bu 4 temel birimin yanında uygulamaya yönelik konum bulma sistemi, güç üretici gibi ek sistemler ya da üniteler kullanılabilir.

Şekil 2.5’de kablosuz sensör düğümlerin genel yapısı ve bu genel yapıya ek olabilecek diğer üniteler görülmektedir.

IRIS, MicaZ, TelosB, Imote2, Cricket gibi birçok hazır sensör düğümü piyasada mevcuttur. Bunların yanında kendi sensör düğümümüzü tasarlayabileceğimiz giriş çıkış birimleri bulunan Zigbee protokolünü destekleyen kablosuz haberleşme modülleride

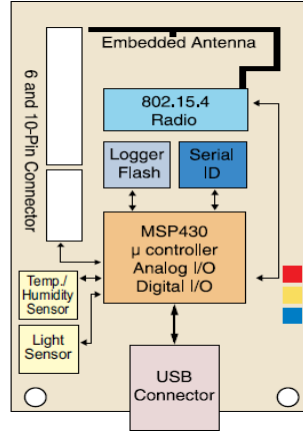
bulunmaktadır. Bu çalışmada Digi firmasının üretmiş olduğu XBee haberleşme modülleri kullanılmış bu modüle ek olarak güç ünitesi merkezi işlem birimi harici olarak bağlanarak kablosuz düğüm elde edilmiştir.



Şekil 2.5. Sensör Düğümün Yapısı

2.5. Memsic TELOS B TPR2420CA Sensör Düğümü

TelosB kablosuz sensör düğümü işlem ünitesi olarak; 16-bit RISC mimarisine sahip, aktif durumda 1.8 mA, uyku modunda ise 5.1 uA akım çeken TI (Texas Instrument) firmasının MSP430 işlemcisini kullanır. Haberleşme ünitesi olarak 2400 Mhz ile 2483.5Mhz frekansları (ISM) arasında açık alanda 75 ile 100 metre kapalı alanda ise 20 m ile 30 metre arasında haberleşmeyi mümkün kılan TI firmasının CC2420 zigbee radyo devresini, algılama ünitesi olarak Hamamatsu S1087 ışık sensörü, Sensorion SHT11 nem sensörü ve bir adet sıcaklık sensörü, güç ünitesi olarak ise 2 adet AA pil kullanır. Şekil 2.6'da TelosB sensör düğümünün blok diyagramı Şekil 2.7'de ise genel görünümü yer almaktadır [13].



Şekil 2.6. TelosB TPR2420CA Blok Diyagramı [13]



Şekil 2.7. TelosB TPR2420CA Genel Görünümü [13]

2.6. Kablosuz Sensör Ağlar İçin Gömülü İşletim Sistemleri

Geleneksel işletim sistemlerinin sistem yazılımları, bilgisayar kaynaklarını yöneten programları, çevresel aygıtların kontrolünü gerçekleştiren programları ve uygulama yazılımlarının yazılım soyutlamasını gerçekleştiren programları içerir. Geleneksel işletim sistemlerinin temel fonksiyonları süreç yönetimi, hafıza yönetimi, CPU zamanı yönetimi, dosya sistemi yönetimi ve cihaz yönetimidir.

Kablosuz sensör düğümlerin kısıtlı kaynaklara sahip olmasından dolayı geleneksel işletim sistemleri kablosuz sensör ağlar için uygun işletim sistemleri değildir. Kablosuz sensör ağlar kendilerine özel yeni işletim sistemlerine ihtiyaç duyar ve bu işletim sistemlerinin ortak özellikleri şunlar olmalıdır.

- Hafızanın en verimli şekilde kullanılmasını hedeflemelidir.
- Gerçek zamanlı uygulamalar için gerçek zamanı desteklemelidirler.
- Güç yönetimini desteklemelidirler.
- Değişik türden düğümlere özgü ara yüzleri içermelidirler.

Kablosuz sensör düğümleri destekleyen işletim sistemlerine TinyOS, Mate, MagnetOS, MANTIS, OSPM, EYES OS, SenOS, EMPERALDS ve PicOS örnek olarak verilebilir.

2.6.1. TINYOS

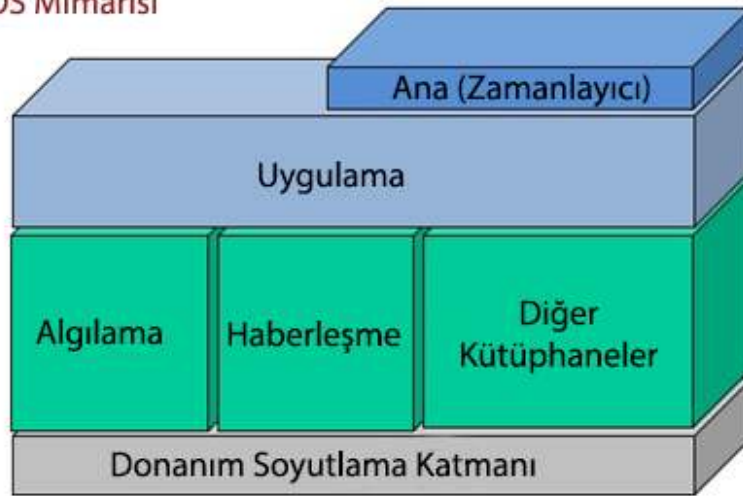
TinyOS açık kaynak kodlu, esnek, bileşen tabanlı kablosuz sensör ağlar için tasarlanmış uygulamaya özel bir işletim sistemidir. TinyOS çok düşük bellek gereksinimleri ile eşzamanlı programları destekler.

TinyOS bileşen kütüphanesi ağ protokollerini, dağıtık hizmetleri, sensör sürücülerini ve veri toplama araçlarını içerir.

TinyOS monolitik mimari sınıfı altında incelenebilir. Monolitik mimari, görevler arası iletişimi, çekirdek sürücülerini ve görevler arasında hızlı geçişi sağlayabilen tek başına büyük bir program tabanlı bir mimaridir. TinyOS paylaşımlı tek bir yığın kullanır ve kullanıcı ve çekirdek alanı bu yığını kullanır.

TinyOS bileşen modelini kullanmaktadır. Bir uygulamanın ihtiyaçlarına göre farklı bileşenlerin düğüm üzerine zamanlanmış statik imajları yapıştırılır. Bir bileşen bir ya da daha fazla ara yüzden oluşur. Bileşenler arası iletişim için komutlar ve olaylar kullanılır. Görevler ise bileşen içi eşzamanlılık için kullanılır. Şekil 2.8’de TinyOS işletim sisteminin genel mimarisi görülmektedir.

TinyOS Mimarisi



Şekil 2.8. TinyOS İşletim Sistemi Mimarisi [14]

2.7. XBee RF Haberleşme Modülleri

XBee modülleri Digi Firması tarafından geliştirilip üretilen farklı çıkış güçlerine sahip 802.15.4, Zigbee Mesh, multipoint haberleşme protokollerini destekleyen kablosuz haberleşme modülleridir. Bu modüllerin en büyük avantajlarından biri mikrodenetleyici sistemler ve bilgisayarlarla AT komut seti kullanmasından dolayı kolay bir şekilde seri haberleşme ile haberleşebilmesidir.

Attention kelimesinin kısaltması olan AT komut seti “Hayes” adında bir telekomünikasyon firmasının belirlediği bir komut seti standardıdır. Komut setindeki komutlar AT karakterleri ile başlar. Genellikle modemlerde kullanılan bu standart XBee modüllerde de kullanılmaktadır.

Uygulamamızda kullandığımız XBee modülü 2mW çıkış gücüne sahip açık alanda 120 metreye kadar haberleşme imkânı sağlayan XB24-CZ7WIT-004 modelidir. Genel özellikleri aşağıdaki gibidir.

- 3.3V @ 40mA
- Veri Hızı: 250kbps
- 2mW çıkış (+0dBm)
- Haberleşme Mesafesi: 120m
- Anten Tipi: Kablo Anten
- FCC sertifikası
- 6 adet 10-bit ADC giriş pini
- 8 adet dijital I/O pini
- 128-bit Şifreleme
- Kapalı veya açık alan konfigürasyonu
- AT ve API ayar komutları

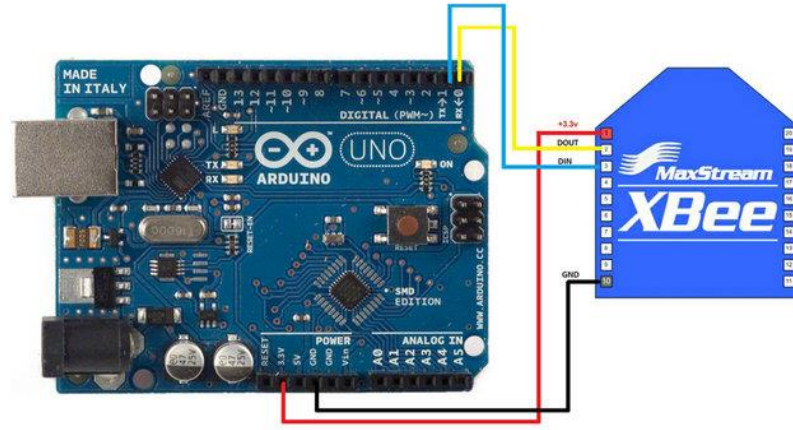
XBee modüller parametre ayarlamaları dışında programlanamadıkları için harici bir merkezi işlem birimine ihtiyaç duyarlar.

Şekil 2.9’de XBee modülün genel görünümüne yer verilmiştir.

Şekil 2.10’da merkezi işlem birimi olarak Arduino elektronik kartı kullanılmış ve XBee modül ile bağlantısı gösterilmiştir.



Şekil 2.9. XBee Modül Genel Görünümü



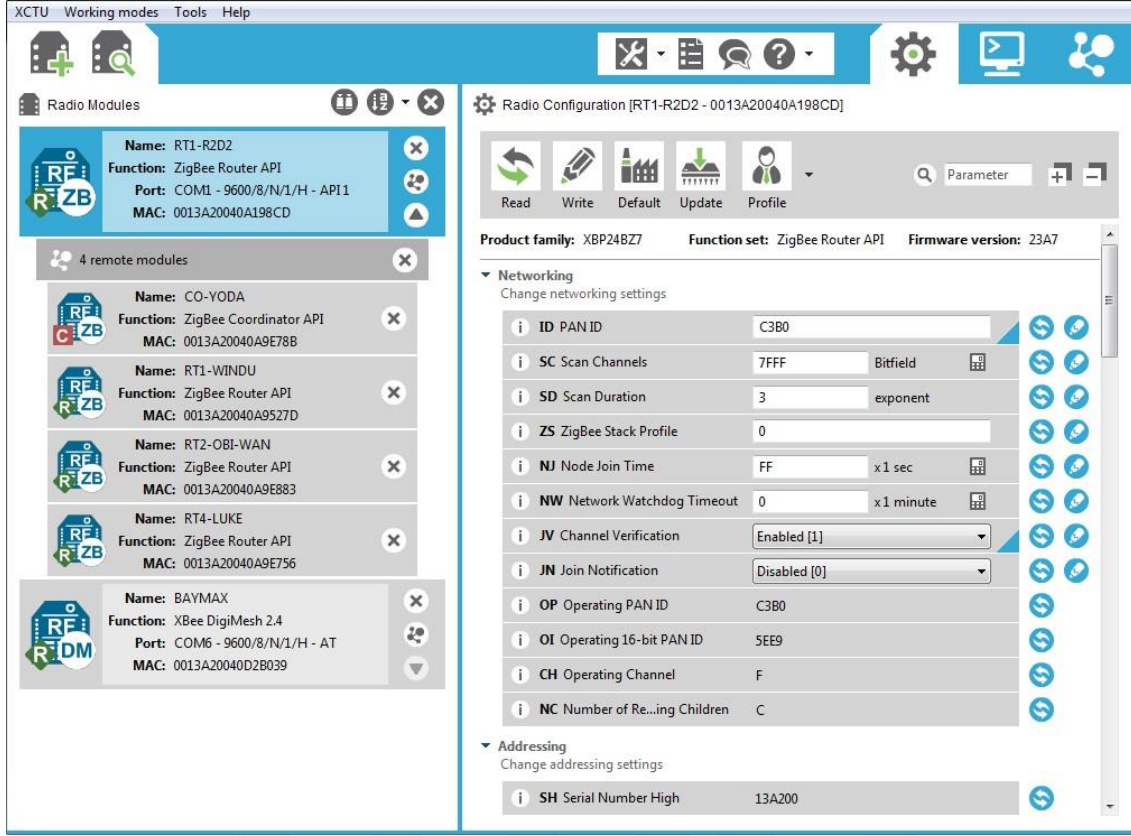
Şekil 2.10. Arduino ve XBee Bağlantı Şeması

2.7.1. XBee AT ve API Çalışma Modları

AT modu "Şeffaf" mod ile eş anlamlıdır. AT modunda, XBee modülüne gönderilen herhangi bir veri, hafızadaki hedef adresi tanımlanan uzak modüle gönderilir. Paket oluşturma gerekli değildir. Veri, gönderici XBee'nin RX'ine yazılır ve hedef düğümün RX'inden okunur. Hiçbir paket oluşturulmadığından, hedef adresi ve türü hep sabittir. API modunda ise frame yapısı oluşturulup değişken adreslere veriler gönderilebilir. AT modunda XBee module sadece veri gönderilirken API modunda çerçevenin tamamı gönderilir.

2.7.2. XBee XCTU Konfigürasyon Yazılımı

XCTU Konfigürasyon yazılımı XBee modüllerin konfigürasyonunu, mesafe testlerini ve ağdaki diğer XBee modüller ile ilişkilerini görüp değiştirebileceğimiz bir yazılımdır. Ayrıca bu yazılım ile gerekli çerçeve tipinde çerçeveler üretebiliriz. XBee modüllerimizin PAN ID ve Node ID gibi önemli parametrelerinin yanı sıra XBee modüllerimiz üzerinde bulunan IO portlarımızın giriş çıkış bilgileri yine bu yazılım ile kolaylıkla değiştirilebilir. Şekil 2.11'de XCTU yazılımının genel görünümüne yer verilmiştir.



Şekil 2.11. XCTU Yazılımın Genel Görünümü

3. KABLOSUZ SENSÖR AĞLARDA KONUM BULMA YÖNTEMLERİ

3.1. Genel Bilgi

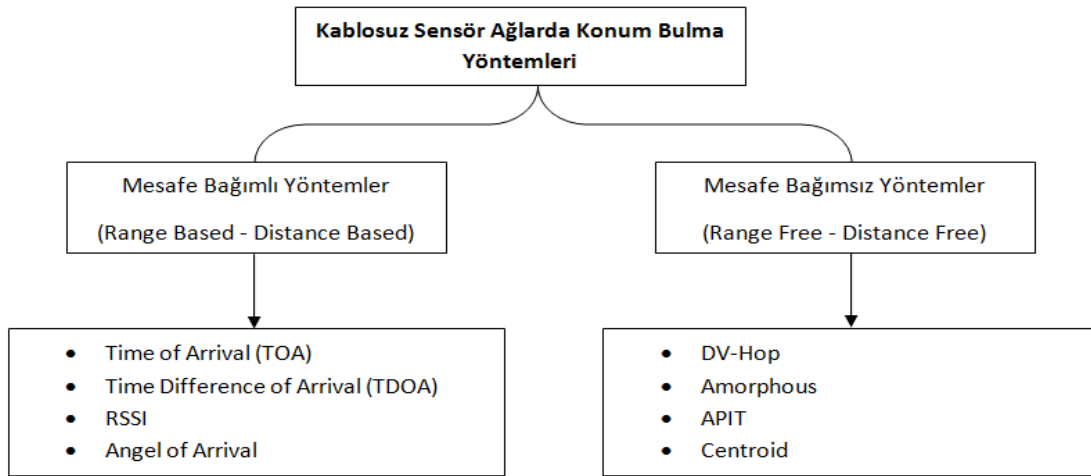
Kablosuz sensör ağlarda konum bulma, GPS kullanılmayan ağlarda ya donanım bağımlıdır ya da topoloji bağımlıdır [15]. Donanım bağımlı algoritmalar sinyal gücü, sinyalin geliş açısı gibi verileri elde etmek için ek donanımlara ihtiyaç duyarlarken topoloji bağımlı algoritmalar ek bir donanıma ihtiyaç duymazlar ancak kurulan topoloji içinde konumları belirli düğümlere ihtiyaç duyarlar.

GPS kullanılmayan kablosuz sensör ağlarda konum bulma algoritmaları temel olarak mesafe bağımlı ve mesafe bağımsız yöntemler olarak ikiye ayrılırlar.

Mesafe bağımlı yöntemler, temelde kablosuz sensör düğümlerinin birbirlerine olan uzaklıklarından yola çıkarak konum bulmayı amaçlarlar. Dağıtık bir mimaride düğümler arasındaki mesafeler bilinmediği için bu mesafeler değişik yöntemlerle tahmin edilmeye çalışılırlar.

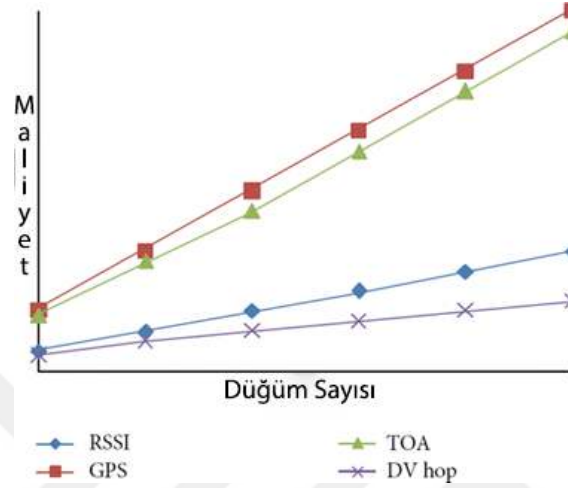
Mesafe bağımsız yöntemler donanım bağımsızdır bu yüzden maliyet yönünden en uygun yöntemlerdir ancak konum bulmadaki hata oranları mesafe bağımlı yöntemlere göre daha yüksektirler. Mesafe yerine, hop sayısı gibi değişik parametreler kullanarak konum bulma amaçlanmıştır.

Şekil 3.1’de temel konum bulma yöntemleri görülmektedir.

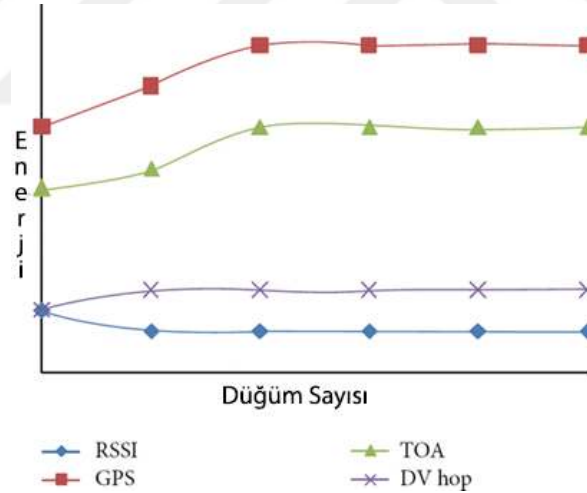


Şekil 3.1. Temel Konum Bulma Yöntemleri

Maliyetin ve güç tüketiminin önemli olmadığı kablosuz sensör ağlarda GPS ile konum bulma en güvenilir ve doğruluğu yüksek yöntemdir. Şekil 3.2 ve Şekil 3.3'te, maliyet ve enerji ile bazı konum bulma yöntemlerinin, ağdaki düğüm sayısına göre olan ilişkisi gösterilmiştir [16].



Şekil 3.2. Konum Bulma Yöntemlerinin Düğüm Sayısına Göre Maliyet Grafiği [16]



Şekil 3.3. Konum Bulma Yöntemlerinin Düğüm Sayısına Göre Enerji Tüketimleri Grafiği [16]

3.2. Mesafe Bağımlı Yöntemler (Range Based – Distance Based)

Mesafe bağımlı konum bulma yöntemlerinin, konum bulmada esas aldığı parametre mesafedir. Mesafeler değişken hata oranlarına sahip farklı yöntemler kullanılarak bulunabilirler.

3.2.1. Time of Arrival (TOA)

Gönderici ve alıcı düğümler arasındaki sinyalin yayılma hızı ve sinyalin gönderiminde geçen süre biliniyorsa, mesafe denklem 3.1 yardımıyla rahatlıkla bulunabilir.

$$\text{Mesafe}=(T_2-T_1)*V \quad (3.1)$$

3.1 numaralı denklemdeki T_2 varış zamanı T_1 sinyalin çıkış zamanını V ise sinyalin yayılma hızı değerleridir. Bu yöntemdeki temel problem alıcı ve gönderici düğüm arasında zaman senkronizasyonu zorunluluğudur[17]. Gönderici ve alıcı arasındaki senkronizasyondaki küçük değişiklikler mesafenin büyük hata oranlarıyla bulunmasına neden olmaktadır.

3.2.2. Time Difference of Arrival (TDOA)

TOA'daki zaman senkronizasyonu sorununu ortadan kaldırmak için kullanılan bir yöntemdir. Yöntemin temel mantığı, yayılma hızları farklı olan iki ayrı sinyal kullanmaktır. Örnek olarak, yollayıcı düğüm tarafından T_1 zamanında radyo sinyali yollanıp, T_2 zamanında alıcıya ulaşsın ve bunu takiben ses sinyali 3.2 numaralı denklem ile bulunan T_3 zamanında yollanıp T_4 zamanında alıcıya ulaşsın. Bu örnekteki mesafe, denklem 3.3 yardımı ile bulunur. Bu yöntemin dezavantajı iki ayrı sinyal için iki ayrı sinyal üretici donanımına ihtiyaç duyması ve bundan dolayı maliyetin artmasıdır.

$$T_3=T_1+T_{(\text{bekleme})} \quad (3.2)$$

$$\text{Mesafe}=(V_1-V_2)*(T_4-T_2-T_{(\text{bekleme})}) \quad (3.3)$$

3.2.3. RSSI

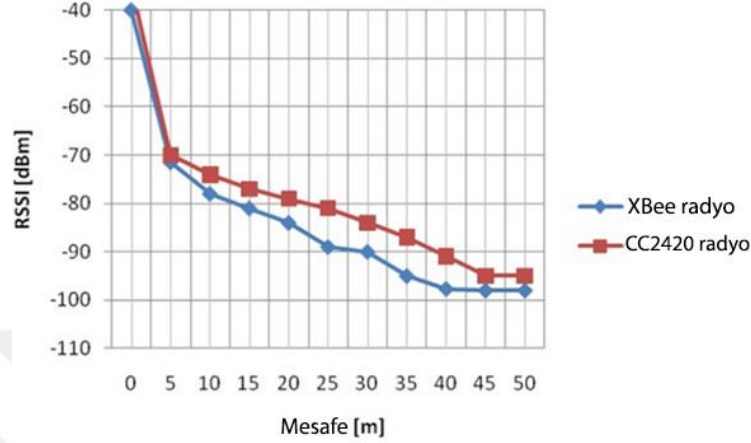
RSSI gelen sinyal gücünü belirten bir değerdir. Her yonga üreticinin RSSI değer aralığı 0 ile 255 arasında farklılık gösterir ve kullanımından önce dBm olarak hesaplanır. Sinyal gücü düştükçe, RSSI'nin dBm değeri eksi yönde düşer. Birçok kablosuz haberleşme yongasında RSSI değeri hesaplanabilir olduğundan ek donanım ihtiyacı yoktur ve bu yüzden RSSI ile konum bulma ekonomik bir yöntemdir.

RSSI değeri mesafe tahmininde özellikle gürültülü ortamlarda bazen yetersiz kalabilmektedir. Böyle durumlarda RSSI değerine ek olarak sinyal kalitesinin de işe dahil edilmesi amacıyla LQI (Link Quality Indicator) değeride hesaplanır. LQI sinyal gücünü 0 ile 255 arasında bir değer ile belirten bir göstergedir. LQI ile RSSI arasında aşağıdaki gibi bir ilişki vardır.

- Zayıf sinyal yüksek gürültüde, RSSI düşük LQI düşük
- Zayıf sinyal az gürültüde, RSSI düşük LQI yüksek

- Güçlü sinyal az gürültüde RSSI yüksek LQI yüksek
- Güçlü sinyal güçlü gürültüde RSSI yüksek LQI düşük.

Şekil 3.4’de iki ayrı kablosuz haberleşme yongasının, RSSI ile Mesafe arasındaki orantılarının grafiği görülmektedir.



Şekil 3.4. XBee ve CC2420 Yongasının RSSI - Mesafe Grafiği [18]

3.2.4. Angel of Arrival (AOA)

Bu yöntem ile kablosuz sensör düğümlerin birbirlerini direk olarak gördüğü açık ve engelsiz bir ortamda, gelen sinyal açısına göre konum saptama yapılır. Gelen sinyal açısını tespit amaçlı çanak, mikrofon gibi ek donanımlar kullanıldığından maliyeti ve konum bulmadaki kesinliği yüksektir.

3.3. Mesafe Bağımsız Yöntemler (Range Free – Distance Free)

3.3.1. DV-Hop

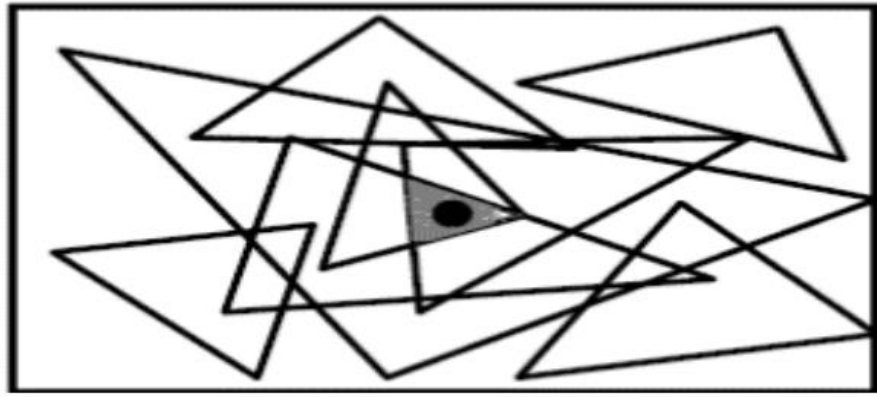
Konum bilgileri bilinen çapa düğümler, konum bilgileri ve başlangıç değeri 0 olan sayaç değişkeni bulunan bir veri paketini, ağdaki komşu düğümler aracılığıyla diğer çapa düğümlere iletirler. Veri paketleri ağdaki her düğüme iletildiğinde veri paketinde bulunan sayaç değeri bir artırılır. Veri paketleri değişik yollardan çapa düğüme ulaştığında, çapa düğüm sayaç değeri en küçük olan veri paketine göre ortalama atlama mesafesini, denklem 3.4’te bulunan matematiksel ifade ile hesaplar [19].

$$\text{HopSize} = \frac{\sum \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{\sum h_j} \quad (3.4)$$

Ortalama atlama mesafesi hesaplandıktan sonra, konumu tespit edilecek düğüm, çapa düğümler arasındaki atlama sayısından yolla çıkarak çapa düğümlere olan uzaklıkları bulunur ve konumu konum bulma algoritmalarından uygun olanı seçilerek bulunur.

3.3.2. APIT

N adet çapa düğüm içinden her seferinde 3 adet çapa düğüm seçilir ve aranan düğümün bu 3 çapa düğümün köşelerini oluşturduğu üçgende olup olmadığı test edilir. Tüm çapa düğümler için bu işlem yapıldıktan sonra aranan düğümün bu üçgenlerin kesişim alanlarında olduğu varsayılır. Şekil 3.5’de örnek bir APIT algoritması görülmektedir [20].



Şekil 3.5. APIT Yöntemi İle Konum Bulma

3.3.3. Centroid

Serbest olan düğümü, en az 3 adet çapa düğümün kesişim noktasına konumlandırıan hata oranı yüksek olan bir yöntemdir. Konum bulma yöntemlerinin en basitidir. Konumlandırma denklem 3.5’teki matematiksel yöntem ile yapılır [19].

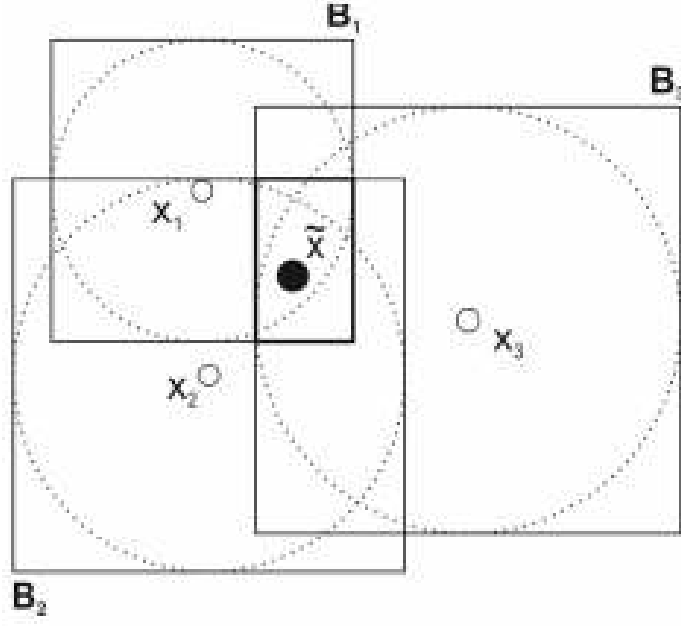
$$(X_{est}, Y_{est}) = \left(\frac{X_1 + \dots + X_N}{N}, \frac{Y_1 + \dots + Y_N}{N} \right) \quad (3.5)$$

3.3.4. Bounding Box Yöntemi

Bounding Box algoritmasının ana mantığı, çapa düğümlere olan tahmini uzaklıklar bulunduktan sonra, çapa düğümler etrafında uzaklık bilgisine göre kutular oluşturularak, konumu tespit edilecek düğümün kutuların kesişim alanında olduğunu varsaymaktır. Tahmin edilen konum kesişimin oluşturduğu alanın merkezidir ve denklem 3.6 ile bulunur.

$$\bar{x} \in \left\{ \bigcap_i B_i \right\} \quad (3.6)$$

Şekil 3.6’da Bounding Box algoritmasının genel yapısı görülmektedir.



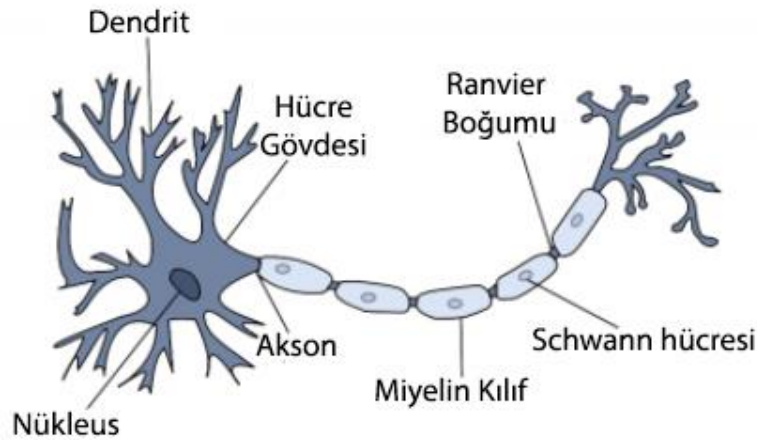
Şekil 3.6. Bounding Box Algoritması ile Konum Bulma [19]

4. YAPAY SİNİR AĞLARI

4.1. Genel Bilgi

Yapay sinir ağıları örneklerden yola çıkarak öğrenme fonksiyonunu yerine getiren bilgisayar sistemleridir. İnsan beyninin yapısından yola çıkılarak modellenen bir yapıdır. İnsan beynindeki sinir hücreleri gibi yapay sinir ağıları da yapay sinir hücreleri ve bu hücrelerin birbirleriyle, bir ağırlık değeri olan yapay sinir ağı bağlantılarından oluşur. Şekil 4.1’de insan sinir hücre yapısı görülmektedir[20]. Bağlantılardaki ağırlık değeri yapay sinir ağına has olup bu ağırlık bilgisini oluşturur. Örnekler ile yapay sinir ağıları eğitilerek bu ağırlıklar hesaplanır ve eğer yapay sinir ağıımızın sonuçlardaki hata oranı kabul edilebilir ise bu ağırlıklar kabul edilir değil ise yapay sinir ağıımız tekrar tekrar eğitilerek hata oranı düşürülmeye çalışılır.

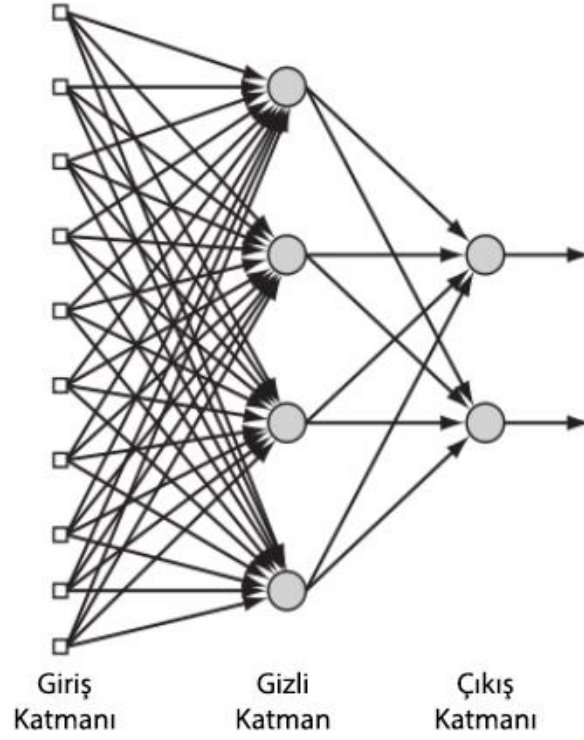
Örüntü tanıma, filtreleme, sınıflandırma, uydurma (fitting) yöntemleri için yapay sinir ağıları en iyi yöntemlerden biridir [21].



Şekil 4.1. İnsan Sinir Hücresi Yapısı [21]

Yapay sinir ağıları birbirlerine bağlı olmak üzere, girdi katmanı (input layer), ara yada gizli katman (hidden layer), ve çıktı katmanı (output layer) olmak üzere üç katmandan oluşur.

Yapay sinir ağlarının eğitimi, testi ve doğrulaması, öğrenmesini istediğimiz sistemden elde edilen örneklerle gerçekleştirilir. Yapay sinir ağıları oluşturulurken, ağı topolojisi, ve eğitim algoritması doğru seçilmeli ve eğitim veri setimizin yeterli sayıda olmalıdır. Şekil 4.2’de bir yapay sinir ağı topolojisi görülmektedir [22].



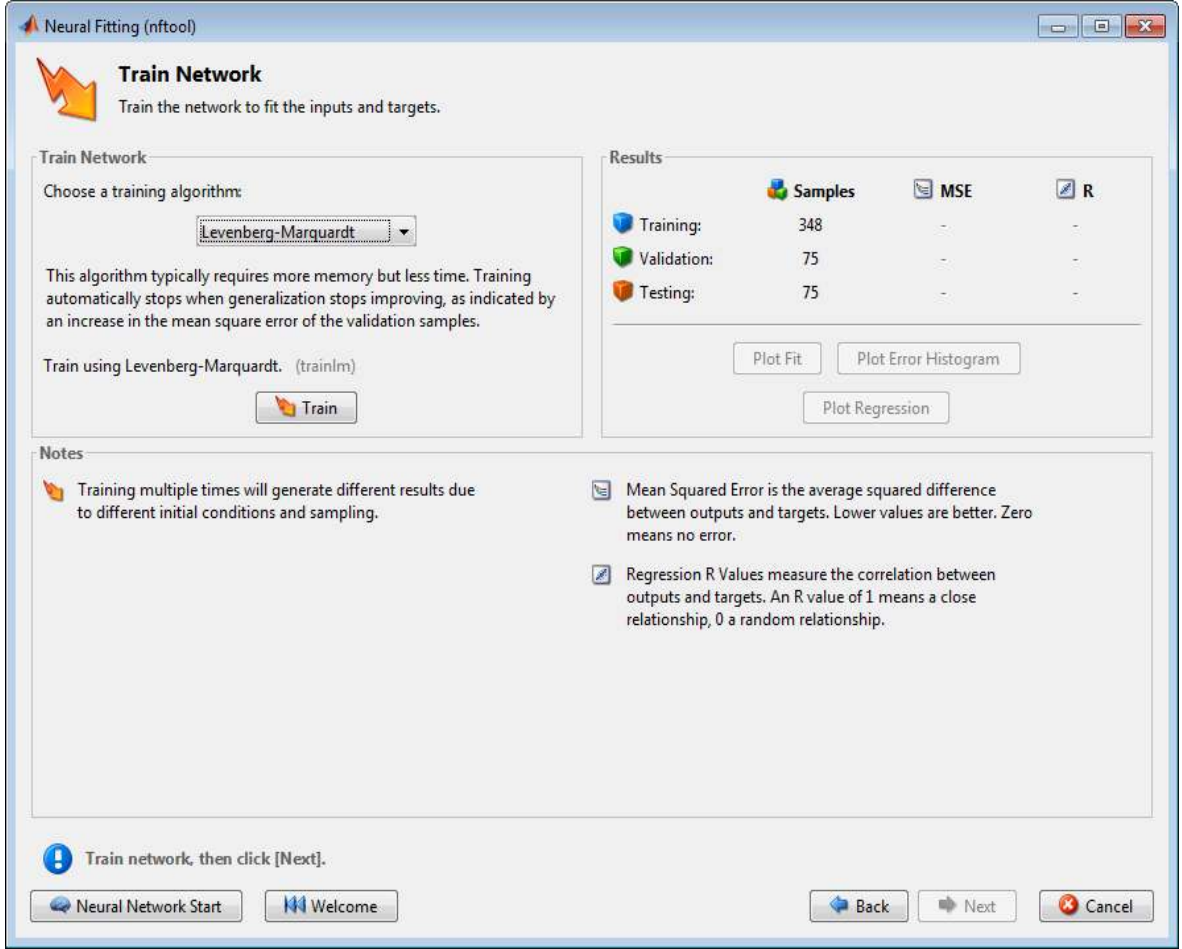
Şekil 4.2. Yapay Sinir Ağı Genel Topolojisi [22]

4.2. Matlab İle Yapay Sinir Ağları Kullanımı

Matlab ortamında eğri uydurma araçlarının başında Neural Fiting eğri uydurma aracı gelmektedir. Bu araç ile oluşturduğunuz yapay sinir ağını eğitim test verilerinizle eğitmeniz gerekmektedir. Bu araca parametre olarak, giriş çıkış matrislerinizi, bu verilerin yüzde kaçının eğitim için yüzde kaçının doğrulama ve yüzde kaçının test için kullanıldığını, gizli katmandaki hücre sayısını ve öğrenme algoritmasını girmeniz gerekmektedir. Bu araçta üç adet öğrenme algoritması mevcuttur bunlar, Levenberg-Marquardt (trainlm), Bayesian Regularization (trainbr) ve Scaled Conjugate Gradient algoritmalarıdır.

Birçok problem için Levenberg-Marquardt (trainlm) önerilir, ancak bazı gürültülü ve küçük problemler için Bayesian Regularization (trainbr) daha uzun sürmesine rağmen daha iyi bir çözüm elde edebilir. Bununla birlikte, büyük problemler için, diğer iki algoritmanın kullandığı Jacobian hesaplamalardan daha verimli bellek kullanımı olan gradient hesaplamaları kullanan Scaled Conjugate Gradient algoritması önerilir [23].

Şekil 4.3'te NeuralFitting aracının genel görünümü görülmektedir.



Şekil 4.3. Matlab Neural Fitting Aracı

Ağ performansından memnun kalınmadığı takdirde, ağ tekrar eğitmeli, nöron sayısını artırmalı ve daha büyük bir eğitim verisi seti oluşturulmalıdır. Eğitim setinin performansı iyi ancak test setinin performansı kötüyse nöron sayısını azaltmakta fayda vardır. Eğitim performansı zayıfsa, nöron sayısını artırmak sonuçları pozitif yönde etkileyecektir. Oluşturulan ağ performansından memnun kalındığı takdirde yapay sinir ağının fonksiyon bloğu oluşturulabilir ve matlab içinde tekrar tekrar bu ağ fonksiyonu çağrılıp kullanılabilir.

5. SİSTEM TASARIMI VE UYGULAMASI

5.1. Genel Açıklama

Amaçlanan uygulamanın gerçekleştirilmesi için 5 adet Zigbee haberleşme modülüne, bu modüllerin kontrolü amaçlı elektronik kontrol kartlarına, hareketli programlanabilir bir araca ve sahadan elde edilen verileri kaydetmek ve kendisine bağlı Zigbee modül aracılığıyla hareketli araca veri göndermek amacıyla bir adet bilgisayara ihtiyaç duyulmuştur.

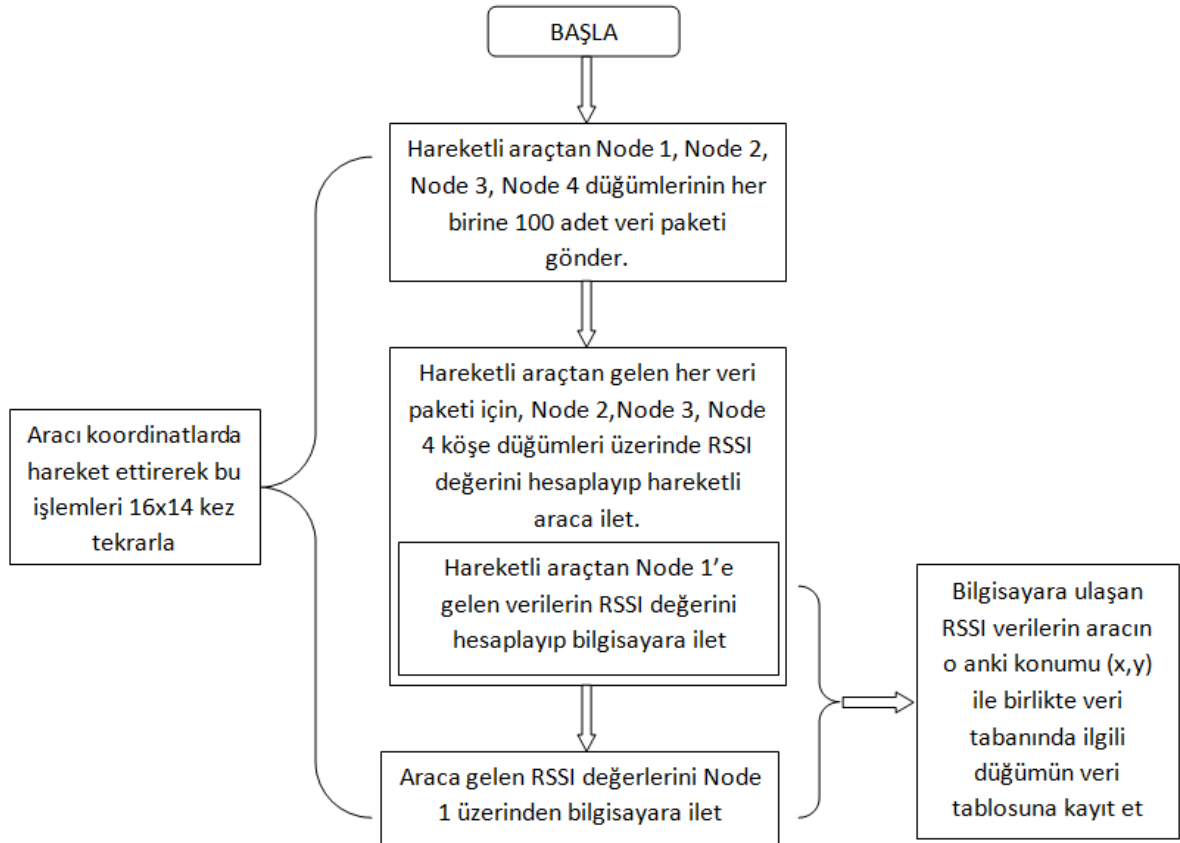
Zigbee haberleşme modülü olarak, programlama ve haberleşme kolaylığı göz önüne alınarak Digi firmasının XBee modülleri seçilmiştir. 5 adet XBee modülden 3 adedi köşe düğüm olarak (Node 2, Node 3, Node 4), 1 adedi ana istasyon (base station, Node 1) ve 4. köşe düğüm olarak, son XBee modül ise aracımızda kullanılmıştır.

Node 2, Node 3 ve Node 4 olarak isimlendirilen düğümlerin elektronik kontrol kartı olarak programlama kolaylığı, zengin kütüphane ve örnek programları nedeniyle ardino uno elektronik kartı seçilmiştir. Bu kartlar sayesinde, XBee modüllere gelen veriler okunabilir ve aynı şekilde seri haberleşme ile XBee modüller üzerinden istenen veri istenilen bir başka XBee modüle iletilir. Arduino uno, XBee modülleri ve uygun bir güç kaynağı ile oluşturulan bu düğümlerin asli görevi, hareketli araçta bulunan XBee modülden gelen sinyalin RSSI değerini hesaplayıp, bu değeri ana istasyon olan Node 1 düğümüne iletmektir.

Node 1 olarak isimlendirilen ana istasyon düğümümüzde, elektronik kart yerine usb port üzerinden bağlı olan bir bilgisayar kullanılmıştır. Bu Xbee modüle gelen veriler usb port üzerinden bilgisayara iletilir. Aynı şekilde bilgisayardan XBee modüle iletilen veri, istenilen adrese iletilir. Bu düğümün asıl görevi hareketli araçtan gelen sinyallerin RSSI değerlerini hesaplamak ve Node 1, Node 2, Node 3 düğümlerinden gelen RSSI değerlerini bilgisayara iletmektir.

Hareketli aracımızın tasarımında, hassas adım sayılarına sahip iki adet step motor ve meOrion kontrol kartını bünyesinde bulunduran Makeblock firmasının mDrawBot kitinde bulunan malzemeler seçilmiştir. Bu kitte bulunan malzemeler kullanılarak oluşturulan aracımıza bir adet XBee modül takılarak çevre düğümlerle haberleşmesi sağlanmıştır. Aracımızın asli görevi uygulama alanındaki birbirinden farklı 224 noktaya hareket ederek,

bu noktaların her birinde Node 1 Node 2, Node 3 ve Node 4 düğümlere 100'er adet içeriği önemsiz veri paketi göndererek, kendisiyle bu düğümler arasındaki sinyal gücü göstergesi olan RSSI değerlerinin hesaplanmasını sağlamaktır. Bu sayede deney ortamından, konum bulma amacıyla tasarlanmış olduğumuz yapay sinir ağlarının eğitim ve test verilerinin elde edilmesi amaçlanmıştır. Şekil 5.1'de yapılan çalışmanın genel algoritması görülmektedir.



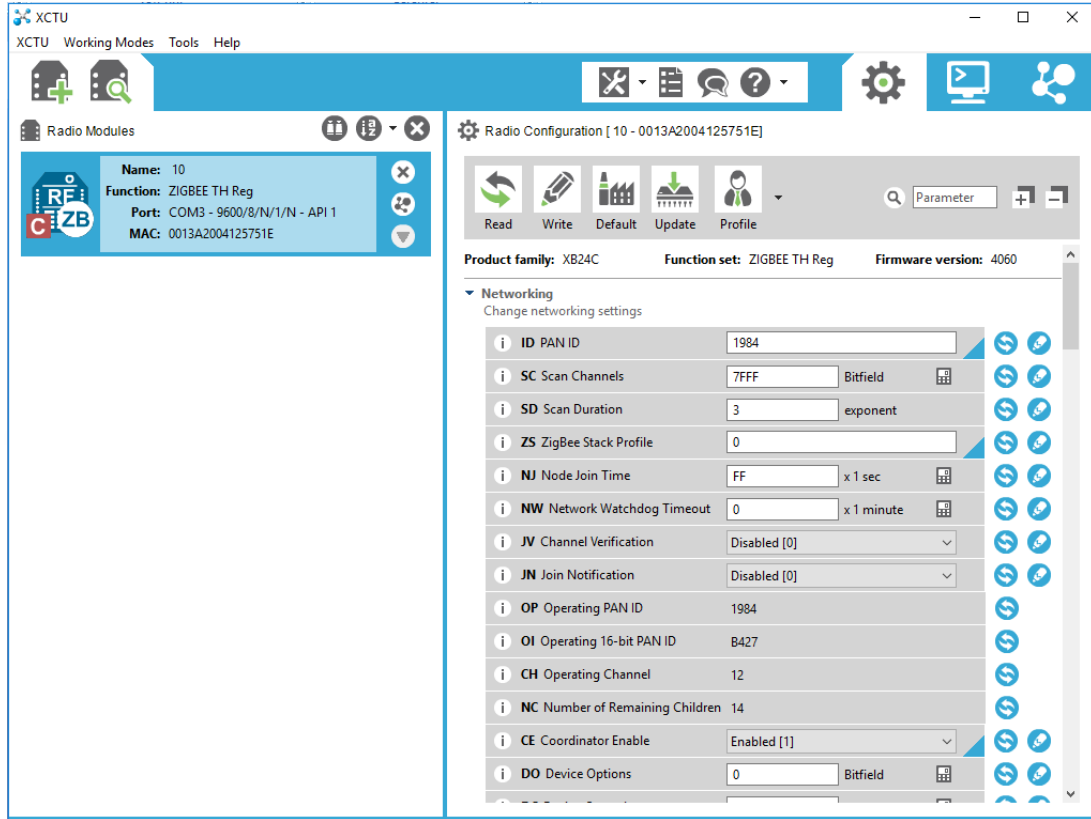
Şekil 5.1. Uygulamanın Genel Algoritması

5.2. Xbee Modüllerin Parametre Ayarlarının Yapılması

Xbee modüllerimizin kendi aralarında, kontrol üniteleri ve bilgisayarla haberleşebilmeleri için konfigürasyon ayarlamalarının yapılması gerekmektedir. Bu ayarları gerçekleştirmek için Digi firmasının XCTU konfigürasyon yazılımı kullanılmıştır. Bu yazılımı, bilgisayara USB port üzerinden bağlı XBee modülün, konfigürasyon ayarlarını okuyup değişiklik yapma imkanı sağlar.

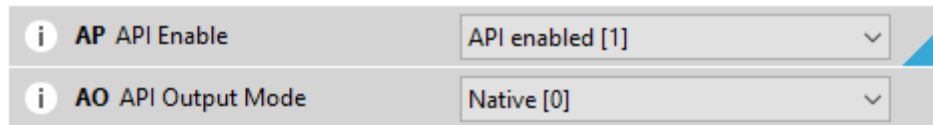
XBee modüllerde yapılması gereken ayarların başında tüm modüllerin aynı ağda olmasını sağlamak yer almaktadır. Bunun için tüm XBee modüllerimizin PAN ID'si 1984

olacak şekilde değiştirilmiştir. Şekil 5.2’de XCTU yazılımının genel görünümü ve PAN ID’imizin değiştirildiği kısım görülmektedir.



Şekil 5.2. XCTU Yazılımının Genel Görünümü

XBee modüllerimizin kontrol devreleri veya bilgisayarlarla seri port üzerinden haberleşebilmeleri için API Enable ve API Output Mode ayarlarının da Şekil 5.3’te görüldüğü gibi yapılması gerekmektedir.



Şekil 5.3. AP ve AO Konfigürasyon Ayarları

Bu ayarlara ek olarak tüm XBee modüllerimizin en son güncel firmware’e sahip olmasında fayda vardır. Bunun için yine XCTU yazılımında Update aracı kullanılarak güncelleme işlemleri yapılabilir.

5.3. Lineer Hareket Eden Araç Tasarımı ve Programlanması

Hareketli aracın oluşturulması için malzemelerin seçilmesinde hassas hareket yapısı göz önüne alınarak step motor ihtiva eden makeblock firmasının mdrawbot kiti seçilmiştir. Bu kit içinde step motorlar, Aurdino IDE ile programlanabilen kontrol ünitesi, motor sürücü devreleri ve aracımızı oluşturmak için kullanacağımız tekerlekler, çerçeveler, vidalar ve somunlar bulunmaktadır.

Tasarlanan araçta iki adet step motor, iki adet tekerlek, bir adet sarhoş tekerlek, iki adet motor sürücü kartı, bir adet kontrol kartı, iki adet seri bağlı 5V, 5200 mah'lık güç ünitesi ve bir adet aracımızın bilgisayar üzerinden kontrolünü sağlayacak XBee modülü bulunmaktadır.

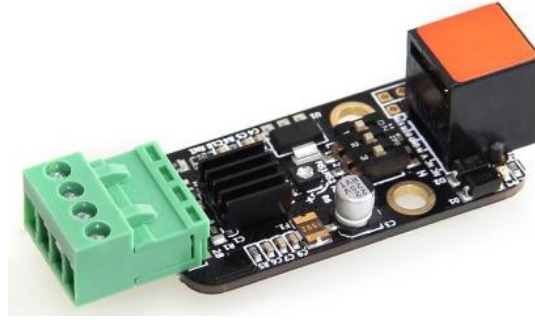
Aracın tasarımıda dikkat ettiğimiz en önemli hususların başında ağırlık merkezinin aracın dikey eksenini üzerinde olması gerekliliğidir. Ağırlık merkezinin dikey eksen üzerinde olmaması, aracın sağa yada sola yalpa yapmasına neden olacağından önemli bir konudur. Yapılan denemelerle, aracın düz hareket etmesi sağlanmış ve gerekli noktalara ekstra ağırlıklar atılmıştır.

XBee modülümüz 3.3 V beslemeyle çalışmaktadır. Araç kontrol devresinde 3.3V'luk gerilim çıkışı bulunmadığından XBee modülümüzü beslemek için LD33V voltaj regülatörü kullanılarak 3.3V'luk voltaj regüle edilmiştir.

Şekil 5.4'de kontrol kartımızın, Şekil 5.5'de ise Step motor sürücü kartımızın genel görünümü görülmektedir.



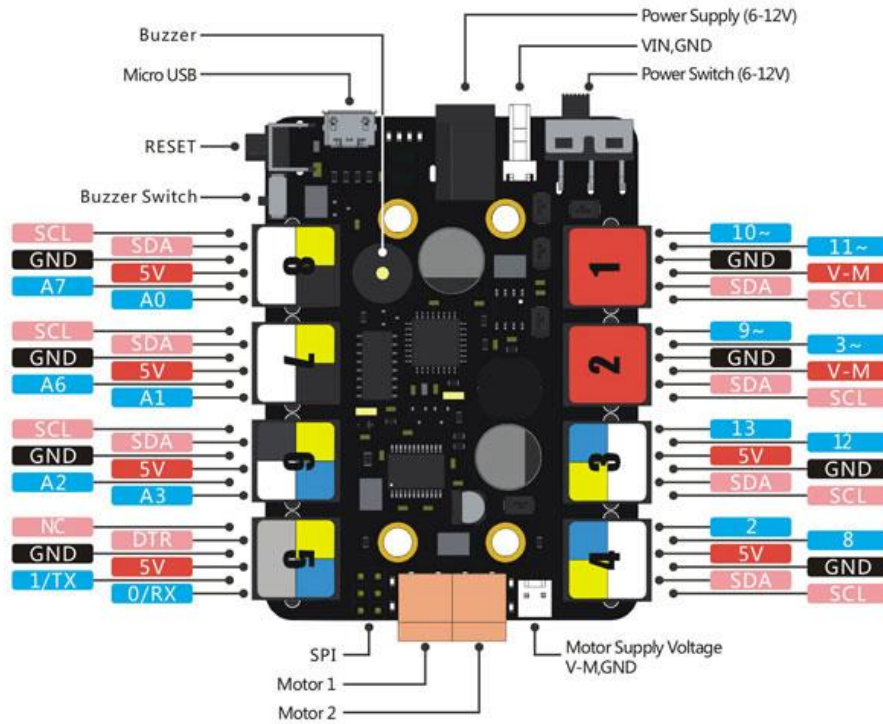
Şekil 5.4. meOrion Kontrol Kartı



Şekil 5.5. Step Motor Sürücü Kartı

Mdrawbot kiti içinde kontrol ünitesi olarak Arduino UNO bootloader içeren mOrion bulunmaktadır. Bu kontrol ünitesi Arduino IDE kullanılarak rahatlıkla programlanabilmektedir. Ayrıca karta özel kütüphane ve örnek uygulamalarda bulunmaktadır.

Mdrawbot kiti içinde kontrol ünitesi olarak Arduino UNO bootloader içeren mOrion bulunmaktadır. Bu kontrol ünitesi Arduino IDE kullanılarak rahatlıkla programlanabilmektedir. Ayrıca karta özel kütüphane ve örnek uygulamalarda bulunmaktadır. Şekil 5.6'da bu kartın pin yapısı görülmektedir.



Şekil 5.6. mOrion Kontrol Kartı Genel Görünümü ve Pin Yapısı

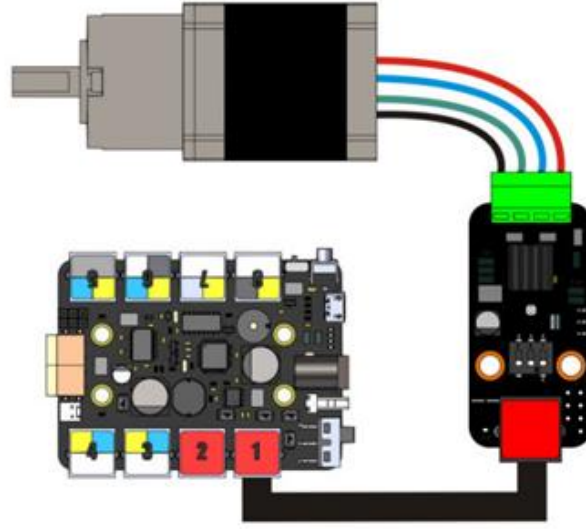
Aracımızın enerjisini sağlamak için iki adet 5V 2.1A çıkışa sahip powerbank seri bağlanarak 10V'luk bir gerilim elde edilmiştir. 10V kullandığımız stepper motorlar için ideale yakın bir gerilimdir.

Adım motorlar sargılarından birinin enerjilenmesi ile 1 adım hareket eden motorlardır. Bu adımın kaç derece olacağı motorun yapısına göre değişebilir. Bizim uygulamada kullandığımız stepper motorlar 200 adımlık stepper motorlardır. Bunun anlamı 200 adımda motor bir tam dönme yapacaktır yani 360 derecelik bir dönme için 200 adım gereklidir. Sonuç olarak her adım 1.8 derecelik bir rotasyona sahiptir. Bir tam dönme için ne kadar çok adım gerekirse stepper motorun hassasiyeti o kadar iyidir. Şekil 5.7'de stepper motorumuzun genel görünümü görülmektedir.

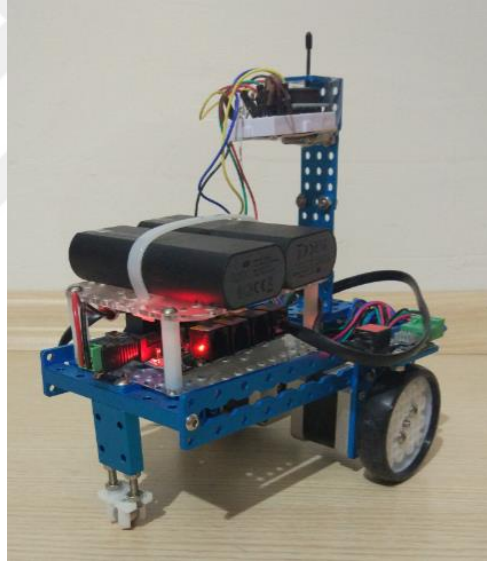


Şekil 5.7. Çalışmada Kullanılan Stepper Motor

Şekil 5.8'de Kontrol kartımız ile adım motorumuzun sürücü kartı aracılığıyla bağlantısı görülmektedir. Şekil 5.9'da ise aracımızın tamamlanmış hali görülmektedir.



Şekil 5.8. Kontrol Kartı, Step Motor Sürücü Kartı ile Step Motor Bağlantı Şeması



Şekil 5.9. Aracımızın Genel Görünümü

Aracımızda kullanılan step motorlarla kaç adımda ne kadar mesafe kat ettiğimizi hesaplayabilmemiz için aracımızda kullanılan teker çevresini ve her stepper motor adımında kaç derecelik bir dönme olduğunu bilmemiz gerekmektedir. Aracımızın hassasiyetini artırmak için motor sürücü kartımızda micro step mod olarak 1/16 step seçilmiştir. Bunun anlamı, 200 adım olan adım sayımız $200 \cdot 16 = 3200$ adım olarak hassasiyeti artırılmış ve 5.1, 5.2, 5.3 denklemleri yardımıyla adım sayısı hesaplanmıştır.

$$\text{Adım Başı Alınan Mesafe} = \text{Teker Çevresi} / 3200 \quad (5.1)$$

$$\text{Alınacak Mesafe} = (\text{Teker Çevresi} / 3200) * \text{Adım Sayısı} \quad (5.2)$$

$$\text{Adım Sayısı} = (3200 * \text{Alınacak Mesafe}) / \text{Teker Çevresi} \quad (5.3)$$

Teker çevremiz yapılan ölçümde yaklaşık olarak 20.3 cm olarak ölçülmüştür. 60 cm’lik bir hareket için 5.1, 5.2 ve 5.3 denklemleri kullanılarak, adım sayısı 9456 olarak hesaplanmıştır.

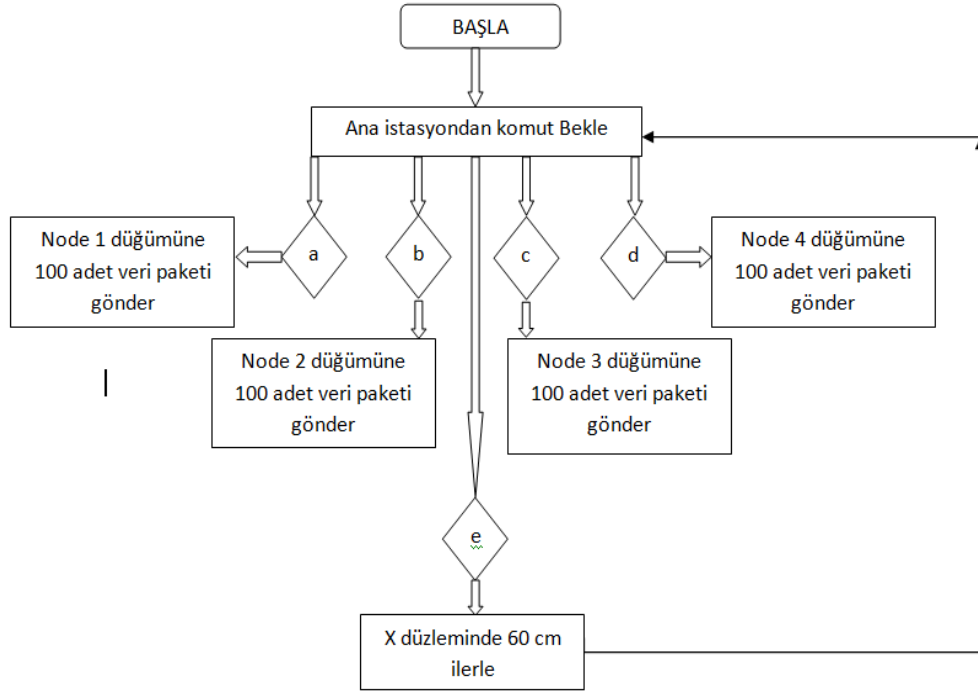
Çıkan sonuca göre her 60 cm’lik doğrusal hareket için her iki stepper motorun 9456 adım atması gerekmektedir. Bu adım sayısı aracımızın programlanması aşamasında kullanılmıştır.

Aracımıza, hem uzaktan kontrol amacıyla hem de köşe düğümlere veri göndermek amaçlı bir adet XBee modül eklenmiştir. XBee modül, kontrol kartının 5. Portunda bulunan UART pinleriyle (rx,tx), XBee modülün UART pinlerine (tx,rx) seri olarak bağlanmıştır. Ayrıca XBee modül 3.3V’luk bir gerilime ihtiyaç duyduğu için harici 3.3 V’luk bir regülatör entegresi üzerinden beslenme sağlanmıştır.

Aracımızın asli görevi bilgisayardan Xbee modül aracılığı ile gelen komutları, araçta bulunan Xbee modül aracılığı ile alıp gerekli işlemleri yerine getirmektir. Bu komutlar “a”, “b”, “c”, “d” ve “e” komutlarıdır. “a” komutu geldiği zaman aracımız XBee modül aracılığı ile düğüm 1’e içeriği önemli olmayan bir veri paketi yollar. Aynı şekilde “b”, “c” ve “d” komutları için ise sırasıyla düğüm 2, düğüm 3 ve düğüm 4’e içeriği önemsiz bir veri paketi gönderilir. Buradaki amaç veri paketinin ulaştığı düğüm ile araç arasındaki sinyal gücünün (RSSI) hesaplanabilmesini sağlamaktır.. Son komut olan “e” komutu iletildiği zaman ise aracımızın 60 cm ilerlemesi sağlanmaktadır.

Aracımız için yazılan arduino programının akış diyagramı Şekil 5.11’deki gibidir.

Yazılan Arduino programında, kontrol devremizin pin yapılarının tanımlandığı “MeOrion.h” ve XBee modül ile seri iletişim sağlamak amaçlı “XBee.h” olmak üzere iki adet hazır kütüphane kullanılmıştır. Aracımızda bulunan XBee modüle kontrol devremizden yollayacağımız çerçeve yapısını oluşturmak için XCTU yazılımının içinde bulunan “XBee API Frame Generator” aracı kullanılmıştır. Zigbee çerçevemizin yapısı Şekil 5.10’daki gibidir.



Şekil 5.10. Aracımızın Kontrol Akış Diyagramı

Protocol: All Mode: API 1 - API Mode Without Escapes

Frame type: 0x10 - Transmit Request

Frame parameters:

- Start delimiter: 7E
- Length: 00 12
- Frame type: 10
- Frame ID: 01
- 64-bit dest. address: 00 13 A2 00 41 25 75 1E
- 16-bit dest. address: FF FE
- Broadcast radius: 00
- Options: 00
- RF data: ASCII HEX, rssi

Generated frame:

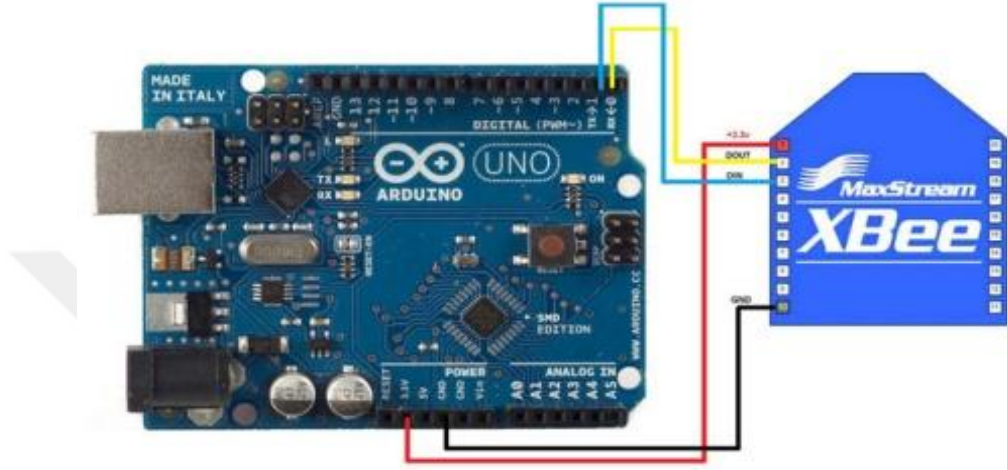
7E 00 12 10 01 00 13 A2 00 41 25 75 1E FF FE 00 00 72 73 73 69 82

Byte count: 22

Şekil 5.11. Aracımızdan Adresi Belirli Bir Düğüme Gönderilen Veri Paketi Çerçeve Yapısı

5.4. Üç Adet Köşe Düğümün Hazırlanıp Programlanması

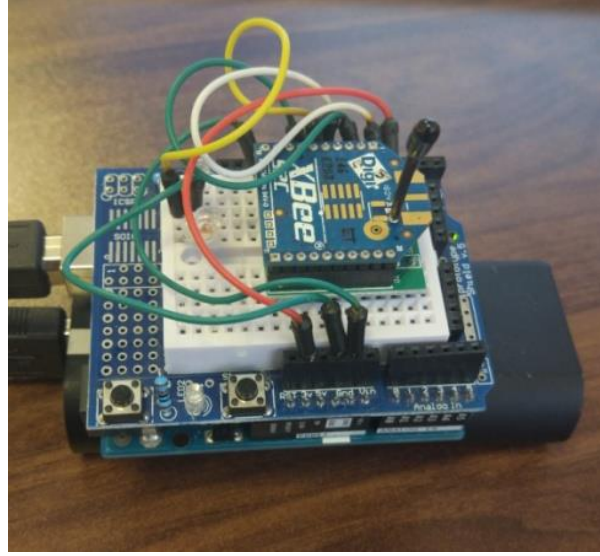
Oluşturulan bu düğümlerin görevleri, Hareketli araçtan gönderilen ve XBee modüle gelen son verinin RSSI değerini hesaplayıp Base Station olarak görev yapan 4. köşe düğüme iletmektir. Bu amaçla oluşturulan düğümlerin Xbee ve Arduino bağlantıları Şekil 5.12’deki gibidir.



Şekil 5.12. Arduino Uno ve XBee Modül Bağlantısı

Köşe düğümlerde kullanılan XBee modülleri, uygulamada kullanılan tüm XBee modüllerde olduğu gibi API modu aktif ve API çıkış modu ise “native” olacak şekilde XCTU aracı ile ayarlanmıştır. Bu ayarlamaların amacı XBee modüllere gelen veri paketlerini direk olarak arduino kontrol kartına, UART seri port üzerinden iletmektir. Arduinoya UART üzerinden veri paketi iletildiği zaman arduino, XBee modüle “DB” AT komutu göndererek RSSI değerini hesaplaması emrini verir. XBee modül bu komutun gereğini yerine getirerek, arduinoya hesapladığı RSSI değerini iletir. Arduino data olarak RSSI değerini adres olarak ise Base Station olan 4. Köşe düğümün adresini içeren bir veri paketi oluşturarak XBee modüle iletir ve RSSI değerinin Base Station düğüme iletilmesi sağlanır. Bu görevleri yerine getiren üç adet köşe düğüm oluşturulmuş ve çalışma alanı olarak seçilen odanın köşelerine yerleştirilmiştir. Ek 9’da bu görevleri yerine getiren arduino kodları verilmiştir.

Şekil 5.13’de 5 voltluk güç kaynağı, arduino ve XBee modülden oluşan köşe düğümlerden biri görülmektedir.



Şekil 5.13. Köşe Düğümün Genel Görünümü

5.5. 4. Köşe Düğümün (Base Station) Hazırlanıp Bu Düğüm İle Bilgisayar Haberleşmesini Sağlayan Yazılımın Yazılması

Dördüncü köşeye yerleştirilmek XBee ve bilgisayar birleşimi olan bir düğüm tasarlanmıştır. XBee modülü diğer düğümlerde olduğu gibi API modu aktif ve API çıkış modu ise “native” olacak şekilde XCTU aracı ile ayarlanmıştır. Diğer üç köşe düğümde Xbee modüle gelen veriler kontrol birimi olarak arduino’ya uart portu üzerinden aktarılırken bu düğümde gelen veriler USB port üzerinden bilgisayara aktarılmaktadır.

Bu düğümün dört ayrı görevi vardır. Bunlar aşağıdaki gibidir.

- Araca bir sonraki konuma geçmesi için hareket et komutu yani e komutunu göndermek
- Aracın ayrı ayrı tüm köşe düğümlere içeriği önemsiz veri paketi göndermesi için a, b, c, d komutlarını göndermek
- Köşe düğümlerden gelen RSSI değerlerini veri tabanında ilgili tabloya o anki araç konumuyla kayıt etmek.
- Kendi üzerine, araçtan gelen sinyalin RSSI değerini “DB” AT komutu yardımıyla hesaplayıp veri tabanına kayıt etmek.

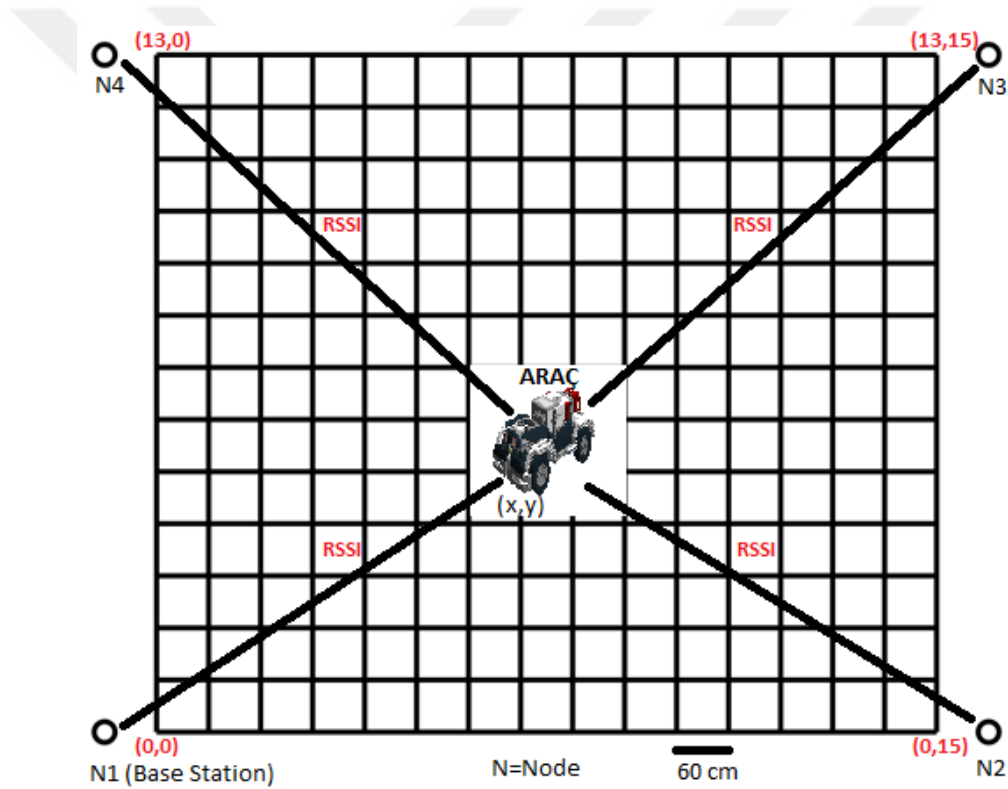
Bahsi geçen görevlerin yerine getirilmesi için bilgisayarda Netbeans IDE programlama ortamında, java programlama dili kullanılarak bir yazılım geliştirilmiştir. Bu yazılımın

geliştirilirken Digi firmasının XBee java kütüphaneleri kullanılmıştır. Yazılan java programının bir kısmı Ek 4’de verilmiştir.

5.6. Deney Ortamının Seçilmesi ve Uygulamanın Gerçekleştirilmesi

Düzgün zemini ve büyüklüğü gözü önüne alınarak, EÜAŞ Keban HES İşletme Müdürlüğü’nün teknik binasının 2. Katında bulunan eski Yük tevzi ana kumanda odası uygulama ortamı olarak seçilmiştir.

Oda zemini 60x60 cm’lik karolarla döşeli olduğundan dolayı ve aracımızın hatalı hareketlerini gözlemleyebilmek için çalışma ortamımız Şekil 5.14’deki gibi düşünülmüş, koordinatlar ve aracın hareketi buna göre uyarlanmıştır.



Şekil 5.14. Çalışma Ortamı ve Köşe Düzgümlerin Konumlarının Grafiksel Görünümü

Her karo köşesi bir koordinata denk gelmekte ve 16x14 koordinat noktası bulunmaktadır. N1,N2,N3,N4 köşe düğümlerdir. Ayrıca N1 basestation düğüm olup bilgisayara bağlıdır.

Uygulamamıza aracımız (0,0) noktasındayken başlanmıştır. İlk aşamada aracımız (0,0) noktasındayken bilgisayardan, bilgisayara bağlı base station olan 1. köşe düğüm aracılığıyla aracımıza 2. köşe düğüme içeriği önemli olmayan bir veri paketi göndermesi

komutu verilir. Veri araçtan, 1. köşe düğümüne vardığında, RSSI değeri bu düğüm üzerinde hesaplanarak, bilgisayara bağlı base station olan 1. Köşe düğümüne iletilir. Base stationa gelen RSSI değeri bilgisayara iletilerek bilgisayarda çalışmakta olan program aracılığıyla RSSI değeri ve o anki araç konumu MySQL veritabanında ilgili tabloya kayıt edilir. Bu adımlar base station dahil her köşe düğüm için 100 kez tekrar edilir. Toplamda, bir nokta için 400 RSSI değeri elde edilir. Daha sonra araca bilgisayardan 60 cm ilerle komutu gönderilerek aracın bir sonraki konuma geçmesi sağlanır ve bu adımlar tekrarlanır. Aracımız x konumunda 15 adım ilerleyip x düzleminde son noktaya vardığı zaman, araç x düzleminde ilk noktaya, y düzleminde ise bir birim yukarıya taşınır ve tüm adımlar aracımız 0,0 noktasından 15,13 noktasına gelene kadar tekrarlanır. Tablo 5.1’de örnek bir veri paketi onaltılık sayı sistemi olarak görülmektedir.

Tablo 5.1. Örnek XBee Veri Paketi

0x7E,0x00,0x12,0x10,0x01,0x00,0x13,0xA2,0x00,0x41,0x25,0x75,0x1E,0xFF,0xFE,0x00,0x00,0x72,0x73,0x73,0x69,0x82

Uygulamanın gerçekleştirildiği odanın genel görünümü şekil 5.15’de görüldüğü gibidir



Şekil 5.15. Uygulamanın Gerçekleştirildiği Odanın Genel Görünümü

Sonuç olarak aracımız 0,0 noktasından 15,13 noktasına vardığında her köşe düğüm için 16x14x100 yani 22400 RSSI değeri elde etmiş olduk.

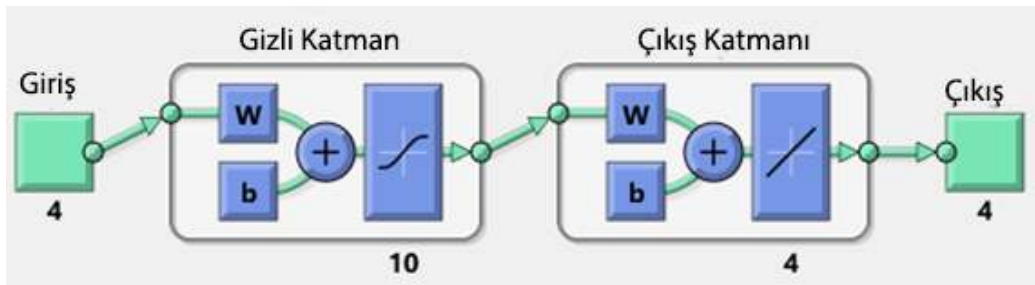
5.7. RSSI - Mesafe Kalibrasyonu ve Konum Bulma

Uygulama sonucu veri tabanımızda her bir köşe düğüm için oluşturulmuş her bir tabloda 22400 RSSI ve x, y konumları kayıt edildi. İlk aşamada koordinatların köşe düğümlere olan mesafeleri hipotenüs teoremi kullanılarak hesaplandı.

Veri tabanımızdaki veriler, hesaplamayla bulduğumuz mesafeler ve x,y koordinatları matlab ortamına import etmek amacıyla üç ayrı excell tablolarına aktarılmıştır. Rssi tablosu her sütun bir köşe düğümün RSSI değerini gösteren 4 adet sütundan (22400x4), mesafe tablosunun her bir sütunu bir köşe düğümün mevcut koordinata olan uzaklığını, konum tablosu ise x,y koordinatlarını içermektedir. Bu tablolar matlab’de import data tool’su kullanılarak numeric matrix olarak import edilmiştir.

Mesafe kalibrasyonu için matlab’in Network Fitting Tool’u (nftool) kullanılmıştır. Bu tool sayesinde yapay sinir ağının giriş - çıkış sayısı, gizli katman sayısı, eğitim algoritması ve eğitim için giriş - çıkış verileri kolaylıkla seçilip ağın eğitilmesi sağlanmaktadır.

Nftool’u kullanarak mesafe kalibrasyonu için oluşturduğumuz yapay sinir ağımız 10 gizli katmanlı, 4 rssi girişli, 4 adet mesafe çıkışlı ve eğitim algoritmamız Bayesian Regularization olacak şekilde oluşturulmuştur. Mesafe kalibrasyonun dört köşe düğüm için ayrı ayrı ama tek bir yapay sinir ağı kullanılarak yapılması amaçlanmış ve bunun tek bir rssi – mesafe kalibrasyonuna nazaran konum bulmada daha kesin sonuçlar elde etmeye yarayacağı öngörülmüştür. Tasarlanan yapay sinir ağımızın temsili görünümü Şekil 5.16’de görülmektedir.

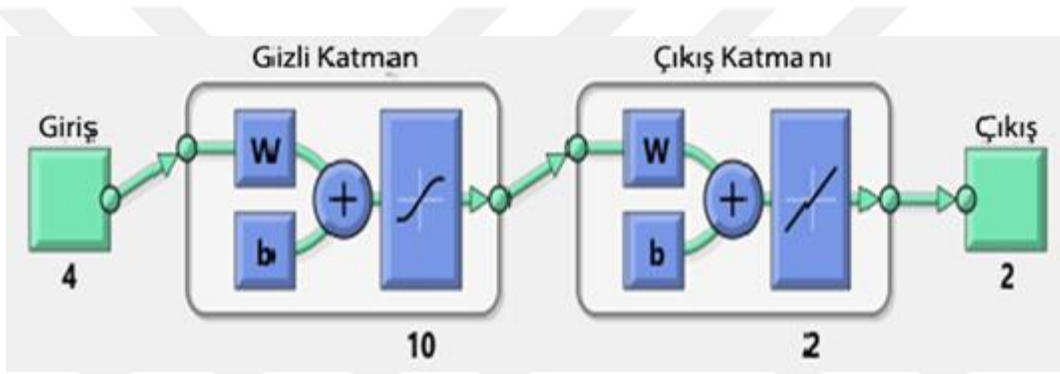


Şekil 5.16. Mesafe Kalibrasyonu Amaçlı Oluşturulan Yapay Sinir Ağımızın Genel Görünümü

Yapay sinir ağımız, daha önce import ettiğimiz rssi ve mesafe tablolarımızla, bu

verilerin %70'i eğitim %15'i test ve geri kalan %15'i ise doğrulama olacak şekilde eğitildikten sonra nftool aracı bize yapay sinir ağının ağırlıklarını da içeren bir yapay sinir ağı fonksiyonu üretmektedir. Bu fonksiyonumuzun adını daha sonra kullanabilmek amacıyla Network1 olarak değiştirip kayıt ettik. Ek 7'de bu fonksiyona ait kodlar verilmiştir.

İkinci aşamada RSSI – Mesafe kalibrasyonu sonucu elde edeceğimiz mesafelerden konum bulmak amacıyla ikinci bir yapay sinir ağı yine nftool aracı kullanılarak tasarlanmıştır. Bu yapay sinir ağımız 4 mesafe girişli ve çıkış olarak 2 (x,y) koordinat çıkışlı olacak şekilde tasarlanmıştır. Ek 7'de bu fonksiyona ait kodlar verilmiştir. Tasarlanan yapay sinir ağımızın temsili görünümü Şekil 5.17'de görülmektedir.



Şekil 5.17. Mesafe Kalibrasyonu Amaçlı Oluşturulan Yapay Sinir Ağımızın Genel Görünümü

Yine bu yapay sinir ağımız, daha önce import ettiğimiz mesafe ve x,y koordinat tablolarımızla, bu verilerin %70'i eğitim %15'i test ve geri kalan %15'i ise doğrulama olacak şekilde eğitilmiştir. Sonuç olarak elde edilen yapay sinir ağı fonksiyonumuzun adını daha sonra kullanabilmek için Network2 olarak değiştirip kayıt ettik. Ek 8'de bu fonksiyona ait kodlar verilmiştir.

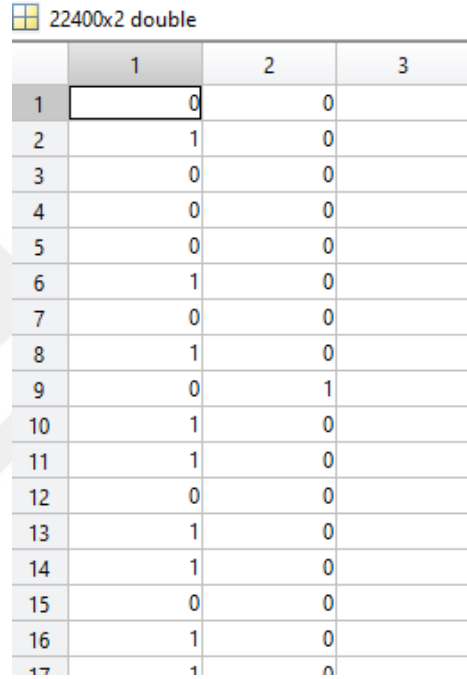
Her iki yapay sinir ağımızı eğitip oluşturduktan sonra, oluşturulan ağ fonksiyonlarından Network1'e daha önce matlab'e import ettiğimiz sahadan elde edilen ve konumları bilinen rssi tablomuzu giriş olarak girdik çıkış olarak elde ettiğimiz mesafeleri ise Network2 ağ fonksiyonumuza giriş olarak girdiğimiz zaman sonuç olarak 22400x2'lik bir koordinat matrisi elde ettik. Bu tablo sahada gerçek konumlarını bildiğimiz RSSI değerlerinin, geliştirmiş olduğumuz YSA'larca tahmin edilen konumlarıdır. Şekil 5.18'de matlab ortamında fonksiyonlarımızın çalıştırma komutları görülmektedir.


```
>>resultn=round(abs(Network2(Network1(rssi))))
```

Şekil 5.18. Matlab Ortamında Konum Bulma Komutumuz

Şekil 5.19’da görülen kod sayesinde hesaplanan 22400x2 ‘lik tahmini x ve y koordinatları, “resultn” değişkenine aktarılmıştır. Bu değişkenin tablo gösterimi Tablo 5.2’deki gibidir. Bu tablodaki birinci sütun x koordinatını ikinci sütun ise y koordinatını göstermektedir.

Tablo 5.2. Bulunan Tahmini Koordinat Tablosu



	1	2	3
1	0	0	
2	1	0	
3	0	0	
4	0	0	
5	0	0	
6	1	0	
7	0	0	
8	1	0	
9	0	1	
10	1	0	
11	1	0	
12	0	0	
13	1	0	
14	1	0	
15	0	0	
16	1	0	
17	1	0	

Tahmin edilen konumlarla gerçek konumlar arasındaki hata oranlarını hesaplamak amacıyla Matlab ortamında bir fonksiyon yazılmıştır. Bu fonksiyon tahmin edilen koordinatların gerçek konumdan ne kadar uzak olduğunu hesaplamakta ve sonuçları 0 metre, 0.6 metre ve 0.8485 metre olacak şekilde gruplamaktadır. Fonksiyonumuzun adı ErrorCount olarak belirlenmiştir ve giriş olarak, tahmin edilen koordinatlar ile gerçek koordinatlar olacak şekilde ayarlanmıştır. ErrorCount fonksiyonumuz Ek 5’de verilmiştir. Kodumuzun çalıştırılmış hali Şekil 5.19’da görüldüğü gibi çalıştırılmış ve sonuç olarak 22400 noktadan, 2725 noktanın doğru tahmin edildiği, 6730 noktanın 0.6 metre hassasiyetle, 4665 noktanın ise 0.8485 metre hassasiyetle tahmin edildiği hesaplanıp gözlemlenmiştir. Geri kalan 8280 nokta ise gerçek noktalarından 0.8485 metreden daha uzakta tahmin edilmiştir.

```
>>ErrorCount(xy,resultn)
ans = 2725    6730    4665
```

Şekil 5.19. Yapay Sinir Ağı İle Bulunan Konumların Hata Oranlarını Bulan Fonksiyonun Çağırılması ve Sonucu

Yapay sinir ağlarımızla elde ettiğimiz konumların, konum bulmada sıkça kullanılan “Bounding Box” algoritmasına göre durumunu sınavabilmek için yine Matlab ortamında Bounding Box algoritmasını gerçekleştiren bir fonksiyon yazılmıştır. Ek 6’da bu fonksiyona ait kodlar verilmiştir. Bu fonksiyonumuz, RSSI – mesafe kalibrasyonu yapan Network1 isimli YSA’mızdan çıkış olarak aldığımız 22400x4’lük mesafe matrisini giriş olarak almakta, çıkış olarak ise tahmini koordinatları vermektedir. Fonksiyonumuz matlabde Şekil 5.20’de görüldüğü gibi çalıştırılmış ve sonuçlar “resultb” değişkenine aktarılmıştır.

```
>>resultb=BoundingBox(Network1(rssi'))
```

Şekil 5.20. Bounding Box algortması ile Konum Bulan Fonksiyonun Çağırılması

Bounding Box algoritması ile elde ettiğimiz koordinatların ErrorCount fonksiyonumuz ile hata oranları, Şekil 5.21’de görüldüğü gibi hesaplanmıştır. Sonuç olarak 22400 noktadan, 396 noktanın doğru tahmin edildiği, 1331 noktanın 0.6 metre hassasiyetle, 1163 noktanın ise 0.8485 metre hassasiyetle tahmin edildiği hesaplanıp gözlemlenmiştir. Geri kalan 19510 nokta ise gerçek noktalarından 0.8485 metreden daha uzakta tahmin edilmiştir.

```
>>ErrorCount(xy,resultb)
ans = 396    1331    1163
```

Şekil 5.21. Bounding Box Algoritması İle Bulunan Konumların Hata Oranlarını Bulan Fonksiyonun Çağırılması ve Sonucu

YSA ve Bounding Box algoritması ile bulunan sonuçlar ve hata oranları Tablo 5.3’de görülen tablodaki gibidir.

Tablo 5.3. Yapay Sinir Ağı ve Bounding Box Yöntemleri İle Elde Edilen Sonuçlar

Yöntem Adı	Hata=0	Hata=0.6	Hata=0.8485	Hata>0.08485	Toplam
Yapay Sinir Ağı	2725	6730	4665	8280	22400
Bounding Box	396	1331	1163	19510	22400



6. SONUÇ

Yapılan bu çalışmada sahadan, 4 köşe düğüme göre toplamda 22400x4 RSSI değeri toplanmış ve eğitimi bu RSSI değerleri ile yapılan bir YSA ile mesafe kalibrasyonu yapılmıştır. Sonraki aşamada köşe düğümlere olan uzaklıklardan yola çıkarak 4 mesafe girişli ve iki konum (x,y) çıkışlı bir YSA oluşturulup eğitimi sahadan elde edilen veriler ile gerçekleştirilmiştir. Konumunu bulmak istediğimiz aracın köşe düğümlere olan RSSI değerleri ilk YSA'ya uygulanıp çıkan mesafeler ikinci YSA'ya uygulanarak tahmini konumlar bulunmuştur. 22400*4 RSSI değeri ilk YSA'ya uygulanıp 22400*4 muhtemel uzaklıklar elde edilmiş daha sonrasında bu muhtemel uzaklıklar ikinci YSA'ya uygulanıp 22400 x,y konum elde edilmiştir. Ayrıca ilk YSA'dan elde edilen 22400*4 mesafe verileri Bounding Box algoritması kullanılarak konumlar tahmini olarak bulunmuştur. YSA ile elde edilen konumlar gerçek konumlarla karşılaştırıldığında 22400 konumdan 2725 tanesi tam doğru olarak 6730 tanesi 0,6 metrelik hata oranıyla, 4665 tanesi 0.8485 metrelik hata oranıyla, geri kalanlar ise 0.845 metreden daha fazla hata oranlarıyla tahmin edilmiştir. Bounding Box algoritması ile elde edilen konumlar ise 22400 konumdan 396 tanesi tam doğru olarak 1331 tanesi 0,6 metrelik hata oranıyla, 1143 tanesi 0.8485 metrelik hata oranıyla, geri kalanlar ise 0.8485 metreden daha fazla hata oranlarıyla tahmin edilmiştir. Çalışma sonucunda oluşturduğumuz YSA'ların RSSI ile konum bulmada klasik algoritmalardan BoundingBox'a göre daha iyi sonuçlar elde edilmiştir.

REFERANSLAR

- [1] Chang, C.Y., Sensor Networks: Evolution, Oppertunities and Challenges, Senior Proceedings of the IEEE, 91, 1247-1256, 2003.
- [2] M. Gerla et al., "The Mars Sensor Network: Efficent, Energy Aware Communications," Procendings of the IEEE Military Communications Conferance (MilCom'01): Communications for Network-Centric Operations-Creatingthe Information Force, McLean, VA, Vol. 1, Oct. 2001.
- [3] S. Lindsey et al., "Data Gathering in Sensor Networks Using the Energy DelayMetric," presented at the International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, San Francisco, CA Apr. 2001.
- [4] Resul DOĞAN, Ebubekir ERDEM, "Temperature and Humidty Control of theTunnels in the DAM Using Wireless Sensor Netwoks" INES 2015. IEEE 19th International Conference on Intelligent Engineering Systems. September 3-5,2015, Bratislava, Slovakia.
- [5] Mehmet Akif Çiftçi, Atilla Elçi "Yapay Zeka ile Kablosuz Algılayıcı Ağları Eniyileme" Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi (2017 Cilt:10-Sayı:2) -64-76.
- [6] Selma Dilek, Suat Özdemir "Sağlık Hizmetleri Sektöründe Kablosuz Algılayıcı Ağlar" BİLİŞİM TEKNOLOJİLERİ DERGİSİ, CİLT: 7, SAYI: 2, MAYIS 2014 p.7-19
- [7] Karwan Muhammed Meheden, Ebubekir Erdem, "Design and Implementation of the Mobile Fire Alarms System Using Wireless Sensor Networks" Fırat Universty Master Thesis, Computer Engineering, January-2017.
- [8] Sercan Vançin, Ebubekir Erdem, "Trafik İzleme Sistemlerinin Kablosuz Manyetik Sensörler Kullanılarak Gerçekleştirilmesi" Fırat Üniversitesi Yüksek Lisans Tezi, Mayıs-2016.
- [9] Shi-Feng QI, Yan-Hua LI "The design of Dam Safety Monitoring Sensor Network" International Journal of Digital Content Technology and its Applications (JDCTA) Vol.7,No.7,April 2013,pp.364-370.
- [10] KerriA.Stone "Network Support for Earthen Emberkment Dam Monitoring Using Wireless Seonsor Networks" PhD thesis, Depertment of Electrical Engineering and Computer Science, Colorado School of Mines, 2013.

- [11] M. Welsh, D. Malan, B. Duncan, T. Fulford-Jones, S. Moulton, "Wireless Sensor Networks for Emergency Medical Care," presented at GE Global Research Conference, Harvard University and Boston University School of Medicine, Boston, MA, Mar. 8, 2004.
- [12] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., Cayirci, E., "Wireless Sensor Networks-A Survey", Elsevier Computer Networks, Vol. 38, 393-422, Mart 2002.
- [13] http://www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf
- [14] <https://www.slideshare.net/ManuelFernandezSantoT/20130918-00-introduction-to-wsn>
- [15] L. Hu, D. Evans, "Localization for Mobile Sensor Networks," Proceedings of the 10th ACM International Conference on Mobile Computing and Networking (MobiCom 04), Philadelphia
- [16] Nabil Ali Alrajeh, Maryam Bashir, Bilal Shams, "Localization Techniques in Wireless Sensor Networks", International Journal of Distributed Sensor Networks, Volume: 9 issue: 6
- [17] Kul G., Özyer T., Tavli B., 2014 , "IEEE 802.11 WLAN Based Real Time Indoor Positioning", Literature Survey and Experimental Investigations, 9th International Conference on Future Networks and Communications (FNC-2014).
- [18] https://www.researchgate.net/figure/RSSI-as-function-of-distance_fig6_230833800
- [19] Ademuwagun, A. and Fabio, V. (2017) Reach Centroid Localization Algorithm. Wireless Sensor Network, 9, 87-101
- [20] He, T., Huang, C., Blum, B.M., Stankovic, J.A. and Abdelzaher, T. (2003) Range Free Localization Schemes for Large Scale Sensor Networks. Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, San Diego, 14-19 September 2003, 81-95.]
- [21] Öztemel, Ercan. Yapay Sinir Ağları/Ercan Öztemel, -İstanbul: Papatya Yayıncılık Eğitim, 2012, ISBN:978-975-6797-39-6.
- [21] Neural networks and learning machines / Simon Haykin.—3rd ed. p. cm. Rev. ed of: Neural networks. 2nd ed., 1999. Includes bibliographical references and index. ISBN-13: 978-0-13-147139-9 ISBN-10: 0-13-147139-2
- [23] <https://www.mathworks.com/help/nnet/gs/fit-data-with-a-neural-network.html>

EKLER

Ek 1- XbeePin Yapısı

Figure 1-03. XBee®/XBee-PRO® RF Module Pin Numbers

(top sides shown - shields on bottom)

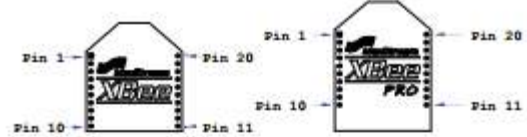


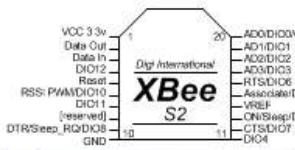
Table 1-02. Pin Assignments for the XBee and XBee-PRO Modules

(Low-asserted signals are distinguished with a horizontal line above signal name.)

Pin #	Name	Direction	Description
1	VCC	-	Power supply
2	DOUT	Output	UART Data Out
3	DIN / <u>CONFIG</u>	Input	UART Data In
4	DO8*	Output	Digital Output 8
5	<u>RESET</u>	Input	Module Reset (reset pulse must be at least 200 ns)
6	PWM0 / RSSI	Output	PWM Output 0 / RX Signal Strength Indicator
7	PWM1	Output	PWM Output 1
8	[reserved]	-	Do not connect
9	<u>DTR</u> / <u>SLEEP_RQ</u> / DI8	Input	Pin Sleep Control Line or Digital Input 8
10	GND	-	Ground
11	AD4 / DIO4	Either	Analog Input 4 or Digital I/O 4
12	<u>CTS</u> / DIO7	Either	Clear-to-Send Flow Control or Digital I/O 7
13	<u>ON</u> / <u>SLEEP</u>	Output	Module Status Indicator
14	VREF	Input	Voltage Reference for A/D Inputs
15	Associate / AD5 / DIO5	Either	Associated Indicator, Analog Input 5 or Digital I/O 5
16	<u>RTS</u> / AD6 / DIO6	Either	Request-to-Send Flow Control, Analog Input 6 or Digital I/O 6
17	AD3 / DIO3	Either	Analog Input 3 or Digital I/O 3
18	AD2 / DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1 / DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0 / DIO0	Either	Analog Input 0 or Digital I/O 0

* Function is not supported at the time of this release

Ek 2- Xbee Referans Doküman



Xbee S2

VCC 3.3V
Data Out
Data In
DIO12
Reset
RSS: PWM/DIO10
DIO11
(Reserved)
DTR/Sleep_RQ/DIO8
GND

AD0/DIO3/CB
AD1/DIO1
AD2/DIO2
AD3/DIO3
RTS/DIO6
Assesdate/DIO5
VREF
CNS/Sleep/DIO9
CTS/DIO7
DIO4

XBee S2 Quick Reference Guide

IEEE 802.15.4 = Zigbee Protocol. XBee is a microcontroller made by digi which uses the Zigbee protocol.

The XBee uses 3.3V and has a smaller pin spacing than most breadboards/proto boards. Because of this, it is often useful to purchase a kit to interface the XBee with a breadboard.

Sept/2012 <http://tunnelsup.com>

Specs

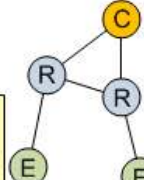
- Operating Voltage: 2.1 – 3.6V
- Operating Current: 40mA@3.3V
- Indoor range: 40 Meters
- Line of sight range: 120 Meters
- Max Analog Pin Reading: 1.2V

Digital I/O pins: 11

- Analog input pins: 4
- Mesh routable
- Self Healing network
- Firmware: ZB ZigBee

RF Data Rate: 250kbps

- Throughput speed: 35kbps
- Frequency: ISM 2.4GHz
- OK Temp: -40 to 85C



XBee Roles

- Coordinator** – 1 required in every network
In charge of setting up the network
Can never sleep
- Router** – multiple may exist
Can relay signals from other routers/EPs
Can never sleep
- End Point** – multiple may exist
Cannot relay signals
Can sleep to save power

XBee Modes

Transparent – Communication through the XBee. If data is not generated from the XBee itself then both XBees should be set to AT.

Command – Communication to the XBee. If one XBee is sensing data, that XBee should be in AT mode while the receiving one should be in API mode.

XBee Setup

Connect the XBee to a TTL Serial FTDI adapter – OR – Arduino hack: Connect RX to RX, TX to TX, RESET to ground to bypass the Arduino entirely and get serial to XBee.

Use the free X-CTU software to configure the XBee.

Baud: 9600 – FC: Hardware – Data Bits: 8 – Parity: None – Stop Bits: 1

Basic Settings

PAN ID – The network to communicate over. If 0, the XBee will join any.

DH/DL – Destination Serial number. Used to send to a specific XBee's Serial. Set to 0 to send to just the Coordinator. Set to 0x0000000000FFFF to broadcast.

JV – Router/EP should be set to 1 so it rejoins the network on startup

Pin Settings

For pin settings to work, receiver XBee must be in API mode

D0 – Set pin 0 to start sensing

IR – Collect data on sensing pins every XX millisecs

API format for Remote AT Command Request

Byte	Example	Description
0	0x7e	Start byte – Indicates beginning of data frame
1	0x00	Length – Number of bytes (ChecksumByte# – 1 – 2)
2	0x10	
3	0x17	Frame type - 0x17 means this is a AT command Request
4	0x52	Frame ID – Command sequence number
5	0x00	64-bit Destination Address (Serial Number)
6	0x13	MSB is byte 5, LSB is byte 12
7	0xA2	
8	0x00	0x0000000000000000 = Coordinator
9	0x40	0x0000000000000000FFFF = Broadcast
10	0x77	
11	0x9C	
12	0x49	
13	0xFF	Destination Network Address
14	0xFE	(Set to 0xFFFF to send a broadcast)
15	0x02	Remote command options (set to 0x02 to apply changes)
16	0x44 (D)	AT Command Name (Two ASCII characters)
17	0x02 (2)	
18	0x04	Command Parameter (queries if not present)
19	0xF5	Checksum

Arduino Connectivity:

Arduino TX connects to XBee RX (Data in)

Arduino RX connects to XBee TX (Data out)

Arduino Integration:

Data sent to Serial.print() will go out TX port of Arduino which is then connected to the RX port of XBee. If XBee is in AT mode it will transmit it wirelessly. Data received from XBee will be sent to the Serial.

Arduino Example: Read an analog value using API

```
// Remote XBee: AT, Base XBee: API
if (Serial.available() >= 21) { // Make sure the frame is all there
  if (Serial.read() == 0x7E) { // 7E is the start byte
    for (int i = 1; i<19; i++) { // Skip ahead to the analog data
      byte discardByte = Serial.read();
    }
    int analogMSB = Serial.read(); // Read the first analog byte data
    int analogLSB = Serial.read(); // Read the second byte
    int analogReading = analogLSB + (analogMSB * 256);
  }
}
```

Arduino Example: Change the pin setting on a remote Xbee

```
// Remote XBee: AT, Base XBee: API
Serial.write(0x7E); // Sync up the start byte
Serial.write((byte)0x0); // Length MSB (always 0)
Serial.write(0x10); // Length LSB
Serial.write(0x17); // 0x17 is the frame ID for sending an AT command
Serial.write((byte)0x0); // Frame ID (no reply needed)
Serial.write((byte)0x0); // Send the 64 bit destination address
Serial.write((byte)0x0); // (Sending 0x00000000000000FFFF (broadcast))
Serial.write((byte)0x0);
Serial.write((byte)0x0);
Serial.write((byte)0x0);
Serial.write(0xFF);
Serial.write(0xFF);
Serial.write(0xFF); // Destination Network
Serial.write(0xFE); // (Set to 0xFFFF if unknown)
Serial.write(0x02); // Set to 0x02 to apply these changes
Serial.write('D'); // AT Command: D1
Serial.write('1');
Serial.write(0x05); // Set D1 to be 5 (Digital Out HIGH)
long checksum = 0x17 + 0xFF + 0xFF + 0xFF + 0xFE + 0x02 + 'D' + '1' + 0x05;
Serial.write( 0xFF - (checksum & 0xFF)); // Checksum
```

API format for I/O Data Sample RX Indicator

Byte	Example	Description
0	0x7e	Start byte – Indicates beginning of data frame
1	0x00	Length – Number of bytes (ChecksumByte# – 1 – 2)
2	0x14	
3	0x92	Frame type - 0x92 indicates this will be a data sample
4	0x00	64-bit Source Address (Serial Number)
5	0x13	MSB is byte 4, LSB is byte 11
6	0xA2	
7	0x00	
8	0x40	
9	0x77	
10	0x9C	
11	0x49	
12	0x36	Source Network Address – 16 Bit
13	0x6A	
14	0x01	Receive Opts. 01=Packet Acknowledged. 02=Broadcast packet
15	0x01	Number of sample sets. Always set to 1 due to XBEE limitations
16	0x00	Digital Channel Mask – Indicates which pins are set to DIO
17	0x20	
18	0x01	Analog Channel Mask – Indicates which pins are set to ADC
19	0x00	Digital Sample Data (if any) – Reads the same as Digital Mask
20	0x14	
21	0x04	Analog Sample data (if any)
22	0x25	There will be two bytes here for every pin set for ADC
23	0xF5	Checksum(0xFF - the 8 bit sum of the bytes from byte 3 to this byte)

Sleep Mode

Endpoints can sleep to save power. An endpoint that only wakes up every 5 minutes to send data may only be awake for 6 seconds a day.

SM – 4 = Cyclic sleep

SP – Sleep time (up to 28 secs)

SN – Number of sleep cycles

ST – Time awake

Pin I/O Options

- 0 – Disabled
- 1 – N/A
- 2 – ADC
- 3 – Digital IN
- 4 – Digital OUT, LOW
- 5 – Digital OUT, HIGH

Digital Ch Mask

First Byte
n/a n/a n/a D12 D11 D10 n/a n/a

Second Byte
D7 D6 D5 D4 D3 D2 D1 D0

Example:
0x00 0x0D = 0000 0000 0000 1101
Pins D3, D2 and D0

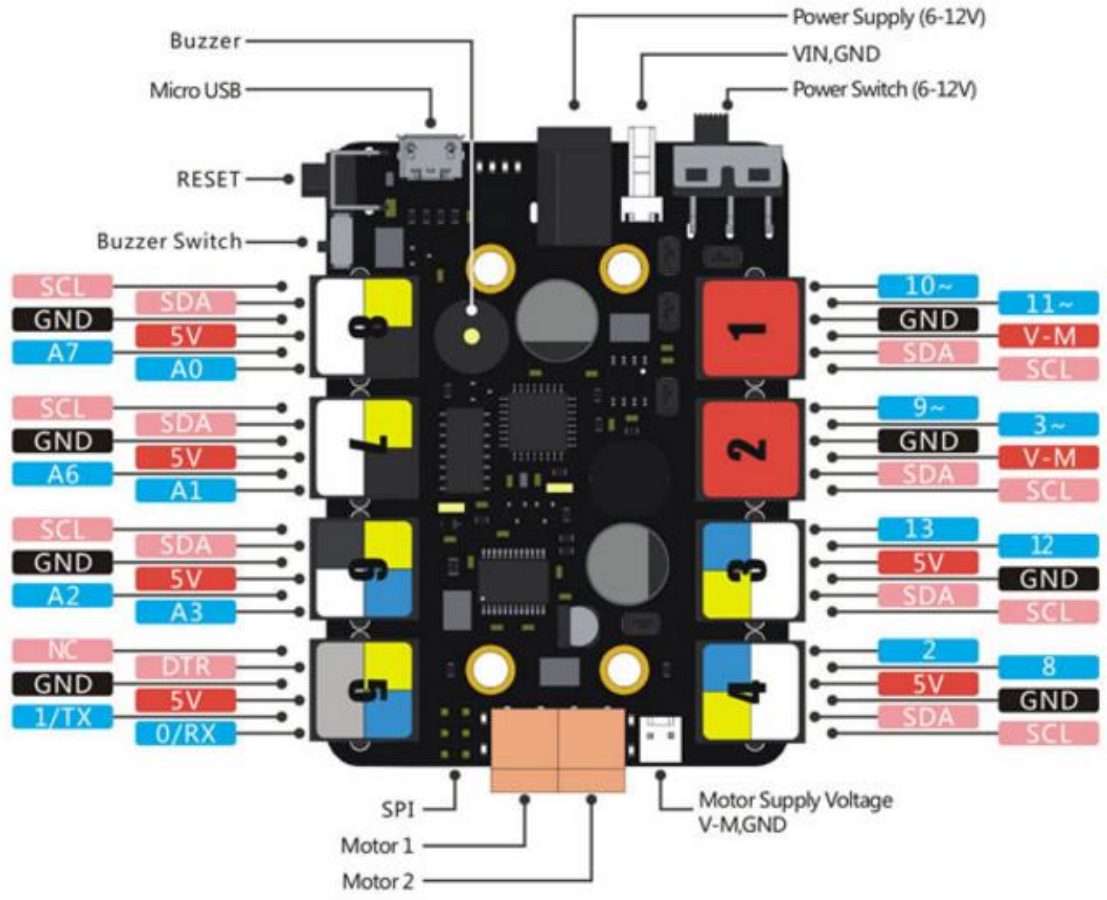
Analog Ch Mask

(volt) n/a n/a n/a A3 A2 A1 A0

Example:
0x05 = 0000 0101 = Pin A2 and A0

Notes

Ek 3- MeOrion Kontrol Kartı Pin Yapısı



Ek 4 - Sistem Kontrol Yazılımı

```
publicclassMainApp{
privatestatic final String PORT = "COM10";
privatestatic final int BAUD_RATE = 9600;
privatestatic final String DATA_TO_SEND_node_1 = "a";
privatestatic final String DATA_TO_SEND_node_2 = "b";
privatestatic final String DATA_TO_SEND_node_3 = "c";
privatestatic final String DATA_TO_SEND_node_4 = "d";
privatestatic final String DATA_TO_SEND_ROTATE_TEN_DEGREES = "t";
privatestatic final XBee16BitAddress XBEE_16BIT_ADDRESS = new
XBee16BitAddress("A4A7");
privatestatic final XBee64BitAddress XBEE_64BIT_ADDRESS = new
XBee64BitAddress("0013A2004125752C");

publicstaticvolatilerssi_data r = newrssi_data();
publicstaticvolatileXBeeDevicemyDevice = newXBeeDevice(PORT, BAUD_RATE);
publicstaticvoidinsert_DB(Stringrssi,Stringdist,Stringang)
throwsSQLException,ClassNotFoundException
{
Class.forName("com.mysql.jdbc.Connection");
Connection con = (Connection)
DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/calibration_rssi","root","qwe
123asd");
Stringquery = "insert intorssi_calibration(distance,angle,rssi)" +
"values("+dist+", "+ang+", "+rssi+")";
Statement stm = (Statement) con.createStatement();
intexecuteUpdate = stm.executeUpdate(query);
System.out.println(Integer.toString(executeUpdate)+"-Success");
}
publicstaticvoid main(String[] args) throwsInterruptedException, SQLException,
ClassNotFoundException {
System.out.println(" +-----+");
System.out.println(" | XBee Java Library Receive Data Sample |");
System.out.println(" +-----+\n");

try
{
myDevice.open();
RemoteXBeeDevicemyRemoteDevicee=new
RemoteXBeeDevice(myDevice,XBEE_64BIT_ADDRESS,XBEE_16BIT_ADDRESS,"12
");

MyDataReceiveListenerreceivelistener=newMyDataReceiveListener();
receivelistener.setXBeeDevice(myDevice);
myDevice.addDataListener(receivelistener);
receivelistener.x=0;
receivelistener.y=120;
```

```

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
for(int i=0; i<14; i++)
{
    for(int j=0; j<16; j++)
    {
        for(int k=0; k<100; k++){
            byte[] dataToSend_node_1 = DATA_TO_SEND_node_1.getBytes();
            myDevice.sendData(myRemoteDevicee, dataToSend_node_1);
            Thread.sleep(300);

            byte[] dataToSend_node_2 = DATA_TO_SEND_node_2.getBytes();
            myDevice.sendData(myRemoteDevicee, dataToSend_node_2);
            Thread.sleep(300);
            byte[] dataToSend_node_3 = DATA_TO_SEND_node_3.getBytes();
            myDevice.sendData(myRemoteDevicee, dataToSend_node_3);
            Thread.sleep(300);
            byte[] dataToSend_node_4 = DATA_TO_SEND_node_4.getBytes();
            myDevice.sendData(myRemoteDevicee, dataToSend_node_4);
            Thread.sleep(300);
        }
        Thread.sleep(1000);
        //byte[] dataToSend_EMPTY = DATA_TO_SEND_EMPTY.getBytes();
        //myDevice.sendData(myRemoteDevicee, dataToSend_EMPTY);
        br.readLine();
        receiverlistener.x = receiverlistener.x + 10;
    }
    br.readLine();
    receiverlistener.x = 0;
    receiverlistener.y = receiverlistener.y + 10;
}
} catch (XBeeException e) {
    e.printStackTrace();
    System.exit(1);
} catch (IOException ex) {
    Logger.getLogger(MainApp.class.getName()).log(Level.SEVERE, null, ex);
}
}
}

```

Ek 5- ErrorCount Fonksiyonu (Matlab)

```
function [err]= ErrorCount(x1,x2)
c=0;
d=0;
e=0;
for i=1:22400
if (x1(i,1)==x2(i,1)) && (x1(i,2)==x2(i,2))
c=c+1;
end
if ((abs(x1(i,1)-x2(i,1))==1) && (abs(x1(i,2)-
x2(i,2))==0)) || ((abs(x1(i,2)-x2(i,2))==1) && (abs(x1(i,1)-x2(i,1))==0))
d=d+1;
end
if (abs(x1(i,1)-x2(i,1))==1) && (abs(x1(i,2)-x2(i,2))==1)
e=e+1;
end
end
err=[c,d,e];
```

Ek 6 - Bounding Box Fonksiyonu (Matlab)

```
function [y1] = BoundingBox(x1)
y2=ones(22400,2)
for i=1:22400
x11=0;
x12=0;
x12=x11+x1(i,1)*100/60;
if x12>15
    x12=15;
end
if x12<0
    x12=0;
end
x21=15;
x22=0;
x22=x21-x1(i,2)*100/60;
if x22>15
    x22=15;
end
if x22<0
    y22=0;
end
x31=15;
x32=0;
x32=x31-x1(i,3)*100/60;
if x32>15
    x32=15;
end
if x32<0
    x32=0;
end
x41=0;
x42=0;
x42=x41+x1(i,4)*100/60;
if x42>15
    x42=15;
end
if x42<0
    x42=0;
end
y11=0;
y12=0;
y12=y11+x1(i,1)*100/60;
if y12>13
    y12=13;
end
if y12<0
    y12=0;
end
y21=0;
y22=0;
y22=y21+x1(i,2)*100/60;
if y22>13
    y22=13;
end
if y22<0
```

```

        y22=0;
    end
    y31=13;
    y32=0;
    y32=y31-x1(i,3)*100/60;
    if y32>13
        y32=13;
    end
    if y32<0
        y32=0;
    end
    y41=13;
    y42=0;
    y42=y41-x1(i,4)*100/60;
    if y42>13
        y42=13;
    end
    if y42<0
        y42=0;
    end
    ss=[x12,x22,x32,x42];
    qq=[y12,y22,y32,y42];
    s=sort(ss);
    q=sort(qq);
    y2(i,1)=round((s(2)+s(3))/2);
    y2(i,2)=round((q(2)+q(3))/2);
    end
    y1=y2;

```

Ek 7- Network1 Yapay Sinir Ağı Fonksiyonu (Matlab)

```
function [y1] =Network1(x1)
%MYNEURALNETWORKFUNCTION neural network simulationfunction.
%
% GeneratedbyNeural Network ToolboxfunctiongenFunction, 18-Dec-2017
09:55:41.
%
% [y1] = myNeuralNetworkFunction(x1) takesthesearguments:
%   x = 4xQ matrix, input #1
% and returns:
%   y = 4xQ matrix, output #1
% where Q is thenumber of samples.

%#ok<*RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
x1_step1_xoffset = [-61;-59;-62;-56];
x1_step1_gain =
[0.06666666666666667;0.0645161290322581;0.0740740740740741;0.0740740740740
741];
x1_step1_ymin = -1;

% Layer 1
b1 = [0.35778222104313939;0.20770601455113669;-
0.25893200109599068;0.96734686070291653;-
0.21384402169295652;0.85440353178877337;-0.78320851157861293;-
1.5633012772640933;-0.88646957969983708;0.37289363370149853];
IW1_1 = [-1.3279472020895773 -0.23246988620834774 1.8153705186778881 -
0.024615391751119441;1.6259328951408536 -0.13531046988739306 -
1.7292874947774612 -0.56913847277930618;-1.5219734441874739 -
0.13181055652696452 1.3082182997018568 0.26135687549843961;-
0.42401204689709354 -1.758452271456556 0.1174767023244828
2.6946533922024876;-1.0111410429712147 0.49211855613728211
0.54150280733794054 -0.5370639045958453;-0.04473887659276389 -
1.8394796806089844 0.47605450610179345
2.2055708468333606;0.1147735831419829 -2.0011571301802302
0.35392264783119143 0.33176188563852227;-2.7317083510752012 -
0.27621937889147868 0.20842889918984495 0.0042476187429890469;-
0.35980819437340866 -1.5635664981404933 0.099980625940713844
0.49072740436190104;-0.46624690940681146 0.86190334148330139 -
0.38287967718947175 0.48175015575769331];

% Layer 2
b2 =
[0.15226609519457771;0.3349652014853518;0.10051021886579516;0.11563002799
546297];
LW2_1 = [-1.3364549822532652 1.7239644784238544 4.0514407726891362 -
0.25862205668196631 0.8475924269750984 0.13264308940895161
0.41164382291454399 -0.82880818418712576 -0.013672895005055676
0.5684797575782119;-0.36340028489255 -0.016165918009673885
0.54111154341872703 -1.1029601277671444 -0.71382987796057651
1.4472477997342865 -1.2694911769957979 0.049324285663711109
1.2040255312790631 -0.41121493432111822;1.3195960009289414 -
1.8106544044472566 -4.1368839196552836 -0.16014049018001481 -
```

```

0.81460552967769362 -0.062602337330606828 0.29888806984673727

0.79958548045155897 -0.12219494935169145 0.30796733010601907;-
0.20338397310532175 -0.36999084819687478 -0.42794312374473731
0.92261323359091174 0.70061999592510593 -1.2576560602831175
1.274360615261847 0.058738734284508126 -1.3948377538667796 -
0.059135903266159383];

% Output 1
y1_step1_ymin = -1;
y1_step1_gain =
[0.169933397233398;0.169933397233398;0.167959297687721;0.167959297687721]
;
y1_step1_xoffset = [0.6;0.6;0.848528137423857;0.848528137423857];

% ===== SIMULATION =====

% Dimensions
Q = size(x1,2); % samples

% Input 1
xp1 = mapminmax_apply(x1,x1_step1_gain,x1_step1_xoffset,x1_step1_ymin);

% Layer 1
a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*xp1);

% Layer 2
a2 = repmat(b2,1,Q) + LW2_1*a1;

% Output 1
y1 = mapminmax_reverse(a2,y1_step1_gain,y1_step1_xoffset,y1_step1_ymin);
end

% ===== MODULE FUNCTIONS =====

% Map Minimum and Maximum InputProcessing Function
function y =
mapminmax_apply(x,settings_gain,settings_xoffset,settings_ymin)
y = bsxfun(@minus,x,settings_xoffset);
y = bsxfun(@times,y,settings_gain);
y = bsxfun(@plus,y,settings_ymin);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n)
a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum OutputReverse-Processing Function
function x =
mapminmax_reverse(y,settings_gain,settings_xoffset,settings_ymin)
x = bsxfun(@minus,y,settings_ymin);
x = bsxfun(@rdivide,x,settings_gain);
x = bsxfun(@plus,x,settings_xoffset);
end

```


Ek 8 - Network2 Yapay Sinir Ağı Fonksiyonu (Matlab)

```
function [y1] = myNeuralNetworkFunction(x1)
%MYNEURALNETWORKFUNCTION neural network simulationfunction.
%
% GeneratedbyNeural Network ToolboxfunctiongenFunction, 01-Dec-2017
09:33:35.
%
% [y1] = myNeuralNetworkFunction(x1) takesthesearguments:
%   x = 4xQ matrix, input #1
% and returns:
%   y = 2xQ matrix, output #1
% where Q is thenumber of samples.

%#ok<*RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
x1_step1_xoffset = [0.6;0.6;0.848528137;0.848528137];
x1_step1_gain =
[0.16993339718796;0.16993339718796;0.167959297615189;0.167959297615189];
x1_step1_ymin = -1;

% Layer 1
b1 = [0.97166113592396874;0.82404262817185747;-
0.77801647080966163;0.34510251202505993;0.34978811434991192;0.82846948958
279065;1.3767962619948413;1.3667041162317908;-
0.84908064270451722;1.5034849092119407];
IW1_1 = [-0.43815549048687086 -0.015859097779788507 -0.047378943411897877
0.013076094715022982;0.16356154788725508 -0.18057383828730378
0.090643752052132126 -0.45779560317636925;0.0690715713123181
0.054463197732132977 0.013463466421823162 0.34034989189856413;-
0.10187235502130673 -0.38634736257008256 -0.067416864717796871 -
0.15172862034186005;0.20752596958746491 0.13513827974387338
1.2302847480397452 0.14152851460163302;0.14508026223052278 -
0.53014735511352462 0.069830810057723194
0.075362693239439296;0.75608965971340125 -0.098562708812361566
0.05012229861546072 -0.13651794458415886;-0.11267086888199508
0.55610453782036573 -0.1075025939924957 -
0.034302465805053528;0.10777996873893204 -0.018582075818001982
0.62900210891299346 -0.0032751549747299814;-0.12665147797050083 -
0.70636645472211823 -0.078944473028924614 -0.072976320611129256];

% Layer 2
b2 = [0.58425207639054377;1.5434268328624186];
LW2_1 = [-1.3942525153034264 2.4706685984553762 -1.4108525137128844 -
2.7770305329660681 -0.11557206679159589 -0.017951206726894514 -
0.74127296424178768 -1.4667556169045359 -3.0513873156544511 -
1.4502367005806751;-4.1295127194554953 0.57112458744486883
2.6137759419628579 1.5422078213616506 -0.03680934375484806
0.78622680787870736 -0.66282338989109735 1.1966971986846855 -
0.46154985997313941 1.1564613629059795];

% Output 1
y1_step1_ymin = -1;
y1_step1_gain = [0.153846153846154;0.133333333333333];
```

```

y1_step1_xoffset = [0;0];

% ===== SIMULATION =====

% Dimensions
Q = size(x1,2); % samples

% Input 1
xp1 = mapminmax_apply(x1,x1_step1_gain,x1_step1_xoffset,x1_step1_ymin);

% Layer 1
a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*xp1);

% Layer 2
a2 = repmat(b2,1,Q) + LW2_1*a1;

% Output 1
y1 = mapminmax_reverse(a2,y1_step1_gain,y1_step1_xoffset,y1_step1_ymin);
end

% ===== MODULE FUNCTIONS =====

% Map Minimum and Maximum InputProcessing Function
function y =
mapminmax_apply(x,settings_gain,settings_xoffset,settings_ymin)
y = bsxfun(@minus,x,settings_xoffset);
y = bsxfun(@times,y,settings_gain);
y = bsxfun(@plus,y,settings_ymin);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n)
a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum OutputReverse-Processing Function
function x =
mapminmax_reverse(y,settings_gain,settings_xoffset,settings_ymin)
x = bsxfun(@minus,y,settings_ymin);
x = bsxfun(@rdivide,x,settings_gain);
x = bsxfun(@plus,x,settings_xoffset);
end

```

Ek 9 - Köşe Düğümler İçin Yazılan Arduino Programı

```
#include<XBee.h>           //http://code.google.com/p/xbee-arduino/
#include<SoftwareSerial.h>

//XBee variables
XBee xbee = XBee();        //XBee object
ZBRxResponse rx = ZBRxResponse(); //XBee receiver response
SoftwareSerial serial1(8, 9); // RX, TX
uint8_t payload[1]={0};
byte
redox[]={0x7E,0x00,0x0F,0x10,0x01,0x00,0x13,0xA2,0x00,0x41,0x25,0x75,0x1E,0xFF,0
xFE,0x00,0x00,0x41,0x02};
void setup(void)
{
  Serial.begin(9600);
  serial1.begin(9600);
  xbee.setSerial(serial1);
}

void loop(void)
{
  xbee.readPacket();

  if (xbee.getResponse().isAvailable()) {

    switch (xbee.getResponse().getApiId()) {      //what kind of packet did we get?

    case ZB_RX_RESPONSE:                          //rx data packet
      xbee.getResponse().getZBRxResponse(rx);    //get the received data
      bytes = rx.length();
      redox[sizeof(redox)-2]=rx[0];
      int cksum = 0; // start with a zero checksum
      for (byte i = 3; i < sizeof(redox)-1; i++) { // skip the Start and len, start with byte 4.
        // (remembering that C arrays start at 0, not 1.)
        cksum += redox[i]; // Add in this byte.
      } // next byte
      cksum &= 0xFF; // low order 8 bits
      cksum = 0xFF - cksum; // subtract from 0xFF
      redox[sizeof(redox)-1] = cksum;

      Serial.print(rx[0]);
      serial1.write(redox, sizeof(redox));

      switch (rx.getOption()) {
      case ZB_PACKET_ACKNOWLEDGED:
```

```

break;
    }
}
}

/*-----*
* returnsreceivedsignalstrengthvalueforthelast RF datapacket. *
*-----*/

byterssi() {
    /* returnsreceivedsignalstrengthvalueforthelast RF datapacket */

    union {byte B; char C;} atCmd[3];
    AtCommandRequestatCmdReq;
    AtCommandResponseatResp;
    byterespLen, *resp, dBm;

    strcpy(&atCmd[0].C, "DB");
    atCmdReq = AtCommandRequest(&atCmd[0].B);
    atResp = AtCommandResponse();

    xbee.send(atCmdReq);
    if (xbee.readPacket(5000)) {
        if (xbee.getResponse().getApiId() == AT_COMMAND_RESPONSE) {
            xbee.getResponse().getAtCommandResponse(atResp);
            if (atResp.isOk()) {
                respLen = atResp.getValueLength();
                if (respLen == 1) {
                    resp = atResp.getValue();
                    dBm = resp[0];
                }
                return dBm;
            }
        }
        else {
            Serial.println("Unexpectedresponse");
        }
    }
    else {
        Serial.println("ERROR");
    }
}
else {
    Serial.println("Unknownresponse");
}
}
else {
    Serial.println("No response");
}
}

```

```

}
uint8_t rss() {

    uint8_t atCmd[] = {'D', 'B'}, respLen, *resp, dBm;
    AtCommandRequest atCmdReq;
    AtCommandResponse atResp;

    atCmdReq = AtCommandRequest(atCmd);

    xbee.send(atCmdReq);
    if (xbee.readPacket(10)) {
        if (xbee.getResponse().getApiId() == AT_COMMAND_RESPONSE) {
            xbee.getResponse().getAtCommandResponse(atResp);
            if (atResp.isOk()) {
                respLen = atResp.getValueLength();
                if (respLen == 1) {
                    resp = atResp.getValue();
                    dBm = resp[0];
                    return dBm;
                }
            }
            else {
                //Serial << "RSS LEN ERR" <<endl; //unexpectedlength
            }
        }
        else {
            //Serial << "RSS ERR" <<endl; //status not ok
        }
    }
    else {
        //Serial << "RSS UNEXP RESP" <<endl; //expecting
        AT_COMMAND_RESPONSE, got something else
    }
}
else {
    //Serial << "RSS NO RESP" <<endl; //timeout
}
}

```

ÖZGEÇMİŞ

Resul DOĞAN

resul.dogan@euas.gov.tr

Elektrik Üretim Anonim Şirketi Keban HES İşletme Müdürlüğü
Keban/ELAZIĞ

1984 yılında Elazığ'da doğdum. Elazığ Anadolu Lisesi'ni 2002 yılında bitirdikten sonra, 2003 yılında Fırat Üniversitesi Bilgisayar Mühendisliği Bölümünü kazandım ve 2009 yılında mezun oldum. RMOS, Kardelen ve ELMAR yazılım şirketlerinde belirli süreler çalıştıktan sonra 2010 yılında EÜAŞ Atatürk HES İşletme Müdürlüğüne atandım ve şuan aynı şirkete bağlı Keban HES işletme Müdürlüğü Bilgi İşlem Şefliğinde, Bilgi İşlem Şefi olarak çalışmaktayım. Yazılım geliştirme, endüstriyel kontrol sistemleri ve haberleşme protokollerinde teknik bilgi sahibiyim. Orta düzey İngilizce bilgisine sahibim.