KAPALI MEKÂNDAKİ MOBİL CİHAZ KONUMLARININ İZLENMESİ İÇİN YEREL AĞ TABANLI BİR SİSTEM TASARIMI

Yiğit Batuhan TUNA
211402101

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı
Danışman: Prof. Dr. Emin Murat ESİN

İstanbul
T.C. Maltepe Üniversitesi
Lisansüstü Eğitim Enstitüsü
Eylül, 2025

KAPALI MEKÂNDAKİ MOBİL CİHAZ KONUMLARININ İZLENMESİ İÇİN YEREL AĞ TABANLI BİR SİSTEM TASARIMI

Yiğit Batuhan TUNA
211402101
ORCID: 0000-0001-6505-7698

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı
Danışman: Prof. Dr. Emin Murat ESİN

İstanbul
T.C. Maltepe Üniversitesi
Lisansüstü Eğitim Enstitüsü
Eylül, 2025



JÜRİ VE ENSTİTÜ ONAYI

YİĞİT BATUHAN TUNA'ın "Kapalı Mekandaki Mobil Cihaz Konumlarının İzlenmesi İçin Yerel Ağ Tabanlı Bir Sistem Tasarımı " başlıklı tezi 17.09.2025 tarihinde aşağıdaki jüri tarafından değerlendirilerek "Maltepe Üniversitesi Lisansüstü Eğitim ve Öğretim Yönetmeliği" nin ilgili maddeleri uyarınca Bilgisayar Mühendisliği Anabilim Dalı Yüksek Lisans/Doktora/Sanatta Yeterlik tezi oy birliğiyle/oy çokluğuyla, başarılı/başarısız olarak kabul edilmiştir.

	Unvanı. Adı ve Soyadı	İmza
Üye (Tez Danışmanı)	Prof. Dr. Emin Murat ESİN (Danışman) Maltepe Üniversitesi	.....
Üye	Dr. Öğr. Üyesi Mahsa MİKAEİLİ Okan Üniversitesi	.....
Üye	Dr. Öğr. Üyesi Rıfat Volkan ŞENYUVA Maltepe Üniversitesi	.....

Prof. Dr. Selva ERSÖZ KARAKULAKOĞLU
Enstitü Müdürü

ETİK İLKE VE KURALLARA UYUM BEYANI

Bu belge, Yükseköğretim Kurulu tarafından 19.01.2021 tarihli "Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge" ile bildirilen 6698 Sayılı Kişisel Verilerin Korunması Kanunu kapsamında gizlenmiştir.

TEŞEKKÜR

Yüksek lisans tez çalışmam süresince bana her zaman destek olan ve yanımda bulunan aileme, sabırları ve sevgileri için en içten teşekkürlerimi sunarım.

Tez sürecim boyunca değerli bilgi ve deneyimleriyle bana yol gösteren, her aşamada desteğini esirgemeyen danışmanım Prof. Dr. Emin Murat Esin hocama en derin şükranlarımı sunarım.

Ayrıca, araştırmam sırasında gerekli imkân ve kolaylıkları sağlayarak kapalı tenis kortunu ölçüm yapmam için bana tahsis ederek çalışmama katkıda bulunan Maltepe Üniversitesi Sağlık Kültür ve Spor Daire Başkanlığı Spor Tesisleri Sorumlusu Sayın Ahmet Kaya'ya da teşekkür ederim.

Bu süreçte bana destek olan tüm kişi ve kurumlara minnettarlığımı belirtmek isterim.

Yiğit Batuhan TUNA

Eylül, 2025

ÖZET

KAPALI MEKÂNDAKİ MOBİL CİHAZ KONUMLARININ İZLENMESİ İÇİN YEREL AĞ TABANLI BİR SİSTEM TASARIMI

Yiğit Batuhan Tuna
Yüksek Lisans Tezi
Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı
Danışman: Prof. Dr. Emin Murat Esin
Maltepe Üniversitesi Lisansüstü Eğitim Enstitüsü, 2025

Dünyada ve ülkemizde kapalı mekânlarda çeşitli kazalar ve kayıplar sıkça yaşanmaktadır. Özellikle büyük binalar, fabrikalar, alışveriş merkezleri, okullar, kamu kurumları ve hastaneler gibi ortamlarda insanların, canlıların ve ekipmanların hassas şekilde izlenmesi, kayıp durumunda yerlerinin hızlıca tespit edilmesi kritik öneme sahiptir. Günümüzde yaygın olarak kullanılan GPS teknolojisi, kapalı alanlarda sinyal kaybı nedeniyle etkin bir çözüm sunamamaktadır.

Bu çalışmanın amacı; mevcut konumlandırma teknolojilerinin kapalı mekânlarda karşılaştığı kısıtlamalara karşı çözüm bulmaktır. Tasarımın en önemli avantajı, karmaşık altyapı gerektirmeden yerel bir ağ üzerinden çalışabilmesi ve düşük gecikmeli gerçek zamanlı konum takibi yapabilmesidir.

Sistem, üç sabit alıcı (anchor) ve bir hareketli vericiden (TAG) oluşmakta olup, verici ile alıcılar arasındaki mesafe UDP protokolü üzerinden aktarılmaktadır. Konum belirleme işlemi için trilaterasyon (üçgenleme) yöntemi kullanılmaktadır. Sistemde veri iletiminin adil ve verimli bir şekilde yönetilmesi için Round Robin algoritması uygulanmıştır. Bu algoritma sayesinde tüm alıcılar eşit zaman dilimlerinde veri alışverişi yapabilmekte ve sistem kaynaklarının dengeli kullanımı sağlanmaktadır. Python programlama dili ve Tkinter kütüphanesi kullanılarak geliştirilen arayüz sayesinde hareketli vericinin ve alıcıların konumları sanal harita üzerinde gerçek zamanlı konum takibine olanak

tanımaktadır. Bu da acil durumlarda hızlı müdahale ve güvenlik sağlanmasına olanak vermektedir.

Geliştirilen bu sistem, özellikle acil durum müdahaleleri, endüstriyel alanlarda ekipman takibi ve sağlık kuruluşlarında hasta izleme gibi kritik uygulamalarda kullanılabilecek potansiyele sahiptir. Mevcut çözümlere kıyasla sunduğu yüksek doğruluk ve güvenilirlik, kapalı mekân konumlandırma problemlerine etkili bir çözüm getirmektedir.

Anahtar Sözcükler: Kapalı Mekân Konumlandırma, Acil Durum Müdahalesi, Güvenlik, UWB (Ultra Wide Band) Teknolojisi, UDP (User Data Protocol), Trilaterasyon, Round Robin Algoritması, Python Programlama Dili, Tkinter Kütüphanesi, Sanal Harita.

ABSTRACT

DESIGN OF A LOCAL LAN BASED SYSTEM FOR MONITORING MOBILE DEVICE LOCATIONS IN INDOOR AREAS

Yiğit Batuhan Tuna
Master's Thesis
Department of Computer Engineering
Computer Engineering Programme
Thesis Advisor: Prof. Dr. Emin Murat Esin
Maltepe University Graduate School, 2025

Accidents and losses frequently occur in enclosed spaces both worldwide and in our country. In particular, in environments such as large buildings, factories, shopping malls, schools, public institutions, and hospitals, the precise monitoring of people, living beings, and equipment, as well as the rapid detection of their locations in case of loss, is of critical importance. The commonly used GPS technology today cannot provide an effective solution in indoor environments due to signal loss.

The aim of this study is to provide a solution to the limitations faced by existing positioning technologies in indoor spaces. The most significant advantage of the proposed design is its ability to operate over a local network without requiring complex infrastructure, while offering low-latency, real-time location tracking.

The system consists of three fixed receivers (anchors) and one mobile transmitter (TAG), with the distance between the transmitter and receivers being transmitted via the UDP protocol. Trilateration (triangulation) is used to determine the position. A Round Robin algorithm is implemented to manage data transmission fairly and efficiently. This algorithm ensures that all receivers can exchange data at equal time intervals, thereby enabling balanced use of system resources. A user interface developed with the Python programming language and the Tkinter library allows real-time location tracking of the

mobile transmitter and receivers on a virtual map. This feature enables rapid response and enhanced safety during emergencies.

The developed system has the potential to be used in critical applications such as emergency response, equipment tracking in industrial environments, and patient monitoring in healthcare facilities. Compared to existing solutions, the high accuracy and reliability it offers provide an effective solution to the problem of indoor positioning.

Keywords: Indoor Positioning, Emergency Response, Security, UWB (Ultra Wide Band) Technology, UDP (User Datagram Protocol), Trilateration, Round Robin Algorithm, Python Programming Language, Tkinter Library, Virtual Map.

İÇİNDEKİLER

JÜRİ VE ENSTİTÜ ONAYI ii

ETİK İLKE VE KURALLARA UYUM BEYANI iii

TEŞEKKÜR iv

ÖZET v

ABSTRACT vii

İÇİNDEKİLER ix

TABLOLAR LİSTESİ xiv

ŞEKİLLER LİSTESİ xv

KISALTMALAR xvii

1. GİRİŞ 1

1.1 Problem 2

1.2 Amaç 3

1.3 Önem 3

1.4 Varsayımlar 5

1.5 Sınırlılıklar 5

1.6 Tanımlar 5

2. GENEL BİLGİLER 10

2.1 Konumlandırma Çeşitleri 12

2.1.1 Açık alan konumlandırması 12
2.1.2 Kapalı alan konumlandırması 12
2.2 Teorik Yaklaşımlar 13
2.2.1 Wifi iç mekân konumlandırması 13
2.2.2 Bluetooth iç mekân konumlandırması 15
2.2.3 Rfıd iç mekân konumlandırması 16
2.2.4 Uwb (ultra wide band) iç mekân konumlandırması 17
2.2.5 Ultrasonik iç mekân konumlandırması 18
2.3 Literatür Araştırması 18
3. YÖNTEM 21
3.1 ToA (Varış Zamanı) ile Mesafe Ölçümü 21
3.1.1 Toa ve tof yöntemlerinin karşılaştırılması 25
3.2 Trilaterasyon (Üçgenleme) ile Konum Belirleme 26
3.3 Esp32 Uwb Cihazı ile ToA Uygulaması 31
3.3.1 Donanımsal mimarisi ve çalışma prensibi 31
3.3.2 Teknik özellikler ve uygulama alanları 33
3.3.3 Veri iletim hızları, güç tüketimi ve çalışma frekansları 34
3.4 Udp Üzerinden Veri İletimi 36
3.4.1 Sabit istasyon (anchor) ayarları 36
3.4.2 Hareketli verici (tag) işleyişi 36
3.4.3 Round robin algoritması 38

3.4.4 Python ile veri işleme ve görselleştirme 40

3.5 Rssı ile Mesafe Hesabı ve Denklemler 42

3.6 Esp32 Uwb'de Rssı Ölçüm Aralığı 45

3.7 Malzemeler 49

3.7.1 Esp32 uwb pro long distance wrover 49

3.7.2 K552 safe 3a örgülü metal başlı type-c şarj kablosu 50

3.7.3 Preo ps2 20000mah powerbank 51

3.7.4 Preo ps2 10000mah powerbank 51

3.7.5 Yd-32 4 port 3.0 usb çoğaltıcı hub 52

3.7.6 Zyxel vmg1312-t20b antenli vdsl2/adsl2 kablosuz modem 53

3.8 Araştırma Modeli 55

3.9 Veriler ve Toplanması 57

3.9.1 Sabit alıcı istasyonların arduino kodları 59

3.9.1.1 Sabit alıcı istasyonların mac ıd adreslerinin belirlenmesi 60

3.9.1.2 Sabit alıcı istasyonların bağlantı pinlerinin ayarlanması 61

3.9.2 Hareketli vericinin arduino kodları 62

3.9.2.1 Hareketli vericinin mac ıd adresinin belirlenmesi 63

3.9.2.2 Statik ıp ile kapalı devre ağının oluşturulması 63

3.9.2.3 Udp haberleşme ayarları 67

3.9.2.4 Hareketli vericinin wi-fi bağlantı yönetimi 68

3.9.2.5 Hareketli vericinin spı ve bağlantı pinlerinin ayarlanması 69

3.9.2.6 Round robin algoritmasının yazılması 70
3.9.2.7 Mesafe ve sinyal gücü verilerinin güncellenmesi 73
3.9.2.8 Konum bildiriminde düzenli udp yayını 75
3.9.2.9 Udp bildirimlerinin dinlenmesi ve konum modu kontrolü 78
3.9.2.10 Yeni cihaz bağlantı ve kesilme geri çağrıları 80
3.9.2.11 Hareketli vericiden mesafe ve seri çıktıların alınması 81
3.10 Verilerin Çözümlenmesi ve Yorumlanması 83
3.10.1 Udp üzerinden veri alımı ve çözümleme 85
3.10.2 Mac adresi ile cihaz tanımlama ve seçimi 86
3.10.3 Sabit istasyonların grafiksel konumlandırılması 87
3.10.4 Round robin algoritması ile veri çakışmasının önlenmesi 89
3.10.5 Koordinat sistemi ve sabit istasyonların grafiksel gösterimi 90
3.10.6 Konum belirleme yöntemi 93
3.10.7 Hareketli vericinin gerçek zamanlı takibi 96
3.10.8 Mesafe ve koordinatların kullanıcıya sunumu 99
3.10.9 Hareketli verici konumunun görüntülenmesi 101
3.10.10 Çoklu istasyon veri güncelleme ve zaman aşımı 104
3.10.11 Esp32 uwb kontrol paneli: cihaz yönetimi ve sistem ayarları 105
4. BULGULAR VE TARTIŞMA 114
4.1 Arduino Bulguları 114
4.2 Terminal Bulguları 116

4.3 Python Arayüz Bulguları 125
4.4 Ölçüm Sürecinde Karşılaşılan Zorluklar ve Çözümler 132
4.5 Yüksekliğin Konumlandırma Doğruluğuna Etkisi 132
4.6 Karşılaştırmalı Performans Analizi 133
4.7 Kısıtlamalar ve Gelecekte Yapılabilecek Çalışmalar 133
5. SONUÇ ve ÖNERİLER 134
KAYNAKÇA 137
EKLER 149
EK 1: Sabit alıcı istasyon kodları 149
EK 2: Hareketli verici kodu 152
EK 3: Arayüz yazılımı 158
ÖZGEÇMİŞ 172

TABLOLAR LİSTESİ

Tablo 1. Alıcı (RX) Duyarlılık Değerleri 45

Tablo 2. Verici (TX) AC Karakteristikleri 47

Tablo 3. Ortalama Hata Payı ve Ortalama Doğruluk Yüzdesi..... 131

ŞEKİLLER LİSTESİ

Şekil 1. Wi-Fi Parmak İzinin Görüntüsü 14

Şekil 2. Varış Zamanı Tekniği Geometrisi. 22

Şekil 3. Uçuş Zamanı Sistemlerinde Zaman Çözünürlüğü. 25

Şekil 4. Üç Çemberin İdeal Kesişimi ve Pratikteki Örtüşme Alanı 26

Şekil 5. İki Alıcılı Sistemde Oluşan Konum Belirsizliği 28

Şekil 6. Üç Alıcılı Sistemde Kesişim Noktasının Belirlenmesi 29

Şekil 7. IC Blok Diyagramı 31

Şekil 8. DW1000 Pin Atamaları 32

Şekil 9. Fonksiyonel Blok Diyagramı 33

Şekil 10. Round Robin Algoritması Zamanlama Şeması 38

Şekil 11. CustomTkinter ile Geliştirilen UWB Konum Takip Arayüzü. 40

Şekil 12. WLAN Ağında RSSI'nin Gelişimi 42

Şekil 13. ESP32 UWB Pro Long Distance Wrover 50

Şekil 14. K552 Safe 3A Örgülü Metal Başlı Type-C Şarj Kablosu 50

Şekil 15. Preo PS2 20000mAh Powerbank 51

Şekil 16. Preo PS2 10000mAh Powerbank 52

Şekil 17. YD-32 4 Port 3.0 USB Çoğaltıcı Hub 53

Şekil 18. ZyXEL VMG1312-T20B Modem 54

Şekil 19. Baz İstasyonlarının Tasarımı. 58

Şekil 20. **(a)** Sürekli Dalga Sinyali, **(b)** Zaman ve Frekans Alanlarında UWB Sinyali 58

Şekil 21. Kapalı Devre Ağının Kullanıcı Adı ve Şifresi. 64

Şekil 22. Kapalı Devre Ağının Özellikleri. 64

Şekil 23. Hareketli Verici İçin Oluşturulan UDP Portunun IP Adresi. 68

Şekil 24. Round Robin Algoritması Akış Şeması. 72

Şekil 25. Sabit Alıcı İstasyonların Eşkenar Üçgen Geometrisi. 88

Şekil 26. Orijin Noktası ve Koordinat Sistemi. 93

Şekil 27. Mesafe İlişkilerinin Görselleştirilmesi. 101

Şekil 28. Cihaz Seçme Ayarı. 110

Şekil 29. Cihazın Seçimi. 111

Şekil 30. Sabit Alıcı İstasyon Mesafelerinin Güncellenmesi. 112

Şekil 31. UWB Takip Sisteminin Görsel Arayüz Analizi. 112

Şekil 32. Sabit Alıcı İstasyonların Konumlandırılması. 126

Şekil 33. Ayaklı Sehpalar. 127

Şekil 34. Birinci Ölçümün Yapılması. 127

Şekil 35. Birinci Ölçüm Sonuçları. 128

Şekil 36. İkinci Ölçümün Yapılması. 129

Şekil 37. İkinci Ölçüm Sonuçları. 129

Şekil 38. Üçüncü Ölçümün Yapılması. 130

Şekil 39. Üçüncü Ölçüm Sonuçları. 131

KISALTMALAR

GPS : Global Positioning System-Küresel Konumlandırma Sistemi

AP : Access Point-Kablosuz Erişim Noktası

Ghz : Gigahertz

IoT : Internet of Things-Nesnelerin İnterneti

IPS : Indoor Positioning System-İç Mekân Konumlandırma Sistemi

NLoS : No Line-of-Sight-Görüş Hattı Eksikliği

RFID : Radio Frequency Identification-Radyo Frekansı ile Tanımlama

RSSI : Received Signal Strength Indicator-Alınan Sinyal Gücü Göstergesi

ToA : Time of Arrival-Varış Zamanı

ToF : Time of Flight-Uçuş Süresi

UDP: : User Datagram Protocol-Kullanıcı Veribloğu İletişim Kuralları

UWB : Ultra Wide Band-Ultra Geniş Bant

Wi-Fi : Wireless Fidelity-Kablosuz Ağ Bağlantısı

WLAN : Wireless Local Area Network-Kablosuz Yerel Ağ

RTT : Round Trip Time-Varış-Dönüş Zamanı

KNN : K-Nearest Neighbours-K-En Yakın Komşu Algoritması

TDoA **:** Time Difference of Arrival-Geliş Zaman Farkı

AoA **:** Angle of Arrival-Geliş Açısı Yöntemi

1. GİRİŞ

Konumlandırma; belli referanslara göre bir nesnenin yerinin matematik olarak tanımlama işlemidir. Son yıllarda, mevcut teknolojiler kullanılarak konumların belirlenmesi sorununu çözmeye yönelik birçok araştırma yapılmıştır. Konumlandırma, mekânik, optik ve benzeri birçok tekniğin kullanıldığı ölçmelere dayanır. Günümüzde bağımsız referans noktalarına dayalı elektronik yöntemler öne çıkmıştır. Bunlardan en bilineni GPS sistemleri olmakla birlikte, referans noktalarını teşkil eden uydulardan alınan sinyallerin duvar gibi yapı elemanlarına penetrasyonunda yaşanan güçlük nedeniyle kapalı mekânlarda kullanılması her zaman verimli olmamaktadır. Bu nedenle; kapalı alanlarda konum tespiti, daha çok işlemin gerçekleştirildiği ortama uygun yerel tekniklerle çözülmeye çalışılmakta ve ayrı bir alan olarak değerlendirilmektedir. Yapılan uygulamaların çoğunda mevcut teknolojilerin amaca uygun olarak uyarlanması yeterli olmaktadır.

Kapalı alan konumlandırma sistemleri, görme engelli bireyler için iç mekân navigasyonu, binalarda cihaz takibi, müzelerde ziyaretçilere rehberlik, dumanlı ortamlarda acil çıkış yollarının bulunması, kalabalık alanlarda çocuk izleme ve değerli ekipman takibi gibi birçok farklı uygulama alanına sahiptir. Bu tür uygulamalar, belirli kalite gereksinimlerini ortaya koyduğundan, kapalı alan konumlandırma sistemlerinin, her uygulamanın ihtiyaçlarına göre dikkatlice seçilmesi gerekmektedir (Alarifi vd, 2016).

Bu noktada, UWB teknolojisi, dış etkenlerden ve yansıtıcı engellerden minimum düzeyde etkilenerek son derece doğru bir izleme imkânı sağlayan yenilikçi bir çözüm olarak öne çıkmaktadır. 500 MHz'in üzerinde bir bant genişliğine ve taşıyıcı frekansın %20'sinden fazlasını kapsayan bir spektruma sahip olan UWB, kısa mesafeli veri aktarımı için tasarlanmış hızlı ve güvenilir bir radyo teknolojisidir. Kapalı alan yerelleştirme sistemlerinde, UWB, gerçek zamanlı izleme ve veri toplama özellikleriyle yüksek doğruluk sağlayan bir teknoloji olarak dikkat çekmektedir. UWB'nin özel sinyal yapısı, 15 ila 25 metrekarelik bir alanda yüksek hassasiyet sunarak iç mekânlardaki konum tespiti ve yer belirleme süreçlerini oldukça etkili hale getirmektedir. Bu özellikleriyle UWB, iç

mekân yerelleştirme sistemleri için ideal bir seçenek olarak ön plana çıkmaktadır (Mohd Sultan vd., 2024).

UWB teknolojisinin iç mekân yerelleştirme sistemlerine entegrasyonu, etiket ile ankraj arasındaki sinyal parametrelerinin belirlenmesini gerektirir. Bu sinyal parametreleri, Varış Zamanı (ToA) ve Varış Zaman Farkı (TDoA) gibi alıcı-verici özelliklerinden yararlanılarak yüksek hassasiyetle ölçülür. Genellikle, alınan her bir sinyal için yalnızca bir sinyal parametresi ölçülmektedir. Ancak, birden fazla sinyal parametresinin eş zamanlı ölçüm imkanı da bulunmaktadır ve bu, yerelleştirme doğruluğunu daha da artırma potansiyeli sunar (Mohd Sultan vd., 2024).

Ayrıca, ABD Federal İletişim Komisyonu'nun (FCC) lisanssız UWB iletişimi kullanımına verdiği onayın ardından, UWB teknolojisinin sivil uygulamaları dünya çapında kapsamlı bir şekilde incelenmiş ve araştırılmıştır. UWB teknolojisini benimseyen uluslararası kablosuz iletişim standartlarının geliştirilmesi de bu alandaki araştırma ve geliştirme çalışmalarını teşvik etmiştir. Bunun sonucunda, UWB konumlandırma performansını geliştirmek amacıyla yeni algoritmalar üzerinde çalışmalar yapmak, aktif bir araştırma konusu olmuştur (Al-Hadhrami vd., 2014).

1.1 Problem

Ofisler, alışveriş merkezleri, hastaneler gibi kapalı mekânlarda kişilerin ve ekipmanın hassas bir şekilde izlenmesi, operasyonel verimlilik, güvenlik ve organizasyonel süreçlerin optimize edilmesi açısından önem arz etmektedir. GPS teknolojisi, kapalı alanlarda sinyal kaybı yaşadığı için etkin bir çözüm sunamamaktadır. Bu bağlamda, UWB teknolojisi, kapalı mekânlardaki hassas konumlandırma ihtiyaçlarını karşılamak için etkili bir çözüm sağlamaktadır. Ancak, UWB tabanlı düşük maliyetli sistemlerin geliştirilmesi ve kullanıcı dostu bir arayüz ile görselleştirilmesi üzerine yapılmış çalışmalar sınırlıdır. Bu çalışmada, ESP32 UWB Pro cihazlarının kullanıldığı UDP tabanlı bir sistem ile kapalı mekânlardaki konumlandırma verilerinin toplanması ve bu verilerin bir arayüz ile görselleştirilmesi hedeflenmektedir (Hapsari, Munadi, Erfianto ve Irawati, 2024).

1.2 Amaç

Bu çalışmanın amacı, iki adımda ifade edilebilir.

Bu çalışmanın amacı, kapalı mekânlarda işçi, ziyaretçi gibi insan veya ekipmanların gerçek zamanlı konum takibini sağlayan, ESP32 UWB Pro cihazlarıyla çalışan düşük maliyetli bir sistem geliştirmektir. Sistem, UDP protokolü üzerinden toplanan konum verilerini bir Python arayüzünde görselleştirerek kullanıcıya sunmaktadır. Bu sistemin, kapalı mekânlarda güvenliği artırması ve süreçleri optimize etmesi hedeflenmektedir.

Bu genel amacı gerçekleştirebilmek için cevaplandırılmaya çalışılan sorular şunlardır:

- ESP32 UWB Pro cihazları kullanılarak, kapalı mekânlarda doğru ve kararlı bir konumlandırma sistemi geliştirilebilmesi
- UDP protokolü üzerinden alınan konum verileri, düşük gecikme ve veri kaybı olmadan bir Python arayüzünde görselleştirilebilmesi
- Geliştirilen sistemin konumlandırma doğruluğunun kapalı mekânlarda kullanılan diğer sistemlerle karşılaştırıldığındaki seviyesinin belirlenebilmesi
- Sistemin 200 metrelik bir çalışma alanında güvenilir ve ölçeklenebilir bir şekilde çalışabilmesi
- Geliştirilen sistemin, kapalı mekânlardaki güvenliği artırma ve operasyonel süreçleri optimize etme potansiyelinin değerlendirilebilmesi

1.3 Önem

Bu çalışmada, kapalı mekânlar için düşük maliyetli, hassas ve taşınabilir bir konum takip sistemi sunulmaktadır. Mevcut çözümlere kıyasla, sistem UWB teknolojisinin yüksek hassasiyet avantajını kullanmakta; Python ve Tkinter ile geliştirilen kullanıcı dostu bir arayüz sayesinde kolay erişim ve görselleştirme imkânı sağlamaktadır. Geliştirilen bu sistem, kapalı alanlardaki operasyonların daha etkin bir şekilde yönetilmesine ve çalışan güvenliğinin artırılmasına önemli katkılar sunmaktadır.

Gelecek nesil iletişim ağlarında, yüksek doğrulukta, güvenilir ve anlık iç mekân konum belirleme teknolojilerinin yanı sıra konuma dayalı protokoller ve servisler büyük bir ihtiyaç haline gelecektir. Bu tür sistemler, mobil cihazların kendi konumlarını tespit etmesine imkan tanımakla kalmaz, aynı zamanda navigasyon, takip ve izleme gibi konum temelli uygulamalarda bu verilerin kullanılmasını mümkün kılar. Kullanıcıların veya cihazların yer bilgisi, ağ tasarımı, dinamik ağ uyarlamaları ve trafik dengelenmesi gibi alanlarda kablosuz ağ performansını önemli ölçüde artırabilir. Nitekim, hastaneler gibi kritik ortamlarda, değerli tıbbi cihazların kaybolmasını veya çalınmasını engellemek amacıyla konum bazlı izleme sistemleri halihazırda kullanılmaktadır. Kapalı ve sürekli değişen iç mekânlarda karşılaşılan belirsizlikler, nesnelerin ve bireylerin konumlarına ilişkin bilgilerin sağlanmasıyla önemli ölçüde azaltılabilir. Bu tür ortamlarda, gerçek zamanlı ve doğru konum verisi sunabilen sistemlere duyulan ihtiyaç giderek artmaktadır. İç mekân konumlandırma sistemleri (IPS), evler, iş yerleri, mağazalar, fitness salonları ve benzeri pek çok farklı kapalı alanlarda konuma dayalı hizmetlerin ve uygulamaların etkin biçimde kullanılabilmesini sağlayan temel altyapıyı oluşturur. Bu sistemler, konum bağlamı sunarak yalnızca yer tespitini değil, aynı zamanda hareket takibini, kaynak yönetimini ve güvenliği de destekleyen kritik bir rol üstlenmektedir (Yassin ve Youssef, 2009).

Bu proje, UWB Teknolojisine dayalı kapalı mekânlarda hassas konumlandırma sistemini önermektedir. Sistem, iletim hızı yüksek, geniş bant genişliğine sahip, yüksek zaman çözünürlüklü, anti çoklu yol parazitine karşı güçlü, çalışırken düşük güç tüketen ve konumlandırma doğruluğunun yüksek olması gibi UWB sinyalinin özelliklerini kullanır (Chen, 2021).

Ultra geniş bant, 1 ns'lik bir darbe sinyali göndererek verileri iletir. Bu sayede, frekans alanında ultra geniş bir bant genişliği sağlar. Çok kısa zaman alanındaki sinyal darbesi nedeniyle, güçlü ve çoklu yol etkilerinde bile çalışabilir. İlk olarak, sabit alıcı istasyonları ile erişim noktası arasında sinyaller göndererek uçuş süresini (ToF) ölçer. Daha sonra, trilaterasyon algoritması kullanılarak mesafe hesaplanır ve doğru konum belirlenir (Wu, 2021).

Ultra geniş bant (UWB) teknolojisi, Wi-Fi ve Bluetooth gibi geleneksel RF tabanlı sistemlerden farklı olarak, zaman tabanlı mesafe ölçümüne dayalı benzersiz bir yaklaşım sunar. UWB sinyallerinin darbe genişliğinin son derece kısa olması, uçuş süresi (ToF) ölçümlerinde milimetrik seviyede hassasiyet sağlar. Bu özellik, konumlandırma sistemlerinde diğer kablosuz teknolojilere kıyasla önemli ölçüde daha düşük hata oranı (<10 cm) ile mesafe ve yön tespiti yapılabilmesine olanak tanır (Al-Okby, Junginger, Roddelkopf ve Thurow, 2024).

1.4 Varsayımlar

- ESP32 UWB Pro cihazları, kapalı mekânlarda kararlı bir şekilde çalışacaktır.
- UDP tabanlı haberleşme protokolü, gerçek zamanlı veri aktarımı için yeterli performansı sunacaktır.
- Python programlama dili ve Tkinter kütüphanesi, kullanıcı dostu bir arayüz geliştirilmesine olanak sağlayacaktır.

1.5 Sınırlılıklar

- Sistem yalnızca 3 ESP32 UWB Pro alıcı ve 1 ESP32 UWB Pro verici cihaz ile test edilmiştir.
- Geliştirilen arayüz, yalnızca Python ortamında çok bilinen Python kütüphaneleri kullanılmıştır. Farklı kütüphanelerle arayüz özellikleri arttırılabilir.
- Çalışma, ofis, hastane gibi belirli kapalı ve farklı ortamlar için adaptasyon gerektirebilir.

1.6 Tanımlar

GPS (Global Positioning System): Küresel Konumlandırma Sistemi (GPS), temelleri 1973 yılında ABD Savunma Bakanlığı'nın askeri amaçlarla geliştirdiği NAVSTAR projesine dayanmaktadır. Sistem, uydular tarafından iletilen radyo sinyallerini kullanarak, tüm hava koşullarında ve gece-gündüz fark etmeksizin hızlı, doğru ve ekonomik bir şekilde üç boyutlu konum belirleme imkanı sağlar. Doğrudan görüş hattı gerektirmeyen bu teknoloji, temel navigasyon ihtiyaçlarının yanı sıra yüksek hassasiyetli

zaman ölçümü ve hız tespiti gibi fonksiyonları da sunmaktadır. GPS'in uzay segmenti, dünya yüzeyinden yaklaşık 20.200 kilometre yükseklikte, ekvator düzlemiyle 55 derecelik açı yapan altı farklı yörünge düzlemine yerleştirilmiş 24 uydudan oluşur. Bu uydular, 12 saatlik periyotlarla yörüngelerinde hareket ederek, dünyanın herhangi bir noktasından her zaman en az dört tanesinin görülebileceği şekilde konumlandırılmıştır. Her uydu, 1575.42 MHz (L1 bandı, yaklaşık 19 cm dalga boyu) ve 1227.60 MHz (L2 bandı, yaklaşık 24 cm dalga boyu) frekanslarında olmak üzere iki farklı sinyal yayını yapmaktadır. Uydu altyapısı Temmuz 1993'te tamamlanmış olup, bu uydular dünya genelinde stratejik noktalara yerleştirilmiş beş izleme istasyonu tarafından sürekli takip edilmektedir. Bu istasyonlar yüksek hassasiyetli atom saatleri ve çift frekanslı alıcılarla donatılmış, konumları milimetrik doğrulukla bilinmektedir (Karaali ve Yildirim, 1996).

IPS (Indoor Positioning System): Kapalı alanlarda insanların, nesnelerin ve ekipmanların gerçek zamanlı konum bilgisini belirlemek için tasarlanmış teknolojik sistemlerdir ve GPS'in aksine bina içindeki fiziksel engellerden (duvarlar, metaller vb.) etkilenmeyen alternatif çözümler sunar. Bu sistemler kızılötesi, RFID, Bluetooth Low Energy (BLE), ultra geniş bant (UWB), Wi-Fi veya jeomanyetik alan gibi farklı teknolojileri kullanarak sinyal özelliklerini (şiddet, varış süresi, açı) analiz eder. Başlıca uygulama alanları arasında acil durum müdahaleleri (yangın anında personel konum takibi), endüstriyel envanter yönetimi (fabrikada ekipman takibi), sağlık hizmetleri (hastane içi hasta/cihaz izleme) ve akıllı bina sistemleri (alışveriş merkezlerinde müşteri trafik analizi) bulunur. Sistemler, self-positioning (cihazın kendi konumunu hesaplaması) veya remote-positioning (merkezi sunucu tabanlı hesaplama) mimarileriyle çalışabilir ve doğruluk seviyeleri kullanılan teknolojiye göre 1-10 metre arasında değişkenlik gösterir (Bernard, 2017).

TDoA (Time Difference of Arrival): Geliş Zaman Farkı, "Hiperbolik Sistem" olarak da bilinir. Baz istasyonlarında ölçülen varış zamanı değerleri arasındaki farkını kullanır. Bu yöntem için en az üç baz istasyonu gereklidir (Gökçen ve Yeşil, 2022).

ToF (Time of Flight): Uçuş Süresi, ultrasonik dalgaların uçuş sürelerini ölçer. Uçuş süreleri ölçülürken, ultrasonik uçuş süresini belirlemek için uçuş süresi hesaplama algoritmaları kullanılır (Medina, Sagura ve de La Torre, 2013).

RTT (Round Trip Time): RTT, bir verinin belirli bir noktadan (örneğin bir tarayıcıdan) gönderilip aynı noktaya geri dönmesi için geçen süredir ve genellikle milisaniye cinsinden ifade edilir. Bu süre, ağ gecikmesi, sayfa yükleme hızı ve genel ağ performansının değerlendirilmesinde önemli bir ölçüt olarak kullanılır. İnternet ve ağ bağlantılarında yaygın olarak kullanılan RTT, iki cihaz arasındaki veri iletiminde, kullanıcının bir sunucuya gönderdiği isteğin yanıtıyla birlikte tamamlanmasına kadar geçen zamanı gösterir. Düşük RTT değerleri daha hızlı bağlantıya, yüksek değerler ise daha yavaş bağlantıya işaret eder. Özellikle çevrim içi oyunlar ve görüntülü görüşmeler gibi gecikmeye duyarlı uygulamalar için kritik bir parametre olup, kullanıcı deneyimini doğrudan etkiler. Bu nedenle ağlarda RTT süresinin mümkün olduğunca düşük olması hedeflenir (Başkan, 2025).

UWB (Ultra Wide Band): Ultra geniş bant, 500 MHz'den fazla frekanslı, 2,5 GHz'ten fazla taşıyıcı frekanslı, ultra kısa ve yüksek bant genişliğinde bir kablosuz iletişim teknolojisidir. Ultra geniş bant, düşük enerji tüketimi ile yüksek ve çok yollu bağışıklığa sahip olması sayesinde geniş bir bant genişliği oluşturur. UWB, yüksek hız ve yüksek veri hızı ile yüksek zaman çözünürlüğü ve kısa dalga boyu ile iletişim kanalıyla çok yollu sönümleme durumlarında güçlendirilir (Asaad ve Maghdid, 2022).

RSSI (Received Signal Strength Indicator): Alınan Sinyal Gücü Göstergesi, bir cihazın bir ulaşım noktasından ya da bir yönlendiriciden gönderilen sinyali ne kadar iyi alabildiğinin ölçülmesidir. Sağlıklı bir kablosuz bağlantı sağlamak için yeterli sinyalin olup olmadığını belirlemek adına yararlı olan bir yöntemdir (Metageek, 2024).

ToA (Time of Arrival): ToA, gezgin cihaz ile baz istasyonu arasında gönderilen radyo sinyalinin gidiş-dönüş süresine dayalı olarak ölçülen bir parametredir. Bu süre hem gezgin cihazda hem de baz istasyonunda tespit edilebilir. Radyo sinyali baz istasyonundan gönderilip tekrar aynı istasyonda sonlandığında, ToA değeri istasyon tarafından ölçülür. Ölçülen sürenin gidiş-dönüş zamanı olduğu dikkate alınarak, gezgin ile baz istasyonu arasındaki mesafe bu sürenin yarısının ışık hızı ile çarpılmasıyla hesaplanır. Bir gezgin cihazın konumu ise, üç farklı baz istasyonundan elde edilen ToA değerlerinin kesiştiği noktadan belirlenir (Şahin, 2011).

AoA (Angle of Arrival): Geliş Açısı tahmini, birçok kablosuz algılama ve iletişim uygulaması için temel bir yöntemdir; özellikle RF kaynak konumlandırmada yaygın olarak kullanılır. Genellikle AoA tahmini, anten dizileri kullanılarak gerçekleştirilir. Bu yöntemde, anten dizisindeki her bir elemanın aldığı sinyaller arasındaki faz farkı, sinyalin geldiği yönün belirlenmesine olanak sağlar. AoA yönteminin iki önemli avantajı vardır: Birincisi, sinyalin fazı, sinyal gücüne kıyasla daha kararlı olduğundan, AoA tabanlı konumlandırma daha yüksek doğruluk sağlar. İkincisi, etkili bir AoA tahmin sistemi ile sadece iki anten dizisi kullanılarak hedefin konumu hassas şekilde bulunabilir; oysa menzil tabanlı yöntemler genellikle üç veya daha fazla sensör gerektirir. Ancak, anten dizileri genellikle pahalı ve karmaşıktır çünkü faz tespiti için anten elemanlarının sıkı koordinasyonu gereklidir. Ayrıca, bu dizilerin sabit eleman konfigürasyonları vardır, bu da sahadaki değişen ihtiyaçlara uyum sağlamayı zorlaştırır. Bu çalışmada, AoA tahmininin, modüler ve ticari olarak temin edilebilen yazılım tanımlı radyo cihazları kullanılarak uygulanabileceği gösterilmiştir. Bu yaklaşım, esneklik, düşük maliyet ve kolay kurulum avantajları sağlarken, makul bir konumlandırma performansı sunar (Chen, Lin, Kung, Lin ve Gwon, 2012).

UDP (User Datagram Protocol): Kullanıcı Datagram Protokolü, ağ katmanında bulunan ve üzerinde çalışan hafif bir veri aktarım protokolüdür. UDP, paketlerde oluşan bozuk verileri tespit etmek için bir mekânizma sağlar. Fakat kayıp veya bozuk paketlerdeki diğer sorunları çözmez. Bu nedenle, UDP, "Güvenilmez Veri Protokolü" olarak da adlandırılabilir. UDP, genellikle hızın, güvenli veri iletilmesinden daha ön planda olduğu durumlarda, yani genellikle gerçek zamanlı veri iletişimini sağlayan uygulamalarda, bilgisayar oyunlarında, video akışlarında kullanılır (Bilişim Evreni, 2018).

Wi-Fi (Wireless Fidelity): Wi-Fi, kablosuz, yüksek hızda internet ve ağ bağlantıları oluşturmak için radyo dalgalarını kullanır. Çok tercih edilen bir kablosuz ağ teknolojisidir. Bu teknolojinin amacı, bilgisayarların, akıllı telefonların, tabletlerin, laptopların ve diğer cihazların internete bağlanması veya belirli bir alanda kablosuz olarak birbirleriyle iletişim kurmasını sağlamaktadır (Kalite Network, 2023).

ESP32 UWB Pro (ESP32 Ultra Wide Band Pro Long Distance Wrover): Ultra geniş bant, radyo frekansı dalgalarını kullanarak çalışan, kısa mesafelerde güvenilir ve hassas

algılama ile mesafe ölçümü sağlayan bir kablosuz iletişim protokolüdür. UWB, kablosuz cihazlar için yeni bir uzamsal bağlam sunarak hem güvenli hem de etkin bir veri iletimi sağlar. DW1000, ultra geniş bant teknolojisini temel alan bir kablosuz alıcı-verici çipidir ve doğru konum tespiti ve iletişim sağlama yeteneğiyle gerçek zamanlı konum belirleme sistemleri, iç mekân konumlandırma, konuma dayalı hizmetler, kablosuz sensör ağları ve Nesnelerin İnterneti uygulamalarına yenilikçi bir yaklaşım getirmektedir (Makerfabs Wiki, 2023).

WSN (Wireless Sensor Network): Kablosuz Sensör Ağı, algılanacak nesnelerle ilgili verilerin bir bilgi sistemine iletilmesini sağlamak amacıyla kablosuz iletişim kurabilen sensör cihazlarından oluşan bir teknolojidir. Bu ağlar, genellikle birbiriyle bağlantılı çok sayıda düğüm içerir ve diğer düğümlerle iletişim kurmak için dijital kodlama ve şifreleme yöntemlerini kullanır. Kablosuz sensör ağları, askeri alanların yanı sıra çevre izleme, sağlık, endüstriyel süreçler ve konut uygulamaları gibi farklı amaçlarla da kullanılmaktadır (Satria, Zulfan ve Hidayat, 2018).

IoT (Internet of Things): IoT terimi, 1999 yılında İngiliz teknoloji öncüsü Kevin Ashton tarafından ortaya atılmıştır. IoT, sensörler, yazılım ve ağ bağlantılarıyla donatılmış fiziksel cihazlar, elektrikli araçlar, ev aletleri ve diğer nesnelerin birbiriyle veri alışverişi yapabilmesini sağlar. Bu sayede, akıllı ve etkileşimli bir ortam oluşturulur (Anbar, 2023).

Tkinter (ToolKit Interface): Tkinter, Python'da bir Grafiksel Kullanıcı Arayüzü (GUI) kitaplığıdır. Python'da bulunan standart kütüphanenin bir bölümü olarak gelir. Bu sayede, kullanıcılar, çapraz platform GUI uygulamaları oluştururlar. Basit bir yapısı olması nedeniyle, Python ile GUI uygulamaları geliştirmek isteyenler için Tkinter çok tercih edilmektedir. Tk GUI toolkit'inin Python sarmalayıcısı olduğu için, kullanıcıların pencere, düğme, menü, metin kutusu ve diğer grafik arayüz bileşenlerini oluşturmalarına imkan sağlar (Medium, 2023).

2. GENEL BİLGİLER

Geçmişten günümüze kadar konum belirlemeye olan ihtiyaç süreklilik göstermiştir. İlk dönemde, konum belirleme amacıyla yıldızlar, rüzgâr yönleri ve deniz fenerleri gibi araçlardan faydalanılmış, zamanla artan ihtiyaçlar ve teknolojik ilerlemeler doğrultusunda farklı yöntemler geliştirilmiştir. Günümüzde ise yapay uydularla yapılan konum belirleme yöntemleri kullanılmakta olup, her geçen gün yeni hesaplama, ölçüm teknikleri ve uydu teknolojileri geliştirilmektedir (Konukseven, 2021).

Konum bilgisi, taşımacılık ve kargo hizmetlerinde araçların izlenmesi amacıyla kullanılmaya başlanmış ve bu kullanım, güvenlik, yol yardımı, insan takibi ve reklamcılık gibi çeşitli alanlara yayılmıştır (Bozacı ve Beğdeş, 2023).

Konum bilgisi, çeşitli alanlarda farklı amaçlarla kullanılmaktadır. Mevcut yerin belirlenmesi, hedeflenen konuma rota oluşturulması, havacılık, araç takip sistemleri, spor uygulamaları, meteorolojik tahminler, bankacılık uygulamaları ve kadastral ölçümler gibi pek çok alanda kullanılmakla birlikte, alışveriş uygulamalarında da civardaki mağazaların kampanya bilgilerini sunmak için konum verisi kullanılmaktadır (Varlı ve Bingöl, 2024).

Sağlık sektöründe de konum bilgisi kritik öneme sahiptir. Acil çağrı merkezlerine yapılan çağrılarda, aracılardaki acil çağrı sistemlerinde, Emniyet Genel Müdürlüğü tarafından sunulan KADES uygulamasında ve arama-kurtarma operasyonlarında konumun belirlenmesi hayati önem taşır. Ayrıca, bazı akıllı saatler, kaza algılama sistemleriyle entegre çalışarak acil durum çağrıları yapabilir ve saatin konum bilgisini paylaşabilir (Varlı ve Bingöl, 2024).

Kapalı mekânlarda konum belirleme (Indoor Positioning), konuma dayalı hizmetlerin sunulmasındaki geniş uygulama alanları nedeniyle günümüzde popüler bir araştırma konusu haline gelmiştir. Araştırmalar, bireylerin zamanlarının %80'inden fazlasını kapalı mekânlarda geçirdiğini göstermekte olup, kapalı mekânlarda konum tabanlı hizmetlere olan gereksinim her geçen gün artmaktadır. Konum Tabanlı Hizmetler (KTH), nitelikli bilgiye erişimi sağlamak amacıyla konum verilerinin kullanıldığı yazılım tabanlı bir hizmet türüdür (Özdemir, 2020).

Bu tür hizmetler, alışveriş merkezlerinde mağazaların gerçek zamanlı navigasyonunun sağlanması, kamu binalarında ofislerin konum tespiti, ambarlarda depolanan ürünlerin yerlerinin belirlenmesi, hastanelerde personel, hasta veya ekipman takibi, yangın anında bina içindeki itfaiyecilerin konumunun belirlenmesi gibi çeşitli uygulamaları kapsamaktadır. Ayrıca, görme engelli bireylere yardımcı sistemler, patlayıcı madde arama çalışmalarında eğitimli polis köpeklerinin izlenmesi, konum bazlı reklamcılık, lokal arama hizmetleri, büyük fuarlarda ziyaretçilerin doğru stantlara yönlendirilmesi gibi alanlarda da bu hizmetlerden yararlanılmaktadır (Özdemir, 2020).

Diğer taraftan, şirketlerde otomatik oturum açma ve kapatma işlemlerinin uygulanması, yaşlı bireylerin ev ortamında tıbbi takibi, acil durum ve düşme tespiti, mobil cihazlarda kullanıcı konumuna dayalı içerik düzenlemeleri, müzelerde ziyaretçi hareketlerinin takibi ve analiz edilmesi gibi uygulamalar da konum tabanlı hizmetlerin kullanım alanları arasında yer almaktadır. Ayrıca, binaların iç mekânlarının ölçülmesi, yeni yapıların geometrik olarak modellenmesi, tünel ve maden gibi zorlu ortamlarda konum belirleme çalışmaları, artırılmış gerçeklik uygulamaları ve diğer birçok alanda da bu hizmetlerden faydalanılmaktadır (Özdemir, 2020).

Kapalı mekân konum belirleme sistemlerinin performansı, belirli kriterler temelinde değerlendirilmektedir. Bu kriterler:

- Doğruluk

- Hassasiyet

- Kullanılabilirlik veya Bulunma Durumu,

- Ölçeklenebilirlik

- Kapsama Alanı,

- Maliyet

- Enerji Tüketimi

- Hesaplama Gereksinimi

- Dayanıklılık (Robustluk) olarak sıralanmaktadır (Özdemir, 2020).

2.1 Konumlandırma Çeşitleri

Mevcut tüm konumlandırma sistemleri, açık alan konumlandırma ve kapalı alan konumlandırma olarak ikiye ayrılmaktadır. Açık alan konumlandırmada, GPS her zaman yüksek doğrulukla destek sağlayabilmektedir. Ancak, Kapalı Alan Konumlandırma Sistemleri (IPS) açık alan konumlandırmasına göre daha fazla zorlukla karşı karşıyadır. Bu zorluklar, yaygın engeller ve etkileşim girişimleri, çok yollu etki, sönümleme, yansıma, derin gölgeleme etkisi ve bozulma gibi faktörlerden kaynaklanmaktadır (Syazwani, J., Abdul Wahab, N., Sunar, N., Ariffin, S., Wong, K. ve Aun, Y., 2022).

2.1.1 Açık alan konumlandırması

Küresel Konumlandırma Sistemi (GPS), son yıllarda açık alan için geliştirilmiştir ve askeri alanlar, hava durumu tahmini, araç takibi, haritalama, çiftçilik gibi birçok farklı açık alan uygulaması sunmaktadır. GPS, açık alanda kesin bir konum, hız ve zamanı belirleyebilme imkânı sağlar. GPS alıcıları, sinyal yaymak yerine bu sinyalleri pasif olarak alır. Ancak, görüş hattı eksikliği (NLoS), zayıf sinyal gücü gibi nedenlerle doğruluk derecesini kaybettiğinden kapalı alanlarda GPS kullanıma uygun değildir (Syazwani vd., 2022).

2.1.2 Kapalı alan konumlandırması

Kapalı alan konumlandırması veya kapalı alan lokalizasyonu, önceden belirlenmiş bir referansa göre konum belirlenmesi sürecidir. Tüm kapalı alan konumlandırma sistemleri, konum bilgisi sağlamak amacıyla tasarlanmıştır ve bazıları bu bilgiyi kullanarak navigasyon yetenekleri de sunmaktadır. Konum bilgisi çoğunlukla fiziksel, mutlak, göreceli ve sembolik olmak üzere dört şekilde sunulabilir. Fiziksel konum, küresel bir referans çerçevesine göre (örneğin, coğrafi koordinat sistemi içinde enlem ve boylam) elde edilir. Mutlak konum, bir yerel referans çerçevesine göre ifade edilir ve çerçevenin çözünürlüğü, ızgara boyutuna bağlıdır. Göreceli konum, kullanıcının çevredeki bilinen yer işaretlerine olan yakınlığını ifade eder. Sembolik konum ise, kullanıcının bulunduğu yeri doğal dilde ifade ederek soyut bir bilgi sunar (örneğin, oturma odasında, mutfakta vb.) (Alhomayani, 2021).

2.2 Teorik Yaklaşımlar

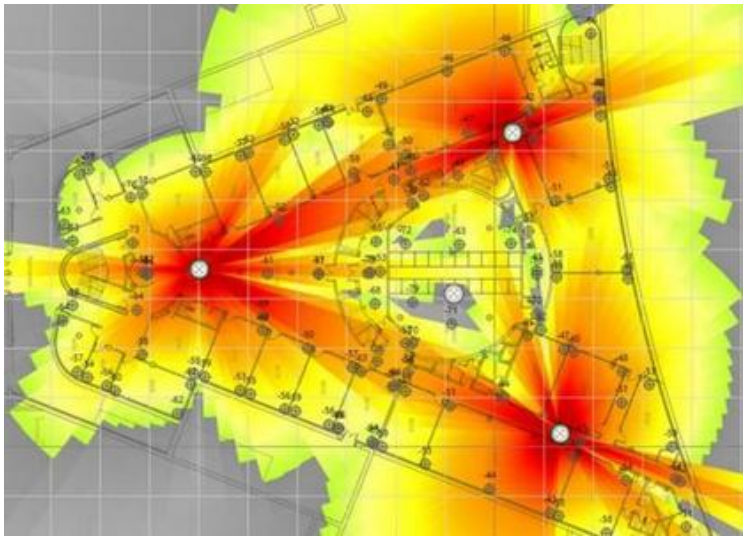
“Konumlandırma”, yani “navigasyon”, bir kişiyi veya nesneyi izlemek demektir. Kapalı mekân konumlandırması, iç mekânda konumlandırmanın gerçekleştirilmesidir [30]. İç mekânlarda GPS sinyallerinin alınamadığında veya yetersiz kalındığında, kapalı alandaki insanların yerini belirlemede navigasyon kullanılır. Kapalı mekân konumlandırma sistemi, küçük veya büyük iç mekânlardaki insanları veya nesneleri izlemek amacıyla özel olarak kullanılan bir teknolojidir (Jasani, 2023).

Bu sistem, hiçbir ek donanım maliyetine gerek kalmadan, dünya üzerindeki tüm noktalardaki kendilerine ait sinyal gücünden destek alarak çalışır. RF, manyetik, akustik ve diğer sensörlerden alınan sinyaller, akıllı bir telefon veya tablet tarafından algılanır ve akıllı bir veri tabanına aktarılır. Böylece, toplanan sinyaller sayesinde iç mekânlardaki konum saptanır (Venuex, 2018).

İç mekân konumlandırmada öne çıkan teknoloji çeşitleri şunlardır: Wifi, RFID (Radio Frequency Identification), Bluetooth Low Energy, Ultra Wide Band (UWB) ve Ultrasonik (Venuex, 2018).

2.2.1 Wifi iç mekân konumlandırması

Wi-Fi tabanlı konumlandırma sistemlerinde en önemli unsur, sinyal gücüdür. Bu sistemler genellikle RSSI (Received Signal Strength Indicator) yani alınan sinyal gücü göstergesine dayanır. Wi-Fi sistemlerinde birçok erişim noktası bulunur ve bu noktalara gelen sinyal gücündeki kayıp, erişim noktaları ile sinyal kaynağı cihaz arasındaki mesafeyle ters orantılıdır. Mesafe azaldıkça RSSI değeri artarken, mesafe arttıkça RSSI azalır. Bu doğrultuda Wi-Fi tabanlı konumlandırma algoritmaları, genellikle sinyal gücündeki kayıp mantığı üzerine kuruludur. Ayrıca, Wi-Fi sinyallerinin duvarlardan geçebilme özelliği, erişim noktalarının çokluğunun yanı sıra önemli bir avantaj sağlar. Ancak, duvarların kalınlıkları ve yalıtım malzemeleri sinyallerin gücünü etkilediği için, her iç mekân için özgün bir konumlandırma haritası çıkarılması gerekmektedir. Bu haritaların oluşturulmasında ise en çok Wi-Fi parmak izi yöntemi kullanılmaktadır (Akleylek, Kılıç, Söylemez, Aruk ve Çavuş Yıldırım, 2020).



Şekil 1. Wi-Fi Parmak İzinin Görüntüsü (Akleylek vd., 2020).

Wi-Fi parmak izi konumlandırma, iç mekanlarda kablosuz erişim noktalarından (AP) alınan sinyal gücü örüntülerinin (RSSI) konumsal veriyle eşleştirilmesine dayanan iki aşamalı bir yöntemdir. Bu teknik, öncelikle referans noktalarında sinyal ölçümlerinin toplandığı çevrimdışı (offline) aşama ve gerçek zamanlı konum tahmininin yapıldığı çevrimiçi (online) aşama olmak üzere iki temel süreçten oluşur (Xia, Liu, Yuan, Zhu ve Wang, 2017).

Çevrimdışı aşamada, bilinen konumlara sahip birçok referans noktasında (RP) farklı erişim noktalarından (AP) alınan tüm Wi-Fi sinyallerinin alınan sinyal gücü göstergesi (RSSI) vektörleri toplanır. Böylece her RP kendi parmak izi ile temsil edilir. Toplanan tüm RSSI vektörleri, çevrimiçi sorgulama aşamasında kullanılmak üzere bir veritabanında saklanır. Çevrimiçi aşamada ise kullanıcı (veya hedef), bulunduğu konumda bir RSSI vektörü ölçer veya örnekler ve bu veriyi sunucuya iletir. Sunucu, sinyal uzayında bir benzerlik metriği (örneğin, Öklidyen mesafe) kullanarak alınan bu vektörü veritabanında saklanan parmak izleriyle karşılaştırır. Daha sonra, hedefin konumu, RSSI'si en çok benzeyen referans noktalarının belirlenmesiyle tahmin edilir (He ve Chan, 2016).

Wifi tabanlı konumlandırma sistemlerindeki hassasiyet, santimetrelerden metrelere kadar kayabilir. Doğruluk oranı 5 metre ile 15 metrelere kadar ulaşabilir. Ayrıca, kapsama alanı 150 metre civarındadır (Akleylek vd., 2020).

Kablosuz Yerel Alan Ağı, yüksek frekanslı radyo dalgaları üzerinden farklı cihaz türlerini (dizüstü bilgisayar, mobil cihaz vb.) birbirine bağlar. Cihazlarda, IEEE 802 WLAN adaptörü kullanılmıştır. WLAN teknolojisi binalarda, hastanelerde, alışveriş merkezlerinde, kampüslerde, müzelerde ve benzeri yerlerde uygulanmıştır. WLAN teknolojisi; cep telefonları, dizüstü bilgisayarlar, tabletler ve televizyonlar gibi elektronik cihazların her türü ile entegre edilmiş ve işleyen bir sistem haline getirilmiştir. WiFi iç konumlandırma sisteminde, konumlandırmanın yapılabilmesi için alınan sinyal gücü göstergesi temeli kullanılarak verici WLAN sinyalleri yeniden kullanılır. WiFi sinyalleri duvarlara ve insan vücuduna temas edebilir. Verici ile alıcı arasında görüş hattına ihtiyaç yoktur, bu nedenle bu sistemin doğruluğu kapalı ortamlara (engeller, duvarlar, mobilyalar vb.), binaların içindeki erişim noktalarının sayısına ve kapsama alanına göre geçerlidir (Doughangi, 2017).

2.2.2 Bluetooth iç mekân konumlandırması

İç mekân konumlandırmasında kullanılan bir diğer yöntem Bluetooth teknolojisidir. Bluetooth standart hâliyle fazla enerji tükettiği için, bu sistemlerde genellikle pil tüketimi çok düşük olan Bluetooth Low Energy (BLE) teknolojisi tercih edilir. BLE, standart Bluetooth'a göre daha düşük bağlantı hızına sahip olsa da cihaz eşleştirmesinde daha hızlıdır. Kapalı mekân konumlandırmasında BLE'nin tercih edilmesinin en önemli nedenleri düşük maliyetli olması ve sinyal bozulmalarının azlığıdır. BLE tabanlı konumlandırma süreci, öncelikle referans noktalarının belirlenmesi ve en uygun RSSI yayılım modelinin oluşturulmasıyla başlar. Daha sonra, alınan sinyal gücü (RSSI) gerçek zamanlı olarak ölçülerek anormallikler tespit edilir ve düzeltilir. Model oluşturulduktan sonra gerçek mekân ile ilişkilendirilir ve BLE referans noktaları için aktif öğrenme süreci başlar; bu süreçte model üzerinde gerekli düzenlemeler yapılır. Sistemlerin doğruluk oranı ise oluşturulan tasarımın kalitesine bağlıdır (Akleylek vd., 2020).

Beacon, bataryalar ile çalışan düşük enerjili bluetooth sinyalleri yayan bir radyo vericisidir. Bu vericiler bluetooth protokolleri yardımı ile bluetooth özelliğine sahip mobil cihazlar aracılığıyla iletişime geçebilirler. Mobil cihazlar, beacon vericilerinden uzaklaştıkça algıladıkları RSSI değeri giderek azalır. Farklı işletim sistemlerine sahip mobil cihazlarda maksimum algılama oranında değişiklikler ortaya çıkmaktadır. 4 dbm

sinyal yayılım gücü, 1,2 metre yükseklikte konumlandırılmış beaconlar, iOS işletim sistemine sahip mobil cihazlarda 85 metreye kadar, android işletim sistemine sahip mobil cihazlarda ise 100 metreye kadar kapsama alanına sahiptirler. Beaconların sinyal yayılım gücü, konumlandırıldığı yükseklik, bulunduğu ortamın sinyal yansıtma oranına göre kapsama alanlarında değişiklik göstermektedir (Venuex, 2018).

Ek olarak, BLE fiziksel ve MAC katmanlarına ilişkin spesifikasyonları kapsayarak kişisel alandaki sabit veya mobil kablosuz cihazların bağlantısını sağlar. BLE sinyalleri, BLE işaretleyicileri (beacon) aracılığıyla algılanır ve bireyin konumu, alınan sinyal gücü göstergesi (RSSI) ile işaretleyicinin konumu birleştirilerek belirlenir. BLE protokolü, faz içi (in-phase) ve kuadratür (quadrature) örnekleme tekniklerini içerir; bu sayede geliş açısı ve ayrılış açısı hesaplanabilir. BLE'nin en önemli avantajlarından biri, son derece düşük güç tüketimine sahip olmasıdır; bu değer genellikle yaklaşık 0,367 mW'tır. Bu düşük tüketim, pille çalışan cihazların uzun süre boyunca, hatta yıllarca pil değişimine ihtiyaç duymadan çalışmasına olanak tanır. Ancak BLE'nin bazı dezavantajları da vardır. Bu dezavantajlardan biri, sinyal algılamayı etkileyebilecek girişimdir. İşaretleyiciler aracılığıyla BLE sinyallerinin güvenilir şekilde algılanması bazı durumlarda zor olabilir ve bu da sistem güvenilirliğini azaltabilir (Li, Wu, Zhao ve Huang, 2024).

2.2.3 Rfıd iç mekân konumlandırması

RFID, bir nesne veya kişiye ait tanıma bilgisini (benzersiz seri sayı biçiminde) kablosuz olarak radyo dalgaları ile iletmek için kullanılan sistemleri tanımlamada kullanılan bir terimdir. RFID sistemlerinde, veri akıllı kart sistemlerinde olduğu gibi elektronik bir veri taşıyıcı aygıtı üzerinde saklanır. Akıllı kart sistemlerinin aksine, veri taşıyan aygıt için gereken güç ve okuyucu ile veri taşıyan aygıt arasındaki veri değişimi sırasındaki temas, yüzeysiz olarak manyetik veya elektromanyetik alanlarla sağlanır. RFID taşıyıcıda saklanan veriyi alabilmek için bir okuyucu gereklidir. Okuyucu, taşıyıcının hafızasında bulunan veriyi almak için antene enerji yükler. Bu sayede anten, radyo sinyalleri yayarak taşıyıcıyı aktifleştirir. Aktif hale gelen taşıyıcı da hafızasında yer alan veriyi bırakır. Anten, okuyucu ve taşıyıcı arasındaki iletişimi sağlar. Antenin şekline ve boyutuna göre frekans aralığı gibi sistemin performansını etkileyen bazı özellikler farklılık gösterir.

Bunun yanında sistemi daha etkin hale getirmek için anten, bilgisayar ve veritabanları kullanılmaktadır (Demiral, Karaş ve Turan, 2013).

RFID, iç mekân konumlandırmada yaygın olarak kullanılan bir teknolojidir. RFID sistemleri pasif veya aktif olabilir ve üç temel bileşenden oluşur: okuyucu, anten ve etiket. RFID konumlandırma genellikle RSSI, geliş açısı (AoA), geliş fazı (phase-of-arrival), parmak izi (fingerprint) ve uçuş zamanı (time-of-flight) gibi tekniklere dayanır. Aktif RFID sistemlerinde referans etiketleri kullanılırken, pasif sistemler daha çok ekipman ve araç takibinde tercih edilir. Faz bilgisi kullanılarak yüksek doğrulukta pasif RFID sistemleri geliştirilebilir. RFID konumlandırmanın avantajları arasında düşük maliyet, hafiflik ve pasif etiketlere sahip varlıkların takibine uygun olması bulunur. Ancak, uygulama için ek altyapı gerekebilir ve çoğu akıllı telefonun RFID desteği olmadığı için tüketici kullanımına uygunluğu sınırlıdır (Ho ve Chan, 2020).

2.2.4 Uwb (ultra wide band) iç mekân konumlandırması

İç mekânlarda konumlandırma yapabilmek için yaygınlaşmış tek bir yöntem yoktur. Genellikle Wi-Fi, Kızılötesi, RFID, UWB, Li-Fi, ZigBee, WLAN ve Bluetooth gibi teknolojiler GPS ile karşılaştırıldığında iç mekân konumlandırmalarında daha başarılıdır. UWB, mevcut dar bant veya yayılmış spektrumlu kablosuz iletişim yapısından tamamen farklıdır. UWB teknolojisi çok kısa darbelerin iletilmesinde kullanılır. Radyo enerjisinin, geniş bir frekans bandı üzerinde çok düşük bir güç spektral yoğunluğu ile yayılmasına imkan sağlayan teknikler kullanır. UWB sensörlerinin güç tüketimleri, diğer teknolojilere göre daha düşüktür. UWB darbelerinin düşük frekansı, sinyalin duvarlar ve nesneler gibi engellerden etkili bir şekilde geçmesini sağlar ve doğruluğu artırır. UWB, konumlandırma ve ölçüm mesafelerindeki hatayı santimetre seviyesine kadar düşürür. UWB teknolojisi, kritik konumlandırma uygulamaları için en uygun seçeneklerden biridir (Alışkan ve Şahin, 2023).

Ultra geniş bant (UWB) sensör teknolojisi, kısa menzilli ve yüksek bant genişliği iletişiminde kullanılan bir radyo teknolojisidir ve iç mekânlarda mesafe ölçümü, konum belirleme, takip ve navigasyon uygulamaları için uygundur. Yüksek bant genişliği sayesinde çoklu yol problemlerine karşı dayanıklıdır ve bu da sinyallerin farklı frekanslarda farklı zamanlarda iletilmesine olanak tanır. UWB vericileri diğer iç mekân

konumlandırma teknolojilerine göre daha düşük güç tüketirken, alıcılar biraz daha yüksek enerji harcar. İletişim frekansları 3.1-10.6 GHz aralığında olup, özellikle 2.4 GHz bandındaki Wi-Fi ve Bluetooth sinyallerinden etkilenmeyi azaltır. İnsan ve nesne yoğunluğunun yüksek olduğu iç mekânlarda görüş hattının (LOS) engellenmesi konum doğruluğunu etkileyebilir; ancak insan vücudunun sinyalleri emici etkisi bu sapmaları belirgin şekilde artırmayabilir. İç mekân konumlandırmada uçuş zamanı (ToF), varış zamanı (ToA), varış zamanı farkı (TDoA) ve varış açısı (AoA) gibi yöntemler kullanılır. UWB teknolojisi özel verici ve alıcı donanımı gerektirdiğinden bilişim pazarında henüz yaygın değildir ancak mobil cihazlarda kullanımının artması beklenmektedir. Tıbbi uygulamalarda özellikle TDoA yöntemi dikkat çekmektedir ve ToF ile TDoA teknikleri yaklaşık 30 cm hata payıyla hassas konumlandırma sağlamaktadır (Arsan, 2018).

2.2.5 Ultrasonik iç mekân konumlandırması

İnternetin gelişimiyle, şirketlerde iş süreci optimizasyonunda insanlar ve varlıkların konumu hakkındaki bilgiler daha önemli hale gelmektedir. Nesne koordinatlarını hesaplayan ve bunları işlenmek üzere özel platforma gönderen teknolojilerden birisi, ultrasonik iç mekân konumlandırmasıdır. Sistem, nesne konumunu birkaç santimetre yakınına kadar tanımlayabildiği için şirketlerin ve işletmelerin altyapı uygulamaları için büyük bir potansiyeldir (Rose, 2021).

Ultrasonik konumlandırma sistemi, artan ya da azalan sinyal iletim moduna dayanır. Kurulum sırasında alıcı cihazların sayısında sınır yoktur. Konumlandırma sırasında vericiler ve sinyal okuyucular arasında doğrudan görünürlük gerektirmez (Rose, 2021).

2.3 Literatür Araştırması

Kapalı mekân konumlandırma sistemleri, son yıllarda özellikle acil durumlarda kullanıcıların konumlarının belirlenmesi, nesne takibi ve otonom sistemlerde navigasyon gibi alanlarda yoğun olarak araştırılmaktadır. İlk çalışmalarda genellikle Wi-Fi ve RSSI tabanlı yöntemler kullanılmıştır. Bahl ve Padmanabhan (2000), radyo frekans sinyal gücünü kullanan ilk RF tabanlı konumlandırma sistemlerinden birini geliştirmiştir. Daha sonra Machaj, Brida ve Piché (2011), RSSI sıralama algoritmasıyla kalibrasyon gereksinimini azaltmış; Shrestha, Talvitie ve Lohan (2013a), eğitim veri tabanını

küçülterek performansı koruyan bir yöntem önermiş; Fang, Deng, Xue, Jiao, Zeng, Zheng ve Lu (2015) ise KNN tabanlı algoritmalarda doğruluğu artırmak için ağırlıklandırma yaklaşımı geliştirmiştir. Bu çalışmalar, UWB gibi yüksek doğruluk sağlayan teknolojilerin gelişimine zemin hazırlamıştır.

UWB (Ultra Wideband) teknolojisi, geniş bant yapısı sayesinde yüksek doğruluklu mesafe ölçümü sağlayarak TOA (Time of Arrival) yöntemleriyle trilaterasyon tabanlı konum belirlemede öne çıkmaktadır. He, Li, Pahlavan ve Wang (2012), TOA yöntemlerinin performansını detaylı olarak incelemiş ve doğruluğu etkileyen faktörleri analiz etmiştir. Rizos, Dempster, Li ve Salter (2007), kapalı mekân konumlandırma tekniklerini karşılaştırmış ve TOA yöntemlerinin avantajlarını vurgulamıştır.

Talvitie, Renfors ve Lohan (2015a), farklı frekans bantlarında TOA performansını deneysel olarak analiz etmiş ve 5 GHz bandında düşey yönde sinyal kaybının daha fazla olduğunu göstermiştir.

Talvitie, Renfors ve Lohan (2015b), TOA tabanlı sistemlerin tasarımında frekans seçiminin doğruluk üzerindeki kritik rolünü ortaya koymaktadır.

Konum doğruluğunu artırmak için filtreleme yöntemleri de yaygın olarak kullanılmaktadır. Park, Jung ve Han (2015), Kalman filtresi ile parçacık filtresinin hibrit kullanımını önermiş ve konum tahmininde doğruluğu artırmıştır. Wu, Jedari, Muscedere ve Rashidzadeh (2016) ise donanımdan bağımsız bir parçacık filtresi geliştirerek ±1.1 m ortalama hata ile yüksek doğruluk elde etmiştir.

Ayrıca, hibrit yöntemler de literatürde yer almaktadır. Khan, Kai ve Gul (2017), trilaterasyon ve parmak izi yöntemlerini birleştirerek hem düşük maliyet hem de yüksek doğruluk sağlayan bir yaklaşım sunmuştur. Li, Yang, Alam Bhuiyan ve Wang (2016) ise sinyal gücüne dayalı kontrolsüz öğrenme algoritması geliştirerek klasik menzil tabanlı algoritmalara kıyasla daha yüksek doğruluk elde etmiştir.

Bu çalışmalar incelendiğinde, UWB tabanlı TOA yöntemlerinin özellikle sabit alıcı istasyonlarla yapılan trilaterasyon uygulamalarında öne çıktığı görülmektedir. Ancak bu sistemlerin Python tabanlı masaüstü arayüzlerle gerçek zamanlı entegrasyonu ve görselleştirilmesi literatürde sınırlı yer bulmaktadır. Bu tez çalışması, üç adet UWB sabit

alıcı istasyon ve bir hareketli verici kullanarak TOA tabanlı trilaterasyon ile konum belirleme ve elde edilen verilerin Python tabanlı bir masaüstü arayüz üzerinden gerçek zamanlı görselleştirilmesini amaçlamaktadır.

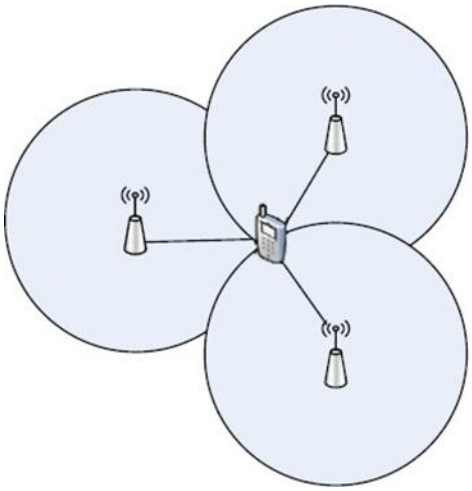
3. YÖNTEM

Bu tez çalışması, santimetre seviyesinde hassas sonuçlar verebilen ultra geniş bant (UWB) bir iç mekân konumlandırma sistemini sunmaktadır. Kapalı bir devre ağından oluşan UWB tabanlı konumlandırma sistemi, birden fazla varlığın konumunu belirlemek için çeşitli yöntemler kullanır. Öncelikle iki yönlü mesafe ölçümü (Two-Way Ranging – TWR) yöntemiyle hareketli cihaz (tag) ve sabit alıcılar (anchor) arasında sinyal gönderilip alınarak, sinyalin gidiş-dönüş süresi hesaplanır ve bu süre ışık hızına göre mesafeye dönüştürülür. Ayrıca sinyalin varış zamanı (Time of Arrival – ToA) ölçülerek cihazlar arası mesafe daha hassas şekilde belirlenir. Elde edilen mesafe verileri, trilaterasyon yöntemiyle işlenerek tag'in koordinat düzlemindeki konumu hesaplanır. ESP32 UWB Pro cihazı ise round robin algoritmasıyla her anchor ile sırayla haberleşerek aynı anda birden fazla tag'in konumunun çakışma olmadan takip edilmesini sağlar.

lk önce, sistemde verici ve alıcılarının, ekran kontrol terminalinin oluşturulması gerekmektedir. Alıcıların konumlandırılması, Varış Zamanı (ToA) ölçüm teknolojisine dayanır. Konumlandırmayı doğru bir şekilde belirlemek için en az üç alıcı ve bir verici kullanılmalı ve bu veriler trilaterasyon yöntemi ile işlenmelidir (Kwon, Yoo, Han ve Jang, 2023).

3.1 ToA (Varış Zamanı) ile Mesafe Ölçümü

ToA tekniğinde, sinyalin çıktığı kaynak ile sinyali alan alıcı arasındaki tek yönlü varış süresi hesaplanmaktadır. Sinyalin tek yönlü varış süresini bulabilmek için kaynak ile alıcının zaman açısından senkronize olmaları gerekmektedir. Kullanılan sinyalde yayılma hızı bilinirse, Varış Zamanı tekniğine göre, kaynak-alıcı arasındaki mesafe de belirlenmiş olur. 2 boyutlu uzayda 3 veya daha fazla alıcı kullanıldığında, hesaplanan mesafe bilgilerine göre, merkezde alıcıların bulunduğu 3 veya daha fazla çember oluşacaktır. Konumu hesaplanmak istenen kaynak, bu çemberlerin kesiştiği yerde bulunacaktır (Kurnaz, 2019).



Şekil 2. Varış Zamanı Tekniği Geometrisi (Özdemir, Ceylan, Alçay ve Yiğit, 2014).

ToA yöntemi kapsamında, mobil cihaz tarafından zaman damgalı sinyaller birden fazla sabit alıcıya iletilir. Alıcılar, sinyalin geliş zamanını kaydeder ve bu zaman bilgisi ile sinyalin yayılma hızı göz önünde bulundurularak, mobil cihaz ile alıcılar arasındaki mesafeler hesaplanır. Bu mesafe hesaplaması, sinyal transfer gecikmesi temel alınarak gerçekleştirilir (Özdemir vd., 2014).

ToA yöntemi, kablosuz yer belirleme sistemlerinde yaygın olarak kullanılan bir konum tahmin tekniğidir. Bu yöntemde, hedef cihaz tarafından gönderilen sinyalin bir alıcıya ulaşması için geçen süre ölçülür. Sinyalin yayılma hızı sabit olduğu için, ölçülen bu zaman gecikmesi sinyal kaynağı ile alıcı arasındaki mesafenin hesaplanmasında doğrudan kullanılabilir. Ancak gerçek ortam koşullarında, özellikle çoklu yol etkileri (yansımalar, kırılmalar gibi) ve ortam gürültüsü, ToA ölçümlerinin doğruluğunu olumsuz etkileyebilir. Bu nedenle, doğru mesafe tahmini için doğrudan yol (Line of Sight - LoS) sinyalinin tespiti kritik önem taşır. LoS sinyalinin belirlenmesinde çeşitli algoritmalar ve sinyal işleme teknikleri kullanılmaktadır. Yoğun çoklu yol ortamlarında bile, uygun filtreleme ve işleme yöntemleri kullanıldığında ToA tabanlı mesafe tahmini başarıyla uygulanabilir. Bu sayede, UWB (Ultra Wideband) sinyallerinin geniş bant özellikleri, ortamın karmaşık yapısına rağmen yüksek doğrulukta konum belirleme sağlar (Falsi, Dardari, Mucchi ve Win, 2006).

Ortam şartlarından ve ölçüm hatalarından kaynaklı, çemberlerin tek bir kesişim noktası olmayacaktır. Bu durum, optimizasyon problemine neden olacaktır. Ölçüm hatalarının olmadığı hallerde, i'inci alıcı için hesaplanan varış zamanı t_i olsun (Kurnaz, 2019).

$$t_i = \frac{d_i}{v}, i = 1,2, \dots, N \tag{3.1}$$

Denklem 3.1'de d_i kaynak ile i'nci sabit düğüm arasındaki mesafeyi göstermekteyken, v de sinyal hızını göstermektedir. Ölçüm hataları, sisteme eklendiğinde ise, t_i ile hesaplanan mesafe bilgisi $r_{toa,i}$ aşağıdaki denklem ile gösterilebilir (Kurnaz, 2019).

$$r_{toa,i} = d_i + n_{toa,i} = \sqrt{(x - x_1)^2 + (y - y_1)^2} + n_{toa,i}, i = 1,2, \dots, N \tag{3.2}$$

Yukarıdaki denklemde $n_{toa,i}$ i'inci hesaplamaya dahil edilen hatayı göstermektedir. Gauss dağılımının 0 ortalamalı olduğu varsayılmaktadır. İfadenin vektörel formu aşağıdaki şekilde yazılabilir (Kurnaz, 2019).

$$r_{toa} = f_{toa}(x) + n_{toa} \tag{3.3}$$

Gürültü ve gürültünün eklendiği mesafe vektörleri aşağıdaki gibi olacaktır (Kurnaz, 2019).

$$n_{toa} = \left[n_{toa,1} n_{toa,2} \dots n_{toa,N}\right]^T \tag{3.4}$$

$$r_{toa} = f_{toa}(x) + n_{toa} \tag{3.5}$$

Hesaplanmak istenen mesafe fonksiyonu aşağıdaki şekli alacaktır (Kurnaz, 2019).

$$f_{toax}(x) = \left[\sqrt{(x-x_1)^2 + (y-y_1)^2}, \sqrt{(x-x_2)^2 + (y-y_2)^2}, \dots, \sqrt{(x-x_N)^2 + (y-y_N)^2}\right] \tag{3.6}$$

Varış zamanları ile hesaplanan r_{toa}'ya mesafe bilgileri de eklenerek x değerinin hesaplanması yapılmaktadır. Varış Zamanı yönteminde ana sorun, kaynak-alıcılar arsındaki senkronizasyonun oluşabilmesidir. Senkronizasyon sorununu çözmek için sinyalin gidiş-dönüş sürelerinden yararlanılmaktadır (Kurnaz, 2019).

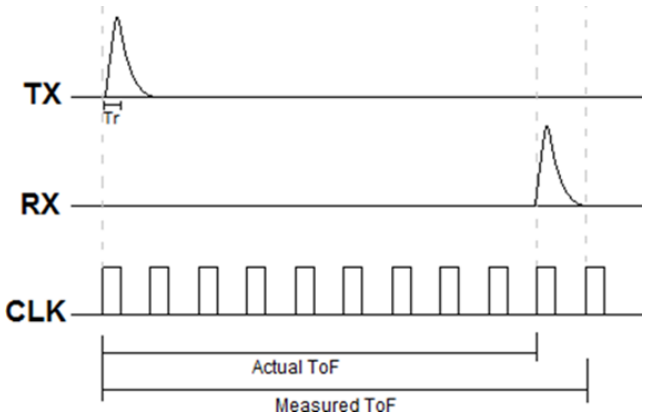
ToA (Zaman Varışı) ve ToF (Uçuş Süresi) yöntemlerinin karşılaştırılması, konum belirleme sistemlerinde doğruluk, maliyet ve uygulanabilirlik açısından en uygun tekniğin seçimine olanak tanır. ToA, senkronize saatler gerektirmesi ve multipath

etkilerine karşı hassasiyeti nedeniyle bazı senaryolarda sınırlı kalırken, ToF (özellikle çift yönlü ölçüm teknikleriyle) senkronizasyon ihtiyacını ortadan kaldırarak daha güvenilir sonuçlar sunar. Örneğin, Tschirschnitz ve Wagner (2019) tarafından yapılan analizler, ToF tabanlı DS-TWR yönteminin saat hatalarına karşı ToA'dan 200 kat daha dirençli olduğunu göstermiştir. Bu karşılaştırma, mühendislerin enerji verimliliği, donanım karmaşıklığı ve çevresel koşullar (NLoS, multipath) gibi kritik faktörlere göre sistem tasarımı yapabilmesi için gereklidir. Ayrıca, ToF'nin otonom araçlar veya endüstriyel robotik gibi yüksek doğruluk gerektiren uygulamalardaki üstünlüğü, bu analizin pratik önemini vurgulamaktadır (Von Tschirschnitz, Wagner, Pahl ve Carle, 2019).

3.1.1 Toa ve tof yöntemlerinin karşılaştırılması

Uçuş Zamanı (Time of Flight - ToF) ve Varış Zamanı (Time of Arrival - ToA) teknikleri, zaman bazlı konum belirleme sistemlerinde sıkça kullanılan iki yöntemdir. Her iki yöntemde de sinyalin yayılma süresi üzerinden mesafe hesaplanır. Ancak uygulama şekilleri ve gereksinimleri farklıdır:

- ToF (Time of Flight), bir sinyalin gidiş-dönüş süresini ölçer. Yani verici tarafından gönderilen sinyal, alıcı tarafından yansıtıldıktan sonra tekrar vericiye döner. Bu yöntem, radar sistemlerinde olduğu gibi iki yönlü haberleşme gerektirir ve genellikle yüksek zaman senkronizasyonu hassasiyeti ister. Saat sapmaları (clock drift) ciddi hatalara neden olabilir ve genellikle sistemde hem verici hem de alıcıda hassas zaman ölçüm birimleri gerektirir (Pereira, 2016).

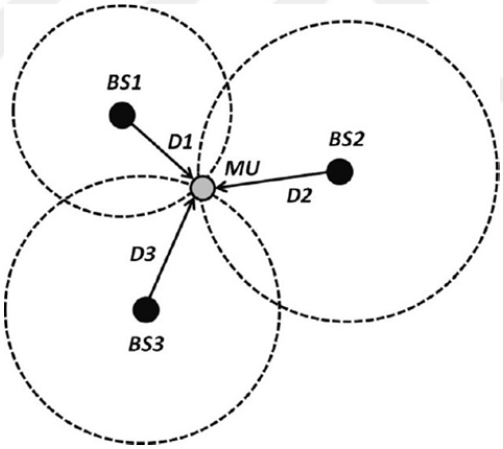


Şekil 3. Uçuş Zamanı Sistemlerinde Zaman Çözünürlüğü (Pereira, 2016).

- ToA (Time of Arrival) ise sinyalin tek yönlü varış zamanını ölçerek mesafeyi belirler. Bu yöntemde genellikle verici sabit zaman bilgisiyle sinyal gönderir ve alıcı cihazlar bu sinyalin geliş zamanını ölçer. ToA yöntemi, uygulama açısından daha az karmaşıktır, çünkü tek yönlü iletişim yeterlidir ve ESP32 UWB modülleri ile bu yapı doğrudan uygulanabilir. Ancak doğru konumlama yapılabilmesi için alıcı cihazlar arasında yüksek düzeyde senkronizasyon şarttır (Kurnaz, 2019).

3.2 Trilaterasyon (Üçgenleme) ile Konum Belirleme

Trilaterasyon, bilinmeyen bir noktanın konumunu belirlemek için sabit noktalara olan mesafeleri kullanan geometrik bir yöntemdir. Bu yöntem, iki boyutlu (2B) ve üç boyutlu (3B) uzayda farklı matematiksel yaklaşımlarla uygulanabilir. İdeal koşullarda, üç sabit alıcıdan (anchor) ölçülen mesafeler, her alıcıyı merkez kabul eden çemberlerin kesişim noktasını verir (Şekil 4) (Ambrosio, 2021).



Şekil 4. Üç Çemberin İdeal Kesişimi ve Pratikteki Örtüşme Alanı (Ambrosio, 2021).

2B düzlemde bir noktanın konumunu belirlemek için en az üç sabit alıcı gereklidir. Bu durumda, alıcılar ile hesaplanan mesafeler kullanılarak çemberlerin kesişim noktası bulunur. Üç sabit alıcı için Denklem 3.7–3.14 lineer bir sistem oluşturarak çözüm sağlar (Kwon vd., 2023).

$$A = -2 \times x_1 + 2 \times x_2 \tag{3.7}$$

$$B = -2 \times y_1 + 2 \times y_2 \tag{3.8}$$

$$C = D_1^2 - D_2^2 - x_1^2 + x_2^2 - y_1^2 + y_2^2 \tag{3.9}$$

$$D = -2 \times x_2 + 2 \times x_3 \tag{3.10}$$

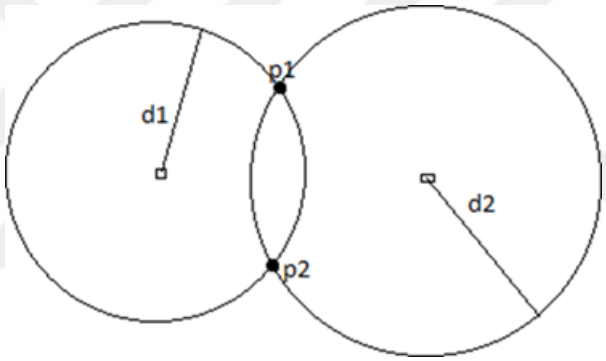
$$E = -2 \times y_2 + 2 \times y_3 \tag{3.11}$$

$$F = D_2^2 - D_3^2 - x_2^2 + x_3^2 - y_2^2 + y_3^2 \tag{3.12}$$

$$x_{Tag} = \frac{(C \times E) - (F \times B)}{(E \times A) - (B \times D)} \tag{3.13}$$

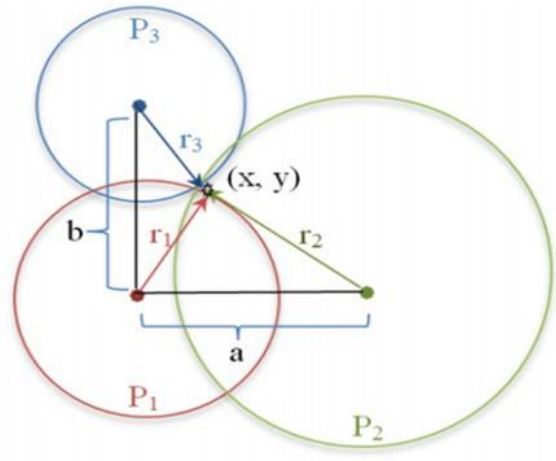
$$y_{Tag} = \frac{(C \times D) - (A \times F)}{(B \times D) - (A \times E)} \tag{3.14}$$

Sabit alıcılar, 2B düzlemde doğrusal olmayan (örneğin eşkenar üçgen) bir geometriye yerleştirilmelidir. Bu düzen, konumlama hatalarını minimize etmede kritik öneme sahiptir. İki alıcı durumundaki konum belirsizliği (Şekil 5), teorik olarak iki olası çözüm noktası üretir ve bu belirsizlik ancak üçüncü bir alıcı eklenerek giderilebilir (Pereira, 2016).



Şekil 5. İki Alıcılı Sistemde Oluşan Konum Belirsizliği (Pereira, 2016).

Alıcıların fiziksel konumları bilinmiyorsa, göreceli mesafelerle sanal koordinatlar üretilebilir; ancak bu yöntem ağ yoğunluğu yetersiz olduğunda hata birikimine yol açabilir (Ambrosio, 2021).



Şekil 6. Üç Alıcılı Sistemde Kesişim Noktasının Belirlenmesi (Masmanacı, 2018).

Bu Python kodu, üç sabit alıcı (IS1(0,0), IS2(2000,0) ve IS3(1000,1732.05)) kullanarak 2B trilaterasyon işlemi gerçekleştirir. Kod öncelikle alıcılara olan mesafelerin mutlak değerlerini alır (r_1, r_2, r_3), ardından lineer denklem sistemi oluşturur: $A = 2 * (x_2 - x_1)$, $B = 2 * (y_2 - y_1)$ ile IS1-IS2 eksen farkları, $C = r_1^2 - r_2^2 - x_1^2 + x_2^2 - y_1^2 + y_2^2$ ile mesafe kareleri farkını hesaplar.

Benzer şekilde, $D = 2 * (x_3 - x_2)$, $E = 2 * (y_3 - y_2)$ ve $F = r_2^2 - r_3^2 - x_2^2 + x_3^2 - y_2^2 + y_3^2$ ile IS2-IS3 çiftini modelleyerek $x = (C * E - F * B)/(A * E - B * D)$ ve $y = (C * D - A * F)/(B * D - A * E)$ formülleriyle hedef koordinatları belirler. Hesaplanan konumlar, alıcıların oluşturduğu üçgenin fiziksel sınırlarına (x: 0-2000 cm, y: 0-1732.05 cm) uygun şekilde kırpılır. Hata durumunda try-except bloğu devreye girerek hata mesajı verir. Bu yöntem, UWB tabanlı konum belirleme sistemlerinde endüstriyel uygulamalar için yüksek doğruluk sağlar.

```
def trilateration_raw(self, r1, r2, r3):
    """Üç sabit noktadan mesafe ölçümüyle konum belirleme"""
    try:
        r1, r2, r3 = abs(r1), abs(r2), abs(r3)
        x1, y1 = 0, 0            # IS1 (orijin noktası)
```

```
x2, y2 = 2000, 0            # IS2 (X ekseni üzerinde)

x3, y3 = 1000, 1732.05      # IS3 (Y ekseni referansı)

# Trilaterasyon formülleri

A = 2 * (x2 - x1)

B = 2 * (y2 - y1)

C = r1**2 - r2**2 - x1**2 + x2**2 - y1**2 + y2**2

D = 2 * (x3 - x2)

E = 2 * (y3 - y2)

F = r2**2 - r3**2 - x2**2 + x3**2 - y2**2 + y3**2

x = (C * E - F * B) / (A * E - B * D)

y = (C * D - A * F) / (B * D - A * E)

# Sınır kontrolleri

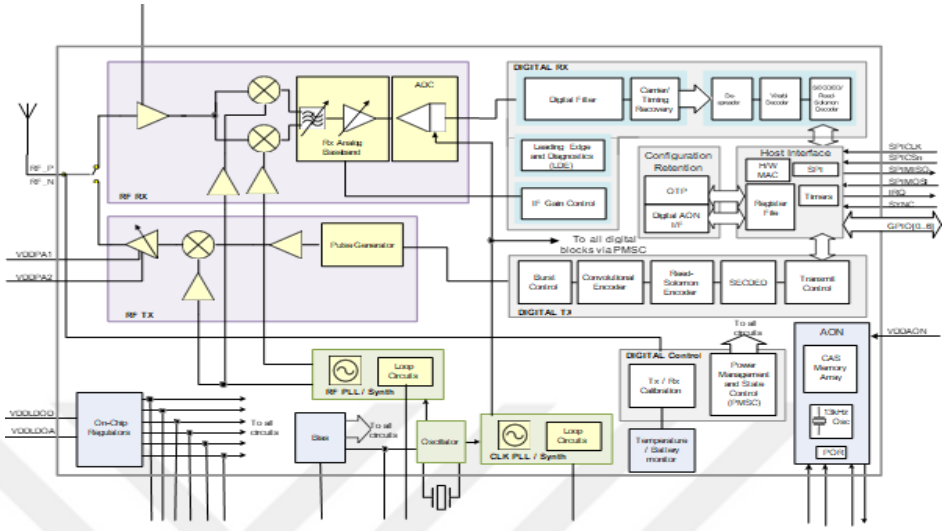
x = max(0, min(x, 2000))

y = max(0, min(y, 1732.05))

return x, y
```

3.3 Esp32 Uwb Cihazı ile ToA Uygulaması

3.3.1 Donanımsal mimarisi ve çalışma prensibi



Şekil 7. IC Blok Diyagramı (Decawave Ltd., 2013).

DW1000 entegresi, IEEE 802.15.4-2011 UWB standardına uyumlu, düşük güç tüketimli ve tek çipli bir CMOS RF alıcı-verici entegresidir. Bu yonga, analog ön uç (alıcı ve verici devresi) ile dijital arka uç olmak üzere iki ana bölümden oluşur. Dijital arka uç, harici bir işlemci ile SPI üzerinden haberleşir. Anten portuna bağlanan TX/RX anahtarlama devresi, verici veya alıcının aktif hale gelmesini sağlar. Ayrıca yonga üzerinde sıcaklık ve gerilim denetleyicileri de entegredir (Decawave Ltd., 2013).

Alıcı birimi, gelen RF sinyalini düşük gürültülü bir yükselteç (LNA) aracılığıyla kuvvetlendirdikten sonra doğrudan baseband seviyesine indirir. Alıcı devresi, yüksek bant genişliği, doğrusal çalışma ve düşük gürültü faktörü göz önünde bulundurularak optimize edilmiştir. Böylece farklı UWB kanallarının minimum bozulma ve ek gürültü ile çözümlenmesi sağlanır. Baseband'e indirgenen sinyal, demodülasyon işleminin ardından, SPI arayüzü aracılığıyla dış işlemciye aktarılır (Decawave Ltd., 2013).

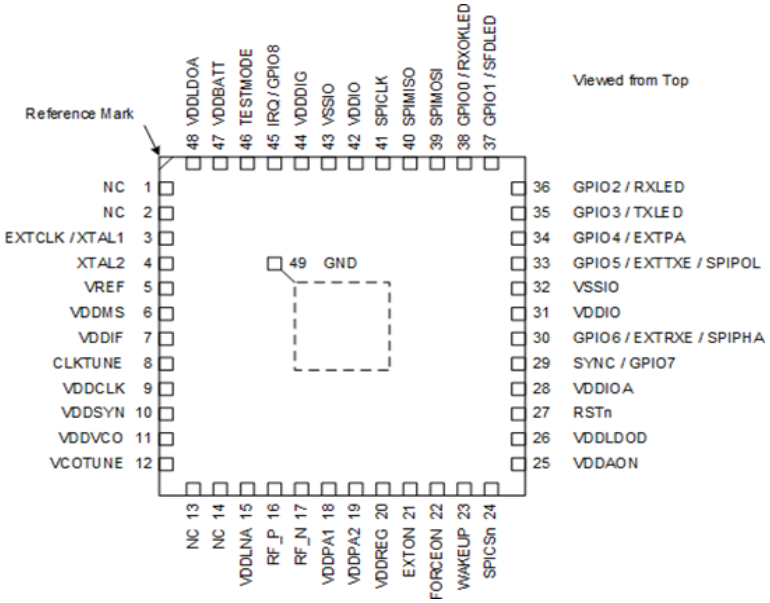
Verici devresi ise, dijital olarak kodlanmış verileri analog bir darbe üreteci aracılığıyla darbeler hâline getirir. Bu darbeler, çift dengeli bir mikser yardımıyla uygun taşıyıcı frekansa (belirlenen UWB kanalına) yükseltilir ve ardından güçlendirilerek dış antene iletilir (Decawave Ltd., 2013).

DW1000 ayrıca OTP (One-Time Programmable) bellek barındırır. Bu hafıza, kalibrasyon verileri (örneğin verici gücü, kristal frekans ayarı ve mesafe düzeltme katsayıları) için kullanılır ve gerekli olduğunda otomatik olarak yüklenebilir. Ayrıca yonga içerisinde yer alan Always-On (AON) bellek, en düşük güç modlarında bile yapılandırma verilerinin korunmasını sağlar (Decawave Ltd., 2013).

Sistem saatleme yapısı üç temel bileşen üzerine kuruludur: kristal osilatör, dijital PLL (CLKPLL) ve RF PLL. Bu yapı 38.4 MHz'lik harici bir kristal ya da sabit saat sinyali ile çalışacak şekilde tasarlanmıştır. CLKPLL, dijital işleme devrelerine gerekli olan saat sinyalini üretirken; RF PLL ise alıcı ve verici zincirlerinde yer alan yerel osilatörleri (LO) oluşturur. Uyku modlarında kullanılmak üzere 13 kHz'lik dahili bir osilatör de yonga içinde mevcuttur (Decawave Ltd., 2013).

Son olarak, DW1000 üzerinde sadece slave modda çalışan bir SPI haberleşme arayüzü bulunmaktadır. Ayrıca MAC katmanında CRC üretimi, hata kontrolü ve veri çerçevesi filtreleme gibi bazı temel protokoller yonga üzerinde donanımsal olarak desteklenmektedir (Decawave Ltd., 2013).

QFN-48 paket tipine sahip olan entegrenin pin konfigürasyonları aşağıdaki gibidir (Alışkan ve Şahin, 2023):



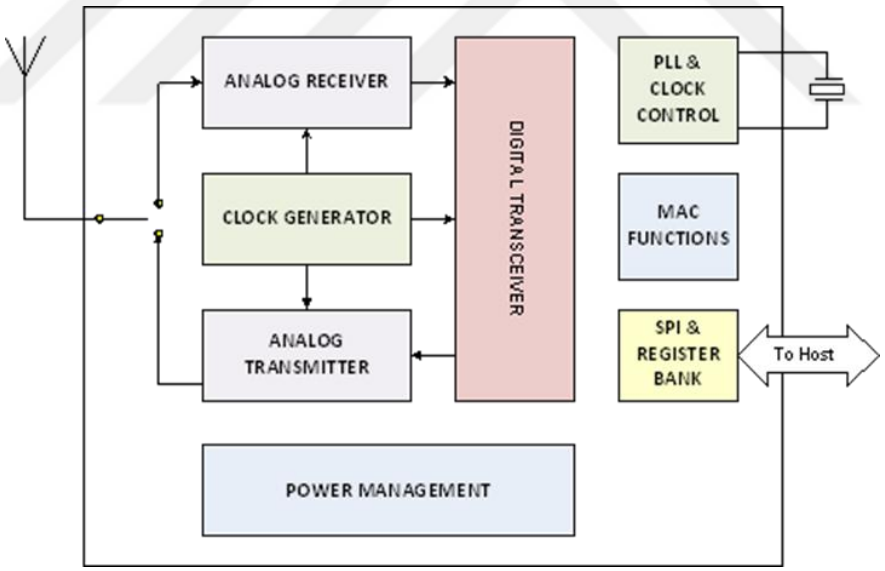
Şekil 8. DW1000 Pin Atamaları (Decawave Ltd., 2013).

3.3.2 Teknik özellikler ve uygulama alanları

Bu entegre, gerçek zamanlı konum belirleme sistemlerinde (RTLS) iç mekânlarda yaklaşık 10 cm hassasiyetle nesnelerin yerini tespit etmeye olanak sağlar (Decawave Ltd., 2017).

Yüksek veri hızlarına (6.8 Mb/s'ye kadar) destek verirken, koherent alıcı teknikleri sayesinde 300 metreye kadar etkili iletişim mesafesi sunmaktadır. Kısa paket süreleri, yoğun etiket ortamlarında (örneğin 20 metre yarıçapta 11.000 etiket) sistemin verimli çalışmasını sağlar. Ayrıca, çoklu yol (multipath) sönümlemelerine karşı yüksek dayanıklılığı sayesinde, sinyalin zayıfladığı ortamlarda güvenilir iletişim imkanı sunar (Decawave Ltd., 2017).

Düşük güç tüketimi sayesinde uzun süreli pil kullanımı mümkün olurken, küçük paket boyutu ve düşük çözüm alanı RTLS ve kablosuz sensör ağlarında (WSN) maliyet etkin uygulamalara olanak tanır (Decawave Ltd., 2017).



Şekil 9. Fonksiyonel Blok Diyagramı (Decawave Ltd., 2017).

Başlıca avantajları şunlardır:

- Etiketlenen nesnelerin yüksek hassasiyetle konumlandırılması, işletme verimliliğini artırır ve maliyetleri azaltır.

- Doğrudan görüş (LOS) ve görüş olmadan (NLOS) uzun çalışma mesafeleri sayesinde altyapı ihtiyacını azaltır.

- Düşük çip maliyeti, ekonomik çözümlerin uygulanmasını kolaylaştırır.

- Düşük güç tüketimi, pil değişim sıklığını azaltarak sistem ömrünü uzatır.

- IEEE 802.15.4-2011 standardına uygunluğu, yaygınlaştırmayı ve entegrasyonu kolaylaştırır (Decawave Ltd., 2017).

Hedef uygulamaları ise şunlardır:

- Tarım

- Bina kontrolü ve otomasyonu

- Fabrika otomasyonu

- Sağlık hizmetleri

- Güvenlik

- Depo ve lojistik (Decawave Ltd., 2017).

3.3.3 Veri iletim hızları, güç tüketimi ve çalışma frekansları

- 110 kbit/s, 850 kbit/s ve 6.8 Mbit/s veri iletim hızlarını destekler.

- 3.5 GHz ile 6.5 GHz arasında değişen 6 farklı frekans bandını kullanabilir.

- Gönderim gücü −14 dBm veya −10 dBm seviyelerindedir.

- Transmit güç yoğunluğu < −41.3 dBm/MHz olarak sınırlandırılmıştır.

- Preamble (ön ek) uzunluğu 64 mikro saniyeden 4 milisaniyeye kadar ayarlanabilir.

- Maksimum 1023 byte boyutunda paketleri destekler.

- BPM modülasyonu ile BPSK tekniklerini kullanır.

- Entegre edilmiş İleri Hata Düzeltme (FEC) ve Döngüsel Artıklık Kontrolü (CRC) mekânizmalarına sahiptir.

- Maksimum 20 MHz hızında SPI arayüzü ile mikrodenetleyicilere kolayca entegre edilebilir.

- Geniş bir mikrodenetleyici yelpazesiyle uyumludur.

- Tek besleme gerilimi aralığı 2.8 V ile 3.6 V arasındadır.

- Düşük güç tüketimi sunar:

- Gönderim modunda yaklaşık 31 mA,

- Alım modunda yaklaşık 64 mA,

- Gözetleme zamanlayıcısı modunda 2 µA,

- Derin uyku modunda ise 100 nA civarındadır.

- Çoklu erişim tekniklerini destekler:

- Frekans Bölmeli Çoklu Erişim (FDMA) ile 6 kanal,

- Kod Bölmeli Çoklu Erişim (CDMA) ile 12 farklı kanal kodu kullanabilir.

- Hem iki yönlü (two-way) hem de tek yönlü (one-way) mesafe ölçümlerini, Uçuş Süresi (Time of Flight - ToF) ve Varış Zamanı Farkı (Time Difference of Arrival - TDoA) yöntemleriyle gerçekleştirebilir.

- 90 nm CMOS teknolojisi ile üretilmiştir.

- Çalışma sıcaklığı aralığı -40°C ile +85°C arasında endüstriyel standartlara uygundur.

- 6 mm x 6 mm boyutlarında, 48 pinli QFN paketinde sunulur.

- DecaWave tarafından sağlanan donanım ve yazılım destek materyalleri mevcuttur (Decawave Ltd., 2017).

3.4 Udp Üzerinden Veri İletimi

3.4.1 Sabit istasyon (anchor) ayarları

Sistemde bulunan üç sabit istasyonun birbirinden farklı adreslere sahip olması gerekmektedir (MiSERY, 2021).

#define ANCHOR_ADD "7D:00:22:EA:82:60:3B:9C" // IS1 MAC ID

#define ANCHOR_ADD "7D:00:22:EA:82:60:3B:9A" // IS2 MAC ID

#define ANCHOR_ADD "7D:00:22:EA:82:60:3B:9B" // IS3 MAC ID

Kullanılan kütüphane varsayılan olarak rastgele kısa adresler atar. Ancak belirlenen statik adreslerin kullanılabilmesi için, UWB modülünün Anchor moduna alınması, LONGDATA_RANGE_LOWPOWER çalışma modunun seçilmesi ve rastgele kısa adreslerin devre dışı bırakılması gerekmektedir (MiSERY, 2021).

// Sabit istasyon olarak başlat

DW1000Ranging.startAsAnchor(ANCHOR_ADD, DW1000.MODE_LONGDATA_RANGE_LOWPOWER, false);

3.4.2 Hareketli verici (tag) işleyişi

Hareketli verici, sabit istasyonlar arasındaki mesafe verilerini okuyarak UDP protokolü üzerinden bilgisayara iletir (MiSERY, 2021).

// Ağ ayarları

const char* ssid = "BatuhanCep";

const char* password = "gozo4482";

WiFi bağlantısı DHCP ile otomatik IP atayacak şekilde yapılandırılmıştır. Bağlantı kurulduğunda, hareketli vericinin IP adresi seri monitörde (Serial Monitor) görüntülenir (MiSERY, 2021).

```
IPAddress local_IP(192, 168, 1, 34);

IPAddress gateway(192, 168, 1, 1);

IPAddress subnet(255, 255, 255, 0);

IPAddress primaryDNS(8, 8, 8, 8);

IPAddress secondaryDNS(8, 8, 4, 4);
```

UWB modülü Tag modunda çalışacak şekilde ayarlanır (MiSERY, 2021).

```
#define TAG_ADD "86:17:5B:D5:A9:9A:E2:9C" // Verici MAC
```

Algılanan sabit istasyonların verileri, MAC:AA:BB:CC,IS1:3.14,IS2:2.5 formatında virgülle ayrılmış bir string olarak bilgisayara gönderilir. Veri iletimi, genellikle her saniye UDP protokolü üzerinden gerçekleştirilir (MiSERY, 2021).

```
void loop() {
   DW1000Ranging.loop();
   bildirimleriDinle();
   switch (suankiDurum) {
      case IS1_OLCUM:
      case IS2_OLCUM:
      case IS3_OLCUM:
         if (millis() - durumBaslangicZamani > durumZamanAsimi) {
            sonrakiIstasyonaGec();
         }
         break;
```

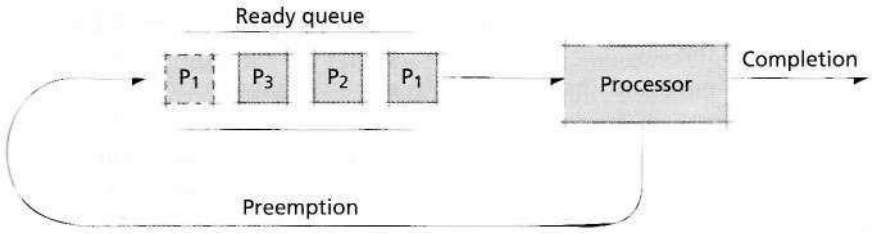
```
    case HAZIR:
        break;
}
```

3.4.3 Round robin algoritması

Round Robin (RR) zamanlama algoritması, işlemlerin İlk Giren İlk Çıkar (FIFO) mantığıyla sıraya alınması esasına dayanır; ancak her işleme sınırlı bir işlem süresi (quantum ya da zaman dilimi) verilir. Eğer bir işlem, kendisine tanınan bu süre içerisinde tamamlanmazsa, sistem bu işlemi duraklatır ve sıradaki bekleyen işleme işlemciyi devreder. Tamamlanmayan işlem ise hazır kuyruğunun sonuna gönderilerek daha sonra yeniden çalıştırılır (Silberschatz, Galvin ve Gagne, 2018).

Bu algoritma özellikle kullanıcı etkileşiminin ön planda olduğu sistemlerde, hızlı yanıt sürelerinin sağlanması açısından oldukça etkilidir. Bağlam değiştirme işleminin sistem üzerindeki yükünü azaltmak adına, bekleyen işlemlerin ana bellekte tutulması ve geçişlerin verimli gerçekleştirilmesi önerilir (Silberschatz vd., 2018)

Round Robin yöntemi, genellikle daha gelişmiş zamanlama algoritmalarının bir bileşeni olarak yer alır; tek başına ana algoritma olarak nadiren tercih edilir. Tüm işlemlerin eşit önceliğe sahip olduğu durumlarda ise, bu gelişmiş algoritmalar sıklıkla FIFO ya da Round Robin mantığına indirgenebilir. Bu nedenle, POSIX gerçek zamanlı sistemler standardı, Round Robin'i desteklenmesi gereken üç zamanlama algoritmasından biri olarak tanımlar (Silberschatz vd., 2018)



Şekil 10. Round Robin Algoritması Zamanlama Şeması (Silberschatz vd., 2018).

Aşağıdaki fonksiyon, hareketli vericinin sabit istasyonlar (IS1, IS2 ve IS3) ile sırasıyla iletişim kurmasını sağlayan Round Robin algoritmasının çekirdek yapısını oluşturmaktadır. Bu yapı sayesinde her ölçümden sonra bir sonraki istasyona geçiş yapılır ve tüm istasyonlarla sırayla haberleşme sağlanır. Üçüncü istasyon olan IS3 ile iletişim tamamlandığında, veriler UDP üzerinden iletilmekte ve döngü yeniden IS1 ile başlamaktadır. Böylece sabit istasyonlarla düzenli ve dengeli bir veri alışverişi gerçekleştirilir.

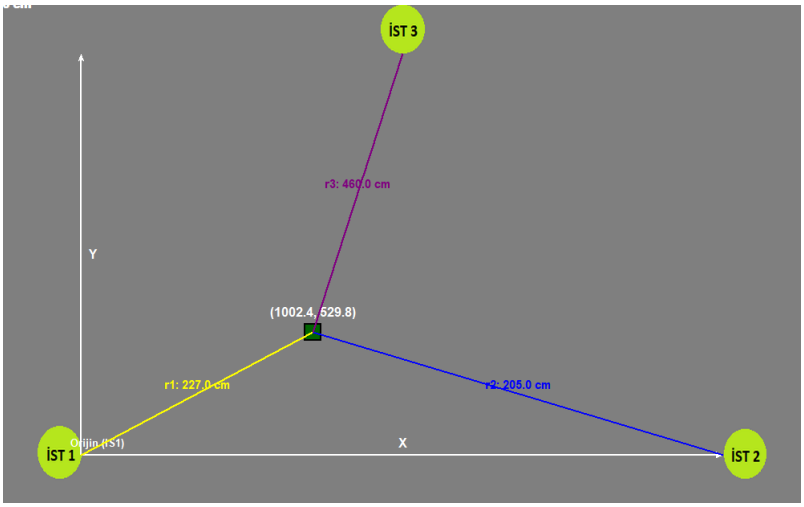
```
void sonrakiIstasyonaGec() {
  switch (suankiDurum) {
    case IS1_OLCUM:
      suankiDurum = IS2_OLCUM;
      Serial.println("IS1 ölçümü tamamlandı, IS2'ye geçiliyor...");
      break;
    case IS2_OLCUM:
      suankiDurum = IS3_OLCUM;
      Serial.println("IS2 ölçümü tamamlandı, IS3'e geçiliyor...");
      break;
    case IS3_OLCUM:
      suankiDurum = IS1_OLCUM;
      Serial.println("IS3 ölçümü tamamlandı, IS1'e geçiliyor...");
      verileriGonder(false);
      break;
```

```
        case HAZIR:
            suankiDurum = IS1_OLCUM;
            break;
    }
    durumBaslangicZamani = millis();
}
```

3.4.4 Python ile veri işleme ve görselleştirme

Python ortamında geliştirilen bu uygulama, ESP32 UWB cihazlarından UDP üzerinden gelen mesafe verilerini alır, trilaterasyon yöntemiyle konum hesaplaması yapar ve grafiksel olarak ekranda gösterir (MiSERY, 2021).

Başlangıçta veri görselleştirme için Matplotlib kütüphanesi değerlendirilmiş ancak gerçek zamanlı güncelleme performansı ve Tkinter entegrasyonunun karmaşıklığı sebebiyle vazgeçilmiştir. Bunun yerine, customtkinter'ın sağladığı CTkCanvas kullanılarak daha hafif ve etkili bir çözüm sunulmuştur. Canvas üzerinde hareketli verici (yeşil kutu), mesafe çizgileri ve eksenler doğrudan çizilebilmiştir (MiSERY, 2021).



Şekil 11. CustomTkinter ile Geliştirilen UWB Konum Takip Arayüzü.

Konum hesaplamasında üç sabit istasyon (IS1, IS2, IS3) kullanılarak standart trilaterasyon formülleri uygulanmıştır. Her istasyonun fiziksel konumları (IS1: (0,0), IS2: (2000,0), IS3: (1000,1732.05) cm) sabit olarak tanımlanmış ve gelen mesafe verileriyle (r1, r2, r3) cihazın (x,y) koordinatları hesaplanmıştır. Helen formülü yerine doğrudan trilaterasyon denklemleri kullanılarak hesaplama basitleştirilmiştir.

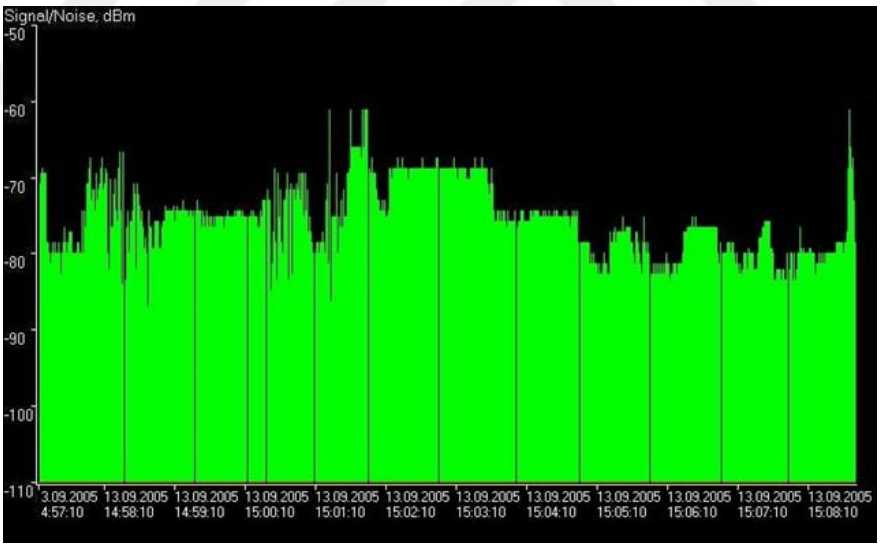
```
def trilateration_raw(self, r1, r2, r3):
    try:
        r1, r2, r3 = abs(r1), abs(r2), abs(r3)
        x1, y1 = 0, 0            # IS1 (0, 0)
        x2, y2 = 2000, 0         # IS2 (2000 cm, 0)
        x3, y3 = 1000, 1732.05     # IS3 (1000, 2000*√3/2 ≈ 1732.05 cm)
        A = 2 * (x2 - x1)
        B = 2 * (y2 - y1)
        C = r1**2 - r2**2 - x1**2 + x2**2 - y1**2 + y2**2
        D = 2 * (x3 - x2)
        E = 2 * (y3 - y2)
        F = r2**2 - r3**2 - x2**2 + x3**2 - y2**2 + y3**2
        x = (C * E - F * B) / (A * E - B * D)
        y = (C * D - A * F) / (B * D - A * E)
        # Konum sınırlarını kontrol et
        x = max(0, min(x, 2000))
        y = max(0, min(y, 1732.05))
```

return x, y

3.5 Rssı ile Mesafe Hesabı ve Denklemler

RSSI, modül tarafından alınan mutlak gücü hesaplamak için işleme ihtiyaç duyan göreceli bir güç göstergesidir ve üreticiler genellikle 0 ile 255 arasında değişen RSSI değeriyle ilişkili alınan gücü dBm cinsinden belirtir; burada "0" en güçlü sinyali ifade eder. Alınan güç, iletilen güç ve çeşitli kayıplarla ilişkilidir; bu kayıplar empedans uyumsuzlukları, polarizasyon hataları veya yayılım kaynaklı olabilir. Bu çalışmada antenler eşlenmiş ve karta doğrudan bağlanmış olduğundan empedans, polarizasyon ve bağlantı kayıpları ihmal edilmiştir; dolayısıyla yalnızca yayılım kaybı dikkate alınmıştır (Tamer, 2023).

Şekil 12'de görülebileceği üzere, bir WLAN sisteminde ölçülen RSSI değerleri dBm birimiyle ifade edilmektedir. Yapılan gözlemler, sinyal gücündeki dalgalanmaların 20 dB seviyesine ulaşabildiğini ve bu değerin yaklaşık 100 katlık bir farka tekabül ettiğini ortaya koymaktadır (Pereira, 2016).



Şekil 12. WLAN Ağında RSSI'nin Gelişimi (Pereira, 2016).

Sinyal Geliş Gücü Tekniği, verici tarafından yayımlanan sinyalin alıcı birimde tespit edilen şiddetinin analiz edilmesi yoluyla uzaklık tahmini gerçekleştirme prensibine dayanmaktadır. Belirli ve değişmeyen bir iletim gücüne sahip olan sinyal, içinde bulunduğu ortamın özelliklerine bağlı olarak alıcıya olan mesafe arttıkça güç kaybına uğramaktadır (Kurnaz, 2019).

i indeksli alıcı düğüm için iletilen sinyal gücü matematiksel olarak modellenebilmektedir:

$$P_i^r = K_i \frac{P_i^t}{d_i^a}, i = 1,2, \dots, N \quad (3.15)$$

Denklem 3.7'de yer alan P^t_i parametresi, kaynağın ilettiği sinyal gücünü ifade ederken, d_i sembolü, verici ile alıcı arasındaki uzaklığı göstermektedir. K_i katsayısı, sistem özelliklerine bağlı sabit bir değer olup, a parametresi ise ortamın zayıflama katsayısını temsil etmektedir. Ortam zayıflama katsayısı a, tipik olarak 2 ile 5 arasında değerler almakta olup, ideal serbest uzay koşullarında bu değer 2 olarak belirlenmiştir. Ölçümlerde karşılaşılan hataları modellemek amacıyla, ortam zayıflama katsayısı log-normal dağılım gösteren bir modelle ifade edilebilir (Kurnaz, 2019).

$$\ln(P_i^r) = \ln(K_i) + \ln(P_i^t) - \alpha \ln(d_i) + n_{rss,i}, i = 1,2, \dots, N \quad (3.16)$$

Denklem 3.16'da kullanılan $n_{rss,i}$ değişkeni, sıfır ortalamalı Gauss gürültü dağılımını karakterize etmektedir. Sinyal zayıflaması ile mesafe arasındaki matematiksel ilişki, mevcut parametreler kullanılarak yeniden formüle edilebilmektedir (Kurnaz, 2019).

$$r_{rss,i} = \ln(P_i^r) - \ln(P_i^t) + \ln(K_i) \quad (3.17)$$

Denklem 3.17 yardımıyla, Denklem 3.8'in daha sade bir forma dönüştürülmesi mümkündür (Kurnaz, 2019).

$$r_{rss,i} = -\alpha \ln(d_i) + n_{rss,i}, i = 1,2, \dots N \tag{3.18}$$

Çoklu alıcı düğümlerinin bulunduğu sistemlerde, bu matematiksel ifadeler vektörel gösterimle yazılabilmektedir (Kurnaz, 2019).

$$r_{rss} = f_{rss}(x) + n_{rss} \tag{3.19}$$

Gürültü bileşenleri ve gürültü eklenmiş mesafe ölçümleri, vektörel formda ifade edilebilmektedir (Kurnaz, 2019).

$$n_{rss} = \left[n_{rss,1} n_{rss,2} \dots n_{rss,N}\right]^T \tag{3.20}$$

$$r_{rss} = \left[r_{rss,1} r_{rss,2} \dots r_{rss,N}\right]^T \tag{3.21}$$

Konum tahmini için kullanılacak olan amaç fonksiyonu, bu şekilde tanımlanabilmektedir (Kurnaz, 2019).

$$f_{rss}(x) = -\alpha \left[\ln\left(\sqrt{(x-x_1)^2+(y-y_1)^2}\right) \ldots \ln\left(\sqrt{(x-x_N)^2+(y-y_N)^2}\right)\right]^T \quad (3.22)$$

Denklem 3.21'in çözümü sonucunda, ölçülen r_{rss} değerleri kullanılarak x konum vektörünün belirlenmesi mümkün olmaktadır. Bu işlem, alıcı düğümlerden elde edilen sinyal gücü ölçümlerinin matematiksel olarak işlenmesiyle hedef konumun en doğru şekilde tahmin edilmesini sağlamaktadır (Kurnaz, 2019).

3.6 Esp32 Uwb'de Rssı Ölçüm Aralığı

Tablo 1. Alıcı (RX) Duyarlılık Değerleri (Decawave Ltd., 2017).

Veri Hızı	**PRF (MHz)**	**Preamble Uzunluğu**	**Hassasiyet (1% PER)**	**Hassasiyet (10% PER)**
110 kbps	16	2048 sembol	-106 dBm	-107 dBm
850 kbps	16	1024 sembol	-101 dBm	-102 dBm
6.8 Mbps	16	256 sembol	-93 dBm	-94 dBm
110 kbps	64	2048 sembol	-102 dBm	-106 dBm
6.8 Mbps	64	256 sembol	-97 dBm	-98 dBm

DW1000'nin alıcı duyarlılık değerlerini gösteren **Tablo 1**, çipin farklı çalışma modlarındaki performansını anlamak için kritik bir referanstır. Bu tablo, belirli paket hata

oranlarını (PER) sağlamak için gereken minimum alınan sinyal gücünü dBm cinsinden sunar (Decawave Ltd., 2017).

Tablonun ilk sütununda veri hızları listelenmiştir. 110 kbps en düşük hız olmasına rağmen en yüksek alıcı duyarlılığını (-106 dBm) sunar. Bu, uzun mesafe iletişimi için ideal bir seçimdir. 850 kbps ve 6.8 Mbps gibi daha yüksek veri hızlarında ise duyarlılık sırasıyla -101 dBm ve -93 dBm'e düşer. Bu durum, hız ve menzil arasındaki klasik trade-off'u yansıtır (Decawave Ltd., 2017).

Pulse Repetition Frequency (PRF) parametresi iki farklı değer almaktadır: 16 MHz ve 64 MHz. 16 MHz PRF modu daha düşük güç tüketimi ve biraz daha iyi duyarlılık sunarken, 64 MHz PRF modu daha yüksek zaman çözünürlüğü sağlar. Ancak 64 MHz kullanımında duyarlılık yaklaşık 1-4 dB daha düşüktür (Decawave Ltd., 2017).

Preamble uzunluğu alıcı performansını önemli ölçüde etkileyen bir diğer parametredir. 2048 sembollük uzun preamble, -106 dBm gibi yüksek duyarlılık değerlerine ulaşmayı sağlarken, 256 sembollük kısa preamble kullanıldığında duyarlılık -93 dBm'e düşer. Preamble uzunluğu, sistemin senkronizasyon yeteneğini doğrudan etkiler (Decawave Ltd., 2017).

Tablo iki farklı paket hata oranı (PER) için değerler sunmaktadır: %1 PER ve %10 PER. %1 PER, yüksek güvenilirlik gerektiren uygulamalar için kullanılırken, %10 PER daha esnek sistemlerde tercih edilebilir. Örneğin, 110 kbps'de %1 PER için -106 dBm gerekirken, %10 PER'de bu değer -107 dBm'e düşer (Decawave Ltd., 2017).

Bu değerlerin tümü Kanal 5 (6489.6 MHz merkez frekans) için geçerlidir. Kanal 2 gibi diğer frekanslarda duyarlılık yaklaşık 1 dB daha düşük olabilir. Ölçümler, standart çerçeve formatında 20 byte'lık yüklerle, oda sıcaklığında ve stabil besleme voltajı altında yapılmıştır (Decawave Ltd., 2017).

Ekstrem duyarlılık değerlerine (örneğin -106 dBm) ulaşmak için "light" RX modu gibi özel yapılandırmalar gerekebilir. Bu mod, alıcının gürültü performansını optimize eder ancak bazı trade-off'lar getirebilir. Kullanıcı kılavuzunda bu özel ayarlar detaylı olarak açıklanmaktadır (Decawave Ltd., 2017).

Tablo 2. Verici (TX) AC Karakteristikleri (Decawave Ltd., 2017).

Parametre	**Minimum**	**Tipik**	**Maksimum**	**Birim**	**Açıklama/Koşullar**
Frekans Aralığı	3244	-	6999	MHz	Desteklenen tüm kanallar (Bkz. Tablo 11)
Kanal Bant Genişliği	-	500	-	MHz	Kanal 1,2,3,5 için (Kanal 4 ve 7: 900MHz)
Çıkış Güç Spektral Yoğunluğu	-41.3	-39	-35	dBm/MHz	Programlanabilir (FCC/ETSI sınırı: -41.3)
Yük Empedansı	-	100	-	Ω	Diferansiyel çıkış
Güç Ayar Aralığı	-	37	-	dB	Toplam kazanç ayar aralığı
Kaba Güç Adımı	-	3	-	dB	Kademeli ayar
İnce Güç Adımı	-	0.5	-	dB	Hassas ayar
Sıcaklıkla Güç Değişimi	-	0.05	-	dB/°C	Tüm kanallarda
Gerilimle Güç Değişimi	-	2.73	-	dB/V	Kanal 2 için (Kanal 5: farklı olabilir)

DW1000 veri sayfasındaki **Tablo 2**, çipin verici (TX) özelliklerini detaylı şekilde açıklayan önemli bir referanstır. Bu tablo, UWB iletişiminde kritik rol oynayan verici parametrelerinin teknik sınırlarını ortaya koymaktadır (Decawave Ltd., 2017).

Frekans aralığı bölümünde DW1000'nin 3244 MHz ile 6999 MHz arasında çalışabildiği belirtilmektedir. Bu geniş aralık, cihazın farklı UWB kanallarında (Kanal 1'den Kanal 7'ye kadar) esnek şekilde kullanılabilmesine olanak tanır. Kanal bant genişliği parametresi, çipin tipik olarak 500 MHz bant genişliğini desteklediğini göstermektedir. Ancak Kanal 4 ve 7 gibi özel durumlarda bu genişliğin 900 MHz'e kadar çıkabildiği not düşülmüştür (Decawave Ltd., 2017).

Çıkış gücü spektral yoğunluğu, vericinin en önemli özelliklerinden biridir. Minimum -41.3 dBm/MHz değeri, FCC ve ETSI düzenlemelerine uyumluluğu gösterirken, tipik -39 dBm/MHz ve maksimum -35 dBm/MHz değerleri, sistem tasarımcılarına esneklik sağlamaktadır. Bu güç değerleri, çip çıkışında ölçülen değerler olup, anten kazancı eklenerek efektif yayılım gücü (EIRP) hesaplanmalıdır (Decawave Ltd., 2017).

Yük empedansı 100 ohm diferansiyel olarak belirtilmiştir. Bu değer, RF tasarımında empedans uyumunun sağlanması için kritik bir parametredir. Güç ayar aralığı 37 dB olarak verilmiştir ve bu geniş aralık, farklı uygulama ihtiyaçlarına göre hassas güç kontrolü yapılabilmesine imkan tanır (Decawave Ltd., 2017).

Güç ayar adımları iki kademelidir. Kaba ayarlar için 3 dB'lik, ince ayarlar için ise 0.5 dB'lik adımlar kullanılabilmektedir. Bu özellik, özellikle regülasyonlara tam uyum sağlamak isteyen tasarımcılar için büyük kolaylık sağlar (Decawave Ltd., 2017).

Sıcaklıkla güç değişimi parametresi, çipin farklı çalışma sıcaklıklarında gösterdiği performansı ortaya koyar. Tipik 0.05 dB/°C değeri, geniş sıcaklık aralıklarında bile kararlı performans sergilendiğini göstermektedir. Gerilimle güç değişimi ise özellikle Kanal 2 için 2.73 dB/V olarak belirtilmiştir, bu değerin diğer kanallarda farklılık gösterebileceği not edilmiştir (Decawave Ltd., 2017).

Tablo 2, kullanıcıların farklı uygulama senaryolarına uygun güç ayarlamaları yapabilmesi için gerekli tüm teknik bilgileri sunmaktadır. Özellikle düzenleyici gerekliliklere uyum konusunda -41.3 dBm/MHz sınırının dikkate alınması gerektiği

vurgulanmaktadır. Anten kazancı ve sistem kayıpları gibi faktörlerin, bu değerler üzerinde etkili olacağı unutulmamalıdır (Decawave Ltd., 2017).

3.7 Malzemeler

Bu tez çalışmasında, konumlandırma sisteminin tasarlanmasında kullanılan malzemeler:

- ESP32 UWB Pro Long Distance Wrover

- K552 Safe 3A Örgülü Metal Başlı Type-C Şarj Kablosu

- Preo PS2 20000mAh Powerbank Siyah

- Preo PS2 10000mAh Powerbank

- YD-32 4 Port 3.0 USB Çoğaltıcı Hub

- ZyXEL VMG1312-T20B Antenli Vdsl2/Adsl2+4 Port Kablosuz Modem

3.7.1 Esp32 uwb pro long distance wrover

ESP32 UWB Pro Long Distance Wrover, standart modele kıyasla daha yüksek güç sağlayarak 200 m'ye kadar mesafe sunar. Bu özellik, dış mekân ve büyük depo gibi geniş alanlarda konumlandırma uygulamalarına olanak tanır (Robodükkan.com, 2018).

Özellikler:

- Decawave DWM1000 ile daha hassas izleme

- ESP32 ile hızlı ve güçlü uygulamalar

- Wifi ve Bluetooth'u destekler

- Arduino uyumlu

- Mikro USB konektörü var

- Kartın USB besleme gerilimi: 4,8~5,5V aralığında, 5,0V ise tipik gerilimidir (Robodükkan.com, 2018).



Şekil 13. ESP32 UWB Pro Long Distance Wrover (Robodükkan.com, 2018).

3.7.2 K552 safe 3a örgülü metal başlı type-c şarj kablosu

K552 Safe TYPE-C Örgü Metal Başlı 1000mm Şarj/Senkronizasyon Kablosu, tüm TYPE-C girişe sahip cihazlarla uyumludur. İp sargı olan ürünün kumaşsı bir yapısı vardır. K552, güvenli aktarım ve hızlı şarj işlemlerinin aynı anda yapılmasını sağlar. Ayrıca, kablonun siyah, gri, kırmızı, mavi ve gold renkleri bulunmaktadır (Linktech, 2024).



Şekil 14. K552 Safe 3A Örgülü Metal Başlı Type-C Şarj Kablosu (Linktech, 2024).

3.7.3 Preo ps2 20000mah powerbank

Lityum polimer pil teknolojisine ve 20000 mAh batarya gücüne sahip olan powerbank cihazın pilinin daha uzun süre kullanılmasını sağlar (Trendyol, 2024).

Preo PS2 powerbank üzerinde 4 adet gösterge ışığı vardır. Bu ışıklar, sarjın dolduğunu ve içindeki miktarını gösterir (Trendyol, 2024).

Powerbank, 2,1 A şarj akımına sahip olduğu için bluetooth kulaklık, kablosuz hoparlör, akıllı telefon, tablet vb. cihazları hızlı şarj eder (Trendyol, 2024).

Yüksek kaliteli materyallerden üretilen powerbank modeli, dış darbelere, neme ve diğer tüm dış etkenlere karşı dayanıklıdır (Trendyol, 2024).

Powerbank üzerinde 1 adet USB Micro B girişi, 1 adet de Type-C girişi bulunur. Bu girişler sayesinde farklı cihaz türleri şarj edilebilir (Trendyol, 2024).



Şekil 15. Preo PS2 20000mAh Powerbank (Trendyol, 2024).

3.7.4 Preo ps2 10000mah powerbank

Taşınabilir şarj cihazı, elektrik şebekesinden ayrı bir şekilde taşınabilen ve hareket halindeyken bile cihazların şarj edebilmesini sağlayan bir bataryadır (Teknosa, 2024).

Preo My Power Pocket Size 10000 mAh powerbank; yüksek aktarım verimi, yüksek güvenlik ve stabilite özellikli polimer hücreli batarya ile aşırı şarj olmasını/boşalmasını,

kısa devreyi engelleyen akıllı koruma programı ve kompakt tasarımı ile kolay kullanım sağlar (Teknosa, 2024).

- Batarya Türü : Lİ-Polimer batarya

- Batarya Kapasitesi : 10000 mAh

- Giriş Voltajı/akım : Micro USB/Type C DC 5V/2.1A

- Çıkış Voltajı/akım : USB 1 : DC 5V/2.4A, USB 2 : DC 5V/2.4 A

- Toplam Çıkış : 5V/2.4A

- Ölçüler : 93*63*24 mm

- Ağırlık : 186 gr (Teknosa, 2024).



Şekil 16. Preo PS2 10000mAh Powerbank (Teknosa, 2024).

3.7.5 Yd-32 4 port 3.0 usb çoğaltıcı hub

YD-32 USB Çoğaltıcı ile birden fazla cihaz USB'ye kolayca bağlanır. Bu özellik sayesinde, bilgisayara aynı anda USB fare, klavye, yazıcı, harici disk ve diğer cihazlar bağlanır (TeknolojiKurdu.com, 2023).

Küçük boyutları sayesinde kolayca taşınır. Bu sayede, istenilen anda ve mekânda ek USB bağlantı noktalarına sahip olunur. Tak-çalıştır özelliği ile kolayca kurulur. Windows, Mac ve Linux gibi işletim sistemleriyle uyumludur (TeknolojiKurdu.com, 2023).



Şekil 17. YD-32 4 Port 3.0 USB Çoğaltıcı Hub (TeknolojiKurdu.com, 2023).

YD-32 4 Port 3.0 USB Çoğaltıcı Hub Teknik Özellikleri:

- Bağlantısı USB 3.0'dır

- 4 adet USB Bağlantı Noktaları var

- Siyah renktedir

- 9.5 x 3.5 x 1.5 cm boyutlarındadır (TeknolojiKurdu.com, 2023).

3.7.6 Zyxel vmg1312-t20b antenli vdsl2/adsl2 kablosuz modem

Test işlemleri sırasında kullanıcı adı ve şifre bilgileri girilerek bağlantının devamlılığı sağlanmakta ve testler bu şekilde gerçekleştirilmektedir. Ayarlar değiştirildikten sonra fabrika ayarlarına dönülmediği durumlar yaşanabilmektedir. Bu durumda, cihaz fabrika ayarlarına döndürülmek için resetleme işlemi uygulanabilir (PttAVM.com, 2024).

Resetleme işlemi için, modemin arkasındaki reset tuşuna 15-20 saniye boyunca basılı tutulmakta ve ışıkların sönüp yeniden yanması beklenmektedir. Farklı bir servis sağlayıcı kullanılacaksa, ilgili servis sağlayıcıdan kullanıcı adı ve şifre temin edilmelidir. Modemin Wi-Fi adı ve şifresi, cihazın üzerinde yer alan etikette belirtilmiştir. Kurulum tamamlandıktan sonra Wi-Fi şifresinin değiştirilmesi mümkündür (PttAVM.com, 2024).

VDSL modem yapılandırması için modem arayüzüne erişim gerekmektedir. Bunun için modeme kablosuz bağlantı sağlanabilir ya da arkasındaki Ethernet girişlerinden biri üzerinden bağlantı yapılabilir. Bir web tarayıcısına '192.168.1.1' adresi girilerek erişim

sağlanır. Modem kullanıcı adı ve şifresi, cihazın üzerinde bulunan etikette yer almaktadır (PttAVM.com, 2024).

Kolay kurulum adımında, servis sağlayıcı tarafından verilen kullanıcı adı ve şifre bilgileri girilerek kurulum işlemi tamamlanır. Teknik destek hizmeti telefonda sağlanmamaktadır; destek alınması gereken durumlarda bir bilgisayar teknik servisine başvurulması gerekmektedir (PttAVM.com, 2024).



Şekil 18. ZyXEL VMG1312-T20B Modem (PttAVM.com, 2024).

- Yüksek hızlı Vdsl2/Adsl2+ internet erişimi

- 802.11n teknolojisi ile 300mbps'e varan kablosuz bağlantı hızı

- Farklı lokasyonlar arasında güvenli bağlantı kurabilmeye olanak sağlayan ıpsec vpn özelliği

- Tr-069 uzak yönetim desteği

- Güçlü spı güvenlik duvarı koruması

- Ağ veri depolama ünitesi bağlantısı olanağı

- 4*10/100mbps lan portu ve mbps wan portu

- 300 mbps kablosuz n usb vdsl / adsl modem router

- Vdsl / Adsl / Fiber / Kablosuz hizmetleri ile uyumlu wan bağlantı noktası özelliği Vdsl2 üzerinden 50-100 mbps'ye ve adsl ile geriye dönük uyumlu geniş bant (PttAVM.com, 2024).

3.8 Araştırma Modeli

Bu çalışma deneysel bir araştırma modeli kapsamında yürütülmüştür. ESP32 UWB Pro cihazları kullanılarak kapalı devre bir ağ oluşturulmuş ve Arduino IDE ortamında UDP tabanlı haberleşme altyapısı geliştirilmiştir. Tasarlanan sistem, gerçek zamanlı konum takibi yaparak elde edilen verilerin görselleştirilmesini sağlamaktadır (Nam Ho, Tran, Su ve Nguyen, 2023).

Çalışmada deneysel model benimsenmiştir. Bu model, ESP32 UWB Pro cihazlarının kapalı ağ ortamında haberleşme performansını değerlendirmek ve toplanan verilerin etkin şekilde görselleştirilmesini sağlamak amacıyla seçilmiştir.

Deneysel model, kapalı mekanlarda gerçek zamanlı konum takibi ve veri görselleştirme sisteminin etkinliğini test etmeye imkan tanıyan yapısı nedeniyle tercih edilmiştir. Bu yaklaşım, cihazların performansını değerlendirmek ve sistemin doğruluğunu ölçmek için en uygun yöntem olarak belirlenmiştir.

Araştırma süreci, donanım altyapısının kurulmasıyla başlamış; 3 adet ESP32 UWB Pro alıcı, 1 adet ESP32 UWB Pro verici ve bir modem kullanılarak kapalı devre ağ oluşturulmuştur. Ardından, Arduino IDE üzerinde UDP tabanlı haberleşme protokolü geliştirilmiştir. Verici cihazdan alınan konum verileri UDP protokolü ile alıcı cihazlara iletilmiş, toplanan veriler Python programlama dili ve Tkinter kütüphanesi kullanılarak sanal bir harita üzerinde görselleştirilmiştir.

Çalışmada kullanılan teknolojiler arasında donanım tarafında ESP32 UWB Pro alıcı ve verici cihazlar ile kapalı devre ağ oluşturmak amacıyla bir modem yer almıştır. Yazılım tarafında ise Arduino IDE ile UDP protokolü üzerinden haberleşme sağlanmış, Python ve Tkinter kütüphaneleriyle kullanıcı dostu bir arayüz oluşturularak konum verileri görselleştirilmiştir.

Toplanan veriler, ESP32 UWB Pro verici cihazdan elde edilen konum bilgilerini içermektedir. Bu bilgiler kapalı devre ağ üzerinden UDP protokolü aracılığıyla alıcı cihazlara aktarılmış ve Python tabanlı arayüzde işlenmiştir.

Araştırmanın temel amacı, ESP32 UWB Pro cihazlarının kapalı mekanlarda düşük maliyetli ve taşınabilir bir sistem kullanarak hassas konum takibi yapabilme yeteneğini göstermek ve bu verilerin gerçek zamanlı olarak görselleştirilebildiğini kanıtlamaktır.

Araştırma modeli birçok avantaj sunmaktadır. Bunlar arasında verilerin düşük gecikmeli olarak gerçek zamanlı takip edilebilmesi, kullanılan ESP32 UWB Pro cihazlarının yüksek hassasiyetli diğer çözümlere göre maliyet avantajı sağlaması ve sistemin kapalı mekanlarda kolayca uygulanabilir ve taşınabilir yapıda olması yer almaktadır.

Model kapsamında test edilen hipotez, ESP32 UWB Pro cihazlarının UDP protokolü aracılığıyla kapalı mekanlarda hassas, düşük maliyetli ve gerçek zamanlı bir konum takip sistemi oluşturabileceği yönündedir.

Araştırmada bazı sınırlılıklar bulunmaktadır. Sistem yalnızca 3 alıcı ve 1 verici cihazla test edilmiş, kullanılan modem ve UDP altyapısı kapalı devre ağ ile sınırlı kalmıştır. Ayrıca, görselleştirme arayüzü yalnızca Python ve Tkinter platformlarında çalışmakta olup, farklı platformlarda uyarlanabilirliği araştırılmamıştır.

Elde edilen sonuçlar, ESP32 UWB Pro cihazlarının kapalı mekanlarda hassas konum takibi sağlama kapasitesine sahip olduğunu ve bu verilerin kullanıcı dostu bir arayüzde dinamik olarak görselleştirilebildiğini göstermiştir. Sistem doğruluğu ve verimliliği, veri aktarımı ve görselleştirme süreçlerinde deneysel olarak ölçülmüştür (Özdemir vd., 2014)

Araştırmanın evreni, kapalı mekanlarda hassas konum takibi ihtiyacı olan tüm uygulama alanlarını kapsamaktadır. Bu kapsamda, ESP32 UWB Pro cihazlarının yaklaşık 200 metrelik menzil dahilinde kapalı mekanlarda konum belirleme performansları değerlendirilmiştir. Evren; ofisler, hastaneler, alışveriş merkezleri, okullar gibi çeşitli alanları içermekte ve özellikle gerçek zamanlı konum takibinin kritik olduğu senaryolar hedeflenmiştir.

Örneklem olarak yaklaşık 200 metrelik kapalı mekan çalışma alanında; 3 adet ESP32 UWB Pro alıcı cihaz, 1 adet ESP32 UWB Pro verici cihaz ve kapalı devre ağ oluşturmak için bir modem seçilmiştir. Bu örneklem, cihazların hassas konum takibi yapma kapasitesini test etmek amacıyla belirlenmiştir. Ancak elde edilen sonuçlar yalnızca bu örneklem bağlamında genellenebilir; daha geniş kapsamlı çıkarımlar için farklı ortam ve koşullarda ek deneylerin yapılması gerekmektedir.

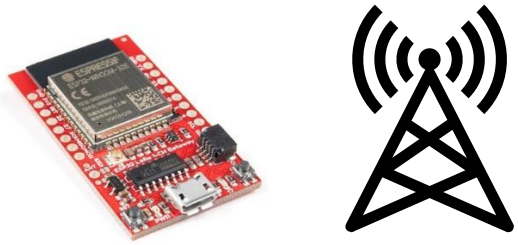
Örneklem seçimi kontrollü bir çalışma grubu yaklaşımıyla gerçekleştirilmiştir. Bu tercih, ESP32 UWB Pro cihazlarının taşınabilirliği, düşük maliyeti ve 200 metrelik menzil içindeki konum belirleme kapasitesine dayanmıştır. Ayrıca, modem ile kurulan kapalı devre ağ, sinyal karışmalarını engellemek ve verilerin güvenilir iletimini sağlamak amacıyla kullanılmıştır.

Çalışma, yaklaşık 200 metrelik kapalı mekan ortamında yürütülmüştür. Veri toplama sürecinde, sinyal yansımaları, elektromanyetik parazitler ve kapalı mekan yapısından kaynaklanan fiziksel etkiler hata payını etkileyebilecek unsurlar olarak değerlendirilmiştir. Bu hata kaynakları, veri işleme ve filtreleme teknikleriyle minimize edilmeye çalışılmıştır. UDP protokolünün düşük gecikmeli ve güvenilir veri aktarımı sağlaması, ölçümlerin doğruluğunu desteklemiştir. Ancak, cihazların teknik kısıtlamaları ve çevresel faktörlerden kaynaklanan küçük sapmalar analiz sürecinde göz önünde bulundurulmuştur.

3.9 Veriler ve Toplanması

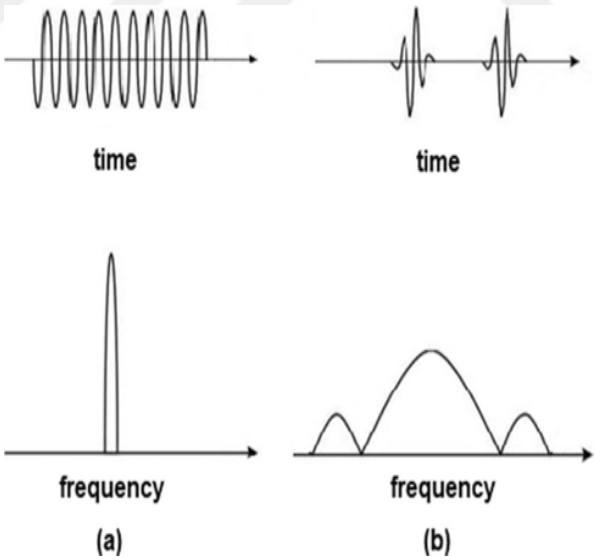
Bu tez çalışmasında üç adet ESP32 UWB Pro cihazı sabit alıcı istasyon (anchor) ve bir adet ESP32 UWB Pro cihazı hareketli verici (tag) olarak konumlandırılmıştır. Sabit alıcı istasyonların fiziksel yerleşimi **Şekil 19**'da gösterilmektedir.

3 ESP 32 UWB Pro = 3 Alıcı İstasyon



Şekil 19. Baz İstasyonlarının Tasarımı.

ESP32 UWB Pro cihazlarının sinyal karakteristikleri incelendiğinde; (a) sürekli dalga sinyali ile geleneksel darbe tabanlı UWB sinyallerine kıyasla daha kararlı bir yapı sunarken, (b) zaman ve frekans alanında 3.5-6.5 GHz aralığında çalışan UWB sinyallerinin dağılımı **Şekil 20**'de detaylı olarak görselleştirilmiştir (Ziegler, Kianfar, Hartmann ve Clausen, 2023). Bu sinyal yapısı, sistemin yüksek doğrulukta konum belirleme yapabilmesine olanak tanımaktadır.



Şekil 20. (a) Sürekli Dalga Sinyali, **(b)** Zaman ve Frekans Alanlarında UWB Sinyali (Ziegler vd., 2023).

Her bir ESP32 UWB Pro cihazı, 200 m'ye kadar menzil sağlayabilmektedir. Bu projede sistemin testleri, üç adet sabit alıcı ve bir adet hareketli verici kullanılarak kapalı bir tenis kortunda gerçekleştirilmiştir.

Sistem, bir modem aracılığıyla kapalı devre ağ içerisinde çalışacak şekilde tasarlanmıştır. Modem bilgisayara bağlanmış olup, hareketli vericiden alınan veriler bu ağ üzerinden bilgisayara iletilmiştir. Hareketli verici için 1234 numaralı bir UDP portu açılmış ve bu port üzerinden veri aktarımı sağlanmıştır.

Konum verilerinin görselleştirilmesi için Python programlama dili kullanılarak bir arayüz geliştirilmiştir. Arayüz, CustomTkinter kütüphanesi ile tasarlanmış ve UDP portu üzerinden gelen verileri işleyerek hareketli vericinin konumunu gerçek zamanlı olarak ekranda göstermektedir.

Geliştirilen arayüzde, hareketli vericinin mesafesi yakın olduğu istasyonun yanında görüntülenmekte ve vericiyi temsil eden simge, alınan mesafe verilerine göre ekranda hareket ettirilmektedir. Ayrıca, harita boyutu ve sabit alıcı istasyonlar arasındaki mesafeler, arayüzdeki ayarlar bölümünden manuel olarak değiştirilebilmektedir.

Bu araştırmada kullanılan veriler, ESP32 UWB Pro cihazlarından elde edilen konum bilgilerini içermektedir. Tüm veriler, kapalı devre ağ içerisinde UDP haberleşme protokolü kullanılarak toplanmış ve geliştirilen arayüz üzerinden işlenerek görselleştirilmiştir.

Bu araştırmada kullanılan veriler, ESP32 UWB Pro cihazlarından elde edilen konum bilgilerini içermektedir. Toplanan veriler, araştırmanın amaçlarına uygun olarak tasarlanmış bir kapalı devre ağ içerisinde, UDP protokolü kullanılarak aktarılmıştır. Verilerin toplanması süreci aşağıdaki başlıklar altında detaylandırılmıştır:

3.9.1 Sabit alıcı istasyonların arduino kodları

Üç ESP32 UWB Pro sabit alıcı istasyon için Arduino IDE programında bazı kodlar yazılmıştır. Bu kodlar:

- Sabit Alıcı İstasyonların MAC ID Adreslerinin Belirlenmesi

- Sabit Alıcı İstasyonların Bağlantı Pinlerinin Ayarlanması

3.9.1.1 Sabit alıcı istasyonların mac ıd adreslerinin belirlenmesi

Bu çalışmada, kapalı mekânda hareketli vericinin konumunun belirlenmesi için Arduino IDE programında hareketli vericin her sabit alıcı istasyonla doğru haberleşebilmesi için her sabit alıcı istasyona farklı MAC ID adresleri verilmiştir.

```
/*
ESP32 UWB - Sabit İstasyon (IS1)
*/
#include <SPI.h>
#include "DW1000Ranging.h"
#define ANCHOR_ADD "7D:00:22:EA:82:60:3B:9C" // IS1 MAC ID
```

Kod içerisinde #define ANCHOR_ADD "7D:00:22:EA:82:60:3B:9C" satırı ile 1. sabit alıcı istasyona özgü MAC ID tanımlanmış olup, bu MAC ID üzerinden hareketli vericiye ait mesafe verilerinin alınması sağlanmaktadır.

```
/*
ESP32 UWB - Sabit İstasyon (IS2)
*/
#include <SPI.h>
#include "DW1000Ranging.h"
#define ANCHOR_ADD "7D:00:22:EA:82:60:3B:9A" // IS2 MAC ID
```

#define ANCHOR_ADD "7D:00:22:EA:82:60:3B:9A" ifadesiyle, 2. sabit alıcı istasyon için özel olarak tanımlanan MAC ID belirtilmiştir. Bu adres aracılığıyla, hareketli vericiden gelen mesafe bilgileri bu istasyona yönlendirilmekte ve işlenmektedir.

```
/*
```

ESP32 UWB - Sabit İstasyon (IS3)

*/

#include <SPI.h>

#include "DW1000Ranging.h"

#define ANCHOR_ADD "7D:00:22:EA:82:60:3B:9B" // IS3 MAC ID

#define ANCHOR_ADD "7D:00:22:EA:82:60:3B:9B" komutu, 3. sabit alıcı istasyona ait özgün MAC ID'nin sisteme tanımlandığını göstermektedir. Bu tanımlama sayesinde, hareketli vericiden gönderilen mesafe verileri doğrudan bu istasyon aracılığıyla alınabilmektedir.

3.9.1.2 Sabit alıcı istasyonların bağlantı pinlerinin ayarlanması

Bu çalışmada, üç sabit alıcı istasyondan verilerin alınabilmesi için bazı bağlantı pinleri Arduino IDE programında oluşturulmuştur. Sabit alıcı istasyon olarak 3 adet ESP32 UWB Pro cihazı kullanılmıştır. Bu yüzden, her üç sabit alıcı istasyon için oluşturulan bağlantı pinleri aynıdır. Bu bağlantı pinleri:

- PIN_RST = 27: Reset pinidir. Çıkış (Output) veya Open-Drain olarak ayarlanabilir. Harici bir bileşeni (örneğin bir sensör, haberleşme modülü veya entegre) sıfırlamak (resetlemek) amacıyla kullanılır.

- PIN_IRQ = 34: Interrupt pinidir. Giriş (Input) pini olup, genellikle kesme (interrupt) desteği olan bir pindir.

- PIN_SS = 4: SPI Slave Select pinidir. Çıkış (Output) pinidir ve SPI (Serial Peripheral Interface) haberleşmesinde slave cihazı seçmek (enable/disable) için kullanılmaktadır.

const uint8_t PIN_RST = 27;

const uint8_t PIN_IRQ = 34;

const uint8_t PIN_SS = 4;

```
float OLÇÜM = 0;

bool aktif = false;
```

Yukarıdaki bölümde, ESP32 kartı ile DW1000 modülü arasındaki bağlantı pinleri tanımlanmıştır. PIN_RST (reset pini) 27 numaralı pine, PIN_IRQ (kesme pini) 34 numaralı pine ve PIN_SS (slave select pini) ise 4 numaralı pine atanmıştır. Bu pinler, DW1000 modülünün düzgün çalışabilmesi için gerekli olan donanımsal bağlantıları temsil eder. Ayrıca, OLÇÜM adlı float türünde bir değişken mesafe ölçüm değerlerini tutmak amacıyla tanımlanmış; aktif isimli bool türündeki değişken ise istasyonun aktif olup olmadığını kontrol etmek için kullanılmıştır.

3.9.2 Hareketli vericinin arduino kodları

Bir ESP32 UWB Pro hareketli için Arduino IDE programında bazı kodlar yazılmıştır.

Bu kodlar:

- Hareketli Vericinin MAC ID adresinin belirlenmesi

- Statik IP ile Kapalı Devre Ağının Oluşturulması

- UDP Haberleşme Ayarları

- Hareketli Vericinin Wi-Fi Bağlantı Yönetimi

- Hareketli Vericinin SPI ve Bağlantı Pinlerinin Ayarlanması

- Round Robin Algoritmasının Yazılması

- Mesafe ve Sinyal Gücü Verilerinin Güncellenmesi

- Konum Bildiriminde Düzenli UDP Yayını

- UDP Bildirimlerinin Dinlenmesi ve Konum Modu Kontrolü

- Yeni Cihaz Bağlantı ve Kesilme Geri Çağrıları

- Hareketli Vericiden Mesafe ve Seri Çıktıların Alınması

3.9.2.1 Hareketli vericinin mac ıd adresinin belirlenmesi

Üç ESP32 UWB Pro sabit alıcı istasyondaki gibi, kapalı mekânda üç sabit alıcı istasyondan alınan mesafe verilerine dayanarak hareketli vericinin konumunun belirlenmesi için Arduino IDE programında 1 ESP 32 UWB Pro hareketli vericiye bir MAC ID adresi verilmiştir.

```
#include <SPI.h>

#include "DW1000Ranging.h"

#include <WiFi.h>

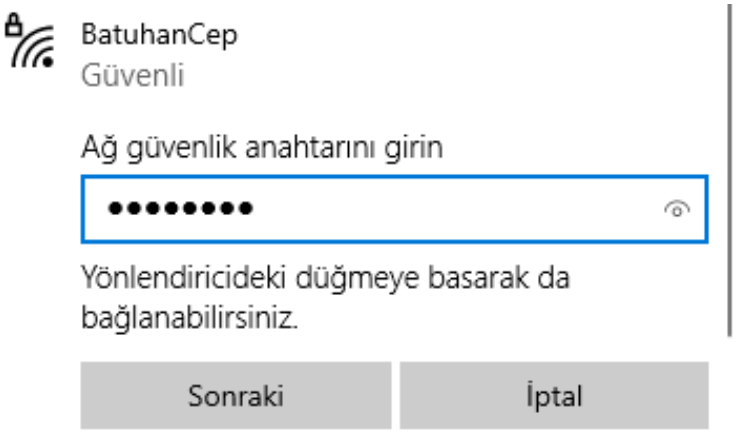
#include <WiFiUdp.h>

#define TAG_ADD "86:17:5B:D5:A9:9A:E2:9C" // Verici MAC ID
```

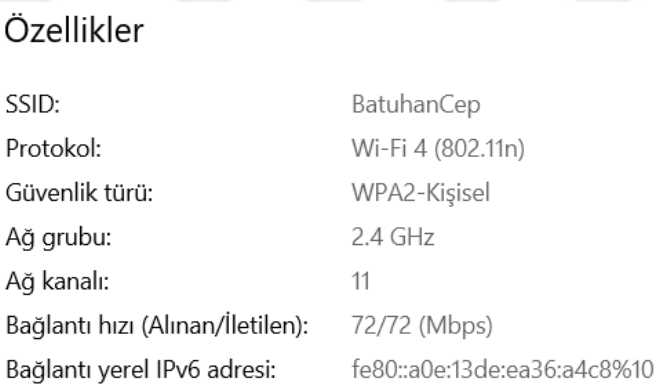
Bu satır, hareketli vericinin benzersiz MAC adresini tanımlar. DW1000 UWB modülü bu MAC ID üzerinden tanınır ve iletişim kurar. Bu MAC ID adresine her sabit alıcı istasyon arasındaki mesafe verileri gelecektir. Bu sayede, hareketli vericinin konumu belirlenebilecektir.

3.9.2.2 Statik ıp ile kapalı devre ağının oluşturulması

Kablosuz Yerel ağ (WLAN) üzerinde IP ataması sabit (statik IP) ile hareketli vericinin 3 sabit alıcı istasyon ile haberleşebilmesi için kapalı devre ağı oluşturulmuştur. Bu kapalı devre ağının güvenliği için bir kullanıcı adı ve şifre oluşturulmuştur. Şifre güvenliğinin güçlü olması için 4 harf ve 4 sayı bulunan 8 haneli şifre yazılarak, kapalı devre ağının güvenliği yükseltilmiştir.



Şekil 21. Kapalı Devre Ağının Kullanıcı Adı ve Şifresi.



Şekil 22. Kapalı Devre Ağının Özellikleri.

```
// Ağ ayarları

const char* ssid = "BatuhanCep";

const char* password = "gozo4482";

IPAddress local_IP(192, 168, 1, 34);

IPAddress gateway(192, 168, 1, 1);

IPAddress subnet(255, 255, 255, 0);

IPAddress primaryDNS(8, 8, 8, 8);
```

```
IPAddress secondaryDNS(8, 8, 4, 4);

void setup() {
  Serial.begin(115200);
  delay(1000);
  // WiFi başlatma
  if (!WiFi.config(local_IP, gateway, subnet, primaryDNS, secondaryDNS)) {
    Serial.println("IP yapılandırma başarısız!");
  }
  WiFi.begin(ssid, password);
  Serial.print("WiFi'ye bağlanıyor...");
  unsigned long wifiBaslangic = millis();
  while (WiFi.status() != WL_CONNECTED && millis() - wifiBaslangic < 10000) {
    delay(500);
    Serial.print(".");
  }
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("\nWiFi bağlantısı başarısız!");
    ESP.restart();
  }
  Serial.println("\nWiFi bağlandı!");
  Serial.print("Sabit IP Adresi: ");
```

Serial.println(WiFi.localIP());

- WiFi SSID ve Şifre Tanımlaması:

const char* ssid ve const char* password değişkenlerinde, bağlanılacak WiFi ağının adı ve şifresi tanımlanır.

- Statik IP Adresi ve Ağ Ayarları:

local_IP, gateway, subnet, primaryDNS ve secondaryDNS değişkenleriyle cihazın statik IP adresi, ağ geçidi, alt ağ maskesi ve DNS sunucuları belirlenir.

Bu yapılandırma, cihazın ağa sabit ve önceden tanımlanmış bir IP ile katılmasını sağlar.

- IP Konfigürasyonu:

WiFi.config() fonksiyonu ile yukarıdaki statik IP ve ağ ayarları ESP32'ye atanır. Eğer bu işlem başarısız olursa, seri port üzerinden hata mesajı yazdırılır.

- WiFi Ağına Bağlanma:

WiFi.begin(ssid, password); fonksiyonu ile ESP32 belirtilen WiFi ağına bağlanmaya başlar.

- Bağlantı Durumunun Kontrolü:

while döngüsü ile cihazın WiFi ağına bağlanması maksimum 10 saniye beklenir. Her 500 milisaniyede bir nokta yazdırılarak bağlantı süreci seri porttan takip edilir.

- Bağlantı Başarısızsa Yeniden Başlatma:

Eğer 10 saniye sonunda bağlantı sağlanamazsa, hata mesajı gösterilir ve cihaz ESP.restart(); ile yeniden başlatılır. Bu, bağlantı sorunlarının otomatik olarak çözülmesini amaçlar.

- Başarılı Bağlantı ve IP Yazdırma:

Bağlantı sağlandığında, cihazın aldığı statik IP adresi seri port üzerinden kullanıcıya bildirilir.

3.9.2.3 Udp haberleşme ayarları

Bu projede, ESP32 UWB cihazı ile bilgisayar arasında çift yönlü iletişim sağlamak amacıyla UDP (User Datagram Protocol) kullanılmıştır. Mesafe verilerinin gönderilmesi ve bilgisayardan alınan komutların (örneğin, konum bildirimi isteği) dinlenmesi için tek bir UDP portu (1234) tanımlanmıştır. Bu sayede, aynı port üzerinden veri gönderme ve alma işlemleri basit bir şekilde yönetilmiştir.

Aşağıda, ilgili UDP yapılandırma kodu yer almaktadır:

```
// UDP ayarları

WiFiUDP UDP;

const int udpPort = 1234;

IPAddress broadcastIP(255, 255, 255, 255);

 // Tek port açılıyor

  if (UDP.begin(udpPort)) {

     Serial.print("UDP portu açıldı: ");

     Serial.println(udpPort);

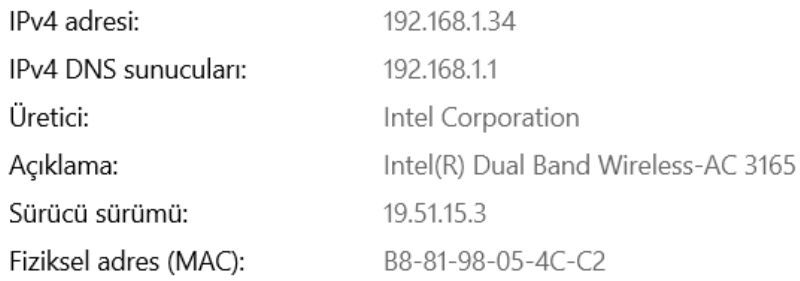
  } else {

     Serial.println("UDP portu açılamadı!");

  }
```

Bu tanımlamayla birlikte, UDP.beginPacket() ve UDP.write() fonksiyonları veri gönderiminde; UDP.parsePacket() ve UDP.read() fonksiyonları ise dışarıdan gelen komutların dinlenmesinde kullanılmaktadır.

Şekil 23'te hareketli vericiye ait UDP portların IP adresi 192.168.1.34 olarak gösterilmektedir.



Şekil 23. Hareketli Verici İçin Oluşturulan UDP Portunun IP Adresi.

3.9.2.4 Hareketli vericinin wi-fi bağlantı yönetimi

Bu bölümde, hareketli vericinin kapalı devre WiFi ağına bağlanma süreci gerçekleştirilmiştir. Cihaz, WiFi.begin(ssid, password); komutu ile tanımlı SSID ve şifre bilgileriyle WiFi bağlantısını başlatır. Bağlantının başarılı olup olmadığı 10 saniyelik bir süre boyunca kontrol edilir. Bu süre zarfında bağlantı kurulamazsa, sistem kendini yeniden başlatır. Bağlantı başarılı olursa, alınan IP adresi seri monitöre yazdırılarak kullanıcı bilgilendirilir.

```
WiFi.begin(ssid, password);
  Serial.print("WiFi'ye bağlanıyor...");
  unsigned long wifiBaslangic = millis();
  while (WiFi.status() != WL_CONNECTED && millis() - wifiBaslangic < 10000) {
    delay(500);
    Serial.print(".");
  }
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("\nWiFi bağlantısı başarısız!");
```

```
    ESP.restart();
  }
  Serial.println("\nWiFi bağlandı!");
  Serial.print("Sabit IP Adresi: ");
  Serial.println(WiFi.localIP());
```

3.9.2.5 Hareketli vericinin spı ve bağlantı pinlerinin ayarlanması

Bu çalışmada, hareketli vericinin üç sabit alıcı istasyondan mesafe verilerini alabilmesi için bazı bağlantı pinleri Arduino IDE programında oluşturulmuştur. Hareketli verici olarak 1 adet ESP32 UWB Pro cihazı kullanılmıştır. Sabit alıcı istasyonda ve hareketli vericide kullanılan cihazlar aynı olduğu için, hareketli verici için oluşturulan bağlantı pinleri de her üç sabit alıcı istasyon için oluşturulan bağlantı pinleriyle aynıdır. Bu bağlantı pinleri:

- PIN_RST = 27: Reset pinidir. Çıkış (Output) veya Open-Drain olarak ayarlanabilir. Harici bir bileşeni (örneğin bir sensör, haberleşme modülü veya entegre) sıfırlamak (resetlemek) amacıyla kullanılır.

- PIN_IRQ = 34: Interrupt pinidir. Giriş (Input) pini olup, genellikle kesme (interrupt) desteği olan bir pindir.

- PIN_SS = 4: SPI Slave Select pinidir. Çıkış (Output) pinidir ve SPI (Serial Peripheral Interface) haberleşmesinde slave cihazı seçmek (enable/disable) için kullanılmaktadır.

```
// SPI pin tanımları
#define SPI_SCK 18
#define SPI_MISO 19
#define SPI_MOSI 23
#define DW_CS 4
```

```
const uint8_t PIN_RST = 27;

const uint8_t PIN_IRQ = 34;

const uint8_t PIN_SS = 4;

// UWB başlatma

   SPI.begin(SPI_SCK, SPI_MISO, SPI_MOSI);

   DW1000Ranging.initCommunication(PIN_RST, PIN_SS, PIN_IRQ);
```

Sabit alıcı istasyonlar ile hareketli verici arasındaki seri haberleşme SPI protokolü üzerinden gerçekleştirilir. Kullanılan SPI pinleri şunlardır:

- SPI_SCK = 18: Saat (Clock) sinyalini taşıyan pin.
- SPI_MISO = 19: Master In Slave Out – veri, sabit alıcı istasyonlardan hareketli vericiye bu pin üzerinden iletilir.
- SPI_MOSI = 23: Master Out Slave In – veri, hareketli vericiden sabit alıcı istasyonlara bu pin üzerinden gönderilir.
- DW_CS = 4: Chip Select (Slave Select) pini olarak atanmıştır. SPI haberleşmesinde cihaz seçimi için kullanılır (PIN_SS ile aynı işlevi görür).

3.9.2.6 Round robin algoritmasının yazılması

Veri çakışmalarını (collision) engellemek ve sistemde düzenli bir ölçüm akışı sağlamak amacıyla, hareketli vericiye Round Robin algoritması Arduino IDE ortamında uygulanmıştır. Bu algoritma sayesinde, her ölçüm döngüsünde yalnızca bir sabit alıcı istasyon aktif hâle gelirken, diğer iki istasyon pasif konumda beklemektedir.

Süreç sırasıyla şu şekilde işlemektedir:

- İlk olarak, hareketli verici yalnızca birinci sabit alıcı istasyondan (IS1) mesafe verilerini alır; IS2 ve IS3 bekleme modundadır.

- Ardından, ikinci sabit alıcı istasyon (IS2) aktif hâle gelir; IS1 ve IS3 pasif kalır.
- Son olarak, üçüncü sabit alıcı istasyon (IS3) ile iletişim kurulur; IS1 ve IS2 beklemededir.

Bu döngü sürekli olarak tekrarlanmakta olup, her bir istasyona ölçüm için belirli bir zaman dilimi (örneğin 1000 ms) ayrılmıştır. Böylece her ölçüm adımı arasında zamanlama uyumu sağlanmakta ve veri çakışmaları önlenmektedir.

```
enum IstasyonDurumu { HAZIR, IS1_OLCUM, IS2_OLCUM, IS3_OLCUM };

IstasyonDurumu suankiDurum = IS1_OLCUM;

unsigned long durumBaslangicZamani = millis();

const long durumZamanAsimi = 1000; // ms

void loop() {

   DW1000Ranging.loop();

   if (millis() - durumBaslangicZamani > durumZamanAsimi) {

      sonrakiIstasyonaGec();

   }

}

void sonrakiIstasyonaGec() {

   switch (suankiDurum) {

      case IS1_OLCUM: suankiDurum = IS2_OLCUM; break;

      case IS2_OLCUM: suankiDurum = IS3_OLCUM; break;

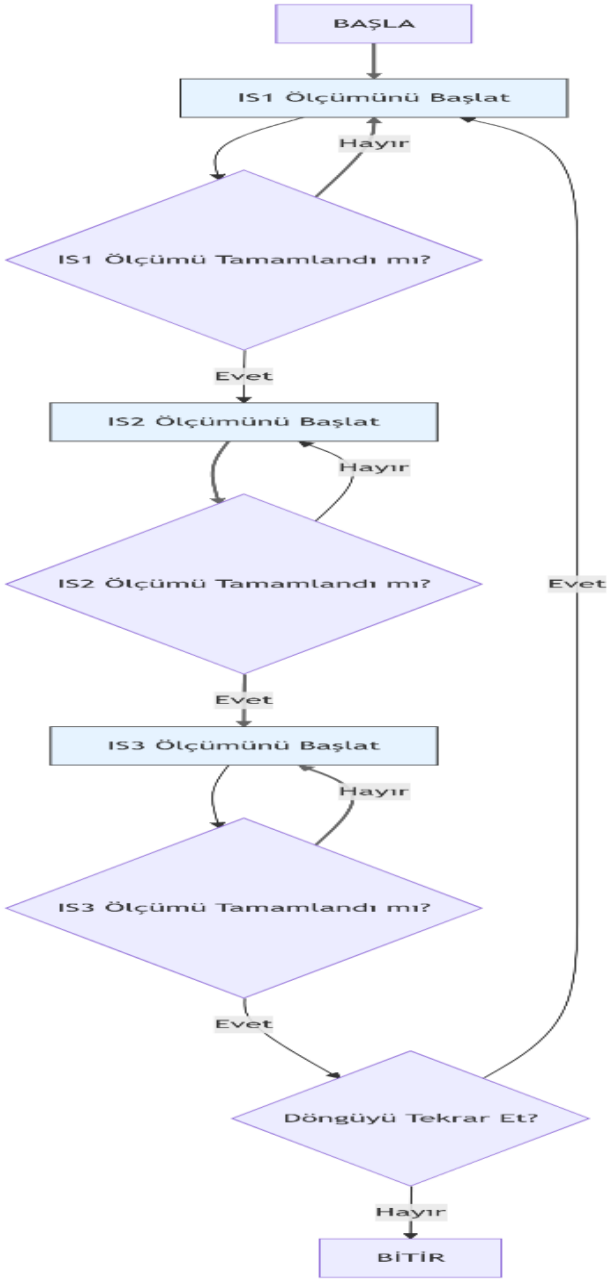
      case IS3_OLCUM: suankiDurum = IS1_OLCUM; break;

      default: suankiDurum = IS1_OLCUM; break;
```

```
    }

    durumBaslangicZamani = millis();

}
```



Şekil 24. Round Robin Algoritması Akış Şeması.

Algoritma, her sabit alıcı ile belirli bir sırayla iletişim kurarak mesafe ve sinyal gücü verilerini toplar. Bu sayede veri iletiminde çakışmalar önlenir ve sistemin stabil çalışması sağlanır.

3.9.2.7 Mesafe ve sinyal gücü verilerinin güncellenmesi

Hareketli Vericinin her sabit alıcı istasyondan mesafe verilerini doğru alabilmesi için Arduino IDE programında bazı düzenlemeler yapılmıştır. Bu düzenlemeler sayesinde hareketli vericiye sabit alıcı istasyonlardan mesafe verileri doğru bir şekilde gelmektedir.

```
// Ölçüm verileri
float mesafeVerileri[3] = {0.0, 0.0, 0.0};
float gucVerileri[3] = {0.0, 0.0, 0.0};
void yeniMesafe() {
  DW1000Device* uzakCihaz = DW1000Ranging.getDistantDevice();
  uint16_t suankiIstasyon = uzakCihaz->getShortAddress();
  switch (suankiDurum) {
    case IS1_OLCUM:
      if (suankiIstasyon == 0x3B9C) {
        mesafeVerileri[0] = uzakCihaz->getRange() - 1.0;
        gucVerileri[0] = uzakCihaz->getRXPower();
        istasyonBilgisiYazdir(1, mesafeVerileri[0], gucVerileri[0]);
      }
      break;
    case IS2_OLCUM:
```

```
            if (suankiIstasyon == 0x3B9A) {
                mesafeVerileri[1] = uzakCihaz->getRange() - 1.0;
                gucVerileri[1] = uzakCihaz->getRXPower();
                istasyonBilgisiYazdir(2, mesafeVerileri[1], gucVerileri[1]);
            }
            break;
        case IS3_OLCUM:
            if (suankiIstasyon == 0x3B9B) {
                mesafeVerileri[2] = uzakCihaz->getRange() - 1.0;
                gucVerileri[2] = uzakCihaz->getRXPower();
                istasyonBilgisiYazdir(3, mesafeVerileri[2], gucVerileri[2]);
            }
            break;
        case HAZIR:
            break;
    }
}
```

Bu kod parçası, üç sabit istasyondan (IS1, IS2, IS3) gelen mesafe ve sinyal gücü (RSSI) verilerinin güncellenmesi işlevini gerçekleştirir. Programda her istasyonun ölçüm aşaması sırasıyla kontrol edilir ve o sırada gelen veriler ilgili dizilere kaydedilir.

Kodda, önce mesafe ve sinyal gücü verilerini saklamak için iki adet float tipinde dizi tanımlanmıştır. Her dizide 3 eleman vardır ve bu elemanlar sırasıyla IS1, IS2 ve IS3 istasyonlarının verilerini tutar.

Fonksiyon yeniMesafe() çağrıldığında:

- En son ölçüm yapılan uzak cihaz (sabit istasyon) nesnesi alınır.

Bu nesne DW1000Ranging.getDistantDevice() ile elde edilir.

- Cihazın kısa adresi okunur.

Kısa adres, istasyonun tanımlayıcı numarasıdır ve her istasyonun benzersizdir.

- Programın o anki durumu (hangi istasyonun ölçüm aşamasında olduğu) kontrol edilir.

Bu durum suankiDurum değişkeni ile tutulur ve üç farklı durum vardır: IS1_OLCUM, IS2_OLCUM, IS3_OLCUM.

- Mevcut durum ve alınan cihaz adresi karşılaştırılır.

Eğer durum ve adres eşleşiyorsa, ilgili istasyondan gelen mesafe ve sinyal gücü değerleri dizilere kaydedilir.

- Mesafe verisi bir miktar kalibrasyon için 1.0 çıkarılarak güncellenir.

Bu sayede çevresel etkiler veya cihaz kaymaları dengelenir.

Sonuç olarak, bu fonksiyon çalıştığında şu anda ölçüm yapılan istasyondan gelen mesafe ve sinyal gücü bilgileri, uygun dizinin ilgili indeksine güncellenmiş olur. Böylece her üç sabit istasyondan alınan veriler ayrı ayrı takip edilir ve konum hesaplama veya analiz işlemleri için hazır tutulur.

3.9.2.8 Konum bildiriminde düzenli udp yayını

```
// Konum bildirimi

bool konumBildirimiAktif = false;
```

```
unsigned long sonBildirimZamani = 0;
const long bildirimAraligi = 10000; // 10 saniyede bir
void loop() {
  DW1000Ranging.loop();
  bildirimleriDinle();
  switch (suankiDurum) {
    case IS1_OLCUM:
    case IS2_OLCUM:
    case IS3_OLCUM:
      if (millis() - durumBaslangicZamani > durumZamanAsimi) {
        sonrakiIstasyonaGec();
      }
      break;
    case HAZIR:
      break;
  }
  // Konum bildirimi aktifse düzenli olarak gönder
  if (konumBildirimiAktif && millis() - sonBildirimZamani > bildirimAraligi) {
    verileriGonder(true);
    sonBildirimZamani = millis();
  }
```

```
}

void verileriGonder(bool bildirimModu) {
   String veri = "MAC:" + String(TAG_ADD) +
                 ",IS1:" + String(mesafeVerileri[0], 2) +
                 ",IS2:" + String(mesafeVerileri[1], 2) +
                 ",IS3:" + String(mesafeVerileri[2], 2);
   if (bildirimModu) {
      veri += ",NOTIFY:ACK";
   }
   UDP.beginPacket(broadcastIP, udpPort);
   UDP.write((const uint8_t*)veri.c_str(), veri.length());
   UDP.endPacket();
   Serial.println("Veriler gönderildi: " + veri);
}
```

konumBildirimiAktif değişkeni, konum bildiriminin açık veya kapalı olduğunu belirtir ve başlangıçta false (kapalı) olarak ayarlanmıştır. sonBildirimZamani değişkeni, en son veri gönderiminin zamanını milisaniye cinsinden tutar. bildirimAraligi ise iki veri gönderimi arasındaki minimum süreyi belirler ve burada 10 saniye (10000 ms) olarak tanımlanmıştır.

Arduino'nun ana döngüsü olan loop() fonksiyonu sürekli çalışır. Bu fonksiyon içinde, konum bildirimi aktifse ve son gönderimden bu yana belirlenen aralık kadar süre geçmişse, verileriGonder(true) fonksiyonu çağrılır. Böylece güncel mesafe verileri UDP

üzerinden gönderilir ve sonBildirimZamani güncellenerek sonraki gönderim için zamanlayıcı sıfırlanır.

verileriGonder() fonksiyonu, hareketli vericinin MAC adresi ile üç sabit istasyondan alınan mesafe verilerini içeren bir veri dizisi oluşturur. Eğer fonksiyon parametresi true ise, veriye "NOTIFY:ACK" etiketi eklenir. Bu veri UDP paketi olarak broadcast IP adresine ve önceden tanımlı UDP portuna gönderilir.

Bu yapı sayesinde, konum bildirimi aktif edildiğinde cihaz, belirlenen aralıklarla düzenli olarak güncel mesafe bilgilerini ağdaki diğer cihazlara veya izleme sistemine ileterek konum takibine olanak sağlar.

3.9.2.9 Udp bildirimlerinin dinlenmesi ve konum modu kontrolü

Her bir sabit alıcı istasyonundan veriler, sırasıyla mesafe ve güç cinsinden gelmiştir. Mesafe için, uzunluk birimi olarak metre (m); sinyal gücü için güç birimi olarak da desibel miliwatt (dBm) kullanılmıştır.

```
void bildirimleriDinle() {
  int packetSize = UDP.parsePacket();
  if (packetSize) {
    char packetBuffer[255];
    int len = UDP.read(packetBuffer, 255);
    if (len > 0) {
      packetBuffer[len] = 0;
    }
    String mesaj = String(packetBuffer);
    Serial.print("Alınan bildirim: ");
```

```
        Serial.println(mesaj);
        if (mesaj.indexOf("NOTIFY:ON") >= 0) {
            konumBildirimiAktif = true;
            Serial.println("Konum bildirimi AKTİF");
        }
        else if (mesaj.indexOf("NOTIFY:OFF") >= 0) {
            konumBildirimiAktif = false;
            Serial.println("Konum bildirimi PASİF");
        }
    }
}
```

Bu fonksiyon, UDP üzerinden gelen bildirim paketlerini dinler ve içeriğine göre konum bildirimi modunu açıp kapatır.

- UDP.parsePacket() ile gelen paketin boyutu kontrol edilir. Eğer paket varsa (packetSize > 0),
- Paketin içeriği packetBuffer dizisine okunur ve sonuna null karakter (\0) eklenerek string olarak işlenir.
- Gelen mesaj String mesaj olarak saklanır.
- Eğer mesaj içinde "NOTIFY:ON" ifadesi geçiyorsa, konumBildirimiAktif değişkeni true yapılır; bu, konum bildiriminin aktif edildiği anlamına gelir.
- Eğer mesaj içinde "NOTIFY:OFF" ifadesi varsa, konumBildirimiAktif false olur ve konum bildirimi kapatılır.

Bu şekilde cihaz, dışarıdan gelen UDP komutlarına göre konum bildirim modunu dinamik olarak açıp kapatabilir.

3.9.2.10 Yeni cihaz bağlantı ve kesilme geri çağrıları

```
DW1000Ranging.attachNewRange(yeniMesafe);
   DW1000Ranging.attachBlinkDevice(yeniCihaz);
   DW1000Ranging.attachInactiveDevice(pasifCihaz);
void yeniCihaz(DW1000Device *cihaz) {
   Serial.print("Yeni cihaz eklendi: ");
   Serial.println(cihaz->getShortAddress(), HEX);
}
void pasifCihaz(DW1000Device *cihaz) {
   Serial.print("Cihaz bağlantısı kesildi: ");
   Serial.println(cihaz->getShortAddress(), HEX);
}
```

Bu kod, DW1000 UWB modülünün bağlantı ve veri işlemleri için geri çağırma (callback) fonksiyonlarını tanımlar ve kullanır.

- DW1000Ranging.attachNewRange(yeniMesafe);

Yeni mesafe ölçümü yapıldığında yeniMesafe fonksiyonunu çağırır. Bu fonksiyon mesafe verilerini işler.

- DW1000Ranging.attachBlinkDevice(yeniCihaz);

Yeni bir cihaz (tag veya sabit istasyon) ağda göründüğünde yeniCihaz fonksiyonu tetiklenir.

- DW1000Ranging.attachInactiveDevice(pasifCihaz);

Daha önce bağlı olan ancak artık ağda aktif olmayan cihazlar için pasifCihaz fonksiyonu çağrılır.

- yeniCihaz(DW1000Device *cihaz) fonksiyonu:

Yeni bir cihaz bağlantısı algılandığında çağrılır. Cihazın kısa adresini (ID) seri port üzerinden hexadecimal formatta yazdırır.

- pasifCihaz(DW1000Device *cihaz) fonksiyonu:

Cihazın bağlantısı kesildiğinde çağrılır. Bağlantısı kesilen cihazın kısa adresini seri portta gösterir.

Bu sayede sistem, ağdaki cihazların bağlanma ve ayrılma durumlarını takip eder ve gerektiğinde işlem yapabilir.

3.9.2.11 Hareketli vericiden mesafe ve seri çıktıların alınması

Her bir sabit alıcı istasyonundan veriler, sırasıyla mesafe ve güç cinsinden gelmiştir. Mesafe için, uzunluk birimi olarak metre (m); sinyal gücü için güç birimi olarak da desibel miliwatt (dBm) kullanılmıştır.

Veriler gönderildi: MAC:86:17:5B:D5:A9:9A:E2:9C,IS1:2.06,IS2:2.19,IS3:4.71

IS1 - Mesafe: 2.22 m, Güç: -97.12 dBm

IS1 - Mesafe: 2.19 m, Güç: -97.37 dBm

IS1 - Mesafe: 2.20 m, Güç: -97.29 dBm

IS1 - Mesafe: 2.26 m, Güç: -97.36 dBm

IS1 - Mesafe: 2.18 m, Güç: -96.31 dBm

IS1 ölçümü tamamlandı, IS2'ye geçiliyor...

IS2 - Mesafe: 2.31 m, Güç: -87.12 dBm

IS2 - Mesafe: 2.36 m, Güç: -86.38 dBm

IS2 - Mesafe: 2.34 m, Güç: -85.58 dBm

IS2 - Mesafe: 2.34 m, Güç: -86.18 dBm

IS2 - Mesafe: 2.33 m, Güç: -87.62 dBm

IS2 ölçümü tamamlandı, IS3'e geçiliyor...

IS3 - Mesafe: 4.71 m, Güç: -98.11 dBm

IS3 - Mesafe: 4.64 m, Güç: -100.42 dBm

IS3 - Mesafe: 4.63 m, Güç: -100.45 dBm

IS3 - Mesafe: 4.68 m, Güç: -100.08 dBm

IS3 - Mesafe: 4.75 m, Güç: -100.52 dBm

IS3 ölçümü tamamlandı, IS1'e geçiliyor...

Çıktının ilk satırı, hareketli vericinin kendi MAC adresi ile üç sabit istasyondan (IS1, IS2, IS3) alınan mesafe verilerinin UDP protokolü üzerinden ağda gönderildiğini gösterir. Bu satır, sistemin ölçüm sonuçlarını özetleyen bir veri paketidir.

Her bir sabit istasyon için, ölçülen mesafe ve sinyal gücü değerleri ardışık satırlarda detaylı şekilde listelenmiştir. Mesafe değerleri metre (m) cinsindendir ve sinyal gücü değerleri desibel miliwatt (dBm) cinsindendir. Sinyal gücü negatif değerler olarak gösterilir; daha az negatif olan değerler daha güçlü sinyali ifade eder.

IS1 sabit istasyonundan alınan bir dizi ölçüm tamamlandıktan sonra, “IS1 ölçümü tamamlandı, IS2’ye geçiliyor...” mesajı ile sistem bir sonraki sabit istasyon olan IS2’ye geçiş yapar. Bu ifade, round robin algoritmasıyla sabit istasyonlar arasında sıralı ve döngüsel ölçüm yapıldığını belirtir.

IS2 ve IS3 istasyonlarından da benzer şekilde mesafe ve güç ölçümleri alınır, her ölçüm serisinin sonunda “ISx ölçümü tamamlandı, ISy’ye geçiliyor...” mesajı ile sıradaki istasyona geçiş bildirilir. Bu sayede ölçüm döngüsü sürekli devam eder.

Bu yapının amacı, hareketli vericinin çevresindeki sabit istasyonlardan düzenli ve güncel mesafe ile sinyal gücü verilerini almasını sağlamaktır. Bu veriler, hareketli cihazın konumlandırılması ve ortam koşullarının değerlendirilmesi için temel oluşturur.

Sonuç olarak, bu seri monitör çıktıları sistemin round robin algoritması ile doğru ve kesintisiz veri topladığını, ayrıca UDP üzerinden verileri ağda başarılı şekilde paylaştığını doğrular.

3.10 Verilerin Çözümlenmesi ve Yorumlanması

Bu bölümde, UDP portu üzerinden alınan ESP32 UWB cihazlarından gelen mesafe verilerinin Python tabanlı grafiksel kullanıcı arayüzünde işlenmesi, analiz edilmesi ve görselleştirilmesi süreci ayrıntılı olarak ele alınmıştır. Sistem, ağ üzerinden gelen verileri gerçek zamanlı olarak dinlemek üzere socket programlama tekniklerini kullanarak UDP protokolü üzerinden güvenilir ve hızlı veri iletimi sağlamaktadır. Alınan veriler içerisinde bulunan MAC adresleri sayesinde her bir cihaz benzersiz olarak tanımlanmış ve takip edilmek istenen hedef cihaz dinamik olarak seçilebilmiştir. Bu sayede, sistemde birden fazla cihazın aynı anda çalışması halinde bile istenilen cihaza ait veriler filtrelenerek doğru takip gerçekleştirilmiştir.

Sabit alıcı istasyonların fiziksel konumları, kullanıcı arayüzünde orantılı ve doğru koordinatlara dönüştürülerek görsel olarak temsil edilmiştir. Bu sayede kullanıcı, sabit noktalar arasındaki geometrik yapı ve ölçüm alanının sınırları hakkında görsel bilgiye sahip olmuştur. Veri çakışmalarını engellemek amacıyla Round Robin algoritması uygulanmış ve her ölçüm çevriminde tek bir alıcı istasyonun aktif olarak veri toplaması sağlanmıştır. Bu yöntem, sistemde oluşabilecek sinyal karışıklıklarını minimize ederek ölçüm doğruluğunu artırmıştır.

Alıcı istasyonlardan elde edilen mesafe verileri, trilaterasyon algoritması kullanılarak iki boyutlu düzlemde gerçek zamanlı (x, y) koordinatlarına dönüştürülmüştür. Trilaterasyon yöntemi, sabit noktalar arasındaki mesafe ölçümlerinden hareketli vericinin konumunu

hesaplama temeline dayanmaktadır ve bu çalışma kapsamında kullanılan formüllerle hassas konum belirleme sağlanmıştır. Elde edilen koordinatlar, arayüzdeki harita ölçeğine uygun şekilde yeniden ölçeklendirilmiş ve böylece ölçülen gerçek dünya mesafeleri ile görsel sunum arasında tutarlılık korunmuştur.

Kullanıcı arayüzü, dinamik olarak hareketli vericinin konumunu yeşil bir kutu ile işaretleyerek gerçek zamanlı takip imkânı sunmuştur. Bu görsel temsil, konum değişimlerinin anlık olarak izlenmesini mümkün kılmıştır. Ayrıca, sistemde veri akışı sürecinde zaman aşımı kontrolleri uygulanmış; belirli bir süre veri alınamadığında Round Robin algoritması aktif istasyonu değiştirerek sistemin sürekli çalışabilirliği garanti altına alınmıştır. Terminal ekranına ise sadeleştirilmiş ve anlamlı veriler yansıtılmış, gereksiz tekrarların önüne geçilmiştir.

Son olarak, kullanıcı dostu arayüz tasarımı sayesinde cihaz seçimi, sabit istasyon konumlarının ayarlanması ve hareketli vericiye konum bildirimi gönderme gibi işlemler kolayca gerçekleştirilmiştir. Bu özellikler, sistemin hem teknik hem de kullanım açısından esnek ve işlevsel olmasını sağlamıştır.

- UDP Üzerinden Veri Alımı ve Çözümleme

- MAC Adresi ile Cihaz Tanımlama ve Seçimi

- Sabit İstasyonların Grafiksel Konumlandırılması

- Round Robin Algoritması ile Veri Çakışmasının Önlenmesi

- Koordinat Sistemi ve Sabit İstasyonların Grafiksel Gösterimi

- Konum Belirleme Yöntemi

- Hareketli Vericinin Gerçek Zamanlı Takibi

- Mesafe ve Koordinatların Kullanıcıya Sunumu

- Hareketli Verici Konumunun Görüntülenmesi

- Çoklu İstasyon Veri Güncelleme ve Zaman Aşımı

- ESP32 UWB Kontrol Paneli: Cihaz Yönetimi ve Sistem Ayarları

3.10.1 Udp üzerinden veri alımı ve çözümleme

```
import customtkinter
import json
import os
import threading
import socket
import time
from PIL import Image, ImageTk
import pandas as pd
UDP_IP = "0.0.0.0"  # Tüm ağ arayüzlerini dinle
PORT = 1234  # Veri alışverişi için kullanılacak port
def fetch_udp(self):
    """UDP paketlerini dinler ve işler"""
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.bind((UDP_IP, PORT))
    sock.settimeout(0.1)
    if not hasattr(self, 'last_received_time'):
        self.last_received_time = {"IS1": 0, "IS2": 0, "IS3": 0}
    while True:
```

```
try:
    data, _ = sock.recvfrom(512)
    message = data.decode('utf-8').strip()
    if not message:
        continue
    print(f"\nAlınan ham veri: {message}")  # Türkçe çıktı
    # MAC adresi tespiti
    if "MAC:" in message:
        mac_start = message.find("MAC:") + 4
        mac_end = message.find(",", mac_start)
        mac = message[mac_start:mac_end] if mac_end != -1 else message[mac_start:]
```

UDP protokolü için IPv4 tipi bir soket (socket.AF_INET) oluşturularak, tüm ağ arayüzlerinden gelen veri trafiğini dinlemek üzere 0.0.0.0 IP adresi ve 1234 numaralı porta bağlanılır. Veri alma işlemi, recvfrom(512) fonksiyonu ile en fazla 512 byte uzunluğundaki UDP paketlerini almak suretiyle gerçekleştirilir. Gelen ham byte verisi UTF-8 formatında çözülerek string tipine dönüştürülür. Bu string içerisinde "MAC:" ifadesi aranarak, her bir cihaza ait benzersiz MAC adresi çıkarılır. Olası ağ hataları ve zaman aşımı durumları, try-except blokları kullanılarak kontrol altına alınır ve sistemin kesintisiz çalışması sağlanır.

3.10.2 Mac adresi ile cihaz tanımlama ve seçimi

```
# MAC adresi tespiti
if "MAC:" in message:
```

```
        mac_start = message.find("MAC:") + 4
        mac_end = message.find(",", mac_start)
        mac = message[mac_start:mac_end] if mac_end != -1 else message[mac_start:]
        self.available_devices[mac] = {
            'name': f"Cihaz-{len(self.available_devices)+1}",  # Türkçe önek
            'last_seen': time.time()
        }
        if not self.target_device_mac:
            self.target_device_mac = mac
            self.target_device_name = self.available_devices[mac]['name']
            self.device_label.configure(text=f"Takip Edilen: {self.target_device_name}")
# Türkçe etiket
        updated = False
```

Cihazların MAC adresleri, available_devices adlı sözlükte saklanır ve sistemde ilk algılanan cihaz varsayılan hedef olarak otomatik seçilir. Seçilen cihazın adı kullanıcı arayüzünün sol panelinde gösterilir ve cihazın son aktif olduğu zaman last_seen değişkeni ile kaydedilerek güncellenir.

3.10.3 Sabit istasyonların grafiksel konumlandırılması

```
class App(customtkinter.CTk):
    APP_NAME = "ESP32 UWB Takip Sistemi"  # Uygulama adı (Türkçe)
    WIDTH = 1000  # Pencere genişliği
    HEIGHT = 500  # Pencere yüksekliği
```

```
def __init__(self, *args, **kwargs):
    super().__init__(*args, **kwargs)
    # Temel pencere ayarları
    self.title(App.APP_NAME)
    self.geometry(f"{App.WIDTH}x{App.HEIGHT}")
    self.minsize(App.WIDTH, App.HEIGHT)
    self.protocol("WM_DELETE_WINDOW", self.on_closing)
    # Başlangıç değişkenleri
    self.target_device_mac = None  # Hedef cihaz MAC adresi
    self.target_device_name = "Cihaz-1"  # Varsayılan cihaz adı
    self.available_devices = {}  # Tespit edilen cihazlar
    self.background_image_path = "arkaplan.png"  # Arkaplan resmi yolu
    self.locations_file = "locations.json"  # Konum ayarları dosyası
```



Şekil 25. Sabit Alıcı İstasyonların Eşkenar Üçgen Geometrisi.

Bu kod parçası, customtkinter.CTk sınıfından türetilen App sınıfı aracılığıyla bir grafiksel kullanıcı arayüzü penceresi oluşturur ve temel özelliklerini yapılandırır. Pencere başlığı (APP_NAME), boyutları (1000x500 piksel) ve minimum boyut kısıtlamaları tanımlanırken, kullanıcı pencereyi kapattığında çalışacak on_closing metodu ayarlanır. Aynı zamanda, UWB cihaz takip sistemi için gerekli olan temel veri yapıları başlatılır: hedef cihazın MAC adresi (target_device_mac), varsayılan cihaz adı ("Cihaz-1"), keşfedilen cihazların saklanacağı sözlük (available_devices), arka plan görseli yolu (arkaplan.png) ve istasyon konum bilgilerinin tutulacağı JSON dosyası (locations.json). Bu yapılandırma, sistemin çekirdek bileşenlerinin çalışması için gerekli olan temel veri modelini ve arayüz özelliklerini hazır hale getirir.

3.10.4 Round robin algoritması ile veri çakışmasının önlenmesi

```
self.device_data = {"IS1": 0, "IS2": 0, "IS3": 0}  # Mesafe verileri

    self.is_data = {"IS1": None, "IS2": None, "IS3": None}  # İstasyon verileri

    self.locations_data = None  # Konum ayarları

    self.current_station_index = 0  # Round Robin için mevcut istasyon indeksi

    self.station_order = ["IS1", "IS2", "IS3"]  # İstasyon sırası

    self.last_update_time = time.time()  # Son güncelleme zamanı

    self.station_timeout = 1.0  # İstasyon zaman aşımı (saniye)
```

Bu kod bloğu, UWB tabanlı konum takip sisteminin temel veri modelini ve kontrol mekânizmalarını oluşturur. device_data sözlüğü, her bir istasyondan (IS1, IS2, IS3) alınan güncel mesafe bilgilerini santimetre cinsinden saklarken, is_data ham ölçüm verilerini tutar. locations_data ise harita konumlandırması için kullanılacak JSON yapılandırmasını bekler. Round Robin algoritmasının düzgün çalışması için current_station_index aktif istasyonun sırasını, station_order dinleme rotasını, last_update_time son başarılı veri alım anını ve station_timeout istasyon değiştirme eşiğini (1 saniye) yönetir. Bu yapı, sistemin gerçek zamanlı veri işleme ve istasyon yönetimi için gerekli tüm durum bilgisini merkezi olarak barındırır.

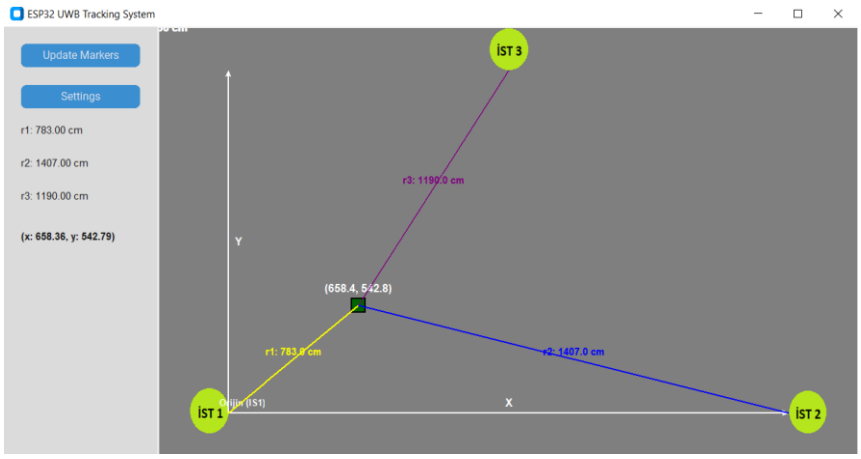
3.10.5 Koordinat sistemi ve sabit istasyonların grafiksel gösterimi

```
def draw_axes(self):
    """Koordinat eksenlerini çizer"""
    canvas_width = self.empty_area.winfo_width()
    canvas_height = self.empty_area.winfo_height()
    # Önceki eksenleri temizle
    for line_id in self.axis_lines:
        self.canvas.delete(line_id)
    self.axis_lines = []
    # Sabit istasyonların göreceli pozisyonları
    anchor_rel_pos = {
        "IS1": (0.1, 0.9),  # Sol alt (orijin)
        "IS2": (0.9, 0.9),  # Sağ alt
        "IS3": (0.5, 0.1)   # Üst orta
    }
    # Orijin noktası koordinatları
    origin_x = canvas_width * anchor_rel_pos["IS1"][0]
    origin_y = canvas_height * anchor_rel_pos["IS1"][1]
    # X ekseni çizgisi (IS1'den IS2'ye)
    is2_x = canvas_width * anchor_rel_pos["IS2"][0]
    is2_y = canvas_height * anchor_rel_pos["IS2"][1]
```

```
# Y ekseni çizgisi (IS1'den IS3'e)
is3_x = canvas_width * anchor_rel_pos["IS3"][0]
is3_y = canvas_height * anchor_rel_pos["IS3"][1]
# X ekseni çizimi
x_axis = self.canvas.create_line(
    origin_x, origin_y, is2_x, origin_y,
    fill="white", width=2, arrow="last"
)
# Y ekseni uzunluğu hesaplama
y_axis_length = origin_y - is3_y
y_axis_end_y = origin_y - y_axis_length
# Y ekseni çizimi
y_axis = self.canvas.create_line(
    origin_x, origin_y, origin_x, y_axis_end_y,
    fill="white", width=2, arrow="last"
)
# Eksen etiketleri
x_label = self.canvas.create_text(
    (origin_x + is2_x)/2, origin_y - 15,
    text="X", fill="white", font=("Arial", 12, "bold")
)
```

```
y_label = self.canvas.create_text(
    origin_x + 15, (origin_y + y_axis_end_y)/2,
    text="Y", fill="white", font=("Arial", 12, "bold")
)
# Orijin etiketi
origin_label = self.canvas.create_text(
    origin_x + 20, origin_y - 15,
    text="Orijin (IS1)", fill="white", font=("Arial", 10, "bold")
```

Arayüzde oluşturulan koordinat sistemi, IS1 istasyonunu orijin noktası (0,0) olarak alan ve fiziksel ortamdaki konumları temsil eden bir referans çerçevesi sunar. Sistem, canvas alanının %10 sol marj ve %90 alt marj kesişiminde yer alan IS1'i (canvas_width×0.1, canvas_height×0.9 koordinatlarında) merkez kabul eder. X ekseni, IS1'den IS2'ye (canvas_width×0.9, canvas_height×0.9) uzanan yatay beyaz çizgiyle, Y ekseni ise IS1'den IS3'e (canvas_width×0.5, canvas_height×0.1) uzanan dikey çizgiyle temsil edilir. Her iki eksen de ok başlarıyla sonlandırılarak pozitif yönler belirtilmiştir. Eksenlerin kesişim noktasında bulunan 'Orijin (IS1)' etiketi, kullanıcıya temel referans noktasını açıkça göstermektedir. Bu koordinat sistemi, trilaterasyon hesaplamalarında kullanılan fiziksel konumlar (IS1: 0cm,0cm; IS2: 2000cm,0cm; IS3: 1000cm,1732.05cm) ile grafik arayüz arasında ölçekli bir eşleme sağlar. Eksenlerin uzunlukları ve yönleri, gerçek dünyadaki UWB ölçüm sisteminin matematiksel modeline karşılık gelecek şekilde tasarlanmıştır.



Şekil 26. Orijin Noktası ve Koordinat Sistemi.

3.10.6 Konum belirleme yöntemi

Trilaterasyon Fonksiyonu oluşturmak için sabit alıcı istasyonların (anchor) koordinatlarını belirtmek gereklidir. Üç sabit alıcı istasyonun (anchor) koordinatları:

- $x_1, y_1 = (0, 0)$

- $x_2, y_2 = (d, 0)$

- $x_3, y_3 = (d/2, d\sqrt{3}/2)$

x_1, y_1 koordinatları, birinci sabit alıcı istasyonu; x_2, y_2 koordinatları, ikinci sabit alıcı istasyonu; x_3, y_3 koordinatları ise üçüncü sabit alıcı istasyonu göstermektedir.

Trilaterasyon hesaplamalarında kullanılan üç sabit alıcı istasyonun koordinatları şu şekilde tanımlanmıştır: Birinci sabit alıcı istasyonun koordinatları x1, y1 = (0, 0), ikinci sabit alıcı istasyonun koordinatları x2, y2 = (2000, 0) ve üçüncü sabit alıcı istasyonun koordinatları x3, y3 = (1000, 1732.05) olarak belirtilmiştir.

```
def trilateration_raw(self, r1, r2, r3):
    """Üç sabit noktadan mesafe ölçümüyle konum belirleme"""
    try:
      r1, r2, r3 = abs(r1), abs(r2), abs(r3)
```

```
            x1, y1 = 0, 0                # IS1 (orijin noktası)
            x2, y2 = 2000, 0             # IS2 (X ekseni üzerinde)
            x3, y3 = 1000, 1732,05       # IS3 (Y ekseni referansı)
            # Trilaterasyon formülleri
            A = 2 * (x2 - x1)
            B = 2 * (y2 - y1)
            C = r1**2 - r2**2 - x1**2 + x2**2 - y1**2 + y2**2
            D = 2 * (x3 - x2)
            E = 2 * (y3 - y2)
            F = r2**2 - r3**2 - x2**2 + x3**2 - y2**2 + y3**2
            x = (C * E - F * B) / (A * E - B * D)
            y = (C * D - A * F) / (B * D - A * E)
            # Sınır kontrolleri
            x = max(0, min(x, 2000))
            y = max(0, min(y, 1732,05))
            return x, y
        except Exception as e:
            print(f"Trilaterasyon hatası: {e}")  # Türkçe hata mesajı
            return self.empty_area.winfo_width()/2, self.empty_area.winfo_height()/2
```

Bu kod bölümü, hesaplanan koordinat değerlerinin geçerli aralıkta olmasını sağlayan bir sınırlama mekânizması ve hata yönetim sistemi içermektedir. Koordinat değerleri, x

ekseni için 0 ile 2000 cm arasında, y ekseni için ise 0 ile 1732.05 cm arasında sınırlandırılmaktadır. Bu sınır değerleri, sistemde tanımlı sabit alıcı istasyonların konumlarına ve ölçüm alanının fiziksel boyutlarına uygun olarak belirlenmiştir. Hesaplanan değerler bu aralıkların dışında kaldığında, otomatik olarak en yakın geçerli sınıra çekilmektedir. Ayrıca try-except bloğu ile olası hesaplama hataları kontrol altına alınmış olup, hata durumunda sistem cihazı ekranın merkezine konumlandırarak kullanıcı deneyiminin kesintisiz devam etmesini sağlamaktadır. Hata mesajları Türkçe olarak verilerek operatörler için anlaşılırlık artırılmıştır. Bu mekânizma hem matematiksel tutarlılığı hem de görsel arayüzün doğru çalışmasını garanti altına almaktadır.

Bu çalışmada, r_1, r_2, r_3 santimetre (cm) cinsinden mesafeler, A, B, C, D, E ve F'ye göre ve dönüş (x,y) yine santimetre (cm) cinsinden belirtilmiştir. Denklem 3.23'te A'yı bulma formülü şu şekilde yazılmıştır:

$$A = 2 \cdot (x_2 - x_1) \tag{3.23}$$

Denklem 3.24'te B'yi bulma formülü şu şekilde gösterilmiştir:

$$B = 2 \cdot (y_2 - y_1) \tag{3.24}$$

Denklem 3.25'te C'yi bulma formülü şöyledir:

$$C = r_1^2 - r_2^2 - x_1^2 + x_2^2 - y_1^2 + y_2^2 \tag{3.25}$$

Denklem 3.26'da D'yi bulma formülü şu şekilde yazılmıştır:

$$D = 2 \cdot (x_3 - x_2) \tag{3.26}$$

Denklem 3.27'de E'yi bulma formülü şu şekilde gösterilmiştir:

$$E = 2 \cdot (y_3 - y_2) \tag{3.27}$$

Denklem 3.28'de F'yi bulma formülü şöyledir:

$$C = r_2^2 - r_3^2 - x_2^2 + x_3^2 - y_2^2 + y_3^2 \tag{3.28}$$

Trilaterasyon Fonksiyonu, x'i bulmak ve y'yi bulmak üzere 2 formülden oluşmuştur. Trilaterasyon Fonksiyonunun x'i bulma formülü, denklem 3.29'da gösterilmiştir:

$$x = (C \cdot E - F \cdot B)/(A \cdot E - B \cdot D) \tag{3.29}$$

Trilaterasyon Fonksiyonunun y'yi bulma formülü, denklem 3.30'da gösterilmiştir:

$$y = (C \cdot D - A \cdot F)/(B \cdot D - A \cdot E) \tag{3.30}$$

3.10.7 Hareketli vericinin gerçek zamanlı takibi

Ham veriler, özel geliştirilen bir normalizasyon algoritmasıyla:

$$scale_x = (canvas_width * 0{,}8)/2000 \quad (3.31)$$

$$scale_y = (canvas_height * 0{,}8)/1732{,}05 \quad (3.32)$$

formülleri kullanılarak piksel koordinat sistemine dönüştürülmüştür.

Bu formüllerde;

scale_x: 2000cm maks X,

scale_y: 1732.05cm maks Y'yi temsil etmektedir.

```
def update_marker_position(self):
    """Hareketli vericinin konumunu günceller"""
    d1_cm = self.device_data["IS1"]
    d2_cm = self.device_data["IS2"]
    d3_cm = self.device_data["IS3"]
    # Konum hesapla
    x_cm, y_cm = self.trilateration_raw(d1_cm, d2_cm, d3_cm)
    # Mesafe etiketlerini güncelle
    self.distance_labels["IS1"].configure(text=f"r1: {d1_cm:.2f} cm")
    self.distance_labels["IS2"].configure(text=f"r2: {d2_cm:.2f} cm")
```

```
self.distance_labels["IS3"].configure(text=f"r3: {d3_cm:.2f} cm")

self.coord_label_widget.configure(text=f"(x: {x_cm:.2f}, y: {y_cm:.2f})")

# Ekran koordinatlarına dönüşüm

canvas_width = self.empty_area.winfo_width()

canvas_height = self.empty_area.winfo_height()

max_x_cm = 400

max_y_cm = 346.41

scale_x = (canvas_width * 0.8) / max_x_cm

scale_y = (canvas_height * 0.8) / max_y_cm

scale = min(scale_x, scale_y)

offset_x = canvas_width * 0.1

offset_y = canvas_height * 0.1

x_pixel = x_cm * scale + offset_x

y_pixel = canvas_height - (y_cm * scale + offset_y)

self.update_marker_on_canvas(x_pixel, y_pixel, x_cm, y_cm)

self.canvas.delete("distance_line")
```

Elde edilen (x, y) koordinatları, ekran üzerinde görsel olarak temsil edilebilmesi için piksel cinsine dönüştürülmüştür. Bu dönüşümde offset_x ve offset_y değişkenleriyle, çizim alanının kenarlarından %10 boşluk bırakılarak konumlama yapılmış; böylece simgelerin ekran kenarına yaslanması önlenmiştir.

Y ekseni için yapılan dönüşümde, ekran koordinat sisteminin üstten aşağı doğru artması nedeniyle canvas_height - (y_pixel) formülü uygulanmış; bu sayede pozitif yöndeki Y ekseni yukarı yönlü olacak şekilde ayarlanmıştır.

Son adımda, hesaplanan x_pixel ve y_pixel değerleri kullanılarak update_marker_on_canvas fonksiyonu çağrılmış ve hareketli vericinin yeni konumu arayüzde yeşil kutu ile işaretlenmiştir. Ayrıca, önceki konumlara ait çizgiler self.canvas.delete("distance_line") komutu ile silinmiş ve görüntü güncel tutulmuştur.

Bu yapı sayesinde, sistem gerçek zamanlı olarak konum güncelleyebilmekte ve kullanıcılara dinamik bir takip deneyimi sunmaktadır.

3.10.8 Mesafe ve koordinatların kullanıcıya sunumu

```
# Sabit istasyonların ekran pozisyonları
    anchor_rel_pos = {
       "IS1": (0.1, 0.9),
       "IS2": (0.9, 0.9),
       "IS3": (0.5, 0.1),
    }
    distances = {"IS1": d1_cm, "IS2": d2_cm, "IS3": d3_cm}
    colors = {"IS1": "yellow", "IS2": "blue", "IS3": "purple"}
    labels = {"IS1": "r1", "IS2": "r2", "IS3": "r3"}
    # Her istasyon için çizgi ve etiket oluştur
    for station in ["IS1", "IS2", "IS3"]:
       rel_x, rel_y = anchor_rel_pos[station]
       anchor_x = canvas_width * rel_x
```

```
anchor_y = canvas_height * rel_y

# Mesafe çizgisi

self.canvas.create_line(anchor_x, anchor_y, x_pixel, y_pixel,

                fill=colors[station], width=2, tags="distance_line")

# Mesafe etiketi

mid_x = (anchor_x + x_pixel) / 2

mid_y = (anchor_y + y_pixel) / 2 - 10

label_text = f"{labels[station]}: {distances[station]:.1f} cm"

self.canvas.create_text(mid_x, mid_y,

                text=label_text,

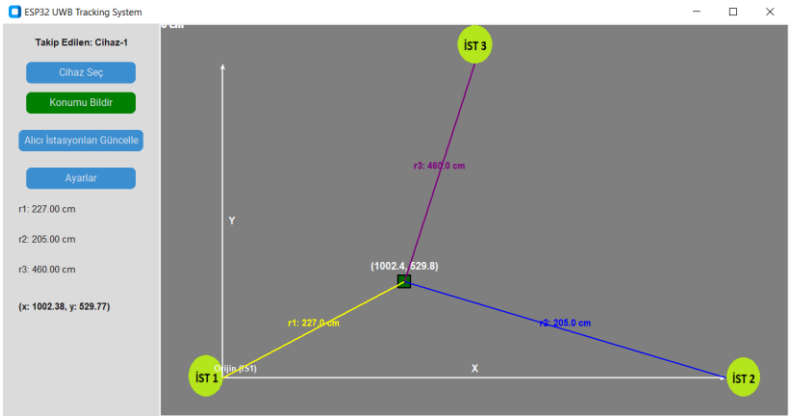
                fill=colors[station],

                font=("Arial", 11, "bold"),

                tags="distance_line")
```

Bu kod parçası, kullanıcı arayüzünde sabit alıcı istasyonlar ile hareketli verici arasındaki mesafe ilişkilerini görsel olarak temsil eden bir mekânizma oluşturur. Sabit istasyonların ekrandaki konumları, pencere boyutlarına göre dinamik olarak hesaplanarak belirlenir; IS1 sol alt köşede (%10 x, %90 y), IS2 sağ alt köşede (%90 x, %90 y) ve IS3 üst orta bölgede (%50 x, %10 y) konumlandırılır. Her bir istasyon için, hareketli vericinin bulunduğu noktaya (x_pixel, y_pixel) renk kodlu çizgiler çizilir - IS1 sarı, IS2 mavi ve IS3 mor renkte olacak şekilde ayarlanmıştır. Bu çizgilerin orta noktalarına, ilgili istasyona olan gerçek mesafeyi gösteren etiketler yerleştirilir; etiketler "r1: 150.3 cm" formatında, Arial yazı tipinde ve 11 punto büyüklüğünde bold olarak görüntülenir. Mesafe bilgileri, distances sözlüğünden alınarak bir ondalık basamak hassasiyetinde sunulur. Tüm bu grafik öğeleri "distance_line" etiketiyle işaretlenerek, sonraki güncellemelerde kolayca

silinip yeniden oluşturulabilir hale getirilmiştir. Bu yapı, kullanıcıların sistemdeki mesafe ilişkilerini anlık olarak ve görsel bir şekilde takip edebilmesine olanak tanır.



Şekil 27. Mesafe İlişkilerinin Görselleştirilmesi.

3.10.9 Hareketli verici konumunun görüntülenmesi

Bu fonksiyon, hareketli vericinin konumunu ve koordinat bilgisini grafik arayüzde gerçek zamanlı olarak güncelleyen bir işlem gerçekleştirir. Fonksiyon, aldığı x ve y piksel koordinatlarını kullanarak yeşil kutunun (self.green_box) boyutlarını ve konumunu belirler; kutunun sol üst (left, top) ve sağ alt (right, bottom) köşe noktalarını hesaplayarak bu değerleri canvas üzerinde günceller. Aynı zamanda, kutunun üst kısmında görüntülenen koordinat etiketini (coord_label) kontrol eder: eğer etiket mevcutsa siler ve yeni bir etiket oluşturarak vericinin gerçek dünya koordinatlarını (x_cm, y_cm) cm cinsinden, beyaz renkte ve Arial yazı tipiyle ekrana yansıtır. Bu işlemler, kullanıcıya hareketli vericinin anlık konumunu net bir şekilde gösterirken, arayüzün tutarlı ve temiz kalmasını sağlar. Koordinat etiketi, yeşil kutunun 15 piksel üstüne yerleştirilerek okunabilirlik artırılır ve tüm grafik öğeleri etiketlerle (tags) işaretlenerek sonradan kolayca yönetilebilir hale getirilir.

```
def update_marker_on_canvas(self, x, y, x_cm, y_cm):

    """Hareketli vericiyi ve koordinat bilgisini günceller"""

    box_size = 20

    left = x - box_size / 2
```

```
top = y - box_size / 2
right = x + box_size / 2
bottom = y + box_size / 2
self.canvas.coords(self.green_box, left, top, right, bottom)
if hasattr(self, 'coord_label'):
    self.canvas.delete(self.coord_label)
self.coord_label = self.canvas.create_text(
    x,
    top - 15,
    text=f"({x_cm:.1f}, {y_cm:.1f})",
    fill="white",
    font=("Arial", 12, "bold"),
    tags="coord_label"
)
```

Bu fonksiyon, hareketli verici cihazın grafik arayüzdeki görsel temsilini oluşturan bir yapılandırma görevi görür. Fonksiyon, 20 piksel boyutunda kare şeklinde bir yeşil kutu (self.green_box) oluşturur ve bu kutuyu ekranın tam ortasına yerleştirir. Kutunun konumu, empty_area widget'ının genişlik ve yükseklik değerleri (winfo_width(), winfo_height()) kullanılarak hesaplanır, böylece pencere boyutu değişse bile otomatik olarak merkezde kalması sağlanır. Kutu, #006400 HEX koduyla koyu yeşil renkte doldurulur ve siyah renkte 2 piksel kalınlığında bir dış çerçeveye sahiptir. "green_box" etiketi (tags parametresi) atanarak bu grafik öğesinin sonradan kolayca manipüle edilebilmesi (konum değişikliği, silme vb.) sağlanır. Bu yapı, kullanıcıya hareketli

vericinin konumunu net bir şekilde göstermek amacıyla tasarlanmış olup, sistemin gerçek zamanlı takip özelliğinin görsel bileşenlerinden birini oluşturur.

```
def create_green_box(self):
    """Hareketli verici için yeşil kutu oluşturur"""
    box_size = 20
    cx = self.empty_area.winfo_width() // 2
    cy = self.empty_area.winfo_height() // 2
    self.green_box = self.canvas.create_rectangle(
        cx - box_size//2, cy - box_size//2,
        cx + box_size//2, cy + box_size//2,
        fill="#006400",  # Koyu yeşil renk
        outline="black",
        width=2,
        tags="green_box"
    )
```

Grafik arayüzün temel bileşenlerinden biri olan create_green_box metodu, hareketli vericinin grafik arayüzde görsel temsilini sağlamak için 20x20 piksel boyutlarında yeşil bir kutu oluşturur ve bu kutuyu ekranın tam merkezine yerleştirerek sistemin başlangıç görselini hazırlar. Kutunun konumu, empty_area widget'ının genişlik ve yükseklik değerleri kullanılarak hesaplanan (cx, cy) koordinatlarına göre belirlenir; koyu yeşil renk (#006400) ile doldurulan dikdörtgen, 2 piksel kalınlığında siyah çerçevesi ve "green_box" etiketiyle hem görsel olarak belirginleştirilir hem de sonraki güncellemeler için referans alınabilir hale getirilir. Bu metodun oluşturduğu görsel element, update_marker_on_canvas metodu tarafından sürekli güncellenerek kullanıcıya hareketli

vericinin gerçek zamanlı konum bilgisini net ve anlaşılır bir şekilde sunar, böylece sistemin takip özelliği etkili bir görsel geri bildirimle desteklenmiş olur.

3.10.10 Çoklu istasyon veri güncelleme ve zaman aşımı

```
for part in message.split(","):
                for key in ["IS1", "IS2", "IS3"]:
                    if key in part:
                        try:
                            distance = float(part.split(":")[1]) * 100  # cm'ye çevir
                            if distance < 20 or distance > 1000:  # Geçersiz aralık kontrolü
                                continue
                            if distance != self.device_data[key]:
                                self.device_data[key] = distance
                                self.last_received_time[key] = time.time()
                                updated = True
                        except Exception as e:
                            print(f"{key} verisi ayrıştırma hatası: {e}")  # Türkçe hata mesajı
            if updated:
                self.last_update_time = time.time()
                self.update_marker_position()
    except socket.timeout:
            current_time = time.time()
```

```
            if current_time - self.last_update_time > self.station_timeout:
                self.current_station_index = (self.current_station_index + 1) % len(self.station_order)
                print(f"\nZaman aşımı - Yeni istasyon: {self.station_order[self.current_station_index]}")  # Türkçe çıktı
                self.last_update_time = current_time
        except Exception as e:
            print(f"\nUDP hatası: {e}")  # Türkçe hata mesajı
```

Bu kod parçası, UDP üzerinden alınan veri mesajının içeriğini istasyonlara göre ayrıştırarak her bir istasyon için güncel mesafe bilgisini elde etmeyi amaçlar. Mesaj virgül ile ayrılarak parçalanır ve her parça içinde "IS1", "IS2" veya "IS3" etiketleri aranır. Etiket bulunursa, ilgili parçadan mesafe bilgisi ayrıştırılır ve santimetreye çevrilir. Geçersiz mesafeler (20 cm'den küçük veya 1000 cm'den büyük) göz ardı edilir. Eğer bu mesafe, daha önce kayıtlı olan mesafeden farklıysa, hem mesafe değeri güncellenir hem de o istasyon için son veri alınma zamanı time.time() ile güncellenir. Bu sırada bir değişiklik olduysa (updated = True), bu güncellemenin ardından son güncelleme zamanı da kaydedilir ve harita üzerindeki konum yeniden çizilir. Eğer belirli bir süre boyunca hiçbir yeni veri alınmazsa (timeout durumu), zaman aşımı kontrolü yapılır; son güncelleme zamanıyla şimdiki zaman arasındaki fark belirli bir eşik değeri (station_timeout) aşarsa, current_station_index değeri döngüsel olarak artırılır ve sıradaki istasyon aktif hâle getirilir. Böylece round robin algoritmasına uygun şekilde istasyonlar arası geçiş sağlanır.

3.10.11 Esp32 uwb kontrol paneli: cihaz yönetimi ve sistem ayarları

```
  def create_widgets(self):
      """Arayüz bileşenlerini oluşturur"""
      # Izgara düzeni ayarları
      self.grid_columnconfigure(0, weight=0)
```

```
        self.grid_columnconfigure(1, weight=1)
        self.grid_rowconfigure(0, weight=1)
        # Sol panel (kontrol butonları ve bilgiler)
        self.frame_left = customtkinter.CTkFrame(master=self, width=150,
corner_radius=0, fg_color=None)
        self.frame_left.grid(row=0, column=0, padx=0, pady=0, sticky="nsew")
        self.frame_left.grid_rowconfigure(9, weight=1)
        # Cihaz bilgi ve kontrol widget'ları
        self.device_label = customtkinter.CTkLabel(
            master=self.frame_left,
            text=f"Takip Edilen: {self.target_device_name}",  # Türkçe etiket
            font=("Arial", 12, "bold")
        )
        self.device_label.grid(pady=(10, 0), padx=(20, 20), row=0, column=0)
        self.device_select_button = customtkinter.CTkButton(
            master=self.frame_left,
            text="Cihaz Seç",  # Türkçe buton metni
            command=self.open_device_selection
        )
        self.device_select_button.grid(pady=(10, 0), padx=(20, 20), row=1, column=0)
        self.notify_button = customtkinter.CTkButton(
```

```
        master=self.frame_left,
        text="Konumu Bildir",  # Türkçe buton metni
        command=self.send_notification,
        fg_color="green"
    )
    self.notify_button.grid(pady=(10, 0), padx=(20, 20), row=2, column=0)
    # Diğer kontrol butonları
    self.update_button = customtkinter.CTkButton(
        master=self.frame_left,
        text="Alıcı İstasyonları Güncelle",  # Türkçe buton metni
        command=self.load_default_locations
    )
    self.update_button.grid(pady=(20, 0), padx=(20, 20), row=3, column=0)
    self.settings_button = customtkinter.CTkButton(
        master=self.frame_left,
        text="Ayarlar",  # Türkçe buton metni
        command=self.open_settings
    )
    self.settings_button.grid(pady=(20, 0), padx=(20, 20), row=4, column=0)
```

Bu kod parçası, create_widgets metodu içinde bir GUI uygulamasının sol panelini oluşturur. Öncelikle, frame_left adında bir çerçeve (CTkFrame) oluşturularak ana

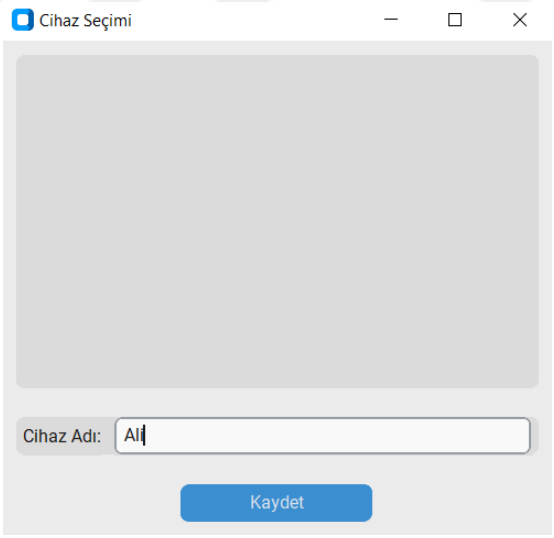
pencerenin sol tarafına yerleştirilir. Bu çerçevenin içine sırasıyla çeşitli bileşenler eklenir: İlk olarak, device_label adında bir etiket (CTkLabel) kullanılarak takip edilen cihazın adı gösterilir ("Takip Edilen: Cihaz-1"). Ardından, "Cihaz Seç" butonu (CTkButton) eklenir ve bu butona tıklandığında open_device_selection metodu çağrılır. Sonraki satırda, "Konumu Bildir" butonu yer alır ve bu buton send_notification metodunu tetikler. Daha sonra, "Alıcı İstasyonları Güncelle" butonu eklenir ve bu buton load_default_locations metodunu çalıştırarak istasyon verilerini yeniler. Son olarak, "Ayarlar" butonu eklenir ve bu buton open_settings metodunu çağırarak kullanıcının uygulama ayarlarını düzenlemesine olanak tanır. Tüm bu bileşenler, grid yöntemi kullanılarak düzenli bir şekilde konumlandırılır ve her birinin farklı satırlarda (row 0'dan row 4'e kadar) ve aynı sütunda (column 0) yer alması sağlanır. Bu yapı, kullanıcı dostu bir arayüz sunarak temel işlevlere erişimi kolaylaştırır.

```
def open_settings(self):
    """Ayarlar penceresini açar"""
    settings_window = customtkinter.CTkToplevel(self)
    settings_window.title("Ayarlar")  # Türkçe pencere başlığı
    settings_window.geometry("500x400")
    labels = []
    entries = []
    locations = self.locations_data if self.locations_data else {}
    # Bölge ayarları için giriş alanları
    for i in range(3):
        region_name = f"Bölge {i+1}"  # Türkçe bölge adı
        default_value = locations.get(f"Region {i+1}", 400)  # Varsayılan 400 cm
        label = customtkinter.CTkLabel(settings_window, text=f"{region_name} (cm)")
```

```
            label.grid(row=i, column=0, padx=10, pady=10)
            entry = customtkinter.CTkEntry(settings_window, width=200)
            entry.insert(0, default_value)
            entry.grid(row=i, column=1, padx=10, pady=10)
            labels.append(label)
            entries.append(entry)
        def save_settings():
            """Ayarları kaydeder ve pencereyi kapatır"""
            new_locations = {}
            for i, entry in enumerate(entries):
                try:
                    new_locations[f"Region {i+1}"] = int(entry.get())
                except ValueError:
                    new_locations[f"Region {i+1}"] = 400  # Varsayılan 400 cm
            with open(self.locations_file, "w") as file:
                json.dump(new_locations, file, indent=4)
            self.load_default_locations()
            settings_window.destroy()
        save_button = customtkinter.CTkButton(settings_window, text="Kaydet",
command=save_settings)  # Türkçe buton metni
        save_button.grid(row=3, column=0, columnspan=2, pady=20)
```

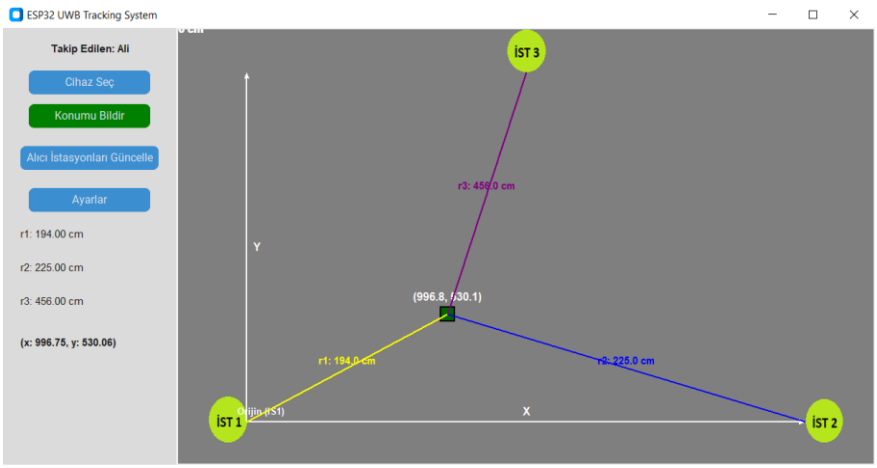
Bu kod, bir ayarlar penceresi oluşturan open_settings metodunu tanımlar. Metod çalıştırıldığında, customtkinter.CTkToplevel kullanarak "Ayarlar" başlıklı yeni bir pencere açar ve bu pencereyi 500x400 piksel boyutunda ayarlar. Pencerede, kullanıcının üç farklı bölge için mesafe değerlerini girebileceği bir arayüz oluşturulur. Her bölge ("Bölge 1", "Bölge 2", "Bölge 3") için bir etiket ve giriş alanı (CTkEntry) oluşturulur, bu alanlara varsayılan olarak locations.json dosyasından okunan değerler veya dosya yoksa 2000 cm değeri yerleştirilir. Kullanıcı değerleri değiştirip "Kaydet" butonuna bastığında, save_settings fonksiyonu çalışır. Bu fonksiyon, girilen değerleri kontrol eder (geçersiz girişlerde varsayılan 2000 cm değerini kullanır) ve yeni değerleri locations.json dosyasına kaydeder. Kayıt işlemi tamamlandıktan sonra load_default_locations metodu çağrılarak ayarlar uygulanır ve pencere kapatılır. Tüm bu işlemler, kullanıcı dostu bir arayüzle basitçe gerçekleştirilir ve uygulamanın konfigürasyonunun kolayca düzenlenmesini sağlar.

Şekil 28'de cihaz seçme ayarı gösterilmiştir.



Şekil 28. Cihaz Seçme Ayarı.

Şekil 29'da ise, cihazın seçimi gösterilmiştir.



Şekil 29. Cihazın Seçimi.

Arayüzün sol tarafında bulunan "Alıcı İstasyonları Güncelle" bölümü, sistemdeki sabit alıcı istasyonlarının uzaklık parametrelerinin manuel olarak ayarlanabilmesini sağlamaktadır. Şekil 32'de görüldüğü üzere, bu bölüm üç temel parametreden oluşmaktadır:

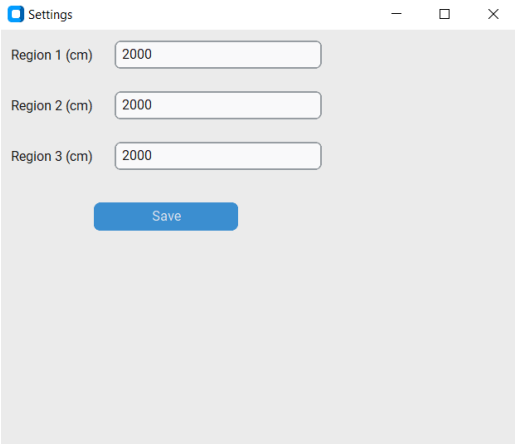
Region 1 (Bölge 1): Birinci sabit alıcı istasyonu (IS1) ile hareketli verici arasındaki gerçek zamanlı uzaklık değerini temsil eder (örnek değer: 194.00 cm)

Region 2 (Bölge 2): İkinci sabit alıcı istasyonu (IS2) ile hareketli verici arasındaki anlık mesafeyi gösterir (örnek değer: 225.00 cm)

Region 3 (Bölge 3): Üçüncü sabit alıcı istasyonu (IS3) ile hareketli verici cihaz arasındaki uzaklık bilgisini içerir (örnek değer: 450.00 cm)

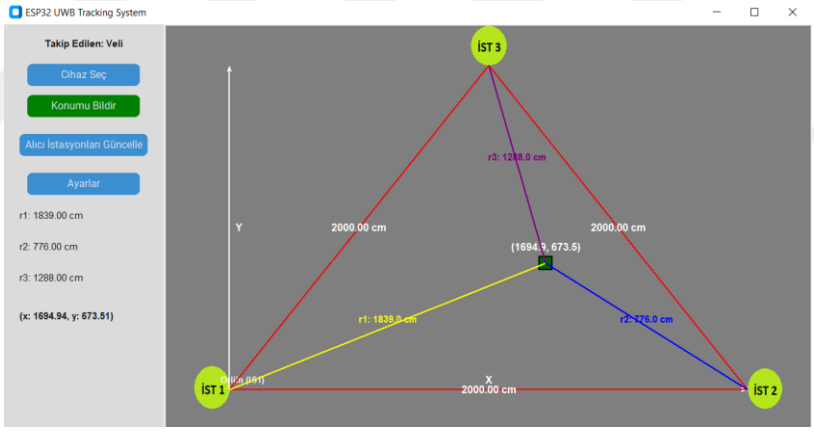
Bu parametreler, sistemin temel konfigürasyon dosyası olan locations.json içinde saklanmakta ve her sistem başlatıldığında bu dosyadan otomatik olarak okunmaktadır. Kullanıcılar, alıcı istasyonların fiziksel konumlarında yaptıkları değişiklikleri bu arayüz üzerinden güncelleyerek, sistemin doğru konum hesaplaması yapmasını sağlayabilirler.

Görselde ayrıca (x: 996.75, y: 530.00) şeklinde belirtilen koordinat değerleri, bu üç istasyondan alınan uzaklık verilerinin trilaterasyon algoritması ile işlenmesi sonucu hesaplanan hareketli vericinin anlık konumunu göstermektedir. Bu entegre yapı sayesinde sistem, kullanıcı dostu bir arayüzle birlikte yüksek hassasiyetli konum takibi yapabilmektedir.



Şekil 30. Sabit Alıcı İstasyon Mesafelerinin Güncellenmesi.

Bütün işlemler tamamlandıktan sonra, konum arayüzünün son hali **Şekil 31**'de gösterilmiştir.



Şekil 31. UWB Takip Sisteminin Görsel Arayüz Analizi.

Arayüzde yer alan görsel öğeler, sistemin mekânsal organizasyonunu ve takip edilen hareketli vericinin konumunu detaylı bir şekilde temsil etmektedir. Kırmızı çizgiler, sabit alıcı istasyonlar arasındaki fiziksel mesafeleri belirtmekte ve bu istasyonların birbirine olan uzaklıklarını ölçekli biçimde görselleştirmektedir. Bu çizgiler, sabit noktaların konumlarının sabit ve bilinir olduğunu ifade ederken, sistemin ölçüm geometrisinin temelini oluşturur. Beyaz çizgiler ise, koordinat düzleminin orijini ve yönlerini ifade etmekte; genellikle X ve Y eksenleri şeklinde gösterilerek, sistemin konumlandırma işlemleri için referans noktası olarak hizmet etmektedir.

Arayüzde, sarı çizgi hareketli verici ile İST1 arasındaki mesafeyi; mavi çizgi hareketli verici ile İST2 arasındaki mesafeyi ve mor çizgi hareketli verici ile İST3 arasındaki mesafeyi temsil eder. Bu çizgiler yalnızca görsel bağlantılar değil; aynı zamanda anlık mesafe ölçümlerinin hangi sabit istasyondan geldiğini ve ölçüm ilişkilerini de kullanıcıya sunar. Her ölçüm güncellendiğinde bu çizgiler yeniden çizilir ve sistemin dinamik yapısını ortaya koyar. Arayüzün sol orta kısmında, "r1", "r2" ve "r3" etiketleri altında sırasıyla İST1, İST2 ve İST3'e olan mesafe bilgileri yer almaktadır. Bu sayısal değerlerin hemen altında, hareketli vericinin hesaplanan (x, y) koordinatları yer alır ve bu bilgiler gerçek zamanlı olarak güncellenir.

Sol üst köşede kullanıcıya sunulan çeşitli kontrol seçenekleri yer almaktadır. "Cihaz Seç" seçeneği, takip edilmek istenen vericiyi belirlemek için kullanılırken; "Konumu Bildir" komutu, seçilen cihaza pozisyonunu paylaşması için talimat gönderir. "Alıcı İstasyonları Güncelle" ve "Ayarlar" seçenekleri ise sistemin yapılandırılabilir parametrelerini değiştirmek ve istasyonlar arası mesafeleri yeniden tanımlamak için kullanılmaktadır. Bu kontrol paneli sayesinde kullanıcı, sistemin genel davranışını ve hangi cihazın izleneceğini kolaylıkla yönetebilir.

Sabit alıcı istasyonlar, açık yeşil renkli dairelerle temsil edilir ve arayüz üzerinde eşkenar üçgen oluşturacak şekilde konumlandırılmıştır. İST1 ekranın sol alt köşesinde, İST2 sağ alt köşesinde ve İST3 üst orta bölümde yer alır. Bu düzen, trilaterasyon algoritması için uygun bir geometrik yapı oluşturur. Hareketli verici ise koyu yeşil renkli küçük bir kare ile gösterilmekte olup genellikle ekranın ortasında, sabit istasyonlardan alınan anlık mesafe verilerine göre konumlandırılmaktadır. Bu kutunun hemen üst kısmında, vericinin (x, y) koordinatları beyaz renkte ve okunaklı bir şekilde görüntülenir. Böylece kullanıcı, görsel olarak sistemin çalışmasını izlerken aynı zamanda sayısal konum bilgilerine de doğrudan ulaşabilmektedir.

4. BULGULAR VE TARTIŞMA

Üç adet ESP32 UWB Pro sabit alıcı istasyon ve bir adet ESP32 UWB Pro hareketli ile oluşturulan UWB (Ultra Geniş Bant) teknolojili kapalı mekânlardaki gerçek zamanlı konum takip sisteminin performansı analiz edilmiştir.

Bu çalışmada, UWB (Ultra Geniş Bant) teknolojisi kullanılarak kapalı mekânlarda gerçek zamanlı konum takibi sağlayan bir sistem tasarlanmış ve test edilmiştir. Sistemin performansı, doğruluk, kararlılık ve ölçeklenebilirlik açısından değerlendirilmiş olup elde edilen bulgular şu şekildedir:

4.1 Arduino Bulguları

Veriler gönderildi: MAC:86:17:5B:D5:A9:9A:E2:9C,IS1:2.13,IS2:2.27,IS3:4.75

IS1 - Mesafe: 2.10 m, Güç: -96.35 dBm

IS1 - Mesafe: 2.09 m, Güç: -96.06 dBm

IS1 - Mesafe: 2.10 m, Güç: -99.86 dBm

IS1 - Mesafe: 2.12 m, Güç: -96.31 dBm

IS1 - Mesafe: 2.15 m, Güç: -95.92 dBm

IS1 - Mesafe: 2.10 m, Güç: -95.83 dBm

IS1 ölçümü tamamlandı, IS2'ye geçiliyor...

IS2 - Mesafe: 2.15 m, Güç: -90.87 dBm

IS2 - Mesafe: 2.18 m, Güç: -90.71 dBm

IS2 - Mesafe: 2.19 m, Güç: -90.81 dBm

IS2 - Mesafe: 2.16 m, Güç: -90.75 dBm

IS2 - Mesafe: 2.14 m, Güç: -90.49 dBm

IS2 - Mesafe: 2.19 m, Güç: -90.28 dBm

IS2 ölçümü tamamlandı, IS3'e geçiliyor...

IS3 - Mesafe: 4.72 m, Güç: -99.04 dBm

IS3 - Mesafe: 4.68 m, Güç: -98.76 dBm

IS3 - Mesafe: 4.67 m, Güç: -98.72 dBm

IS3 - Mesafe: 4.71 m, Güç: -98.52 dBm

IS3 - Mesafe: 4.70 m, Güç: -98.78 dBm

IS3 - Mesafe: 4.70 m, Güç: -98.86 dBm

IS3 ölçümü tamamlandı, IS1'e geçiliyor...

Hareketli verici cihaz ile sabit alıcı istasyonlar (IS1, IS2, IS3) arasında gerçekleştirilen iletişim, round robin (döngüsel) algoritmasına dayalıdır. Arduino seri monitöründen elde edilen çıktılar, bu algoritmanın sırasıyla uygulandığını ortaya koymaktadır. Ölçüm döngüsü boyunca her sabit istasyondan kısa süreli olarak ardışık mesafe ve sinyal gücü (RSSI) verileri alınmakta; ardından sistem sıradaki istasyona geçmektedir.

İlk olarak IS1 ile 1 saniyelik sürede 6 adet ölçüm yapılmakta, mesafeler 2.09–2.15 metre, sinyal güçleri ise -95.83 dBm ile -99.86 dBm arasında değişmektedir.

Ardından sistem otomatik olarak IS2’ye geçmekte ve burada da benzer şekilde 6 ölçüm gerçekleştirilmektedir. Bu istasyonda mesafeler 2.14–2.19 metre, sinyal güçleri ise -90.28 dBm ile -90.87 dBm aralığında kaydedilmiştir.

Son olarak IS3 ile iletişim kurulmakta; ölçülen mesafeler 4.67–4.72 metre, sinyal güçleri ise -98.52 dBm ile -99.04 dBm arasında yer almaktadır. Üç istasyona ait veriler tamamlandığında sistem döngüye yeniden IS1 ile başlayarak ölçüm sürecini sürdürmektedir.

Bu yapı, tüm istasyonlardan eşit zaman aralıklarında ve eşit sayıda veri toplanmasını sağlayarak sistemin kararlı ve tutarlı çalıştığını göstermektedir. Ayrıca, mesafe arttıkça sinyal gücünde azalma gözlemlenmiş, bu da sistemin fiziksel çevre koşullarına duyarlılığını ortaya koymuştur. Sonuç olarak, sistem; hareketli vericiden sabit alıcılara doğru sıralı ve düzenli veri aktarımı gerçekleştirmekte, konum hesaplaması için gereken güncel verileri sağlamaktadır.

4.2 Terminal Bulguları

```
import socket

import threading

import time

UDP_IP = "0.0.0.0"

PORT = 1234

# Round Robin için değişkenler

station_order = ["IS1", "IS2", "IS3"]

current_station_index = 0

last_update_time = time.time()

station_timeout = 1.0

# Veri saklama yapısı

device_data = {"IS1": 0.0, "IS2": 0.0, "IS3": 0.0}

# Takip edilecek hedef cihaz MAC adresi

target_device_mac = None  # Arayüzden bu atanmadan veri işlenmeyecek
```

```
def set_target_mac(mac_address):
    global target_device_mac
    target_device_mac = mac_address
    print(f"Hedef MAC adresi ayarlandı: {target_device_mac}")
def handle_udp_data():
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.bind((UDP_IP, PORT))
    sock.settimeout(0.1)
    print(f"Port {PORT} üzerinde dinleniyor...")
    global current_station_index, last_update_time, device_data, target_device_mac
    while True:
        try:
            data, addr = sock.recvfrom(1024)
            message = data.decode('utf-8').strip()
            print(f"\nAlınan ham veri: {message}")
            # MAC adresini kontrol et
            mac_address = None
            if "MAC:" in message:
                mac_start = message.find("MAC:") + 4
               mac_end = message.find(",", mac_start)
```

```
            mac_address = message[mac_start:mac_end] if mac_end != -1 else message[mac_start:]
            if target_device_mac:
                if mac_address != target_device_mac:
                    print(f"Hedef cihaz değil ({mac_address}), veri işlenmedi.")
                    continue
            # Veri işleme
            updated = False
            for part in message.split(","):
                if ":" in part and "MAC" not in part:
                    key, value = part.split(":")
                    key = key.strip()
                    if key in device_data:
                        try:
                            mesafe = float(value) * 100  # metre → cm
                            if 20 <= mesafe <= 2000:
                                if mesafe != device_data[key]:
                                    device_data[key] = mesafe
                                    print(f"{key} verisi güncellendi: {mesafe:.2f}")
                                    updated = True
                        except ValueError:
```

```
                    print(f"{key} veri ayrıştırma hatası: {value}")
            if updated:
                last_update_time = time.time()
                print("\n--- Güncel Veriler ---")
                print("Ham veriler (cm):")
                for device_id, value in device_data.items():
                    print(f"{device_id}: {value:.2f}")
                print("="*30)
        except socket.timeout:
            current_time = time.time()
            if current_time - last_update_time > station_timeout:
                current_station_index = (current_station_index + 1) % len(station_order)
                print(f"\nTimeout - Yeni istasyon: {station_order[current_station_index]}")
                last_update_time = current_time
        except Exception as e:
            print(f"\nHata oluştu: {str(e)}")
# Gönderilecek komut: KONUM_BILDIR,MAC:<hedef_mac>
def send_position_request(target_mac, port=1234):
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    message = f"KONUM_BILDIR,MAC:{target_mac}"
    sock.sendto(message.encode(), ("127.0.0.1", port))
```

```
    sock.close()
    print(f"'{message}' mesajı gönderildi.")
# Thread başlat
udp_thread = threading.Thread(target=handle_udp_data)
udp_thread.daemon = True
udp_thread.start()
# Örnek MAC belirlemesi (GUI tarafından ayarlanmalı)
set_target_mac("86:17:5B:D5:A9:9A:E2:9C")  # Örnek hedef MAC
try:
    while True:
        time.sleep(1)
except KeyboardInterrupt:
    print("\nSunucu kapatılıyor...")
```

- Temel Yapı:

Bu sistem, UDP protokolü kullanılarak ağ üzerinden veri alımını gerçekleştirmektedir. 1234 numaralı port üzerinden dinleme yapılır ve gelen veri paketleri analiz edilir. Ölçüm sistemi, Round Robin algoritması ile üç farklı sabit istasyondan (IS1, IS2, IS3) sırayla veri toplamak üzere yapılandırılmıştır.

- Ana Bileşenler:

station_order değişkeni, veri toplanacak istasyonların sırasını tanımlar ve Round Robin algoritmasının sırasını belirler. device_data yapısı, her sabit istasyondan alınan en son mesafe verilerini saklar. target_device_mac değişkeni, yalnızca hedef olarak tanımlanmış

MAC adresine sahip cihazlardan gelen verilerin işlenmesini sağlar. Böylece diğer cihazlardan gelen veriler filtrelenmiş olur.

- Çalışma Mantığı:

Sistemin temel çalışma yapısı handle_udp_data() fonksiyonu ile sağlanır. Bu fonksiyon sürekli olarak gelen UDP paketlerini dinler ve işler. Her gelen mesaj öncelikle MAC adresi kontrolünden geçirilir. Yalnızca hedef MAC adresi ile eşleşen veriler işlenir. Ardından mesaj içerisindeki "ISx:mesafe" formatındaki veri parçaları ayrıştırılır. Ölçümler metre cinsinden geldiği için 100 ile çarpılarak santimetreye dönüştürülür. Bu veriler 20 cm ile 2000 cm aralığında geçerli kabul edilir ve device_data içine kaydedilir.

- Round Robin Mekânizması:

Sistem, her istasyon için belirlenen 1 saniyelik station_timeout süresini bekler. Bu süre içerisinde veri alınmazsa, Round Robin algoritması sayesinde sıradaki istasyona geçilir. Bu döngü sürekli tekrar eder ve böylece her sabit istasyondan düzenli aralıklarla veri toplanması sağlanır.

- Veri İşleme:

Sisteme ulaşan tipik bir ham veri örneği şu şekildedir: "MAC:86:17:5B:D5:A9:9A:E2:9C,IS1:2.13,IS2:2.27,IS3:4.75". Bu veri alındığında, her istasyon için güncellenen mesafe bilgileri terminal ekranına yazdırılır. Veriler başarılı bir şekilde ayrıştırıldığında “güncellendi” mesajı gösterilirken, geçersiz veya hatalı veriler durumunda sistem hata mesajı basar.

- Ek Fonksiyonlar:

set_target_mac() fonksiyonu, sistemin takip edeceği hedef cihazın MAC adresini tanımlar. Bu adres tanımlanana kadar hiçbir veri işlenmez. send_position_request() fonksiyonu ise, hedef cihaza "KONUM_BILDIR" komutu göndererek konum verisi yayınlamasını tetikler. Bu yapı sayesinde sistem, ihtiyaç duyduğunda cihazdan manuel olarak veri talebinde bulunabilir.

- Çalıştırma:

UDP dinleyici sistemi ayrı bir iş parçacığında (thread) başlatılır. Bu sayede ana program akışı kesintiye uğramaz. Kodun başında örnek bir MAC adresi tanımlanarak sistem başlatılır. Program, KeyboardInterrupt sinyali (örneğin Ctrl+C) alınana kadar sürekli çalışır durumda kalır.

- Kullanım Senaryosu:

Bu sistemde, hareketli bir cihaz belirli aralıklarla konum verisi yayınlar. Sunucu tarafında çalışan bu kod, gelen verileri alır, işler ve her sabit istasyon için mesafe verilerini günceller. Bu güncel veriler, başka bir sistemde kullanılmak üzere saklanabilir ya da harita üzerinde konum hesaplama amacıyla kullanılabilir.

- Hata Yönetimi:

Sistem, üç temel hata senaryosuna karşı önlem almıştır. Bunlar arasında: veri alınamaması durumunda socket zaman aşımı, ölçüm verilerinin hatalı formatta olması durumunda veri ayrıştırma hataları ve sistem geneline etki eden beklenmedik istisnalar yer alır. Tüm bu hatalar kullanıcıya bilgilendirici mesajlarla sunulur ve sistemin kesintisiz çalışması sağlanır.

Alınan ham veri: MAC:86:17:5B:D5:A9:9A:E2:9C,IS1:1.34,IS2:1.95,IS3:2.39

IS1 cihazının MAC adresi kaydedildi: 86

IS1 verisi güncellendi: 134.00

IS2 verisi güncellendi: 195.00

IS3 verisi güncellendi: 239.00

--- Güncel Veriler ---

Ham veriler (cm):

IS1: 134.00

IS2: 195.00

IS3: 239.00

==========

Timeout - Yeni istasyon: IS3

Timeout - Yeni istasyon: IS1

Alınan ham veri: MAC:86:17:5B:D5:A9:9A:E2:9C,IS1:1.36,IS2:1.93,IS3:2.35

IS1 cihazının MAC adresi kaydedildi: 86

IS1 verisi güncellendi: 136.00

IS2 verisi güncellendi: 193.00

IS3 verisi güncellendi: 235.00

--- Güncel Veriler ---

Ham veriler (cm):

IS1: 136.00

IS2: 193.00

IS3: 235.00

==========

Timeout - Yeni istasyon: IS2

Timeout - Yeni istasyon: IS3

Alınan ham veri: MAC:86:17:5B:D5:A9:9A:E2:9C,IS1:1.34,IS2:1.92,IS3:2.38

IS1 cihazının MAC adresi kaydedildi: 86

IS1 verisi güncellendi: 134.00

IS2 verisi güncellendi: 192.00

IS3 verisi güncellendi: 238.00

--- Güncel Veriler ---

Ham veriler (cm):

IS1: 134.00

IS2: 192.00

IS3: 238.00

==========

Timeout - Yeni istasyon: IS1

Timeout - Yeni istasyon: IS2

Alınan ham veri: MAC:86:17:5B:D5:A9:9A:E2:9C,IS1:1.35,IS2:1.85,IS3:2.35

IS1 cihazının MAC adresi kaydedildi: 86

IS1 verisi güncellendi: 135.00

IS2 verisi güncellendi: 185.00

IS3 verisi güncellendi: 235.00

--- Güncel Veriler ---

Ham veriler (cm):

IS1: 135.00

IS2: 185.00

IS3: 235.00

==========

- Alınan Ham Veri:

Sistem UDP üzerinden, içinde hedef cihazın MAC adresi ile üç sabit istasyonun (IS1, IS2, IS3) mesafe verilerinin bulunduğu paketleri alır. Örneğin, MAC:86:17:5B:D5:A9:9A:E2:9C,IS1:1.34,IS2:1.95,IS3:2.39 şeklindedir.

- IS1 Cihazının MAC Adresinin Kaydedilmesi:

Gelen verideki hedef cihazın MAC adresinden ilk iki hane (örneğin "86") alınarak takip edilen cihazın kimliği belirlenir ve sistemde saklanır. Bu sayede, sadece istenen cihaza ait veriler işlenir.

- IS1, IS2 ve IS3 Verilerinin Güncellenmesi:

Her istasyon için ölçülen mesafe verileri metre cinsinden alınır ve 100 ile çarpılarak santimetreye dönüştürülür (örneğin 1.34 m → 134 cm). Bu güncel mesafe değerleri sistemin veri yapısına kaydedilir.

- Güncel Verilerin Terminalde Gösterilmesi:

İşlenen ve güncellenen mesafe değerleri terminal ekranında okunabilir şekilde listelenir. Bu sayede kullanıcı veya sistem, anlık olarak her sabit istasyonun ölçtüğü güncel mesafeyi görebilir.

- Timeout ve Yeni İstasyon Geçişleri:

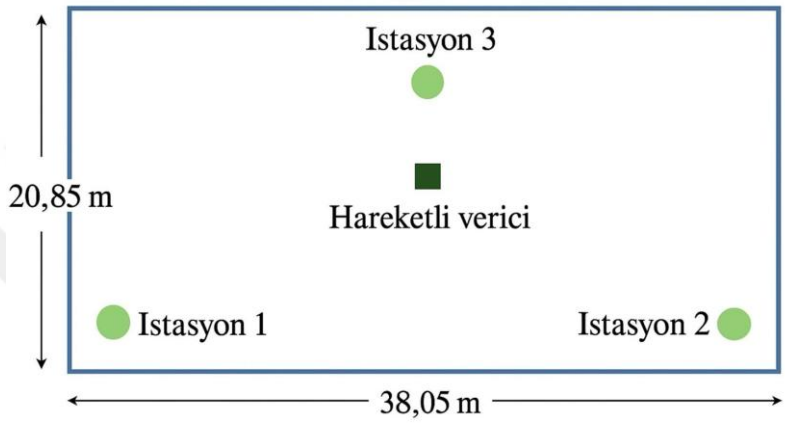
Round Robin algoritması gereği, belirlenen zaman aşımı süresi içinde bir istasyondan veri alınamazsa, sistem otomatik olarak sıradaki istasyona geçer. Bu işlem terminalde Timeout - Yeni istasyon: ISx şeklinde bildirilir ve sürekli veri akışı sağlanır.

4.3 Python Arayüz Bulguları

ESP32 UWB Pro cihazlarıyla oluşturulan UWB (Ultra Geniş Bant) teknolojili kapalı mekânlardaki gerçek zamanlı konum takip sisteminin doğruluğunu tespit etmek için, büyük bir kapalı mekânda ölçümlerin yapılması gereklidir.

Gerçek zamanlı takip sisteminin ölçümleri, kapalı tenis kortunda yapılmıştır. Kapalı tenis kortunun eni 38,05 m olup, boyu 20,85 m'dir.

Sabit alıcı istasyonlar, kapalı tenis kortunda, birbirleri arasında 20 metre (2000 cm) olacak şekilde konumlandırılmıştır. Ayrıca, trilaterasyon (üçgenleme) yönteminin yapılabilmesi için her sabit alıcı istasyon, eşkenar üçgen şeklinde dizayn edilmiştir.



Şekil 32. Sabit Alıcı İstasyonların Konumlandırılması.

Sabit alıcı istasyonlar ve hareketli verici, ölçümlerin başarılı bir şekilde yapılabilmesi için yerden 1,5 m yükseklikteki ayaklı sehpalara konulmuştur.



Şekil 33. Ayaklı Sehpalar.

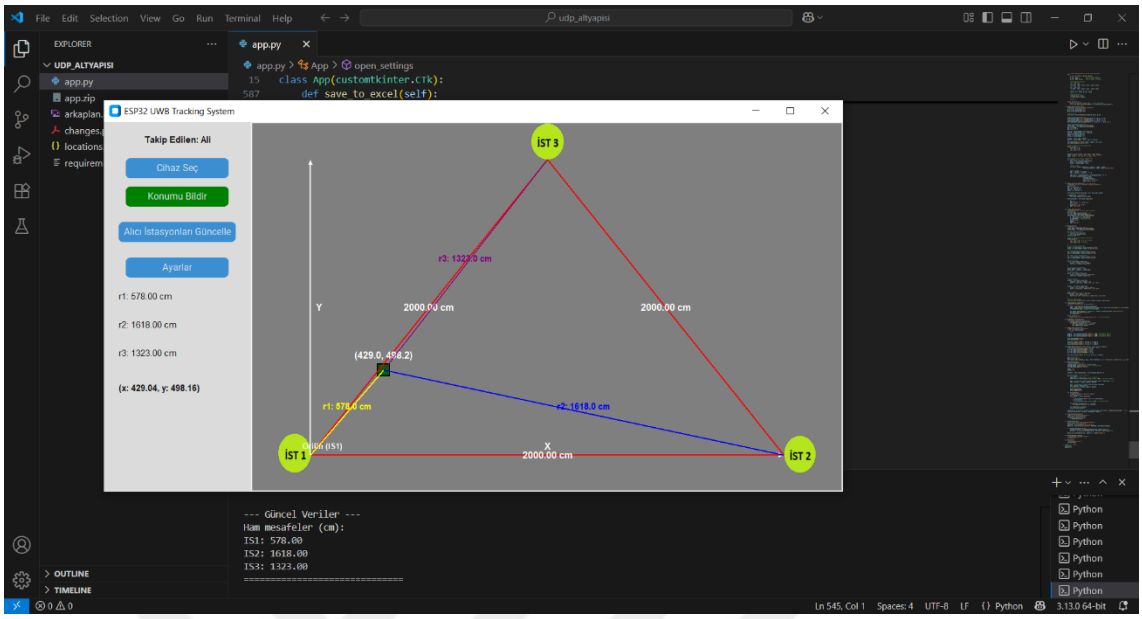
Tasarlanan bu sistem ile 3 farklı ölçüm yapılmıştır:

Birinci ölçümde, hareketli verici, birinci sabit alıcı istasyondan (IS1) 578 cm (5,78 m) uzaklıkta olacak şekilde konumlandırılmıştır.



Şekil 34. Birinci Ölçümün Yapılması.

Bu konumlandırılmaya göre arayüz çalıştırılmıştır ve arayüz sonuçları kaydedilmiştir.



Şekil 35. Birinci Ölçüm Sonuçları.

Şekil 35'te hareketli vericiye gelen mesafe verilerine göre, birinci sabit alıcı istasyon mesafesi (IS1) 578 cm (5,78 m) olarak ölçülmüştür. Birinci ölçüm sonuçlarına göre:

Gerçek Mesafe: 578 cm (5,78 m)

Ölçülen Mesafe: 578 cm (5,78 m)

Hata Payı: 0 cm (0,00 m)

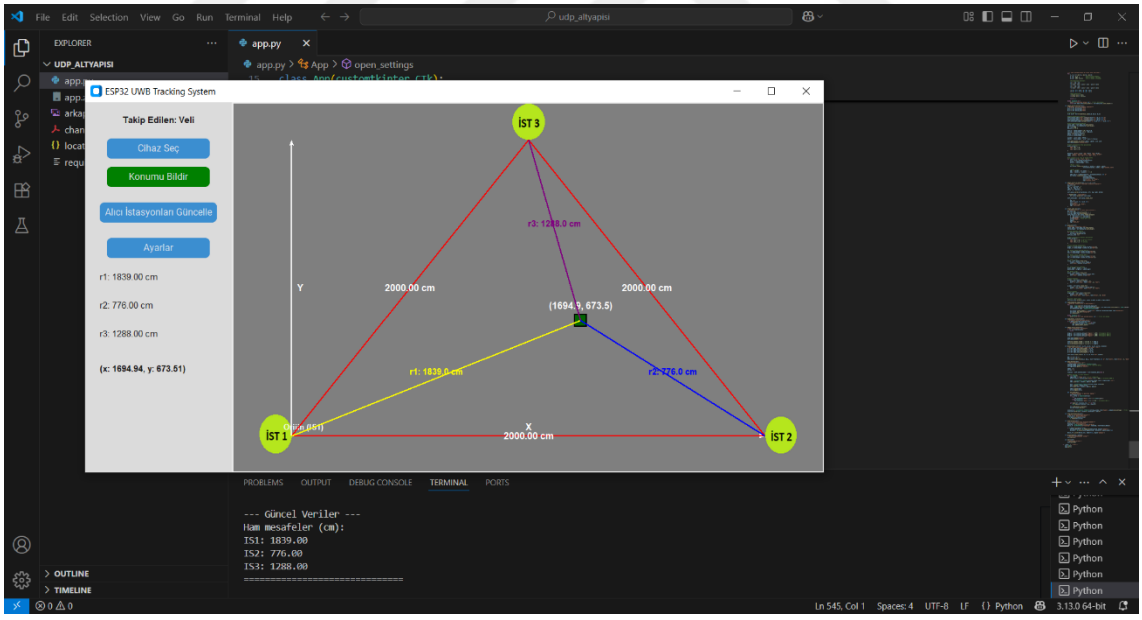
Doğruluk Oranı: %100.00

İkinci ölçümde, hareketli verici, ikinci sabit alıcı istasyondan (IS2) 777 cm (7,77 m) uzaklıkta olacak şekilde konumlandırılmıştır. İkinci ölçüm, birinci ölçüme göre daha uzak mesafede yapılmıştır.



Şekil 36. İkinci Ölçümün Yapılması.

Bu konumlandırılmaya göre arayüz çalıştırılmıştır ve arayüz sonuçları kaydedilmiştir.



Şekil 37. İkinci Ölçüm Sonuçları.

Şekil 37'de hareketli vericiye gelen mesafe verilerine göre, ikinci sabit alıcı istasyon mesafesi (IS2) 776 cm (7,76 m) olarak ölçülmüştür. İkinci ölçüm sonuçlarına göre:

Gerçek Mesafe: 777 cm (7,77 m)

Ölçülen Mesafe: 776 cm (7,76 m)

Hata Payı: 1 cm (0,01 m)

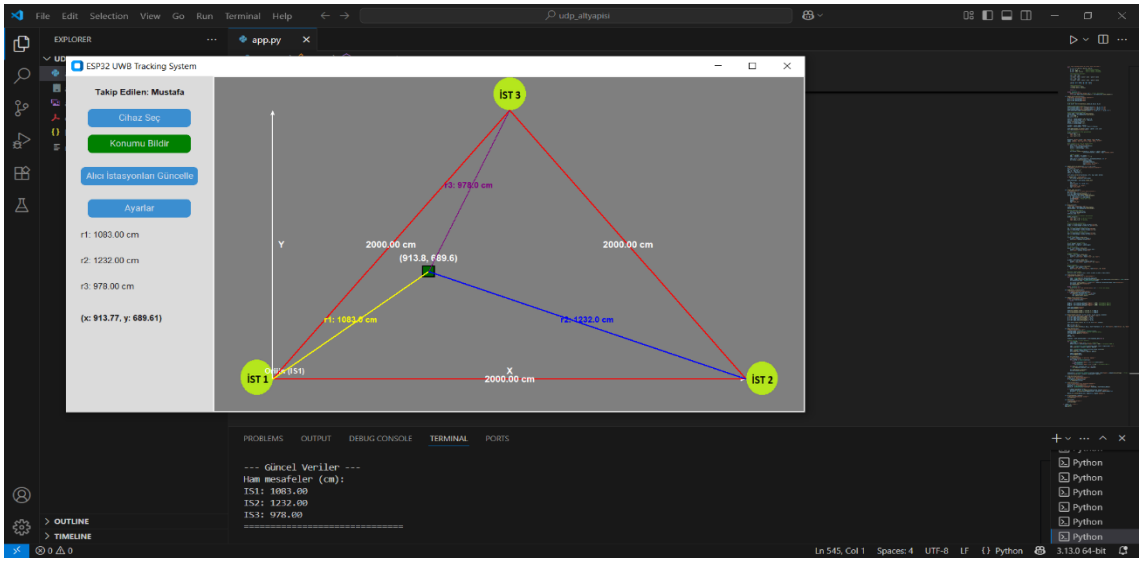
Doğruluk Oranı: %99,8712 ≈ %99,87.

Üçüncü ölçümde, hareketli verici üçüncü sabit alıcı istasyondan (IS3) 980 cm (9,80 m) uzaklıkta olacak şekilde konumlandırılmıştır. Üçüncü ölçüm, ikinci ölçüme göre daha da uzak mesafede yapılmıştır.



Şekil 38. Üçüncü Ölçümün Yapılması.

Bu konumlandırılmaya göre arayüz çalıştırılmıştır ve arayüz sonuçları kaydedilmiştir.



Şekil 39. Üçüncü Ölçüm Sonuçları.

Şekil 39'da hareketli vericiye gelen mesafe verilerine göre, üçüncü sabit alıcı istasyon mesafesi (IS3) 980 cm (9,80 m) olarak ölçülmüştür. İkinci ölçüm sonuçlarına göre:

Gerçek Mesafe: 980 cm (9,80 m)

Ölçülen Mesafe: 978 cm (9,78 m)

Hata Payı: 2 cm (0,02 m)

Doğruluk Oranı: %99,7959 ≈ %99,80.

Şekil 35, **Şekil 37** ve **Şekil 39**'daki ölçüm sonuçlarına göre; ortalama hata payı ve ortalama doğruluk oranı, aşağıda **Tablo 3**'te gösterilmiştir.

Tablo 3. Ortalama Hata Payı ve Ortalama Doğruluk Yüzdesi

ORTALAMA HATA PAYI (cm)	ORTALAMA DOĞRULUK YÜZDESİ
1 cm	%99,8890 ≈ %99,89

Tablo 3'teki ortalama hata payı ve ortalama doğruluk yüzdesi, şu şekilde hesaplanmıştır: Ortalama Hata Payı (cm): (0 cm + 1 cm + 2 cm) / 3 = 1 cm.

Ortalama Doğruluk Yüzdesi (%100,00 + %99,8712 + %99,7959) / 3 = %99,8890 ≈ %99,89.

Bu çalışmada, kapalı mekânlarda hassas konumlandırma sağlamak amacıyla UWB (Ultra Geniş Bant) teknolojisine dayalı bir sistem tasarlanmış ve uygulanmıştır. Sistemin testleri, büyük ve kapalı bir mekân olarak üniversitenin kapalı tenis kortunda gerçekleştirilmiştir. Bu mekânın seçilmesindeki temel neden, geniş alanı ve kontrollü ortam koşulları sunmasıydı. Ancak, ölçümler sırasında karşılaşılan bazı pratik zorluklar, sistemin optimizasyonu için önemli dersler sağlamıştır.

4.4 Ölçüm Sürecinde Karşılaşılan Zorluklar ve Çözümler

İlk ölçüm denemelerinde, sabit alıcı istasyonları sandalyelerin üzerine yerleştirilerek veri toplanmaya başlanmıştır. Ancak, alıcıların yerden yüksekliğinin yetersiz olması (yaklaşık 0,5 m), zemin yansımalarından kaynaklanan sinyal bozulmalarına ve ölçüm hatalarına yol açmıştır. Bu durum, özellikle UWB sinyallerinin yüksek frekanslı olması ve zemin malzemesinden (beton) kaynaklanan fazladan yansımalar nedeniyle beklenen bir sonuçtu. Sinyal kalitesinin düşmesi, başlangıçta tutarlı veri elde edilmesini engellemiş ve sistemin doğruluğunu olumsuz etkilemiştir.

Bu problemi aşmak için alıcı istasyonlarının yüksekliği, yerden 1,5 metre yükseklikteki ayaklı sehpalara taşınmıştır. Bu değişiklik, zemin yansımalarının minimize edilmesinde kritik bir rol oynamıştır ve sinyal kalitesinde belirgin bir iyileşme sağlamıştır. Yüksekliğin artırılmasıyla birlikte, alıcılar ile hareketli verici (TAG) arasındaki doğrudan görüş hattı (Line-of-Sight, LoS) koşulları iyileştirilmiş ve çoklu yol girişimleri (multipath interference) azaltılmıştır. Bu optimizasyon sonucunda, sistemin hata payı 1 cm gibi dikkate değer bir seviyeye düşürülmüş ve yüksek başarı oranı yakalanmıştır.

4.5 Yüksekliğin Konumlandırma Doğruluğuna Etkisi

Yapılan bu çalışma, UWB tabanlı sistemlerde alıcı ve vericilerin konumlandırma yüksekliğinin ne kadar kritik olduğunu ortaya koymuştur. Özellikle kapalı mekânlarda:

- Zemin Yansımaları: Fazla yüksek olmayan yerlerde, sinyallerin zeminden yansıyarak alıcıya ulaşması, mesafe ölçümlerinde belirsizliğe neden olabilmektedir.

- Engellerin Etkisi: İnsan hareketliliği veya fiziksel engeller (örneğin, tenis kortlarındaki fileler), alçak konumlandırmada sinyali daha fazla bozabilmektedir.

- Optimal Yükseklik: 1,5 metre gibi bir yükseklik, insan boyuyla uyumlu olduğundan ve çoğu engeli aşabildiğinden, LoS iletişimi garanti altına almaktadır.

4.6 Karşılaştırmalı Performans Analizi

Elde edilen 1 cm'lik hata payı, aşağıdaki çalışmalarla karşılaştırıldığında:

- He ve diğerleri (2012): 15–30 cm hata (TOA tabanlı UWB sistemi)

- Bahl ve Padmanabhan (2000): 20–30 cm hata (Wi-Fi/RSSI tabanlı sistem)

- Machaj ve diğerleri (2011): 15–25 cm hata (RSSI sıralama algoritması kullanılan sistem)

Bu sistemde, kapalı spor alanlarında literatürdeki birçok çalışmadan daha iyi performans gösterdiği görülmektedir. Ancak bu sonuçların farklı bina geometrilerinde test edilmesi gerekliliği devam etmektedir.

4.7 Kısıtlamalar ve Gelecekte Yapılabilecek Çalışmalar

Bu çalışmanın temel kısıtlamaları:

- Test ortamı çeşitliliğinin sınırlı olması

- Tek hareketli verici (TAG) ile yapılan ölçümler

- Sabit ortam koşulları

Gelecekte yapılabilecek çalışmalar için öneriler:

- Dinamik ortamlarda çoklu hareketli verici (TAG) performansı

- Yapay zeka destekli sinyal kalibrasyonu

- 3B konumlandırma için z-ekseni optimizasyonu

5. SONUÇ ve ÖNERİLER

Bu tez çalışmasında geliştirilen sistem, UWB teknolojisinin sunduğu yüksek zaman çözünürlüğü sayesinde kapalı mekânlarda santimetre düzeyinde konum belirleme başarısı göstermiştir. Yapılan testler sonucunda, sistemin ortalama hata payı 1 cm olarak ölçülmüştür ve bu doğruluk seviyesi, GPS'in kapalı ortamlarda sağlayamadığı hassasiyetin üzerine çıkıldığını ortaya koymuştur. ESP32 UWB Pro modülleri ile oluşturulan yapı, düşük maliyetli ve enerji verimli bir çözüm sunarken, güç amplifikatörü desteğiyle 200 metreye kadar genişletilebilen algılama menzili sayesinde büyük ölçekli uygulamalarda da etkinliğini göstermiştir.

Veri iletimi, UDP protokolü kullanılarak gerçekleştirilmiş ve bu yöntem sayesinde düşük gecikme süresi ile yüksek hızlı veri aktarımı elde edilmiştir. Verici ile sabit alıcılar arasında aktarılan mesafe verileri, Python programlama dili kullanılarak geliştirilen arayüzde işlenmiştir ve kullanıcıya anlık olarak sunulmuştur. Tkinter kütüphanesiyle hazırlanan masaüstü arayüzü üzerinde hareketli vericinin konumu görsel olarak takip edilebilmiştir. Böylece sistem hem kullanıcı dostu bir deneyim sunmuş hem de kolay izlenebilir bir yapı kazanmıştır.

Geliştirilen sistemin pratikteki uygulanabilirliği değerlendirildiğinde, özellikle acil durum senaryolarında hızlı ve doğru konum tespiti yapabilme kapasitesiyle güvenliği önemli ölçüde artırdığı görülmüştür. Sistem, hastane, fabrika, okul ve alışveriş merkezi gibi kalabalık ve karmaşık iç mekânlarda güvenliğin sağlanmasında etkili bir şekilde kullanılabilir durumdadır. UWB teknolojisinin duvarlar ve fiziksel engellerden etkilenme oranının düşük olması, sistemin farklı ortam koşullarında da istikrarlı bir performans sergilemesine olanak tanımıştır.

Karşılaştırmalı analizler incelendiğinde, UWB tabanlı sistemin Wi-Fi ve Bluetooth temelli çözümlere göre çok daha yüksek doğruluk sunduğu ve sinyal parazitine karşı daha dayanıklı olduğu tespit edilmiştir. Öte yandan RFID tabanlı sistemlere kıyasla, UWB'nin daha geniş bir kapsama alanı sunması ve altyapı gereksinimlerinin daha az olması, sistemin kurulum ve ölçeklenebilirlik açısından avantaj sağladığını göstermiştir. Bu

yönleriyle geliştirilen sistem hem teknik yeterlilik hem de pratik fayda açısından güçlü bir alternatif olarak öne çıkmaktadır.

Sistemin doğruluğunu daha da artırmak amacıyla sabit alıcı sayısının artırılması önerilmektedir. Üç sabit alıcı ile gerçekleştirilen ölçümler, temel bir trilaterasyon hesaplaması için yeterli olmakla birlikte, alıcı sayısının artırılması ile sistem daha fazla referans noktası elde edecek ve bu sayede konum hesaplamaları daha hassas ve güvenilir bir şekilde yapılabilecektir.

Sistemin performansını daha geniş bir yelpazede değerlendirebilmek adına çok katlı binalarda testler yapılması faydalı olacaktır. Böylece sistemin yalnızca yatay düzlemde değil, aynı zamanda dikey düzlemde de ne ölçüde etkili çalıştığı analiz edilebilir. Bu durum, özellikle çok katlı hastaneler, alışveriş merkezleri ve oteller gibi yapılarda sistemin dikey eksende konumlandırma yeteneğini ölçmek açısından önem taşımaktadır.

Farklı büyüklükteki alanlarda ve çeşitli sinyal engellerinin bulunduğu ortamlarda test edilmesi de büyük önem arz etmektedir. Bu tür testler, sistemin genel geçerliliğini ve farklı koşullara adaptasyon yeteneğini ortaya koymak açısından faydalı olacaktır. Özellikle, kullanılan ESP32 UWB Pro modüllerinin sinyal yansımalarından daha az etkilenmesi adına, alıcı istasyonların sabitlenmesinde yükseklik ayarlı platformlar veya elektromanyetik yalıtım sağlayan yapılar tercih edilebilir. Bu sayede çevresel etkenlerin sinyal kalitesine olan olumsuz etkisi en aza indirilebilir. Mevcut sistem yalnızca üniversitenin kapalı tenis kortunda test edilmiştir; dolayısıyla farklı kapalı mekânlarda, örneğin alışveriş merkezleri, fabrikalar veya metal yapı içeren alanlarda da performans testlerinin yapılması, sistemin genellenebilirliğini artıracaktır.

Geliştirilen Python tabanlı masaüstü arayüz, kullanım kolaylığı açısından yeterli bir çözüm sunmakla birlikte, sistemin erişilebilirliğini artırmak ve saha uygulamalarına uygunluğunu geliştirmek amacıyla Android ve iOS tabanlı mobil uygulamaların da geliştirilmesi önerilmektedir. Bu sayede kullanıcılar konum verilerini mobil cihazlar üzerinden anlık olarak izleyebilecek ve sistem daha esnek bir kullanım alanına kavuşacaktır.

Hareketli vericinin enerji tüketimini optimize etmek, sistemin taşınabilirliğini ve

kullanım süresini doğrudan etkileyen önemli bir konudur. Pil ömrünü uzatmak ve sistemin sahada daha uzun süre çalışabilmesini sağlamak amacıyla düşük güç tüketimli yazılım modüllerinin tercih edilmesi, gerektiğinde cihazın otomatik olarak uyku moduna geçmesini sağlayan mekânizmaların uygulanması önerilmektedir.

Böylece sistemin yalnızca belirli türdeki kapalı alanlarda değil, çok çeşitli iç ve dış mekân senaryolarında da etkin çalıştığı ortaya konulabilecektir.

KAYNAKÇA

Akleylek, S., Kılıç, E., Söylemez, B., Aruk, E., T., ve Çavuş Yıldırım, A. (2020). Kapalı Mekân Konumlandırma Üzerine Bir Çalışma, *Mühendislik Bilimleri ve Tasarım Dergisi,* 8(5), 90-105. 18 Aralık 2024'te https://dergipark.org.tr/tr/pub/jesd/article/831775 adresinden indirildi.

Alarifi, A., Al-Salman, A., AlSaleh, M., Alnafessah, A., Al-Hadhrami, S., Al-Ammar, M. A., vd. (2016, 05 16). Ultra Wideband Indoor Positioning Technologies: Analysis and Recent Advances. *Sensors (Basel)*, *16*(5), 707. 18 Aralık 2024'te https://www.mdpi.com/1424-8220/16/5/707 adresinden indirildi.

Al-Hadhrami, S., Al-Salman, A., Al-Khalifa, H., Alarifi, A., Alnafessah, A., Alsaleh, M., vd. (2014, 11 16). Ultra Wideband Positioning: An Analytical Study of Emerging Technologies. *SENSORCOMM 2014: The Eighth International Conference on Sensor Technologies and Applications*, 66-74. 25 Aralık 2024'tehttps://personales.upv.es/thinkmind/SENSORCOMM/SENSORCOMM_2014/sensorcomm_2014_3_30_10103.html adresinden indirildi.

Alhomayani, F., (2021). *Deep Learning Methods for Fingerprint-Based Indoor and Outdoor Positioning* (Yayın no. 1887). [Yayınlanmış doktora tezi, University of Denver]. https://digitalcommons.du.edu/cgi/viewcontent.cgi?article=2879&context=etd.

Alışkan, İ. ve Şahin, A., (2023). Ultra Geniş Bant Teknolojisi ile Kapalı Alanlarda Konum Belirleme. *Karaelmas Fen ve Mühendislik Dergisi*, *13*(1), 179-190. 26 Kasım 2023'te https://dergipark.org.tr/tr/pub/karaelmasfen/issue/78699/1233265 adresinden indirildi.

Al-Okby, M. F. R., Junginger, S., Roddelkopf, T., ve Thurow, K. (2024). UWB-Based Real-Time Indoor Positioning Systems: A Comprehensive Review. *Applied*

Sciences, *14*(23), 11005. 18 Kasım 2024'te https://www.mdpi.com/2076-3417/14/23/11005 adresinden indirildi.

Ambrosio, M. (2021). *UWB anchors self-localization for indoor environments* (Yayın no. 19119). [Yüksek lisans tezi, Politecnico di Torino]. Webthesis. https://webthesis.biblio.polito.it/19119/1/tesi.pdf.

Anbar, Y. (2023). *Ultra-wideband technology and its applications for indoor positioning* (Yayın no. 1885177). [Yayınlanmış yüksek lisans tezi, Mälardalen University]. DiVA portal. https://www.diva-portal.org/smash/get/diva2:1885177/FULLTEXT01.pdf.

Arsan, T. (2018). Improvement of indoor positioning accuracy of ultra-wide band sensors by using big bang-big crunch optimization method. *Pamukkale Journal of Engineering Sciences*, (Research Article). 9 Temmuz 2025'te https://pajes.pau.edu.tr/en/jvi.aspx?pdir=pajes&plng=eng&un=PAJES-59365 adresinden indirildi.

Asaad, Safaar, M. ve Maghdid, Halgurd, S. (2022, 07 20). A Comprehensive Review of Indoor/Outdoor Localization Solutions in IoT era: Research Challenges and Future Perspectives. *Computer Networks*, Volume 212, 109041. 9 Temmuz 2025'tehttps://www.sciencedirect.com/science/article/pii/S1389128622001918 adresinden indirildi.

Bahl, P., ve Padmanabhan, V. N., (2000). RADAR: An in-building RF-based user location and tracking system*, Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No. 00CH37064)*, 775-784. 9 Temmuz 2025'te https://ieeexplore.ieee.org/document/832252 adresinden indirildi.

Başkan, P. (2025, Ocak 2). Round Trip Time (RTT) nedir ve nasıl ölçülür? *Sempeak*. 31 Temmuz 2025, https://www.sempeak.com/blog/round-trip-time-rtt-nedir-ve-nasil-olculur adresinden alındı.

Bernard, R. (2017). *Indoor Positioning Systems*. Security Industry Association. 4 Aralık 2024'te https://www.securityindustry.org/wp-content/uploads/2017/11/Indoor-Positioning-Systems.pdf adresinden indirildi.

Bilişim Evreni, (2023, 02 11). UDP-User Datagram Protocol. 4 Aralık 2024, https://bilisimevreni.com.tr/udp-user-datagram-protocol/ adresinden alındı.

Bozacı, İ., ve Beğdeş, R., N., (2019). Konum Tabanlı Mobil İletişimlerin Benimsenmesini Etkileyen Faktörler Üzerine Bir Araştırma. *İşletme Araştirmalari Dergisi*, *11*(3), 2084-2093. 28 Aralık 2024'te https://isarder.org/2019/vol.11_issue.3_article55_full_text.pdf adresinden indirildi.

Chen, H.-C., Lin, T.-H., Kung, H. T., Lin, C.-K., ve Gwon, Y. (2012). *A low complexity multipath mitigation approach to time-of-arrival estimation in UWB systems*. 31 Temmuz 2025'te https://www.eecs.harvard.edu/~htk/publication/2012-milcom-chen-lin-kung-lin-gwon.pdf adresinden indirildi.

Chen, W. (2021, 04 16). Personnel precise positioning system of coal mine underground based on UWB. Journal of Physics: *Conference Series, Volume 1920*, 012115. 6 Aralık 2024'te https://iopscience.iop.org/article/10.1088/1742-6596/1920/1/012115 adresinden indirildi.

Decawave Ltd. (2013). *DW1000 Product Brief*. Decawave Ltd. 9 Temmuz 2025, https://www.qorvo.com/products/d/da007945 adresinden alındı.

Decawave Ltd. (2017). *DW1000 Datasheet* (Version 2.22). Decawave Ltd. 9 Temmuz 2025, https://www.qorvo.com/products/d/da007946 adresinden alındı.

Demiral, E. Karaş İ. R. ve Turan, M. K. (2013). RFID Sistemleri ile Konum Belirleme Uygulamaları. *TMMOB Harita ve Kadastro Mühendisleri Odası, 14. Türkiye Harita Bilimsel ve Teknik Kurultayı*, Ankara. 23 Kasım 2023'te

https://web.karabuk.edu.tr/ismail.karas/files/4_%20Karas%20Demiral%20Turan.pdf adresinden indirildi.

Doughangi, H., (2017). *Kapalı Alanda Konum Belirleme Sistemi* (Yayın no. 457953). [Yayınlanmış yüksek lisans tezi, İstanbul Ticaret Üniversitesi]. https://acikbilim.yok.gov.tr/bitstream/handle/20.500.12812/95265/yokAcikBilim_10144364.pdf?sequence=-1&isAllowed=y.

Eroğlu, O. ve Doğan, M. (2019). Konum Tabanlı Hizmetler, Bluetooth Tabanlı İç Mekân Konumlandırma Hizmeti, Antalya İli Örneği. *TMMOB 6. Coğrafi Bilgi Sistemleri Kongresi*, Ankara. 23 Kasım 2023'te https://obs.hkmo.org.tr/show-media/resimler/ekler/8b5e9c543b7b378_ek.pdf adresinden indirildi.

Falsi, C., Dardari, D., Mucchi, L., ve Win, M. Z. (2006). Time of arrival estimation for UWB localizers in realistic environments. *EURASIP Journal on Advances in Signal Processing, 2006*(2), 032082. 23 Kasım 2023'te https://doi.org/10.1155/ASP/2006/32082 adresinden indirildi.

Fang, Y., Deng, Z., Xue, C., Jiao, J., Zeng, H., Zheng, R., vd. (2015), Application of an improved K nearest neighbor algorithm in WiFi indoor positioning. *China Satellite Navigation Conference (CSNC) 2015 Proceedings: Volume III*, 517-524. 23 Kasım 2023'te https://doi.org/10.1007/978-3-662-46632-2_45 adreainden indirildi.

Gökçen, A., ve Yeşil, B., (2022). Evaluation and Compensation of Temperature Effects on Ultrasonic Flow Measurement. *European Journal of Science and Technology*, (37), 119-122. 23 Kasım 2023'te https://dergipark.org.tr/en/download/article-file/2511260 adresinden indirildi.

Gu, Y., Lo, A., ve Niemegeers, I. (2009). A survey of indoor positioning systems for wireless personal networks. *IEEE Communications Surveys & Tutorials, 11*(1), 13-32. 5 Temmuz 2025'te https://doi.org/10.1109/SURV.2009.090103 adresinden indirildi.

Hapsari, G., I., Munadi, R., Erfianto, B., ve Irawati, I., D., (2024). Accuracy Improvement for Indoor Positioning Using Decawave on ESP32 UWB Pro with Display and Regression. *Journal of Robotics and Control (JRC) Volume 5*(3). 5 Temmuz 2025'te https://journal.umy.ac.id/index.php/jrc/article/view/20825/9134 adresinden indirildi.

He, J., Li, S., Pahlavan, K. ve Wang, Q., (2012). A realtime testbed for performance evaluation of indoor TOA location system, *2012 IEEE International Conference on Communications (ICC)*, 482-486. 5 Temmuz 2025'te https://ieeexplore.ieee.org/document/6364455 adresinden indirildi.

He, S., ve Chan, S.-H. G. (2016). Wi-Fi fingerprint-based indoor positioning: Recent advances and comparisons. *IEEE Communications Surveys & Tutorials, 18*(1), 466-490. 25 Kasım 2023'te https://doi.org/10.1109/COMST.2015.2464084 adresinden indirildi.

Ho, Y. H., ve Chan, H. C. B. (2020). Decentralized adaptive indoor positioning protocol using Bluetooth Low Energy. *Computer Communications, 158,* 148–160. 25 Kasım 2023'te https://doi.org/10.1016/j.comcom.2020.04.041 adresinden indirildi.

Jasani, M. (2023, 09 29). *How Does An Indoor Positioning System Work With Smart Devices?* 25 Kasım 2023'te https://www.excellentwebworld.com/indoor-positioning-system/ adresinden indirildi.

Kalite Network (2023, 07 12). *Wi-Fi (Wireless) Nedir?* 4 Aralık 2024, https://kalitenetwork.com/blog/wifi-wireless-nedir adresinden alındı.

Karaali, C., ve Yıldırım, Ö. (1996). GLOBAL KONUM BELİRLEME SİSTEMİ (GPS). *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi, 2*(2), 103–108. 4 Temmuz 2025'te https://dergipark.org.tr/tr/download/article-file/191588 adresinden indirildi.

Khan, M., Kai, Y. D. ve Gul, H. U., (2017). Indoor Wi-Fi positioning algorithm based on combination of Location Fingerprint and Unscented Kalman Filter, *2017 14th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, 693-698. 4 Temmuz 2025'te https://ieeexplore.ieee.org/document/7868127 adresinden indirildi.

Konukseven C. (2021). *Aktif Küresel Navigasyon Uydu Sistemleri ile Rtk Performansının İncelenmesi* (Yayın no. 671764). [Yayınlanmış doktora tezi, Necmettin Erbakan Üniversitesi]. https://acikerisim.erbakan.edu.tr/items/5b3fdf2b-8fa9-41e7-a99c-d9d2ecf46179.

Kurnaz, M., (2019). *UWB ile varış zamanı farkına dayalı konum kestirimi* (Yayın no. 604132). [Yayınlanmış yüksek lisans tezi, Gebze Teknik Üniversitesi]. https://tez.yok.gov.tr/UlusalTezMerkezi/TezGoster?key=aEzj_IdWAsjiSAfK3qwrBrCCLuXmGdtveHCNxi1k3QVUjv5STUYHnWXzzcqO80Nb.

Kwon, B., Yoo, H., Han, S., ve Jang, B. J., (2023). Implementation of High-speed Indoor Positioning System based on IEEE802.15.4z UWB. *Wireless Sensing Lab, Kookmin University, South Korea*. 4 Temmuz 2025'te https://ceur-ws.org/Vol-3581/182_WiP.pdf adresinden indirildi.

Li, L., Yang, W., Alam Bhuiyan, M. Z. ve Wang, G., (2016). Unsupervised learning of indoor localization based on received signal strength, *Wireless Communications and Mobile Computing, 16* (15), 2225-2237. 3 Ekim 2024'te https://onlinelibrary.wiley.com/doi/epdf/10.1002/wcm.2678 adresinden indirildi.

Li, P., Wu, W., Zhao, Z., ve Huang, G. Q. (2024). Indoor positioning systems in industry 4.0 applications: Current status, opportunities, and future trends. *Digital Twin and Embedded Systems, 1*, 100020. 3 Ekim 2024'te https://doi.org/10.1016/j.dte.2024.100020 adresinden indirildi.

Linktech, (2024). K552 Safe 3A Örgülü Metal Başlı Type-C Şarj Kablosu. 13 Ağustos 2024, https://linktech.com.tr/shop/k552-safe-fast-1m-orgulu-metal-basli-type-c-sarj-kablosu-27756#attr=16715,16714,16716,125 adresinden alındı.

Machaj, J., Brida, P. ve Piché, R., (2011). Rank based fingerprinting algorithm for indoor positioning, *2011 International Conference on Indoor Positioning and Indoor Navigation*, 1-6. 21 Aralık 2024'te https://wiki.makerfabs.com/ESP32_UWB.html adresinden indirildi.

Makerfabs Wiki, (2023). *ESP32 UWB Pro*. 21 Aralık 2024, https://wiki.makerfabs.com/ESP32_UWB.html adresinden alındı.

Masmanacı, M. N., (2018). *Bluetooth Beacon Temelli Kapalı Alan Mobil Konum Tespit ve Analiz Sistemi Geliştirilmesi* (Yayın no. 515849). [Yayınlanmış yüksek lisans tezi, Erciyes Üniversitesi]. https://tez.yok.gov.tr/UlusalTezMerkezi/tezDetay.jsp?id=kFiN4nZJEHkuo21y9laSyg&no=Q1F17LecWZuctYXKu56uJQ.

Medina, C., Segura, J. C. ve de La Torre A. (2013). Ultrasound Indoor Positioning System Based On A Low-Power Wireless Sensor Network Providing Sub-Centimeter Accuracy. *Sensors, 13*, 3501-3526. 22 Kasım 2023'te www.mdpi.com/journal/sensors adresinden indirildi.

Medium (2023, 11 09). *Tkinter Kütüphanesi Nedir?* 20 Aralık 2024, https://medium.com/@dizinbilgi/tkinter-k%C3%BCt%C3%BCphanesi-nedir-15288f131450 adresinden alındı.

Metageek, (2024). *Understanding RSSI*. 13 Ağustos 2024, https://www.metageek.com/training/resources/understanding-rssi/ adresinden alındı.

MiSERY. (2021, June 1). *ESP32 UWB Indoor Positioning Test*. Instructables. 9 Temmuz 2025, https://www.instructables.com/ESP32-UWB-Indoor-Positioning-Test/ adresinden alındı.

Mohd Sultan, J., Kamaruzaman, N. N., Chsudhary, A. R., Yusop, A. M., Manap, Z., ve Mohd Ali, D., (2024, 05 20). Precision Indoor Positioning with Ultra-Wideband (UWB) Technology. *Przegląd Elektrotechniczny*, *100*(5). 6 Aralık 2024'te.https://eprints.utem.edu.my/id/eprint/27527/2/Precision%20indoor%20positioning%20with%20ultrawideband%20%28UWB%29%20technology.pdf adresinden indirildi.

Nam Ho, T., H., Tran, V., Su, ve Nguyen, N., T., M., (2023). Design an Indoor Positioning System Using ESP32 Ultra-Wide Band Module. *ICIT 2023, LNDECT 187*, 382–393. 6 Aralık 2024'te https://www.researchgate.net/publication/374864904_Design_an_Indoor_Positioning_System_Using_ESP32_Ultra-Wide_Band_Module adresinden indirildi.

Niculescu, D., ve Nath, B. (2003). Ad hoc positioning system (APS) using AoA. *Proceedings of the IEEE INFOCOM 2003, 3*, 1734–1743. 10 Temmuz 2025'te https://doi.org/10.1109/INFCOM.2003.1209196 adresinden indirildi.

Özdemir, B. N., Ceylan, A., Alçay, S., ve Yiğit, C. Ö. (2014). Kapalı mekânlarda uygulanan konum belirleme yöntemleri ve karşılaştırılması. *7. Mühendislik Ölçmeleri Sempozyumu Bildiriler Kitabı*, 15–17 Ekim 2014, Hitit Üniversitesi, Çorum. 5 Temmuz 2025'te https://www.hkmo.org.tr/resimler/ekler/5ec7b25e5a5b4ee_ek.pdf adresinden indirildi.

Özdemir, B., N., (2020). *Kapalı Mekânlarda Rss Tabanlı Konum Belirleme Yaklaşımı: Parmak İzi Metodunun Uygulanması ve Performans Analizi* (Yayın no. 655200). [Yayınlanmış doktora tezi, Konya Teknik Üniversitesi]. https://gcris.ktun.edu.tr/handle/20.500.13091/1099.

Park, N., Jung, S. ve Han, D., 2015, An adaptive hybrid filter for practical WiFi-based positioning systems, *ICT Express, 1*(2), 82-85. 10 Temmuz 2025'te https://www.sciencedirect.com/science/article/pii/S2405959515000090 adresinden indirildi.

Pereira, F. J. L., (2016). *Positioning Systems for Underground Tunnel Environments* (Yayın no. 2215397). [Yayınlanmış doktora tezi, Porto University]. https://cds.cern.ch/record/2215397.

PttAVM.com, (2024). ZyXEL VMG1312-T20B Antenli Vdsl2/Adsl2+ 4 Port Kablosuz Modem (Kutulu-Yenilenmiş). 21 Aralık 2024, https://www.pttavm.com/zyxel-vmg1312-t20b-antenli-vdsl2adsl2--4-port-kablosuz-modem-kutulu-yenilenmis-p-1060203395?srsltid=Afm adresinden alındı.

Rizos, C., Dempster, A. G., Li, B. ve Salter, J., 2007, *Indoor positioning techniques based on wireless LAN*. 5 Temmuz 2025'te https://opus.lib.uts.edu.au/bitstream/10453/19580/1/113_Li.pdf adresinden indirildi.

Robodükkan.com, (2018). *ESP32 UWB Pro Long Distance Wrover*. 13 Ağustos 2024, https://www.robodukkan.com/urun/esp32-uwb-pro-long-distance-wrover-2572 adresinden alındı.

Rose, N. (2021, 05 26). *How ultrasonic local positioning system works*. 26 Kasım 2023, https://navigine.com/blog/how-ultrasonic-indoor-positioning-works/ adresinden alındı.

Satria, D., Zulfan, Munawir, ve Hidayat, T., (2018, 04 23). Implementation of wireless sensor network (WSN) on garbage transport warning information system using GSM module. *Journal of Physics: Conference Series, Volume 1175*, 012054. 5 Temmuz 2025'te https://doi.org/10.1088/1742-6596/1175/1/012054 adresinden indirildi.

Shrestha, S., Talvitie, J. ve Lohan, E. S., (2013a). Deconvolution-based indoor localization with WLAN signals and unknown access point locations, *2013 International Conference on Localization and GNSS (ICL-GNSS)*, 1-6. 9 Temmuz 2025'te https://ieeexplore.ieee.org/document/6577256/authors#authors adresinden indirildi.

Silberschatz, A., Galvin, P. B., ve Gagne, G. (2018). *Operating system concepts (10th ed.)*. Wiley. 10 Temmuz 2025'te https://os.ecci.ucr.ac.cr/slides/Abraham-Silberschatz-Operating-System-Concepts-10th-2018.pdf adresinden indirildi.

Syazwani, C., J., N., Abdul Wahab, N., H., Sunar, N., Ariffin, S. H. S., Wong, K., Y., ve Aun, Y., (2022). Indoor Positioning System: A Review. *International Journal of Advanced Computer Science and Applications, 13*(6). 9 Temmuz 2025'te https://thesai.org/Downloads/Volume13No6/Paper_59-Indoor_Positioning_System_A_Review.pdf adresinden indirildi.

Şahin, Y. (2011). *Ölçüm gürültüsünün karma geliş zamanı/geliş açısı yöntemine etkisinin incelenmesi* (Yayın no. 295395). [Yayınlanmış Yüksek lisans tezi, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü]. https://polen.itu.edu.tr:8443/server/api/core/bitstreams/8d878091-e14e-4e01-847f-fb34e3572a75/content.

Talvitie, J., Renfors, M. ve Lohan, E. S., (2015a). A comparison of received signal strength statistics between 2.4 GHz and 5 GHz bands for WLAN-based indoor positioning. *2015 IEEE globecom workshops (GC Wkshps)*, 1-6. 12 Temmuz 2025'te https://ieeexplore.ieee.org/document/7414038 adresinden indirildi.

Talvitie, J., Renfors, M. ve Lohan, E. S., (2015b). *Distance-based interpolation and extrapolation methods for RSS-based localization with indoor wireless signals*. *IEEE transactions on vehicular technology, 64*(4), 1340-1353. 12 Temmuz 2025'te https://ieeexplore.ieee.org/document/7024940 adresinden indirildi.

Tamer, Ö. (2023). RSSI ve ToA tabanlı mesafe tahminlerini kullanarak kablosuz sensör düğümlerinin bağıl lokalizasyonu. *DEÜ FMD, 25*(75), 647–658. 21 Aralık 2024'te https://doi.org/10.21205/deufmd.2023257511 adresinden indirildi.

TeknolojiKurdu.com, (2023). YD-32 4 Port 3.0 USB Çoğaltıcı Hub. 13 Ağustos 2024, https://www.teknolojikurdu.com/urun/torima-yd-32-usb-cogaltici adresinden alındı.

Teknosa, (2024). *Preo Pocket Size PS1 10.000 mAh Çift Usb Çıkışlı Powerbank Mavi*. 4 Aralık 2024, https://www.teknosa.com/preo-pocket-size-ps1-10000-mah-cift-usb-cikisli-powerbank-mavi-p-145059458 adresinden alındı.

Trendyol, (2024). *Preo Ps2 20000mah Powerbank Siyah*. 13 Ağustos 2024, https://www.trendyol.com/preo/ps2-20000mah-powerbank-siyah-p-94984261 adresinden alındı.

Varlı, H., ve Bingöl, N., (2024, 04 25). Konum Tabanlı Yeni Yöntem ile Acil Ulaşım Sorunlarına Karar Destek Sistemleri: Ambulans Örnek Uygulama Canlandırması (Çam ve Sakura Hastanesi Örneği). *Bilişim Teknolojileri Dergisi, 17*(2). 13 Ağustos 2024'te https://dergipark.org.tr/tr/pub/gazibtd/issue/84331/1431518?utm_source=chatgpt.com adresinden indirildi.

Venuex (2018, 03 08). *İç Mekân Konumlandırma ve Haritalama Nedir?* 25 Kasım 2023, https://medium.com/@venuex.io/i%CC%87%C3%A7-mekan-konumland%C4%B1rma-ve-haritalama-nedir-cdc0fe471180 adresinden alındı.

Von Tschirschnitz, M., Wagner, M., Pahl, M. O., ve Carle, G. (2019). Clock error analysis of common time of flight based positioning methods. *Proceedings of the 2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN 2019)*, Article 8911772. Institute of Electrical and Electronics Engineers Inc. 23 Haziran 2024'te https://doi.org/10.1109/IPIN.2019.8911772 adresinden indirildi.

Wu, P., (2021, 11 05). Comparison between the Ultra-wide Band based indoor positioning technology and other technologies. *Journal of Physics: Conference Series, 2187*, 012010. 9 Temmuz 2025'te https://iopscience.iop.org/article/10.1088/1742-6596/2187/1/012010/pdf adresinden indirildi.

Wu, Z., Jedari, E., Muscedere, R. ve Rashidzadeh, R., 2016, Improved particle filter based on WLAN RSSI fingerprinting and smart sensors for indoor localization,

Computer Communications, 83, 64-71. 9 Temmuz 2025'te https://doi.org/10.1016/j.comcom.2016.03.001 adresinden indirildi.

Xia, S., Liu, Y., Yuan, G., Zhu, M., ve Wang, Z. (2017). Indoor fingerprint positioning based on Wi-Fi: An overview. *ISPRS International Journal of Geo-Information, 6*(5), 135. 4 Temmuz 2025'te https://doi.org/10.3390/ijgi6050135 adresinden indirildi.

Yassin, A., ve Youssef, M. (2009). Challenges: A robust indoor localization technique. *Proceedings of the 2009 IEEE International Conference on Communications, ICC 2009*, 1–6. 4 Temmuz 2025'te https://doi.org/10.1109/ICC.2009.5198892 adresinden indirildi.

Ziegler, M., Kianfar, A. E., Hartmann, T., ve Clausen, E., (2023, 07 07). Development and Evaluation of a UWB-Based Indoor Positioning System for Underground Mine Environments. *Mining, Metallurgy & Exploration, 40*, 1021–1040. 14 Aralık 2024'te https://link.springer.com/article/10.1007/s42461-023-00797-z adresinden indirildi.

EKLER

EK 1: Sabit alıcı istasyon kodları

Birinci Sabit Alıcı İstasyonu (IS1), ultra geniş bant (UWB) teknolojisine dayalı Decawave DW1000 entegre devresini kullanarak 6.8 GHz merkez frekansında ve 500 MHz bant genişliğinde çalışmaktadır. Sistem, gerçekleştirdiği mesafe ölçümlerinde ±10 cm'lik yüksek doğruluk seviyesine ulaşmaktadır. Ölçülen mesafe verileri, 100 Hz'lik sabit örnekleme frekansıyla gerçek zamanlı olarak toplanmakta ve IEEE 802.15.4 standart çerçeve yapısına uygun şekilde JSON formatında paketlenmektedir. Paketlenen veriler, düşük gecikmeli ve yüksek hızlı veri iletimi için optimize edilmiş UDP/IP protokolü üzerinden, ağ yapısında bulunan hareketli vericiye iletilmektedir. Bu iletişim sürecinde, veri bütünlüğünü sağlamak amacıyla 16-bit CRC hata kontrol mekânizması uygulanmaktadır.

```
/*
ESP32 UWB - Sabit İstasyon (IS1)
*/

#include <SPI.h>
#include "DW1000Ranging.h"

#define ANCHOR_ADD "7D:00:22:EA:82:60:3B:9C" // IS1 MAC ID

#define SPI_SCK 18
#define SPI_MISO 19
#define SPI_MOSI 23
#define DW_CS 4

const uint8_t PIN_RST = 27;
const uint8_t PIN_IRQ = 34;
const uint8_t PIN_SS = 4;

float OLÇÜM = 0;
bool aktif = false;

void setup()
{
   Serial.begin(115200);
   delay(1000);
```

```
    // SPI başlatma
    SPI.begin(SPI_SCK, SPI_MISO, SPI_MOSI);
    DW1000Ranging.initCommunication(PIN_RST, PIN_SS, PIN_IRQ);

    // Callback fonksiyonları
    DW1000Ranging.attachNewRange(yeniMesafe);
    DW1000Ranging.attachNewDevice(yeniCihaz);
    DW1000Ranging.attachInactiveDevice(pasifCihaz);

    // Sabit istasyon olarak başlat
    DW1000Ranging.startAsAnchor(ANCHOR_ADD,
DW1000.MODE_LONGDATA_RANGE_LOWPOWER, false);
}

void loop()
{
    DW1000Ranging.loop();

    // Vericiden gelen sinyal hedefin kim olduğunu içeriyor olmalı
    if (Serial.available()) {
        String gelen = Serial.readStringUntil('\n');
        if (gelen.indexOf(ANCHOR_ADD) != -1) {
            aktif = true;
            Serial.println("Sıra bende, veri alınacak.");
        } else {
            aktif = false;
        }
    }
}

void yeniMesafe()
{
    if (!aktif) return;

    OLÇÜM = ((DW1000Ranging.getDistantDevice()->getRange()) - 1.0);
    if (OLÇÜM > 0.1)
    {
        Serial.print("IS1: ");
        Serial.print(OLÇÜM);
        Serial.println(" m");
    }
}

void yeniCihaz(DW1000Device *cihaz)
{
    Serial.print("Yeni cihaz eklendi -> ");
    Serial.print(" Kısa Adres:");
```

```
    Serial.println(cihaz->getShortAddress(), HEX);
}

void pasifCihaz(DW1000Device *cihaz)
{
    Serial.print("Pasif cihaz silindi: ");
    Serial.println(cihaz->getShortAddress(), HEX);
}
```

İkinci Sabit Alıcı İstasyonu (IS2), ultra geniş bant (UWB) teknolojisini temel alan Decawave DW1000 entegre devresini kullanarak 6.8 GHz merkez frekansında ve 500 MHz bant genişliğinde çalışmaktadır. Sistem, yapılan mesafe ölçümlerinde ±10 cm'lik bir doğruluk sağlamaktadır. Ölçülen mesafe verileri, 100 Hz'lik sabit örnekleme frekansıyla JSON formatında yapılandırılmakta ve IEEE 802.15.4 standart çerçeve yapısı kullanılarak UDP/IP protokolü üzerinden hareketli vericiye gerçek zamanlı olarak aktarılmaktadır.

```
/*
ESP32 UWB - Sabit İstasyon (IS2)
*/

#include <SPI.h>
#include "DW1000Ranging.h"

#define ANCHOR_ADD "7D:00:22:EA:82:60:3B:9A" // IS2 MAC ID
```

Üçüncü Sabit Alıcı İstasyonu (IS3), Decawave DW1000 UWB çipini kullanarak 6.8 GHz merkez frekansında çalışır. ±10 cm doğrulukla ölçülen mesafe verileri, 100 Hz örnekleme hızında JSON formatında paketlenir ve UDP protokolü üzerinden hareketli vericiye gerçek zamanlı olarak iletilir.

```
/*
ESP32 UWB - Sabit İstasyon (IS3)
*/
#include <SPI.h>
#include "DW1000Ranging.h"

#define ANCHOR_ADD "7D:00:22:EA:82:60:3B:9B" // IS3 MAC ID
```

EK 2: Hareketli verici kodu

Hareketli UWB verici cihazının (tag) ESP32 üzerinde çalışan ana kontrol yazılımıdır. DECAWAVE DW1000 çipiyle 6.5 GHz UWB frekans bandında çalışarak, sabit alıcı istasyonlarıyla (IS1-IS2-IS3) çift yönlü mesafe ölçümü yapar. Mesafe verilerini ±5 cm doğrulukta ölçüm yapabilmektedir. Ölçülen mesafe verilerini 100 Hz örnekleme hızıyla UDP protokolü üzerinden Python tabanlı merkezi arayüz sistemine JSON formatında iletir.

```
/*
ESP32 UWB - Hareketli Verici (IS1, IS2, IS3 ile iletişim)
Güncellenmiş Sürüm:
- MAC adresi bildirimi eklendi
- Konum bildirimi alım desteği eklendi
- Round Robin optimizasyonu
*/

#include <SPI.h>
#include "DW1000Ranging.h"
#include <WiFi.h>
#include <WiFiUdp.h>

#define TAG_ADD "86:17:5B:D5:A9:9A:E2:9C" // Verici MAC ID

// Sabit istasyon MAC adresleri
const char* SABIT_ISTASYON_MAC[] = {
  "7D:00:22:EA:82:60:3B:9C", // IS1 MAC
  "7D:00:22:EA:82:60:3B:9A", // IS2 MAC
  "7D:00:22:EA:82:60:3B:9B"  // IS3 MAC
};

// Ağ ayarları
const char* ssid = "BatuhanCep";
const char* password = "gozo4482";

IPAddress local_IP(192, 168, 1, 33);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 255, 0);
IPAddress primaryDNS(8, 8, 8, 8);
IPAddress secondaryDNS(8, 8, 4, 4);

// UDP ayarları
WiFiUDP UDP;
const int udpPort = 1234;
```

```
IPAddress broadcastIP(255, 255, 255, 255);

// Durum makinesi
enum IstasyonDurumu { HAZIR, IS1_OLCUM, IS2_OLCUM, IS3_OLCUM };
IstasyonDurumu suankiDurum = HAZIR;
unsigned long durumBaslangicZamani = 0;
const long durumZamanAsimi = 1000; // ms

// Ölçüm verileri
float mesafeVerileri[3] = {0.0, 0.0, 0.0};
float gucVerileri[3] = {0.0, 0.0, 0.0};

// Konum bildirimi
bool konumBildirimiAktif = false;
unsigned long sonBildirimZamani = 0;
const long bildirimAraligi = 10000; // 10 saniyede bir

// SPI pin tanımları
#define SPI_SCK 18
#define SPI_MISO 19
#define SPI_MOSI 23
#define DW_CS 4

const uint8_t PIN_RST = 27;
const uint8_t PIN_IRQ = 34;
const uint8_t PIN_SS = 4;

void setup() {
  Serial.begin(115200);
  delay(1000);

  // WiFi başlatma
  if (!WiFi.config(local_IP, gateway, subnet, primaryDNS, secondaryDNS)) {
    Serial.println("IP yapılandırma başarısız!");
  }

  WiFi.begin(ssid, password);
  Serial.print("WiFi'ye bağlanıyor...");
  unsigned long wifiBaslangic = millis();
  while (WiFi.status() != WL_CONNECTED && millis() - wifiBaslangic < 10000) {
    delay(500);
    Serial.print(".");
  }

  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("\nWiFi bağlantısı başarısız!");
    ESP.restart();
  }
```

```
    Serial.println("\nWiFi bağlandı!");
    Serial.print("Sabit IP Adresi: ");
    Serial.println(WiFi.localIP());

    // Tek port açılıyor
    if (UDP.begin(udpPort)) {
        Serial.print("UDP portu açıldı: ");
        Serial.println(udpPort);
    } else {
        Serial.println("UDP portu açılamadı!");
    }

    // UWB başlatma
    SPI.begin(SPI_SCK, SPI_MISO, SPI_MOSI);
    DW1000Ranging.initCommunication(PIN_RST, PIN_SS, PIN_IRQ);

    DW1000Ranging.attachNewRange(yeniMesafe);
    DW1000Ranging.attachBlinkDevice(yeniCihaz);
    DW1000Ranging.attachInactiveDevice(pasifCihaz);

    DW1000Ranging.startAsTag(TAG_ADD,
DW1000.MODE_LONGDATA_RANGE_LOWPOWER);

    // İlk durum
    suankiDurum = IS1_OLCUM;
    durumBaslangicZamani = millis();
    Serial.println("Round Robin başlatıldı - IS1 ile iletişim kuruluyor...");
}

void loop() {
    DW1000Ranging.loop();
    bildirimleriDinle();

    switch (suankiDurum) {
        case IS1_OLCUM:
        case IS2_OLCUM:
        case IS3_OLCUM:
            if (millis() - durumBaslangicZamani > durumZamanAsimi) {
                sonrakiIstasyonaGec();
            }
            break;
        case HAZIR:
            break;
    }

    // Konum bildirimi aktifse düzenli olarak gönder
    if (konumBildirimiAktif && millis() - sonBildirimZamani > bildirimAraligi) {
```

```
        verileriGonder(true);
        sonBildirimZamani = millis();
    }
}

void bildirimleriDinle() {
    int packetSize = UDP.parsePacket();
    if (packetSize) {
        char packetBuffer[255];
        int len = UDP.read(packetBuffer, 255);
        if (len > 0) {
            packetBuffer[len] = 0;
        }

        String mesaj = String(packetBuffer);
        Serial.print("Alınan bildirim: ");
        Serial.println(mesaj);

        if (mesaj.indexOf("NOTIFY:ON") >= 0) {
            konumBildirimiAktif = true;
            Serial.println("Konum bildirimi AKTİF");
        }
        else if (mesaj.indexOf("NOTIFY:OFF") >= 0) {
            konumBildirimiAktif = false;
            Serial.println("Konum bildirimi PASİF");
        }
    }
}

void sonrakiIstasyonaGec() {
    switch (suankiDurum) {
        case IS1_OLCUM:
            suankiDurum = IS2_OLCUM;
            Serial.println("IS1 ölçümü tamamlandı, IS2'ye geçiliyor...");
            break;

        case IS2_OLCUM:
            suankiDurum = IS3_OLCUM;
            Serial.println("IS2 ölçümü tamamlandı, IS3'e geçiliyor...");
            break;

        case IS3_OLCUM:
            suankiDurum = IS1_OLCUM;
            Serial.println("IS3 ölçümü tamamlandı, IS1'e geçiliyor...");
            verileriGonder(false);
           break;

        case HAZIR:
```

```
            suankiDurum = IS1_OLCUM;
            break;
    }

    durumBaslangicZamani = millis();
}

void verileriGonder(bool bildirimModu) {
    String veri = "MAC:" + String(TAG_ADD) +
                  ",IS1:" + String(mesafeVerileri[0], 2) +
                  ",IS2:" + String(mesafeVerileri[1], 2) +
                  ",IS3:" + String(mesafeVerileri[2], 2);

    if (bildirimModu) {
        veri += ",NOTIFY:ACK";
    }

    UDP.beginPacket(broadcastIP, udpPort);
    UDP.write((const uint8_t*)veri.c_str(), veri.length());
    UDP.endPacket();

    Serial.println("Veriler gönderildi: " + veri);
}

void yeniMesafe() {
    DW1000Device* uzakCihaz = DW1000Ranging.getDistantDevice();
    uint16_t suankiIstasyon = uzakCihaz->getShortAddress();

    switch (suankiDurum) {
        case IS1_OLCUM:
            if (suankiIstasyon == 0x3B9C) {
                mesafeVerileri[0] = uzakCihaz->getRange() - 1.0;
                gucVerileri[0] = uzakCihaz->getRXPower();
                istasyonBilgisiYazdir(1, mesafeVerileri[0], gucVerileri[0]);
            }
            break;

        case IS2_OLCUM:
            if (suankiIstasyon == 0x3B9A) {
                mesafeVerileri[1] = uzakCihaz->getRange() - 1.0;
                gucVerileri[1] = uzakCihaz->getRXPower();
                istasyonBilgisiYazdir(2, mesafeVerileri[1], gucVerileri[1]);
            }
            break;

        case IS3_OLCUM:
            if (suankiIstasyon == 0x3B9B) {
                mesafeVerileri[2] = uzakCihaz->getRange() - 1.0;
```

```
            gucVerileri[2] = uzakCihaz->getRXPower();
            istasyonBilgisiYazdir(3, mesafeVerileri[2], gucVerileri[2]);
          }
          break;

        case HAZIR:
          break;
    }
}

void istasyonBilgisiYazdir(int istasyonNo, float mesafe, float guc) {
    Serial.print("IS");
    Serial.print(istasyonNo);
    Serial.print(" - Mesafe: ");
    Serial.print(mesafe);
    Serial.print(" m, Güç: ");
    Serial.print(guc);
    Serial.println(" dBm");
}

void yeniCihaz(DW1000Device *cihaz) {
    Serial.print("Yeni cihaz eklendi: ");
    Serial.println(cihaz->getShortAddress(), HEX);
}

void pasifCihaz(DW1000Device *cihaz) {
    Serial.print("Cihaz bağlantısı kesildi: ");
    Serial.println(cihaz->getShortAddress(), HEX);
}
```

EK 3: Arayüz yazılımı

ESP32 UWB alıcılarından gelen real-time mesafe verilerinin görselleştirilmesi ve nesne konumunun kolayca izlenebilmesi için bir arayüz programı geliştirilmiştir. Programın temel özellikleri arasında veri işleme (UDP protokolü üzerinden ESP32 alıcılarından (IS1, IS2, IS3) gelen JSON formatlı mesafe verilerini (±10 cm doğrulukla) işleme ve trilaterasyon algoritmasıyla 2B konum hesaplaması yapma (2000x1732.05 cm çalışma alanı)), görselleştirme (CustomTkinter tabanlı dinamik arayüzde real-time konum takibi, mesafe çizgileri (r1, r2, r3) ve koordinat eksenleriyle etkileşimli harita, yeşil marker ile hareketli vericinin anlık pozisyonu) ve sistem yönetimi (cihaz seçimi ve konfigürasyonu (MAC adresi tanımlama), alıcı istasyonların konum ayarlarını yönetme, verileri SQLite veritabanına kaydetme ve CSV export) bulunmaktadır.

Teknik altyapı olarak CustomTkinter (Modern Tkinter) GUI çerçevesi, Matplotlib (arka planda) ve PIL (resim işleme) grafik kütüphaneleri, asenkron UDP soketleri (100 Hz örnekleme), JSON formatlı paketler ve Pandas DataFrame veri yapıları kullanılmıştır. Uygulama Windows/Linux/macOS platformlarında uyumludur.

Kod yapısı, App(customtkinter.CTk) sınıfı üzerinden arayüz bileşenlerini (create_widgets() ile kontrol paneli ve harita alanı), konum hesaplama fonksiyonlarını (trilateration_raw()) ve gerçek zamanlı güncelleme mekânizmalarını (update_marker_position()) içerir. Veri akışı, ESP32 alıcılarından UDP/JSON formatında gelen verilerin (örnek: {"IS1": 150.2, "IS2": 210.5, "IS3": 180.3}) trilaterasyon ile (x,y) koordinatlarına dönüştürülmesi, GUI'de marker ve çizgilerin güncellenmesi ve SQLite'a kaydedilmesi şeklinde işler.

Kritik fonksiyonlar arasında UDP veri alımını yöneten fetch_udp(), veri işleme thread'ini çalıştıran handle_udp_data(), 3 noktalı konum hesaplama yapan trilateration_raw() ve grafik güncellemelerini sağlayan update_marker_position() bulunur.

```
import customtkinter
import json
import os
import threading
import socket
import time
```

```
from PIL import Image, ImageTk
import pandas as pd

UDP_IP = "0.0.0.0"  # Tüm ağ arayüzlerini dinle
PORT = 1234  # Veri alışverişi için kullanılacak port

customtkinter.set_default_color_theme("blue")  # Mavi tema kullan

class App(customtkinter.CTk):
    APP_NAME = "ESP32 UWB Takip Sistemi"  # Uygulama adı (Türkçe)
    WIDTH = 1000  # Pencere genişliği
    HEIGHT = 500  # Pencere yüksekliği

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

        # Temel pencere ayarları
        self.title(App.APP_NAME)
        self.geometry(f"{App.WIDTH}x{App.HEIGHT}")
        self.minsize(App.WIDTH, App.HEIGHT)
        self.protocol("WM_DELETE_WINDOW", self.on_closing)

        # Başlangıç değişkenleri
        self.target_device_mac = None  # Hedef cihaz MAC adresi
        self.target_device_name = "Cihaz-1"  # Varsayılan cihaz adı
        self.available_devices = {}  # Tespit edilen cihazlar
        self.background_image_path = "arkaplan.png"  # Arkaplan resmi yolu
        self.locations_file = "locations.json"  # Konum ayarları dosyası
        self.device_data = {"IS1": 0, "IS2": 0, "IS3": 0}  # Mesafe verileri
        self.is_data = {"IS1": None, "IS2": None, "IS3": None}  # İstasyon verileri
        self.locations_data = None  # Konum ayarları
        self.current_station_index = 0  # Round Robin için mevcut istasyon indeksi
        self.station_order = ["IS1", "IS2", "IS3"]  # İstasyon sırası
        self.last_update_time = time.time()  # Son güncelleme zamanı
        self.station_timeout = 1.0  # İstasyon zaman aşımı (saniye)
        self.distance_labels = {}  # Mesafe etiketleri
        self.coord_label_widget = None  # Koordinat etiketi
        self.axis_lines = []  # Eksen çizgileri

        # Arayüzü oluştur
        self.create_widgets()
        self.load_default_locations()  # Varsayılan konumları yükle
        self.create_green_box()  # Yeşil işaretçi kutusu oluştur
        self.empty_area.bind("<Configure>", lambda event:
self.load_background_image())

        # Arka plan iş parçacıklarını başlat
        self.thread_start = threading.Thread(target=self.fetch_udp)
```

```
        self.thread_start.daemon = True
        self.thread_start.start()

        self.thread_handle = threading.Thread(target=self.handle_udp_data)
        self.thread_handle.daemon = True
        self.thread_handle.start()

    def create_widgets(self):
        """Arayüz bileşenlerini oluşturur"""
        # Izgara düzeni ayarları
        self.grid_columnconfigure(0, weight=0)
        self.grid_columnconfigure(1, weight=1)
        self.grid_rowconfigure(0, weight=1)

        # Sol panel (kontrol butonları ve bilgiler)
        self.frame_left = customtkinter.CTkFrame(master=self, width=150,
corner_radius=0, fg_color=None)
        self.frame_left.grid(row=0, column=0, padx=0, pady=0, sticky="nsew")
        self.frame_left.grid_rowconfigure(9, weight=1)

        # Cihaz bilgi ve kontrol widget'ları
        self.device_label = customtkinter.CTkLabel(
            master=self.frame_left,
            text=f"Takip Edilen: {self.target_device_name}",  # Türkçe etiket
            font=("Arial", 12, "bold")
        )
        self.device_label.grid(pady=(10, 0), padx=(20, 20), row=0, column=0)

        self.device_select_button = customtkinter.CTkButton(
            master=self.frame_left,
            text="Cihaz Seç",  # Türkçe buton metni
            command=self.open_device_selection
        )
        self.device_select_button.grid(pady=(10, 0), padx=(20, 20), row=1, column=0)

        self.notify_button = customtkinter.CTkButton(
            master=self.frame_left,
            text="Konumu Bildir",  # Türkçe buton metni
            command=self.send_notification,
            fg_color="green"
        )
        self.notify_button.grid(pady=(10, 0), padx=(20, 20), row=2, column=0)

        # Diğer kontrol butonları
        self.update_button = customtkinter.CTkButton(
            master=self.frame_left,
            text="Alıcı İstasyonları Güncelle",  # Türkçe buton metni
            command=self.load_default_locations
```

```
        )
        self.update_button.grid(pady=(20, 0), padx=(20, 20), row=3, column=0)

        self.settings_button = customtkinter.CTkButton(
            master=self.frame_left,
            text="Ayarlar",  # Türkçe buton metni
            command=self.open_settings
        )
        self.settings_button.grid(pady=(20, 0), padx=(20, 20), row=4, column=0)

        # Mesafe bilgisi etiketleri
        for i, device_id in enumerate(["IS1", "IS2", "IS3"]):
            self.distance_labels[device_id] = customtkinter.CTkLabel(
                master=self.frame_left,
                text=f"r{i+1}: 0.00 cm",  # Mesafe bilgisi (cm cinsinden)
                font=("Arial", 12)
            )
            self.distance_labels[device_id].grid(
                pady=(10, 0),
                padx=(20, 20),
                row=5+i,
                column=0,
                sticky="w"
            )

        # Koordinat bilgisi etiketi
        self.coord_label_widget = customtkinter.CTkLabel(
            master=self.frame_left,
            text="(x: 0.00, y: 0.00)",  # Koordinat bilgisi
            font=("Arial", 12, "bold")
        )
        self.coord_label_widget.grid(
            pady=(20, 0),
            padx=(20, 20),
            row=8,
            column=0,
            sticky="w"
        )

        # Sağ panel (harita görüntüleme alanı)
        self.frame_right = customtkinter.CTkFrame(master=self, corner_radius=0)
        self.frame_right.grid(row=0, column=1, rowspan=1, pady=0, padx=0,
sticky="nsew")
        self.frame_right.grid_rowconfigure(1, weight=1)
        self.frame_right.grid_columnconfigure(0, weight=1)

        # Çizim alanı
```

```
        self.empty_area = customtkinter.CTkFrame(self.frame_right, corner_radius=0,
fg_color="gray")
        self.empty_area.grid(row=1, column=0, sticky="nswe")

        # Canvas oluştur
        self.canvas = customtkinter.CTkCanvas(self.empty_area, width=App.WIDTH,
height=App.HEIGHT, bg="gray")
        self.canvas.place(x=0, y=0, relwidth=1, relheight=1)

    def open_device_selection(self):
        """Cihaz seçim penceresini açar"""
        selection_window = customtkinter.CTkToplevel(self)
        selection_window.title("Cihaz Seçimi")  # Türkçe pencere başlığı
        selection_window.geometry("400x300")

        devices_frame = customtkinter.CTkFrame(selection_window)
        devices_frame.pack(pady=10, padx=10, fill="both", expand=True)

        # Cihaz adı düzenleme alanı
        name_frame = customtkinter.CTkFrame(selection_window)
        name_frame.pack(pady=10, padx=10, fill="x")

        customtkinter.CTkLabel(name_frame, text="Cihaz Adı:").pack(side="left",
padx=5)  # Türkçe etiket
        name_entry = customtkinter.CTkEntry(name_frame)
        name_entry.insert(0, self.target_device_name)
        name_entry.pack(side="left", padx=5, fill="x", expand=True)

        def save_selection():
            """Seçimi kaydeder ve pencereyi kapatır"""
            self.target_device_name = name_entry.get()
            self.device_label.configure(text=f"Takip Edilen: {self.target_device_name}")  #
Türkçe etiket
            selection_window.destroy()

        save_button = customtkinter.CTkButton(
            selection_window,
            text="Kaydet",  # Türkçe buton metni
            command=save_selection
        )
        save_button.pack(pady=10)

    def send_notification(self):
        """Seçili cihaza konum bildirimi gönderir"""
        if self.target_device_mac:
            print(f"Bildirim gönderildi: {self.target_device_name} (MAC:
{self.target_device_mac})")  # Türkçe çıktı
            self.show_notification_popup()
```

```
        else:
            print("Hata: Hedef cihaz MAC adresi belirtilmemiş")  # Türkçe hata mesajı

    def show_notification_popup(self):
        """Bildirim onay penceresini gösterir"""
        popup = customtkinter.CTkToplevel(self)
        popup.title("Bildirim Gönderildi")  # Türkçe pencere başlığı

        msg = f"{self.target_device_name} cihazına konum bildirimi gönderildi"  # Türkçe
mesaj
        label = customtkinter.CTkLabel(popup, text=msg)
        label.pack(pady=20, padx=20)

        popup.after(3000, popup.destroy)  # 3 saniye sonra otomatik kapanır

    def fetch_udp(self):
        """UDP paketlerini dinler ve işler"""
        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        sock.bind((UDP_IP, PORT))
        sock.settimeout(0.1)

        if not hasattr(self, 'last_received_time'):
            self.last_received_time = {"IS1": 0, "IS2": 0, "IS3": 0}

        while True:
            try:
                data, _ = sock.recvfrom(512)
                message = data.decode('utf-8').strip()
                if not message:
                    continue

                print(f"\nAlınan ham veri: {message}")  # Türkçe çıktı

                # MAC adresi tespiti
                if "MAC:" in message:
                    mac_start = message.find("MAC:") + 4
                    mac_end = message.find(",", mac_start)
                    mac = message[mac_start:mac_end] if mac_end != -1 else
message[mac_start:]

                    self.available_devices[mac] = {
                        'name': f"Cihaz-{len(self.available_devices)+1}",  # Türkçe önek
                        'last_seen': time.time()
                    }

                    if not self.target_device_mac:
                        self.target_device_mac = mac
                        self.target_device_name = self.available_devices[mac]['name']
```

```
                    self.device_label.configure(text=f"Takip Edilen:
{self.target_device_name}")  # Türkçe etiket

                updated = False
                for part in message.split(","):
                    for key in ["IS1", "IS2", "IS3"]:
                        if key in part:
                            try:
                                distance = float(part.split(":")[1]) * 100  # cm'ye çevir
                                if distance < 20 or distance > 1000:  # Geçersiz aralık kontrolü
                                    continue

                                if distance != self.device_data[key]:
                                    self.device_data[key] = distance
                                    self.last_received_time[key] = time.time()
                                    updated = True

                            except Exception as e:
                                print(f"{key} verisi ayrıştırma hatası: {e}")  # Türkçe hata mesajı

                if updated:
                    self.last_update_time = time.time()
                    self.update_marker_position()

            except socket.timeout:
                current_time = time.time()
                if current_time - self.last_update_time > self.station_timeout:
                    self.current_station_index = (self.current_station_index + 1) %
len(self.station_order)
                    print(f"\nZaman aşımı - Yeni istasyon:
{self.station_order[self.current_station_index]}")  # Türkçe çıktı
                    self.last_update_time = current_time
            except Exception as e:
                print(f"\nUDP hatası: {e}")  # Türkçe hata mesajı

    def handle_udp_data(self):
        """UDP verilerini işler ve konsola yazdırır"""
        last_values = {"IS1": None, "IS2": None, "IS3": None}

        while True:
            try:
                if (self.device_data["IS1"] != last_values["IS1"] or
                    self.device_data["IS2"] != last_values["IS2"] or
                    self.device_data["IS3"] != last_values["IS3"]):

                    print("\n--- Güncel Veriler ---")  # Türkçe başlık
                    print("Ham mesafeler (cm):")  # Türkçe etiket
                    for device_id, value in self.device_data.items():
```

```
                print(f"{device_id}: {value:.2f}")
                last_values[device_id] = value

            print("="*30)

        time.sleep(0.1)

    except Exception as e:
        print(f"\nVeri işleme hatası: {e}")  # Türkçe hata mesajı
        time.sleep(0.5)

def trilateration_raw(self, r1, r2, r3):
    """Üç sabit noktadan mesafe ölçümüyle konum belirleme"""
    try:
        r1, r2, r3 = abs(r1), abs(r2), abs(r3)
        x1, y1 = 0, 0             # IS1 (orijin noktası)
        x2, y2 = 400, 0           # IS2 (X ekseni üzerinde)
        x3, y3 = 200, 346.41      # IS3 (Y ekseni referansı)

        # Trilaterasyon formülleri
        A = 2 * (x2 - x1)
        B = 2 * (y2 - y1)
        C = r1**2 - r2**2 - x1**2 + x2**2 - y1**2 + y2**2
        D = 2 * (x3 - x2)
        E = 2 * (y3 - y2)
        F = r2**2 - r3**2 - x2**2 + x3**2 - y2**2 + y3**2

        x = (C * E - F * B) / (A * E - B * D)
        y = (C * D - A * F) / (B * D - A * E)

        # Sınır kontrolleri
        x = max(0, min(x, 400))
        y = max(0, min(y, 346.41))

        return x, y

    except Exception as e:
        print(f"Trilaterasyon hatası: {e}")  # Türkçe hata mesajı
        return self.empty_area.winfo_width()/2, self.empty_area.winfo_height()/2

def update_marker_position(self):
    """Hareketli vericinin konumunu günceller"""
    d1_cm = self.device_data["IS1"]
    d2_cm = self.device_data["IS2"]
    d3_cm = self.device_data["IS3"]

    # Konum hesapla
    x_cm, y_cm = self.trilateration_raw(d1_cm, d2_cm, d3_cm)
```

```
# Mesafe etiketlerini güncelle
self.distance_labels["IS1"].configure(text=f"r1: {d1_cm:.2f} cm")
self.distance_labels["IS2"].configure(text=f"r2: {d2_cm:.2f} cm")
self.distance_labels["IS3"].configure(text=f"r3: {d3_cm:.2f} cm")
self.coord_label_widget.configure(text=f"(x: {x_cm:.2f}, y: {y_cm:.2f})")

# Ekran koordinatlarına dönüşüm
canvas_width = self.empty_area.winfo_width()
canvas_height = self.empty_area.winfo_height()
max_x_cm = 400
max_y_cm = 346.41

scale_x = (canvas_width * 0.8) / max_x_cm
scale_y = (canvas_height * 0.8) / max_y_cm
scale = min(scale_x, scale_y)
offset_x = canvas_width * 0.1
offset_y = canvas_height * 0.1

x_pixel = x_cm * scale + offset_x
y_pixel = canvas_height - (y_cm * scale + offset_y)

self.update_marker_on_canvas(x_pixel, y_pixel, x_cm, y_cm)
self.canvas.delete("distance_line")

# Sabit istasyonların ekran pozisyonları
anchor_rel_pos = {
   "IS1": (0.1, 0.9),
   "IS2": (0.9, 0.9),
   "IS3": (0.5, 0.1),
}

distances = {"IS1": d1_cm, "IS2": d2_cm, "IS3": d3_cm}
colors = {"IS1": "yellow", "IS2": "blue", "IS3": "purple"}
labels = {"IS1": "r1", "IS2": "r2", "IS3": "r3"}

# Her istasyon için çizgi ve etiket oluştur
for station in ["IS1", "IS2", "IS3"]:
   rel_x, rel_y = anchor_rel_pos[station]
   anchor_x = canvas_width * rel_x
   anchor_y = canvas_height * rel_y

   # Mesafe çizgisi
   self.canvas.create_line(anchor_x, anchor_y, x_pixel, y_pixel,
                           fill=colors[station], width=2, tags="distance_line")

   # Mesafe etiketi
   mid_x = (anchor_x + x_pixel) / 2
```

```
        mid_y = (anchor_y + y_pixel) / 2 - 10

        label_text = f"{labels[station]}: {distances[station]:.1f} cm"
        self.canvas.create_text(mid_x, mid_y,
                                text=label_text,
                                fill=colors[station],
                                font=("Arial", 11, "bold"),
                                tags="distance_line")

def update_marker_on_canvas(self, x, y, x_cm, y_cm):
    """Hareketli vericiyi ve koordinat bilgisini günceller"""
    box_size = 20
    left = x - box_size / 2
    top = y - box_size / 2
    right = x + box_size / 2
    bottom = y + box_size / 2

    self.canvas.coords(self.green_box, left, top, right, bottom)

    if hasattr(self, 'coord_label'):
        self.canvas.delete(self.coord_label)

    self.coord_label = self.canvas.create_text(
        x,
        top - 15,
        text=f"({x_cm:.1f}, {y_cm:.1f})",
        fill="white",
        font=("Arial", 12, "bold"),
        tags="coord_label"
    )

def create_green_box(self):
    """Hareketli verici için yeşil kutu oluşturur"""
    box_size = 20
    cx = self.empty_area.winfo_width() // 2
    cy = self.empty_area.winfo_height() // 2
    self.green_box = self.canvas.create_rectangle(
        cx - box_size//2, cy - box_size//2,
        cx + box_size//2, cy + box_size//2,
        fill="#006400",  # Koyu yeşil renk
        outline="black",
        width=2,
        tags="green_box"
    )

def draw_axes(self):
    """Koordinat eksenlerini çizer"""
    canvas_width = self.empty_area.winfo_width()
```

```
canvas_height = self.empty_area.winfo_height()

# Önceki eksenleri temizle
for line_id in self.axis_lines:
    self.canvas.delete(line_id)
self.axis_lines = []

# Sabit istasyonların göreceli pozisyonları
anchor_rel_pos = {
    "IS1": (0.1, 0.9),  # Sol alt (orijin)
    "IS2": (0.9, 0.9),  # Sağ alt
    "IS3": (0.5, 0.1)   # Üst orta
}

# Orijin noktası koordinatları
origin_x = canvas_width * anchor_rel_pos["IS1"][0]
origin_y = canvas_height * anchor_rel_pos["IS1"][1]

# X ekseni çizgisi (IS1'den IS2'ye)
is2_x = canvas_width * anchor_rel_pos["IS2"][0]
is2_y = canvas_height * anchor_rel_pos["IS2"][1]

# Y ekseni çizgisi (IS1'den IS3'e)
is3_x = canvas_width * anchor_rel_pos["IS3"][0]
is3_y = canvas_height * anchor_rel_pos["IS3"][1]

# X ekseni çizimi
x_axis = self.canvas.create_line(
    origin_x, origin_y, is2_x, origin_y,
    fill="white", width=2, arrow="last"
)

# Y ekseni uzunluğu hesaplama
y_axis_length = origin_y - is3_y
y_axis_end_y = origin_y - y_axis_length

# Y ekseni çizimi
y_axis = self.canvas.create_line(
    origin_x, origin_y, origin_x, y_axis_end_y,
    fill="white", width=2, arrow="last"
)

# Eksen etiketleri
x_label = self.canvas.create_text(
    (origin_x + is2_x)/2, origin_y - 15,
    text="X", fill="white", font=("Arial", 12, "bold")
)
```

```
        y_label = self.canvas.create_text(
            origin_x + 15, (origin_y + y_axis_end_y)/2,
            text="Y", fill="white", font=("Arial", 12, "bold")
        )

        # Orijin etiketi
        origin_label = self.canvas.create_text(
            origin_x + 20, origin_y - 15,
            text="Orijin (IS1)", fill="white", font=("Arial", 10, "bold")
        )

        # Çizilen öğeleri sakla
        self.axis_lines.extend([x_axis, y_axis, x_label, y_label, origin_label])

    def load_background_image(self):
        """Arka plan resmini yükler ve boyutlandırır"""
        try:
            image = Image.open(self.background_image_path)
            image = image.resize((self.empty_area.winfo_width(),
self.empty_area.winfo_height()), Image.LANCZOS)
            self.background_image = ImageTk.PhotoImage(image)

            self.canvas.create_image(0, 0, anchor="nw", image=self.background_image,
tags="background")
            self.canvas.tag_lower("background")
            self.draw_axes()

        except Exception as e:
            print(f"Arka plan resmi yükleme hatası: {e}")  # Türkçe hata mesajı

    def load_default_locations(self):
        """Varsayılan konum ayarlarını yükler"""
        if os.path.exists(self.locations_file):
            with open(self.locations_file, "r") as file:
                self.locations_data = json.load(file)
                self.update_markers_event()

    def update_markers_event(self):
        """Sabit istasyonları günceller"""
        if not self.locations_data:
            return

        bolge_1 = self.locations_data.get("Region 1", 400)  # Varsayılan 400 cm
        bolge_2 = self.locations_data.get("Region 2", 400)  # Varsayılan 400 cm
        bolge_3 = self.locations_data.get("Region 3", 400)  # Varsayılan 400 cm

        self.canvas.delete("marker")
        self.canvas.delete("texts")
```

```
        self.draw_distance_line(0.1, 0.9, 0.9, 0.9, bolge_1)
        self.draw_distance_line(0.1, 0.9, 0.5, 0.1, bolge_2)
        self.draw_distance_line(0.9, 0.9, 0.5, 0.1, bolge_3)

    def draw_distance_line(self, x1_rel, y1_rel, x2_rel, y2_rel, distance):
        """İki nokta arasına mesafe çizgisi ve etiketi çizer"""
        x1 = self.empty_area.winfo_width() * x1_rel
        y1 = self.empty_area.winfo_height() * y1_rel
        x2 = self.empty_area.winfo_width() * x2_rel
        y2 = self.empty_area.winfo_height() * y2_rel

        self.canvas.create_line(x1, y1, x2, y2, fill="red", width=2)

        mid_x = (x1 + x2) / 2
        mid_y = (y1 + y2) / 2
        self.canvas.create_text(mid_x, mid_y, text=f"{distance:.2f} cm", fill="white",
font=("Arial", 12, "bold"), tags="texts")

    def open_settings(self):
        """Ayarlar penceresini açar"""
        settings_window = customtkinter.CTkToplevel(self)
        settings_window.title("Ayarlar")  # Türkçe pencere başlığı
        settings_window.geometry("500x400")

        labels = []
        entries = []

        locations = self.locations_data if self.locations_data else {}

        # Bölge ayarları için giriş alanları
        for i in range(3):
            region_name = f"Bölge {i+1}"  # Türkçe bölge adı
            default_value = locations.get(f"Region {i+1}", 400)  # Varsayılan 400 cm

            label = customtkinter.CTkLabel(settings_window, text=f"{region_name} (cm)")
            label.grid(row=i, column=0, padx=10, pady=10)

            entry = customtkinter.CTkEntry(settings_window, width=200)
            entry.insert(0, default_value)
            entry.grid(row=i, column=1, padx=10, pady=10)

            labels.append(label)
            entries.append(entry)

        def save_settings():
            """Ayarları kaydeder ve pencereyi kapatır"""
            new_locations = {}
```

```
            for i, entry in enumerate(entries):
                try:
                    new_locations[f"Region {i+1}"] = int(entry.get())
                except ValueError:
                    new_locations[f"Region {i+1}"] = 400  # Varsayılan 400 cm

            with open(self.locations_file, "w") as file:
                json.dump(new_locations, file, indent=4)

            self.load_default_locations()
            settings_window.destroy()

        save_button = customtkinter.CTkButton(settings_window, text="Kaydet",
command=save_settings)  # Türkçe buton metni
        save_button.grid(row=3, column=0, columnspan=2, pady=20)

    def clear_existing_files(self):
        """Mevcut veri dosyalarını temizler"""
        device_file = "device_data.xlsx"
        if os.path.exists(device_file):
            os.remove(device_file)

    def save_to_excel(self):
        """Ölçüm verilerini Excel dosyasına kaydeder"""
        device_file = "device_data.xlsx"
        timestamp = time.strftime("%Y-%m-%d %H:%M:%S")
        device_df = pd.DataFrame([{"timestamp": timestamp, **self.device_data}])

        if os.path.exists(device_file):
            existing_device_df = pd.read_excel(device_file, engine='openpyxl')
            device_df = pd.concat([existing_device_df, device_df], ignore_index=True)

        device_df.to_excel(device_file, index=False, engine='openpyxl')

    def on_closing(self, event=0):
        """Pencere kapatıldığında çalışır"""
        self.destroy()

    def start(self):
        """Uygulamayı başlatır"""
        self.mainloop()

if __name__ == "__main__":
    app = App()
    app.start()
```

ÖZGEÇMİŞ

Yiğit Batuhan TUNA

Eğitim

Yüksek Lisans	2025	Maltepe Üniversitesi, Lisansüstü Eğitim Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı
Lisans	2018	İstanbul Okan Üniversitesi, Mühendislik Fakültesi, Elektrik-Elektronik Mühendisliği (İngilizce)

İş/İstihdam (Varsa)

Yıl		Görev
2016	Tübitak UME Elektromanyetik Laboratuvarı	Stajyer
2017	Tübitak UME Optik Laboratuvarı	Stajyer