T.C. İSTANBUL TİCARET ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

KONUMA BAĞLI DEPO YÖNETİM SİSTEMLERİNDE RFID VE BARKOD YÖNTEMLERİNİN KARŞILAŞTIRILMASI

Hüseyin Cahit TOSUN

Danışman
Prof. Dr. Abdül Halim ZAİM

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
İSTANBUL - 2018

KABUL VE ONAY SAYFASI

Hüseyin Cahit TOSUN tarafından hazırlanan **"KONUMA BAĞLI DEPO YÖNETİM SİSTEMLERİNDE RFID VE BARKOD YÖNTEMLERİNİN KARŞILAŞTIRILMASI"** adlı tez çalışması 25/09/2018 tarihinde aşağıdaki jüri üyeleri önünde başarı ile savunularak, İstanbul Ticaret Üniversitesi Fen Bilimleri Enstitüsü **Bilgisayar Mühendisliği Anabilim Dalı**'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Danışman	Prof. Dr. Abdül Halim ZAİM İstanbul Ticaret Üniversitesi
Jüri Üyesi	Dr. Öğr. Üyesi Mustafa Cem KASAPBAŞI İstanbul Ticaret Üniversitesi
Jüri Üyesi	Dr. Öğr. Üyesi Muhammed Ali AYDIN İstanbul Üniversitesi

Onay Tarihi : 17.10.2018

Prof. Dr. Necip ŞİMŞEK
Enstitü Müdürü

AKADEMİK VE ETİK KURALLARA UYGUNLUK BEYANI

İstanbul Ticaret Üniversitesi, Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında,

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- ve bu tezin herhangi bir bölümünü bu üniversitede veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

25/09/2018

Hüseyin Cahit TOSUN

İÇİNDEKİLER

	Sayfa
İÇİNDEKİLER	i
ÖZET	iii
ABSTRACT	iv
TEŞEKKÜR	v
ŞEKİLLER DİZİNİ	vi
ÇİZELGELER DİZİNİ	viii
SİMGELER VE KISALTMALAR DİZİNİ	ix
1. GİRİŞ	1
2. LİTERATÜR ÖZETİ	2
3. DEPO YÖNETİM SİSTEMLERİ	3
3.1. Depo Yönetim Sistemleri Nedir?	3
3.2. Ana Veri Tanımları	3
3.2.1. Şirket tanımları	3
3.2.2. Malzeme ana verileri tanımı	4
3.2.2.1. Malzeme ana verilerisi örneği	4
3.2.3. Depo içi lokasyon kaynaklarının tanımı	5
3.2.4. Personel ve ekipman gibi kaynakların tanımı	6
3.3. Giriş Hareketleri	6
3.3.1. Satınalma siparişi	6
3.3.2. Depo giriş siparişi	7
3.3.2. Gerçekleşen mal kabul bilgisi	7
3.4. Depo İçi Hareketler	8
3.4.1. Yerleştirme	8
3.4.2. Transfer	9
3.4.3. Karantina sebebi ile atama ya da blokaj koyma	9
3.4.4. Kit yapma / bozma	9
3.4.5. Sayım	9
3.4.6. Diğer katma değerli işlemler	9
3.5. Çıkış Hareketleri	10
4. STANDART BARKOD SİSTEMLERİ İLE DEPO YÖNETİMİ UYGULAMALARI	10
5. RFID SİSTEMLERİ İLE DEPO YÖNETİMİ UYGULAMALARI	13
5.1. RFID Teknolojisi	13
5.2. 2 Boyutlu düzlemde RFID konum belirleme	13
5.3. 3 Boyutlu düzlemde RFID konum belirleme	15
5.3.1. TOA(Time of arrival/Geliş zamanı) metodu	16
6. STANDART BARKOD SİSTEMLERİ İLE RFID SİSTEMLERİN KARŞILAŞTIRILMASI	18
7. RFID İLE KONUM BELİRLEMEDE KULLANILAN TEKNOLOJİLER	20
7.1 Deneysel Çalışma Ortamı	20
7.1.1. Kullanılan programlama dilleri, uygulamalar ve donanımlar	20
7.1.2. Mesaj kuyruğu	21
7.1.2.1 ActiveMQ ile mesaj kuyruğu	22
7.1.2.2 RabbitMQ ile mesaj kuyruğu	23

7.1.2.3 RabbitMQ ve ActiveMQ karşılaştırması 26
7.1.3. NoSQL veritabanı 29
7.1.3.1. MongoDB ve Cassandra karşılaştırması 31
7.1.4. MQTT Nedir? 32
7.1.4.1. MQTT Çalışma prensibi 32
7.1.4.2. MQTT İletişim amaçlı mesaj türleri 32
7.1.4.3. MQTT Genel özellikleri 33
7.1.4.4. MQTT de Güvenlik 33
7.1.4.5. MQTT de Servis kalitesinin garanti altına alınması 33
7.1.4.5.1. QoS 0 33
7.1.4.5.2. QoS 1 34
7.1.4.5.3. QoS 2 35
7.1.4.5.4. QoS 0 Nezaman kullanılır? 37
7.1.4.5.5. QoS 1 Nezaman kullanılır? 37
7.1.4.5.6. QoS 2 Nezaman kullanılır? 37
7.1.4.5.7. MQTT Yayın – abonelik 37
7.1.4.5.8. MQTT Kullanıcı güvenliği 38
7.1.4.5.9. MQTT ve AMQP karşılaştırması 38
7.1.4.5.10. Mesajlaşma senaryolarına göre MQTT ve AMQP karşılaştırması 38
7.1.4.5.11. İletişimde kullanılan mesaj yapısı 39
7.1.4.5.12. ARDUINO uygulaması 39
7.1.4.5.13. Servis uygulaması 41
7.1.5. Servis uygulaması arayüz 44
8. SONUÇ VE ÖNERİLER 46
KAYNAKLAR 48
ÖZGEÇMİŞ 50

ÖZET

Yüksek Lisans Tezi

KONUMA BAĞLI DEPO YÖNETİM SİSTEMLERİNDE RFID VE BARKOD YÖNTEMLERİNİN KARŞILAŞTIRILMASI

Hüseyin Cahit TOSUN

İstanbul Ticaret Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Abdül Halim ZAİM

2018, 50 sayfa

Günümüzde yaygın olarak kullanılan depo takip sistemlerinin kurumlara olan verimlilik maliyetleri, çevresel ve sağlık etkilerinin ne düzeyde olduğu tam olarak bilinmemektedir. Tüketimin her geçen gün artışı, depo yönetim sistemlerinde verimliliğin gerekliliğini ön plana çıkartmaktadır. Bununla birlikte çevresel fayda ve zararların da göz ardı edilmemesi gerekmektedir. Kurumların yapmış olduğu depo yatırımları, kurgulanan sürece göre faydadan çok zararlara sebep olabilmektedir. Bu dokümanda depo yönetimi amacı ile kullanılan, standart barkod yapısına alternatif olarak RFID sistemlerle konum belirleme yöntemlerinin kullanımı, verimlilik, finans ve çevresel etkilerinin karşılaştırılması açıklanmıştır. Bu dokümanın, mevcut teknolojilerin iyileştirilmesi ve kurumların depo yönetimi uygulamalarında tercihlerini bir karşılaştırma üzerine kurgulamasında fayda sağlaması beklenmektedir. Karşılaştırmanın amacı iki farklı yönetim sisteminin getireceği avantaj ve dezavantajları göz önüne koymak ve farklılıkların maliyet/amortisman ve fayda ekseninde değerlendirilmesini sağlamaktır. Ayrıca bu dokümanda, ülkemizin teknolojik alanlarda göstermiş olduğu gelişmeler ile Endüstri 4.0 a atılan adımlarda Nesnelerin İnterneti (Internet of Things - IoT)'nin depo yönetim sistemlerinde kullanımı vurgulanmaktadır. Araştırmalar göstermiştir ki, ilk yatırım bedeli yüksek olsa da devamlılığı ve çevresel faydaları göz önüne alındığında RFID depo yönetim sistemleri tercih sebebi olmalıdır.

Anahtar Kelimeler: RFID, Nesnelerin interneti, kablosuz konum tahmini, (AOA)Sinyal geliş açısı, (TOA) Sinyal geliş süresi, Çok yüksek frekans-UHF.

ABSTRACT

M.Sc. Thesis

COMPARISON OF RFID AND BARCODE METHODS IN LOCATION RELATED STORAGE MANAGEMENT SYSTEMS

Hüseyin Cahit TOSUN

İstanbul Commerce University
Graduate School of Applied and Natural Sciences
Department of Computer Engineering

Supervisor: Prof. Dr. Abdül Halim ZAİM

2018, 50 pages

It is not known exactly what the efficiency costs of the warehouse monitoring systems used today and the environmental and health effects are to the institutions. Every day increase in consumption, efficiency is required in warehouse management systems. However, environmental benefits and losses should not be overlooked. Warehouse investments made by corporations can cause damages rather than benefits according to the process that is being constructed. This document describes the comparison of the use, efficiency, financial and environmental impacts of positioning methods with RFID systems as an alternative to the standard bar code structure used with warehouse management purposes. It is expected that this document will be useful for improving existing technologies and for building a comparison of the preferences of institutions in warehouse management practices. The aim of the comparison is to take into account the advantages and disadvantages of the two different management systems and to ensure that the differences are assessed on the basis of cost / depreciation and utility. In addition, this document highlights the developments that our country has shown in technological areas and the use of the Internet of Things (IoT) in warehouse management systems in the steps taken in Industry 4.0. Research has shown that although the initial investment cost is high, RFID storage management systems should be the reason for preference given the continuity and environmental benefits.

Keywords: RFID, IoT, Wireles location estimation,(AOA) Angel of arrival, (TOA)Time of arrival,Ultra high frequency-UHF.

TEŞEKKÜR

Bu araştırma için beni yönlendiren, karşılaştığım zorlukları bilgi ve tecrübesi ile aşmamda yardımcı olan değerli Danışman Hocam Prof. Dr., Abdül Halim ZAİM'e teşekkürlerimi sunarım.

Tezimin her aşamasında beni yalnız bırakmayan yol arkadaşım Gökçe LAKERTA'ya sonsuz sevgi ve saygılarımı sunarım.

Hüseyin Cahit TOSUN
İSTANBUL, 2018

ŞEKİLLER

Sayfa

Şekil 3.1. Depo Yönetim Sistemleri Mal Kabul Süreçleri 8
Şekil 4.1. UPC/EAN Barkod 11
Şekil 4.2. QR CODE 12
Şekil 5.1. RFID Sistem Bileşenleri 13
Şekil 5.2. RSSI Kalite Hesabı 15
Şekil 5.3. RFID TOA Kurgusu 17
Şekil 5.4. RFID AOA Kurgusu 17
Şekil 6.1. UHF Anten Konumlandırma Haritası 19
Şekil 7.1. Mesaj Kuyruğu 22
Şekil 7.2. ActiveMQ Mesaj Kuyruğu 23
Şekil 7.3. RabbitMQ Kuyruk Yapısı Anten Grupları 26
Şekil 7.4. ActiveMQ Kodu 27
Şekil 7.5. RabbitMQ Kodu 28
Şekil 7.6. RabbitMQ Zamansal Sonuçları 28
Şekil 7.7. ARDUINO UNO Geliştirme Kartı 29
Şekil 7.8. Cassandra ve MongoDB Zamansal Karşılaştırması 31
Şekil 7.9. QoS 0 Anlatımı 34
Şekil 7.10. QoS 1 PUBACK Anlatımı 34
Şekil 7.11. QoS 1 PUBACK Paket Yapısı 35
Şekil 7.12. QoS 2 Yapısı 35
Şekil 7.13. QoS 2 PUBREC Paket Yapısı 36
Şekil 7.14. QoS 2 PUBCOMP Paket Yapısı 36
Şekil 7.15. QoS 2 PUBREL Paket Yapısı 37
Şekil 7.16. Servis Uygulaması Arayüzü 45

ÇİZELGELER

Sayfa

Çizelge 5.1. Hesaplama Yöntemlerine Göre Doğruluk Oranları..... 15
Çizelge 6.1. Yıllık Maliyet Tablosu..... 19
Çizelge 7.1. RabbitMQ ve ActiveMQ Karşılaştırması..... 29
Çizelge 7.2. MongoDB ve Cassandra Karşılaştırması..... 31

SİMGELER VE KISALTMALAR

AoA	Angel of Arrival
DÇS	Depo Çıkış Siparişi
DGS	Depo Giriş Siparişi
IoT	Internet of Things
JSON	JavaScript Object Notation
MQ	Message Queue
MQTT	Message Queue Telemetry Transport
QoS	Quality of Service
QRCODE	Quick Response Code
RFID	Radio-frequency identification
RSSI	Radio Signal Strenght Indicatior
SKT	Son Kullanma Tarihi
ToA	Time of Arrival
UHF	Ultra High Frequency
UPC	Universal Product Code
ÜRT	Üretim Tarihi
WMS	Warehouse Management System

1. GİRİŞ

Ülkemizde ve Dünyada kurumsal yapıların depo yönetim sistemleri maliyetleri göz önünde bulundurulduğunda çok ciddi bir öneme sahip olmaktadır. Özellikle üretim yapan kurumların depo yönetim sistemleri üretimin hızı ile doğru orantılı olarak gelişmektedir. Bu hız ile birlikte daha az alan ve daha hızlı ulaşım fikri kurumları yeni depo yönetim sistemlerine itmektedir. Depoların fiziki şartlarının iyileştirilmesinin önemi kadar depoda yer alan malzemelerin takibi de bir okadar önem arz etmektedir. Güncel teknolojiler ve Endüstri 4.0 gibi kapsamlı bir yeniliğin ülkemizde de adımları atılmakta ve doğru uygulama şeklinin belirlenmesi her geçen gün önemini arttırmaktadır. Var olan mevcut sistemlerin yerine gelecek olan yeni sistemlerin daha iyi sonuçlar üretmesi ve daha verimli bir prensip ile çalışması her kurumun temel isteğidir. Bu gibi bir tercih noktasında kurumların daha fazla yol gösterici kaynaklara sahip olması ülkemiz adına da ekonomik anlamda fayda sağlayacaktır. Bu dokümanda var olan standart barkod sistemleri ve RFID ile depo yönetim sistemleri tariflenmekte ve RFID sistemlerinin kullanımı için derinlemesine bir araştırma gündeme getirilmektedir. Güncel teknolojiler ve özellikle açık kaynaklı uygulamalar sayesinde RFID sistemlerinin depolarda kullanım maliyetleri ve bununla birlikte getirmiş olduğu avantajlar göz önünde bulundurulmuştur.

2. LİTERATÜR ÖZETİ

Zhao , Patwari , Agrawal ve Rabbat (2012), RFID etiketlerinin insan takibi için konum belirleme ve buna bağlı güvenlik uyglamalarında kullanımına deneysel çalışmalar yapmıştır. Çalışmalar göstermiştir ki kişinin taktığı cihazın anten kazanç paterni üzerinde büyük etkisi bulunmaktadır. Bu durum farklı alanlarda değerlendirildiğinde insan ya da herhangi başka bir nesnenin de aynı şekilde etkileneceği öngörülmüştür. Bu yan etkilerin bertaraf edilmesi için standart RSSI ölçümlerine alternatif bir kazanç ve konum tahmini (AGAPE) algoritmasını önetmiştir. Sonuçlar konum tahminlerinde oluşan hata paylarını büyük oranda azaltmaktadır.

Y. Zhang, L. Xie, Y. Bu, Y. Wang, J. Wu and S. Lu (2018), Yapmış oldukları çalışmada RFID etiketlerini kullanarak 3 boyutlu lokalizasyon gerçekleştiren 3Dloc önermektedir. Yapılan çalışmada asıl ilgilenilen nokta AoA kavramının ikili anten yükseltileri ve minimum 3 anten grubu içerisindeki nesne için tahminleri göstermektedir. Tahminler neticesinde varılan nokta ortalama 10 ila 15.3 cm arasında sapmalarla 3 boyutlu lokasyon belirlemenin gerçekleştirilmesi şeklinde sonuçlanmıştır.

Han (2008), RFID etiketlerinin insan hayatında hızlıca yer kaplaması ile birlikte bunun avantajlı kullanım alanı olarak Depo Yönetim Sistemlerinde tercih edilmesi üzerine yapmış olduğu çalışmada standart barkod sistemleri ile RFID etiketlerin kullanımın karşılaştırmasında bulunmuş ve sonuçlar göstermiştir ki RFID taşımacılık, depo yönetimi satış ve benzeri alanlarda oldukça büyük bir marketin kapılarını aralamaktadır.

3. DEPO YÖNETİM SİSTEMLERİ

3.1. Depo Yönetim Sistemleri Nedir?

WMS (Warehouse Management System) yani depo yönetim sistemi; mamul, yarı mamul, ham madde ve son yıllarda eklenen yeni bir kalem olarak basılı evrakların depo olarak tarif edilen tesis içerisinde takibini sağlayan ve depolanan kalemler ile birlikte kaynakların operasyon ihtiyacına göre en doğru ve verimli biçimde kullanılmasını sağlayan metodolojiler ve yazılım bütünüdür.

Yazılım içerisinde bilgiler veri tabanı ve farklı yazılım dilleri ile yazılmış yazılımlar ile fiziksel operasyonlara göre kurgulanır. Depo içerisindeki kurgulara göre fonksiyonlar aşağıdaki ana başlıklar altında gruplanabilir.

1. Ana veri tanımlamaları
2. Depo içi lokasyonların ve personel, ekipman gibi kaynakların tanımlamaları
3. Giriş hareketleri
4. Depo içi hareketler
5. Çıkış hareketleri

Bilginin işlenmesi bakımından ise yazılım 3 ana kısımda incelenebilir.

1. Tanımlamalar
2. İşlemler
3. Raporlar

3.2. Ana Veri Tanımları

3.2.1. Şirket tanımları

Depo içerisinde ürün hareketlerinin sebep ve sonuçlarını belirleyen şirketlerin tanımlamalarıdır. Tanımlanan şirketler Mal Sahibi, Tedarikçi, Gönderen, Müşteri, Taşeron gibi roller ile tanımlanabilirler. Bu roller depo içerisinde yapılacak operasyonları etkileyebildği gibi bilginin takip edilmesinde ve raporların çeşitli kırılımlar ve gruplamalar ile verilmesi için de kullanılmak üzere kaydedilebilirler. Aynı şirket aynı depo operasyonları içerisinde hem mal sahibi hem de tedarikçi olarak ya da hem tedarikçi hem de müşteri gibi çoklu tanımlar ile kullanılabilirler. Şirket kodu, ünvanı gibi bir şirket için yalnızca 1 tane olması gereken

tanımlamalar gibi farklı tiplerdeki adres tanımlamaları gibi bir şirket için birden fazla tanımlanan bilgiler şeklinde de olabilir.

3.2.2. Malzeme ana verileri tanımı

Depo içerisinde tüm giriş, çıkış ve hareketlere konu olacak mamul, yarı mamul, ham madde ya da basılı evrakların tanımlamalarını tarifler. Temel olarak aşağıdaki bileşenlerden oluşur:

- *Başlık tanımları:* Malzeme için yalnızca 1 kez tanımlanması gereken malzeme kodu, malzeme açıklaması gibi bilgileri tanımlar.
- *Paket tipleri:* Malzemenin depo içerisinde işlem görecek kaplarını ve bu kap çeşitlerinin birbirleri ile ilişkilerini belirtir. Depo içinde elleçlenebilen en küçük paket tipine tüketim birimi ismi verilir ve tüm diğer paket tipleri bu birim üzerinden değerlendirilirler.
- *Barkodlar:* Depo içerisinde fiziksel işlemlerin yapılması için doğrudan ürüne, ürün ile birlikte paket tipine ya da (Barkodun ne olduğunun tanımı ayrıca yapılacak, eklere koyulabilir.)
- Malzeme Özellikleri: Malzemelerin hayatları boyunca değişmeyecek ve aynı kabul edilecek özellikleri belirtir.
- *Stok Özellikleri:* dedpoya giren her bir adet için değişebilecek özellikleri belirtir. Bu sebeple malzeme ana verisi üzerinde yalnızca bu özelliklerin takip edilip edilmeyeceği ya da takip edilecek ise ne şekilde takip edileceği belirtilir.

3.2.2.1. Malzeme ana verisi örneği

Başlık Tanımları:

Malzeme Kodu: 231854621

Malzeme Açıklaması: Pastörize, tam yağlı, kaymaklı yogurt 500g.

Paket Tipleri:

- Tüketim Birimi: Adet
- Paket Tipi: Koli (10 Adet)
- Paket Tipi: Kutu (40 Adet / 4 paket)

Barkodlar:

- Barkod 1: 69789654120001515 – 1 Adet
- Barkod 2: 69789654120001577 – 1 Koli (10 Adet)

Malzeme Özellikleri:

- Yağ Oranı: %7
- PH: 4.2

Stok Özellikleri:

- Seri Numarası Takibi: Hayır
- Lot Numarası Takibi: Evet
- Üterim Tarihi Takibi: Evet
- Son Tüketim Tarihi Takibi: Evet

3.2.3. Depo içi lokasyon kaynaklarının tanımı

Lokasyon Tanımlamaları: Lokasyonlar depo içerisindeki her bir ayrı alanı belirten local koordinatlama sistemidir. Buna göre depo içerisinde tanımlanan her bir yerin kendine ait tekrar etmez bir ismi bulunur. Depo içi işlemlerde stokların depo içerisinde nerede oldukları bu koordinat isimlendirme sistemi ile bilinir. Aynı şekilde depo içi hareketler gerçekleştiğinde WMS'I kullanan kullanıcıların ürünün nerede işlem gördüğü ya da hangi kaynak lokasyondan hangi hedef lokasyona gittiği bilgilerini kayıt altına alması beklenir.

Lokasyonların isimlendirilmesinde genellikle insanların alışılagelmiş kullanım alışkanlıkları kullanılarak satranç tahtasında da kullanılan koordinatlama sistemi kullanılagelmektedir. Buna göre depo enlem ve boylamlara bölünerek her bir enlem ve boylam isimlendirilir. Lokasyonların bu mantıksal bölümlere göre hangi enlem ve boylamda olduğu bilgisi de lokasyonun depo içerisinde 2 boyutlu düzelmde yani zemin üzerinde nerede olduğunu gösterir. Eğer ayrıca bir dikine raf sistemi var ise yükseklik için de ayrıca bir kod daha eklenerek lokasyonun kaçıncı kata olduğu (mantıksal olarak zeminden yüksekliği) da bu kodlama yordamına eklenir. Eğer depo birden fazla kat ya da fiziksel olarak birbirlerinden tamamen farklı kısımlara sahip ise bunların da koda eklenmesi ile tam bir tanımlama sağlanabilir.

Depo içerisinde fiziksel durumlarına göre lokasyonlar da gruplara ayrılır. En genel hali ile aşağıdaki şekilde gruplanabilir.

Rampalar: Depoya giriş ve çıkış hareketlerinin yapıldığı yerlerdir. Bu tip alanlar için genelde kapasite kısıtları kullanılmaz. Ayrıca mal kabul ve sevkiyat gibi işlemlerin bu lokasyon tipleri içerisinde yapılması beklenir. Aynı şekilde sistem

üzerinde de bu işlemlere "yalnızca bu tip lokasyonlarda yapılabilir" kısıtı ile operasyonların doğru yapılması için kısıtlamalar da getirilebilir.

Alanlar: Depo içerisindeki büyük alanları ifade eder. Genelde serbest kullanım ve kapasite kısıtlarının kullanılmadığı lokasyonlar için kullanılır.

Raflar: Raflı depolama sistemleri için kullanılır. Kullanım şekillerine göre kendi içinde de alt kırılımları olabilir. Sırt sırta raflar, direve in raflar, dirve through, vb… Bu tip lokasyonlar kapasite kısıtlarının ve kuralların en sık uygulanabildiği tiplerdir. Buna parallel olarak raporlama ve detaylı takipler de bu tip lokasyonlarda çok daha keskin yapılabilir.

İş istasyonları: Tüm depolarda kullanılmamakla beraber basit montaj ya da katma değerli işlemlerin yapıldığı depolarda depo içi ürün özelliklerinin farklılaştırılması için ayrılan alanların tanımları için kullanılırlar. Ayrıca lokasyonların her birine ait kodları barkod şeklinde lokasyonların üzerine asılarak kullanıcıların nerede işlem yapıldığı bilgisini sistem içerisine en hızlı ve hatasız biçimde aktarılması da genel kullanım yöntemleri arasındadır.

3.2.4. Personel ve ekipman gibi kaynakların tanımı

Depo içi işlerin yapılması için kullanılan kaynakların tümünü ifade eder. Personel tanımlamaları her bir çalışanın ne zaman hangi işi yaptığının takip edilebilmesi için kaçınılmazken ekipman tanımlamaları daha nadiren takip edilirler. Ayrıca her bir personel çalışma ve erişim tiplerine göre yetkilendirilerek hem kullanıcıların yalnızca kendi uzmanlıkları ile erişimler sağlanarak iş çalışan adına sadeleştirilecek hem de yüksek yetki seviyesinde kullanıcıların daha fazla bilgiye erişimi sağlanarak bilgi gizliliği sağlanmış olacaktır.

3.3. Giriş Hareketleri

Giriş hareketleri ürünler fiziksel olarak depoya ulaşmadan önce depoya gelmesi planlanan ürünlerin anlaşmalarının tanımlanması ile başlar ve ürünlerin depoya ulaşıp mal kabul yapılması ve stoğa alınmasına kadar olan süreçlerin tümünü ifade eder.

3.3.1. Satınalma siparişi

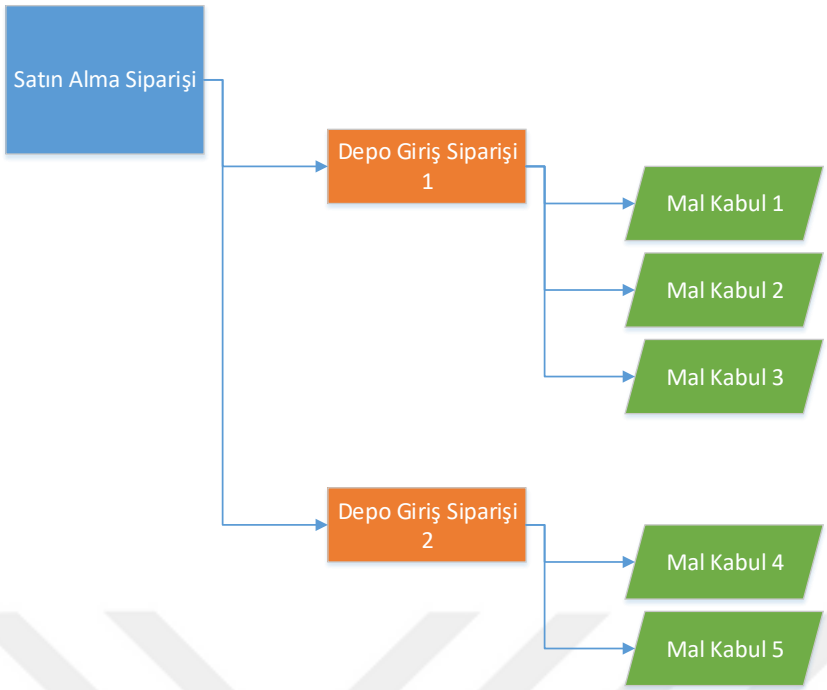
Mal sahibi ile tedarikçinin uzun vadeli toptan satın alma anlaşmalarını ifade eder. Buna göre tedarikçi ve mal sahibi depoya gelmesi gereken ürün kodları ve miktarları bazında anlaşarak genel bir sınır tanımlarlar. Burada tanımlanan ürünler farklı miktarlar ile farklı zazmanlarda depoya gelebilir. WMS'in buradaki en büyük rolü, mal kabul sırasında gelen ürünlerin Satın Alma Siparişine uygunluğu ve yapılan anlaşma kuralları içerisinde olup olmadığını kontrol etmek ve bazen sınırlandırmak, bazen ise bu karşılaştırmalar sonucunda detaylı raporlar çıkartıp yönetimsel bazda doğruluk ve uygunluğun ölçülmesidir.

3.3.2. Depo giriş siparişi

Depoya bir irsaliye ile tek zamanda gelecek ürünlerin tanımlanması için kullanılan beklenen teslim alma bilgilerini ifade eder. Tedarikçinin o geliş ile bildireceği malzeme bilgisi, miktar, palet numarası (LPN/ SSCC), Lot No, SKT, ÜRT, karantina durumu, hangi müşteri için bu stokların mal kabul yapılacağı gibi detay bazında tüm bilgleri içerebilir. Gelececek olan malzemelerin bilgileri ne kadar detaylı biçimde DGS içerisine tanımlanabilir ise mal kabul sonrası yapılacak karşılaştırma ve rapor kesinlikleri o kadar doğru olur.

3.3.3. Gerçekleşen mal kabul bilgisi

Depoya fiziksel olarak gelen malzemelerin gerçek adet ve özellikleri ile sayımını ifade eder. Mal kabul süreci tamamlanan malzemeler stok olarak kabul edilir. Genellikle yapılan işi gerçek zamanlı yakip edebilmek ve işin doğruluğunu arttırmak için günümüzde mal kabul operasyonları sahada kullanılan teknolojik ekipmanlar yardımı ile yapılmaktadır. Bu destekleyici ekipman ve operasyonel yaklaşımlar ilerleyen bölümlerde Operasyon yordamları ve teknoloji kısmında anlatılacaktır.



Şekil 3.1. Depo Yönetim Sistemleri Mal Kabul Süreçleri

3.4. Depo İçi Hareketler

Bu tipteki hareketler depoya giriş ya da depodan çıkış hareketlerinin dışında kalan tüm işlem tiplerini içerir. Genel olarak aşağıdaki başlıklar altında işlenebilir.

3.4.1. Yerleştirme

Mal kabulden hemen sonra ürünlerin depo içerisinde ürüne en uygun yere koyulması işlemidir. İdeal depo operasyon tasarımlarında WMS sisteminin kendi içinde tanımlanmış kurallar ya da algoritmalar ile statik ya da dinamik olarak ürünlerin; kısa, orta ya da uzun vadede tercih edilmiş avantajlara bağlı olarak depo içerisinde en uygun yere koyulması şeklinde de tarif edilebilir. Yerleştirme işlemi ne kadar doğru analiz edilir ve ne kadar iyi yapılır ise sonrasında yapılacak depo içi hareketleri azaltarak operasyonun verimliliğini aynı oranda yükseltir. Yerleştirme işlemleri için genel olarak benimsenmiş yordamlar bulunabilir ya da bu algoritma ya da kurallar operasyona özel olarak tasarlanıp kurgulanabilirler.

3.4.2. Transfer

Yerleştirmesi yapılmış malzemelerin ürün nitelikleri ya da özelliklerinde bir değişim o0lmadan depo içerisinde yaptığı tüm hareketler transfer işlemi olarak adlandırılabilir. Transfer işlemleri kullanıcıların insiyatifinde yapılabilmekte ayrıca ikmal işlemleri gibi sistem tarafından belirli kurallara ya da algoritmalar ile emir olarak oluşturulmuş şekilde de yapılabilir.

3.4.3. Karantina sebebi ile atama ya da blokaj koyma

Genel olarak depodaki ürünlerin hasarlanma, ambalaj bozukluğu, blokaj koyma gibi işlemler ile siparişlere tamamen kapatılması ya da koşullu olarak kapatılması için stoğun bir durumu olarak tutulan bilgilerdir. Bu işlem stok özelliklerinde bir değişiklik yapmaz ancak stoğun depo içerisinde nasıl bekleyeceğini ve siparişlere nasıl konu olacağını belirler.

3.4.4. Kit yapma / bozma

Birden fazla ürünün bir araya getirilerek yeni bir ürün kodu alması ya da kit kodu alması işlemi olarak tanımlanabilir. Aynı şekilde birleşen ve kit oluşturan ürünlerin bundan sonraki hayat döngülerinde tek bir yeni malzeme gibi devam etmesi beklenir. Kit bozma da kit yapma sürecinin tam tersidir ve bir ürünün bileşenlerine parçalanmasını tarifler.

3.4.5. Sayım

Depo içerisinde bulunan stokların sayılması ve sistem üzerindeki stok ile sayılan stokların karşılaştırılmasını ifade eder. Bunun yanında sayılmış olan stoğun yeni stok olarak güncellenmesi de tercihe bağlı olarak bu işlemin bir bileşenidir. Sayımlar tüm deponun kapatılarak yapılabildği gibi deponun bir kısmı ya da yalnızca bir lokasyon için de kısmi sayım olarak yapılabilir.

3.4.6. Diğer katma değerli işlemler

Depo içerisinde yapılan ancak temel lojistik faaliyetlerini içermeyen tüm işlemler için kullanılabilir. Montaj, paketleme, kalite kontrol, ütüleme, temizleme, ambalaj değişimi, elektronik için yazılım yükleme gibi çok geniş bir yelpazede değerlendirilebilir.

3.5. Çıkış Hareketleri

Depodan sevkiyat ile gidecek ürünlerin tüm süreçlerini kapsar. En yalın hali ile aşağıdaki opeasyonları kapsar.

1. Depo Çıkış Siparişinin kaydı:

   Depodan sevk edilmesi istenen ürünlerin miktarları ile birlikte varsa özellikleri de belirtilerek emir şeklinde depoya iletilmesidir.

2. Siparişin depo içi stoklardan reserve edilmesi:

   Gelmiş olan siparişin belirli bir politikaya bağlı olarak depo içerisindeki stoklar ile eşleştirilmesi ve belirlenen stokların o sipariş için reserve edilmesi işlemidir. Bu rezervasyon sırasında depo personeline yapılmak üzere görevler oluşur.

3. Toplama işlemi:

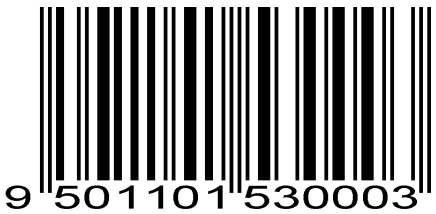
   Depo içerisinde reserve edilmiş ürünlerin görev şeklinde kullanıcılara bildirilmesi ve kullanıcıların bu ürünleri o sipariş için depo içerisinden alıp sevkiyata hazırlama aşamasını ifade eder.

4. Yükleme ve sevkiyat işlemi:

   Depo içerisinde siparişler için hazırlanmış olan stokların depodan çıkışı sağlayacak araçlara yüklenmesi ve depodan çıkarak sistem üzerinde stokların düşümünün yapıldığı aşamadır.

4. STANDART BARKOD SİSTEMLERİ İLE DEPO YÖNETİMİ UYGULAMALARI

Günümüzde birçok ürünün üzerinde görüleceği üzere, farklı şekillerde ve farklı fiziksel yapılarda olan barkodlar bulunmaktadır. Bu barkodların yapısal farklılıkları, kullanım alanları ve tercihlere göre değişkenlik gösterebilmektedir. Çoğunlukla tercih edilen barkodlar UPC,EAN(Şekil 4.1) ve QRCODE(Şekil 4.3) türündekilerdir. Barkod belirli bir kombinasyona sahip, kalınlık, bulunduğu konum gibi bilgilerle bir değerin görselleştirilmiş hali olarak düşünülebilir. Barkod çeşitliliği ile beraber bu barkodlarda gizlenen bilgilerin saklayabileceği bilgi miktarı da artmıştır. Depo yönetim sistemlerinde her bir barkod kendisi ile tekil ilişkiye sahip ürün malzeme ağacındaki bir ürüne tekabül eder. Ürünlerin takibi amacı ile ürün, lokasyon, üretim tarihi, son kullanım tarihi ve parti numarası gibi ihtiyaçlar doğrultusunda şekillenen muhtelif detaylara hızlı erişim amacı ile ilgili barkodun içerdiği tekil numara ya da alfanumerik içerik, erişim anahtarı olarak kullanılır. Bu barkodun içerdiği bilgi sayesinde ürüne ait tüm stok hareketleri, ürün geçmişi gibi konular kolayca kayıt altına alınabilir ya da sorgulanabilir. Uzun yıllardır kullanılan bu yöntem, kurumsal kaynak planlaması ya da malzeme kaynak planlaması gibi uygulamalarda hazır paketler olarak müşterilere sunulmaktadır.



Şekil 4.1. UPC/EAN Barkod

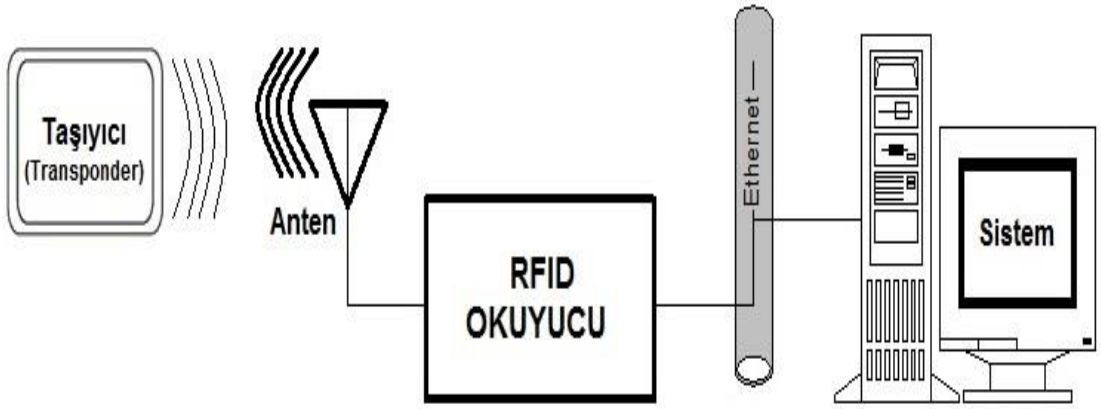
Şekil 4.2. QR CODE

Şekil 4.2'te, QR CODE örneği gösterilmektedir. QR CODE, Japonya'da faaliyet gösteren ve Toyota'nın bir yan kuruluşu olan Denso Wave firması tarafından geliştirilen 2 boyutlu bir barkod sistemidir. Adını ingilizcede Çabut Tepki anlamına gelen Quick Response kelimesinin baş harflerinden alır. İçeriği bir metin, web sitesi adresi, video link dahil herhangi bir veri olabilir. QR Code okuyucu bir yazılım vasıtasıyla bir çok donanımda rahatlıkla okunabilir ve okunacak bilginin türüne göre servis tetikleyebilir.

5. RFID SİSTEMLERİ İLE DEPO YÖNETİMİ UYGULAMALARI

5.1. RFID Teknolojisi

RFID, bir nesne veya kişiye ait tanıma bilgisini (benzersiz seri sayı biçiminde) kablosuz bir şekilde radyo dalgaları ile iletmek için kullanılan sistemleri tanımlamak amacıyla ifade edilen genel bir terimdir. RFID sistemleri okuyucu anten ve etiket olmak suretiyle iki temel bileşen üzerine kurgulanır. Okuyucu diye adlandırılan yapı kendi enerjisini olan bir anten şeklinde kablosuz yayın yapan cihazlardır. Etiket olarak adlandırılan yapı aktif, pasif ve yarı pasif şeklinde üç sınıfa ayrılır. Aktif etiket kendi enerjisine sahiptir fakat pasif etiket okuyucunun enerjisine ihtiyaç duyar. Tüm bu etiketler üreticileri tarafından kendilerine atanmış tekil numaraya sahiptirler. RFID sistemlerin iki temel bileşeni olan okuyucu ve taşıyıcıların hareketli veya sabit olma durumlarına göre okuyucu sabit taşıyıcı hareketli ve taşıyıcı sabit okuyucu hareketli sistemler olmak üzere iki başlık altında toplanmıştır.

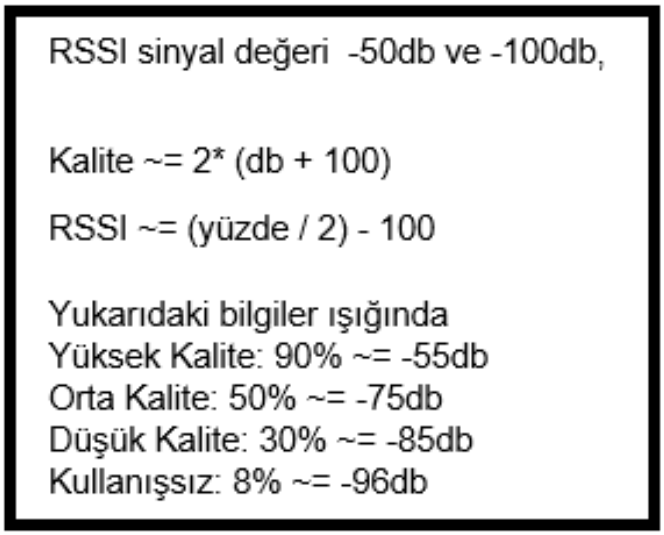


Şekil 5.1. RFID Sistem Bileşenleri

5.2. 2 Boyutlu Düzlemde RFID Konum Belirleme

Günümüzde konumlandırma amacı ile genellikle GPS teknolojisi kullanılmaktadır. Bu teknoloji oldukça yaygın olmasına rağmen hem maliyetleri hem de kapalı mekanlarda işe yaramaması sebebi ile farklı yaklaşımlar araştırılmakta ve uygulanmaktadır. Bu yaklaşımlardan birisi olan RFID ile konum belirlemede amaç etiketin sabit okuyucuya olan uzaklığı ile kestirimlerde bulunmaktır. Antenleri doğası gereği yayın yaptıkları sinyal tek bir nokta odaklı olmadığından ötürü bu tip bir yaklaşım ancak ve ancak birden çok antenin farklı noktalara konumlandırılması ile başarıya ulaşabilmektedir. Çalışmalarda

gözlemlenen RSSI verisine dayalı mesafe ölçümü benimsenmiş bir yöntemdir. RSSI(Received signal strength indication) bir istemci cihaza alınan sinyal kalitesini ölçmek için kullanılan bir terimdir. Ancak bu değer mutlak değere sahiptir. IEEE 802.11 standartlarında her yonga üreticisi kendi "RSSI_Max" değerini tanımlamaktadır. Bu değer 0-255 arasında olabilir. Örneğin; Atheros 0-60 arasını kullanırken, Cisco 0-100 arasında kullanmaktadır. Genel olarak tüm üreticilerde 0 a yakın değer daha yüksek kaliteyi ifade etmektedir yani -40 değeri -50 değerinden daha kıymetlidir. RSSI değeri hesabı yüzdesel dilim üzerinden yapılır, çünkü RSSI değeri kullanıldığı yere göre kabuller üzerinden hesaplanır. Kalite ve RSSI ilişkisi Şekil 5.2.1 de açıklanmıştır. Son zamanlarda gelişen teknoloji ile beraber kablosuz haberleşme ve sensör teknolojileri de hızla gelişmektedir. RSSI bilgisi ek bir donanım gerektirmeden bir çok cihaz ile birlikte sunulan bir özellik olarak yerini almaktadır. Konum belirlemede farklı konumlandırılmış ve birbirlerine olan vektörel uzaklıkları bilinen okuyucu antenlerin her biri tarafından alınan RSSI değeri ile farklı algoritmalar kullanılarak çıkarımda bulunulabilir. Örneğin; ortamda belirli konumlara önceden yerleştirilen taşıyıcılardan alınan RSSI değerleri ile konumu belirlenmeye çalışılan nesnelerin üzerindeki taşıyıcılara ait RSSI değerleri karşılaştırılarak k-NN (k-NN: K Nearest Neighbors / k En Yakın Komşuluk) algoritması ile konum tahminleri gerçekleştirilmiştir . Bir diğer konum belirleme yöntemi TdoA(Time Difference of Arrival/ Varış Süresi Farkı) yöntemidir. Bu yöntem okuyucuların almış olduğu sinyallerin arasında geçen süre hesaplamasıdır. Fakat çalışmalar göstermiştirki bulunan ortamdaki değişkenler ve işlem yapan mikro denetleyicinin oluşturduğu zaman farklılıkları hata payını çok arttırmaktadır. Tüm bu çalışmalar temelde nesnenin 2 boyutlu düzlemdeki konumunu vermektedir. Farklı tip çalışmalar Tablo 1 deki gibidir.



Şekil 5.2. RSSI Kalite Hesabı

Çizelge 5.1. Hesaplama Yöntemlerine Göre Doğruluk Oranları

Referans	**Kullanım Yöntemi**	**Doğruluk**
SpotOn (J.Hightower vd. 2000)	RSSI değerleri kullanılarak üçgenleme methodu ile konum kestirme	3 m
(Bechteler ve Yenigün 2003)	3 okuyucu ile RSSI değerleri kullanılarak üçgenleme methodu ile konum kestirme	Ortalama 20 cm
(Stelzer vd. 2004)	TDoA (Time Difference of Arrival / Varış Süresi Farkı) Ağırlıklı ortalama kareler	10 m
(J. Zhou ve J. Shi 2011)	Multilaterayon yöntemi 3' ten fazla okuyucu ile daha hassas konum tahmini yapılmıştır.	0,0524 m 0,053 m
LANDMARC (Ni vd.2004)	Referans taşıyıcılar yerleştirilmiş ve k-NN algoritması kullanılmıştır.	2 m

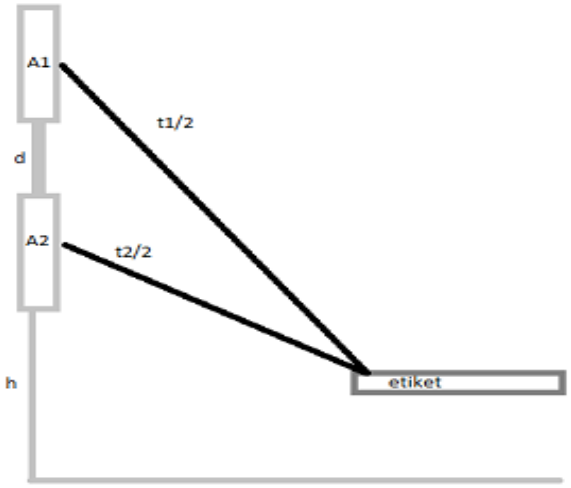
5.3. 3 Boyutlu Düzlemde RFID Konum Belirleme

2 boyutlu düzlemde saptanan konumlar düz alanlarda kullanıma elverişli olsa da gerçek hayat şartlarında özellikle depolarda yer alan raf yapıları için doğru konumun kestirimi 3. Boyut ihtiyacını doğurmaktadır. Raf yapıları depoda maksimum verimlilik amacı ile ilgili ürün gruplarının bir arada tutulduğu ve

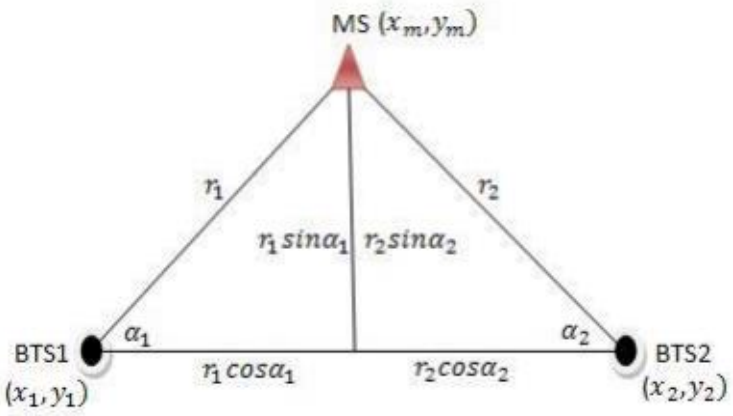
alandan tasarrum amacı ile kullanılır. Etiketli ürünün bulunduğu konumda raftaki yüksekliğinin tespiti için gereken asıl unsur ilgili üçgenlemede yer alan antenlerin üstüne birer anten yerleştirmekle mümkün olabilmektedir. Burada amaç her iki antenin aynı etiketten okuduğu sinyalin açısına göre geliş açısı yada geliş zamanı ile hesap yapmaktır. TOA(Time of Arrival/Geliş zamanı) aynı dikey konumdaki ve farklı yükseklikteki antenlerin okumuş olduğu sinyalin geliş sürelerinin hesaplanması açısal değeri ortaya çıkartmaktadır. İlgili tüm antenlerde yer alan bu ilave anten sayesinde etikete sahip nesnenin konumunu kestirmek mümkündür.

5.3.1. TOA(Time of Arrival/Geliş Zamanı) metodu

TOA, okuyucu antenden gönderilen radyo sinyalinin etikete ve etiketten tekrar okuyucu antene gönderilmesiyle geçen gidiş dönüş süresidir. Arada geçen süre gidiş ve geliş olduğundan süre hesabı t/2 olarak hesaplanır. Mesafe bu sürenin ışık hızıyla çarpımından bulunur. Etiketin açısı iki adet zamansal senkronizasyonu tamamlanmış anten aradıcığı ile bulunur. Antenlerin nesneye olan uzaklıkları ve antenler arasındaki uzaklık hesaba katıldığında nesnenin konumu hesaplanmış olur. Şekil 2 de tariflenen yerden belirli yükseklikte ve bu bilgiye sahip olduğumuz antenlerin arasındaki mesafe ve antenlerden gönderilen ve alınan sinyallerin geliş sürelerinin yarısı ile mesafe kestirimleri yapılabilmektedir. TOA da iki anten arasındaki uzaklık ve antenlerin yerden yükseklikleri baz alındığında antenler e geliş süresi farkı açıyı bulmayı sağlar. Ortaya çıkan bu açı değeri bizim için AOA(Angle of Arrival/Geliş Açısı) dır. AOA da mesafe ölçümü Şekil 3 de tariflenmiştir. Burada doğruluğu arttırmak adına 2 den fazla anten konumlandırma yöntemleri kullanılmaktadır. Her bir anten matrisinin gönderdiği sinyal kendisine özgü olmak zorundadır, çünkü aynı sinyal karışıklığa sebep olacaktır. Antenlerin farklı tip sinyal göndermesi halinde birbirlerinin sinyalini gürültü olarak tanımlayacak ve sadece kendi sinyallerini kullanacaklardır. Bu tip bir işlemde doğruluğu arttırmak için mutlaka sinyal işleme süreleri ve ilgili ortamda ki materyal yoğunluğu göz önüne alınır ve buna göre hata payı minimuma indirilebilir.



Şekil 5.3. RFID TOA Kurgusu



Şekil 5.4. RFID AOA Kurgusu

6. STANDART BARKOD SİSTEMLERİ İLE RFID SİSTEMLERİN KARŞILAŞTIRILMASI

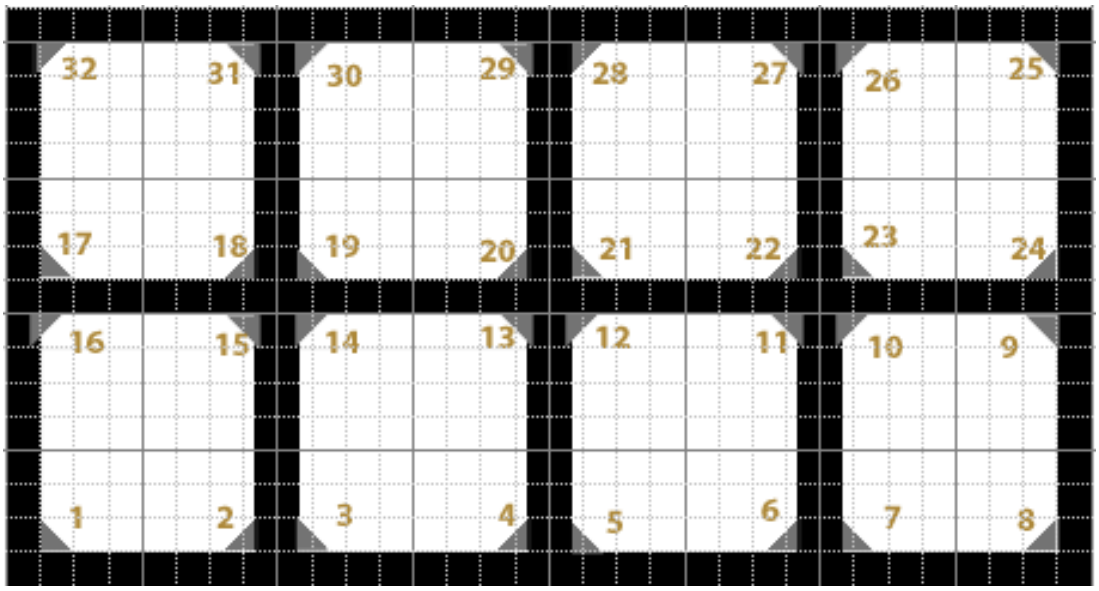
Her iki sistemin karşılaştırmasında Gemi İnşaa sektöründe faaliyet gösteren bir kurumun deposu baz alınarak yapılmıştır. Bu depoda ürün çeşitliliği, ürün adetleri ve stok devir hızı baz alındığında ortalama yıllık 600.000 farklı kalem için değerlendirme yapılmıştır. Standart barkod belirli bir materyalden imal edilen yüzeye Bölüm 2 de de tariflendiği gibi farklı formatlarda işlenen tekil numara bilgisidir. Burada kullanılan materyallerin ömürlerine göre maliyetler farklılık göstermektedir. Ancak dokümanın amacı kapalı ortamlarda stok takibi olduğu için basit kağıt olarak kullanılan barkod etiketi baz alınmıştır. Bir üründe kullanılan barkod tekrar kullanılamamaktadır. Bu sebeple çevresel olarak zararları düşünüldüğünde 600.000 adet kağıt parçasının geri dönüşümsüz olarak kullanımı çok ciddi zararlara sebep olmaktadır. İlgili barkodların ürün ile ilişkilendirilip yazdırılması için gerekli olan yazıcının vermiş olduğu toksik zararlar da göz önünde bulundurulduğunda çevresel anlamda faydalı bir ürün olduğunu söylemek mümkün değildir. Maliyetleri bakımından değerlendirildiğinde her bir barkod etiketi 0.02$ a tekabül etmektedir. Bu barkodların yazdırma maliyetleri ile beraber toplam değeri 0.03$ dır. Bahse konu tersane yıllık olarak sadece yazdırma işlemine 18.000$ seviyesinde bir bedel ödemektedir. İlgili barkodları okumak için kullanılan cihazların uzun ömürlü olması ve aynı cihazların rfid etike okuma yetenekleri göz önüne alındığında bu donanım her iki sistem içinde cihaz başına amortisman göz önünde bulundurularak 500$ ve depoda çalışan personel sayısı kadar olacağından yıllık toplam 5000$ şeklindedir. Tüm bu donanım ve sarf malzemeleri 23000$ seviyesinde olup bu maliyet her yıl tekrar yenilenmekte ve yükselmektedir.

UHF antenler ile lokasyon takibi için kullanılacak etiketlerin maliyeti 0.06$ şeklindedir. Bu etiketlerin tüm bir yılı kapsayacak şekilde alımında ödenecek tutat 36.000$ dır. Normar barkod etiketinden farklı olarak bu etiketler tekrar kullanılabilir durumdadır ve ömürleri 3 yıldır. Dolayısıyla ilk yatırım bedeli yüksek olsa da kullanım amacı göz önünde bulundurulduğunda yıllık gerçek maliyet 12.000$ seviyesindedir. Çevresel etkisi düşünüldüğünde ise uzun ömürlü olmasının getirdiği büyük bir avantaj bulunmaktadır. Ayrıca rfid etiketlerin

malzemeleri geridönüşüme tabidir. Tersane deposu ölçüleri 80x160 m yani 12.800m2 dir. Şekil 4 de görüleceği üzere 32 adet UHF anten matrisi tüm ihtiyacı karşılar niteliktedir. Her UHF anteni 2 anten matrisinden oluşmaktadır ve ürün 350$ fiyata sahiptir. Toplam anten maliteti 11.200$ dır. Tüm maliyetlerin karşılaştırması Tablo 2 de gösterilmektedir. Tabloda yer alan UHF antenlerin her biri aslında bir anten matrisini tarif etmektedir çünkü bu antenlerin 3. Boyut hesabı için 2 adet anten yongası içermeleri gerekmektedir.

Çizelge 6.1. Yıllık Maliyet Tablosu

ÜRÜN	**ADET**	**BİRİM FİYAT**	**TOPLAM FİYAT**
UHF ANTEN	32	350$	11.200$
RFID ETİKET	200.000	0.06$	12.000$
RFID OKUYUCU	10	500$	5.000$
BARKOD OKUYUCU	10	500$	5.000$
BARKOD ETİKET	600.000	0.03$	18.000$
RFID SİSTEM TOPLAM=28.200$			
BARKOD SİSTEM TOPLAM=23.000$			



Şekil 6.1. UHF Anten Konumlandırma Haritası

7. RFID İLE KONUM BELİRLEMEDE KULLANILAN TEKNOLOJİLER

7.1. Deneysel Çalışma Ortamı

Bu çalışmada hedeflenen antenlerin sağlıklı bir şekilde konumlandırılmasından öte anten ve sunucu arasındaki haberleşmenin asnkron olarak sağlanması ve RSSI değerinin ölçümüdür. Antenlerin sinyalleri simule edilmiş olup anten değerleri rastgele oluşturulmuştur. 3 adet ARDUINO UNO R3 antenlerden gelen sinyali ve gerekli diğer bilgileri işlemek için sunucya taşıyan MQTT Publisher görevini üstlenmiştir. ARDUINO ve Sunucu arasında iletişimi sağlayacak olan ethernet protokolü için Ethernet Shield kullanılmıştır. Ethernet sonlanma noktası olarak TP-Link 1Gb bant genişliğindeki switch ve 3 adet CAT6 kablo kullanılmıştır.

Uygulamanın donanımsal kısmının geliştirilmesinde ARDUINO IDE kullanılmış ve bu ide vasıtasyla C++ dilindeki kütüphanelerden faydalanılmıştır.

Sunucu bacağında yer alan uygulama için Microsoft.NET 4.5 framework'ü üzerinde C# dili ile executable bir uygulama yazılmıştır. Her iki platform hakkında detaylı yazılım bilgisi yazılım detayları başlığı altında değerlendirilmektedir. İki platformun da iletişimini sağlamak amacı ile RabbitMQ ile mesajlaşma yöntemi tercih edilmiş olup ayrıca veri saklamak amacı ile NoSQL bir veritabanı olan MongoDB tercih edilmiştir. MongoDB tercih sebebi saklanacak verinin yalınlığı ve çok hızlı cevap verebilecek şekilde tasarlanmasıdır. RabbitMQ ve Mongo tercihleri yapılan test çalışmaları ile kararlaştırılmıştır.

7.1.1. Kullanılan programlama dilleri, uygulamalar ve donanımlar

Uygulama geliştirme sürecinde donanımsal ve yazılımsal ihtiyaçların karşılanması adına aşağıdaki platformlardan faydalanılmıştır.

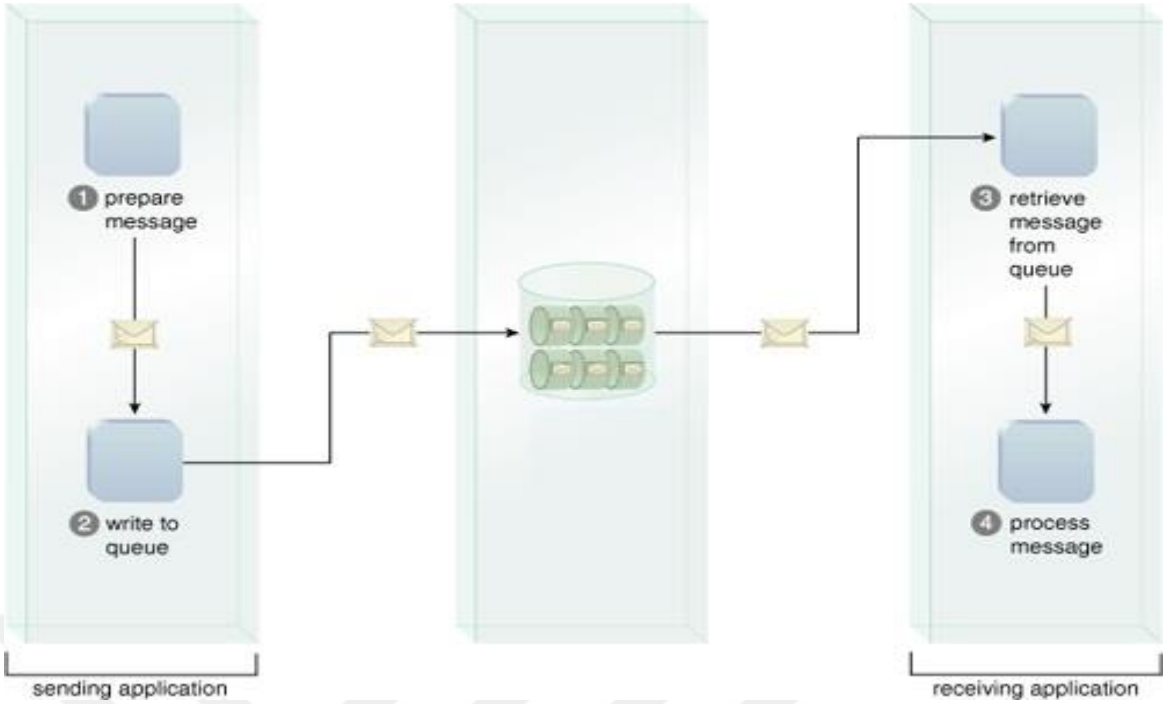
- C# .NET
- Arduino IDE(C++)
- MongoDB
- Cassandra
- AcrtiveMQ

- RabbitMQ
- MQTTClient
- MQTT RabbitMQ Plugin
- JSON

7.1.2 Mesaj kuyruğu

Aralarında sürekli bir bağlantı gerektirmeyen dağıtık sistemlerde uygulamaların emniyetli bir şekilde iletişim kurmalarını sağlayan bir protokoldür. Dağıtık sistemlerde çalışan bir uygulama mesaj kuyruğu üzerinde kuyruğa bir mesaj gönderir ve sıradaki işlemlerden kendi iç sürecini işletmeye devam eder. Sistemde bu mesaja ihtiyacı olan diğer bir uygulama bu kuyruğa giderek ilgili mesaj varsa alır ve kendi sürecini başlatır. Aşağıdaki resimde bu durum bir grafik üzerinde ifade edilmiştir. Bu sayede birbirinden bağımsız çalışan iki ayrı sistem, birbirlerini beklemeden görevlerini yerine getirebilmektedir.

Örneğin bir web sitesi üzerinde yapılan işlemler sonrasında kullanıcı bilgilendirme amaçlı E-Posta gönderiliyor olsun. E-Posta gönderme işlemi ayrı bir mail sunucu üzerinden yapılacaktır. Kullanıcı, bir üyelik formunu doldurup gönderdikten sonra sürecin tamamlanmasını beklemeye başlar. Bu aşamada form bilgileri veri bankasına yazılır ve kullanıcıya bir bilgilendirme mesajı gönderilir. Ancak E-Posta sunucusu geç yanıt verirse veya arızalı durumda ise kullanıcının kayıt süreci yarım kalacaktır ve eksik bir işlem yapılmış olacaktır. Bunun sebebi web uygulamasının, ayrık bir sistem olan E-Posta sunucusuna bağımlı olmasıdır. Bu gibi durumları garanti altına alabilmek için kullanıcıya gönderilecek olan E-Posta bilgileri kayıt sürecinde MSMQ sistemi üzerinde kuyruğa eklenir. Bir başka uygulama bu kuyruğu kontrol ederek E-Posta gönderme işlemlerini gerçekleştirir. Yani iki ayrık sistem birbirinden izole hale getirilerek çalışma süreçleri birbirini etkilememiş olur.



Şekil 7.1. Mesaj Kuyruğu

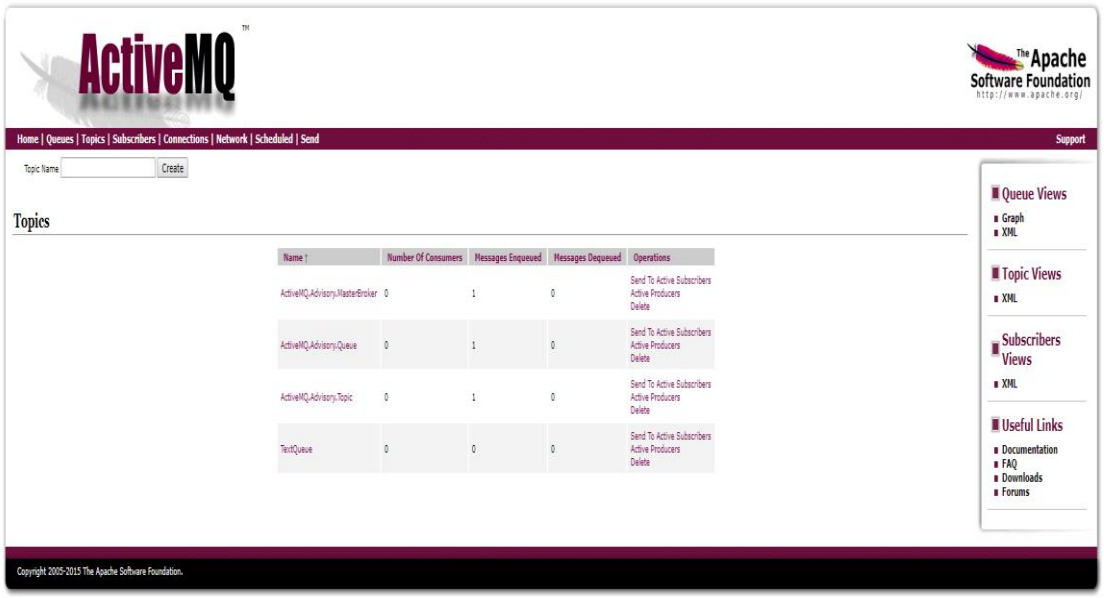
7.1.2.1 ActiveMQ ile mesaj kuyruğu

Büyük projelerde sisteminiz çeşitli alt modüllerden hatta çeşitli alt uygulamalardan oluşacaktır. Eğer projeniz alt modüllerden oluşuyorsa, modüllerin haberleşmesini sağlamak göreceli olarak daha kolaydır. En basitinden bir Observer Pattern ile modüllerinizin haberleşmesini sağlayabilirsiniz. Fakat projeniz alt modüllerden değilde alt uygulamalardan oluşursa Observer pattern gibi yapılar sizi kurtarmayacaktır. Bu gibi durumlar için çeşitli çözümler üretilmiştir. Bu tip çözümler yazılım alanında prosesler araşı haberleşme (Inter process communication) başlığı altında incelenmektedir. Mesajlaşma kuyruğu ise bu çözümlerin bir tanesidir.

Genel olarak her şirket prosesler arası haberleşme için kendi çözümünü kullanmaktadır. Kimi bellek alanları üzerinden haberleşme yaparken kimi kolaya kaçıp veritabanı ya da dosyalar üzerinden bu haberleşmeyi sağlamaktadır. Kimi de Mesaj kuyrukları kullanmaktadır. Java içerisinde mesajlaşma ile haberleşme için JMS API'ı geliştirilmiştir.

ActiveMQ bu standart baz alınarak oluşturulmuş bir JMS kütüphanesidir. Genel olarak sistemden ayrı şekilde kurulmu yapılarak kullanılır. Yani nasıl

veritabanını ayrı bir sunucuda ya da aynı makinede bile olsa ayrı bir proses içerisinde ayağa kalkıyorsa aynı şekilde ActiveMQ'nunda ayrı bir proses ile ayağa kaldırılması gerekir. Tabi bu bahsettiğim yoğun şekilde **ActiveMQ** kullanıldığı yani bir çok prosesin farklı ya da aynı kuyruklar üzerinden haberleştiği durumlar için geçerli.



Şekil 7.2. ActiveMQ Mesaj Kuyruğu

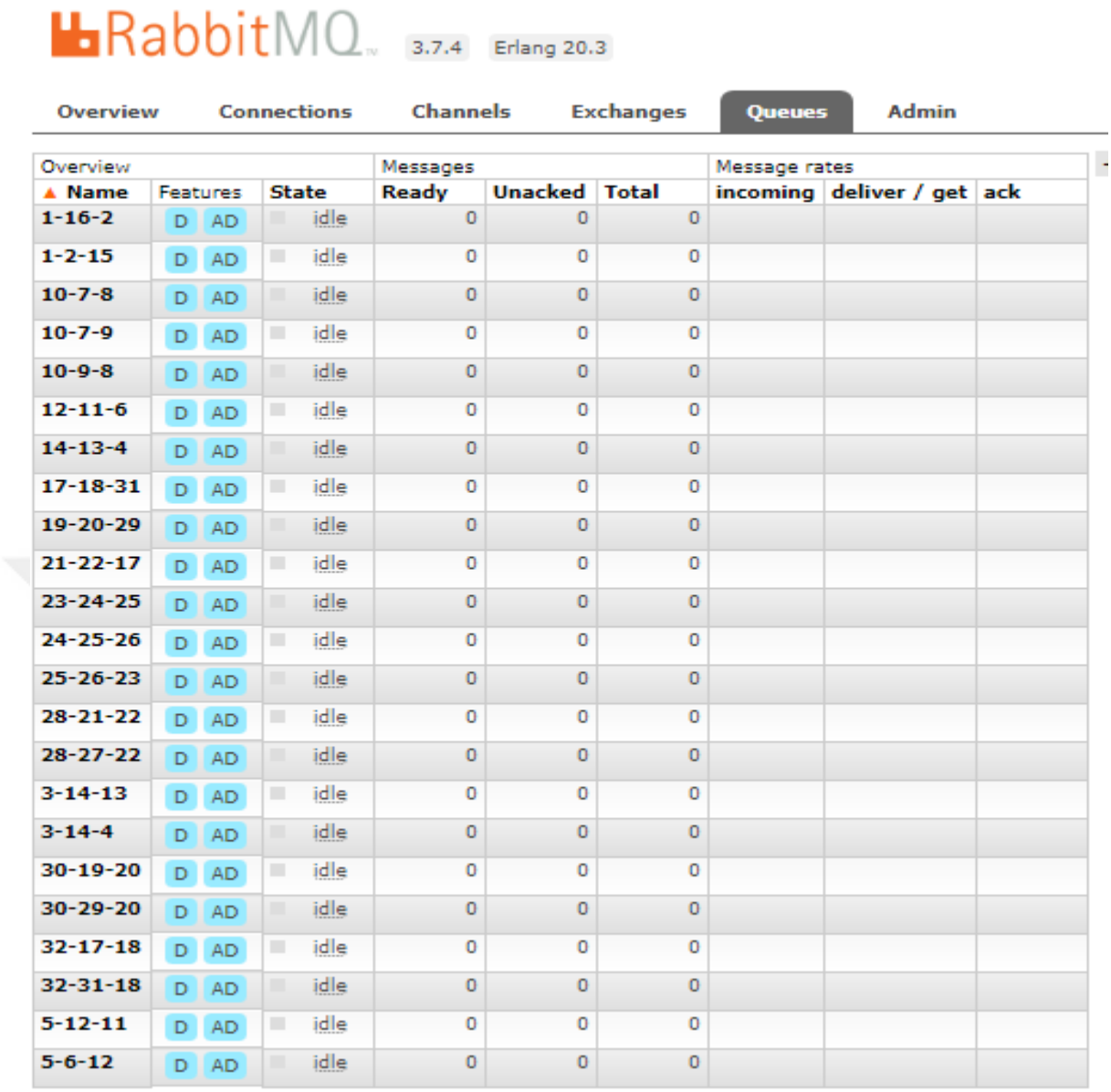
7.1.2.2 RabbitMQ ile mesaj kuyruğu

RFID ile konumlandırma yaparken sadece yalın veri kayıt altına alınmayacaktır. Toplanan tüm yalın veriler işleme tabi tutulduklarında anlamlı hale gelecektir. Örneğin tek başına bir antenin almış olduğu sinyal ve buna bağlı RSSI değeri bizim için hiçbir sadece nesnenin o antene olan uzaklığı hakkında kabaca bir bilgi verecektir. Bilgiler gruplar halinde ve işlenmiş olduklarında anlamlıdır. Bilgilerin işlenmesi ve kullanıcıya gösterimi projenin başka farklı katmanlarıdır. IoT cihazları projenin bu aşamasında devreye girmektedir. Cihazların amacı antenlerden alınan yalın bilginin bir veri tabanına kayıt altına alınmasını sağlamaktır. Bu proje için geliştirm amaçlı ARDUINO UNO R3 cihazı tercih edilmiştir. Cihazın üzerinde hazırda yer almayan bileşen ise Ethernet Portu olan ARDUINO ETHERNET Yongasıdır. Uygulamanın yapılacağı ortamda UHF frekanslarında kirliliğe sebep olmaması için sunucu ile aradaki iletişimde kablosuz bağlantı tercih edilmemiştir. IoT cihazlarının doğası ve amacı gereği

işlem yapma yetenekleri düşüktür. Bu tip cihazlar ile işlem yapmak ve verileri anlamlı hale getirmek işlem zamanının yaratacağı trafik göz önünde bulundurulduğunda tercih edilmemelidir. Bir depo içerisinde yer alan her bir UHF antende bu donanımın eklenti yapılacağı düşünüldüğünde verinin miktarı oldukça fazla olacak ve süreklilik göz önüne alındığında trafik bir süre sonra darboğaza girecektir. Bu gbi sorunlar göz önünde bulundurulduğunda ARDUINO cihazların yapması gereken işlemleri kapasitesi yüksek ve özellikle iş zekası yeteneklerine sahip sunucular vasıtası ile yapmak daha anlamlı olacak ve cihazların yine doğası gereği ARDUINO cihazların ömürlerini uzatacaktır. Bu durumda ARDUINO cihazlar sadece antenden alınan verinin sunucuya ulaşmasını sağlamak ile yükümlüdür. Günümüzün gelişen teknolojileri sadece donanım değil aynı zamanda yazılım anlamında da ilerlemektedir. Endüstri 4.0 ile hayatımıza kavramsal olarak giren nesnelerin interneti teknolojilerinde yükü hafifletilmiş veri mimarileri kullanılmaktadır. Bu konu MQTT, AMQP ve karşılaştırmaları bağlığı altında detaylıca açıklanmaktadır. Bu proje çerçevesinde bizim tercihimzi MQTT altyapısının esnekliği sebebi ile tercih sebebidir.

MQTT aracılığı ile veriler sunucuya direk olarak akabilir fakat bunu yapmamız halinde her işlemin sonucunu beklememiz ve zaman faktörü için antenlere bağlanan ARDUINO cihazlarda zamandan kazanmamız mümkün olmayacaktır. Bu tip durumlarda ara bir mesaj sıralama katmanı kullanmak her zaman daha karlı olacaktır. Message Queue (Mesaj Sırası) olarak bilinen teknolojiler son zamanlarda bulut teknolojinin hayatımıza girmesi ile önem kazanmıştır. Özellikle nesnelerin interneti cihazlarında oluşan mesajların sıralamasına göre işleme tabi tutulması için tercih sebebi olmaktadır. Araştırmalar ve kullanım kolaylıkları göz önünde bulundurulduğunda RabbitMQ sunucusu bizim açımızdan en uygun yazılım altyapısıdır. RabbitMQ tıpkı diğer örneklerinde olduğu gibi farklı protokoller için farklı portları üzerinde barındıran ve üzerine gelen mesajları duruma göre saklama yeteneğine sahip bir uygulama katmanıdır. Uygulamanın temelinde asenkron mesajlaşmak yatmaktadır. Asenkron mesajlaşmanın bizim açımızdan en büyük artısı ARDUINO cihaz tarafından iletilen bir mesajın sonucunun yine aynı cihaz tarafından beklemesine gerek olmamasıdır. Cihaz sonuç almadan yoluna devam etmek istese dahi yazılımların doğası gereği belirgin bir işlem süresine ihtiyaç duyulmaktadır. RabbitMQ bizim için öncelikli

olarak bu yeteneği ile tüm antenlerden gelen mesajları sıraya koyacak ve bu sıradan FIFO kuralına göre çekerek kayıt altına alınmasını sağlayacak olan dinleyiciye iletecektir. RabbitMQ da mesajın içeriği veya anlamı tamamen geliştiricinin insiyatifindedir. RabbitMQ temelde AMQP protokolünü kullanırken bizim uygulamamızda MQTT protokolü aktifleştirilmiştir. Bu protokolün tercih sebebi IoT cihazları için uygunluğu ve esnek yapısıdır. MQTT bilindiği üzere TOPIC/BODY/QOS LEVEL olarak üç bilgi ile mesajlaşmayı sağlamaktadır. Bu yapısı bizim açımızdan temel bir sorunu ortaya çıkartmaktadır. Bu sorun Antenlerin gruplanma ihtiyacı ve bu ihtiyaca bağlı olarak mesaj kuyruğuna girmeleridir. Her bir anten grubunun kendi kuyruğunda olması dinleyici uygulama tarafından kayıt metodunu daha verimli olarak yapabilmesini sağlayacaktır. Burada bahsedilmek istenen resimde tariflenen antenlerin her birinin kendi içerisinde kuyruğa alınması ve bu kaide ile incelenmesidir. RabbitMQ mesaj kuyruğu mantığı ile çalışır ve kuyruklar dağıtıcı tarafından oluşturulan yada tercih edilen grub algoritmasıdır. Kuyrukları ARDUINO cihazların oluşturmasını beklemektense yapımızda her bir kuyruk özel olarak adlandırılarak RabbitMQ nun yaması olan Erlang uygulaması üzerinden oluşturulmuştur. Bu sayede her kuyruk kendine has antenlerden veri alacak ve bunu ise kendi grup sıralamasında dinleyiciye iletebilecektir.



Şekil 7.3. RabbitMQ Kuyruk Yapısı Anten Grupları

7.1.2.3 RabbitMQ ve ActiveMQ karşılaştırması

Her iki platform da temelde aynı amaca hizmet etse de kullanılan yazılım teknolojileri ve performanslarında değişimler görülmektedir. İki platformda MQTT protokolünü desteklemektedir. Çizelge 7.1. genel hatları ile iki platformu da karşılaştırmaktadır. Yapılan çalışmada iki platform için aynı mesaj datasının kuyruğa aktarılma süreleri asenkron ve multitask olarak 100 kayıt üzerinde denenmiştir. Denemelerin sonucunda görülmüştürki ilk bağlantı açılma süresi her iki platform için zaman alıcı olmuştur. Bunun haricinde ortaya çıkna sonuçta milisaniye seviyesindedir. Her iki ortam için denenen ve mesaj aşağıdaki gibidir.

{"AntenGrup":"AG1","RSSI":"87","AntenNumarasi":"AG1_2041175501","Barkod":"326D423D-B2E1-4F61-92F0-0B8C00471D14","OkumaTarihi":"2018-05-19T17:26:56.1363464+03:00"}

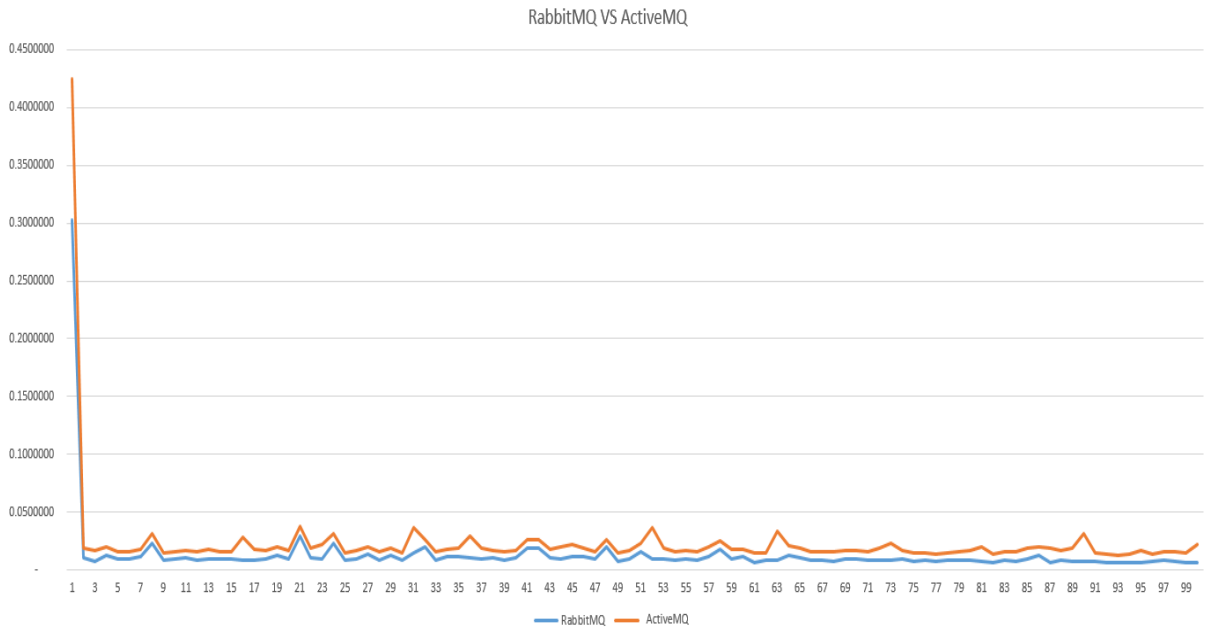
Mesajların distributer uygulama metodları Şekil 7.4. ve Şekil 7.5. da gösterilmektedir. Kuyruğa mesaj gönderme süreleri ise Şekil 7.6. de sonuçlar 100 iterasyon içinde RabbitMQ nun başarılı olduğunu göstermiştir. Uygulamamızda RabbitMQ nun teknik avantajları cloud desteği ve zamansal anlamda göstermiş olduğu başarıları yönünden uygun bulunmuştur.

```
public static void SendNewMessageActiveMQ()
{
    for (int i = 0; i < 100; i++)
    {
        Stopwatch stopwatch = new Stopwatch();
        stopwatch.Start();
        string topic = "TextQueue";
        string guid = Guid.NewGuid().ToString();
        Console.WriteLine($"Adding message to queue topic: {topic}");
        Uri uri = new Uri("activemq:tcp://localhost:61616");
        string brokerUri = $"activemq:tcp://localhost:61616";
        Apache.NMS.IConnectionFactory factory = new Apache.NMS.ActiveMQ.ConnectionFactory(uri);
        using (Apache.NMS.IConnection connection = factory.CreateConnection())
        {
            if (!connection.IsStarted)
                connection.Start();
            using (Apache.NMS.ISession session = connection.CreateSession(Apache.NMS.AcknowledgementMode.AutoAcknowledge))
            using (Apache.NMS.IDestination dest = session.GetTopic(topic))
            using (Apache.NMS.IMessageProducer producer = session.CreateProducer(dest))
            {
                producer.DeliveryMode = Apache.NMS.MsgDeliveryMode.Persistent;
                string message = "{\"AntenGrup\":\"AG1\",\"RSSI\":\"" + rastgele.Next(50, 100) + "\",\"AntenNumarasi\":\"AG1_2041175" + rastgele.Next(5, 50)
                    + "\",\"Barkod\":\"" + guid.ToUpper() + "\",\"OkumaTarihi\":\"" + DateTime.Now + "\"}";
                producer.Send(session.CreateTextMessage(message));
                stopwatch.Stop();
                Console.WriteLine($"Sent {message} messages" + " Time Elapsed: " + stopwatch.Elapsed);
                ts.Add(stopwatch.Elapsed);
            }
        }
    }
}
```

Şekil 7.4. ActiveMQ Kodu

```
public static void SendNewMessageRabbitMQ()
{
    for (int i = 0; i < 100; i++)
    {
        Stopwatch stopwatch = new Stopwatch();
        stopwatch.Start();
        _rabbitMQService = new RabbitMQService();
        string guid = Guid.NewGuid().ToString();
        string message = "{\"AntenGrup\":\"AG1\",\"RSSI\":\"" + rastgele.Next(50, 100) + "\",\"AntenNumarasi\":\"AG1_2041175"
            + rastgele.Next(5, 50) + "\",\"Barkod\":\"" + guid.ToUpper() + "\",\"OkumaTarihi\":\"" + DateTime.Now + "\"}";
        using (var connection = _rabbitMQService.GetRabbitMQConnection())
        {
            using (var channel = connection.CreateModel())
            {
                channel.QueueDeclare(queueName, true, false, true, null);
                channel.BasicPublish("", queueName, null, Encoding.UTF8.GetBytes(message));
                stopwatch.Stop();
                Console.WriteLine("{0} queue'su üzerine, \"{1}\" mesajı yazıldı.", queueName, message);
                ts.Add(stopwatch.Elapsed);
            }
        }
    }
}
```

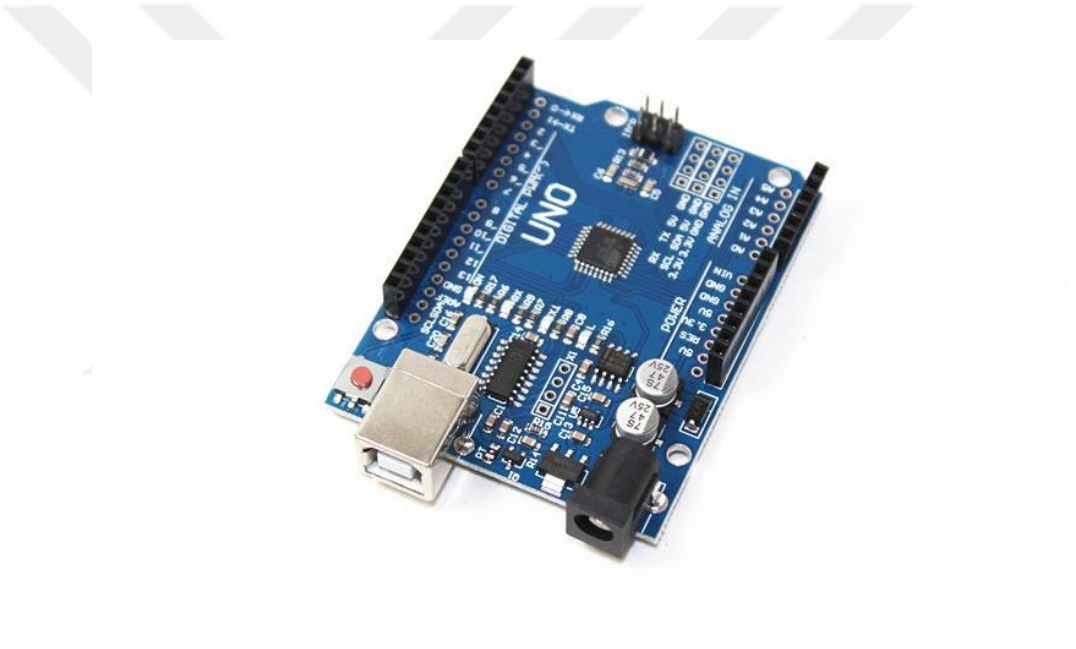
Şekil 7.5. RabbitMQ Kodu



Şekil 7.6. RabbitMQ Zamansal Sonuçları

Çizelge 7.1. RabbitMQ ve ActiveMQ Karşılaştırması

	RabbitMQ	**ActiveMQ**
Hız	Hızlı	Yavaş
Kurulum	Kolay	Zor
Arayüz	Erlang	Default
Standart Protokol	AMQP	AMQP
İlave Arayüz İhtiyacı	Var	Yok
MQTT Desteği	Var	Var
Lisans	Open	Open
Cloud Desteği	Var	Yok



Şekil 7.7. ARDUINO UNO Geliştirme Kartı

7.1.3 NoSQL veritabanı

Aslında NoSQL isimlendirme konusunda biraz kafa karışıklığı oluşturmaktadır. Akla gelen ilk anlamı sql değil anlamındadır. Bu adın kullanılmasının en büyük nedeni yeni bir teknolojinin etkileyici bir şekilde sunulmak istenmesidir. Diğer adları çok tercih edilmemiştir. Diğer adlarından bazıları NonRel, NoRDBMS'dir (ilişkisel değil anlamında). NoSQL için "not only sql" tabiri de kullanılmaktadır. Yani sadece sql değil denilmektedir. Bunun nedeni sql sorgularına alışmış

geliştiriciler olarak aynı sorgu mantığının adapte edilmeye çalışılmasında yatmaktadır.

NoSQL, ilişkisel veritabanı sistemlerine alternatif bir çözüm olarak ortaya çıkan, yatay olarak ölçeklendirilen bir veri depolama sistemidir. İnternetin hızlanması ile beraber, sistemlerde çok fazla kullanıcının aktif rol alması, birçok ihtiyacın yanında veritabanı şemasının artan bir sıklıkla değiştirilmesi zorunluluğunu ortaya çıkardı. Hepimizin kullandığı ilişkisel veritabanlarındaki hiyerarşi, önce tabloları ve her tabloya ait sütunları oluşturmak sonrasında ise bilgileri satır satır eklemektir. Fakat burada karşımıza çıkan bir sıkıntı, tanımı olmayan alana bilgi kaydetmektir. Bu sıkıntıdan kurtulmak için yapmamız gereken öncelikle tabloyu tekrardan ele alıp yeni sütunlar eklemektir. Tabi bununla birlikte veritabanındaki tüm tablo ve ilişkileri etkileyecek durumlar da ortaya çıkabilmektedir. Bu işlemi gerçekleştirmek sistemi tekrardan ele almayı gerektirdiği için çok maliyetli olabilmektedir.

Özetle söylemek gerekirse İlişkisel Veritabanı Sistemlerinde maliyetin yükselmemesi adına yapıyı önceden çok iyi planlamak gerekmektedir. Fakat NoSQL veritabanları sistemlerinde, bilgilerin kayıt altına alınması adına bu yapıyı sonradan kurmak ek bir maliyet getirmemekte ya da az bir maliyet getirmektedir. Nedeni ise NoSQL sistemlerde tablo ve sütun kavramının olmamasıdır. NoSQL sistemlerde yeni bir alana ihtiyacınız olduğu durumlarda kaydı direk eklemeniz yeterli olmaktadır. NoSQL sizin yerinize alanı oluşturur ve değeri kaydeder. Kayıtların maliyetsizce gerçekleşmesinin nedeni ise verilerin tablo ve sütunlarda saklanması yerine JSON ve XML formatına benzer yapıda saklanmasıdır.

Tüm bu anlatımlardan “NoSQL veritabanları, ilişkisel veritabanlarından çokdaha iyidir” gibi bir sonucu çıkarmamamız gerekmektedir. Çünkü ikisini de yerine göre kullanmakta fayda bulunmaktadır. Örneğin: ilişkisel olayların çok yoğun gerçekleştiği bir bankacılık uygulamasında NoSQL bir veritabanı kullanmamız uygun değildir. NoSQL, Fire and Forget prensibi ile çalıştığı için bankacılık, alışveriş gibi para üzerinden işlem yapılan kritik uygulamalarda kullanılmamalıdır. Aksine verinin %100 önem arz etmediği durumlarda kullanımı daha uygundur. Uygulamamızda MongoDB ve Cassandra değerlendirilmiştir.

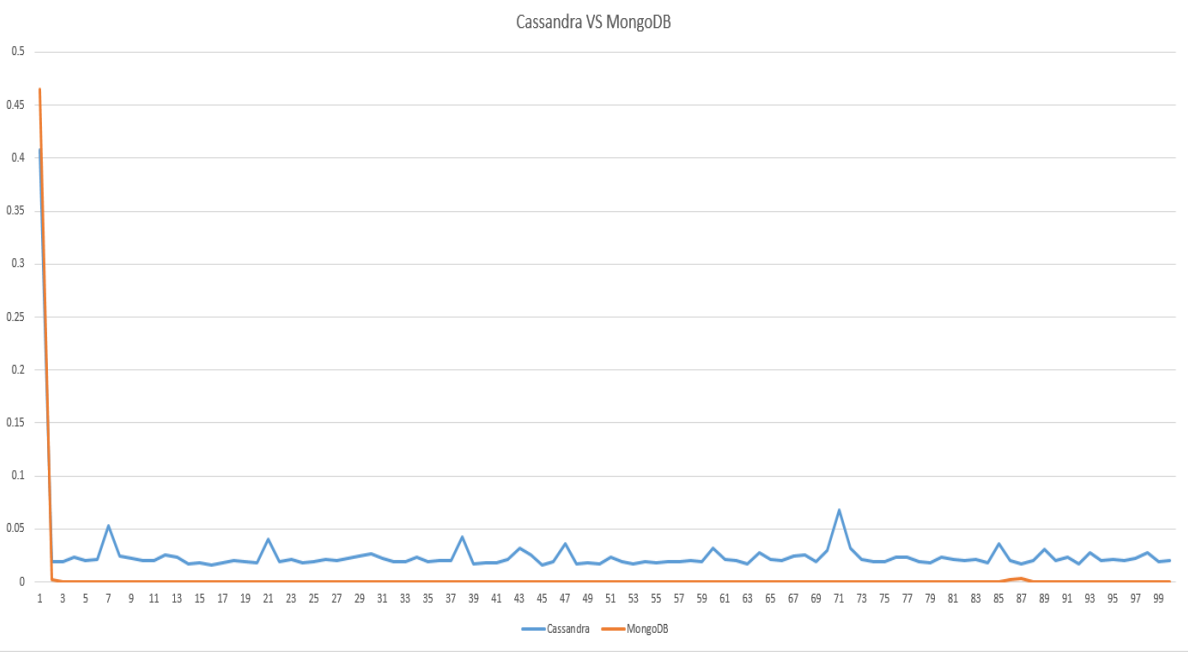
7.1.3.1 MongoDB ve Cassandra karşılaştırması

İki uygulamaya ait karşılaştırma genel kullanım amaçları açısından Çizelge 7.2. de değerlendirilmiştir. 100 iterasyon ile kayıt atma deneyinde zamansal grafik Şekil 7.8. da gösterilmiştir.

Her iki uygulama da aslında benzer amaçlara hitap etsede özellikle kullanım kolaylıkları ve zamansal farklılıklar ve lisans sorununun olmaması nedeniyle MongDB tercih edilmiştir.

Çizelge 7.2. MongoDB ve Cassandra Karşılaştırması

	Cassandra	**MongoDB**
Kullanım Alanları	Gerçek Zamanlı analitik, e-ticaret, sahtekarlık denetimi, müzik katalogları, IoT, Online eğitim, video yayıncılığı	Gerçek Zamanlı analitik, içerik yönetim sistemleri, mobil cihazlar, IoT
Veri Saklama	Kolon bazlı	Dosya bazlı
Kopyalama	Kolay	Zor
Dil	SQL Benzeri CQL	Spesifik dil
Lisans	Enterprice Ücretli	Ücretsiz



Şekil 7.8. Cassandra ve MongoDB Zamansal Karşılaştırması

7.1.4. MQTT nedir?

MQTT MQ Telemetry Transport üzerine kurgulanmıştır. Kısıtlı cihazlar ve düşük bant genişliği, yüksek gecikmeli veya güvenlimez ağlar için tasarlanmış bir yayın/abone protokolüdür. Tasarım prensipleri, ağ bant genişliği ve cihaz kaynak gereksinimlerini en za indirirken aynı zamanda güvenilirliği ve bir dereceye kadar teslimat güvencesini sağlamaya çalışmaktır. Bu ilkeler, protokoln ortaya çokan "Makineden makineye"(M2M) veya bağlı cihazların "Nesnelerin interneti" dünyasına ve bant genişliği ve pil gücünün yüksek olduğu mobil uygulamalar için ideal hale getirilmesiyle sonuçlanmaktadır. MQTT 1999 Yılında IBM ve ARCOM tarafından geliştirilmiştir. MQTT Standart olarak 1883 TCP/IP portunu kullanmaktadır ayrıca, SSL 8883 portu ile tanımlıdır. Bu portlar isteğe göre değişitirilebilmektedir.

7.1.4.1. MQTT çalışma prensibi

MQTT dört farklı çalışma bölümünden oluşur

- Bağlantı
- Kimlik Doğrulama
- İletim
- Sonlandırma

7.1.4.2. MQTT iletişim amaçlı mesaj türleri

CONNECT: Sunucuyla bağlantı kurmak için gönderilen mesajdır.
CONNECTACK: Bağlantı mesajına karşılık gönderilen onay mesajıdır.
PUBLISH: Mesajı yayınlamak için gönderilen mesajdır.
PUBACK: Yayınlanan mesaja karşılık gönderilen onay mesajıdır.
PUBREC: Yayınlama mesajlarının yayınlandığına dair gönderilen mesaj türüdür.
PUBCOMP: Yayınlama işleminin tamamlandığına dair gönderilen mesajdır.
SUBSCRIBE: Abonelik için gönderilen mesaj türüdür.
SUBACK: Abonelik mesajının onayıdır.
UNSUBSCRIBE: Abonelikten çıkmak için gönderilen mesaj türüdür.
UNSUBACK: Abonelik iptal mesajının onayıdır.
PINGREQ: Bağlantı kontrolü için gönderilen mesajdır.
PINGRESP: Bağlantı kontrolü mesajına cevap olarak gönderilen mesajdır.

DISCONNECT: Bağlantı koptuğunda gönderilen mesaj türüdür**.**

7.1.4.3. MQTT genel özellikleri

MQTT genel özellikler aşağıdaki gibi sıralanabilir.

- Asenkron olarak çalışır.
- Broker temelli haberleşme mekanizması üzerine kurgulanmıştır.
- Konuya(Topic) dayalı adresleme vardır.
- Güvenlik olarak SSL/TLS destekler.
- En düşük kaynak kullanımını hedefler.
- TCP/IP bağlantı türünü kullanır.
- Varsayılan port 1883'tür.

7.1.4.4. MQTT de güvenlik

V3.1 içeriğinde MQTT ye basit mantıkla bir kullanıcı adı ve şifre ile bağlanmak mümkündür. Şifreleme ise 8883 portu üzerinden SSL ile sağlanabilmektedir. Tüm bunların dışından yayınlanan tüm içerikler özel olarak şifrelenebilir ve şifreler istemcilerde çözümlenebilir. Bu esnek yapısı sayesinde güvenlik ön plandadır.

7.1.4.5. MQTT de servis kalitesinin garanti altına alınması

MQTT protokolü için 3 farklı kalite garantisi "QoS" ilkesinden bahsetmek mümkündür. Kalite amacı güvenilmez veya zayıf ağlarda bilginin istemci(ler) üzerine ulaşmasının garanti altına alınması yada tam aksi için geçerlidir.

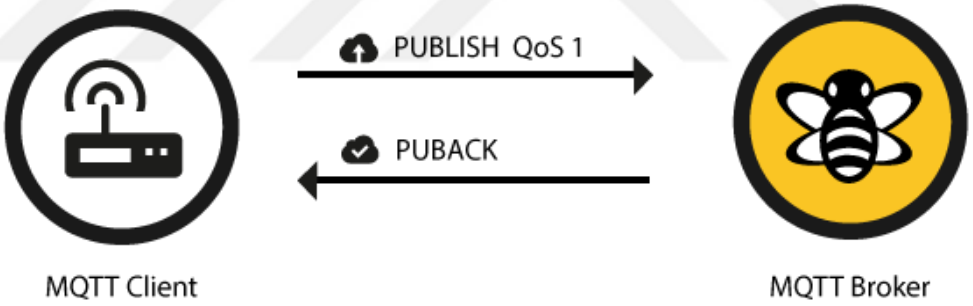
7.1.4.5.1 QoS 0

Minimum kalite seviyesi olan QoS 0 da amaç en fazla bir istemciye erişimin sağlandığını garanti etmektir. Burada asıl söylelen hiçkimseye ulaşmasa da olur,çünkü iletişim tek yönlüdür ve geri besleme söz konusu değildir. En güvenilmez yayındır fakat en hızlı olanıdır.

Şekil 7.9. QoS 0 Anlatımı

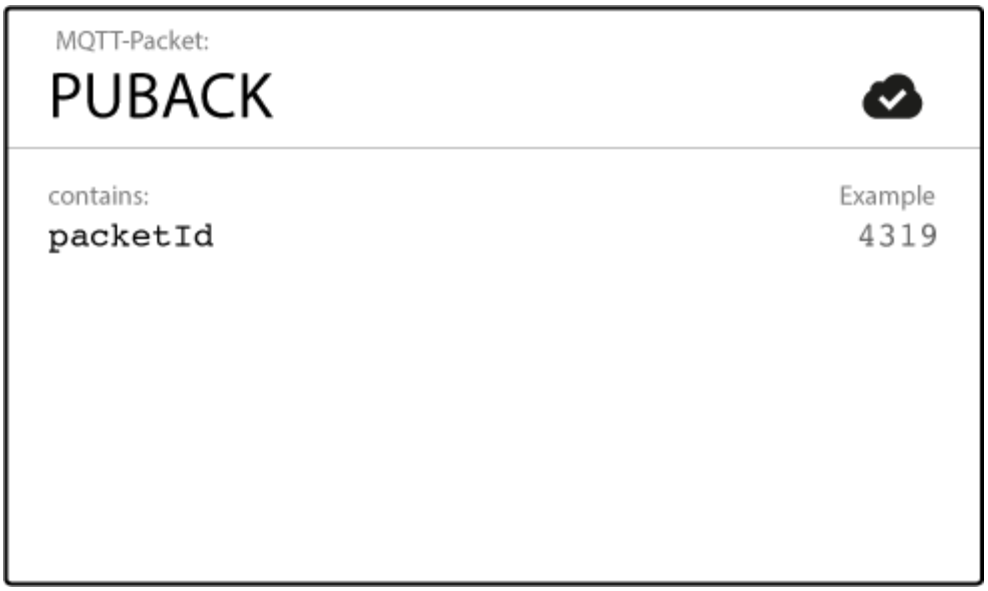
7.1.4.5.2 QoS 1

Eğer QoS 1 seviyesinde bir kaliteyle yayın yapılıyor ise istemcilerden en az bir tanesinin yayınlanan mesajı aldığı garanti edilmesi hedeflenir. Bunu sağlamanın tek yolu geri beslemedir.



Şekil 7.10. QoS 1 PUBACK Anlatımı

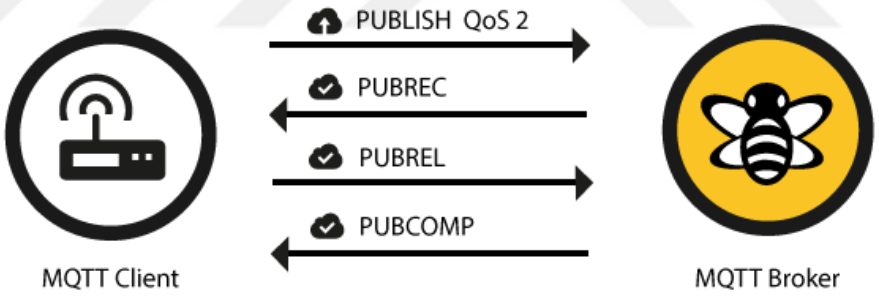
Yayınlanan her bir içerik için sağlayıcı en az bir adet PUBACK komutu bekleyecektir. Her paketin kendine ait tekil bir tanımlayıcısı bulunmaktadır. Bahse konu tekil tanımlayıcı ile geri dönen bilginin ardından yayınlanan içerik talebe göre sistemden silinir, yada yayını durdurulur.



Şekil 7.11. QoS 1 PUBACK Paket Yapısı

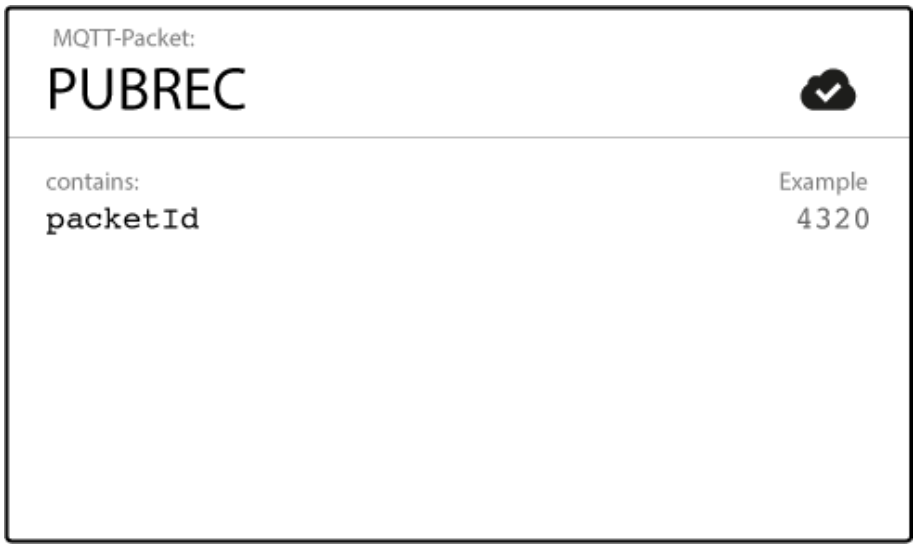
7.1.4.5.3 QoS 2

Kalite 2 seviyesinde her mesajın sadece bir defa iletildiği garanti edilir. En güvenilir yayındır fakat en yavaş olanıdır.



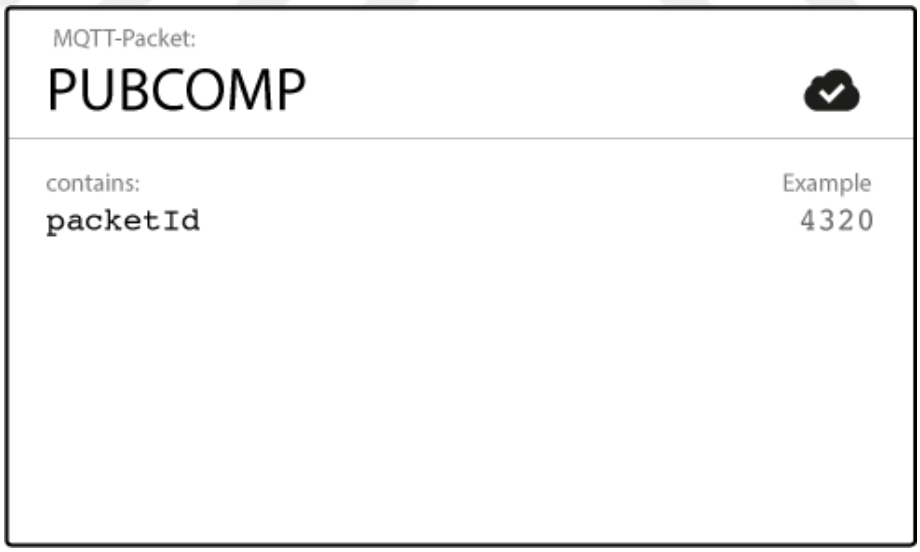
Şekil 7.12. QoS 2 Yapısı

Eğer bir istemci QoS 2 seviyesinde yayınlanmış bir mesajı alır ise PUBREC mesajını yayıncıya gönderecektir.

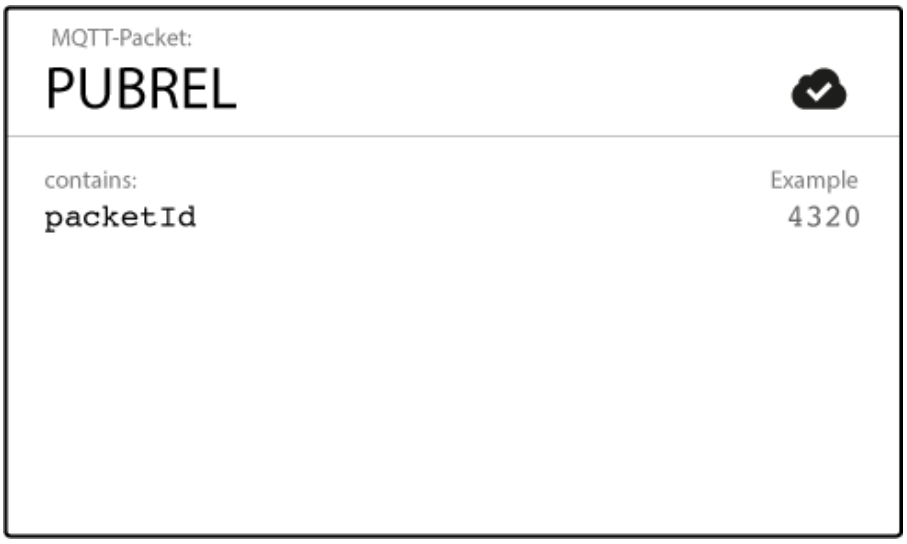


Şekil 7.13. QoS 2 PUBREC Paket Yapısı

İstemci mesaj paketinin tekil tanımlayıcısını PUBCOMP komutuna kadar saklayacaktır. Bunun amacı aynı mesajı birden fazla defa işlememek içindir. PUBREC komutu yayıncıya ulaştığında yayıncı PUBREL komutunu aynı paket için yayınlar. PUPREL komutunun istemciye olaşması ile PUBCOMP komutu yayınlanır ve mesaj işlemi sıralaması tamamlanmış olur.



Şekil 7.14. QoS 2 PUBCOMP Paket Yapısı



Şekil 7.15. QoS 2 PUBREL Paket Yapısı

Kalite istemci baskın bir özelliktir. Sunucu QoS 2 yayın yapsa dahi istemci QoS 1 ile abone oluyor ise sunucuya bu seviyede cevap verecektir. Bu durumda sunucunun QoS seviyesinin bir önemi kalmamaktadır.

7.1.4.5.4. QoS 0 nezaman kullanılır?

- İstemci ve sunucu arasında stabil bir bağlantı var ise.
- Sunucu tarafından yayınlanan mesajların istemci tarafından alınıp alınmadığı önemli değil ise.
- İşlem maliyetinin en düşük oranda tutulması gerektiği gibi bir durum var ise.

7.1.4.5.5. QoS 1 nezaman kullanılır?

Her mesajı almak isteniliyorsa ve mesaj çoklamaları uygulama tabanında çözüme uşatı ise.

7.1.4.5.6. QoS 2 nezaman kullanılır?

- Uygulamada her mesajın tam anlamıyla bir deva alınacağı bir durum var ise.
- Hız ile ilgili bir sorun yok ise.

7.1.4.5.7. MQTT yayın – abonelik

MQTT protokolü mesajların yayınlanması ve yayınlanan konulara abone olunması ilkesine dayanmaktadır. Birden fazla müşteri bir aracıya bağlanır ve ilgilendikleri konulara abone olur. İstemciler ayrıca aracıya bağlanır ve iletileri

konulara yayınlar. Birçok müşteri aynı konulara abone olabilir ve bilgileri istedikleri gibi alabilirler. Ayrıca MQTT, bağlanacak herşey için basit ve ortak bir arayüz görevi görür. Basit tip veritabanlarından kompleks tiplere kadar tüm veri tabanlarına kolay bir erişim sağlamak mümkündür.

7.1.4.5.8. MQTT kullanıcı güvenliği

MQTT kullanıcı adı ve kısa şifreler ile güvenlik sağlayabilmektedir. Ancak bu durum gelişen teknolojide yeterli entropiye sahip değildir. AMQP bu konuda SASL mekanizmasına sahiptir ve protocol değiştirmeksizin güvenlik seçeneklerini arttırabilir. AMQP mesajlaşmadan öne güvenliği sağlar ve ardından mesajlaşır. Ancak bu durum iletişimde zaman kaybına sebep olmaktadır.

7.1.4.5.9. MQTT ve AMQP karşılaştırması

- [3]Her ikiside basit mesajlaşma ihtiyacını karşılar, AMQP çok zengin bir mesajlaşma senaryosu için kullanılabilir.
- MQTT daha düşül tip mesajlar için uygundur. Kolay implementasyon ve özellikle gömülü yazılımlar için tercih edilir.
- AMQP HTTP'nin asenkton tamamlayıcısı niteliğindedir.
- Her ikisi de ortamlar arası geçi için oldukça esnektir.
- AMQP finans komitesi altyapılıdır ve içerik MQTT ye nazaran daha komplekstir.
- MQTT üretici tabanlı gelişmiştir ve ihtiyaçlar için oldukça basit bir yapıdadır.
-

7.1.4.5.10. Mesajlaşma senaryolarına göre MQTT ve AMQP karşılaştırması

- MQTT yayın-abonelik prensibi ile konulara bağlanma yöntemini destekler. Dez avantaj ise uzun ömürlü mesaj kuyruklarının oluşmamasıdır.
- AMQP aynı mantığa göre işler ancak kuyruk üzerindeki ömür 5 farklı tipte değerlendirilebilir. Bu yapısı sayesinde daha esnek bir kuyruk ömür süresine sahiptir.

7.1.4.5.11. İletişimde kullanılan mesaj yapısı

RabbitMQ aracılığı ile sağlanacağından en mantıklı çözüm olark mesajın tamamını belirli br şablonda string olarak göndermek gerekliliği doğmuştur. Bu durum neticesinde string formatın en kolay yöntemi olan JSON parsing işlemi tercih edilmiş ve buna uygun bir model belirlenmiştir. Model içeriği aşağıdaki gibidir.

```
public string AntenGrup { get; set; }
public string RSSI { get; set; }
public string AntenNumarasi { get; set; }
public string Barkod { get; set; }
public DateTime OkumaTarihi { get; set; }
```

Modelin detayında,

- Anten Grup bilgisi; antenin hangi üçlü grupta yer alacağını gösteren bilgidir ve ön tanımlı olarak her bir Arduino cihaza eklenmiştir.
- RSSI bilgisi; antenin almış olduğu dijital RSSI değeri içindir.
- Anten Numarası; her bir anten için tekil olarak ön tanımlıdır.
- Barkod bilgisi; UHF antenden gelen barkod bilgisinin dijital değeridir.
- Okuma tarihi; kayıt gönderme işleminin yapıldığı andaki değerdir. Bu değer 1 dakikalık aralıktaki 3 lü anten grubunda farklı antenlerden gelen RSSI ve Barkod tekil bilgisi ile işlenecek ve gelen değere göre RFID etiketin yeri tespit edilecektir.

ARDUINO sayesinde Üretilecek olan JSON a örnek aşağıdaki gibidir.

{"AntenGrup":"AG1","RSSI":"87","AntenNumarasi":"AG1_2041175501","Barkod":"326D423D-B2E1-4F61-92F0-0B8C00471D14","OkumaTarihi":"2018-05-19T17:26:56.1363464+03:00"}

7.1.4.5.12. ARDUINO uygulaması

```
#include <SPI.h>
#include <PubSubClient.h>
#include <Ethernet.h>
#include <DateTime.h>
#include <DateTimeStrings.h>
#define MQTT_SERVER "localhost"
#define MQTT_PORT 1883
```

```
#define MQTT_CLIENTID "1-16-2_1"
#define MQTT_USERNAME "guest"
#define MQTT_PASSWORD "guest"
#define MQTT_PUBLISHTOPIC "1-16-2"
byte MAC_ADDRESS[] = { 0x90, 0xA2, 0xDA, 0x0D, 0x31, 0xB8 };
int RSSIPin = 7;
int RSSIValue = 0;
String barcodeValue="";
String BarcodePin=7;
EthernetClient ethClient;
PubSubClient client;
void setup()
{
 Serial.begin(9600);
 if (Ethernet.begin(MAC_ADDRESS) == 0)
 {
    Serial.println("Failed to configure Ethernet using DHCP");
    return;
 }
 delay(500);
 Serial.println("Connecting...");
 for (byte thisByte = 0; thisByte < 4; thisByte++) {
  Serial.print(Ethernet.localIP()[thisByte], DEC);
  Serial.print(".");
 }
 client.setClient(ethClient);
 client.setServer(MQTT_SERVER,MQTT_PORT);
}
void loop()
{
 if (!client.connected())
 {
 RSSIValue = digitalRead(RSSIPin);
```

```
 barcodeValue =digitalRead(BarcodePin);
    unsigned long prevtime = DateTime.now();
    String responseBodyString = "{"
     "\"AntenGrup\":\"1-16-2\",";
     "\"RSSI\":\""+RSSIValue+"\","
     "\"AntenNumarasi\":"+MQTT_CLIENTID+","
     "\"Barkod\":\""+barcodeValue+"\","
     "\"OkumaTarihi\":\""+prevtime+"\""
    "}";
    client.connect(MQTT_CLIENTID, MQTT_USERNAME, MQTT_PASSWORD);
    Serial.println("Server to connected.");
    client.publish(MQTT_PUBLISHTOPIC, responseBodyString);
    delay(60000);
  }
}
```

7.1.4.5.13. Servis uygulaması

```
MqttClient clientSub;
delegate void SetTextCallback(string text);
public MainForm()
{
InitializeComponent();
}
private void MainForm_Load(object sender, EventArgs e)
{
try
{
string BrokerAddress = "localhost";

clientSub = new MqttClient(BrokerAddress);
clientSub.MqttMsgPublishReceived += new
MqttClient.MqttMsgPublishEventHandler(EventPublished);
```

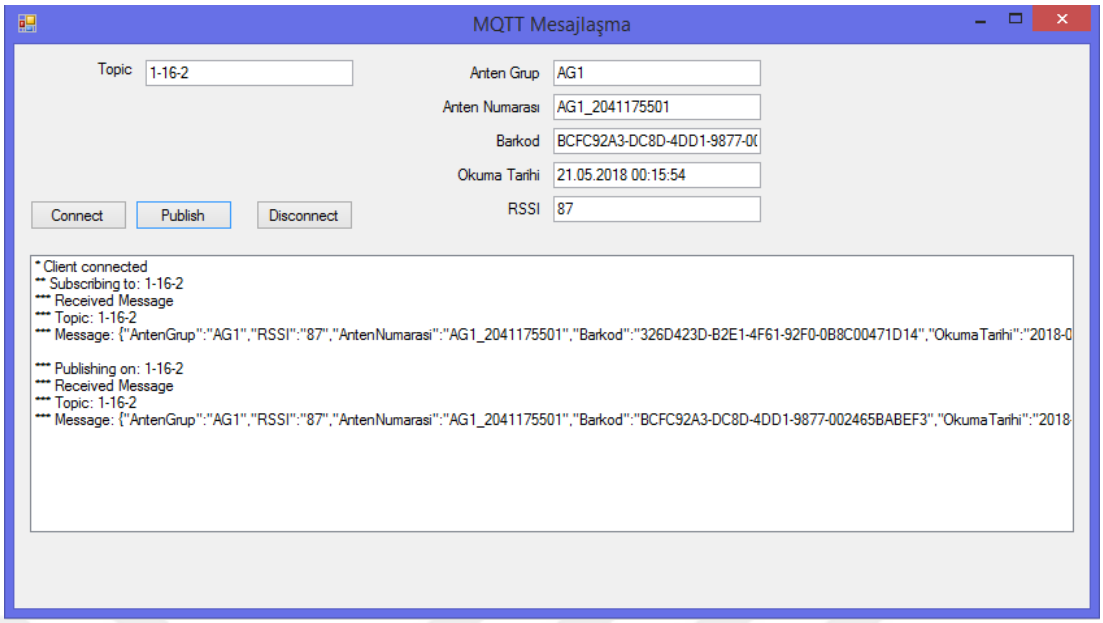
```
}
catch (InvalidCastException ex)
{
}
}
private void EventPublished(Object sender,
uPLibrary.Networking.M2Mqtt.Messages.MqttMsgPublishEventArgs e)
{
try
{
SetText("*** Received Message");
SetText("*** Topic: " + e.Topic);
SetText("*** Message: " +
System.Text.UTF8Encoding.UTF8.GetString(e.Message));
SetText("");
}
catch (InvalidCastException ex)
{
}
}
private void SetText(string text)
{
if (this.lsMessages.InvokeRequired)
{
SetTextCallback d = new SetTextCallback(SetText);
this.Invoke(d, new object[] { text });
}
else
{
this.lsMessages.Items.Add(text);
}
if (text.Contains("Message:"))
{
```

```
text = text.Remove(0, 13);
var obj = JsonConvert.DeserializeObject<MessageModel>(text);
MessageModel mm = new MessageModel();
mm = (MessageModel)obj;
txtAntenGrup.Text = mm.AntenGrup;
txtAntenNumarasi.Text = mm.AntenNumarasi;
txtBarkod.Text = mm.Barkod;
txtOkumaTarihi.Text = mm.OkumaTarihi.ToString();
txtRSSI.Text = mm.RSSI;
}
}
private void btnDisconnect_Click(object sender, EventArgs e)
{
try
{
clientSub.Disconnect();
lsMessages.Items.Add("* Client disconnected");
}
catch (InvalidCastException ex)
{
}
}
private void btnConnect_Click(object sender, EventArgs e)
{
try
{
byte result = clientSub.Connect(Guid.NewGuid().ToString(), "guest", "guest",
false, 500);
if (clientSub.IsConnected)
{
lsMessages.Items.Add("* Client connected");
clientSub.Subscribe(new string[] { txtTopic.Text }, new byte[] {
MqttMsgBase.QOS_LEVEL_AT_MOST_ONCE });
```

```
lsMessages.Items.Add("** Subscribing to: " + txtTopic.Text);
}
}
catch (InvalidCastException ex)
{
}
}
private void btnPublish_Click(object sender, EventArgs e)
{
try
{
Random rnd = new Random(10);
MessageModel mm = new MessageModel();
mm.AntenGrup = "AG1";
mm.AntenNumarasi = "AG1_" + (rnd.Next()).ToString();
mm.Barkod = Guid.NewGuid().ToString().ToUpper();
mm.OkumaTarihi = DateTime.Now;
mm.RSSI = (rnd.Next(50, 100)).ToString();
string json = JsonConvert.SerializeObject(mm);
clientSub.Publish(txtTopic.Text, Encoding.UTF8.GetBytes(json),
MqttMsgBase.QOS_LEVEL_AT_MOST_ONCE, true);
lsMessages.Items.Add("*** Publishing on: " + txtTopic.Text);
}
catch (InvalidCastException ex)
{
}
}
```

7.1.5. Servis uygulaması arayüz

Servis uygulamasında hem publish hem de sunscribe işlemleri gerçekleşmektedir. Bu sayede random üretilmiş RSSI değerleri mesaj içeriği olarak alınabilmekte ve kullanıcıya gösterilebilmektedir.



Şekil 7.16. Servis Uygulaması Arayüzü

8. SONUÇ VE ÖNERİLER

Standart barkod ve RFID ile karşılaştırma yapılmasındaki temel amaç Endüstri 4.0 ile hayatımıza girecek olan otonom depo yönetim sistemlerinin doğru yatırımlarla ve daha sağlam bir süreç mimarisi ile işlenebilir olmasını açıklamaktır. Depolarda otomasyon daha az insan gücü, daha az maliyet ve daha kesin bilgi temellerine dayanmaktadır. Karşılaştırma sonucunda çıkan veriler ilgili yatırımların doğru yönlendirilmesi için yol gösterici olmayı hedeflemektedir. Çalışmalar değişmez fiziki şartlara sahip bir depo ve ortalama tüketimi belli bir üretim yapısında gerçekleştiği için yatırım bütçeleri açısından değerler oldukça yol göstericidir. Standart barkod etiketleme yöntemi ile yapılan işlemlerin artısı barkodun basılı olduğu etikette sadece tekil anahtar değil aynı zamanda ürüne ait bilgileride içerebilir olmasıdır. Bunun en önemli avantajı depo çalışanının ürüne ait belirli bilgileri hızlıca okuyabilmesidir. Bu özellik RFID etikette mümkün olmamaktadır. Standart barkod içerisinde yer alan bilgiler ilgili envanterin kesin konum bilgisini içerecek şekilde olsa da gerçek zamanlı bir durumda bu garanti edilemez ve sadece yazılı bir bilgiden ibaret olacaktır. Yatırım maliyeti açısından düşünüldüğünde RFID yapısından bir miktar daha ucuza gelmektedir. Ancak çevresel açıdan oldukça fazla geridönüşümsüz atık bırakması dezavantaj olarak karşımıza çıkmaktadır. RFID antenlerde fiyat dezavantajının temel sebebi anten adetlerini fazla olmasından kaynaklıdır ancak bunun asıl sebebi UHF antenlerin ölçüm yeteneklerinin ve kesin konum bilgisinin arttırılmasıdır. Şekil 5 üzerinde ki yan yana konumlandırılan antenler görüş açılarındaki sebebi ile çoklamaları kesinlik oranını arttırmak amacı ile konumlandırılmıştır. RFID nin fiyat dezavantajı ile birlikte kullanım kolaylığı, depoda yer alan ürünlerin türüne ve sayısına ilişkin değerlerin anlık olarak okunabilmesi. Ürünün hangi koordinatta olduğu hakkında bilgi vermesi hem depo yönetim süreci açısında hızlı hemde güven esasına dayalı bir yapı oluşturması bakımından avantajlı durumdadır. Ayrıca çevresel faktörler ve geridönüşümlü ürünlerinde ülke ve dünya genelindeki etkileri göz ardı edilemez. RFID ile lokasyon takibi pek çok alanda kullanılmakla birlikte benzeri depo yönetim sistemleri içinde oldukça uygun bir çözümdür. Yapılan araştırmalar sonucunda RFID ile depo yönetimi yenilikçi bir bakış açısı katmasından, adam

saat açısından ve güven esaslı bir yapıya oturtulmasından ve çevreci olmasından ötürü tercih edilebilir olarak değerlendirilmiştir.

KAYNAKLAR

Zhao , Patwari , Agrawal ve Rabbat, 2012. Directed by Directionality: Benefiting from the Gain Pattern of Active RFID Badges. 01.11.2017. https://www.computer.org/web/csdl/index/-/csdl/trans/tm/2012/05/ttm2012050865.html

Y. Zhang, L. Xie, Y. Bu, Y. Wang, J. Wu and S. Lu, "3-Dimensional Localization via RFID Tag Array," 2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), Orlando, FL, USA, 2018, pp. 353-361.

Tan, 2008. The Application of RFID Technology in the Warehouse Management Information System. 01.11.2017. https://www.computer.org/web/csdl/index/-/csdl/trans/tm/2012/05/ttm2012050865.html

Khong, G., White, S., (2005), Moving right along: Using RFID for Collection Management at the Parliamentary Library, InformationOnline 12 th Exhibition & Conference, Sydney, 1-12

Demiral E, Karas İR, Turan MK. RFID sistemleri ile konum belirleme uygulamaları. 14. Türkiye Harita Bilimsel ve Teknik Kurultay, Ankara; 2013.

Ni, L. M., Y. Liu, Y. C. Lau and A. P. Patil, (2004), LANDMARC: Indoor Location Sensing Using Active RFID, Wireless Networks, 701–710.

Stelzer A., Pourvoyeur K., Fischer A., (2004), Conceptand application of LPM — a novel 3-D local position measurement system. IEEE Trans. Microwave Theory Techniques; 52(12):2664–9.

Videla and Williams, (2012) RabbitMQ in action. Manning,. Distributed Messaging for Everyone: ISBN 9781935182979

Nesta Elektronik, IoT Protokollerine Genel Bir Bakış, Erişim Tarihi:01.06.2017 http://www.nestaelektronik.com/blog/index.php?article=iot_review PrismTech Corporation, 2017, Messaging Technologies for the Industrial Internet and the Internet of Things Whitepaper, Erişim Tarihi:05.07.2017

http://www.prismtech.com/sites/default/files/documents/MessagingWhitepaper-051217.pdf RabbitMQ Organization, AMQP 0-9-1 Model Explained, Erişim Tarihi:14.05.2017

https://www.rabbitmq.com/tutorials/amqp-concepts.html Erişim Tarihi:14.05.2017

https://lists.oasisopen.org/archives/amqp/201202/msg00086/StormMQ_WhitePaper_-_A_Comparison_of_AMQP_and_MQTT.pdf Erişim Tarihi:14.05.2017

https://www.bayramucuncu.com/microsoft-message-queuing-msmq-nedir/
Erişim tarihi:15.09.2018

http://www.bahadirakin.com/tag/activemq-ornek/
Erişim Tarihi:10.07.2018

https://kodcu.com/2014/03/nosql-nedir-avantajlari-ve-dezavantajlari-hakkinda-bilgi/
Erişim Tarihi: 10.07.2018

ÖZGEÇMİŞ

Adı Soyadı : Hüseyin Cahit TOSUN

Doğum Yeri ve Yılı : GAZİANTEP, 14/06/1985

Medeni Hali : Evli

Yabancı Dili : İngilizce, İspanyolca

E-posta : cahit_tosun@hotmail.com

Eğitim Durumu

Lise : Gaziantep Anadolu Lisesi, 2003

Lisans : Atılım Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü

Mesleki Deneyim

GATE Elektronik	2009-2011
Ülkem Bilişim	2011-2012
Sedef Gemi İnşaatı A.Ş	2012-2014
Teknoser A.Ş	2014-2015
Sedef Gemi İnşaatı A.Ş	2015-2017
Logiwa Yazılım	2017-...(Devam Ediyor)

Yayınları

Tosun, C., Zaim, A. 2018. RFID Sistemleri ile Depo Yönetim Sistemlerinde Konum Belirlemede Kullanılacak Yöntem ve Teknolojilere Genel Bakış. İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi(Kabul Edildi).