T.C.
İSTANBUL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

SAĞLIK KURUMLARINDA NESNELERİN İNTERNETİ TEKNOLOJİSİNİN KULLANIMI: BİR VAKA ÇALIŞMASI

Murat TEKBAŞ

Enformatik Anabilim Dalı

Enformatik Programı

Uygundur
21.08.2019
Prof. Dr. S. Gülseçen

DANIŞMAN
Prof. Dr. Sevinç GÜLSEÇEN

II. DANIŞMAN
Dr. Emre AKADAL

Ağustos, 2019

İSTANBUL

Bu çalışma 21.08.2019 tarihinde aşağıdaki jüri tarafından Enformatik Anabilim Dalı Enformatik Programında Yüksek Lisans Tezi olarak kabul edilmiştir.

Tez Jürisi

Prof. Dr. Sevinç GÜLSEÇEN (Danışman)
İstanbul Üniversitesi
Enformatik Bölümü

Prof. Dr. Birgül KUTLU BAYRAKTAR
Boğaziçi Üniversitesi
Uygulamalı Bilimler Yüksekokulu

Doç. Dr. Zümrüt ECEVİT SATİ
İstanbul Üniversitesi
Siyasal Bilgiler Fakültesi

20.04.2016 tarihli resmi gazetede yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince; Bu Lisansüstü teze, İstanbul Üniversitesi′nin abonesi olduğu intihal yazılım programı kullanılarak Fen Bilimleri Enstitüsü′nün belirlemiş olduğu ölçütlere uygun rapor alınmıştır.

ÖNSÖZ

Tez çalışmam sırasında kıymetli bilgi, birikim ve tecrübeleri ile bana yol gösteren danışman hocam ve bölüm başkanımız Sayın Prof. Dr. Sevinç GÜLSEÇEN'e,

Tez çalışmam süresince kıymetli görüşlerinden yararlandığım ve benden yakın ilgisini esirgemeyen yol gösterici ikinci danışman hocam Sayın Dr. Emre AKADAL'a,

Tez çalışmamda bana yol gösteren, tavsiye ve yorumlarıyla bana destek olan Sayın Öğr. Gör. Dr. Murat GEZER'e,

Yüksek lisans eğitimim boyunca yardımlarını esirgemeyen, bilgi ve birikimlerini aktaran İstanbul Üniversitesi'nin tüm öğretim üyelerine ve çalışanlarına,

Tez çalışmamın gerçekleştirildiği süreçte, gösterdikleri sabır ve destekleri için pek sevgili ailemin tüm fertlerine ve önerileriyle bana destek olan tüm dostlarıma en içten teşekkürlerimi sunarım.

Ağustos, 2019 Murat TEKBAŞ

İÇİNDEKİLER

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

TABLO LİSTESİ

Sayfa No

SİMGE VE KISALTMA LİSTESİ

Kısaltmalar **Açıklama**

AMQP : İleri Mesaj Dizisi Protokolü
API : Uygulama Programlama Arayüzü
CoAP : Kısıtlayıcı Uygulama Protokolü
EMRAM : Elektronik Medikal Sağlık Kaydı Adaptasyon Modeli
HDTBS : Hasta Dosyası Takip ve Bilgi Sistemi
HIMSS : Sağlık Bilgi ve Yönetim Sistemleri Topluluğu
HTTP : Hiper Metin İletim Protokolü
IETF : İnternet Mühendisliği Görev Grubu
IP : İnternet Protokolü
IoT : Nesnelerin İnterneti
M2M : Makineden Makineye
MQTT : Telemetri Mesajlaşma Protokolü
QoS : Ağ İletişimi Hizmet Kalitesi
RFID : Radyo Frekansı Tanımlama
REST : Temsilî Durum Aktarımı
UDP : Kullanıcı Veribloğu İletişim Kuralları
TCP : İletim Kontrol Protokolü
XMPP : Genişletilebilir Mesajlaşma ve Varlık Protokolü

ÖZET

YÜKSEK LİSANS TEZİ

SAĞLIK KURUMLARINDA NESNELERİN İNTERNETİ TEKNOLOJİSİNİN KULLANIMI: BİR VAKA ÇALIŞMASI

Murat TEKBAŞ

İstanbul Üniversitesi

Fen Bilimleri Enstitüsü

Enformatik Anabilim Dalı

Danışman: Prof. Dr. Sevinç GÜLSEÇEN

II. Danışman: Dr. Emre AKADAL

Yüksek performanslı hesaplama cihazlarının boyutlarının, güç gereksinimlerinin ve maliyetlerinin azalması, fiziksel ortamdan veri toplayabilen algılayıcılara erişimin kolaylaşması ve tüm bu cihazların birbirleriyle haberleşebilmesinin çeşitli protokoller yardımıyla mümkün hale gelmesi; Nesnelerin İnterneti üzerine yapılan çalışmaların ve uygulamaların artmasını sağlamıştır. Günlük hayat içerisinde her geçen gün daha çok yer alan bu teknoloji, sağlık alanında hasta verilerinin toplanmasında, uzaktan hasta bakımında ve çeşitli mobil tıbbi uygulamalarda kullanılabilmektedir.

Bu tez çalışması kapsamında, sağlık kurumlarında Nesnelerin İnterneti teknolojisinin kullanımıyla ilgili araştırmalar yapılmıştır. Yapılan alan yazın taramasından elde edilen bilgilerle, Nesnelerin İnterneti teknolojisinin ne olduğu, kullanılan haberleşme teknolojileri ve alınması gereken güvenlik önlemleri açıklanmıştır. Sağlık kurumları için önerilen Nesnelerin İnterneti temelli sistemler incelenmiş ve sağlık hizmeti sektöründe katkıları değerlendirilmiştir.

Bu çalışmada, sağlık kurumlarında fiziksel olarak saklanması gereken dosyaların takibi için, Nesnelerin İnterneti konseptinde çalışan bir hasta dosyası takip ve bilgi sistemi geliştirilmiştir. Sistem düşük maliyetli bir platform üzerinde çalışmakta olup dosyaların oda giriş çıkışlarında otomatik tanınmasını ve kayıt altına alınmasını sağlamaktadır.

Ağustos 2019, 88 sayfa.

Anahtar kelimeler: Nesnelerin interneti, dijital dönüşüm, dijital hastane, MQTT.

SUMMARY

M.Sc. THESIS

THE USE OF INTERNET OF THINGS TECHNOLOGY IN HEALTH INSTITUTIONS: A CASE STUDY

Murat TEKBAŞ

İstanbul University

Institute of Graduate Studies in Sciences

Department of Informatics

Supervisor: Prof. Dr. Sevinç GÜLSEÇEN

Co-Supervisor: Dr. Emre AKADAL

Reducing the size, power and cost of high-performance computing devices, easy access to sensors which can collect data from the physical environment and possibility communication of these devices with each other through various protocols have led to an increase in the studies and applications on Internet of Things. This technology which is taking place more and more in daily life can be used for collecting patient medical records, monitoring patients remotely and developing various mobile medical applications.

Within the scope of this thesis, research have been done on the use of Internet of Things technology in health institutions. What the technology of Internet of Things is, communication technologies between devices and the security measures to be taken are explained with the information obtained from the literature research.

In this study, a patient file tracking and information system which is working in the Internet of Things concept was developed to follow up the files that should be physically conserved in health institutions. The system works on a low-cost platform, enabling automatic recognition and recording of files at room entrance and exit.

August 2019, 88 pages.

Keywords: Internet of things, digital transformation, digital hospital, MQTT.

1. GİRİŞ

İnsanların tarih boyunca gelişmelerinin ve bugünkü duruma gelmelerinin şüphesiz ki en önemli sebeplerinden biri iletişim kurabilmeleri ve ortak iş yapabilme gücünden gelmektedir. Beraber çalışmak; karşılaşılan sorunları daha hızlı çözmeyi, bu problemleri çözerken yeni bilgiler öğrenmeyi ve bu bilgiyi diğer insanlarla paylaşarak yaymayı; nihayetinde de bu bilgiler neticesinde hayatı kolaylaştıran yeni icatlar yapmayı sağlamıştır. Güçlü ve Sotirofski (2006), veriyi, özümlenmemiş ve yorumlanmamış gözlemler, enformasyonu ise düzenlenmiş veri olarak tanımlamışlardır. Çapar (2005); bilgi, enformasyon ve veri arasındaki ilişkiyi aşağıdaki gibi belirtmiştir.

- Veri ve enformasyon, bilginin temelini oluşturmaktadır.
- Bilgi, enformasyonun akıl süzgecinden geçmesiyle ve yorumlanmasıyla oluşmaktadır.
- Bilgi, karar verme, analiz, tahmin gibi eylemlerin ve uygulamaların temelini oluşturmaktadır.

İnsanların karar vermesinde ve davranışlarını şekillendirmesinde bilgi daima yol gösterici olmuştur (Gülseçen, 2016). Bilginin insanlar arasında aktarılması tarih boyunca çivi yazısı, papirüs, matbaa gibi çeşitli şekillerde gerçekleşmiş ve yayılma hızı ise teknolojiye bağlı olarak artmıştır. Endüstri Devrimi ile birlikte bilimin tüm alanlarındaki gelişmesi de ivmelenmiştir (Arslan, 2017). Geçtiğimiz yüzyıl içerisinde bilişim alanında yaşanan gelişmelerle artık bilginin yayılma hızı geçmişteki hızla karşılaştırılamayacak kadar büyüktür. Bu gelişmelerle bilgi yönetimi önem kazanmış ve bilginin çeşitli iş süreçlerinde kullanımı mümkün hale gelmiştir (Ives ve diğerleri, 1997).

Son zamanlarda yaşanan teknolojik gelişmeler çevreden veri toplayabilen algılayıcıların çok daha az maliyetle geliştirilmesine olanak sağlamıştır (Özdemir ve diğerleri, 2017). Algılayıcılardan alınan veriler yardımıyla kararlar verebilen uygulamaların yapılabilmesi sağlanmıştır. Otomatik tanımlama cihazlarının yaygınlaşması ve haberleşme teknolojilerinin çeşitlenmesiyle algılayıcı cihazlardan alınan verilerin başka cihazlar tarafından okunabilmesi sağlanmış ve verilerin işlenerek süreç sonunda bilgi üretilebilmesi sağlanmıştır. Bilgi,

eskiden sadece insanlar arasında aktarılabilirken artık cihazlar -bir başka deyişle nesneler- arasında da aktarılabilir duruma gelmiştir. Toplanan bu verilerin analiz edilebilmesi veri madenciliği gibi yeni bilim alanlarının oluşmasına olanak sağlamıştır (Şişmanyazıcı ve Doğan, 2016). İnsan faktörünün aradan çıkarak nesnelerin birbirleriyle uyumlu şekilde haberleşmesi ve veri paylaşarak kararlar verebilmesi sayesinde bir çok alanda teknolojik gelişmelerin önü açılmıştır.

Nesnelerin haberleşme yeteneklerinin ve kendi arasında veri aktarımı yapabilmesinin her gün daha da arttığı günümüzde, verilerin toplanması, paylaşılması ve analiz edilmesi gibi işlemleri gerçekleştiren bilgi sistemleri oldukça ön plana çıkmaktadır. Yüksek performanslı hesaplama cihazlarının boyutlarının ve maliyetlerinin azalması, fiziksel ortamdan veri toplayabilen algılayıcılara erişimin kolaylaşması ve de tüm bu cihazların birbirleriyle haberleşebilmesinin çeşitli protokoller yardımıyla mümkün hale gelmesi; Nesnelerin İnterneti (Internet of Things, IoT) üzerine yapılan çalışmaların ve uygulamaların artmasını sağlamıştır. Günlük hayat içerisinde her geçen gün daha çok yer alan bu teknoloji sayesinde; sağlık alanında, uzaktan hastanın fizyolojik verilerinin toplanması, analiz edilmesi ve bu analizler sonucunda hastaya uygun tedavi yöntemine karar verilmesi gibi birçok alanda yararlanılmaktadır.

Tezin bölümleri incelendiğinde; genel kısımlar bölümünde, IoT teknolojisinin ne olduğundan, hangi teknolojilerin kullanıldığından, nesnelerin haberleşmesinde kullanılan protokollerden ve güvenlik zaafiyetlerine değinilmiş sonrasında dijital dönüşüm ve IoT teknolojisinin sağlık alanında kullanılmasıyla ilgili bilgiler sunulmuştur.

Yöntem ve Araçlar bölümünde, gerçekleştirilen çalışmanın aşamaları ve bu aşamalarda kullanılan yöntem ve araçlardan bahsedilmiştir.

Sonuç başlığı altında çalışmanın faydası tartışılmış, çalışma sonrası elde edilen verilerden yola çıkılarak sonuç görüşü belirtilmiştir.

Tezin son bölümünde çalışmada kullanılan kaynaklar ve ekler yer almaktadır.

1.1. PROBLEM TANIMI

Bu çalışma sayesinde çözmeyi hedeflediğimiz problem; sağlık kurumlarında dijital dönüşüm kapsamında IoT teknolojisini kullanarak iş süreçlerinin otomatize edilmesini sağlamak ve böylece kurum içerisinde yaşanan olumsuz durumların önüne geçmeye yardımcı olmaktır. Çalışma kapsamında, dijital dönüşüm ve IoT teknolojisi incelenmiş, sağlık kurumlarında kullanım alanları araştırılmış ve örnek bir vaka çalışması yapılmıştır. Örnek vaka çalışması, hasta dosyalarının yoğunluktan dolayı kaybolması durumunun önüne geçilmesi olarak belirlenmiştir. Belirlenen örnek vakaya bağlı olarak ortaya çıkan alt problemler aşağıda belirtildiği gibi tespit edilmiştir.

- Hasta dosyalarının kaybolmasından ilgili sağlık kurumu sorumludur, bu durum ağır hizmet kusuru olarak görülmekte ve sağlık kurumunun hukukî yaptırımlarla yüzleşmesine neden olmaktadır. Danıştay 10. Dairesi E: 2007/3301 K: 2008/2939 K.T.: 29.04.2008 emsal kararına göre hastaya ait grafilerin gerekli şekilde muhafaza edilmemesi ve hasta dosyasının kaybedilmesi, sağlık hizmetinin işletilmesinde ağır kusur oluşturur kararına varılmıştır (Bozdağ, 2010).
- Hasta, dosyası kaybolduğu için tedavi sürecinde zaman kaybı yaşanmakta ve bu durum geri dönüşü olmayan kayıplara sebebiyet verebilmektedir.
- Hasta dosyalarının arşive ne zaman girdiği ve çıktığı sistemsel olarak takip edilememektedir.
- Tıbbî müdahalelerde her saniye önem arz etmektedir. Hasta dosyaları zaman içerisinde birikerek sehven olması gereken yerden başka bir yere konulduğunda dosyanın bulunması için fazla zaman harcanmaktadır.

1.2. ARAŞTIRMANIN AMACI VE ÖNEMİ

Dünya nüfusunun hızlı artışına bağlı olarak bulaşıcı hastalıklar, ruhsal bozukluklar ve kronik hastalıkların görülme sıklığı artmıştır (Aydoğan ve Metintaş, 2017). Dünya nüfusunun hızlı artışı ve yaşam süresinin uzamasıyla yaşlı nüfusun genç nüfusa oranı geçtiğimiz yıllara oranla artmıştır (Tekin ve Kara, 2018). Bu durum sağlık hizmetlerine duyulan ihtiyacı arttırmakla beraber hizmetlerin kullanım oranı ve maliyetini de arttırmaktadır.

Sağlık hizmetlerine duyulan ihtiyaç ve maliyet artışı, hizmetlerin sağlanmasında bir takım aksaklıklara sebebiyet vermekte ve herkesin ihtiyaç duyduğu anda bu hizmetlere erişebilmesine engel teşkil etmektedir (Gözlü ve Tatlıdil, 2015). Son yıllarda teknoloji alanında yaşanan gelişmelerle çevreden veri toplayan cihazlardan alınan veriler uzak sunuculara gönderilerek işlenebilmekte, analiz edilen veriler ışığında gereken önlemlerin alınması için harcanan süre kısaltılmakta ve böylece hastalara uzaktan bakım hizmeti sunulması daha kolay hale getirilmektedir. Bu gelişmelerle özellikle yaşlı nüfusun sağlık kurumlarına gitmesine gerek kalmadan hayati fonksiyonlarının sürekli izlenebilmesi ve acil durumlarda uyarı verilmesi sağlanabilmektedir (Misra ve diğerleri, 2018). Veri toplayan bu cihazların haberleşebilmesi ve veri aktarabilmesi sayesinde sağlık hizmetlerinde yeni teknolojilerin önü açılmıştır (Alqahtani, 2018).

Sağlık kurumlarındaki yoğun çalışma koşulları insan kaynaklı hataların oluşmasına sebebiyet verebilmektedir. IoT tabanlı çözümlerle oluşan hataların önlenmesi sağlanabilmekte ve süreçlerin otomatize edilmesiyle personel kaynakları daha fazla fayda sağlanabilecek alanlara aktarılabilmektedir. Nesnelerin birbirleriyle haberleşebilmesi ve sonucunda bilgi üreterek karar verebilmesi sağlık kurumlarında süreçlerin çok daha hızlı ve hatasız yapılabilmesine olanak sağlamıştır. Otomatize edilen süreçler için personel kaynakları azaltılmakta ve ihtiyaç olan diğer alanlarda kullanılabilmektedir. Tüm bu faydalar sağlık kurumlarında zaman ve personel kaynaklarının etkin bir şekilde kullanılmasına ön ayak olmaktadır. Bu tez çalışması kapsamında sağlık kurumları özelinde yapılan dijital dönüşüm çalışmaları, IoT tabanlı sistemler ve bu sistemlerde kullanılan teknolojiler incelenmiş ve aşağıda detayları verilmiş olan örnek bir vaka çalışması yapılmıştır. Yapılan vaka çalışmasıyla nesnelerin birbirleriyle haberleşmesi sağlanmış ve insan faktörünün aradan çıkarılması mümkün hale getirilmiştir.

Sağlık kurumları, sağlık hizmeti vermenin yanı sıra hizmet verdiği kişilere ait sağlık bilgilerini tutmakla ve korumakla da yükümlüdür (Bozdağ, 2010). Bu durum kişisel verilerin korunumu açısından önemli olduğu kadar bu bilgilere ihtiyaç duyulduğu an, zaman kaybına sebebiyet verilmeden, verilere hızlıca erişilebilmesi de acil durumlarda hayatî önem taşımaktadır. "Otomatik Tanımlama Sistemleri" insan faktörüne ihtiyaç duymadan canlıların ya da nesnelerin kimliğini verebilen sistemlerdir (Yüksel ve Zaim, 2009). Bu sistemlerin uygulamalarda kullanılmasıyla işyerlerinde personel devam takibi, vasıta giriş çıkış zamanlarının tutulması gibi insan gözetimine ihtiyaç duyan faaliyetler, birbirlerini

tanıyan ve haberleşebilen nesneler sayesinde otomatize edilmiştir (Tao ve diğerleri, 2016).

Yapılan araştırmanın amacı; ana ve alt problem cümleleriyle ifade edilen hasta dosyalarının kaybolması veya takibinin zor olması sebebiyle meydana gelen olumsuzlukları, geliştirilen sistemle çözüme kavuşturmaktır. Geliştirilen sistem sayesinde sistemin kurulduğu sağlık kurumunda Otomatik Tanımlama Sistemi'ne dahil edilen dosyaların binanın neresinde olduğu hızlıca tespit edilebilmektedir. Geliştirilen internet tabanlı bir web sitesi aracılığıyla sistemi kullanan kişiler hastaya ait dosyanın nerede olduğunu sorgulayabilmektedirler. Talep edilen dosyanın yerinin bilinmesi hem sağlık kurumu hem de hasta için zaman kaybının önüne geçmektedir. Sistemin sağlayacağı en önemli fayda ise dosya kayıplarının önüne geçilmesi olarak belirtilebilir. Bina dışına çıkarılan her dosyanın bilgisi sistemde muhafaza edilecek olup, gerektiği takdirde bina dışına çıkarılan dosya hakkında sağlık kurumunca ilgili kişilere bildirim yapılabilecektir.

Bu tez kapsamında geliştirilen sistem sayesinde sağlık kurumu içerisinde bulunan her hasta dosyasının bulunduğu lokasyonlar takip edilebilmekte olup, istenildiği takdirde dosyanın hangi odalar arasında geçiş yaptığı bilgisi de tarihçe olarak listelenebilmektedir. Bu sayede hasta gizliliği ihlali olup olmadığı sistem sayesinde hızla sorgulanabilmektedir.

Geliştirilen sistemin kolayca genişletilebilir olması hedeflenmiştir. Binaya yeni eklenen katlar ya da bölümler olması halinde ortamdan veri okuyan cihazların mevcut sisteme eklenmesiyle sistemin çok fazla müdahaleye gerek olmadan yeni eklenen bölümleri de kapsayarak çalışabilecek potansiyelde olması planlanmıştır.

2. GENEL KISIMLAR

Bu bölümde, araştırmanın genel kavramları, yapılan alan yazın taramalarıyla desteklenerek açıklanmıştır.

2.1. DİJİTAL DÖNÜŞÜM

Sanayi Devrimi sonrası yaşanan teknolojik gelişmeler sonucunda çeşitli alanlarda çalışmalar yapılmış ve bu çalışmalar neticesinde yeni bilgiler ortaya çıkmıştır. İnternet teknolojisinin gelişmesiyle bu bilgiler geniş kitlelere ulaştırılabilmiştir. Bilgi teknolojilerinin gelişmesinde kilit rol oynayan gelişme analog materyallerin sayısallaştırılması sürecidir (Yankın, 2019). Sayısallaştırma sürecinin iş ve yaşam alanına uygulanması dijitalleştirmeyi ve sonrasında da dijital dönüşüme yol açmaktadır.

2.1.1. Sayısallaştırma

Sayısallaştırma, analog materyalin bilgisayarda depolanması amacıyla sayısal formata dönüştürülmesi işlemidir (Karakaş ve diğerleri, 2009). Analog bir materyal olan el yazısıyla yazılmış metinlerin dijital formata dönüştürülmesi, video kaset ortamında bulunan bir görüntünün bilgisayarların gösterebileceği dijital veriye dönüştürülmesi uzun yıllardır yapılan sayısallaştırma örneklerindendir. Sayısallaştırmayla dijital ortama aktarılan verilere erişim ve bu verilerin başka bir noktaya iletilmesi çok daha hızlı bir şekilde gerçekleştirilebilmektedir. Verilerin hızlı iletimi, bilginin uzun mesafeler arasında da paylaşılabilmesini ve bu sayede ortak çalışmaların yürütülebilmesini sağlamıştır (Yoo, 2010).

2.1.2. Dijitalleştirme

Dijitalleştirme; iş dünyasında, sayısallaştırılmış verileri dijital teknolojiler yoluyla kullanarak iş operasyonlarını, işlevlerini, modellerini, süreçlerini ve faaliyetlerini etkinleştirmek, iyileştirmek veya dönüştürmek anlamına gelir (Yankın, 2019). Müşterilere

katalogları basılı bir şekilde postayla yollamak yerine dijital ortama aktarılmış kataloğu e-posta üzerinden göndermek dijitalleştirmeye bir örnektir.

2.1.3. Dijital Dönüşüm

Dijital Dönüşüm Portalı[1] tarafından yapılan tanıma göre dijital dönüşüm; "hızla gelişen bilgi ve iletişim teknolojilerinin sunduğu imkânlar ve değişen toplumsal ihtiyaçlar doğrultusunda, organizasyonların daha etkin, verimli hizmet vermek ve faydalanıcı memnuniyeti sağlamak üzere insan, iş süreçleri ve teknoloji unsurlarında gerçekleştirdiği bütüncül dönüşümü" olarak tanımlanır.

Bilgi teknolojilerinde yaşanan hızlı gelişmeler sayesinde sayısallaştırma ve dijitalleştirme süreçleri de aynı oranda gelişme kaydetmiş ve yaşamın neredeyse her alanında iş yapma şekillerini değiştirmeye başlamıştır (Altuntaş, 2018). İş yapma şekillerinde meydana gelen bu değişimlerle beraber akıllı sistemler, e-ticaret, sosyal medya gibi uygulamaların yaygınlaşmasıyla bir çok sektörde de değişimler yaşanmaktadır. Teknolojinin sağlamış olduğu bu yenilikler işlerin daha hızlı ve maliyetsiz yapılmasına olanak sağlamakla beraber iş süreçleri sonucu oluşan verilerin daha hızlı işlenmesini ve sonucunda bilgi üretilebilmesini de sağlamaktadır (Nuroğlu, E. ve Nuroğlu, HH., 2018).

Dijital teknolojiler ile öncelikle analog kayıtlar dijital ortamda işlenir hale getirilir ve sayısallaştırma işlemi gerçekleştirilir. Sayısallaştırma sonrasında süreçler dijital ortama aktarılarak dijitalleştirme yapılır. Son olarak tüm kurumsal varlıklar ve paydaş ilişkileri dijital ortamda yeniden tanımlanarak dijital dönüşüm tamamlanmış olur. Dijital dönüşüm içerisinde bulut bilişim, büyük veri, yapay zeka, IoT gibi bir çok farklı teknoloji barındırır (Tozkoparan ve Ernur, 2018). Tüm bu teknolojilerden faydalanılarak iş süreçlerinin daha etkin yürütülebilmesi için dijital teknolojinin bir işletmenin tüm alanlarına entegre edilmesi dijital dönüşümün temelini oluşturur.

2.1.4. Sağlık Kurumlarında Dijital Dönüşüm

Teknoloji alanında yaşanan gelişmeler sağlık kurumları içindeki işleyişin daha etkili şekilde işlemesine, insan kaynaklı hataların azaltılmasına, maliyetlerin düşürülmesine ve hasta memnuniyetinin artırılmasına imkan sağlamıştır (Tüfekçi ve diğerleri, 2017).

[1] $https://www.dijitaldonusum.gov.tr/dijital-donusum-nedir/$ internet sitesinden alınmıştır.

Sayısallaştırma, dijitalleştirme ve dijital dönüşüm süreçlerinin uygulanmasıyla teknolojinin en üst seviyede kullanılması, sağlık kurumlarında hizmet kalitesinin artmasıyla süreçlerin hızlı ve etkili bir şekilde dijital kanallardan geçmesine olanak sağlamış ayrıca "Dijital Hastane" kavramını literatüre kazandırmıştır.

Dijital Hastane, sağlık kurumu içerisindeki tüm bilgi sistemlerinin medikal ve medikal olmayan her türlü teknolojilerle tam entegre olduğu, güvenilir veri akış standartlarının belirlendiği, sağlık personellerinin yetkilerini daha az zaman ve enerji harcayarak hastane ve hasta bilgilerine her yerden mobil olarak erişimini sağlayan, el ile işlem yapılmayan, kağıtsız ve filmsiz olarak çalışan, doğru ilaç ve medikal tedavi uygulamalarının kontrol edildiği, tüm işlemlerin tam otomasyon sistemiyle yapıldığı, kontrol edildiği ve yönetildiği hastanelerdir (Ak, 2010).

Dijital hastaneler; hekimlerin hasta bilgilerine veri güvenliği sağlanmak şartıyla her yerden erişebilmesini, sağlık kurumunun idari ve klinik hizmetlerinin bilgisayar ortamından yürütülebilmesini, hastane içindeki ve dışındaki sistemlerin birbirleriyle tam entegrasyonun sağlanmasını ve hasta bakım süreçlerinde otomasyonun sağlanması gibi amaçlara ulaşmayı hedeflerler.

1961 yılında Chicago'da kurulan; Amerika, Avrupa ve Asya'da yapılanmaları bulunan ve kâr amacı gütmeyen bir organizasyon olarak bilinen Sağlık Bilgi ve Yönetim Sistemleri Topluluğu (HIMMS) dijital hastanelerin uluslararası standartlarını oluşturmak için sağlık kurumlarını 0'dan 7'ye kadar derecelendiren ve akredite eden Elektronik Medikal Sağlık Kaydı Adaptasyon Modeli (EMRAM) skorlamasını oluşturmuştur. EMRAM mevcut durum analizi yapan ve mevcut durumların eksikliklerin belirlenmesini sağlayan sonrasında bu kaydedilen aşamaların sertifikalanmasını sağlayan sürecin adıdır (Sebetci ve diğerleri, 2017).

Kılıç (2016), HIMMS tarafından Türkiye'de 7. seviye dijital hastane olarak kabul gören Tire Devlet Hastanesinin yapmış olduğu çalışmaları aşağıdaki gibi sıralamıştır.

- Hasta kayıt işlemleri, yatış ve diğer klinik işlemler, konsültasyon ve sevkler kağıtsız olarak dijital platformdan yapılmaktadır.
- Hastanede e-reçete kullanılmaktadır.
- MR, Röntgen, EKG, kan ve diğer yapılan test (işitme testi vb.) istemleri kâğıtsız

olarak bilgisayar ortamında yapılmakta, yine bu istemlere ait sonuçlar dijital ortamda sunulmaktadır. Bu sonuçlara hem sağlık görevlileri hem de hastalar istedikleri yerden telefon ve tabletler aracılığıyla ulaşabilmektedirler.

- Üretilen bütün veriler (kayıtlar, sonuçlar, faturalar vb.) dijital ortamda arşivlenmekte ve bilgi güvenliği sağlanmaktadır.
- Hekimlerin tedavi talepleri anlık olarak ve uzak erişim sağlanarak tamamen çevrimiçi ortamda yapılmaktadır.
- Hasta odalarında bulunan bilgisayar terminali sayesinde hemşireler uyguladıkları tedavileri hiçbir evrak ve kâğıt kullanmadan sisteme girmekte, bu sayede eczane, stok takip ve faturalama sistemi anlık olarak giriş-çıkışları kayıt altına alabilmektedir.
- Kapalı Devre İlaç Sistemi sayesinde, doğru ilaç, doğru zamanda, doğru dozda doğru hastaya uygulanmaktadır.
- Hastanedeki bütün idari evrak ve yazışmalar (Yasa gereği satın alma evrakları hariç) elektronik sistemde takip edilmekte ve e-imza kullanılmaktadır.
- Kaynakları tam zamanlı görebilmek için bütçe ve stok uyarı sistemleri gibi programlar kullanılmaktadır.
- Yangın, güvenlik, elektrik, su, doğalgaz gibi altyapı unsurlarının merkezi sistemle takibi yapılmaktadır. Acil durumlarda bu teknolojiler devreye girebilmektedir.
- Hastanede üretilen hiçbir veri kaybolmamakta, istenildiği an her yerden ulaşılabilmektedir.
- Kâğıt kullanımı olmadığı için kırtasiye giderlerinden tasarruf sağlanmaktadır.
- Akıllı yazılımlar sayesinde, hastane hizmetleri hızlı ve etkin yapılabilmektedir.

Dijital dönüşüm kavramıyla ortaya çıkan Dijital Hastaneler iş süreçlerinde hız ve verimliliği arttırmakta, yanlış ilaç verilmesi gibi insan kaynaklı hataları azaltmakta ve bu sayede hasta memnuniyetinin artmasına olanak sağlamaktadır. Sağlık kurumunun idari işlerinin otomasyon üzerinden sensörler yardımıyla takip edilmesi ve acil durumlarda uyarı verilerek önlem alınması iş süreçlerinin hızlı ve verimli bir şekilde

yürütülmesine olanak sağlamaktadır. Kurum içerisinde oluşan tüm veriler saklanarak veri analizlerinin yapılmasına, analizler sonucu oluşan bilgilerin karar destek sistemlerinde değerlendirilmesine ve daha doğru kararların alınabilmesine imkan sağlamaktadır.

2.2. NESNELERİN İNTERNETİ

Kevin Asthon'un 1999 yılında IoT terimini ilk kez kullanmasından sonra bu alanda yapılan çalışmaların sayısı giderek artmaktadır. Yapılan çalışmaların sayısının artmasına ve IoT teriminin giderek yaygınlaşmasına rağmen bu terimi tüm detaylarıyla kapsayan bir tanım bulunmamaktadır (Wortmann ve Flücher, 2015). Genel olarak IoT kavramı, benzersiz bir şekilde adreslenebilir nesnelerin kendi aralarında oluşturduğu dünya çapında yaygın bir ağ ve bu ağdaki nesnelerin belirli bir protokol sayesinde birbirleriyle iletişim içinde olmalarını sağlayan bir sistem olarak tanımlanmaktadır (Kutup, 2016). Gelecek teknolojileriyle ilgili eğilimler ve öngörüler sunan dünyaca ünlü araştırma kuruluşlarından Gartner'ın araştırmasına göre internete bağlı cihaz sayısının 2021 yılına kadar 25 milyarı bulması öngörülmektedir. Bu öngörüye bağlı olarak gelecek yıllarda IoT ile birbirine bağlı nesneler, cihazlar ve sensörlerin çok daha artacağı söylenebilir. İnternete bağlı bu cihazların birbirleriyle etkileşimde olmaları ve verileri paylaşmaları sonucunda bilgi üretebilmeleri sayesinde önümüzdeki yıllarda birçok alanda IoT kavramının kullanılabileceğini de öngörebiliriz. IoT,

- Nesneler
- Nesneleri birbirine bağlayan iletişim ağları
- Nesnelerden nesnelere akan verileri kullanan bilgisayar sistemleri (Ulaş, 2010).

olarak belirtilen üç ana bileşenden oluşmaktadır (Özvural, 2015).

Nesneler birbirleriyle cihazdan-cihaza (Device-To-Device), cihazdan-sunucuya (Device-To-Server), sunucudan-sunucuya (Server-To-Server) olmak uzere 3 farklı şekilde iletişim kurmaktadır (Karakaplan, 2010). IoT kavramında, nesnelerin birbirleriyle haberleşerek veri alışverişi yapabilmesi en belirgin ve temel özellik olarak öne çıkmaktadır. Nesnelerin iletişiminde insan müdahalesine gerek duyulmadan makineler birbirleri ile haberleşebilirler (Bozdoğan, 2015). IoT, bu iletişim teknolojilerini de kapsayan daha geniş

bir teknolojiyi betimler. Nesnelerin birbirleriyle iletişiminde sürece insan müdahalesi gerekmezken, IoT teknolojisinde insan-makine etkileşimi de dahil olabilir.

IoT; sağlık, ev otomasyonu, tarım, şehircilik, hayvancılık, endüstri, güvenlik gibi birçok alanda kullanılmaktadır.

Endüstri alanında Endüstri 4.0 olarak tanımlanan IoT teknolojisiyle, akıllı üretim sistemlerinin geliştirilmesi ve buna bağlı olarak maliyetlerin ve üretim için ihtiyaç duyulan enerji miktarının azaltılması sağlanmaktadır. Bu tür sistemlerin geliştirilmesiyle üretim miktarı ve kalitesinin arttırılması hedeflenmektedir. Bu hedeflere ulaşmak için sensörlerden alınan veriler toplanmakta ve bu veriler "Büyük Veri"yi oluşturarak veri analizi yapabilecek sunucu sistemlerine ya da bulut bilişim servislerine yönlendirilmektedir. Makine öğrenimi yöntemleriyle analiz edilen veriler ilgili alanda iyileştirmelerin yapılmasına katkı sağlamaktadırlar (Bozuklu, 2016). Endüstri 4.0 bir başka deyişle insanlardan neredeyse bağımsız olarak kendi kendilerini yöneterek üretim yapabilecek "akıllı fabrikalar" demektir (Bozkurt ve Durdu, 2016).

Ev otomasyonu alanında güvenlik sistemi olarak kullanılmasının yanında ev içi enerji sistemlerinin uzaktan kontrolünün sağlanmasıyla akıllı evler kavramının ortaya çıkması IoT teknolojisiyle sağlanmıştır. Bu sayede bir mobil cihaz yardımıyla neredeyse ev içi tüm nesneler uzaktan izlenebilir ve kontrol edilebilir duruma gelmiştir (Gündüz ve Daş, 2018).

Sağlık alanı, IoT teknolojisinin kullanıldığı ve yenilikçi çalışmaların yapıldığı önemli uygulama alanlarından biridir. Hastadan farklı cihazlar yardımıyla toplanan verilerin analiz edilerek hastanın uzaktan takibinin yapılabilmesi ve böylece acil durumlarda anında duruma müdahale edilebilmesinin sağlanması en önemli uygulama alanlarındandır (Sezer ve diğerleri, 2018). Halk Sağlığı Genel Müdürlüğü tarafından hizmete sunulan Aşı Takip Sistemi, tüm aşı saklama ve taşıma araçlarında bulunan sıcaklık ölçüm cihazlarından veri toplamakta ve bu verileri merkezi bir sunucuya göndererek kayıt altına almaktadır. Aşının kişiye uygulanana kadar ki tüm geçmiş sıcaklık değerleri kayıt altına alınarak kontrol edilmekte ve aşının kişiye uygulanmasından önce yapılan aşı uygunluk sorgulamasında aşı sıcaklık değerinde bir bozulma olduğu tespit edilirse aşının uygulanmasına izin verilmemektedir.[2]

[2] Sağlık Bakanlığı Aşı Takip Sistemi için hazırlanmış olan *https://asi.saglik.gov.tr/* internet adresinden alınmıştır.

Şehircilik alanında; trafik, hava durumu, altyapı hizmetleri, toplu taşıma gibi şehri oluşturan unsurların IoT teknolojisiyle takip edilmesi ve böylece insanların bu bilgileri kullanarak zaman ve kaynak tasarrufu sağlamaları hedeflenmektedir (Köseoğlu ve Demirci, 2018). Şehirlerde trafik yoğunluğunu takip eden sistemler sayesinde insanlar anlık olarak trafik yoğunluğuna göre oluşturulan alternatif rotaları kullanabilmekte, böylece insanların yaşam kalitesinin arttırılması sağlanmaktadır.

2.2.1. Nesnelerin İnternetinde Kullanılan Teknolojiler

IoT, içinde bir çok farklı teknolojiyi barındırmaktadır. Birbirleriyle veri alışverişi yapabilen algılayıcılar, otomatik tanımlama sistemleri, toplanan verileri işleyebilen sunucular, bu verilerin tutulduğu veritabanları ile tüm bu nesnelerin birbirleriyle haberleşmesini sağlayan iletişim protokolleri bu sistemin en önemli parçalarıdır. Fiziksel ortam değişikliklerini algılayan cihazlar, nesneleri ve canlıları tanımlamaya yarayan otomatik tanımlama sistemleri ve haberleşmeyi sağlayan teknolojiler ilerleyen kısımlarda detaylı olarak incelenmiştir.

2.2.1.1. Algılayıcılar

Fiziksel ortam (sıcaklık, basınç, uzaklık vb.) değişikliklerini algılayan cihazlara algılayıcı denilmektedir. Algılayıcılar sayesinde bir çok alanda otomasyona geçilmiştir. Seracılık alanında seranın nemi azaldığında insan müdahalesine gerek kalmadan sisleyicilerin çalışması, seranın ısısı belli bir değeri geçtiğinde havalandırmanın devreye girerek ortam sıcaklığının düzenlenmesi algılayıcılar sayesinde mümkün olmaktadır. Elimizi yaklaştırdığımızda otomatik olarak açılan musluklar, otomatik kapılar, duman dedektörleri gibi sistemlerin temelini algılayıcılar oluşturmaktadır. Basit bir karar mekanizmasıyla oluşturulan bu sistemlerde algılayıcıdan gelen veriler elektronik sinyallere çevrilerek yorumlanmakta ve karar mekanizmasına uyan koşula göre belirtilmiş işlemlerin yapılması sağlanmaktadır. Musluğa yaklaşıldığında hareketin algılanması ve musluk içerisinde bulunan elektronik vananın açılarak suyun akışının sağlanması bu sistemlere bir örnek oluşturmaktadır. Basit bir karar mekanizmasıyla çalışan bu sistemlerin daha karmaşık kararlar verebilmesi istendiğinde algılayıcılardan alınan bu sinyaller daha güçlü sistemlere gönderilerek analiz edilmekte ve üretilen bilgiye göre yapılması gereken işlemler belirlenmektedir. Birden çok algılayıcıdan oluşan sistemlerde de bu algılayıcılardan alınan veriler toplanarak veri işleyebilen bir sisteme gönderilmekte ve tüm bu veriler

analiz edilmektedir. Trafik yoğunluğunu analiz edebilmek, alternatif rotalar önerebilmek, öngörülen seyahat sürelerini hesaplayabilmek ve en önemlisi geleceğe dönük şehir planlaması yapabilmek için trafik verilerini toplayabilen algılayıcılar yollara ya da asfalt altlarına konumlandırılmakta ve toplanan bu veriler analiz edilerek istenilen bilgilere ulaşılmaktadır. Bu sistemlere akıllı sistemler denmektedir. Son yıllarda çokça duyduğumuz akıllı şehirler, akıllı fabrikalar, akıllı hastaneler gibi kavramların temelini bu algılayıcılar ve bu algılayıcıların üretmiş olduğu verileri işleyebilen akıllı sistemler oluşturmaktadır. Bazı alanlar ve bu alanlarda algılanabilecek örnek büyüklükler;

- Mekanik : Uzunluk, alan, miktar, kütlesel akış, kuvvet, tork (moment), basınç, hız, ivme, pozisyon, ses dalgaboyu ve yoğunluğu
- Termal : Sıcaklık, ısı akışı
- Elektriksel : Voltaj, akım, direnç, endüktans, kapasitans, dielektrik katsayısı, polarizasyon, elektrik alanı ve frekans
- Manyetik : Alan yoğunluğu, akı yoğunlugu, manyetik moment, geçirgenlik
- Işıma : Yoğunluk, dalgaboyu, polarizasyon, faz, yansıtma, gönderme
- Kimyasal : Yoğunlaşma, içerik, oksidasyon/redaksiyon, reaksiyon hızı, pH miktarı

olarak sınıflandırılmaktadır (Özcan, 2011).

Dijital algılayıcılar, ayrık sinyaller üretmektedirler. Dijital algılayıcılardan alacağımız bilgiler belli adımlarla yükselen değerlere sahiptir.

Analog algılayıcılar, fiziksel büyüklükleri kendisine referans olarak verilen akım ya da gerilim değerleri arasında bir değeri çıktı olarak veren sensörlerdir.

Pasif algılayıcılar, çevrelerinden aldıkları sinyalleri ölçen algılayıcılardır. Bir sinyal gelmediği sürece durağan pozisyonda olmaktadırlar. Anahtar tipi algılayıcılar bu tipte algılayıcılardandır.

Aktif algılayıcılar, sinyallerini kendileri üretip bu sinyallerin dış ortamla etkileşimlerini ölçen algılayıcılardır. Mesafe algılayıcıları bu tipte algılayıcılardandır (Özcan, 2011).

2.2.1.2. Haberleşme Teknolojileri

IoT için haberleşme teknolojileri şüphesiz ki en önemli unsurların başında gelmektedir. Günümüzde birçok iletişim teknolojisi bulunmakla beraber uygulamaya bağlı olarak güç talebi, veri boyutu, güvenlik gibi faktörler göz önünde bulundurularak uygulama için en uygun haberleşme teknolojisinin seçimi önem arz etmektedir (Gültunca, 2018).

Farklı mekânlardaki sıcaklık, nem, ışık, ses, basınç, kirlilik, toprak bileşimi, gürültü seviyesi, titreşim, nesne hareketleri gibi fiziksel ya da çevresel koşulları kooperatif bir şekilde izlemek için algılayıcı kullanan ve birbirinden bağımsız çalışan araçlar içeren kablosuz ağlara "Kablosuz Algılayıcı Ağ" denmektedir (Akyıldız, 2002). Bu ağlar içerisinde kullanılan veri toplama ve ağdaki diğer bağlantılı düğümlerle haberleşme yeteneklerine sahip düğümlere algılayıcı düğümler denilmektedir. Algılayıcıların, algılama ve veri işleme gibi giderek artan sayıdaki özellikleri kablosuz algılayıcı ağlarının da yaygınlaşmasının önünü açmaktadır. Başlıca kullanım alanları; tıbbi hizmetler, akıllı sistemler, akıllı şehirler olarak sıralanmaktadır (Özdağ, 2017). Uygulama alanlarının çeşitlenmesi, etkili iletişim protokellerine duyulan ihtayacı arttırmış ve bu protokollerin çeşitlenmesine olanak sağlamıştır.

IoT alanında kullanılan protokoller uygulama katmanı ve veri bağlantı katmanı protokolleri olmak üzere sınıflandırılmaktadır. AMQP, MQTT, COAP ve XMPP; IoT teknolojisinde en çok yararlanılan uygulama katmanı protokolleri olup ilerleyen kısımda öncelikle bu protokeller hakkında bilgi verilmektedir. Sonrasında Zigbee, Bluetooth, 6LoWPAN, Hücresel ve Wi-Fi olarak isimlendirilmiş olan veri bağlantı katmanı protokolleri incelenmektedir.

AMQP

AMQP (Advanced Message Queuing Protocol – İleri Mesaj Dizisi Protokolü), uygulama katmanı protokolüdür. Mesaj kuyruklama yapısını uygulayarak asenkron olarak gönderilen mesajları işlemeyi sağlamaktadır. Mesaj kuyruklama teknolojisi asenkron işlenmesi gereken işlerin sıraya alınarak uygulanmasıdır. Örneğin bir e-ticaret sitesinde satış işlemi sonrasında arka planda, stok düşümü, kargo kaydı, faturalandırma gibi işlemler istenmeyen bekleme sürelerine sebebiyet verebilmektedir. Bu aşamada mesaj kuyruklama protokolleri sayesinde

her işlem ilgili mesaj kuyruğuna yönlendirilmekte ve asenkron olarak tüm işlemlerin bitmesi sağlanmaktadır. Tüm bu işlemler yapılırken kullanıcının işlem sonucunu almak için işlemi gerçekleştirdiği internet sitesinde ya da uygulamada beklemesine gerek olmamaktadır. Tüm işlemler bittiğinde kullanıcıya ilgili kuyruktan bilgilendirme maili ya da mesajı atılmaktadır. Böylece kullanıcının bekleme süresinden kaynaklanan kötü bir deneyim yaşamasının önüne geçilmektedir. AMQP protokolü mesajların kuyruklanarak iletilmesine bağlı bir haberleşme yöntemidir.

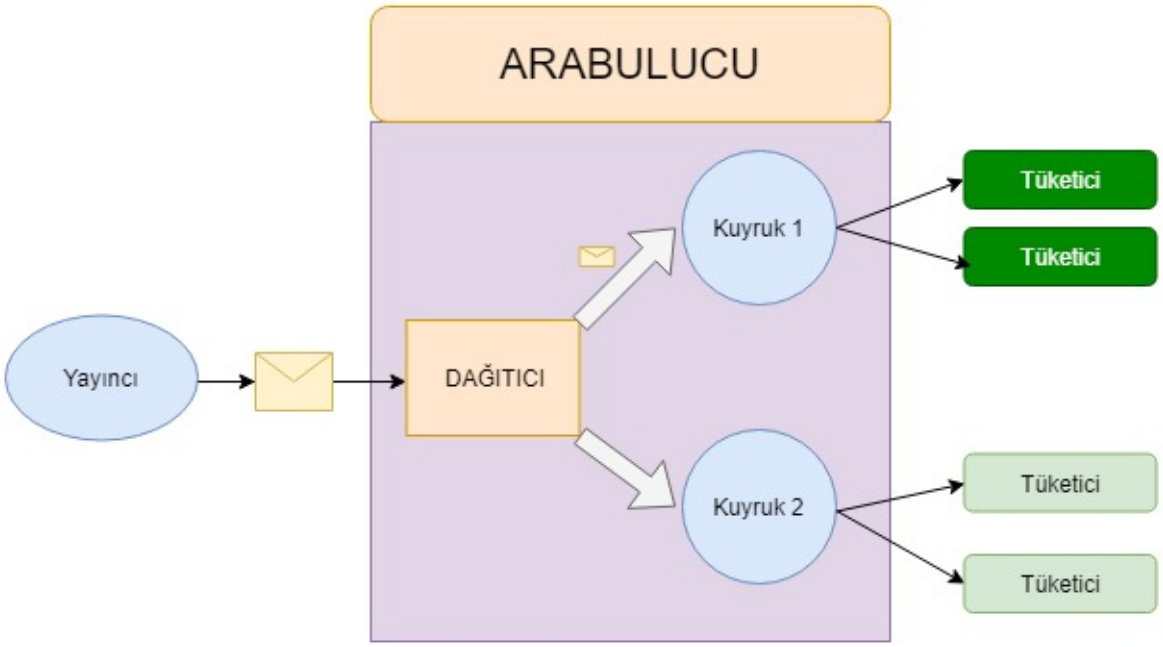
Mesajların hatadan bağımsız ve istenildiği şekilde gönderilmesi temel amaç olarak belirlenmiştir. AMQP protokolü, belirleyici özellikleri olan güvenilirlik, güvenlik (noktadan noktaya ve publish-subscribe), gelişmiş mesaj yönlendirme gibi özellikleriyle çoğunlukla finans sektöründe kullanılmaktadır. AMQP protokolü, banka endüstrisi için güvenilir bir mesajlaşma protokolü olup mesaj kaybına tahammülü yoktur. Mesajın alınıp işlendiğine dair göndericiye bilgi verilmektedir. Güvenilir noktadan noktaya bağlantı için TCP üzerinden çalışmaktadır. IoT tarafında ise AMQP kontrol katmanı veya sunucu tabanlı analiz fonksiyonları için uygundur. Daha çok sunucudan sunucuya (S2S) sistemler için hizmet görür (Gültunca, 2018).

AMQP protokolünün nasıl çalıştığını anlayabilmek için protokol kapsamında kullanılan bazı kavramları bilmek gerekmektedir. Aşağıda protokolün temel bileşenlerinin tanımı yapılmıştır:

- Arabulucu (broker) : Haberleşme trafiğinin kontrol merkezidir.
- İleti (message) : Yayıncı tarafından oluşturulan ve broker üzerine gönderilerek abone olan uygulama ya da cihazlara dağıtılan veriye "İleti" denilmektedir.
- Tüketici (consumer) : Yayıncı tarafından Arabulucu üzerine gönderilen İletiyi alan uygulama ya da cihazlara "Tüketici" denilmektedir.
- Yayıncı (publisher) : İleti içeriğini oluşturan ve Arabulucu üzerine gönderen uygulama ya da cihazlara "Yayıncı" denilmektedir.

AMQP protokolünün çalışma mantığı Şekil 2.1'de gösterildiği gibidir.

- Yayıncı tarafından üretilen ileti arabulucu içerisinde bulunan dağıtıcıya gönderilir.



Şekil 2.1: AMQP mimarisi.

- Dağıtıcı belirlenmiş olan kurallara göre iletiyi ilgili kuyruklara yönlendirir.
- İletiler kuyruklar üzerinden alıcılara yönlendirilerek mesaj iletimi tamamlanır.

İletilerin nasıl yönlendirileceği, hangi şekilde sıralanacağı ve tüm bu işlemlerin sonucunda bu iletileri gönderen ve alan uygulamaların belirlenmiş kurallar çerçevesinde nasıl davranması gerektiği aşağıda belirtilen AMQP bileşenleri tarafından sağlanmaktadır:

- Dağıtıcı (exchange): Arabulucunun bir parçası olup gönderilen iletileri karşılamakta ve ilgili ileti kuyruklarına yönlendirme işlemini gerçekleştirmektedir.
- İleti Kuyruğu (queue): Alıcıların bağlanmış olduğu ve iletilerin dağıtıcı tarafından yönlendirildiği bağlantılardır.
- Bağlantılar (bindings): İletileri dağıtıcı üzerinden ileti kuyruklarına yönlendirmek için oluşturulmuş kurallardır.

Yayıncılar tarafından oluşturulan iletiler, dağıtıcı tarafından karşılanmakta ve dağıtıcı iletileri işleyerek bir ya da birden çok ileti kuyruğuna bu mesajları yönlendirmektedir. İleti kuyruklarına yönlendirme işlemi dağıtıcı tipine bağlı olarak değişmektedir. Dört farklı yapıda dağıtıcı tipi uygulanabilmektedir.

Noktadan-Noktaya Dağıtım (Direct Exchange) kullanıldığında, iletiye bir yönlendirme anahtarı (routing key) eklenmekte ve dağıtıcı gelen yönlendirme anahtarına göre iletiyi ilgili ileti kuyruğuna yönlendirmektedir. Bu mesajlaşma mimarisi kullanıldığında gelen ileti sadece bir alıcı tarafından işlenmektedir. İletinin sadece belirlenmiş bir tüketiciye gönderilmesi istendiğinde tercih edilen mimaridir.

Toplu Dağıtım (Fanout Exchange) iletilerdeki yönlendirme anahtarı alanını dikkate almamakta ve bağlı olan tüm ileti kuyruklarına gelen iletiyi göndermektedir. Birden çok tüketiciye aynı iletinin gönderilmesi gerektiği durumlarda kullanılmaktadır.

Konu Bazlı Dağıtım (Topic Exchange) mimaride nokta karakteriyle ayrılmış bir kelime listesi ileti içerisinde iletilmektedir. Kelimeler 255 byte uzunluğa kadar olabilmektedir. İleti kuyruğu bağlantı anahtarları (binding key) da aynı şekilde nokta karakterleriyle ayrılmış yapıda olmalıdır. Yönlendirme anahtarı içerisinde * ve # karakterleri kullanılarak iletinin hangi kuyruğa gitmesi gerektiğine dağıtıcı tarafından karar verilmektedir. Eğer bir ileti kuyruğuna # karakteriyle bağlantı anahtarı eklenmişse, dağıtıcı tarafından gönderilen tüm iletiler alınmaktadır. Bu şekilde bir mesajlaşma toplu dağıtım mimarisine benzer şekilde davranmaktadır. İleti kuyruğu için * ve # karakterlerinden hiçbiri yönlendirme anahtarı içerisinde kullanılmamışsa bu durumda noktadan noktaya dağıtım mimarisine benzer şekilde bir davranış sergilemektedir.

Başlık Bazlı Dağıtım (Header Exchange) mimaride iletinin başlığına başlık değerleri eklenerek yollanmaktadır. Dağıtıcı, gelen iletinin başlık değerlerine bakarak şablona uyan kuyruklara iletileri yönlendirmektedir. Eşleşme tipi (x-match argümanı) *Any* ya da *All* olmalıdır.

MQTT

MQTT (Message Queuing Telemetry Transport - Telemetri Mesajlaşma Protokolü), mesajların cihazlar ya da uygulamalar arasında aktarılması için kullanılan bir uygulama katmanı protokolüdür. Abone olma ve yayınlama mantığına dayalı olarak çalışmaktadır. Bu haberleşme trafiğini kontrol eden yöneticiye arabulucu (broker), mesaj yayınına yayınlama (publish) ve bu mesaj yayınına abone olanlara alıcı (subscriber) denmektedir. Güvenlik, şifreleme gibi özellikler TCP/IP protokolü üzerinden sağlandığı için basit bir yapısı

vardır. Konu (topic) bilgisine bağlı bir haberleşme yapılmaktadır. Arabulucu birimine bağlı birimler bir konu başlığına veri gönderebilmekte ya da bir konu başlığına bağlı verileri alabilmektedirler. Arabulucu gelen mesajları konu bilgisine bakarak ilgili alıcı birimlere iletmekten sorumludur.

MQTT protokolünün genel özellikleri;

- Alıcı ve verici cihazlar arasında mesaj haberleşmesini kontrol eden arabulucu bulunmaktadır. Tüm cihazlar arabulucu üzerinden haberleşir.
- Kaynak tüketimini en az seviyede tutmak adına hafif bir protokol olarak tasarlanmıştır.
- Hafif bir protokol oluşu nedeniyle sınırlı kaynaklı uygulamalarda kullanımı uygundur.
- Yayıncı ve alıcı birimleri üzerinde kullanılabilecek bir çok açık kaynak kodlu kütüphane mevcuttur.
- Arabulucu olarak kullanılabilecek *Mosquitto* gibi açık kaynak kodlu uygulamalar vardır.
- Farklı iletişim modellerini destekler.
  - 1:1, 1:N, M:N
- Gönderilen mesajın iletilmesinin önemine göre belirlenebilecek üç farklı Servis Kalitesi (QoS) seçeneği vardır.
  - QoS 0: Mesajın teslim edildi bilgisini almaz. Mesaj sadece bir kez gönderilir. Bu seçenekle çalıştırılan sistem ağ kaynaklarını en az düzeyde kullanır.
  - QoS 1: Mesajın en az bir kez teslim alındığını garantiler. Gönderici mesajı tutar ve alıcıdan mesajı aldığına dair bir onay (PUBACK) paketi bekler. Onay paketi alınmadığında tekrar mesajı göndermeyi dener. Bu seçenek uygulandığında alıcı gönderilen mesajı birden fazla kez alabilir.
  - QoS 2: Mesajın bir kez ulaştığını ve daha fazla sayıda gönderilmediğini garanti eder. Ağ kaynaklarını diğer iki seçeneğe göre en çok tüketen seçenektir. Gönderici alıcıdan mesajın alındığına dair onay (PUBREC) paketi bekler, paket gelmezse belli bir süre sonra tekrar (DUP) etiketiyle yeniden mesajı

gönderir. Onay paketi alındığında gönderici göndermiş olduğu mesajı kaldırır ve bilgi (PUBREL) paketi göndererek alıcıya bilgi gönderir. Alıcı tamanlandı (PUBCOMP) paketiyle göndericiye, gönderici de aynı şekilde tamamlandı paketiyle alıcıya işlemin tamamlandığı bilgisini döner.

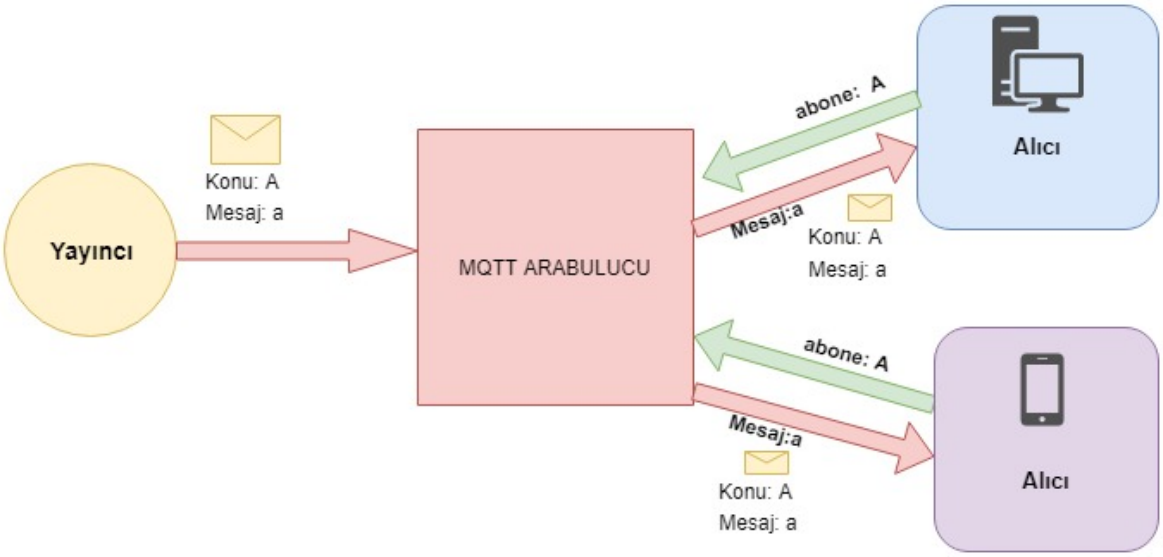
- Mesajlar sabit iki byte büyüklüğünde başlığa (header) sahiptir. HTTP protokolüyle kıyaslandığında bu özellik ağ üzerinde oluşan yükün azalmasına iyi yönde etki etmektedir.
- TCP/IP bağlantısı üzerinde TLS/SSL desteği sunar.

olarak sıralanmıştır (Gültunca, 2018).

MQTT protokolünün nasıl çalıştığını anlayabilmek için protokol kapsamında kullanılan bazı kavramları bilmek gerekmektedir. Aşağıda protokolün temel bileşenlerinin tanımı yapılmıştır.

- Konu (topic) : Mesajlar istemciler arasında konu başlıklarıyla iletilmektedir; yayıncı mesaja konu başlığı eklemekte ve bu konu başlığıyla mesaj bekleyen alıcılara mesajlar yönlendirilmektedir.
- Arabulucu (broker) : Haberleşme trafiğinin kontrol merkezi olup, gelen mesajları almakta, filtrelemekte ve mesajın konu bilgisine bakarak hangi alıcılara mesajı göndereceğine karar vermektedir.
- Mesaj (message) : Yayıncı tarafından oluşturulan ve arabulucu üzerine gönderilerek abone olan uygulama ya da cihazlara dağıtılan veriye mesaj denilmektedir.
- Alıcı (subscriber): Yayıncı tarafından arabulucu üzerine gönderilen mesajı alan uygulama ya da cihazlara alıcı denilmektedir.
- Yayıncı (publisher) : Mesaj içeriğini oluşturan ve arabulucu üzerine gönderen uygulama ya da cihazlara yayıncı denilmektedir.
- Müşteri (client) : Arabulucu birimine abone olmuş yayıncı ve alıcılara denilmektedir.

MQTT protokolünün çalışma mantığı aşağıdaki Şekil 2.2'de gösterildiği gibidir.



Şekil 2.2: MQTT mesaj iletimi.

MQTT protokolünde en önemli unsur arabulucu birimidir. Haberleşme, arabulucu tarafından belirlenmiş varsayılan bir port ya da özel bir port numarası belirlenerek yapılabilmektedir. Varsayılan port numaraları; şifreli bağlantılar için 8883 ve şifresiz bağlantılar için 1883 olarak belirlenmiştir. Arabulucu yayıncıdan gelen mesajları alıcılara ulaştırmaktan sorumludur. Bu sorumluluğunu gelen mesajları konu başlıklarına göre filtreleyerek ve ilgili konuya abone olan alıcılara bu mesajları ileterek gerçekleştirmektedir. Alıcılar, bu mesajları almak için arabulucu birimine önceden belirlenmiş konu başlıklarıyla abone olmaktadırlar.

Konu bir kanal vazifesi görmekte olup bu kanalın yönetimi arabulucu birimindedir. Böylece yayıncıların ve alıcıların birbirlerini tanımasına gerek olmamaktadır. Yayıncı ve alıcı sadece arabulucu birimini tanımaktadır. Bu durum MQTT protokolü üzerinden çalışan sistemlerin kolay ölçeklenebilir olmasına olanak sağlamaktadır.

Yayıncılar, mesaj yayınlayan birimlerdir. Mesajları konu bilgisiyle beraber arabulucu birimine iletmekte, arabulucu birimi de bu mesajı konuya abone olmuş alıcılara yönlendirmektedir.

Müşteriler arabulucu birimiyle TCP/IP bağlantısı üzerinden haberleşmektedirler. Yayıncı tarafından oluşturulan mesaj bir konuya bağlı olarak MQTT arabulucu birimine yönlendirilmektedir. Arabulucu gelen mesajı almakta ve konu başlığına bakarak hangi alıcılara yönlendireceğine karar vermektedir. Arabulucu birimi sadece göndereceği mesajın

konu başlığına abone olmuş alıcılara mesajı iletmektedir.

Hafif bir protokol oluşu sayesinde düşük güç tüketimi sağladığından IoT temelli uygulamalarda özellikle tercih edilmektedir.

CoAP

CoAP (Constrained Application Protocol – Kısıtlı Uygulama Protokolü), IETF (Internet Engineering Task Force - İnternet Mühendisliği Görev Grubu) tarafından geliştirilmiş bir uygulama katmanı protokolüdür. Kısıtlı kaynaklara sahip cihazlar ve sınırlı bant genişliğine sahip ağlarda kullanılmak üzere tasarlanmıştır. Kısıtlı kaynakların etkin kullanımı için varsayılan olarak UDP (User Datagram Protocol - Kullanıcı Veribloğu İletişim Kuralları) üzerinde çalışmaktadır. TCP (Transmission Control Protocol - İletim Kontrol Protokolü) veya diğer taşıma katmanı protokolleri üzerinde çalışabilecek şekilde de düzenlenebilmektedir (Chen, 2016). Protokol özellikle akıllı endüstriyel sistemler ve ev otomasyonları gibi makineden-makineye (M2M) uygulamalarda kullanılmak üzere tasarlanmıştır.

CoAP, HTTP (Hyper Text Transfer Protocol - Hiper Metin İletim Protokolü) protokolüne benzer bir döküman transfer protokolüdür. CoAP'ın etkileşim modeli HTTP'nin sunmuş olduğu istemci/sunucu modeline benzemekle beraber CoAP makineden-makineye uygulamalarda kullanılmakta olduğundan CoAP hem istemci hem de sunucu rollerini üstlenmektedir (Gültunca, 2018). HTTP protokolüyle en önemli fark CoAP'ın daha kısıtlı kaynaklara sahip cihazlar için geliştirilmiş olmasıdır (Kantekin, 2018). CoAP, *GET, PUT, POST, DELETE* metotlarını HTTP üzerindeki kullanımına benzer şekilde kullanmakta ancak HTTP'den farklı olarak UDP üzerinden asenkron haberleşme sağlamaktadır. Servisler ve kaynakların keşfi için yerleşik desteğe sahiptir.

Güvenirlilik onay gerektiren mesajlar gönderilerek sağlanmaktadır. Zaman aşımı mekanizmasıyla onay mesajı alınana kadar mesaj gönderilmeye devam edilmektedir. Gönderilen mesaja ait onay mesajı verilemeyecek yani istek işlenemeyecek ise sıfırlama mesajı gönderilmektedir. Gönderilen mesajın karşılığında bir onay mesajı alınması gerekmiyorsa, onay gerektirmeyen mesaj türünde iletim yapılmaktadır. Bu durum dört farklı mesajlaşma şekliyle gerçekleştirilmektedir:

- Onay gerektiren (confirmable): Karşılığında bir onay mesajı beklenen mesajlardır.
- Onay gerektirmeyen (non-confirmable): Karşılığında bir onay mesajı beklenmeyen mesajlardır.
- Onay (acknowledgment): Onay gerektiren mesajlara karşılık olarak gönderilir.
- Sıfırlama (reset): Bir isteğin yerine getirelemediği zaman ya da bir cihazın erişilebilir olup olmadığı kontrol edilmek istendiğinde kullanılan mesaj türüdür.

CoAP, dört byte uzunluğunda bir başlığa sahiptir. Mesaj bir kaç kısımdan oluşmaktadır (Şekil 2.3):

Şekil 2.3: CoAP mesaj içeriği.

CoAP başlığının içeriği;

- Ver: İki bit uzunluğunda CoAP versiyon numarasıdır.
- Tip: Mesaj tipini belirten iki bit uzunluğundaki alandır.
  - Onay gerektiren: 0
  - Onay gerektirmeyen: 1
  - Onay: 2
  - Reset: 3
- Jeton Uzunluğu: Değişken uzunlukta olabilen jeton boyutunu belirten dört bit uzunluğundaki alandır.

- Kod: İstek, hata ve cevap bilgisinin tutulduğu sekiz bitlik bir alandır.
- Mesaj Kimliği: Mesaj tekrarlarını engellemek için mesaja ait kimlik bilgisinin tutulduğu onaltı bitlik bir alandır.

şeklinde beş alandan oluşmaktadır (Gültunca, 2018).

XMPP

XMPP (Extensible Messaging and Presence Protocol- Genişletilebilir Mesajlaşma ve Varlık Protokolü), internet üzerinde bulunan iki ucun herhangi bir bilgiyi birbirleri arasında karşılıklı ve eş zamanlı aktarabilmelerini sağlayan XML protokol ve teknolojilerinden oluşan bir yapıdır. Kullanıcıların birbiriyle metin tabanlı mesaj üzerinden iletişim kurabilmelerini sağlamak amacıyla anlık bir mesajlaşma programı olarak geliştirilmiştir. XMPP adından anlaşılabileceği üzere genişletilebilir bir protokoldür ve genişletilebilir olması metin tabanlı anlık mesajlaşma uygulamaları dışında içerik paylaşımı, dosya paylaşımı, oyun oynama gibi birçok yeni fikrin uygulanmasına olanak sağlamıştır (Seçkin, 2011).

XMPP'nin genel özellikleri;

- 1998 yılında geliştirilmeye başlanmış ve günümüzde yüksek oranda kararlı bir duruma gelmiştir.
- Kullanıma açıktır ve özgürdür, istemciler, sunucular, bileşenler ve kod kütüphaneleri için topluluk tarafından geliştirilmiş birden fazla uygulama mevcuttur.
- Standarttır: IETF'nin standartlaşma sürecine uygun olarak XMPP standartlar kuruluşu tarafından geliştirilmeye devam etmektedir.
- Güvenlidir: Herhangi bir XMPP sunucu, XMPP ağından tamamen izole edilebilir. Örneğin sadece yerel bir ağ içerisinde kullanılabilir.
- Genişletilebilir: Herhangi birisi XML'den faydalanarak çekirdek protokol üzerine kendi istediği fonksiyonları inşa edebilir.

- Esnektir: Mesajlaşmanın dışında; ağ yönetmek, dosya ve içerik paylaşmak, uzaktaki sistemlerin durumunu görüntülemek, oyun oynamak gibi birçok farklı alanda kullanılabilir.

olarak sıralanmıştır (Gültunca, 2018).

Zigbee

Zigbee, kişisel alan ağları, IEEE 802.15.4 standardına dayanan, düşük güç tüketen, üst düzey haberleşme protokolleri için uygun bir standarttır. Zigbee standardının temel özellikleri;

- Düşük güç tüketimi,
- Düşük veri hızı,
- Düşük maliyet,
- Tek bir ağ için 65000 düğüm desteği,
- Wi-Fi ve Bluetooth ile kıyaslandığında daha küçük paket kullanımı,
- Kendi ağında otomatik olarak kurulabilmesi

olarak verilmiştir (Ajgaonkar, 2010).

Zigbee standardı ile uygun ortam koşullarında ve radyo alıcı-vericinin gücüne bağlı olarak 1600 metreye kadar veri iletimi mümkündür (Uğuz ve diğerleri, 2013). Enerji tüketiminin düşük oluşu özellikle elektrik enerjisinin sağlanmasının zor olduğu coğrafi ortamlarda önemli bir avantaj sağlamaktadır. Diğer protokellere kıyasla veri iletim hızı çok daha düşük olduğundan yüksek veri hızı gerektiren endüstriyel uygulamalar için tercih edilmemektedir. Düşük güç gerektiren, düşük veri hızında seyrek veri alışverişinin yapıldığı uygulamalar başlıca kullanım alanlarını oluşturmaktadır. Düşük maliyet, güvenirlilik ve ağ üzerindeki düğüm sayısının fazla oluşu diğer protokollere oranla Zigbee' yi öne çıkaran özelliklerdir.

Bluetooth

Bluetooth, 1994 yıllında Ericsson firması tarafından geliştirilen ve kısa mesafeli haberleşme yapılmasına olanak sağlayan bir iletişim protokolüdür. Kablosuz dosya transferi yapabilmek ve cihazları birbirine bağlamak amacıyla kullanılmaktadır (Aydalka, 2018).

Kısa dalga boyuna sahip radyo dalgaları kullanılarak iki veya daha fazla cihazın birbirlerinin görüş alanında olmasına gerek olmadan haberleşme ve veri alışverişi yapabilmesine olanak sağlamaktadır. 10 metre mesafede etkili olmakla birlikte çeşitli yollarla bu mesafe 100 metreye kadar arttırılabilmektedir (Yücel, 2015).

IoT teknolojisinin yaygınlaşmaya başlamasıyla Bluetooth protokolüne de eklemeler yapılmış ve Düşük Enerjili Bluetooth (Bluetooth Low Energy) geliştirilmiştir. Düşük Enerjili Bluetooth, Bluetooth 4.0' ın bir parçaşı olarak 2010 yılında duyurulmuş ve güvenlik, sağlık, tüketici elektroniği gibi çok fazla alanda az enerji tüketimi sebebiyle kullanılmaya başlanmıştır (Aydalka, 2018).

Düşük Enerjili Bluetooth, istemci-sunucu modeliyle haberleşmektedir. Bu modelde algılayıcılar sunucu rolünü üstlenmekteyken bilgisayar veya telefonlar da istemci rolünü üstlenmektedir. Örneğin Bluetooth protokolünü kullanan giyilebilir bir sensör, telefonda bulunan uygulamaya veri göndererek verinin telefon aracılığıyla uzak bir sunucuya gönderilmesini sağlayabilmektedir.

6LoWPAN

IEEE 802.15.4 standardına dayalı olarak cihazlar arasında IPv6 bağlantılı kablosuz ağ sağlayan bir iletişim protokolüdür. Büyük ağ yapılarını desteklemek için örgü teknolojisini kullanmaktadır. 6LoWPAN protokolü Ethernet, Wi-Fi, GPRS ve uydu gibi diğer IP ağlarına bağlanabilmekte ve yönlendiriciler aracılığıyla en kısa yoldan en iyi sinyali iletmeye olanak sağlamaktadır.

6LoWPAN iletişim protokolünün avantajları ve özellikleri;

- Kod ve belleğin minimum şekilde kullanımı,
- Doğrudan uçtan uca (end-to-end) internet entegrasyonu,

- Çoklu topoloji seçenekleri,
- Uzun batarya ömrü (1000 güne kadar),
- Yüksek veri kapasitesi,
- Orta mesafe kapsama alanı (75m),
- 16 ve 64 bitlik 802.15.4 adreslemeyi destekler,
- Unicast, multicast ve broadcast desteği,
- Fragmantasyon (1280 byte'lık IPv6 maksimum iletim birimini 127 byte'lık 802.15.4 çerçeveleri şeklinde parçalama),
- IP routing desteği,
- Link katmanında mesh yapısı desteği (802.15.5)

olarak belirtilmektedir (Demir ve diğerleri, 2016).

Hücresel

Hücresel haberleşme sistemleri, spektral verimlilik ve kullanıcı kapasitesinin arttırılması için geliştirilmiş bir tekniktir (Geylani ve diğerleri, 2016). Hücresel haberleşmede hizmet verilecek alan güçlü bir tek baz istasyonu yerine, küçük ve kapsama alanı daha düşük baz istasyonlarıyla beslenmektedir. Bu küçük baz istasyonlarına hücre denilmektedir. Veri hücreler aracılığıyla hedefe ulaştırılmaktadır.

Uzun mesafeli iletim yapılması gereken uygulamalarda 3G ve 4G hücresel iletişim seçenekleri tercih edilebilmektedir. Pil tüketiminin az olması istenen, küçük veri iletimi yapılacak uygulamalarda 3G bağlantı tercih sebebi olmakla beraber daha büyük verilerin hızlı bir şekilde iletilmesi hedefleniyorsa 4G bağlantı seçeneği kullanılabilmektedir.

Wi-Fi

Kablosuz Bağlantı Alanı (Wi-Fi), kişisel bilgisayarlar, mobil cihazlar ve oyun konsolları gibi cihazların kablosuz olarak birbirlerine bağlanmasını sağlayan bir haberleşme protokolüdür.

IEEE (The Institute of Electrical and Electronics Engineers - Elektrik ve Elektronik Mühendisleri Enstitüsü) 802.11 standardı olarak IEEE 802.3 Ethernet standardının kablosuz versiyonu olarak kabul edilmiştir (Hendriks, 2016).

Kablosuz Bağlantı Alanı sayesinde büyük veri aktarımı yapılabildiğinden ev kullanıcıları ve şirketler tarafından yoğun olarak kullanılmaktadır. Bu durum Kablosuz Bağlantı Alanının geniş bir altyapıya sahip olmasına olanak sağlamıştır. IoT uygulamalarında ise kullanımı güç tüketiminin fazla olmasına sebep olduğundan, büyük veri aktarımının yapılacağı ve güç tüketiminin problem yaratmadığı uygulamalarda tercih edilmektedir (Hendriks, 2016).

2.2.1.3. Otomatik Tanımlama Sistemleri

Otomatik tanımlama, nesneleri tespit etmeyi ve tanımayı sağlayan teknolojilere verilen genel bir tanımdır. Otomatik veri toplanan sistemlerde varlıkların insan müdahalesine gerek kalmadan tanınması sağlanmakta ve böylece veri girişi hataları azaltılarak verimlilik artışı sağlanmış olmaktadır. Otomatik tanımlama alanında kullanılan teknolojilerden bazıları aşağıda listelenmiştir.

- RFID (Radio Frequency Identification - Radyo Frekanslı Tanımlama),
- Barkod,
- Akıllı kart,
- Sesli tanımlama,
- Retina taraması

IoT alanında RFID teknolojisi otomatik tanımlama sistemi olarak tercih edilmektedir (Aktaş, Çeken ve Erdemli, 2016). RFID, radyo dalgalarını kullanarak varlıkların tanınmasını sağlayan bir otomatik tanıma sistemidir. RFID tabanlı otomatik tanıma sistemlerini oluşturan elemanlar;

- Belirli bir varlığa atanmış tekil bir kimlik bilgisi,
- Varlık üzerine eklenmiş, veri depolama kapasitesine sahip ve çevresi ile iletişim kurabilen bir kimlik etiketi,

- Çok sayıda etiketten gelen sinyali yüksek bir hızda okuma ve doğru bir şekilde işleme yeteneğine sahip, RFID okuyucuları ve veri işleme sistemlerinden oluşan bir ağ yapısı,
- Çok sayıda ürün bilgisini depolama yeteneğine sahip ağ içinde yer alan bir veya birden fazla veri tabanı

şeklinde tanımlanmaktadır (McFarlane ve Sheffi, 2003).

Barkod teknolojisi, varlıkların etiketlenmesi ve otomatik tanımlanmasında günümüzde yaygın olarak kullanılsa da bazı kısıtlara sahiptir:

- Barkod içerisinde saklanabilen veri boyutu düşüktür.
- Barkod etiketinin okunabilmesi için okuyucunun görüş alanı içerisinde olması gerekmektedir.
- İnsan faktörü olmadan kullanımı çok zordur.

RFID etiketleri, içerisinde yüksek miktarda veriyi depolayabilmekte ve bu verileri toplu halde okuyup yazabilmektedirler. RFID okuyucular barkod okuyuculardan farklı olarak, etki alanı içinde olduğu sürece uzak mesafeden veri iletişimi sağlayabilmektedirler.

2.2.2. Nesnelerin İnterneti Güvenliği

IoT uygulamalarında güvenli haberleşmenin sağlanması için bir takım gereksinimlere ihtiyaç duyulmaktadır. Bu gereksinimler gizlilik, bütünlük, kimlik doğrulama, veri güncelliği, hizmet bütünlüğü, heterojenik ve anahtar yönetim sistemleri gibi maddeler olup bu bölümde başlıklar halinde incelenmiştir (Görmüş ve diğerleri, 2018).

2.2.2.1. Gizlilik

IoT cihazları üye olunan ağda haberleşirken ağa dahil olmayan bir nesnenin ağa erişimi engellenmelidir (Shi ve Perrig, 2004). Ağa yetkisiz erişimi engellemek için ağ üzerindeki düğümler güvenli bir haberleşme kanalını kendi aralarında sağlamalıdırlar. Güvenli iletişimi sağlamak ve ağ dışından varlıkların veriye erişimini engellemek için iletimi yapılan mesajlar şifrelenmelidir. Ağ üzerindeki trafik, kullanılan protokollere göre farklı yöntemler kullanılarak şifrelenebilir.

2.2.2.2. Veri Bütünlüğü

Kaynak tarafından iletilen bir mesajın değiştirilmeden hedefe ulaştırılmasının sağlanması veri bütünlüğünün korunmuş olmasını sağlamaktadır. İletilen mesajın değiştirilmediğinin kontrol edilmesi ve değişim algılanırsa mesajın reddedilmesi Mesaj Asıllama Kodu (MAK) gibi çeşitli yöntemler yardımıyla gerçekleştirilebilmektedir.

2.2.2.3. Kaynak Doğrulama

Kaynak Doğrulama, veri iletişimi yapan nesnelerin kimliklerini doğrulayarak birbirleriyle haberleşmelerini sağlamaktadır. Kimliğini doğrulayamayan nesnelerin haberleşmeye dahil olması engellenmektedir.

2.2.2.4. Veri Güncelliği

Ağ üzerinde iletimi yapılan verilerin yeni olması ve eski verilerin tekrar gönderilmesinin önüne geçilmesi, verinin güncel olmasını sağlamak için önem arz etmektedir. Bu konuda varsayılan yaklaşım, iletilen mesajların içerisine bir zaman damgası eklenmesidir (Görmüş ve diğerleri, 2018).

2.2.2.5. Hizmet Bütünlüğü

Ağ üzerinde toplanan verilerin bozulmadan ağ geçidine ya da verinin işleneceği cihaza aktarılabilmesi, yapılacak analiz sonuçlarının doğru çıkmasına yardımcı olmaktadır. Bu nedenle bozulmuş düğümlerden alınan verilerin tespit edilmesi ve verinin işleneceği cihazlar üzerinde bu verilerin kullanılmasının engellenmesi hizmet bütünlüğünü sağlamak adına önemlidir.

2.3. SAĞLIK KURUMLARINDA NESNELERİN İNTERNETİ TEKNOLOJİSİNİN KULLANIMI

IoT teknolojisinin en çok kullanıldığı alanlardan biri sağlık hizmetleri sektörüdür. Sağlık hizmetlerinde IoT; uzaktan hasta bakımı, dijital hastanelerde idari işlemlerin otomasyon üzerinden takip edilebilmesi, giyilebilir teknolojiler sayesinde sağlıklı yaşama destek verilmesi gibi bir çok alanda kullanılmaktadır (Alqahtani, 2018).

Hasta verilerinin algılayıcılar yardımıyla toplanması, toplanan verilerin analiz edilmek üzere uzak sunucuya gönderilmesi ve bu işlemler sonucunda tespit edilen problemlere anında cevap verilebilmesi IoT teknolojisinin sağlık hizmetlerinde kullanıldığı alanların başında gelmektedir (Can ve diğerleri, 2016).

Son yıllarda sağlık kurumlarının tüm iş süreçlerini dijital ortam üzerinden yeniden yapılandırması dijital dönüşümün uygulanmasıyla dijital hastane kavramının ortaya çıkmasını sağlamıştır (Kılıç, 2016). Dijital hastane kavramı içerisinde IoT teknolojisinden yararlanılabilmekte ve bu sayede insan faktörüne gerek kalmadan çoğu idari işlemler ve hasta bakımı işlemleri otomasyon üzerinden takip edilebilmektedir.

2.3.1. Sağlık Kurumlarında Nesnelerin İnterneti Üzerine Yapılan Çalışmalar

Elsokah ve diğerleri (2019)'nin yapmış olduğu çalışmada, sağlık kurumlarında akıllı yatakların kullanımında IoT teknolojisinin kullanımıyla ilgili bilgiler verilmektedir. Çalışmada akıllı yataklar için kullanılan algılayıcılar yardımıyla hastanın hareketlerine uygun olarak yatak, gerekli pozisyona bir sağlık çalışanı yardımına ihtiyaç duyulmadan, otomatik olarak geçmektedir.

Paiva ve diğerleri (2018)'nin yaptıkları çalışmada, hastaneler içerisinde varlık takibinin yapılabilmesi için gerçek zamanlı konum belirleme sistemi önerileri sunulmuş ve hali hazırda kullanılan çözümlere değinilmiştir.

Kanase ve Gaikwand (2016)'ın yaptığı çalışmada; hastane odalarının algılayıcılar yardımıyla sıcaklık, nem, ışık gibi fiziksel durumları ölçülerek uzak sunucuya gönderilmekte ve alınan verilere göre sistemin otomatik karar vermesi sağlanarak odanın ideal şartlarda tutulması amaçlanmaktadır.

Alumona ve diğerleri (2014) yaptıkları çalışmada, kablosuz algılayıcı ağlar teknolojisinden yararlanarak uzaktan hasta takibinin yapılmasıyla ilgili sistem önerisi sunulmuştur. Sistem sayesinde hasta bilgileri sağlık kurumu tarafından takip edilebilmekte ve acil durumda haberdar olunabilmesi sağlanmaktadır.

Shams ve diğerleri (2014)'nin yapmış olduğu çalışmada, TeleSağlık alanında IoT teknolojisinin kullanımına dair alanlar incelenmiştir. Telesağlık alanında kablosuz algılayıcı ağlar ve uzaktan tanılama cihazları sayesinde sağlık kurumunda bulunan bir hasta, uzak bir

lokasyonda olan doktor tarafından da incelenebilmektedir ve doğru teşhisin konulabilmesi adına hasta verileri konum farketmesizin diğer uzmanlarla paylaşılabilmektedir.

Al-Odat ve diğerleri (2018), hastanın değerlerini kontrol eden ve gerektiği takdirde vücuda insülin enjekte edilmesini sağlayan bir insülin pompası ve kontrol sistemi önermişlerdir.

Jaisree ve diğerleri (2017)'nin yapmış olduğu çalışmada, hasta odaları algılaycılar yardımıyla sürekli kontrol edilmekte, hastaya verilen serumların durum takibi algılayıcılar yardımıyla izlenebilmekte ve serum bittiğinde hemşireye uyarı yollanabilmektedir.

Chiuchisan ve diğerleri (2014), hastanelerin yoğun bakım ünitelerinde kullanılmak üzere bir çok algılayıcının kullanıldığı bir sistem önerisi sunmuşlardır. Önerilen sistem sayesinde yoğun bakım ünitelerinde hastanın hayati fonksiyonları ve bulunduğu odanın fiziksel koşulları sürekli kontrol altında tutularak gerektiği takdirde uyarı mekanizmalarının çalışması sağlanmakta ve böylece yoğun bakım ünitesinde çalışan personel sayısı ve insan kaynaklı hataların azaltılması mümkün olmaktadır.

Sağlık alanında yukarıda değinilmiş olan çalışmalara bakıldığında, IoT teknolojisi çoğunlukla yaşlı bakımı, sağlık verisi toplama, uzaktan hasta takibi, gerçek zamanlı lokasyon ve dijital hastane kavramı kapsamında sağlık kurumlarının idari işlerinin otomasyonunun sağlanması gibi alanlarda kullanılmaktadır.

Yaşam süresinin uzaması dünya üzerindeki yaşlı nüfusun artmasına neden olmaktadır (Gökbunar ve diğerleri, 2016). Yaşlı nüfusun sağlık verilerinin sürekli olarak kontrol altında tutulabilmesi çeşitli IoT cihazlarıyla mümkün olmaktadır ve böylece insanların kontrol için sürekli bir sağlık kurumuna gitmesine gerek kalmamaktadır. Alzheimer gibi hastalıklara sahip kişiler ani unutkanlıklar yaşamakta ve bu durumda kaybolma vakaları yaşanmaktadır. IoT teknolojisiyle gerçek zamanlı lokasyon bilgilerinin sağlanması sayesinde bu tür hastalıklara sahip kişilerin yaşadıkları kaybolma vakaları azalmaktadır (Shaikh ve diğerleri, 2017).

Sağlık alanında IoT teknolojisiyle oluşan veriler büyük veriyi oluşturur ve bu verilerin analiz edilmesiyle sağlık alanında kullanılabilecek bilgilere ulaşılır. Bu alanda büyük veri de IoT teknolojisinin kullanım amaçlarının başında yer almaktadır (Dimitrov, 2016).

3. YÖNTEM VE ARAÇLAR

Bu tez çalışmasında sağlık kurumlarında IoT teknolojisinin kullanımı incelenmiştir. Alan yazın taraması sonucunda kullanılan teknolojiler karşılaştırılmıştır. Örnek vaka çalışması kapsamında sağlık kurumları için hasta dosyalarının anlık olarak nerede olduğunun tespit edilmesi ve geçmişe dönük dosya hareketlerinin incelenebilmesi için Hasta Dosyası Takip ve Bilgi Sistemi (HDTBS) tasarlanmıştır. IoT teknolojsi tabanlı ve MQTT üzerinden haberleşmeye olanak sağlayan sistemin geliştirilmesi esnasındaki yöntem ve kullanılan araçlar alt başlıklarda sunulmaktadır.

3.1. YÖNTEM

Alan yazın taraması yapıldıktan sonra, HDTBS (Hasta Dosyası Takip ve Bilgi Sistemi) geliştirilmeden önce sağlık kurumları özelinde IoT tabanlı projeler ve öneriler incelenmiştir. IoT teknolojisinde kullanılan haberleşme yöntemleri ve cihazlar karşılaştırılmış, HDTBS için uygun olan haberleşme yöntemi ve cihazlar belirlenmiştir. Cihazların birbirleriyle haberleşebilmeleri için cihazlara yüklenmesi gereken kod hazırlanmış ve test edilmiştir. Dosyaların takibinde Otomatik Tanımlama Sistemi olarak RFID teknolojisi seçilmiş olup, tez çalışması kapsamında kısıtlı kaynaklar nedeniyle düşük menzilden okuma yapabilen okuyucu cihazlar seçilmiştir. Gelecek çalışmalarda daha geniş mesafeden okuma yapabilen bir RFID okuyucu kullanılması menzil problemini ortadan kaldıracaktır. RFID etiketleri okunarak oluşan veri MQTT haberleşme protokolü üzerinden MQTT Arabulucu'ya gönderilmiş ve işlenerek veri tabanına kayıt işlemi yapılmıştır. MQTT Arabulucu olarak tek bir karttan oluşan mini bir bilgisayar kullanılmıştır. Bir oda için giriş ve çıkış yönü olmak üzere cihazlar hazırlanmış ve örnek hasta dosyaları cihazlara okutularak oda giriş çıkışlarının kayıtları incelenmiştir. Testler sonrası gerekli güncellemeler yapılmış ve sistemin çalışmasının kontrol edilebileceği bir web sitesi tasarlanarak, anlık giriş çıkış takibinin yapılabilmesi sağlanmıştır. Son olarak hazırlanan web sitesi arayüzünden iki hasta dosyasının oda giriş çıkışları testi yapılmıştır.

3.2. ARAÇLAR

Geliştirilen sistem üzerinde veri okuma cihazı olarak, RFID uygulamalarında sıklıkla kullanılan RC522 RFID okuyucu ve NodeMcu V3 Wifi Geliştirme Modülü kullanılmıştır. Cihazların birbirleriyle iletişim kurabilmeleri için gerekli kod Arduino IDE (Interface Development Environment - Arayüz Geliştirme Ortamı) üzerinde yazılmış ve cihazlara yüklenmiştir.

MQTT Arabulucu olarak açık kaynak kodlu ve ücretsiz olan Mosquitto uygulaması kullanılmıştır. Mosquitto, kredi kartı büyüklüğünde bir mini bilgisayar olan Raspberry Pi 3 Model B+ üzerine kurulmuş ve NodeMcu üzerinden gelen RFID okuma bilgilerini alarak kendisine abone olmuş uygulamalara aktarmıştır.

Veritabanı kullanımında açık kaynak kodlu ve kullanımı ücretsiz olan MySql veritabanı kullanılmıştır. Veritabanı üzerinde hasta bilgileri, oda bilgileri ve RFID okuyucudan gelen verilere göre dosyaların lokasyon bilgileri geçmişe dönük olmak üzere tutulmuştur.

Mosquitto Arabulucu'ya gelen mesajların veritabanına uygun şekilde kaydedilmesi ve giriş çıkış durumlarının tespit edilmesi için Python programlama diliyle bir uygulama yazılmıştır.

Veritabanına kaydedilmiş ve anlık olarak gelen giriş çıkış bilgilerinin takip edilebilmesi için JavaScript temelli Angular kütüphanesiyle bir internet uygulaması yapılmıştır. Veritabanı üzerinde bulunan kayıtların uygulama üzerinden çekilebilmesi için Python Flask ile Rest (Representational State Transfer - Temsilî Durum Aktarımı) API (Application Programming Interface - Uygulama Programlama Arayüzü) uygulaması yazılmıştır.

Geliştirilen sistemin beklenildiği gibi işlemesi ve dosya takibinin yapılabilmesi için genel hatlarıyla belirtilmiş olan teknolojiler ve yazılımlar alt başlıklar halinde aşağıda sunulmaktadır.

3.2.1. RC522 RFID Modülü

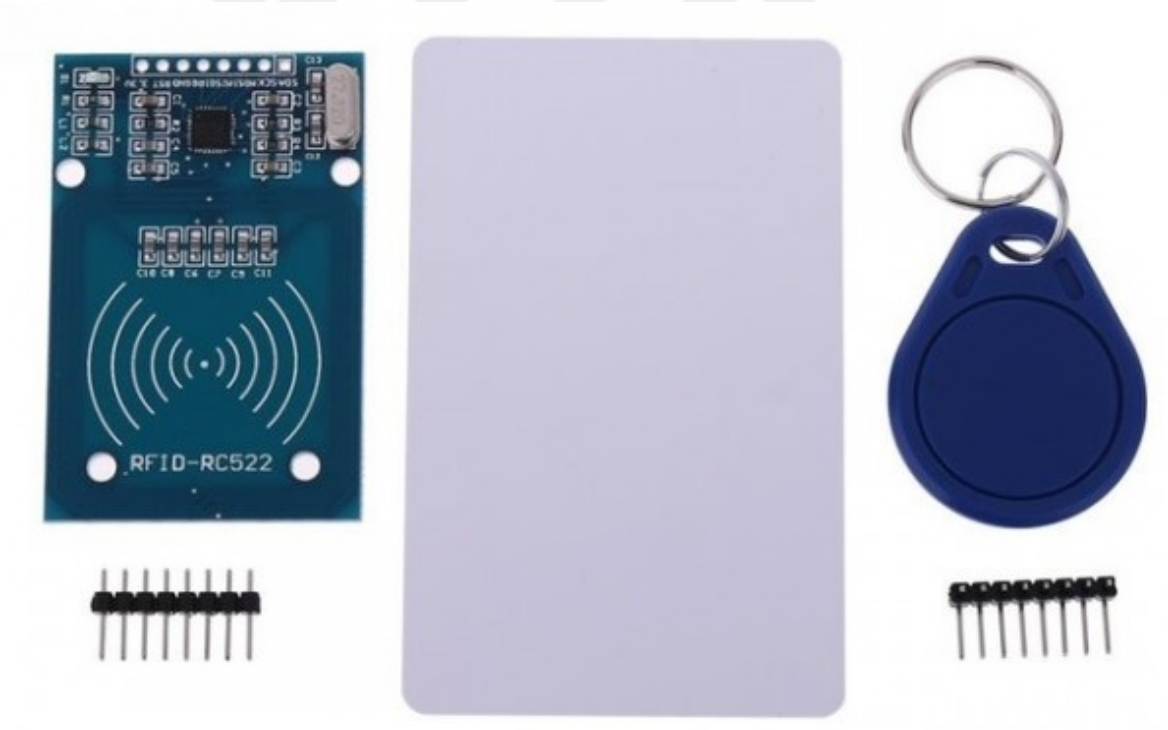
RC522 RFID modülü bir çok mikrodenetleyici platform üzerinde rahatlılıkla kullanılabilir olarak tasarlanmış olup, 424 kbit/s haberleşme hızına sahiptir.

RC522 RFID modülünün teknik özellikleri Tablo 3.1'de gösterilmiştir.

Tablo 3.1: RC522 rfid modülü teknik özellikleri (Sekyere-Asiedu, 2018).

Çalışma Frekansı	13,56 MHz
Çalışma Gerilimi	3,3V
Çalışma Akımı	13-26mA
Uyku Akımı	80 uA
Okuma menzili	Yaklaşık 3 cm
Haberleşme Protokolü	SPI
Haberleşme Maksimum Veri Transfer Hız	10Mbit/s
Haberleşme Kart Boyutları	40x60mm

Modülün genel görünümü Şekil 3.1'deki gibidir.



Şekil 3.1: RC522 rfid okuyucu ve etiketi.

3.2.2. NodeMcu Geliştirme Kartı

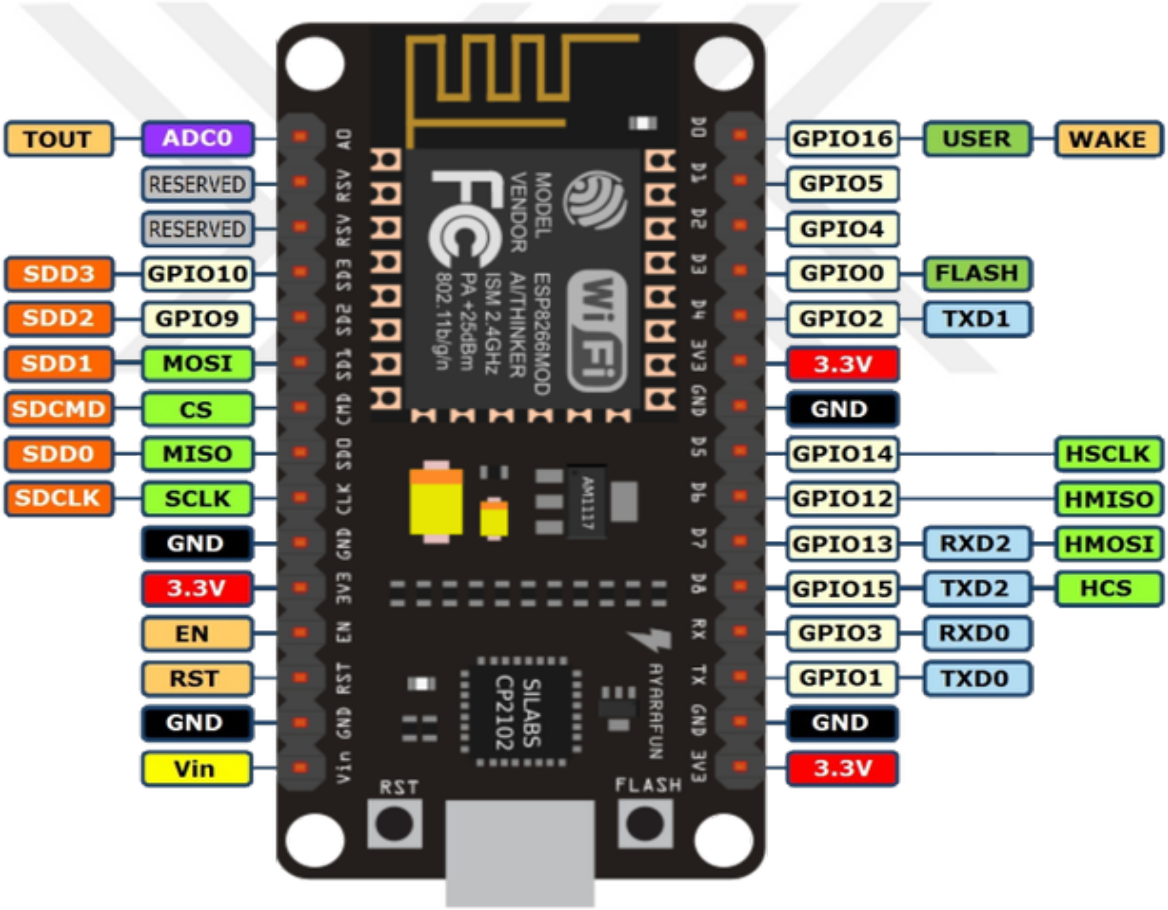
Geliştirilen HDTBS üzerinde bulunan NodeMcu; RFID ve Mosquitto Arabulucu arasındaki veri alışverişini sağlayan donanımdır. NodeMcu açık kaynak kodlu ve küçük boyutlu

bir kart olup, üzerinde bulunan ESP8266-12E Wi-Fi modülüyle kablosuz ağa kolayca bağlanabilmektedir. Üzerinde bulunan mikro USB portu üzerinden Arduino IDE aracılığıyla ya da LUA programlama diliyle programlanabilmektedir.

NodeMcu üzerinde 17 adet dijital giriş/çıkış pini ve ayrıca reset tuşu bulunmaktadır. MQTT protokolü üzerinden mesaj göndermeyi destekler ve az enerji tüketir.

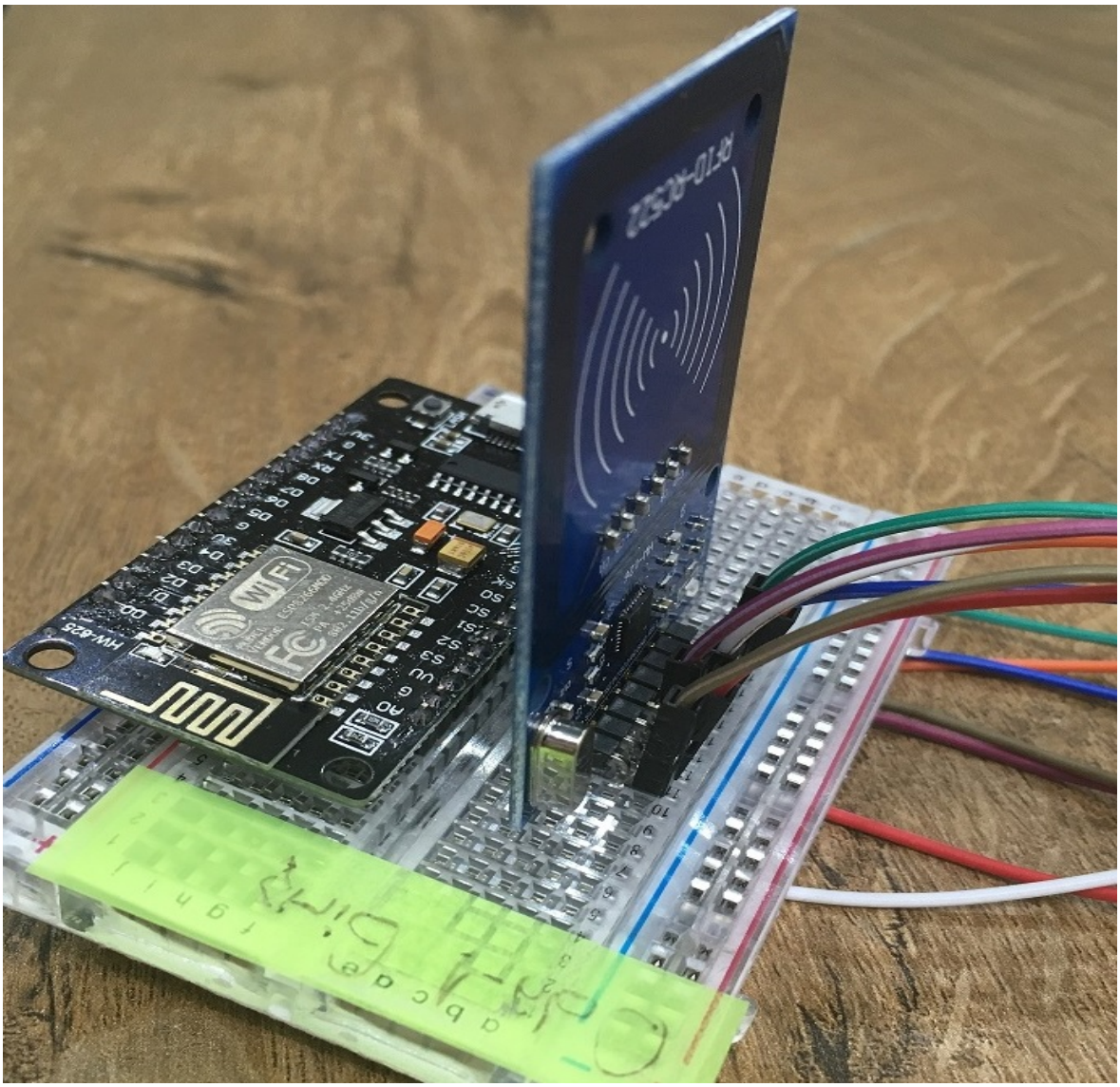
NodeMcu, Arduino gibi geliştirici kartlarlar beraber kullanılabildiği gibi üzerinde bulunan mevcut işlemcisiyle programlanarak da kullanılabilmektedir.

Kartın genel görünümü ve pin dizilimi Şekil 3.2'deki gibidir.



Şekil 3.2: NodeMcu geliştirme kartı ve pin dizilimi.

Tasarımı yapılmış cihaza ait görüntü Şekil 3.3'de görülmektedir.



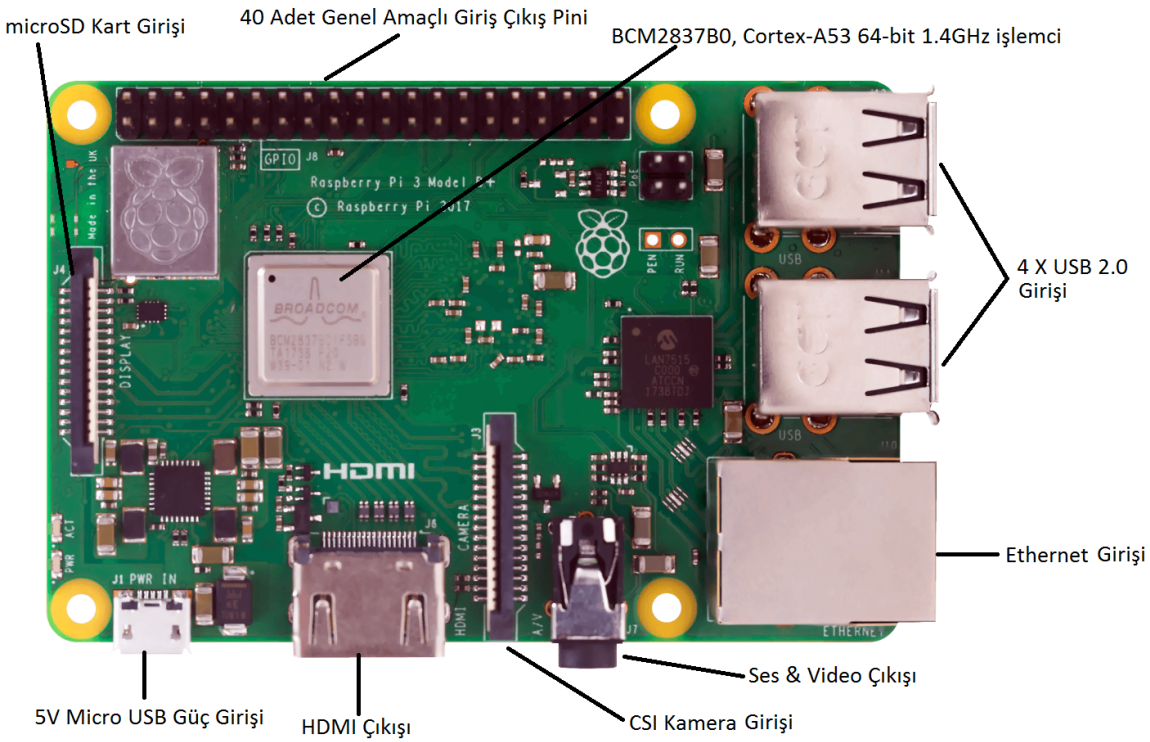
Şekil 3.3: Algılayıcı cihaz.

3.2.3. Raspberry Pi 3 Model B+

Raspberry Pi, İngilterde bulunan Raspberry Pi Vakfı tarafından desteklenen; kredi kartı büyüklüğünde ve tek karttan oluşan bir mini bilgisayardır. İşletim sistemi gerektiren gömülü sistem uygulamalarında; küçük boyutları ve düşük maliyeti sebebiyle sıklıkla tercih edilmektedir (Kaymak, 2016).

Raspberry Pi, donanımsal farklılıklar ihtiva eden çeşitli modellerle piyasada bulunmaktadır. Geliştirilen HDTBS' de Raspberry Pi 3 Model B+ kullanılmıştır.

Kartın genel görünümü Şekil 3.4'deki gibidir.



Şekil 3.4: Raspberry pi 3 model b+ görseli.

Klavye, fare ve ekran gibi çevre birimleri bağlanarak bir bilgisayar gibi de kullanılabilen Raspberry Pi üzerinde işletim sisteminin kurulabileceği dahili bir sabit disk bulunmamaktadır. Kart üzerinde bulunan microSD kart girişine hafıza kartı takılarak işletim sistemi kurulabilir. Raspberry Pi üzerinde kullanılmak üzere Linux tabanlı Raspian işletim sistemi hazırlanmış olup Raspberry Pi Vakfı' nın internet sitesinde sunulmaktadır.

HDTBS' de MQTT Arabulucu olarak kullanılan Mosquitto, Raspberry Pi 3 Model B+ üzerinde yapılandırılmıştır. Python programlama diliyle yazılan bir abone uygulama yine Raspberry Pi üzerinde çalışmakta ve gelen mesajları analiz ederek oda giriş çıkış hareketlerini veritabanına kayıt etmektedir.

HDTBS üzerinde kullanılan Raspberry Pi 3 Model B+ cihazının bazı teknik özellikleri Raspberry Pi Vakfı internet sitesi üzerinde aşağıdaki tabloda gösterildiği gibi belirtilmiştir.

Tablo 3.2: Raspberry pi 3 model b+ teknik özellikleri.

BCM2837B0, Cortex-A53 64-bit 1.4GHz işlemci
1 GB LPDDR2 SDRAM
2.4GHz ve 5GHz IEEE 802.11.b/g/ n/ac kablosuz LAN, Bluetooth 4.2, BLE
USB 2.0 üzerinden Gigabit Ethernet (maksimum çıkış 300 Mbps)
Genişletilmiş 40 pinli GPIO başlığı
4 USB 2.0 bağlantı noktası
DSI dokunmatik ekran portu
İşletim sisteminizi yüklemek ve veri depolamak için Micro SD port
5V / 2.5A DC güç girişi sisteminizi yüklemek ve veri depolamak için Micro SD port
Ethernet üzerinden Güç (PoE) desteği (ayrı PoE HAT genişletme kartı gerektirir)

HDTBS üzerinde kullanılan Raspberry Pi 3 Model B+ cihaza ait görüntü aşağıdaki resimde gösterilmiştir.

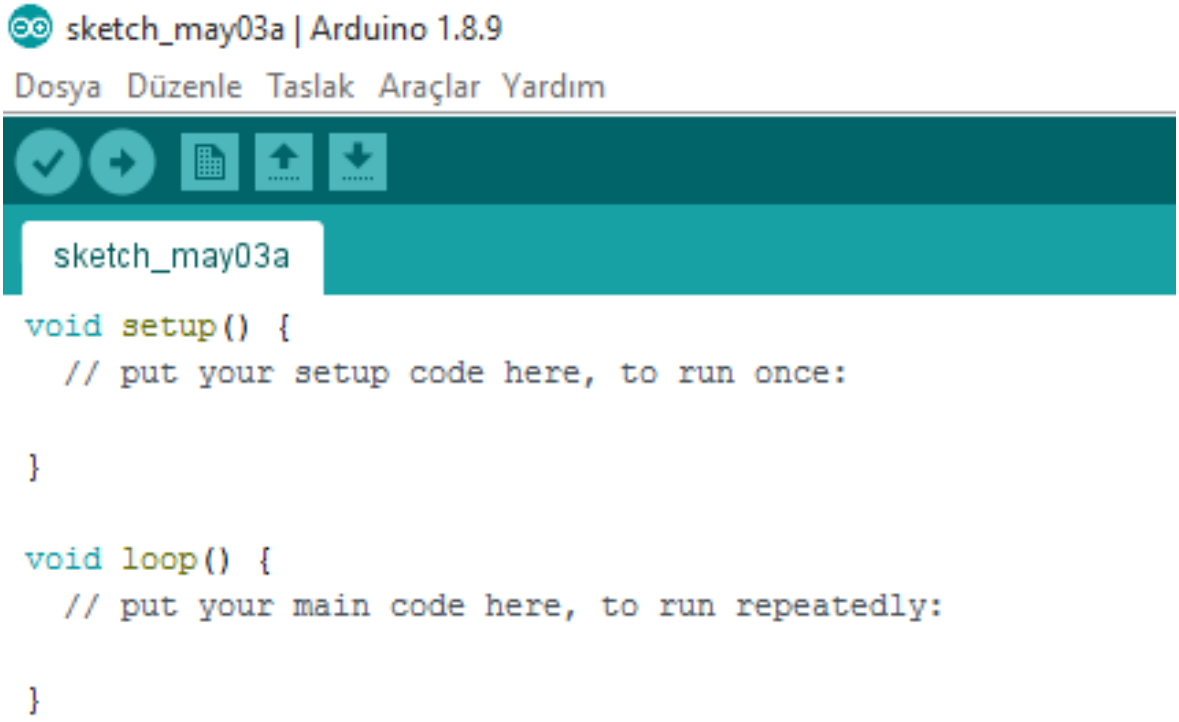
Şekil 3.5: Raspberry pi 3 model b+.

3.2.4. Arduino Arayüz Geliştirme Ortamı

Arduino Arayüz Geliştirme Ortamı, elektronik modüller için kod yazılmasında ve yazılan kodların modüle yüklenmesinde kullanılan bir uygulamadır. Açık kaynak kodlu olup Türkçe dil desteğine sahiptir. Windows, Mac Os X ve Linux ortamları için dağıtımları bulunmaktadır.

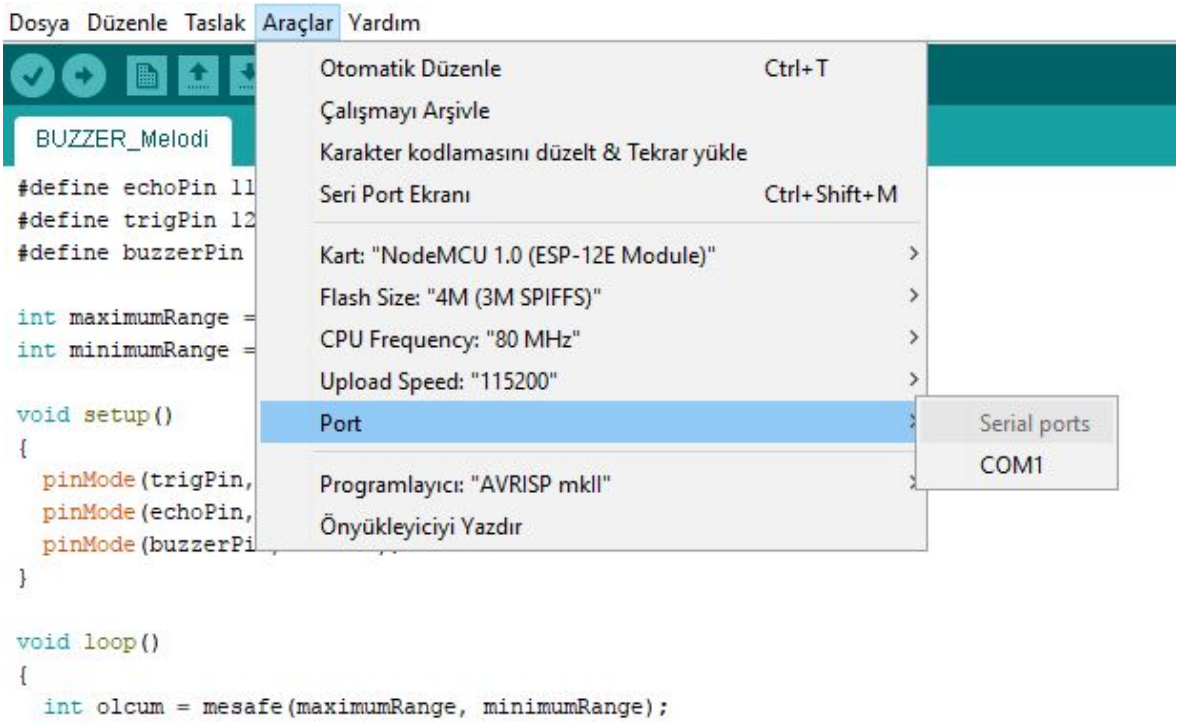
HDTBS üzerinde kullanılan RC522 RFID Modülü ve NodeMcu kartı arasında iletişimin kurulması ve oluşan verinin MQTT protokolüyle Raspberry Pi 3 Model B+ üzerinde bulunan Mosquitto Arabulucuya gönderilmesi için gerekli olan kodlama, Arduino Arayüz Geliştirme Ortamı üzerinden yapılmıştır. Çalışmanın yapıldığı an itibariyle en güncel sürüm olan Arduino 1.8.9 sürümü kullanılmıştır.

Arduino Arayüz Geliştirme Ortamı' nın genel görüntüsü Şekil 3.6'daki gibidir.



Şekil 3.6: Arduino arayüz geliştirme ortamı.

NodeMcu kartını Arduino uygulamasında programlayabilmek için Arduino Arayüz Geliştirme Ortamı üzerinde bulunan menüden Araçlar sekmesi açılır. Açılan pencere üzerinden "Additional Boards Manager Urls:" kutucuğuna *http* : *//arduino.esp*8266*.com/stable/package_esp*8266*com_index.json* adresi girilir ve Tamam butonuna tıklanır. Arduino Arayüz Geliştirme Ortamı (Şekil 3.7) kapatılıp yeniden açıldığında Araçlar sekmesi altında bulunan Kart menüsünde "Boards Manager" seçilir ve açılan pencere üzerinde bulunan kutucuğa ESP8266 yazılarak çıkan kütüphane kurulurak Arduino Arayüz Geliştirme Arayüzü tekrar başlatılır. Araçlar > Kart menüsünde NodeMCU 1.0 (ESP-12E Module) ve Port menüsünden de cihazın bilgisayara bağlı olduğu giriş seçilerek kodlama ve cihaza yazılımı yükleme işlemleri yapılabilir.



Şekil 3.7: NodeMCU bağlı arduino arayüz geliştirme ortamı.

3.2.5. Mosquitto Arabulucu

MQTT protokolünün kullanıldığı sistemlerde cihazlar doğrudan birbirleriyle iletişim kurmamaktadırlar. Arabulucu olarak adlandırılan bir uygulama gelen mesajları, bu mesajın konu başlığına abone olmuş uygulamalara yönlendirir. Arabulucu olarak kullanılabilen çeşitli uygulamalar mevcuttur. HDTBS'de açık kaynak kodlu ve geniş bir kullanıcı kitlesine sahip Mosquitto Arabulucu tercih edilmiştir.

Raspian işletim sistemine sahip bir Raspberry Pi cihazına Mosquitto Arabulucu uygulamasını kurmak için terminal penceresi açılır ve sırasıyla aşağıda listelenen komutlar yazılır.

- sudo apt update
- sudo apt install -y mosquitto mosquitto-clients
- sudo systemctl enable mosquitto.service

Uygulamanın başarıyla kurulduğunu test etmek için terminal penceresinde "mosquitto -v"

yazılır. Sonuç olarak Mosquitto versiyon bilgisiyle beraber hangi portun dinlenildiği ve hangi ayar dosyasının kullanıldığı bilgisi gelmelidir.

3.2.6. MySQL

Tez çalışması kapsamında geliştirilen HDTBS uygulamasında veritabanı olarak MySQL tercih edilmiştir.

MySQL veri tabanının güçlü yanları;

- GPL (General Public License - Genel Kamu Lisansı) altında kullanımı ücretsiz olup ticari kullanımlarda ise diğer ticari ürünlere kıyasla daha makul seviyede ücretlendirilmektedir.
- GPL lisanslamaya sahip olduğundan özelleştirilebilmektedir.
- Yüksek performanslıdır.
- Öğrenim ve yapılandırma kolaylığı sağlar.
- Çoğu işletim sistemi üzerinde çalışabilmektedir.
- Büyük bir kullanıcı topluluğuna sahip olduğundan destek hizmeti geniştir.
- Güvenlidir. Belirli bir kullanıcıya ya da gruba yönelik yetkilendirme yapılabilmesini sağlayan esnek bir yetkilendirme sistemine sahiptir.

olarak belirtilmektedir (Valade, 2010).

3.2.7. Python Programlama Dili

Guido Van Rossum tarafından 1991 yılında geliştirilen Python, çok güçlü ve yüksek seviyeli, dinamik nesne yönelimli programlama dilidir (Harwani, 2011). Python programlama dili, genel bir kullanım alanına sahiptir. İnternet tabanlı uygulama geliştirmeden veri madenciliğine, yapay zekadan IoT uygulamalarına kadar gibi birçok alanda geliştirmeler yapılabilmektedir.

Raspberry Pi üzerinde Python programlama diliyle geliştirmeler yapmak oldukça kolaydır. Raspian işletim sistemiyle beraber gelen Python geliştirici araçlarıyla uygulama yazmaya

başlanabilir ve cihaz üzerine bağlanmış olan sensörlerden veri okunarak istenilen işlemlerin yapılması sağlanabilmektedir. Raspberry Pi üzerinde sağlanan bu kolay kullanılabilirlik IoT projelerinde Python dilinin kullanılmasına ön ayak olmaktadır.

HDTBS'de arabulucudan gelen mesajların işlendiği uygulama Python programlama diliyle yazılmıştır. Gelen mesajlara göre dosyanın hangi odaya giriş yaptığı ya da çıkış yaptığı bilgisi veri tabanına yazılarak kayıt altına alınmaktadır.

Python programlama dilinin olumlu yönleri;

- Açık kaynak kodlu olması,
- Kolay öğrenilebilirlik ve kullanım için tasarlanmış olması,
- Yorumlanan bir dil olmasından dolayı yorumlayıcı üzerinde etkileşimli olarak çalışabilmesi,
- Nesne yönelimli programlamayı desteklemesi,
- Geniş bir kullanıcı kitlesine sahip olması,
- Kütüphanelerinin ve araçlarının geniş olması,
- Öğrenme kaynaklarının genişliği,
- Standart kütüphanelerinin genişliği,

olarak belirtilmektedir (Malkoç, 2012).

3.2.8. Angular

Angular, Google tarafından desteklenen, Javascript tabanlı açık kaynak kodlu ve MVC deseni üzerine yapılandırılmış bir kütüphanedir.

Angular' ın resmi web sayfası olan $https://angular.io/$ sitesine göre, kütüphanenin sunmuş olduğu avantajlar;

- Web, masaüstü, mobil gibi birçok farklı platform için uygulama yazılabilir.

- Yüksek performanslıdır.
- Çok çeşitli geliştirici araçları bulunmaktadır.
- Geniş bir topluluk tarafından kullanılmakta ve desteklenmektedir.
- Test odaklı kod yazılmasına olanak sağlamaktadır.

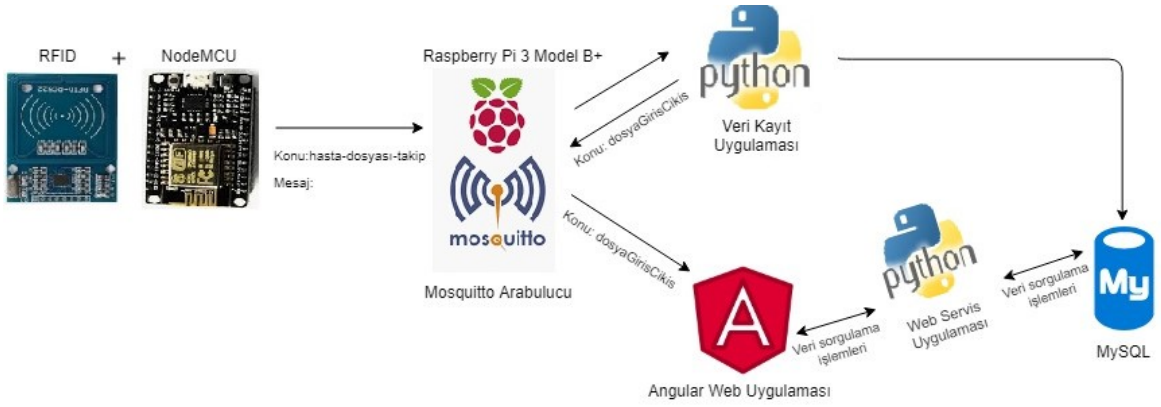
olarak belirtilmektedir. [3]

[3] Angular Kütüphanesi resmi internet adresi $https://angular.io/$ internet sitesinden alınmıştır.

4. HASTA DOSYASI TAKİP VE BİLGİ SİSTEMİ (HDTBS) UYGULAMASI

Son yıllarda sağlık kurumlarında dijital hastanelere dönüşüm süreci hızlanmış olsa dahi, hastaya ait bir takım dökümanların dijital ortam dışında da tutulması tercih edilmektedir. Islak imzanın halen daha önemli olduğu süreçler için bu durum daha çok gözlemlenmektedir. Hastalara ait bu dosyaların korunması ilgili sağlık kurumunun sorumluluğundadır. Hastaya ait bir dosyanın kaybolması hem sağlık kurumu için hem hasta için olumsuz sonuçlar doğurmaktadır. Bu olumsuzlukları azaltmak için hasta dosyası takip sistemi oluşturmak tasarlanan sistemin konseptini oluşturmaktadır.

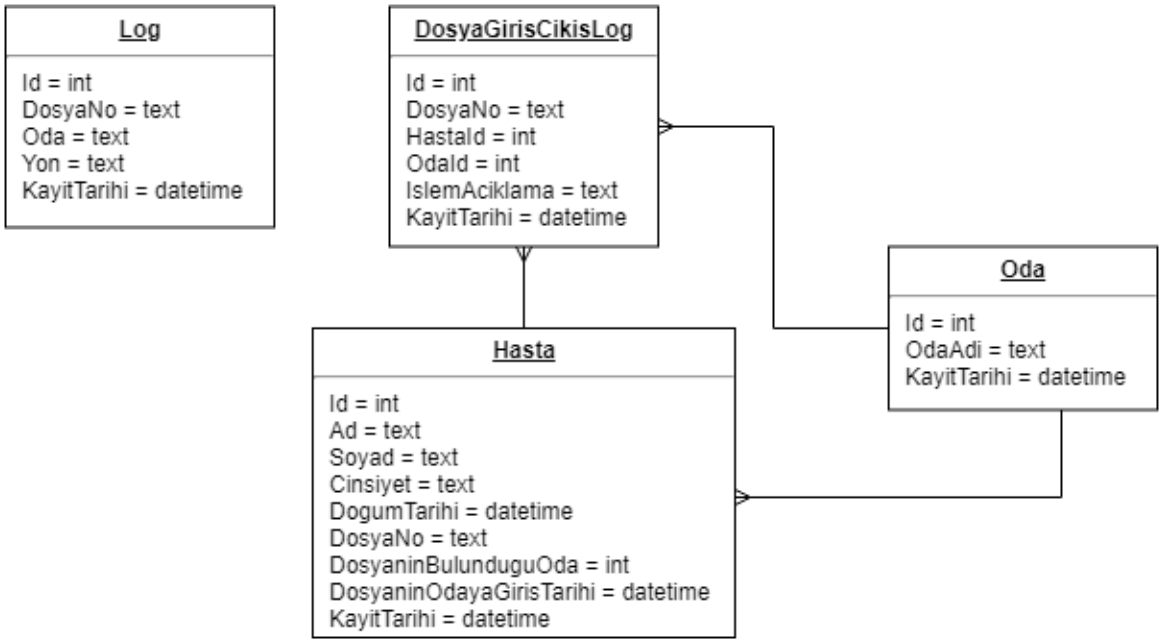
Bu tez çalışması kapsamında, sağlık kurumlarının hasta dosyalarının takibinde kullanabileceği HDTBS adlı IoT teknolojisi tabanlı bir sistem tasarlanmıştır. Sistem, RFID okuyucu ve Wi-Fi modülüne sahip bir kart üzerinden okunan dosya verisinin MQTT protokolüyle uzak sunucuya gönderilmesini ve giriş/çıkış bilgisinin veritabanına kaydedilmesini sağlamaktadır. Giriş çıkış verilerinin anlık ve geriye dönük olarak takip edilebilmesi için bir web arayüz geliştirilmiş ve test işlemleri bu arayüz üzerinden yapılmıştır. Sistemin genel mimarisi Şekil 4.1'de gösterilmiştir.



Şekil 4.1: HDTBS mimarisi.

4.1. VERİTABANI

HDTBS verilerinin kayıt ortamı için MySQL veritabanı seçilmiştir. HastaDosyasiTakip adında bir veritabanı oluşturulmuştur. Şekil 4.2'de veritabanı E/R diyagramı görülmektedir.



Şekil 4.2: Veritabanı e/r diyagramı.

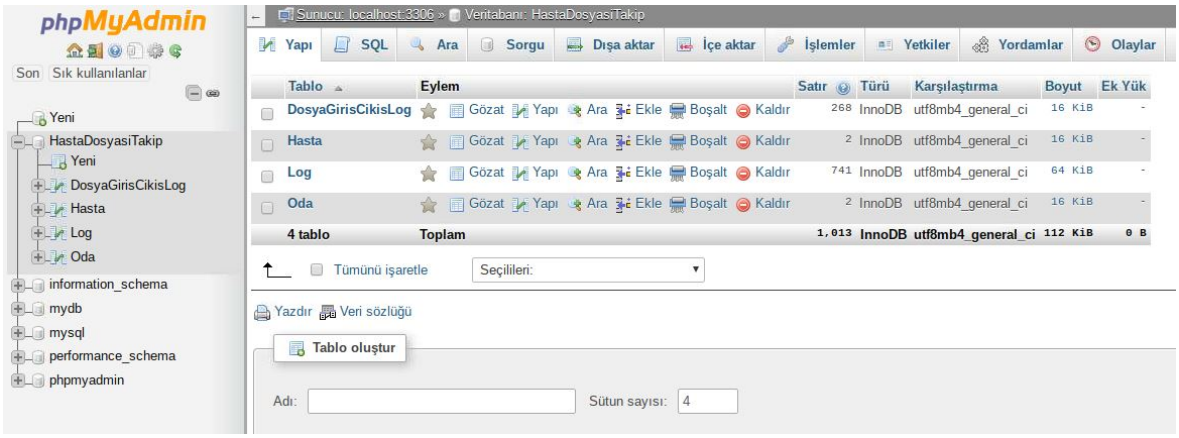
DosyaGirisCikisLog, dosyaların oda giriş çıkışlarının loglandığı tablodur. IslemAciklama alanı hareketin tipini belirtmektedir. Hareket tipi giriş ya da çıkış olarak belirlenmiştir.

Hasta, hasta ve hastaya ait dosya bilgilerinin tutulduğu tablodur. Hastaya ait dosyanın bulunduğu lokasyon bilgisi bu tablo üzerinde tutulmaktadır. Eğer dosyanın lokasyonunda bir değişme olursa hasta kaydı güncellenmektedir.

Oda, sağlık kurumu içerisinde bulunan odalar için tanım tablosu olarak kullanılmaktadır.

Log, MQTT arabulucudan gelen tüm mesajların kayıt altına alındığı tablodur.

Oluşturulan tabloların veritabanı yönetim sistemi üzerinden görüntüsü Şekil 4.3'te gösterilmiştir.



Şekil 4.3: Veritabanı tablolar.

4.2. ALGILAYICI CİHAZ TASARIMI

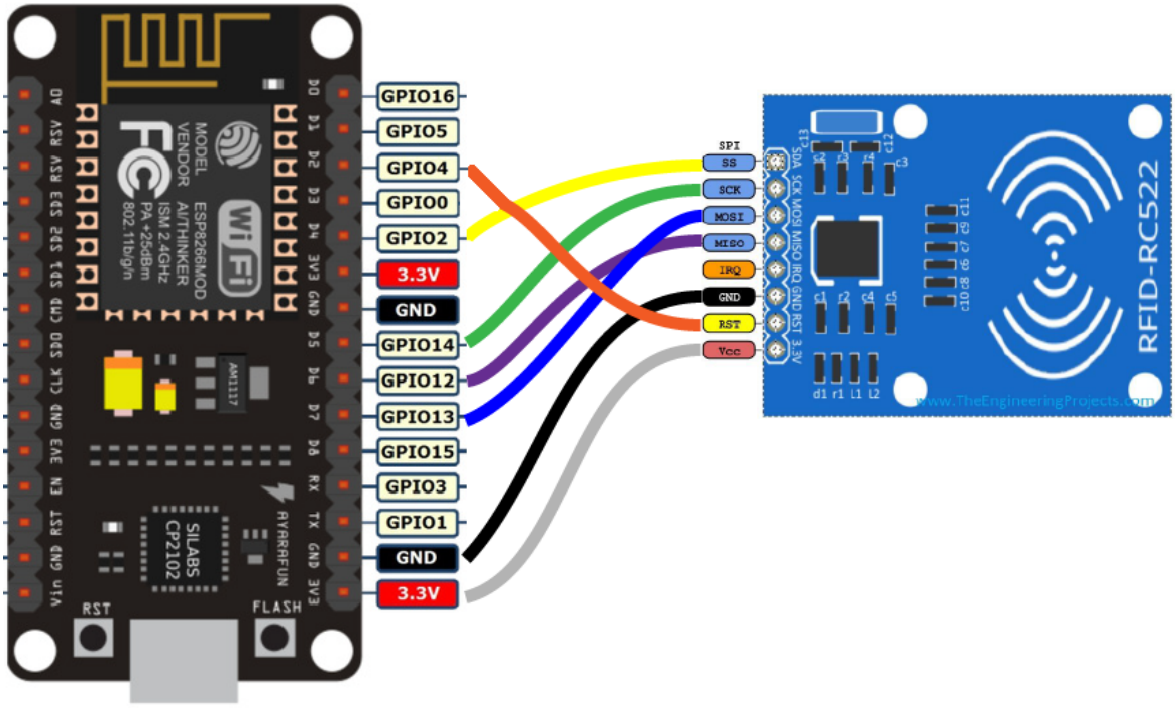
Sistemin veri toplama kısmı RF522 RFID cihaz ve NodeMCU karttan oluşmaktadır. NodeMcu kartı, RF522'den gelen etiket bilgilerini ve NodeMcu kartının bulunduğu lokasyon bilgisini MQTT protokolüyle JSON formatında arabulucunun kurulmuş olduğu sunucu cihaza göndermektedir. NodeMcu cihaz dahili Wi-Fi modüle sahip olduğundan ayrıca bir Wi-Fi modül kullanmaya gerek olmamaktadır.

Arduino Arayüz Geliştirme Ortamı aracılığıyla NodeMcu kart üzerine RFID modülüyle ve MQTT Arabulucuyla iletişim kurulabilmesi için gerekli kodlar yazılmıştır. Kodlar yazılırken yardımcı kütüphanelere de ihtiyaç duyulmuştur. Kullanılan kütüphaneler aşağıda listelenmiştir.

- SPI.h : Cihazlar arası seri haberleşme yapılması için kullanılmıştır.
- MFRC522.h : RC522 RFID okuyucu modül ile iletişim kurulması ve cihazdan gelen etiket bilgilerinin okunması için kullanılmıştır.
- ESP8266WiFi : NodeMcu üzerinde bulunan dahili ESP8266 Wi-Fi modülü üzerinde işlemler yapılması için kullanılmıştır.
- PubSubClient.h : MQTT protokolüne bağlı işlemler yapmak için kullanılmıştır.
- ArduinoJson.h : MQTT üzerinden JSON formatında veri göndermek için kullanılmıştır.

Yazılan kodlar Arduino Arayüz Geliştirme Ortamı üzerinden NodeMcu kartına yüklenmiştir. Çalıştırılabilir kodlar Ek-1'de verilmiştir.

RF522 modül, NodeMcu karta üzerinde bulunan pinler aracılığıyla bağlanmaktadır. Cihazlar arasındaki pin bağlantıları Şekil 4.4'de belirtildiği gibi yapılmıştır.



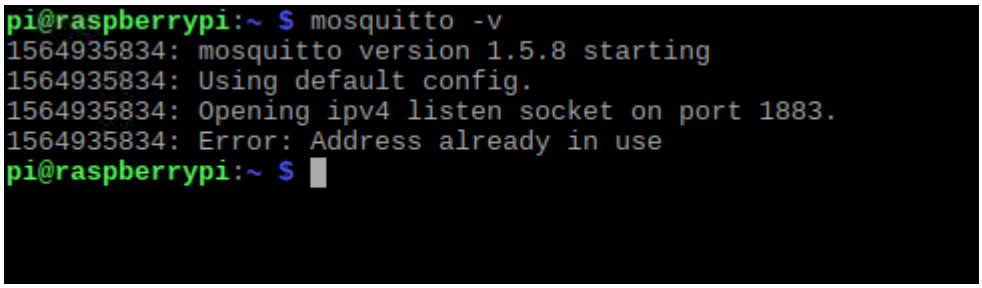
Şekil 4.4: NodeMcu ve rc522 rfid pin bağlantısı.

4.3. MQTT ARABULUCU TASARIMI

Algılayıcı cihaz üzerinden alınan verilerin merkezi bir sunucuda toplanması ve veritabanına kayıt edilmesi için hafif bir protokol olan MQTT protokolü kullanılması tercih edilmiştir. MQTT protokolünde gelen mesajların yönlendirilmesi için bir arabulucuya ihtiyaç vardır. HDTBS için açık kaynak kodlu ve geniş bir kullanıcı kitlesine sahip Mosquitto Arabulucu tercih edilmiştir.

Raspberry Pi 3 Model B+ tek kart bilgisiyar üzerine Mosquitto Arabulucu 1.5.8 versiyonu kurulmuştur (Şekil 4.5).

Algılayıcı cihazların arabulucunun bulunduğu cihaza erişmesinde problem yaşanmaması için Raspberry Pi üzerinde Mosquitto Arabulucunun kullanmış olduğu portlara erişim izinleri



Şekil 4.5: Mosquitto versiyon bilgisi.

verilmiş ve statik IP ataması yapılmıştır.

Algılayıcı cihazlar, "hasta-dosyasi-takip" adında bir konu başlığına, RFID okuyucu modül aracılığıyla okunan etiket bilgileriyle beraber NodeMcu üzerinde bulunan lokasyon bilgisini de göndermektedirler. Mosquitto Arabulucu gelen mesajı konu başlığına abone olmuş uygulamalara yönlendirmektedir.

JSON formatında gönderilen mesaj içerisinde dosya no, oda ve yön bilgisi bulunmaktadır (Şekil 4.6). Dosya no, hangi dosyanın RFID okuyucu tarafından okunduğunu belirtmektedir. Oda bilgisi NodeMcu cihazın hangi lokasyonda bulunduğunu ve yön bilgisi ise odanın giriş ya da çıkış yönünde mi olduğunu belirtmektedir.

Şekil 4.6: MQTT üzerinden gönderilen mesaj örneği.

4.4. VERİ KAYIT UYGULAMASI

Mosquitto Arabulucu "hasta-dosyasi-takip" konu başlığına gelen mesajları, bu konuya abone olmuş olan Veri Kayıt Uygulamasına göndermektedir. Veri Kayıt Uygulaması, gelen mesajları loglar ve mesaj içeriğine göre dosyanın odaya giriş mi çıkış mı yaptığına karar vererek veritabanında dosya lokasyonunu güncellemektedir.

Veri kayıt uygulaması, Raspberry Pi tek kart bilgisayar üzerinde çalışmaktadır. Uygulama, Python diliyle yazılmış olup uygulamaya ek fonksiyonlar katmak için yardımcı kütüphaneler kullanılmıştır. Kullanılan kütüphaneler aşağıda listelenmiştir.

- paho.mqtt.client : MQTT protokolü üzerinden haberleşme yapabilmek için kullanılmıştır.
- MySQLdb : MySQL veritabanına bağlanmak ve işlemler yapabilmek için kullanılmışır.
- json : Arabulucudan gelen JSON formatında ki mesajları okumak ve JSON formatında mesajlar yollamak için kullanılmıştır.

Mesaj içeriğinde gelen bilgilere göre dosyanın bir odaya giriş ya da çıkış yaptığı belirlenmişse, "dosyaGirisCikis" adlı MQTT konu başlığına uygulama tarafından bilgilendirme mesajı gönderilmektedir (Şekil 4.7). Bu konu başlığına abone olmuş uygulamalar bu bilgiyi anlık olarak alabilmektedirler.

Şekil 4.7: Veri kayıt uygulamasından gönderilen mqtt mesaj örneği.

Arabulucudan gelen mesajda belirtilen dosya no eğer bir hastayla ilişkili değilse veritabanına sadece log kaydı atılır. Aynı şekilde eğer mesajda gelen oda bilgisi veritabanında tanımlı bir oda değilse sadece log kaydı atılır, diğer işlemlere devam edilmez. Sistemin asıl amacı hastalara ait dosyaların takibinin yapılması olduğundan dosyaların hastalarla ilişkilendirilmesi gerekmektedir.

Uygulamaya ait çalıştırılabilir kodlar Ek-1'de verilmiştir.

4.5. WEB SERVİS UYGULAMASI

HastaDosyasiTakip veritabanından sorgulama yapılabilmesi için Pyhton programlama diliyle bir web servis uygulaması yazılmıştır. Servis aracılığıyla dosya hareketleri geçmişe dönük olarak sorgulanabilmektedir.

Servis üzerinde ayrıca hasta ve oda için kayıt ekleme ve güncelleme işlemlerinin yapılabileceği metotlar da bulunmaktadır.

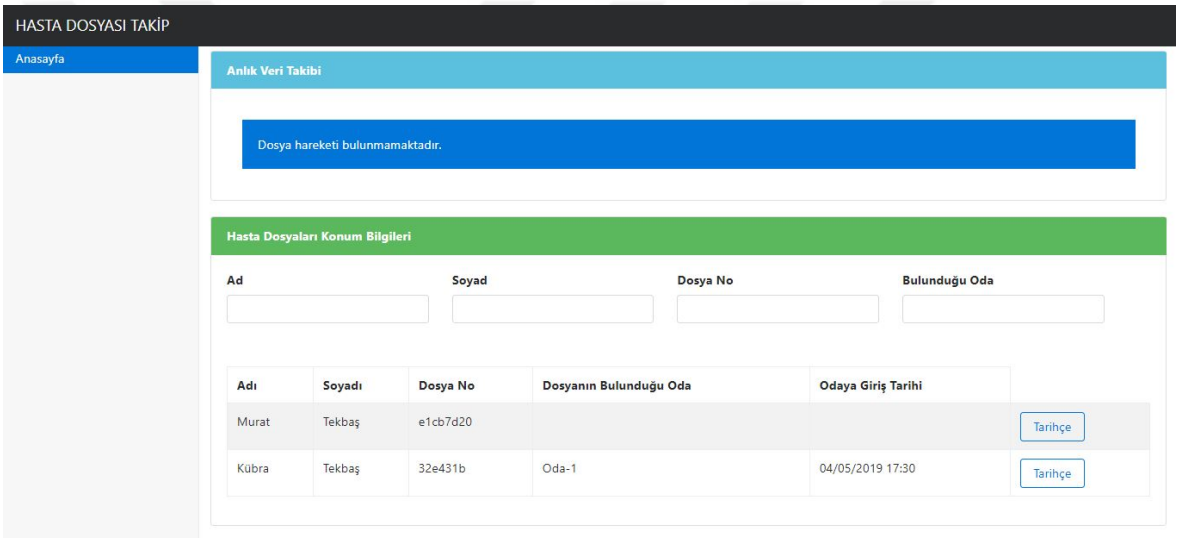
Servis aracılığıyla; web, mobil ve masaüstü için yazılacak tüm uygulamalar dosya giriş/çıkış sorgulaması, hasta ve oda kayıtlarına ekleme ya da güncelleme işlemleri yapabilmektedirler.

4.6. ANGULAR WEB UYGULAMASI

Hasta dosyası hareketlerinin anlık takip edilebilmesi ve dosya hareketlerinin geçmişe yönelik incelenmesi adına Angular kütüphanesiyle bir web arayüzü hazırlanmıştır (Şekil 4.8).

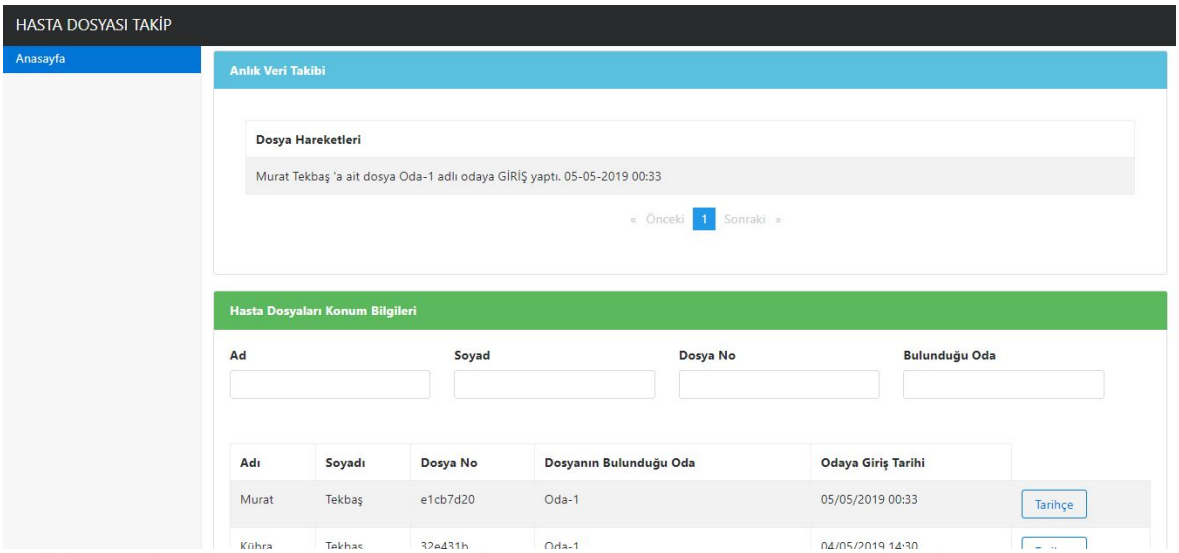
Uygulama, MQTT protokolü üzerinden dosyaGirisCikis konusuna abone yapılarak dosya giriş çıkışlarının arayüz üzerinden anlık olarak görüntülenmesi sağlanmıştır.

Uygulama, Web Servis Uygulaması aracılığıyla MySQL veritabanı üzerinden sorgulama yapabilmektedir. Dosya hareketlerinin geçmişe yönelik incelenmesi servisten alınan verilerle yapılmaktadır.



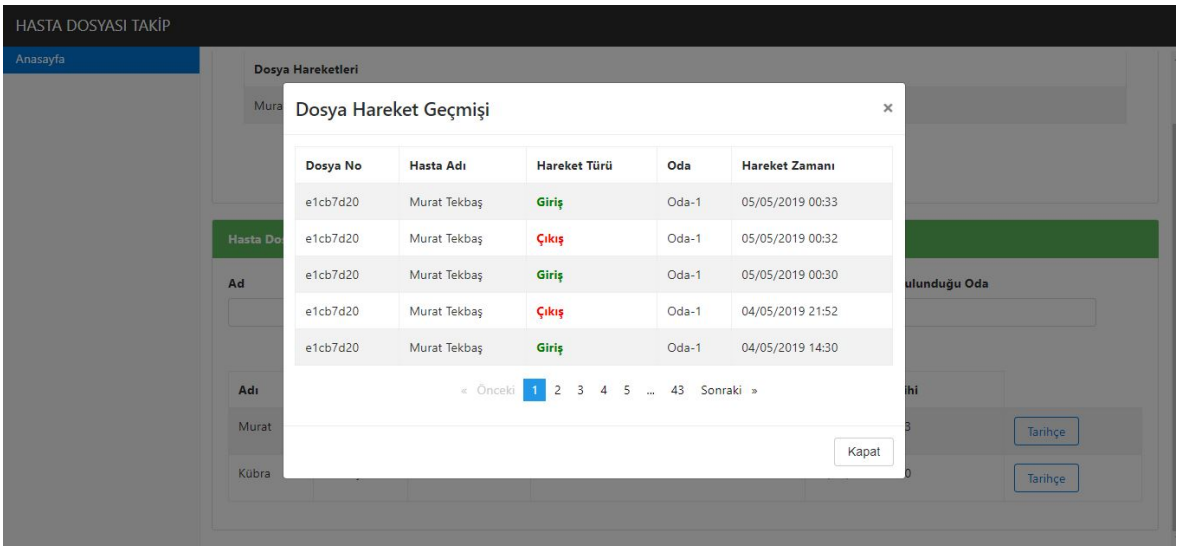
Şekil 4.8: Angular web uygulaması açılış ekranı.

Uygulama ilk açıldığında üst bölümde Anlık Veri Takibi adında anlık verilerin takip edildiği bir bölüm bulunmaktadır. Bu bölüm dosyaGirisCikis konusuna gelen mesajları dinlemektedir. Mesaj geldiğinde liste üzerinde bulunan bilgiler gelen mesaja göre güncellenmektedir (Şekil 4.9).



Şekil 4.9: Angular web uygulaması mqtt mesajı.

Anlık Veri Takibi bölümü altında ise Hasta Dosyaları Konum Bilgileri listesi bulunmaktadır. Liste üzerinde ayrıca filtreleme işlemi de yapılabilmektedir. Her kişi kaydına ait bir Tarihçe butonu bulunmaktadır. Bu buton servisten dosya hareketlerini sorgulayıp listelemektedir (Şekil 4.10).



Şekil 4.10: Angular web uygulaması dosya hareket geçmişi.

5. TARTIŞMA VE SONUÇ

Sağlık kurumlarında IoT teknolojilerine dayanan sistemler kurulması, hastalara daha iyi bir sağlık hizmeti sunmak için önem arz etmektedir. Bu kurumlar için son yıllarda öne çıkan bir kavram olan Dijital Hastane kavramı kapsamında kağıt kullanımının ortadan kaldırılması ve tüm süreçlerin dijital kanallardan yürütülmesi önerilmektedir. Türkiye'de dijital hastane süreçlerinin uygulandığı hastaneler olmakla beraber ülkemizde bulunan çoğu sağlık kurumunda süreçlerin tamamiyle dijital kanallardan yürümesi ve kağıtsız ortama geçilmesi için zaman gerekmektedir. Bu aşamada hastalara ait bilgilerin bulunduğu dosyaların saklanması ve korunması sağlık kurumlarının sorumluluğundadır. Hastaya ait sağlık bilgilerinin bulunduğu dosyaların kaybolması sağlık kurumları için ağır hizmet kusuru sayılmaktadır. Böyle bir olayın yaşanması hem hasta hem de sağlık kurumları için olumsuz sonuçların oluşmasına sebebiyet vermektedir.

Düşük güç tüketen entegre devreler ve kablosuz teknoloji alanındaki son gelişmelerle beraber birbirleriyle iletişim kurabilen cihazların tasarlanması mümkün hale gelmiştir. Cihazların birbirleriyle haberleşebilmeleri IoT teknolojisi alanına yapılan yatırımların artmasını ve çalışmaların çeşitlenmesini sağlamıştır. Cihazların haberleşmelerinde nesnelerin ve canlıların otomatik tanımlanması uygulamanın amacına bağlı olarak önem arz etmektedir. Otomatik tanımlama sistemleri için ulaşılabilirlik arttıkça IoT alanında bu sistemlerin de entegre biçimde kullanılması mümkün hale gelmiş ve otomatik tanımlama sistemlerini içeren IoT tabanlı çalışmaların sayısı artmıştır.

IoT teknolojisinin sağlık kurumlarında kullanılması iş süreçlerinin hızlanmasını ve insan kaynaklı hataların azaltılmasını sağlamaktadır. Sağlık hizmetleri zaman ve kaynak yönetiminin efektif kullanılmasını gerektiren alanlardan biridir. IoT teknolojisiyle sağlık kurumlarında insanlar tarafından yönetilmesi gereken çoğu iş süreci uygun cihazlar ve bu cihazların birbirleriyle haberleşmesi sonucu alınacak kararlarla insan müdahalesine gerek kalmadan yönetilebilmektedir. Böylece uygulanması gereken işlemler çok daha hızlı bir şekilde uygulanmakta ve personel kaynakları gerekli olan diğer bölümlere yönlendirilebilmektedir. Günümüzde yapılan çalışmalarla hasta yataklarının

hasta hareketlerine uygun olarak pozisyon alması, bina içinde ısı kaynaklarının kontrol edilerek ideal oda koşullarının sağlanması, hastaya bağlı serumların izlenerek serum bittiğinde hemşireye uyarı gönderilmesi, hastanın değerleri kontrol edilerek gerektiğinde otomatik ilaç pompalarıyla hastaya ilaç enjekte edilmesi gibi birçok işlemin otomasyona geçmesi sağlanmaktadır. Yapılan bu çalışmalar sağlık kurumlarının dijital dönüşüm süreçlerinin hızlanmasını sağlamakta, zaman ve kaynak yönetiminin etkili bir biçimde gerçekleştirilmesinin önünü açmaktadır.

Bu tez çalışması kapsamında, Otomatik Tanımlama Sistemleri ve IoT teknolojilerinden faydalanılarak bir hasta dosyası takip sistemi önerilmiştir. Geliştirilen sistem için oda girişlerine, giriş ve çıkış yönlü olarak algılayıcı cihazlar konulmuştur. Algılayıcı cihazlar, hasta dosyaları üzerine yerleştirilmiş olan RFID etiketlerini okuyarak MQTT haberleşme protokolüyle okunan dosya bilgisini ve hangi lokasyondan bilginin okunduğunu uzak sunucuya göndermektedir. Uzak sunucu üzerinde bulunan Mosquitto Arabulucu gelen mesajı ilgili uygulamaya aktarmaktadır. Uygulama, mesaj içeriğini ve önceden gelmiş olan mesaj loglarını inceleyerek hareketin giriş mi ya da çıkış mı olduğunu tespit etmektedir. Giriş ya da çıkış olduğuna karar verilen hareket için, veritabanında güncelleme ve loglama işlemleri gerçekleştirilmektedir. Uygulama üzerinden ayrıca MQTT haberleşme protokolüyle "dosyaGirisCikis" konu başlığına hangi dosyanın hangi oda için ne işlemi yaptığına dair bilgilendirme mesajı gönderilmektedir. Konu başlığına abone olmuş olan tüm uygulamalar bu bilgiyi anlık olarak takip edebilmektedirler.

Geliştirilen sistem için Angular kütüphanesiyle bir arayüz oluşturulmuştur. Arayüz üzerinden dosya giriş çıkışları anlık olarak takip edilebilmiş ve dosya hareketlerinin geçmişe yönelik incelenmesi sağlanmıştır. Arayüz, "dosyaGirisCikis" konu başlığına abone yapılmış ve bu sayede anlık giriş çıkış bilgileri sistem üzerinde gösterilmiştir. Yapılan testlerde, dosya giriş çıkış bilgilerinin başarıyla sistem üzerinden gösterildiği gözlemlenmiştir. Geçmişe dönük dosya hareketlerinin listelenmesi için Python programlama diliyle yazılmış Web Servis Uygulamasından veri sorgulaması yapılmıştır. Gözlemlenen sorunlar tespit edilmiş, güncellemeler yapılarak tekrar test edilmiş ve sonuçların başarıyla listelendiği gözlemlenmiştir. Böylece dosyaların herhangi bir zaman aralığında nerede oldukları tespit edilebilmiştir.

Sistem sayesinde insan müdahelesine gerek kalmadan hareket halinde olan dosyaların

lokasyon bilgileri takip edilmiş ve uzak sunucu üzerinde bulunan veritabanında kayıt altına alınmıştır. Tüm hareket geçmişi veritabanında kayıt altında tutulduğundan geçmişe dönük dosya hareketleri incelenebilmiştir.

Geliştirilen sistem üzerinde kullanılan RC522 RFID Okuyucu cihazın okuma menzili düşüktür. Bu nedenle dosyaların yakın mesafeden okutulması gerekmektedir. Daha uzak mesafeden okuma yapabilen RFID Okuyucu cihazlar piyasada mevcut olup şu an için tercih edilen cihaza göre daha maliyetlidir. Okuma menzili daha yüksek olan bir cihaz sisteme entegre edilerek, kişinin dosyayı algılayıcı cihaza yaklaştırmasına gerek kalmadan otomatik olarak okuma yapabilmesi sağlanabilir.

RFID etiketi olarak kart tipinde olan etiketler kullanılmıştır. Hasta dosyasının bulunduğu klasöre bu kartların ekleneceği varsayılmıştır. Daha uzak mesafeden okuma yapabilen RFID okuyucular sisteme entegre edildiğinde yapışkan tipinde olan daha ufak etiketlerin kullanılması sağlanabilir.

Kablosuz haberleşmede kullanılan MQTT protokolünün güvenliğinin sağlanması adına, konu başlıklarına mesaj gönderiminin kullanıcı adı ve şifreyle yapılması sağlanabilir. Cihazların bulunduğu ağ ortamı, dış ağ ortamından izole edilerek özel bir ağ oluşturulabilir. Diğer bir güvenlik önlemi de cihazlar arası iletişimin SSL (Secure Sockets Layer-Güvenli Giriş Katmanı) üzerinden yapılması olarak belirtilebilir.

Dosyaların bina dışına çıkışı yapıldığında belirlenecek kişi ya da birimlere bilgi verecek bir geliştirme yapılabilir. Böylece dosyaların izinsiz bina dışına çıkarılmasının önüne geçilebilir.

Geliştirilen sistem sağlık kurumu içerisindeki anlık olarak gözlemlenmesi gereken gereken diğer varlıkların takibi için de kullanılabilir. Takip edilmesi istenen varlıklar otomatik tanımlama sisteminin tanıyabileceği etiketle işaretlenebilir ve yazılıma yapılacak ufak eklemelerle diğer varlıklar için lokasyon değişiklikleri kayıt altına alınabilir.

Geliştirilen sistem sayesinde insan müdahelesine gerek kalmadan algılayıcı cihazların birbirleriyle haberleşebilmesi sağlanmış ve tüm dosya hareketleri kayıt altına alınmıştır. Dosyaya ait tüm hareketler anlık olarak geliştirilmiş olan web arayüzü üzerinde gösterilmiştir. Sisteme, ihtiyaca göre farklı özellikler kolayca eklenebilir ve dijital dönüşüm kapsamında IoT teknolojisinden faydalanılması sağlanabilir.

KAYNAKLAR

Ajgaonkar, P., 2010, *Simulation Studies on ZigBee Communications for Home Automation and Networking*, Yüksek Lisans Tezi, The University of Toledo.

Ak, B., 2010, Tıp Bilişiminde Mobilite Uygulamaları, *Akademik Bilişim'10 - XII. Akademik Bilişim Konferansı Bildirileri*, 10 - 12 Şubat 2010 Muğla, Ankara, Nokta Matbaacılık, 1-5.

Aktaş, F., Çeken, C., Erdemli, Y.E., 2016, Nesnelerin İnterneti Teknolojisinin Biyomedikal Alanındaki Uygulamaları, *Düzce Üniversitesi Bilim ve Teknoloji Dergisi*, 4 (1), 37-54.

Akyildiz, F., Su., W., Sankarasubramaniam, E., Cayirci, E., 2002, Wireless Sensor Networks: A Survey", *Computer Networks*, 38 (4), 393-422.

Al-Odat, Z.A., Srinivasan, S.K., Al-qtiemat, E., Dubasi M.A.H., Shuja, S., 2016, IoT-Based Secure Embedded Scheme for Insulin Pump Data Acquisition and Monitoring, *The Third International Conference on Cyber-Technologies and Cyber-Systems*, 18-22 Kasım 2018 Atina - Yunanistan, Atina, IARIA, ISBN :978-1-61208-683-5, 90-93.

Alqahtani, F.H., 2018, The Application of the Internet of Things in Healthcare,*International Journal of Computer Applications*, 180 (18), 19-23.

Altuntaş, E.Y., 2018, Dijital Dönüşüm Uygulamalarının Kurumların Marka Değeri Üzerindeki Etkisi, *Ege Üniversitesi İletişim Fakültesi Medya ve İletişim Araştırmaları Hakemli E-Dergisi*, (2), 1-18.

Alumona, T.L., Idigo, V.E., Nnoli, K.P., 2014, Remote Monitoring of Patients Health using Wireless Sensor Networks (WSNs),*IPASJ International Journal of Electronics & Communication (IIJEC)*, 2 (9), 90-95.

Aydalka, S.C., 2018, *Bluetooth Beacon Teknolojisi Kullanarak Lokasyon ve Müzer Eser Tanıtım Uygulaması Gerçekleştirilmesi*, Yüksek Lisans Tezi, Balıkesir Üniversitesi.

Aydoğan, S., Metintaş, S., 2017, Türkiye'ye Gelen Dış Göç Ve Sağlığa Etkileri, *Türk Dünyası Uygulama Ve Araştırma Merkezi Halk Sağlığı Dergisi*, 2 (2), 37-45.

Bozdağ, Z., 2015, *Nesnelerin İnterneti İçin Tasarım Mimarisi*, Yüksek Lisans Tezi, Düzce Üniversitesi Fen Bilimleri Enstitüsü.

Bozdoğan, A., 2010, İdare Hukukunda İdarenin Hizmet Kusuru Ve Danıştay Uygulaması, *Türk İdare Dergisi*, (468), 33-48.

Bozkur, Ü.İ, Durdu, A., 2016, Akıllı Fabrikalarda Dağıtılmış Kontrol Sistemleri Uygulaması ve RFID Yaklaşımı, *Dicle Üniversitesi Mühendislik Fakültesi Mühendislik Dergisi*, 8 (3), 515-523.

Bozuklu, M., 2016, *Çevresel veriler ile gerçek zamanlı nesnelerin interneti uygulaması*, Yayınlanmamış Yüksek Lisans Tezi, Gaziosmanpaşa Üniversitesi.

Can, Ö., Sezer, E., Bursa, O., Ünalır M.O., 2016, Nesnelerin İnterneti ve Güvenli Bir Sağlık Bilgi Modeli Önerisi, *4th International Symposium on Innovative Technologies in Engineering and Science*, 3-5 Kasım 2016 Alanya/Antalya, 1201-1209.

Chen, N., 2016, *Bluetooth Low Energy Based CoAP Communication in IoT*, Yüksek Lisans Tezi, University of Saskatchewan.

Chiuchisan, I., Costin, H., Geman, O., 2014, Adopting the Internet of Things Technologies in Health Care Systems , *2014 International Conference and Exposition on Electrical and Power Engineering*, 16-18 Ekim 2014 Yaş/Romanya, New Jersey, IEEE, ISBN:978-1-4799-5849-8, 532-535.

Çapar, B., 2005, *Bilgi Yönetimi". Bilgi Çağı Bilgi Yönetimi ve Bilgi Sistemleri*, Çizgi Kitabevi, Konya.

Demir, B., Ayrancıoğlu, G., Gezer, C., Gözüaçık, N., 2016,6LoWPAN Kullanan Bir Algılayıcı Ağ Sistemi , *National Conference on Electrical, Electronics and Biomedical Engineering (ELECO)*, 1-3 Aralık 2016 Bursa, New York, IEEE, ISBN: 978-605-01-0923-8, 710-714.

Dimitrov, D.V., 2016, Medical Internet of Things and Big Data in Healthcare, *Healthcare Informatics Research*, 22 (3), 156–163.

Elsokah, M.M., Farkash, H.M., Zerek, A.R., 2019, Smart Beds For Hospitals with Internet of Things Solutions, *International Journal of Computer Science, Communications & Information Tecnology (CSIT)*, (7), 20-25.

Geylani, M., Çıbuk, M., Çınar, H., Ağgün, F., 2016, Geçmişten Günümüze Hücresel Haberleşme Teknolojilerinin Gelişimi, *Dokuz Eylül Üniversitesi Fen ve Mühendislik Dergisi*, 18 (3), 606-623.

Gökbunar, A.R., Uğur, A., Duramaz, S., 2016, Yaşlı Nüfusa Yönelik Sağlık Harcamalarının Azaltılmasında Kamusal Politikaların Önemi, *Ekonomik ve Sosyal Araştırmalar Dergisi*, 12 (1), 109-122.

Görmüş, S., Aydın, H., Ulutaş,G., 2018, Nesnelerin interneti teknolojisi için güvenlik: var olan mekanizmalar, protokoller ve yaşanılan zorlukların araştırılması, *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi*, 33 (4), 1247-1272.

Gözlü, M., Tatlıdil, H., 2015, Türkiye'deki 81 İlin Kamu Tarafından Sunulan Sağlık Hizmetlerine Erişim Durumları, *Sosyal Güvenlik Dergisi*, 5 (2), 145-165.

Güçlü, N., Sotirofski, K., 2006, Bilgi Yönetimi, *Türk Eğitim Bilimleri Dergisi*, 4 (4), 351-371.

Gülseçen, S., 2016, *Bilgi Yönetimi*, Papatya Yayıncılık Eğitim, İstanbul, ISBN: 978-605-4220-83-0.

Gültunca, C., 2018, *Nesnelerin İnternetinde Uygulama Katmanı Üzerindeki Haberleşme Protokollerinin İncelenmesi Ve Deneysel Karşılaştırılması*, Yüksek Lisans Tezi, T.C. İstanbul Ticaret Üniversitesi.

Gündüz, M.Z., Daş, R., 2017, Nesnelerin interneti: Gelişimi, bileşenleri ve uygulama alanları, *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, 24 (2), 327-335.

Harwani, B.M., 2011, *Introduction to Python Programming and Developing GUI Applications with PyQT*, Cengage Learning PTR, Boston, ISBN: 978-1435460973.

Hendriks, S., 2016, *How the world will be connected in 2025*, Yüksek Lisans Tezi, Utrecht University.

Ives, W., Torrey, B., Gordon, C., 1997, nowledge management: an emerging discipline with a long history, *Journal of Knowledge Management*, 1 (4), 269-274.

Jaisree, K., Sharmila, J., Jeevitha, J., Chandrakala K.R.S., 2017, Smart Hospitals Using Internet of Things(IoT), *International Journal of Trendy research in Engineering and Technology*, 1 (3), 1-3.

Kanase, P., Gaikwad, S., 2016, Smart Hospitals Using Internet of Things(IoT), *International Research Journal of Engineering and Technology*, 6 (7), 1220-1223.

Kantekin, U., 2018, *Nesnelerin İnternetinde Coap Protokolü İle Kablosuz Algılayıcı Ağların Güvenliğinin Sağlanması*, Yüksek Lisans Tezi, T.C. Maltepe Üniversitesi.

Karakaş, S., Rukancı, F., Anameriç, H., 2009, *Belge yönetimi ve arşiv terimleri sözlüğü*, Devlet Arşivleri Genel Müdürlüğü, Ankara.

Kaymak, Ç., 2016, *Raspberry Pi Devre Kartı Kullanarak Nesne Bulma ve Tanıma Algoritmalarının Bir Robot Kol Üzerinde Uygulanması*, Yüksek Lisans Tezi, T.C. Fırat Üniversitesi.

Kılıç, T., 2016, *E-Sağlık ve Teletıp: Hollanda ve Dünyadan İyi Uygulama Örnekleriyle*, Az Kitap, İstanbul, ISBN: 6059272339.

Köseoğlu, Ö., Demirci, Y., 2018, Akıllı Şehirler ver Yerel Sorunları Çözümünde Yenilikçi Teknolojilerin Kullanımı, *Uluslararası Politik Araştırmalar Dergisi*, 4 (2), 40-57.

Kutup, N., 2016, Nesnelerin Interneti; 4H Her yerden, Herkesle, Her zaman, Her nesne ile bağlantı,*XVI. Türkiye'de İnternet Konferansı*, 30 Kasım-2 Aralık 2011 İzmir, Ege Üniversitesi, 151-156.

Sebetci, Ö., Hanaylı, M.C., Dönük, G.G., 2017, Hastanelerin Dijitalleşme Sürecinde HIMSS-EMRAM Modeli Kullanımının Dünyada ve Türkiye'deki Genel Durumunun İncelenmesi, *İşletme Araştırmaları Dergisi*, 9 (4), 360-374.

Sekyere-Asiedu, D., 2018, *A Raspberry Pi Based System To Help The Visually Impaired At Home*, Yüksek Lisans Tezi, Yakın Doğu Üniversitesi.

Sezer, E., Ünalır, M.O., Yıldız,F., Gümüşkavak, A., Akçay, N., 2018, Sağlık Alanı için Kişiselleştirilmiş Nesnelerin İnterneti Platformu, *6th International Symposium on Innovative Technologies in Engineering and Science*, 09-11 Kasım 2018 Antalya, Sakarya, Apjes, 311-320.

Shaikh, A.A., Gupta, N.S.,Khan, A.D.M., Artist, H.T., 2017, Android and Internet of Things (IOT) Based Alzheimer Care/Rehabilitation System to Moniter and Progress Patient Health Condition, *International Journal of Innovative Research in Computer and Communication Engineering*, 5 (3), 5531-5539.

Shams, R., Khan, F.H.,Aamir, M., Saleem, F., 2014, Internet of Things in Telemedicine: a Discussion Regarding to Several Implementation, *Journal of Computer Science of Newports Institute of Communications and Economics*, 5 (2014), 17-26.

Shi, E., Perrig, A., 2004, Designing secure sensor networks, *IEEE Wireless Communications Management*, 11 (6), 38-43.

Malkoç, B., 2012, Temel Bilimler ve Mühendislik Eğitiminde Programlama Dili Olarak Python, *Akademik Bilişim'12 - XIV. Akademik Bilişim Konferansı Bildirileri*, 1 - 3 Şubat 2012 Uşak Üniversitesi, İstanbul, İnternet Teknolojileri Derneği , 201-210.

McFarlane, D., Agnihotri, P., Dwivedi, JK., 2018, Advanced IoT Based Combined Remote Health Monitoring and Alarm System, Int*International Journal of Advance Research and Development*, 3 (4), 6-11.

Misra, A., Sheffi, Y., 2003, The impact of automatic identification on supply chain operations, *The International Journal of Logistics Management*, 14 (1), 1-17.

Nuroğlu, E., Nuroğlu, H.H., 2018, Türkiye ve Almanya'nın Sanayide Dijital Dönöüşümü: Yol Haritaları ve Şirketlerin Karşılaştırması, *Süleyman Demirel Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi*, Endüstri 4.0 ve Örgütsel Değişim Özel Sayısı, 23, 1537-1560.

Özcan, Ö., 2011, *Kablosuz Sensör Ağları İçin Pic Tabanlı Sensör Düğümü Tasarımı*, Yüksek Lisans Tezi, T.C. Selçuk Üniversitesi.

Özdağ, R., 2017, Kablosuz Algılayıcı Ağlarda Olasılıksal Kapsama Oranının Optimizasyonu için Yeni Bir Meta-Sezgisel Yaklaşım, *Bilişim Teknolojileri Dergisi*, 10 (4), 461-473.

Özdemir, N.C., Hürses, İ., Özgün, Y., 2017, *İnsan Odaklı Hareket ve Varlık Algılama Sensörleri*, http://www.emo.org.tr/ekler/b99e1cdf4361091_ek.pdf, [Ziyaret Tarihi: 4 Mayıs 2019]

Özvural, G., 2016, *Nesnelerin İnterneti İçin Sistem Tasarımı Ve Kablosuz Kişisel Alan Ağlarında Ağ Kodlama Uygulamalar*,Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü.

Paiva, S., Brito, D., Leiva-Marcon, L., 2018, Real Time Location Systems Adoption in Hospitals - A Review and A Case Study for Locating Assets, *Acta Scientific Medical Sciences* , 2 (7), 2-17.

Seçkin, A.Ç., 2011, *Kablosuz Sensör Ağlarının Ip Tabanlı Ağlarla Birleştirilmesi*, Yüksek Lisans Tezi, Pamukkale Üniversitesi Fen Bilimleri Enstitüsü.

Şişmanyazıcı, D., Doğan, B., 2016, Nesnelerin İnternetinde Veri Madenciliği, *Tekirdağ, International Conference on Computer Science and Engineering*, 20-23 Ekim 2016 Tekirdağ, 420-425.

Tao, F., Fan, T.,Lai, K.K.,Li, L., 2016, Impact of RFID technology on inventory control policy, *Journal of the Operational Research Society*, 68 (2), 207-220.

Tekin, Ç.S., Kara, F., 2018, Dünyada ve Türkiyede Yaşlılık, *Uluslararası Bilimsel Araştırmalar Dergisi*, 3 (1), 219-229.

Tozkoparan, G., Ernur, O., 2018, *Dijital Dönüşüm Perspektifinde Endüstri 4.0 Sürecindeki İşletmelerin Karşılaştığı Durumlar Üzerine Bir Vaka Çalışması*, https://www.researchgate.net/publication/328432979_dijital_donusum_perspektifinde_endustri_40_surecindeki_isletmelerin_karsilastigi_durumlar_uzerine_bir_vaka_calismasi, [Ziyaret tarihi: 2 Şubat 2019].

Tüfekçi, N., Yorulmaz, R., Cansever, İ.H., 2017, Dijital Hastane, *Journal of Current Researches on Health Sector*, 7 (2), 143-156.

Uğuz, S., Kılıç, B., Şişeci, M., 2013, *Akıllı Ev Otomasyonu Sistemlerinde Zigbee Tabanlı Ağ Uygulamaları*, https://www.researchgate.net/publication/305001027_akill_ev_otomasyonu_sistemlerinde_zigbee_tabanli_ag_uygulamalari, [Ziyaret tarihi: 19 Nisan, 2019].

Ulaş, S., 2015, *Nesnelerin İnterneti Ekosisteminde Makineler Arası Özerk İletişim*, Yayınlanmamış Yüksek Lisans Tezi, Gazi Üniversitesi.

Wortmann, F., Flüchter, K., 2015, Internet of Things, *Business Information Systems Engineering*, 57 (3), 221-224.

Yankın, F.B., 2019, Dijital Dönüşüm Sürecinde Çalışma Yaşamı,*Trakya Üniversitesi İktisadi ve İdari Bilimler Fakültesi E-Dergi*, 7 (2), 1-38.

Yücel, Ş., 2015, *Use Of Bluetooth Technology For Traffic Analysis In Urban Road Networks*, Yüksek Lisans Tezi, Orta Doğu Teknik Üniversitesi.

Yüksel, M.E.,Zaim, A.H., 2009, *Otomatik Nesne Tanımlama ve Takibinde, Veri Yönetimi ve Analiz Sistemlerinde Rfid Üstünlükleri*, http://www.emo.org.tr/ekler/c005118de912f94_ek.pdf, [Ziyaret tarihi: 15 Nisan 2019].

Yoo, Y., 2010, *Digitalization and Innovation*, https://www.researchgate.net/publication/48926382_Digitalization_and_Innovation, [Ziyaret tarihi: 5 Mayıs 2019].

Valade, J., 2010, *PHP and MySQL for dummies*, Wiley Publishing Inc., New Jersey, ISBN:978-0470527580.

EKLER

EK 1. ÇALIŞTIRILABİLİR KODLAR

NodeMcu Kodları

```
#include <SPI.h>
#include <MFRC522.h>
#include <PubSubClient.h>
#include <ESP8266WiFi.h>
#include <ArduinoJson.h>

const char* ssid = "Kablosuz_Ag_Adi";
const char* password = "Kablosuz_Ag_Sifresi";
const char* mqtt_server = "Arabulucu_Adresi";
#define mqtt_port 1883
#define MQTT_USER "username"
#define MQTT_PASSWORD "password"
#define MQTT_SERIAL_PUBLISH_CH "hasta-dosyasi-takip"
#define MQTT_SERIAL_RECEIVER_CH "hasta-dosyasi-takip"
#define Oda "Oda-1"
#define Yon "G"
#define SS_PIN D4
#define RST_PIN D2

WiFiClient wifiClient;

PubSubClient client(wifiClient);

MFRC522 mfrc522(SS_PIN, RST_PIN);

void setup() {
```

```
        Serial.begin(9600);
        SPI.begin();        //  SPI bus
        mfrc522.PCD_Init(); //  MFRC522
        setup_wifi();
        client.setServer(mqtt_server, mqtt_port);
        client.setCallback(callback);
        reconnect();
        Serial.println("RFID okunuyor");
}


void loop() {
if ( mfrc522.PICC_IsNewCardPresent())
{
        String rfidTag = "";
        if ( mfrc522.PICC_ReadCardSerial())
        {
                Serial.print("Tag UID:");
                for (byte i = 0; i < mfrc522.uid.size; i++) {
                String rfidTag_part = String
                (mfrc522.uid.uidByte[i], HEX);
                Serial.print(mfrc522.uid.uidByte[i]
                 < 0x10 ? " 0" : " ");
                Serial.print(mfrc522.uid.uidByte[i], HEX);
                rfidTag += rfidTag_part; }
                Serial.println();
                StaticJsonDocument<200> doc;
                doc["dosyaNo"] = rfidTag;
                doc["oda"] = Oda;
                doc["yon"]=Yon;
                String mesaj = doc.as<String>();
                Serial.print("Gonderilecek mesaj: ");
                Serial.println(mesaj);
```

```
                int mesajBoyutu = mesaj.length() + 1;
                char charDizisi[mesajBoyutu];
                mesaj.toCharArray(charDizisi, mesajBoyutu);
                Serial.println("MQTT brokera mesaj
                yollaniyor...");
                publishSerialData(charDizisi);
                Serial.println();
                mfrc522.PICC_HaltA();
                }
        }
}


void setup_wifi() {
        delay(10);
        // We start by connecting to a WiFi network
        Serial.println();
        Serial.print("Baglaniliyor ");
        Serial.println(ssid);
        WiFi.begin(ssid, password);
        while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
        }
        randomSeed(micros());
        Serial.println("");
        Serial.println("WiFi agina baglanildi");
        Serial.println("IP adresi: ");
        Serial.println(WiFi.localIP());
}


void reconnect() {
```

```
        // Baglanti saglanana kadar dene
        while (!client.connected()) {
        Serial.print("MQTT brokera
        baglaniliyor...");
        // Create a random client ID
        String clientId = "NodeMCUClient-";
        clientId += String(random(0xffff), HEX);
        // Attempt to connect
        if (client.connect(clientId.c_str())) {
                Serial.println("baglanti saglandi.");
        } else {
                Serial.print("failed, rc=");
                Serial.print(client.state());
                Serial.println(" 5 saniye
                icerisinde tekrar baglanmaya calisilacak.");
                // Tekrar denemeden once 5 saniye bekle
                delay(5000);
          }
        }
}

void callback(char* topic, byte *payload,
unsigned int length) {
        Serial.println("--Arabulucudan yeni mesaj--");
        Serial.print("channel:");
        Serial.println(topic);
        Serial.print("data:");
        Serial.write(payload, length);
        Serial.println();
}

void publishSerialData(char *serialData){
        if (!client.connected()) {
```

```
                reconnect();
        }
        client.publish(MQTT_SERIAL_PUBLISH_CH, serialData);
}
```

Veri Kayıt Uygulaması

```
import paho.mqtt.client as mqtt
import MySQLdb
import json
from collections import namedtuple
import datetime

class Log:
        def __init__(self, oda, dosyaNo,yon,kayitTarihi):
                self.Oda = oda
                self.DosyaNo=dosyaNo
                self.Yon=yon
                self.KayitTarihi=kayitTarihi

class Hasta:
        def__init__(self,id,ad,soyad,cinsiyet,dogumTarihi
        ,kayitTarihi,
        dosyaNo,sayisallastirildi,dosyaninBulunduguOda,
        dosyaninOdayaGirisTarihi,
        dosyaninBulunduguOncekiOda):
                self.Id = id
                self.Ad=ad
                self.Soyad=soyad
                self.Cinsiyet=cinsiyet
                self.DogumTarihi=dogumTarihi
                self.KayitTarihi=kayitTarihi
                self.DosyaNo=dosyaNo
                self.Sayisallastirildi=sayisallastirildi
```

```
                self.DosyaninBulunduguOda=
                dosyaninBulunduguOda
                self.DosyaninOdayaGirisTarihi=
                dosyaninOdayaGirisTarihi
                self.DosyaninBulunduguOncekiOda=
                dosyaninBulunduguOncekiOda

db = MySQLdb.connect(host="localhost",
                user="root",
                passwd="sifre",
                db="HastaDosyasiTakip")
                mycursor = db.cursor()
                db.set_character_set('utf8')
                mycursor.execute('SET NAMES utf8;')
                mycursor.execute('SET CHARACTER SET utf8;')
                mycursor.execute
                ('SET character_set_connection=utf8;')



odalar=[]
def on_connect(client, userdata, flags, rc):
        print("MQTT Broker' a baglanildi "+str(rc))
        client.subscribe("hasta-dosyasi-takip")

def logKaydet(dosyaNo,oda,yon,kayitTarihi):
        try:
                sql = "INSERT INTO Log (DosyaNo, Oda,
                Yon,KayitTarihi) VALUES (%s, %s,%s,%s)"
                val = (dosyaNo, oda, yon, kayitTarihi)
                mycursor.execute(sql, val)
                db.commit()
                print("Log Veritabani: 1 kayit girildi, ID:"
```

```
                , mycursor.lastrowid)
        except Exception as e:
                db.rollback()
                print("Hata olustu {}", e)

def odaBilgileriniGetir():
        try:
                odaSorgusu = "select * from Oda"
                cursor = db.cursor()
                cursor.execute(odaSorgusu)
                records = cursor.fetchall()
                for row in records:
                odalar.append(row)
                cursor.close()
        except Exception as e:
                db.rollback() #rollback if any exception
                occured
                print("Oda bilgileri veritabanindan
                cekilirken bir hata olustu: {}", e)
                cursor.close()
                db.close()

def logKayitlariniGetirByDosyaNo(dosyaNo):
        try:
                logList=[]
                logSorgusu = "select * from Log
            where DosyaNo= %s ORDER BY KayitTarihi DESC"
                cursor = db.cursor()
                cursor.execute(logSorgusu,(dosyaNo,))
                records = cursor.fetchall()
                if not records:
                        print("Dosya No ile iliskili
                        log kaydi bulunamadi.
```

```
                    Sisteme ilk kez girisi yapiliyor
                    olabilir.")
                        return None
                for row in records:
                logList.append(Log(row[1],row[2],
                row[3],row[4]))
                cursor.close()
                return logList
        except Exception as e:
                print("Log kayitlari veritabanindan
                cekilirken
            bir hata olustu: {}", e())

def hastaKaydiniGetir(dosyaNo):
        try:
                hastaSorgusu = "select * from Hasta
                where DosyaNo= %s"
                cursor = db.cursor()
                cursor.execute(hastaSorgusu,(dosyaNo,))
                record = cursor.fetchone()
                if not record:
                        print("Dosya No ile iliskili bir
                        hasta kaydi
                        bulunamadi. Hasta kaydini sistemden
                        kontrol ediniz.
                        Isleme devam edilemiyor.")
                        return None
                hastaKaydi=Hasta(record[0],record[1],
                record[2],
                record[3],record[4],record[5],record[6],
                record[7],
                record[8],record[9],record[10])
                cursor.close()
```

```
                return hastaKaydi
        except Exception as e:
                print("Hasta kaydi veritabanindan
                cekilirken bir hata olustu: {}", e())


def hastaDosyasiGirisGuncelle(hastaKayitId,odaId,dosyaNo):
        try:
                hastaDosyasiGirisGuncelle = "update Hasta set
            DosyaninBulunduguOda= %s,
            DosyaninOdayaGirisTarihi=%s where Id=%s"
                cursor = db.cursor()
                cursor.execute(hastaDosyasiGirisGuncelle,
                (odaId,datetime.datetime.now(),hastaKayitId))
                db.commit()
                print(cursor.rowcount, "kayit etkilendi.")
                if (cursor.rowcount>0):
                        hastaDosyasiGirisLoguKaydet(
                        hastaKayitId
                        ,odaId,dosyaNo)
                        cursor.close()
        except Exception as e:
                db.rollback()
                print("Hasta kaydi dosya giris yapilmis
                olarak
            guncellenirken bir hata olustu: {}", e())

def hastaDosyasiGirisLoguKaydet(hastaKayitId,odaId,dosyaNo):
        try:
                sql = "INSERT INTO DosyaGirisCikisLog
                (DosyaNo, HastaId, IslemAciklama,KayitTarihi
                ,OdaId) VALUES (%s, %s,%s,%s,%s)"
                val = (dosyaNo, hastaKayitId,
```

```
                "Giris", datetime.datetime.now(),odaId)
                mycursor.execute(sql, val)
                db.commit()
                print("HastaGirisCikisLog Veritabani:
                 1 kayit eklendi, ID:", mycursor.lastrowid)
        except Exception as e:
                db.rollback()
                print("Hata olustu {}", e)

def hastaDosyasiCikisGuncelle(hastaKayitId,
dosyaninBulunduguOda,dosyaNo):
        try:
                hastaDosyasiCikisGuncelle = "UPDATE Hasta
            SET DosyaninBulunduguOda= %s,
            DosyaninOdayaGirisTarihi=%s,
            DosyaninBulunduguOncekiOda=%s WHERE Id=%s"
                cursor = db.cursor()
                cursor.execute(hastaDosyasiCikisGuncelle,
                (None,None,dosyaninBulunduguOda,
                hastaKayitId))
                db.commit()
                print(cursor.rowcount, "kayit eklendi")
                if (cursor.rowcount>0):
                        hastaDosyasiCikisLoguKaydet(
                        hastaKayitId,
                        dosyaninBulunduguOda,dosyaNo)
                        cursor.close()
        except Exception as e:
                db.rollback()
                print("Hasta kaydi dosya
                cikisi yapilmis olarak
                guncelllenirken bir hata olustu: {}", e())
```

```
def hastaDosyasiCikisLoguKaydet
(hastaKayitId ,odaId ,dosyaNo):
        try:
                sql = "INSERT INTO
                DosyaGirisCikisLog (DosyaNo,
            HastaId, IslemAciklama ,KayitTarihi ,OdaId)
            VALUES (%s, %s,%s,%s,%s)"
                val = (dosyaNo, hastaKayitId,
                "Cikis", datetime.datetime.now(),
                odaId)
                mycursor.execute(sql, val)
                db.commit()
                print("HastaGirisCikisLog Veritabani:
                1 kayit eklendi, ID:", mycursor.lastrowid)
        except Exception as e:
                db.rollback()
                print("Hata olustu {}", e)

#MQTT uzerinden mesaj geldiginde asagidaki metod calisir
def on_message(client, userdata, msg):
        mesaj=msg.payload.decode()
        print("Gelen mesaj: ",mesaj)
        #json formatina cevrilir
        data = json.loads(msg.payload.decode())
        dosyaNo=data["dosyaNo"]
        oda=data["oda"]
        yon=data["yon"]
        kayitTarihi=datetime.datetime.now()
        odaAdi=data["oda"]
        #log kaydetme islemi yapilir
        logKaydet(dosyaNo,oda,yon,kayitTarihi)
        for kayit in odalar:
                if kayit[1] == oda:
```

```
            oda=kayit[0]
            break
    else:
            odaBilgileriniGetir()
            if kayit[1] == oda:
                    oda=kayit[0]
                    break
            else:
                    oda=None
                    break

    if (oda==None):
            print("Mesajda belirtilen
            oda sistemde
        kayitli gorunmuyor. Oda kaydinini
        yapilmasi gerekmektedir.
        Isleme devam edilemiyor.")
            return

    #Hasta kaydi getirilir
    hastaKaydi=hastaKaydiniGetir(dosyaNo)

    if (hastaKaydi==None):
            print("Hasta kaydi bulunmadigindan
            isleme devam edilmiyor.")
            return

    #karsilastirma icin log kayitlari getirilir
    loglar=logKayitlariniGetirByDosyaNo(dosyaNo)
    print(len(loglar))
    if (len(loglar)==1):
            print("Belirtilen dosya no
        ile iliskili bir
```

```
            log kaydi bulunamadi.")
                return

        if (loglar[1].Yon!=yon and
        loglar[1].Oda==odaAdi):
                print("Hasta tablosu guncellenmeli")
                if (yon=="C"):
                        print("Dosya odadan cikis
                        yapiyor.")
                        print(hastaKaydi.Id)
                        hastaDosyasiCikisGuncelle
                        (hastaKaydi.Id,
                        hastaKaydi.
                        DosyaninBulunduguOda,dosyaNo)
                        client.publish
                        ("dosyaGirisCikis","{} '
                        a ait dosya {} adli
                        odadan CIKIS yapti."
                        .format(hastaKaydi.Ad+" "
                        +hastaKaydi.Soyad,odaAdi))
                elif (yon=="G"):
                        print("Dosya odaya giriyor.")
                        hastaDosyasiGirisGuncelle
                        (hastaKaydi.Id,
                        oda,dosyaNo)
                        client.publish
                        ("dosyaGirisCikis","{} 'a
                    ait dosya {} adli odaya
                    GIRIS yapti."
                    .format(hastaKaydi.Ad
                    +" "+hastaKaydi.Soyad,
                    odaAdi))
                else:
```

```
                                    print("Hasta tablosu
                                    GUNCELLENMEMELI")

def on_publish(client,userdata,result):
        print("mesaj gonderildi \n")

odaBilgileriniGetir()
client = mqtt.Client()
client.connect("192.168.2.169",1883,60)

client.on_connect = on_connect
client.on_message = on_message
client.on_publish=on_publish

client.loop_forever()
```

Web Servis Uygulaması

```
from flask import Flask, jsonify
import MySQLdb
import json
from collections import namedtuple
import datetime
from flask import Response
from flask import flash, request

db = MySQLdb.connect(host="localhost",
user="root",
passwd="sifre",
db="HastaDosyasiTakip")
mycursor = db.cursor()
db.set_character_set('utf8')
mycursor.execute('SET NAMES utf8;')
mycursor.execute('SET CHARACTER SET utf8;')
```

```
mycursor.execute('SET character_set_connection=utf8;')

class Hasta:
        def __init__(self, id, ad,soyad,cinsiyet,
        dogumTarihi,
        kayitTarihi,dosyaNo,sayisallastirildi,
        dosyaninBulunduguOda,dosyaninOdayaGirisTarihi,
        dosyaninBulunduguOncekiOda):
                self.Id = id
                self.Ad=ad
                self.Soyad=soyad
                self.Cinsiyet=cinsiyet
                self.DogumTarihi=dogumTarihi
                self.KayitTarihi=kayitTarihi
                self.DosyaNo=dosyaNo
                self.Sayisallastirildi=
                sayisallastirildi
                self.DosyaninBulunduguOda=
                dosyaninBulunduguOda
                self.DosyaninOdayaGirisTarihi=
                dosyaninOdayaGirisTarihi
                self.DosyaninBulunduguOncekiOda=
                dosyaninBulunduguOncekiOda
        def serialize(self):
                return {
                        'id': self.Id,
                        'ad': self.Ad,
                        'soyad': self.Soyad,
                        'cinsiyet':self.Cinsiyet,
                        'dogumTarihi':self.DogumTarihi,
                        'kayitTarihi':self.KayitTarihi,
                        'dosyaNo':self.DosyaNo,
                        'sayisallastirildi':self.
```

```
                        Sayisallastirildi,
                        'dosyaninBulunduguOda': self.
                        DosyaninBulunduguOda,
                        'dosyaninOdayaGirisTarihi': self.
                        DosyaninOdayaGirisTarihi,
                        'dosyaninBulunduguOncekiOda': self.
                        DosyaninBulunduguOncekiOda,
                        }

class Oda:
        def __init__(self, id,odaAdi,kayitTarihi):
                self.Id = id
                self.OdaAdi=odaAdi
                self.KayitTarihi=kayitTarihi
        def serialize(self):
                return {
                        'id': self.Id,
                        'odaAdi': self.OdaAdi,
                        'kayitTarihi': self.KayitTarihi,
                        }

class DosyaGirisCikisLog:
        def __init__(self, id,dosyaNo,hastaId,
        islemAciklama,kayitTarihi,odaId):
                self.Id=id
                self.DosyaNo=dosyaNo
                self.HastaId=hastaId
                self.IslemAciklama=islemAciklama
                self.KayitTarihi=kayitTarihi
                self.OdaId=odaId
        def serialize(self):
                return {
                        'id':self.Id,
```

```
                        'dosyaNo': self.DosyaNo,
                        'hastaId': self.HastaId,
                        'islemAciklama': self.IslemAciklama,
                        'kayitTarihi': self.KayitTarihi,
                        'odaId': self.OdaId,
                }

class Log:
        def __init__(self, oda, dosyaNo,yon,kayitTarihi):
                self.Oda = oda
                self.DosyaNo=dosyaNo
                self.Yon=yon
                self.KayitTarihi=kayitTarihi
        def serialize(self):
                return {
                        'id': self.Id,
                        'dosyaNo': self.DosyaNo,
                        'yon': self.Yon,
                        'kayitTarihi': self.KayitTarihi,
                }

app = Flask(__name__)
app.config['JSON_AS_ASCII'] = False

@app.route('/service/api/hastalar', methods=['GET'])
def hastaBilgileriniGetir():
        try:
                hastaListesi=[]
                hastaSorgusu = "select * from Hasta"
                cursor = db.cursor()
                cursor.execute(hastaSorgusu)
                records = cursor.fetchall()
                print(records)
```

```
                if not records:
                        print("Dosya No ile iliskili
                        bir hasta kaydi bulunamadi.
                        Hasta kaydini sistemden
                    kontrol ediniz.
                    Isleme devam edilemiyor.")
                        return None
                for record in records:
                        hastaListesi.append(Hasta(record[0],
                        record[1],record[2],record[3],
                        record[4],record[5],record[6],
                        record[7],record[8],record[9],
                        record[10]))
                hastaListesiJson=jsonify(hastalar=
                [e.serialize() for e in hastaListesi])
                print(hastaListesiJson)
                return hastaListesiJson
        except Exception as e:
                return '500 Server error',500
        finally:
                cursor.close()

@app.route('/service/api/odalar', methods=['GET'])
def odaBilgileriniGetir():
        try:
                odaListesi=[]
                odaSorgusu = "select * from Oda"
                cursor = db.cursor()
                cursor.execute(odaSorgusu)
                records = cursor.fetchall()
                print(records)
                if not records:
                        print("Oda kaydi bulunamadi.")
```

```
                return None
            for record in records:
            odaListesi.append(Oda(record[0],record[1],
            record[2]))
            print(odaListesi)
            odaListesiJson=jsonify(odalar=
            [e.serialize() for e in odaListesi])
            print(odaListesiJson)
            return odaListesiJson
        except Exception as e:
            return '500 Server error',500
        finally:
            cursor.close()


@app.route('/service/api/dosyagiriscikis', methods=['GET'])
def dosyaGirisCikisLogListele():
        try:
            dosyaGirisCikisLogListe=[]
            dosyaGirisCikisLogSorgusu = "select *
            from DosyaGirisCikisLog ORDER BY KayitTarihi
            DESC"
            cursor = db.cursor()
            cursor.execute(dosyaGirisCikisLogSorgusu)
            records = cursor.fetchall()
            print(records)
            if not records:
                print("Dosya giris cikis log kaydi
                bulunamadi.")
                return None
            for record in records:
            dosyaGirisCikisLogListe.append(
            DosyaGirisCikisLog(record[0],record[1]
```

```
                ,record[2],record[3],record[4],record[5]))
                print(dosyaGirisCikisLogListe)
                dosyaGirisCikisLogListeJson=jsonify
                (dosyaGirisCikis=[e.serialize() for
                e in dosyaGirisCikisLogListe])
                print(dosyaGirisCikisLogListeJson)
                return dosyaGirisCikisLogListeJson
        except Exception as e:
                return '500 Server error',500
                finally:
                cursor.close()


@app.route('/service/api/hastakaydet', methods=['POST'])
def hastaKaydet():
        try:
                jsonRequest = request.json
                ad = jsonRequest['ad']
                soyad = jsonRequest['soyad']
                cinsiyet = jsonRequest['cinsiyet']
                dogumTarihi = jsonRequest['dogumTarihi']
                kayitTarihi = jsonRequest['kayitTarihi']
                dosyaNo = jsonRequest['dosyaNo']
                sayisallastirildi =
                jsonRequest['sayisallastirildi']
                dosyaninBulunduguOda =
                jsonRequest['dosyaninBulunduguOda']
                dosyaninOdayaGirisTarihi = jsonRequest
                ['dosyaninOdayaGirisTarihi']
                dosyaninBulunduguOncekiOda = jsonRequest
                ['dosyaninBulunduguOncekiOda']
                sql = "INSERT INTO Hasta (Ad, Soyad,
                Cinsiyet, DogumTarihi, KayitTarihi, DosyaNo,
```

```
                Sayisallastirildi , DosyaninBulunduguOda ,
                dosyaninOdayaGirisTarihi ,
                DosyaninBulunduguOncekiOda)
                VALUES (%s , %s,%s,%s,%s ,
                %s,%s,%s,%s,%s)"
                val = (ad, soyad, cinsiyet ,
                dogumTarihi ,kayitTarihi ,dosyaNo ,
                sayisallastirildi , dosyaninBulunduguOda ,
                dosyaninOdayaGirisTarihi ,
                dosyaninBulunduguOncekiOda)
                cursor = db.cursor()
                cursor.execute(sql , val)
                db.commit()
                if (cursor.rowcount>0):
                        print("Hasta Veritabani:
                        1 kayit girildi , ID:",
                        cursor.lastrowid)
                if (dosyaninBulunduguOda>0):
                        hastaDosyasiGirisLoguKaydet(
                        cursor.lastrowid ,
                        dosyaninBulunduguOda ,dosyaNo)
                else:
                        db.rollback()
                        return 'Kayit eklenirken bir
                        hata olustu.',400
                return 'Kayit basariyla eklendi.',200
        except Exception as e:
                db.rollback()
                return '500 Server error',500



@app.route('/service/api/hastaguncelle', methods=['POST'])
```

```
def hastaGuncelle():
        try:
                jsonRequest = request.json
                id=jsonRequest['id']
                ad = jsonRequest['ad']
                soyad = jsonRequest['soyad']
                cinsiyet = jsonRequest['cinsiyet']
                dogumTarihi = jsonRequest['dogumTarihi']
                kayitTarihi = jsonRequest['kayitTarihi']
                dosyaNo = jsonRequest['dosyaNo']
                sayisallastirildi = jsonRequest
                ['sayisallastirildi']
                dosyaninBulunduguOda = jsonRequest
                ['dosyaninBulunduguOda']
                dosyaninOdayaGirisTarihi = jsonRequest
                ['dosyaninOdayaGirisTarihi']
                dosyaninBulunduguOncekiOda = jsonRequest
                ['dosyaninBulunduguOncekiOda']
                sql = "UPDATE Hasta SET Ad=%s, Soyad=%s,
                Cinsiyet=%s, DogumTarihi=%s, KayitTarihi=%s,
                DosyaNo=%s, Sayisallastirildi=%s,
                DosyaninBulunduguOda=%s,
                dosyaninOdayaGirisTarihi=%s,
                DosyaninBulunduguOncekiOda=%s Where Id=%s"
                val = (ad, soyad, cinsiyet, dogumTarihi,
                kayitTarihi ,dosyaNo,
                sayisallastirildi , dosyaninBulunduguOda,
                dosyaninOdayaGirisTarihi,
                dosyaninBulunduguOncekiOda,id)
                cursor = db.cursor()
                cursor.execute(sql, val)
                db.commit()
                if (cursor.rowcount>0):
```

```
                print("Hasta Veritabani: 1 kayit
                guncellendi, ID:", cursor.lastrowid)
            if (dosyaninBulunduguOda>0):
                hastaDosyasiGirisLoguKaydet
                (cursor.lastrowid,
                dosyaninBulunduguOda,
                dosyaNo)
            else:
                db.rollback()
                return 'Kayit guncellenirken
                bir hata olustu.',400
            return 'Kayit basariyla guncellendi.',200
        except Exception as e:
            print(e)
            db.rollback()
            return '500 Server error',500


def hastaDosyasiGirisLoguKaydet(hastaKayitId,odaId,dosyaNo):
        try:
            sql = "INSERT INTO DosyaGirisCikisLog
            (DosyaNo, HastaId, IslemAciklama,KayitTarihi,
            OdaId)
            VALUES (%s, %s,%s,%s,%s)"
            val = (dosyaNo, hastaKayitId,
            "Giris", datetime.datetime.now(),odaId)
            mycursor.execute(sql, val)
            db.commit()
            print("HastaGirisCikisLog Veritabani:
            1 kayit eklendi, ID:", mycursor.lastrowid)
        except Exception as e:
            print(e)
```

```
                db.rollback()
                return '500 Server error',500


@app.route('/service/api/odakaydet', methods=['POST'])
def odaKaydet():
        try:
                jsonRequest = request.json
                print(jsonRequest)
                odaAdi = jsonRequest['odaAdi']
                kayitTarihi = jsonRequest['kayitTarihi']

                sql = "INSERT INTO Oda
                (OdaAdi, KayitTarihi) VALUES (%s, %s)"
                val = (odaAdi, kayitTarihi)
                cursor = db.cursor()
                cursor.execute(sql, val)
                db.commit()
                if (cursor.rowcount>0):
                        print("Oda Veritabani: 1
                        kayit girildi,
                        ID:", cursor.lastrowid)
                else:
                        db.rollback()
                        return 'Kayit eklenirken
                        bir hata olustu.',400
                return 'Kayit basariyla eklendi.',200
        except Exception as e:
                db.rollback()
                return '500 Server error',500
```

```
@app.route('/service/api/odakguncelle', methods=['POST'])
def odaGuncelle():
        try:
                jsonRequest = request.json
                print(jsonRequest)
                id=jsonRequest['id']
                odaAdi = jsonRequest['odaAdi']
                kayitTarihi = jsonRequest['kayitTarihi']

                sql = "Update Oda SET OdaAdi=%s,
                KayitTarihi=%s WHERE Id=%s"
                val = (odaAdi, kayitTarihi,id)
                cursor = db.cursor()
                cursor.execute(sql, val)
                db.commit()
                if (cursor.rowcount>0):
                        print("Oda Veritabani: 1 kayit
                        guncellendi, ID:", cursor.lastrowid)
                else:
                        db.rollback()
                        return 'Kayit guncellenirken
                         bir hata olustu.',400
                return 'Kayit basariyla guncellendi.',200
        except Exception as e:
                db.rollback()
                return '500 Server error',500



@app.route('/service/api/dosyagiriscikisbydosyano',
methods=['POST'])
def dosyaGirisCikisLogListeleByDosyaNo():
        try:
```

```
            dosyaGirisCikisLogListe=[]
            jsonRequest = request.json
            dosyaNo = jsonRequest['dosyaNo']
            print(jsonRequest)
            dosyaGirisCikisLogSorgusu = "select *
        from DosyaGirisCikisLog where DosyaNo= %s
        ORDER BY KayitTarihi DESC"
            cursor = db.cursor()
            cursor.execute(dosyaGirisCikisLogSorgusu,
            (dosyaNo,))
            records = cursor.fetchall()
            print(records)
            if not records:
                    print("Dosya giris cikis log kaydi
                    bulunamadi.")
                    return 'Kayit bulunumadi.',204
            for record in records:
                    dosyaGirisCikisLogListe.append
                    (DosyaGirisCikisLog(record[0],
                    record[1],
                    record[2],
                    record[3],
                    record[4],
                    record[5]))
                    print(dosyaGirisCikisLogListe)
            dosyaGirisCikisLogListeJson=jsonify
            (dosyaGirisCikis=[e.serialize() for
            e in dosyaGirisCikisLogListe])
            print(dosyaGirisCikisLogListeJson)
            return dosyaGirisCikisLogListeJson
    except Exception as e:
            return '500 Server error',500
            finally:
```

```
                cursor.close()
if __name__ == '__main__':
app.run(host='0.0.0.0')
```

ÖZGEÇMİŞ

Kişisel Bilgiler	
Adı Soyadı	Murat TEKBAŞ
Doğum Yeri	İstanbul
Doğum Tarihi	11.06.1983
Uyruğu	[X] T.C. [ ] Diğer:
E-Posta Adresi	murat_tekbas@yahoo.com

Eğitim Bilgileri	
Lisans	
Üniversite	Eskişehir Osmangazi Üniversitesi
Fakülte	Mühendislik - Mimarlık Fakültesi
Bölümü	Bilgisayar Mühendisliği
Mezuniyet Yılı	2011

Yüksek Lisans	
Üniversite	İstanbul Üniversitesi
Enstitü Adı	Fen Bilimleri
Anabilim Dalı	Enformatik Anabilim Dalı
Programı	Enformatik Programı
Mezuniyet Tarihi	2019