MARMARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

AKILLI BİNALARDA WEB TEKNOLOJİLERİNİN KULLANIMI

MEHMET ŞENSOY

523119037

YÜKSEK LİSANS TEZİ

Elektrik Elektronik Mühendisliği Anabilim Dalı
Elektrik Elektronik Mühendisliği Programı

DANIŞMAN

Doç. Dr. Nazmi EKREN

İSTANBUL, 2022

TEŞEKKÜR

Öncelikle desteklerinden dolayı tez danışmanım Doç. Dr. Nazmi EKREN'e, teşvikleri için Prof. Dr. Mustafa AYKAÇ'a ve eğitim hayatıma katkı sağlayan tüm Marmara Üniversitesi Elektrik Elektronik Mühendisliği Bölümü öğretim üyelerine teşekkür ederim.

Son olarak ta bu süreçte her zaman yanımda olan kıymetli dostlarıma ve değerli aileme teşekkür ederim.

Mayıs 2022 **Mehmet ŞENSOY**

İÇİNDEKİLER

ÖZET

AKILLI BİNALARDA WEB TEKNOLOJİLERİNİN KULLANIMI

Günümüzde birçok alanda Internet of Things (IoT) kullanımı yaygınlaşmakta ve her geçen gün İnternet'e bağlı cihaz sayısı artmaktadır. IoT uygulamaları yaygınlaştıkça beraberinde bir takım problemleri de getirmektedir. IoT uygulamalarında tasarlanan uygulamaların çoğu yeniden geliştirilebilir yapıda değildir ve birbirinden farklı haberleşme protokolleri kullanmaktadır. IoT teknolojisinin kullanım alanlarından olan akıllı binalarda da bu durum geçerlidir. Akıllı bina sistemlerinde kullanılan ve farklı protokoller üzerinden haberleşen çeşitli cihazlar piyasada bulunmaktadır. Farklı haberleşme protokolleri kullanan cihazların bir arada çalışabilmesi ve sistemlerin yeniden programlama özelliği kazanabilmesi için Web of Things (WoT) konsepti geliştirilmiştir.

Bu tez çalışmasında Web of Things (WoT) konsepti ile IoT tabanlı akıllı bina otomasyonu oluşturmak için .NET Core yazılım çerçevesinden yararlanılmıştır. Bu kapsamda RESTful HTTP API, Worker Service ve kullanıcılar için Web uygulama ara yüzü geliştirilmiştir. Yapılan çalışmada akıllı bina sistemleri için farklı platformlar, sensörler, aktüatörler ve haberleşme protokolleri kullanılmıştır. Her sistem farklı bir protokol üzerinden tasarlanan WoT Gateway ile çift yönlü iletişim kurmaktadır. Hypertext Transfer Protocol üzerinden izleme ve kontrolün yapıldığı uygulamada gerçek zamanlı bildirimler için de WebSocket protokolünden yararlanılmıştır. Veri güvenliği, rol bazlı yetki kontrolü ve veri tabanı işlemleri için .NET kütüphanelerinden yararlanılmıştır. WoT konsepti ile geliştirilen sisteme yeni bir alt sistem eklenmek istendiğinde, önceden eklenen diğer alt sistemlere, protokollere ve yazılım dillerine bağlı olmadan oldukça kolay bir şekilde sisteme entegre edilebilir. Bu çalışmada yer alan modüler otomasyon sistemi pratik olarak uygulanıp test edilmiştir. Yerel ağda Web sunucusuna HTTP GET ile yapılan isteklerde veri uzunluğuna bağlı olarak 35 ms ile 245 ms, HTTP POST ile yapılan isteklerde ortalama 35ms ek gecikme gözlemlenmiştir. Gecikme süreleri kullanıcı deneyimini etkilemeyecek kadar az olduğu için farklı protokollerin kullanıldığı çeşitli senaryolar için yeterlidir.

Anahtar Kelimeler: Akıllı Binalar, Web of Things, Internet of Things, REST API, WebSocket, .NET Core

ABSTRACT

USING WEB TECHNOLOGIES IN SMART BUILDINGS

Nowadays, the use of Internet of Things (IoT) is spreading in many fields and the number of devices connected to the Internet is increasing day by day. When IoT applications become extensive, they also carry some problems. Most of the designed IoT applications don't have redevelopment features and use different protocols. The case is the similar for smart buildings, which are among the IoT application fields. Many different devices working with diverse protocols in the market are used for smart building systems. Web of Things (WoT) concept developed so that devices using different communication protocols could work in common and systems can obtain reprogramming feature.

In this thesis, the .NET Core software framework was used to perform IoT-based smart building automation with the Web of Things (WoT) concept. In this context, RESTful HTTP API, Worker Service and a Web application for users were developed. In this study, various platforms, sensors, actuators and communication protocols were used for hardware where smart building subsystems are represented. Each system communicates bidirectionally with the WoT Gateway designed over a different protocol. WebSocket protocol was also used for real-time notifications in this study, where monitoring and control is carried out over the Hypertext Transfer Protocol. .NET libraries were used for role based authorization control, database operations and data security, When a new subsystem is wanted to be added to the system enhanced with the WoT concept, it can be easily added into the system without being addicted on other previously added subsystems, protocols and software languages. Modular automation system designed in this study applied practically and tested. Depending on the data length, an additional delay of 35 ms to 245 ms was observed for requests made with HTTP GET to the Web server in the local network, and an average of 35 ms for requests made with HTTP POST. It is convenient for scenarios where various communication protocols are used, as the delayed times are so low that they do not affect the user experience.

Keywords: Smart Buildings, Web of Things, Internet of Things, REST API, WebSocket, .NET Core

KISALTMALAR

WoT	**:**	Web of Things
IoT	**:**	Internet of Things
WWW	**:**	World Wide Web
W3C	**:**	World Wide Web Consortium
TD	**:**	Thing Description
JSON	**:**	JavaScript Object Notation
XML	**:**	Extensible Markup Language
JWT	**:**	JSON Web Token
HTTP	**:**	Hyper-Text Transfer Protocol
HTTPS	**:**	HTTP Secure
SMTP		Simple Mail Transfer Protocol
SSL	**:**	Secure Sockets Layer
REST	**:**	Representational State Transfer
OSI	**:**	Open Systems Interconnection
IP	**:**	Internet Protocol
TCP	**:**	Transmission Control Protocol
HVAC	**:**	Heating, Ventilating and Air Conditioning
MQTT	**:**	Message Queuing Telemetry Transport
PAN	**:**	Personal Area Network
LAN	**:**	Local Area Network
MAN	**:**	Metropolitan Area Network
WAN	**:**	Wide Area Network
RTOS	**:**	Real Time Operating System
BMS	**:**	Building Management Systems
LED	**:**	Light Emitting Diode
IC	**:**	Integrated Circuit
URI	**:**	Uniform Resource Identifier
URL	**:**	Uniform Resource Locator
URN	**:**	Uniform Resource Name
REST	**:**	Representational State Transfer
Kb	**:**	Kilobayt
ms	**:**	Milisaniye

ŞEKİL

TABLO LİSTESİ

1. GİRİŞ

Akıllı binalar, geleneksel binalara karşın enerji tasarrufu, kullanıcı konforu ve güvenliği sağlayabilmektedir. Bu özelliklerin sağlanabilmesi için binalarda aydınlatma, güvenlik ya da iklimlendirme gibi alanlarda çeşitli IoT uygulamaları tasarlanmaktadır. IoT cihazlarının birbirleriyle haberleşmesi için Bluetooth, Wi-Fi, RFID vb. iletişim protokolleri kullanılmaktadır [1]. Bununla birlikte akıllı bina otomasyonları için ticari standartlar ve protokoller dünya genelinde oldukça yaygın olarak tercih edilmektedir. Günümüzde cihaz üreticileri ticari çıkarları için genellikle farklı üreticilerin cihazları ile uyumlu olmayan, kendi iletişim protokollerini ve platformlarını kullanan cihazlar üretmektedir [2]. Bununla beraber üreticiler, kasıtlı olarak ürün entegrasyonlarını daha da zorlaştırarak farklı cihazlar arası haberleşmeyi engelleyecek şekilde programlayabilir [3]. Farklı standart ve protokolleri kullanan cihazların yaygınlaşması ile birlikte sistemlerin uyum içerisinde çalışması zor hale gelebilmektedir [4]. Bu gibi nedenlerden dolayı akıllı binaların önündeki en önemli problemlerden biri iletişim protokolü standardının olmamasıdır [5]. Akıllı binalarda bulunan sistemler arasında uyum olmadığında cihazlar arası entegre işlemleri ciddi çaba ve ileri seviye programlama becerileri gerektirebileceği için oldukça maliyetli ve daha karmaşık hale gelebilir [6][7]. Ayrıca kullanılan cihazların üreticileri farklı olduğunda tek bir uygulama ile kontrol ve bilgilendirme yapılabilmesi de oldukça zorlaşır. Binada bulunan sistemlerin birbirleriyle tamamen uyumlu olması durumunda tüm işlemler sadece tek bir uygulama üzerinden kontrol edilebilir. Böylelikle hem kullanıcı memnuniyeti artar hem de daha verimli bir kontrol sağlanmış olur.

Akıllı ortamların oluşmasını sağlayan IoT uygulamaları gün geçtikçe yaygınlaşıyor ve 2026 yılına kadar İnternet'e bağlı cihaz sayısının 50 milyara ulaşması beklenmektedir [8]. IoT'nin temel amacı, farklı hizmetler sunmak için ağ katmanında cihazlar arası iletişimi sağlamaktır [9] [10]. Bununla birlikte IoT teknolojisinin kullanıldığı ortamlarda otomatik kontrol sistemleri de geliştirilmiştir [11]. Ancak bu teknolojinin yaygınlaşması ile cihaz heterojenliği, düşük performans, verilerin güvenilirliği ve kalitesi ile ilgili bir takım zorluklar da beraberinde gelmektedir [12]. Elektrik ve Elektronik Mühendisleri Enstitüsü'ne (IEEE) göre birlikte çalışabilirlik; bir sistemin diğer sistemler veya cihazlarla basit bir şekilde etkileşim kurma yeteneğini ifade eder [13]. Farklı protokol kullanan cihazların birlikte çalışabilirliğin basit bir şekilde sağlanabilmesi için IoT sistemlerde uygulama katmanı olarak Web'in kullanılabilmesi fikri 2007'de ortaya çıkmaya başladı ve Dominique Guinard ile Vlad Trifa bu konsept üzerinde paralel olarak çalışmaya başladı. Daha sonra Guinard ve Trifa tarafından Web of Things (WoT) standartı kavramsal olarak 2009'da önerilmiştir [14]. Bu çalışmalar, WoT, IoT hizmetlerini Web kaynakları olarak kullanılabileceğini göstermiştir [15]. İnterneti kullanan ve her yerden erişilebilirlik sağlayan Web, bu sorunları çözmek için iyi bir çözümdür [16][17]. WoT, gömülü cihazlar arası haberleşmeyi sağlamak için Web teknolojilerinden yararlanmaktadır [18] [19].

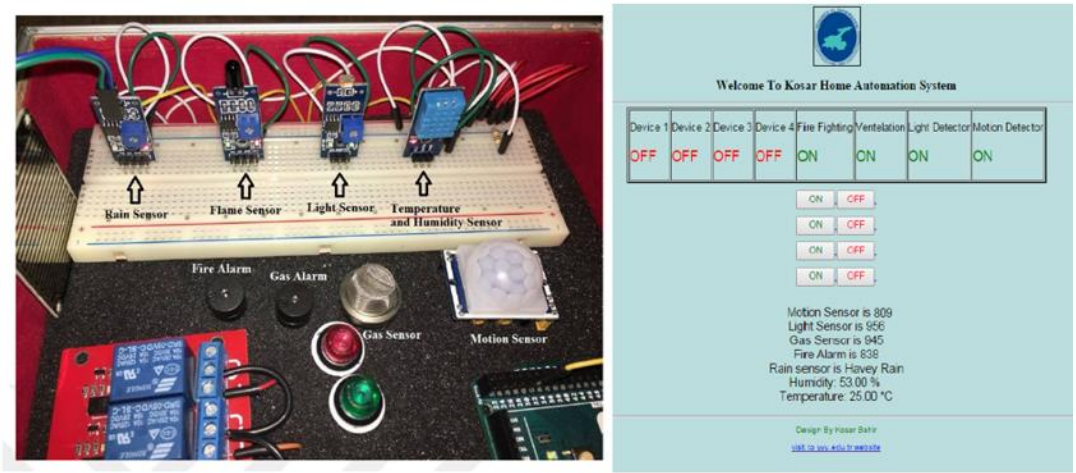
1.1 Literatürdeki İlgili Çalışmalar

Literatürde, IoT tabanlı akıllı bina otomasyon sistemleri ile ilişkin birçok çalışma bulunmaktadır. Ancak geleneksel IoT yöntemleri ve tescilli protokoller ile geliştirilen uygulamalarda yeniden geliştirme yapmak veya üreticileri farklı cihazları, sisteme entegre etmek yüksek çaba ve zaman gerektirebilmektedir. Bu sorunlara çözüm olarak sunulabilecek WoT konsepti ile ilgili de farklı alanlarda çeşitli çalışmalar bulunmaktadır.

Çalışma [20]'de cihaz satıcılarına olan bağımlılığı ortadan kaldırmak için IoT çözümlerinden yararlanılmıştır. Bu çalışmada bir sunucu, binada içerisinde yer alan elektronik sistemler ile bulut arasına eklenmiştir. Yapılan otomasyon sisteminde klasik bina otomasyonlarında bulunan PLC'ler yerine daha küçük boyutlara sahip Raspberry Pi türevi bir bilgisayar kullanılmıştır. Linux işletim sisteminde çalışan bu bilgisayar üzerinde yüksek seviyeli dil olan Java program dili ile geliştirme yapılmıştır. Binada tüketilen enerjinin değerlendirilmesi için sunucuda toplanan veriler bulut ortamına iletilmiştir. Sistem bir yere kadar farklı protokoller ile birlikte çalışabilir ancak sistem içinde bulunan cihazların detaylı açıklaması yer almadığı için tekrar programlamada zorluk çekilebilir. Ayrıca bu sistemde kullanıcılar için gerçek zamanlı bilgilendirme bulunmamaktadır. Şekil 1.1'de bu çalışmada test edilen prototip gösterilmiştir.

Şekil 1.1 Prototip [20]

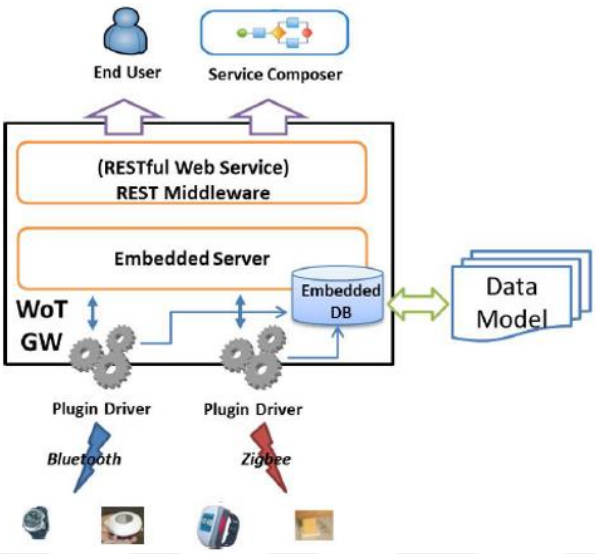
Çalışma [21]'de akıllı ev otomasyon sistemi için oluşturulan IoT uygulaması bir Web tarayıcısı yardımıyla görüntülenmekte ve kontrolleri gerçekleştirilmektedir. Çeşitli sensörlerin kullanıldığı sistemde otomatik aydınlatma, ortam sıcaklık ölçümü, gaz kaçağı ve yangın alarm sistemi yer almaktadır. Yapılan sistem, ekonomik açıdan oldukça uygundur ancak sisteme farklı protokol kullanan cihazları kolay bir şekilde destekleyemeyecektir. Şekil 1.2'de çalışma kapsamında hazırlanan donanım ve Web sitesi gösterilmektedir.



A **B**

Şekil 1.2 A - Sistem donanımı, B - Web sitesi [21]

Makale [22]'de akıllı cihazlar için WoT konseptinde iki katmanlı entegrasyon çerçevesi önerilmiştir. Cihaz heterojenliği olan herhangi bir sistemde tek bir Web uygulaması ile tüm cihazların görüntülenmesi ve kontrolü hedeflenmiştir. Akıllı cihazların birbirleriyle entegre etmek için RESTful Web servislerinden yararlanılmıştır. İnternete direk erişme durumu olmayan cihazlar öncelikle WoT Gateway ile haberleşmektedir. HTTP metotlarının kullanıldığı sistemde gerçek zamanlı bildirimler yer almamaktadır. Şekil 1.3'te oluşturulan WoT ağ geçidi yapısı gösterilmiştir.



Şekil 1.3 RESTFul WoT Gateway [22]

Makale [23]'te WoT yardımıyla geliştirilen akıllı şebeke sistemi sunulmaktadır. Akıllı şebeke mimarisinin amacı, güneş enerjisinden en iyi derecede yararlanarak tüketicilere temiz enerji kaynağı sağlamaktır. Linux işletim sisteminde çalışan bir Raspberry Pi, REST API sağlayan bir sunucu olarak görev yapmaktadır. Enerji sayaçları ölçüm verilerini PIC16F877A mikro denetleyicisine aktarmaktadır. Daha sonra bu bilgiler ZigBee protokolü ile Raspberry Pi'ya iletilir. Kullanıcılar enerji tüketim grafiklerinin sunulduğu Web sitesine bakarak, enerji kaynaklarını nasıl ve ne zaman kullanacağını önceden planlar. Şekil 1.4'te uygulamanın Web sitesi gösterilmiştir.

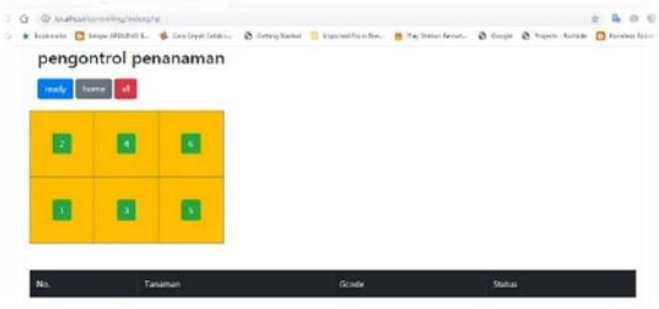
Şekil 1.4 Web sitesi [23]

Makale [24]'te otonom hava kalite yönetim sistemi WoT Gateway ile tasarlanmıştır. Ortamda hava kirliliği belirlenen seviyenin üstüne çıktığında ESP8266 tarafından WoT Gateway'e ilgili bildirim mesajı gönderilir. Hava kirliliği risk içerecek eşiğe ulaşır ise ESP8266 tarafından havalandırma cihazlarına havanın temizlenmesi için otomatik sinyaller iletilir. WoT Gateway, cihazların komutlarını ve değerlerini okumak ve yazmak için KNX veri yolları ile etkileşime girmektedir. Şekil 1.5'te hava kalite sensörüne ait Nesne Açıklaması (Bölüm 2'de detaylı değinilecektir) gösterilmiştir.


```
"security": {
    "authorizationUrl": "/auth",
    "scheme": "bearer",
    "format": "JWT"
},
"access": "http://192.168.1.130/aqsensor",
"@type": [
    "Thing",
    "Air quality sensor bello"
],
"name": "AQ_Sensor",
"href": "/things/5c41f3134dfc992fa097f62c",
"@context": [
    "http://w3c.github.io/wot/w3c-wot-td-context.jsonld",
    "http://w3c.github.io/wot/w3c-wot-common-context.jsonld",
    "http://iot.schema.org"
],
"device": "external",
"properties": {
    "String": {
        "link": [
            {
                "mediaType": "application/json",
                "href": "/things/5c41f3134dfc992fa097f62c/properties/sta
            }
        ],
        "outputType": {
            "type": "String"
        },
        "inputType": {
            "type": "String"
        },
        "@type": [
            "Property",
            "String"
        ],
        "writable": true
    }
}
```


Şekil 1.5 Nesne açıklaması [24]

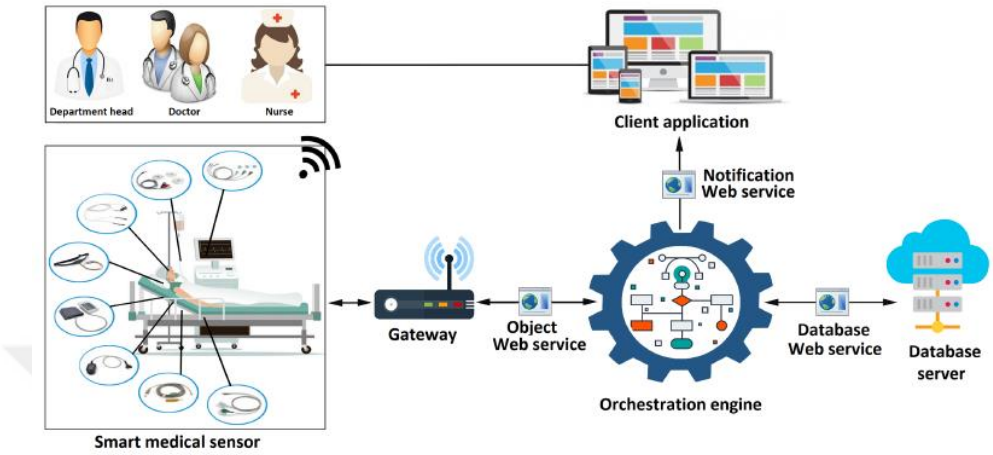
Makale [25]'te akıllı çiftlik için uzaktan tohum ekimi, sulama ve toplama işlemlerinin yapılabilmesinde WoT yapısından faydalanılmıştır. Raspberry Pi, UART Seri Haberleşme aracılığı ile kullanıcının girdiği koordinatları Arduino'ya Gcode dosyası olarak gönderir. Seri haberleşmeden sonra Arduino bu Gcode dosyasını çalıştırarak tohumları istenilen yerlere hazırlanan düzenek ile ekmektedir. Şekil 1.6'da sisteme ait görseller gösterilmiştir.



A B

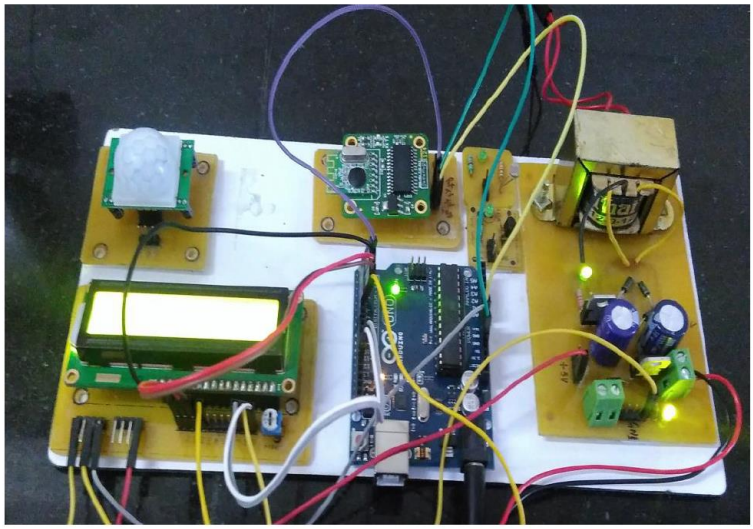
Şekil 1.6 A - Prototip, B - Web sitesi [25]

Çalışma [26]'da hastanede uzaktan hasta sağlığının izlenmesi ve sağlık çalışanlarının görevlerinin hafifletilmesi için SOA mimarisiyle WoT kullanılmıştır. Hastaya gerekli bazı tedavilerin uzaktan kontrol ile sağlanabildiği sistemde, acil durumlarda sağlık personelleri anlık olarak uyarılabilmektedir. Anlık uyarı bildirimler için WebSocket protokolünden yararlanılmıştır. Şekil 1.7'de önerilen sisteme ait şemaya yer verilmiştir.



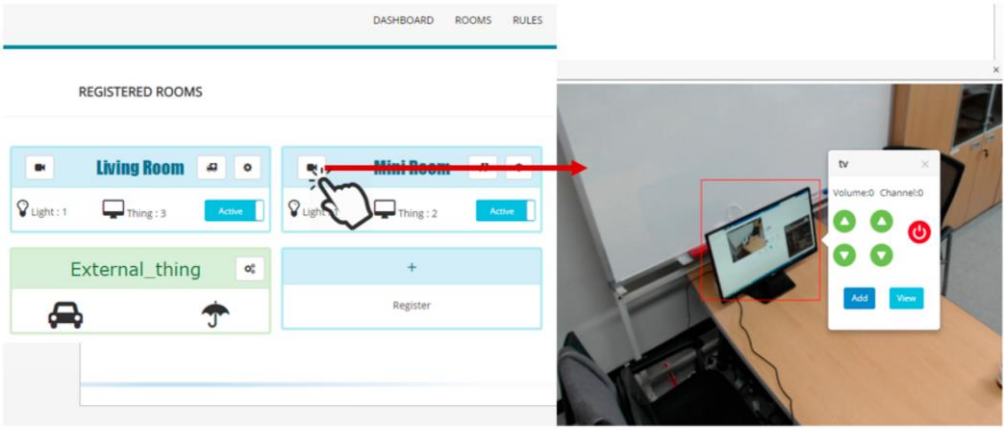
Şekil 1.7 E-sağlık platform şeması [26]

Çalışma [27]'de Arduino kartına bağlı çeşitli sensörlerin bağlandığı IoT tabanlı bina otomasyon sistemi önerilmiştir. Ortam sıcaklığının ve ışık şiddeti seviyesinin ölçüldüğü sistemde ışıkların açılıp kapatılması ve klima kontrolü otonom olarak tasarlanmıştır. Ayrıca gaz kaçağı tespit edildiğinde sistem alarm ile uyarı vermektedir. Ancak bu sistem klasik IoT uygulamalarına örnektir ve farklı protokollerin kullanıldığı uygulamalarda yetersiz kalacaktır. Şekil 1.8'de sunulan sistemin prototipi gösterilmiştir.



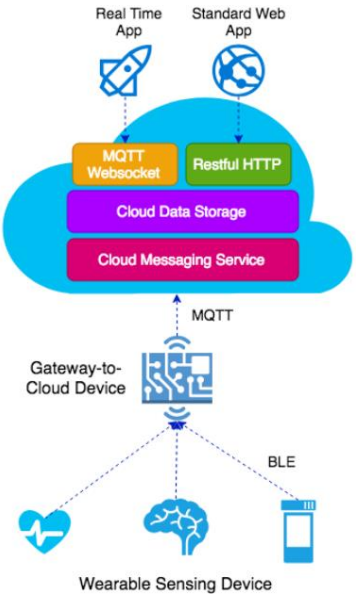
Şekil 1.8 Prototip [27]

Makale [28]'de Web tabanlı bir IoT sistemi, ev cihazlarının kontrolü için tasarlanmıştır. Derin nesne algılama tekniğinden yararlanan sistemde gerçek zamanlı video akışları yardımıyla seçilen cihazların kontrolü yapılabilmektedir. Şekil 1.9'da tasarlanan sisteme ait görsel yer almaktadır.



Şekil 1.9 Video akışı ile kontrol uygulaması [28]

Makale [29]'da giyilebilir cihaz verilerinin sağlık hizmetlerinde kullanılabilmesi için Web tabanlı yazılım uygulaması geliştirilmiştir. Giyilebilir cihazlardan toplanan veriler, Bluetooth Low Energy (BLE) protokolü aracılığıyla bir ağ geçidine gönderilir. Gönderilen veriler ağ geçidinden Message Queuing Telemetri Transport (MQTT) protokolü ile veri tabanına aktarılır. Şekil 1.10'da sistemin genel mimarisi gösterilmiştir.



Şekil 1.10 Sistem mimarisi [29]

Makale [30]'da akıllı sistemleri Web üzerinden kontrol etmek için hafif bir platform önermektedir. Önerilen sistemde HTTP protokolü yanında eş zamanlı iletişim için XMLHttpRequest ve sistemlerin Web ile etkileşimi için de EventSource JavaScript nesneleri kullanılmıştır.
Dünya genelinde bazı büyük teknoloji firmaları ve üniversiteler tarafından WoT konsepti üzerine araştırmalar ve geliştirmeler devam etmektedir.

1.2 Tezin Katkısı

Bu tez çalışmasının hedefi, açık kaynak kodlu gelişmiş yazılım çerçevesi olan .NET Core ile oluşturulan REST API ve Worker Service arka plan uygulamalarını birlikte kullanarak akıllı binalar ve benzer senaryolar için IoT tabanlı bir çözüm ortaya koymaktır.

Bu tezin başlıca katkıları şunlardır:

- Akıllı binalarda farklı protokol kullanan cihazların bir arada çalışabilirliğini sağlamak için yeni bir IoT çözümü olan WoT yapısını, açık kaynaklı ve yüksek performansa sahip .NET Core ile uygulamak
- Uygulama içerisinde önceden kullanılan protokollere, donanımlara bakılmaksızın yeniden geliştirilebilir IoT uygulamasını sunmak
- Hazırlanan donanımda yer alan sensör ve aktüatörlerin tüm etkileşimlerinin RESTful API ile servis etmek
- Gerçek zamanlı, olay tabanlı etkileşimler için WebSocket protokolünden yararlanmak
- Veri tabanı işlemleri ve belirli görevler için Worker Service yazılımından faydalanmak
- Rol bazlı cihaz kontrolü ve veri güvenliği için gelişmiş yazılım kütüphanelerini kullanmak

1.3 Genel Bakış

Bu tez yapısı şu şekilde düzenlenmiştir: Bölüm 2, tez çalışmasında sunulan kavramları anlamak için gerekli olan genel bilgilerini sunar. Bölüm 3, oluşturulan proje için geliştirilen yazılımları ve pratik uygulamayı açıklamaktadır. Bölüm 4, uygulama için yapılan test sonuçlarını sunmaktadır. Son olarak Bölüm 5'te tezin sonuç kısmı yer almaktadır.

2. GENEL BİLGİLER

Bu bölümde, yapılan tez projesinin daha iyi anlaşılabilmesi için bilgiler sunulmuştur. İlk olarak Bölüm 2.1'de akıllı binalar genel hatları ile ele alınmaktadır. Bölüm 2.2'de IoT, Bölüm 2.3'de WoT hakkında bilgiler verildikten sonra Bölüm 2.4'de IoT ve WoT karşılaştırılması yapılmaktadır.

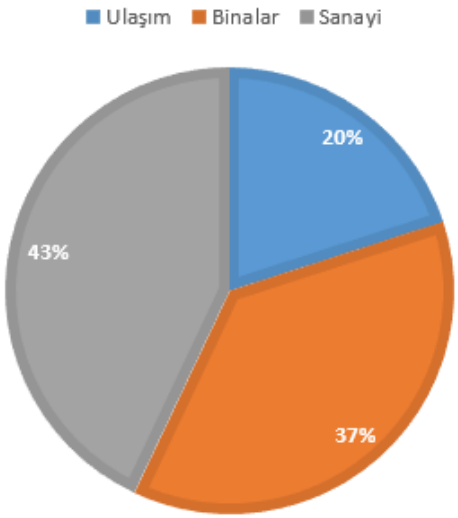
2.1 Akıllı Binalar

Akıllı binalar, aydınlatma, güvenlik, enerji yönetimi ve iklimlendirme (HVAC) gibi belirli işlevleri gerçekleştirmek için bilgisayar teknolojileri ile yönetilen binalardır [31][32]. Akıllı bina kavramı ilk olarak ticari amaçla ortaya atılmıştır ve ilk akıllı bina Amerika'da 1981 yılında inşasına başlanılan City Place'dır. Basit otomasyon sistemlerinin bulunduğu binada, iletişim için yerel alan ağından (LAN) yararlanılmıştır. Amerika'dan sonra Japonya, İngiltere ve Çin'de akıllı binalar inşa edilmeye başlanmıştır. Tablo 2.1'de bina tipleri ve genel özellikleri verilmiştir.

Tablo 2.1 Bina tipleri ve özellikleri

Bina Tipi	**Kontrol**	**Kullanılan Sistemler**	**Verimlilik ve Kullanıcı Etkileşimi**
Geleneksel (basit) bina	Manuel	Manuel olarak kontrol edilebilir sistemler	Zorunlu etkileşim ve düşük verimlilik
Otomasyonlu bina	Geri bildirimli	Otomasyon sistemleri	Otomatik kontrol, yüksek verim
Akıllı Bina	Etkileşimli	Entegre edilebilir sistemleri	Kullanıcı için daha fazla kontrol ve yüksek verim

Hem ülkemizde hem de dünyada enerjinin büyük bir bölümü bina içerisindeki aktiviteler sonucu harcanmaktadır. Bu oran ülkemizde %40'a yaklaşmaktayken dünyada ise %50'e kadar artabilmektedir [33]. Şekil 2.1'de Türkiye'deki tüketilen enerji dağılımı gösterilmektedir.



Şekil 2.1 Türkiye'de tüketilen enerji dağılımı

Geleneksel binalara kıyasla akıllı binalarda %30 ile %50 arasında enerji tasarrufu sağlanabilir [34]. Tablo 2.2'de çeşitli binalarda akıllı sistemlerin kullanılması ile yapılacak tahmini enerji tasarrufu gösterilmektedir.

Tablo 2.2 Bina tiplerinde tahmini enerji tasarrufu

Bina Tipi	**Teknoloji**	**Tahmini Tasarruf**
Ofis	Aydınlatma sistemleri, HVAC sistemleri	% 23
Hastane	Aydınlatma sistemleri, veri analitiği yazılım paketleri	% 18
Otel	Misafir odası doluluk kontrolleri	% 6
Okul	Doluluk sensörleri, Web tabanlı aydınlatma kontrol yönetim sistemi	% 11

Birçok araştırmanın hedefi konumunda olan akıllı binalar, sürdürülebilirliği artırarak yaşamlarımızı önemli ölçüde iyileştirmeyi vaat etmektedir [35]. Avrupa Komisyonu 2030 yılına kadar enerji israfını %32.5 oranında azaltmayı planlamaktadır ve IoT tabanlı akıllı binalarda yaklaşık %23 oranında bir enerji tasarrufu söz konusudur [36]. IoT teknolojisiyle tasarlanan akıllı sistemler ile binalar daha güvenli, verimli ve konforlu yaşam ortamlarına dönüşmüştür [37]. Kullanıcıların istek ve ihtiyaçları doğrultusunda gelişen akıllı binalarda sağlanması gereken nitelikler:

- Enerji tasarrufu
- Can ve mal güvenliğini
- Çevre dostu
- Konfor

2.1.1 Bina Otomasyon Sistemleri (Building Management Systems - BMS)

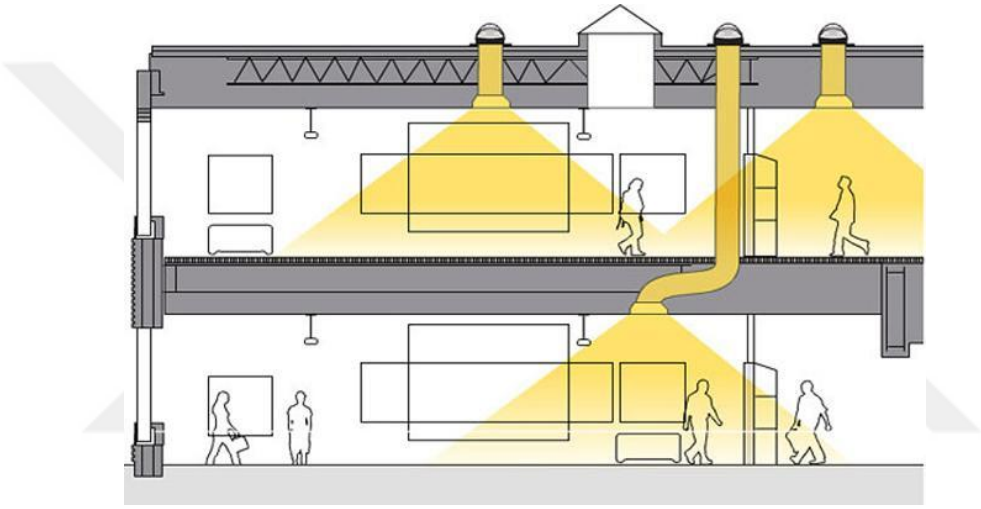
Bina otomasyon sistemleri, bilgisayarlar yardımıyla bina içerisinde ısıtma, soğutma, aydınlatma, alarm uyarısı verme gibi faaliyetleri gerçekleştirmektedir. Bina otomasyon sistemleri bina içerisinde bulunan alt sistemleri tek bir noktadan denetlemek ve kontrol etmek üzere tasarlanmaktadır. Bina otomasyon sistemlerinde genellikle verilerin işlenmesi için bir sunucu, ortamdan verileri alabilmek için sensörler, istenen kontroller için ilgili aktüatörler ve çeşitli cihazlar bulunabilmektedir. Bir bina otomasyon sisteminden beklenen özellikler şunlardır:

- Bina içerisinde yer alan tüm sistemlerin koordineli olarak çalışması
- Olası can ve ya mal güvenliğini tehdit eden durumlarda kullanıcıların uyarılması
- Enerji tüketimi veri analizi ve raporlaması
- İklimlendirme sistemlerinin otomatik kontrolü
- Aydınlatmada enerji verimliliğini sağlanması

Akıllı binalarda; aydınlatma, güvenlik, iklimlendirme, acil durumlarda anons ve enerji yönetimi gibi alanlarda akıllı sistemler tasarlanabilir [38]. Tasarlanan sistemler ile enerji giderleri azaltılabilir, kullanıcılar için konfor ve güvenlik ihtiyaçları karşılanabilir.

2.1.2 Aydınlatma Sistemleri

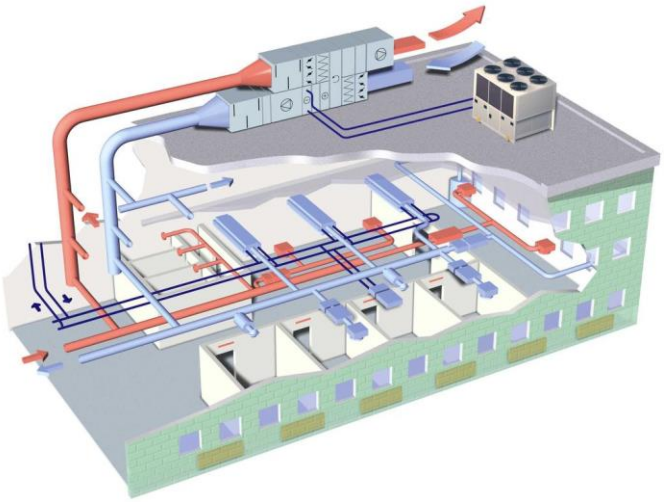
Akıllı binalarda yer alan aydınlatma sistemleri, kullanıcıların konfor düzeyini iyileştirerek enerji tasarrufu sağlayan sistemlerdir. Bu sistemler ile hem yapay aydınlatma hem de ışık hasadı gibi doğal aydınlatma ile iç ortam ışık seviyeleri düzenlenebilir. Binalarda toplam elektrik enerjisinin %20-%25 gibi büyük bir kısmı aydınlatmada kullanılmaktadır [39]. Aydınlatma sistemlerinde kullanılacak sensörler ve LED armatürler enerjinin daha verimli bir şekilde tüketilmesini sağlayabilir. Akıllı bir aydınlatma sisteminde kullanıcılar uzaktan kontrol ile aydınlatma cihazlarını kontrol edebilir, lambaları açıp kapatabilir, LED lambaların ışık seviyelerini değiştirebilir, belirli saatlere göre otomatik olarak sistemi çalıştırabilir ve iç ortam parlaklığını sabit bir seviyede tutmak isteyebilir. Kullanılan enerjinin azaltılması içinde yapay aydınlatmanın yanında gün ışığı hasadından da yararlanılabilir. Bunun için kullanılan ışık boruları çok yüksek düzeyde yansıma sağlayabilmektedir. Şekil 2.2'de ışık boruları ile aydınlatma gösterilmiştir.



Şekil 2.2 Işık boruları ile aydınlatma [40]

2.1.3 HVAC Sistemleri

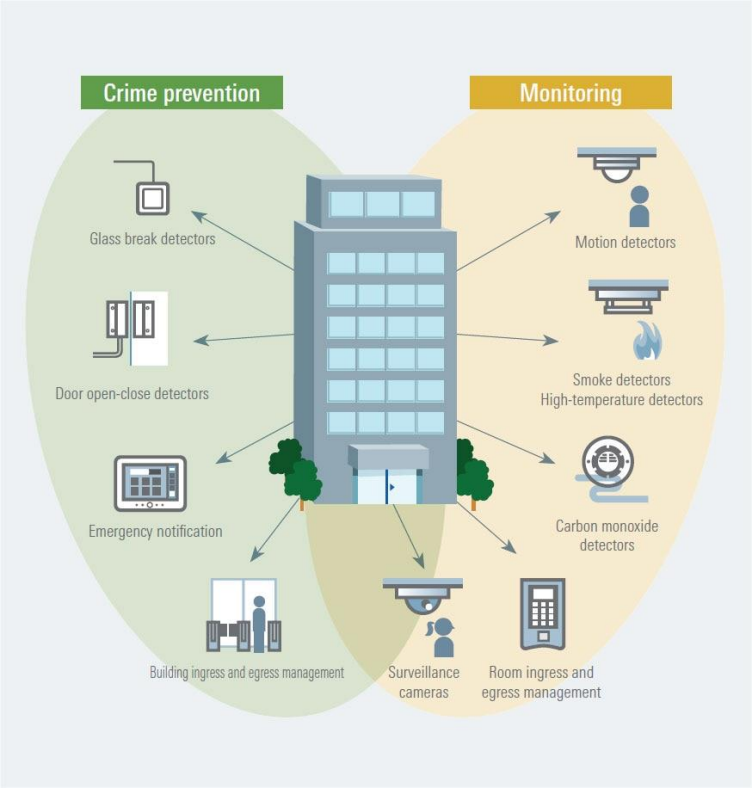
Binalarda ısıtma, havalandırma ve soğutma gibi iklimlendirme yöntemleri oldukça sık kullanılmaktadır. HVAC sistemi kontrollerin doğru bir şekilde yapılması enerji tasarrufunun önemli bir parçasını oluşturabilmektedir. HVAC sistemleri ile bina içi hava kalitesi, sıcaklık ve nem kontrolleri sağlanmaktadır. Isıtıcılar, klimalar, soğutma ekipmanları gibi birçok iklimlendirme cihazı bu sistemlerde kullanılmaktadır. Bu sistemlerde uzaktan kontrol ve ortam sıcaklığının istenilen parametrelerde tutulması beklenen özelliklerdendir. Hali hazırda bina iklimlendirme işlemleri için çeşitli otomasyon sistemleri piyasada kullanılmaktadır. Şekil 2.3'te binada HVAC modellemesi gösterilmiştir.



Şekil 2.3 HVAC modellemesi [41]

2.1.4 Güvenlik Sistemleri

Binalarda çeşitli güvenlik sistemleri ile can ve mal varlığının korunabilmesi için bir takım önlemler alınmaktadır. Olası bir yangın durumunda binada bulunanların acil olarak uyarılması, izinsiz girişlerin ve hırsızlığın önlemesi, tehlikeli gaz kaçağının tespit edilmesi gibi kritik önlemlerin alınması, akıllı binalardan beklenen özellikler arasındadır. Şekil 2.4'te bina için alınabilecek güvenlik önlemleri gösterilmiştir.



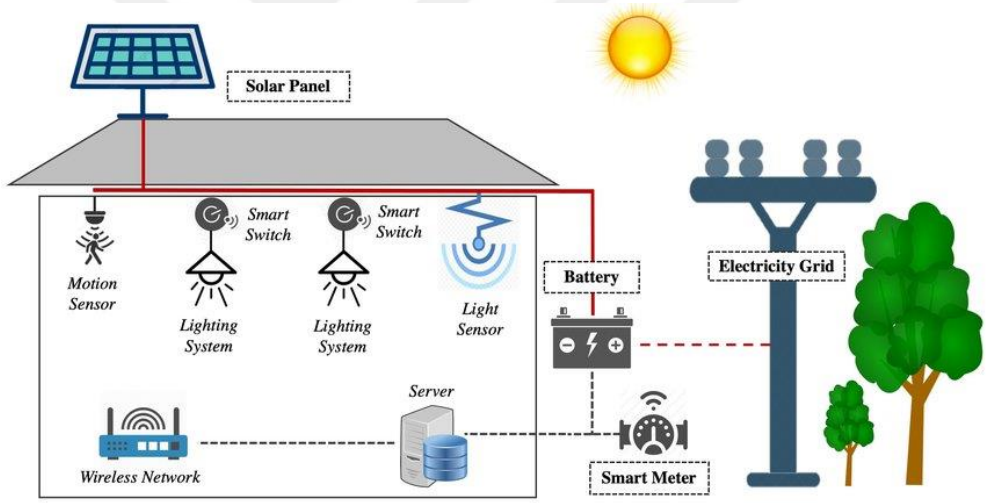
Şekil 2.4 Bina için güvenlik sistemleri [42]

Can ve mal güvenliği için güvenlik sistemlerinden beklenen özellikler şunlardır:

- Kamera güvenlik sistemlerinin anlık izlenebilmesi
- Yangın gibi tehlike anlarında binada bulunanların sesli anons ile uyarılması
- Kartlı geçiş sistemlerinin bulunması
- Hırsızlara karşı alarm ve ihbar sisteminin olması
- Sunucular için siber güvenlik önlemlerinin alınması

2.1.5 Enerji Yönetim Sistemleri

Binalarda enerji yönetimi ve bilgi sistemleri için çok çeşitli donanım ve yazılım sistemleri bulunmaktadır. Bina içi kullanılan enerji miktarının belirlenmesi, iklimlendirme, güvenlik ve aydınlatma sistemlerinin verimliliği tespiti ve tespit edilen sonuçlara göre performansın iyileştirilmesi bina enerji yönetim sistemleri kapsamına girmektedir. Enerji tüketiminin belirli aralıklar ile raporlanıp otomatik olarak e-posta ya da mobil yollarla kullanıcılara bilgilendirme yapılması akıllı binalarda kullanıcı memnuniyetini arttıran niteliklerdendir. Çalışma [43]'te geliştirilen IoT tabanlı güneş enerjisi tüketimi kontrol mekanizması bu sistemlere örnek gösterilebilir. Şekil 2.5'te önerilen sistem modeli gösterilmiştir.



Şekil 2.5 IoT tabanlı bina enerji yönetim sistemi modeli [43]

2.1.6 Akıllı Binalarda İletişim Protokolleri

Bina içerisinde yer alan sistemlerin çalışabilmesi için özel protokoller ve açık protokoller bulunmaktadır. Özel protokoller isminden de anlaşılacağı üzere belirli üreticiler tarafından ticari amaçlı geliştirilen protokollerdir. Açık protokoller ise kuruluşlar tarafından geliştirilen protokollerdir. Açık kaynaklı protokoller diğer protokol ve standartları desteklemeye yönelik tasarlanmaktadır. BACnet, LonWorks, KNX ve Modbus açık kaynaklı ve kablolu iletişimle çalışan protokollere örnek olarak gösterilebilir. Kablosuz açık protokollere ise EnOcean, LR-WPAN protokolleri örnek verilebilir.

2.1.6.1 BACnet

Amerikan kökenli açık protokol olan BACnet, aydınlatma, güvenlik, HVAC sistemlerinin yönetimi için geliştirilmiştir. Amerikan Ulusal Standartlar Enstitüsü bu protokolü 1995'te ulusal bir protokol olarak onaylamıştır.

2.1.6.2 LonWorks

BACnet gibi bina otomasyonlarında yaygın kullanılan LonWorks protokolünü, Amerika kökenli Echelon firması geliştirmiştir. LonWorks protokolü ile aydınlatma, güvenlik ve HVAC sistemleri tasarlanabilmektedir. Ayrıca LonWorks akıllı binalar haricinde farklı alanlarda da kullanılmaktadır.

2.1.6.3 KNX

BACnet ve LonWorks protokollerinin aksine KNX, Amerika değil Avrupa kökenli bir protokoldür. Aralarında Bosch, Siemens, Schneider gibi büyük firmaların bulunduğu dokuz şirket tarafından Avrupa iletişim standartlarına uygun bir açık protokol olarak 1999'da geliştirilmiştir.

2.1.6.4 Modbus

PLC sistemleri öncü firmalarından olan Modicon tarafından 1978'de geliştirilmiştir. Modbus, akıllı binaların varlığından önce endüstriyel alanlarda iletişim protokolü olarak kullanılmaktaydı. Basit bir yapıya sahip olsa da geniş alanlarda kullanıma oldukça uygundur.

2.1.6.5 EnOcean

EnOcean teknolojisi, bina otomasyon sistemlerinin yanında endüstri, ulaşım, lojistik gibi alanlarda da kullanılmaktadır. EnOcean, enerji hasadı ile uyumlu bir kablosuz teknolojidir. Düşük enerji tüketiminde yüksek veri hızı sağlar. Aynı anda birden fazla verici cihazla ilgilenir.

Görüldüğü üzere akıllı binalarda çeşitli protokoller bulunmakla birlikte tek bir standart bulunmamaktadır. Özel protokollerinde devreye girmesi ile heterojen cihaz ortamı oluşmaktadır. Örneğin BACnet protokolü kullanılan sistemde, özel protokol ile çalışan başka alt sistem entegre edilmek istendiğinde cihazlar arasında veri aktarımı zorlaşabilir [3]. Bu sorunu çözmek için geleneksel bina otomasyon sistemlerine paralel olarak IoT tabanlı otomasyonlar geliştirilmektedir.

2.1.7 IoT ile Bina Otomasyonu

IoT tabanlı bir akıllı binada sensörler aracılığı ile toplanan veriler öncelikle ağ geçidine daha sonra bulut ortamına iletilmektedir. Böylelikle İnternete erişme yeteneği bulunmayan cihazlar ağ geçidi üzerinden verileri alınması sağlanabilir. Ağ geçidi ile yeni teknolojiler ve protokoller kademeli olarak sisteme entegre edilebilir. Daha sonra kullanıcılar Web tarayıcılardan ya da mobil cihazlarından bilgi ve bildirim alabilmektedir.

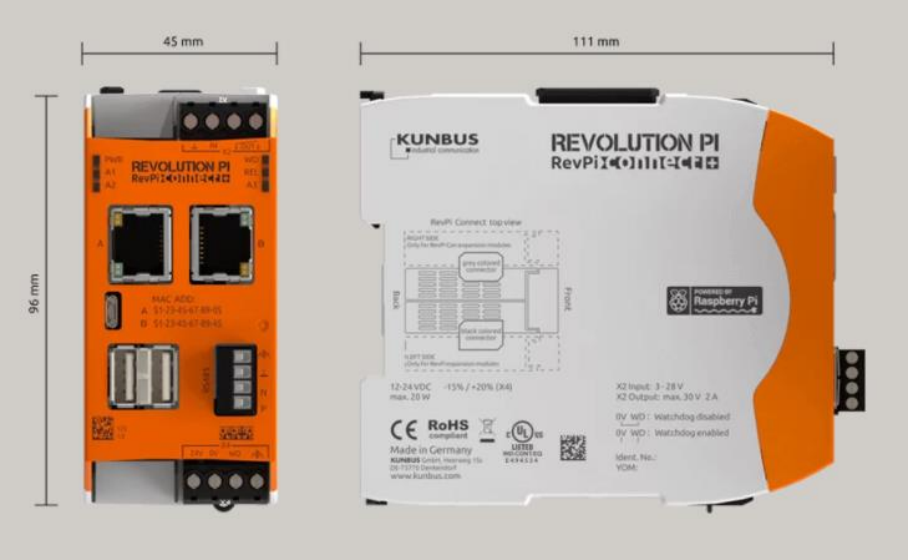
IoT tabanlı çeşitli sistemler ile binada enerji verimliliği sağlanabilir. IoT ile aydınlatma, güvenlik, HVAC, enerji yönetimi gibi bina alt sistemleri tasarlanabilir. Şekil 2.6'da akıllı binalar için IoT tabanlı oluşturulabilecek çeşitli sistemler resmedilmiştir.

Şekil 2.6 Akıllı binalarda IoT

Mevcut bina otomasyon sistemleri ile karşılaştırıldığında IoT entegrasyonu ile sağlanabilecek özellikler:

- Bina otomasyon sistemlerinin sağladığı benzer hizmetler
- Sadece bulunduğu bina değil diğer binalar ile etkileşime girme
- Gelecek uygulamalar için cihaz entegre potansiyeli
- Yapay zeka, makine öğrenmesi gibi teknolojiler için alt yapı sunabilme
- Yüksek seviyeli yazılım dilleri ile esnek programlama

Akıllı binalarda yer alan elektronik ve mekanik sistemlerde daha akıllı ve uyarlanabilir hale getirebilen çeşitli IoT sensörleri kullanabilir. Bu sensörler ile veriler bina analitiği için kullanılabilir [44]. Piyasada bulunan bina otomasyon sistemlerinde PLC birimleri yaygın olarak kullanılmaktadır. Endüstri 4.0 ile birlikte gelişen çeşitli PC tabanlı IoT platformları ile de PLC deki benzer işlevler gerçekleştirilmektedir. Bu kapsamda geliştirilen bir platform olan Revolution Pi, endüstriyel IoT ağ geçidi olarak kullanılabilmesi için bazı yetenekler kazandırılmış bir Raspberry Pi kartıdır. Farklı yüksek seviyeli diller ile programlanabilen Linux tabanlı bir işletim sisteminde çalışıyor olması Revolution Pi'ya programlanabilme konusunda kolaylık sağlamaktadır. Şekil 2.7'de Revolution Pi gösterilmiştir.



Şekil 2.7 Revolution Pi

2.2 Nesnelerin İnterneti (Internet of Things – IoT)

Nesnelerin İnterneti veya IoT, çeşitli ağ ara yüzleri üzerinden iletişim kuran ve sonunda daha geniş ağlara bağlanabilen elektronik cihazlarla keşfedilebilen, izlenebilen, kontrol edilebilen veya etkileşime girebilen bir fiziksel nesneler sistemidir [45]. Gömülü sistemler, kablosuz sensör ağları, otomasyon alanlarındaki gelişmeler bu alanında gelişmesini sağlamıştır. "Internet of Things" kavramında yer alan "Thing" dijital olarak ifade edilen fiziksel bir nesne ya da sanal bir hizmet olabilir. Ev ve bina otomasyonları, ulaşım, altyapı, enerji yönetimi gibi birçok ticari ve endüstriyel alanda IoT sistemleri yaygın bir biçimde kullanılmaktadır. Günümüzdeki bina otomasyon sistemlerinde IoT cihazları ile aydınlatma, iklimlendirme, güvenlik ve kamera sistemleri kolaylıkla tasarlanabilmektedir [46]. Bina enerji yönetim sistemlerinde IoT ile kamu elektrik tarifesi dikkate alınarak harcanan elektrik üzerinde tasarruf yapılabilmektedir [47]. Önümüzdeki yıllarda birçok ev aletinin IoT protokolleri ile İnternete entegre olacağı öngörülmektedir [48].

Cihazların birbirlerine bilgi aktarmakta kullandıkları lisana "protokol" adı verilmektedir [49]. Cihazların ve sistemlerin birlikte çalışabilmesi için ortak bir iletişim protokolünü kullanmaları gerekir. IoT, OSI modelindeki Katman 1'den Katman 6'ya kadar olan geniş ölçekte çoklu protokol yığınlarını kullanmaktadır.

2.2.1 İnternet

İnternet, elektronik cihazlar ve ağlar arasında iletişim kurmak için İnternet protokol paketini (TCP / IP) kullanan birbirine bağlı bilgisayar ağları sistemidir. İnternetin temelleri 1960'larda ABD Savunma Bakanlığı tarafından atılmıştır. Web uygulamaları, elektronik postalar, dosya paylaşımı gibi çeşitli hizmetler İnternet aracılığı ile çalışmaktadır.

2.2.2 OSI Modeli

OSI modeli, bir telekomünikasyon veya bilgi işlem sisteminin iletişim işlevlerini standartlaştıran kavramsal bir modeldir. Amacı, çeşitli iletişim sistemlerinin standart iletişim protokolleriyle birlikte çalışabilirliğidir. OSI'de süreçler arası iletişim 7 bağımsız katmana bölünmüştür [50]. Tablo 2.3'te OSI modeli katmanları gösterilmiştir.

Tablo 2.3 OSI modeli katmanları

Katman	**Sorumlu Olduğu Alanlar**	**Protokoller**
Uygulama	İnsan ve bilgisayar etkileşimli iki uygulamanın birbiriyle konuştuğu alanlarla ilgilenir.	HTTP, WebSocket, SMTP, MQTT
Sunum	Verilerin kullanılabilir formatta olmasını sağlar ve şifreleme ile ilgilenir.	SSL, TLS
Oturum	Bağlantı noktalarını ve oturumları koruyarak iki ana bilgisayar arasında bağlantı kurulumu ve sonlandırılması ile ilgilenir.	PPTP, FTP
Taşıma	İki bilgisayar arasındaki iletişimle ilgilenir.	TCP, UDP
Ağ	IP adresini veri paketinden okuma ile ilgilenir.	IP, IPv4, IPv6
Veri bağlantı	MAC adresini veri paketinden okur.	Ethernet, Wi-Fi
Fiziksel	Ham bit akışının fiziksel ortam aracılığıyla iletiminden sorumludur.	RS-485, RES-232, ISDN

2.2.3 İnternet Protokol Paketi

TCP/IP modelinde kullanılan temel protokoller İnternet Protokolü (IP) ve taşıma katmanında yer alan TCP veya UDP protokolleridir [51]. Genel olarak bilgisayar ağlarında kullanılan iletişim protokollerini kapsar. OSI modelinin aksine iletişim protokolleri 7 katmanda değil 4 katmanda gruplandırılmıştır. Tablo 2.4'te TCP/IP modelinde yer alan katmanlar gösterilmiştir.

Tablo 2.4 TCP/IP model katmanları

Katman	**Görevi**	**Protokoller**
Uygulama	Ağ kaynaklarına erişime izin vermeyi sağlar.	HTTP, TLS/SSL, MQTT, WebSocket, SMTP, POP3
Taşıma	Mesaj teslimi ve hata teslimi için güvenilir süreç sağlar.	TCP, UDP, RSVP
İnternet	Paketleri kaynaktan hedefe taşır.	IP (IPv4, IPv6), ICMP, RIP, DHCP
Veri bağlantı	Aynı ağdaki iki cihaz arasındaki aktarımdan sorumludur.	MAC, ARP, NPD

2.2.3.1 İnternet Protokolü (IP)

İnternet Protokolü (IP), TCP/IP modelinde verilerin ağ sınırları boyunca aktarılması için kullanılan temel iletişim protokolüdür. IPv4 ve IPv6 aktif olarak kullanılan en yaygın İnternet protokolleridir. Bu protokol ile İnternet'te dolaşan veriler, paketler halinde parçalara bölünerek taşınır. IP adresleri, İnternet'e bağlı cihazların ağ üzerinde birbirleri ile iletişim kurmasını sağlayan özel adreslerdir. IPv4 protokolü için IP adresleri 32 bit olarak tanımlanmıştır ve yaklaşık 4.3 milyar civarında farklı IP adresi oluşturabilmektedir. IPv6 protokolü ise 128 bit olarak tanımlanmıştır ve IPv4 e göre çok daha fazla IP adresi oluşturabilmektedir. Buna karşılık IPv6 protokolünü cihazlarda enerji tüketimini ve yanıt gecikme sürelerini arttırır. Tablo 2.5'te IPv4 ve IPv6 karşılaştırılması verilmiştir.

Tablo 2.5 IPv4 ve IPv6 karşılaştırma

Nitelik	**IPv4**	**IPv6**
Çıkış tarihi	1981	1996
Boyut	32 bit	128 bit
Adres sayısı	2^{32} ~ 4.3 milyar	2^{128} ~ 3.4e+38
Konfigürasyon	DHCP / manuel	Otomatik

Cihazlarda IP adresleri dışında MAC adresleri de bulunmaktadır. MAC, ağa bağlı donanımların tanınmasını sağlayan veri bağlantı katmanında yer alan bir protokoldür. Her donanım için özel bir MAC adresi bulunmaktadır. Cihaz üreticileri tarafından kodlanan 48 bit'lik MAC adresi ile sadece yerel ağlarda haberleşmeyi sağlar. IP adresinden farklı olarak ağın dışına taşınabilecek bir adres değildir.

Paketler hedeflerine ulaştığında, IP ile birlikte hangi taşıma protokolünün kullanıldığına bağlı olarak farklı şekilde işlenirler. En yaygın taşıma protokolleri TCP ve UDP'dir.

2.2.3.2 İletim Kontrol Protokolü (TCP)

TCP, verilerin iletilme ve alınma şeklini belirleyen bir aktarım protokolüdür. IP protokolü ile birlikte kullanıldığında TCP/IP olarak adlandırılır. TCP/IP ile gönderilen her veri paketinde TCP başlığı bulunur. TCP üzerinden haberleşme başlatıldığında tüm paketlerin sıra ile gönderilmesi için alıcı ile bir bağlantı açılır. Böylelikle herhangi bir pakette iletim tam anlamıyla yapılmadığında o paket tekrar iletilecektir. Anlaşılacağı üzere TCP ile haberleşmede en önemli olan konu verilerin başarılı bir şekilde gönderilebilmesidir. Verilerin gönderildiğinden emin olabilmek için sürekli kontrol yaptığı için UDP'ye göre veri aktarımı daha yavaştır. HTTP, MQTT, SSH, POP3, FTP gibi uygulama katmanında yer alan protokoller tarafından kullanılmaktadır. Web, e-posta, uzaktan kontrol ve dosya aktarımı gibi alanlarda TCP'den yararlanılır.

2.2.3.3 Kullanıcı Datagram Protokolü (UDP)

UDP aktarım protokolü TCP ye göre daha hızlı ancak daha az güvenilirdir. TCP'de olduğu gibi veri paketlerinin teslim edilmesi kontrolü yapılmaz ve bağlantı açılmaz. Haberleşmede yaşanan gecikme süresini en az indirgemek için tasarlanan bu protokol, geniş alan ağlarında (WAN)

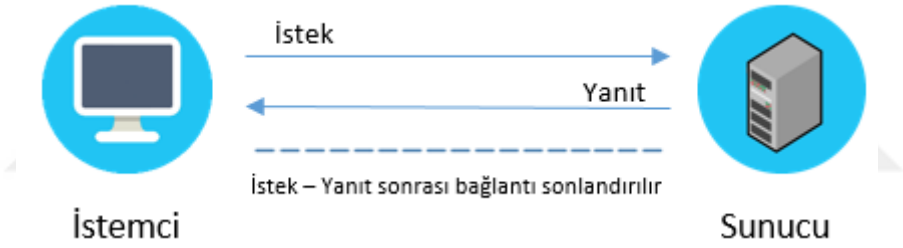
gerçek zamanlı haberleşmeler için tercih edilir. Özelliklede ses ve video aktarımlarında UDP kullanılır. Bant genişliği fazla olmadığı için TFTP ve DNS gibi uygulama katmanı protokolleri tarafından kullanılmaktadır. Güvenlik katmanı protokollerinden TLS/SSL de genellikle bu protokol ile çalışmaktadır.

2.2.4 Uygulama Protokolleri

İnternet protokol paketinin en üst katmanında uygulama protokolleri yer almaktadır.

2.2.4.1 HTTP (Hyper-Text Transfer Protocol)

HTTP, bilgi sistemleri için bir uygulama katmanı protokolüdür [52]. HTTP, istemci-sunucu modelinde bir istek-yanıt protokolü olarak işlev görür. World Wide Web için en önemli iletişim protokolüdür. İstemci, sunucuya bir HTTP isteği gönderir. Gönderilen isteği alan sunucu, çeşitli kontrolleri ve yönlendirmeleri yaptıktan sonra istemciye HTML olarak bir yanıt mesajı geri döner. Yanıt, istekle ilgili tamamlanma durumu bilgilerini içerir ve ayrıca mesaj gövdesinde istenen bilgiyi içerebilir. Şekil 2.8'de HTTP ile haberleşen sunucu ve istemci iletişim yönleri gösterilmektedir.



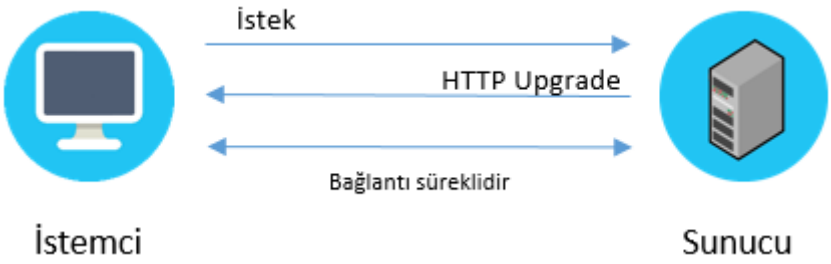
Şekil 2.8 HTTP istemci-sunucu ilişkisi

HTTPS, HTTP'nin güvenli uzantısıdır. İstemci ve sunucu arasındaki veri güvenliği için geliştirilen SSL ve TLS teknolojileri Web'de kimlik doğrulamada ve iletişim gizliliğinde yaygın bir rol oynar. SSL ya da TLS sertifikalarının yüklü olduğu Web siteleri ile sunucular arasındaki tüm iletişim şifrelenir ve böylelikle güvenli bağlantı sağlanır. Ancak HTTPS, altyapı maliyetleri, iletişim gecikmesi, veri kullanımı ve enerji tüketimi açısından ek yük getirebilir [53]. HTTP protokolün kullanıldığı URI adresleri http:// ya da https// ile başlar.

2.2.4.2 WebSocket

WebSocket, tek bir TCP bağlantısı üzerinden çift yönlü iletişim kurmayı sağlayan bir protokoldür. WebSocket protokolü istemci ve sunucu arasında gerçek zamanlı veri aktarımlarını sağlayacak şekilde tasarlanmıştır. WebSocket kullanımı ile sunucu ve istemci arasında sürekli veri yoklama yerine olaya dayalı yanıtlar alınılabilir ve böylelikle sunucu meşgul edilmez. WebSocket, HTTP sunucuları tarafından bir Upgrade isteği olarak yorumlanmaktadır [54]. İletişim TCP 80 portu (veya TLS ile şifrelenmiş bağlantılarda 443 portu) üzerinden gerçekleşmektedir. WebSocket protokolün kullanıldığı URI adresleri ws://

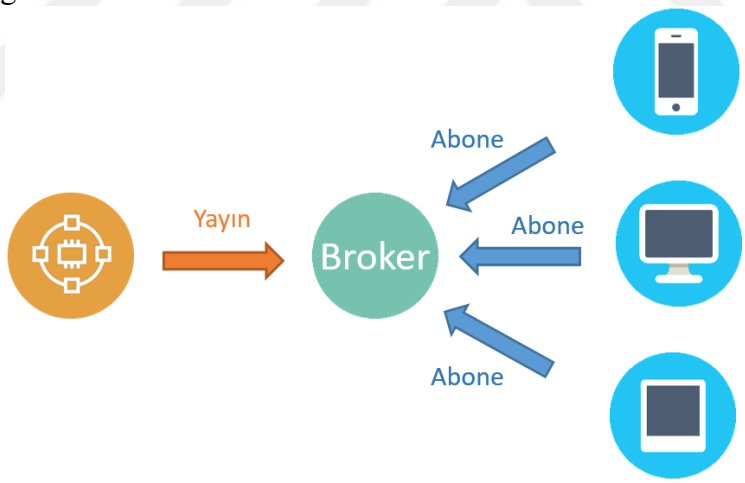
veya wss:// ile başlar. Şekil 2.9'da WebSocket ile haberleşen sunucu ve istemci iletişim yönleri gösterilmektedir.



Şekil 2.9 WebSocket istemci-sunucu ilişkisi

2.2.4.3 MQTT (Message Queing Telemetry Transport)

MQTT, düşük bant genişliğine sahip kısıtlı cihazlar için tasarlanmış TCP/IP protokolü üzerinde çift yönlü çalışan hafif bir mesajlaşma protokolüdür. Asenkron çalışabilirliği, verimli kaynak tüketimi gibi avantajları sebebiyle IoT cihazları arasında veri alışverişi yapmak için yaygın olarak kullanılmaktadır. Şekil 2.10'da görüldüğü üzere cihazlar yayıncı – abone yapısında iletişim sağlamaktadır. IoT cihazları bir konuda yayın yaptığında abonelere bu konudaki veriler iletilmektedir. Bir konuya birden fazla istemci abone olabilir ve tüm abonelerin aynı anda çalışma zorunluluğu bulunmamaktadır.



Şekil 2.10 MQTT yayın-abone ilişkisi

2.2.4.4 SMTP (Simple Mail Transfer Protocol)

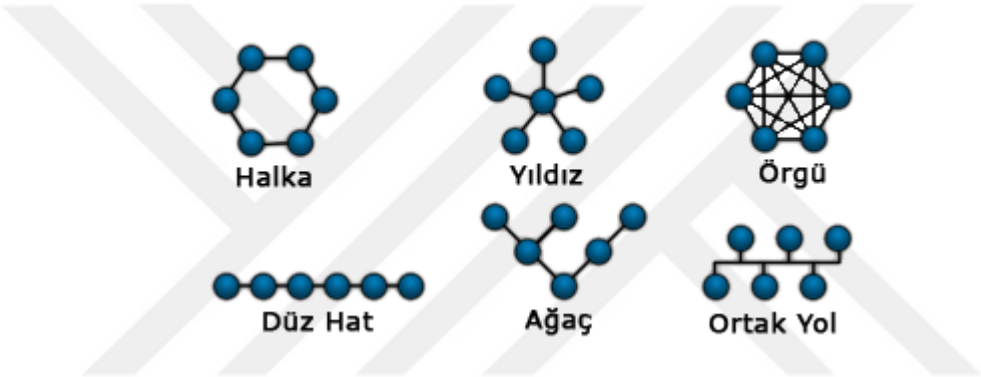
SMTP, sunucu ve istemci arasında e-posta alışverişini sağlayan TCP/IP protokolüdür. Genellikle POP3 ve IMAP ile birlikte kullanılır.

Bu protokollerin dışında İnternet iletişim kuralları dizisinde yer alan uygulama katmanında ve diğer alt katmanlarda birçok farklı protokol ve standart bulunmaktadır.

2.2.5 Ağ Topolojileri

Ağ topolojisi bir haberleşme ağında bulunan bağlantı ve düğümlerin düzenlenmesidir [55]. Radyo dalgaları, endüstriyel fieldbus sistemler ve bilgisayar ağları gibi çeşitli haberleşme ağlarının düzenlenmesinde kullanılır. Ağ topolojisinde, iletişim cihazları düğüm, cihazlar arası bağlantılar ise düğümler arasındaki çizgiler olarak modellenmiştir. Cihazların konumu, birbirlerine olan mesafeleri, iletim hızları ve sinyal türleri ağ topolojisi bileşenleri arasında yer almaktadır.

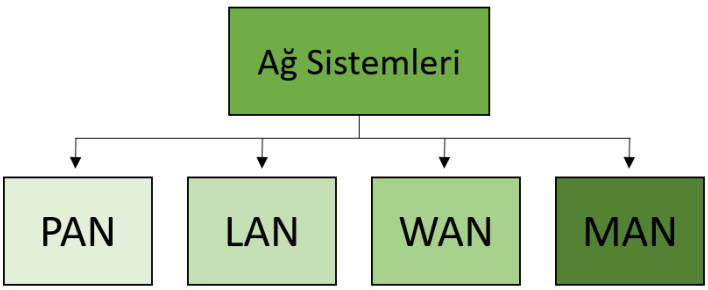
Ağ topolojileri fiziksel ve mantıksal olmak üzere ikiye ayrılır. Haberleşme cihazları arasındaki bağlantının sağlaması ve iletim ortamının düzenlenmesi, ağın fiziksel topoloji bölümünü oluşturmaktadır. Ağın mantıksal topolojisi ise haberleşme sinyallerinin ağ üzerinde izlediği yol ile ilgilenmektedir. Yerel alan ağları (LAN) ve geniş alan ağlarında (WAN) kullanılan doğrusal, halka, yıldız, ağaç ve örgüsel gibi çeşitli ağ topolojileri bulunmaktadır. Şekil 2.11'de farklı ağ topolojileri gösterilmiştir.



Şekil 2.11 Ağ topolojileri

2.2.6 Ağ Sistemleri ve Protokoller

Ağ sistemleri, bilgisayar ve elektronik aygıtların haberleşmesini sağlayan iletişim sistemleridir. İnternet en büyük bilgisayar ağı sistemidir. Büyüklüklerine ve amaçlarına göre farklı bilgisayar ağları ve bu ağlarda kullanılan çeşitli teknolojiler bulunmaktadır. Şekil 2.12'de görüleceği üzere ağ sistemleri kapsama alanlarına göre genel olarak dört bölümde incelenir.



Şekil 2.12 Ağ sistemleri

Kapsama alanlarına göre ayrılan ağ sistemlerinde farklı protokoller kullanılmaktadır. Tablo 2.6'da ağ sistemlerinin menzili ve bazı yaygın kullanılan protokoller gösterilmiştir.

Tablo 2.6 Ağ protokolleri

Mekânsal Kapsam	Menzil	Teknoloji
PAN	< 50 m	IEEE 802.15, Bluetooth, ZigBee
LAN	< 1 km	IEEE 802.11 (Wi-Fi), Ethernet
MAN	< 50 km	IEEE 802.16
WAN	< 150 km	IEEE 802.20, 5G, 4G, GSM, SigFox, LoRa

2.2.6.1 Kişisel Alan Ağı (PAN)

IoT'de oldukça popüler olan PAN, bireysel çalışma alanına odaklanan elektronik cihazları birbirine bağlamak için kullanılan bir bilgisayar ağıdır [56]. PAN sistemleri için en yaygın kullanılan protokoller Tablo 2.7'de verilmiştir.

Tablo 2.7 PAN protokollerinin karşılaştırması

PAN	Güç Tüketimi	Maksimum Menzil	İnternet Entegrasyonu
Bluetooth	Çok Düşük	<50 m	Yok
Wi-Fi	Yüksek	<30 m	Var
ZigBee	Düşük	<50 m	Yok
EnOcean	Çok Düşük	<30 m	Yok

2.2.6.1.1 IEEE 802.15

Cihazlar arası mesafenin az olduğu durumlarda IEEE 802.15 protokolü, iletişim için düşük maliyetli ve hızlı olduğundan tercih edilmektedir. Özellikle ev otomasyon uygulamalarında yaygın kullanılmaktadır. Bu protokol birçok IoT PAN protokolünün temelidir. Ancak bu protokol, diğer cihazlar ile İnternet üzerinden doğrudan iletişim kuramamaktadır. Bu sorun 6LoWPAN ile giderildi.

2.2.6.1.2 6LoWPAN

6LoWPAN, "Düşük güçlü kablosuz kişisel alan ağları üzerinden IPv6" anlamına gelir. IEEE 802.15.4 tabanlı ağlar üzerinden IPv6 paketleri gönderme ve alma yeteneğini sağlar. Yalnızca IPv6 adreslerini kullanmakla kalmaz, aynı zamanda kısıtlı kaynaklara sahip cihazlarda kullanılacak IPv6 başlıklarının boyutunu da optimize eder.

2.2.6.1.3 Bluetooth

Ericsson şirketi tarafından 1994'te geliştirilen Bluetooth, IoT cihazları için önemli bir yer tutmaktadır. Diğer PAN protokollerinden farklı olarak IEEE 802.15.4 protokolünü temel almamaktadır. İlk çıktığından beri önemli ölçüde geliştirilmiştir. Bluetooth 4.0, Bluetooth'u birçok IoT uygulaması için mükemmel bir aday olarak konumlandırdı.

2.2.6.1.4 ZigBee

ZigBee, IEEE 802.15.4'ü temel alan tescillenmiş popüler bir protokoldür. Ağ örgüsünü destekleyen bu protokol OSI modelinde fiziksel katmandan uygulama katmanına kadar uzanır.

2.2.6.2 Yerel Alan Ağı (LAN)

Yerel alan ağı, bina, ofis, okul vb. yerlerde kullanılan küçük ölçekli ağlardır. Bu ağda bilgisayarlar, sucunu ve diğer elektronik aygıtlara bağlanabilir. Yerel ağlarda dış bağlantı olmadığı için en güvenli ağ sistemleridir. LAN ağları küçük boyutlu olduğu için önemli ölçüde hızlıdır ve veri aktarım hızı 1000 Mbps'ye kadar ulaşabilir.

2.2.6.2.1 Wi-Fi

Teknik olarak IEEE 802.11 olarak adlandırılan Wi-Fi, kablosuz bağlantı denilince akla gelen ilk protokoldür. Her yerde bulunduğu için Wi-Fi, OSI modelinin fiziksel katmanı için mükemmel bir eşleşme gibi görünüyor ve bu nedenle TV, mikrodalgalar, müzik çalarlar ve gibi artan sayıda tüketici elektroniği Wi-Fi'yi destekliyor. Ancak Wi-Fi protokolleri (802.11a – n) bazı IoT uygulamaları için sınırlıdır. En büyük sorun enerji tüketimi ve kapama menzilidir. Bu sorunlar üzerinde çalışılıyor ve IoT için optimize edilmiş yeni bir Wi-Fi standardı (IEEE 802.11ah9) hazırlanıyor ve daha iyi bir menzile ve çok daha düşük güç tüketimine sahip olması planlanıyor.

2.2.6.3 Metropolitan Alan Ağı (MAN)

İki veya daha fazla LAN, iletişim amacıyla bağlandığında MAN ağını oluşturur. Örneğin, ticari bir şirketin farklı konumlarda bulunan şubelerinin bir ağ üzerinden birbirleri ile bağlanması sonucu MAN ağı oluşturulabilir. MAN ağlarında veri iletim hızının yüksek olabilmesi için fiber optik kablolar tercih edilir. Şehirlerde kablolu TV ağlarında, endüstriyel, askeri vb. alanlarda bu ağ türü kullanılabilir.

2.2.6.4 Geniş Alan Ağı (WAN)

Cihazlar arası mesafenin fazla olduğu daha geniş ölçekli IoT uygulamaları için WAN sistemler tercih edilmektedir. Geniş bir alanda PAN sistemini kullanmak için çok fazla ağ geçidi ve amplifikatörlere ihtiyaç duyulacağı için maliyet artar. WAN da ise yüksek güçlü antenler ve baz istasyonları iletişimi sağlamaktadır. Cep telefonu ağları WAN sistemlerine bir örnektir. 5G ağları gelene kadar cep telefonları ağı en fazla güç gerektiren protokollerdi. Tablo 2.8'de WAN sistemlerinde kullanılan protokoller hakkında bilgi verilmiştir.

Tablo 2.8 WAN protokollerinin karşılaştırması

WAN Protokolü	Enerji Tüketimi	Kapsam
SigFox	Çok düşük	Orta
LoRa	Çok düşük	Orta
GPRS/3G/4G	Yüksek	Yüksek
5G	Çok düşük	Yüksek

2.2.6.4.1 Mobil Ağlar (GSM)

Mobil ağlar ya da daha iyi bilenen adıyla GSM, mobil ses ve veri hizmetlerini iletmek için oluşturulmuş dijital bir hücresel ağ teknolojisidir. IoT uygulamalarında elektronik cihazlardaki hücresel modemler ile mobil telefon ağlarına kablosuz bağlantı yapılarak İnternet erişimi rahatlıkla gerçekleştirilebilir. Ancak hücresel modemler çok fazla enerji harcamaktadır ve mobil ağlar da başlangıçta IoT için tasarlanmamıştır. Pille çalışan cihazlar için mobil ağlar üzerinden veri gönderip almak uygun değildir. Ayrıca cep telefonu ağlarının milyarlarca cihazın bağlanabilmesi için tasarlanmamıştır. Baz istasyonlarının belli sınırlaması bulunmaktadır. Bir ağ geçidine herhangi bir zamanda sınırlı sayıda cihaz bağlanabilir. Mobil ağlar IoT için kullanımı kolay gözükse de maliyet açısından uygun değildir. Ancak IoT için 5G mobil ağ teknolojisi geliştirilmektedir.

2.2.6.4.2 5G

5G kendisinden önceki mobil ağlara göre daha geniş bir bant aralığına sahip beşinci nesil teknoloji standardıdır [57]. 4G gibi, 5G ağları da hizmet alanının hücre adı verilen küçük coğrafi alanlara bölündüğü hücresel ağlardır. Bir hücredeki tüm 5G kablosuz cihazlar, hücredeki yerel bir anten aracılığıyla radyo dalgaları üzerinden İnternet'e ve telefon ağına bağlanır. 5G ağlarının en büyük avantajı daha yüksek bant genişliğine sahip olmaları ve saniyede 10 Gbit/sn gibi yüksek indirme hızları sunmalarıdır. Bant genişliğinin artması ile aynı anda daha fazla kullanıcı İnternet'e girebilecektir ve bu ağı sadece mobil cihazlar değil, bilgisayar ve IoT cihazları da rahatlıkla kullanabilecektir. IoT cihazlarını yaygınlaşması da 5G alt yapılarının geliştirilmesindeki etkenler arasında yer almaktadır. Ericsson, Huawei, Nokia, Qualcomm ve Samsung gibi bazı şirketler 5G teknolojisi için alt yapılar geliştirmekle beraber 6G teknolojisi için de çalışmalar yapmaktadır. Tablo 2.9'da hücresel ağların karşılaştırılması gösterilmiştir.

Tablo 2.9 Hücresel ağların karşılaştırılması

Nitelik	3G	4G	5G
Kullanıma başlangıç tarihi	2004	2006	2020
Bant genişliği	2mbps	200mbps	>1gbps
Gecikme süresi	100-500 milisaniye	20-30 milisaniye	<10 milisaniye
Ortalama hız	144 kpbs	25 mbps	200 -400 mbps

IoT uygulamalarında doğru ağ protokol yığınını seçmek için cihazlar arası mesafe, kullanılacak enerji miktarı, maliyet gibi kıstaslar göz önünde bulundurulması gerekmektedir.

2.2.7 IoT Platformları

Endüstri 4.0 ile birlikte IoT platformlarının kullanımı hızlı bir şekilde yaygınlaşmaktadır. IoT, haberleşme için gömülü sistem teknolojisini ve İnternet'i kullanan fiziksel nesneler ağıdır. IoT kapsamında kullanılan gömülü cihazlar, sensörlerden veri okuma, aktüatörleri kontrol etme gibi faaliyetleri yönetebilmek için gerçek zamanlı verileri İnternet üzerinden aktarır. Endüstri, sağlık, ulaşım ve askeri alanlar gibi birçok alanda IoT platformları kullanılmaktadır. Şekil 2.13'te IoT uygulamalarında oldukça yaygın kullanılan Raspberry Pi 4 gösterilmiştir.

Şekil 2.13 Raspberry Pi 4

2.2.7.1 Gömülü Sistemler

Tüm IoT cihazlarının gömülü sistemleri vardır, ancak tüm gömülü sistemler IoT değildir. Gömülü bir sistem, herhangi bir elektronik cihazın yönetimini sağlayan çekirdek kısmıdır. Gömülü sistemler içlerinde bulunan entegre devreler sayesinde düşük güç tüketimi ve hızlı işlem kabiliyetine sahiptirler. Cep telefonlarından arabalara, klimalardan insansız araçlara kadar tüm elektronik cihazlarda gömülü sistemler bulunmaktadır. Asıl amaçları kullanıldıkları sistemde veri alışverişini sağlamaktır. Kullanım alanlarına göre çeşitlilik gösteren gömülü sistemler, kullanıcı tarafından kontrol edilebilir, tam otomatik çalışabilir, İnternet'e bağlanabilir ya da bağlanmayabilir. İnternete bağlanan gömülü cihazlar, verileri Wi-Fi, 5G, MQTT gibi haberleşme kanalları ile bulut ortamına iletebilir.

Piyasada endüstriyel ve hobi amaçlı kullanılan çeşitli IoT platformları bulunmaktadır. Hobileri hedefleyen platformlar RAM ve CPU gibi özellikler bakımından daha gelişmiş olsalar da maliyet açısından daha pahalıdır. Endüstriyel platformlar ise daha düşük özellikler sunma eğilimindedir ve maliyetler genellikle daha azdır. Tablo 2.10'da bazı IoT platformlarına ait bilgiler verilmiştir.

Tablo 2.10 IoT platformları

Marka	**CPU**	**RAM**	**Protokoller**	**İşletim Sistemi**
Arduino	ATmega, Intel Curie	16 kB - 64 MB	Wi-Fi, BLE, ZigBee	RTOS, Linux
NodeMCU	ESP8266	128 kB	Wi-Fi	RTOS
Raspberry Pi	ARMv6, ARMv7 1.2GHz	256 MB - 1 GB	Ethernet, USB, BLE	Linux
Intel Edison	Intel Atom 500 MHz	1 GB	Wi-Fi, BLE	Linux
TI CC3200	ARM 80 MHz	512 MB - 2 GB	Wi-Fi, BLE, ZigBee,	Linux
Marvell 88MC200	ARM 200 MHz	256 kB	Wi-Fi, BLE, ZigBee,	RTOS
Broadcom WICED	ARM 120 MHz	256 kB	Wi-Fi, BLE, ZigBee, Thread	RTOS

Mikro denetleyicilere veya mikroişlemcilere sahip donanımlarla ilişkilendirilen gömülü sistemlerde bir diğer önemli konuda yazılımdır. İşletim sistemleri, sürücü programları ve kullanıcı ara yüzleri için yazılım oldukça önemlidir. Gömülü sistemlerde C, Python, Java gibi yaygın yazılım dilleri kullanılmaktadır. Bunun yanında kullanım alanlarına göre farklı işletim sistemleri de bulunmaktadır. Gömülü yazılımlar, sistemin kendisinin ötesinde çalışmasına ve iletişim kurmasına olanak tanır.

2.2.7.2 Gömülü İşletim Sistemleri

İşletim sistemleri, donanım ve yazılım arasındaki bağlantıyı sağlayan yazılımlardır. Şekil 2.14'te donanım ve yazılım arasındaki katmanlar gösterilmektedir. İşletim sistemleri, kaynak kullanımı, bellek yönetimi, güvenlik ve birtakım önemli görevden sorumludur. Temel amacı kullanıcı uygulamalarının verimli bir şekilde çalışabilmesi için bir ortam sağlamaktır. Bilgisayarlar, cep telefonları, beyaz eşyalar ve çeşitli elektronik aygıtlarda işletim sistemleri bulunmaktadır. Kullanım amaçlarına göre farklı işletim sistemleri bulunmaktadır.

Şekil 2.14 Donanım ve yazılım katmanları

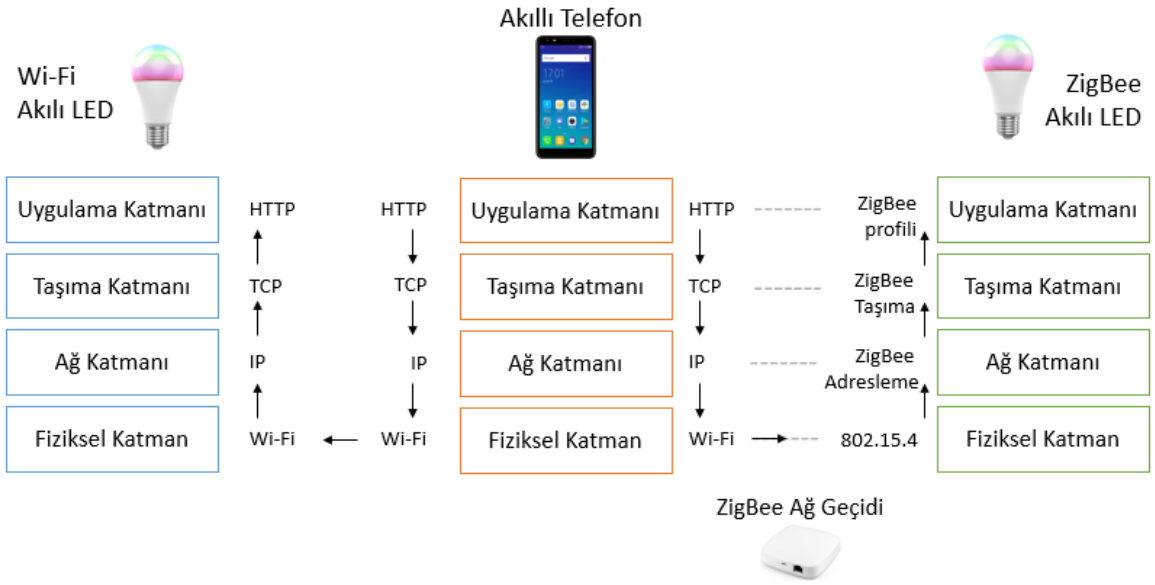
Gömülü işletim sistemleri, gömülü bilgisayar sistemlerinde kullanılması için özel olarak tasarlanmış sistemleridir. Bu işletim sistemlerinin küçük boyutlarda olmaları, kaynakları verimli bir şekilde kullanmaları ve güvenilir olmaları önemlidir. Sınırlı özelliklere sahip oldukları için belirli işler için özel olarak tasarlanmaktadır. Gömülü işletim sistemleri gerçek zamanlı işletim sistemleri (RTOS) kategorisinde değerlendirilmektedir. Bu sistemler donanım kaynaklarını kontrol ederek ve cihazların oluşturulduğu belirli görevler için yanıt sürelerini en aza indirerek verimliliği artırır. Mikro denetleyici (MCU) platformlara kolayca entegre edilebilir. Embedded Linux, Android, Wind River ve VxWorks gibi gömülü işletim sistemleri endüstriyel uygulamalarda yaygın olarak kullanılmaktadır. Gömülü işletim sistemlerinde gerçek zamanlı uygulamaları geliştirmek ve yürütmek kolaydır. Buna karşılık bu sistemler karmaşıktır ve kritik CPU döngülerini tüketebilir.

2.3 Nesnelerin Ağı (Web of Things – WoT)

Klasik IoT uygulamalarında farklı üreticiler tarafından üretilmiş çeşitli IoT sistemleri yer almaktadır. Bu çeşitlilik, iletişim protokollerindeki çeşitliliği, haberleşmek için veri modellerini ve güvenlik gereksinimlerini içerir. IoT uygulamaları genellikle belirli bir kullanım amacı için geliştirilir. Ancak çoğu IoT uygulamasının tekrardan geliştirilmesi ya da bakımının yapılması zordur.

IoT uygulamalarında cihazların kontrolleri bilgisayarlar ya da akıllı telefonlar ile yapılabilmektedir. Örneğin Wi-Fi ya da Bluetooth protokolü ile çalışan bir akıllı LED, akıllı telefonlar ile direkt olarak kontrol edilebilir. Çünkü akıllı cep telefonları bu protokolleri desteklemektedir. Wi-Fi ile çalışan akıllı LED ile akıllı telefon arasındaki iletişim sırasıyla HTTP – TCP – IP – Wi-Fi şeklinde olacaktır. Ancak ZigBee gibi farklı bir ticari protokolü ile kontrol edilen bir LED, akıllı telefonlar ile doğrudan kontrol edilemez. Bunun sebebi akıllı telefonların ZigBee protokolü ile konuşamamasıdır. Bu durumda sağlıklı bir haberleşmenin sağlanabilmesi için ek donanımlar kullanılarak her katmandaki protokolün çevrilmesi

gerekmektedir. Şekil 2.15'te akıllı telefon ile LED'lerin haberleşme yönleri gösterilmektedir [45]. ZigBee için bir ağ geçidi kullanılarak bu sorun çözülebilir. Ancak bu sadece ilgili LED'in kontrolünü sağlayacaktır. Her farklı protokol için ağ geçitlerinin kullanılması ekonomik olarak uygun olmamakla beraber sistemler arasında birlikte çalışabilirliği sağlayamaz. Ayrıca tüm sistemin tek bir mobil ya da Web uygulamasından izlenebilmesi mümkün olmayacak kadar meşakkatli bir iştir.

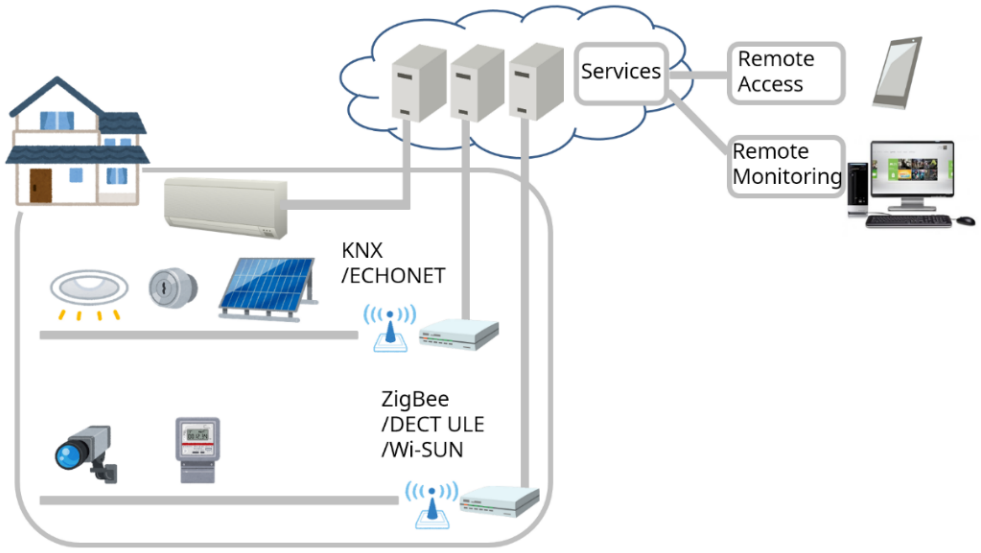


Şekil 2.15 Wi-Fi ve ZigBee LED kontrolü

Bu soruna karşın Web of Things (WoT) bir IoT çözümü olarak sunulabilir. WoT, IoT platformları ve uygulama alanları arasında birlikte çalışabilirliği sağlamak için tasarlanmıştır. WoT, World Wide Web Consortium (W3C) tarafından farklı IoT platformlarının birlikte çalışabilirliği için bir dizi standardı tanımlar. Genel olarak WoT'nin amacı, mevcut IoT standartlarını ve çözümlerini korumak ve tamamlamaktır. Şekil 2.16'da W3C tarafından kullanılan WoT logosu gösterilmiştir.

Şekil 2.16 WoT logosu

Şekil 2.17'de WoT konsepti temel alınarak geliştirilmiş bina içi IoT uygulamasında yer alan cihazlar, sensörler ve aktüatörler farklı protokoller ile ağ geçitlerine bağlanmaktadır [58]. Ağ geçitleri ile veriler sunuculara ya da bulut ortamlara taşınabilir. Daha sonra kullanıcılar için geliştirilen ara yüz uygulamalarında ilgili bilgilendirmeler yapılabilir.



Şekil 2.17 IoT tabanlı akıllı bina [58]

WoT, IoT gibi İnternet'in ağ katmanını kullanmak yerine Hypertext Transfer Protocol, WebSocket, CoAP vb. Web protokollerini kullanarak, uygulama katmanının altındaki herhangi bir protokolden bağımsız olarak birlikte çalışabilirliği sağlamaktadır. Böylece herhangi bir cihazın hangi protokolü kullandığına bakılmaksızın oluşturulacak sistem hem modüler hem de yeniden programlanabilecek şekilde olabilir. WoT, akıllı cihazlar arasında iletişim kurmak için HTTP protokolünden faydalan REST mimarisini kullanmaktadır [59].

Nesnelerin Ağı, farklı standart ve protokol ile çalışan cihazlar arası birlikte çalışabilirliği mümkün kılar. Bunu yaparken de standart Web protokollerini kullanarak gömülü sistemlerle uygulama katmanında haberleşir. Yeni bir standart oluşturmak yerine, yaygın olarak Web gibi evrensel olarak desteklenen bir şeyi yeniden uyarlamaktadır. Bu, geliştirme ve bakım maliyetlerinde azalmaya, zamandan ve paradan tasarrufu sağlar.

2.3.1 Web

Web olarak bilinen World Wide Web, belgelerin ve Web kaynaklarının URL'ler ile tanımlandığı ve İnternet üzerinden erişilebildiği bilgi sistemleridir [60]. Web'i icat eden bilim insanı Sir Timothy Berners-Lee, ilk Web tarayıcısını 1990'da İsviçre, CERN'de çalışırken oluşturdu [61]. Web'de veriler HTTP protokolü üzerinden aktarılır. HTTP, Web için veri iletişiminin temelidir. HTML belgelerinin görüntülenmesinde bu protokolden yararlanılır. Web'de bulunan verilere tarayıcılar ya da web servisler aracılığı ile ulaşabilir. Web kaynakları, kullanıcılıların erişebilmeleri için bir Web sunucusunda yayınlanır.

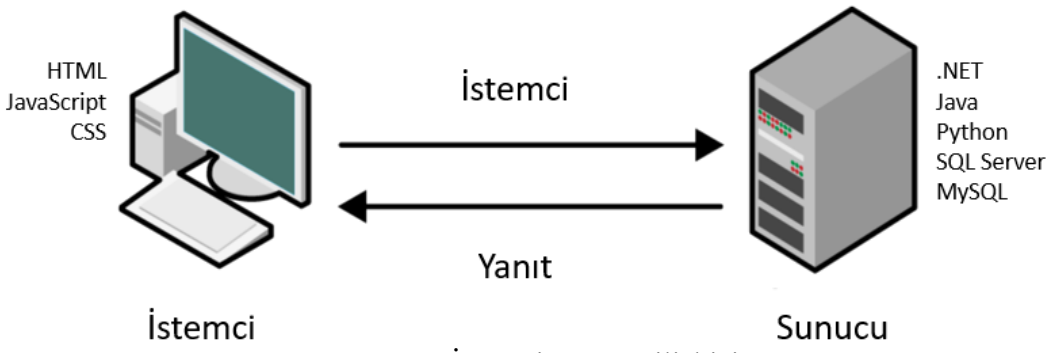
İnternet kullanıcı sayısının artması ve farklı cihazların İnternet'e erişebilir hale gelmesi ile birlikte Web alanında farklı dönemler oluşmuştur. Tablo 2.11'de Web dönemlerine ait karşılaştırma verilmiştir.

Tablo 2.11 Web dönemleri

Nitelik	**Web 1.0**	**Web 2.0**	**Web 3.0**	**Web 4.0**
İçerik türü	Web formları	Web uygulamaları	Akıllı uygulamalar	Yapay zeka, IoT, otonom sistemler
İçerik özellikleri	Sadece okunabilir	Hem okunup hem yazılabilir	Kişiselleştirilebilir	Kendi kendine öğrenebilir
Odak noktası	Doküman	İnsan	Bilgi	Sanal gerçeklik
Kitle	Şirket	Toplum	Bireysel	Yapay zeka
Davranış	Statik	Sosyal	Semantik	Akıllı

2.3.1.1 İstemci – Sunucu

Bir istemci işlemi aynı cihazda çalışabileceği gibi bir ağ üzerinden farklı bir sunucuda da çalışabilir. Sunucu, istemciye bir hizmet sağlar [62]. Genelde sunucular farklı bir bilgisayardadır ve istemciler hizmetlere bir ağ üzerinden erişirler. Sunucudan hizmet almak için istek gönderen Web tarayıcısı, mobil uygulama veya bir bilgisayar istemci durumundadır. Veri tabanı sunucusu, Web sunucusu dosya sunucusu gibi farklı türde sunucular bulunmaktadır. Bir Web sitesinin ya da hizmetinin oluşturulması için hem istemci tarafında hem de sunucu tarafında programlama yapılması gerekmektedir. İstemci tarafında HTML, CSS ve Javascript ile geliştirmeler yapılabilir. Sunucu tarafında ise Java, .NET C#, Python, Node.js gibi diller kullanılarak veri tabanı bağlantıları ve gelen istek için güvenlik ve diğer kontroller yapılarak yanıtlar oluşturulur. Web, tarayıcıdan girmek için kullanıcı ara yüzleri olarak kullanılabileceği gibi uygulamalar ve cihazlar için de ara yüz olmadan API olarak kullanılabilir. Şekil 2.18'de istemci-sunucu ve yaygın kullanılan diller gösterilmiştir.



Şekil 2.18 İstemci-sunucu ilişkisi

İstemci tarafından sunucuya gönderilen isteklerde mesaj gövdesi bulunabilir. İsteğe karşılık sunucu tarafından bir durum kodu ve kaynak yanıt olarak döndürülür. HTTP yanıt mesajlarının ilk satırı durum satırı olarak adlandırılır ve durum kodlarını barındırır. Durum kodları atılan isteğin durumunu belirtir. Eğer kaynak bulunamadıysa 404, yanıt başarılı geldi ise 200 durum kodunu yanıt mesajı içerisinde tutar. Durum kodları üç rakamdan oluşur. Baştaki sayı durum kategorisini gösterir. Örnek: 404 kodu hatanın istemci tarafından kaynaklandığını belirtirken, 500 kodu ise hatanın sunucu kaynaklı olduğunu belirtir. Tablo 2.12'de Web durum kodları verilmiştir.

Tablo 2.12 Web durum kodları

KOD	KATEGORİ
1XX	Bilgi
2XX	Başarı
3XX	Yönlendirme
4XX	İstemci (Client) Hatası
5XX	Sunucu (Server) Hatası

HTTP oturumu ile istemci-sunucu arasındaki iletişimde istemci bilgilerinin sunucuda saklanması ve yapılan isteklerin sorgulanması yapılabilir [63].

2.3.1.2 HTML (Hypertext Markup Language)

HTML, Web sayfalarını oluşturmak için yaygın olarak kullanılan bir işaretleme dilidir. XML sözdizimini kullanmaktadır. Sayfalara görsellik katmak için kullanılan CSS ve dinamiklik sağlayan JavaScript dilleri tarafından desteklenebilir. HTML içerisinde çeşitli amaçlara hizmet eden standart ögeler bulunur. Bu öğelere CSS ve JavaScript ile farklı özellikler kazandırılabilir.

2.3.1.3 URI (Uniform Resource Identifier)

URI, İnternet'teki bir kaynağa atıfta bulunan benzersiz karakter dizisinden oluşan tanımlayıcıdır [64]. Kaynağın hem adını hem de konumunu tutabilir. Bir URI dizisi sırasıyla şema, yetki, yol, sorgu ve parça bölümlerini içerir. Aşağıda URI formatı verilmiştir.

Şema : *schema://authority/path?query#fragment*

Şema (Scheme): URI'nin ilk bileşeni olan şema bilgisi ile atama, belirtim veya protokol bilgisi belirtilir. Birçok farklı şema türü vardır ancak URI içerisinde kullanılacak tüm şemaların İnternet Atanmış Numaralar Otoritesi (IANA)'ne kayıtlı olması gerekmektedir. Yaygın olarak kullanılan şemalara http, wss, ldap, mailto, telnet ve ftp örnek olarak verilebilir.

Yetki (Authority): URI dizesi içerisinde şema bilgisinden sonra isteğe bağlı olarak gelen ve iki eğik çizgi (//) ile başlayan yetki bileşeninin üç alt kümesi bulunmaktadır.

1-) Kullanıcı bilgisi (Userinfo): İsteğe bağlı olarak İki nokta üst üste (:) ile ayrılmış bir kullanıcı adı ve isteğe bağlı bir şifre içerebilir. Kullanıcı bilgisi olan URI'lerde @ sembolünden sonra ana bilgisayar bilgisi yer alabilir.

2-) Ana bilgisayar (Host): Zorunlu olan bu alanda IP adresi veya kayıtlı bir alan adı bilgisi bulunabilir. İki nokta üst üste işaretinden (:) sonra port bilgisi gelebilir.

3-) Bağlantı Noktası (Port): Zorunlu olmayan port bilgisi isteğe bağlı şekilde URI içerisinde yer alabilir. HTTP için 80 numaralı bağlantı noktası ve HTTPS için 443 numaralı bağlantı noktası kullanılır.

Yol (Path): URI içerisinde zorunlu olarak yer alan bu bilgi ile erişilmek istenen kaynağın yolu belirtilir. Şema ve yetki bilgilerinden sonra bir eğik çizgi (/) ile başlar.

Sorgu (Query): URI içerisinde zorunlu değildir. Kullanılması durumunda yol bilgisinden sonra

bir soru işaretinden (?) sonra eklenir. İsim ve değer çifti olarak eklenir. Kullanılabilmesi için sunucu tarafında girilen sorguyu karşılayacak bir kontroller bulunması gerekir.

Parça (Fragment): URI'nin son bileşeni hash(#) sembolü ile başlar ve isteğe bağlıdır. Web sayfalarında sayfa içerisinde yer alan farklı bölümlere gitmek için kullanılabilir.

Örnek URI : *telnet://192.168.43.36:5001/*

URI, kaynağı tanımlamayı ve kaynağın adını veya kaynağın konumunu kullanarak onu diğer kaynaklardan ayırmayı amaçlar. URL ve URN olmak üzere iki alt kümesi bulunmaktadır.

2.3.1.3.1 URL (Uniform Resource Locator)

URL, bir kaynağın konumunu tanımlar ve yalnızca kaynağın konumunu içerebildiğinden URI'nin alt kümesidir. Esas olarak Web sayfalarına erişmek için kullanılır. URL şemaları genellikle HTTP, HTTPS, WebSocket gibi bir protokol ile başlar.

Örnek URL : *https://mehmet@marmara.com:33/forum?tag=master#top*

2.3.1.3.2 URN (Uniform Resource Name)

URN, bir kaynağın adını benzersiz ve kalıcı bir şekilde tanımlar ve konum bilgisini vermez. URN'ler genelde "urn" ön eki ile başlar.

Örnek URN : *urn:uuid:7b6a5e5f-8b99-4d6c-be9e-a0d4b58b16b8*

2.3.1.4 XML (Extensible Markup Language)

XML, verileri depolamak ve verilerin hem insan hem de makine tarafından okunabilmesi için tasarlanmış bir biçimlendirme dilidir. Büyük / küçük harfe duyarlıdır. XML, işaretleme öğelerini tanımlamanızı ve özelleştirilmiş biçimlendirme dili oluşturmanızı sağlar. XML'deki temel birim, öğe olarak bilinir. XML dosyasının uzantısı .xml'dir

2.3.1.5 JSON (JavaScript Object Notation)

JSON, verilerin düzenli ve erişimi kolay bir şekilde depolamak için XML'e alternatif olarak geliştirilmiştir. XML'e göre daha okunaklıdır ve boyut olarak da daha az yer kaplamaktadır [65]. JSON dosya uzantısı .json'dır. Tablo 2.13'te XML ve JSON karşılaştırması gösterilmiştir.

Tablo 2.13 XML ve JSON karşılaştırması

XML	**JSON**
XML verileri tipsizdir ve sadece metinsel (string) ifadeler bulunur.	JSON nesneleri metinsel (string), sayısal (number), dizi (array) veya mantıksal (boolean) olabilir.
Daha güvenlidir.	Daha az güvenlidir.
Verileri okumak daha zordur.	Verileri okumak daha kolaydır.
Bir biçimlendirme dili olduğu için verileri görüntüleme yeteneği sunar.	JSON'un görüntüleme yeteneği yoktur.
XML verilerinin ayrıştırılması gerekiyor.	Verilere JSON nesneleri olarak kolayca erişilebilir.
Verilerin arasında yorum yazılmasına izin verir.	Yorumları desteklemez.

<sensorInfo>
<sensorValues>
<element>
<name>temperature</name>
<value>34</value>
</element>
<element>
<name>humidity</name>
<value>34</value>
</element>
</sensorValues>
</sensorInfo>
	{
	"sensorInfo": {
	"sensorValues": [
	{
	"name": "temperature",
	"value": 34
	},
	{
	"name": "humidity",
	"value": 34
	}
	]
	}
	}

2.3.1.6 Web Servis

Web servisler, bir ağ üzerinden Web belgelerini sunabileceği gibi elektronik cihazların birbirleriyle iletişim kurmasını da sağlayabilir. Web servisler makine tarafından okunabilen XML ve JSON veri taşıma formatlarını HTTP protokolü üzerinden aktarmak için kullanılır.
Web hizmetlerinin iki ana sınıfını tanımlayabiliriz:

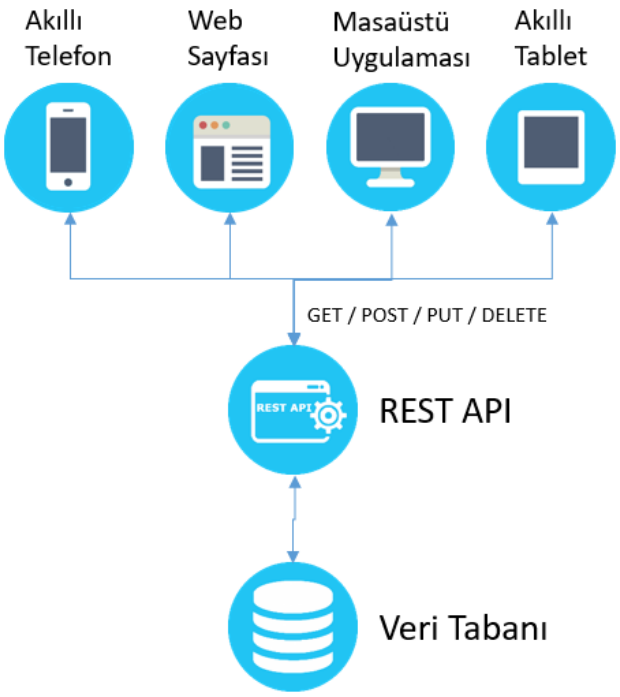
- Hizmetin birincil amacının, tek tip bir durum bilgisi olmayan işlemler kümesi kullanarak Web kaynaklarının XML temsillerini değiştirmek olduğu REST uyumlu Web hizmetleri
- Hizmetin rastgele bir işlem kümesini açığa çıkarabileceği Web hizmetleri [66]

Bir Web hizmetine erişmek için SOAP, WSDL, RPC ve ya REST yapılarından yararlanılabilir.

2.3.1.7 REST (Representational State Transfer)

REST temel olarak HTTP protokolünü kullanan yazılım mimarisidir [67]. Nesnelerin Ağı (WoT) cihazlar arası bağlantıda REST mimarisini takip etmektedir. Nispeten daha karmaşık SOAP, WSDL ve RPC gibi Web servislerine alternatif olarak ortaya çıkan REST günümüzde son derece popülerdir. REST mimarisinin prensiplerini sağlayan servisler RESTful olarak

adlandırılır [68]. RESTful API'ler, ara yüzlerini desteklemek için XML tabanlı Web hizmeti protokolleri (SOAP ve WSDL) gerektirmez. RESTful servisler, farklı platform arası veri paylaşımını sağlayabilmektedir. Sunucu tarafında istenilen derleyici dili kullanılabilir. Yanıt olarak döndürülecek veri JSON veya XML olacağından istemci için kullanılan derleyici dilinin önemi yoktur. Şekil 2.19'da REST API uygulamalar ile etkileşimi gösterilmektedir.



Şekil 2.19 REST API'nin uygulamalar ile etkileşimi

RESTful sistemlerin en önemli özelliklerinden biri istemci ve sunucu haberleşmesinde HTTP metotlarının kullanmasıdır. GET, POST, PUT ve DELETE gibi HTTP metotları RESTful servislerinde kullanılmaktadır [69]. HTTP, isteğin atıldığı Web kaynağın gerçekleştirilmesi istenen eylemi belirtmek için belirli yöntemler kullanılabilir. Tablo 2.14'te HTTP metotları görevleri ile beraber verilmiştir

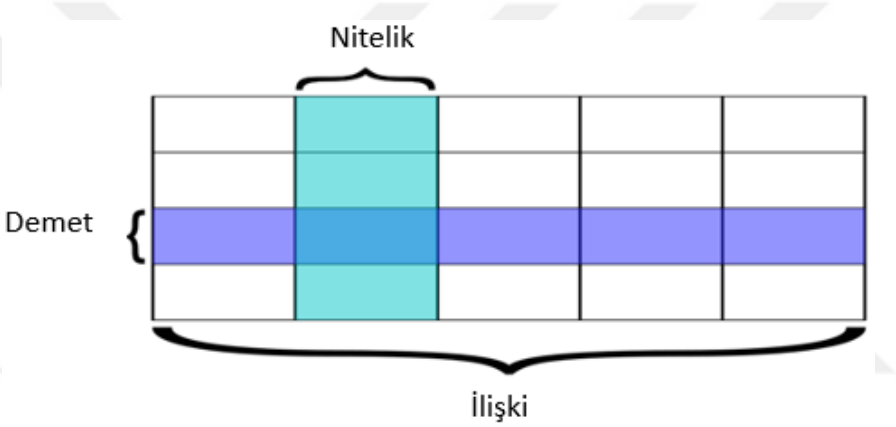
Tablo 2.14 HTTP metotlarının görevleri

HTTP Metodu	**Görevi**
GET	Sunucudan sadece istenen verileri getirmek için kullanılır.
POST	Sunucu tarafında yeni bir veri eklemek için kullanılır. Bu metot kullanılırken isteğin mesaj gövdesine eklenecek olan verilere ait bilgiler yazılır.
PUT	Sunucu tarafında önceden var olan bir veriyi güncellemek için kullanılmaktadır. Post metoduna benze bir şekilde veriler mesaj gövdesi içinde gönderilir. Put metodundan farklı olarak mesaj gövdesinde güncellenecek veriye ait benzersiz anahtarı içerir.
DELETE	İstek ile belirtilen verinin silinmesini sağlar.

2.3.1.8 Veri Tabanı

Veri tabanları, verileri depolayıp daha sonra kullanılabilmesine olanak sağlayan sistemlerdir. Kullanım amaçlarına göre farklı veri tabanı sistemleri bulunmaktadır. Verilerin tablolar halinde saklandığı ve SQL dilinin kullanıldığı veri tabanları ilişkisel veri tabanı olarak adlandırılmaktadır. Microsoft SQL Server, PostgreSQL, MySQL ilişkisel veri tabanlarına örnek olarak verilebilir. Web uygulamalarında .NET, Java gibi diller ile veri tabanı bağlantısı yapılarak veri oluşturma, güncelleme, silme ve okuma işlemleri gerçekleştirilir. Veri tabanı kullanan Web uygulamaları dinamik bir yapıda iken veri tabanı kullanmayanlar ise statik yapıdadır.

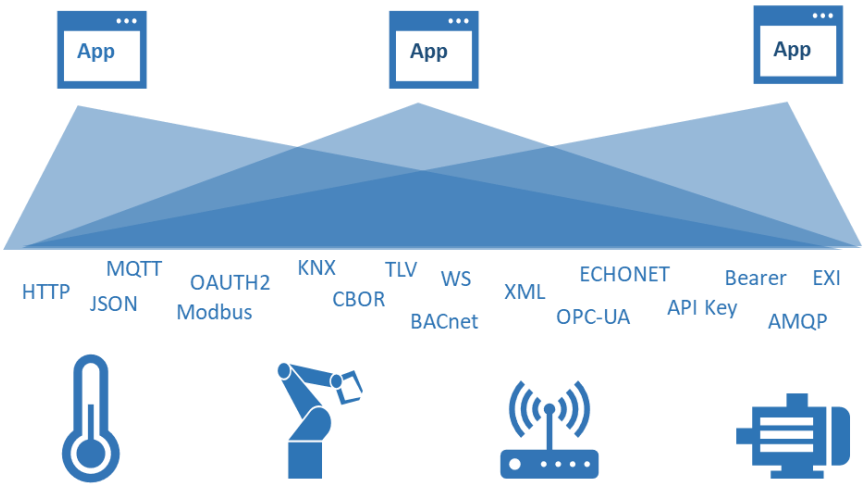
Şekil 2.20'de ilişkisel veri tabanı şeması gösterilmiştir. Örneğin arabalara ait bilgilerin tutulduğu arabalar tablomuz olsun. Arabanın modeli, fiyatı, markası gibi saklanması istenilen her özellik tabloda sütunlara yani niteliklere karşılık gelmektedir. Demet ile de tüm özellikleri ile kaydedilen bir araba kastedilmektedir. Bu sistemde kaydedilen verilerin alabileceği özellikler belli olduğu için okuma, kaydetme gibi işlemler hızlı bir şekilde gerçekleştirilebilir.



Şekil 2.20 İlişkisel veri tabanı yapısı

2.3.2 WoT Yapısı

WoT, standartlaşmış Web teknolojilerini kullanarak IoT uygulamaların geliştirilmesini, bakımını kolaylaştırır. Genel olarak, WoT protokolden bağımsız yaklaşımdır ve HTTP, CoAP veya MQTT gibi belirli protokollerin WoT'nin özellik-eylem-olay etkileşimlerini soyutlamasına nasıl eşleştirilebileceğini tanımlamak için ortak bir mekanizma sağlar. Şekil 2.21'de IoT uygulamalarında sıklıkla kullanılan bazı protokoller yer almaktadır [58].



Şekil 2.21 IoT projelerinde kullanılan protokoller [58]

WoT için tasarlanan mimaride sistem içerisinde yer alan nesnelerin bilgilerinin tutulduğu açıklama dosyaları, protokoller için bağlama şablonları, gizlilik yönergeleri gibi yapı taşları bulunur.

2.3.2.1 WoT Mimarisi

IoT cihazlarını Web kaynakları olarak ortaya çıkaran bir kavram olarak WoT, fiziksel dünyanın Web tabanlı uygulamalarla bağlantı kurmasını sağlamaktadır. OSI modelinin aksine, WoT mimari yığını katı anlamda katmanlardan değil, işlevsellik düzeylerinden oluşur ve İnternet Protokol Yığınının bittiği 7. katmanda başlar [45]. Tablo 2.15'te WoT'e ait 4 katmanlı mimari gösterilmiştir. Her katman, nesneleri Web'e daha fazla entegre etmeye yardımcı olarak, uygulamalar ve insanlar için erişilebilir hale getirmektedir.

Tablo 2.15 Nesneler Ağı katmanları

Katman	**Açıklama**	**Teknolojiler**
4 - Oluşturma	Ögeleri ve sanal Web hizmetlerini içeren uygulamalar oluşturmayı daha da basit hale getirmektedir.	Web veya UI Uygulamaları, sistem entegrasyonu
3 - Paylaşım	Nesneler tarafından oluşturulan verilerin Web üzerinden verimli ve güvenli bir şekilde paylaşılabilmesini sağlar.	API Token, TLS, JWT
2 - Bulma	API aracılığı ile sunulan verilerin ve hizmetlerin kullanımını hakkında bilgi sunar.	Web Thing Model, Thing Description
1 - Erişim	Nesnelerin sağladığı hizmetlere, cihazların ve uygulamaların kolayca iletişim kurabilmesi için verileri Web Nesnesine dönüştürmekten sorumludur.	HTML, JSON, WebSockets, HTTP, URI/URL
Nesneler	Gömülü sistem cihazları, sensörler, aktüatörler ve çeşitli protokoller gibi ağ bağlantılı nesneleri içerir.	Bluetooth, Wi-Fi, 5G, QR, 6LoWPAN

2.3.2.1.1 WoT Ağ Geçidi (WoT Gateway)

Gömülü sistemleri Web'e entegre etmek için HTTP sunucularını, gömülü sistemlere yerleştirmek gerekmektedir. Ancak gömülü sistemler belirli bir amaca göre tasarlandıkları için sınırlı kaynaklara sahiptir ve HTTP sunucuları bu sistemlere yerleştirmek mümkün olmayabilir. Bu durumda Web entegrasyonu için bir ağ geçidine ihtiyaç duyulur. WoT Gateway, İnternet erişimi olmayan fiziksel cihazları İnternet'e bağlar ve bu cihazların verilerini ve yeteneklerini programlanabilir Web Hizmeti API'sine soyutlar. WoT tabanlı sistemlerde cihazlar doğrudan API erişimi yoluyla, ya da bir akıllı ağ geçidi üzerinden dolaylı API erişimi ile İnternet'e erişebilmektedir. WoT Gateway, WoT mimarisinin bir uygulamasıdır ve IoT hizmeti olarak erişme, bulma, paylaşma ve oluşturma hizmetlerini sunmaktadır.

2.3.2.1.2 WoT Etkileşimleri

WoT mimarisi, özellikler, olaylar ve eylemler olmak üzere üç temel etkileşim soyutlaması sunar. Bu soyutlamalar ile IoT uygulamasının ya da hizmetinin meta verileri kolayca elde edilerek cihazların işlevleri rahatlıkla anlaşılır. Şekil 2.22'de WoT mimarisinde yer alan etkileşimler gösterilmiştir.

Şekil 2.22 WoT mimarisindeki etkileşimler

Özellikler (Properties): WoT mimarisinde bir nesnenin durumunu veya değerini gösteren etkileşim türüdür. Sadece okunabilen sensör değerleri, okuma ve yazma işlemlerinin yapılabileceği aktüatör durum bilgileri veya konfigürasyon parametreleri bu etkileşim altında verilir. Örnek olarak dim edilebilen bir LED aydınlatmanın mevcut dim değeri verilebilir.

Eylemler (Actions): WoT mimarisinde bir nesneye ait işlevin bir öğe üzerinden çağrılmasına izin veren etkileşim türüdür. Birden çok özelliğin aynı anda değiştirilmesi veya uzun süreli süreçlerde bu etkileşim altında verilir. Otomatik bir kapıyı açmak örnek verilebilir.

Olaylar (Events): WoT mimarisinde belirli bir koşul sağlandığında eş zamansız olarak gerçekleşebilecek olaylar bu etkileşim ile ifade edilebilir. Bir yangın alarmın tetiklenmesi, butona basılması, çalışan bir cihazın sıcaklığının aşırı yükselmesi vb. planlı olmayan durumlar, olaylara örnek olarak verilebilir.

2.3.2.1.3 Hiper Ortam Kontrolleri

WoT mimarisi, REST temellerinden biri olan hiper ortam ilkesinden yararlanır. Nesnelerden bilgi almak ya da onları kontrol etmek için hiper linkleri kullanır. Hiper ortam denetimleri bir Web sunucusunda gerçekleştirilir. Web istemcisinden yapılan istek ile sunucu arasında etkileşim başlar. Ardından sunucu, istemcinin mevcut durumunu ve yetkilendirme gibi diğer faktörlerini kontrol ederek, istemciyi Web uygulamaları üzerinden dinamik olarak yönlendirebilir.

2.3.2.2 WoT Yapı Taşları

Web of Things (WoT) yapı taşları, soyut WoT mimarisi ile uyumlu sistemlerin uygulanmasına izin verir.

2.3.2.2.1 Nesne Açıklaması (Thing Description - TD)

World Wide Web Consortium tarafından önerilen WoT'nin merkezinde fiziksel cihazların meta verilerini makine tarafından okunabilir formatta sunan Nesne Açıklaması (Thing Description - TD) bulunur. TD, URL'lerin erişilebilir olduğu durumlarda, Web üzerinden IoT cihazlarına erişime izin vererek [70] IoT'nin heterojen uygulamalarla entegrasyonuna yardımcı olur [45].

WoT yapı taşlarının temek bileşeni Nesne Açıklamasıdır. Nesnelerin Ağı'nda yer alan "Nesne" kavramı, fiziksel cihaz ya da sanal hizmetlerin soyutlanmasıdır ve her nesne standartlaştırılışmış zengin meta verilerle tanımlanmaktadır. Bu meta veriler kullanıcılara TD ile sunulmaktadır. Hem makine hem de insanlar tarafından anlaşılabilmesi için JSON formatında olan TD'ler, nesnelerin özelliklerini, nasıl kullanılabilecekleri vb. bilgileri barındırır. Son derece esnek olan TD'ler içerisinde kimlik ve yetkilendirme bilgileri de bulunabilir. Bu ortak soyutlamayı etkinleştirmek için gereken tüm bilgileri içeren bir IoT cihazının meta verileri, WoT Nesne Açıklaması (TD) olarak adlandırılan belgede belgelenmiştir. Tablo 2.16'da TD içerisinde yer alan terimler açıklamaları ile birlikte verilmiştir.

Tablo 2.16 Nesne Açıklamaları (TD) terimleri

TD terimleri	Açıklama	Zorunluluk	Veri Tipi
@context	Bir TD belgesi boyunca kullanılan terimler olarak adlandırılan kısa el adlarını tanımlar.	Zorunlu	Metinsel veya Dizi
@type	Nesneyi anlamsal türlerle etiketler.	İsteğe bağlı	Metinsel veya Dizi
id	Nesneye erişebilmek için URI biçimindeki diziyi temsil eder.	İsteğe bağlı	Metinsel
title	Kullanıcılar için okunabilir bir başlık sağlar.	Zorunlu	Metinsel
description	Nesneye ait ek bilgileri sunar.	İsteğe bağlı	Metinsel
version	Sürüm bilgisi sağlar.	İsteğe bağlı	Metinsel
properties	Nesneye ait tüm özellikleri sunar.	İsteğe bağlı	Dizi
actions	Nesneye ait tüm eyleme dayalı etkileşimleri sunar.	İsteğe bağlı	Dizi
events	Nesneye ait tüm olaya dayalı etkileşimleri sunar.	İsteğe bağlı	Dizi
links	Nesne Açıklaması ile ilgili kaynaklara Web bağlantıları sağlar.	İsteğe bağlı	Dizi
security	Seçilen güvenlik tanımı kümesidir. Kaynaklara erişim için getirilmelidir.	Zorunlu	Metinsel veya Dizi

2.3.2.2.2 Bağlama Şablonları

IoT uygulamalarında çeşitli iletişim protokolleri kullanılmaktadır. WoT bağlama şablonları ile bu çeşitliliğin üstesinden gelmektedir. Bağlama şablonları, farklı protokollerin kullanılmasıyla gelen sorunu ortadan kaldırmak için meta veri planları koleksiyonu sağlar. Bağlama şablonları bir kez oluşturulduktan sonra bir daha oluşturulmasına gerek yoktur. Web protokolleri (HTTP gibi) ve standartlar, WoT'nin başarılı bir şekilde uygulanmasında kritik bir faktördür. Protokol bağlaması, HTTP, MQTT veya CoAP gibi Web protokollerinin somut mesajlarına eşlemesidir. Birlikte çalışabilirliği desteklemek için REST yapısını takip ettiği için tüm haberleşme protokolleri Protokol Bağlamalarını uygulamaya uygun değildir. WoT mimarisinin uygulanabilmesi için en az bir protokol bağlamasının uygulanması gerekmektedir. Kullanılacak protokol türü ile hiper ortam kontrolleri seri hale getirilmelidir.

2.3.2.2.3 Güvenlik ve Gizlilik Yönergeleri

WoT mimarisinde, güvenlik ve gizlilik konularına da önem verilmiştir. TD içerisinde yer alan meta verilerin ve komut dosyaların güvenliği, kimlik yönetim kontrolleri gibi Web teknolojileri kullanılarak sağlanmaktadır.

2.4 IoT ve WoT Karşılaştırması

IoT, nesneler, şeyler, insanlar, hizmetler ve uygulamalardan oluşan bir ağ oluşturmayı amaçlarken, WoT bunları Web'e entegre eder. WoT, IoT ile yakından ilişkilidir. WoT önemli bir aşamayı temsil eden IoT çözümlerinde ölçeklenebilir bağlantı sağlayan yeni bir teknolojidir. Tablo 2.17'de IoT ve WoT karşılaştırılması verilmiştir.

Tablo 2.17 IoT ve WoT karşılaştırılması

IoT	WoT
IoT her şeyi İnternet'e bağlamak için bir donanım katmanıdır.	WoT, her şeyi Web'e bağlamak için bir yazılım katmanıdır.
IoT, nesneler, insanlar, sistemler ve uygulamalardan oluşan bir ağ oluşturur.	WoT, nesneleri, insanları, sistemleri ve uygulamaları Web'e entegre etmeye çalışır.
IoT, bir RFID etiketinden akıllı şehre ve aralarındaki her şeye kadar kullanılan sensörler, aktüatörler ve iletişim ara yüzü ile ilgilenir.	WoT, protokoller ve web sunucuları ile ilgilenir.
Her IoT cihazı için farklı bir protokol kullanılabilir.	WoT, birden çok IoT cihazı için birkaç Web protokollerinden birini kullanarak bunu kolaylaştırır.
IoT'de, yüzlerce uyumsuz protokolün birlikte çıkması, çeşitli cihazlardan veri ve hizmetlerin entegrasyonunu son derece karmaşık ve maliyetli hale getirir.	WoT, standart Web protokolleri kullanılarak herhangi bir cihaza erişir, heterojen cihazları Web'e bağlar, sistemler ve uygulamalar arasında entegrasyonu çok basit hale getirir.
IoT genellikle OSI yığınının alt katmanlarına odaklanır.	WoT yalnızca OSI yığınının 7.katmanı olan uygulama katmanı ile ilgilenir.
Ağ katmanı üzerinden iletişim sağlanır.	Uygulama katmanı üzerinden iletişim sağlanır.
Standart bir iletişim protokolü eksikliği vardır.	RESTful API aracılığıyla iletişim vardır.
Nesneler RFID, QR-Kodu gibi tanımlanabilir.	Nesneler URI ile tanımlanır.
IoT standartları ve prototipleri özel olarak finanse edilir ve herkesin erişimine açık değildir.	WoT herkes için ücretsizdir ve her zaman her yerden erişilebilir.
IoT platformlarının çoklu protokoller nedeniyle programlanması zordur.	Ortak API'ler nedeniyle protokolü işlemek için WoT programlaması daha kolaydır.

3. UYGULAMA

Bu çalışmada modüler bir akıllı bina otomasyonu için akıllı binalarda bulunan üç farklı alt sistem tek bir IoT uygulaması altında birleştirilmiştir. Her sistem içerisinde farklı sensörler ve aktüatörler yer almaktadır. Ve her donanım farklı protokol veya standart ile WoT Gateway aracılığıyla haberleşmektedir. Uygulamada yer alan WoT Gateway, WoT mimarisi temel alınarak tasarlanmıştır ve IoT teknolojisine dayalı hizmetlere erişme, bulma ve paylaşma imkânı sunmaktadır. Aşağıda tasarlanan WoT Gateway'in mimari katmanlarının işlevleri hakkında bilgi verilmiştir.

Katman 1 (Erişim): WoT Gateway'e yapılan istekler kimlik doğrulama işlemine tabi tutulmaktadır. Sisteme ait verilere erişebilmek için istemcinin "/authenticate/login" uç noktasına kullanıcı adı ve şifresini HTTP POST metodu ile istek atması gerekmektedir. İstek gövdesinde yer alan bilgiler doğru ise erişim için JSON Web Token (JWT) yanıt olarak döndürülür. Bu JWT, istemci tarafından bir sonraki işlemler için kullanılacaktır.

Katman 2 (Bulma): Kullanıcılar tarafından "/things" uç noktasına yapılacak GET isteği ile sistemde bulunan tüm cihazlar görüntülenebilir.

Katman 3 (Paylaşım): İlgili uç noktaya yapılacak istek sonucu nesneye ait Nesne Açıklaması (TD) yanıt olarak gönderilir. Böylece sensör verileri okunabilir, aktüatörlerin nasıl kontrol edileceği anlaşılabilir.

Katman 4 (Cihaz Entegre): Sisteme yeni bir cihaz eklenmek istendiğinde yapılması gereken sadece sunucu tarafında gerekli dosyaların oluşturulmasıdır. Böylelikle eklenecek cihaz, otomatik olarak WoT Gateway'in cihaz listesine dâhil edilir.

Uygulama için WoT Gateway olarak içerisinde Linux işletim sistemi koşulan bir Raspberry Pi cihazı seçilmiştir. Raspberry Pi, diğer donanımlar ile farklı standartlar üzerinden çift yönlü haberleşmektedir. Raspberry Pi, istemcileri, cihazları, hizmetleri ve bunların olası tüm etkileşimlerini yönetebilen, HTTPS protokolünü destekleyen ve JSON Web Token (JWT) kimlik doğrulamasını uygulayan bir REST sunucusu konumundadır. WoT'nin temel bileşeni olan TD otomatik olarak burada üretilir. Sensörlerden veri okumak için GET, aktüatörlere komut göndermek için POST metodu kullanılır.

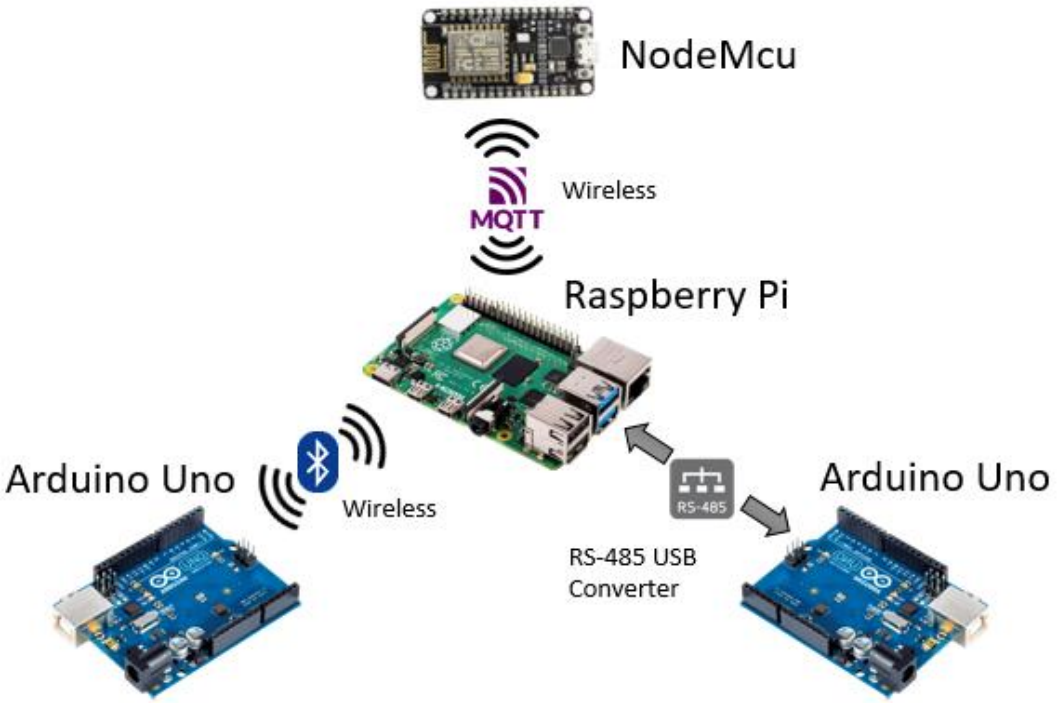
Aşağıda sıcaklık ve nem sensörüne ait özelliklerin, güvenlik şemasının, bağlam tanımının, cihaz URL bilgilerinin ve genel açıklamasının sunulduğu TD gösterilmektedir. Sensörün sağladığı özellikler "Properties" alanı içerisinde yer almaktadır. Ayrıca TD içerisinde yer alan linkler ile de sadece sensöre ait istenen veriler çekilebilir. Böylelikle kullanıcılara istenilen şekilde bilgilendirme yapılabilir.

```
[
    ...
    {
        "@context": "https://www.w3.org/2022/wot/td/v1.1",
        "security": {
          "authorizationUrl": "/api/authenticate/login",
          "scheme": "bearer",
          "format": "JWT"
        },
        "@type": "Sensor",
        "id": "https://localhost.com/things/dht11",
        "title": "Temperature & Humidity Sensor",
        "description": "Ambient temperature and humidity measurement",
        "system": "HVAC",
        "properties": [
          {
            "type": "double",
            "title": "Temperature",
            "description": "Temperature Property",
            "unit": "degree celsius",
            "value": 28.53,
            "links": [
              {
                "href": "https://localhost.com/things/dht11/properties/temperature"
              }
            ]
          },
          {
            "type": "double",
            "title": "Humidity",
            "description": "Humidity Property",
            "unit": "percent",
            "value": 63.55,
            "links": [
              {
                "href": "https://localhost.com/things/dht11/properties/humidity"
              }
            ]
          }
        ],
        "actions": null,
        "events": null,
        "links": [
          {
            "rel": "properties",
            "href": "https://localhost.com/things/dht11/properties"
          }
        ]
    }
    ....
]
```


Sıcaklık ve nem sensörü için yazılan Nesne Açıklaması (TD)

3.1 Donanım

Tez kapsamında geliştirilen yazılımların test edilebilmesi için bir donanım modeli hazırlanmıştır. Uygulama donanımı: 1 adet ARM tabanlı mikroişlemciye sahip Raspberry Pi 4, 2 adet Arduino Uno, 1 adet NodeMCU ESP8266 mikroişlemcileri ile onlara bağlı sensör ve aktüatörlerden oluşmaktadır. Raspberry Pi, WoT Gateway olarak tüm sistemi kontrol etmektedir. Arduino Uno ve NodeMCU ESP8266 ise kendilerine bağlı sensörler ve aktüatörler ile alt sistemleri temsil etmektedir. Sistemin geliştirilebilir ve modüler olduğunu göstermek amacıyla farklı donanım kartları ve protokoller seçilmiştir. Şekil 3.1'de tasarlanan donanım şeması gösterilmektedir.



Şekil 3.1 Uygulama donanım şeması

Donanımda kullanılan kartlarının bazı teknik özellikleri aşağıda verilmiştir.

Raspberry Pi :

- 4 Çekirdekli Cortex-A72 (ARM v8)
- 2GB LPDDR4-3200 SDRAM
- 4/5.0 GHz 802.11ac destekli kablosuz ağ, BLE destekli
-

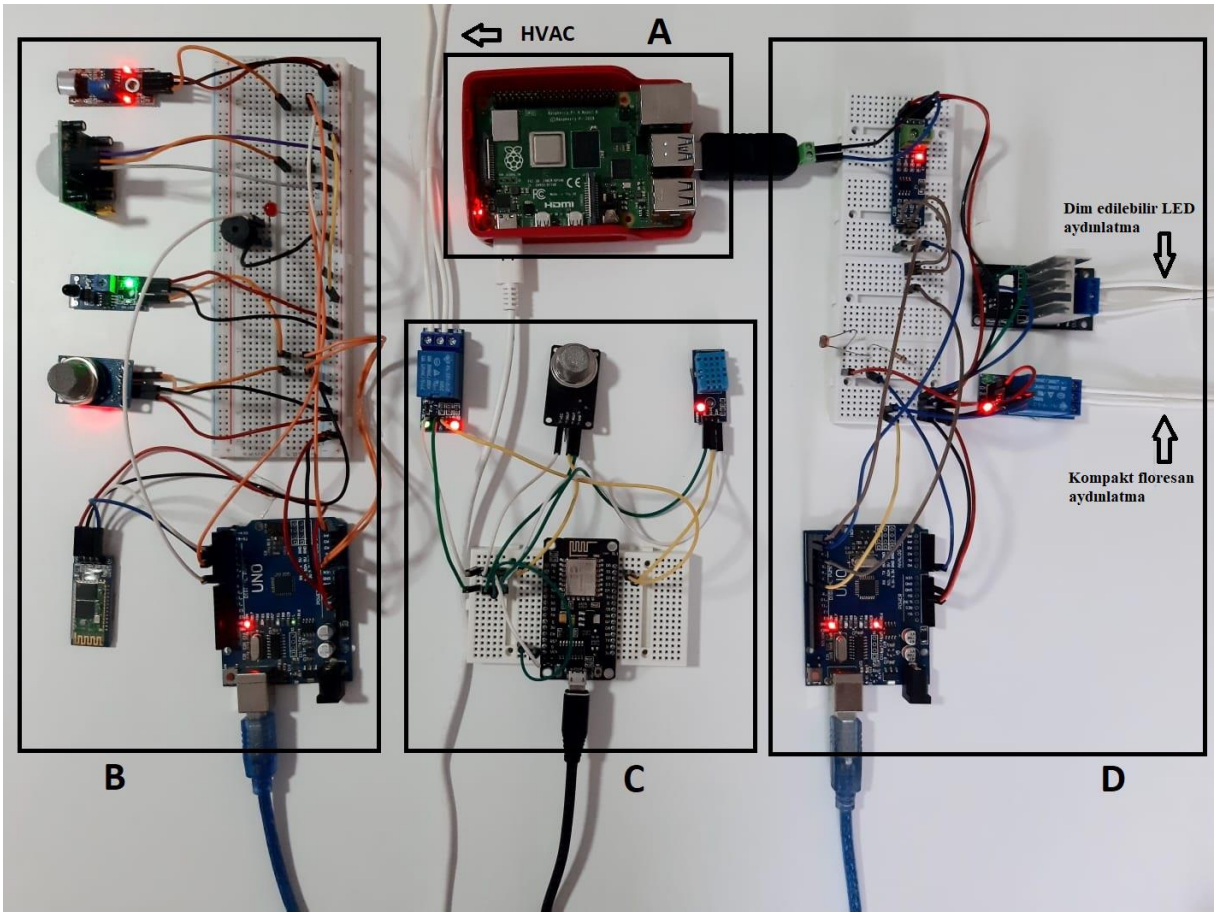
Arduino Uno:

- ATmega328 mikro denetleyici
- 32 Kb hafıza

NodeMCU V3 ESP8266:

- ESP8266 SDK tabanlı
- Wi-Fi, MQTT, CoAP, GPIO, PWM, IIC desteği

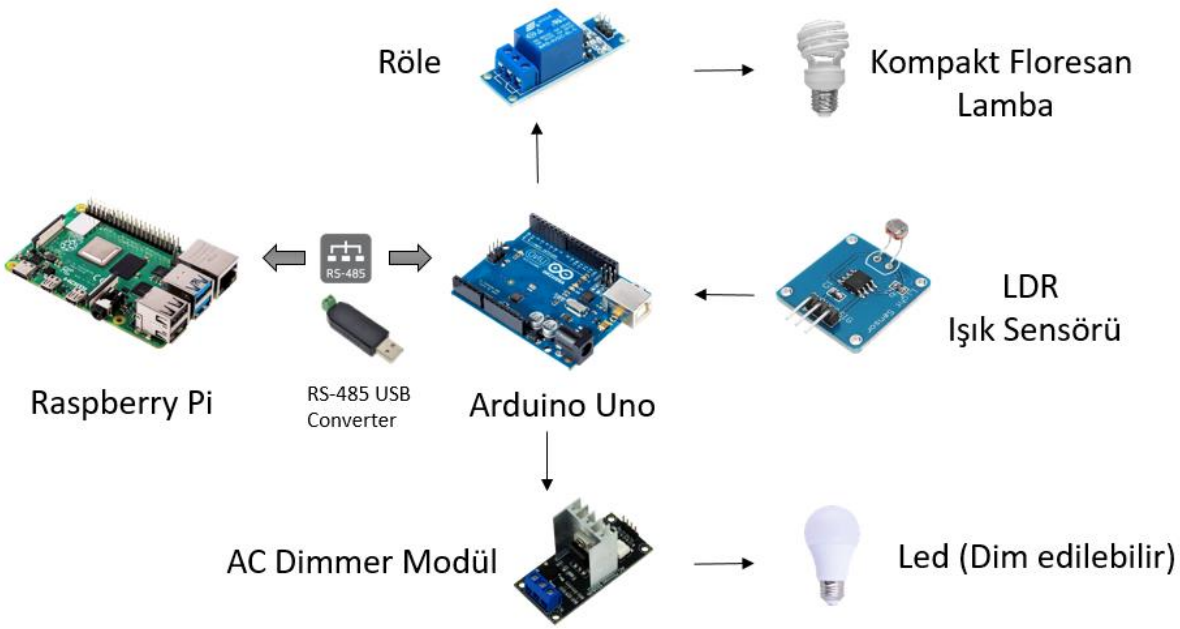
Emniyet ve güvenlik alt sistemi Bluetooth üzerinden, HVAC alt sistemi MQTT protokolü üzerinden ve aydınlatma sistemi de RS-485 standardı üzerinden ağ geçidi ile haberleşmektedir. Şekil 3.2'de oluşturulan donanım sistemi gösterilmektedir.



Şekil 3.2 A-WoT Gateway, B-Emniyet ve güvenlik sistemi, C-HVAC sistemi, D-Aydınlatma sistemi

3.1.1 Aydınlatma Sistemi

Aydınlatma sistemi için konumlandırılan cihazlar, Raspberry Pi ile RS485 standardı üzerinden haberleşen Arduino'ya bağlanmıştır. Arduino, röle ile kompakt floresan lambayı (CFL) ve AC dimmer modülü ile de LED lambayı kontrol etmektedir. Ortamdaki ışık şiddetinin ölçülmesi için de ışık sensörü (LDR) kullanılmıştır. Şekil 3.3'te aydınlatma sistemi için tasarım şeması gösterilmektedir.



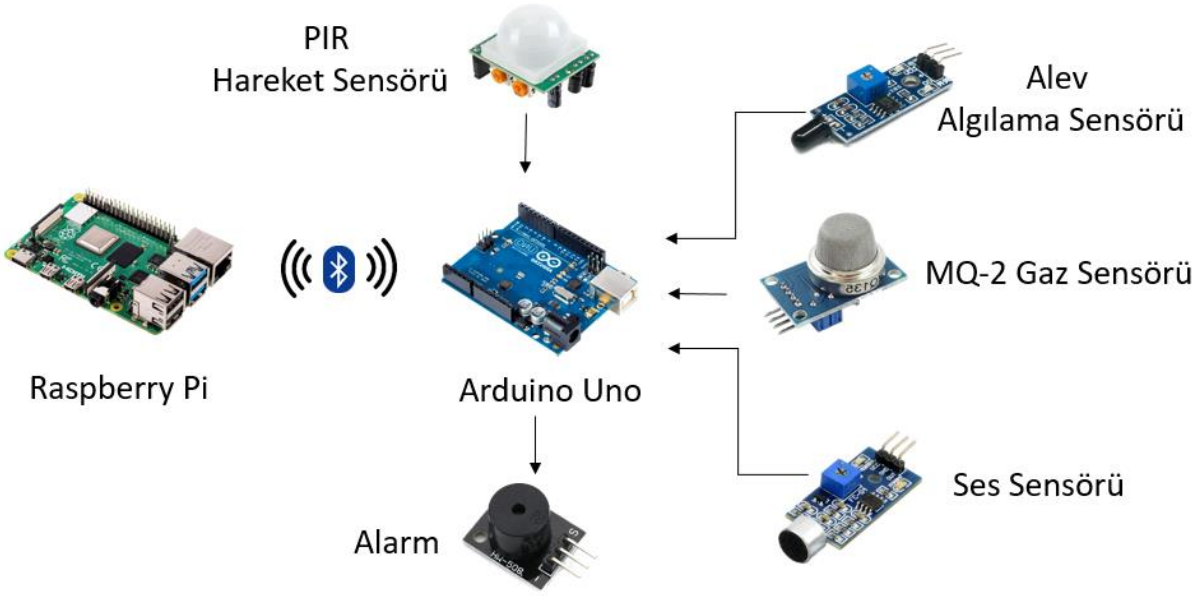
Şekil 3.3 Aydınlatma sistemi şeması

RS-485, OSI modelinde fiziksel katmanda yer alan endüstriyel uygulamalarda yaygın kullanılan iletişim standardıdır. Yüksek hızlı veri iletiminin yanında uzun kablolama mesafelerini de desteklemektedir. Master – Slave yapısına bağlı çalışarak birden fazla cihazın tek bir hat üzerinden iletişim kurmasına izin verir.

Arduino tarafından sensör değerlerinin okunması ve lambaların kontrol edilebilmesi için C programlama dilinden ve "ModbusRtu" kütüphanesinden yararlanılmıştır. Arduino tarafından gönderilen verilerin Raspberry Pi tarafında kullanılabilmesi için ise Python dilinden "minimalmodbus" kütüphanesinden yararlanılmıştır. İki cihaz arasında çift yönlü haberleşme RS-485 üzerinden sağlanmaktadır.

3.1.2 Emniyet ve Güvenlik Sistemi

Raspberry Pi ile Bluetooth protokolü üzerinden haberleşen Arduino'da ortamdaki hareketliliği algılamak için hareket ve ses sensörü, emniyet için alev algılama sensörü ve MQ-2 zararlı gaz ölçüm sensörü kullanılmıştır. Ayrıca aktif edildiğinde ses ile uyarı verebilen bir buzzer alarm bulunmaktadır. Şekil 3.4'te emniyet ve güvenlik sistemi için tasarım şeması gösterilmektedir.

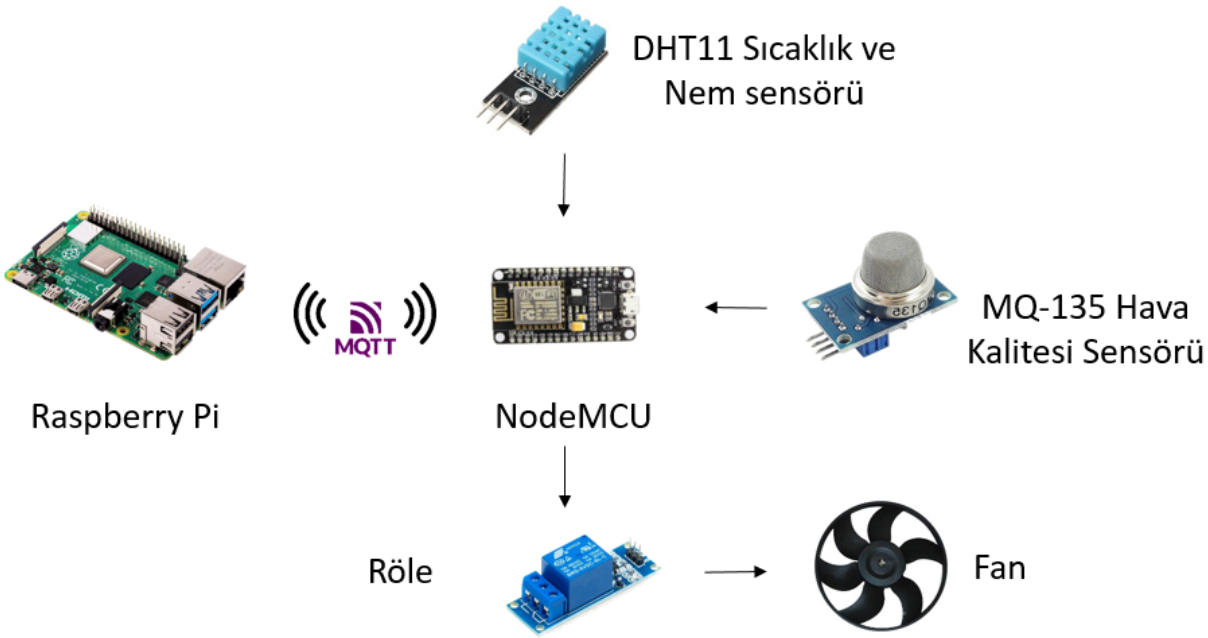


Şekil 3.4 Emniyet ve güvenlik sistemi şeması

Arduino tarafından sensör değerlerinin okunması ve alarmın kontrol edilebilmesi için C programlama dilinden yararlanılmıştır. Arduino tarafından gönderilen verilerin Raspberry Pi tarafında kullanılabilmesi için ise Python dilinden yararlanılmıştır. İki cihaz arasında çift yönlü haberleşme Bluetooth üzerinden sağlanmaktadır.

3.1.3 HVAC Sistemi

HVAC için tasarlanan sistemde Raspberry Pi ile MQTT üzerinden iletişim kuran NodeMCU kartı bulunmaktadır. Ortam sıcaklığını ve nem değerini ölçmek için DHT11 sıcaklık-nem sensörü, hava kalitesini ölçmek için MQ135 sensörü ve röle ile kontrol edilebilen fan kullanılmıştır. Şekil 3.5'te HVAC sistemi için tasarım şeması gösterilmektedir. MQTT ile haberleşmede Raspberry Pi yayımcı, NodeMCU ise abone konumundadır.



Şekil 3.5 HVAC sistemi şeması

NodeMCU tarafından sensör değerlerinin okunması ve fanın kontrol edilebilmesi için C programlama dilinden yararlanılmıştır. Arduino tarafından gönderilen verilerin Raspberry Pi tarafında kullanılabilmesi için ise Python dilinden yararlanılmıştır. İki cihaz arasında çift yönlü haberleşme MQTT üzerinden sağlanmaktadır.

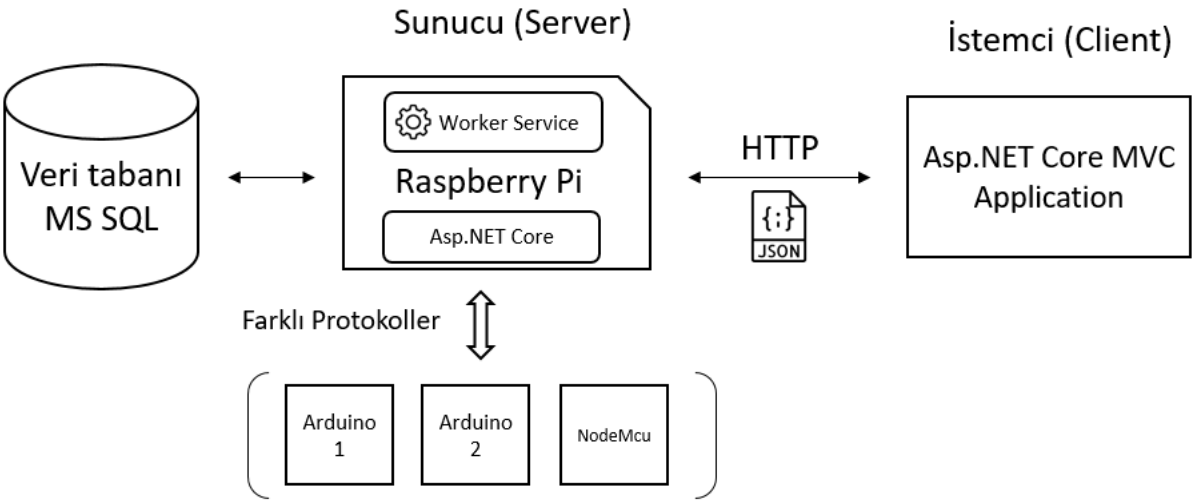
Donanım kartlarından verilerin işlenebilmesi için C ve Python dillerinden yararlanılmıştır. Bunların dışında WoT konsepti için geliştirilen yazılımlar için .NET C# ve JavaScript yazılım dilleri ağırlıklı olarak kullanılmıştır. Bölüm 3.2'de WoT konsepti için geliştirilen yazılımlar anlatılmıştır.

3.2 Yazılım

Sensör ve aktüatör bilgileri, WoT'nin yapı taşlarından olan TD ile JSON formatında servis edilmektedir. WoT Gateway, Web uygulamasından ilgili URL adreslerine yapılan HTTP isteklerini karşılamaktadır. Sunucu tarafında API dışında Worker Service ile veri tabanı ve gerçek zamanlı veri aktarım işlemleri yapılmaktadır. Yazılım için yararlanılan araçlar:

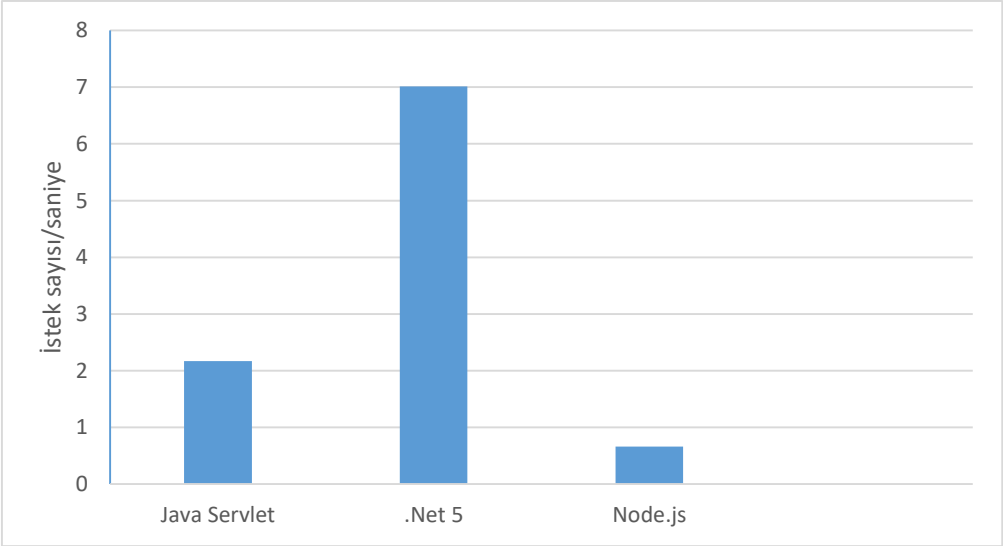
- API arayüzü için Asp.NET Core Web API [71]
- Worker Service için .NET Core Application [72]
- Kimlik doğrulama için Microsoft Identity [73]
- WebSocket sunucusu için SignalR [74]
- Veri tabanı işlemleri için MS SQL ve EntityFramework [75]
- API belgeleri için Swagger [76]

Şekil 3.6'da sistemin sunucu istemci çalışma şeması gösterilmiştir.



Şekil 3.6 Sunucu-istemci şeması

Uygulamanın merkezinde olan Raspberry Pi içerisinde Linux işletim sistemi çalışmaktadır. Açık kaynaklı, yüksek performanslı ve çeşitli uygulamalar ile entegrasyonu kolay olduğu için uygulamada yer alan yazılımlar .NET ile geliştirilmiştir. .NET Core, Windows, Linux ve macOS işletim sistemleri için ücretsiz, hafif, bellek kullanımı düşük, hızlı bir bilgisayar yazılımı çerçevesidir. MIT Lisansı altında yayınlanmıştır ve Microsoft tarafından .NET Foundation aracılığıyla geliştirilmektedir. Yapılan performans testlerinde .NET 5'in Node.js, Java gibi alternatiflerinden daha performanslı olduğu Şekil 3.7'de gösterilmiştir [77].



Şekil 3.7 .NET 5 performans karşılaştırması

3.2.1 REST API

Çalışma kapsamında Visual Studio ortamında Asp.NET Core Web API projesi ile Rest API hazırlanmıştır. Her nesne (sensör veya aktüatör) için HTTP isteklerini karşılayacak kontrolör sınıfı C# dili ile oluşturulmuştur. Ana URL adresi için oluşturulan kontrolör içerisinde tüm nesnelerin bilgilerini jenerik olarak sunulmaktadır. Sensör verilerinin ve aktüatör durum bilgilerinin okunması ve güncellenmesi için sunucu tarafında konfigürasyon dosyalarından yararlanılmaktadır. Nesnelere ait veriler bağlı olduğu mikroişlemciden bu konfigürasyon dosyalarının içerisine yazılmaktadır. HTTP GET isteği ile dosyalardaki ilgili veriler okunurken, HTTP POST isteği ile de dosyalarda yazma işlemi yapılmaktadır.

Aşağıda sistem içerisinde yer alan tüm cihazların TD'leri gösterilmesini sağlayan ana kontrolör içeriği gösterilmektedir.

```
[Authorize]
    [Route("/")]
    [ApiController]
    public class HomeController : ControllerBase
    {
        [HttpGet]
        public IActionResult Index()
        {
            List<Thing> things = ReflectiveEnumerator.GetEnumerableOfType<Thing>();
            var model = ThingModelHelper.GetThingsJsonModel(things);
            return Ok(model);
        }
    }
```

Ana kontrolör sınıfı

Kontrolör sınıfı üzerinde yer alan "Authorize" ile yetki kontrolü yapılmaktadır. Tüm cihazlar yazılım tarafında bir "Thing" sınıfından türetilmektedir. Thing sınıfı içerisinde nesneye ait özellikler, eylemler, olaylar ve cihaz bilgileri yer almaktadır. Sisteme bir cihaz entegre edilmek istendiğinde öncelikle Thing sınıfından kalıtım alınması gerekmektedir. Daha sonra ilgili alanlar doldurulduktan sonra otomatik olarak TD belgesi üretilmektedir. Aşağıda LDR sensörü için oluşturulmuş Thing sınıfı gösterilmiştir.

```
public class LDRSensor : Thing, ISensor
    {
        public override string Id => base.BaseId + Const.LDR.ToLower();
        public override string Title => "LDR Sensor";
        public override string Type => Const.Sensor;
        public override string Description => "LDR is a special type of resistor
that works on the photoconductivity principle means that resistance changes
according to the intensity of light.";
        public override string System => Const.Lighting;

        public override IEnumerable<Property> Properties
        {
            get => AddProp();
            set => base.Properties = value;
        }
...
    }
```

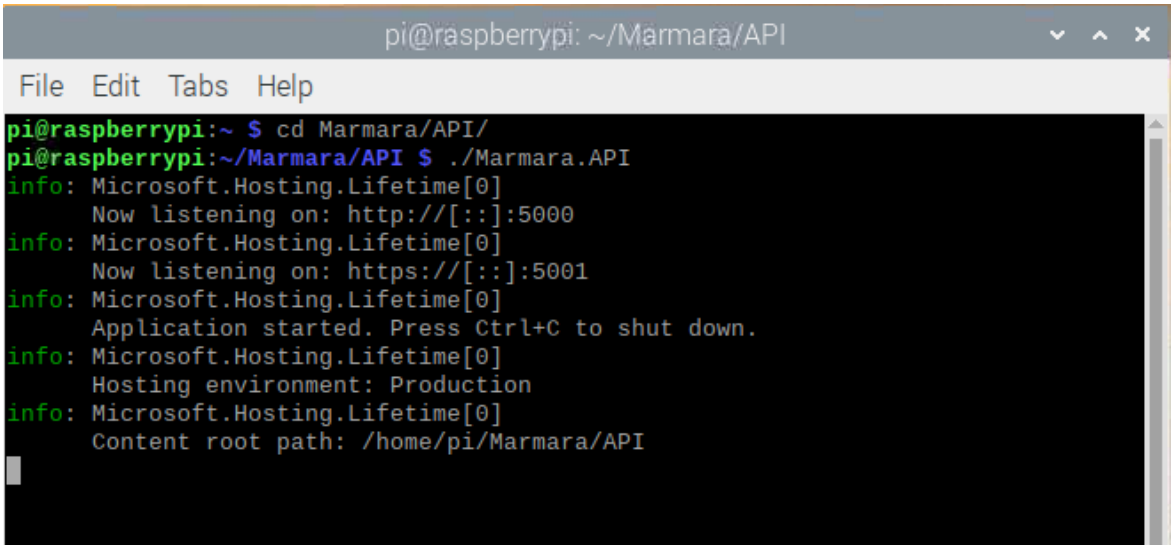
LDR sensörü sınıfı

Tüm nesneler için TD'leri sağlayan kontrolör sınıfları ortak bir kontrolör sınıfından türetilmiştir. Aşağıda temel kontrolör sınıfının içeriği gösterilmiştir.

```
public class ThingBaseController : ControllerBase
  {

      public Thing thing { get; set; }

      [HttpGet]
      public IActionResult Index()
      {
          var model = ThingModelHelper.GetThingJsonModel(thing);
          return Ok(model);
      }

      [HttpPost("actions")]
      public IActionResult actions()
      {
          return Ok(thing.Actions ?? new List<Action>());
      }

      [HttpPost("properties")]
      public IActionResult properties()
      {
          return Ok(thing.Properties ?? new List<Property>());
      }
      [HttpPost("events")]
      public IActionResult events()
      {
          return Ok(thing.Events ?? new List<Event>());
      }
  }
```

Temel nesne kontrolör sınıfı

Şekil 3.8'de REST API'nin Linux komut penceresinden çalıştırılmasındaki görseli verilmiştir.



Şekil 3.8 Terminal ekranında REST API

Veri güvenliğinin sağlanması için JavaScript Object Notation Web Token (JWT) standartı ve Microsoft.Identity kütüphanesi seçilmiştir. Kullanılan kütüphaneler ile birlikte 256 bit uzunlukta çıkış karmasına sahip HmacSha256 algoritması ile veriler şifrelenmiştir. İstemci tarafından API ye erişebilmek için kullanıcı adı ve şifresi, HTTP POST isteği ile sunucuya gönderilir. Girilen bilgiler doğru ise sunucu tarafında anahtar (token) oluşturulur ve istemciye yanıt olarak iletilir. İstemci bundan sonra API üzerindeki URL adreslerine yapacağı tüm HTTP isteklerinde bu token bilgisini kullanması gerekmektedir. Oluşturulan sistemde JWT'ler için kullanım süresi 24 saatlik olarak ayarlanmıştır.

Asp.NET Core içerisinde bulunan "Authorize" özelliği ile nesneler için hazırlanmış kontrolör sınıfları yetkisiz işlemleri engellemektedir. "Authorize" özelliğine ek olarak bazı kontrolör sınıflarına rol özelliği de eklenmiştir. Bu özellik ile de kullanıcılar arası rol bazlı kısıtlamalar yapılabilmektedir. Uygulama veri tabanında, normal ve yönetici olmak üzere iki çeşit rol bulunmaktadır. Yönetici rolüne sahip kullanıcılar, aktüatörleri çalıştırabilir, komut gönderebilir ve yeni kurallar tanımlayabilir. Yönetici rolüne sahip olmayıp sistemde kayıtlı olan normal kullanıcılar ise sensör verilerini görebilir ve alarm bildirimlerini alabilir. Aşağıda TD içerisinde yer alan güvenlik şeması yer almaktadır.

```
public class Security
    {
        public string authorizationUrl { get; set; } = "/api/authenticate/login";
        public string scheme { get; set; } = "bearer";
        public string format { get; set; } = "JWT";
    }
```

TD içerisinde yer alan güvenlik şeması

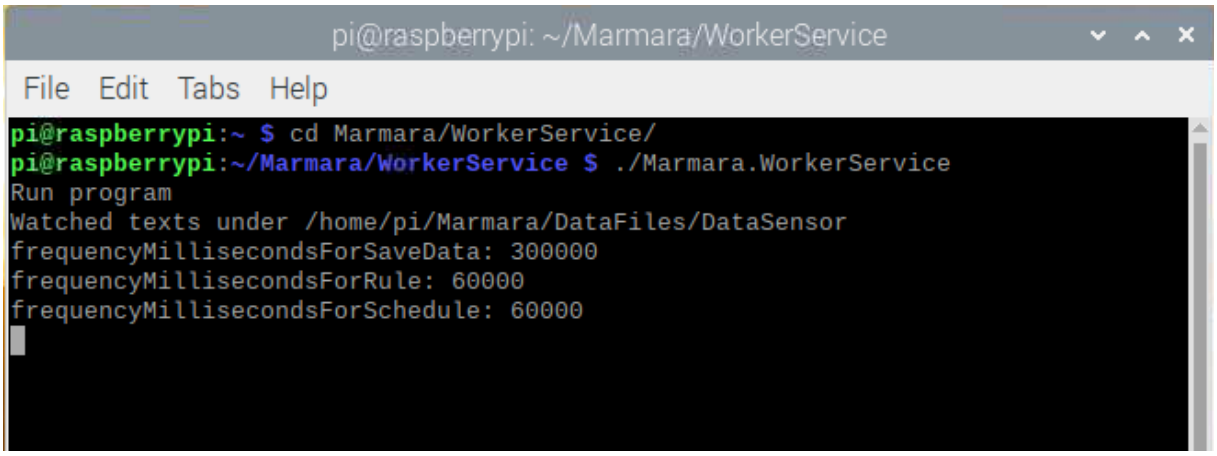
3.2.2 Worker Service

Raspberry Pi içerisinde çalışan çok işlevli bir arka plan uygulaması .NET Core ile geliştirildi. Worker Service uygulamasının görevleri:
- Gerçek zamanlı veri aktarımı
- Verileri bulut ortamına kaydetme
- Planlamaları kontrol etme
- Tanımlanan kuralları gerçekleştirme
- Kullanıcıları gerekli durumlarda mail ile bilgilendirme

Yazma dosyalarında bir değişiklik olduğunda, Worker Service programında değişikliği algılayan kod bloğu aşağıda verilmiştir.

```
private static void OnChanged(object sender, FileSystemEventArgs e)
{
    try
    {
        if (e.ChangeType != WatcherChangeTypes.Changed) return;
            var thingName = Path.GetFileNameWithoutExtension(e.Name);
                RunActivity(thingName);
    }
 ...
}
```

Şekil 3.9'da Worker Service uygulamasının Linux komut penceresinden çalıştırılmasındaki görseli verilmiştir.

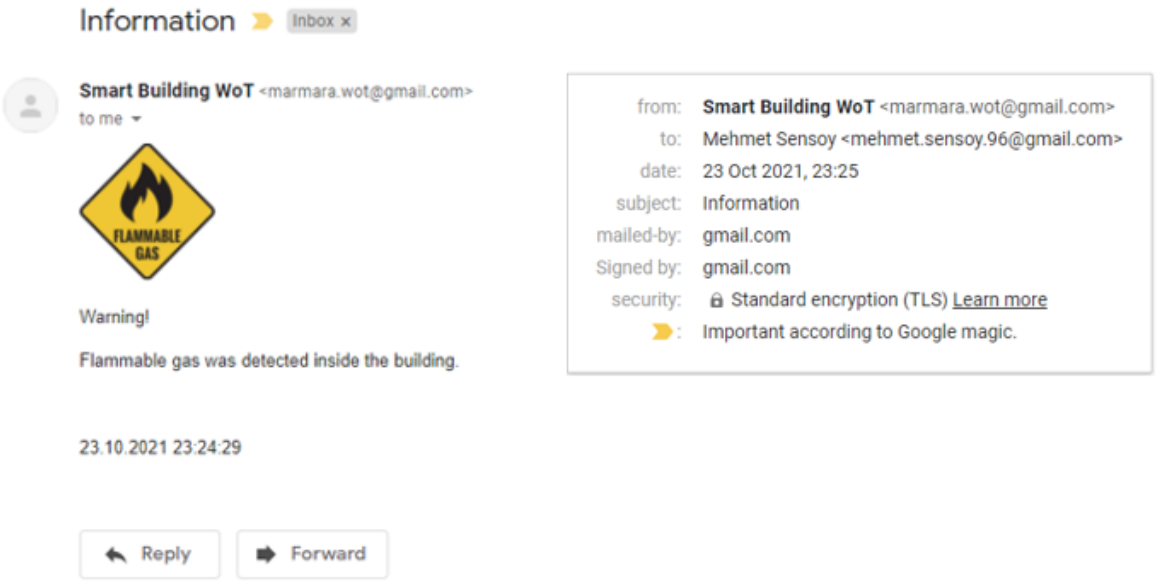


Şekil 3.9 Terminal ekranında Worker Service

Sunucu tarafından nesne verilerini tutan dosyalarda bir güncelleme olduğunda hazırlanan programlar ile bu algılanır ve ilgili nesneye ait Hub URL'e istek atılır. Her ne kadar HTTP ile istek atılmış olsa da isteği karşılayan metot içerisinde SignalR Hub sınıfı çağrılır ve böylece gerçek zamanlı olarak tüm kullanıcılara veriler aktarılır. Gerçek zamanlı veri aktarımı için Microsoft.SignalR kütüphanesinden yararlanılmıştır. Bu kütüphane yardımı ile WebSocket teknolojisinden yararlanılabilmektedir. Uygulamaya WebSocket kullanımının dâhil edilmesiyle birlikte eş zamansız olarak gerçekleşebilecek olaylar meydana gelir gelmez veya bir etkileşim değeri değişir değişmez, manuel yoklama yapma zorunluluğunda kalmadan anında bilgilendirme yapılmaktadır.

Sensör bilgileri belirli periyotlar ile bulut ortamındaki SQL Server 2019 veri tabanına kaydedilmektedir. Verilerin modellenmesi ve kaydedilmesi Microsoft.EntityFramework kütüphanesi ile gerçekleştirilmiştir. Ayrıca Worker Service ile cihazların çalışma saatleri planlanabilmektedir. Örneğin aydınlatmaların 5 saat sonra kapatılması gibi bazı işler planlama ile yapılmak istenebilir. İstemci tarafında HTTP POST ile gönderilen veriler, bu iş için önceden hazırlanmış tabloya kaydedilir. Ardından program periyodik olarak tabloyu kontrol eder ve sırası gelen işlemi gerçekleştirir. Veri tabanında planlama dışında kural tablosu da eklenmiştir. Bu tabloda planlamadakine benzer olarak periyodik olarak kontrol edilmektedir. Örneğin ortamdaki sıcaklığın belirli seviye üstesine çıkmaması için DHT11 sensöründen sıcaklık değeri okunması ve fan kontrol işlemleri gerçekleşmektedir.

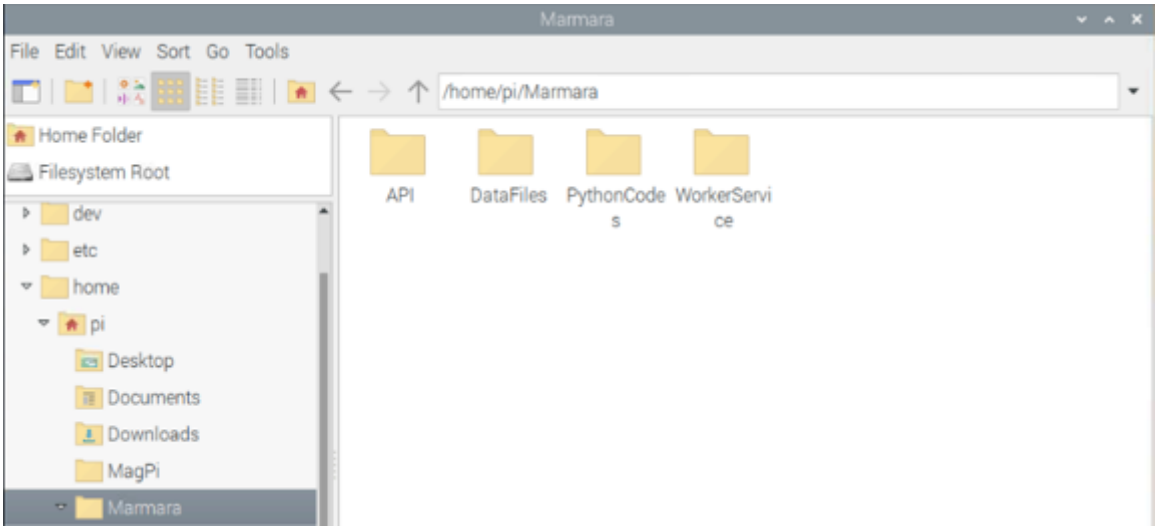
Son olarak kullanıcılara bildirim amaçlı mail gönderimi de yine Worker Service ile yapılmaktadır. Mail gönderimi için System.Net.Mail kütüphanesi kullanılmıştır. Bu kütüphane SMTP ile çalışmaktadır. Birden fazla kullanıcıya aynı anda bilgilendirme yapılabilmektedir. Şekil 3.10'da gönderilen mailin içeriği ve detayları gösterilmiştir.



Şekil 3.10 Mail içeriği

Oluşturulan sisteme yeni bir cihaz veya alt sistem eklendiğinde takip edilmesi gereken adımlar:

1- TD içerisinde görülmesi istenen nesnenin (sensör veya aktüatör) .txt uzantılı dosya ana klasör dizini içerisinde oluşturulur. Şekil 3.11'de ana klasör dizini gösterilmiştir. Bu dosya, yazma ve okuma işlemlerinde kullanılacaktır. WoT Gateway ile haberleşecek cihazlar, veri okuma ve yazma işlemlerini bu dosyalara yazacak şekilde haberleşme sağlanmalıdır.
2- Ardından REST API projesinde nesneye ait kontrolör sınıfı oluşturulur. Bu sınıfın içerisinde yazma ve okuma işlemleri için oluşturulan .txt dosya yolu sınıf metotları içerisinde belirtilir. Nesneye ait özellikler sınıf içerisinde belirtildikten sonra TD belgesi alt yapı ile birlikte otomatik oluşacaktır.
3- Gerçek zamanlı bildirimler için API içerisinde nesneye ait Hub sınıfı oluşturulur. WebSocket, SignalR ile kullanıldığı için istemci tarafında .js uzantılı dosya da güncellenmelidir. WorkerService tarafında veri tabanı işlemleri, kural tanımları, zamanlama işlemleri için de kolayca geliştirme yapılabilir.
4- API kullanacak olan uygulama ara yüzü istenildiği şekilde tasarlanabilir.



Şekil 3.11 Dosya dizini

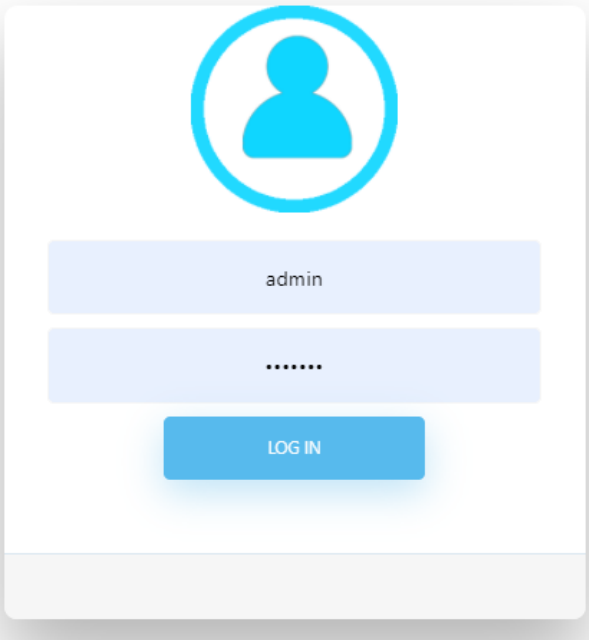
3.3 Web Uygulaması

Sunucu tarafında üretilen API kullanılarak Web sitesi, mobil uygulama veya masaüstü uygulama geliştirilebilir. Bu kapsamda Model-View-Controller (MVC) konseptinde geliştirilen Web uygulaması ile kullanıcılar tarayıcılardan veya akıllı telefonlardan tüm sistemi izleme ve kontrol etme imkânı bulmaktadır. Web uygulaması için kullanılan araçlar:

- Asp.NET Core MVC [78]
- Javascript (jQuery) [79]
- ChartJS [80]
- Bootstrap Kütüphanesi [81]

Responsive olarak tasarlanan Web sitesi mobil cihazlar ile de sorunsuz olarak çalışmaktadır. SignalR ile birden fazla kullanıcı sayfayı yenilemeden, gerçek zamanlı olarak verileri görebilmektedir. Alarm durumunda ise uygulama üzerinden kullanıcılar görsel ve sesli olarak bilgilendirilmektedir. Web sayfasının sol tarafında sistemler için özel sayfa yönlendirmeleri, planlama ve kural sayfaları yer almaktadır.

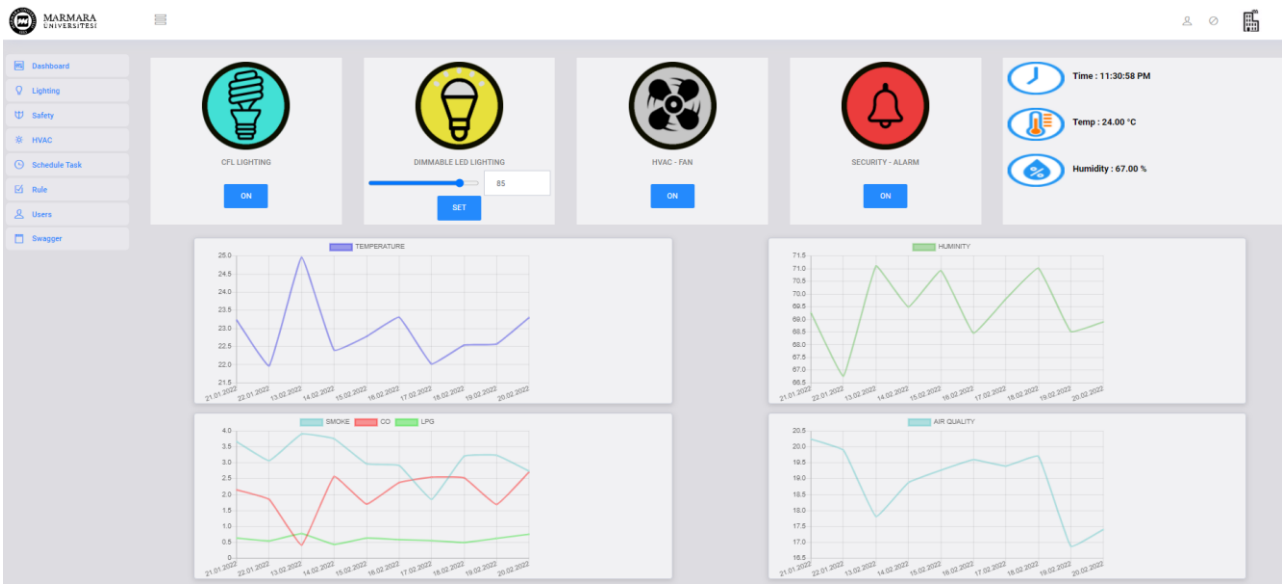
Kullanıcılar, Web uygulamasını kullanmak istediklerinde ilk olarak giriş yapmaları gerekmektedir. Şekil 3.12'de giriş ekranı gösterilmiştir. Başarılı bir şekilde giriş işleminin yapılması ile sunucu tarafından GET ve POST işlemlerinde kullanılması için jeton (token) üretilir. Bu token içerisinde kullanıcının giriş bilgileri ve sahip olduğu roller bulunmaktadır.



Şekil 3.12 Giriş ekranı

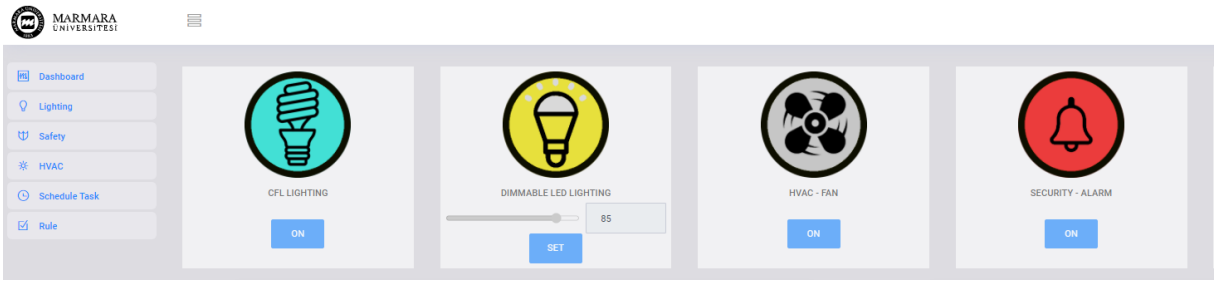
Siteye giriş yaptıktan sonra kullanıcıları Şekil 3.13'teki ana sayfa ekranı karşılamaktadır. Bu ekranın üst kısımda aktüatörler için kontrol butonları, alt kısımda sensörlerden elde edilen verilerin gösterildiği grafikler ve sol kısımda Web uygulamasındaki diğer sayfalara gidilmesi için bir menü yer almaktadır. Ana sayfada yer alan kontrol butonları ve işlevleri:

- CFL Lighting: Aydınlatma sisteminde yer alan lambanın açılıp kapanmasını sağlamaktadır.
- Dimmable LED Lighting: Bu buton sayesinde aydınlatma sisteminde yer alan LED lamba parlaklığı istenilen şekilde ayarlanmaktadır. Metin kutusuna girilen değer 0 ile 100 arasında olabilir.
- HVAC - Fan: HVAC sistemi için eklenen fanın kontrol edilmesini sağlar.
- Security – Alarm: Bu buton ile güvenlik sistemindeki alarm çalıştırılarak kullanıcılara sesli bilgilendirme yapılabilir. Ayrıca ortamda sensörler aracılığı ile alev algılanırsa bu buton Web sitesi üzerinde yanıp sönerek kullanıcılara sesli uyarı yapabilmektedir.



Şekil 3.13 Web uygulamasının ana sayfası

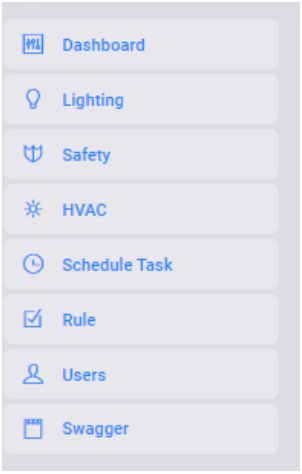
Kontrol butonları ancak yönetici rolüne sahip kullanıcılar tarafından kullanılabilmektedir. Giriş yapan kullanıcı eğer yönetici rolüne sahip değilse kontrol butonları Şekil 3.14'teki gibi pasif gözükecektir ve kullanılamayacaktır. Ancak ekrandaki diğer sensör verilerini ve alarm durumunda bilgilendirme butonundan gelen sinyali görebileceklerdir.



Şekil 3.14 Kontrol butonlarının pasif durumu

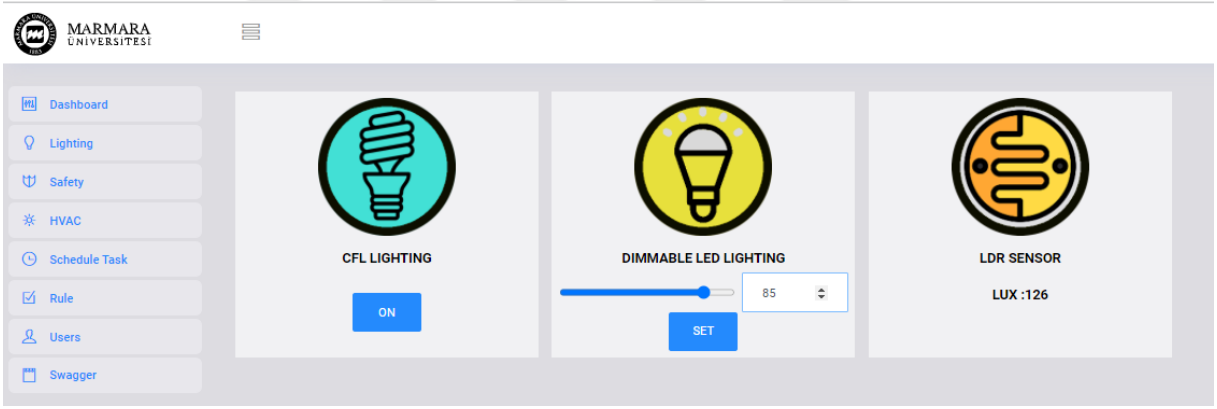
Şekil 3.15'te ana sayfanın sol kısmında yer alan menü ile uygulama içerisindeki diğer Web sayfalarına gidilebilmektedir. Menü içerisinde yer alan butonlar ve yönlendirdikleri sayfalar:

- Dashbord: Ana sayfa
- Lighting: Aydınlatma sistemi için tasarlanan sayfa
- Safety: Güvenlik sistemi için tasarlanan sayfa
- HVAC: HVAC sistemi için tasarlanan sayfa
- Schedule Task: Cihazların çalışmasının planlandığı sayfa
- Rule: Sistem için tanımlanan kuralların bulunduğu sayfa
- Users: Kullanıcı listesinin bulunduğu sayfa (sadece yönetici tarafından görülebilir)
- Swagger: API dokümanın detaylarının bulunduğu sayfa (sadece yönetici tarafından görülebilir)



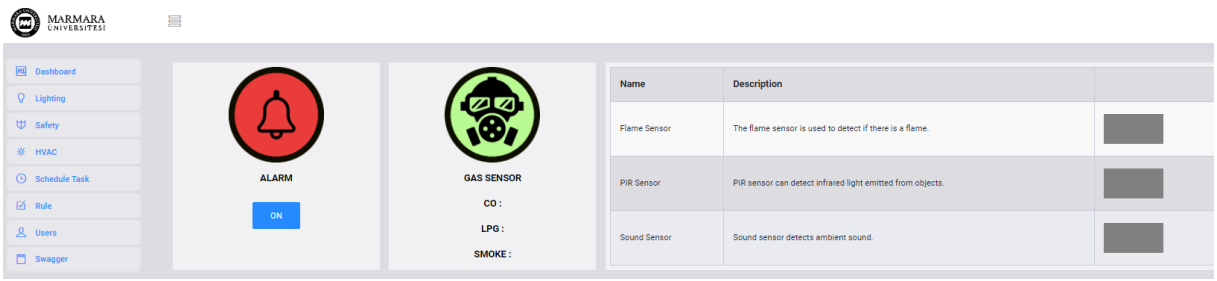
Şekil 3.15 Sayfa menüsü

Şekil 3.16'da yer alan görselde aydınlatma sistemi için tasarlanan sayfa gösterilmektedir. Sayfa üzerinde kontrol butonları bulunmaktadır. Ayrıca ışık sensörü ile okunan değer anlık olarak bu sayfada gösterilmektedir.



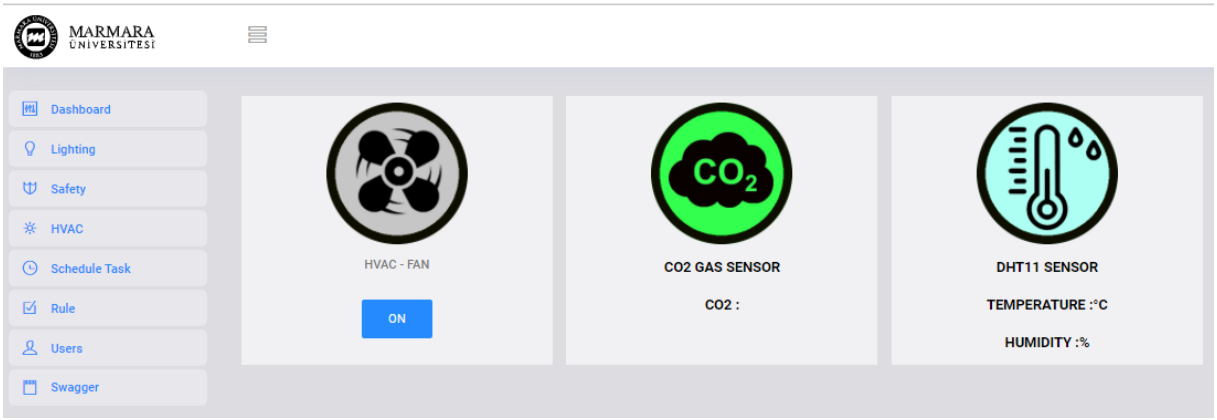
Şekil 3.16 Aydınlatma sistemi Web sayfası

Şekil 3.17'de gösterilen güvenlik sistemi için tasarlanan sayfada alarm butonu, gaz sensöründe okunan değerin anlık olarak gösterildiği bir metin alanı, yangın, hareket ve ses sensöründe anlık değerlerin gösterildiği bir alan yer almaktadır.



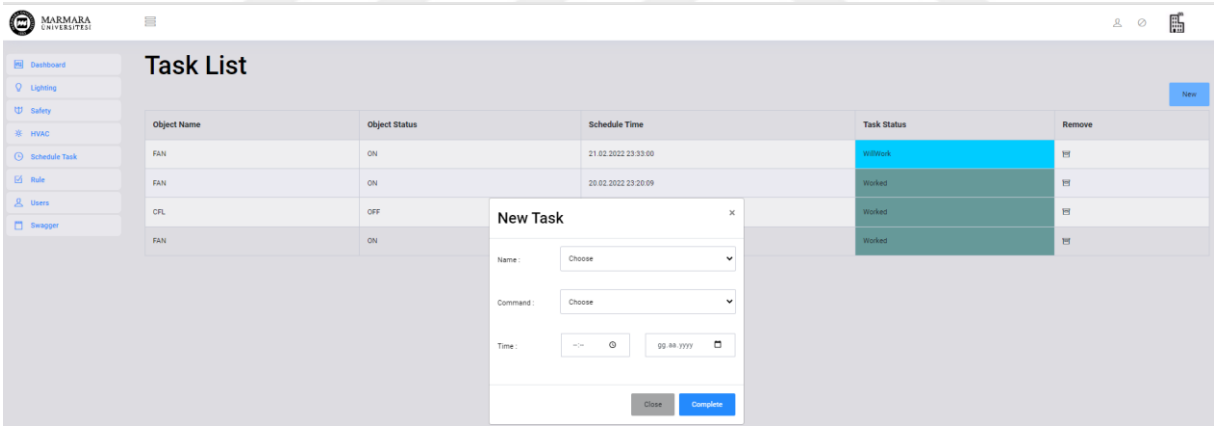
Şekil 3.17 Güvenlik sistemi Web sayfası

Şekil 3.18'de gösterilen HVAC sistemi için tasarlanan sayfada fan kontrol butonu, sıcaklık - nem sensör ile CO_2 sensörü verilerinin gösterildiği alanlar yer almaktadır.



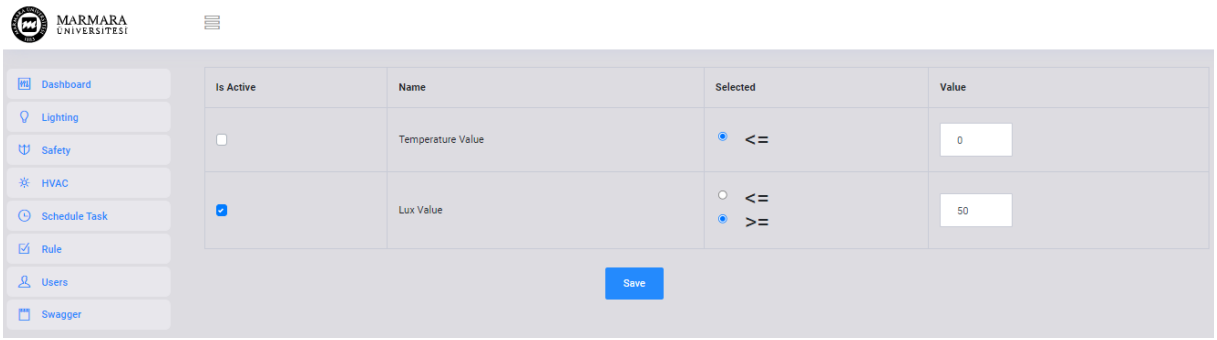
Şekil 3.18 HVAC sistemi Web sayfası

Şekil 3.19'da aktüatörlerin planlanan zamanda çalıştırılabilmesi için oluşturulan sayfa gösterilmiştir. Ekrandaki "New" butonuna tıklandıktan sonra açılan pencere üzerinde seçilen aktüatör ve çalışma durumu için veri tabanında bir kayıt oluşturulmaktadır. İlgili aktüatörün çalışma zamanı geldiğinde, gerekli komutlar aktüatöre iletilmektedir.



Şekil 3.19 Planlama Web sayfası

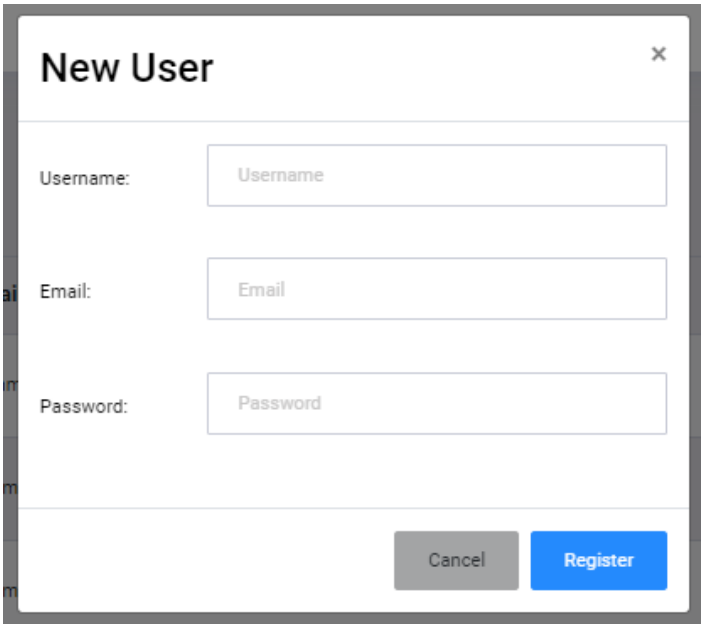
Şekil 3.20'de sıcaklık ve ışık için tanımlanabilecek kuralların bulunduğu sayfanın görseli bulunmaktadır. İstenilirse sıcaklığın bir değer altında tutulması için ekrandan seçim yapılarak kural tanımlanabilir. Sıcaklık, kural ile tanımlanmış seviyenin üstüne çıktığı zaman HVAC sisteminde yer alan fan otomatik olarak çalıştırılır. Benzer şekilde ortamdaki aydınlık düzeyi içinde kural tanımlanabilir. Tanımlanan kurala göre ışık seviyesi için LED otomatik olarak dim edilebilmektedir.



Şekil 3.20 Kural tanımlama Web sayfası

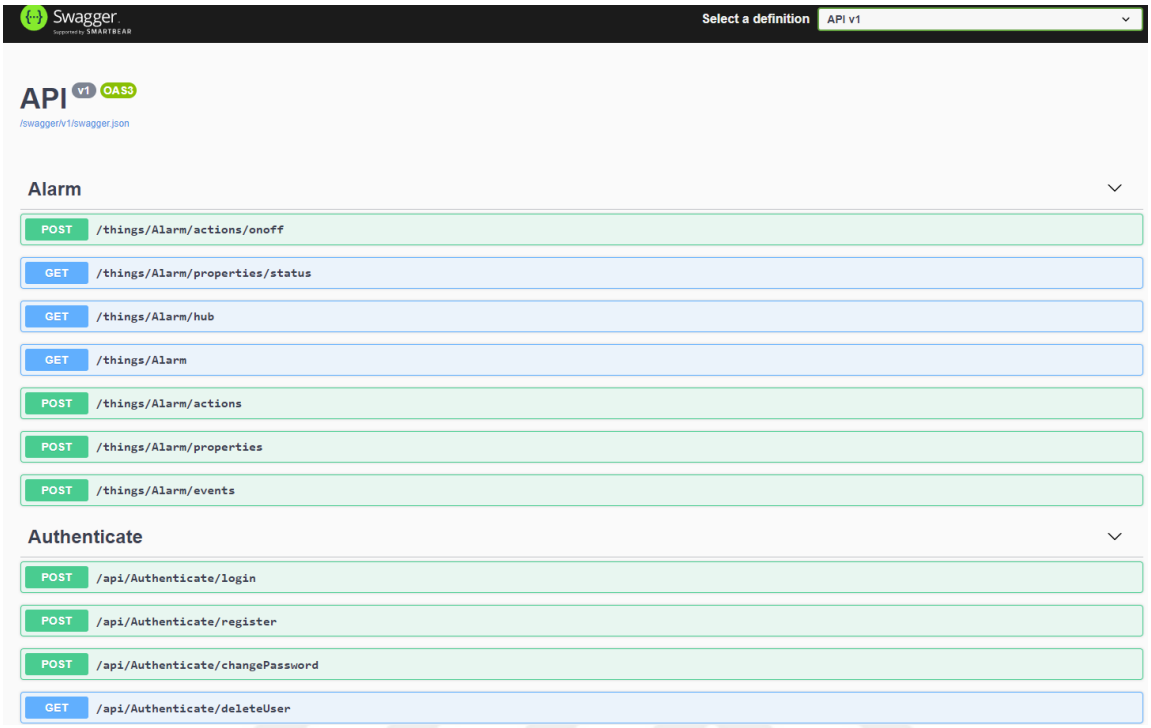
Şekil 3.21'de yönetici tarafından görülebilen sistemde kayıtlı kullanıcı listesinin yer aldığı sayfa bulunmaktadır. Bu sayfada sağ üst buton ile yeni bir kullanıcı sisteme kayıt edilebilir. Şekil 3.22'de yeni kullanıcı kayıt penceresi gösterilmektedir. Sayfada yer alan liste üzerindeki silme butonları ile kullanıcılar sistemden çıkartılabilir.

Şekil 3.21 Kullanıcı listesi Web sayfası



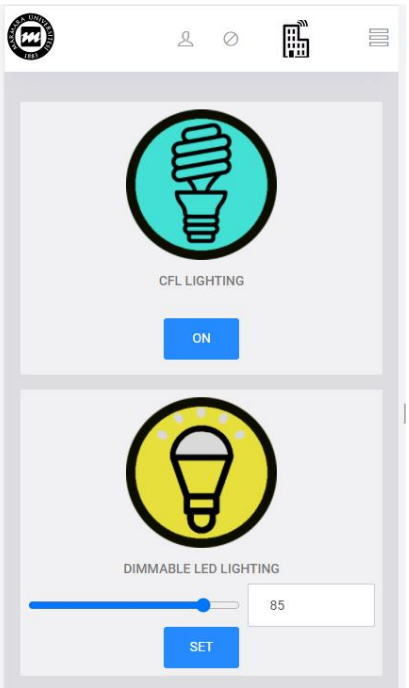
Şekil 3.22 Kullanıcı kayıt penceresi

Şekil 3.23'te yönetici tarafından görülebilen API dokümanı bulunmaktadır. WoT Gateway tarafından sağlanan ve kullanılabilecek tüm URL adreslerini içeren belge HTTP metotları ve örnekleri ile sunulmaktadır.



Şekil 3.23 Swagger dokümanı

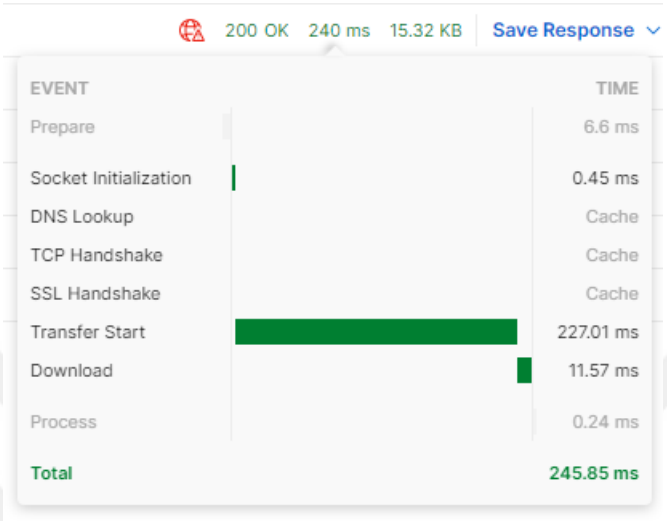
Şekil 3.24'te Web sitesinin akıllı telefondan giriş yapıldığındaki ana sayfa gösterilmiştir. Responsive olarak tasarlanmıştır.



Şekil 3.24 Web sitesinin mobil cihaz görünümü

4. BULGULAR

Uygulama çerçevesinde işlevsellikleri doğrulamak için ilgili testler yapıldı. WoT Gateway testleri için Postman uygulaması [82] kullanılarak cihaz durum bilgisi alındı ve kontrolü sağlandı. Şekil 4.1'de Postman uygulamasından atılan isteğin detayları gösterilmiştir.



Şekil 4.1 İstek detay bilgileri

Tablo 4.1'de ilgili URL adreslerine yapılan isteklerin yanıt süreleri ve alınan dosya boyutları gösterilmiştir. Yerel ağ üzerinde yapılan testlerde, alınan dosya boyutları sabit iken İnternet hızına göre yanıt süreleri değişebilmektedir. GET isteği ile sensörlerin ve aktüatörlerin tüm bilgilerini tutan Nesne Açıklaması (TD) bilgileri alınmıştır.

Tablo 4.1 GET isteklerine ait yanıt detayları

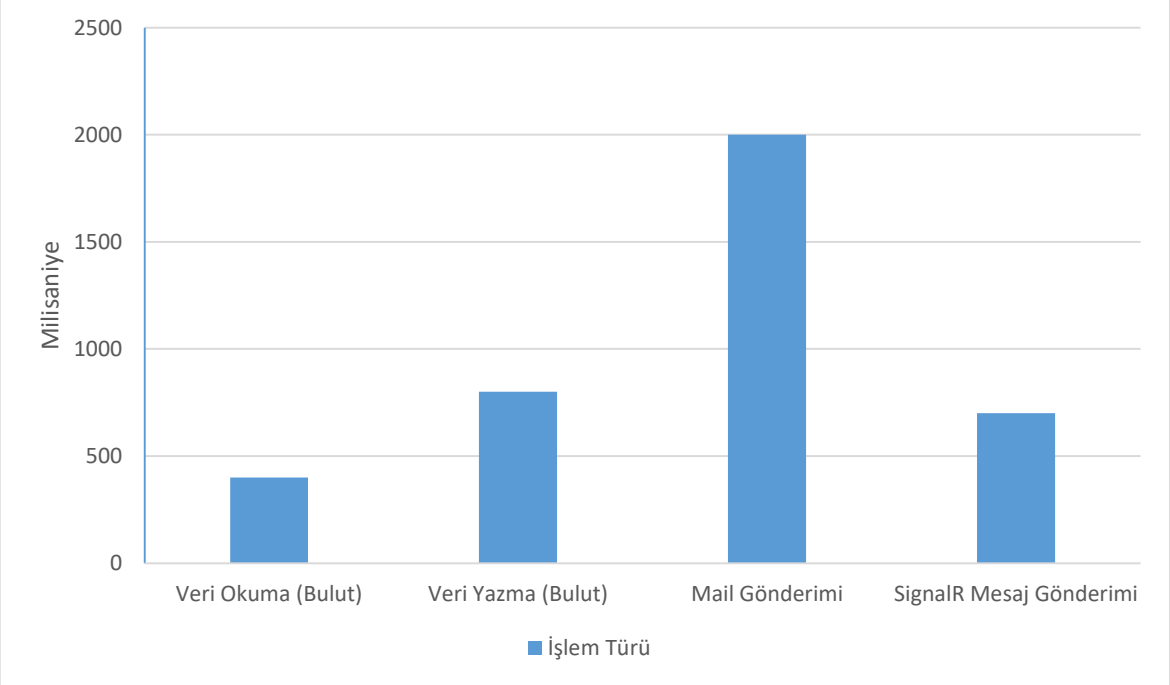
Yapılan İstek	**Minimum yanıt süresi (ms)**	**Maksimum yanıt süresi (ms)**	**Byte**
Tüm Nesne Açıklamaları	143	245	14.62k
Kompakt floresan lamba	39	104	1.38k
Dim edilebilir LED	59	199	1.72k
LDR sensörü	40	121	1.16k
Fan	35	112	1.05k
MQ135 sensörü	42	135	1.34k
DHT11 sensörü	47	140	1.44k
MQ2 sensörü	60	201	1.79k
Flame sensörü	35	115	1.09k
Ses sensörü	35	113	1.08k
PIR sensörü	36	113	1.08k
Alarm	42	138	1.36k

Tablo 4.2'de POST isteği yanıt süreleri gösterilmiştir. POST metodu ile yapılan isteklerde yanıt olarak sadece başarılı ya da başarısız sonucu döndürüldüğü için GET metoduna göre çok daha kısa yanıt süresi bulunmaktadır.

Tablo 4.2 POST isteklerine ait yanıt detayları

Yapılan İstek	**Minimum yanıt süresi (ms)**	**Maksimum yanıt süresi (ms)**	**Byte**
Kompakt floresan lamba	8	32	99
Dim edilebilir LED	11	34	214
Fan	8	32	99
Alarm	8	32	99

Öte yandan oluşturulan Worker Service programını değerlendirmek için testler gerçekleştirilmiştir. Yanıcı gaz tespit edildiğinde anlık olarak mail gönderimi gözlemlenmiştir. Aktüatörler için ayarlanan çalışma zamanlarında dosyalara ilgili komutun yazılması ve kural tablosunda bulunan şartların kontrol edilerek basit hesaplamalar sonucu ilgili dosyalarda güncellenme işemleri gözlemlenmiştir. Şekil 4.2'de Worker Service uygulamasında yapılan işlemler için ortalama gecikme süreleri gösterilmiştir.



Şekil 4.2 İşlemlerin gecikme süreleri

5. SONUÇ

Bu çalışmada, IoT projelerinde Web teknolojilerinin kullanımı ile farklı haberleşme standartlarının bir arada çalışabilirliği gösterilmiştir. WoT ile tasarlanan uygulamalar, yeni geliştirmelere açık ve yeniden programlamaya uygun olduğundan IoT projeleri için büyük önem arz etmektedir. WoT, yeniden programlama için ciddi çaba ve yüksek programlama tecrübesi gerektirmediğinden dolayı maliyet açısından da büyük avantaj sağlayabilir. Bununla birlikte uygulama katmanında Web protokollerinin kullanılması ile veri iletiminde bir miktar gecikme meydana gelmektedir. Ancak bu gecikme kullanıcı deneyimini etkilemediği için birçok uygulama senaryosu için yeterlidir. Web'in haberleşme için kullanılmasında dikkat edilmesi gereken bir konu da veri güvenliğinin sağlanmasıdır. Çalışmada kullanılan kütüphane ve araçlarla veriler şifrelenmiştir. Sonuç olarak yapılan çalışma ile .NET teknolojilerinden faydalanılarak farklı protokolleri kullanan modüler, gelişime açık ve veri güvenliğini sağlayan bir sistem başarılı bir şekilde oluşturulmuştur. Bu tez kapsamında sunulan basit ve esnek yazılım mimarisi sadece akıllı binalar için değil farklı senaryolar içinde uygulanabilir. Yapılan çalışmanın daha da geliştirilmesi için kullanılan HTTP protokolüne ek olarak CoAP gibi farklı Web teknolojileri de sisteme dâhil edilebilir. Bu çalışmada oluşturulan sistem farklı senaryolarda süreçlerin verimliliğini arttırmak için uygulanabilir. WoT, akıllı binalar arasında haberleşmeye imkân sağlayabilir ve böylelikle akıllı şehirlerin gelişmesine de katkı sağlayabilir. Ayrıca çok büyük bir geliştirici kitlesine sahip olan Web'in hızlı bir şekilde gelişmesi IoT alanında da gelişmeleri beraberinde getirecektir.

KAYNAKLAR

1. Al-Sarawi, S., Anbar, M., Alieyan, K., & Alzubaidi, M. (2017). Internet of Things (IoT) communication protocols. In 2017 8th International conference on information technology (ICIT) (pp. 685-690). IEEE.
2. Schwab, L., & Durand, A. (2018). Universal Explorer for the Web of Things. Master's thesis. Faculty of Science University of Bern.
3. Snoonian, D. (2003). Smart buildings. IEEE spectrum, 40(8), 18-23.
4. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. IEEE communications surveys & tutorials, 17(4), 2347-2376.
5. İskender, S. K (2019). Akıllı Binalar. Tesisat Mühendisliği, 2019(172), 67 - 74.
6. Al-Qaseemi, S. A., Almulhim, H. A., Almulhim, M. F., & Chaudhry, S. R. (2016). IoT architecture challenges and issues: Lack of standardization. In 2016 Future technologies conference (FTC) (pp. 731-738). IEEE.
7. Trifa, V., Wieland, S., Guinard, D., & Bohnert, T. M. (2009). Design and implementation of a gateway for web-based interaction and management of embedded devices. Submitted to DCOSS, 1-14.
8. Seydoux, N., Drira, K., Hernandez, N., & Monteil, T. (2016). Autonomy through knowledge: how IoT-O supports the management of a connected apartment. In Semantic Web Technologies for the Internet of Things (SWIT). CEUR-WS.
9. Vermesan, O., Friess, P., Guillemin, P., Gusmeroli, S., Sundmaeker, H., Bassi, A., ... & Doody, P. (2011). Internet of things strategic research roadmap. Internet of things-global technological and societal trends, 1(2011), 9-52.
10. Gershenfeld, N., Krikorian, R., & Cohen, D. (2004). The internet of things. Scientific American, 291(4), 76-81.
11. Parwekar, P. (2011). From internet of things towards cloud of things. In 2011 2nd international conference on computer and communication technology (ICCCT-2011) (pp. 329-333). IEEE.
12. Barros, V. A., Junior, S. A., Bruschi, S. M., Monaco, F. J., & Estrella, J. C. (2019). An IoT multi-protocol strategy for the interoperability of distinct communication protocols applied to web of things. In Proceedings of the 25th Brazillian Sym.
13. Tolk, A. (2013). Interoperability, composability, and their implications for distributed simulation: Towards mathematical foundations of simulation interoperability. In 2013 IEEE/ACM 17th International Symposium on Distributed Simulation and Real.
14. Wilde, E. (2007). Putting things to REST.
15. Guinard, D., & Trifa, V. (2009). Towards the web of things: Web mashups for embedded devices. In Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferen.
16. Guinard, D. (2011). A web of things application architecture: Integrating the real-world into the web (Doctoral dissertation, ETH Zurich).
17. Mainetti, L., Mighali, V., & Patrono, L. (2015). A software architecture enabling the web of things. IEEE Internet of Things Journal, 2(6), 445-454.
18. Kamilaris, A. (2013). Enabling smart homes using web technologies.
19. Guinard, D., Trifa, V., & Wilde, E. (2010). A resource oriented architecture for the web of things. In 2010 Internet of Things (IOT) (pp. 1-8). IEEE.
20. Winroth, R. (2020). Evaluating the Next Generation of Building Automation–IoT SmartBuildings.
21. Bahir, K. S. (2017). Arduino Based Smart Home Automation System. Master's Thesis, Yüzüncü Yıl University, Van, Turkey.

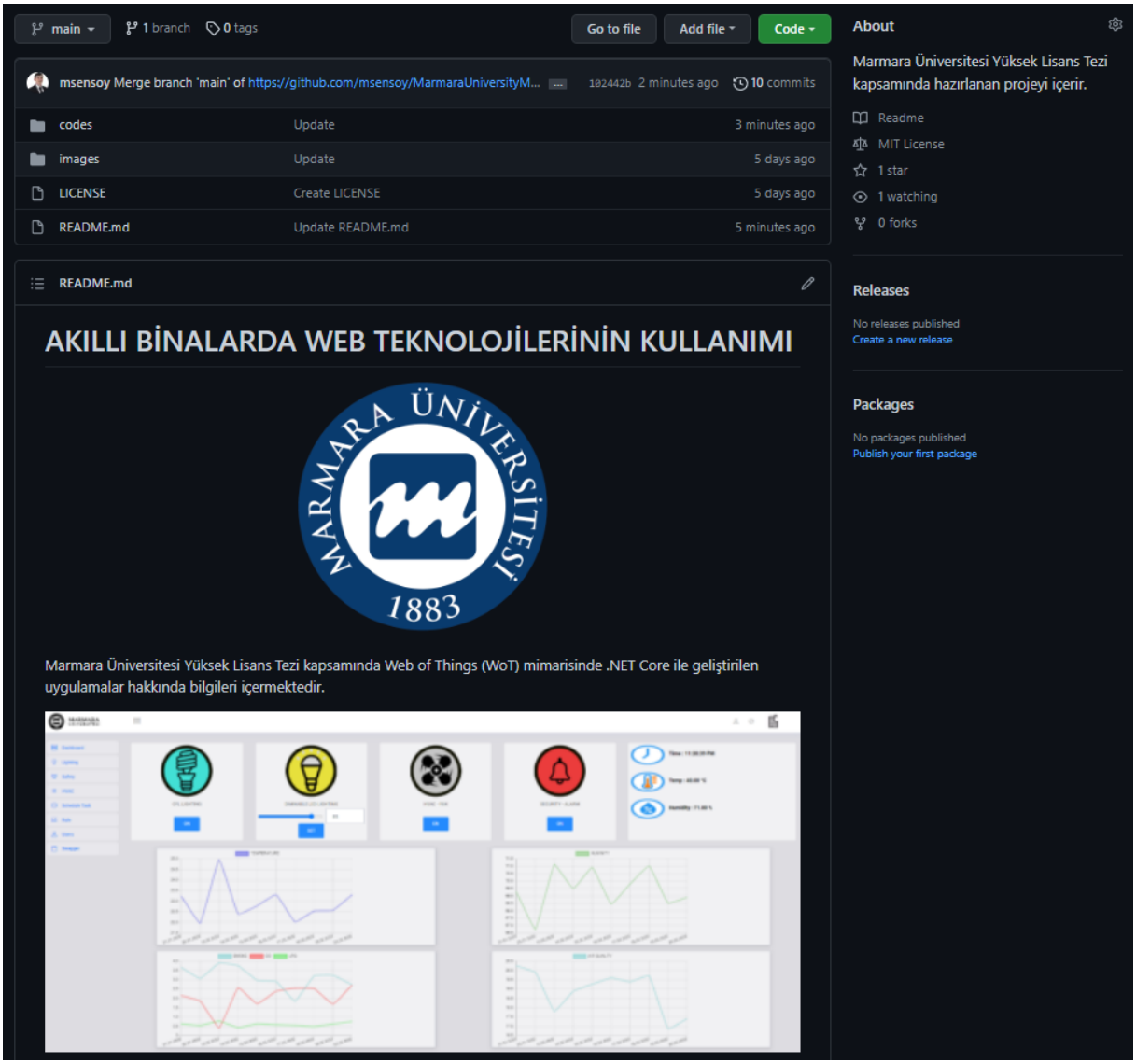
22. Wu, Z., Itälä, T., Tang, T., Zhang, C., Ji, Y., Hämäläinen, M., & Liu, Y. (2012). A web-based two-layered integration framework for smart devices. EURASIP Journal on Wireless Communications and Networking, 2012(1), 1-12.
23. Kadam, A.T. (2016). Web of Things Based Smart Grid Architecture to Monitor and Control Energy Sources. International Journal of Advanced Research in Computer and Communication Engineering.
24. Teriús-Padrón, J. G., Simeoni, E., García-Betances, R. I., Liappas, N., Gaeta, E., Cabrera-Umpiérrez, M. F., & Waldmeyer, M. T. A. (2019). Autonomus air Quality Management System Based on Web of Things Standard Architecture. In 2019 IEEE SmartWorl.
25. Sulistyanto, M. P. T., Febriawan, E., & Sulistiyowati, I. (2021). Web of Things to control planting seeds and watering plants for indoor smart farm. In IOP Conference Series: Materials Science and Engineering (Vol. 1098, No. 5, p. 052109). IOP Publ.
26. Mezenner, I., Bouyakoub, S., & Bouyakoub, F. M. H. (2021). Towards a Web of Things-based system for a smart hospital. In 2020 2nd International Workshop on Human-Centric Smart Environments for Health and Well-being (IHSH) (pp. 22-27). IEEE.
27. Chasta, R., Singh, R., Gehlot, A., Mishra, R. G., & Choudhury, S. (2016). A smart building automation system. International Journal of Smart Home, 10(8), 91-98.
28. Hwang, C. E., Lee, S. H., & Jeong, J. W. (2019). VisKit: Web-based interactive IoT management with deep visual object detection. Journal of Sensor and Actuator Networks, 8(1), 12.
29. Bhawiyuga, A., Pramukantoro, E. S., & Kirana, A. P. (2019). A web of thing middleware for enabling standard web access over BLE based healthcare wearable device. In 2019 IEEE 1st Global Conference on Life Sciences and Technologies (LifeTech).
30. Tiberkak, A., Hentout, A., & Belkhir, A. (2021). Lightweight remote control of distributed web-of-things platforms: First prototype. In 2020 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS) (pp. 103-108). IEEE.
31. Jahn, M., Jarke, M., & Acquaviva, A. (2017). Turning smart buildings into innovation environments: enabling smart building research'in the wild'by conceptualizing user involvement and supporting prototyping (No. RWTH-2017-00772). Fraunhofer-Institut für A.
32. Cayci, F., Callaghan, V., & Clarke, G. (2000). A distributed intelligent building agent language (dibal). In Proceedings of the 6th International Conference on Information Systems, Analysis and Synthesis, Orlando, Florida.
33. Yilmaz, Z. (2006). Akilli binalar ve yenilenebilir enerji. Tesisat Muhendisligi Dergisi,(91), 7-15.
34. King, J., & Perry, C. (2017). Smart buildings: Using smart technology to save energy in existing buildings. Amercian Council for an Energy-Efficient Economy.
35. Khanda, K., Salikhov, D., Gusmanov, K., Mazzara, M., & Mavridis, N. (2017, March). Microservice-based iot for smart buildings. In 2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA) (pp. 302-308). IEEE.
36. Moreno, M. V., Úbeda, B., Skarmeta, A. F., & Zamora, M. A. (2014). How can we tackle energy efficiency in IoT based smart buildings?. Sensors, 14(6), 9582-9614.
37. Jia, R., Jin, B., Jin, M., Zhou, Y., Konstantakopoulos, I. C., Zou, H., ... & Spanos, C. J. (2018). Design automation for smart building systems. Proceedings of the IEEE, 106(9), 1680-1699.
38. Polu, S.K. (2019) Design of a Multi-Sensor based Smart Home System using Artificial Intelligence, International Journal for Innovative Research in Science & Technology.
39. Demir, H., ÇIRACI, G., Reyhan, K., & Ünver, Ü. (2020). Aydinlatmada Enerji Verimliliği: Yalova Üniversitesi Mühendislik Fakültesi Durum Değerlendirmesi. Uludağ University Journal of The Faculty of Engineering, 25(3), 1637-1652.
40. Systems, Solatube. (2012). Construction technologies 3(86)/2012. Solar Lighting.

41. Koldin, A.(2018). Integrated air conditioning systems for buildings based on chilled beams. https://docplayer.com/74012734.
42. Renesas. (2022). Building Security Systems, www.renesas.com/us/application/industrial/ .
43. Hossein Motlagh, N., Khatibi, A., & Aslani, A. (2020). Toward sustainable energy-independent buildings using internet of things. Energies, 13(22), 5954.
44. Bashir, M. R., & Gill, A. Q. (2016). Towards an IoT big data analytics framework: smart buildings systems. In 2016 IEEE 18th International Conference on High-Performance Computing and Communications, IEEE 14th International Conference on Smart C.
45. Guinard, D. D., & Trifa, V. M. (2016). Building the web of things: with examples in node. js and raspberry pi. Simon and Schuster.
46. Kang, W. M., Moon, S. Y., & Park, J. H. (2017). An enhanced security framework for home appliances in smart home. Human-centric Computing and Information Sciences, 7(1), 1-12.
47. Haase, J., Alahmad, M., Nishi, H., Ploennigs, J., & Tsang, K. F. (2016). The IOT mediated built environment: A brief survey. In 2016 IEEE 14th international conference on industrial informatics (INDIN) (pp. 1065-1068). IEEE.
48. Mavromoustakis, C. X., Mastorakis, G., & Batalla, J. M. (Eds.). (2016). Internet of Things (IoT) in 5G mobile technologies (Vol. 8). Springer.
49. Kurt, A. (2005). Akıllı bina sistemlerinin optimum çalıştırılarak enerji tasarrufu elde edilmesi. Yüksek Lisans, Marmara Üniversitesi Fen Bilimleri Enstitüsü, İstanbul, Türkiye.
50. Day, J. D., & Zimmermann, H. (1983). The OSI reference model. Proceedings of the IEEE, 71(12), 1334-1340.
51. Forouzan, B. A. (2002). TCP/IP protocol suite. McGraw-Hill Higher Education.
52. Berners-Lee, T., Fielding, R., & Frystyk, H. (1996). Hypertext transfer protocol--HTTP/1.0.
53. Naylor, D., Finamore, A., Leontiadis, I., Grunenberger, Y., Mellia, M., Munafò, M. & Steenkiste, P. (2014). The cost of the" s" in https. In Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Tech.
54. Fette, I., & Melnikov, A. (2011). The websocket protocol.
55. Groth, D., & Skandier, T. (2005). Network+ study guide. SYBEX Inc..
56. Gratton, D. A. (2013). The handbook of personal area networking technologies and protocols. Cambridge University Press.
57. Hassan, N., Yau, K. L. A., & Wu, C. (2019). Edge computing in 5G: A review. IEEE Access, 7, 127276-127289.
58. W3C. (2020). Web of Things (WoT) Architecture. https://www.w3.org/WoT/ . (30.03.2022). World Wide Web Consortium W3C, Recommendation.
59. Ruby, S., & Richardson, L. (2007). RESTful web services (p. 454). O'Reilly.
60. W3C. (2012). What is the difference between the Web and the Internet?. https://www.w3.org/Help/ (30.03.2022) World Wide Web Consortium, Help and FAQ .
61. McPherson, S. S. (2009). Tim Berners-Lee: Inventor of the World Wide Web. Twenty-First Century Books.
62. Sinha, A. (1992). Client-server computing. Communications of the ACM, 35(7), 77-98.
63. Fielding, R., & Reschke, J. (2013). Hypertext transfer protocol (http/1.1): Authentication. draft-ietf-httpbis-p7-auth-24 (work in progress).
64. Berners-Lee, T., Fielding, R., & Masinter, L. (1998). Uniform resource identifiers (URI): Generic syntax.
65. Crockford, D. (2006). JSON: The fat-free alternative to XML. http://www. json. org/xml. html.
66. W3C. (2017). Web Services Architecture Relationship to the World Wide Web and REST Architectures. https://www.w3.org/TR/ws-arch/ (15.03.2022). World Wide Web Consortium.
67. Fielding, R.T. (2000). "Chapter 5: Representational State Transfer (REST)". Architectural Styles and the Design of Network-based Software Architectures (Ph.D.). University of

California, Irvine.
68. Benslimane, D., Dustdar, S., & Sheth, A. (2008). Services mashups: The new generation of web applications. IEEE Internet Computing, 12(5), 13-15.
69. Fielding, R., & Reschke, J. (2014). Hypertext transfer protocol (http/1.1): Semantics and content.
70. Sahni, N., Bose, J., & Das, K. (2018, September). Web APIs for Internet of Things. In 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (pp. 2175-2181). IEEE.
71. Roth, D., Anderson, R., & Luttin, S. (2021). Introduction to ASP.NET Core, https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-5.0 (09/22/2021). .
72. Troelsen, A., & Japikse, P. (2017). Pro C# 7: With. net and. net Core (Vol. 1328). Apress.
73. Freeman, A. (2014). Getting Started with Identity. In Pro ASP. NET MVC 5 Platform (pp. 297-331). Apress, Berkeley, CA. .
74. Aguilar, J. M. (2014). SignalR Programming in Microsoft ASP. NET (Vol. 40). Microsoft Press.
75. Schwichtenberg, H. (2018). Introducing entity framework core. In Modern Data Access With Entity Framework Core (pp. 1-14). Apress, Berkeley, CA.
76. De, B. (2017). API documentation. In API Management (pp. 59-80). Apress, Berkeley, CA.
77. Microsoft Developer. (2021) .NET 6 deep dive; what's new and what's coming ,YouTube. Retrieved May 27, 2021.
78. Smith, S. (2021) Overview of ASP.NET Core MVC, , from https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-5.0. (10/03/2021).
79. Sulistiono, H., Kom, S., & Kom, M. (2018). Coding Mudah dengan CodeIgniter, JQuery, Bootstrap, dan Datatable. Elex Media Komputindo. .
80. Da Rocha, H. (2019). Learn Chart. js: Create interactive visualizations for the web with chart. js 2. Packt Publishing Ltd. .
81. Shah, M. (2015). Responsive web development using the Twitter Bootstrap framework.
82. Kotecha, K. (2018). API Testing using Postman. https://medium.com/aubergine-solutions/api-testing-using-postman.

EKLER

Proje kapsamında geliştirilen tüm yazılımlar açık kaynaklıdır ve izin verilen MIT lisansı kapsamında lisanslanmıştır. API, Worker Service ve MVC uygulaması kurulumlarının açıklandığı ve kaynak kodlarının indirilebileceği GitHub linki:
https://github.com/msensoy/MarmaraUniversityMasterThesis



ÖZGEÇMİŞ

Mehmet Şensoy, 2018 yılında Sakarya Üniversitesi Mühendislik Fakültesi Elektrik Elektronik Mühendisliği Bölümü'nden mezun oldu. 2019 yılında Bahçeşehir Üniversitesi'nde Web ve Mobil programlama alanında eğitim aldı. Yazılım geliştiricisi olarak Ericsson'da, danışman olarak ta Turkcell bünyesinde aktif olarak çalışmaktadır. 2020 yılında Marmara Üniversitesi Fen Bilimleri Enstitüsü Elektrik Elektronik Mühendisliği Anabilim Dalı Elektrik-Elektronik Mühendisliği Yüksek Lisans programında eğitimine başladı. Aktif olarak IoT ve Web programlama üzerine çalışmalarını sürdürmektedir.