KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

YÜKSEK LİSANS TEZİ

ENDÜSTRİYEL NESNELERİN İNTERNETİ BULUT UYGULAMALARI İÇİN DÜŞÜK MALİYETLİ MODBUS/MQTT AĞ GEÇİDİ TASARIMI VE GERÇEKLEŞTİRİLMESİ

HAMDİ ERDOĞAN

KOCAELİ 2022

KOCAELİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ

ANABİLİM DALI

YÜKSEK LİSANS TEZİ

ENDÜSTRİYEL NESNELERİN İNTERNETİ BULUT UYGULAMALARI İÇİN DÜŞÜK MALİYETLİ MODBUS/ MQTT AĞ GEÇİDİ TASARIMI VE GERÇEKLEŞTİRİLMESİ

HAMDİ ERDOĞAN

Prof.Dr. Kerem KÜÇÜK		
Danışman,	**Kocaeli Üniversitesi**	**.....**
Prof.Dr. Cüneyt BAYILMIŞ		
Jüri Üyesi,	**Sakarya Üniversitesi**	**.....**
Dr.Öğr.Üyesi Alpaslan Burak İNNER		
Jüri Üyesi,	**Kocaeli Üniversitesi**	**.....**

Tezin Savunulduğu Tarih: 05.01.2022

ETİK BEYAN VE ARAŞTIRMA FONU DESTEĞİ

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

- Bu tezin bana ait, özgün bir çalışma olduğunu,
- Çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalarında bilimsel etik ilke ve kurallara uygun davrandığımı,
- Bu çalışma kapsamında elde edilen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi,
- Bu çalışmanın Kocaeli Üniversitesi'nin abone olduğu intihal yazılım programı kullanılarak Fen Bilimleri Enstitüsü'nün belirlemiş olduğu ölçütlere uygun olduğunu,
- Kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- Tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı,

beyan ederim.

☒ Bu tez çalışmasının herhangi bir aşaması hiçbir kurum/kuruluş tarafından maddi/alt yapı desteği ile desteklenmemiştir.

☐ Bu tez çalışması kapsamında üretilen veri ve bilgiler ..... tarafından ..... no'lu proje kapsamında maddi/alt yapı desteği alınarak gerçekleştirilmiştir.

Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçları kabul ettiğimi bildiririm.

.....

Hamdi ERDOĞAN

YAYIMLAMA VE FİKRİ MÜLKİYET HAKLARI

Fen Bilimleri Enstitüsü tarafından onaylanan lisansüstü tezimin tamamını veya herhangi bir kısmını, basılı ve elektronik formatta arşivleme ve aşağıda belirtilen koşullarla kullanıma açma izninin Kocaeli Üniversitesi'ne verdiğimi beyan ederim. Bu izinle Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet haklarım bende kalacak, tezimin tamamının ya da bir bölümünün gelecekteki çalışmalarda (makale, kitap, lisans ve patent vb.) kullanımı bana ait olacaktır.
Tezin kendi özgün çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Tezimde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanılması zorunlu metinlerin yazılı izin alarak kullandığımı ve istenildiğinde suretlerini Üniversiteye teslim etmeyi taahhüt ederim.
Yükseköğretim kurulu tarafından yayınlanan ***"Lisanüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge"*** kapsamında tezim aşağıda belirtilen koşullar haricinde YÖK Ulusal Tez Merkezi/ Kocaeli Üniversitesi Kütüphaneleri Açık Erişim Sisteminde erişime açılır.

☐ Enstitü yönetim kurulu kararı ile tezimin erişime açılması mezuniyet tarihinden itibaren 2 yıl ertelenmiştir.
☐ Enstitü yönetim kurulu gerekçeli kararı ile tezimin erişime açılması mezuniyet tarihinden itibaren 6 ay ertelenmiştir.
☒ Tezim ile ilgili gizlilik kararı verilmemiştir.

.....

Hamdi ERDOĞAN

ÖNSÖZ VE TEŞEKKÜR

Teknolojinin hızla gelişmesi ve internetin yaygınlaşması ile birlikte internete bağlanabilen cihazların sayısında önemli ölçüde artış meydana gelmiştir. Bunun ile birlikte hayatımızı kolaylaştıran cihazlar geliştirilen protokoller vasıtası ile birbirleriyle haberleşebilir hale gelmiştir. Bu doğrultuda Nesnelerin İnterneti (Internet of Things, IoT) kavramı ortaya çıkmıştır. Günlük yaşantımızın hemen hemen her alanında bu teknolojiden yararlanılmaktadır. Nesnelerin İnterneti yalnızca günlük yaşantımızı değil endüstride de kayda değer gelişmelere öncülük etmektedir. Bu alandaki gelişmeler ile birlikte Endüstriyel Nesnelerin İnterneti (Industrial Internet of Things, IIoT) kavramı ortaya çıkmıştır.

Endüstride makinelerin birbirleri ile haberleşmesi, verilerin toplanarak bulut platformlarında depolanması ve raporlanması, oluşturulan büyük eri (Big Data) ile verinin anlamlandırılıp öngörüde bulunulması, yapay zeka (Artificial Intelligence, AI) çalışmaları gibi daha pek çok alanda IIoT'den yararlanılmaktadır. Tüm bu adımların en başında verinin sahadan toplanması işlemi bulunmaktadır. Bu adımda yeri geldiğinde farklı protokollerin bir arada çalışması gerekmektedir. Nesnelerin İnterneti'nde ise farklı protokollerin birlikte çalışması, çözülmesi gereken konulardan biridir.

Bu tez çalışmasında Nesnelerin İnterneti alanında farklı protokollerin bir arada çalışması ve sahadan toplanan verilerin bir ağ geçidi vasıtasıyla bulut platforma aktarılması sağlanmıştır. Endüstriyel cihazların birbirleri arasında haberleşmesinde sıklıkla kullanılan Modbus protokolü ile Nesnelerin İnterneti alanında veri iletiminde yaygın kullanılan MQTT (Message Queuing Telemetry Transport) protokolü bir arada çalışacak şekilde bir ağ geçidi tasarlanmıştır.

Tez çalışması boyunca bana olan desteğini hiçbir zaman eksik etmeyen, karşılaştığım her zorlukta beni yönlendiren ve bu çalışmayı başarılı bir şekilde sonuçlandırabilmem için desteğini hiçbir zaman eksik etmeyen değerli danışmanım Prof. Dr. Kerem KÜÇÜK'e saygılarımı ve teşekkürlerimi bir borç bilirim. Aynı zamanda da eğitim hayatım boyunca her anlamda benden desteğini esirgemeyen sevgili aileme teşekkürlerimi sunarım.

Ocak – 2022 Hamdi ERDOĞAN

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

%	:	Yüzde

Kısaltmalar

AI	:	Artificial Intelligence
AMQP	:	Advanced Message Queuing Protocol (İleri Mesaj Kuyruklama Protokolü)
API	:	Application Programming Interface (Uygulama Programlama Arayüzü)
ASCII	:	American Standard Code For Information Interchange (Bilgi Değişimi İçin Amerikan Standart Kodlama Sistemi)
B2B	:	Business To Business (Şirketten Şirkete)
B2C	:	Business To Consumer (Firmadan Tüketiciye)
BLE	:	Bluetooth Low Energy (Düşük Enerji Tüketimli Bluetooth)
CAN	:	Controller Area Network (Kontrol Alan Ağı Veri yolu)
COM	:	Communication Port (Haberleşme Kapısı)
CRC	:	Cyclic Redundancy Check (Döngüsel Artıklık Denetimi)
DCS	:	Distributed Control System (Dağıtık Kontrol Sistemi)
DDS	:	Data Distribution Service (Veri Dağıtım Hizmeti)
DIP	:	Dual In-Line Package (Çift Sıralı Paket)
EKG	:	Elektrokardiyografi
EPC	:	Electronic Product Code (Elektronik Ürün Kodu)
HTTP	:	Hypertext Transfer Protocol (Hiper Metin Transfer Protokolü)
I2C	:	Inter-Integrated Circuit (Arası Entegre Devre)
IBM	:	International Business Machines (Uluslararası İş Makineleri)
IIoT	:	Industrial Internet of Things (Endüstriyel Nesnelerin İnterneti)
IoT	:	Internet of Things (Nesnelerin İnterneti)
IP	:	Internet Protocol (İnternet Protokolü)
ITU	:	International Telecommunication Union (Uluslararası Telekomünikasyon Birliği)
JSON	:	JavaScript Object Notation (JavaScript Nesne Notasyonu)
KB	:	Kilobyte (Kilobayt)
LCD	:	Liquid Crystal Display (Sıvı Kristal Ekran)
LPWAN	:	Low Power Wide Area Networks (Düşük Güçlü Geniş Alan Ağı)
LRC	:	Longitudinal Redundancy Check (Boyuna Artıklık Kontrolü)
MAC	:	Media Access Control (Ortam Erişim Kontrolü)
MB	:	Megabyte (Megabayt)
MBAP	:	Modbus Application Protocol (Modbus Uygulama Protokolü)
MQTT	:	Message Queuing Telemetry Transport (Mesaj Kuyruk Telemetri Ulaştırma Protokolü)
MQTT-SN	:	Message Queuing Telemetry Transport for Sersor Network
Ms	:	Milisaniye
P&G	:	Procter&Gamble
PDU	:	Protocol Data Unit (Protokol Veri Birimi)
PLC	:	Programmable Logic Controller (Programlanabilir Mantıksal Denetleyici)

PWM	:	Pulse Width Modulation (Sinyal Genişlik Modülasyonu)
QoS	:	Quality of Service (Servis Kalitesi)
RFID	:	Radio Frequency Identification (Radyo Frekansı İle Tanımlama)
RTU	:	Remote Terminal Unit (Uzak Kontrol Cihazı)
SOAP	:	Simple Object Access Protocol (Basit Nesne Erişim Protokolü)
SPI	:	Serial Peripheral Interface (Seri Çevresel Arayüz)
SSL	:	Secure Sockets Layer (Güvenli Giriş Katmanı)
TCP	:	Transmission Control Protocol (Aktarma Kontrol Protokolü)
TEV	:	Transient Earth Voltages (Değişken Toprak Voltajı)
TLS	:	Transport Layer Security (Taşıma Katmanı Güvenliği)
UART	:	Universal Asynchronous Receiver Transmitter (Evrensel Asenkron Alıcı Verici)
uCODE	:	Universal Computer Code
USB	:	Universal Serial Bus (Evrensel Seri Veri yolu)
V	:	Voltage (Voltaj)
Wi-Fi	:	Wireless Fidelity (Kablosuz Bağlantı Alanı)
WSN	:	Wireless Sensor Network (Kablosuz Sensör Ağı)

ENDÜSTRİYEL NESNELERİN İNTERNETİ BULUT UYGULAMALARI İÇİN DÜŞÜK MALİYETLİ MODBUS/MQTT AĞ GEÇİDİ TASARIMI VE GERÇEKLEŞTİRİLMESİ

ÖZET

Günümüzde, endüstride bulut teknolojilerinin kullanılması sahada kullanılan cihaz ve makinelerden elde edilen verilerin gerçek zamanlı olarak takip edilmesinde önemli rol oynamaktadır. Literatürde endüstriyel verilerin bulut platformlarına aktarılmasında birçok farklı ağ geçitlerinden faydalanılmaktadır. Bu tez çalışmasında endüstride tercih edilebilecek düşük kaynak tüketimine sahip, düşük maliyetli Nesnelerin İnterneti (IoT) tabanlı bir ağ geçidi prototipi tasarımı önerilmektedir. Bulut platformunda gerçek zamanlı takip edilecek verilerin sahadan elde edilerek ağ geçidinde toplanması için endüstride geniş kullanıma sahip olan Modbus protokolü kullanılmıştır. Verilerin ağ geçidinde toplandıktan sonra görselleştirilmek için bulut platformuna kablosuz olarak aktarımında IoT'de sıklıkla kullanılan Message Queuing Telemetry Transport (MQTT) protokolü tercih edilmiştir. Önerilen ağ geçidi tasarımı veriler üzerindeki değişimi algılayan bir yazılıma sahiptir. Verilerin bulut platformunda görselleştirilmesi için IoT'ye hızlı bir giriş yapmamıza yardımcı olan IBM Watson IoT Platformu kullanılmıştır. Önerilen ağ geçidi tasarımının performans analizi MQTT sunucusu olarak yerelde çalışan Mosquitto kullanılarak gerçekleştirilmiştir. Performans analizinde deneysel testlerde farklı boyutlarda mesajlar kullanılarak farklı Quality of Service (QoS) seviyelerinde uçtan uca gecikme süreleri ve istatistiksel sonuçları verilmiştir.

Anahtar Kelimeler: Ağ Geçidi, Modbus, Mosquitto, MQTT, Nesnelerin İnterneti (IoT).

DESIGN AND IMPLEMENTATION OF LOW-COST OF MODBUS/MQTT GATEWAY FOR INDUSTRIAL INTERNET OF THINGS CLOUD APPLICATIONS

ABSTRACT

Today, the use of cloud technologies in the industry plays an important role in real-time monitoring of the data obtained from the devices and machines used in the field. In the literature, many different gateways are used to transfer industrial data to cloud platforms. In this thesis, a low-cost Internet of Things (IoT) based gateway prototype design with minimum resource consumption that can be preferred in the industry is proposed. The Modbus protocol, which is widely used in the industry, is used to collect the data to be monitored in real time on the cloud platform from the field and collect it on the gateway. Message Queuing Telemetry Transport (MQTT) protocol, which is frequently used in IoT, is preferred for wireless transmission of data to the cloud platform for visualization after collection at the gateway. The proposed gateway design has software that detects the change on the data. IBM Watson IoT Platform is used to visualize the data on the cloud platform, which helps us get a quick introduction to IoT. The performance analysis of the proposed gateway design has been performed using Mosquitto running locally as the MQTT server. In the performance analysis, end-to-end delay times and statistical results are given at different Quality of Service (QoS) levels by using messages of different sizes in experimental tests.

Keywords: Gateway, Modbus, Mosquitto, MQTT, Internet of Things (IoT).

1. GİRİŞ

Teknolojinin gün geçtikçe gelişim göstererek daha ileriye gitmesi ile birlikte gündelik yaşantımızdaki gerekli ihtiyaçlar ve işlerimizde bizlere kolaylık sağlayacak ve zamandan tasarruf etmemizde yardımcı olacak birbirleri arasında iletişim kurabilen nesnelerin sayısındaki artış hızla devam etmektedir. Bu nesneler çeşitli iletişim protokolleri ve teknolojiler ile internete bağlanabilme yeteneğine sahiptir (Aledhari ve diğ., 2015). Nesnelerin sayısının giderek artması ve birbirleri arasında iletişim kurabilmeleri üzerinde Nesnelerin İnterneti (Internet of Things, IoT) teknolojisi hayatımıza girmiştir (Buyya ve diğ., 2013). Endüstride kullanılan cihazlar ve makineler birbirleri ile haberleşme yeteneğine sahip olması ile Nesnelerin İnterneti'nin kullanım alanları arasına endüstri de katılmıştır. Endüstrideki Nesnelerin İnterneti, Endüstriyel Nesnelerin İnterneti (Industrial Internet of Things, IIoT) olarak bilinmektedir (Gilchrist, 2016).

IIoT, makinelerin kendi aralarındaki iletişimi, toplanan verilerin işlenmesi, üretim ve tüketim alanlarının takibi gibi pek çok alanda maliyet ve zaman anlamında fayda sağlamaktadır. Nesnelerin İnterneti teknolojisinde kritik bir problem olarak farklı iletişim teknolojilerinin birlikte çalışabilirliği olarak ele alınabilir. Modbus protokolü, endüstriyel nesnelerin birbirleri arasında iletişim kurabilmeleri için geliştirilmiş bir iletişim protokolüdür (Deveci, 2016). Modbus, açık kaynak kodlu olması, kolaylıkla öğrenilebilir olması ve birçok endüstriyel cihaza kolay adapte edilebilir olması ile yüksek kullanıma sahiptir. Modbus protokolünde yaygın olarak ASCII (American Standard Code For Information Interchange), RTU (Remote Terminal Unit) ve TCP/IP (Transmission Control Protocol/Internet Protocol) versiyonları kullanılır. Modbus teknolojisi, üreticiye ihtiyaç duymadan aynı ağda bağlı olan diğer Modbus desteğine sahip tüm cihazlarla iletişim sağlanabilmektedir. Endüstrinin gelişimi ile birlikte sahadan elde edilen verilerin gerçek zamanlı olarak izlenebilmesi ve verilerin depolanabilmesi üzerine doğan ihtiyaçlar doğrultusunda bulut teknolojileri ortaya çıkmıştır. Bulut teknolojileri sıklıkla internet tabanlı veri iletişiminde kullanılmak üzere geliştirilen MQTT protokolünü tercih etmektedir (Jadhav ve diğ., 2016). Bulut platformların MQTT protokolünü tercih etmesinde protokolün, minimum kaynak tüketimine sahip olması, TCP/IP tabanlı veri iletimini desteklemesi ve yüksek hızlı veri iletimine sahip olması etkili olmaktadır. IoT tabanlı bir endüstriyel uygulamada farklı protokollerin bir arada çalışabilmesi,

teknolojinin ilerlemesi ve farklı iletişim protokollerinin geliştirilmesi ile kritik bir konu olarak değerlendirilmektedir.

Endüstriyel Nesnelerin İnterneti uygulamalarında ihtiyaç duyulan gereksinimlerin sağlanabilmesinde ağ geçitleri kritik öneme sahiptir (Astarloa ve diğ., 2016). Ağ geçidi, sahadan toplanan verilerin kendi üzerinde toplayarak bulut platformuna aktarılmasında görev alır. Kullanılacak ağ geçidi genele hitap edecek protokol desteğine sahip olmalıdır. Bu ihtiyacın karşılanabilmesinde IoT için MQTT, endüstri için ise Modbus kullanılabilir. Önerilen prototip tasarımının endüstride kullanılabilirliğinin tespit edilmesinde yine endüstride elektriksel kısmi deşarj durumunun oluştuğu alanlarda kullanılan PD Annunciator cihazı temel alınmıştır. Elektriksel kısmi deşarj, iki iletken elektrot arasındaki dielektrik malzemesinin yapısında meydana gelen problemlerden kaynaklanarak, tam bir köprü oluşturulamadığı durumda meydana gelen elektriksel boşalma veya kıvılcımdır (Lundgaard, 1992). Yapılan araştırmalara göre orta ve yüksek gerilimli sistemlerde karşılaşılan sorunların önemli bir kısmı (%80) bahsedilen problemden ortaya çıkmaktadır (IEEE Standart, 2006). Dielektrik malzemenin yapısına zarar verecek orandaki gerilimin olduğu ortamlarda elektriksel kısmi deşarj ortaya çıkabilir. Üzerinde durulan problem nedeniyle karşılaşılan sorunların, IIoT içeren bir teknoloji ile giderilmesi güncel bir problemdir.

Bu tez çalışmasında, IIoT uygulamalarındaki önemli parametreler olarak belirtilen performans, maliyet ve farklı iletişim protokollerinin birlikte çalışabilirliği konusundaki problemler ele alınarak bu problemlerin çözümünde öncü olabilecek ve güncel IoT teknolojilerini destekleyebilecek bir ağ geçidi prototipi geliştirilmesi hedeflenmiştir (Erdoğan ve diğ., 2020). Tasarımın minimum maliyetle geliştirilmesi için Arduino geliştirme kartı tercih edilmiştir. Önerilen ağ geçidi prototipinin performansının geliştirilmesi ve ağ trafiğinin azaltılması için verilerdeki farklılığı tespit edebilecek bir yazılım geliştirilmiştir. Önerilen sistemin IoT platformlar ve endüstri alanında kullanılan cihazlarla ile iletişiminin yüksek bir oranda olması için MQTT ve Modbus protokolleri kullanılmıştır. Prototip, farklı protokoller arası iletişim problemini gidermek için MQTT ve Modbus protokollerinin birbirleri arasındaki veri iletişimi için gerekli olan dönüşümleri sağlamaktadır. Verilerin bulut platformuna aktarılması ve görselleştirilmesi için IBM Watson IoT platformu tercih edilmiştir (IBM, 2015). IBM Watson IoT

Platformu vasıtası ile verilerin gerçek zamanlı takip edilmesi sağlanmıştır. Prototipin performansının test edilmesinde uçtan uca ortalama gecikme süreleri ve gönderilen toplam kayıt (yazmaç) sayısı, yerelde çalışan Mosquitto sunucusu üzerinde test edilmiştir.

MQTT protokolü için iki farklı seviyeli servis kalitesi (Quality of Service, QoS) kullanılmıştır. Veriler Modbus slave (köle) cihazdan okunarak bulut platformuna gönderme sıklığı olarak saniyede bir gönderilecek şekilde ayarlanmıştır. Modbus protokolünde ise slave cihazların sayısı ve her bir iterasyonda okunacak kayıt sayısının performans üzerinde etkili olacağı göz önünde bulundurularak belirtilen parametreler üzerinde farklı varyasyonlar denenerek test sonuçları elde edilmiştir.

2. LİTERATÜR ÖZETİ

Literatüre bakıldığında, ilk gerçekleştirilen IoT ağ geçidi tasarımı çalışması 2013 yılında Dong-kweon ve arkadaşları tarafından gerçekleştirilmiştir (Hong ve diğ., 2013). Shinho ve arkadaşları Nesnelerin İnterneti uygulamalarında sıklıkla tercih edilen MQTT protokolünün performansı hakkında çalışma yapmışlardır. Gerçekleştirdikleri çalışma ile protokol kablolu ve kablosuz olarak iki farklı varyasyonda incelenmiştir. Murat ve arkadaşları (Kuzlu ve diğ., 2018), yapılarda artan enerji tüketiminin nedeniyle tüketilen enerjinin bulut teknolojisi vasıtasıyla takip edilmesi ve kontrol altına alınması için bir IoT ağ geçidi tasarlamışlardır. Phuc ve arkadaşı (Nguyen-Hoang ve Vo-Tan, 2019), endüstride kullanılabilecek bir IoT ağ geçidi geliştirmişlerdir. Tasarlanan ağ geçidi Linux tabanlıdır, farklı endüstriyel saha veri yollarından ve IoT protokollerinden hızlı ve kolay veri alışverişini sağlayabilecek şekilde geliştirmişlerdir. Ghenadie ve arkadaşı (Corotinschi ve Gaitan, 2018), yerel veri işleme yaparak Modbus ağlarını IoT uygulamalarına genişletmeyi hedeflemişlerdir. Geliştirdikleri sistemin performans analizini birden fazla sunucularda test etmiş ve çalışmalarına eklemişlerdir. Yin ve arkadaşları (Ding ve diğ., 2019), Modbus protokolü için yeni bir adapte etme yöntemi üzerine çalışmışlardır. Önerilen yöntemde, Modbus slave cihazına ait veriler ile uygulama verilerini algılama/çalıştırma verileri arasında çift yönlü dönüşüm sağlamaktadır. Andrei ve arkadaşları (Mihu ve diğ., 2017), endüstriyel alanlarda Modbus slave cihazlarından verileri toplayan ZigBee temelli kablosuz veri iletişim sistemi önermişlerdir. Sistemin kablosuz olması, yapılacak olan Modbus sorgularının tüm cihazlara paralel bir şekilde iletilmesi ile sorgulama hızını arttırmaktadır ve sistemin kurulumunu kolaylaştırmaktadır.

Deniz araçlarında yaşanan olumsuzluklara karşı Yaşam ve Kerem tarafından önerilen sistemde, IoT teknolojisinden yararlanılarak deniz araçlarını uzaktan izleme ve kontrol edilebilmesi sağlanmaktadır (Küçük ve Ustaoğlu, 2020). Sistem özellikle deniz araçlarının bataryası ile ilgili problemlere odaklanmaktadır. Batarya durumunun anlık olarak Firebase bulut platformuna aktarmakta ve geliştirilen Android uygulama ile izlenmesi sağlanmaktadır. Ayrıca deniz araçlarının bataryasının şarj durumu güncel olarak takip edilmekte ve otomatik olarak şarj edilebilmesi sistem tarafından kontrol edilebilmektedir.

Brennan ve arkadaşı yaptıkları çalışmada (Crispin ve Shaout, 2018), farklı gömülü sistemler ile iletişim sağlayabilen MQTT temelli bir çalışma sunmuşlardır. Kun ve arkadaşları (Guo ve diğ., 2019), Modbus ve MQTT protokollerinden yararlanarak IIoT uygulamalarında tercih edilebilecek ağ geçidi tasarımını Raspberry Pi kullanarak önermişlerdir. Tasarım saha cihazları ve bulut uygulamaları arasında iki yönlü iletişimi desteklemektedir. Sistemin konfigürasyonu için geliştirilen web uygulaması ile gerçekleştirmektedirler. Gizem ve Mehmet ortaya koydukları araştırma makalesinde (Köse ve Kurutkan, 2021), sağlık hizmetinde Nesnelerin İnterneti uygulamalarını içeren bilimsel yayınların araştırılma eğilimlerini incelemektedirler. 2001-2019 yılları arasında yayınlanan İngilizce makalelerde R programlama dilini kullanarak bibliyometrik araştırmasını ortaya koymuştur. Bu çalışmada R programlama dilinin Biblioshiny paketi kullanılmıştır. Yayınlar çeşitli kategorilere göre değerlendirilmiştir.

Fatma ve arkadaşları IoT'nin inşaat sektöründe, uzun ömürlü ve sürdürülebilir binalar inşa etmek için gün geçtikçe öneminin arttığını vurgulayarak, alçı numunelerinin iç sıcaklık ve nem ölçümlerinin uzaktan takibini amaçlamaktadır (Kılçık ve diğ., 2021). Yapıların uzun ömürlülüğünün sağlanması için ve bakım maliyetlerinin en aza indirilmesinde çeşitli önerilerde bulunmuşlardır. İsmail yaptığı araştırmada (Bütün, 2021) IIoT teknolojisinin güvenlik açısından etkilerini incelemiştir. Bu doğrultuda Modbus, PROFIBUS ve PROFINET gibi ağ teknolojilerini incelemiştir. Ve bu alanda çalışmakta olan ve çalışacak olan araştırmacılara yeni araştırmalar önermiştir. Özkan ve arkadaşları yaptıkları çalışmada (Akın ve diğ., 2021), yüz tanımalı IoT posta kutusu tartışılmaktadır. Hücresel bağlantıyı görüntü işleme ile birleştirerek, kullanıcıya kritik belgelerinin güvenli bir şekilde teslim edilmesini sağlamaktadır. Geliştirilen prototip, verileri işleyen ve GSM modülü üzerinden ağ bağlantısını kuran Arduino Uno'ya bağlı parmak izi okuyucu, kamera, elektromanyetik kilit, küçük LCD (Liquid Crystal Display) ekran, mikrofon ve hoparlörden oluşmaktadır. Ayrıca önceden kaydedilmiş bir görüntü setine dayalı yüz tanıma gerçekleştirmek için kamera Raspberry Pi'ye bağlı ve OpenCV ve Python yazılımı kullanılmaktadır. Mahmut ve Mert yaptıkları çalışmada endüstride IoT teknolojisinden yararlanarak sistemin gerçek zamanlı izlenmesi ve yapay zeka kullanarak arızaların önceden kestirimi üzerinde çalışmıştır (Durgun ve Kışlakçı, 2021). Ayrıca önerdikleri sistem herhangi bir problemden kaynaklı veri iletiminde oluşabilecek veri kaybını önleyecek sınırlı bilişim katmanında saklayarak problem çözüldüğünde verinin tekrar

aktarılmasını sağlayarak veri kaybının önüne geçmektedir. Harun ve arkadaşları çalışmalarında (Dolcel ve diğ., 2021), tarım alanında üretim verimliliğini arttırmak ve ürüne özel sulama sistemi kullanarak su israfını azaltmak için IoT ve makine öğrenmesinden yararlanmıştır. Çalışmanın kapsamı olarak IoT tabanlı, uzaktan kontrol edilebilen ve yapay zekaya dayalı katmanlı bir yapı sunmaktadırlar.

Uğur ve arkadaşları, IoT teknolojisinin günlük hayatta insan hayatı üzerindeki önemli etkisini vurgulayarak aynı zamanda da büyük verinin elde edilmesindeki katkısını belirterek bu doğrultuda IoT hakkında detaylı inceleme yapmışlardır (Demiroğlu ve diğ., 2021). IoT çekirdek bileşenleri, katmanları ve saldırı türleri konularını ele almışlardır. Fatih ve Kadir yaptıkları çalışmada (Baz ve Uludağ, 2021), donanım olarak Raspberry Pi, Arduino kartları ve çeşitli sensörlerden yararlanarak açık kaynak kodlu veri merkezi izleme üzerinde çalışmışlardır. Geliştirdikleri sistem ile veri merkezi takibi için gerekli ihtiyaçları karşılamaktadır.

Nesnelerin İnterneti teknolojisinin kullanıldığı alanlardan biri de eğitimdir. Abdulkerim ve arkadaşları yaptıkları araştırmada (Aydın ve diğ., 2021), Nesnelerin İnterneti teknolojisinin eğitimde kullanıldığı alanları tespit ederek etkisi incelenmiştir. Çalışmada doküman incelemesi yöntemi ile “Web of Science” ve “ERIC” veri tabanlarında eğitim ve Nesnelerin İnterneti anahtar kelimeleri İngilizce olarak tarama yapılmıştır. Neticede 2020’ye kadar 748 çalışmaya ulaşılmıştır. Bu çalışmada 131 tanesi hariç diğerleri gerekli kriterlere uymadığı için elenmiştir. Çalışma sonucunda mühendislik eğitimi ve laboratuvar uygulamaları gibi bir çok alanda kullanıldığı tespit edilmiştir.

Seyfullah ve Fırat yaptıkları çalışmada (Arslan ve Aydemir, 2021), sağlık alanında IoT teknolojisinden yararlanarak koronavirüs salgını döneminde çocukların mikroplardan korunmaları için eve geldiklerinde ellerini yıkayıp yıkamadıklarını takip edebilmek için bulut tabanlı bir sistem geliştirilmiştir. Sistem iki farklı düğümden gelen verilere göre ebeveynleri bilgilendirmektedir. İlk düğüm ile çocuk eve girdiğinde kimlik tanıması yapılır. İkinci düğümle ise lavaboya gelen çocuğun ellerini yıkayıp yıkamadığı kimlik tespiti ve ses tanıma ile kontrol edilmekte. El yıkama işlemi gerçekleştirilmediğinde ebeveynler bilgilendirilir. Mustafa ve arkadaşları çalışmalarında (Akkaş ve diğ., 2021), Nesnelerin İnterneti teknolojisinden yararlanarak hastanelerde kullanılan EKG (Elektrokardiyografi) cihazı ile monitör arasındaki kabloları ortadan kaldırarak kablosuz

elektrokardiyogram cihazı tasarımını önermiştir. Bu çalışma vasıtasıyla Covid-19 salgını süresince hastaların hastanelere gitmesine gerek kalmadan uzaktan izlenebilirliği sağlanmıştır.

Bu tez çalışmasında literatürdeki diğer çalışmalardan farklı olarak Arduino temelli olması ve bu sayede maliyeti düşüktür. Çalışmada yazılım ihtiyaçları göz önünde bulundurularak maliyetin düşük olması için iki farklı Arduino geliştirme kartı kullanılmıştır. Wi-Fi (Wireless Fidelity) tabanlı Arduino Mega kartı ağ geçidi olarak kullanılmıştır. PD Annunciator cihazı slave olarak görev yapmakta ve Arduino Mega kartı tarafından temsil edilmektedir. MQTT üzerinden bulut platformuna aktarılan veriler, farklı hassasiyetlerde gönderilerek veriler üzerindeki değişimi algılayabilen ağ trafiğine fazla yük getirmeyen yazılım tarafından aktarılmaktadır. Toplanan veriler bulut platformunda görselleştirilmesinde iletim yönü tek yönlüdür. Endüstride kullanılan PD Annunciator cihazından elde edilen veri setindeki veriler Arduino Mega kartından okunarak bulut platformda görselleştirilmek için IBM Watson IoT Platformu tercih edilmiştir. Tasarlanan ağ geçidi iki farklı protokol arasında dönüşüm işlemini gerçekleştirerek farklı protokollerin birlikte çalışmasını sağlamaktadır.

3. KULLANILAN MATERYALLER VE YÖNTEMLER

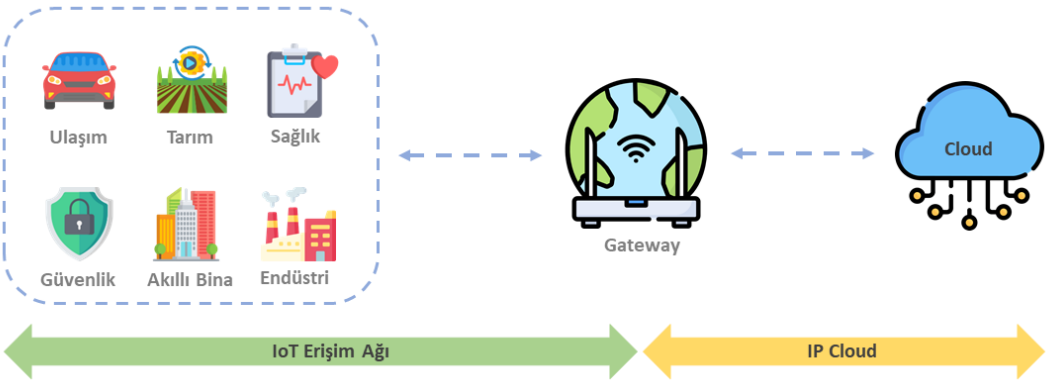
Literatürde Nesnelerin İnterneti alanında birçok materyal ve yöntemler kullanılmaktadır. Bu bölümde çalışmada kullanılan materyal ve yöntemler hakkında detaylı bilgi verilecektir.

3.1. IoT

Nesnelerin İnterneti farklı türde iletişim protokolleri kullanarak kendi aralarında haberleşebilen, veri toplayarak işleyebilen tüm nesnelerin/cihazların birlikte oluşturduğu ağdır (Bayılmış ve Küçük, 2019). IoT ilk olarak 1999 yılında Kevin Ashton tarafından Procter&Gamble (P&G) firmasının tedarik zincirinde Radyo Frekansı ile Tanımlama (Radio Frequency Identification, RFID) teknolojisinin kullanımına ve faydalarına değindiği sunumuyla hayatımıza girmiştir (Madakam ve diğ., 2015). Nesnelerin İnterneti uygulamalarının ilk örneği 1999 yılında 15 akademisyen tarafından kameralı sistem ile kahve makinesinin çevrimiçi izlenebilmesidir. 2000 yılında LG firması internete bağlı ilk buzdolabını piyasaya sürmüştür. 2005 yılında Uluslararası Telekomünikasyon Birliği (International Telecommunication Union, ITU) hakkında bir rapor yayınlamıştır. IBM CEO'su S. J. Palmisano tarafından 2009 yılında Akıllı Gezegen (Smart Planet) kavramının önerilmesiyle tanınırlığını arttırmıştır. 2014 yılında ise mobil cihaz ve makinelerin sayısı insan nüfusunu geçmiştir. Cisco tarafından 2030 yılına kadar internete bağlı olacak cihazların sayısının 500 milyarı bulacağı öngörülmektedir (Cisco, 2015).

Çok geniş bir kapsama ve uygulama alanlarına sahip Nesnelerin İnterneti, kullandığı protokoller ile internete bağlanabilen, sonrasında dünyanın her yerinden erişilebilir ve yönetilebilir cihazlardan oluşan küresel bir ağdır. Şekil 3.1'de genel bir IoT ağına ait sistem tasarımı verilmiştir. Sistem üç ana başlık altında incelenebilir:

1. IoT teknolojisini destekleyen ve uygulamalarda kullanılabilen cihazlar,
2. IoT cihazlardan toplanan verileri çeşitli haberleşme protokolleriyle internete bağlanabilmesini ve veri aktarımını sağlayan ağ geçidi,
3. IoT uygulamalarının izlenebildiği ve gelen verilerin depolanarak analiz edilebildiği ve anlamlı bilgilerin elde edildiği bulut platformları.



Şekil 3.1. Temel IoT sistemi

3.1.1. IoT Cihazları

Nesnelerin İnterneti teknolojisinin çalışma mantığında süreç ilk olarak IoT cihazında başlar. IoT cihazının amacı bünyesindeki algılayıcılar ile verileri toplamak, kablolu ya da kablosuz iletişim teknolojilerini kullanarak diğer cihazlar ile iletişim kurmak ve uzaktan haberleşmeyi desteklemek için internete bağlanmaktır.

Her bir IoT cihaz internet üzerinden haberleşebilmek için bir adres içerir ve kendine has eşsiz bir kimliğe sahip olur. Bunun sebebi internet üzerinden erişim sağlayabilmek ve diğer cihazlar arası iletişimi sağlamaktır. Kimlik tanımlamak için Wireless Sensor Networks (WSN), Radio Frequency Identification, Electronic Product Code (EPC) ve Ubiquitous Code (uCode) gibi tanımlamalar kullanılır.

Bir IoT cihazının esas amaçlarından biri olan veri toplama işleminde sensör ya da sensörlere sahip olması gerekir. Bunlar; hareket, sıcaklık, nem, kimyasal reaksiyon, ışık, basınç, sismik aktivite, akustik ve manyetik ölçümler için sensörler olabilir.

Cihazların çeşitli görevleri yerine getirebilmesi için kullanılacak uygulamaların çalıştırılabilmeleri için mikroişlemcili bir gömülü sisteme ihtiyaç duyar. Böylece veri toplama, işleme ve gönderimi gibi eylemlerde kullanılacak çevre birimlerinin yönetilmesi sağlanmış olur.

IoT cihazın veriyi dışarıya kablolu ya da kablosuz olarak aktarmak için birçok haberleşme teknolojisi kullanabilir. Bunlara örnek olarak; hücresel ağlar, Wi-Fi, BLE (Bluetooth Low Energy) ve Bluetooth verilebilir.

3.1.2. IoT Uygulama Katmanı Haberleşme Protokolleri

Verilerin IoT nesnelerden/cihazlardan toplandıktan sonra internet ortamına ulaşması için kullanılan ya da ağ üzerinde birlikte çalışan makinelerin kendi aralarındaki etkileşimin sağlanması için geliştirilen ve web servis olarak da adlandırılan sistemler, haberleşme protokolleri olarak bilinir.

IoT sistemler düşük kaynaklara (işlemci, bellek, enerji vb.) sahip olduğu için hafif haberleşme protokollerinin kullanımı önemlidir. IoT sistemleri temel olarak istemci-sunucu ve yayın-abone bazlı çalışmaktadır. Bu doğrultuda en sık kullanılan haberleşme protokolü olarak aşağıdaki protokoller verilebilir:

1. Mesaj Kuyruk Telemetri Ulaştırma Protokolü,
2. Basit Nesne Erişim Protokolü (Simple Object Access Protocol, SOAP),
3. İleri Mesaj Kuyruklama Protokolü (Advanced Message Queuing Protocol, AMQP),
4. Veri Dağıtım Hizmeti (Data Distribution Service, DDS),
5. Üstün Metin Transfer Protokolü (Hyper Text Transfer Protocol, HTTP).

3.1.3. Nesnelerin İnterneti'nde Bulut Platformlar

IoT'nin en önemli bileşeni olan bulut, birçok alanda uygulamalara önemli hizmetler sunar. Bulut platformlar uygulamalara cihaz yönetimi, sistem yönetimi, veri yönetimi, depolama dağıtım, izleme, görselleştirme ve araştırma araçları gibi çeşitli çözümler sunmaktadır. Ayrıca IoT cihazlar tarafından toplanarak elde edilen büyük veri üzerinde çeşitli analizler ile veri anlamlandırma yapılabilmektedir.

Günümüzde Nesnelerin İnterneti alanında pek çok farklı bulut platformları kullanılmaktadır. Bu platformlara örnek olarak; IBM Watson IoT, Oracle IoT Cloud, Google Cloud Platform, Cisco IoT Cloud Connect, Microsoft Azure IoT ve Amazon Web Service IoT verilebilir.

3.1.4. Nesnelerin İnterneti Kullanım Alanları

IoT hayatımızın pek çok alanında bizimle beraberdir. Günlük yaşantımızı kolaylaştırarak bize pek çok alanda destek sağlamaktadır. Örnek olarak IoT, aşağıdaki alanlarda karşımıza çıkmaktadır;

- Akıllı Ev: İnternete bağlı aydınlatma ve ısıtma gibi alet ve sistemlerin uzaktan izlenebilmesi ve yönetilebilmesini sağlamaktadır.
- Sağlık Uygulamaları: Uzaktan hasta gözlemlenmesi, kronik hastalıkların takibi ve hasta bilgilerinin farklı cihazlardan toplanarak analiz edilmesi gibi alanlarda kullanılmaktadır.
- Akıllı Şehir: Yenilenebilir ve sürdürülebilir enerji yönetimi, akıllı enerji ve mikro şebekeler, akıllı sayaçlar ve hava kirliliği izleme sistemleri gibi bir çok sistem kullanılmaktadır.
- Lojistik: Lojistikte IoT, varlıkların gerçek zamanlı olarak izlenebilmesi ve depo yönetimi uygulamalarında kullanılırken aynı zamanda da analitik ve veriye dayalı öngörüler üzerinde çalışabilmek için bir veri kaynağı olarak kullanılır.
- Endüstri: Üretim makineleri ağ üzerinden birbirleri ile haberleşerek üretimi kontrol etmekte, yöneticiler üretimdeki tüm süreçleri istenilen yer ve zamanda takip edebilmekte ve arızalar daha önceden tespit edilerek önceden önlem alınabilmektedir.

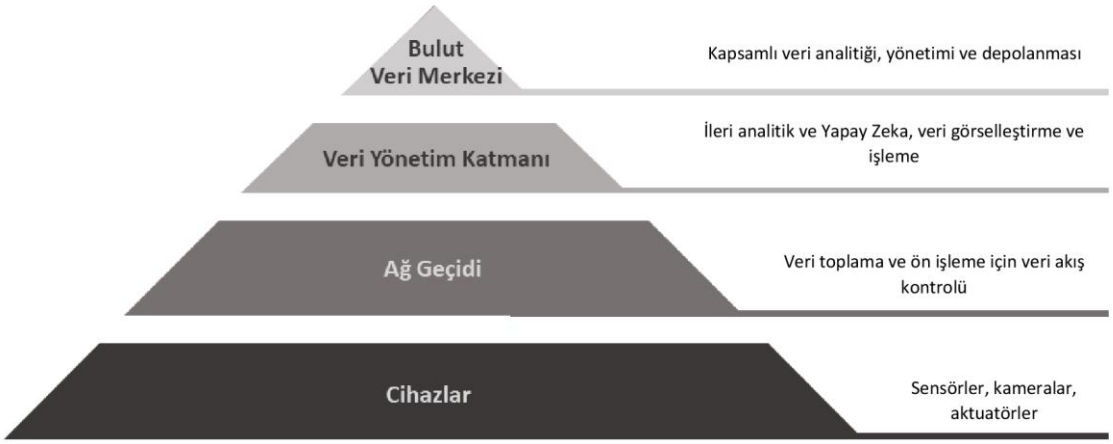
3.2. IIoT

Endüstriyel Nesnelerin İnterneti ilk olarak 1968'de Dick Morley tarafından General Motors otomatik şanzıman üretim bölümünde kullanılan programlanabilir mantıksal denetleyicisi (Programmable Logic Controller, PLC) icadı ile ortaya çıkmıştır (Boyes ve diğ., 2018). Bu PLC'ler yardımı ile üretimdeki elementlerin kontrol edilebilmesi sağlanmıştır. 1975'te ise Honeywell ve Yokogawa dünyanın ilk Dağıtık Kontrol Sistemlerini (Distributed Control System, DCS) tanıtmıştır. Bu sistemler sırasıyla TDC 2000 ve Centum sistemleridir. Bu sistemlerle beraber tüm kontrolü sisteme dağıtarak esnek bir kontrol mekanizması elde edilmiştir. 1980'de Ethernet'in ortaya çıkması ile birlikte 1982'de insanlar akıllı cihazlardan oluşan bir ağ kavramını keşfetmeye başlamıştır. İlk internet bağlantılı cihaz Carnegie Mellon Üniversitesi'ndeki stok miktarı ve yeni koyulan içeceğin soğuk olup olmadığı hakkında bilgi veren Kola makinesidir.

3.2.1. IIoT Mimarisi

Geçmişten günümüze endüstride kullanılan cihazların gelişimi internete bağlanabilen cihazların sayısındaki önemli artış ile birlikte artık üretim alanında geniş bir IIoT ağı kurulabilmekte ve makineler birbirleri ile haberleşerek üretimi kontrol edebilir bir

yeteneğe sahip olmuştur. Prensipte IoT ve IIoT aynı şekilde çalışır. Her ikisi de cihazları internete bağlar ve onları daha akıllı hale getirir. Aradaki fark, IIoT'nin üretim tesislerinde güvenliği ve verimliliği artırmak için çalışırken, IoT tüketicilerin daha rahat ve kolay yaşamasını sağlamak için çalışmasıdır. İnternete bağlanabilen ve uzaktaki bir veri sunucusuna veri aktarabilen herhangi bir cihaz, Nesnelerin İnterneti olarak adlandırılır. IoT cihazları yalnızca endüstriyel amaçlarla kullanıldığında Endüstriyel Nesnelerin İnterneti olarak adlandırılır. IIoT, IoT'nin alt kümesidir. Tüm IIoT cihazları, IoT cihazlarıdır. Ancak tüm IoT cihazları IIoT cihazları değildir. IIoT cihazları, elektrik santralleri, arıtma tesisleri gibi kritik altyapılarda kullanılabilir. IIoT daha büyük ölçekli ağlarda kullanılır, IoT ise daha küçük sistemlerde kullanılır. Aynı zamanda da IIoT cihazları IoT cihazlarına göre daha dayanıklı ve uzun ömürlüdür. IoT, B2C (Business To Consumer)'dir ve IIoT, B2B (Business To Business)'dir. Verileri izleme ve analiz etme, tüm tedarik zincirinde kestirimci bakım yapma ve aynı sistemden personeli yönetme yeteneği, üretim verimliliğini önemli ölçüde arttırarak ve birçok evrak işini ortadan kaldırmaktadır. Makinelerin kullanıldığı her alanda mekanik ya da elektriksel arızalar önceden tespit edilebilir ve önceden müdahale edilebilmektedir. Anlık stok miktarı ve buna benzer otomasyon sistemleri ile hammadde tedariği hızlı bir şekilde sağlanarak zamandan tasarruf sağlanmaktadır. IIoT sayesinde üretimde kullanılan sensör ve benzeri algılayıcılar ile toplanan veriler internet üzerindeki bulut platformlarda toplanıp raporlanarak yöneticiler tarafından her an izlenebilmektedir. Şekil 3.2'deki görselde temel bir IIoT mimarisi verilmiştir.



Şekil 3.2. Temel IIoT sistemi

IIoT'nin endüstride önemli etkileri bulunmaktadır. Üretimde maliyet, zaman tasarrufu, enerji verimliliği ve performans gibi önemli parametrelerde olumlu faydalar sağlamaktadır. IIoT sayesinde kullanılan düşük güç tüketimine sahip ve kablosuz iletişimi destekleyen algılayıcılarla hem maliyet hem de enerji verimliliği sağlanmaktadır. Algılayıcılar ile sahadan toplanan ham veri depolama, işleme ve izleme gibi işlemlerde kullanılır. Aynı zamanda da sahadan gelen sürekli veri ile oluşturulan büyük veriyi değerlendirip analiz edilerek karar verme uygulamalarında kullanılmaktadır. Büyük veri oluşturmak için işletmelere önemli katkı sağlamaktadır.

3.2.2. IIoT Temel Özellikleri

Endüstriyel Nesnelerin İnterneti'nin temel özellikleri Endüstriyel Sensörler/Cihazlar, Bağlanabilirlik, Veri Akışı ve Analitiği, Entegrasyon ve Otomasyon olarak ayrı başlıklar altında incelenmiştir.

3.2.2.1. Endüstriyel Sensörler/Cihazlar

Nesnelerin İnterneti teknolojisi, sensörlerin/cihazların kullanımıyla ayırt edici özelliğe sahiptir. Sensörler/Cihazlar Nesnelerin İnterneti sisteminin kalbidir. Endüstriyel kullanım durumları, sürekli okuma ve kendi kendini kalibre etme durumlarında daha kararlı ve sağlam sensörler kullanmak sistemin sağlığı açısından gereklidir. Genellikle, online ve inline sensörler olarak adlandırılırlar. Bu tür sensörler uzun ömürlü ve dayanıklı olması sebebi ile pahalıdırlar. Bu sensörler ve IoT teknolojisi, standart bir ağı, endüstriyel ortamda aktif bir sisteme dönüştürür.

3.2.2.2. Bağlanabilirlik

Endüstriyel IoT, sorunsuz bir şekilde veri göndermek ve almak için bağlantı hızı gerektirir. 4G/5G gibi modern kablosuz teknolojiler IoT ağ iletişimi ve bağlanabilirliğinin daha kolay ve hızlı hale getirilmesinde etkili rol oynamaktadır. Sensörü aktif tutabilmek için daha az enerji tüketen teknolojiler geliştirilmiştir. Wi-Fi ve Bluetooth gibi geleneksel kablosuz teknolojilerin çalışması için yüksek pil gücü gerektirir. Endüstriyel IoT, uzak cihazları daha uzun süre çalışır durumda tutmak için BLE ve LPWAN (Low Power Wide Area Networks) gibi daha yeni kablosuz teknolojilerden yararlanır.

3.2.2.3. Veri Akışı ve Analitiği

İşleri ve makineleri daha akıllı hale getirmek için daha derin bir analitik düşünme ve işleme gereklidir. Endüstriyel IoT platformu, sistemi veya makineyi akıllı hale getirmek için analitik düşünme yeteneklerini uygulamak için yüksek hızlı veri akışı ve işleme yeteneğine sahip olmalıdır. IIoT karmaşık endüstriyel kullanım durumlarını desteklemek için gelişmiş makine öğrenimi ve yapay zeka yeteneklerine sahiptir.

3.2.2.4. Entegrasyon

Makine öğrenmesi ve yapay zeka gibi ileri veri işleme yöntemleri için gerekli olan büyük verinin elde edilebilmesi için sistemin, diğer veri akışının sağlandığı sistemlerle entegre çalışabilmesi kritik bir noktadır. Bu bağlamda IIoT sisteminin diğer uygulamalarla entegre çalışabilir bir yeteneğe sahiptir.

3.2.2.5. Otomasyon

Veri toplamaya ek olarak, IIoT otomasyon yetenekleri sağlayarak kendisini geleneksel IoT platformundan ayırır. Otomasyon, operasyonel süreçleri kontrol etmek için manuel ve insan gücünü azaltmanın modern yoludur. Otomasyon özelliklerine sahip IIoT platformu, sensör verilerine dayanarak otomatik işlemlerin tetiklenmesine yardımcı olur (örn: su seviyesi düşük olduğunda motor pompasını uzaktan açma). Otomasyon, makinelere veya akıllı cihazlara harekete geçmesi için talimatlar göndererek sağlanabilir.

3.3. Modbus RTU

Endüstride ve elektronik alanda yapılan çalışmalarda cihazların birbiriyle haberleşmesi için çeşitli metotlar kullanılmaktadır. Bu metotlar ile kullanıcı tarafından parametreler alınabileceği gibi aynı zamanda da cihazların daha temelinde olan çeşitli parçaları arasında da veri alışverişi yapılabilmesi için bu metotlardan yararlanılmaktadır. Günümüzde cihazların haberleşmesi için daha çok Ethernet, USB (Universal Serial Bus) gibi haberleşme metotları kullanılırken biraz daha temelde olan ve az bilinen SPI (Serial Peripheral Interface), I2C (Inter-Integrated Circuit), UART (Universal Asynchronous Receiver Transmitter) ve CAN (Controller Area Network) gibi haberleşme teknikleri

bulunmaktadır. Bu iletişim metotları haberleşmenin yapılacağı fiziksel altyapı ve ortama göre değişiklikler gösterebilmektedir.

Modbus ilk olarak 1979 yılında, mikro denetleyicilerin olmadığı ve mikroişlemcilerin yeni yeni kullanıldığı yıllarda PLC ve buna benzer sistemlerin haberleşebilmesi için Modicon firması tarafından yayımlanan seri iletişim protokolüdür (Agashe ve diğ., 2015).

Modbus, sunucu/istemci tabanlı bir iletişim protokolüdür. Endüstri ortamında sıklıkla tercih edilmesinde akıllı endüstriyel sistemler için geliştirilmesi, açık kaynak olması, telif hakkı bulundurmaması, kendini kanıtlamış ve tanınmış olması, kolay adapte edilebilir olması ve performanslı olması gibi pozitif özellikleri ile kullanımı yaygınlaşmıştır.

3.3.1. Modbus Protokolünün Yapısı

Modbus protokolü, haberleşmede farklı mesaj yapıları kullanmaktadır. Modbus protokolü haberleşme için master/slave ilişkisi kullanır. Modbus RTU ve Modbus ASCII haberleşmesinde ağda 1 adet master cihaz ve 247 adet slave cihaz çalışabilmektedir.

İletişimde slave cihazlar, sürekli hattı dinler ve bir istek olup olmadığını kontrol eder. Slave birim kendisine verilen eşsiz kimlik numarasına göre master cihazdan gelen isteğe gerekli cevabı hazırlar ve gönderir.

Modbus TCP/IP haberleşmesinde ağda birden fazla master cihaz bulunabilir. İletişim çift yönlüdür. Çalışma durumunda iken slave cihaz master, master cihaz ise slave olarak davranabilir. Bu durumda iletişim çift yönlü olarak sağlanabilir.

Modbus protokolünde her bir versiyonun kendine göre mesaj yapısı bulunmaktadır. Bu versiyonlar arasında en sık kullanılan versiyonlar Modbus RTU, Modbus ASCII ve Modbus TCP/IP versiyonlarıdır.

3.3.2. Modbus RTU ve Modbus ASCII Mesaj Yapısı

Modus RTU ve Modbus ASCII versiyonlarında mesaj yapısı oldukça benzerdir. Modbus protokolünde mesaj boyutu maksimum 255 byte'tır. İletişimde kullanılan mesajlar başlangıç ve bitiş bölümleri hariç toplamda 4 bölümden oluşmaktadır. Bu bölümler adres,

fonksiyon, veri ve hata kontrolüdür. Modbus RTU ve Modbus ASCII mesaj yapısı Şekil 3.3 ve Şekil 3.4'te verilmiştir.

Başlangıç	Adres	Fonksiyon	Veri	CRC Kontrol	Bitiş
≥ 3.5 char	8 bit	8 bit	N x 8 bit	16 bit	≥ 3.5 char

Şekil 3.3. Modbus RTU

Başlangıç	Adres	Fonksiyon	Veri	LCR	Bitiş
1 char	2 chars	2 chars	0'dan 2x252 char kadar	2 char	2 char2 char CR,LF

Şekil 3.4. Modbus ASCII

Modbus RTU ve Modbus ASCII protokollerinde iletişimde kullanılan mesajın hangi slave birimine gönderileceği belirlenmesinde bu adres bölümü kullanılır. Adres bölümü 1 ile 247 arasında bir değer alabilir. 0 yayın adresi ve 248 ile 255 arasındaki adresler ayrılmıştır. Master ünitesi göndereceği mesajın hangi slave üniteye gönderecek ise o ünitenin adresini mesaja ekler. Slave ünite gelen mesajın adres bölümüne bakar ve mesaj kendine ait değilse değerlendirmeye almaz, eğer mesaj kendine ait ise gerekli işlemleri yaparak mesaja kendi adresini ekler ve geri gönderir. Bu sayede master cihaz gönderdiği mesajın hangi üniteden geldiğini anlayabilir.

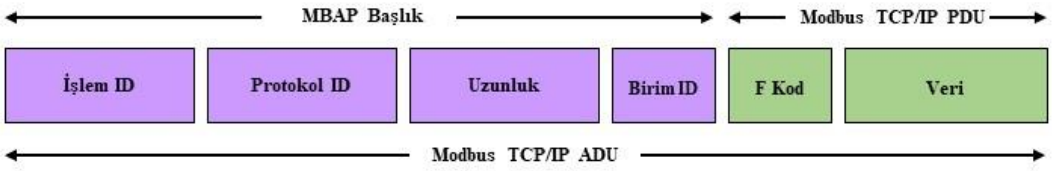
Fonksiyon bölümü, master cihazdan slave cihaza veri gönderiminde slave cihazın yapacağı işlem çeşidini belirtir. Bu bölüm, işlemin herhangi bir sebepten dolayı yapılamaması durumunda hata kodu içerebilmektedir. En çok kullanılan fonksiyonlar 3, 6 ve 16 numaralı fonksiyonlardır. Örneğin fonksiyon kodu 3 olarak belirlendiğinde, master cihaz slave cihaza hafızasında tuttuğu verileri okuyacağını belirtir. Slave cihaz sensörlerden okuduğu verileri hafızasında tutar. Ayrıca master cihaz hangi yazmaçtan okumaya başlayacağını ve kaç adet yazmaç okuyacağını belirtir.

Data bölümü, gelen fonksiyon koduna göre hangi verinin kullanılacağını belirtir. Gelen fonksiyonun koduna göre ek bilgiler içerebilir.

Modbus protokolünde iletişim mesajlar kullanılarak yapıldığından mesajların bütünlüğünün kontrol edilmesi gerekir. Hata kontrolünün yapılmasında Modbus RTU ve Modus ASCII protokollerinde farklı yapılar bulunmaktadır. Modbus RTU CRC (Cyclic Redundancy Check) yöntemini kullanmaktadır. CRC bölümü 16 bit uzunluğundadır. Modbus ASCII ise LRC (Longitudinal Redundancy Check) yöntemini kullanır. Bu bölüm ise 2 karakterden oluşmaktadır.

3.3.3. Modbus TCP/IP Mesaj yapısı

Modbus TCP/IP mesaj yapısı 2 ana bölümden oluşmaktadır. Bu bölümler sırasıyla Modbus Application Protocol (MBAP) header ve Modbus TCP/IP PDU (Protocol Data Unit) bölümleridir. Modbus TCP/IP mesaj yapısı şekil 3.5'de verilmiştir.



Şekil 3.5. Modbus TCP/IP

3.3.3.1. MBAP Header

Bunların ilki MBAP bölümüdür. MBAP bölümü de 4 bölümden oluşmaktadır ve uzunluğu 7 byte'dır. MBAP bölümü master ve slave cihazların aralarında haberleşmesi için gerekli bilgileri içermektedir.

İşlem tanımlayıcı bölümü, 2 byte uzunluğundadır. Bu bölüm master ve slave olarak belirlenen cihazların birbirleriyle ilişkilendirilerek haberleşebilmelerini sağlamaktadır.

Protokol tanımlayıcı bölümü, günümüzde kullanılmamaktadır. Çoklu sistemler için ayrılmıştır. Gelecek yıllarda kullanılmasına ihtimal verilerek bu alan oluşturulmuştur. Modbus için 0 değerini alır. Bu bölüm 2 byte uzunluğundadır.

Uzunluk bölümü, birim tanımlayıcı bölümü ve PDU bölümü dahil olmak üzere bu bölümlerin uzunluğunu byte formatında tutar. Bu bölüm toplamda 2 byte uzunluğundadır.

Birim tanımlayıcı bölümü, yönlendirme amaçlı kullanılmaktadır. Ağ üzerinde olmayan uzak birimi belirlemek için kullanılır. Bu bölüm 1 byte uzunluğundadır.

3.3.3.2. Modbus TCP/IP PDU

Modbus TCP/IP mesaj yapısının ikinci bölümü PDU bölümüdür. Bu kısımda 2 bölüme ayrılmıştır. Modbus TCP/IP mesaj yapısında PDU bölümü, işlemlere yönelik bilgiler içermektedir. Fonksiyon kodu ve data bölümlerinden oluşmaktadır.

Fonksiyon kodu bölümü, master cihazın slave birimden yapılmasını istediği işleme yönelik belirlenen kodu içermektedir. Fonksiyon kodunun uzunluğu 1 byte'dır.

Data bölümü, master cihazdan slave cihaza gönderilen mesajda yapılması istenen işlem için özel bilgiler içerir. Slave cihaz cevabı hazırladığında bu bölüme talep edilen işleme karşılık gelen cevap ve hata kodlarını ekleyerek mesajı master cihazına gönderir.

Bu çalışmada haberleşme katmanında Modbus RTU kullanılmaktadır. Bunun sebebi, master ve slave cihazlar arasında fiziksel katmanda kablolu olarak seri iletişim tercih edilmiştir. Master cihaz slave cihaza kablolu bir şekilde bağlantı sağlamaktadır. Modbus ASCII yerine Modbus RTU kullanılmasının sebebi ise ASCII'de veriler tek tek gönderilirken, RTU'da ise veriler tek seferde gönderilerek ASCII'ye göre daha hızlı bir iletişim sağlamasıdır. Veriler master ile alındıktan sonra IoT bulut platformuna Wi-Fi kullanılarak aktarılmaktadır.

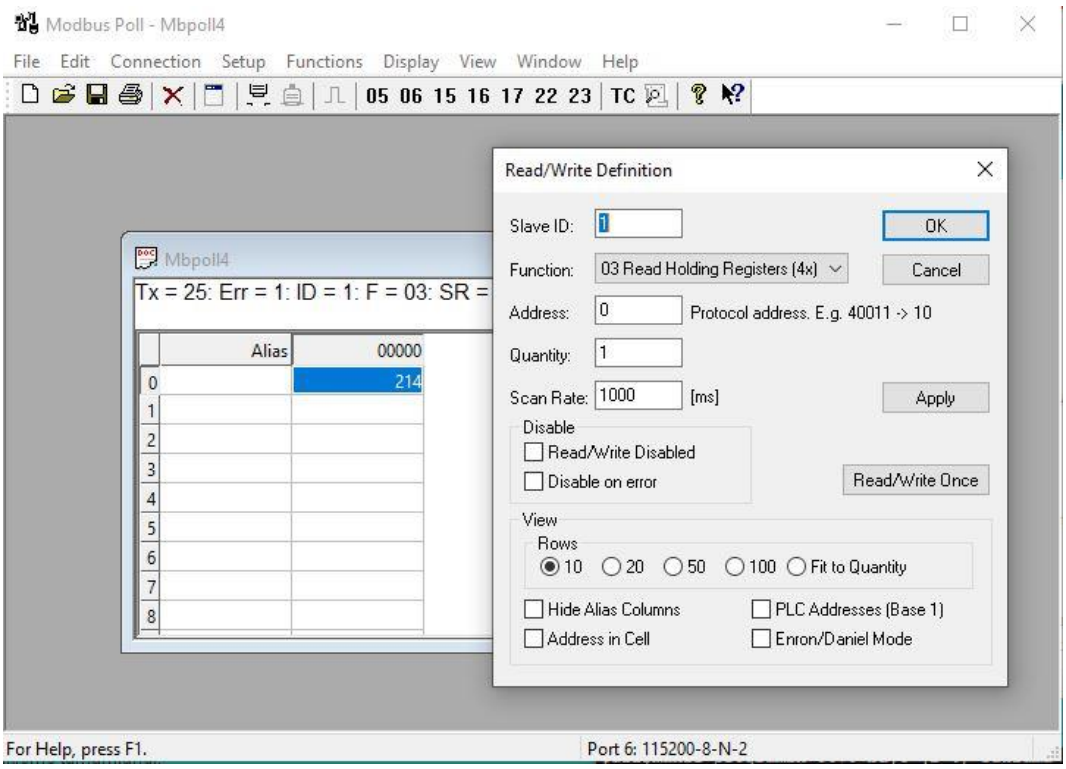
3.3.4. Modbus Poll ve Modbus Slave Uygulamaları

Modbus master ve slave birimlerini test etmek için iki farklı uygulama bulunmaktadır. Bu uygulamalar vasıtası ile belirli konfigürasyonlarla kodlar test edilebilmektedir. Modbus master ve slave kodlarını test edebilmek için Modbus Poll ve Modbus Slave uygulamaları kullanılmıştır.

3.3.4.1. Modbus Poll

Modbus Poll uygulaması, Modbus slave cihazlarından veri okumak, Modbus protokolünü test etmek ve benzetim yapmak isteyenlere yardımcı olmak için tasarlanmış bir Modbus master simülatörüdür. Çoklu belge arayüzü ile aynı anda birden fazla Modbus slave

cihazlarından veri izlenebilmektedir. Her bir arayüzde okunacak slave ID, fonksiyon kodu, okunacak adres bilgisi, okunacak veri miktarı ve okuma sıklığını belirterek uygulama üzerinden slave cihazlar test edilebilmektedir. Şekil 3.6'da Modbus Poll uygulamasına ait arayüz ekranı verilmiştir.

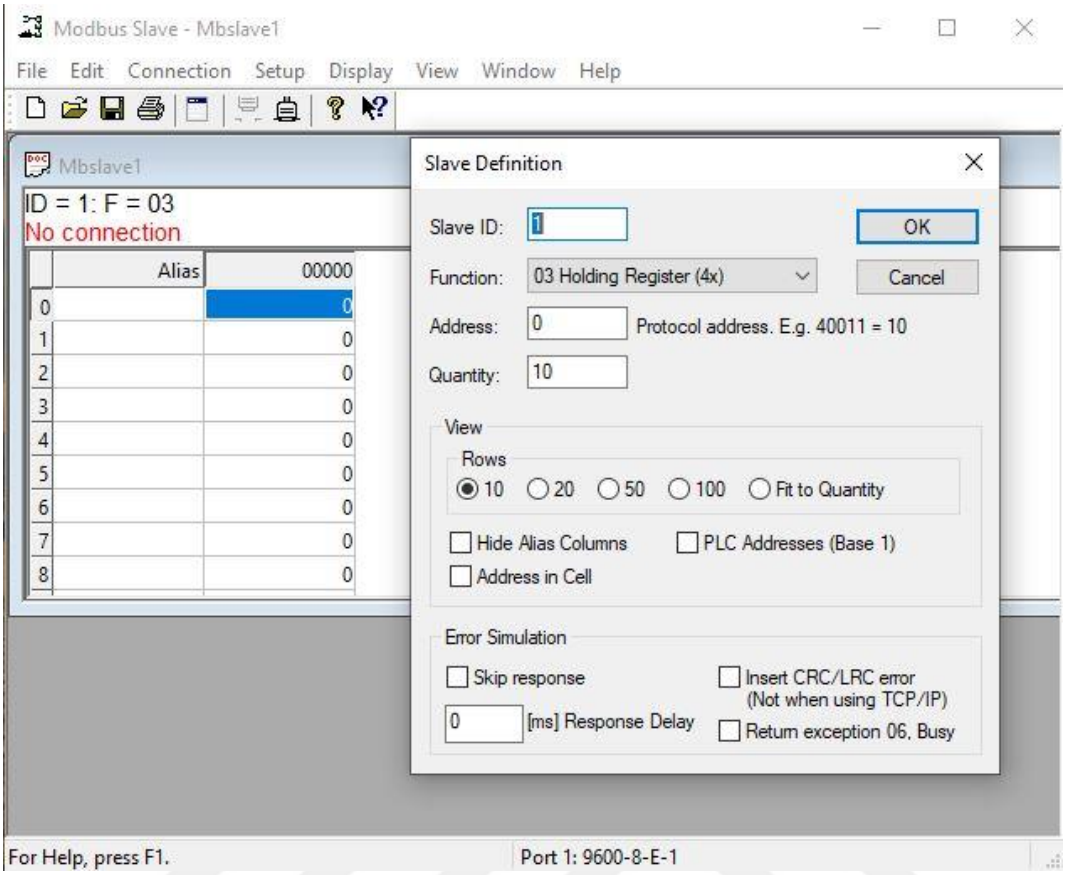


Şekil 3.6. Modbus poll uygulaması

3.3.4.2. Modbus Slave

Bir slave simülatörü, Modbus master için bir veri kaynağı sağlar ve geliştirilmekte olan bir cihaz için bir test iş akışı kurarken bir veri kaynağı olarak kullanılmaktadır. Bir slave simülatörü, geliştirilmekte olan yeni bir cihazın modelini oluşturmak için de kullanılabilir ve geliştirilmekte olan cihazı doğrulamak için hem geliştirme spesifikasyonu hem de beklenen davranış kaynağı olarak hareket edebilir.

Modbus Slave, 100 ayrı pencerede 100 slave cihazın benzetiminin yapılması için kullanılabilmektedir. Modbus Slave, çoklu belge arayüzü kullanır. Bu, birden fazla pencerenin açılabileceği anlamına gelmektedir. Tek arayüzden birden fazla slave cihazdan okunan veriler takip edilebilmektedir. Şekil 3.7'de Modbus master kodlarının test edildiği Modbus Slave uygulamasına ait görsel bulunmaktadır.



Şekil 3.7. Modbus slave uygulaması

3.4. MQTT Protokolü

MQTT, 1999 yılında IBM firmasında çalışan Dr. Andy Standford-Clark ve Arcom firmasında çalışan Arlen Nipper tarafından geliştirilmiştir. MQTT protokolünün v5.0 ve kablosuz algılayıcı ağlar için MQTT-SN (MQTT for Sersor Network) olarak iki temel versiyonu bulunmaktadır.

MQTT terimi, endüstri alanında olmazsa olmaz diye tabir edebileceğimiz IoT ile hayatımıza girmiştir. MQTT daha farklı olarak akıllı ev sistemlerinde ve akıllı şehirler gibi daha pek çok IoT uygulamalarında kullanılmaktadır.

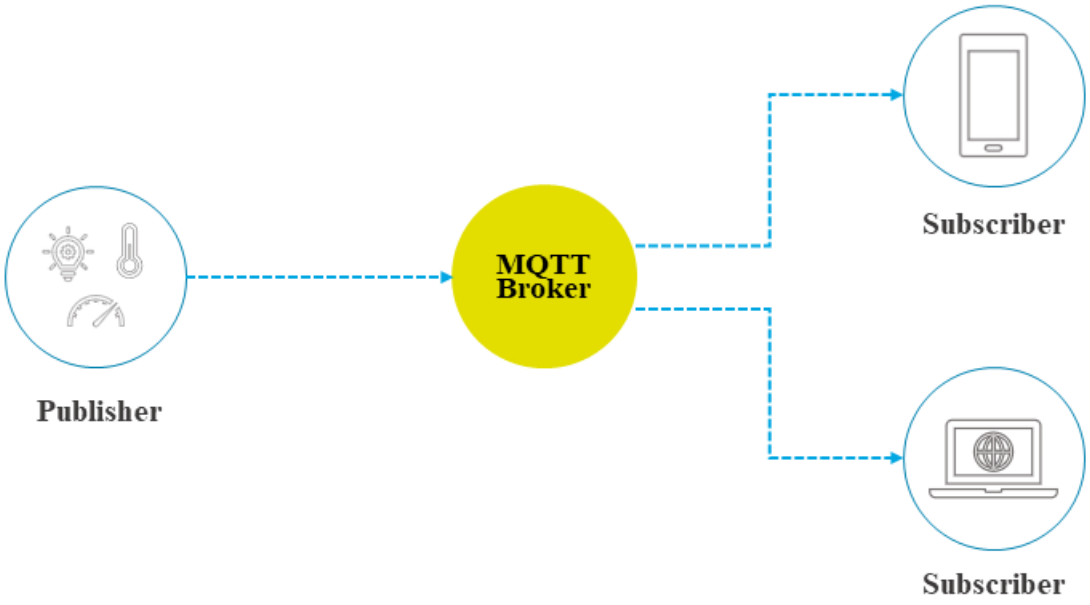
3.4.1. MQTT Özellikleri

- Güvenlik olarak SSL/TLS (Secure Sockets Layer/ Transport Layer Security) desteklemektedir.
- Asenkron çalışan bir protokoldür.

- Hızlı haberleşme sağlamaktadır (Milisaniye seviyesinde).
- Kaynak kullanımı minimum seviyededir.
- TCP/IP yapısını kullanan Windows, Linux, MacOS, Android ve IOS işletim sistemlerinde çalışabilmektedir.
- Broker üzerinden haberleşme temeline dayanmaktadır.

3.4.2. MQTT Protokolü Çalışma Prensibi

MQTT, temel çalışma mantığı olarak bir yayına abone olma ve mesaj yayınlamaya dayalı mesaj iletim protokolüdür. MQTT ağındaki cihazlar, bir MQTT sunucu vasıtası ile mesaj yayımı yapabilir veya tercih edilen mesajı dinlemek için o mesaja ait konuya abone olabilirler. Broker, haberleşme trafiğinin kontrol edilmesinde merkez birime verilen isimdir. Yayımcı ve aboneler arasında haberleşmeyi sağlar. Yayımcı, belirli konuya ait mesajların yayınlamasından sorumludur. Mesajları broker'a gönderir. Abone, yayımcı tarafından broker'a gönderilen konulara abone olan birimdir. Mesajları analiz etmek ve işlemek için alan hedef kullanıcıdır. Konu ise yayımcı ve abone arasında mesajların bir başlık altında tutulduğu kısımdır. MQTT protokolünün çalışma prensibi Şekil 3.8'te gösterilmiştir.



Şekil 3.8. MQTT çalışma prensibi

MQTT broker varsayılan portu 1883'tür. Ancak bu şifresiz bağlantılar içindir. Şifreli bağlantılar için 8883 portu kullanılmaktadır. MQTT sunucu olarak genellikle Mosquitto, HiveMQ tercih edilmektedir.

TCP protokolü üzerine kurulu olan MQTT, güvenlik olarak SSL/TLS kullanır. Bağlantı işlemi sırasında kullanıcı adı ve şifre kullanır.

3.4.3. MQTT Mesaj Yapısı

MQTT protokolü mesaj yapısı istenilen boyuta ayarlanabilen esnek bir yapıya sahiptir. Mesaj boyutu minimum 2 byte olarak sabit bir başlık alanı bulunmaktadır. Mesaj tipi alanı mesaj türünü temsil eder. DUP, mesajın kopyalandığını ifade eden bayraktır. QoS, yayımlanacak olan mesajın hangi servis kalitesinde iletileceğini belirler. Retain, sunucuya ulaşan son mesajı sunucuya bildirir. Abone olan yeni istemciye ilk mesaj olarak iletilmesi sağlanır. Kalan uzunluk, gönderilecek olan mesajın kalan boyutunu gösterir. Değişken Başlık bölümü protokol bilgilerini içerir. Son olarak yük bölümü ise kullanıcı, şifre, mesaj gibi bilgiler içermektedir. MQTT mesaj yapısı Şekil 3.9'da verilmiştir.

Şekil 3.9. MQTT çalışma prensibi

MQTT dört bölümden oluşmaktadır. Bunlar; bağlantı, kimlik doğrulama, iletim ve bağlantı sonlandırmadır. Bu işlemler MQTT paketleri tarafından yönetilir. Paket tipleri Tablo 3.1'de verilmiştir.

Tablo 3.1. MQTT paket tipleri

Ad	Değer	Mesaj Yönü	Açıklama
Rezerve	0	Yasak	Rezerve edilmiş
CONNECT	1	İstemciden Sunucuya	İstemciden sunucuya bağlanma talebi
CONNACK	2	Sunucudan İstemciye	Bağlantı onayı
PUBLISH	3	İstemciden Sunucuya ya da Sunucudan İstemciye	Mesaj yayınlama
PUBACK	4	İstemciden Sunucuya ya da Sunucudan İstemciye	Yayınlama onayı
PUBREC	5	İstemciden Sunucuya ya da Sunucudan İstemciye	Yayın alındı (Güvenilir teslimat bölümü 1)
PUBREL	6	İstemciden Sunucuya ya da Sunucudan İstemciye	Yayın gönderme (Güvenilir teslimat bölümü 2)
PUBCOMB	7	İstemciden Sunucuya ya da Sunucudan İstemciye	Yayın tamam (Güvenilir teslimat bölümü 3)
SUBSCRIBE	8	İstemciden Sunucuya	İstemci abonelik talebi
SUBACK	9	Sunucudan İstemciye	Abonelik onayı
UNSUBSCRIBE	10	İstemciden Sunucuya	Aboneliği kaldırma isteği
UNSUBACK	11	Sunucudan İstemciye	Abonelik kaldırma isteği onayı
PINGREQ	12	İstemciden Sunucuya	PING talebi
PINGRESP	13	Sunucudan İstemciye	PING yanıtı
DISCONNECT	14	İstemciden Sunucuya	İstemci bağlantıyı sonlandırıyor
Rezerve	15	Yasak	Rezerve edilmiş

3.4.4. QoS 0 İle Mesaj Gönderimi

QoS 0 servis kalitesinde veri iletiminde mesajın en fazla bir kez iletir. Mesaj sunucu tarafında depolanmadığı için kopya mesaj oluşmaz. Mesajın ulaşıp ulaşmadığı kontrol edilmez. Yayımcı mesajı alıcıya bir kez gönderir. Mesajın karşı tarafa iletilip iletilmediği kontrol edilmediği için mesaj kaybı yaşanabilir. QoS 0 servis kalitesinde veri iletimine ait görsel Şekil 3.10'da verilmiştir.

Şekil 3.10. MQTT QoS 0 mesaj gönderimi

3.4.5. QoS 1 İle Mesaj Gönderimi

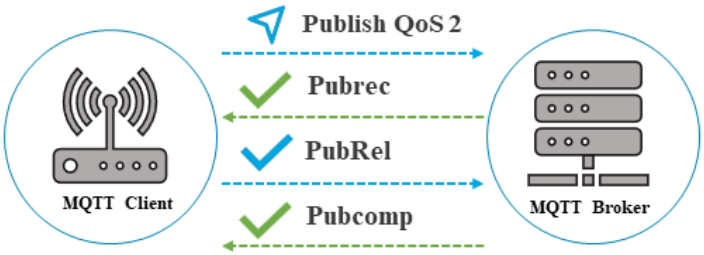
QoS 1 servis kalitesinde veri iletiminde mesajın karşı tarafa minimum bir kez iletileceği kesindir. Ancak mesaj sunucu tarafında depolandığı için mesaj birden fazla kez iletilebilir. Yayımcı mesajı alıcıya bir kez gönderir. Alıcı mesajı aldığında mesajı aldığına dair PUBACK mesaj tipini geri döner. Çeşitli nedenlerden dolayı yayımcı belirli süre içerisinde PUBACK paketini alamadığında DUP bayrağını 1 yaparak mesajı tekrar gönderir. Bu durumda alıcıda mesajın birden fazla kopyası oluşabilir. Yayımcı PUBACK mesajını aldığında veri iletimi tamamlanmış olur. Mesaj iletimi süresince veriler depolandığı için mesaj kaybı yaşanmaz, kopya mesaj oluşabilir. QoS 1 servis kalitesinde veri iletimine ait görsel Şekil 3.11'de verilmiştir.

Şekil 3.11. MQTT QoS 1 mesaj gönderimi

3.4.6. QoS 2 İle Mesaj Gönderimi

QoS 2 servis kalitesinde mesaj iletiminde verinin alıcıya kesinlikle bir kez iletileceği garanti edilir. Fakat trafiğin en yoğun ve yavaş olduğu mesaj iletim yöntemidir. Yayımcı ilk olarak PUBLISH paketi gönderir. Alıcı bu paketi aldıktan sonra yayımcıya PUBREC paketi ile yanıt verir. Bu paket bir paket tanımlayıcı içerir ve PUBCOMP paketi alana kadar bu değeri saklar. Bu işlem bir mesajın birden fazla kez iletilmesini engellemek için kullanılır. Yayımcı PUBREC paketini aldığında ilk adımda sorun yaşanmadan verinin karşı tarafa iletildiğini anlar ve PUBLISH paketini silerek PUBREC paketini tutar ve PUBREL paketini göndermektedir. PUBREL, PUBREC ile aynı paket tanımlayıcısına

sahip olur. Alıcı PUBREL paketini aldığında son adım olarak PUBCOMP paketini gönderir ve veri iletimi başarılı bir şekilde tamamlanmış olur. QoS 2 servis kalitesinde veri iletimi diğer servis kalitelerine göre en garanti ve güvenli yöntemdir. Veri kaybı yaşanmaz ve kopya mesaj oluşmaz. QoS 2 servis kalitesinde veri iletimine ait görsel Şekil 3.12'de verilmiştir.



Şekil 3.12. MQTT QoS 2 mesaj gönderimi

3.4.7. MQTT-SN

Bu protokol bir kablosuz algılayıcı ağdaki her bir cihazın internete bağlı olmasını gerektirmez. Ağda bir tane iletilecek verilerin toplandığı düğüm bulunmaktadır. Bu düğüm vasıtasıyla veriler internet ortamına aktarılmaktadır. Bu düğüm verileri broker'a aktarmak için MQTT protokolü ile bağlanarak verileri aktarır.

MQTT-SN, verileri sunucuya iki farklı mimari kullanarak iletir. Bunlardan ilki birleştirilmiş olarak adlandırılır. Bu mimaride veriler ağ geçidinde toplanır ve sunucuyla sadece bir bağlantı kurularak veriler aktarılır. Diğer bir mimari ise şeffaf olarak adlandırılır. Şeffaf mimaride ağ geçidi her bir düğüm için sunucuda ayrı bağlantılar oluşturur. Burada ağ geçidi yönlendirici gibi davranmaktadır. Birleştirilmiş mimari şeffaf mimariye göre sunucu üzerinde daha az yük oluşturur. Bunun sebebi sunucu üzerinden tek bir bağlantı kurmasıdır.

2.5. IBM Watson IoT Platformu

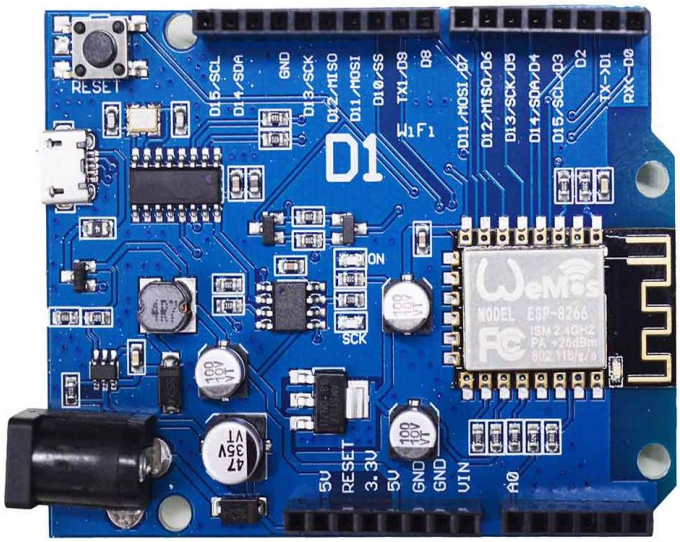
Endüstride bulut teknolojilerinin, toplanan verilerin gerçek zamanlı takibi ve depolanması için kullanımı gün geçtikçe artmaktadır. IBM Watson IoT platformu endüstrinin belirtilen ihtiyacının karşılanmasında gerekli hizmetleri sunan bulut platformlarından birisidir. IoT'ye hızlı bir giriş yapmamıza yardımcı olan kısa süreli ücretsiz bir üyelik

sağlamaktadır. Bulut platforma gönderilecek verilerin kolayca denetimi, görselleştirilmesi ve depolanması, bağlantı ve aygıt kaybı gibi yetenekler sağlamaktadır.

Watson IoT Platformu farklı dillerde çalışmak için API (Application Programming Interface)'ler sunsa da Watson IoT Platformunun desteklemediği bir dil veya platformda çalışmak istendiğinde, Watson IoT Platformuna veri gönderebilmek için MQTT protokolü kullanılması gerekmektedir. Aygıtların ve uygulamaların IBM Watson IoT Platform hizmeti ile iletişim kurmak için kullandığı birincil protokol MQTT protokolüdür.

3.6. Wemos D1

Wemos D1, ESP8266 Wi-Fi modülüne sahip geliştirme kartıdır. Wi-Fi modülü ile internete bağlanabilmektedir. İstemci ya da yayıncı olarak görev yapabilmektedir. Wemos D1 geliştirme kartı 1 adet analog girişe ve 11 adet dijital giriş/çıkışa sahiptir. Çalışma gerilimi olarak 9-24V (Voltage) arası çalışmaktadır. Hafıza olarak 4 MB (Megabyte) flash belleğe sahiptir. Tasarlanan sistemde herhangi bir pin kullanılmamıştır. Bu kart yalnızca istemci olarak çalışarak abone olunan konudan gelen verilerin alınmasında görev yapmaktadır. Arduino ve NodeMCU ile uyumludur. Karta ait görsel Şekil 3.13'te verilmiştir.



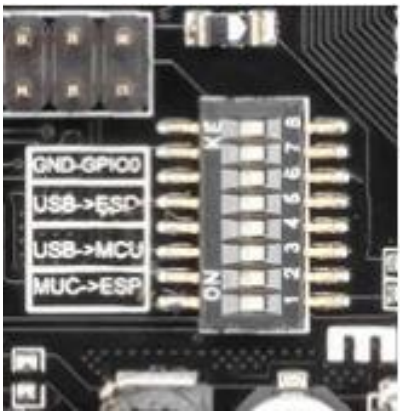
Şekil 3.13. Wemos D1 kartı

3.7. Wi-Fi Tabanlı Arduino Mega

Wi-Fi tabanlı Arduino Mega kartı normal bir Arduino Mega kartondan farklı olarak bünyesinde ekstra olarak ESP8266 Wi-Fi modülü bulundurmaktadır. Bu sayede ekstra kart kullanmadan ve bağlantılar gerektirmeden tek bir kart ile internete bağlanmamıza yardımcı olur. Bu kart ile hem Arduino Mega özellikleri kullanılabilir hem de esp8266 modülünün özelliklerini kullanılabilmektedir. Her ikisine de kod yazılarak istenildiğinde paralel, istenildiğinde ise kodlar mikro denetleyici bağımsız çalıştırılabilmektedir. Şekil 3.14'te karta ait görsel verilmiştir.

Şekil 3.14. Wi-Fi Tabanlı Arduino Mega kartı

Wi-Fi tabanlı Arduino Mega kartı üzerindeki mikro denetleyicilerden hangisine kodun atılacağı ve kodların nasıl çalıştırılacağını belirlemek için 8li DIP anahtar bulunmaktadır. Anahtara ait görsel Şekil 3.15'de verilmiştir.



Şekil 3.15. 8'li DIP anahtar

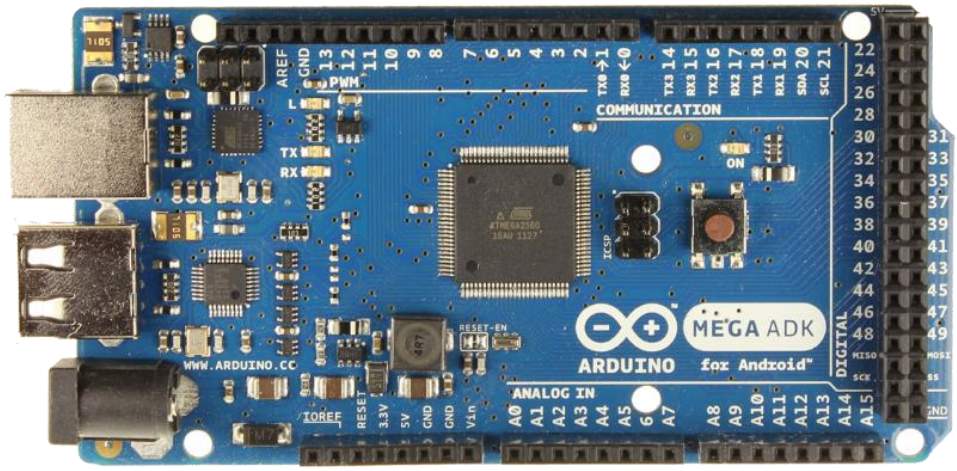
Şekil 3.15'de gösterilen anahtarlama, modları değiştirilerek kodlar karta yüklenmektedir. Modlar değiştirilerek Mega ya da ESP8266'nın beraber ya da tek tek çalışması sağlanabilir. Tablo 3.2'te anahtarlama modları verilmiştir.

Tablo 3.2. Anahtarlama modları

Yönlendirme	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
CH340 ESP8266 ya bağlı (kod yükleme)	OFF	OFF	OFF	OFF	ON	ON	ON	Kullanılmıyor
CH340 ESP8266 ya bağlı (bağlantı)	OFF	OFF	OFF	OFF	ON	ON	OFF	Kullanılmıyor
CH340 AtMega328P ye bağlı (kod yükleme)	OFF	OFF	ON	ON	OFF	OFF	OFF	Kullanılmıyor
CH340 AtMega328P ye ve COM3 ESP8266 ya bağlı	ON	ON	ON	ON	OFF	OFF	OFF	Kullanılmıyor
AtMega328P + ESP8266	ON	ON	OFF	OFF	OFF	OFF	OFF	Kullanılmıyor
Bütün modüller bağımsız	OFF	OFF	OFF	OFF	OFF	OFF	OFF	Kullanılmıyor

3.8. Arduino Mega

Arduino Mega kartı, Atmel firması tarafından üretilen bünyesinde AtMega2560 mikro denetleyici barındıran bir geliştirme kartıdır. Uno ve nano gibi diğer Arduino kartlarına göre daha kapsamlı projelerde kullanılmaktadır. 54 adet dijital giriş/çıkış ucuna ve 16 adet analog giriş ucuna sahip olmasından dolayı 3D yazıcılar ve robotik projeler gibi daha kapsamlı çalışmalar için kullanışlıdır. 54 dijital uçtan 15 tanesi PWM (Pulse Width Modulation) çıkışı sağlamaktadır. PWM uçlar 0-5V arası gerilim ve kare dalga olarak iletim sağlamaktadır. Çalışma voltajı olarak 7-12V önerilmektedir fakat limit değerleri 6-20V arasındadır. Ayrıca yaşanabilecek elektriksel problemlere karşı güçlü mimariye sahiptir. Hafıza birimleri olarak 256 KB (Kilobyte) flash bellek, 8 KB SRAM ve 4 KB EEPROM birimlerine sahiptir. Şekil 3.16'da karta ait görsel verilmiştir.



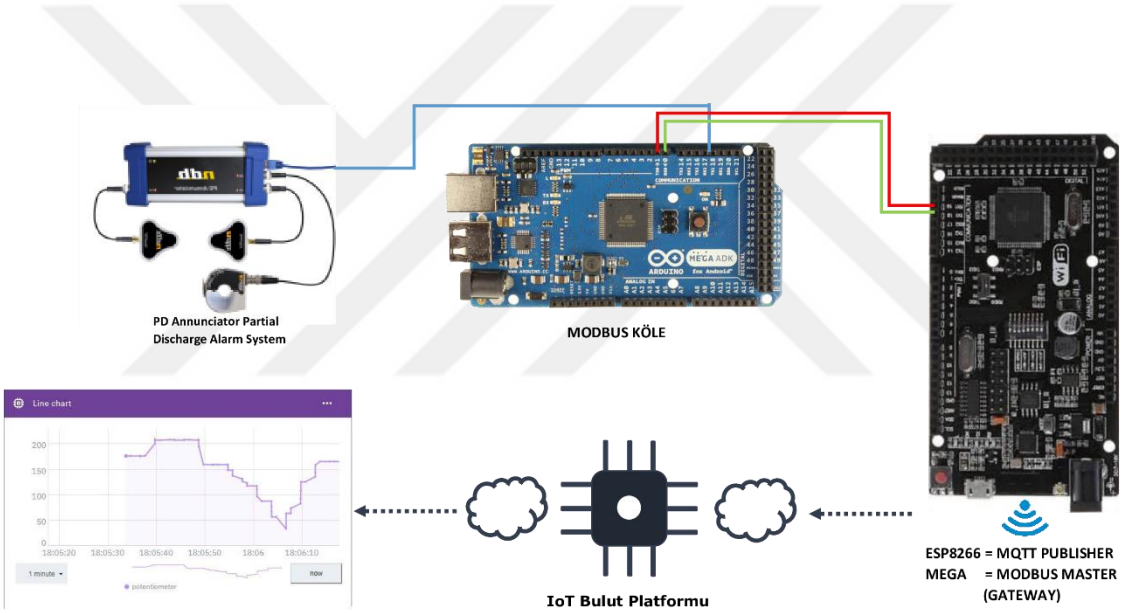
Şekil 3.16. Wi-Fi Tabanlı Arduino Mega kartı

2.8. PD Annunciator

PD Annunciator, içerdiği sensörler yardımı ile metal kaplı dolaplar ve trafo merkezi aparatları benzeri yüksek elektriğe sahip malzemelerde oluşabilecek kısmi deşarj aktivitesini tespit etmek için kullanılmaktadır. Yüksek ölçüde kısmi deşarj tespit edilirse, cihaz bu durumu tespit ederek uyarı vermektedir. Her bir cihaz üç adet sensör içermektedir. Ayrıca 247 adet cihaz birbirine seri bağlantı ile bağlanabilmektedir. PD Annunciator Modbus protokolünü desteklemekte ve kullanmaktadır.

4. IIOT SİSTEM TASARIMI

Endüstriyel uygulamalarda kullanılabilecek ağ geçidi tasarımını içeren mimarinin genel yapısı Şekil 4.1'de verilmiştir. Mimari Wi-Fi tabanlı Arduino Mega kartı, Arduino Mega kartı, PD Annunciator veri seti ve IBM Watson IoT platformu bileşenlerinin bir araya gelmesi ile oluşmaktadır. Sistemin çalışma prensibi şu şekildedir: Wi-Fi tabanlı Arduino Mega kartı iki ayrı mikro denetleyicide iki ayrı iletişim protokolü çalışmaktadır. Esp8266'da çalışmakta olan MQTT, Arduino Mega kartında çalışmakta olan Modbus protokolünden verileri toplayarak bulut platformuna göndermektedir. Bulut platformuna gönderilen veriler bu aşama sonrasında internet erişiminin olduğu herhangi bir konumdan erişilebilmektedir.



Şekil 4.1. Endüstriyel IoT ağ geçidi içeren sistem mimarisi

Bu tez çalışmasında önerilen prototip sistemin amacı, endüstriyel uygulamalarda kullanılabilecek veri iletiminde minimum gecikmeli, minimum maliyetli, ağ trafiğinde fazla yüke neden olmayacak ve yaygın olarak kullanılan protokoller ile endüstrideki birçok cihaz ile entegre çalışabilecek bir çözüm sunmaktır. Önerilen sistemde, ağ geçidi görevini üstlenen Wi-Fi tabanlı Arduino Mega kartı hem Modbus master olarak hem de MQTT yayıncı olarak çalışmaktadır. Modbus master; kart üzerinde bulunan Mega mikro denetleyicisinde, MQTT yayıncı ise ESP8266 mikro denetleyicisinde çalışmaktadır. Modbus ağındaki slave cihazlara ise, kısmi deşarj aktivitesi tespitinde kullanılan akustik sensör (Airborne) ve toprak voltajları temas sensörlerinin (Transient Earth Voltages,

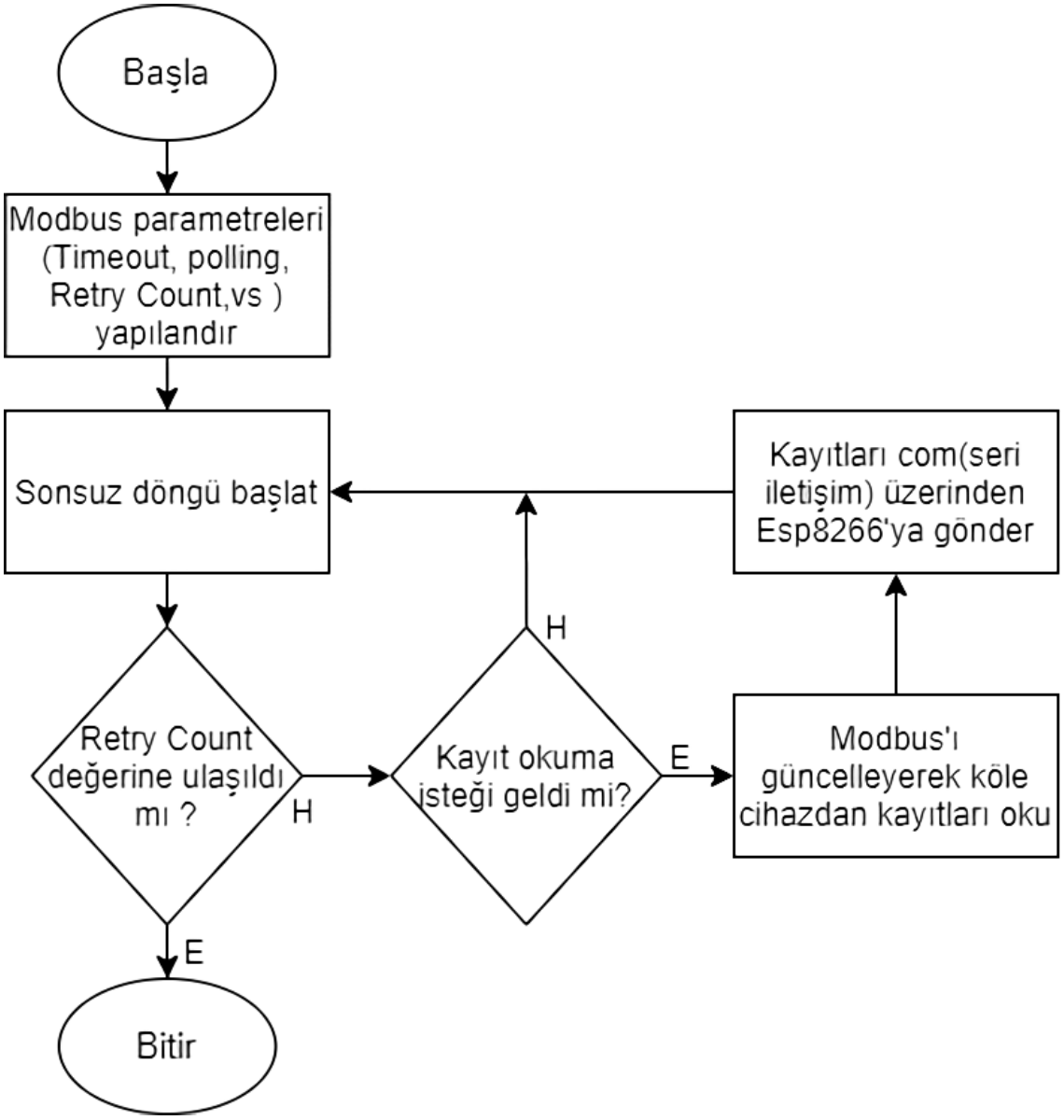
TEV) bağlı olduğu PD Annunciator cihazına ait gerçek zamanlı ölçülmüş veri seti entegre edilmiştir. Böylece slave cihaz olarak kullanılan Arduino Mega kartı mevcut sensörlerin yerini alabilmektedir. Slave cihazlardan toplanan veriler ağ geçidi yardımıyla IBM Watson IoT Platformuna aktarılmakta ve gerçek zamanlı görselleştirme elde edilmektedir.

4.1. Veri Toplama

Modbus ağında okunacak veriler, master cihaz tarafından saniyede bir kez yapılan okuma istekleri ile slave cihazlardan okunmaktadır. Modbus veri okuma isteği yapılmadan önce master cihaz üzerinde bazı parametrelerin konfigüre edilmelidir. Parametreler aşağıda verilmiştir:

- Timeout: Bir okuma isteği yapıldığında cevabın beklenme süresidir.
- Polling: Modbus okuma istediğinin sıklığını belirler. Polling rate için belirlenmiş bir minimum maksimum zaman yoktur. İsteğin türü ve ne kadar veri istendiği gibi yanıt süresini etkileyebilecek birçok faktör devreye girmektedir.
- Retry count: Okuma isteği sonucunda cevap gelmediği durumda tekrar deneme sayacı belirlenir. Eşik değerine ulaşıldığı taktirde master daha fazla istekte bulunmaz.
- Total number of register: Slave cihazlardan okunacak kayıt sayısını belirler.

Verilen parametreler; Timeout = 1000 ms (Milisaniye), polling = 1000 ms, retry count = 100 adet ve total no of register = 6, 12, 18, 24, 36, 48 adet olarak belirlendikten sonra master yaptığı her okuma isteğiyle, aldığı Modbus cevabındaki verileri kendi hafızasında bulunan “regs” olarak adlandırılan kayıt dizisine yazmaktadır. Ağ geçidi görevini yapan Wi-Fi tabanlı Arduino Mega kartındaki Mega mikro denetleyicisi, COM (Communication Port) port vasıtası ile ESP8266 modülüyle haberleşmektedir. Master, dinlediği COM port aracılığı ile MQTT’den aldığı kontrol sinyali sonrası kayıt dizisinde tuttuğu verileri string formatında tekrar isteği aldığı aynı port üzerinden MQTT’ye göndermektedir. Bu işlem ile verilerin seri iletişim kanalından gönderilmesi iki farklı iletişim protokollerinin birlikte çalışma problemini ortadan kaldırmaktadır. Veri toplama algoritmasının akış diyagramı Şekil 4.2’de verilmiştir.



Şekil 4.2. Modbus ile veri toplama

4.1. Veri Yayınlama

Geliştirilen prototip tasarımının en kritik noktası sahadan toplanan verinin bulut platformuna aktarılmasıdır. Bu noktada verinin aktarım periyodu ve veri boyutu sistem ve ağın üzerinde performans açısından önemli faktörlerdir. Çalışmada kullanılan MQTT ve Modbus protokollerinin paralel bir şekilde çalışabilmeleri için MQTT tarafında verilerin gönderim sıklığı saniyede bir kez olacak şekilde konfigürasyonu yapılmıştır. Konfigürasyonun bu şekilde yapılması ağı fazla meşgul etmez ve Modbus tarafında veri kaybının önlenmesi sağlanmıştır. Veri boyutu miktarının ağ üzerindeki olumsuz etkisinin göz önünde bulundurulması ile sistem veri üzerindeki değişikliği algılayabilen bir

yazılıma sahiptir. Sahadaki Modbus ağından toplanan her sensör verisi bir sonraki adımda kontrol edilmesi için hafızaya kaydedilmektedir. Her bir veri bir sonraki iterasyonda karşılaştırılmaktadır. MQTT ağına yeni gelen veri bir önceki veriye göre değişim göstermediği taktirde ağı gereksiz meşgul etmemek ve bulut platformda aynı verilerin çoklanmaması için MQTT ile bulut platformuna gönderilmez. Bu filtreleme yöntemi tüm farklı veri hassasiyetlerinde uygulanmaktadır.

IBM Watson IoT Platformu diğer tüm bulut platformlarda olduğu gibi belirli bir MQTT veri formatını desteklemektedir. IBM Watson IoT Platform'u, MQTT ile veri gönderiminde JSON (JavaScript Object Notation) veri formatını kullanmaktadır. Her bir iterasyonda gelen veri filtrelendikten sonra, gönderilmesine karar verildiğinde veri JSON formatına çevrilerek bulut platformuna gönderilir. Şekil 4.3'te verileri gönderiminde kullanılan algoritmanın sözde kodu verilmiştir.

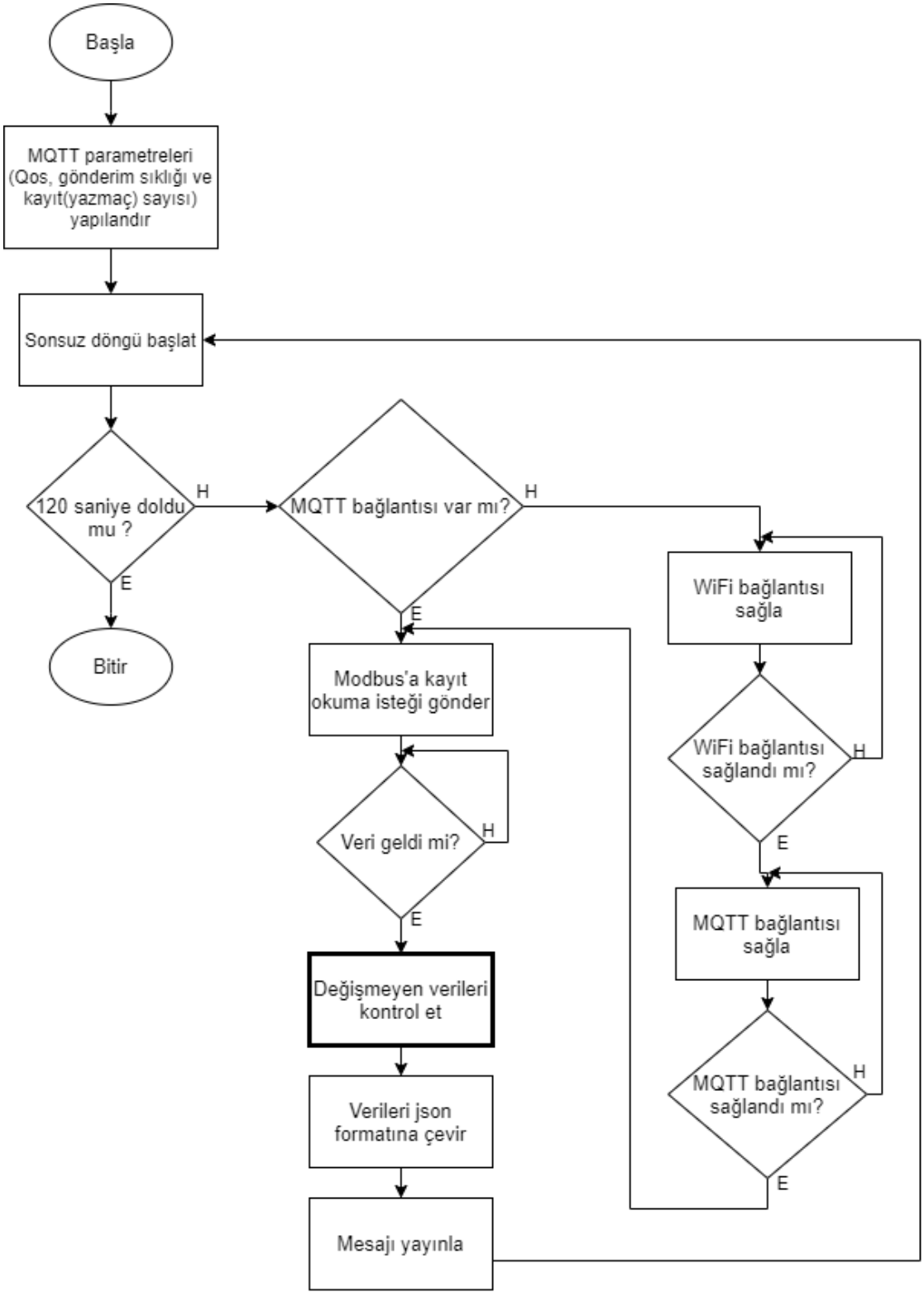
Algoritma_1 –GatewayMesajYayınlama(Data)

Giriş: Data-Modbus'ın slave cihazlardan okuduğu kayıtlar
Çıkış: Payload-Okunan kayıtları kullanılarak MQTT için hazırlanan mesaj

1. Mesaj yayınlama programını başlat
2. MQTT ve Wi-Fi bağlantıları için kullanıcı adı, parola ve sunucu parametreleri tanımla
3. MQTT ve Wi-Fi bağlantılarını sağla
4. Eğer MQTT bağlantısı yoksa 3.adıma git
5. 120 saniyelik çalışma süresi doldu mu?
6. Evet ise 12.adıma git hayır ise devam et
7. Modbus'a veri okuma isteği gönder
8. Değişmeyen verileri kontrol et
9. Gönderilmesine karar verilen verileri uygun MQTT mesaj formatına çevir
10. Belirlenen konuda (topic) yayın yap
11. 5.adıma git
12. Bitir

Şekil 4.3. Mesaj yayınlama algoritması

Şekil 4.4'te ise Şekil 4.3'te verilen algoritmanın ait akış diyagramı verilmiştir. Akış diyagramına bakıldığında ilk olarak algoritmadaki gerekli parametreler belirlenir. Ardından sonsuz döngü başlatılır ve 120 saniye boyunca devam eder. IoT platform ile bağlantı olup olmadığı kontrol edilir eğer yoksa bağlantı kurulur. Modbus'a veri okuma isteği iletilir. Veri geldiğinde verilerde bir önceki iterasyona göre değişim gösterenler uygun veri formatına dönüştürülür. MQTT mesaj paketi yayınlanır.



Şekil 4.4. MQTT ile mesaj yayınlama

4.3. Verilerin Bulut Platformunda Görselleştirilmesi

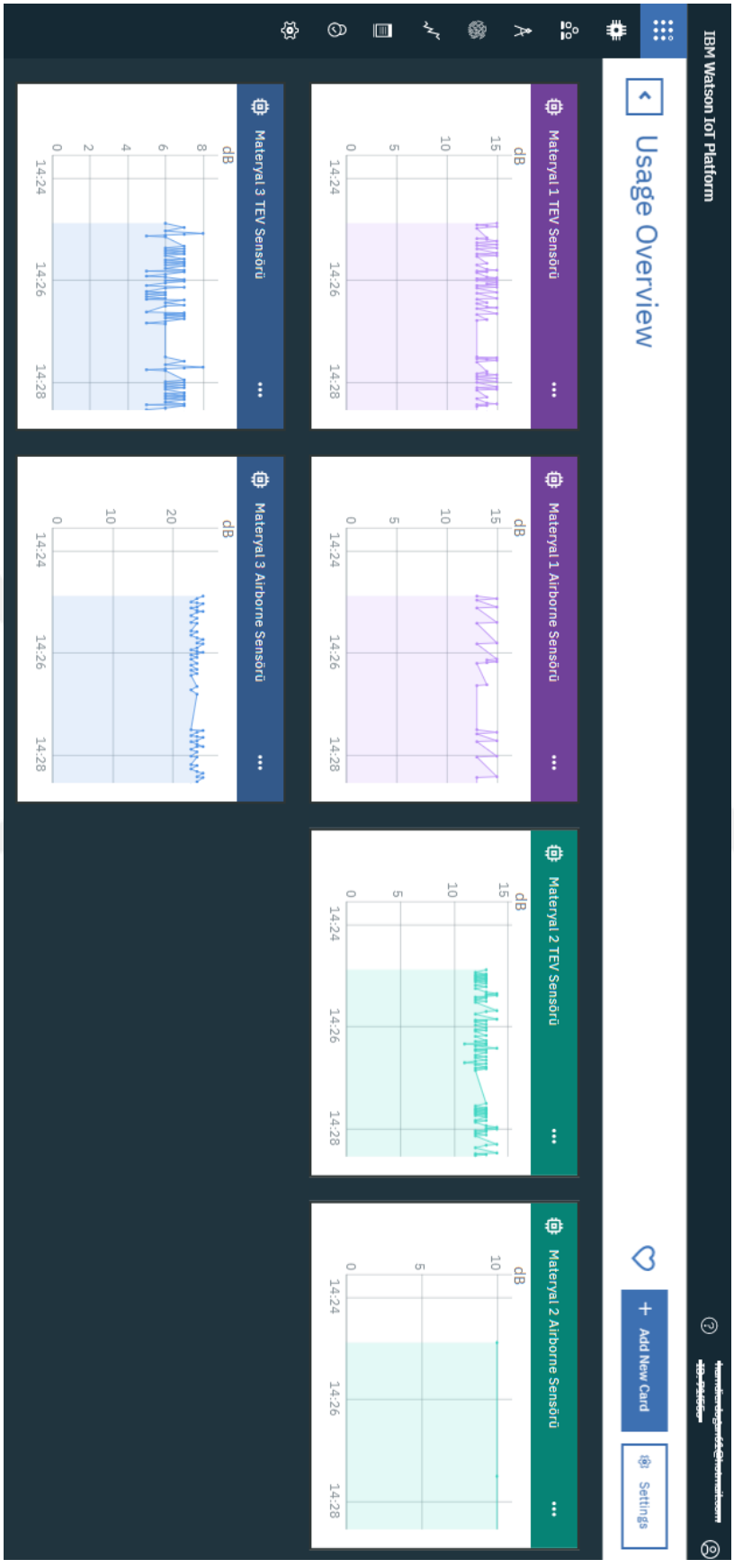
Ağ geçidi tasarımının üçüncü adımında verilerin bulut platformuna gönderilerek görselleştirilmesi için IBM Watson IoT platformu kullanılmıştır. IBM Watson IoT platformu yarıca bir MQTT sunucusu olarak çalışmaktadır. Verinin platforma gönderilmesi ile birlikte, platformda çalışan sunucuya abone olunduğu taktirde gönderilen veriye farklı bir MQTT biriminden ulaşılabilmektedir.

Verilen prototip sistemde MQTT subscriber olarak görev yapan Wemos D1 kartları ile sisteme abone olunarak, MQTT publisher tarafından platforma gönderilen verilere ulaşılmıştır.

IoT Bulut platformuna erişim sağlayabilmek için üyelik adımları tamamlandıktan sonra benzersiz bir kimlik numarası verilmektedir. Platforma veri gönderecek cihazın platform tarafında tanıtılması için bir adet isimlendirme ve cihaza ait MAC (Media Access Control) numarasına ihtiyaç duyulmaktadır. Cihaz eklenmesi sonrası sistem tarafından otomatik olarak bir adet token oluşturulur. IoT bulut platforma erişebilmek için benzersiz kimlik numarası, cihazın tanıtımında kullanılan isimlendirme ve MAC adresi bilgilerini birleştirerek istemci benzersiz kimlik numarası oluşturulur. Kullanıcı adı bilgisi olarak platformda “use-token-auth” kullanılır. Parola olarak otomatik oluşturulan token bilgisi kullanılmaktadır. Tüm bu parametreler elde edildikten sonra MQTT protokolü ile bulut platformuna bağlantı sağlanmaktadır.

Verilerin görselleştirilmesi adımında platformda grafikler oluşturulur. Grafikler oluşturulurken etkinlik (event) ve konu (topic) isimleri belirlenmektedir. Veriler etkinlik ve konu isimlerine bağlı olarak MQTT yayıncı tarafından mesaj paketi içerisinde yayınlanarak platformda oluşturulan grafiklerde görselleştirilir.

Platformda oluşturulan her bir grafikte sahadan elde edilen bir sensöre ait veri bilgileri bulunmaktadır. Şekil 4.5’te bulut platformundaki gösterge panelinde verilerin görselleştirilmesi verilmiştir.



Şekil 4.5. IBM Watson IoT platformu

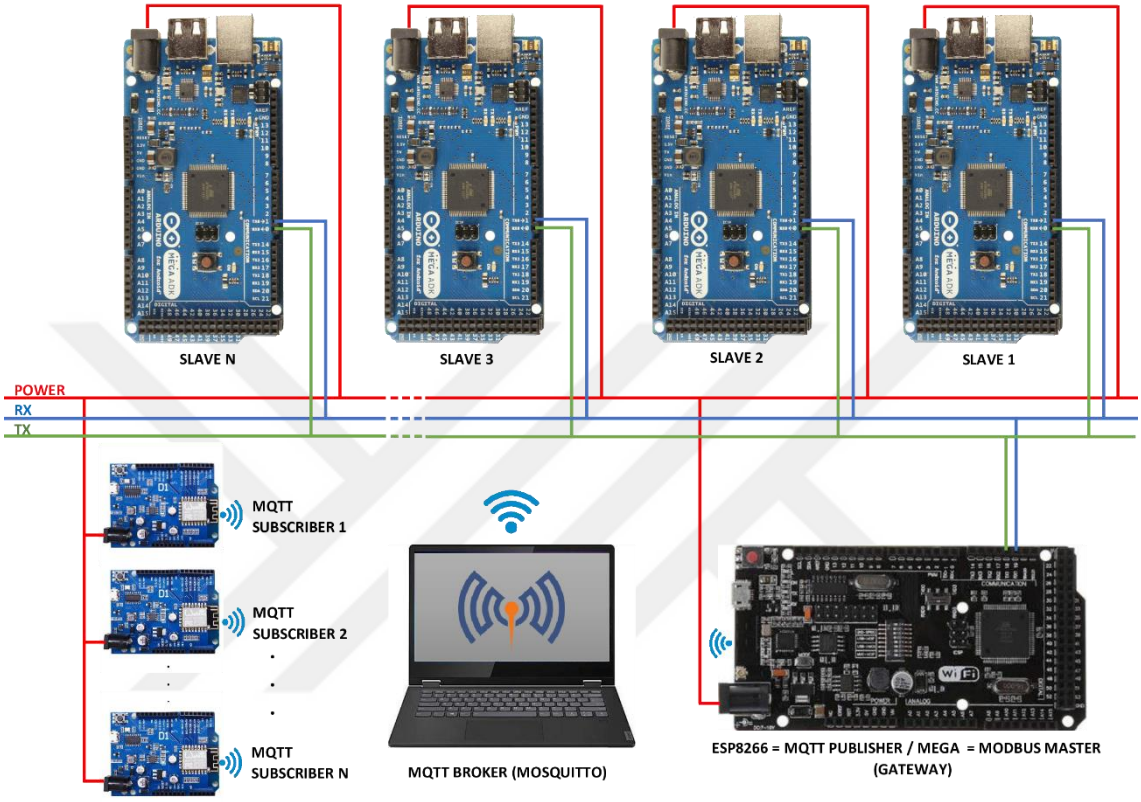
5. PERFORMANS VE TEST SONUÇLARI

Bu bölümde tasarımı önerilen ağ geçidine ait veri iletimi üzerine yapılan testler ve bu testlerde elde edilen performans sonuçları verilmiştir. Test adımlarında tasarlanan ağ geçidi prototipinde kullanılan Modbus ve MQTT protokollerine ait performansı etkileyebilecek Modbus slave sayısı, Modbus ağından okunacak kayıt sayısı, okunan verileri hassasiyeti ve MQTT protokolündeki QoS seviyesi parametrelerinde çeşitli varyasyonlar denenerek ağ geçidinin performansı elde edilmiştir. Elde edilen test sonuçları, veriler üzerindeki değişimi algılayabilen algoritma içeren sistem ve herhangi bir kontrol algoritması içermeyen sistem olarak iki farklı sistem üzerinde elde edilmiştir.

Geliştirilen ağ geçidi, Windows 10, 64 bit işletim sistemi 16 GB ram ve 128 GB SSD diske sahip dizüstü bilgisayarda yerelde çalışan Mosquitto broker'ı üzerinde test edilmiştir. Performans test sonuçları ev ortamında elde edilmiştir. Endüstriyel ortamda yapılması durumunda gürültü, titreşim, ortam malzemeleri gibi etkenler göz önünde bulundurulmalıdır. Ayrıca endüstriyel ortamda karşılaşılabilecek fazla akım gibi elektriksel olumsuzluklar dikkate alınmalıdır. Modbus ağındaki slave cihaz sayısı ve slave cihazlardan okunacak toplam kayıt sayısı performans açısından önemli rol oynamaktadır. MQTT ağında ise verilerin yayınlamasında kullanılan QoS seviyeleri ve veri boyutu miktarı performans üzerinde farklı etkiler göstermektedir. Test süresince bu parametreler üzerinde değişik senaryolar kullanılarak sisteme etkileri gözlemlenmiştir.

Modbus master, slave cihazlara gömülen PD Annunciator cihazına ait gerçek zamanlı okunan sensör verilerini okumaktadır. Test aşamasında her bir test 120 saniyelik periyodlar kullanılarak elde edilmiştir. Slave cihazlardan okunacak kayıt sayılarına karar verilmesinde gerçek veriler, sensör sayısının katları şekilde birleştirilmiştir. Okunan her bir sayısal değer slave cihaza bağlı olan bir adet sensörden elde edilen veriye karşılık gelmektedir. Performansa etki edebilecek diğer bir parametre ise slave sayısıdır. Test sonuçları master cihaza bağlı 1 ve 2 adet slave cihaz üzerinde incelenmiştir. Prototipte yapılan test süresince MQTT mesaj iletiminde QoS 0 ve QoS 1 seviyeleri kullanılmıştır. MQTT mesaj boyutunun ağ trafiği üzerinde olumsuz etkileri olmaktadır. Bu doğrultuda tasarlanan kontrol algoritmasına sahip sistemin belirtilen durumlara olumlu olarak katkı sağlaması amaçlamaktadır. Verilerin değişmemesi durumunda veri hassasiyetinin performans üzerindeki etkisi göz ardı edilmeyecek kadar önemlidir. Veri setinde bulunan

değerler üç farklı veri hassasiyetinde kullanılmıştır. Değerler tam sayı, onda bir ve yüzde bir basamak olacak şekilde bulut platformuna gönderilmektedir. Tasarlanan kontrol algoritmasına sahip olan sistemin olumlu etkisi belirtilen üç farklı veri hassasiyetinde incelenmiştir. Şekil 5.1'de test boyunca kullanılan sistem tasarımına ait model verilmiştir.



Şekil 5.1. Sistem tasarımı test yapısı

Tablo 5.1'de ağ trafiği ve sistem performansı üzerinde tasarlanan kontrol algoritmasına sahip sistemin ve herhangi bir kontrol algoritması bulunmayan sistemin etkileri verilmiştir. Elde edilen sonuçlar 120 saniyelik ölçümlere dayanmaktadır. İki sistem için, slave sayısı, QoS seviyesi, okunan kayıt sayısı ve veri hassasiyeti üzerinde farklı varyasyonlar kullanılarak bulut platformuna gönderilen toplam kayıt sayıları (her bir sensörden okunan değer) verilmiştir. Örnek olarak slave sayısı 1, QoS seviyesi 0 ve slave cihazdan okunacak kayıt sayısı 6 dikkate alındığında kontrol algoritmasına sahip olmayan sistemde bulut platformuna 120 saniye sonunda toplam 720 kayıt gönderilirken, tasarlanan kontrol algoritmasına sahip olan sistemde tam sayı, onda bir ve yüzde bir hassasiyetinde olacak şekilde sırasıyla 232, 348 ve 400 adet kayıt sayısı bulut platformuna aktarılmıştır. Tasarlanan kontrol algoritmasına sahip sistemin herhangi bir kontrol

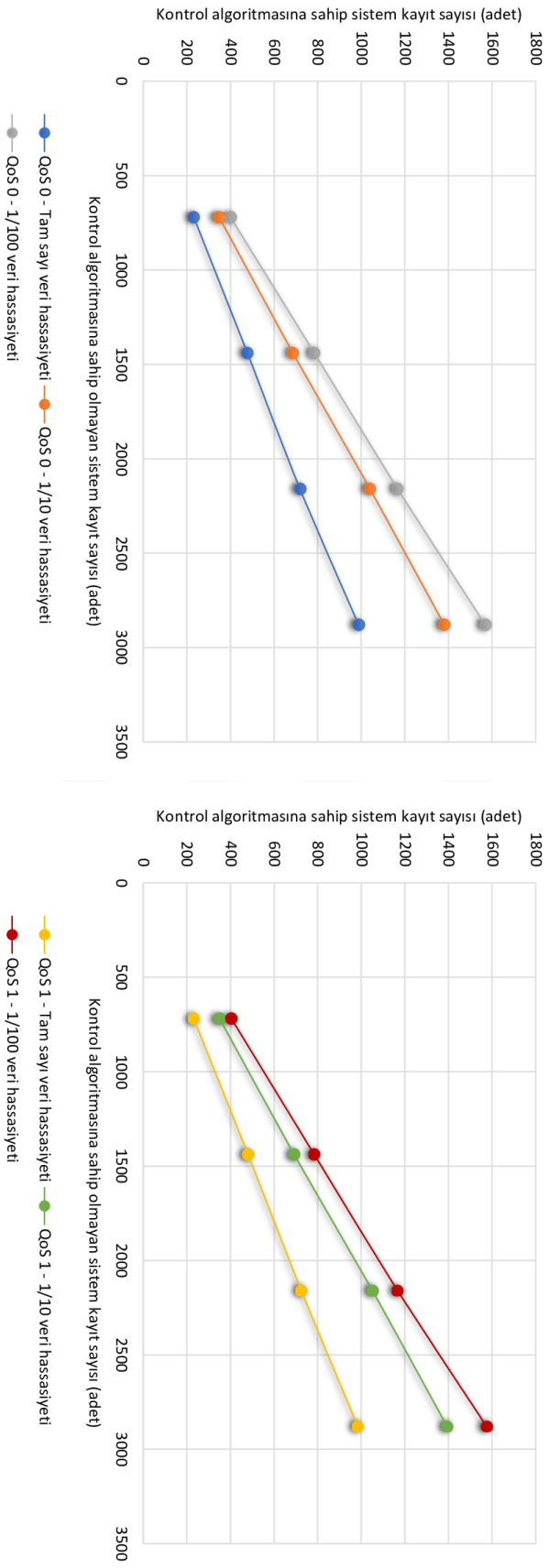
algoritması bulunmayan sisteme göre ağ trafiği üzerinde olumlu etkisi olduğu gözlemlenmektedir.

Tablo 5.1. Her iki sistem için toplam gönderilen kayıt sayıları

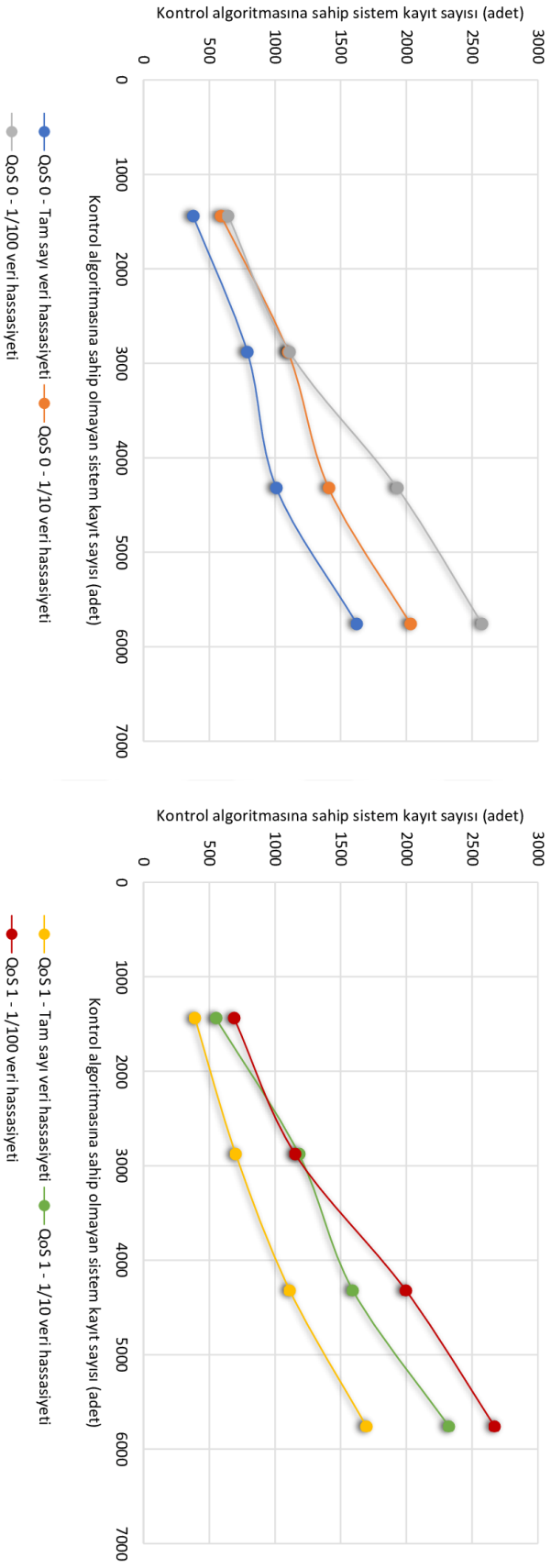
Slave Sayısı	QoS Seviyesi	Okunan Kayıt	120 Saniyede Gönderilen Toplam Kayıt Sayıları					
			Kontrol Algoritmasına Sahip Olmayan Sistem Veri Formatı			Tasarlanan Kontrol Algoritmasına Sahip Olan Sistem Veri Formatı		
			T. S.	1/10 V.H.	1/100 V.H.	T. S.	1/10 V.H.	1/100 V.H.
1	0	6		720		232	348	400
1	0	12		1440		478	687	785
1	0	18		2160		719	1040	1167
1	0	24		2880		988	1381	1569
1	1	6		720		232	351	405
1	1	12		1440		481	693	785
1	1	18		2160		725	1051	1167
1	1	24		2880		984	1394	1575
2	0	12		1440		378	591	645
2	0	24		2880		790	1100	1110
2	0	36		4320		1011	1409	1931
2	0	48		5760		1624	2031	2577
2	1	12		1440		392	551	690
2	1	24		2880		703	1183	1155
2	1	36		4320		1113	1593	1996
2	1	48		5760		1696	2323	2672

*T.S: Tam Sayı
*V.H: Veri Hassasiyeti

Şekil 5.2'da Modbus slave sayısı 1 olduğu durumda tasarlanan kontrol algoritmasına sahip olan sistemin kontrol algoritmasına sahip olmayan sisteme göre olumlu etkisi grafiksel olarak verilmiştir. Buna göre, tasarlanan kontrol algoritmasına sahip sistemin sensör verilerinin hassasiyetine göre 120 saniyelik belirlenen süre içerisinde gönderdiği toplam kayıt sayısı sunulmuştur. Veri hassasiyetleri dikkate alındığında, veriler tam sayı olarak gönderildiğinde değişim diğer iki hassasiyete göre daha az görülmektedir. Bundan kaynaklı olarak bulut platformuna gönderilen kayıt (yazmaç) sayısı, veri hassasiyeti tam sayı olduğunda diğer iki hassasiyete göre daha az sayıda gönderilmektedir. Bu durum ağ üzerindeki yükün hafiflemesine yardımcı olmaktadır.



Şekil 5.2. 1 slave için gönderilen toplam kayıt sayısı



Şekil 5.3. 2 slave için gönderilen toplam kayıt sayısı

Şekil 5.2'de slave sayısı 1 iken gönderilen toplam kayıt sayıları verilirken, Şekil 5.3'de ise slave sayısı 2 olduğunda gönderilen toplam kayıt sayılarına ait sonuçlar verilmiştir. Slave sayısı 2 olduğunda okunan kayıt sayısındaki artış bulut platformuna gönderilen toplam kayıt sayısını etkilemektedir. Ancak veri hassasiyeti yüzde bir ve QoS seviyesi 1 olduğu durum göz önüne alındığında tasarlanan kontrol algoritmasına sahip sistem, değişimi algılayamayan sisteme göre bulut platformuna gönderilen veri sayısında büyük oranda düşüş sağladığı görülmektedir. Bu doğrultuda slave sayısındaki ve kayıt sayılarındaki artışın, kontrol algoritmasına sahip sistem sayesinde ağ üzerindeki yüklenmeye etkisini minimum düzeye indirgeye yardımcı olmaktadır.

Tablo 5.2'de sistemin performansına etki edecek Modbus slave sayısı, okunan kayıt (yazmaç) sayısı, veri hassasiyeti ve MQTT QoS seviyesi parametrelerinin değişimine göre 120 saniyelik veri gönderimi sonucunda oluşan uçtan uca ortalama gecikme süreleri verilmiştir. Örnek olarak tasarlanan kontrol algoritmasına sahip sistemde slave sayısı 1, QoS seviyesi 0, okunacak kayıt sayısı 6 ve veri hassasiyeti tam sayı iken gecikme süresi 1.90 ms olarak ölçülmüştür. Aynı parametrelerde kontrol algoritmasına sahip olmayan sistemde ise ölçülen gecikme süresi 2.24 ms'dir. Elde edilen sonuçlara bakıldığında kontrol algoritmasına sahip sistemin daha performanslı olduğu açıktır. Aynı parametreler kullanılarak yalnızca QoS seviyesi 1 olduğu durumda gecikme süresinde ciddi artış gözlemlenmiştir. Bunun sebebi ise MQTT QoS 1 ile mesaj iletişiminde mesajın en az bir kez iletileceği garanti edilir. Bunu için aboneden mesajı aldığına dair cevap mesajı beklenir. Bu nedenle QoS 1 ile mesaj iletişiminde gecikme sürelerinde artış görülmektedir. Verilen parametreler kullanıldığında kontrol algoritmasına sahip olmayan sistemde gecikme süresi 61.06 ms ölçülmüştür. Aynı parametreler kullanılarak kontrol algoritmasına sahip sistemde gecikme süresi 60.94 ms'dir. Sonuçlara bakıldığında kontrol algoritmasına sahip sistem, diğer sisteme göre hem QoS 0 hem de QoS 1 seviyesinde mesaj iletişiminde olumlu sonuçlar vermiştir. Okunan kayıt sayılarındaki farkların fazla değişim göstermemesinin sebebi Arduino kullanılan Modbus Kütüphanesi bir slave cihazından maksimum 29 kayıt okuyabilir olmasıdır (Crespo, 2016). Kayıt sayılarında önemli bir artış göstermemesine rağmen QoS 0 seviyesinde tutarlı değişimler gözlemlenmektedir.

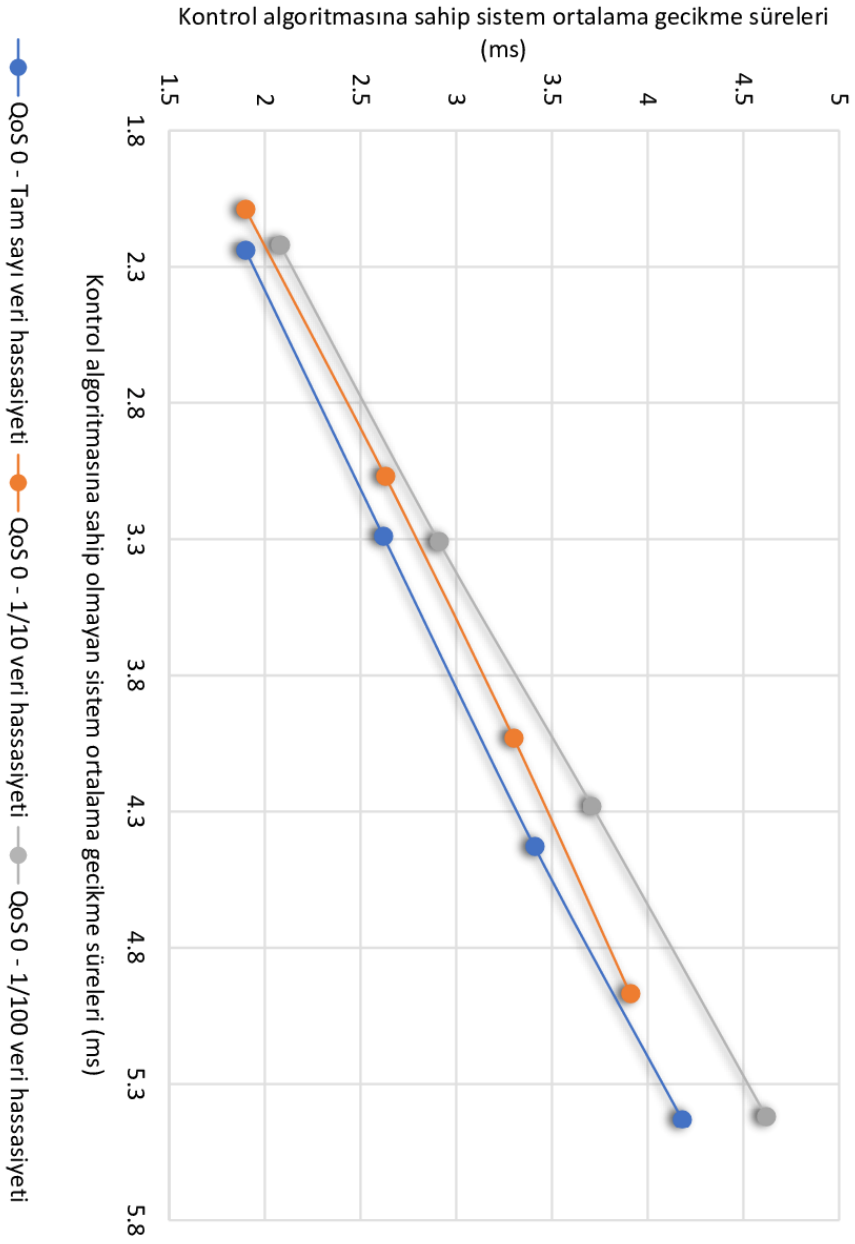
Tablo 5.2. Her iki sistem için ortalama gecikme süreleri

Slave Sayısı	QoS Seviyesi	Okunan Kayıt	120 Saniyelik İstatistiksel Sonuçlar					
			Kontrol Algoritmasına Sahip Olmayan Sistem İçin Ortalama Gecikme Süreleri (ms)			Tasarlanan Kontrol Algoritmasına Sahip Olan Sistem İçin Ortalama Gecikme Süreleri (ms)		
			T.S.	1/10 V.H.	1/100 V.H.	T.S.	1/10 V.H.	1/100 V.H.
1	0	6	2,24	2,09	2,22	1,90	1,90	2,08
1	0	12	3,29	3,07	3,31	2,62	2,63	2,91
1	0	18	4,43	4,03	4,28	3,41	3,30	3,71
1	0	24	5,43	4,97	5,42	4,18	3,91	4,62
1	1	6	61,06	61,39	61,29	60,94	61,09	61,45
1	1	12	60,92	61,48	61,35	61,29	61,33	61,76
1	1	18	61,42	61,68	61,81	61,51	61,55	61,92
1	1	24	62,08	61,78	62,22	61,67	61,63	62,03
2	0	12	3,43	3,07	3,34	2,65	2,52	2,81
2	0	24	5,5	5,07	5,50	3,92	3,69	6,32
2	0	36	7,46	6,97	7,55	5,38	4,93	5,87
2	0	48	9,79	8,73	9,67	6,89	6,21	7,36
2	1	12	61,28	61,13	61,07	61,21	61,22	62,26
2	1	24	61,88	61,83	62,06	61,97	61,39	62,78
2	1	36	63,29	62,79	64,01	62,65	62,03	63,26
2	1	48	64,55	64,73	65,35	63,48	62,62	64,93

*T.S: Tam Sayı
*V.H: Veri Hassasiyeti

Tablo 5.1 ve Tablo 5.2 tablolarında verilen değerler test süresince elde edilen sonuçlardan oluşturulan özet tablolardır. Bu tablolarda yalnızca uçtan uca gönderim süreleri ve gönderilen toplam kayıt sayılarına ait değerler içermektedir.

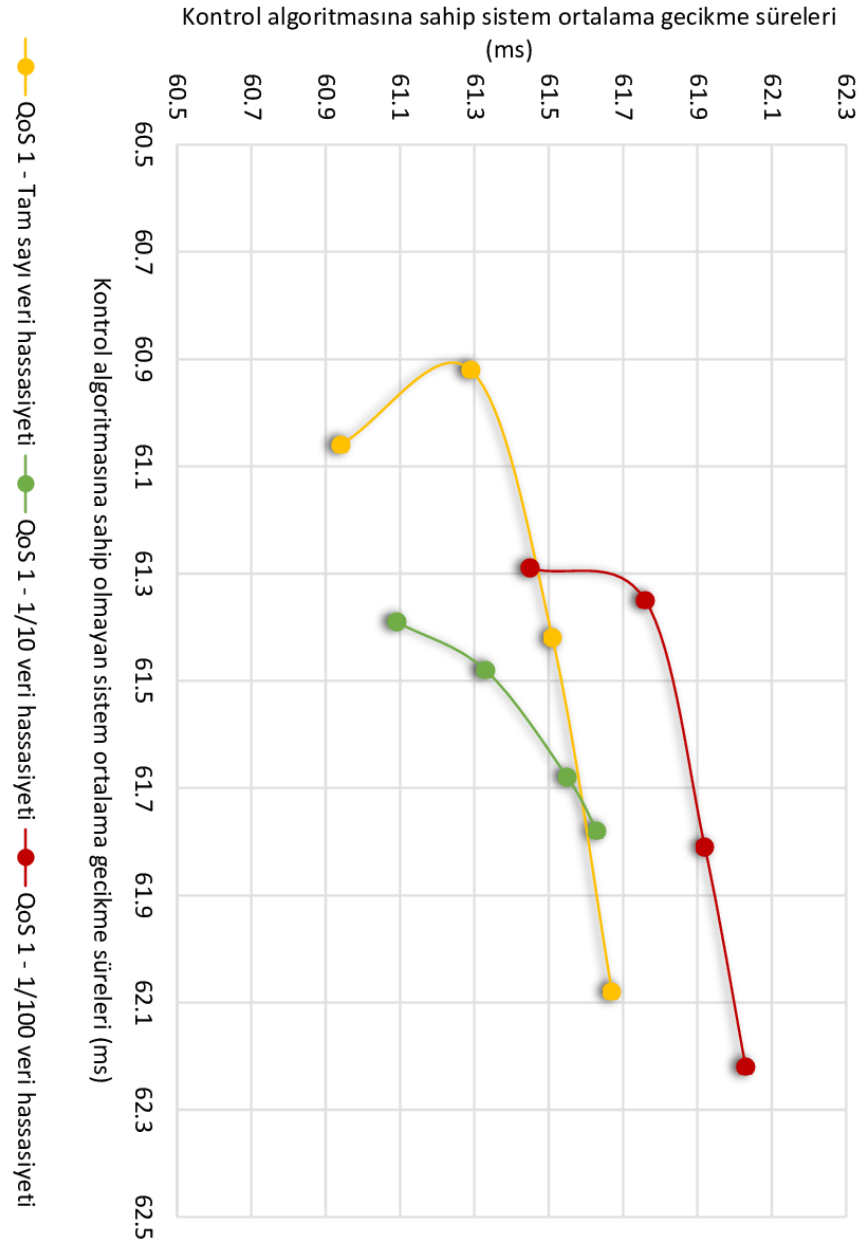
Şekil 5.4'te, Tablo 5.2'deki farklı parametreler ile elde edilmiş uçtan uca gecikme sürelerinin yalnızca slave sayısı 1 olduğu ve QoS parametresinin 0 seçildiği durumda elde edilmiş verilerin grafiksel gösterimi verilmiştir. Grafikte görüldüğü üzere, beklenildiği gibi veri hassasiyetine göre eğri sıralaması tam sayı, 1/10 basamak ve 1/100 basamak olarak sıralanmıştır. En hızlı veri iletimi eğrisi en aşağıda kalandır. Sebebi ise kontrol algoritmasına ait sistemin tam sayı hassasiyetinde daha fazla benzer veri tespit ederek daha az veri gönderilmesini sağlaması ile paket boyutunun azalmasıdır.



Şekil 5.4. 1 slave ve QoS 0 için uçtan uca gecikme süreleri

Şekil 5.5'te, Tablo 5.2'de verilen uçtan uca gecikme sürelerinin slave sayısı 1 olduğu ve QoS parametresinin 1 seçildiği durumda elde edilen verilerin grafiksel gösterimi verilmiştir. Grafikteki sonuçlar ele alındığında 1/10 veri hassasiyetinde veri iletiminde elde edilen uçtan uca gecikme süreleri tam sayı veri hassasiyeti ile veri iletimi ile karşılaştırıldığında daha hızlı iletildiği görünmektedir. Bu duruma sebep olarak, gönderilen veri sayısının azlığı, paket yükünün hafifliği ve test aşamasında internette yaşanan dalgalanmalar verilebilir. Genele bakıldığında ise 1/100 veri hassasiyeti ile veri

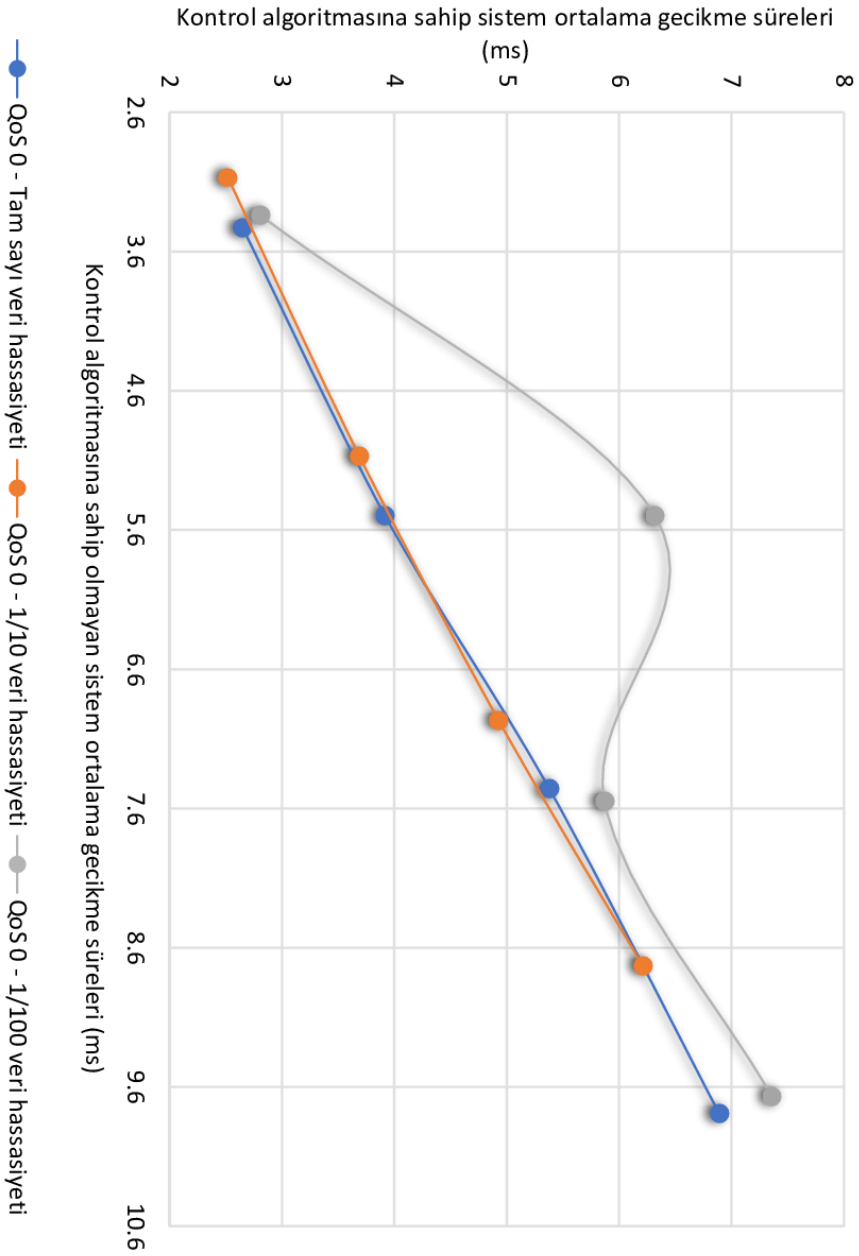
iletimi tekrar en uzun süreli veri iletimi olarak görünmektedir. Verilerin değişimin az olması bu duruma etki olarak gösterilebilir.



Şekil 5.5. 1 slave ve QoS 1 için uçtan uca gecikme süreleri

Şekil 5.6'te, Tablo 5.2'dek verilen uçtan uca gecikme sürelerinin yalnızca slave sayısı 2 olduğu ve QoS parametresinin 0 seçildiği durumda hesaplanan verilerin grafiksel gösterimi verilmiştir. Grafikte veri hassasiyeti tam sayı ve 1/10 olduğu durum ele alındığında eğriler birbirlerine çok yakındır. Hassasiyetin birbirine yakın olması ile veri

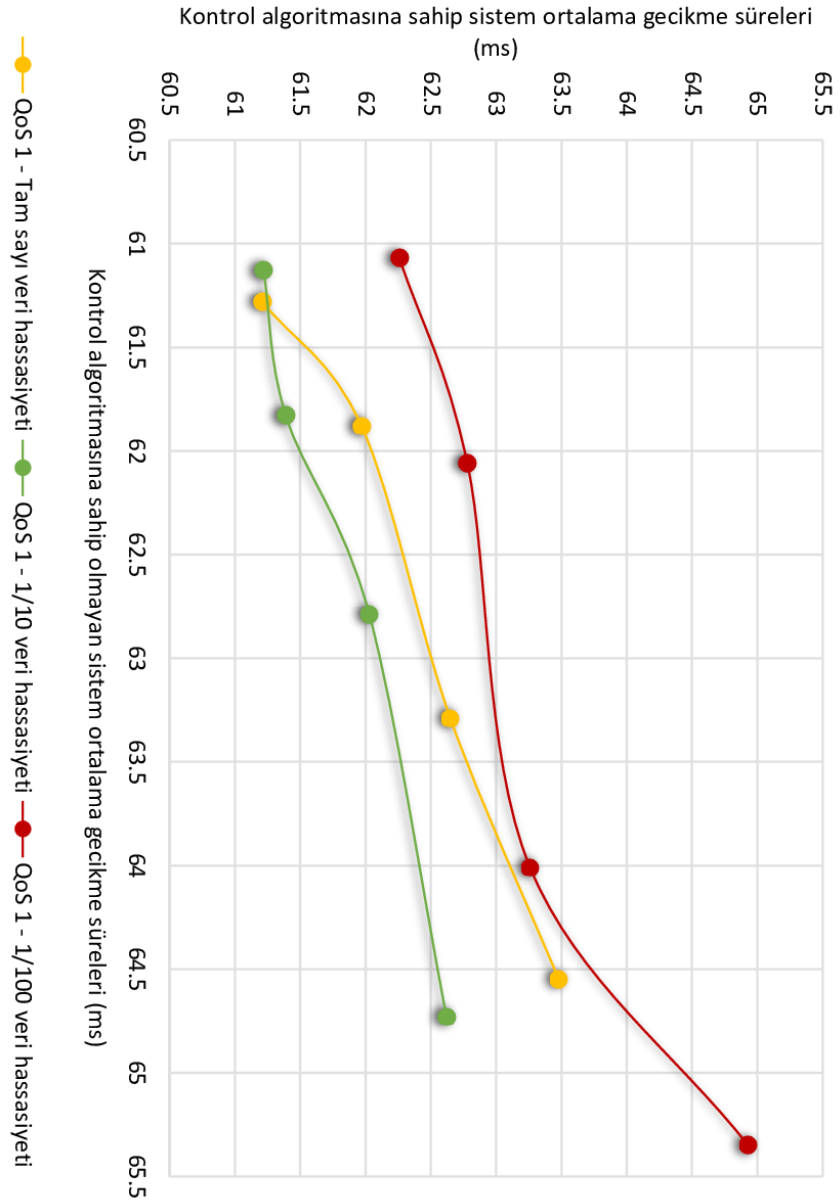
değişimi 1/100 olduğu duruma göre daha azdır. Bu nedenler her iki eğri birbirine çok yakındır. Daha detaya bakıldığında 3.iterasyonda 1/10 veri hassasiyetinde veri iletimi daha hızlı gerçekleşmiştir. Bu duruma neden olarak daha önce belirtildiği gibi paket boyutunun az olması ve internet üzerindeki dalgalanmalar neden olmaktadır.



Şekil 5.6. 2 slave ve QoS 0 için uçtan uca gecikme süreleri

Şekil 5.7'te, Tablo 5.2'deki çeşitli varyasyonlar ile elde edilmiş uçtan uca gecikme sürelerinin slave sayısı 2 olduğu ve QoS parametresinin 1 seçildiği durumda elde edilmiş

verilerin grafiksel gösterimi verilmiştir. Grafikte görüldüğü üzere QoS 1 ile veri iletiminde uçtan uca gecikme süreleri QoS 0 ile veri iletimine göre çok daha uzun sürmektedir. Nedeni ise QoS 1 ile veri iletiminde alıcıdan mesajı aldığına dair beklenen PUBACK mesajıdır. QoS 0 ile veri iletiminde ortalama süre 5 milisaniye seviyesinde iken QoS 1 ile veri iletiminde bu süre ortalama 62 milisaniye seviyelerindedir. Grafiğin geneline bakıldığında ise beklendiği gibi minimum veri iletim süreleri sırası ile tam sayı, 1/10 ve 1/100 veri hassasiyetinde elde edilmiştir.



Şekil 5.7. 2 slave ve QoS 1 için uçtan uca gecikme süreleri

Tablo 5.3'te akıllı sistem ile elde edilmiş detaylı sonuçlar verilmiştir. Test aşamasında 120 saniye boyunca test verileri virgülden itibaren 1 basamak hassasiyetle gönderilmiştir ve aynı zamanda istatistiksel sonuçları da hesaplanmıştır. Detaylı sonuçların verildiği tablolardaki kolon ifadelerin açıklaması;

1. Slave: Test aşamasında her iterasyonda kullanılan toplam Modbus slave sayısını ifade eder
2. QoS: Farklı varyasyonlar için belirlenen ve ilgili iterasyonlarda kullanılan QoS seviyesi
3. Ort: 120 saniye sonucunda elde edilen milisaniye cinsinden ortalama mesaj iletim süresi
4. Min: 120 saniye sonucunda elde edilen milisaniye cinsinden minimum mesaj iletim süresi
5. Max: 120 saniye sonucunda elde edilen milisaniye cinsinden maksimum mesaj iletim süresi
6. S.S: 120 saniye sonucunda elde edilen milisaniye cinsinden standart sapma değerini ifade eder
7. Var: 120 saniye sonucunda elde edilen milisaniye cinsinden varyans değerini ifade eder
8. G.G: Farklı varyasyonlar için belirlenen parametreler ile gönderilmesi gereken toplam kayıt sayısını ifade eder
9. G: Farklı varyasyonlar için belirlenen parametreler ile gönderilen toplam kayıt sayısını ifade eder

Tablo 5.3 incelendiğinde, örnek olarak slave sayısı 1, QoS 0 ve okunan kayıt sayısı 6 olduğunda ortalama uçtan uca gecikme süresi ortalama 1.9 milisaniye seviyesindedir. Ayrıca tabloda verilen min max kolonlarında, 120 saniyelik test sürecinde elde edilen minimum ve maksimum veri iletim süreleri verilmiştir. Her iki kolon değerleri üzerinde servis kalitesi ve kayıt sayılarının etkisi standart sapmanın hesaplandığı kolonda görülmektedir.

Standart sapma, bize elde edilen sürelerin ortalamadan ne kadar saptığı ifade etmektedir. Bu doğrultuda, QoS ve okunan kayıt sayılarının etkisi takip edilebilmektedir.

Tablo 5.3. Akıllı sistem gecikme süreleri (virgülden sonra bir basamak)

Slave	QoS	Kayıt	Ort (ms)	Min (ms)	Max (ms)	S.S (ms)	Var (ms)	G.G	G.
1	0	6	1,90	1	3	0,44	0,19	720	348
1	0	12	2,63	2	4	0,50	0,25	1440	687
1	0	18	3,30	3	4	0,46	0,21	2160	1040
1	0	24	3,91	3	5	0,45	0,20	2880	1381
1	1	6	61,09	51	71	3,09	9,53	720	351
1	1	12	61,33	55	78	2,90	8,44	1440	693
1	1	18	61,55	51	78	3,79	14,38	2160	1051
1	1	24	61,63	51	86	4,70	22,12	2880	1394
2	0	12	2,52	2	3	0,50	0,25	1440	591
2	0	24	3,69	3	5	0,53	0,28	2880	1100
2	0	36	4,93	4	6	0,54	0,30	4320	1409
2	0	48	6,21	5	8	0,68	0,46	5760	2031
2	1	12	61,22	53	79	3,28	10,77	1440	551
2	1	24	61,39	54	89	4,03	16,27	2880	1183
2	1	36	62,03	50	82	5,51	30,34	4320	1593
2	1	48	62,62	53	102	5,95	35,45	5760	2323

*M.S: Milisaniye

Tablo 5.4'te standart sistem ile elde edilmiş detaylı sonuçlar verilmiştir. Test aşamasında 120 saniye boyunca test verileri virgülden itibaren 1 basamak hassasiyetle gönderilmiştir.

Tablo 5.4'te farklı olarak standart sistem gelen veriyi filtrelemeden direkt gönderdiği için yalnızca toplam gönderilen kayıt sayısı bulunmaktadır. Diğer kolonlar akıllı sistem detayları ile aynıdır. Standart sistem ile akıllı sistem uçtan uca gecikme süreleri Tablo 5.3 ve Tablo 5.4'e bakılarak karşılaştırıldığında slave sayısı 1, QoS 1 ve okunan kayıt sayısı 6 olduğu durum ele alındığında, akıllı sistemde ortalama gecikme süresi 61.09 milisaniye iken standart sistemde bu değer 61.39 milisaniyedir. Ayrıca akıllı sistemde toplam gönderilen kayıt sayısı standart sisteme göre 372 kayıt daha azdır. Verilen örneğe bakıldığında kontrol algoritmasına sahip olan akıllı sistemin, kontrol algoritmasına sahip

olmayan sisteme göre daha başarılı sonuçlar verdiği görülmektedir. Diğer detaylı tablolarda farklı örnekler üzerinde incelendiğinde de bu durum görülmektedir.

Tablo 5.4. Standart sistem gecikme süreleri (virgülden sonra 1 basamak)

Slave	QoS	Kayıt	Ort (ms)	Min (ms)	Max (ms)	S.S (ms)	Var (ms)	G
1	0	6	2,09	2	3	0,29	0,08	720
1	0	12	3,07	2	4	0,31	0,10	1440
1	0	18	4,03	3	5	0,43	0,18	2160
1	0	24	4,97	4	6	0,44	0,18	2880
1	1	6	61,39	52	84	3,99	15,92	720
1	1	12	61,48	55	77	3,14	9,83	1440
1	1	18	61,68	54	77	3,39	11,52	2160
1	1	24	61,78	56	80	3,88	15,04	2880
2	0	12	3,07	2	4	0,36	0,13	1440
2	0	24	5,07	4	6	0,38	0,15	2880
2	0	36	6,97	6	8	0,51	0,26	4320
2	0	48	8,73	8	10	0,46	0,21	5760
2	1	12	61,13	54	72	2,29	5,23	1440
2	1	24	61,83	53	85	3,92	15,41	2880
2	1	36	62,79	54	78	5,06	25,60	4320
2	1	48	64,73	57	79	5,82	33,91	5760

*M.S: Milisaniye

Tablo 5.5'te akıllı sistem ile elde edilmiş detaylı sonuçlar verilmiştir. Test aşamasında 120 saniye boyunca test verileri virgülden itibaren 2 basamak hassasiyetle gönderilmiştir. En yüksek veri iletim süreleri 1/100 veri hassasiyetinde elde edilmektedir. Örnek olarak slave sayısı 2, QoS 1 ve okuna kayıt sayısı 12 olduğunda maksimum veri iletim süresi 86 milisaniye olarak elde edilmiştir. Bu değer 1/10 veri hassasiyetinde 79 milisaniyedir. Daha az veri değişiminden kaynaklı paket boyutunun daha az olması bu duruma bir etkendir.

Tablo 5.5. Akıllı sistem gecikme süreleri (virgülden sonra 2 basamak)

Slave	QoS	Kayıt	Ort (ms)	Min (ms)	Max (ms)	S.S (ms)	Var (ms)	G.G	G.
1	0	6	2,08	1	3	0,38	0,14	720	400
1	0	12	2,91	2	4	0,43	0,18	1440	785
1	0	18	3,71	3	5	0,51	0,26	2160	1167
1	0	24	4,62	3	5	0,50	0,25	2880	1569
1	1	6	61,45	54	84	3,64	13,21	720	405
1	1	12	61,76	52	88	4,75	22,60	1440	785
1	1	18	61,92	52	93	5,49	30,15	2160	1167
1	1	24	62,03	51	80	4,85	23,48	2880	1575
2	0	12	2,81	2	4	0,50	0,25	1440	645
2	0	24	6,32	3	5	0,52	0,27	2880	1110
2	0	36	5,87	5	8	0,64	0,42	4320	1931
2	0	48	7,36	6	9	0,81	0,66	5760	2577
2	1	12	62,26	49	86	4,86	23,63	1440	690
2	1	24	62,78	53	87	5,60	31,40	2880	1155
2	1	36	63,26	53	99	6,58	43,24	4320	1996
2	1	48	64,93	56	100	7,84	61,53	5760	2672

*M.S: Milisaniye

Tablo 5.6'da standart sistem ile elde edilmiş detaylı sonuçlar verilmiştir. Test aşamasında 120 saniye boyunca test verileri virgülden itibaren 2 basamak hassasiyetle gönderilmiştir. Tablo 5.6'da verilen test sonuçlarına göz atıldığında ve slave sayısı 1 olduğu durum özelinde incelendiğinde, QoS 0 ve QoS 1 arasındaki standart sapma değerleri göze çarpmaktadır. QoS 1 seçildiği durumda yöntem gereği PUBACK paketinin beklenmesi durumunda iletişimdeki gecikmeler nedeniyle bazı durumlarda uçtan uca gecikme sürelerindeki minimum ve maksimum değerleri, QoS 0 seçildiği duruma göre çok daha fazla sapmalara sebep olmaktadır. Bu doğrultuda standart sapma değeri yüksek çıkmaktadır. Neticede QoS seçiminin performans üzerinde önemli etkisi olduğu gözlemlenmiştir.

Tablo 5.6. Standart sistem gecikme süreleri (virgülden sonra iki basamak)

Slave	QoS	Kayıt	Ort (ms)	Min (ms)	Max (ms)	S.S (ms)	Var (ms)	G
1	0	6	2,22	2	3	0,42	0,17	720
1	0	12	3,31	3	4	0,46	0,21	1440
1	0	18	4,28	4	5	0,45	0,20	2160
1	0	24	5,42	5	6	0,49	0,24	2880
1	1	6	61,29	50	94	4,20	17,66	720
1	1	12	61,35	53	93	4,55	20,71	1440
1	1	18	61,81	53	81	4,71	22,19	2160
1	1	24	62,22	53	87	5,24	27,46	2880
2	0	12	3,34	3	5	0,49	0,24	1440
2	0	24	5,50	5	7	0,56	0,32	2880
2	0	36	7,55	7	8	0,50	0,25	4320
2	0	48	9,67	9	10	0,47	0,22	5760
2	1	12	61,07	52	75	2,77	7,66	1440
2	1	24	62,06	55	92	5,11	26,07	2880
2	1	36	64,01	55	95	6,63	43,99	4320
2	1	48	65,35	57	97	7,15	51,08	5760

*M.S: Milisaniye

Tablo 5.7'de akıllı sistem ile elde edilmiş detaylı sonuçlar verilmiştir. Test aşamasında 120 saniye boyunca test verileri virgüllü ifade harici tam sayı olacak şekilde gönderilmiştir.

Tablo 5.7'de verilen sonuçlar incelendiğinde ve slave sayısı 2 ve QoS 1 seçildiğinde gönderilmesi gereken toplam kayıt sayısı 1440 iken, kontrol algoritmasına sahip sistemde aynı şartlarda gönderilen kayıt sayısı 392 olarak elde edilmiştir. İstatistiksel olarak hesaplandığında kontrol algoritmasına sahip sistemin slave sayısı 2 ve QoS 1 olduğu durumda %72.8 oranında iyileşme gösterdiği görülmektedir. Bu doğrultuda kontrol algoritmasına sahip olan sistemin, kontrol algoritmasına sahip olmayan sisteme göre daha başarılı olduğu görülmektedir.

Tablo 5.7. Akıllı sistem gecikme süreleri (tam sayı)

Slave	QoS	Kayıt	Ort (ms)	Min (ms)	Max (ms)	S.S (ms)	Var (ms)	G.G	G.
1	0	6	1,90	1	3	0,37	0,25	720	232
1	0	12	2,62	2	4	0,52	0,27	1440	478
1	0	18	3,41	3	5	0,52	0,27	2160	719
1	0	24	4,18	3	6	0,49	0,24	2880	988
1	1	6	60,94	53	73	2,60	6,74	720	232
1	1	12	61,29	55	84	3,44	11,86	1440	481
1	1	18	61,51	50	78	4,14	17,15	2160	725
1	1	24	61,67	56	83	4,00	15,99	2880	984
2	0	12	2,65	2	4	0,49	0,24	1440	378
2	0	24	3,92	3	5	0,49	0,24	2880	790
2	0	36	5,38	4	7	0,57	0,32	4320	1011
2	0	48	6,89	6	8	0,63	0,40	5760	1624
2	1	12	61,21	57	85	3,02	9,11	1440	392
2	1	24	61,97	55	85	4,34	18,80	2880	703
2	1	36	62,65	56	86	5,15	26,54	4320	1113
2	1	48	63,48	56	94	6,52	42,57	5760	1696

*M.S: Milisaniye

Tablo 5.8’de standart sistem ile elde edilmiş detaylı sonuçlar verilmiştir. Test aşamasında 120 saniye boyunca test verileri virgüllü ifade harici tam sayı olacak şekilde gönderilmiştir.

Akıllı sistem test sonuçlarının bulunduğu Tablo 5.7 ve standart sistem test sonuçlarının bulunduğu Tablo 5.8’deki veriler karşılaştırıldığında slave sayısı 2 ve QoS 0 olarak ele alındığında, akıllı siste uçtan uca ortalama gecikme süresi 2.65 milisaniye standart sistemde ise 3.43 milisaniye olarak elde edilmiştir. Toplam gönderilen kayıt sayılarının daha az olduğu akıllı sistemin olumlu etkisi örnekte verilmiştir. Sistemin başarısı hem uçtan uca ortalama gecikme sürelerinde hem de gönderilen toplam kayıt sayılarında görülmektedir.

Tablo 5.8. Standart sistem gecikme süreleri (tam sayı)

Slave	QoS	Kayıt	Ort (ms)	Min (ms)	Max (ms)	S.S (ms)	Var (ms)	G
1	0	6	2,24	2	3	0,43	0,18	720
1	0	12	3,29	3	4	0,45	0,21	1440
1	0	18	4,43	4	7	0,60	0,36	2160
1	0	24	5,43	5	6	0,49	0,24	2880
1	1	6	61,06	53	74	3,01	9,09	720
1	1	12	60,92	56	70	2,10	4,39	1440
1	1	18	61,42	51	85	4,37	19,06	2160
1	1	24	62,08	54	85	5,00	24,98	2880
2	0	12	3,43	3	6	0,54	0,30	1440
2	0	24	5,5	5	6	0,50	0,25	2880
2	0	36	7,46	7	8	0,50	0,25	4320
2	0	48	9,79	9	11	0,45	0,20	5760
2	1	12	61,28	55	80	3,84	14,77	1440
2	1	24	61,88	54	82	4,64	21,57	2880
2	1	36	63,29	56	83	5,61	31,47	4320
2	1	48	64,55	57	82	6,08	37,01	5760

**M.S: Milisaniye*

6. SONUÇLAR VE ÖNERİLER

Bu çalışmada endüstriyel IoT uygulamalarında kullanılabilecek verilerin sahadan toplanarak bulut platformuna aktarılmasını sağlayacak bir ağ geçidi prototipi sunulmuştur. Prototip içerisinde barındırdığı kontrol algoritmasına sahip sistem ile kontrol algoritmasına sahip olmayan bir ağ geçidine göre amaçlandığı doğrultuda daha performanslı sonuçlar vermiştir. Hem bulut platformuna gönderilen kayıt sayısı hem de ortalama gecikme süresi açısından olumlu sonuçlar elde edilmiştir. Mesajların gönderilmesinde ağ trafiğindeki iyileşmeye bakıldığında, bir mesajın boyutu ortalama 50 bayttır. Genel olarak bir örnek üzerinde bakıldığında Tablo 5.1'de veri hassasiyeti 1/10 iken kontrol algoritmasına sahip olan sistem, kontrol algoritmasına sahip olmayan sisteme göre ağ trafiğinde ortalama %39,69'luk iyileşme göstermiştir. Tasarlanan prototip, bulut platformlarına bağlantı desteği sağladığı için sahada toplanan verilerin istenilen yer ve zamanda gerçek zamanlı olarak izlenmesini sağlamaktadır. Aynı zamanda sistem Arduino tabanlı olduğu için düşük maliyetlidir. Sistem, IIoT uygulamalarında sıkça kullanılan Modbus ve MQTT protokollerinin bir arada çalışmasını destekleyerek IoT'deki birlikte çalışabilirlik problemine çözüm sağlamaktadır. Uygulamalarda kullanılabilirlik açısından, endüstride önerilen kompleks ve maliyetli sistemlerle karşılaştırıldığında hem maliyet hem de performans parametreleri göz önüne alındığında tercih edilebilir.

KAYNAKLAR

Agashe, S., Rawale, M., Tamboli, S., Thoraiet, R. (2015). Implementation of Modbus RTU and Modbus TCP Communication using Siemens S7-1200 PLC for Batch Process. *2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*, Avadi,Chennai, India, 06-08 May 2015.

Akın, Ö., Aslan, E., Sokullu, R. (2021). Face Recognition Based Multifunction IoT Smart Mailbox. *Journal of Emerging Computer Technologies, 1* (2), 34-37.

Akkaş, M. A., Çiftçi, B., Şen, Z. (2021). Nesnelerin İnterneti Tabanlı Kablosuz Taşınabilir EKG Cihazı. *Avrupa Bilim ve Teknoloji Dergisi*, 26(26), 91-95.

Aledhari, M., Al-Fuqaha, A. Ayyash, M., Guizani, M., Mohammadi, M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys & Tutorials, 17*, 2347-2376.

Arslan S., Aydemir F. (2021). COVID-19 Pandemi Sürecinde Çocukların El Yıkama Alışkanlığının Nesnelerin İnterneti Tabanlı Sistem ile İzlenmesi. *Mühendislik Bilimleri ve Araştırmaları Dergisi, 3*(2), 161-168.

Astarloa, A., Bidarte, U., Jimenez, J., Lazaro, J. Zuloaga, A. (2016). Intelligent gateway for Industry 4.0-compliant production. *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, Italy, 23-26 October 2016.

Aydın A., Göktaş Y., Usanmaz B. (2021). Nesnelerin İnterneti'nin eğitimde kullanıldığı alanlar ve bu alanlara etkileri. *Yükseköğretim ve Bilim Dergisi/Journal of Higher Education and Science, 11*(2), 425-436.

Bayılmış, C., Küçük, K. (2019). *Nesnelerin İnterneti: Teori ve Uygulamaları* (1). İMAK Ofset Basım Yayın, Türkiye: Papatya Bilim.

Baz, F. Ç., Uludağ, K. (2021). Veri Merkezi Güvenliğinin Sağlanmasında IoT Sensörlerinin Kullanımı Üzerine Bir Uygulama. *Avrupa Bilim ve Teknoloji Dergisi, 27*, 392-397.

Boyes, H., Cunningham, J., Hallaq, B., Watson T. (2018). The industrial internet of things (IIoT): An analysis framework. *Computers In Industry, 101*, 1-12.

Buyya, R., Gubbi, J., Marusic, S., Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29, 1645-1660.

Bütün İ., (2021). Security ımplications of underlying network technologies on ındustrial ınternet of things. *Politeknik Dergisi*, 1

Cisco. (2015). *Internet of Things*. https://www.cisco.com, https://www.cisco.com/c/en/us/products/collateral/se/internet-of-things, (Ziyaret tarihi: 09 Ekim 2021).

Corotinschi, G., Gaitan, V.G. (2018). Enabling IoT connectivity for Modbus networks by using IoT edge gateways. *14th International Conference on DEVELOPMENT AND APPLICATION SYSTEMS*, Romania, 24-26 June 2018.

Crespo, E. *Simple Modbus Master*. https://github.com, https://github.com /jecrespo/simple-Modbus/blob/master/Modbus RTU libraries for Arduino/SimpleModbusMasterManual.pdf, (Ziyaret tarihi: 28 Kasım 2019).

Crispin, B., Shaout, A. (2018). Using the MQTT Protocol in Real Time for Synchronizing IoT Device State. *The International Arab Journal of Information Technology, 15*, 515-521.

Demiroğlu, U., Şenol, B., Yıldırım, M. (2021). An in-depth exam of IoT, IoT Core Components, IoT Layers, and Attack Types. *European Journal of Science and Technology, 28*, 665-669.

Deveci, F. (2016). *Tarihi Haberleşme Metodu: Modbus RTU*. http://www.firatdeveci.com/, http://www.firatdeveci.com/tarihi-haberlesme-metodu-Modbus-rtu/, (Ziyaret tarihi: 25 Ekim 2019).

Ding, Y., Lu, H., Shu, F. (2019). Novel Modbus Adaptation Method for IoT Gateway, *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, China, 06 June 2019.

Dolcel, H., Durgun, M., Gökrem, L. (2021). IoT Tabanlı ve Makine Öğrenmesine Dayalı Seçici Sulama Sistemi. *European Journal of Science and Technology, 28*, 395-401.

Durgun, M., Kışlakçı, M. (2021). IoT Sistemlerde Sınır Bilişim Destekli Anomali Tespiti. *Avrupa Bilim ve Teknoloji Dergisi, 28*, 481-488.

Erdoğan, H., Khan, S. A., Küçük, K. (2020). Endüstriyel IoT Bulut Uygulamaları için Düşük Maliyetli Modbus/MQTT Ağ Geçidi Tasarımı ve Gerçekleştirilmesi. *Bilecik Şeyh Edebali Üniversitesi Fen Bilimleri Dergisi, 7* (1), 170-183.

Gilchrist, A.(2016). *Industry 4.0: The Industrial Internet of Things. Apress*, Thailand,259.

Guo, K., Hu, D., Ma, J., Sun, C., Xu, Z. (2019). Design and Development of Modbus/MQTT Gateway for Industrial IoT Cloud Applications Using Raspberry Pi. *IEEE Internet of Things Journal*, 2, 2267-2271.

Hong, D.K., Ju, H., Kim, H., Lee, S. (2013). Correlation Analysis of MQTT Loss and Delay According to QoS Level, *The International Conference on Information Networking 2013 (ICOIN)*, Bangkok, Thailand, 28-30 January 2013.

IBM. (2015). *IBM Watson IoT Platform*. https://www.ibm.com/watson/, https://www.ibm.com/cloud/watson-IoT-platform, (Ziyaret tarihi: 06 Aralık 2019).

IEEE Standart (2006*). IEEE Recommended Practice for the Design of Reliable Industrial and Commercial Power Systems* (P493/D4). IEEE.

Jadhav Y.R., Salpe, V.D, Shinde, S.A., Singh, S., Nimkar, P.A. (2016). MQTT-Message Queuing Telemetry Transport protocol. *International Journal of Research, 3*, 240-244.

Kılçık F.M, Topçu İ.B., Uygunoğlu T. (2021). Nesnelerin İnternetinin (IoT) İnşaat Mühendisliğindeki Rolü: Gömülü Sensör Kullanımı. *Int. J. of 3D Printing Tech. Dig. Ind., 5*(3): 390-399.

Köse, G., Kurutkan, M.N. (2021). Sağlık Hizmetlerinde Nesnelerin İnterneti Uygulamalarının Bibliyometrik Analizi. *Avrupa Bilim ve Teknoloji Dergisi, 27*, 412-432.

Kuzlu, M., Nugur, A., Pipattanasomporn, M., Rahman, S. (2018). Design and Development of an IoT Gateway for Smart Building Applications. *IEEE Internet of Things Journal, 6*, 1-10.

Küçük K., Ustaoglu Y. (2020) Deniz Araçları için Nesnelerin İnterneti Teknolojileri Temelli Şarj Kontrol Sistemi tasarımı, *6. Uluslararası Mühendislik ve Tasarım Kongresi (MMT),* İstanbul, Türkiye, 17-18 Aralık 2020.

Lundgaard, L.E. (1992). Partial discharge. XIV. Acoustic partial discharge detection-practical application. *IEEE Electrical Insulation Magazine, 8*, 34-43.

Madakam, S., Ramaswamy, R., Tripathi, S. (2015). Internet of Things (IoT): A Literature Review. *Journal of Computer and Communications, 3*, 164-173.

Mihu, A.C., Palacean, A.V., Rosner, D., Tranca, D.C. (2017). ZigBee based wireless Modbus Aggregator for Intelligent Industrial Facilities, *2017 25th Telecommunication Forum (TELFOR),* Serbia, 21-22 November 2017.

Nguyen-Hoang, P., Vo-Tan, P. (2019). Development An Open-Source Industrial IoT Gateway. 2019 *19th International Symposium on Communications and Information Technologies*, Vietnam, 25-27 September 2019.

EKLER

Ek-A

Esp8266 ile IBM Watso IoT platformuna veri aktarımı için kullanılan kod verilmiştir. (MegaWi-FiIBMWatson.ino).

```
#include <ESP8266Wi-Fi.h>
#include <Wi-FiClient.h>
#include <PubSubClient.h>

#define dataSize 6
#define cycle 120000

char* ssid = "ssid";
char* password = " password ";
#define ORG "org"
#define DEVICE_TYPE "device_type"
#define DEVICE_ID "device_type"
#define TOKEN "token"

String topics[] = {"IoT-2/evt/sensor1/fmt/json","IoT-2/evt/sensor2/fmt/json","IoT-2/evt/sensor3/fmt/json",
                   "IoT-2/evt/sensor4/fmt/json","IoT-2/evt/sensor5/fmt/json","IoT-2/evt/sensor6/fmt/json"
                  };
String sensors[] = {"""",\"sensor1\":""","",\"sensor2\":""","",\"sensor3\":""","",\"sensor4\":""",
                    """",\"sensor5\":""","",\"sensor6\":"""
                   };
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
float datas[dataSize];
float previousDatas[dataSize];

String previousData = ".";
int i = 0;
int sayac = 0;
int sayac2 = 0;
long firstMsg = 0;

Wi-FiClient Wi-FiClient;
PubSubClient client(server, 1883, NULL, Wi-FiClient);

void ibmConnect(){
  if (!client.connected()) {
       while (!client.connect(clientId, authMethod, token)) {
          delay(500);
```

```
      }
    }
}

void setup() {
    Serial.begin(115200);
    Wi-Fi.begin(ssid, password);
    while (Wi-Fi.status() != WL_CONNECTED) {
      delay(500);
    }
    ibmConnect();
}

void loop() {
  String nowData;
    client.loop();
    if(i == 0){
      firstMsg = millis();
      i++;
    }
    long now = millis();
    if (!client.connected()){
      ibmConnect();
    }
    int check[dataSize];
    Serial.println(1);
    if((now - firstMsg) < cycle && recvdata(datas)){
      for(int j = 0; j < dataSize; j++){
        if(previousDatas[j] == datas[j]){
          check[j] = 1;
          sayac++;
        }
        else{
          check[j] = 0;
          sayac2++;
        }
        //nowData += String(s[n]) + "-";
      }

      for(int j = 0; j < dataSize; j++){
        if(check[j] == 0)
            sendDataToIbm(topics[j],sensors[j],datas[j]);
        previousDatas[j] = datas[j];
      }

    }else{
      Serial.print("Gönderilmeyen:");
      Serial.println(sayac);
```

```
      Serial.print("Gönderilen:");
      Serial.println(sayac2);
    }
    previousData = nowData;
    delay(1000);
}

void sendDataToIbm(String topicName, String event, float data){
  String payload1 = "{\"d\":{\"Name\":\"" DEVICE_ID "\"";
        payload1 += event;
        payload1 += data;
        payload1 += "}}";
  char topic[topicName.length() + 1];
  topicName.toCharArray(topic, topicName.length()+1);
  client.publish(topic, (char*) payload1.c_str());
}

bool recvdata(float *d){
  char c;
  String a[dataSize];
  int j = 0;
  while(Serial.available()>0){
    c=Serial.read();
    if(c !=';'){
      a[j] = a[j]+c;
    }if(c == ';'){
      j++;
    }
  }
  for(int i = 0; i< dataSize; i++){
    d[i] = round(a[i].toInt()/100.0);
  }
  return true;
}
```

Ek-B

Modbus master ile Modbus slave'den alınan veriler MQTT protokolüne aktarımı için kullanılan kodlar verilmiştir. (MegaWi-FiIBMWatsonModbusMaster.ino).

```
#include <SimpleModbusMaster.h>

////////////////////// Port information //////////////////
#define baud 115200
#define timeout 1000
#define polling 1000 // the scan rate
#define retry_count 100

// used to toggle the receive/tansmit pin on the driver
#define TxEnablePin 7

//SoftwareSerial RS485(15,14); //RX TX

// The total amount of available memory on the master to store data
#define TOTAL_NO_OF_REGISTERS 6
// Add as many as you want. TOTAL_NO_OF_PACKETS
// This is the easiest way to create new packets
// is automatically updated.
enum
{
  PACKET1,
  PACKET2,
  TOTAL_NO_OF_PACKETS // leave this last entry
};

// Create an array of Packets to be configured
Packet packets[TOTAL_NO_OF_PACKETS];

unsigned long sendd = 0;
// Masters register array
unsigned int regs[TOTAL_NO_OF_REGISTERS];

void setup()
{
  Serial.begin(baud);
  Serial2.begin(baud);
  Serial3.begin(baud);

  //fonksiyon sonrası ilki hangi adresten okunacak
  //ikincisi okunacak kayıt sayısı
  //sonuncusu verinin yazılmaya başlayacağı regs başlangıç indexi
```

```
  Modbus_construct(&packets[PACKET1], 1, READ_HOLDING_REGISTERS, 0, TOTAL_NO_OF_REGISTERS, 0);
  //Modbus_construct(&packets[PACKET2], 2, READ_HOLDING_REGISTERS, 0, TOTAL_NO_OF_REGISTERS/2, TOTAL_NO_OF_REGISTERS/2);

  //SERIAL_8E1
  Modbus_configure(&Serial1, baud, SERIAL_8N2, timeout, polling, retry_count, TxEnablePin, packets, TOTAL_NO_OF_PACKETS, regs);

}
int k = 0;
void loop()
{
  Modbus_update();
  while(Serial3.available()>0){
    char c=Serial3.read();
    Serial.print(c);
    if(c == '1'){
       String s;
       Serial2.write(String(k).c_str());
       k++;

       for(int i=0; i< TOTAL_NO_OF_REGISTERS; i++){
        s += String(regs[i]) + ";";
       }
       Serial.println();
       Serial.println(s);
       Serial3.write(s.c_str());

       if(k == 120){
          k = 0;
       }
    }
  }

  //delay(1000);
}
```

Ek-C

Modbus master'a belirli periyotlarda veri sağlayan Modbus slave kodları verilmiştir. (ModbusSlave.ino).

```
#include <SimpleModbusSlave.h>

#define baud 115200
#define TxEnablePin 7
#define SlaveID 1
#define regsSize 6


unsigned int holdingRegs[regsSize];
int veriler[120][28] = [Dataset]

void setup()
{
 Serial.begin(baud);
 Serial2.begin(baud);
 Modbus_configure(&Serial1, baud, SERIAL_8N2,SlaveID, TxEnablePin , regsSize, holdingRegs);

 Modbus_update_comms(baud, SERIAL_8N2, SlaveID);
}
int data;
String sayac;
bool ok = true;
void loop()
{
 while(Serial2.available()>0){
   if(ok){
     sayac = "";
     ok = false;
   }
   char c=Serial2.read();
   sayac += c;
 }
 ok = true;
 data = sayac.toInt();
 //Serial.println(data);
 for(int j = 0; j < regsSize; j++){
   holdingRegs[j] = veriler[data][j];
 }
 Modbus_update();
 delay(250);
}
```

Ek-D

Lokalde kurulan MQTT broker ile yapılan testlerde kullanılan verilerin, değişip değişmediği dikkate almadan ve tüm okunan verinin aktarımında yapılan istatistik sonuçları elde etmek için kullanılan kodlar verilmiştir. Test süresince MQTT tüm servis kalitesinde testler gerçekleştirilmiştir. Test süresince Modus master ve slave için Ek-2 ve Ek-3 de kullanılan kodlar kullanılmıştır. (mosquitto_QoS123_standart.ino).

```
#include <ESP8266Wi-Fi.h>
#include <MQTT.h>
#include <Average.h>

#define QoS 0
#define dataSize 6
#define cycle 120

Wi-FiClient net;
MQTTClient client(512);
char* ssid = " ssid";
char* pass = " pass";
float datas[dataSize];
float previousDatas[dataSize];
unsigned long sendd = 0;
unsigned long receivee = 0;
int i = 0;
int minTime = 0;
int maxTime = 0;
int data = 0;
int sayac = 0;
Average<float> allTime(cycle);

void connect() {
  while (Wi-Fi.status() != WL_CONNECTED) {
    delay(1000);
  }
  while (!client.connect("", "", "")) {
    delay(1000);
  }
  client.subscribe("test",QoS);
}

void messageReceived(String &topic, String &payload) {
  receivee = millis();
  if(i > 2){
    if((receivee - sendd)>=1000){
      data = (receivee - sendd)-1000;
      allTime.push(data*1.0);
    }else{
```

```
      data = (receivee - sendd);
      allTime.push(data*1.0);
    }
  }

  client.publish("feedback", "sendd: " + String(sendd) + " // " + " receivee: " + String(receivee) + " // " + " fark: " + String(receivee - sendd),false,0);
}

void setup() {
  Serial.begin(115200);
  Wi-Fi.begin(ssid, pass);
  client.begin("LocalIp",1883, net);
  client.onMessage(messageReceived);
  connect();
}

void loop() {

  client.loop();
  //delay(20);  // <- fixes some issues with Wi-Fi stability
  if (!client.connected()) {
    connect();
  }
  int check[dataSize];
  Serial.println(1);
  sendd = millis();
  if(i < cycle+2 && recvdata(datas)){
    String data;
    for(int j = 0; j< dataSize; j++){
       data += String(datas[j]) + "-";
    }
    client.publish("test", data.c_str(),false, QoS);
    i++;
  } else{
    float meann = allTime.mean();
    float sS = allTime.stddev();
    float var = sq(sS);
    int mi = allTime.minimum(&minTime);
    int ma = allTime.maximum(&maxTime);
    String data = "Mean:" + String(meann) + " min:" +String(mi) +" max:" + String(ma) + " StdDev:" + String(sS) + " Var:" + String(var) + " gonderilmesi gereken:" + String(120*dataSize) + " Gonderilen:" + String(sayac);
    client.publish("feedback", data.c_str(),false,0);
  }

  delay(1000);
}
```

```
bool recvdata(float *d){
  char c;
  String a[dataSize];
  int j = 0;
  while(Serial.available()>0){
    c=Serial.read();
    if(c !=';'){
      a[j] = a[j]+c;
    }if(c == ';'){
      j++;
    }
  }
  for(int i = 0; i< dataSize; i++){
    d[i] = a[i].toInt()/100.0;
  }
  return true;
}
```

Ek-E

Lokalde kurulan MQTT broker ile yapılan testlerde kullanılan verilerin, değişip değişmediği dikkate alan ve tüm okunan verinin aktarımında yapılan istatistik sonuçları elde etmek için kullanılan kodlar verilmiştir. Test süresince MQTT tüm servis kalitesinde testler gerçekleştirilmiştir. Test süresince Modus master ve slave için Ek-2 ve Ek-3 de kullanılan kodlar kullanılmıştır. (mosquitto_QoS123.ino).

```
#include <ESP8266Wi-Fi.h>
#include <MQTT.h>
#include <Average.h>

#define QoS 0
#define dataSize 6
#define cycle 120

Wi-FiClient net;
MQTTClient client(512);
char* ssid = " ssid";
char* pass = " pass";
float datas[dataSize];
float previousDatas[dataSize];
unsigned long sendd = 0;
unsigned long receivee = 0;
int i = 0;
int minTime = 0;
int maxTime = 0;
int data = 0;
int sayac = 0;
Average<float> allTime(cycle);

void connect() {
  while (Wi-Fi.status() != WL_CONNECTED) {
    delay(1000);
  }
  while (!client.connect("", "", "")) {
    delay(1000);
  }
  client.subscribe("test",QoS);
}

void messageReceived(String &topic, String &payload) {
  receivee = millis();
  if(i > 2){
    if((receivee - sendd)>=1000){
      data = (receivee - sendd)-1000;
      allTime.push(data*1.0);
    }else{
```

```
    data = (receivee - sendd);
    allTime.push(data*1.0);
   }
  }

  client.publish("feedback", "sendd: " + String(sendd) + " // " + " receivee: " +
String(receivee) + " // " + " fark: " + String(receivee - sendd),false,0);
}

void setup() {
  Serial.begin(115200);
  Wi-Fi.begin(ssid, pass);
  client.begin("LocalIp",1883, net);
  client.onMessage(messageReceived);
  connect();
}

void loop() {

  client.loop();
  //delay(20);  // <- fixes some issues with Wi-Fi stability
  if (!client.connected()) {
   connect();
  }
  int check[dataSize];
  Serial.println(1);
  sendd = millis();
  if(i < cycle+2 && recvdata(datas)){
   for(int j = 0; j < dataSize; j++){
    if(previousDatas[j] == datas[j]){
     check[j] = 1;
    }
    else{
     check[j] = 0;
     sayac++;
    }
   }

   String data;
   for(int j = 0; j< dataSize; j++){
    if(check[j] == 0){
     data += String(datas[j]) + "-";
    }
     previousDatas[j] = datas[j];
   }
   client.publish("test", data.c_str(),false, QoS);
   i++;
  } else{
```

```
    float meann = allTime.mean();
    float sS = allTime.stddev();
    float var = sq(sS);
    int mi = allTime.minimum(&minTime);
    int ma = allTime.maximum(&maxTime);
    String data = "Mean:" + String(meann) + " min:" +String(mi) +" max:" + String(ma) +
" StdDev:" + String(sS) + " Var:" + String(var) + " gonderilmesi gereken:" +
String(120*dataSize) + " Gonderilen:" + String(sayac);
    client.publish("feedback", data.c_str(),false,0);
  }

  delay(1000);
}

bool recvdata(float *d){
  char c;
  String a[dataSize];
  int j = 0;
  while(Serial.available()>0){
    c=Serial.read();
    if(c !=';'){
      a[j] = a[j]+c;
    }if(c == ';'){
      j++;
    }
  }
  for(int i = 0; i< dataSize; i++){
    d[i] = a[i].toInt()/100.0;
  }
  return true;
}
```

KİŞİSEL YAYINLAR VE ESERLER

Erdoğan, H., Khan, S. A., Küçük, K. (2020). Endüstriyel IoT Bulut Uygulamaları için Düşük Maliyetli Modbus/MQTT Ağ Geçidi Tasarımı ve Gerçekleştirilmesi. *Bilecik Şeyh Edebali Üniversitesi Fen Bilimleri Dergisi, 7* (1), 170-183.

ÖZGEÇMİŞ

İlk, orta ve lise eğitimini Kocaeli'nde tamamladı. 2014 yılında kazandığı Karabük Üniversitesi Bilgisayar Mühendisliği bölümünden 2019 yılında mezun oldu. Mezun olduktan kısa bir süre sonra Kocaeli Üniversitesi Fen bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda yüksek lisans eğimine başladı ve yazılım geliştirici olarak bir süre Teknopark firmasında çalıştı. 2022 yılında Kocaeli Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda yüksek lisans eğitimine devam etmektedir. Yüksek lisans eğitimi boyunca "Endüstriyel IoT Bulut Uygulamaları için Düşük Maliyetli Modbus/MQTT Ağ Geçidi Tasarımı ve Gerçekleştirilmesi" adında bir çalışma yapmıştır. 2021 yılından beri özel sektörde yazılım uzmanı olarak görev yapmaktadır.