BŞEÜ Fen Bilimleri Dergisi
7(1), 170-183, 2020

BSEU Journal of Science
DOI: 10.35193/bseufbd.708445

e-ISSN: 2458-7575 (http://dergipark.gov.tr/bseufbd)

Araştırma Makalesi - Research Article

Endüstriyel IoT Bulut Uygulamaları için Düşük Maliyetli Modbus/MQTT Ağ Geçidi Tasarımı ve Gerçekleştirilmesi

Hamdi Erdoğan[1], Kerem Küçük[2*], Sajjad Ahmad Khan[3]

Geliş / Received: 24/03/2020 ***Revize / Revised: 14/05/2020*** ***Kabul / Accepted: 18/05/2020***

ÖZ

Günümüzde, bulut teknolojileri endüstriyel alanlarda kullanılan cihaz ve makinelerden toplanan verilerin gerçek zamanlı olarak takip edilmesinde önemli çözümler sunmaktadırlar. Sahada elde edilen verilerin bulut platformlarına aktarılabilmesi için farklı türden ağ geçitlerinden yararlanılmaktadır. Bu çalışmada endüstriyel alanlarda kullanılabilecek düşük güç tüketimine sahip, düşük maliyetli nesnelerin interneti (Internet of Things, IoT) temelli bir ağ geçidi prototipi tasarımı sunulmaktadır. Bulut platforma aktarılacak verilerin sahadan ağ geçidi üzerine alınması için endüstriyel veri iletişiminde sıklıkla kullanılan Modbus protokolü kullanılmıştır. Ağ geçidinde toplanan verilerin bulut platformuna kablosuz ortamda aktarımını sağlamak için IoT uygulamalarında sıkça kullanılan Mesaj Kuyruk Telemetri Ulaştırma Protokolü (Message Queuing Telemetry Transport, MQTT) protokolü kullanılmıştır. Tasarlanan ağ geçidi, veriler üzerinde farklılığı algılayabilecek bir algoritma ile programlanmıştır. Verilerin hızlı bir şekilde görselleştirilmesi ve depolanması için IBM Watson IoT Platformu tercih edilmiştir. Ayrıca çalışmada tasarlanan ağ geçidinin performans analizi için MQTT sunucusu olarak Mosquitto kullanılmıştır. Deneysel test çalışmalarında, farklı boyutlarda yüklere sahip veri mesajlarının farklı seviyelerde servis kalitesinde (Quality of Service, QoS) uçtan uca gecikme süreleri ve istatistiksel sonuçları verilmiştir.

Anahtar Kelimeler- Modbus, MQTT, Ağ geçidi, Mosquitto, Nesnelerin interneti (IoT)

[1]İletişim: 195112003@kocaeli.edu.tr (https://orcid.org/0000-0002-8893-2165)
Bilgisayar Mühendisliği, Kocaeli Üniversitesi, Umuttepe Kampüsü, 41001 İzmit Kocaeli
[2]*Sorumlu yazar iletişim: kkucuk@kocaeli.edu.tr (https://orcid.org/0000-0002-2621-634X)
Bilgisayar Mühendisliği, Kocaeli Üniversitesi, Umuttepe Kampüsü, 41001 İzmit Kocaeli
[3]iletişim: sajjad.khan@kocaeli.edu.tr (https://orcid.org/0000-0002-0787-1357)
Bilgisayar Mühendisliği, Kocaeli Üniversitesi, Umuttepe Kampüsü, 41001 İzmit Kocaeli

BŞEÜ Fen Bilimleri Dergisi
7(1), 170-183, 2020

BSEU Journal of Science
DOI: 10.35193/bseufbd.708445

e-ISSN: 2458-7575 (http://dergipark.gov.tr/bseufbd)

Design and Implementation of Low-Cost of Modbus/MQTT Gateway for Industrial IoT Cloud Applications

ABSTRACT

Nowadays, cloud technologies offer remarkable solutions in real-time tracking of data collected from devices and machines used in industrial areas. In such cases, different types of gateways are used to transfer the obtained data to the cloud platforms. This study presents a new design of gateway prototype based on the Internet of Things (IoT), which can be used in industrial areas with low-cost and low power consumption features. Modbus protocol, which is commonly used for data communication in industrial areas, is used to gather the data to be transferred to the cloud platform from the industrial area to the gateway. The Message Queuing Telemetry Transport (MQTT) protocol, which is popularly used in IoT applications, has been employed to transfer the data collected in the proposed gateway to the cloud platform in a wireless environment. The designed gateway is programmed with an algorithm that is capable of differentiating the gathered data. IBM Watson IoT Platform is adopted for rapid monitoring and storing the data. Also, Mosquitto was used as MQTT server for performance analysis of the gateway designed in the study. Furthermore, for experimental test studies, end-to-end latency and statistical results of different levels of data messages with different levels of service quality such as quality of service (QoS) are shown.

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

BŞEÜ Fen Bilimleri Dergisi
7(1), 170-183, 2020

BSEU Journal of Science
DOI: 10.35193/bseufbd.708445

e-ISSN: 2458-7575 (http://dergipark.gov.tr/bseufbd)

I. GİRİŞ

İletişim teknolojilerinin her geçen gün daha fazla gelişim göstermesi ile birlikte günlük ihtiyaçlarımızı ve işlerimizi kolaylaştıran iletişim yeteneğine sahip nesnelerin sayısında artışa yol açmaktadır. Bu nesnelerin birçoğu farklı teknoloji ve iletişim protokolleri ile internete bağlanabilmektedir [1]. İnternete bağlanabilen nesnelerin sayısındaki artış ile birlikte nesnelerin kendi aralarında haberleşebilmeleri üzerine nesnelerin interneti (Internet of Things, IoT) kavramı ortaya çıkmıştır [2]. Nesnelerin interneti uygulamalarının kullanım alanlarından biri de endüstridir. Endüstride bu alan endüstriyel nesnelerin interneti (Industrial Internet of Things, IIoT) olarak adlandırılmaktadır [3].

Makinelerin haberleşmesi, sensörlerden gelen verilerin işlenebilmesi ve depoların takip edilmesi gibi çalışmalarda IIoT uygulamaları hem maliyet hem de zaman açısından avantajlar sağlamaktadır. Endüstriyel nesnelerin interneti uygulamalarında önemli bir problem, verilerin iletişiminde farklı protokollerin birlikte çalışabilirliğidir. Modbus protokolü endüstriyel cihazların kendi aralarında veri iletişimini sağlayabilmeleri için geliştirilmiştir [4]. Bu endüstriyel iletişim protokolünün açık kaynak kodlu, kolay anlaşılabilir ve cihazlara kolay uyarlanabilir yapısı nedeniyle üreticiler tarafından sıkça tercih edilen bir iletişim protokolü olmasını sağlamaktadır. Modbus protokolü ile üreticiden bağımsız olarak aynı ağa bağlı olan Modbus destekli tüm cihazlarla iletişim kurulabilmektedir. Verilerin sahadan toplanarak izlenebilmesi ve depolanabilmesi ihtiyacı üzerine bulut teknolojileri geliştirilmiştir. İnternet tabanlı veri iletişimi için geliştirilen Mesaj Kuyruk Telemetri Ulaştırma Protokolü (Message Queuing Telemetry Transport, MQTT) protokolü bulut teknolojilerinde sıkça kullanılmaktadır [5]. Günümüzde IoT platformlarının büyük bir kısmı protokolün minimum kaynak tüketimini hedeflemesi, TCP/IP'nin kullanıldığı işletim sistemlerini (Windows, Linux, MacOS, Android ve IOS) desteklemesi ve milisaniye (ms) düzeyinde haberleşmeye imkan sağlaması gibi güçlü özelliklerinden dolayı MQTT protokolünü tercih etmektedir. Bu doğrultuda verilerin sahadan toplanarak bulut platformlarında izlenmesinde farklı protokollerin birlikte çalışabilmesi önemli hale gelmiştir.

Endüstriyel nesnelerin interneti uygulamalarının belirtilen ihtiyaçları karşılayabilmesinde ağ geçitleri önemli rol oynamaktadır [6]. Cihazlardan toplanan veriler ağ geçidi üzerinden bulut platformuna aktarılmaktadır. Kullanılan ağ geçidi kapsamlı bir protokol desteği sunmalıdır. Bu doğrultuda endüstride ve IoT'nde yaygın olarak kullanılan Modbus ve MQTT protokolleri bu gereksinimleri karşılamaktadır. Tasarlanan prototipin endüstriyel alanda yetkinliğinin belirlenebilmesi için yine endüstride elektriksel kısmi deşarj oluşabilecek 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 [7]. Yapılan çalışmalara göre orta ve yüksek gerilimli sistemlerde oluşan sorunların büyük bir kısmı (%80) belirtilen problemden kaynaklanmaktadır [8]. Dielektrik malzemenin yapısını bozabilecek güçte gerilimin olduğu ortamlarda elektriksel kısmi deşarj meydana gelebilir. Belirtilen problemden kaynaklı sorunların, IIoT temelli bir yapı ile çözülmesi güncel bir problem olarak ortaya çıkmaktadır.

Bu çalışmada, IIoT uygulamalarında ortaya çıkan maliyet, performans ve farklı protokoller arası iletişim problemleri göz önünde bulundurularak bu problemlerin üstesinden gelebilecek ve güncel IoT teknolojilerini destekleyebilen bir ağ geçidi prototipi geliştirilmesi amaçlanmaktadır. Tasarımın minimum maliyetli olması için Arduino geliştirme kartı kullanılmıştır. Önerilen ağ geçidi prototipinin performansının iyileştirilmesi ve ağ trafiğinin hafifletilmesi için veriler üzerinde farklılığı algılayabilecek bir yazılım geliştirilmiştir. Sistemin IoT platformları ve endüstride kullanılan cihazlarla iletişimi yüksek oranda desteklemesi için MQTT ve Modbus protokolleri tercih edilmiştir. Prototip, farklı protokoller arası iletişim problemini ortadan kaldırmak için MQTT ve Modbus protokolleri arasındaki veri iletişiminde gerekli dönüşümleri desteklemektedir. Verilerin bulut platformunda görselleştirilmesini sağlamak için IBM Watson IoT platformu kullanılmıştır [9]. Bu platform ile verilerin gerçek zamanlı izlenebilmesi 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 köle cihazdan okunarak bulut platformuna saniyede bir kez gönderilecek şekilde belirlenmiştir. Modbus protokolünde ise köle cihazlarının sayısı ve okunacak kayıt sayısının performans üzerinde etkili olduğu göz önünde bulundurularak bu parametrelerde farklı varyasyonlar kullanılarak test sonuçları elde edilmiştir.

Çalışmanın organizasyonu şu şekilde oluşturulmuştur. Bölüm 2'de ağ geçitleri için literatürde yapılmış çalışmalar, Bölüm 3'te tasarlanan ağ geçidi prototipi için önerilen sistem mimarisi ve mimaride kullanılan araçlar hakkında bilgiler verilmiştir. Bölüm 4'te tasarımda verilerin okunması, iletilmesi ve görselleştirme aşamaları hakkında detaylıca söz edilmiştir. Bölüm 5'te oluşturulan tasarımın farklı koşullar altında performans ve test sonuçları sunulmuştur. Sonuçlar bölümü ile çalışma sonlandırılmıştır.

II. LİTERATÜR ÇALIŞMASI

Literatüde, ilk IoT ağ geçidi çalışması 2013 yılında Shinho ve arkadaşları tarafından yapılmıştır [10]. Shinho ve arkadaşları Nesnelerin İnterneti uygulamalarında sıkça tercih edilen MQTT protokolünün performans analizi üzerine çalışmışlardır. Çalışmada protokol kablolu ve kablosuz olarak iki ayrı kısımda test edilmiştir. Aditya ve arkadaşları [11], binalarda enerji tüketiminin gün geçtikçe artması üzerine tüketilen enerjinin bulut teknolojisi ile izlenmesi ve kontrol edilebilmesi için IoT ağ geçidi tasarımı gerçekleşmişlerdir. Phuc ve Phuoc [12], endüstriyel alanda çalışabilecek bir IoT ağ geçidi prototipi tasarlamışlardır. Bu prototip Linux tabanlı olup, farklı endüstriyel saha veri yollarından ve IoT protokollerinden hızlı ve kolay veri alışverişini sağlayabilecek şekilde tasarlanmıştır. Ghenadie ve Vasile [13], yerel veri işleme gerçekleştirerek Modbus ağlarının IoT uygulamalarına genişletmeyi amaçlamışlardır. Sistemin performansını farklı sunucular üzerinde test ederek çalışmalarında sunmuşlardır. Feng ve arkadaşları [14], Modbus için yeni bir adaptasyon yöntemi önermişlerdir. Önerdikleri yöntem, Modbus köle verileri ile uygulama verilerini algılama/çalıştırma verileri arasında çift yönlü dönüşüm sağlamaktadır. Dimitru ve çalışma arkadaşları [15], endüstriyel tesislerde Modbus kölelerinden veri toplayan ZigBee tabanlı kablosuz bir veri iletişim sistemi sunmuşlardır. Sistemin kablosuz olması, yapılacak olan Modbus sorgularının tüm cihazlara paralel bir şekilde iletilmesi ile sorgulama hızını arttırmaktadır ve sistemin kurulumunu kolaylaştırmaktadır.

Adnan ve Brennan önerdikleri sistemde [16], farklı gömülü sistemlerle iletişim kurabilen MQTT tabanlı bir tasarım ve uygulama sunmuşlardır. Changqing ve arkadaşları [17], Modbus ve MQTT protokollerini kullanarak IIoT uygulamalarında kullanılabilecek ağ geçidi için Raspberry Pi ile tasarım sunmuşlardır. Tasarım saha cihazları ve bulut uygulamaları arasında iki yönlü iletişimi desteklemektedir. Ağ geçidinin yapılandırılması web sayfaları aracılığıyla geçekleştirilmektedir.

Bu çalışmada literatürde yapılan diğer çalışmalardan farklı olarak Arduino tabanlı olması ile düşük maliyetlidir. Çalışmada yazılım gereksinimleri dikkate alınarak maliyet etkinliği için iki farklı Arduino geliştirme kartı kullanılmıştır. Kablosuz Bağlantı Alanı (Wireless Fidelity, WiFi) tabanlı Arduino Mega kartı ağ geçidi olarak görev yapmaktadır. PD Annunciator cihazını köle olarak Arduino Mega kartı temsil etmektedir. MQTT üzerinden bulut platformuna aktarılan veriler, farklı hassasiyetlerde gönderilerek veriler üzerindeki değişimi algılayabilen trafiğine fazla yük getirmeyecek bir yazılıma sahiptir. Veriler bulut platformuna tek yönlü olarak iletilmektedir. PD Annunciator cihazına ait veri setindeki değerlerin okunarak görselleştirilmesi için IBM Watson IoT Platformu kullanılmıştır. İki farklı protokol arasında dönüşüm işlemini sağlamaktadır. Tasarımın genel olarak düşük maliyetli ve yüksek performanslı olması amaçlanmaktadır.

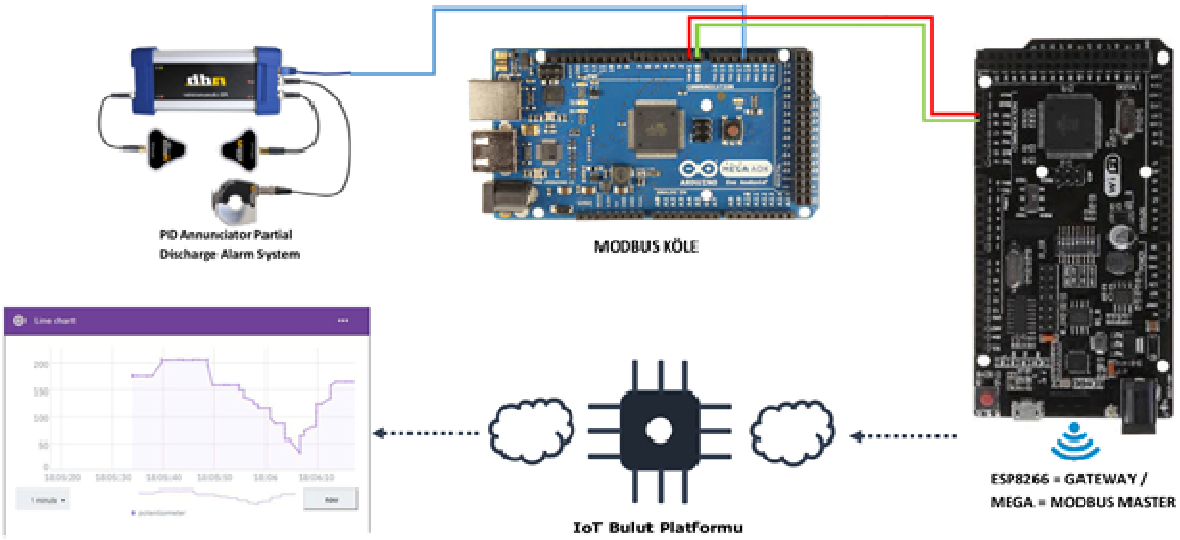
III. ÖNERİLEN SİSTEM BİLEŞENLERİ

Endüstriyel alanda kullanılabilecek ağ geçidini içeren mimarinin genel tasarımı Şekil 1'de verilmiştir. Mimari bir adet WiFi tabanlı Arduino Mega (Esp8266 + Atmega2560) kartı, bir adet Arduino Mega, PD Annunciator veri seti ve IBM Watson IoT platformu bileşenlerinden oluşmaktadır. Sistemin çalışma prensibi şu şekildedir: WiFi tabanlı Arduino Mega kartındaki iki ayrı çipte iki ayrı protokol çalışmaktadır. Esp8266'da çalışan MQTT, Arduino Mega üzerinde çalışan Modbus'tan verileri okuyarak bulut platformuna aktarmaktadır. Bulut platformuna aktarılan verilere bu aşamanın ardından internet erişimi olan herhangi bir noktadan erişim sağlanabilmektedir.

BŞEÜ Fen Bilimleri Dergisi
7(1), 170-183, 2020

BSEU Journal of Science
DOI: 10.35193/bseufbd.708445

e-ISSN: 2458-7575 (http://dergipark.gov.tr/bseufbd)



Şekil 1. Endüstriyel IoT ağ geçidi içeren sistem mimarisi

A. WiFi Tabanlı Arduino Mega

WiFi tabanlı Arduino Mega kartı standart Arduino Mega kartlarından farklı olarak üzerinde ESP8266 WiFi modülü barındırmaktadır [18]. Bu durum ek bağlantılar gerektirmeden tek kart kullanılarak internet bağlantısı sağlamaktadır. Kart hem Arduino Mega özelliklerini hem de Esp8266 modülünün özelliklerini bir arada kullanılmasına yardımcı olmaktadır. Her iki özelliğin kullanılmak istenmesi durumunda kod yazılarak eş zamanlı bir şekilde iki kod çalıştırılabilir ya da kodların bağımsız olarak çalıştırılması mümkündür. WiFi tabanlı Arduino Mega kartının üzerinde programların hangi mikro denetleyiciye atılacağını ve kodların nasıl çalıştırılacağını belirlemek için sekizli bir anahtar grubu bulunmaktadır.

B. Arduino Mega

Arduino Mega, Atmel firmasının ürettiği Atmega2560 mikro denetleyicisine sahip bir geliştirme kartıdır. Uno ve Nano gibi diğer Arduino kartlarına göre daha karmaşık projeler için tasarlanmıştır [19]. 54 adet dijital giriş/çıkış pinine ve 16 adet analog giriş pinine sahip olmasından dolayı 3D yazıcılar ve robotik projeler gibi daha geniş çalışmalar için önerilen karttır. Aynı zamanda da elektriksel problemlere karşı dayanıklı mimariye sahiptir.

C. PD Annunciator

PD Annunciator, geliştirilen sensörler ile metal kaplı dolaplar ve trafo merkezi aparatları gibi yüksek oranda elektrikli malzemelerde kısmi deşarj aktivitesini tespit etmek ve raporlamak için tasarlanmıştır [20]. Önemli ölçüde kısmi deşarj tespit edilirse, cihaz uyarı vermektedir. Her bir cihaza üç adet sensör bağlanabilmektedir ve 247 adet cihaz birbirine seri şekilde bağlanabilmektedir. PD Annunciator Modbus Uzaktan Bağlantı Ünitesi (Remote Terminal Unit, RTU) ve Amerikan Standart Kodlama Sistemi (American Standard Code for Information, ASCII) protokollerini kullanmaktadır. Bu protokolün kullanılmasında ki en büyük avantaj, Modbus ağını destekleyen tüm sistemlerle iletişim kurabilmesidir.

D. IBM Watson IoT Platformu

Sektörde sahada elde edilen verilerin takibi ve depolanması için son zamanlarda bulut teknolojilerinin kullanımı yaygınlaşmıştır [21]. IBM Watson IoT platformu bu hizmeti sunan platformlar arasında yer almaktadır. Nesnelerin interneti uygulamalarında ücretsiz üyelik sunarak hızlı bir başlangıç yapmamıza yardımcı

olmaktadır. IoT verilerinin kolay bir şekilde görselleştirilmesi ve depolanması, denetim, bağlantı ve aygıt kaybı gibi yetenekler sağlamaktadır [9].

E. MQTT

Temelde abone olma ve yayınlama esasına dayanan telemetri mesaj protokolüdür [5]. MQTT mimarisinde cihazlar, bir MQTT sunucu kullanarak yayım yapabilir veya istenilen mesajı dinlemek için abone olabilirler. Verilerin güvenliği için Güvenli Giriş Katmanı (Secure Sockets Layer, SSL)/Taşıma Katmanı Güvenliği (Transport Layer Security, TLS) yöntemlerini desteklemektedir. Haberleşmede milisaniye seviyelerinde hızlı veri iletimi ve minimum kaynak tüketimi sağlamaktadır. Sunucu üzerinden haberleşme temeline dayanır. Verilerin istemciye ulaşıp ulaşmadığını kontrol edebilen üç farklı servis kalitesinde (QoS) gönderim sağlamaktadır.

F. Modbus

Modbus, usta/köle (master/slave) tabanlı çalışan bir iletişim protokolüdür [4]. Master köle cihazlardan Modbus isteği yaparak veri okumaktadır. Köle cihazlar kendisine bağlı olan her bir bileşene ait değeri kendi hafızasında tutmaktadır. Master cihazdan Modbus isteği geldiği taktirde tuttuğu kayıtları (yazmaç) hafızadan okuyarak göndermektedir. Endüstriyel ortamda sıkça kullanılarak yaygınlaşmasında; akıllı endüstriyel uygulamalar için geliştirilmiş olması, açık bir protokol olması, telif hakkı gerektirmemesi, kendini kanıtlamış bir protokol olması, kullanımının kolay olması ve cihazlara hızlı uyarlanabilir olması gibi etkenler etkili olmuştur.

Modbus protokolü, haberleşme ve fiziksel katman olarak iki kısımdan oluşmaktadır. Bu çalışmada haberleşme katmanında kablolu iletişimde kullanılan daha etkin performans sağlayan RTU metodu kullanılmıştır. Fiziksel katmanda ise ek bir modül kullanılmadan kablolu seri iletişim kullanılmıştır. Modbus RTU haberleşme ağında 1 adet master ve 247 adet köle cihaz çalışabilmektedir. Verilerin okunmasında ve güncellenmesinde farklı fonksiyonlar kullanılmaktadır. Veri güvenliği için Çevrimsel Hata Denetimi (Cyclic Redundancy Check, CRC) doğrulama anahtarına sahiptir.

IV. IIOT SİSTEM TASARIMI

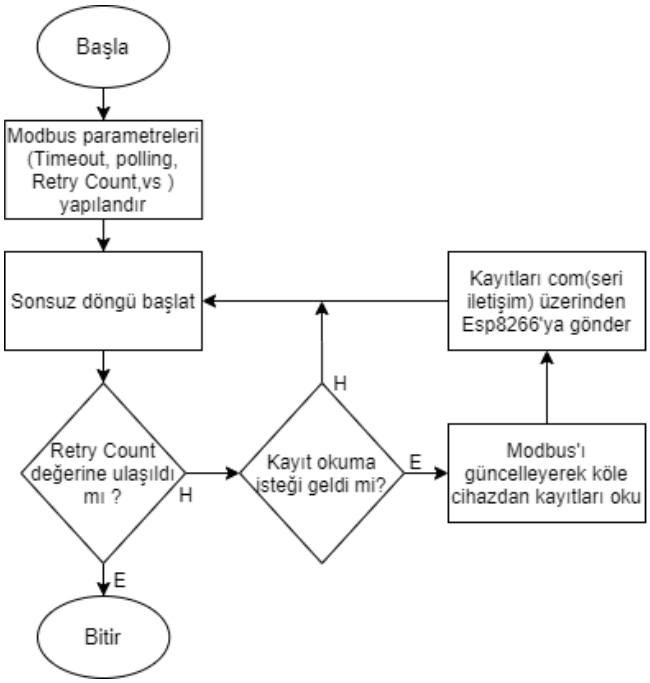
Bu çalışmada tasarlanan prototip sistemin amacı, endüstriyel alanda kullanılabilecek düşük maliyetli, düşük gecikmeye sahip, ağ trafiğine fazla yük getirmeyecek ve yaygın olarak kullanılan protokoller ile endüstri için çözüm sunabilecek bir ağ geçidi tasarımı sunmaktır. Önerilen sistemde, ağ geçidi olarak görev yapan WiFi tabanlı Arduino Mega hem Modbus master hem de MQTT yayıncı (publisher) olarak görev yapmaktadır. Modbus master; kart üzerinde bulunan Mega'da, MQTT yayıncı ise aynı kart üzerine yerleştirilen ESP8266'da çalışmaktadır. Modbus ağında bulunan köle cihazlara, kısmi deşarj aktivitesi tespiti için kullanılan akustik sensör (Airborne) ve toprak voltajları temas sensörlerinin (Transient earth voltages, TEV) bağlı olduğu PD Annunciator cihazına ait gerçek zamanlı ölçülmüş veriler kaydedilmiştir. Bu sayede köle olarak kullanılacak Arduino Mega kartı gerçek sensörleri temsil edebilmektedir. Toplanan veriler ağ geçidi aracılığıyla IBM Watson IoT Platformuna aktarılarak görselleştirme işlemi sağlanmaktadır.

A. Veri Toplama

Modbus ağında veriler, master cihaz tarafından saniyede bir kez yapılan istekler doğrultusunda köle cihazlardan okunmaktadır. Modbus isteği yapılmadan önce master üzerinde belirli parametrelerin ayarlanması gerekmektedir. Bu parametreler aşağıda verilmiştir:

- Timeout: Bir okuma isteği yapıldığında veri cevabın beklenme süresidir.
- Polling: Modbus okuma istediğinin sıklığını belirler.
- 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: Köle cihazlardan okunacak kayıt sayısını belirler.

Timeout = 1000 ms, polling = 1000 ms, retry count = 100 adet ve total no of register = 6, 12, 18, 24, 36, 48 adet parametreleri belirlendikten sonra master her istek yaptığında, gelen cevaptaki verileri kendi hafızasında bulunan "regs" adı verilen kayıt dizisinde tutmaktadır. Ağ geçidi olarak kullanılan Arduino kartında bulunan Mega, COM port üzerinden ESP8266 modülüyle haberleşebilmektedir. Master, COM portu aracılığıyla MQTT'den aldığı kontrol sinyali üzerine kayıt dizisinde tuttuğu verileri string formatına dönüştürerek tekrar aynı port üzerinden MQTT'ye göndermektedir. COM portu kullanılarak verilerin seri iletişim kanalından gönderilmesi iki farklı protokol arasındaki birlikte çalışma problemini ortadan kaldırmaktadır. Veri toplama algoritmasına ait akış diyagramı Şekil 2'de verilmiştir.



Şekil 2. Modbus ile veri toplama

B. Veri Yayınlama

Tasarlanan prototipin en önemli aşaması verilerin bulut platformuna aktarılmasıdır. Bu aşamada verilerin gönderim sıklığı ve veri boyutu sistemin ve ağın performansını önemli ölçüde etkilemektedir. MQTT ile veri gönderiminde, yayınlama sıklığı Modbus protokolüyle paralel bir şekilde çalışması amaçlanarak veriler saniyede bir kez gönderilecek şekilde ayarlanmıştır. Böylelikle hem ağın çok fazla kullanımının önüne geçilmiş hem de Modbus'ta yenilenen verilerde oluşabilecek senkronizasyon problemi sonrasında oluşabilecek kayıpların önlenmesi gerçekleştirilmiştir. Veri boyutunun ağın üzerindeki yüke etkisinin azaltması için prototip ağ geçidi üzerinde gelen verilerin kontrol edildiği bir yazılıma sahiptir. Modbus ağından okunan her sensöre ait veri bir sonraki iterasyonda kullanılmak üzere hafızada tutulmaktadır. Tutulan sensör verileri yeni gelen veriler ile karşılaştırılmaktadır. Eğer sensörlerden okunan verilerden değişim göstermeyen veri olduğu durumda, ağın aşırı yüklenmesini ve aynı verilerin tekrar gönderilmesini engellemek için değişim göstermeyen sensör verisi MQTT mesajına eklenmemektedir. Bu işlem farklı veri hassasiyetlerinde gerçekleştirilebilmektedir. Böylece ağ üzerinde gereksiz mesajların gönderilmesinin önüne geçilmektedir.

Kullanılan IBM Watson IoT Platformu diğer bulut teknolojilerinde olduğu gibi belirli bir MQTT formatını kullanmaktadır. Platform, gönderilen tüm verilerin JSON mesaj formatında gönderilmesini desteklemektedir. Gelen veriler işlenerek gönderilmesine karar verilen değerler belirlenen JSON formatına dönüştürülerek bulut platformuna aktarılmaktadır. Verilerin gönderilmesinde kullanılan algoritmanın sözde kodu Şekil 3'te verilmiştir.

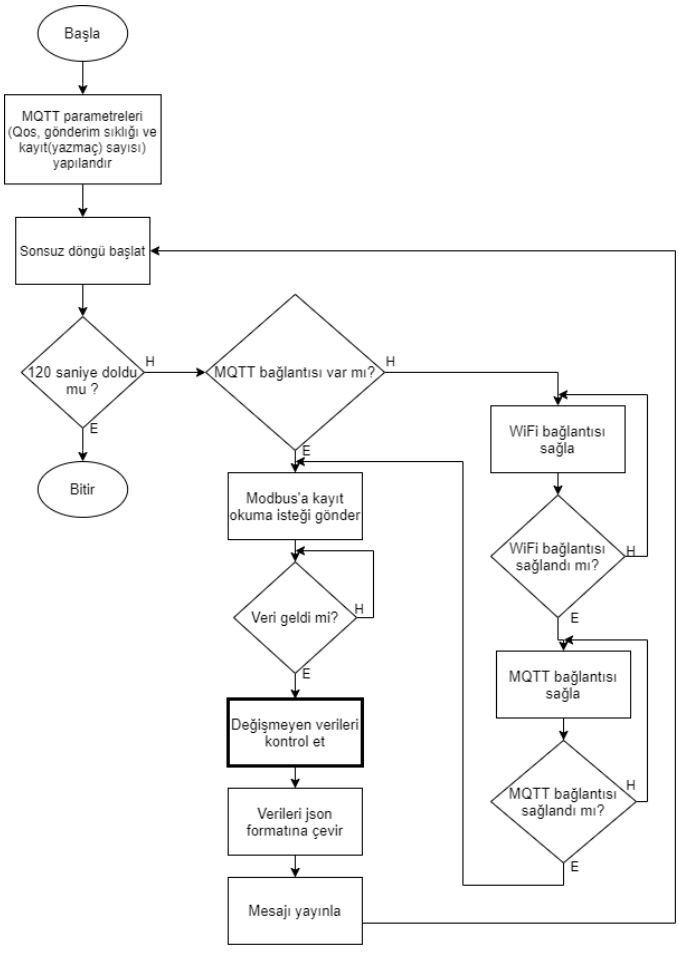
Algoritma_1 –GatewayMesajYayınlama(Data)

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

1. Mesaj yayınlama programını başlat
2. MQTT ve WiFi bağlantıları için kullanıcı adı, parola ve sunucu parametreleri tanımla
3. MQTT ve WiFi 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 3. Mesaj yayınlama algoritması

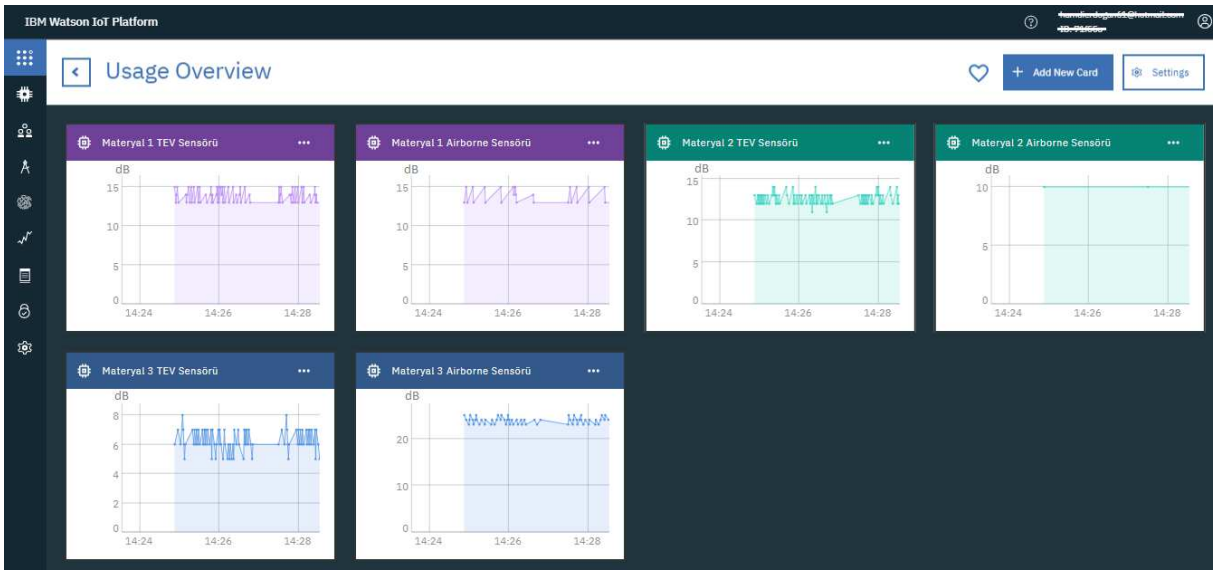
Şekil 3'te verilen algoritmaya ait akış diyagramı Şekil 4'te verilmiştir.



Şekil 4. MQTT ile mesaj yayınlama

C. *Verilerin Bulut Platformunda Görselleştirilmesi*

Ağ geçidinin üçüncü kısmı olan verilerin bulut platformunda görselleştirilmesi aşamasında IBM Watson IoT platformu kullanılmıştır. IBM Watson IoT platformu aynı zamanda MQTT sunucusu olarak görev yapmaktadır. Platforma veri gönderildiği gibi, platformda çalışan broker üzerinden veriler okunabilmektedir. Bulut platformuna bağlantı sağlayabilmek için üyelik işlemi tamamlandığında benzersiz kimlik numarası oluşturulmaktadır. Platforma veri gönderecek cihaz platforma kaydedilirken, cihazı sistem üzerinde isimlendirmek için belirlenen bir isim ve cihazın MAC adresi kullanılarak cihaz platforma kaydedilmektedir. Cihaz eklendikten sonra otomatik olarak token kodu oluşturulur. Bağlantı sağlamak için benzersiz kimlik numarası, cihaza verilen isim ve cihazın MAC adresi birleştirilerek istemci benzersiz kimliği oluşturulur. Kullanıcı adı olarak platformda sabit olarak "use-token-auth" kullanılır. Parola olarak ise otomatik oluşturulan token kullanılmaktadır. Belirtilen parametreler oluşturulduktan sonra MQTT kullanılarak bulut platformu ile bağlantı sağlanabilmektedir. Verilerin görselleştirilmesi için platformda tablolar oluşturulur. Tablo oluşturulurken etkinlik (event) ve konu (topic) adları belirlenir. Veriler etkinlik ve konu adlarına göre veriler MQTT yayıncı tarafından yayınlanarak platformda oluşturulan tablolarda görüntülenir. Platformda oluşturulan her bir tabloda sahadan toplanan bir adet sensöre ait veri gösterilmektedir. Bulut Platformunda verilerin görselleştirilmesi Şekil 5'te verilmiştir.



Şekil 5. IBM Watson IoT Platformu

V. PERFORMANS VE TEST ANALİZİ

Bu bölümde tasarlanan ağ geçidinin veri iletimi üzerinde yapılan testler ile elde edilen performans sonuçları verilmiştir. Test aşamasında tasarlanan ağ geçidinde kullanılan Modbus ve MQTT protokollerinde performansa etki edebilecek Modbus köle sayısı, Modbus okunacak kayıt sayısı, okunan verilerin hassasiyeti ve MQTT QoS seviyesi parametreleri üzerinde farklı varyasyonlar uygulanarak ağ geçidinin performansı ölçülmüştür. Test sonuçları, veriler üzerindeki değişimi kontrol edebilen algoritma ile tasarlanan sistem ve herhangi bir kontrol algoritması bulunmayan sistem olmak üzerinde iki yarı sistem kullanılarak elde edilmiştir.

Geliştirilen sistem, Windows 10 işletim sisteminde yerelde çalışan Mosquitto sunucusu kullanılarak test edilmiştir. Modbus protokolündeki köle cihazlar ve köle cihazlardan okunacak kayıt sayıları performans üzerinde belirleyici etken olarak rol oynamaktadır. MQTT protokolünde ise verilerin yayınlamasında kullanılan servis kalitesi seviyeleri ve veri yükü performansa etki etmektedir. Test süresince bu parametrelerin sistem üzerindeki etkileri gözlemlenmiştir.

Sistem bileşenlerinden olan Modbus master, köle cihazlara gömülen PD Annunciator cihazından gerçek zamanlı okunan sensör verilerini toplamaktadır. Test aşamasında bu verilerin 120 saniyelik süre içerisinde elde edilen sonuçları kullanılmıştır. Köle cihazlardan okunan kayıt sayılarının belirlenmesinde gerçek veriler, sensör sayısının katları şekilde birleştirilmiştir. Okunan her bir kayıt köle cihaza bağlı olan bir adet sensörden alınan değere karşılık gelmektedir. Performansı etkileyecek olan diğer bir Modbus parametresi ise köle sayısıdır. Test sonuçları master cihaza 1 ve 2 adet köle cihaz bağlanarak iki ayrı durumda incelenmiştir. Prototipte yapılan testlerde MQTT mesaj iletimi için QoS 0 ve QoS 1 seviyelerini kullanılmıştır. MQTT mesaj yükü ağ trafiğini ve sistem performansını olumsuz etkilemektedir. Bu doğrultuda tasarlanan kontrol algoritmasına sahip sistem belirtilen hususlara katkı sağlamayı amaçlamaktadır. Verilerin değişmeme durumunda veri hassasiyetinin etkisi göz ardı edilmeyecek kadar önemlidir. Veri setindeki değerler üç farklı hassasiyette kullanılmıştır. Okunan değerler tam sayı, onda bir ve yüzde bir olacak şekilde bulut platformuna aktarılmaktadır. Tasarlanan kontrol algoritmasına sahip sistemin etkisi belirtilen üç farklı hassasiyette incelenmiştir.

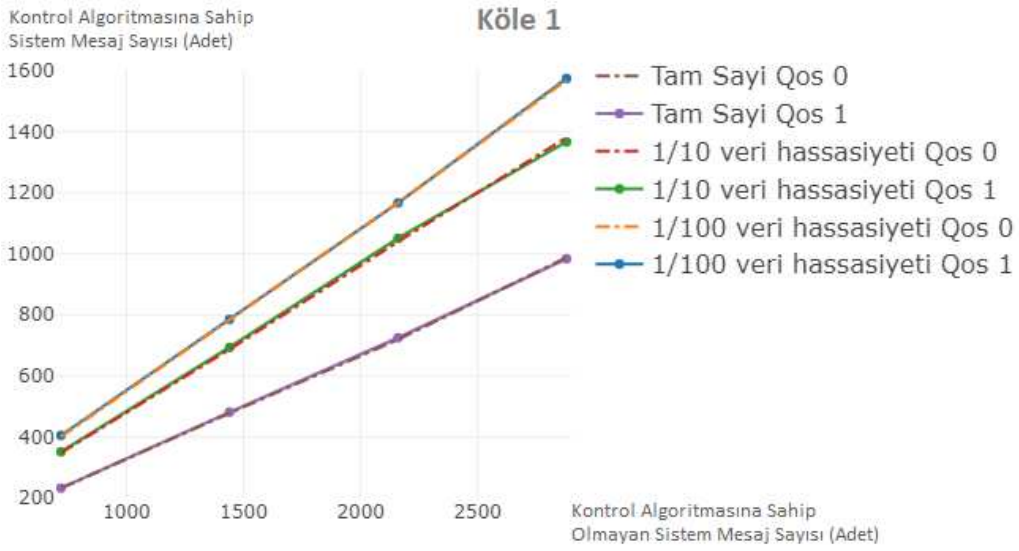
Tablo 1'de ağ trafiği ve sistem performansı üzerinde tasarlanan kontrol algoritmasına sahip sistemin ve herhangi bir kontrol algoritması bulunmayan sistemin etkileri verilmiştir. Elde edilen sonuçlar 120 saniyelik ölçümlere dayanmaktadır. İki sistem için, köle sayısı, QoS seviyesi, okunan kayıt sayısı ve veri hassasiyeti üzerinde farklı varyasyonlar kullanılarak bulut platformuna gönderilen toplam kayıt sayıları (herbir sensörden okunan değer) verilmiştir. Örnek olarak köle sayısı 1, QoS seviyesi 0 ve köle 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 1. Kontrol algoritmasına sahip olan ve olmayan sistemler için toplam gönderilen kayıt sayıları

Köle 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		
			Tam Sayı	1/10 veri hassasiyeti	1/100 veri hassasiyeti	Tam Sayı	1/10 veri hassasiyeti	1/100 veri hassasiyeti
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

Şekil 6'da Modbus köle 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 6. 1 köle için gönderilen toplam kayıt sayısı

Şekil 6'da köle sayısı 1 iken gönderilen toplam kayıt sayıları verilirken, Şekil 7'de ise köle sayısı 2 olduğunda gönderilen toplam kayıt sayılarına ait sonuçlar verilmiştir. Köle 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 köle sayısındaki ve kayıt sayılarındaki artışın, kontrol algoritmasına sahip sistem sayesinde ağ üzerindeki yüklenmeye etkisini minimum düzeye indirgeye yardımcı olmaktadır.

Şekil 7. 2 köle için gönderilen toplam kayıt sayısı

BŞEÜ Fen Bilimleri Dergisi
7(1), 170-183, 2020

BSEU Journal of Science
DOI: 10.35193/bseufbd.708445

e-ISSN: 2458-7575 (http://dergipark.gov.tr/bseufbd)

Tablo 2'de sistemin performansına etki edecek Modbus köle 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 köle 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 köle cihazından maksimum 29 kayıt okuyabilir olmasıdır [22]. Kayıt sayılarında önemli bir artış göstermemesine rağmen QoS 0 seviyesinde tutarlı değişimler gözlemlenmektedir.

Tablo 2. Kontrol algoritmasına sahip olan ve olmayan sistemler için ortalama gecikme süreleri

Köle 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)		
			Tam Sayı	1/10 veri hassasiyeti	1/100 veri hassasiyeti	Tam Sayı	1/10 veri hassasiyeti	1/100 veri hassasiyeti
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

VI. SONUÇLAR

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 vermiştir. Mesajların gönderilmesinde ağ trafiğindeki iyileşmeye bakıldığında, bir mesajın boyutu ortalama 50 bayttır. Genel olarak bir örnek üzerinde bakıldığında Tablo 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 desteklemektedir. Uygulamalarda kullanılabilirlik açısından hem maliyet olarak hem de performans olarak tercih edilebilir.

TEŞEKKÜR

Yazarlar, bu çalışmanın gerçekleştirilmesinde her türlü katkı ve desteklerini sağlayan GENETEK Güç Enerji Elektrik Sistemleri Özel Eğitim ve Danışmanlık San. Tic. Ltd. Şti. teşekkürlerini sunmaktadır.

KAYNAKLAR

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

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

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

[4] Deveci, F. (2016). *Tarihi Haberleşme Metodu: Modbus RTU*. http://www.firatdeveci.com/, http://www.firatdeveci.com/tarihi-haberlesme-metodu-modbus-rtu/, (25.10.2019).

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

[6] Astarloa, A., Bidarte, U., Jimenez, J., Zuloaga, A., Lazaro, J. (2018). Intelligent gateway for Industry 4.0-compliant production. IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society, 23-26 October 2016, Italy, 4902-4907.

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

[8] IEEE Standart (2006). IEEE Recommended Practice for the Design of Reliable Industrial and Commercial Power Systems, in IEEE Std P493/D4.

[9] IBM. (2015). *IBM Watson IoT Platform*. https://www.ibm.com/watson/, https://www.ibm.com/cloud/watson-iot-platform, (06.12.2019).

[10] Lee, S., Kim, H., Hong, D.K., Ju, H. (2013). Correlation Analysis of MQTT Loss and Delay According to QoS Level, The International Conference on Information Networking 2013 (ICOIN), 11 April 2013, 714-717.

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

[12] Nguyen-Hoang, P., Vo-Tan, P. (2019). Development An Open-Source Industrial IoT Gateway. 2019 19th International Symposium on Communications and Information Technologies, 25-27 September 2019, Vietnam, 201-204.

[13] 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, 24-26 June 2018, Romania, 175-179.

[14] Shu, F., Lu, H., Ding, Y. (2019). Novel Modbus Adaptation Method for IoT Gateway, 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), 06 June 2019, China, 632-637.

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

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

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

[18] Koyanagi, F. (2018). *Arduino MEGA 2560 With WiFi Built-in - ESP8266*. https://www.instructables.com/, https://www.instructables.com/id/Arduino-MEGA-2560-With-WiFi-Built-in-ESP8266/, (28.11.2019).

[19] Erman, Z. (2017). *Arduino MEGA 2560*. http://roboromania.ro/, http://roboromania.ro/datasheet/Arduino-Mega-2560-roboromania.pdf, (28.11.2019).

[20] Genetek. *PD Annunciator - GENETEK Güç, Enerji, Elektrik Sistemleri*. https://genetek.com.tr/, https://genetek.com.tr/wp-content/uploads/2019/03/PD-Annunciator-TR.pdf, (18.09.2019).

[21] Bayılmış, C., Küçük, K. (2019). Nesnelerin İnterneti: Teori ve Uygulamaları. Papatya Bilim, Türkiye, 256.

[22] Crespo, E. (2016). *Simple Modbus Master*. Github, https://github.com/jecrespo/simple-modbus/blob/master/Modbus%20RTU%20libraries%20for%20Arduino/SimpleModbusMasterManual.pdf, (28.11.2019).