

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MODBUS RTU OTOMASYON PROTOKOLÜNÜN RS485 SERİ HAT
ÜZERİNDE ÇALIŞAN HABERLEŞME KATMANINDA
UYGULANACAK BİR YÖNTEMLE HIZININ ARTTIRILMASI

SALİH BAKIR
YÜKSEK LİSANS TEZİ
ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

GEBZE
2019

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MODBUS RTU OTOMASYON
PROTOKOLÜNÜN RS485 SERİ HAT
ÜZERİNDE ÇALIŞAN HABERLEŞME
KATMANINDA UYGULANACAK BİR
YÖNTEMLE HIZININ ARTTIRILMASI

SALİH BAKIR
YÜKSEK LİSANS TEZİ
ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMANI
DOÇ. DR. FUAD ALIEW

GEBZE
2019

T.R.
GEBZE TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

**IMPROVING COMMUNICATION SPEED
OF THE MODBUS RTU AUTOMATION
PROTOCOL WHICH WORKS ON RS485
SERIAL LINE WITH A METHOD APPLIED
IN THE COMMUNICATION LAYER**

SALİH BAKIR
**A THESIS SUBMITTED FOR THE DEGREE OF
MASTER OF SCIENCE**
DEPARTMENT OF ELECTRONIC ENGINEERING

THESIS SUPERVISOR
ASSOC. PROF. DR. FUAD ALIEW

GEBZE

2019

GTÜ Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 23/01/2019 tarih ve 2019/05 sayılı kararıyla oluşturulan jüri tarafından 18/02/2019 tarihinde tez savunma sınavı yapılan Salih Bakır'ın tez çalışması Elektronik Mühendisliği Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRİ

ÜYE

(TEZ DANIŞMANI) : Doç. Dr. Fuad ALIEW



ÜYE

: Doç.Dr. Fatih ERDEN



ÜYE

: Dr. Öğr. Üyesi Erkul BAŞARAN



ONAY

Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun
...../...../..... tarih ve/..... sayılı kararı.

ÖZET

Dijitalleşme ile mevcut kapalı ağlarda çalışan ya da tek başına çalışan cihazlar dahi artık internete bağlanmaya başlamıştır. Bu akım kendini nesnelerin interneti adı altında tanıtmıştır. Her yeni cihaz her geçen gün gelişen donanım parçaları ile daha güçlü ve hızlı bir şekilde internete bağlanabilmektedir. Günümüzde sahada eski ve zayıf donanım parçalarına sahip olan cihazların, yeni cihazların hızlarına ulaşmaları mümkün olmamaktadır. Bu cihazlar genellikle kapalı özel bir ağda oldukları için sadece birbirleriyle özel haberleşme protokolleri ile konuşmaktadır. Bu cihazların çoğunun doğrudan internete bağlanması mümkün olmadığından, internet bağlantıları ancak ağ geçitleri aracılığıyla gerçekleşmektedir. Bu kapalı ağ protokollerinden biri de Modbus RTU'dur. Günümüzdeki hızlar göz önüne alındığında Modbus RTU protokolünün desteklediği hızlar oldukça düşük kalmaktadır. Ağ geçitlerinin internete bağlanma hızı yüksek olmasına rağmen Modbus RTU protokolünün mevcut haberleşme hızı ile darboğaz oluşturması kaçınılmazdır. Kapalı ağ içinde yapılacak hız iyileştirmeleri, sınırlı hızla haberleşen sistemlerin performansını artırmak açısından oldukça önem kazanmaktadır. Hız iyileştirmesi için mevcut cihazların donanımlarını değiştirmek mümkün olmadığından, geriye uyumluluğu bozmadan protokolünün kendisinde yapılması gerekmektedir. Bu çalışmada bir otomasyon protokolü olan Modbus RTU üzerinde, haberleşme hızını iyileştiren bir yöntem ortaya konmuştur. Örnek bir Modbus RTU ağında yapılan testler ile yöntemin haberleşme hızındaki etkisi gösterilmiştir.

Anahtar Kelimeler: Modbus RTU, RS-485, Modbus CRC-16, Otomasyon Protokolleri, Haberleşme Hızı İyileştirme, Nesnelerin İnterneti.

SUMMARY

With digitalization, electronic devices that are working in a closed network or as a standalone are started to connect to the internet. Nowadays, this trend has introduced itself as the Internet of Things. Each new device can be connected to the internet more rapidly than it was before, by having more advanced hardware components. Devices with old and slow hardware components in the field now are not able to reach the speed of new devices. These devices are usually in a private network which are closed to the outside world, and they only communicate with each other through special communication protocols. Most of these devices are not available on the internet, and accessing the internet is only possible via the IoT Gateways. One of these closed network protocols is Modbus RTU. Considering the current communication speeds, the Modbus RTU protocol supports very low communication speeds. Although the speed of the gateways that are connected to the internet is high, it is inevitable that the Modbus RTU protocol will create a bottleneck with the current communication speed. The communication speed improvements to be made in the closed network gain importance in terms of increasing the performance of the limited communication systems. The communication speed improvement is not possible by changing the equipment of existing devices in the field; it should be done in the protocol itself without breaking the backward compatibility. In this study, a new method is introduced for Modbus RTU, which is an automation protocol, to increase its communication speed. By applying tests on Modbus RTU network is shown the effect of the method on the communication speed.

Key Words: Modbus RTU, RS-485, Modbus CRC-16, Automation Protocols, Communication Speed Improvement, Internet of Things.

TEŞEKKÜR

Başta yüksek lisans eğitimimde ve akademik hayatımda desteğini ve yardımlarını hiçbir zaman esirgemeyip bilgisi ile bu çalışmanın oluşmasının yolunu açan danışmanım Doç. Dr. Fuad ALIEW'e, yüksek lisans eğitimimi almış olduğum bilim yuvası Gebze Teknik Üniversitemizin üzerimde emeği bulunan tüm akademik ve idari birimlerinin yönetici ve personellerine en derin şükranlarımı arz ediyorum.

Son olarak tez çalışmamda yardımcı olan Zeynep ARSLAN'a, manevi desteğini her an yanımda hissettiğim aileme ve sevgili eşim Melike BAKIR'a en içten teşekkürlerimi sunarım.

İÇİNDEKİLER

	<u>Sayfa</u>
ÖZET	i
SUMMARY	vi
TEŞEKKÜR	vii
İÇİNDEKİLER	viii
SİMGELER ve KISALTMALAR DİZİNİ	ix
ŞEKİLLER DİZİNİ	xi
TABLolar DİZİNİ	xiii
1. GİRİŞ	1
1.1. Tezin Amacı, Katkısı ve İçeriği	2
1.2. Literatür Özeti	3
2. ENDÜSTRİYEL OTOMASYON	5
3. MODBUS HABERLEŞME PROTOKOLÜ	8
3.1. Modbus Seri Hat Protokolleri	12
3.1.1. Modbus RTU Protokolü	14
3.1.2. Modbus ASCII Protokolü	17
3.2. Modbus TCP/IP Protokolü	19
4. YÖNTEM	21
4.1. Parça Karakter CRC Hesaplama Yaklaşımı	21
4.2. Zaman Aşımı Atlatma Yaklaşımı	22
4.3. Parça Karakter CRC Hesaplamaya Bağlı Zaman Aşımı Atlatma Yöntemi	24
5. UYGULAMA VE YAZILIM	27
5.1. Donanım Tasarımı	27
5.2. Yazılım Tasarımı	33
5.3. Devre, Yapılan Testler ve Sonuçlar	35
6. SONUÇLAR ve YORUMLAR	41
KAYNAKLAR	43
ÖZGEÇMİŞ	46
EKLER	47

SİMGELER ve KISALTMALAR DİZİNİ

<u>Simgeler ve</u>	<u>Açıklamalar</u>
<u>Kısaltmalar</u>	
ADU	: Application Data Unit
AREF	: Analog Read Reference
ASCII	: American Standard Code for Information Interchange
CAN	: Control Area Network
CR	: Carriage Return
CRC	: Cyclic Redundancy Check
EEPROM	: Electronically Erasable Programmable Read-Only Memory
EIA	: Electronic Industries Alliance
FPGA	: Field programmable gate array
G/Ç	: Giriş/Çıkış
GND	: Ground
HEX	: Hexadecimal
HMI	: Human Machine Interface
ID	: Identifier
IDE	: Integrated Development Environment
IEC 61107	: International Electrotechnical Commission 61107
IEEE	: Institute of Electrical and Electronic Engineers
I/O	: Input/Output
IoT	: Internet of Things
IP	: Internet Protocol
ISO	: International Standards Organization
I2C	: Inter-Integrated Circuit
KB	: Kilobyte
LED	: Light Emitting Diode
LF	: Line Feed
LRC	: Longitudinal Redundancy Check
LSB	: Least Significant Bit
mA	: Mili Amper
mm	: Milimetre

MBAP	: Modbus Application Protocol Header
MISO	: Master In, Slave Out
MOSI	: Master Out, Slave In
Ms	: Milisaniye
MSB	: Most Significant Bit
OSI	: Open System Interconnection
OS	: Operating System
PDU	: Protokol Data Unit
PLC	: Programmable Logic Controller
POLY	: Polynomial
PROFIBUS-DP	: Process Field Bus - Decentralised Peripherals
PROFIBUS-FMS	: Process Field Bus - Field bus Message Specification
PWM	: Pulse Width Modulation
QNX	: QNX Software Systems (Realtime OS)
RS232	: Recommended Standard for 232
RS485	: Recommended Standard for 485
RTU	: Remote Terminal Unit
RX	: Receive
SCADA	: Supervisory Control and Data Acquisition
SCK	: Serial Clock
SDA	: Serial Data
SPI	: Serial Peripheral Interface
SRAM	: Static Random Access Memory
SS	: Slave Select
TCP	: Transmission Control Protocol
TIA	: Telecommunications Industry Association
TTL	: Transistor Transistor Logic
TX	: Transmit
USB	: Universal Serial Bus
V	: Volt
XOR	: Exclusive Or
3G	: 3rd Generation

ŞEKİLLER DİZİNİ

<u>Sekil No:</u>	<u>Sayfa</u>
1.1: Nesnelerin internetinin çalışma yapısı.	1
2.1: Endüstriyel otomasyonun gelişimi.	5
2.2: Endüstri devrimleri	6
3.1: Modbus ağ mimarisi.	8
3.2: Modbus haberleşme katman yapısı.	9
3.3: Genel Modbus protokolü yapısı.	9
3.4: Örnek Modbus sorgu-cevap çalışma yapısı.	10
3.5: (0x03) Köle cihazdan tutucu yazmaç değeri okuma sorgu yapısı.	11
3.6: 108-110 adreslerindeki yazmaçları okumak için örnek bir sorgu.	11
3.7: Modbus Protokolü ve OSI Model katman yapısı.	12
3.8: Örnek Modbus ağındaki efendi-köle sorgusu.	13
3.9: Modbus protokolünün seri hattaki paket yapısı.	13
3.10: Eşlik bitine göre bir bayt Modbus RTU karakter yapısı	14
3.11: Detaylı Modbus RTU paket yapısı.	15
3.12: Modbus RTU CRC-16 hesaplama algoritması.	15
3.13: Detaylı Modbus ASCII paket yapısı.	17
3.14: Eşlik bitine göre bir bayt Modbus ASCII karakter yapısı.	18
3.15: Modbus TCP/IP protokolü haberleşme yapısı.	19
3.16: Örnek Modbus TCP/IP protokolü haberleşme ağı.	19
3.17: Modbus TCP protokolü paket yapısı.	20
4.1: Örnek Modbus RTU protokolü ağı.	21
4.2: Modbus RTU protokolü karakterler arası zaman aralığı.	23
4.3: Modbus RTU protokolü paketler arası zaman aralığı.	23
4.4: Uygulanan yöntemin Modbus RTU durum diyagramındaki yeri.	25
5.1: Arduino Uno R3 kartı.	27
5.2: Arduino Uno kartının devre şeması.	29
5.3: Arduino Uno devre kartındaki pinlerin konumu.	31
5.4: Arduino MAX485 TTL/RS485 bağlantı dönüştürücü modülü ve MAX485 entegresi.	32
5.5: Arduino kartı için yazılım geliştirme ortamı.	33

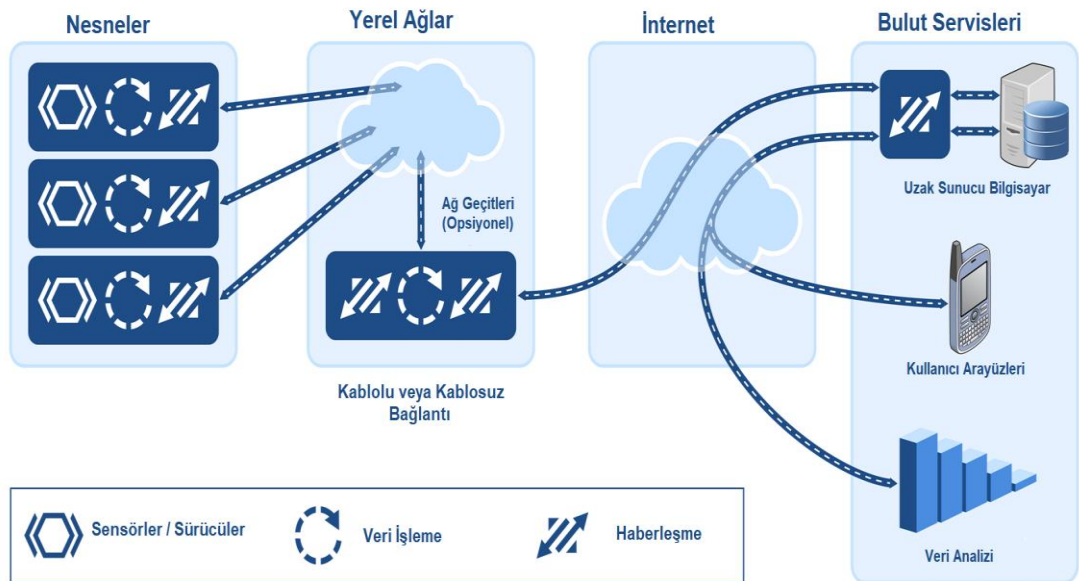
5.6.	Temsili Modbus RTU ağı test düzeneği.	36
5.7:	Modbus Poll programı ve Modbus RTU protokolü haberleşme ayarı.	37
5.8:	Modbus Poll programı üzerinden Modbus RTU haberleşme trafiği izleme.	37
5.9:	Ortalama $t_{1,5}$ karakter zaman aşımı kazancı.	38
5.10:	Ortalama parça karakter CRC hesaplama kazancı.	39

TABLÖLAR DİZİNİ

<u>Tablo No:</u>	<u>Sayfa</u>
3.1: Modbus cihazı yazmaç yapısı.	10
3.2: Genel Modbus haberleşme protokolü fonksiyon kodları	11
3.3: Modbus Protokolü hata kodları.	12
3.4: Seri hat üzerindeki bir bayt için Modbus RTU karakter formatı.	14
3.5: Modbus CRC-16 hesaplama kodu.	16
3.6: Seri Hat üzerindeki bir bayt için Modbus ASCII karakter formatı.	17
3.7: Modbus ASCII LRC hesaplama kodu.	18
3.8: Modbus TCP protokolü MBAP başlık yapısı.	20
5.1: Arduino Uno kartının teknik özellikleri.	28
5.2: MAX485 entegresinin pinlerinin açıklaması.	32
5.3: Parça karakter CRC hesaplama kodu.	34
5.4 Yöntemin uygulandığı kod parçası.	35
5.5: Örnek detaylı Modbus RTU efendi-köle haberleşmesi.	36
5.6: Uygulanan yöntemin, yaklaşıma bağlı paket başına zaman kazançları.	39

1. GİRİŞ

Nesnelerin interneti için farklı tanımlar mevcut olsa da nesnelerin interneti fiziksel cihazların birbirleriyle ya da daha büyük sistemlerle konuşabildiği ağlar bütünü olarak tanımlanabilir. Her cihazın kendine ait bir adresinin olması ile birlikte karışıklık olmadan ortak çalışarak daha büyük değerler üretmesi amaçlanmıştır. Nesne kavramının çok geniş bir anlama sahip olmasının yanında burada bahsedilen nesnelerin elektronik cihaz olarak veri toplayıp, birbiriyle iletişime geçebilen ve kendi aralarında veri alışverişi yapabilen cihazlar olması amaçlanmıştır. Nesnelerin interneti özellikle endüstri, enerji, tüketici elektroniği, sağlık ve tarım gibi alanlarda kullanılmaktadır [1]. Bu akım ortaya çıkmadan önce, bu akımı mümkün kılacak önemli teknolojik gelişmeler yaşandı. Zaman içinde gelişen teknoloji ile hem daha güçlü işlemci gücüne sahip hem de saklama alanına sahip süper bilgisayarlar ortaya çıktı. Haberleşme hızının artması ve internetin gelişmesi ile de herhangi bir yerdeki veriye kısa süreler içinde erişim mümkün hale geldi. Endüstri, enerji, sağlık, tüketici elektroniği gibi alanlardaki cihazların daha akıllı hale gelmesi ve bu cihazların ürettiği veri miktarının artması, akıllara bu verilerden bir değer elde edilebilir mi sorusunu getirdi. Bu süreçte gelişen yapay zekânın bu sorunun cevabını verebileceği ortaya çıktı ve bu şekilde verilerden katma değer elde etmek mümkün hale geldi.



Buna paralel olarak sahadaki akıllı cihazlardan üretilen veriler ağ geçitleri vasıtasıyla Şekil 1.1'deki gibi uzak sunucu bilgisayarlara yollanmaya başlandı. Bu şekilde üretilen veriler internet vasıtasıyla daha güçlü uzaktaki bilgisayarlara yollanıp yüksek kapasite ile depolandı [2]. Depolanan veriler yapay zekâ yardımıyla işlenerek katma değeri yüksek işler mümkün hale geldi.

1.1. Tezin Amacı, Katkısı ve İçeriği

Endüstrideki uygulamalarda, sahadaki mevcut kapalı ağların genelde bir ağ geçidi vasıtasıyla internete bağlanarak daha büyük sistemlerle iletişime geçebilmesi sağlanır. Her kapalı ağ kendi içinde farklı bir otomasyon protokolü ile haberleşse de bu şekilde merkezi bir yerden tüm ağlar kontrol edilerek daha büyük sistemler oluşturulur. Genelde endüstride kullanılan bu kapalı devre otomasyon ağları günümüzden yıllar önce kurulduğu için bugünkü şartlara göre haberleşme hızı anlamında yavaş kalmaktadırlar.

Bu çalışmada endüstriyel otomasyon haberleşme protokollerinden biri olan Modbus RTU (Remote Terminal Unit - Uzaktan Bağlantı Ünitesi) protokolünün çalışma yapısı ve haberleşme hızını iyileştirme üzerine odaklanılmıştır. Daha önce geliştirilen Modbus RTU CRC-16 (Cyclical Redundancy Checking - Döngüsel Artıklık Denetimi) hesaplama algoritması bu yöntem için temel alınmıştır [3].

Modbus RTU protokolünün haberleşme katmanında paketin geldiğinin anlaşılması için bazı zaman gecikmeleri mevcuttur. Bu zaman gecikmelerinin, Modbus RTU protokolü içinde bulunan CRC hesaplama algoritmasından faydalanan bir yöntem ile daha akıllı bir şekilde kullanılmasıyla haberleşme hızının artırılması hedeflenmiştir.

Geliştirilen yöntem ile Modbus RTU/RS485 alt yapısını kullanan büyük sistemlerde, haberleşme trafiğinin artması durumunda karşılaşılan hız problemleri azaltılacaktır. Yöntemin mevcut Modbus RTU haberleşme protokolüne uyumlu olması ve kolay uygulanabilirliği mevcut otomasyon sistemlerinde kullanımını teşvik edecektir. Yöntemin mevcut fiziksel katmanlardan (RS232/RS485) bağımsız olması, Modbus RTU protokol standartlarına uygun yazılım geliştirilmesine olanak sağlayacaktır.

1.2. Literatür Özeti

Endüstriyel alanda oldukça yaygın kullanılan Modbus protokolü bağımsız bir kuruluş olan "Modbus Organization" tarafından standartları belirlenen ve geliştirilen bir protokoldür. Modbus protokolünün uygulama katmanındaki çalışma yapısı "Modbus Application Protocol Specification V1.1b3." standardizasyon dokümanında belirtilmiştir [4]. Seri hatlar üzerinde çalışan Modbus RTU ve ASCII (American Standard Code for Information Interchange - Amerikan Standart Kodlama Sistemi) protokolünün fiziksel katman üzerindeki uygulaması "Modbus over Serial Line Specificaiton and Implementation guide V1.02" dokümanında anlatılmıştır [5]. Ethernet üzerinde çalışan Modbus TCP/IP (Transmission Control Protocol / Internet Protocol) protokolünün uygulama şekli "Modbus Mesaging on TCP/IP Implementation Guide V1.0a" dokümanında tanımlanmıştır [6]. Bu dokümanların tamamına "Modbus Organization" resmi sitesinden ücretsiz bir şekilde erişim mümkündür.

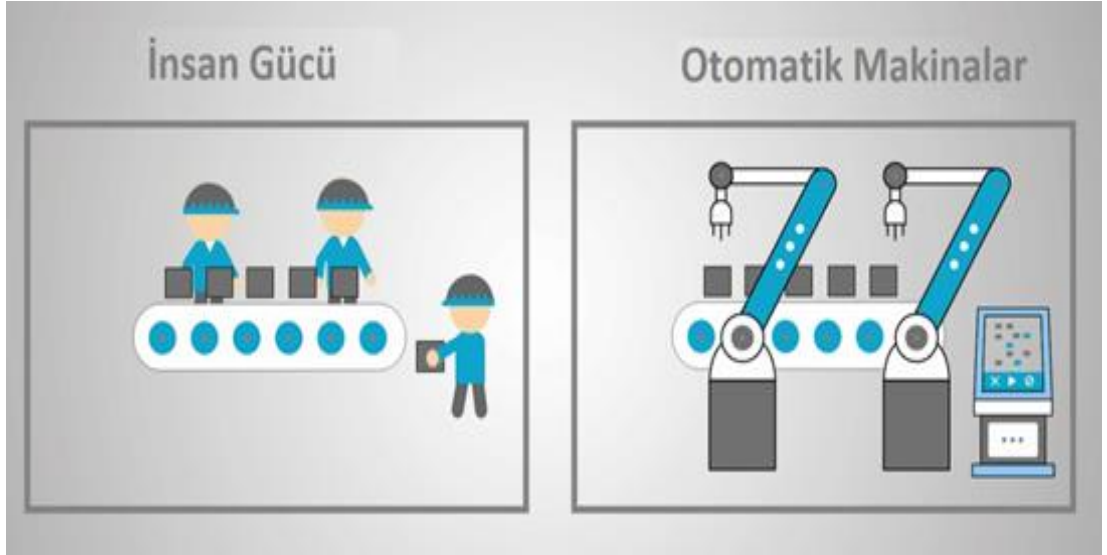
Konu ile ilgili yapılan uluslararası akademik çalışmalardan kısaca bahsetmek faydalı olacaktır. Modbus RTU protokolünün RS485 hattı üzerinde geliştirilmiş bir uygulaması incelenmiştir. Bu uygulama 32-bit ARM Coretx-MO mikrodenetliyiçi üzerinde gerçek zamanlı bir işletim sistemi olan μ C/OS-II ile geliştirilmiştir. Geliştirilen uygulama ile Modbus RTU protokolünün enerji endüstrisinde kullanılabilecek kadar güvenilir ve hızlı olduğu gösterilmiştir [7][8]. Modbus protokolünün QNX (QNX Software Systems) işletim sistemi üzerinde çalışan bir uygulaması geliştirilmiştir. Bu çalışmada, yapılan uygulamanın SIEMENS S7-200 PLC (Programmable Logic Controller) ile yapılan testler ile hızlı, kararlı ve güvenilir olduğu gösterilmiştir [9]. Endüstriyel kontrol sistemlerinde yaygın kullanılan iki haberleşme protokolü olan, seri hat üzerinde çalışan Modbus RTU/ASCII ve CAN protokolün birbirleriyle doğrudan haberleşmesine olanak sağlayan bir arayüz cihazı tasarlanmıştır. Bu şekilde iki tane farklı endüstriyel haberleşme protokolünün birbirleriyle doğrudan haberleşmesine olanak sağlanmıştır [10]. Aynı şekilde Modbus ve Profibus ağlarının haberleşmesini sağlayan bir arayüz ağ geçidi çalışması incelenmiştir [11]. Modbus RTU/ASCII ağını kontrol edip izleyen Linux işletim üzerinde çalışan bir SCADA (Supervisory Control and Data Acquisition) uygulaması incelenmiştir. Çalışmada Modbus Efendi cihazı olarak çalışan SCADA uygulamasının

Linux işletim üzerinde maliyeti düşük, uygulaması kolay, güç tüketimi az ve kararlı olabileceği gösterilmiştir [12]. Arduino üzerinde köle cihazı olarak çalışan Modbus RTU/ASCII protokol uygulaması tasarlanmıştır. Bu tasarımda Mitsubishi Fx2N serisi bir PLC efendi cihazı olarak tanımlanmıştır. Bu şekilde örnek bir Modbus RTU efendi-köle cihazı haberleşmesi geliştirilmiştir [13]. Endüstriyel otomasyonda kullanılan CAN, PROFIBUS-DP, PROFIBUS-FMS, Ethernet, ControlNet, DeviceNet ve MODBUS PLUS protokollerinin performanslarının karşılaştırılması incelenmiştir. Bu şekilde protokollerin kullanım alanına göre üstünlükleri, birbirlerine göre avantajları dezavantajları anlaşılmıştır [14][15]. Nesnelerin internetine ağ geçidi yardımıyla bağlanan örnek bir Modbus ağı incelenmiştir. Bu şekilde Modbus ağındaki veriler buluta aktarılıp, buluttaki Mikroservis mimarisine uygun tasarlanmış servislerin veri işleme performansı gösterilmiştir [16].

Ülkemizde yapılan çalışmalarda ise Modbus RTU ve TCP/IP protokollerini inceleyen, bu protokolleri destekleyen bir Modbus ağıyla uyumlu çalışabilecek FPGA tabanlı bir ağ geçidi tasarlanmıştır. Tasarlanan FPGA tabanlı cihaz örnek bir endüstriyel otomasyon sisteminde test edilmiştir [17]. Başka bir çalışmada dünya üzerinde kullanılan endüstriyel otomasyon sistemlerinin internete açıldığında karşılaştıkları güvenlik saldırılarından bahsedilmiştir. Çalışmada aynı zamanda bir güvenlik saldırısına karşı geliştirilen bir yöntem tanıtılmıştır [18]. Modbus RTU ve TCP/IP protokolleri için 3G haberleşme teknolojisi kullanılarak bir ağ geçidi tasarlanmıştır. Tasarlanan bu ağ geçidi Linux işletim sisteminde koşturma olup veri haberleşmesindeki bazı paketlere öncelik vererek daha akıllı ve verimli çalışmaktadır [19]. Bir başka çalışmada Modbus haberleşme protokolü için uyumlu bir kablosuz haberleşme modülü tasarlanıp test edilmiştir [20]. Bir çalışmada ise endüstriyel otomasyon sistemleri ve uzaktaki sayaçları okumada kullanılan Modbus-RTU ve IEC 61107 Mod C haberleşme protokolleri incelenmiştir. Aynı şekilde iki protokol ile uyumlu çalışabilecek Visual Basic programlama diliyle geliştirilen bir test yazılımı tanıtılmıştır [21].

2. ENDÜSTRİYEL OTOMASYON

Otomasyon, herhangi bir işlemin minimum insan gücüyle gerçekleştirilmesini sağlayan teknolojidir [22]. İnsan gücünün yüksek olduğu otomasyon sistemleri yarı otomasyon olarak adlandırılırken makine gücünün daha etkin olduğu sistemler tam otomasyon sistemler olarak adlandırılır. Otomasyon sistemleri yiyecek-içecek, maden, ulaşım, bina, lojistik ve endüstri gibi birçok alanda kullanılmaktadır. Endüstri alanında kullanım şekli endüstriyel otomasyon olarak adlandırılır. Günümüzdeki endüstriyel otomasyon sistemleri Şekil 2.1’de görüldüğü gibi gittikçe insan gücünden tam otomasyona doğru kaymaktadır [23].



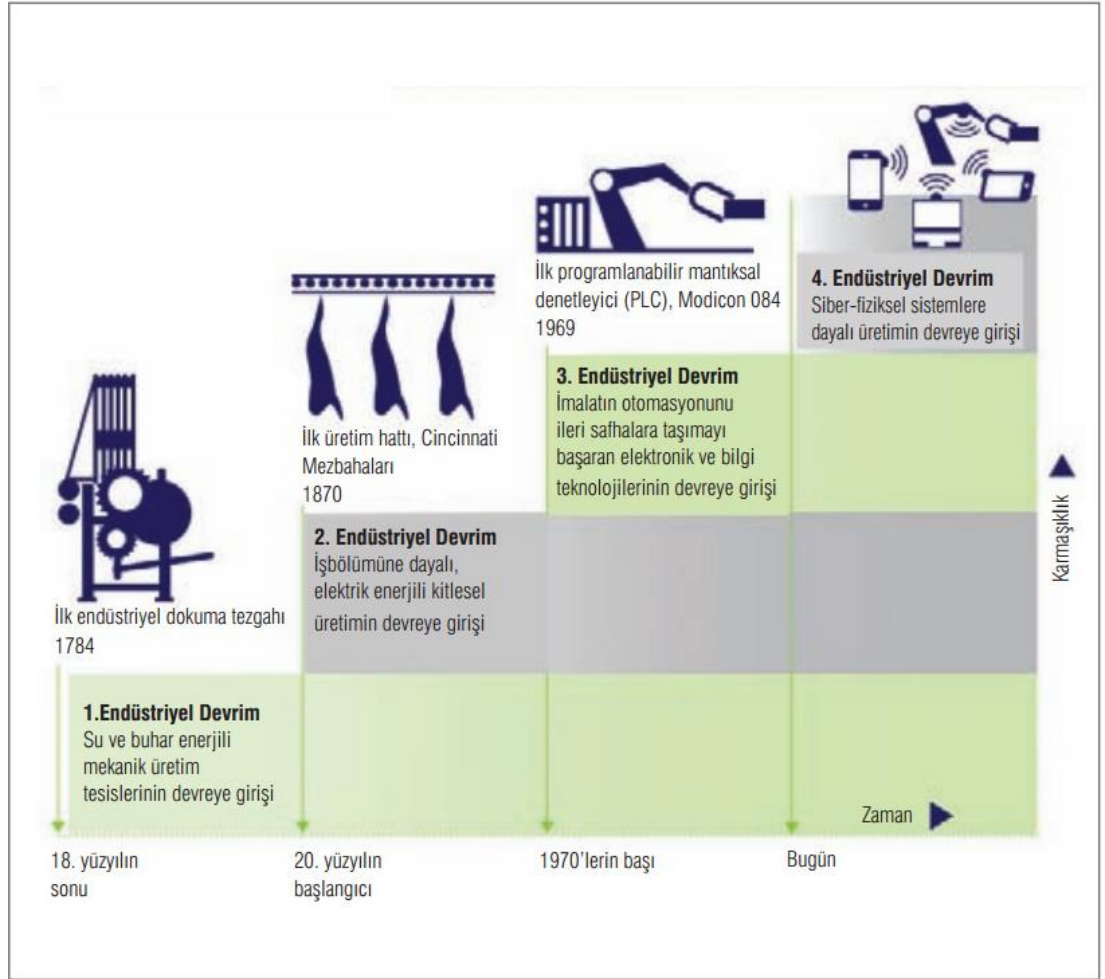
Şekil 2.1: Endüstriyel otomasyonun gelişimi.

Zaman içinde endüstri birçok evrim geçirmiştir. Bu evrim Şekil 2.2’de gösterildiği gibi ilk sanayi devrimi (1.0) su ve buhar gücünü kullanan mekanik üretim sistemleri ile başladı [24]. Su ve buhar gücüne dayalı bu sistemler endüstrinin temelini oluşturur. Su buharı kinetik enerjiye dönüştürülerek sistemdeki pistonlar tetiklenir. Pistonların motor içindeki hareketi, motora bağlı diğer hareketli aksamaların çalışmasını sağlar. Böylelikle istenilen görev, bu enerji akışının tamamlanmasıyla yerine getirilir.

İkinci sanayi devrimi (2.0) elektrik gücünü kullanarak günümüzdeki seri üretim anlayışını ortaya çıkarmıştır. Bu sayede yapılan geniş ölçekli üretim ile büyük kitlelerin ihtiyaçları giderilmiştir.

Üçüncü sanayi devriminde (3.0) mevcut olan üretimin, bilgisayarların artık hayatımıza girmesiyle daha hızlı, verimli ve kaliteli şekilde gerçekleştirilmesi sağlanmıştır. Bu devirde üretim hattında yapılacak değişiklikler PLC sayesinde mevcut donanımına müdahale etmeden sadece yazılımın değişmesiyle mümkün hale gelmiştir.

Son yıllarda ise internetin ve akıllı cihazların gelişmesi ile dördüncü sanayi devrimi (4.0) ortaya çıkmıştır. Kısaca endüstri 4.0 olarak adlandırılan bu devir, sahadaki akıllı cihazların oluşturduğu verilerin değerlendirilip, üretimin optimize edildiği daha akıllı bir üretim sistemidir.



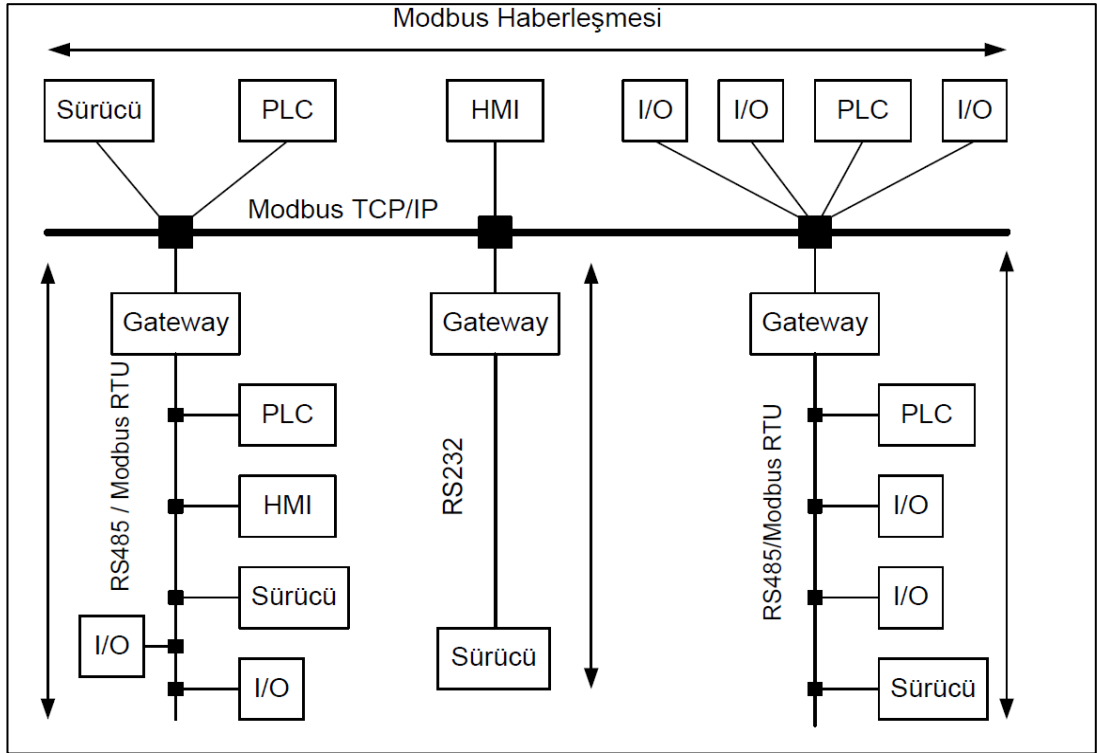
Şekil 2.2: Endüstri devrimleri.

Günümüzdeki herhangi bir endüstrinin işleyişinin arkasında otomasyon sistemleri yatmaktadır. Otomasyon sistemleri sahadaki PLC, giriş/çıkış birimleri, robotlar, motorlar gibi birçok fiziksel cihazdan oluşmakta olup bu cihazlar birbirleri ile özel otomasyon protokolleri ile haberleşmektedir [25]. En yaygın kullanılan protokollerden

bazıları Modbus, Profibus, CAN ve Profinet gibi haberleşme protokolleri örnek gösterilebilir. Bu çalışmada özellikle Modbus protokolünün seri hatlar üzerinde çalışan türü olan Modbus RTU protokolüne değinilecektir.

3. MODBUS HABERLEŞME PROTOKOLÜ

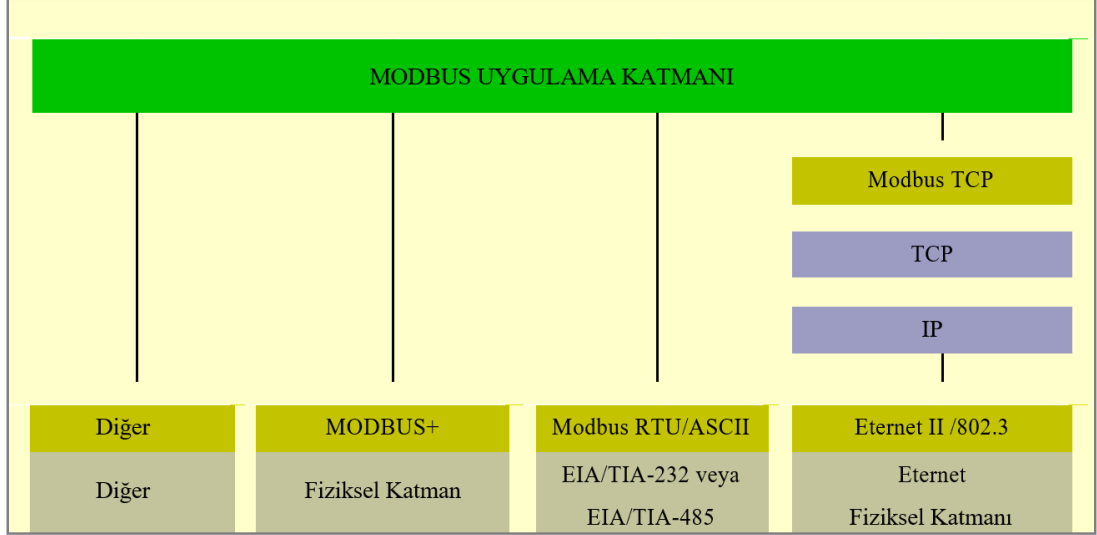
Endüstriyel otomasyon haberleşme protokollerinden biri olan Modbus protokolü, Modicon firması tarafından 1979 yılında PLC'lerin haberleşmesi için geliştirilen bir protokoldür. Kullanımı kolay ve fazla kısıtlaması olmayan bir protokol olduğundan endüstriyel alanda oldukça yaygın kullanılmaktadır. Çalışma mantığı efendi-köle (Master-Slave) yapısına dayanır. Modbus haberleşme ağında PLC, HMI, I/O (Giriş/Çıkış Birimleri), ağ geçitleri (Gateway) ve sürücüler gibi birçok farklı tipte cihaz bulunabilir. Şekil 3.1'de görüldüğü gibi farklı fiziksel katmanlarda çalışan farklı Modbus protokol ağları birbirlerine ağ geçidi aracılığıyla bağlanabilir [4].



Şekil 3.1: Modbus ağ mimarisi.

Şekil 3.2'de görüldüğü gibi Modbus protokolünün seri hatlar üzerinde çalışan Modbus RTU/ASCII olmak üzere iki tane farklı, Ethernet üzerinden çalışan Modbus TCP ve özel bir donanımla çalışan Modbus+ versiyonu vardır [4]. En çok kullanılan versiyonu ise seri hat üzerinde çalışan Modbus RTU'dur [7]. Modbus uygulama katmanı, altındaki diğer katmanlardan bağımsız olarak tüm Modbus protokol çeşitleri için ortaktır. Uygulama katmanındaki veri yapısına protokol veri birimi (Protocol Data

Unit, PDU) adı verilir. Uygulama veri birimi (Application Data Unit, ADU), alt katmanlardaki ilgili Modbus protokol türüne göre tanımlanmıştır. ADU, protokol türüne bağlı olup farklılık gösterir.

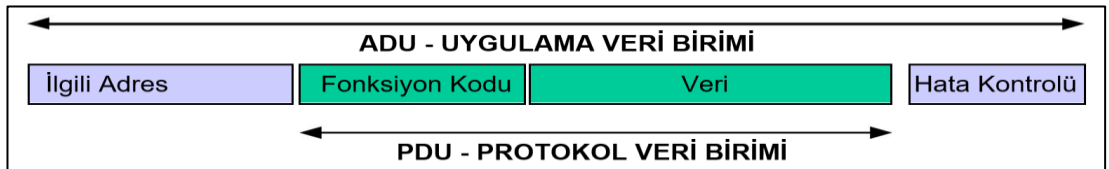


Şekil 3.2: Modbus haberleşme katman yapısı.

Şekil 3.3'te genel Modbus protokolünün hangi kısımlardan oluştuğu görülebilir. Modbus protokolünün ADU kısmı bulunduğu fiziksel katmana göre değişiklik göstermekte olup, boyutu aşağıdaki gibi hesaplanır;

$$\text{RS232/RS485 ADU} = 253 \text{ bayt} + \text{Adres (1 bayt)} + \text{CRC (2 bayt)} = 256 \text{ bayt} \quad (3.1)$$

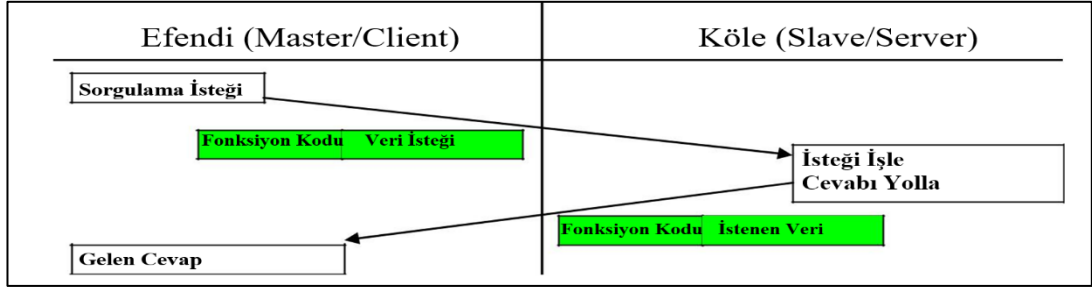
$$\text{Modbus TCP ADU} = 253 \text{ bayt} + \text{MBAP (7 bayt)} = 260 \text{ bayt} \quad (3.2)$$



Şekil 3.3: Genel Modbus protokolü yapısı.

Modbus Protokolü veri kısmı sorgulama, cevap ve hata olmak üzere üç tip veri tipinden oluşur. Örnek bir Modbus Efendi-köle cihazı sorgulaması Şekil 3.4'te görülebilir. Bir Modbus haberleşme paketi içindeki veriler en öncelikli bit (MSB -

Most Significant Bit) önce gidecek şekilde yollarır. Kısaca veri yollarken ve alırken “Big Endian” yaklaşımı kullanılır.



Şekil 3.4: Örnek Modbus sorgu-cevap çalışma yapısı.

Modbus protokolü ile haberleşen cihazlar ilgili verileri yazmaç (Register) adı verilen bölgelerde saklarlar. Yazmaçlar kısaca cihazın bilgi tuttuğu bellekteki yerlerdir. Yazmaçların boyutu 1 bitlik ya da 2 baytlık veri tutacak büyüklüktedir. Yazmaçlar veri modeline göre dört çeşit olabilir. Cihaz yazmaçlarının tutulduğu alana Modbus bellek haritası denir ve bu alanda her yazmacın bir adresi vardır. Hangi adresin hangi yazmaç tipine ait olduğu, bu haritada bildirilmek zorundadır. Efendi cihaz, bu bellek haritasına bakarak hangi verinin nerede saklandığını bilir ve yazmaç tipine göre uyumlu fonksiyon kodunu seçerek sorgulamasını yapar [8]. Örnek Modbus yazmaç yapısı Tablo 3.1’de gösterilmiştir.

Tablo 3.1: Modbus cihazı yazmaç yapısı.

Yazmaç (Register) Tipi	Adres Veri Boyutu	Erişim Şekli	Açıklama
Bobin	1 bit	Okuma/Yazma	Uygulamaya bağlı genel amaçlı kullanılabilir.
Ayrık Cihaz Girişi	1 bit	Sadece Okuma	Cihazın tek girişini okumak için kullanılır.
Cihaz Giriş Yazmacı	2 Bayt	Sadece Okuma	Cihazın çoklu girişlerini okumak için kullanılır.
Tutucu Yazmaç	2 Bayt	Okuma/Yazma	Uygulamaya bağlı genel amaçlı kullanılabilir.

Efendi cihazdan gelen sorguyu köle cihazı değerlendirip bir problem yoksa bu isteğe cevap verir. İlgili fonksiyon koduna göre cevap içeriği paketin veri kısmında iletilir. Bütün değerler onaltılık (hexadecimal) olup, paket içinde en ağırlıklı kısım her zaman önce gönderilir. Genel kullanılan Modbus fonksiyon kodları Tablo 3.2’de gösterilmiştir [4].

Tablo 3.2: Genel Modbus haberleşme protokolü fonksiyon kodları.

Fonksiyon Kodu	Açıklama
01	Köle cihazdan tek bobin durumu okumak için kullanılır.
02	Köle cihazdan ayırık giriş durumu okumak için kullanılır.
03	Köle cihazdan tutucu yazmaç değerini okumak için kullanılır.
04	Köle cihazdan giriş yazmaç değerini okumak için kullanılır.
05	Köle cihazın tek bobinine veri yazmak için kullanılır.
06	Köle cihazın tutucu yazmacına veri yazmak için kullanılır.
15	Köle cihazın bir veya daha fazla bobinine veri yazmak için kullanılır.
16	Köle cihazın bir veya daha fazla tutucu yazmacına veri yazmak için kullanılır.

Örnek bir sorgulama, cevap ve hata paketi içeriği Şekil 3.5 ve Şekil 3.6'daki gibidir [4].

Sorgu		
Fonksiyon Kodu	1 Bayt	0x03
Başlangıç Adresi	2 Bayt	0x0000 to 0xFFFF
Yazmaç Sayısı	2 Bayt	1 - 125 (0x7D)
Cevap		
Fonksiyon Kodu	1 Bayt	0x03
Bayt sayısı	1 Bayt	2 x N*
Yazmaç Değeri	N* x 2 Bayt	
*N = Yazmaç Sayısı		
Hata Durumu		
Fonksiyon Hata Kodu	1 Byte	0x83
Hata Kodu	1 Byte	01 or 02 or 03 or 04

Şekil 3.5: (0x03) Köle cihazdan tutucu yazmaç değeri okuma sorgu yapısı.

Sorgu		Cevap	
Alan Adı	(Hex)	Alan Adı	(Hex)
Fonksiyon Kodu	03	Fonksiyon Kodu	03
Yazmaç Adres MSB	00	Bayt Sayısı	06
Yazmaç Adres LSB	6B	Yazmaç Değeri MSB (108)	02
Yazmaç Sayısı MSB	00	Yazmaç Değeri LSB (108)	2B
Yazmaç Sayısı LSB	03	Yazmaç Değeri MSB (109)	00
		Yazmaç Değeri LSB (109)	00
		Yazmaç Değeri MSB (110)	00
		Yazmaç Değeri LSB (110)	64

Şekil 3.6: 108-110 adreslerindeki yazmaçları okumak için örnek bir sorgu.

Modbus haberleşmesinde iletim esnasından ve hatalı isteklerden kaynaklanmak üzere iki çeşit hata kodu türü bulunur. Kullanılan hata kodları ve açıklamaları Tablo 3.3'te görülebilir.

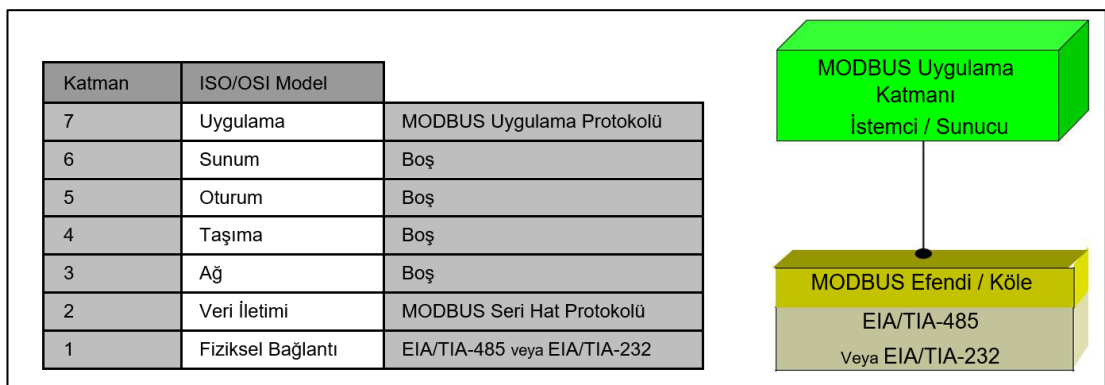
Tablo 3.3: Modbus Protokolü hata kodları.

Hata Kodu	Açıklama
01	Geçersiz fonksiyon kodu.
02	Geçersiz yazmaç adresi.
03	Geçersiz veri değeri.
04	Köle cihazı hatası.
05	Bilgilendirme mesajı (Paket alınamıyor).
06	Köle cihazı meşgul.
08	Bellek eşlik hatası.
0A	Ağ geçidine erişilemiyor.
0B	Ağ geçidi cevap vermiyor

Sonraki bölümlerde fiziksel katman olarak RS485/232 seri hat üzerinde çalışan Modbus RTU/ASCII protokollerinden ve Ethernet üzerinde çalışan Modbus TCP protokolünden bahsedilmiştir.

3.1. Modbus Seri Hat Protokolleri

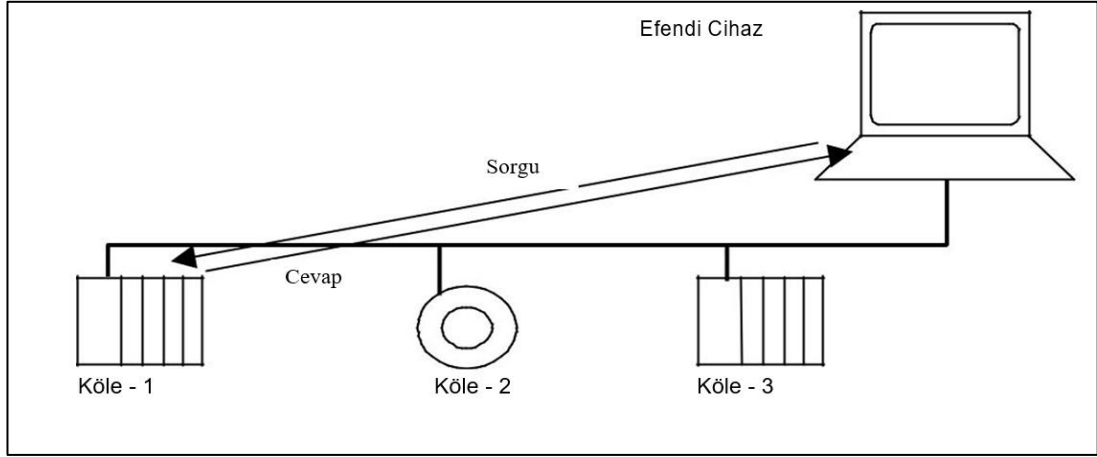
Modbus protokolünün seri hat üzerinde çalışan Modbus RTU ve ASCII olmak üzere iki türü vardır. Modbus RTU ve ASCII protokolü Şekil 3.7’de görülen OSI (Open System Interconnection) modelinin 2. katmanını olan veri katmanında çalışır. Genelde fiziksel yapı olarak bire bir haberleşme ve kısa mesafeler için RS-232, uzak mesafeler ve birden fazla cihazla haberleşmek için RS-485 kullanılır [5].



Şekil 3.7: Modbus Protokolü ve OSI Model katman yapısı.

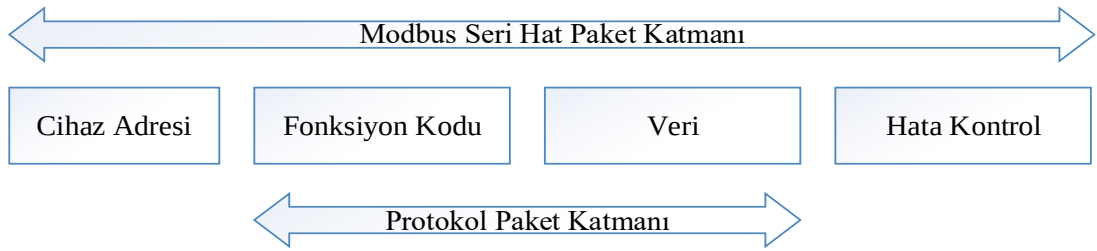
Bir seri hat üzerinde çalışan Modbus ağında bir tane efendi ve en fazla 247 tane köle cihazı bulunabilir. Ağdaki köleler 1 ile 247 arasında kendilerini tanımlayan bir

adres alır. Ağı doğrudan efendi cihazı kontrol eder ve efendi cihazı hariç hiçbir köle haberleşmeyi başlatamaz. Örnek bir Modbus ağı Şekil 3.8’de gösterilmiştir [5].



Şekil 3.8: Örnek Modbus ağındaki efendi-köle sorgusu.

Efendi cihazı ilgili köle cihaza mesaj yollamak ve köle cihazdan mesaj almak için bu adres aralığını kullanır. "0" adresini ise ağdaki bütün köle cihazlara mesaj yollamak için kullanır. Köle cihazlar bu adresi tanımak zorundadır fakat bu mesaja cevap vermezler. Efendi cihazı bu adresi bütün köle cihazlara veri yazmak için kullanır. Örnek Modbus paket yapısı Şekil 3.9’da gösterilmiştir.



Şekil 3.9: Modbus protokolünün seri hattaki paket yapısı.

$$\begin{aligned} \text{Modbus Seri Hat PDU Boyutu} &= 256 - \text{Köle Cihazı Adresi (1} \\ &\text{bayt)} - \text{Hata Kontrol (2 bayt)} = 253 \text{ bayt} \end{aligned} \quad (3.3)$$

Paket içeriğinde en başta ilgili köle cihaz adresi gönderilir. Ağdaki efendi cihazı sorgulama yaparken hangi bilgiyi istediğini belirtmek için fonksiyon kodlarını kullanır. Fonksiyon koduna göre köle cihazın vereceği cevap değişir. Fonksiyon kodları cihaz yazmaçlarından veri okumak ve yazmak için kullanılır.

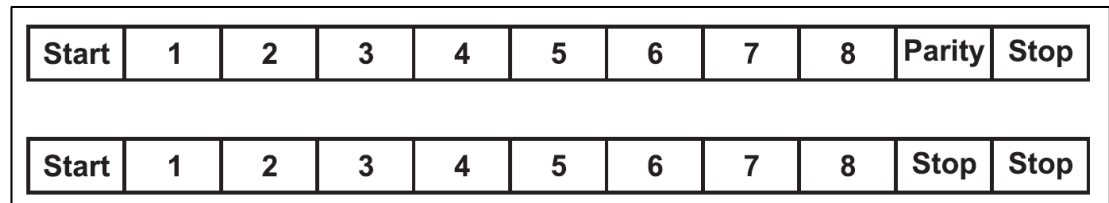
Paket iletimi sırasında bir hatanın oluşup oluşmadığı ise paket sonundaki 2 baytlık hata kontrol kodu ile saptanır. Eğer paket alımında herhangi bir hata oluşmuş ise köle cihaz bu hatayı koduyla beraber efendi cihaza iletir. Hata kontrol kodu hesaplaması Modbus RTU veya ASCII protokolüne göre farklı yapılır.

3.1.1. Modbus RTU Protokolü

Modbus RTU protokolünde veriler her bayt 8 bit olacak şekilde onaltılık (hexadecimal) karakter olarak yollanılır. Tablo 3.4'te seri hat üzerindeki bir baytlık bir karakterin içeriği, Şekil 3.10'da ise eşlik bitine göre bir baytlık paket yapısı görülebilir.

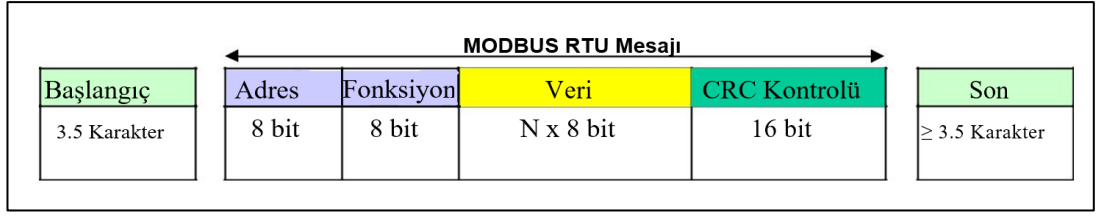
Tablo 3.4: Seri hat üzerindeki bir bayt için Modbus RTU karakter formatı.

Kodlama	8 - Bit
Bayt Başına Bitler	1 Başlangıç Biti (Start Bit)
	8 Veri Biti (Data Bit)
	1 Eşitlik Biti (Parity Bit)
	1 Durma Biti (Stop Bit)
Toplam	11 Bit



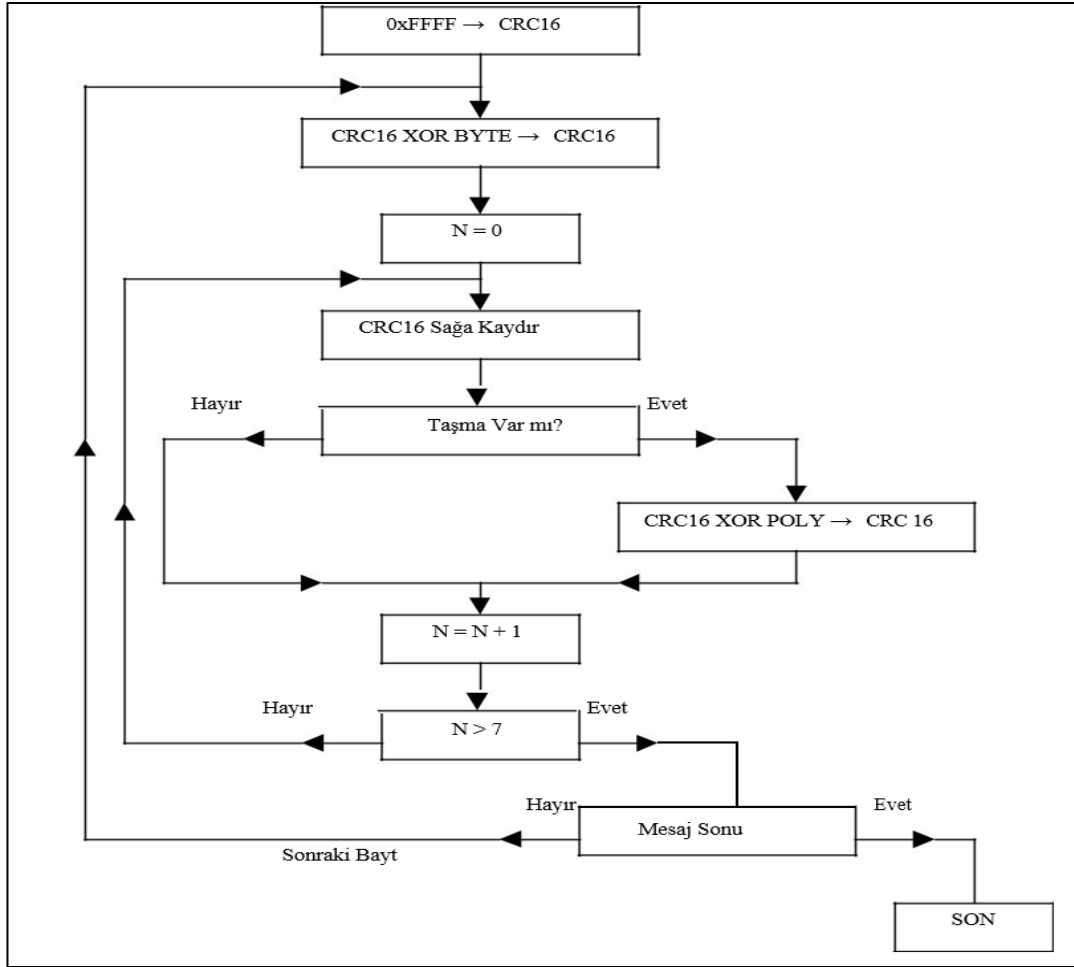
Şekil 3.10: Eşlik bitine göre bir bayt Modbus RTU karakter yapısı

Şekil 3.11'de Modbus RTU mesaj yapısı görülebilir. Bir Modbus RTU mesajı ilgili cihaz adresi, fonksiyon kodu, veri kısmı ve hata kontrol olarak CRC'den oluşur [5]. Mesajlar arası bir zaman aralığı vardır ve bu zaman aralığı 3,5 karakter zaman aralığı olarak adlandırılır. Bir karakter bir bayt olarak değerlendirilir ve bir karakterlik süre bir baytın seri hattaki iletim süresi olarak hesaplanır. Detaylı zaman hesaplaması yöntemle beraber ilgili başlıkta anlatılmıştır.



Şekil 3.11: Detaylı Modbus RTU paket yapısı.

Modbus RTU CRC hesaplama algoritması Şekil 3.12’de gösterilmiştir [5].



Şekil 3.12: Modbus RTU CRC-16 hesaplama algoritması.

Algoritmadaki ilgili işlemlerin açıklaması aşağıdaki gibidir;

- XOR = Dışlayıcı veya
- N = Bit Sayısı
- POLY = CRC 16 Hesaplama Polinomu = 1010 0000 0000 0001
(Polinom Kodu = $1 + x^2 + x^{15} + x^{16}$)

Modbus RTU protokolü standartlarındaki CRC hesaplama kodu aşağıdaki gibidir [5]. Gelen paketin bittiği anlaşıldıktan sonra verilen tablodaki CRC değerleri kullanılarak gelen mesaj işleme sokulur. İşlem sonunda 2 baytlık Modbus RTU CRC kodu elde edilir. Bu değer gelen mesaj sonundaki CRC değeri ile aynı ise mesaj iletim başarılı kabul edilir, aksi takdirde ilgili hata kodu alıcıya yollanır.

Tablo 3.5: Modbus CRC-16 hesaplama kodu.

```

/* CRC Değer Tablosu MSB */
static unsigned char modbus_auchCRCHi[] = {
0x00, 0xC1, ...
};

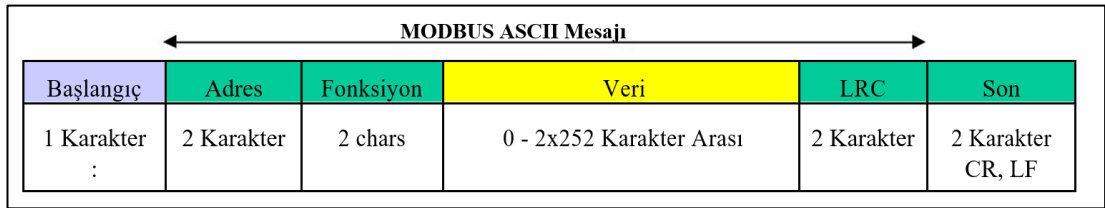
/* CRC Değer Tablosu LSB */
static char modbus_auchCRCLo[] = {
0x00, 0xC0, ...
};

unsigned short modbus_CRC16 ( /* Hesaplanan CRC değeri unsigned short olarak döner */
unsigned char *incomingData , /* Hesaplanacak Mesaj */
unsigned short dataLen ) /* Mesajdaki bayt sayısı */
{
    unsigned char modCRCHi = 0xFF ; /*Hesaplanacak CRC MSB ilk değer */
    unsigned char modCRCLo = 0xFF ; /* Hesaplanacak CRC LSB ilk değer */
    unsigned modCRCIndex ; /* CRC Değer Tablosu indeksi */
    while (dataLen --) /* Tüm mesaj bitene kadar işleme devam et */
    {
        modCRCIndex = modCRCLo ^ * incomingData ++ ; /* CRC hesapla */
        modCRCLo = modCRCHi ^ modbus_auchCRCHi[uIndex] ;
        modCRCHi = modbus_auchCRCLo[uIndex] ;
    }
    return (modCRCHi << 8 | modCRCLo) ; /* Hesaplanan CRC'yi geri dön */
}

```

3.1.2. Modbus ASCII Protokolü

Modbus ASCII protokolü verilerin ASCII kodlama sistemine göre kodlanarak yollanmasına dayanır. Her karakter bu şekilde 2 bayt olacak şekilde yollanır. Bu yüzden Modbus RTU protokolünde yollanan bir mesaj boyutu Modbus ASCII protokolünde iki katı kadar daha büyüktür ve bu da Modbus ASCII protokolü iletim hızının, Modbus RTU protokolüne göre daha yavaş olmasına yol açmaktadır. Modbus ASCII protokolünde verilerin arka arkaya yollanması, hataya neden olmadan verilerin iletilmesine olanak sağlar. Her mesaj Şekil 3.13'teki gibi ":" (0x3A) karakteri ile başlar, satır başı (CR - Carriage Return, 0x0D) ve yeni bir satır (LF - Line Feed, 0x0A) karakterleri ile sonlandırılır. Bu şekilde baytlar arası zaman gecikmesine imkan tanıdığı için bazı modemler ile veri aktarımını da mümkün kılar [5]. Fakat protokol kararlılığı için zaman aşımı en fazla 1 saniye olacak şekilde tanımlanmıştır.

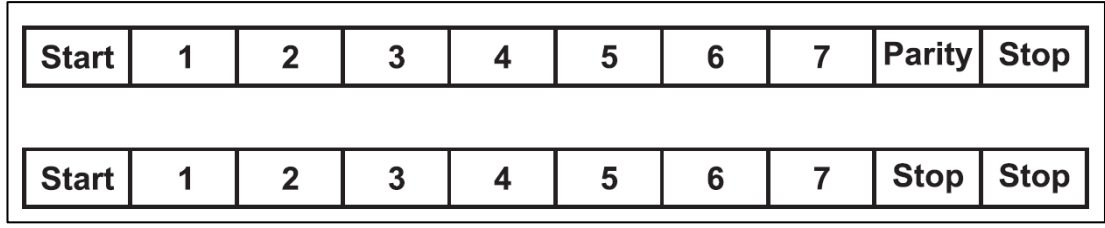


Şekil 3.13: Detaylı Modbus ASCII paket yapısı.

Modbus ASCII protokolünde veri bit sayısı 7 olacak şekilde ayarlanmıştır. Bir karakterin seri hat üzerinde 10 bit olarak yollanması için kalan bitler eşlik ve durdurma biti ile doldurulmuştur. Tablo 3.6'te seri hat üzerindeki bir baytlık bir karakterin içeriği, Şekil 3.14'te ise eşlik bitine göre bir baytlık paket yapısı görülebilir.

Tablo 3.6: Seri Hat üzerindeki bir bayt için Modbus ASCII karakter formatı.

Kodlama	Onaltılık – ASCII karakter (0-9, A-F)
Bayt Başına Bitler	1 Başlangıç Biti (Start Bit)
	7 Veri Biti (Data Bit)
	1 Eşitlik Biti (Parity Bit)
	1 Durma Biti (Stop Bit)
Toplam	10 Bit



Şekil 3.14: Eşlik bitine göre bir bayt Modbus ASCII karakter yapısı.

Modbus ASCII protokolünde hata kontrolü olarak Boylamsal Artıklık Kontrolü (LRC - Longitudinal Redundancy Check) kullanılır. LRC iki karakterden oluşmaktadır ve her gelen paket için özel hesaplanır. Paket iletiminin bittiği paket sonundaki “CRLF” karakterlerinin alınmasıyla anlaşılır. Gelen paketteki LRC hesaplanarak, paket içindeki LRC karşılaştırılır. Eğer LRC değerleri aynı ise paket iletimi başarılı bir şekilde tamamlanmış demektir. Aksi taktirde paket hatalı sayılıp ilgili hata kodu alıcıya yollanır. LRC hesaplaması CRC hesaplamasına göre daha basit olup aşağıdaki adımları izleyerek hesaplanır [5];

- Paket içerisindeki her bayt, paket sonundaki LRC değerine kadar art arda toplanır ve taşma olursa görmezden gelinir.
- Çıkan sonucun iki tümleyeni alınarak hesaplanan son LRC değeri bulunur.

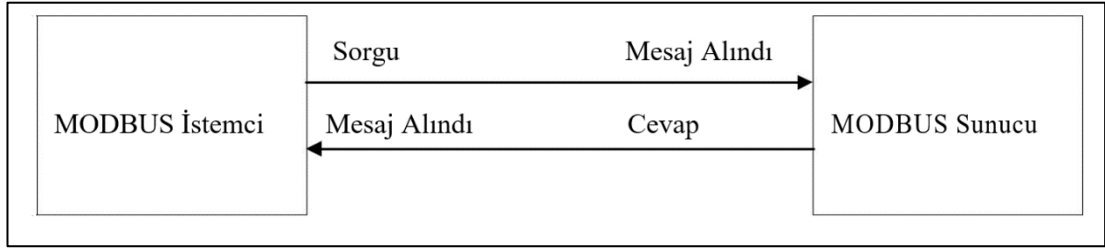
Modbus ASCII protokolü standartlarındaki LRC hesaplama kodu ise Tablo 3.7’deki gibidir;

Tablo 3.7: Modbus ASCII LRC hesaplama kodu.

```
unsigned char LRC ( /* Hesaplanan LRC değeri unsigned char olarak döner */
unsigned char *auchMsg , /* LRC hesaplanacak mesaj */
unsigned short usDataLen ) /* Mesajdaki bayt sayısı */
{
    unsigned char uchLRC = 0 ; /* Hesaplanacak LRC ilk değer atama */
    while (usDataLen—) /* Tüm mesaj bitene kadar işleme devam et */
        uchLRC += *auchMsg++ ; /* Sıradaki baytı ekle */
    return ((unsigned char)(—((char)uchLRC))) ; /* Sonucu iki tümleyeni olarak geri döndür */
}
```

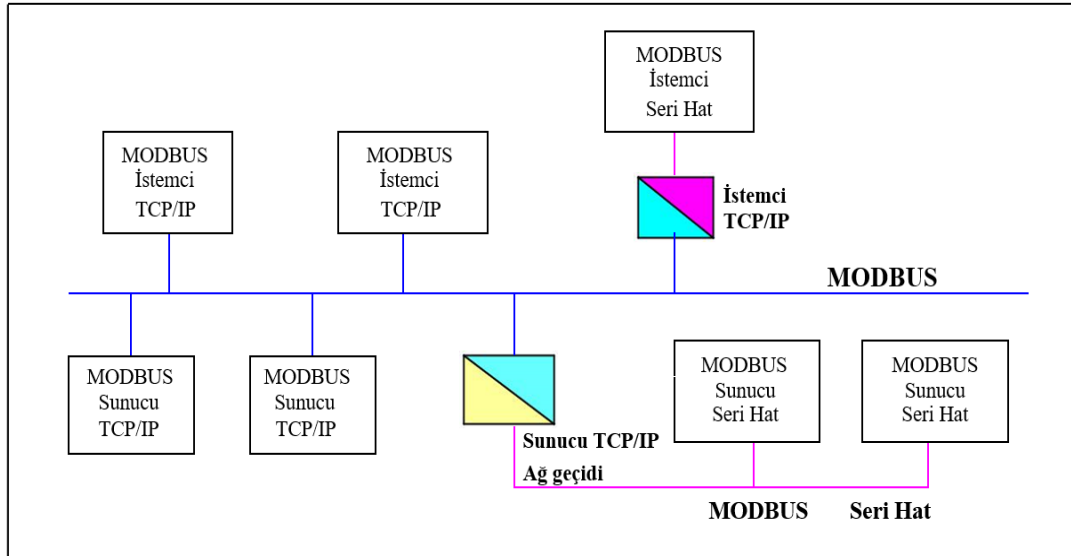
3.2. Modbus TCP/IP Protokolü

Modbus protokolü ilk olarak seri hat üzerinde çalışması için tasarlanmıştır. Teknolojinin gelişmesi ile birlikte Ethernet üzerinde çalışabilen TCP versiyonu ortaya çıkmıştır. Şekil 3.15'teki gibi Modbus TCP protokolünde de Modbus RTU/ASCII protokolündeki gibi efendi/köle cihazı vardır ve bu haberleşme yapısı istemci/sunucu olarak adlandırılır.



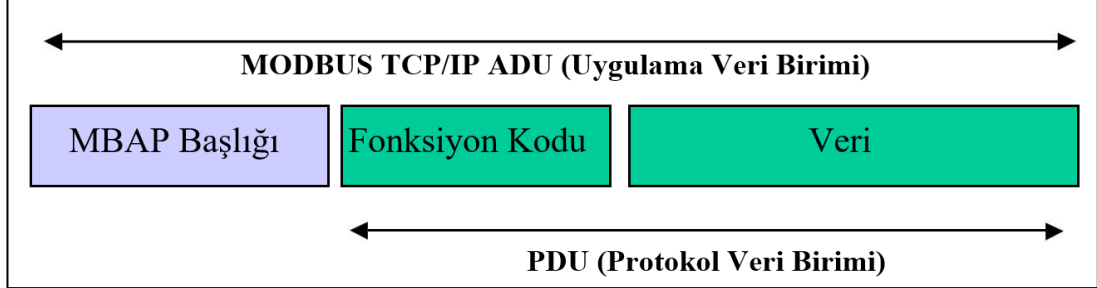
Şekil 3.15: Modbus TCP/IP protokolü haberleşme yapısı.

Modbus TCP protokolü Şekil 3.16'daki gibi Ethernet üzerinde çalıştığı için seri hatlar üzerindeki çalışan Modbus RTU ve ASCII protokollerine göre daha hızlıdır [6]. Bu protokolün Modbus RTU ve ASCII protokollerinden farkı, verilerin TCP/IP taşıma katmanını içinde yollanmasıdır.



Şekil 3.16: Örnek Modbus TCP/IP protokolü haberleşme ağı.

Modbus TCP PDU katmanı Şekil 3.17'deki gibidir. Modbus RTU/ASCII protokolünden farklı olarak hata kontrolü ve adres birimi TCP/IP protokolünün sorumluluğunda olduğundan dolayı PDU içerisinde yer almaz [6].



Şekil 3.17: Modbus TCP protokolü paket yapısı.

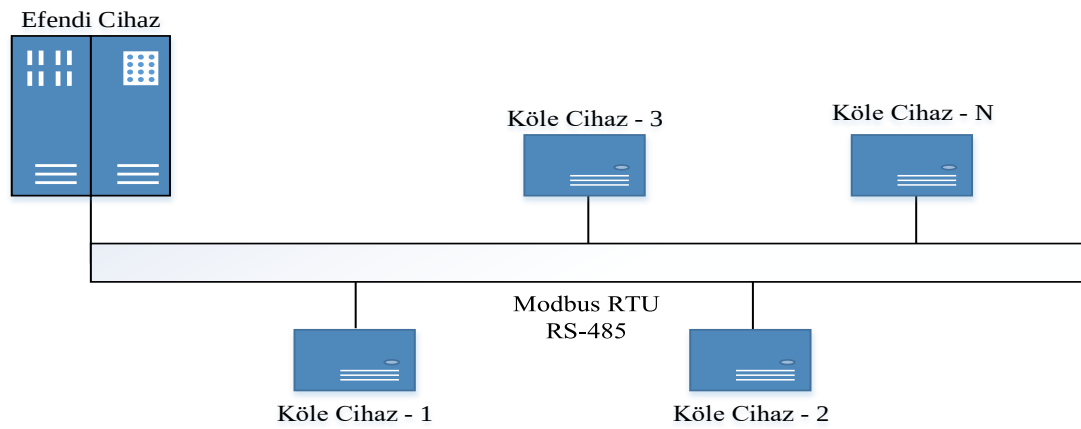
MBAP başlığı Modbus TCP/IP uygulama veri biriminin bir parçası olup Tablo 3.8'deki gibi bir içeriğe sahiptir. Bu başlık bilgisinin boyutu 7 bayttır. TCP/IP protokolü sayesinde Modbus RTU/ASCII protokolüne göre tek seferde daha büyük veriler taşınabilmektedir. Belli bir boyutun üstündeki verilerin parçalanması gerektiği için bu veriler parçalı şekilde yollar. İlgili verilerin hangi Modbus sorgu veya cevabına ait olduğunun anlaşılması için MBAP başlık bilgisi kullanılır. Bu şekilde Modbus TCP haberleşmesi hatasız bir şekilde gerçekleşmiş olur [6].

Tablo 3.8: Modbus TCP protokolü MBAP başlık yapısı.

Alanlar	Boyut	Açıklama	İstemci	Sunucu
İşlem Tanımlayıcı	2 Bayt	İlgili Modbus sorgu/cevap isteğinin bilgisi	İstemci tarafından başlatılır	Sunucu tarafından alınan istekten kopyalanır
Protokol Tanımlayıcı	2 Bayt	0 = MODBUS protokolü	İstemci tarafından belirlenir	Sunucu tarafından alınan istekten kopyalanır
Uzunluk	2 Bayt	Paketteki bayt sayısı	Sorgu olarak istemci tarafından belirlenir	Cevap olarak sunucu tarafından belirlenir
Birim tanımlayıcı	1 Bayt	Farklı ağlardaki cihazları haberleştirmek için kullanılır	İstemci tarafından belirlenir	Sunucu tarafından alınan istekten kopyalanır

4. YÖNTEM

Yöntem Şekil 4.1’deki gibi bir Modbus RTU ağında uygulanmış olup temel olarak sonraki başlıklarda bahsedilecek olan parça karakter CRC hesaplama ve zaman aşımı atlatma yaklaşımından oluşmaktadır. Yöntemin anlaşılması için öncelikle parça karakter CRC hesaplamasının ne olduğundan ve zaman aşımı atlatmanın ne amaçla kullanılacağından bahsedilecektir. Daha sonra bu iki yaklaşımın birleşimiyle oluşan yöntemden bahsedilecektir.



Şekil 4.1: Örnek Modbus RTU protokolü ağı.

4.1. Parça Karakter CRC Hesaplama Yaklaşımı

Modbus RTU protokolünde paket iletimi tamamlandıktan sonra CRC kontrolü yapılarak paketin doğrulaması yapılır. Paket doğrulandıktan sonra istenilen cevap hazırlanıp tekrar aynı kurallar ile karşı tarafa yollanır. Standarttaki Modbus CRC hesaplamasında polinom kodu olarak $1 + x^2 + x^{15} + x^{16}$ kullanılır ve hesaplama işlemi aşağıdaki gibi yapılır [5].

- (1) 16 bitlik CRC değişkenine “0xFFFF” (onaltılık) ilk değeri yüklenir.
- (2) Mesajın ilk baytını, 16 bitlik CRC değişkeninin en az ağırlıklı baytı ile XOR (dışlayıcı veya) işlemine sokulur. Sonucu CRC değişkenine yazılır.

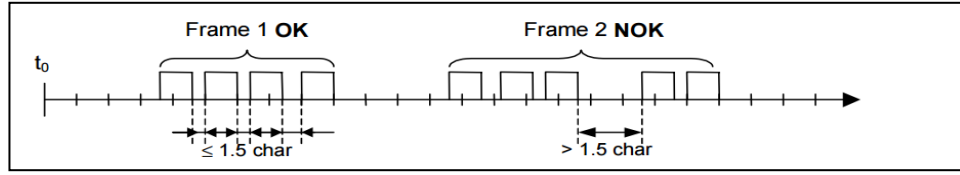
- (3) CRC değişkeni 1 bit sağa kaydırılırken (en az ağırlıklı bite doğru) en baştaki en çok ağırlıklı bit "0" ile doldurulur. Çıkan değer ilk değerden çıkartılır ve en az ağırlıklı bit incelenir.
- (4) En az ağırlıklı bit "0" ise 3. adım tekrarlanır değilse CRC değişkenini "0xA001" onaltılık değeriyle XOR (dışlayıcı veya) işleme sokulur.
- (5) 3. ve 4. adımlar 8 kere sağa kaydırma yapılana kadar tekrarlanır. Tekrarlama bittiğinde mesajın bir baytına karşılık gelen işlem tamamlanmış olur.
- (6) Aynı şekilde mesajın diğer baytları için de 2. ve 5. adımlar uygulanır. Bu işlem mesajın tüm baytları bitene kadar yapılır.
- (7) Bu işlemler sonunda CRC değişkeninin içeriği CRC değeri olacaktır.
- (8) CRC değeri mesajın sonuna eklenirken en az ve en çok ağırlıklı baytları yer değiştirilip eklenir.

Yukardaki yöntem kullanılarak paket iletimi bittikten sonra toplam CRC hesaplaması yapılır. Bu hesaplama paket boyutuna ve donanım işlemci gücüne bağlı olarak zaman almaktadır. Burada önerilen yaklaşım ise; en sonda toplam CRC hesaplama yapmak yerine, karakter iletimi sırasında yapılmasıdır. Her gelen karakter için toplam CRC hesaplanma başlatılır. Bir sonraki gelen karaktere kadar geçen süre ise bu şekilde ilgili karakter için CRC hesaplanmasında kullanılır. Paket iletimi bittiği anlaşıldıktan sonra da eldeki hesaplanan CRC değeri, paketin sonunda gelen CRC değerine eklenmesi ile beraber "0x0000" onaltılık değerine ulaşması beklenir. Eğer bu değere ulaşılmış ise paket doğru bir şekilde alınmış demektir. Aksi takdirde paket iletimi sırasında bir problem olmuş demektir. Önerilen yaklaşım ile paket boyutunun büyüklüğüne bağlı olarak paket kontrolü sırasında zaman kazancı sağlanması düşünülmüştür.

4.2. Zaman Aşımı Atlatma Yaklaşımı

Modbus RTU protokolünde, karşı tarafa paketler iletilirken karakterler halinde iletilir ve her karakterin boyutu bir bayt boyutundadır. Şekil 4.2’de gösterildiği gibi bir paket içerisindeki karakterler arası zaman gecikmesi en fazla 1,5 karakter boyutunda

olabilir ve bu gecikme $t_{1,5}$ olarak adlandırılır [5]. Bu süre içerisinde gönderilmeyen karakterler ilgili köle cihaz tarafından paketin değerlendirilmemesine neden olur.



Şekil 4.2: Modbus RTU protokolü karakterler arası zaman aralığı.

Paketler arası süre ise Şekil 4.3'teki gibi en az 3,5 karakter olmalıdır. Modbus standardında 19200 baud hızı veya daha yüksek hızlarda haberleşme yapılıyorsa $t_{1,5}$ karakter süresi 750 mikro saniye ve paket sonu gecikmesi $t_{3,5}$ 1,75 mili saniye olarak kabul edilir [5]. 19200 baud hızı altındaki zamanlamalar önemli olduğundan özel olarak hesaplanmalıdır [26]. Aşağıda örnek bir hesaplama yapılmıştır.

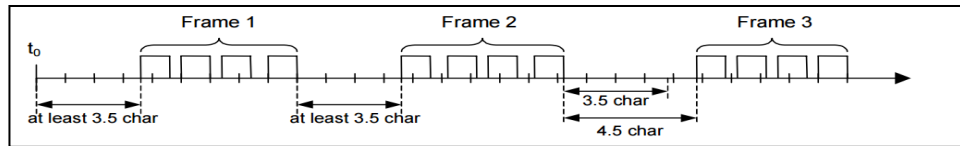
Seri hat haberleşme özellikleri 9600 Baud hızı, her veri 8 veri biti ve 2 bitişi biti olmak üzere toplam 10 bit olarak belirlenirse;

$$\text{Saniyedeki Karakter Sayısı} = \quad (4.1)$$

$$9600 \text{ baud hızı} / 10 \text{ bit} = 960 \text{ karakter/saniye}$$

$$t_{1,5} = 1,5 \times (1000/960) = 1,5625 \text{ mili saniye} \quad (4.2)$$

$$t_{3,5} = 3,5 \times (1000/960) = 3,6458 \text{ mili saniye} \quad (4.3)$$



Şekil 4.3: Modbus RTU protokolü paketler arası zaman aralığı.

Paketler arası zaman aşımı $t_{3,5}$ süresi aynı zamanda paket iletiminin bittiğinin anlaşılması için kullanılır. Bu süre bittikten sonra paket kontrolü yapılır. Eğer paket doğruysa işlenip karşı tarafa ilgili cevap yollanır, değilse karşı tarafa ilgili hata kodu

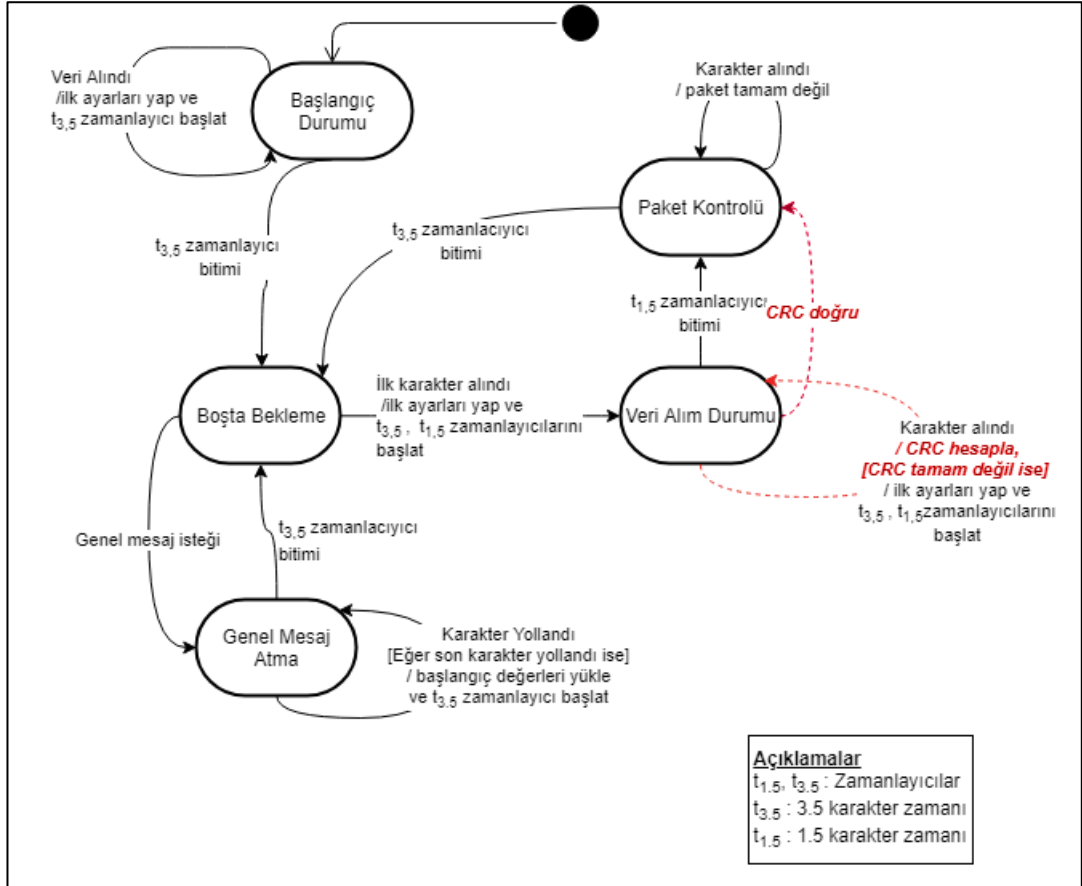
oluşturulup yollanır. Bu yüzden paket kontrolünün yapılması için bu zaman aşımının beklenmesi gerekir. Burada önerilen yaklaşımda ise $t_{3,5}$ zaman aşımı bitmeden, parça karakter CRC hesaplamasındaki yaklaşım kullanılarak paket sonu anlaşılmasına çalışılır. Önceki önerilen yaklaşımda olduğu gibi cihazın her karakter alımında Modbus RTU CRC hesaplanarak paket sonundaki CRC kısmının hesaba katılması ile beraber toplam CRC değeri “0x0000” onaltılık değerine ulaşılması beklenir. Bu değere ancak haberleşme paketinde bir hata yoksa ulaşılması beklenir ve paket iletiminin bittiği bu hesaplama ile mümkün hale gelir. İlgili paketin son karakteri geldiğinde, eğer iletim hatası yoksa CRC hesaplaması “0x0000” onaltılık değerine ulaşacağı için bir sonraki karakter için $t_{1,5}$ karakter zaman aşımını beklemeye gerek kalmaz. Bu şekilde paket kontrolünün daha erken yapılması mümkün hale gelir ve karşı taraf için cevap daha hızlı hazırlanır. Eğer paket iletimi sırasında bir hata oluşmuş ise CRC hesaplaması bu değere ulaşamayacak ve zaman aşımı oluşacaktır. Paket kontrolü yapılması esnasında CRC değerinin hatalı olduğu görülecek ve karşı tarafa ilgili hata kodu yine aynı şekilde iletilecektir. Kısaca önerilen yaklaşım ile paket iletimi sırasında bir hata yoksa $t_{1,5}$ karakter zaman aşımı kadar zaman kazanılması hedeflenmiştir.

4.3. Parça karakter CRC Hesaplamaya Bağlı Zaman Aşımı Atlatma Yöntemi

Bu çalışmada ortaya konan yöntem daha önceki başlıklarda bahsedilen iki yaklaşımın beraber kullanılmasını temel almaktadır. Bu iki yaklaşımı birleştiren yöntem, parça karakter CRC hesaplamaya bağlı zaman aşımı atlatma yöntemi olarak adlandırılmıştır.

Uygulanan yöntemin yeri kesikli ok işaretleri ile Şekil 4.4’teki Modbus RTU durum diyagramında gösterilmiştir. İlgili cihaz mesaj isteğini yaptıktan sonra karşı taraftan gelecek cevap mesajını beklediği “Veri Alım Durumu” aşamasına geçmektedir. Yöntem tam olarak bu aşamada uygulanmaktadır. Cihaz veri alım durumunda iken karşı taraftan gelen cevabı bir baytlık karakter olacak şekilde almaktadır. Cihaza gelen her karakter, parça karakter CRC hesaplaması yaklaşımında bahsedildiği gibi toplam CRC hesaplamasına katılır. Eğer toplam CRC hesaplaması “0x0000” onaltılık değerine ulaşırsa zamanlayıcılar durdurularak zaman aşımı atlatma yaklaşımında belirtildiği gibi bir sonraki aşama olan “Paket Kontrolü” aşamasına

geçilir. Bu aşamaya geçildikten sonra paket alımının bittiği anlaşılır ve paketin işlenmesi başlayabilir. Paket içindeki gelecek toplam bayt sayısı alınan bayt sayısının aynı olması da ikinci bir kontrol olarak haberleşmedeki oluşabilecek hataları ortadan kaldıracaktır.



Şekil 4.4: Uygulanan yöntemin Modbus RTU durum diyagramındaki yeri.

Durum diyagramında görüldüğü gibi iki farklı yaklaşımla haberleşme hızında iyileştirmeye katkı sağlanmaktadır. Uygulanan yöntem ile ilk olarak $t_{1,5}$ zamanlayıcı zaman aşımı bekleme süresinden kazanç sağlanmıştır. İkinci iyileştirme ise normal CRC hesaplamasını paket iletimi bittikten sonra yapmak yerine her karakter alındığında yapılması ile sağlanır. CRC hesaplamasını karakter alım sırasında yapmak sonraki karakter gelene kadar geçen sürenin daha verimli kullanılmasını sağlar. Diğer türlü tüm paket alındıktan sonra CRC hesaplaması yapmak ilgili cihazın cevap verme süresini geciktirecektir. Bu gecikme süresi gelen paketin büyüklüğü ve cihazın donanım gücüne bağlı olup değişkenlik gösterecektir. Oluşacak hatalar ise paket kontrolü aşamasında tekrar değerlendirilecektir. Paket kontrolü aşamasında herhangi

bir karakterin gelmesi ya da beklenmeyen bir durum olması yine Modbus RTU standartlarında belirtildiği şekilde cihazın yanıt vermesi ile olacaktır. Bir Modbus RTU haberleşmesinde en çok okuma sorguları kullanıldığı için yöntem ilk aşamada efendi cihaz tarafında ve sadece okuma sorgularının hızlandırılmasında kullanılmıştır. Bu şekilde okuma sorgularında gelen bayt sayısı da CRC kontrol işlemine dâhil edilip oluşabilecek beklenmedik durumlar ortadan kaldırılmıştır.

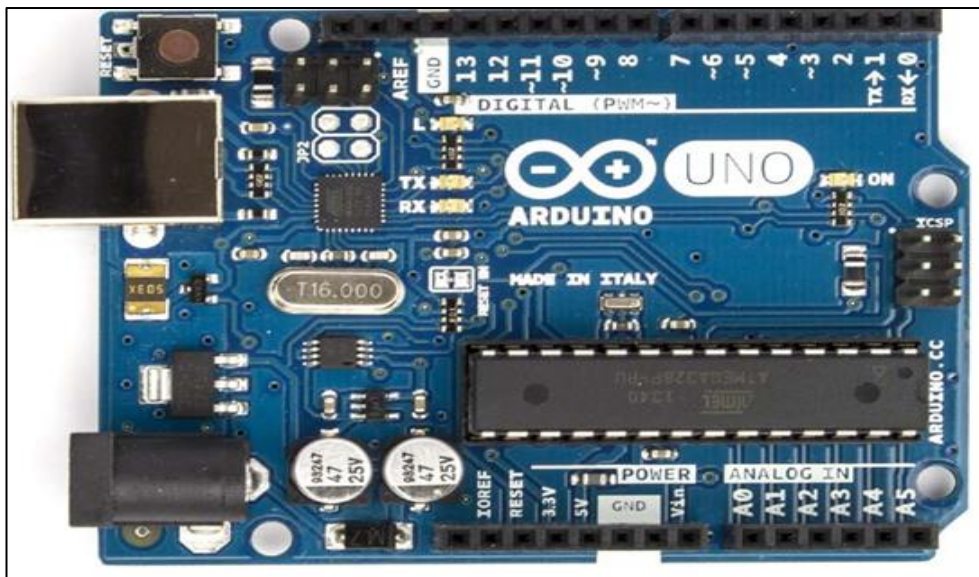
5. UYGULAMA VE YAZILIM

Yöntemin uygulanması için örnek bir Modbus RTU ağına ihtiyaç vardır. Bu ağ tasarlanırken hem donanım hem de yazılım tasarımı yapılması gerekmiştir. Yöntem, donanım tasarımına müdahale etmeden sadece yazılım seviyesinde uygulanmıştır. Donanım ve yazılım tasarımı yapıldıktan sonra ilgili testler, oluşturulan Modbus RTU ağı üzerinde uygulanıp yöntemin etkisi gösterilmiştir.

5.1. Donanım Tasarımı

Örnek bir Modbus RTU ağında bir efendi ve bir köle cihaz olmak üzere en az iki tane cihaza ihtiyaç vardır. Fakat, gerçekçi bir senaryoya yakın olması açısından ağda en az iki tane köle cihazı olması düşünülmüştür.

Modbus RTU yazılımının koşması için piyasada yaygın olan, giriş seviyesi uygulamalarının hızlıca geliştirilebileceği Arduino Uno devre kartı seçilmiştir. Arduino Uno ATmega328 mikrodenetleyici içeren bir devre kartıdır. Arduino 'nun en yaygın kullanılan kartıdır. Arduino Uno 'nun ilk modelinden sonra Arduino Uno R2, Arduino Uno SMD ve son olarak Arduino Uno R3 çıkmıştır [27]. Örnek bir Arduino Uno R3 Şekil 5.1’de gösterilmiştir.



Şekil 5.1: Arduino Uno R3 kartı.

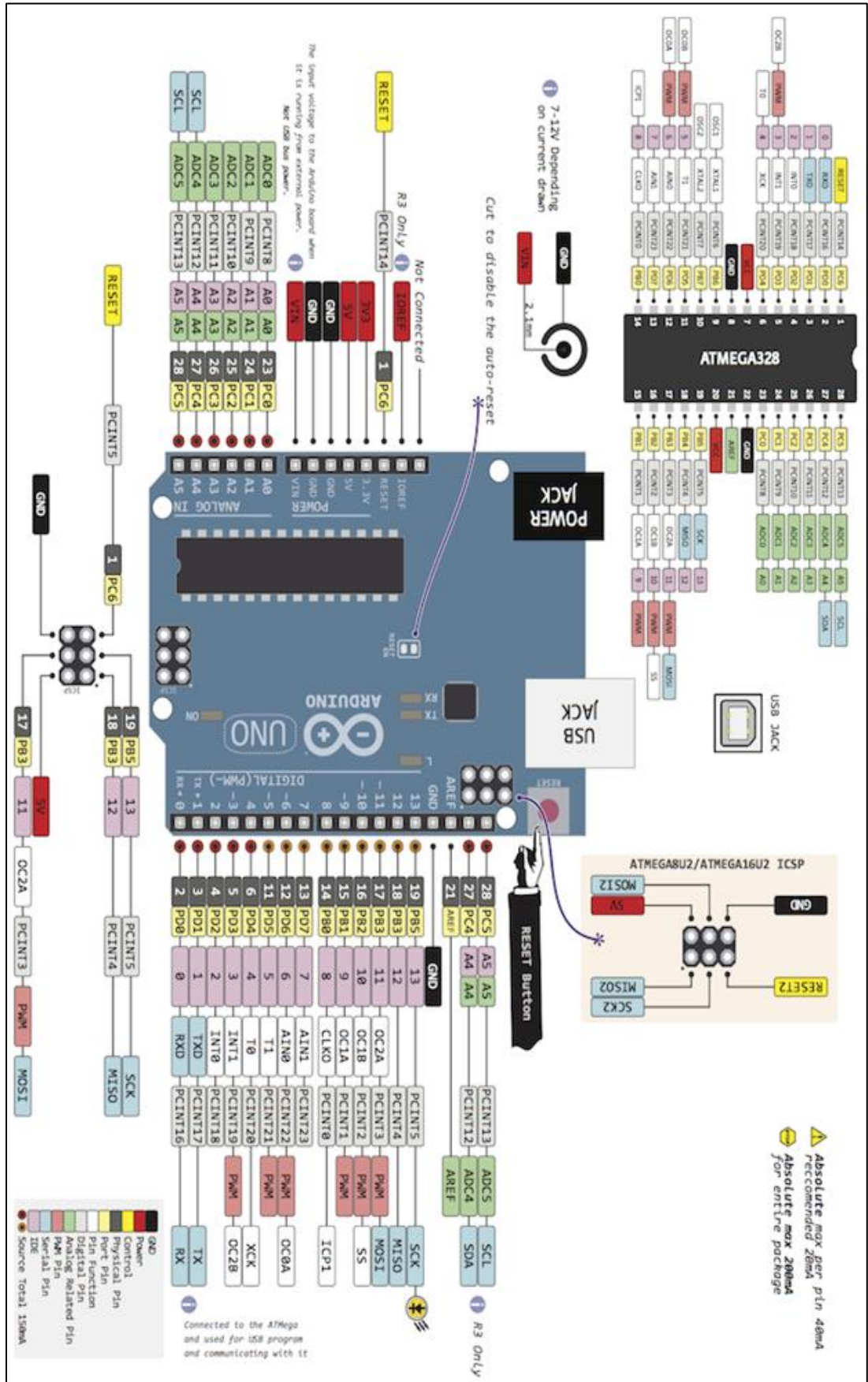
Arduino Uno devre kartının teknik özellikleri Tablo 5.1’de gösterilmiştir. Arduino Uno devre kartında R_X ve T_X mikrodenetleyici pinleri ile seri haberleşme mümkündür. Bu pinler sayesinde, fiziksel katman olarak RS485 kullanan Modbus RTU protokolü için veri alışverişi mümkündür. Atmega16u2 USB-seri dönüştürücü de bilgisayarda sanal bir seri port açarak Atmega328 ile bilgisayar arasında haberleşmeyi sağlar. Arduino Yazılım Geliştirme Ortamı (IDE - Integrated Development Environment) içerisinde yer alan seri monitör ile Arduino ile bilgisayar arasında metin temelli bilgilerin gönderilip alınmasını sağlar [28].

Tablo 5.1: Arduino Uno kartının teknik özellikleri.

Mikrodenetleyici	ATmega328
Çalışma Gerilimi	5V
Giriş Gerilimi (önerilen)	7-12V
Giriş Gerilimi (limit)	6-20V
Dijital G/Ç Pinleri	14 (6 tanesi PWM çıkışı)
Analog Giriş Pinleri	6
Her G/Ç için Akım	40 mA
3.3V Çıkış için Akım	50 mA
Flash Hafıza	32 KB (ATmega328)
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Saat Hızı	16 MHz
Uzunluk	68.6 mm
Genişlik	53.4 mm
Ağırlık	25 g

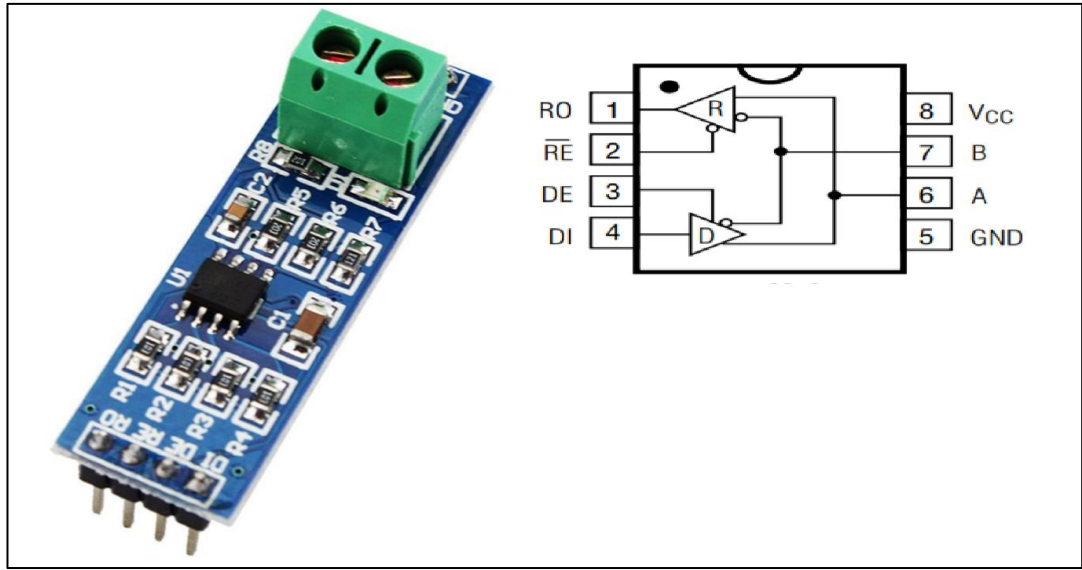
Arduino Uno, gücünü USB üzerinden veya adaptör girişinden alabilir. Harici güç kaynağı olarak 6-20V arası limit değerleri bulunmaktadır. Önerilen harici besleme gerilimi ise 7-12 V arasındır. Bunun sebebi 7V altındaki gerilimin istikrarlı çalışmayıp, 12V üzeri gerilimin de aşırı ısınma sebebi olabilmesidir. Kart üzerinde bulunan düzenleyici (regülatör) sayesinde 7-12V arası gerilim 5V’a düşürülür ve kart bu şekilde çalışır. Kart devre şeması Şekil 5.2’de görülebilir [29].

Arduino Uno devre kartı üzerinde 14 adet dijital, 6 adet analog giriş/çıkış pini bulunmaktadır. Bu pinlerin tamamının lojik seviyesi 5V'dur. Her pin maksimum 40 mA giriş ve çıkış akımı ile çalışır. Arduino Uno devre kartındaki tüm pinlerin listesi Şekil 5.3'te incelenebilir. Önemli olan pinlerden kısaca bahsetmek gerekirse; seri haberleşme için kullanılan mikrodnetleyici pinleri 0 ve 1 numaralı pinlerdir. TTL (Transistor Transistor Logic) seri veri alıp (RX), vermek (TX) için kullanılır. Bilgisayardan karta program yüklenirken veya bilgisayar ile UNO kartı arasında karşılıklı haberleşme yapılırken de bu pinlerden faydalanılır. Harici kesme için kullanılan mikrodnetleyici pinleri 2 ve 3 numaralı pinlerdir. Bu pinler yükselen kenar, düşen kenar veya değişiklik kesmesi pinleri olarak kullanılabilir. PWM için kullanılan mikrodnetleyici pinleri 3, 5, 6, 9, 10 ve 11 numaralı pinlerdir. Bu pinler 8-bit çözünürlükte PWM çıkış pinleridir. PWM çıkışları ile motor hız kontrolü, ı ışık parlaklık gibi analog kontrollerin yapılabilmesi sağlanır. SPI haberleşmesi için kullanılan mikrodnetleyici pinleri 10 (SS), 11 (MOSI), 12 (MISO) ve 13 (SCK) numaralı pinlerdir. Arduino Uno devre kartında üzerinde dahili bir LED bulunmaktadır. Bu LED mikrodnetleyicinin 13 numaralı pinene bağlıdır. Analog giriş için A0, A1, A2, A3, A4 ve A5 olmak üzere 6 adet 10-bit çözünürlüğünde mikrodnetleyicinin analog giriş pini bulunmaktadır. Bu pinler dijital giriş ve çıkış için de kullanılabilir. Pinlerin ölçüm aralığı 0-5V'tur. A4 veya SDA pini ve A5 veya SCL mikrodnetleyici pinleri I2C haberleşmesi için kullanılır. AREF pini analog girişler için ölçüm referansı pini olarak kullanılırken, Reset pini ise Arduino Uno devre kartının tüm modüllerini yeniden başlatmak için kullanılır [29].



Şekil 5.3: Arduino Uno devre kartındaki pinlerin konumu.

Modbus RTU haberleşme ağı için fiziksel katman olarak RS-485 seçilmiştir. Arduino Uno devre kartının RS-485 hattı üzerindeki gerilim seviyelerini ayarlayabilmesi için MAX485 entegresine ihtiyaç vardır. Bu ihtiyaç için MAX485 entegresi içeren Arduino MAX485 TTL/RS485 bağlantı dönüştürücü modülü kullanılmıştır. Şekil 5.4'te Arduino MAX485 TTL/RS485 bağlantı dönüştürücü modülü ve MAX485 entegresinin pinleri; Tablo 5.2'de ise teknik özellikleri görülebilir [30].



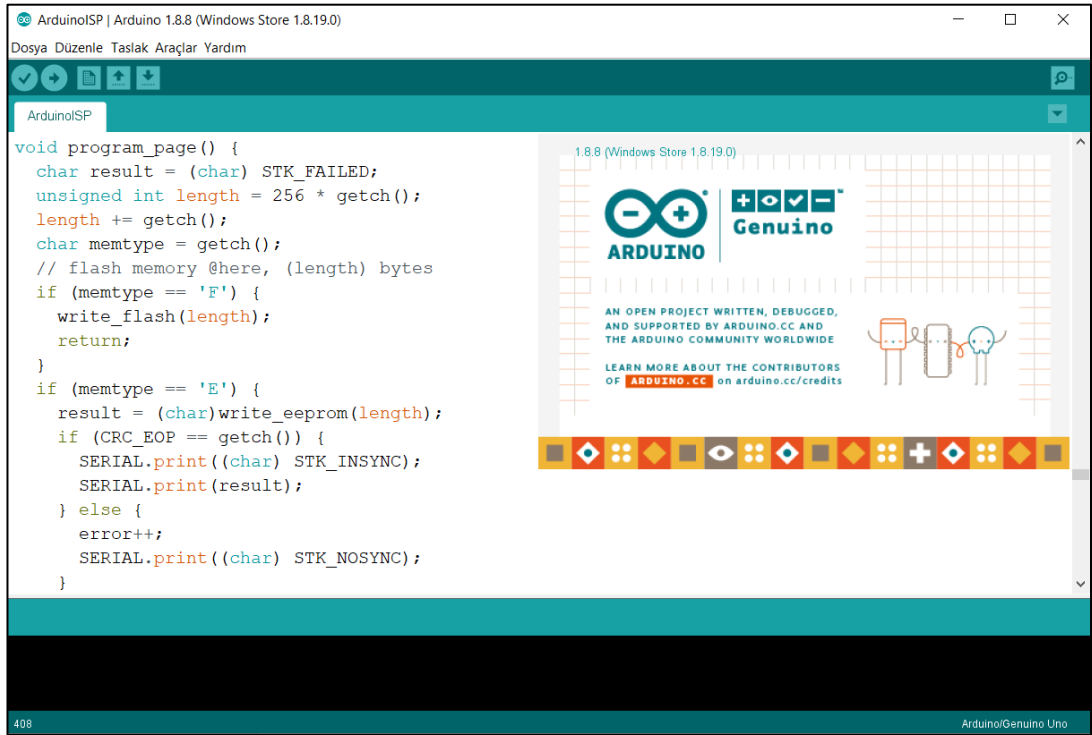
Şekil 5.4: Arduino MAX485 TTL/RS485 bağlantı dönüştürücü modülü ve MAX485 entegresi.

Tablo 5.2: MAX485 entegresinin pinlerinin açıklaması.

Pin No	Açıklama
1	Veri çıkış pini (Receive Output)
2	Veri çıkış etkinleştirme pini (Receive Output Enable)
3	Sürücü çıkış etkinleştirme pini (Driver Output Enable)
4	Sürücü giriş pini(Driver Input)
5	Toprak bağlantısı (Ground Signal)
6	Alıcı girişini sinyali ters çevirmeyen pin (Non-Inverting Receiver Input)
7	Alıcı giriş sinyali ters çeviren pin (Inverting Receiver Input)
8	Besleme bağlantısı (Power Supply)

5.2. Yazılım Tasarımı

Yöntemin uygulanacağı Modbus RTU ağı bir efendi cihazı ve iki tane köle cihazdan oluşmaktadır. Bir tane efendi cihaz, iki tane köle cihaz olmak üzere efendi ve köle cihaz gömülü sistem yazılımı geliştirilmiştir. Bu yazılım Arduino Uno devre kartı üzerinde koşacağı için IDE (Integrated Development Environment) olarak Şekil 5.5'te görülen Arduino yazılım geliştirme ortamı kullanılmıştır.



Şekil 5.5: Arduino kartı için yazılım geliştirme ortamı.

Yazılım geliştirmesi için açık kaynak olan bir Modbus kütüphanesinden faydalanılmıştır [31]. Programlama dili olarak mevcut Modbus kütüphanesine uyumluluk açısından “C” programlama dili seçilmiştir [32]. Modbus RTU haberleşmesi, Arduino Uno R3 kartının mevcut seri portlarını kullanılarak yapılmıştır.

Yöntem, efendi cihazı üzerinde uygulanmıştır ve efendi cihaz sorgusunu yaptıktan sonra gelen cevabın işlenmesi sırasında aktif olur. Yöntemi gerçeklemek için ilk olarak CRC hesaplamasını yapan “*modbus_calc_crc*” fonksiyonu standarttaki hesaplamadan farklı olarak, her gelen karakteri hesaplamaya katacak şekilde Tablo 5.3'teki gibi tasarlanmıştır.

Tablo 5.3: Parça karakter CRC hesaplama kodu.

```
/* CRC Değer Tablosu MSB */
static unsigned char modbus_rtuCRCHi [] = {
0x00, 0xC1, ...
};
/* CRC Değer Tablosu LSB */
static char modbus_rtuCRCLo [] = {
0x00, 0xC0, ...
};
void modbus_calc_crc(char incoming_data)
{
    byte dataIndex ;
    dataIndex = (modbus_rtu_crc16.b[1]) ^ incoming_data;
    modbus_rtu_crc16.b[1] = (modbus_rtu_crc16.b[0]) ^ modbus_rtuCRCHi[uIndex];
    modbus_rtu_crc16.b[0] = modbus_rtuCRCLo[uIndex];
}
```

Efendi cihaz gelen cevabı işlediği yerde de “*#ifdef BOOST_ON*” ön işlemci komutu kullanıldı. Bu ön işlemci komutu yöntemi aktifleştirerek istenen kod parçasının çalışmasını sağladı. Seri porttan okunan her karakter için “*modbus_calc_crc*” fonksiyonu çağrılarak, toplam CRC hesaplaması parça parça yapıldı. Toplam CRC değerini tutan “*modbus_serial_crc.d*” değişkeni “0x0000” değerine ulaşınca program akışı veri alma durumundan paket kontrolüne atlatıldı. Bu şekilde bir sonraki karakteri bekleme süresi olan $t_{1,5}$ zaman aşımının atlatılması sağlandı. İlgili kısımlar, kod parçasında kalın harflerle gösterilmiştir.

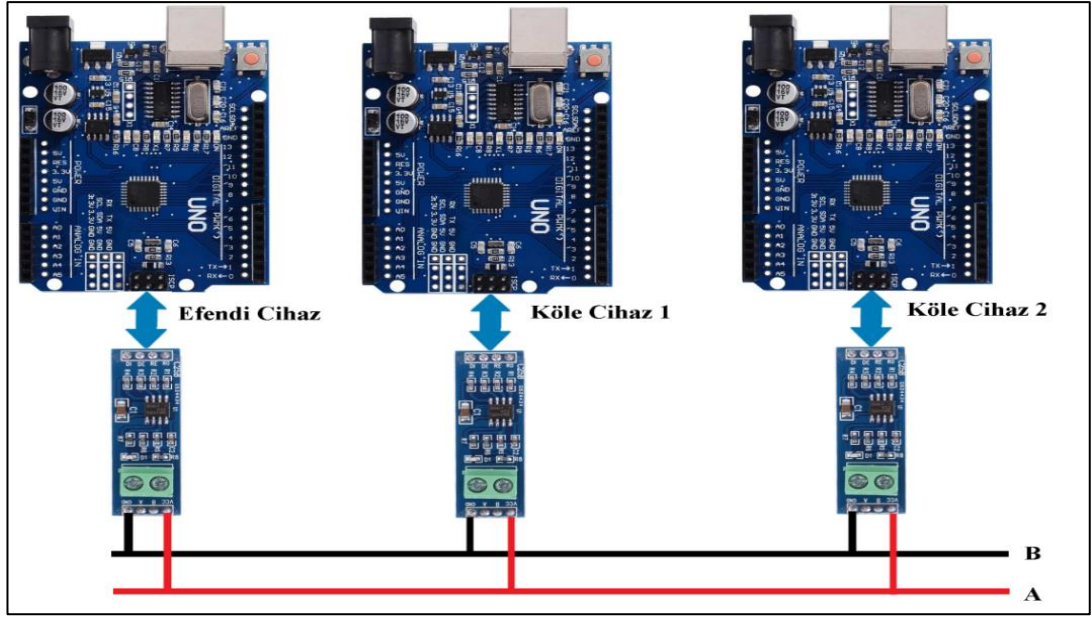
Tablo 5.4: Yöntemin uygulandığı kod parçası.

```
...
//Veri alma kısmı
while ((*ModbusPort).available())
{
    ...
    if (overflow)
    {
        (*ModbusPort).read();
    }

    else
    {
        if (buffer == BUFFER_SIZE)
        {
            overflow = true;
        }
        frame[buffer] = (*ModbusPort).read();
        //Gelen karakter için CRC hesaplaması yap
        #ifdef BOOST_ON
            modbus_calc_crc(frame[buffer]);
        #endif
        buffer++;
    }
    //Eğer CRC hesaplaması "0x0000" değerine ulaşmışsa, paket iletimi bitmiş demektir.
    #ifdef BOOST_ON
        if(modbus_serial_crc.d == 0x0000)
        {
            end_transmission = true;
            break;
        }
    #endif
    ...
}
...
```

5.3. Devre, Yapılan Testler ve Sonuçlar

Yöntemi gerçeklemek için Şekil 5.6’da gösterilen temsili test düzeneği 3 adet Arduino Uno devre kartı ve RS485-TTL seri dönüştürücü kartı ile kurulmuştur. Ardunio devre kartlarından biri efendi cihazı, diğer ikisi köle cihazı olarak tanımlanıp programlanmıştır.



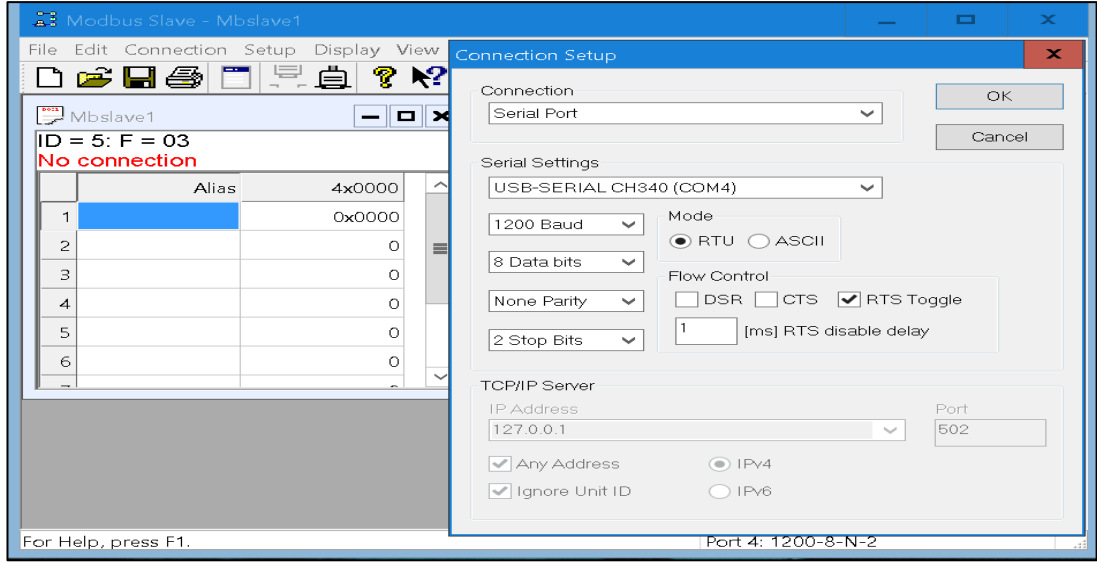
Şekil 5.6. Temsili Modbus RTU ağı test düzeneği.

Testlerde kullanılan örnek başarılı bir efendi-köle cihaz haberleşmesi Tablo 5.5’te görülebilir. Örnek haberleşmede efendi cihaz köle cihazdan “0x03” fonksiyon kodu ile verilen adresteki tutucu yazmaçtaki değeri sorgulamaktadır.

Tablo 5.5: Örnek detaylı Modbus RTU efendi-köle haberleşmesi.

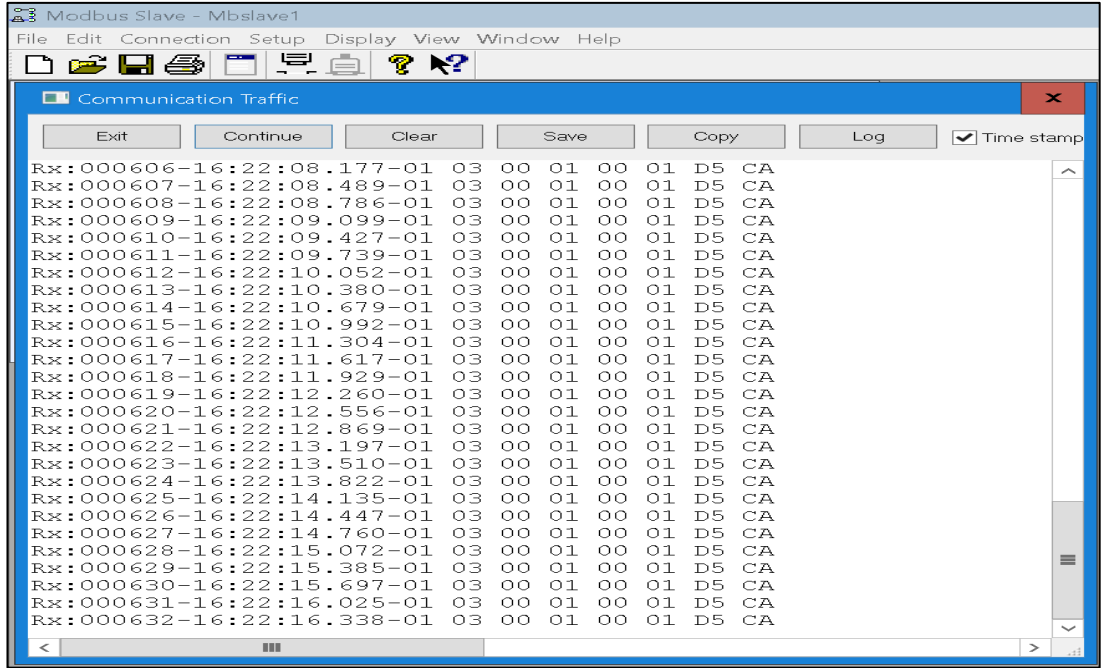
Efendi Cihazı Sorgu		Köle Cihazı - 1 Cevap	
Alan Adı Açıklama	Değer	Alan Adı Açıklama	Değer
Köle Cihaz Adresi	0x01	Köle Cihaz Adresi	0x01
Fonksiyon Kodu	0x03	Fonksiyon Kodu	0x03
Yazmaç Adresi En Soldaki Basamak	0x00	Yollanan Bayt Sayısı	0x02
Yazmaç Adresi En Sağdaki Basamak	0x01	Yazmaç Değeri En Sağdaki Basamak	0xBB
Yazmaç Sayısı En Soldaki Basamak	0x00	Yazmaç Değeri En Soldaki Basamak	0xAA
Yazmaç Sayısı En Sağdaki Basamak	0x01	CRC Kodu En Soldaki Basamak	0x4A
CRC Kodu En Soldaki Basamak	0xD5	CRC Kodu En Sağdaki Basamak	0xCB
CRC Kodu En Sağdaki Basamak	0xCA		

Modbus RTU haberleşmesini takip edebilmek için Şekil 5.7’deki Modbus Poll programı kullanılıp seri haberleşme için gerekli ayarlar yapılmıştır [33].



Şekil 5.7: Modbus Poll programı ve Modbus RTU protokolü haberleşme ayarı.

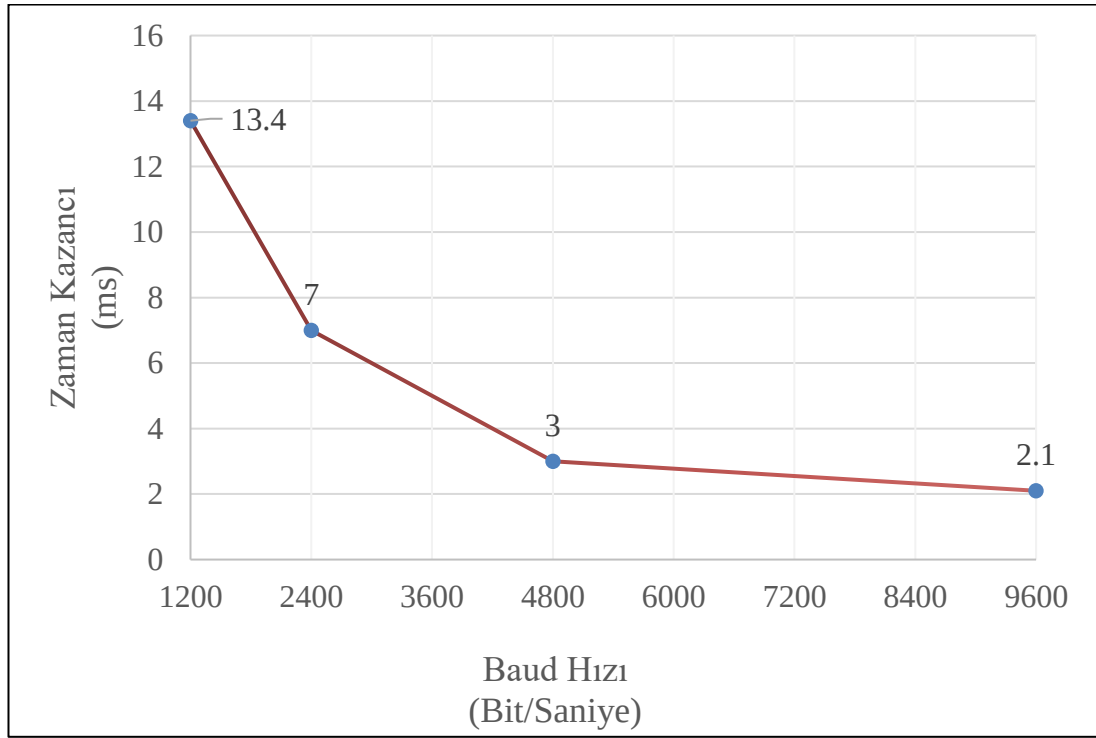
Efendi cihazın veri alma kısmı Şekil 5.8'deki gibi Modbus Poll bilgisayar programı yardımıyla dinlenip gelen paketlerin zamanlama bilgisi tutulmuştur.



Şekil 5.8: Modbus Poll programı üzerinden Modbus RTU haberleşme trafiği izleme.

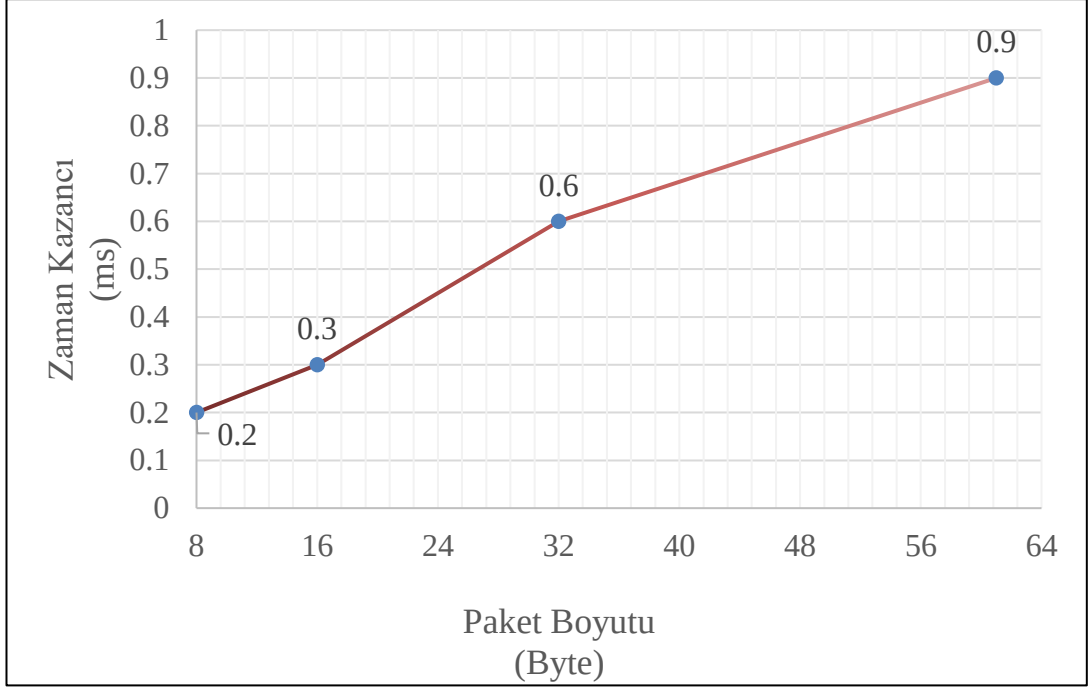
Testler Modbus RTU protokolünün desteklediği 1200, 2400, 4800 ve 9600 baud hızları için yapılmıştır. Her baud hızı için 8 baytlık ve 61 baytlık olmak üzere iki farklı paket boyutu kullanılmıştır. Bu şekilde uygulanan yöntemin hem baud hızına göre hem

de paket boyutuna göre kazancı hesaplanmıştır. Mevcut donanım kısıtlamasından dolayı bir paket boyutu en fazla 61 bayt ve hesaplamaların hassaslığı mili saniye olacak şekilde belirlenmiştir. Yöntem $t_{1,5}$ zaman aşımı atlatma yaklaşımı, parça karakter Modbus RTU CRC hesaplama yaklaşımı ve iki yaklaşım birlikte olacak şekilde üç aşamada uygulanmıştır. $T_{1,5}$ zaman aşımı atlatma yaklaşımının etkisi ise Şekil 5.9'da gösterilmiştir. İlgili şekilde görüldüğü gibi paket boyutundan bağımsız olarak baud hızı 1200 bit/saniye'de normal haberleşmeye göre ortalama zaman kazancı 13,4 ms olurken baud hızı 9600 bit/saniye'ye çıkınca ortalama zaman kazancı 2,1 ms olmuştur. Test sonuçlarına göre bu yaklaşım için baud hızının artması, normal haberleşmeye göre elde edilen zaman kazancının düştüğünü göstermiştir.



Şekil 5.9: Ortalama $t_{1,5}$ karakter zaman aşımı kazancı.

Parça karakter Modbus RTU CRC hesaplama yaklaşımının etkisi Şekil 5.10'da gösterilmiştir. Şekilde görüldüğü gibi normal haberleşmeye göre ortalama zaman kazancı baud rate hızından bağımsız paket boyutu arttıkça artmaktadır. Paket boyutu 8 bayt iken ortalama zaman kazancı 0,2 ms iken paket boyutu 61 bayt olunca ortalama zaman kazancı 0,9 ms olmuştur. Bu kazanç aynı zamanda kullanılan donanıma bağlı olduğu için farklı donanımlara göre değişkenlik gösterecektir.



Şekil 5.10: Ortalama parça karakter CRC hesaplama kazancı.

Son olarak Tablo 5.6’te mevcut test düzeneği ile yöntemin normal haberleşmeye göre bir paket seviyesindeki toplam zaman kazanç sonuçları gösterilmiştir. Toplam zaman kazancına ek olarak karşılaştırma amaçlı, yaklaşımların zaman kazançları da ayrı ayrı tabloya eklemiştir.

Tablo 5.6: Uygulanan yöntemin, yaklaşıma bağlı paket başına zaman kazançları.

Baud Hızı (bit/saniye)	1200		2400		4800		9600	
Paket Boyutu (Bayt)	8	61	8	61	8	61	8	61
Parça Karakter CRC Hesaplama Kazancı (ms)	0,169	1,397	0,132	0,992	0,092	0,726	0,160	0,722
T _{1,5} Karakter Zaman Aşımı Kazancı (ms)	13,267	13,580	6,892	7,158	2,960	3,122	2,192	2,118
Toplam Yöntem Kazancı (ms)	13,275	14,356	7,300	8,068	2,992	3,952	2,308	2,578

Test sonuçlarına göre en düşük hız olan 1200 bit/saniye baud hızında, 61 baytlık bir paketteki toplam kazanç 14,356 ms iken 8 baytlık bir paketteki toplam zaman kazancı 13,275 ms olarak çıkmıştır. Buradaki tek değişken paket boyutu olduğu için

asıl kazanca etki parça karakter CRC hesaplamasından gelmiştir. En yüksek baud hızı olan 9600 bit/saniye baud hızında ise 61 baytlık bir paketteki toplam kazanç 2,578 ms iken 8 baytlık bir paketteki toplam kazanç 2,308 ms olarak gözlemlenmiştir. 9600 bit/saniye baud hızında da parça karakter CRC hesaplama paket boyutuna bağlı olarak zaman kazancı sağlamıştır. $T_{1,5}$ karakter zaman aşımı kazancı ise doğrudan haberleşme hızına bağlı olduğu için baud hızının 1200 bit/saniye'den 9600 bit/saniye'ye çıkması toplam zaman kazancını negatif yönde etkilemiştir. Paket boyutu 61 bayt olarak sabit alınırsa, normal haberleşmeye göre toplam zaman kazancı 14,356 ms'den 2,578 ms'ye düşerek yaklaşık %82 azalmıştır. Bu da yöntemin normal haberleşmeye göre zaman kazancının yüksek baud hızlarında etkisinin daha az olacağını göstermiştir.

6. SONUÇLAR ve YORUMLAR

Günümüzde nesnelerin internetinin yaygınlaşmasıyla beraber özel kapalı bir ağda çalışan ve en yaygın otomasyon protokollerinden biri olan Modbus RTU protokolünün de bir ağ geçidi kullanılarak internete erişimi mümkün hale gelmiştir. Modbus RTU protokolünün mevcut diğer yüksek hızlı haberleşme protokollerine göre RS485 seri hat üzerinde çalıştığı için haberleşme hızı düşük kalmaktadır ve sistem genelinde darboğaz oluşturması kaçınılmazdır. Modbus RTU protokolünde efendi cihaz ağdaki tüm köle cihazları sorgulayıp ağdaki köle cihazlara ait bilgileri güncelleyene kadar bir zaman gecikmesi olmaktadır. Bu zaman gecikmesi ağdaki köle cihaz sayısına bağlı olup ağdaki oluşabilecek olayların efendi cihaza geç ulaşmasına neden olur. Efendi cihaz sahip olduğu bilgiyi zamanında güncelleyemediği için bir üst sisteme bu bilgiyi geç gönderir. Bu da efendi cihazların bağlı olduğu merkezi kontrol sistemlerinin olaylara karşı gerçek zamanlı tepki verme süresini geciktirir. Modbus RTU protokolünün haberleşme hızında yapılacak iyileştirmeler bu yüzden oldukça önem taşımaktadır.

Bu çalışmada ortaya konan yöntem ile efendi cihazın köle cihazlardan bilgiyi daha hızlı alma imkanı doğar. Ağdaki efendi cihazlar aynı zamanda ağ geçidi olarak çalışabilmektedir. Efendi cihaz bu şekilde çalışırken birden fazla ağı kontrol etmesi gerekebilir. Bu şekilde çalışan efendi cihazlar, kendine bağlı ağlardaki köle cihazlardan gelen verilerin gerçek zamanlı alması daha fazla önem taşımaktadır. Çalışmada bu durumlar düşünülerek ortaya konan yöntemin temelini iki farklı yaklaşım oluşturur. Bu iki yaklaşım parça karakter CRC hesaplama ve $t_{1,5}$ karakter zaman aşımı yaklaşımı olarak adlandırılmıştır. Bu çalışmada bu yüzden ek olarak her birinin ayrı ayrı ve beraber kullanılmasının Modbus RTU protokolünün haberleşme hızına etkisi incelenmiştir. Test sonuçlarına göre uygulanan yöntemin Modbus RTU haberleşmesindeki efendi cihazın sorgulama hızını iyileştirdiği gözlemlenmiştir. Parça karakter CRC hesaplama yaklaşımı efendi cihazın sorgu sonucu aldığı paket boyutuna bağlı olarak zaman kazancı sağlamıştır. Bu yaklaşım sayesinde efendi cihazın kendisine köle cihazdan gelen paketin boyutu büyüdükçe, efendi cihazın gelen paketleri işleme hızı da artmıştır. Gelen paketlerin işleme hızı aynı zamanda efendi cihazın sahip olduğu donanım gücüne bağlı olarak değişkenlik gösterecektir. Diğer bir yaklaşım olan $t_{1,5}$ karakter zaman aşımı yaklaşımı ise doğrudan seri hat haberleşme

hızına bağlı olduğu için sadece baud hızının değişmesiyle zaman kazancı değişmektedir. Bunun sebebi $t_{1,5}$ karakter zaman aşımının seri hat haberleşme hızına göre belirlenmesinden kaynaklanmaktadır. $T_{1,5}$ karakter zaman aşımı süresi 1,5 karakterin seri hat üzerindeki iletim süresi olarak tanımlandığı için seri hat haberleşme hızı arttıkça düşmektedir. Tam tersi durumda da $t_{1,5}$ karakter zaman aşımı kazancı artmaktadır. Bu yaklaşımın etkisi özellikle daha düşük hızlarda çalışan ağlarda daha belirgin ortaya çıkmıştır. Bu yaklaşıma efendi cihaza köle cihazdan gelen verinin boyutunun artmasının ya da azalmasının belirgin herhangi bir etkisi gözlemlenmemiştir. İki yaklaşımın birleşiminden oluşan yöntemin en çok zaman kazancını, efendi cihaza köle cihazdan gelen paket boyutun büyük olması ve seri hat haberleşme hızının düşük hızda olmasıyla elde ettiği gözlenmiştir. Yapılan testler paket boyutunun küçük ve haberleşme hızının yüksek olması durumunda elde edilen zaman kazancının düştüğünü göstermiştir. Efendi cihaz ortaya konan yöntem ile her bir sorgudan zaman kazancı elde etmektedir. Toplam sistemsel zaman kazancı ise bir ağdaki köle cihazı sayısına ve ağ sayısını bağlı olarak artacaktır. Buradaki ağ sayısı ancak efendi cihaz ağ geçidi olarak çalışıyorsa ve birden fazla ağ kontrol ediyorsa geçerlidir. Bu şekilde sistemdeki sorgu yapılacak cihaz sayısı ile sorgu başına elde edilen zaman kazancı değeri çarpılırsa, sistem için toplam gerçek kazanç ortaya çıkacaktır. Bu sayede büyük sistemlerin bu ortaya çıkacak toplam kazanç değerine göre gerçek zamanlı davranışının iyileşeceği görülecektir.

Uygulanan yöntem sadece efendi cihaz tarafında ve köle cihazdan veri okuma sorgularında kullanılmıştır. Mevcut donanım ile yapılan testler ancak mili saniye hassaslığında sağlıklı ölçüm yapmaya imkân vermiştir. Daha hassas sonuçlar elde etmek için daha güçlü donanımların kullanılması gerektiği ortaya çıkmıştır. Ortamdaki gürültü, haberleşme esnasındaki bazı paket gecikmeleri ve kayıpları test sonuçlarını etkilese de tahmini hesaplanan değerlere paralel sonuçlar çıkmıştır.

İleriye yönelik çalışmalarda ortaya konan yöntemin diğer endüstriyel otomasyon protokolleri açısından da uygulanabilirliği araştırılabilir [34]. Yöntem uygulanmak istenen otomasyon protokollerinin kullandığı fiziksel katmandan bağımsız olduğu için ilgili otomasyon protokolün yazılım katmanında geliştirebilir. Yöntemin zaman kazancı özellikle düşük hızlarda daha belirgin olduğu için yöntem öncelikle düşük hızlarda çalışan endüstriyel otomasyon protokollerine uygulanabilir. Bu sayede ortamdaki gürültü, kullanılan zayıf donanım ve haberleşmedeki oluşacak hatalar test sonuçlarını daha az etkileyecektir.

KAYNAKLAR

- [1] 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(7), 1645-1660.
- [2] Web 1, (2018), <https://www.micrium.com/iot/devices/>, (Eriřim Tarihi: 15/12/2018).
- [3] Sideng H., Zhengming Z., Yingchao Z., Shuping W., (2008), "A Novel Modbus RTU-based Communication System for Adjustable Speed Drives", 2008 IEEE Vehicle Power and Propulsion Conference, 1-5, Harbin, Hei Longjiang, China, 3-5 September.
- [4] Web 2, (2012), http://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf, (Eriřim Tarihi: 10/11/2018).
- [5] Web 3, (2006), http://modbus.org/docs/Modbus_over_serial_line_V1_02.pdf, (Eriřim Tarihi: 10/11/2018).
- [6] Web 4, (2006), http://modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf, (Eriřim Tarihi: 10/11/2018).
- [7] Peng D., Zhang H., Zhang K., Li H., Xia F., (2009), "Research and development of the remote I/O data acquisition system based on embedded ARM platform", 2009 International Conference on Electronic Computer Technology, 341-344, Macau, China, 20-22 February.
- [8] Wang K., Peng D., Song L., Zhang H., (2014), "Implementation of Modbus communication protocol based on ARM Cortex-M0. 2014", IEEE International Conference on System Science and Engineering, 69-73, Shanghai, China, 11-13 July.
- [9] Xu S., Pan H., Ren J., Su J., (2008), "Design of the Modbus Communication Through Serial Port in QNX Operation System", 2008 International Colloquium on Computing, Communication, Control and Management, 434-438, Guangzhou, China, 3-4 August.
- [10] Guohuan L., Hao Z., Wei Z., (2009), "Research on Designing Method of CAN Bus and Modbus Protocol Conversion Interface", International Conference on Future, 180-182, Sanya, China, 13-14 December.
- [11] Zhang H., Li Y., Zhu H., (2011), "Development for protocol conversion gateway of PROFIBUS and Modbus", Procedia Engineering, 15, 767-771.
- [12] Peng D. G., Zhang H., Yang L., Li H., (2008), "Design and Realization of Modbus Protocol Based on Embedded Linux system", 2008 International

Conference on Embedded Software and Systems Symposia, 275-280, Sichuan, China, 29-31 July.

- [13] Kuang Y., (2014), "Communication Between PLC and Arduino Based on Modbus Protocol", Fourth International Conference on Instrumentation and Measurement, Computer, Communication and Control, 370-373, Harbin, China, 18-20 September.
- [14] Benito M. R., Fuertes J. M., Kahoraho E., Arzoz N. P., (1999), "Performance Evaluation of Four Field Buses", 7th IEEE International Conference on Emerging Technologies and Factory Automation, 2, 881-890, Barcelona, Spain, 18-21 October.
- [15] Lian F. L., Moyne J. R., Tilbury D. M., (2001), "Performance evaluation of control networks: Ethernet, ControlNet, and DeviceNet.", IEEE Control Systems Magazine, 21(1), 66-83.
- [16] Corotinschi G., Găitan V. G., (2018), "Enabling IoT connectivity for Modbus networks by using IoT edge gateways", 2018 International Conference on Development and Application Systems, 175-179, Suceava, Romania, 24-26 May.
- [17] Akkaya Ş., (2015), "FPGA Tabanlı Modbus Ağ Geçidi Tasarımı", Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi.
- [18] Erkek İ., (2018), "Modbus Temelli SCADA Sistemlerinin Siber Güvenliği İçin Yeni Bir Yaklaşım", Yüksek Lisans Tezi, Gazi Üniversitesi.
- [19] Demir B., (2012), "Endüstriyel Ağlar için Linux Tabanlı 3G Haberleşme Özellikli Modbus Mesaj Önceliklendirme Mekanizmalı Modbus Gateway Tasarımı", Yüksek Lisans Tezi, Gebze Teknik Üniversitesi.
- [20] Türkan O., (2008), "Development of a Wireless Transmission Module for Modbus Applications", Yüksek Lisans Tezi, Atılım Üniversitesi.
- [21] Süzer E. S., (2006), "Uzaktan Sayaç Okuma Teknikleri ve Modbus RTU, IEC61107 Mod C Protokolleri ile Örnek Yazılım", Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi.
- [22] Groover M. P., (2007), "Automation, Production Systems, and Computer-integrated Manufacturing", 4th Edition, Prentice Hall Press.
- [23] Web 5, (2018), <https://realpars.com/what-is-industrial-automation/>, (Erişim Tarihi: 15/12/2018).
- [24] Web 6, (2018), <https://schedule.sxsw.com/2018/events/PP80535>, (Erişim Tarihi: 11/12/2018).
- [25] Groover M. P., (2007), "Fundamentals of Modern Manufacturing: Materials Processes, and Systems", 5th Edition , John Wiley & Sons.

- [26] Künzel G., Ribeiro M. A. C., Pereira C. E., (2014), “A Tool For Response Time and Schedulability Analysis in Modbus Serial Communications”, 2014 12th IEEE International Conference on Industrial Informatics, 446-451, Porto Alegre, Brazil, 27-30 July.
- [27] Web 7, <https://www.arduino.cc/en/Main.ArduinoBoardUno>, (Erişim Tarihi: 10/10/2018).
- [28] Banzi M., Shiloh M., (2014), "Getting Started with Arduino: The Open Source Electronics Prototyping Platform", 3th Edition, Maker Media Inc.
- [29] Web 8, (2018), <https://maker.robotistan.com/arduino-uno/>, (Erişim Tarihi: 11/12/2018).
- [30] Web 9, (2018), <http://datasheets.maximintegrated.com>, (Erişim Tarihi: 21/12/2018).
- [31] Web 10, (2016), <https://github.com/jecrespo/simple-modbus>, (Erişim Tarihi: 12/01/2019).
- [32] Kernighan B. W., Ritchie D. M., (1988), "The C programming language.", 2nd Edition, Prentice Hall.
- [33] Web 11, (2018), https://www.modbustools.com/modbus_poll.html, (Erişim Tarihi: 19/12/2018).
- [34] Kay J. A., Mazur D. C., Entzminger R. A., (2018), “Basics of Communication Networks for Electrical Engineers in the Forest Products Industries”, 2018 IEEE IAS Pulp, Paper and Forest Industries Conference, 1-5, Appleton, WI, USA, 17-21 June.

ÖZGEÇMİŞ

Salih BAKIR 1986 yılında Trabzon’da doğdu. 2004 yılında başladığı Yıldız Teknik Üniversitesi Mühendislik Fakültesi Elektronik Mühendisliği Bölümünü 2008 yılında başarıyla tamamlayarak 2013 yılında yüksek lisans eğitimine Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik Mühendisliği Anabilim Dalında başladı. 2011 yılından bu yana Siemens A.Ş’de endüstriyel otomasyon konularında yazılım ürün geliştirici ve yazılım mimarı olarak çalışmaktadır.

EKLER

Ek A: Tez Çalışması Kapsamında Yapılan Yayınlar

Bakir S., Aliew F., (2018), "Modbus RTU Protokolünün Haberleşme Hızının İyileştirilmesine Yönelik bir Yöntem", 12. İstanbul Bilişim Kongresi, 13-14, İstanbul, Türkiye, 6 Aralık.