

T.C.
GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ
ENSTİTÜSÜ

ENDÜSTRİYEL UYGULAMALAR İÇİN
LİNX TABANLI 3G HABERLEŞME
ÖZELLİKLİ AKILLI MODBUS MESAJ
ÖNCELİKLENDİRME MEKANİZMALI
MODBUS GATEWAY TASARIMI

Beşir DEMİR
YÜKSEK LİSANS TEZİ
ELEKTRONİK MÜHENDİSLİĞİ
ANABİLİM DALI

GEBZE

2012

T.C.
GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ
ENSTİTÜSÜ

ENDÜSTRİYEL UYGULAMALAR İÇİN
LINUX TABANLI 3G HABERLEŞME
ÖZELLİKLİ AKILLI MODBUS MESAJ
ÖNCELİKLENDİRME MEKANİZMALI
MODBUS GATEWAY TASARIMI

Beşir DEMİR
YÜKSEK LİSANS TEZİ
ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

TEZ DANIŞMANLARI
Doç. Dr. Oğuz KUCUR
Yrd. Doç. Dr. Enver ÇAVUŞ

GEBZE

2012



YÜKSEK LİSANS TEZİ JÜRİ ONAY SAYFASI

G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 22.07.2012 tarih ve 2012/27... sayılı kararıyla oluşturulan jüri tarafından 11.10.2012 tarihinde tez savunma sınavı yapılan Besir DEMİR.....'ın tez çalışması Elektronik Mühendisliği Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRİ

ÜYE

(1.TEZ DANIŞMANI)

: Doç. Dr. Oğuz Kucur

ÜYE

(2. TEZ DANIŞMANI)

: Yrd. Doç. Dr. Enver GAVUŞ

ÜYE

: Yrd. Doç. Dr. Serdar S. ERDEM S. S. Guller

ÜYE

: Doç. Dr. Abdetkadir Bekki, A. Bekki

ÜYE

: Doç. Dr. Serkan Aksoy

ONAY

G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun/...../20... tarih ve/..... sayılı kararı.

İMZA/MÜHÜR

ÖZET

TEZİN BAŞLIĞI: ENDÜSTRİYEL UYGULAMALAR İÇİN LINUX TABANLI 3G HABERLEŞME ÖZELLİKLİ AKILLI MODBUS MESAJ ÖNCELİKLENDİRME MEKANİZMALI MODBUS GATEWAY TASARIMI

YAZAR ADI: BEŞİR DEMİR

Bu çalışmada, SCADA sistemlerinde RTU'ların uzaktan izlenmesi ve kontrol edilmesi için kullanılan bir Modbus gateway cihazı tasarlanmıştır. Tasarlanan cihazda, haberleşme protokolü için Modbus, kablosuz olarak yüksek hızlarda veri haberleşmesi içinse 3G teknolojisi kullanılmıştır. Modbus protokolü kullanılarak yapılan sorgulama işlemlerinde gelen istekler için ID önceliklendirme mekanizması gerçekleştirilmiştir. Bu sayede, düşük ID'li RTU'ların sorgulanması hızlandırılmıştır. Kullanılan Modbus TCP ve Modbus RTU protokolleri, 3G teknolojisi ve gerçekleşen ID önceliklendirme mekanizmaları ile sistem tasarımının artan karmaşıklığı, açık kaynak kodlu gömülü Linux işletim sistemi kullanılarak kolaylaştırılmıştır.

Bu çalışmada ARM9 tabanlı EP9302 işlemcisi kullanılmıştır. Tasarlanan cihazda donanım olarak dijital girişler, dijital çıkışlar, röle çıkışları ve RS485 çıkışı bulunmaktadır. Sunucu modunda çalışan Modbus gateway cihazı aynı anda birden fazla istemciye cevap verebilecek şekilde tasarlanmıştır. Modbus protokolünde sorgulama işlemi yapan bu istemciler Master olarak adlandırılırlar. Modbus gateway cihazına aynı anda birden fazla Master sorgulama işlemi yapabildiğinden Multi-Master (çok yöneticili) olarak çalışabilmektedir. Birden fazla istemciden gelen Modbus veri paketleri, ID ve kuyrukta kalma süresi parametrelerine göre iki farklı kuyrukta sıralanmaktadır. Daha sonra, kuyrukta belli bir sürenin üzerinde bekleyenlerin sıralandığı kuyruğa öncelik verilerek, istekler cevaplanmaktadır. Tasarlanan cihazın birden fazla istemciye cevap verebilmesi için işletim sisteminin çok izlekli (multi-thread) özelliği kullanılmıştır.

SUMMARY

TITLE OF THESIS: THE DESIGN OF A LINUX BASED, 3G CAPABLE MODBUS GATEWAY SYSTEM WITH INTELLIGENT MODBUS MESSAGE PRIORITIZATION MECHANISM FOR INDUSTRIAL APPLICATIONS

AUTHOR: BEŞİR DEMİR

In this study, a Modbus gateway system is designed to remotely monitor and control the RTUs in SCADA systems. In the designed system, Modbus is used for communication protocol and 3G technology is utilized to provide high speed data wireless communication. An ID prioritization mechanism is implemented for query request by using Modbus protocol. In this way, the query process is become faster for the lower ID devices in question. The increased design complexity due to use of Modbus TCP and Modbus RTU protocols, 3G technology and implemented ID prioritization mechanisms, is simplified by using open source embedded Linux operating system.

In this work, an ARM based EP9302 processor is used. The Modbus gateway device has digital inputs, digital outputs, relay outputs and RS485 outputs as hardware design components on it. In working server mode, Modbus gateway device is designed to respond multiple clients simultaneously. Clients or server devices that make query processing on Modbus protocol are called Masters. As more than one Master is capable of operating query processing on designed system, the Modbus gateway device can work as a Multi-Master. Data packets that came from more than one client are sorted in two different queues according to their Slave ID and duration of stay in the queue. Then, the queries are answered by prioritizing the queue that holds the queries waiting in line more than a predetermined time. In order to respond more than one client, the designed device uses the multi-threaded feature of the operating system.

TEŞEKKÜR

Yüksek lisans eğitimim boyunca desteğini esirgemeyen Yrd. Doç. Dr. Enver Çavuş ve Doç. Dr. Oğuz Kucur hocalarıma, teknik bilgi, maddi ve manevi desteklerini esirgemeyen Mikrodev firmasının çalışanlarına ve beni yetiştirip bu günlere gelmemi sağlayan aileme, arkadaşlarıma ve Mehmet Emin Okudan arkadaşşıma teşekkürlerimi sunarım.

İÇİNDEKİLER DİZİNİ

	<u>Sayfa</u>
ÖZET	iv
SUMMARY	v
TEŞEKKÜR	vi
İÇİNDEKİLER DİZİNİ	vii
SİMGELER VE KISALTMALAR DİZİNİ	ix
ŞEKİLLER DİZİNİ	xi
TABLolar DİZİNİ	xiii
1. GİRİŞ	1
2. GÖMÜLÜ LINUX SİSTEMLERİ	8
2.1. Gömülü Sistemler	8
2.2. Gömülü Linux İşletim Sistemi	9
2.3. Gömülü Linux Uygulamalarının Geliştirilmesi	11
2.3.1. Gömülü Linux Geliştirmek için Gerekli Uygulamaların Kurulumu	13
2.3.1.1. Gömülü Sistem Geliştirme Kiti (ELDK)	13
2.3.1.2. TFTP Sunucusu	14
2.3.1.3. NFS Sunucusu	15
2.3.2. Linux Çekirdeğinin (Kernelinin) Derlenmesi	17
2.3.3. UImage Dosyasının Modbus Gateway Cihazına Yüklenmesi	21
2.3.4. Cihaz Üzerinde Uygulama Geliştirme ve Çalıştırma	23
3. 3G HABERLEŞME TEKNOLOJİSİ	27
3.1. 3G Teknolojisi ve Gelişimi	27
3.2. Modbus Gateway Cihazının 3G Modem ile İnternet Bağlantısının Sağlanması	28
3.2.1. Sim5218 3G Modem	29
3.2.2. Sim5218 3G Modemi ile Cihazın İnternet Bağlantısının Sağlanması	33
4. MODBUS PROTOKOLÜ	40
4.1. Modbus Protokolünün Modbus Gateway Cihazı Tarafından Kullanılması	40
4.2. Modbus TCP Protokolü	43
4.2.1. Modbus TCP Paket Yapısı	44
4.2.1.1. MBAP Header Paketinin İçeriği	45

4.2.2.	Modbus TCP Veri Paketi İerięi	47
4.2.3.	Modbus TCP Protokolü Uygulama Örneęi	49
4.3.	Modbus RTU Protokolü	51
4.3.1.	Modbus RTU Sorgulama Örneęi	54
4.4.	Modbus Veri Modeli	56
4.5.	Modbus Fonksiyon Kodları ve Tanımlamaları	58
4.6.	Modbus Protokolü Hata Kodları	68
5.	MODBUS GATEWAY CİHAZI TASARIMI	70
5.1.	Donanım Tasarımı	71
5.1.1.	Dijital Girişler	74
5.1.2.	Dijital Çıkışlar	75
5.1.3.	Röle Çıkışları	77
5.1.4.	Güç Bağlantı Şeması	78
5.1.5.	RS485 Bağlantı Şeması	78
5.2.	Yazılım Tasarımı	80
5.2.1.	Soket Programlama	82
5.2.1.1.	Soket Türleri ve IP Protokolü	83
5.2.1.2.	Uygulamadaki Temel Prensipler	83
5.2.2.	Threads (İzlekler)	87
5.2.3.	İstekleri Kuyruęa Alma ve Kuyruktan Silme İşlemi	92
5.2.3.1.	İsteklerin Kuyruęa Eklenmesi İşlemi	93
5.2.3.2.	Kuyruktan Eleman Silme İşlemi	96
5.3.	Sistemin Çalıştırılması ve Sonuçlar	97
5.3.1.	Test Ortamı ve Sistemin çalıştırılması	97
5.3.2.	Sonuçlar	103
6.	SONUÇ VE ÖNERİLER	107
	KAYNAKLAR DİZİNİ	109
	ÖZGEÇMİŞ	112

SİMGELER VE KISALTMALAR DİZİNİ

SCADA	: Supervisory Control and Data Acquisition
MTU	: Master Terminal Unit
PLC	: Programmable Logic Controller
DCS	: Distributed Control System
LCD	: Liquid Crystal Display
3G	: 3rd Generation
ID	: Identifier
TCP	: Transmission Control Protocol
GPRS	: General Packet Radio Service
PC	: Personal Computer
USB	: Universal Serial Bus
IP	: Internet Protocol
PPP	: Point to Point Protocol
IPSEC	: Internet Protocol Security
GSM	: Global System for Mobile Communications
TFTP	: Trivial File Transfer Protocol
NFS	: Network File System
ROM	: Read-only memory
PDA	: Personal Digital Assistant
OS	: Operating System
CAN	: Controller Area Network
ELDK	: Embedded Linux Development Kit
HSCSD	: High Speed Circuit Switched Data
EDGE	: Enhanced Data Rates in a GSM Environment
HSPA	: High Speed Packet Access
WCDMA	: Wideband Code Division Multiple Access
UART	: Universal Asynchronous Receiver/Transmitter
UMTS	: Universal Mobile Telecommunications System
3GPP	: 3rd Generation Partnership Project
GPS	: Global Positioning System
CSD	: Circuit Switched Data
HSUPA	: High-Speed Uplink Packet Access

HR	: Half Rate
FR	: Full Rate
EFR	: Enhanced Full Rate
AMR	: Adaptive Multi-Rate
DTMF	: Dual-tone multi-frequency
FCC	: Federal Communications Commission
GPIO	: General Purpose Input/Output
DC	: Direct Current
RF	: Radio frequency
USIM	: Universal Subscriber Identity Module
RTC	: Real-Time clock
I2C	: Inter IC Bus
ADC	: Analog to Digital Converter
NMEA	: National Marine Electronics Association
ADU	: Application Data Unit
PDU	: Protocol Data Unit
CRC	: Cyclic Redundancy Check
LRC	: Longitudinal Redundancy Check
ASCII	: American Standard Code for Information Interchange
MSB	: Most Significant Byte
LSB	: Least Significant Byte
CPU	: Central Processing Unit
POSIX	: Portable Operating System Interface for Unix
FIFO	: First In- First Out
UDP	: User Datagram Protocol
RAM	: Random-Access Memory
PWM	: Pulse Width Modulation
RTU	: Remote Terminal Unit
MHz	: Megahertz
Kbps	: Kilobits per second
Mbps	: Megabit per second
MΩ	: Mega ohm
A	: Amper
Ms	: Milisaniye

ŞEKİLLER DİZİNİ

<u>Sekil</u>	<u>Sayfa</u>
1.1. SCADA sisteminin genel görünümü	2
1.2. Yapılan çalışmanın blok diyagramı	6
2.1. Gömülü sistemin genel yapısı	9
2.2. Host bilgisayar ile gömülü sistem bağlantısı	11
2.3. Gömülü Linux geliştirme adımları	11
2.4. Çapraz derleme (Cross Compile) işlemi	14
2.5. TFTP sunucu kurulumu	15
2.6. NFS sunucusunun aktifleştirilmesi ve ayarlarının yapılması	16
2.7. Linux çekirdeğinin yazılım ve donanım arasındaki konumu	17
2.8. Çekirdek konfigürasyon ayarlarının yapılması	19
2.9. Konfigürasyon ayarlarından sonra terminalin görüntüsü	20
2.10. uImage oluşturma	20
2.11. uImage oluşturulması ve cihaza yükleme adımları	21
2.12. Minicom ekran görüntüsü	22
2.13. Cihaz üzerinde uygulama çalıştırma adımları	23
2.14. Programın Arm işlemcisi için derlenmesi	24
2.15. NFS Sunucu ile host bilgisayara mount işleminin gerçekleştirilmesi	25
2.16. Mount işleminin gerçekleştirilmesi	26
3.1. 3G modem ile internet bağlantısının sağlanması	28
3.2. Sim5218 ve geliştirme modülü	31
3.3. Gömülü bilgisayar ile 3G modem bağlantısı	34
3.4. USB sürücüsünün modül olarak ayarlanması	37
4.1. Modbus ağ mimarisi	41
4.2. Modbus gateway cihazı istek/cevap veri paketi dönüşüm şeması	42
4.3. İstemci sunucu mesaj akışı	43
4.4. Modbus TCP/IP istek/cevap mesajı (Hatasız cevap)	44
4.5. Modbus TCP/IP istek/cevap mesajı (Hatalı cevap durumunda)	44
4.6. Modbus TCP/IP istek/cevap paketi	45
4.7. Modbus TCP uygulaması	49
4.8. Modbus sorgu ve cevap paketi	52
4.9. Modbus RTU bağlantı şeması	54
4.10. Modbus veri modeli	57
4.11. Bobin durumlarının okunması, istek/cevap mesajlarının yapısı	60
4.12. Bobin durumlarının okunması, istek/cevap paketi içerikleri örneği	60
4.13. Tutucu yazmaçların okunması, istek/cevap mesajlarının yapısı	61
4.14. Tutucu yazmaçların okunması, istek/cevap paketi içerikleri örneği	62
4.15. Giriş yazmaçlarının okunması, istek/cevap mesajlarının yapısı	62
4.16. Giriş yazmaçlarının okunması, istek/cevap paketi içerikleri örneği	63
4.17. Tek bir yazmaca yazma, istek/cevap mesajlarının yapısı	64

4.18. Tek bir yazmaca yazma, istek/cevap paketi içerikleri örneği	64
4.19. Birden fazla bobine yazma, istek/cevap mesajlarının yapısı	65
4.20. Birden fazla bobine yazma, istek/cevap paketi içerikleri örneği	66
4.21. Birden fazla tutucu yazmaçlara yazma, istek/cevap mesajlarının yapısı	67
4.22. Birden fazla tutucu yazmaçlara yazma, istek/cevap paketi içerikleri örneği	67
5.1. Modbus gateway cihazı genel görünümü	70
5.2. ARM CPU modülü	71
5.3. Modbus gateway cihazı fiziksel giriş çıkış bağlantı şeması	73
5.4. Modbus gateway cihazının görünümü	74
5.5. Dijital girişlerin devresi	74
5.6. Dijital girişlerin bağlantı şeması	75
5.7. Dijital çıkışların devresi	76
5.8. Dijital çıkışların bağlantı şeması	76
5.9. Röle çıkışlarının devresi	77
5.10. Röle çıkışların bağlantı şeması	78
5.11. Güç bağlantı şeması	78
5.12. RS485 bağlantı şeması	79
5.13. Soket açma	85
5.14. send() fonksiyonu kullanımı	86
5.15. recv() fonksiyonu kullanımı	87
5.16. Oluşturulan thread mekanizmaları	88
5.17. Oluşturulan mutex yapıları	89
5.18. Mutex lock ve mutex unlock işlemlerinin bağlantı durumunda kullanılması	90
5.19. Mutex lock ve mutex unlock işlemlerinin kuyruktan eleman silerken kullanımı	90
5.20. Kuyruğa alma ve kuyruktan eleman silme işlemleri için thread'lerin oluşturulması	91
5.21. Bağlı liste	92
5.22. Veri yapısı parametreleri	93
5.23. İstekleri kuyruğa ekleme fonksiyonu	94
5.24. İsteklerin kuyruğa alınması	95
5.25. Kuyruğun belirlenen periyotlarda taranarak güncellenmesi	96
5.26. Kuyruktan eleman silme fonksiyonu	97
5.27. Çalışan sistemin görünümü	98
5.28. Test ortamı	99
5.29. Test için kullanılan fiziksel giriş/çıkışlar.	99
5.30. Entegre giriş çıkışların sorgulanması	102
5.31. Modpoll sorgulama ekranı	102
5.32. Modpoll programı ile yapılan sorgulama işleminde parametrelerin anlamı	103
5.33. Kuyruğa eklenen modbus istekleri	104
5.34. Kuyruğun kontrol edilmesi	105
5.35. Kuyruğun güncellenmesi	105

TABLOLAR DİZİNİ

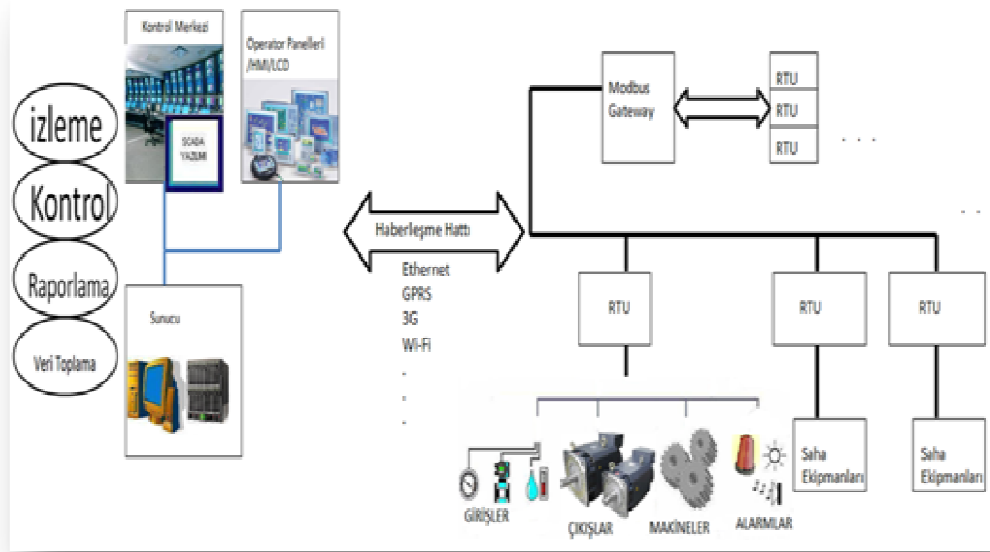
<u>Tablo</u>	<u>Sayfa</u>
3.1. Sim5218 modemin özellikleri	29
3.2. 3G modem geliştirme kiti birimleri	32
3.3. Eşleşmiş USB sanal portlar	38
4.1. MBAP header paketi içeriği	46
4.2. Modbus TCP istek paketi içeriği	47
4.3. Modbus TCP cevap paketi içeriği	48
4.4. Modbus TCP sorgu paketi	50
4.5. Modbus TCP cevap paketi	51
4.6. Modbus RTU istek paketi	55
4.7. Modbus RTU cevap paketi	56
4.8. Değişik fonksiyonlar için değişik veri grupları	57
4.9. Fonksiyon kodları ve açıklamaları	59
4.10. Hata fonksiyon kodları	68
5.1. ARM CPU modül özellikleri	72
5.2. Dijital girişlerin özellikleri	75
5.3. Dijital çıkışların özellikleri	76
5.4. Röle çıkışlarının özellikleri	77
5.5. İstemci ve sunucu arasındaki işlemler	84
5.6. Dijital girişlerin ayrıştırılması	100
5.7. Dijital çıkışların ayrıştırılması	101
5.8. Röle çıkışların ayrıştırılması	101

1. GİRİŞ

Dünyada özellikle 1970’li yıllardan sonra, endüstriyel ürünlerin tasarımı ve üretiminde köklü değişikliklerin ortaya çıkması ile birlikte sistemlerin daha verimli ve daha az hata ile çalışabilmesi gerekliliği, uzaktan izleme ve kontrol ihtiyaçlarını da beraberinde getirmiştir. Uzaktan izleme ve kontrol sistemleri, ilk başlarda uzak noktalardaki elektrik ve su dağıtım şebekeleri için geliştirilmiştir. Teknolojinin ilerlemesi ve maliyetlerin düşmesiyle, bu sistemler kendilerine daha geniş uygulama alanları bulabilmişlerdir. Günümüzde kimya endüstrisinden tarıma, hava kirliliği kontrolünden bina güvenliğine kadar birçok kullanım alanı vardır. Örnek olarak endüstriyel proses kontrol, tarımsal sulama sistemleri, sıcaklık alarmı, bina kontrol sistemleri, su ve elektrik sayaçlarının uzaktan okunması gibi uygulamalar verilebilir.

Uzaktan izleme ve kontrol sistemleri genel olarak denetleyici gözetim ve veri toplama sistemi veya SCADA (Supervisory Control and Data Acquisition) sistemi olarak da adlandırılmaktadır. SCADA sistemleri PLC (Programmable Logic Controller), döngü kontrolörleri, dağıtık kontrol sistemleri (DCS, Distributed Control System), giriş/çıkış sistemleri ve akıllı sensörler gibi çeşitli cihazlardan gelen verileri (çalışıyor, duruyor, sıcaklık, basınç, hız vb.) gerçek zamanlı olarak endüstriyel otomasyon, bina otomasyonu, enerji izleme gibi uygulama alanlarından toplayan sistemlerdir. SCADA sistemleri aracılığıyla elde edilen gerçek zamanlı veriler sayesinde işletmedeki eksiklikler anında tespit edebilmekte ve bu eksiklikler zamanında giderilebilmektedir. Bu şekilde üretimde aksayan noktaların bulunması ve kalitede meydana gelebilecek aksaklıkların kısa sürede giderilmesi sağlanmaktadır [26].

Şekil 1.1’de günümüzde kullanılan genel bir SCADA sistemi verilmiştir. Şekil 1.1’den de anlaşılacağı üzere SCADA sistemleri; kontrol merkezi, iletişim sistemi ve RTU (Remote Terminal Unit) olmak üzere üç ana bileşenden oluşmaktadır.



Şekil 1.1. SCADA sisteminin genel görünümü

Kontrol merkezi, SCADA sistemlerinde ana kontrolör görevini üstlenen ana istasyon veya bilgisayardır. Ana terminal birimi veya MTU (Master Terminal Unit) olarak da adlandırılmaktadır. Uzak noktadaki RTU'lar tarafından toplanan verileri, operatör panelleri, HMI (Human Machine Interface) ve LCD (Liquid Crystal Display) ekranları aracılığıyla izlenmesini ve gerekli kontrol işlemlerinin gerçekleştirilmesi işlemlerini gerçekleştirir [26]. Kontrol merkezi bir bilgisayar, bilgisayar ağı, yazıcılar veya yedekleme üniteleri gibi cihazlardan oluşmaktadırlar. MTU birimleri diğer sistemler ile iletişim hatları veya yerel ağlar sayesinde bağlantı sağlarlar [28].

SCADA sistemlerin önemli bileşenlerinden olan iletişim sistemi, RTU ve kontrol merkezi ve sunucu arasında karşılıklı olarak veri haberleşmesini sağlamaktadır. İletişim sistemlerinin önemli bileşenlerinden bir tanesi veri haberleşmesidir. Bu veri haberleşmesi, çevre şartlarına bağlı olarak kablolu veya kablosuz olarak yapılabilmektedir. Teknolojinin gelişmesi ile beraber endüstriyel uygulamalarda uzaktan izleme ve kontrol, her noktadan internet erişimi ile kolaylıkla yapılabilmektedir. İnternet için istenen her yere kablolu bağlantı imkanı olmadığında veya kablo maliyetlerinin arttığı noktalarda, GPRS, 2G veya 3G modemleri gibi

kablosuz haberleşme sistemleri kullanılmaktadır. GPRS ve 2G modemlerin sınırlı veri aktarım hızından dolayı, gerçek zamanlı sorgulama işlemlerinde 3G modemler kullanılmaktadır [5][21].

SCADA sistemlerindeki iletişim sistemlerinin önemli bileşenlerinden bir tanesi de haberleşme protokolüdür. Haberleşme protokolleri, haberleşecek cihazların veya birimlerin arasında veri alışverişi sırasında kullanılması gereken veri formatını ve izlenmesi gereken kuralları belirtir. Endüstriyel ortamlarda proseslerin gözlenmesi, cihazların kontrolü, makinelerden verilerin elde edilmesi, testlerin yapılması veya kullanıcı müdahalesi gerektiren işlerde yaygın olarak bir haberleşme protokolü kullanılmaktadır. Endüstride yaygın olarak cihazlar arasında haberleşme için genellikle Profibus, DeviceNet ve Canbus gibi endüstriyel bus protokolleri kullanılmasına rağmen, endüstriyel kontrol ve otomasyon işlemlerinde Modbus protokolü ücretsiz olması, gerçek zaman performansı, güvenilirlik, esneklik gibi özelliklerinden dolayı öne çıkmıştır [3]. Modbus, endüstriyel uygulamalarda kullanılan en eski seri iletişim protokollerinden birisidir. 1978 yılında Modicon firmasının kendi ürünleri arasında haberleşmeyi sağlamak için geliştirdiği bir haberleşme protokolü olan Modbus, zamanla PLC cihazlar arasındaki veri alışverişini sağlayan standart bir iletişim protokolü olarak sektörde yerini almıştır. Diğer iletişim protokolleri ile kıyaslandığında, Modbus bir PC veya bir mikroişlemci ile birlikte kullanılabilmekte ve bunun yanında sağlam geçmişi ve basit altyapısı ile artan sayıda imalatçı tarafından desteklenmektedir. Bu sebeple mevcut endüstriyel cihazların çoğu ile iletişim kurulabilmektedir [8].

SCADA sistemlerinin bir diğer bileşeni, RTU olarak adlandırılan uzak terminal birimleridir. RTU, genellikle mikroişlemci tabanlı, sistem değişkenlerini toplayan, gerektiğinde depolayan, ayrıca bu bilgileri kontrol merkezine, belirli bir iletişim yolu ile gönderen bir SCADA birimidir. İlk SCADA sistemlerinde, uzak erişim noktalarındaki RTU cihazları sadece bir görevi üstlenen mikroişlemci kontrollü basit bir sistem yapısına sahip olmasına karşın, günümüzde bu cihazlar çok sayıda sayısal ve analog giriş/çıkışlara sahip, akıllı kontrol işlemleri yapabilen, çeşitli haberleşme donanımlarını ve haberleşme protokollerini destekleyen sistemlerdir. Modern RTU cihazlarının bu çok yönlü özelliklerinden dolayı ortaya çıkan sistem karmaşıklığını en aza indirmek ve tasarım süresini kısaltıp maliyetleri düşürmek için, genellikle

tasarlanacak bir sistemin hazır bir platform üzerinden kurgulanması gerekmektedir. Bu doğrultuda, USB host kütüphanesi, Ethernet sürücüler, TCP/IP soket kütüphanesi, PPP, IPSEC gibi açık kaynak kodlu hazır kütüphanelerinin olması, sahip olduğu geniş donanım desteği, uygulama yazılımlarının çokluğu ve kalitesinden dolayı, tasarlanacak bir RTU cihazı için hazır platform olarak kullanılabilecek en uygun işletim sistemi Linux olmaktadır [2].

Şekil 1.1’de gösterilen modern SCADA sistemlerinin bir diğer önemli bileşeni ise Modbus gateway cihazlarıdır. Modbus gateway cihazları uç noktalarda bulunan ve Modbus destekli RTU birimlerine ethernet veya GSM kanalları üzerinden erişim sağlamaktadır. Kontrol merkezinden gelen talepler, Modbus gateway cihazları tarafından Modbus protokolü kullanılarak RS485 portu üzerinden RTU birimlerine ulaştırılmakta ve aynı şekilde Modbus gateway cihazlarının RS485 portuna gelen cevaplar, GSM veya ethernet üzerinden kontrol merkezine ulaştırılmaktadır. Uzak noktalarda sorgulanacak cihazlar birbirine yakın mesafede iken Modbus RTU protokolünü destekleyen cihazların Modbus gateway cihazları kullanılarak sorgulanması zaman ve maliyet açısından daha verimli olmaktadır. Örneğin uzak noktadaki 10 adet RTU’yu sorgulamak için 10 ayrı 3G modem kullanılması yerine 1 adet modem kullanılarak Modbus gateway cihazı üzerinden sorgulanması tasarım ve haberleşme maliyetleri açısından daha verimli olmaktadır.

Literatürde endüstriyel uygulamalarda uzaktan izleme ve kontrol sistemlerinde kullanılmak üzere tasarlanmış birçok Modbus gateway cihazı yer almaktadır. Bu çalışmalardan, gömülü Linux işletim sistemi, 3G haberleşme özelliği ve Modbus protokollerini bir arada kullanan tasarımlardan ilki 2010 yılında gerçekleştirilmiştir [21]. Bu çalışmada, 3G erişim imkanı sağlayan bir Modbus gateway cihazı üretimi gerçekleştirilmiştir. Çin’de kullanılan Modbus protokolünün daha çok kablolu veya 2G haberleşme imkanları kullanılarak gerçekleştirilmekte iken, tasarlanan 3G haberleşme imkanı sağlayan Modbus gateway cihazı sayesinde RTU birimleri ile daha hızlı veri haberleşmesi imkanı sağlanmıştır. Bu konudaki diğer bir çalışmada, Power PC kullanılarak uzaktan izleme ve kontrol platformu, Modbus protokolü kullanılarak gerçekleştirilmiştir [3]. Power PC MPC8242 işlemcisi ile gömülü Linux işletim sistemleri kullanılarak, Modbus RTU Master sayesinde endüstriyel uygulamalar için veri toplama, uzaktan izleme, kontrol ve ağ haberleşmesi yapabilen

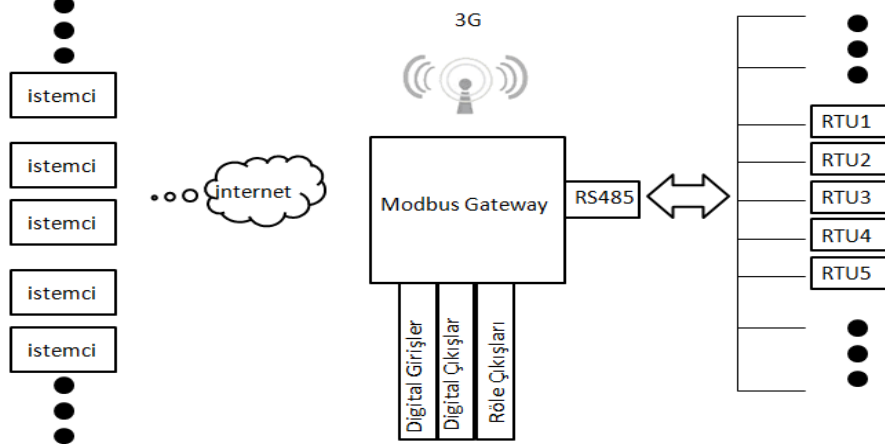
bir platform tasarımı gerçekleştirilmiştir. ARM9 ve 3G teknolojileri kullanılarak akıllı ev sistemleri için uzaktan izleme imkanı sağlayan bir çalışma da 2009 yılında gerçekleştirilmiştir [5]. S3C2410 donanım platformu ve gömülü Linux kullanılarak gerçek zamanlı video izleme sayesinde ev, ofis gibi mekanların 3G haberleşme özelliği sayesinde uzaktan izlenmesi sağlanmıştır. Yapılan çalışmada video sıkıştırma, alarm girişleri, alarm çıkışları ve kontrol birimi sayesinde 3G üzerinden izleme ve kontrol işlemleri gerçekleştirilebilmektedir.

Literatürde yapılan çalışmalar haricinde, uzaktan izleme ve kontrol işlemlerinde kullanılmak üzere endüstride geliştirilen birçok Modbus gateway cihazı mevcuttur. Bu ürünlerden Moxa firmasına ait “ioLogik W5340-HSDPA” ürünü ve Systech firmasının Modbus gateway ailesi ürünleri, bu tez çalışması kapsamında gerçekleştirilen tasarıma benzerlik göstermektedir. Moxa firmasının ioLogik ürününde, bir sunucu tarafından cihaz üzerindeki giriş çıkışlar izlenebilmekte ve aynı zamanda Ethernet portuna bağlanan cihazlara, 3G üzerinden yönlendirme işlemi ile erişim sağlanabilmektedir. Cihazdaki IP gateway özelliği sayesinde, Ethernet portuna bağlı cihazlara 3G üzerinden erişim sayesinde bağlı RTU’lar sorgulanabilmektedir [19]. Systech firmasının Modbus gateway ailesi ürünlerinde ise ioLogik ürününden farklı olarak ethernet haberleşme özelliği ve birden fazla istemci tarafından cihaza seri porttan bağlı RTU’lar sorgulanabilmektedir. İstemciler tarafından aynı anda yapılan sorgulama işlemlerinde istekler geliş sırasına göre kuyruğa alınmaktadır [20].

Yukarıda bahsedilen çalışmalardan farklı olarak, bu tez çalışması kapsamında, Linux tabanlı 3G haberleşme özellikli ve akıllı mesaj önceliklendirme mekanizmalı bir Modbus gateway cihazının tasarlanması amaçlanmıştır. Tasarlanan cihazın üzerindeki entegre giriş çıkışlar ve bağlanan RTU’lara ait parametreler sorgulanma özelliğine sahiptir. Yapılan tez çalışmasının önceki çalışmalardan en önemli farkı, sorgulama işlemi birden fazla istemci tarafından aynı anda, ID önceliğine göre gerçekleştirilmesidir. Bu çalışmada gerçekleştirilen cihazın Moxa firmasına ait ürününde bulunmayan birçok özelliği bulunmaktadır. Tasarlanan cihaz sunucu modunda çalışarak Modbus TCP ve Modbus RTU protokollerini desteklemekte ve entegre giriş çıkışlar sorgulanabilmektedir. Bununla birlikte, tasarlanan cihazda 3G haberleşme imkanı ve birden fazla kullanıcının aynı anda bağlantı kurma imkanı sağlanmıştır. Aynı şekilde, Systech firmasının ürününün sıralama ve cevap verme

işleminde farklı olarak, bu çalışmada tasarlanan sistem istemci tarafından gelen istekler ID önceliğine göre sıralanmakta ve cevap verme işlemi de ID önceliğine göre yapılmaktadır.

Yapılan çalışmanın blok diyagramı Şekil 1.2’de görüldüğü gibidir. Bu çalışmada ARM9 tabanlı EP9302 işlemcisi kullanılmıştır. Üzerine EP9302 yaması eklenmiş Linux çekirdeği derlenip işlemciye kurulduğunda gömülü Linux ile çalışma yapabilecek platform hazır hale getirilmiştir. Gömülü Linux kullanılmasıyla beraber gelen açık kaynak kodlu olma özelliği ile gerekli kütüphanelerin kullanımı kolaylaşmıştır [2]. Gömülü Linux işletim sistemi kullanılarak 3G haberleşme özelliği imkanı sağlayan Modbus gateway cihazı tasarlanmıştır. Modbus protokolünün gerçekleştirilmesi işleminde, birden fazla istemcinin cihaza aynı anda bağlantı kurarak sorgulama yapabilme özelliği eklenmiştir. Burada bağlantı kuran istemciler tarafından gelen istekler bir kuyruğa alınarak birden fazla istemci tarafından gelen isteklere cevap verme işlemi gerçekleştirilmiştir. Birden fazla kullanıcı bağlanabilmesinden dolayı, cihaz multimaster olarak çalışabilmektedir [4]. Tasarlanan cihaz Modbus TCP ve Modbus RTU protokollerini desteklemektedir. Cihaz Modbus TCP protokolünde Slave olarak çalışırken, Modbus RTU protokolünde Master olarak çalışmaktadır.



Şekil 1.2. Yapılan çalışmanın blok diyagramı

Tezin geri kalan kısmı şu şekilde organize edilmiştir: 2. Bölümde gömülü sistemler, gömülü Linux işletim sistemi kullanma aşamalarına değinilmiştir 3. Bölümde kablosuz haberleşme yöntemi ve yapılan çalışmada kullanılan 3G modem kullanımı üzerinde durulmuştur. Yapılan çalışmada kullanılan Modbus protokolü ile ilgili detaylar 4. Bölümde anlatılmıştır. 5. Bölümde Modbus gateway cihazı tasarlanırken kullanılan donanımlar, yazılım teknikleri ve yazılım algoritmaları açıklanarak sistemin çalışır hale getirilme aşamaları açıklanmıştır. Sistemin çalıştırılması ve test sonuçları ile bölüm sonlandırılmıştır. 6. Bölümde de sonuçlar ve gelecek çalışmalar ile ilgili yapılabilecekler açıklanmıştır.

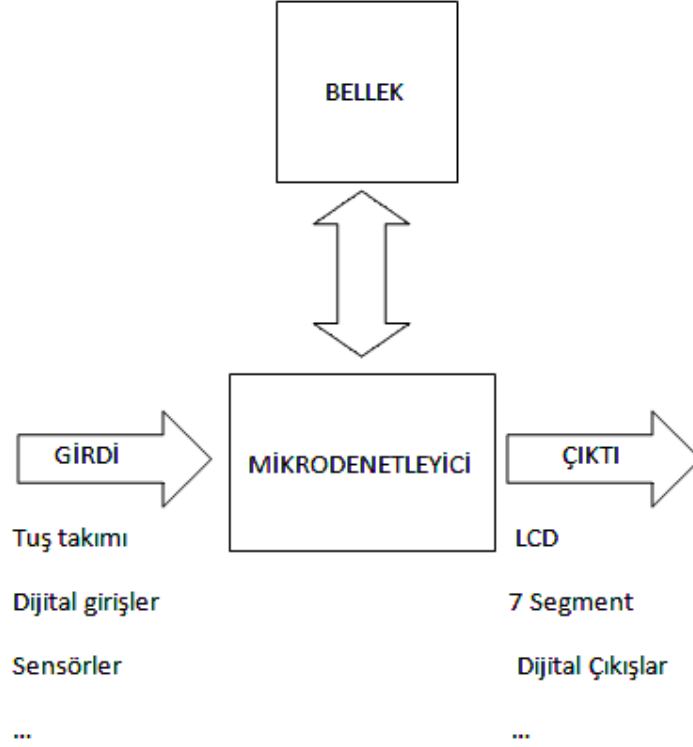
2. GÖMÜLÜ LINUX SİSTEMLERİ

Tezin bu kısmında, Linux tabanlı 3G haberleşme özellikli mesaj önceliklendirme mekanizmalı Modbus gateway tasarımı çalışmaları sırasında kullanılan gömülü sistemler, gömülü Linux işletim sistemleri, ve gömülü Linux uygulamalarının geliştirilmesinden bahsedilecektir. Gömülü Linux uygulamalarının geliştirilmesi başlığında, Linux çekirdeğinin derlenmesi ve cihaza yüklenmesi, oluşturulan uImage dosyasının cihaza yüklenmesi için kullanılan TFTP (Trivial File Transfer Protocol) sunucusunun kurulumu, programın platform üzerinde çalıştırılması için kullanılan NFS (Network File System) sunucusunun kurulumu ve çalışmaya hazır hale getirilen cihaz üzerinde uygulamaların çalıştırılması işlemleri anlatılacaktır.

2.1. Gömülü Sistemler

Gömülü sistemler, tasarlandıktan sonra yıllarca hatasız çalışabilen, üretim, kalite kontrolü gibi endüstriyel alanlarda veya şifreleme, uçuş kontrolü, görüntüleme gibi birçok alanlarda kullanılan, kişisel bilgisayarlardan farklı olarak tasarımcısı tarafından tanımlanmış görevleri yerine getiren sistemlerdir. Genel amaçlı tasarlanmış kişisel bilgisayarlarda kullanıcı tarafından istenen yazılımlar çalıştırılabiliyorken, gömülü sistemler sistem yazılımında tanımlanan kendisine özel görevleri yerine getirmektedir. Sistemin yerine getirmesi istenen görevler, firmware adı verilen ve cihazın içinde gömülü bulunan yazılımlar tarafından tanımlanır. Şekil 2.1'de, gömülü bir sistemin genel yapısı verilmiştir. Şekil 2.1'de görüldüğü gibi gömülü sistemler bellek, mikrodenetleyici, giriş ve çıkış birimlerinden oluşmaktadır [24]. Bir gömülü sistemde mikrodenetleyiciler veya sayısal sistemler sistemin çekirdeğini oluştururlar. Mikrodenetleyiciler, ROM ya da flash bellek çipleri içerisinde gömülü olarak bulunan yazılımı okuyarak, programlanan şekilde sistemi çalıştırmaktadır. Gömülü sistemlerin giriş birimlerinde, fiziksel bilginin elektrik sinyallerine dönüştürülmesi ile mikrodenetleyici tarafından dış ortamdaki butona basılması, hareket sensöründen hareketin algılanması, sıcaklık sensörü ile ortamın sıcaklığının okunması gibi bilgiler yorumlanmaktadır. Çıkış birimlerinde ise dış ortamdan elde edilen bilgiler doğrultusunda belirlenen senaryolarda elektriksel

işaretler fiziksel olarak dış ortama aktarılmakta veya çeşitli haberleşme yöntemleri ile gerekli yerlere bilgi olarak gönderilebilmektedir.



Şekil 2.1. Gömülü sistemin genel yapısı

2.2. Gömülü Linux İşletim Sistemi

Linux, açık kaynak kodlu olarak kişisel bilgisayarlar, endüstriyel bilgisayarlar ve mobil cihazlar gibi platformlarda çalışabilen, ilgili ve meraklı kişiler tarafından geliştirilmekte olan işletim sistemidir. Geliştirilen Linux işletim sisteminin her aşaması internette yayınlandığından, kullanıcılar tarafından sürekli test edilebilmekte ve tespit edilen hata ve eksikler, dünyanın çeşitli yerlerinde ki gönüllü kullanıcılar tarafından düzeltilmekte ve geliştirilmektedir. Linux gün geçtikçe daha çok kullanıcıya ulaşarak son yıllarda hızlı bir gelişme göstermiştir.

Değişik kurum ve kuruluşlar Linux sistemi ve uygulama yazılımlarını bir araya getirerek, GNU/Linux dünyasına katkılarıyla öne çıkan Ubuntu, Debian, Slackware,

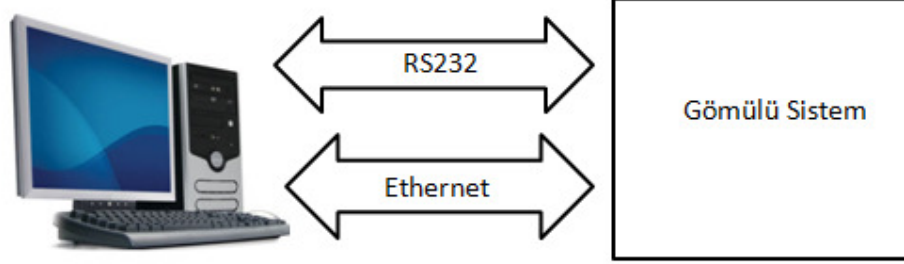
Mandriva Linux (Mandrake), Pardus, Red Hat, Fedora Core, SUSE gibi dağıtımların kullanımını yaygınlaştırmışlardır [13]. Linux işletim sisteminin kullanımının yaygınlaşması ve aynı zamanda taşınabilir elektronik cihazlar, medya oynatıcılar ve diğer tüketici elektroniği cihazların kullanımının artması, Linux işletim sisteminin gömülü olarak taşınabilir cihazlarda kullanılmasını sağlamıştır. Gömülü Linux, cep telefonları, PDA'lar, taşınabilir elektronik cihazlar, endüstriyel cihazlar, medya oynatıcılar ve diğer tüketici elektroniği cihazları gibi cihazlarda kullanılan Linux işletim sistemine verilen addır.

Linux gömülü işletim sisteminin, VxWorks, QNX, LynxOS, Windows CE, Windows NT Embedded, Palm OS gibi diğer gömülü işletim sistemlerine göre bir çok avantajı bulunmaktadır [25]. Linux işletim sisteminin açık kaynak kodlu olması en önemli avantajıdır. Linux'un açık kaynak kodlu olmasından dolayı kullanıcı tarafından sistem çok özel ihtiyaçlara uyarlanabilir ve yeni özellikler eklenebilir şekilde geliştirilebilmektedir. Çekirdek kodu GPL lisansı altında dağıtılmaktadır ve kodun içeriğine bakılıp yapılan işlemler görülebilmektedir. Güvenlik açısından açıkların olabileceği kısımlar kullanıcı tarafından değiştirilme özelliğine sahiptir.

Linux işletim sisteminin açık kaynak kodlu olmasının yanı sıra, genellikle ücretsiz olması, 10 yıldan fazla bir süredir pek çok cihazın içinde kullanılmakta olmasından dolayı olgun ve kararlı olması, modüler olma özelliğinden dolayı çekirdeğe çalışma zamanında fonksiyonların eklenip çıkarılması, iyi bir destek ağına sahip olması, birçok değişik donanım platformlarını ve aygıtlarını desteklemesi, Ethernet, Profibus, CAN, RS485 gibi haberleşme protokolü ve yazılım standartlarını desteklemesi gibi özellikleri birçok gömülü sistem uygulamasında tercih edilmesini sağlamıştır [2].

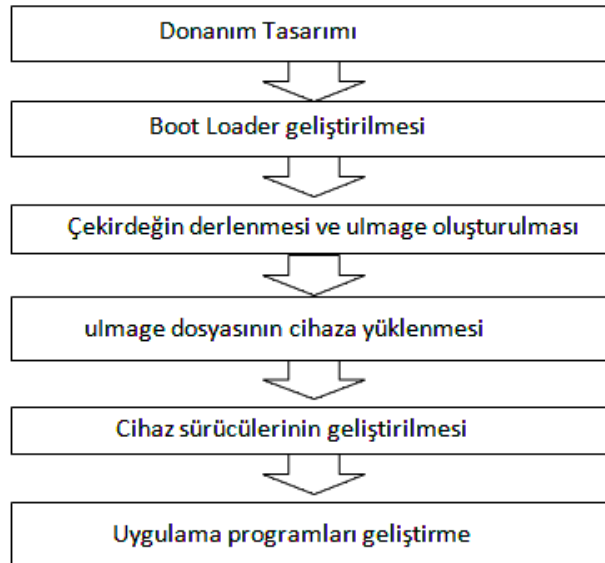
2.3. Gömülü Linux Uygulamalarının Geliştirilmesi

Gömülü Linux uygulamaları geliştirilirken bir host bilgisayar ve gömülü sistem kullanılmaktadır. Şekil 2.2’de host olarak kullanılan bilgisayar ile gömülü sistem bağlantısı görülmektedir.



Şekil 2.2. Host bilgisayar ile gömülü sistem bağlantısı

Bir gömülü Linux uygulaması geliştirilirken takip edilen adımlar, Şekil 2.3’te verilmiştir.



Şekil 2.3. Gömülü Linux geliştirme adımları

Gömülü Linux geliştirilirken ilk önce üzerinde çalışma yapılacak donanımın tasarlanması gerekmektedir. Donanım tasarımı gerçekleştirilirken sistem gereksinimleri ve geliştirme aşamaları göz önünde bulundurularak gerekli donanım aygıtları (USB, Ethernet, RS232...) eklenmelidir. Donanım tasarımından sonra bootloader geliştirilmelidir. Boot loader yükleyici anlamına gelmektedir. Bilgisayar sistemi ilk açıldığında işletim sistemini başlatan ön yükleyicidir. Bootloader tasarımı gerçekleştirildikten sonra çekirdek derleme işlemi gerçekleştirilir. Çekirdek, programlar ve donanımlar arasında iletişimi sağlayan işletim sisteminin kalbi olarak tanımlanan birimdir. Çekirdek derlendikten sonra oluşturulan “uImage” dosyası cihaza yüklenir. uImage dosyası cihaza yüklendikten sonra cihaz üzerinde sürücü ve uygulamalar geliştirilebilmektedir. Donanım bileşenlerinin düzgün bir şekilde çalışabilmesi için gerekli yazılımlara sürücü denir. Tasarımcı tarafından gömülü sistemin yerine getirmesi gereken görevlerin belirlendiği yazılımlar ise cihaz üzerinde geliştirilen uygulamalardır.

Yapılan tez çalışmasında, geliştirilen gömülü sistem için Mikrodev firmasının tasarlamış olduğu geliştirme kiti kullanıldığından, geliştirme aşamalarından donanım tasarımı ve bootloader geliştirilmesi işlemlerinden bahsedilmeyecektir. Bu konularda ayrıntılı bilgi kullanma kılavuzunda yer almaktadır [29]. Bu tez çalışmasında sürücü geliştirme işlemi ile ilgili olarak 3G modem için USB sürücüsü kurulumu gerçekleştirilmiştir. 3G modem için USB sürücüsünün kurulumu Bölüm 3’te anlatılmıştır. Gömülü sistem geliştirmenin diğer aşamaları olan çekirdeğin derlenmesi ve uImage oluşturulması, uImage dosyasının cihaza yüklenmesi ve uygulama programları geliştirerek cihaz üzerinde çalıştırma işlemleri detaylı bir şekilde Bölüm 2.3.2’de anlatılacaktır. Öncelikle bu aşamaların gerçekleştirilebilmesi için host olarak kullanılan bilgisayar üzerinde ELDK (Embedded Linux Development Kit), TFTP sunucusu ve NFS sunucusu yazılımlarının kurularak kullanılabilecek şekilde ayarlanması gerekmektedir. Host bilgisayar üzerinde kurulması gereken bu uygulamalar Bölüm 2.3.1’de anlatılacaktır.

2.3.1. Gömülü Linux Geliştirmek için Gerekli Uygulamaların Kurulumu

Gömülü Linux geliştirirken host olarak kullanılan, üzerinde uygulama yazılımları geliştirilen bilgisayarda ELDK, TFTP sunucusu ve NFS sunucusu uygulamalarının kurularak kullanılabilecek şekilde ayarlanması gereklidir. ELDK çekirdeğin ve uygulamaların derlenmesi işlemlerinde kullanılmaktadır. TFTP sunucusu, derlenen çekirdek dosyasından elde edilen uImage dosyasının gömülü sisteme yüklenmesi işleminde kullanılır. NFS sunucusu ise host bilgisayar üzerinde geliştirilen uygulamaların gömülü sistem üzerine taşınması veya gömülü sistem üzerinde çalıştırılması işlemlerinde kullanılmaktadır.

2.3.1.1. Gömülü Sistem Geliştirme Kiti (ELDK)

ELDK, Linux çekirdeğini kullandığımız platformlarda derleme işlemi için kullanılmaktadır. ELDK'nın kurulum işlemleri aşağıda anlatıldığı gibidir.

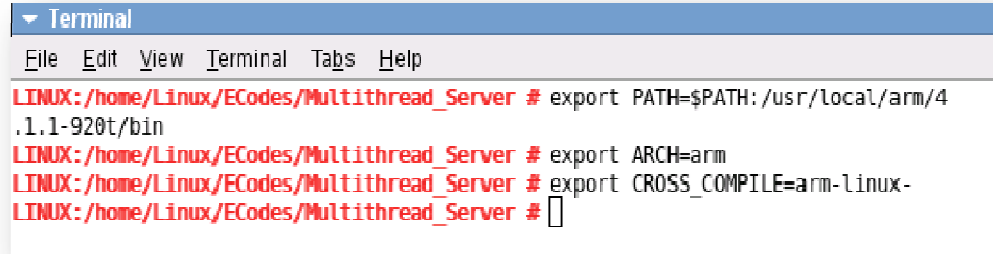
<ftp://ftp.denx.de/pub/eldk/4.1/arm-linux-x86/iso/> adresinden .iso dosyası indirilir. Daha sonra .iso uzantılı dosya istenilen dosya konumuna kopyalanır. Belirlenen dosya konumunda aşağıdaki komutlar kullanılarak ELDK'nın kurulumu gerçekleştirilir.

```
> linuxhost > mkdir eldk
> linuxhost > mkdir install_mount
> linuxhost > cd eldk/
> linuxhost/eldk > su
> linuxhost/eldk # mount -o loop,exec ../arm-2007-01-21.iso../install_mount/
> linuxhost/eldk # exit
> linuxhost/eldk > ../install_mount/install arm
```

ELDK'yı kullanmak için ELDK'nın bulunduğu dosya dizinine gidilir ve aşağıdaki komut yazılarak ELDK başlatılır.

```
> ./eldk_init arm
```

ELDK'nın kurulduğu dosya konumu komut satırına yazılarak, çekirdek derleme işleminde veya yazılan kodların derlenmesi işleminde çapraz derleme (cross compile) işlemi gerçekleştirilir [15]. Şekil 2.4'te çapraz derleme işlemi için ELDK'nın dosya konumu verilmiştir.



```

▼ Terminal
File Edit View Terminal Tabs Help
Linux:/home/Linux/ECodes/Multithread_Server # export PATH=$PATH:/usr/local/arm/4
.l.1-920t/bin
Linux:/home/Linux/ECodes/Multithread_Server # export ARCH=arm
Linux:/home/Linux/ECodes/Multithread_Server # export CROSS_COMPILE=arm-linux-
Linux:/home/Linux/ECodes/Multithread_Server #

```

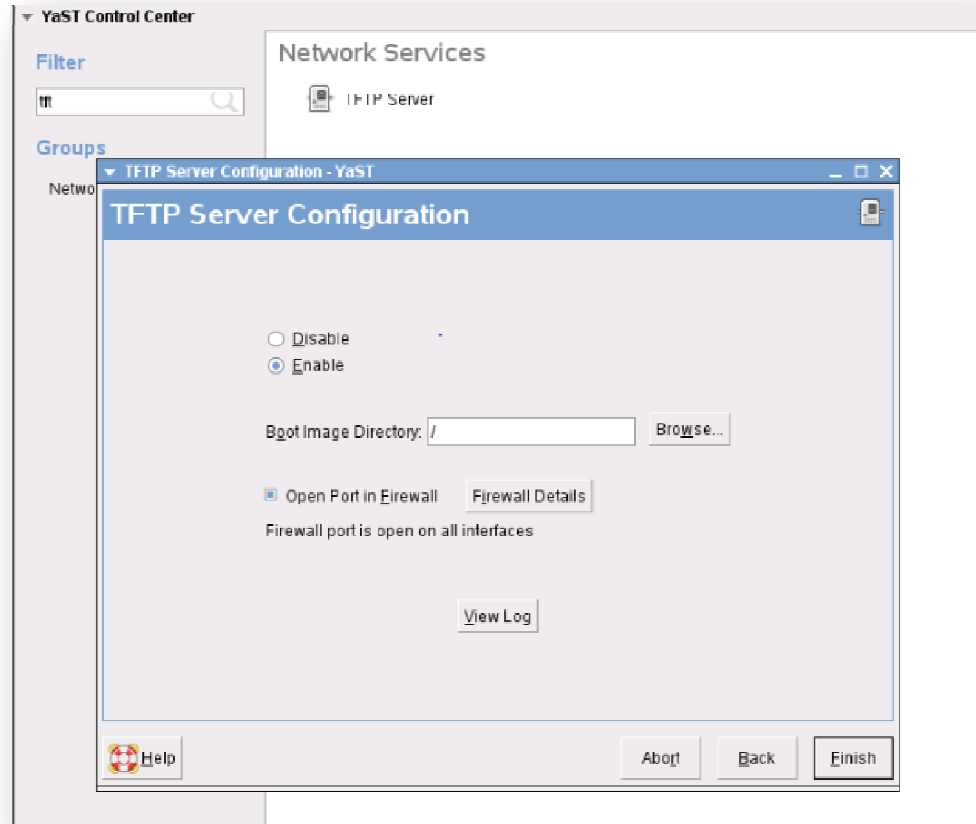
Şekil 2.4. Çapraz derleme (Cross Compile) işlemi

2.3.1.2. TFTP Sunucusu

TFTP (Trivial File Transfer Protocol), FTP'nin işlevsel özelliklerine benzer olarak 1980 yılında tanımlanmış, basit bir dosya transfer protokolüdür. Çok az bellek tüketmelerinden dolayı, bellek ihtiyacının çok fazla olmadığı yönlendirici olarak kullanılan bilgisayarların önyüklemesi gibi uygulamalarda tercih edilmektedir. Şifreleme (Encryption) ve yetkilendirme (Authentication) mekanizmalarının olmaması güvenlik açısından kritik uygulamalarda kullanılabilirliğini azaltmaktadır [27]. TFTP protokolü günümüzde genellikle ağ üzerinde host makineler arasındaki küçük boyutlu dosyaların transfer işleminde kullanılmaktadır [14].

TFTP Sunucusu, router hafızasının, başlangıç (startup) ve çalışan ayarlar (running configuration) dosyalarının yedeklerini almak için TFTP protokolünü kullanan bir sunucudur [14]. Yapılan çalışmada oluşturulan uImage dosyasının işlemciye yüklenmesi işleminde TFTP sunucu kullanılmıştır. Şekil 2.5'te görüldüğü gibi TFTP sunucu kurulumu yapılarak, uImage dosyasının kopyalanacağı dosya

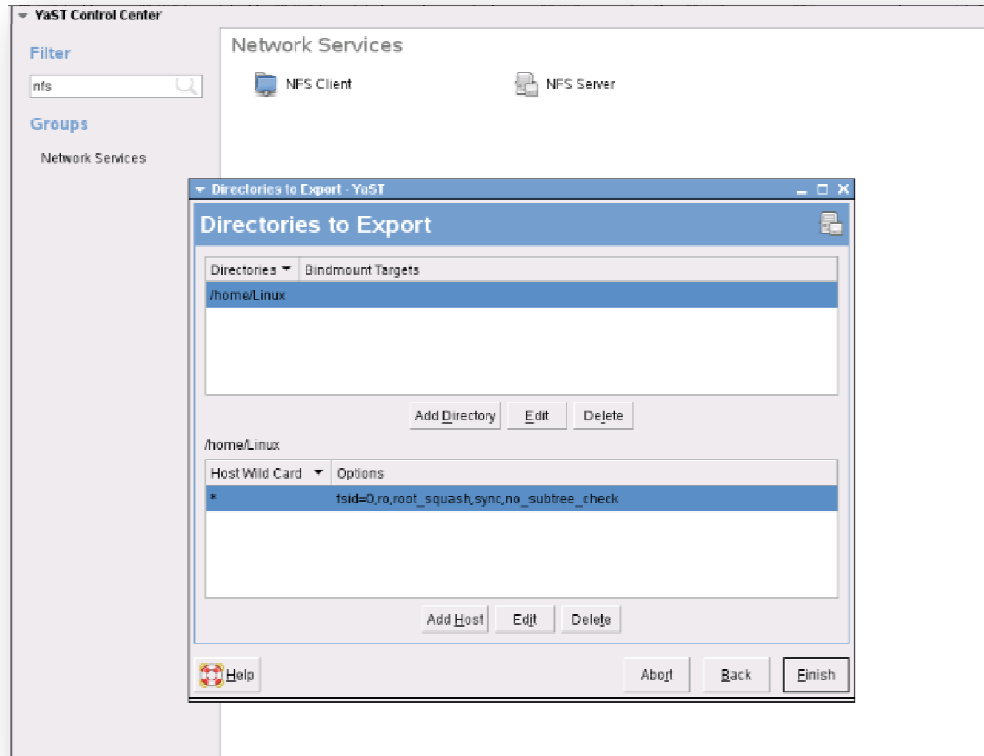
konumu belirlenir. Gömülü sistem tarafından dosyanın çekileceği dosya konumuna izin verilir. Daha sonra TFTP sunucusu başlatılır.



Şekil 2.5. TFTP sunucu kurulumu

2.3.1.3. NFS Sunucusu

NFS (Network File System) uzak Linux sunucularındaki dizinleri yerel sunucuda bir dizine “mount” ederek dosya ve dizin paylaşılması için geliştirilmiştir. Şekil 2.6’da NFS sunucunun dosya konumunun belirlenmesi işlemi ve gerekli ayarların yapılması işlemi görülmektedir.



Şekil 2.6. NFS sunucusunun aktifleştirilmesi ve ayarlarının yapılması

NFS sunucu kurulduktan sonra bilgisayarın ip adresi girilerek belirlenen dosyaya mount işlemi gerçekleştirilir. NFS sunucu ile host bilgisayardaki bir dosya gömülü sistemdeki bir dosyaya mount edilmektedir. Bu şekilde host bilgisayarda gerçekleştirilen çalışmaları gömülü sistem üzerinde çalıştırma imkanı sağlanmaktadır. Bu işlemi gerçekleştirmek için “ mount -o nolock 192.168.2.3:/home/Linux /mnt” komutu kullanılmaktadır. Bu komutta host bilgisayarın “/home/Linux” dosya konumu gömülü sistemdeki “/mnt” dosya konumunun altına mount edilmiştir. Bu şekilde NFS sunucu tarafından gömülü sistemin /mnt dosyasından host bilgisayarın /home/Linux dosyasına erişim izni verilmektedir.

2.3.2. Linux Çekirdeğinin (Kernelinin) Derlenmesi

Yazılım ile donanım arasındaki haberleşmeyi sağlayan birime çekirdek denir. Şekil 2.7’de görüldüğü gibi yazılım ile donanım arasındaki haberleşme, aygıt sürücülerini sayesinde çekirdek üzerinden sağlanmaktadır. Örnek olarak USB aracılığıyla 3G modem haberleşmesinin sağlanması verilebilir.



Şekil 2.7. Linux çekirdeğinin yazılım ve donanım arasındaki konumu

Sistemin çalışmasından, kaynakların etkin kullanımından ve kaynak erişimlerinin düzenlenmesinden, bellek işlemci yöntemi, donanım erişim yöntemi, süreçlerin yönetimi, giriş/çıkış işlemlerinin yönetimi gibi işlemler çekirdek tarafından yapılabilmektedir. Linux işletim sistemi kullanıcılarına çekirdek ile oynama, serbestçe değiştirme, yeniden derleme ve güncelleme imkanı sağlamaktadır. Çekirdeği derleyerek rol yükleme veya kaldırma işlemleri gerçekleştirilmektedir. Linux işletim sistemi kullanırken yeni çekirdeğin uygun olmayan veya gereksiz olan donanım desteğini kaldırabilir, yazılım desteğini kaldırabilir ya da modül olarak eklenebilmektedir. Bu şekilde sadece gerekli donanım ve yazılımları aktif olarak kullanarak daha küçük ve hızlı açılan bir çekirdeğe sahip olmak mümkündür. Yeni çekirdek kaynak kodlarını ftp.kernel.org veya ftp.linux.org adreslerinden ya da bu adreslerin Türkiye yansıısı olan ftp.linux.org.tr den indirilebilmektedir.

Yapılan çalışmada sistem için gerekli donanım ve yazılım araçları dikkate alınarak gömülü Linux işletim sistemi için çekirdeği derleme işlemi gerçekleştirilmiştir. Çekirdek derleme işlemi, bu tez çalışmasında kullanılan ARM

işlemcisi için anlatılacaktır. Öncelikle <http://kernel.org/> sitesinden kaynak kodu indirilmesi gerekir. Bu çalışmada, EP9302 işlemcisinin geliştirme kiti platformu için tavsiye edilen 2.6.20.4 versiyonlu kaynak kodu kullanılmıştır. Ayrıca <http://arm.cirrus.com/> sitesinden EP9302 için istenen yazılım ve donanım dokümanları bulunabilmektedir. EP9302 ile çalışabilecek çekirdek versiyonu ve yaması bu siteden indirilebilir. Çekirdek derlenirken takip edilen adımlar aşağıda sıralanmıştır.

Öncelikle çekirdek derleme işlemi için gerekli olan çekirdek versiyonu dosyası ve kullanılan işlemci için gerekli yama dosyası aşağıda belirtilen internet adreslerinden indirilmesi gerekir.

```
$ wget http:// http://arm.cirrus.com/files/index.php?path=linux%2Fpackages/

$ wget http://
http://arm.cirrus.com/files/index.php?path=linux%2Fleases%2Flinux-2.6%2F1.0.3/
```

Uyumlu çekirdek versiyonu ve EP9302 dosyaları indirildikten sonra sıkıştırılmış dosyalar açılır. Daha sonra yama işlemi aşağıdaki gibi gerçekleştirilir.

```
$ tar -xjf linux-2.6.20.4.tar.bz2
$ cd linux-2.6.20.4/ // bu dosyaya patch dosyası kopyalanır.
$ cd linux-2.6.20.4/patch -p1 linux1-0-3-9302.tar.bz2
```

EP9302 için yama işlemi gerçekleştirilirken aşağıdaki mesajlar ekrana gelmektedir.

```
patching file arch/arm/mach-ep93xx/core.c
patching file cs-e9302_kernel_config
patching file drivers/net/arm/ep93xx_eth.c
```

```

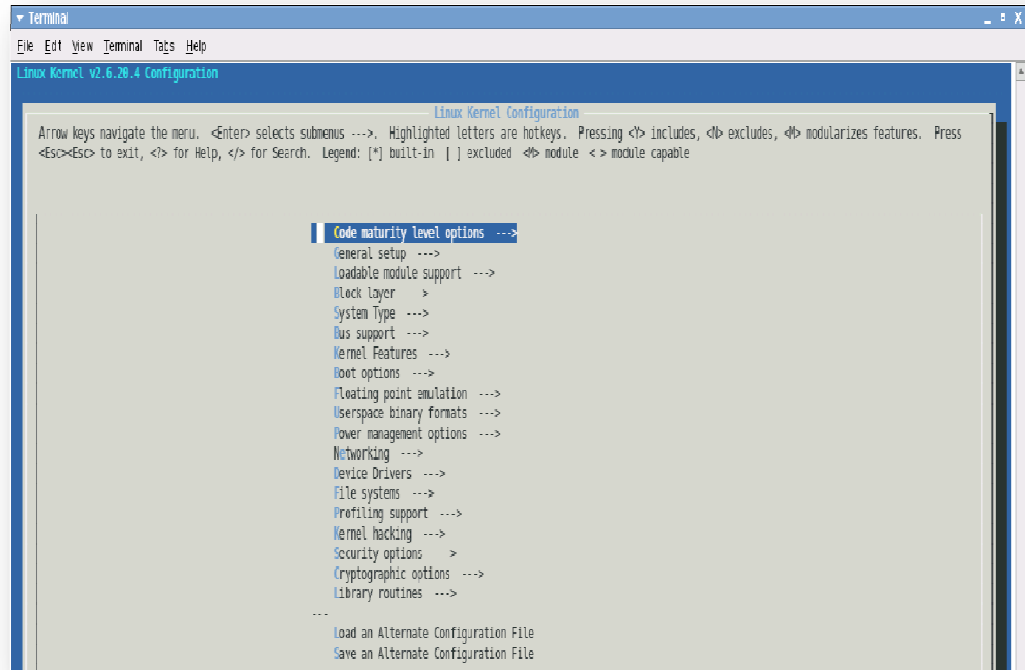
patching file drivers/net/arm/ep93xx_eth.h
patching file drivers/spi/Kconfig
patching file drivers/spi/Makefile
patching file drivers/spi/spi_ep93xx.c
patching file include/asm-arm/arch-ep93xx/ep93xx-regs.h

```

Daha sonra ELDK kurulu dizininin dosya konumu girilerek EP9302 için çekirdek derlenir ve “uImage” dosyası oluşturulur. Bunun için “make menuconfig” komutu kullanılır.

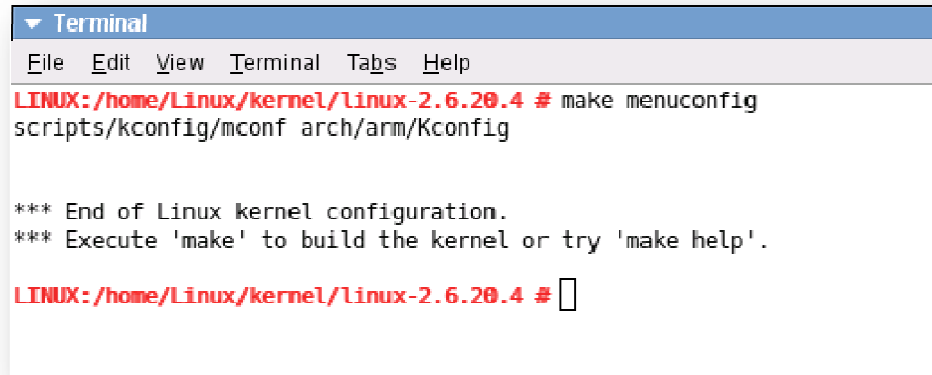
```
$ make menuconfig
```

Daha sonra Şekil 2.8’deki ekran gelmektedir. Gelen ekrandan gerekli ayarlar yapılır. Örneğin yapılan çalışma için EP9302 işlemcisinin seçilmesi, USB sürücüsünün modül olarak derlenmesi işlemleri gereken ayarlardır.



Şekil 2.8. Çekirdek konfigürasyon ayarlarının yapılması

İstenen ayarlar gerçekleştirildikten sonra çekirdek ayarlarından çıkılır. Şekil 2.9'da konfigürasyon ayarlarından çıktıktan sonraki ekran görüntüsü verilmiştir.



```

▼ Terminal
File Edit View Terminal Tabs Help
LINUX:/home/Linux/kernel/linux-2.6.28.4 # make menuconfig
scripts/kconfig/mconf arch/arm/Kconfig

*** End of Linux kernel configuration.
*** Execute 'make' to build the kernel or try 'make help'.

LINUX:/home/Linux/kernel/linux-2.6.28.4 #

```

Şekil 2.9. Konfigürasyon ayarlarından sonra terminalin görüntüsü

İstenen ayarlar gerçekleştirildikten sonra ELDK'nın bulunduğu dosya konumu verilerek uImage oluşturma işlemi gerçekleştirilir.

```

$ export ARCH=arm
$ export PATH=$PATH:/usr/local/arm/4.1.1-920t/bin/
$ make // bu komut eğer çekirdek daha önce derlenmemişse kullanılır.
$ make uImage // bu komut ile uImage dosyası oluşturulur

```



```

LINUX:/home/Linux/kernel/linux-2.6.28.4 # make uImage
CHK include/linux/version.h
make[1]: `include/asm-arm/mach-types.h' is up to date.
CHK include/linux/utsrelease.h
CHK include/linux/compile.h
MODPOST vmlinux
Kernel: arch/arm/boot/Image is ready
Kernel: arch/arm/boot/zImage is ready
Image arch/arm/boot/uImage is ready
LINUX:/home/Linux/kernel/linux-2.6.28.4 #

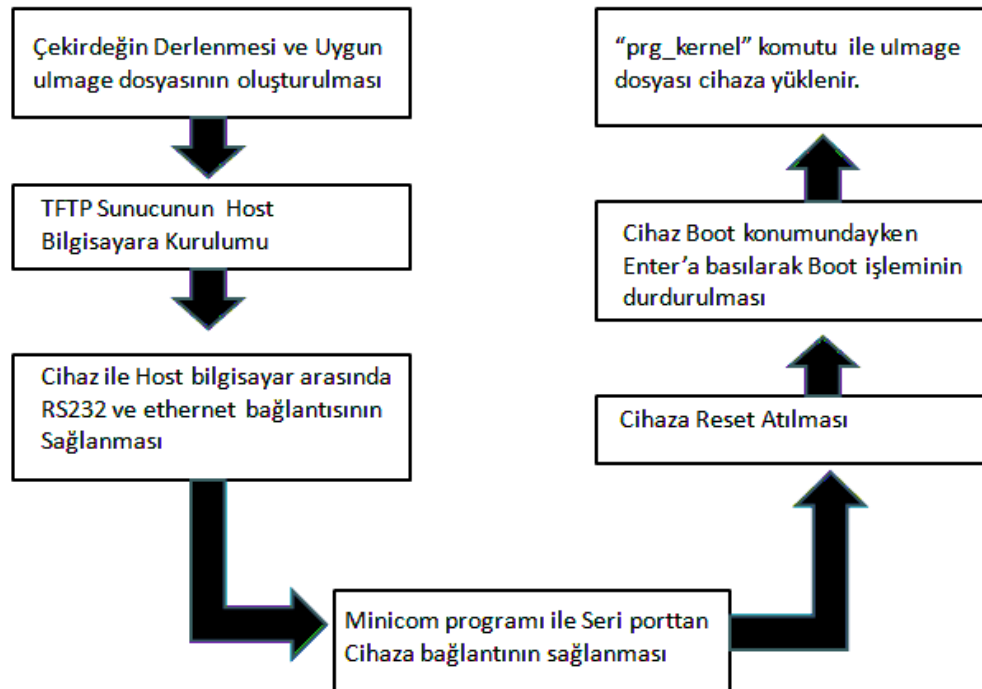
```

Şekil 2.10. uImage oluşturma

Şekil 2.10’da görüldüğü gibi uImage dosyası çekirdek dosyasının “arch/arm/boot/uImage” konumunda oluşmaktadır. uImage dosyasının cihaza yüklenilmesi için gerekli adımlar Bölüm 2.3.2’de anlatılacaktır.

2.3.3. UImage Dosyasının Modbus Gateway Cihazına Yüklenmesi

uImage dosyasının oluşturulmasından sonra cihaza yüklenmesi için takip edilen adımlar Şekil 2.11’de özetlenmiştir.



Şekil 2.11. uImage oluşturulması ve cihaza yükleme adımları

Şekil 2.11’de görüldüğü üzere, çekirdeğin derlenmesi ve uImage dosyasının oluşturulmasından sonra, oluşturulan bu dosyanın TFTP (Trivial File Transfer Protocol) sunucusu ile gömülü sisteme yüklenmesi gerekir.

Üzerinde geliştirme işlemleri yapılan host bilgisayar ile gömülü sistem arasında RS232 seri portu ve ethernet üzerinden bağlantı sağlandıktan sonra, uImage dosyası cihaza yüklenebilir.

Cihaza uImage dosyasını yükleyebilmek için seri port üzerinden cihaza minicom programı ile bağlantı sağlanır. Komut satırına minicom yazıldığında, daha önceden ayarlanan parametreler ile minicom programı başlatılmış olur ve Şekil 2.12’de verilen ekran görüntüsü elde edilir.

```

Terminal
File Edit View Terminal Tabs Help

Welcome to minicom 2.2

OPTIONS: I18n
Compiled on Sep 22 2007, 00:14:00.
Port /dev/ttyS0

Press CTRL-A Z for help on special keys

BusyBox v1.1.3 (2007.07.23-13:22+0000) Built-in shell (ash)
Enter 'help' for a list of built-in commands.

/ # AT S7=45 S0=0 L1 V1 X4 &c1 E1 Q0
-sh: c1: not found
/ # -sh: AT: not found

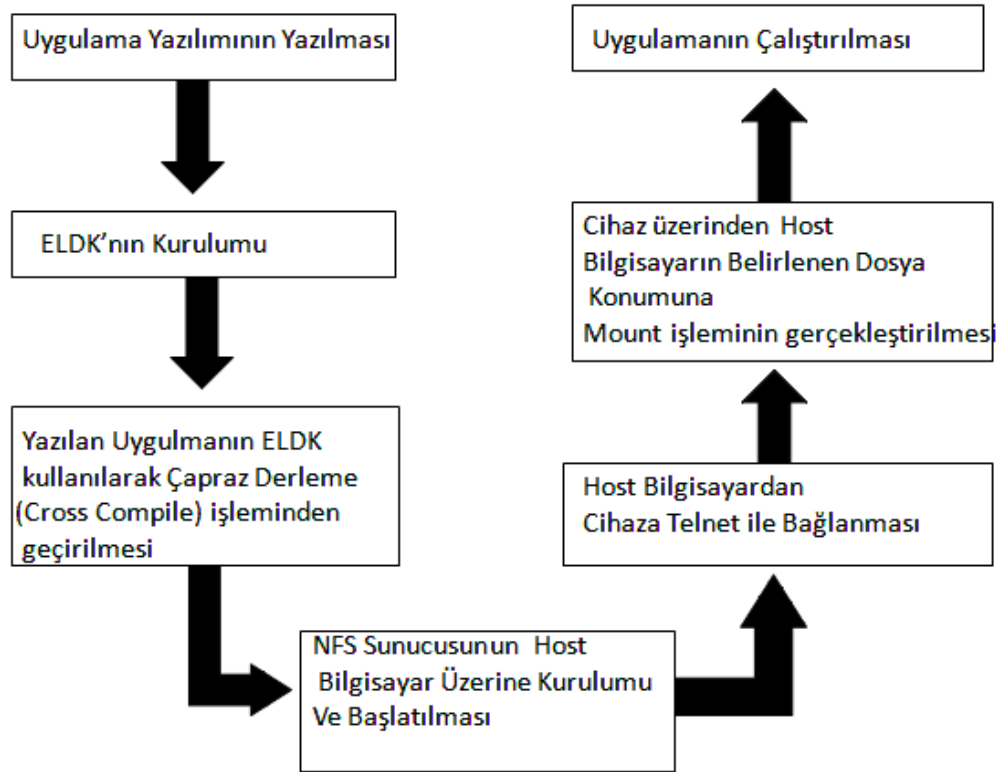
```

Şekil 2.12. Minicom ekran görüntüsü

Minicom açıldıktan sonra ilk olarak cihaza reset atılması gerekmektedir. Bunun için, cihaz boot konumunda iken Enter tuşuna basılarak boot konumunda durması sağlanır. Sonra “print” yazıldığında parametreler ekrana gelir. Bu parametrelerde istenen değişiklikler yapılarak, oluşturduğumuz uImage dosyasını TFTP sunucu kullanarak cihaza yüklenebilir. uImage dosyasını cihaza yüklemek için “prg_kernel” komutu kullanılır. Bu komuttan sonra uImage dosyasının cihaza yüklenme işleminin bitmesi ile platform uygulama geliştirmek için hazır hale getirilmiştir. Uygulama geliştirme, cihaz için derleme ve uygulamayı cihaz üzerinde çalıştırma işlemleri Bölüm 2.3.4’te anlatılacaktır.

2.3.4. Cihaz Üzerinde Uygulama Geliştirme ve Çalıştırma

uImage dosyası cihaza yüklenerek uygulama geliştirmeye hazır hale getirildikten sonra geliştirilecek uygulamaların kullanılan işlemciye göre derlenmesi ve geliştirilen uygulamaların NFS sunucusu kullanarak cihaz üzerinde çalıştırılması gerekmektedir. Uygulama geliştirme ve cihaz üzerinde çalıştırma aşamaları Şekil 2.12’de görülmektedir.



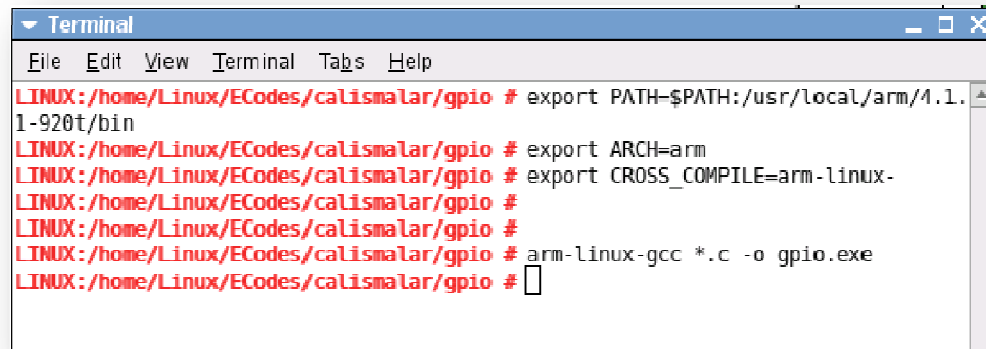
Şekil 2.13. Cihaz üzerinde uygulama çalıştırma adımları

Kullanıcı tarafından uygulama yazılımı yazıldıktan sonra daha önce kurulmuş olan ELDK sayesinde çapraz derleme yöntemi ile uygulamanın üzerinde çalışacağı işlemciye göre derlenme işlemi gerçekleştirilir. Yazılan kodların kullanılacak platform için derlenmesi için öncelikle dosya konumu olarak ELDK'nın kurulu olduğu dosya dizini gösterilir.

```
export ARCH=arm
export PATH=$PATH:/usr/local/arm/4.1.1-920t/bin/
```

Daha sonra “ arm-linux-gcc gpio.c -o gpio.exe “ denilerek yazılan C kodu ARM işlemcisi üzerinde çalışabilecek şekilde derlenmiş olacaktır. Şekil 2.14’te görüldüğü gibi “gpio.exe” programı Arm işlemcisi için çapraz derleme yöntemi ile derlenmiştir.

```
arm-linux-gcc gpio.c -o gpio.exe
```



```

Terminal
File Edit View Terminal Tabs Help
LINUX:/home/Linux/ECodes/calismalar/gpio # export PATH=$PATH:/usr/local/arm/4.1.1-920t/bin
LINUX:/home/Linux/ECodes/calismalar/gpio # export ARCH=arm
LINUX:/home/Linux/ECodes/calismalar/gpio # export CROSS_COMPILE=arm-linux-
LINUX:/home/Linux/ECodes/calismalar/gpio #
LINUX:/home/Linux/ECodes/calismalar/gpio #
LINUX:/home/Linux/ECodes/calismalar/gpio # arm-linux-gcc *.c -o gpio.exe
LINUX:/home/Linux/ECodes/calismalar/gpio #

```

Şekil 2.14. Programın Arm işlemcisi için derlenmesi

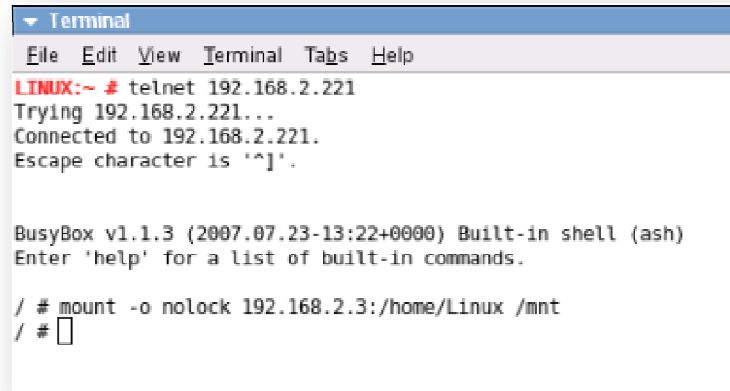
Cihaz üzerinde çalışabilecek uygulamayı hazır hale getirdikten sonra sıra uygulamanın cihaz üzerinde çalıştırılmasına gelmiştir. Derlenen bu “.exe” uzantılı uygulamanın işlemci üzerinde çalışabilmesi için aşağıda listelenen 5 işlem sırasıyla gerçekleştirilmelidir.

1. NFS sunucusunun host olarak kullanılan bilgisayara kurulur.
2. Host bilgisayardan Telnet ile cihaza bağlantı sağlanır. Telnet, internet ağı üzerindeki bir makineye uzaktaki bir makineden bağlantı sağlamak için geliştirilmiş bir TCP/IP protokolüdür.

3. Cihaz üzerinden Linux çalışan bilgisayarımıza mount işlemi gerçekleştirilir. Mount işlemi farklı makineler arasındaki dosyalara erişim izni verilerek dosya paylaşımı işlemlerinde kullanılmaktadır. Aşağıdaki komut ile gerçekleştirilir.

```
$ mount -o nolock 192.168.2.3/home/Linux /mnt
```

Şekil 2.15'te görüldüğü gibi host olarak kullanılan bilgisayara mount işlemi gerçekleştirilir. Şekil 2.16'da görüldüğü gibi “/home/Linux” dosya konumundaki dosyalar “/mnt” dosyasının altına mount edilmiştir.



```

Terminal
File Edit View Terminal Tabs Help
LINUX:~ # telnet 192.168.2.221
Trying 192.168.2.221...
Connected to 192.168.2.221.
Escape character is '^]'.

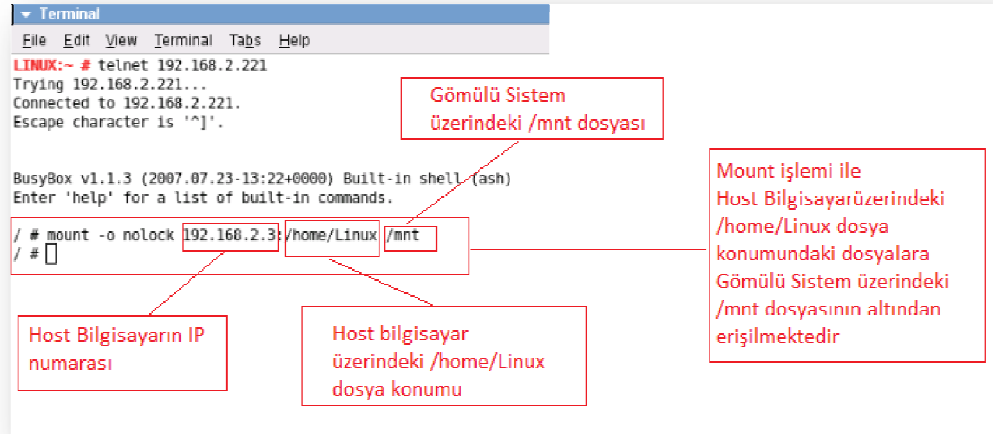
BusyBox v1.1.3 (2007.07.23-13:22+0000) Built-in shell (ash)
Enter 'help' for a list of built-in commands.

/ # mount -o nolock 192.168.2.3:/home/Linux /mnt
/ #

```

Şekil 2.15. NFS Sunucu ile host bilgisayara mount işleminin gerçekleştirilmesi

4. Daha önceden oluşturulmuş .exe uzantılı programın bulunduğu konuma gidilir.
5. Programın bulunduğu dosya konumunda “./gpio.exe” denilerek .exe dosyası çalıştırılır.



Şekil 2.16. Mount işleminin gerçekleştirilmesi

Yukarıda anlatıldığı gibi “mnt/Ecodes/calısmalar/gpio” konumundaki “gpio.exe” programı “./gpio.exe” işleminden sonra artık cihaz üzerinde çalıştırılabilir. Bu şekilde host bilgisayar üzerinde yazılan bir program çapraz derleme yöntemi ile derlendikten sonra gömülü sistem üzerinde çalıştırılmıştır.

3. 3G HABERLEŞME TEKNOLOJİSİ

Tezin bu bölümünde tasarlanan Modbus gateway cihazının tasarımında kullanılan 3G haberleşme teknolojisi ve gelişimi, kullanılan 3G modem ile ilgili teknik bilgiler, tasarlanan cihaz ile 3G modem haberleşmesinin sağlanması için gerekli işlemler ve 3G modem ile internete bağlanma işlemleri anlatılacaktır.

3.1. 3G Teknolojisi ve Gelişimi

Analog hücreli mobil telekomünikasyon sistemleri olarak bilinen 1G mobil teknolojileri, 1970 yılının sonlarına doğru kullanılmaya başlanmıştır. Mikrodalga ve yarı iletken teknolojilerinin gelişmesi ve 1G teknolojisinin kullanıcıların artan ihtiyaçlarına cevap vermekte yetersiz kalmasından dolayı, 1991 yılında sayısal veri haberleşmesine olanak sağlayan ve bu sayede daha yüksek ses kalitesi ve sesi ve verileri şifreleme imkanı sağlayan 2G teknolojisi kullanılmaya başlanmıştır. 2G teknolojisi ile beraber, 9.6 kbps (kilobits per second) veri taşıma kapasitesine ulaşılmıştır. İnternetin yaygınlaşması ile beraber hızla artan veri haberleşme talebini karşılayabilmek için taşıma kapasiteleri sırasıyla 28.8 kbps, 171.2 kbps ve 384 kbps olan HSCSD (High Speed Circuit Switched Data), GPRS (General Packet Radio Service) ve EDGE (Enhanced Data Rates in a GSM Environment) teknolojileri geliştirilmiştir. Mobil alanlarda yaygınlaşan internet kullanımı sebebiyle yeterli seviyede hızlı olmayan GPRS ve EDGE teknolojileri yerini, 2 Mbps hızında veri aktarma imkanı sağlayan 3G teknolojisine bırakmıştır.

3G teknolojisi ile paket anahtarlama verilerin daha iyi spektral verimlilikte aktarılması sağlanmıştır. Bu özellik sayesinde 3G sisteminde cihazlar bant genişliğini sadece veri alış verişi esnasında meşgul ederler. 3G için, ilk olarak 1992 yılında 1885-2025 MHz ile 2110-2200 MHz frekans bantları belirlenmesine rağmen, 3G teknolojisinin zaman içerisinde gelişmesi ile beraber artan ihtiyaçları karşılamak için 2000 yılında 806-960 MHz, 1710-1885 MHz ve 2500-2690 MHz frekans bantları da 3G uygulamaları için tahsis edilmiştir [27].

3G teknolojisinin hücresel bir ağ sistemi kullanmasından dolayı kapsama alanı ile ilgili meydana gelen sorunları çözmesi beklenen dördüncü nesil kablosuz telefon teknolojisi olan 4G teknolojisi geliştirilmiştir. 4G ile bağlantı hızı cep telefonlarında 100 Mbps, Wi-Fi ağlarda 1Gbps hızına ulaşmaktadır. 4G ile kullanıcılara daha çok uçtan uca IP çözümü ile daha yüksek veri hızlarında haberleşmeyi herhangi bir zamanda herhangi bir yerde sağlaması amaçlanmaktadır.

3.2. Modbus Gateway Cihazının 3G Modem ile İnternet Bağlantısının Sağlanması

Yapılan çalışmada Modbus gateway cihazının kablosuz olarak internet bağlantısı 3G modem kullanılarak gerçekleştirilmiştir. Şekil 3.1’de görüldüğü gibi 3G modem kullanılarak sunucu modunda çalışan Modbus gateway cihazına istemci tarafından cihazın IP (188.57.178.227) ve port (502) numarasına bağlantı sağlanabilmektedir. Modbus gateway cihazı ile 3G modem Şekil 3.1’de görüldüğü gibi USB(Universal Serial Bus) ile bağlanmıştır.



Şekil 3.1. 3G modem ile internet bağlantısının sağlanması

3G modem olarak Simcom firmasının Sim5218 modemi kullanılmıştır. Bölüm 3.2.1’de yapılan çalışmada kullanılan Sim5218 3G modeminin ve geliştirme kitinin özellikleri anlatılacaktır. Bölüm 3.2.2’de ise Modbus gateway cihazı üzerinde 3G

modem ile haberleşmenin sağlanması için USB sürücüsünün kurulumu ve aktifleştirilmesi işlemi anlatılacaktır.

3.2.1. Sim5218 3G Modem

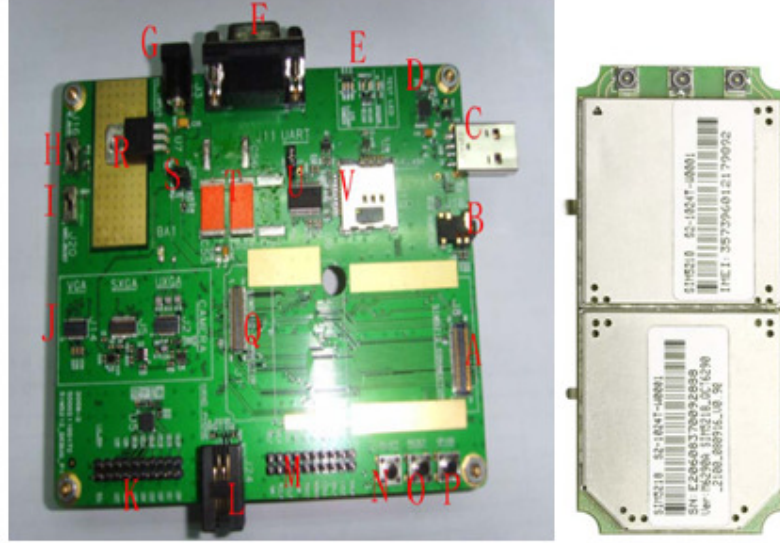
Yapılan çalışmada, Modbus gateway cihazının 3G haberleşmesini sağlamak için Simcom firmasına ait Sim5218 modemi kullanılmıştır. Sim5218 modem serisi Tri-Band/single-Band HSPA (High Speed Packet Access) / WCDMA (Wideband Code Division Multiple Access) ve Quad-Band GSM/GPRS/EDGE modül çözümleri sunan 7.2 Mbps indirme hızı ve 5.76 Mbps yükleme hızı imkanı sağlamaktadır. UART (Universal Asynchronous Receiver/Transmitter), USB2.0 yüksek hızlı haberleşme imkânı sağlayan ara yüzlere sahiptir. Sim5218 modemnin sahip olduğu özellikler Tablo 3.1’de belirtilmiştir [11].

Tablo 3.1. Sim5218 modemnin özellikleri

Genel Özellikler	• HSDPA 7.2Mbps • HSUPA 5.76Mbps • UMTS/HSDPA/HSUPA 900/1900/2100MHz • Quad-Band GSM 850/900/1800/1900MHz • GPRS multi-slot sınıf 12 • EDGE multi-slot sınıf 12 • UMTS/HSDPA 3GPP release 5 • UMTS/HSUPA 3GPP release 6 • GSM 3GPP release 99 • Çıkış gücü: UMTS 2100/1900/850: 0.25W • GSM850/GSM900: 2W • DCS1800/PCS1900: 1W • AT Komutlarla kontrol • Besleme voltaj aralığı: 3.4V~ 4.2V • Sıcaklık Aralığı: -15°C to +55°C • Saklama Sıcaklığı : -40°C to +85°C
-------------------------	---

	Boyut: 58 ×26 ×4.5 mm
GPS Özellikleri	A-GPS modu: MS-Tabanlı, MS-Destekli
Veri Transferi ile İlgili Özellikler	- HSDPA: Max. 7.2Mbps (DL) UE Kategorisi: 4, 6, 8, 12 - HSUPA: Max. 5.76Mbps (UL) - HSUPA+HSDPA: Max. 2.0Mbps (UL)+3.6Mbps (DL) - WCDMA: Max.384Kbps (DL), Max.384Kbps (UL) - EDGE Class: Max. 236.8Kbps (DL),Max.118Kbps (UL) - GPRS: Max. 85.6Kbps (DL), Max.42.8Kbps (UL) - CSD: GSM Veri aktarım hızı 14.4Kbps - WCDMA veri aktarma hızı 57.6Kbps
Ses ile İlgili Özellikler	- Ses kodekleri HR (Half Rate), FR (Full Rate), EFR (Enhanced Full Rate) - AMR (Adaptive Multi-Rate) - DTMF (Dual-tone multi-frequency signaling)
Görüntülü Görüşme İçin Gerekli Özellikler	- DTMF on H245 - Standart WCDMA 64kbps (CS) Görüntülü arama
Belgeler	CE & FCC

Bu çalışma sırasında kullanılan Sim5218 geliştirme modülü Şekil 3.2'de gösterilmiştir.



Şekil 3.2. Sim5218 ve geliştirme modülü

Şekil 3.2’de geliştirme modülünün üzerinde harflerle işaretlenmiş birimlerin karşılıkları Tablo 3.2’de verilmiştir.

Tablo 3.2. 3G modem geliştirme kiti birimleri

A	SIM5210/5218 modül arayüzü
B	Kulaklık Çıkışı
C	USB arayüzü
D	USB besleme DC-DC
E	GPIO led
F	AT Komutları için UART Arayüzü
G	DC besleme girişi
H	RF Kontrol Anahtarı
I	Seri portu kapatma Anahtarı
J	Kamera Arayüzü
K	IIC genişletilebilir IO arayüzü
M	Ses & GPIO test arayüzü
N	SIM5210/5218 modemleri için Güç Aç/Kapat
O	RESET tuşu
P	GPIO0 tuşu
R	DC-DCLM317
S	Besleme seçme pini
T	Kapasite
U	RS232
V	SD&SIM kart arayüzü

Sim5218 modeminin fiziksel ara yüzü mobil uygulamalar için 70 adet pin kullanıcının tasarımında kullanabilmesi için eklenmiştir. Bu pinler modem ile kullanıcının tasarımı arasında RF anten haricindeki bütün donanımsal arayüzleri barındırmaktadır.

Bu donanım arayüzleri aşağıdadır [11].

- Veri haberleşme portu olarak seri port (RS232) ve USB 2.0 alternatif olarak kullanılabilir.
- USIM arayüzü. Sim kart desteği:3V&1.8V
- Güç Açma/Kapatma, Reset Sinyali gönderme

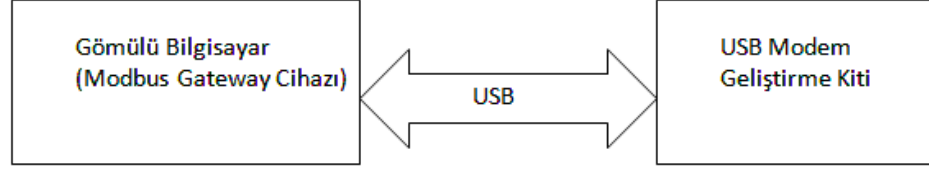
- RTC
- 6 GPIO: 1 adet kesme, 1 adet uçuş modu, 1 adet durum ledi, 2 adet çıkış kontrol, 1 adet giriş için, Ayrıca PCM girişi olarak da kullanılabilir.
- 3 adet ses kanalı, 2 adet mikrofon girişi, 3 adet ses çıkışı barındırmaktadır. Bunlar AT komutlarla sağlanabilir.
- Kamera arayüzü desteklenmektedir
- I2C arayüzü desteklenmektedir
- Bir adet ADC arayüzü
- 4 bit SD kart arayüzü
- Bir adet PCM arayüzü

Sim5218 modeminde TCP/IP protokolü entegre olarak bulunmaktadır. Kullanıcılar TCP/IP protokolü sayesinde AT komutları kullanarak internet kullanımını gereken işlemlerde 3G modemi rahatlıkla kullanabilmektedirler. Sim5218 modemimde UART ve USB2.0 olmak üzere iki adet haberleşme arayüzü bulunmaktadır. USB arayüzü 5 adet sanal porta eşlenebilir. Bu sanal portlar “SIMTECH USB Modem”, “SIMTECH NMEA Device”, “SIMTECH ATCOM Device”, “SIMTECH Diagnostic interface” ve “SIMTECH Wireless Ethernet Adapter. “SIMTECH USB Modem” ve “SIMTECH ATCOM Device” AT komutlarına cevap vermektedir. Bu portlardan herhangi birini aktif bir şekilde kullanabilen host cihazlar ile Sim5218 3G modemi USB’den haberleşme sağlayabilir.

3.2.2. Sim5218 3G Modemi ile Cihazın İnternet Bağlantısının Sağlanması

Bu çalışmada, host olarak kullanılan gömülü bilgisayar, internet bağlantısı için kullanılan Sim5218 3G modeminin USB2.0 portuna bağlanmıştır. Gömülü bilgisayar, EP9302 işlemcisi kullanılarak üzerinde gömülü Linux çalışmaları yapılabilen bir platformdur. Gömülü bilgisayar ile 3G modem bağlantısı Şekil 3.2’de görüldüğü gibi USB’den bağlanmıştır. 3G modemin USB portundan bağlanabilmesi için gerekli Linux sürücüsünün cihaza kurulması gerekmektedir. Bu çalışmada gerekli sürücünün kurulması için Linux çekirdeğini derleme esnasında bazı

kütüphane dosyalarında modem üreticisi tarafından belirlenen değişiklikler yapılmış ve daha sonra Linux çekirdeği derlenmiştir.



Şekil 3.3. Gömülü bilgisayar ile 3G modem bağlantısı

Öncelikle Sim5218 3G modem ile haberleşmenin sağlanabilmesi için gerekli olan USB sürücüsü kurulum işlemi gerçekleştirilmiştir. USB sürücüsünün kurulumu aşağıdaki gibidir:

Linux çekirdeğinin 2.6.20 ve sonrasındaki versiyonlarında bulunan “usbserial” sürücüsü Sim5218 ile USB’den haberleşmek için kullanılabilmektedir. Ancak NMEA (National Marine Electronics Association) portu için kullanılamamaktadır. NMEA portu için de kullanılabilmesi isteniyorsa çekirdeğin “drivers\usb\serial\Usb-serial.c” ve “drivers\usb\serial\Generic.c” değişikliklerin yapılması gerekmektedir. İsimleri belirtilen dosyalara aşağıda kırmızı ile yazılmış kodlar eklenmiştir.[12]. Belirtilen değişiklikler yapıldıktan sonra Linux çekirdeği yeniden derlenmiştir.

“drivers\usb\serial\Usb-serial.c: “ dosyasında yapılan değişiklikler kırmızı renkli yazılmıştır.

```

int usb_serial_probe(struct usb_interface *interface, const struct usb_device_id
*id)
{
    .....
    int max_endpoints;
    int isnmea = 0;
    .....
  
```

```

/* descriptor matches, let's find the endpoints needed */
/* check out the endpoints */
iface_desc = interface->cur_altsetting;
/*
    * we check if this interface is our nmea port, the number
    * of nmea interface in our device is 1.
*/
if(dev->descriptor.idVendor == 0x05c6 &&
    dev->descriptor.idProduct == 0x9000 &&
    iface_desc->desc.bInterfaceNumber == 1)
    isnmea = 1;
for (i = 0; i < iface_desc->desc.bNumEndpoints; ++i) {
    endpoint = &iface_desc->endpoint[i].desc;
    .....
    if (usb_endpoint_is_bulk_out(endpoint)) {
        /* we found a bulk out endpoint */
        dbg("found bulk out on endpoint %d", i);
        if(!isnmea) {
            /*it mustn't send data to our nmea port*/
            bulk_out_endpoint[num_bulk_out] = endpoint;
        }
        .....
    }
    .....
#ifdef CONFIG_USB_SERIAL_GENERIC
    if (type == &usb_serial_generic_device) {
        num_ports = num_bulk_out;
        if (num_ports == 0 && !isnmea) {
            unlock_kernel();
            dev_err(&interface->dev, "Generic device with no bulk
                out, not allowed.\n");
        }
    }
}

```

```

    kfree (serial);
    return -EIO;
}
}
#endif
.....

```

“drivers\usb\serial\Generic.c” dosyasında yapılan değişiklik aşağıdadır.

```

} /* usb_serial_probe()*/
int usb_serial_generic_write(struct usb_serial_port *port, const unsigned char
                           *buf, int count)
{
    .....
    /* no bulk out, so return 0 bytes written */
    /*
     * as most applications will not return until all the data be sent out,
     * so if the write operation is doing on our nmea port we just return the
     count
     * and send nothing. Other interface using this driver will not go here as
     * usb_serial_probe will filter them if they have on bulk out endpoint.
     */
    //return 0;
    return count;
}

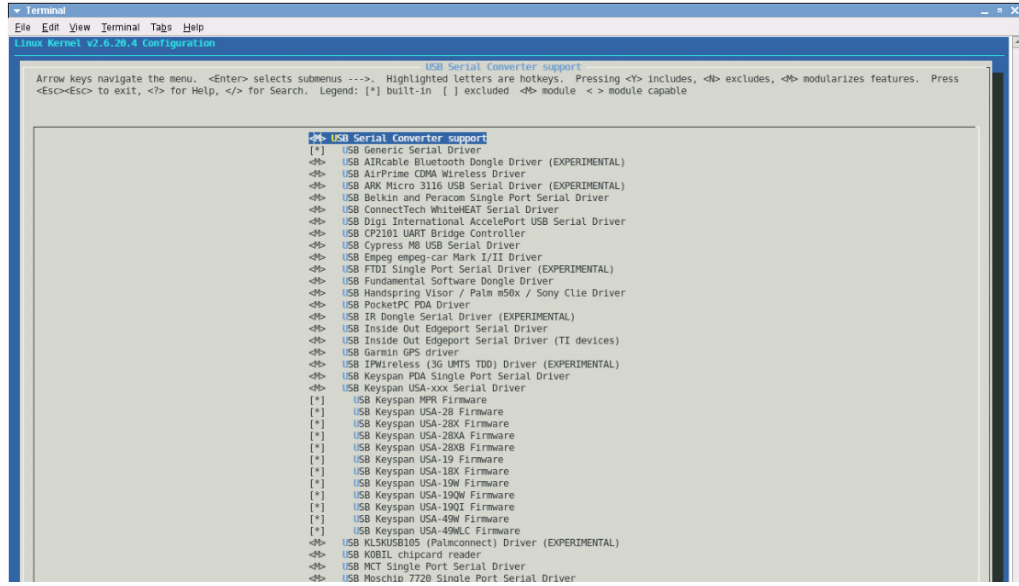
```

Bu değişiklikler yapıldıktan sonra Linux çekirdeği derlenir. (Eğer NMEA portu kullanılmayacaksa bu değişikliklere gerek yoktur.) Linux çekirdeği derlenirken Bölüm 2’de belirtilen adımlar takip edilir. Burada 3G modem ile gömülü bilgisayarın haberleşmesi için gerekli USB sürücüsünün kurulumu esnasında yapılması gerekenler aşağıdaki gibi sıralanmıştır [12].

1.Önce Linux kaynak kodu www.kernel.org adresinden indirilir.

2. Sıkıştırılmış dosyalar açılır. Daha sonra 3G modem USB sürücüsü kurulumu için gerekli değişiklikler yapılır.

3. “make menuconfig” ile USB sürücüsü modül olarak derlenir. Şekil 3.3 te görüldüğü gibi USB sürücünün ayarlanması için açtığımız sayfadan gerekli ayarlar yapılır.



Şekil 3.4. USB sürücüsünün modül olarak ayarlanması

4. “make modules” komutu ile usbserial sürücüsü “usbserial.ko” olarak derlenir.

5. USB sürücüsü “make modules_install” komutu kullanılarak “usbserial.ko” lib/modules/... dosyasında oluşturulmuş olur. Bu aşamada USB için “usbserial.ko” sürücüsü elde edilmiştir.

6. Elde edilen sürücü “insmod usbserial.ko vendor=0x05c6 product=0x9000” komutu kullanılarak cihaz üzerinde kurulmalıdır. Bu komutta belirlenen “vendor” ve “product” değerleri kullanılan Sim5218 cihazının ID’sidir.

7. USB sürücüsü hazır hale getirildikten sonra artık 3G modem USB bağlantısı sağlanabilmektedir. Yapılan bu işlemler her seferinde tekrarlanmak istenmiyor ve sistemin çalışmaya başlamasından sonra çalışması isteniyorsa sürücünün kurulumunun sistem başlangıç script'lerine eklenilmesi gerekmektedir. Kullanılmaya hazır USB sürücüsü kullanımı aşağıdaki gibidir.

- Modem ile Modbus gateway cihazı USB'den bağlanır.
- USB sürücüsü kurulduktan sonra “/dev” dizini altında Sim5218 3G modeminin aygıt ara yüzleri ile eşleşebilecek “ttyUSB0”, “ttyUSB1”, “ttyUSB2”, “ttyUSB3”, “ttyUSB4” aygıt dosyaları oluşmaktadır. Bu aygıtlar ile modem aygıt ara yüzleri Tablo 3.3'te görüldüğü gibi eşleşmektedir.

Tablo 3.3. Eşleşmiş USB sanal portlar

Aygıt Dosyası	Simcom Aygıt Arayüzü
ttyUSB0	Diag Arayüzü
ttyUSB1	NMEA Arayüzü
ttyUSB2	AT Arayüzü
ttyUSB3	Modem Arayüzü
ttyUSB4	Wireless Ethernet Arayüzü

Yapılan çalışmada AT komutlar için yazılmış chat script'ler kullanılmıştır. Dolayısıyla sanal port olarak 3G modem üzerindeki AT ara yüz kullanılmıştır. Modem üzerindeki AT ara yüz cihaz üzerindeki “ttyUSB2” aygıt dosyası ile eşleşmektedir. Bu sebepten “ppp” aygıtı ile “ttyUSB2” aygıtı eşleştirilerek cihazın internet bağlantısı sağlanmıştır.

Sim5218 3G modem için USB sürücüsü kurulduktan sonra “/etc/ppp” dosya konumunda iken “pppd” komutu ile 3G üzerinden internete bağlantı sağlanabilmektedir. Aşağıda ekrana gelen mesajlardan modem sayesinde cihazın almış olduğu IP değeri görebilmektedir.

```
/etc/ppp # pppd
BESIR ROUTER...
ECHO OFFING ..
Setting APN
Dialing...
Serial connection established.
using channel 1
Using interface ppp0
Connect: ppp0 <--> /dev/ttyUSB2
.
.
.
local IP address 188.57.178.227
remote IP address 10.64.64.64
primary DNS address 86.108.132.164
secondary DNS address 86.108.132.165
```

Yukarıda görüldüğü gibi 3G modem sayesinde Modbus gateway cihazı internete bağlanabilecek hale getirilmiştir. Sunucu modunda çalışan cihazımızın IP adresi “188.57.178.227” olarak alınmıştır. İstemciler tarafından sunucu modunda çalışan cihaz üzerinde açılan port ve cihaz IP’sine bağlantı sağlanarak istenen işlemler yapılabilmektedir.

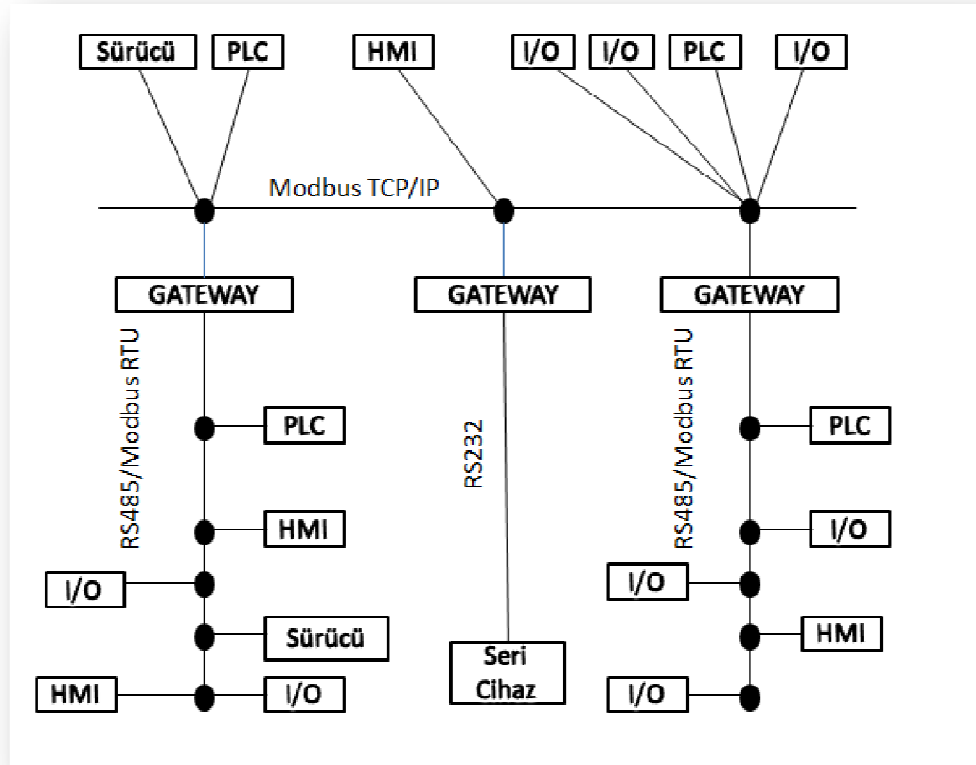
4. MODBUS PROTOKOLÜ

Modbus, endüstriyel uygulamalarda diğer standart iletişim protokollerinin yanında sağlam geçmişi ve basit altyapısı sayesinde sürekli artan sayıda imalatçı tarafından desteklenmekte ve çoğu endüstriyel sistemle haberleşebilen en eski seri iletişim protokollerinden biridir [8]. Modbus, ağ üzerindeki cihazlar arasında istemci/sunucu haberleşmesini sağlayan OSI (Open System Interconnection) modelinin 7. seviyesindeki uygulama katmanıdır. Özelleştirilen fonksiyon kodları sayesinde istek/cevap işlemleri gerçekleştirilmektedir.

Tezin bu bölümünde yapılan çalışmada kullanılan Modbus protokolü detaylı bir şekilde incelenecektir. Öncelikle, Modbus gateway cihazda Modbus protokolünün kullanılması anlatılacaktır. Daha sonra, Modbus gateway cihazı tasarımında kullanılan Modbus TCP ve Modbus RTU protokolleri, bu protokoller ile ilgili örnek uygulamalar, Modbus protokolü veri modeli ve sonrasında Modbus protokolünde kullanılan fonksiyon kodları ve tanımları anlatılacaktır.

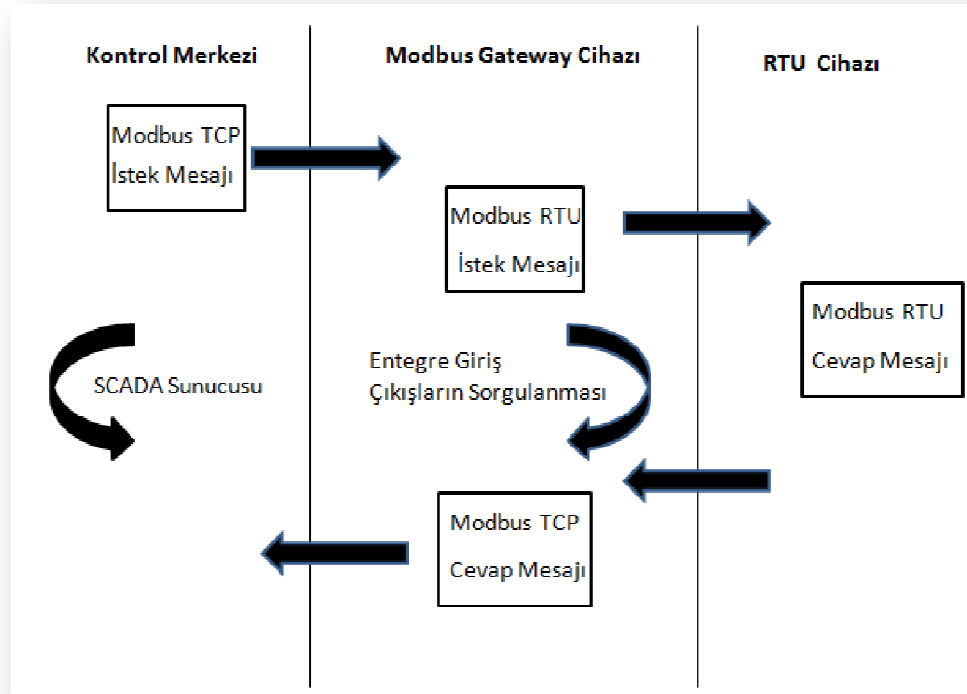
4.1. Modbus Protokolünün Modbus Gateway Cihazı Tarafından Kullanılması

Şekil 4.1’de Modbus protokolü için genel bir uygulama çerçevesi görülmektedir. Şekil 4.1’de görüldüğü gibi, Modbus protokolü sayesinde uzaktan Modbus protokolünü destekleyen cihazlara erişim imkânı, Modbus gateway özelliği veya Modbus TCP protokolü kullanılarak gerçekleştirilmektedir. Modbus protokolü, Modbus TCP ve Modbus RTU protokolleri ile sırayla internet ve seri haberleşme yöntemleri ile kullanılabilir. Modbus TCP protokolü Ethernet veya GSM haberleşmesi ile yapılabilirken, Modbus RTU protokolü RS485/RS232 haberleşme ara yüzleri ile yapılabilir.



Şekil 4.1. Modbus ağ mimarisi

SCADA sistemlerinde, birbirine yakın mesafedeki RTU cihazlarını sorgulama işleminde tasarım ve haberleşme maliyetlerini en aza indirmek için Modbus gateway cihazları kullanılması tercih edilmektedir. Modbus gateway cihazları kontrol merkezinden gelen bir Modbus TCP istek paketini, Modbus RTU protokolüne dönüştürerek gerekli sorgulama işlemini gerçekleştirirler. Şekil 4.2’de Modbus gateway cihazının Modbus protokolü kullanımı görülmektedir.



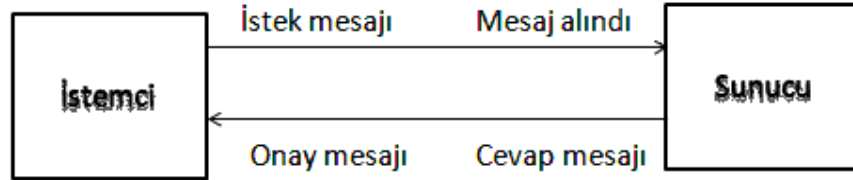
Şekil 4.2. Modbus gateway cihazı istek/cevap veri paketi dönüşüm şeması

Şekil 4.2’de, kontrol merkezinden gelen bir Modbus TCP paketi, bağlı bulunan RTU cihazları sorgulanırken, Modbus gateway cihazı tarafından alınarak Modbus RTU paketine dönüştürülmektedir. Daha sonra sorgulanan RTU cihazından gelen cevap mesajı, tekrar Modbus TCP paketine dönüştürülerek kontrol merkezine cevap olarak gönderilmektedir. Eğer Modbus gateway cihazı üzerindeki dijital girişler, dijital çıkışlar veya röle çıkışları gibi entegre giriş/çıkışlar sorgulanıyorsa, gelen Modbus TCP paketi içeriğine göre gerekli işlem yapıldıktan sonra gelen isteğe gerekli cevap Modbus TCP paketi ile gönderilir. Modbus gateway cihazı tarafından veri alışverişinde kullanılan Modbus TCP protokolü Bölüm 4.2’de ve Modbus RTU protokolü Bölüm 4.3’te anlatılacaktır.

4.2. Modbus TCP Protokolü

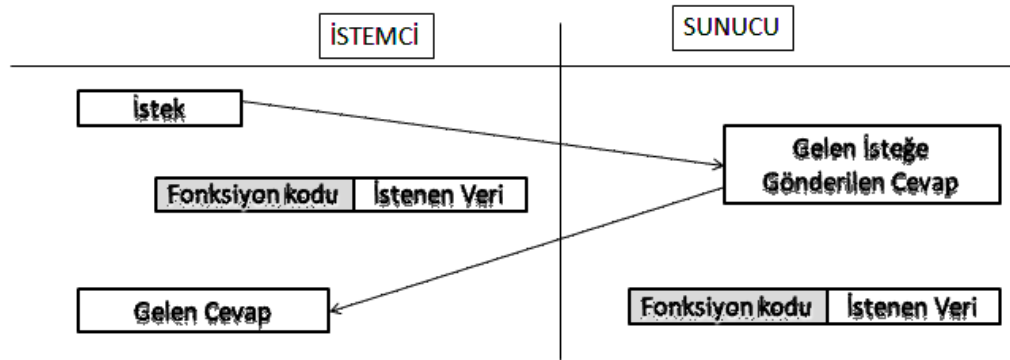
Modbus TCP protokolü, Ethernet TCP/IP ağı üzerindeki istemci ve Modbus gateway sunucusu arasındaki haberleşmedir. İstemci ve sunucu arasında, Modbus istek mesajı, Modbus mesaj işareti (indication), Modbus mesajı gönderildi onay mesajı (confirmation) ve Modbus cevap mesajı olmak üzere 4 tip mesaj vardır [10].

Şekil 4.3'te, Modbus haberleşme protokolünün istemci ve Modbus gateway sunucusu arasındaki işlemi görülmektedir. Burada istemci tarafından istek gönderildiği görülmektedir. Modbus TCP bağlantısı kurulduktan sonra, istemci ve sunucu arasında haberleşme işlemi, Modbus TCP protokolünde Master (yöneten) olarak ayarlanan cihaz tarafından, Slave (yönetilen) olarak tanımlanan cihaz sorgulanarak gerçekleştirilir. Bu doğrultuda istemci veya sunucu, TCP protokolünde Master veya Slave olarak tanımlanabilir. Modbus TCP protokolünde istemci ve sunucudan hangisi Master olarak belirlenmişse, o cihaz tarafından sorgulama işlemi gerçekleştirilir.

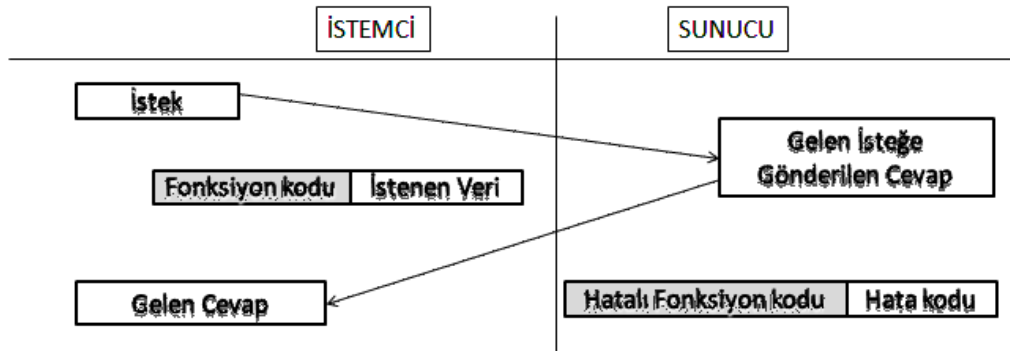


Şekil 4.3. İstemci sunucu mesaj akışı

Modbus TCP protokolünde, sunucuya gelen bir istek, gelen fonksiyon koduyla beraber istemciye cevap olarak gönderilir. Hatasız bir cevaplama işleminde sunucu, istemciye fonksiyon kodunu ve istenen veri paketini gönderir. İstemci tarafından gelen isteğin hatalı olması durumunda, hata fonksiyon kodu sunucu tarafından istemciye gönderilir. Şekil 4.4 ve 4.5’de istemci ve sunucu arasında hatasız ve hatalı fonksiyon gönderilmesi işlemleri gösterilmiştir.



Şekil 4.4. Modbus TCP/IP istek/cevap mesajı (Hatasız cevap)



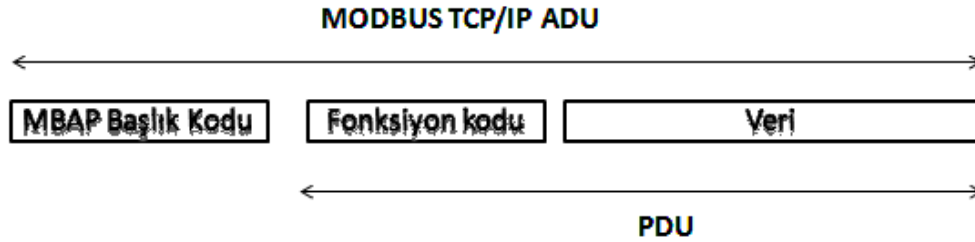
Şekil 4.5. Modbus TCP/IP istek/cevap mesajı (Hatalı cevap durumunda)

Modbus protokolünde kullanılan TCP paket yapısı Bölüm 4.2.1’de anlatılacaktır. Bölüm 4.2.2’de Modbus veri paketi içeriği ve sonrasında Bölüm 4.2.3’te de Modbus TCP protokolü uygulama örneği anlatılacaktır.

4.2.1. Modbus TCP Paket Yapısı

Modbus istek paketi, istemci tarafından oluşturulur. Modbus TCP/IP istek/cevap paketi Şekil 4.6.’da görüldüğü gibi, MBAP (Modbus Application Protocol) başlık (header) kodu, fonksiyon kodu ve veri alanlarından oluşmaktadır. Bu alanların hepsi veri uygulama birimini (ADU, Application Data Unit)

oluşturmakta iken, fonksiyon kodu ve veri alanı ise protokol veri ünitesini (PDU, Protocol Data Unit) oluşturmaktadır. PDU, Modbus RTU protokolüne dönüştürülen verinin oluşturulmasında kullanılmaktadır. Modbus, TCP/IP ağına paketi taşıırken, veri paketi MBAP başlık verisi eklenerek TCP paketine dönüştürülür ve belirlenen noktaya gönderilir [10].



Şekil 4.6. Modbus TCP/IP istek/cevap paketi

MBAP, Modbus protokolü uygulama başlık verisi olarak tanımlanmaktadır. MBAP header paketi içeriği Bölüm 4.2.1.1'de, Modbus TCP veri paketi içeriği ise Bölüm 4.2.2'de anlatılmaktadır.

4.2.1.1. MBAP Header Paketinin İçeriği

MBAP Header kodu 7 bayt uzunluğundan oluşmaktadır [10]. Tablo 4.1'de MBAP header paket içerikleri ve açıklamaları verilmiştir. Daha sonrasında MBAP header paketi içerisindeki alanların işleyişi aşağıda açıklanmıştır.

Tablo 4.1. MBAP header paketi içeriği

Alanlar	Veri Uzunluğu	Açıklama	İstemci	Sunucu
Transaction ID (İşlem ID'si)	2 Bayt	Modbus İstek/cevap numarası	İstemci tarafından oluşturulur	İstemciden gelen transaction ID
Protocol ID (Protokol ID'si)	2 Bayt	Modbus protokolü = 0	İstemci tarafından oluşturulur	Sunucu tarafından geri gönderilir
Length (Uzunluk)	2 Bayt	Pakette bundan sonra kaç bayt olacağını belirtir	İstemci tarafından hesaplanır.	Sunucu tarafından gönderilecek cevap paketine göre oluşturulur.
Slave ID (Köle ID'si)	1 Bayt	Sorgulanacak RTU cihazın ID' si veya Modbus gateway cihazın ID'sidir.	İstemci tarafından oluşturulur.	Sunucu tarafından cevap gönderilirken oluşturulur.

Transaction ID: İstemci tarafından oluşturulur. Gönderilen her paketten sonra 1 artar. 2 baytlık veridir. Sunucu tarafından cevap verildiğinde, gelen transaction ID değeri cevap verisine eklenerek geri gönderilir.

Protocol ID: Modbus protocol numarasını temsil etmektedir. Modbus protokolü için 0x0000 olarak bilinir. 2 baytlık veridir.

Length: Veri paketinin bu bayt değerinden sonra kaç bayt olduğunu belirtir. Yani Unit ID ve verinin toplam kaç bayt olduğunu belirtir. 1 baytlık veridir.

Slave ID: 1 baytlık veridir. Modbus TCP/IP protokolü ile bir cihazla haberleşme işleminde veya Modbus gateway modunda çalışan cihazlar sayesinde RTU cihazlarla haberleşme işleminde sorgulanan cihazların ID'sidir. Modbus ID'si

istemci tarafından gönderilir. İlgili ID'ye sahip cihaz tarafından cevap olarak kendi ID'si veri paketi ile beraber gönderilir [10].

4.2.2. Modbus TCP Veri Paketi İçeriği

Modbus TCP protokolünde istek ve cevap işlemlerinde kullanılan Modbus TCP istek ve Modbus TCP cevap veri paketlerinin içeriği sırayla Tablo 4.2 ve Tablo 4.3'te açıklanmıştır.

Tablo 4.2. Modbus TCP istek paketi içeriği

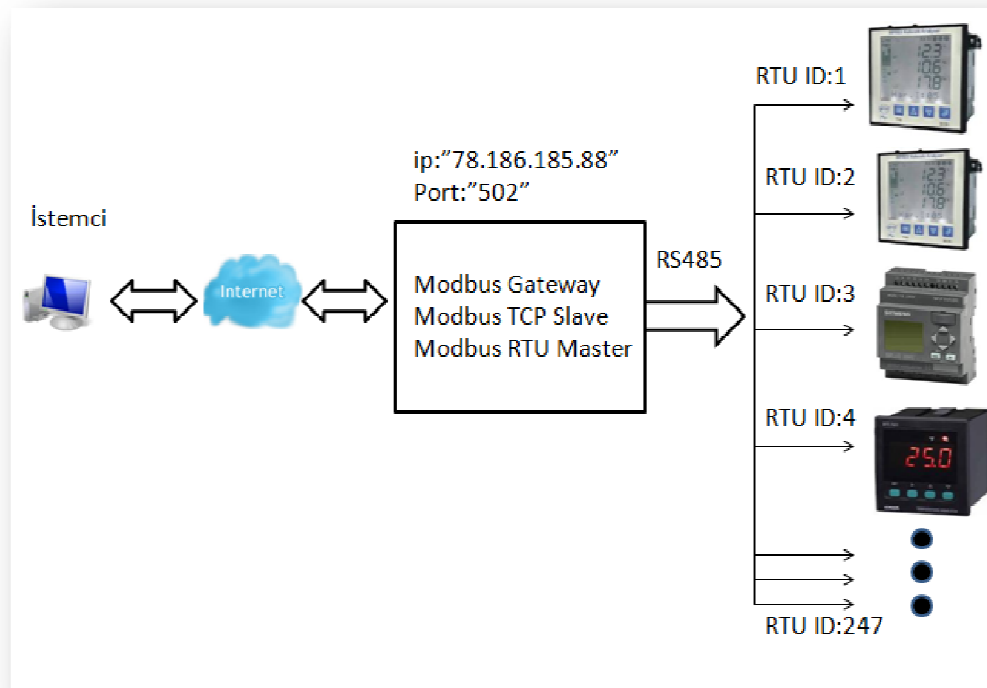
Alanlar	Veri Uzunluğu	Açıklama
Transaction ID	2 Bayt	Modbus İstek/cevap numarası
Protocol ID	2 Bayt	Modbus protokolü = 0
Length	2 Bayt	Pakette bundan sonra kaç bayt olacağını belirtir
Slave ID	1 Bayt	Sorgulanacak RTU cihazın ID'si veya Modbus gateway cihazın ID'sidir.
Function	1 Bayt	Modbus protokolünde yapılacak işlemin fonksiyon kodu
Start Address High Byte	1 Bayt	Sorgulanacak verinin başlangıç adresinin yüksek değerlikli baytı
Start Address Low Byte	1 Bayt	Sorgulanacak verinin başlangıç adresinin düşük değerlikli baytı
Number of Register High Byte	1 Bayt	Başlangıç adresinden itibaren sorgulanacak yazmaç (register) sayısının yüksek değerlikli baytı
Number of Register Low Byte	1 Bayt	Başlangıç adresinden itibaren sorgulanacak yazmaç sayısının düşük değerlikli baytı

Tablo 4.3. Modbus TCP cevap paketi içeriği

Alanlar	Veri Uzunluğu	Açıklama
Transaction ID	2 Bayt	Modbus İstek/cevap numarası
Protocol ID	2 Bayt	Modbus protokolü = 0
Length	2 Bayt	Pakette bundan sonra kaç bayt olacağını belirtir
Slave ID	1 Bayt	Sorgulanmış RTU cihazın ID'si veya Modbus gateway cihazın ID'sidir.
Function	1 Bayt	Modbus protokolünde yapılan işlemin fonksiyon kodu
Byte Number	1 Bayt	Sorgulanan verinin cevap olarak gönderilirken toplamda kaç byte olduğunu belirtir
1.Register High Byte	1 Bayt	1.Yazmaç (Register) MSB
2.Register Low Byte	1 Bayt	1. Yazmaç (Register)LSB
.	.	.
.	.	.
.	.	.
N.Register High Byte	1 Bayt	N. Yazmaç (Register)MSB
N.Register Low Byte	1 Bayt	N. Yazmaç (Register)LSB

4.2.3. Modbus TCP Protokolü Uygulama Örneği

Yapılan çalışmada tasarlanan Modbus gateway cihazı ile gerçekleştirilen Modbus TCP protokolü uygulaması, Şekil 4.7’de görülmektedir. Bu uygulamada, istemci (Modbus TCP protokolünde Master) tarafından gelen bir Modbus TCP paketi, önce Modbus TCP paketi içeriğinden, Modbus RTU veri paketine dönüştürülmektedir. Burada gelen Modbus TCP paketinde, öncelikle veri paketindeki Slave ID değerine bakılır. Eğer Slave ID değeri, Modbus gateway cihazının ID’si ile eşleşirse, Modbus gateway cihazı üzerindeki entegre giriş çıkışlar sorgulanarak Modbus TCP paketi oluşturulur ve ilgili sorguya cevap verilir. Eğer Slave ID değeri Modbus gateway cihazının Slave ID’si ile eşleşmezse, Modbus gateway cihazına bağlı bulunan RTU’lardan Slave ID değerine sahip olan RTU, Modbus RTU protokolü kullanılarak sorgulanır. Daha sonra Modbus RTU protokolü ile elde edilen cevap verisi ile Modbus TCP paketi oluşturularak istemciye cevap olarak gönderilir. Örnek uygulama ve açıklamalar aşağıda anlatılmaktadır.



Şekil 4.7. Modbus TCP uygulaması

Şekil 4.7'de, istemci tarafından belirlenen IP ve port adresine internetten sağlanan bağlantı sonrasında, Modbus gateway cihazına bağlı RTU cihazları sorgulanabilmektedir. Aşağıda verilen örnek uygulamada, gelen Modbus TCP veri paketine, yine Modbus TCP veri paketi oluşturularak cevap verilmektedir. Burada, istemci herhangi bir SCADA yazılımı veya Modbus TCP protokolünü destekleyen ve TCP Master olarak çalışabilen endüstriyel bir cihaz olabilmektedir. Gönderilen pakette, Modbus gateway cihazına bağlı RTU ID değeri “1” olan cihazın 0x0000 adresinden başlayarak 3 adet yazmaç (register) değeri okunacaktır. Burada istemci tarafından gönderilen Modbus TCP sorgu paketi ve gelen Modbus TCP cevap paketi içerikleri, Tablo 4.4 ve Tablo 4.5’te gösterilmiştir.

Tablo 4.4. Modbus TCP sorgu paketi

Alanlar	Uzunluk	Hex Değeri
Transaction ID	2 bayt	0001
Protocol ID	2 bayt	0000
Length	2 bayt	0006
Slave ID	1 bayt	0001
Function	1 bayt	03
Start Address High Byte	1 bayt	00
Start Address Low Byte	1 bayt	00
Number of Register High Byte	1 bayt	00
Number of Register Low Byte	1 bayt	03

Tablo 4.5. Modbus TCP cevap paketi

Alanlar	Uzunluk	Hex Değeri
Transaction ID	2 bayt	0001
Protocol ID	2 bayt	0000
Length	2 bayt	0009
Slave ID	1 bayt	0001
Function	1 bayt	03
Byte Number	1 bayt	06
1. Register High Byte	1 bayt	00
1. Register Low Byte	1 bayt	2B
2. Register High Byte	1 bayt	00
2. Register Low Byte	1 bayt	00
3. Register High Byte	1 bayt	00
3. Register Low Byte	1 bayt	64

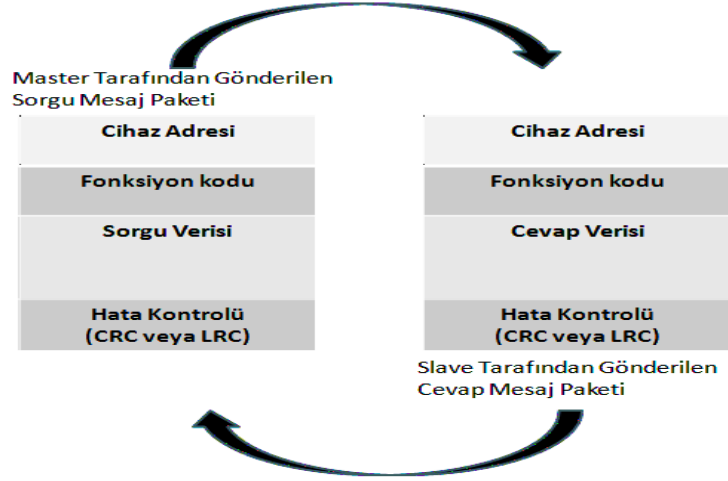
Görüldüğü gibi, istemci tarafından 1, 2 ve 3 numaralı adreslerden sorgulanan veriler: 0 → 0x002B, 1 → 0x0000, 2 → 0x0064 olarak okunmaktadır.

4.3. Modbus RTU Protokolü

Yapılan çalışmada, Modbus gateway cihazına bağlanan RTU cihazların sorgulanması işleminde, Modbus RTU protokolü kullanılmaktadır. Modbus RTU protokolü, seri haberleşme sağlayan cihazlar arasında kullanılan bir haberleşme protokolüdür. Bir Modbus RTU ağında, genellikle bir Master cihaz ve maksimum 247 Slave cihaz bulunabilmektedir. Master, ağ üzerinde haberleşme işlemini başlatan birimdir. Slave ise sürekli olarak Master tarafından gelen mesajları dinlemektedir.

Slave, gelen mesajlarda adres bilgisini kontrol eder. Kendisinin adresi ile eşleşirse, ilgili mesajda istenenler yapılır ve gelen sorguya cevap gönderilir.

Master ve Slave arasındaki Modbus haberleşme protokolünde oluşturulan mesaj paketleri Şekil 4.8’de verilmiştir. Modbus RTU protokolü veri paketi içeriği: adres bölümü, fonksiyon bölümü, veri bölümü ve hata kontrolü bölümlerinden oluşmaktadır. Bir RTU mesaj paketinde, bu 4 blok sırayla gönderilir. Bu mesaj paketinde ilk olarak adres bilgisi ile mesajın hangi Slave cihaza gönderildiği belirtilir. Daha sonra işlem türünü belirten fonksiyon kodu belirtilir. Ardından pakete veri eklenir. Bu verilerden elde edilen CRC/LRC (Cyclic Redundancy Check / Longitudinal Redundancy Check) kodu ise paketin sonuna eklenir. CRC/LRC kodu, alıcı tarafından verinin hatasız olarak ulaşp ulaşmadığının kontrolü amacıyla kullanılmaktadır. Master tarafından sorgu paketi oluşturulurken izlenen adımların aynısı, Slave tarafından cevap paketi oluşturulurken de izlenmektedir. Modbus RTU protokolünde kullanılan veri paketi içeriğinin her bir parametresi ve Modbus RTU uygulaması aşağıda anlatılacaktır.



Şekil 4.8. Modbus sorgu ve cevap paketi

Adres Bölümü: Adres bölümü Master tarafından gönderilen mesajda, Slave cihazın adresidir. Geçerli Slave ID olarak bilinen adres numarası, 1-247 değerleri arasındadır. Ağdaki Master tarafından sorgulanacak cihazın Slave ID'si yazılır. Gelen sorgu paketine cevap olarak Slave cihaz tarafından kendi adresi yazılarak cevap verilir.

Fonksiyon Kodu: Master tarafından Slave cihazın ne tür işlem yapacağını belirleyen veri alanıdır. Slave cihaz gelen sorgulamalarda hangi işlemin yapılacağını fonksiyon kodlarına bakarak belirlemektedir [9]. Geçerli fonksiyon kodları 1-255 arasındadır. Bölüm 4.5'te Modbus protokolünde kullanılan fonksiyon kodları ve tanımlamaları örnek uygulamalar ile açıklanmıştır.

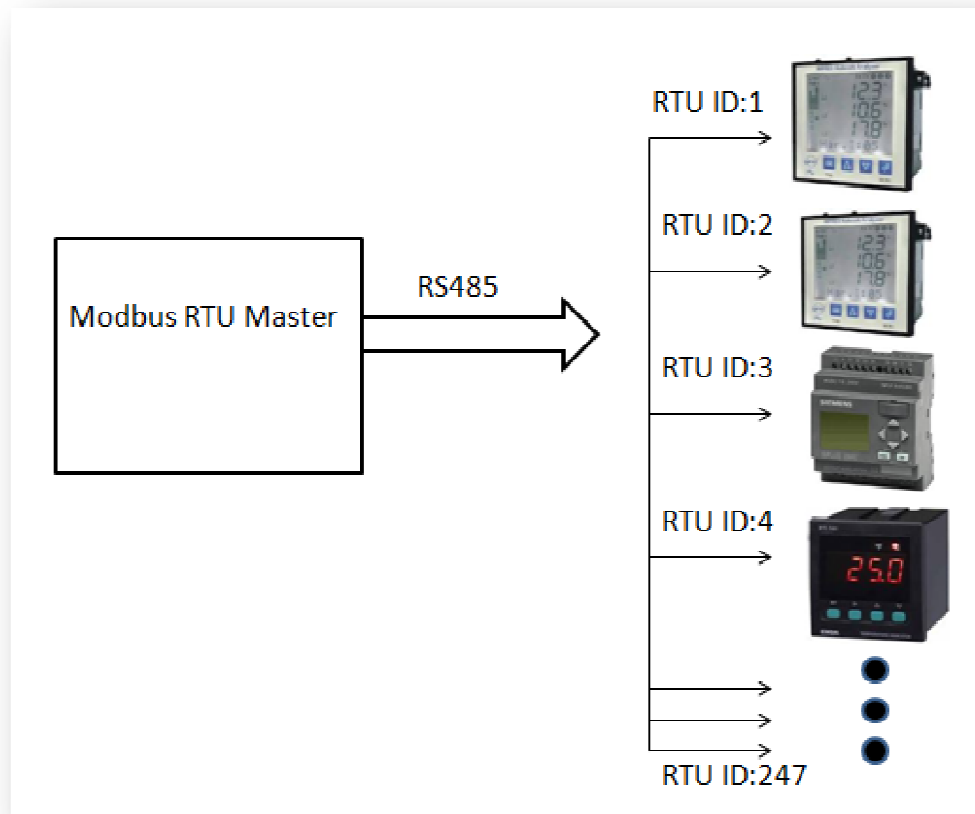
Veri (Data) Bölümü: Gönderilen mesaj ile üzerinde işlem yapılacak veya sorgulanacak verilerin adres ve değerlerini tutar. Veri bölümünün uzunluğu 1-244 bayt arasında olabilmektedir. Bölüm 4.4'te, Modbus protokolünde kullanılan veri türleri açıklanmıştır.

Modbus Protokolünde Veri (Data) Kodlama: Modbus adres ve veri baytları için "Big-Endian" veri kalıbını kullanmaktadır. Bunun için önce en büyük bayt (MSB, Most Significant Byte) gönderilir. Örneğin 16-bitlik 0x1234 verisi gönderilirken önce ilk bayt gönderilir daha sonra ikinci bayt gönderilir. Yani önce 0x12 sonra 0x34 verileri gönderilir.

Hata Kontrol Bölümü: ASCII (American Standard Code for Information Interchange) ve RTU gibi iki değişik mesaj çerçevesi kullanılarak, Master ile Slave arasında veri haberleşmesinin doğruluğu sağlanmaktadır. RTU çerçevelemesinde CRC, ASCII çerçevelemesinde ise LRC yöntemi kullanılmaktadır. Yapılan çalışmada sadece Modbus RTU protokolü uygulandığı için sadece CRC kontrolü kullanılmıştır. CRC kodu kullanılarak veri alış verişinin doğruluğu garantilenmektedir. CRC-16 algoritması kullanılarak, 2 baytlık CRC verisi mesaj paketine eklenmektedir. Gelen mesaj paketinde veri doğrulaması için gönderilen CRC kodu ile hesaplanan CRC değeri aynı değilse hata kodu geri gönderilmektedir.

4.3.1. Modbus RTU Sorgulama Örneği

Modbus RTU protokolünde Master modda çalışan cihaz, Slave cihazlara sorgulama isteği gönderir; cevap olarak istenen veri, Slave cihazlar tarafından geri gönderilir. Şekil 4.9’da Modbus RTU sorgulama örneği görülmektedir. Örnekte verilen Modbus RTU sorgulama işleminde, RTU Master tarafından gönderilen istek mesajı ve Slave tarafından gönderilen cevap mesajı içerikleri aşağıda açıklanacaktır.



Şekil 4.9. Modbus RTU bağlantı şeması

Master tarafından 0x01 adresli cihaza gönderilen birden fazla yazmaca veri yazma isteği Tablo 4.6’da belirtilen değerlerde gönderilmiştir.

Tablo 4.6. Modbus RTU istek paketi

Alanlar	Uzunluk	Hex Değeri
Cihaz Adresi	1bayt	01
Fonksiyon kodu	1 bayt	10
Adres MSB	1 bayt	00
Adres LSB	1 bayt	00
Yazmaç Sayısı MSB	1 bayt	00
Yazmaç Sayısı LSB	1 bayt	02
CRC MSB	1 bayt	40
CRC LSB	1 bayt	34

Modbus RTU Slave tarafından gönderilen cevap paketi ise Tablo 4.7’de verilmiştir. Gelen cevap paketinde görüldüğü gibi 0x0000 ve 0x0001 adreslerinden sorgulanan verilerin değeri 0x0000 →0x0064, 0x0001→0x0000 olarak okunmuştur.

Tablo 4.7. Modbus RTU cevap paketi

Alanlar	Uzunluk	Hex Değeri
Cihaz Adresi	1bayt	01
Fonksiyon kodu	1 bayt	10
Adres MSB	1 bayt	00
Adres LSB	1 bayt	00
Yazmaç Sayısı MSB	1 bayt	00
Yazmaç Sayısı LSB	1 bayt	02
Bayt Sayısı	1 bayt	04
1. Veri MSB	1 bayt	00
1. Veri LSB	1 bayt	64
2. Veri MSB	1 bayt	00
2. Veri LSB	1 bayt	00
CRC MSB	1 bayt	BE
CRC LSB	1 bayt	76

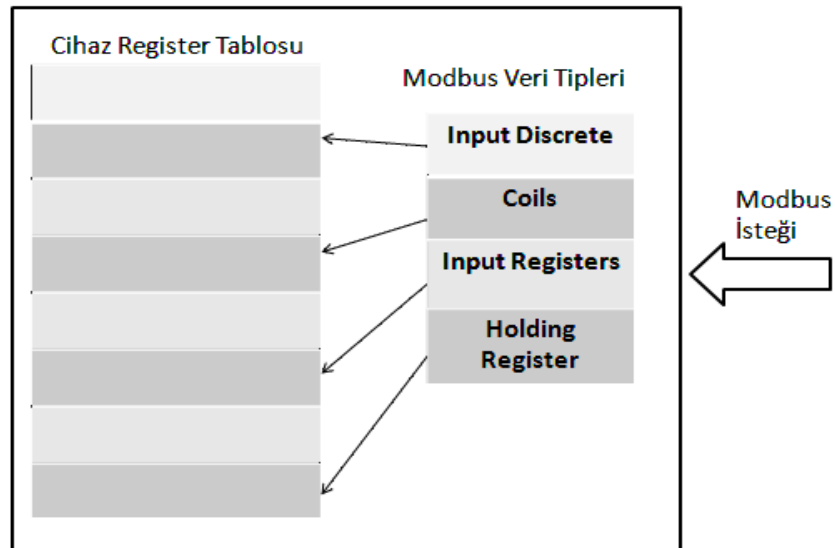
4.4. Modbus Veri Modeli

Modbus protokolünde dört farklı veri modeli bulunmaktadır. Kullanılan fonksiyonlar için değişik veri tipleri belirlenmiştir. Her bir veri tipi için yapılan işlemler belirlenen fonksiyonlar aracılığı ile yapılmaktadır. Bu veri tipleri için kullanılan erişim şekli, boyut ve kullanılan adres bölgeleri Tablo 4.8’de belirtilmiştir.

Tablo 4.8. Değişik fonksiyonlar için değişik veri grupları

Veri Tipi	Nesne Tipi	Erişim Şekli	Adres Aralığı (Hex)
Ayrık (Discrete) Girişler	1 bit	Sadece okunabilir	0- FFFF
Bobinler (Coils)	1 bit	Okunabilir ve Yazılabilir	0- FFFF
Giriş Yazmaçları	16-bit word	Sadece Okunabilir	0- FFFF
Tutucu Yazmaçları (Holding Registers)	16-bit word	Okunabilir ve Yazılabilir	0- FFFF

Modbus protokolünü destekleyen cihazlarda verilerin okunup yazılması için veri tiplerine göre 4 ayrı bellek bölgesi tahsis edilmektedir. Belirlenen adres bölgesinde, her bir veri için belirlenen aralıklar ayrıdır. Şekil 4.10'da bu bölgeler belirtilmiştir. Her veri tipi için belirlenen fonksiyonlar aracılığı ile ilgili bellek bölgesine veri yazılabilmekte veya ilgili veri bölgesinden veri okunabilmektedir.

**Şekil 4.10. Modbus veri modeli**

Bobinler (Coils) : Bir bitlik veridir. Sistem değişkenlerinin değiştirilebilmesini ya da gözlemlenmesini sağlar. Dijital giriş, dijital çıkış veya rölenin açık veya kapalı oluşunu temsil eder. 0-FFFF adres aralığında kullanılan veri tipleridir.

Ayrık Girişler (Discrete Inputs) : Dijital girişlerin açık veya kapalı olduğu bilgisinin okunması için kullanılır. Bu veri tipi sadece okuma işleminde kullanılmaktadır. 0-FFFF adres aralığında kullanılan veri tipleridir.

Giriş Yazmaçları (Input Registers): Bu veri tipi cihazlardan, analog girişleri okumak için ya da cihazlardan bilgi toplamak için kullanılmaktadır. Bu veri tipi 16 bit uzunluğunda olup sadece verileri okuma amacıyla kullanılmaktadır. 0-FFFF adres aralığında kullanılan veri tipleridir.

Tutucu Yazmaçları (Holding Registers): Cihaz üzerindeki analog çıkışların değerlerini ya da cihazlara gönderilen bilgilerin yazılabildiği alanlardır. 0-FFFF adres aralığında kullanılan veri tipleridir.

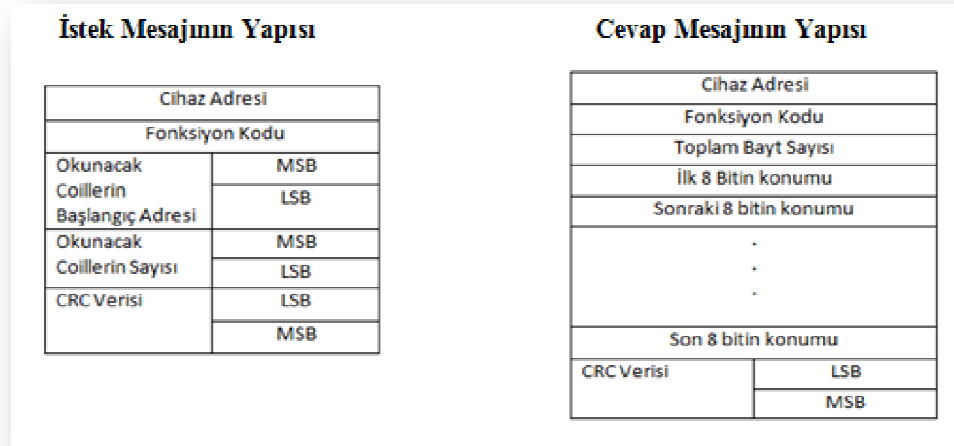
4.5. Modbus Fonksiyon Kodları ve Tanımlamaları

Modbus protokolünü destekleyen cihazların belirlenen adres bölgesinde, her bir veri tipine uygun işlemlerin yapılabilmesi için fonksiyon kodları belirlenmiştir. Aşağıda her bir veri tipi için kullanılacak fonksiyon kodları ile yapılacak işlemler Tablo 4.9'da açıklanmıştır.

Tablo 4.9. Fonksiyon kodları ve açıklamaları

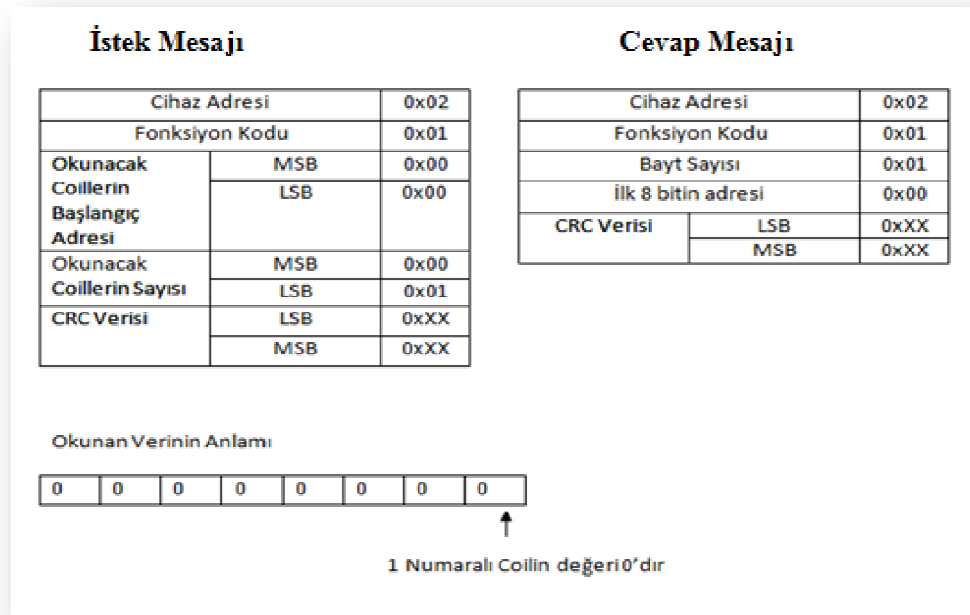
Fonksiyon Kodu	Fonksiyon Kodu ile Yapılan işlem
0x01	Bobin Durumlarını Oku (Read Coil Status)
0x02	Ayrık Girişleri Oku (Read Discrete Inputs)
0x03	Tutucu Yazmaçları Oku (Read Holding Registers)
0x04	Giriş Yazmaçlarını Oku (Read Input Registers)
0x05	Tek Bir Bobine Yaz (Write Single Coil)
0x06	Tek Bir Yazmaca Yaz (Write Single Register)
0x0F	Birden Fazla Bobine Yaz (Write Mutiple Coils)
0x10	Birden Fazla Yazmaca Yaz (Write Mutiple Register)

Bobin Durumlarını Oku (Read Coil Status) (0x01) : 0-FFFF adres aralığında belirlenen bölgede bit düzeyindeki verileri okumak için kullanılan fonksiyon kodudur. Her zaman 8 bit olarak düşünülür. Eğer üzerinde işlem yapılacak bit sayısı 8'den düşük ise diğerlerine 0 yazılarak 8'e tamamlanır. Şekil 4.11'de bobin durumlarının okunması için oluşturulan istek/cevap mesajlarının yapısı görülmektedir.



Şekil 4.11. Bobin durumlarının okunması, istek/cevap mesajlarının yapısı

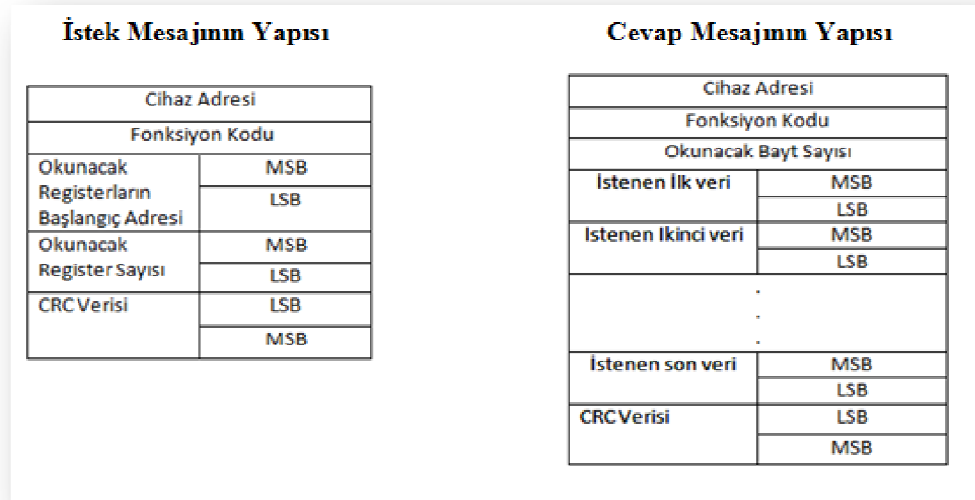
Aşağıdaki Şekil 4.12’de, 0x0000 adresindeki bobinlerin okunması için hazırlanan örnek istek mesajı ve cevap mesajı verilmiştir. 0x0000 adresindeki 1 numaralı bobin değeri 0 olarak okunmuştur.



Şekil 4.12. Bobin durumlarının okunması, istek/cevap paketi içerikleri örneği

Ayrık Girişleri Oku (Read Discrete Inputs) (0x02) : 0-FFFF adres aralığında belirlenen bellek alanlarında bit düzeyindeki kontrol çıkışlarını okumak için kullanılan fonksiyon kodudur.

Tutucu Yazmaçları Oku (Read Holding Registers) (0x03) : Cihaz veri tablosunun belirlenen başlangıç adreslerinden başlayarak okunacak yazmaç sayısının belirlenerek okunması işlemini gerçekleştiren fonksiyon kodudur. Gelen veriler önce en ağırlıklı bayt (MSB) sonra en düşük değerlikli bayt (LSB) olarak gelir. Şekil 4.13'te, tutucu yazmaçların durumlarının okunması için oluşturulan istek ve cevap mesajının yapısı verilmiştir.



Şekil 4.13. Tutucu yazmaçların okunması, istek/cevap mesajlarının yapısı

Aşağıda Şekil 4.14'te, 0x02 adresine sahip bir cihazdan 0x0010 adresinden başlayarak iki adet tutucu yazmacın okunması örnek olarak verilmiştir.

İstek Mesajı			Cevap Mesajı		
Cihaz Adresi		0x02	Cihaz Adresi		0x02
Fonksiyon Kodu		0x03	Fonksiyon Kodu		0x03
Okunacak Registerların Başlangıç Adresi	MSB	0x00	Bayt Sayısı		0x04
	LSB	0x10	Okunacak İlk Holding Register	MSB	0x00
Okunacak Register Sayısı	MSB	0x00		LSB	0x11
	LSB	0x02	İkinci Holding Register	MSB	0x00
CRC Verisi	LSB	0xXX		LSB	0x00
	MSB	0xXX	CRC Verisi	LSB	0xXX
				MSB	0xXX

Şekil 4.14. Tutucu yazmaçların okunması, istek/cevap paketi içerikleri örneği

Şekil 4.14'te görüldüğü gibi cevap mesajı kısmında ilk yazmaç: 0x0010 → 0x0011, ikinci yazmaç: 0x0011 → 0x0000 olarak okunmuştur.

Giriş Yazmaçlarını Oku (Read Input Registers) (0x04) : Input yazmaçların başlangıç adresi belirtilerek belirlenen adresten itibaren belirlenen sayıda yazmaç okunmaktadır. Gelen veriler önce en ağırlıklı bayt (MSB) sonra en düşük değerlikli bayt (LSB) olarak gelir. Şekil 4.15'te giriş registerlarının okunması için oluşturulan istek/cevap mesajlarının yapısı görülmektedir.

İstek Mesajının Yapısı			Cevap Mesajının Yapısı		
Cihaz Adresi			Cihaz Adresi		
Fonksiyon Kodu			Fonksiyon Kodu		
Okunacak Registerların Başlangıç Adresi	MSB		Okunacak Bayt Sayısı		
	LSB		İstenen İlk veri	MSB	
Okunacak Register Sayısı	MSB			LSB	
	LSB		İstenen İkinci veri	MSB	
CRC Verisi	LSB			LSB	
	MSB		.		
			.		
			İstenen son veri	MSB	
				LSB	
			CRC Verisi	LSB	
				MSB	

Şekil 4.15. Giriş yazmaçlarının okunması, istek/cevap mesajlarının yapısı

Aşağıda Şekil 4.16’da 0x02 adresine sahip cihazdan 0x0000 adresinden başlayarak 1 adet giriş yazmacı okunmaktadır.

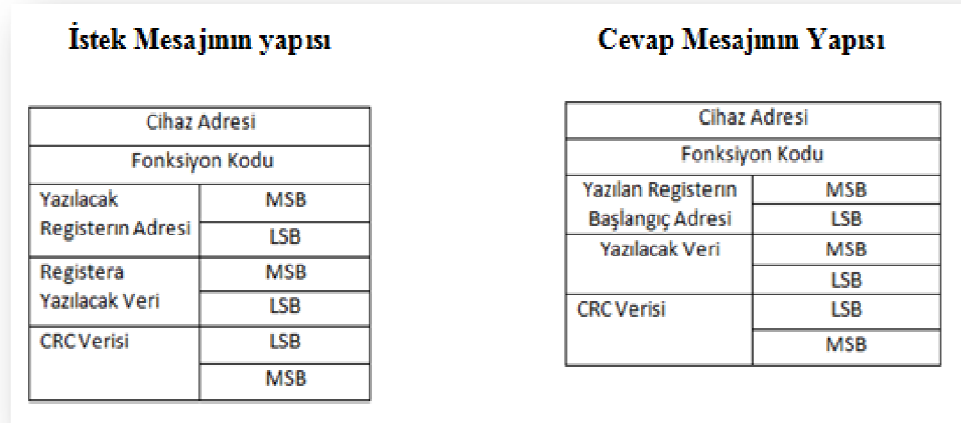
Komut Mesajı			Cevap Mesajı		
Cihaz Adresi		0x02	Cihaz Adresi		0x02
Fonksiyon Kodu		0x04	Fonksiyon Kodu		0x04
Okunacak Registerların Başlangıç Adresi	MSB	0x00	Bayt Sayısı		0x02
	LSB	0x00	Okunacak İlk Holding Register	MSB	0x00
Okunacak Register Sayısı	MSB	0x00		LSB	0x11
	LSB	0x01	CRC Verisi	LSB	0xFF
CRC Verisi	LSB	0xFF		MSB	0xFF
	MSB	0xFF			

Şekil 4.16. Giriş yazmaçlarının okunması, istek/cevap paketi içerikleri örneği

Şekil 4.16’da görüldüğü gibi 0x0000 adresindeki yazmaç değeri 0x0000 → 0x0011 olarak okunmuştur.

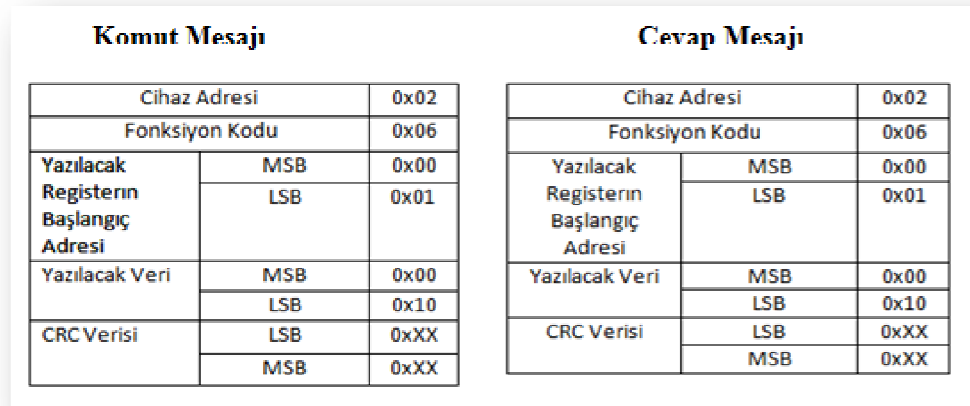
Tek Bir Bobine Yaz (Write Single Coil) (0x05) : Belirlenen adresteki bobinin durumunun değiştirilmesi işleminde kullanılan fonksiyondur.

Tek Bir Yazmaca Yaz (Write Single Register) (0x06) : Belirlenen hafızadaki bir tutucu (holding) yazmaca istenen değerin yazılması işleminde kullanılır. Şekil 4.17’de tek bir yazmaca yazma işlemi için oluşturulan istek/cevap mesajlarının yapısı görülmektedir.



Şekil 4.17. Tek bir yazmaca yazma, istek/cevap mesajlarının yapısı

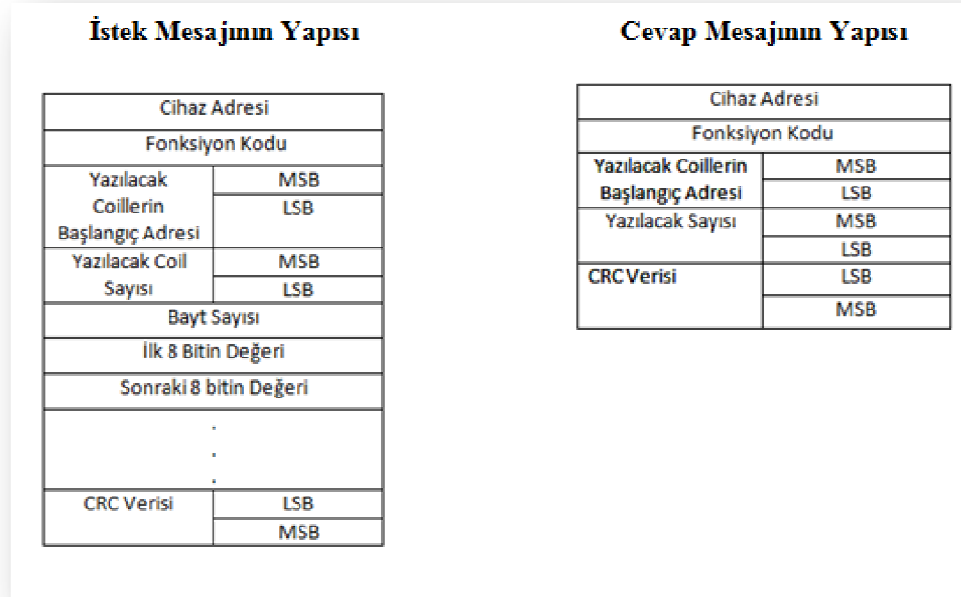
Şekil 4.18’de 0x02 adresli cihazın 0x0001 adresinden başlayarak 1 adet yazmaca 0x0010 değeri yazılmaktadır.



Şekil 4.18. Tek bir yazmaca yazma, istek/cevap paketi içerikleri örneği

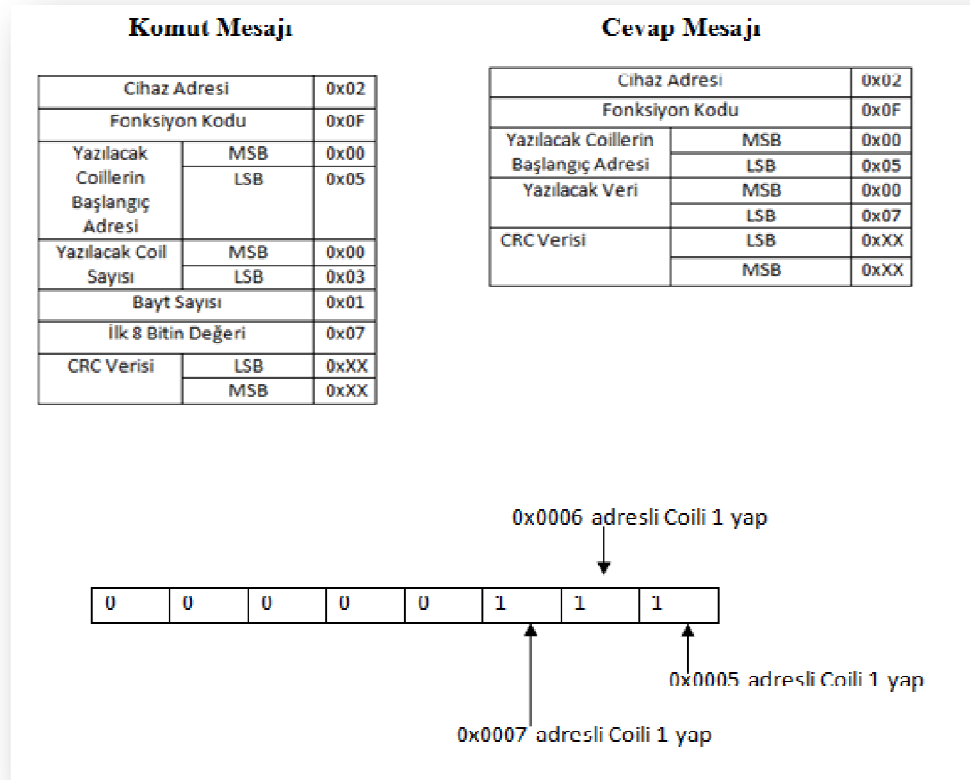
Şekil 4.18’de görüldüğü gibi 0x0001 adresindeki yazmaca 0x0001 → 0x0010 değeri yazılmıştır.

Birden Fazla Bobine Yaz (Write Multiple Coils) (0x0F) : Başlangıç adresi ve sayısı belirtilerek bobinlerin açık ya da kapalı yapılması işlemleri bu fonksiyon kodu aracılığıyla gerçekleştirilir. Şekil 4.19’da birden fazla bobine yazma işlemi için oluşturulan istek/cevap mesajlarının yapısı görülmektedir.



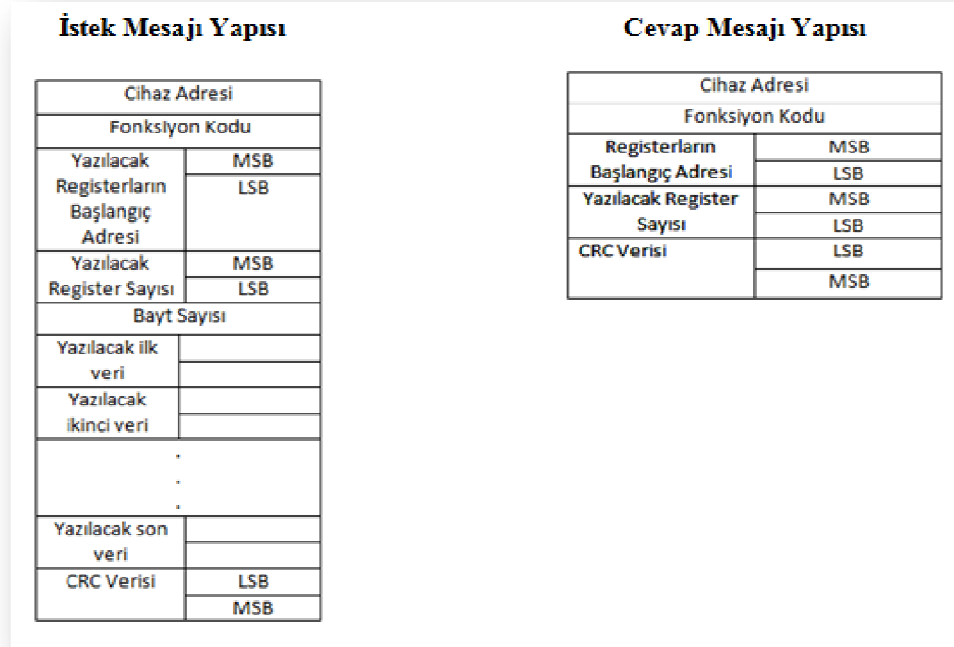
Şekil 4.19. Birden fazla bobine yazma, istek/cevap mesajlarının yapısı

Şekil 4.20’de 0x02 adresli cihazın 0x0005 adresinden başlayarak 3 adet bobinin değeri 1 yapılmıştır.



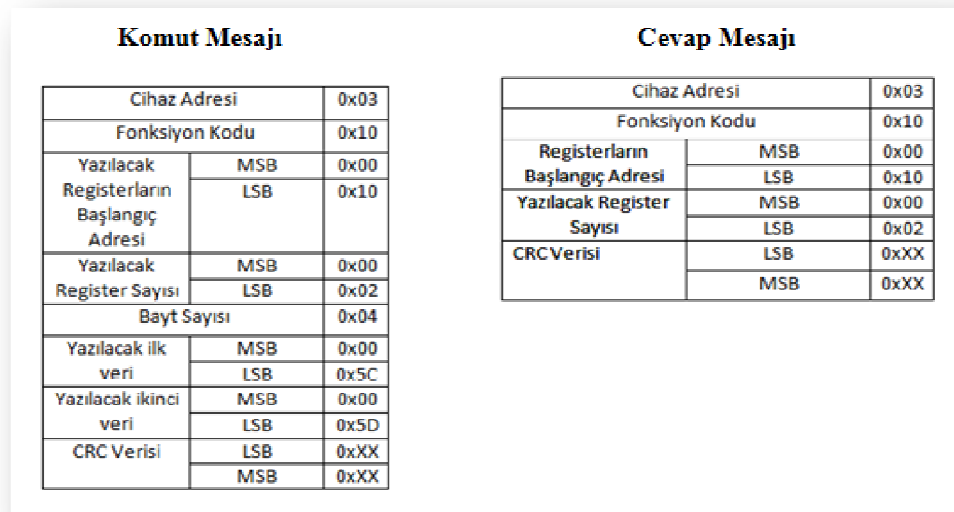
Şekil 4.20. Birden fazla bobine yazma, istek/cevap paketi içerikleri örneği

Birden Fazla Tutucu Yazmaçlara Yazma (Write Multiple Holding Registers) (0x10) : Başlangıç adresi ve yazılacak yazmaç sayısı belirtilerek tutucu yazmaçlara veri yazma işlemini gerçekleştirmek için kullanılan fonksiyon kodudur. Gönderilen veriler önce en ağırlıklı bayt (MSB) sonra düşük değerlikli bayt (LSB) olarak gönderilmelidir. Şekil 4.21’de birden fazla tutucu yazmaçlara yazma işlemi için oluşturulan istek/cevap mesajlarının yapısı görülmektedir.



Şekil 4.21. Birden fazla tutucu yazmaçlara yazma, istek/cevap mesajlarının yapısı

Şekil 4.22’de 0x02 adresli cihazın 0x0010 adresinden başlayarak 2 adet tutucu yazmacına 0x0010 →0x005C ve 0x0011 →0x005D değerleri yazılmıştır.



Şekil 4.22. Birden fazla tutucu yazmaçlara yazma, istek/cevap paketi içerikleri örneği

Modbus protokolünde yapılan sorgulamalarda hatasız olma durumlarında ilgili fonksiyonlara cevap olarak yukarıda anlatılan şekliyle cevap gönderilmektedir. Ancak herhangi bir hata olması durumunda ise hata fonksiyon kodları oluşturularak cevap gönderilmektedir. Bölüm 4.6’da oluşan hata durumunda, hata fonksiyon kodu oluşturma işlemi anlatılacaktır.

4.6. Modbus Protokolü Hata Kodları

Modbus protokolünde gerçekleştirilen sorgulama işlemlerinde, kullanılan veri türüne göre fonksiyonlar kullanılmaktadır. Yapılan sorgulama işlemlerinde hata kontrol kodları sayesinde verinin doğruluğu test edilmektedir. Eğer gelen sorgu paketinde hatalı bir istek olduğu anlaşılırsa, cevap olarak hata kodu gönderilir. Kullanılan bu fonksiyonlar ile yapılan sorgulama işlemlerinde beklenmedik bir durum karşısında cevap olarak Tablo 4.10’daki kodlar gelen fonksiyon koduna eklenerek geri gönderilir.

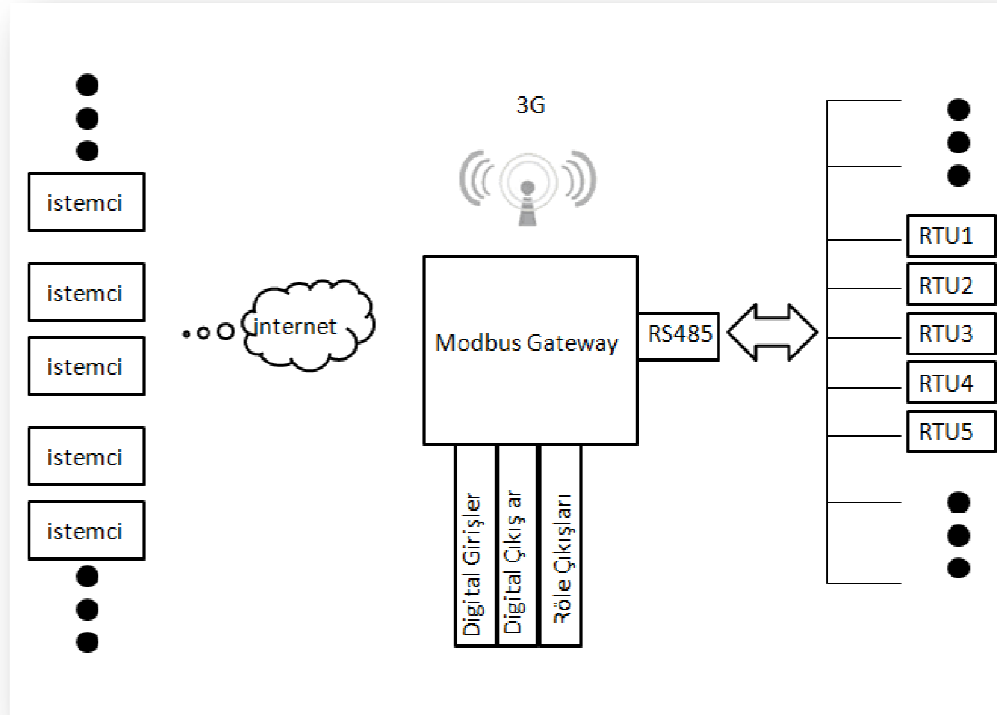
Tablo 4.10. Hata fonksiyon kodları

Hata Kodu	Modbus Adı	Açıklama
01	İllegal fonksiyon Kodu	Fonksiyon kodu sunucu tarafından bilinmiyor
02	İllegal veri adresi	Olmayan veri adresi
03	İllegal veri değeri	Olmayan veri
04	Sunucuya erişim yok	Sunucuya erişilemiyor
05	Bilgi mesajı	Sunucu Modbus paketini alamıyor
06	Sunucu meşgul	Sunucu meşgul
0A	Gateway Hatası	Gateway kullanılamıyor
0B	Gateway Hatası	Cihaz cevap vermiyor

Tablo 4.10’da belirtilen hatalardan herhangi birinin oluşması durumunda cevap olarak ilgili hata durumunu belirten fonksiyon koduna 0x80 değeri eklenerek veri paketinde fonksiyon kodunun değeri olarak geri gönderilir. Bu şekilde hata durumlarının sebebi de istemci tarafından öğrenilebilmektedir.

5. MODBUS GATEWAY CİHAZI TASARIMI

Tezin bu kısmında Modbus gateway cihazı için donanım tasarımı, yazılım tasarımı, sistemin çalıştırılması ve sonuçları anlatılacaktır. Şekil 5.1’de tasarlanan Modbus gateway cihazının genel görünümü görülmektedir.



Şekil 5.1. Modbus gateway cihazı genel görünümü

Bu bölümde öncelikle cihaz üzerindeki dijital girişler, dijital çıkışlar, röle çıkışları ve RS485 gibi donanım tasarımları anlatılacaktır. Donanım tasarımından sonra Şekil 5.1’de görüldüğü gibi, istemciler tarafından gelen bağlantı istekleri, bu isteklerin cevaplanması, birden fazla kullanıcıya cevap verebilecek yazılımın tasarımı, akıllı mesaj önceliklendirme mekanizması, ID önceliklendirme algoritmaları, tasarlanan kuyruk mekanizmaları ve zaman parametresine göre ikinci bir kuyruğun oluşturulması işlemleri sırası ile anlatılacaktır. Daha sonra tasarlanmış

cihazın çalıştırılması, test ortamı ve sonuçların irdelenmesi ile bölüm tamamlanacaktır.

5.1. Donanım Tasarımı

Tasarlanan cihaz donanımsal olarak CPU modül, geliştirme kiti üzerindeki çevresel birimler (USB, RS232, Ethernet..), dijital giriş devreleri, dijital çıkış devreleri, röle çıkış devreleri ve RS485 devrelerinden oluşmaktadır.

CPU modülü tasarımında EP9302 işlemcisi kullanılmıştır. Mikrodev firmasının geliştirmiş olduğu ARM CPU modülü kullanılarak üzerinde gerekli yazılım çalışmaları yapılmıştır [29]. Kullanılan ARM CPU modülü Şekil 5.2’de görüldüğü gibidir. ARM CPU modülünün özellikleri Tablo 5.1’de verilmiştir.



Şekil 5.2. ARM CPU modülü

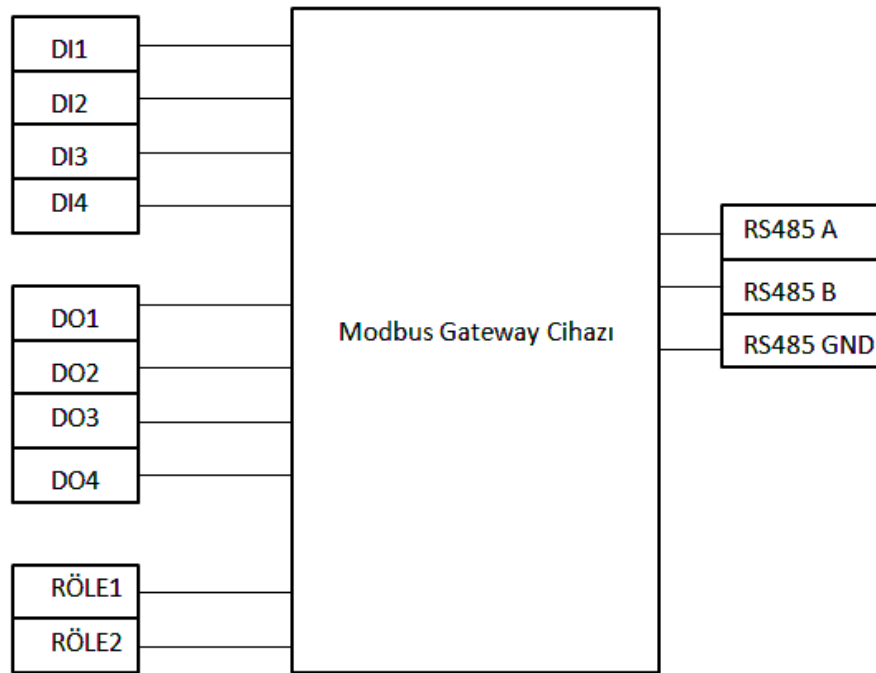
Tablo 5.1. ARM CPU modül özellikleri

İşlemci	• Cirrus Logic EP9302 • 32-bit RISC • ARM920T - 200MHz • 16KB I-Cache, 16KB D-Cache • MaverickCrunch and MaverickKey
SDRAM	• 100MHz speed • Up to 64MB capacity • 16-bit Data Bus
FLASH	• Up to 32MB capacity - NOR Flash • 16-bit Data Bus
Ethernet	• Micrel KS8721BL MII PHY on board • 1/10/100Mbps speed
USB	• 2xUSB 2.0 Host Full Speed
I/O Interface	• 2xUARTs (16550 type) • IRDA Interface • AC'97 (2-channel serial audio interface) • I2S (6-channel serial audio interface) • I2C
ADC	• 5-Channel 12-bit resolution • 0..3.3V input range
Timers	• 2x16-bit and 1x32-bit general purpose • 1x40-bit debug purpose • Watchdog timer
GPIO	• 16 GPIOs with interrupt capability • 8 optionalGPIOs
Power	• Single Voltage 3.3V DC@ ...mA

Modülün şematik, PCB tasarımı ve gömülü Linux çalışmalarında kullanılabilecek şekilde geliştirme kiti tasarımı Mikrodev firması tarafından gerçekleştirilmiştir. Yapılan çalışma için tasarlanmış CPU Modül ve geliştirme kiti kullanıldığından, gömülü sistem tasarımında gereken donanım için herhangi bir tasarım yapılmamıştır. Modülün özellikleri Tablo 5.1'de görülmektedir. Geliştirme kitinin üzerindeki pinlerden dışarı alınan pinler kullanılarak harici bir devre tasarımı

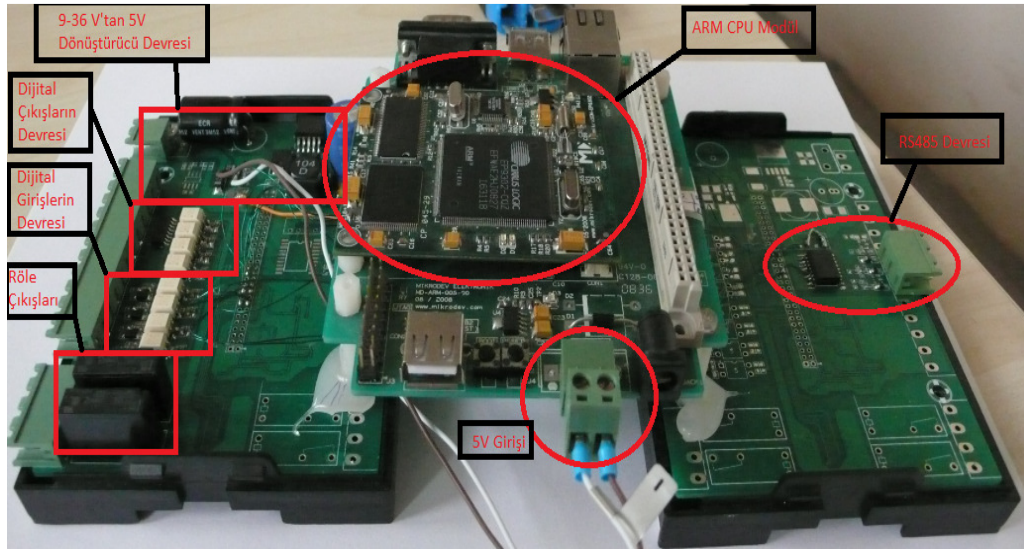
yapılmıştır. 3G modem için de 3G modem geliştirme kiti kullanılmıştır. Kullanılan 3G modem, USB portu sayesinde CPU modüle bağlanarak cihazın internete erişimi sağlanmıştır.

Modbus gateway cihazı 4 adet dijital giriş, 4 adet dijital çıkış, 2 adet Röle çıkışı ve 1 adet RS485 çıkışı gibi fiziksel giriş/çıkışlara sahiptir. Cihazın fiziksel giriş/çıkış şeması Şekil 5.3'te gösterilmiştir.



Şekil 5.3. Modbus gateway cihazı fiziksel giriş çıkış bağlantı şeması

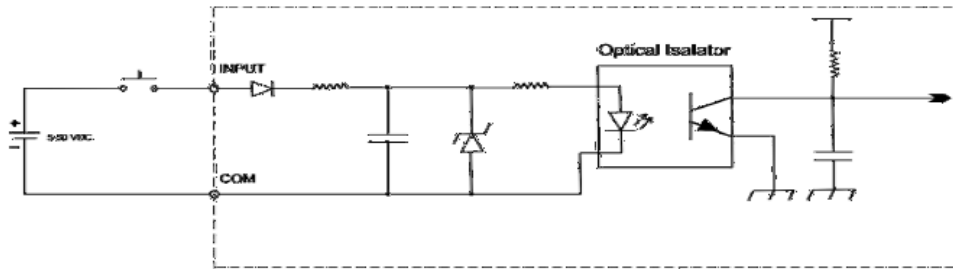
Şekil 5.4'te geliştirme kiti kullanılarak tasarlanan Modbus gateway cihazı görülmektedir. Şekil 5.4'te, Modbus gateway cihazı üzerinde bulunan CPU modül, RS485 devresi, geliştirme kiti, dijital giriş/çıkış devreleri, röle çıkışları ve dönüştürücü devresi işaretlenerek gösterilmiştir.



Şekil 5.4. Modbus gateway cihazının görünümü

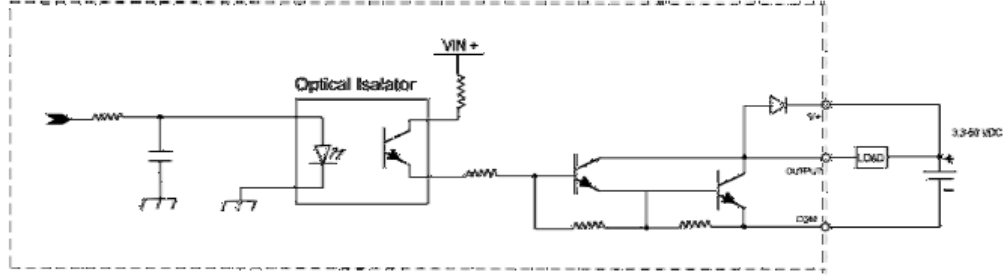
5.1.1. Dijital Girişler

Tasarımda kullanılan dijital girişler optik izolasyonlu olacak şekilde tasarlanmıştır. Burada optik izolasyon için “optocoupler” entegresi kullanılmıştır. Tasarımda kullanılan diğer donanımlarda Şekil 5.5’de görülmektedir.



Şekil 5.5. Dijital girişlerin devresi

Dijital girişlerin özellikleri Tablo 5.2’de ve bağlantı şeması Şekil 5.6’da verilmiştir.

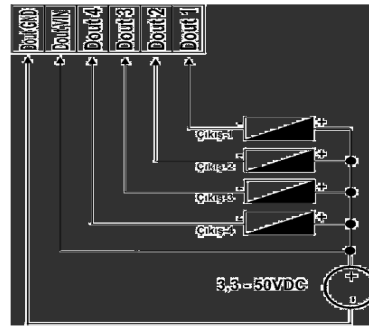


Şekil 5.7. Dijital çıkışların devresi

Dijital çıkışların özellikleri Tablo 5.3'te ve bağlantı şeması Şekil 5.8'de verilmiştir.

Tablo 5.3. Dijital çıkışların özellikleri

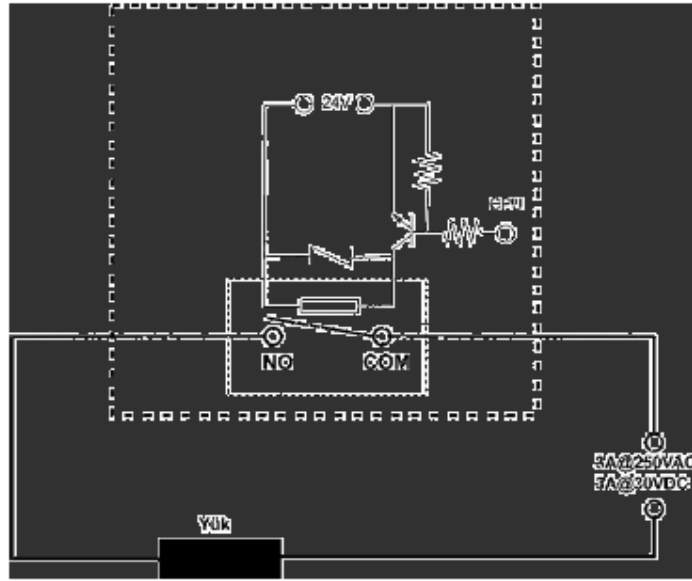
Özellik	Karakteristiği
Dijital Çıkışların Sayısı	4 adet dijital çıkış
Dijital çıkışların voltaj aralığı	3.3 - 50 VDC
Çıkış Özelliği	GND ortak uç 1 (4 points/common) Optik İzalasyonlu
Max. çekilen akım	Max. 0.5 A (All Outputs On)



Şekil 5.8. Dijital çıkışların bağlantı şeması

5.1.3. Röle Çıkışları

Yapılan çalışmada 24V'luk DC gerilim ile çalışan röleler kullanılarak 2 adet röle çıkışı eklenmiştir. Tasarlanan devre Şekil 5.9'da görülmektedir.

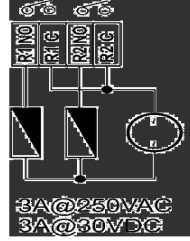


Şekil 5.9. Röle çıkışlarının devresi

Röle çıkışların özellikleri Tablo 5.4'te ve bağlantı şeması Şekil 5.10'da verilmiştir.

Tablo 5.4. Röle çıkışlarının özellikleri

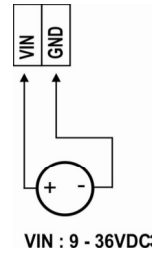
Özellik	Karakteristiği
Röle Çıkışlarının Sayısı	2 adet
Röle kontak çıkışları	Com-NO (Normalde Açık)
Açık Konumda iken max. çekilen akım	3A@250VAC



Şekil 5.10. Röle çıkışların bağlantı şeması

5.1.4. Güç Bağlantı Şeması

Modbus gateway cihazı DC gerilimle, 9-36V aralığında çıkış veren bir anahtarlama modunda çalışan herhangi bir adaptör ile çalışabilmektedir. Eğer röle çıkışları kullanılacaksa, 24V'luk adaptör kullanılmalıdır. Rölelerin çalışması için 24V gerekmektedir. Şekil 5.11'de güç bağlantı şeması verilmiştir.

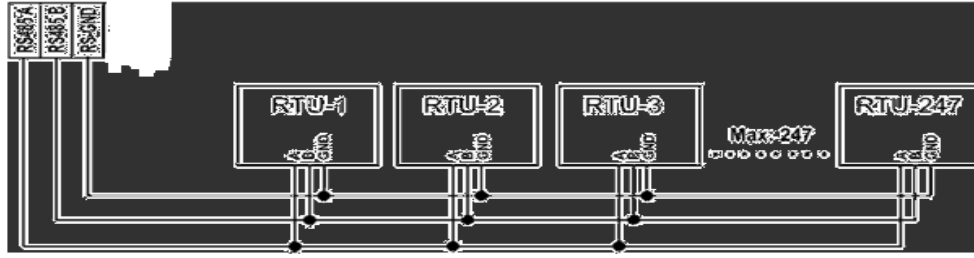


Şekil 5.11. Güç bağlantı şeması

5.1.5. RS485 Bağlantı Şeması

RS485 iki kablo ile haberleşme imkanı sağlayan bir seri haberleşme standardıdır. Bu kablolardan biri Tx (verici) ucuna, diğeri Rx (alıcı) ucuna bağlanmaktadır. Rx ve Tx arasındaki gerilim farkına bakılarak, bu kablolardaki sinyal değerleri ölçülmektedir. Dolayısıyla, ortamda gürültü olma durumunda, her iki kabloda aynı miktarda etkilenmektedir. Bu sebeple, gürültülü sinyal ile gürültüsüz sinyal arasında taşınan bilgi açısından bir fark oluşmamaktadır. Bu özelliğinden dolayı RS485 ile uzun mesafelerde güvenli haberleşme işlemleri sağlanmaktadır.

RS485 bağlantı şeması, Şekil 5.12’de gösterilmiştir. RS485 hattına en fazla 247 adet RTU bağlanabilmektedir. RS485 devresi için ADM2486 entegresi kullanılmıştır.



Şekil 5.12. RS485 bağlantı şeması

5.2. Yazılım Tasarımı

Yapılan çalışmada, belirlenen çerçevede cihazın fonksiyonlarını yerine getirebilmesi için tasarlanan yazılım algoritmaları bu kısımda anlatılacaktır. Tasarlanacak cihaz için belirlenen isterler aşağıda listelenen 6 madde ile sıralanmaktadır.

- 1- Modbus gateway cihazı, Modbus TCP protokolünde Slave olarak çalışırken Modbus RTU protokolünde Master olarak çalışması,
- 2- Modbus gateway cihazına birden fazla istemcinin aynı anda bağlantı sağlayabilmesi ve sorgulama yapabilmesi (Multi-master olarak çalışabilmesi),
- 3- Yapılan sorgulamaların ID önceliğine göre yapılarak, ID'si düşük olan cihazların diğerlerine göre önceliklendirilmesi,
- 4- ID önceliğine göre oluşturulan kuyruk mekanizmasının belirlenen periyotlarda taranarak, kuyruktaki sorguların kuyrukta kalma sürelerinin kontrol edilmesi,
- 5- Kuyrukta kalma süreleri kontrol edilerek belirlenen süreleri aşan sorgulamaların zaman parametresine göre oluşturulan kuyruğa alınması ve her iki kuyruğun güncellenmesi,
- 6- Zaman parametresine göre oluşturulan kuyruğun ID önceliğine göre oluşturulan kuyruğa göre önceliklendirilmesi,

özellikleri tasarlanacak sistemin yerine getirmesi beklenmektedir. Modbus gateway cihazına Şekil 5.1'te görüldüğü gibi birden fazla istemci aynı anda Modbus gateway cihazına bağlantı sağlayabilmektedir. Bu isterlere göre tasarlanan algoritma ve yazılımları aşağıda listelenen 7 madde ile anlatılmıştır.

- 1- Birden fazla bağlantı isteğinin kabul edilmesi için yeterli sayıda soket açılması,
- 2- Bağlantı isteklerinin görülmesi ve aynı zamanda cevap verilmesi için işletim sisteminin multi-thread (çok izlekli) yapısı kullanılarak gelen isteklerin işlenmesi,
- 3- Gelen Modbus TCP paketinin, Modbus RTU paketine dönüştürülmesi ile bağlı bulunan RTU cihazların sorgulanması ve gelen mesaj paketinin bağlı liste yapısı kullanılarak kuyruğa alınması,
- 4- Modbus TCP paketinde bulunan Slave ID değerine göre istemcilerden gelen isteklerin istek kuyruğuna ID önceliğine göre alınması,
- 5- İstemciler tarafından gelen istekler ve kuyruğa alınmış isteklere cevap verilmesi işleminde thread (izlek) mekanizması kullanılması,
- 6- Kullanılan iki thread mekanizmalarından biri gelen isteklerin kuyruğa alınması işleminde kullanılıyorken; diğer thread mekanizmasının kuyruktaki isteklere cevap verilmesi için kullanılması,
- 7- Kuyruğa eleman ekleme ve kuyruktan eleman silme işleminin aynı anda denk gelme tehlikesinden dolayı, mutex yapısı kullanılarak aynı hafızaya aynı anda erişim engellenmesi gibi yazılım tasarımları gerçekleştirilmiştir.

Yazılım tasarımı aşamasında yukarıda bahsedilen fonksiyonlar gerçekleştirilmiştir. Sunucu modunda çalışan cihazımız soket uygulamalarını kullanmaktadır. Bölüm 5.2.1’de soket uygulamalarından bahsedilecektir. Kullanılan thread mekanizmaları ve oluşturulan kuyruk mekanizmaları sırasıyla Bölüm 5.2.2 ve Bölüm 5.2.3’te anlatılacaktır.

5.2.1. Soket Programlama

Süreçler arası haberleşme yöntemlerinden biri de soketlerdir. Soyut anlamda soketler haberleşme uç noktalarıdır. Soketler dosyalara benzemektedirler. Bu sebeple soketten okumak ile dosyadan okumak arasında hiçbir fark yoktur. POSIX (Portable Operating System Interface for Unix) sistemlerde dosyalar üzerinde gerçekleştirilebilen işlemler soket gibi birimler üzerinde de gerçekleştirilebilmektedir. Modbus gateway cihazı tasarımında Linux işletim sistemi kullanılmıştır. Linux işletim sistemi de bir Unix türevidir olduğundan dosyadan okuma özelliği Linux için de geçerlidir. /dev dizini altında her cihaz için major ve minor numaraları ile belirlenmiş özel dosyalar bulunur. Örneğin ses kartı için /dev/audio, Ethernet kartı için /dev/eth0, disk için /dev/fd0 dosyalarıdır. Bu dosyalar gibi terminal, konsol, FIFO ve pipe gibi veri yapıları da kullanıcı için birer dosyadır. Soketler birer dosya olduğundan soketlere yazma ve soketlerden okuma işlemleri için write() ve read() gibi fonksiyonlar kullanılmaktadır.

Soket programlamaya özel bazı terimler aşağıda izah edilmiştir.

İstemci (Client): istediği zaman sunucu ile bağlantı sağlayarak hizmet isteyen, görev veren ve bağlantıyı koparan soket programıdır.

Sunucu (Server): İstemci tarafından gelen isteklere cevap verebilecek şekilde sürekli çalışan, hizmet veren soket programıdır.

Port: Bir bilgisayarda birden fazla soket bulunduğundan bu soketleri birbirinden ayırmak ve istemci-sunucu ikilisini birbiri ile buluşturmak için her sokete bir numara verilmektedir. Bu numaraya port denilmektedir.

IP: İstemci ve sunucu arasında sağlanan haberleşme IP numarası sayesinde gerçekleştirilmektedir. Bir istemci program bağlantı kuracağı sunucuya önce IP numarası ile bağlanmakta daha sonra da port numarasını kullanarak istediği sunucu programı ile temasa geçmektedir.

5.2.1.1. Soket Türleri ve IP Protokolü

Soket türlerinden en çok kullanılanlardan olan "Stream Soketler" ve "Datagram Soketler" den bahsedilecektir. Programlarda soket açarken SOCK_STREAM ve SOCK_DGRAM isimleri ile soket türü belirlenmektedir. Stream soketlere bağlantı yönelimli (connection oriented) soketler, Datagram soketlere ise bağlantısız (connectionless) soketler denir [17].

Stream soketler, TCP/IP protokolünün taşıma katmanında bulunan TCP'yi kullanırken datagram soketler ise UDP (User Datagram Protocol) protokolünü kullanırlar. Bu iki soket türünün özellikleri ve farkları aşağıdaki gibi sıralanmaktadır.

1. Stream soketler verileri sıralı göndermeyi garanti ederken, datagram soketleri garanti etmez.
2. Stream soketler güvenliği garanti eder ancak datagram soketler güvenliği garanti etmezler. Yani bir paket gönderildiği zaman karşı tarafın paketi aldığını bildirmeden paketi göndermiş sayamaz ve tekrar gönderir. Bunun yanında paketin doğruluğunu test etmek için başlık bilgisinde CRC kontrolü yapılmaktadır. UDP protokolünde de CRC kontrolü yapılmaktadır ancak yanlış olursa aynı paketi tekrar göndermez.
3. Stream soketler sürekli bağlantı halinde kalmaktadırlar. Datagram soketler ise sadece veri göndereceği zaman bağlantı kurar ve işi bitince bağlantıyı koparırlar.

5.2.1.2. Uygulamadaki Temel Prensipler

Soketler alıcı ve gönderici olmak üzere iki uca sahiptirler. Gönderilecek mesajlar mantıksal 1 ve 0'a karşılık gelen elektriksel sinyallere dönüştürülürler. Mesajlara protokol başlıkları eklenerek fiziksel katmandan elektriksel sinyaller olarak gönderilirler.

Soket programları istemci veya sunucu modunda çalışabilmektedirler. Bazı soket programları istemci ve sunucu modunda çalışarak her iki görevi de

yapabilmektedirler. İstemci ve sunucu programı arasında bazı farklar vardır. Aşağıdaki Tablo 5.5’te istemci ve sunucu arasındaki işlemler görülmektedir [16].

Tablo 5.5. İstemci ve sunucu arasındaki işlemler

İstemci	Sunucu
İstemci tarafından Soket oluşturulması, socket()	• Sunucu tarafından soket oluştur socket() • Sunucu adres bilgilerini yerleştir “struct sockaddr_in” “server_addr” • Sunucu tarafından Soketi adres ve isim bilgileri ile ilişkilendirme, bind() fonksiyonu ile kullanılması • Sunucunun soketi dinleme moduna geçmesi, listen() fonksiyonu ile
İstemci tarafından bağlantının sağlanması, connect() fonksiyonu ile	Sunucu tarafından bağlantının kabul edilmesi, accept() fonksiyonu ile
İstemci tarafından verinin gönderilmesi, send() fonksiyonu ile	İstemci tarafından gönderilmiş verinin sunucu tarafından alınması, recv() fonksiyonu ile
İstemci tarafından gelen verinin alınması, recv() fonksiyonu ile	Sunucu tarafından verinin göndeilmesi, send() fonksiyonu ile
Veri gönderme ve alma işlemleri	Veri gönderme ve alma işlemleri
...	...
...	...
...	...
Soketin istemci tarafından kapatılması, close() fonksiyonu ile	Soketin kapatılması, close() fonksiyonu ile

Sunucu ve istemci internet adres bilgilerini “sockaddr_in” yapısının içerisinde tutmaktadırlar. Bu yapının parametreleri aşağıdaki gibidir.

```

struct sockaddr_in {
    short            family;

    unsigned short   port;

    struct in_addr   addr;

    char             zero[8];
}

```

Bu yapı içerisinde IP adresi, port numarası ve soket ailesi parametreleri saklanmaktadır. Şekil 5.13'te soket açma ve gerekli parametrelerin atanması işlemi görülmektedir.

```

int main(int argc, char **argv)
{
    char c;
    struct sockaddr_in srv, cli;
    int clifd;
    int tid;
    int i;
    int choosen;

    gpio_init();

    signal(SIGPIPE, SIG_IGN);

    if( (listenfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        perror("sockfd\n");
        exit(1);
    }

    bzero(&srv, sizeof(srv));
    srv.sin_family = AF_INET;
    srv.sin_addr.s_addr = INADDR_ANY;
    //srv.sin_port = htons(PORT);
    srv.sin_port = htons(atoi(argv[1]));
}

```

Soket Ailesi

Soket Tipi

Parametrelerin
srv yapısına atanması

Şekil 5.13. Soket açma

Soket uygulamaları ile ilgili parametreler “sys/types.h” ve “sys/socket.h” başlık dosyalarında tanımlanmıştır. Dolayısıyla soket uygulamaları ile ilgili çalışmalarda bu başlık dosyalarının kullanılması gerekmektedir.

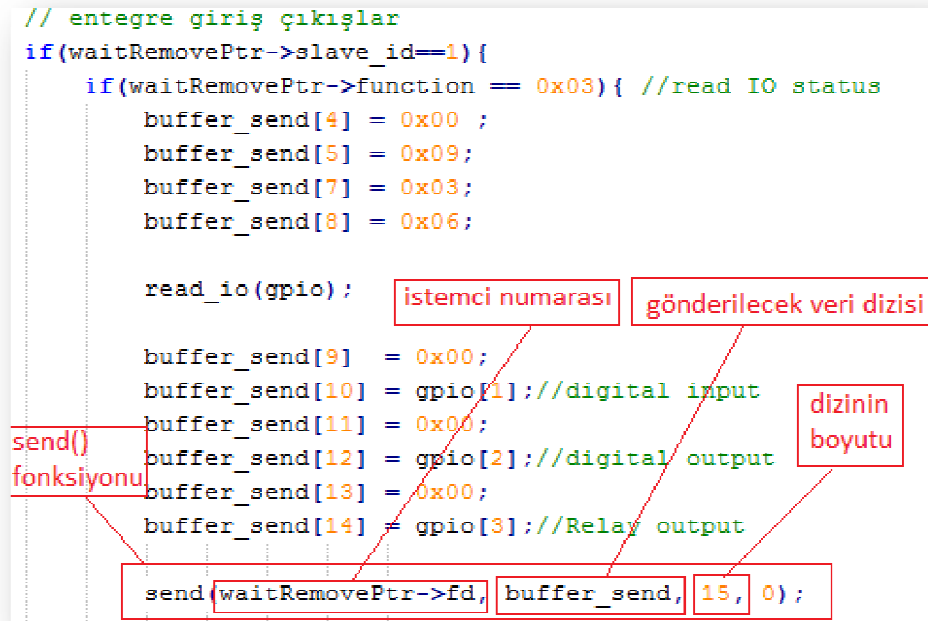
Veri Alış-verişi:

İki soket arasında tanımlanmış protokollere göre veri haberleşmesi yapılabilir. Soketler sayesinde veri gönderme ve alma işlemleri için send() ve recv() fonksiyonları kullanılabilir. Aşağıda send() ve recv() fonksiyonları ile parametreleri belirtilmiştir [16].

```
int send(int s, const void *message, int length, unsigned int flags);
```

```
int recv(int s, const void * message, int length, unsigned int flags);
```

send() ve recv() fonksiyonlarının kullanımı Şekil 5.14 ve Şekil 5.15'te görülmektedir.



```
// entegre giriş çıkışlar
if(waitRemovePtr->slave_id==1){
    if(waitRemovePtr->function == 0x03){ //read IO status
        buffer_send[4] = 0x00 ;
        buffer_send[5] = 0x09;
        buffer_send[7] = 0x03;
        buffer_send[8] = 0x06;

        read_io(gpio);

        buffer_send[9] = 0x00;
        buffer_send[10] = gpio[1];//digital input
        buffer_send[11] = 0x00;
        buffer_send[12] = gpio[2];//digital output
        buffer_send[13] = 0x00;
        buffer_send[14] = gpio[3]//Relay output

        send(waitRemovePtr->fd, buffer_send, 15, 0);
    }
}
```

Annotations in the image:

- send() fonksiyonu**: Points to the `send` function call.
- istemci numarası**: Points to the first argument `waitRemovePtr->fd`.
- gönderilecek veri dizisi**: Points to the second argument `buffer_send`.
- dizinin boyutu**: Points to the third argument `15`.

Şekil 5.14. send() fonksiyonu kullanımı

```

for(i = 0; i < MAX_CLIENT_PER_THREAD; i++)
{
    if(threads[tid].clients[i] != 0)
    {
        n = recv(threads[tid].clients[i], buffer, MAX_CLIENT_BUFFER, 0);
        if(n == 0)
        {
            printf("client %d closed connection 0\n", threads[tid].clients[i]);
            threads[tid].clients[i] = 0;
            threads[tid].client_count--;
            memset((char *) &buffer, 0, sizeof(buffer));
        }
    }
}

```

istemci numarası

alınan veri dizisi

recv() fonksiyonu

alınan verinin boyutu

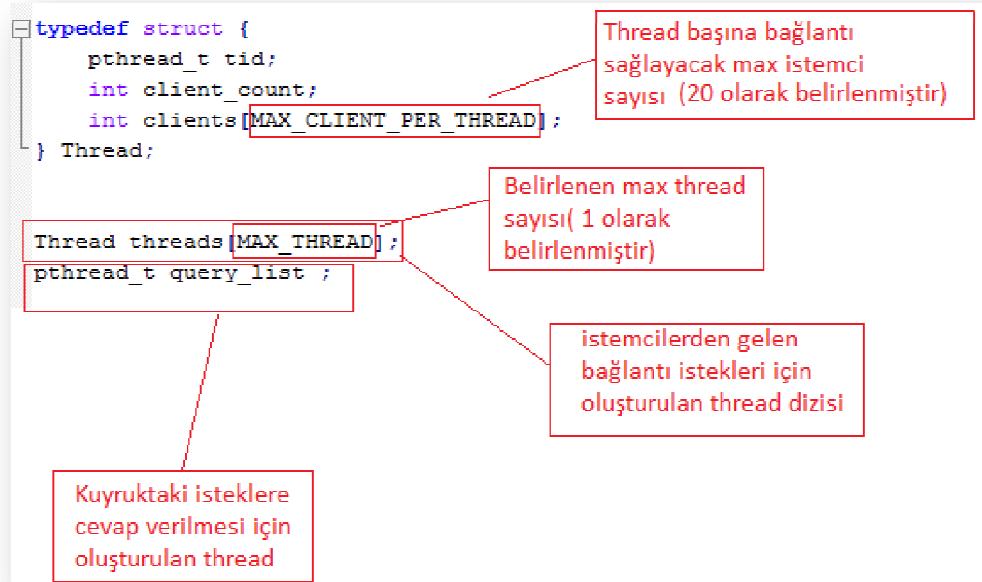
Şekil 5.15. recv() fonksiyonu kullanımı

5.2.2. Threads (İzlekler)

Yapılan çalışmada birden fazla istemcinin bağlantı kurabilmesi, bu isteklerin kuyruğa alınması, kuyruktan çekilmesi gibi işlemlerin yapılabilmesi için bu işlerin iş parçacıklarına ayrılması gerekmektedir. Bu işlemi gerçekleştirmek için thread mekanizmaları kullanılmıştır. Bir işin eş zamanlı olarak işlenen her bölümüne iş parçacığı olarak tanımlanmaktadır. İş parçacığı olarak tanımlanan thread yapısı, iş parçacığının oluşturulduğu ve kaynakların paylaşılması işlemlerinde kullanılmaktadır. İş parçacıkları, genellikle işlemlerin içinde yer alırlar. Çoklu iş parçacıkları, çoğu bilgisayar sistemlerinde uygulanabilmektedir. Tek işlemci kullanılan durumlarda birden fazla iş parçacıklı uygulamalar, zaman dilimleme yöntemi ile gerçekleştirilmektedir. Bu uygulamalarda, tek işlemci ile farklı iş parçacıkları arasında çok hızlı geçiş yapılarak işlemler arasında eş zamanlı çalışıyormuş izlenimi vermeye çalışılmaktadır. Çok işlemcili sistemlerde ise farklı iş parçacıkları farklı işlemciler üzerinde eş zamanlı olarak çalışabilirler [18].

Yapılan çalışmada istemci tarafından gelen isteklerin kuyruğa alınması için threadler oluşturulmuştur. Oluşturulan bu threadler sayesinde gelen istekler bir kuyruğa alınmaktadır. İstemcilerden gelen istekler için bir adet thread tanımlanarak, 20 adet istemci bağlanmasına izin verilmiştir. Aynı şekilde, kuyruktaki istekler de cevap verildikçe silinmektedirler. Kuyruktan çekme işlemi için de bir adet thread tanımlanmıştır. Kuyrukta eleman olduğu sürece işlem devam etmektedir. Şekil

5.16’da gelen isteklerin kuyruğa alınması ve kuyruktaki isteklere cevap verilmesi için oluşturulan thread mekanizmalarının tanımlanması görülmektedir.



Şekil 5.16. Oluşturulan thread mekanizmaları

Sonuç olarak iki adet thread tanımlanmıştır. Biri gelen istekleri kuyruğa alma işleminde kullanılmaktadır. Diğeri ise kuyruktan istekleri çekerek cevap vermektedir. Kuyruğa alma ve kuyruktan çekme işlemleri için tanımlanan threadler aynı hafızaya erişmektedirler. Yani biri kuyruğa eleman eklerken diğeri kuyruktan eleman silmektedir. Bu işlemin aynı anda olmasını engellemek ve yapılan işlemi diğerrinin beklemesini sağlamak için kuyruğa alma ve kuyruktan çekme işlemlerinde mutex (mutual exclusion) yapıları kullanılmıştır. Mutex aynı anda okuma ve yazma yapılabilen bir veri tabanına, aynı anda sadece bir threadin erişimine izin veren bir senkrenizasyon yapısıdır. Bu sayede, mutex kullanılarak kuyruğa ekleme ve kuyruktan çekme işlemleri ayrı zamanlarda gerçekleştirilmiştir. Şekil 5.17’de tanımlanan mutex yapıları görülmektedir.

```

Thread threads[MAX_THREAD];
pthread_t query_list;

pthread_cond_t new_connection_cond = PTHREAD_COND_INITIALIZER;

pthread_mutex_t new_connection_mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t queue_lock = PTHREAD_MUTEX_INITIALIZER;

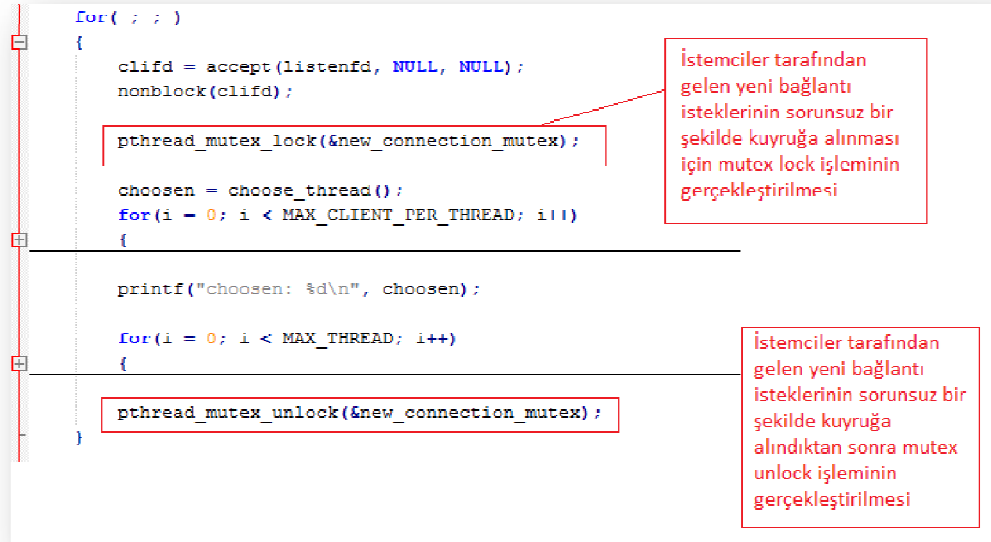
```

Kuyruktan cevap
verilerek silme işleminin
yapılabilmesi için
kuyruğu kitlemek için
tanımlanan mutex yapısı

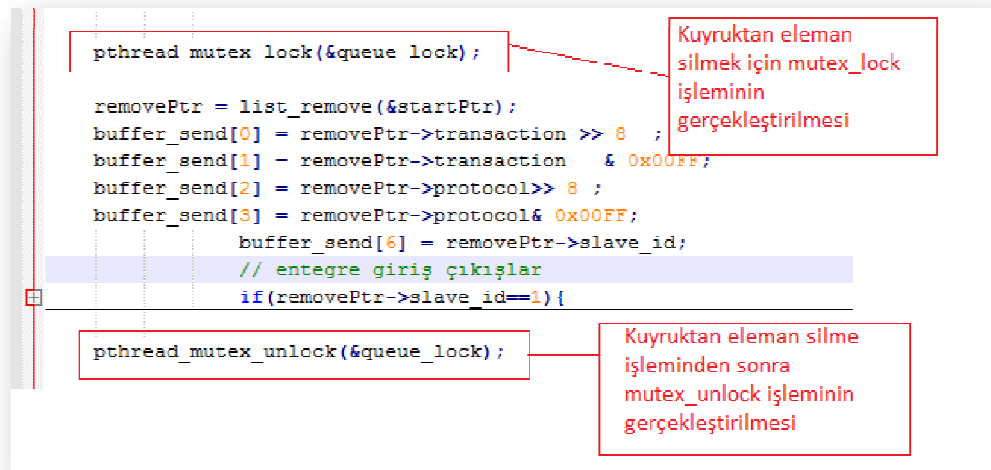
Yeni bir bağlantının
kuyruğa alınması için
tanımlanan mutex

Şekil 5.17. Oluşturulan mutex yapıları

Şekil 5.18’de istemciler tarafından gelen bağlantı isteklerinin kuyruğa alınması işlemi için kullanılan mutex_lock ve mutex_unlock işlemleri görülmektedir. Kullanılan mutex_lock ve mutex_unlock yapıları ile aynı anda kuyruğa eleman ekleme ve eleman silme işlemlerinin çakışması önlenmiştir. Bu sayede, istemciler tarafından yeni bağlantı isteklerinin sorunsuz bir şekilde kuyruğa alınması ve işlem tamamlanınca kuyruğun serbest bırakılması gerçekleştirilmiştir. Aynı şekilde Şekil 5.19’da da kuyruktan eleman silme işlemi için kullanılan mutex_lock ve mutex_unlock yapıları görülmektedir.



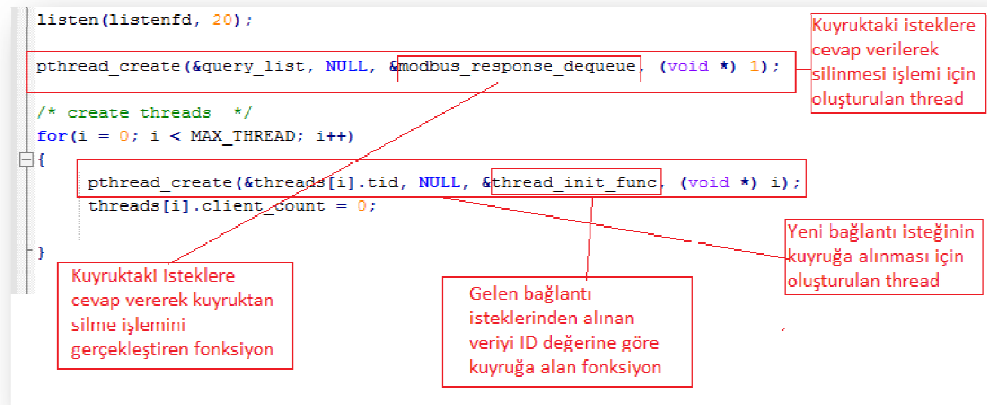
Şekil 5.18. Mutex lock ve mutex unlock işlemlerinin bağlantı durumunda kullanılması



Şekil 5.19. Mutex lock ve mutex unlock işlemlerinin kuyruktan eleman silerken kullanımı

Belirlenen isterler doğrultusunda thread ve mutex tanımları yapılarak tanımlanan threadler oluşturulduktan sonra; istemciler tarafından bağlantı isteğinin ve kuyruğa cevap verilmek üzere bekleyen isteklerin meydana gelmesi ile her bir olay için thread mekanizması ile tanımlanan fonksiyonlar çağrılmaktadır. Şekil

5.20’de görüldüğü gibi gelen isteklerin kuyruğa alınması ve kuyruktaki isteklere cevap verilerek silinmesi işlemleri için threadler oluşturulmuştur. Yeni bağlantı isteği oluşması durumunda “thread_init_func” fonksiyonu çağrılmaktadır. Bu fonksiyon ile gelen Modbus TCP paketindeki veriler ayrıştırılarak bir veri yapısına atanmaktadır. Daha sonra Slave ID değerine göre küçükten büyüğe doğru sıralanacak şekilde kuyruğa eklenmektedir. Aynı şekilde kuyruktaki isteklere cevap verilmesi işlemi için oluşturulan thread ile “modbus_response_dequeue” fonksiyonu çağrılmaktadır. Bu fonksiyon ile kuyrukta istek olduğu sürece kuyruğun ilk elemanından başlayarak cevap verme ve devamında kuyruktan silme işlemleri yapılmaktadır. Ayrıca belirlenen periyotlarda kuyruğun taranarak kuyrukta gereğinden fazla bekleyen isteklerin ikinci bir kuyruğa alınması işleminin gerçekleştirilmesi, sonrasında ikinci kuyrukta eleman olduğu sürece ikinci kuyruğun birinci kuyruğa göre önceliklendirilmesi işlemleri gerçekleştirilmektedir. İkinci bir kuyruğa olan gereksinimin nedeni Bölüm 5.2.3.1’de isteklerin kuyruğa eklenmesi işlemi konusunda izah edilmiştir.

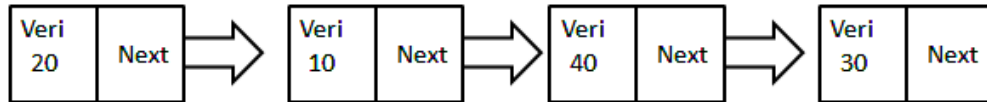


Şekil 5.20. Kuyruğa alma ve kuyruktan eleman silme işlemleri için thread'lerin oluşturulması

Sistem gereksinimleri göz önünde bulundurularak oluşturulan thread mekanizmaları ve çağrılan fonksiyonlar yukarıda anlatılmıştır. Bölüm 5.2.3’te gelen isteklerin kuyruğa alınması ve kuyruktaki isteklerin cevap verilerek kuyruktan silinmesi işlemleri için kullanılan algoritmalar anlatılacaktır.

5.2.3. İstekleri Kuyruğa Alma ve Kuyruktan Silme İşlemi

Yapılan çalışmada istemciler tarafından gelen istekler, bağlı liste (linked list) yapısı kullanılarak bir kuyruğa eklenmiştir. Bağlı liste, dinamik bellek kullanımı sağlamak amacıyla oluşturulmuş veri yapılarıdır. Bağlı liste yapısı yerine dizi kullanıldığında ise dizinin elemanları sabit olmaktadır. Yani diziyi tanımlarken boyutu belirtilmek zorundadır. Bağlı listeler sayesinde, boyutun başlangıçta bilinmediği durumlarda fazladan eleman eklenemeyecek ve gereksiz yere hafıza ayrılması önlenmiş olacaktır. Dizi kullanma mantığı kısaca şöyledir. Örneğin C'de bir dizi tanımlandığı zaman dizinin adı (Array[]) aynı zamanda dizinin RAM'deki başlangıç bellek adresini tutar ve başlangıç değeri kullanılarak dizinin istenen indisine ulaşılabilir. Dizilerde, dizi elemanları RAM'de art arda bulunduğundan basit bir algoritma ile bütün elemanları gezilebilir. Bağlı listeler ise dizilerden farklıdır. Şekil 5.21'de bir bağlı liste örneği görülmektedir. Şekilde de görüldüğü gibi listedeki her eleman veri ve adres tutucu (next) değişkenlerine sahiptir. Başlangıç adresi ve her bir elemanın adres tutucu değeri bilindiğinden liste elemanları üzerinde gezilebilir. Listenin sonlandığı adres tutucu değerinin boş (NULL) olduğu görüldüğünde anlaşılmaktadır.



Şekil 5.21. Bağlı liste

Yapılan çalışmada istemcilerden gelen sorgulamalar için bağlı liste yapısı kullanılarak gelen istekler Slave ID değerine göre kuyruğa alınmıştır. Kuyruğa alınma işleminde, ID önceliğine göre, küçükten büyüğe doğru sıralanarak kuyruğa alınmaktadır. İstemcilerden gelen veri istek paketi, Şekil 5.22'de görüldüğü gibi “Modbus” adında bir yapıya aktararak ayrıştırılmaktadır. Ayrıştırılma işleminden sonra Slave ID değerine göre kuyruğa eklenmektedir. Daha sonra sırayla kuyruktaki her bir isteğe cevap verilerek kuyruktan silinmektedirler. Gelen istek paketindeki veriler Modbus” yapısına Şekil 5.22’de görülen parametrelere ayrıştırılarak aktarılmaktadır.

```

typedef struct Modbus {
    // socket
    int fd;
    // used for MODBUS TCP as transaction number
    int transaction;
    // modbus TCP protocol 000
    int protocol;
    // packet length
    int length;
    //rtu slave ID in the top packet
    int slave_id;
    // function code in the tcp packet
    char function;
    // data length in the tcp packet
    char data_len;
    // Register Adress
    char reg_adress;
    // number of register
    char reg_num;
    // value for writing
    char reg_val;
    // IP address
    char ip[16];
    // TCP port */
    int port;
    /* pointer to next node*/
    struct Modbus *nextPtr;
    //// kayıt zamani
    struct timeval tpstart;
} pModbus_t;

```

Şekil 5.22. Veri yapısı parametreleri

5.2.3.1. İsteklerin Kuyruğa Eklenmesi İşlemi

İstemciler tarafından gelen istek paketi ayrıştırılarak veri yapısına aktarıldıktan sonra, Slave ID değerine göre kuyruğa alınarak sıralanmaktadır. Kuyruğa alınma işleminde bağlı liste yapısı kullanılmıştır. Şekil 5.23'te görüldüğü gibi “list_add(ModbusListNodePtr *sPtr, ModbusListNodePtr newPtr)” fonksiyonu ile istekler kuyruğa eklenmektedir.

```

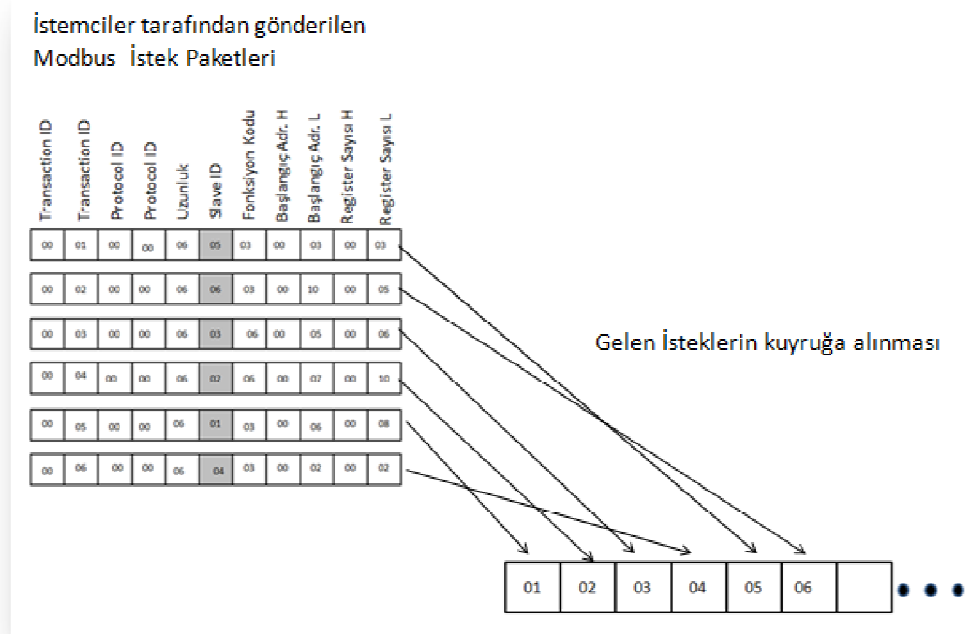
void list_add( ModbusListNodePtr *sPtr, ModbusListNodePtr newPtr )
{
    int value = newPtr->slave_id;
    ModbusListNodePtr previousPtr; /* pointer to previous node in list */
    ModbusListNodePtr currentPtr; /* pointer to current node in list */
    if ( newPtr != NULL )
    { /* is space available */
        newPtr->slave_id = value; /* place value in node */
        newPtr->nextPtr = NULL; /* node does not link to another node */
        previousPtr = NULL;
        currentPtr = *sPtr;
        /* loop to find the correct location in the list */
        while ( currentPtr != NULL && value >= currentPtr->slave_id )
        {
            previousPtr = currentPtr; /* walk to ... */
            currentPtr = currentPtr->nextPtr; /* ... next node */
        } /* end while */
        if ( previousPtr == NULL ) /* list_add new node at beginning of list */
        {
            newPtr->nextPtr = *sPtr;
            *sPtr = newPtr;
        } /* end if */
        else
        { /* list_add new node between previousPtr and currentPtr */
            previousPtr->nextPtr = newPtr;
            newPtr->nextPtr = currentPtr;
        } /* end else */
    } /* end if */
    else
    {
        printf( "%d not list_added. No memory available.\n", value );
    } /* end else */
} /* end function list_add */

```

Slave ID değerine göre
Küçükten büyüğe doğru
sıralanarak kuyruğa
eklenmesi

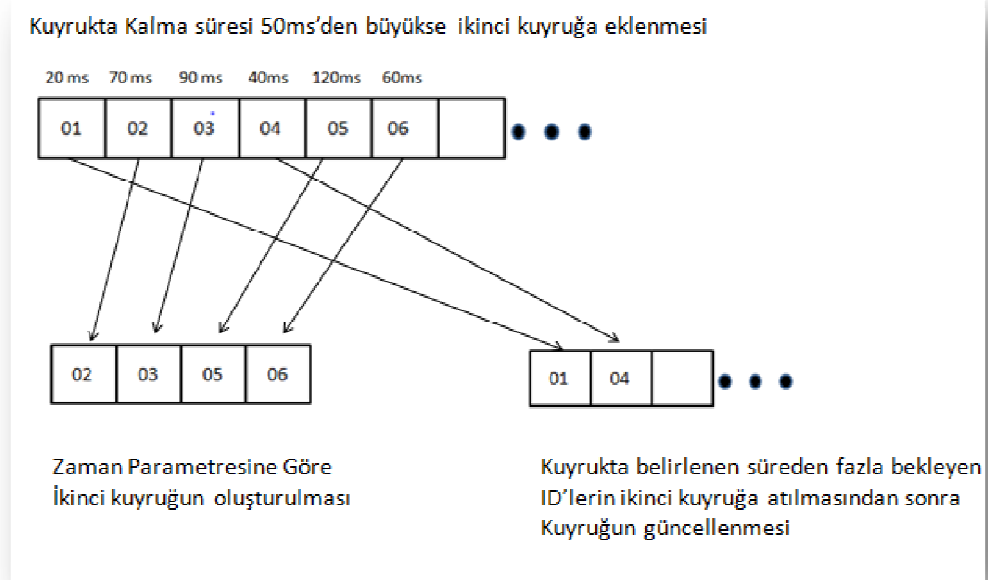
Şekil 5.23. İstekleri kuyruğa ekleme fonksiyonu

Gelen Modbus veri paketinin ayrıştırılarak Slave ID değerine göre kuyruğa alınması işlemi Şekil 5.24'te örneklendirilmiştir.



Şekil 5.24. İsteklerin kuyruğa alınması

Burada kuyruğa alınan istekler Slave ID değerine göre küçükten büyüğe doğru sıralanarak alınmaktadır. Kuyruktan çekme işleminde ise her zaman kuyruğun ilk elemanı çekilmektedir. İstemci sayısının fazla olduğu durumlar düşünüldüğünde, düşük ID'li RTU'lara gelen isteklerin sayısı arttığında kuyruktaki yüksek ID'li RTU'lar kuyrukta gereğinden fazla bekleme durumu ile karşılaşabilirler. Bu durumda yüksek ID'li RTU'lara cevap verilmeme durumu ortaya çıkabilmekte veya sorgulama zaman aşımına uğrayabilmektedir. Bu problemin çözümü için ikinci bir kuyruk oluşturulmuştur. Belirlenen zaman aralıklarında ID değerine göre oluşturulan kuyruk taranarak, her bir RTU'nun kuyrukta kalma süreleri kontrol edilir. Kuyruktaki RTU'lardan kuyrukta belirlenen süreden fazla kalan varsa ikinci kuyruğa eklenir. Cevap verme işleminde ise, zamana göre oluşturulan kuyrukta eleman olduğu sürece cevap verme işlemi bu kuyruktan yapılacağından zaman aşımı problemi de ortadan kaldırılarak daha güvenilir bir sorgulama işlemi gerçekleştirilmiştir. Şekil 5.25'te, ID değerine göre oluşturulan kuyruğun taranarak her bir isteğin kuyrukta kalma süresi belirlenen süre ile karşılaştırılmaktadır.



Şekil 5.25. Kuyruğun belirlenen periyotlarda taranarak güncellenmesi

Şekil 5.25'te görüldüğü gibi Şekil 5.24'teki uygulamada ID değerine göre oluşturulan kuyruğun tarama sıklığı tarama esnasında 1500 ms'den büyükse tarama işlemi gerçekleştirilmektedir. Tarama işleminde, Slave ID değerine göre oluşturulmuş kuyrukta bekleme süresi 50 ms'den fazla olan elemanlar, zaman parametresine göre oluşturulan kuyruğa alınmaktadır. Daha sonra kuyruklar güncellenmektedir. Cevap verme işleminde ise zaman parametresine göre oluşturulan kuyruk önceliklidir. Bu şekilde sorgulama işleminde zaman aşımı problemi ortadan kaldırılarak ID önceliklendirme özelliği ile daha güvenli sorgulama işlemi sisteme eklenmiştir.

5.2.3.2. Kuyruktan Eleman Silme İşlemi

İstemciler tarafından gelen istekler kuyruğa alındıktan sonra ID önceliğine göre cevap verilerek kuyruktan silinmektedirler. Kuyruktan silme işlemi için `list_remove()` fonksiyonu kullanılmaktadır. Kullanılan fonksiyon Şekil 5.26'da görülmektedir.

```

/* list_remove a list element */
ModbusListNodePtr list_remove( ModbusListNodePtr *sPtr)//, char value )
{
    ModbusListNodePtr tempPtr;    /* temporary node pointer */
    int value;

    value = ( *sPtr )->slave_id ;
    tempPtr = *sPtr; /* hold onto node being removed */
    *sPtr = ( *sPtr )->nextPtr; /* de-thread the node */
    //free( tempPtr ); /* free the de-threaded node */
    return tempPtr;
} /* end function list_remove */

```

Şekil 5.26. Kuyruktan eleman silme fonksiyonu

5.3. Sistemin Çalıştırılması ve Sonuçlar

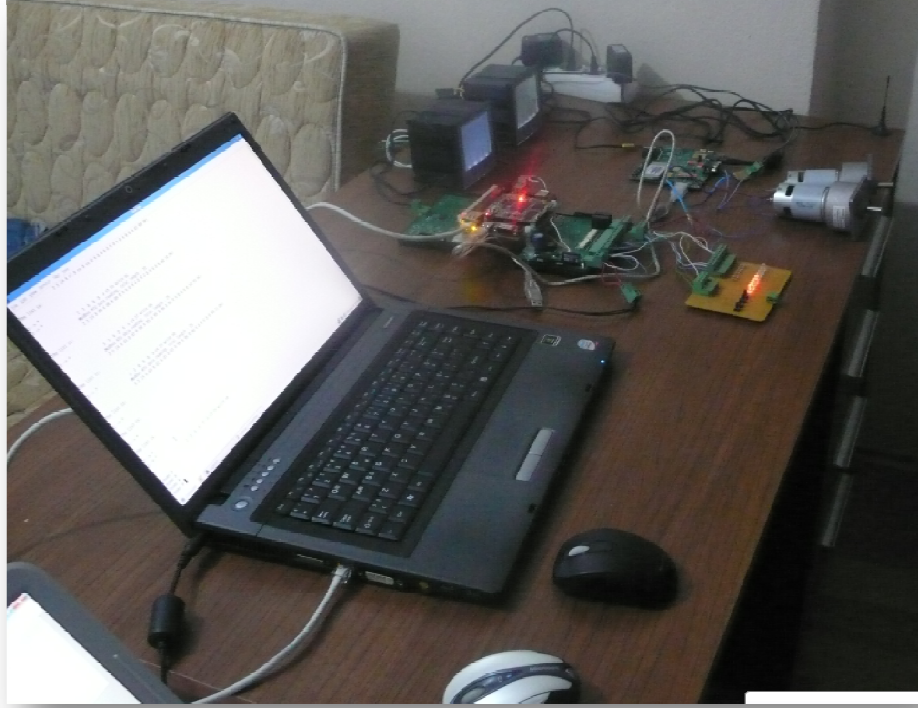
Donanım ve yazılım tasarım aşamalarından sonra çalıştırılmaya hazır hale getirilen sistemin denenmesi için uygun test ortamı oluşturularak çalıştırılması işlemi gerçekleştirilmiştir. Bölüm 5.3.1’de test ortamı ve sistemin çalıştırılması, Bölüm 5.3.2’de ise sonuçlar anlatılacaktır.

5.3.1. Test Ortamı ve Sistemin çalıştırılması

Bu çalışmada tasarlanan Modbus gateway cihazının çalışan bir sistemde kullanımı ise Şekil 5.27’de verilmiştir. Modbus gateway cihazına Ethernet portundan Telnet ile bağlanan bilgisayar sayesinde, istemciler tarafından gelen isteklerin kuyruğa alındığı gibi cihaz üzerindeki işlemler bilgisayar ekranında görülebilmektedir.

Tasarlanan cihazın, tanımlanan fonksiyonları doğru olarak gerçekleştirip gerçekleştirmediğini anlamak amacıyla bir test düzeneği oluşturulmuştur. Kurgulanan test düzeneği Şekil 5.28’de ve Şekil 5.29’da gösterilmiştir. Şekil 5.28’de istemci olarak kullanılan 3 adet bilgisayar ve ekran olarak kullanılan bir bilgisayar ile sistemin çalışma ortamı görülmektedir. Bu 3 adet bilgisayar ile endüstriyel uygulamalarda kullanılan Modbus TCP ve Modbus RTU protokollerini destekleyen

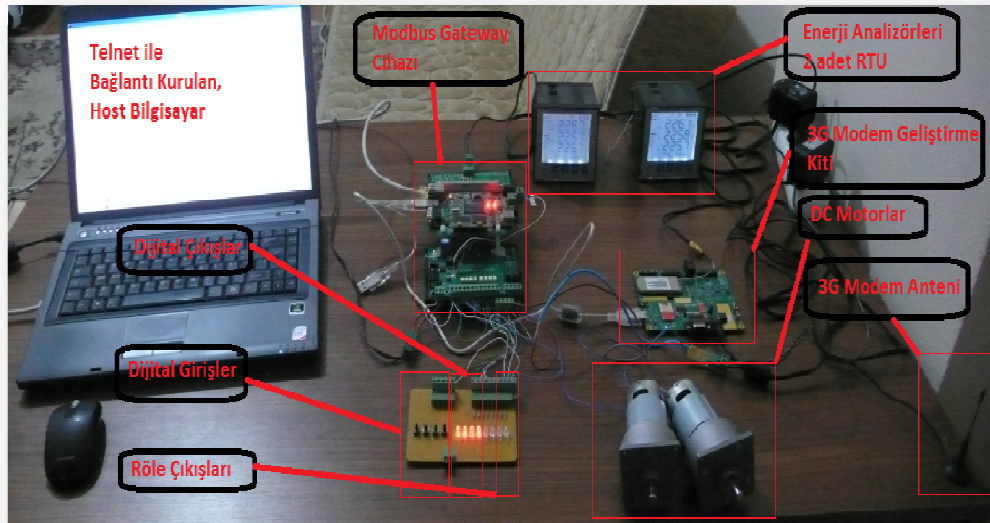
Modpoll programı kullanılarak sorgulama işlemi gerçekleştirilmiştir. Şekil 5.29'da test için kullanılan fiziksel girişler ve çıkışlar gösterilmiştir. RTU olarak 2 adet enerji analizörü kullanılmıştır. Test işlemleri için dijital girişlere butonlar, dijital çıkışlara ledler ve röle çıkışlarına ledler veya 2 adet DC motor bağlanarak çalıştırılmıştır.



Şekil 5.27. Çalışan sistemin görünümü



Şekil 5.28. Test ortamı



Şekil 5.29. Test için kullanılan fiziksel giriş/çıkışlar.

Her bir istemci ile bağılı bulunan 2 adet RTU cihaz ve Modbus gateway cihazı üzerindeki entegre giriş çıkışlar da sorgulanmıştır. Yapılan sorgulama işlemi Modpoll yazılımı ile yapılmıştır. Cihaz üzerindeki entegre giriş çıkışlar 0-2 aralığındaki yazmaçlara map edilmiştir. Modpoll programında cihaz üzerindeki dijital girişler, dijital çıkışlar ve röle çıkışlarını sorgulama işlemi, Şekil 5.30'da görülmektedir. Modbus gateway cihazı üzerinde dijital girişler “0”, dijital çıkışlar “1” ve röle çıkışları ise “2” numaralı adreslerde ondalık değer olarak tutulmaktadır. Dijital giriş değerleri Tablo 5.6, dijital çıkışlar Tablo 5.7 ve röle çıkışları Tablo 5.8’de gösterilmiştir.

Tablo 5.6. Dijital girişlerin ayrıştırılması

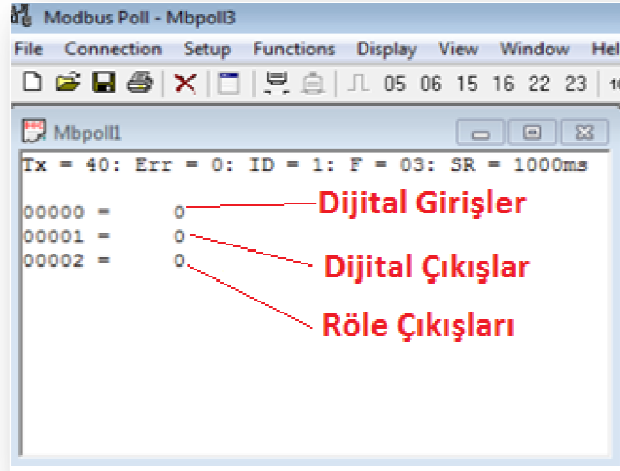
“0” Adresindeki Değer	Dijital Giriş IV	Dijital Giriş III	Dijital Giriş II	Dijital Giriş I
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Tablo 5.7. Dijital çıkışların ayrıştırılması

“1” Adresindeki Değer	Dijital Çıkış IV	Dijital Çıkış III	Dijital Çıkış II	Dijital Çıkış I
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

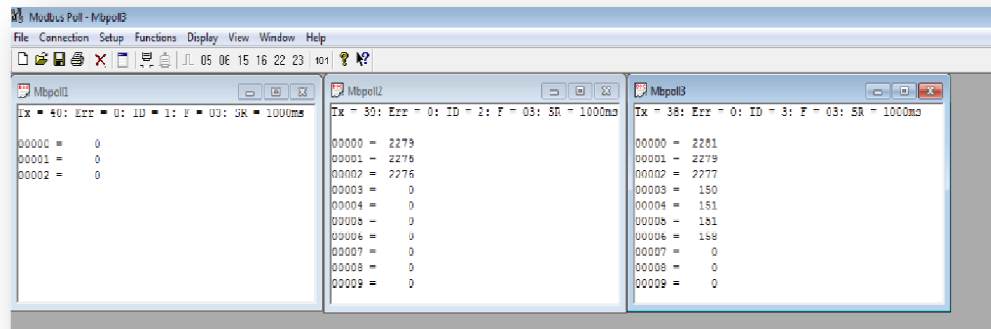
Tablo 5.8. Röle çıkışların ayrıştırılması

“2” Adresindeki Değer	Röle Çıkışı II	Röle Çıkışı I
0	0	0
1	0	1
2	1	0
3	1	1



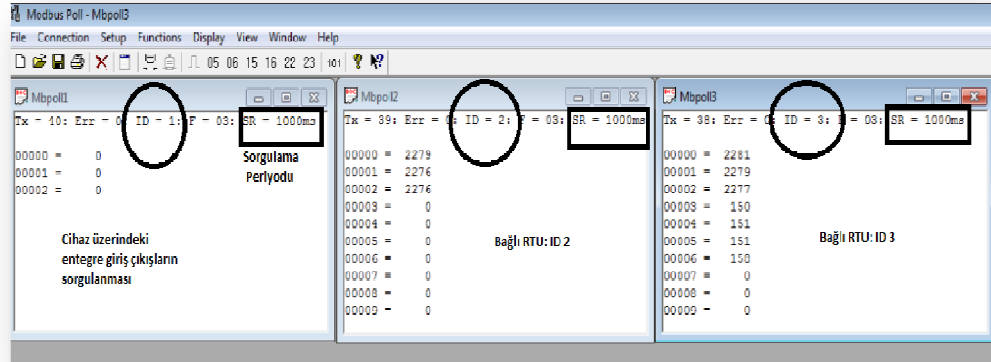
Şekil 5.30. Entegre giriş çıkışların sorgulanması

Aynı şekilde bağlı bulunan RTU cihazların parametrelerinin sorgulanması da Şekil 5.31’de gösterilmiştir. Şekil 5.31’de 3 adet sorgu ekranı görülmektedir. Bu ekranların her biri sırayla RTU ID değeri 1, 2 ve 3 olan RTU cihazlarının resimde görülen adreslerinin sorgulanması işlemini göstermektedir. Şekil 5.31’de ID değeri 1 olan sorgulama ekranı Modbus gateway cihazı üzerindeki entegre giriş/çıkışların sorgulanması işlemini göstermektedir. ID değeri 2 ve 3 olan sorgulama ekranlarında ise Modbus gateway cihazına bağlı RTU olarak kullanılan enerji analizörlerinin 0x0000 ile 0x0009 adresleri arasındaki yazmaçlarının sorgulanması işlemi gösterilmektedir.



Şekil 5.31. Modpoll sorgulama ekranı

Modpoll program ile entegre giriş çıkışlar ve RTU'ların sorgulanması işleminde ekrandaki parametreler ve açıklamaları Şekil 5.32'de görülmektedir.



Şekil 5.32. Modpoll programı ile yapılan sorgulama işleminde parametrelerin anlamı

5.3.2. Sonuçlar

Cihaz telnet ile bağlantı kuran bilgisayarda ekrana bastırılan, kuyruktaki RTU ID sayıları Şekil 5.33'te görüldüğü gibidir.

```

The list is:
1--> * *
2--> * *
3--> *

The list is:
1--> *
2--> * *
3--> *

The list is:
2--> * *
3--> *

The list is:
2--> *
3--> *

```

Şekil 5.33. Kuyruğa eklenen modbus istekleri

Burada her bir RTU için gelen istekler bir adet “*” işareti ile belirlenmiştir. İlgili RTU’ya ait istek sayısı kadar yanına “*” basılmıştır. Kuyrukta bulunan RTU ID değerlerine baktığımızda ID’si düşük olanlara önce cevap verilip kuyruktan silindiği görülmektedir. ID önceliğine göre oluşturulan kuyrukta düşük ID’li RTU’ların sorgulanması sıklaştığında kuyruktaki yüksek ID’li RTU’lara zaman aşımından dolayı cevap verilememektedir. Bu problemi ortadan kaldırmak için belli periyotlarla kuyruk kontrol edilerek, belirlenen süreden fazla kuyrukta kalmış eleman varsa ikinci kuyruğa alınmıştır. Şekil 5.34’te belirlenen periyotlarda kuyruk kontrol edilmekte ve belirlenen süreden fazla kuyrukta kalan RTU ikinci kuyruğa atılmaktadır. Kuyruğu kontrol etme süresi, kuyrukta eleman olduğunda 1500 ms olarak belirlenmiştir.

```

-----IT is TIME OF CHECKING-----
Time PERIOD OF CHECKING :1996 ms
1--> *

1--> *

-----IT is TIME OF CHECKING-----
Time PERIOD OF CHECKING :2016 ms
1--> *

1--> *

-----IT is TIME OF CHECKING-----
Time PERIOD OF CHECKING :1996 ms
1--> *

1--> *

-----IT is TIME OF CHECKING-----
Time PERIOD OF CHECKING :1996 ms
1--> *

1--> *

```

Şekil 5.34. Kuyruğun kontrol edilmesi

Şekil 5.34'te görüldüğü gibi 1500 ms'den fazla süre geçince kuyruk kontrol edilmektedir. Şekil 5.35'te ise kuyruğu kontrol etme esnasında kuyrukta 50 ms'den fazla kalan RTU varsa ikinci kuyruğa eklenmesi gösterilmiştir.

```

-----IT is TIME OF CHECKING-----
Time PERIOD OF CHECKING :1642 ms
-----QUEUE is REFRESHING*****
Slave ID : 1 --- Time in Queue: 12 ms
Slave ID : 2 --- Time in Queue: 193 ms

1--> *
2--> *

-----RTU ID in WAIT QUEEU-----

2--> *

```

Şekil 5.35. Kuyruğun güncellenmesi

Yukarıda görüldüğü gibi kuyruğu kontrol etme esnasında kuyrukta RTU 1 ve RTU 2 bulunmaktadır. RTU 2'nin kuyrukta bekleme süresi olan 193 ms,

belirlediğimiz 50 ms değerinden büyük olduğundan, zaman parametresine göre oluşturulan ikinci kuyruğa alınmıştır.

6. SONUÇ VE ÖNERİLER

Yapılan çalışmada; Linux tabanlı, 3G haberleşme özellikli, akıllı modbus mesaj önceliklendirme mekanizmalı Modbus gateway cihazı tasarlanmıştır. Tasarlanan cihaz, bağlı bulunan RTU cihazlara erişmek için bir köprü olarak kullanılmaktadır. Modbus gateway cihazına bağlı RTU'ların sorgulanmasının yanında, cihaz üzerindeki entegre giriş/çıkışlar sayesinde de cihazın bulunduğu ortamda kontrol ve izleme imkanı sağlanmaktadır. Cihazın giriş/çıkışlarının sorgulanması ve bağlı RTU'ların sorgulanma işleminde, ID önceliğine göre sorgulama yapılarak, ID'si düşük olan RTU'ların sorgulanması önceliklendirilmiştir. Modbus gateway cihazının ID değeri "1" olarak seçildiğinden cihaz her zaman yapılan sorgulamalarda önceliklidir.

Yapılan çalışma ile endüstriyel iletişim protokollerinden biri olan Modbus protokolü için bazı fonksiyonlar gerçekleştirilmiştir. Çalışmada gömülü Linux kullanılması ve 3G haberleşme özelliğinin eklenmesi, geliştirilebilir olmasını ve aynı platforma farklı özelliklerin eklenebilirliğini arttırmıştır. Bu özelliklerle beraber RS485 hattına bağlı RTU parametreleri sorgulanmıştır. RS485 hattına bağlı RTU'ların sorgulanması işleminin yanında, Modbus TCP protokolünde Slave olarak çalışan Modbus gateway cihazı üzerindeki entegre giriş/çıkışların sorgulanması işlemi de gerçekleştirilmiştir.

Sunucu olarak çalışan cihaza birden fazla istemci bağlanabilmekte ve isteyen istemci istediği RTU parametresini sorgulayabilmektedir. Sorgulama işleminde birden fazla istemcinin aynı RTU'ya ait yazmaçları okuması veya istediği yazmaca yazması gerçekleştirilebilmektedir. Birden fazla kullanıcının bağlı kurabilmesinden dolayı, Modbus gateway cihazının multimaster olarak çalışabilmesi sağlanmıştır. Bu işlemleri yaparken kullanılan kuyruk yapısı ile istekler kuyruğa alınmakta, cevap verildikçe kuyruktan silinmektedirler. İstekler kuyrukta ID önceliğine göre sıralanmakta olduğundan, düşük ID'li sorgulama işlemleri diğerlerine göre daha gerçek zamanlı olma özelliğine sahiptirler.

Mevcut sistemde kuyruk yapısı belli bir kullanıcı sayısına kadar problem oluşturmamaktadır. Ancak istemci sayısı arttıkça düşük ID'li isteklere cevap verilmesi işlemi öncelikli olduğu için, kuyrukta düşük ID'li istek olduğu sürece, yüksek ID'li istekler kuyrukta gereğinden fazla bekleyebilirler. Bu problemi ortadan kaldırmak için, kuyruk belirlenen zaman aralıklarında kontrol edilerek, kuyrukta bekleme süresi belirlenen süreden fazla olanlar ikinci bir kuyruğa alınmaktadır. Zaman parametresine göre oluşturulan ikinci kuyruk ID parametresine göre oluşturulan kuyruğa göre her zaman önceliklidir. Bu şekilde daha güvenli bir sorgulama işlemi gerçekleştirilmiştir.

Tasarlanan Modbus gateway cihazının yerine getirmesi beklenen özellikler yapılan çalışma çerçevesinde gerçekleştirilmiştir. Bu özelliklere ek olarak, Ethernet portu kullanılabilir şekilde ayarlanarak, IP gateway özelliği de eklenebilir. Bu şekilde Ethernet portu olan cihazlara GSM köprüsü olarak kullanılabilen cihaz sayesinde erişim imkanı sağlanmış olacaktır. Eklenen bu özellik ile hem IP gateway hem de Modbus gateway olarak kullanılacak bir GSM köprüsü vazifesi gören cihaz tasarlanmış olacaktır. Endüstriyel uygulamalar için aşağıdaki özellikler de eklenerek daha fonksiyonel bir cihaz tasarımı ileriki çalışmalarda yapılabilir.

- Analog giriş/çıkışlar
- Yüksek sayıda giriş çıkış kontrolü eklenebilir. (CAN Bus, RF modüller veya RS485)
- PWM çıkışı
- Sayıcı girişi, Hızlı sayıcı girişi
- LCD, HMI

Donanımsal olarak bu özelliklerin eklenmesi ile daha kullanışlı hale gelecek olan cihaz için bir web ara yüzü, telnet'ten bağlanmak için bir kullanıcı ara yüz tasarımı yapılarak kullanımı daha da kolaylaştırılabilir.

KAYNAKLAR DİZİNİ

- [1] Michael E. Anderson, Selecting the Right RTOS, A Comparative Study, Embedded Systems Conference, Mart 2002
- [2] By Bob Japenga, Why Use Linux for Real-Time Embedded Systems
- [3] Daogang Peng, Hao Zhang, Jiannian Weng, Hui Li, Fei Xia, Design and Development of Modbus/RTU Master Monitoring System Based on embedded PowerPC Platform, Seoul Olympic Parktel, Seoul, Korea July 5-8, 2009
- [4] P. Dao-gang, Z. Hao, Y. Li, L. Hui, Design and Realization of Modbus Protocol Based on Embedded Linux System, CIMS Research Center, Tongji University, Shanghai, China
- [5] Yongqiang Zhang, Guozhen Zhao, Yongjian Zhang, A Smart Home Security System Based on 3G, 2009, China
- [6] M.Çayır, Unintrusively Linux kernel real-time performance measurement, Haziran 2006
- [7] A. S.Yılmaz, M. Görmemiş, Enerji Analizörleri için Geliştirilen Delphi Tabanlı Bir Enerji İzleme Yazılımı, Kahramanmaraş, Mart 2006
- [8] M. Görmemiş, Dağıtım şebekelerinde enerji kalitesi ölçümlerinde haberleşme uygulaması, Ocak 2006
- [9] Modbus Application Protocol Specification V1.1b, Modbus IDA
- [10] Modbus Messaging on TCP/IP Implementation Guide V1.0b, Modbus IDA

- [11] Sim5218 Hardware Design, Shanghai SIMCom Wireless Solutions Ltd., 2009
- [12] How to use linux driver for Sim5218
- [13] <http://www.belgeler.org/lis/archive-tlkg-lis-1g.html>
- [14] http://tr.wikipedia.org/wiki/TFTP_server
- [15] http://processors.wiki.ti.com/index.php/ELDK_usage
- [16] <http://web.deu.edu.tr/doc/yardim/socket8.htm#turler>
- [17] Brian Hall, Socket ağ protokolleri, 2002
- [18] http://tr.wikipedia.org/wiki/%C4%B0%C5%9F_Par%C3%A7%C4%B1%C4%9F%C4%B1
- [19] http://www.moxa.com/product/ioLogik_W5340-HSDPA.htm
- [20] <http://www.systech.com/pdf/Modbus.pdf>
- [21] Zhang Yu, Wu Zhisheng, He Hong, Liu Feng, Remote Monitoring Application based on Modbus and China Unicorn 3G, 2010 IEEE, China
- [22] A. Akbal, Hücresel Haberleşmenin dünü, bugünü, yarını
- [23] <http://tr.wikipedia.org/wiki/3G>

- [24] <http://intercomponents.com/upload/user/arm%20genel.pdf>
- [25] <http://www.cizgi-tagem.org/e-kutuphane/topic.aspx?id=897&p=20>
- [26] MEGEP, SCADA Programlama, Ankara 2007
- [27] 3.Nesil Mobil Telekomünikasyon sistemleri ve 3G Lisanslarının verilmesi konusunda Dünyadaki Uygulamalar ile Türkiye Analizi, Nisan 2001
- [28] <http://www.bidb.itu.edu.tr/?d=1044>
- [29] F. Justin E. Crandall, Commercial automated brewing systems
- [30] http://www.mikrodev.com/industrialcomputer_arm9.html

ÖZGEÇMİŞ

Beşir DEMİR, 1985 tarihinde Diyarbakır'ın Ergani ilçesinde doğdu. İlkokulu Kortaş Köyünde, ortaokulu Ergani ilçesinde, liseyi Diyarbakır'da okumuştur. 2006 yılında Gebze Yüksek Teknoloji Enstitüsü Elektronik Mühendisliği Bölümüne yerleşerek 2010 yılında mezun olmuştur. Eylül 2010 tarihinde Gebze Yüksek Teknoloji Enstitüsü Elektronik Mühendisliği Elektronik Anabilim Dalında yüksek lisans eğitimine başlamıştır. Şubat 2011-Haziran 2012 tarihleri arasında Mikrodev Bilişim firmasında Ar-Ge Mühendisi olarak çalışmıştır.

