İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

FPGA TABANLI MODBUS AĞ GEÇİDİ TASARIMI

YÜKSEK LİSANS TEZİ

Şirin AKKAYA

Mekatronik Mühendisliği Anabilim Dalı

Mekatronik Mühendisliği Programı

Ocak, 2015

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

FPGA TABANLI MODBUS AĞ GEÇİDİ TASARIMI

YÜKSEK LİSANS TEZİ

Şirin AKKAYA
(518121038)

Mekatronik Mühendisliği Anabilim Dalı

Mekatronik Mühendisliği Programı

Tez Danışmanı: Yrd. Doç. Dr. Ali Fuat ERGENÇ

Ocak, 2015

İTÜ, Fen Bilimleri Enstitüsü'nün 518121038 numaralı Yüksek Lisans Öğrencisi **Şirin AKKAYA,** ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"FPGA TABANLI MODBUS AĞ GEÇİDİ TASARIMI"** başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Yrd. Doç. Dr. Ali Fuat ERGENÇ** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. Müştak Erhan YALÇIN** İstanbul Teknik Üniversitesi	.....
	Yrd. Doç. Dr. Levent UCUN Yıldız Teknik Üniversitesi	.....

Teslim Tarihi : **15 Aralık 2014**
Savunma Tarihi : **27 Ocak 2015**

Sevgili Nişanlım Dr. Omır AKBATI'ya,

ÖNSÖZ

Bu tezin hazırlanması sırasında bana her konuda yardımcı olan, yardım ve katkıları ile beni yönlendiren, değerli hocam ve danışmanım Yrd. Doç. Dr. Ali Fuat ERGENÇ'e ve çalışmanın uygulama aşaması gerekli malzemelerin temini için projemizi desekleyen İstanbul Teknik Üniversitesi ÖYP Kurum Koordinatörlüğü'ne teşekkürlerimi sunarım.
Ayrıca tasarım ve uygulama konularında karşılaştığım her problemde bana yardımcı olan ve destekleyen sevgili nişanlıım Dr. Onur AKBATI'ya ve beni bu günlere getiren aileme ve özellikle beni her konuda destekleyen sevgili anneme sonsuz sevgi ve saygılarımı sunarım.

Aralık 2014

Şirin AKKAYA
Mekatronik Mühendisi

İÇİNDEKİLER

Sayfa

KISALTMALAR

PLC : Programmable Logic Controller
SCADA : Supervisory Control and Data Acquisition
RTU : Remote Terminal Unit
TCP : Transmission control Protocol
FPGA : Field Programmable Gate Array
OSI : Open System Interconnection
ECU : Electronic Control Unit
CPN : Coloured PetriNet
CAN : Control Area Network
EPA : Ethernet for Plant Automation
PC : Personal Computer
HMI : Human Machine Interface
I/O : Input/Output
PDU : Protokol Data Unit
ADU : Application Data Unit
MBAP : Modbus Application Protocol Header
CRC : Cyclic Redundancy Check
LRC : Longitudinal Redundancy Check
IP : Internet Protocol
IEEE : Instutite of Electrical and Electronic Engineers
ARP : Address Resolution Protocol
RARP : Reverse Address Resolution Protocol
SDRAM : Synchronous Dynamic Random Access Memory
JTAG : Joint Test Action Group
UART : Universal Asynchronous Receiver/Transmitter
CPLD : Complex Programmable Logic Device
VHDL : VHSIC Hardware Description Language
RTL : Register-Transfer Level
GUI : Graphical User Interface
USB : Universal Serial Bus
SPI : Serial Peripheral Interface
PLL : Phase Locked Loop
GPIO : General Purpose Input/Output
MHz : Megaherz
RAM : Random Access Memory
ROM : Read Only Memory
UDP : User Datagram Protocol
SCSn : Slave Select
SCLK : Slave Clock
MOSI : Master Output Slave Input
MISO : Master Input Slave Output
LED : Light Emitting Diode
EEPROM : Electronically Erasable Programmable Read-Only Memory

HDL **:** Hardware Description Language
RISC **:** Reduced instruction set computing
MAC **:** Medium Access Control
IPv4 **:** Internet Protocol Version 4
ICMP **:** Internet Control Message Protocol
ARP **:** Address Resolution Protocol

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

FPGA TABANLI MODBUS AĞ GEÇİDİ TASARIMI

ÖZET

Protokol lisanları aynı iki olgu arasında, belirli bir düzeyde iletişimi sağlamak amacı ile gerekli kuralları ortaya koyan bir sistemdir. Endüstriyel haberleşme protokolleri endüstriyel sistemler arasında veri alış verişini sağlayan standart dillerdir. İki cihaz arasında veri iletimi olabilmesi için cihazların aynı endüstriyel protokol ile haberleşmeleri ya da arada dönüştürücü bir cihaz kullanmaları kullanılmalıdır.

Günümüzde en çok kullanılan endüstriyel haberleşme protokolleri *Fieldbus*, *Profibus*, *Modbus*, *Canbus*, *Devicenet* ve *Controlnet* olarak sıralanabilir. Modbus protokolü 1970'li yıllarda geliştirilmiş ve zamanla yeni sürümleri oluşturulmuştur. Kolay, hızlı ve açık kaynaklı bir protokol olması günümüzde elektronik cihazlarda çok yaygın olarak kullanılmasının en büyük etkenlerindendir. Modbus protokolü ilk defa seri hat üzerinden gerçeklenmiştir. Modbus seri hat protokolü Modbus RTU ve Modbus ASCII olmak üzere iki alt protokolden oluşur. Ethernet teknolojinin gelişmesi ve yaygınlaşması ile beraber Modbus protokolünü Ethernet ağına uygun hale getirme ihtiyacı duyulmuş ve Modbus TCP protokolü geliştirilmiştir.

Bu tez çalışmasında Modbus RTU ve TCP protokolleri incelenmiş ve bu iki protokolü destekleyen bir ağ geçidi (gateway) cihaz tasarlanmış ve tasarlanan cihaz kurulan bir endüstriyel sistem üzerinde test edilmiştir. Modbus RTU protokolü OSI modelinin veri bağlantı katmanında (ikinci katman) çalışır. Bu protokol kullanıldığı zaman fiziksel katmanda veri iletimi EIA/TIA 485 (RS-485) ya da EIA/TIA 232 (RS-232) veri iletim modları kullanılabilir. Tez çalışmada fiziksel katmanda veri iletimi için EIA/TIA 485 (RS-485) standardı kullanılmıştır. OSI modelinin üçüncü, dördüncü, beşinci ve altıncı katmanları boştur ve uygulama katmanı olan yedinci katmanda Modbus uygulama protokolü çalışır. Modbus RTU protokolü bir çeşit master/slave protokolüdür ve seri ağ üzerinde bir ana cihaz (master cihaz) ve buna bağlı çalışan bir ya da birden fazla bağımlı cihaz (slave cihaz) bulunur. Ana cihaz istediği bağımlı cihaza belirlenmiş protokol formatında veri gönderir ve bağımlı cihaz veriyi alır, istenilen işlevi yerine getirdikten sonra ana cihaza cevap gönderir. Bağımlı cihazlar kendi aralarında haberleşemezler ve ağ üzerinde tek bir paket akışı olur. Verinin kontrolü protokol içersinde CRC hata kontrol yöntemi ile denetlenir. Modbus TCP protokolü Ethernet TCP/IP ağı üzerindeki cihazlar ile client/server (istemci/sunucu) haberleşmesi sağlar. İstemci cihaz haberleşmek istediği sunucu cihaza talep gönderir ve sunucu cihaz talebi alır, istenilen fonksiyonu yerine getirdikten sonra istemciye cevap gönderir. Modbus TCP protokolünde ağ üzerinde birden fazla paket alışverişi olabilir. Modbus TCP protokolü OSI modeline göre incelendiği zaman fiziksel katmanın ve veri bağlantı katmanının iç içe çalıştığı görülür ve bu iki katmana ağ ulaşım katmanı adı verilir. Ağ ulaşım katmanında (birinci ve ikinci katman) Ethernet / MAC protokolü, ağ katmanında IP protokolü, taşıma katmanında TCP protokolü ve beşinci, altıncı ve yedinci katmanında Modbus uygulama protokolünü çalıştırılır. TCP'nin görevi bütün veri paketlerinin doğru bir şekilde alınmasını sağlamaktır. IP ise paketin doğru bir şekilde adreslenmesini ve yönlendirilmesini sağlar. TCP/IP protokolü sadece bir taşıma protokolüdür ve

mesajın içeriği hakkında bir bilgi vermez. Modbus protokolü uygulama katmanında standart olan ve değişmeyen bir protokol veri birimi oluşturur. Protokol veri birimi yapılacak işlemi tanımlayan Modbus fonksiyon kodu bilgisini ve veri bilgisini taşır. Alt katmanlarda çalışan protokollere göre protokol veri biriminin başına ve sonuna ek birimler eklenerek Modbus uygulama veri birimi oluşturulur.

Tez çalışmasında ağ geçidi tasarımı için Türkçeye "Alanda Programlanabilir Kapı Dizileri" şeklinde çevrilen FPGA kullanılmıştır. FPGA, üretimden sonra istenilen uygulamaya göre donanım yapısı kullanıcı tarafından değiştirilebilen bütünleşik devre şeklinde tanımlanabilir. FPGA içerinde VHDL veya Verilog dilleri kullanılarak donanım tasarımı yapılabileceği gibi içersinde yazılımsal bir işlemci oluşturularak gömülü bir sistem de kurulabilir. Tez çalışmasında Altera firmasının DE0 Nano FPGA Geliştirme ve Eğitim kartı kullanılmıştır ve firmanın üretmiş olduğu FPGA'ler için geliştirdiği Nios II/e işlemcisi kullanılarak gömülü bir sistem oluşturulmuştur. TCP/IP yönetimini sağlaması için sisteme Wiznet firmasının ürünü olan Wiznet W5500 modülü eklenmiştir. Bu modül OSI modeline TCP/IP işlemlerini ve fiziksel katmanda veriyi fiziksel ağa gönderme işlemlerini yerine getirmektedir. Wiznet modülü kendine özgü 32 bitlik SPI protokolünü kullanarak haberleşir ve sistemde FPGA ile haberleşmesi için FPGA içersinde VHDL dili kullanılarak bir *SPI modülü* tasarlanmıştır. Ayrıca RTU ağı üzerinde bulunan cihazların FPGA ile haberleşmesi için MAX 485 entegresi kullanılarak tasarlanan RS485 dönüştürücü devresi tasarlanmıştır. Bu devre Modbus RTU protokolünün fiziksel katmanda veriyi RTU ağına iletmesi sağlamaktadır.

Altera firmasının FPGA programlamak için ücretsiz olarak kullanıcılara sunduğu *Quartus II* programlama platformu ile sistemin yazılımı hazırlanmıştır. Quartus platformu donanım tasarlamak, FPGA içersinde işlemci ve çevresel birimleri oluşturmak ve işlemcinin programlanmasını sağlamak için alt araçlar içeren kapsamlı bir yazılım paketidir. Programın sunduğu *Qsys sistem tasarlama aracı* kullanılarak NiosII/e işlemcisi ve çevresel birimleri oluşturulmuştur. Oluşturulan çevresel birimlerin birbirleri ile bağlantısı yapılmış ve adres ayarları program tarafından otomatik olarak ayarlanmıştır. İşlemcinin programlanması yine Quartus programı içersinde yer alan *Nios II Software Build Tools for Eclipse* aracı kullanılarak C dilinde yapılmıştır. Programlama sırasında Qsys aracının oluşturduğu adres bölgelerinden faydalanılır. Sistemin tasarımı ve programlanması tamamlandıktan sonra FPGA içersine kalıcı olarak gömülmüştür.

Sistem bir tane Modbus TCP istemci cihaz, altı tane Modbus TCP sunucu cihazı ve altı tane Modbus RTU bağımlı cihazı destekleyecek şekilde oluşturulmuştur. Bu sayı uygulamaya göre artırılabilir. TCP cihazlar birden altıya kadar numaralandırılmış (adreslendirilmiş) ve RTU cihazlar da yediden on ikiye kadar numaralandırılmıştır. Sistemde istemci cihaz olarak Modbus TCP ağı üzerinde çalışan bir PLC bulunmaktadır. Sunucu ve bağımlı cihazlar Modbus giriş/çıkış modülleridir. İstemci cihaz TCP ağı üzerinden Wiznet modülü ve FPGA içersinde tasarlanmış olan SPI modülü aracılığı ile Nios işlemciye talebini bildirir. İşlemci gelen talep paketini inceler ve paketi yeniden düzenler. Paket TCP cihaza gönderilecekse SPI modülü ve Wiznet üzerinden paketi TCP cihaza gönderir ve cevap da aynı yol ile önce işlemciye sonra da istemci cihaza gönderilir. Eğer paket RTU cihaza gönderilecekse işlemci paketi TCP formatından RTU formatına çevirir ve RS485 dönüştürücü devresi ile RTU cihaza talebi iletir. RTU cihazın cevabı işlemci ile TCP formatına dönüştürülür ve istemci cihaza gönderilir. Böylece TCP ve RTU ağları arasında veri alışverişi tamamlanmış olur. Sistemde bulunan cihazların konfigürasyon bilgileri DE0 Nano

kartı üzerinde bulunan EEPROM modülüne yüklenmiştir İşlemcinin EEPROM ile haberleşmesi için FPGA içerisinde tasarlanmış olan I2C modülü kullanılır.

Bu tez çalışmasında ile seri hat üzerinde haberleşen Modbus RTU cihazları ile Ethernet teknolojisi kullanarak haberleşen Modbus TCP cihazların FPGA tabalı bir ağ geçidi tasarlanarak birbirleri ile veri alış verişi yapmaları sağlanmıştır. Tasarlanan ağ geçidi cihaz protokol dönüştürme işlemi yapmaktadır. Çalışma içersinde Modbus protokolleri detaylı olarak incelenmiş, FPGA ve gömülü sistem oluşturma, Wiznet modülü, RS 485 dönüştürücü devre tasarımı ve ek birimler tanıtılmıştır. Bunlara ek olarak FPGA içersinde donanım tasarımı (SPI modülü ve I2C modülü) ve işlemcinin programlanması anlatılmış ve son bölümde tasarlanan cihazın test sonuçlarına yer veriilmiştir.

FPGA BASED MODBUS GATEWAY DESIGN

SUMMARY

Protocol is a system that outlines the rules with the aim of achieving a certain level of communication between two same cases. Industrial communication protokols are standart languages that allow data exchange among industrial devices and systems. Data transmission between two devices can obtain either using the same indutrial communication protocol or using an industrial protocol converter.

Nowadays, the most used industrial protocols can list as Fieldbus, Profibus, Modbus, Canbus, Devicenet and Controlnet. Modbus protocol was developed in 1970s and new versions were developed in years. Electronic devices are widely used Modbus protocols nowadays, because of easy, fast and open source. Modbus protocol was first implemented over serial line. There are two serial transmission mode which is commonly used are Modbus RTU and Modbus ASCII. The transmission mode and serial port parameters must be the same for all devices on a Modbus serial line. Although, the ASCII mode is required in some specific applications, but generally for interoperability between Modbus devices RTU mode is used. Modbus RTU transmit data using specific data frame, but Modbus ASCII transmit data with using ASCII characters. With the development and expansion of Ethernet technology, Modbus protocol need to adapt Ethernet network and Modbus TCP protocol has been developed.

In this thesis, Modbus RTU and Modbus TCP protocols are examined and a gateway device, which supports both Modbus protocols, is designed and tested on an industrial system that have different Modbus devices. Modbus RTU protocol works on the data link layer of the OSI model. On the physical layer, data transmission obtained using EIA/TIA 485 (RS-485) or EIA/TIA 232 (RS-232) data transmission modes. For RTU protocol, the third, fourth, fifth and sixth layer of OSI model is empty. Modbus application protocol works on application layer (seventh layer) of OSI model. Modbus serial line protocol is a kind of master/slave protocol. A master/slave type system has one master node that issues explicit commands to one-slave nodes and processes responses. Slave nodes are not typically transmit data without a request from the master node and do not communicate with other slaves. The control of the data obtains on the protocol frame using CRC error checking method. Modbus TCP protocol provides a server/client communication between devices connected on Ethernet TCP/IP network. TCP refers to the Transmission Control Protocol and IP refers to the Internet Protocol, which provides the transmission medium for Modbus TCP/IP messaging. Modbus TCP/IP uses TCP/IP and Ethernet to carry the data of the Modbus message structure between compatible devices. That is, Modbus TCP/IP combines a physical network (Ethernet), with a networking standard (TCP/IP), and a standard method of representing data. Essentially, the Modbus TCP/IP message is simply a Modbus communication encapsulated in an Ethernet TCP/IP wrapper. TCP/IP Standard Model, Ethernet handles the bottom 2 layers (1 & 2) of the seven-layer OSI stack, while TCP/IP

handles the next two layers (3 & 4). The application layer lies above TCP, IP, and Ethernet and is the layer of information that gives meaning to the transmitted data.

In this thesis, FPGA (Field Programmable Gate Array) is used to design Modbus gateway device. FPGA as the integrated circuits of which their hardware configuration can be changed by the user according to the desired functions. Using VHDL and Verilog hardware description language, hardware design can create using FPGA and embedded processor system can be design on it. In this thesis, Altera DE0 Nano FPGA Development and Education board has used for operation system and Nios II/e embedded processor, which is also Altera's software microprocessor, is embed on FPGA. Wiznet W5500 module has added on system for TCP/IP management. This module is used for fulfill TCP/IP operations of OSI models and sending data on physical layer. Wiznet module supports 32-bit SPI module to communicate FPGA, so a hardware developed on FPGA using VHDL language, named SPI module. To communicate RTU devices, a circuit is design using MAX485 integrated device that works on RTU physical layer.

Quartus II programming platform, which is free and download from Altera website is used to program FPGA device. Ouartus is a multitasking development platform, which provide both hardware design and allow embedding microprocessor and peripheral devices on FPGA. Using Qsys system development tool, Nios II/e microprocessor and peripheral devices can design and connected each other. The program also creates the address map of processor. After generating the system, a software-embedded system is created on FPGA. *Nios II Software Build Tools for Eclipse* software also reached on Quartus platform and used to program the microprocessor using C language.

In the testing industrial system, there are six Modbus TCP server devices and six Modbus RTU devices. The number of the devices can increase depending on the application. TCP devices are numbered from one to six (addressed was) and RTU devices are numbered from seven to twelve. The system includes a PLC that works on the Modbus TCP network as the client device. Servers and Modbus slave devices are Modbus input / output modules. The client device over a TCP network reaches the FPGA gateway via Wiznet and SPI module and informs the processor about which devices can send request. Processor examines the incoming request packet and redesigns it according to application. If the package send to the TCP device, it sends over the SPI module and Wiznet to TCP devices and the response is sent to the client device in the same way after the processor ago. If the package sends to the RTU device, processor translates it to the RTU format and sends it to the RTU network over the RS 485 converter circuit. RTU device receives the packet to the physical layer, performs the action and initiate the response. The response sent to RTU devices converts into the TCP format on processor and sent to the client device. Thus, exchange of data between TCP and RTU network is completed.
Communication between Modbus TCP devices using Ethernet technology, Modbus RTU devices using serial line has been provided in this study by designed a FPGA based gateway. In addition, this thesis presents detail information about Modbus protocol over serial line and Ethernet in order to used by all system designer when they need to implement Modbus protocols. Moreover, the basic information is presented about gateway design which is used not only between Modbus protocols but also can be used another protocol using on industrial system.

Hardware design and embedded processor design using FPGA is examined and presented. SPI and I2C communication protocols details, data formats, pin connections, data reading and writing examples is presented. About system components, such as PLC, switch, additional modules some sufficient information is given, and finally the results of application that perform on the testing system is added.

1. GİRİŞ

Endüstriyel sistemler olarak adlandırılan otomasyon, mekatronik ve kontrol sistemleri sensor, aktüatör, valf, röle ve sürücü devreler gibi birbirlerinden yapı olarak bağımsız birçok alt sistemden oluşur. Bu sistemlerin uyum içersinde çalışabilmesi için birbirleri ile haberleşebilmesi ve kontrol edilebilir olması gerekmektedir. 1970'li yıllardan sonra endüstriyel sistemlerde yaşanan büyük gelişmeler sonucunda karmaşık sistemlerin oluşturulması endüstriyel haberleşme alanında bir takım gelişmelerin yaşanmasına neden olmuştur. Endüstriyel sistemlerin kontrolü ve takibi için PLC'ler ve SCADA sistemleri geliştirilmiştir. Bu sistemlerin alt sistemlerle haberleşebilmesi için endüstriyel protokoller oluşturulmaya başlanmış ve bu protokoller standart haline getirilmiştir. Günümüzde en çok kullanılan endüstriyel protokoller *Fieldbus*, *Profibus*, *Modbus*, *Canbus*, *DeviceNet* ve *ControlNet* olarak sıralanabilir. Endüstriyel cihaz üreticileri genellikle ürünlerini bu protokolleri destekleyecek şekilde üretmektedirler. Modbus protokolü 1979 yılında *Modicon* firması tarafından PLC'lerde kullanılmak üzere bir seri haberleşme protokolü olarak geliştirilmiştir. Kolay, hızlı ve açık kaynaklı bir protokol olması, günümüzde elektronik cihazlarda çok yaygın olarak kullanılmasının en büyük etkenlerindendir. Modbus ilk olarak seri hat üzerinden haberleşen bir protokol olarak tasarlanmış ve zamanla gelişen teknoloji ile farklı versiyonları geliştirilmiştir. Modbus RTU seri hat üzerinde en yaygın olarak kullanılan Modbus protokolüdür. Fakat Ethernet teknolojisinin gelişmesi ile protokol TCP ağına adapte edilerek ve Modbus TCP protokolü oluşturulmuştur. Günümüzde Ethernet ağı altyapısının yaygınlaşması Modbus TCP protokolünün kullanımı artmasını sağlamıştır. Buna bağlı olarak seri hat üzerinden haberleşen cihazlarla Ethernet üzerinden haberleşen cihazlar arasındaki veri alışverişini gerçekleştirmek için endüstriyel protokol dönüştürücü cihazlarına ihtiyaç duyulmuştur. Ağ geçidi (Gateway) olarak isimlendirilen bu cihazlar bir ağ üzerinden başka bir ağa veri aktarmaya yarayan noktalardır. Modbus Ağ geçidi cihazları ise farklı Modbus protokolleri arasında (bu

çalışmada Modbus RTU ve Modbus TCP arasında) veri alışverişini sağlayan birimlerdir.

1.1 Tezin Amacı

Bu tez çalışmasında, Ethernet altyapısını kullanan ve kullanımı giderek yaygınlaşan Modbus TCP ağı ile seri hat üzerinden haberleşen Modbus RTU ağı arasında veri alışverişi sağlayan FPGA tabanlı Modbus Ağ geçidi cihazı tasarlanmış ve tasarlanan cihaz, üzerinde farklı Modbus modüllerinin bulunduğu endüstriyel bir test düzeneği kurularak test edilmiştir. Bu çalışmanın temel amacı, Modbus RTU ile haberleşen cihazların yeni Ethernet tabanlı endüstriyel ağlarda kullanılmasına olanak sağlayan bir ara birim tasarlamaktır. Tasarım için günümüzde kullanımı giderek yaygınlaşan FPGA tercih edilmiştir. "Alan Programlanabilir Kapı Dizileri" şeklinde dilimize çevrilen FPGA, üretimden sonra donanım yapısı kullanıcı tarafından uygulamaya göre değiştirilebilen entegre devre şeklinde tanımlanabilir. Tasarım sırasında FPGA'in hem donanım tasarlama özelliğinden hem de gömülü devre oluşturma özelliğinden faydalanılmıştır. Oluşturulan test sisteminde istemci cihaz, Modbus TCP ağı üzerinde bulunan ve istemci görevini üstlenen PLC'dir. Sistemin çalışma şekli şöyle özetlenebilir. İstemci cihaz, veri alışverişi yapabilmek için FPGA tabanlı Modbus ağ geçidi birimine talep gönderir ve gönderilen talep tasarlanan ağ geçidi cihazında yeniden düzenleyerek istenilen Modbus TCP ya da RTU ağındaki cihaza ulaştırır. Cihazlar talebi aldıktan sonra ilgili işlemleri gerçekleştirerek bir cevap oluşturur. Cevap yeniden tasarlanan ağ geçidine gelir ve buradan istemci cihaza yönlendirilir. Böylece veri alışverişi tamamlanmış olur.

1.2 Literatür Araştırması

Modbus protokolü, bağımsız bir kuruluş olan *'Modbus Organization'* tarafından geliştirilen bir protokoldür. Organizasyon, Modbus protokolünün endüstriyel sistemlerde kullanımını artırmak için çalışmakta ve aynı zamanda protokol ile ilgili bilgi paylaşımı yapmaktadır. Bu paylaşımlar içersinde, temel protokol yapısı, fonksiyon kodları ve akış diyagramları ve hata kodları ile ilgili ayrıntılı bilgiler "*Modbus Application Protocol Specification V1.1b3."* içersinde anlatılır [1]. Modbus protokolünün seri hat üzerinden nasıl gerçekleştirileceği, Modbus RTU protokol yapısı, veri iletimi, ana cihaz, bağımlı cihaz blok şemaları ve hata kontrolünün nasıl

yapıldığı "*Modbus over Serial Line Specificaiton and Implementation guide V1.02"* içersinde anlatılmıştır [2]. Modbus protokolünün Ethernet hattı üzerinde nasıl gerçeklendiği, TCP bağlantı yönetimi ve OSI modelinde Modbus TCP protokolü katmanları "*Modbus Mesaging on TCP/IP Implementation Guide V1.0a"* içersinde anlatılmıştır [3]. Ayrıca Modbus TCP protokolü ve protokol katmanları ile ilgili detaylı bilgi için Prosoft-Technology firmasının üretmiş olduğu AcroMag modülü için hazırlamış olduğu teknik dokümandan yararlanılmıştır [4].

Konu ile ilgili akademik çalışmalar incelenmiş ve bazıları bu bölümde kaynak olarak verilmiştir. Endüstriyel otomasyon sistemlerinde gömülü veri alışverişinde ekranlama sistemlerinin taleplerini karşılamak için Linux altında Modbus seri hat haberleşme protokolünü temel alan bir gömülü ekranlama platformu [5] içersinde tasarlanmıştır. Modbus protokolünün güvenliği analizi üzerine çalışmışlar, protokol üzerine saldırılar düzenleyip sonuçlara çalışmalarında [6]'da yer vermişlerdir. Yine Modbus protokolü kullanan sistemlerin güvenlikleri üzerine çalışmışlar ve kötü niyetli yazılımların ağa saldırmasını engelleyen, aynı zamanda Modbus protokolünü geliştirmeyi hedefleyen bir şifreleme düzeni [7] içersinde sunmuşlardır. Başka bir çalışmada, hibrit elektrikli araç bilgilendirme sistemi için Modbus RTU protokolü kullanılmıştır. Çalışmada elektronik kontrol birimi (ECU) ve uzak kontrol birimi (RTU) arasında haberleşme Modbus RTU protokol ile 38400kb/s veri hızında gerçeklenmiştir [8]. Modbus protokolünü ve QNX operasyon sistemini temel alan, seri arabirim cihaz sürücüleri için genel bir tasarım yapısı [9]'da sunmuşlardır. QNX-OS adres-uzay mekanizmasını koruyan ve mikro-kernel oluşturan birçoklu görevli gerçek zamanlı operasyon sistemidir. Seri port haberleşmesi QNX altında C programı kullanılarak gerçekleştirilmiş ve ilgili testler SIEMENS S7-200 PLC kullanılarak yapılmıştır. Sonuç, bu metodun gerçek zamanlı, kararlı ve güvenilir olduğunu göstermiştir. Modbus seri hat haberleşme protokolünün CPN modeli [10] içerisinde sunmuştur. CPN (Coloured Petri Nets), genellikle haberleşme protokolleri ve paralel prosesler için kullanılan bir modelleme dilidir. Bir başka çalışmada, Modbus protokolü kullanılan endüstriyel ağlarda, ağ noktalarında meydana gelen bağlantı problemleri üzerine bir çalışma yapılmış ve çözümler sunmuşlardır [11]. Network haberleşmesi ve non-linear kontrolcü tasarımı gerçekleştirdikleri çalışmalarında endüstriyel ağlarda Ethernet trafiğini izleyerek ve Modbus haberleşmesini analizi [12] içersinde gerçekleştirmişlerdir. Modbus TCP ağları için

tasarlanan model tabanlı saldırı tespit sistem, (model based intrusion detection system) [13]'de tasarlamıştır. Modbus ve EPA (Ethernet for Plant Automation) için ağ geçidinin nasıl tasarlanıp geliştirileceğini anlatan çalışmada ARM tabanlı gömülü sistem ve µC/OS gerçek zamanlı işletim sistemi kullanmışlardır. Ağ geçidi EPA protokolü ve Modbus protokolü arasında çift yönlü veri akışını gerçekleştirmiş ve güç santralleri süreç kontrolü için güvenli, kararlı ve gerçek zamanlı haberleşme sağlamıştır [14].

Modbus konulu ülkemizde yapılan tez çalışmaları incelendiği zaman, uzaktan sayaç okuma tekniklerinde kullanılan Modbus TCP protokolü ve IEC 61107 Mod C protokollerini incelemiş ve Visual Basic 6.0 yazılımı kullanılarak her iki protokolde çalışan bir test yazılımı geliştirilerek ve tasarlanan sistem analiz edilmiştir [15]. Başka bir çalışmada, SCADA sistemlerinde RTU cihazların uzaktan izlenmesi ve kontrol edilebilmesi için kullanılan bir Modbus ağ geçidi tasarlamış ve tasarlanan sistemde haberleşme protokolü olarak Modbus RTU, kablosuz yüksek hızlarda veri haberleşmesi için de 3G teknolojisi kullanılmıştır. Çalışmada ARM 9 tabanlı EP9102 işlemcisi kullanılmıştır. Modbus ağ geçidi cihazı aynı anda birden fazla istemciye cevap verecek şekilde tasarlanmıştır [16]. Endüstriyel otomasyonda bileşenlerin entegrasyonu ve birbirleri ile uyumlu çalışan bilmesi için, PC ve dokunmatik panelli Modbus haberleşme protokolü üzerinden C++ ve Visual Basic yazılım dilleri kullanılarak hazırlanan program ile verileri uzaktan izleme ve kontrolü yapılmıştır [17]. Modbus uygulamaları için kablosuz bir modül [18]'de tasarlanmıştır. Çalışma RF modülünün tasarımı, haberleşme performansı ve maliyeti hakkında detaylı bilgi vermiştir. Sonuçlar önerilen senaryolarda uygulanmış ve sistem başarılı olmuştur. Başka bir çalışmada, güç analizörlerinden elde edilen temel elektriksel büyüklüklerin PC'ye aktarılması sağlanmıştır. Aktarılan bilgiler bilgisayar ekranında gerçek zamanlı olarak görüntülenmiş ve istenilen zaman aralıklarında kayıt altına alınmıştır. Çalışmada Borland Delphi 7.0 programı kullanılarak yazılım geliştirilmiş ve haberleşme protokolü olarak Modbus RTU kullanılmıştır [19].

Aşağıda Modbus protokolü ila ilgili yurt dışında yapılan tez çalışmaları incelenmiştir. ABB güç üretim sistemleri için tasarlanan sistemde PLC'den harici bir sisteme zaman bilgisi gönderilmesi amaçlamıştır. Milisaniyelik belirli olaylar için zaman bilgisi oluşturma ve depolamak için bir sistem tasarlanmıştır. Zamanı doğru bir şekilde temsil edebilmek için 48 bitlik çözünürlük kullanılmıştır. Modbus TCP

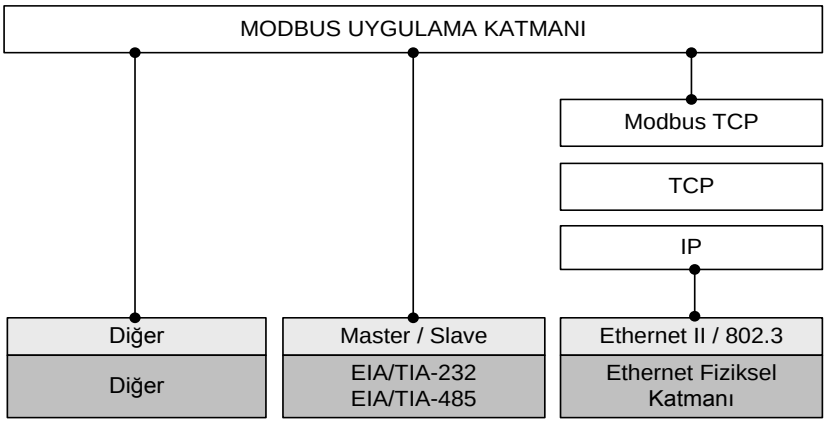
üzerinden gerçekleştirilen haberleşmede zaman verileri bitler şeklinde ya da word olarak saklanmıştır [20]. Endüstriyel ethernetin incelendiği çalışmada Common Industrial Protokol tabanlı Ethernet-IP ve Modbus TCP/IP protokolleri karşılaştırmıştır [21]. [22]'de Modbus TCP protokolü incelenmiş ve protokolün n-3.13 simülatör ağına entegrasyonu gerçekleştirmiştir.

1.3 Hipotez

Endüstriyel sistemlerin birbirleri ile haberleşebilmesi için standart haline getirilmiş ve endüstriyel haberleşme protokolleri olarak adlandırılan protokoller kullanılır. Teknolojinin gelişmesi ile birlikte bu protokollerin yeni versiyonları oluşturulur ya da yeni protokoller hazırlanır. Protokoller sayesinde endüstriyel cihazlar tek bir dil kullanarak haberleşmiş olurlar. Fakat yeni protokoller oluşturulduğu zaman eski protokolleri destekleyen cihazların bu protokollerle uyumlu çalışması gerekir. Bu durumda protokol dönüşümünü gerçekleştirecek cihazlara ihtiyaç duyulur. Tez çalışmasında günümüzde endüstriyel sistemlerde an çok kullanılan protokollerden biri olan Modbus protokolü incelenmiştir. Çalışmada Modbus protokolünü seri hat üzerinden gerçekleyen Modbus RTU protokolü ile Ethernet üzerinden gerçekleyen Modbus TCP protokolünü birbirine bağlayan gömülü sistem içersinde bir ağ geçidi tasarlanmıştır. Gömülü sistem için Altera FPGA kullanılmış ve FPGA ile hem donanım tasarımı yapılmış hem de içersine aynı firmanın işlemcisi olan Nios II işlemcisi gömülmüştür. Çalışma, Modbus protokolünün gömülü sisteme entegrasyonunu gerçekleştirmiştir. Bunu için bazı ek birimlere ihtiyaç duyulmuş ve protokolün TCP yönetimi için hazır bir modül kullanılmıştır. Sistemi test etmek için oluşturulan test düzeneğinde farklı firmaların ürettiği Modbus protokolünü destekleyen cihazlar kullanılmış ve sistem test edilmiştir.

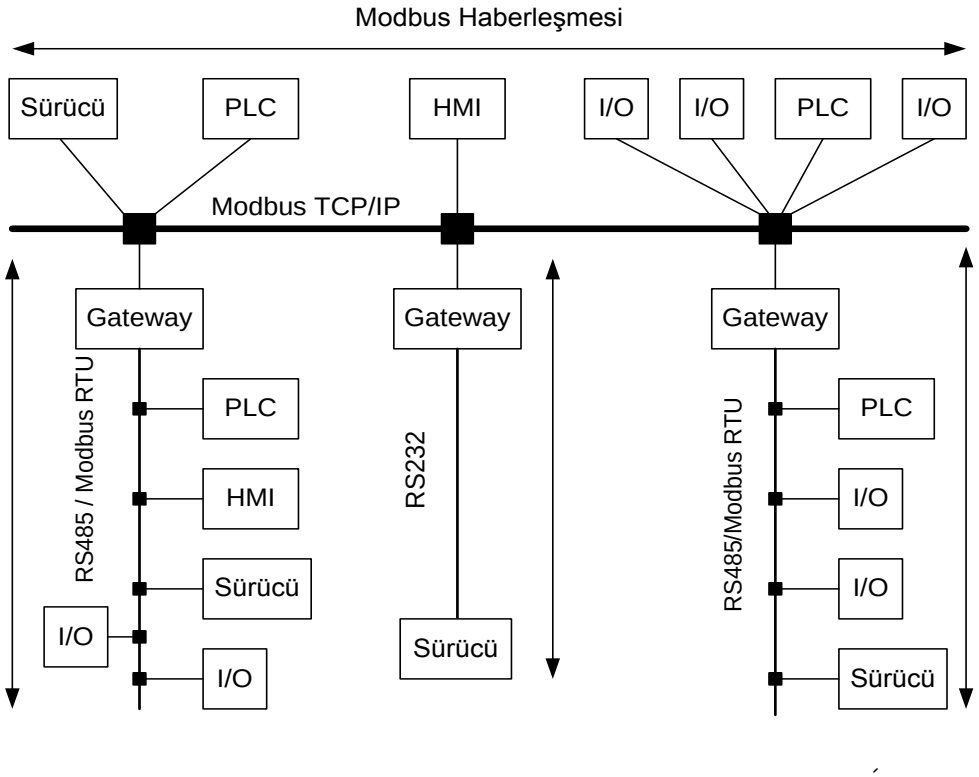
2. MODBUS HABERLEŞME PROTOKOLÜ

Modbus, 1979 yılında *Modicon* firması tarafından programlanabilir lojik kontrolörler (PLC) için geliştirilmiş bir haberleşme protokolüdür. Açık kaynak olarak yayınlanması, basit altyapısı ve kolay uygulanabilirliği sayesinde birçok imalatçı firma tarafından desteklenmekte ve çoğu endüstriyel haberleşme sistemlerinde tercih edilmektedir. Bir ana cihaz ve bu cihaza bağlı olarak çalışan bir veya birden fazla bağımlı cihaz arasında aynı ağ üzerinde veri alışverişini gerçekleştirir. Endüstriyel otomasyon cihazlarının birçoğu bu haberleşme protokolünü kullanmaktadır. Endüstriyel teknolojinin gelişmesi ile Modbus protokolü de birçok gelişim sürecinden geçmiş ve çeşitli alt protokollere ayrılmıştır. Günümüzden en çok kullanılan Modbus protokolleri seri hat üzerinden gerçeklenen Modbus RTU ve Ethernet üzerinden gerçeklenen Modbus RTU protokolleridir. Her iki protokolün ortak noktası alt katmanlardan bağımsız olarak uygulama katmanında çalışan protokol veri birimidir (Protokol Data Unit, PDU). Modbus protokolü, altındaki haberleşme katmanlarından bağımsız olarak basit bir protokol veri birimi (protokol data unit, PDU) tanımlar. Protokol, kulandığı alt protokollere göre uygulama veri birimi (application data unit, ADU) içersinde bir takım ek alanlar oluşturabilir. Bu ek alanlar Modbsu RTU ve TCP protokollerinde farklılık gösterir. Şekil 2.1'de genel olarak Modbus uygulama katmanının seri hat üzerinden gerçeklenen ve Ethernet üzerinden gerçeklenen protokollere nasıl bağlandığı gösterilmiştir [1].



Şekil 2.1 : Modbus haberleşme yığını.

Modbus haberleşme protokolünü destekleyen her cihaz (PLC, HMI, Control Paneli, Sürücü, I/O Cihazları...) Modbus protokolünü kullanabilir özelliktedir. Seri hat üzerindeki cihazlar Modbus RTU veya ASCII haberleşmesini kullanırken Ethernet hattı üzerindeki cihazlar Modbus TCP protokolünü kullanırlar. Bu iki protokol arasındaki dönüşüm işlemi ise Modbus Ağ geçidi (gateway) cihazları ile sağlanır. Şekil 2.2'de örnek bir Modbus ağ mimasisi gösterilmiştir [2].



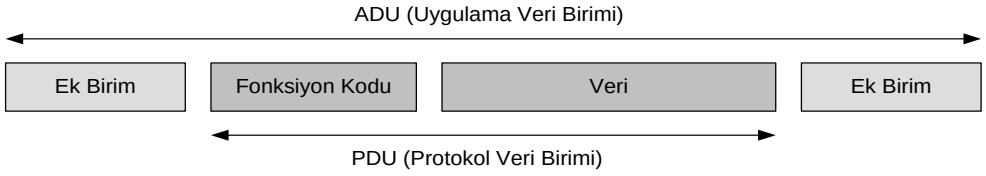
Şekil 2.2 : Modbus ağ mimarisi.

Tez çalışmasının bu bölümünde tasarlanan Modbus ağ geçidi cihazı için kullanılan Modbus RTU ve Modbus TCP protokolleri detaylı olarak anlatılmıştır. Bölüm 2.1'de Modbus Uygulama Katmanı içersinde genel Modbus veri yapısı, protokol veri birimi incelenmiş ve çalışmada kullanılan Modbus fonksiyon kodları ve hata kodları akış diyagramları verilerek anlatılmıştır. Bölüm 2.2'de Modbus RTU Protokolünün OSI modeline göre nasıl gerçeklendiği ve Modbus verisinin hata denetiminin nasıl yapıldığı incelenmiş ve Bölüm 2.3'de Modbus TCP protokolünün OSI modeline göre nasıl gerçeklendiği, her katmanda meydana gelen değişiklikler dikkate alınarak anlatılmıştır.

2.1 Modbus Uygulama Katmanı

Modbus uygulama katmanı haberleşme protokolü, OSI (Open System Interconnection) modelinin yedinci katmanında bulunan, farkı türlerdeki veri yolları ve ağlarda çalışan cihazlar arasında istemci/sunucu (client/server) haberleşmesi sağlayan bir protokolüdür (Application Layer Protokol). Servis edilen belirli fonksiyon kodlarını kullanarak bir talep/cevap (request/reply) haberleşmesi sağlar.

Modbus protokolü uygulama katmanında basit bir protokol veri birimi (PDU) tanımlar. Bu bölüm içersinde kullanılacak fonksiyon kodu ve veri bilgisi bulunur. Alt katmanlarda kullanılan protokollere göre protokol veri birimine ek alanlar eklenerek uygulama veri birimi (ADU) oluşturulur. En genel şekilde Modbus veri yapısı Şekil 2.3'de gösterilmiştir.



Şekil 2.3 : Modbus veri yapısı.

Bu bölümde protokol veri birimi içersinde yer alan ve çalışma kapsamında kullanılan fonksiyon kodları ve hata kodları incelenmiş ve fonksiyon kodları için akış diyagramları, talep, cevap ve hata durumları anlatılmıştır.

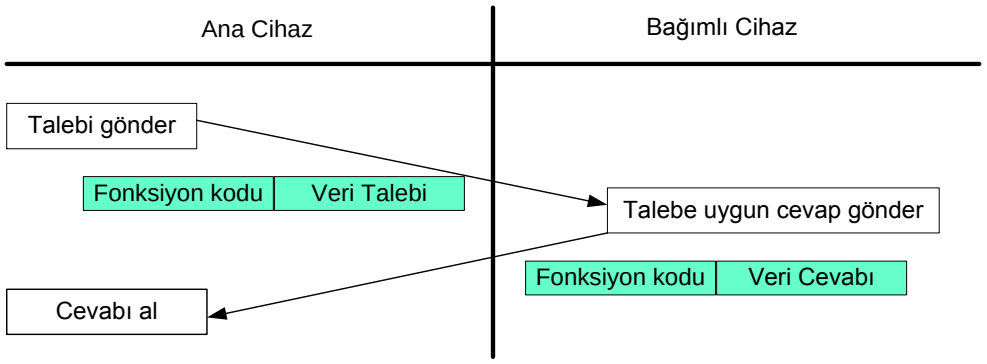
Modbus uygulama protokolü ana cihaz tarafından başlatılan bir talep formatı kurar ve uygulama veri birimi ile hangi bağımlı cihaza ne tür bir işlem yapacağını bildirir.

Modbus veri biriminin fonksiyon kod alanı bir byte ile kodlanır ve geçerli değer aralığı 1 - 255 arasındaki ondalık sayılardır. Fonksiyon kod olarak 0 geçerli bir değer değildir. Bu fonksiyon kodları bağımlı cihazların ne tür bir işlem yapacaklarını ya da yaptıklarını tanımlar. Çoklu eylemleri tanımlamak için bazı fonksiyon kodlarına alt fonksiyon kodları da eklenebilir.

Ana cihaz tarafından gönderilen mesajın veri alanı, ana cihazın ya da bağımlı cihazın talep ya da cevap parametrelerini içerir. Bu bilgiler, dijital giriş çıkışların ya da kaydedicilerin (analog giriş/çıkışların) adresleri, yapılacak işlem miktarı ya da veri alanındaki gerçek veri değeri olabilir. Veri alanı, ana cihazın ek bilgiye ihtiyaç

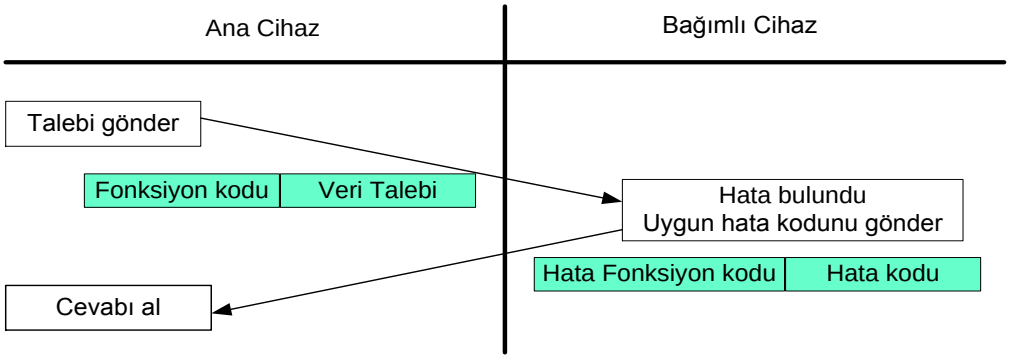
duymadığı durumlarda bulunmayabilir. Bu durum gönderilen fonksiyon koduna bağlıdır.

Ana cihaz tarafından talep edilen uygulama veri biriminde bir hata yoksa bağımlı cihaz istenilen işlemi gerçekleştirir ve ana cihazın talebine cevap verir. Fakat bir hata oluşursa bağımlı cihaz ana cihaza bir hata mesajı göndererek işlemlerin devam etmesini sağlar. Normal cevap durumunda ara cihaz talepte bulunan orjiinal fonksiyon kodunu kullanır. Şekil 2.4'de ana cihazın talebine verilen hatasız bir cevap gösterilmiştir.



Şekil 2.4 : Hatasız durumda Modbus veri iletimi.

Gönderilen mesajda hata varsa bağımlı cihaz hatalı fonksiyon kodunu ve hata kdunu mesaja ekleyerek Şekil 2.5'teki gibi cevap gönderir.



Şekil 2.5 : Hatalı durumda Modbus veri iletimi.

Modbus protokol veri biriminin boyutu, protokolün ilk kez seri hat üzerinden geliştirilmesinden miras kalan boyut sabitleri tarafından sınırlandırılır. Sonuç olarak;

RS232/RS485 ADU = 253 bytes + Adres (1 byte) + CRC (2 bttes) = 256 bytes
Modbus TCP ADU = 253 bytes + MBAP (7 bytes) = 260 bytes

Modbus protokolü üç ayrı protokol veri birimi tanımlar. Bunlar; Modbus talep protokol veri birimi, Modbus cevap protokol veri birimi ve Modbus hata cevabı protokol veri birimi şeklinde isimlendirilebilir. Protokol adresleri ve data bilgilerini temsil etmek için "big Endian" kullanır. Yani bir byte'tan büyük sayısal nicelik iletilirken en önemli bit (most significant bit) ilk önce gönderilir.

Modbus veri modeli, bir dizi ayırt edici özelliklere sahip tablosal bir yapıdan meydana gelir ve veriler bu dört tablo içersinde saklanır. İki tablo dijital çıkış (coil, bobin) değerlerini ve dijital giriş (kontak) değerlerini on/off (açık/kapalı) formatında tek bit olarak saklar. Diğer iki tablo da saklayıcı kaydedicileri (analog çıkışlar, holding register) ve giriş kaydedicileri (analog girişler, input register) değerlerini nümerik formatta word (16 bit) olarak saklar. Her tablo 9999 değer saklayabilir. Bu tablosal yapı Çizelge 2.1'de gösterilmiştir;

Çizelge 2.1 : Modbus veri tablosu.

Coil/contact/ register numarası	Veri Adresi	Özellik	Tablo Adı
1-9999	0000-270E	Okuma/Yazma	Dijital Çıkışlar (Coil)
10001-19999	0000-270E	Okuma	Dijital Girişler (Contact)
30001-39999	0000-270E	Okuma	Giriş Kaydedicileri
40001-49999	0000-270E	Okuma/Yazma	Saklayıcı Kaydediciler

2.1.1 Modbus fonksiyon kodları

Fonksiyon kodları bağımlı cihazların yapacakları ya da yaptıklarını işlemleri ifade eder. Tez kapsamında kullanılan fonksiyon kodları Çizelge 2.2'de listelenmiştir;

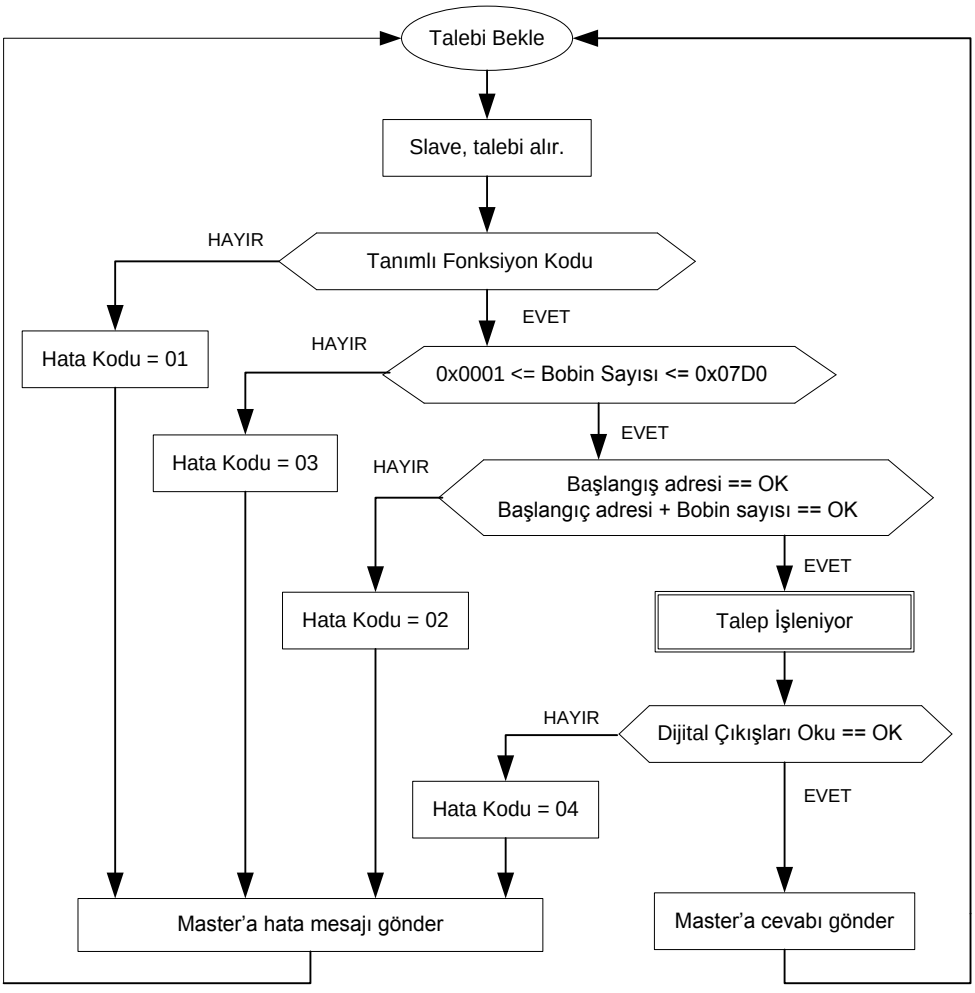
Çizelge 2.2 : Kullanılan fonksiyon kodları.

Fonksiyon Kodu	Görevi	Açıklama
01 (01 hex)	Okuma	Dijital çıkışları (bobin, coil) okuma
02 (02 hex)	Okuma	Dijital girişleri okuma
03 (03 hex)	Okuma	Saklayıcı kaydedicileri okuma
04 (04 hex)	Okuma	Giriş kaydedicilerini okuma
05 (05 hex)	Tekli Yazma	Tek bir bobine yazma
06 (06 hex)	Tekli Yazma	Tek bir tutucu kaydediciye yazma
15 (0F hex)	Çoklu Yazma	Birden fazla bobine yazma
16 (10 hex)	Çoklu Yazma	Birden fazla tutucu kaydediciye yazma

Aşağıda tez kapsamında kullanılan modbus fonksiyon kodları açıklanmış, talep, cevap, hata durumları ve akış diyagramlarına yer verilmiştir.

- **Fonksiyon Kod 1 (0x01 hex): Dijital Çıkışları Okuma**

Fonksiyon Kod 1, verilen adresteki cihazın dijital çıkışlarının (coil, bobin) durumlarını okumak için kullanılır. Tek bir dijital çıkışın durumu okuyabileceği gibi bitişik olmak şartıyla birden fazla çıkış durumu okunabilir. Talep eden protokol veri birimi, okunacak ilk bobinin adresini başlangıç adresini belirler ve devamında okunacak olan çıkış sayısı verilir. Cevap veren mesajın protokol veri biriminde fonksiyon kodu belirtildikten sonra okunan çıkış sayısı (byte sayısı) ve devamında okunan çkışların durumları bildiriilir. Lojik 1 açık (on) ve Lojik 0 kapalı (off) durumuna karşılık gelir. Bu fonksiyon kodu için akış diyagramı aşağıdaki gibidir;



Şekil 2.6 : Fonksiyon Kod 1 için akış diyagramı.

Bu fonksiyon kodu için talep, cevap ya da hata durumu için protokol veri birimi içerisinde bulunan bilgiler Çizelge 2.3'te gösterildiği gibi olur.

Çizelge 2.3 : Fonksiyon Kod 1 için talep, cevap ve hata formatı.

Modbus Talep		
Fonksiyon kodu	**Başlangıç adresi**	**Okunacak çıkış sayısı**
0x01	0x0000 – 0xFFFF	1-2000 (0x7D0)
(1 Byte)	(2 Byte)	(2 Byte)

Modbus Cevap		
Fonksiyon kodu	**Byte sayısı**	**Çıkış durumları**
0x01	N (0-255)	0 ya da 1
(1 Byte)	(1 Byte)	(n=N yada N+1 byte)

Modbus Hata	
Fonksiyon kodu	**Hata kodu**
Fonksiyon Kod + 0x80	01-02-03-04
1 Byte	1 Byte

Çizelge 2.4 ve 2.5'da, Fonsiyon kod 1 kullanılarak oluşturulan talep ve cevap mesajları için bir örnek verilmiştir;

17 adresli cihazın 20'den 56'ya kadar olan dijital çıkışlarının durumunu okumak için cihaza aşağıdaki gibi talep gönderilir.

Çizelge 2.4 : Fonksiyon Kod 1 için Modbus talep örneği.

Modbus Talep : 11 01 0013 0025 0E84	
11	Cihaz adresi (17 = 11 Hex)
01	Fonksiyon kodu (Dijital Çıkışları Oku)
0013	Okunacak ilk çıkışın adresi (Çıkış 20-1=19=13 Hex)
0025	Okunacak çıkış miktarı (Çıkış 20'den 56'ya =37=25 hex)
0E84	CRC hata kontrolü

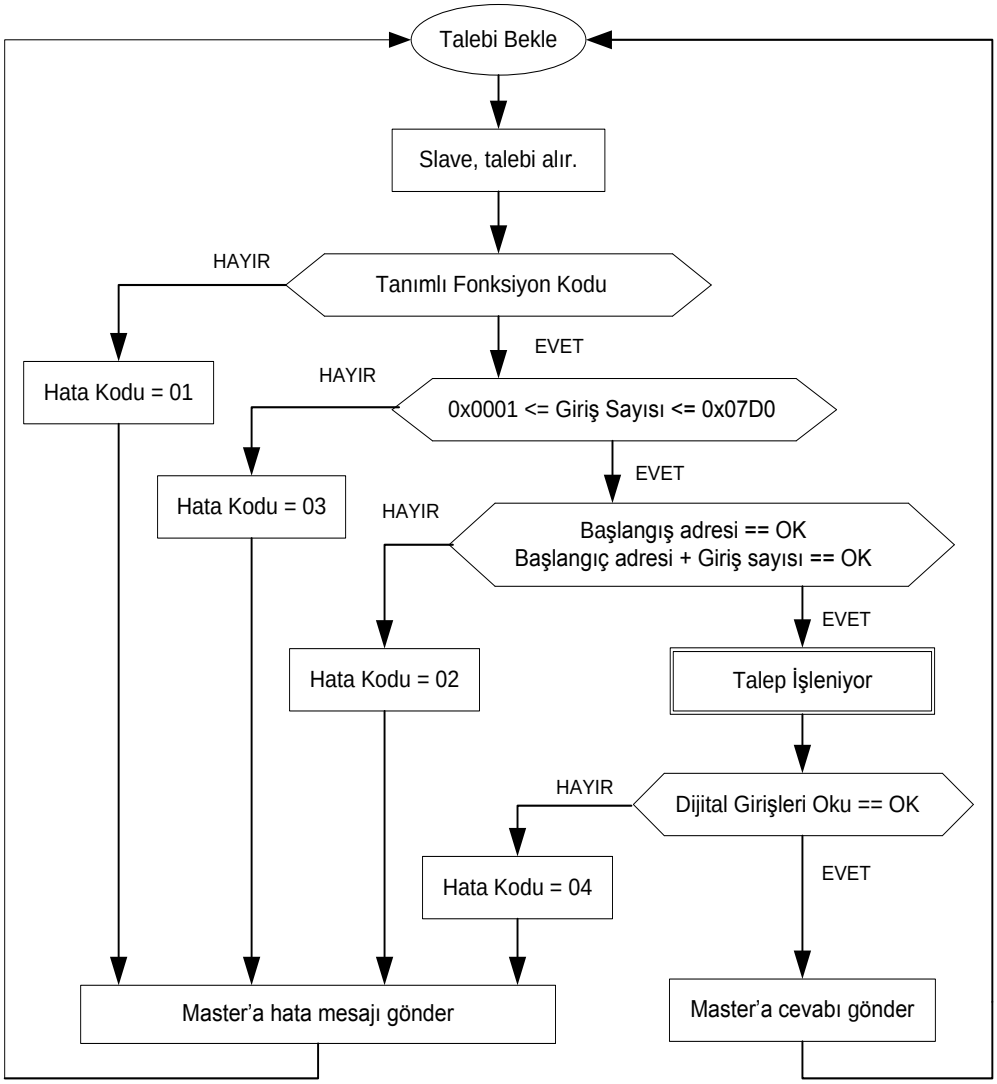
17 adresli cihazın bu talebe cevabı aşağıdaki gibi olabilir.

Çizelge 2.5 : Fonksiyon Kod 1 için Modbus cevap örneği.

Modbus Cevap : 11 01 05 CD6BB20E1B 45E6	
11	Cihaz adresi (17 = 11 Hex)
01	Fonksiyon kodu (Dijital Çıkışları Oku)
05	Okunan çıkışların byte sayısı (37 çıkış / 8bit = 5 byte)
CD	Çıkış 27-20 (1100 1101)
6B	Çıkış 35-28 (0110 1011)
B2	Çıkış 43-36 (1011 0010)
0E	Çıkış 51-44 (0000 1110)
1B	000 & Çıkış 56-52 (0001 1011)
45E6	CRC hata kontrolü

- **Fonksiyon Kod 2 (0x02 hex): Dijital Girişleri Okuma**

Fonksiyon Kod 2, verilen adresteki cihazın dijital girişlerinin durumlarını okumak için kullanılır. Tek bir girişin durumu okuyabileceği gibi bitişik olmak şartıyla birden fazla girişin durumu da okunabilir. Talep eden protokol veri birimi, okunacak ilk girişin adresini başlangıç adresini olarak belirler ve devamında okunacak olan giriş sayısı verilir. Cevap veren mesajın protokol veri biriminde aynı şekilde fonksiyon kodu belirtildikten sonra okunan giriş sayısı (byte sayısı) ve devamında okunan giriş durumları bulunur. Lojik 1 açık (on) ve Lojik 0 kapalı (off) durumuna karşılık gelir. Bu fonksiyon kodu için akış diyagramı Şekil 2.7'de gösterilmiştir



Şekil 2.7 : Fonksiyon Kod 2 için akış diyagramı.

Bu fonksiyon kodu için talep, cevap ya da hata durumu için protokol veri birimi içerisinde bulunan bilgiler Çizelge 2.6'te gösterildiği gibi olur.

Çizelge 2.6 : Fonksiyon Kod 2 için talep, cevap ve hata formatı.

Modbus Talep		
Fonksiyon kodu	**Başlangıç adresi**	**Okunacak bobin sayısı**
0x02	0x0000 – 0xFFFF	1-2000 (0x7D0)
1 Byte	2 Byte	2 Byte

Modbus Cevap		
Fonksiyon kodu	**Byte sayısı**	**Bobin durumları**
0x02	N	
1 Byte	1 Byte	N Byte

Modbus Hata	
Fonksiyon kodu	**Hata kodu**
Fonksiyon Kod + 0x80	01-02-03-04
1 Byte	1 Byte

Çizelge 2.7 ve 2.8'da, Fonsiyon kod 2 kullanılarak oluşturulan talep ve cevap mesajları için bir örnek verilmiştir;

17 adresli cihazın 10197'den 10218'e kadar olan dijital girişlerinin durumunu okumak için cihaza aşağıdaki gibi talep gönderilir.

Çizelge 2.7 : Fonksiyon Kod 2 için Modbus talep örneği.

Modbus Talep : 11 02 00C4 0016 BAA9	
11	Cihaz adresi (17 = 11 Hex)
02	Fonksiyon kodu (Dijital Girişleri Oku)
00C4	Okunacak ilk girişin adresi (10197-10001=196=C4 Hex)
0016	Okunacak giriş miktarı (197'den 218'e =22=16 hex)
BAA9	CRC hata kontrolü

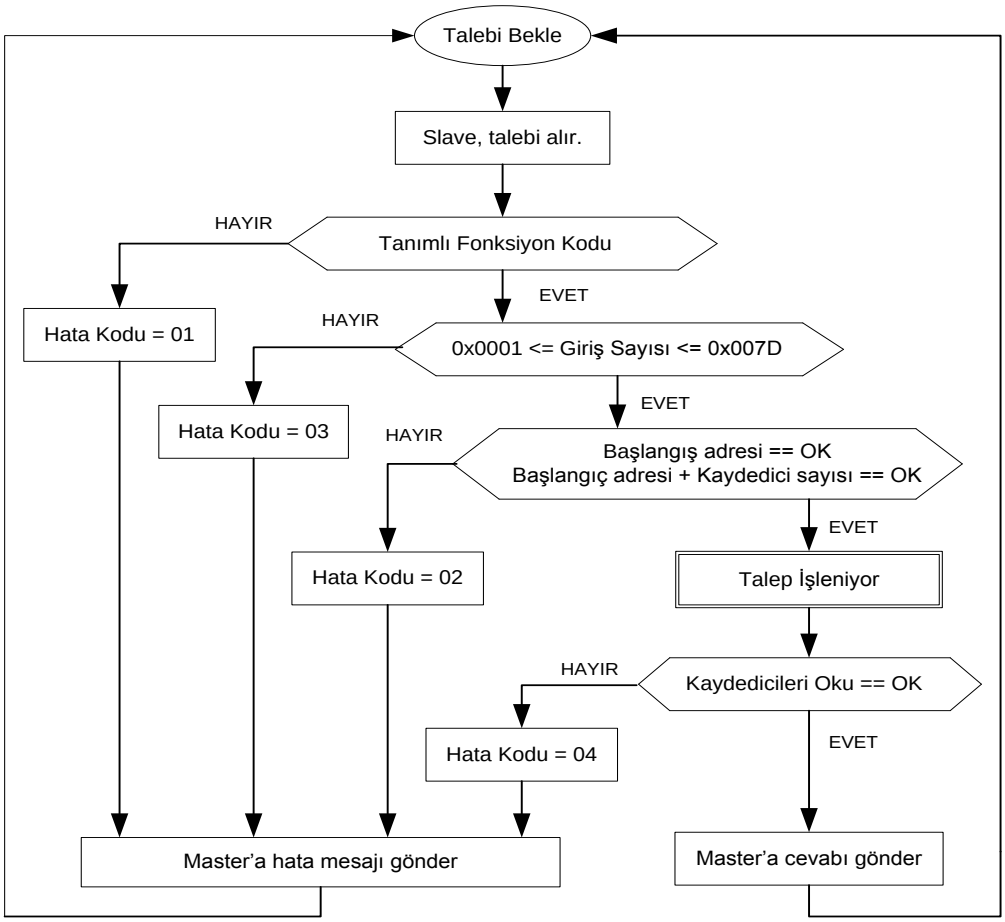
17 adresli cihazın bu talebe cevabı aşağıdaki gibi olabilir.

Çizelge 2.8 : Fonksiyon Kod 1 için Modbus cevap örneği.

Modbus Cevap : 11 02 03 ACDB35 2018	
11	Cihaz adresi (17 = 11 Hex)
02	Fonksiyon kodu (Dijital Girişleri Oku)
03	Okunan girişlerin byte sayısı (22 çıkış / 8bit = 3 byte)
AC	Çıkış 10204-10197 (1010 1100)
DB	Çıkış 10212-10205 (1101 1011)
35	00 & Çıkış 10218-10213(0011 0101)
2018	CRC hata kontrolü

- **Fonksiyon Kod 3 (0x03 hex): Saklayıcı Kaydedicileri Okuma**

Fonksiyon Kod 3, verilen adresteki cihazın saklayıcı kaydedici birimlerinin durumlarını olumak için kullanılır. Saklayıcı Kaydediciler genellikle analog çıkış birimlerinden veri okumak ya da bu birimlere veri yazmak için kullanılır. Talepte bulunan protokol veri birimi başlangıç kaydedici adresini ve okunacak kaydedici sayısını belirtir. Tek bir saklayıcı kaydedicinin durumu okuyabileceği gibi birden fazla kaydedicinin durumu da bu fonksiyon kodları kulanılarak okunabilir. Okunacak adres miktarı 1'den 125'e kadar olabilir. Cevap veren mesajdaki veri alanındaki her bir kaydedicinin durumu 2 btye'a karşılık gelir. Her kaydedici için ilk byte yüksek değerlikli bitleri ve ikinci byte düşük değerlikli bitleri kaydeder. Şekil 2.8'de bu fonksiyon kodu ile ilgili akış diyagramlar ve talep, cevap ve hata formatı durumları verilmiştir;



Şekil 2.8 : Fonksiyon Kod 3 için akış diyagramı.

Bu fonksiyon kodu için talep, cevap ya da hata durumu için protokol veri birimi içerisinde bulunan bilgiler Çizelge 2.9'da verilmiştir.

Çizelge 2.9 : Fonksiyon Kod 3 için talep, cevap ve hata formatı.

Modbus Talep		
Fonksiyon Kodu	**Başlangıç adresi**	**Okunacak bobin miktarı**
0x03	0x0000 – 0xFFFF	1-125 (0x07D)
1 Byte	2 Byte	2 Byte

Modbus Cevap		
Fonksiyon Kodu	**Byte Sayısı**	**Bobin Durumları**
0x03	2 x N	
1 Byte	1 Byte	N*2 Byte (N: kaydedici sayısı)

Modbus Hata	
Fonksiyon Kodu	**Hata Kodu**
Fonksiyon Kod + 0x80	01-02-03-04
1 Byte	1 Byte

Çizelge 2.10 ve 2.11'de, Fonsiyon kod 3 kullanılarak oluşturulan talep ve cevap mesajları için bir örnek verilmiştir;

17 adresli cihazın 40108'den 40110'a kadar olan analog çıkış birimlerinin durumunu okumak için cihaza aşağıdaki gibi talep gönderilir.

Çizelge 2.10 : Fonksiyon Kod 3 için Modbus talep örneği.

Modbus Talep : 11 03 006B 0003 7687	
11	Cihaz adresi (17 = 11 Hex)
03	Fonksiyon kodu (Analog çıkış kaydedicilerini oku)
006B	İlk kaydedici adresi (40108-40001=107=6B Hex)
0003	Kaydedici miktarı (40108'den 40110'a =3 hex)
7687	CRC hata kontrolü

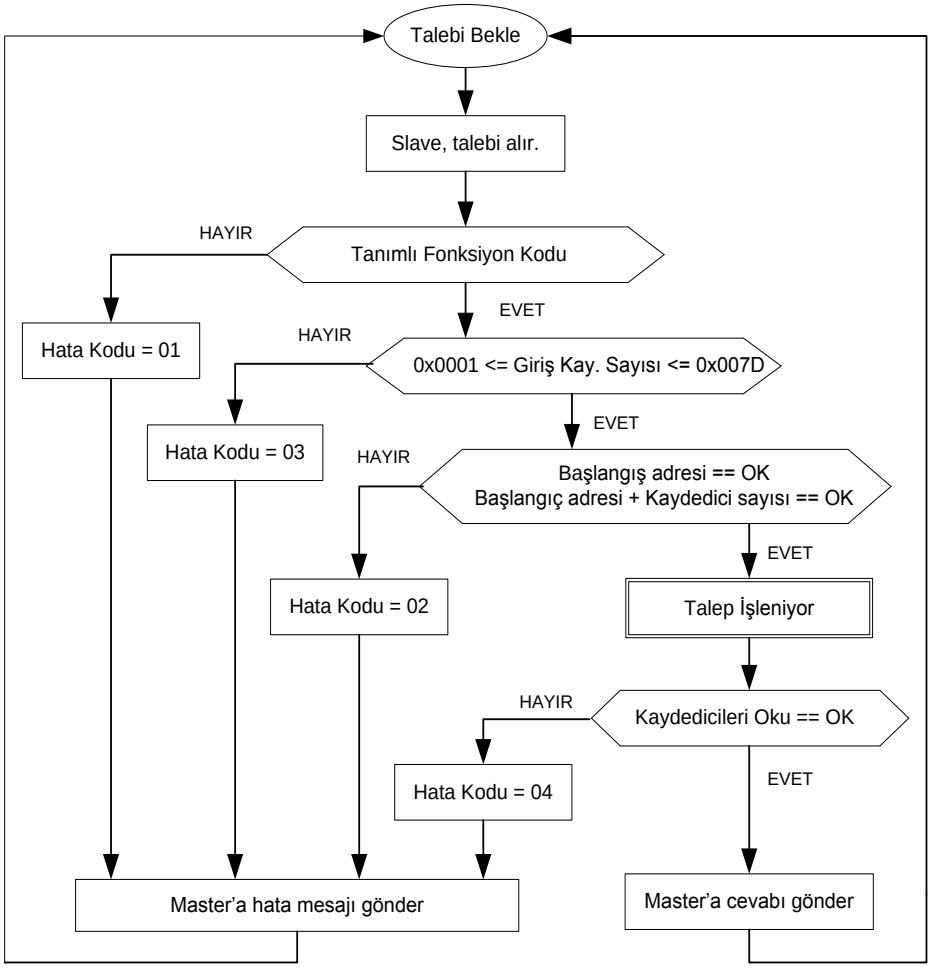
17 adresli cihazın bu talebe cevabı aşağıdaki gibi olabilir.

Çizelge 2.11 : Fonksiyon Kod 1 için Modbus cevap örneği.

Modbus Cevap : 11 03 06 AE41 5652 4340 49AD	
11	Cihaz adresi (17 = 11 Hex)
03	Fonksiyon kodu (Dijital Girişleri Oku)
06	Okunan byte sayısı (3 kaydedici x 2 byte = 6 byte)
AE41	40108 kaydedicisinin içeriği
5652	40109 kaydedicisinin içeriği
4340	40110 kaydedicisinin içeriği
2018	CRC hata kontrolü

- **Fonksiyon Kod 4 (0x04 hex): Giriş Kaydedicileri Okuma**

Fonksiyon Kod 4, verilen adresteki cihazın giriş kaydedici birimlerinin durumlarını olumak için kullanılır. Giriş kaydediciler genellikle analog giriş birimlerinden veri okumak için kullanılır. Talepte bulunan protokol veri birimi başlangıç kaydedici adresini ve okunacak kaydedici sayısını belirtir. Tek bir saklayıcı kaydedicinin durumu okuyabileceği gibi bitişik olması şartıyla birden fazla kaydedicinin durumu da bu fonksiyon kodları kulanılarak okunabilir. Okunacak adres miktarı 1'den 125'e kadar olabilir. Cevap veren mesajın protokol veri biriminde bulunan veri alanında her bir bir kaydedicinin durumu 2 btye'a karşılık gelir. Her kaydedici için ilk byte yüksek değerlikli bitleri ve ikinci byte düşük değerlikli bitleri içerir. Şekil 2.9'da bu fonksiyon kodu ile ilgili akış diyagramı verilmiştir.



Şekil 2.9 : Fonksiyon Kod 4 için akış diyagramı.

Çizelge 2.12'da bu fonksiyon kodu için talep, cevap ya da hata durumu için protokol veri birimi içerisinde bulunan bilgiler verilmiştir.

Çizelge 2.12 : Fonksiyon Kod 4 için talep, cevap ve hata durumları.

Modbus Talep		
Fonksiyon Kodu	**Başlangıç adresi**	**Okunacak bobin miktarı**
0x04	0x0000 – 0xFFFF	1-125 (0x0001- 0x07D)
1 Byte	2 Byte	2 Byte

Modbus Cevap		
Fonksiyon Kodu	**Byte Sayısı**	**Bobin Durumları**
0x04	2 x N	
1 Byte	1 Byte	N*2 Byte (N: kaydedici sayısı)

Modbus Hata	
Fonksiyon Kodu	**Hata Kodu**
Fonksiyon Kod + 0x80	01-02-03-04
1 Byte	1 Byte

Çizelge 2.13 ve 2.14'da, Fonsiyon kod 4 kullanılarak oluşturulan talep ve cevap mesajları için bir örnek verilmiştir;

17 adresli cihazın 30009'dan 30011'ye kadar analog giriş kaydedicilerinin durumunu okumak için cihaza aşağıdaki gibi talep gönderilir.

Çizelge 2.13 : Fonksiyon Kod 3 için Modbus talep örneği.

Modbus Talep : 11 04 0008 0001 B298	
11	Cihaz adresi (17 = 11 Hex)
04	Fonksiyon kodu (Analog giriş kaydedicileri oku)
0008	İlk kaydedici adresi (30009-30001=8 Hex)
0001	Kaydedici miktarı (3 kaydedici = 3 Hex)
B298	CRC hata kontrolü

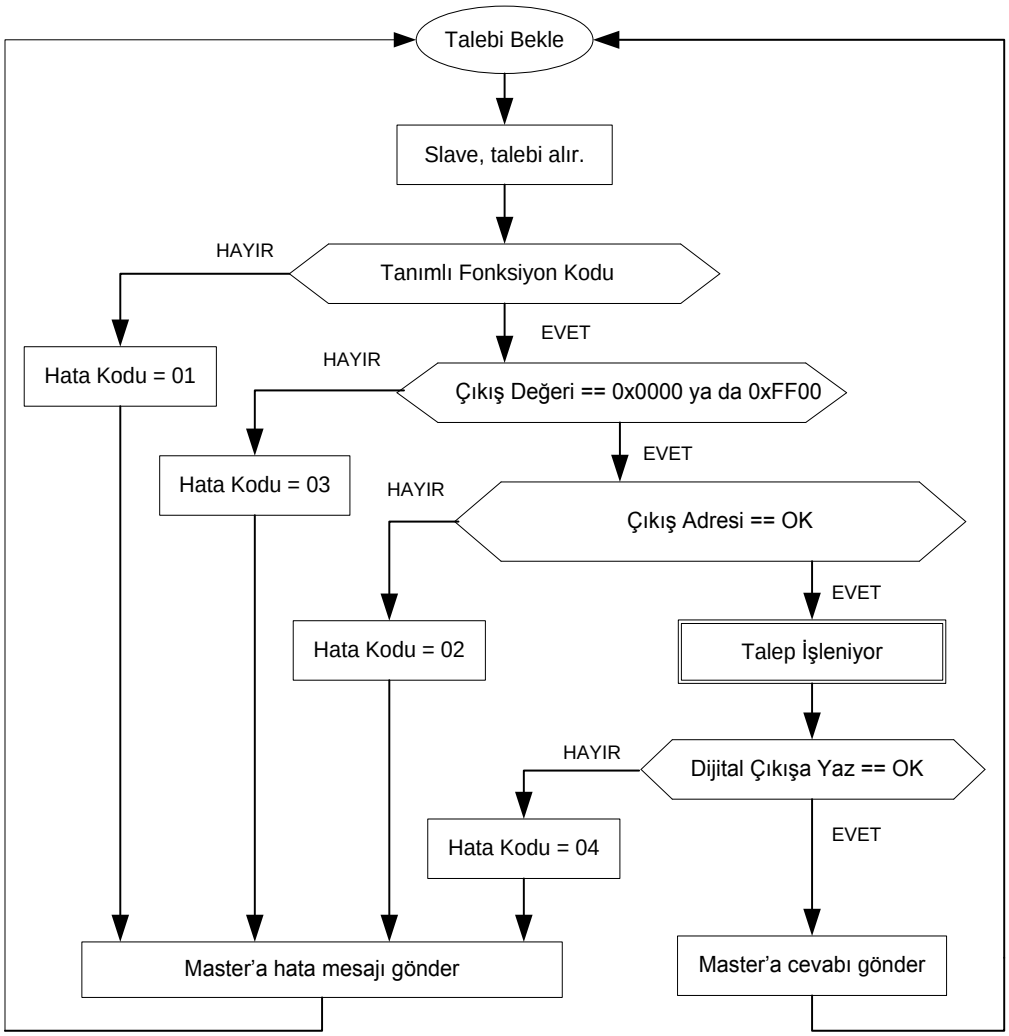
17 adresli cihazın bu talebe cevabı aşağıdaki gibi olabilir.

Çizelge 2.14 : Fonksiyon Kod 1 için Modbus cevap örneği.

Modbus Cevap : 11 04 02 000A 5652 4340 F8F4	
11	Cihaz adresi (17 = 11 Hex)
04	Fonksiyon kodu (Analog giriş kaydedicileri oku)
02	Okunan byte sayısı (3 kaydedici x 2 byte = 6 byte)
000A	30009 kaydedicisinin içeriği
5652	30010 kaydedicisinin içeriği
4340	30011 kaydedicisinin içeriği
F8F4	CRC hata kontrolü

- **Fonksiyon Kod 5 (0x05 hex) : Tek Dijital Çıkışa Yazma**

Fonksiyon Kod 5 kullanılarak belirtilen adresteki cihazın tek bir dijital çıkış biriminin durumu, açık (ON) ya da kapalı (OFF) durumuna getirilir. Çıkışı açık konumuna getirmek için FF00 hex değeri, kapalı konuma getirmek için 0000 hex değeri protokol veri birimi içersinde bulunan veri alanına yazılarak ile ilgili cihaza gönderilir. Talep gönderen protokol veri birimi içersinde kullanılacak fonksiyon kodu belirtildikten sonra sırası ile hangi çıkışın değiştirileceği ve değişim değerinin ne olacağı belirtilir. Cevap veren mesajda fonksiyon kodu belirtildikten sonra durumu değiştirilen çıkış adresi ve devamında çıkış değeri gönderilir. 0xFF00 ve 0x0000 dışındaki değerler bu fonksiyon kodu için geçerli değildir ve dijital çıkışın durumunu etkilemezler. Şekil 2.10'da bu fonksiyon kodu ile ilgili akış diyagramı verilmiştir.



Şekil 2.10 : Fonksiyon Kod 5 için akış diyagramı.

Çizelge 2.15'da talep, bu fonksiyon kodu için talep, cevap ya da hata durumu için protokol veri birimi içerisinde bulunan bilgiler verilmiştir.

Çizelge 2.15 : Fonksiyon Kod 5 için talep, cevap ve hata durumları.

Modbus Talep		
Fonksiyon Kodu	**Adres**	**Yazılacak değer**
0x05	0x0000 – 0xFFFF	0x0000 ya da 0xFFFF
1 Byte	2 Byte	2 Byte

Modbus Cevap		
Fonksiyon Kodu	**Bobin adresi**	**Yazılan Değer**
0x05	0x0000 – 0xFFFF	0x0000 ya da 0xFFFF
1 Byte	2 Byte	2 Byte

Modbus Hata	
Fonksiyon Kodu	**Hata Kodu**
Fonksiyon Kod + 0x80	01-02-03-04
1 Byte	1 Byte

Çizelge 2.16 ve 2.17'da, Fonsiyon kod 5 kullanılarak oluşturulan talep ve cevap mesajları için bir örnek verilmiştir;

17 adresli cihazın 173 numaralı dijital çıkış biriminin durumunu değiştirmek için cihaza aşağıdaki gibi talep gönderilir.

Çizelge 2.16 : Fonksiyon Kod 3 için Modbus talep örneği.

Modbus Talep : 11 05 00AC FF00 4E8B	
11	Cihaz adresi (17 = 11 Hex)
05	Fonksiyon kodu (Tek çıkışa yaz)
00AC	Çıkış adresi (173-1 = 172 =AC Hex)
FF00	Çıkışa yazılacak değer (FF00 = ON, 0000=OFF)
4E8B	CRC hata kontrolü

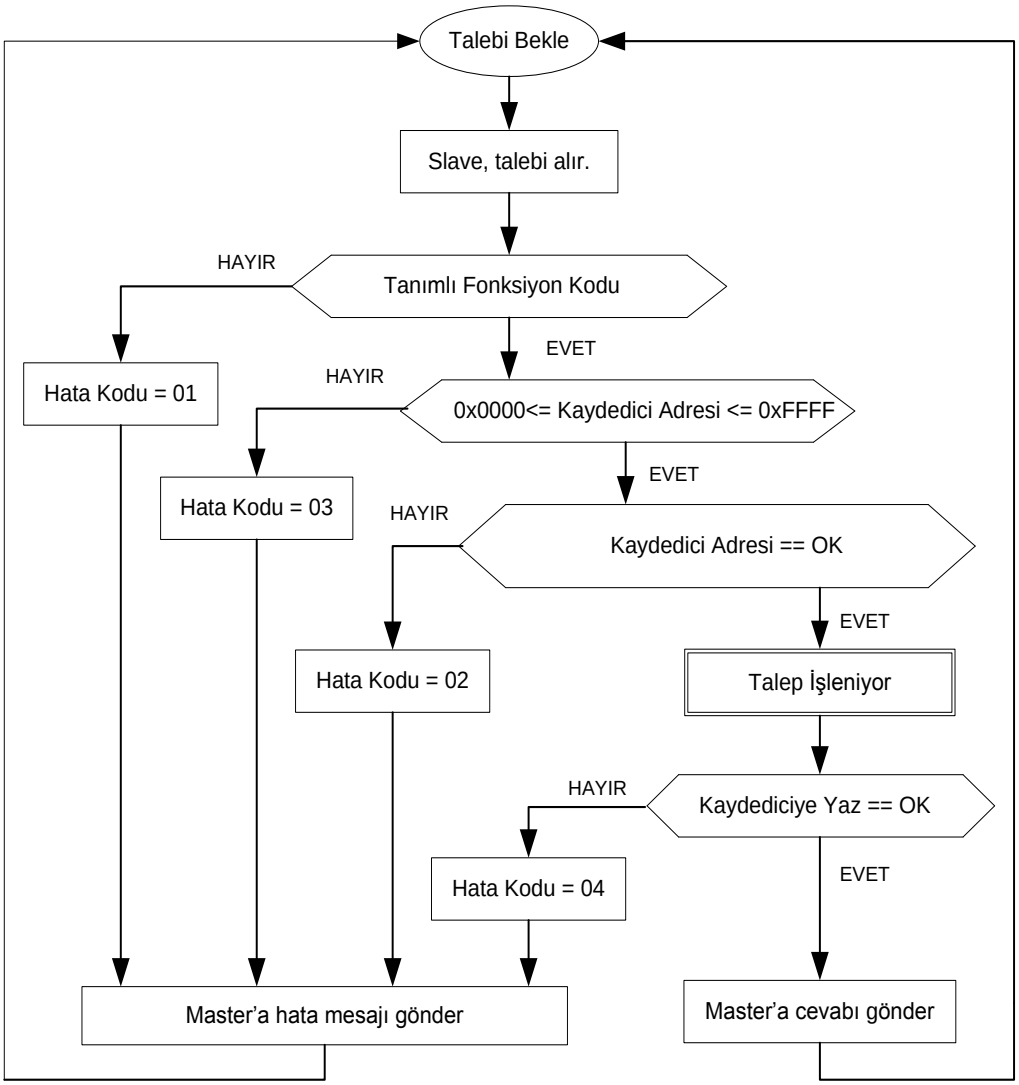
17 adresli cihazın bu talebe cevabı aşağıdaki gibi olabilir.

Çizelge 2.17 : Fonksiyon Kod 1 için Modbus cevap örneği.

Modbus Cevap : 11 05 00AC FF00 4E8B	
11	Cihaz adresi (17 = 11 Hex)
05	Fonksiyon kodu (Tek çıkışa yaz)
00AC	Çıkış adresi (173-1 = 172 =AC Hex)
FF00	Çıkışa yazılacak değer (FF00 = ON, 0000=OFF)
4E8B	CRC hata kontrolü

- **Fonksiyon Kod 6 (0x06 hex) : Tek Saklayıcı Kaydediciye Yazma**

Fonksiyon Kod 6, belirtilen adresteki cihazın tek bir saklayıcı kaydedici birimine veri yazmak için kullanılır. Talep gönderen protokol veri birimi içerisinde kullanılacak fonksiyon kodu belirtildikten sonra sırası ile hangi kaydedicinin değiştirileceği ve değişim değerinin ne olacağı belirtilir. Kaydedici adresleri sıfırdan başlar, bu nedenle bir numaralı kaydedici sıfırıncı adrese karşılık gelir. Cevap veren mesajda fonksiyon kodu belirtildikten sonra durumu değiştirilen kaydedici adresi ve devamında değeri gönderilir. Şekil 2.11'de bu fonksiyon kodları ile ilgili akış diyagramı ve Çizelge 2.18'da bu fonksiyon kodu için talep, cevap ya da hata durumu için protokol veri birimi içerisinde bulunan bilgiler verilmiştir.



Şekil 2.11 : Fonksiyon Kod 6 için akış diyagramı.

Çizelge 2.18 : Fonksiyon Kod 6 için talep, cevap ve hata durumları.

Modbus Talep		
Fonksiyon Kodu	**Adres**	**Yazılacak değer Aralığı**
0x06	0x0000 – 0xFFFF	0x0000 - 0xFFFF
1 Byte	2 Byte	2 Byte

Modbus Cevap		
Fonksiyon Kodu	**Bobin adresi**	**Yazılan Değer Aralığı**
0x06	0x0000 – 0xFFFF	0x0000 - 0xFFFF
1 Byte	2 Byte	2 Byte

Modbus Hata	
Fonksiyon Kodu	**Hata Kodu**
Fonksiyon Kod + 0x80	01-02-03-04
1 Byte	1 Byte

Çizelge 2.19 ve 2.20'da, Fonsiyon kod 6 kullanılarak oluşturulan talep ve cevap mesajları için bir örnek verilmiştir;

17 adresli cihazın 40002 numaralı analog çıkış biriminin durumunu değiştirmek için cihaza aşağıdaki gibi talep gönderilir.

Çizelge 2.19 : Fonksiyon Kod 3 için Modbus talep örneği.

Modbus Talep : 11 06 0001 0003 9A9B	
11	Cihaz adresi (17 = 11 Hex)
05	Fonksiyon kodu (Tek kaydediciye yaz)
0001	Çıkış adresi (40002-40001=1 Hex)
0003	Çıkışa yazılacak değer
9A9B	CRC hata kontrolü

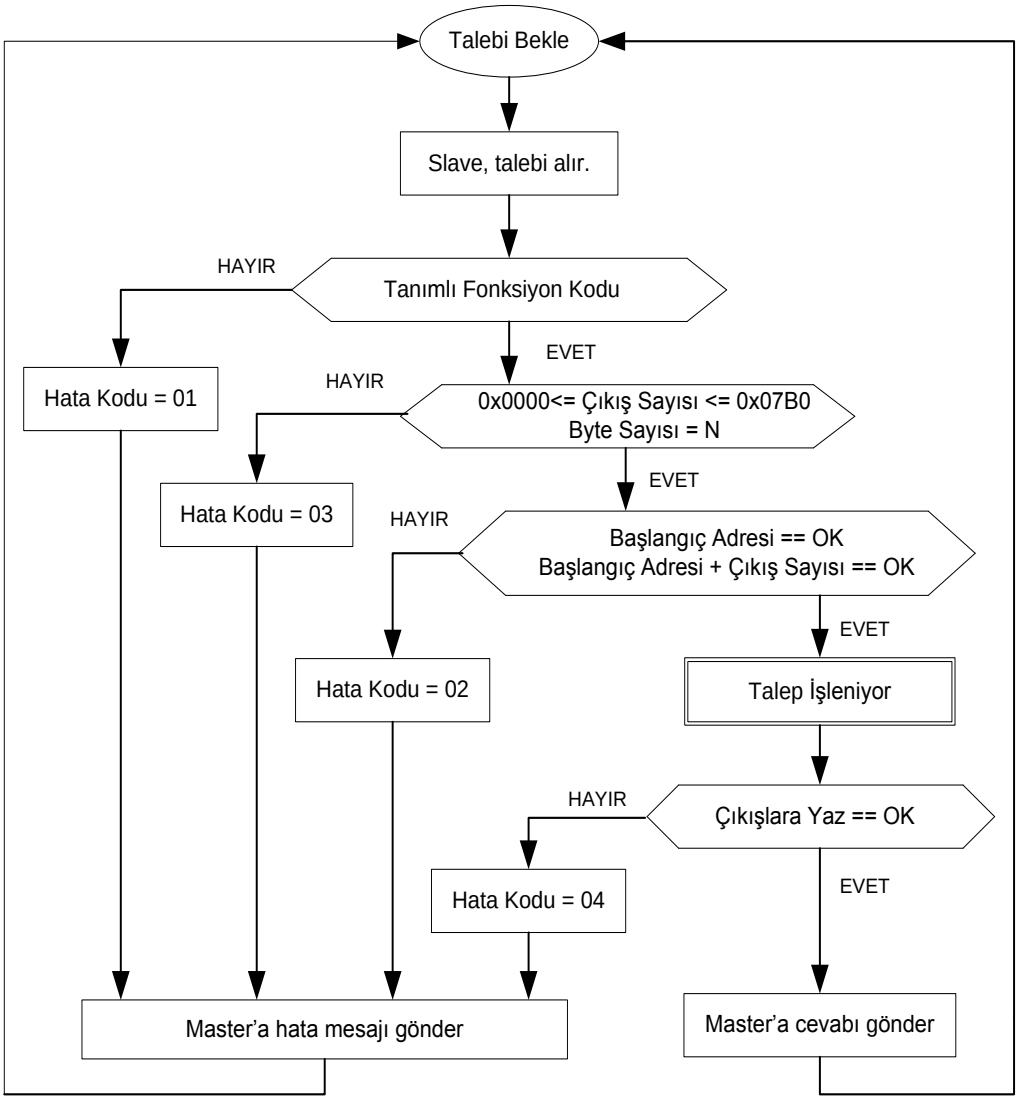
17 adresli cihazın bu talebe cevabı aşağıdaki gibi olabilir.

Çizelge 2.20 : Fonksiyon Kod 1 için Modbus cevap örneği.

Modbus Cevap : 11 05 00AC FF00 4E8B	
11	Cihaz adresi (17 = 11 Hex)
05	Fonksiyon kodu (Tek kaydediciye yaz)
0001	Çıkış adresi (40002-40001=1 Hex)
0003	Çıkışa yazılacak değer
9A9B	CRC hata kontrolü

- **Fonksiyon Kod 15 (0x0F): Çoklu Dijital Çıkışlara Yazma**

Fonksiyon Kod 15 kullanılarak belirtilen adresteki cihazın bitişik dijital çıkış birimlerinin durumlarını açık (ON) ya da kapalı (OFF) durumuna getirilir.. Protokol veri birimi içersinde kullanılacak fonksiyon kodu belirtildikten sonra sırası ile durumu değiştirilecek ilk bobinin adres bilgisini, bobin sayısı, byte sayısı ve çıkış değerlerini belirtilir. Cevap veren mesajda ise fonksiyon kodu, başlangıç adresi ve durumu değiştirilen bobin sayısı yer alır. Şekil 2.12'de bu fonksiyon kodunun nasıl çalıştığını anlatan akış diyagramına yer verilmiştir.



Şekil 2.12 : Fonksiyon Kod 15 için akış diyagramı.

Çizelge 2.21'de bu Fonksiyon bu fonksiyon kodu için talep, cevap ya da hata durumunda protokol veri birimi içerisinde bulunan bilgiler verilmiştir.

Çizelge 2.21 : Fonksiyon Kod 15 için talep, cevap ve hata durumları.

Modbus Talep				
Fonk.Kodu	**Baş. Adres**	**Çıkış Miktarı**	**Byte Sayısı**	**Değer**
0x0F	0x0000 – 0xFFFF	0x0000 - 0x07B0	N	
1 Byte	2 Byte	2 Byte	1 byte	N*1Byte

Modbus Cevap		
Fonksiyon Kodu	**Baş. Adres**	**Çıkış Miktarı**
0x06	0x0000 – 0xFFFF	0x0000 - 0x07B0
1 Byte	2 Byte	2 Byte

Modbus Hata	
Fonksiyon Kodu	**Hata Kodu**
Fonksiyon Kod + 0x80	01-02-03-04
1 Byte	1 Byte

Çizelge 2.22 ve 2.23'da, Fonsiyon kod 15 kullanılarak oluşturulan talep ve cevap mesajları için bir örnek verilmiştir;

17 adresli cihazın 20'den 29'a kadr olan 10 dijital çıkış biriminin durumunu değiştirmek için cihaza aşağıdaki gibi talep gönderilir.

Çizelge 2.22 : Fonksiyon Kod 3 için Modbus talep örneği.

Modbus Talep : 11 0F 0013 000A 02 CD01 BF0B	
11	Cihaz adresi (17 = 11 Hex)
0F	Fonksiyon kodu (Çoklu çıkışa yaz)
0013	İlk Çıkış adresi (20-1=19=13 Hex)
000A	Yazılacak çıkışların sayısı (10 = 0A Hex)
02	Byte sayısı (10 / 8 = 2 byte)
CD	Çıkış 27-20 (1100 1101)
01	000000 & çıkış 29-28 (0000 0001)
BF0B	CRC hata kontrolü

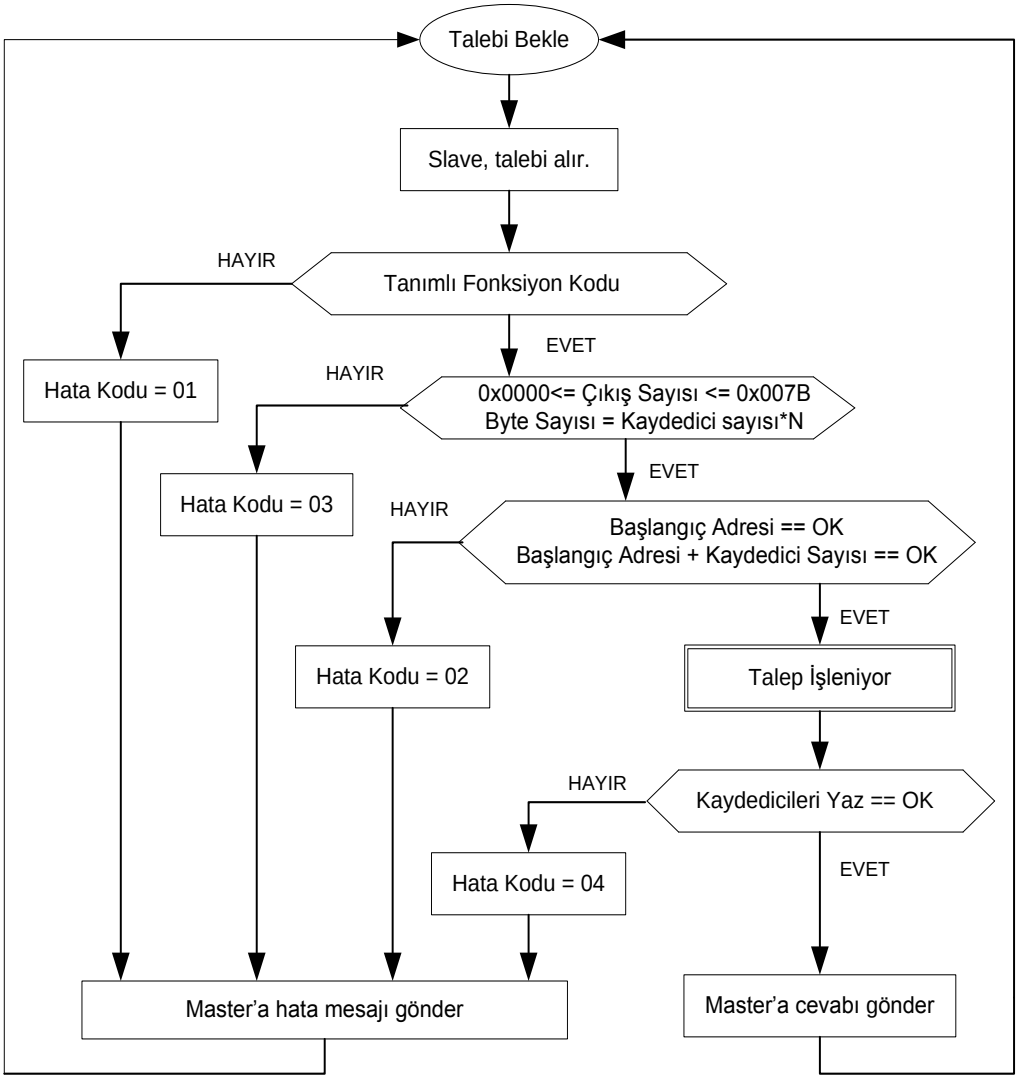
17 adresli cihazın bu talebe cevabı aşağıdaki gibi olabilir.

Çizelge 2.23 : Fonksiyon Kod 1 için Modbus cevap örneği.

Modbus Cevap : 11 0F 0013 000A 2699	
11	Cihaz adresi (17 = 11 Hex)
0F	Fonksiyon kodu (Çoklu çıkışa yaz)
0013	İlk Çıkış adresi (20-1=19=13 Hex)
000A	Yazılacak çıkışların sayısı (10 = 0A Hex)
2699	CRC hata kontrolü

- **Fonksiyon Kod 16 (0x10): Çoklu Saklayıcı Kaydedicilere Yazma**

Fonksiyon Kod 16 kullanılarak belirtilen adresteki cihazın bitişik saklayıcı kaydedicilerinin durumları değiştirilir. Talep göndren protokol veri birimi içersinde kullanılacak fonksiyon kodu belirtildikten sonra sırası ile durumu değiştirilecek ilk kaydedicinin adres bilgisini, kaydedici sayısı, byte sayısı ve çıkış değerlerini belirtilir. Cevap veren mesajda ise fonksiyon kodu, başlangıç adresi ve durumu değiştirilen kaydedici sayısı yer alır. Şekil 2.13'de bu fonksiyon kodunun nasıl çalıştığını anlatan akış diyagramına yer verilmiş ve Çizelge 2.24'da Fonksiyon kodu 15 için talep, cevap ve hata formatı durumları verilmiştir.



Şekil 2.13 : Fonksiyon Kod 15 için akış diyagramı.

Çizelge 2.24 : Fonksiyon Kod 16 için talep, cevap ve hata durumları.

Modbus Talep				
Fonk.Kodu	**Baş. Adres**	**Kayd. Miktarı**	**Byte Sayısı**	**Değer**
0x10	0x0000 – 0xFFFF	0x0000 - 0x007B	N	
1 Byte	2 Byte	2 Byte	1 Byte	N*1Byte

Modbus Cevap		
Fonksiyon Kodu	**Baş. Adres**	**Çıkış Miktarı**
0x10	0x0000 – 0xFFFF	0x0000 - 0x7B0
1 Byte	2 Byte	2 Byte

Modbus Hata	
Fonksiyon Kodu	**Hata Kodu**
Fonksiyon Kod + 0x80	01-02-03-04
1 Byte	1 Byte

Çizelge 2.25 ve 2.26'da, Fonsiyon kod 16 kullanılarak oluşturulan talep ve cevap mesajları için bir örnek verilmiştir;

17 adresli cihazın 40002'den 40003'e kadar analog çıkış biriminin durumunu değiştirmek için cihaza aşağıdaki gibi talep gönderilir.

Çizelge 2.25 : Fonksiyon Kod 3 için Modbus talep örneği.

Modbus Talep : 11 10 0001 0002 04 000A 0102 C6FO	
11	Cihaz adresi (17 = 11 Hex)
10	Fonksiyon kodu (Çoklu kaydediciye yaz)
0001	İlk kaydedici adresi (40002-40001=1 Hex)
0002	Kaydedici sayısı (2 kaydedici)
04	Byte sayısı (2 kaydedic x 2 = 4 byte)
000A	40002 kaydediciye yazılacak değer
0102	40003 kaydediciye yazılacak değer
C6FO	CRC hata kontrolü

17 adresli cihazın bu talebe cevabı aşağıdaki gibi olabilir.

Çizelge 2.26 : Fonksiyon Kod 1 için Modbus cevap örneği.

Modbus Cevap : 11 10 0001 0002 1298	
11	Cihaz adresi (17 = 11 Hex)
10	Fonksiyon kodu (Çoklu kaydediciye yaz)
0001	İlk kaydedici adresi (40002-40001=1 Hex)
0002	Kaydedici sayısı (2 kaydedici)
1298	CRC hata kontrolü

2.1.2 Modbus hata kodları

Modbus protokolünde iletişim hataları ve işletim hataları olmak üzere olmak üzere iki çeşit hata bulunmaktadır. İletişim hataları gönderilen verinin iletim hattı üzerinde bozulmasından kaynaklanmaktadır. İletişim hatalarının önlenmesi için eşlik bitinin kontrolü ve CRC kontrolü kullanılmaktadır. Alıcı cihaz, gelen verinin eşlik bitini ve CRC algoritması kontrolünü gerçekleştirir. Eğer eşlik bitinde ya da CRC denetiminde hata oluşursa mesaj alınmamış kabul edilir, fakat gelen verinin formatı doğru olmasına rağmen istenilen işlem herhangi bir nedenle gerçeklenmiyor ise işletim hatası oluşur. Bu durumda alıcı cihaz hatayı tespit ederek bir hata mesajı oluşturur. Modbus protokolü kullanılarak gerçeklenen uygulamalarda meydana gelen hatalar için belirlenen genel hata kodları Çizelge 2.27'de listelenmiştir;

Çizelge 2.27 : Kullanılan hata kodları.

Hata Kodu	Açıklama
01 (01 Hex)	Geçersiz fonksiyon kodu
02 (02 Hex)	Geçersiz veri adresi
03 (03 Hex)	Geçersiz veri değeri
04 (03 Hex)	Bağımlı cihaz hatası
05 (05 Hex)	Bilgi mesajı (Paket alınamıyor)
06 (06 Hex)	Bağımlı cihaz meşgul
08 (08 Hex)	Bellek eşlik hatası
10 (0A Hex)	Ağ geçidi hatası (Ağ geçidi kullanılmıyor)
11 (0B Hex)	Ağ geçidi hatası (Cihaz cevap vermiyor)

Modbus protokolü kullanan bütün cihazlar bu hata kodlarının tamamını destekelmek zorunda değildir. Uğgulamaya yönelik olarak tasarlanan cihazların istenilen işlemleri gerçekleştirmek için bir kaç hata kodunu destekelmesi yeterlidir. Tez çalışmasında tasarlanan FPGA tabanlı Modbus ağ geçidi cihazı sadece 1-3 10 ve 11 numaralı hata kodlarını destekleyecek şekilde programlanmıştır.

- **Hata Kodu 01 : Gçeresiz Fonksiyon Kodu**

Modbus mesajında alınan fonksiyon kodunu, RTU ağında bulunan bağımlı cihazın (slave node), ya da TCP ağında bulunan sunucu cihazın (server node), izin vermediği bir işleme karşılık geldiğini ifade eder. Bu durum genellikle gönderilen fonksiyon kodunun daha üst cihazlar tarafından desteklendği ya da cihazların bu fonksiyon koduna göre yapılandırılmadığı durumlarda meydana gelir.

- **Hata kodu 02: Geçersiz Veri Adresi**

Mdbus mesajında bulunan veri adresi, mesajı alan cihazın adres haritası dışındaki bir bölgedeki veriye erişilmek istendiği zaman o bölgede geçerli veri bulunmadığını ifade etmek için kullanılır. Daha belirgin bir biçimde belirtilen başlangıç adresi ve veri boyutu hatalı olduğu durumlarda bu hata kodu kullanılır. Örneğin 100 saklayıcı adresi bulunan bir kontrolcü için, protokol veri biriminde ilk saklayıcı adresi için 0 ve son saklayıcı adresi için 99 kullanılır. Eğer talep gönderen protokol veri biriminde başlangıç adresi 95 ve kaydedici sayısı 5 olarak belirtilmişse, talep işlemi 95 - 96 - 97 - 98 - 99 adresli kaydediciler üzerinde başarılı bir şekilde gerçeklenir, fakat başlangıç adresi 96 ve kaydedici sayısı 5 olarak belirlenmişse, talep 96 - 97 - 98 - 99 - 100 saklayıcıları üzerinde gerçeklenmek ister. Fakat 100 adresli kaydedici olmadığı için Hata kodu 2 (Geçersiz veri adresi) kullanılarak talep gönderen cihaz bilgilendirilir.

- **Hata kodu 03: Geçersiz Veri Değeri**

Modbus mesajı veri alanı içersinde bulunan değer, bağımlı cihazlar ya da sunucu cihazlar için kabuledilebilir bir değer olmadığı zaman bu hata kodu kullanılır. Başka bir değişle belirtilen adrese gönderilen bilgi Modbus protokolü tarafından belirtilen sınırların dışında ise bu hata kodu gönderilir.

- **Hata kodu 04: Bağımlı Cihaz Hatası**

Bağımlı cihaz (slave) ya da sunucu cihaz (server) Modbus talebinde istenilen istenilen eylemi gerçekleştirmek için çalışırken cihaz tarafından düzeltilemeyen bir hata oluşturğu zaman bu hata kodu üretilir.

- **Hata kodu 05: Bilgi Mesajı**

Özel olarak kullanılan bir hata kodudur. Cihaz talebi kabul edip istenilen eylemi yerine getirmeye başladığı durumlarda, bu eylem için uzun bir süre gerekebilir. Bu hata kodu kullanılarak ana cihazın zaman aşımı hatasını engellemek için ana cihaza bu fonksiyon kodu ile bilgi mesajı gönderilir.

- **Hata kodu 06: Bağımlı Cihaz Meşgul**

Programlama komutları ile beraber özel olarak kullanılan bir hata kodudur. Bağımlı cihaz ya da sunucu cihaz uzun süreli program kodunu işlerken ana cihazın talep göndermesini engellemek için bu hata kodu kullanılır. İşlem tamamlandıktan ve cihaz boşta (idle) durumuna geldikten sonra ana cihaz ya da istemci cihaz yeniden talep mesajını gönderir.

- **Hata kodu 08: Bellek eşlik hatası**

Bazı fonksiyon kodları ile bağlantılı çalışan özel bir hata kodudur. Bağımlı cihaz ya da sunucu cihaz kayıt dosyasını okumak için girişimde bulunup, hafıza da eşlik hatası keşfettiği zaman bu hata kodu kullanılarak ana cihaz (master) ya da istemci cihaz(client) bilgilendirilir.

- **Hata kodu 0A: Ağ Geçidi Hatası**

Modbus ağı üzerinde ağ geçidi cihazlar kullanıldığı durumlarda ve genellikle ağ geçidi yanlış konfigüre edildiği ya da aşırı yüklendiği zaman bu hata kodu kullanılılarak hata mesajı oluşturulur.

- **Hata kodu 0B: Ağ Geçidi Hatası**

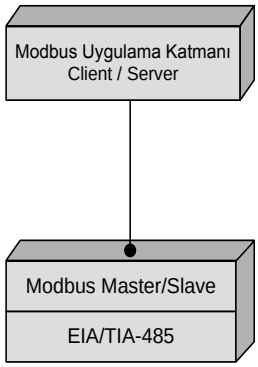
Modbus ağı üzerinde ağ geçidi cihazlar kullanıldığı zaman ve ağ geçidine ulaşılamadığı zaman bu hata kodu kullanılır. Bu kod, ağ geçidi ile bağlantının koptuğunu ve ağ üzerinde cihazın bulunmadığını ifade eder.

2.2 Modbus RTU Protokol Yapısı

Modbus RTU protokolü bir çeşit ana birim/bağımlı birim (master/slave) arasında veri haberleşmesi yapan bir protokolüdür yani ağ üzerinde bir ana cihaz (master node) ve bu cihaza bağlı çalışan bağımlı cihazlar (slave node) bulunur. Bağımlı cihazlar ana cihazdan bir talep gelmediği sürece mesaj iletmezler ve birbirleri ile haberleşmezler. Ana cihaz, belirlediği bağımlı cihaza talep gönderir ve bağımlı cihazda bu talebe karşılık olarak bir cevap gönderir. Ağ üzerinde aynı anda ana cihaz ile sadece bir bağımlı cihaz haberleşir. Modbus kulanılarak seri haberleşme yapılırken, OSI modelinin yedinci katmanında *Modbus Uygulama Protokolü* çalışır.

OSI modelinin altıncı (Sunum), beşinci (Oturum), dördüncü (Taşıma) ve üçüncü (Ağ) katmanları boştur. İkinci katmanda (Veri bağlantı katmanı) *Modbus Seri Haberleşme Protokolü* çalışır. Modbus seri hat protokolü Mobbus RTU ve Modbus ASCII olmak üzere iki farklı iletim modunu destekler. OSI modelinin birinci katmanda (Fiziksel katman), ise veri iletimi için genellikle *EIA/TIA-485 (RS 485)* ya da *EIA/TIA-232 (RS 232)* fiziksel ara birimi kullanılır. En çok tercih edilen *EIA/TIA-485* standardı İki Kablolu Birim'dir (Two-Wire Interface). Bu çalışma veri bağlantı katmanınva Modbus RTU iletim modu ve fiziksel katmanda veri iletimi için İki Kablolu RS 485 standartı kullanılmıştır. Şekil 2.14'de Modbus RTU protokolünün OSI modeline göre katmanları gösterilmiştir.

Katman	ISO/OSI Modeli	
7	Uygulama Katmanı(application layer)	Modbus Uygulama Protokolü
6	Sunum Katmanı (presentation layer)	Boş
5	Oturum Katmanı (session layer)	Boş
4	Taşıma Katmanı (transport layer)	Boş
3	Ağ Katmanı (network layer)	Boş
2	Veri Bağlantı Katmanı (data link layer)	Modbus Seri Hat Protokolü
1	Fiziksel Katman (physical layer)	EIA/TIA-485 (EIA/TIA-232)

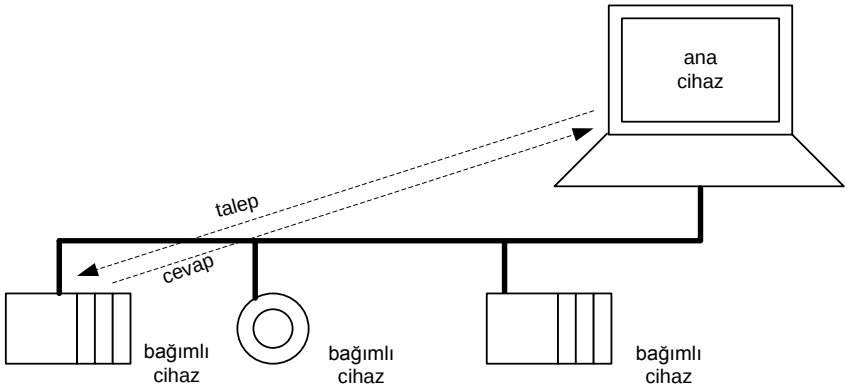


Şekil 2.14 : OSI modeline göre Modbus RTU.

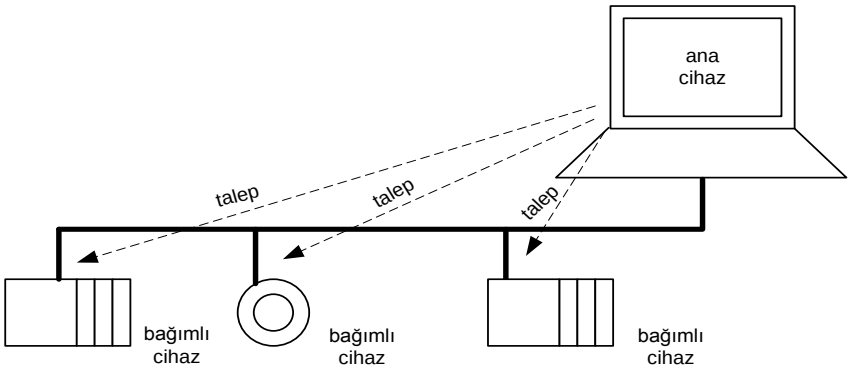
Uygulama katmanında gerçelenen Modbus Uugulama Protokolü ile ilgli yeterli bilgi Bölüm 2.1'de verilmiştir. Bu bölümde Modbus seri haberleşme protokolünün OSI modelinin veri bağlantı katmanında (ikinci katman) ve fiziksel katmanda nasıl gerçeklendiği ve veri paketinde hata kontrolünün nasıl yapıldığı anlatılmıştır.

Modbus seri haberleşme ağı üzerinde bir adet ana cihaz ve maksimum 247 adet bağımlı cihaz bulunur. Ana cihaz, bağımlı cihazlara tek yönlü (unicast mode) ve yayın modu (broadcast) olmak üzere iki türlü mesaj gönderebilir. Tek yönlü modda ana cihaz tek bir bağımlı cihaz ile haberleşir. Bağımlı cihaz talebi alıp işledikten sonra ana cihaza geri mesaj gönderir. Her bağımlı cihazın tek bir adresi vardır (1'den 247'ye kadar). Yayın modunda ana cihaz ağ üzerindeki bütün bağımlı cihazlara talep gönderir ve hiç bir bağımlı cihaz ana cihaza geri dönüş yapmaz. Sıfır adresi yayın modu için ayrılmıştır ve bütün bağımlı cihazlar tarafından tanınır. Bu haberleşme modu bağımlı cihazların mesajı alıp almadığından kesin olarak emin olunamadığı

için çok fazla tercih edilmez. Bu çalışmada ana cihaz ve bağımlı cihazlar arasında tek yönlü (unicast mode) haberleşme gerçekleştirilmiştir. Şekil 2.15 ve 2.16'de verilen şekillerde her iki haberleşme modu ifade edilmiştir.



Şekil 2.15 : Tek yönlü iletim modu.



Şekil 2.16 : Yayın modu.

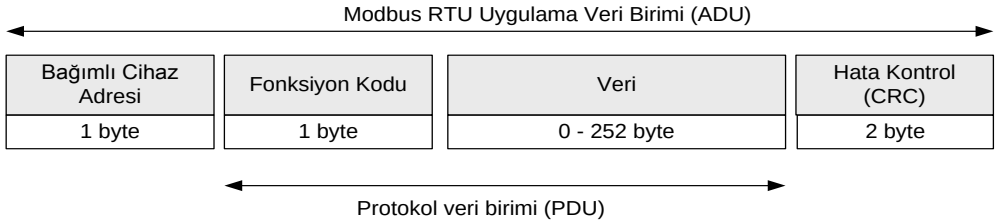
2.2.1 Modbus RTU uygulama katmanı

Bölüm 2.1 Modbus uygulama protokolü bölümünde, protokol veri biriminin alt katmanlardan bağımsız olarak nasıl oluşturulduğu anlatılmış ve alt katmanlarda çalışan protokollere göre bazı ek birimlerin eklenebileceği ifade edilmiştir. Modbus RTU haberleşme protokolünde protokol veri birimine ek olarak adres birimi ve hata kontrol birimi (CRC) eklenir. Modbus adres uzayı 1'den 256'ya kadar numaralandırılır. Ana cihazın adresi yoktur, fakat her bağımlı cihazın kendine özel bir adresi vardır. Sıfırıncı adres yayın modu (broadcast) için ayrılmıştır. Şekil 2.17'de Modbus adres uzayı gösterilmiştir.

0	1-247	248-256
Yayın modu adresi	Bağımlı cihaz adresleri	Rezerve

Şekil 2.17 : Modbus adres uzayı.

Adres biriminde ana cihazın talep göndermek istediği bağımlı cihazın adresi yazılır. Bağımlı cihazlar için geçerli adres aralığı 1-247'dir. Bağımlı cihaz talebe cevap verirken adres bölümüne kendi adresini yazar ve böylece ana cihaz hangi bağımlı cihazın cevap gönderdiğini anlar. Hata kontrol birimi de gönderilen mesajın doğruluğunu kontrol eder. Modbus RTU protokolü hata denetimi için Döngüsel artıklık denetimi (CRC) algoritmasını kullanır. CRC ile ilgili detaylı bilgi Bölüm 2.2.4'de verilmiştir. Şekil 2.18'de Modbus seri hat protokolünün veri yapısı gösterilmiştir.



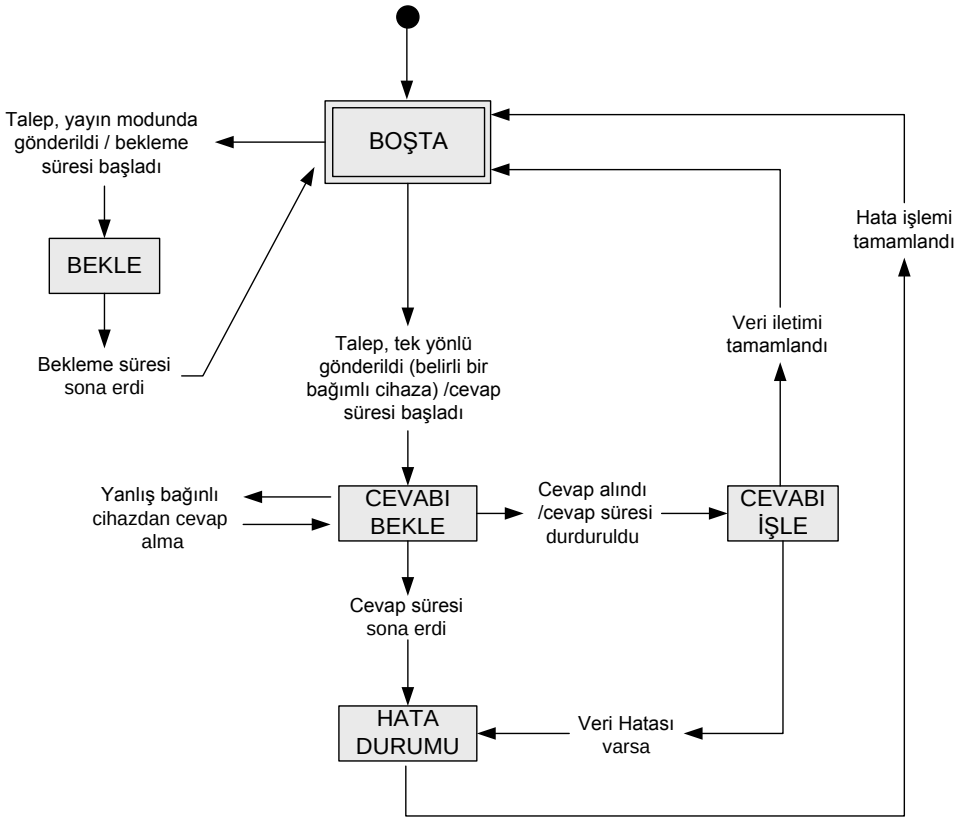
Şekil 2.18 : Modbus RTU veri paketi yapısı.

Modbus RTU protokolü bağımlı cihaz adresi için 1 byte, fonksiyon kodu için 1 byte veri alanı için maksimum 252 byte ve CRC hata kontrol alanı için 2 byte ayrımaktadır. Böylece bir Modbus RTU uygulama veri biriminin maksimum uzunluğu 256 byte olur.

2.2.2 Modbus RTU veri bağlantı katmanı

Modbus seri hat protokolü desteklediği seri iletim modlarından (RTU ve ACSII modları) bağımsız olarak ana cihaz ve bağımlı cihazlar arasında veri iletimini gerçekleştirir. Şekil 2.19'da verilen akış diyagramı ana cihazın seri haberleşme modundan bağımsız olarak veri iletimini nasıl gerçekleştirdiğini ifade eder. Sisteme güç geldikten sonra ana cihaz ilk olarak **boşta (idle)** durumuna gelir. Bu durum beklenen bir cevap olmadığını ve cihazın talep göndermeye hazır olduğunu ifade eder. Ana cihaz sadece bu durumda iken talep gönderir. Eğer ana cihaz tek yönü bir talep göndermişse (yani belirli bir bağımlı cihaza talep gönderilmişse) **cevabı bekle (waiting for reply)** durumuna geçer ve cevap bekleme süresi başlatılır. Cevap

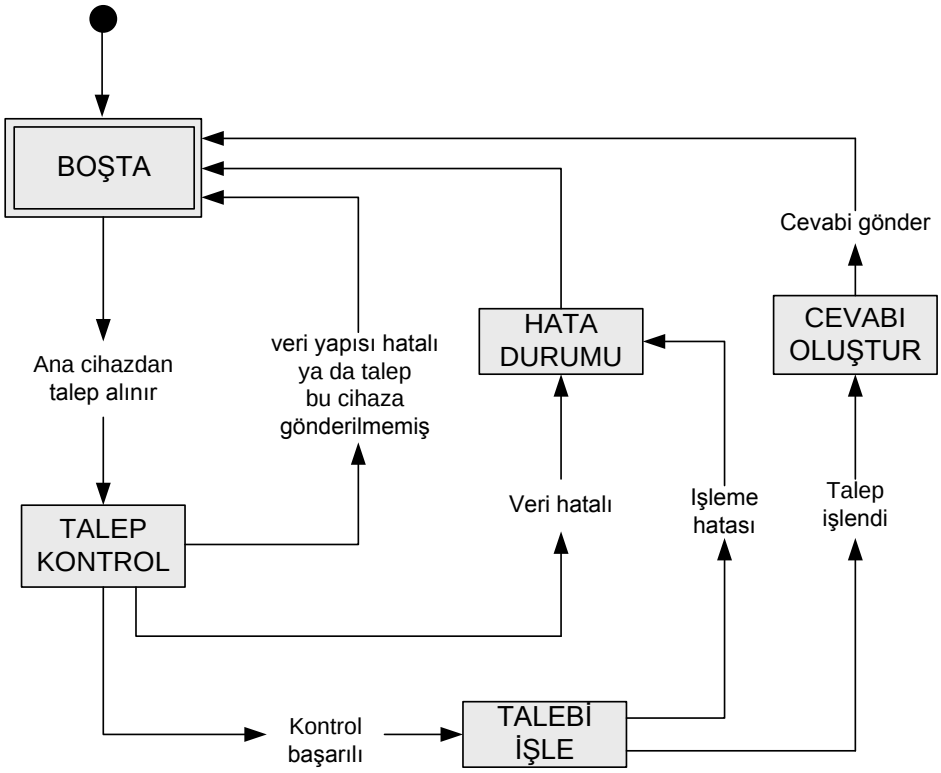
bekleme süresi tanımlanmasının amacı cevap gelmediği zaman cihazın sürekli bu durumda beklemesini engellemek ve süre dolduğu zaman bu durumdan başka bir duruma geçmesini sağlamaktır. Cevap bekleme süresi uygulamaya bağlı olarak programcı tarafından ayarlanabilir. Cevap alındığı zaman ana cihaz veriyi işlemeden önce doğruluğunu kontrol eder. Eğer alınan veride bir hata varsa ana cihaz **hata durumuna (processing error)** geçer ve cevap bekleme süresi durdurulur. Veri doğru ise ana cihaz **cevabı işle (processing reply)** durumuna geçer ve veri iletimi tamamlanarak yeniden **boşta (idle)** durumuna geçileren talep göndermeye hazır hale gelir. Başka bir bağımlı cihazdan veri alınmışsa cevap bekleme süresi devam eder. Cevap alınmazsa cevap bekleme süresi biter ve hata kodu oluşturulur ve ana cihaz tekrar **boşta (idle)** durumuna döner. Ana cihaz yayın modunda bir talepte bulunmuşsa, bağımlı cihazlardan bir cevap beklenmez. Ana cihaz belirlenen bir bekleme süresi (turnaround delay) kadar bekler ve sonra yine başlangıç durumuna gelir.



Şekil 2.19 : Ana cihaz için durum diyagramı.

Şekil 2.20'de verilen akış diyagramı bağımlı cihazın seri haberleşme modundan bağımsız olarak veri iletimini nasıl gerçekleştirdiğini ifade eder. **Boşta (idle)**

durumu bağımlı cihazın ana cihazdan talep almak için beklediği durumdur. Talep alındığı zaman bağımlı cihaz önce talebi kontrol etmek için **talep kontrol (checking request)** durumuna geçer. Bu durumda, talep farklı bir cihaza gönderilmişse yani adres bilgisi hatalı ise ya da veri yapısı hatalı ise cihaz **boşta (idle)** durumuna geçer. Geçersiz fonksiyon kodu, ya da veri formatı hatalarında hata varsa cihaz **hata durumuna (error reply)** durumuna geçer ve ana cihaza hata cevabı gönderilir. Eğer gönderilen talepte bir hata yoksa cihaz **talebi işle (processing request)** durumuna geçerek talep işler, işleme durumunda bir hata oluşmazsa **cevabı oluştur (formatting reply)** durumuna geçer ve ana cihaza cevap gönderilerek yeniden **boşta (idle)** durumuna geçilir.



Şekil 2.20 : Bağımlı cihaz durum diyagramı.

Cihazlar RTU (Remote Terminal Mod) modu ile Modbus seri haberleşmesi yaparken, mesajdaki her 8-bitlik byte, iki 4-bitlik hexedecimal karakter içerir. Bu haberleşme modunun en büyük avantajı, daha büyük karakter yoğunluğu sayesinde aynı haberleşme hızında ASCII moda göre daha fazla veri iletebilmesidir. Modbus RTU modunda 1 byte'lık veri paketi içeriği Çizelge 2.28'de gösterilmiştir.

Çizelge 2.28 : Modbus RTU 1 byte'lık veri paket içeriği.

Kodlama Sistemi	8-bit ikili sistem (binary)
Byte başına bitler	1 başlangıç biti (start bit)
	8 bit veri biti (data bits) en az önemli bit ilk gönderilir.
	1 eşitlik biti (parity bit)
	1 stop biti
Toplam	11 bit

Karakterler ya da byte verileri seri olarak soldan sağa doğru, en az önemli bitten (least significant bit) en önemli bite (most significant bit) doğru gönderilir. Sırasıyla ilk gönderilen bit başlangıç biti (start bit) olarak isimlendirilir, ardından 8-bitlik veri bitleri, devamında varsa eşlik biti (parity bit) ve durma biti (stop bit) gönderilir. Eğer eşitlik biti kullanılacaksa bit gönderim sırası Şekil 2.21'deki gibi olur.

Şekil 2.21 : Modbus RTU 1 byte'lık veri paketi (eşlik biti içeren).

Eşlik biti olarak genellikle çift eşlik (even parity) kullanılır. Fakat buna alternatif olarak tek eşlik (odd parity) biti kulanılabilir ya da eşlik biti kullanılmaz (no parity). Eşitlik biti kullanılmadığı durumda, iki adet durma biti kullanılır ve bit sırası Şekil 2.22'deki gibi olur;

Şekil 2.22 : Modbus RTU 1 byte'lık veri paketi (eşlik biti içermeyen).

Kullanıcı, cihazları çift eşliğe ya da tek eşliğe göre konfigüre etmeli ya da eşlik biti kullanmamalıdır. Bu konfigürasyon ayarı ile eşlik bitinin her karakter dizisinin içersine nasıl yerleştirileceğine karar verilir. Çift ya da tek eşlilik seçilmişse, her karakter bilgisinin veri bölümü (RTU için 8 bit) içersindeki '1' bit değerleri sayılır. Eşlik biti değeri çift eşlilik seçilmişse '1' bitinin toplam sayısı çift yapacak şekilde, tek eşlilik seçilmişse tek yapacak şekilde '1' ya da '0' olarak ayarlanır. Örneğin karakter bilgisi içersindeki veri bölümü 1100 0101 ise '1' bitlerinin toplam sayısı dörttür ve çift eşlik kullanılacaksa eşlik biti değeri '0', tek eşlik kullanılacaksa eşlik biti değeri '1' atanır. Mesaj iletildiği zaman eşlik biti hesaplanır ve her karakter bilgisi için uygulanır. Mesajı alan cihazda '1' değerlikli bitleri sayar ve konfigüre edildiği

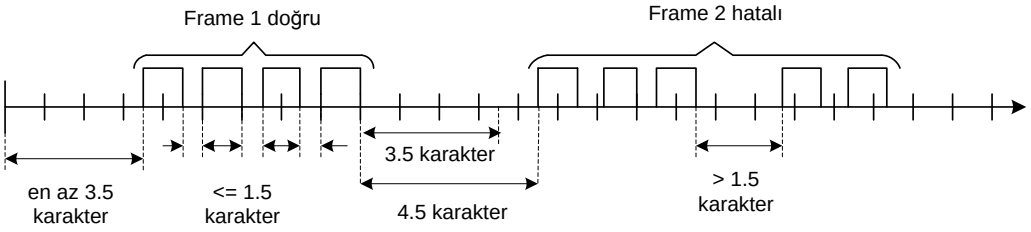
eşlik biti bilgisi ile uyuşmazsa hata mesajı oluşturur. Bu nedenle Modbus seri heberleşme hattı üzerindeki tüm cihazlar aynı eşlik bitine göre konfigüre edilmelidir.

Sonuç olarak modbus veri yapısı Çizelge 2.29'de gösterildiği gibi 1 Byte bağımlı cihaz adres bilgisi, 1 Byte fonksiyon kodu bilgisi, maksimum 252 Byte veri bilgisi ve 2 Byte hata kontrol bilgisi içerir. Bir RTU mesajı içerinde en fazla 256 byte veri gönderilir.

Çizelge 2.29 : Modbus RTU veri paketi.

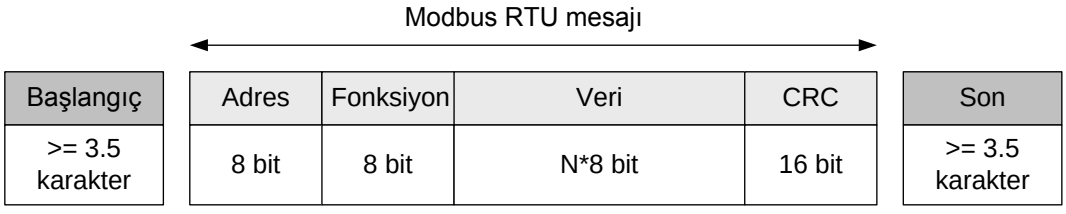
Adres	Fonk. Kodu	Veri	CRC
1 Byte	1 Byte	0 – 252 Byte	2 Byte

Modbus mesajı, mesajı iletecek cihaz tarafından başlangıcı ve sonu bilinen bir veri alanı içersine yerleştirilir. RTU modda veri alanları en az 3.5 karakterlik bir bekleme süresi ile birbirinden ayrılır. Tüm mesajın bir karakter akışı şeklinde iletilmesi gerekir. Eğer iletilen iki karakter arasında 1.5 karakter süresinden daha fazla zaman geçerse, mesaj tamamlanmamış ve hatalı olur, alıcı tarafından kabul edilmez. Şekil 2.23'de iki Modbus mesajının iletimi sırasında başlangıç ve bitiş süreleri ile bir mesajın karakterleri arasındaki iletim süresi ifade edilmiştir.



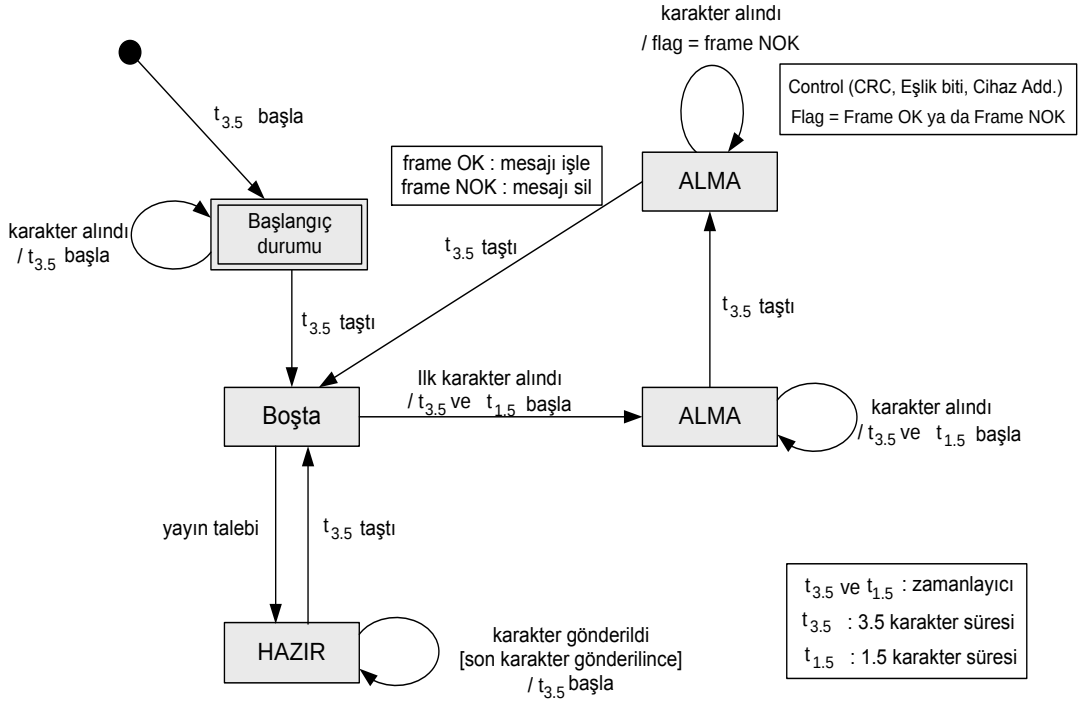
Şekil 2.23 : Modbus RTU mesaj iletimi süreleri.

Şekil 2.24'de Modbus seri haberleşme ağı içersinde veri alanına bir modbus mesajının nasıl yerleştirileceği gösterilmiştir.



Şekil 2.24 : Modbus RTU mesajı başlangıç ve sonu.

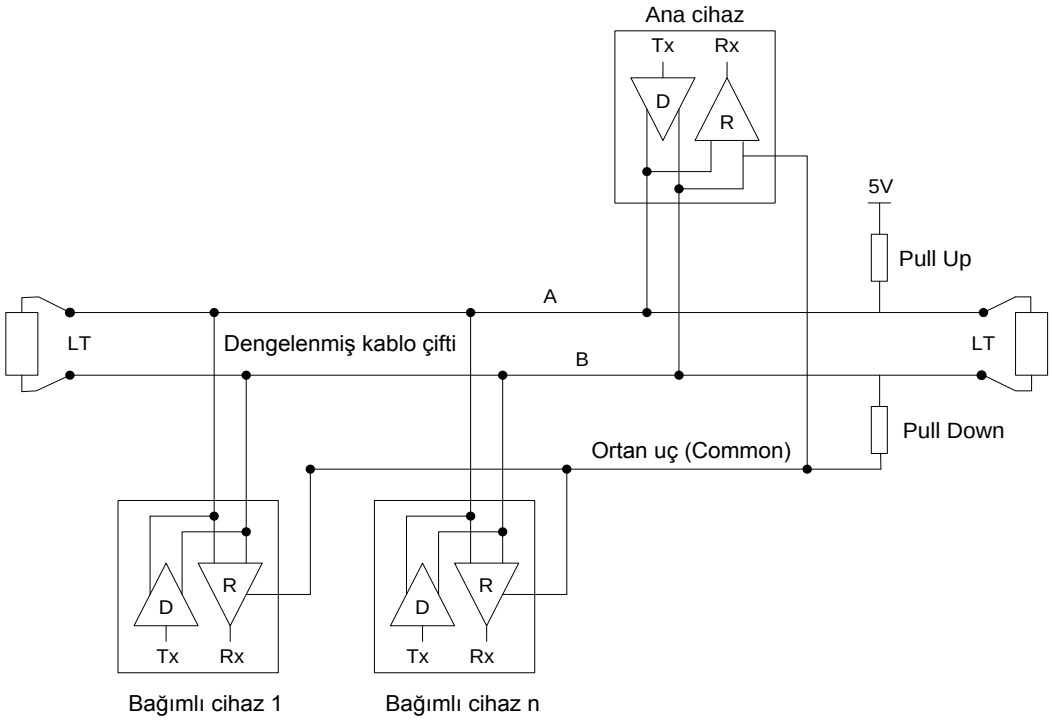
Bu durumda ağ üzerinde Modbus RTU protokolü ile veri iletimi yapan ana ve bağımlı cihazlar Şekil 2.25'de verilen akış diyagramına göre çalışırlar. RTU modunda çalışan ana ve bağımlı cihazların, veri almaya ya da göndermeye hazır durumunu ifade eden boşta (idle) durumuna gelmeden önce 3.5 karakter iletim süresi ($t_{3.5}$) kadar beklemeleri gerekmektedir. Bu süre mesaj paketleri arasında olması gereken gecikme süresidir. Cihaz çalıştığında $t_{3.5}$ süresi başlar ve cihaz **başlangıç durumuna (initial state)** gelir. Bu durumda karakter alımı olursa $t_{3.5}$ süresi yeniden başlatılır. Süre dolduğu zaman cihaz **boşta (idle)** durumuna gelir. Bu durum cihazın mesaj göndermeye ya da almaya hazır olduğunu ifade eder. Cihaz mesaj gönderecekse **yayın (emission)** durumuna geçer ve gönderme işlemi sona erince cihaz yeniden **boşta (idle)** durumuna gelir. Cihaz mesaj alacaksa **kabul (reception)** durumuna geçer. Alınan verinin kontrolü için **kontrol ve bekleme (control and waiting)** durumuna geçilir. Burada, verinin doğru cihaza geldiğini kontrol etmek için bağımlı cihaz adresi, karakterlerin verilerini kontrol etmek için eşlik bitleri kontrol edilir ve tüm Modbus mesajının doğruluğunu kontrol etmek için verinin Döngüsel Artıklık Denetimi (Cyclical Redundancy Checking (CRC)) ile veri denetimi yapılır. Buradaki işlemler sona erince, mesaj doğru ise cihaz mesajı işler, yanlış ise siler ve yeniden **boşta (idle)** durumuna gelir.



Şekil 2.25 : Modbus RTU iletim modu durum diyagramı.

2.2.3 Modbus RTU fiziksel katman

Modbus RTU seri hat haberleşmesi fiziksel katmanda EIA/TIA-485 standartını (RS485) kullanarak veri iletimi sağlar. RS485, uzun mesafelerde, gürültülü ortamlarda, daha yüksek hız gerektiren, simetrik bağlantıya ihtiyaç duyan ve daha fazla alıcı verici birimin bulunduğu sistemlerde kullanılmak üzere geliştirilmiş bir seri iletim ortamıdır. RS485 ortamı, veriyi üst katmanlarda çalışan protokollerden bağımsız olarak iletir. Simetrik haberleşmede iki veri hattı arasındaki diferansiyel gerilim ölçülür. Veri aktarımı iki kablolu bağlantı ya da dört kablolu bağlantı ile sağlanabilir. Ey yaygın kullanım şekli iki kablo ile yapılan bağlantıdır (half-dublex transmission). Ağa bağlı olan her cihazın R_x ve T_x olmak üzere iki ucu bulunur. T_x ucu sürücü (driver) birime ve R_x ucu da alıcı (receiver) birime bağlıdır. RS485 dengelenmiş hat üzerinden (balanced pair) verinin iletilmesi mantığına dayanır. Veri, iki veri kablosu A ve B arasındaki gerilim farkı üzerinden diferansiyel olarak aktarılır. Bu iki uç dengelenmiş hatlardır. Ana cihaz ve bağımlı cihazlar arasında en yaygın kullanılan iki kablolu bağlantı, R_x ve T_x uçları ve dengelenmiş kablo çifti (A ve B) Şekil 2.26'de gösterilmiş ve Çizelge 2.30'de bağlantılar açıklanmıştır.



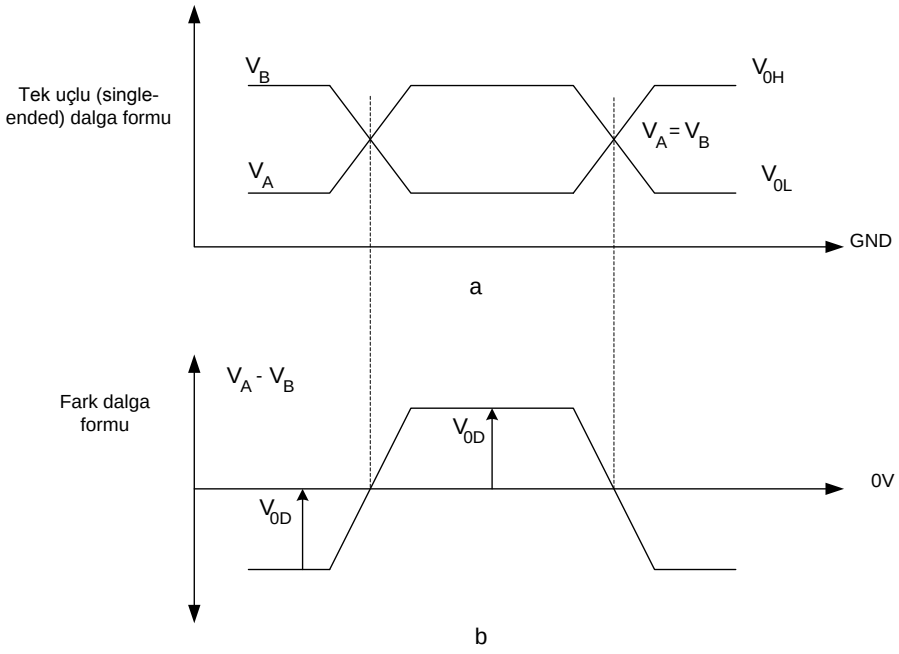
Şekil 2.26 : İki koblolu tapoloji.

Çizelge 2.30 : RS485 devre bağlantıları.

RS485 adı	Açıklama
B	V1 gerilimi (V1>V0 ise 1 (LOW) durumu)
A	V0 gerilimi (V0>V1 ise 0 (HIGH) durumu)
C	Ortak uç, Common mode

Dengelenmiş hatlarda vericilerde diferansiyel sürücüler (fark kuvvetlendiriciler, driver), alıcılarda ise diferansiyel alıcılar (fark alıcılarlar, receiver) bulunur. Dengelenmiş hattın mantığı, veri iletilirken iletim ortamında, iki hatta aynı yolla gürültü bulaşırsa, alıcının girişindeki fark kuvvetlendiriciler bu iki hattın farkını alacağından toplam gürültü miktarı sıfıra yakın olacaktır. Dolayısıyla iki hat boyunca veriye binen gürültünün alıcı girişindeki etkisi minimuma inecektir. Ayrıca fark kuvvetlendiricilerin ortak nokta (common mode) özelliğinin iyi olması verici ve alıcının topraklama farklarından oluşabilecek sorunları sıfıra indirgeyecektir. RS485 sürücülerinin bu özelliğine ek olarak, burulmuş kablo (twisted pair) ve doğru sonlandırma direnci kullanılması ile veriler çok uzun mesafelere (1200 m) çok yüksek hızlarda (10 Mbit/s) taşınabilmektedir. Burulmuş kablo iki nokta arasında faz farkı yaratarak gürültünün azalmasına yardımcı olurken, sonlandırma direnci ile veri yolundaki yansımayı minimuma indirecektir.

RS485 standartı alıcı ve vericinin toprakları arasında 7 Volt'a farka kadar izin verir. Bu da RS485 ağının, 5 Volt besleme gerilimine göre -7 Volt ile +12 Volt (5+7) arasında kaymaya müsaade etmesi demektir. Böylece alıcı ve vericinin farklı güç kaynaklarından dolayı oluşacak toprak farkından dolayı oluşa sorunlar minimuma indirgenir. RS 485 sürücüleri 60 mA akım sağlayacak şekilde dizayn edilir. Bu değer 32 adet alıcının devreye bağlanması, sonlandırma direncinin devreye dahil edilmesi ve en kötü şartlar göz önüne aınarak belirlenmiştir. Sürücüler termal shut down (ısı ile kapanma) özelliğine sahip olup A ve B uçlarından fazla akım çekilmesine izin vermeyerek sistemi korurlar. Ayrıca RS485 alıcılarının giriş direnci 12 Kohm olarak standartlaştırılmışlardır. Şekil 2.27 a'da A ve B kablolarından taşınan sıra ile V_a ve V_b sinyallerinin nasıl değiştiği ve b'de V_a-V_b sinyallerinn farkı gösterilmiştir.



Şekil 2.27 : a) A ve B'den geçen sinyalin değişimi, b) Va-Vb sinyal farkı.

RS 485 ortamı ile veri, yukarıda ifade edildiği gibi A ve B olarak isimlendirilen (bazı kaynaklarda + ve – ya da D0 ve D1) iki kablo üzerinden iletilir. Kablolardan biri üzerinde düşük sinyal taşınırken diğerinde yüksek sinyal taşınır. İletilen verinin durumu bu iki sinyalin aritmetik farkından tesbit edilir. Eğer Va-Vb değerlerinn farkı 200mV'dan büyük ya da eşit ise iletilen verinin değeri lojik 1, -200mV'dan büyük ya da eşit ise iletilen verinin değeri lojik 0 kabul edilir.

RS485 veri hattı üzerindeki cihazlar veri alışverişi olmadığı durumlarda yüksek empedans moduna geçip durağan potansiyel sağlamaktadır (tri-state mode). Bu mod dinleme modu olarak da isimlendirilebilir. Bu durumda hattı güvenli bir seviyede tutabilmek için pull-up ve pull-down dirençleri kullanılır. Hattın sonunda (genelde ana cihaz tarafında) A hattı bir direnç ile 5 Volta, B hattı da bir direnç ile toprağa bağlanır. Bu direnç değerlerinin seçimi, sonlandırma direncinin değerine bağlı olarak hesaplanır. Sonlandırma direnci (Line Termination), hat üzerindeki yansımaları engelleyerek verinin doğru bir şekilde iletilmesine yardımcı olur. Eğer iletim hattının empedansı uç noktaların empedansı ile uyumlu olmazsa iletilen sinyal alıcı tarafından tamamen alınamaz ve sinyalin bir kısmı hatta yansır. Bir hatta yansıma var ise hattaki sinyal kararlı düzeye gelemeyeceği için verinin doğru tespitinde sorunlar yaşanır, çünkü hattaki yansımanın etkisi ile sinyal seviyesi sürekli değişir. Sistemde sonlandırma yapılıp yapılmayacağına kablo uzunluğu ve veri hızına göre karar

verilir. Bunun için önerilen kural, eğer iletim hattının gecikme süresi (propogation delay) bir bit süresinden daha az ise sonlandırmaya gerek yoktur.

2.2.4 Döngüsel artıklık denetimi (CRC) ile Modbus RTU veri denetimi

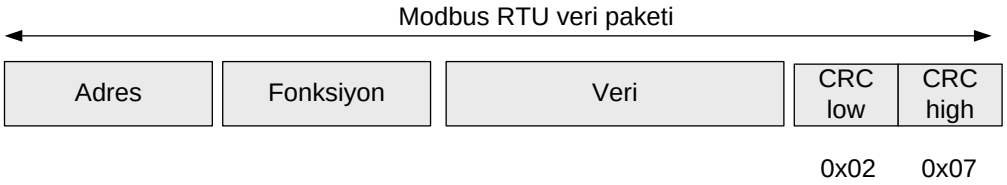
Döngüsel artıklık denetimi (Cyclical Redundancy Checking (CRC)) genellikle veri iletiminde kulanılan, veride meydana gelen hatalı değişimleri algılayan uygulaması kolay ve güvenliği güçlü bir hata bulma yöntemidir. RTU haberleşme modunda verinin doğru iletildiğini kontrol etmek için meajın son bölümünde hata kontrol alanı (error checking field) bulunur ve mesajda hata olup olmadığı Döngüsel Artıklık Denetimi (Cyclical Redundancy Checking (CRC)) yöntemi ile test edilir. Modbus veri birimi içersinde CRC iki byte yer kaplar ve 16 bitlik ikili değer tutar. CRC değeri mesajı gönderen cihaz tarafında oluşturularak mesajın son bölümüne eklenir ve mesajı alan cihaz CRC değerini mesaj verilerini alırken yeniden hesaplar ve hesapladığı değeri mesajda gelen CRC değeri ile karşılaştırır. Eğer iki değer birbirine eşit değilse hata mesajı üretir.

CRC ileri düzey bir hata düzeltme algoritmasıdır ve veri bitlerini polinom kodlar olarak ele alır. Modbus RTU, CRC 16 polinomunu kullanarak hata denetimi yapar. CRC 16 polinomu aşağıdaki gibidir.

CRC16 : $1+X_2+X_{15}+X_{16}$ = 1010 0000 0000 0001

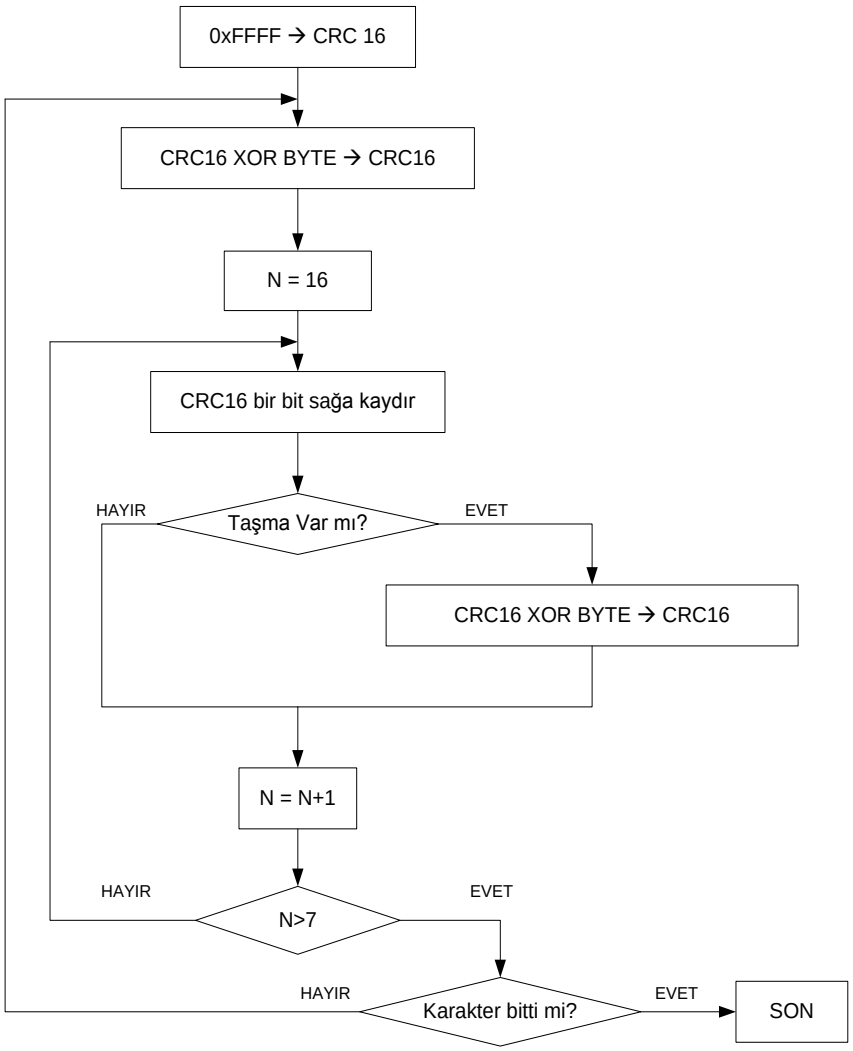
Modbus mesajının her karakterindedeki sekiz bitlik veri CRC değerini üretmek için kullanılır. Başlangıç biti, eşlik biti ve bitiş biti kulanılmaz. Algoirtma ‘1’ değerinden oluşan 16 bitlik (0xFFFF) değerin bir kaydediciye (CRC16 kaydedicisi) yüklenmesi ile başlar. Sonra Modbus mesajının iletilecek ilk sekiz bitlik veri değeri CRC16 kaydedicisindeki değerin düşük değerli byte bilgisi ile XOR işlemine sokulur. Elde edilen sonuç CRC16 kaydedicisine yüklenir ve CRC16 kaydedicisi en az anlamlı bit (least significant bit, LSB) yönünde bir bit kaydırılır. En anlamlı bitin yerine de 0 değeri eklenir. CRC16 kaydedicinin dışına çıkan en az anlamlı bit incelenir. Eğer LSB değeri ‘1’ ise CRC16 kaydedicisindeki ötelenmiş 16 bitlik bilgi CRC polinomu ile tekrar XOR işlemine sokulur, ‘0’ ise işleme sokulmaz. Bu işlem en başta ele alınan sekiz bit bitene kadar sürer ve bir sonraki sekiz bit yüklenerek CRC oluşturumaya devam edilir.

CRC değeri, Modbus RTU veri paketininin sonunda 16 bit (2 sekiz bitlik byte) olarak yer alır ve önce düşük sıralı byte (low order bit), sonra yüksek sıralı byte iletilir. Örneğin CRC değeri 0207 hex ise bu değer modbus veri paketinde Şekil 2.28'deki gibi iletilir;



Şekil 2.28 : Modbus RTU veri paketi byte iletim sırası.

CRC 16 algoritması Şekil 2.29'deki akış diyagramına göre çalışır.



Şekil 2.29 : CRC hata kontrol algoritması.

Örnek olarak verilen 0207 hex değeri kullanılarak hata kontrol algoritması Çizelge 2.31'de gibi oluşturulur;

Çizelge 2.31 : CRC hata kontrol algoritmasının gerçeklenmesi.

CRC16 Kaydedicisi	1111 1111 1111 1111
XOR ilk karakter	0000 0000 0000 0010
	1111 1111 1111 1101
Sağa kaydır 1	0111 1111 1111 1110 1
Flag = 1, XOR polinom	1010 0000 0000 0001
	1101 1111 1111 1111
Sağa kaydır 2	0110 1111 1111 1111 1
Flag = 1, XOR polinom	1010 0000 0000 0001
	1100 1111 1111 1110
Sağa kaydır 3	0110 0111 1111 1111 0
Sağa kaydır 4	0011 0011 1111 1111 1
Flag = 1, XOR polinom	1010 0000 0000 0001
	1001 0011 1111 1110
Sağa kaydır 5	0100 1001 1111 1111 0
Sağa kaydır 6	0010 0100 1111 1111 1
Flag = 1, XOR polinom	1010 0000 0000 0001
	1000 0100 1111 1110
Sağa kaydır 7	0100 0010 0111 1111 0
Sağa kaydır 8	0010 0001 0011 1111 1
Flag = 1, XOR polinom	1010 0000 0000 0001
	1000 0001 0011 1110
XOR ikinci karakter	0000 0000 0000 0111
	1000 0001 0011 1001
Sağa kaydır 1	0100 0000 1001 1100 1
Flag = 1, XOR polinom	1010 0000 0000 0001
	1110 0000 1001 1101
Sağa kaydır 2	0111 0000 0100 1110 1
Flag = 1, XOR polinom	1010 0000 0000 0001
	1101 0000 0100 1111
Sağa kaydır 3	0110 1000 0010 0111 1
Flag = 1, XOR polinom	1010 0000 0000 0001
	1100 1000 0010 0110
Sağa kaydır 4	0110 0100 0001 0011 0
Sağa kaydır 5	0011 0010 0000 1001 1
Flag = 1, XOR polinom	1010 0000 0000 0001
	1001 0010 0000 1000
Sağa kaydır 6	0100 1001 0000 0100 0
Sağa kaydır 7	0010 0100 1000 0010 0
Sağa kaydır 8	0001 0010 0100 0001 0
	MSB LSB

2.3 Modbus TCP Protokol Yapısı

Modbus TCP protokolü temelde ethernet üzerinden TCP arabirimi ile Modbus RTU protokolünün gerçekleştirilmesidir. Modbus mesajı, Ethernet TCP/IP ağı üzerine bağlı cihazlar arasında istemci/sunucu (client/server) haberleşmesi sağlar. Modbus RTU protokolü ile karşılaştırıldığında ana cihaz (master) TCP protokolünde istemci cihaza (client) ve bağımlı cihaz (slave) ise sunucu cihaza (server) karşılık gelir.

TCP, Transmission Control Protokol (İletim Kontrol Protokolü) ve IP, Internet Protokol (Internet Protokolü) terimlerinin kısaltılmasıdır. TCP'nin temel görevi bütün veri paketlerinin doğru bir şekilde alınmasını sağlamaktır. IP ise, mesajın doğru bir şekilde adreslenmesini ve yönlendirilmesini sağlar. TCP/IP protokolü sadece bir taşıma protokolüdür ve verinin ne anlama geldiğini ya da nasıl yorumlanacağı hakkında bilgi vermez. Özetle Modbus TCP, uyumlu cihazlar arasında Modbus mesaj verisini taşımak için TCP/IP ve Ethernet kullanır. Yani Modbus TCP protokolü, fiziksel bir ağı (Ethernet), ağ standartı (TCP/IP) ve Modbus protokol veri birimini birleştirir. Eski endüstriyel cihazlar genellikle RTU protokollünü kullandıkları için farklı bir cihazla beraber çalışması durumunda bağlantı problemleri yaşar. Modbus TCP protokolü bu duruma çözüm olarak geliştirilmiştir.

Modbus TCP protokolü OSI modelinde Şekil 2.30'deki gibi ifade edilir;

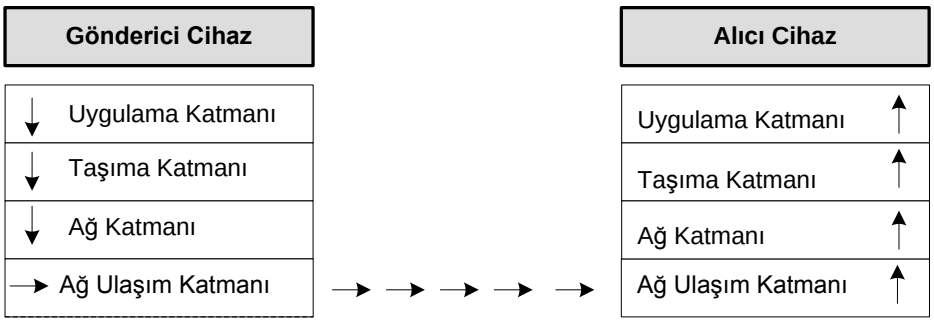
Katman	ISO/OSI Modeli	
7	Uygulama Katmanı(application layer)	Modbus Uygulama Protokolü
6	Sunum Katmanı (presentation layer)	
5	Oturum Katmanı (session layer)	
4	Taşıma Katmanı (transport layer)	Taşıma Katmanı (TCP)
3	Ağ Katmanı (network layer)	Ağ Katmanı (Internet Layer, IP)
2	Veri Bağlantı Katmanı (data link layer)	Ağ Ulaşım Katmanı (Network Access Layer)
1	Fiziksel Katman (physical layer)	

Şekil 2.30 : OSI modelinde Modbus TCP/IP yığını.

Modbus RTU protokolü üç katmanlı bir model sunmasına karşın, TCP protokolü dört katmanlı modelden oluşur. Her katmanda bir veya birden fazla protokol çalışır ve

Modbus uygulama katmanında oluşturulan protokol veri birimine her katmanda ek bilgiler eklenerek Modbus TCP mesajı oluşturulur. OSI referans modeli dikkate alındığında Modbus TCP protokolünde fiziksel katman ve veri bağlantı katmanı içiçe çalışır ve bu iki katman ağ ulaşım katmanı olarak tanımlanır. Ağ ulaşım katmanı dört katmanlı TCP/IP modelinin en alt katmanıdır ve verinin ağa fiziksel olarak nasıl gönderildiği tanımlar. Veriyi iletebilmek için Ethernet fiziksel katmanını kullanır. Veri bitleri ağa bağlı olan donanım cihazları tarafından elektriksel ya da optik sinyallere dönüştürülerek gönderilir. Bu katman veriyi Ethernet ağına genellikle RJ-45 bağlantı soketi ve CAT5 ya da CAT6 Ethernet kablosu aracılığı ile iletir. Bu katmanda ayrıca CSMA/CD (Carrier Sense Multiple Access / Collision Detection) ve MAC (Medium Access Protocol) protokolleri çalışır. Ağ katmanı ya da internet katmanı taşıma katmanının altında yer alır ve bu katman paketlerin ağa yönlendirilmesinden sorumludur. Modbus TCP protokolü bu katmanda IP protokolünü kullanır. Taşıma katmanı uygulama katmanının altında yer alır ve bu katmanda TCP protokolü çalışır. Uygulama katmanı protokolü, gönderilen ve alınan veriyi anlamlandıran bilgi katmanıdır ve bu katmanda standart Modbus protokol veri birimine ek birimler eklenerek Modbus TCP uygulama veri birimi oluşturulur.

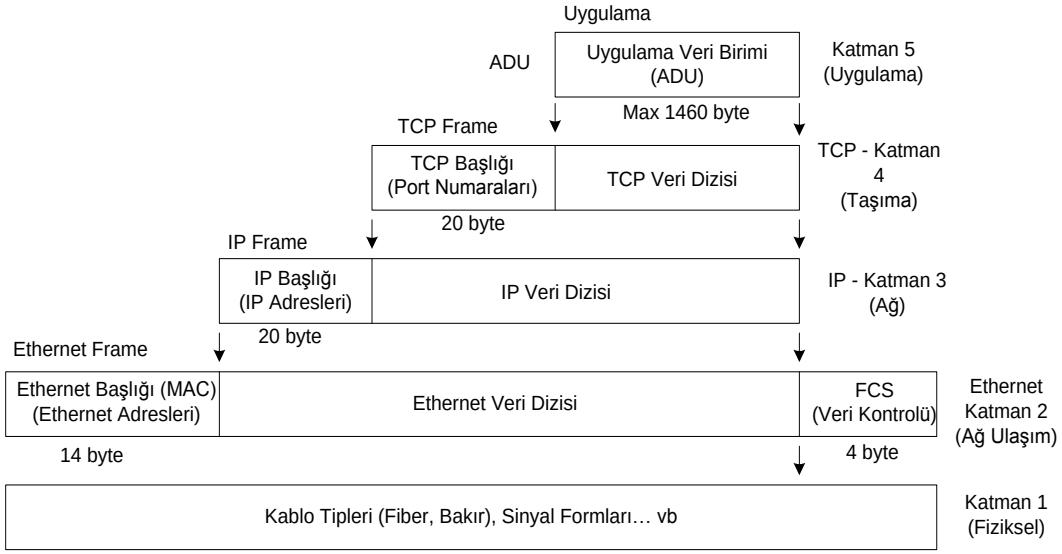
OSI modeli temel alındığında, iki cihazın haberleşebilmesi için aynı ağa bağlı olmaları ve aynı uygulama katmanı protokolünü kullanmaları gerekir. TCP/IP aslında internet haberleşmesi temel alan bir protokol takımını temel alır. Bu protokol takımı protokol yığını (protokol stack) olarak adlandırılır. Ağ üzerindeki her cihazın bu protokol yığınını çalıştırması gerekir. Modbus TCP/IP protokolünde istemci ve sunucu arasında verinin nasıl iletileceği Şekil 2.31'de gösterilmiştir;



Şekil 2.31 : Cihazlar arasında Modbus TCP veri akışı.

Modbus TCP protokolünün uygulama katmanı standart Modbus protokol veri birimine bazı ek birimler eklenerek oluşturulur ve Modbus uygulama veri birimi bir

alt katmandaki TCP veri dizisine gömülerek ağ içersinde ilerler. Uygulama veriyi ağa gönderdiği zaman, veri bütün katmanlardan geçer. Birbirini takip eden her katmanın belirlenmiş bir fonksiyonu vardır ve kendi protokolünü veri paketinin önüne başlık dosyası olarak ekler. Modbus TCP protokolü cihazlar arasında işlerken istemci cihaz talebini oluşturur ve ağ yapısındaki alt katmanlara gönderir. Bu katmanlar kendi protokol bigilerini veri paketinin başına veya bazen de sonuna eklerler. Sonuç olarak paket fiziksel katmana gelir ve buradan sunucu cihaza elektronik olarak iletilir. Mesaj sunucu cihaza ulaştıktan sonra fiziksel katmandan uygulama katmanına doğru iletilmeye başlar ve her katman kendi protokolünü mesajdan çözerek siler. Son olarak cihaz istenilen uygulamayı yerine getirir. TCP/IP ethernet data paketinin ağ içersinde ilerleyişi Şekil 2.32'de gösterilmiştir;

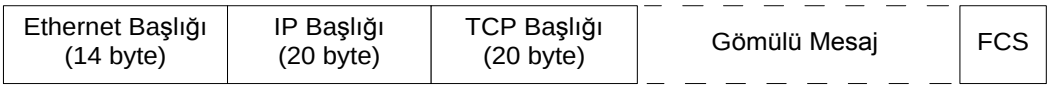


Şekil 2.32 : TCP/IP ethernet data paketinin çözülmesi.

Modbus mesajının en üst katmandan başlayarak her katmanda ilgili protokolün ek bilgileri eklenerek birinci katmana gelmesi olayına mesaj giydirme işlemi (mesaj encapsulation process) olarak adlandırılır. Protokol veri biriminde Modbus fonksiyon kodu ve veri alanı bulunur. Orijinal Modbus RTU mesajında bulunan hata kontrol alanı Modbus TCP mesajında yer almaz. Bunun yerine TCP/IP bağlantı katmanındaki veri kontrol yöntemlerine güvenilir. Modbus RTU'da bulunan adres alanı da Birim Tanımlayı'cının içersine gömülür. TCP/IP protokolünde cihaz adreslerine ihtiyaç duyulmaz, çünkü Ethernet cihazlarının kendisine özgü MAC (Media Access Control, Ortam Erişim Yönetimi) adresi bulunur.

Seri hat Modbus haberleşmede ana cihaz ağa bir talep gönderir ve ikinci talebi göndermeden önce ilk talebe cevap alana kadar bekler. Buna karşın Modbus TCP/IP haberleşmesinde istemci cihaz aynı sunucu cihaza bir önceki talebin cevabını beklemeden ard arda talep gönderebilir. Bu durumda MBAP bağlığı içersinde bulunan İşlem tanımlayıcı alanı talep/cevap eşleştirilmesi için kullanılır. Bu işlem basit bir şekilde TCP sıra numarasının bir bir sayıcı ile artırılarak yapılır. İstemcinin ard arda gönderebileceği talep, sayıcı cihazdan cihaza değişir, fakat genellikle bu değer bir ile onaltı arasında olur.

Modbus TCP/IP uygulama veri birimi, TCP veri çerçevesi içersine gömüldükten sonra TCP ile Modbus için özel ayrılmış olan 502 numaralı porta gönderilir ve ağa iletilmeden önce TCP/IP/MAC protokol giydirme işlemleri yapılır. Fiziksel katmanda veri iletilmeden önce Şekil 2.33'deki gibi olur.



Şekil 2.33 : Fiziksel katmanda Modbus TCP/IP protokolü.

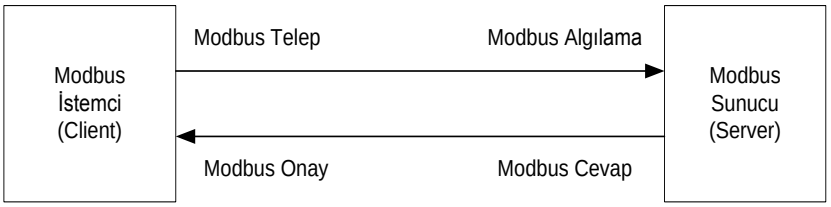
TCP, bağlantı yönelimli bir protokol olduğu için Modbus mesajı gönderilmeden önce TCP bağlantısının kurulması gerekir. Bağlantı kurulması işlemi istemci tarafından yapılır. TCP bağlantısı açık bir şekilde istemci kullanıcı uygulama yazılımı ile yapılabileceği gibi, otomatik olarak istemci TCP bağlantı yöneticisi tarafından da yapılabilir. Bağlantı işlemi için genellikle ikinci yöntem tercih edilir. Modbus TCP/IP bağlantıları iki cihaz arasında yapılan haberleşme bağlantıları olduğundan her yön için kaynak adresine, hedef adresine ve bağlantı kimliğine ihtiyaç duyar. Bu nedenle Modbus TCP/IP haberleşmesi sadece yayın modunda yapılır. Modbus uygulamarı için özel olarak 502 numaralı port ayrılmıştır. Modbus sunucusu haberleşme için bu portu dinler. İstemci sunucuya bir talep göndermek istediği zaman, sunucu cihazın 502 numaralı portu ile bağlantı kurar. Bağlantı kurulur kurulmaz veri bu bağlantı ile gönderilir. TCP bağlantısı istemci cihazlar tarafından gerçekleştirilir, sunucu cihazlar haberleşmeyi başlatmazlar. Her Modbus mesajı için sunucu ile bağlantıyı açıp kapamak yerine sürekli açık tutmak tavsiye edilir. Gerektiği zaman istemci bağlantıyı kapatabilir ve aynı zamanda sunucu da bağlantının kapatılması için istemciye mesaj gönderebilir. Bazı cihazlar hem istemci hem de sunucu gibi davranabilirler, bu durumda iki yönlü veri iletişimi, iki ayrı

bağlantı ile sağlanır. Modbus istemcisi aynı anda eş zamanlı TCP bağlantıları açabilir. Mesajı göndermek için yerel portlarını kullanır. Fakat sunucu cihaz veriyi 502 numaralı port üzerinden alır.

Modbus istemcisi ve sunucusu dört tip mesaj alış verişi gerçekleşir;

- **Modbus Talep (modbus request):** İstemci tarafından ağa gönderilen ve iletimi başlatan mesajdır.
- **Modbus Algılama (modbus indication):** Sunucu tarafından alınan talep mesajıdır.
- **Modbus Cevap (modbus response):** Sunucunun talebe karşılık olarak gönderdiği cevap mesajıdır.
- **Modbus Onay (modbus confirmation):** İstemci tarafından alınan cevap mesajıdır.

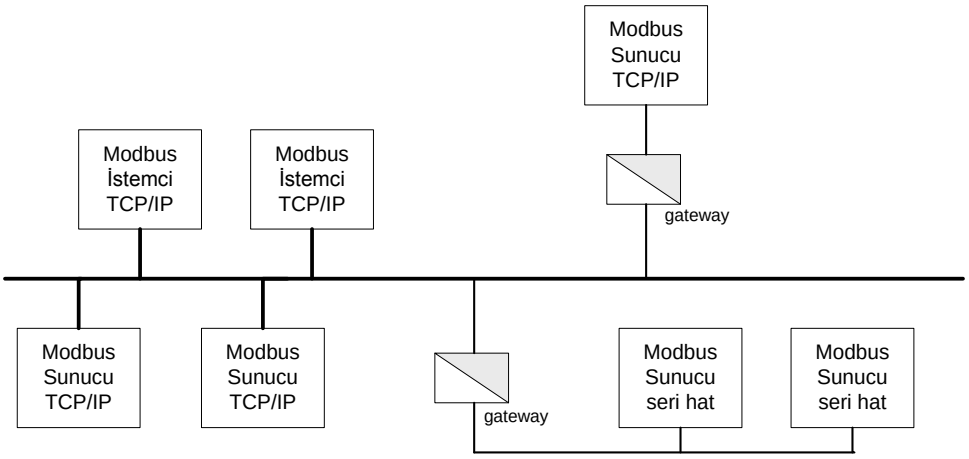
Şekil 2.34'de istemci ve sunucu arasında gerçekleşen veri alışverişi gösterilmiştir;



Şekil 2.34 : Modbus istemci/sunucu modeli.

Modbus TCP/IP üzerinden gerçekleştirilen haberleşme sistemi farklı türdeki cihazları içerebilir, fakat istemci ve sunucu cihazların TCP/IP ağına bağlı olması gerekir. Ayrıca bridge (köprü), router (dağıtıcı) ve gateway (ağ geçidi) cihazları kullanılarak seri hat üzerinden haberleşen cihazlar TCP/IP ağına bağlanabilir.

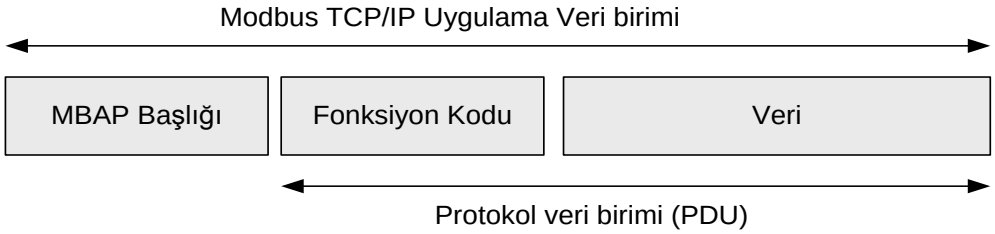
Tez kapsanında FPGA kullanılarak tasarlanan ağ geçidi cihazı Modbus TCP kullanarak TCP/IP ağı üzerinde haberleşen modbus cihazları ile Modbus RTU kullanarak seri haberleşme ağı üzerinde haberleşen cihazlar arasında veri alış verişini sağlamaktadır. Şekil 2.35'de Modbus TCP haberleşme mimarisine yer verilmiştir;



Şekil 2.35 : Modbus TCP/IP haberleşme mimarisi.

2.3.1 Modbus TCP uygulama katmanı

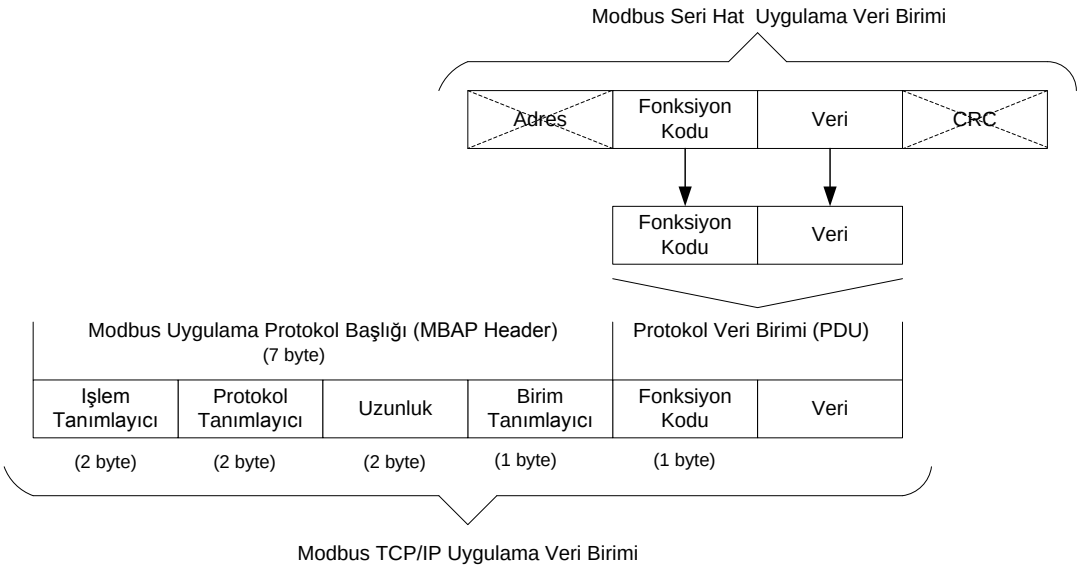
Bölüm 2.1'de anlatılan Modbus protokolünün alt katmanlardan bağımsız olarak tanımladığı protokol veri birimine Modbus TCP haberleşmesinde Şekil 2.36'de gösterilen MBAP ek birimi eklenmektedir.



Şekil 2.36 : Modbus TCP veri çerçevesi.

Seri hat üzerinden haberleşen standart Modbus RTU haberleşme protokolünde protokol veri birimine ek olarak bağlanılacak cihazı belirleyen adres birimi ve veri çerçevesinin kontrolünü yapan CRC hata kontrol bölümü bulunur. Bu na karşılık Ethernet üzerinden haberleşen Modbus TCP protokolünde bu birimler bulunmaz ve protokol veri birimine ek olarak 7 byte veri içeren MBAP başlık dosyası eklenerek Modbus TCP uygulama veri birimi oluşturulur.

Şekil 2.37'de standart Modbus RTU veri çerçevesinin Modbus TCP veri çerçevesine nasıl dönüştüğü gösterilmiştir.



Şekil 2.37 : Modbus RTU protokolünden Modbus TCP protokolüne geçiş.

Modbus TCP/IP haberleşmesinde istemci modbus uygulama veri birimini oluşturarak haberleşmeyi başlatır ve kullandığı fonksiyon kodu ile sunucuya ne tür bir işlem yapacağını bildirir. Fonksiyon kodları modbus protokolleri için standarttır ve değişmez. Modbus TCP uygulama veri birimini oluşturak için protokol veri birimine eklenen MBAP başlık birimi içersinde, işlem tanımlayıcı (transaction identifier), protokol tanımlayıcı (protokol identifier), uzunluk (length) ve birim tanımlayıcı (unit identifier) alt bölümleri bulunur. MBAP başlık birimini oluşturan alt birimler aşağıda anlatılmıştır;

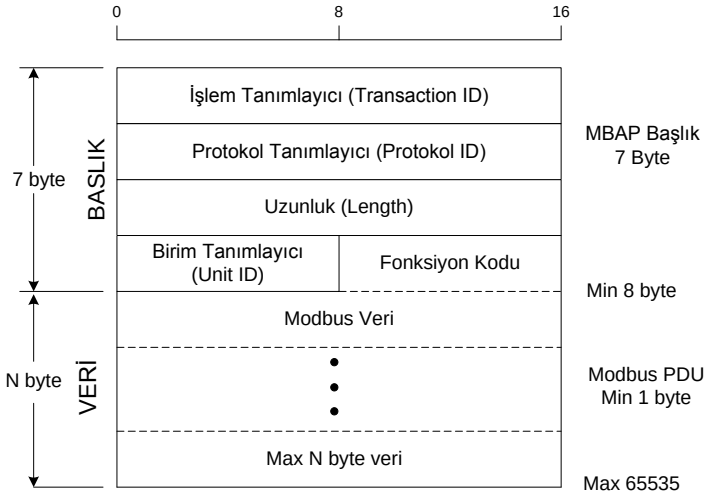
- **İşlem Tanımlayıcı (Transaction Identifier, 2 Btye) :** İstemci tarafından aynı TCP bağlantısı üzerinden bir önceki cevap beklenmeden birden fazla mesaj gönderildiği durumda bu tanımlama alanı işlem eşleştirmek için kullanılır. Sunucu bu bilgiyi istemcinin talebinden alarak cevaba kopyalar.

- **Protokol Tanımlayıcı (Protokol Identifier, 2 Btye):** Modbus protokolü için bu değer 0'dır. İstemci tarafından tanımlanır ve sunucu bu bilgiyi talep içerinden öğrenerek cevaba kopyalar.

- **Uzunluk (Lenght, 2 Btye):** Uzunluk alt birimi mesajın devamındaki (birim tanımlayıcıda dâhil) veri uzunluğunu ifade eder. İstemci, bu değeri göndereceği talep için tanımladığı gibi sunucu da cevabı için tanımlar.

- **<u>Birim Tanımlayıcı (Unit Identifier, 1 Btye):</u>** Modbus TCP ağı üzerindeki bir cihazla, Modbus seri hat üzerindeki bir bağımlı cihazı ağ geçidi kullanarak haberleştirmek için kullanılır. Modbus TCP sunucu uygulamasında bu değer 00 ya da FF şeklinde ayarlanır, sunucu tarafından önemsenmez ve cevaba aynı değer kopyalanır.

IEEE 802.3 Ethernet dünya çapında kabul görmüş, uzun süredir kullanılan ve büyük ölçüde alt yapısı kurulmuş bir ağ protokolüdür. Açık bir standart olduğu için bir çok üretici desteklemektedir. Dolayısıyla, bu standartın TCP/IP ile birleştirilmiş hali dünya çapında kullanılmakta ve hatta internet sunucuları ağının temelini oluşturmaktadır. Bir çok cihaz Ethernet'i desteklediği için endüstriyel uygulamalarda daha fazla tercih edilmeye başlamıştır. Modbus protokolü de Ethernet gibi ücretsiz, kolay ulaşılabilir ve bir çok endüstriyel cihaz üreticisi tarafından desteklenmektedir. Sonuç olarak Modbus uygulama protokolünü, IEEE 802.3 Ethernet protokolü ile birleştirilmiş ve Modbus TCP/IP endüstriyel haberleşme standartı oluşturulmuştur. Modbus TCP, geleneksel IEEE 802.3 Ethernet ile aynı fiziksel ve veri bağlantı katmanını ve ayrıca aynı TCP/IP protokolünü kullanır. Bu sebeple Modbus TCP protokolü, daha önceden kurulmuş olan Ethernet altyapısı, kablolar, bağlantı sağlayıcılar, ağ kartları, hublar ve anahtarlayıcılar gibi cihazlarla uyumlu çalışır. Modbus TCP uygulama veri birimi (ADU) Şekil 2.38'de gösterildiği gibi son şeklini alır.



Şekil 2.38 : Modbus TCP uygulama veri birimi.

2.3.2 Modbus TCP taşıma katmanı

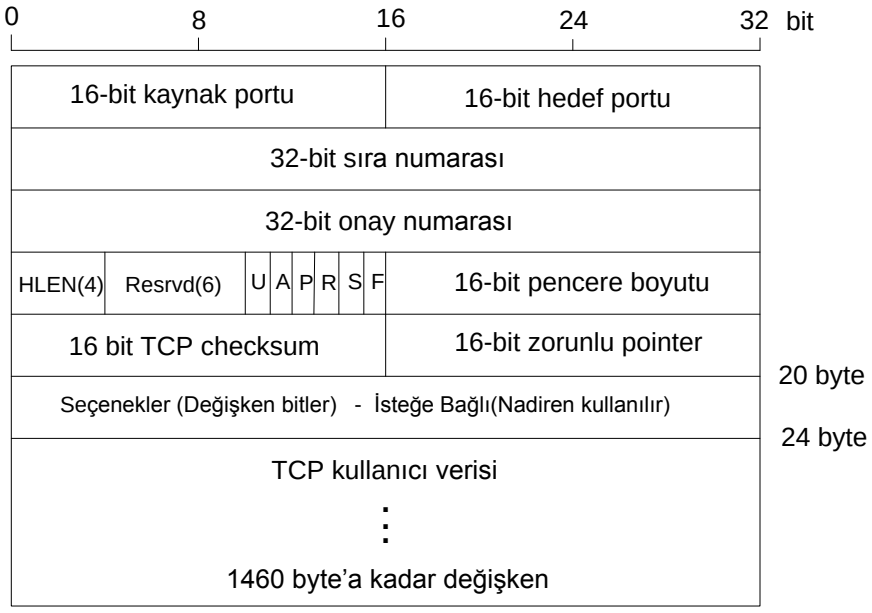
Taşıma katmanı, uygulama katmanının altında yer alır. Verinin taşınmasından, alınmasından ve hata kontrolünden sorumludur. Bu katmanda çalışabilen çok sayısa Taşıma Katmanı protokolü mevcuttur, Modbus TCP/IP bu katmanda TCP (Transport Control Protokol) protokolünü kullanır.

TCP, IP (Internet Protokol) protokolünün bir üst katmanında yer alır ve uygulama verinin taşınmasından ve güvenliğinden sorumludur. Buna karşılık olarak IP, verinin doğru bir şekilde adreslenmesinden ve teslimiden sorumludur. TCP paketi oluşturulduktan sonra IP paketinde ilgili yere yerleştirilir. IP tek başına güvenli bir protokol değildir, bağlantısız bir protokoldür ve TCP ile birlikte çalışması gerekir. Bu nedenle TCP genellikle IP platformunun üstünde çalışır ve TCP/IP beraber bütünleşik bir yapı oluşturur. Eğer veri alıcı cihaza ulaşmazsa yeniden gönderilmesi gerekir. Bu tür veri iletimi açık mesajlaşma (explicit messaging) olarak adlandırılır. TCP mesajı iletirken açık masajlaşma kullanır ve böylece verinin karşı tarafa iletildiğini garanti eder. TCP bağlantı yönelimli bir protokol olduğu için veri iletimi srasında iki ağ istasyonu arasında bağlantı kurar ve bağlantı kurulurken veri paketlerinin boyutu gibi özellikler belirlenir. TCP ayrıca isteci / sunucu haberleşme modelini destekler. Ağ üzerinde bağlantıyı başlatan ve kuran istasyon TCP istemci ve bağlantı kururan istasyonda TCP sunucu olarak adlandırılır. Modbus TCP prtotokolünde haberleşme herzaman bir ana cihaz (Modbus istemci) tarafından kontrol edilir. Sunucu haberleşmeyi başlatamaz ve istemcinin kendisi ile bağlantı kurmasını bekler.

TCP gönderilen veriyi, sağlama toplamı ile doğrular ve her gönderdiği paket için sıra numarası atar. TCP paketini alıcısı, alınan veri paketinin doğruluğunu yeniden kontrol eder ve gönderilen verideki değer ile karşılaştırır. TCP sunucusu paketi doğru bir şekilde alır almaz, önceden belirlenmiş bir algoritmayı kullanarak, gönderici TCP'nin atadığı sıra numarasından bir onay numarası üretir ve bu onay numarası istemci cihaza gönderilen cevaba yazılarak mesajın doğru olarak alındığı teyit edilir. Sunucu da ayrıca göderdiği pakete sıra numarası ekler. Bu işlem ağ üzerinde birden fazla paket gönderildiği zaman paketlerin kaybolmasını önler ve eğer paket kaybı olmuşsa tekrar aynı sırada gönderilmesini sağlar. TCP ayrıca veri paketini, uygulama katmanında kullanılan protokoller için özel olarak belirlenmiş port numaralarını

kullanrak hedef cihaza yönlendirir. Modbus protokolü için özel olarak ayırılan port numarası 502'dir. Port, tasıma katmanında kullanılan bir adres bilgisidir ve aynı ağ içinde paketin kaynak ve hedefini belirler. Port numaraları, tanımlanmış port numaraları (0-1023), kayıtlı kullanıcı port numaraları (1024- 49151) ve özel/dinamik port numaraları (49152-65535) olarak gruplandırılır. Portlar TCP/IP'ye farklı uygulama işlemine gitmesi gereken IP veri dizisini kodlamak ve çözmek için kullanılır.

TCP taşıma katmanı özetlenirse, gönderici alıcı cihazın veri paketini doğru bir şekilde almasını bekler. Paketin doğru bir şekilde alınmaması mesajın yeniden gönderilmesine ya da bağlantının kırılmasına neden olur. Gönderici gönderilen her pakete bir sıra numarası atar ve böylece alıcı cihaz gelen veri paketlerinde eksiklik olup olmadığını denetleyebilir. Eğer herhangi bir veri kayıpsa, devamında gelen tüm veriler buffer içersinde depolanır. Veriler tamamlandığında ve doğru sıra numaralarına göre sıralandığında uygulama katmanına gönderilir. Bu bağlantı yönelimli hata denetleme mekanizması TCP protokolünün diğer taşıma katmanı ptorokollerine göre daha yavaş çalışmasına neden olur, fakat verinin güvenli bir şekilde iletilmesini sağlar. Şekil 2.39'de TCP paket yapısının içeriği gösterilmiştir.



Şekil 2.39 : TCP paket yapısı.

Şekil 2.29'da görüldüğü gibi TCP paketi bir çok alt birimden oluşmaktadır. Bu alt birimler aşağıda kısaca açıklanmıştır.

- **Kaynak Portu** (Source Port, SP, 16 bit): Gönderici uygulamanın portunu tanımlar. (Gönderici portu, hedef cihazdangelecek cevap için bekleme durumuna geçer.)

- **Hedef Portu** (Destination Port, DP, 16 bit): Alıcı uygulamanın port numarasını tanımlar.

- **Sıra Numarası** (Sequence Number, SN, 32 bit): TCP gönderdiği her pakete bir sıra numarası atar ve alıcı taraf bu sıra numaralarını kullanarak paketleri geliş sırasına göre sıralar ve eksik paket olup olmadığını denetleyebilir. Veri paketindeki ik veri byte bilgisinin sıra numarasıdır.Eğer SYN bayrağı set edilirse, sıra numarası ilk sıra numarasıdır(n) ve ilk veri byter bilgisi n+1'dir.

- **Onay Numarası** (Acknowledgment Number, AN, 32 bit): Alıcı birim veriyi aldığını gönderici birime bu alt birimi kullanarak bildirir. Eğer ACK kontrol biti set edilirse, bu alan alıcının almayı beklediği bir sonraki paketin sıra numarasını içerir.

- **Başlık Uzunluğu** (Header Length): Veri paketinin başlık dosyasının zuznluğunu belirtir.

- **Rezerve** (Rezerved): İleride kullanmak için saklanmış olan alanı ifade eder. Bu yüzden değeri 0'dır.

- **UARPSF Bayrakları** (URG, ACK, PSH, RST, SYN, FIN): URG bayrağı acil gönderilecek öncelilki paketlerde 1 yapılır. ACK, paketin alaındığını karşı tarafa bildiren onay paketidir. PSH, alıcı bu paketi aldığında hemen uygulama katmanına çıkar. RST, iletişim sırasında bir sorun olduğu zaman iletişimi keserek yeniden başlatılmasını sağlayan pakettir. FIN, iletişimi sonlandıran pakettir.

- **Pencere Boyutu** (Window Size): Bir anda ne kadar veri bilgisi gönderileceğini kontrol etmek için kullanılır. Bu alt birimin amacı her paketin gönderilmesinden sonra alıcıya ulaşıp ulamadığı ile ilgili onay beklemek yerine paketeri onay beklemeksizin pencere boyutuna göre göndermektir. Yavaş ağlar kullanılarak yapılan haberleşmede ona beklemek habereşmeyi

daha da yavaşlatırken, hızı ağlar üzerinde yapılan haberleşme de sürekli paket gönderilmesi alıcının alabileceğinden daha fazla bir paket yığılmasına neden olarak haberleşme de problemler oluşturabilir. Dolayısyla her iki taraf o anda ne kadar bilgi alabileceğini pencere boyutunu kulanarak karşı tarafa bildirir.

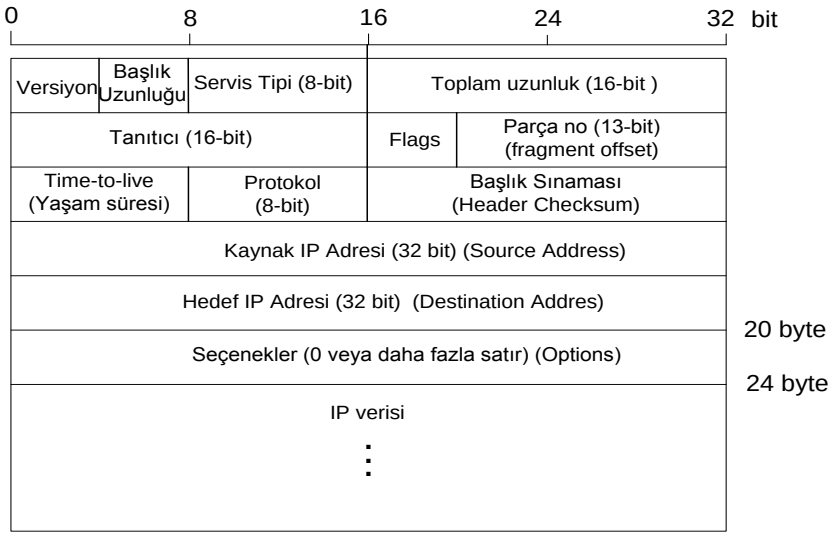
- **TCP Checksum** (TCPCS): Göderilen paketin hata denetimini yapan alt birimdir.

- **TCP Kullanıcı Verisi** (TCP User Data): Modbus uygulama katmanında oluşturulan Modbus TCP uygulama veri birimini içeren alt birimdir.

2.3.3 Modbus TCP ağ katmanı (network layer, internet layer)

Ağ katmanı ya da internet katmanı, taşıma katmanının altında yer alır ve paketleri ağa yönlendirmekten sorumludur. Bu katmanda çalışan bir çok ağ katmanı protokolü olmasına rağmen, bunlar arasında en fazla kullanılanı IP, ARP ve RARP'dir. Modbus TCP protokolü bu katmanda IP (Internet Protokol) protokolünü kullanır. IP katmanı kendisine gelen TCP segmenti içinde ne olduğu ile ilgilenmez. Sadece veri paketlerini adreslemeyi ve teslim etmeyi yönetir. Amacı paketi istenilen nokyata ulaştırmaktır ve bunun için kendi başlık bilgisini TCP katmanından gelen segmente ekler. TCP katmanından gelen segmente IP başlığının eklenmesi ile oluşturulan IP paket birimlerine datagram adı verilir. IP ağdaki iki cihaz arasında veri paketi göndermek için kaynağı belirlenmemiş biy yöntem kullanır, verinin doğruluğunu ve güvenliğini garanti etmez. Bu işi üst katmandaki protokollere bırakır. IP ayrıca, alt katmanların fiziksel uygulamalarını dikkate almaksızın sınırsız sayısa bireysel ağıdaha büyük bir ağ içersinde birleştirebilir. Yani veri, bir ağdan diğerine şeffaf bir şekilde aktarılabilir.

IP paketi, standart internet protokolünü kullanarak Ethernet üzerinden gönderilen bir veri yığınıdır. Her paket adresleme ve sistem kontrol bilgilerini içeren bir başlık birimi içerir. IP paketinin içeriği 32 bitlik satırlar halinde düşünülebilir. İlk beş satır her IP paketinin başlık bölümünde standart olarak kullanılırken (20 byte), altıncı satır özel seçenekler gerekli olduğu durumlarda kullanılır. Veri bölümünde ise taşıma katmanından gelen giydirilmiş veri bulunur. Şekil 2.40'da IP paketinin içeriği gösterilmiştir.



Şekil 2.40 : TCP paket yapısı.

Aşağıda IP paketinin içeriğinde bulunan alt birimler hakkında kısaca bilgi verilmiştir;

- **Versiyon (Version, sürüm):** 4 bitlik veri alanıdır ve kullanılan internet protokolün versiyonu burada belirtilir. (IP veriyon 4)

- **Başlık Uzunluğu (Header Length):** 4 bitlik veri alanıdır ve satır cinsinden IP başlık biriminin ne kadar yer kapladığını belirler. Örneğin, "0101" (5) değeri, ilk 5 satırın IP başlık birimi olduğunu ve altıncı satırdan itibaren IP veri bilgisinin başladığını belirtir.

- **Servis Tipi :** Paketin servis sınıfını belirtir.

- **Toplam Uzunluk :** 16 bitlik veri alanıdır ve başlık ve verinin toplam uzunluğunu byte cinsinden verir.

- **Tanıtıcı (Identification):** 16 bitlik veri alanıdır ve parçalanmış IP datagramlarının birleştirilmesine yardımcı olur.

- **Bayraklar (Flags):** DF (Don't Fragment) biti datagramın parçalanmaması gerektiğini belirtir. MF (More Fragment) biti arkadan aynı datagrama ait başka bir parça gelip gelmediğini gösterir.

- **Parça No (Fragment Offset):** 12 bitlik veri alanıdır ve ilgili parçanın bütündeki yerini gösterir.

- **Time-to-live (Yaşam Süresi):** 8 bitlik veri alanıdır. Değer her sekmede bir azaltılır ve sıfıra eşitlendiğinde hala varış yerine ulaşmadıysa paket iptal edilir.

- **Protokol :** 8 bitlik veri alanıdır ve hangi taşıma protokolünün kullanıldığını gösterir.

- **Başlık Sınaması (Header Checksum):** Başlık birimindeki hataları fark etmek için kullanılır. Hatalı bir başlığa sahip olan paket yok edilir.

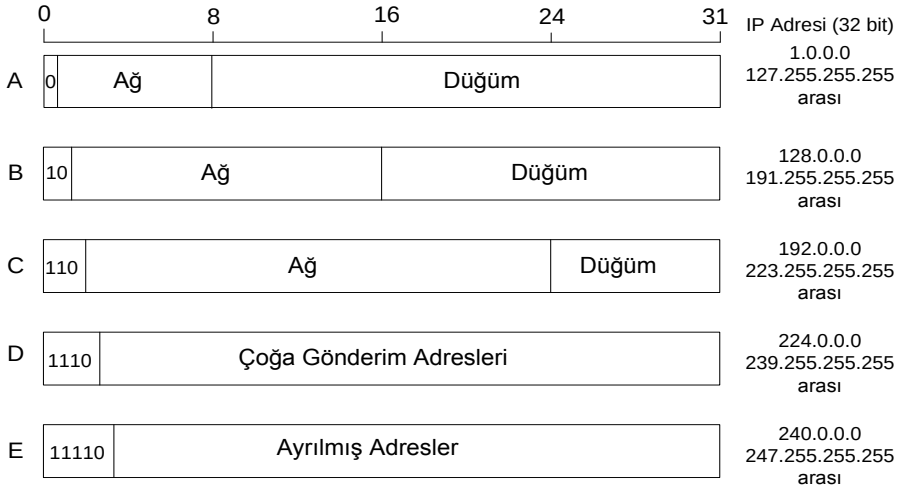
- **Kaynak IP Adresi (Source IP Adress):** 32 bitlik veri alanıdır ve IP paketini ağa gönderen ana cihazın IP adresidir.

- **Hedef IP Adresi (Destination IP Adres):** 32 bitlik veri alanıdır ve IP veri paketinin yönlendirildiği alıcı cihazın IP adresidir.

- **Seçenekler (Options):** Protokolün gelecek süsürmleri için bırakılan alanlardır.

- **IP Verisi (IP Dada):** Üst katmanlarda kullanıan protokollerin bildilerini içeren kısımdır.

IP adresi 32 bitten oluşur ,8 bitlik gruplardan oluşur ve ağ üzerideki her cihaz için tektir. IP adresleri Şekil 2.41'de gösterildiği gibi 5 snıfa ayrılır.



Şekil 2.41 : IP adres formatı.

Her adres sınıfı özel bir bit dizisi ile başlar. A, B ve C sınıfı adresler ağ (önek, prefix) ve düğüm(sonek, suffix) olmak üzere iki bölümden oluşur. Son eki en uzun adres A sınıfı olduğu için, en fazla düğüm A sınıfına ait bir ağda adreslenebilir. D sınıfı adresler çoğagönderim gruplar için kullanılırken, E sınıfı adresler daha sonra kullanılmak üzere ayrılmışlardır. Adreslerin dağıtılması ICANN (*Internet Corporation for Assigned Names and Numbers*) tarafından organize edilir ve her ülke de ICANN'ye bağlı çalışan yerel birimler bulunur.

Aynı ağın içersindeki tüm birimler aynı ağ adresine (önek) sahiptir. Bir yönlendirici cihaz (router) kendisine gelen paketin ağ adresine bakarak o ağa hangi port üzerinden ulaşılabileceğini bulur ve paketi uygun şekilde yönlendirir. Bir IP adresinden ağ adresini çekebilmek için maske kullanılır. Bir IP adresi kendi ağının maskesi ile bit bazında mantıksal VE işlemine girdiğinde sonuç olarak ağ adrsi ortaya çıkar. Ağ adresleri de paketlerin yönlendirilmesinde kullanılır. Adres sınıflarına göre maskeler aşağıdaki gibidir.

A sınıfı ağ maskesi: 255.0.0.0
B snıfı ağ maskesi: 255.255.0.0
C sınıfı ağ maskesi: 255.255.255.0

Ayrıca ağ katmanında TCP/IP'nin bir fonksiyonu olan ve IP ile beraber çalışan ARP (Address Resolution Protocol) protokolü, Ethernet adreslerini (MAC adres) IP adresleri ile eşler ve ağ cihazının içersinde bir eşleme tablosu oluşturur. Başka bir deyişle bu protokol IP adresinden fiziksel adrese dönüşüm işlemini gerçekleştirir.

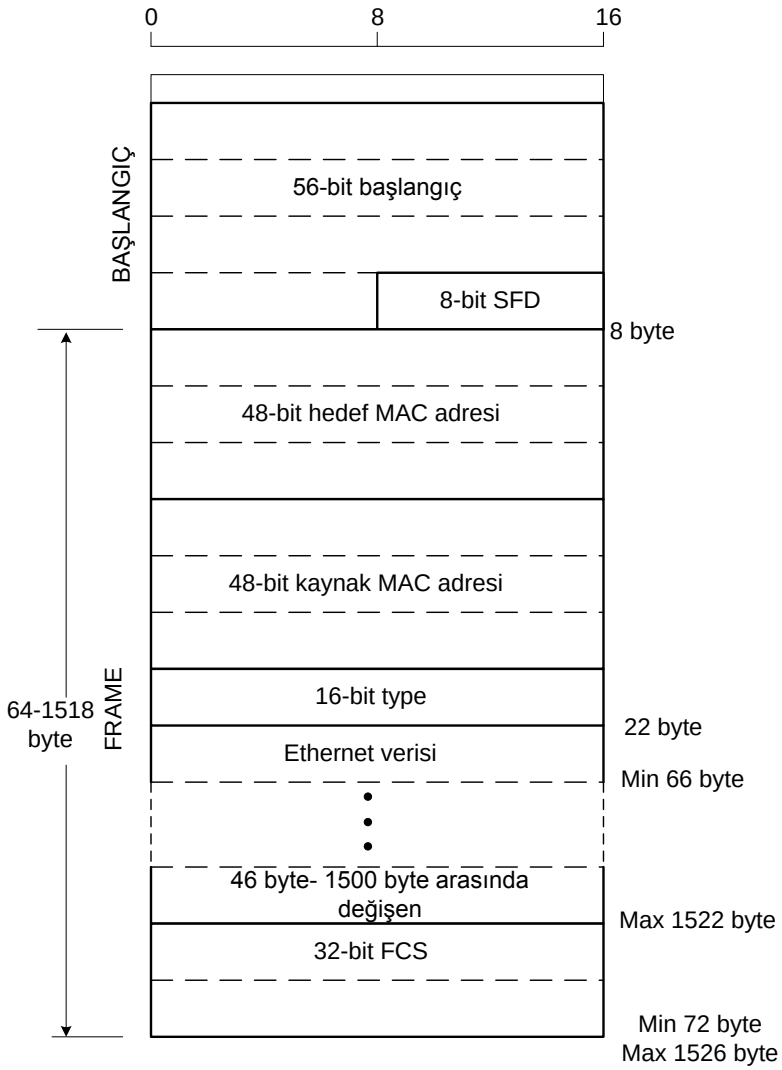
2.3.4 Modbus TCP ağ ulaşım katmanı

Modbus TPC ağ ulaşım katmanı, veri bağlantı katmanı ve fiziksel katmanın birleştirilmiş hali olarak düşünülebilir. IP paketlerini fiziksel ağa göndermekle sorumludur. Başka bir değişle bu katman TCP/IP protokol yığını ve fiziksel ağ arasında bir arabirim oluşturur.

Veri bağlantı katmanında paket giydirme işleminin son aşaması gerçekleştirilir. Bu katmanda CSMA/CD (Carrier Sense Multiple Access/Collision Detection) protokolü ve MAC (Medium Access Protocol) protokolü çalışır.

Ağda bulunan cihazlar ağa aynı anda veri göndermek istediği zaman ağ üzerinde çakışmalar meydana gelir. CSMA/CD bu çakışmaları engellemek için ağ üzerinde iletişimi kontrol eden protokoldür. Bu protokol şu şekilde çalışır; Ağa bilgi göndermek isteyen cihaz öncelikle ağı kontrol eder. Ağ boş ise göndermek istediği verileri ağa gönderir. Ağ boş değilse bekleyip gönderme işlemi tekrar edilir. İşleme başlamanadn önce maksimum deneme sayısının geçip geçilmediği kontol edilir. Eğer geçilmişse veri gönderme işlemi iptal edilerek üst katmanlar bilgilendirilir. CSMA/CD protokolü aslında switch bulunmayan ve tek kablo üzerinde birden fazla Ethernet cihazının bulunduğu sistemler için geliştirilmiştir. Günümüzde kullanılan endüstriel ağlarda switch kullanıldığı için full-dublex bağlantılar için çakışma söz konusu olmaz. Fakat half-dublex hatlarda cihaz veri gönderirken veri alma durumu oluşlabileceği için hat üzerinde çakışma meydana gelebilir ve CSMA/CD protokolü kullanılır.

MAC protokolü, veri bağlatı katmanını oluşturmak için gerekli olan servisleri sağlar. Veriye öncelikle protokol kontrol bilgilerini içeren bir başlık dosyası giydirilir. Daha alıcı ve vericinin MAC adresleri ve sonra veriye hata kontrol kodu (CRC) eklenerek veri paketlenir ve fiziksel katmana aktarır. Ethernet Adresi ya da MAC (Media Access Control) adresi, ağ bağlantı cihazını donanımsal olarak tanımlar. 48-bit boyutunda cihaza özel bir adrestir. CRC kodu, veri paketlerinin iletimi sırasında meydana gelebilecek bozulmaların tespiti için kullanılmaktadır. Gönderici cihaz veri üzerinde CRC algoritmasını kullanarak 32 bitlik CRC kodunu oluşturur ve veriye ekler. Alıcı cihaz da veriyi alırken kendi CRC kodunu hasaplar ve alınan verideki kod ile karşılaştırır. Eğer iki kod aynı ise veri doğru bir şekilde iletilmiş olur, eşit değilse paket iptal edilir ve paketin yeniden gönderilmesi için talepte bulunur. Ethernet verisi fiziksel katmanda paketler halinde parçalara ayrılarak ağa göderilmektedir. Pakatlerin gönderilme sırası solda sağa doğru en az anlamlı bitten en anlamlı bite doğru gönderilir. Ağ ulaşım katmanında oluşturulan ve fiziksel ağa gönderilen Ethernet paketin içeriği şekil 2.42'de gösterilmiştir;



Şekil 2.42 : Ethernet paket yapısı.

Ethernet paket yapısının içeriğinde bulunan alt birimler hakkında aşağıda kısaca bilgi verilmiştir.

- **Başlangıç & SFD (56 bit & 8 bit SFD):** MAC tarafından otomatik olarak eklenmektedir ve istasyonlar arasında sinyalleri senkronize etmek için kullanılır. Başlangıç bilgisi yedi adet 0x55 ve SFD değeri de 0xD5 değerinden oluşmaktadır. Alıcıbu alt birimleri paketten oromatik olarak çıkarmaktadır.

- **Hedef MAC adresi (48 bit):** 6 bitlik adres alanıdır ve hedef birimin Ethernet adresi (MAC) adresi bu bölümde tanımlanır. En az anlamlı biti adresin unicast(0) ya da multicast(1) olduğunu belirtir.

- **Kaynak MAC adresi (48 bit):** 6 bitlik veri alanıdır ve veriyi gönderen birimin Ethernet (MAC) adresi bu bölümde tanımlanır.

- **Tip (16 bit):** 2 byte veri alanıdır. Hangi ağ katmanı protokolünün kulanıldığı bu bölümde tanımlanır.

- **Veri:** Ağa gönderilecek veri bu alt birimde bulunur.

- **FCF (CRC, Cyclic Redundancy Check, 32 bit):** Veri paketinin sonuna eklenen ve verinin doğruluğunu kontrol eden alt birimdir.

Tez çalışmasında tasarlanan FPGA ağ geçidi cihazın Modbus TCP taşıma, ağ ve ağ ulaşım katmanları Wiznet modülü tarafından gerçekleştirildiği için yukarıda verilen bilgiler yeterli görülmüştür.

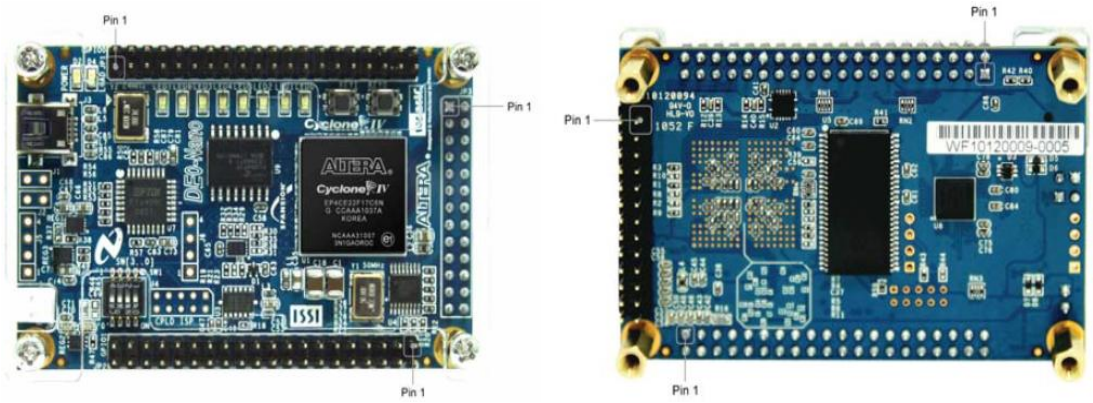
3. SİSTEM BİLEŞENLERİ

FPGA tabanlı Modbus ağ geçidi cihazın tasarımı ve test edilmesi için oluşturulan sistemde kullanılan temel elemanlar bu bölümde anlatılmıştır. Bu elemanlar, FPGA, Wiznet TCP Bağlantı Modülü, Modbus Giriş/Çıkış modülleri, Switch, PLC ve RS485 dönüştürücü devresidir. Bu bölümde sistemi oluşturan temel bileşenler tanıtılmıştır.

3.1 FPGA ve Nios İşlemcisi

FPGA, Türkçe "Sahada Programlanabilir Kapı Dizileri" anlamına gelen "Field Programmable Gate Array" ifadesinin kısaltmasıdır. FPGA, programlanabilir mantık blokları ve bloklar arasındaki ara bağlantılardan oluşan, geniş uygulama alanlarına sahip bir çeşit sayısal tümleşik devredir. Başka bir tanım da, üretimden sonra istenilen fonksiyona göre donanım yapısı kullanıcı tarafından değiştirilebilen bütünleşik devre şeklinde yapılabilir. FPGA 1984 yılında Xilinx şirketi kurucularından Roos Freeman ve ekibi tarafından tasarlanmış ve kullanılmıştır. FPGA, dijital tasarımlarda kullanılan farklı lojik kapıların milyonlarcasını tek bir entegre devre üzerinde barındırır ve bu lojik kapılar istenildiği gibi programlanabilir, böylece istenilen herhangi bir dijital tasarım bu elemanlar üzerinde gerçeklenebilir [23].

Tez kapsamında FPGA üzerinde, Modbus TCP ve Modbus RTU haberleşme ağları arasında veri dönüşümü ve alış verişini gerçekleştirecek amacıyla Modbus ağ geçidi (gateway) cihazı tasarlanmıştır. Tasarım Altera firmasının üretmiş olduğu DE0-Nano Geliştirme ve Eğitim Kartı üzerinde gerçekleştirilmiştir. DE0-Nano Geliştirme ve Eğitim Kartı, üzerinde Altera FPGA bulunması ve küçük yer kaplaması (7,5 cm - 5 cm) nedeniyle bu çalışma için tercih edilmiştir. Kartın ön ve arka yüzeyinin görüntüsü Şekil 3.1'de gösterilmiştir.



Şekil 3.1 : DE0 Nano FPGA Geliştirme ve eğitim Kartı.

Çizelge 3.1'de bu çalışma için önemli olan DE0-Nano kartının temel bileşenleri listeleniştir. En önemli temel bileşenler, FPGA, EPCS, SDRAM (Synchronous Dynamic Random Access Memory) ve JTAG (Joint Test Action Group) UART birimleridir. EPSC, konfigürasyon verilerini saklayan bir çeşit flash hafıza cihazıdır. Sisteme güç verildiği zaman konfigürasyon dosyasını FPGA'e yüklemekten sorumludur. JTAG UART ise sistemi konfigürasyonu ve hata denetiminden sorumludur [24].

Çizelge 3.1 : De0-Nano kartının temel bileşenleri.

Bileşen	**Özellik**
FPGA	Altera Cyclone 4 EP4CE22F16C6N FPGA
Konfigürasyon	USB Blaster ve EPCS
Hafıza	32 MB SDRAM ve 2KB I2C EEPROM
Saat Sinyali	Harici 50 Mhz Osilator
Güç	USB ya da 2 Pin Harici Güç Girişi
Çevresel Birimler	2x40 Pin Expansion Header (72 GPIO Pin)
	8 LED, 2 Pushbutton, 4 Dipswitch
	8 Kanal 12 bit ADC (200 ksps)
	3-eksen İvmeölçer

Program geliştirme aracı olarak Altera firmasının Quartus II programlama platformu tercih edilmiştir. Quartus II, FPGA ve CPLD (Complex Programmable Logic Device) cihazları için tasarım, sentez ve analiz programıdır. İçerisinde, sistem geliştirme, fonksiyonel ve zamanlama simülasyonu, güç analizi, zamanlama analizi ve doğrulama gibi birçok alt araçları bulunur. Ayrıca, blok/şematik tasarım ve VHDL ve Verilog yazılım dillerini kullanarak RTL (Register Transfer Level) tasarım

yapmayı da destekler. Bu temel fonksiyonlara ek olarak Quartus II içerisinde gömülü yazılım paketleri bulunur. Tez kapsamında yapılan çalışmada bu gömülü yazılım paketlerinden Qsys Sistem Entegrasyon Aracı (Qsys System Integration Tool) kullanılmıştır.

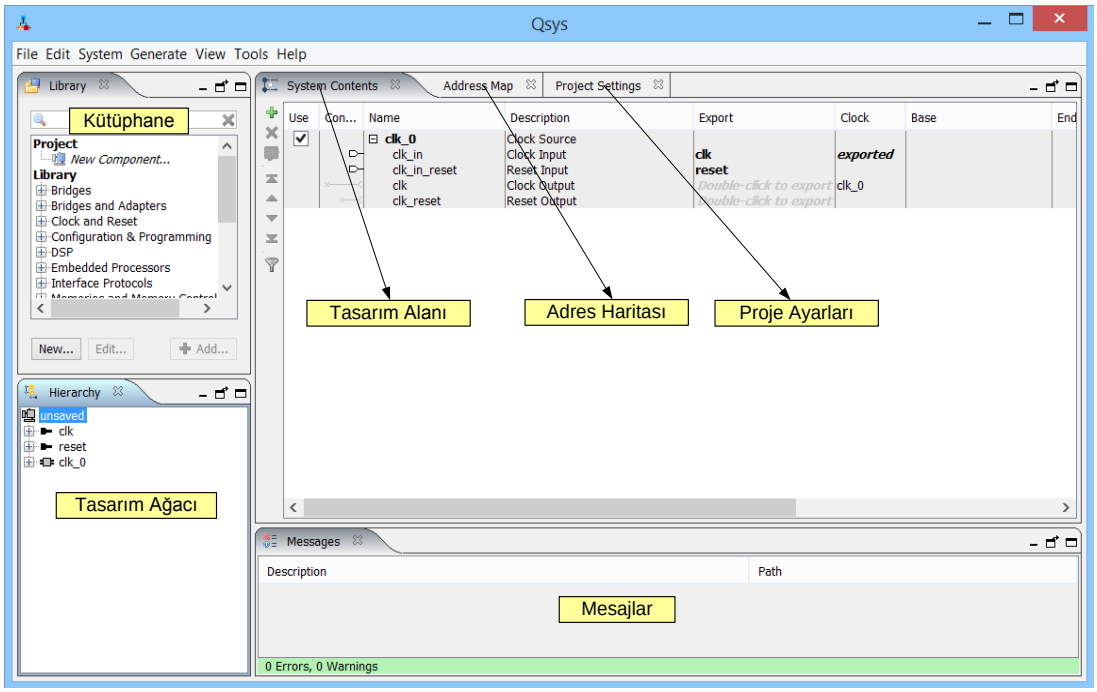
Gömülü sistem tasarımı donanım (hardware) ve yazılım (software) olarak iki ana bölüme ayrılır. Donanım bölümü, donanım paketleri, donanım tasarlama dilleri yada lojik kapılar kullanılarak tasarlanır. Yazılım bölümü ise C, C++ gibi yüksek seviye programlama dilleri kullanılarak belirlenen sistem çalışması için gerekli olan adımların sıralanması ile gerçekleştirilir. Çoğu gömülü sistem teknolojilerinde donanım sabittir ve donanım/yazılım bölümlemesi yapmaya gerek yoktur. FPGA kullanılarak tasarlanılan sistemlerde hangi modüllerin donanımsal, hangilerinin yazılımsal olacağı çok net değildir. Bu ayrım tasarımcıya ve uygulamaya göre değişebilir. FPGA kullanılarak gerçekleştirilen donanım tasarlama işlemi, VHDL ve Verilog gibi yüksek seviye donanım tasarlama dilleri kullanılarak, kütüphaneler kullanılarak ya da kapı seviyesinde tasarımlar yapılarak gerçekleştirilebilir. Genellikle tasarım ortamı tasarıma entegre edilebilen "intellectual property" adı verilen alt bölümlerden oluşur. Konfigüre edilebilir bölgeler veya parçalar geliştirilir ve tasarım içersine entegre edilebilir. Tasarım sürecinin yazılım bölümü makine dili veya seçilmiş bir yüksek seviyeli dil kullanılıp program geliştirilerek oluşur.

Sisteminin tasarımında FPGA'in hem donanım tasarlama özelliğinden hem de gömülü işlemci oluşturabilme özelliğinden faydalanılmıştır. Donanımsal tasarımlar için VHDL dili kullanılmıştır. Donanımsal tasarım olarak FPGA'in Wiznet modülü ile haberleşmesini sağlamak için 32 bitlik bir *SPI modülü* ve DE0 Nano üzerinde bulunan EEPROM ile haberleşebilmesi içim *I2C modülü* tasarlanmıştır. Sisteme işlemci ekleme işlemi Qsys Sistem Entegrasyon Aracı kullanılarak gerçekleştirmiştir. İşlemci içersine gömülen program için C dili kullanılmış ve Quartus II platformu içersinde bulunan Eclipse için Nios II Yazılım Geliştirme Aracı (Nios II Software Build Tool for Eclipse) kullanılmıştır.

Aşağıdaki bölümde Qsys tasarım alanı ve gömülü sistemde kullanılan temel bileşenler tanıtılmıştır.

Qsys, Grafiksel Kullanıcı Arabirimi (GUI, Graphical User Interface) kullanılarak, FPGA tabanlı donanımsal sistemin bileşenlerini oluşturmaya izin veren bir sistem geliştirme aracıdır. Donanımsal birimlere örnek olarak, hafıza birimi, işlemci, seri birimler (USP, UART, SPI), genel amaçlı giriş/çıkış birimleri (GPIO), PLL, zamanlayıcı (timer) vb. verilebilir. Tasarım Qsys kullanılarak gerçekleştirildikten sonra gerekli ara bağlantılar yapılır. Qsys tasarımda kullanılan her bir donanım modülü için HDL (Hardware Description Language) dosyaları oluşturur. Bu dosyalar Altera FPGA için yapılandırma (konfigürasyon) dosyalarını oluşturur.

Qsys tasarım alanına, Quartus II programı içerisinde *Araçlar (Tools) / Qsys* yolu kullanılarak ulaşılır. Qsys ilk açıldığı zaman karşımıza çıkan pencere Şekil 3.2'de gösterilmiştir. Kütüphane bölümünden tasarımda kullanılan bileşenler seçilerek tasarım alanına taşınır ve gerekli bağlantıları yapılır. Tasarım ağacında kullanılan bileşenler listelenir. Tasarım sırasında gerekli uyarılar mesajlar bölümünden kullanıcıya bildirilir. Adres haritası bölümünde bileşenlerin adresleri yer alır ve proje ayarları bölümünden kullanılan FPGA seçilir. Bileşenler seçilip bağlantıları tamamlandıktan sonra *Generate* sekmesi kullanılarak bileşenlerin HDL kodu üretilir.



Şekil 3.2 : Qsys tasarım alanı.

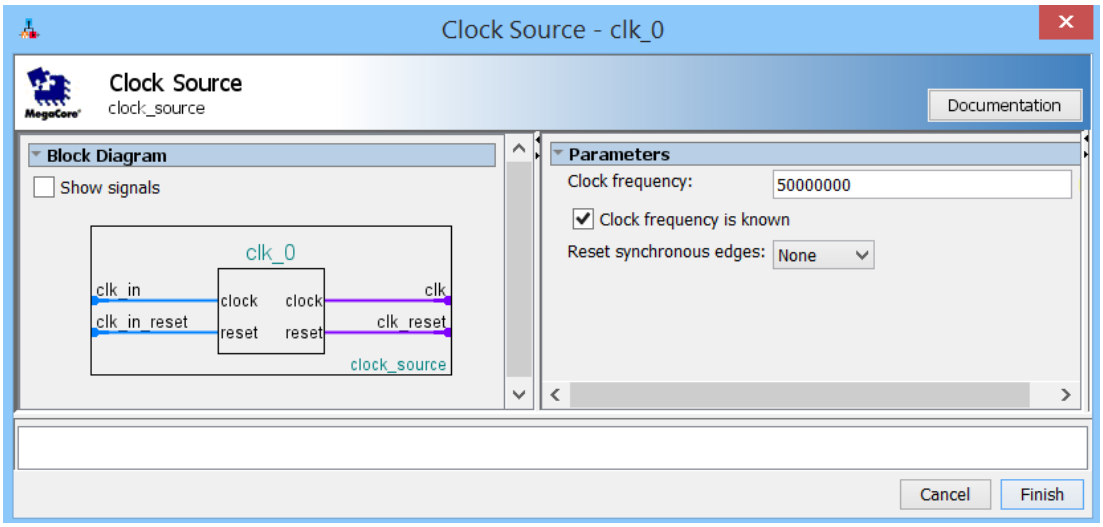
Qsys kullanılarak tasarlanan sistem içersinde, bir mikroişlemcide bulunan temel bileşenlerin hepsi yer alır. Sistemde bulunması istenilen bileşen kütüphaneden seçilir,

uygulamaya göre konfigüre edilir ve sisteme eklenir. Bu açıdan bakıldığı zaman FPGA içersindeki program uygulamaya yönelik olarak yeniden oluşturulabilir. Temel bir Qsys sisteminde, işlemci, saat sinyali ve hafıza birimi bulunmalıdır. Bunlara ek olarak sisteme çeşitli çevresel birimler eklenebilir.

Aşağıda Qsys ile gömülü sistem tasarımında kullanılan temel bileşenler tanıtılmıştır;

- **Saat Sinyali (clock):**

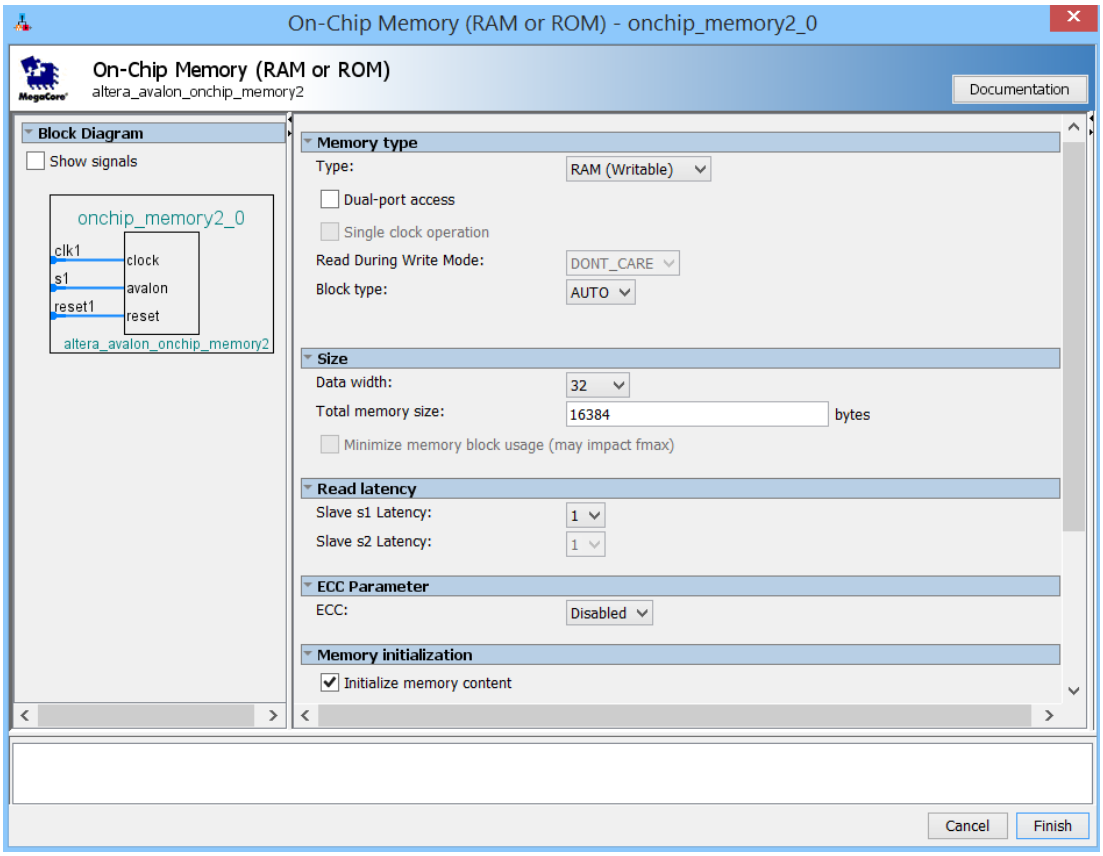
Saat sinyali kaynağı DE0-Nano kart üzerinde bulunan 50 MHz kristaldir. Bu sinyal özel olarak ayrılmış FPGA pini ile doğrudan FPGA'in içersine alınır ve yazılımsal olarak Qsys sistemine bağlanır. Qsys açıdığı zaman saat sinyali sisteme eklenmiş olur ve gerekiyorsa ilgili ayarlamalar Şekil 3.3'de gösterilen ekran üzerinden yapılır.



Şekil 3.3 : Saat sinyali ayarlanması.

- **Hafıza (on-chip memory)**

İşlemci sistemimin verileri ve talimatlari (instructions) için bellek alanına ihtiyacı vardır. Kütüphane içersinden on-chip memory (RAM or ROM) seçilerek tasarım alanına eklenir ve tasarım için gerekli ayarlamalar tasarım alanına getirilen On-chip Memory bileşeni çift tıklanarak Şekil 3.4'de gösterilen pencere açılarak gerekli ayarlamalar yapılır.



Şekil 3.4 : Hafıza biriminin ayarlanması.

Bu birim ile ilgili daha fazla bilgi Şekil 3.4'te gösterilen Doküman (Documentation) seçeneği tıklanarak elde edilebilir. Tasarıma özel olarak hafıza biriminin ismi tasarım alanında değiştirilebilir.

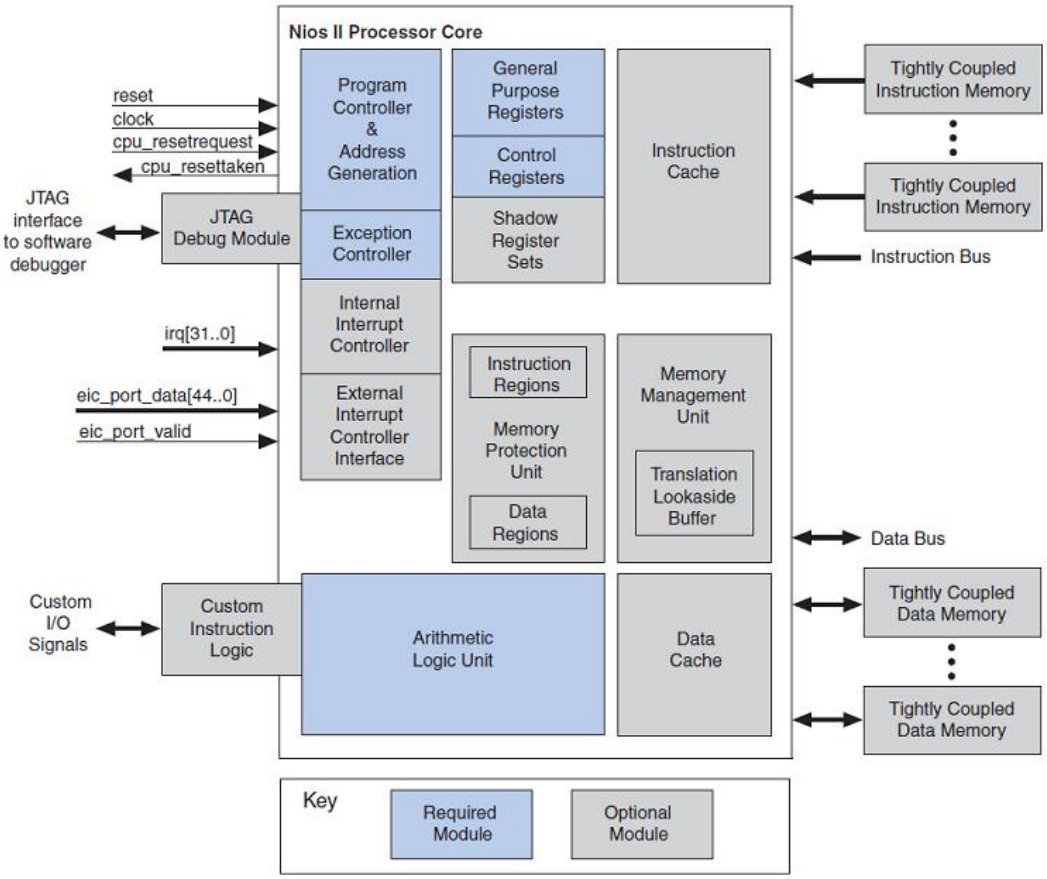
- **Nios II İşlemcisi (Nios II Processor)**

Altera firmasının ürettiği FPGA'ler için tasarlamış olduğu yazılımsal işlemci Nios II'dir. 32 - bitlik genel amaçlı bir işlemcidir. Nios II işlemcisi, sabit yapılı ve satışa hazır mikro kontrolcülerden farklı olarak, konfigüre edilebilir yazılımsal bir IP (intellectual property) çekirdektir. Sistem performansı ve fiyat hedeflerini karşılayacak şekilde uygulamaya göre çeşitli yapılar eklenebilir ya da çıkartılabilir. Yazılımsal terimi, işlemcinin bir silikon yapı içersinde sabit olmayıp, herhangi bir FPGA cihazı için oluşturulabileceğini ifade eder. Her tasarım için Nios II işlemcisi oluşturmaya gerek olmayıp, daha önceden hazırlanan işlemci farklı uygulamalarda kullanılabilir. Ek birimler sisteme kolayca eklenip çıkarılabilir. Ayrıca Nios II işlemcisini diğer sabit işlemcilerden ayıran en önemli özelliği esnek bir çevresel birim setine sahip olmasıdır. İşlemci programlanabilir lojik içersinde gerçeklendiği

için hedeflenen uygulama için tam bir çevresel birim yapısı oluşturulabilir. Esnek çevresel birimler sayesinde esnek bir adres haritasına sahiptir. Nios II, RISC (Reduced Instruction Set Computer) işlemci mimarisine sahip bir işlemcidir ve aşağıdaki özelliklere sahiptir;

- Sabit 32 bit komut seti (Fixed 32 bit instruction format)
- 32 bit veri yolu (32 bit data path)
- 32 bit adres uzayı (32 bit address space)
- Bellek eşlemeli giriş/çıkış uzayı (Memory-mapped I/O space)
- 32 adet kesme kaynağı (32-level interrupt request)
- 32 genel amaçlı kaydediciler (32 genel amaçlı kaydediciler)

Nios II işlemci çekirdeğinin blok diyagramı Şekil 3.5'de verilmiş ve devamında şekilde gösterilen ana bloklar listelenmiştir.



Şekil 3.5 : Nios II işlemcisi blok diyagramı.

- Genel amaçlı kaydediciler, kontrol kaydedicileri ve gölge kaydedicileri
- ALU (Aritmetik ve lojik birimler)

- Harici ve dâhili kesme birimleri (Exception and interrupt birimleri)
- İsteğe bağlı komut önbelleği ve veri önbelleği
- İsteğe bağlı hafıza yönetim birimi (MMU, memory management unit)
- İsteğe bağlı hafıza kestirim birimi (MPU, memory protection unit)
- İsteğe bağlı JTUG modülü

Altera firmasının Nios II işlemcisi için üç farklı versiyon sunar;

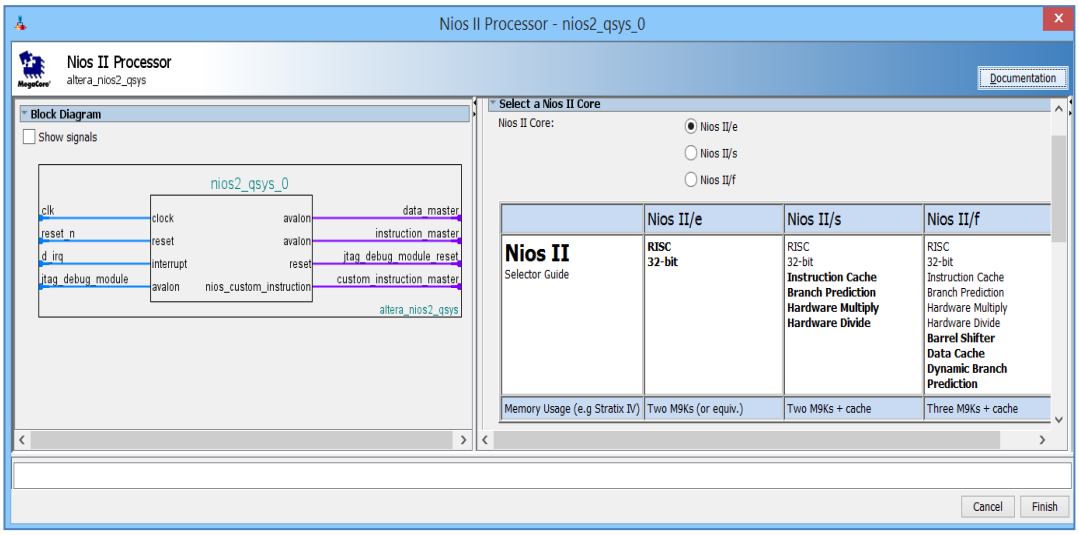
- **Nios II/f:** Optimal performans için tasarlanan hızlı çalışan çekirdektir. 6 kademeli iletişim hattı, komut önbelleği, veri önbelleği ve dinamik dallanma kestirimi özelliklerini sunar.

- **Nios II/s:** İyi performans sağlamak için tasarlanan küçük boyutlu işlemcidir. 5 kademeli iletişim hattı, komut önbelleği, veri önbelleği ve dinamik dallanma kestirimi özelliklerini sunar.

- **Nios II/e:** Optimal boyutta tasarlanan ücretsiz sunulan çekirdektir. İletişim hattı, komut önbelleği ve veri önbelleği özellikleri bulunmaz. Ücretsiz olduğu için tez çalışmasında bu işlemci kullanılmış ve sonuçlar yeterli görülmüştür.

Her üç işlemci versiyonu performansı ve boyutu farklı olmasına rağmen aynı komut setini paylaşırlar. Bu nedenle Qsys aracında aynı ara yüzle gösterilir. İşlemci HDL yazılım dili ile tanımlandıktan sonra ve yazılım kodları içersinde işlemcinin içyapısının değiştirilmesine izin verilmez. İşlemci program içersinde belirlenen işlemleri gerçekleştiren bir kara kutu şeklinde davranır. Nios II işlemcisinde otuzi ki tane 32 bitlik genel amaçlı kaydedici bulunur. Aritmetik lojik birim, genel amaçlı kaydediciler içerisindeki veriler üzerinde aritmetik işlemleri (toplama, çıkarma, çarpma ve bölme), lojik işlemleri (and, or, nor, xor) ve kaydırma işlemlerini gerçekleştirir [25].

Kütüphane içersinden Nios II Processor seçilerek tasarım alanına taşınır. Tasarım alanında Nios II Processor bileşeni çift tıklanarak Şekil 3.6'da gösterilen konfigürasyon penceresi açılır ve buradan Nios II-e seçilir.



Şekil 3.6 : İşlemci biriminin ayarlanması.

- **Paralel Giriş / Çıkış Birimleri (Paralel Input/Output, PIO)**

Paralel giriş/çıkış birimleri Nios II işlemci sisteminin giriş sinyallerini okumasını ve çıkış sinyallerini sürülmesini sağlar. PIO, genel amaçlı giriş/çıkış portları ve Nios II işlemcisi Avalon ara bağlantıları arasında bellek eşlemeli bir ara yüz sağlar. Karmaşık sistem uygulamalarında sisteme çok sayıda PIO eklenebilir. Kütüphane içersinden PIO (Parallel I/O) seçilerek tasarım alanına eklenir ve uygulamaya göre isimleri değiştirilebilir. PIO birimi sisteme eklendiği zaman aşağıda ifade edilen dört alanın doldurulması gerekir [25].

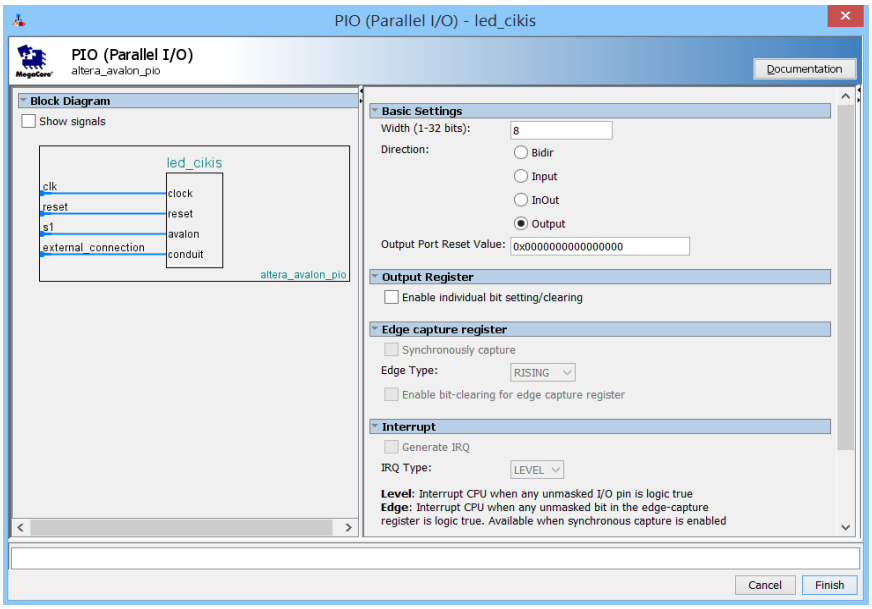
- Basic Settings: Bu alanda portun uzunluğunu ifade eden bit sayısı maksimum 32 bit olacak şekilde ayarlanır. Portun yönelimi yani giriş (input), çıkış (output), hem giriş hem çıkış (inout) ya da çift yönlü (bidirectional) belirlenir. Çıkış portunun reset adresi bulunur.

- Output Register: Bu alan port çıkış portu olarak ayarlandığı zaman her bite ayrı ayrı ulaşmak istenildiği zaman işaretlenir.

- Edge Capture Register: Bu alan verinin yükselen ya da düşen kenarda alınması ile ilgili ayaları içerir.

- Interrupt: Bu alan port kesme işleminde kullanıldığı zaman gerekli olan ayarlamaları içerir.

PIO bileşeninin konfigürasyon penceresi Şekil 3.7'de gösterilmiştir.



Şekil 3.7 : Giriş / Çıkış biriminn ayarlanması.

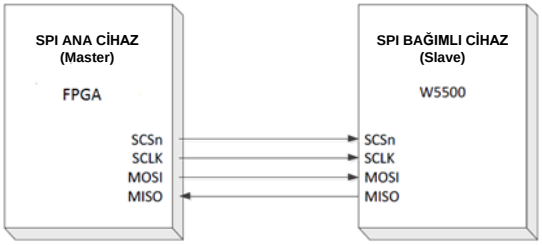
3.2 Wiznet TCP Bağlantı Modülü

Wiznet 5500 çipi, gömülü sistemlere internet bağlantısı sağlayan bir çeşit donanımsal TCP/IP Ethernet kontrolcüsüdür. W5500, tek bir çip içersinde TCP /IP yığını, MAC Ethernet ve fiziksel katmanı barındırır ve kullanıcılara sistemlerine internet bağlantısı kullanmasını sağlar. TCP, UDP, IPv4, ICMP ve ARP gibi bir çok protokolü destekler. Ethernet paketlerini işleyebilmek için içersinde 32 Kb hafızası vardır. Kullanıcalara aynı anda sekiz bağımsız soket açma imkanı tanır. Şekil 3.8'de Wiznet modülünün genel görüntüsü verilmiştir [26].

Şekil 3.8 : Wiznet TCP Modulü.

Wiznet TCP/IP Ethernet modülü gömülü sistemlerle haberleşmek için SPI (Serial Peripheral Interface) bağlantısı kullanır. Qsys kullanılarak tasarlanan işlemci ile bu birimi haberleştirebilmek için FPGA içersinde VHDL donanım programlama dili kullanılarak *SPI modülü* tasarlanmıştır. SPI full dublex modda çalışan senkron bir

veri bağlantısı standartıdır. Ana cihaz ve bağımlı cihaz (master/slave) haberleşmesini destekler. Veri alışverişini ana cihaz başlatır. Wiznet modülü SPI veri arabirimini kullanarak gömülü sisteme dört sinyalle bağlanır ve bağımlı cihaz (slave) modunda çalışır. Bu modül SPI protokolü ile çalışırken, değişken uzunlukta veri (variable fixed data) ve sabit uzunlukta veri (fixed length data) olmak üzere iki farklı çalışma modunu destekler. Şekil 3.9'da Wiznet ile FPGA'in bağlantısı gösterilmiştir. İki cihaz birbirine SCSn, SCLK, MOSI ve MISO olmak üzere dört sinyal ile bağlanır.



Şekil 3.9 : Wiznet ile FPGA'in bağlantısı.

Bu sinyallerle ilgili açıklama Çizelge 3.2'de verilmiştir;

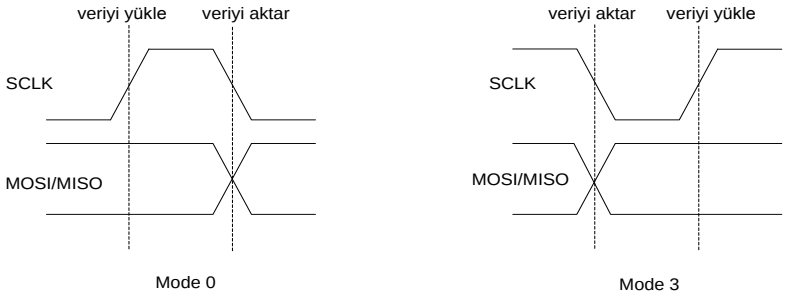
Çizelge 3.2 : W5500 sinyalleri açıklaması.

Sinyal	**Açıklama**
SCSn	Slave Chip Select (Bağımlı Cihaz Seçimi)
SCLK	Slave Clock (Bağımlı Cihaz Tetikleme Sinyali)
MOSI	Master Output Slave Input
MISO	Master Input Slave Output

SCSn sinyali, SPI ağı üzerinde birden fazla bağımlı cihaz varsa ana cihazhın hangisi ile haberleşeceğini belirlemek için kullanılır. Fakat SPI ağı sadece birtek cihaza bağlanacaksa SCSn sinyali doğrudan toprağa bağlanabilir. Tez çalışmasında Wiznet modülü değişken uzunlukta veri modunda çalıştırılmış ve SCSn sinyalı FPGA tarafından kontrol edilmiştir. SCLK sinyali senkron veri akışını sağlamak için FPGA'in Wiznet modülüne sağladığı saat sinyali pinidir. MOSI (Master Output-Slave Input) pini FPGA'den Wiznet'e veri aktarma için ve MISO (Master Input-Slave Output) pini de Wiznet'den FPGA'e veri aktarmak için kullanılır.

SPI protokolü dört farklı operasyon modunu destekler (Mode 0, 1, 2, 3). Bu çalışma modları SCLK sinyalinin polarite ve fazına göre belirlenir W5500 modülü sadece Mode 0 ve Mode 3'ü desteklemektedir. Mode 0 ve Mode 3'ün tek farkı SCLK

sinyalinin aktif olmayan durumda polaritelerinin farklı olmasıdır. Aktif olmayan durumda Mode 0'da SCLK düşük değerde (low, 0), Mode 3'de yüksek değerde (high, 1) olur. Bu iki modda veri SCLK'nın yükselen kenarında yüklenir ve düşen kenarında iletilir. Her iki modda MOSI ve MISO sinyalleri veri en önemli bitten en an önemli bite doğru iletir. Mode 0 ve Mode 3 operasyon modları kullanılarak veri aktarımı Şekil 3.10'daki gibi olur [26].

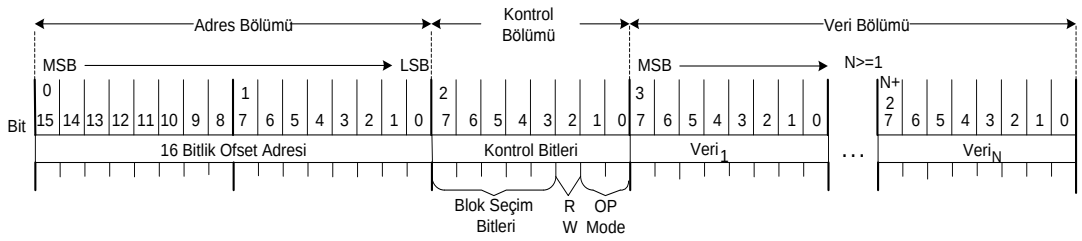


Şekil 3.10 : Mode 0 ve Mode 3 veri iletim modları.

Değişken uzunukta veri operasyon modunda SCSn sinyali FPGA (ana cihaz) tarafından kontrol edilir ve sinyal lojik 1 seviyesinden lojik 0 seviyesine düşünce Wiznet SPI üzerinden veri haberleşmesini başlatır ve sinyal lojik 0 seviyesindden lojik 1 seviyesine gelince veri aktasımı sonlandırılır.

Wiznet SPI veri çerçevesi, adres bölümü, kontrol bölümü ve veri bölümü olmak üzere üç alt bölümden oluşur. Adres bölümü Wiznet kaydedicileri veya Tx/Rx hafıza birimleri için ofset adreslerini belirler. Kontrol bölümü, ofset adresinnin ait olduğu bloğu, okuma ya da yazma modunu ve SPI operasyon modunu (değişken/sabit uzunlukta veri) belirler. Veri bölümü ise uygulamaya göre 1 byte, 2 byte, 4 byta ya da N byte ($N \geq 1$) olarak belirlenebilir.

Şekil 3.11'da Wiznet modülü veri paketi içeriği gösterilmiştir.



Şekil 3.11 : W5500 veri paketi.

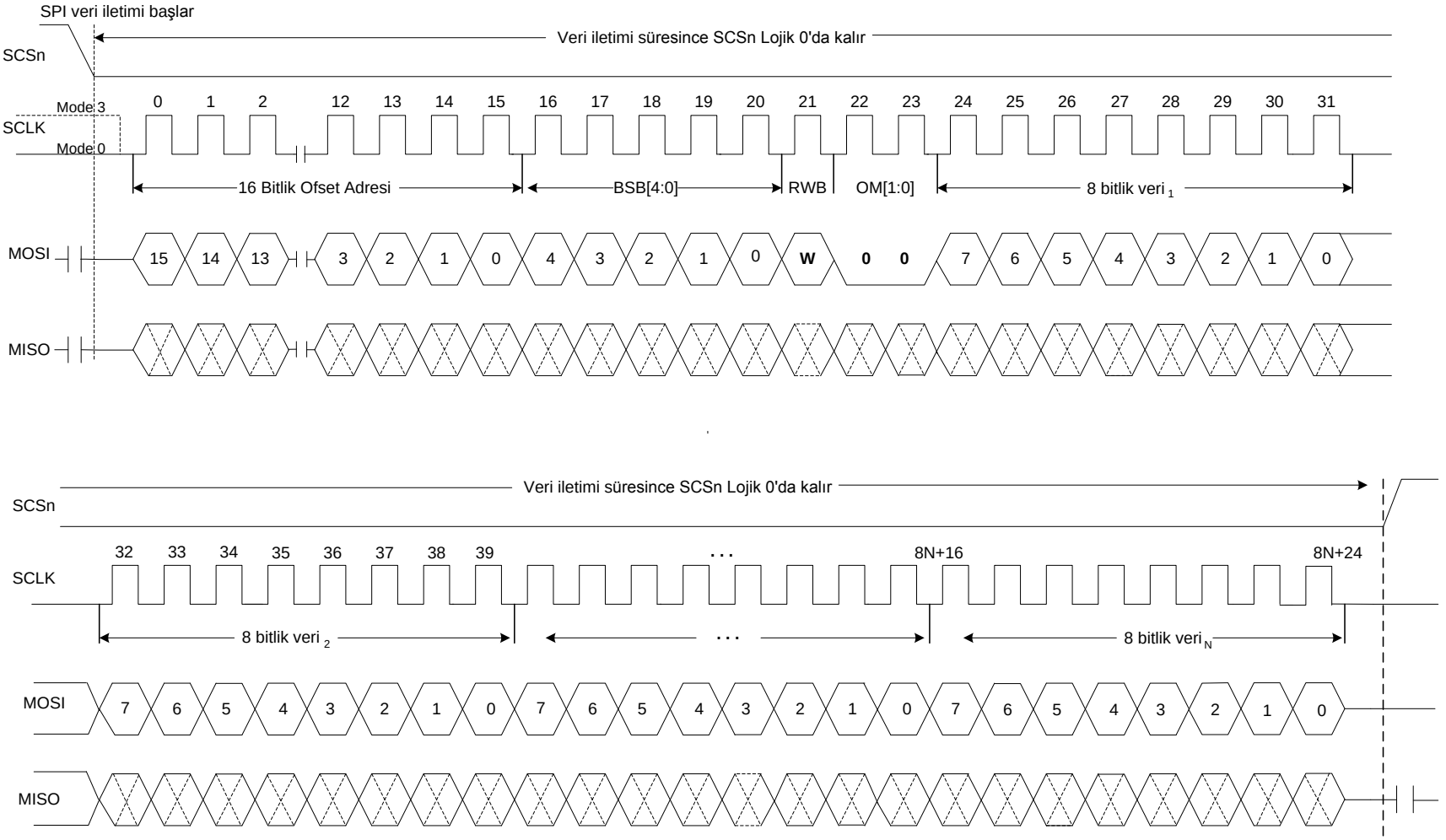
Adres bölümünde değerler en önemli bitten en az önemli bite doğru aktarılır. Kontrol bölümünde bulunan 5 bitlik blok seçim bitleri ofset adresinin hangi bloğa ait olduğunu belirler. W5500 bir adet genel kaydedici (common register), 8 soket kaydedicisi (soket register) ve her soket için Tx/Rx blokları bulunur. Okuma/Yazma modu için bir bit ayrılmıştır ve '0' değeri okuma modunu, '1' değeri yazma modunu aktif eder. Operasyon modu için iki bit ayrılmıştır. Değişken veri boyutu ve Sabit veri boyutu modu için Çizelge 3.3'de dört farklı operasyon modu tanımlanmıştır [26].

Çizelge 3.3 : W5500 operasyon modları.

OP Modu	Açıklama
00	Değişken Uzunlukta Veri Modu, N-byte veri (N≥1)
01	Sabit Uzunlukta Veri Modu, 1 byte veri
10	Uzunlukta Veri Modu, 2 byte veri
11	Uzunlukta Veri Modu, 4 byte veri

Tez çalışması kapsamında Wiznet modülü değişken uzunlukta veri modunda kullanılmıştır ve aşağıda bu modda veri okuma ve yazma işlemleri anlatılmıştır. Bu durumda kontrol bölümünde operasyon modu bitleri '00' olarak atanmıştır.

Değişken uzunluktaki veri modunda veri yazma ve okuma işlemi aşağıdaki gibi gerçekleştirilir. Bu modda, Wiznet modülü, FPGA tarafından SCSn sinyali kontrol edilerek veri yazma ve okuma işlemi gerçekleştirilir. Veri yazma işlemi için RW (read/write) biti '0', okuma için ‘1’ yapılır ve operasyon modu '00' olarak ayarlanır. W5500 çipi veri yazma işlemi SCSn sinyalinin lojik 1 seviyesinden lojik 0 seviyesine düşmesi ile başlar ve yazma işlemi bittikten sonra SCSn sinyali FPGA tarafından lojik 1'e çekilerek işlem tamamlanır. Şekil 3.12’de veri yazma işlemi için SPI veri çerçevesinin paket içeriğinin nasıl değiştirği gösterilmiştir.



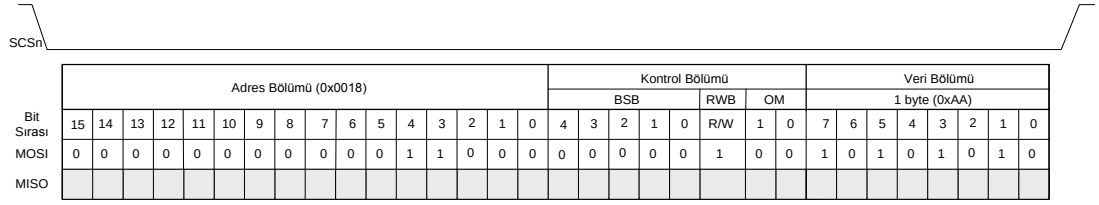
Şekil 3.12 : W5500 modülü için veri yazma paketi.

Wiznet modülüne bir byte yazma işlemi aşağıdaki gibi gerçekleştirilir.

Çizelge 3.4 : W5500 için bir bitlik veri.

Ofset Adresi	0x0018	Adres bilgisi
BSB [4:0]	'00000'	Blok seçim bilgisi
RWB	'1'	Veri yazma
OM	'00'	Operasyon modu
Veri	0xAA	Veri

Bu işlem sonucunda Wiznet veri çerçevesi Şekil 3.13'deki gibi olur;



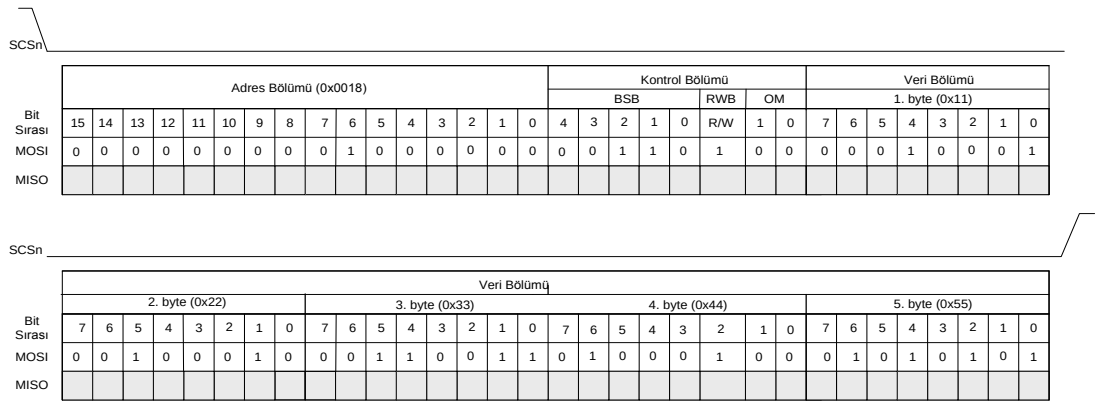
Şekil 3.13 : W5500 molülü bir bitlik veri paketi.

Wiznet modülüne N byte yazma işlemi aşağıdaki gibi gerçekleştirilir.

Çizelge 3.5 : W5500 için N bitlik veri.

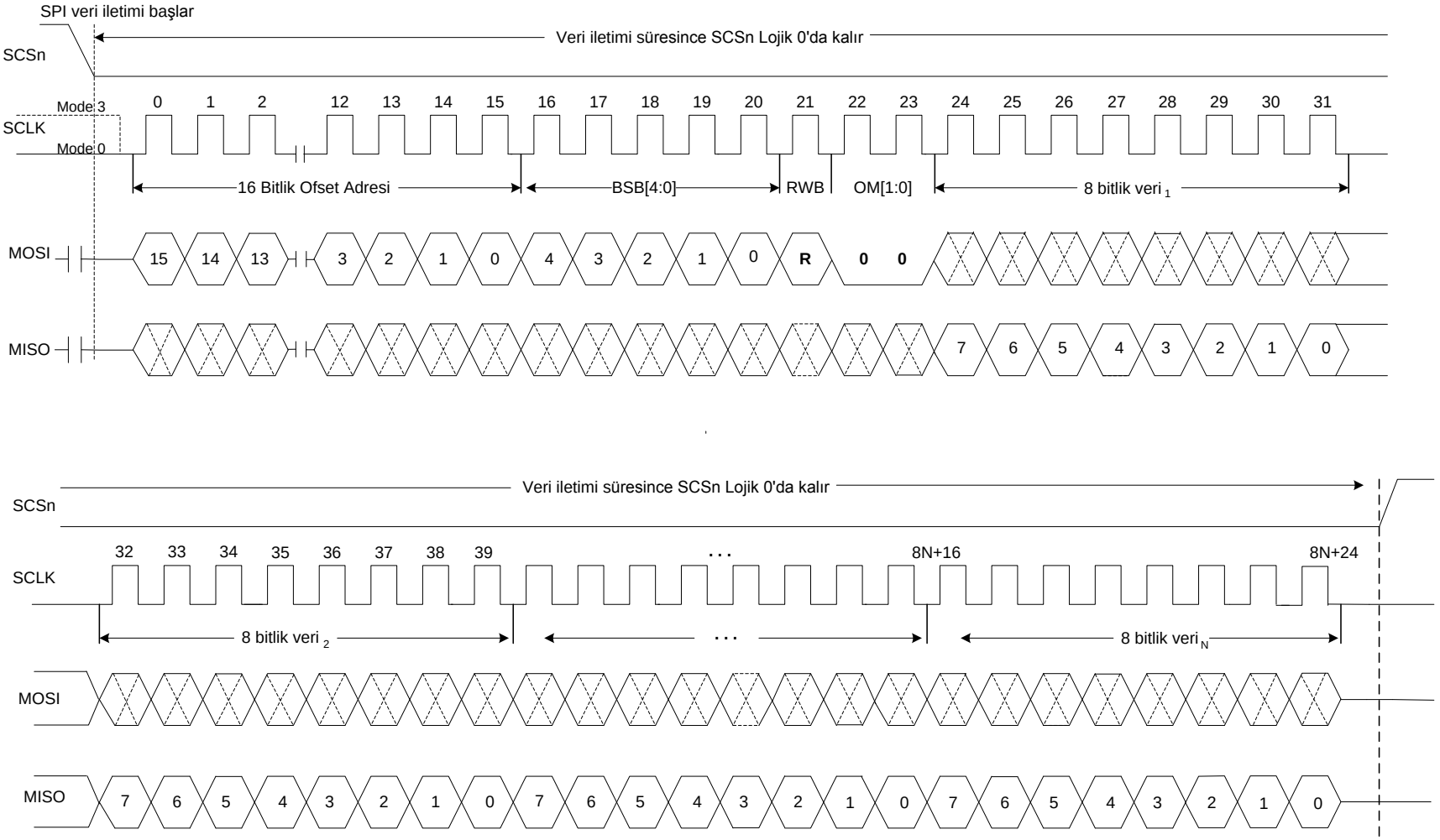
Ofset Adresi	0x0040	Adres bilgisi
BSB [4:0]	'00110'	Blok seçim bilgisi
RWB	'1'	Veri yazma
OM	'00'	Operasyon modu
Veri	0x11 0x22 0x33 0x44 0x55	Veri

Bu işlem sonucunda Wiznet veri çerçevesi Şekil 3.14'deki gibi olur;



Şekil 3.14 : W5500 molülü N bitlik veri paketi.

Şekil 3.15'de veri yazma işlemi için SPI veri çerçevesi verilmiştir;



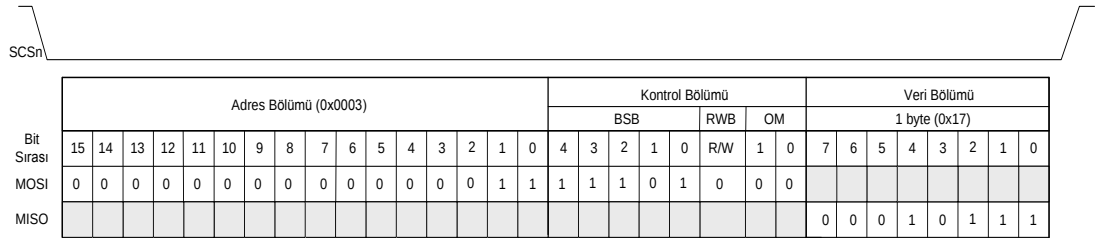
Şekil 3.15 : W5500 modülü için veri okuma paketi.

Şekil 3.15'de verilen veri yazma diyagramı dikkate alındığı zaman Wiznet modülünden bir byte okuma işlemi aşağıdaki gibi gerçekleştirilir.

Çizelge 3.6 : W5500 için bir bitlik veri.

Ofset Adresi	0x0003	Adres bilgisi
BSB [4:0]	'11101'	Blok seçim bilgisi
RWB	'0'	Veri yazma
OM	'00'	Operasyon modu
Veri	0x17	Veri

Bu işlem sonucunda Wiznet veri çerçevesi Şekil 3.16'deki gibi olur;

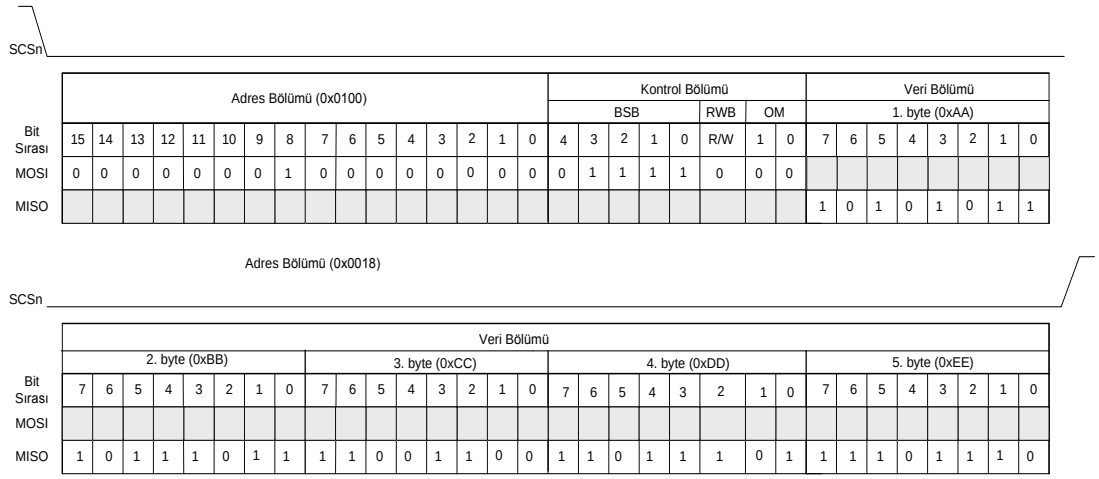


Şekil 3.16 : W5500 molülü bir bitlik veri okuma paketi.

Wiznet modülünden N byte okuma işlemi aşağıdaki gibi gerçekleştirilir.

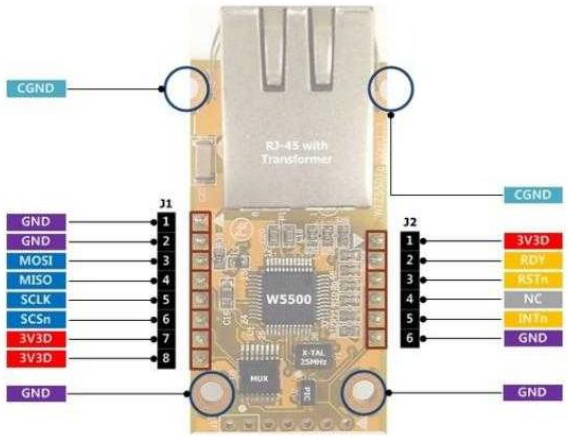
Çizelge 3.7 : W5500 için N bitlik veri.

Ofset Adresi	0x0100	Adres bilgisi
BSB [4:0]	'01111'	Blok seçim bilgisi
RWB	'0'	Veri yazma
OM	'00'	Operasyon modu
Veri	0xAA 0xBB 0xCC 0xDD 0xEE	Veri



Şekil 3.17 : W5500 molülü N bitlik veri okuma paketi.

Wiznet modülü üzerindeki pinler Şekil 3.18'de gösterilmiştir;



Şekil 3.18 : W5500 pin diyagramı.

W5500 modulünün pin açıklamaları Çizelge 3.8'de açıklanmıştır.

Çizelge 3.8 : W5500 pin açıklamaları.

Pin	Pin No	Açıklama
CGND		Şase Toprağı
GND	J1-1-2, J2-6	Toprak
MOSI	J1-3	Master Çıkış-Slave Giriş
MISO	J1-4	Master Giriş- Slave Çıkış
SCLK	J1-5	Slave Saat Sinyali
SCSn	J1-6	Slave Seçim Biti
3V3D	J1-7-8, J2-1	3.3 Volt
RDY	J2-2	Hazır
RSTn	J2-3	Reset
NC	J2-4	Bağlı değil
INTn	J2-5	Kesme Biti

3.3 Modbus Giriş - Çıkış Modülleri

Tez çalışmasında yapılan uygulamada Advantech firmasının ADAM Modbus RTU ve TCP giriş çıkış modulleri kullanılmıştır. Firmanın ürünleri incelenmiş ve iki adet TCP modulu ve iki adet RTU modulü seçilmiştir. Seçilen ürünler ve model numaraları Çizelge 9'da verilmiştir.

Çizelge 3.9 : ADAM cihazları ve özellikleri.

OP Modu	Açıklama
ADAM 4024	4 Kanal Modbus RTU Analog Çıkış Modülü
ADAM 4055	16 Kanal Modbus RTU Dijital G/Ç Modülü
ADAM 6066 (2)	12 Kanal Modbus TCP Dijital G/Ç Modülü

Modbus giriş çıkış modülleri ile ilgili konfigürasyon ayarları teknik dökümanlarında anlatıldığı gibi yapılmıştır. Bu modüller ile ilgili detayı bilgi [27-28] numaralı kaynaklardan edinilebilir. Şekil 3.19'da ADAM Modbus giriş/çıkış modülleri gösterilmiştir.

Şekil 3.19 : Adam Modbus giriş/çıkış modülleri.

Bu modüllere ek olarak Phoenix firmasının IL ETH DI8 DO4 Modbus TCP giriş çıkış modüllerinden 2 adet kullanılmıştır. Şekil 3.20'de bu modül gösterilmiştir. Bu modül ile ilgili detaylı bilgiye [29] numaralı kaynaktan edinilebilir.

Şekil 3.20 : IL ETH DI8 DO4 Modbus giriş çıkış modülü.

3.4 Switch

Switch kendine bağlı bulunan bütün cihazların MAC adreslerini ve hangi potuna bağlı olduğu bilgisini hafızazında tutabilen akıllı cihazlardır. Switch kullanılan endüstriyel sistemlerdeki bir birim başka bir birim ile haberleşmek istediği zaman, paketi switche gönderir ve switch bu paketin alıcı MAC adresine bakar, tablosundaki MAC adresleri ile karşılaştırır. Eğer MAC adresi switchin adres tablosundaki MAC adreslerinden biri ile eşleşirse paketi o portuna yönlendirir, eşleşmezse paketin geldiği port hariç bütün portlarına paketi gönderir. Bir birim switche ilk kez bağlandığında ARP paketi gönderir ve switche kendisini tanıtır. Switch bu ARP paketini alarak cihazın MAC adresini ve portunu tablosuna kaydeder ve daha sonra

bu MAC adresi ile eşleşen bir paket geldiği zaman paketi bu birimin portuna yönlendirir [30].

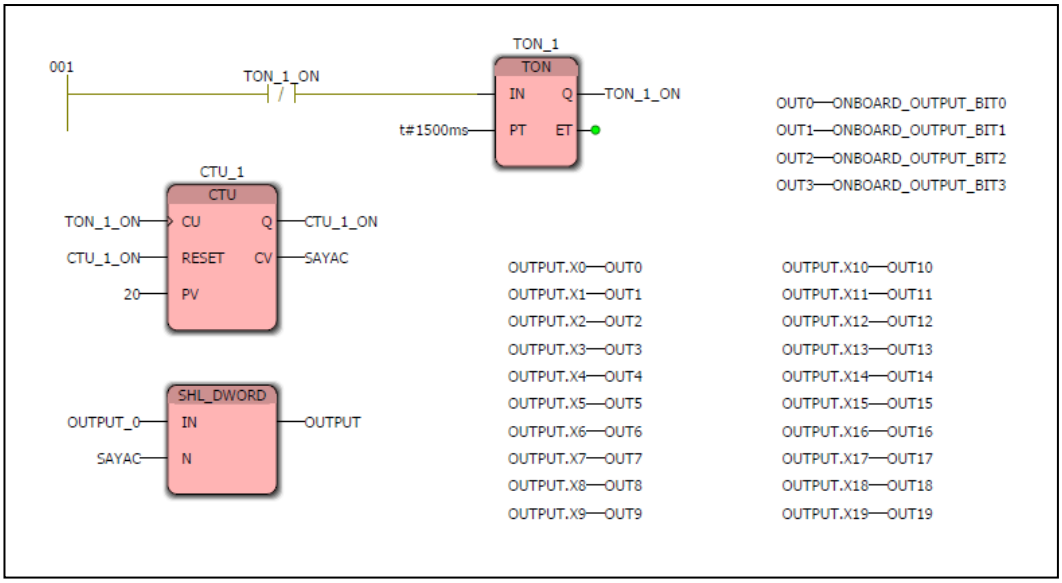
Tez çalışmasında TP-Link yönetilebilir (managable) switch kullanılmıştır. Bu switch aynı zamanda port aynalama (port mirroring) özelliğine sahiptir ve herhangi bir portundan gönderilen veri paketlerini başka bir portuna da kopyalayarak yardımcı yazılımlar sayesinde Ethernet ağı üzerinde veri paketlerinin gözlemlenmesine izin verir. Bu çalışmada TCP pakelerini gözlemleyebilmek için, ücretsiz olan *Wireshark* programı kullanılmıştır. Wireshark Windows, Linux, MacOS gibi bir çok işletim sisteminde çalışabilen ve bilgisayara bağlı olan her türlü ağ kartındaki tüm TCP/IP masajlarını analiz eden bir programdır.

3.5 PLC

PLC, dış algılayıcılardan aldığı bilgiyi, kendisine yüklenen programa göre işleyen ve iş elemanlarına aktaran mikroişlemci tabanlı bir cihazdır. PLC'ler endüstriyel otomasyon sistemlerinin kumanda ve kontrol devrelerine uygun yapıda giriş/çıkış birimleri ve iletişim ara birimeri ile donatılmış ve endüstriyel ortamlarda çalışabilecek şekilde tasarlanmış bilgisayarlardır. Üzerinde bulunan dijital ve analog giriş-çıkış modülleri sayesinde bir çok endüstriyel cihaz kontrol edilebilmektedir. Daha geniş bir tanımla PLC, mantık, sıralama, sayma, veri işleme, karşılaştırma ve aritmetik işlemler gibi fonksiyonları programlama desteği ile girişleri değerlendirip çıkışlara atayan, hafıza, giriş/çıkış, CPU ve programlayıcı bölümlerinden oluşan, otomasyon devrelerinde yardımcı röleler, zaman röleleri, sayıcılar gibi kumanda elemanlarının yerine kullanılan mikroişlemci tabanlı endüstriyel bir cihazdır. PLC'ler desteklediği haberleşme protokollerini kullanarak diğer endüstriyel cihazlar ile veri alış verişi yapabilirler. Tez çalışmasında Phoenix Contact firmasının ILC550ETH PLC'si kullanılmıştır. ILC150ETH, TCP/IP uyumlu cihazlar ile haberleşebilen ve Modbus TCP protokolünü destekleyen bir cihazdır. PLC içersine PC-Works programı kullanılarak yazılan uygulama programı ile FPGA tabanlı Modbus ağ gerçekleştirilmiştir [31].

Tasarlanan FPGA tabanlı ağ geçidi cihazın Modbus RTU ve TCP protokollerini kullanarak doğru bir şekilde veri haberleşmesi yaptığını göstermek için basit bir uygulama programı hazırlanmıştır. Program iki alt bloktan oluşur. Şekil 3.21'de

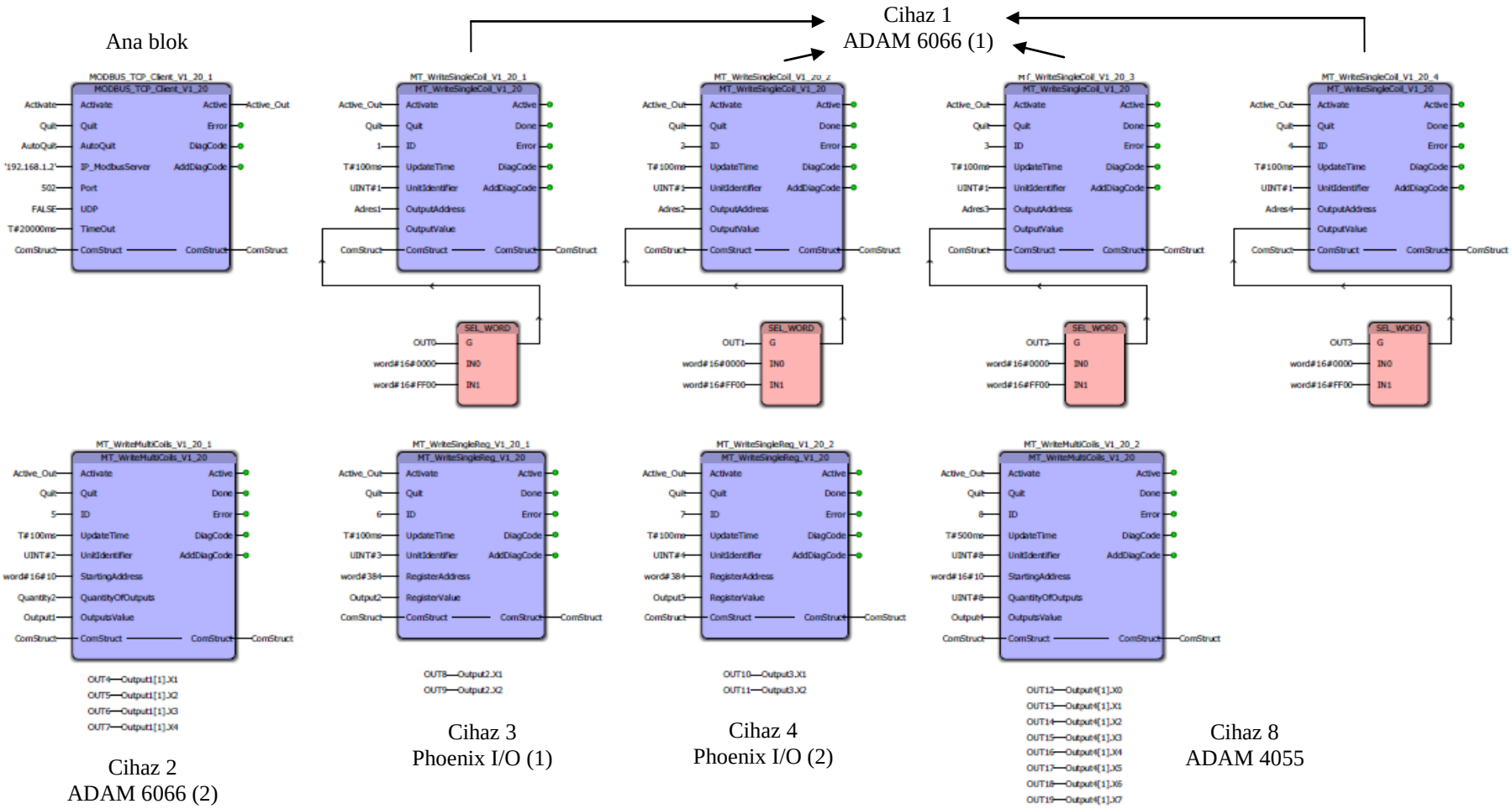
verilen ilk program blogu TCP ağı üzerinde çalışan istemci PLC'nin tasarlanan FPGA tabanlı ağ geçidi cihaz üzerinden TCP ve RTU ağlarına bağlı cihazlara göndermek istediği talebi anlatır. İstemci cihaz Şekil 3.21'de ifade edilen işlem ile sıra ile tüm çıkış bitlerini önce lojik 1, sonra lojik 0 yapar.



Şekil 3.21 : İstemci PLC cihazın talep programı.

Programda bir tane zamanlayıcı bloğu (timer, TON_1), bir tane sayıcı bloğu (counter, CTU_1) ve bir tane sola kaydırma bloğu (SHL_DWORD) kullanılmıştır. Zamanlayıcı bloğu (TON_1), her 1500 ms'de bir pulse sinyali üretir ve bu sinyal ile sayıcı blok (CTU_1) değerini bir artırır. Sayıcı bloğun maksimum değeri 20'dir ve bu değere ulaştığı zaman kendini resetler. Dolayısı ile sola kaydırma bloğu (SHL_DWORD) maksimum 19 bit kaydırma işlemi yapılır. PLC'nin üzerindeki 4 çıkış ve PLC'ye bağlanan ek modüllerde toplam 16 çıkış bulunur. Sonuç olarak toplam 20 çıkış üzerinde kaydırma işlemi gerçekleştirilir.

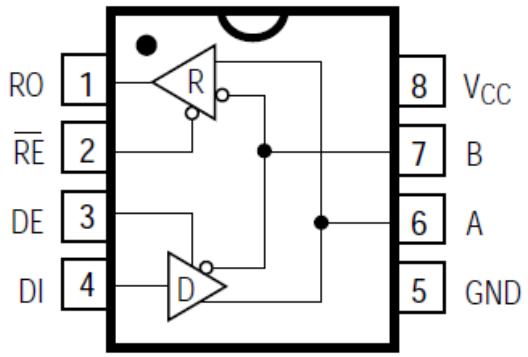
Şekil 3.21'de ikinci program bloğu ise talebi tasarlanan ağ geçidi üzerinden diğer Modbus cihazlara göndermek için hazırlanan haberleşme programıdır. Cihaz 1 ile haberleşmek için Fonksiyon Kod 5, Cihaz 2 ile haberleşmek için fonksiyon Kod 15, Cihaz 3 ve 4 ile haberleşmek için Fonksiyon Kod 6 ve Cihaz 8 ile haberleşmek için Fonksiyon Kod 15 kullanılmıştır. Haberleşme bloğu alt programı Şekil 3.22'de gösterilmiştir.



Şekil 3.22 : Haberleşme bloğu.

3.6 RS485 Dönüştürücü Devresi

Modbus RTU haberleşmesi ile veri haberleşmesi yapılırken fiziksel katmanda verinin RTU hattına gönderilmesi için gerilim seviyeni ayarlamak için FPGA ile Modbus RTU modulleri arasında ek bir devreye ihtiyaç duyulur. Bu ek devrenin kurulması için MAX 485 entegresi kullanılmıştır. MAX485, RS-485 ve RS-422 haberleşmesi için düşük güçlü bir dönüştürücü entegredir. Şekil 3.23'de MAX 485 entegresinin pin bağlantıları ve Çizelge 3.10'da pin açıklamaları verilmiştir.



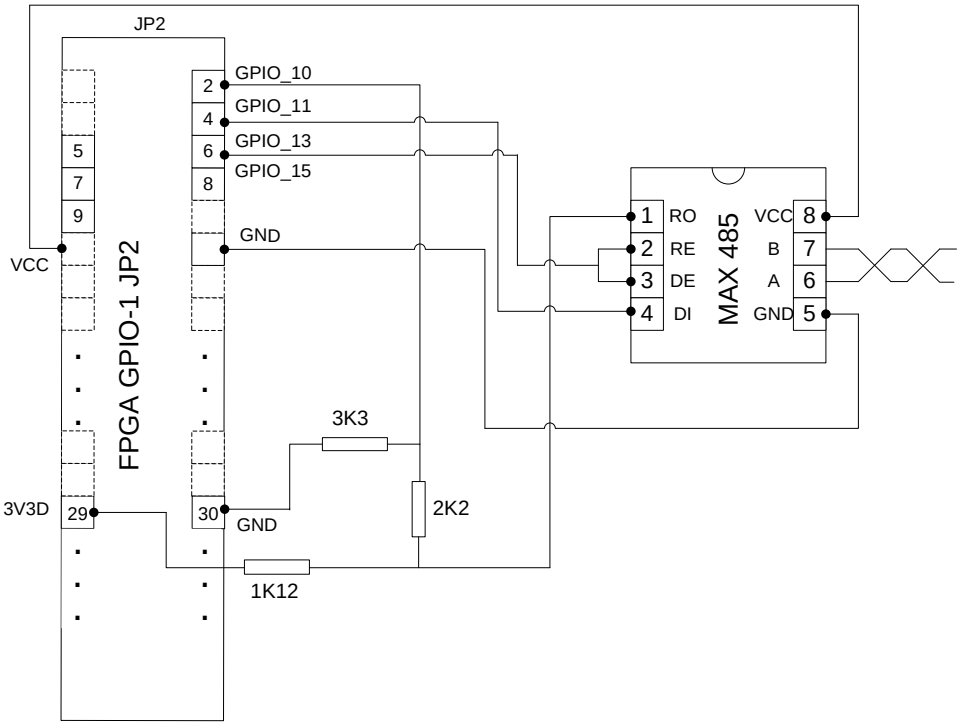
Şekil 3.23 : MAX 485 entegresi [32].

Çizelge 3.10 : MAX 485 pin açıklamaları [32].

Pin	Pin No	Açıklama
RO	1	Alıcı çıkış pini (Receive Output)
RE	2	Alıcı çıkışı etkinleştirme pini (Receive Output Enable)
DE	3	Sürücü çıkışı etkinleştirme pini (Driver Output Enable)
DI	4	Sürücü girişi (Driver Input)
VCC	5	Besleme sinyali (Power Supply)
B	6	Eviren alıcı çıkışı (Inverting Receiver Input)
A	7	Evirmeyen alıcı girişi (Non-Inverting Receiver Input)
GND	8	Toprak sinyali (Ground Signal)

MAX 485 entegresinin besleme ve toprak bağlantıları DE0 Nano üzerinden sağlanır ve RO, RE, DE ve DI pinleri DE0 Nano kartının genel amaçlı giriş çıkış pinlerine (JP2) bağlanır. A ve B uçları veriyi ileten RS 485 verisini taşıyacak kablolara bağlanır.

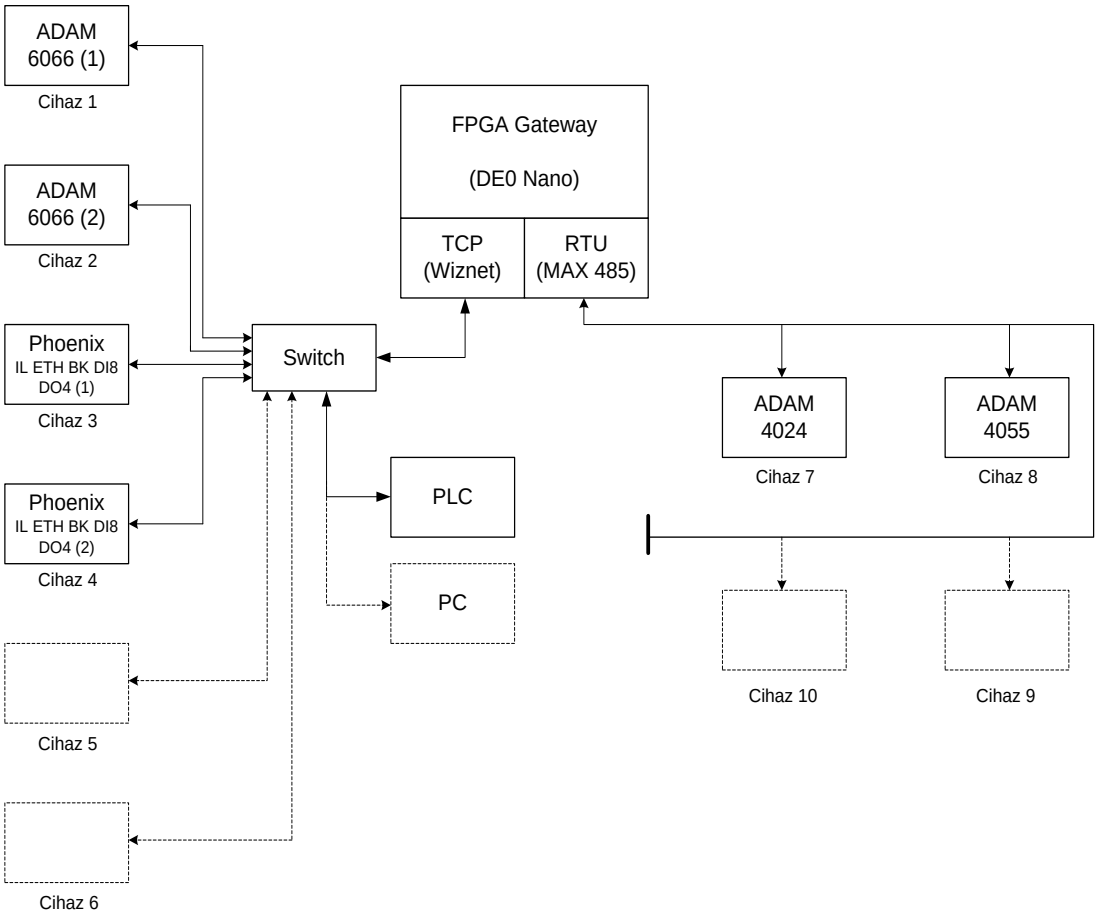
RS 485 dönüştürücü devresinin elektronik şeması Şekil 2.24'de gösterilmiştir.



Şekil 3.24 : MAX 485 ve FPGA bağlantısı.

4. UYGULAMA VE YAZILIM

Tez çalışmasının uygulama bölümünde yukarıda anlatılan bilgiler kullanılarak FPGA tabanlı Modbus Ağ geçidi cihazı tasarlanmış ve hazırlanan endüstriyel test düzeneği üzerinde bu tasarım test edilmiştir. Test düzeneği blok şeması Şekil 4.1'de verilmiştir.



Şekil 4.1 : Uygulama Platformu.

Ağ geçidi, Modbus TCP Ethernet haberleşme ağı ile Modbus RTU seri haberleşme ağı arasında veri alışverişi sağlayan bir cihazdır. Tasarlanan FPGA tabanlı ağ geçidi, Modbus TCP ağı üzerine bağlı istemci cihazın (PLC ya da PC) taleplerine uygun olarak, Modbus TCP ağına bağlı sunucu cihazlara ve Modbus RTU ağına bağlı bağımlı cihazlara ulaşabilmektedir. Bu durumda FPGA, Modbus TCP ağında hem

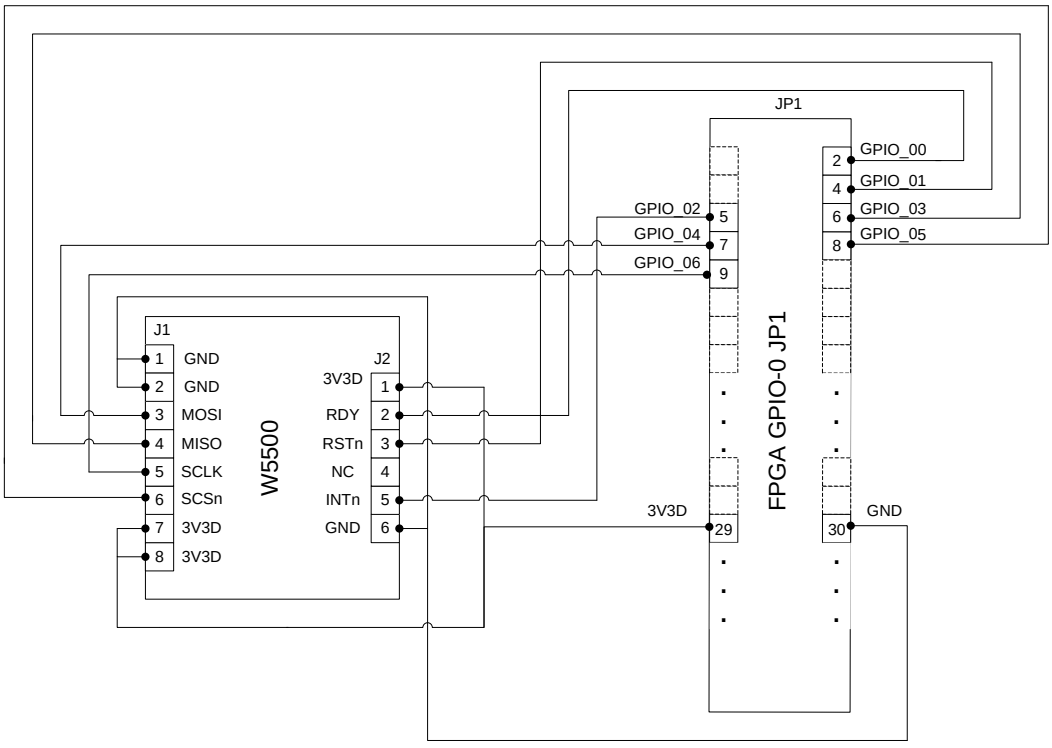
istemci hem de sunucu olarak davranmakta ve Modbus RTU ağında ana cihaz olarak işlev görmektedir.

Bu bölümde uygulama için hazırlanan yazılım anlatılmış ver test düzeneği tanıtılmıştır. Öncelikle VHDL dili kullanılarak FPGA içersinde tasarlanan donanım modülleri anlatılmıştır. FPGA'in Wiznet modülü ile haberleşebilmesi FPGA içersinde *SPI modülü* tasarlanmıştır. Buna ek olarak modüller ile ilgili gerekli konfigürasyon ayarlarının depolandığı EEROM modülü ile haberleşebilmek için *I2C modülü* tasarlanmıştır. Devamında FPGA içersine gömülen işlemci içersine yazılan program anlatılmış, deney düzeneği elemanları ve görevleri tanıtılmış ve uygulama sonuçlarına yer verilmiştir

4.1 SPI Modülü

Wiznet modülü 32-bitlik SPI haberleşmesini desteklemektedir ve sistemdeki görevi TCP bağlantısını sağlamak, veriyi işleyerek Modbus uygulama katmanına için uygun hale getirmektir. Wiznet Modülü ve FPGA'in haberleşmesi için FPGA içerinde VHDL dili kullanılarak SPI modülü tasarlanmıştır.

Wiznet Modülünün FPGA ile bağlantı şekli Şekil 4.2'de verilmiştir.



Şekil 4.2 : Wiznet Modülü ve FPGA'in bağantı şekli.

Wiznet modülü ve FPGA arasındaki bağlantı Şekil 4.2'deki gösterildiği gibi modulün üzerinde bulunan giriş/çıkış pinleri FPGA DE0 Nano kartının genel amaçlı giriş/çıkışlarına (JP1) bağlanmıştır.

FPGA dışarıdan gelen 50 Mhz'lik saat sinyali ile tetiklenir. Bu nedenle Wiznet modülünü tetikleyen SCLK sinyalinin 50 Mhz'lik ana saat sinyalinden üretilmesi gerekir. SCLK sinyali Nios II işlemcisi içersinden komut geldiği zaman üretilmeye başlanır. Wiznet modülü için 3.125 Mhz çalışma frekansı belirlenmiştir.

SPI modülü için Quartus II programında VHDL dili kullanılarak aşağıdaki kod blokları yazılmıştır. Şekil 4.3'te verilen kod parçasında *"clk_clk"* sinyali FPGA'i dışarıdan besleyen 50 MHz'lik osilatör sinyalini ifade eder. *"process"* blogu FPGA'in içersinde paralel olarak çalışan kod parçalarını yazmak için kullanılır ve duyarlılık listesi içersinde yer alan sinyal tetiklendiği zaman bu blok içersindeki kod parçası çalışır.

```
process (clk_clk)
   begin
      if(output_export(0) = '1' and output_export(0)'event) then
         input_export(0)  <= '1';
      end if;
      if (clk_clk = '1' and clk_clk'event and input_export(0) = '1') then
         counter_spi <= counter_spi + '1';
      end if;
      if (counter_spi = "1111111111") then
         input_export(0)  <= '0';
         counter_spi <= "0000000000";
      end if;
end process;
```

Şekil 4.3 : "counter_spi" alt bloğu.

İşlemci (Nios) "*output_export*" sinyalinin sıfırıncı bitini lojik 1 değerine getirince SPI modülü, "*input_export*" sinyalinin sıfırıncı bitini lojik 1 seviyesine çeker ve bu bilgiyi işlemciye gönderir. Bu işlemden sonra SPI modülü içersinde bulunan 10 bitlik (0 downto 9) "*counter_spi*" sinyali saymaya başlar. Sinyal üst seviyeye ulaşında kendini sıfırlar ve "*input_export*" sinyalinin sıfırıncı biti lojik 0 seviyesine çekilir. Bu aşamada FPGA'in paralel işlem yapabilme yeteneğinden faydalanılır ve "*counter_spi*" sinyali kullanılarak SLCK sinyali ve SCSn sinyali ayarlanarak veri okuma ya da yazma işlemi gerçekleştirilir. "*counter_spi*" sinyali bit seviyesinde sayma işlemi gerçekleştirdiği için sinyal içindeki her bir ayrı bir tetikleyici sinyal olarak kullanılabilir.

Şekil 4.4'te verilen kod parçası "*counter_spi"* sinyalinin değişimi ile Wiznet modulünden veri okuma ve veri yazma işleminin nasıl gerçekleştirildiğini ifade eder.

```
process (counter_spi)
begin
   if (counter_spi(1) = '1' and counter_spi(1)'event) then
      --yazma
      if(counter_spi(9)='0' and spi_wr_export(10) = '1' and counter_spi(3)='0') then
         MOSI <= spi_wr_export(31-conv_integer(counter_spi(8 downto 4)));
         spi_rd_export0 <= X"55";
      end if;
      --okuma
      if(counter_spi(9)='0' and spi_wr_export(10) = '0') then
         if(counter_spi(8 downto 4)<"11000"  and counter_spi(3)='0') then
            MOSI <= spi_wr_export(31-conv_integer(counter_spi(8 downto 4)));
         end if;
         if(counter_spi(8 downto 4)>="11000"  and counter_spi(3)='1') then
            spi_rd_export0(7-conv_integer(counter_spi(6 downto 4))) <= MISO;
         end if;
      end if;

      SCLK <= counter_spi(3) and (not counter_spi(9));
      if (counter_spi(9) = '1' and counter_spi(5) = '1') then
         SCSn <= '1';
      end if;
      if (counter_spi(9) = '0' and input_export(0) = '1') then
         SCSn <= '0';
      end if;

      input_export(1)  <= RDY;
      input_export(2)  <= Intn;
      input_export(3)  <= counter_spi(3) and (not counter_spi(9));
      RSTn <= output_export(2);
   end if;
end process;
```

Şekil 4.4 : "spi veri okuma/yazma" *alt bloğu.*

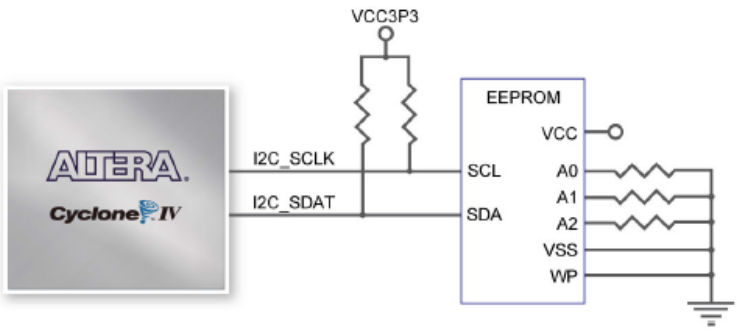
Wiznet modulü ile veri okuma ve yazma işlemi "*counter_spi"* sinyalinin birinci birinin lojik 1 seviyesine gelmesi ile başlar. Veri yazma işlemi "*counter_spi"* sinyalinin dokuzuncu ve üçüncü bitinin lojik 1 ve "*spi_wr_export"* sinyalinin onuncu bitinin lojik 1 seviyesine gelme koşulu sağlandığı zaman başlar. "*spi_wr_export"* sinyali Nios işlemcisi içersinde Wiznet modülüne gönderilecek veriyi depolar ve onuncu biti Bölüm 3.2'de alnatılan Wiznet modülü içim Şekil 3.12'de gösterilen okuma/yazma bitine karşılık gelir ve bu bit değeri lojik 1 yapıldığında Wiznet'e veri yazılacağı ifade edilmiş olur ve *"MOSI"* sinyali üzerinden veriler Wiznet modulüne aktarılır. Veri okuma işlemi yapılırken önce hangi blogunun adresinin okunacağı Wiznet modülüne bildirilmesi gerekir, yani önce module yazma işlemi yapılır, sonra ilgili adresten veri okunur. Bu işlem, işlemi "*counter_spi"* sinyalinin dokuzuncu bitinin lojik 1 ve "*spi_wr_export"* sinyalinin onuncu bitinin lojik 0 seviyesine gelme koşulu sağlandığı zaman başlar. "*spi_wr_export"* sinyalinin onuncu bitinin lojik 0 seviyesine getirilmesi Wiznet modulünden veri okunacağını belirtir ve modül önce adres bilgisini bekler. *"MOSI"* sinyali üzerinden adres bilgisi Wiznet'e aktarıldıktan

sonra *"MISO"* sinyali üzerinden SPI modulü sinyali alarak Nios işlemcisine aktarır. Okuma ve yazma işlemlerinin doğru bir şekilde gerçekleşmesi için Wiznet modulünü tetikleyen *"SCLK"* saat sinyalinin ve *"SCSn"* sinyalinin (slave select biti) doğru bir şekilde çalışması çok önemlidir. *"SCLK"* saat sinyali, "*counter_spi"* sinyalinin üçüncü bitinin ve dokuzuncu bitinin değilinin *"and"* operatöründen geçirilmesi ile elde edilir. *"SCSn"* sinyali, "*counter_spi"* sinyalinin dokuzuncu ve beşinci birinin lojik 1 seviyesinde olduğu zaman lojik 1 seviyesinde ve "*counter_spi"* sinyalinin dokuzuncu bitinin lojik 0 ve "*input_export"* sinyalinin sıfırıncı bitinin lojik 0 seviyesine getirlildiğinde lojik 0 seviyesinde tutularak sistemin doğru bir şekilde çalışması sağlanmıştır.

4.2 I2C Modülü

I2C (Inter-Integrated Circuit), Philips firması tarafından geliştirilmiş, düşük hızlı çevre birimlerini anakart, gömülü sistem gibi cihazlara bağlamak için kullanılan toprağa referanslı bir seri veriyoludur. Pull-up dirençleri ile pozitif beslemeye bağlanmış iki adet çift yönlü open-drain sinyal hattı kullanır. Sinyalerden biri veri taşıma sinyalidir (Serial Line, SDA) ve diğeri de EEPROM modülü için saat sinyalidir (Serial Clock, SCL). Daha düşük ve yüksek gerilimlerine izin verilmekle birlikte genelikle +5V ve + 3.3V gerilim değerleri kulanılır. Her iki hat pull-up dirençleri ile besleme gerilimine çekilmesinin nedeni haberleşme olmadığı durumda her iki hattı da lojik 1 seviyesinde tutmaktır.

DE0-Nano FPGA Eğitim ve Geliştirme Kartı üzerinde 2Kbit EEPROM (Electrically Erasable PROM) bulunur. EEPROM iki kablolu I2C seri arabirimi ile konfigüre edilir. EEPROM cihazının FPGA ile bağlantısı Şekil 4.5'de gösterilmiştir.



Şekil 4.5 : EEPROM ve FPGA bağlantısı.

DE0-Nano üzerinde bulunan EEPROM, Mikrochip firmasının 24LC02B kodlu 256x8bit hafızası olan ve I2C haberleşmesi ile veri alış verişi sağlayan modüldür. EEPROM modülüne, ağ üzerinde bulunan Modbus cihazlarının konfigürasyon bilgileri depolanmıştır. Böylece sistem ilk enerjilendiğinde gerekli ayarlar EEPROM üzerinde okunarak Modbus cihazlara gönderilmektedir. EEPROM ile Nios işlemcisi arasında veri alış verişini sağlamak için FPGA içersine VHDL dili kullanılarak *I2C modülü* tasarlanmıştır.

24LC02B EEROM modulü çift yönlü iki hatlı veri yolu ve veri iletim protokolünü (I2C) destekler. Veri yoluna veriyi ileten cihaz gönderici (transmitter) ve veriyi alan cihaz da alıcı (receiver) olarak adlandırılır. Veri yolu, saat sinyalini (serial clock, SCL) üreten ana cihaz (master) tarafından kontrol edilir. EEPROM modulü de bağımlı cihaz (slave) olarak çalışır. Hem ana cihaz hem de bağımlı cihaz (EEPROM) gönderici ve alıcı olarak çalışabilmesine rağmen hangi çalışma modunun seçileceği sadece ana cihaz tarafından belirlenir. Veri iletimi sadece veri yolu meşgul değilken başlatılır ve veri iletimi sırasında saat sinyali yükselen kenera geldiği zaman veri sinyali sabit kalmalıdır. Aksi durum başlangıç ya da bitiş durumu olarak yorumlanıp veri iletiminde hata oluşabilir. Veri hattının açık olduğunu ifade etmek için hem saat sinyali hem de veri sinyali lojik 1 seviyesinde tutulur (Bus not busy). Saat sinyali lojik 1 seviyesinde iken veri sinyalinin lojik 1'den lojik 0 durumuna geçmesi veri alışverişinin başlayacağını ifade eder (Start data transfer). Saat sinyali lojik 1 seviyesinde iken veri sinyalinin lojik 0'dan lojik 1 durumuna geçmesi veri alışverişinin sona erdiğini ifade eder (Stop data transfer). *"start"* bitini gönderildikten sonra veri yolunun durumu geçerli veri bilgisini temsil eder. Veri yolu saat sinyalinin yükselen kenarında sabit kalmalıdır ve veri iletimi düşen kenarda gerçekleştirilmelidir. Her bitin veri yoluna gönderimi bir saat sinyali içinde tamamlanmalıdır. Her veri iletim işlemi *"start biti"* ile başlar ve *"stop biti"* ile sonlanır. Gönderilecek olan veri biti sayısı ana cihaz tarafından belirlenir. Her alıcı cihaz veri alma işlemi sırasında, aldığı her byte bilgisinden sonra bir onay biti (acknowledge) üretir. Ana cihaz onay biti için fazladan bir saat sinyali üretir. EEPROM üzerinden veri okuma ve yazma işlemi aşağıdaki gibi gerçekleştirilir.

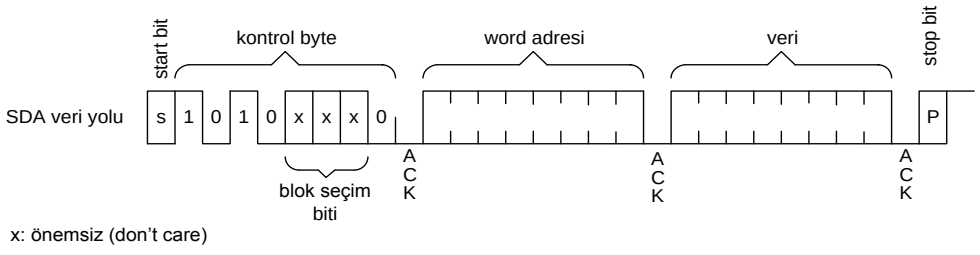
Veri okuma ve yazma işlemi için tasarlanan I2C modününce öncelikle 13 bitlik *"counter_i2c"* sinyali oluşturulmuştur. Bu işlemi gerçekleştiren kod parçası Şekil 4.6'da gösterilmiştir.

```
process (clk_clk)
   begin
      if(output_export(4) = '1' and output_export(4)'event) then
         input_export(4)  <= '1';
      end if;
      if (clk_clk = '1' and clk_clk'event and input_export(4) = '1') then
         counter_i2c <= counter_i2c + '1';
      end if;
      if (counter_i2c = "1111111111111") then
         input_export(4)  <= '0';
         counter_i2c <= "0000000000000";
      end if;
end process;
```

Şekil 4.6 : "counter_i2c" alt bloğu.

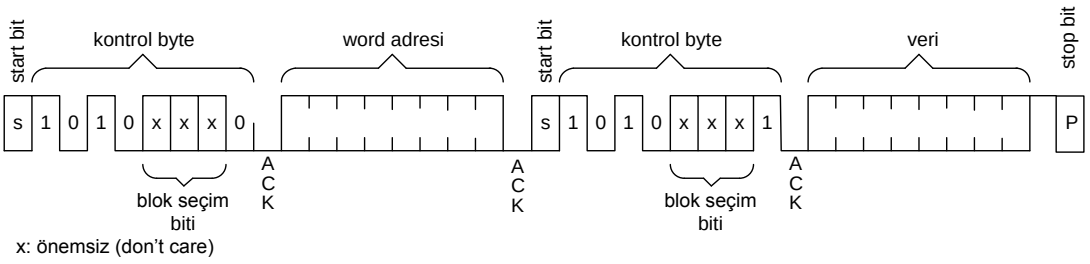
Şekil 4.6'da gösterildiği gibi Nios işlemcisi tarafından "*output_export"* sinyalinin dördüncü biti lojik 1 seviyesine getirildiği zaman, "*input_export"* sinyalinin dördüncü bitinin lojik 1 yapılır ve *"counter_i2c"* sinyali artmaya başlar. *"counter_i2c"* sinyali maksimum değere ulaşınca "*input_export"* sinyalinin dördüncü biti lojik 0 seviyesine getirilir ve *"counter_i2c"* sinyali sıfırlanır.

24LC02B EEROM, byte yazma (byte write) ve sayfa yazma (page write) olmak üzere iki çeşit veri yazma modunu destekler. I2C modülü byte yazma moduna göre hazırlanmıştır. Veri yazma işlemi Şekil 4.7'de verilen veri çerçevesine göre gerçekleştirilir.



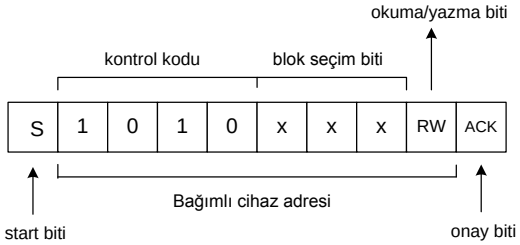
Şekil 4.7 : I2C veri yazma çerçevesi.

24LC02B EEROM, adres okuma (address read), rasgele okuma (random read) ve sıralı okuma (sequential write) olmak üzere üç çeşit veri okuma modunu destekler. *I2C modülü* rasgele okuma moduna göre yazılmıştır. Veri okuma işlemi Şekil 4.8'de verilen veri çerçevesine göre gerçekleştirilir.



Şekil 4.8 : I2C veri okuma çerçevesi.

Ana cihaz tarafından başlatılan veri alış verişi işleminde veri yoluna ilk gönderilen byte bilgisi "k*ontrol byte"* dır. İlk dört biti kontrol kodu olarak adlandırılır ve 24XX02 EPROM'lar için bu değer "1010" olarak kodlanır. Sonraki üç bit önemsizdir (don't care). Sonraki bit çalışma modunu belirler. '1' değeri için okuma ve '0' değeri için yazma işlemleri gerçekleştirilir. "k*ontrol byte"* verisinin içeriği Şekil 4.9'de gösterilmiştir.



Şekil 4.9 : "k*ontrol byte"* veri içeriği.

Veri yazma işleminde ikinci gönderilen byte bilgisi adres bilgisini ve üçüncü gönderien byte bilgisi de veri bilgisini içerir. Ayrıca veri yazma işlemi sırasında gönderilen her byte bilgisinden sonra onay biti beklenir. Veri okuma işlemi için önce hangi adresin okunması gerektiği EEPROM'a yazılır bu işlem için sırası ile kontrol byte bilgisi (okuma yazma biti: 0 olacak şekilde) ve adres bilgisi gönderilir, sonra tekrar kontrol byte bilgisi (okuma yazma biti: 1 olacak şekilde) EEPROM'a gönderilir ve devamında veri okuma işlemi gerçekleştirilir.

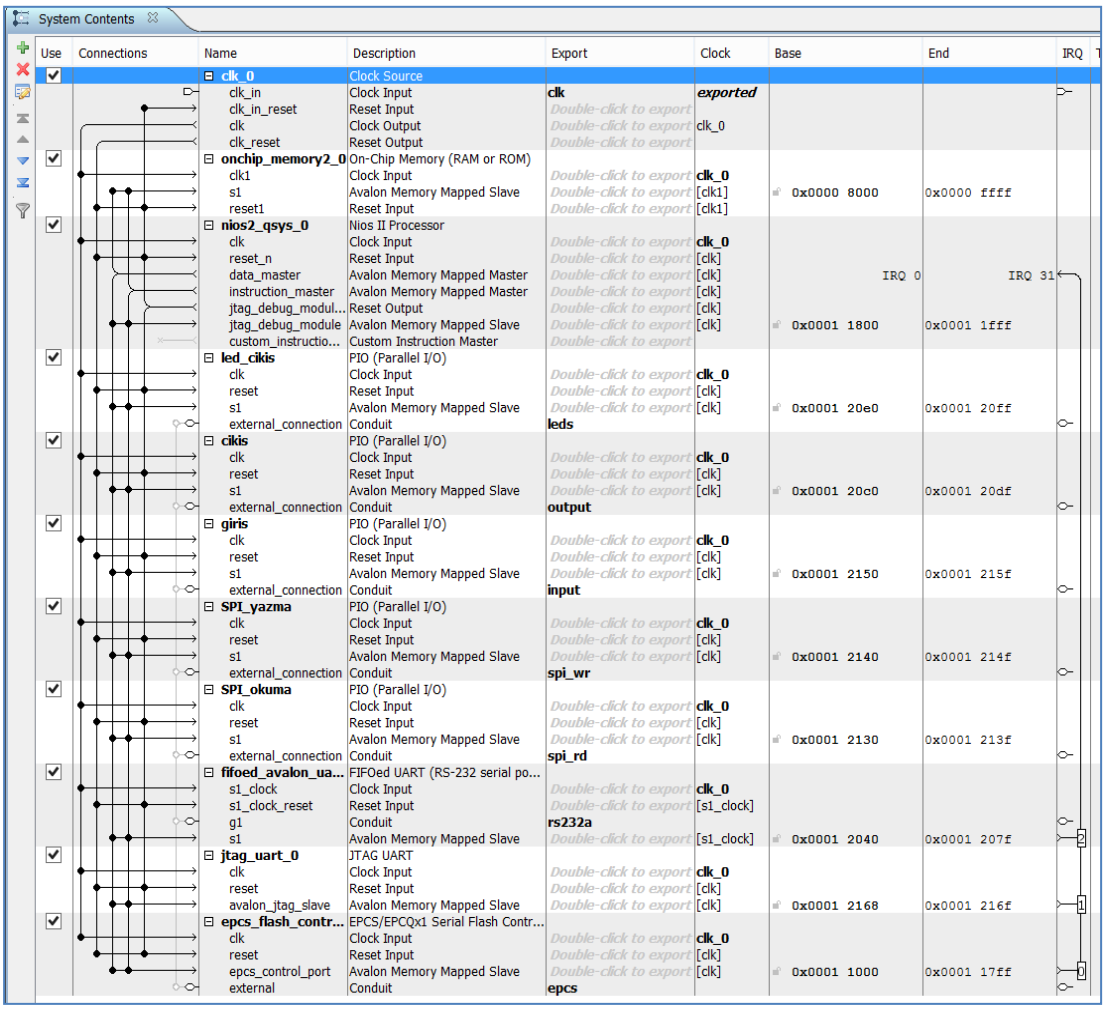
Şekil 4.7 ve 4.8'de verilen veri çerçeveleri dikkate alınarak EEPROM üzerine veri okuma ve yazma işlemi için VHDL dili ile Şekil 4.10'da gösterildiği gibi yapılır. *"counter_i2c"* sinyalinin değişimi ile *"process"* bloğuna girilir ve Nios işlemcisi tarafından "*spi_wr_export"* sinyalinin onaltıncı biti lojik 0 yapıldığı zaman EEPROM modulüne yazma işlemi ve lojik 1 yapıldığı zaman okuma işlemi gerçekleştirilir.

```
process (counter_i2c)
 begin
   if (counter_i2c(1) = '1' and counter_i2c(1)'event) then
      --yazma
      if(spi_wr_export(16) = '0') then
         if(counter_i2c(12 downto 5) = "00000000") then
            SDA <= '1'; --START
         elsif(counter_i2c(12 downto 5)= "00000001" or  counter_i2c(12 downto 5)= "00000010") then
            SDA <= '0'; --START
         elsif(counter_i2c(12 downto 5)>="00000011" and counter_i2c(12 downto 5)<="00000110") then
            SDA <= '1'; --control 7.bit
         elsif(counter_i2c(12 downto 5)>="00000111" and counter_i2c(12 downto 5)<="00001010") then
            SDA <= '0'; --control 6.bit
                                                    .
                                                    .
                                                    .
         elsif(counter_i2c(12 downto 5)>="01100111" and counter_i2c(12 downto 5)<="01101010") then
            SDA <= spi_wr_export(0);      --data 0.bit
         elsif(counter_i2c(12 downto 5)>="01101011" and counter_i2c(12 downto 5)<="01101110") then
            SDA <= 'Z';                   --data ack bit
         elsif(counter_i2c(12 downto 5)= "01101111" or counter_i2c(12 downto 5)=  "01110000") then
            SDA <= '0';                   --STOP
         elsif(counter_i2c(12 downto 5)>="01110001") then
            SDA <= '1';                   --STOP
         end if;

         if(counter_i2c(12 downto 5)>="01110000") then
            SCL <= '1';
         else
            SCL <= not counter_i2c(6);
         end if;
      end if;

      --okuma
      if(spi_wr_export(16) = '1') then
         if(counter_i2c(12 downto 5) = "00000000") then
            SDA <= '1'; --START
         elsif(counter_i2c(12 downto 5)= "00000001" or  counter_i2c(12 downto 5)= "00000010") then
            SDA <= '0';
         elsif(counter_i2c(12 downto 5)>="00000011" and counter_i2c(12 downto 5)<="00000110") then
            SDA <= '1'; --control 7.bit
         elsif(counter_i2c(12 downto 5)>="00000111" and counter_i2c(12 downto 5)<="00001010") then
            SDA <= '0'; --control 6.bit
                                                    .
                                                    .
                                                    .
            spi_rd_export1(1) <= SDA;  --data 1.bit
         elsif(counter_i2c(12 downto 5)="10010001") then
            spi_rd_export1(0) <= SDA;  --data 0.bit
         elsif(counter_i2c(12 downto 5)>="10010011" and counter_i2c(12 downto 5)<="10010110") then
            SDA <= '1';                --data no ack bit
         elsif(counter_i2c(12 downto 5)= "10010111" or counter_i2c(12 downto 5)=  "10011000") then
            SDA <= '0';                   --STOP
         elsif(counter_i2c(12 downto 5)>="10011001") then
            SDA <= '1';                   --STOP
         end if;
         if(counter_i2c(12 downto 5)>="10011000") then
            SCL <= '1';
         else
            SCL <= not counter_i2c(6);
         end if;
      end if;
   end if;
 end process;
```

Şekil 4.10 : EEPROM veri yazma alt bloğu.

4.3 Nios İşlemci, Çevresel Birimler ve İşlemcinin Programlanması

Qsys tasarım alanı kullanılarak tasarlanan Nios işlemcisi ve çevre birimleri Şekil 4.11'de gösterilmiştir.



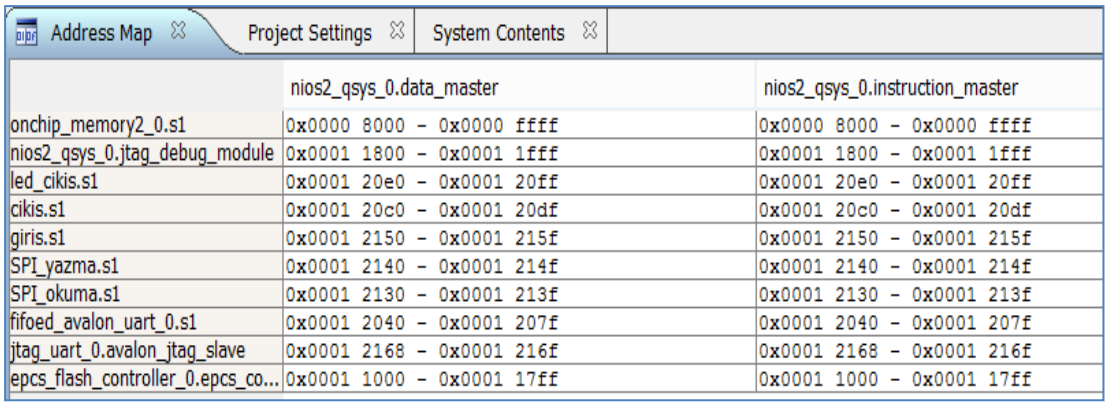
Şekil 4.11 : Tasarlanan işlemci.

Tasarlanan gömülü işlemci içersinde saat sinyali *clk_0* biriminden sağlanmaktadır. Sisteme hafıza birimi *onchip_memory2.0,* işlemci *nios2_qsys_0,* genel amaçlı giriş çıkışlar için *cikis* ve *giris* birimleri eklenmiştir. DE0 Nano kart üzerindeki ledler için '*led_cikis'*, SPI biriminden yazma işlemi yapmak için '*SPI_yazma'*, okuma yapmak için '*SPI_okuma* birimleri sisteme eklenmiştir. Bu birimler ayrıca I2C modülü ile EEPROM'a veri yazma ve okuma işlemleri için de kullanılmıştır. RTU birimlerinden gelen veriyi kaydetmek ve okumak için '*fifo_avalon_uart'* birimi ve konsola mesaj yazabilmek için '*jtag_uart_0'* birimi sisteme eklenmiştir. Ayrıca işlemciyi FPGA içersine kalıcı olarak gömebilmek için *'epcs_flash_controller_0'* birimi sisteme eklenmiştir.

Önce işlemci ve çevresel birimler kütüphane içinden seçilmiş ve tasarım alanına taşınmıştır. Sonra, gerekli ayarlamalar yapılmıştır. *'clk_0',* saat sinyalini FPGA'in harici 50MHz osilatör sinyalinden alır, işlemci ve tüm çevresel birimler bu hızla çalışır. *'onchip_memory2.0'* hafıza birimi RAM olarak ayarlanmış, 32 bitlik veri uzunluğunda ve toplam 32768 byte büyüklüğündedir. İşlemci olarak Altera'nin önerdiği üç işlemci arasından Nios II/e seçilmiş, reset ve kesme vektörleri ayarlanmıştır. Sistemde gerçekleşen belirli olayların ledler üzerinde görülebilmesi için kart üzerinde bulunan 8 led çıkış olarak ayarlanmış ve *'led_cikis'* birimi ile kontrol edilmektedir. Bu birim Qsys bileşenlerinden *Paralel Input/Output* biriminin 8-bitlik çıkış olarak ayarlanması ile oluşturulmuştur. İşlemcinin fonksiyonlarını yerine getirmesi için ihtiyaç duyulan ek sinyaller için kullanılan *'cikis'* ve *'giris',* birimleri de PIO'ların düzenlenmesi ile elde edilmiştir ve 8 bit boyutunda ayarlanmıştır. *'SPI_yazma'* birimi 32 bitlik çıkış ve *'SPI_okuma*, birimi 8 bitlik giriş olarak tanımlanmıştır. *'fifo_avalon_uart'* birimi kullanılarak sistemde bulunan RTU cihazlarının kontrolü sağlanmaktadır. 8 bitlik veri uzunluğu ve 9600 veri akış hızı seçilmiş ve eşlik biti kullanılmamıştır. *'jtag_uart_0'* birimi ile belirli işlemlerin konsola yazılması sağlamaktadır. Ayrıca tasarlanan işlemci ve çevresel birimleri FPGA içersine kalıcı olarak gömebilmek için *'epcs_flash_controller_0'* modülünden yararlanılır. *'fifo_avalon_uart', 'jtag_uart_0'* ve *'epcs_flash_controller_0'* birimlerinin ayarları default olarak ayarlanmıştır. *Paralel Input/Output* kullanılarak oluşturulan *'led_cikis', 'cikis' , 'giris', 'SPI_yazma' ve 'SPI_okuma*, birimlerinin çıkış pinlerinin isimleri tasarım alanının *export* bölümünde sırası ile *leds, output, input, spi_wr, spi_rd* olarak ve *'fifo_avalon_uart'* birimi için *rs232s* olarak isimlendirilmiştir. Bu pinleri Nios işlemcisinin dış birimlere bağlantısı olarak düşünülebilir.

Tasarım alanındaki birimlerin birbirine bağlanması Şekil 4.11'de gösterildiği gibi yapıldıktan sonra tüm sistem *Generate* sekmesi tıklanarak HDL kodları ile yazılımsal olarak oluşturulmuştur. Yazılımsal olarak oluşturulan işlemci ve arabirimler üzerinde VHDL kodları ile değiştirilemez, program yalnızca modüllerin giriş/çıkış bağlantılarının sinyallerle eşleştirilmesine izin vermektedir.

Qsys tasarım alanı kullanılarak tasarlanan sistemin adres haritasını otomatik Şekil 4.12'de gösterildiği gibi oluşturur. Bu adres bilgileri "*Nios II Software Built for Eclipse*" aracı ile C dili kullanılarak işlemci programlanırken kullanılmıştır.



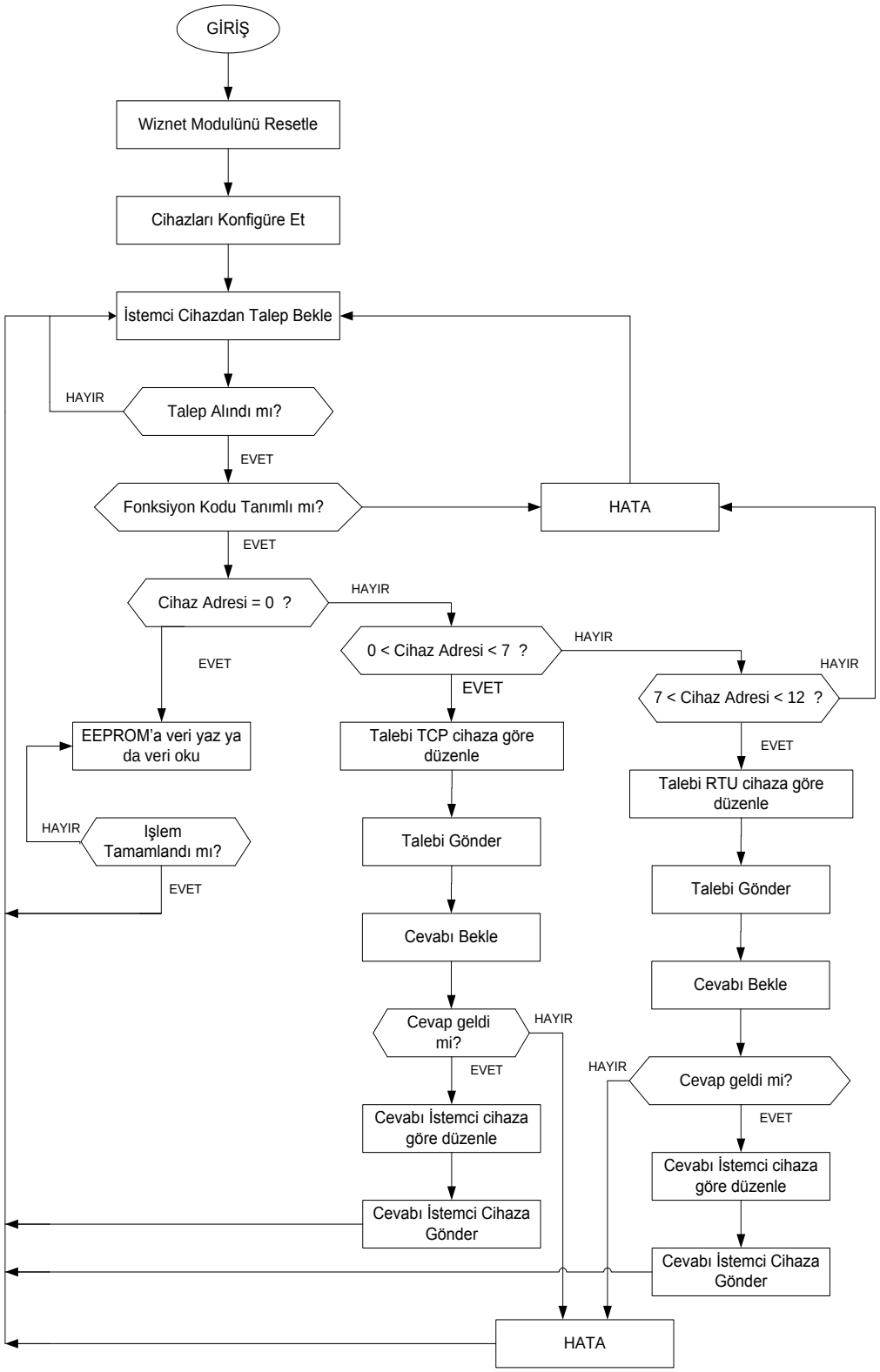
Address Map | Project Settings | System Contents

	nios2_qsys_0.data_master	nios2_qsys_0.instruction_master
onchip_memory2_0.s1	0x0000 8000 - 0x0000 ffff	0x0000 8000 - 0x0000 ffff
nios2_qsys_0.jtag_debug_module	0x0001 1800 - 0x0001 1fff	0x0001 1800 - 0x0001 1fff
led_cikis.s1	0x0001 20e0 - 0x0001 20ff	0x0001 20e0 - 0x0001 20ff
cikis.s1	0x0001 20c0 - 0x0001 20df	0x0001 20c0 - 0x0001 20df
giris.s1	0x0001 2150 - 0x0001 215f	0x0001 2150 - 0x0001 215f
SPI_yazma.s1	0x0001 2140 - 0x0001 214f	0x0001 2140 - 0x0001 214f
SPI_okuma.s1	0x0001 2130 - 0x0001 213f	0x0001 2130 - 0x0001 213f
fifoed_avalon_uart_0.s1	0x0001 2040 - 0x0001 207f	0x0001 2040 - 0x0001 207f
jtag_uart_0.avalon_jtag_slave	0x0001 2168 - 0x0001 216f	0x0001 2168 - 0x0001 216f
epcs_flash_controller_0.epcs_co...	0x0001 1000 - 0x0001 17ff	0x0001 1000 - 0x0001 17ff

Şekil 4.12 : Adres tablosu.

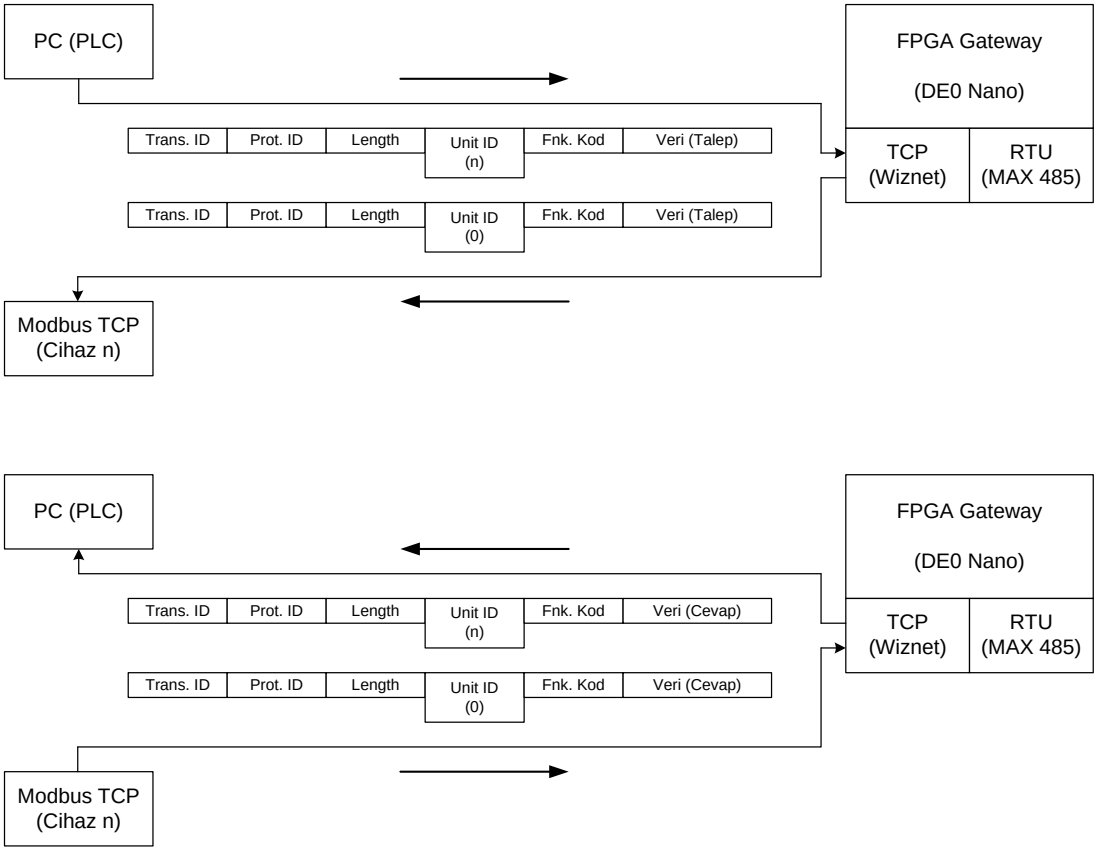
FPGA içinde tasarlanan gömülü işlemci Quartus programlama platformu içinde bulunan "*Nios II Software Built for Eclipse*" aracı ile C dili kullanılarak programlanır. Programın baş kısmında gerekli kütüphane tanımlaması yapılır. Program çalışmaya başlamadan önce Wiznet modülünün resetlenmesi gerekmektedir. Bunun amacı Modülün IP adresinin programcı tarafından belirlenebilmesini sağlamak ve işlemci içersinde önceden kalan verileri sıfırlamaktır. Bu aşamadan sonra cihazlarının konfigürasyon ayarları yapılır. Konfigürasyon ayarları daha önceden DE0 Nano üzerinde bulunan EEPROM'a gömülmüştür ve I2C modülü ile veriler buradan okunarak cihazlara gönderilir. Bu aşamadan sonra program istemci cihazdan talep bekler. Sistem altı adet TCP cihazı ve altı adet RTU cihazına göre hazırlanmıştır. Bu sayılar uygulamaya göre arttırılabilmektedir. Cihaz adreslemesinde birden altıya kadar olan cihazlar TCP cihazlarına ve yediden on ikiye kadar olan cihazlar RTU cihazlarını adreslemektedir. İstemci cihaz TCP ağı üzerinden işlemciye bağlanır ve gönderdiği talep doğrultusunda TCP ya da RTU cihazlara talebi işlemci tarafında yeniden düzenlenerek gönderilir.

Operasyon programının nasıl çalıştığını ifade eden blok diyagramı Şekil 4.13'de gösterilmiştir.



Şekil 4.13 : FPGA tabanlı ağ geçidi cihazın akış şeması.

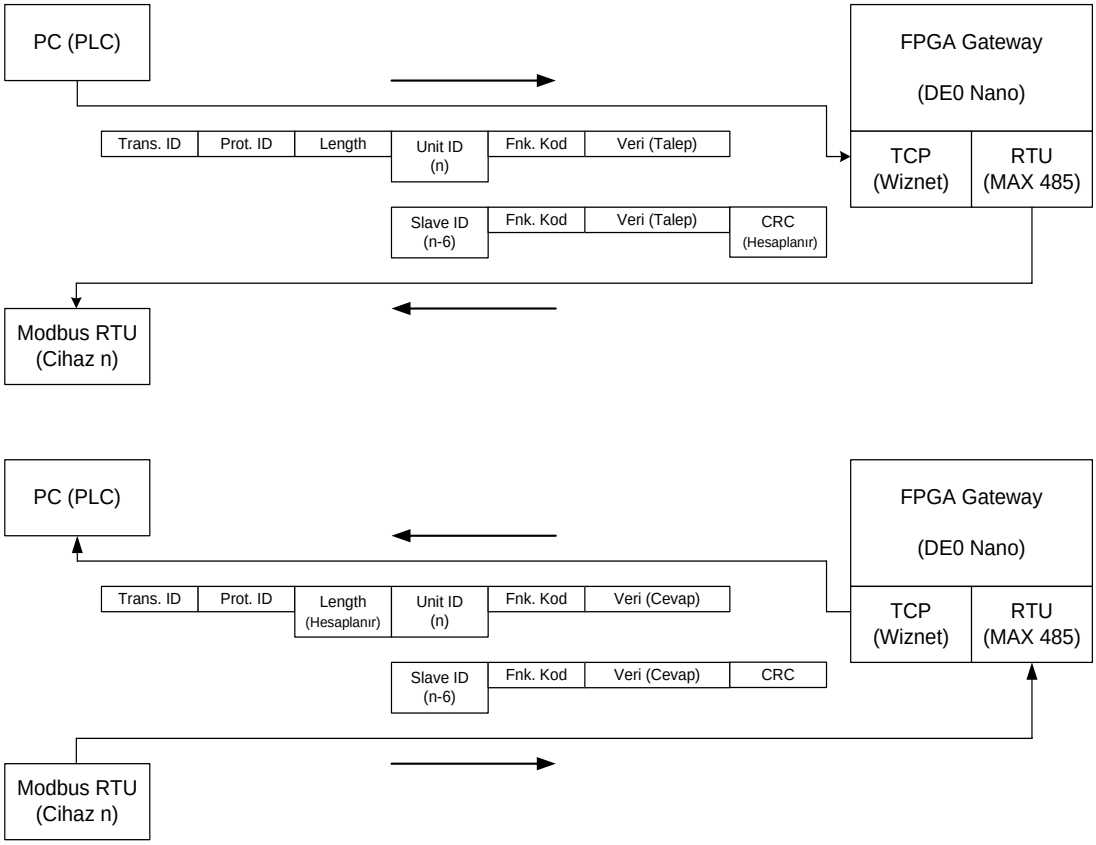
Eğer istemci cihaz bir ağ geçidi üzerinden bir TCP cihazına talep göndermek istiyorsa talep gönderme ve alma işlemi Şekil 4.14'deki gibi gerçekleşir.



Şekil 4.14 : İstemci ve TCP cihaz arasında veri alış verişi.

Şekil 4.14'de görüldüğü gibi olan istemci cihaz (PLC), TCP ağı üzerinden talebi Wiznet modülü ile FPGA içersindeki SPI modülüne iletir ve buradan talep işlemciye gelir. İşlemci talebi yeniden düzenleyip ilgili TCP cihazına gönderir. Gelen Modbus TCP paketi içersinde bulunan *Birim Tanımlayıcı* (*Unit Identifier*) bölümü talebin hangi cihaza gönderileceğini ağ geçidi cihazına bildirir. Ağ geçidi talebi adreslenen cihazın istediği forma çevirir *Birim Tanımlayıcı* bölümüne '0' değeri yazarak Wiznet üzerinden cihaza gönderir. TCP cihazı talebi işler, istenilen fonksiyonu gerçekleştir, cevabı oluşturur ve Wiznet üzerinden FPGA'e gönderir. İşlemci yeniden cevabı düzenler ve istemci cihaza işlemin tamamlandığı ya da hatalı olduğuna dair bilgi verir. Böylece veri alış veriş işlemi tamamlanmış olur. Tasarlanan sistem tek yönlü çalışır yani ağ üzerinden sadece tek paket akışı gerçekleşir.

Eğer istemci cihaz bir ağ geçidi üzerinden bir RTU cihazına talep göndermek istiyorsa talep gönderme ve alma işlemi Şekil 4.15'deki gibi gerçekleşir.

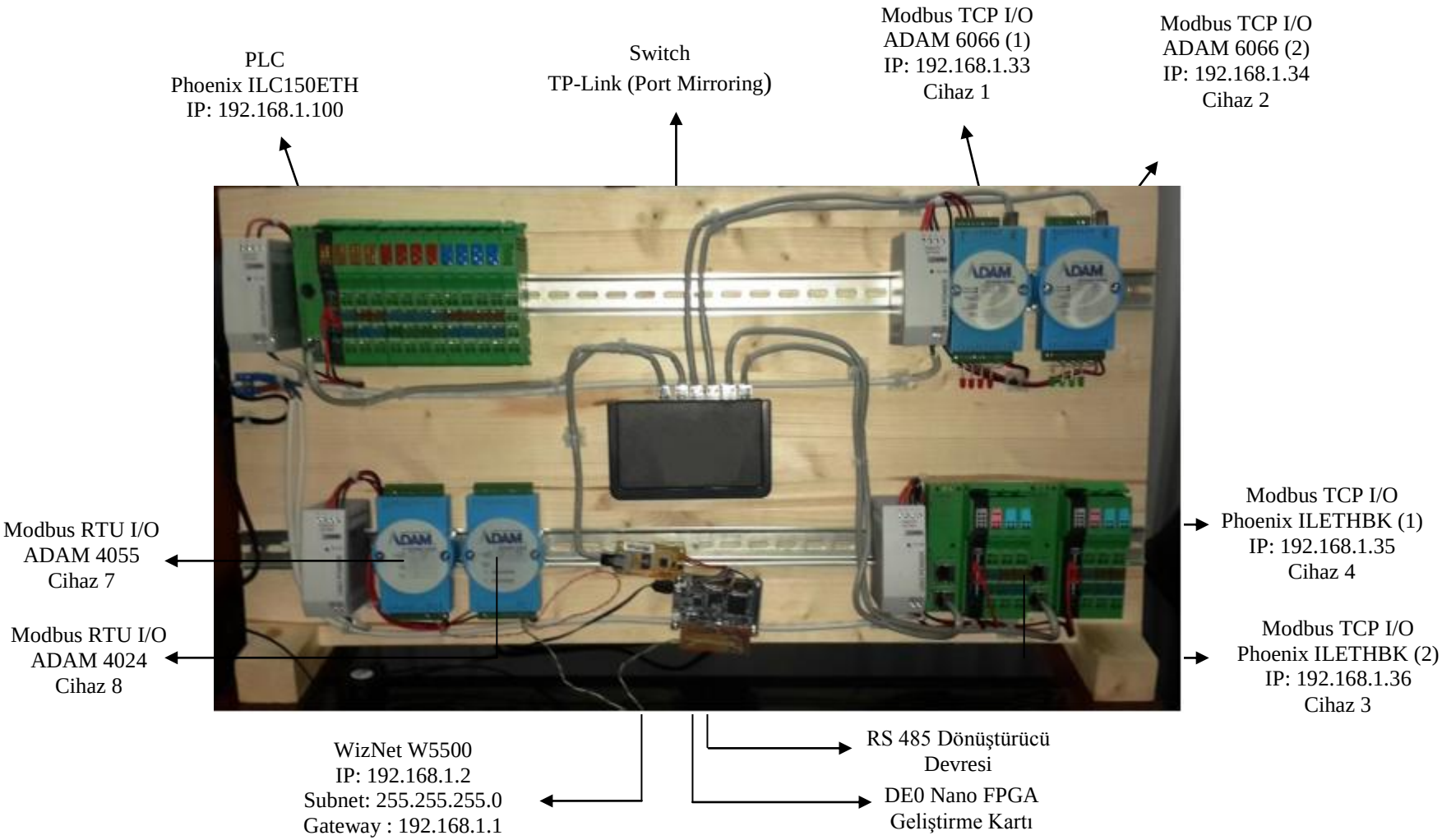


Şekil 4.15 : İstemci ve RTU cihaz arasında veri alışverişi.

Şekil 4.15'de görüldüğü gibi olan istemci cihaz (PLC), TCP ağı üzerinden talebi Wiznet modülü ile FPGA içersindeki SPI modülüne iletir ve buradan talep işlemciye gelir. İşlemci talebi yeniden düzenler ve ilgili RTU cihazına MAX 485 entegresi kullanılarak tasarlanmış olan RS485 dönüştürücü devre üzerinden ilgili RTU cihazına gönderir. Ağ geçidi cihazın içersinde Modbus RTU cihazlar yediden on ikiye kadar adreslenmesine rağmen, genel adresleme kuralına uyularak RTU cihazların adresleri birden başlatılmıştır. Bu nedenle RTU cihaza gönderilen mesajın Adres (Slave ID) bölümüne *n-6* yazılır. RTU cihazı talebi işler ve cevabı RTU modülü üzerinden FPGA'a gönderir. FPGA (işlemci) cevabı RTU'dan TCP formatına dönüştürür ve Wiznet üzerinden istemci cihazı bilgilendirir. Böylece veri alışverişi tamamlanmış olur.

4.4 Uygulama Platformu

Tasarlanan FPGA tabanlı Modbus ağ geçidi cihazı test etmek için Şekil 4.16'daki gibi bir test düzeneği hazırlanmıştır.



Şekil 4.16 : Uygulama test düzeneği.

Şekil 4.16'da verilen uygulama platformu incelendiği zaman, istemci cihaz görevini üstlenen PLC ağ geçidi üzerinden Modbus TCP protokolü ile TCP I/O cihazlarına ve Modbus RTU protokolü ile RTU I/O cihazlarına ulaşabilmektedir. Cihazların birbiri ile haberleşmesini gözlemleyebilmek için internet üzerinden ücretsiz sunulan *Wireshark* programı kullanılmıştır. *Wireshark* TCP/IP mesajlarını analiz eden bir programdır. Switch'in aynalama yapılacak portu ayarlanıp bu port bilgisayara bağlandıktan sonra *Wireshark* üzerinden paket akışı gözlemlenebilmektedir.

Sistemin doğru bir şekilde çalıştığını göstermek için PLC içersine Şekil 3.20'de ve 3.21'de gösterildiği gibi iki alt program yazılmıştır. Programda PLC, üzerinde bulunan çıkış ledlerini sıra ile açık ve kapalı durumu getirdikten sonra bu işlemi FPGA tabanlı ağ geçidi kullanılarak sürekli bir şekilde TCP ağı üzerindeki ve RTU ağı üzerindeki cihazlarında yapmasını talep etmektedir. PLC (istemci cihaz) bu talebi Cihaz 1'e gönderdiğinde Wireshark üzerinde Şekil 4.17'deki gibi bir veri akışı oluşmaktadır.

	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.1.2	192.168.1.100	Modbus/TCP	66	Response: Trans: 11667; Unit: 8, Func: 15: Write Multiple Coils
4	0.025692000	192.168.1.100	192.168.1.2	Modbus/TCP	66	Query: Trans: 11668; Unit: 1, Func: 5: Write Single Coil
5	0.026742000	192.168.1.2	192.168.1.33	Modbus/TCP	66	Query: Trans: 11668; Unit: 0, Func: 5: Write Single Coil
6	0.027503000	192.168.1.33	192.168.1.2	Modbus/TCP	66	Response: Trans: 11668; Unit: 0, Func: 5: Write Single Coil
7	0.028253000	192.168.1.2	192.168.1.100	Modbus/TCP	66	Response: Trans: 11668; Unit: 1, Func: 5: Write Single Coil
10	0.054021000	192.168.1.100	192.168.1.2	Modbus/TCP	66	Query: Trans: 11669; Unit: 1, Func: 5: Write Single Coil

Şekil 4.17 : Cihaz 1 için Modbus TCP veri akışı.

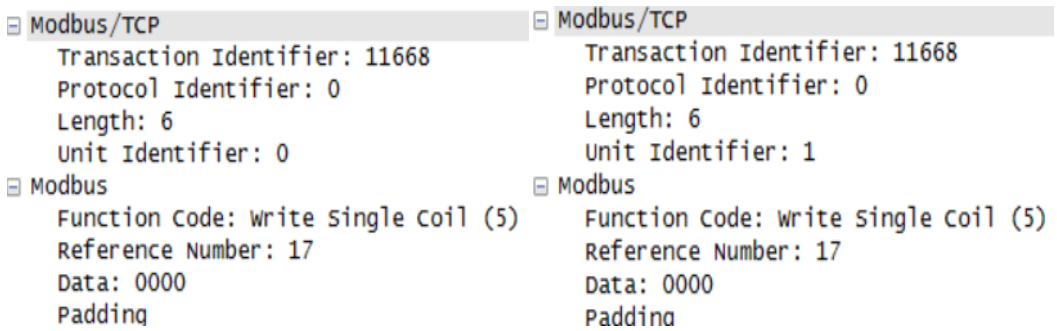
Şekil 4.18'de görüldüğü gibi 192.168.1.100 IP adresli PLC cihazı Modbus TCP talebini önce 192.168.1.2 IP adresli Wiznet modülüne gönderir. Wiznet modülü Nios işlemcisinde düzenlenen talebi, 192.168.1.33 IP adresli Cihaz 1'e (ADAM 6066) gönderir. Bu talepler ile ilgili bilgi Şekil 4.18'de gösterilmiştir. Bu veri haberleşmesi Fonksiyon Kod 5 yani tek çıkışa yazma (write single coil) kullanılmıştır.

```
Modbus/TCP
    Transaction Identifier: 11668
    Protocol Identifier: 0
    Length: 6
    Unit Identifier: 1
Modbus
    Function Code: Write Single Coil (5)
    Reference Number: 17
    Data: 0000
    Padding
```

```
Modbus/TCP
    Transaction Identifier: 11668
    Protocol Identifier: 0
    Length: 6
    Unit Identifier: 0
Modbus
    Function Code: Write Single Coil (5)
    Reference Number: 17
    Data: 0000
    Padding
```

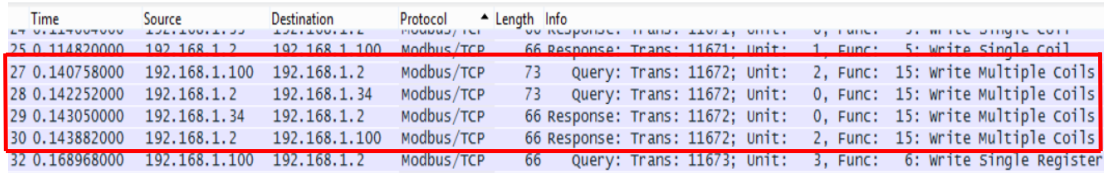
Şekil 4.18 : PLC'den Wiznet'e ve Wiznet'den Cihaz 1'e talep bilgisi.

Şekil 4.19'de Modbus/TCP bölümü altında verilen bilgiler MBAP başlık birimini ve Modbus bölümü altında verilen bilgiler Modbus protokol veri birimini tanımlar. Nios işlemcisi Modbus TCP talebini yeniden düzenleyerek Birim Tanımlayıcı (Unit Identifier) bölümünü 0 yaparak Cihaz 1'e gönderir. Cihaz 1 aldığı Modbus talebini işler ve cevabı oluşturarak Wiznet modulü üzerinden Nios işlemcisine gönderir. İşlemcide düzenlenen talep Wiznet üzerinden PLC'ye gönderir ve veri alış verişi tamamlanmış olur. Bu cevap ile ilgili bilgi Şekil 4.20'de gösterilmiştir.



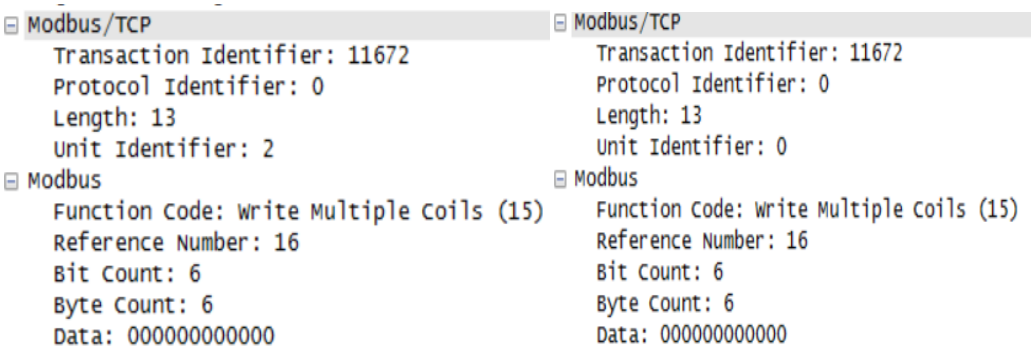
Şekil 4.19 : Cihaz 1 'den Wiznet'e ve Wiznet'ten PLC'ye cevap.

PLC'nin talebi Cihaz 2'ye gönderdiğinde Wireshark üzerinde Şekil 4.20'deki gibi bir veri akışı oluşmaktadır.



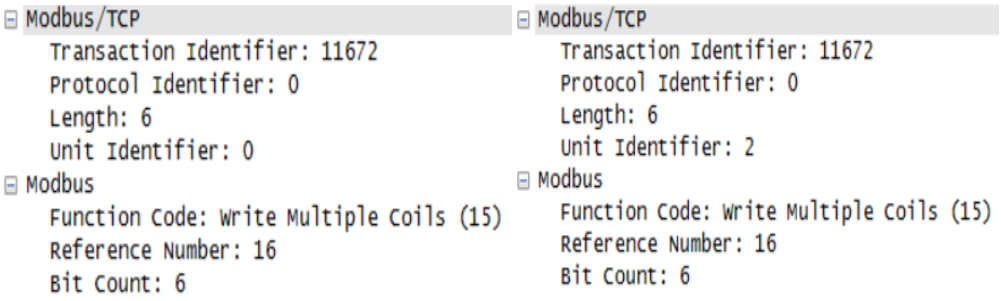
Şekil 4.20 : Cihaz 2 için Modbus TCP veri akışı.

Şekil 4.20'de görüldüğü gibi 192.168.1.100 IP adresli PLC cihazı Modbus TCP talebini önce 192.168.1.2 IP adresli Wiznet modulüne gönderir. Wiznet modulü Nios işlemcisinde düzenlenen talebi 192.168.1.34 IP adresli Cihaz 2'ye (ADAM 6066) gönderir. Bu talepler ile ilgili bilgi Şekil 4.21'de gösterilmiştir. Bu veri haberleşmesi Fonksiyon Kod 15 yani çoklu çıkışa yazma (write multiple coil) kullanılmıştır. Modbus/TCP bölümü altında verilen bilgiler MBAP başlık birimini ve Modbus bölümü altında verilen bilgiler Modbus protokol veri birimini tanımlar.



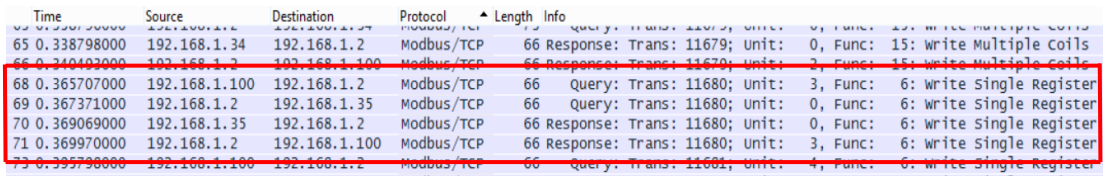
Şekil 4.21 : PLC'den Wiznet'e giden talep **ve** Wiznet'ten Cihaz 2'e giden talep.

Nios işlemcisi Modbus TCP talebini yeniden düzenleyerek Birim Tanımlayıcı (Unit Identifier) bölümünü 0 yaparak Cihaz 2'e gönderir. Cihaz 2 aldığı Modbus talebini işler ve cevabı oluşturarak Wiznet modulü üzerinden Nios işlemcisine gönderir. İşlemcide düzenlenen talep Wiznet üzerinden PLC'ye gönderir ve veri alış verişi tamamlanmış olur. Bu cevap ile ilgili bilgi Şekil 4.22'de gösterilmiştir.



Şekil 4.22 : Cihaz 2 'den Wiznet'e ve Wiznet'ten PLC'ye cevap.

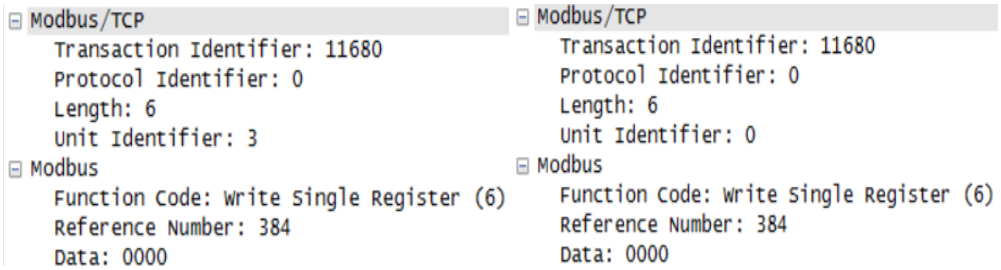
Cihaz 3 ve 4 için PLC aynı talebi göndermektedir. Bu cihazlara talep gönderilirken Fonksiyon kod 6 yani tekli kaydediciye yazma (write single register) kullanılmıştır. Cihaz 3 için (Cihaz 4 ile aynı) talep gönderdiğinde Wireshark üzerinde Şekil 4.23'deki gibi bir veri akışı oluşmaktadır.



Şekil 4.23 : Cihaz 3 & 4 için Modbus TCP veri akışı.

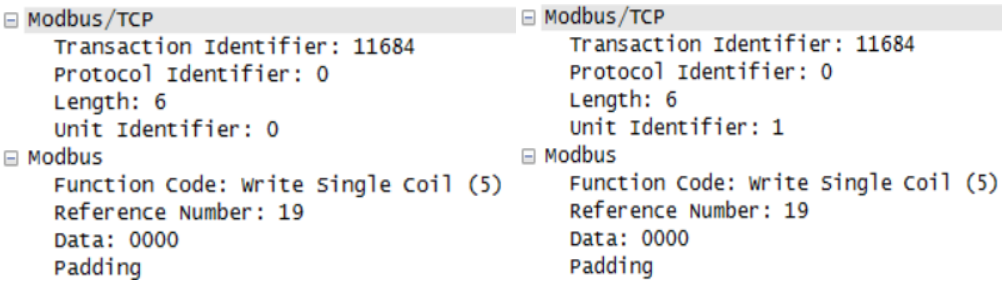
Şekil 4.23'de görüldüğü gibi 192.168.1.100 IP adresli PLC cihazı Modbus TCP talebini önce 192.168.1.2 IP adresli Wiznet modulüne gönderir. Wiznet modulü Nios

işlemcisinde düzenlenen talebi 192.168.1.35 IP adresli Cihaz 2'ye (Phoenix I/O) gönderir. Bu talepler ile ilgili bilgi Şekil 4.24'de gösterilmiştir.



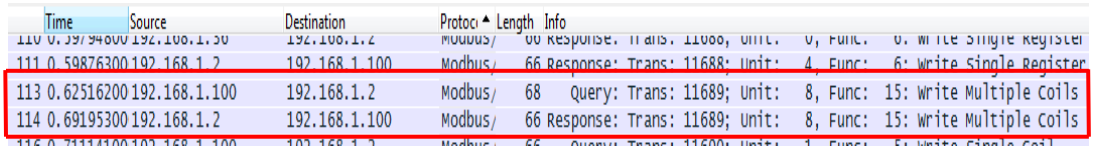
Şekil 4.24 : PLC'den Wiznet'e giden talep ve Wiznet'ten Cihaz 3'e giden talep.

Modbus/TCP bölümü altında verilen bilgiler MBAP başlık birimini ve Modbus bölümü altında verilen bilgiler Modbus protokol veri birimini tanımlar. Nios işlemcisi Modbus TCP talebini yeniden düzenleyerek Birim Tanımlayıcı (Unit Identifier) bölümünü 0 yaparak Cihaz 3'e gönderir. Cihaz 3 aldığı Modbus talebini işler ve cevabı oluşturarak Wiznet modulü üzerinden Nios işlemcisine gönderir. İşlemcide düzenlenen talep Wiznet üzerinden PLC'ye gönderir ve veri alış verişi tamamlanmış olur. Bu cevap ile ilgili bilgi Şekil 4.25'de gösterilmiştir.



Şekil 4.25 : Cihaz 3 'den Wiznet'e ve Wiznet'ten PLC'ye cevap.

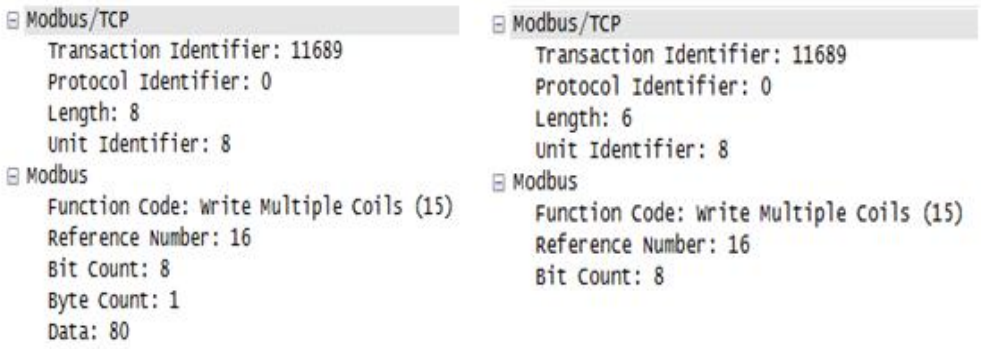
PLC'nin talebi Cihaz 7'ye (ADAM 4055) gönderdiğinde Wireshark üzerinde Şekil 4.26'deki gibi bir veri akışı oluşmaktadır.



Şekil 4.26 : Cihaz 7 için Modbus TCP veri akışı.

Cihaz 7, RTU ağı üzerindedir. PLC bu cihaz ile haberleşmek istediği zaman önce TCP ağı üzerinden Wiznet modülü ile işlemciye ulaşmakta ve işlemci TCP protokolünü RTU protokolüne dönüştürecek gerekli işlemleri tamamladıktan sonra

paket RTU dönüştürücü devresi üzerinden Cihaz 7'ye ulaşır. Cihaz paketi işler ve cevap oluşturur. Cevap paketi RTU ağı üzerinden işlemciye gelir, burada protokol dönüşümü işlemleri tamamlandıktan sonra Wiznet paketi TCP ağı üzerinden PLC!ye gönderir ve veri alış verişi tamamlanır. Şekil 4.26'da sadece TCP ağı üzerindeki paketler görülmektedir. Cihaz 7'nin istenilen işlemi yapması için Fonksiyon Kod 5 kullanılmıştır. Yani çoklu dijital çıkışlara yazma işlemi gerçekleşmiştir. Bu talep ve cevap ile ilgli bilgi Şekil 4.27'de gösterilmiştir.



Şekil 4.27 : PLC'den Wiznet'e giden talep ve Wiznet'ten PLC'ye giden cevap.

5. SONUÇLAR VE ÖNERİLER

Bu çalışmada endüstriyel sistemlerde en çok kullanılan protokollerden biri olan Modbus protokolünün iki alt versiyonu olan Modbus RTU ve Modbus TCP incelenmiştir. Modbus RTU protokolü Modbus verisini seri hat üzerinden iletirken Modbus TCP protokolü TCP Ethernet ağını kullanır. Ethernet alt yapısının dünya genelinde yaygınlaşması nedeniyle yeni endüstriyel cihazlar bu altyapıyı desteleyecek şekilde üretilmektedirler. Bu durumda RTU cihazları TCP altyapısına uyumlu şekilde çalışmasını sağlayacak dönüştürücü modüllere ihtiyaç duyar. Bu çalışmada Modbus TCP ağı üzerinde çalışan istemci cihazın hem TCP hem de RTU cihazlar ile tek bir ağ üzerinden haberleşmesini sağlama amacı ile bir ağ geçidi tasarımı yapılmıştır. Tasarım için son yıllarda kullanım olanı oldukça artan FPGA birimi kullanılmıştır. FPGA uygulamaya göre yeniden programlanması ve gerekli durumlarda donanım tasarımını desteklemesi ve içersinde gömülü işlemci tasarımı yapılabilmesi nedeniyle bu tasarım için uygun görülmüştür. FPGA ile birçok karmaşık devre ile tasarlanıp entegre işleminde uzaklaşarak çoğu işlem tek bir birim içersinde gerçekleştirilir. Bu da sistemlerin karmaşıklığını ve maliyetini azaltır.

Sistemde TCP ağından gelen verileri Modbus OSI modelinin uygulama katmanı için hazır hale getiren Wiznet modülü kullanılmış ve RTU ağına fiziksel katmanda veri gönderebilmek için RS 485 dönüştürücü devresi tasarlanmıştır. Wiznet modülü ile TCP ağından gelen veriler FPGA içersinde tasarlanan SPI modülü ile FPGA içersine gömülen işlemciye aktarılmaktadır. Gelen veri burada yeniden düzenlenerek ilgili TCP ya da RTU ağına gönderilmektedir. Program hazırlanan uygulama düzeneğinde test edilmiş ve güvenilir, kararlı ve düzenli veri akışı sağladığı görülüştür. Test sonuçları bölüm 4.5'de gösterilmiştir.

Bu çalışma ile Modbus protokolünü ayrıntılı olarak inceleyen Türkçe bir kaynak oluşturulmuş ve Modbus ağ geçidi tasarımı için ileri seviyede protokol bilgisi verilmiştir. Ayrıca FPGA kullanılarak gömülü sistem tasarımının nasıl yapılacağına

dair temel seviyede bilgiler verilmiştir. Modbus RTU kullanan cihazların TCP ağına uyarlanabilmesi sağlanmıştır.

Gelecek çalışmalarda, sistemin kontrolünün daha kolay sağlanabilmesi için sisteme HMI ya da bazı görsel ek birimler eklenebilir v sistem SCADA ağına bağlanabilir. Ayrıca Wiznet ile sağlanan TCP yönetimi için gerekli olan TCP/IP yığınları FPGA içersinde donanımsal olarak tasarlanabilir. Program en çok kullanılan fonksiyonları destekleyecek şekilde tasarlanmıştır. Bir sonraki çalışmalarda bütün Modbus fonksiyonlarını ve hata kodlarını destekleyecek üst sürümler hazırlanabilir. Ayrıca endüstriyel sistemlerde kullanılan CanBus ve Profibus gibi protokol ağlarını destekleyen daha geniş kapsamlı bir ağ geçidi tasarlanabilir.

KAYNAKLAR

[1] **Modbus Application Protocol Specification V1.1b3.,** Modbus.org. Alındığı tarih: 05.12.2014, adres: http://modbus.org/specs.php

[2] **Modbus over Serial Line Specificaiton and Implementation guide V1.02,** Modbus.org. Alındığı tarih: 05.12.2014, adres: http://modbus.org/specs.php

[3] **Modbus Mesaging on TCP/IP Implementation Guide V1.0a,** Modbus.org. Alındığı tarih: 05.12.2014, adres: http://modbus.org/specs.php

[4] **Technical Reference - Modbus TCP/IP,** Alındığı tarih: 05.12.2014, adres: http://www.prosoft-technology.com/

[5] **Peng, D., Zhang, H., Yang, L. ve Li, H.** (2008). Design and Realization of Modbus Protocol Based on Embedded Linux System. In Embedded Software and Systems Symposia, 2008. ICESS Symposia'08. International Conference on (Sf. 275-280). IEEE.

[6] **Phan, R. W.** (2012) Authenticated Modbus Protocol for Critical Infrastructure Power Delivery, IEEE Transactions on, 27(3), 1687-1689.

[7] **Liao, G. Y., Chen, Y. J., Lu, W. C., & Cheng, T. C.** (2008). Toward Authenticating the Master in the Modbus Protocol. Power Delivery, IEEE Transactions on, 23(4), 2628-2629.

[8] **Xuehua, S., Min, L., Hesheng, W., Hong, W., & Fei, L. (2011).** The Solution of Hybrid Electric Vehicle Information System by Modbus Protocol. In Electric Information and Control Engineering (ICEICE), 2011 International Conference on (pp. 891-894). IEEE.

[9] **Xu, S., Pan, H**., **Ren, J. ve Su, J.** (2008). Design of the Modbus Communication Through Serial Port in QNX Operation. *International Colloquium on Computing, Communication, Control, and Management*, vol:2, DOI: 10.1109/CCCM.2008.271, Sf: 434-438

[10] **Dudak, J. H. ve Cicak, P**. (2010). Dudak, J., & Cicak, P. (2010, June). CPN model of the MODBUS protocol. *In Mechatronika, 2010 13th International Symposium.* Sf. 43-45. IEEE.

[11] **Chiculita, C., Liviu, E., & Cristea, M. L.** (2013). Towards Multi-port Modbus Gateway. *In Electrical and Electronics Engineering (ISEEE), 2013 4th International Symposium on.* Sf. 1-4.

[12] **Ucun L, Akbati O, Gansever G.** (2013) Ethernet Based Automation Network Systems Mobile Laboratory: A Case Study in Nonlinear Control of Water Tank System. *Turkish Journal of Electrical Engineering & Computer Sciences* (Accepted).

[13] **Goldenberg, N., & Wool, A.** (2013). Accurate Modeling of Modbus/TCP for Intrusion Detection in SCADA Systems. *International Journal of Critical Infrastructure Protection*, 6(2), 63-75.

[14] **Hui, L., Hao, Z., & Daogang, P.** (2012). Design and Application of Communication Gateway of EPA and MODBUS on Electric Power System. Energy Procedia, 17, 286-292.

[15] **Süzer, E. S.**. (2006). Uzaktan Sayaç Okuma Teknikleri ve Modbus RTU, IEC61107 Mod C Protokolleri ile Örnek Yazılım, Türkiye, (yüksek lisans tezi), ITU, İstanbul

[16] **Demir, B**. (2012). Endüstriel Ağlar için Linux Tabanlı 3G Haberleşme Özellikli Modbus Mesaj Önceliklendirme Mekanizmalı Modbus Gateway Tasarımı, Türkiye, (yüksek lisans tezi), GYTE, Gebze.

[17] **Kaya, S**. (2010). Endüstriyel Otomasyon Sistemlerinde Saha Veri Yolu Teknolojisi, Türkiye, (yüksek lisans tezi), MU, İstanbul

[18] **Türkan, O**. (2008). Development of a Wireless Transmission Module for Modbus Applications, Türkiye, (yüksek lisans tezi), AU, Ankara

[19] **Görmemiş, M**. (2006). Dağıtım Şebekelerinde Enerji Kalitesi Ölçümlerinde Haberleşme Uygulaması, Türkiye, (yüksek lisans tezi), KSU, Kahraman Maraş.

[20] **Wargh, J**. (2013). Timestamping over Modbus TCP from Siemens S7, Finlandiya, (yüksek lisans tezi), Novia University of Applied Sciences, Vaasa

[21] **Klein, M**. (2011). Industrial Ethernet - Challenges and Drawbacks Comparison of Modbus TCP/IP and Ethernet IP, Avusturya, (lisans tezi), Viana University of Technology.

[22] **Sahraei, M**. R. (2013). Ns-Modbus: Integration of Modbus with ns-3 Network Simulator. Kanada, (yüksek lisans tezi), Simon Fraser University, Burnaby

[23] **Akkaya, Ş**. R. (2014). Sahada Programlanabilir Kapı Dizileri (FPGA) Tabanlo Robot Kontrolü. (yüksek lisans tezi), YTU, İstanbul.

[24] **DE0 Nano User Manual v1.9,** altera.com. Alındığı tarih: 06.12.2014, adres: http:// www.altera.com

[25] **Pong P. Chu.** (2011), Embedded SOPC Design with Nios II Processor and VHDL Examples, Clevand State University, John Wiley & Sons, Inc.

[26] **Wiznet W5500 Datasheed v1.0.4.** wiznet.co.kr. . Alındığı tarih: 06.12.2014, adres: http://www.wiznet.co.kr

[27] **Adam-6000 Series** - Ethernet Based Data Acquasion and Control Modules, User Manual. advantech.com. Alındığı Tarih: 12.10.2014, adres: http://www.advantech.com

[28] **Adam - 4000 Series** User Manual, advantech.com. Alındığı Tarih: 12.10.2014, adres: http://www.advantech.com

[29] **IL ETH BK DI8 DO4 2TX-PAC -** 2703981 User Manual, phoenixcontact.com. Alındığı Tarih: 12.10.2014, adres: https://www.phoenixcontact.com

[30] **TL-SG108E 8-Port Gigabit Easy Smart Switch,** User Manual, tp-link.com Alındığı Tarih: 12.10.2014, adres: http://www.tp-link.com

[31] **ILC 150 ETH - 2985330** User Manual, phoenixcontact.com Alındığı Tarih: 12.10.2014, adres: https://www.phoenixcontact.com

[32] **MAX 485 Datasheed.** maximintegrated.com Alındığı Tarih: 12.10.2014, adres: http://datasheets.maximintegrated.com

EK A

Tasarlanan FPGA tabanlı Modbus ağ geçidi cihazın, "*Nios II Software Built for Eclipse*" aracı kulllanılarak oluşturulan C kodları teze eklenmemiştir. Gerektiği takdirde sakkaya@itu.edu.tr adresinden istenebilir.

ÖZGEÇMİŞ

Ad Soyad : **Şirin AKKAYA**

Doğum Yeri ve Tarihi : **Giresun 03.05.1988**

E-Posta : **sakkaya@itu.edu.tr**

ÖĞRENİM DURUMU:

- **Lisans** : 2010, Kocaeli Üniversitesi, Mühendislik Fakültesi, Mekatronik Mühendisliği Bölümü
- **Yükseklisans** : 2014, Yıldız Teknik Üniversitesi, Kontrol ve Otomasyon Mühendisliği Anabilim Dalı
- **Yükseklisans** : 2015, İstanbul Teknik Üniversitesi, Mekatronik Mühendisliği Anabilim Dalı

DİĞER YAYINLAR, SUNUMLAR VE PATENTLER:

- **Akkaya Ş.,** Akbatı O. Görgün H., 2014. Multiple Close Loop Control with Digital PID Controllor Using FPGA, *International Conference on Control, Decision and Information Technologies, (Codit'14), Nowember 3-5,2014 Metz, France*