T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

UZAK BİLGİSAYAR KONTROLLÜ ROBOT KOL

H. ORHUN TAŞKAYA

YÜKSEK LİSANS TEZİ
ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ ANABİLİM DALI
ELEKTRONİK PROGRAMI

DANIŞMAN
YRD. DOÇ. DR. LALE ÖZYILMAZ

İSTANBUL, 2012

T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

UZAK BİLGİSAYAR KONTROLLÜ ROBOT KOL

H. Orhun TAŞKAYA tarafından hazırlanan tez çalışması 08/10/2012 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Yrd. Doç. Dr. Lale ÖZYILMAZ

Yıldız Teknik Üniversitesi

Jüri Üyeleri

Yrd. Doç. Dr. Lale ÖZYILMAZ

Yıldız Teknik Üniversitesi _____

Prof. Dr. Tülay YILDIRIM

Yıldız Teknik Üniversitesi _____

Yrd. Doç. Dr. M. Elif KARSLIGİL

Yıldız Teknik Üniversitesi _____

ÖNSÖZ

Sanayi alanında robot kollarla yaşanan gelişmeler, hem kalite açısından yükselmelere; hem de zaman ve maliyetler bakımından azalmalara imkan vermiş ve vermeye de devam etmektedir. Sanayide kullanımının getirdiği bu özellikler, robot kolların farklı uygulama alanlarında da kullanımının önünü açmış ve bu yönde gelişmelere neden olmuştur. Yaşanan bu gelişmeler göz önünde bulundurularak robot kol projesi gerçekleştirilmiştir.

Hazırladığım bu tez çalışmasında, yardımlarını hiçbir zaman esirgemeyen tez danışmanım Yrd. Doç. Dr. Lale ÖZYILMAZ'a, ayrıca öğrenim hayatım boyunca bana emeği geçen tüm hocalarıma teşekkürü bir borç bilirim.

Ayrıca bugünlere gelmemde en büyük paya sahip olan, beni her konuda daima destekleyen aileme ve tez sürecinde sürekli yanımda olan eşime de teşekkür ederim.

Ağustos, 2012

H. Orhun TAŞKAYA

İÇİNDEKİLER

Sayfa

KISALTMA LİSTESİ

ADC	Analog to Digital Converter
ARP	Address Resolution Protocol
ATM	Asynchronous Transfer Mode
CISC	Complex Instruction Set Computing
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSMA	Carrier Sense Multiple Access
DC	Direct Current
DHCP	Dynamic Host Configuration Protocol
DMA	Direct Memory Access
DNS	Domain Name System
FTP	File Transfer Protocol
GPIO	General Purpose Input/Output
G/Ç	Giriş/Çıkış
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IGMP	Internet Group Management Protocol
IP	Internet Protocole
I2C	Inter-Integrated Circuit
LAN	Local Area Network
LIN	Local Interconnect Network
LVD	Low Voltage Detection
MCU	Multipoint Control Unit
NMI	Non-Maskable Interrupt
OSPF	Open Shortest Path First
POP3	Post Office Protocol 3
POR	Power on Reset
PWM	Pulse Width Modulation
RARP	Reverse Address Resolution Protocol
RIP	Router Information Protocol
SFR	Special Feature Register
SMTP	Simple Mail Transfer Protocol
SPI	Serial Peripheral Interface
TCP	Transmission Control Protocol
TDST	Tibbo Device Server Toolkit
UART	Universal Asynchronous Receive Transmit
UDP	User Datagram Protocol
WAN	Wide Access Network

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖZET

UZAK BİLGİSAYAR KONTROLLÜ ROBOT KOL

H. Orhun TAŞKAYA

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Tez Danışmanı: Yrd. Doç. Dr. Lale ÖZYILMAZ

Endüstriyel üretim tesislerinde sıkça rastlanan robot işçileri ve robot kol mekanizmaları pek çok ağır ve tehlikeli işi en verimli şekilde yerine getirmektedir. Özellikle otomotiv endüstrisinde yoğun şekilde kullanılmalarının yanında tamamen robot işçilerden oluşmuş fabrikalar da mevcut durumdadır. Bunun yanısıra internet erişim ağının genişlemesi de uzak bağlantı sorunlarının ve zorluklarının önüne geçmiştir. Bu amaçla projede robot kolların uzaktan kontrol edilmesinin gerekli görüldüğü endüstriyel, bilimsel ya da eğlence amaçlı sistemler simüle edilmiştir.

Gerçekleştirilen Uzak Bilgisayar Kontrollü Robot Kol projesinde, tasarlanan robot kolun 4 eklemli iskelet yapısı 7 ayrı servo motor tarafından kontrol edilmektedir. Mikrodenetleyici üzerinden PWM sinyalleri ile yönlendirilen bu servo motorlar, Delphi programlama dili ile hazırlanan Uzak Bilgisayar Kontrollü Robot Kol (RCRA) arayüz programı ile bilgisayar üzerinden klavye ve mouse aracılığıyla kontrol edilmektedir. Klavye ve mouse ile bilgisayar üzerinden eşzamanlı kontrolün yanı sıra iki farklı demo hareketi ile önceden tanımlanan bu iki farklı görevin komutlarının robot kol tarafından sırasıyla yapılması sağlanmıştır.

Bilgisayar üzerinden kontrolde uzak bağlantıyı da desteklemesi için seri-Ethernet dönüştürücü kullanıldığından dolayı internet ve yerel ağ üzerinden robot kola istenilen

komutları göndermek mümkün olmaktadır. Bu sayede robot kola internet olan herhangi bir yerden erişim ve kontrol de sağlanmaktadır. Erişim güvenliği açısından arayüz programına şifre koruması dahil edilmiştir. Bu sayede yetkisiz kullanıcıların robot kola erişimi de engellenmiş olmaktadır.

Projede, robot kolun motorlarını sürmek ve arayüz aygıtları ile bağlantı kurmak için mikrodenetleyici olarak Renesas R8C/25 ailesinden R5F21256SNFP mikrodenetleyicisi kullanılmıştır. Bu mikrodenetleyiciye ilişkin donanım yapısı ve özellikleri hakkında bilgi verilmiş ve C dilinde hazırlanmış mikrodenetleyici program kodu da eklenmiştir. Ayrıca Delphi programlama dilinde hazırlanan arayüz programının, menü ve sayfa işlevleri ile programa ait yazılım kodu da verilmiştir. Mekanik kısımda ise robot kolun hareketini sağlamak amacıyla Hitec marka servo motorlar kullanılmıştır. Servo motorların sürme ve çalışma prensiplerine de ayrıca değinilmiştir.

Anahtar Kelimeler: Robot kol, uzak kontrol, mikrodenetleyici, servo motor

ABSTRACT

REMOTE COMPUTER CONTROLLED ROBOT ARM

H. Orhun TASKAYA

Department of Electronics and Communications Engineering

MSc. Thesis

Advisor: Assist. Prof. Dr. Lale OZYILMAZ

Robot and robot arm mechanisms works so effective at harmful and dangerous industrial production. Especially at automotive industry, there are lots of factory only with robot workers. In addition to this, evolution at internet access network solved remote access problems. As a reason, this project simulates industrial, scientific and entertainment systems that requires remote controlled robot arm.

Remote Computer Controlled Robot Arm project designed 4-jointed skeletal structure controlled by 7 separate servo motor. These motors driven by PWM signals from the microcontroller, is controlled by Remote Computer Controlled Robot Arm (RCRA) interface program designed in Delphi programming language via computer by keyboard and mouse. It is possible to send commands to the robot arm over local network and internet due to using of computer serial-Ethernet converter. Password for access security is included in the program interface. By this way, unauthorized users access to the robot arm is blocked.

In the project of the robot arm, R5F21256SNFP from Renesas R8C/25 microcontroller family microcontroller is use to drive motors and connect to interface devices. Information about the hardware structure and characteristics of this microcontroller is given and microcontroller program code in C language added. In addition to this,

Delphi programming language interface program menu and the page functions are also added with the software code of the program. Hitec brand servo motors are use for in order to provide the mechanical part and the movement of the robot arm. Working principles of the servo motors are also mentioned.

Key words: Robot arm, remote control, microcontroller, servo motor

BÖLÜM 1

GİRİŞ

1.1 Literatür Özeti

Robotlar bir yazılım aracılığı ile yönetilen ve yararlı bir amaç için iş ve değer üreten karmaşık makinelerdir. Ayrıca robotlar bilgisayar kontrollü, programlanabilir, fonksiyonel sistemler olarak da tanımlanabilirler.

Çekoslovakça'da "zorla çalıştırılan işçi" anlamına gelen "robota"dan dilimize geçen robot sözcüğünü ilk kez Karel Çapek adlı yazar 1921 yılında yazdığı RUR (Rossum's Universal Robots – Rossum'un Evrensel Robotları) adlı tiyatro oyununda kullanmıştır [1].

Robot kollar hakkında günümüze kadar birçok faklı çalışmanın yapıldığı bilinmektedir. Robot teknolojisinin gelişimi özellikle 20. yy'dan itibaren hızını artırsa da aslında robot teknolojisinin tarihi MÖ 3000'li yıllara kadar dayanmaktadır. Bunların arasında Homeros'un İlyada adlı eseri ilk kanıtlarındandır. Yeniçağa gelindiğinde ise Leonardo Da Vinci 'nin tasarımları ilk akla gelen örneklerdendir.

Bu süreç içinde ilk sibernetikçi olarak kabul edilen Cezeri'nin yazdığı "Kitab-ül'-Camü Beyne'l-İlmi-i ve'l-amelen-Nafi' Fi Sınaati'l-Hiyel" adlı kitapta robot teknolojisi konusunda çok sayıda ve zamanına göre çok ileri öneri ve uygulamalar bulunmaktadır [2].

Programlanabilir robotların öncüleri olan robot tipi mekanik aygıtlar, 1700 yıllarından itibaren "otomata" olarak adlandırılmıştır. Halkın yoğun ilgisinden dolayı yetenekli birçok bilim adamı bu alanda çalışma yapmayı tercih etmiştir. Bunların arasında

1774’te Droz tarafından yapılan tarihteki en karmaşık otomatlardan biri, 1801'de Marie Jacquard nümerik olarak kontrol edilebilen ilk makinesi, 1805'de Maillardet’in geliştirdiği yaylarla çalışan, İngilizce ve Fransızca yazabilen otomatı sayılmaktadır [1].

20. yy’a gelindiğinde ise Londra'da elektrikle çalışan bir robot yapılmıştır. 1930'lu yıllarda uçak tasarımcıları uçaklar için otomatik robot pilotu tasarlamıştır. 1940'larda Westinghouse, yatay düzlemde tamamen bağımsız olarak hareket eden iki robot icat etmiştir. 1940’ların sonlarına doğru ise Oak Ridge ve Argonne Ulusal Laboratuarlarında radyoaktif malzemeleri işlemek için uzaktan kontrollü mekanik robotların araştırma programları başlatılmıştır[2]. Bu sistemler insan operatörler tarafından yapılan el ve kol hareketlerini yerine getirmişlerdir.

1953 yılında Grey Walter duyumsal uyarlamayla hareket eden ilk otonom robotu geliştirmiştir. Bu robot ufak noktasal ışık kaynaklarının yerleştirildiği karanlık bir odada ışık dedektörleri ile ışığı algılayıp, ışık şiddetine bağlı olarak ışık kaynağına doğru yönelmekte ve ışık kaynağından uzaklaşmaktaydı [3].

1959 yılında Devol ve Joseph F. Engelberger’in kurduğu Unimaton Inc. tarafından tanıtılan robot, ilk endüstriyel robotlara öncülük etmiştir [2]. Bu robotun özelliği, robot kol ile bilgisayarın uyumlu çalışması sonucu ortaya çıkan birçok değişik görevi otomatik olarak gerçekleştiren bir makine olmasıdır. 1960’larda bu cihazların esnekliklerinin duyusal geri beslemeyle önemli ölçüde geliştirilebileceği belirginleşmiştir.

1961'de MIT'ye bağlı Lincoln Laboratuvarlarında, dokunma algılayıcıları olan bir robotun kolu bir bilgisayara bağlanmıştır [4]. Burada yapılan deney ve gözlemlere dayanılarak sadece dokunma algılayıcılarının yetersiz kalması, görüntü işleme araştırmalarına daha fazla önem verilmesine yol açmıştır.

1962’de H. A. Ernst, bilgisayar kontrollü dokunma sensörlü mekanik bir elin gelişiminden bahsetmiştir [2]. MH–1 diye anılan bu cihaz, blokları algılayabilmekte ve bu blokları operatör yardımı olmaksızın istif yapabilmekteydi. Bu çalışma uyarlamalı davranış kapasiteli bir robotun ilk örneklerinden biridir. Aynı zaman dilimi içerisinde Tomovic ve Boni (1962), nesneyi hisseden ve bir motora iki kavrama kalıbından birini seçmesi için bir geri besleme sağlayan basınç sensörlü prototip el geliştirmişlerdir [2].

1968 ile 1972 yılları arasında Stanford Araştırma Enstitüsü'nde geliştirilen Shakey ismindeki bilgisayar kontrollü robot, iki yönde hareket edebilme ve hareketlerini kendi kendine tasarlayabilme özelliğine sahipti [5].

1973'te Richard Hohn, Cincinatti Milacron yapımı ilk iş robotu olan T3'ü geliştirip pazara çıkardı. T3'ün özelliği bilgisayar denetiminde çalışan ilk ticari iş robotu olmasıdır. Hemen hemen aynı zaman diliminde IBM'den Will ve Grossman (1975) bir bilgisayar kontrollü, dokunma ve kuvvet sensörlü robotu 20 parçalı bir daktilo üretiminde kullanmak üzere geliştirilmişlerdir.

1980'lerin başında araştırmacı Rodney Brooks "dünyanın en iyi modeli kendisidir" düşüncesini savunarak hiçbir planlama yapmadan sadece çevresindeki etkilere tepki gösterecek bir robotun daha başarılı olacağını savundu. Tasarımını tamamladığı robotun üzerinde 20'den fazla motor ve 100'den fazla algılayıcı bulunmaktaydı [6].

1990'lar ise robotların uzay araştırmalarında kullanılmak üzere tasarlandığı yıllar olarak bilinir. Bu dönemde Mars yüzeyinde araştırma yapması için üretilen Amber ve Antartika'da gök taşı avına çıkmak ve aydaki buzullarla ilgili araştırmalar için tasarlanan NASA'nın da desteklediği Nomad ilk akla gelenlerdir.

2000'lere gelindiğinde ise Honda'nın geliştirdiği, ismi 'Yenilikçi Hareketlilikte İleri Adım' anlamına gelen '**A**dvanced **S**tep in **I**nnovative **Mo**bility' kelimelerinin kısaltılmasından oluşan ASIMO humanoid robotu en popüler olanıdır. NASA tarafından uzaydaki astronotları ameliyat etmek için tasarlanan Da Vinci robotu da 2009 yılından beri ülkemizde yapılan ameliyatlarda kullanılmaktadır.

Son yıllarda özellikle Japonya ve Amerika'nın yaptığı robot araştırmaları endüstride önem kazanmıştır. Robotlar sadece bir üretim aracı olarak değil, bağımsız birer ürün olarak da ekonomik değer taşımaktadırlar. Ülkemizde de robot çalışmaları özellikle üniversite destekli olarak 2000'li yılların başında hız kazanıp uluslararası projeler başlatılmıştır. Bunun da dünyada oluşan yeni pazarda Türkiye'ye de bir yer bulmak açısından önemi büyüktür.

1.2 Tezin Amacı

Günümüz üretim tesislerinde sıkça rastlanan robot işçileri ve robot kol mekanizmaları pek çok ağır ve tehlikeli işi yerine getirmektedir. Tamamen robot işçilerden oluşmuş fabrikalar bile mevcut durumda ve özellikle otomotiv endüstrisinde yoğun olarak kullanılmaktadır. Projenin gerçekleştirilmesindeki amaç bu uygulamaların yanı sıra robot kolların uzaktan kontrol edilmesinin gerekli görüldüğü, bilimsel ya da eğlence amaçlı sistemler kurmaktır.

1.3 Hipotez

Endüstride yoğun şekilde kullanılmaya başlanan robot kollar, hem üretim kalitesini artırmakta hem de iş emniyeti konusunda yaşanılan sorunların önüne geçilmesini sağlamaktadır. Endüstri haricinde de robot kolların kullanımı, hem güvenlik açısından hem de kontrol kolaylığı getirmesi nedeniyle giderek yaygınlaşmaktadır. Tüm sistemlerde olduğu gibi yaygınlaşmanın en büyük şartı düşük maliyet ve yüksek güvenilirlik olmuştur. Gerçekleştirilen proje de robot kolların bir hazırlık aşaması niteliğindedir.

BÖLÜM 2

MİKRODENETLEYİCİLER

Çağımızda bilgi ve teknoloji hızla ilerlemekte ve hayatımızın her alanına girmiş bulunmaktadır. Karmaşık ve uzun zaman alan hesaplamaların çözümlenmesi için bilim dünyası sürekli bir arayış halindeydi. Alman bilim adamı Zuse 1936 yılında mekanik anahtarlı Z1 adında ilk bilgisayarı yaptıktan sonra 1939 yıllında manyetik röle ile çalışan Z2 bilgisayarını tasarlamıştır. 1942 yılında Lowa State Üniversitesinin profesörü John Atanasoff ile öğrencisi Clifford Berry ilk elektronik bilgisayar olan vakum tüpleriyle ABC'nin yapımını gerçekleştirmiştir. 1946 yılında ABD'de askeri amaç için düşünülen bomba izlerinin hesaplanmasında kullanılacak olan ENIAC yapılmıştır. Bu bilgisayarlar manyetik röle ve vakum tüplü olduklarından, fiziki olarak bir oda büyüklüğünde ve sadece özel amaçlı kullanılmaktaydı. 1948 yılında yarı iletkenlerin keşfi ve 1950 yılında transistörlerin kullanılmasıyla birlikte bilgisayarlar yeni bir boyuta taşınmıştı. Eskisine göre daha küçük ve az enerji harcayan bilgisayarlar üretilmeye başlanmıştı. 1970 yılında Intel firması bilgisayarın beyni sayılan mikroişlemciyi (CPU) tek bir entegre olarak tasarlamıştır [8]. Gelen istekler doğrultusunda sürekli geliştirilen mikroişlemciler, sadece bilgisayarlarda kullanılmayıp otomobil, telefon sistemleri, beyaz eşya, robotlar, müzik aletleri, güvenlik sistemleri ve endüstride kullanılmaktaydı. Bellek, giriş/çıkış birimleri ve işlemciden meydana gelen basit bir mikroişlemcili sistem daha sonraları günümüzde adı geçen elemanların tek bir entegre haline getirilmesiyle mikrodenetleyici adını almıştır. Mikroişlemcili sistemi meydana getiren birimlerin kırpılmış özelliklerinin mikrodenetleyici sistemde kullanılmasından dolayı maliyet düşmüş, programlanması kolaylaşmış ve dolayısıyla boyutları da küçülmüştür.

Mikrodenetleyiciler sürekli geliştirilmekte, özellikleri ve performansları da gün geçtik.e arttırılmaktadır. Mikrodenetleyiciler, bilindik kullanım yerlerinin yanında endüstride ve günlük yaşantımızda kendisine sürekli yeni kullanım alanları açmaktadır. Bir mikrodenetleyici özet olarak, kullanıldığı sistemin birçok özelliğini aynı anda gözleme (monitoring), ihtiyaç anında gerçek-zamanda cevap verme (real-time response) ve sistemi denetlemeden (control) sorumludur.

Bir mikrodenetleyici aşağıdaki elemanlardan oluşur.

Mikroişlemci: Ön belleğine yazılan programı işleterek istenilen çıkışlara yönlendiren birimdir. Mikroişlemci veya sayısal bilgisayarlar üç temel kısımdan oluşurlar (CPU-Merkezi işlem ünitesi, Giriş-Çıkışlar ve bellek).

Merkezi İşlem Birimi (CPU – Central Proccesing Unit) : Sistemin kalbidir. Birim hesapları yapmak ve verileri idare etmek için çalışır.

I/O (Giriş -Çıkış) : Sayısal analog ve özel fonksiyonlardan oluşur. Mikroişlemcinin dış dünya ile ilişkisini sağlar. Mikroişlemciye verilen ve işlemlerden alınan veriler bu hat üzerinden sağlanır.

Bellek : Mikroişlemcinin hafıza deposu olarak görev alır.

2.1 R8C/25 Mikrodenetleyici Ailesi

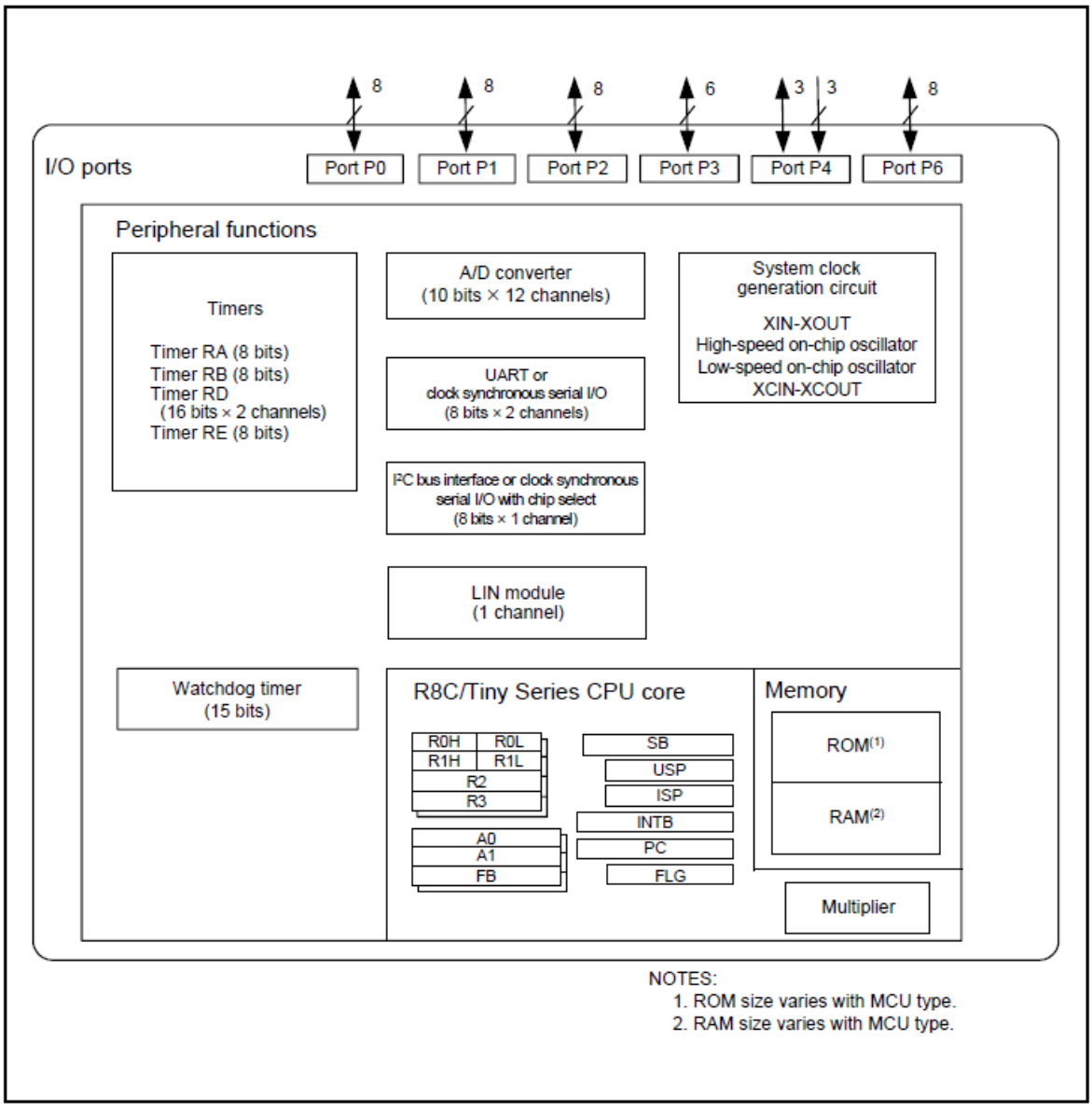
R8C ailesinin 16 bit CISC mikrodenetleyicilerinden biri olan R5F21256SNFP ürünü, yüksek ROM kod verimliliği, üstün parazit performansı, düşük güç tüketimi ve uygulamalar için yüksek işlem performansı gibi üstün özelliklere sahiptir. Kapsamlı zamanlayıcı fonksiyonları ve çeşitli seri iletişim fonksiyonları gibi on-chip çevre birimleri avantajı ile R8C ailesi çok çeşitli uygulama alanlarında destek verebilir. Rezonatör gerekliliğini ortadan kaldıran yüksek hızlı on-chip osilatörle entegre olan ürünler, besleme reset devresi (POR), düşük voltaj algılama devresi (LVD) veya sistem güvenliği desteklemek için bir osilatör çalışma detektörü ile düşük maliyetli olarak güvenli sistemlerin gelişmesine katkıda bulunmaktadır. Daha kısa geliştirme süreleri ve toplam sistem maliyetlerinde düşüşler için, Renesas düşük maliyetli geliştirme araçları yanı sıra on-chip flaş ürünler dizisi sunmaktadır. R8C ailesi, M16C ailesi ile uyumluluğa sahiptir; böylece yazılım kaynakları ve geliştirme araçları paylaşılabilir [9].

Kullanılan mikrodenetleyici Renesas firmasının R8C/25 serisinden 16 bitlik R5F21256FAHP'dir. Tezde servo motorların ve tüm çevresel birimlerin kontrolü, robot kol ile haberleşme ve bilgisayar arayüz programından komut alma/işleme mikrodenetleyici tarafından yapılmıştır. Yapılan geliştirmelerde C programlama dili kullanılmıştır.

2.1.1 R8C/25 Mikrodenetleyici Ailesi Genel Özellikleri

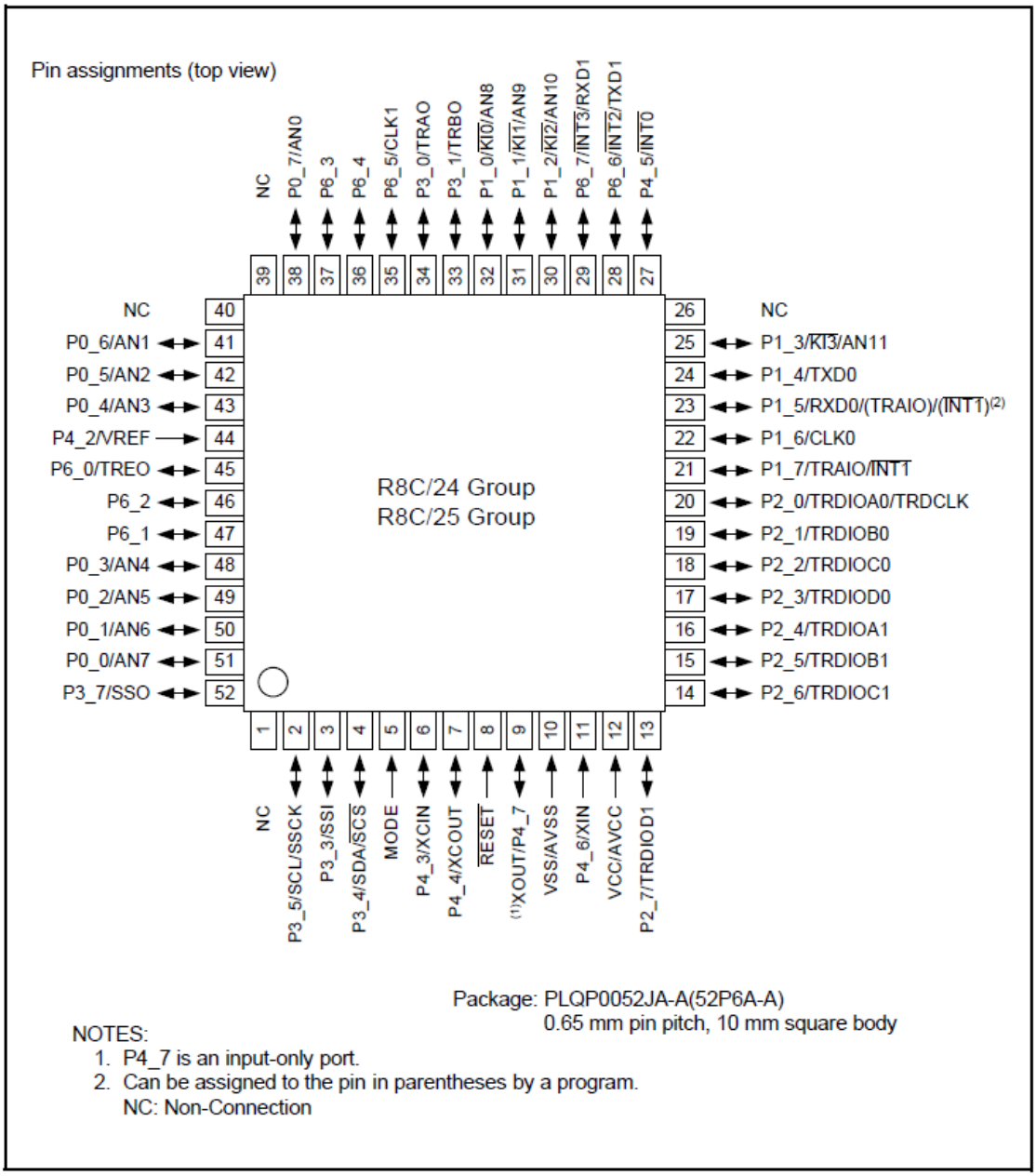
R8C/25 ailesi üyesi olan mikrodenetleyici, maksimum 20 MHz çalışma frekansına sahiptir. Uygulamada da 20 MHz'lik bir kristal kullanılmıştır. Mikrodenetleyicinin temel özelliklerini aşağıdaki gibi listelenebilir [10]:

- 8 bitlik çok fonksiyonlu 2 adet zamanlayıcı (Timer A ve Timer B).
- 16 bitlik giriş yakalama/çıkış karşılaştırma zamanlayıcısı. (PWM özellikli)
- 2 Kanal UART ve senkron seri arayüz
- 1 Kanal I2C data yolu
- 1 kanal LIN data yolu
- 12 Kanal 10 bit analog dijital çevirici
- Donanımsal CRC devresi
- 15 bitlik watchdog zamanlayıcı
- Saat darbesi üreteci devresi
- Genel amaçlı 41 giriş/çıkış pini; 3 adet giriş pini
- 11 kanal dahili; 5 kanal harici; 4 kanal yazılım kesmeleri
- Data flash alanı (2Kb*2 bank) dahili çalışma anında yazılabilir kalıcı bellek alanı
- Doğrudan bellek erişimi kanalları (DMA, Direct Memory Access)
- Düşük gerilim algılama devresi
- Power on reset devresi



Şekil 2.1 R8C/25 mikrodenetleyici ailesi iç yapısı

Kullanılan mikrodenetleyici UART, SPI, DMA, ADC, GPIO gibi birçok dahili çevresel birime sahiptir. Tüm çevresel birimlerin ayrı kontrol yazmaçları (registers) olmasına rağmen sınırlı sayıda bacak bulunduğu için her bir bacağın birden fazla çalışma modunda kullanımı mevcuttur. Portların hangi çevresel birim için kullanılacağı, çevresel birimlerin mod yazmaçları ile ayarlanıp belirlenir. Kullanılan mikrodenetleyicinin mevcut bacak bağlantılarını ve ortak kullanılan bacakları gösteren şekil aşağıdaki gibidir.

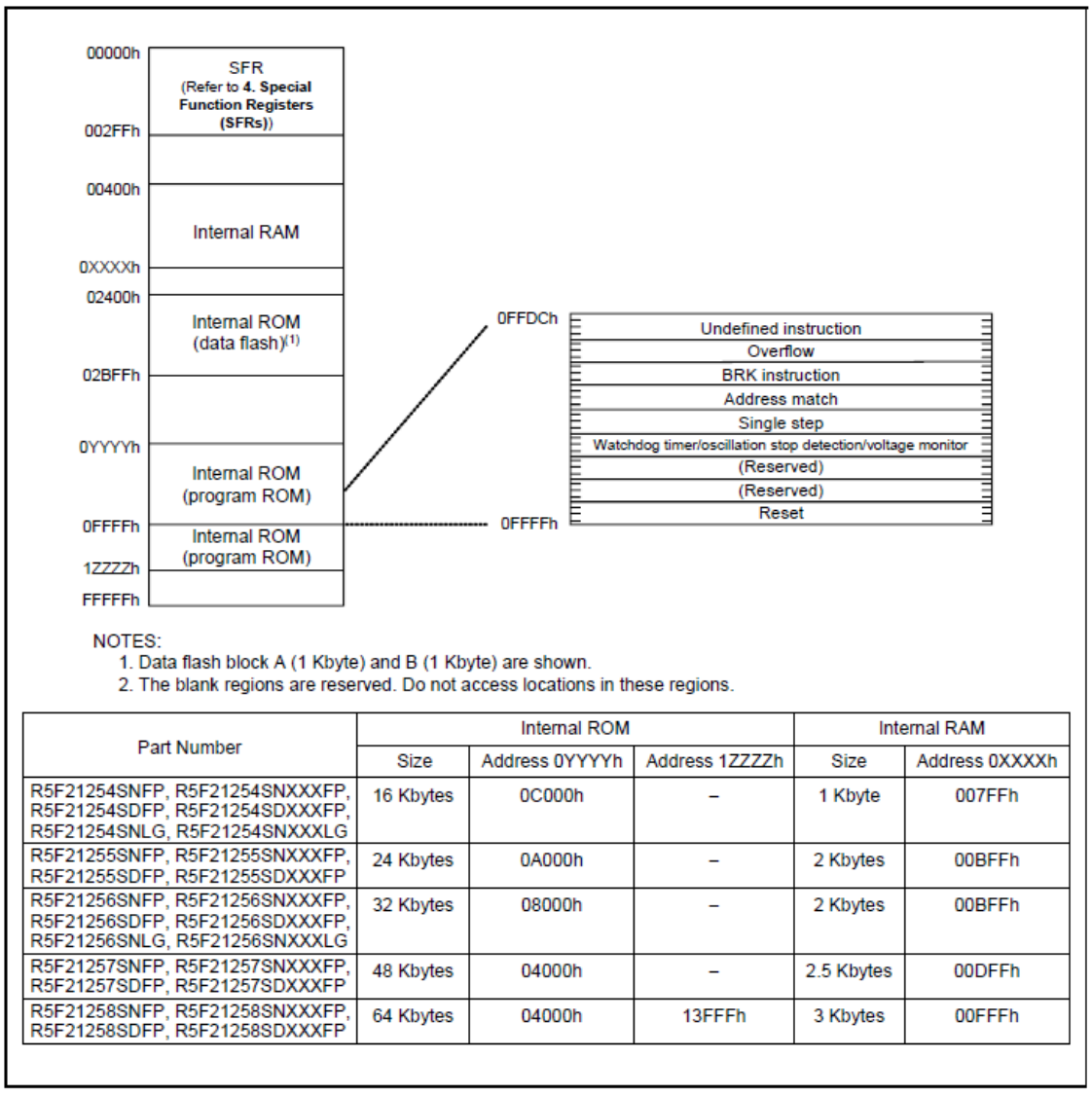


Şekil 2.2 R8C/25 mikrodenetleyici ailesi pin diyagramı

2.1.1.1 Bellek

R8C/25 grubu için 0x00000-0xFFFFF aralığında 1 MByte'lık adreslenebilir bellek alanı mevcuttur. Kullanılan mikrodenetleyici içerisinde yazılan kodların saklandığı kalıcı bellek olarak bir Flash alanı ve değişkenlerin tutulduğu; enerji kesildiğinde silinen bir RAM alanı mevcuttur. Bunlar dahili ROM (Internal ROM) ve dahili RAM (Internal RAM) alanları olarak tanımlanmıştır. Kullanılan mikrodenetleyici 32 KB ROM ve 2 KB RAM alanına sahiptir.

Program yükleme dışında dahili yazmaçların (register) bulunduğu bir SFR Alanı (Special Feature Register Area, Özel Tanımlı Yazmaç Alanı), çalışma esnasında veri yazılımına imkan sağlayan 2 KB'lık bir ROM alanı bulunmaktadır. Ayrıca Kesme Vektör Tablosu (Interrupt Vector Table) dahili ROM alanının son kısmında yer almaktadır [10]. Bellek yerleşimini gösteren şekil aşağıdaki gibidir.



NOTES:
1. Data flash block A (1 Kbyte) and B (1 Kbyte) are shown.
2. The blank regions are reserved. Do not access locations in these regions.

Part Number	Internal ROM			Internal RAM	
	Size	Address 0YYYYh	Address 1ZZZZh	Size	Address 0XXXXh
R5F21254SNFP, R5F21254SNXXXFP, R5F21254SDFP, R5F21254SDXXXFP, R5F21254SNLG, R5F21254SNXXXLG	16 Kbytes	0C000h	–	1 Kbyte	007FFh
R5F21255SNFP, R5F21255SNXXXFP, R5F21255SDFP, R5F21255SDXXXFP	24 Kbytes	0A000h	–	2 Kbytes	00BFFh
R5F21256SNFP, R5F21256SNXXXFP, R5F21256SDFP, R5F21256SDXXXFP, R5F21256SNLG, R5F21256SNXXXLG	32 Kbytes	08000h	–	2 Kbytes	00BFFh
R5F21257SNFP, R5F21257SNXXXFP, R5F21257SDFP, R5F21257SDXXXFP	48 Kbytes	04000h	–	2.5 Kbytes	00DFFh
R5F21258SNFP, R5F21258SNXXXFP, R5F21258SDFP, R5F21258SDXXXFP	64 Kbytes	04000h	13FFFh	3 Kbytes	00FFFh

Şekil 2.3 R8C/25 mikrodenetleyici ailesi bellek yapısı

2.1.1.2 Programlanabilir Giriş/Çıkışlar

Sayısal Giriş/Çıkış bütün kontrolörler için çok önemli bir özelliktir. Bu nedenle üretici firmalar sayısal G/Ç port sayısını mümkün olduğunca artırmaya çalışmaktadırlar. Her bir G/Ç hattı için yonga üzerinde bir ayak bulunmalıdır.

G/Ç hatları genellikle programlanabilirler. Programlanabilir G/Ç işlemlerinde portun giriş ya da çıkış olarak kullanılacağı yazılımla belirlenir. Bunun için her bir porta ait kontrol yazmaçlarının uygun şekilde ayarlanmaları gereklidir.

Projede kullanılan mikrodenetleyicide 41 adet programlanabilir G/Ç bulunmaktadır. Harici kristal kullanılması halinde 2 adet ve analog-dijital çevirici kullanılmadığında da 1 adet olmak üzere toplamda 44 adet giriş/çıkış pinine sahiptir[10].

Çizelge 2.1 R8C/25 mikrodenetleyici ailesi pin yapısı

Ports	I/O	Type of Output	I/O Setting	Internal Pull-Up Resister
P0 to P2, P6	I/O	CMOS3 State	Set per bit	Set every 4 bits(1)
P3_0, P3_1, P3_3 to P3_4, P3_5, P3_7	I/O	CMOS3 State	Set per bit	Set every 3 bits(1)
P4_3	I/O	CMOS3 State	Set per bit	Set every bit(1)
P4_4, P4_5	I/O	CMOS3 State	Set per bit	Set every 2 bits(1)
P4_2(2) P4_6, P4_7(3)	I	(No output function)	None	None

G/Ç pinlerinin yönleri PDi yazmaçları ile belirlenir. Bu pinlerin ayrıca dahili pull up dirençlerini devreye sokmak için kullanılan PURi yazmaçları da vardır. Bazı pinler birden fazla görev üstlenmiştir. Bu pinlerin görevlerini belirlemek üzere çeşitli yazmaçlar bulunur.

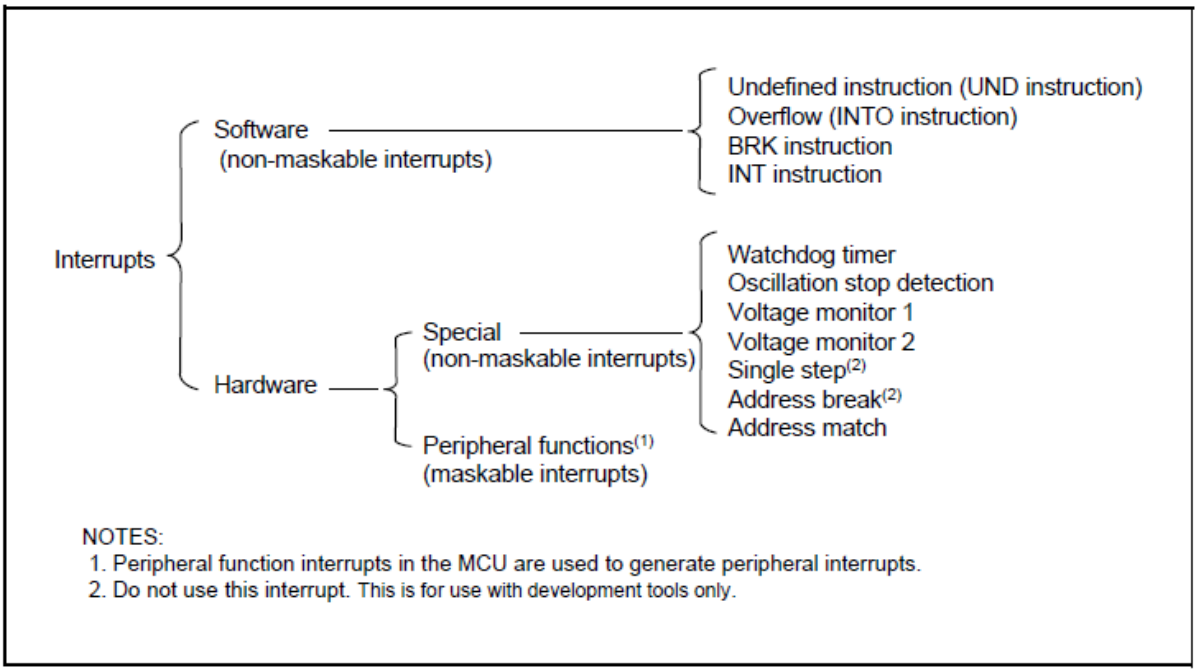
2.1.1.3 Kesmeler

Kesme, MCU'nun çalışmasını kesip özel bir program kodunun çalıştırılmasını sağlayan bir sinyaldir. Kesmeler dışarıdan gelebileceği gibi içeriden de alınabilirler. Kesme hizmet programına ait program parçasının ilk adresi bellekte belirli bir yerde tutulur. Buna kesme vektörü adı verilir.

Kesme hizmet programları genellikle küçük bir işi yerine getirmek için kullanılırlar. Kesme alındığında kesmeye gidilmeden önce geri dönüş adresi genellikle donanımla yığında saklanır. Eğer kesme hizmet programında kullanılacak yazmaçlar varsa bunlar da yığında saklanmalıdır. Kesme hizmet programına gidildiğinde kesme kendisini pasifleştirerek sürekli bir kesme alınmasını engeller. Kesme sona erdiğinde ise program kesmenin alındığı noktadan itibaren çalışacak şekilde tekrar hazır hale getirilir. Eğer yazmaçlar yığına atılmış ise doğru bir sırada geri çekilmeleri gereklidir. Kesmeden dönüşte aynı zamanda pasifleştirilmiş olan kesme tekrar etkin hale getirilir.

Kesmenin kullanılmaması halinde MCU pek çok yararlı iş yapabilecekken gereksiz yere bazı olayların yoklanması ile meşgul olacaktır. Aynı zamanda yoklama yazılımla yapıldığından çok kısa süreli darbelerin kaçırılması olasılığı da vardır.

Mikrodenetleyici içinde yazılımsal ve donanımsal kesmeler mevcuttur. Bu kesmeler maskelenebilir (kesmenin gelmesi durumunda ihmal edilmesi, önceliğinin değiştirilebilmesi ve kesme bayrağı ile çalışır duruma getirilip durdurulabilmesi) ve maskelenemeyen (bir kesme bayrağı ile açılıp kapatılamayan ve önceliği değiştirilemeyen) olarak gruplanabilir. Yapılarına göre gruplanmış kesme halleri aşağıdaki şekilde verilmiştir.



Şekil 2.4 Kesme tipleri

2.1.1.3.1 Yazılımsal Kesmeler

Yazılımsal kesmeler maskelenemez kesmelerdir. Tanımlanmamış bir komut işletiminde operatör işlemlerindeki taşmalar, özel birkaç komut ve özel INT operatörü kullanılarak tanımlanmış kesme fonksiyonları bu gruba girer.

2.1.1.3.2 Donanımsal Kesmeler

İki grupta toplanabilir. İlki NMI, Watchdog, düşük voltaj algılama özel amaçlı işlemcinin çalışması ile ilgili kesmeler; diğer grup ise mikrodenetleyici içerisindeki çevresel birimler için oluşturulmuş kesmelerdir. Seri port paketinin yollanması ile yollama

tamam kesmesi, seri porttan bir paket geldiğinde alma tamam kesmesi, bir zamanlayıcı kesmesinin içerisinde verilen hedef değere ulaşıp onu aşmasıyla oluşan kesme ya da düşen ya da yükselen bir kenarı yakalamak için kurulmuş bir kesme bu tipteki kesmelerdendir.

Kesmeler için vektör adreslerini ve yerleşimini gösteren çizelge "R8C/25 mikrodenetleyici ailesi kesme vektörleri çizelgesi" olarak aşağıda verilmiştir.

Çizelge 2.2 R8C/25 mikrodenetleyici ailesi kesme vektörleri çizelgesi

Interrupt Source	Vector Addresses(1) Address (L) to Address (H)	Software Interrupt Number	Interrupt Control Register	Reference
BRK instruction(3)	+0 to +3 (0000h to 0003h)	0	–	R8C/Tiny Series Software Manual
(Reserved)		1 to 2	–	–
(Reserved)		3 to 7	–	–
Timer RD (channel 0)	+32 to +35 (0020h to 0023h)	8	TRD0IC	14.3 Timer RD
Timer RD (channel 1)	+36 to +39 (0024h to 0027h)	9	TRD1IC	
Timer RE	+40 to +43 (0028h to 002Bh)	10	TREIC	14.4 Timer RE
(Reserved)		11 to 12	–	–
Key input	+52 to +55 (0034h to 0037h)	13	KUPIC	12.3 Key Input Interrupt
A/D	+56 to +59 (0038h to 003Bh)	14	ADIC	18. A/D Converter
Clock synchronous serial I/O with chip select / I²C bus interface(2)	+60 to +63 (003Ch to 003Fh)	15	SSUIC/IICIC	16.2 Clock Synchronous Serial I/O with Chip Select (SSU), 16.3 I²C bus Interface
(Reserved)		16	–	–
UART0 transmit	+68 to +71 (0044h to 0047h)	17	S0TIC	15. Serial Interface
UART0 receive	+72 to +75 (0048h to 004Bh)	18	S0RIC	
UART1 transmit	+76 to +79 (004Ch to 004Fh)	19	S1TIC	
UART1 receive	+80 to +83 (0050h to 0053h)	20	S1RIC	
$\overline{\text{INT2}}$	+84 to +87 (0054h to 0057h)	21	INT2IC	12.2 $\overline{\text{INT}}$ Interrupt
Timer RA	+88 to +91 (0058h to 005Bh)	22	TRAIC	14.1 Timer RA
(Reserved)		23	–	–
Timer RB	+96 to +99 (0060h to 0063h)	24	TRBIC	14.2 Timer RB
$\overline{\text{INT1}}$	+100 to +103 (0064h to 0067h)	25	INT1IC	12.2 $\overline{\text{INT}}$ Interrupt
$\overline{\text{INT3}}$	+104 to +107 (0068h to 006Bh)	26	INT3IC	
(Reserved)		27	–	–
(Reserved)		28	–	–
$\overline{\text{INT0}}$	+116 to +119 (0074h to 0077h)	29	INT0IC	12.2 $\overline{\text{INT}}$ Interrupt
(Reserved)		30	–	–
(Reserved)		31	–	–
Software interrupt(3)	+128 to +131 (0080h to 0083h) to +252 to +255 (00FCh to 00FFh)	32 to 63	–	R8C/Tiny Series Software Manual

2.1.1.4 Zamanlayıcılar

Zamanlayıcılar MCU'nun dahili saat işareti ile çalışır. Bu saat işareti ölçeklenerek farklı saat işaretleri ile de kullanılabilir (/2, /4, /16, /256...). Ayrıca taşma durumunda kesme üretmeleri için kontrol yazmaçları kullanılarak çeşitli ayarlar yapılabilir. Zamanlayıcılar belirli aralıklarla yapılması gerekli işlerin bulunduğu uygulamalarda hayati bir önem taşırlar.

Sayıcılar da zamanlayıcılarla aynı şekilde çalışırlar fakat sayıcılarda saat işareti yerine

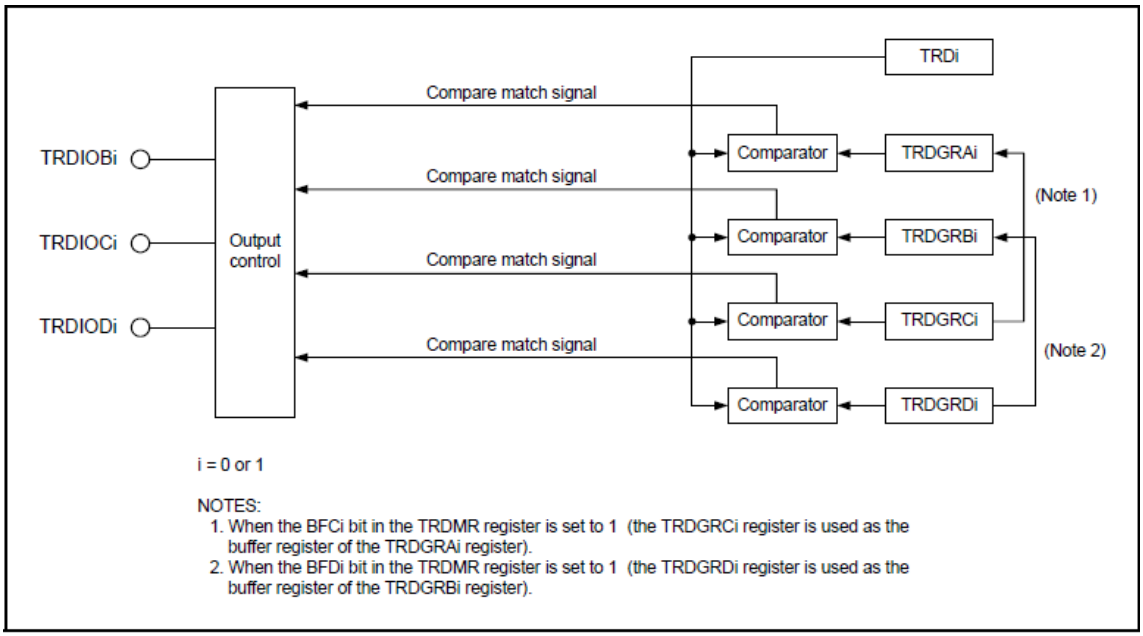
dışarıdan gelecek bir sinyal ile sayma işlemi gerçekleştirilir. Sayıcı içeriği kaç darbe alındığının belirlenmesi için yazılımla okunabilir. Çoğu MCU bir ya da daha fazla zamanlayıcı/sayıcıya sahiptir.

R8C/25 mikrodenetleyici ailesinde 2 adet 8 bit önoranlı 8 bit zamanlayıcı (RA ve RB); 2 adet 16 bit zamanlayıcı (RD) ve 1 adet de çıkış karşılaştırma özellikli 4 bit ve 8 bit sayıcı zamanlayıcı (RE) bulunur. Tüm zamanlayıcılar birbirlerinden bağımsız çalışabilmektedir [10].

2.1.1.4.1 PWM

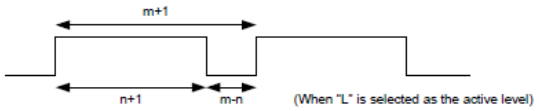
Renesas firmasına ait R8C/25 ailesi mikrodenetleyicileri dahili PWM modülüne sahiptir. Bu modülün modları Giriş Yakalama (Input Capture), Çıkış Karşılaştırma (Output Compare) ve PWM (PWM modu, Reset Senkron PWM Modu, Tümleyen PWM modu ve PWM3 modu) modu olarak adlandırılmıştır.

PWM modunda bir kanaldan aynı anda aynı periyotlu 3 PWM dalga şekli çıkışı kadar çıkış verilebilir. Ayrıca kanal 0 ve kanal 1 senkronize edilerek aynı periyotlu 6 dalga şekli çıkışı olabilir. Bu fonksiyonun kombinasyonları TRDIOji (i = 0 veya 1, j = B, C, veya D) ile belirlenir. Şekil 2.5'de PWM Modu blok diyagramı gösterilmiştir [11].



Şekil 2.5 R8C/25 mikrodenetleyici ailesi PWM modu blok diyagramı

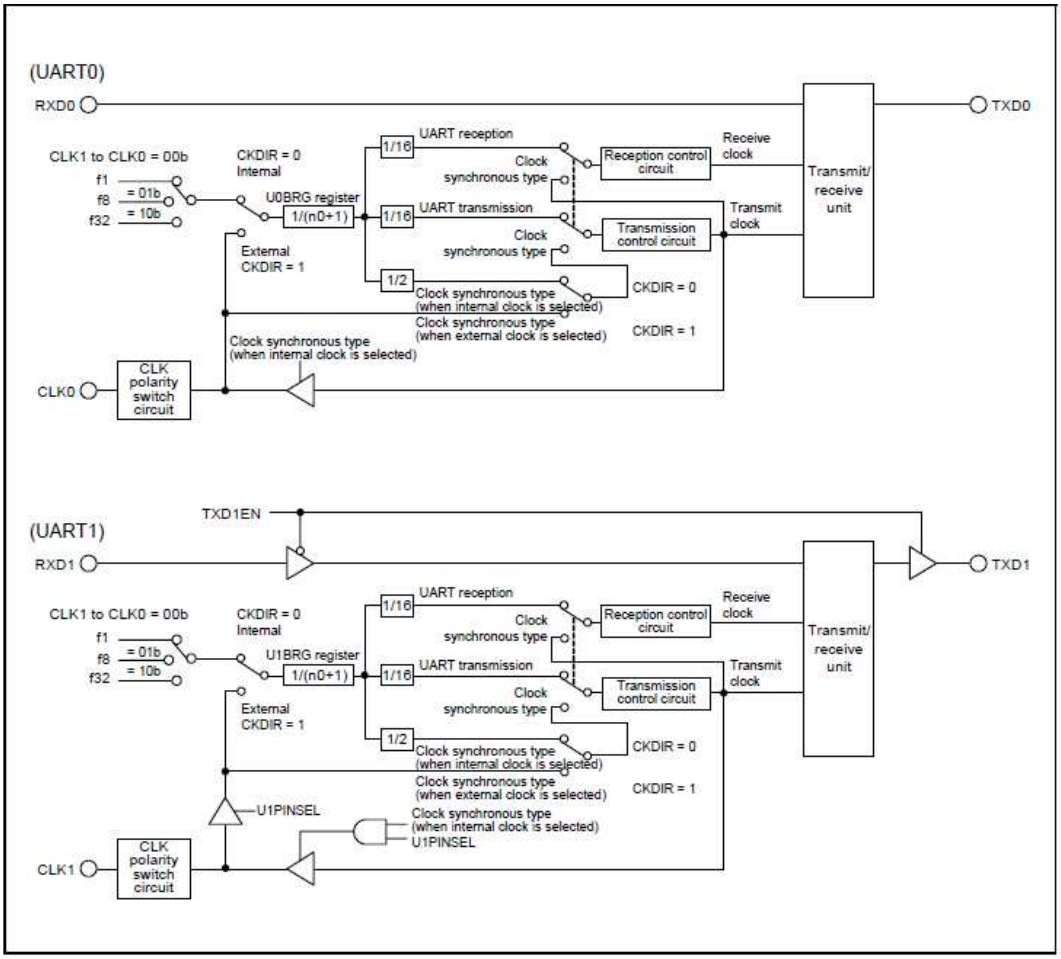
Çizelge 2.3 R8C/25 mikrodenetleyici ailesi PWM modu özellikleri

Item	Specification
Count sources	f1, f2, f4, f8, f32, fOCO40M External signal input to the TRDCLK pin (valid edge selected by a program)
Count operations	Increment
PWM waveform	PWM period: 1/fk x (m+1) Active level width: 1/fk x (m-n) Inactive level width: 1/fk x (n+1) fk: Frequency of count source m: Value set in the TRDGRAi (i = 0 or 1) register n: Value set in the TRDGRji (j = B, C, or D) register 
Count start condition	1 (count starts) is written to the TSTARTi bit in the TRDSTR register.
Count stop conditions	• 0 (count stops) is written to the TSTARTi bit in the TRDSTR register when the CSELi bit in the TRDSTR register is set to 1. • The PWM output pin holds output level before the count stops. • When the CSELi bit in the TRDSTR register is set to 0, the count stops at the same time as the TRDi register is set to 0000h at the compare match in the TRDGRAi register. • The PWM output pin holds level after output change by compare match.
Interrupt request generation timing	• Compare match (the content of the TRDi register matches the content of the TRDGRji register.) • TRDi register overflows
TRDIOA0 pin function	Programmable I/O port or TRDCLK (external clock) input
TRDIOA1 pin function	Programmable I/O port
TRDIOB0, TRDIOC0, TRDIOD0, TRDIOB1, TRDIOC1, TRDIOD1 pin functions	Programmable I/O port or pulse output (selectable by pin)
$\overline{\text{INT0}}$ pin function	Programmable I/O port, pulse output forced cutoff signal input, or $\overline{\text{INT0}}$ interrupt input
Read from timer	The count value can be read by reading the TRDi register.
Write to timer	The value can be written to the TRDi register.
Select functions	• One to three PWM output pins selected per channel • Either one pin or multiple pins of the TRDIOBi, TRDIOCi or TRDIODi pin. • The active level selected by pin. • Initial output level selected by pin. • Synchronous operation (refer to **3.4 Synchronous Operation**) • Buffer operation (refer to **3.3 Buffer Operation**) • Pulse output forced cutoff signal input (refer to **3.5 Pulse Output Forced Cutoff**)

PWM Modunda yazmaçlardan TRDGRAi (i = 0 veya 1) PWM periyodunu belirler. Her kanalın 3 farklı olan çıkışlarının değişim noktaları ise TRDGRji (j = B, C, D; i = 0, 1) yazmaçları ile belirlenir [11].

2.1.1.5 Seri Arayüz

Seri arayüz iki ayrı kanaldan oluşmaktadır (UART0 ve UART1). Her kanal farklı transfer saat darbesi üreten ve bağımsız çalışan özel zamanlayıcılara sahiptir. Bu kanallar senkron ve asenkron olarak iki farklı modda çalışabilirler.



Şekil 2.6 R8C/25 mikrodenetleyici ailesi UART arayüzü

2.1.1.5.1 UART (Asenkron Seri İletişim Arabirimi)

Standart iletişim protokollerini kullanarak işlemcinin dış dünya ile iletişimini sağlar. Bu özellik MCU'nun harici bir aygıt ile iletişim yapmasının gerektiği durumlarda kullanılır. Örneğin bir LCD monitör ya da başka bir bilgisayar ile UART üzerinden iletişim sağlanabilir. Bu şekilde bir arabirime sahip olmayan yongaya harici olarak SPI üzerinden bir UART eklenebilir ya da UART görevini yerine getirebilecek bir yazılım kullanılabilir.

2.1.1.5.2 SPI (Serial Peripheral Interface)

Harici sistemlerle iletişimde UART'ın kullanıldığı yerlerde SPI (Seri Çevre Arabirimi) diğer yongalarla iletişimde kullanılır. SPI portlarına sahip pek çok yonga çeşidi vardır. UART, A/D çeviriciler ve hatta bellek yongaları bunlardan bazılarıdır. Paralel iletişim kadar hızlı olmamasına rağmen SPI iletişiminin kullanımı kolaydır. Bütün adresleme ve veri hatlarının kullanılmasının yerine 3 ya da 4 hat ile iletişim sağlanabilir.

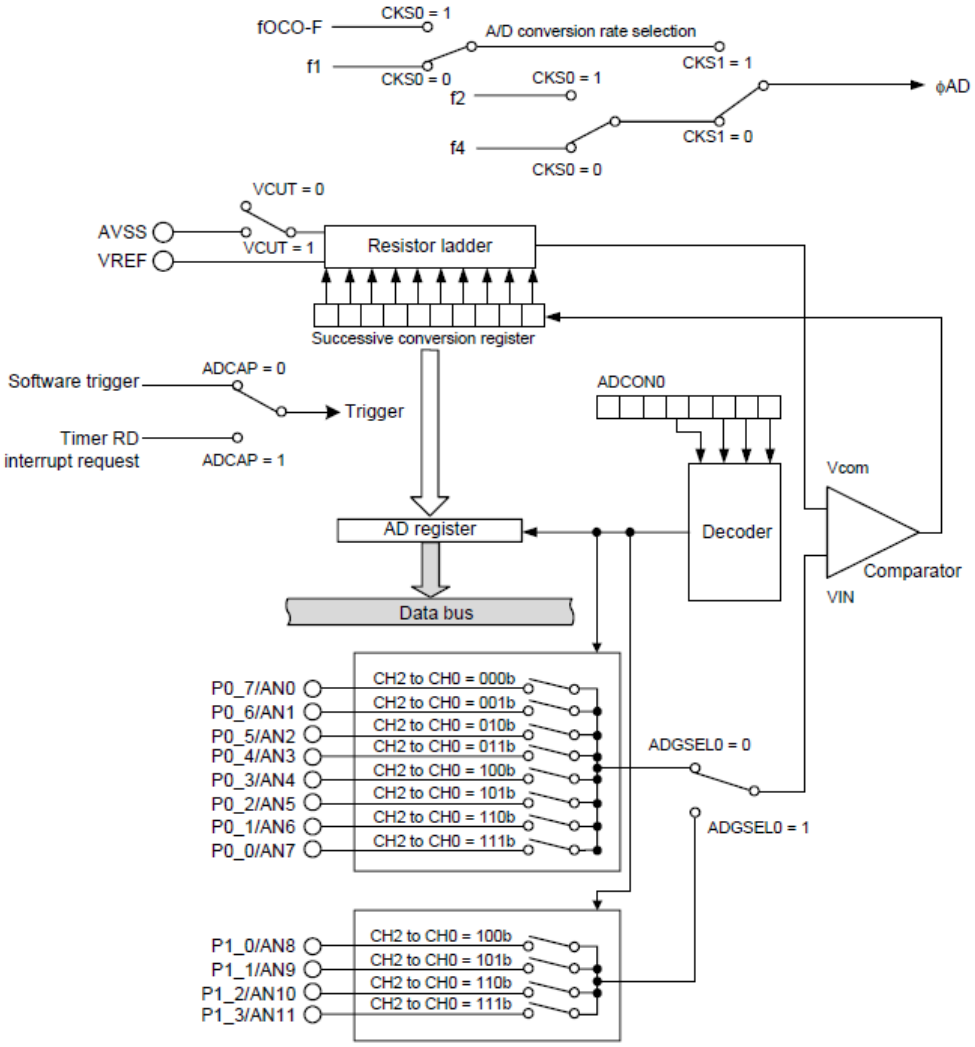
2.1.1.5.3 I2C (Inter Integrated Circuit)

Sadece iki hat kullanarak SPI ile aynı görevi yapar. I2C protokolü birden fazla aygıtın aynı anda iletişim yapabilmesine olanak sağlar. SPI'da ise aynı anda sadece iki aygıt iletişim halinde bulunabilir.

2.1.1.6 Analog Dijital Çevirici

Gerçek dünyadaki işaretler genellikle mikrodenetleyicilerin anlayamayacağı sürekli işaretlerden oluşur. Bu nedenle fiziksel dünyadaki analog değişimlerin izlenebilmesi için MCU'larda analog dijital çeviriciler (ADC) ya da analog karşılaştırıcılar bulunur.

R8C/25 ailesinde 8bit/10 bit çözünürlüğü ayarlanabilen 12 kanallı analog dijital çevirici bulunmaktadır.



Şekil 2.7 R8C/25 mikrodenetleyici ailesi ADC iç yapısı

2.1.1.7 Saat Kontrolü

Ana saat darbesi, mikroişlemcinin ana saat darbe osilatörü devresi tarafından üretilir. Reset emri geldiğinde, saat darbesi ayarlanan yazmaca göre bölünür. Ana saat darbe üretecinde stop biti gelirse darbe üretimi de durur. Saat darbe üretimi durdurularak, güç tüketimi en aza indirgenir.

2.1.1.8 Watchdog Zamanlayıcısı

Watchdog zamanlayıcısı, programın dışında yürütülen bir fonksiyondur. Watchdog zamanlayıcısı aslında iç zamanlayıcıyı kullanarak saat darbesi türeten 15 bitlik bir sayıcıdır.

Watchdog zamanlayıcının 7 tane biti kontrol biti olup bu bitler önoran bölme oranını belirler. Mikroişlemci bir watchdog zamanlayıcı kesmesi isteği bulunduğunda watchdog zamanlayıcı başlangıç yazmacı üzerine yazmaya başlar. Önoran sadece mikroişlemci resetlendiğinde devreye girer. Reset işlemi tamamlandıktan sonra watchdog zamanlayıcısı ve önoran durur.

2.1.1.9 Reset

Mikroişlemcili sistemlerde reset devreleri çok önemli bir yere sahiptir. G/Ç, program sayacı ve kontrol yazmaçları gibi belirli yazmaçlar için bilinen bir durumun varlığından mikroişlemcinin emin olmasını sağlar. Saatin doğru frekansa yerleşmesi için yeterli zamanın verilmesini ve son olarak da şebeke geriliminde bir sorun varsa, mikroişlemcinin düzgün olarak yeniden başlayacağından emin olunmasını sağlar.

BÖLÜM 3

SERVO MOTORLAR

Konum kontrolü uygulamalarında kullanılan motorlar genellikle adım motoru ve servo motorlardır. Adım motorlarının kullanım alanı küçük, güçlü ve düşük moment gerektiren kontrol sistemleri ile sınırlıdır. Buna karşılık büyük güç, yüksek moment ve hızlı tepki gerektiren sistemlerde ise daha çok servo motorlar kullanılır. Ayrıca servo motorların kalkış ve duruş anında motor kontrolü daha yumuşak bir şekilde gerçekleştirilebilmektedir. Motorun bu şekilde kontrol edilmesi sisteme iki şekilde fayda sağlar. Birincisi motor milinde bir yük olması durumunda motorun kalkışı ve duruşu sırasında mile bağlı yükün zarar görmesi önlenebilir. İkinci fayda ise motorun kalkışı sırasında yüksek akım çekmesinin önlenmesidir. Servo motorların çok çeşitli uygulamalarda kullanılmasının, güvenilir olmasının yanında diğer nedenleri ise;

- Yüksek tork
- Doğru konumlama
- Kolay kurulum
- Kontrol kolaylığı
- Ekonomik oluşudur.

Şekil 3.1 Servo motor dış görünümü

İngilizce “servant” (hizmetçi, uşak) kelimesinden gelen servo motor, bir DC motorun şaftının dişli sistemiyle bir potansiyometreye bağlanmasından oluşur. Servo motorlar, üzerindeki dahili kontrol devresi ile potansiyometredeki voltaj bölünmesine göre pozisyonu hakkında bilgi edinir ve gelen sinyalle kıyaslayıp doğru pozisyona döner. Servo motor açılı dönebildiği için çok geniş kullanım alanları vardır.

Servo motorlar, kontrol motorları olarak da adlandırılan elektrik motorları olup, özellikle geri beslemeli kontrol sistemlerinde çıkış hareketini kontrol edici olarak tasarlanmışlardır. Gömülü kontrol devrelerine sahiptirler ve birkaç watt ile birkaç yüz watt olacak şekilde farklı güçlerde üretilirler. Yüksek hız tepkisine sahip olmaları rotor ataletinin düşük olmasını gerektirir ve yapı olarak daha küçük çaplı ve daha uzundurlar. Servo motorların en yaygın kullanım alanlarına örnek olarak; robotlar, radarlar, bilgisayar malzemeleri, takım tezgahları, izleme ve yol gösterme sistemleri ve işlev denetleyiciler gösterilebilir [12].

Servo motorlar içlerinde DC motor, konum algılamak için potansiyometre, motor çıkışını güçlendirmek için redüktör ve istenilen açının sağlanması için kontrol entegresi barındırır ve 180 derece dönüş açısına sahiptirler. Modifiye edilmiş modelleri ise sürekli dönebilirler. Potansiyometre, servo motorun miline bağlıdır ve milin o anki açısını kontrol entegresine iletir. Kontrol entegresi ise motoru istenilen açıya gelene kadar döndürür.

Şekil 3.2 Servo motor iç görünümü

Servo motorlar model uçak, araba, tekne ve küçük güçteki robot uygulamalarında kullanılan motor çeşitlerindendir. İçerisinde bir DA motor, kod çözücü, motoru sürmekte kullanılan elektronik bir devre ve motor gücünü arttırmakta kullanılan plastik veya metal dişliler bulunur. Servo motorun elektronik sürücüsü kendi içerisinde bulunduğu ve fazla akım çekmediği için mikrodenetleyici ile direkt olarak sürülebilir. Genellikle metal ya da plastik dişli sistemine sahiptirler. Nadiren başka materyalden yapılmış dişli sistemine sahip servo motorlar da bulunmaktadır. (Karbonit dişli servolar) Her iki tip dişlinin de uygulama alanına göre artı ve eksileri vardır. En temel fark, metal dişlinin birbirine daha güçlü tutunması ve kırılmalara karşı plastik dişlilere oranla daha dayanıklı olmasıdır.

Servo motorların çalışma gerilimleri 4,8 - 6V civarındadır ve en çok kullanılan servo motorun gücü 1 cm/3.5 kg'dır. Görüldüğü gibi servo motorun miline bağlanacak kolun uzunluğu ile kolun gücü arasında ters orantı vardır.

Servo motorun devreye bağlantısında üç adet kablo kullanılır. Bu kablolardan ikisi enerji, biri sinyal girişi içindir. Artı (+) hat genelde kırmızı renk kablo ile; eksi (-) hat siyah veya kahverengi renkli kablo ile ve sinyal hattı sarı, turuncu veya beyaz renkli kablo ile iletilir.

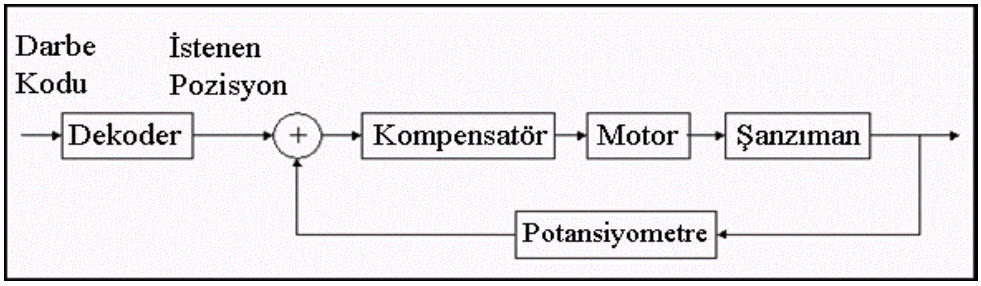
Servo motorun DC motoru, herhangi bir DC oyuncak motorundan farklı olmayan çift mıknatıslı bir statora ve fırçalı bobin rotora sahiptir. Motor mili 1:200 ile 1:300 arası

dönme oranına sahip bir dişli sistemine bağlanır, bu sayede oldukça yüksek bir tork değerine ulaşılır.

Dişli sisteminin çıkışında 5kΩ'luk bir potansiyometre, mil konumunu elektronik kumanda devresine iletir. Elektronik devrenin görevi, mil konumunu gelen veri konumuna gelinceye kadar motorun dönmesini sağlayıp tam yerinde durdurmaktır.

3.1 Servo Motor Temel Fonksiyonları

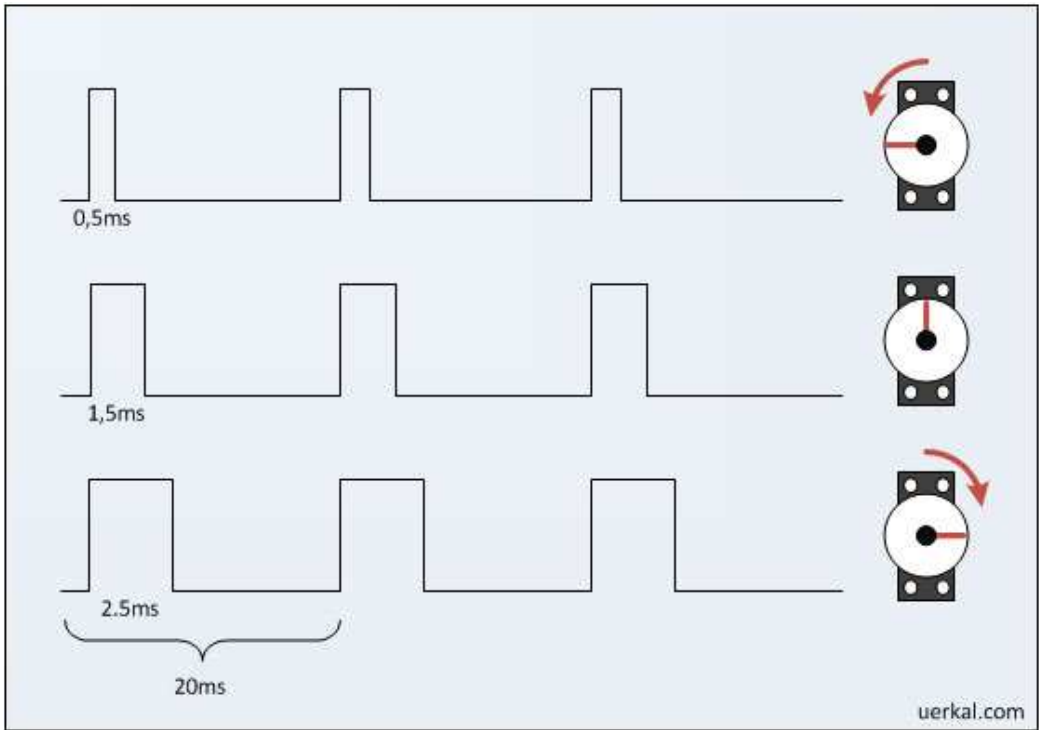
Servo motorun çalışma prensibi, gelen darbe koduna göre konum değiştirmektir. Aşağıda verilen blok diyagramı, servo motorun gerçekleştirdiği temel fonksiyonu çok iyi açıklamaktadır. İstenen konum ile şaftın pozisyonu karşılaştırılır. Kompensatör ise gelen bu bilgiyi düzenler ve giriş işareti olarak ayarlar. Motorun şanzımana bağlı olmasından dolayı çıkışta düşük hızda bile yüksek tork gücü elde edilir. Şafta bağlı olan potansiyometrenin görevi ise geri besleme sinyalini sağlamaktır.



Şekil 3.3 Servo motorun çalışma şeması

3.2 Servo Motor Kontrolü

Servo motorları kontrol etmek için motora PWM sinyali gönderilir. PWM, çoğu mikrodenetleyicinin donanımsal olarak desteklediği bir özelliktir. Genellikle saniyede 50 defa (50Hz) 0,5 ile 2,5 milisaniye arasında uygulanan sinyal ile motora dönmesi istenilen açı bildirilir.



Şekil 3.4 Periyoda bağlı olarak değişen servo motor hareketi

Servo motorların konumları PWM (Darbe Genişlik Modülasyonu) yöntemiyle kontrol edilirler. Şekil 3.4'de servo motora uygulanan PWM sinyalinin şekline göre servo motorun aldığı konum gösterilmektedir. Bu darbenin (sinyalin lojik 1 kısmı) süresi 0,5 ms ise servo motor en sola döner (0 derece), darbenin süresi 0,5 – 1,5 ms arasında artırılırsa servo motor ortaya doğru konumlanır (90 derece). Darbenin değeri 2,5 ms'ye doğru artırılırsa servo motor sağa doğru konumlanır (180 derece). Darbenin değeri sabit tutulup 20 ms'de bir tekrarlanırsa servo motor aynı konumda kalmaya devam edecektir. Servo motorun darbe değerleri markalarına göre de farklılık gösterebilir. Bazı markalarda tam sol için darbe değeri 1ms, orta konum için 1,5 ms ve tam sağ için 2 ms süreli darbe uygulanır. Darbe değeri bilinmeyen bir servo motorun süreleri ancak deneme yanılma yoluyla bulunabilir.

3.3 Darbe Genişlik Modülasyonu (Pulse Width Modulation – PWM)

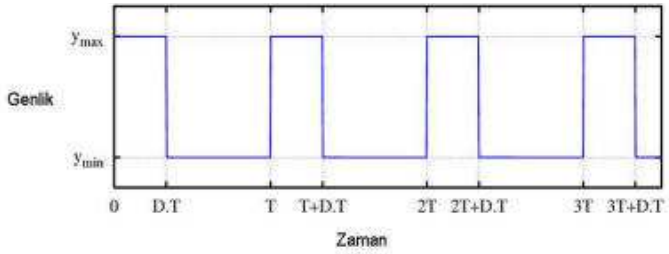
Darbe genişlik modülasyonu, üretilecek olan darbelerin genişliklerini kontrol ederek çıkışta üretilmek istenen analog elektriksel değerin veya sinyalin elde edilmesi tekniğidir.

PWM, elektrik ve elektronikte birçok alanda farklı amaçlar için kullanılmaktadır. Telekomünikasyon, güç, voltaj düzenleyiciler, ses üreteçleri veya yükselteçler gibi çeşitli uygulama alanları ve farklı uygulamaları bulunmaktadır.

3.3.1 Darbe Genişlik Modülasyonu Temelleri

Üretilen kare dalga darbe sinyallerinin genişliklerinin ortalaması, çıkışta üretilecek olan analog değerin elde edilmesini sağlar.

En iyi açıklama aşağıdaki şekil üzerinden yapılabilir.



Şekil 3.5 Darbe genişlik modülasyonu

Kare dalganın frekansına f(t), en düşük genlik değerine y_{min}, en yüksek genlik değerine y_{max} ve sinyal oranına (duty cycle) D diyelim, ortalama sinyal,

$$\bar{y} = \frac{1}{T}\int_0^T f(t)\,dt \qquad (1)$$

f(*t*) kare dalga olduğundan, *f*(*t*), y_{max} için $0 < t < D \cdot T$ ve y_{min} için $D \cdot T < t < T$ değerlerini alabilir.

Buradan,

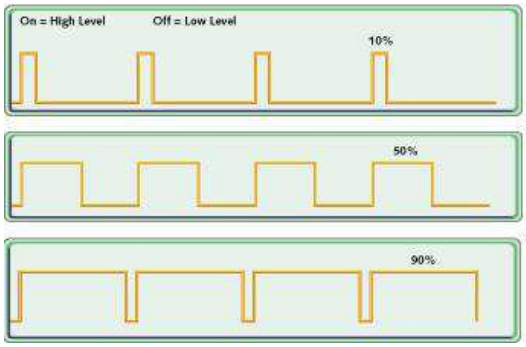
$$\begin{aligned} \bar{y} &= \frac{1}{T}\left(\int_0^{DT} y_{max}\,dt + \int_{DT}^{T} y_{min}\,dt\right) \\ &= \frac{D\cdot T\cdot y_{max} + T(1-D)y_{min}}{T} \\ &= D\cdot y_{max} + (1-D)\,y_{min} \end{aligned} \qquad (2)$$

elde edilir [13].

Yukarıda verilen formül, genellikle y_{min} = 0 iken $\bar{y} = D \cdot y_{max}$ olarak kullanılır. Görüldüğü gibi elde edilecek ortalama değer direkt sinyal oranına (duty cycle) bağlıdır.

3.3.2 Sayısal Kontrol

PWM tekniği, motor sürücülerinde, voltaj düzenleyicilerde, telekomünikasyonda kodlama ve çözme teknikleri gibi birçok alanda kullanılmaktadır.



Şekil 3.6 PWM darbe örneği

En üstte verilen grafik, Duty Cycle %10, ortadaki %50, alttaki ise %90 için çizilmiştir. Bu oranlarla üretilecek olan 5V sinyalin analog sinyal değeri, %10 için 0.5V, %50 için 2.5V ve %90 için 4,5V olacaktır.

Burada üretilen sinyalin frekansı da göz ardı edilmemelidir. Çok düşük bir frekans ile üretilen darbe sinyalleri ve bunlarla kontrol edilen bir anahtar ile kontrol edilen bir lambada, lambanın yanma ve sönme zamanları hissedilecektir. Bu durum ışığın şiddetinin değişikliğinden öte, titreme şeklinde görünecektir. Bunu engellemek için anahtarlama frekansı yükseltilmelidir.

3.3.3 Donanım Kontrolörleri

Yukarıda da bahsedildiği gibi birçok mikrodenetleyici dahili PWM modülüne sahiptir. PWM uygulamalarına başlamadan önce, mikrodenetleyicilerde aşağıda belirtilen yazmaç ayarlarının gerçekleştirilmesi gerekmektedir.

- Dahili sayaçlar ayarlanarak, kare dalganın modülasyonunda kullanılacak çalışma periyodu ayarlanmalıdır
- PWM kontrol yazmaçları ayarlanmalıdır
- PWM çıkışının sağlanacağı pin ayarları yapılmalıdır
- Sayaç başlatılmalıdır
- PWM kontrolü aktif hale getirilmelidir

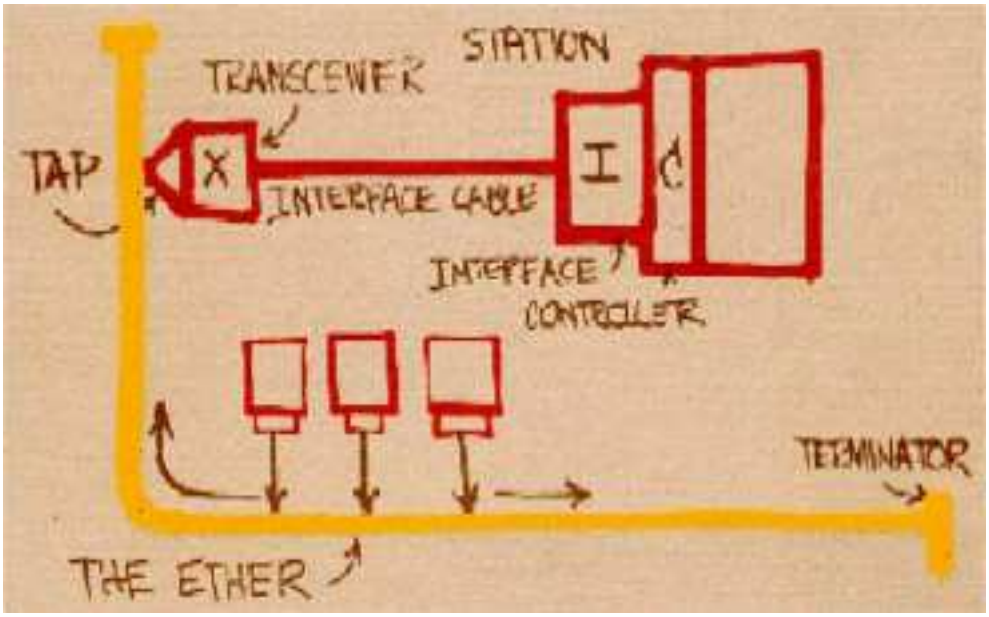
BÖLÜM 4

ROBOT KOL İLE HABERLEŞME

Robot kol ile haberleşme için sistemde Seri – Ethernet Çevirici modül (Tibbo EM202) kullanılmıştır. Seri Ethernet çevirici ile çift taraflı olarak seri port (UART) üzerinden gönderilen veriler Ethernet paketlerine dönüştürülerek karşı tarafa iletilmektedir. Bilgisayar tarafında ise sanal seri port ile bu gönderim sağlanmaktadır. Gönderimin sağlanması için modül ayarlarının yapılmasının ardından bilgisayarda bir adet sanal port kurularak bu sanal port ile modülün IP adresi eşleştirilmelidir. İnternet üzerinden bağlantıda ise robot kolun bulunduğu ağın WAN IP adresi eşleştirileceği için WAN IP adresinin bilinmesi gerekmektedir. Ayrıca internet üzerinden bağlantıda erişimin kesintisiz şekilde sağlanması için WAN IP adresinin statik olması önerilmektedir.

4.1 Ethernet

Ethernet 40 yıla yakın bir geçmişe sahip ağ teknolojisidir. Bundan dolayı günümüzde bir protokolü tanımlamaktan ziyade birden fazla teknolojiyi içine alan bir terim haline gelmiştir. İlk Ethernet, yerel alan ağ kategorisindeki bilgisayarların birbirleriyle haberleşmelerini sağlamak amacıyla 1973 yılında Xerox firmasında araştırmacı olarak çalışan Bob Metcalfe tarafından icat edilmiştir. Sonrasında Bob Metcalfe ve David Boggs ikilisi tarafından yazılan ve 1976 yılındaki National Computer konferansında sunulan bir makale içerisindeki çizimle (Şekil 4.1) literatüre geçmiştir. Dolayısıyla ilk Ethernet şekilde gösterilen yapıda 2,94 Mbps hızında Taşıyıcı Duyarlı Çoklu Erişim / Çarpışmayı Sezme (Carrier Sense Multiple Access / Collision Detection – CSMA/CD) erişim mekanizmasına sahip deneysel bir koaksiyel kablo ağı olarak tanımlanmıştır [14].



Şekil 4.1 İlk Ethernet sistemi çizimi

Xerox, DEC ve Intel firmalarının oluşturduğu bir konsorsiyum, Ethernet'in açık bir standart olabilmesi için 1976 yılında bir proje başlatmış ve bu ortak çalışmanın sonucu olarak 1982 yılında 10 Mbps'lik Ethernet (V2.0) protokolünü geliştirmiştir. Bu versiyon aynı zamanda Ethernet II ya da DIX standardı olarak da tarihe geçmiştir [14].

Yukarıdaki çalışmaların ardından Ethernet'in uluslararası alanda kabul görmesi ve farklı firmalar tarafından standart bir yapıda üretiminin sağlanabilmesi için IEEE bir çalışma içerisine girmiş ve bu çalışmanın adını da IEEE 802.3 olarak ilan etmiştir.

IEEE, iletişim dünyasına çok hızlı ve etkili bir giriş yapan ATM (Asenkron Transfer Mod) protokolüne alternatif olarak 1998-1999 yılları arasında Gigabit Ethernet'i geliştirmiştir. ATM, hem LAN, hem WAN, hem de WAN bağlantılarında ses, veri ve görüntü trafiklerinin istenilen servis kalitesi içerisinde aktarımını sağlayan bir çoğullama teknolojisidir. Fast Ethernet, ATM'nin sunduğu bu özellikleri karşılamadığından dolayı hızın artırımı ile birlikte farklı paket yapılarının kullanımı da dahil edilerek Gigabit Ethernet geliştirilmiştir. Bu gelişme ile Ethernet, ATM'nin LAN bağlantılarında kullanımını devre dışı bırakmıştır; fakat WAN bağlantılarında ATM protokolüne üstünlük kuramamıştır.

Ethernet'in günümüze yansıyan bir diğer gelişmesi de Gigabit Ethernet'tir. Bu gelişme Kuzey Amerika'da Internet2 olarak isimlendirilen yüksek hızlı bağlantılarda kullanılmak üzere tasarlanmıştır. Çizelge 4.1, IEEE tarafından günümüze kadar geliştirilen, Ethernet çeşitlerini ve özelliklerini göstermektedir [15].

Çizelge 4.1 IEEE 802.3 protokolleri

Standart İsmi	Tanımı	Tarihi	Band Genişiiği	Maksimum Mesafe	Kullanılan Kablo
IEEE802.3	10Base-5 (Thicknet)	1985	10Mbps	500 metre	50 ohm sonlandırıcı kalın koaksiyel kablo
IEEE 802.3a	10Base-2 (Thinnet)	1986	10Mbps	185 metre	50 ohm sonlandırıcı ince koaksiyel kablo
IEEE 802.3i	10Base-T	1991	10Mbps	100 metre	Cat 3, Cat 4, Cat 5 UTP
IEEE 802.3j	10Base-F	1994	10Mbps	2 km	Fiber Opitk
IEEE 802.3u	100Base-TX	1995	100Mbps	100 metre	Cat 5 UTP veya Type 1 STP
IEEE 802.3u	100Base-T4	1995	100Mbps	100 metre	Cat 3, Cat 4, Cat 5 UTP
IEEE 802.3u	100Base-FX	1995	100Mbps	450 metre- 2 km	Fiber Optik
IEEE 802.3z	1000Base-LX	1998	1000 Mbps	440 metre- 3 km	Single Mod veya Multi Mod Fiber Optik Kablo
IEEE 802.3z	1000Base-SX	1998	1000 Mbps	260-550 metre	Multi Mod Fiber Optik Kablo
IEEE 802.3z	1000Base-CX	1998	1000 Mbps	25 metre	Bakır Kablo
IEEE 802.3ab	1000Base-T	1999	1000 Mbps	100 metre	Cat 5 UTP
IEEE 802.3ae	10GBase-xR	2003	10Gbps	26 metre – 80 km	Single Mod veya Multi Mod Fiber Optik Kablo
IEEE 802.3ae	10GBase-xW	2003	10Gbps	26 metre – 80 km	Single Mod veya Multi Mod Fiber Optik Kablo
IEEE 802.3an	10GBase-T	2006	10Gbps	100 metre	Cat 6 UTP veya STP

4.1.1 TCP/IP Protokolleri

Bilgisayarlar ile veri iletme/alma birimleri arasında organizasyonu sağlayan, böylece bir yerden diğerine veri iletişimini olanaklı kılan pek çok veri iletişim protokolüne verilen genel addır. Bu protokollere örnek olarak, dosya alma/gönderme protokolü FTP (File Transfer Protocol), elektronik posta iletişim protokolü SMTP (Simple Mail Transfer Protocol), İnternet üzerindeki başka bir bilgisayarda etkileşimli çalışma için geliştirilen *login* protokolü TELNET verilebilir.

Adı sıkça duyulan, internet ortamında birbirine veri transferi ve dataların iletimini sağlayan protokol ise Hyper Text Transfer Protocol (HTTP) olarak adlandırılmaktadır. TCP/IP protokolü aynı zamanda diğer iletişim ağlarında da kullanılabilir. Özellikle pek çok farklı tipte bilgisayarı veya iş istasyonlarını birbirine bağlayan yerel ağlarda (LAN) kullanımı yaygındır.

4.1.2 Bazı TCP/IP Protokolleri

TCP/IP protokolleri katmanlarına göre aşağıdaki gibi sınıflandırılabilir.

4.1.2.1 Donanım Katmanındaki Protokoller

- **ARP** (Adres Çözümleme Protokolü), bir IP adresinin hangi ağ kartına (yani MAC adresine) ait olduğunu bulmaya yarar. TCP/IP'de veri gönderiminde gönderilecek bilgisayarın hangisi olduğunu bulmak için kullanılır. Ayrıca IP adresini yeni almış olan bir makine, o IP adresinin sadece kendisinde olduğunu ARP kullanarak teyit eder.
- **RARP** (Ters ARP) protokolü ARP'nin tersi işlemi yapar, yani hangi MAC adresinin hangi IP adresini kullandığını bulur. Bir TCP/IP ağında RARP'nin çalışacağı garanti değildir, zira RARP bir RARP sunucusuna ihtiyaç duyar.

4.1.2.2 IP Katmanındaki Protokoller

- **ICMP** (Internet Yönetim Mesajlaşması Protokolü), hata ve bilgi mesajlarını ileten protokoldür. Örneğin, ping programı ICMP'yi kullanır.
- **RIP** (Router Bilgi Protokolü), routerların yönlendirme tablolarını otomatik olarak üretebilmesi için yaratılmıştır.
- **OSPF** (İlk Açık Yöne Öncelik), aynı RIP gibi routerların yönlendirme tablolarını otomatik olarak üretebilmesini sağlar. OSPF, RIP'ten daha gelişmiş bir protokoldür.
- **IGMP** (Internet Grup Mesajlaşma Protokolü), bir sistemin internet yayınlarına (multicast) abone olmasına ve aboneliği durdurmasına yarar. Bu yayınlar, UDP üzerinden yapılır ve genelde çoklu ortam (radyo veya video) içerikli olurlar.
- **DHCP** (Dinamik Cihaz Ayar Protokolü), TCP/IP ağına bağlanan bir cihaza otomatik olarak IP adresi, ağ maskesi, ağ geçidi ve DNS sunucusu atanmasına yarar.

4.1.2.3 Taşıma Katmanındaki Protokoller

- **UDP** (Kullanıcı Veri Protokolü), IP üzerinden veri yollamaya yarar. Verilerin ulaşacağını garanti etmez ve UDP paketlerinin maksimum boy sınırları vardır. Öte yandan, UDP son derece basit ve bağlantı gerektirmeyen bir protokoldür.
- **TCP** (Gönderim Kontrol Protokolü), IP üzerinden ulaşma garantili ve herhangi bir boyda veri gönderilmesine imkan tanıyan bir protokoldür. UDP'den farklı olarak, TCP'de iki cihazın iletişim kurabilmesi için önce birbirlerine bağlanmaları gerekmektedir.

4.1.2.4 Uygulama Katmanındaki Protokoller

- **DNS** (Alan Adı Sistemi), alan adı verilen isimler (mesela www.wikipedia.org) ile IP adreslerini birbirine bağlayan sistemdir. Paylaştırılmış bir veritabanı olarak çalışır. UDP veya TCP üzerinden çalışabilir.
- **HTTP** (Hiper Metin Yollama Protokolü), ilk başta HTML sayfaları yollamak için yaratılmış olan bir protokol olup günümüzde her türlü verinin gönderimi için kullanılır. TCP üzerinden çalışır.
- **HTTPS** (Güvenli HTTP), HTTP'nin RSA şifrelemesi ile güçlendirilmiş halidir. TCP üzerinden çalışır.
- **POP3** (Postane Protokolü 3), e-posta almak için kullanılan bir protokoldür. TCP üzerinden çalışır.
- **SMTP** (Basit Mektup Gönderme Protokolü), e-posta göndermek için kullanılır. TCP üzerinden çalışır.
- **FTP** (Dosya Gönderme Protokolü), dosya göndermek ve almak için kullanılır. HTTP'den farklı olarak kullanıcının mutlaka sisteme giriş yapmasını gerektirir. Veri ve komut alış verişi için iki ayrı port kullanır. TCP üzerinden çalışır.
- **FTPS** (Güvenli FTP), FTP'nin RSA ile güçlendirilmiş halidir. TCP üzerinden çalışır.

Tüm bu protokoller ve daha fazlası sayesinde TCP/IP her geçen gün daha da popülerleşen bir protokol olmuştur.

4.1.3 UDP ve TCP

Genelde UDP ile TCP'nin veri teslim yöntemleri arasındaki fark telefonla aramayla postayla kart gönderme arasındaki farka benzer. TCP hedefin erişilebilir ve iletişime hazır olduğunu doğrulayarak bir telefon çağrısı gibi çalışır. UDP ise kart gibi işler; iletiler küçüktür ve teslimat büyük olasılıkla gerçekleşir fakat bu asla garanti edilemez.

UDP genelde bir seferde küçük miktarlarda veri ileten veya gerçek zamanlı işlemlere gereksinim duyan programlar tarafından kullanılır. Bu durumlarda UDP'nin düşük üstbilgi yükü ve çok noktaya yayın yetenekleri (örneğin bir veri birimi için birden çok alıcı) TCP'den daha elverişlidir.

UDP, TCP'nin sağladığı hizmet ve özelliklerle tam karşıttır. Aşağıdaki tabloda verilerin taşınmasında UDP veya TCP kullanılmasına bağlı olarak TCP/IP iletişiminin nasıl işlendiği gösterilmektedir.

Çizelge 4.2 UDP – TCP karşılaştırması

UDP	**TCP**
Bağlanma olmadan verilen hizmet; ana bilgisayarlar arasında oturum açılmaz.	Bağlanarak verilen hizmet; ana bilgisayarlar arasında oturum açılır.
UDP verilerin sırasını veya teslimini garanti etmez veya doğrulamaz.	TCP, doğrulamayla verinin hedefe ulaşmasını ve verilerin sıralı teslimini garanti altına alır.
UDP kullanan programlar verilerin taşınması için gerekli güvenilirliği kendileri sağlamak zorundadır.	TCP kullanan programlara güvenli veri taşıma garantisi sağlanır.
UDP hızlıdır, ek yükü azdır, noktadan noktaya ve noktadan birden çok noktaya iletişimi destekleyebilir.	TCP daha yavaştır, ek yükü fazladır ve yalnızca noktadan noktaya iletişimi destekler.

UDP ve TCP, her TCP/IP programı için iletişimi tanımlamak üzere bağlantı noktalarını kullanır.

4.2 UART (Universal Asynchronous Receiver Transmitter)

UART, PC ve mikroişlemciler arasındaki seri iletişim arayüzüdür. Bu iletişim 5 ile 9 bit arası data uzunluğu taşıma özelliğine sahiptir. Ancak genellikle kullanım sırasında 8 veya 9 bit tercih edilir.

4.2.1 UART Genel Özellikleri

UART iletişim arayüzü kullanılırken öncelikle veri taşıma hızı (Baud) ayarlanır. Örneğin, iletişim hızı 9600 Baud olan bir seri iletişim arayüzü saniyede 960 Byte göndermektedir (Eğer 1 adet Stop-Bit göz önüne alınırsa). Bir byte uzunluğunda veri için 1 Start-Bit, 8 Data-Biti ve 1-2 adet Stop-Bit gönderilir. Genellikle 2. Stop-Biti, verici alıcıdan daha hızlı olduğu durumlardaki UART uygulamalarında kullanılır. UART data formatı genelde şu şekilde yazılmaktadır:

- 8N1 : 8 Data-Bit, No Parity, 1 Stop-Bit

- 9E1 : 9 Data-Bit, Even Parity, 1 Stop-Bit

- 9O2 : 9 Data-Bit, Odd Parity, 2 Stop-Bit

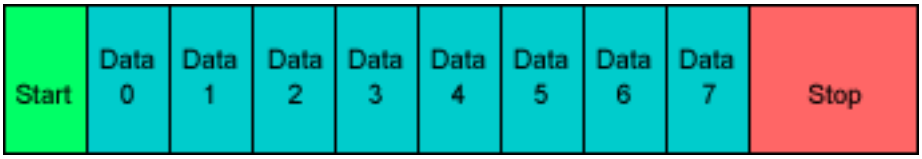
Asenkron veri aktarımının hatasız (ya da en az hata ile) çalışabilmesi için verici ve alıcının iletişim hızları aynı ya da çok yakın olarak ayarlanmalıdır. Örneğin, 10 bitlik bir veri paketi en fazla %4'lük hata payına kadar çıkabilmektedir. Güvenli bir data aktarımı için %2'lik hata payına kadar göz ardı edilebilir. Bu gecikme zamanında oluşan hata payı içerisinde yedekleme süresi, çalıştırma ve durdurma süreleri vb. de barındırmaktadır [16].

Kullanım alanları:

- PC ile iletişim
- Robot içi alt sistemler arası iletişim
- Bootloader
- SPI iletişim arayüzü
- Wireless, Ethernet iletişimi

4.2.2 UART İletişimi

Asenkron bir seri iletişiminde bir byte uzunluğunda veri gönderilirken aşağıdaki gibi bir yapı görülmektedir:



Şekil 4.2 UART haberleşme yapısı

Bu durumda vericinin TX (Transmit) pini, alıcının RX (Receive) pini ile devamlı olarak iletişim halindedir. Sinyal, iletişimin IDLE durumunda iken HIGH konumunda bulunur. Eğer veri aktarımı başlıyor ise, verici Start-Bit olarak sinyalin konumunu LOW yapar ve alıcı bunu fark ettiğinde dinlemeye geçer. Eğer hardware UART kullanılıyor ise, tüm işlemler işlemci içerisinde otomatik olarak gerçekleşir. Lojik 0 sinyalini algılayan alıcı, gelen veri için kullanacağı UART veri yazmacına (UART Data Register) sinyalin durumuna göre 0 veya 1 bitlerini kaydeder. Eğer iletişim sırasında herhangi bir hata oluşmuş ise Stop-Bit gönderici tarafından gönderilemeyeceği için alıcı tarafından alınamayacaktır. Bu durumda alıcının kilitlenme ihtimalinden ötürü, ya dışarıdan bir buton ile işlemci resetlenir veya içeriden bir kesme ile işlem sonlandırılır. Fakat software UART ile bir seri iletişim kurulmuş ise, veri aktarımı için ayrılmış olan süre dolduğunda zamanlayıcı (Timer) kesme (Interrupt) bayrağını kaldırır ve işlemci kilitlenmekten kurtulur.

4.3 Robot Kol İle Haberleşme

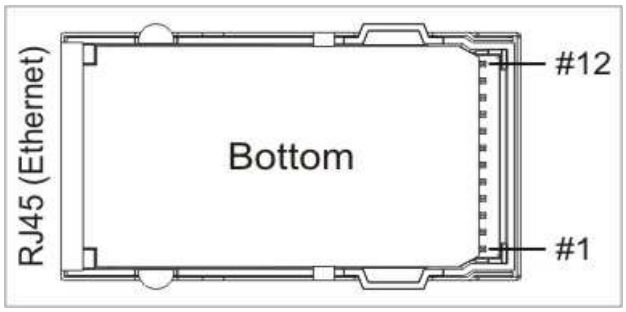
Görüldüğü üzere akıllı bir sistemin kurulmasının ilk aşaması olan, elektronik eşyaya ait bilgilerin elde edilmesi kısmı, belli kısıtlamalar dahilinde tamamlanmıştır.

Buradaki kısıtlamalar, dış dünya haberleşme biriminin seri haberleşme (RS232 iletişim kuralı) olmasından dolayı, uzak mesafelerde veri kaybının olması ve PC'nin izlenecek eşyanın yanına götürülmesi zorunluluğudur. Sıradaki adım seri haberleşmeyi Ethernet'e çevirmek ve makinenin yerel ağ (LAN) ya da İnternet (WAN) üzerinden kontrol ve gözlemlenmesini sağlamaktır.

Şekil 4.3 Tibbo seri Ethernet çevirici

EM202, kart üzerine montaj için RJ45 formunda bir Ethernet modülüdür. Modül donanımsal olarak bir adet 100BaseT Ethernet portu (Dahili Ethernet mıknatısları ve RJ45 konnektörü), bir adet seri port (CMOS) ve içerisindeki yazılımı Ethernet ile seri port arasında köprü görevi gören bir işlemci bulundurur. Modülün ağ tarafındaki RJ45 Ethernet konnektörüne standart Ethernet kablosu takılabilir. Seri tarafı ise mikrodenetleyicilerin seri portuna direkt olarak bağlanabilir.

Donanım açısından bakıldığında, EM202 bir ağ çeşidi ve seri haberleşme ile ilgili uygulamalarını çalıştırmak için uygun evrensel bir platform olarak görülebilir. Bu işlevselliğin en önemli nedeni ise donanım değil içerisindeki yazılımdır. Destekleyen uygulama yazılımı EM202'yi, seri cihazları Ethernet ağına bağlamak için kullanılan bir Seri Aygıt Sunucusuna dönüştürür. Modülün uygulama yazılımı, modül üzerindeki seri port ya da Ethernet portu üzerinden güncellenebilir [17].



Şekil 4.4 Tibbo EM202 modülü alttan görünümü

Çizelge 4.3 Tibbo EM202 modülü pinleri

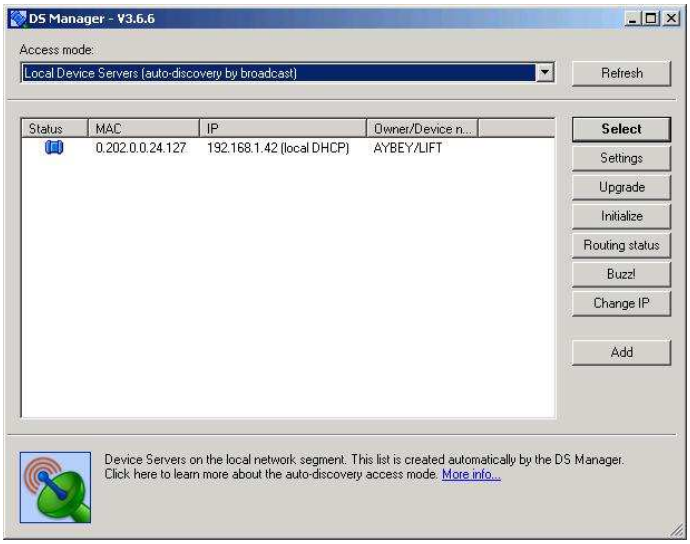
No	Adı	Tipi	Özellik
#1	MD (MD)	Giriş	Mod seçim pini
#2	RST	Giriş	Reset
#3	P3 (DTR)	Giriş/Çıkış	Genel amaçlı Giriş/Çıkış hattı Data hattı hazır çıkışı
#4	P2 (DSR)	Giriş/Çıkış	Genel amaçlı Giriş/Çıkış hattı Data ayar hazır girişi
#5	L3 (SG)	Çıkış	LED çıkış 3 (Yeşil durum ledi)
#6	L4 (SR)	Çıkış	LED çıkış 4 (Kırmızı durum ledi)
#7	VCC	-	Besleme pozitif girişi, 5V nominal, +/- 5%, 220mA
#8	GND	-	Besleme negatif girişi
#9	RX	Giriş	Seri alma hattı
#10	TX	Çıkış	Seri gönderme hattı
#11	P4 (CTS/SEL)	Giriş/Çıkış	Genel amaçlı Giriş/Çıkış hattı CTS girişi / Tam/yarı-dubleks seçim girişi
#12	P5 (RTS/DIR)	Giriş/Çıkış	Genel amaçlı Giriş/Çıkış hattı RTS çıkışı (tam-dubleks modda) Data yönü kontrol çıkışı (yarı-dubleks modda)

Ürünün özellikleri aşağıdaki gibidir

- Uzaktan kontrol edilebilir seri bağlantı noktaları
- Tek veya çift yönlü işlem
- Seri bağlantı noktası ya da Ethernet üzerinden kurulum ve programlama
- Sabit ya da dinamik (DHCP destekli) IP
- ADSL modem üzerinden kontrol edilebilirlik
- UDP, TCP, ARP, ICMP (PING), DHCP, PPPoE ve LCP iletişim kurallarını desteklemesi
- TIBBO BASIC programı ile programlanabilirlik
- FLASH bellek
    - Temel olarak BASIC programlama dilini almıştır.
    - Sistem uygulamaları dışında içerdiği örün sunucusu sayesinde örün uygulamalarını da (HTML, JavaScript destekler) çalıştırabilir.
    - Çeşitli yüksek-seviye nesneler:
- Sock: TCP-UDP olarak en fazla 16 bağlantı
- Ser: UART haberleşme
- IO: Seri bağlantı noktası hatlarının denetimi
- STOR: EEPROM'dan okuma ve EEPROM'a yazma
- ROMFILE: Değişmeyen verilerin saklanması

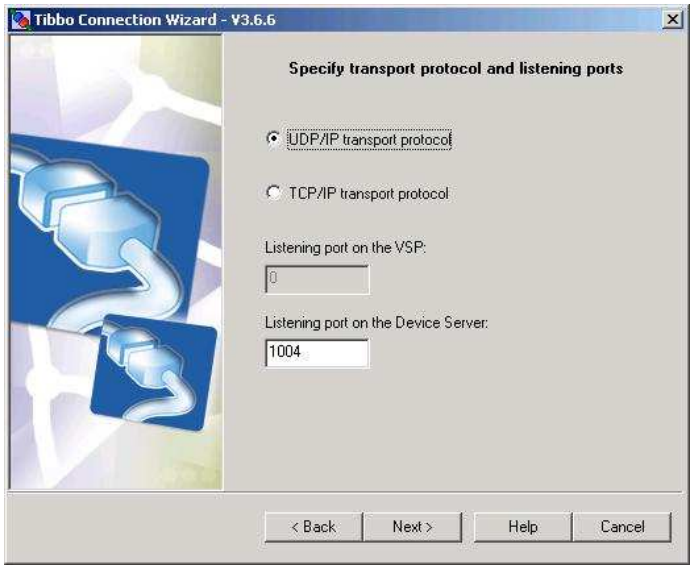
Ürünün en basit özelliği olan Ethernet-seri bağlantı noktası dönüşümü çift taraflı olacak ve döngü okuma halinde sürecektir. Bu nedenle haberleşme sürekli şekilde sağlanacaktır.

Tibbo modül ayarları için TDST (Tibbo Device Server Toolkit) isimli programı kullanıcılarına sunulmuştur. Bu programda bulunan DSManager programı ile modülün bağlantı için gereken ayarları yapılmaktadır.



Şekil 4.5 DSManager görünümü

Yapılan ayarların ardından bilgisayar için Sanal Seri Port kurmak gerekmektedir. Bu sanal port üzerinden modülle bilgisayar arasında bağlantı yapmak mümkün olmaktadır. Sanal port kurulum aşamasında haberleşme tipi olarak UDP seçilmesi ile internet üzerinden ve gerekli olması durumunda birden çok bağlantıya olanak vermektedir.



Şekil 4.6 Sanal port tanımlama

İnternet üzerinden bağlantıda ise öncelikle robot kolun bağlı olduğu modemden 2 adet port yönlendirme yapılması gerekmektedir. Bu port yönlendirmelerden biri uzaktan erişim yapılacak bilgisayar üzerindeki DSManager programı üzerinden robot kola erişmek ve gerekli ayarları yapmak ve diğer port ise RCRA bilgisayar arayüz programı ile haberleşmede kullanılmak üzere yönlendirilmektedir.

BÖLÜM 5

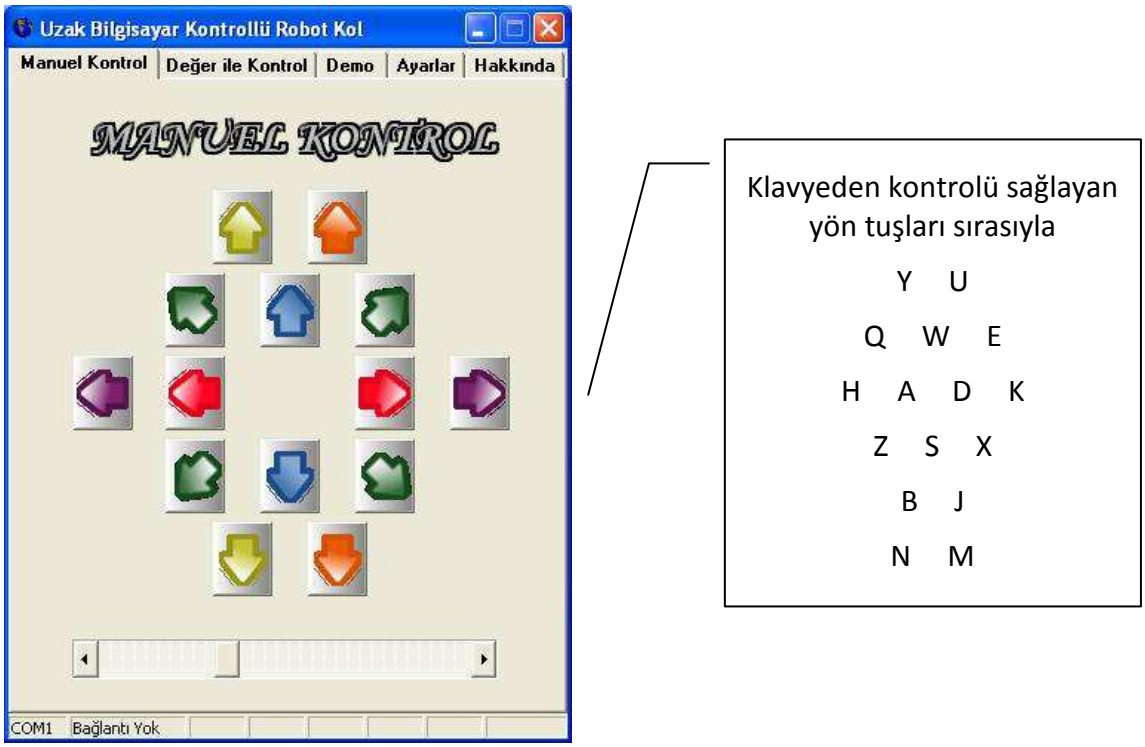
ROBOT KOL ARAYÜZ PROGRAMI

Robot kolu uzak bilgisayardan kontrol etmek için hazırlanan RCRA programı Delphi programlama dili ile geliştirilmiştir. Hazırlanan arayüz programı ile robot kolun 2 farklı şekilde (adım adım ya da değer ile) eşzamanlı olarak kontrolü veya önceden hazırlanmış demo hareketlerinin yaptırılması mümkün olmaktadır.

Şekil 5.1 RCRA programı

5.1 Manuel Kontrol

Arayüz programının manuel kontrol kısmında bulunan kontrol (yön) tuşları ve klavye üzerinden önceden belirlenmiş olan 16 ayrı tuşla robot kolun kontrolü yapılabilmektedir.



Şekil 5.2 Manuel kontrol sayfası

İki ayrı kontrol alternatifine sahip olan robot kolun kontrolü klavyeden yapılacaksa, klavye tuşlarının aktif olabilmesi için öncelikle programın Manuel Kontrol sayfasının aktif olması gerekmektedir. Klavye tuşlarının aktivasyonu sağlandıktan sonra, klavyeden yukarıda belirlenen tuşlardan herhangi biri basılı tutulduğu sürece, robot kol tuşun ilişkilendirildiği hareketi gerçekleştirir. Bu işlem arayüzdeki yön butonlarıyla (mouse aracılığıyla) yapıldığında ise; her basma sonunda robot kol, butonun yönettiği yönde bir derece hareket etmektedir. Kolun bir yönde hareket edebileceği maksimum açı değerleri de önceden belirlenmiştir.

5.2 Değer ile Kontrol

Değer ile kontrolde yapılması istenen hareketler, hareketi yapacak olan servo motorun numarası seçilerek ve girilen/ayarlanan açı değerleri ile yapılır. Değer girilebilmesi/ayarlanabilmesi için hareket ettirilmek istenen servo motorun sahip olduğu kutucuğun işaretli olması gerekmektedir. Ayarlanan değer eşzamanlı olarak servo motora gönderilmektedir. Değer hanesine elle girilen değeri göndermek için ise hanenin yanındaki gönderme tuşu kullanılır.

Şekil 5.3 Değer ile kontrol sayfası

5.3 Demo

Robot kolun kontrol arayüzü programında bulunan demo butonlarıyla, robot kolun önceden hazırlanmış olan hareketleri sırasıyla yapması sağlanmaktadır. Demo sayfası açıldığında 4 adet demo seçim butonu ekrana gelmektedir. Ekrana gelen demo butonlarından seçilen butona göre robot kol, önceden programcı tarafından hazırlanan hareketleri sırasıyla yapmaktadır.

Menüde bulunan Demo 3 ve Demo 4 ise eş zamanlı hareket kaydı desteğine sahiptir. Bu sayede kullanıcı kayıt seçeneğini aktif ettiğinde, kayıt bitirme tuşuna basılana kadar robot kolun yaptığı hareketler seçilen demo hareketine kaydedilir.

Şekil 5.4 Demo sayfası

5.4 Ayarlar ve Şifre Değişikliği

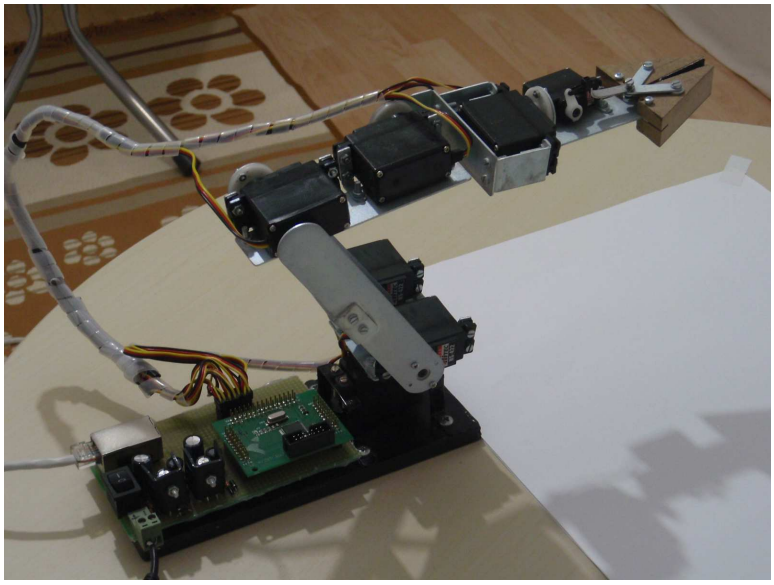
Robot kol ile bağlantı kurulması, bağlantı ayarlarının yapılması, lisan seçimi ve şifre değiştirilmesi için hazırlanan pencerede ayrıca robot kolun yapılan hareketlerden sonra konumunun ilk hale getirilmesi veya robot kolun hareketlerinin tamamlanması sonucunda bekleme moduna getirilmesi için başlangıç komutu butonu bulunmaktadır. Robot kola bağlantı sağlamak için bağlı olduğu seri port listeden seçilmeli ve bağlantı şifresi girilmelidir. Seri port doğru olarak seçildiğinde programın alt kısmında "Port Açıldı" şeklinde uyarı görülür. Portun seçilmesinin ardından şifre hatalı girilmişse "Bağlan" tuşuna basıldığında kullanıcı sistem tarafından uyarılır. Bağlantı sağlanamazsa buton rengi kırmızı renge; bağlantı sağlandığı takdirde ise yeşil renge dönmektedir. Bağlan tuşu ile bağlantı sağlanmadan program robot kola erişime izin vermemektedir.

Önceden ayarlanmış lisan ve seri port değerleri program çalıştırıldığında otomatik olarak sisteme aktarılmaktadır. Bu butonların altında ise şifre değişikliği için kullanılan kısım bulunmaktadır. Şifre değişikliğinde güvenlik amacıyla eski şifre girildikten sonra kaydedilecek şifrenin onay amaçlı olarak 2 kez girilmesinin ardından gönder tuşu ile sisteme gönderilir ve sistemin onayın ardından şifre değiştirme işlemi tamamlanmış olur.

Şekil 5.5 Ayarlar ve şifre değişikliği sayfası

BÖLÜM 6

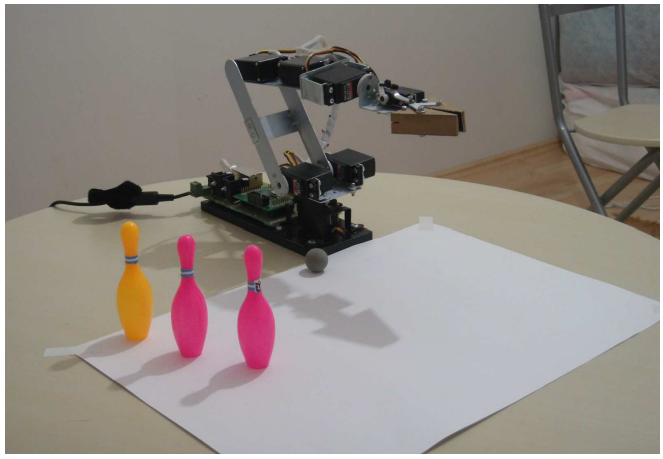
UYGULAMA



Şekil 6.1 Robot kolun görünümü

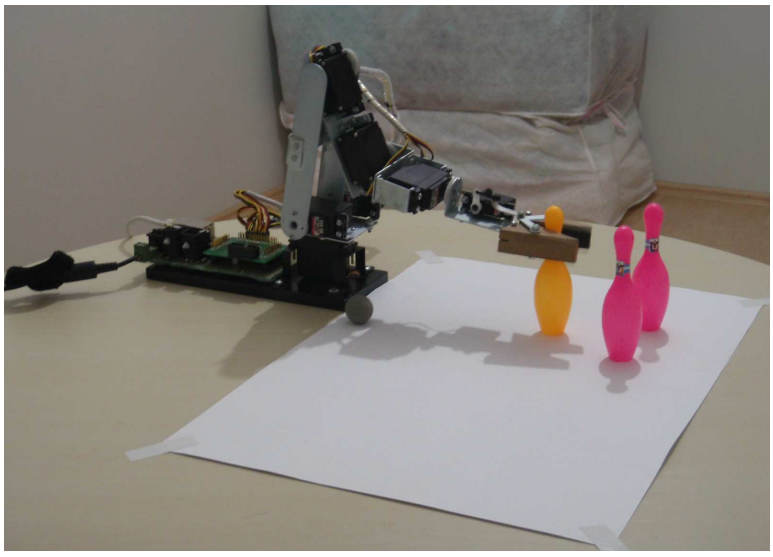
Projenin kontrol kısmının arayüz programında manuel ve değer ile kontrol pencereleri üzerinden kontrol edilmesi, robot kolun bilgisayar üzerinden eşzamanlı olarak kontrolünü sağlamaktadır. Önceden tanımlanmış bir görevin komutlarının ardı ardına yapılmasını simüle eden kısım ise programın Demolar kısmında bulunmaktadır. Burada yer alan 4 adet demo butonu sistemde kayıtlı olan hareketlerin ardı ardına yapılmasını sağlamaktadır. Bu demolardan ilk iki tanesi sadece sisteme önceden kaydedilmiş komutları çalıştırmak için kullanılmakta iken, diğer ikisi ise çalışma anında verilen komutları yani hareketleri eşzamanlı olarak hafızasına alma desteği sayesinde istenilen zamanda yeni görevlerin kaydedilmesine olanak sağlamaktadır.

Sisteme önceden kaydedilmiş demo hareketlerinden ilki, robot kolun 3 adet oyuncak bowling labutlarını sırasıyla bulunduğu yerden alıp robot kolun önünde bulunan boşlukta yaklaşık 15 cm mesafeye dizmesinin ardından sağ kısmında duran top ile labutları düşürmeye çalışmak; yani bir çeşit bowling oyunu simülasyonu şeklindedir.



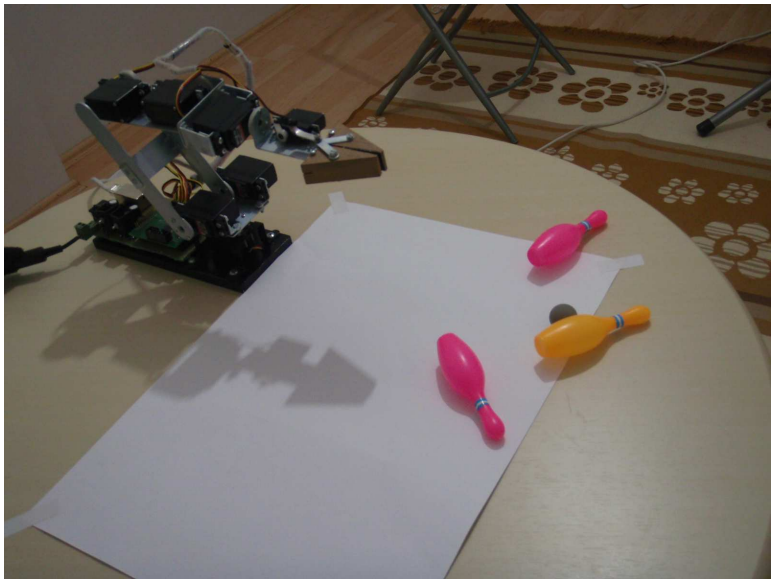
Şekil 6.2 Demo 1-Başlangıç durumu

Bu kısımda robot kol sağ çaprazında duran 3 adet labutu sırasıyla alıp ön tarafta bulunan boşluğa hassas bir şekilde dizmektedir.



Şekil 6.3 Demo 1-Labutların dizilişi

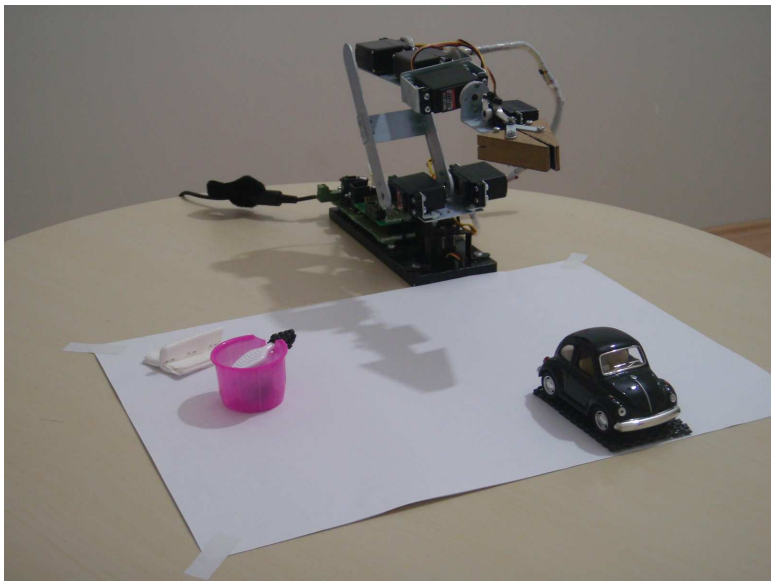
Dizme işleminin tamamlanmasının ardından topu alıp önünde duran labutlara fırlatmak için kol hareketine benzer şekilde hareket ederek yani 3. eklem kısmını önce geriye doğru yaklaşık 20 derece çekmesinin ardından 40 derece öne doğru hızla getirip yaklaşık 25. derecede ağzın açılması ile topu labutlara fırlatmaktadır.



Şekil 6.4 Demo 1-Bitiş durumu

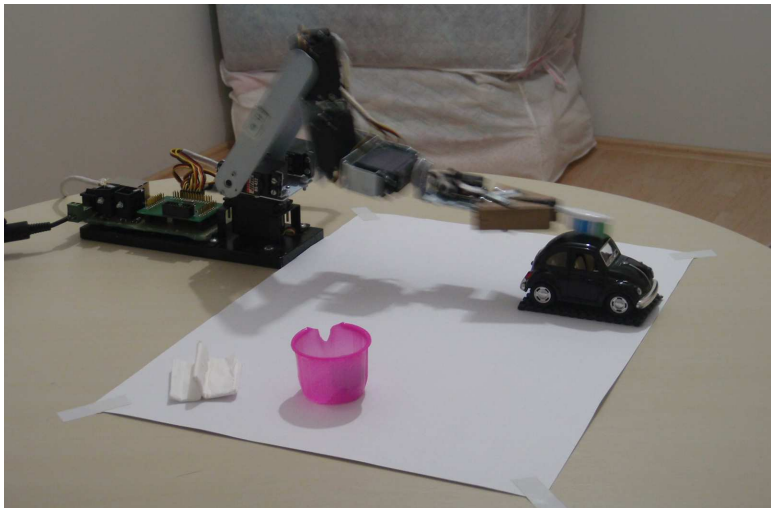
Burada geriye doğru çekilen kolun açısı, topun ne kadar ileriye atılacağını belirlediğinden dolayı önem taşımaktadır. Ayrıca atılan topun labutlardan daha ağır olması da labutların yere daha kolay düşmesini sağlayacağından dolayı bilgisayar faresinde bulunan top kullanılmıştır. Topun ağırlığı ise özellikle Servo2, Servo 3 ve Servo 4 üzerindeki yüke etki edeceğinden dolayı buradaki servo motorların ve topun seçimi birbirini etkilemektedir.

İkinci demo ise oyuncak arabayı önce fırça ile fırçalayıp daha sonra bez ile silinmesi demosu yani araba yıkama simülasyonudur.



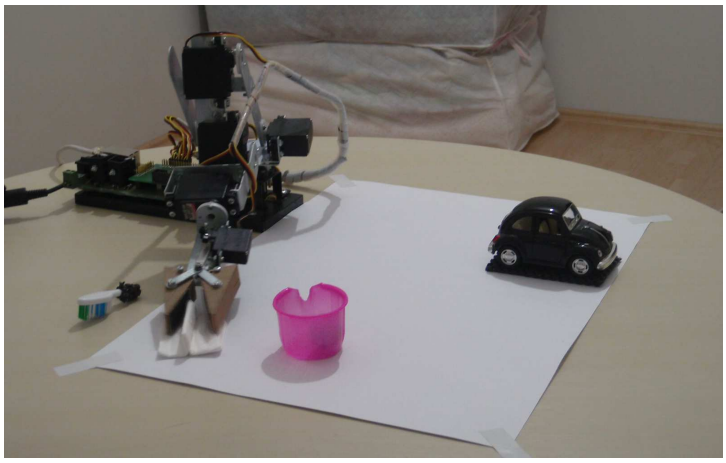
Şekil 6.5 Demo 2-Başlangıç durumu

Bu kısımda ise öncelikle robot kol sağ çaprazında duran kovanın içindeki fırçayı alıp kovanın içinde karıştırma hareketi yapmaktadır. Karıştırma hareketinin ardından robot kolun önünde duran arabanın önce üst tarafı, sonra sağ ve son olarak da sol tarafı robot kolun ileri geri hareketleri ile fırçalanır.



Şekil 6.6 Demo 2-Fırçalama hareketi

Yan tarafların fırçalanması ağız kısmının sağa ve sola 90 derece dönmesini sağlayan eklem ile mümkün olmuştur. Fırçalama işlemlerinin ardından robot kolun fırçayı bırakıp demonun ikinci kısmına başlamak için yine sağ çapraz kısmında bulunan bezi alması gerekmektedir. Buradaki önemli nokta ise bezi sıkıca kavrayabilmesi için robot kolun ağız kısmının yere paralel şekilde konumlanıp yere değebilecek konuma gelebilmesidir. Bu kısım mekanik tasarımda dikkat edilmesi gereken önemli unsurlardan biridir.



Şekil 6.7 Demo 2- Bitiş durumu

Bezin alınması ile arabanın önce üst tarafı, sonra sağ ve son olarak da sol tarafı bez ile silinir. Silme işleminin ardından robot kol bezi aldığı yere geri bırakarak demo hareketini tamamlar.

Projede hareket yeteneğini belirleyen servo motor seçimi, en önemli noktalardan biridir. Motor seçimi kriterleri özetlenecek olursa, robot kolun en alt tarafında yer alan servo motorun (Servo 1), robotu sadece sağ ve sol tarafa döndürecek olmasından dolayı burada standart güce yakın bir servo motor (HS-311 3kg/cm-4.8V) seçilmiştir. İleriye doğru hareket eden servo motorun (Servo 2) yükü, sistemde taşınan en büyük yük olduğundan dolayı tek servo motor yerine standart güçte (HS 422 3,3kg/cm-4.8V) iki servo motor kullanılmış ve bu iki servo motorun senkron hareket etmesi sağlanmıştır. Bunların üst kısmında yer alan servo motor (Servo 3) ise Servo 2'den daha az yüke maruz kalacağından dolayı bu kısımda standart servo motorlardan yaklaşık %50 daha güçlü (HS 485 4,8kg/cm-4.8V) tek servo motor kullanılmıştır. Servo 4 ile ağız kısmının (Servo 6) eksen hareketini sağlayan servo motorlar da standart güçte servo motorlardır (HS422 3,3kg/cm-4.8V). Ağız kısmında ise çok büyük yüklere maruz kalınmayacağından ve diğer servo motorların yükünü artıracağından dolayı diğer servo motorlara nazaran daha hafif ve nispeten daha güçsüz (HS-81 2,6kg/cm-4.8V) bir servo motor seçilmiştir.

Hareket kabiliyeti olarak Servo 1 ve Servo 5, 0 - 180 derece arası dönebilmektedir. Servo2, Servo 3 ve Servo 4 ise herhangi bir hareket engeli olmadığı durumlarda 0 – 180 derece arası konumlanabilmektedir. Ağız kısmının çektirmeli yapısı nedeniyle Servo 6'nın ise sadece 60 ile 180 derece arasında dönmesine izin verilmektedir.

BÖLÜM 7

SONUÇ VE ÖNERİLER

Gerçekleştirilen projede robot kolun uzak bilgisayar üzerinden kontrolü sağlanmıştır. Bilgisayar kontrol programıyla yerel ağ ya da internet üzerinden robot kola bağlantı sağlanarak, bilgisayar üzerinden klavye ve mouse aracılığıyla kontrol kartına gönderilen komutların servo motorlara aktarılmasıyla robot kol kontrol edilmiştir.

İnternet üzerinden erişim ve kontrol desteği, robot kolun kullanım alanının genişlemesinin önünü açmıştır. Robot kolun internete bağlantısı sürekli olarak sağlandığı takdirde internet olan herhangi bir yerden robot kola uzaktan erişim ve bu sayede istenilen komutların gönderilmesi mümkün olmaktadır. Erişim desteğindeki şifre koruması da yetkisiz kullanıcıların erişimine izin vermeyerek robot kolun bağlantı güvenliğini sağlamıştır.

Servo motorların kontrolünün PWM sinyalleriyle yapılması da, hareket hassasiyetinin artmasına ve sinyallerin diğer işlemlerden daha öncelikli olması nedeniyle sinyal kaybı ya da gecikmesi yaşanmadığından dolayı robot kolda kontrol dışı hareketlerin oluşmasının önüne geçilmesini sağlamıştır.

İletişimin uzak kontrollü bilgisayar ile sağlandığından dolayı herhangi bir haberleşme hatasında robot kol ile bilgisayar arasındaki haberleşmenin kopması durumunda robot kol son konumunu koruyacağından dolayı kontrolsüz bir hareket yapması önlenmiştir ve bu da emniyet güvenliğini artırmaktadır.

Projede hareket yeteneğinin servo motorlara bağımlı olması nedeniyle, servo motor seçimi üzerinde durulması gereken en önemli unsurlardan biri olmalıdır. Ekstra gücün beraberinde robot koldaki toplam yük ve maliyet artışı getireceğinden dolayı optimum

çözüm için uygun servo motor seçimine önem verilmelidir. Ayrıca servo motorlarda yüke bağlı olarak çekilen akımın amper seviyelerine çıkmasından ve güç devresinde oluşan aşırı yüklenmelerin istenmeyen hareketlere neden olacağından dolayı güç devresinin kullanılan servo motorlara uygun olarak tasarlanması gerektiği de önemli tasarım maddelerinden biridir.

Geleceğin teknolojisi olan robot kolların endüstrinin yanı sıra tıpta, askeri uygulamalarda ve uzay bilimlerinde kullanımı yoğun şekilde artacak; hatta evlerimizde en büyük yardımcımız olmaya başlayacaktır. Sensörler aracılığıyla bir insanın algılama yeteneğine sahip olması, ayrıca yapay zeka ile öğrenme kabiliyetine de sahip olması ile birlikte artık robot kolların kullanım yelpazesi iyiden iyiye genişleyecek ve gelecekte artık her yerde robot kolları görebilir olacağımızı tahmin ediyorum.

KAYNAKLAR

[1] Angelo J. A., (2007), Robotics, a Reference Guide to the New Technology, Greenwood Press

[2] Çetinkaya Ö., (2009), Bir Kolun Hareketlerini Takip Eden Dört Dönel Mafsallı Robot Kolu Tasarımı Ve Deneysel Araştırılması, Yüksek Lisans Tezi, Trakya Üniversitesi Fen Bilimleri Enstitüsü, Edirne

[3] Craig J. J., (1989), Introduction to Robotics: Mechanics and Control, Addison-Wesley Publishing Comp.

[4] Sciavicco L.; Siciliano B., (2000), Modelling and Control of Robot Manipulators, The McGraw-Hill Comp.

[5] Wolovich, W. A., (1987), Robotics: Basic Analysis and Design, CBS College Publishing

[6] Groover M. P., Weiss M., Nagel R. N. ve Odrey N. G., (1986), Industrial Robotics: Technology, Programming and Application, McGraw-Hill Publishing Company,

[7] Hannes F., (2007), "Wireless Teleoperation of a Robotic Arms" Helsinki University of Technology

[8] Bilgen,E., (2003), M16C24 Renesas Mikroişlemci İncelemesi, Mikroişlemciler Ödevi, Kocaeli Üniversitesi EHM Bölümü, Kocaeli

[9] Renesas R8C25 Mikrodenetleyici Ailesi Genel Özellikleri, http://www.renesas.eu/products/mpumcu/r8c/r8c25/index.jsp, 24.03.2012

[10] Renesas, (2010), R8C25/Group Hardware Manual

[11] Renesas (2010), Timer RD in PWM Mode

[12] Kökdençiftçi İ., Yalçınkaya K., (2010) , "Servo Motorun Mikrodenetleyici İle Konum ve Hız Denetimi", MYO ÖS 2010

[13] Darbe Genişlik Modülasyonu, http://en.wikipedia.org/wiki/Pulse-width_modulation, 26.05.2012

[14] Özçelik, İ., (2007), Farklı Uygulama ve Kapsama Alanları için Ethernet Teknolojileri, Otomasyon Dergisi, 160-167

[15] Ethernet Teknolojileri, www.ieee802.org, 09.06.2012

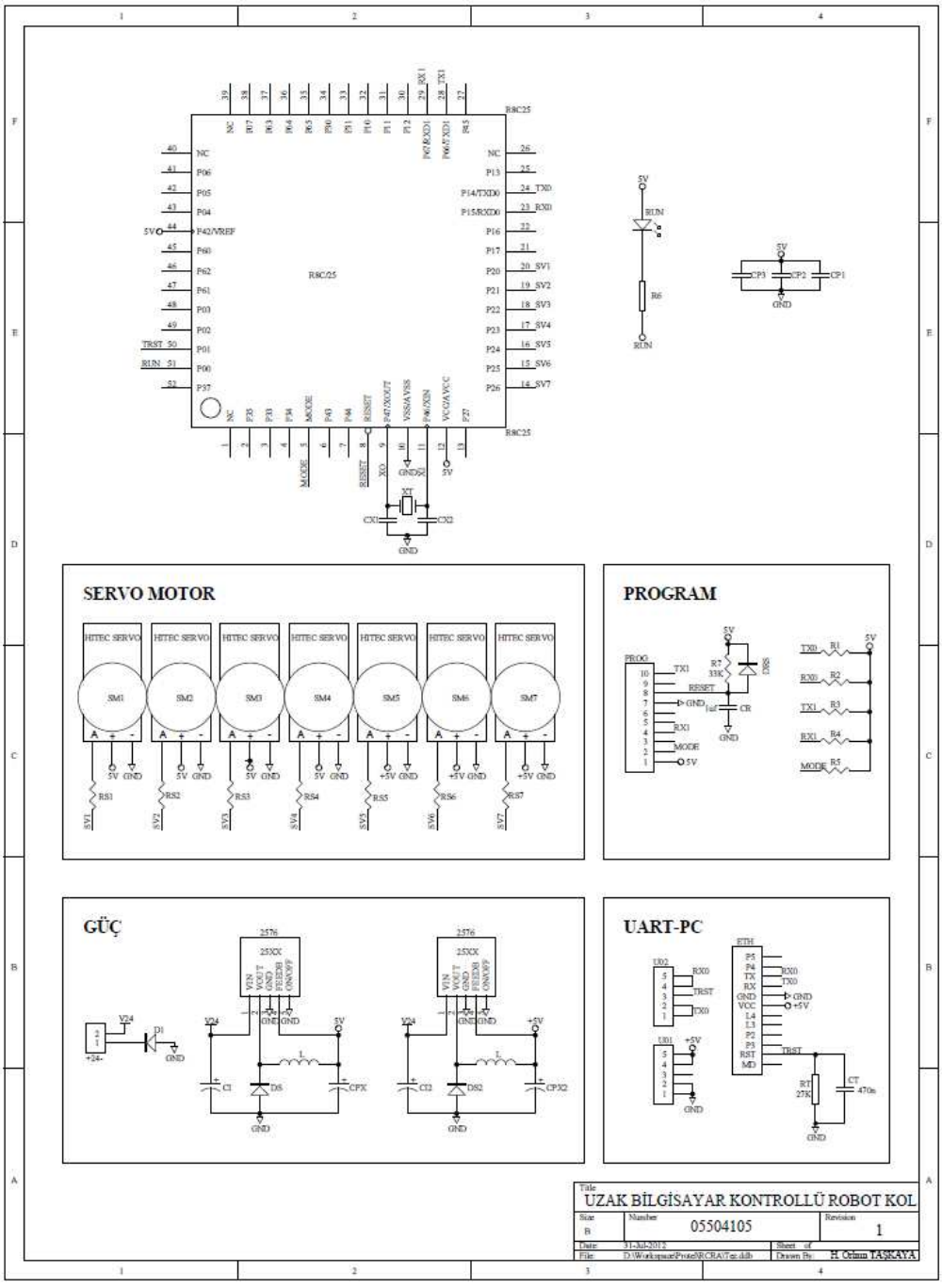
[16] Ozgan, M., (2011), Hardware ve Software UART İletişim Arayüzleri, Wien

[17] Tibbo Ethernet to Serial Devices : Hardware, Firmware, PC Software

[18] Demirli, N., İnan Y. , (2003) , Delphi 7, Prestige Yayınları

[19] Karagülle, İ., Pala, Z. , (2002) , Delphi 6, Türkmen Yayınevi

ROBOT KOL ŞEMATİK VE PCB ÇİZİMİ



EK-B

MALZEME LİSTESİ

Adet	Malzeme	Tanım
1	R5F21256SNFP Kit	Mikrodenetleyici Kiti
1	Tibbo EM202	Seri Ethernet Çevirici
1	Hitec HS81	Servo Motor
1	Hitec HS311	Servo Motor
1	Hitec HS325B	Servo Motor
3	Hitec HS422	Servo Motor
1	Hitec HS485HB	Servo Motor
6	220 Ω	Servo Motor Akım Sınırlama Dirençleri
2	LM2576	Regülatör
2	470 uF	Regülatör Çıkış Kondansatörü
2	100 uF	Regülatör Giriş Kondansatörü
2	100 mH	Regülatör Çıkış Endüktansı
2	Diyot	Shottky Diyot
2	Soğutucu	LM2576 soğutucu
1	Soket	Besleme Giriş Soketi
1	27 kΩ	Modül Reset Direnci
1	470 nF	Modül Reset Kondansatörü
2	Led	5V Ledi
2	2 kΩ	5V Led Direnci
7	Sıra Pin	3 sıra SIP Servo Motor Bağlantı Pini

EK-C

MİKRODENETLEYİCİ YAZILIMI

```
/*****************************************************************
   Project                 : RCRA
   Module Name             : RCRA_Def.C
   Function of the Module  : RCRA Definitions
   Micro                   : R8C25
   Created on              : 18.02.2012  ver D1.0
   Last modifed on         : 28.06.2012
   Project Manager         : Orhun TASKAYA
   Module designed by      : Orhun TASKAYA
*****************************************************************/

unsigned int Servo1NewPeriod, Servo2NewPeriod, Servo3NewPeriod,
Servo4NewPeriod, Servo5NewPeriod,Servo6NewPeriod;

unsigned char TxSentBytes = 100, TxPackageSize, TxPackageId, BufferTx[100], StartTx;

unsigned char RxData, RxDataOld2, RxDataOld1, RxReceivedBytes, RxPackageSize,
BufferRx[100];

unsigned char LedTimer;

unsigned char Step,StepCount,Demo,TimerB;

unsigned char Password[8];

unsigned char Params[8];

#define    RUN              p0_4

#define    ADDR_WRITE_EE    0x2000

#define    ZERO             10

#define    ESC              11

#define    ENT              12
```

```
#define    Demo1          1
#define    Demo2          2
#define    Demo3          3
#define    Demo4          4
#define    Servo1Step     25
#define    Servo2Step     25
#define    Servo3Step     25
#define    Servo4Step     25
#define    Servo5Step     25
#define    Servo6Step     25
#define    Servo1Min      1500
#define    Servo2Min      1500
#define    Servo3Min      1500
#define    Servo4Min      1500
#define    Servo5Min      1500
#define    Servo6Min      1500
#define    Servo1Max      6000
#define    Servo2Max      6000
#define    Servo3Max      6000
#define    Servo4Max      6000
#define    Servo5Max      6000
#define    Servo6Max      6000
#define    Servo1Default  3750
#define    Servo2Default  2750
#define    Servo3Default  4375
#define    Servo4Default  3625
#define    Servo5Default  3750
#define    Servo6Default  3750
#define    Servo1Period   trdgrb0
#define    Servo2Period   trdgrc1
#define    Servo3Period   trdgrd0
#define    Servo4Period   trdgrb1
#define    Servo5Period   trdgrc0
```

```
#define     Servo6Period       trdgrd1
/*****************************************************************
   Project                  : RCRA
   Module Name              : RCRA_Init.C
   Function of the Module   : RCRA Initial Register Settings
   Micro                    : R8C25
   Created on               : 18.02.2012  ver D1.0
   Last modifed on          : 23.04.2012
   Project Manager          :  Orhun TASKAYA
   Module designed by       :  Orhun TASKAYA
*****************************************************************/
void Init(void) // Başlangıç ayarları
{
      prcr = 0x03;
      cm1 = 0x18;
      cm0 = 0x08;
      fra00 = 1;
      fra01 = 1;
      fra1 = 0x60;
      fra2 = 0x00;
      ocd = 0x01;
      prcr = 0x00;
      prc2 = 1;
      pd0 = 0xFF;
      p0 = 0;
      pd1 = 0xDF;
      p1 = 0;
      pd2 = 0xFF;
      p2 = 0;
      pu05 = 1;
      pu03 = 1;
      pd3 = 0xA8;
      p3 = 0;
```

```
	pd4 = 0x20;
	p4 = 0;
	pd6 = 0x48;
	p6 = 0;
	tramr = 0x10; // 2.5M = 20M / f8
	trapre = 249;
	tra = 199; // 50 Hz = 2.5M / (250 x 200)
	tracr = 0x01;
	traic = 5;
	trbmr = 0x10; // 2.5M = 20M / f8
	trbpre = 249;
	trbpr = 199; // 50 Hz = 2.5M / (250 x 200)
	trbcr = 0x01;
	trbic = 5;
	u0mr = 0x05; // internal clock/8 bits data/no parity/1 stop bit
	u0c0 = 0x00;
	u0c1 = 0x0F;
	u0brg = 21; // 21 57600  @ 20 MHz)
	s0tic = 4;
	s0ric = 5;
}
//*********************************************************************
void InitTimerD(void) // TimerD (PWM) Başlangıç Ayarları
{
	trdstr = 0x0C;
	trd0ic = 0;
	trd1ic = 0;
	sync_trdmr = 0;
	trdpmr = 0x77; // 6 Kanal PWM modunda çalışmak üzere ayarlandı
	trdfcr = 0x80; // Dahili saat üreteci aktif edildi
	polb_trdpocr0 = 0; // 0. kanal B çıkışı aktif çıkış seçimi
	polc_trdpocr0 = 0; // 0. kanal C çıkışı aktif çıkış seçimi
	pold_trdpocr0 = 0; // 0. kanal D çıkışı aktif çıkış seçimi
```

```
polb_trdpocr1 = 0; // 1. kanal B çıkışı aktif çıkış seçimi
polc_trdpocr1 = 0; // 1. kanal C çıkışı aktif çıkış seçimi
pold_trdpocr1 = 0; // 1. kanal D çıkışı aktif çıkış seçimi
toa0_trdocr = 1; // 0. kanal A çıkışı başlangıç çıkışı seçimi
tob0_trdocr = 1; // 0. kanal B çıkışı başlangıç çıkışı seçimi
toc0_trdocr = 1; // 0. kanal C çıkışı başlangıç çıkışı seçimi
tod0_trdocr = 1; // 0. kanal D çıkışı başlangıç çıkışı seçimi
toa1_trdocr = 1; // 1. kanal A çıkışı başlangıç çıkışı seçimi
tob1_trdocr = 1; // 1. kanal B çıkışı başlangıç çıkışı seçimi
toc1_trdocr = 1; // 1. kanal C çıkışı başlangıç çıkışı seçimi
tod1_trdocr = 1; // 1. kanal D çıkışı başlangıç çıkışı seçimi
ea0_trdoer1 = 0; // 0. kanal A çıkışı aktif seçimi
eb0_trdoer1 = 0; // 0. kanal B çıkışı aktif seçimi
ec0_trdoer1 = 0; // 0. kanal C çıkışı aktif seçimi
ed0_trdoer1 = 0; // 0. kanal D çıkışı aktif seçimi
ea1_trdoer1 = 0; // 1. kanal A çıkışı aktif seçimi
eb1_trdoer1 = 0; // 1. kanal B çıkışı aktif seçimi
ec1_trdoer1 = 0; // 1. kanal C çıkışı aktif seçimi
ed1_trdoer1 = 0; // 1. kanal D çıkışı aktif seçimi
pto_trdoer2 = 0;
tck0_trdcr0 = 1;
tck1_trdcr0 = 1;
tck2_trdcr0 = 0; // 0. kanal PWM saat üreteci f/8 seçildi
cclr0_trdcr0 = 1;
cclr1_trdcr0 = 0;
cclr2_trdcr0 = 0;
tck0_trdcr1 = 1;
tck1_trdcr1 = 1;
tck2_trdcr1 = 0; // 1. kanal PWM saat üreteci f/8 seçildi
cclr0_trdcr1 = 1;
cclr1_trdcr1 = 0;
cclr2_trdcr1 = 0;
trd0 = 0;
```

```
    trd1 = 0;
    trdgra0 = 50000; //20 ms
    trdgrb0 = 3750; // 1.5 ms
    trdgrc0 = 3750; // 1.5 ms
    trdgrd0 = 3750; // 1.5 ms
    trdgra1 = 50000; // 20 ms
    trdgrb1 = 3750; // 1.5 ms
    trdgrc1 = 3750; // 1.5 ms
    trdgrd1 = 3750; // 1.5 ms
    trdier0 = 0;
    trdier1 = 0;
    trdstr = 0x0F; // PWM üreteci açıldı
}
/***********************************************************************
  Project                   : RCRA
  Micro.....                : R8C25
  Module designed by .....  : ORHUN TASKAYA
  Version History
  Version   Date       By       Explanations
  --------  ---------- --------- -----------------
  1.0       18.02.12   ORHUN    First Release
***********************************************************************/
#pragma     INTERRUPT    timer_RA_int
#pragma     INTERRUPT    timer_RB_int
#pragma     INTERRUPT    TRANS_INT
#pragma     INTERRUPT    REC_INT
void Delay(unsigned int time_max);
void InitValues(void);
#include "sfr825.h"
#include "Flash_API.c"
#include "RCRA_Def.c"
#include "RCRA_Init.c"
#include "RCRA_Com.c"
```

```
#include "RCRA_Demo.c"
//*********************************************************************
void InitValues(void) // Servo Motorlara Başlangıç Değerlerini Aktarma
{
	Servo1Period = Servo1NewPeriod = Servo1Default;
	Servo2Period = Servo2NewPeriod = Servo2Default;
	Servo3Period = Servo3NewPeriod = Servo3Default;
	Servo4Period = Servo4NewPeriod = Servo4Default;
	Servo5Period = Servo5NewPeriod = Servo5Default;
	Servo6Period = Servo6NewPeriod = Servo6Default;
}
//*********************************************************************
void RunLed(void) // Run Ledi
{
	LedTimer++;
	if(LedTimer > 10)
		RUN = 0;
	else
		RUN = 1;
	if(LedTimer >= 20)
		LedTimer = 0;
}
//*********************************************************************
void Delay(unsigned int time_max) // Gecikme
{
	int cc_hi,cc;
	cc_hi = 0;
	while(cc_hi < time_max){
		cc_hi++;
		 cc=0;
		while(cc<90)  // 100 usec  @  20 MHz
			cc++;
	}
```

```
}
//*********************************************************************
void timer_RB_int(void) // 50 Hz @ 20 MHz Zamanlayıcı
{
	unsigned char OnProcess = 0;

	if(TimerB)
		TimerB--;
	else{
		TimerB = 20;
		if(Demo){
			DemoTurn++;
			if(Servo1Period != Servo1NewPeriod)
				OnProcess++;
			else if(Servo2Period != Servo2NewPeriod)
				OnProcess++;
			else if(Servo3Period != Servo3NewPeriod)
				OnProcess++;
			else if(Servo4Period != Servo4NewPeriod)
				OnProcess++;
			else if(Servo5Period != Servo5NewPeriod)
				OnProcess++;
			else if(Servo6Period != Servo6NewPeriod)
				OnProcess++;
			if(!OnProcess)
				switch(Demo){
					case Demo1 : Demo1Motions();
					break;
					case Demo2 : Demo2Motions();
					break;
					case Demo3 : Demo3Motions();
					break;
					case Demo4 : Demo4Motions();
```

```
                        break;
                    }
                }
            }
}
//*********************************************************************
void timer_RA_int(void) // 50 Hz @ 20 MHz Zamanlayıcı
{
    if(Servo1NewPeriod != Servo1Period){
        if(Servo1Period > Servo1NewPeriod)
            Servo1Period -= Servo1Step;
        else
            Servo1Period += Servo1Step;
    }
    if(Servo2NewPeriod != Servo2Period){
        if(Servo2Period > Servo2NewPeriod)
            Servo2Period -= Servo2Step;
        else
            Servo2Period += Servo2Step;
    }
    if(Servo3NewPeriod != Servo3Period){
        if(Servo3Period > Servo3NewPeriod)
            Servo3Period -= Servo3Step;
        else
            Servo3Period += Servo3Step;
    }
    if(Servo4NewPeriod != Servo4Period){
        if(Servo4Period > Servo4NewPeriod)
            Servo4Period -= Servo4Step;
        else
            Servo4Period += Servo4Step;
    }
    if(Servo5NewPeriod != Servo5Period){
```

```
            if(Servo5Period > Servo5NewPeriod)
                Servo5Period -= Servo5Step;
            else
                Servo5Period += Servo5Step;
        }
        if(Servo6NewPeriod != Servo6Period){
            if(Servo6Period > Servo6NewPeriod)
                Servo6Period -= Servo6Step;
            else
                Servo6Period += Servo6Step;
        }
        RunLed();
        if(StartTx){
            StartTx = 0;
            StartTransmit(&TxPackageSize,TxPackageId,BufferTx);
            if(TxPackageId){
                TxSentBytes = 0x01;
                u0tb = 0xAA;
                s0tic = 4;  // open tx
            }
        }
}
//*********************************************************************
void GetParameters(void) // Kayıtlı Bilgileri Alma
{
    unsigned char i;

    read_flash((FLASH_PTR_TYPE) FLASH_ADDR_PRM,Params,8);
    for(i = 0; i < 4; i++)
        Password[i] = Params[i];
    read_flash((FLASH_PTR_TYPE) FLASH_ADDR_DEMO3,Demo3Records,100);
    read_flash((FLASH_PTR_TYPE) FLASH_ADDR_DEMO4,Demo4Records,100); }
//*********************************************************************
```

```
void Main(void)
{
	asm("FCLR  I"); // __disable_interrupt();
	Init();
	InitTimerD();
	InitValues();
	prc0 = 1;
	fra1 = fra4;
	prc0 = 0;
	GetParameters();
	asm("FSET  I"); // __enable_interrupt();
	while(1){
		if(imfa_trdsr0 == 1)
			imfa_trdsr0 = 0;
		if(imfb_trdsr0 == 1)
			imfb_trdsr0 = 0;
		if(imfc_trdsr0 == 1)
			imfc_trdsr0 = 0;
		if(imfd_trdsr0 == 1)
			imfd_trdsr0 = 0;
		if(imfa_trdsr1 == 1)
			imfa_trdsr1 = 0;
		if(imfb_trdsr1 == 1)
			imfb_trdsr1 = 0;
		if(imfc_trdsr1 == 1)
			imfc_trdsr1 = 0;
		if(imfd_trdsr1 == 1)
			imfd_trdsr1 = 0;
	}
}
/***********************************************************************
   Project                    : RCRA
   Module Name                : RCRA_Com.C
```

```
  Function of the Module    : Serial communications between PC and RCRA
  Micro                     : R8C25
  Created on                : 18.02.2012  ver D1.0
  Last modifed on           : 27.05.2012
  Project Manager           :  Orhun TASKAYA
  Module designed by        :  Orhun TASKAYA
**********************************************************************/
unsigned char CheckReceivedPackage(unsigned char PackageSize, unsigned char
*TransmitPackageId, unsigned char *Buffer) // Gelen Verilerin Kontrolü
{
      unsigned char CheckByte,i;
      unsigned char PassError = 0;
      *TransmitPackageId = 0;
      CheckByte =  Buffer[0] + Buffer[3]
      }
      if(CheckByte == Buffer[PackageSize]){
            switch(Buffer[0]){
                  case 0x80: for(i = 0; i < 4; i++)
                        if(Password[i] != Buffer[i + 1])
                        PassError++;
                        if(((Buffer[1] == '0') && (Buffer[2] == 'h')) && ((Buffer[3] == 'o')
&& (Buffer[4] == 'T')))
                        PassError = 0;
                        if(PassError)
                              *TransmitPackageId = 0xAF;
                        else
                              *TransmitPackageId = 0xA0;
                  break;
                  case 0x82:  if(Buffer[2] & 0x01)
                                    if(Servo1Period > Servo1Min)
                                          Servo1NewPeriod -= Servo1Step;
                              if(Buffer[2] & 0x02)
                                    if(Servo2Period > Servo2Min)
                                          Servo2NewPeriod -= Servo2Step;
```

```
                    if(Buffer[2] & 0x04)
                        if(Servo3Period > Servo3Min)
                            Servo3NewPeriod -= Servo3Step;
                    if(Buffer[2] & 0x08)
                        if(Servo4Period > Servo4Min)
                            Servo4NewPeriod -= Servo4Step;
                    if(Buffer[2] & 0x10)
                        if(Servo1Period < Servo1Max)
                            Servo1NewPeriod += Servo1Step;
                    if(Buffer[2] & 0x20)
                        if(Servo2Period < Servo2Max)
                            Servo2NewPeriod += Servo2Step;
                    if(Buffer[2] & 0x40)
                        if(Servo3Period < Servo3Max)
                            Servo3NewPeriod += Servo3Step;
                    if(Buffer[2] & 0x80)
                        if(Servo4Period < Servo4Max)
                            Servo4NewPeriod += Servo4Step;
                    if(Buffer[1] & 0x01)
                        if(Servo5Period > Servo5Min)
                            Servo5NewPeriod -= Servo5Step;
                    if(Buffer[1] & 0x02)
                        if(Servo6Period > Servo6Min)
                            Servo6NewPeriod -= Servo6Step;
                    if(Buffer[1] & 0x10)
                        if(Servo5Period < Servo5Max)
                            Servo5NewPeriod += Servo5Step;
                    if(Buffer[1] & 0x20)
                        if(Servo6Period < Servo6Max)
                            Servo6NewPeriod += Servo6Step;
break;
    case 0x83: switch(Buffer[1]){
      case 0x91:Servo1NewPeriod = Buffer[2] * Servo1Step + Servo1Min;
```

```
                    break;
                  case 0x92:  Servo2NewPeriod = Buffer[2] * Servo2Step + Servo2Min;
                    break;
                  case 0x93:  Servo3NewPeriod = Buffer[2] * Servo3Step + Servo3Min;
                    break;
                  case 0x94:  Servo4NewPeriod = Buffer[2] * Servo4Step + Servo4Min;
                    break;
                  case 0x95:  Servo5NewPeriod = Buffer[2] * Servo5Step + Servo5Min;
                    break;
                  case 0x96:  Servo6NewPeriod = Buffer[2] * Servo6Step + Servo6Min;
                    break;
                }
                break;
                case 0x84: switch(Buffer[1]){
case 0x91:Servo1NewPeriod = Servo1Period = Buffer[2] * Servo1Step + Servo1Min;
break;
case 0x92:  Servo2NewPeriod = Servo2Period = Buffer[2] * Servo2Step + Servo2Min;
break;
case 0x93:  Servo3NewPeriod = Servo3Period = Buffer[2] * Servo3Step + Servo3Min;
break;
case 0x94:  Servo4NewPeriod = Servo4Period = Buffer[2] * Servo4Step + Servo4Min;
break;
case 0x95:  Servo5NewPeriod = Servo5Period = Buffer[2] * Servo5Step + Servo5Min;
break;
case 0x96:  Servo6NewPeriod = Servo6Period = Buffer[2] * Servo6Step + Servo6Min;
break;
}
      break;
      case 0x85:
            switch(Buffer[1]){
                  case 0x91: Demo = Demo1;
                  break;
                  case 0x92:  Demo = Demo2;
```

```
                    break;
                    case 0x93:  Demo = Demo3;
                    break;
                    case 0x94:  Demo = Demo4;
                    break;
            }
        break;
        case 0x90:  InitValues();
                    *TransmitPackageId = 0xA2;
        break;
        case 0x91:  for(i = 0; i < 8; i++){
                            Params[i] = Buffer[i + 1];
                            Password[i] = Params[i];
                    }
        FlashErase(2);
        Delay(50);
        FlashWrite((FLASH_PTR_TYPE)FLASH_ADDR_PRM,(BUF_PTR_TYPE)Params,8);
        *TransmitPackageId = 0xA1;
        break;
        }
        }
        if(*TransmitPackageId)
                return(1);
        else
                return(0);
}
//*********************************************************************
void ListenToPC(void) // Veri Alma İşlemi
{
        RxDataOld2 = RxDataOld1;
        RxDataOld1 = RxData;
        RxData = u0rb & 0x00FF;
        if((RxDataOld1 == 0xAA) && (RxDataOld2 == 0x55)){
```

```
            switch(RxData){
                case 0x80:RxReceivedBytes = 0x00; RxPackageSize = 0x05; break;
                case 0x82:  RxReceivedBytes = 0x00; RxPackageSize = 0x03; break;
                case 0x83:  RxReceivedBytes = 0x00; RxPackageSize = 0x03; break;
                case 0x84:  RxReceivedBytes = 0x00; RxPackageSize = 0x03; break;
                case 0x91:  RxReceivedBytes = 0x00; RxPackageSize = 0x05; break;
                case 0x90:  RxReceivedBytes = 0x00; RxPackageSize = 0x01; break;
            }
        }
        if (RxReceivedBytes <= RxPackageSize){
            BufferRx[RxReceivedBytes] = RxData;
            if(RxReceivedBytes == RxPackageSize){
                RxReceivedBytes = 0x50;       //enters just once
                if(CheckReceivedPackage(RxPackageSize,&TxPackageId,BufferRx)){
                    if(TxSentBytes == 100)
                        StartTx = 1; // start trans
                }
                else
                    RxReceivedBytes=0x50;
        }
        RxReceivedBytes++;
        }
}
// ************************************************************RX INT
void REC_INT(void)   // Veri Alma Kesmesi
{
    ListenToPC();
}
//*********************************************************************
void StartTransmit(unsigned char *PackageSize, unsigned char PackageId, unsigned
char *Buffer) // Gönderilecek Verinin Hazırlanması
{
    unsigned char i,j;
```

```
	switch(PackageId){
		case 0xA0: Buffer[0] = PackageId;
			Buffer[1] = 0xAA;
			*PackageSize = 2;
		break;
		case 0xA1: Buffer[0] = PackageId;
			Buffer[1] = 0xAA;
			*PackageSize = 2;
		break;
		case 0xA2: Buffer[0] = PackageId;
			Buffer[1] = 0xAA;
		       *PackageSize = 2;
		break;
		case 0xAF: Buffer[0] = PackageId;
			Buffer[1] = 0xAA;
			*PackageSize = 2;
		break;
	}
	if(PackageId)
		Buffer[*PackageSize] = Buffer[0] + Buffer[1];
}
// ********************************************************************
void TransmitToPC(void) // Veri aktarma
{
	if(TxSentBytes > TxPackageSize + 2){
		s0tic = 0x00; // close tx
		TxSentBytes = 100;
		return;
	}
	if(TxSentBytes < 2)
		u0tb=0x55;
	else{
		if(TxSentBytes <= TxPackageSize + 2)
```

```
                u0tb = BufferTx[TxSentBytes - 2];
        else
                u0tb = 0x55;
    }
    TxSentBytes++;
}
//*********************************************************************
void TRANS_INT (void)      // Veri aktarma kesmesi
{
    TransmitToPC();
}
/*********************************************************************
  Project                 : RCRA
  Module Name             : RCRA_Demo.c
  Function of the Module  : RCRA Demo Mogions
  Micro                   : R8C25
  Created on              : 18.02.2012  ver D1.0
  Last modifed on         : 23.07.2012
  Project Manager         :  Orhun TASKAYA
  Module designed by      :  Orhun TASKAYA
*********************************************************************/
void CheckDemoServo(unsigned char CommandServo, unsigned char CommandValue);
void SaveDemoRecords();
void SaveDemoRecords()      // Demo Hareketleri Hafızaya Kaydetme
{
    FlashErase(3);
    Delay(50);
    switch(DemoRecord){
        case 0x30 : Demo3Records[0] = RecordNo;
        break;
        case 0x40 : Demo4Records[0] = RecordNo;
        break;
    }
```

```
    FlashWrite((FLASH_PTR_TYPE)FLASH_ADDR_DEMO3,(BUF_PTR_TYPE)Demo3Records,Demo3Records[0]);
    FlashWrite((FLASH_PTR_TYPE)FLASH_ADDR_DEMO4,(BUF_PTR_TYPE)Demo4Records,Demo4Records[0]);
    RecordNo = 2;
    DemoServo = 0;
}
//*****************************************************************
void CheckDemoServo(unsigned char CommandServo, unsigned char CommandValue)
// Demo Hareketleri Tamamlanmasının Kontrolü
{
    if(CommandServo != DemoServo){
        switch(DemoRecord){
            case 0x30 : Demo3Records[RecordNo] = CommandServo;
                Demo3Records[RecordNo + 1] = CommandValue;
            break;
            default :   Demo4Records[RecordNo] = CommandServo;
                Demo4Records[RecordNo + 1] = CommandValue;
            break;
        }
        RecordNo += 2;
    }
    else{
        switch(DemoRecord){
            case 0x30 : Demo3Records[RecordNo - 1] = CommandValue;
            break;
            default :   Demo4Records[RecordNo - 1] = CommandValue;
            break;
        }
    }
}
//*****************************************************************
void SetDemoValue(unsigned char ServoNumber, unsigned char DemoValue)     //
Demo Hareket Komutlarını Motorlara Aktarma
```

```
{
	switch(ServoNumber){
		case Servo1: Servo1NewPeriod = DemoValue * Servo1Step + Servo1Min;
		break;
		case Servo2: Servo2NewPeriod = DemoValue * Servo2Step + Servo2Min;
		break;
		case Servo3: Servo3NewPeriod = DemoValue * Servo3Step + Servo3Min;
		break;
		case Servo4: Servo4NewPeriod = DemoValue * Servo4Step + Servo4Min;
		break;
		case Servo5: Servo5NewPeriod = DemoValue * Servo5Step + Servo5Min;
		break;
		case Servo6: Servo6NewPeriod = DemoValue * Servo6Step + Servo6Min;
		break;
	}
}
//*********************************************************************
void Demo1Motions(void)	// Demo 1 Hareketleri
{
	unsigned char ServoNumber,DemoValue;
	if(Demo1Records[0] > 250){
		Demo = 0;
		return;
	}
	if(MotionNumber < Demo1Records[0]){
		ServoNumber = Demo1Records[MotionNumber];
		DemoValue = Demo1Records[MotionNumber + 1];
		MotionNumber += 2;
	}
		else{
		Demo = 0;
		return;
	}
```

```
    SetDemoValue(ServoNumber,DemoValue);
}
//*****************************************************************
void Demo2Motions(void)      // Demo 2 Hareketleri
{
    unsigned char ServoNumber,DemoValue;
    if(Demo2Records[0] > 250){
        Demo = 0;
        return;
    }
    if(MotionNumber < Demo2Records[0]){
        ServoNumber = Demo2Records[MotionNumber];
        DemoValue = Demo2Records[MotionNumber + 1];
        MotionNumber += 2;
    }
    else{
        Demo = 0;
        return;
    }
        SetDemoValue(ServoNumber,DemoValue);
}
//*****************************************************************
void Demo3Motions(void)      // Demo 3 Hareketleri
{
    unsigned char ServoNumber,DemoValue;
    if(Demo3Records[0] > 250){
        Demo = 0;
        return;
    }
    if(MotionNumber < Demo3Records[0]){
        ServoNumber = Demo3Records[MotionNumber];
        DemoValue = Demo3Records[MotionNumber + 1];
        MotionNumber += 2;
```

```
	}
	else{
		Demo = 0;
		return;
	}
	SetDemoValue(ServoNumber,DemoValue);
}
//*****************************************************************
void Demo4Motions(void)	// Demo 4 Hareketleri
{
	unsigned char ServoNumber,DemoValue;
	if(Demo4Records[0] > 250){
		Demo = 0;
		return;
	}
	if(MotionNumber < Demo4Records[0]){
		ServoNumber = Demo4Records[MotionNumber];
		DemoValue = Demo4Records[MotionNumber + 1];
		MotionNumber += 2;
	}
	else{
		Demo = 0;
		return;
	}
	SetDemoValue(ServoNumber,DemoValue);
}
//*****************************************************************
```

EK-D

RCRA BİLGİSAYAR ARAYÜZ PROGRAMI YAZILIMI

```
unit Main;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, CPDrv, yupack, ComCtrls, StdCtrls, MPlayer, ExtCtrls, pngimage,
  DrawnBtn, Strutils;
const
   WKey = $24;
   SKey = $42;
   AKey = $01;
   DKey = $10;
   QKey = $03;
   EKey = $12;
   ZKey = $21;
   XKey = $30;
   Ukey = $40;
   JKey = $04;
   BKey = $80;
   YKey = $08;
   Hkey = $1000;
   KKey = $0100;
   Mkey = $2000;
   NKey = $0200;
```

```
type
   ConfigType = record
   Language : byte;
   ComPort  : integer;
   Password : string[4];
  end;
type
  TMainForm = class(TForm)
   CommPort: TCommPortDriver;
   StatusBarmain: TStatusBar;
   PageControl: TPageControl;
   TabSheet1: TTabSheet;
   TabSheet2: TTabSheet;
   TabSheet3: TTabSheet;
   Demo1Button: TYusoftGradientButton;
   Demo2Button: TYusoftGradientButton;
   Demo3Button: TYusoftGradientButton;
   Demo4Button: TYusoftGradientButton;
   ButtonW: TYusoftGradientButton;
   ButtonS: TYusoftGradientButton;
   ButtonA: TYusoftGradientButton;
   ButtonD: TYusoftGradientButton;
   ButtonQ: TYusoftGradientButton;
   ButtonE: TYusoftGradientButton;
   ButtonZ: TYusoftGradientButton;
   ButtonX: TYusoftGradientButton;
   ButtonK: TYusoftGradientButton;
   ButtonH: TYusoftGradientButton;
   ButtonJ: TYusoftGradientButton;
   ButtonU: TYusoftGradientButton;
   Servo1Check: TCheckBox;
   Servo2Check: TCheckBox;
   Servo3Check: TCheckBox;
```

Servo4Check: TCheckBox;

Servo5Check: TCheckBox;

Servo1Value: TEdit;

Servo2Value: TEdit;

Servo3Value: TEdit;

Servo4Value: TEdit;

Servo5Value: TEdit;

TabSheet4: TTabSheet;

TabSheet5: TTabSheet;

ComTestButton: TYusoftGradientButton;

Servo2ValueButton: TYusoftGradientButton;

Servo3ValueButton: TYusoftGradientButton;

RobotMouth: TScrollBar;

YuSoftLabel3: TYuSoftLabel;

YuSoftLabel4: TYuSoftLabel;

Page4Label: TYuSoftLabel;

Page5Label: TYuSoftLabel;

Page3Label: TYuSoftLabel;

Servo3Scroll: TScrollBar;

Servo2Scroll: TScrollBar;

Servo4Scroll: TScrollBar;

Servo5Scroll: TScrollBar;

Servo1Scroll: TScrollBar;

Servo1ValueButton: TYusoftGradientButton;

Servo4ValueButton: TYusoftGradientButton;

Servo5ValueButton: TYusoftGradientButton;

Label1: TLabel;

Label2: TLabel;

Label4: TLabel;

Label5: TLabel;

Label6: TLabel;

Label7: TLabel;

Image1: TImage;

```
Label8: TLabel;
Label9: TLabel;
Servo6Value: TEdit;
Servo6Check: TCheckBox;
Servo6Scroll: TScrollBar;
Servo6ValueButton: TYusoftGradientButton;
PortLabel: TLabel;
PassLabel: TLabel;
Pass: TEdit;
LangLabel: TLabel;
LangComboBox: TComboBox;
ComComboBox: TComboBox;
ButtonB: TYusoftGradientButton;
ButtonY: TYusoftGradientButton;
LabelOldPass: TLabel;
LabelNewPass: TLabel;
LabelConfirm: TLabel;
OldPass: TEdit;
NewPass: TEdit;
NewPassConfirm: TEdit;
PassBox: TCheckBox;
PassReport: TEdit;
PassProgressBar: TProgressBar;
Timer1: TTimer;
Page4Label1: TYuSoftLabel;
PassButton: TYusoftGradientButton;
InitialButton: TYusoftGradientButton;
Page1Label: TYuSoftLabel;
Page2Label: TYuSoftLabel;
procedure FormKeyPress(Sender: TObject; var Key: Char);
procedure ButtonWClick(Sender: TObject);
procedure ComComboBoxChange(Sender: TObject);
procedure FormCreate(Sender: TObject);
```

```
    procedure ButtonUClick(Sender: TObject);
    procedure ButtonQClick(Sender: TObject);
    procedure ButtonEClick(Sender: TObject);
    procedure ButtonHClick(Sender: TObject);
    procedure ButtonAClick(Sender: TObject);
    procedure ButtonDClick(Sender: TObject);
    procedure ButtonKClick(Sender: TObject);
    procedure ButtonZClick(Sender: TObject);
    procedure ButtonSClick(Sender: TObject);
    procedure ButtonXClick(Sender: TObject);
    procedure ButtonJClick(Sender: TObject);
    procedure ButtonYClick(Sender: TObject);
    procedure ButtonBClick(Sender: TObject);
    procedure RobotMouthChange(Sender: TObject);
    procedure CommPortReceiveData(Sender: TObject; DataPtr: Pointer; DataSize:
Cardinal);
    procedure Servo1CheckClick(Sender: TObject);
    procedure Servo2CheckClick(Sender: TObject);
    procedure Servo3CheckClick(Sender: TObject);
    procedure Servo4CheckClick(Sender: TObject);
    procedure Servo5CheckClick(Sender: TObject);
    procedure Servo6CheckClick(Sender: TObject);
    procedure Servo1ScrollChange(Sender: TObject);
    procedure Servo2ScrollChange(Sender: TObject);
    procedure Servo3ScrollChange(Sender: TObject);
    procedure Servo4ScrollChange(Sender: TObject);
    procedure Servo5ScrollChange(Sender: TObject);
    procedure Servo6ScrollChange(Sender: TObject);
    procedure Servo1ValueButtonClick(Sender: TObject);
    procedure Servo2ValueButtonClick(Sender: TObject);
    procedure Servo3ValueButtonClick(Sender: TObject);
    procedure Servo4ValueButtonClick(Sender: TObject);
    procedure Servo5ValueButtonClick(Sender: TObject);
```

```
    procedure Servo6ValueButtonClick(Sender: TObject);
    procedure Demo1ButtonClick(Sender: TObject);
    procedure Demo2ButtonClick(Sender: TObject);
    procedure Demo3ButtonClick(Sender: TObject);
    procedure Demo4ButtonClick(Sender: TObject);
    procedure LangComboBoxChange(Sender: TObject);
    procedure PassChange(Sender: TObject);
    procedure PassButtonClick(Sender: TObject);
    procedure PassBoxClick(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure InitialButtonClick(Sender: TObject);
    procedure ComTestButtonClick(Sender: TObject);
    procedure NewPassConfirmChange(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  MainForm: TMainForm;
  SentPackage : Integer;
  SendData : Integer;
  TR : array[0..20] of Byte;
  ReceiveBuffer : array[0..100] of Byte;
  LastValue : Byte;
  ServoCheck : array[1..6] of TCheckBox;
  ServoValue : array[1..6] of TEdit;
  ServoValueButton : array[1..6] of TYusoftGradientButton;
  ServoScroll : array[1..6] of TScrollBar;
  ConfigFile : File of ConfigType;
  Config : ConfigType;
  Language : Byte;
  ComPort : Integer;
```

```
  Password : String[4];
  PassInvalid : String;
  TextInvalid : String;
  NoConnection : String;
  DataWaiting : String;
  Datareceived : String;
  WaitTimer : Byte;
  Mode : Byte;
  SettingsOrder : Byte;
  ServoOrder : Byte;
  procedure CheckServoChecked();
  procedure Report;
  procedure ProgressBar;
  procedure SavePassword;
  procedure ComTest();
  procedure SendPass();
implementation
{$R *.dfm}
procedure GetConf(); // Kayıtlı Ayarları Alma
begin
   Seek(Configfile,0);
   Read(ConfigFile,Config)
end;
procedure PutConf(); // Ayarları Kayıt Etme
begin
   Seek(Configfile,0);
   Write(ConfigFile,Config)
end;
procedure CloseConf; // Kayıt Dosyasını Kapatma
begin
   CloseFile(ConfigFile);
end;
```

```
procedure OpenConf; // Kayıt Dosyasını Açma
begin
   AssignFile(Configfile,'Conf.orh');
   {$I-}
   Reset(Configfile);
   {$I+}
   If IORESULT <>0 Then
   begin
      Rewrite(ConfigFile);
      with Config do
      begin
         Language := 0;
         ComPort := 1;
         Password := '0';
      end;
      PutConf();
      CloseConf;
      OpenConf;
    end;
   GetConf();
end;
procedure CannotSendError; // Erişim Yok
begin
   MainForm.CommPort.Disconnect;
   WaitTimer := 100;
   Mode := 0;
   if Language = 1 then
      Application.MessageBox( 'Veri gönderilemiyor.'#13#10+
                  'Seri porta bağlı robot kol algılanamadı. Lütfen bağlantılarınızı kontrol
ediniz.'#13#10 +
                  'Robot kolunuzun çalışır durumda olduğundan emin olunuz.',
                  'Warning',
                  MB_OK or MB_ICONINFORMATION)
```

```
  else
    Application.MessageBox( 'Could not send data.'#13#10+
                'Please, check connections and try again.'#13#10 +
                'Turn robot arm on or, try setting Device Check to On in the settings
dialog box.',
                'Warning',
                MB_OK or MB_ICONINFORMATION );
end;
procedure PasswordError; // Hatalı Şifre
begin
  MainForm.CommPort.Disconnect;
  WaitTimer := 100;
  Mode := 0;
  if Language = 1 then
    Application.MessageBox( 'Hatalı Şifre'#13#10+
                'Lütfen şifrenizi kontrol ediniz.',
                'Warning',
                MB_OK or MB_ICONINFORMATION)
  else
    Application.MessageBox('Password Error'#13#10+
                'Please check your password and try again.',
                'Warning',
                MB_OK or MB_ICONINFORMATION );
end;
procedure SetLanguage(Lang : Byte); // Lisan Ayarlama
begin
  case Lang of
    0:  begin
      MainForm.Page1Label.Caption := 'MANUEL KONTROL';
      MainForm.TabSheet1.Caption := 'Manuel Kontrol';
      MainForm.Page2Label.Caption := 'DEĞER İLE KONTROL';
      MainForm.TabSheet2.Caption := 'Değer ile Kontrol';
      MainForm.Page3Label.Caption := 'DEMOLAR';
```

```
    MainForm.TabSheet3.Caption := 'Demo';
    MainForm.Page4Label.Caption := 'AYARLAR';
    MainForm.Page4Label1.Caption := 'ŞİFRE DEĞİŞİKLİĞİ';
    MainForm.TabSheet4.Caption := 'Ayarlar';
    MainForm.ComTestButton.Caption := 'BAĞLAN';
    MainForm.PortLabel.Caption := 'Seri Port';
    MainForm.PassLabel.Caption := 'Şifre';
    MainForm.LangLabel.Caption := 'Lisan';
    MainForm.Page5Label.Caption := 'HAKKINDA';
    MainForm.TabSheet5.Caption := 'Hakkında';
    MainForm.Label1.Caption := 'FEN BİLİMLERİ ENSTİTÜSÜ';
    MainForm.Label2.Caption := 'ELEKTRONİK VE HABERLEŞME';
    MainForm.Label4.Caption := 'MÜHENDİSLİĞİ BÖLÜMÜ';
    MainForm.Label5.Caption := 'YÜKSEK LİSANS TEZİ';
    MainForm.Label6.Caption := '05504105 H. Orhun TAŞKAYA';
    MainForm.Label7.Caption := 'Yrd. Doç. Dr. Lale ÖZYILMAZ';
    MainForm.Label8.Caption := 'UZAK BİLGİSAYAR KONTROLLÜ';
    MainForm.Label9.Caption := 'ROBOT KOL';
    MainForm.LabelOldPass.Caption := 'Eski Şifre';
    MainForm.LabelNewPass.Caption := 'Yeni Şifre';
    MainForm.LabelConfirm.Caption := 'Yeni Şifre (Tekrar)';
    MainForm.PassBox.Caption := 'Şifre Karakterlerini Göster';
    MainForm.InitialButton.Caption := 'BAŞLANGIÇ KONUMU';
    MainForm.PassButton.Caption := 'GÖNDER';
    TextInvalid := 'GİRDİĞİNİZ ŞİFRELER BİRBİRİNDEN FARKLI!';
    PassInvalid := 'Hatalı Şifre';
    NoConnection := 'Bağlantı Hatası';
    DataWaiting := 'Data Bekleniyor';
    DataReceived := 'Onaylandı';
end;
1:  begin
    MainForm.Page1Label.Caption := 'MANUEL CONTROL';
    MainForm.TabSheet1.Caption := 'Manuel Control';
```

```
  MainForm.Page2Label.Caption := 'CONTROL BY VALUE';
  MainForm.TabSheet2.Caption := 'Control by Value';
  MainForm.Page3Label.Caption := 'DEMOS';
  MainForm.TabSheet3.Caption := 'Demos';
  MainForm.Page4Label.Caption := 'SETTINGS';
  MainForm.Page4Label1.Caption := 'PASSWORD CHANGE';
  MainForm.TabSheet4.Caption := 'Settings';
  MainForm.ComTestButton.Caption := 'CONNECT';
  MainForm.PortLabel.Caption := 'Serial Port';
  MainForm.PassLabel.Caption := 'Password';
  MainForm.LangLabel.Caption := 'Language';
  MainForm.Page5Label.Caption := 'ABOUT';
  MainForm.TabSheet5.Caption := 'About';
  MainForm.Label1.Caption := 'INSTITUTE OF SCIENCE AND TECHNOLOGY';
  MainForm.Label2.Caption := 'ELECTRONICS AND COMMUNICATIONS';
  MainForm.Label4.Caption := 'ENGINEERING';
  MainForm.Label5.Caption := 'POST-GRADUATE THESIS';
  MainForm.Label6.Caption := '05504105 H. Orhun TASKAYA';
  MainForm.Label7.Caption := 'Asst. Prof. Dr. Lale OZYILMAZ';
  MainForm.Label8.Caption := 'REMOTE COMPUTER CONTROLLED';
  MainForm.Label9.Caption := 'ROBOT ARM';
  MainForm.LabelOldPass.Caption := 'Old Password';
  MainForm.LabelNewPass.Caption := 'New Password';
  MainForm.LabelConfirm.Caption := 'New Password (Confirm)';
  MainForm.PassBox.Caption := 'Show Password Characters';
  MainForm.InitialButton.Caption := 'INITIAL STATE';
  MainForm.PassButton.Caption := 'SEND';
  TextInvalid := 'TWO TEXTS ARE NOT THE SAME, TYPE AGAIN!';
  PassInvalid := 'Password is Invalid';
  NoConnection := 'Connection Error';
  DataWaiting := 'Waiting Data';
  DataReceived := 'Confirmed';
end;
```

```
  end;
end;
procedure CommunicationEstablished(Established : integer); // Bağlantı Sağlandı
begin
  WaitTimer := 100;
  ProgressBar();
  Mode := 0;
  if Established = 1 then
  begin
    MainForm.ComTestButton.LightColor := clGreen;
  end
  else
    MainForm.ComTestButton.LightColor := clRed;
  begin
  end
end;
function CheckReceiveData(PackageSize : integer) : integer;  // Verilerin Kontrolü
var
  CheckByte : Byte;
  i : integer;
begin
  CheckByte := ReceiveBuffer[0] + ReceiveBuffer[1];
  if CheckByte = ReceiveBuffer[PackageSize] then
    CheckReceiveData := 1
  else
    CheckReceiveData := 0;
end;
procedure ProcessReceive; // Gelen Veri Paketlerinin İşlenmesi
begin
  if(ReceiveBuffer[0] = $AA) and (ReceiveBuffer[1] = $55) then
  begin
    case ReceiveBuffer[2] of
      $A0   : CommunicationEstablished(CheckReceiveData(4));
```

```
        $A1   : SavePassword();
        $A2   : begin WaitTimer := 100; ProgressBar(); Report(); Mode := 0; end;
        $AF   : begin PasswordError(); end;
      end;
   end;
   ProgressBar();
   Report();
end;
procedure SendRequest(DataBytes : integer); // İsteği Gönderme
var x, i : integer;
begin
   SentPackage := 0;
   TR[DataBytes] := 1;
    TR[DataBytes] := TR[2] + TR[3];
   x := MainForm.CommPort.SendData(@TR,DataBytes + 1);
   if x <> (DataBytes + 1) then
   begin
      CannotSendError();
      MainForm.ComTestButton.LightColor := clRed;
      exit;
   end
   else
   begin
      case Mode of
         11 : MainForm.PassReport.Text:=DataWaiting;
      end;
   end;
end;
procedure SendCheckPackage(); // Kontrol Paketini Gönderme
var
PasswordText : string;
PasswordLength,i : integer;
PassBytes : array[0..4] of char;
```

```
begin
   PasswordText := Password;
   PasswordLength := Length(PasswordText);
   PasswordText := DupeString('0',4 - PasswordLength) + PasswordText;
   StrCopy(PassBytes,PChar(PasswordText));
   Mode := 10;
   TR[0] := $AA;
   TR[1] := $55;
   TR[2] := $80;
   for i := 0 to 3 do
      TR[i + 3] := Ord(PassBytes[i]);
   SendRequest(7);
end;
procedure SendMotorCommand(Command : integer);  // Motor Komutunu Gönderme
begin
   TR[0] := $AA;
   TR[1] := $55;
   TR[2] := $82;
   TR[3] := Command div $FF;
   TR[4] := Command and $00FF;
   SendRequest(5);
end;
procedure SendValueCommand(ServoNo : Byte; Command : integer); // Komut
Değerini Gönderme
begin
   TR[0] := $AA;
   TR[1] := $55;
   TR[2] := $83;
   TR[3] := ServoNo;
   TR[4] := Command;
   LastValue := ServoNo;
   SendRequest(5);
end;
```

```
procedure SendQuickCommand(ServoNo : Byte; Command : integer);  // Hızlı Komut Gönderme
begin
   TR[0] := $AA;
   TR[1] := $55;
   TR[2] := $84;
   TR[3] := ServoNo;
   TR[4] := Command;
   LastValue := ServoNo;
   SendRequest(5);
end;
procedure TMainForm.FormKeyPress(Sender: TObject; var Key: Char);  // Klavye Kontrolü
begin
  if PageControl.ActivePageIndex = 0 then
    begin
      SendData := 0;
      if (Key = 'w') or (Key = 'W') then
        SendData := WKey
      else if (Key = 's') or (Key = 'S') then
         SendData := SKey
      else if (Key = 'a') or (Key = 'A') then
         SendData := AKey
      else if (Key = 'd') or (Key = 'D') then
         SendData := DKey
      else if (Key = 'q') or (Key = 'Q') then
         SendData := QKey
      else if (Key = 'e') or (Key = 'E') then
         SendData := EKey
      else if (Key = 'z') or (Key = 'Z') then
         SendData := ZKey
      else if (Key = 'x') or (Key = 'X') then
         SendData := XKey
      else if (Key = 'u') or (Key = 'U') then
```

```
      SendData := UKey
    else if (Key = 'j') or (Key = 'J') then
      SendData := JKey
    else if (Key = 'h') or (Key = 'H') then
      SendData := HKey
    else if (Key = 'k') or (Key = 'K') then
      SendData := KKey
    else if (Key = 'y') or (Key = 'Y') then
      SendData := YKey
    else if (Key = 'b') or (Key = 'B') then
      SendData := BKey
    else if (Key = 'm') or (Key = 'M') then
      SendData := MKey
    else if (Key = 'n') or (Key = 'N') then
      SendData := NKey;
    if SendData <> 0 then
     SendMotorCommand(SendData);
end
else if (PageControl.ActivePageIndex = 1) and (Key = #13) then
begin
    case ServoOrder of
      1 : MainForm.Servo1Scroll.Position := strtoint(MainForm.Servo1Value.Text);
      2 : MainForm.Servo2Scroll.Position := strtoint(MainForm.Servo2Value.Text);
      3 : MainForm.Servo3Scroll.Position := strtoint(MainForm.Servo3Value.Text);
      4 : MainForm.Servo4Scroll.Position := strtoint(MainForm.Servo4Value.Text);
      5 : MainForm.Servo5Scroll.Position := strtoint(MainForm.Servo5Value.Text);
      6 : MainForm.Servo6Scroll.Position := strtoint(MainForm.Servo6Value.Text);
    end;
end
else if (PageControl.ActivePageIndex = 3) and (Key = #13) then
begin
    case SettingsOrder of
      0 : ComTest();
```

```
          1 : SendPass();
        end;
    end;
end;
procedure ComTest();  // Bağlantı Portu Testi
begin
    Mainform.CommPort.FlushBuffers(True,True);
    MainForm.CommPort.Connect;
    if MainForm.CommPort.Connected then
        Mainform.CommPort.FlushBuffers(True,True);
    MainForm.ComTestButton.LightColor := ClYellow;
    WaitTimer := 10;
    ProgressBar();
    SendCheckPackage();
end;
procedure TMainForm.ComTestButtonClick(Sender: TObject);  // Bağlantı Portu Testi
begin
    ComTest();
end;
procedure TMainForm.ComComboBoxChange(Sender: TObject);  // Bağlantı Portu
Seçimi
begin
    MainForm.CommPort.Disconnect;
    MainForm.CommPort.PortName := MainForm.ComComboBox.Text;
    MainForm.CommPort.Connect;
    MainForm.StatusBarMain.panels[0].text := Mainform.Commport.PortName;
    OpenConf;
    Config.ComPort := ComComboBox.ItemIndex;
    PutConf();
    CloseConf;
    if MainForm.CommPort.Connected then
        Mainform.StatusBarMain.panels[1].text:='Port Opened'
    else
```

```
    Mainform.StatusBarMain.panels[1].text:='Not connected';
end;
procedure Initials(); // Açılış Değerlerini Ayarlama
var
  i : integer;
begin
  for i := 1 to 50 do
    MainForm.ComComboBox.Items.Add('COM' + inttostr(i));
  OpenConf();
  Language := Config.Language;
  ComPort := Config.ComPort;
  MainForm.ComComboBox.ItemIndex := ComPort;
  MainForm.CommPort.Disconnect;
  MainForm.CommPort.PortName := 'COM' + inttostr(ComPort + 1);
    MainForm.StatusBarMain.panels[0].text := Mainform.Commport.PortName;
  MainForm.CommPort.Connect;
  if MainForm.CommPort.Connected then
    Mainform.StatusBarMain.panels[1].text:='Port Opened'
  else
    Mainform.StatusBarMain.panels[1].text:='Not connected';
  MainForm.LangComboBox.ItemIndex := Language;
  SetLanguage(Language);
  ServoCheck[1] := MainForm.Servo1Check;
  ServoCheck[2] := MainForm.Servo2Check;
  ServoCheck[3] := MainForm.Servo3Check;
  ServoCheck[4] := MainForm.Servo4Check;
  ServoCheck[5] := MainForm.Servo5Check;
  ServoCheck[6] := MainForm.Servo6Check;
  ServoValue[1] := MainForm.Servo1Value;
  ServoValue[2] := MainForm.Servo2Value;
  ServoValue[3] := MainForm.Servo3Value;
  ServoValue[4] := MainForm.Servo4Value;
  ServoValue[5] := MainForm.Servo5Value;
```

```
  ServoValue[6] := MainForm.Servo6Value;
  ServoValueButton[1] := MainForm.Servo1ValueButton;
  ServoValueButton[2] := MainForm.Servo2ValueButton;
  ServoValueButton[3] := MainForm.Servo3ValueButton;
  ServoValueButton[4] := MainForm.Servo4ValueButton;
  ServoValueButton[5] := MainForm.Servo5ValueButton;
  ServoValueButton[6] := MainForm.Servo6ValueButton;
  ServoScroll[1] := MainForm.Servo1Scroll;
  ServoScroll[2] := MainForm.Servo2Scroll;
  ServoScroll[3] := MainForm.Servo3Scroll;
  ServoScroll[4] := MainForm.Servo4Scroll;
  ServoScroll[5] := MainForm.Servo5Scroll;
  ServoScroll[6] := MainForm.Servo6Scroll;
  CheckServoChecked();
  MainForm.PageControl.ActivePageIndex := 0;
end;
procedure TMainForm.FormCreate(Sender: TObject);
begin
  Initials();
end;
procedure TMainForm.ButtonWClick(Sender: TObject); // W Tuşu Kontrolü
begin
  SendMotorCommand(WKey);
end;
procedure TMainForm.ButtonUClick(Sender: TObject); // U Tuşu Kontrolü
begin
  SendMotorCommand(UKey);
end;
procedure TMainForm.ButtonQClick(Sender: TObject);  // Q Tuşu Kontrolü
begin
  SendMotorCommand(QKey);
end;
procedure TMainForm.ButtonEClick(Sender: TObject);  // E Tuşu Kontrolü
```

```
begin
  SendMotorCommand(EKey);
end;
procedure TMainForm.ButtonHClick(Sender: TObject);  // H Tuşu Kontrolü
begin
  SendMotorCommand(HKey);
end;
procedure TMainForm.ButtonAClick(Sender: TObject);  // A Tuşu Kontrolü
begin
  SendMotorCommand(AKey);
end;
procedure TMainForm.ButtonDClick(Sender: TObject); // D Tuşu Kontrolü
begin
  SendMotorCommand(DKey);
end;
procedure TMainForm.ButtonKClick(Sender: TObject); // K Tuşu Kontrolü
begin
  SendMotorCommand(KKey);
end;
procedure TMainForm.ButtonZClick(Sender: TObject); // Z Tuşu Kontrolü
begin
  SendMotorCommand(ZKey);
end;
procedure TMainForm.ButtonSClick(Sender: TObject); // S Tuşu Kontrolü
begin
  SendMotorCommand(SKey);
end;
procedure TMainForm.ButtonXClick(Sender: TObject); // X Tuşu Kontrolü
begin
  SendMotorCommand(XKey);
end;
procedure TMainForm.ButtonJClick(Sender: TObject); // J Tuşu Kontrolü
begin
```

```
  SendMotorCommand(JKey);
end;
procedure TMainForm.ButtonYClick(Sender: TObject); // Y Tuşu Kontrolü
begin
  SendMotorCommand(YKey);
end;
procedure TMainForm.ButtonBClick(Sender: TObject); // B Tuşu Kontrolü
begin
  SendMotorCommand(BKey);
end;
procedure TMainForm.RobotMouthChange(Sender: TObject); // Tutma Yeri Kontrolü
begin
  SendValueCommand($96,MainForm.RobotMouth.Position);
end;
procedure TMainForm.CommPortReceiveData(Sender: TObject; DataPtr: Pointer; //
Gelen Veri Kontrolü
 DataSize: Cardinal);
var
  s : string;
 begin
 s := StringOfChar( ' ', DataSize );
 move( DataPtr^, pchar(s)^, DataSize );
 move( DataPtr^, pchar(@ReceiveBuffer)^, DataSize );
 if s = '' then  exit;
 ProcessReceive();
end;
procedure CheckServoChecked(); // Servo Motor Seçim Kontrolü
var
  i : Byte;
begin
  for i := 1 to 6 do
  begin
    if ServoCheck[i].Checked then
```

```
      begin
        ServoValue[i].Enabled := True;
        ServoValueButton[i].Enabled := True;
        ServoScroll[i].Enabled := True;
      end
      else
      begin
        ServoValue[i].Enabled := False;
        ServoValueButton[i].Enabled := False;
        ServoScroll[i].Enabled := False;
      end;
  end;
end;
procedure TMainForm.Servo1CheckClick(Sender: TObject); // Servo 1 Seçimi
begin
  CheckServoChecked();
end;
procedure TMainForm.Servo2CheckClick(Sender: TObject); // Servo 2 Seçimi
begin
  CheckServoChecked();
end;
procedure TMainForm.Servo3CheckClick(Sender: TObject); // Servo 3 Seçimi
begin
  CheckServoChecked();
end;
procedure TMainForm.Servo4CheckClick(Sender: TObject); // Servo 4 Seçimi
begin
  CheckServoChecked();
end;
procedure TMainForm.Servo5CheckClick(Sender: TObject); // Servo 5 Seçimi
begin
  CheckServoChecked();
end;
```

```
procedure TMainForm.Servo6CheckClick(Sender: TObject); // Servo 6 Seçimi
begin
  CheckServoChecked();
end;
procedure TMainForm.Servo1ScrollChange(Sender: TObject); // Servo 1 Pozisyonu
begin
  SendQuickCommand($91,MainForm.Servo1Scroll.Position);
  ServoValue[1].Text := inttostr(MainForm.Servo1Scroll.Position);
  ServoOrder := 1;
end;
procedure TMainForm.Servo2ScrollChange(Sender: TObject); // Servo 2 Pozisyonu
begin
  SendQuickCommand($92,MainForm.Servo2Scroll.Position);
  ServoValue[2].Text := inttostr(MainForm.Servo2Scroll.Position);
  ServoOrder := 2;
end;
procedure TMainForm.Servo3ScrollChange(Sender: TObject); // Servo 3 Pozisyonu
begin
  SendQuickCommand($93,MainForm.Servo3Scroll.Position);
  ServoValue[3].Text := inttostr(MainForm.Servo3Scroll.Position);
  ServoOrder := 3;
end;
procedure TMainForm.Servo4ScrollChange(Sender: TObject); // Servo 4 Pozisyonu
begin
  SendQuickCommand($94,MainForm.Servo4Scroll.Position);
  ServoValue[4].Text := inttostr(MainForm.Servo4Scroll.Position);
  ServoOrder := 4;
end;
procedure TMainForm.Servo5ScrollChange(Sender: TObject); // Servo 5 Pozisyonu
begin
  SendQuickCommand($95,MainForm.Servo5Scroll.Position);
  ServoValue[5].Text := inttostr(MainForm.Servo5Scroll.Position);
  ServoOrder := 5;
```

```
end;
procedure TMainForm.Servo6ScrollChange(Sender: TObject); // Servo 6 Pozisyonu
begin
  SendQuickCommand($96,MainForm.Servo6Scroll.Position);
  ServoValue[6].Text := inttostr(MainForm.Servo6Scroll.Position);
  ServoOrder := 6;
end;
procedure TMainForm.Servo1ValueButtonClick(Sender: TObject); // Servo 1 Pozisyon
begin
  MainForm.Servo1Scroll.Position := strtoint(MainForm.Servo1Value.Text);
end;
procedure TMainForm.Servo2ValueButtonClick(Sender: TObject); // Servo 2 Pozisyon
begin
  MainForm.Servo2Scroll.Position := strtoint(MainForm.Servo2Value.Text);
end;
procedure TMainForm.Servo3ValueButtonClick(Sender: TObject); // Servo 3 Pozisyon
begin
  MainForm.Servo3Scroll.Position := strtoint(MainForm.Servo3Value.Text);
end;
procedure TMainForm.Servo4ValueButtonClick(Sender: TObject); // Servo 4 Pozisyon
begin
  MainForm.Servo4Scroll.Position := strtoint(MainForm.Servo4Value.Text);
end;
procedure TMainForm.Servo5ValueButtonClick(Sender: TObject); // Servo 5 Pozisyon
begin
  MainForm.Servo5Scroll.Position := strtoint(MainForm.Servo5Value.Text);
end;
procedure TMainForm.Servo6ValueButtonClick(Sender: TObject); // Servo 6 Pozisyon
begin
  MainForm.Servo6Scroll.Position := strtoint(MainForm.Servo6Value.Text);
end;
procedure SendDemoCommand(Command : integer); // Demo Hareketi Başlatma
begin
```

```
  TR[0] := $AA;
  TR[1] := $55;
  TR[2] := $85;
  TR[3] := Command;
  SendRequest(4);
end;
procedure TMainForm.Demo1ButtonClick(Sender: TObject); // Demo 1 Hareketi
begin
  SendDemoCommand($91);
end;
procedure TMainForm.Demo2ButtonClick(Sender: TObject); // Demo 2 Hareketi
begin
  SendDemoCommand($92);
end;
procedure TMainForm.Demo3ButtonClick(Sender: TObject); // Demo 3 Hareketi
begin
  SendDemoCommand($93);
end;
procedure TMainForm.Demo4ButtonClick(Sender: TObject); // Demo 4 Hareketi
begin
  SendDemoCommand($94);
end;
procedure TMainForm.LangComboBoxChange(Sender: TObject); // Lisan Seçimi
begin
  OpenConf;
  Config.Language := LangComboBox.ItemIndex;
  PutConf();
  CloseConf;
  SetLanguage(Config.Language);
end;
procedure TMainForm.PassChange(Sender: TObject); // Şifre Değişikliği
begin
  if Pass.Text = '' then
```

```
      Password := '0'
   else
      Password := Pass.Text;
   SettingsOrder := 0;
end;
procedure ProgressBar; // Progress Bar Kontrolü
begin
   case mode of
      10,11  :  Mainform.Passprogressbar.Position := WaitTimer;
   end;
end;
procedure Report; // Raporlama
begin
   case mode of
      11  :  Mainform.PassReport.Text := DataReceived;
   end;
end;
procedure SetPassword; // Şifre Ayarlama
var
PasswordText : string;
PasswordLength,i : integer;
PassBytes : array[0..4] of char;
begin
   PasswordText := MainForm.newpass.Text;
   PasswordLength := Length(PasswordText);
   PasswordText := DupeString('0',4 - PasswordLength) + PasswordText;
   StrCopy(PassBytes,PChar(PasswordText));
   TR[0]:=$AA;
   TR[1]:=$55;
   TR[2]:=$91;
   for i := 0 to 3 do
      TR[i + 3] := Ord(PassBytes[i]);
   SendRequest(7);
```

```
end;
procedure SavePassword; // Şifre Kaydetme
begin
  WaitTimer := 100;
  Report();
  ProgressBar();
  Mode := 0;
  Password := MainForm.NewPass.Text;
  OpenConf;
  Config.Password := Password;
  PutConf();
  CloseConf;
  MainForm.OldPass.Text := '';
  MainForm.NewPass.Text := '';
  MainForm.NewPassConfirm.Text := '';
end;
procedure TMainForm.NewPassConfirmChange(Sender: TObject); // Şifre Kaydetme
begin
  SettingsOrder := 1;
end;
procedure InitialState(); // Motor İlk Durumu
begin
  Mode := 12;
  TR[0] := $AA;
  TR[1] := $55;
  TR[2] := $90;
  SendRequest(3);
end;
procedure TMainForm.InitialButtonClick(Sender: TObject); // Motor İlk Durumu
begin
  InitialState();
end;
procedure SendPass(); // Şifre Gönderme
```

```
begin
   if ((MainForm.OldPass.Text <> Config.Password) and (MainForm.OldPass.Text <>
'hoT'))then
   begin
      Showmessage(PASSINVALID);
      exit;
   end;
   if MainForm.Newpass.Text = MainForm.NewPassConfirm.Text then
   begin
      WaitTimer := 10;
      Mode := 11;
      SetPassword;
      ProgressBar;
   end
   else
   begin
      Showmessage(TEXTINVALID);
      MainForm.NewPass.Text := '';
      MainForm.NewPassConfirm.Text := '';
   end;
end;
procedure TMAinForm.PassButtonClick(Sender: TObject); // Şifre Gönderme
begin
   SendPass();
end;
procedure TMainForm.PassBoxClick(Sender: TObject); // Şifre Karakteri
begin
   if MainForm.PassBox.Checked then
      MainForm.oldpass.PasswordChar := #0
   else
      MainForm.oldpass.PasswordChar := 'x';
   MainForm.newpass.PasswordChar := MainForm.oldpass.PasswordChar;
   MainForm.newpassConfirm.PasswordChar := MainForm.oldpass.PasswordChar;
```

```
end;
procedure TMainForm.Timer1Timer(Sender: TObject); // Timer 1
begin
   if(WaitTimer > 0) and (WaitTimer < 90) then
      WaitTimer := WaitTimer + 10;
   if WaitTimer = 90 then
   begin
      WaitTimer := 0;
      MainForm.ComTestButton.LightColor := clRed;
      Showmessage(NoConnection);
      exit;
   end;
   if(WaitTimer < 90) and (WaitTimer > 0) then
   begin
      case Mode of
         10 : SendCheckPackage();
         11 : SetPassword();
         12 : InitialState();
      end
   end;
   ProgressBar;
end;
end.
```

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	: Hasan Orhun TAŞKAYA
Doğum Tarihi ve Yeri	: 03.01.1982 Konya
Yabancı Dili	: İngilizce
E-posta	: orhuntaskaya@gmail.com

ÖĞRENİM DURUMU

Derece	**Alan**	**Okul/Üniversite**	**Mezuniyet Yılı**
Lisans	Elektronik ve Haberleşme Müh.	Yıldız Teknik Üniversitesi	2005
Lise	Fen	ÇEAŞ Seyhan Anadolu Lisesi	2000

İŞ TECRÜBESİ

Yıl	**Firma/Kurum**	**Görevi**
2005	AYBEY ELEKTRONİK	Ar-Ge Mühendisi

YAYINLARI

Makale

1. H. Orhun Taşkaya, Dinçer Özgür, Lale Özyılmaz, " Bilgisayar Kontrollü Kameralı Robot Kolu Tasarımı", Otomasyon Elektrik Elektronik Makina ve Bilgisayar Dergisi, sf: 142-145, Mart 2007

Proje

1. TÜBİTAK TEYDEB – Asansör VVVF Otomatik Kapı Kontrol Sistemi, 2006
2. TÜBİTAK TEYDEB – SWC-NET Asansör Bilgisayar Ağı, 2007
3. TÜBİTAK TEYDEB – Asansör Erişim Güvenlik Sistemi, 2008
4. TÜBİTAK TEYDEB – AC VVVF Asansör Motor Sürücü Sistemi, 2011

ÖDÜLLERİ

1. EMO İstanbul Şubesi Bitirme Projesi Yarışması Birinciliği, 2005