

T.C
MARMARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BULANIK KONTROLLÜ DC MOTOR
EĞİTİM SETİNİN OLUŞTURULMASI

HAZIRLAYAN
SADIK OKUYUCU

TEZ DANIŞMANI
YRD. DOÇ. DR. A. FEVZİ BABA

ELEKTRONİK-BİLGİSAYAR EĞİTİMİ
YÜKSEK LİSANS TEZİ

104665

104665

İSTANBUL 2001



T.C
MARMARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BULANIK KONTROLLÜ DC MOTOR
EĞİTİM SETİNİN OLUŞTURULMASI

T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ

SADIK OKUYUCU

DANIŞMAN VE KOORDİNATÖR : Yrd. Doç. Dr. Fevzi BABA

JÜRİ ÜYESİ : Prof. Dr. Burhanettin CAN

JÜRİ ÜYESİ : Yrd. DoçDr. Reşit ERÇETİN

ELEKTRONİK-BİLGİSAYAR EĞİTİMİ
YÜKSEK LİSANS TEZİ

İSTANBUL 2001

TEŐEKKÜR

Akademik alıŐmalarımın her safhasında danıŐmanlıđımı yapan ve her tŸrlŸ yardımlarını esirgemeyen deđerli hocam Sayın Yrd. Do. Dr. A. Fevzi BABA'ya, alıŐmalarımda yol gŸsteren Sayın Prof. Dr. Burhannettin CAN'a, hocalarım Yrd. Do. Dr. Hasan ERDAL'a, ArŐ GŸr A.Emin KUZUCUOđLU'na, ArŐ. GŸr. Mustafa ONAT'a, alıŐmalarım esnasında yardımlarını esirgemeyen ArŐ. GŸr. Erkan KAPLANOđLU'na, programın yazım aŐamasında elinden gelen desteđi esirgemeyen arkadaŐım Orion bilgisayar Ltd. Őti. Teknik Mdr. YŸksek Teknik Ÿđretmen Serkan DOđAN'a, tezin bŸtŸn aŐamalarında beni yalnız bırakmayan deđerli arkadaŐım Sadık AYHANA'a, adını burada belirlemediđim tŸm dostlarıma ve fedakâr davranıŐlarından dolayı anneme, babama ve eŐime teŐekkŸr ederim.



ÖZET

Bu çalışmada Marmara Üniversitesi Teknik Eğitim Fakültesi Elektronik-Bilgisayar Eğitimi Bölümü Kontrol Ana Bilim Dalı laboratuvarında elektrik makineleri FH2 MkIV eğitim setinde bulunan DC motor, Bulanık Mantık kontrol (Fuzzy Logic Controller) yöntemiyle kontrol edilecektir. DC motor, endüstriyel süreçlerde, ev ve eğlence araç gereçlerinde sıkça kullanılmaktadır. DC motor kontrolünde, ON-OFF, PID(Proportional, Integral, Derivative), MRAC(Model Reference Adaptive Controller) gibi birçok kontrol yöntemi uygulanmaktadır. Son yıllarda Bulanık mantık kontrol yöntemleri hızla yaygınlaşmakta diğer kontrol tekniklerinin yerini almaktadır.

Bulanık mantık, pazarlama için anahtar bir sözcük haline gelmiştir. Bir çok ev ve iş yerlerinde kullanılan elektronik cihazların bulanık kontrol ile denetlendiğinin belirtilmesi bir moda halini almıştır. Bulanık mantık yeterliliğinden yoksun elektronik aletler aşamalı olarak ölü bir stoğa dönüşmektedir.

FH2 MkIV Eğitim Seti, DC, AC, fırçasız ve adım motorlarını içeren eğitim ve araştırma amaçlı tasarlanmış bir sistemdir. Mevcut set trifaze uygunlaştırıcı (set İngiliz menşeli olduğundan) transformator üzerinden üçfazla beslenmektedir. DC motor beslemesi için gerekli olan enerji set içindeki anahtarlamalı güç kaynağı tarafından sağlanmaktadır. DC motor set üzerinde bulunan fren dinamometresine mili vasıtasıyla kuple edilmekte, hız bilgisi gösterge üzerinden izlenebilmekte ve 5 li bir DIN soket uçlarından analog olarak alınabilmektedir. FH2 MkIV Seti ile DC motoru açık çevrim çeşitli konumlarda çalıştırmak mümkün olmakla birlikte herhangi bir kontrol sistemi içermemektedir.

DC motorun kontrolü, bilgisayarın seri portu üzerinden bulanık kontrol ile kontrol edilmesi sağlanacaktır. Bu amaçla bir mikro denetleyici kart tasarlanacaktır. Yazılım olarak Delphi programlama dili kullanılarak kontrol eğitiminde kullanılacak genel amaçlı bulanık kontrolör yazılımı gerçekleştirilecektir. Tasarlanan bulanık kontrolörün girişleri hız ve hız değişimi olarak seçilecektir. Çıkışta ise mikrodenetleyici kart vasıtasıyla motorun istenilen hızda dönmesi sağlanacaktır. Ayrıca bilgisayar üzerindeki yazılım, bulanık mantığın kavranması düşünülerek eğitim amaçlı tasarlanacaktır.

Endüstrinin ve Mesleki Eğitimin ihtiyaç duyduğu güncel bilgilerle donatılmış nitelikli elemanın yetiştirilmesi kuşkusuz Teknik Eğitimin ana hedefidir. Yapılan

alıřmada bilgisayar da tasarlanan grsel ve uygulamalı eđitim amalı yazılım sayesinde bir DC motorun bulanık mantık ile kontrol edilmesi sađlanacaktır.



ABSTRACT

In this study, the DC Motor on FH2 MkIV Training set which is in the laboratory of Electronics and Computer Education department of Technical Education Faculty in Marmara University will be controlled by using Fuzzy Logic. DC motors are used widely in industry, home and office machines, toys and entertainment equipment etc. Those systems are usually controlled by ON-OFF, PID or MRAC. Today's Fuzzy control techniques are very popular in the world.

Fuzzy Logic is a key word for marketing. To notify "with Fuzzy Controlled..." home or office machines became a fashion. Without Fuzzy logic control machines will be dead stock step by step.

FH2 MkIV Training set designed for education and research which includes DC, AC, Brushless motors fundamentals. DC motor supplies from a switchmode power supply in the set. The motor coupling with its pivot to Eddy Current Dynamometer Brake. The speed data can read on a scale and take out from a 5 pin DIN socket for a computer. The motor can operate with open loop on FH2 MkIV Training set but doesn't include any control method.

To control the motor, designed a microcontroller card which communicates a computer on RS-232 port and written a software on the pc. The speed data will read from port by microcontroller card. The data will process here and then will send a signal to control the motor.

The main target of Technical Education is graduate with new technologies. In this study, to control a DC motor with fuzzy, written on a PC which is visual and applicable software.

ŞEKİLLER TABLOSU

Şekil 2-1 Klasik mantık ile fan hızının kontrolü	5
Şekil 2-2 Bulanık mantık ile fan hızının kontrol bulanıklaştırılması.....	6
Şekil 2-3 Venn şeması ile gösterim.....	7
Şekil 2-4 "S" tipi üyelik fonksiyonu	10
Şekil 2-5 Çan eğrisi tipi üyelik fonksiyonu	10
Şekil 2-6 Üçgen fonksiyonu	11
Şekil 2-7 Bir fan motoruna ait devir hızının sözel ifadesi	12
Şekil 2-8 İki bulanık kümenin kesişimi.....	13
Şekil 2-9 İki bulanık kümenin birleşimi.....	13
Şekil 2-10 Bir bulanık kümenin tümleyeni	14
Şekil 2-11 Klasik küme ile bulanık kümenin karşılaştırılması	15
Şekil 3-1 Bulanık kontrolörün yapısı[.].....	18
Şekil 4-1 Bulanık mantık ile kontrol edilen set	23
Şekil 4-2 A Alanı.....	24
Şekil 4-3 5 iğneli DIN soketi	25
Şekil 4-4 Güç kaynağı bağlantı yapısı	27
Şekil 4-5 Kontrol düğmeleri.....	28
Şekil 4-6 Referans hız ile ölçülen hız arasındaki ilişki.....	30
Şekil 4-7 Çeşitli hızlara ait DC motorun davranışı.....	31
Şekil 4-8 Çeşitli aralıktaki hızların "0" a düşüş eğrileri.....	31
Şekil 4-9 Hızın 0 dan 1300 rpme çıkış ve 1300 rpm den 0 a inişi.....	32
Şekil 4-10 Hızın 0 dan 720 rpme çıkış ve 720 rpm den 0 a inişi.....	32
Şekil 5-1 Sistem için tasarlanan kontrolörün genel yapısı	33
Şekil 5-2 Mikro Denetleyici kartın yapısı	34
Şekil 5-3 Bilgisayarlarda bulunan 9 pin.....	35
Şekil 5-4 9pin konnektor bağlantı uçları	35
Şekil 5-5 Mikro denetleyici kartında kullanılan RS232 sürücü devresi	37
Şekil 5-6 Mikro denetleyici katı	38
Şekil 5-7 Analog giriş katı	39
Şekil 5-8 DAC katı	39
Şekil 5-9 Kontrol sisteminde kullanılan üyelik fonksiyonu.....	40
Şekil 5-10 Hız'a ait üyelik fonksiyonları.....	42
Şekil 5-11 Hız değişiminin bulanıklaştırılması.....	42
Şekil 5-12 Programın basitleştirilmiş akış diyagramı.....	46
Şekil 5-13 Bulanık kontrolör işlemlerinin yapıldığı program.....	47
Şekil 5-14 Üye penceresi	48
Şekil 5-15 Üyelik fonksiyonlarının seçilmesi	49
Şekil 5-16 Üyelik fonksiyonlarının oluşturulması.....	49
Şekil 5-17 Üyelik fonksiyonlarının sınır değerlerinin ayarlanması.....	50
Şekil 5-18 Kuralların belirlenmesi	51
Şekil 5-19 Grafik çıktıları.....	51
Şekil 6-1 700 rpm'e ait hız grafiği	53
Şekil 6-2 Motorun 700 rpm, 1000 rpm ve 1200 rpm deki davranışı	54
Şekil 6-3 Hızın 1300 rpm'den 500 rpm'e indirilmesi.....	54
Şekil 6-4 Hızın 435 rpm'e çıkarılması	52

BÖLÜM 3 BULANIK KONTROLÖR	16
3.1 Giriş.....	16
3.2 BULANIK KONTROLÖR	16
3.3 BULANIK KONTROLÖRÜN YAPISI.....	17
3.3.1 <i>Bulanıklaştırma Birimi</i>	18
3.3.2 <i>Bilgi Tabanı</i>	19
3.3.3 <i>Çıkarım Ünitesi</i>	19
3.3.4 <i>Berraklaştırma Birimi</i>	21
BÖLÜM 4 ELEKTRİK MAKİNELERİ EĞİTİM SETİ (FH2 MKIV)	23
4.1 Giriş.....	23
4.2 SİSTEMİN GENEL YAPISI	23
4.3 A ALANI	24
4.4 B ALANI – MAKİNE BAĞLANTI SOKETLERİ	25
4.5 E ALANI – D.C. GÜÇ KAYNAĞI.....	26
4.6 F ALANI – D.C. MOTOR KONTROLÜ.....	27
4.7 FH50 DC MOTORUN DAVRANIŞININ İNCELENMESİ.....	29
4.7.1 <i>Yapılan Deneyler</i>	29
BÖLÜM 5 BULANIK KONTROLÖR TASARIMI.....	33
5.1 Giriş.....	33
5.2 TASARLANAN KONTROLÖRÜN GENEL YAPISI.....	33
5.3 MİKRO DENETLEYİCİ KART	34
5.3.1 <i>RS-232 Dönüştürücü Birimi</i>	34
5.3.2 <i>Mikro Denetleyici Katı</i>	37
5.3.3 <i>ADC (Analog Dijital Çevirici) Katı</i>	38
5.3.4 <i>DAC (Dijital Analog Çevirici) Katı</i>	39
5.3.5 <i>Motor Sürücü</i>	39
5.4 BULANIKLAŞTIRMA İŞLEMLERİ	40
5.4.1 <i>Üyelik Fonksiyonlarının Seçimi</i>	40
5.4.2.1 <i>Hızın Bulanıklaştırılması</i>	41
5.4.2.2 <i>Hız değişiminin Bulanıklaştırılması</i>	42
5.4.2 <i>Kural Tabanı</i>	43
5.4.2.1 <i>Bulanık Kontrol Kuralları</i>	43
5.5 KONTROL İŞARETİNİN BULUNMASI.....	44
5.5.1 <i>Kontrol İşaretinin Berraklaştırılması</i>	45
5.6 BİLGİSAYAR YAZILIMI	45
5.6.1 <i>Referans bilgi girişi</i>	47
5.6.2 <i>Üyelik Fonksiyonları Penceresi</i>	48
5.6.2.1 <i>Üyelik fonksiyonlarının sınır değerlerinin belirlenmesi</i>	49
5.7 KURAL TABANI.....	50
BÖLÜM 6 DENEYLER VE SONUÇLAR	52
6.1 Giriş.....	52
6.2 GERÇEKLEŞTİRİLEN KONTROLÖRÜN TEST EDİLMESİ	52
6.3 SONUÇ VE ÖNERİLER	55
KAYNAKLAR	56
EKLER.....	57

İÇİNDEKİLER

TEŞEKKÜR	II
ÖZET	III
ABSTRACT	V
ŞEKİLLER TABLOSU	VI
İÇİNDEKİLER	VII
BÖLÜM 1 GİRİŞ VE AMAÇ	1
BÖLÜM 2 BULANIK MANTIK	4
2.1 GİRİŞ	4
2.2 KLASİK MANTIK	4
2.3 BULANIK MANTIK	5
2.4 KLASİK KÜME İLE BULANIK KÜMENİN KARŞILAŞTIRILMASI	6
2.5 KLASİK KÜMELER	6
2.5.1. Liste Yöntemi	6
2.5.2. Ortak Özellik (Genelleme) Yöntemi	6
2.5.3. VENN Şeması ile Gösterim	7
2.5.4. Karakteristik Fonksiyon ile Gösterim	7
2.6 KLASİK KÜMELERDE TEMEL KÜME İŞLEMLERİ	7
2.6.1. Birleşim İşlemi	7
2.6.2. Kesişim İşlemi	8
2.6.3. Bir Kümenin Tümleyeni	8
2.7 KLASİK KÜMELERDE DİĞER ÖZELLİKLER	8
2.7.1. Değişme Özelliği	8
2.7.2. Birleşme Özelliği	8
2.7.3. Dağılma Özelliği	8
2.8 BULANIK KÜMELER	8
2.8.1 Sıralı İkili ile Gösterim Yöntemi	9
2.8.2 Toplam Formunda Gösterim Yöntemi	9
2.9 ÜYELİK FONKSİYON ŞEKİLLERİ	9
2.10.1. S Tipinde Üyelik Fonksiyonu	10
2.10.2. Çan Eğrisi Tipinde Üyelik Fonksiyonu	10
2.10.3. Üçgen Şeklinde Üyelik Fonksiyonu	11
2.10 BULANIK KÜMELERE İLİŞKİN TEMEL KAVRAMLAR	11
2.10.1 Sözel Niceleyiciler	11
2.11 BULANIK KÜMELERDE TEMEL KÜME İŞLEMLERİ	12
2.11.1 Bulanık Kümelerin Kesişimi	12
2.11.2 İki Bulanık Kümenin Birleşimi	13
2.11.3 Bir Bulanık Kümenin Tümleyeni	14
2.11.4 Bulanık Kümelerde Çarpma ve Toplama İşlemi	14
2.11.5 Bulanık Küme ile Klasik Küme Arasındaki Farklar	14

BÖLÜM 3 BULANIK KONTROLÖR	16
3.1 Giriş.....	16
3.2 BULANIK KONTROLÖR.....	16
3.3 BULANIK KONTROLÖRÜN YAPISI.....	17
3.3.1 <i>Bulanıklaştırma Birimi</i>	18
3.3.2 <i>Bilgi Tabanı</i>	19
3.3.3 <i>Çıkarım Ünitesi</i>	19
3.3.4 <i>Berraklaştırma Birimi</i>	21
BÖLÜM 4 ELEKTRİK MAKİNELERİ EĞİTİM SETİ (FH2 MKIV)	23
4.1 Giriş.....	23
4.2 SİSTEMİN GENEL YAPISI	23
4.3 A ALANI	24
4.4 B ALANI – MAKİNE BAĞLANTI SOKETLERİ	25
4.5 E ALANI – D.C. GÜÇ KAYNAĞI.....	26
4.6 F ALANI – D.C. MOTOR KONTROLÜ.....	27
4.7 FH50 DC MOTORUN DAVRANIŞININ İNCELENMESİ.....	29
4.7.1 <i>Yapılan Deneyler</i>	29
BÖLÜM 5 BULANIK KONTROLÖR TASARIMI.....	33
5.1 Giriş.....	33
5.2 TASARLANAN KONTROLÖRÜN GENEL YAPISI.....	33
5.3 MİKRO DENETLEYİCİ KART	34
5.3.1 <i>RS-232 Dönüştürücü Birimi</i>	34
5.3.2 <i>Mikro Denetleyici Katı</i>	37
5.3.3 <i>ADC (Analog Dijital Çevirici) Katı</i>	38
5.3.4 <i>DAC (Dijital Analog Çevirici) Katı</i>	39
5.3.5 <i>Motor Sürücü</i>	39
5.4 BULANIKLAŞTIRMA İŞLEMLERİ	40
5.4.1 <i>Üyelik Fonksiyonlarının Seçimi</i>	40
5.4.2.1 <i>Hızın Bulanıklaştırılması</i>	41
5.4.2.2 <i>Hız değişiminin Bulanıklaştırılması</i>	42
5.4.2 <i>Kural Tabanı</i>	43
5.4.2.1 <i>Bulanık Kontrol Kuralları</i>	43
5.5 KONTROL İŞARETİNİN BULUNMASI.....	44
5.5.1 <i>Kontrol İşaretinin Berraklaştırılması</i>	45
5.6 BİLGİSAYAR YAZILIMI	45
5.6.1 <i>Referans bilgi girişi</i>	47
5.6.2 <i>Üyelik Fonksiyonları Penceresi</i>	48
5.6.2.1 <i>Üyelik fonksiyonlarının sınır değerlerinin belirlenmesi</i>	49
5.7 KURAL TABANI.....	50
BÖLÜM 6 DENEYLER VE SONUÇLAR	52
6.1 Giriş.....	52
6.2 GERÇEKLEŞTİRİLEN KONTROLÖRÜN TEST EDİLMESİ.....	52
6.3 SONUÇ VE ÖNERİLER	55
KAYNAKLAR	56
EKLER.....	57

EK-1 PC PROGRAM LİSTELERİ	57
EK-2 MİKRODENETLEYİCİYE AİT “C” DİLİ KODU	101
EK-3 MİKRO DENETLEYİCİ KARTA AİT ŞEMA VE YERLEŞİM PLANI.....	115
EK-4 MOTOR SÜRÜCÜYE AİT ŞEMA	118
ÖZGEÇMİŞ.....	119



BÖLÜM 1

GİRİŞ VE AMAÇ

Hemen hemen bütün kaynaklar Bulanık Mantığın kurucusu olarak Amerika Berkeley, Kaliforniya üniversitesi profesörlerinden Azerbaycanlı bilim adamı Lütfi.A Zadeh'i gösterirler. L.A. Zadeh 1965 yılında bulanık mantık küme teorisi hakkındaki ilk bağıntıları "Information and Control" dergisinde yayınladığı makalesinde açıkladı[1]. 1966 yılında Bulanık Mantık Bell Laboratuvarları Dr Peter MARINOS tarafından kuruldu. Araştırmacılar teorisinin uygulamasından çok matematiksel yönleriyle ilgilendiler. Bu durum bulanık küme teorisine karşı bir boşluğun doğmasına sebep oldu.

1970 li yıllar bulanık mantığın gerçek sistemlere uygulanışının doğuş yılları oldu. 1974 yılında Mamdani karmaşık bir sistemin kontrol edilmesinde bulanık mantığın kullanılabileceğini ispatladı. 1975 yılında Mamdani Assilian ile birlikte ilk endüstriyel uygulamayı bir buhar makinası için gerçekleştirdiler[2]. Bu çalışma bulanık mantık denetleyicili uzman bir sistemle türbin hızının ve veriminin başarılı bir şekilde kontrol edilebileceğini gösterdi. 1978 yılının sonlarında Danimarkada Holmblad ve Ostergaard tamamen bulanık mantık ile kontrol edilen endüstriyel uygulamayı geliştirerek bir çimento üretimini başardılar. Bu uygulamada, hassas bir denge ile oranlanması gereken sıcaklık ve oksijen ayarı en uygun şekilde yapılmıştır. Bu çalışmalar araştırmacıları teorisinin uygulamasıyla ilgilenmeye teşvik etti.

Bulanık mantık ile ilgili çalışmalar ABD li mühendisler arasında gereken ilgiyi görmedi. Bu yeni teknoloji Japon mühendislerin hemen ilgisini çekti. Bulanık mantık sistemin matematik modeline ihtiyaç duymuyordu. Matematiksel ifade zorluğundan dolayı geleneksel kontrol yöntemlerinin uygulanamadığı sistemlerde bulanık mantık kolayca uygulanabilirdi. 1980 yılında, Sugeno ilk Japon uygulamasını su arıtma sistemini bulanık mantık denetleyici ile başardı. 1983'te bulanık mantık denetleyicili robot çalışmaları yaparak bu alana öncülük etti. 1985 yılında Nishida ile birlikte sesle kumanda edilerek kendikendine park edebilen bir araba geliştirdiler. 1987 yılında geliştirilen bulanık mantık temelli bir uzman sistem, 1987 yılının Ekim ayındaki "kara Pazartesi" olarak bilinen büyük ekonomik krizi 18 gün öncesinden haber vermiştir.

1987 yılında Japonyada Sandai metrosunun bulanık mantık ile kontrol edilmesi dünyada büyük yankı yaptı[2]. Eski sisteme göre yolculara daha rahat, daha

güvenli bir yolculuk sağlıyor ve zaman kaybı azalmıştı. Ayrıca, trenin istenilen yerde durması üç kat daha iyileşmiş ve enerji tüketimi %10 azalmıştır.

1987 Temmuz ayında Sendai metrosunun açılışından üç gün sonra ikinci uluslararası bulanık mantık sistemleri konferansı Tokyo da düzenlendi. Bu konferansta bulanık mantık denetleyici ile kontrol edilerek pin pon oynayan bir robot gösterisi yapıldı. Ayrıca ters sarkaçın bulanık mantık ile dengede tutulması gösterildi. Bundan önce Japonya da bulanık mantık pek bilinmiyordu. Konferanstan sonra mühendisler arasında, devlet kademelerinde ve iş kuruluşlarında bulanık mantık rüzgarı esti. Aralarında IBM, NCR, Hitachi, Omron SGS-Thomson gibi dünya devlerinin bulunduğu 51 firma tarafından LIFE (Laboratory for International Fuzzy Engineering) laboratuvarları kuruldu. 1990 yılının başlarında pazarı bulanık mantık ile kontrol edilen eşyalar kaplamaya başladı.

Bulanık mantığın Japonya daki bu başarısı ABD ve Avrupada mühendisler arasında büyük bir dalgalanmaya neden oldu. 1992 Şubat ayında ilk IEEE bulanık sistemler konferansı San Diego da yapıldı. Bu konferans bulanık mantığın en büyük mühendislik organizasyonu IEEE tarafından kabul edildiğini sembolize ediyordu.

Bulanık mantık insanın düşünme yeteneğini model alarak, beynin karmaşık yapıları çözme biçiminden yararlanır. Sistemin giriş ve çıkış değerleri vardır. Önce bu değerler için bir uzmandan yada deney sonucundan elde edilen verilerden üyelik fonksiyonları belirlenir. Üyelik fonksiyonları 0 ile 1 arasında değer alabilir. Bu fonksiyonlar EĞER – İSE şeklinde sözel ifadelere çevrilerek kurallar belirlenir. Bu kurallardan yararlanılarak çıkış değerleri için karar verme tablosu oluşturulur. Karar verme sonucunda kesin değerler alınarak çıkışta kontrol sinyali elde edilir.

Bulanık denetleyici temel olarak dört kısımdan oluşur. Bunlar bulanıklaştırma (fuzzification), bilgi tabanı (knowledge base) çıkarım ünitesi (inference engine) ve berraklaştırma (defuzzification) birimleridir. Önce sensörlerden alınan bilgiler bulanıklaştırma biriminde 0 ile 1 arasında üyelik fonksiyonuna ve sözel ifadelere çevrilir. Bilgi tabanındaki verilere göre çıkarım ünitesine gönderilir. Burada bilgi tabanında belirtilen kurallar işletilir. Berraklaştırma kısmında kesin değerler şeklinde çıkış sinyali oluşturulur.

Çalışmamızda Bulanık mantık denetleyici incelenmiş ve elektrik makineleri eğitim setinde (FH2 MkIV) bulunan DC motor kontrolünde Bulanık Mantık uygulaması gerçekleştirilmiş olup, şu konulara yer verilmiştir.

Birinci bölümde; Bulanık mantık hakkında genel bir giriş yapılmış geçmiş çalışmalardan bahsedilmiştir. İkinci bölümde, bulanık mantık anlatılmıştır.

Üçüncü bölümde; Bulanık kontrolör hakkında bilgi verilmiştir.

Dördüncü bölümde; Bulanık mantık uygulanmadan önceki sistem hakkında bilgi verilmiştir.

Beşinci bölümde; Bulanık mantık ile yapılan tasarım anlatılmış burada kullanılan mikro denetleyici kartı incelenmiştir. Ayrıca uygulamayı görsel (visual) hale getiren bilgisayar programı hakkında bilgi verilmiştir.

Son bölüm, tasarlanan kontrolör ile yapılan deney sonuçlarını ve önerileri içermektedir.

Ek 1'de; Bulanık kontrol algoritması yazılımı,

Ek 2'de; Mikrodenetleyici karta ait "C" dili kodu,

Ek 3'de; Mikrodenetleyici karta ait devre şemaları ve yerleşim planları,

Ek 4'de; Motor sürücüyeye ait devre şeması,

Verilmiştir.

BÖLÜM 2

BULANIK MANTIK

2.1 Giriş

Bulanık Mantık matematik eşitlikler yerine sözel terimlere dayanan bir sistemin işlevsel yasalarını ifade etmek üzere tasarlanmış bir yöntemdir[3]. Pek çok sistem kesin matematik modellemesi yapılamayacak kadar karmaşıktır. Fakat bulanık mantığın sözel ifadelerle dayalı olması böyle bir sistemin belirleyici özelliklerini tanımlayıcı elverişli bir yöntem sunar. Bu sözel ifadeler, çoğu kez mantıksal önermeler formunda ifade edilir. Örneğin;

Eğer Hava **Sıcaksa**, Fan hızını **orta** hıza ayarla

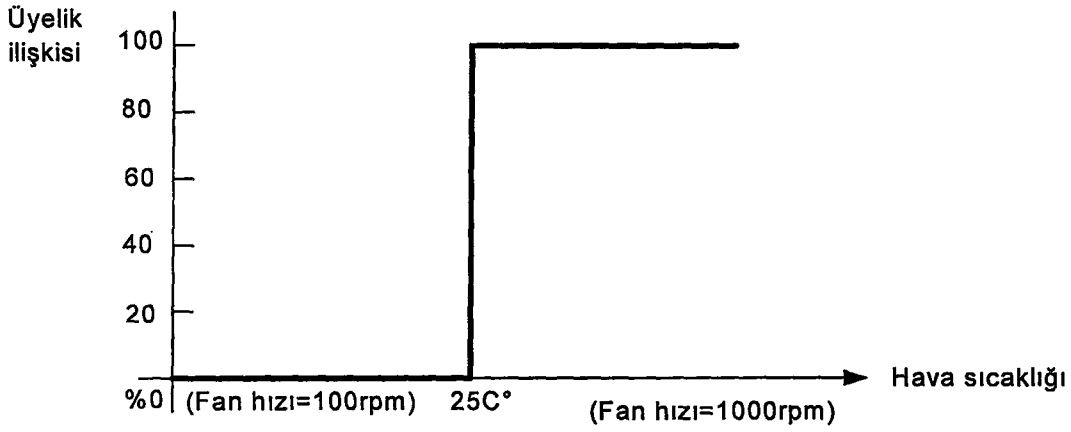
Sıcak ve orta tabirleri fiilen üyelik fonksiyonları olarak bilinen değerlerin kademelerini tanımlayan kümelerdir. Değişken hava sıcaklığı verisini tanımlamak üzere tek bir kesin değer vermek yerine bir değerler skalası seçmek suretiyle değişken fan hızı çıktısını daha kesin bir şekilde kontrol edebiliriz. Bulanık mantık denetleyicileri çoğu kez bir kontrol sisteminin performansını geliştirebilirler. Bunu, ölçülmüş giriş değişkenlerindeki değişikliklerin sebebiyet verebileceği çıkıştaki denetlenemeyen fonksiyonların şansını ortadan kaldırarak yapar. Bu bölümde Bulanık mantık genel olarak açıklanarak klasik mantıkla olan farklılıkları belirtilecektir.

2.2 Klasik Mantık

Klasik mantıkta, bir düşünce kesinlik belirtir. "Bu gün hava sıcak" ifadesinde havanın kesin bir şekilde sıcak olduğunu ifade eder. Sıcaklığın ara değerleri yoktur. Bir olayın sonucu kesin bir ifade ile belirtilir. Buna göre olay; ya doğrudur yada yanlıştır, ya vardır yada yoktur . Sonuç bu iki değerden başka bir değer alamaz. Klasik mantık tabanlı kümelerde bir nesne yada değişken kümenin ya tam üyesidir "1(var)" yada üyesi değildir "0 (yok)". Şekil 2-1 'de klasik mantık kümesine örnek olarak fan hızı kontrolüne ilişkin bir grafik görülmektedir. Burada,

Eğer hava sıcaklığı $\geq 25C^\circ$ ise, Fan hızını "1000" e

Eğer hava sıcaklığı $< 25C^\circ$ ise, Fan hızını "100" e ayarlamaktadır.



Şekil 2-1 Klasik mantık ile fan hızının kontrolü

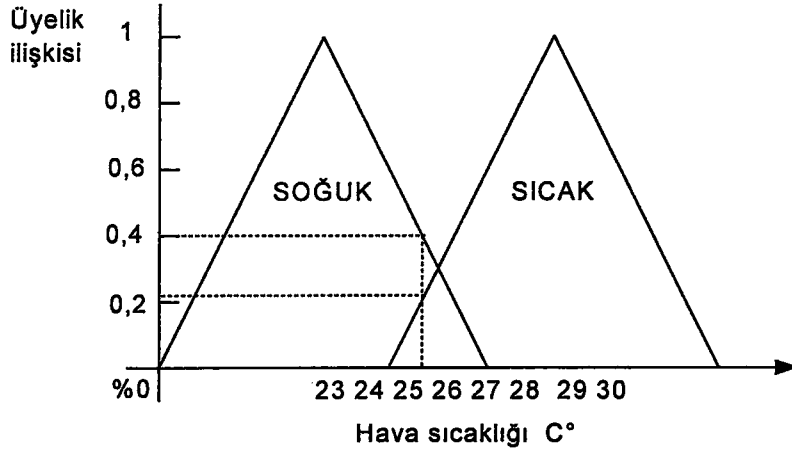
Şekil 2.1'de görüleceği gibi klasik mantık çok keskin bir yapıya sahiptir.

Hava sıcaklığı 25 C° nin altına veya üstüne çıktığında fan hızındaki değişim 10 kat olacaktır. $24,9\text{ C}^\circ$ ile $25,1\text{ C}^\circ$ arasındaki sıcaklık değişimi gerçek hayatımızda çoğu kere önemli değildir. Halbuki yaşantımızda keskin değerler yerine çok sıcak, az sıcak, soğuk gibi sözel ifadeler kullanırız.

2.3 Bulanık Mantık

Bulanık mantık klasik (keskin) mantığın aksine, isminden de anlaşılacağı gibi bir belirsizliği, bulanıklığı ifade eder. Gerçek hayatta kullandığımız düşünce mantığını model alır[4]. Klasik mantıkta rakamlarla belirtilen kesin değerlerin yerini bulanık mantıkta sıcak, soğuk gibi gerçek hayatta kullanılan sözel ifadeler almıştır.

Bulanık kümelerde bir ifade 0 ile 1 arasında herhangi bir üyelik değerine $\mu(x)$ sahiptir. Şekil 2-2'de hava sıcaklığı sıcak ve soğuk olarak iki adet bulanık küme ile temsil edilmiştir.



Şekil 2-2 Bulanık mantık ile fan hızının kontrol bulanıklaştırılması

Burada 25C° soğuk bulanık kümesinin 0,4 oranında üyesi iken, sıcak bulanık kümesinin 0,21 oranında üyesidir.

2.4 Klasik Küme ile Bulanık Kümenin Karşılaştırılması

Bulanık kümeler, klasik kümelerin geliştirilmiş türleridir. Birçok işlem klasik kümelerle aynı olmasa da benzerdir. Bundan dolayı klasik kümelerin incelenmesi bulanık kümelerin anlaşılmasını kolaylaştıracaktır.

2.5 Klasik Kümeler

Bir küme, küme elemanı olarak adlandırılan birbirinden farklı nesnelere topluluğudur. Kümeyi oluşturan nesnelere her birine o kümenin elemanı denir ve \in sembolü ile gösterilir. $A=\{x,y,z,(t)\}$ ise $x\in A$, $y\in A$, $z\in A$, $(t)\in A$ olarak gösterilir.

Klasik kümelerin çeşitli gösteriliş yöntemleri vardır. Bunlar aşağıda incelenecektir.

2.5.1. Liste Yöntemi

$A=\{x,y,z,(t)\}$ şeklinde gösterim biçimidir. Genelde sınırlı (sonlu elemana sahip) sayıda elemana sahip olan kümeleri belirtmek için kullanılır.

2.5.2. Ortak Özellik (Genelleme) Yöntemi

Bir kümenin elemanlarını ortak özellikleri ile tanımlayabiliyorsak ortak özellik yöntemi ile gösterebiliriz.

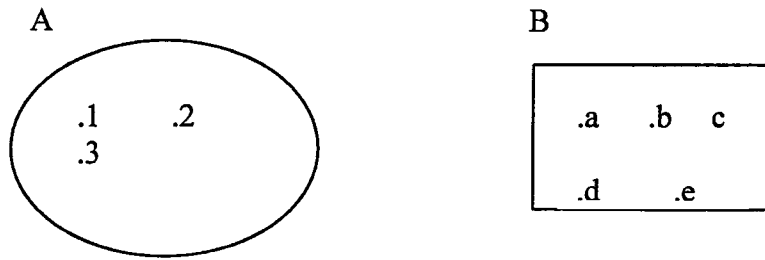
$A = \{20\text{'den küçük asal sayılar}\}$

$B = \{t \mid t \in \mathbb{N} \text{ ve } t < 5\}$

2.5.3. VENN Şeması ile Gösterim

Bir kümenin elemanlarını, kapalı bir eğri içinde yazarak gösterme biçimidir.

Şekil 2-3'te venn şemasına ilişkin gösterim görülmektedir.



Şekil 2-3 Venn şeması ile gösterim

2.5.4. Karakteristik Fonksiyon ile Gösterim

Uzayda E; evrensel kümeyi gösterebilir. E evrensel kümesi içinde A gibi bir kümeye ait karakteristik fonksiyon E'nin herhangi bir elemanı A kümesine ait ise karakteristik fonksiyon "1" değerini alır, ait değilse "0" değerini alır.

$$\mu_A(x) = \begin{cases} 1 & \text{eğer } x \in A \\ 0 & \text{eğer } x \notin A \end{cases}$$

Karakteristik fonksiyon klasik kümelerde fazla kullanılmayan bir yöntemdir. Bulanık küme teorisinde üyelik fonksiyonuna karşılık gelir ve sıklıkla kullanılır.

2.6 Klasik Kümelerde Temel Küme İşlemleri

Bütün kümeleri kapsayan kümelere evrensel küme denir. Her küme evrensel kümenin bir alt kümesidir.

2.6.1. Birleşim İşlemi

İki veya daha fazla kümenin elemanlarından oluşan kümeye birleşim kümesi adı verilir. " \cup " ile gösterilir.

$$A = \{1, 2, 3, 4\}$$

$$B = \{2, 4, 6, 8\}$$

$$A \cup B = \{1, 2, 3, 4, 6, 8\}$$

2.6.2. Kesişim İşlemi

İki veya daha fazla kümenin ortak elemanlarından oluşan kümeye, bu kümelerin kesişim kümesi adı verilir ve " \cap " ile gösterilir.

$$A=\{1,2,3,4\}$$

$$B=\{2,4,6,8\}$$

$$A\cap B=\{2,4\}$$

2.6.3. Bir Kümenin Tümlenyeni

E evrensel küme, $A\subseteq E$ olmak üzere A kümesinin tümlenyeni \bar{A} ile gösterilir ve

$$A\cup \bar{A} =E \text{ olarak tanımlanır.}$$

2.7 Klasik Kümelerde Diğer Özellikler

2.7.1. Değişme Özelliği

$$A\cup B= B\cup A$$

$$A\cap B= B\cap A$$

2.7.2. Birleşme Özelliği

$$A\cup(B\cap C)= (A\cup B)\cap C$$

$$A\cap(B\cup C)= (A\cap B)\cup C$$

2.7.3. Dağılma Özelliği

$$A\cup(B\cap C)= (A\cup B)\cap (A\cup C)$$

$$A\cap(B\cup C)= (A\cap B)\cup (A\cap C)$$

2.8 Bulanık Kümeler

Klasik kümeler karakteristik fonksiyonlarla ifade edilmelerine karşılık bulanık kümelerde üyelik fonksiyonu ile tanımlama sözkonusudur [5]. Üyelik fonksiyonları ise verilen kümelerin elemanları için çeşitli üyelik derecelerini içeren fonksiyonlardan ibarettir. Bulanık kümelerin sıralı ikili ve toplam formunda gösterim yöntemi vardır[6].

2.8.1 Sıralı İkili ile Gösterim Yöntemi

Sıralı ikili gösterim yönteminde E evrensel küme olmak üzere, E üzerinde tanımlanan x bulanık kümesi sıralı ikililerden oluşan bir küme olarak ifade edilebilir.

$$X=\{x_1, x_2, x_3, \dots, x_n\} \quad (2-1)$$

$$E=\{(x | \mu_{X(x)}) | x \in X\} \quad (2-2)$$

Bu gösterimde birinci kısım elemanı, ikinci kısım ise elemanın üyelik derecesini ifade eder.

$$X=\{1,2,3,4,5,6,7\}$$

Sonlu bir domen olmak üzere

Z bulanık kümesi ise

$$Z=\{(1,0.1), (2,0.3), (3,0.9), (4,1.0), (5,0.8), (6,0.6), (7,0.3)\} \quad (2-3)$$

Şeklinde ifade edilir.

X	1	2	3	4	5	6	7
$\mu_{Z(x)}$	0.1	0.3	0.9	1.0	0.8	0.6	0.3

2.8.2 Toplam Formunda Gösterim Yöntemi

X bir evrensel küme ve bu küme üzerinde tanımlanan A bir bulanık küme olmak üzere;

X'in sonlu bir küme olması durumunda üyelik fonksiyonu Dk. 2-4 deki gibi ifade edilir.

$$A = \frac{\mu_A(X_1)}{X_1} + \frac{\mu_A(X_2)}{X_2} + \dots + \frac{\mu_A(X_i)}{X_i} = \sum_{i=1}^n \frac{\mu_A(X_i)}{X_i} \quad (2-4)$$

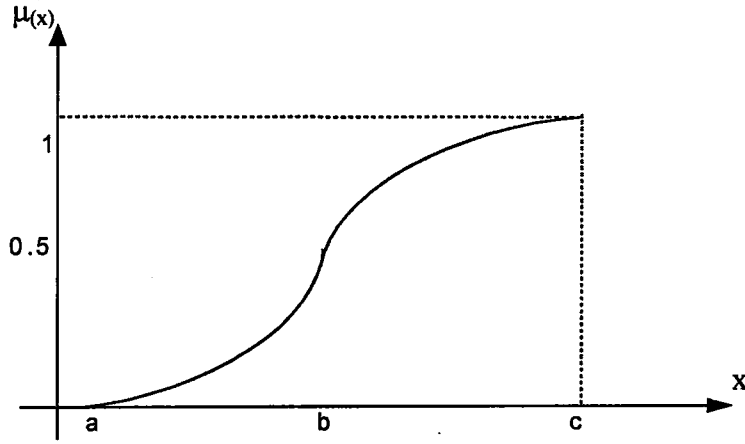
X'in sürekli olması durumunda üyelik fonksiyonu aşağıdaki gibi ifade edilir.

$$A = \int_x \frac{\mu_A(X_i)}{X_i} \quad (2-5)$$

2.9 Üyelik Fonksiyon Şekilleri

Üyelik fonksiyonları için kullanılan şekiller ele alınan durumlara göre farklılık gösterir[7]. Üyelik fonksiyonlarının şekillerine ait hiçbir kısıtlama yoktur, değişik şekillerde seçilebilir. Tamamıyla tasarımcının istek ve tecrübesine bağlıdır. Aşağıda en çok kullanılan değişik tipte üyelik fonksiyonları verilmektedir.

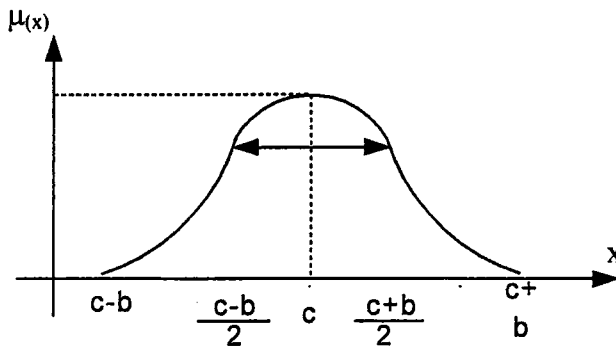
2.9.1. S Tipinde Üyelik Fonksiyonu



Şekil 2-4 "S" tipi üyelik fonksiyonu

$$S(X;A,B,C)=\begin{cases} 0 & x < a \\ 2 * \left(\frac{x-a}{c-a} \right)^2 & a \leq x \leq b \\ 1 - 2 * \left(\frac{x-a}{c-a} \right)^2 & b \leq x \leq c \\ 1 & x > c \end{cases} \quad (2-6)$$

2.9.2. Çan Eğrisi Tipinde Üyelik Fonksiyonu

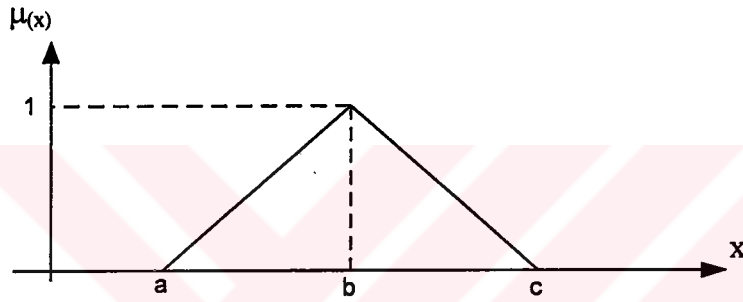


Şekil 2-5 Çan eğrisi tipi üyelik fonksiyonu

$$\Pi(x;a,b,c)=\begin{cases} S(x,c-a,\frac{c-b}{a},c) & x>c \\ S(x,c-a,\frac{c+b}{a},c) & x<c \end{cases} \quad (2-7)$$

bu fonksiyon iki adet "s" fonksiyonu gibi düşünülebilir

2.9.3. Üçgen Şeklinde Üyelik Fonksiyonu



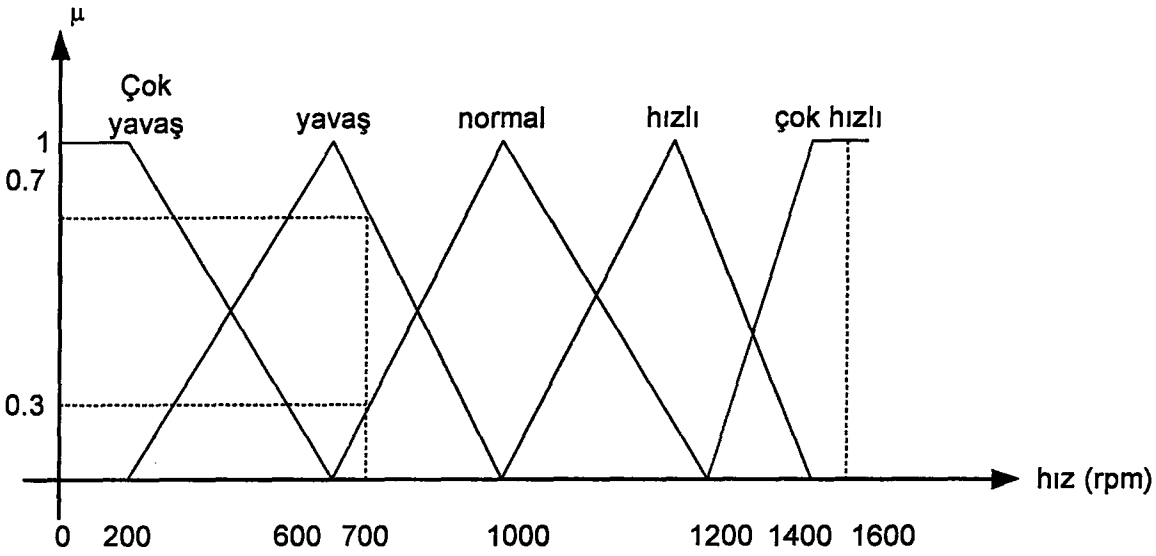
Şekil 2-6 Üçgen fonksiyonu

$$U(x;a,b,c)=\begin{cases} 0 & x<a \\ \frac{x-a}{b-a} & a\leq x\leq b \\ \frac{c-x}{c-b} & b\leq x\leq c \\ 0 & x>c \end{cases} \quad (2-8)$$

2.10 Bulanık Kümelere İlişkin Temel Kavramlar

2.10.1 Sözel Niceleyiciler

Bulanık kümelere ait en önemli dilsel niceleyiciler "sözel değişken" ve "sözel değerler" kavramlarıdır. Şekil 2-7'de bir fan motoruna ait devir hızı bulanıklaştırılması gösterilmiştir.



Şekil 2-7 Bir fan motoruna ait devir hızının sözel ifadesi

Burada "hız" Yatay ekseninde yer alan (0-1600 rpm) değerlerini alabilen sözel değişkendir. (0..1) arasında değer alabilen üyelik fonksiyonlarının aldığı "çok yavaş,.....,çok hızlı" gibi ifadeler sözel değerler olarak adlandırılırlar. Her bir sözel değer üyelik fonksiyonu ile temsil edilir. Örneğin 700 rpm hız değerini ele alacak olursak, hem "yavaş" hem de "normal" üyelik fonksiyonunu kesmektedir ve 0.7'lik bir derece ile "yavaş", 0.3'lük bir derece ile de "normal" üyelik fonksiyonunun elemanı olmaktadır.

2.11 Bulanık Kümelerde Temel Küme İşlemleri

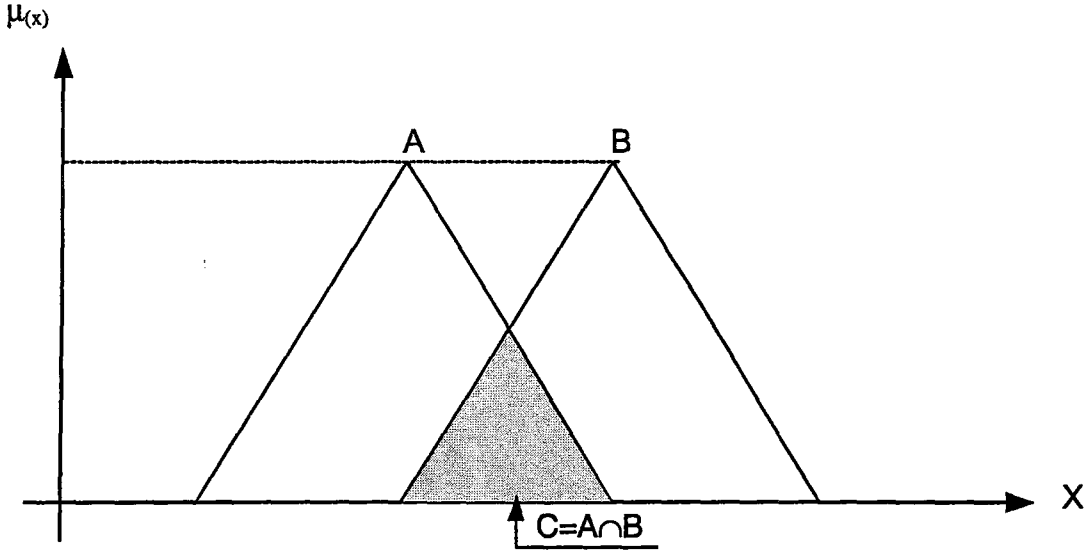
Klasik kümelerde bahsedilen bazı küme işlemlerinin benzerlerinden bulanık kümelerde de bahsetmek mümkündür. Hiç şüphesiz bulanık kümenin temelini üyelik fonksiyonu özelliği oluşturmaktadır. Diğer bütün özellikler ve işlemler üyelik fonksiyonu üzerinden gerçekleştirilir. Aşağıda bulanık kümelere ilişkin özellikler verilmiştir.

2.11.1 Bulanık Kümelerin Kesişimi

A,B,C bulanık kümeler olmak üzere X ve Y bulanık kümelerinin kesişimi, $C=A \cap B$, Dk.2-9 daki gibi ifade edilir.

$$\mu_C(x) = \min[\mu_A(x), \mu_B(x)] \rightarrow x \in X \quad (2-9)$$

Şekil 2-8'de kesişim işlemi görülmektedir.

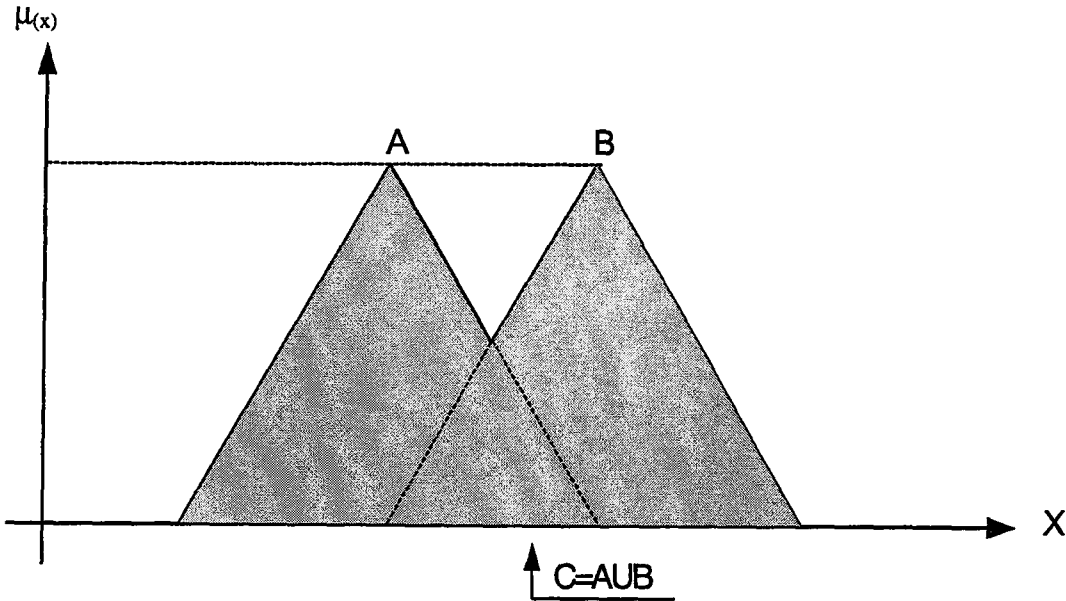


Şekil 2-8 İki bulanık kümenin kesişimi

2.11.2 İki Bulanık Kümenin Birleşimi

A,B,C bulanık kümeler olmak üzere A ve B bulanık kümelerinin birleşimi, $C=A \cup B$, Dk. 2-10'da verildiği gibi ifade edilir. Şekil 2-9'da kesişim işlemi görülmektedir.

$$\mu_C(x) = \max[\mu_A(x), \mu_B(x)] \rightarrow x \in X \quad (2-10)$$

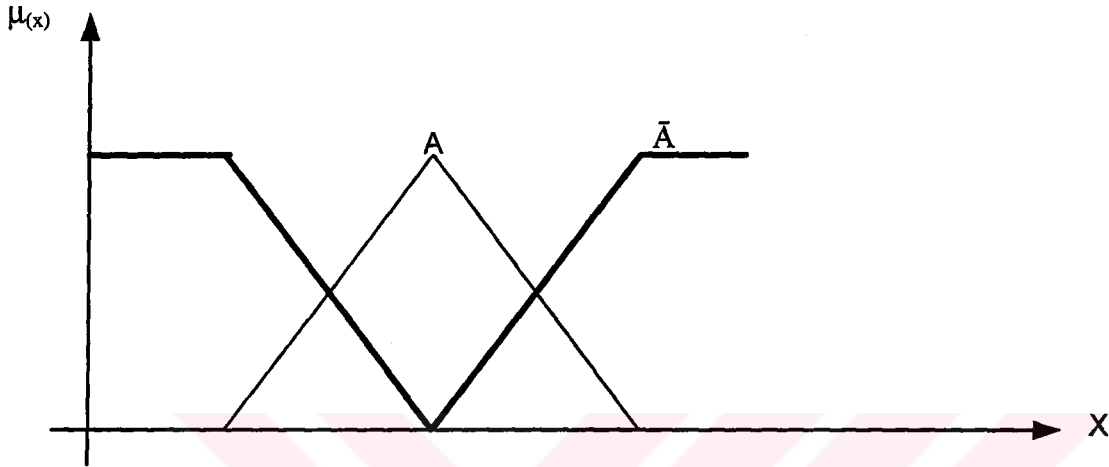


Şekil 2-9 İki bulanık kümenin birleşimi

2.11.3 Bir Bulanık Kümenin Tümlenyeni

A bulanık küme olmak üzere ; A kümesine ait elemanların 1'den çıkarılmasıyla elde edilen kümeye A bulanık kümesinin tümlenyeni denir ve \bar{A} ile gösterilir. Dk.211'de ifade şekli verilmiştir. Şekil 2-10'da tümleme işlemi görülmektedir.

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \rightarrow x \in X \quad (2-11)$$



Şekil 2-10 Bir bulanık kümenin tümlenyeni

2.11.4 Bulanık Kümelerde Çarpma ve Toplama İşlemi

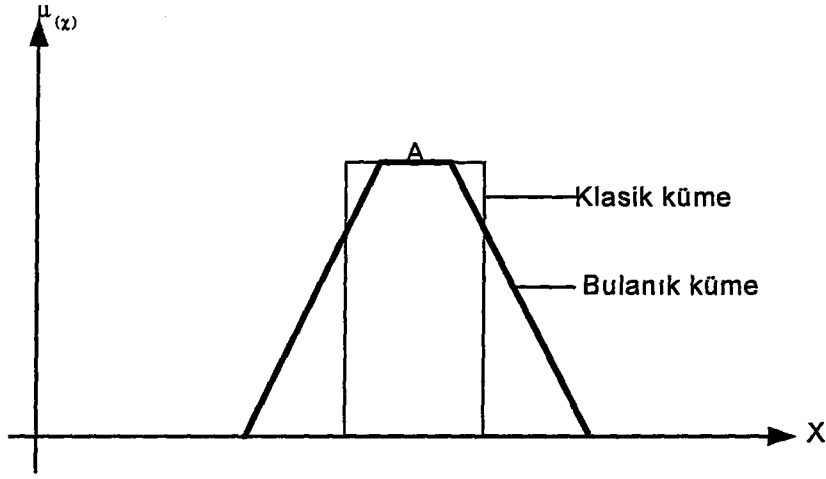
A ve B bulanık kümelerinin toplam ve çarpımı sırasıyla Dk. 2-12 ve Dk. 2-13 de ki gibi ifade edilir.

$$\mu_{A \cdot B}(x) = \mu_A(x) \cdot \mu_B(x) \quad (2-12)$$

$$\mu_{A+B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x) \quad (2-13)$$

2.11.5 Bulanık Küme ile Klasik Küme Arasındaki Farklar

Şekil 2-11'de bulanık küme ile klasik kümenin karşılaştırılması görülmektedir[8].



Şekil 2-11 Klasik küme ile bulanık kümenin karşılaştırılması

Bulanık küme teorisi, birleşme, değişme, dağılma ve De Morgan teoremi gibi özellikler ile klasik mantık kümesine benzemekle birlikte bulanık küme teorisinde verilen Dk. 2-14 ve Dk. 2-15'deki eşitsizlikler klasik küme teorisinde eşitlik olarak ifade edilir[8].

$$\mu_A(x) \cap \mu_{\bar{A}}(x) \neq \emptyset \quad (2-14)$$

$$\mu_A(x) \cup \mu_{\bar{A}}(x) \neq E \quad (2-15)$$

Burada; \emptyset ;boş kümeyi, E, evrensel kümeyi göstermektedir.

BÖLÜM 3

BULANIK KONTROLÖR

3.1 Giriş

Bütün kontrol sistemlerinde çalışma biçimlerini tanımlayan mantık, kontrol yöntemine uygun bir biçimde ifade edilir. Bu ifade biçimi klasik kontrol tekniklerinde kesin ve keskin (crisp) iken, bulanık kontrolde ifadeler bulanık olarak tanımlanır. Bulanık olarak tanımlama işlemi sözel ifadelerin kurallar dizisine dönüştürülmesiyle elde edilir.

Bu bölümde bulanık kontrolörün çalışması ve yapısını oluşturan birimler ayrıntılı olarak anlatılacaktır.

3.2 Bulanık Kontrolör

Mükemmel bir kontrol sisteminin tasarımı kontrol edilecek sistemi tanımlayan matematiksel modelin uygunluğu ile direk olarak ilişkilidir. Bazı sistemlerin matematiksel modellerinin elde edilmesi çok zordur. Aynı zamanda elde edilen matematiksel modeller çok karmaşık veya non-lineer olabilir. Dolayısıyla matematiksel modeller ile kontrol sisteminin gerçekleştirilmesi mümkün olmayabilir. Bu gibi durumlarda klasik kontrol sistemlerinin kullanımı mümkün olmaz. Böyle bir sistemin kontrolü gerektiğinde, eğer mümkünse uzun süre bu işle uğraşmış bir insanın (operatör) kullanılması uygun bir çözümdür. İnsanın işlerini kolaylaştıracak bir sistemde insan duyu organlarını kullanarak, içgüdülerine güvenerek gerekli kontrol işaretinin üretilmesini bir denetleyici maharetiyle gerçekleştirebilir. Dolayısıyla aklımıza şu soru gelmektedir. "Acaba insan gibi veya doğadaki olaylar gibi hareket eden bir denetleyici yapılamaz mı?" işte bu sorudan yola çıkarak , insanın karar verme şekli ile paralel olarak karar verebilen bulanık mantık doğmuştur[2].

Ancak her yöntemde olduğu gibi, bulanık mantık temelli kontrolün kullanılmasına karar vermeden önce sistemi iyice incelemek ve neden bulanık kontrolün kullanılacağına karar vermek gerekir. İkinci aşama olarak sistem iyice tanınmalı ne tür bir denetleyici kullanılacağına karar vermek gerekmektedir. Ancak bu aşamalardan sonra eğer daha iyi bir yöntem bulunamıyor ise bulanık kontrolün uygulamasına geçilmelidir. Diğer uygulamalarda olduğu gibi yanlış uygulama alanının

seçilmesi ya da yukarıdaki işlevlerin yerine getirilmemesi kontrol sistemini sadeleştireceği yerde daha karmaşık hale de getirebilir.

Bütün kararlar verildikten sonra kontrol sisteminin tasarımı ve kontrol algoritmasının belirlenmesine gelinmiştir. Bu aşamada da önemli kararlar ve seçilmesi gereken birtakım parametreler vardır. Bunlar arasında kontrol parametrelerinin ve sistemden geri besleme yolu ile alınacak durum değişkenlerinin belirlenmesi, bulanık kümelerin uygun seçilmesi ve üyelik fonksiyonlarının tanımlanması ve gerekli kontrol işlemlerinin yürütülmesi için kontrol kurallarının tanımlanması en önemli işlevlerdir. Ayrıca karar verme mekanizmasının belirtilmesi de gerekmektedir. Görüleceği üzere bir bulanık kontrol sisteminin tasarlanması ve gerekli kontrol algoritmasının gerçekleştirilmesi başlı başına bir zorluktur.

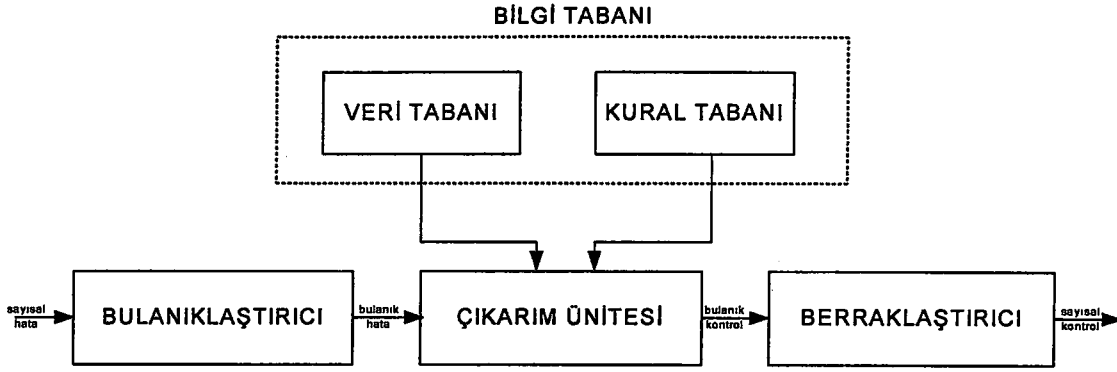
Kontrol algoritması belirlendikten sonra algoritmanın bilgisayara veya mikro denetleyici birimlerine program olarak aktarılması gerekmektedir. Bu aşama gerçek zamanda kontrol işlemlerinin yapılması için en önemli aşamalardan biridir. Programın yazılması esnasında kontrol edilecek sistemin gereksinimleri dikkate alınarak esnek ve efektif bir program yazılmalıdır.

Bulanık kontrol sistemleri, teorik ve pratik olarak en çok araştırma yapılan sahalardan biri olmuştur. Pratikte bulanık kontrol sistemlerini üstün kılan sebepleri şu şekilde ifade edebiliriz[8].

- Sistemin genel bir matematik modelini elde etmeye gerek yoktur.
- Ele alınan işlem çok karmaşık ve buna ait bilgiler sınırlı veya sistem lineer değilse bu durumda bulanık kontrol ön plana çıkar sistemin kontrolü bulanık mantıkla gerçekleştirilebilir.
- Bulanık denetleyiciler sistemin daha esnek çalışmasını temin eder farklı durumlarda minimum değişiklik yapılarak kontrol işleminin yapılmasını sağlar.
- Bulanık denetleyici kullanılarak elde edilen sonuçlar, en az klasik kontrol metotları ile elde edilen sonuçlar kadar iyidir.

3.3 Bulanık Kontrolörün Yapısı

Genel olarak bir bulanık kontrol sistemini Şekil 3-1'deki gibi gösterebiliriz.



Şekil 3-1 Bulanık kontrolörün yapısı[8]

Bulanık kontrolör genel olarak 5 ana kısımdan oluşur[8]:

1. Kontrol kurallarından oluşan “kural tabanı”
2. Kural tabanının oluşturulmasında kullanılan sözel terimlerin üyelik fonksiyonlarını tanımlayan bir “veri tabanı”
3. Bulanık kontrol kurallarını değerlendiren çıkarım ünitesi
4. Algılayıcılardan alınan sistem bilgilerini bulanıklaştıran “bulanıklaştırma birimi”
5. Bulanık bilgiyi sayısal değere çevirerek sisteme uygulayan “berraklaştırma birimi”

Bulanık kontrolörü oluşturan Şekil 3-1’deki birimler ayrıntılı olarak anlatılacaktır.

3.3.1 Bulanıklaştırma Birimi

Gerçek dünyaya ait işaretler bu birimde bulanıklaştırılarak sözel değişkenler şeklinde ifade edilmeleri sağlanır. Sayısal olarak alınan işaretleri bulanıklaştırmak için, tanımlanmış evrensel küme içerisinde üyelik fonksiyonları (membership functions) tanımlanır. Bu tanımlama “en az”, “az”, “sıfır”, “orta” gibi sözel ifadelerdir. Üyelik fonksiyonlarında üçgen, yamuk, trapez, çan eğrisi gibi grafik şekilleri kullanılır.

Fonksiyon tanımlamalarında, üyelik sayısı, biçimi tamamen tasarımcının tecrübesiyle ilgilidir. Bulanık kümelerle ilgili belirsiz bilginin temsili sayısal bilgisayar sisteminde, bilginin ölçülmesi ve değerlendirilmesi problemini ortaya çıkartır. Eğer bulanık küme sürekli formda temsil edilmiş ise ayrıştırılarak ayrık sabit mantık kümesi

elde edilir. Ayırıklaştırma işlemi, bulanık kümeyi belli sayıda seviyelere ayırarak (kuantalayarak) sağlanır.

Bulanıklaştırıcı biriminde aşağıdaki işlemler gerçekleştirilir.

1. Algılayıcılar vasıtasıyla dış dünyadan gelen işaret değerleri alınır.
2. Bu işaretlere göre sözel ifadelere çevrilip üyelik fonksiyonu hazırlanır.
3. Giriş değerleri üyelik fonksiyonlarına göre bulanıklaştırılır.

3.3.2 Bilgi Tabanı

Bulanık kontrol sistemlerindeki bilgi tabanı; veri ve kural tabanı olmak üzere iki kısımdan oluşur[8]. Bilgi tabanı, bulanık kontrol kuralları ve bulanık verileri tanımlamak için kullanılır. Bu veriler, tamamen uzmanların ve operatörlerin tecrübe ve yargılarına bağlıdır. Bulanık kümelerin oluşturulması sırasında üyelik fonksiyonlarının seçim ve uygulamasında önemli rol oynar. Veri Tabanı, bulanık mantık kontrolcusundaki bulanık verileri ve dilsel kontrol kurallarına bağlı bilgileri tanımlar. Kural tabanı ise, dilsel kontrol kuralları kümesindeki kontrol durumlarını ve kontrol amaçlarını belirler[9]. Yani sistemin giriş ve çıkışları arasındaki ilişkiyi açıklar. Bu uzman bilgi kümeleri genellikle "Eğer-İse" kurallarıyla oluşturulur.

3.3.3 Çıkarım Ünitesi

Sistemden okunan fiziksel değerler, minimum ve maksimum sınır arasında bir noktaya düşmektedir. Bu noktaya karşılık gelen bir dilsel ifade ismi bulunmaktadır ve bu dilsel ifade ismine karşılık gelen üyelik derecesi oluşmaktadır. Böylece, bulanıklaştırıcı ile keskin (crisp) fiziksel değerler 0-1 arasında bir üyelik derecesine sahip olarak bulanıklaştırılmaktadır. Ayrıca, fiziksel değerlerin minimum ve maksimum sınırlarının belirlenmesi, her üyelik fonksiyonun A1 gibi hangi fiziksel değere eş düştüğünün saptanmasına ölçeklendirme (scalling) denmektedir.

Farklı evrensel kümelerde tanımlanmış bulanık kümelerin birleştirilmesi ile (bağlanması ile) bulanık bağıntılar oluşur. Mesela;

IF $x=az$ AND $y=çok$ THEN <sonuc>

EĞER $x=az$ VE $y=çok$ İSE <sonuc>

Şeklinde bir IF...THEN... kuralı dikkate alınacak olursa şartlar kısmını içeren ($x=az$ VE $y=çok$) ifadenin birbirine bir bağlaçla bağlandığı görülür. Şartları içeren ifade çözümlenecek olursa; x için verilen "az" ifadesi x 'e ait kavramsal durumu ifade

ederken, y için verilen “çok” ifadesi y'ye ait kavramsal durumu ifade eder. Farklı evrensel kümeler birbirleriyle ilişkiye girdiğinden dolayı “AND (VE)” bağlantısı ile iki kademeli bir bulanık bağıntı oluşturulur. Bulanık bağıntının üyelik fonksiyonu “AND (VE)” işleminin modellenmesi için MIN operatörü kullanılarak Dk.3-1 deki gibi ifade edilir.

$$\mu_R(x) = \text{MIN}(\mu_{\text{az}}(x), \mu_{\text{çok}}(y)) \quad (3-1)$$

EĞER $x=A$ İSE $y=B$ 'dir

Şeklindeki tek şartlı ve tek sonuçlu bir kuralı incelersek;

Bulanık çıkarımda kullanılan yukarıdaki gibi bir kural genel manada “şart” ve “sonuç” kısımlarından meydana gelir. Burada şart kısmı $x=A$, x değişkeninin A kavramsal ifadesi ile karakterize edilir. Sonuç kısmı ise $y=B$ şeklinde y değişkeninin B kavramsal ifadesi ile karakterize edilir. Kural, “EĞER x değişkeni A kavramsal ifadesine eşit İSE B kavramsal ifadesine eşit olmalıdır.” Şeklinde değerlendirilir. Burada A ve B kavramsal ifadeleri bulanık ifadeler olduğundan şartlar ve sonuç ifadeleri keskin olmayan ifadelerdir.

Buna göre kural, bulanık çıkarım olarak ifade edilir ve $A \Rightarrow B$ ile gösterilir.

Klasik ifade mantığının tersine burada, şartlar kısmı “sadece doğru ya da yanlış” değerleri ile ifade edilmeyip “daha doğru veya daha az doğru ya da daha az yanlış” gibi duruma bağlı olarak değişen, çıkarım talimatına ihtiyaç vardır.

KURAL :EĞER $x=A$ İSE $y=B$ dir

OLAY : x daha az ya da daha fazla A'nın özelliğine sahiptir.

SONUÇ :x'in durumuna bağlı olarak y de sonuç olarak B'nin daha az ya da daha fazla özelliğine sahiptir.

Yukarıdaki kuralda “x” ve “y” gibi iki büyüklük ilişkiye girdiğinden dolayı, bu kural iki kademeli bir R bağıntısı ile bulanık küme matematiğinde ifade edilir. Bulanık bağıntı üyelik fonksiyonu $\mu_R(x,y)$, şart kısmına ait üyelik fonksiyonu $\mu_A(x)$ ve sonuç kısmına ait üyelik fonksiyonu $\mu_B(y)$ olarak dikkate alınır. Buradaki en önemli sorun üyelik fonksiyonlarının birleştirilmesinde hangi operatör tanımının dikkate alınacağıdır. Zira literatürde Tablo 3-1'de verilen pek çok operatör tanımı yapılmıştır.

Yukarıdaki incelemelerde “Mamdani çıkarım metodu” adı verilen metot kullanılmıştır. Bu çıkarım metodunda sonuçta elde edilen çıkarım kavramının doğruluk değeri şartlar için elde edilen doğruluk değerine eşit olması ya da küçük olması kavramı ile çıkarım işlemi gerçekleştirilmektedir. Mesela, şartların üyelik

derecesi ele alınan durum için 0.5 değerine karşılık geliyorsa, böyle bir durumda oluşması gereken sonuç çıkarım değerinin üyelik değeri en fazla 0.5 değerini göstermelidir. Buna göre kuralın $A \Rightarrow B$ üyelik fonksiyonu basitçe, her iki üyelik fonksiyonunun minimumunu veren “VE” bağlacının seçilmesi durumu ile aşağıdaki şekilde ifade edilir.

$$\mu_{R:A \Rightarrow B}(x,y) = \min(\mu_A(x), \mu_B(y)) \quad (3-2)$$

Bulanık çıkarım işlemlerinde kullanılan bu kural, EĞER...İSE...DİR formatının elde edildiği en çok kullanılan yöntemlerden biridir. “MIN” operatörü yerine üyelik fonksiyonlarının cebirsel çarpımını çıkarım metodu olarak ele alan “MAX-PROD” metodu da en çok kullanılan metotlardan biridir.

Aşağıda literatürde sıkça karşılaşılan bazı çıkarım metotları ifade edilmiştir.

Tablo 3-1 Çıkarım yöntemleri

BULANIK ÇIKARIM METODU	$\mu_{A \Rightarrow B}(x,y)$
MAMDANI (MAX-MIN)	$\min(\mu_A(x), \mu_B(y))$
MAX-PROD	$\mu_A(x) * \mu_B(y)$
ZADEH	$\max[\min(\mu_A(x), \mu_B(y)), 1 - \mu_A(x)]$
LUKASIEWICS	$\min(1, 1 - \mu_A(x) + \mu_B(y))$
GÖDEL	$1 \rightarrow \mu_A(x) < \mu_B(y)$ $\mu_B(y) \rightarrow \text{diğer}$
KLEENE-DIENES	$\max(1 - \mu_A(x), \mu_B(y))$

3.3.4 Berraklaştırma Birimi

Çıkarım sonucunda elde edilen bulanık kontrol işaretinin, sisteme uygulanabilmesi için kontrol işaretinin sayısal (crisp) değere dönüştürülmesi gerekir. Bu işleme “berraklaştırma” denir.

Literatürde en çok karşılaşılan Berraklaştırma metotları aşağıda açıklanmıştır.

Maksimum berraklaştırma metodunda, aktif olan kuralların en büyük üyelik derecesi kontrol işareti olarak alınır[10].

Ortalama değer metodunda ise aktif kuralların çıkışlarına ilişkin üyelik fonksiyonlarının maksimum değerdeki ağırlıklarının ortalaması kontrol işareti olarak alınır. Maksimum ortalama değer metodu Dk. 3-3 deki gibi ifade edilebilir.

$$U = \sum_{i=1}^k \frac{w_i}{k} \quad (3-3)$$

Bu ifadede U sayısal kontrol işaretini, w_i , $\mu(w_i)$ üyelik fonksiyonunun maksimum seviyeye ulaştığı değeri, k ise aktif kural sayısını göstermektedir.

Ağırlık merkezi yöntemi, en yaygın kullanılan yöntemdir. Burada, aktif kuralların bulanık çıkışlarına ilişkin üyelik fonksiyon değerleri ile skaler ağırlıkları, çarpılarak, toplamları alınır. Elde edilen değer, üyelik fonksiyon değerlerinin toplamına bölünmesiyle sayısal kontrol işareti bulunur. Bu yöntem aşağıdaki gibi ifade edilebilir.

$$U = \frac{\sum_{i=1}^n w_i \cdot \mu(w_i)}{\sum_{i=1}^n \mu(w_i)} \quad (3-4)$$

Burada n, kural sayısını, $\mu(w_i)$ üyelik fonksiyonunu temsil etmektedir.

BÖLÜM 4

ELEKTRİK MAKİNELERİ EĞİTİM SETİ (FH2 MKIV)

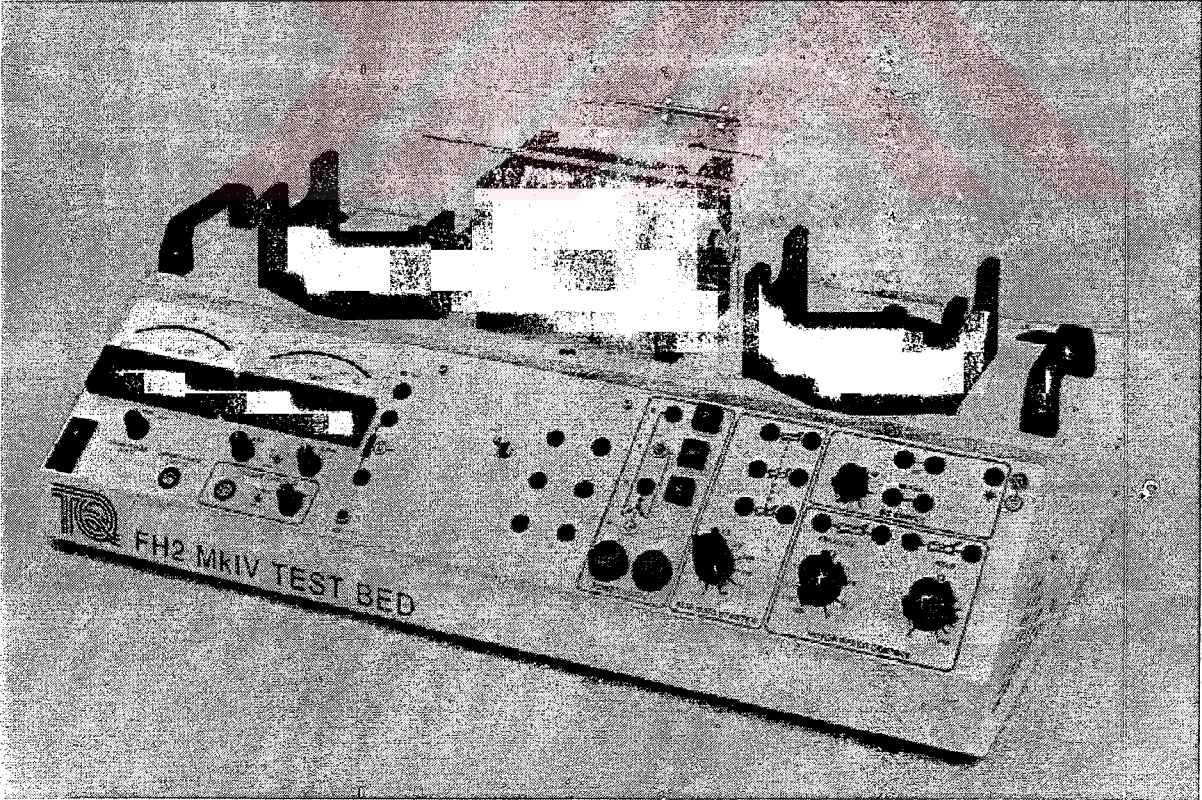
4.1 Giriş

Bu bölümde FH2 MkIV setinin yapısı ve çalışma prensibi anlatılmıştır. Sistemin davranışını incelemek amacıyla DC motor kontrolsüz olarak değişik hız değerlerinde çalıştırılmıştır.

4.2 Sistemin Genel Yapısı

Bu set kendi boyutlarına uygun motor ve jeneratörlerin çalışma karakteristiklerinin tesbit edilmesinde gerekli olan çelik bir şase üzerine monte edilmiş güç kaynağı, fren sistemi ve makine bağlama kızaklarından oluşmuştur.

Şekil 4-1 FH2 MkIV setinin kontrol panelini göstermektedir. Daha anlaşılır olması için panel fonksiyonel alanlara bölünmüştür[11].



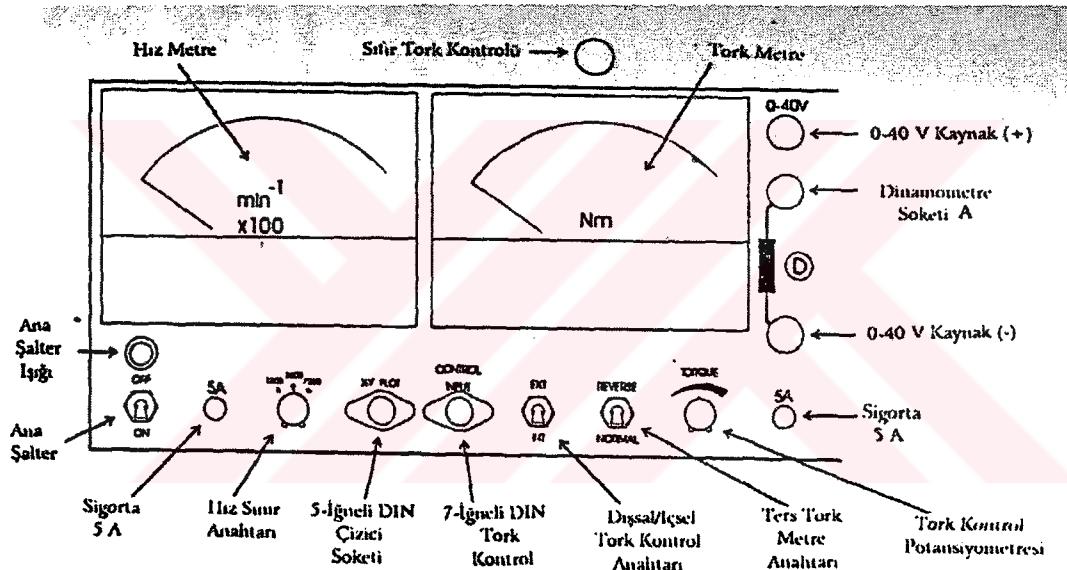
Şekil 4-1 FH2 MkIV Elektrik makineleri eğitim seti

4.3 A Alanı

Bu bölümdeki birimlerin en önemli fonksiyonları motorların dönüş hızlarını görüntülemek ve izlemek; aynı zamanda cihazın üst kısmına yerleştirilen manyetik dinamometre freni ile oluşan torku kontrol etmek, izlemek ve görüntülemektir.

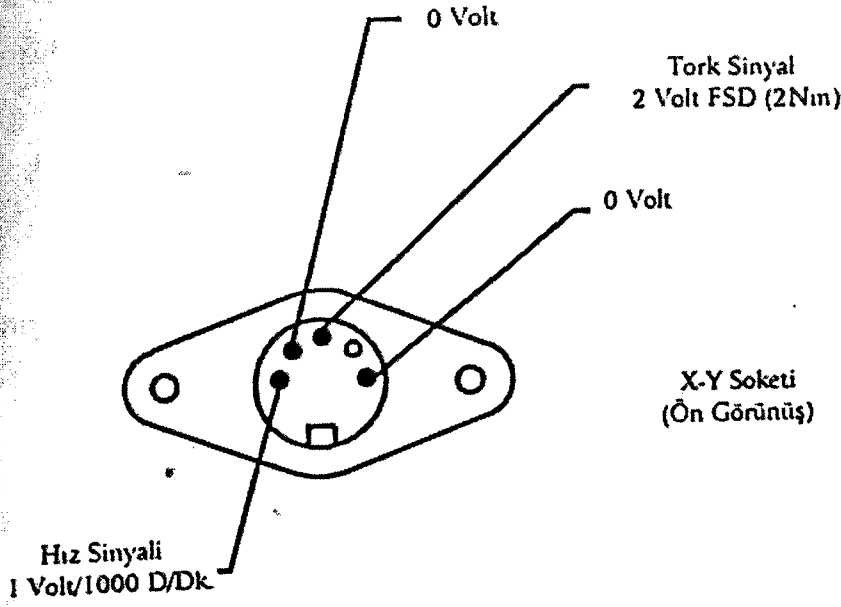
Dinamometre kutusunun içine motor dönüş hızına göre orantılı frekansta palsler üreten bir optik sensör yerleştirilmiştir. Elektronik devre bu frekansı uygun D.C. gerilimine dönüştürülüp , hız metre kadranında hız değeri cinsinden görüntüler.

Hız sınır anahtarı 0-1800, 0-3600 ve 7200 dev/dakika şeklinde hız ayarlama imkanı verir. Motor dönüş yönü ne olursa olsun, gösterge her zaman pozitif hız gösterir. Şekil 4.2 de A alanı görülmektedir[11].



Şekil 4-2 A Alanı

Tork ve hız için çıkış sinyalleri 5 iğneli DIN X-Y soketi yardımıyla dış ortama alınmaktadır. Şekil 4.3'te 5 iğneli DIN X-Y soketinin şekli görülmektedir.



Şekil 4-3 5 iğneli DIN soketi

Burada;

1'nci iğne 1V/1000 dev/dak şeklinde analog hız sinyal çıkışı üretir.

2'nci iğne 0-2 Nm tork değerleri için 0-2 V değerinde analog tork sinyal çıkışları üretir.

3 ve 4 'ncü iğneler 0V'luk referans sağlarlar.

A alanında bunların yanında şebeke anahtarı, şebeke gösterge ışığı ve 5A'lik şebeke sigortası vardır.

Şebeke anahtarı OFF durumunda iken FH2 MkIII'deki elektronik devrelere güç gitmez. ON durumunda iken şebeke gösterge ışığı yanar. Elektronik devreler ve diğer elmanlar çalışır durumdadır.

4.4 B Alanı – Makine Bağlantı Soketleri

Kontrol panelinin bu alanı 4mm'lik 6 tane emniyet soketi ile 3 durumlu bir anahtardan oluşmaktadır.

Test makinesi sağtarafındaki kızığa yerleştirilip 10-yol ve toprak soketleri uygun yerlerine takıldıktan sonra iç elektrik bağlantıları yapılabilir.

Kullanıcı makine sarımlarının iç bağlantılarını ilgili deneye uygun şekilde yapıp bu şekilde enerji, kontrol elemanı ve enstrüman bağlantılarını yapar. Bu yolla kullanılabilir ve doğru çalışan bir test düzeneği kurulabilir.

FH2 MkIII grubu makinelerin herbirisinin kendine has bağlantı gereksinimleri olduğundan her makine için özel mimik diyagramları da ayrıca verilmektedir. Bu diyagram makine sargı bilgilerini şematik olarak göstermektedir. Diyagram makine bağlantı soketleri ve START/STOP/RUN anahtarı üzerine konulduğunda bağlantıların nereye yapılacağı açık olarak görülür. Mimik diyagramı için panelde bulunan yerleştirme uçları diyagramın doğru pozisyonda durmasını sağlamaktadır.

3 kademeli anahtar FH80 tipi kapasitör motor ve FH70 tipi yardımcı sargılı endüksiyon motor için BAŞLA/DUR/ÇALIŞTIR, FH60 tipi A.C. gölgeli kutuplu endüksiyon motoru ve FH50 tipi Kompaund makineler için BAŞLA/DUR olanaklarını sağlamaktadır.

FH80 tipi kapasitör motor ve FH70 tipi yardımcı sargılı Endüksiyon motorlar için başlatma işlem sırası şöyledir.

Anahtar START(BAŞLA) durumunda iken, anahtar cihazın gerisine doğru basılı tutulur. Bu durumda motor özel kalkış devresi çalışır. Eğer anahtar bırakılırsa otomatik olarak (STOP) (DUR) durumuna gelecektir ve enerji kesilecektir. Motor yeter devir hızına ulaştınca ve akım güvenli çalışılabilir seviyeye inince anahtar hızlı bir şekilde STOP(DUR) üzerinden RUN(ÇALIŞTIR) durumuna alınır.

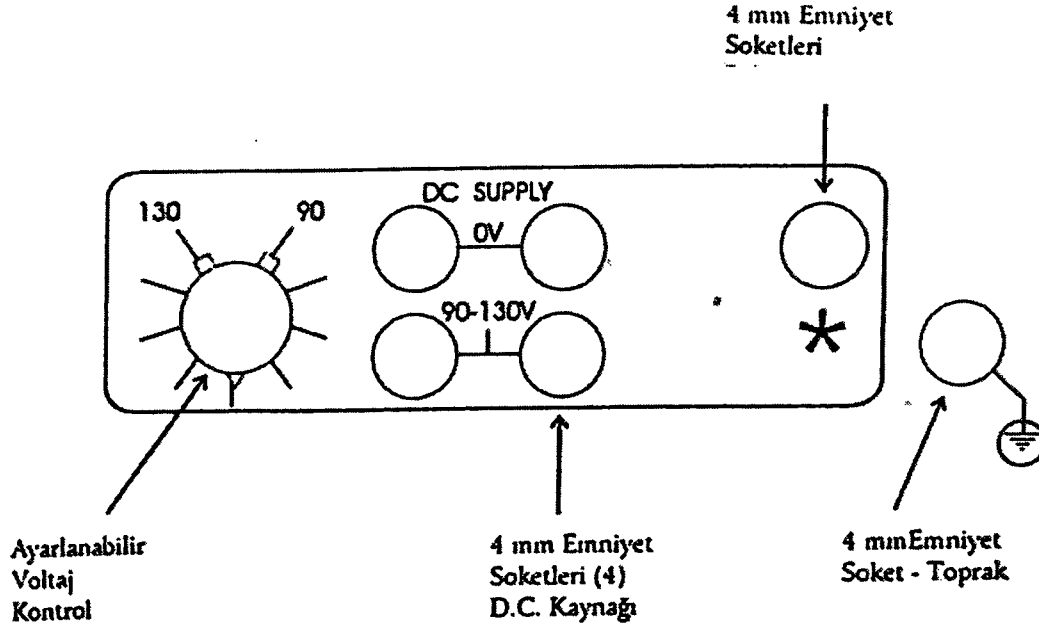
Yukarıdaki belirtildiği şekilde çalıştırma işlemi gerektiren makinelerde anahtarın el ile kontrol edilmesi bu makinelerin BAŞLA/DUR/ÇALIŞTIR durumlarının araştırılması ve değerlendirilmesine imkan vermektedir.

Diğer bazı makineler için başlatma işlemi direk STOP dan RUN'a geçilerek yapılabilir.FH50 tipi D.C. kompaund maline ve FH60 tipi gölgeli kutuplu motor bunlara örnek olarak sayılabilirler.

Motor yüklü iken bu tip anahtarlama işlemi yapılmamalıdır.

4.5 E Alanı – D.C. Güç Kaynağı

FH2 MkIII test yatağı 90-130V arası D.C.güç kaynağına sahiptir. 3A akıma kadar sınırı vardır. Aşırı yük durdurucu çalışıp D.C güç kaynağını devreden çıkarana kadar da bu akım artırılabilir. Aşırı yüklemeyen sonra aşırı yüklenme sebebi giderilerek D.C. güç kaynağı tekrar devreye sokulabilir. Bunun için D.C. güç kaynağını besleyen butonu kapatıp tekrar açmak yeterli olacaktır. Bu işlem ön paneldeki OFF ve ON basmalı butonlarla da yapılabilir. Şekil 4.4'te güç kaynağı bağlantı şekli görülmektedir.



Şekil 4-4 Güç kaynağı bağlantı yapısı[11]

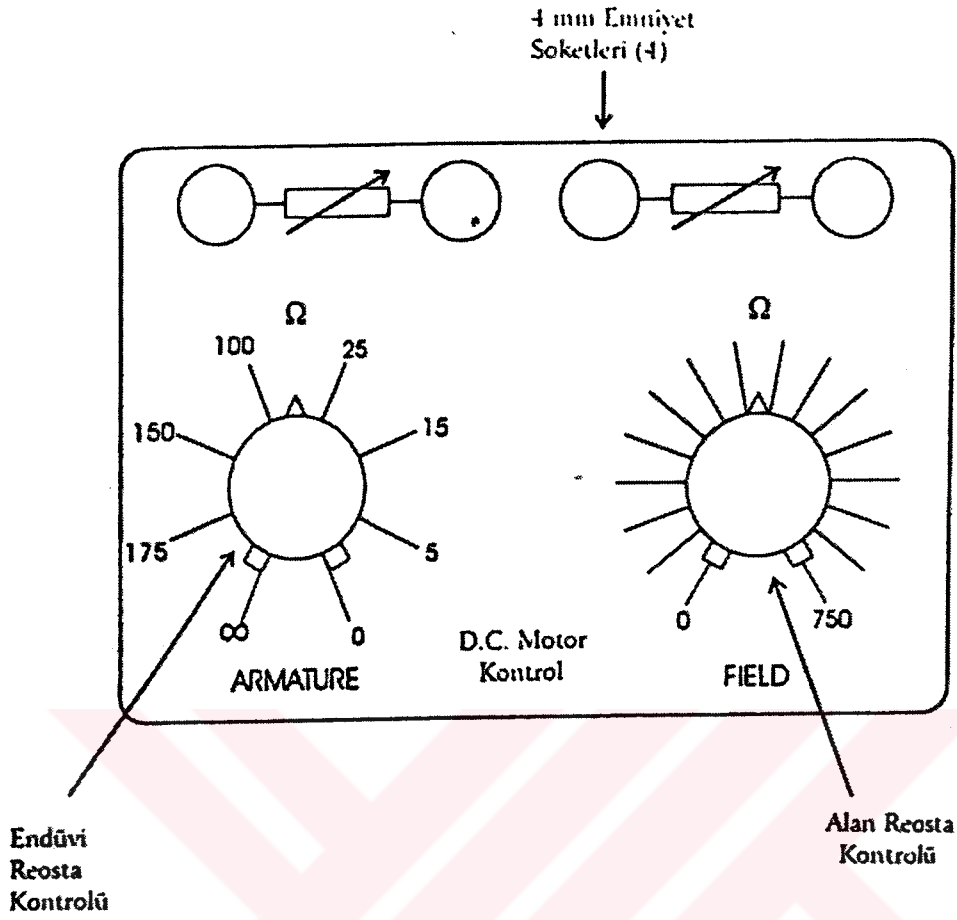
D.C. güç kaynağı şekilde görülen ikili soketler vasıtası ile kullanılabilir. İstenilen gerilim soketlerin yanındaki kontrol potansiyometresi yardımı ile ayarlanabilir. Kompaund makine sol kızağa ve çoklu yol soketine bağlandığında (*) ile işaretlenen soket iç taraftan FH50 kompaund makinenin şönt bobini ve endüvi ortak uçlarına bağlanır. (*) ile işaretli soket negatif hattın tamamlanması için D.C. güç kaynağının negatif soketine bağlanmalıdır (0Volt). Sol kızağa bağlanan FH50 bu bağlantılar olmadan jeneratör testlerinde primer motor olarak çalışmayacaktır.

Bu güç kaynağı 110V nominal çalışma gerilimi olan FH50 tipi D.C. kompaund makineye uygun olması için gerekli şekilde kademelendirilmiştir.

4.6 F Alanı – D.C. Motor Kontrolü

Bu kısımda bulunan 2 adet reosta FH50 tipi kompaund makinenin şönt bobin ve endüvi devresinde kontrol dirençleri olarak kullanmak içindir. FH50 tipi makine her iki kızağa da bağlanabilir. Fakat reostalar sadece bir makineyi kontrol edebilirler.

FH2 MkIII test cihazının dinamometresinin solundaki kızak, FH50 tipi makineler için hazırlanmıştır. Bu makine A.C. D.C. jeneratör testleri için primer motor olarak gerekecektir. Senkron motorları senkronize hıza ulaştırmak için sürücü motor olarak da kullanılabilir. D.C. jeneratör denyeleri için biri esas jeneratör, diğeri primer motor olmak üzere iki adet FH50 tipi makine kullanmak gerekir. Şekil 4.5'te kontrol düğmeleri görülmektedir.



Şekil 4-5 Kontrol düğmeleri[11]

FH50 tipi kompaund makine, jeneratör/senkron motor deneylerinde kullanılacağı zaman, dinamometrenin sol tarafındaki kızağa bağlanır. 10-yol fişi ve toprak ucu uygun yerlere takılırlar. Bundan sonra D.C. makine ile olan bağlantı 175 Ohm'luk endüvi reostası ve 750 ohm'luk sargı reostası üzerinden otomatik olarak yapılır. Endüvi ve sargı reosta ara bağlantıları (90-130V) D.C. kaynağın pozitif ve E ALANININ sağında bulunan (*)işaretili negatif (0V) ucuna yapılması gerekir.

Endüvi sargı reostalarının değerlerinin değiştirilmesi motor dönme hızlarında direk kontrol imkanı vermektedir. Bunu hız göstergesinden izlemek mümkündür.

Cihazın üst kısmında Fuko akımlı dinamometre freni ve iki adet makine test yeri vardır. Dinamometre freni her iki ucundan çapraz yataklara dayanan makine çerçevesini içerir.

Dinamometrenin tork hissedici yük kirişi vasıtası ile önlenmiştir. Bu, çerçevenin altına gizlenmiş cihazın şasesinde korumalı bir yuvaya yerleştirilmiştir. Dinamometre çerçevesine gelen herhangi bir yük kirişin esnemesine sebep olacaktır. Bu küçük esnemeler bir strain geyç köprüsü vasıtası ile elektrik sinyallerine dönüştürülürler. Bu

sinyaller uygulanan tork ile orantılıdır. Kutuplaşmalar dönüş yönü tarafından kontrol edilmektedir.

Dinamometrenin ölçüm sınırı 0-2 Nm'dir. Dinamometreyi elle 2 Nm'den daha fazla yükleme imkanı yoktur. Bu çok önemlidir. Eğer 2 Nm'den fazla yüklenebilseydi yük kirişi özelliğini kaybeder ve ölçüm değerlerinin hassasiyetinde ve doğrusallığında hatalar oluşurdu.

Sinyaller tork metre vasıtasıyla işlenir ve görüntülenir. Tork sinyalinin polaritesi Ters tork metre anahtarı vasıtasıyla tork metreye uygun hale dönüştürülmektedir.

Motorlara yol verme tork testi yapılacaksa, FH2 MkIII ile verilen bağlama pimleri kullanılarak dinamo kavraması şafta kilitlemelidir. Bunun için şaftı elle döndürüp pim yerlerine uydurmak gerekebilir.

Dinamometre çerçevesinin içine hız ile orantılı olarak sinyal olarak sinyal üreten optik bir sensör konulmuştur. Hız metrede asıl hızı görüntülemek için bu sinyaller cihazda bulunan elektronik devre tarafından işlenir ve sonuç hız metrede görüntülenir.

Tork ve hız ölçüm sistemleri fabrikada kalibre edilmişlerdir. kullanmadan önce yeniden kalibre etmeye gerek yoktur. Fakat belli bir süre kullanıldıktan sonra kalibrasyondaki değişmelerin tekrar kalibre edilerek giderilmeleri gerekir.

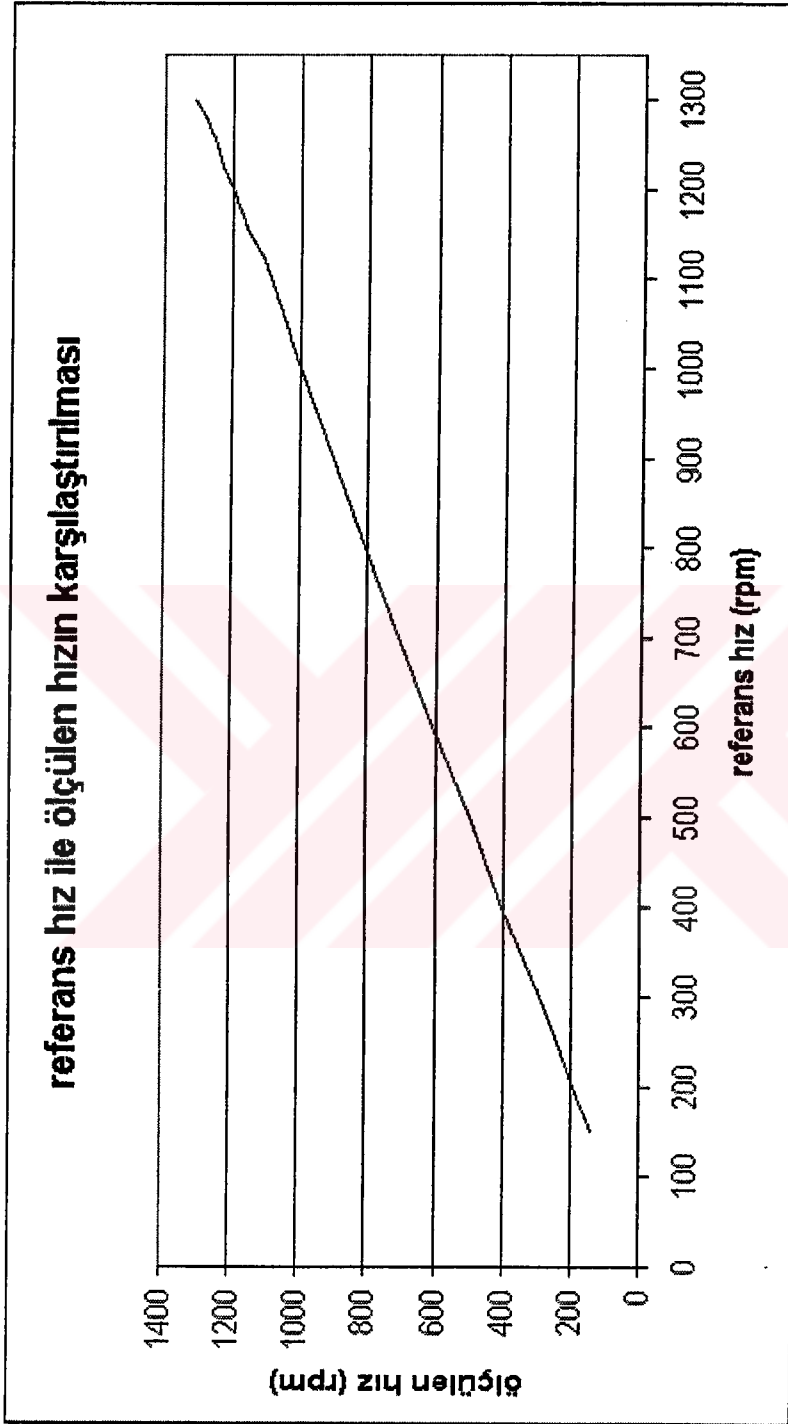
4.7 FH50 DC Motorun Davranışının İncelenmesi

FH2MkIV setinde kullanılan FH50 tipi DC motor bulanık kontrolör sistemini tasarlayabilmek için kontrol edilmesi amacıyla, önce FH50 DC motorun davranışını incelememiz gerekmektedir. Bu amaçla bir dizi deneyler yapılmıştır. Bu deneyler bulanık kontrolörün tasarımında yardımcı olmuştur.

4.7.1 Yapılan Deneyler

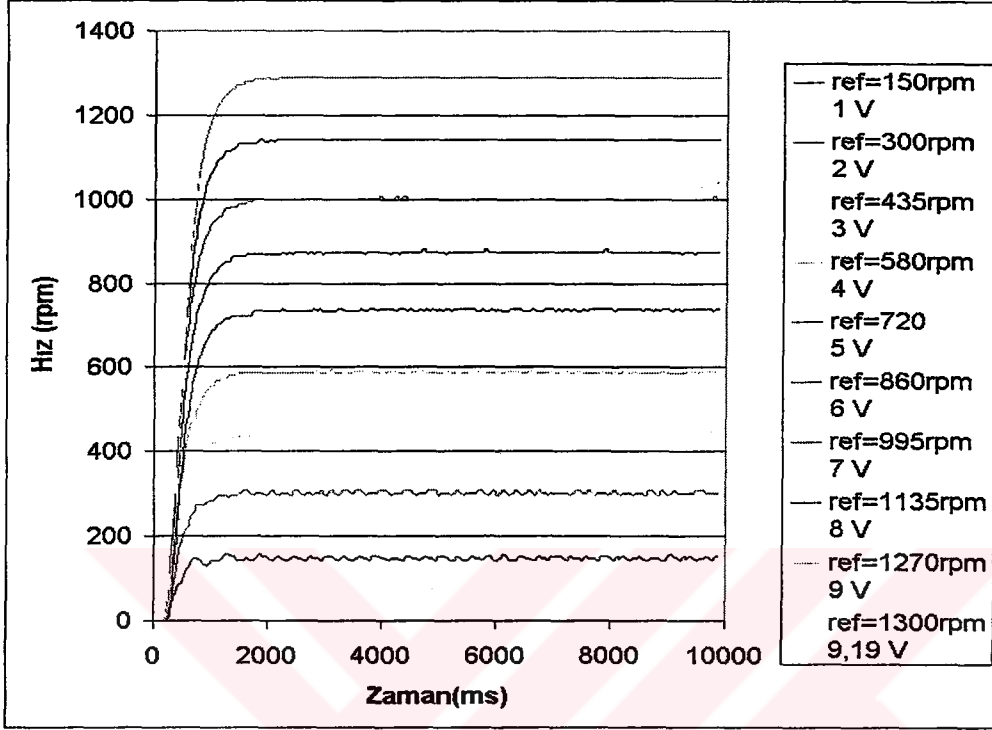
FH2MkIV setinde kullanılan FH50 tipi DC motorun davranışını incelemek amacıyla sürücü vasıtasıyla DC motora çeşitli sinyaller uygulanmıştır. Hız ve hız değişimleri incelenmiştir. Bunun sonucunda açık çevrim cevabı elde edilmiştir. Bu veriler vasıtasıyla sistem giriş değişkenleri olarak motor hız ve hız değişimi bilgilerinden üyelik fonksiyonları oluşturularak bulanıklaştırma işlemi gerçekleştirilmiştir. Temin edilen (FH2MkIV setinde bulunmamaktadır) DC motor sürücü 0-10V arasında çalışmaktadır. Çeşitli gerilimlerde sistemin davranışına ait grafikler çıkarılmıştır. Şekil 4.6'da sistemin kontrolsüz çalışmada hızının yaklaşık

lineer olduđu gözlenmiştir. Ancak sistemin 150rpm kadarki hızları motorun kalkınamaması nedeniyle okunamamıştır.



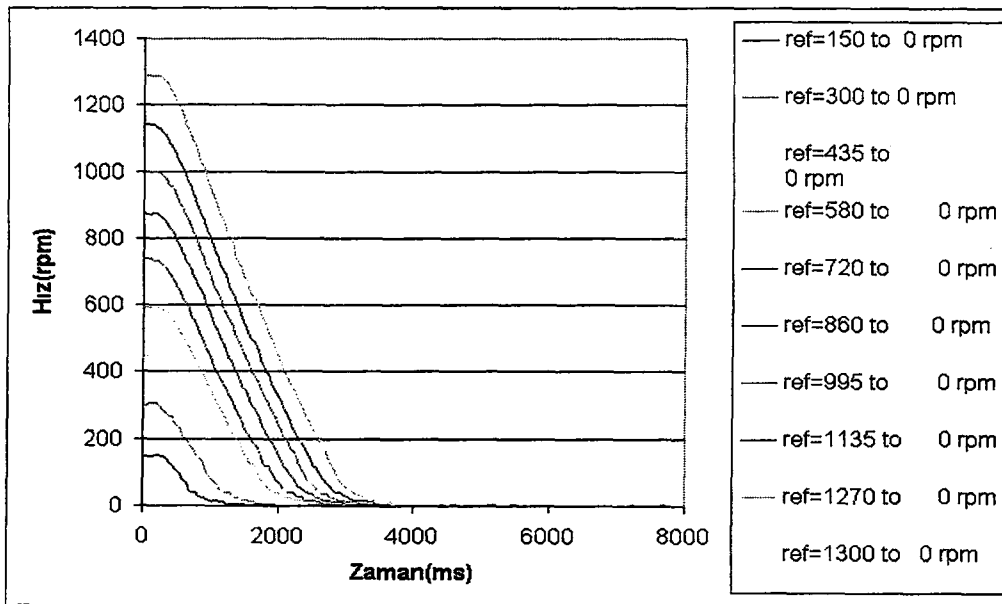
Şekil 4-6 Referans hız ile ölçülen hız arasındaki ilişki

DC motor çeşitli hızlarda kontrolsüz olarak çalıştırılmış ve bunlara ait grafik şekil 4.7'de gösterilmiştir. Bu grafikte sistemin düşük hızlarda titreşimli çalıştığı, hız yükseldikçe bu titreşimin azaldığı görülmüştür.



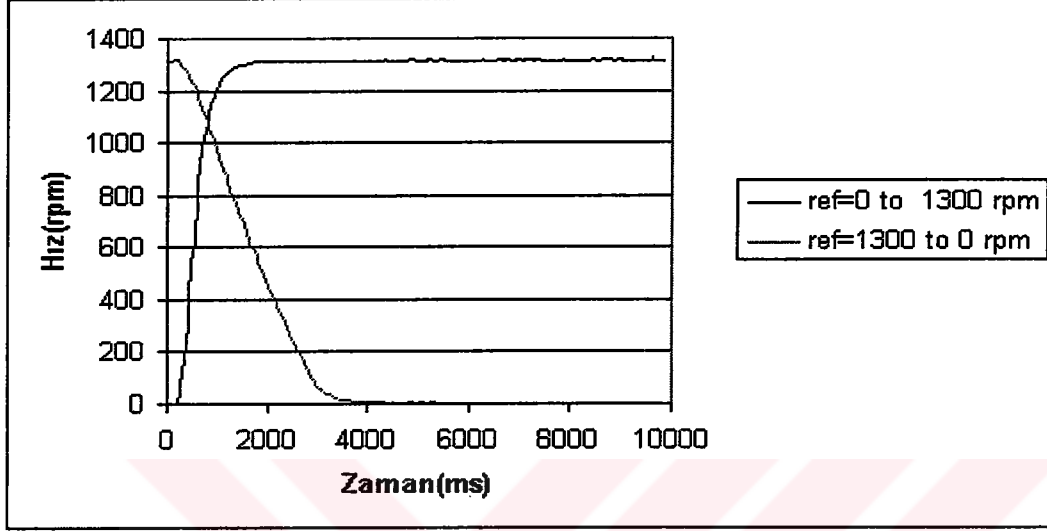
Şekil 4-7 Çeşitli hızlara ait DC motorun davranışı

DC motor hızı şekil 4.7'de ki hızlara çıkarılmış daha sonra bu hızlardan 0 rpm hıza düşürülmüş bunlara ait grafik Şekil 4.8'de görülmektedir.



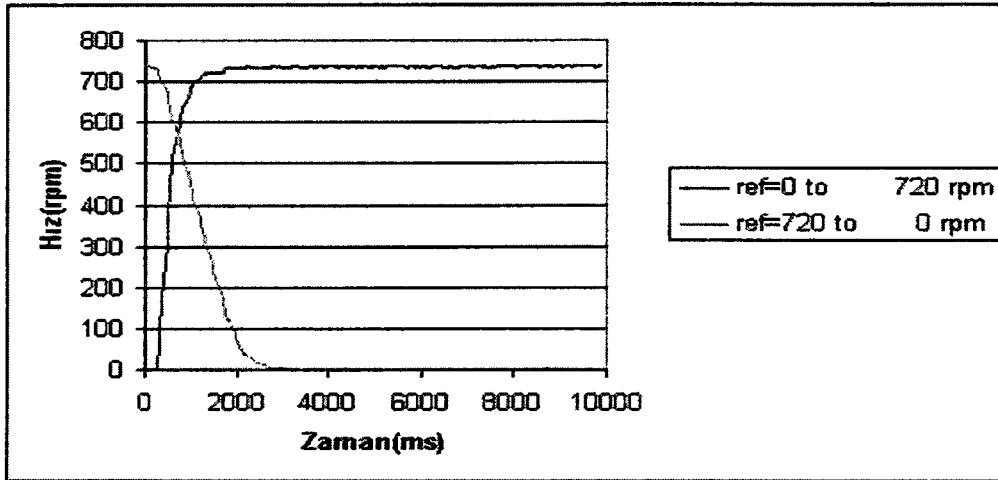
Şekil 4-8 Çeşitli aralıktaki hızların "0" a düşüş eğrileri

İkinci denemede motor hızı 1300 rpm'e ayarlanmış motor bu hızda 10sn kadar çalıştırılmıştır. Motorun yaklaşık 1,8 sn'de referans değerine ulaştığı görülmüştür. Sonraki denemede motor 1300 rpm'den 0 rpm'e indirilmiştir. Buradada motor hızının 3,8 sn de 0'a ulaştığı görülmüştür.



Şekil 4-9 Hızın 0'dan 1300 rpme çıkış ve 1300 rpm den 0'a inişi

İkinci deneme ise 720 rpm için yapılmıştır. Buna ait grafik Şekil 4.10'da görülmektedir. Grafikten de görüleceği gibi motor hızı yaklaşık 1,7sn'de referans değerine ulaşırken, 2,8sn'de 0'a ulaştığı görülmektedir.



Şekil 4-10 Hızın 0 dan 720 rpme çıkış ve 720 rpm den 0 a inişi

BÖLÜM 5

BULANIK KONTROLÖR TASARIMI

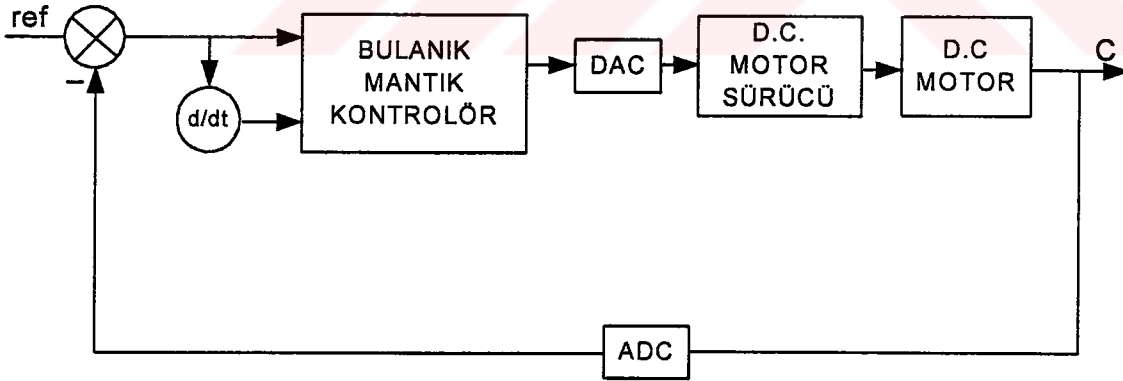
5.1 Giriş

Elektrik Makineleri Eğitim Seti (FH2 MkIV)'nin bilgisayar seri portu üzerinden bulanık kontrolör ile kontrol edilmesi gerçekleştirilmiştir. Bu amaçla bir mikro denetleyici kart (data acquisition) tasarlanmıştır. Ayrıca bilgisayar ile kontrolün yapılabilmesi, kuralların tanımlanması, bunların kullanıcı tarafından değiştirilmesi, grafiklerin çizilmesi amacıyla da bir kullanıcı arabirimi (user interface) program geliştirilmiştir.

Bu bölümde DC motor kontrolü için gerçekleştirilen Mikro Kontrol Kartının yapısı ve bulanık kontrolör yazılımı ayrıntılı olarak anlatılacaktır.

5.2 Tasarlanan Kontrolörün Genel Yapısı

DC motor kontrolü için gerçekleştirilen kontrol sisteminin blok diyagramı Şekil 5.1'de görülmektedir. Sistemi oluşturan elemanlar pentium tabanlı bir bilgisayar, haberleşme ve dijital-analog dönüşümleri yapan Mikro Denetleyici Kart ve DC motor sürücünden oluşmaktadır.



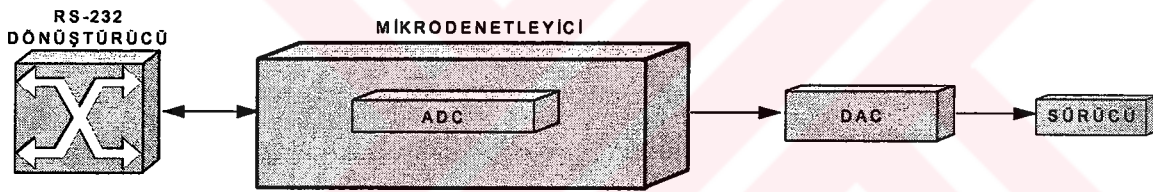
Şekil 5-1 Sistem için tasarlanan kontrolörün genel yapısı

Bilgisayar üzerindeki programa motorun hangi hızda dönmesi gerektiği, bulanık kuralların neler olacağı bilgileri girilir. Motorun hangi hızda döneceği seri port üzerinden haberleşen mikrokontroller kartına bildirilir. Mikro denetleyici kart buna göre bir sinyal üreterek sürücü üzerinden DC motora uygular. Elektrik Makineleri Eğitim Seti (FH2 MkIV) üzerinde 5'li DIN soketinin 1 ve 3 nolu pinlerinden analog

olan hız bilgisi Mikro Denetleyici Kart'ın ADC girişine uygulanmaktadır. Burada sayısal işarete çevrilen hız bilgisi Mikro Denetleyici Kart tarafından bilgisayara aktarılır. Bilgisayar tarafından alınan sayısal hız bilgisi bulanıklaştırılıp çıkarma ünitesinde elde edilen bulanık kontrol işareti, berraklaştırma ünitesinde sayısal kontrol işaretine çevrilir. Elde edilen sayısal kontrol işareti seri port üzerinden Mikro Denetleyici Kart'a iletilir. Mikro Denetleyici Kart da bu bilgiyi sürücü vasıtasıyla motora uygulayarak hız kontrolünü gerçekleştirir.

5.3 Mikro Denetleyici Kart

Sitemin taşınabilir ve kolay uygulanabilir olması düşünülmüştür. Bu amaçla bir bilgisayar içine yerleştirilen hazır veri işleme kartları yerine, bütün bilgisayarlarda (dizüstü, masaüstü, endüstriyel vb.) standart olan RS-232 seri portu üzerinden haberleşebilen bir kart dizayn edilmiştir. Şekil 5-2'de mikro denetleyicinin yapısı görülmektedir. Mikro Denetleyici Kart'a ait devre şemaları Ek-3'te verilmiştir.



Şekil 5-2 Mikro Denetleyici kartın yapısı

Tasarlanan Mikro Denetleyici Kart üzerinde bir mikro denetleyici ile veriler işlenmekte ve bilgisayara geri aktarılmaktadır. Mikro Denetleyici Kart dört ana birimden oluşmaktadır.

RS-232 Seri haberleşme çevre birimi

Mikro denetleyici

ADC (analog dijital çevirici)

DAC (dijital analog çevirici)

Aşağıdaki bölümlerde bu birimler ayrıntılı olarak açıklanacaktır.

5.3.1 RS-232 Dönüştürücü Birimi

Mikro Denetleyici Kart'ın dış dünya ile haberleşmesi için serihaberleşme biçimi seçilmiştir. Seri haberleşmenin, kablo bağlantı azlığı, uzak mesafelerde sinyal bozulmaması, hemen her bilgisayar ve diğer bilgi işlem cihazlarında bir seri portun(com) bulunması gibi vazgeçilemez bir çok avantajı vardır. Herhangi bir masa

üstü veya dizüstü bilgisayar ile (herhangi bir bilgisayara bağlı değilsiniz) haberleşme sağlanabilir.

Seri haberleşmede kullanılan RS-232 standardının özellikleri şunlardır;

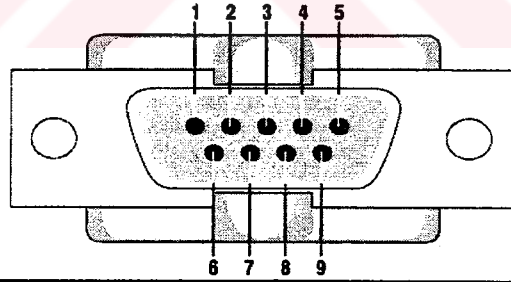
Değişik üreticiler tarafından yapılmış veri haberleşme cihazlarının uyumluluğunu sağlamak amacıyla, EIA (Electronics Industries Association) tarafından RS232 olarak adlandırılan standart 1960 yılında belirlendi. 1963 yılında ilk standart değiştirildi ve RS232A olarak adlandırıldı. Daha sonra 1965'te RS232B ve 1969 da RS232C standartları ilan edildi.

Günümüzde RS232 en yaygın kullanılan seri I/O arabirim standardıdır. Bu standart TTL lojik ailesinden çok önceleri belirlendiği için, giriş ve çıkış seviyeleri TTL uyumlu değildir. RS232 de lojik 1; -3V ile -25V arasında, lojik 0; +3V ile +25V arasında tanımlanır. -3V ile +3V arası tanımsızdır. Bu yüzden herhangi bir RS232 portu bir mikro işlemci cihaza bağlamak için RS232 dönüştürücü entegre devreler kullanılır. RS-232 portu ve bağlantı uçları Şekil 5.3 ve Şekil 5.4 te, bağlantı uçlarının açıklamaları Tablo 5-1'de verilmiştir.

Kartımızda RS-232 işaret seviye dönüştürücü olarak MAXIM firmasının MAX232 entegresi kullanılmıştır. Bu entegre $\pm 7V$ eşliğinde bir dönüştürme yapar. Buda bilgisayar ile haberleşmede yeterlidir.



Şekil 5-3 Bilgisayarlarda bulunan 9 pin COM port.



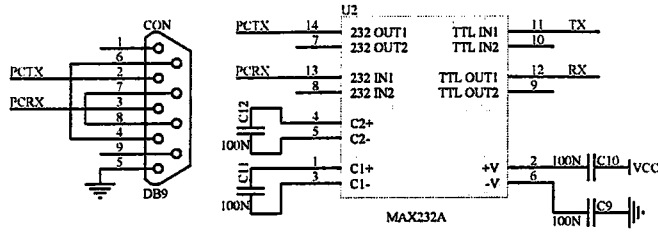
Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

Şekil 5-4 9pin konnektor bağlantı uçları

Tablo 5-1 9 Pin bağlantı uçlarının açıklamaları

Uç No	Sinyal adı	Açıklama
1	Data Carrier Detect (DCD)	Modem, telefon hattından bir taşıyıcı (carrier) belirlediğinde, bu hattı aktif yaparak DTE ye bu durumu haber verir.
2	Receive Data (RxD)	Alınan veri
3	Transmit Data (TxD)	Gönderilen veri
4	Data Terminal Ready (DTR)	DTE (Data Terminal Equipment-PC COM portu) çıkış, modem giriş ucudur. Bu hat modeme terminalin veri iletişimi için hazır olduğunu belirtir. COM prtunda bir problem var ise, bu sinyal aktif yapılmaz.
5	Signal Ground (GND)	Sinyal ground
6	Data Set Ready (DSR)	DTE giriş, modem çıkış ucudur. Bu hat terminale modemin veri iletişimi için hazır olduğunu belirtir. Eğer modem herhangi bir nedenden dolayı telefon hattına bağlanamıyorsa, bu sinyal pasif yapılarak, veri göndermek veya almak için modemin hazır olmadığı PC'ye belirtilir.
7	Request To Send (RTS)	DTE göndereceği verisi olduğunda, bu uçla haber verir.
8	Clear To Send (CTS)	RTS sinyaline cevap olarak, veri almaya hazır olduğunu belirtir.
9	Ring Indicator (RI)	Zil işareti

Şekil 5-5'te kartta kullanılan RS232 dönüştürücü devresi görülmektedir. Şekilde aynı zamanda 9 pin bağlantı şeklide gösterilmiştir.



Şekil 5-5 Mikro denetleyici kartında kullanılan RS232 sürücü devresi

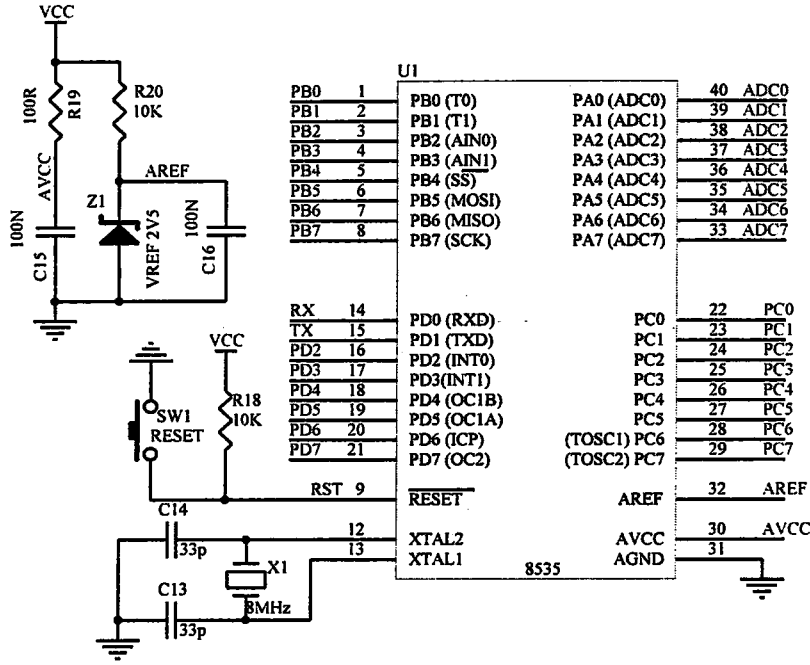
Seri haberleşme amacıyla mikro denetleyici aşağıda ki kod yardımıyla ayarlanır.

```
// UART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// UART Receiver: On
// UART Transmitter: On
UCR=0xD8;
// UART Baud rate: 9600
UBRR=0x19;
```

C dilinde yazılmış kodun tamamı Ek-2'de verilmiştir.

5.3.2 Mikro Denetleyici Kartı

Mikro Denetleyici Kart'ın en önemli kısmı mikro denetleyici katıdır. Buna ait şema Şekil 5.6 da verilmiştir.



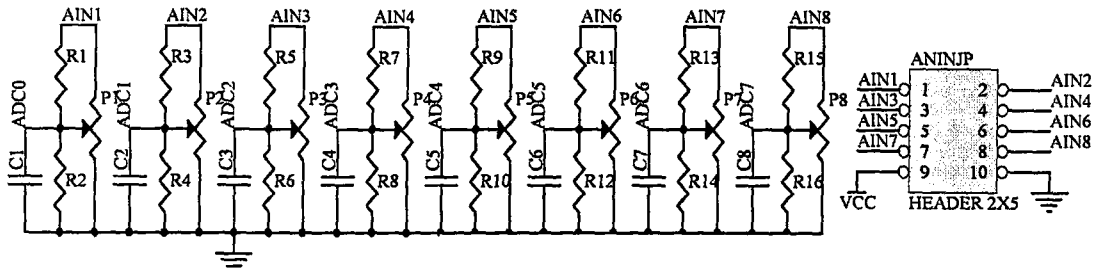
Şekil 5-6 Mikro denetleyici kartı

Kartın en önemli kısmı olan mikro denetleyici (mikrokontrolör) entegre olarak ATMEL® firmasına ait AT90S-8535 mikro denetleyicisi kullanılmıştır. Bu entegre ; 1 saykılada 1 komut işleyebilme (dallanma komutları hariç), 8 kanal 10 bit ADC, 1 programlanabilen uart, 3 sayıcı/zamanlayıcı, 2 pwm kanalı, flash program hafıza, (incircuit programmable) kart üzerinde sökülmeden programlanabilme gibi daha bir çok özelliklere sahiptir[12].

Karttaki beyin görevini üstlenmektedir. Haberleşme, Analog-dijital çevrim işlerini yönetir.

5.3.3 ADC (Analog Dijital Çevirici) Kartı

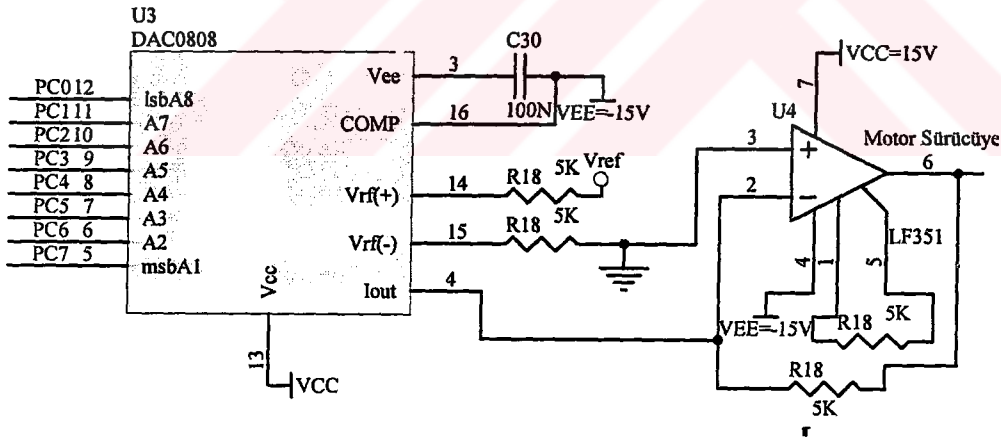
Bu birim gerçekte AT90S8535 içinde bulunmaktadır. Ancak dışarıdan analog bilgiler Şekil 5.7 yardımıyla alınmaktadır.



Şekil 5-7 Analog giriş kartı

5.3.4 DAC (Dijital Analog Çevirici) Kartı

DC Motorun kontrol edilebilmesi için, motor sürücü devresine 0-10 V arası bir gerilim uygulamak gerekir. Bu da mikro denetleyicinin C portu sayısal çıkışlarının DAC0808 tüm devresine uygulanmasıyla elde edilir. DAC0808 çıkışında akımı gerilime çevirmek amacıyla, LF 351 tümdevresiyle voltaj follower yapılarak sağlanmıştır. Şekil 5.8'de DAC katına ait şema görülmektedir.



Şekil 5-8 DAC kartı

5.3.5 Motor Sürücü

Motor ikaz sargılarına 115V, armature uçlarına max 130V gerekmektedir. Set üzerinde bunu sağlamak amacıyla 0-130V ayarlanabilir bir anahtarlmalı güç kaynağı bulunmaktadır. Ancak, set elektrik makinaları temel eğitimi vermek amacıyla tasarlandığından motor kontrol devresi içermemektedir. Bu sebepten dolayı bir DC

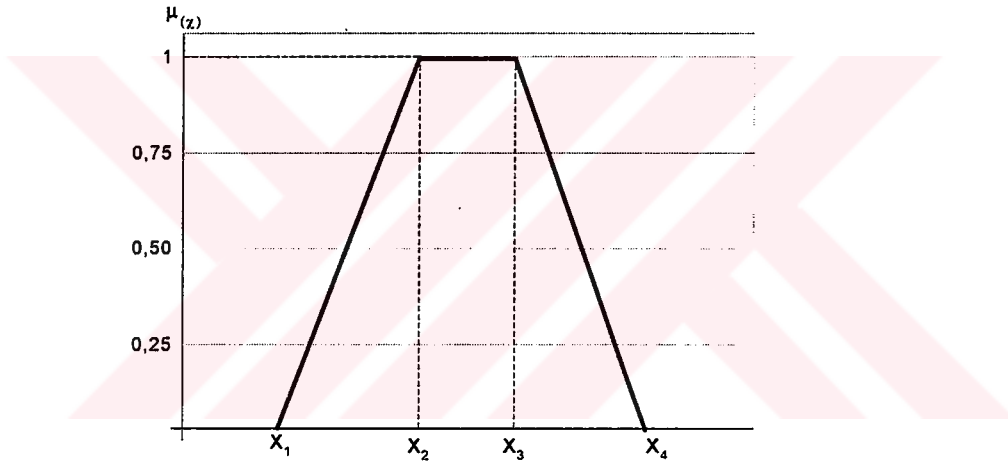
motor sürücü temin edilmiştir. Motor ikaz sargıları için gerekli olan gerilim set üzerinde sağlanmakta , armature gerilimi sürücü tarafından sağlanmaktadır. Sürücü kontrol girişlerine 0-10V uygulanarak hızın kontrol edilmesi sağlanmıştır. Motor sürücü devresine ait şema Ek- 4 te verilmiştir.

5.4 Bulanıklaştırma İşlemleri

Tasarlanan bulanık kontrolörün giriş değişkenleri motor hızı ve hızdaki değişim olarak alındı. Aşağıdaki bölümlerde değişkenlere ait bulanıklaştırma işlemleri açıklanmıştır.

5.4.1 Üyelik Fonksiyonlarının Seçimi

Bulanık kontrolörün giriş değişkenleri olan hız, hız değişimi ve referans Şekil 5-9'da verilen üyelik fonksiyonu kullanılarak bulanık forma dönüştürülmüştür.



Şekil 5-9 Kontrol sisteminde kullanılan üyelik fonksiyonu

Bu fonksiyonu dört bölgede inceleyecek olursak: x_1-x_2 ve x_3-x_4 bölgeleri birinci dereceden bir polinomdur. x_2-x_3 bölgeleri ise birim basamak fonksiyonudur. $x < x_1$ ve $x > x_4$ durumlarında ise $\mu(x)$ sıfırdır. Polinomun ifadesi aşağıda verilmiştir.

$$\mu(x)=0 \quad x < x_1 \quad (5-1)$$

$$\mu(x)=\frac{x-x_1}{x_2-x_1} \quad x_1 < x \leq x_2 \quad (5-2)$$

$$\mu(x)=1 \quad x_2 < x \leq x_3 \quad (5-3)$$

$$\mu(x) = \frac{X_4 - X}{X_4 - X_3} \quad X_3 < X \leq X_4 \quad (5-4)$$

$$\mu(x) = 0 \quad X > X_4 \quad (5-5)$$

5.4.2.1 Motor hız bilgisinin bulanıklaştırılması

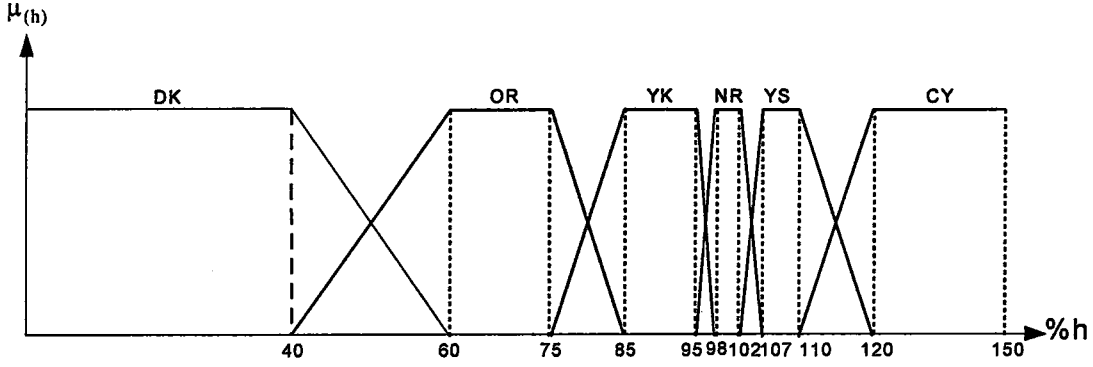
Sistemde motor hızı 0 ile 1300 rpm arasında değişmektedir. Hızın bulanıklaştırılmasında hız değerleri %hız olarak alınmıştır. Böylece değişik referans hız değerlerinde bulanık kontrolörün sistemi daha duyarlı bir şekilde kontrol etmesi sağlanmıştır.

Kullanılan üyelik fonksiyonları "Düşük (DK)", "Orta(ORT)", "Yakın(YK)", "Normal(NR)", "Yüksek(YS)" ve "Çok yüksek(CY)" olmak üzere altı adettir. Üyelik fonksiyonları ve sınır değerleri Tablo 5.2'de görülmektedir.

Tablo 5-2 Hızın bulanıklaştırılması

Bulanık Küme	Sınırlar	x ₁	x ₂	x ₃	x ₄
Düşük (DK)	0,00 ≤ Hız < 0,60	0,00	0,00	0,40	0,60
Orta (ORT)	0,40 ≤ Hız < 0,85	0,40	0,60	0,75	0,85
Yakın (YK)	0,75 ≤ Hız < 0,98	0,75	0,85	0,95	0,98
Normal (NR)	0,95 ≤ Hız < 1,05	0,95	0,98	1,02	1,05
Yüksek (YS)	1,02 ≤ Hız < 1,20	1,02	1,07	1,10	1,20
Çok Yüksek (CY)	1,10 ≤ Hız < 1,50	1,10	1,20	1,50	1,50

Hıza ait üyelik fonksiyonunun gösterimi Şekil 5.10'da gösterilmektedir.



Şekil 5-10 Hız'a ait üyelik fonksiyonları

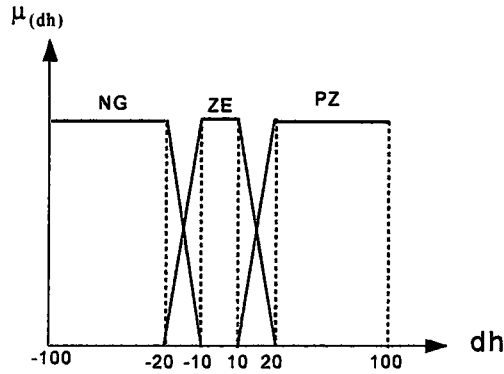
5.4.2.2 Motor hız değişiminin Bulanıklaştırılması

Hız değişiminin bulanıklaştırılması için üç ayrı bulanık küme kullanılmıştır, bunlar ; "Negative(NG)", "Zero (ZE) ve "Positive (PZ) "dir. Hız değişimine ilişkin üyelik fonksiyonları ve sınır değerleri Tablo 5.3'te verilmiştir.

Tablo 5-3 Hız değişiminin bulanıklaştırılması

Bulanık Küme	Sınırlar	x_1	x_2	x_3	x_4
Negatif (NG)	$-100 \leq \text{Hız} < -10$	-100	-100	-20	-10
Zero (ZE)	$-20 \leq \text{Hız} < 20$	-20	-10	10	20
Pozitif (PZ)	$10 \leq \text{Hız} < 100$	10	20	100	100

Hız değişimine ait üyelik fonksiyon grafiği Şekil 5.11'de görülmektedir.



Şekil 5-11 Hız değişiminin bulanıklaştırılması

5.4.2 Kural Tabanı

DC motor hız kontrolü için tasarlanan kontrolörde kuralların yapısı; eğer ve ise terimlerinden oluşmaktadır. kurallardaki “Eğer” teriminde giriş değişkenlerinin bulanık ifadeleri, “ise” teriminde de her kurala ait kontrol davranışının sayısal değerlerini içermektedir. Kontrol aşamasında hız ve hız değişimi ile ilgili 18 kural yazılmıştır. Her bir kural tesbit edilen kendi kontrol davranışı oranında çıkış kontrol işaretine etki etmektedir.

5.4.2.1 Bulanık Kontrol Kuralları

Bulanık kontrol tasarımında kural tabanı aşağıda verilmiştir. 18 adet kuraldan oluşmaktadır.

- # 1: Hız düşük ve Değişim negatif veya zero ise kontrol sinyali:0,75
- # 2: Hız düşük ve Değişim pozitif ise kontrol sinyali:0,5
- # 3: Hız düşük veya orta ve Değişim Pozitif ise kontrol sinyali:0,4
- # 4: Hız orta ve Değişim pozitif ise kontrol sinyali:0,1
- # 5: Hız orta veya yakın ve Değişim pozitif ise kontrol sinyali:0,02
- # 6: Hız yakın ve Değişim pozitif ise kontrol sinyali:0,02
- # 7: Hız yakın veya normal ve Değişim pozitif ise kontrol sinyali:0,3
- # 8: Hız normal ve Değişim pozitif ise kontrol sinyali:0
- # 9: Hız normal ve Değişim zero ise kontrol sinyali:0
- # 10: Hız normal veya yüksek ve Değişim zero ise kontrol sinyali:-0,01
- # 11: Hız normal veya yüksek Değişim zero veya negatif veya pozitif ise kontrol sinyali:-0,01
- # 12: Hız normal veya yüksek Değişim zero veya pozitif ise kontrol sinyali:-0,075
- # 13: Hız çok yüksek Değişim pozitif veya zero ise kontrol sinyali:-0,06
- # 14: Hız çok yüksek ve Değişim negatif ise kontrol sinyali:-0,3
- # 15: Hız yüksek ve Değişim negatif ise kontrol sinyali:-0,3
- # 16: Hız orta ve Değişim zero ise kontrol sinyali:0,002
- # 17: Hız orta veya yakın ve Değişim zero ise kontrol sinyali:0,065
- # 18: Hız yakın ve Değişim zero ise kontrol sinyali:0,02

Tablo 5.4'te bulanık kontrol kuralları verilmiştir.

Tablo 5-4 Bulanık kontrol kuralları

KURAL NO	HIZ	DEĞİŞİM	KONTROL İŞARETİ
1	DK	NG,ZE	0.75
2	DK	PZ	0.5
3	DK,OR	PZ	0.4
4	OR	PZ	0.1
5	OR,YK	PZ	0.02
6	YK	PZ	0.02
7	YK,NR	PZ	0.3
8	NR	PZ	0
9	NR	ZE	0
10	NR,YS	ZE	-0.01
11	NR,YS	NG,ZE,PZ	-0.01
12	YS,NR	ZE,PZ	-0.075
13	CY	ZE,PZ	-0.6
14	CY	NG	-0.3
15	YS	NG	-0.3
16	OR	ZE	0.002
17	OR,YK	ZE	0.065
18	YK	ZE	0.02

5.5 Kontrol İşaretinin Bulunması

DC motorun, hız bilgisi analog olarak ölçüldükten sonra sayısal değişkenlere dönüştürülür. Bu değişkenleri bulanıklaştırabilmek için daha önceden belirlenmiş bulanık kümelerden hangisine ait olduğu tespit edilir. Bir değişken aynı grup içerisinde birden fazla bulanık kümeye girebilir. Değişkenin ait olduğu kümeler belirlendikten sonra, bulanıklaştırılacak sayısal bilginin, ait olduğu bulanık kümelerin ne oranda elemanı olduğu denklemlerle hesaplanır .

Bu şekilde bulanıklaştırılmış değişkenler, bulanık kontrol kurallarının hepsine uygulanarak aktif olan kurallar tespit edilir. Burada da birden fazla kural aktif olabilir aktif olan kurallar içerisinde aynı gruba ait değişkenlerinin üyelik değerlerini minimumu

seçilir. Elde edilen minimum değer ile , aynı kural içindeki diğer değişkene ait üyelik değerleri karşılaştırılarak en büyük değerlikte olanı seçilir. Bu değer, aktif olan kuralın "Gerçekleştirme Oranı" (Degree of fulfilment, DOF) olarak tanımlanır.

Aktif olan kurallara ait gerçekleştirme oranı değerleri bulunduğundan sonra uzman kişi tarafından daha önceden tespit edilen kurallara ait kontrol davranışı değerleri (action) seçilir.

5.5.1 Kontrol İşaretinin Berraklaştırılması

Çıkarım ünitesinde elde edilen bulanık kontrol işareti, berraklaştırıcı biriminde sayısal kontrol işaretine çevrilir. Berraklaştırma işlemi ağırlık merkezi yöntemi ile gerçekleştirilmiştir.(Dk 5.6)

$$U = \frac{\left[\sum_{i=1}^n U(i) \cdot \text{DOF}(i) \right]}{\left[\sum_{i=1}^n \text{DOF}(i) \right]} \quad (5-6)$$

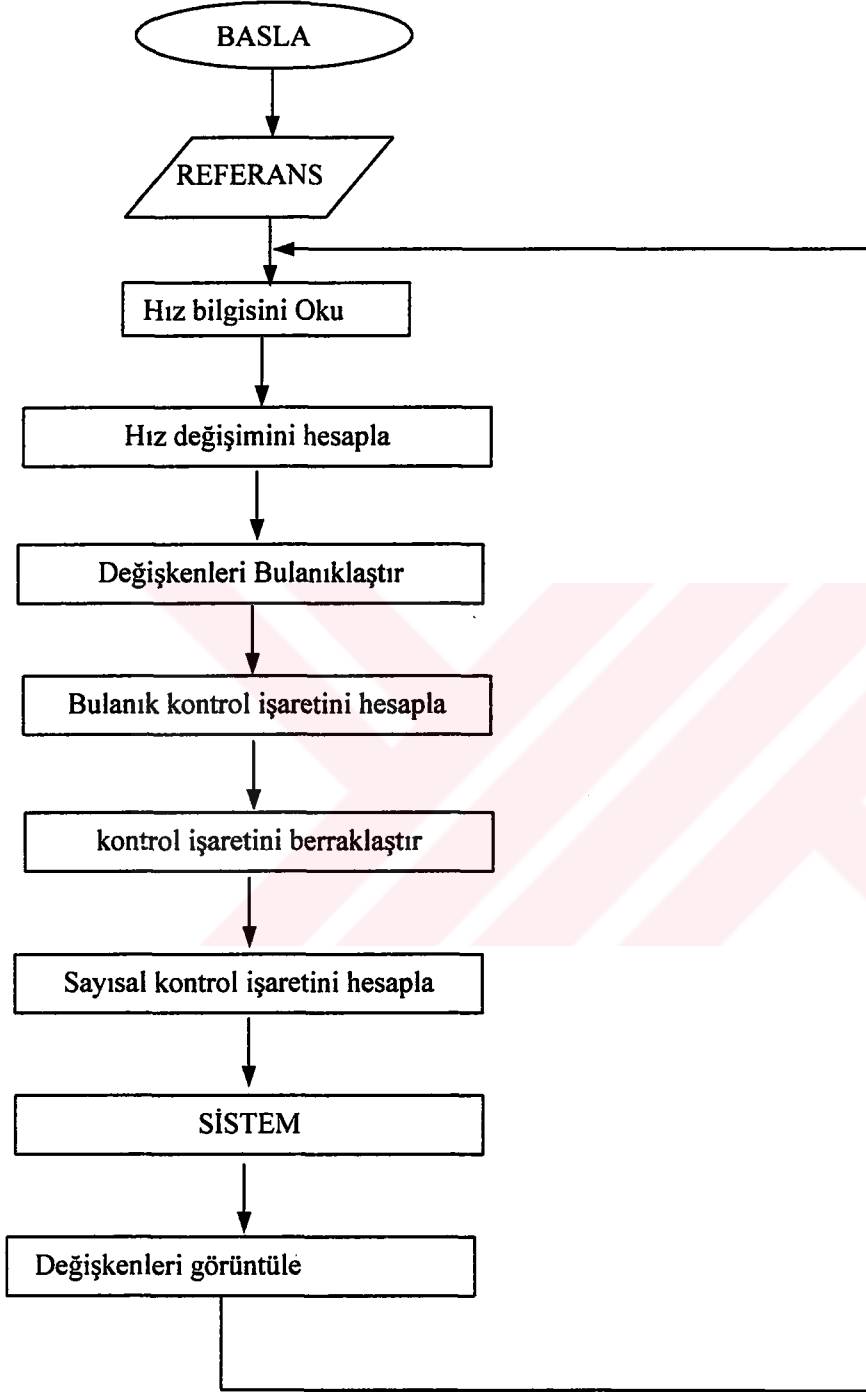
Burada; U, sayısal kontrol işaretini ; n, kural sayısını; u(i) ve DOF(i) sırasıyla , i nolu kuralın kontrol davranışını ve gerçekleştirme oranını göstermektedir.

0-1 arasında elde edilen bulanık kontrolör çıkış işareti, DC Motor sürücünün kontrol girişine uygulanmak üzere, 0-10V. arasında DC gerilime çevrilir. Bulanık kontrolör çıkışının 1 olması demek, DC Motor sürücünün kontrol girişine 10V.'luk gerilimin uygulanması demektir. Bu da DC Motorun en yüksek hızda dönmesi demektir. Bulanık kontrolör çıkışının 0 olması durumda ise DC Motor sürücünün kontrol girişine 0V.'luk gerilimin uygulanması demektir.

5.6 Bilgisayar Yazılımı

Kontrolör yazılımı DELPHI 5.0 programlama dili ile yazılmıştır. DELPHI 5.0 programlama dili görsel niteliklere sahip olan bir dil olduğu için grafiksel görünüm programın alt kısımlarıyla birlikte bir form oluşturmaktadır. Tasarlanan bulanık

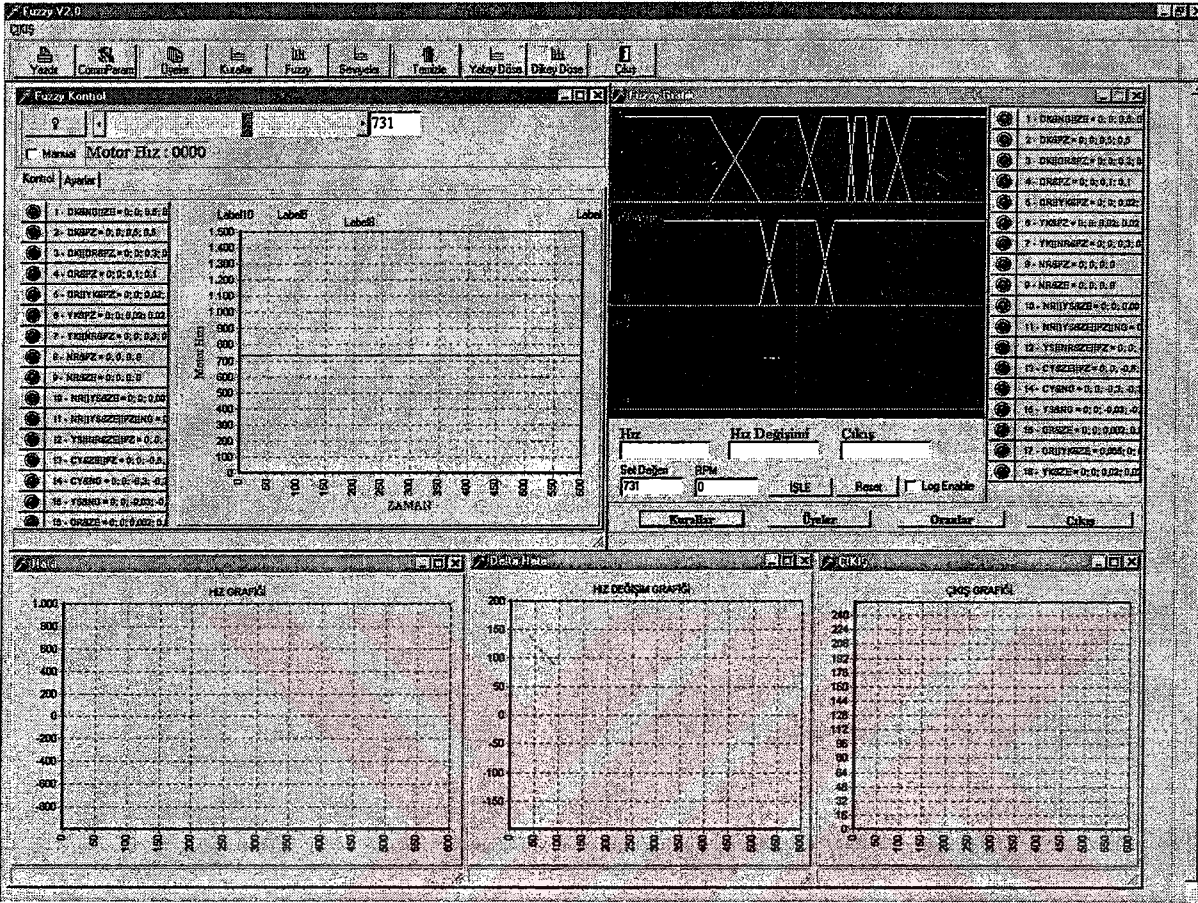
kontrolörün basitleştirilmiş genel akış diyagramı Şekil 5.12'de verilmiştir. Bilgisayar programının genel görüntüsü ise Şekil 5.13'de görülmektedir.



Şekil 5-12 Programın basitleştirilmiş akış diyagramı

Program bulanık mantık'ın anlaşılmasına katkıda bulunmak amacıyla interaktif (etkileşimli) olarak tasarlanmıştır. Referans değeri, üyelik fonksiyonları ve fonksiyon sınırları, kurallar programın kaynak koduna ihtiyaç duymadan değiştirilebilmekte,

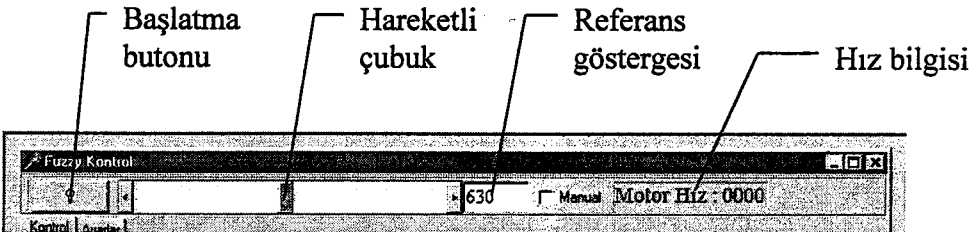
kaydedilebilmekte ve yenileri eklenebilmektedir. Şekil 5-13'te gerçekleştirilen bulanık kontrol programının ekran genel görünümü verilmiştir.



Şekil 5-13 Bulanık kontrolör ekran görünümü

5.6.1 Referans bilgi girişi

Referans hız bilgi girişi şekil 5.14'te görülmektedir.



Şekil 5-14 Referans bilgi girişi

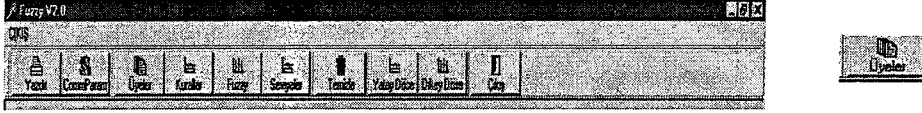
Motorun hangi hızda dönmesi isteniyorsa yatay hareketli çubuk (slide bar) vasıtasıyla istenilen hıza getirilir veya direk olarak referans bilgisinin görüntülediği

referans göstergesine girilerek belirtilir. Çalıştırılmak isteniyorsa başlatma butonuna basılarak motor çalıştırılır.

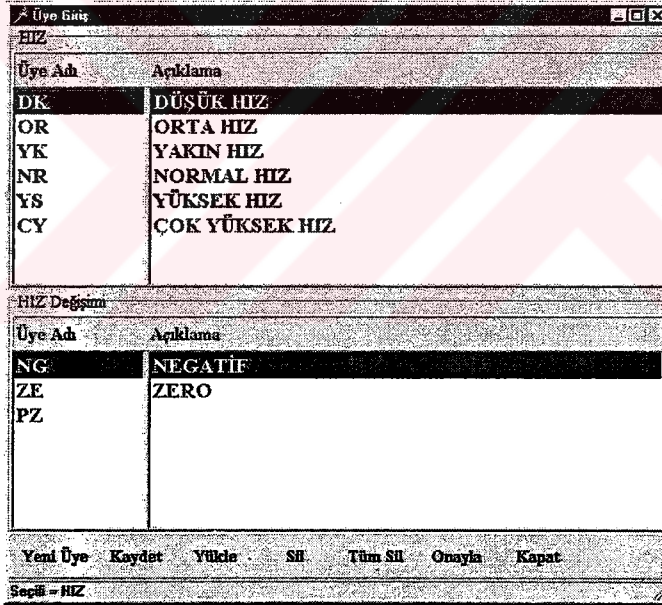
Eğer motor kontrolsüz çalıştırılmak istenirse "Manual" işaretlenir.

5.6.2 Üyelik Fonksiyonları Penceresi

Üyelik fonksiyonları, program üzerinden trapez olarak seçilebilmekte sınır noktaları programla ayarlanabilmektedir. Önce program menüsünden "üyeler" seçilir.



"Üyeler" seçildiğinde Şekil 5.15' deki pencere açılır. Eğer yeni üye seçimi yapılacaksa "yeni üye" ye tıklanır ve üyeler oluşturulur.



Şekil 5-15 Üye penceresi

Oluşturulan bu üyeler alttaki menü yardımıyla istenilen işlem gerçekleştirilir. örneğin "kaydet" menüsü ile daha sonra üyeleri çağırmak amacıyla kaydeder.

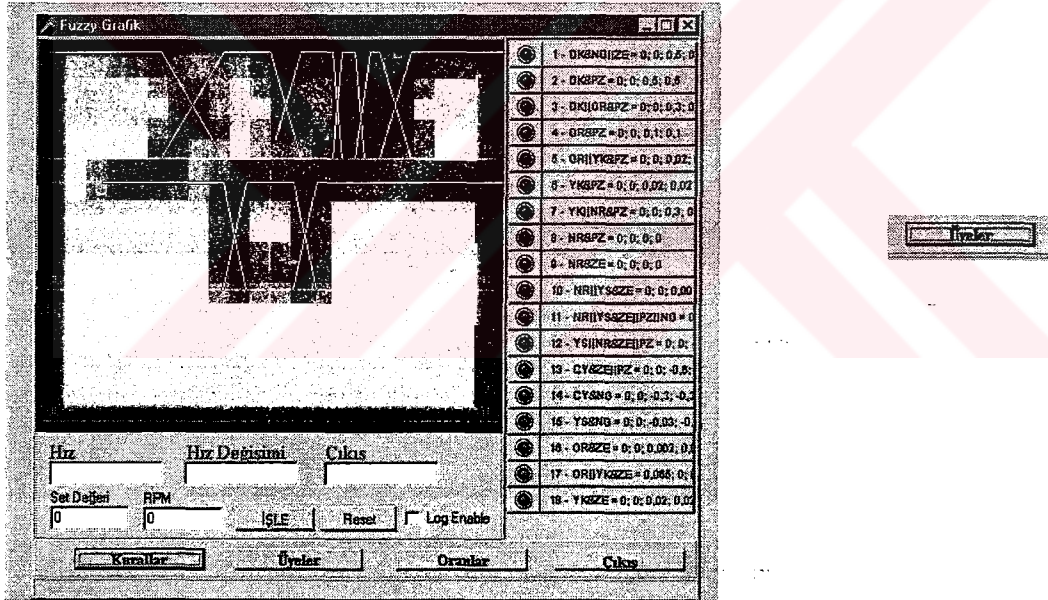
5.6.2.1 Üyelik Fonksiyonlarının Sınır Değerlerinin Belirlenmesi

Üyelik fonksiyonu şekli olarak trapezoid seçilmiştir. Şekil 5.15 yardımıyla oluşturulan Üyelik fonksiyonlarının sınır değerleri önce Şekil 5.16 yardımıyla fuzzy sekmesi işaretlenir.



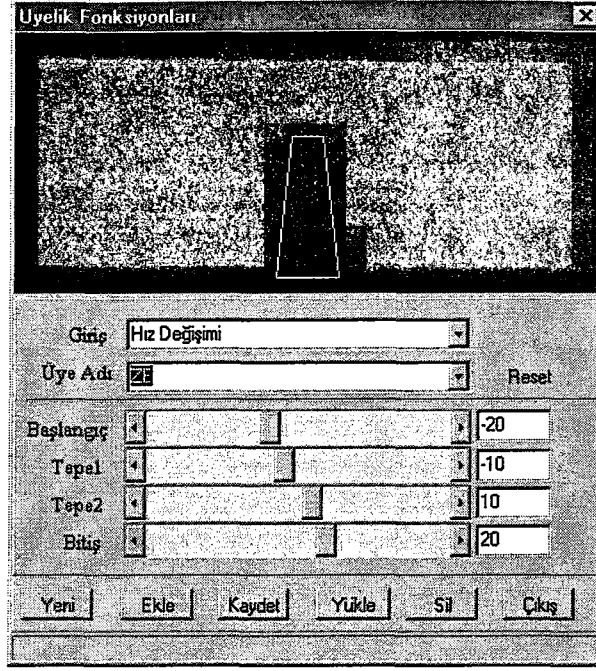
Şekil 5-16 Üyelik fonksiyonlarının seçilmesi

Fuzzy sekmesi işaretlendiğinde Şekil 5.17'deki pencere açılacaktır. Burada hız'a, hız değişimine ve çıkışa ait grafikler görülecektir.



Şekil 5-17 Üyelik fonksiyonlarının oluşturulması

Burada üyeler düğmesi işaretlenirse her üyeye ait sınır değerlerinin belirlenebileceği Şekil 5.18'de görülen pencere açılacaktır. Bu pencerede hız ve hız değişimine ait üyelik fonksiyonlarının sınır değerleri ayarlanabilir.

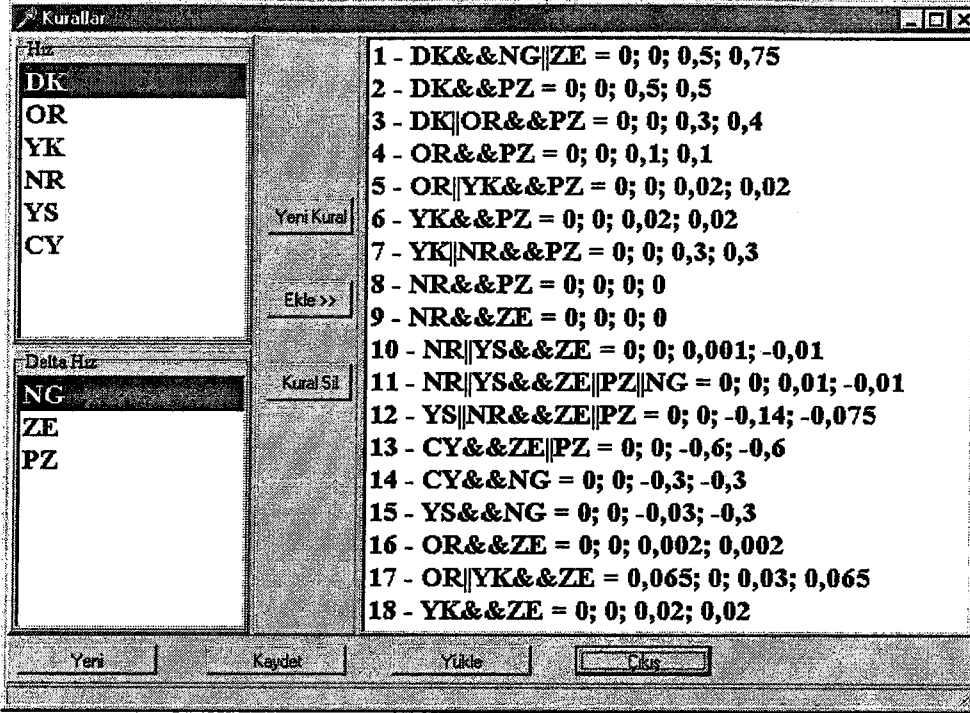


Şekil 5-18 Üyelik fonksiyonlarının sınır değerlerinin ayarlanması

5.7 Kural Tabanı

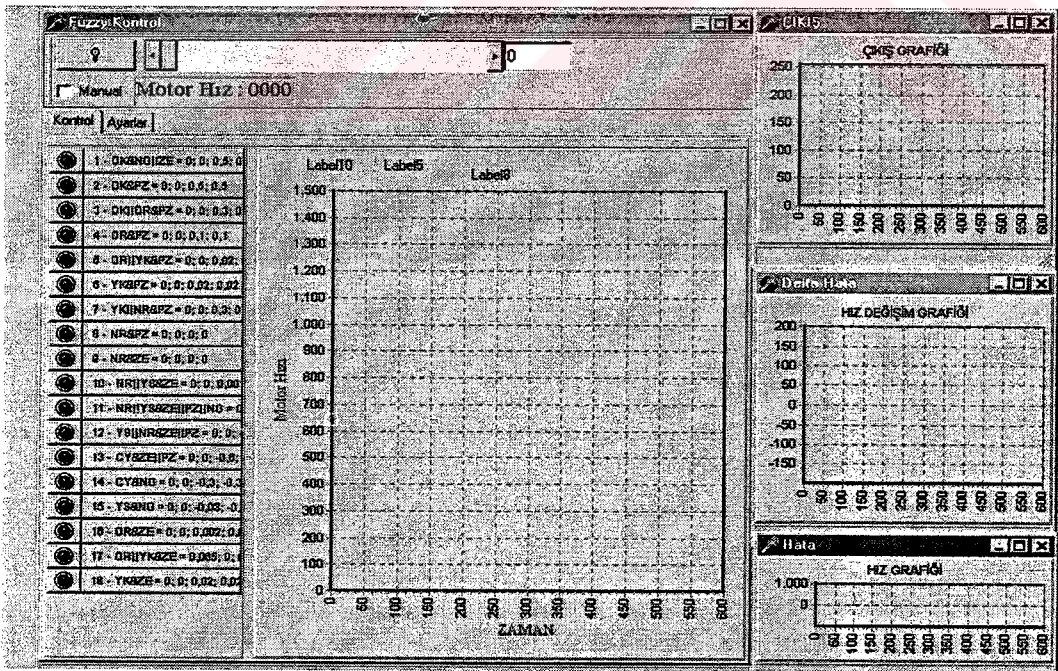
Yapılan çalışma eğitim amaçlı olduğundan kuralların seçimi ve yazımı, üyelik fonksiyonlarının seçimi gibi değişkenlik gerektiren kısımlar kullanıcının üzerinde değişiklik yaparak etkilerinin görülmesi amacıyla esnek tasarlanmıştır.

Kurallar program üzerinden girilebilmekte, değiştirilebilmekte ve yeni kurallar eklenebilmektedir. Kurallar penceresi Şekil 5.19'da görülmektedir.



Şekil 5-19 Kuralların belirlenmesi

Çıkış, Hız, Hız değişimi gibi çıktılar şekil 5.20'deki grafikler yardımıyla gerçek zamanlı olarak izlenebilir.



Şekil 5-20 Grafik çıktıları

Sistem çalışırken aktif olan kuralların ışığı yanmak suretiyle hız görülebilmektedir.

BÖLÜM 6

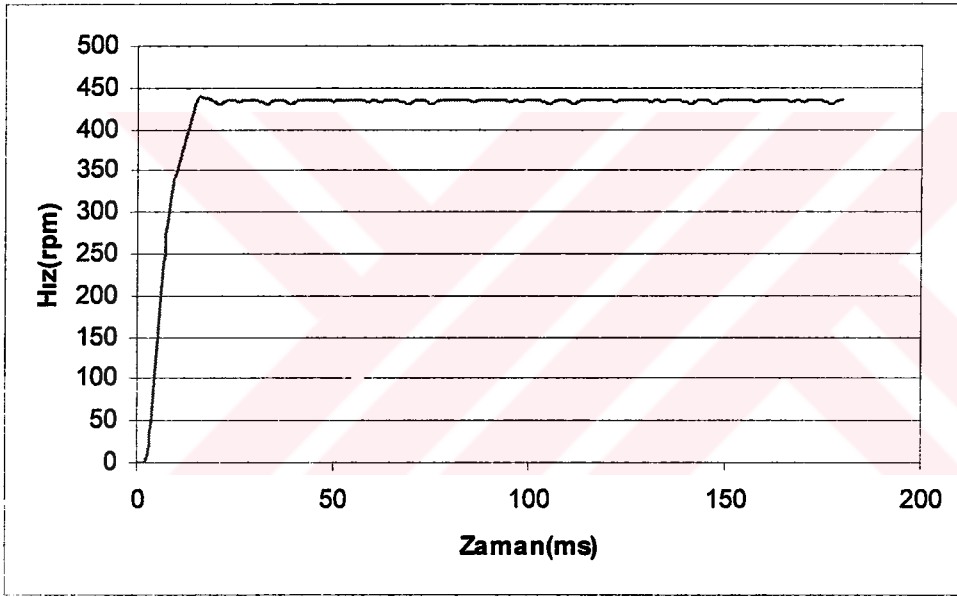
DENEYLER VE SONUÇLAR

6.1 Giriş

Bu bölümde Elektrik Makineleri Eğitim Seti (FH2 MkIV) de bulunan FH-50 DC motor için tasarlanan bulanık kontrolöre ait deney sonuçları anlatılacaktır.

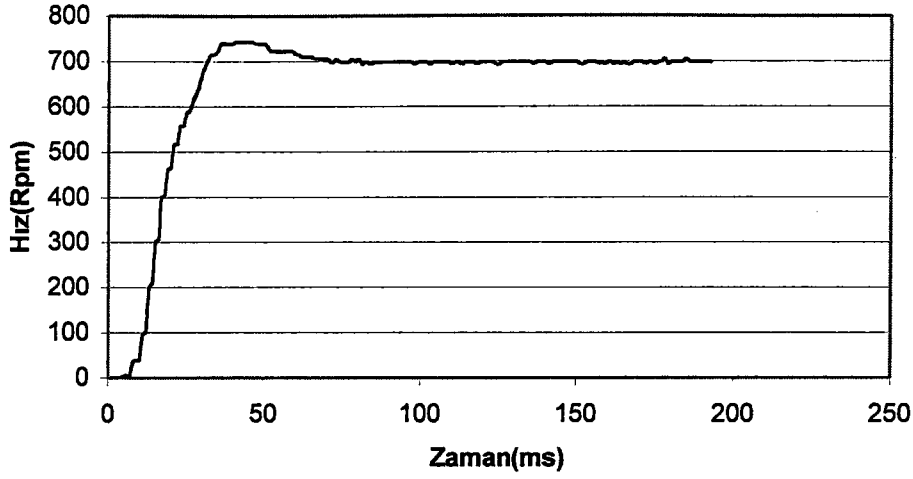
6.2 Gerçekleştirilen Kontrolörün Test Edilmesi

İlk uygulamada hız 0 dan 435 rpm'e çıkarılmış buna ait sistem davranışı Şekil 6-1'de verilmiştir. Bulanık kontrolör motorun hızını 435 rpm'de sabit tutabilmiştir.



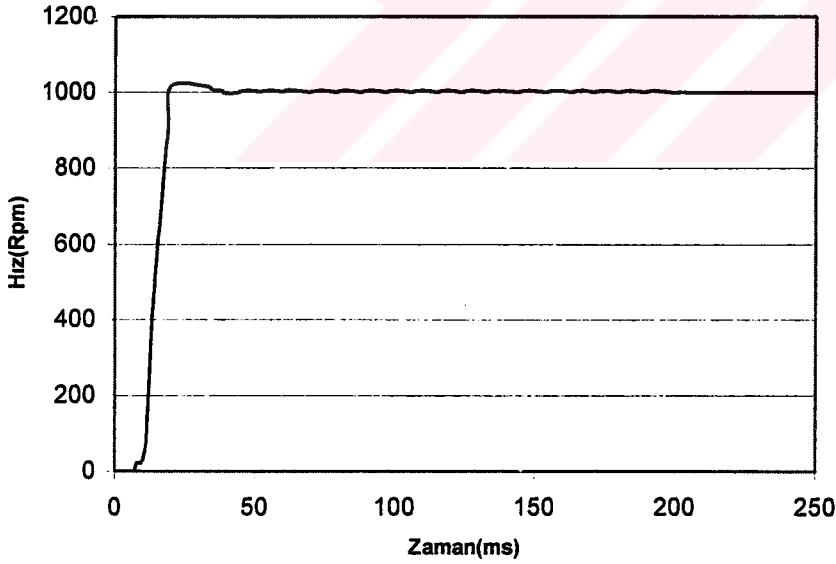
Şekil 6-1 Hızın 435 rpm'e çıkarılması

İkinci uygulamada hız referans değeri olarak 700 rpm seçilmiştir. Buna ait grafik Şekil 6.2'de görülmektedir. Hızın 20 rpm'lik aşım ile 60 ms'de referans değerine ulaştığı görülmüştür.



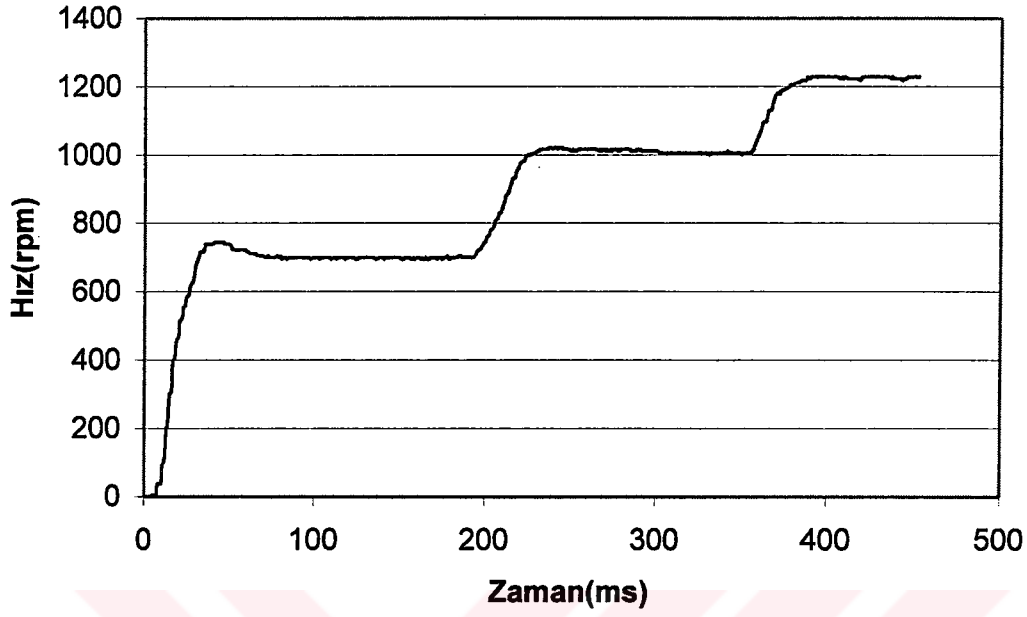
Şekil 6-2 700 rpm'e ait hız grafiği

Üçüncü uygulamada hız referans değeri olarak 1000 rpm seçilmiştir. Buna ait grafik Şekil 6.3'te görülmektedir. Hızın 30 rpm'lik kabul edilebilir bir aşım ile referans değerinde sabit kalmıştır.



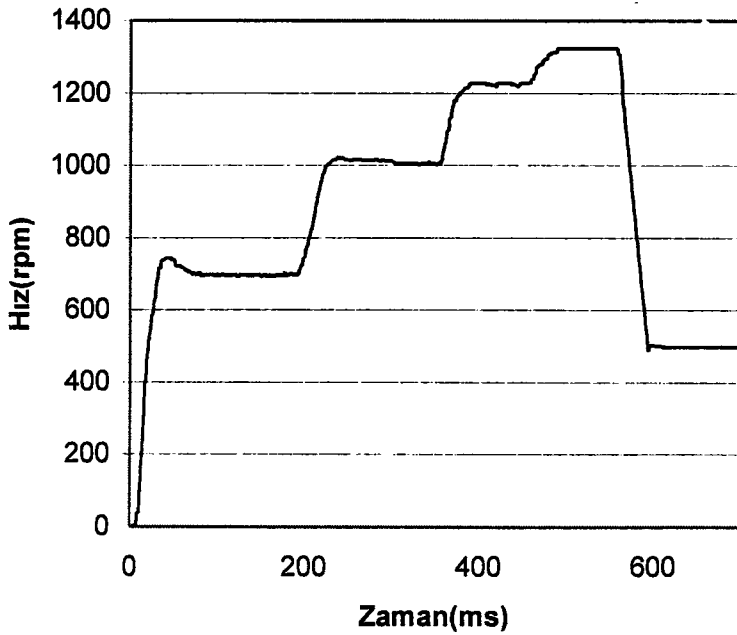
Şekil 6-3 1000 rpm'e ait hız grafiği

Dördüncü uygulamada sistem çalışırken değişik hız seviyelerine geçiş cevabını görebilmek için hız 700 rpm'den 1000 ve 1200 rpm'e çıkarılmış bunlara ait grafikler ise Şekil 6.4'te görülmektedir. Burada da hız değerleri kabul edilebilir aşım miktarları ile referansa ulaştığı görülmektedir.



Şekil 6-4 Motorun 700 rpm, 1000 rpm ve 1200 rpm deki davranışı

Beşinci uygulamada ise hız önce kademeli olarak 700, 1000, 1200 ve 1300 rpm çıkarılmış ve 500 rpm'e indirilmiştir. Şekil 6.5'te de görüleceği gibi referans değerine kabul edilebilir aşım miktarları ile ulaşılmıştır.



Şekil 6-5 Hızın sırasıyla 700, 1000, 1200 ve 1300 rpm çıkarılıp 500 rpm'e indirilmesi

Sonuç olarak , DC motor 400,435,500, 700,1000,1200 ve 1300 rpm referans değerlerinde test edilmiştir. Bulanık kontrolör kabul edilebilir hata aralıklarında sabit tutabildiği görülmüştür. Kontrolör ayrıca sistem çalışmakta iken referans değerleri değiştirerek te test edilmiştir. FH2 MkIV Elektrik Makineleri Eğitim Setinin kapalı çevrim kontrolünde kullanılmak üzere tasarlanan bulanık kontrol, sistemi kontrol edebilmektedir.

6.3 Sonuç ve Öneriler

Bu çalışmada Elektrik Makineleri Eğitim Seti (FH2 MkIV) de bulunan FH-50 DC motor için bulanık kontrolör tasarımı yapılmıştır. Eğitim seti herhangi bir DC motor kontrolör içermemektedir. Kontrol için gerekli verileri almak ve işlemek amacıyla bir mikrodenetleyici kartı tasarımı yapılmıştır. Ayrıca bilgisayarda mikrodenetleyici karttan gelen verileri değerlendirmek ve bulanık mantık ile kontrolörün öğrenilmesine katkıda bulunmak amacıyla bir bulanık kontrolör programı yazılmıştır.

DC motorun bulanık kontrol aşamasında giriş değişkenleri olarak, %hız, hızdaki değişim ve referans değerleri seçilmiştir. Bulanıklaştırma işleminde üyelik fonksiyonlarının sınır değerlerinin tesbiti için çeşitli uygulamalar yapılmıştır. Yapılan bu uygulamalarda DC motoru kontrol etmek için kullanılan sürücüyü çeşitli gerilim değerleri gönderilerek motorun çeşitli hızlarda dönme davranışı incelenmiştir. Daha sonra hızların sürücüyü 0 volt verilerek "0" a düşmeleri incelenmiştir. Yapılan incelemeler sonunda hızların lineere yakın olduğu görülmüştür.

Kontrol aşamasında hız bilgisinin okunması sırasında dış ortamdan gelen bozucu etkilerden dolayı bazı hatalar olduğu görülmüştür. Bu istenmeyen hataları düzeltmek için bir sayısal filtre kullanılmıştır.

Öneriler ;

Dc motor'a daha hassas bir tako ilave edilerek hız bilgilerinde karşılaşılan hatalar azaltılabilir. Elektrik Makineleri Eğitim Seti (FH2 MkIV) için bir sürücü devre tasarlanarak sete ilave edilebilir. Yazılan PC programı geliştirilmeye açık tasarlanmıştır. Programa üyelik fonksiyon şekilleri ve çıkarım metodları eklenerek geliştirilebilir.

KAYNAKLAR

- [1] Lotfi A.ZADEH, "Fuzzy Set",
Information and Control, vol.8.pp.338-353, 1965
- [2] Chuen Chien Lee, "Fuzzy Logic In Control Systems: Fuzzy Logic Controller,
Part1", IEEE Transactions on System Man. And Cybernatic, vol.20, No.2, March/April
1990
- [3] Bellman, R.E. and Zadeh, L.A., "Decision Making in a Fuzzy environment,
management Sci", 1970 Vol.17.pp.141-164
- [4] Cansever, G., Özgüven, Ö.F., Uzam, M., 1993b. "Fuzzy Logic Controller in Modern
Controller System. An industrial Applications", International AMSE conference
Systems, London, England, vol2 pp.45-67
- [5] Brae, M., and Rutherford, D.A, 1979. "Theoretical and Linguistic Aspects of the
Fuzzy Logic Controller", Automatica, vol.15.pp.553-557.
- [6] Timothy J.Ross, "Fuzzy Logic and Control", Prentice Hall, 1996 New jersey
- [7] Riza C. Berkan, Sheldon L. Trubatch, "Fuzzy System Design Principles", IEEE
Press USA-1997
- [8] BABA, A.F. "İTÜ Triga Mark II Reaktörünün Bulanık Kontrolü", Doktora Tezi
Marmara Üniv. Fen Bil. Enstitüsü. 1995 İstanbul
- [9] H.J. Zimmerman, "Fuzzy Set Theory And its Applications, second edition", Kluwer
Academic Publishers, Boston, London
- [10] Mizumoto, M., "Fuzzy Controls Under Various Fuzzy Reasoning Methodes",
Information Sciences, No.45, (1988), 129-151
- [11] Elektrik Makineleri Eğitim Seti (FH2/3 MkIV), user Manuel
- [12] AVR RISC Microcontroller data book, August 1999, pp.7-3

EKLER

EK-1 PC PROGRAM LİSTESİ

Bu ekte gerçekleştirilen bulanık kontrolörün Delphi 5.0 programlama dilinde yazılan program listeleri verilmektedir.

```
program FuzzMot;
```

```
uses
```

```
Forms,
```

```
FuzKont in 'FuzKont.pas' {FormFuzKont},
```

```
Ana in 'Ana.pas' {FormAna},
```

```
ComPar in 'ComPar.pas' {FormSerPar},
```

```
CrcBuf in 'CrcBuf.pas',
```

```
Sicil_ in 'Sicil_.pas',
```

```
Fuzz in 'Fuzz.pas' {FormFuzzy},
```

```
Trap in 'Trap.pas' {FormUye},
```

```
MemEnt in 'MemEnt.pas' {FormMemEnt},
```

```
EntUye in 'EntUye.pas' {FormMem},
```

```
Util in 'Util.pas',
```

```
Kural in 'Kural.pas' {FormKural},
```

```
HataGrp in 'HataGrp.pas' {FormHata},
```

```
Oran in 'Oran.pas' {FormOran},
```

```
DHataGrp in 'DHataGrp.pas' {FormDHata},
```

```
CikisGrp in 'CikisGrp.pas' {FormCikis},
```

```
Member in 'Member.pas',
```

```
yazdir in 'yazdir.pas' {FormYazdir},
```

```
Unit1 in 'Unit1.pas' {Form1},
```

```
Log in 'Log.pas' {FormLog},
```

```
Uyeent in 'C:\Program Files\Common Files\Borland
```

```
Shared\Images\Buttons\Uyeent.pas' {FormUyeEnt},
```

```
OutVal in 'OutVal.pas' {FormOutVal},
```

```
CikDeg in 'CikDeg.pas' {FormDegDeg},
```

KurDuz in 'KurDuz.pas' {FormKurDuz},
LEvel in 'LEvel.pas' {FormLevels};

{\$R *.RES}

begin

```
Application.Initialize;  
Application.CreateForm(TFormAna, FormAna);  
Application.CreateForm(TFormFuzKont, FormFuzKont);  
Application.CreateForm(TFormUye, FormUye);  
Application.CreateForm(TFormKural, FormKural);  
Application.CreateForm(TFormMemEnt, FormMemEnt);  
Application.CreateForm(TFormFuzzy, FormFuzzy);  
Application.CreateForm(TFormSerPar, FormSerPar);  
Application.CreateForm(TFormMem, FormMem);  
Application.CreateForm(TFormHata, FormHata);  
Application.CreateForm(TFormOran, FormOran);  
Application.CreateForm(TFormDHata, FormDHata);  
Application.CreateForm(TFormCikis, FormCikis);  
Application.CreateForm(TFormYazdir, FormYazdir);  
Application.CreateForm(TForm1, Form1);  
Application.CreateForm(TFormLog, FormLog);  
Application.CreateForm(TFormUyeEnt, FormUyeEnt);  
Application.CreateForm(TFormOutVal, FormOutVal);  
Application.CreateForm(TFormDegDeg, FormDegDeg);  
Application.CreateForm(TFormLevels, FormLevels);  
Application.Run;
```

end.

unit Ana;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
Menus, ComCtrls, ToolWin, ImgList;

type

```
TFormAna = class(TForm)
  MainMenu1: TMainMenu;
  IKI1: TMenuItem;
  CoolBar1: TCoolBar;
  ToolBar1: TToolBar;
  ToolButton1: TToolButton;
  ToolButton2: TToolButton;
  ToolButton3: TToolButton;
  ImageList1: TImageList;
  StatusBar1: TStatusBar;
  SaveDialog: TSaveDialog;
  ToolButton4: TToolButton;
  ToolButton5: TToolButton;
  ToolButton6: TToolButton;
  ToolButton10: TToolButton;
  ToolButton8: TToolButton;
  ToolButton7: TToolButton;
  ToolButton9: TToolButton;
  ToolButton11: TToolButton;
  ToolButton12: TToolButton;
  ToolButton13: TToolButton;
  ToolButton14: TToolButton;
  procedure IKI1Click(Sender: TObject);
  procedure ToolButton2Click(Sender: TObject);
  procedure ToolButton3Click(Sender: TObject);
  procedure VeriKaydet(fname : string);
  procedure ToolButton1Click(Sender: TObject);
  procedure ToolButton4Click(Sender: TObject);
  procedure FormClose(Sender: TObject; var Action: TCloseAction);
  procedure ToolButton10Click(Sender: TObject);
```

```

procedure ToolButton6Click(Sender: TObject);
procedure ToolButton9Click(Sender: TObject);
procedure ToolButton12Click(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure ToolButton13Click(Sender: TObject);
procedure ToolButton8Click(Sender: TObject);
procedure ToolButton14Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

var
  FormAna: TFormAna;
  f      : Text;

implementation

uses ComPar, FuzKont, Sicil_, MemEnt, yazdir, Unit1, Uyeent, Kural, Fuzz,
  LLevel;

{$R *.DFM}

procedure TFormAna.Verikaydet(fname : string);
var
  i : integer;
  S : string;
begin
  AssignFile(f, fname);
  ReWrite(f);
  for i:= 0 to FormfuzKont.Series1.Count - 1 do
  begin

```

```

        S := FloatToStr(FormfuzKont.Series1.XValue[i]+1) + ' ' +
FloatToStr(FormfuzKont.Series1.YValue[i]);
        WriteLn(f, S);
    end;
    CloseFile(f);
end;

```

```

procedure TFormAna.IK1Click(Sender: TObject);
begin
    Close;
end;

```

```

procedure TFormAna.ToolButton2Click(Sender: TObject);
begin
    FormSerPar.ShowModal;
end;

```

```

procedure TFormAna.ToolButton3Click(Sender: TObject);
begin
    if FormFuzzy.WindowState = wsNormal then
        FormFuzzy.WindowState := wsMinimized
    else
        if FormFuzzy.WindowState = wsMinimized then
            FormFuzzy.WindowState := wsNormal;
end;

```

```

procedure TFormAna.ToolButton1Click(Sender: TObject);
begin
    FormYazdir.Show;
end;

```

```

procedure TFormAna.ToolButton4Click(Sender: TObject);
begin
    // FormfuzKont.ListBox1.Clear;

```

```
FormfuzKont.Series1.Clear;  
FormfuzKont.ResetCnt;  
end;
```

```
procedure TFormAna.FormClose(Sender: TObject; var Action: TCloseAction);  
begin  
    RegUpdate;  
end;
```

```
procedure TFormAna.ToolButton10Click(Sender: TObject);  
begin  
    Close;  
end;
```

```
procedure TFormAna.ToolButton6Click(Sender: TObject);  
begin  
    // FormUyeEnt.ShowModal;  
    FormMemEnt.ShowModal;  
end;
```

```
procedure TFormAna.ToolButton9Click(Sender: TObject);  
begin  
    FormAna.TileMode:= tbHorizontal;  
    Tile;  
end;
```

```
procedure TFormAna.ToolButton12Click(Sender: TObject);  
begin  
    FormAna.TileMode:= tbVertical;  
    Tile;  
end;
```

```
procedure TFormAna.FormShow(Sender: TObject);  
begin
```

```

    FormFuzKont.CommPortDriver.Connect;
    FormFuzKont.Show;
end;

procedure TFormAna.ToolButton13Click(Sender: TObject);
begin
    Form1.ShowModal;
end;

procedure TFormAna.ToolButton8Click(Sender: TObject);
begin
    FormKural.ShowModal;
    FormFuzzy.KuralButon;
    FormFuzzy.KuralButon2;
    FormFuzzy.DrawMembers;
end;

procedure TFormAna.ToolButton14Click(Sender: TObject);
begin
    FormLevels.ShowModal;
end;

end.

unit KurDuz;

interface

uses
    Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
    StdCtrls, ExtCtrls, ComCtrls, Buttons;

type
    TFormKurDuz = class(TForm)

```

```

StatusBar1: TStatusBar;
Panel1: TPanel;
Label1: TLabel;
Panel2: TPanel;
SpeedButton1: TSpeedButton;
SpeedButton2: TSpeedButton;
StaticText1: TStaticText;
Bevel1: TBevel;
Bevel2: TBevel;
SpeedButtonYeniHiz: TSpeedButton;
SpeedButtonYeniHizDeg: TSpeedButton;
SpeedButtonUyeSil: TSpeedButton;
Bevel3: TBevel;
procedure SpeedButton1Click(Sender: TObject);
procedure ComboCreate1(index : integer);
procedure ComboCreate2(index : integer);
procedure ChangeFormula(f : string; KuralNo : integer);
procedure EditCreate(index : integer);
procedure LabelCreate(s : string);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure SpeedButtonYeniHizClick(Sender: TObject);
procedure FreeControls;
procedure SpeedButtonYeniHizDegClick(Sender: TObject);
procedure SpeedButtonUyeSilClick(Sender: TObject);
procedure CalcKural;
procedure SpeedButton2Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

const
  TopL = 58;

```

var

FormKurDuz: TFormKurDuz;

CB : TComboBox;

EB : TEdit;

SB : TLabel;

Comp : integer = 0;

Coord : integer;

KN : integer;

Ss : string;

implementation

uses Kural, Member;

{\$R *.DFM}

procedure TFormKurDuz.ComboCreate1(index : integer);

begin

 CB := TComboBox.Create(nil);

 CB.Parent := Panel1;

 CB.Width := 48;

 CB.Left := Coord;

 CB.Top := TopL;

 CB.Tag := 1;

 CB.Items := FormKural.ListBox1.Items;

 CB.ItemIndex := Index;

end;

procedure TFormKurDuz.ComboCreate2(index : integer);

begin

 CB := TComboBox.Create(nil);

 CB.Parent := Panel1;

 CB.Width := 48;

```

    CB.Left := Coord;
    CB.Top := TopL;
    CB.Tag := 2;
    CB.Items := FormKural.ListBox3.Items;
    CB.ItemIndex := Index;
end;

procedure TFormKurDuz.EditCreate(index : integer);
var
    i : integer;
begin
    for i := 0 to 3 do
        begin
            EB := TEdit.Create(nil);
            EB.Parent := Panel1;
            EB.Width := 48;
            EB.Left := Coord;
            EB.Top := TopL;
            EB.Tag := 3 + i;
            case i of
                0 : EB.Text := FloatToStr(KuralSabit1[Index]);
                1 : EB.Text := FloatToStr(KuralSabit2[Index]);
                2 : EB.Text := FloatToStr(KuralSabit3[Index]);
                3 : EB.Text := FloatToStr(KuralSabit4[Index]);
            end;
            Coord := Coord + 50;
        end;
    end;

procedure TFormKurDuz.LabelCreate(s : string);
begin
    SB := TLabel.Create(nil);
    SB.Parent := Panel1;
    SB.Left := Coord;

```

```

SB.Font.Size := 10;
SB.Font.Style := [fsBold];
SB.Top := TopL;
SB.Tag := 7;
SB.Caption := s;
SB.AutoSize := True;
SB.ShowAccelChar := false;
end;

procedure TFormKurDuz.ChangeFormula(f : string; KuralNo : integer);
var
  L : integer;
  i : integer;
  j : integer;
  KB : integer;
begin
  StaticText1.Caption := f;

  Ss := f;
  i := 0;
  Comp := 0;
  coord := 8;
  while KuralArray[KuralNo].Hata[i] <> -1 do
  begin
    ComboCreate1 (KuralArray[KuralNo].hata[i]);
    i := i + 1;
    Comp := Comp + 1;
    Coord := Coord + 50;
  end;

  i := 0;
  KN := KuralNo;
  LabelCreate('&&');
  Comp := Comp + 1;

```

```

Coord := Coord + 28;
while KuralArray[KuralNo].DeltaHata[i] <> -1 do
begin
  ComboCreate2(KuralArray[KuralNo].Deltahata[i]);
  i := i + 1;
  Comp := Comp + 1;
  Coord := Coord + 50;
end;
LabelCreate('=');
Coord := Coord + 10;
Comp := Comp + 1;
EditCreate(KuralNo);
end;

```

```

procedure TFormKurDuz.SpeedButton1Click(Sender: TObject);
begin
  CalcKural;
  close;
end;

```

```

procedure TFormKurDuz.CalcKural;
var
  i : integer;
  x, y : integer;
begin
  x := 0;
  y := 0;
  for i := 0 to Panel1.ControlCount - 1 do
  begin
    case Panel1.Controls[i].Tag of
      1 :begin
          if TComboBox(Panel1.Controls[i]).ItemIndex >= 0 then
            begin
              KuralArray[KN].hata[x] := TComboBox(Panel1.Controls[i]).ItemIndex;
            end;
        end;
    end;
  end;
end;

```

```

    X := x + 1;
end;
end;
2 :begin
  if TComboBox(Panel1.Controls[i]).ItemIndex >= 0 then
  begin
    KuralArray[KN].Deltahata[Y] :=
TComboBox(Panel1.Controls[i]).ItemIndex;
    Y := Y + 1;
  end;
end;
3 :begin
  try
    KuralSabit1[Kn] := StrToFloat(TEdit(Panel1.Controls[i]).Text);
  except
  end;
end;
4 :begin
  try
    KuralSabit2[Kn] := StrToFloat(TEdit(Panel1.Controls[i]).Text);
  except
  end;
end;
5 :begin
  try
    KuralSabit3[Kn] := StrToFloat(TEdit(Panel1.Controls[i]).Text);
  except
  end;
end;
6 :begin
  try
    KuralSabit4[Kn] := StrToFloat(TEdit(Panel1.Controls[i]).Text);
  except
  end;
end;

```

```
end;
```

```
end;
```

```
end;
```

```
KuralArray[KN].hata[x] := -1;
```

```
KuralArray[KN].Deltahata[y] := -1;
```

```
end;
```

```
procedure TFormKurDuz.FormClose(Sender: TObject; var Action:  
TCloseAction);
```

```
begin
```

```
Release;
```

```
end;
```

```
procedure TFormKurDuz.FreeControls;
```

```
var
```

```
  i : integer;
```

```
begin
```

```
  i := 0;
```

```
  while i < Panel1.ControlCount do
```

```
  begin
```

```
    if Panel1.Controls[i].Tag > 0 then
```

```
    begin
```

```
      Panel1.Controls[i].Free;
```

```
      i := i + 1;
```

```
    end;
```

```
  end;
```

```
end; //for i
```

```
end;
```

```
procedure TFormKurDuz.SpeedButtonYeniHizClick(Sender: TObject);
```

```
var
```

```

i : integer;
begin
  CalcKural;
  FreeControls;
  for i := 0 to 19 do
  begin
    if KuralArray[KN].hata[i] = -1 then
    begin
      KuralArray[KN].hata[i] := 0;
      KuralArray[KN].hata[i+1] := -1;
      Break;
    end;
  end;//for i
  ChangeFormula(Ss, KN);
end;

procedure TFormKurDuz.SpeedButtonYeniHizDegClick(Sender: TObject);
var
  i : integer;
begin
  CalcKural;
  FreeControls;
  for i := 0 to 19 do
  begin
    if KuralArray[KN].Deltahata[i] = -1 then
    begin
      KuralArray[KN].Deltahata[i] := 0;
      KuralArray[KN].Deltahata[i+1] := -1;
      Break;
    end;
  end;//for i
  ChangeFormula(Ss, KN);
end;

```

```

procedure TFormKurDuz.SpeedButtonUyeSilClick(Sender: TObject);
var
  i : integer;
begin
  for i := 0 to Panel1.ControlCount - 1 do
  begin
    if Panel1.Controls[i] is TComboBox then
    begin
      if TComboBox(Panel1.Controls[i]).Focused then
      begin
        Panel1.Controls[i].Free;
        Break;
      end;
    end;
  end;
  CalcKural;
  FreeControls;
  ChangeFormula(Ss, KN);
end;

```

```

procedure TFormKurDuz.SpeedButton2Click(Sender: TObject);
begin
  Close;
end;

```

end.

unit LLevel;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
 StdCtrls, Buttons, ExtCtrls, ComCtrls;

type

```
TFormLevels = class(TForm)
  StatusBar1: TStatusBar;
  Panel1: TPanel;
  BitBtnOnayla: TBitBtn;
  BitBtnVazgec: TBitBtn;
  GroupBox1: TGroupBox;
  Label1: TLabel;
  Label2: TLabel;
  Label3: TLabel;
  Label4: TLabel;
  EditCokKucuk: TEdit;
  EditOrta: TEdit;
  EditKucuk: TEdit;
  EditBuyuk: TEdit;
  procedure BitBtnVazgecClick(Sender: TObject);
  procedure BitBtnOnaylaClick(Sender: TObject);
  procedure Bosalt;
  procedure FormShow(Sender: TObject);
  procedure DegerAl;
  procedure FormKeyPress(Sender: TObject; var Key: Char);
private
  { Private declarations }
public
  { Public declarations }
end;
```

var

```
FormLevels: TFormLevels;
```

implementation

uses member;

```
{ $R *.DFM }
```

```
procedure TFormLevels.BitBtnVazgecClick(Sender: TObject);
begin
    Close;
end;
```

```
procedure TFormLevels.Bosalt;
begin
    EditCokKucuk.Text := "";
    EditKucuk.Text := "";
    EditOrta.Text := "";
    EditBuyuk.Text := "";
end;
```

```
procedure TFormLevels.BitBtnOnaylaClick(Sender: TObject);
begin
    try
        CikisLevels[0] := StrToInt(EditCokKucuk.Text);
        CikisLevels[1] := StrToInt(EditKucuk.Text);
        CikisLevels[2] := StrToInt(EditOrta.Text);
        CikisLevels[3] := StrToInt(EditBuyuk.Text);
        Close;
    except
        Bosalt;
    end;
```

```
end;
```

```
procedure TFormLevels.DegerAl;
begin
    EditCokKucuk.Text := IntToStr(CikisLevels[0]);
    EditKucuk.Text := IntToStr(CikisLevels[1]);
    EditOrta.Text := IntToStr(CikisLevels[2]);
    EditBuyuk.Text := IntToStr(CikisLevels[3]);
end;
```

```
procedure TFormLevels.FormShow(Sender: TObject);
begin
  DegerAl;
  EditCokKucuk.SetFocus;
end;
```

```
procedure TFormLevels.FormKeyPress(Sender: TObject; var Key: Char);
begin
  if Key = #13 then
  begin
    Key := #0;
    Perform(WM_NEXTDLGCTL, 0, 0);
  end;
end;

end.
```

```
unit EntUye;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls, ExtCtrls, Buttons;
```

```
type
```

```
TFormMem = class(TForm)
```

```
  Panel1: TPanel;
```

```
  Panel2: TPanel;
```

```
  Edit1: TEdit;
```

```
  Edit2: TEdit;
```

```
  StaticText1: TStaticText;
```

```
  StaticText2: TStaticText;
```

```
SpeedButton1: TSpeedButton;  
SpeedButton2: TSpeedButton;  
procedure SpeedButton2Click(Sender: TObject);  
procedure SpeedButton1Click(Sender: TObject);  
procedure FormKeyPress(Sender: TObject; var Key: Char);  
procedure FormActivate(Sender: TObject);  
private  
  { Private declarations }  
public  
  { Public declarations }  
end;
```

```
var
```

```
  FormMem: TFormMem;
```

```
implementation
```

```
uses MemEnt;
```

```
{$R *.DFM}
```

```
procedure TFormMem.SpeedButton2Click(Sender: TObject);
```

```
begin
```

```
  Close;
```

```
end;
```

```
procedure TFormMem.SpeedButton1Click(Sender: TObject);
```

```
begin
```

```
  if Edit1.Text = '' then
```

```
  begin
```

```
    Application.MessageBox('Üye Adı Boş Olamaz', 'Uyarı', mb_OK);
```

```

    Exit;
end;
if selected= 0 then
begin
    FormMemEnt.ListBoxMem.Items.Add(Edit1.Text);
    FormMemEnt.ListBoxMemDet.Items.Add(Edit2.Text);
end else
begin
    FormMemEnt.ListBox1.Items.Add(Edit1.Text);
    FormMemEnt.ListBox2.Items.Add(Edit2.Text);
end;
Close;
end;

```

```

procedure TFormMem.FormKeyPress(Sender: TObject; var Key: Char);
begin
    if Key = #13 then
    begin
        Perform(WM_NEXTDLGCTL, 0, 0);
        Key := #0;
    end;
end;

```

```

procedure TFormMem.FormActivate(Sender: TObject);
begin
    Edit1.Text := "";
    Edit2.Text := "";
    Edit1.SetFocus;
end;

```

end.

unit MemEnt;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
ExtCtrls, StdCtrls, ComCtrls, Buttons;

type

```
TFormMemEnt = class(TForm)
  StatusBar1: TStatusBar;
  GroupBox1: TGroupBox;
  Panel2: TPanel;
  Splitter1: TSplitter;
  ListBoxMem: TListBox;
  ListBoxMemDet: TListBox;
  Panel3: TPanel;
  StaticText1: TStaticText;
  StaticText2: TStaticText;
  OpenFileDialog1: TOpenDialog;
  SaveDialog1: TSaveDialog;
  OpenFileDialog2: TOpenDialog;
  SaveDialog2: TSaveDialog;
  Panel1: TPanel;
  SpeedButton8: TSpeedButton;
  SpeedButton9: TSpeedButton;
  SpeedButton10: TSpeedButton;
  SpeedButton11: TSpeedButton;
  SpeedButton12: TSpeedButton;
  SpeedButton13: TSpeedButton;
  SpeedButton14: TSpeedButton;
  GroupBox2: TGroupBox;
  Panel4: TPanel;
  Splitter2: TSplitter;
  ListBox1: TListBox;
  ListBox2: TListBox;
  Panel5: TPanel;
```

```

StaticText3: TStaticText;
StaticText4: TStaticText;
Splitter3: TSplitter;
procedure SpeedButton1Click(Sender: TObject);
procedure ListBoxMemDetClick(Sender: TObject);
procedure ListBoxMemClick(Sender: TObject);
procedure SpeedButton2Click(Sender: TObject);
procedure SpeedButton3Click(Sender: TObject);
procedure SpeedButton4Click(Sender: TObject);
procedure SpeedButton6Click(Sender: TObject);
procedure SpeedButton5Click(Sender: TObject);
procedure SpeedButton7Click(Sender: TObject);
procedure ListeyeEkle;
procedure Yukle(s : string);
procedure FormCreate(Sender: TObject);
procedure ListBox1Click(Sender: TObject);
procedure ListBox2Click(Sender: TObject);
procedure WriteSelection;
private
  { Private declarations }
public
  { Public declarations }
end;

var
  FormMemEnt : TFormMemEnt;
  FMem      : text;
  Selected   : integer = 0;

implementation

uses EntUye, Util, Trap, Kural;

{$R *.DFM}

```

```
procedure TFormMemEnt.SpeedButton1Click(Sender: TObject);
begin
    FormMem.ShowModal;
end;
```

```
procedure TFormMemEnt.WriteSelection;
begin
    if Selected = 0 then
        StatusBar1.SimpleText := 'Seçili = HIZ'
    else
        StatusBar1.SimpleText := 'Seçili = HIZ Değişimi'
end;
```

```
procedure TFormMemEnt.ListBoxMemDetClick(Sender: TObject);
begin
    ListBoxMem.ItemIndex := ListBoxMemDet.ItemIndex;
    Selected := 0;
    WriteSelection;
end;
```

```
procedure TFormMemEnt.ListBoxMemClick(Sender: TObject);
begin
    // Label1.Caption := IntToStr(ListBoxMem.ItemIndex);
    ListBoxMemDet.ItemIndex := ListBoxMem.ItemIndex;
    Selected := 0;
    WriteSelection;
end;
```

```
procedure TFormMemEnt.SpeedButton2Click(Sender: TObject);
begin
    if Selected = 0 then
        begin
            ListBoxMemDet.Clear;
```

```

    ListBoxMem.Clear;
end;

if Selected = 1 then
begin
    ListBox1.Clear;
    ListBox2.Clear;
end;
end;

procedure TFormMemEnt.SpeedButton3Click(Sender: TObject);
begin
    Close;
end;

procedure TFormMemEnt.SpeedButton4Click(Sender: TObject);
begin
    if Selected = 0 then
    begin
        ListBoxMemDet.Items.delete(ListBoxMem.ItemIndex);
        ListBoxMem.Items.delete(ListBoxMem.ItemIndex);
    end;

    if Selected = 1 then
    begin
        ListBox1.Items.delete(ListBoxMem.ItemIndex);
        ListBox2.Items.delete(ListBoxMem.ItemIndex);
    end;

end;

procedure TFormMemEnt.Yukle(s : string);
var
    n, i : integer;

```

```

begin
  AssignFile(FMem, s);
  Reset(FMem);

  ListBoxMemDet.Clear;
  ListBoxMem.Clear;

  ListBoxMemDet.Clear;
  ListBoxMem.Clear;

  ReadLn(FMem, s);
  n := StrToInt(s);
  for i := 0 to n - 1 do
  begin
    ReadLn(FMem, s);
    ListBoxMem.Items.Add(s);
    ReadLn(FMem, s);
    ListBoxMemDet.Items.Add(s);
  end;
  ReadLn(FMem, s);
  n := StrToInt(s);
  for i := 0 to n - 1 do
  begin
    ReadLn(FMem, s);
    ListBox1.Items.Add(s);
    ReadLn(FMem, s);
    ListBox2.Items.Add(s);
  end;

  CloseFile(FMem);
end;

procedure TFormMemEnt.SpeedButton6Click(Sender: TObject);
var

```

```

    s : string;
begin
    if OpenFileDialog1.Execute then
        begin
            Yukle(OpenDialog1.FileName);
        end;
    end;

procedure TFormMemEnt.SpeedButton5Click(Sender: TObject);
var
    i : integer;
begin
    if SaveDialog1.Execute then
        begin
            AssignFile(FMem, SaveDialog1.FileName);
            ReWrite(FMem);
            WriteLn(FMem, IntToStr(ListBoxMem.Items.Count));
            for i := 0 to ListBoxMem.Items.Count - 1 do
                begin
                    WriteLn(FMem, ListBoxMem.Items[i]);
                    WriteLn(FMem, ListBoxMemDet.Items[i]);
                end;
            WriteLn(FMem, IntToStr(ListBox1.Items.Count));
            for i := 0 to ListBox1.Items.Count - 1 do
                begin
                    WriteLn(FMem, ListBox1.Items[i]);
                    WriteLn(FMem, ListBox2.Items[i]);
                end;
            CloseFile(FMem);
        end;
    end;
end;

```

```
procedure TFormMemEnt.ListeyeEkle;
var
  i : integer;
begin
  MemList.Clear;
  for i := 0 to ListBoxMem.Items.Count - 1 do
  begin
    MemList.Add(ListBoxMem.Items[i]);
  end;
  MemList2.Clear;
  for i := 0 to ListBox1.Items.Count - 1 do
  begin
    MemList2.Add(ListBox1.Items[i]);
  end;
end;
```

```
procedure TFormMemEnt.SpeedButton7Click(Sender: TObject);
begin
  ListeyeEkle;
  FormUye.ListeYukle;
  FormKural.ListeyeEkle;
end;
```

```
procedure TFormMemEnt.FormCreate(Sender: TObject);
begin
  if FileExists('Def.MEM') then
  begin
    Yukle('Def.MEM');
    SpeedButton7Click(nil);
  end;
end;
```

```
procedure TFormMemEnt.ListBox1Click(Sender: TObject);
begin
```

```
ListBox2.ItemIndex := ListBox1.ItemIndex;  
Selected := 1;  
WriteSelection;  
end;  
  
procedure TFormMemEnt.ListBox2Click(Sender: TObject);  
begin  
    ListBox1.ItemIndex := ListBox2.ItemIndex;  
    Selected := 1;  
    WriteSelection;  
end;  
  
end.
```



```

unit Oran;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, ExtCtrls;

type
  TFormOran = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    SpeedButton1: TSpeedButton;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    SpeedButton2: TSpeedButton;
    procedure SpeedButton2Click(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormOran: TFormOran;

implementation
uses Fuzz;
{$R *.DFM}

```

```
procedure TFormOran.SpeedButton2Click(Sender: TObject);
```

```
begin
```

```
  close;
```

```
end;
```

```
procedure TFormOran.SpeedButton1Click(Sender: TObject);
```

```
begin
```

```
  try
```

```
    HataOran := StrToFloat(Edit1.Text);
```

```
  except
```

```
    raise Exception.Create('Hatalı Oran Değeri.'+#13+'Lütfen Düzeltin!');
```

```
  end;
```

```
  try
```

```
    DeltaHataOran := StrToFloat(Edit2.Text);
```

```
  except
```

```
    raise Exception.Create('Hatalı Oran Değeri.'+#13+'Lütfen Düzeltin!');
```

```
  end;
```

```
  try
```

```
    CikisOran := StrToFloat(Edit3.Text);
```

```
  except
```

```
    raise Exception.Create('Hatalı Oran Değeri.'+#13+'Lütfen Düzeltin!');
```

```
  end;
```

```
  Close;
```

```
end;
```

```
procedure TFormOran.FormCreate(Sender: TObject);
```

```
begin
```

```
  SpeedButton1Click(nil);
```

```
end;
```

```
end.
```

```
unit OutVal;  
interface  
uses  
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,  
  Buttons, StdCtrls, Mask, ExtCtrls;
```

```
type
```

```
  TFormOutVal = class(TForm)  
    Bevel1: TBevel;  
    SpeedButton1: TSpeedButton;  
    Edit1: TEdit;  
    procedure SpeedButton1Click(Sender: TObject);  
    procedure Edit1KeyPress(Sender: TObject; var Key: Char);
```

```
  private
```

```
    { Private declarations }
```

```
  public
```

```
    { Public declarations }
```

```
  end;
```

```
var
```

```
  FormOutVal: TFormOutVal;
```

```
implementation
```

```
{ $R *.DFM }
```

```
procedure TFormOutVal.SpeedButton1Click(Sender: TObject);
```

```
begin
```

```
  Close;
```

```
end;
```

```
procedure TFormOutVal.Edit1KeyPress(Sender: TObject; var Key: Char);
```

```
begin
```

```
  if Key = #13 then
```

```

begin
  Key := #0;
  Close;
end;
end;

end.
unit ComPar;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, ExtCtrls, ComDrv32;

type
  TFormSerPar = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    RadioGroupSerPort: TRadioGroup;
    RadioGroupSerHiz: TRadioGroup;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure SetCommPar;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var

```

```
FormSerPar: TFormSerPar;
```

```
implementation
```

```
uses FuzKont, Sicil_;
```

```
{SR *.DFM}
```

```
procedure TFormSerPar.SetCommPar;
```

```
begin
```

```
FormFuzKont.CommPortDriver.DisConnect;
```

```
case RadioGroupSerPort.ItemIndex of
```

```
0 : FormFuzKont.CommPortDriver.ComPort := pnCOM1;
```

```
1 : FormFuzKont.CommPortDriver.ComPort := pnCOM2;
```

```
2 : FormFuzKont.CommPortDriver.ComPort := pnCOM3;
```

```
3 : FormFuzKont.CommPortDriver.ComPort := pnCOM4;
```

```
end;
```

```
case RadioGroupSerHiz.ItemIndex of
```

```
0 : FormFuzKont.CommPortDriver.ComPortSpeed := br2400;
```

```
1 : FormFuzKont.CommPortDriver.ComPortSpeed := br4800;
```

```
2 : FormFuzKont.CommPortDriver.ComPortSpeed := br9600;
```

```
3 : FormFuzKont.CommPortDriver.ComPortSpeed := br14400;
```

```
4 : FormFuzKont.CommPortDriver.ComPortSpeed := br19200;
```

```
5 : FormFuzKont.CommPortDriver.ComPortSpeed := br38400;
```

```
6 : FormFuzKont.CommPortDriver.ComPortSpeed := br56000;
```

```
7 : FormFuzKont.CommPortDriver.ComPortSpeed := br115200;
```

```
end;
```

```
if FormFuzKont.SpeedButton1.Down then
```

```
begin
```

```
FormFuzKont.CommPortDriver.Connect;
```

```
if not FormFuzKont.CommPortDriver.Connected then
```

```
Application.MessageBox('COM PORTUNA BAĞLANILAMADI','UYARI',
```

```
IDOK);
```

```
end;  
end;
```

```
procedure TFormSerPar.BitBtn2Click(Sender: TObject);  
begin  
  SetCommPar;  
  Close;  
end;
```

```
procedure TFormSerPar.BitBtn1Click(Sender: TObject);  
begin  
  Close;  
end;
```

```
procedure TFormSerPar.FormCreate(Sender: TObject);  
begin  
  RegInit;  
  SetCommPar;  
end;
```

```
end.
```

```
unit Trap;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,  
StdCtrls, Buttons, ExtCtrls, ComCtrls, Member;
```

```
type
```

```
TFormUye = class(TForm)  
  Image1: TImage;  
  StatusBar1: TStatusBar;
```

Panel1: TPanel;
Panel2: TPanel;
ScrollBar1: TScrollBar;
Edit1: TEdit;
StaticText1: TStaticText;
StaticText2: TStaticText;
ScrollBar2: TScrollBar;
Edit2: TEdit;
StaticText3: TStaticText;
ScrollBar3: TScrollBar;
Edit3: TEdit;
StaticText4: TStaticText;
ScrollBar4: TScrollBar;
Edit4: TEdit;
BitBtn1: TBitBtn;
BitBtn2: TBitBtn;
BitBtn3: TBitBtn;
BitBtn4: TBitBtn;
BitBtn5: TBitBtn;
BitBtn6: TBitBtn;
Splitter1: TSplitter;
ComboBoxInput: TComboBox;
StaticText5: TStaticText;
Bevel1: TBevel;
ComboBox1: TComboBox;
StaticText6: TStaticText;
SpeedButton1: TSpeedButton;
SaveDialog1: TSaveDialog;
OpenDialog1: TOpenDialog;
procedure FormActivate(Sender: TObject);
procedure BitBtn1Click(Sender: TObject);
procedure BitBtn3Click(Sender: TObject);
procedure ScrollBar1Change(Sender: TObject);
procedure BitBtn4Click(Sender: TObject);

```

procedure ComboBox1Change(Sender: TObject);
procedure GetTrpzData(No : integer; Giris : integer);

procedure BitBtn5Click(Sender: TObject);
procedure BitBtn6Click(Sender: TObject);
procedure SetBars;
procedure SpeedButton1Click(Sender: TObject);
procedure Listeyukle;
procedure DefaultLoad;
procedure FormCreate(Sender: TObject);
procedure Edit1Change(Sender: TObject);
procedure Edit2Change(Sender: TObject);
procedure Edit3Change(Sender: TObject);
procedure Edit4Change(Sender: TObject);
procedure FormResize(Sender: TObject);
procedure SetBar(b : MemberQuantize);
procedure ComboBoxInputChange(Sender: TObject);
procedure ListeHiz;
procedure ListeHizDegisimi;
private
  { Private declarations }
public
  { Public declarations }
end;

var
  FormUye: TFormUye;
  v1, v2, v3, v4 ,v5 : integer;
  Change :boolean = True;

implementation
uses Draw, Util;

```

```
{$R *.DFM}
```

```
procedure TFormUye.GetTrpzData(No : integer; Giris : integer);
```

```
begin
```

```
    MemberNo := No;
```

```
    v1 := MemberArray[Giris, MemberNo].TabanBas;
```

```
    v2 := MemberArray[Giris, MemberNo].Tavan1;
```

```
    v3 := MemberArray[Giris, MemberNo].Tavan2;
```

```
    v4 := MemberArray[Giris, MemberNo].TabanSon;
```

```
end;
```

```
procedure TFormUye.SetBars;
```

```
begin
```

```
    Change := False;
```

```
    ScrollBar1.Position := v1;
```

```
    ScrollBar2.Position := v2;
```

```
    ScrollBar3.Position := v3;
```

```
    ScrollBar4.Position := v4;
```

```
    Change := True;
```

```
    ScrollBar1.Change(nil);
```

```
end;
```

```
procedure TFormUye.Listeyukle;
```

```
var
```

```
    i : integer;
```

```
begin
```

```
{ ComboBox1.Clear;
```

```
    for i := 0 to MEMList.Count - 1 do
```

```
        ComboBox1.Items.Add(MEMList.Strings[i]);}
```

```
end;
```

```
procedure TFormUye.FormActivate(Sender: TObject);
```

```
begin
```

```
    ComboBoxInput.ItemIndex := 0;
```

```

ComboBox1.ItemIndex := 0;
ComboBoxInputChange(nil);
GetTrpzData(0,0);
SetBars;
Image1.Canvas.Brush.Color := clBlack;
Image1.Canvas.FillRect(Rect(0, 0, Image1.Width, Image1.Height));
DrawGrid(Image1, 10, 10, clNavy);
// DrawTrpzd(Image1, (v1+10)*10, (v2+10)*10, (v3+10)*10, (v4+10)*10,
clWhite);
ScrollBar1Change(nil);
end;

```

```

procedure TFormUye.BitBtn1Click(Sender: TObject);
begin
    DrawTrpzd(Image1, 10, 100, 120, 210, clWhite);
end;

```

```

procedure TFormUye.BitBtn3Click(Sender: TObject);
begin
    Close;
end;

```

```

procedure TFormUye.ScrollBar1Change(Sender: TObject);
var
    CenterG : integer;
    W : single;
begin
    if not change then exit;
    v1 := ScrollBar1.Position;
    v2 := ScrollBar2.Position;
    v3 := ScrollBar3.Position;
    v4 := ScrollBar4.Position;

    Edit1.Text := IntToStr(v1);

```

```

Edit2.text := IntToStr(v2);
Edit3.Text := IntToStr(v3);
Edit4.text := IntToStr(v4);

TekrarCiz(Image1, clBlack, clNavy);
W := 100 / MemberQuantization[ComboBoxInput.ItemIndex].Bitis ;
CenterG := Image1.Width div 2;
DrawTrpzd(Image1, Trunc(v1*W)+CenterG, Trunc(v2*W)+CenterG,
           Trunc(v3*W)+CenterG, Trunc(v4*W)+CenterG, clWhite);
end;

procedure TFormUye.BitBtn4Click(Sender: TObject);
begin
  AddMember;
end;

procedure TFormUye.ComboBox1Change(Sender: TObject);
begin
  SetBar(MemberQuantization[ComboBoxInput.ItemIndex]);
  GetTrpzData(ComboBox1.ItemIndex, ComboBoxInput.ItemIndex);
  SetBars;
  TekrarCiz(Image1,clBlack, clNavy);
  ScrollBar1 Change(nil);
end;

procedure TFormUye.BitBtn5Click(Sender: TObject);
begin
  if SaveDialog1.Execute then
    SaveMembers(SaveDialog1.FileName);
end;

procedure TFormUye.DefaultLoad;
begin
  LoadMembers('MEMBER.DAT');

```

```
    ComboBox1.Change(nil);  
end;
```

```
procedure TFormUye.SetBar(b : MemberQuantize);  
begin  
    ScrollBar1.Min := b.Baslangic;  
    ScrollBar1.Max := b.Bitis;  
    ScrollBar2.Min := b.Baslangic;  
    ScrollBar2.Max := b.Bitis;  
    ScrollBar3.Min := b.Baslangic;  
    ScrollBar3.Max := b.Bitis;  
    ScrollBar4.Min := b.Baslangic;  
    ScrollBar4.Max := b.Bitis;  
end;
```

```
procedure TFormUye.BitBtn6Click(Sender: TObject);  
begin  
    if OpenFileDialog1.Execute then  
        begin  
            LoadMembers(OpenDialog1.FileName);  
        end;  
    ComboBox1.Change(nil);  
end;
```

```
procedure TFormUye.SpeedButton1Click(Sender: TObject);  
begin  
    change := False;  
    ScrollBar1.Position := 0;  
    ScrollBar2.Position := 0;  
    ScrollBar3.Position := 0;  
    ScrollBar4.Position := 0;  
    change := True;  
    ScrollBar1.Change(nil);  
end;
```

```
procedure TFormUye.FormCreate(Sender: TObject);
begin
    ComboBox1.ItemIndex := 0;
    ComboBoxInput.ItemIndex := 0;
    DefaultLoad;
    ComboBoxInputChange(nil);
end;
```

```
procedure TFormUye.Edit1Change(Sender: TObject);
begin
    try
        ScrollBar1.Position := StrToInt(Edit1.Text);
    except

end;
end;
```

```
procedure TFormUye.Edit2Change(Sender: TObject);
begin
    try
        ScrollBar2.Position := StrToInt(Edit2.Text);
    except

end;
end;
```

```
procedure TFormUye.Edit3Change(Sender: TObject);
begin
    try
        ScrollBar3.Position := StrToInt(Edit3.Text);
    except

end;
```

```
end;
```

```
procedure TFormUye.Edit4Change(Sender: TObject);
```

```
begin
```

```
  try
```

```
    ScrollBar4.Position :=StrToInt(Edit4.Text);
```

```
  except
```

```
  end;
```

```
end;
```

```
procedure TFormUye.FormResize(Sender: TObject);
```

```
begin
```

```
  SetBars;
```

```
  TekrarCiz(Image1,clBlack, clNavy);
```

```
  DrawTrpzd(Image1, (v1+10)*10, (v2+10)*10, (v3+10)*10, (v4+10)*10,  
clWhite);
```

```
end;
```

```
procedure TFormUye.ListeHiz;
```

```
var
```

```
  i : integer;
```

```
begin
```

```
  ComboBox1.Clear;
```

```
  for i := 0 to MEMList.Count - 1 do
```

```
    ComboBox1.Items.Add(MEMList.Strings[i]);
```

```
  ComboBox1.ItemIndex := 0;
```

```
end;
```

```
procedure TFormUye.ListeHizDegisimi;
```

```
var
```

```
  i : integer;
```

```
begin
```

```
  ComboBox1.Clear;
```

```
for i := 0 to MEMList2.Count - 1 do
  ComboBox1.Items.Add(MEMList2.Strings[i]);
  ComboBox1.ItemIndex := 0;
end;

procedure TFormUye.ComboBoxInputChange(Sender: TObject);
begin
  case ComboBoxInput.ItemIndex of
    0 : ListeHiz;
    1 : ListeHizDegisimi;
  end;
  ComboBox1.Change(nil);
end;

end.
```



EK-2 MİKRODENETLEYİCİYE AİT "C" DİLİ KODU

/*****

Project : fuzzy dc motor control

Version : 1.1

Date : 14.05.1999

Author : sadık okuyucu

Company : Istanbul, Turkey

Comments: Bulanık D.C motor kontrol tez çalışması

Chip type : AT90S8535

Clock frequency : 4,000000 MHz

Memory model : Small

Internal RAM size: 512

External RAM size: 0

Data Stack size : 128

*****/

#include <90s8535.h>

#pragma used+

#pragma regalloc-

// UART Receiver buffer

char rx_buffer[10];

int adc_ok;

unsigned char rx_wr_index,rx_rd_index,rx_counter;

// This flag is set on UART Receiver buffer overflow

bit rx_buffer_overflow;

#pragma regalloc+

#pragma used-

// UART Receiver interrupt service routine

```

#pragma savereg-
interrupt [UART_RXC] void uart_rx_isr(void)
{
#asm
    .equ __rx_buffer_size=10
    push r26
    in r26,sreg
    push r26
    push r27
    push r30
#endasm
#ifdef _MODEL_TINY_
#asm
    ldi r26,_rx_buffer
    lds r30,_rx_wr_index
    add r26,r30
    clr r27
#endasm
#endif
#ifdef _MODEL_SMALL_
#asm
    push r31
    ldi r26,low(_rx_buffer)
    ldi r27,high(_rx_buffer)
    lds r30,_rx_wr_index
    clr r31
    add r26,r30
    adc r27,r31
#endasm
#endif
#asm
    in r30,udr
    st x,r30
    lds r30,_rx_wr_index

```

```

    inc r30
    cpi r30,__rx_buffer_size
    brlo __uart_rx_isr0
    clr r30
__uart_rx_isr0:
    sts _rx_wr_index,r30
    lds r30,_rx_counter
    inc r30
    sts _rx_counter,r30
    cpi r30,__rx_buffer_size+1
    brlo __uart_rx_isr1
#endasm
rx_buffer_overflow=1;
#asm("__uart_rx_isr1:")
#ifdef _MODEL_SMALL_
#asm("pop r31")
#endif
#asm
    pop r30
    pop r27
    pop r26
    out sreg,r26
    pop r26
#endasm
}
#pragma savereg+

#ifdef _DEBUG_TERMINAL_IO_
// Get a character from the UART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma warn-
#pragma used+
char getchar(void)
{

```

```

#asm
    lds r30,_rx_counter
    tst r30
    breq __getchar
#endasm
#ifdef _MODEL_TINY_
#asm
    ldi r26,_rx_buffer
    lds r30,_rx_rd_index
    add r26,r30
    clr r27
#endasm
#endif
#ifdef _MODEL_SMALL_
#asm
    ldi r26,low(_rx_buffer)
    ldi r27,high(_rx_buffer)
    lds r30,_rx_rd_index
    clr r31
    add r26,r30
    adc r27,r31
#endasm
#endif
#asm
    inc r30
    cpi r30,__rx_buffer_size
    brlo __getchar0
    clr r30
__getchar0:
    sts _rx_rd_index,r30
    ld r30,x
    cli
    lds r26,_rx_counter
    dec r26

```

```

    sts _rx_counter,r26
    sei
#endasm
}
#pragma used-
#ifdef _WARNINGS_ON_
#pragma warn+
#endif
#endif
#pragma used+
#pragma regalloc-
// UART Transmitter buffer
char tx_buffer[10];
unsigned char tx_wr_index,tx_rd_index,tx_counter;
#pragma regalloc+
#pragma used-
// UART Transmitter interrupt service routine
#pragma savereg-
interrupt [UART_TXC] void uart_tx_isr(void)
{
#asm
    .equ __tx_buffer_size=10
    push r26
    in r26,sreg
    push r26
    lds r26,_tx_counter
    tst r26
    breq __uart_tx_isr1
    dec r26
    sts _tx_counter,r26
    push r27
    push r30
#endasm
#ifdef _MODEL_TINY_

```

```

#asm
    ldi r26,_tx_buffer
    lds r30,_tx_wr_index
    add r26,r30
    clr r27
#endasm
#endif
#ifdef _MODEL_SMALL_
#asm
    push r31
    ldi r26,low(_tx_buffer)
    ldi r27,high(_tx_buffer)
    lds r30,_tx_wr_index
    clr r31
    add r26,r30
    adc r27,r31
#endasm
#endif
#asm
    inc r30
    cpi r30,___tx_buffer_size
    brlo __uart_tx_isr0
    clr r30
__uart_tx_isr0:
    sts _tx_wr_index,r30
    ld r30,x
#endasm
#asm("out udr,r30")
#ifdef _MODEL_SMALL_
#asm("pop r31")
#endif
#asm
    pop r30
    pop r27

```

```

__uart_tx_isr1:
    pop r26
    out sreg,r26
    pop r26
#endasm
}
#pragma savereg+

#ifndef _DEBUG_TERMINAL_IO_
// Write a character to the UART Transmitter buffer
#define _ALTERNATE_PUTCHAR_
#pragma warn-
#pragma used+
void putchar(char c)
{
#asm
    lds r30,__tx_counter
    cpi r30,__tx_buffer_size
    brsh _putchar
    tst r30
    brne __putchar0
    sbis usr,udre
    rjmp __putchar0
    ld r30,y
    out udr,r30
    rjmp __putchar3
__putchar0:
#endasm
#ifdef _MODEL_TINY_
#asm
    ldi r26,__tx_buffer
    lds r30,__tx_rd_index
    add r26,r30
    clr r27

```

```

#endasm
#endif
#ifdef _MODEL_SMALL_
#asm
    ldi r26,low(_tx_buffer)
    ldi r27,high(_tx_buffer)
    lds r30,_tx_rd_index
    clr r31
    add r26,r30
    adc r27,r31
#endasm
#endif
#asm
    inc r30
    cpi r30,__tx_buffer_size
    brlo __putchar1
    clr r30
__putchar1:
    sts _tx_rd_index,r30
    ld r30,y
    st x,r30
    cli
    lds r30,_tx_counter
    inc r30
    sts _tx_counter,r30
    sei
__putchar3:
#endasm
}
#pragma used-
#ifdef _WARNINGS_ON_
#pragma warn+
#endif
#endif

```

```

// Standard Input/Output functions
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <delay.h>
unsigned int adc_data;

// ADC interrupt service routine
#pragma savereg-
interrupt [ADC_INT] void adc_isr(void)
{
#asm
    push r30
    push r31
#endasm
// Read the ADC conversion result
adc_data=ADCW;
#asm
    pop r31
    pop r30
#endasm

}
#pragma savereg+

// Read the ADC conversion result
// with noise canceling
unsigned int read_adc(unsigned char adc_input)
{
#ifdef BANDGAP
ADMUX=adc_input|0x40;
#else
ADMUX=adc_input;

```

```

#endif
#asm
    in r30,mcucr
    sbr r30,__se_bit
    cbr r30,__sm_mask
    out mcucr,r30
    sleep
    cbr r30,__se_bit
    out mcucr,r30
#endasm
return adc_data;
}

```

```

// Declare your global variables here

```

```

char b;
char s[15];

```

```

// int fv,v,durum,tb,i,j;

```

```

int fv,durum,tb,i, j;

```

```

char v;

```

```

int goncnt;

```

```

//double toplam;

```

```

void main(void)

```

```

{

```

```

// Declare your local variables here

```

```

// Input/Output Ports initialization

```

```

// Port A

```

```

DDRA=0x00;

```

```

PORTA=0x00;

```

```

// Port B

```

```

DDRB=0x00;

```

```

PORTB=0x00;

```

```
// Port C
DDRC=0xff;
PORTC=0x00;

// Port D
DDRD=0x20;
PORTD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Output Compare
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 3,906 kHz
// Mode: 8 bit Pulse Width Modulation
// OC1A output: Non-Inv.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x91;
TCCR1B=0x01;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
```

```

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Output Compare
// OC2 output: Disconnected
TCCR2=0x00;
ASSR=0x00;
TCNT2=0x00;
OCR2=0x00;

// UART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// UART Receiver: On
// UART Transmitter: On
UCR=0xD8;
// UART Baud rate: 9600
UBRR=0x19;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;

// ADC initialization
// ADC Clock frequency: 2000,000 kHz .
ADCSR=0x89;

// Global enable interrupts
#asm("sei")

goncnt = 0;

while (1)
{

```

```

if (goncnt == 0)
{

    {
        //fv=read_adc(0);
        fv=v;
    }

    itoa(fv, s);
    goncnt = s[0];
    i = -1;
}

while (rx_counter > 0)
{
    b=getchar();
    if (b== '@') durum = 2;
    switch (durum)
    {
        case 1: if (b== '@') durum = 2;
                break;
        case 2: if (b== 'H')
                {
                    durum = 3;
                    v = 0;
                    tb = 1000;
                }
                break;
        case 3: if (b == '*') durum = 4;
                else
                {
                    if ((b >= '0') && (b <= '9'))

```

```

        {
            v = v + (b - '0')*tb;
            tb = tb / 10;
        }
    }
    break;
case 4 : durum = 1;
        PORTC=v & 0xff;
            break;
} //switch
} //while

```

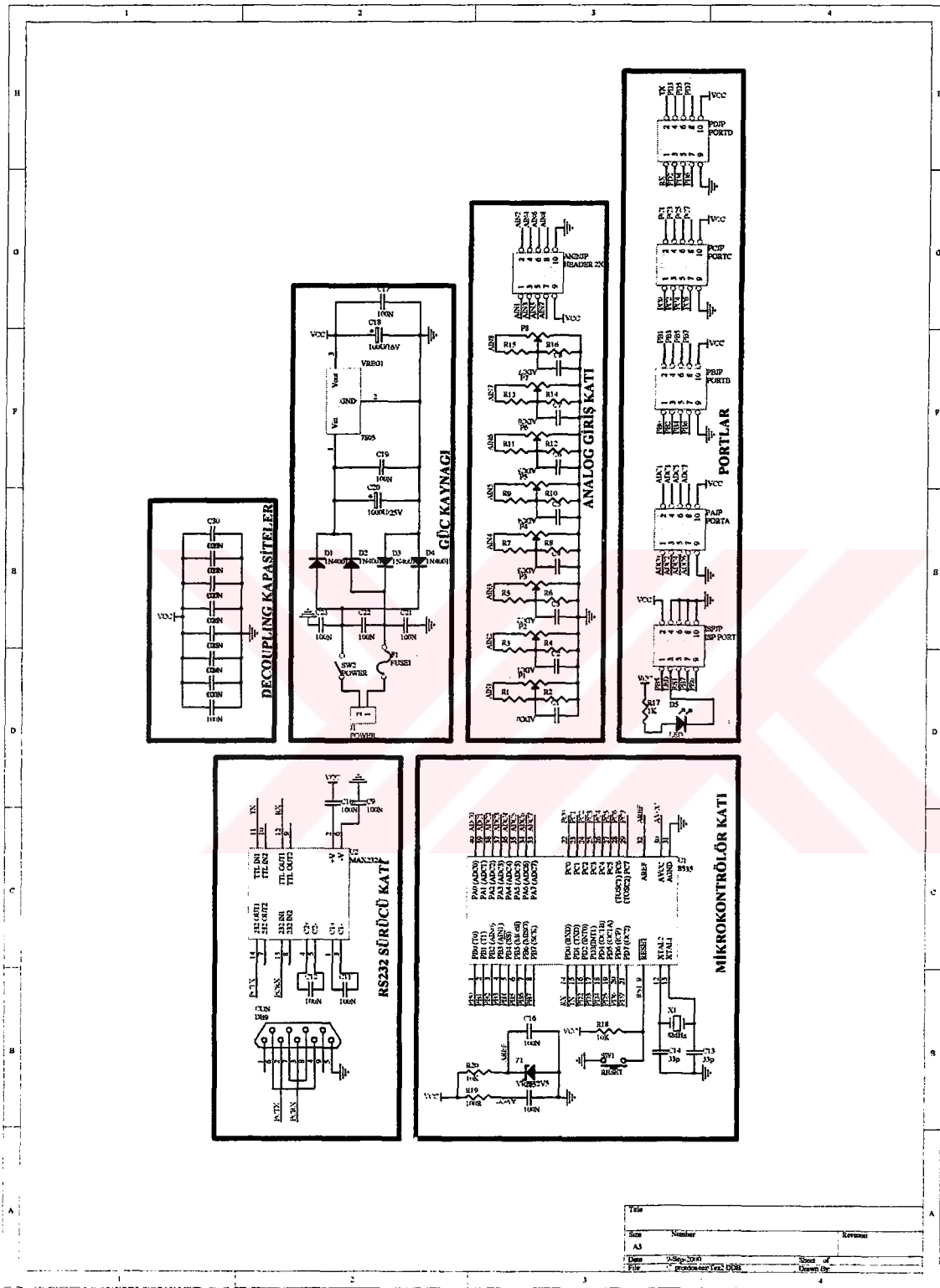
```

if (goncnt!=0)
{
    if (i== -1) putchar('@');
    else
    {
        goncnt=s[i];
        if (goncnt==0) putchar('*');
        else putchar(s[i]);
    }
    i++;
}

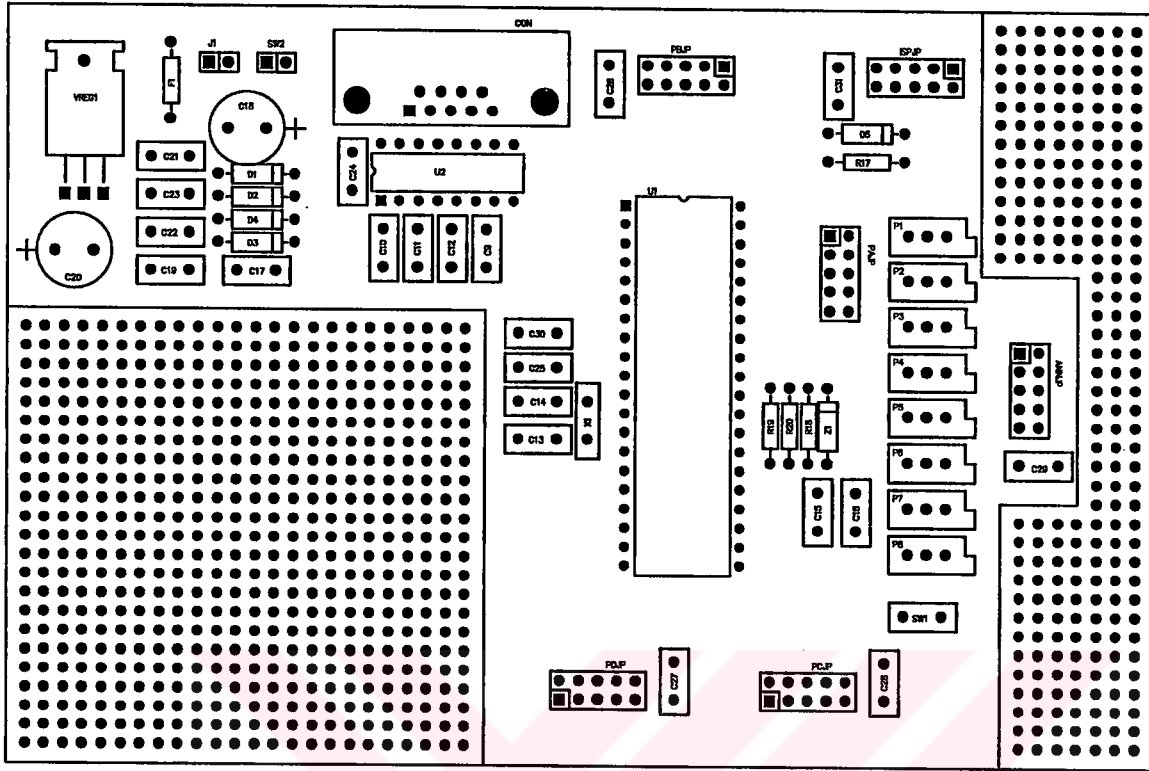
};
}

```

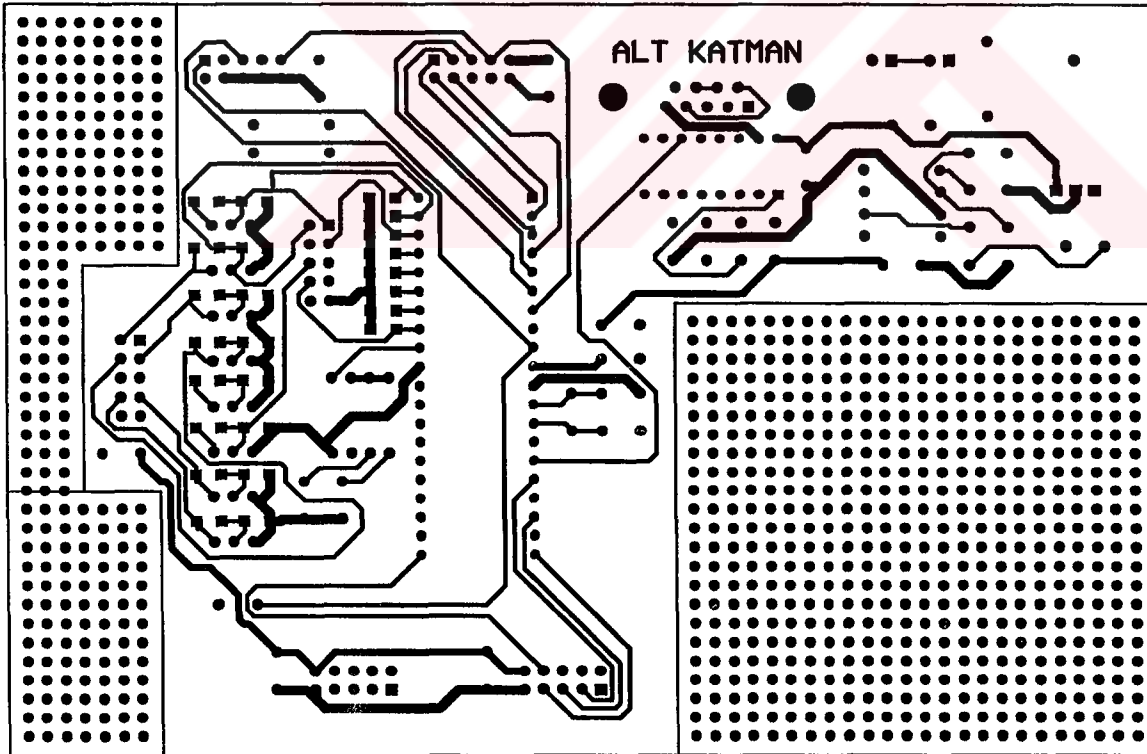
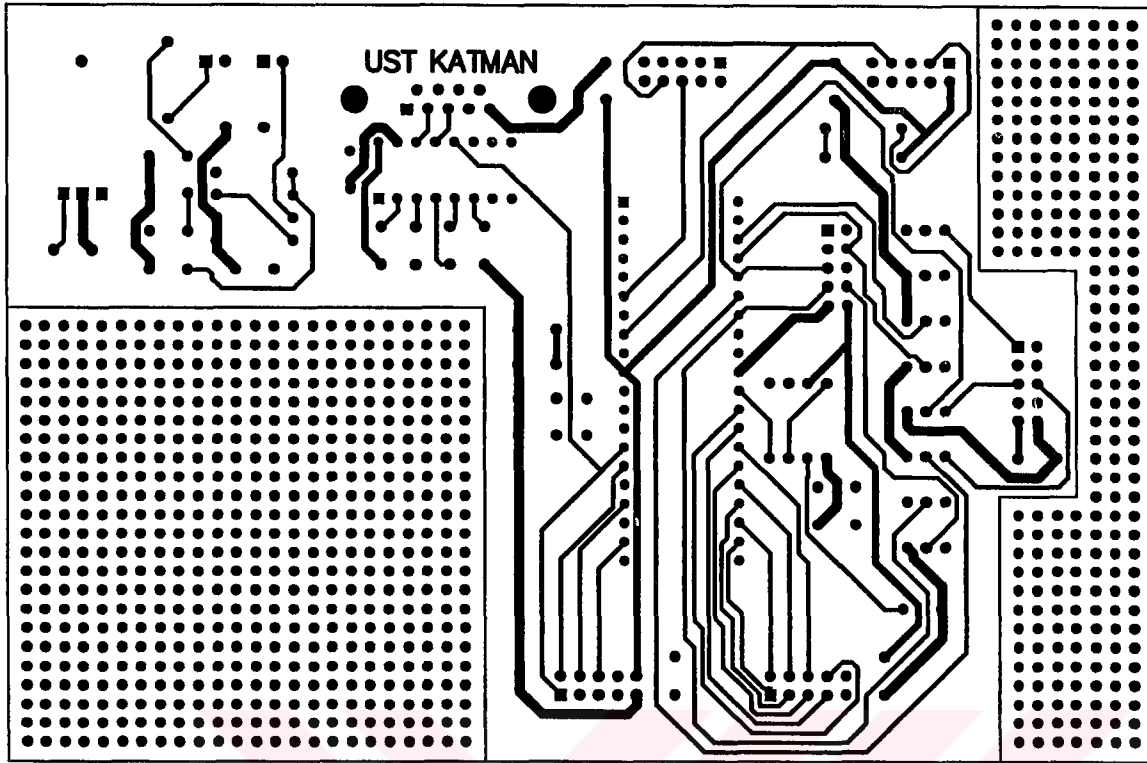
EK-3 MİKRO DENETLEYİCİ KARTA AİT ŞEMA VE YERLEŞİM PLANI



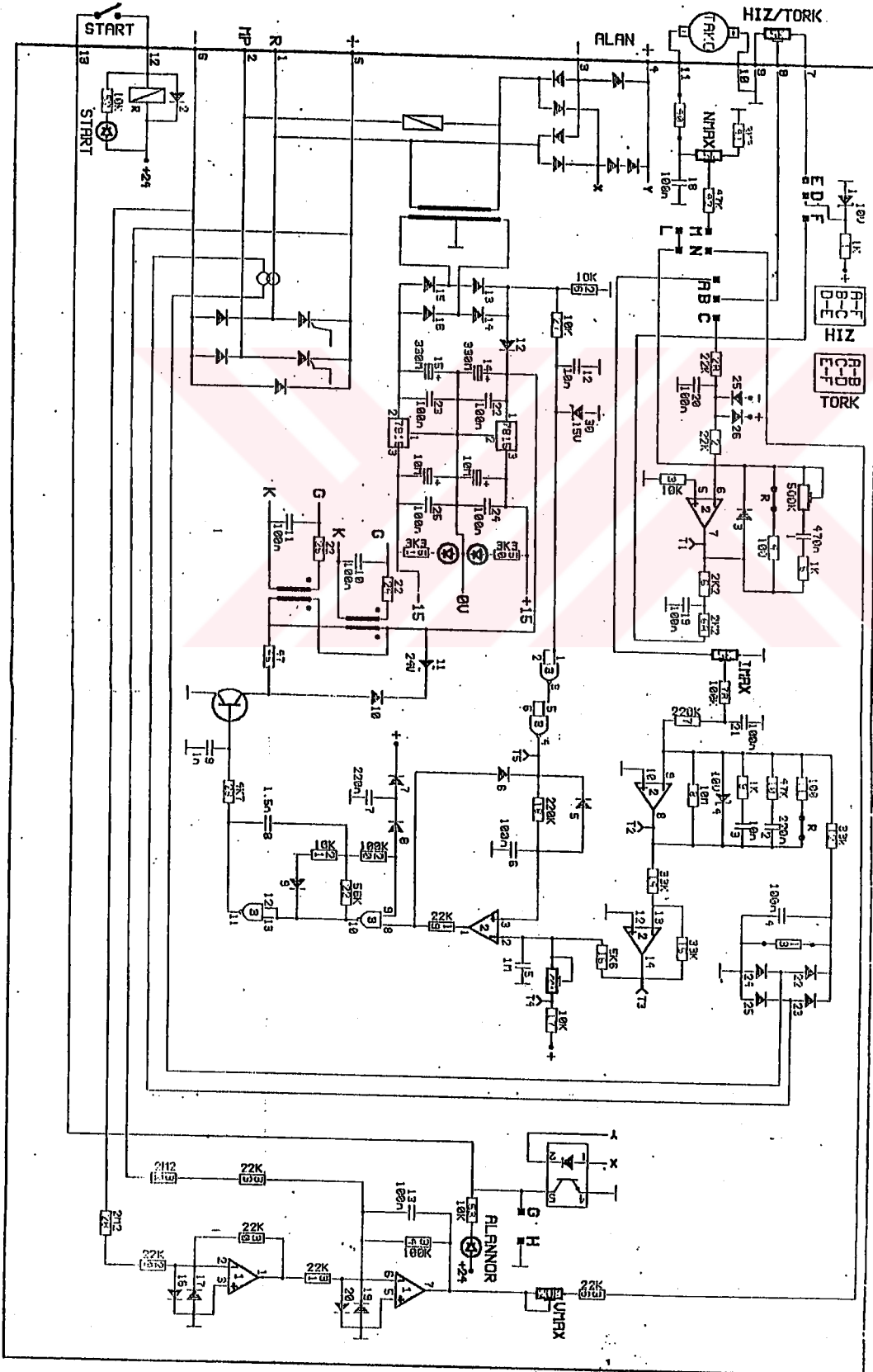
EK 3-1 MİKRODENETLEYİCİ KART BASKI DEVRE YERLEŞİM PLANI



EK 3-1 MİKRODENETLEYİCİ KART BASKI DEVRE GÖRÜNÜŞÜ



EK-4 DC MOTOR SÜRÜCÜ DEVRE ŞEMASI



ÖZGEÇMİŞ

1971 yılında Kastamonu, Tosya ilçesinde doğdu. İlköğrenimini Üsküdar'da tamamladı. Orta öğrenimini İstanbul Beykoz Denizcilik ve Su Ürünleri Meslek Lisesi Elektronik bölümünde tamamladı. 1991 yılında M.Ü. Teknik Eğitim Fakültesi Elektronik-Bilgisayar bölümünü kazandı ve 1996 yılında Bilgisayar-Kontrol Anabilim Dalından mezun olduktan sonra Milli Eğitim Bakanlığında öğretmen olarak göreve başladı. İki yıl süreyle kendi mezun olduğu meslek lisesinde bilgisayar bölüm şefliği yaptı. 1998-1999 yılında yarım dönem özel Florya kolejinde bilgisayar öğretmenliği ve bilgi işlem müdürlüğü yaptı. 1999 yılında Tübitak Ulusal Elektronik ve Kriptoloji Enstitüsünde uzman yardımcısı olarak çalışmaya başladı. Halen bu göreve devam etmektedir.

