

## İÇİNDEKİLER

|                                                                 |            |
|-----------------------------------------------------------------|------------|
| <b>KISALTMALAR</b>                                              | <b>iii</b> |
| <b>TABLO LİSTESİ</b>                                            | <b>iv</b>  |
| <b>ŞEKİL LİSTESİ</b>                                            | <b>v</b>   |
| <b>SEMBOL LİSTESİ</b>                                           | <b>vi</b>  |
| <b>ÖZET</b>                                                     | <b>vii</b> |
| <b>ABSTRACT</b>                                                 | <b>ix</b>  |
| <br>                                                            |            |
| <b>1. GİRİŞ</b>                                                 | <b>1</b>   |
| 1.1. Giriş ve Çalışmanın Amacı                                  | 1          |
| <br>                                                            |            |
| <b>2. INTERRUPT (KESME) SİNYALLERİ</b>                          | <b>2</b>   |
| 2.1 İnterruptlar                                                | 2          |
| 2.2 Donanım İnterruptları                                       | 2          |
| 2.3 Programlanabilir Interrupt Kontrol Edici (PIC)              | 4          |
| 2.4 PIC Konfigürasyonu                                          | 7          |
| <br>                                                            |            |
| <b>3. PARALEL PORT ÖZELLİKLERİ VE INTERRUPT SİNYALLERİ</b>      |            |
| <b>ALMAYA YÖNELİK AYARLARININ YAPILMASI</b>                     | <b>8</b>   |
| <br>                                                            |            |
| 3.1 ParalelPort Temel Özellikleri                               | 8          |
| 3.2 Paralel Portun Interrupt(Kesme) Sinyallerine Açılması       | 10         |
| <br>                                                            |            |
| <b>4. WINDOWS 9x İŞLETİM SİSTEMİ İÇİN SANAL SÜRÜCÜ TASARIMI</b> | <b>14</b>  |
| 4.1 Windows İşletim Sistemi Hakkında Genel Bilgiler             | 14         |
| 4.2 VMM - Sanal Makina Yöneticisi (VIRTUAL MACHINE<br>MANAGER)  | 15         |
| 4.3 Sanal Cihaz Sürücü (VXD)                                    | 15         |
| 4.4 VxD'ler arasında iletişim                                   | 16         |
| 4.4.1 Kontrol Mesajları                                         | 16         |
| 4.4.2 Servis API'leri                                           | 17         |
| 4.4.3 CallBack Fonksiyonları                                    | 18         |
| 4.5 Virtual Machine Manager (Sanal Makina Yönetici)             | 18         |
| 4.5.1 Bellek Yönetimi                                           | 19         |
| 4.6 İnterrupt Kontrolü                                          | 20         |
| 4.7 Uygulama zamanlamalarının ayarlanması                       | 20         |

|                                                                                  |           |
|----------------------------------------------------------------------------------|-----------|
| 4.8 Sanal Cihaz Sürücüsü İskelet Yapısı                                          | 21        |
| 4.8.1 LE Dosya yapısı                                                            | 21        |
| 4.8.2 VxD İskeleti                                                               | 25        |
| 4.9 VxD'nin Derlenmesi                                                           | 27        |
| 4.10 Nesne Tabanlı Yazılım ile VxD'nin Dinamik Olarak Yüklenmesi ve Kaldırılması | 28        |
| <b>5. PC İLE SİNYAL ÜRETİMİ</b>                                                  | <b>32</b> |
| <b>6. YAPAY SİNİR AĞLARI</b>                                                     | <b>34</b> |
| 6.1 YSA'da Hesaplama Yaklaşımları                                                | 35        |
| 6.2 Ağ Mimarileri                                                                | 37        |
| 6.3 Öğrenme Kuralları                                                            | 38        |
| 6.3.1 Hebb Kuralı                                                                | 40        |
| 6.3.2 Delta kuralı                                                               | 41        |
| 6.3.3 Genelleştirilmiş Delta kuralı (Geri Dağılım)                               | 42        |
| <b>7. YAPAY SİNİR AĞLARI İLE SİSTEM TANIMLAMA</b>                                | <b>44</b> |
| 7.1 Tanımlama konsepti                                                           | 45        |
| 7.2 Yapay sinir ağının oluşturulması, Eğitimi ve Kullanımı                       | 47        |
| <b>8. SİSTEMİN ve YAZILIMIN TASARLANMASI ve GENEL ÖZELLİKLERİ</b>                | <b>51</b> |
| 8.1 Sistemi Oluşturan Temel Elemanlar ve Özellikleri                             | 51        |
| 8.1.1. Motorun Özellikleri                                                       | 52        |
| 8.1.2. Optik Enkoder Özellikleri                                                 | 53        |
| 8.1.3. Güçlendirici                                                              | 54        |
| 8.2 Yazılım Genel Özellikleri                                                    | 56        |
| 8.2.1 VXD yazılımı                                                               | 56        |
| 8.2.2 PWM darbelerinin üretimi, VxD kontrolü ve YSA algoritmaları                | 57        |
| 8.3 Kullanıcı Arayüzü                                                            | 61        |
| <b>9. SONUÇ</b>                                                                  | <b>66</b> |



## **KISALTMALAR**

|              |                                                    |
|--------------|----------------------------------------------------|
| <b>PIC</b>   | : Programmable Interrupt Controller                |
| <b>LAPIC</b> | : Local Advanced Programmable Interrupt Controller |
| <b>VMM</b>   | : Virtual Machine Management                       |
| <b>VxD</b>   | : Virtual x Driver                                 |
| <b>YSA</b>   | : Yapay Sinir Ağları                               |
| <b>DC</b>    | : Direct Current                                   |
| <b>DDK</b>   | : Device Development Kit                           |
| <b>ISR</b>   | : Interrupt Service Routine                        |
| <b>EOI</b>   | : End of interrupt                                 |
| <b>ICW</b>   | : Initialization Command word                      |
| <b>OCW</b>   | : Operational Command word                         |
| <b>LE</b>    | : Linear Executable                                |

## TABLO LİSTESİ

|                    |                                        | <b><u>Sayfa No</u></b> |
|--------------------|----------------------------------------|------------------------|
| <b>Tablo 2.1</b>   | PIC1 İşlem Kontrol Değerleri (0x21)    | 3                      |
| <b>Tablo 2.2</b>   | PIC2 İşlem Kontrol Değerleri (0xA1)    | 3                      |
| <b>Tablo 3.2.1</b> | Interrupt Vektörleri Tablosu           | 11                     |
| <b>Tablo 4.8.1</b> | Segment bildirim makroları             | 24                     |
| <b>Tablo 5.1</b>   | 8253 Zamanlayıcının Portları           | 30                     |
| <b>Tablo 5.2</b>   | 8253 zamanlayıcı kontrol bit değerleri | 31                     |

## ŞEKİL LİSTESİ

|                    |                                                      | <b><u>Sayfa No</u></b> |
|--------------------|------------------------------------------------------|------------------------|
| <b>Şekil 2.3.1</b> | PIC blok diyagramı                                   | 5                      |
| <b>Şekil 2.4.1</b> | PIC konfigürasyonu                                   | 7                      |
| <b>Şekil 3.1.1</b> | Paralel port giriş- çıkışları                        | 9                      |
| <b>Şekil 3.1.2</b> | Paralel port portları                                | 9                      |
| <b>Şekil 6.1</b>   | Yapay nöronun yapısı                                 | 32                     |
| <b>Şekil 6.2</b>   | İki katmanlı ileri beslemeli ağ mimarisi             | 35                     |
| <b>Şekil 6.3</b>   | Öğrenme aşamalarının blok diyagram şekilde gösterimi | 35                     |
| <b>Şekil 6.4</b>   | Aktivasyon fonksiyonlarının şekli ve özellikleri     | 36                     |
| <b>Şekil 6.5</b>   | Farklı mimari konfigürasyonlar                       | 36                     |
| <b>Şekil 7.1</b>   | Tanımlama modeli                                     | 37                     |
| <b>Şekil 8.1</b>   | Sistemin Blok Şeması                                 | 44                     |
| <b>Şekil 8.2</b>   | Güçlendirici devre şeması                            | 45                     |
| <b>Şekil 8.3</b>   | Sistemin yapay sinir ağı modeli                      | 46                     |

## RESİM LİSTESİ

|                   |                             | <b><u>Sayfa No</u></b> |
|-------------------|-----------------------------|------------------------|
| <b>Resim 8.1</b>  | Tasarlanan Sistem           | 52                     |
| <b>Resim 8.2</b>  | DC servo motor              | 53                     |
| <b>Resim 8.3</b>  | Optik enkoder               | 53                     |
| <b>Resim 8.4</b>  | Delikli Disk                | 54                     |
| <b>Resim 8.5</b>  | Güçlendirici                | 55                     |
| <b>Resim 8.6</b>  | Ana Form                    | 62                     |
| <b>Resim 8.7</b>  | Dosya işlemleri             | 62                     |
| <b>Resim 8.8</b>  | YSA ayarları penceresi      | 63                     |
| <b>Resim 8.9</b>  | PWM ayar penceresi          | 64                     |
| <b>Resim 8.10</b> | VxD yükleme-kaldırma menüsü | 65                     |

## SEMBOL LİSTESİ

**A, B, C ...** : Sistem matrisleri

**d<sup>(i)</sup>** : İstenen ağ çıkışı

**e** : Ağ hatası

**u(k)** : Giriş sinyali

**U** : Giriş uzayı

**Y** : Çıkış uzayı

**W<sub>ji</sub>** : Ağırlık matrisi

**x<sub>i</sub>** : Nöron girişi

**x** : Giriş vektörü

**y<sub>p</sub>(k)** : Sistem çıkışı

|                           |                                       |
|---------------------------|---------------------------------------|
| <b>Üniversitesi</b>       | <b>: İstanbul Teknik Üniversitesi</b> |
| <b>Enstitüsü</b>          | <b>: Fen Bilimleri</b>                |
| <b>Anabilim Dalı</b>      | <b>: Uçak Mühendisliği</b>            |
| <b>Programı</b>           | <b>: Uçak Mühendisliği</b>            |
| <b>Tez Danışmanı</b>      | <b>: Doç. Dr. Ramazan TAŞALTIN</b>    |
| <b>Tez Türü ve Tarihi</b> | <b>: Yüksek Lisans – Mayıs-2002</b>   |

## **ÖZET**

### **NESNEL TABANLI YAZILIM VE YAPAY SİNİR AĞI TEMELLİ DC MOTOR KONTROLÜ**

**Ufuk BEKÇİ**

Bu çalışmada kişisel bilgisayarın paralel portu kullanılarak DC motor kontrolü yapılmıştır. Pratikte ADC ve DAC kartları kullanılarak yapılan kompleks darbe genişlik modülasyonu algoritmaları, Windows 98 ortamında yazılan nesnel tabanlı yazılım paketi Delphi kullanılarak yapılmıştır. Geliştirilen yöntem nesnel tabanlı ve görsel ağırlıklı yazılımların avantajları ile ADC ve DAC donanımı bulunmayan standart dizüstü bilgisayarın avantajlarını birleştirmiştir..

Geliştirilen algoritma motorun en kısa zamanda verilen referans açığa gelmesini hedeflemektedir. Gerekli sinyaller bilgisayarın paralel portundan gelen kodlara göre güçlendirici devre tarafından üretilmektedir. DC motor darbe genişlik modülasyonu (PWM) temelli çalışmaktadır. PWM için gerekli hesaplar yapay sinir ağları tarafından motorun daha önceki giriş çıkış bilgileri kullanılarak yapılmaktadır. Böylece motor değiştirildiğinde veya motorun yükü değiştiğinde yeni durum için gerekli hesapları YSA yapmakta ve dinamik denklemlerin her durum için bilinmesine gerek duymamaktadır.

**DOS ortamında PC paralel portunu kontrol etmek çok kolaydır. Windows ortamında ise PC paralel portu kontrol etmek için gerekli kesme işaretlerinin kullanılması oldukça karmaşıktır. Bu çalışmada bu problem bir sanal sürücü(VxD) tanımlanarak çözümlenmiştir. Geliştirilen sürücü yazılımı bilgisayara yeni bir aygıt takıldığında olduğu gibi, kontrol sistemini bilgisayara bir defaya mahsus olarak tanıtmaktadır. Bu çalışma kapsamında tasarlanan ve gerçekleştirilen güçlendirici devre 5KHz'e kadar PWM darbelerini kuvvetlendirebilmektedir. Bu hızda sinyal üretebilmek için yüksek hassasiyetli zamanlayıcılara gerek vardır ve bu problem bilgisayar anakartı üzerinde bulunan 8253 zamanlayıcı programlanarak çözümlenmiştir.**

**Anahtar Kelimeler: VxD, İnterrupt, YSA(Yapay Sinir Ağı), paralel port, nesne tabanlı yazılım**

**University** : İstanbul Technical University  
**Institute** : Institute of Science and Technology  
**Science Programme** : Aeronautical Engineering  
**Programme** : Aeronautical Engineering  
**Supervisor** : Doç. Dr. Ramazan TAŞALTIN  
**Degree Awarded and Date** : Master- May-2002

## **ABSTRACT**

### **DC MOTOR CONTROL USING NEURAL NETWORKS AND OBJECT ORIENTED SOFTWARE**

**Ufuk BEKÇİ**

In this thesis, a DC motor is controlled by a personal computer using its parallel port. The complex pulse width modulation algorithms which is usually achieved by data acquisition systems is accomplished by using only PC parallel port and object oriented software Delphi which works on Windows 98 (and higher) environment.

The developed method combines the advantages of object oriented software packages and that of a portable PC without any data acquisition systems. The developed algorithm aims that the motor reaches the given reference position in shortest time. The necessary signals is produced by a power amplifier according to the received codes from the PC parallel port. The DC motor works on pulse width modulation base. The necessary calculation for PWM is carried out by neural networks algorithms which uses the previous input-output relations of DC motor.. The advantage of this system is that if motor is changed or if the load of the motor varies, then YSA can easily calculate the new PWM codes without the dynamic equations of the system.

**As it is known that it is easy to control the parallel port of a PC computer in a DOS environment. But it is a complex problem in a Windows environment. The Windows does not allow the use of interrupt signals very easily. In this work this problem is solved by writing a virtual driver program. The developed driver program presents the control system to the computer, once, as a new hardware. The amplifier circuit which is designed and implemented for this project is capable of amplifying PWM pulses up to 5 KHz. In order to produce signals at this frequency, accurate timers are required and 8253 timer circuit on the computer board is used.**

**Keywords: VxD, Interrupt, Neural Network, parallel port, object oriented software**

## 1. GİRİŞ

### 1.1 Çalışmanın amacı ve hedefleri

Yapılan çalışmada, Windows 98 işletim sistemi altında, PC paralel portu kullanılarak, bir DC servo motorun pozisyon ve hız kontrolünün yapılmasını sağlayacak nesne tabanlı bir yazılım geliştirilmiştir. .

Yazılım kullanıcı ara yüzü ve donanım sürücüsü olmak üzere iki kısımdan oluşmaktadır. Kullanıcı ara yüzü nesne tabanlı bir yazılım olan Delphi’de yazılmıştır. Donanım sürücüsü ise Microsoftun ücretsiz kullanımına izin verdiği Windows98 DDK kullanılarak tasarlanmıştır. Geliştirilen yazılım aynı anda dört servo motorun kontrolüne imkan vermektedir.

Donanım sürücüsü kullanılmasının sebebi, Windows işletim sisteminin, program yetkilerini sınırlayan, katmanlı bir yapıda olmasıdır. Windows işletim sistemi üzerinde çalışan programlar, en az yetkinin olduğu ring3 olarak adlandırılan katmanda bulunurlar. Ring3 yetkileri kullanılarak bilgisayarı oluşturan donanım elemanlarının pek çok özelliği üzerinde değişiklik yapılamaz. Dolayısıyla Visual C++, Delphi gibi Windows üzerinde çalışan programlar kullanılarak pek çok donanım ayarı yapılamaz. Donanım sürücüler ise Windows işletim sisteminin ring0 olarak adlandırılan en geniş yetkilerin bulunduğu katmanında çalışırlar ve bilgisayarı

oluşturan donanım elemanlarının her türlü özellik ayarlarının yapılmasına olanak sağlarlar.

Yapılan çalışmada tasarlanan donanım sürücüsü yardımıyla, PC paralel portunun bir takım temel özellikleri değiştirilerek, paralel portun, yüksek hızda, güvenilir veri alabilen bir aygıtı dönüşümü sağlanmıştır. Paralel port üzerinde yapılan ayarlamalar, paralel portun kesme(interrupt) sinyalleri almasını sağlayacak ayarlamaların yapılması temeline dayanmaktadır. Kesme sinyalleri, PC işlemcisini uyarma özelliğine sahiptirler ve işlemci tarafından mutlaka işleme tâbii tutulurlar. Bu yöntem sayesinde motordan, yüksek hızda gelen pozisyon sinyalleri hatasız olarak elde edilebilmiştir.

Sadece paralel portun kesme sinyallerini almaya açılması, paralel porta gelen kesme sinyallerinin bilgisayar tarafından kesme sinyali olarak algılanmasına yeterli olmamaktadır. Paralel portun kesme sinyallerini almaya ayarlanmasının yanında, anakart üzerinde bulunan ve PIC(Programlanabilir kesme kontrol edici)adı verilen tümdevrenin de paralel port sinyallerini alacak şekilde programlanması gerekmektedir. Programlama işleminin nasıl yapılacağı 2.Bölümde ayrıntılı olarak açıklanmaktadır. Pentium Pro, Pentium II, Pentium III, Pentium IV ve Pentium Xeon işlemcilerde, çok işlemcili sistemleri desteklemek için ek olarak işlemci içine ayrıca gömülü olarak bulunan, yerel ileri programlanabilir kesme kontrol edici(LAPIC) bulunmaktadır. LAPIC yazılım yoluyla devre dışı bırakılabilmektedir. Bu çalışmada kullandığımız makinada Pentium 200MMX işlemci bulunmasından dolayı LAPIC bulunmamaktadır. Bundan dolayı tasarlanan yazılım LAPIC bulunmayan işlemci mimarisini destekleyecek şekilde tasarlanmıştır.

Tasarlanan yazılımın Windows uyumlu olması, nesne tabanlı kullanıcı arayüzleri tasarlanmasına imkan tanınması açısından önemlidir. Microsoft tarafından, Windows2000 DDK'nın henüz ücretsiz dağıtımı yapılmadığı için sistemin Windows 2000 uyumlu versiyonu henüz geliştirilememiştir.

Piyasada ki, PC tabanlı kontrolörler genellikle Win95 işletim sistemi altında çalışmaktadır. Bu da, Windows 2000 uyumlu sürücü geliştirmemizi ikinci plana atmamızın sebeplerinden birisidir. Buna bir örnek olarak, Epson firmasının

endüstriyel robotlar için geliştirdiği PC tabanlı bir kontrolör olan SRC500 gösterilebilir. Söz konusu kontrolörde de, bu çalışmada kullanılan bilgisayar gibi Pentium 200 işlemci kullanılmıştır ve sistem Windows 95 işletim sistemiyle çalışmaktadır. Bunun yanında SRC500 kontrolör veri iletimini 2 adet seri port üzerinden yapmakta, seçime bağlı olarak veri iletiminde ethernet kartı kullanımını desteklemektedir. Bunların dışında bir veri işleme kartı kullanmamaktadır. Bu veriler, bu çalışmada geliştirmiş olduğumuz sistemin, güncel teknolojiye yakın bir uygulama olduğunu göstermektedir.

Bir sistemin kontrolü yapılırken, sistemden gelen verilerin hassaslığı kadar, sisteme yollanan kontrol sinyallerinin hassaslığıda önemlidir. Delphi gibi yüksek seviyeli programlama dilleriyle kontrol algoritmalarında kullanabileceğimiz hassasiyette sinyaller üretemeyiz. Bu problemin çözümüne yönelik olarak sinyal üretiminde IBM tabanlı PC'lerdeki zamanlama işlemlerinde kullanılan, ve anakart üzerinde bulunan, 8253 zamanlayıcı tümdevresinden yararlanılmıştır. Bu zamanlayıcı programlanarak değişik darbe genişliklerinde sinyaller üretilmiştir. Bu şekilde üretilen sinyaller, motoru, PWM(darbe genişliği modülü) yöntemiyle kontrol edebilmemizi mümkün kılmıştır.

Paralel portan alabileceğimiz ve paralel porta verebileceğimiz gerilim-akım değerleri oldukça küçük değerlerdir. Paralel portun kontrol amaçlı kullanımında giriş-çıkış sinyallerinin güçlendirilmesi bir zorunluluktur. Bu amaçla tasarlanan güçlendirici, paralel port çıkışını 60W'a çıkarmanın yanında, çift yönde kontrolü sağlayan H-köprüsü mimarisi sayesinde, PWM kontrolü de etkin bir şekilde desteklemektedir. Devrede yapılacak bir takım değişikliklerle, daha yüksek çıkış değerlerine ulaşılması mümkündür.

Çalışmanın ana hedeflerinden birisi, günümüzde ki bilgisayarların ulaşmış oldukları yüksek kapasiteyi verimli bir şekilde kullanarak, daha önceki dönemlerde kullanılması çoğu zaman kaçınılmaz olan, veri işleme kartlarını kullanmadan, yüksek hassasiyette kontrol işleminin gerçekleştirilmesini sağlamaktır. Bunun en büyük faydası, bir takım sistemlerin, standart dizüstü bilgisayarlar ile kontrolünü sağlayacak olmasıdır. Bu ise sistemin, açık arazide, yada ulaşım araçlarında, ihtiyaç duyulması

muhtemel kontrol sistemi uygulamalarında kullanımını kolaylaştırmakta ve sistem maliyetlerini önemli ölçülerde düşürmektedir.

Ohio Üniversitesinden, LONG-BING BENSON[10]'un 1996 yılında yapmış olduğu çalışmada, PC ile veri toplamada izlediği yöntem ile bu çalışmada kullanılan yöntemle benzer özellikler taşımaktadır. Söz konusu çalışmada da dış ortamdaki veriler bilgisayar ortamına kesme sinyalleri şeklinde aktarılmakta ve bu işlem için donanım sürücüsü tanımlanmaktadır.

Yapılan yayın taramaları sonucunda ülkemizde windows uyumlu donanım sürücüsü tasarımı konusunda yayınlanmış bir çalışma bulunamamıştır. Bu sebeple bu çalışmanın hedeflerinden birisi de, windows altında sanal donanım aygıtı tanımlamanın temellerini ortaya koymak ve bu konuda uygulama geliştirmek isteyenlere bir başlangıç noktası olabilmektir.

Bu çalışmada DC motor kontrolünde, kontrolör olarak YSA kullanılması düşünülmüştür. Literatürde bu konuda yapılmış çok sayıda çalışma mevcuttur. K.Kadir ŞAFAK'ın yapmış olduğu çalışma[2] ve Weerasooriya ve El-Sharkawi'nin [4] yapmış oldukları çalışma, YSA kullanılarak bir DC motorun dinamik modelinin elde edilebildiğini gösteren örneklerdendir.

Motorun dinamik modellemesinde, iki katmanlı, gizli katmanında 16 nöron bulunan ileri beslemeli, geri dağılımlı bir YSA kullanılmıştır.

## **2 İNTERRUPT (KESME) SİNYALLERİ**

### **2.1 Kesme(İnterrupt) Sinyalleri**

Bilgisayarla veri alırken ve giriş-çıkış portlarındaki durum değişikliklerini incelerken, izlenen iki metod vardır. Bu metodlardan birisi portu taramaya dayanır. Bu metoda göre belirli zaman aralıklarında, portun durumunda herhangi bir değişiklik olup olmadığını kontrol ederiz. Bir değişiklik olduğunda, bu değişiklik için yazmış olduğumuz program parçası devreye girer. Bu yöntem, bilgisayarın işlemcisini çok meşgul eder. Bu metoda alternatif olan diğer metod ise, kesme sinyalleri ile veri almaktır. Bu durumda işlemci portu kontrol etmez. Porta bir kesme sinyali geldiğinde işlemci uyarılır. Bu metodla, daha hızlı ve güvenilir veri alınabilirken işlemci daha az meşgul edilmiş olur.

### **2.2 Donanım Kesme Sinyalleri**

Donanım kesme sinyallerini, programlanabilir kesme kontrol edici(PIC) değerlendirir. Çoğu bilgisayarda farklı adreslerde yerleşmiş iki ayrı PIC bulunur. Bunlardan birisi 0 ve 7 arası IRQ'ları değerlendirirken, diğeri 8 ve 15 arasındaki IRQ'ları değerlendirir. İkinci PIC, birinci PIC'e IRQ2 ile bağlıdır.

PIC'lerin bilgisayara tanıtımı BIOS tarafından yapılmaktadır. PIC'in IRQ'ları maskeleye yeteneği vardır. Maskelenen IRQ'ların sinyalleri işlemciye ulaşmaz.

Birinci PIC, 0x20h taban adresine yerleşmiştir ve 0-7 arası IRQ'ları kontrol eder. Birinci PIC'in kontrol bit değerleri aşağıdaki tabloda gösterilmiştir. Kontrol port adresi 0x21h'tır.

**Tablo 2.1.** PIC1 İşlem Kontrol Değerleri (0x21)

| Bit | Maskelenecek IRQ | Fonksiyon                 |
|-----|------------------|---------------------------|
| 7   | IRQ7             | Paralel Port              |
| 6   | IRQ6             | Floppy Disk Kontrol Edici |
| 5   | IRQ5             | Ses Kartı                 |
| 4   | IRQ4             | Seri Port                 |
| 3   | IRQ3             | Seri Port                 |
| 2   | IRQ2             | PIC2                      |
| 1   | IRQ1             | Klavye                    |
| 0   | IRQ0             | Sistem Saati              |

İkinci PIC 0xA0h taban adresine yerleşmiştir ve 8-15 arası IRQ 'ları kontrol eder. Aşağıda ikinci PIC'in ayarlamalarında kullanılan tablo görülmektedir.

**Tablo 2.2.** PIC2 İşlem Kontrol Değerleri (0xA1)

| Bit | Maskelenecek IRQ | Fonksiyon            |
|-----|------------------|----------------------|
| 7   | IRQ15            | Boş                  |
| 6   | IRQ14            | Sabit Sürücü         |
| 5   | IRQ13            | Matematik Ön işlemci |
| 4   | IRQ12            | PS/2 Fare            |
| 3   | IRQ11            | Boş                  |
| 2   | IRQ10            | Boş                  |
| 1   | IRQ9             | IRQ2 bağlantısı      |
| 0   | IRQ8             | Gerçek Zaman Saati   |

Yukarıdaki tablolar herhangi bir IRQ'yu maskelemek için gereken bitleri göstermektedir. Maskeyi kaldırmak için ilgili bitin tersi alınır. Mesela IRQ3'ün maskesini kaldırmak istersek PIC1'in kontrol portuna 0xF7 maske değerini yollamamız gerekir. Fakat bunu yaparken bazı durumları gözönüne almamız gerekir. Mesela daha önce maskeleyesi kaldırılmış bir IRQ'nun maskelenmesine sebep

olabiliriz. Bunun için önce port değerini okumalıyız. Daha sonra okuduğumuz port değerini ve maskeyi, AND mantıksal işlemcisiyle işleriz ve elde edilen değeri porta yollarız. Bu durumda, sadece ilgilendiğimiz IRQ'nun maskesini kaldırmış oluruz. Aşağıda bunun C kodu ile ifadesi görülebilir.

**outportb(0x21,(inportb(0x21)&0xF7));**      IRQ3'ün maskesinin kaldırılması

Aynı yöntem, işlemiz bittikten sonra, IRQ'nun tekrar maskelenmesi için de kullanılır. Fakat budurumda, OR mantıksal işlemcisi kullanılır ve kontrol portunun içeriği, 0x08 byte değeri ile OR işlemine tâbii tutulur. Bunun C koduna bakacak olursak;

**outportb(0x21,(inportb(0x21)|0x08));** IRQ3'ün maskelenmesi

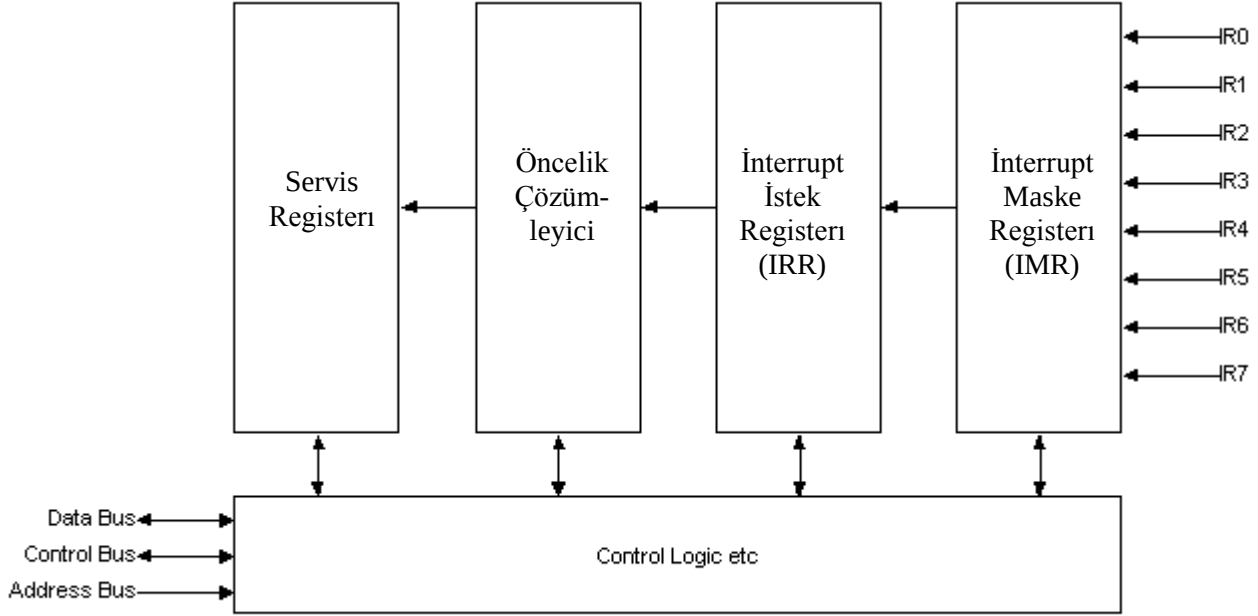
PIC ile ilgili bir diğer komut ise kesme sonu(EOI) dur. Bu komut, ISR tamamlandıktan sonra PIC'e yollanır ve ISR'nin tamamlandığını PIC'e anlatır. Bu komutu şöyle ifade ederiz;

**outportb(0x20,0x20);** PIC1 için

**outportb(0xA0,0x20);** PIC2 için

### 2.3 Programlanabilir Kesme Kontrol Edici (PIC)

Daha önce de bahsettiğimiz gibi, bilgisayarda kesme sinyalleri, iki adet 8259 programlanabilir kesme kontrol ediciyle yönetilir. **Pentium Pro, Pentium II, Pentium III, Pentium IV ve Pentium Xeon işlemcilerde, çok işlemcili sistemleri desteklemek için ek olarak işlemci içine ayrıca gömülü olarak bulunan, yerel ileri programlanabilir kesme kontrol edici(LAPIC) bulunmaktadır.** Bu çalışmada Pentium 200MMX işlecili bilgisayar kullanıldığı için sadece PIC anlatılmıştır. Aşağıda bir PIC'in ana şeması görülmektedir.



**Şekil 2.3.1** PIC blok diyagramı

PIC'in blok şemasını inceleyecek olursak, 8 adet kesme istek girişi, ilk olarak kesme maske registerı(IMR)'ye maskelenmiş olup olmadıklarının anlaşılması için gelir. Eğer maskelenmişlerse kesme ile ilgili daha fazla işlem yapılmaz. Ama eğer maskelenmemişlerse, kesme istek registerı (IRR) ile kesme istek bildirimlerini yaparlar.

IRR, işlemleri yerine getirilinceye kadar bütün IRQ'ları hafızasında tutar. Eğer ihtiyaç duyulursa, bu registerın değerleri, OCW3(Operational Control Word3)'ün uygun bitleri ayarlanarak, okunabilir. Öncelik Çözümleyici ise IRQ'lardan en yüksek önceliğe sahip olanı seçmek için kullanılır. IRQ numarası azaldıkça önceliği artar. Örnek olarak en yüksek öncelik IRQ0'ındır.

Bu noktada PIC, hangi IRQ'nun işleneceğine karar vermiştir ve işlemcinin ISR'yi yerine getirmek üzere uyarılmasının zamanı gelmiştir. Bu işlem, işlemcinin INT girişine sinyal yollanmasıyla gerçekleştirilir. Bu durumda işlemci o anda

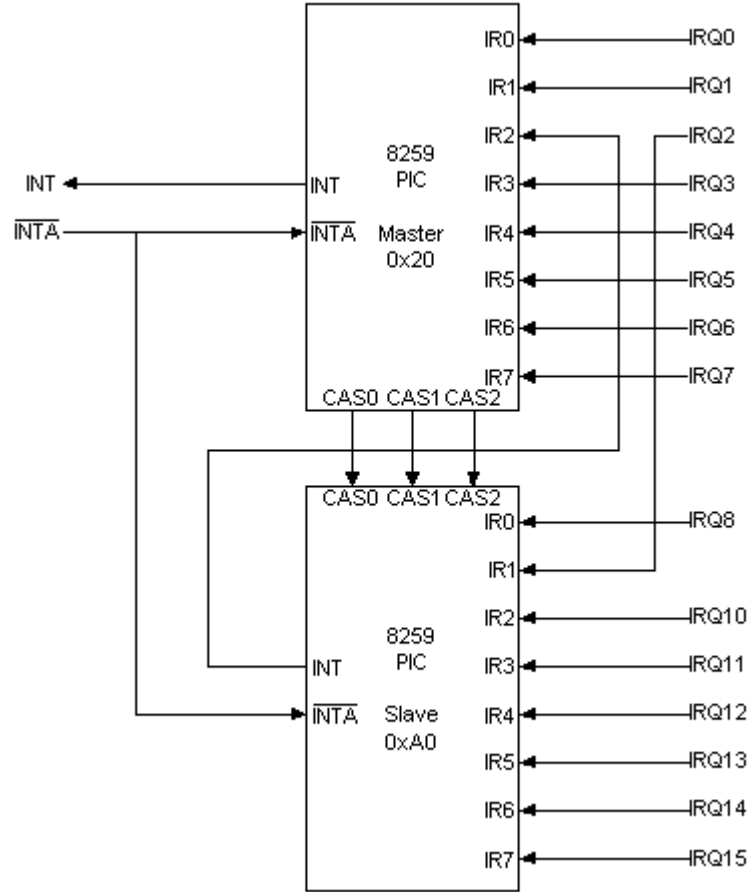
yapmakta olduđu işleme ara verecek, register değeriğini saklayacak ve hazır olunca INTA sinyaliyle PIC'i uyaracaktır.

İşlemciden INTA sinyali gelince PIC'in işlemekte olduđu IRQ, hangi IRQ'nun işlenmekte olduğunu belirten interrupt servis register(ISR)'ye aktarılacaktır. İşlemciden bir INTA sinyali daha gelir, buda PIC'in, ilgili IRQ numarasına karşılık gelen 8 bitlik bir pointer'ı, veri yoluna yollamasını sağlar. Eğer söz konusu IRQ, PIC2'ye ait ise bu durumda pointerı, veri yoluna, PIC2 yollar. Bu durumda ana PIC1, PIC2'yi pointerı veri yoluna yollaması için seçer. Bu pointerın yüksek anlamlı 5 bitlik kısmı, Initialization Command Word2(ICW2) kullanılarak ayarlanır. 5 bitlik bu kısma 00001 yazarsak PIC1, 01110 yazarsak PIC2 seçilmiş olur. Alçak değeri 3 bit ise hangi IRQ'nun seçildiğinin anlaşılması için kullanılır. Mesela IRQ3 servis isterse, 8 bitlik veri, yüksek anlamlı 5 bitlik 00001 ve alçak anlamlı 3 bitlik 011'in birleşimi olan 00001011 veya 0x0B olur. Bu değerin IRQ3'ün kesme vektöründeki karşılığı olduđu görülmektedir.

ISR yapması gereken işlemleri tamamladıktan sonra, PIC'e EOI komutunu verir. Bu durumda PIC interrupt servis registerını temizler ve sıradaki IRQ işleme sokulur. Eğer PIC2 ye ait bir IRQ'ya ait ISR işletilmişse EOI komutu her iki PIC'e de ayrı ayrı yollanmalıdır.

## 2.4 PIC Konfigürasyonu

İlk çıkan TX makinalarda tek PIC ve dolayısıyla 8 IRQ vardı. Zamanla 8 IRQ'nun yetmemesi üzerine bilgisayarlara 7 IRQ daha eklendi. Bu da, ikinci bir PIC'in sisteme eklenmesini beraberinde getirdi. İkili PIC konfigürasyonu aşağıdaki şekilde görülmektedir.



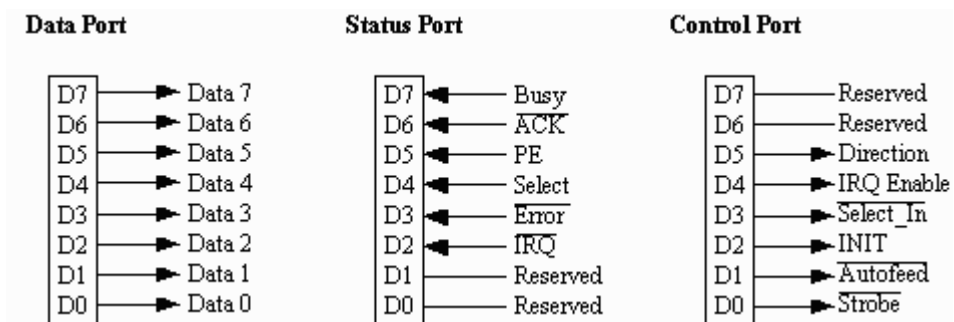
Şekil 2.4.1 PIC konfigürasyonu

İşlemcide sadece bir kesme sinyali girişi vardır, bundan dolayı PIC2, PIC1'e master/slave konfigürasyonu şeklinde bağlanmalıdır. IRQ2 bu amaçla seçilmiştir.

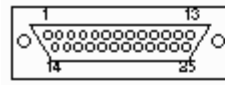
### 3. PARALEL PORT ÖZELLİKLERİ VE KESME SİNYALLERİ ALMAYA YÖNELİK AYARLARININ YAPILMASI

#### 3.1 Paralel Port Temel Özellikleri

Tüm yazıcı portları üç port adresinden oluşur. Bunlar; data, status ve control port adresleridir. Bu port adresleri birbirini takip eden sıradadır. Mesela data portunun adresi 0x0378h ise, status portunun adresi 0x0379h ve control portunun adresi 0x037Ah' tır.



Şekil 3.1.1 Paralel port portları



View is looking at  
Connector side of  
DB-25 Male Connector.

#### **Pin    Description**

|    |                                  |           |
|----|----------------------------------|-----------|
| 1  | $\overline{\text{Strobe}}$       | PC Output |
| 2  | Data 0                           | PC Output |
| 3  | Data 1                           | PC Output |
| 4  | Data 2                           | PC Output |
| 5  | Data 3                           | PC Output |
| 6  | Data 4                           | PC Output |
| 7  | Data 5                           | PC Output |
| 8  | Data 6                           | PC Output |
| 9  | Data 7                           | PC Output |
| 10 | $\overline{\text{ACK}}$          | PC Input  |
| 11 | Busy                             | PC Input  |
| 12 | Paper Empty                      | PC Input  |
| 13 | Select                           | PC Input  |
| 14 | $\overline{\text{Auto Feed}}$    | PC Output |
| 15 | Error                            | PC Input  |
| 16 | Initialize Printer               | PC Output |
| 17 | $\overline{\text{Select Input}}$ | PC Output |

Pin Assignments

Note: 8 Data Outputs  
4 Misc Other Outputs

5 Data Inputs

Note: Pins 18-25 are  
Ground

**Şekil 3.1.2** Paralel port giriş- çıkışları

Şekil 3.1.1’den görüldüğü gibi Data portta sekiz çıkış bulunmaktadır. Bunlara ek olarak Kontrol portunun ilk 4 pinini içeren /SELECT\_IN, INIT, /AUTOFEED ve /STROBE çıkıştır.

Data portta bulunan çıkışların tamamı doğrusal mantıklıdır, yani data porttaki bitlerden herhangi birine mantıksal bir vermemiz durumunda çıkışta bir olur. Ancak /SELECT\_IN, /AUTOFEED ve /STROBE çıkışları çevrilmiş mantıklıdır. Yani bu bitlerden birisine mantıksal bir yazmamız durumunda ilgili bitin çıkışı mantıksal sıfır olacaktır.

Status(durum )portunu gösteren şekle bakacak olursak, yazıcıdan gelen beş adet durum giriş pini görürüz. Bunlar BSY, /ACK, PE(Paper Empty), SELECT ve

ERROR'dur. SELECT pinindeki bir mantıksal değeri yazıcının hatta bağlı olduğuna işaret eder. BSY veya PE 'deki mantıksal bir değeri ise yazıcının meşgul olduğunu veya kağıtın tükendiğini belirtir. /ACK'daki sıfır mantıksal değeri yazıcıya herhangi bir bilgi ulaştığını belirtir. ERROR pinindeki bir sıfır ise yazıcıda bir hata oluştuğunu ifade eder.

Paralel portun elektriksel özelliklerine bakacak olursak, paralel portun çıkışlarından, 2.4V gerilim değerinde 2.6mA akım çekebiliriz ve paralel portun girişlerinden 4.2V gerilim değerinde 24mA akım verebiliriz.

Sonuç olarak, bir paralel portta, sekiz tanesi data portta, 4 tanesida kontrol portta olmak üzere toplam 12 çıkış ve tamamı status portta bulunan 5 giriş bulunmaktadır.

### **3.2 PARALEL PORTUN INTERRUPT(KESME) SİNYALLERİNE AÇILMASI**

Kesme sinyallerinin özelliği işlemciyi uyararak işlemcinin yaptığı işe ara verdirmeleri ve kendileriyle ilgili program parçacığının işletilmesini sağlamalarıdır. Bu mekanizma şöyle işler. Bilgisayarda 256 adet kesme sinyali bulunur. Tablo 1.1'i inceleyerek;

**Tablo 3.2.1** Interrupt Vektörleri Tablosu

| INT<br>(Hex) | IRQ                        | Genel Kullanım     |
|--------------|----------------------------|--------------------|
| 00 – 01      | Ayrıcalıklı durum kontrolü |                    |
| 02           | Non-Maskable IRQ           |                    |
| 03-07        | Ayrıcalıklı durum kontrolü |                    |
| 08           | IRQ0                       | Sistem Saati       |
| 09           | IRQ1                       | Klavye             |
| 0A           | IRQ2                       | Yönlendirme        |
| 0B           | IRQ3                       | COM2/COM4          |
| 0C           | IRQ4                       | COM1/COM3          |
| 0D           | IRQ5                       | Ses Kartı          |
| 0E           | IRQ6                       | Floppy Disk        |
| 0F           | IRQ7                       | Paralel Port       |
| 10-6F        | Yazılım İnt.               |                    |
| 70           | IRQ8                       | Gerçek Zaman Saati |
| 71           | IRQ9                       | Yönlendirme        |
| 72           | IRQ10                      | Boş                |
| 73           | IRQ11                      | Boş                |
| 74           | IRQ12                      | PS/2 Fare          |
| 75           | IRQ13                      | Math.işlemci       |
| 76           | IRQ14                      | SabitDisk          |
| 77           | IRQ15                      | Boş                |
| 78-FF        | Yazılım İnt.               |                    |

Bunların 16 adedi donanım kesme sinyali diğerleri ise yazılım kesme sinyalidir. Her kesme sinyalinin bir program parçacığı vardır. Bu program, ISR(Interrupt Service Routine) olarak adlandırılır ve kesme oluştuğunda bilgisayarın yapacağı işlemleri içerir. Kesme oluştuğunda, anakart üzerinde bulunan PIC (Programmable Interrupt Controler) denilen 8259 nolu çipe ulaşır, eğer sözkonusu kesme sinyali PIC çipinde enabled olarak tanımlanmışsa, PIC işlemcinin kesme sinyali girişine sinyal yollar ve ilgili ISR'nin adresini işlemciye aktarır. İşlemci o adrese gidip ilgili programı çalıştırarak kesme sinyalinin gereğini yapar ve daha sonra da kaldığı yerden eski işine döner.

Projede, paralel port kesme sinyali kullanılmıştır. Paralel porta her sinyal gelişinde programımızdaki sayaç değeri bir artacak şekilde ayarlandı. Bunun için Kesme sinyalleri vektöründe paralel port kesme sinyali olan IRQ7'nin işaret ettiği program adresi değiştirilerek, sayacın bulunduğu programın adresi verildi. Ayrıca

PIC IRQ7 kesme sinyallerinin alınmasını sağlayacak şekilde ayarlandı. Paralel portun kesme sinyallerini alması için yapılması gereken işlemleri özetleyecek olursak; Paralel portun, control portunda bulunan IRQEnable bitinin değeri mantıksal bire eşitlenir. Böylece status portun ACK girişindeki her 1 değerinden 0 değerine geçiş bir kesme sinyali olarak algılanır. Fakat bu, işlemcinin paralel porttan gelen kesme sinyallerini algılaması için yeterli olmaz, bir kesme sinyalinin işlemciyi tetikleyebilmesi için PIC 'in ilgili kesme ile ilgili bittinin enable yapılması gerekir. Bunun için PIC içerisinde IRQ7 kesme sinyaline karşılık gelen bitin değeri sıfıra eşitlenerek PIC, IRQ7 kesim sinyallerine açılmış olur. Bizim bu çalışmada kesme sinyallerini değiştirmek istememizdeki amaç, paralel portan yüksek hızda ve güvenilir veri alabilmektir. Paralel porttan veri elde etmede kullanabileceğimiz iki yol vardır. Bunlardan birisi, yazacağımız program içerisine yerleştireceğimiz bir döngü ile sürekli olarak paralel portaki ilgili giriş bitlerini okumak, bir diğeri ise bizim bu çalışmada yaptığımız gibi paralel portu kesme sinyallerine açarak, paralel porta gelen sinyallerin algılanabilmesini sağlamaktır.

Birinci durum paralel portun standart ayarlarına uygun olduğu için gerçekleştirilmesi oldukça kolay bir yöntemdir. Fakat bu yöntemde, paralel portun sürekli olarak kontrol edilmesi zorunluluğu ve güvenilir sonuç elde edilebilmesi için okuma hızının veri hızının 10 katı bir hızla olması gerekliliği, işlemciye çok fazla yük binmesine sebep olur ve başka işlemlerin gerçekleştirilmesi zorlaşır.

Bunun yanında bizim çalışmamızdaki veri hızının 25 kHz civarında olmasını hesap ettiğimizi göz önüne aldığımızda, güvenli veri okuma için işlemcinin 250 kHz hızıyla sürekli olarak paralel portu kontrol etmesinin gerektiği görülür. Bu hıza, Delphi gibi yüksek seviyeli dillerin kullanılması durumunda yazılım yoluyla ulaşılması, teknik olarak oldukça güçtür. Bilgisayarın tek işinin paralel portu okumak olmadığını göz önüne de alacak olursak bu yolla elde edilecek bilgilere güvenilemeyeceği açıktır.

Verilerin, kesme sinyalleri şeklinde okunması durumunda ise işlemci paralel portu hiç kontrol etmez. Bir kesme sinyali gelmesi durumunda işlemci uyarılarak yaptığı işe ara verir ve ilgili kesme sinyalinin ISR'sini yerine getirdikten sonra tekrar

kaldığı yerden eski işine geri döner. Bu yolla hem işlemcinin üzerine binen yük azaltılmış olur, hem de daha güvenilir veri elde edilmiş olur.

Biz bu çalışmada, bilgisayarımızın paralel portundan yolladığımız sinyalleri bir güçlendirici yardımıyla güçlendirdikten sonra bir dc motora uyguladık ve motorda meydana gelen açı değişimini de, bir sensor yardımıyla kesme sinyalleri şeklinde tekrar paralel porttan aldık. Kesme sinyalleriyle, motorun dönme açı değişim miktarını tespit edebilmek için Kesme vektörleri tablosunda bulunan IRQ7 kesme vektörünü kendi yazdığımız ve gelen sinyalleri sayma işlevini yerine getiren bir ISR ile değiştirdik. Bu değişiklikleri Windows 98 işletim sistemi ortamında çalışan nesnel tabanlı bir programlama dili olan Delphi 5'i kullanarak yapmak istediğimizden paralel portun özelliklerini istediğimiz şekle getiren bir Windows98 donanım sürücüsü geliştirdik. Paralel port değişikliklerini yapabilmek için donanım sürücüsü tanımlamamızın sebeplerini ve Windows işletim sistemi ve donanım tanımlamayla ilgili detaylı bilgileri takip eden bölümde bulabilirsiniz.

## **4. WINDOWS 9x İŞLETİM SİSTEMİ İÇİN SANAL SÜRÜCÜ TASARIMI**

### **4.1 Windows İşletim Sistemi Hakkında Genel Bilgiler**

Windows işletim sisteminin temel özelliklerinden birisi aynı anda birden fazla programı çalıştırabilmesidir. Bu, Windows işletim sisteminin, donanım aygıtlarını programlar arasında paylaşırabilmesiyle mümkün olmaktadır. Windows piyasaya ilk çıktığında, piyasada pekçok başarılı Dos programı kullanılmaktaydı. Windows'un yaygınlaşabilmesi için Dos programlarını da desteklemesi bir zorunluluktur. Dos programlarının, Windows programlarından farklı olarak, çalıştıkları sürece tüm donanım aygıtlarını tek başlarına kullanmaları, aynı anda birden fazla programın çalışmasına izin vermez. Bundan dolayı, Windows işletim sisteminin Dos programlarını da destekleyebilmesi için Sanal Makina(virtual machine) kavramı geliştirildi. Sanal makina, makinayı oluşturan elemanların(İşlemci, monitör, klavye gibi) yazılım yoluyla sanal olarak tanımlanmasıdır. Programlar bu sanal makina içinde, gerçek makina içinde çalışıyormuşçasına çalışırlar. Buna göre, Dos programları sanal bir 8086 makinada çalışırken, Windows programları da Sistem Sanal Makinası(System Virtual Machine) denilen ayrı bir sanal makinada çalışır. Windows dönüşümlü olarak her iki sanal makinaya da işlemci(CPU)'yi kullanma süresi verir. Böylece aynı anda birden fazla uygulamanın çalışması sorunsuz bir şekilde sağlanmış olur.

## 4.2 VMM - Sanal Makina Yöneticisi (VIRTUAL MACHINE MANAGER)

VMM 32 bitlik bir korumalı mod programıdır. Ana görevi, sanal makinaı destekleyen çerçevenin oluşturulması ve hatalara karşı korunmasıdır. Bunun yanında, sanal makinaların oluşturulması, çalıştırılması ve kaldırılmasından sorumludur. Makinaya Windows 95'in yüklenme aşamalarını incelersek;

1. io.sys hafızaya yüklenir
2. config.sys ve autoexec.bat işlemleri gerçekleştirilir.
3. Win.com çağrılır.
4. Win.com, VMM32.VXD dosyasını çalıştırır.
5. VMM32.VXD, XMS sürücüyü kullanarak VMM'yi artırılmış(extended) hafızaya yükler
6. VMM kendisini ve diğer sanal cihaz sürücülerini tanıtır.
7. VMM makinayı korumalı moda geçirir ve sistem sanal makinasını oluşturur.
8. En son yüklenen Sanal Shell cihazı, krnl386.exe yi çalıştırarak sistem sanal makinasında Windows'u başlatır.
9. Krnl386.exe Windows 95 shell'indeki tüm diğer dosyaları yükler.

## 4.3 Sanal Cihaz Sürücü (VXD)

Intel 80386'dan itibaren işlemcilerde yetki katmanları uygulanmaya başlanmıştır. Ring0, Ring1, Ring2 ve Ring3 olmak üzere toplam dört yetki katmanı vardır. En fazla yetki Ring0 katmanında, en az yetki ise Ring3 katmanında bulunur. Bir işlemci her hangi bir programın hangi yetki seviyesinde olduğunu yetki seviyesini belirleyici registerlar yardımıyla belirler. Windows programlarının tamamı ring3'te çalışır. Dolayısıyla windows üzerinde yazacağımız programlar ancak ring3 yetkilerini kullanabilirler. Bundan dolayı, DOS altında çalışan C gibi programlarla işlemcinin

tüm özelliklerini kullanabilirken, Windows altında çalışan yeni sürüm C, Delphi gibi programlarda pek çok işlemi gerçekleştiremeyiz. Bu sorunu çözmek için sanal sürücülerden yararlanılır. Sanal cihaz sürücüler, donanımların yazılım eşdeğerleri olarak tanımlanabilir ve bunların tanımlanması ring0 yetki seviyesindeki yetkilerin kullanılmasını gerektireceğinden, sanal cihaz sürücüler ring0 da çalışan program parçacıklarıdır ve VXD (Virtual x Driver) ile ifade edilirler. Aradaki x sürücü hangi cihazı tanımlıyor ise (keyboard, mouse, monitor, CD vb. ) ona karşılık gelir. VxD'ler iki tiptir.

- Statik
- Dinamik

Statik VxD'ler işletim sisteminin makineye yüklendiği sırada yüklenir ve sistem kapatılıncaya kadar yüklü kalır. Dinamik VxD'ler ihtiyaç duyulduğunda yüklenir, daha sonra da kaldırılabilir. Dinamik VxD'ler ring3'te çalışan win32 uygulamaları içerisinden yüklenip kaldırılabilir.

#### **4.4 VxD'ler arasında iletişim**

VxD'ler birbirleriyle iletişimi üç mekanizma vasıtasıyla kurarlar

- Kontrol Mesajları
- Servis API'leri
- Callback Fonksiyonları

##### **4.4.1 Kontrol Mesajları**

Bazı özel durumlar oluştuğunda, VMM sistemde yüklü tüm VxD'lere sistem kontrol mesajları yollar. Kontrol mesajları ring-3 windows uygulamalarında ki, windows mesajlarına benzerdir. Her VxD'nin, cihaz kontrol prosedürü adı verilen ve

mesajları alıp, işleyen bir fonksiyonu vardır. Sistem kontrol mesajlarına ek olarak, bir VxD, diğer VxD'ler ile iletişim kurmak için, kendi kontrol mesajlarını da tanımlayabilir

#### 4.4.2 Servis API'leri

VMM'yi içeren VxD'ler, diğer VxD'ler tarafından çağrılabilen bir takım fonksiyonlar içerirler. Bu fonksiyonlar VxD servisleri olarak adlandırılırlar. Bu VxD servislerinin çağrılma mekanizması, diğer Ring-3 uygulamalarındaki çağırma mekanizmalarından farklıdır. VxD servisi sağlayan her VxD'nin kendine has 16 bitlik bir ID numarası olması gerekir. Bu ID numaralarının ne olduğu Microsoft'tan öğrenilebilir. Örnek olarak;

|                            |                   |
|----------------------------|-------------------|
| <b>UNDEFINED_DEVICE_ID</b> | <b>EQU 00000H</b> |
| <b>VMM_DEVICE_ID</b>       | <b>EQU 00001H</b> |
| <b>DEBUG_DEVICE_ID</b>     | <b>EQU 00002H</b> |
| <b>VPICD_DEVICE_ID</b>     | <b>EQU 00003H</b> |
| <b>VDMAD_DEVICE_ID</b>     | <b>EQU 00004H</b> |
| <b>VTD_DEVICE_ID</b>       | <b>EQU 00005H</b> |

Yukarıdaki ifadelerden de görüldüğü gibi VMM'nin ID numarası 1, VPICD'in ID numarası 3'tür. VMM bu ID numaralarını istenen VxD servisini sunan VxD'yi tesbit etmek için kullanır. Kullanılmak istenen servis, ayrıca ilgili VxD'nin servis indeksinden seçilmelidir. Bir VxD sunduğu servislerin adreslerini bir tabloda saklar. VxD servisleri çağırmanın ana mekanizması int 20h interrupt komutuna dayanır. Kodunuz int 20h'tan sonra gelen ve cihaz ID numarası ile servis indeksini içeren bir dword değerini işler. Mesela cihaz ID numarası 000dh olan VxD'nin sunduğu 1 numaralı servisi çağırarak istersek;

```
Int 20h  
dd 000D0001h
```

Burada int 20h komutundan sonra gelen dword ifadenin yüksek anlamlı kısmı Cihaz ID sini içerirken, düşük değerli kısım servis indeksini içerir.

#### **4.4.3 Callback Fonksiyonları**

Callback fonksiyonları diğer VxD'ler tarafından da çağırılabilen VxD fonksiyonlarıdır. Callback'lerin VxD servislerinden farkı public fonksiyonlar olmamalarıdır.

#### **4.5 Virtual Machine Manager (SANAL MAKİNA YÖNETİCİ)**

Sanal makina yönetici (VMM), Windows 9x'in arkasındaki asıl işletim sistemidir. Sanal makinayı yönetmek için gerekli olan çerçeve yapısını kurup, denetlemenin yanında diğer VxD'ler için de pek çok önemli servis sağlar. Sağladığı üç servis şunlardır.

- Bellek yönetimi
- Interrupt kontrolü
- Uygulama zamanlamalarının ayarlanması

##### **4.5.1 Bellek Yönetimi**

VMM, sistem sanal adres uzayını oluşturmak için Intel 80386 ve sonrası işlemcilerin 32-bit bölümlere ayırma yeteneğini kullanır ve adres uzayını birbirinden farklı dört alana böler. Bunlar;

- **V86 bölgesi** 0h-10FFEFh aralığını kaplar. Mevcut çalışmakta olan Sanal makinanın kullandığı alandır.
- **Özel uygulama bölgesi** 4Mb'tan başlar 2Gb'a kadar uzanır. Bu alan Win32 uygulamalarının çalıştığı alandır. Her win32 uygulamasının (2Gb – 4Mb)'lık bir alanı vardır.
- **Paylaşılmış uygulama bölgesi** 2Gb – 3Gb adres aralığını kaplar. Bu alan sistem sanal makinasındaki tüm uygulamalar tarafından paylaşılır. Bu bölge user32, kernel32, ve gdi32 gibi sistem DLL'lerinin bulunduğu kısımdır. Bunun yanında Win16 uygulamalarında bu kısımda çalışır.
- **Paylaşılmış sistem bölgesi** 3Gb – 4Gb adres aralığını kaplar. VxD ve VMM'ler bu bölgede bulunur.

VMM, VxD'lere üç ayrı bellek servisi sağlar. Bunlar;

- **Sayfa tabanlı bellek servisleri:** Bu tip servisler belleği 4Kb'lık sayfalar halinde düzenler. Mevcut olan en alt seviyeli bellek servisleridir. Tüm diğer bellek servisleri, sayfa tabanlı bellek servislerini temel alırlar.
- **Blok bellek servisleri:** Daha küçük bellek blokları oluşturur. Sayfa tabanlı bellek servisleri temel alan bir üst seviyeli bellek yönetim servisi.
- **Liste servisleri:** Bağlı listeler oluşturmaya uygun sabit boyutlu bellek blokları oluşturur.

## 4.6 İnterrupt Kontrolü

Korumalı modda interruptlar, Interrupt Tanımlayıcı Tablo(Interrupt Descriptor Table(IDT))'ya yönlendirilirler. VMM, sanal makinanın IDT'lerini VxD'ler yardımıyla kontrol eder. VMM, birinci seviyede interrupt kontrolü sağlar, interrupt oluşması durumunda o anda çalışmakta olan programın register değerlerini yığın(stack)da saklar ve asıl işlemin gerçekleştirilmesi için interrupt kontrolünü ikinci seviye interrupt kontrolü yapabilecek çeşitli VxD'lere devreder. İkinci derece interrupt kontrol edici işlemi tamamladığında, kontrolü, register değerlerini interrupt sinyalinden önceki duruma getirecek olan sonlandırıcı programa devreder ve program interrupttan önce kaldığı yerden devam eder. VxD'ler interrupt kontrol edicileri, Set\_PM\_Int veya Hook\_V86\_Int\_Chain gibi VMM servisleri yardımıyla yükleyebilirler.

## 4.7 Uygulama zamanlamalarının ayarlanması

VMM, sanal makinalar ve uygulamalar arasında aynı anda birden fazla uygulama işlevini yerine getirebilmek için iki zamanlama ayarlayıcı bileşen kullanır. Bunlar;

- Birincil zamanlayıcı
- Zaman parçalayıcı veya ikincil zamanlayıcıdır.

Birincil zamanlayıcının görevi çalışma önceliği en büyük olan uygulamayı belirlemektir. Bu seçim VMM bir interrupt'a servis vereceği zaman olur. Seçimin sonucunda interrupt gerçekleştikten sonra kontrolün hangi uygulamaya geçeceği belirlenir.

İkincil zamanlayıcı CPU zamanını uygulamalar arasında paylaştırır, bunu yaparken biricil zamanlayıcının servislerini kullanır. Bir uygulama için ayrılan süre dolduğunda, ikincil zamanlayıcı sırası gelen uygulamanın öncelik derecesini artırır, böylece biricil zamanlayıcı sırası gelen uygulamayı çalıştırılmak üzere seçer.

#### 4.8 Sanal Cihaz Sürücüsü İskelet Yapısı

VxD sürücüsü yazmak ve derlemek için Win98 DDK (Device Development Kit) ve MASM32'den yararlanıldı. Aşağıda VxD'nin genel yapısı ve VxD yazımı ile ilgili temel bilgiler anlatılmaktadır.

##### 4.8.1 LE Dosya yapısı

VxD doğrudan çalıştırılabilir dosya formatı(LE) kullanır. Bu OS/2'nin 2. sürümü için geliştirilen dosya formatıdır. Bu format, VxD'nin ihtiyaç duyduğu 16 ve 32 bitlik kodları saklayabilir. Windows 3.x'ten Windows NT'ye kadar olan windows işletim sistemleri bilgisayarı dos üzerinden başlatır. Bu sebeble, korumalı moda geçinceye kadar 16 bitlik kod kullanımı gerekir. Fakat Windows NT'nin böyle bir sorunu yoktur ve Windows NT'de LE yerine PE dosya formatı kullanılır.

LE dosya yapısında kod ve veriler farklı kullanım özelliklerine göre değişik segmentlerde saklanırlar. Aşağıda gerekli olan segment sınıfları görülebilir.

- **LCODE (Page-locked code):** Bu segment hafızada sürekli bulunması gereken kod ve veriler için kullanılır. Özellikle donanım interrupt kontrolörleri bu segmentte bulunur.

- **PCODE (Pageable code):** Bu segment hafızada sürekli bulunması zorunlu olmayan kod ve veriler için kullanılır. VMM belleğe ihtiyaç duyarsa bu segmenti sabit diske atar.
- **ICODE (Initialization-only code)** Bu segmentteki kod sadece VxD'nin tanıtılması sırasında kullanılır. Tanıtım işlemi tanımlandıktan sonra bu segment bellekte yer açmak için VMM tarafından kaldırılır.
- **DBOCODE (Debug-only code)** Bu segmentteki kod ve veri VxD, debugger altında çalıştığında kullanılır.
- **SCODE (Static code and data)** Bu segment VxD kaldırıldığında sürekli bellekte kalır. Bu segment, sürekli olarak yüklenip kaldırılan ve bir önceki yüklenişindeki verileri korumak isteyen dinamik VxD'ler için de kullanışlıdır.
- **RCODE (Real-mode initialization code and data):** Bu segment 16 bitlik asıl modda tanıtım işlemleri için kullanılır.
- **16ICODE (Protected-mode initialization data):** Bu 16 bitlik segment VxD'nin korumalı moddan, V86 moduna aktaracağı kodları içerir.
- **MCODE (Locked message strings):** Bu segment, VMM'nin mesaj macroları yardımıyla derlenebilen mesaj verilerini içerir, bu da yazdığınız sürücünün uluslararası sürümlerini oluşturmanıza yardımcı olur.

Bir VxD'de içerisinde bu segmentlerin tümünün bulunması gerekmez, sadece ihtiyaç duyulan segmentler kullanılır. Segmentler yukarıda anlatılan sınıflara göre bildirilmelidir. Segment bildirimleri .def uzantılı modül tanımlama dosyalarıyla yapılır. Aşağıda tüm modülleri içeren bir bildirim dosyası görülmektedir.

VXD **VxDAd1**

SEGMENTS

\_LPTEXT CLASS 'LCODE' PRELOAD NONDISCARDABLE  
\_LTEXT CLASS 'LCODE' PRELOAD NONDISCARDABLE  
\_LDATA CLASS 'LCODE' PRELOAD NONDISCARDABLE  
\_TEXT CLASS 'LCODE' PRELOAD NONDISCARDABLE  
\_DATA CLASS 'LCODE' PRELOAD NONDISCARDABLE  
CONST CLASS 'LCODE' PRELOAD NONDISCARDABLE  
\_TLS CLASS 'LCODE' PRELOAD NONDISCARDABLE  
\_BSS CLASS 'LCODE' PRELOAD NONDISCARDABLE  
\_LMGTABLE CLASS 'MCODE' PRELOAD NONDISCARDABLE IOPL  
\_LMSGDATA CLASS 'MCODE' PRELOAD NONDISCARDABLE IOPL  
\_IMSGTABLE CLASS 'MCODE' PRELOAD DISCARDABLE IOPL  
\_IMSGDATA CLASS 'MCODE' PRELOAD DISCARDABLE IOPL  
\_ITEXT CLASS 'ICODE' DISCARDABLE  
\_IDATA CLASS 'ICODE' DISCARDABLE  
\_PTEXT CLASS 'PCODE' NONDISCARDABLE  
\_PMSGTABLE CLASS 'MCODE' NONDISCARDABLE IOPL  
\_PMSGDATA CLASS 'MCODE' NONDISCARDABLE IOPL  
\_PDATA CLASS 'PDATA' NONDISCARDABLE SHARED  
\_STEXT CLASS 'SCODE' RESIDENT  
\_SDATA CLASS 'SCODE' RESIDENT  
\_DBOSTART CLASS 'DBOCODE' PRELOAD NONDISCARDABLE  
CONFORMING  
\_DBOCODE CLASS 'DBOCODE' PRELOAD NONDISCARDABLE  
CONFORMING  
\_DBODATA CLASS 'DBOCODE' PRELOAD NONDISCARDABLE  
CONFORMING  
\_16ICODE CLASS '16ICODE' PRELOAD DISCARDABLE  
\_RCODE CLASS 'RCODE'

EXPORTS

**VxDAd1**\_DDB @1

İlk satır VxD adının bildirimidir. VxD adı mutlaka büyük harflerle olmalıdır. Diğer satırlar segment bildirimleridir. Bildirimler üç kısımdan oluşur; segment adı, segment sınıfı ve segmentin istenen çalışma âni özellikleri. Yukarıdaki örnekten görülebileceği gibi **\_LPTEXT**, **\_LTEXT**, **\_LDATA** gibi aynı segment sınıfına ve özelliklere ait segmentler tanımlanabilir. Segmentlerin bu şekilde bildirilmesi kodun daha anlaşılır olmasını sağlar. Mesela, programlayıcı verileri **\_LDATA** segmentinde saklarken, kodları **\_LTEXT** segmentinde saklar. Sonuçta program derlenip .exe dosyası oluşturulduğunda ikiside tek bir segmentte birleştirilir. VxD, dışarıya sadece tek bir sembol yollar, bu cihaz tanımlama bloğu(Device Descriptor Block(DDb))dir. Gerçekte DDB, VMM'nin VxD hakkında bilmesi gerekenleri içeren bir yapıdır. DDB, modül tanımlama dosyası(.def)nda belirtilmelidir.

Vmm.inc, segmenleri kaynak dosyasında bildirmek için pek çok makro içerir, bu makroların bir kısmı aşağıdaki tabloda verilmiştir

**Tablo 4.8.1** Segment bildirim makroları

|                 |                                |
|-----------------|--------------------------------|
| <b>LTEXT</b>    | <b>VxD_LOCKED_CODE_SEG</b>     |
| <b>PTEXT</b>    | <b>VxD_PAGEABLE_CODE_SEG</b>   |
| <b>DBOCODE</b>  | <b>VxD_DEBUG_ONLY_CODE_SEG</b> |
| <b>ITEXT</b>    | <b>VxD_INIT_CODE_SEG</b>       |
| <b>LDATA</b>    | <b>VxD_LOCKED_DATA_SEG</b>     |
| <b>IDATA</b>    | <b>VxD_IDATA_SEG</b>           |
| <b>PDATA</b>    | <b>VxD_PAGEABLE_DATA_SEG</b>   |
| <b>STEXT</b>    | <b>VxD_STATIC_CODE_SEG</b>     |
| <b>SDATA</b>    | <b>VxD_STATIC_DATA_SEG</b>     |
| <b>DBODATA</b>  | <b>VxD_DEBUG_ONLY_DATA_SEG</b> |
| <b>_16ICODE</b> | <b>VxD_16BIT_INIT_SEG</b>      |
| <b>RCODE</b>    | <b>VxD_REAL_INIT_SEG</b>       |

Her macronun sonlandırılması gerekir, mesela kaynak dosyamızda bir **\_LTEXT** segmenti tanımlarsak, bunu aşağıdaki gibi ifade etmemiz gerekir.

```

VxD_LOCKED_CODE_SEG
    < Kodlar buraya yazılacak >
VxD_LOCKED_CODE_ENDS

```

#### 4.8.2 VxD İskeleti

VxD programlarında makrolar yoğun bir şekilde kullanılır. Makroların içerikleri include dosyalarından öğrenilebilir. Aşağıda temel bir VxD yazılımının bölümleri görülebilir.

```
.386p  
include vmm.inc  
  
DECLARE_VIRTUAL_DEVICE FIRSTVXD,1,0,  
FIRSTVXD_Control, UNDEFINED_DEVICE_ID,  
UNDEFINED_INIT_ORDER  
  
Begin_control_dispatch FIRSTVXD  
End_control_dispatch FIRSTVXD  
  
End
```

Kodu adım adım inceleyecek olursak;

**.386p:** assembler derleyicisine 80386 işlemcinin, ayrıcalıklı CPU komutlarını kullanacağımızı belirtir.

**.include vmm.inc:** kullanacağımız makroların tanımlamalarını içerir. İhtiyaç duyulan başka makroların include dosyalarıda eklenir.

```
DECLARE_VIRTUAL_DEVICE FIRSTVXD,1,0,  
FIRSTVXD_Control, UNDEFINED_DEVICE_ID,  
UNDEFINED_INIT_ORDER:
```

Daha öncede belirtildiği gibi, VMM, VxD hakkında bilmesi gerekenleri VxD'nin cihaz tanımlama bloğu(DDB)ndan alır. Cihaz tanımlama bloğu VxD'nin adını, cihaz ID'sini, ve VxD servislerinin giriş noktaları gibi bilgileri içeren bir yapıdır. Vmm.inc içerisinde bizim için bu yapıyı oluşturan makro **DECLARE\_VIRTUAL\_DEVICE** makrosudur. Bu makronun yapısı şöyledir;

**Declare\_Virtual\_Device** *Name, MajorVer, MinorVer , CtrlProc, DeviceID, InitOrder, V86Proc, PMProc, RefData*

VxD kaynak kodunda kullanılan terimler büyük yada küçük harf olabilirler, parametreleri tek tek incelersek;

- **Name:** VxD'nin ismi. En fazla 8 karakter olabilir. Büyük harflerle olmak zorundadır. İsim system içindeki VxD isimleri içinde tek olmalıdır. Macro, DDB ismini de *Name\_DBB* şeklinde kendisi üretir.
- **MajorVer** ve **MinorVer:** VxD'nin versiyon parametreleri
- **CtrlProc:** VxD'nin cihaz kontrol prosedürünün ismidir. Cihaz kontrol prosedürü, bir VxD için kontrol mesajlarını alan ve işleyen fonksiyondur. Cihaz kontrol prosedürümüzü oluşturmak için **Begin\_Control\_Dispatch** macrosunu kullanırız.
- **DeviceID:** VxD'nin 16 bitlik tekil tanımlayıcısıdır. Yalnızca VxD aşağıdaki durumları ele alıyorsa gereklidir.
  - ◆ VxD başka VxD'lerin kullanımı için servis sunuyorsa.
  - ◆ VxD'yi yüklemek için int2Fh, fonksiyon 1605h'i kullanan bir gerçek mod yazılımı(TSR) olması halinde.

Bu haller mevcut değilse, tekil bir cihaz ID'sine ihtiyaç yok demektir, ve bu alana **UNDEFINED\_DEVICE\_ID** yazılır.

- **InitOrder:** Tanıtım yada yükleme sırasıdır. VMM, VxD'leri belirtilen sırada yükler. Mesela;

|                            |                      |
|----------------------------|----------------------|
| <b>VMM_INIT_ORDER</b>      | <b>EQU 00000000H</b> |
| <b>DEBUG_INIT_ORDER</b>    | <b>EQU 00000000H</b> |
| <b>DEBUGCMD_INIT_ORDER</b> | <b>EQU 00000000H</b> |
| <b>PERF_INIT_ORDER</b>     | <b>EQU 00090000H</b> |
| <b>APM_INIT_ORDER</b>      | <b>EQU 00100000H</b> |

Görüldüğü gibi **VMM**, **DEBUG** ve **DEBUGCMD** ilk yüklenen VxDler olurken, **PERF** ve **APM** onları takip etmiştir. Yükleme sırası değeri en düşük olan VxD, önce yüklenmektedir. Eğer VxD başka VxD'lerin sunduğu servislere ihtiyaç duyuyorsa, servisine ihtiyaç duyduğu VxD'den sonra yüklenmelidir. Eğer VxD'nin yüklenme sırası önemli değilse bu alana **UNDEFINED\_INIT\_ORDER** yazılır.

- **V86Proc** ve **PMProc**: VxD, V86 ve korumalı mod programları için API sağlayabilir. **V86Proc** ve **PMProc** bu API'lerin adreslerini belirtir.

**Begin\_Control\_Dispatch FIRSTVXD** ve **End\_Control\_Dispatch**  
**FIRSTVXD**: Bu makrolar, VxD için kontrol mesajı geldiğinde, VMM tarafından çağırılan cihaz kontrol prosedürlerini tanımlarlar. Makro, verilen isme **Control** ifadesini ilave eder. Burada dikkat edilecek husus, bu isimle **Declare\_virtual\_device** makrosunun **CtrlProc** alanında tanımlanan ismin aynı olmasıdır. Bu makro VxD yüklü olduğu sürece bellekte sabit duran **VxD\_LOCKED\_CODE\_SEG** içerisinde kullanılmak zorundadır. Örnekte

verilen cihaz kontrol prosedürü herhangi bir işlem gerçekleştirmez. Prosedürün bir işlem gerçekleştirmesi için VxD'nin hangi control mesajlarıyla ilgileneceği ve bunları işleme sokmak için hangi fonksiyonlardan yararlanacağı belirtilmelidir. Bu amaçla **Control\_Dispatch** makrosu kullanılır. Bu makronun yapısı şöyledir;

**Control\_Dispatch** *kontrol mesajı, fonksiyon*

Mesela, VxD sadece **Device\_Init** mesajıyla ilgilenecek olursa;

```
Begin_Control_Dispatch FIRSTVXD  
    Control_Dispatch Device_Init, OnDeviceInit  
End_Control_Dispatch FIRSTVXD
```

Burada, **OnDeviceInit**, **Device\_Init** mesajının alınması durumunda yapılacak işlemin fonksiyonunun ismidir.

VxD kaynak kodu **end** direktifiyle sona erdirilir.

#### 4.9 VxD'nin Derlenmesi

VxD kaynak kodlarının derlenmesinin, normal Win32 kaynak kodlarının derlenmesinden temel farkı, ml.exe ve link.exe programlarında kullanılan komutlardan kaynaklanmaktadır. VxD kaynak kodları derlenirken, ml.exe ve link.exe programların da kullanılan komutların önemli bir kısmı aşağıda bir örnek üzerinde açıklanmaktadır.

```
ml -coff -c -Cx -DASM6 -DBLD_COFF -DIS_32 firstvxd.asm
```

**-coff** coff nesne dosyası formatının kullanılacağını belirtir.

**-c** Sadece derle demektir. Obj uzantılı nesne dosyasının link.exe programı çalıştırılarak bağlanması önler. Link.exe'yi daha sonra çeşitli parametrelerle çağırmanız gerektiğinde kullanılır.

**-Cx** iç ve dış etiketlerin büyük küçük harf özelliklerini korur.

**-D<text>** Bir text makrosu tanımlar. Mesela **-DBLD\_COFF**'taki, **BLD\_COFF** conditional assembly'de kullanılan bir text makrosudur.

**link -vxd -def:firstvxd.def firstvxd.obj**

**-vxd** .obj uzantılı nesne dosyasından vxd dosyası üretmek istediğimizi ifade eder.

**-def:<.DEF dosyası>** VxD'nin modül tanım dosyasının ismini tanımlamak için kullanılır.

#### **4.10 Nesne Tabanlı Yazılım ile VxD'nin Dinamik Olarak Yüklenmesi ve Kaldırılması**

Nesne tabanlı yazılımlar, Windows API'lerine dayalı olarak çalışmaktadır. Bu çalışmada, nesne tabanlı yazılım olarak Delphi5 kullanılmıştır. Delphi 5 içerisinde Windows API'leri doğrudan kullanılabilir.

Bir VxD dosyasının dinamik olarak yüklenip kaldırılabilmesi için öncelikle VxD dosyasının Windows sistem klasörüne yerleştirilmesi gerekir. Daha sonra eğer VxD'yi yüklemek istersek **CreateFile** API'sini kullanırız.. **CreateFile** API'sine bakacak olursak, bu API'nin bazı parametreleri olduğunu görürüz.

**CreateFile(VxDAdresi,0,0,nil,0,FILE\_FLAG\_DELETE\_ON\_CLOSE,0);**

Burada ilk parametre VxD dosyasının yerine işaret etmektedir. İkinci ve üçüncü parametreleri okuma yazma modlarıyla ilgilidir, dördüncü parametre çocuk uygulamalarla bağlantı durumunun belirtilmesinde kullanılır, beşinci parametre dosyanın oluşumunda kazanacağı özelliklerin belirlenmesinde kullanılır.

**CreateFile** API'si çağrıldığında dosyayı oluşturur ve bize bir handle değeri verir. Bu handle oluşturulan dosyanın bellekteki adresini ifade eder. VxD ile yapacağımız işlemlerde bu handle'ı kullanırız. Dosyayı oluşturduktan sonra artık yazdığımız VxD kullanıma hazır hale gelmiş olur. Hazırladığımız VxD'nin yaptıklarından haberdar olabilmek için VxD ile haberleşmemiz gerekir. VxD ring0'da bizim Delphi5'te API'leri kullanarak yazdığımız programımız ring3'te çalıştığı için bunu yapmamız biraz güç gözükebilir. Bu noktada yardımımıza **DeviceIOControl** API'si yetişir. **DeviceIOControl** API'side birtakım parametreler içerir bunlara bakacak olursak;

**DeviceIoControl(HVxDHandle,1,NIL,0,addr(sayacgost),4,bytesreturned,nil);**

Birinci parametre oluşturulan VxD'nin handle'dır. İkinci parametre yapılacak işlemin kodudur. Yapılacak işlem parametre gerektiriyorsa bu parametrelerin bellekte bulunduğu adres üçüncü parametrede verilir ve varsa bu parametrenin boyutu dördüncü parametre olarak verilir. Beşinci parametre yapılan işlem sonunda bir veri alınacaksa alınacak verinin bellek adresini, altıncı parametrede bu verinin byte olarak boyutunu ifade eder. Yazılan programda buraya girilen değerler ile VxD'den motorun dönme miktarı bilgilerinin okunması işlemi gerçekleştirilmiştir, yedinci parametre okunan verinin boyutunu saklayan değişkenin adresidir.

VxD'yi kaldırmamız gerektiğinde ise **CloseHandle** API'sini kullanırız. **CreateFile** API'si içerisinde kullandığımız **FILE\_FLAG\_DELETE\_ON\_CLOSE** parametresi sayesinde, oluşturduğumuz VxD'nin handle'ını kapattığımızda, VxD dinamik olarak kaldırılmış olur.

## 5. PC İLE SİNYAL ÜRETİMİ

Motorun kontrolünün hassas bir şekilde gerçekleştirebilinmesi için yüksek frekanslı ve düzgün sinyaller gerekmektedir. Delphi gibi yüksek seviyeli diller kullanılarak kontrol amacıyla kullanacağımız nitelikte sinyal üretmek pek mümkün değildir. Bu problemin çözümü için PC Anakartı üzerinde bulunan 8253 zamanlayıcı tümdevresinden yararlanılmıştır. Tablo 5.1’de görüldüğü gibi 8253 tüm devresi 3 ayrı zamanlayıcı içerir. Bunların ilk ikisi bilgisayar işlemlerinde kullanıldığı için değiştirilmeleri sorun yaratır. Fakat 3. zamanlayıcı ses üretmek için kullanıldığından sinyal üretiminde kullanılması sorun yaratmaz.

**Tablo 5.1** 8253 Zamanlayıcının Portları

| IO Port | Zamanlayıcı   | Kullanım      |
|---------|---------------|---------------|
| 40h     | Zamanlayıcı 0 | Timer/disk    |
| 41h     | Zamanlayıcı 1 | Hafıza        |
| 42h     | Zamanlayıcı 2 | Haportör      |
| 43h     | Kontrol       | Timer Kontrol |

8253 zamanlayıcı kullanılarak 1.193.180 Hertz frekansında sinyal üretilir. Sinyal üretimi assembler kodu kullanılarak yapılmıştır bu kod Delphi içine yerleştirilerek, motorun 1.8 derece hassasiyette yakın açı kontrolü yapılmıştır. Zamanlayıcının programlanması 43h Kontrol portuna aşağıdaki tablodan seçilecek değerlerin girilmesiyle yapılır.

**Tablo 5.2** 8253 zamanlayıcı kontrol bit değerleri

| 7                               | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Fonksiyon                                    |
|---------------------------------|---|---|---|---|---|---|---|----------------------------------------------|
| <b>Sayma Methodu</b>            |   |   |   |   |   |   |   |                                              |
|                                 |   |   |   |   |   |   | 0 | Count in binary                              |
|                                 |   |   |   |   |   |   | 1 | Count in BCD                                 |
| <b>Sayma Modu</b>               |   |   |   |   |   |   |   |                                              |
|                                 |   |   |   | 0 | 0 | 0 |   | Int. on terminal count                       |
|                                 |   |   |   | 0 | 0 | 1 |   | HardW. Retriggerable one shot                |
|                                 |   |   |   | - | 1 | 0 |   | Rate generator                               |
|                                 |   |   |   | - | 1 | 1 |   | Kare dalga                                   |
|                                 |   |   |   | 1 | 0 | 0 |   | Software retriggerable strobe                |
|                                 |   |   |   | 1 | 0 | 1 |   | Hardware retriggerable strobe                |
| <b>Okuma/Yazma Kontrolü</b>     |   |   |   |   |   |   |   |                                              |
|                                 |   | 1 | 0 |   |   |   |   | Counter latch operation                      |
|                                 |   | 0 | 1 |   |   |   |   | Read/Write least significant byte of counter |
|                                 |   | 1 | 0 |   |   |   |   | Read/Write most significant byte of counter  |
|                                 |   | 1 | 1 |   |   |   |   | Read/Write least, then most significant byte |
| <b>Kullanılacak Zamanlayıcı</b> |   |   |   |   |   |   |   |                                              |
| 0                               | 0 |   |   |   |   |   |   | Sayaç 0                                      |
| 0                               | 1 |   |   |   |   |   |   | Sayaç 1                                      |
| 1                               | 0 |   |   |   |   |   |   | Sayaç 2                                      |
| 1                               | 1 |   |   |   |   |   |   | Yok                                          |

## 6. YAPAY SINİR AĞLARI

Güdümlü bilimciler ilk canlıdan itibaren bilgi işlemenin mevcut olduğunu söylemektedir. Bu bakış açısıyla dijital bilgisayarların kullandığı tekniklerden farklı bilgi işleme teknikleri olduğunu düşünebiliriz. Bu tip teknikleri araştırmada, insan ve diğer canlıların kullandığı bilgi işleme tekniklerini taklit etmeyi bir yol olarak görebiliriz. Bu da bizi normal dijital bilgisayar yaklaşımından nöron-bilgisayar yaklaşımına götürür. Dijital bilgisayarda dış dünyadan gelen sinyaller sembolik hale getirilerek işlenir. Nöron-bilgisayarlar ise yüksek oranda paralel yapılardır ve bilgiyi dış dünyadan aldıkları şekilde işleyebilirler.

Nöronlar sinir hücreleridir. Sinir ağları ise birbirlerine bir ağ oluşturacak şekilde bağlanan nöronlardan oluşur. Doğal sinir ağlarına örnek beyindir. Yapay sinir ağının geliştirilmesinin arkasında, psikolojik ve anatomik yaklaşımlar vardır. Nöronların hesaplama özellikleri araştırmacıların dikkatini çekmiştir. Nöronların ve nöronların birbirlerine bağlanmalarıyla oluşan sinir ağlarının, paralel bilgi işlemedeki yüksek kapasiteleri onları hesaplama açısından çok güçlü yapmaktadır. Normal bilgisayarlar için çözümü karmaşık olan, yüz tanıma, bir dili konuşmayı ve anlamayı öğrenme, yüksek oranda lineer olmayan yapıları kontrol etme ve yapıların karmaşık dinamik yapılarını tanımlama gibi işlemler, nöron-bilgisayarlar yardımıyla daha rahat yapılabilmektedir.

YSA'daki hesaplama işlemi şöyle olur; Bir yapay sinire giriş, diğer nöronların belli sayıdaki bağlantılarından veya dış stimulustan gelir. Bu girişlerin ağırlıklar toplamı aktivasyon fonksiyonunda kullanılır. Aktivasyon fonksiyonu, nöronun özelliğini tanımlar ve genellikle nonlineerdir. En çok kullanılan aktivasyon

fonksiyonları birim basamak, sigmoid ve lineer fonksiyonlardır. Aktivasyon fonksiyonundan elde edilen sonuç nöronun çıkışıdır. Bu çıkış, diğer nöronların ağırlıklı giriş bağlantılarına dağılır. Tüm bu nöron bağlantı yapısı ağdaki bilgi akışını tanımlar ve ağın mimarisi olarak adlandırılır. Tek tabakalı ağlar, çok tabakalı ağlar, ileri beslemeli ağlar, geri beslemeli ağlar ve lateral ağar, sıklıkla kullanılan ağ tipleridir. Bu tip ağlarda, ağırlık bağlantıları çok önemlidir. Bundan dolayı da, bağlantılı hesaplama modeli olarak adlandırılırlar. Bilgisayarlarda kullanılan hafıza konseptinin ana kısmı bağlantı ağırlıklarının ayarlanmasına dayanır. Programlama konsepti, ağın eğitimine karşılık gelir, ve ağın eğitiminde kullanılan yöntem öğrenme kuralı olarak adlandırılır. Bir başka deyişle, YSA'lar programlanamaz ama eğitilir. Öğrenme kuralları, kontrollü ve kontrolsüz olmak üzere ikiye ayrılır. Kontrollü öğrenme kurallarından en popüler olanı geri dağılımlı öğrenme kuralıdır. Özet olarak, YSA'nın üç temel yapıtaşısı aktivasyon fonksiyonu, ağın mimarisi ve öğrenme kuralıdır.

## 6.1 YSA'da Hesaplama Yaklaşımları

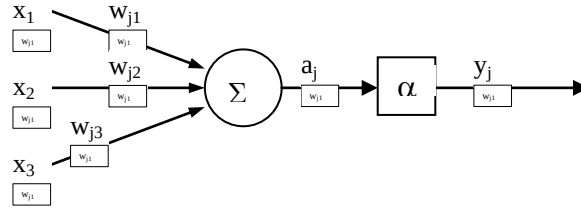
Şekil 2.1'de YSA'nın en temel elemanı olan yapay nöronun temel yapısı görülmektedir. t anında, bu nöronun aktivasyon durumu  $a_j(t)$  veya kısaca  $a_j$  olarak ifade edilir.  $a_j(t)$ 'nin değerinin genelde 1 ile 0 arasında nadiren -1 ile 1 arasında değiştiği kabul edilir. Matematiksel olarak aşağıdaki denklemle ifade edilir.

$$a_j = f(x_i, w_{ji}) = \sum_{i=0}^n w_{ji} x_i \quad (6.1)$$

Burada  $x_i$  giriş,  $w_{ji}$  ise ağırlık matrisidir. Bu nöronun çıkışı olan  $y_j$  non-lineer aktivasyon fonksiyonu parametre olarak  $a_j(t)$  durumunu kullanan  $\alpha$ 'nın çıkışı olarak ifade edilebilir. Bu durumda ifade;

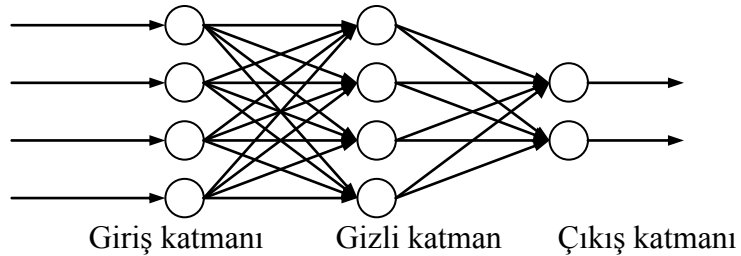
$$y_j = \alpha(a_j) \quad (6.2)$$

Çıkış, ağıdaki diğer nöronlar için giriş olabileceği gibi YSA'nın çıkışı da olabilir.



**Şekil 6.1** Yapay nöronun yapısı

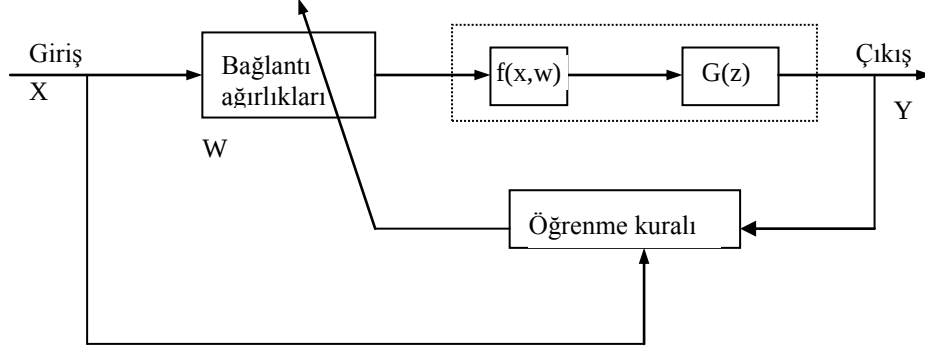
Şekil 6.2'de nöronların birbirine bağlanmasında mümkün olan pek çok bağlantı şeklinden birisi görülmektedir. İlk tabaka olan giriş sütunu dış dünyadan gelen bilgileri içerir. Giriş katmanının çıkışları, ikinci katmana(gizli katmana) ağırlıklar kümesi şeklinde bağlanmaktadır. Çok sayıda gizli katman kullanmak mümkündür. İlk tabakadan giren giriş, işleme elemanlarının bulunduğu katmanlardan ileri doğru çıkışa ulaşınca kadar katmanalar boyunca yayılır. Eğer kontrollü öğrenme yöntemi tercih edilmişse YSA'nın çıkışı, olması istenen çıkış değeriyle karşılaştırılır ve ikisi arasındaki hata her bağlantı ağırlık değeri için ayrı ayrı, bir ayarlama değeri olan  $\Delta w_{ji}$ 'nin hesaplanması için kullanılır. Eğitimin amacı, YSA'ya aynı giriş bir daha girdiğinde hatayı azaltmaktır. Böylece elde edilen çıkış, istenen çıkış değerine yaklaşacaktır.



**Şekil 6.2** İki katmanlı ileri beslemeli ağ mimarisi

Şekil 6.3 tipik bir işlem elemanında meydana gelen işlemlerin basamaklarını kontrol teorisi notasyonu ile göstermektedir. Her nöronun transfer fonksiyonu iki

elemanlı olarak modellenenebilir. Giriş operatörü, girişleri, bağlantı ağırlıklarını ve nonlinear aktivasyon fonksiyonunu içerir. Giriş operatörü  $f$ , a durum değerini bulmak için girişleri ve bağlantı ağırlık değerlerini kullanır. Nonlinear aktivasyon fonksiyonu  $\alpha$ , elemanın son çıkış değeri  $y$ 'yi üretir.



**Şekil 6.3** Öğrenme aşamalarının blok diyagramı şeklinde gösterimi

Şekil 6.4'te en yaygın olarak kullanılan aktivasyon fonksiyonları ve özellikleri gösterilmektedir.

| <i>Aktivasyon fonksiyonu</i> | <i>Şekli</i> | <i>Özellikleri</i>                                                                                                               |
|------------------------------|--------------|----------------------------------------------------------------------------------------------------------------------------------|
| Birim Basamak                |              | <ul style="list-style-type: none"> <li>Hesabı kolay</li> <li>Tersi yok</li> <li>Lineer değişimi yok</li> </ul>                   |
| Lineer Rampa                 |              | <ul style="list-style-type: none"> <li>Hesabı kolay</li> <li>Tersi yok</li> <li>Lineer değişimi var</li> </ul>                   |
| Sigmoid                      |              | <ul style="list-style-type: none"> <li>Tersi var ve sürekli</li> <li>Hesaplanması zor</li> <li><math>y'=y(1-y)</math></li> </ul> |
| Hiperbolik Tanjant           |              | <ul style="list-style-type: none"> <li>Tersi var ve sürekli</li> <li>Hesaplanması zor</li> </ul>                                 |

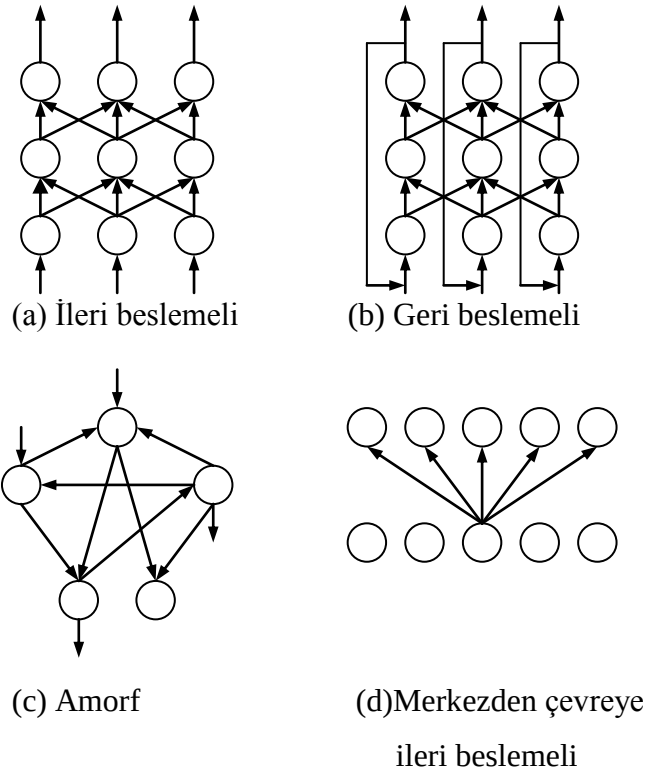
**Şekil 6.4** Aktivasyon fonksiyonlarının şekli ve özellikleri

## 6.2 Ağ Mimarileri

Ağ için uygun ağ mimarisinin seçimi bazı etkenlere bağlıdır. Ağ için mimari seçiminde en önemli etkenleri şöyle sıralayabiliriz;

- Biyolojik bir sistemin taklit edilmesi gerekiyor mu?
- İleri beslemeli mi, geri beslemeli mi olmalı?
- Kaç gizli katmana ihtiyaç duyuluyor?
- Sürekli sistemlerle mi, yoksa ayırık sistemlerle mi ilgileniyoruz?

Şekil 6.5'te çeşitli ağ yapıları gösterilmektedir. Bu yapılardan ileri beslemeli yapı en çok kullanılanıdır. 2.5c de görülen yapı, amorf mimari olarak adlandırılır ve beyin modeli topolojisinde kullanılır. Eğer bu mimari, ağdaki her nöron diğer nöronlarla bağlanacak şekilde düzenlenirse, oluşan yeni ağı Hopfield ağı denir.



Şekil 6.5 Farklı mimari konfigürasyonlar

Farklı mimari yapıları, farklı öğrenme kurallarıyla birleşince 30 dan fazla ağ modeli geliştirilmiştir. Bugün, bu modellerin yaklaşık 13 tanesi kullanılmaktadır.

#### 6.4 Öğrenme Kuralları

Kuşkusuz öğrenme YSA'nın en önemli özelliğidir. Öğrenme kurallarını bulmanın belli bir yolu yoktur. Diğer taraftan, bir öğrenme metoduna ulaşıldığında, araştırmalar bu metod üzerine yoğunlaşır ve bu o metodun daha gelişmiş tiplerinin bulunmasını sağlayabilir. Kabul edilebilir bir öğrenme algoritması şu soruları çözebilmelidir. (1) Ağ bir konuyu öğrenebilir mi? (2) Ne kadar hızlı öğrenir? (3) Öğrenilecek problemi göz önüne aldığımızda eğitim için ne kadar çaba harcamalıyız.

Öğrenme metodları pek çok kategoriye ayrılabilir. Bu ayrımlardan birini ele alacak olursak, öğrenme kontrollü, kontrolsüz ve zorlamalı olarak üçe ayrılır. Kontrollü öğrenmede, giriş değerleri, uygun çıkış değerleriyle birlikte ağa verilir. Zorlamalı öğrenmede ağa sokulan giriş “iyi” yada “kötü” performans oranını ifade eder. Başka bir deyişle, ağa sağlanan geri besleme yönlendiricidir fakat kesin değildir. Kontrolsüz öğrenmede, ağ giriş sinyalleri içinde düzenli yapılar yakalamaya çalışır.

Yukarıda anlatılan öğrenme kurallarının tümü ağ topolojisinin önceden tanımlanmış olmasını ve öğrenme işlemi süresince değiştirilmemesini şart koşar. Öğrenme sırasında ağın topolojisini değiştirmeye dayanan öğrenme teknikleride vardır.

Öğrenme işleminin arka planı ve analizleri en iyi özel bir durumu inceleyerek anlaşılır. İşlem elemanlarından oluşan bir ağı ele alalım. Ağda, girişleri  $x_1, x_2, \dots, x_k$ , ağırlıkları  $w_{ij}$  ( $i=1,2,\dots,k$ ) olan  $j$  elemanını ele alalım. Tipik bir  $x$  giriş vektörü şöyle olur;

$$\mathbf{x} = [x_1 \ x_2 \ ... \ x_k] \quad (6.3)$$

$j$ . elemanın çıkışı  $y_j$  ile gösterilir. Eğer bu eleman ağa dahil olan bir eleman ise,  $j$  1,2,...,N arasında bir değer alır, ve çıkış N boyutlu bir vektör olan  $\mathbf{y}$  ile gösterilebilir.

Kontrollü öğrenme metodlarıyla, bir ağ öncelikle arzulanan işlemleri yapacak şekilde eğitilmelidir. Eğitim için, T giriş vektörleri kümesinin ve buna karşılık gelen arzulanan çıkış vektörleri kümesinin mevcut olması gerekir.

$$\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, ..., \mathbf{x}^{(T)} \quad (6.4)$$

$$\mathbf{d}^{(1)}, \mathbf{d}^{(2)}, \mathbf{d}^{(3)}, ..., \mathbf{d}^{(T)} \quad (6.5)$$

Bu giriş-çıkış( $\mathbf{x}^{(t)}, \mathbf{d}^{(t)}$ ),  $\{t = 1, 2, ..., T\}$  çiftleri eğitim seti olarak tanımlanır. Bu giriş-çıkış çiftleri ağa, öğrenme algoritması tarafından aktarılır. Her bir çift için ağın ağırlıkları, ağın çıkışı  $\mathbf{y}$ , arzulanan değerler  $\mathbf{d}$ 'ye istenen derecede yaklaşıp yaklaşacak şekilde ayarlanır. Ağırlıkların ayarlanmasında izlenen yola, öğrenme kuralı denir. Öğrenme kuralının esas amacı  $w_{ij}$ 'yi ayarlayarak ağın cevabı ile arzulanan çıkış arasındaki hata,  $\mathbf{e} = \mathbf{y} - \mathbf{d}$ 'yi mümkün olan en kısa zamanda minimum yapmaktır. Aşağıda sıklıkla kullanılan bazı öğrenme kuralları anlatılmıştır.

### 7.3.1 Hebb Kuralı

En eski ve en iyi bilinen öğrenme kuralı olan Hebb kuralı adını psikolog Hebb'ten almıştır. Hebb kuralının arkasındaki düşünce şöyledir. Ağırlık, bağlantının gücünü gösterir. Bir başka deyişle, bağlı elemanların meydana gelme sıklığını ifade eder. Eğer çıkışı  $y_j$  olan bir ağ elemanının girişine, çıkışı  $x_k$  olan başka bir eleman çıkışı girilirse, Hebb kuralına göre bağlantı ağırlığı değeri  $\Delta w_{jk}$  değeri ile ayarlanır.

$$\Delta w_{jk} = \eta y_j x_k \quad (6.6)$$

Burada,  $\Delta w_{jk} = \Delta w_{jk}(t) = w_{jk}(t+1) - w_{jk}(t)$  ve  $\eta > 0$  sabitide öğrenme oranı olarak adlandırılır. Ayrıca;

$$y_j = B \sum_k w_{jk} x_k \quad (6.7)$$

Burada dikkat edersek ağırlık ayarlaması sadece  $x$  ve  $y$  sıfırdan farklı olduğunda gerçekleşmektedir. Hebb kuralı kontrolsüz öğrenmeye bir örnektir.

### 7.3.2 Delta kuralı

Bu öğrenme kuralı, lineer aktivasyon fonksiyonlu ağlara uygulanan Widrow-Hoff kuralı olarak adlandırılır.

$$y_j = \sum_k w_{jk} x_k \quad (6.8)$$

Hata terimini şu şekilde tanımlayalım,

$$E = \sum_{t=1}^T \sum_{j=1}^N (d_j^{(m)} - y_j^{(m)})^2 \quad (6.9)$$

Burada,  $j$  elemanı için,  $d_j$  arzulanan çıkış,  $y_j$  ise ağın çıkışıdır.  $E$  hatası aşağıdaki kuralla göre ağırlıkların ayarlanması ile minimum bir değere azaltılabilir.

$$w_{ij}^{(new)} = w_{ij}^{(old)} + \Delta w_{ij}^{(new)} \quad (6.10)$$

burada;

$$\Delta w_{ij} = \eta (d_j - y_j) x_k = \eta \delta_j x_k \quad (6.11)$$

Burada,  $\delta_j$ ,  $(d_j - y_j)$  yerine kullanılmıştır. Delta kuralı kontrollü öğrenme kuralıdır. Bu kuralın ana problemi  $d_j$ 'nin bir gizli katman ağ elemanının çıkışı olması

durumunda ne şekilde hesaplanacağıdır. Bu problem genelleştirilmiş delta kuralıyla çözülmüştür. Delta kuralıyla ilgili bir başka problem ise, ağırlıkların bir alt optimal noktada takılması ve gerçek optimal noktanın kaçırılmasıdır.

### 6.3.3 Genelleştirilmiş Delta kuralı (Geri Dağılım)

Geri dağılım olarak da bilinen ve çok tabakalı ileri beslemeli ağlar için geliştirilmiş olan genelleştirilmiş delta kuralı belki de en çok kullanılan öğrenme kuralıdır. Bu kural kullanılarak hatâ ağ ağırlıkları ile ifade edilebilir, buda ağırlıkların hataların mutlak karelerinin, gradyent yaklaşımıyla minimum yapılması yoluyla ayarlanması imkanını sunar. Geri dağılım metodunda gradyent hesabı gerekli olduğu için ağ elemanlarının aktivasyon fonksiyonu türevi alınabilir fonksiyon olmalıdır. Diğer gradyent metodlarında olduğu gibi bu metoddaki da özellikle lokal minimumlar civarında yavaş yakınsama problemi olabilmektedir. Bir lokal minimuma rastlanıldığında genel minimuma ulaşmak mümkün olmaz. Çıkış katmanındaki gradiyent hesabı ileri doğrudur, çünkü arzulanan ve ulaşılmak istenen değerler oradadır. Diğer taraftan gizli katmanlardaki elemanların çıkışları için arzulanan değerler tanımlanamadığı için, gizli katmanlarda gradient hesabı zor bir hal alır. Genelleştirilmiş delta kuralı, delta kuralı ile hata terimi,  $\delta_j$ , hariç aynı yolu takip eder. İnceleyecek olursak. Çıkış katmanını nodları için

$$\delta_j = d_j - y_j \quad (6.12)$$

ve gizli katman nodları için

$$\delta_j = \sum_i w_{ij} \delta_i \quad (6.13)$$

Geri dağılımla öğrenme kuralı üzerinde pekçok değişiklik yapılmaktadır. Bunlardan birisi fonksiyona, momentum terimi olarak  $\mu \Delta w_{ij}^{(t-1)}$  ifadesini eklemektir. Bu durumda ifade şu hali alır.

$$\Delta w_{ij}^{(t)} = \mu \Delta w_{ij}^{(t-1)} + \eta \delta_j^{(t-1)} x_i^{(t-1)} \quad (6.14)$$

sembol üst değerleri t. eğitim döngüsünü ifade etmektedir. Gizli katmanların sayısı arttıkça ağırlık ayarlamaları için gereken hesaplama işleri oldukça artmaktadır. Geri dağılımlı öğrenme algoritmasına yönelik algoritma geliştirme çalışmaları oldukça aktif araştırma alanlarından birisidir.

Genelleştirilmiş delta kuralının bir başka versiyonu da Durbin ve Rumelhart tarafından geliştirilmiştir. Bu yöntemde ağırlıklar toplamı yerine ağırlıklar çarpımı kullanılmıştır.

## 7. YAPAY SİNİR AĞLARI İLE SİSTEM TANIMLAMA

Geçen 50 yıl içerisinde ki, matematiksel sistem teorisindeki gelişmeye bağlı olarak, dinamik sistemlerin sentez ve analiz teknikleride gelişmiştir. Sağlam temeller üzerine oturtulmuş lineer sistem teorisi, lineer sistemlerin, lineer cebir, karmaşık sayılar ve differansiyel denklemler ile ifade edilmesine yardımcı olmuştur. Dinamik sistemlerin dizayn teknikleri, sistemin kararlılık(stabilite) özelliklerine bağlıdır. Lineer sistemlerin kararlılık analiz teknikleri geçtiğimiz yüzyılda geliştirilmiştir. Ancak lineer olmayan sistemler için, henüz uygun analiz teknikleri bilinmemektedir. Lineer olmayan sistemlerin kararlılık, dinamik cevap ve gürbüzlük gibi özellikleri ancak çok az sayıda bilinen sistem için analiz edilebilmektedir.

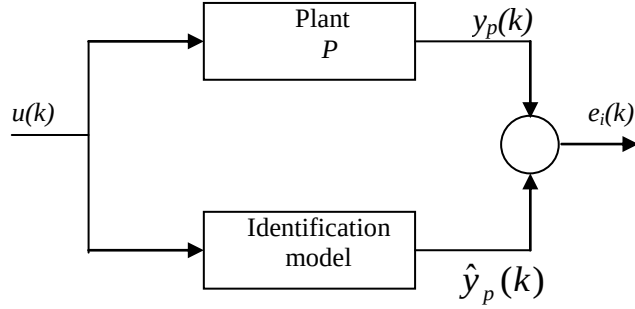
YSA alanında iki ağ yapısı türü oldukça dikkat çekmektedir. Bunlar; 1. Çok katmanlı YSA, ve 2. recurrent YSA. Çok katmanlı sinir ağları kalıp tanıma alanında oldukça başarılıyken, recurrent ağlar ise çeşitli hafıza ve optimizasyon problemlerinde kullanılmaktadır. Sistem teorisi noktasından baktığımızda, çok katmanlı sinir ağları, lineer olmayan statik kalıplar olarak kabul edilirse, recurrent ağlar, geribeslemeli dinamik sistemlere karşılık gelmektedir. Her iki grup arasındaki farklara rağmen, dinamik sistemlerde her iki yapıyla da karşılaşıldığı için, iki kategoriyi de içine alan bir yapıyı ele alamak uygun olacaktır.

## 7.1 TANIMLAMA KONSEPTİ

Sistem teorisinin temel problemleri sistemin tanımlanması ve karakterize edilmesidir. Karakterize etme problemi, ilgilenilen sistemi ifade edecek matematiksel sınıfın belirlenmesidir. Sistemin modeli, çıkışı  $\mathbf{Y}$ , girişi  $\mathbf{U}$  olan  $P$  operatörü ile ifade edilir. Hedef  $P$ 'nin ait olduğu  $\mathbf{P}$  sınıfının belirlenmesidir.  $P \in \mathbf{P}$  olmak üzere verilen bir  $\mathbf{P}$  sınıfında tanımlama problemi,  $\hat{\mathbf{P}} \subset \mathbf{P}$  olmak üzere bir sınıf ve  $\hat{P} \in \hat{\mathbf{P}}$  olmak üzere bir eleman tanımlamaktır. Böylece  $\hat{P}$ ,  $P$ 'ye istenen bir oranda yakınsar. Statik sistemlerde,  $\mathbf{U}$  ve  $\mathbf{Y}$  uzayları,  $\mathbf{R}^n$  ve  $\mathbf{R}^m$ 'in alt kümeleri olurken, dinamik sistemlerde,  $[0, T]$  veya  $[0, \infty]$  aralığında Lebesgue integrale edilebilir fonksiyonlarıyla sınırlandıkları kabul edilir. Her iki durumda da,  $P$  operatörü verilen giriş-çıkış çiftleriyle tanımlanır. Statik tanımlamaya bir örnek, kalıp tanıma (pattern recognition) problemidir. Bir karar fonksiyonu  $P$  ile,  $U_i \subset \mathbf{R}^n$  kümesi elemanları,  $y_i \subset \mathbf{R}^m$  kümesi elemanlarına aktarılır. Dinamik sistemlerde,  $P$  operatörü, zamana bağlı,  $u(t)$ ,  $y(t)$ ,  $t \in [0, T]$ , giriş-çıkış çiftiyle tanımlanır. Her iki durumda da, tanımlama problemi,  $\hat{P}$ 'yi, istenen,  $\varepsilon > 0$  değeri ve çıkış uzayında uygun şekilde tanımlanmış bir norm ( $\| \cdot \|$  ile ifade edilmiştir.) ile ifade etmektir.

$$\|\hat{y} - y\| = \|\hat{P}(u) - P(u)\| \leq \varepsilon, \quad u \in \mathbf{U} \quad (7.1)$$

Buradaki denklemde,  $\hat{P}$  tarafından üretilen çıkış ve çıkıştan gözlenen  $y$  arasındaki hata  $\hat{y} - y = e$  olmak üzere,  $\hat{P}(u) = \hat{y}$ , tanımlama modelinin çıkışını gösterir.



**Şekil 7.1** Tanımlama modeli

Dinamik sistemler vektör differansiyeli veya fark denklemleriyle ifade edilir. Sistem teorisinde bu gösterim metodu oldukça iyi gelişmiştir ve pekçok sistem sınıfına uygulanmaktadır.

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}(t) = \Phi[\mathbf{x}(t), \mathbf{u}(t)] \quad (7.2)$$

$$\mathbf{y}(t) = \Psi[\mathbf{x}(t)]$$

$\mathbf{x}(t) = [x_1(t), x_2(t), \dots, x_n(t)]^T$ ,  $\mathbf{u}(t) = [u_1(t), u_2(t), \dots, u_p(t)]^T$ ,  $\mathbf{y}(t) = [y_1(t), y_2(t), \dots, y_m(t)]^T$  olmak üzere  $n$ . dereceden,  $p$  girişli,  $m$  çıkışlı sistem belirtir.  $u_i(t)$  girişleri,  $x_i(t)$  durum değişkenlerini,  $y_i(t)$  çıkışları ifade etmektedir.  $\Phi$  ve  $\Psi$ ,  $\Phi: \mathbf{R}^n \times \mathbf{R}^p \rightarrow \mathbf{R}^n$  ve  $\Psi: \mathbf{R}^n \rightarrow \mathbf{R}^m$ , olarak tanımlanan statik, lineer olmayan atamalardır.  $\mathbf{x}(t)$  vektörü, sistemin  $t$  anındaki durumunu ifade eder.  $\mathbf{u}(t)$  girişi,  $[t_0, t)$  aralığında tanımlıdır.  $\mathbf{y}(t)$  çıkışı tamamıyla,  $t$  anında sistemin durumuna bağlıdır. (3.2) denklemi sistemin giriş-durum-çıkış gösterimi ile ifade edilmiştir. Sistemin ayrık zamanlı gösterimi, benzer şekilde gösterilebilir.

$$\mathbf{x}(k+1) = \Phi[\mathbf{x}(k), \mathbf{u}(k)] \quad (7.3)$$

$$\mathbf{y}(k) = \Psi[\mathbf{x}(k)]$$

Eğer sistem zamandan bağımsız, lineer sistem ise şu şekilde ifade edilebilir.

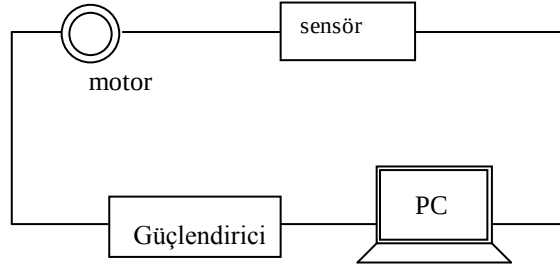
$$\begin{aligned}\mathbf{x}(k+1) &= \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) \\ \mathbf{y}(k) &= \mathbf{C}\mathbf{x}(k)\end{aligned}\tag{7.4}$$

burada  $\mathbf{A}, \mathbf{B}, \mathbf{C}$  sırasıyla,  $(n \times n)$ ,  $(n \times p)$ ,  $(m \times p)$  boyutunda matrislerdir. Zamandan bağımsız lineer sistemler teorisi, bize,  $\{ \mathbf{A}, \mathbf{B}, \mathbf{C} \}$  veya  $\{\Phi, \psi\}$  bilindiği zaman, gözlemlenebilirlik, kararlılık ve kontrol edilebilirlik gibi konseptleri analiz etme imkanı verir. Diğer taraftan,  $\{\Phi, \psi\}$  bilinen fonksiyonları nonlineer olduğunda, aynı analizler, benzer türde güçlü araçların bulunmadığı, lineer olmayan matematiksel denklemlerle çözülür.

## 8. SİSTEMİN ve YAZILIMIN TASARLANMASI ve GENEL ÖZELLİKLERİ

### 8.1 Sistemi Oluşturan Temel Elemanlar ve Özellikleri

Bu bölümde bir DC motorun deneysel olarak kontrolü ve yazılımın yapısı incelenecektir. Aşağıda sistemin genel yapısını gösteren şema görülmektedir.

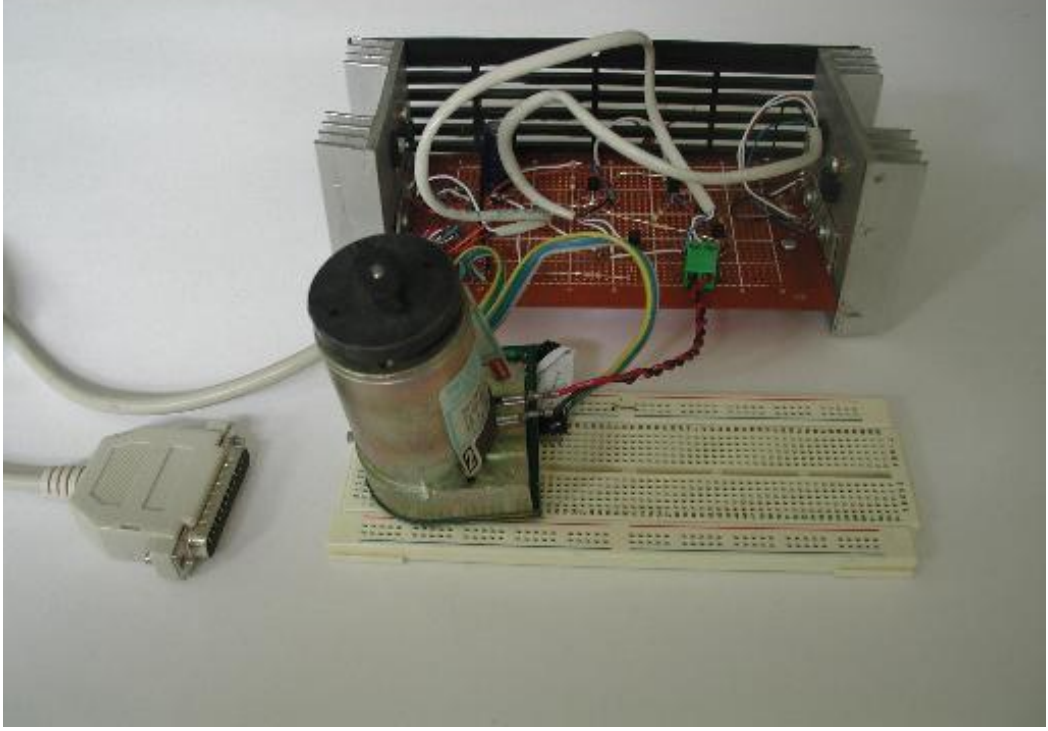


**Şekil 8.1** Sistemin Blok Şeması

Sistem Temel olarak PC, motor, encoder ve güçlendiriciden oluşmaktadır. Tasarlanan sistemin resmi resim-8.1’de görülmektedir. Oluşturulan sistemde PC'nin işlevleri motoru tahrik eden sabit genlikli darbe katarı işareti üretmek, motordan gelen sinyalleri algılayarak motorun konumunu ve hızını hesaplamak bu sinyallerden yararlanarak, oluşturulmuş olan YSA'yı eğitmek ve eğitilen YSA'yı kullanarak verilen referans konumuna motoru taşımaktır

Bu işlemleri gerçekleştirirken bilgisayar içerisinde çalışan iki farklı tipte yazılım geliştirildi. Yazılımlardan birisi PC'nin donanım elemanlarından olan paralel portun kesme sinyalleriyle ilgili ayarlar yaparken, diğeri YSA algoritmalarını

alıřtırmak ve motor kontrol in gerekli olan sinyallerin retimi iřlemlerini yrtmektedir.



**Resim-8.1** Tasarlanan sistem

řimdi sistemi oluřturan elemanları ayrı ayrı ele alalım.

#### **8.1.1. Motorun zellikleri**

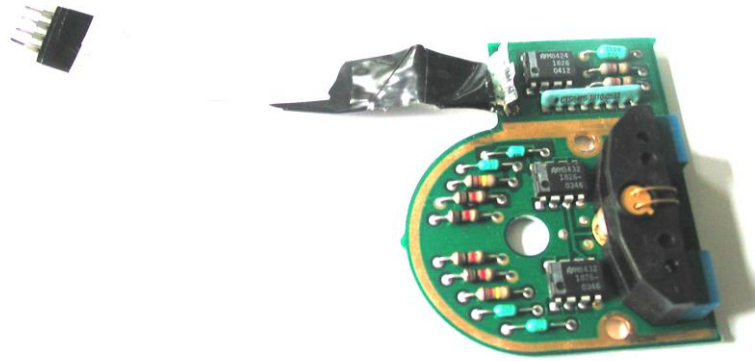
alıřmada 20V gerilim altında alıřan, ve eski bir plotterden sklen bir fıralı DC servo motor kullanılmıřtır. Bu motorun tercih edilmesinin temel sebebi optik pozisyon sensrne(enkoder) sahip olmasıdır. Enkoderi olmayan motorlar da, bir pozisyon enkoderi monte edilerek kullanılabilir. Bu tip motorların ortalama servis mrleri 2000 saattir. Motor resim-8.2’de grlmektedir.



**Resim-8.2** DC servo motor

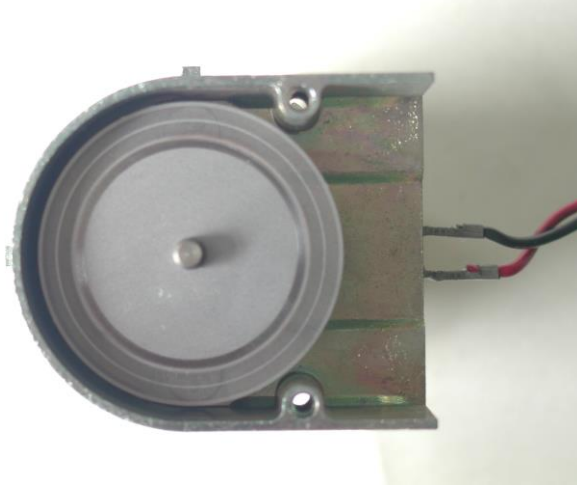
### 8.1.2. Optik Enkoder Özellikleri

Çalışmada pozisyon bilgilerini elde etmek için kullanılan enkoder, optik sesörlerden oluşan bir enkoderdir. Bu sensörler, motorun rotor olarak adlandırılan hareketli kısmına takılan bir diskin üzerindeki deliklerden geçen ışığı algılama özelliğine sahiptir. Enkoderin resim-8.3'te görülmektedir.



**Resim 8.3** Optik enkoder

Disk üzerinde iki sıra halinde delikler bulunmaktadır(resim-8.4). Yapılan çalışmada dışta kalan halkadaki deliklerden gelen sinyaller okunmuştur. Dış halkada toplam 500 delik bulunmaktadır.



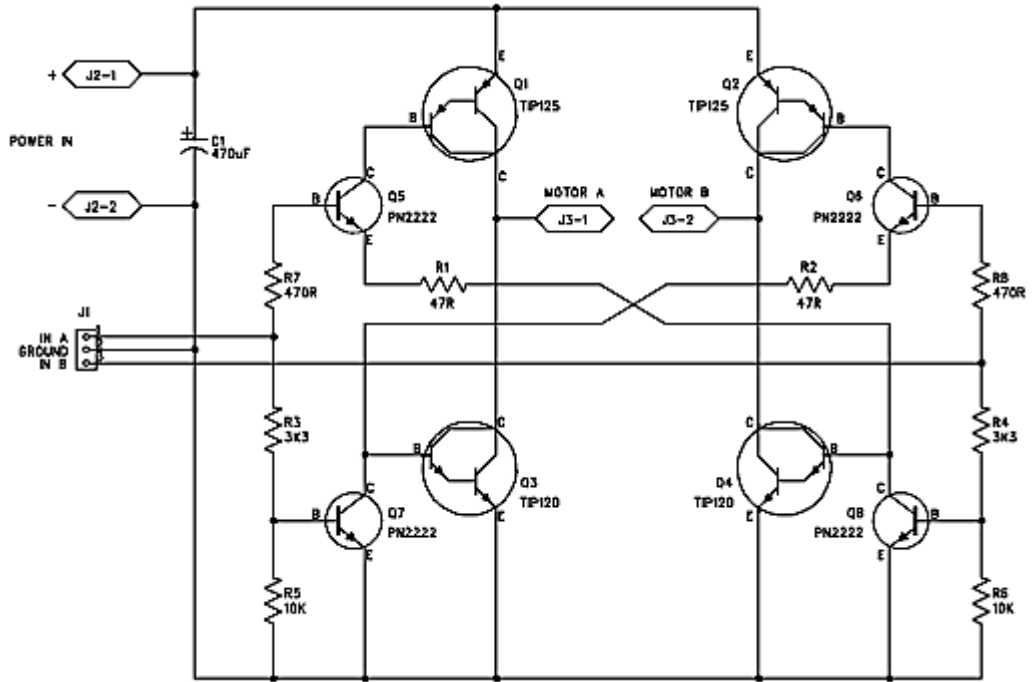
**Resim-8.4** Delikli Disk

Buna göre motorun bir tam dönüşünde enkoder 500 sinyal üretmektedir. Sensöre ışık gelmesi durumunda enkoder mantıksal bire karşılık gelen 5V'luk bir sinyal vermektedir. Işık gelmemesi durumunda ise mantıksal 0'a karşılık gelen 0V'luk bir sinyal vermektedir. Paralel porta bağlı bulunan enkoderdeki her mantıksal birden mantıksal sıfıra geçiş bilgisayar tarafından bir kesme sinyali olarak algılanmakta ve yazılan ISR'deki sayaç değişkeninin değeri bir artmaktadır.

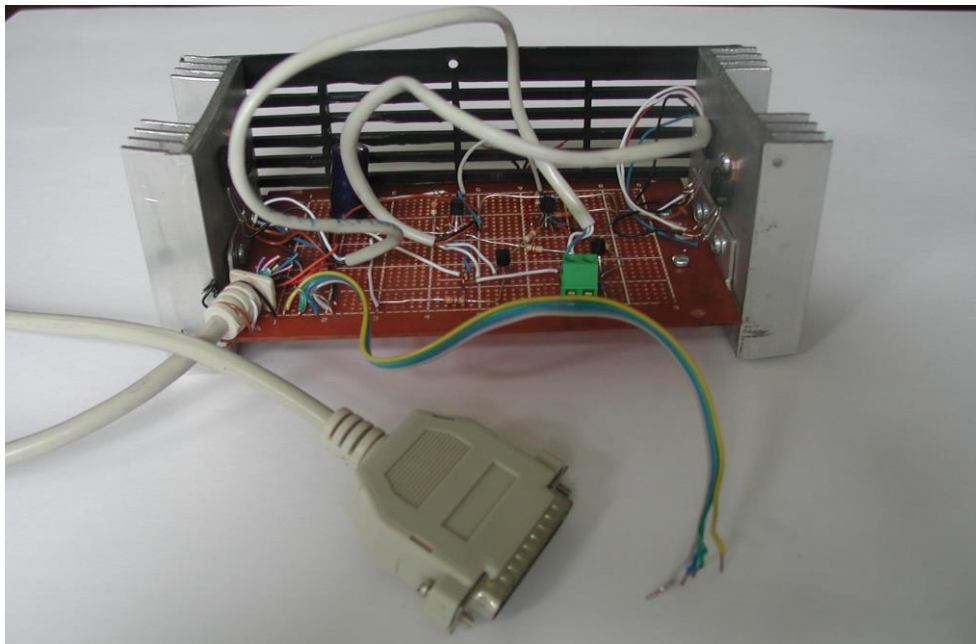
### **8.1.3. Güçlendirici**

Paralel port kullanılarak bir eyleyicinin tetiklenmesi için paralel port sinyallerinin güçlendirilmesi gerekmektedir. Paralel porttan 2.4V gerilimde 2.6mA akım çekilebilmekte ve paralel porta 4.2 V gerilimde 24mA akım verilebilmektedir. Bu çalışma için oluşturulan güçlendirici yardımıyla, paralel port

çıkışı yaklaşık 60W'a çıkarılmıştır. Güçlendirici H-Köprüsü yapısında olduğu için çift yönde kontrolü desteklemektedir. Güçlendiricinin verilen darbeleri çift yönde yüksek hızda güçlendirmesi PWM kontrol yapılmasını olanaklı kılmaktadır. Güçlendirici 40V'luk bir güç kaynağı ile beslenmekte ve motora 40V gerilimde 1.5A akım sağlamaktadır. Güçlendiricinin devre şeması aşağıda görülmektedir.



Şekil 8.2 Güçlendirici devre şeması



Resim 8.5 Güçlendirici

## 8.2 Yazılım Genel Özellikleri

### 8.2.1 VXD yazılımı

Endüstriyel kontrolde doğrudan doğruya PC kullanımı, PC'lerdeki performans artışına bağlı olarak her geçen gün daha da artmaktadır. Endüstriyel kontrolde PC kullanımı daha esnek özellikler sağlayan sistemler tasarlanmasını mümkün kılmaktadır. PC kullanılarak bir dinamik sistemin kontrolü için sistem ile PC arasında etkin bir iletişim kurulması zorunluluğu vardır. PC ile sistem arasındaki iletişimde PC üzerinde bulunan PCI, SCSI, Paralel Port, Seri Port, USB port gibi portlar kullanılabilir.

SCSI ve PCI gibi anakart üzerindeki portların kullanılması genellikle veri alış verişini sağlayan özel kartların kullanımıyla gerçekleştirilir. Bu tür kartlar veri iletişimini sağlamanın yanında D/A ve A/D dönüşümleri de mümkün kılmaktadır. Ancak bu tür kartlar hem pahalıdır hemde her zaman her yerde bulunmayabilir. Bunun yanında dizüstü bilgisayarlara bu tür kartların takılması biraz güçtür. Veri kartlarının bu dezavantajlarından dolayı bu çalışmada PC'lerde standart olarak bulunan paralel portun donanım özelliklerini geliştirdiğimiz dinamik sistemin veri iletim ihtiyaçlarını karşılayacak şekilde ayarladık.

Tasarlanan dinamik sistemde bir motorun pozisyon kontrolünün yapılması hedeflenmiştir. Bu yapılırken bilgisayarda değişik genişliklerde darbeler üretilerek motora uygulanmakta, bu darbelere karşılık motorun dönüş konum bilgileri motora monte edilen bir enkoderden bilgisayara ulaşmakta, bilgisayar da bu bilgileri kullanarak dinamik sistemin bir YSA eşdeğerini oluşturmaktadır. Tasarımda iki nokta kritik önem arz etmektedir. Bunlardan birisi üretilen darbelerin hassaslığı diğeri ise enkoderden yüksek hızda gelen (3000 devir/dk için 25kHz) verilerin eksiksiz şekilde algılanmasıdır. PC ile veri okumanın en verimli ve güvenli yolu verilerin, PC tarafından kesme sinyali olarak algılanmasını sağlamaktır. Kesme sinyalleri daha önce de belirttiğimiz gibi işlemcinin yaptığı işe ara vermesini ve

kesme sinyalinin işaret ettiği işlemi gerçekleştirdikten sonra işlemcinin tekrar kaldığı yerden eski işlemlerini yapmaya devam etmesini sağlayan bir sinyal tipidir. Windows işletim sistemi altında PC paralel portunun enkoder verilerini kesme sinyalleri şeklinde algılamasını sağlamak için bir donanım sürücüsü yardımıyla paralel portun bazı özelliklerini ayarlamamız gerekir.

Tasarlanan sistemde PC, içerisine yüklediğimiz ve 2.Bölümde anlattığımız VxD yazma tekniklerini kullanarak yazdığımız sanal donanım sürücüsü(VxD) ile bu ayarlar yapılmaktadır. Yazılan VxD sürücüsü bilgisayarın paralel portunun kesme sinyalleri vektör tablosu değerini, bizim yazmış olduğumuz ISR'nin adresi ile değiştirmekte ve ISR ile Windows işletim sistemi arasında veri alış verişini sağlamaktadır.

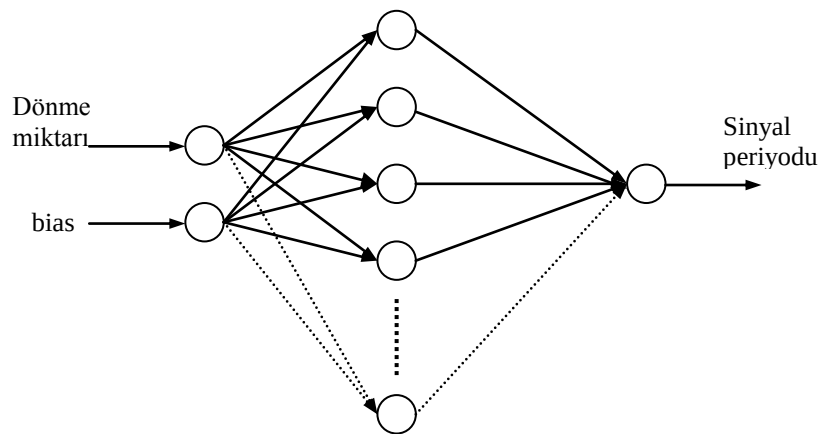
### **8.2.2 PWM darbelerinin üretimi, VxD kontrolü ve YSA algoritmaları**

Delphi 5 kullanarak yazdığımız kısım ise, assembler kodları yardımıyla önce rastgele darbeler üreterek bu darbeleri paralel port yardımıyla güçlendiriciye oradan da motora ulaştırmakta, daha sonra da motorda meydana gelen dönme miktarlarını algılayan VxD ile iletişim kurarak bu bilgileri VxD'den almaktadır. Delphi gibi yüksek seviyeli programlama dilleri kullanılarak motor kontrolünde kullanılabilecek hassasiyette sinyaller üretilemez, bu sebeble 5.Bölümde anlattığımız 8255 zamanlayıcı tümdevresi Delphi içinde kullanılan assembler kodları yardımıyla istenen nitelikte sinyaller verecek şekilde programlanmıştır. Bu şekilde PWM kontrol yapmak için gereken darbelerin frekans ayarları yapılabilmektedir. Delphi 5'te yazılan kodla ayrıca YSA algoritmasında oluşturulmuş ve elde edilen veriler kullanılarak bu YSA'nın eğitimi işlemi gerçekleştirilmiştir. Hazırlanan kullanıcı arayüzü sayesinde, oluşturulan YSA ile motorun pozisyon kontrolünde gerçekleştirilmiştir. Aşağıda oluşturulan kullanıcı arayüzü görülmektedir.

Modern kontrol teorisi yöntemlerinde olduğu gibi motorun dinamik modelinin oluşturulması ve buna uygun bir kontrol devresi oluşturulması yoluyla

oluşturulacak bir devre hiç şüphesiz YSA'ya göre daha yüksek hassasiyette ve güvenilir kontrol yapılmasını sağlayacaktır. Ancak motorun bu tür bir modelinin oluşturulması için motorun dinamik özelliklerinin tam olarak bilinmesinin yanın da motorun tahrik edeceği sistemin dinamik özelliklerinin de bilinmesi gerekir. Eğer kontrol edilecek sistemin dinamik özellikleri değişkense, o zaman sistemin dinamik modelinin elde edilmesi iyice zorlaşabilir. Böyle durumlarda YSA kullanılması iyi bir çözüm olabilir. YSA kullanılması sayesinde, motorun özelliklerinde değişiklik meydana gelmesi, mesela motorun sökölüp yerine daha farklı dinamik özelliklerde bir motor takılması veya motorun tahrik ettiği sistemin dinamik özelliklerinin değişmesi, mesela sisteme yeni bir dişli sistemi eklenmesi, durumunda kontrol sistemi kendisini kısa sürede yeni sistemin özelliklerine göre ayarlayabilmektedir.

Çalışmada iki katmanlı, gizli katmanında 22 nöron bulunan ileri beslemeli geri dağılımlı bir YSA kullanılmıştır. YSA'nın eğitim aşaması ve kullanım aşamaları vardır. Eğitim aşamasında motora rastgele periyodlarda verilen akım ve gerilimde motorun dönme miktarları tespit edilip bir dosyada saklanmaktadır. Elde edilen bu giriş-çıkış bilgileri motorun dinamik karakteristiğini YSA'nın ağırlık (weight) değerlerine taşımaktadır. Motora uygulanması gereken darbenin genişliği hesaplanırken YSA'daki bu bilgiler kullanılmaktadır. Nöron sayılarının ne olacağını belirlemede, deneme yanılma yöntemi kullanılmaktadır. Yapılan denemelerde en iyi sonuçlar gizli katmanda 22 nöron kullanılması durumunda elde edilmiştir. Daha az nöron kullanılması durumunda hassasiyet azalırken, daha fazla nöron kullanınca hem eğitim süresi uzamakta hem de YSA mantıksız sonuçlar vermektedir. Nöronlarda threshold fonksiyonu olarak sigmoid fonksiyonu kullanılmıştır.



**Şekil 8.3** Sistemin yapay sinir ağı modeli

YSA, eğitim sürecinde kullanılan değerlerin sınırlarının dışında verilerin girilmesi durumunda hatalı sonuçlar vermektedir. Bunun önüne geçmek için YSA ile birlikte sistemin dinamik modelinin de kullanılması ve YSA'nın, dinamik modelin verdiği sonuçtan çok fazla miktarlarda sapması durumunda, dinamik modelin verdiği sonucun kullanılması, daha iyi sonuç veren bir sistemin oluşturulmasını mümkün kılacaktır.

Buna bir başka çözümde öğrenme verilerini bazı sınırlara ayırarak her sınır aralığı için bir YSA tanımlamak olabilir. Bu yöntem de önce tüm veriler toplanır ve daha sonra önceden belirlenmiş sınırlandırmalara göre veriler gruplara ayrılır ve her gruba veri, o gruba ait olan YSA'nın eğitiminde kullanılır. Kullanım aşamasında ise, motordan istenen dönme miktarı hangi grubun sınırlarına giriyorsa o grubun YSA'sı motorun kontrolünde kullanılan darbenin üretiminde kullanılır.

Geliştirilen sistemde veri alışverişinde sadece paralel portun kullanılmış olması geliştirilen sistemin dizüstü bilgisayarlarda kullanımını mümkün kılmıştır. Bu özellik sistemin, masaüstü bilgisayarların taşınmasının güç olduğu yerlerde ve durumlarda kullanılmasını sağlamaktadır. Çalışmanın en büyük özelliği, tanımlanan paralel port sürücüsü sayesinde, Windows işletim sistemi üzerinde çalışan bilgisayarların, paralel portlarına kazandırılan yüksek hızda güvenilir veri alma yeteneğidir. Bu sayede analogtan dijitalde yada dijitalden analoga çevirme işlemlerinin olmadığı durumlarda veri işleme kartlarının yerini alabilecek bir sistem oluşturulmuştur. Sisteme eklenecek birkaç küçük devreyle A/D ve D/A sinyal çevrimi işlemleri de gerçekleştirilebilir. Bu durum hem kontrol sisteminin maliyetini azaltmakta hem de sistemin yapısının basitleşmesini sağlamaktadır.

Motor kontrolünde pozisyon kontrolünün yanında hız, ivme ve tork parametrelerinin kontrolünün yapılması da gereklidir. Bu parametrelerin kontrolünün yapılmasında PWM sinyallerinden yararlanılmaktadır.

Hız kontrolünün yapılmasında PWM sinyallerinden şu şekilde yararlanılır. DC motora belli bir gerilimde , elektrik akımı uygulandığında motor ani olarak en yüksek hıza ulaşmaz, ancak belli bir gecikmeden sonra bu hıza ulaşır. PWM ile hız

kontrolünde, motorun hızlanma sürecinde istenen hız değerine ulaşıldığında akım kesilir hızın azalma süreci başlarken kısa bir süre için yine motora akım verilir ve motor istenen hızda bu şekilde uygulanan darbelerle tutulur. Hız azaltılmak istenirse bu şekilde uygulanan darbelerin frekansı azaltılır. Hız artırılmak istenirse darbelerin frekansı artırılır.

Bazı durumlarda ivme kontrolünün yapılmasında gerekebilir. Mesela motor şişe taşıyan bir yürüyen bantı tahrik ediyor olsun bu durumda ani hızlanmalar şişelerin devrilmesine sebep olabilir. Bu durumun önüne geçilmesi için ivme kontrolü yapılması gerekir. İvme kontrolünde hız kontrolü gibi PWM sinyalleri yardımıyla yapılır. Belirlenen bir hız değerine tek bir darbe yerine birden fazla darbe verilerek ulaşması ilkesine dayanır.

Hız kontrolünde çift yönlü PWM darbeleri de kullanılabilir. Bu yöntemde istenen yönde dönme hızını artırıcı darbe üretilir eğer hız aşımı olursa hızın istenen düzeye inmesi için ters yönde sinyal uygulanır. Geliştirilen sistem her iki yöntemi de destekleyebilecek şekilde tasarlanmıştır. Güçlendiricinin H-köprüsü yapısında olması motora çift yönde darbe verilmesini olanaklı kılmaktadır. Ters yönde darbe vermek, özellikle hassas kontrolün gerekli olduğu durumlarda motoru ani olarak durdurmak için de kullanılır.

Endüstriyel uygulamalarda kullanılan bir diğer motor tipide fırçasız motorlardır. Fırçasız motorlar uzun süre bakım gerektirmeden çalışabildikleri için tercih edilirler ve bu motorlar sadece PWM sinyalleri kullanılarak çalıştırılabilirler. Bu motorları çalıştırmak için gereken PWM sinyallerinin üretiminde bilgisayar belleğine yerleştirilen bir sinüs tablosundan yararlanılır. Sinüs fonksiyonunun tablodan okunan değerlerinin uygun bir katsayı ile çarpılmasıyla elde edilen değerlere göre üretilen analog sinyal karakterindeki darbeler motorun dönmesini sağlar. Fırçasız motor maliyetlerinin yüksek olmasından dolayı bu çalışmada fırçasız motor kontrolü yapılmamıştır. Fakat darbe üretme algoritmalarında yapılacak bir ayarlamayla bu tipten motorların kontrolünde geliştirilen sistem kullanılarak yapılabilir. Bunun için sinüs fonksiyonunun belli periyotlardaki değerleri bilgisayar içerisinde oluşturulan bir tabloya kaydedilir. Daha sonra oluşturulacak bir döngü ile belirlenen bir hızda bu veriler okunarak, belirlenen bir katsayı ile çarpılır elde edilen

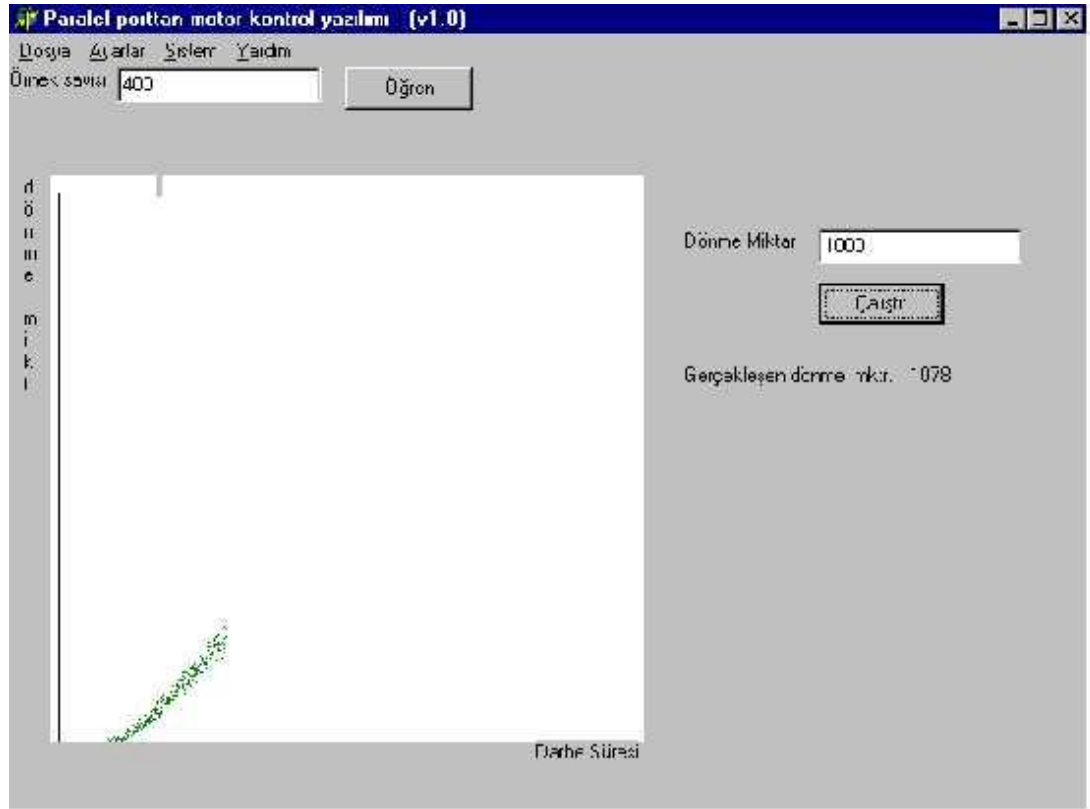
sonuca görede analog karakterde darbeler üretilerek güçlendirici üzerinden motora uygulanır.

Tork ayarlarında hız ayarlarına benzer. Tork ayarlarında da PWM sinyallerinden yararlanılır. Motora darbeleri sinyaller verilerek impulsif etki yaratılması yoluyla motorun torku artırılabilir.

Bilgisayar teknolojilerindeki hızlı gelişmeye paralel olarak bilgisayarların ürettikleri darbe kaliteleride artmaktadır. Bu da, hergeçen gün standart bilgisayarların herhangi bir ek donanım olmadan endüstriyel uygulamalarda kullanılmalarının önünü biraz daha açmaktadır. Bu projede geliştirilen sistemin, hızlı işlemciye ve donanıma sahip bilgisayarlarda daha kesin ve tatmin edici sonuçlar vereceği açıktır. Bu çalışmada bilgisayarın standart bir donanım aygıtı olan paralel portun kullanılması bu açıdan bakıldığında da oldukça önemlidir.

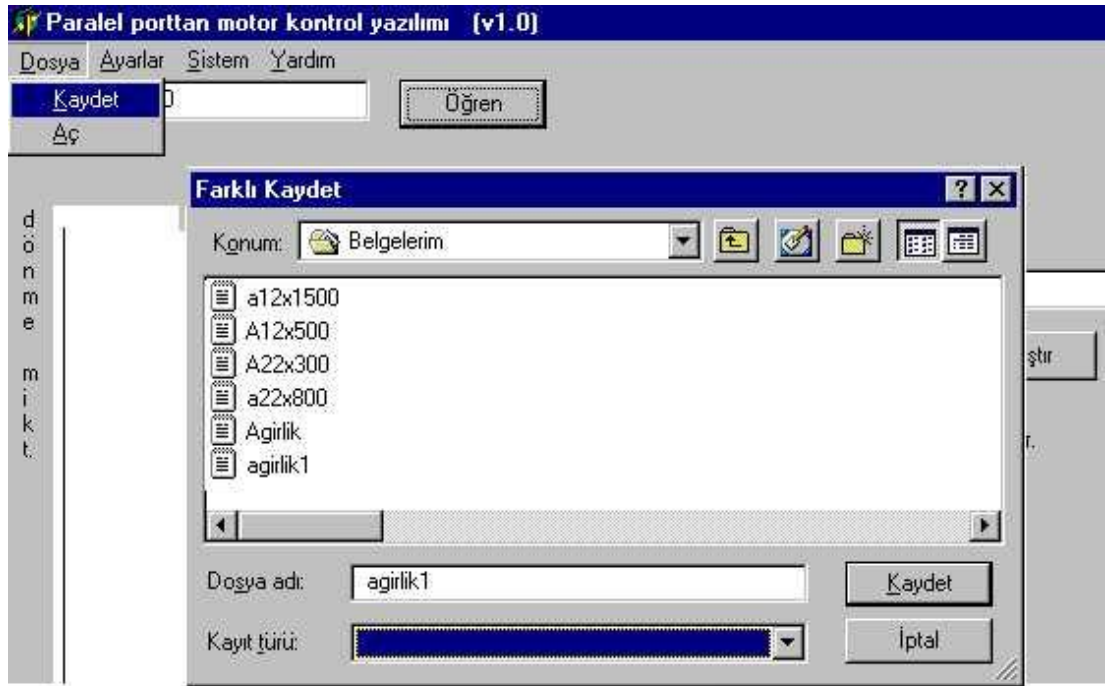
### 8.3 Kullanıcı Arayüzü

Program çalıştırıldığında kullanıcı resim-8.6'da görülen formla karşılaşmaktadır. Menülerden gereken ayarlar yapıldıktan sonra Eğitimde kullanılacak örnek sayısı belirlenerek **Örnek Sayısı** boşluğuna girilir. Daha sonra öğren düğmesine basılmasıyla program rastgele genişlikte darbeler üretmeye başlar. Form üzerine yerleştirilen grafikten darbe genişliğine karşılık gelen dönme miktarları görülebilir. Eğitim sürecinin tamamlanmasında sonra motordan istenen dönme miktarı, **Dönme Miktarı** boşluğuna yazılır. Daha sonra form üzerinde bulunan **Çalıştır** düğmesine basılarak motorun istenen miktarda dönmesi sağlanır. Gerçekleşen dönme miktarı form üzerinde **Gerçekleşen dönme mikt.** ifadesinde sonra görülür. İşlemciye ulaşan çeşitli kesme sinyallerinden dolayı işlemcinin yeterince hassas sinyal üretememesinden dolayı motor belli sapmalarla istenen dönme miktarına yaklaşmaktadır. Şekil 8-6' daki anaform üzerindeki grafikten de, bilgisayarın ürettiği sinyaller arasındaki farktan kaynaklanan farklı dönme miktarları görülebilir.



**Resim 8.6** Ana Form

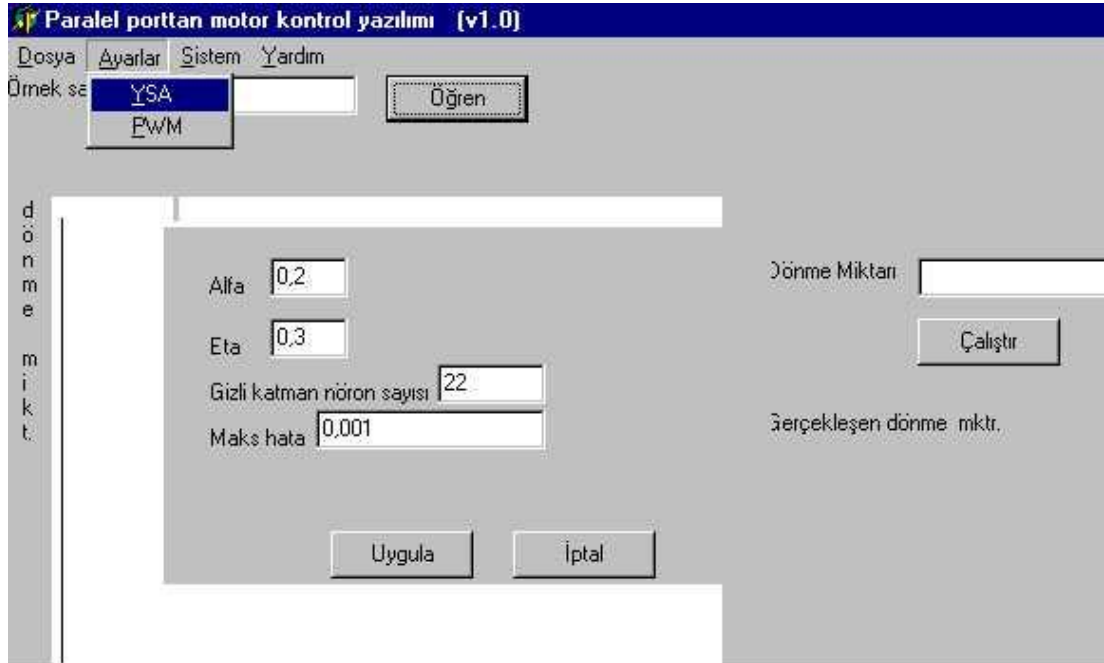
Anaform üzerinde bunlara ek olarak çeşitli menü seçenekleri vardır şimdi bu menülere bakalım



**Resim 8.7** Dosya işlemleri

Menü elemanlarından ilki **Dosya**'dır. **Dosya** menü elemanı Kaydet ve Aç adında iki alt elemanı daha içerir. Kaydet ifadesi seçildiğinde resim 8.7'de görülen Windows kaydetme formu açılır. Bu form kullanılarak,YSA'nın eğitim işlemi tamamlandıktan sonra ağırlıkları dosyaya kaydetme işlemini gerçekleştirebiliriz. Daha sonra yine aynı menüden Aç ifadesini seçerek açılan Windows dosya açma formu yardımıyla daha önce kaydetmiş olduğumuz ağırlık bilgilerini tekrar yükleyerek aynı öğrenme işlemini tekrarlamaktan kurtulmuş oluruz.

İkinci menü elemanımız **Ayarlar**'dır. Ayarlar menüsünde **YSA** ve **PWM** olmak üzere iki alt elemandan oluşur. **YSA** seçeneği seçildiğinde, YSA ayarları ile ilgili resim 8.8'deki pencere açılır. Bu pencerede YSA algoritmalarında kullanılan alfa, öğrenme oranı katsayısı, eta, atalet katsayısı, Emax, maksimum hata gibi parametreler yanında, 3 katmandan oluşan YSA'nın gizli katman nöron sayısını ilgili boşluklara gireriz. Girdiğimiz değerlerin YSA algoritmalarına aktarılmasını istersek pencere içindeki **Uygula** düğmesine basmamız yeterli olur. Eğer yaptığımız değişikliklerin YSA algoritmalarına aktarılmasını istemezsek **İptal** düğmesine basmamız yeterli olur.



**Resim 8-8** YSA ayarları penceresi

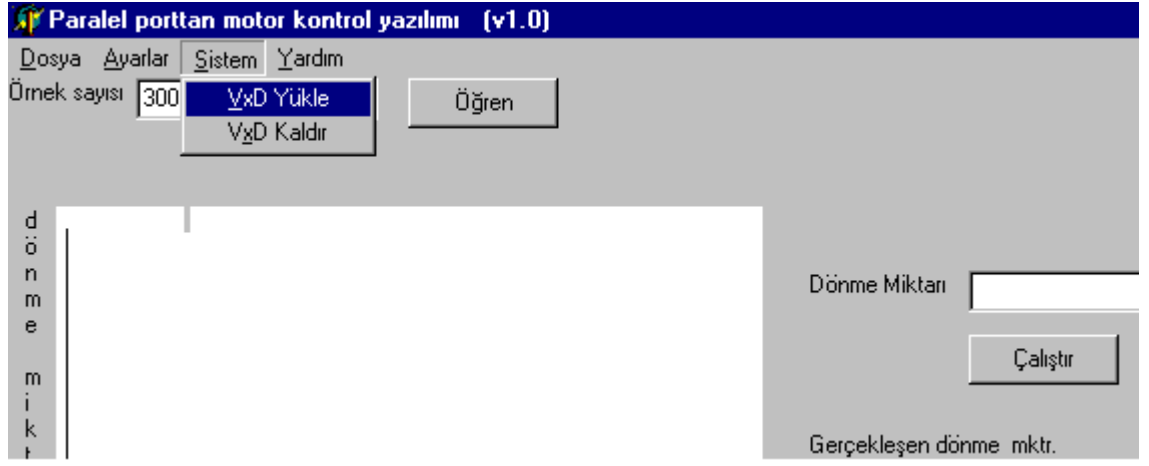
Ayarlar menüsünde bulunan elemanlardan bir diğeri PWM'dir. PWM menüsü seçildiğinde ise Resim 8.8'da görülen pencere açılır.



**Resim 8-9** PWM ayar penceresi

Bu pencerede PWM ile motorun hız ve ivme özellikleri ile dönme yönü özellikleri ayarlanır. Darbe genişliği ile pozitif darbelerin periyodları ayarlanırken, hız ayarı ile sıfır darbelerinin periyod değerleri ayarlanır. Motorun dönme yönünde yine bu pencere kullanılarak ayarlanabilir.

Bir diğer menü, sistem menüsü de, VxD yükle ve VxD kaldır olmak üzere iki alt elemandan oluşur(Resim 8-10). VxD yükle seçeneğini seçmemiz durumunda tasarlamış olduğumuz ve Windows sistem dosyasına kaydettiğimiz paralel port sanal sürücüsü dinamik olarak yüklenir. VxD kaldır seçeneğini seçmemiz durumunda ise dinamik olarak yüklenen VxD dinamik olarak kaldırılır ve paralel port tekrar eski standart ayarlarına döner.



Şekil 8.10 VxD yükleme-kaldırma menüsü

## 9. SONUÇ

Paralel portun Windows altında, interrupt sinyallerine açılmasıyla, yüksek hızda, güvenilir veri alınması sağlanmıştır. Elde edilen veriler, YSA eğitiminde kullanılmıştır. Ancak, bilgisayarın işlemcisine her interrupt gelişinde, diğer işlemlerin kesintiye uğraması, anakart üzerindeki zamanlayıcıyı kullanmış olmamıza rağmen, istediğimiz hassasiyette sinyaller üretmemize engel olmuştur.

Daha yüksek hızda işlemciler kullanılarak bu hatanın oranının azaltılması sağlanabilir. Yapılan çalışmada motorun 0.8 derecelik adımlarla dönmesi sağlanmıştır. Fakat üretilen sinyallerin bilgisayar içindeki interrupt sinyallerinden etkileniyor olması bu hassasiyette sürekli kontrolün sağlanmasına engel olmaktadır. Üretilen sinyallerin hatası, motorun dönüşünde  $\pm 180^\circ$  lik hataya sebep olmaktadır.

Üretilen sinyalin hassasiyetini artırmak için, pozisyon okuma yada sinyal üretimi işlemlerinden birisinin bilgisayardan alınarak hazırlanacak bir devre yardımıyla halledilmesi yolu seçilebilir. Fakat sağlıklı çözüm için, sinyal üretiminin ilave bir devre yardımıyla yapılması daha uygun olacaktır. Çünkü pozisyon okumadan kaynaklanan interruptlar dışında da, daha pek çok interrupt işlemciye ulaşarak, hassas sinyal üretimini engellemektedir. Hatalı gelen sinyallerde YSA'nın eğitim aşamasında hatalı veriler kullanılmasına sebep olmakta ve eğitim aşamasında üretilen her hatalı sinyal YSA'nın istenen hassasiyete ulaşmasına engel olmaktadır. Epson firmasının PC tabanlı robot kontrolörü SRC500'de de benzer bir yol izlenmekte ve motor kontrol sinyalleri ayrı bir frekans üretici yardımıyla üretilmektedir.

Bu amaçla 16F84 nolu PIC tipinde mikroişlemci ile yüksek hassasiyette sinyaller üretilmeye çalışılmaktadır. Bu şekilde üretilen darbe sinyallerinin hassasiyetinin artacağı ve hem daha doğru verilerle eğitilen YSA'nın daha iyi neticeler vereceği hem de daha kaliteli PWM kontrolün mümkün olacağı düşünülmektedir.

Yapılan çalışmanın windows işletim sistemi uyumlu olması daha etkin kullanıcı arayüzü geliştirilmesine imkan vermektedir. Bunun yanında, PC veri işleme kartı kullanılmaması sayesinde, sistem dizüstü bilgisayarlarla da kullanılabilir. Sistem, endüstriyel kontrol uygulamalarına yönelik olarak tasarlanmış olsa da, sistemin, dizüstü bilgisayarlarla kullanıma uygun olması açık arazi çalışmalarında da kullanılabilmesini sağlamaktadır.. Mesela jeolojik araştırmalar için açık arazi çalışmalarında veri toplama çalışmalarında dizüstü bilgisayar kullanımı gerekli olduğu durumlarda tasarlanan yazılımın kesme sinyalleriyle veri alma kısmı kullanılabilir.

Sonuç olarak, bu çalışmada kullandığımız kesme sinyalleriyle veri okuma tekniği, yüksek hızda ve güvenilir veri okumanın zorunlu olduğu her alanda kullanılabilir. Üretilen sinyal hassasiyetinin de artırılmasıyla oluşturulan sistem endüstriyel kontrol uygulamalarında kullanılabilir hale gelecektir.

## KAYNAKLAR

- (1) Karen Hazzah, "Writing Windows VxD's and Device Drivers," *R&D Books*, 1997
- (2) K.Kadir ŞAFAK, "Identification of UMD C Dynamics Using Feedforward Neural Networks," Boğaziçi Üniversitesi, 1997
- (3) Martin T.Hagan, Mohammad B.Menjaj, "Training Feedforward Networks with the Marquard Algorithm," *IEEE Transactions on Neural Networks*, Vol.5, No.6, November 1994, pp.989-993
- (4) Weerasooriya, S., El-Sharkawi, M.A., "Identification and Control of a DC Motor Using Back-Propagation Neural Networks," *IEEE Trans. On Energy Conversion*, Vol. 6, No.4, pp.663-669, 1992
- (5) Peter H. Anderson, "Use of a PC Printer Port for Control and Data Acquisition.," <http://et.nmsu.edu>, Fall, 1996
- (6) Justin Deltener, "Timer Interrupt Programming," <http://www.inversereality.org>
- (7) Bob Blick, "H-Bridge," <http://bobblick.com>
- (8) Zhahai Steward, "Interfacing the IBM PC Parallel Printer Port," <http://nyx10.cs.du.edu:8001/~zstewart/parport.html> January, 1994

- (9) VxD tutorials, Iczelion's tutorial series, <http://win32asm.cjb.net>
- (10) Hsieh, Long-Bing-Benson, "A Study of the accuracy and reliability of traffic recording devices(vehicle signatures)," Ohio Universitesi Doktora Tezi, 1996

**EK-I**

**.386p**

**include C:\masm32\include\mmm.inc**

**include C:\masm32\include\vwins32.inc**

**include C:\masm32\include\v86mmgr.inc**

**include C:\masm32\include\VPICD.inc**

**VxDName TEXTEQU <VXDINT>**

**ControlName TEXTEQU <VXDINT\_Control>**

**VxDMajorVersion TEXTEQU <1>**

**VxDMinorVersion TEXTEQU <0>**

**VxD\_LOCKED\_DATA\_SEG**

**IRQHANDLE dd ?**

**LPT\_IRQ equ 7**

**irq\_desc VPICD\_IRQ\_Descriptor<LPT\_IRQ,0,offset32 Sayac>**

**sayacgost dd ?**

**VxD\_LOCKED\_DATA\_ENDS**

**VXD\_LOCKED\_CODE\_SEG**

```
DECLARE_VIRTUAL_DEVICE  
%VxDName,%VxDMajorVersion,%VxDMinorVersion,  
%ControlName,UNDEFINED_DEVICE_ID,UNDEFINED_INIT_ORD  
ER
```

```
BeginProc %ControlName
```

```
Control_Dispatch W32_DEVICEIOCONTROL, OpenPort
```

```
Control_Dispatch Sys_Dynamic_Device_Init,  
VIMED_Device_Init
```

```
Control_Dispatch Sys_Dynamic_Device_Exit,  
VIMED_Device_Exit
```

```
clc
```

```
ret
```

```
EndProc %ControlName
```

```
BeginProc OpenPort
```

```
assume esi:ptr DIOCParams
```

```
.if [esi].dwIoControlCode==DIOC_Open
```

```
xor eax,eax
```

```
mov sayacgost,00h
```

```
.elseif [esi].dwIoControlCode==2
```

```
Push cx
```

```
Push dx
```

```
mov edi,[esi].lpvOutBuffer
```

```
mov eax,sayacgost
```

```
mov [edi],eax
```

```
xor eax,eax
```

```
Pop cx
```

```
Pop dx
```

```
.endif
```

```
clc
```

```
ret
```

```
EndProc OpenPort
```

```

BeginProc VIMED_Device_Init
    mov edi,offset32 irq_desc
    VXDCall VPICD_Virtualize_IRQ
        mov [IRQHandle],eax
        clc
        xor eax,eax
        ret
EndProc VIMED_Device_Init
BeginProc VIMED_Device_Exit
    mov sayacgost,00000004h
    mov  eax, IRQHANDLE
    VxDcall VPICD_Force_Default_Behavior
        clc
        ret
EndProc VIMED_Device_Exit
BeginProc Sayac
    pushfd
    pushad
    inc sayacgost
    popad
    popfd
    VxDCall VPICD_Phys_EOI
        clc
        ret
EndProc Sayac
VXD_LOCKED_CODE_ENDS
End

```

## **ÖZGEÇMİŞ**

Ufuk BEKÇİ, 1976 yılında Isparta’da doğdu. İlk ve orta öğrenimini Isparta’da tamamladı. 2000 yılında İstanbul Teknik Üniversitesi Uçak ve Uzay Bilimleri Fakültesinden uçak mühendisi olarak mezun oldu. Aynı yıl İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü Uçak Mühendisliği Ana Bilim dalında yüksek lisansa başladı.