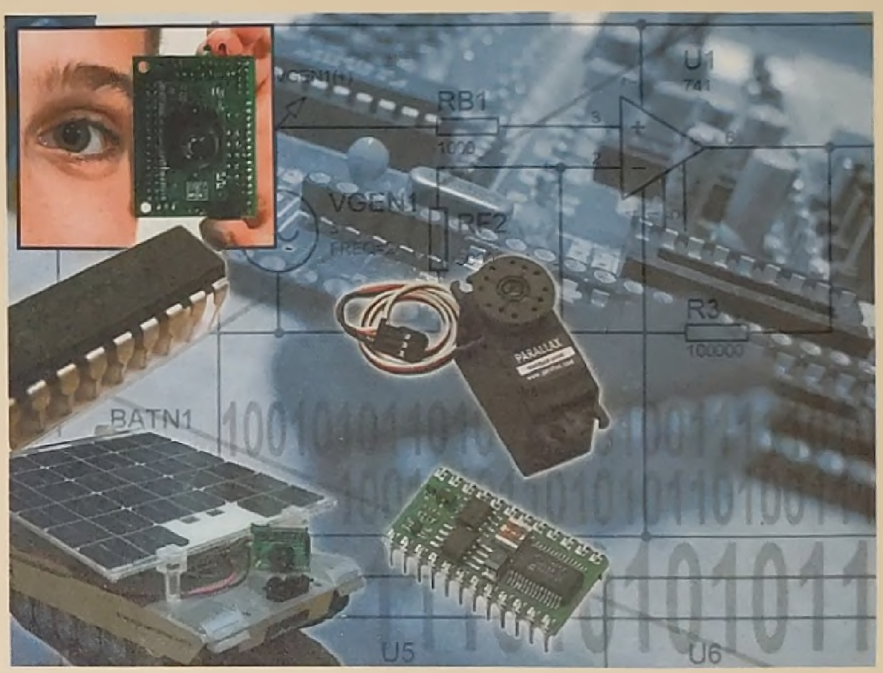
Bilgisayarlar ve mikroişlemciler ile dış dünyaya hükmedin!

- Elektronik hobi dünyasına ilk adımı atın.
- Bilgisayarınızla evinizdeki cihazları kontrol edin.
- Bilgisayar ve mikroişlemci elektroniğini uygulayarak öğrenin.
- Kendi robotunuzu tasarlayın.
- Paralel port uygulamaları
- Adım motor, servo motor ve DC motor kontrolleri
- LCD panel ve röle kontrolleri
- Bilgisayar kontrollü hırsız alarmı, termometre, robot gözü ve model araba
- Internet üzerinden ev otomasyonu
- LINUX ile otomasyon
- PIC programlama ve uygulamaları
- PIC Basic uygulamaları ve BASIC tabanlı mikroişlemciler
- Seri port uygulamaları
- Radyo ve kızıl ötesi kontrollü uygulamalar
- Baskı devre teknikleri
- Çizgi izleyen robot tasarımı
- Disket sürücüden basit robot tasarımı
- Mobil Araştırma Robotu projesi

Elektronik Hobi



Güçlü Tuğay

13. BASKI

Herkes İçin!®

ALFA

Herkes İçin

Elektronik Hobi

Güçlü Tuğay

Editör: Selçuk Tüzel

ALFA

Alfa Yayınları **1461**
Bilgisayar Dizisi **292**

Herkes İçin
ELEKTRONİK HOBİ

Güçlü Tuğay
Editör Selçuk Tüzel

1. Basım : 2004
13. Basım : Kasım 2011
ISBN : 978-975-297-481-4
Sertifika No : 10905

Yayıncı ve Genel Yayın Yönetmeni M. Faruk Bayrak
Genel Müdür Vedat Bayrak
Yayın Yönetmeni ve Editör Rana Alpöz
Dizi Editörü Mehmet Çömlekçi
Teknik Editör Zeynep Çömlekçi

© 2004, ALFA Basım Yayım Dağıtım San. ve Tic. Ltd. Şti.

Kitabın Türkçe yayın hakları Alfa Basım Yayım Dağıtım San. ve Tic. Ltd. Şti.'ne aittir. Yayınevinden yazılı izin alınmadan kısmen ya da tamamen alıntı yapılamaz, hiçbir şekilde kopya edilemez, çoğaltılamaz ve yayımlanamaz.

Baskı ve Cilt
Melisa Matbaacılık
Tel: (212) 674 97 23 Faks: (212) 674 97 2
Sertifika No :12088

Alfa Basım Yayım Dağıtım San. ve Tic. Ltd. Şti.
Ticarethane Sokak No: 53 34110 Cağaloğlu İstanbul, Turkey
Tel: (212) 511 53 03 - 513 87 51 - 512 30 46 Faks: (212) 519 33 00
www.alfakitap.com
info@alfakitap.com

UYARI:
Bu kitap, bilgisayar kontrollü elektronik uygulamalar içermektedir.
Bu uygulamaları gerçekleştirirken, ortaya çıkabilecek elektriksel sorunlar sonucu, kullanılan bilgisayar veya devre hasar görebilir.
Bu nedenle, gerçekleştirilen tüm uygulamalarda sorumluluk, uygulayan kişiye aittir.
Oluşabilecek herhangi bir zarar durumundan yazar ya da yayınevi sorumlu tutulamaz.

İçindekiler

KISIM I: BİLGİSAYAR KONTROLLÜ UYGULAMALAR 1

BÖLÜM 1: PARALEL PORT 3

1'ler ve 0'lar 4
Paralel Port Hakkında 4
Paralel Portun Yapısı 4
Neden 8 Tane Toprak Hattı Var? 5
Paralel Portun Adresleri 6
Paralel Port ile Veri Çıkışı 7
2'lik (Binary) Sayı Sistemi 7
LED Kontrolü Uygulaması 9
Board'lar 10
LED Diyotlar 10
Dirençler 11
Paralel Port ile Veri Girişi 14
Tuş Butonlar 14
STATUS Portu ile Veri Girişi Uygulaması 15
XOR Kullanımı 16
Onaltılık (Hexadecimal) Sayı Sistemi 17

BÖLÜM 2: PARALEL PORT İLE RÖLE KONTROLÜ 19

Röle Kontrol Uygulaması 20
Röleler 20
Transistörler 21
Diyotlar 21
Röle Kontrol Devresi 22
Röle Kontrol Yazılımı 24
Röle Kontrol Devresi Nasıl Çalışıyor? 24

BÖLÜM 3: ADIM (STEP) MOTOR KONTROLÜ 27

Adım Motorlar 28
Adım Motor Kontrolü Uygulaması 28
Adım Motor Entegreli Sürücü Devresi 28
Adım Motor Kontrolü Devresi Nasıl Çalışıyor? 28
Transistörlü Adım Motor Sürücü Devresi 29
Adım Motor Kontrolü Yazılımı 30

BÖLÜM 4: LCD PANEL KONTROLÜ 33

LCD Paneller 34

LCD Panel Kontrol Uygulaması 34
LCD Panel Kontrol Devresi Nasıl Çalışıyor? 36
LCD Panel Kontrol Yazılımı 37

BÖLÜM 5: ANALOG GİRİŞ VE BASİT ROBOT GÖZÜ 41

Analog Sinyaller 42
Sayısal-Analog Dönüşüm Uygulaması 42
Sayısal - Analog Dönüştürücü (ADC) Devresi 43
ADC Devresi Nasıl Çalışıyor? 44
Cismin Uzaklığını Işık Şiddeti ile Algılamak 45
Basit Robot Gözü Uygulaması 45
Foto Transistörler 46

BÖLÜM 6: DC MOTOR KONTROLÜ 49

DC Motorlar 50
DC Motor Kontrol Uygulaması 50
UDN2993 ile DC Motor Kontrol Devresi 51
L293B ile DC Motor Kontrol Devresi 54

BÖLÜM 7: BİLGİSAYAR KONTROLLÜ MODEL ARABA 57

Radyo Sinyalli Model Arabalar 58
Paralel Porttan Model Araba Kontrol Uygulaması 58
Model Araba Kontrol Yazılımı 61

BÖLÜM 8: SERİ İLETİŞİM 65

Seri Port 66
Seri Portun Avantajları 66
Seri Port ile Veri Aktarımı 66
UART Tümdevresi 67
Kesmelerin (IRQ) Kullanımı 67
RS232 Standardı 68
PC ile Modemin Haberleşmesi 69
Kızıl Ötesi İletişim 72

BÖLÜM 9: TEMEL MANTIK KAPILARI 73

Mantık Kapıları 74
AND Kapısı 74
OR Kapısı 74
Ters Çevirici (Inverter): 75
NAND Kapısı 75
NOR Kapısı: 75
XOR Kapısı: 75

BÖLÜM 10: INTERNET ÜZERİNDEN EV OTOMASYONU 79

Ev Otomasyonu 80
İnternet Üzerinden Ev Otomasyonu Uygulaması 80
TCP/IP ile Röle Kontrol Programı 82
TCP/IP ile Röle Kontrol Programı Nasıl Çalışıyor? 82

BÖLÜM 11: BASKI DEVRE TEKNİKLERİ 89

Baskı Devreler 90
P-N-P Baskı Devre Uygulaması 91

BÖLÜM 12: LINUX İLE OTOMASYON 97

Linux ve Paralel Port 98
Linux'ta Metin Editörü 98
Linux'ta C Kodu Derlemek 99
C++ ile Paralel Port Kontrolü 99

BÖLÜM 13: PC KONTROLLÜ TERMOMETRE 103

PC Kontrollü Termometre Uygulaması 104
Termometre Devresi 104
PC Kontrollü Termometre Yazılımı 105

BÖLÜM 14: LAZER İLE HIRSIZ ALARMI 107

Lazerler 108
Lazer ile Hırsız Alarmı Uygulaması 108

BÖLÜM 15: RADYO SİNYALİ İLE BEDAVA CHAT 111

Radyo Sinyali ve Mors Alfabesi ile İletişim 112
Radyo Sinyali ile Chat Uygulaması 112
Verici Kontrolü 113
Alıcı Kontrolü 114

BÖLÜM 16: CMUCam ELEKTRONİK GÖZ 117

Görüntü Algılama 118
CMUCam 118
CMUCam'in Terminal Üzerinde Kullanımı 119
CMUCam Kontrol Yazılımı 120

KISIM II: PIC PROGRAMLAMA 123

BÖLÜM 17: PIC PROGRAMLAMAYA GİRİŞ 125

PIC Programlama 126
PIC Nedir? 126

PIC Programlama ve Deneme Kartı 127
PIC16F84'ün Uçları 127
Vss (5) ve Vdd (14) Uçları (Besleme) 128
OSC1/CLK IN (16) ve OSC2/CLK OUT (15) (Clock Giriş ve Çıkışları) 128
Giriş/Çıkış Portları (RA0-RA3 ve RB0-RB7) 128
Reset Devresi ve Ucu (4) 128

BÖLÜM 18: LED KONTROLÜ 129

PIC Assembly 130
PIC ile Led Kontrol Uygulaması 130

BÖLÜM 19: BUTONA BASINCA SÖNEN LED 139

Butona Basınca Sönen Led Uygulaması 140

BÖLÜM 20: SIRALI YANAN LED'LER 145

Sıralı Yanan Led Uygulaması 146
Gecikme Nedir? Nasıl Yapılır? 146

BÖLÜM 21: LCD PANEL KONTROLÜ 149

LCD Paneller 150
PIC ile LCD Panel Kontrol Uygulaması 152

BÖLÜM 22: PIC İLE ADIM MOTOR KONTROLÜ 161

Adım Motorlar 162
PIC ile Adım Motor Kontrol Uygulaması 163

BÖLÜM 23: PIC BASIC 167

PIC Basic'e Giriş 168
PIC Basic Uygulamaları 169
Uygulama I: PIC Basic ile Led Kontrolü 169
Uygulama II: PIC Basic ile LCD Panel Kontrolü 170
PIC BASIC ile RS232 Terminali 172

BÖLÜM 24: PIC BASIC İLE SERVO MOTOR KONTROLÜ 177

Servo Motor Nedir? 178
Seri Port ile Servo Kontrolü 179

BÖLÜM 25: BASIC TIGER 183

Tek Çipte Bilgisayar 184
BASIC Tiger ile LCD Panel ve LED Kontrolü 185

BÖLÜM 26: BASIC STAMP 187

BASIC Stamp'ler 188
BASIC Stamp ile Led Kontrol Uygulaması 189

KISIM III: ROBOTİK UYGULAMALAR 193

BÖLÜM 27: ÇİZGİ İZLEYEN ROBOT 195

Robotlar 196
Çizgi İzleyen Robot 197
Çizgi Nasıl Algılanır? 198
Comparator'ler (Kıyaslayıcı, Karşılaştırıcı) 198
Robotun Mekaniği 199
Çizgi Sensörü Devresi 202
Voltaj Regülatörü Devresi 204
Motor Kontrol Devresi 204
PIC Çalıştırma Devresi 205
Devre ve PIC Programı Nasıl Çalışıyor 206

BÖLÜM 28: MOBİL ARAŞTIRMA ROBOTU PROJESİ 211

Giriş 212
MAR Projesi 212
Ana Modüller 212
Performans Sorunları 213
MAR Mekaniği 215
MAR Elektroniği 216
BASIC-Tiger Mikro İşlemci Devresi 217
UKS Radyo Modem Devresi 217
L293B Motor Kontrol Devresi 220
Cisim Algılama Devresi 221
Pil Şarj ve Enerji Devresi 222
Kamera Yönü İçin Servo Motor Kontrolü 223
Kamera Devresi 223
MAR Yazılımı 224
MAR Komut Yorumlayıcısı ve Komut Seti 226
MAR Terminal Arayüzü ve Projenin Devamı 227

BÖLÜM 29: FLOPPY ROBOT 229

Teşekkür

Kısım 3'te yer alan MAR (Mobil Araştırma Robotu) projesinin gerçekleşmesinde bana danışmanlık yaparak her türlü bilgi, katkı, zaman ve desteğini esirgemeyen İstanbul Üniversitesi Astronomi ve Uzay Bilimleri Bölümü öğretim üyelerinden, Prof. Dr. Adnan ÖKTEN'e,

MAR projesine bulunduğu katkılardan ötürü Dimitri STOYANOF'a,

MAR projesi ve kitaptaki diğer proje ve uygulamalara bulunduğu katkılardan ötürü Ergem PEKER'e,

MAR projesinde bana verdiği destekten ötürü İstanbul Üniversitesi Astronomi ve Uzay Bilimleri Bölümü öğretim üyelerinden Doç. Dr. Talat SAYGAÇ'a

Teşekkürlerimi sunarım.

Uyarı

Bu kitap, bilgisayar kontrollü elektronik uygulamalar içermektedir. Bu uygulamaları gerçekleştirirken, ortaya çıkabilecek elektriksel sorunlar sonucu, kullanılan bilgisayar veya devre hasar görebilir. Bu nedenle, gerçekleştirilen tüm uygulamalarda sorumluluk, uygulayan kişiye aittir. Oluşabilecek herhangi bir zarar durumundan yazar ya da yayınevi sorumlu tutulamaz.

Osman Kerim Erverdi'nin anısına...

Yazar Hakkında

1976'da İstanbul'da doğdum. Her şey, 9 yaşında Commodore 64 ile tanışmamla başladı. Kendi kendime C64 Basic'i öğrenip, 12 yaşında çalışan ve işe yarayan ilk programımı yazdım. Daha sonra, 1988 yılında Amiga ile tanıştım ve AMOS Basic ile program yazmaya devam ettim. O dönemin, "64'ler" ve "Amiga Dünyası" dergilerinde makalelerim yayınlandı. 1990 yılında bir yandan PC'lere ve GWBASIC kullanmaya merak saldım. 1992 yılında PC ile PASCAL ve C++ programları üzerinde çalıştım ve C++ ile birçok program geliştirdim. Mikroişlemci programlamaya o dönemlerde başladım ve amatör olarak Assembly programları yazdım. Daha sonra, kendi firmamda Visual Basic ve SQL veritabanı uygulamaları üzerinde program geliştirmeye başladım. Ardından, bir firmada Network ve Windows NT üzerine eğitim verdim, 1997'ye kadar çeşitli firmalarda Network sistemleri üzerine çalıştım ve bir yandan Visual Basic ile bazı firmalara özel yazılım geliştirmeye devam ettim. 1998 yılında kendi şirketimi kurdum. Emlak firmaları için EMLAKBOX adında bir program geliştirdim ve `www.emlakbox.com` sitesini kurdum. Bu sıralarda elektronik ve robotlarla ilgilenmeye başladım ve bilgisayar elektroniğini öğrendim. 2001 yılında, geliştirdiğim elektronik uygulamalarını PC LIFE dergisinde köşe haline getirmek için bir teklif aldım ve "Elektronik Hobi" köşesi bu yıl oluştu. Köşe, tahmin ettiğimizden çok daha fazla ilgi gördü. Daha sonra yazılara BYTE dergisinde devam ettim ve bugün yaklaşık 7000 üyesi olan `www.elektronikhobi.com` sitesini kurdum. Bu dönemlerde bir yandan okuduğum İstanbul Üniversitesi Astronomi ve Uzay Bilimleri Bölümü'nü bitirdim ve bu kitapta da yayınladığım bir Uzay Araştırma Robotu yaptım. Şu anda yazılım, Web tasarımı, otomasyon, e-ticaret uygulamaları, sistem destek ve hosting hizmetlerini vermeye devam eden kendi şirketimde yazılım projeleri ile ilgilenmekte ve aynı zamanda BYTE dergisinde "Elektronik Hobi" köşesini yazmaya devam etmekteyim.

Önsöz

Bu kitap, Mart 2001'den itibaren PC LIFE dergisinde, Haziran 2002'den itibaren ise BYTE dergisinde sürdürdüğüm ve hala devam eden "Elektronik Hobi" köşesi uygulamalarından derlenmiştir.

Birçok kimse, bilgisayar elektroniğini öğrenmenin ve bilgisayar ya da mikroişlemciler ile dış dünyaya hükmedebilmenin çok ileri düzey yazılım ve elektronik bilgisi gerektirdiğini düşünür. Gerçekten de, birçok kaynak kitap, bu konuları ileri düzeyde anlatmaktadır. Çoğu zaman bir uygulamaya kalkıştığınızda, karşınıza elektronikle ya da yazılımla ilgili birçok terim çıkar. Bu da, daha uygulamaya başlamadan okuyucunun sıkılmasına ve vazgeçmesine neden olur.

Bu nedenle, bu kitap farklı olarak doğrudan uygulamalarla başlamaktadır. Gerekli teorik bilgiler ise uygulama içerisinde gerektiği zamanda verilmiştir. Kitap içerisinde ilerledikçe evinizdeki elektronik cihazları bilgisayar ile kontrol etmekten tutun, motor, LCD panel, model araba kontrollerinden, PIC mikroişlemci programlamaya ve kendi robotunuzu yapmaya kadar ilerleyen birçok uygulama bulacaksınız. Bilgisayar ve mikroişlemci aleminde sözü geçen "1" ve "0" değerleri ile neler yapabildiğinize inanamayacaksınız. Yıllardır öğrenmek istediğiniz ve nereden başlayacağınızı bir türlü bilemediğiniz bilgisayar ile mikroişlemci elektroniğine ve "Elektronik Hobi" dünyasına bu kitap sayesinde giriş yapmış olacaksınız.

Uygulama Dosyaları

Kitap içerisinde çoğunlukla Visual Basic uygulamaları vardır. Her ne kadar kodların ne iş yaptığı açıklansa da, Visual Basic'i anlatmak bu kitabın işi değildir. Bu nedenle, bazı uygulamalardaki kodları kavrayabilmek için, Visual Basic'e hakim olmak gereklidir.

Kitap içerisinde yer alan bütün uygulamaların dosya ve kodlarını `http://www.elektronikhobi.com` adresinden indirebilirsiniz. Aynı zamanda site üyeleri tarafından yayınlanan birçok faydalı bilgi ve dokümana da bu siteden ulaşabilirsiniz. Kitabın buradan indireceğiniz uygulama dosyaları, kod ağırlıklıdır. Her ne kadar kodların derlenmiş ve uygulama haline getirilmiş `EXE` dosyaları bulunsa da, kodları derlemeden ve görmeden, uygulamalara hakim olmak mümkün olmayacaktır. Doğrudan `EXE` dosyaları ile çalıştırılan uygulamalarda, herhangi bir durumda hatanın yazılımdan mı, yoksa devreden mi kaynaklandığını anlamak mümkün olmayacaktır. Ayrıca, işletim sisteminden kaynaklanabilecek problemler `EXE` dosyalarında sıkıntı yaratabilir. Bu nedenle, mutlaka Visual Basic kodlarını Visual Basic 6.0 programı altında kendiniz derleyerek çalıştırınız. `EXE` dosyalarının işletim sisteminde çalışmaması durumunda, kullanılan işletim sistemine göre derlenmesi gibi bir hizmetin, yazar ya da yayınevi tarafından verilmeyeceğini unutmayınız.

Yazar ile İletişim

Bu kitap ile ilgili her türlü sorunuzu, `http://www.elektronikhobi.com` adresindeki forum üzerinden sorabilirsiniz.

BİRİNCİ KISIM

BİLGİSAYAR KONTROLLÜ UYGULAMALAR

BİRİNCİ BÖLÜM

1

PARALEL PORT

1'ler ve 0'lar

Bir bilgisayarın en temel elemanı, mikroişlemcisidir. Mikroişlemciler verileri "1" ve "0" değerleri üzerinden işlerler. Bunu genelde birçok bilgisayar kullanıcısı duymuştur, fakat bu değerlerin bize bir görüntüyü, sesi ya da benzeri bir veriyi nasıl oluşturduğu pek bilinmez. Bu iki değer, elektroniğin ve lojik programlamanın temelini oluşturur. Bu değerleri anlayabilmenin en güzel yolu, gözünüzle izleyebilmektir. Bu nedenle, ilk uygulamamızda bu değerlerle, bilgisayarımızın paralel portunu kullanarak bir çıkış elde edeceğiz ve gönderdiğimiz "1" ve "0"larla 8 tane ledi yakıp söndüreceğiz. Bu uygulamayı gerçekleştirmek, lojik programlamayı anlamanın en kolay yollarından biridir. Uygulamaya geçmeden önce paralel port ile ilgili ön bilgi vermemiz gerekmektedir.

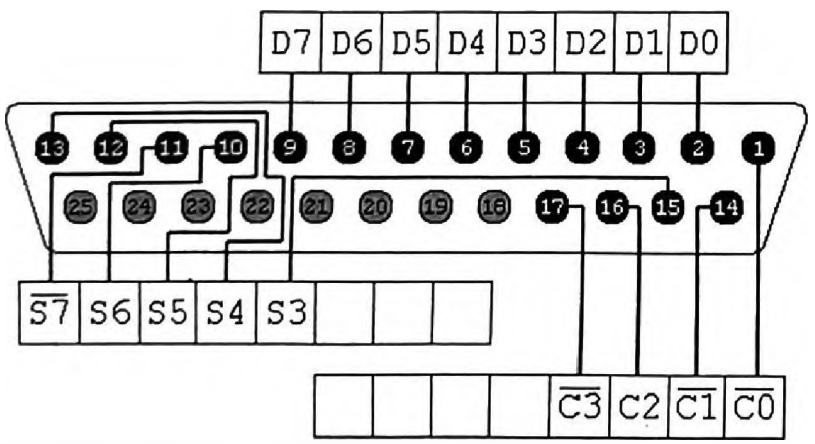
Paralel Port Hakkında

Paralel Portun Yapısı

Paralel port, bilgisayarın en kolay programlanabilen portudur. Üzerinde 25 pin vardır, aynı anda 8 BIT veri aktarabilir ve 5 Volt ile çalışır. Enerjisini bilgisayardaki anakarttan alır. Veri çıkışı port üzerindeki 2-9 numaralı pinler üzerinden elde edilir. Paralel port üzerinde yeri aktarımı için kullanılan bu pinlerin tümüne *DATA Portu* denir.

Data portu üzerindeki bu 8 pinin değerleri özel bir durum olmadığı sürece "0" dır. Bu pinlerden istediklerimizi "1" durumuna getirmek suretiyle istediğimiz çıkışı elde edebiliriz. Paralel port üzerindeki bir data pininin "1" olması, o pinin +5 Volt olması anlamına gelir. Elde edilen düşük amperli bu voltaja ise *lojik voltaj* denir. Bu voltaj, büyük bir elektronik eşyayı çalıştırmak için yeterli olmayabilir, ancak böyle bir cihazı çalıştırabilecek basit bir devreyi tetiklemek için yeterlidir.

Port üzerinde DATA portundan başka, STATUS ve CONTROL portları da vardır. Hangi pinlerin hangi porta ait olduğu Şekil 1.1'de gösterilmiştir. Bu portların dışında kalan 18-25 arası pinler ise toprak pinleridir. Portun en dışındaki metal kısım da toprak olarak kullanılır. Biz ilk uygulamamızda toprak hattı olarak en dıştakini kullanacağız. Bu hatların farklarına ilerleyen uygulamalarımızda değineceğiz.



ŞEKİL 1.1: Paralel port pinleri.

Veri Şekil 1.1'deki 2-9 arasındaki pinler üzerinden akacaktır. Bu pinler sırası ile Şekil 1.4'teki örnek program ekranında görüldüğü gibidir. Bizim diğer pinler ile şimdilik bir işimiz olmayacaktır. Yine de paralel portun diğer pinlerinin de ne işe yaradığını bilmek gerekir. Tablo 1.1'de, hangi pinin ne amaçla kullanıldığı belirtilmiştir.

TABLO 1.1: Paralel Portun Pinleri

Sinyal Adı	BIT	PIN	Yön
-Strobe	¬C0	1	Output
+Data Bit 0	D0	2	Output
+Data Bit 1	D1	3	Output
+Data Bit 2	D2	4	Output
+Data Bit 3	D3	5	Output
+Data Bit 4	D4	6	Output
+Data Bit 5	D5	7	Output
+Data Bit 6	D6	8	Output
+Data Bit 7	D7	9	Output
-Acknowledge	S6	10	Input
+Busy	¬S7	11	Input
+Paper End	S5	12	Input
+Select In	S4	13	Input
-Auto Feed	¬C1	14	Output
-Error	S3	15	Input
-Initialize	C2	16	Output
-Select	¬C3	17	Output
Ground -		18-25	Toprak

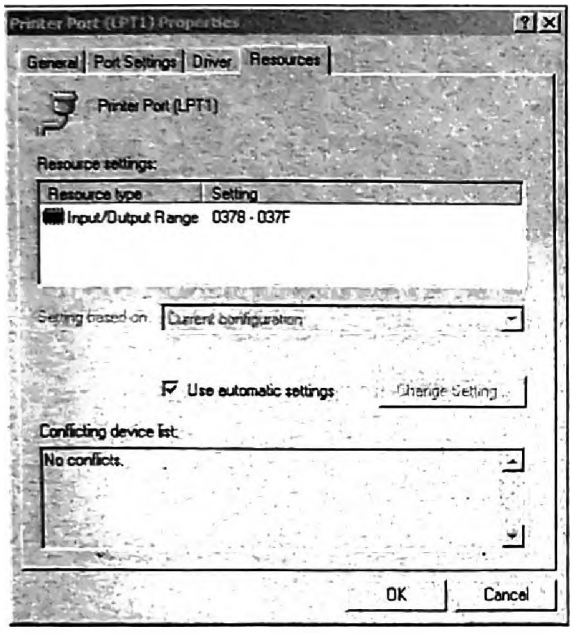
Neden 8 Tane Toprak Hattı Var?

Her devrede akım, kaynağına geri dönmek zorundadır. Genelde toprak hatlarının voltajı olmadığı düşünülür. Ancak her bir hattın bir empedansı vardır ve her kablodaki akım, bir voltaj meydana getirir (indükler). Bir toprak girişi çeşitli sinyaller tarafından paylaşıldığı zaman, her bir giriş diğerlerinin neden olduğu toprak voltajını fark edecektir. Toprak akımları en düşük direnç gösteren yolu takip edecektir. Dolayısı ile bir akımın belli bir telden geçmesini garanti etmek söz konusu olamaz. Çok sayıdaki toprak hattı toprak dönüşlerinin genel empedansını düşürecektir. Bu da toprak akımlarını küçültür. Paralel port üzerindeki 8 toprak hattından 7'sini kaldırıp hepsini tek bir tanesine bağlamak mümkündür. Bu durumda düşük hızda ve kısa mesafede arabirimin çalışması muhtemeldir. Tabii ki hassas uygulamalar ve arabirimler için 8 ayrı pinle topraklamayı tercih etmekte fayda vardır.

Paralel Portun Adresleri

Bilgisayar içerisindeki veri iletişimi, bilgisayarı oluşturan aygıtların adresleri üzerinden gerçekleşir. Bilgisayar içerisindeki her aygıtın bir adres aralığı vardır. Eğer paralel portun DATA portuna herhangi bir veri yerleştirmek istiyorsak, önce DATA portunun adresini bilmemiz gerekir.

Paralel portunuzun adresini öğrenmek için, Windows'ta da Denetim Masası (Control Panel) altından Sistem (System) seçeneğine girin. Oradan eğer 95 ya da 98 kullanıyorsanız doğrudan Aygıt Yöneticisi'ne (Device Manager) gelin. NT, 2000 ya da XP kullanıyorsanız, Donanım (Hardware) sekmesinden Aygıt Yöneticisi'ne (Device Manager) gelin. Daha sonra, Bağlantı Noktaları (Ports) altından Yazıcı Portu'nu (Printer Port) bulun ve çift tıklayarak özelliklerini açın. Yukarıdaki sekmelerden Kaynaklar (Sources) bölümüne gelin. Geldiğiniz ekran Şekil 1.2'dekine benzer olmalıdır.



ŞEKİL 1.2: Windows altında paralel portun adresi.

Adres aralığını şekilde işaretlenmiş yerlerden okuyabilirsiniz. Ekranda 0378-037F gibi bir değer aralığı görüyor olmalısınız. Bu değerler paralel portunuzun adres aralığıdır. İlk yazan değer portunuzun taban adresidir. Buna portun *Base Adresi* denir. Az önce bahsettiğimiz DATA portu, taban adresi ile aynıdır. Bu durumda taban (base) adresi 0378 ise;

DATA portu 0378,
STATUS portu 0378+1, yani 0379,
CONTROL portu ise 0378+2, yani 037A olur.

Dikkat ederseniz, burada yazan adres değerlerinin içerisinde rakamların dışında harfler de geçmektedir. Bunun sebebi, adres değerlerinin Hexadecimal (Onaltılık) sayı sisteminde gösteriliyor olmasıdır. Bölümün sonunda Hexadecimal sayı sistemi ile bilgi verilecektir.

Paralel Port ile Veri Çıkışı

DATA portunun adresini bildiğimize göre artık veriyi nereye göndereceğimizi biliyoruz demektir. Sıra, veriyi bu adrese nasıl göndereceğimize geldi. Windows altında istediğimiz herhangi bir adrese bir veri göndermek için Visual Basic kullanabiliriz. Önce göndereceğimiz verilerin port üzerine nasıl yerleştiğini inceleyelim.

Tüm verilerin "1"ler ve "0"lar şeklinde işlendiğinden bahsetmiştik. Bu rakamlar yan yana gelerek bize her türlü sayısal veriyi döndürebilir. Bunun için, sayı sistemlerinden kısaca bahsetmekte fayda vardır.

2'lik (Binary) Sayı Sistemi

Bizler günlük hayatımızda bu sayı sistemlerinden 10'luk taban olarak adlandırdığımız sayı sistemini kullanırız. Yani sayı sistemimiz 10 adet rakamdan oluşur (0-1-2-3-4-5-6-7-8-9). Bu anlamda, eğer 3'lük bir taban kullanacaksak, sayı sistemimiz en fazla 3 rakamdan oluşabilir; (0-1-2). Bu durumda, eğer sadece "1" ve "0" rakamlarını kullanacaksak bize 2'lik taban yeterli olacaktır. Her sayının 2'lik tabanda bir karşılığı vardır. DATA portuna göndereceğimiz değerler port üzerine 2'lik sistemdeki bir sayı gibi yerleşirler. DATA portuna hiçbir veri göndermediğimiz zamanki değeri "00000000"dir. Dikkat ederseniz 8 tane rakam vardır. Bunun sebebi portun 8 bit olmasıdır. Örneğin data portuna 25 değerini gönderelim. 25 değerinin ikilik sayı sisteminde karşılığı "00011001"dir. Bu durumda D4, D3 ve D0 pinlerine karşılık gelen lojik değerler "1" olduğundan, o pinler +5 Volt olacaktır. Bu pinlere "YÜKSEK duruma gelmiş" de denir. Tablo 1.2'de DATA portuna gönderdiğimiz bazı değerler ile hangi pinlerin "1" olacağını gösteren örnek bir tablo görüyorsunuz.

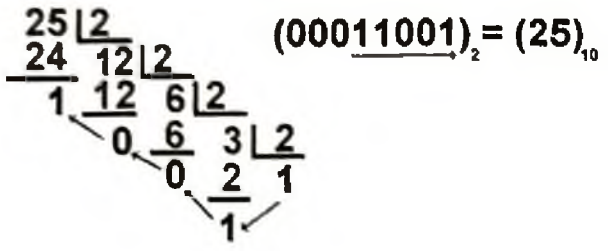
TABLO 1.2: Verilerin 2'lik Tabanda DATA Portuna Yerleşme Şekli

VERİ	D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	0	0	1
4	0	0	0	0	0	1	0	0
16	0	0	0	1	0	0	0	0
25	0	0	0	1	1	0	0	1
70	0	1	0	0	0	1	1	0

Bu dönüşümleri bilimsel bir hesap makinesi ile ya da Windows altındaki hesap makinesinin "Bilimsel" özelliğini kullanarak yapabilirsiniz. Visual Basic altından bu verileri direk 2'lik tabanda göndermek de mümkündür. Bu nedenle uygulamalarımızda, göndermek istediğimiz her veriyi formülde yerine koyup, 2'lik tabanda nasıl gözüktüğünü bilmemiz gerekmemektedir. Biz veriyi yine her zaman kullandığımız sayı sistemi olan 10'luk tabana göre göndermeye devam edeceğiz. Yine de 10'luk tabandan 2'lik tabana dönüşümün matematiksel olarak nasıl yapıldığını hatırlatmakta fayda vardır. T=Taban, K=Katsayı olmak üzere "n" basamaklı bir sayının taban dönüşümü aşağıdaki formül yardımı ile yapılabilir.

$$(\text{Sayı})_T = K_n T^n + K_{n-1} T^{n-1} + K_{n-2} T^{n-2} + \ldots\ldots + K_2 T^2 + K_1 T^1 + K_0 T^0$$

2'lik tabana dönüşüm için bu formülde temelde yapılan, sürekli 2'ye bölme işlemi sonucunda elde edilen kalanların sağdan sola okunmasıdır. "25" değeri için bu dönüşümün kağıt üzerinde nasıl yapıldığını Şekil 1.3'te görebilirsiniz.

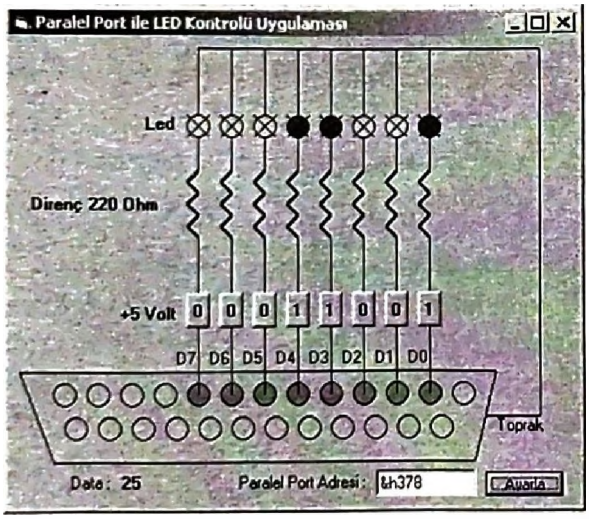


ŞEKİL 1.3: "25" değerinin kağıt üzerinde 2'lik tabana dönüşümü.

Gönderdiğimiz değerlerin DATA portu üzerine nasıl yerleştiğini öğrendiğimize göre, sıra bu değerlerin port üzerine nasıl gönderileceğine geldi. Bu işlem için Windows altında, Visual Basic kullanacağımızdan bahsetmiştik. Visual Basic, Windows altında uygulama geliştirilebilecek en kolay programdır. Visual Basic altından portları kontrol edebilmek için, ek bir dosyaya ihtiyacımız vardır. `inpout32.dll` adlı bu dosya, Windows'unuzun, 95, 98 ve ME için `System` klasörüne, NT, 2000 ve XP için `System32` klasörüne kopyalanmalıdır ya da uygulamalar ile aynı klasörde olmalıdır.

Tüm programlama dillerinde olduğu gibi, Visual Basic'te de işlemler komutlar aracılığı ile yapılır. Biz de DATA portuna istediğimiz veriyi, `OUT` komutu sayesinde gönderebiliriz. Bu komut, `OUT <DATA portunun adresi>, <Gönderilecek veri>` şeklinde kullanılır. Aşağıda, bu komutun kullanımına dair bir örnek görebilirsiniz:

```
OUT &h378,25
```



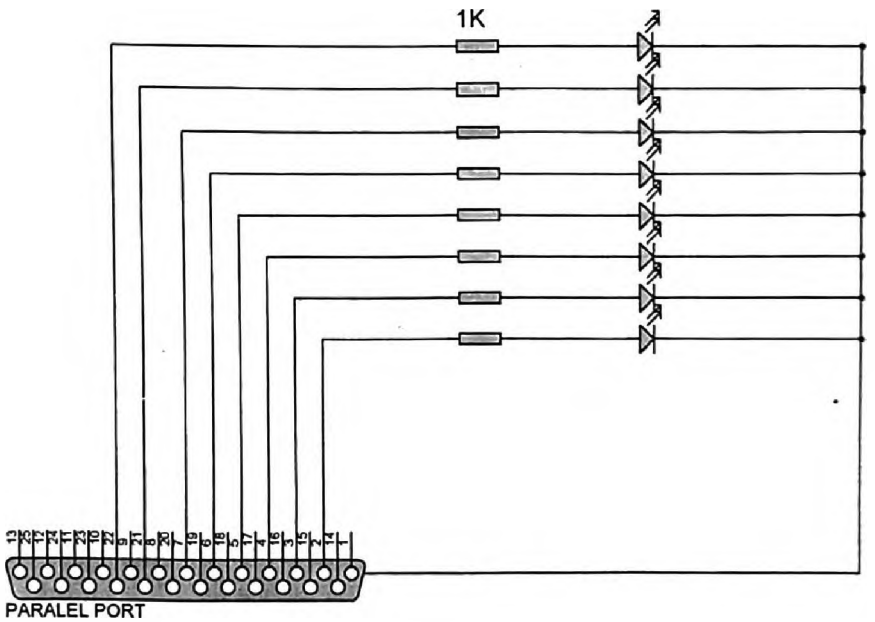
ŞEKİL 1.4: Paralel port ile veri çıkışı (led kontrolü) uygulaması.

Bu komut, `&h378` adresine 25 değerini göndermektedir. `&h378`, paralel portunuzun BASE yani DATA portu adresidir. Bu adres sizin bilgisayarınızda farklı olabilir. Portunuzun adresini nasıl bulacağımızdan az önce bahsetmiştik. Adres değerinin "`&h`" ile başlayarak yazılması kafanızı karıştırmasın. Bu Visual Basic'te Hex sisteminde bir sayı yazmak için sayının başına konulan bir ibaredir. Hex sayı sistemi ile ilgili detaylı bilgiyi bölümün sonunda bulabilirsiniz.

LED Kontrolü Uygulaması

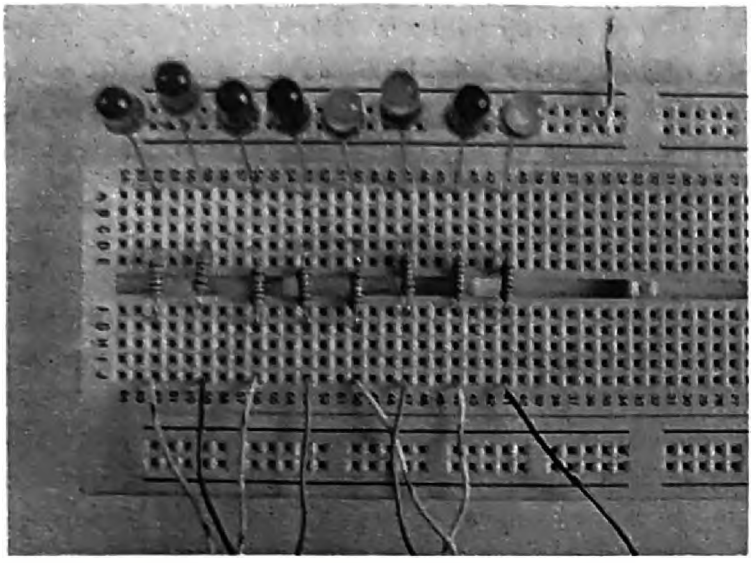
Paralel port ile veri çıkışı uygulamasının ekran görüntüsü Şekil 1.4'teki gibidir. Bu uygulama, paralel veri çıkışını anlatan en iyi örnektir. Uygulamada her buton DATA portunun bir biti ile ilişkilendirilmiştir. Hangi butonu "1" durumuna getirirseniz, o butona bağlı olan led yanacaktır.

Devremiz şu şekilde çalışmaktadır: Paralel port üzerindeki data pinlerinden bir tanesini "1" yaparak, o pinin elektriksel değeri +5 Volt yapılır. Buradan çıkan +5 Voltu direnç üzerinden geçirerek üzerindeki gerilimin değeri ledi bozmayacak değere düşürülür. Bunun için 220-1kOhm arasında bir direnç kullanılabilir. Dirençten geçen akımı da led diyotun (+) ucuna bağlanır. Ledin diğer ucu da TOPRAK kısmına bağlanarak devre kapatılır.



ŞEKİL 1.5: Paralel port ile veri çıkışı (led kontrolü) devre şeması.

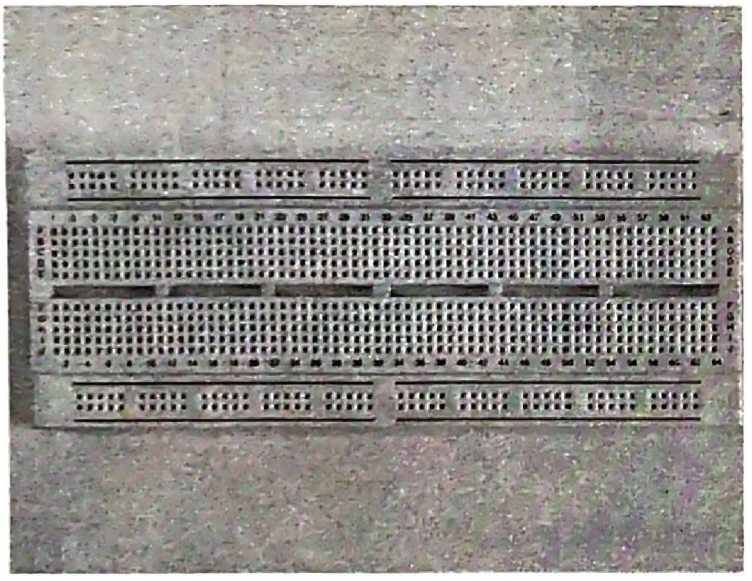
Uygulamalarımızda ilerlediğimiz sürece, ilk kez kullandığımız devre elemanlarını bir kereye mahsus tek tek açıklayacağız.



ŞEKİL 1.6: Devrenin board üzerinde kurulmuş hali.

Board'lar

Uygulamalarımızda devrelerimizi *board* denilen bir delikli plaka üzerinde kuracağız. Şekil 1.7'de görüldüğü gibi, board üzerindeki delikler, satır ve sütunlara ayrılmıştır. Satırlar, A-J arası harfler ile, sütunlar ise, 1-64 arasındaki sayılar ile gösterilmiştir. Her sütun, A-E ve F-J arası harfler ile gösterilen iki bölüme ayrılmıştır ve kendi içlerinde iletken 5 delikten oluşur.

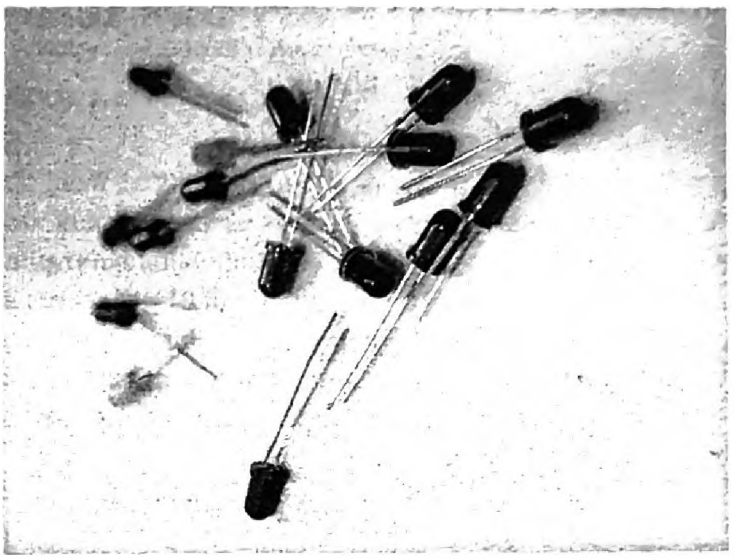


ŞEKİL 1.7: Montaj için delikli montaj plakası (board) kullanılır.

LED Diyotlar

Ledler (Light Emitting Diode) yani ışık yayan diyotlar yapı itibari ile elektrik enerjisini ışığa çeviren kimyasal maddelerden oluşurlar. Kullanılan kimyasal maddenin türüne göre farklı

renkte ışık verirler. Örneğin galyum fosfid kullanılan ledler yeşil ışık verir. Galyum fosfid'e oksijen ve çinko karıştırılarak yapılan ledler kırmızı ışık verir. Bunun yanında Galyum arsenid kullanılarak yapılan ledler ise kızıl ötesi ışık verirler. Ledlerin çalışma voltajları, içinde kullanılan maddeye göre değişiklik gösterir. Kızıl ötesi ışık veren bir ledin çalışma voltajı 1.4V iken yeşil ışık veren ledin çalışma voltajı 2.26V'tur.



ŞEKİL 1.8: LED diyotlar lojik voltajı izlemenin en basit yoludur.

Ledlerin bu çalışma gerilimi aşıldığında içindeki kimyasal maddenin gireceği reaksiyondan dolayı led bozulur. Bu yüzden çalışma gerilimini aşan bir voltaj uygulanacağı zaman uygun bir direnç ile led korunmalıdır. Ledler genellikle endüstriyel ve amatör elektronik alanında ikaz ve görsel efektler amacıyla kullanılır.

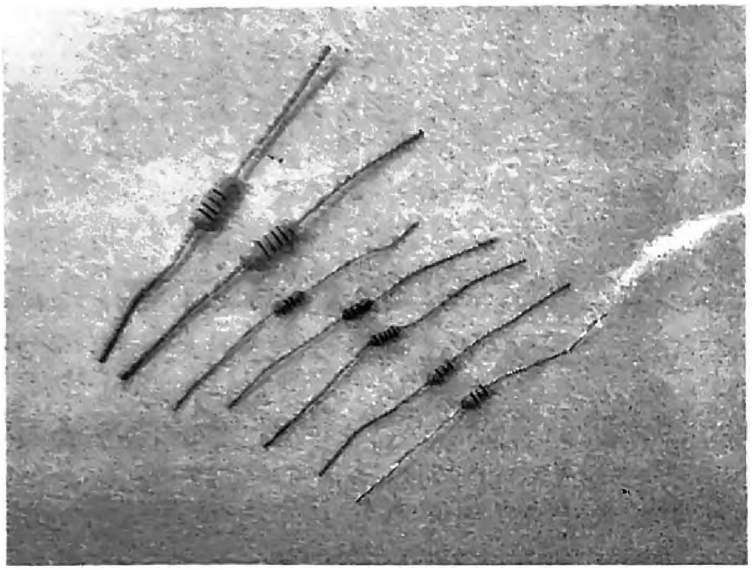
Dirençler

Kitaptaki uygulamaları gerçekleştirebilmeniz için direnç okumayı bilmeniz ya da hesaplamanız gerekmemektedir. Bu işi otomatik olarak yapan program, kitabın kaynak sitesinde bulunmaktadır. Ayrıca uygulamalarda kullanılan dirençlerin değerleri her uygulamada belirtilmiştir. Yine de, bu uygulamada da kullandığımız dirençlerin nasıl çalıştığını bilmekte fayda vardır.

Dirençler elektronik devrelerin vazgeçilmez elemanlarıdır. Yaptıkları iş ise devre içinde kullanılan diğer aktif elemanlara uygun gerilimi temin etmektir. Elektronik devreler sabit bir gerilim ile çalışır, ancak devrede kullanılan elemanların hepsi farklı gerilimlere ihtiyaç duyarlar. İşte bu anda dirençler devreye girer. Tüm devre içinde her eleman için farklı gerilimler kullanmak yerine, uygun dirençler kullanılarak her elemana gereken gerilim dirençler üzerinde düşürülerek temin edilir.

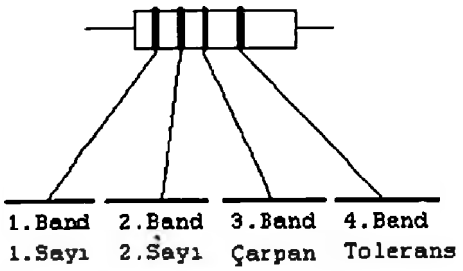
Dirençler, yapımlarında kullanılan malzemeye göre ikiye ayrılır. Karbon dirençler ve telli dirençler. Yine bu iki tip kendi arasında ikiye ayrılır. Bunlar sabit dirençler ve ayarlı dirençlerdir. Bunun yanında bazı özel dirençler de vardır (foto direnç, termistör). Dirençlerin değerleri ohm

ile ölçülür ve sembolü de Ω şeklinde gösterilir. Direncin değeri büyüdükçe kiloohm veya megaohm olarak ölçülmeye başlanır. Bir direncin değerinin ne olduğu, üzerine doğrudan yazılabileceği gibi, en çok kullanılan yöntem olan renk kodları ile üzerine kodlanmasıdır. Doğrudan değeri üzerine yazılmış bir direncin değerini okumak çok kolaydır, ancak renk kodları ile kodlanmış bir direnci okumak için renk kodlarının anlamlarını bilmek gerekir.



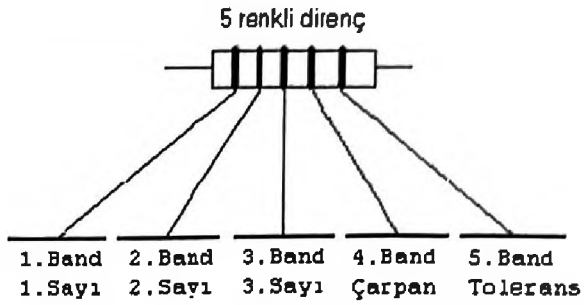
ŞEKİL 1.9: Dirençler

Genellikle dirençler üzerinde 4 veya 5 adet renk bandı bulunur. Bu renkler direnç üzerine kodlanırken, direncin bir tarafına daha yakın olarak yerleştirilir. Değer okuması yaparken de renk bandının kenara en yakın olanından başlamak gerekir. Aşağıdaki şekiller 4 ve 5 renk ile kodlanmış bir direncin görüntüsünü vermektedir.



ŞEKİL 1.10: 4 renkli direnç.

Dikkat ederseniz, renkler sol taraftaki uca daha yakındır ve okumaya bu sol tarafa en yakın renkten başlanmalıdır. 4 renkli dirençlerde 1'nci ve 2'nci bantlar sayıyı, 3'ncü bant çarpanı ve 4'ncü bant ise direncin toleransını verir. 5 renkli dirençlerde ise 1, 2 ve 3'ncü bantlar sayıyı, 4'ncü bant çarpanı ve 5'nci bant ise toleransı verir. 5 bantlı dirençler genellikle daha hassas değerlere sahip dirençlerdir ve özel devreler için imal edilirler.



ŞEKİL 1.11: 5 renkli direnç.

Şekil 1.12'de ise renklerin rakamsal karşılıkları verilmiştir. Dirençlerde tolerans değeri olarak kullanılan renk kodları, o direncin hassasiyetini verir. Örneğin, tolerans değeri olarak gümüş rengi kullanılmış ise, o direnç +/– %10 toleransa sahiptir ve üzerine kodlanan değerin %10 üzeri veya %10 altında olabilir demektir. Tolerans renginden hemen önce gelen renk kodu ise çarpan değerini verir. Bu değer kendisinden önce gelen sayı renk kodlarının çarpılacağı değeri verir. Çarpan renk kodundan önce gelen bütün renk kodları ise sayı değeridir. Bu renklerin karşılık geldiği rakamlar yan yana konur ve çarpan rengi ile çarpılarak direncin değeri bulunur.

RENK	SAYI	ÇARPAN	TOLERANS
Renksiz	--	--	+/- % 20
Gümüş	--	10^{-2}	+/- % 10
Altın	--	10^{-1}	+/- % 5
Siyah	0	10^{0}	--
Kahverengi	1	10^{1}	+/- % 1
Kırmızı	2	10^{2}	+/- % 2
Turuncu	3	10^{3}	--
Sarı	4	10^{4}	--
Yeşil	5	10^{5}	+/- % 0,5
Mavi	6	10^{6}	+/- % 0,25
Mor	7	10^{7}	+/- % 0,1
Gri	8	10^{8}	+/- % 0,05
Beyaz	9	10^{9}	--

ŞEKİL 1.12: Direnç renklerinin rakamsal karşılıkları.

Örnek:

1.Renk = Kahverengi

2.Renk = Siyah

3.Renk = Kırmızı

4.Renk = Altın

Sayı değeri olarak kahverengi 1, Siyah 0 olduğuna göre sayı değeri 10, çarpan rengi olan 3'ncü renk kırmızı olduğuna göre, çarpan değeri 100'dür. Bu durumda direncin değeri 10 x 100 = 1000 Ohm, yani 1 Kohm'dur. Tolerans rengi olan 4'ncü renk de altın rengi olduğuna göre direnç, +/– % 5 toleransa sahiptir. Yani değeri 950 ohm olabileceği gibi 1050 ohm da olabilir.

Dirençler paralel veya seri olarak birbirine bağlandıkları takdirde değerleri değişir. Birbirine seri olarak bağlı bir direnç grubunun değeri tüm dirençlerin değerlerinin toplamına eşittir. Paralel bağlantıda ise işler tamamen değişmektedir. Öncelikle bilinmesi gereken konu, paralel

bağlı dirençler grubunun yeni değerinin gruptaki en küçük dirençten daha küçük olacağıdır. Eğer paralel bağlanacak dirençlerin tümünün değeri aynı ise, oluşacak yeni değeri bulmak için bir direncin değerinin toplam direnç sayısına bölünmesi yeterlidir. Yani 3 adet 10 Kohm'luk direnç paralel bağlandığında ortaya çıkacak olan yeni değer; 10/3=3.3 Kohm'dur.

Farklı değerlerdeki dirençler paralel olarak bağlandığında ise, sonucu bulmak için izlenecek yol şu şekildedir; Öncelikle 1 sayısı tek tek paralel bağlanacak tüm dirençlerin değerlerine bölünür, çıkan değerler toplanır ve bu değer toplam direnç sayısı ile çarpılır. Daha anlaşılır olması için bir örnekle anlatmak gerekirse;

Örnek:

Paralel bağlanacak dirençler = 2.2 Kohm, 10 Kohm ve 100 Kohm olsun. Yapılacak işlemler, sırası ile şöyledir:

1 / 2.2 = 0.45

1 / 10 = 0.1

1 / 100 = 0.01

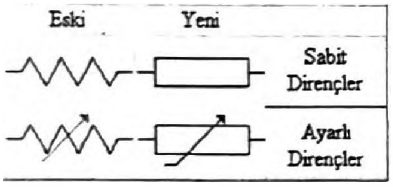
Bu değerleri birbirleriyle toplarsak:

0.45 + 0.1 + 0.01 = 0.56

Şimdi bu değeri toplam direnç sayısı ile çarparsak:

0.56 x 3 = 1.68 Kohm eder.

Şekil 1.13'te sırası ile sabit ve ayarlı bir direncin devre çizimlerinde kullanılan yeni ve eski şekilleri görülmektedir. Eski şekil, devre çizimlerinde artık kullanılmamaktadır.



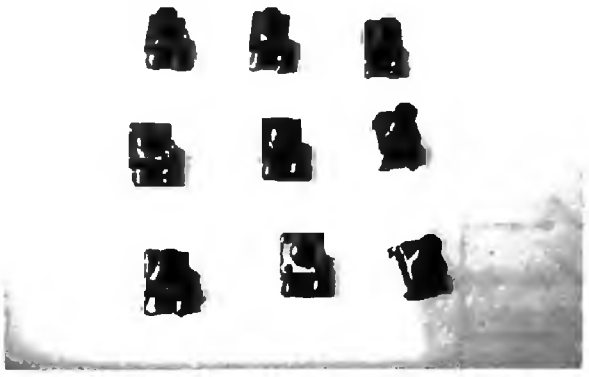
ŞEKİL 1.13: Dirençlerin devre şemalarında gösterilişi.

Paralel Port ile Veri Girişi

Bir önceki uygulamamızda, paralel porttan basit veri çıkışını incelemiş ve porta bağlı 8 tane ledi yakıp söndürmüştük. Şimdiki uygulamamızda, paralel porta bağladığımız 5 tane buton ile bilgisayara sayısal veri girişi yapacağız ve girilen veriyi ekrandan izleyeceğiz.

Tuş Butonlar

Butonlar, devrelerde anahtar görevini gören elemanlardır. Tuş butonlar birçok elektronik cihazın içerisinde kullanılır. Bu tuşlara basılı tutulduğu sürece sol ve sağ bacaklarını iletimde tutarlar. En basitinden, bilgisayarınızın faresinin tuşlarında, buna benzer butonlar kullanılmıştır.



ŞEKİL 1.14: Tuş Butonlar.

Paralel porttan veri girişi STATUS ya da CONTROL portuna ait pinlerden gerçekleştirilebilir. Hatta bazı ayarlamalar yaparak, eğer bilgisayarınız da destekliyorsa, DATA pinlerinden giriş yapmamız da mümkün olabilir. Biz bu uygulamada STATUS portundan veri girişini inceleyeceğiz.

STATUS Portu ile Veri Girişi Uygulaması

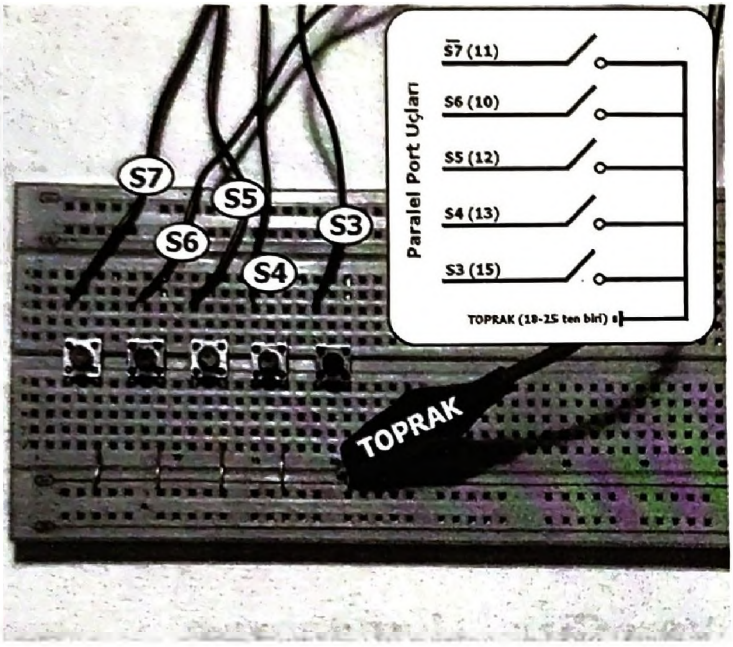
STATUS portu sayesinde, 15-13-12-11-10 numaralı pinlerden 5 bit sayısal giriş yapabiliriz. STATUS portu paralel portunuzun taban adresinin +1 fazlasında bulunmaktadır. Örneğin, paralel portunuzun taban adresi h0378 ise STATUS portu h0379'da bulunacaktır. Veriyi paralel portun şeklinde gözüktüğü gibi S7, S6, S5, S4, S3 pinlerinden yapacağız. Bu 5 pinden herhangi bir müdahale bulunmadan okuyacağınız lojik değer "1" olacaktır. Eğer voltmetrenizle bu pinlerdeki voltajı ölçerseniz +5 Volt olduğunu göreceksiniz. Bu pinlere bağlayacağımız butonlar ile pinleri topraklayarak lojik değerlerini "0" yapıp bir nevi veri girişi gerçekleştirmiş olacağız.

STATUS portundaki veriyi okumak için **INP** komutunu kullanacağız. Kullanım şekli, aşağıdaki gibidir:

```
inp(taban_adres+1)
```

Paralel porta veri giriş işlemi, temel olarak aynı olsa da, veri çıkışından biraz daha karışıktır. Bunun sebebi, STATUS portunda bulunan sinyallerden birinin donanım tarafından terslenmiş olmasıdır. Şimdi, veriyi 8 bit olarak okuduğumuzu varsayalım. Bu durumda STATUS portundaki binary (ikilik sisteme karşılık gelen) değer, "11111111" olacaktır. Bu da 10 tabanında 255'e denk gelmektedir. Fakat bu porttan veriyi okuduğumuzda, bambaşka bir değer olan 127'ye rastlayacağız. Dikkat ederseniz, 127 değerini binary olarak okursak "01111111" değerini elde edeceğiz. İşte burada, 127 değerinin okunmasının sebebi, pinlerden bir tanesi olan S7'nin terslenmiş olmasıdır. Yani, S7'ye karşılık gelen pine veri yolladığımızda o pinin değeri "1" olacaktır. Diğerlerine veri yolladığımızda ise değerleri "0" olacaktır. Veriyi 8 bit olarak okuduğumuzu varsaymıştık. İşte, okuduğumuz değerin son 3 bitini esasında okuyamıyoruz. Yani, 127 değerinin okunabilir karşılığı "01111xxx" olacaktır. Buradaki "x"lerin bir değeri yoktur. İptal edilmiş bitleri

göstermek için kullanılmıştır. Gerçekte okunan değer "01111" olacaktır. Bu da STATUS portuna veri girişinin 5 bit olduğunu bize bir daha gösterecektir.



ŞEKİL 1.15: Devrenin board üzerinde kurulmuş hali.

Daha rahat işlem yapabilmek için terslenmiş olan S7 bitini bizim yazılım olarak tekrar terslememiz gerekmektedir. Yani, porta hiç bir müdahalede bulunmadan okumak istediğimiz değer "11111xxx" olmalıdır. Bu durumda en baştaki biti terslememiz gerekecektir. Bu işlemi yapmanın en kolay yolu `XOR` kullanmaktır.

XOR Kullanımı

`XOR` (Exclusive-Or) operatörü bir byte (8 bit) içerisindeki bir veya daha fazla biti terslemek için kullanılabilecek bir metottur. `XOR` operatörünün BASIC'te kullanımı aşağıdaki gibidir.

```
(okunan_byte) XOR (maske_byte)
```

Burada maske değerinde terslemek istediğimiz bitler için "1", dokunmayacağımız bitler için "0" yazmalıyız. Örneğin "11100111" değerini "11111110" maskesi ile `XOR`'larsak, elde ettiğimiz değer, "00011001" olacaktır. Biz paralel porta göndereceğimiz byte verisini "10000000" değerinin onluk tabanındaki karşılığı olan "128" ile `XOR`'layacağız. Böylece, okuduğumuz verinin en başta bulunan S7'den gelen kısmı "1" ise "0", "0" ise "1" olacaktır. Sonuçta taban adresi `h378` olan paralel porttan veri okumak için yazmamız gereken satır aşağıdaki gibi olacaktır:

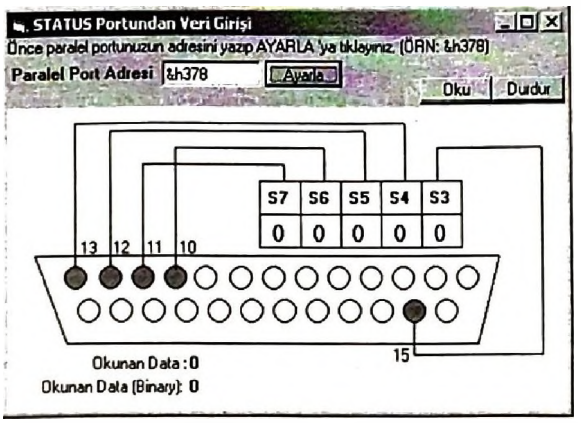
```
Deger = inp(&h379) XOR 128
```

Aşağıdaki tabloda, hangi butonlara bastığımızda yukarıdaki komutla hangi veriyi okuyabileceğimizi gösteriyorum.

TABLO 1.3: STATUS Portundan Okunan Değerler

Butonlar	S7	S6	S5	S4	S3	S2	S1	S0	Değer
S5	1	1	0	1	1	1	1	1	223
S4	1	1	1	0	1	1	1	1	239
S5 ve S4	1	1	0	0	1	1	1	1	207
Tümü	0	0	0	0	0	1	1	1	7

Kullanmadığımız S0, S1 ve S2 değerleri genelde "1" olarak gelmektedir, fakat bazen bu değişebilir. Eğer portunuzdaki bu değerler değişik ise farklı bir sonuç elde edebilirsiniz. Benim denediğim diğer bir bilgisayarda `XOR` kullanmadan ilk okuduğum değer 120 idi. Bu da ikilik sistemde "01111**000**" değerine karşılık geliyordu. Bu durumda, kullanılmayan S2, S1 ve S0 değerlerinin "0" olarak geldiğini gördüm. Bu değeri yukarıda anlattığım şekilde S7'ye ve S3, S2, S1 pinlerine yönelik "1000111" maskesi ile `XOR`'layıp, ilk okunan değeri "11111111" yani 255 olarak alabildim. Buradaki gibi `XOR` kullanmak tamamen size kalmıştır. Yani veriyi `XOR`'lamak zorunda değilsiniz. Bu sadece, hesaplamalarda ve mantıkta biraz kolaylık sağlayacaktır. Önemli olan, okunan verinin siz butona bastıkça değişmesidir.



ŞEKİL 1.16: STATUS Portundan veri giriş yazılımı.

Onaltılık (Hexadecimal) Sayı Sistemi

Bilgisayarların tamamen ikilik sistemde çalıştığını söylemiştik, fakat görüldüğü gibi, ikilik sistemde sayıların ifade edilmesi hem çok uzun, hem de zahmetlidir. Bu yüzden, yazım ve algılama kolaylığı sağlamak için onaltılık ve sekizlik sayı sistemleri de kullanılmaktadır. Onaltılık ve sekizlik sayı sistemlerinde sayılar daha yoğun olarak kodlanıp kullanabilir.

Bölümün başında da söz edildiği gibi, onluk sistemde 10, ikilik sistemde ise 2 sembol bulunmaktadır. Bu durumda onaltılık sayı sisteminde de 16 sembol bulunur. İlk 10 sembol onluk sistemde kullanılan sembollerle tamamen aynıdır: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Daha sonraki semboller aşağıdaki gibidir:

A = 10
B = 11

C = 12
D = 13
E = 14
F = 15

Onaltılık sayı sisteminde yazılmış bir sayıyı onluk sisteme çevirmek için, en sağdan başlayarak basamak değerleri onaltının artan kuvvetleriyle çarpılır:

01AF = (15 * 1) + (10 * 16) + (1 * 256) + (0 * 4096) = 431

Onluk sistemde yazılmış bir sayıyı onaltılık sisteme çevirmek için onluk sistemden ikilik sisteme yapılan dönüşümlerdekine benzer şekilde sayı sürekli 16'ya bölünerek, kalanlar soldan sağa doğru yazılır. Pratikte onaltılık sayı sistemlerinin getirdiği önemli bir yarar vardır. Bu yarar, onaltılık sayı sistemi ile ikilik sayı sistemi arasındaki dönüşümlerin kolay bir şekilde yapılmasıdır. Onaltılık sistemdeki her basamak ikilik sistemdeki 4 bit ile ifade edilebilir.

TABLO 1.4: Onaltılık Sembollerin İkilik Tabanda Karşılıkları

0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

Örneğin, 0x2ADF sayısının ikilik sayı sistemindeki ifadesi aşağıdaki gibidir. 0x2ADF ifadesindeki 0x öneki, sayının onaltılık tabanda yazılmış olduğunu gösterir:

2 = 0010
A = 1010
D = 1101
F = 1111
0x2ADF = 0010 1010 1101 1111

İkilik sistemden onaltılık sisteme yapılacak dönüşümler de benzer şekilde yapılabilir. Önce sayıları sağdan başlayarak dörder dörder ayırırız, en son dörtlük eksik kalırsa sıfır ile tamamlarız. Sonra, her bir dörtlük grup için doğrudan onaltılık sayı sistemindeki karşılığını yazarız.

1010 1110 1011 0001 = 0xAEB1
0010 1101 0011 1110 = 0x2D3E

Bu sayı dönüşümlerini, gerektiğinde Windows altındaki hesap makinesinin bilimsel seçeneğini kullanarak da yapabilirsiniz.

İKİNCİ BÖLÜM

2

PARALEL PORT İLE RÖLE KONTROLÜ

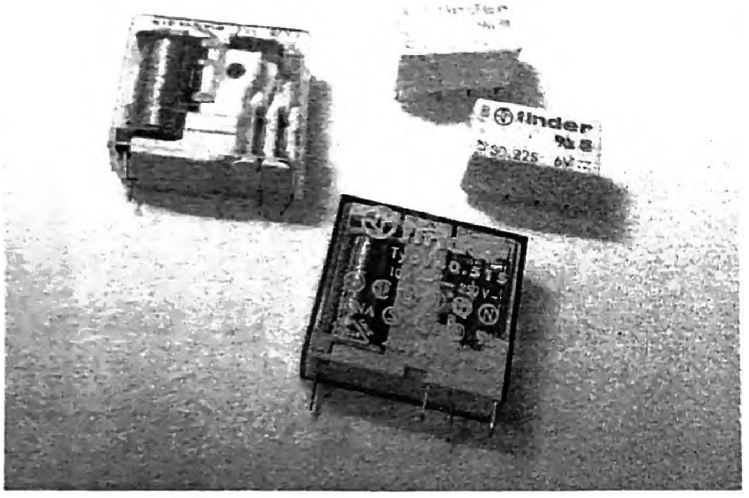
Röle Kontrol Uygulaması

Paralel port kontrolü için yine Visual Basic kullanacağız. Daha öncede bahsettiğimiz gibi **OUT** komutunu kullanarak DATA portuna 0-255 arasında gönderdiğimiz sayısal değer ile istediğimiz data pinlerinin lojik değerlerini "1" yapabiliyorduk. Data pinlerinden bir tanesinin "1" olması o pinin +5 Volt olması anlamına geliyordu. Bu uygulamamızda yine DATA portunu kullanacağız ve elde ettiğimiz bu lojik voltaj ile elektronik cihazı çalıştırabilecek bir devreyi tetikleyeceğiz. Önce, ilk kez kullanacağımız devre elemanlarını kısaca tanıyalım.

Röleler

Röle düşük akımlar ile çalışan elektromanyetik bir anahtardır. Üzerinde bulunan elektromanyetik bobine rölenin türüne uygun olarak bir gerilim uygulandığında bobin mıknatıs özelliği kazanır ve karşısında duran metal bir paleti kendine doğru çekerek bir veya daha fazla kontağı birbirine irtibatlar ve bir anahtar görevi yapar.

Tristör ve triyakların imal edilmesinden sonra popülerliğini kaybeden röleler yinede birçok alanda hala kullanılmaktadır. Tristör ve triyaklara göre tek avantajı, tek bir bünye içinde birden fazla anahtar kontağına sahip olabildiği için birden fazla yükü aynı anda açabilmesi veya kapatabilmesi ve hatta aynı anda bazı yükleri açıp bazılarını kapatabilmesidir. Bu işlem tamamen rölenin kontaklarının dizaynı ile ilişkilidir.

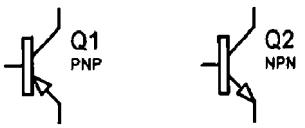


ŞEKİL 2.1: Röleler.

Dezavantajları ise daha fazladır. Öncelikle, mekanik olarak çalıştığı için, daha çabuk arıza yapar. Kontaklar sürekli birbirine kapanıp açıldığı için, oluşan elektrik atlamaları zamanla kontakların oksitlenmesine ve iletimini kaybetmesine neden olur. İletime geçme süresi tristör ve triyaklara oranla daha uzundur. Ayrıca kontakların çekilip bırakılması sırasında çıkardıkları sesler pek hoş değildir.

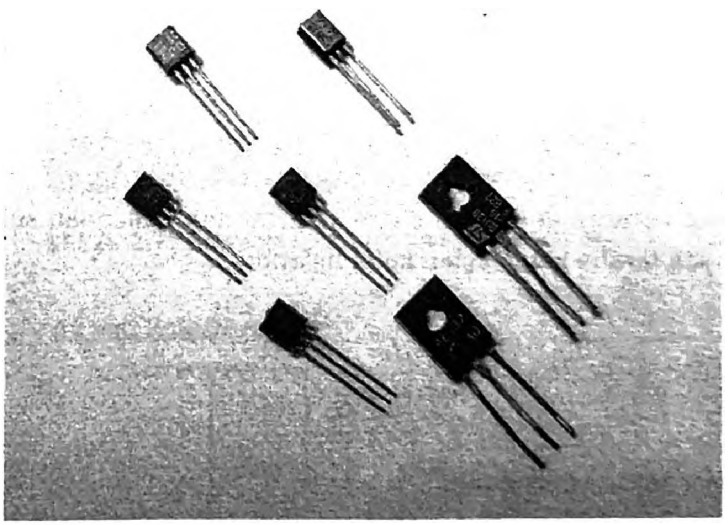
Transistörler

Transistörler elektroniğin ve bilgisayarın en temel elemanlarıdır. İnsanların yapı taşları nasıl hücreler ise, transistörler de işlemcilerin yapı taşlarıdır. Tam çalışma prensipleri biraz karışıktır. Transistörler PNP ve NPN olmak üzere iki çeşittir. Emiter, kollektör ve beyz olmak üzere 3 adet bağlantı ucu vardır. En çok kullanılan yarı iletken devre elemanıdır ve devrelerde bir anahtar görevi görür. PNP transistörlerde kollektör eksi (–), emiter artı (+) ve beyz ucu da emitere göre eksi (–), kollektöre göre artı (+) polarize alır. NPN tipinde ise durum tam tersidir. Ayrıca transistörler birbirlerini tetikleyecek şekilde bağlandığında akım kazancı elde edebilen devre elemanlarıdır.



ŞEKİL 2.2: PNP ve NPN tipi transistörlerin devre şemalarında kullanılan sembolleri.

Paralel portun herhangi bir DATA pininden elde edebileceğimiz+5 voltluk voltaj, yine 5 voltluk bir röleyi bile tetikleyemeyecek kadar düşük bir akıma sahiptir. Bu nedenle röleyi dışarıdan ayrıca besleyeceğiz.



ŞEKİL 2.3: Transistörler

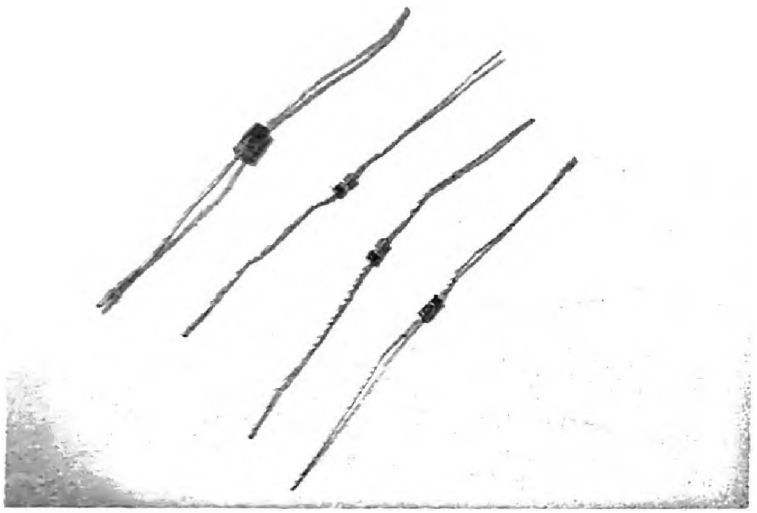
Diyotlar

Diyotlar bir yönde akım geçiren, diğer yönde akımı geçirmeyen devre elemanlarıdır. Anot ve katot uçlarına sahiptir. Her zaman anoduna artı (+), katoduna eksi (–) gerilim verildiğinde iletime geçer, tersi durumda yalıtkandır. Bu özelliğinden dolayı AC gerilimi DC gerilime çevirmek için kondansatörler ile beraber kullanılan bir elemandır.

Diyotların yapı malzemeleri germanyum veya silisyumdur. Silisyum diyotlar doğru yönde bağlandığı taktirde 0.6 volt civarında iletime geçerler, geriye kalan gerilimi doğrudan üzerinden geçirir. Germanyum diyotlar ise 0.3 volt civarında iletime geçer. AC gerilimi DC gerilime çevirmek için silisyum diyot kullanılır.

Ayrıca voltaj ayarlamalarında kullanılan zener diyotlar vardır ki, bu diyotlar sabit bir voltaj değerinden fazlasını üzerinden geçirmez. Bu şekilde sabit ve kararlı bir gerilime ihtiyaç duyan cihazlara gerekli gerilimi vermek amacıyla zener diyotlar ile regüle devreleri yapılır.

Diyotların değerleri doğrudan üzerlerine yazılır. Zener diyotlarda sabit voltaj değeri yazılırken, diğer diyotlarda diyotun modeli yazılır. Mesela bir zener diyot üzerinde 2v7 yazıyorsa bu 2.7V bir zener diyot anlamına gelir. Diğer diyotlarda ise 1N4001, 1N4148 gibi diyotun modeli yazılır. Genellikle katot ucuna yakın tarafa bir çizgi konularak anot katot uçlarının kolay bulunması sağlanır.



ŞEKİL 2.4: Diyotlar

Şekil 2.5'te sırası ile zener diyot ve normal diyotun devre şemalarında kullanılan sembolü görülmektedir. Her iki diyotun da sağ tarafta kalan uçları katot uçlarıdır.



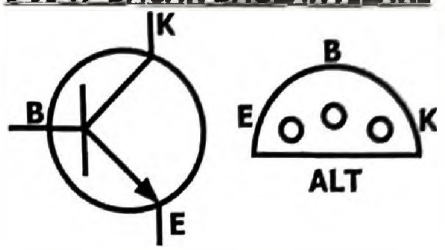
ŞEKİL 2.5: Diyotların devre şemalarında kullanılan sembolleri.

Röle Kontrol Devresi

Devremiz için aşağıdaki malzemelere ihtiyacımız olacaktır.

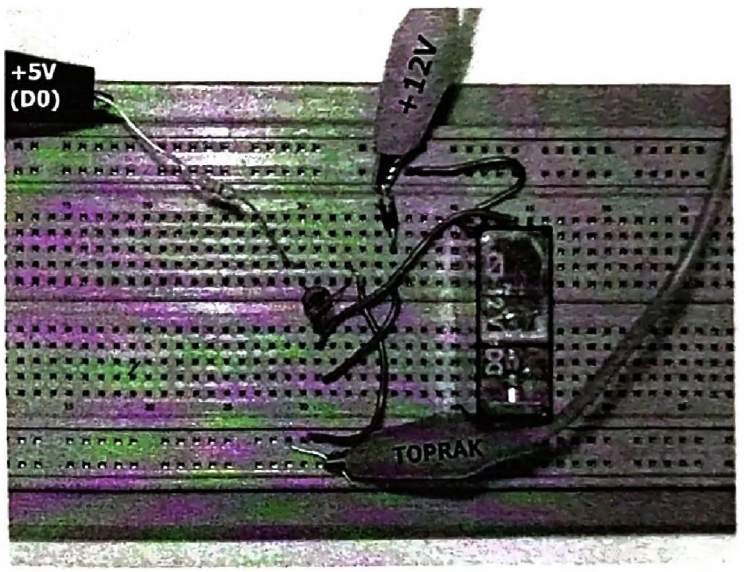
- 1 adet 4.7K Direnç (R1)
- 1 adet BC547 Transistör
- 1 adet 1N4007 Diyot (D1)
- 1 adet 12Volt 10A-250v Röle

BC547 BACAK BAĞLANTILARI



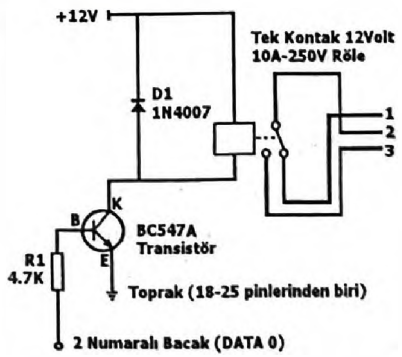
ŞEKİL 2.6: BC547 Bacak bağlantıları.

Devrede kullanacağımız NPN tipi Bi-Polar transistördür. Uzun süre transistörler ile çalışmadıysanız, hangi bacağının beyz, hangisinin emitör olduğunu bulmak kolay olmayacaktır. Bunun için size BC547 NPN tip transistörünün bağlantı şeklini ve alttan baktığınızda hangi bacağının hangi uç olduğunu şekilde gösteriyorum.



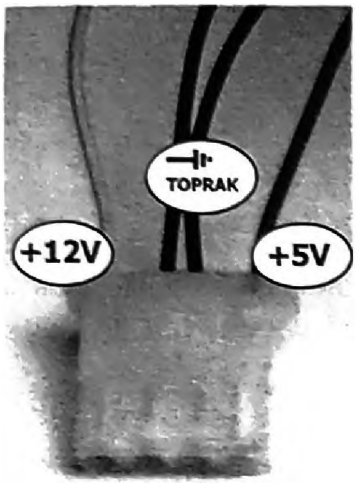
ŞEKİL 2.7: Devrenin board üzerinde kurulmuş hali.

12 Volt Röle Devresi



ŞEKİL 2.8: Röle kontrolü devre şeması.

Röleyi beslemek için gereken 12 Voltluk gerilimi, şekildeki hard disk güç kablosundan elde edebilirsiniz. Sarı renkli kablo üzerinde +12 Volt gerilim vardır.



ŞEKİL 2.9: Hard Disk güç kablosu.

Röle Kontrol Yazılımı

Elimizin altında, istediğimiz DATA pininin lojik değerini değiştirebilen bir program zaten olduğundan, bu uygulama için ikinci bir program yazmaya gerek yoktur. Paralel Port ile veri çıkışı uygulamasında, led kontrol etmek için kullandığımız Visual Basic programını bu uygulamada da kullanabiliriz. Yine de, röle kontrolü için yeniden bir program yazmak isteyenler için kullandığımız komutlardan kısaca bahsedelim.

DATA portunun 0'ncı pininin lojik değerini "1" yapmak için kullanmamız gereken `OUT` komutu `OUT &h378,1` şeklinde olmalıdır. Buradaki `&h378` parametresi paralel portumuzun BASE adresini, ikinci kullandığımız parametre olan `1` ise paralel porta "00000001" değerini gönderir. Böylece 8 tane DATA pininden ilkinin (DATA0) lojik değeri "1" olur. Örneğin DATA0 yerine DATA3'ü kullanmak isteseydik, porta "00001000" değerine 10'luk sayı sisteminde karşılık gelen "8" değerini göndermemiz gerekirdi. Bu komut sayesinde DATA pininin lojik değeri "1" olduğu sürece röle aktif halde kalacaktır. Röleyi tekrar pasif hale getirmek için röleyi bağladığımız data pinin lojik değerini "0" yapmalıyız. Bunun için, `OUT &h378,0` komutunu kullanabiliriz.

Röle Kontrol Devresi Nasıl Çalışıyor?

Röleyi kontrol etmek için, veri girişi uygulamamızdaki led kontrol programını, devreyi tetiklemek için ise herhangi bir DATA pinini kullanıyoruz. Devreye enerji verildiğinde, +12 Voltluk gerilim rölenin ayaklarından birine kadar geliyor, fakat transistör iletimde olmadığı için rölenin diğer bacağından toprağa ulaşamıyor. Bu nedenle röle aktif hale geçemiyor. Program aracılığı ile devreye bağlı olan DATA pinini "1" yaptığımızda gelen +5 voltluk lojik voltaj transistörü tetikliyor ve transistördeki kollektör-emitör (CE) iletime geçiyor. Dolayısı ile rölenin diğer ba-

cağı toprağa ulaşabiliyor ve röle kapanıyor. Bu durumda röleye bağlı olan elektronik cihaz çalışıyor.

Neden 12 Volt?

Akla gelen ilk soru, neden elimizde +5 voltluk hazır bir gerilim varken, neden +5 volt ile çalışan bir röle kullanmadığımız olacaktır. Paralel portun data pinlerinden elde ettiğimiz +5 Voltluk lojik akımın değeri yaklaşık 50mA (miliamper) yani 0.05 A'dır (Amper). Bu akım, 5 voltluk bir röleyi açabilmek için çok düşüktür. Bu akımı yine transistörler ile yükseltmenin yolları vardır, fakat paralel porttan 50mA'nın üzerinde akım çekmek sağlıklı bir iş değildir.

ÜÇÜNCÜ BÖLÜM

3

ADIM (STEP) MOTOR KONTROLÜ

Adım Motorlar

Adım motorların endüstriyel ve elektronik uygulamalarda kullanımı oldukça fazladır. En basitinden, bilgisayarımızdaki floppy disket sürücüsünde ve hard disklerde bu teknolojiye başvurulmuştur. Adım motorlar, girişlerine uygulanan lojik sinyalleri dönme hareketine çevirirler. İstediğiniz yönde ve derecede döndürebileceğiniz adım motorlar, hassas hareketleri sayesinde, birçok cihazda konum kontrolü amacıyla kullanılmaktadır. Dışarıdan bakınca çok komplike bir iş gibi gözükse de, PC ile adım motor kontrolü oldukça kolay ve eğlencelidir.

Adım motorun kablolarından bir veya iki tanesi ortaktır (vMotor). Yaptığımız işlem basit olarak bu ortak kabloya sürekli +12 Volt göndermek ve diğer uçları ise belli bir sırada toprağa göndererek bir adım hareketi elde etmek. Bunu iki farklı şekilde yapabiliriz. Bunlardan bir tanesi Bölüm 2'de kullandığımız 12 Voltluk röle devresine benzer bir devrenin 4 transistörlüsü ile bu sinyalleri göndermek. Bir diğer yolu ise, içerisinde bu transistörleri halihazırda bulunduran bir entegre kullanmak. Burada, her iki yöntemi de göreceğiz. Yine de benim kolaylık açısından tavsiyem, entegreli devreyi uygulamanız.

Adım Motor Kontrolü Uygulaması

Adım Motor Entegreli Sürücü Devresi

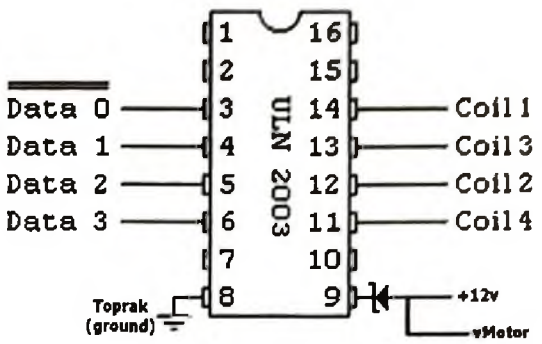
Devremiz için aşağıdaki malzemelere ihtiyacımız vardır:

- 1 adet ULN2003 Entegre
- 1 adet IN4001 Diyot
- 1 adet 12 Volt 5 Kablolu Adım Motor

Kullanacağımız adım motoru, bulabiliyorsanız, eski 5 ¼ disket sürücülerinden kolayca sökebilirsiniz. Eğer bulamıyorsanız adım motor için sanırım biraz elektronikçi dolaşmanız gerekecektir. Bulacağınız adım motoru 4, 5 ya da 6 kablolu olabilir. Biz uygulamamızda 5 kablolu adım motoru kullanacağız. Çalışma prensipleri aynı olduğundan kablo sayısının değişmesi, yazacağımız programın mantığının değişmesi anlamına gelmeyecektir.

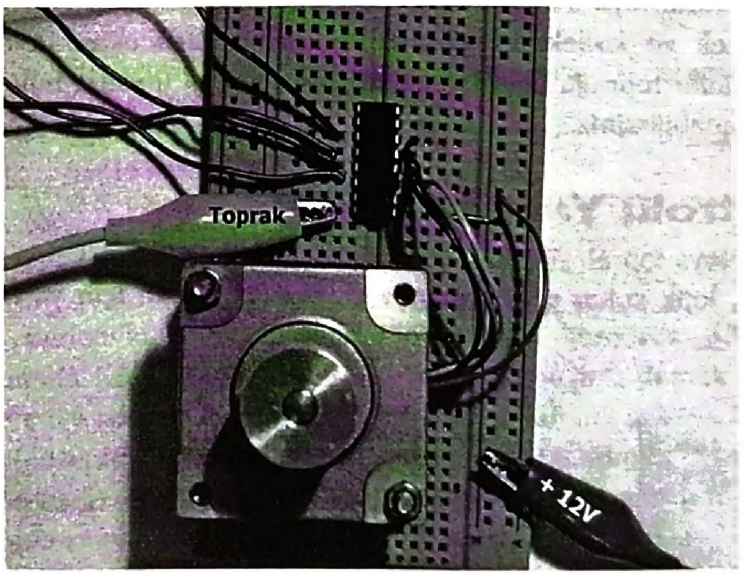
Adım Motor Kontrolü Devresi Nasıl Çalışıyor?

Adım motor sürücüsü olarak ULN2003 entegresini kullanıyoruz. Bu entegreyi bundan sonraki birçok uygulamamızda bol bol kullanacağız. Sürücü devresi olarak kullanılan ULN2003 İçerisinde 7 adet NPN transistör ve dahili diyot barındırmaktadır. Böylelikle bizi transistör bacaklarıyla uğraşmaktan kurtarmaktadır. Kullanımı ise oldukça kolaydır. Devre şemasından da anlaşılabileceği gibi, entegrenin 9 numaralı bacağına +12 Volt ve 8 numaralı bacağına da Toprak (ground) uyguluyoruz. Daha sonra 3, 4, 5 ve 6 numaralı bacaklara da paralel portun DATA pinlerinden gelen +5 Voltluk değerleri uygulayacağız. Bu sayede, örneğin 3 numaralı bacağa +5 Volt (lojik voltaj) uyguladığımızda 14 numaralı bacak toprak olacaktır. Aynı şekilde sırayla 4 için 13, 5 için 12, 6 için ise 11 numaralı bacaklar toprak olacaktır.



ŞEKİL 3.1: ULN2003 ile adım motor ve paralel port bağlantısı.

5 kablolu adım motorunun kablolarından bir tanesi vMotor dediğimiz ortak kablodur. Önemli olan bu kablonun hangisi olduğunu bulmaktır. Bunun için voltmetrenizi OHM ölçere getirin ve kabloların uçlarını ikişer ikişer ölçün. Tüm uçlar ile arasındaki direnç aynı olan kablo ortak vMotor kablosudur. Bunun için biraz deneme yapmanız gerekebilir. Şekilde gözüktüğü gibi, diğer 4 kablo motor kömürlerine (coil) gitmektedir. Bu 4 kablonun da bir sırası vardır. Bu sırayı da deneme yanılma yöntemiyle bulmak mümkün olacaktır. Eğer bu kabloları yanlış sırada bağlarsanız, motor dönmek yerine sadece titreme yapacaktır. Yukarıda da bahsedildiği gibi motora adım attırmak için yapmamız gereken, vMotor kablosuna +12 Volt verirken, diğer kömürlere bağlı kablolara belli bir sıra ile toprak göndermektir.

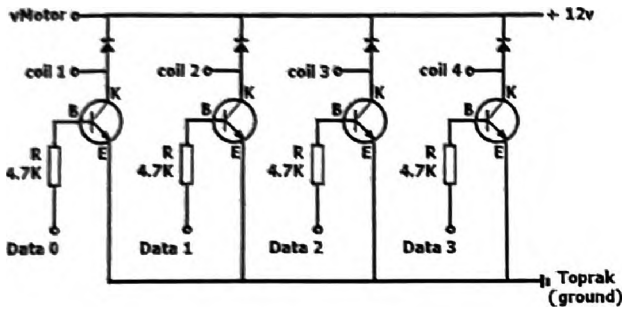


ŞEKİL 3.2: Devrenin board üzerinde kurulmuş hali.

Transistörlü Adım Motor Sürücü Devresi

Bu devrenin tek farkı, bir önceki devrede entegrenin yaptığı işlerin transistörlere yaptırılmasıdır.

- 4 adet BC547B transistör
- 4 adet 1N4001 diyot
- 1 adet 12 Volt 5 kablolu adım motor



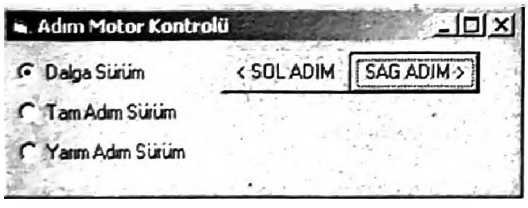
ŞEKİL 3.3: Transistörlü adım motor sürücü devresi.

Her iki devrede de kullandığımız diyotun değerinin herhangi bir önemi yoktur. Motorda adım attıktan sonra oluşacak ters EMK bu diyot sayesinde absorbe edilebilir. Yine de benim tavsiyem, entegreli sürücü devresini kullanmanızdır. Bu entegrenin nasıl çalıştığını iyi anlarsanız, birçok uygulamada transistörler ile uğraşmak zorunda kalmadan 12 Voltu kontrol edebilirsiniz.

Transistörlü sürücü devresi, Bölüm 2'deki Röle kontrol devresine benzemektedir. Bu sefer +12 Voltluk akım vMotor kablosuna bağlıdır. Motorun diğer kablolarını sıra ile toprağa göndermemiz gerekmektedir. Transistörün Beyz ucuna uygulayacağımız data pininden gelen +5Volt sayesinde transistör tetiklenecek ve kollektör-emitör (CE) iletime geçecektir. Dolayısı ile transistörün C ucuna bağlı kablolar toprağa ulaşabilecektir. Transistörün uçlarını bulmak için Bölüm 2'deki şekillerden yararlanabilirsiniz.

Adım Motor Kontrolü Yazılımı

Yukarıda bahsettiğimiz bu 4 kabloya toprak sinyalini göndermek için entegrenin 3, 4, 5 ve 6'ncı bacaklarına sıra ile +5 Voltluk lojik voltaj göndermemiz gerekiyor. Bunun için yine paralel portun DATA pinlerini kullanıyoruz.



ŞEKİL 3.4: Adım motor kontrolü yazılımı.

Programı Visual Basic altında geliştiriyoruz. Adım motora sinyalleri göndermek için D0, D1, D2, D3 ile gösterilen ilk dört DATA pinini kullanıyoruz. DATA pinlerinden çıkış almak için ise yine `OUT` komutunu kullanıyoruz. Fakat, sinyalleri peşi sıra göndermek için `OUT` komutunu alt

alta yazmamız işe yaramayacaktır. Çünkü veriyi DATA pinlerine ufak da olsa belli zaman aralıkları ile göndermemiz gerekiyor. Bunun için her **OUT** komutunun arasına bekletmek için çok eski yöntemlerden biri olan **FOR-NEXT** döngüsünü yazıyoruz. Böylece her komuttan sonra, program bir sayma işlemine giriyor ve sayma işlemi bitene kadar bize gecikme sağlamış oluyor. Dikkat etmeniz gereken nokta, **FOR-NEXT** döngüsünü bitirdiğiniz sayıdır. Ben burada "2000" kullandım. Bu sayı bir Pentium PC için yeterli beklemeyi sağlıyor. Siz de sisteminizin hızına göre buradaki sayıyı değiştirmelisiniz. Örneğin eski bir 486 PC için "500" yeterli olabilir. Aşağıdaki Visual Basic program örneği D0, D1, D2 ve D3 pinlerini sıra ile +5 Volt yapacak ve entegrenin bu pinlerine bağlı olan bacaklarının karşılıklarını toprağa çekecektir. Neticede motorumuz bir adım atmış olacaktır.

ADIM MOTOR KONTROLÜ (Dalga Sürüm) – Visual Basic kodu

```
Sub bekle(sayi as long)
   'Bekletme alt programı
   Dim i as long
      For i=1 to sayi:next i
End sub

Sub Command1_Click()
  Bu örnek program, dalga sürümü sinyallerini, motora göndermektedir.
   Dim adres as integer
   Adres=&h378
   OUT adres, 0 'Tüm data pinlerinin değerlerini "0" olacaktır
   Bekle 2000 'Tanımladığımız BEKLE fonksiyonu ile bir sürelik gecikme sağlıyoruz.
   OUT adres,1 'D0 pininin lojik değeri "1" olacak ve +5 Volt yüklenecek.
   Bekle 2000
   OUT adres,2 'D1 pininin lojik değeri "1" olacak ve +5 Volt yüklenecek.
   Bekle 2000
   OUT adres,4 'D2 pininin lojik değeri "1" olacak ve +5 Volt yüklenecek.
   Bekle 2000
   OUT adres,8 'D3 pininin lojik değeri "1" olacak ve +5 Volt yüklenecek.
   Bekle 2000
End Sub
```

Benim kullandığım adım motorlar 1.8 derecelikti. Bu, motora attıracağınız her normal adımda 1.8 derecelik bir dönme elde edeceğiniz demek oluyor. Bu da motorun bir tur atması için 200 normal adım atması gerektiği anlamına geliyor. Motorun vMotor dışında kalan diğer 4 kablosuna göndereceğiniz sinyallere göre bu adımın yönünü ve açısını değiştirmeniz mümkün olacaktır. En basitinden, motora ters adım attırmak için, sinyalleri D3'ten D0'a doğru göndermeniz yeterli olacaktır. Çok hassas çalışmadığımızı, motorumuzun 2 derecelik olduğunu ve 45 derecelik bir dönme gerçekleştirmek istediğimizi düşünelim. Bunun için yukarıda anlattığım normal adım sinyalleri yeterli olmayacaktır. Bu durumda motoru 1'er derecelik açılarla döndürebilmemiz gerekmektedir. Yarım adım attırma metodu ile bu işi kolayca yapmamız mümkündür. Bir diğer metot ise dalga sürümü sinyalleridir. Hassas hareketler üzerinde çalışmayacaksanız, dalga sinyallerini kullanabilirsiniz. Aşağıdaki tablolarda tam adım, yarım adım ve dalga sürümü için uçlara göndermeniz gereken sinyal çeşitlerini ve **OUT** komutu ile

göndermeniz gereken değerleri görebilirsiniz. Programda gönderdiğimiz değerlerin ikilik sistemdeki karşılıklarının, D3-D0 sütunlarını soldan sağa doğru okuduğumuzdaki değerlerine eşit olduğunu aşağıdaki tablolarda görebilirsiniz. Eğer ters yönde dönüş elde etmek istiyorsanız, sinyalleri ters yönde (tablodaki satırları aşağıdan yukarıya doğru okuyarak) gönderebilirsiniz.

TABLO 3.1: Yarım Adım Metodu

Değer	D3	D2	D1	D0
9	1	0	0	1
1	0	0	0	1
2	0	0	1	0
6	0	1	1	0
4	0	1	0	0
2	0	0	1	0
12	1	1	0	0
8	1	0	0	0

TABLO 3.2: Tam Adım Metodu

Değer	D3	D2	D1	D0
9	1	0	0	1
3	0	0	1	1
6	0	1	1	0
12	1	1	0	0

TABLO 3.3: Dalga Sürümü Adım Metodu

Değer	D3	D2	D1	D0
1	0	0	0	1
2	0	0	1	0
4	0	1	0	0
8	1	0	0	0

DÖRDÜNCÜ BÖLÜM

4

LCD PANEL KONTROLÜ

LCD Paneller

Farkına varmasak bile LCD paneller günlük hayatımızın bir parçası gibidir. En basitinden, bazı cep telefonlarında, hesap makinelerinde ve birçok elektronik eşyada bu teknoloji kullanılmaktadır. LCD paneller hafiftir ve az yer kaplarlar. Bazen basit bir veri çıkışı almak için büyük bir monitöre ihtiyacınız olmayabilir. Yaptığınız devrelere küçük bir LCD panel bağlayarak, devreden kullanıcıya göndermek istediğiniz mesajları bu ekrana yazdırabilirsiniz.

LCD Panel Kontrol Uygulaması

Önceki uygulamalarımızda, paralel portun DATA ve STATUS portlarını kullanmıştık. Şimdi, daha önce hiç kullanmadığımız CONTROL portunu da kullanacağız. CONTROL portunun adresi, paralel portunuzun taban adresinin iki fazlasıdır. Yani paralel portun taban adresi h0378 ise, CONTROL portunun adresi h0378 +2 yani h037A olur.

Paralel port üzerinde CONTROL portuna ait 4 tane pin vardır. Bu pinlerden C0, C1, C3 pinleri terslenmiştir. Yani terslenmiş olan bu pinlere veri göndermediğimiz zaman durumu "1"dir. Bu pinlerden bir tanesine veri gönderdiğimizde, o pinin durumu "0" olacaktır. Diğer bir deyişle, CONTROL pinlerinin düşük olan 4 bitine "1111" yazmanız halinde port üzerindeki bitler "0100" olacaktır.

Standart LCD paneller 8 bit programlanabilir. Yani paralel portun 8 tane pini ile istediğimiz veriyi LCD panele gönderebiliriz. Bir LCD panel gelen verileri iki şekilde değerlendirebilir. Gönderilen sinyallerin LCD tarafından komut olarak algılanmasını istiyorsak, paneli veri kontrolü için ayarlarız. Eğer ekrana yazı göndermek istiyorsak, bu sefer paneli veri saklayıcısı için programlamamız gerekir.

Standart LCD paneller, +5 Volt ile çalışır. Bu voltajı bilgisayarınızın içerisindeki hard disk kablosundan almanız mümkündür. Şekildeki kırmızı olan kablo üzerinde +5 Volt gerilim vardır. Benim kullandığım LCD panelin modeli (2 X 16) SAMSUNG-1602D idi. Eğer bunu bulamıyorsanız, kitaplarda sık rastlanan HITACHI HD44780 model LCD panel de işinizi görebilir. Genel olarak 2X16'lık ve 14 ucu bulunan LCD panellerin bacak bağlantıları aynı sayılır. Önemli olan LCD panelin hangi pininin hangi işlev için kullanıldığını bilmektir. Standart bir LCD panelin üzerinde 14 tane uç bulunur. Bu pinlerin görevleri aşağıdaki gibidir.

LCD panel veri kontrolüne ayarlandığı zaman, 8 tane DATA pini ile panele değişik komutlar gönderebiliriz. Örneğin ekranı silme, imleci açıp kapatma gibi. Veri saklayıcısına ayarladığımız zaman ise yine aynı 8 tane DATA pini ile ekrana istediğimiz karakteri yazdırabiliriz. İşte bu veri kontrolü ya da saklayıcısını belirlememiz için bize bu 8 DATA pininin dışında 2 tane daha pin gerekir. Bu pinleri de istediğimiz gibi lojik değerlerini "1" ya da "0" yapabilmeliyiz. İşte bu seçim işlemi için CONTROL portunun C0 ve C3 pinlerini kullanacağız.

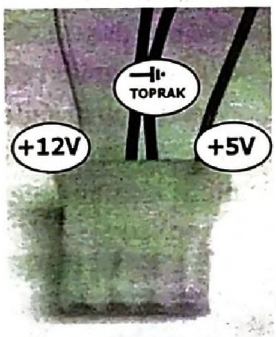
TABLO 4.1: LCD Pinlerinin İşlevleri

LCD Pinleri	Pin İşlevi
1	Toprak (Ground)
2	VCC +5 Volt

TABLO 4.1: LCD Pinlerinin İşlevleri

LCD Pinleri	Pin İşlevi
3	VO (Kontrast)
4	RS (Register Select)
5	R/W (Okuma ya da Yazma Modu)
6	E (Enable)
7-14	D0-D7 (DATA girişleri)
15*	LCD Panel ışığı (+5 Volt)
16*	Toprak (Ground)

* 15 ve 16 numaralı uçlar bazı LCD panellerde bulunmayabilir. Eğer LCD panel ışıklı ise iç ampulü bu pinlerden yakmak mümkündür.



ŞEKİL 4.1: LCD paneli PC güç kablosu üzerindeki +5V ile besliyoruz.

RS (register select): RS girişi "0" (düşük) olduğu zaman LCD panelin komut saklayıcısı, yüksek olduğu zaman ise veri saklayıcısı seçilmiş olur. Biz LCD panelin RS girişini paralel portumuzun 17 numaralı pinine (select) bağlıyoruz. Bu da şekildeki C3 CONTROL portuna karşılık gelmektedir.

R/W (read/write): LCD panelden okuma yapmayı düşünmediğimizden, 5 numaralı girişi (R/W) doğrudan topraklıyoruz. Bu şekilde bu girişi "0" (düşük) yapmış oluyoruz. Yani paneli YAZMA (write) moduna almış oluyoruz. Fakat bu durumda LCD ye yazma işleminin bitip bitmediğini belirten meşgul bayrağını (busy flag) okuyamayız. Bu sebeple **FOR-NEXT** döngüleri ile gecikmeler sağlamamız gerekmektedir.

E (enable): Paralel portun veri yoluna (data pinlerine) 8 bitlik veriyi yerleştirdikten sonra LCD panele E ucu ile buradaki veriyi okumasını sağlayabiliriz. Bunun için E ucuna bağlı olan paralel portumuzun 1 numaralı pinini (strobe) "0" (düşük) yapmalıyız. Daha sonra biraz beklemeli ve bir sonraki byte için tekrar "1" (yüksek) durumuna getirmeliyiz.

VO (kontrast): 3 numaralı VO kontrast girişini doğrudan 10K'lık bir direnç üzerinden toprağa gönderiyoruz. Buradaki direncin değerini artırarak kontrastı düşürebilir, azaltarak ise kontrastı yükseltebilirsiniz. Eğer bu kontrast ile sürekli oynamak istiyorsanız buraya 10K'lık bir ayarlı direnç bağlayabilirsiniz.

DATA: 7-14 numaralı pinleri doğrudan paralel portumuzun DATA pinlerine bağlıyoruz. Bu sayede istediğimiz komut ya da veriyi LCD panele göndermemiz mümkün olacaktır. Aşağıdaki TABLO 4.4'te standart LCD panelin komut saklayıcısının fonksiyonlarını yazıyorum. Veri saklayıcısına ise göndereceğimiz veriler, yazdırmak istediğimiz karakterlerin ASCII kodlarıdır. Örneğin panele "A" harfi yazdırmak istediğimiz zaman bu harfin bilgisayarınızdaki ASCII koduna karşılık gelen 65 değerini DATA pinine göndermeliyiz. ALT tuşunu basılı tutup nümerik klavyenizi kullanarak 65 yazmayı deneyin. Bu şekilde ASCII kodlarına karşılık gelen harfleri ya da karakterleri bulmanız mümkündür.

LCD Panel Kontrol Devresi Nasıl Çalışıyor?

Programımızı yazmaya başlamadan önce CONTROL portunun pinlerindeki lojik voltajı istediğimiz gibi kontrol edebildiğimizden emin olmalıyız. Yukarıda da belirttiğim gibi biz CONTROL portunun sadece C3 ve C0 pinlerini kullanıyoruz. 4 adet pin olduğundan CONTROL portuna göndereceğimiz veri 4 bit olacaktır.

Bunun anlamı, göndereceğimiz verinin ikilik sistemdeki son 4 hanesi dışındakiler ile bir işimiz olmayacağıdır. Örneğin C3 pininin voltajı ile oynamak için, CONTROL portuna "232" verisi ya da "8" verisini göndermenin hiçbir farkı yoktur. Çünkü 232'nin ikilik sistemdeki karşılığı "11101000"dır. 8 in ikilik sistemdeki karşılığı ise "00001000"dır. Dikkat ederseniz, bizi sadece C3 pinine karşılık gelen veri ilgilendiriyor. Yani C7 – C4 arasına karşılık gelen verilerin CONTROL portu için hiçbir anlamı yoktur. Aşağıdaki tablolarda var olmayan bu pinlere karşılık gelen verileri "X" olarak işaretledim.

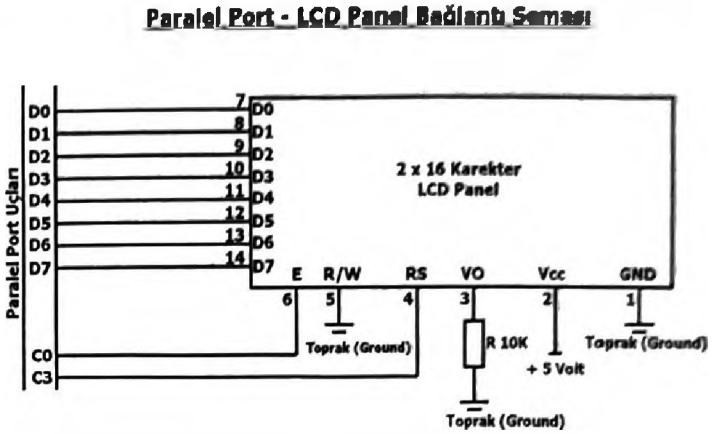
TABLO 4.2: Verinin CONTROL Portuna Yerleşme Şekli

Veri	C7	C6	C5	C4	C3	C2	C1	C0
232	X	X	X	X	1	0	0	0
232	1	1	1	0	1	0	0	0

Yukarıda da bahsettiğimiz gibi, biz sadece C3 ve C0 pinlerini kullanacağız. Dolayısıyla, sadece bu pinlerin "1" ya da "0" oluşu bizi ilgilendiriyor. LCD panelin RS ucunu, paralel portun 17 numaralı (C3) pinine, E ucunu ise paralel portun 1 numaralı (C0) pinine bağlıyoruz. Bizim CONTROL portuna göndereceğimiz verileri ve ne işe yaradıklarını aşağıdaki tabloda yazıyorum.

TABLO 4.3: RS ve E Uçlarına Gönderilecek CONTROL Portu Sinyalleri

Veri	İşlev	C7	C6	C5	C4	C3	C2	C1	C0
8	RS girişini düşük (0) yapar.	X	X	X	X	1	0	0	0
9	RS ve E girişlerini aynı anda düşük (0) yapar.	X	X	X	X	1	0	0	1
1	E girişini düşük (0), RS yi yüksek (1) yapar.	X	X	X	X	0	0	0	1
0	E ve RS girişlerini aynı anda yüksek (1) yapar.	X	X	X	X	0	0	0	0



ŞEKİL 4.2: LCD Panel kontrol devresi.

Dikkat ederseniz pine "1" verisini göndererek pini düşük yani "0" yapıyoruz. Bunun sebebi, kullandığımız C3 ve C0 pinlerinin yukarıda bahsettiğim gibi donanım tarafından terslenmiş olmasıdır. DATA portuna göndereceğimiz verilerde böyle bir terslenme problemi yaşamayacağız. Hangi portu yüksek (1) yapmak istiyorsak, o pine "1" verisi göndermeliyiz.

Yukarıdaki bu verileri kullanarak kontrol saklayıcısını ya da veri saklayıcısını seçeceğiz. Standart bir LCD panelde kontrol saklayıcısını seçtiğimiz zaman DATA pinleri aracılığı ile gönderebileceğimiz komutlar ve ne işe yaradıkları aşağıdaki gibidir.

TABLO 4.4: Standart LCD Panel Komut Saklayıcısı Fonksiyonları

İşlem	D7	D6	D5	D4	D3	D2	D1	D0	Açıklama
Ekranı Sil	0	0	0	0	0	0	0	1	Ekranı siler
Ekran seçenekleri	0	0	0	0	1	P	K	Y	(P) Panel AÇ (1) / KAPA (0)(K) Kürsör AÇ (1) / KAPA (0)(Y) Yanıp Sönme AÇ (1) / KAPA (0)
Fonksiyon ayarları	0	0	1	D	N	0	X	X	(D) Veri girişi 8-bit (1) / 4-bit (0)(N) Çift satır (1) / Tek satır (0)
CGRAM Adres ayarı (ilk satırın adresleri)	0	1	C5	C4	C3	C2	C1	C0	(C5-C0) CGRAM'e gidecek veriler.
DDRAM Adres ayarı (ikinci satırın adresleri)	1	D6	D5	D4	D3	D2	D1	D0	(D6-D0) DDRAM'e gidecek veriler.

LCD Panel Kontrol Yazılımı

Programımızda ilk olarak LCD paneli yazmaya programlamak için komut saklayıcısına sırayla 1, 15 ve 56 sayılarını gönderiyoruz. Tablo 4.4'ten de anlaşılacağı gibi 1 (00000001) verisini ekranı silmek için, 15 (00001111) verisini, paneli, imleci ve imlecin yanıp sönmesini (blink) açmak

için gönderiyoruz. 56 (00111000) verisi ile de LCD paneli 8 bitlik, çift satır veri girişi için ayarlıyoruz. Bu komutları göndermek için yarattığımız `komutGonder()` fonksiyonunu kullanıyoruz. Veri saklayıcısına veri (karakter) göndermek için ise `veriGonder()` fonksiyonunu kullanacağız.

Aşağıdaki fonksiyonlarda `data` değişkeni paralel portunuzun taban adresine, `kontrol` değişkeni ise CONTROL portunuzun adresine (taban adresi + 2) karşı gelmektedir. Benim paralel portumun taban adresi h378 idi. Dolayısı ile CONTROL portumun adresi de h378+2 yani h37A olacaktır. Bu sabitleri programın başındaki `CONST` değerleri ile tanımlıyoruz.

`komutGonder()` Fonksiyonu: Bu fonksiyon sayesinde LCD panelin komut saklayıcısını seçebilir ve Tablo 4.4'teki komutlardan istediğinizi gönderebilirsiniz. Bu fonksiyon ile ilk olarak gönderilecek komut, DATA portlarına yazılıyor. Daha sonra RS girişini düşük yaparak panelin komut saklayıcısı seçiliyor. Ardından, E ucu da düşük yapılıyor ve komut gönderilmiş oluyor. `FOR-NEXT` döngüleri ise panelin veriyi okuyabilmesi için gereken gecikmeleri sağlıyor.

`veriGonder()` Fonksiyonu: Bu fonksiyon sayesinde LCD panelin veri saklayıcısını seçebilirsiniz. Daha sonra yazdırmak istediğiniz karakterin ASCII kodu DATA pinlerine gönderilir. Ardından C0 pinine "0" gönderilerek pin yüksek yapılır ve ardından biraz beklenip "1" gönderilerek düşük (0) yapılır.



ŞEKİL 4. 3: LCD panel kontrol programı.

İmleci birinci satırın başına almak için `CGRAM` komutu gönderilir. Tablo 4.4'ten de anlaşılacağı gibi `CGRAM` (1.satır) için "01000000" yani 64 verisini göndermeliyiz. Her satırın birinci karakterinin adresi 64'tür. Yani birinci satırın birinci sütunu için 64 + 64 = 128 verisini komut saklayıcısına göndermeliyiz. İkinci karakter için ise 64+65=129 olmalı. Bu şekilde gidiyor. Fakat her seferinde bu adresleri bir bir girmenize gerek kalmayacaktır. Çünkü siz LCD panele karakter gönderdikçe LCD bu adresi otomatik olarak artıracak ve gönderdiğiniz karakterleri sıra ile yazdıracaktır. Biz bu komutu sadece hangi satıra yazmak istediğimizi belirtmek için kullanacağız. İkinci satırın ilk karakterine gitmek için `DDRAM` fonksiyonunu "10000000" ile çağıracağız. Bu satırda da ilk karakterin adresi 64 olduğundan 128+64=192 verisini komut saklayıcısına göndermemiz ikinci satırın başına gitmemiz için yeterli olacaktır. Buradaki `ASC` fonksiyonu ise içerisine yazdığımız karakterin ASCII kodunu döndürmeye yarıyor.

Aşağıdaki program örneği LCD panelin ilk satırına "S1", ikinci satırına ise "S2" yazmaya yarıyor.

LCD.BAS – LCD Panel Kontrolü VISUAL BASIC Kodu

```
Const data = &H378  ' Paralel portunuzun taban adresi
Const kontrol = &H378 + 2  ' CONTROL portunun adresi
Private Sub Command1_Click()
   KomutGonder 15 ' Paneli, kürsörü ve kürsörün yanıp sönmesini açıyoruz.
```

```
   KomutGonder 1 ' Ekranı temizliyoruz.
   KomutGonder 56 'LCD paneli 8 bitlik, çift satır veri girişi için ayarlıyoruz.
   KomutGonder 64+64   'Birinci Satırın birinci karekterine git.
   VeriGonder ASC("S") ' S harfinin karekter kodunu gönderiyoruz.
   VeriGonder ASC("1") ' 1 rakamının karekter kodunu gönderiyoruz.
   KomutGonder 128+64  'İkinci satırın birinci karekterine git.
   VeriGonder ASC("S") ' S harfinin karekter kodunu gönderiyoruz.
   VeriGonder ASC("2") ' 2 rakamının karekter kodunu gönderiyoruz.
End Sub

Sub komutGonder(veri As Integer)
   Out data, veri
   Out kontrol, 8
   For i = 1 To 100000: Next i
   Out kontrol, 9
   For i = 1 To 100000: Next i
End Sub

Sub veriGonder(veri As Integer)
   Out data, veri
   Out kontrol, 0
   For i = 1 To 100000: Next i
   Out kontrol, 1
   For i = 1 To 100000: Next i
End Sub
```



ŞEKİL 4.4: Programlanmış LCD panel görüntüsü.

BEŞİNCİ BÖLÜM

5

ANALOG GİRİŞ VE BASİT ROBOT GÖZÜ

Analog Sinyaller

Bizim bu bölüme kadar uğraştığımız sinyallerin hepsi sayısaldı. Yani, sinyalin "1" ya da "0" olması durumunu inceliyorduk. Dolayısı ile lojik voltajımızın değeri, "1" olduğunda +5 Volt, "0" olduğunda ise 0 Volt oluyordu. Analog sinyallerde ise bu voltajın değeri 0 ile 5 volt arasında değişebilir. Bir ADC ile (Analog - Digital Converter) fiziksel değerleri (ısı, basınç, kütle, ışık yoğunluğu vb.) sayısal değerlere çevirebiliriz. Böylece PC'mizi dış dünyayı izleme aracına çevirebiliriz. Daha sonra bu ölçümler sonucunda üst düzey kararlar vermeye programlayabiliriz. Örneğin, oda sıcaklığı belli bir dereceye ulaştığında PC'mizle ısıtma sistemini kapatabiliriz.

Bir sıcaklık sensörü, elde ettiği sıcaklığın sonucunda bize 0-5 volt arasında bir değer verebilir. Biz de bu değer üzerinden gerekli dönüşümü yaparak hangi voltajın hangi sıcaklığa denk geldiğini hesaplayabilir ve işlemler yapabiliriz. Herhangi bir ölçüm sonucunda elde edeceğimiz voltaj eğer 5 volt ise bu voltajı ADC ile sayısal veriye dönüştürdüğümüzde elde edebileceğimiz değer 255 olacaktır. Kısacası, 0 ile 5 volt arasındaki analog değerler, 8-bitlik sayısal veriye dönüştüğünde 0-255 arasında değerlere denk gelecektir.

Analog-Dijital dönüşüm yapılabildiği gibi bunun tam tersini de yapmak mümkündür. Örneğin, DATA pinlerine göndereceğimiz 0-255 arası sinyaller ile 0-5 Volt arasında bir voltaj gerilimi elde etmeniz mümkün olabilir. Ayrıca kullandığınız ADC ve DAC (Digital-Analog Converter) tipine göre analog giriş değerleri değişebilir.

Sayısal-Analog Dönüşüm Uygulaması

Bölüm 1'deki uygulamalarımızda paralel portun STATUS pinlerini kullanarak bir tür veri girişi gerçekleştirmiştik. Şimdiki uygulamamızda paralel portu kullanarak ilk gerçek veri girişini gerçekleştireceğiz.

Paralel portun DATA pinlerini bugüne kadar sadece veri çıkışı için kullandık. Yeni teknolojiye sahip paralel portlarda bu DATA pinlerini 8-bitlik veri girişi için kullanmak da mümkündür. Bu işlemi gerçekleştirebilmek için ana kartınızın Bi-Directional port tipini desteklemesi gerekmektedir. Bilgisayarınızın BIOS ayarlarına girdiğinizde, paralel port tipini değiştirmeniz mümkün olacaktır. DATA pinleri ile veri girişi yapabilmemiz için port tipinizi Bi-Directional yapmalısınız.

Bi-Directional paralel port ile veri girişi STATUS portundan farklıdır. STATUS portu ile girişi, pinin üzerindeki "1" lojik voltajını topraklayarak gerçekleştiriyorduk. DATA portları ile veri girişini ise, bu pinlere lojik voltaj göndererek gerçekleştirebiliyoruz.

DATA portundan 8 bitlik veri çıkışını tekrar hatırlayalım. DATA portuna hiçbir veri göndermediğimiz zamanki değeri "00000000"dır. Dikkat ederseniz, 8 tane "0" var. Örneğin, data portuna 25 değerini gönderelim. 25 değerinin ikilik sayı sisteminde karşılığı "00011001"dir. Bu durumda D4, D3 ve D0 pinlerine karşılık gelen lojik değerler "1" olduğundan o pinler +5 Volt olacaktır. Bu uygulamamızda ise, Bi-Directional port desteği sayesinde DATA portlarındaki veriyi okuyacağız. Yani D4, D3 ve D0 pinlerine karşılık gelen pinlere +5 voltluk lojik voltaj uyguladığımızda, bu porttan okuyacağımız değer 25 olacaktır.

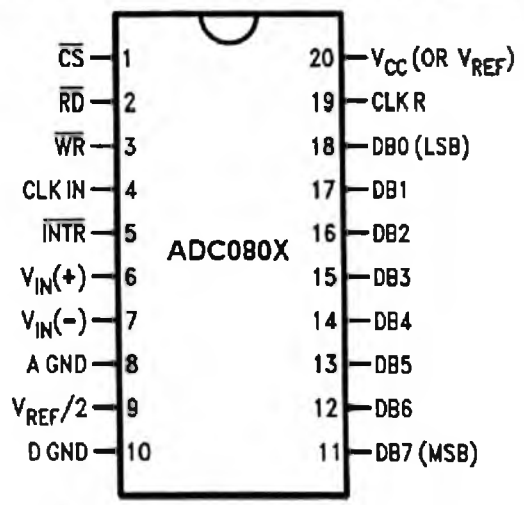
Sayısal - Analog Dönüştürücü (ADC) Devresi

Devre için ihtiyacımız olan malzemeler aşağıdaki gibidir.

- 1 adet ADC0804 analog sayısal dönüştürücü
- 1 adet 150 pF kondansatör
- 1 adet 10mF kondansatör (isteğe bağlı)
- 1 adet 10K direnç
- 1 adet BP109 ya da uyumlu foto transistör
- 1 adet 1,5 Volt kalem pil

Bir ADC entegresini programlamak oldukça kolaydır. Bölüm 4'te anlattığım LCD panel kontrolüne benzer. ADC'yi programlayabilmek için veriyi okuyacağımız DATA pinlerinin dışında, ADC'nin RD ve RW pinlerini kontrol etmek için 2 tane CONTROL pinine, ERROR kontrolü içinde 1 tane de STATUS pinine ihtiyacımız olacak. Kullandığımız ADC0804, National Semiconductor firmasının yaygın olarak kullanılan analog-sayısal dönüştürücülerinden biridir. Entegrenin yapısı hakkında daha detaylı bilgi için `http://www.national.com` adresine bakabilirsiniz.

ADC080X ANALOG/SAYISAL Dönüştürücü



ŞEKİL 5.1: ADC080X analog-sayısal dönüştürücüsü.

TABLO 5.1: ADC0804 Pinlerinin İşlevleri

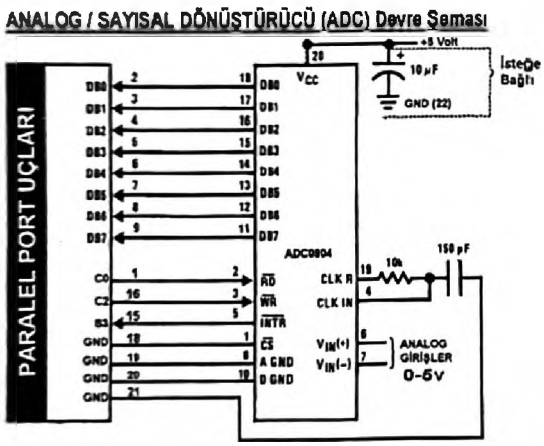
ADC0804	Pin İşlevi
1	Toprak (Ground)
2	RD Okuma modu
3	WR Yazma modu
4	CLK In sayaç giriş (150pF Kondansatör ile sağlanacak)
5	Intr (Hata kontrolü için kesme üretir)

TABLO 5.1: ADC0804 Pinlerinin İşlevleri

ADC0804	Pin İşlevi
6	Analog V+ (Analog giriş)
7	Analog V- (Analog giriş)
8	Analog toprak
9	Vref/2 (Devreye bağlamıyoruz) NC (Not Connected)
10	Sayısal toprak
11-18	D7-D0 (Sayısal çıkışlar)
19	CLK R Sayaç
20	Vcc +5 Volt çalışma voltajı

ADC Devresi Nasıl Çalışıyor?

Önce CONTROL portuna "32" sayısal değerini göndererek, C0 pinini "0" pini yüksek yapıyoruz. Çünkü C0 pini donanım tarafından terslenmiştir. Dolayısı ile pine "0" gönderdiğimiz zaman o pin +5 volt oluyor. Sonuçta, ilk işlem C0 pinine bağlı olan RD pinini yüksek yapmak oluyor. Ardından, "36" sayısal değeri ile de WR pinini de yüksek yapıyoruz. Böylece entegre ADC veri dönüşümü için ayarlanmış oluyor. Daha sonra dönüştürülmüş sayısal değeri okumak için C2 pini yüksek tutulurken, C0 pini düşük yapılıyor. Veri bu sırada okunuyor ve hemen ardından C0 tekrar yüksek yapılıyor.



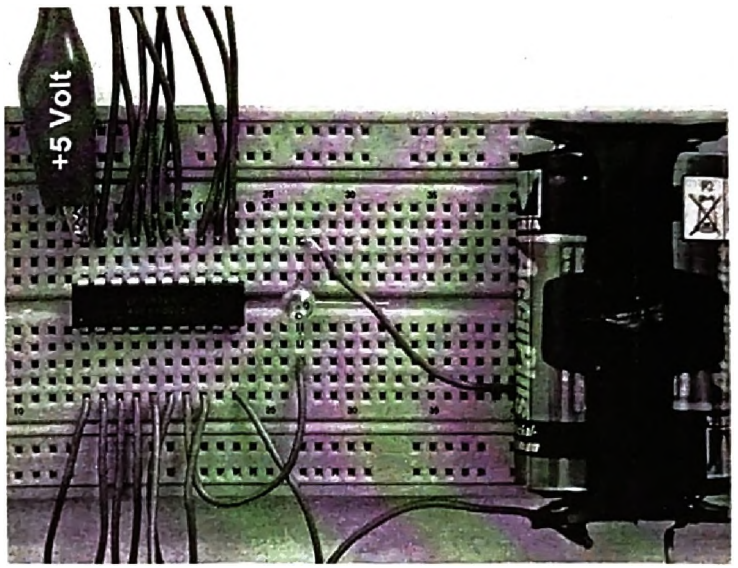
ŞEKİL 5.2: ADC ile paralel portun bağlantısı.

Örnek programlarımız, ADC'nin 6 ve 7 numaralı konnektörlerine uygulanan 0-5 volt arası değerlerin 0-255 arasındaki sayısal değerlerini ekrana yazdırmaya yarıyor. Örneğin, bu analog çıkışlara bir tane kalem pil bağladığınız zaman, ekranda elde edeceğimiz değer 70-80 arasındadır. Gerçekten de 5 volt için elde edilecek değer 255 ise, 1,5 Volt için yaklaşık değer, tam dolu bir pil için 76'dır.

TABLO 5.2: ADC'nin RD ve WR Konnektörlerine Gönderilecek Control Portu Sinyalleri

Veri	İşlev	C7	C6	C5	C4	C3	C2	C1	C0
32	RD (2) girişini yüksek (1) yapar.	X	X	X	X	0	0	0	0
36	RD (2) ve WR (3) girişlerini aynı anda yüksek yapar.	X	X	X	X	0	1	0	0
37	WR'yi (3) yüksek , RD'yi (2) düşük yapar	X	X	X	X	0	1	0	1

ADC entegreleri oldukça hassastır. Yani ortamdaki voltajdan kolayca etkilenebilir. Elinizle analog girişlere dokunduğunuz zaman bile, ekrandaki sayısal değerde 1-2 derecelik bir artış meydana gelecektir. O bakımdan devreyi ortam voltajlarından etkilenmeyecek bir şekilde yapmanız gerekmektedir. Örneğin, toprak olarak birbirinden farklı toprak kanallarını kullanmanızı öneriyorum. Paralel port üzerinde 18-25 arası pinler topraktır. ADC'nin 8 ve 10 numaralı konnektörlerini birbirinden farklı topraklara bağlamak bu etkilenmeyi azaltacaktır.



ŞEKİL 5.3: ADC devresinin board üzerinde kurulmuş hali.

Elde edilen sayısal değerler ile bazı kararlara varmadan önce, tavsiyem birden fazla değerin ortalamasını almak ve belirli bir aralıkta ise bir işlem yaptırmak. Şimdi anlatacağım robot gözü uygulamasında bu yöntemden yararlanmanız mümkündür.

Cismin Uzaklığını Işık Şiddeti ile Algılamak

Basit Robot Gözü Uygulaması

Temel olarak gerçekleştireceğimiz uygulama oldukça basit. Foto transistör üzerine düşen ışık sayesinde, transistörün değişen iç direncini kullanarak ışığa bağımlı bir analog sinyal elde edeceğiz. Eğer foto transistöre bir cisim yaklaşırsa, üzerindeki ışık kaynağı azalacak ve iç direnç artacaktır. Direnç arttığında ise analog sinyalin değeri düşecektir.



ŞEKİL 5.4: Transistörün ADC'ye bağlanması.

Foto Transistörler

Foto transistörlerin normal transistörlerden tek farkı beyz ucundan yapılan tetiklemenin ışık şiddeti ile yapılmasıdır. Yine normal transistörler gibi bir beyz ucu vardır ve istenirse bu uç kullanılarak da tetiklenebilir. Devreye normal bir transistör gibi bağlanır, ancak normal şartlarda beyz ucu kullanılmaz. Tetikleme işlemi, üzerindeki merceğe ışık uygulanarak yapılır. Uygulanan ışığın şiddetine göre Emiter-Kollektör arasındaki iç direnç değişir ve yükün akımı kontrol edilebilir. Işığın şiddeti ile Emitör-Kollektör arası iç direnç ters orantılıdır. Yani, ışık şiddeti arttıkça Emiter-Kollektör arası iç direnç düşecektir. Foto transistörler kızıl ötesi (Infra Red) ışınlara daha hassas olduğu için genellikle ışık kaynağı olarak kızıl ötesi (Infra Red) ledler ile birlikte kullanılır.

Buradaki tek dezavantajımız, foto transistörün gün ışığından etkilenmesidir. Foto transistörü, ADC'nin analog girişlerine bağladığınızda, ekrandan okuduğunuz sayısal değer normal aydınlatmada, 100-150 civarındadır. Transistöre cisim yaklaştıkça, yani ışık şiddeti düştükçe okunan bu değer 5-10'lara kadar düşebilmekte ve böylece cismin yaklaştığını anlamak mümkün olmaktadır.

Eğer çok aydınlık bir ortamda iseniz, okunan bu değişimlerin değerleri farklı olacaktır. Bu problemden kurtulmak için ise, ortam ışığının değerini robotunuza bildirmek yeterli sayılır. Bunun için, robotun önüne koyacağınız cisim algılama amaçlı foto transistörlerin yanında, robotun tepesine öndeki cisimlerin gölgelerinden etkilenmeyecek birkaç foto transistör yerleştirmek ve bu transistörlerden okunan değerler ile bir ortam ışığı sayısal değeri belirleyip, bu değerler üzerinden çalışma yapmak gerekebilir.

ADC veri okuma programı VISUAL BASIC kodu

```
Const data = &H378
Sub Command1_Click()
Screen.MousePointer = 11
Dim value As Integer, t As Integer, i As Integer
value = 0
    For i = 0 To 5000
        Out data + 2, 32
            Label2.Caption = "/": Label2.Refresh
            For t = 1 To 15000: Next t 'Gecikme için
            Label2.Caption = "\": Label2.Refresh
        Out data + 2, 36
```

```
            For t = 1 To 15000: Next t
            Do While (((Inp(data + 1) And 8) = 0))
                Out data + 2, 37 ' set CO- ('D' pin 1 düşük yapar)
                For t = 1 To 15000: Next t
                value = Inp(data)
                Out data + 2, 36 ' unset CO- ('D' pin 1 yüksek yapar)
                For t = 1 To 15000: Next t
                Label1.Caption = value
                Label1.Refresh
            Loop
    Next
    Screen.MousePointer = 0
    End Sub
```

Bir diğer olasılık da, robotunuzun tamamen karanlık bir ortamda kalmasıdır. Düşünülenin tam tersine karanlık ortamlar robotların yön bulabilmeleri için mükemmel ortamlardır. En önemlisi, ortam ışığı yoktur. Foto transistörün arkasına ya da yanına bir ışık kaynağı takılır. Robot ilerleyince, eğer önüne bir cisim çıkarsa, arkadan gelen bu ışık cisim üzerinden foto transistörün merceğine yansır.

Son anlattıklarım biraz teoride kalıyor, çünkü üzerinde paralel port ve bir PC taşıyan bir robot yapmak biraz zor olabilir. Ama tabii ki imkansız değildir. İleride PIC programlamayı öğrendiğinizde, bu devreleri PC'den bağımsız hale getirebileceksiniz. PIC'ler ufak mikroişlemcilerdir. Üzerlerinde birden fazla paralel port vardır. Düşük enerji ile çalışırlar ve az yer kaplarlar. Fakat PIC konusuna geçmeden önce paralel port ile ilgili uygulamaları iyi kavramış olmak gerekir.

ALTINCI BÖLÜM

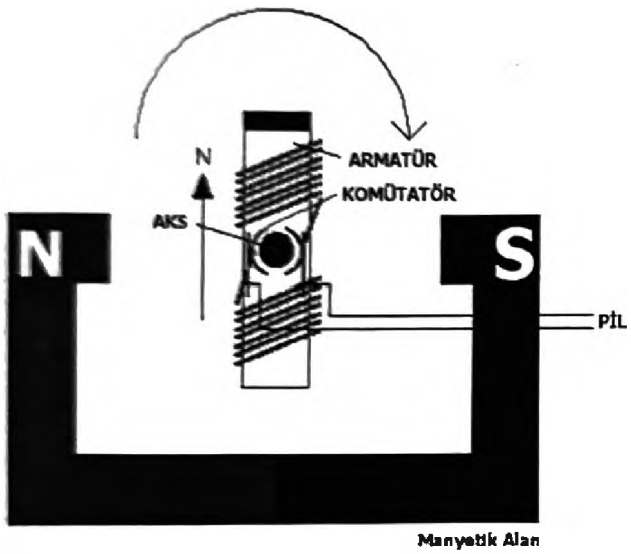
6

DC MOTOR KONTROLÜ

DC Motorlar

DC motorlar birçok elektronik cihazda yaygın olarak kullanılmaktadır. En basitinden, bilgisayarınızın işlemcisini soğutmaya yarayan fan pervanesi, bir DC motor ile çalışır. DC motorların PC'ye bağlanarak robotik ve mekanik araçlarda kullanılması yaygınlık kazanmıştır. Örneğin, simetrik metal parçaların imalatında kullanılan CNC tezgahları DC motorlar içermektedir.

DC motorların çalışma prensipleri çok basit temellere dayanır. Bir çoğumuz bir mıknatısı inceleme fırsatı bulmuşuzdur. Mıknatısın iki kutbu vardır. Bunlar, N (North) ve S (South) kutuplarıdır. Zıt kutuplar birbirini çeker ve aynı kutuplar birbirlerini iterler. DC motorlar, manyetik kutupların birbirleri ile bu etkileşim ilişkisi üzerine kurulmuş, dönme hareketi sağlayan cihazlardır. DC motorlar hakkında daha detaylı bilgi için `http://dummy.kreta.dyn.priv.at/temp/howstuffworks/motor.htm` adresine mutlaka göz atın.



ŞEKİL 6.1: DC motorun iç yapısı.

DC Motor Kontrol Uygulaması

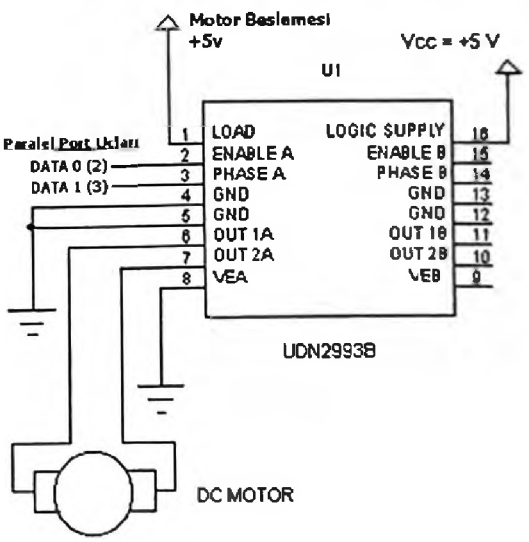
DC motor kontrol etmenin birçok yolu vardır. DC motora uygulayacağımız akımın yönüne göre motorun yönü, voltajın büyüklüğüne göre de hızı değişecektir. Kontrol yöntemlerinden bir tanesi de DAC (Digital-Analog Converter) kullanmaktır. Bir DAC'ye gönderilen 8 bitlik sinyal sonucunda istediğimiz voltajı elde etmemiz mümkün olabilir. Fakat bu işlem biraz daha karışıktır ve yükseltici devre gerektirir. DAC kullanmak, Bölüm 5'teki Analog-Sayısal dönüşümüne çok benzer bir olaydır. Halbuki DC motor kontrolü için üretilmiş çok verimli yongalar vardır. Bunlardan bir tanesi Allegro firması (`http://www.allegromicro.com`) tarafından üretilen H-Bridge sürücüsüne sahip ve 30V ve 0.5 amperlik yüklere dayanıklı UDN2993B entegresidir. Fakat bu entegreyi bulmak konusunda sıkıntı yaşayabilirsiniz. Bu nedenle, aynı işlevi gören L293B adlı bir entegre ile aynı uygulamayı gerçekleştireceğiz. Bu motoru, tezgahlardan alabileceğiniz küçük oyuncak arabaların içerisinden sökebilirsiniz.

UDN2993 ile DC Motor Kontrol Devresi

UDN2993B entegresinin, motorun yönünü ve hızını belirlemek için iki konnektörünü kullanıyoruz. Bunlardan bir tanesi PHASE A (3) diğeri ise ENABLE A (2) girişleridir. Bu entegre ile iki tane motor kontrol etmek mümkündür. 2-8 numaralı konnektörler birinci motor için, 9-15 arası konnektörler ise ikinci motor için tasarlanmıştır. PHASE girişi motorun yönünü belirlemeye yarar. Bu girişe uygulayacağımız lojik voltaj ile motorun dönme yönünü değiştirebiliriz. ENABLE girişine uygulayacağımız PWM (Pulse Width Modulation) sinyalleri ile de motorun hızını kontrol edebiliriz.

PWM sinyalini yazılım ile üretmek çok kolaydır. Temel olarak, motorun dönme hızı, darbeler şeklinde ENABLE girişine uygulanacak lojik voltajın darbe zaman aralığına bağlıdır. PWM sinyali, motor hızını kontrol etmek için başvurulan verimli bir yöntemdir. PWM bisiklet binmeye benzer. Bisikleti sürerken, pedal çevirdiğimiz ve çevirmediğimiz zamanlar olur. Yani bazen kendi momentinizle hareket ederiz. Çevirdiğimiz zamanları on-time, çevirmediğimiz zamanları off-time olarak düşünelim. Eğer biraz yavaşladığımızı hissedersek, tekrar pedal çevririz. PWM sinyalleri de bu mantığa dayanır. PWM sinyalleri görev çevrimi (duty cycle) oranı ile tanımlanır. Görev çevrimi süresi, on-time süresi ile toplam süre arasındaki orana eşittir. 100% DC (duty cycle) demek, motor tam performans ile çalışıyor demektir. Eğer motorun hızı ile oynamak istemiyorsanız, yapmanız gereken sadece ENABLE (2) girişine, sürekli lojik "1" yani + 5 Volt uygulamaktır. Bu durumda motor tam güç ile çalışacaktır.

DC Motor Kontrol Devresi

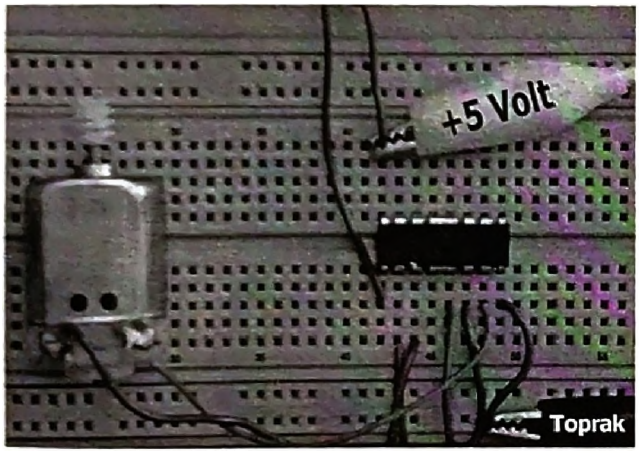


ŞEKİL 6.2: UDN2993 ile DC Motor kontrol devresi.

Paralel portun DATA0 pinini UDN2993B'nin ENABLE (2) girişine, DATA1 pinini de PHASE (3) girişine bağlıyoruz. ENABLE girişine yukarıda bahsettiğimiz şekilde PWM sinyallerini DATA0 pininden uyguluyoruz. PHASE girişine ise DATA1 pinini yüksek (1) ya da düşük (0) yaparak yön değiştirmeyi sağlıyoruz. Dolayısı ile paralel porta, motoru bir yönde döndürmek için "1" değe-

rini gönderiyoruz. Diğer yönde döndürmek için ise, DATA0'a PWM sinyallerini uygularken DATA1'i de "1" yapmamız gerekiyor. Bunun için paralel porta her iki biti de yüksek yapacak "3" değerini gönderiyoruz. Kısacası, motor bir yönde dönerken paralel porttan uygulanan sinyaller 1-0-1-0-1-0-1-0-... şeklinde giderken, diğer yöne çevirdiğimizde, 2-3-2-3-2-3-2-3-... şekline dönüyor. Bu şekilde, PHASE biti yüksek tutulurken, ENABLE bitine PWM sinyaller uygulanmaya devam ediliyor.

Aşağıdaki program örneği Visual Basic altında PWM sinyalleri ile DC motor kontrol etmeye yarıyor. Tipik olarak DC motor kontrol devreleri, zamanlama devreleri ile birlikte tasarlanmaktadır. Biz zamanlama olayını yazılım aracılığı ile çözüyoruz. Böylece devremiz hem daha verimli hem de daha hesaplı oluyor. Motorun en verimli çalışma hızını elde etmek için, PWM fonksiyonundaki `total_time` değişkeni ile oynayabilirsiniz. Bu değer 0.1 saniye olarak ayarlanmıştır. Bu değeri değiştirdikçe motorun çalışma hızı ve şekli değişecektir.



ŞEKİL 6.3: UDN2993 kullanılan devrenin board üzerinde kurulmuş hali.

DELAY Fonksiyonu: Bu fonksiyon içerisine, saniye cinsinden girilen `delay_time` değeri kadar bekletme sağlar.

PWM Fonksiyonu: Bu fonksiyon, aldığı `duty_cycle` değerine ve `total_time` değerine göre `OnTime` ve `OffTime` sürelerini hesaplayıp, DATA0 çıkışına yani entegrenin ENABLE girişine, belli aralıklarda lojik "1" uyguluyor. Burada kullanılan PHASE değişkeni motorun yönünü değiştirmeye yarıyor.



ŞEKİL 6.4: DC Motor kontrol programı.

UDN2993 için PWM sinyali ile DC Motor kontrolü, Visual Basic Kodu

```
Public Enable As Integer, Phase As Integer, duty_cycle As Integer, a As Integer
Const PortA = &H378 'Paralel port adresi
```

```
Sub delay(delay_time As Double)
    'Gecikmeyi sağlayan fonksiyon
    temp_timer = Timer
    Do
    Loop While (Timer - temp_timer) < delay_time
End Sub

Sub pwm(total_time As Double, duty_cycle As Integer)
    Dim OnTime As Double
    Dim OffTime As Double
    OnTime = total_time * duty_cycle / 100
    OffTime = total_time - OnTime
    Enable = 1
    Out PortA, Enable + Phase
    delay OnTime
    Enable = 0
    Out PortA, Enable + Phase
    delay OffTime
    Label1.Caption = duty_cycle
    Label1.Refresh
End Sub

Private Sub Command1_Click()
If duty_cycle < 100 Then
    duty_cycle = duty_cycle + 5
    Enable = 1
    Label1.Caption = duty_cycle
    Label1.Refresh
End If
End Sub

Private Sub Command2_Click()
a = 0
Do
    DoEvents
    If a = 1 Then Exit Do
    pwm 0.1, duty_cycle
Loop
End Sub
Private Sub Command3_Click()
If duty_cycle > 5 Then
    duty_cycle = duty_cycle - 5
    Enable = 1
    Label1.Caption = duty_cycle
    Label1.Refresh
End If
End Sub

Private Sub Command4_Click()
'Ters yönde çalıştırma rutini
Enable = 1
If Phase = 0 Then
    Phase = 2
Else
    Phase = 0
```

```
End If
End Sub

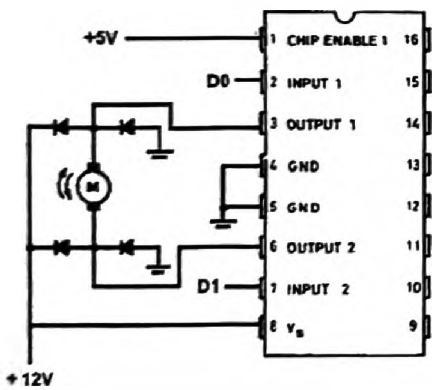
Private Sub Command5_Click()
'Durdurma ve Do-Loop'tan çıkış
a = 1
Out PortA, 0
End Sub

Private Sub Form_Load()
duty_cycle = 50
Phase = 0
End Sub
```

L293B ile DC Motor Kontrol Devresi

Allegro Firmasının ürettiği UDN2993 entegresini bulmakta sıkıntı yaşarsanız, aynı uygulamayı SGS THOMSON firması tarafından üretilen L293B ile de gerçekleştirebilirsiniz. Fakat bu durumda, bağlantılarda ve yazılımda birkaç değişiklik yapmak gerekecektir. Kitabın uygulama dosyalarında her iki entegre için yazılmış örnek programı bulabilirsiniz. L293B ile iki ayrı motor kontrol edebilirsiniz. Çalışma prensipleri çok basittir. Motoru besleyeceğimiz gerilim entegrenin 8 numaralı pinine sağlanır. Üzerindeki OUTPUT çıkışları, INPUT girişlerine herhangi bir sinyal verilmedikçe 0 Volt durumundadır. Eğer INPUT1 girişine lojik "1" uygularsanız, 8 numaralı pinden gelen besleme OUTPUT1 çıkışına akacaktır. Aynı şekilde INPUT2 girişi, beslemeyi OUTPUT2'ye aktaracaktır. Motor, OUTPUT1 ve OUTPUT2 pinlerine takılır. Bu durumda gerilim hangi çıkışa yönlendirilirse motor o yönde dönecektir.

LM293B, motor beslemesini "8" numaralı pini üzerinden almaktadır. Ben motor beslemesi için 12V kullandım. Ayrıca LM293'ü çalışır durumda tutabilmek için "1" numaralı ENABLE pinini sürekli +5 Volt ile beslemelisiniz. "4" ve "5" numaralı pinleri Paralel portun topraklarından birine bağlayabilirsiniz. Bunların dışında Şekil 6.5'te görüldüğü gibi, OUTPUT çıkışlarına, portu motordan geri dönebilecek ters EMK'ya karşı korumak amacı ile dört tane diyot kullanabilirsiniz. Bunun için diyot olarak 1N4007 kullanabilirsiniz.



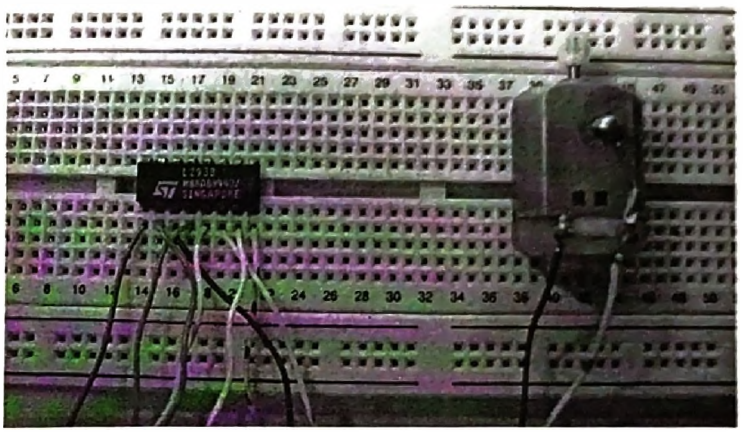
ŞEKİL 6.5: LM293 ile DC Motor kontrol devresi.

INPUT girişleri için gereken lojik sinyalleri paralel portun D0 ve D1 pinlerinden elde ediyoruz. Bu durumda eğer motoru bir yöne çevirmek istiyorsak, DATA portuna "1" değerini, diğer

yöne çevirmek istiyorsak, "2" değerini gönderiyoruz. Bu değerlerin DATA portuna ve INPUT girişlerine nasıl yerleştiğini Tablo 6.1'de görebilirsiniz.

TABLO 6.1: Motoru Farklı Yönlere Döndürmek İçin Data Portuna Gönderilen Sinyaller

Dönüş Yönü	Değer	D7	D6	D5	D4	D3	D2	D1 (INPUT2)	D0 (INPUT1)
Sağ	1	0	0	0	0	0	0	0	1
Sol	2	0	0	0	0	0	0	1	0



ŞEKİL 6.6: L293B kullanılan devrenin board üzerinde kurulmuş hali.

Aşağıdaki Visual Basic kodunda, UDN2993'ün kodundan farklı olarak, **Phase** ve **Enable** değişkenleri çıkarılmıştır. Bunun yerine veri **input_data** değişkeni üzerine yüklenmektedir. PWM fonksiyonunda ise, DATA portuna **OnTime** beklemesi için **input_data** içerisindeki veri, **OffTime** beklemesi için"0" değeri gönderilmektedir. Örneğin motor sola dönüyor olsun. Bu durumda **input_data** değişkeninin değeri "2" olacaktır. Dolayısı ile, PWM sinyali 2-0-2-0-2-0 şeklinde porta gidecektir. Eğer **input_data** değeri "1" olursa, yani motor sağa dönerse, bu sefer de PWM sinyali 1-0-1-0-1-0-1 şeklinde porta gidecektir.

L293B için PWM sinyali ile DC Motor kontrolü, Visual Basic Kodu

```
Public input_data As Integer, duty_cycle As Integer, a As Integer
Const PortA = &H3BC 'Paralel port adresi

Sub delay(delay_time As Double)
    'Gecikmeyi sağlayan fonksiyon
    temp_timer = Timer
    Do
    Loop While (Timer - temp_timer) < delay_time
End Sub

Sub pwm(total_time As Double, duty_cycle As Integer)
    Dim OnTime As Double
    Dim OffTime As Double
```

```
    OnTime = total_time * duty_cycle / 100
    OffTime = total_time - OnTime
    Out PortA, input_data
    delay OnTime
    Out PortA, 0
    delay OffTime
    Label1.Caption = duty_cycle
    Label1.Refresh
End Sub

Private Sub Command1_Click()
If duty_cycle < 100 Then
    duty_cycle = duty_cycle + 5
    Label1.Caption = duty_cycle
    Label1.Refresh
End If
End Sub

Private Sub Command2_Click()
a = 0
Do
    DoEvents
    If a = 1 Then Exit Do
    pwm 0.1, duty_cycle
Loop
End Sub
Private Sub Command3_Click()
If duty_cycle > 5 Then
    duty_cycle = duty_cycle - 5
    Label1.Caption = duty_cycle
    Label1.Refresh
End If
End Sub

Private Sub Command4_Click()
'Yön çevirme rutini
If input_data = 1 Then
    input_data = 2
Else
    input_data = 1
End If
End Sub

Private Sub Command5_Click()
'Durdurma ve Do-Loop tan çıkış
a = 1
Out PortA, 0
End Sub

Private Sub Form_Load()
duty_cycle = 50
input_data = 1
End Sub
```

YEDİNCİ BÖLÜM

7

BİLGİSAYAR KONTROLLÜ MODEL ARABA

Radyo Sinyalli Model Arabalar

Bu bölümde kumandalı oyuncak arabayı, bilgisayarın klavyesindeki yön tuşları ile nasıl kumanda edebileceğimizi öğreneceğiz. Ben uygulamayı gerçekleştirdikten sonra, uzun bir süre başından kalkamadım. Çünkü klavye ile kumanda etmek, normal kumandadan daha zevkli. Sanki oyun oynarmış gibi oluyor. Çok zevkli ve kolay bir uygulama. Üstelik maliyeti de çok düşük. Yapacağımız uygulama basit bir oyuncağı kontrol etmek gibi görünse de, işin temelinde elektronik cihazlara PC ile nasıl müdahalede edebileceğimiz var. Uygulama için ilk gereken şey tabii ki bir uzaktan kumandalı araba. Dikkat etmeniz gereken, ön tekerlekleri iki yönde dönebilen bir araba olması. Bazı kumandalı arabalarda, tek manevra düğmesi vardır ve buna bastıkça araba geriye doğru dönerek manevra yapar. Bu durumda istediğimiz yere yönlendirmemiz zor olacaktır. Kısacası, uzaktan kumanda üzerinde, sağ, sol, ileri ve geri tuşları mutlaka olmalıdır. Bazı kumandalar, joystick şeklindedir. Yani üzerinde 2 tane kol vardır. Bu tip kumanda da olabilir, fakat mümkünse tuşlu olsun ki, daha rahat müdahale edebilelim.



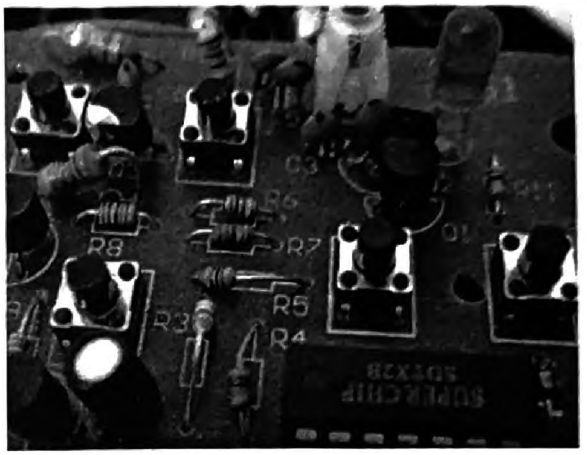
ŞEKİL 7.1: Uzaktan kumandalı model araba.

Paralel Porttan Model Araba Kontrol Uygulaması

Ben Şekil 7.1'deki oyuncak arabayı, Toys 'R' Us mağazasından aldım. Plastiği biraz kötü, fakat bizim uygulamamız için yeterli. Yapacağımız temel işlem, kumanda üzerindeki tuşları, bilgisayarın paralel portunu kullanarak anahtarlamak. Bunun en basit yolu, 4 tane röle kullanmak ve bir röle kontrol devresi ile tuşları anahtarlamak. Bu yöntemle zaten istediğiniz her elektronik devreyi anahtarlayabilirsiniz. Fakat bu yöntem biraz külfetli olacaktır. En azından, röleler için enerjiye ihtiyacımız olacak. Bu durumda devre için bir ek besleme gerekecek. Bizim kullanacağımız yöntem daha kolay ve daha az enerjiye ihtiyaç duyuyor.

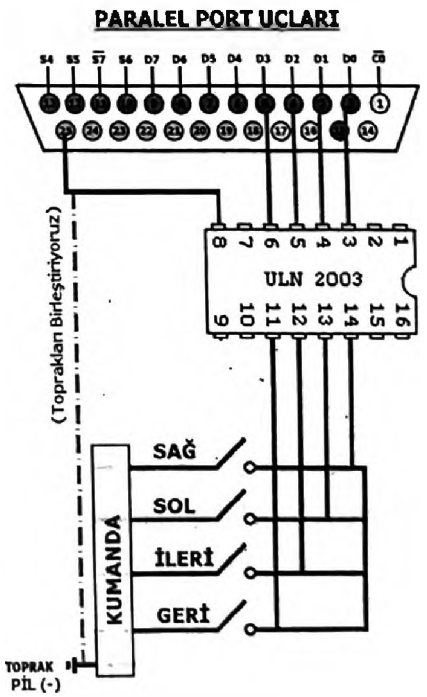
İlk olarak kumandayı dikkatlice sökün. Her bir tuşun altına denk gelen birer buton göreceksiniz (Şekil 7.2). Bizi ilgilendiren, sağ, sol, ileri ve geri butonları. Temel olarak, devre üzerindeki herhangi bir butona basıldığı zaman, devre topraklanıyor. Yani her butonun bir ucu ortak olan toprağa gidiyor. Bunu iyi anlamak için, bir tane krokodil kabloyu pilden gelen (–) uca bağlayın ve butonların arka taraflarına, tek tek krokodil kablonun öteki ucunu değdirin. Sanki butona basmışsınız gibi devrenin çalıştığını göreceksiniz. Bu durumda, her butonun dev-

reyi ortak bir toprak bağlantısına anahtarladığını göreceksiniz. Eğer biz, butonların topraklanması gereken uçlarına dışarıdan toprak uygularsak, devre yine çalışacaktır.



ŞEKİL 7.2

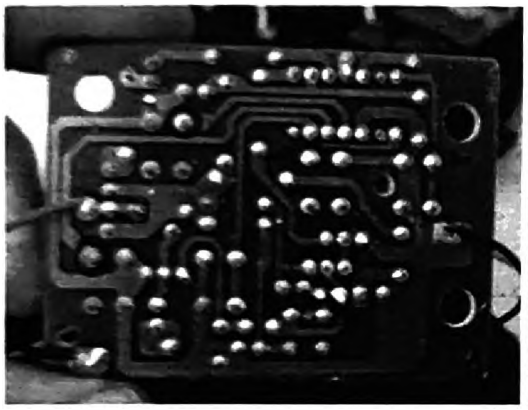
Yapacağımız işlem, butonların anahtarlanması için gereken toprağı bir devre aracılığı ile devreye göndermek. Bu devrede de her zamanki gibi ULN2003 entegresi yardımımıza yetişiyor. ULN2003 İçerisinde 7 adet NPN transistör ve dahili diyot barındırmaktadır. Kullanımı ise oldukça kolaydır. Devre şemasından da anlaşılabileceği gibi 3, 4, 5 ve 6 numaralı pinlere paralel portun DATA pinlerinden gelen +5 Voltluk değerleri uygulayacağız. Bu sayede, örneğin 3 numaralı pine +5 Volt (lojik voltaj) uyguladığımızda, 14 numaralı pin toprak olacaktır. Aynı şekilde, sırayla 4 için 13, 5 için 12, 6 için ise 11 numaralı pinler toprak olacaktır.

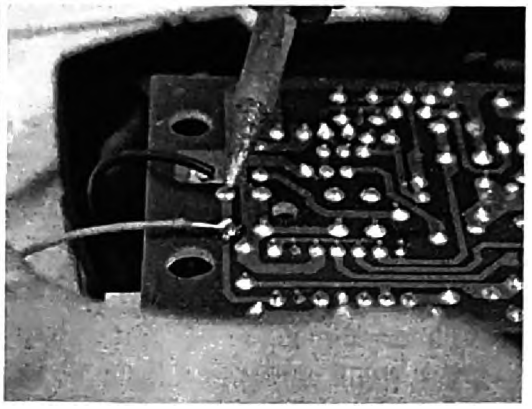


ŞEKİL 7.3: ULN2003 ile port ve kumandanın bağlantısı.

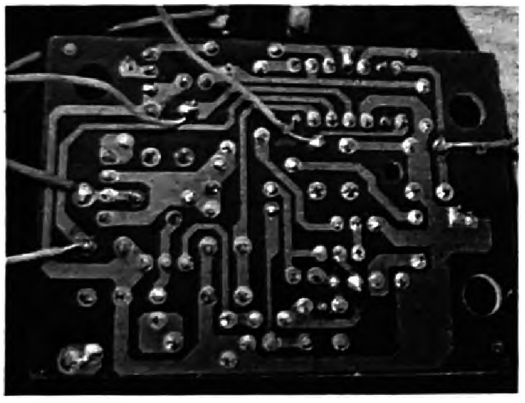
Burada dikkat etmeniz gereken nokta, paralel portun toprağı ile kumanda devresinin toprağını birleştirmek. Bu durumda, ULN2003 entegresine gerekli sinyalleri uygulayarak elde ettiğimiz toprak, kumanda üzerinde de geçerli olacaktır. Devrenin şemasını, Şekil 7.3'te görebilirsiniz.

Kumanda devresinin içerisinde saplama yapacağımız gerekli noktaları, yukarıda bahsettiğim gibi, krokodil kablo ile toprak gönderip deneyerek bulduktan sonra, bir kalem ile işaretleyin (Şekil 7.4). Bu aşamada havya ile, saplama yapacağımız noktalardaki lehim miktarını biraz arttırmakta fayda var. Bunun için o noktayı ısıtarak biraz lehim yedirin. Daha sonra, ucunu soyduğunuz bir montaj kablosunun ucunu yine havya ile ısıtarak bu noktanın içerisine sokun (Şekil 7.5). Lehimleme işleminin sonucunda Şekil 7.6'daki gibi bir görüntü elde etmiş olmalısınız.



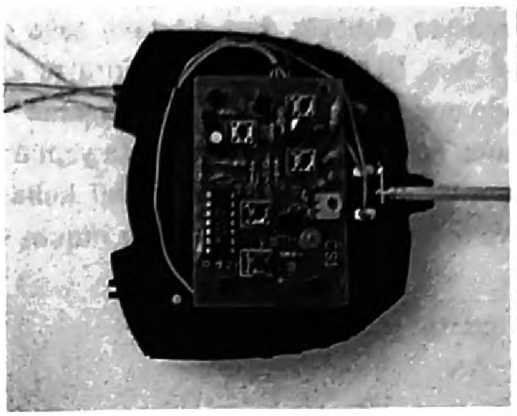


ŞEKİL 7.4: (solda) ve **ŞEKİL 7.5:** (sağda).



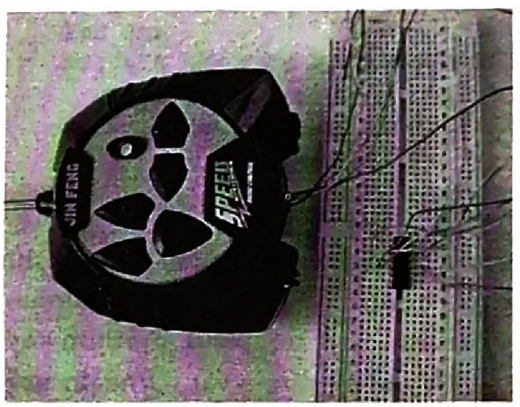
ŞEKİL 7.6

Daha sonra, devreyi tekrar eski yerine oturtun (Şekil 7.7). Bir tane de toprak için, pilin (–) ucundan gelen kabloya saplama yapmayı unutmayın. Daha sonra, kumandanın arka tarafını havya ile hafif eriterek, kabloları çıkarmak için bir delik açabilirsiniz (Şekil 7.8). Sıra gerekli uçları ULN2003 entegresine bağlamaya geldi. Ben bunun için yine delikli board kullandım (Şekil 7.9). Daha taşınabilir bir şey yapmak istiyorsanız. Bu devreyi kumandanın içine de sokabilirsiniz. Hatta kumanda üzerinde bir paralel port çıkışı bile koyabilirsiniz. İşin o kısmı sizin yaratıcılığınıza kalmış. ULN2003 ile paralel port arasındaki bağlantıları doğrudan montaj kabloları ile gerçekleştirdim (Şekil 7.10).





ŞEKİL 7.7: (solda) ve **ŞEKİL 7.8:** (sağda).





ŞEKİL 7.9: (solda) ve **ŞEKİL 7.10:** (sağda).

Model Araba Kontrol Yazılımı

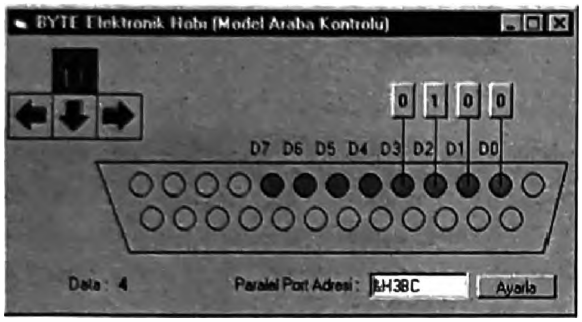
Programı, daha önce LED yakıp söndürmek için yazdığımız kodun üzerinde geliştirdim. Bizi paralel portun ilk 4 pini (0-3) ilgilendiriyor. Programımız, klavyenin yön tuşlarını kullanarak, gerekli sinyalleri porta gönderecek.

Örneğin, tekerleği sağa döndürmek istiyorsak, DATA 0 pinini "1" yani yüksek yapmalıyız. Bunun için paralel porta "00000001" değerini göndermeliyiz. Bu durumda ULN2003'ün 3 numaralı pini "1" yani yüksek olacaktır. Böylece 14 numaralı pin de düşük, yani toprak olacaktır. Bu toprağı da kumandadaki butonlara uyguluyoruz zaten.

TABLO 7.1: Paralel Porttan Gönderilecek Sinyaller

İşlem	Gönderilecek Değer	Ondalık Karşılığı
Sağa Dön	00000001	1
Sola Dön	00000010	2
İleri Git	00000100	4
Geri Git	00001000	8

Bir diğer olasılık da iki işlemin aynı anda olmasıdır. Örneğin, ileri giderken sağa dönmek gibi. Bu durumda iki butona birden basılmalıdır. Yani, ULN2003'e giden 2 pin aynı anda "1" olmalıdır. Sağ ileri için, "00000101" gibi. Bu değerin ondalık karşılığı, dikkat ederseniz "5". Yani yukarıdaki tabloda, ileri ve sağ değerlerinin toplamı. Eğer zıt işlemler aynı anda yapılmak istenirse, ilk giden sinyal dikkate alınıyor. Bu durum bizimle alakalı değil, çünkü kullanıcı kumanda üzerinde de hem ileri hem de geriye aynı anda basabilir. Bu nedenle bunun engelini koymak bize düşmüyor.



ŞEKİL 7.11: Model araba kontrol yazılımı.

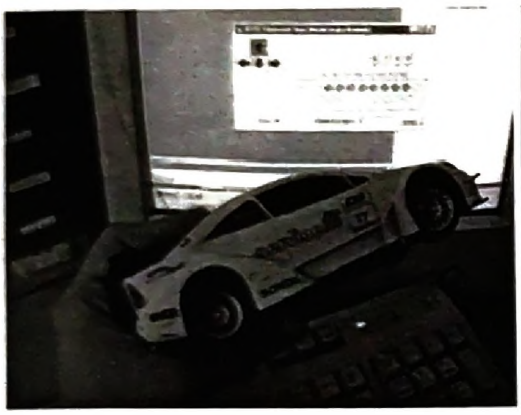
Programda aynı zamanda paralel portun pinlerinin ve klavyeden basılan tuşların durumunu izleyebiliyorsunuz (Şekil 7.12). Klavyeden hangi ok tuşuna basıldığını anlayabilmemiz için, kullandığımız Visual Basic formunun **KeyPreview** özelliğini açık tutuyoruz. Formun **KeyDown** ve **KeyUp** olaylarını kullanıyoruz. Her basılan tuş için bir değişkenimiz var. Örneğin sağ tuşuna basıldığında **data(0)** değişkeni "1" oluyor. İleri basıldığında ise, **data(2)** değişkeni "4" oluyor ve her seferinde bu dört değişkenin toplamı (**data(0)+data(1)+data(2)+data(3)**) paralel porta gönderiliyor. Böylece hangi tuşlara basılı ise o işlemi gerçekleştirecek veri devreye gönderilmiş oluyor.

PC ile model araba kontrolü – Visual Basic Kodu

```
Dim data(7) As Integer, top_data As Integer, adres As Integer
Private Sub Command2_Click(Index As Integer)
    If adres = 0 Then MsgBox "Lütfen PARALEL port adresini girip ""Ayarla"" 'ya
tıklayınız. Örnek:&h378", vbCritical + vbOKOnly, "Uyarı": Exit Sub
    Select Case Command2(Index).Caption
        Case "0"
            led_yak (Index)
            Command2(Index).Caption = "1"
        Case "1"
            led_sondur (Index)
            Command2(Index).Caption = "0"
    End Select
    lbl_data = top_data
End Sub
Private Sub Command3_Click()
    adres = Int(Val(txt_adres.Text))
End Sub
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
```

```
    Select Case KeyCode
        Case vbKeyUp
            led_yak (2)
            Command2(2).Caption = "1"
        Case vbKeyLeft
            led_yak (1)
            Command2(1).Caption = "1"
        Case vbKeyDown
            led_yak (3)
            Command2(3).Caption = "1"
        Case vbKeyRight
            led_yak (0)
            Command2(0).Caption = "1"
    End Select
    lbl_data = top_data
End Sub
Private Sub Form_KeyUp(KeyCode As Integer, Shift As Integer)
    Select Case KeyCode
        Case vbKeyUp
            led_sondur (2)
            Command2(2).Caption = "0"
        Case vbKeyLeft
            led_sondur (1)
            Command2(1).Caption = "0"
        Case vbKeyDown
            led_sondur (3)
            Command2(3).Caption = "0"
        Case vbKeyRight
            led_sondur (0)
            Command2(0).Caption = "0"
End Select
lbl_data = top_data
End Sub
Private Sub Form_Load()
    adres = Int(Val(txt_adres.Text))
    Out adres, 0
End Sub
Sub led_yak(pin_no As Integer)
    led(pin_no).FillColor = &HFF&
    top_data = 0
    Select Case pin_no
        Case 0
            data(0) = 1
        Case 1
            data(1) = 2
        Case 2
            data(2) = 4
        Case 3
            data(3) = 8
        Case 4
            data(4) = 16
        Case 5
            data(5) = 32
```

```
        Case 6
            data(6) = 64
        Case 7
            data(7) = 128
    End Select
    For i = 0 To 7
        top_data = top_data + data(i)
    Next i
    Out adres, top_data
End Sub
Sub led_sondur(pin_no As Integer)
    led(pin_no).FillColor = &HC0C0C0
    top_data = 0
    data(pin_no) = 0
    For i = 0 To 7
        top_data = top_data + data(i)
    Next i
    Out adres, top_data
End Sub
```





ŞEKİL 7.12: (solda) ve **ŞEKİL 7.13:** (sağda).

Yukarıdaki kod içerisindeki en önemli fonksiyon, **led_yak** fonksiyonudur. Bu fonksiyon, içerisine gönderdiğimiz pin numarasını yüksek yapmaya yarıyor. Ayrıca, aynı şekilde pini düşük yapan bir **led_sondur** fonksiyonumuz da var. **Form_KeyDown** olayında ise klavyeden basılan tuş kontrol ediliyor. Örneğin İLERİ tuşuna basılmış ise, **KeyCode=vbKeyUp** oluyor ve **led_yak(2)** fonksiyonu çağırılıyor. Sonucunda tekerlek dönmüş oluyor (Şekil 7.13).

Bu kod içerisindeki **adres** değişkeni, paralel portunuzun BASE adresinin değerini içermeli. Benimkisi &H3BC idi. Bu konulara yabancı iseniz, sitemizdeki temel paralel port uygulamalarını inceleyin. Ayrıca herhangi bir paralel port uygulamasına başlamadan önce, portun DATA pinlerini bir led ile deneyin ve sinyal alındığından emin olun.

SEKİZİNCİ BÖLÜM

8

SERİ İLETİŞİM

Seri Port

Bu bölüme kadar uygulamalarımızı hep paralel port üzerinden gerçekleştirdik. Bunun sebebi, paralel portun lojik veriler ile programlanmasının kolay oluşuydu. Bu bölümde, seri iletişimin temellerini ve iki bilgisayarın seri port aracılığı ile nasıl haberleşebileceğini öğreneceğiz. Aynı zamanda, bilgisayar dünyasında çok sık duyulan birçok terimin ne anlama geldiğini göreceğiz.

Seri veri ile paralel verinin iletim farkı ise temelde çok ufaktır. Paralel port üzerinde, bitler yani lojik 1 ya da 0 değerleri 8 tane ayrı kablo ile aynı anda iletilir. Seri port üzerinde ise bu lojik değerler tek bir kablo aracılığı ile peşi sıra iletilir. Seri portun kullanımı ve programlaması, paralel porta nazaran daha karışıktır, fakat bunun yanında seri iletişimin de avantajları yok değildir.

Seri Portun Avantajları

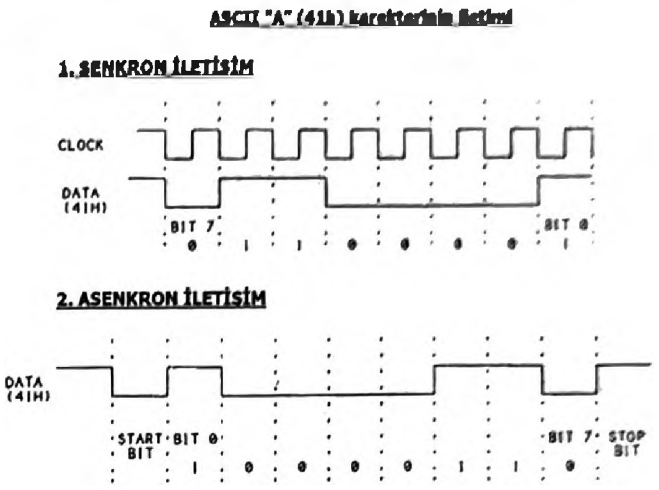
1. Seri kablolar, paralel kablolara göre daha uzun olur. Seri port, lojik değerleri –3 volt ile +25 volt arasında iletebilir. Paralel portta ise "0", 0 volt ile, "1" ise +5 Volt ile iletilir. Dolayısı ile, seri portun 50V maksimum voltaj değişim aralığına sahiptir. Paralel portta ise bu aralık 5 volttur. Bu nedenle, kabloda oluşan kayıp, seri portlarda paralel portlardaki gibi önemli değildir.
2. Seri iletişimde, paralel porta göre çok daha az tel kullanılır. Cihaz ile bilgisayar arasındaki 3 telli kablo seri iletişim için yeterlidir. 3 telli bir kablo, 25 telli bir kabloya göre daha ucuz olacaktır.
3. Daha önce bahsettiğimiz, seri haberleşmeyi kullanan kızıl ötesi (infra red) cihazlar, veriyi ancak seri olarak iletebilirler. Böyle bir haberleşmenin paralel olması imkansızdır.

Seri haberleşmede, gönderici kısmında 8-bit veri, paralelden seriye çevrilir ve daha sonra tek bir hattan karşıya gönderilir. Alıcı, seri veriyi paralele çevirerek 8-bit veriyi oluşturur. Veri telefon hattı ile iletilecekse, bu lojik değerler ses sinyallerine dönüştürülür. Bu tür haberleşmede, göndericideki lojik değerler modülatör ile ses sinyaline çevrilir. Ses sinyali karşı tarafa ulaştığında, demodülatör ile tekrar lojik değerlere dönüştürülür. Modülatör ve demodülatör işlemlerini hepimizin bilgisayarında bulunan ve yakından tanıdığı modem (Modulator/Demodulator) adlı cihazlar gerçekleştirir.

Seri Port ile Veri Aktarımı

Bir linkteki veri akışının kontrolü için, gerekli sinyallerden biri saat (clock) sinyalidir. Hem gönderici, hem de alıcı cihazda, bir bitin ne zaman gönderileceğine veya alınacağına karar verilirken bir saat sinyali kullanılır. Veri gönderen ve alan uçların belli kurallar çerçevesinde haberleşmesi gerekir. Verinin nasıl paketleneceği, bir karakterdeki bit sayısını, verinin ne zaman başlayıp biteceği gibi bilgileri bu kurallar belirler. Bu kurallar çerçevesine, protokol adı verilir.

Eğer veri sadece bir yönde aktarılıyorsa half duplex, aynı anda her iki yönde aktarılıyorsa full duplex olarak adlandırılır. İki çeşit seri iletim formatı vardır. Senkron ve asenkron. Her biri saatleri farklı şekilde kullanır.



ŞEKİL 8.1: Bit iletimi.

Senkron gönderimde, her cihaz, kendisi ya da dışarıdan bir cihaz tarafından üretilen aynı saat sinyali darbelerini kullanır. Saatin frekansı sabit ya da düzensiz aralıklarda değişken de olabilir. İletilen her bit, bir saat darbesi geçişinden, yani şekildeki yükselen veya alçalan kenardan sonraki belirli bir zamanda geçerli olur. Senkron formatlar, iletimi başlatırken ya da bitirirken, çok çeşitli formatlar kullanırlar. Bunlara start-stop bitleri denir. Fakat uzun mesafeli linklerde senkron format uygun değildir. Çünkü bahsettiğimiz saat sinyalinin iletimi, parazit nedeniyle ek bir hat gerektirebilir. Bu durumda, asenkron gönderim kullanılır.

Asenkron iletişimlerde, linkte saat hattı bulunmaz. Her uç kendi sinyalini sunmaktadır. Bu iletişimde de, uçların saat frekansında anlaşmaları gerekir. Bu nedenle iletilen her byte'ta saatleri eşlemek üzere bir start biti ve iletimin bittiğini bildirmek üzere bir stop biti bulunur.

Seri iletişimde veri aktarım hızı, saniyedeki bit sayısı (bps – bits per second) olarak belirtilir. Veri aktarım hızını belirlemede yaygın olarak kullanılan diğer terim ise baud rate'tir.

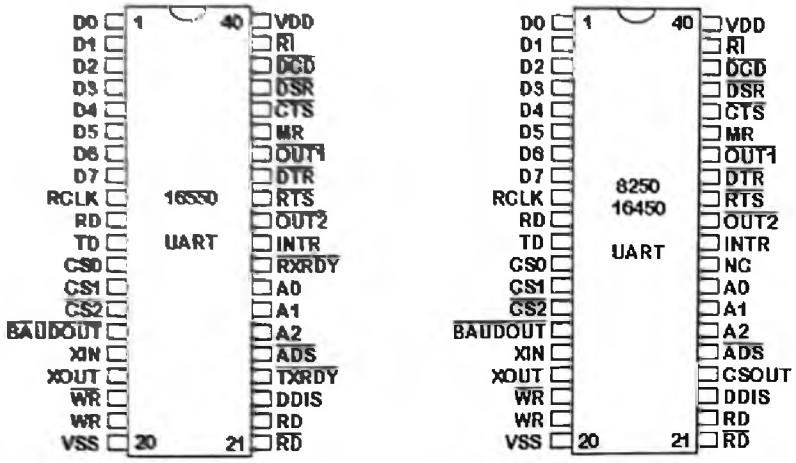
UART Tümdevresi

Bölümün başında, iletişim seri şekilde olsa bile, verinin işlenebilmesi için paralele çevrilmesi gerektiğinden bahsetmiştik. UART tümdevresi, seri ve paralel veriyi birbirine çevirmeye yarar. CPU'nun sistem bus'ında verinin okunabilmesi için, UART ile paralel dönüşüm yapılır. UART hem full duplex hem de half duplex veri iletişimini destekler. Veri hatları yanında UART, RS-232 el sıkışma sinyali ile RTS, CTS, DTR gibi kontrol sinyallerini de destekler. Orijinal PC'lerde, seri porta kumanda eden UART, 57,600 bps maksimum hızı olan bir 8250 idi. Günümüzdeki PC'lerde ise 115,200 bps hıza imkan veren 16550 ya da eşleniği olan daha gelişkin devreler bulunur.

Kesmelerin (IRQ) Kullanımı

Kesme, basit olarak, CPU'ya acil olarak yapılması gereken işleri bildirmeye yarayan bir sinyaldir. Normal bir PC, 16 kesme hattını destekler. Donanım kesmelerine başvuran bir programda kesme servis rutinleri (ISR) olmak durumundadır. Bu rutinler, bir kesme meydana geldiğinde

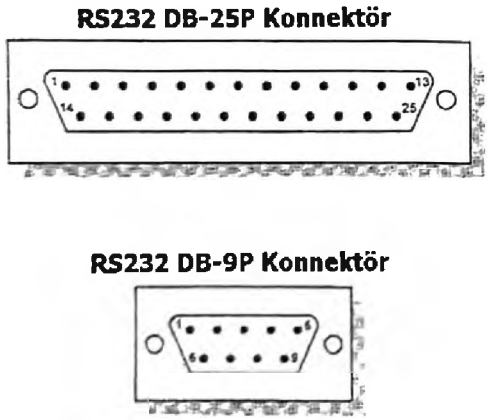
gereken fonksiyonu yerine getirirler. Cihaz, kesme yapmak için, IRQ hattını bir darbe ile uyarır. Sistemin kesme kontrolü bu isteği algılar ve durumu CPU'ya bildirir. CPU o anda yaptığı işi durdurarak ISR'yi devreye sokar. ISR'in devreye girişi ile birlikte CPU kaldığı yerden devam etmek üzere işine döner.



ŞEKİL 8.2: UART tümdevresi.

RS232 Standardı

Değişik üreticiler tarafından yapılmış veri haberleşme cihazlarının uyumluluğunu sağlamak amacıyla, EIA (Electronics Industries Association) tarafından 1960 yılında, RS232 olarak adlandırılan standart belirlenmiştir. Günümüzde de RS232 en yaygın kullanılan seri I/O arabirim standardıdır.



ŞEKİL 8.3: RS232 standartları.

RS232 için, ilk olarak DB-25 ile erişilen toplam 25 uç tanımlanmıştır. Modern bilgisayarlarda bu 25 uca gerek olmadığı için IBM, DB-9 seri I/O standardını geliştirmiştir.

TABLO 8.1: 9 ve 25 Uçlu RS232 Sinyalleri

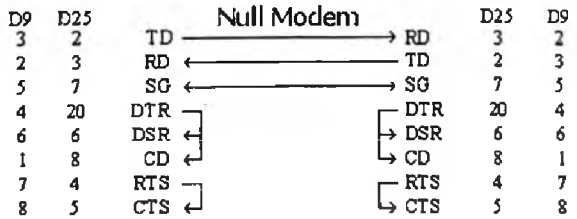
Uç No (9)	Uç No(25)	Sinyal Adı
1	8	Data Carrier Detect (DCD)
2	3	Receive Data (RxD)
3	2	Transmit Data (TxD)
4	20	Data Terminal Ready (DTR)
5	7	Signal Ground (GND)
6	6	Data Set Ready (DSR)
7	4	Request to Send (RTS)
8	5	Clear to Send (CTS)
9	22	Ring Indicator (RI)

PC ile Modemin Haberleşmesi

DTR ve DSR sinyalleri, PC ve modem tarafından hazır olma durumlarını belirlemede kullanılır. RTS ve CTS ile veri akışı kontrol edilir. PC, veri göndermek istediğinizde, RTS hattını aktif yapmakta, buna karşılık modem veri kabul etmeye hazırsa, yani veri için yer varsa, CTS sinyalini gönderir. Eğer yer yoksa modem CTS'yi aktif yapmaz. Bu durumda PC'nin tekrar aynı şekilde denemesi gerekir.

Boş (Null) Modem Bağlantısı

Boş modem bağlantısı iki DTE cihazını bağlamakta kullanılır. Bu yöntem, Zmodem veya Xmodem protokolü gibi bir protokol kullanarak, ağ haberleşmeleri veya bilgisayarlar arası dosya aktarımında kullanmak oldukça ucuz bir yoldur. Şekilde de görüldüğü gibi, boş modem bağlantısında sadece 3 hat gereklidir. Bunlar TD, RD ve GND uçlarıdır. Bu bağlantı şekli ile amaçlanan, bilgisayarın diğer bir bilgisayar ile değil, sanki bir modem ile haberleşiyormuş gibi düşünmesini sağlamaktır. Kısacası, modülasyon ve demodülasyon işlemi yoktur. Dolayısı ile modeme de gerek yoktur.

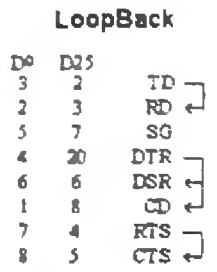


ŞEKİL 8.4: Null modem bağlantısı.

Şekilde TD ve RD veri hatları karşılıklı çapraz şekilde bağlanmış ve topraklar ortaklanmıştır. Her iki portta da DTR sinyalleri, DSR ve DCD uçlarına bağlanmıştır. DTR sinyali aktif yapıldığında, DSR ve DCD hemen aktif olur. Bu sayede, bilgisayar bağlı olduğu sanal bir modemin hazır olduğunu ve bu modemin DCD sinyali aldığını düşünür. Her iki bilgisayar aynı hızda

haberleştiği için, akış kontrolüne gerek yoktur. Bu yüzden RTS ve CTS sinyalleri her iki portta birbirine bağlanmıştır. Bir bilgisayar verisini göndermek istediğinde, RTS sinyalini aktif yapar. Bu sinyal CTS'ye geri bağlı olduğundan, gönderme için tamam cevabını hemen alır ve bu işlemi yerine getirir. RI sinyali ise her iki portta da bağlanmamıştır. Bu sinyal, telefon hattındaki çalma işlemini bilgisayara bildirir. Telefon hattına bağlı bir modem olmadığı için, bu hat bağlantısız bırakılmıştır.

Geri Besleme (Loop-Back) Bağlantısı



ŞEKİL 8.5: Loop-back bağlantısı.

Şekilde gördüğünüz geri besleme bağlantısı, genelde seri RS232 haberleşme programları yazarken, geliştirme aşamasındaki testlerde kullanılır. Eğer evinizde ikinci bir PC yoksa, geri besleme bağlantısı ile seri port uygulaması geliştirebilirsiniz. Bu bağlantıda, alma (RD) ve gönderme (TD) hatları birbirlerine bağlanmıştır. Seri olarak dışarıya gönderilen veri, yine aynı porttan içeri alınır. Bu durumda bilgisayar kendi kendine bağlantı kurmuş olur.

DTE/DCE Hızları İle Veri Akış Kontrolü

DTE bir bilgisayar, DCE ise bir modem olarak tanıtılabilir. Bu ikisi arasındaki hız, bilgisayar ile modem arasındaki hızdır. DCE-DCE arasındaki hız ise modemler arasındaki hızdır. Günümüzde 56K modemler yaygın olarak kullanılır. Yukarıda bahsettiğimiz yaygın olarak kullanılan UART 16550A 115, 200 BPS maksimum DTE-DCE bağlantısını destekler. Bunun yanında modemler, veri sıkıştırma ve açma fonksiyonlarına sahiptir. Eğer bilgisayar ile modem yani DTE-DCE arasındaki hız, modemler arasındaki hızdan çok fazla ise, PC'nin gönderdiği veri, modemin buffer alanını kısa zamanda doldurduğundan bir veri akış kontrolü gerekir. Akış kontrolü için yazılım ve donanım olmak üzere iki yöntem vardır.

Yazılım akış (Software Flow) kontrolünde, bazı yerlerde Xon/Xoff olarak belirtilen iki karakter kullanılır. Genelde Xon ASCII17, Xoff ise ASCII19'dur. Modemin buffer'ı dolduğunda, modem Xoff karakteri göndererek, bilgisayara daha fazla veri göndermemesini bildirir. Buffer boşaldığında ise Xon sinyali gönderilir.

Donanım akışında ise (Hardware Flow) bilgisayar veri göndermek istediğinde RTS (Request to Send) hattını aktif yapar. Eğer modemde veri için yer var ise CTS (Clear to Send) hattını aktif yapar ve bilgisayar veri göndermeye başlar. Eğer modemde yer yoksa CTS sinyali gönderilmez. Dolayısı ile bu şekilde yapılan bir akış kontrolü iletişim hızını yavaşlatmaz.

Bilgisayarınız, COM olarak belirtilen 4 taneye kadar bahsettiğimiz bu seri portlara sahiptir. Bu portlar PC açıldığında, power on self test (POST) esnasında 4 COM portundan her biri için, UART tümdevresi test edilir. Eğer COM port bulunur ise, I/O port, adresleri hafızada 0040:0000 ile 0040:0007 arasına yazılır. Bu portlara, BIOS tabanlı INT 14H kesme komutu kullanılarak erişmek de mümkündür. INT 14H komutunun çeşitli seçenekleri AH saklayıcısı ile belirlenir. Bu komut ile iki PC arasında karakter alışverişi çok kolaydır. Fakat şimdilik işin bu kadar derinine inemiyoruz. Çünkü işin içerisine assembly programlama da giriyor. Bahsettiğimiz INT 14H kesme komutu, assembly programlama komutudur.

Eğer seri iletişimin temelini bilirseniz, Visual Basic altındaki MSComm kontrol objesini kullanmak çocuk oyuncağı gelecektir. Seri portu programlamanın Windows altındaki en kolay yolu MSComm kullanmaktır. Yukarıda bahsettiğimiz sinyalleri bu kontrol objesi ile kontrol etmek oldukça kolaydır. Aşağıdaki örnek Visual Basic programı RS232 bağlantısı ile bilgisayarınızdaki bir dosyayı diğerine aktarmanızı sağlar. Eğer biraz Visual Basic bilginiz varsa, aşağıdaki örnek kod ile, çalışma prensibini anlamanız kolay olacaktır.

Visual Basic MSCOMM ile dosya transferi

```
'Dosya gönderme kodu.
   Dim FileName As String: FileName = "C:\BinFiles\Test.Exe"
   Dim Offset As Long
   Dim FileData As Byte

   Open FileName For Binary Access Read As #1
      For Offset = 1 To FileLen(FileName)
         Get #1, Offset, FileData
         MSComm1.Output = Chr$(FileData)
      Next Offset
   Close #1

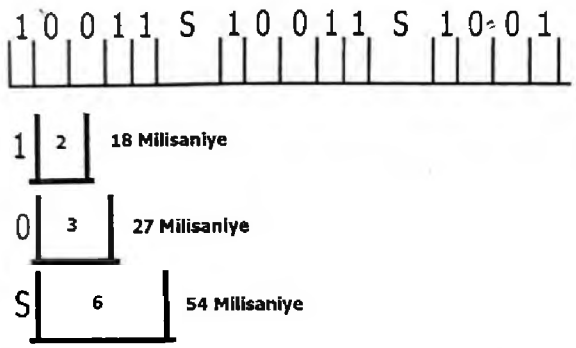
'Receiver Code.
   Private Sub MSComm1_OnComm()
      Dim TmpStr As String
      Dim StrLen As Long, I As Long
      Dim FileData As Byte

      If MSComm1.CommEvent = comEvReceive Then
         While MSComm1.InBufferCount > 0
            TmpStr = MSComm1.Input
            StrLen = Len(TmpStr)
            For I = 1 To StrLen
               FileData = CByte(Asc(Mid(TmpStr, I, 1)))
               ByteCount = ByteCount + 1
               Put #1, ByteCount, FileData
            Next I
            TmpStr = ""
         Wend
      End If
   End Sub
```

Kızıl Ötesi İletişim

Kızıl ötesi alıcı-verici devreleri, verileri seri olarak iletirler. Bilgisayar üzerinde gerçek bir kızıl ötesi veri girişi yapmak istiyorsanız, seri port kullanmanız gerekmektedir. Seri sinyallerin, paralel sinyallerden farkı, gönderilen "1" ve "0" değerlerinin yan yana 8'er bit şeklinde değil, peş peşe tek ve sürekli bir sinyal şeklinde gönderilmesidir. Bu sinyaller tabii ki belli zaman aralıklarında gönderilmektedir. Bu nedenle bu tarz vericilerde bir zamanlayıcı ya da diğer bir adıyla yüksek frekansta çalışabilen bir sayıcı kullanılmaktadır. En sık kullanılan zamanlayıcı entegre devresi 555 adında bir entegredir.

Verici (transmitter) üzerinde veriler bahsettiğimiz bu kızıl ötesi led ya da ledler sayesinde 5 bit koda çevrilir. Bu bitler Şekil 8.6'daki gibi kodlanır.



ŞEKİL 8.6: Kızıl ötesi sinyallerin kodlanması.

Bu şekildeki seri iletişimde bugüne kadar öğrendiğimiz "1" ve "0" lojik sinyallerinin dışında, bir de "S" sinyali vardır. "S" sinyali sınırlayıcı olarak çalışır ve bütün 5 bitin gönderildiğini ifade etmek için kullanılır. Peş peşe gönderilen bu sinyallerin hangisinin "1" hangisinin "0" ya da "S" sinyali olduğu, ışığın milisaniyede göndereceği yanıp sönme zaman aralıkları ile bağlantılıdır. "1", "0" ve "S" sinyalleri arasında 2:3:6 ilişkisi vardır. Tipik olarak "1" için 18 milisaniye, "0" için 27 milisaniye ve "S" için ise "54" milisaniyelik kızıl ötesi ışık uygulanmaktadır. Bu şekilde kodlanmış olan sinyaller siz kumandanın tuşuna bastığınız sürece gönderilir. Alıcının (receiver), gönderilen darbenin "1", "0" ya da "S" olup olmadığını anlamak için, peşi sıra gelen bu sinyallerin zaman aralıklarını bulması gerekmektedir. Bu amaçla sabit ve yüksek frekansta çalışabilen bir sayıcı kullanılmaktadır. Bilgisayarınızın RS232 standardındaki seri portu dahili sayıcı içermekte ve RX pini sayesinde bu sinyalleri algılayabilmektedir. "S" sinyali alındığında 5-bit kodun tamamı okunabilmiş demektir. Birçok cihazda bu sinyal grubu alındıktan sonra bir önceki sinyal grubu ile karşılaştırır. Bu şekilde yansımalara ve sahte sinyallere karşı bir güvenlik elde edilir. Kullanılan alıcı entegre devrelerinin sayısal çıkışları genelde kanal değiştirme, ses kesme gibi işlemlerde, analog çıkışları ise ses ve benzeri ince ayarlamalar için kullanılır.

DOKUZUNCU BÖLUM

9

TEMEL MANTIK KAPILARI

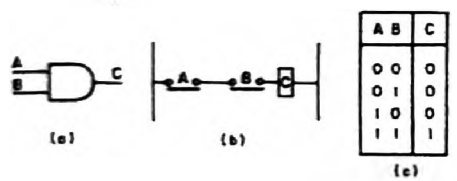
Mantık Kapıları

Bu bölümde, şimdiye kadar incelediğimiz lojik değerleri, mantık kapıları ile kontrol etmeyi öğreneceğiz. Daha öncede belirttiğimiz gibi, pek çok kontrol yapısı sadece iki durumdan birisi olabilen kararlara dayanır. Bu durumlar, "1" ve "0" sinyalleridir. Bu sinyallerinin durumlarını kontrol etmek için sık kullanılan yöntemlerden bir tanesi de TTL entegre devreleridir. Yaptığımız devrelere her zaman bir bilgisayar ekleyemeyiz. Alınacak çok basit kararlar için mantık kapıları kullanabiliriz. Çok soyut bir kavram gibi görünse de, bu mantık kapılarını TTL entegresi olarak elinizde tutmanız ve dokunmanız mümkündür. Örneğin, bir asansör devresi, garaj kapısı kontrolü, su seviyesi kontrolü ve benzeri uygulamaları, bu mantık kapıları ile geliştirmek mümkündür.

Komplike gibi görülen devreler bu mantık kapıları ile basit şekilde gerçekleştirilebilir. Bir TTL devresi kurarken dikkat etmeniz gereken en önemli nokta, mümkün olduğu kadar az kapı ile, mümkün olduğu kadar çok iş yaptırabilmek ve devrenin verimini arttırabilmektir. Bunun için geliştirilmiş birçok yöntem vardır. Bu yöntemlerden önce, temel mantık kapılarına şöyle bir göz atalım.

AND Kapısı

En çok kullanılan kapılardan bir tanesi de AND kapısıdır. Bu kapının çıkışının "1" olması için bütün girişlerinin de "1" olması gerekmektedir. Örneğin, aşağıdaki C çıkışı sadece ve sadece hem A hem de B girişleri "1" olduğu zaman "1" olur. Diğer bütün durumlarda sonuç "0" dır. Doğruluk tablosu aşağıdaki gibidir.

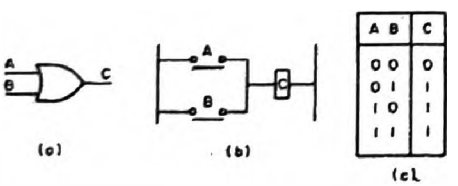


AND kapısı a) sembol b) röle eşdeğeri c) doğruluk tablosu

ŞEKİL 9.1: AND kapısı.

OR Kapısı

İkinci sık kullanılan kapı ise OR kapısıdır. Bu kapının "1" çıktısını vermesi için, girişlerinden sadece bir tanesinin "1" olması yeterlidir. C çıkışının "1" olması için A ya da B girişlerinden herhangi bir tanesinin "1" olması gereklidir. Eğer tüm girişler "0" ise çıkış da "0" olacaktır.

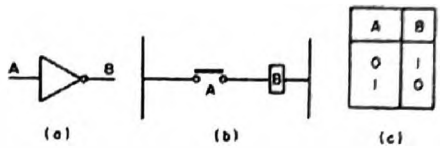


OR kapısı a) sembol b) röle eşdeğeri c) doğruluk tablosu

ŞEKİL 9.2: OR kapısı.

Ters Çevirici (Inverter):

Ters çevirici, adından da belli olduğu gibi, giren sinyalin tersinin çıkışını sağlar. Örneğin, A girişine "1" girerse, B çıkışı "0" olacaktır.

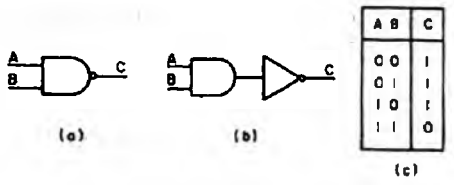


Dönüştürücü a) sembol b) röle eşdeğeri c) doğruluk tablosu

ŞEKİL 9.3: Ters çevirici.

NAND Kapısı

NOT AND kapısı, AND kapısının değilidir. Yani AND kapısının sonucun terslenmiş halini elde etmek için kullanılır. Yani AND kapısına eklenmiş bir Inverter'dan oluşur.

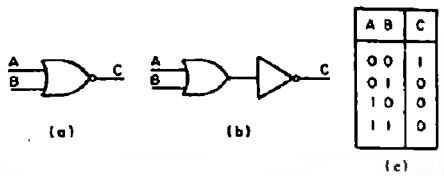


NAND kapısı a) sembol b) eşdeğer mantık devresi c) doğruluk tablosu

ŞEKİL 9.4: NAND kapısı.

NOR Kapısı:

NOT OR kapısı, OR kapısının sonuna eklenmiş bir Inverter'dan oluşur.



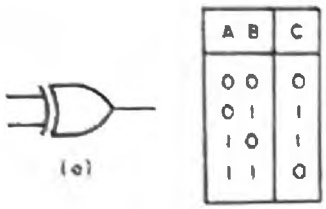
NOR kapısı a) sembol b) eşdeğer mantık devresi c) doğruluk tablosu

ŞEKİL 9.5: NOR kapısı.

XOR Kapısı:

Exclusive OR kapısı çok sık kullanılmaz. İki girişin aynı olup olmadığını kontrol eder. Eğer girişler farklı ise sonuç "1" olacaktır. Aynı ise sonuç "0" dır.

Şimdi, bu mantık kapıları ile nasıl bir devre oluşturulacağına gelelim. Uygulama olarak basit bir devre gerçekleştirelim. Örneğin, elimizde 3 adet buton olsun ve bu butonlardan herhangi 2 tanesine aynı anda bastığımızda bir ledi yakalım. Ama üçüne aynı anda bastığımızda yanmasın. Hiç birine basmadığımız zaman da yanmasın. Basit bir uygulamadan başka hiçbir işe yaramayacak bir devre gibi gözükse de, üşenmeden gerçekleştirmenizi öneriyorum.



XOR kapısı a) sembol b) doğruluk tablosu

ŞEKİL 9.6: XOR kapısı.

İlk bakışta bu kapılardan bir devre çıkarmak çok zor bir iş gibi gözüküyor. Fakat iş esasında o kadar zor değil. 19. yüzyılda Cambridge matematikçilerinden George Boole, mantık ifadelerinin gösterilmesi ve işlem yapılabilmesi için kullanılan bir cebirsel işaretleme sistemi geliştirdi. Bu cebir, bileşik mantık devrelerinin tasarımında da kullanıldı. Bu cebir sisteminde AND fonksiyonu nokta (.) ile, OR fonksiyonu artı (+) ile gösterilir. NOT (Değil) fonksiyonu ise, sinyalin üzerine bir çubuk çizilerek gösterilir. Kullanılacak kapıların minimum düzeye indirilmesinde kullanılan bir diğer yöntem de Karnaugh haritalarıdır. Bu haritalar, doğruluk tablolarının iki boyutlu gösteridir. Fakat şimdilik işin çok matematiksel derinliğine inmeyelim. Daha ileride bu harita yöntemini kullanarak devre tasarımı gerçekleştireceğiz.

Şimdi yukarıda bahsettiğimiz bu devrenin tüm giriş-çıkış olasılıklarını göz önüne getirelim. A, B ve C bizim girişlerimiz olsun. Bu durumda, butonların herhangi iki tanesinin "1" olduğu durumlarda elde edeceğimiz çıkış "1" olsun. Aşağıdaki tabloda hangi durumda hangi çıkışı almak istediğimiz i belirleyelim.

TABLO 9.1: Durumlara Göre Alınan Çıkışlar

DURUM	A	B	C	Çıkış
1	0	0	0	0
2	0	0	1	0
3	0	1	0	0
4	0	1	1	1
5	1	0	0	0
6	1	0	1	1
7	1	1	0	1
8	1	1	1	0

Görüldüğü gibi bizi sadece 4, 6 ve 7 numaralı durumlarla ilgileniyoruz. Şimdi bu durumları Boole cebiri ile gösterelim.

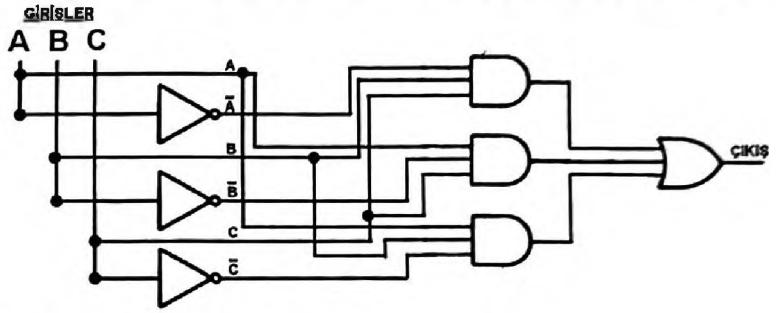
Örnek Devrenin BOOLE Cebiri ile gösterilişi

$$Z = \underbrace{(\bar{A}.B.C)}_{4.DURUM} + \underbrace{(A.\bar{B}.C)}_{6.DURUM} + \underbrace{(A.B.\bar{C})}_{7.DURUM}$$

ŞEKİL 9.7: Durumların Boole cebiri ile gösterilişi.

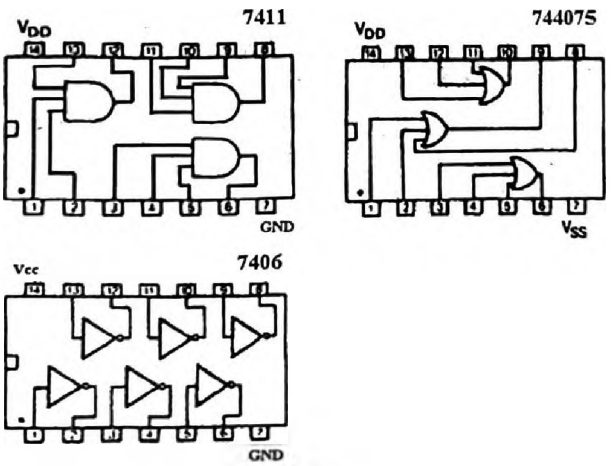
4. durumda sadece B ve C butonlarına basıldığı varsayılıyor. 6. durumda ise A ve C butonlarına basıldığı varsayılıyor. 7 numaralı durumda ise A ve B butonlarına aynı anda basıldığı varsayılıyor. İşte bu sonuçların herhangi bir tanesinin doğru olması durumunda ledimiz yanacak. Bu nedenle fonksiyonu yazarken bu üç durumu (+) ile yan yana yazarak belirtebiliyoruz.

ÖRNEK DEVRENİN MANTIK KAPILARI İLE GÖSTERİLİŞİ



ŞEKİL 9.8: Fonksiyonun mantık şeması.

Devremizi kurarken, AND kapısı için 1 adet içerisinde üç tane üç girişli AND kapısı içeren 7411 , OR kapısı için, 1 adet içerisinde üç adet üç girişli OR kapısı içeren 744075, Inverter (tersleyici) için ise, içerisinde altı adet inverter olan 7406 kullanabilirsiniz. Bu entegrelerin iç yapıları aşağıdaki gibidir. Yukarıdaki fonksiyon şemasına göre bu entegreleri kullanarak lojik voltajımızı mantık kapılarından geçirebilirsiniz. Sonuçta elde ettiğiniz çıkış değeri ile, herhangi bir ledi, ya da transistörlü röle devresini tetiklemeniz mümkün olabilir. Girişteki lojik voltajları ise paralel portunuzun DATA pinlerinden elde etmeniz mümkündür.



ŞEKİL 9.9: TTL entegreleri.

ONUNCU BÖLÜM

10

INTERNET ÜZERİNDEN EV OTOMASYONU

Ev Otomasyonu

Evimize uzaktaki bir bilgisayardan bağlanıp, evdeki elektronik cihazları denetlemek oldukça işe yarar olurdu. Örneğin, sıcak ve yorucu bir günün sonrasında eve döndüğünüzde şöyle sıcak bir duş almak istiyorsunuz. Fakat eve döndüğünüzde, elektrikli şofbeni açmadığınızı fark ediyorsunuz. Mecburen açıyorsunuz, fakat su ısınana kadar uyuya kalıyorsunuz ve sabah kalktığınızda 80 dereceye ulaşmış olan şofbeni kapatarak tekrar işe yetişmeye çalışıyorsunuz.

Halbuki, işyerinizden çıkmadan önce, evinizdeki bir sunucuya Internet üzerinden bağlansanız ve şofbeni açması için bir komut gönderseniz, oldukça kullanışlı olurdu herhalde. Böyle bir sistemi kurduğunuzda, her türlü cihazı kontrol etmeniz mümkün olur. Veri de toplayabilirsiniz. Örneğin, evin sıcaklığını okutarak bir analog-dijital dönüştürücü sayesinde bunu sunucuya aktarabilir ve uzaktan bu sıcaklığa göre kombiyi yakabilirsiniz. Ya da evinizdeki bütün ışıkları bu sunucuya bağlarsınız ve uzaktan evinizi hırsız moduna alırsınız. Bu mod, değişik saatlerde rasgele ışıklarınızı yakıp söndürür ya da evdeki bir radyoyu açabilir.

Biraz daha ilerletelim. Örneğin, hırsız modunda iken, evinizin değişik yerlerine yerleştireceğiniz hareket sensörlerinden alınacak veriye göre evde hareket varsa cep telefonunuza SMS çektirtebilirsiniz. Hızımızı almışken biraz daha zorlayalım. Mesela, evinizdeki kapının kilidini elektronik alırsınız. Kapınızın girişine bir parmak izi tarayıcısı yerleştirirsiniz. Sunucu tarafındaki program bu parmak izini tanıyarak kapıyı açar. Bu durumda evde kimin olduğunu uzaktan görebilirsiniz. Biraz da mekanik bilginiz ve gerekli malzemeleriniz varsa, pencereleri ve perdeleri açıp kapayabilirsiniz, çiçeklerinizi bile sulayabilirsiniz. Kısacası, uzaktan her türlü elektronik cihazı kontrol edebilirsiniz. Elektronik olmayanları ise, mekanik sayesinde elektronikleştirebilirsiniz.

Internet Üzerinden Ev Otomasyonu Uygulaması

Uygulamamız şimdilik sadece uzaktan veri gönderme amaçlı olacak. Temelde öğreneceğimiz, sunucu üzerindeki röle devresinin nasıl çalıştığı ve uzaktan nasıl kontrol edilebileceği olacak. Bu röle devrelerini evinizin neresinde kullanacağınız size kalıyor. Sunucu dediğimiz, sıradan bir PC olabilir. Tek gereken, üzerinde bir paralel port bulunmasıdır.

Bir paralel porttan kolaylıkla 8 adet elektronik cihazı tetikleyebilirsiniz. Bunun için bugüne kadar defalarca gerçekleştirdiğimiz paralel porttan röle kontrol devresine ihtiyacımız var. Uygulamamızın amacı, sunucuya TCP/IP protokolü ile bağlanıp, sunucu üzerinde bulunan paralel port üzerindeki 8 bitlik çıkıştan istediğimizi yüksek yapmak. Bunu da izlemenin en güzel yolu, yine paralel port üzerine bağlı ledler kullanmak.

Sıra, evdeki sunucuya nasıl bağlanacağımıza geldi. Bunu teorik olarak eve çekeceğiniz bir seri kablo ya da ağ kablosu ile yapmanız mümkün. Tabii, pratikte böyle bir kabloyu sokak sokak çekemeyeceğimiz ortada. Sunucuya uzaktan veri gönderebilmek için, her iki bilgisayar da aynı ağ üzerinde olmalıdır. Eğer her iki bilgisayar da Internet'e bağlı ise, bu bilgisayarlar bir şekilde aynı ağ üzerinde demektir. Kısacası, sunucu olarak bahsettiğimiz evdeki bilgisayarın sürekli Internet'e bağlı olması gerekmektedir. Telefon bağlantısı ile bu çok sağlıklı olmayabilir. Böyle bir uygulama için, kablolu Internet daha kullanışlı olacaktır.

Bağlantı için bize gerekli olan sabit bir IP adresidir. Herhangi bir servis sağlayıcı tarafından Internet'e bağlandığınızda bilgisayarınıza bir IP adresi atanacaktır. Telefonla bağlandığınızda,

bu adres her seferinde değişecektir. Kablolu İnternet'te ise bu değişim her seferinde olmamaktadır. Eğer ISS (İnternet Servisi Sağlayıcı) firmaları size statik bir IP adresi verselerdi, kendi Web sunucunuzu evinizde tutabilirdiniz. Bunun önüne geçebilmek için ISS'ler tarafından dinamik IP adresi verilmektedir. Statik IP adreslerinin ücretleri ise daha fazladır.

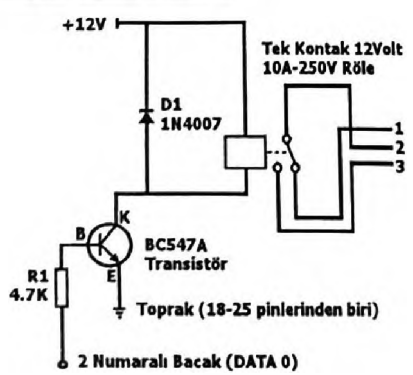
Statik IP adresiniz olmadığı sürece, evinizde bulunan sunucunun IP adresinin ne olduğunu ya da değişip değişmediğini bilemeyiz. Fakat uzaktan bağlanabilmek için bu IP adresine ihtiyacımız var. Bunun önüne geçmek için, sunucu tarafında, belirli aralıklarla IP adresini internet üzerindeki sabit bir alana yazan bir program çalıştırabilirsiniz. Bunun için ise FTP ile bağlanabileceğiniz ve dosya gönderebileceğiniz bir Web alanına, IP adresini bir metin dosyasına yazarak, bu dosyayı periyodik olarak gönderebilirsiniz. Daha sonra bu Web alanından, sunucunuza bağlanmadan önce, sunucunuzun son aldığı IP adresini öğrenerek bu IP adresine bağlanabilirsiniz. Fakat bu işlemler biraz programlama bilgisi gerektirecektir.

Uygulamayı denemek için tek bir bilgisayar da işinizi görecektir. Aynı bilgisayarı hem sunucu, hem istemci olarak kullanabilirsiniz. Uygulamayı uzaktan gerçekleştirmeden önce, kendi kendine çalıştığından emin olmalısınız.

Paralel port ile röle kontrolünü Bölüm 2'de incelemiştik. Bu bölümde de aynı röle devresini kullanıyoruz. Yinede röle devresinin nasıl çalışacağını kısaca hatırlayalım. Bilgisayarın paralel portu üzerinde 8 adet DATA pini vardır. Bu pinler, porta gönderdiğimiz veriye göre yüksek ya da düşük olacaktır. Örneğin DATA portuna "115" değerini gönderelim. Bu durumda, bu sayıya karşılık gelen ikilik değer "1110011" şeklinde 8 bitlik DATA portuna yerleşecektir. Bu da "1" olan pinlerin +5 Volt, yani yüksek, "0" olan pinlerinde düşük, yani 0 Volt olduğunu gösteriyor.

"1" olan pinlerden elde edilen, bu +5 volta, lojik voltaj denir. Bu akım ile, hiçbir cihazı çalıştıramazsınız. Fakat yüksek voltajlı bir cihazı çalıştıracak bir röle devresini tetikleyebilirsiniz. Röle devresini bağlamadan önce, ledlerle, portunuzun doğru çalışıp çalışmadığını test etmelisiniz. Örnek programda bunu yüksek yapmak istediğiniz pinleri gösteren butonlara basarak deneyebilirsiniz. Devrenin şemasını Şekil 10.1'de görebilirsiniz. Röleye, istediğiniz herhangi bir elektronik cihazı bağlayabilirsiniz. Hangi DATA pinini yüksek yaparsanız, o pine bağlı röle devresi, gereken cihazı çalıştıracaktır.

12 Volt Röle Devresi

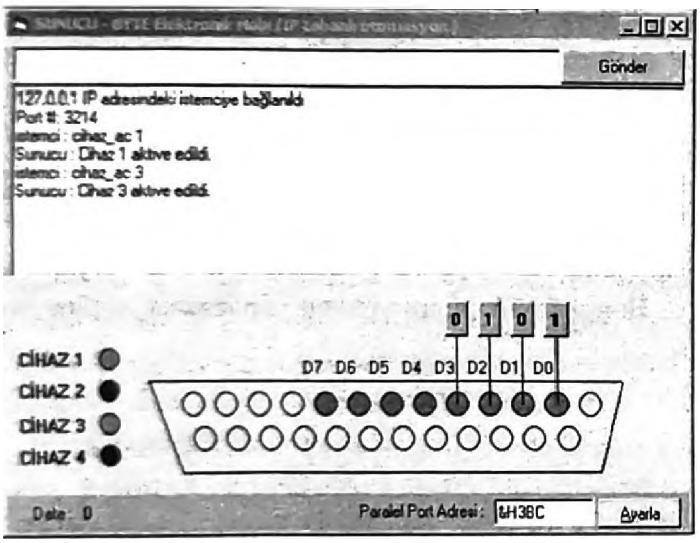


ŞEKİL 10.1: Röle kontrol devresi.

Şimdi, uzaktan IP adresi ile, bu DATA pinlerini nasıl uzaktan yüksek yapabileceğimize bakalım. Nasıl DATA portunun çalıştığını ledlerle denediysek, TCP/IP bağlantısının da çalıştığından önce emin olmalıyız. Bunun için, sunucu ve istemci tarafında çalışacak basit bir chat programı yazacağız. Bağlantı kurulduktan sonra, birbirlerine mesaj gönderebilecekler. Bu programın çalıştığından emin olduktan sonra, göndereceğimiz mesajları, sunucu tarafının komut olarak değerlendirmesini sağlayacağız. Ardından sunucu, bu komut çerçevesinde gerekli DATA pinlerini yüksek ya da düşük yapacak. Dolayısı ile, bu DATA pinlerine bağlı röleler aktif hale geçecek ve istenilen cihazlar çalışacaktır.

TCP/IP ile Röle Kontrol Programı

TCP/IP ile bağlanmadan önce, paralel porttan röle devrenizin çalışıp çalışmadığını kontrol etmelisiniz. Hatta bundan önce ledlerle, lojik voltajları izlemenizi öneriyorum. Program, 4 adet cihazı destekliyor. Bu cihazları aktive edecek röleleri tetiklemek için ise paralel portunuzun 0-3 pinlerini kullanıyoruz. Yazacağımız iki tane program var. Bunlar, sunucu ve istemci programları. Sunucu programı, uzaktan gelen komutları algılayarak, paralel porta gerekli sinyalleri gönderiyor. İstemci programı ise komutları sunucuya IP adresi aracılığı ile göndermeye yarıyor. Her iki program da Microsoft Winsock Component'i kullanmaktadır. Winsock sayesinde, seri bağlantı ve IP üzerinden birçok uygulama gerçekleştirebilirsiniz.



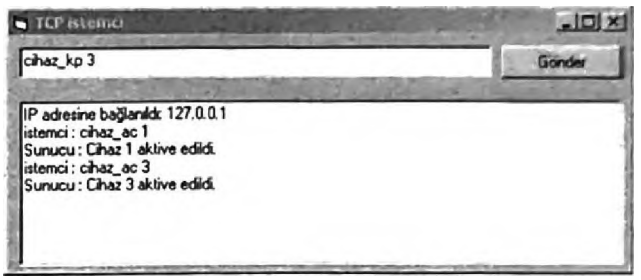
ŞEKİL 10.2: TCP/IP ile röle kontrol programı.

TCP/IP ile Röle Kontrol Programı Nasıl Çalışıyor?

Sunucu programı şu şekilde çalışıyor: İlk olarak form yüklenirken, protokolün hangi portu kullanacağınızı tanımlıyoruz. 1024'ten küçük portlar rezervedir. Bu nedenle programda ben 5002 portunu kullandım. 1024-65535 arası herhangi bir sayı kullanabilirsiniz. Dikkat etmeniz gereken, her iki programda da aynı portların kullanılmasıdır. Daha sonra, bu portu dinlemeye geçiriyoruz. Dışarıdan bağlanan istemci programını ilk açtığınızda ise size sunucunun IP numarası soruluyor. Buraya sunucunun IP adresini girmelisiniz. Bunu sunucunun DOS komut yorumlayıcısındaki IPCONFIG ya da Windows un WINIPCFG programını çalıştırarak görebilirsiniz. Ayrıca form yüklendiği sırada paralel portunuzun BASE adresi de burada tanımlanıyor. Be-

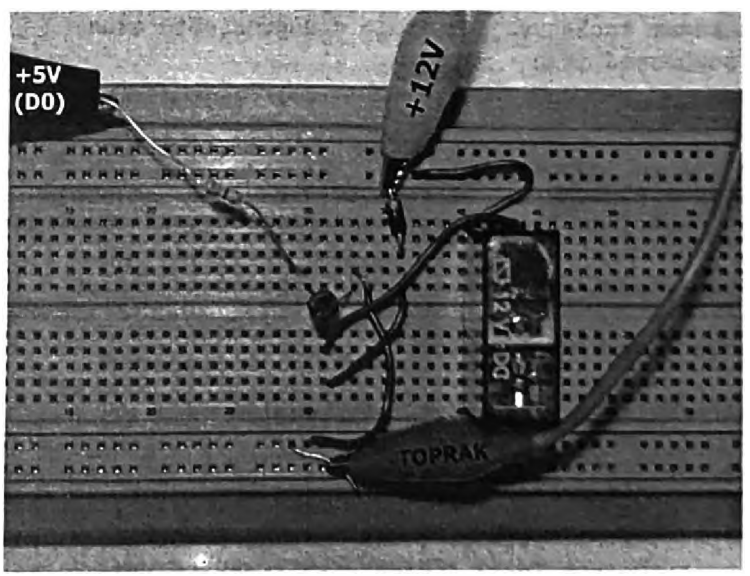
nimki &H3BC idi, genelde &H378 oluyor. Buna Windows'un Aygıt Yöneticisi'nden bakabilirsiniz.

IP adresini girdikten sonra, bağlantı sağlanacaktır. Bağlantı sağlandıktan sonra, istemci tarafından gönderilen kelimeler, sunucunun ekranında çıkacaktır. Sunucudan istemciye mesaj göndermek de mümkündür. Sunucuya bir veri geldiğinde, gelen verinin bir komut olup olmadığını kontrol etmeliyiz. Burada sunucu sadece `cihaz_ac` ve `cihaz_kp` komutlarını algılayabiliyor. Bu komutları istediğiniz gibi oluşturabilirsiniz.



ŞEKİL 10.3: İstemci yazılımı.

Gelen verinin komut olup olmadığını anlamak için, ilk sekiz karakterine bakıyoruz. Eğer bu bir komut ise, sağdan ilk karakteri, yani 1-8 arasında göndereceğimiz parametrenin değerini alıyoruz. Eğer 1-8 arası değilse, istemciye "Hatalı parametre" mesajını gönderiyoruz. Örneğin `cihaz_ac` komutunun parametresi 1-8 arasında ise, sunucu programı `set_pin()` fonksiyonu ile, gönderilen parametreye ait pini yüksek yapıyor. `set_pin()` fonksiyonu, `Inpout32.dll` dosyasından gelen `OUT` komutunu kullanarak, paralel portunuzun verilen pinini yüksek (çıkış) yapmaya yarıyor. Bit yüksek iken o bite bağlı röle, yani cihaz çalışıyor. `set_low()` fonksiyonu ise verilen biti düşük yapıyor ve cihazı durduruyor. Eğer gönderilen komut değil ise, istemciye "Hatalı komut" mesajı gönderiliyor.



ŞEKİL 10.4: Tekli röle kontrol devresinin board üzerinde kurulmuş hali.

TCP/IP ile Röle Kontrolü Sunucu Visual Basic Kodu

```
Dim data(7) As Integer, top_data As Integer, adres As Integer
Private Sub cmdSend_Click()
    ' İstemciye gönderilen veri.
    Call tcpServer.SendData("Sunucu : " & txtSend.Text)
    ' İstemciye gönderilen veri sunucunun txtOutput.Text penceresinde tekrarlanır.
    txtOutput.Text = txtOutput.Text & "Sunucu :" & txtSend.Text & vbCrLf
    ' txtSend.Text penceresi temizlenir"
    txtSend.Text = ""
    txtOutput.SelStart = Len(txtOutput.Text)
End Sub
Private Sub Form_Load()
    '  Paralel portun adresi tanımlanıyor
    adres = Int(Val(txt_adres.Text))
    Out adres, 0
    cmdSend.Enabled = False
    '  Lokal port açılmakta ve Internet bağlantısı beklenmektedir.
    tcpServer.LocalPort = 5002
    Call tcpServer.Listen
End Sub
Private Sub Form_Terminate()
    'Program kapanırken bağlantı kesiliyor.
    Call tcpServer.Close
End Sub
Private Sub tcpServer_Close()
    cmdSend.Enabled = False
    Call tcpServer.Close ' İstemci bağlantıyı keserse sunucu da kesecektir.
    txtOutput.Text = txtOutput.Text & "Bağlantı istemci tarafından kesildi." &
vbCrLf
    txtOutput.SelStart = Len(txtOutput.Text)
    Call tcpServer.Listen ' sonraki bağlantı için beklemede kal.
End Sub
Private Sub tcpServer_ConnectionRequest(ByVal requestID As Long)
    ' Yeni bir bağlantı yapmadan tcpServer'ın kapalı olduğuna emin olun.
    If tcpServer.State <> sckClosed Then
        Call tcpServer.Close
    End If
    cmdSend.Enabled = True
    Call tcpServer.Accept(requestID)
    ' Yeni bağlantı istemini kabul et.Sunucu durum penceresine aşağıdaki mesajı
gönder.
    txtOutput.Text = tcpServer.RemoteHostIP & " IP adresindeki istemciye
bağlanıldı" & vbCrLf & _
        "Port #: " & tcpServer.RemotePort & vbCrLf
End Sub
Private Sub tcpServer_DataArrival(ByVal bytesTotal As Long)
    Dim message As String
    Call tcpServer.GetData(message) ' gelen mesajı al ve message değişkeninde
sakla.
    txtOutput.Text = txtOutput.Text & "istemci : " & message & vbCrLf
    txtOutput.SelStart = Len(txtOutput.Text) 'gelen mesajı ekrana yaz
    Select Case Left(message, 8) 'gelen ilk 8 harf bir komut mudur?
```

```
        Case "cihaz_ac"
            If Val(Right(message, 1)) > 0 And Val(Right(message, 1)) < 8 Then
                    'komutun parametresini alıyoruz.
                     set_pin (Val(Right(message, 1)) - 1) 'Parametreye ait biti
yüksek yapıyoruz.
                     Command2(Val(Right(message, 1)) - 1).Caption = "1"
                Call tcpServer.SendData("Sunucu : Cihaz " & Right(message, 1) & "
aktive edildi.") 'İstemciye onay gönderiyoruz.
                txtOutput.Text = txtOutput.Text & "Sunucu : Cihaz " &
Right(message, 1) & " aktive edildi." & vbCrLf
                txtOutput.SelStart = Len(txtOutput.Text)
            Else
                'parametre hatalı ise onaylanmadı mesajı gidiyor
                Call tcpServer.SendData("Sunucu : Hatalı ya da eksik parametre")
                txtOutput.Text = txtOutput.Text & "Sunucu : Hatali Parametre" &
vbCrLf
                txtOutput.SelStart = Len(txtOutput.Text)
            End If
        Case "cihaz_kp" 'cihaz_ac komutunun tam tersi
            If Val(Right(message, 1)) > 0 And Val(Right(message, 1)) < 8 Then
                     low_pin (Val(Right(message, 1)) - 1)
                     Command2(Val(Right(message, 1)) - 1).Caption = "0"
                Call tcpServer.SendData("Sunucu : Cihaz " & Right(message, 1) & "
kapatıldı.")
                txtOutput.Text = txtOutput.Text & "Sunucu : Cihaz " &
Right(message, 1) & " kapatıldı." & vbCrLf
                txtOutput.SelStart = Len(txtOutput.Text)
            Else
                Call tcpServer.SendData("Sunucu : Hatali ya da eksik parametre")
                txtOutput.Text = txtOutput.Text & "Sunucu : Hatali Parametre" &
vbCrLf
                txtOutput.SelStart = Len(txtOutput.Text)
            End If
        Case Else
            Call tcpServer.SendData("Sunucu : Hatalı komut")
            txtOutput.Text = txtOutput.Text & "Sunucu : Hatalı komut" & vbCrLf
            txtOutput.SelStart = Len(txtOutput.Text)
    End Select
End Sub
Private Sub tcpServer_Error(ByVal Number As Integer, Description As String, ByVal
Scode As Long, ByVal Source As String, ByVal HelpFile As String, ByVal HelpContext
As Long, CancelDisplay As Boolean)
    Dim result As Integer
    result = MsgBox(Source & ": " & Description, _
        vbOKOnly, "TCP/IP hatası")
    End
End Sub
Private Sub Command2_Click(Index As Integer)
If adres = 0 Then MsgBox "Lütfen PARALEL port adresini girip ""Ayarla""ya
tıklayınız. Örnek:&h378", vbCritical + vbOKOnly, "Uyarı": Exit Sub
Select Case Command2(Index).Caption
Case "0"
    set_pin (Index)
    Command2(Index).Caption = "1"
```

```
Case "1"
    low_pin (Index)
    Command2(Index).Caption = "0"
End Select
lbl_data = top_data
End Sub
Private Sub Command3_Click()
    adres = Int(Val(txt_adres.Text))
End Sub
Sub set_pin(pin_no As Integer)
    'verilen pini yüksek yapar
    led(pin_no).FillColor = vbGreen
    top_data = 0
    Select Case pin_no
        Case 0
            data(0) = 1
        Case 1
            data(1) = 2
        Case 2
            data(2) = 4
        Case 3
            data(3) = 8
        Case 4
            data(4) = 16
        Case 5
            data(5) = 32
        Case 6
            data(6) = 64
        Case 7
            data(7) = 128
    End Select
    For i = 0 To 7
        top_data = top_data + data(i)
    Next i
    Out adres, top_data
End Sub
Sub low_pin(pin_no As Integer)
    'verilen pini düşük yapar
    led(pin_no).FillColor = vbRed
    top_data = 0
    data(pin_no) = 0
    For i = 0 To 7
        top_data = top_data + data(i)
    Next i
    Out adres, top_data
End Sub
Private Sub txtSend_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 Then Call cmdSend_Click
End Sub
```

TCP/IP ile Röle Kontrolü İstemci Visual Basic Kodu

```
Option Explicit
Private Sub cmdSend_Click()
    ' Sunucuya veri gönderilir.
    Call tcpClient.SendData(txtSend.Text)
    txtOutput.Text = txtOutput.Text & _
        "istemci : " & txtSend.Text & vbCrLf
    txtOutput.SelStart = Len(txtOutput.Text)
    txtSend.Text = ""
End Sub
Private Sub Form_Load()
    cmdSend.Enabled = False
   ' Local port kurulur ve bağlantı için bekleme durmuna geçilir.
    tcpClient.RemoteHost = InputBox("Bağlanacağınız bilgisayarın IP adresini
giriniz", _
        "IP Address", "localhost")
    If tcpClient.RemoteHost = "" Then
        tcpClient.RemoteHost = "localhost"
    End If
    tcpClient.RemotePort = 5002 ' sunucu portu
    Call tcpClient.Connect ' Diğer bilgisayara bağlantı gerçekleştiriliyor.
End Sub
Private Sub Form_Terminate()
    Call tcpClient.Close
End Sub
Private Sub tcpClient_Close()
    cmdSend.Enabled = False
    Call tcpClient.Close ' Sunucu bağlantıyı kesitiği için istemci de bağlantıyı
keser.
    txtOutput.Text = txtOutput.Text & "Bağlantı sunucu tarafından kesildi." &
vbCrLf
    txtOutput.SelStart = Len(txtOutput.Text)
End Sub
Private Sub tcpClient_Connect()
    ' Bağlantı gerçekleştiğinde aşağıdaki mesaj gönderilir.
    cmdSend.Enabled = True
    txtOutput.Text = "IP adresine bağlanıldı: " & _
        tcpClient.RemoteHostIP & vbCrLf
End Sub
Private Sub tcpClient_DataArrival(ByVal bytesTotal As Long)
    Dim message As String
    Call tcpClient.GetData(message) ' sunucudan veri alınıyor.
    txtOutput.Text = txtOutput.Text & message & vbCrLf
    txtOutput.SelStart = Len(txtOutput.Text)
End Sub
Private Sub tcpClient_Error(ByVal Number As Integer, Description As String, ByVal
Scode As Long, ByVal Source As String, ByVal HelpFile As String, ByVal HelpContext
As Long, CancelDisplay As Boolean)
    ' İstemcinin sunucu bağlantısı gerçekleşmezse aşağıdaki komutlar icra
edilecektir.
    Dim result As Integer
```

```
    result = MsgBox(Source & ": " & Description & vbCrLf & "Sunucu bağlantı
hatası!", _
        vbOKOnly, "TCP/IP hatası")
    End
End Sub
Private Sub txtSend_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 Then Call cmdSend_Click
End Sub
```

ONBİRİNCİ BÖLÜM

11

BASKI DEVRE TEKNİKLERİ

Baskı Devreler

Bu bölümde devrenin nasıl yapıldığını öğreneceğiz. Bugüne kadar tüm uygulamalarımızı board üzerinde gerçekleştirdik. Bunların hepsi uygulama amaçlı devrelerdi. Gerçekte işe yarayacak ve çalışacak bir devreyi board üzerinde tutmak çok sağlıklı bir iş değildir. Devrenin çalıştığından emin olunduktan sonra, bu devreyi bir kart üzerinde hazırlamak gerekir. Bunun bir yöntemi delikli plaka kullanmaktır. Devre malzemeleri delikli plaka üzerine yerleştirilerek, alt taraftan montaj kabloları ile gerekli atlamalar yapılabilir. Fakat bu da çok sağlıklı bir iş değildir. Sadece ufak devrelerde kullanılabilir. Tek avantajı, plakayı delmeye gerek olmayışıdır. Bir diğer yöntem ise, bakır plaka ile baskı devre kalemi kullanmaktır. Yani devreyi bakır plaka üzerine kendiniz el ile de çizebilirsiniz. Bu da karmaşık devrelerde kolay bir iş değildir.

Baskı devre kalemi ile kafadan çizim yapmak zor olduğu için, baskı devrenin çizimini bilgisayar ile gerçekleştirmek gerekir. Bunun için birçok program bulmak mümkün. Ben aşağıda bir tanesinden bahsedeceğim. Yöntem şöyle: Baskı devre tasarlandıktan sonra, şeffaf bir kağıda çıktısı alınır. Daha sonra bu çıktının üzerinden kurşun kalem ile geçilir. Sonra şeffaf kağıdın arkası çevrilir ve az önce kurşun kalem ile çizilen kısım plakanın bakır yüzeyine gelecek şekilde konulur. Daha sonra kağıdın arkasından kurşun kalem izleri üzerinden bir daha geçilir. Böylece devrenin izi bakır plakaya aktarılmış olur. Ardından, bakır plakadaki bu izin üzerinden dikkatli bir şekilde baskı devre kalemi ile geçilir. Böylece bakır plaka aside atılmaya hazır hale gelmiş olur.

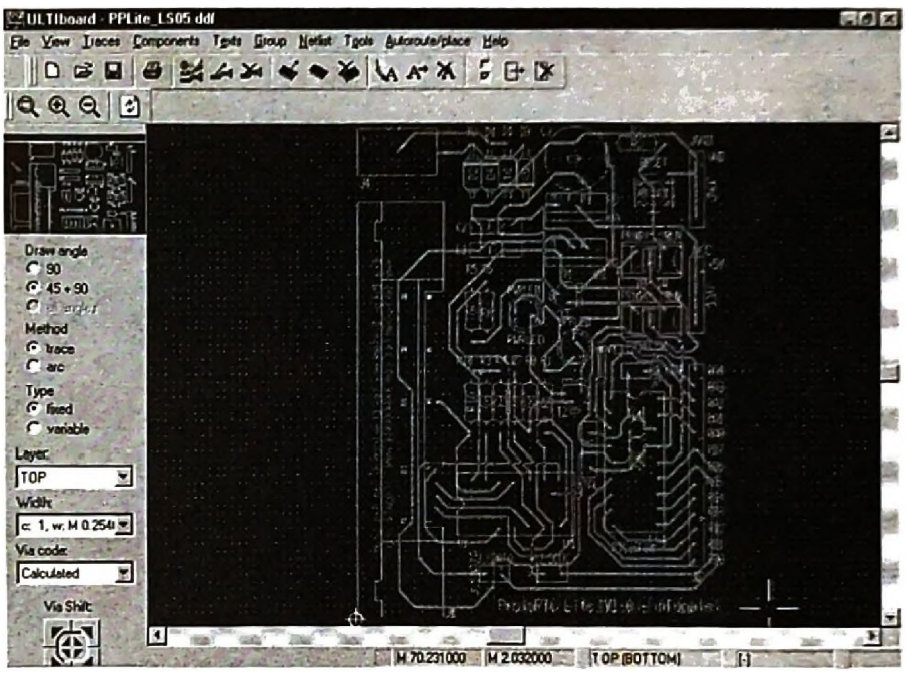


ŞEKİL 11.1: Baskı devre için gereken malzemeler.

Yukarıdaki yöntem de oldukça zahmetli ve zor bir iştir. Bunların dışında, Positive20, oje, serigraf gibi baskı devre hazırlama yöntemleri de vardır. Bizim kullanacağımız yöntem ise hepsinden daha kolay ve oldukça kaliteli bir yöntem olan P-n-P baskılı devre transfer filmi kullanmak. P-n-P nedir, kısaca bahsedelim. P-n-P prototip hazırlamada veya düşük hacimli baskı devre imalatında kullanılmak üzere tasarlanmış, kullanımı ev ütüsü dışında özel araçlar

gerektirmeyen bir üründür. Bir tarafı mat, diğer tarafı parlak mavi bir kağıt şeklindedir. P-n-P ile baskı devre, 10 dakikada aside atılmaya hazır hale gelmektedir.

P-n-P uygulamasını adım adım gerçekleştirmek için, geçenlerde Infogate firmasına bir ziyarette bulundum. Tüm uygulamayı oradaki laboratuvarda gerçekleştirdik ve fotoğrafları orada çektik. Öncelikle bir devreye ihtiyacımız vardı. Bunun için bugüne kadar gerçekleştirdiğimiz PIC uygulamalarımızda kullandığımız, Infogate in ProtoPIC Lite PIC programlama ve deneme kartını seçtik. Bu kartın baskı devresi UltiBoard adlı baskı devre programı ile tasarlanmış. Sizde baskı devrelerinizde bu programı kullanabilirsiniz. Bu programı ve P-n-P filmlerini Kaynaklar bölümünden bilgilerine ulaşabileceğiniz Infogate firmasından temin edebilirsiniz.

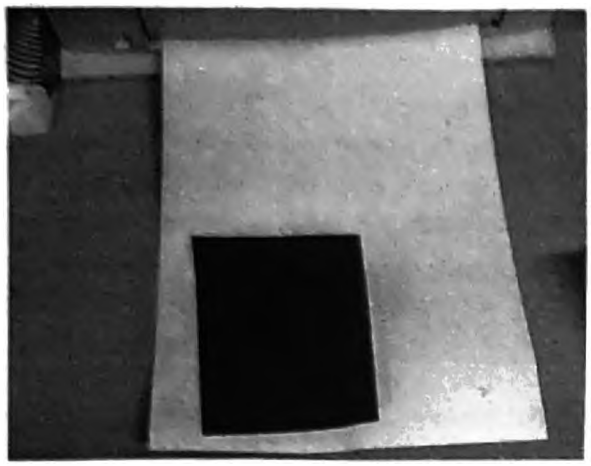
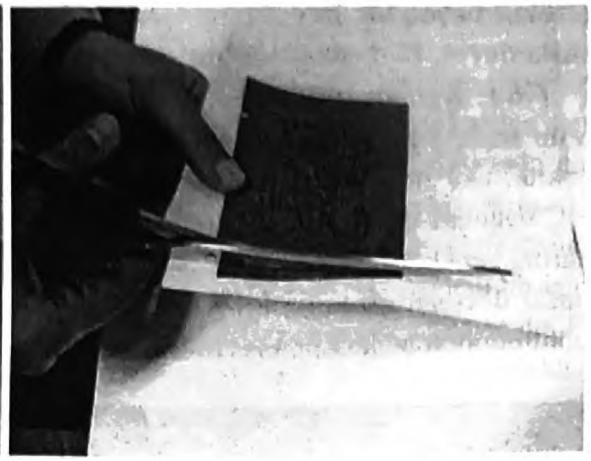


ŞEKİL 11.2: UltiBoard baskı devre dizayn programı.

P-N-P Baskı Devre Uygulaması

Yapmamız gereken ilk şey P-n-P kağıdına tonerli bir yazıcı ile kağıdın mat tarafına çıktı almak. Eğer lazer yazıcınız yoksa, boş bir kağıda çıktı alıp onu daha sonra fotokopi makinesi ile P-n-P'ye aktarabilirsiniz. Filmi hesaplı kullanmak istiyorsanız, ilk önce bildiğimiz boş kağıda çıktıyı alın. Daha sonra çıktı aldığınız kağıt üzerinde devrenin basılı olduğu yeri kesin ve yerine P-n-P kağıdından yapıştırın. Daha sonra, filmin mat tarafına yazılacak şekilde kağıdı tekrar yazıcınıza yerleştirin ve yeniden çıktı alın. Aynı işlemi fotokopi makinesi ile de gerçekleştirebilirsiniz. Lazer yazıcıyı ya da fotokopi makinesini en koyu duruma ayarlamayı unutmayın. Çıktıyı baskı devrenin aynadaki görüntüsü şeklinde almalısınız. Bunu baskı devre programları ayarlıyor zaten.

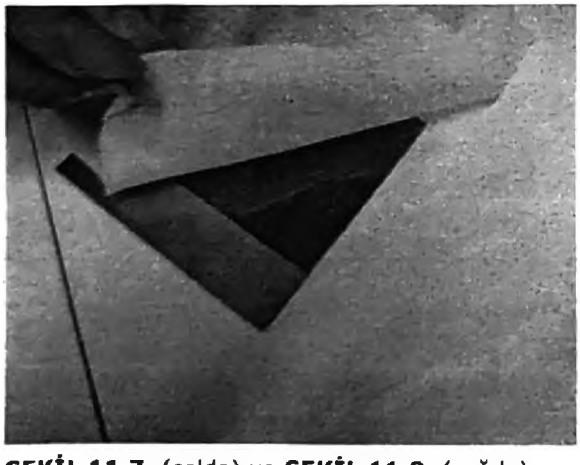
Bakır plakayı iyice temizleyin. Kağıda çıktıyı aldıktan sonra (Şekil 11.3), filmi plaka boyutlarında kesip (Şekil 11.4) bakır plakanın üzerine, baskılı (mat) tarafı bakır yüzeye gelecek şekilde (Şekil 11.5) yerleştirin. Ütüleme sırasında kaymaması için filmi kenarlarından bakır plakaya bantlayın (Şekil 11.6).

ŞEKİL 11.3. (solda) ve **ŞEKİL 11.4.** (sağda)



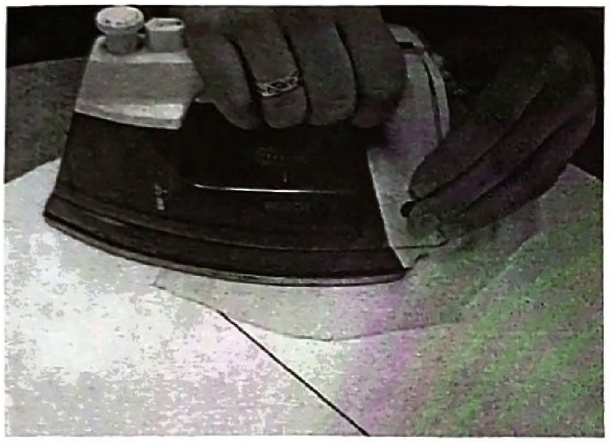

ŞEKİL 11.5. (solda) ve **ŞEKİL 11.6.** (sağda)




ŞEKİL 11.7. (solda) ve **ŞEKİL 11.8.** (sağda)

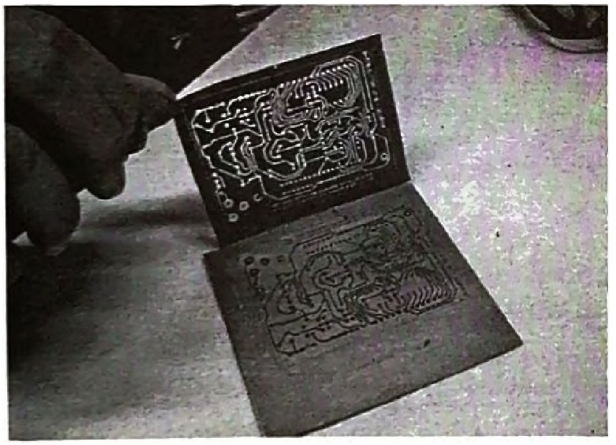
Daha sonra, üzerine ince bir ütü bezi yerleştirin (Şekil 11.7). Ardından, ütüyü yünlü/pamuklu ayarına getirdikten sonra kızdırıp buharsız konuma getirin. Aksi taktirde devreyi suda haşlamış olursunuz. Hafifçe bastırarak, bezin üzerinden ütülemeye başlayın (Şekil 11.8).

Ütünün arka kısmı ile ütüyü dairesel hareket ettirerek ütülemeye devam edin (Şekil 11.9). Bu şekilde 3-5 dakika arası plakayı ütülemeye devam edin. Plakanın her tarafının eşit düzeyde ısıtılmasına dikkat edin. Baskılı devre çizimi filmde iyice belirginleşmeye başladığında ütülemeyi kesin. Plaket iyice soğuduktan sonra filmi kendinize doğru yavaşça çekin (Şekil 11.10) ve plaketten ayırın (Şekil 11.11). Eğer baskıda ve yollarda kesiklikler varsa hataları baskı devre kalemi kullanarak elinizle tamamlayabilirsiniz.





ŞEKİL 11.9. (solda) ve **ŞEKİL 11.10.** (sağda)



ŞEKİL 11.11.

Şimdi sıra geldi plakayı aside atmaya. Asit dediğimizde akla, gayet fantastik bir şekilde fokurdayan ve içine ne atsak eriten bir sıvı akla geliyor. Burada bahsi geçen asit biraz daha hafif ve kolaylıkla elde edilebilir. İhtiyacımız olan malzeme, perihedrol ve tuz ruhu. Perihedrolü eczanelerde bulabilirsiniz. Tuz ruhunu ise marketlerde bulmak mümkündür. Ölçeği ise 1/3 şeklinde. Yani 1 ölçek perihedrole (Şekil 11.13), 3 ölçek tuz ruhu (Şekil 11.12) katıyorsunuz. Elde edilen aside elinizi sokmamanız gerektiğini söylememe gerek yok herhalde. Çok yakıcı olmasa bile, yine de zararlı olduğunu ve sorumluluğun size ait olduğunu unutmayın. Kap içinde, devrenin tamamı karışıma girecek şekilde bir seviye ayarlayın. Bu, sizin kullandığınız kabın boyutuna da bağlı. Devreyi aside atın (Şekil 11.14) ve kabı hafif hafif ileri geri oynatarak asidin

dalgalanmasını ve devrenin asit içerisinde gezmesini sağlayın. Asidin rengi hafiften yeşile dönmeye (Şekil 11.15) başlayacaktır. Bu bakırın plaka yüzeyinden ayrılmaya başladığını gösterir. Bu esnada kötü bir koku yayılacaktır. Bu kokuyu akciğerlerinizle paylaşmasanız daha iyi olur. Bir süre sonra bakırın, plaka üzerinden ayrıldığını ve sadece baskılı kısımları devre üzerinde kaldığını (Şekil 11.16) göreceksiniz.





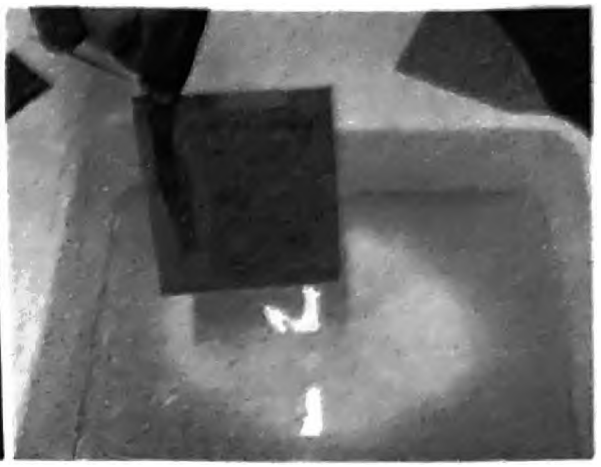
ŞEKİL 11.12. (solda) ve **ŞEKİL 11.13.** (sağda)



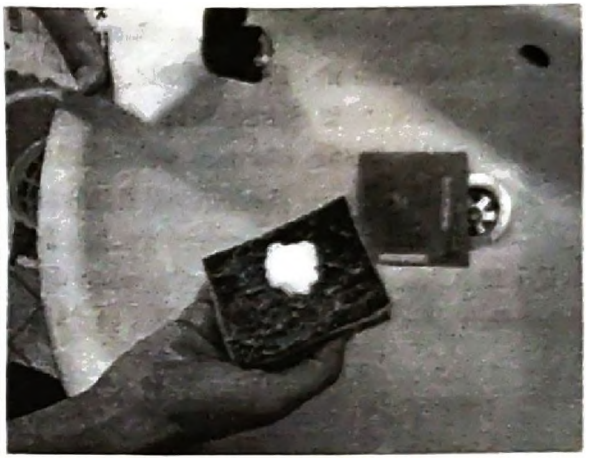


ŞEKİL 11.14. (solda) ve **ŞEKİL 11.15.** (sağda)

Artık plakayı asitten çıkarabilirsiniz (Şekil 11.17). Daha sonra asidi tuvalete dökün ve kabı suyla doldurup plakayı su içinde durulayın (Şekil 11.18). Daha sonra, bir sıvı yüzey temizleyicisi (Şekil 11.19) ve sert yüzeyli bir (Şekil 11.20) sünger ile plakayı iyice çitileyin ve siyah baskı ve toner izlerinden kurtulun. Sadece bakır izler kalana kadar çitilemeye devam edin (Şekil 11.21). Daha sonra plakayı suda temizleyin ve kurulayın (Şekil 11.22). Ardından plaka kenarındaki fazlalıkları maket bıçağı (Şekil 11.23) ile ayırın.

ŞEKİL 11.16. (solda) ve **ŞEKİL 11.17.** (sağda)

ŞEKİL 11.18. (solda) ve **ŞEKİL 11.19.** (sağda)

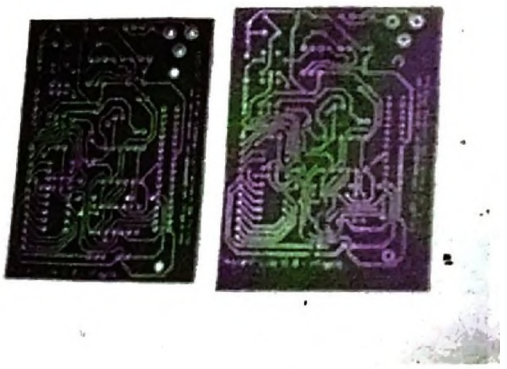



ŞEKİL 11.20. (solda) ve **ŞEKİL 11.21.** (sağda)




ŞEKİL 11.22. (solda) ve **ŞEKİL 11.23.** (sağda)

Sıra geldi devreyi delmeye. Bunun için mini uçlu, devre delmek için kullanılan matkaplardan kullanmalısınız. Bu matkabı, eğer İstanbul'daysanız, Karaköy Selanik Pasajı'nda ya da Kadıköy Yazıcıoğlu İşhanı'nda, İstanbul dışındaysanız elektronik malzeme satan yerlerde bulabilirsiniz. Devre elemanı yerleştirilecek yerlerden devreyi deldikten sonra (Şekil 11.24) baskı devremiz hazır. Neredeyse fabrikasyona yakın bir baskı devre elde ettik. Doğru sıcaklık ve zamanlama, lazer yazıcının ve fotokopi makinesinin toner tipine ve kalitesine doğrudan bağlı olduğundan, mükemmel sonuca ulaşmak için birkaç deneme yapmanız gerekebilir. Aynı devrenin fabrikasyon çıkışını Şekil 11.25'te görebilirsiniz.

ŞEKİL 11.24. (solda) ve **ŞEKİL 11.25.** (sağda)

ONİKİNCİ BÖLÜM

12

LINUX İLE OTOMASYON

Linux ve Paralel Port

Şimdiye kadar gerçekleştirdiğimiz birçok otomasyon uygulamasını, Windows ya da DOS tabanlı PC'ler için gerçekleştirmiştik. Şimdi, bu uygulamaları Linux altında nasıl gerçekleştirebileceğimizi öğreneceğiz. PC ile otomasyon yapabilmek için kullandığımız temel arabirim paralel porttur. Paralel port üzerinde 8 bitlik data portu bulunmaktadır. Uygulamamızda bu 8 pinin 4 tanesine bağladığımız ledleri yakacağız. Daha önce de bahsettiğimiz gibi, eğer port üzerinden bir led yakıp söndürebiliyorsak, aynı port üzerinden birçok otomasyon uygulamasını gerçekleştirebiliriz demektir.



Linux'ta Metin Editörü

Linux işletim sisteminde bir metin editörü kullanmak Windows altında kullanılan Notepad yazılımından daha farklıdır. Linux, Unix tabanlı bir işletim sistemi olduğundan, çok gelişmiş konsol uygulamaları vardır. Metin dosyalarımızı görüntülemek, değiştirmek ve daha sonra kaydetmek için bu kuvvetli yazılımlardan birisi olan "vi" programını kullanacağız. Daha sonra C kodunu yazacağımız bu programı kullanmayı öğrenmek gerekir, çünkü "vi" tüm linux dağıtımları ile birlikte varsayılan durumda yüklenir ve bizi hiç bir zaman yarı yolda bırakmaz.

Burada "vi" metin işleme programının tüm özelliklerine değinmeyeceğim, fakat bir metin yazabilmemiz ve kaydedebilmemiz için gerekli her şeyi anlatmaya çalışacağım. "vi" programı tüm diğer konsol uygulamaları gibi "vi" yazılarak çalıştırılır; bu şekilde boş bir metin dosyası açmış oluruz. Eğer halihazırda mevcut olan bir metin dosyası açmak istiyorsak, metin dosyasını "vi" programına parametre olarak göndermemiz gerekir. (örneğin, `vi metin_dosyasi_adi`)

"vi" programının iki durumu bulunmaktadır. Bunlardan ilki read mode diğeri ise insert mode olarak bilinir. "read mode" içerisinde dosyada herhangi bir değişiklik yapamazsınız bu mod sadece dosya görüntüleyebilmenizi sağlar. "insert mode" ise yeni metin ekleyebileceğiniz, silme ve değiştirme işlemlerini yapabileceğiniz durumdur. vi açıldığında varsayılan olarak "read mode" ile açılır, eğer "insert mode"a geçmek istiyorsanız, bu durumda "i" tuşuna basmalısınız. Eğer satır sonuna gidip "insert mode"a geçmek istiyorsanız "A" karakterini kullanmalısınız. Tekrar "read mode"a geçmek istendiğinde "esc" tuşuna basmanız yeterlidir. Aşağıda verilebilecek karakterleri açıklamaları ile veriyorum:

I: insert mode'a geç (insert)

A: satır sonuna git ve insert mode'a geç (append)

r: bulunduğun karakteri değiştir (replace)
rw: bulunduğun kelimeyi değiştir (replace word)
x: bulunduğun karakteri sil (delete)
xw: bulunduğun kelimeyi sil (delete word)
dd: bulunduğun satırı sil
dd 10: 10 satır sil
yy: satır kopyala
p: kopyaladığın satırı yapıştır

"vi" metin editörüne komut verebilmek için önce "read mode"da iken ":" karakterine basmalısınız. Verilecek komutlar bittikten sonra ise "enter" tuşuna basılmalıdır. Verilebilecek komutların bir listesi aşağıda verilmektedir:

:w – kaydet (write)
:q – çıkış (quit)
:wq – önce kaydet, sonra çık
:q! – dosyada yapılan değişiklikleri kaydetme ve acilen çık.

Linux'ta C Kodu Derlemek

Bir metin dosyası yazıp kaydetmeyi öğrendiğimize göre, artık bir C programını nasıl derleyebileceğimi görebiliriz. C kodu derlemek için Linux altında cc programı kullanılabilir. Bu program C kodu bulunan metin dosyasını ve oluşturulacak olan makine kodunu içerecek olan dosya adını parametre alarak çalıştırır. Örneğin metin dosyamızın adı `test.txt` olsun, bu durumda, aşağıdaki komut çalıştırıldığında cc (c compiler) `test.txt` metin dosyası içerisindeki C kodunu derleyerek test binary dosyasını oluşturacaktır:

```
cc test.txt –o test
```

Test dosyasını çalıştırmak için ise aşağıdaki komutu çalıştırmak yeterli olacaktır:

```
./test
```

Eğer metin dosyasındaki C kodunda bir yazım hatası yaptıysanız cc size bu hatanın bildirimini yine standart output olan ekrana verecektir. Gerek vi metin editörünün, gerekse cc c derleyicisinin burada anlatılanlardan çok daha farklı opsiyonları ve parametreleri olduğunu, fakat burada, bir c programı yazıp derleyebilmek için gerekli olan yeterli bilgiyi verdiğimizi belirtmek gerekir.

C++ ile Paralel Port Kontrolü

Kontrollü ilerleyebilmek amacı ile her kod kısmının altında ne işe yaradıklarını kısaca açıklayarak ilerlemeyi uygun gördüm.

```
#include <stdio.h>
#include <string.h>
```

```
#include <stdlib.h>
#include <unistd.h>
#include <sys/io.h>
```

Önce gerekli kütüphane dosyalarını programa tanıtıyoruz. Temel olarak kullanacağımız olan **outb()** fonksiyonu **io.h** dosyasının içerisindedir. Bir de, gerekecek olan **ioperm()** fonksiyonunu da **unistd.h** kütüphanesinden çekiyoruz. Diğer dosyalar ise standart C fonksiyonları için gerekli olan kütüphane dosyalarıdır.

```
#define DATA 0x378
#define STATUS DATA+1
#define CONTROL DATA+2
```

Gerekli port değişkenlerini tanıtıyoruz. Burada 0x378 değeri paralel portunuzun base adresi ile aynı olmalıdır. Bu adresi, PC'nizin BIOS ayarlarından izleyebilir ve değiştirebilirsiniz. Genelde bu adres 0x378, yani h0378 şeklinde gelir.

```
int main()
{
int test;
int again = 1;
int datatoport;
int datafromport;
printf("\n*****Parallel Port Yonetim*****\n\n");
```

Programın ana fonksiyonunu ve kullanacağımız değişkenleri yaratıyoruz. Sonra **printf** komutu ile ekrana programın adını yazıyoruz.

```
while (again)
{
printf("\nParallel porta gonderilecek 8 bitlik veriyi ondalik olarak giriniz\n");
printf("(orn. 255): ");
scanf("%d", &datatoport);
```

Programın bize **again** değişkeni "0" olana kadar porta göndermek istediğimiz değerleri sürekli olarak sorması için, bir **while** döngüsü yaratıyoruz. Daha sonra kullanıcıdan porta gönderilmek istenen veriyi alıyoruz ve bu veriyi **datatoport** değişkenine yüklüyoruz.

```
test = ioperm(DATA,3,1);
/* (port, num, turn on) */
if (test)
{
printf("\nparallel port is not accessable\n");
exit (-1);
}
else
printf("\naccessed to parallel port\n");
```

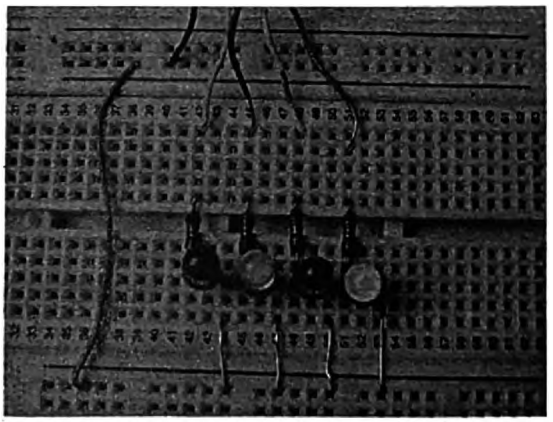
Burada **ioperm()** fonksiyonu ile, kullanıcıya paralel porta erişim hakkı veriyoruz. Eğer herhangi bir problem ile karşılaşılmaz ise bu fonksiyon bize **true** döndürüyor ve bu değeri **test** değişkenine yüklüyor. Sonra bu **test** değişkenini kontrol ediyoruz ve ekrana paralel porta erişilme durumunu yazdırıyoruz.

```
outb(datatoport,DATA);
printf ("\ndata is written to the port\n");
printf ("again? (1/0): ");
scanf ("%d", &again);
}
```

Sonunda **outb()** fonksiyonu ile, az önce kullanıcıdan alınan **datatoport** değişkenini DATA portuna yazıyoruz. Daha sonra başka bir veri göndermek istenip istenilmediği soruluyor ve duruma göre az önceki **while** döngüsünün başına dönülüyor ya da programdan çıkılıyor.

```
outb(0000, DATA);
ioperm(DATA, 3, 0);
exit (0);
}
```

Programdan çıkmadan önce data portundaki değerleri sıfırlıyoruz ve erişimi durduruyoruz. Programı çalıştırdığımız zaman bize porta göndermek istediğimiz veri soruluyor. Örneğin programa "13" değerini gönderelim. Bu değer DATA portu üzerine "1101" şeklinde yerleşecektir. Bu durumda lojik "1"e denk gelen ledler yanacaktır. Ledlerin zarar görmemesi amacı ile 470 ohm'luk direnç kullanabilirsiniz. Devrenin şeması Şekil 12.1'deki gibidir. Led kontrol devresi ile ilgili detaylı bilgiyi Bölüm 1'de bulabilirsiniz.



ŞEKİL 12.1: Led kontrol devresinin board üzerinde kurulmuş hali.

Bu uygulamada temel olan lojik çıkışları elde edebilmekti. Bunları her zamanki gibi ledler aracılığı ile izledik. Bu noktadan sonra, program diğer uygulamalar için geliştirilebilir. Örneğin, herhangi bir cihazı açıp kapatmak için bu lojik voltaj ile bir röle devresini tetikleyebilirsiniz.



ŞEKİL 12.2: Devrenin paralel porta bağlantısı.

ONÜÇÜNCÜ BÖLÜM

13

PC KONTROLLÜ TERMOMETRE

PC Kontrollü Termometre Uygulaması

İnternet üzerinden ev otomasyonu projesini anlatırken, evin sıcaklığını algılayabileceğimizden bahsetmiştik. Bu yazı dergide yayımlandıktan sonra, sıcaklık algılama işinin nasıl olduğu ile ilgili oldukça fazla sayıda e-posta almıştım. Bu nedenle, bu bölümde PC üzerinden sıcaklığı izleyebileceğimiz bir devre gerçekleştireceğiz. Uygulamamız sayesinde iç ve dış sıcaklığı Visual Basic ile yazdığımız bir program ile Windows altından izleyebileceğiz.

Bölüm 10'daki İnternet tabanlı ev otomasyonu projesini, bu projeyle birleştirmek mümkündür. Örneğin, evinizin sıcaklığını bu program ile algılayarak kaloriferin ya da ısıtıcının otomatik devreye girmesini sağlayabilirsiniz. Detaylı olarak konuya girmeden önce, termometrelerden kısaca bahsedelim. Standart termometrelerin çalışma prensipleri, civanın ısıya verdiği tepkiye dayanır. Sıcaklığa göre cıva, dereceli bir bar üzerinde hareket eder. Bu yöntem, sıcaklığı algılamak için oldukça ilkel bir yöntemdir. Birçok analog veri, günümüz teknolojisinde sensörler aracılığı ile dijital ortama aktarılabilmektedir. Bunlardan bir tanesi de sıcaklık sensörüdür. Devremizin kilit elemanı, bu sıcaklık sensörlerinden bir tanesi olan, Dallas Semiconductor firmasının ürettiği DS1621 entegresidir.

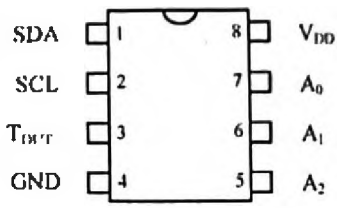
Bu devrenin tasarımı bana ait değil. Uygulamaya kaynak olarak kullandığım sitenin adresi `http://www.riccibitti.com`. Fakat bu devrenin esas tasarımcısı Claudio Lanconelli isminde bir PonyProg programcısı. PonyProg ise kullanıcı arayüzü oldukça kolay olan seri bir programlayıcı. Bu programlayıcı ile bir Atmel ve PIC gibi çok seri aygıtı programlamak mümkün. Daha detaylı bilgiyi Claudio'nun `http://www.cs.unibo.it/~lanconel` adresli sayfasından elde edebilirsiniz. Aynı sayfada, bu bölümdeki devremizle de çalışabilecek Web tabanlı bir termometre yazılımını ve kodlarını bulabilirsiniz.

Termometre Devresi

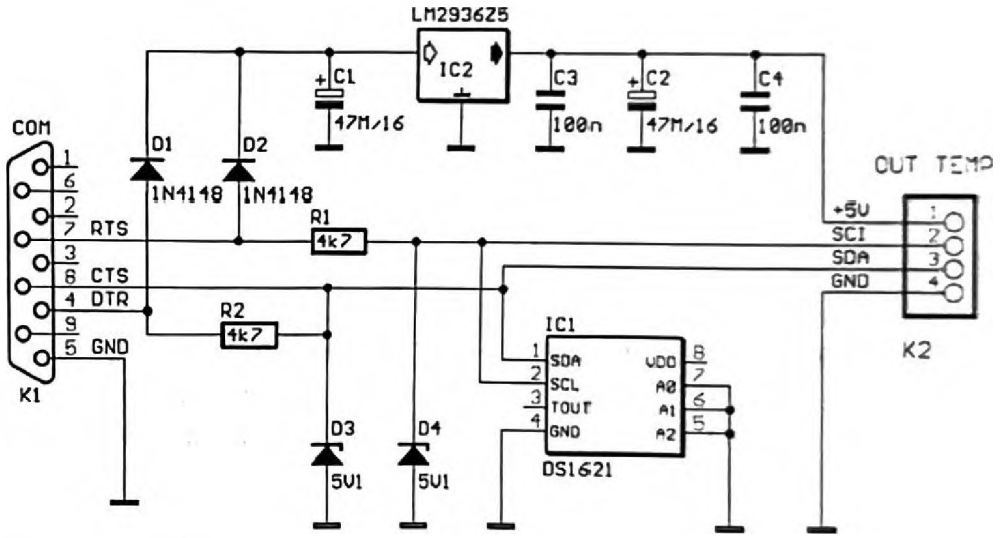
Yukarıda da bahsettiğimiz DS1621, sıcaklık ölçümlerinde sık kullanılan bir entegredir. 9-bitlik dijital ölçüm yapabilir. Entegrenin uygulamamızda kullanmadığımız birçok faydalı özelliği var. Örneğin, entegrenin üzerinde termal alarm çıkışı var. Bu özellik sayesinde ortam kullanıcı tarafından belirlenen bir sıcaklığa ulaştığında TOUT pini aktif hale geçebilmekte ve bu çıkış, ortam sıcaklığı yine kullanıcı tarafından belirlenen bir değere geri düşene kadar aktif durumda kalabilmektedir. Bunun gibi kullanıcı-tanımlı derece ayarları, entegre üzerinde kalıcı bir hafızaya yerleşiyor. Bu nedenle ayarların devre kurulacak olan sisteme dahil edilmeden önce yapılması gerekiyor. Tüm bu ayarlar, okuma işleminin de gerçekleştiği entegre üzerindeki 2 yönlü seri giriş ve çıkış portu sayesinde yapılabiliyor. Bu işlemleri yapabileceğiniz komutların olduğu bir tabloyu `http://www.dallassemiconductor.com` adresinden elde edebileceğiniz DS1621 datasheet'inde bulabilirsiniz.

DS1621'in sıcaklığı ölçmek için temelde ihtiyacı olan tek şey +5 Volttur. Genellikle, bunun gibi IIC veriyolu ile çıkış veren entegrelerden elde edilen veriler paralel olarak okunur. Fakat bu uygulamada verileri seri port aracılığı ile algılayacağız. Zaten entegre üzerinde seri giriş ve çıkışını sağlayan bir pin (SDA) bulunmakta. Seri portu tercih etmemizin diğer sebepleri ise, kendi üzerinde 12 Volt olması ve Visual Basic ile gelen MSComm kontrolü ile rahat bir şekilde kontrol edebiliyor olmasıdır. Fakat IIC veri yolunu, seri port (RS232) üzerinden çalışabilir hale getirmek için bir kaç numara yapmamız gerekmektedir. IIC devreleri +5 Volt ile çalışır. RS232 ise –

12...+12 Volt arası çalışır. RS232 sinyallerinin IIC girişlerine zarar vermemesi için, DS1621'in SDA data girişine ve SCL saat girişine bir voltaj limitleyici (R1,DZ2,R2,DZ1) devresi eklenmiş durumdadır. SDA, pini data transferi için kullanılmaktadır. Bu pini seri portun CTS pinine bağlıyoruz. SCL pini ise saat hızı için kullanılmaktadır. Devrenin tek termometreli şemasını Şekil 13.2'de görebilirisiniz. Program çift termometre ile ölçüm yapabilmektedir. Eğer aynı devre ile hem iç hem de dış sıcaklığı ölçmek istiyorsanız, `http://www.riccibitti.com/pc_therm.gif` adresindeki devre şemasını kullanmalısınız.



ŞEKİL 13.1: DS1621 entegresi.



ŞEKİL 13.2: PC Kontrollü termometre devresi.

Devre ile ilgili ufak bir uyarıda bulunmak istiyorum. Devre düşük voltaj ile çalıştığından, ek bir beslemeye ihtiyaç duyulmamaktadır. RS232 den gelen +12 Voltu LM2936 devresi ile regüle ediyoruz. Bu entegreyi, dışarıdan 9Voltluk pil ile beslemek istemediğiniz sürece regülasyon için alışılmış olan L7805 entegresi ile değiştirmeyiniz.

PC Kontrollü Termometre Yazılımı

Termometre yazılımını Visual Basic ile geliştiriyoruz. Programı derleyebilmek için Visual Basic 5 ya da 6 sürümlerine ihtiyacınız var. Kodların sadece bir kısmını yayınlayabiliyoruz. Tamamını isterseniz, `http://www.riccibitti.com/software/pc_therm_sources.zip` adresinden çekebilirsiniz. Programı çok hızlı bir sistemde deniyorsanız, `iic_wait()` fonksiyonuna bir bekletme rutini eklemeniz gerekebilir.



ŞEKİL 13.3: PC kontrollü termometre programı.

Programı ilk çalıştırdığınızda setup dosyasının bulunamadığına dair bir uyarı alacaksınız. Bu dosya otomatik olarak yaratılacak ve bir daha girdiğinizde bu mesajı almayacaksınız. Program başlatıldığında, minimize olmuş durumda çalışıyor ve taskbar'da duruyor. Bu sayede aynı sistem saati gibi ortam sıcaklığı hakkında bilgi alabiliyorsunuz. Bu ikonun üzerine tıkladığınızda, ayar penceresi açılıyor. Bu pencerede, devrenin takılı olduğu portu doğru olarak seçmelisiniz. Aksi takdirde yanlış ölçüm sonuçları ya da "unable do read" hatası alabilirsiniz. Eğer "log data on file" seçeneğini işaretlerseniz, ölçümler, `pc_therm.txt` adlı dosyaya peşi sıra kaydedilecektir.



ŞEKİL 13.4: Termometre programı araç çubuğuna yerleşiyor.

Devrede çok eleman olduğu için yapması ve board üzerinde kurması biraz zahmetli olabilir. Bu nedenle devreyi uygulayacaksanız, arkalı-önlü baskı devre şemasını (PCB) `http://www.riccibitti.com/pc_thermometer_single_channel.pdf` adresinden indirmenizi, hatta mümkünse Bölüm 11'de bahsettiğim baskılı devre kağıdı kullanarak denemenizi tavsiye ederim.

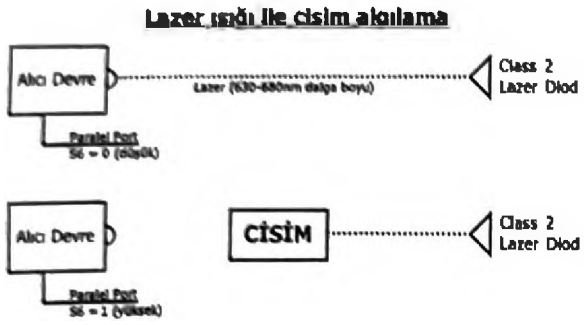
ONDÖRDÜNCÜ BÖLÜM

14

LAZER İLE HIRSIZ ALARMI

Lazerler

Bu bölümümüzde lazer kalemi kullanarak paralel port üzerinden basit bir tetikleme işlemi gerçekleştireceğiz. Bu tetikleme ile paralel portun STATUS pinlerinden bir tanesini toprağa göndereceğiz.



ŞEKİL 14.1: Lazer ile cisim algılama.

Foto transistör ile lazer ışığı arasına bir cisim girdiğinde transistör tetiklemesi kesilecek ve STATUS pini yüksek olacak. Yazacağımız Visual Basic kodu ile paralel portun bu STATUS pinini kontrol edeceğiz ve ekrana araya cisim girdiği zaman bir uyarı mesajı yazdıracağız.Fakat hepsinden önce lazer ışığından biraz bahsetmekte fayda var.

Sıradan kaynaklardan elde edilen ışıklar rasgele fazlarda yayılır ve ışık tek renkli bile olsa, bir miktar yok olma oluşur. Lazer ışığı ise tam fazdadır ve yok olma yerine destekleme oluşmaktadır. Bir lazer ışığı elde etmek için normal seviyeden daha çok atomu uyarılma seviyesine çıkarmak gerekir. Buna population inversion (nüfus evrilmesi) denir. Bir atomu ısıtma gibi yollarla, elektronlar dıştaki yörüngeye kayarlar. Örneğin bir hidrojen atomunu 1'nci seviyeye çıkarmak için 10eV gerekir. Bir madde ısıtıldığında atomlar yüksek seviyelere uyarılır ve düşerler. Yüksekten düşüğe ve düşükten yükseğe yapılan her geçişte değişik bir dalga boyu yayılır. Bütün bunların birleşimi olarak, ısınan bu maddenin parladığını görebiliriz. Lazer ışığı ise sadece tek bir geçişten elde edilir ve bu yüzden doğası gereği tek renklidir. Tipik bir lazer, flaş tüpü ile çevrilmiş bir yakut çubuktan oluşur. Çubuğun uçları birbirine paraleldir. Bir ucu ayna ile gümüşlenmiş, diğer ise yarım gümüşlenmiştir.

Lazer ile Hırsız Alarmı Uygulaması

Transistörü tetiklemek için lazer ışığını kullanabileceğimiz gibi, basit bir kızıl ötesi led de kullanabiliriz. Kızıl ötesi led kullanmak istiyorsanız kendinize kızıl ötesi ışık saçan basit bir devre yapabilirsiniz. Transistörü tetiklemek için lazer kullanırsak neredeyse 100 metreden bile transistörü tetikleyebiliriz. Fakat kızıl ötesi led ile bu mesafe oldukça kısalacaktır. Piyasada kolaylıkla bulabileceğiniz lazer kalemleri ya da anahtarlıkları kullanarak, foto transistörü uzaktan tetikleyebilirsiniz.

Bu uygulamayı gerçekleştirebilmek için Bölüm 1'deki paralel port ile veri girişi uygulamasını ve STATUS portunu kullanmayı iyi anlamış olmalısınız. Ben buradaki uygulamamızda S6

pinini kullandım. Aşağıdaki program örneği de bu pinin adresleri için yazılmıştır. STATUS portundaki veriyi Visual Basic altında okumak için yine **Inp** komutunu kullanıyoruz.

Paralel portun S6 STATUS girişini kontrol eden Visual Basic Programı

```
Dim oku as Boolean,adres as integer
Private Sub Command1_Click()
   oku = False 'döngüyü durdurmaya yarayacak
End Sub
Private Sub Command2_Click()
adres= &h378 'paralel portunuzun taban adresi
oku = True
Dim okunan_deger As Integer
Do
   okunan_deger = Inp(adres + 1)
   ' STATUS portlarına dokunmadan okunacak değer 127 olmalıdır.
   a = DoEvents()
   If oku = False Then Exit Do ' Eğer oku butonuna basılırsa döngü duracaktır.
   If okunan_deger = 63 then
      MsgBox "Cisim tespit edildi",vbOkOnly+vbExclamation,"Uyarı"
      Exit Do
      'S6 pinini topraklադığımız zaman status portunun değeri 63 olacaktır.
End if
Loop
End Sub
```

Gerekli Malzemeler

1 adet BP109 (ya da uyumlu foto transistör)

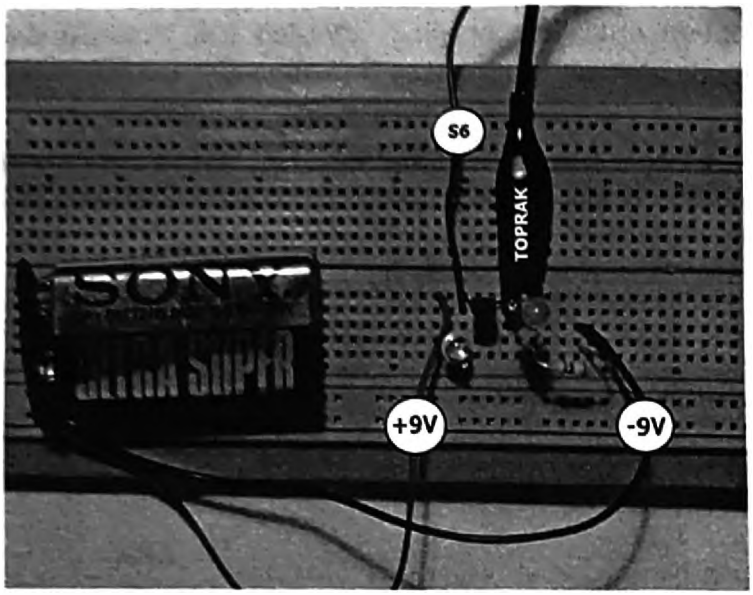
1 adet BC238C ya da BC547 transistör

1 adet kırmızı led

1 adet direnç 390 ohm 0.25W

Devrenin çalışma prensibi oldukça basit. Foto transistörü (Q1) ışık (lazer ya da kızıl ötesi) ile tetikliyoruz. Böylece foto transistör iletime geçiyor. Ardından da Q2 transistörü iletime geçiyor ve + 9 Volt normal ledin (+) anoduna akıyor. Yani buradaki NPN transistörü yükseltici amaçlı kullanmış oluyoruz. NPN transistör de iletime geçtiğinde S6 pininden gelen ve transistörün kollektörüne bağlı lojik voltaj da toprağa ulaşmış oluyor. Transistörün Emitörüne bağladığımız normal led ve direnç sayesinde, alıcı devre herhangi bir sinyal algıladığında, yani foto transistör iletime geçtiğinde bu led yanacaktır.

Evinizdeki televizyonun uzaktan kumandasını foto transistöre yakın bir şekilde tutarak herhangi bir tuşuna bastığınızda ledin yandığını ve transistörün iletime geçtiğini göreceksiniz. Devreyi tamamladığınızda, herhangi bir ışık göndermediğiniz halde, bu led hafif de olsa yanacaktır. Bunun sebebi, bulunduğunuz ortamın ışığıdır. Eğer karanlıkta denerseniz foto transistöre herhangi bir ışık gelmedikçe ledin hiç yanmadığını göreceksiniz. Ortam ışığına rağmen lazer ile yaptığınız tetiklemede ledin en parlak halini elde edeceksiniz. Led tam parlaklıkta yandığı zaman da S6 pini toprağa ulaşmış ve düşük olmuş demektir.



ŞEKİL 14.3: Devrenin board üzerinde kurulmuş hali.

ONBEŞİNCİ BÖLÜM

15

RADYO SİNYALİ İLE BEDAVA CHAT

Radyo Sinyali ve Mors Alfabesi ile İletişim

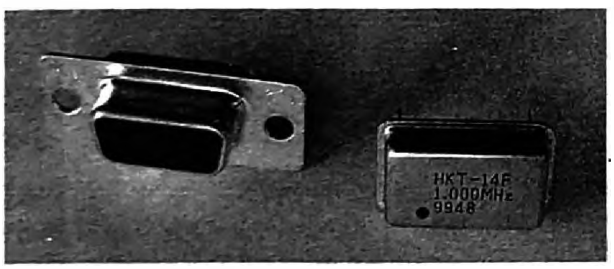
Arkadaşınızla, hiçbir ücret ödemeden, Internet ya da kablolu herhangi bir bağlantı kurmadan chat yapmak nasıl olurdu diye bir düşünün. Bu bölümde, bilgisayarın seri portunu kullanan basit bir AM radyo vericisi yapacağız. Bu verici sayesinde, ses kartından, AM çekebilen bir radyoya bağlı bir bilgisayara mesaj gönderebileceğiz. Bilgisayarda, klavyeden girilen karakterleri, radyo sinyaline çeviren ücretsiz bir program kullanacağız. Mesajları algılayacak olan bilgisayarda ise, bir ses kartı olması yeterli. Bu ses kartının Line girişine AM çekebilen bir radyo takacağız ve aldığı radyo sinyallerini karaktere çevirebilen, yine ücretsiz bir program kullanacağız.

Konuya girmeden önce, kaynak olarak kullandığım çok faydalı bir siteden bahsetmek istiyorum. Sitenin adresi `http://www.scitoys.com/`. Bu siteyi mutlaka ziyaret edin. Çok faydalı uygulamalar var, fakat çoğu elektronik tabanlı değil. Örneğin, evdeki eşyaları kullanarak, manyetik bir motor, radyo, güneş enerjisi uygulamaları gibi termodinamik, elektromanyetik, optik ve hatta biyolojik uygulamalar bile görebileceğiniz bir site.

Uygulamayı denemek için, ilk aşamada ikinci bir bilgisayara gereksinim yok. Ayrıca bilmeniz gerekiyor ki, karakterler radyo sinyali ile, Mors alfabesini kullanarak iletilmektedir. Biraz pratik yapmış ve Mors alfabesini bilen bir kişi, yakındaki bir radyodan, sinyalleri dinleyerek mesajlarınızı çözebilir. Tabii ki bu uygulama için Mors alfabesini ezbere bilmenize gerek yok. Zaten alıcı bilgisayara yükleyeceğimiz programın yaptığı işin esası, bu Mors karakterlerini algılayıp, bilgisayar ekranına yazdırmaktır.

Radyo Sinyali ile Chat Uygulaması

Uygulama için gereken malzemeleri şöyle sıralayabiliriz; Bir tane 1.000 Mhz kristal osilatör. Eğer alıcı olarak kullanılacak radyo destekliyorsa, başka frekans destekleyen osilatör de kullanabilirsiniz. Seri porta bağlantı için bir tane 9 pin RS232 seri konektör. Mors sinyallerini izlemek için bir led. Anten için bir miktar yalıtılmış kablo ve son olarak test için gerekecek birkaç tane krokodil kablo.



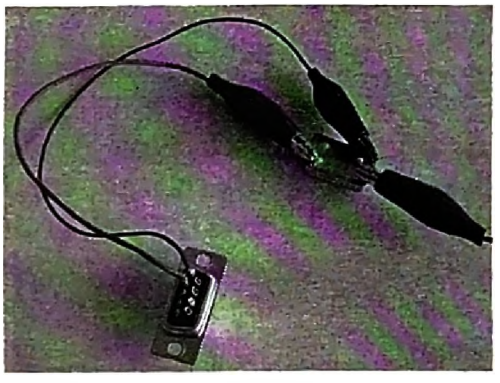
ŞEKİL 15.1: Malzemeler.

Eğer mesaj gönderdiğiniz bilgisayarın da size cevap yazmasını istiyorsanız, bu devreden bir tane daha yapmanız gerekiyor. İlk aşamada bir tane verici, devreyi test edebilmemiz için yeterli. İlk olarak, krokodil kabloyu ortadan ikiye keserek uçlarını soyun. Daha sonra Şekil 15.2'deki gibi birini 5'nci pine, diğerini de 4'ncü pine lehimleyin. 5 numaralı pine bağladığımız siyah kablo, portun toprak (GROUND) çıkışıdır. Bu kabloyu osilatörün toprak girişine bağlıyo-

ruz. Seri portun DTR çıkışı olan 4'ncü pini de osilatörün power pinine bağlıyoruz. Yeşil kablo ise antene bağlı. Anten yalıtılmış herhangi bir uzunca (20-30 cm) tel olabilir. Osilatörün keskin köşeli ucundaki çıkışı ise kullanmıyoruz. Sonuçta Şekil 15.3'teki gibi bir devre elde etmelisiniz. İlk denemeden sonra devreye sinyalleri izlemek için bir led ekleyebilirsiniz.



ŞEKİL 15.2: Seri portun sadece iki pinini kullanıyoruz.



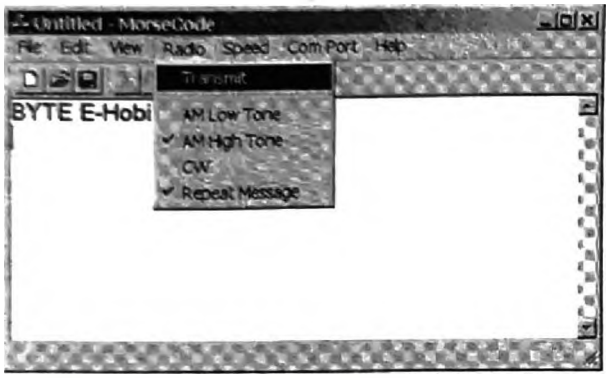
ŞEKİL 15.3: Kristal ile deneme yapmak için krokodil kablo kullanıyoruz.

Verici Kontrolü

Mesaj gönderebilmek için, yazdığımız karakterleri Mors koduna çevirecek ve osilatörü kısa ve uzun aralıklarla açıp kapayabilecek bir yazılıma ihtiyacımız var. Windows altında bu işi ücretsiz yapabilen bir programı `http://www.scitoys.com/scitoys/scitoys/radio/computer/MorseCodeExe.zip` adresinden indirebilirsiniz. Programı çalıştırdığınızda, Şekil 15.4'teki gibi bir ekran göreceksiniz. İlk başta Radio menüsünden "Repeat Message" seçeneğini aktif hale getirin. Bu seçenek sayesinde, mesaj vericiye peşi sıra sürekli gönderilir. Böylece AM radyo alıcısı ile sinyali yakalama şansımız kolaylaşacaktır. Eğer ilgileniyorsanız, bu programın C++ kodlarını, `http://www.scitoys.com/scitoys/scitoys/radio/computer/MorseCode.zip` adresinden indirebilirsiniz. Tüm bu dosyalar aynı zamanda kitabın kendi uygulama dosyalarının içerisinde de mevcuttur.

Sinyalleri yakalayabilmek için bir adet AM radyoya ihtiyacımız var. AM radyoyu 1.000 kilohertz frekansına ayarlamamız gerekiyor. Eğer kullandığınız radyonun dijital indikatörü varsa bunu ayarlaması daha kolay olacaktır. Eğer radyoda çevirmeli frekans ayarlayıcısı var ise, temiz

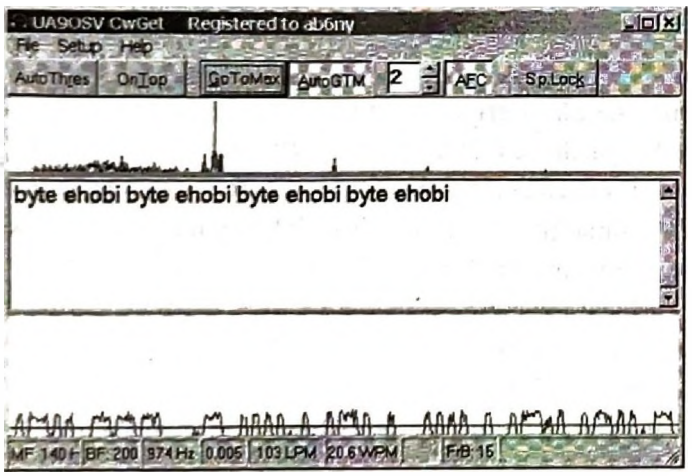
Mors kodlarını duyana kadar ince ayar yapmanız gerekecektir. Bunun için radyoyu antene yaklaştırmanız, daha net sinyal almanıza yardımcı olacaktır. Speed menüsünden ise, dakikada gönderilecek olan kelime sayısını ayarlayabilirsiniz. Ayrıca Radio menüsünden, "AM High Tone" seçeneğini işaretlemelisiniz, bu işlem sinyal tonunu 1000 Hertz'e göre ayarlayacaktır. Son olarak , Com Port menüsünden devreyi taktığınız portu seçin. Ekrana bir şeyler yazın ve Radio menüsünden Transmit'e basın. Şu anda mesajınız yayında. Sıra mesajı nasıl alacağımıza geldi.



ŞEKİL 15.4: Verici kontrol yazılımı.

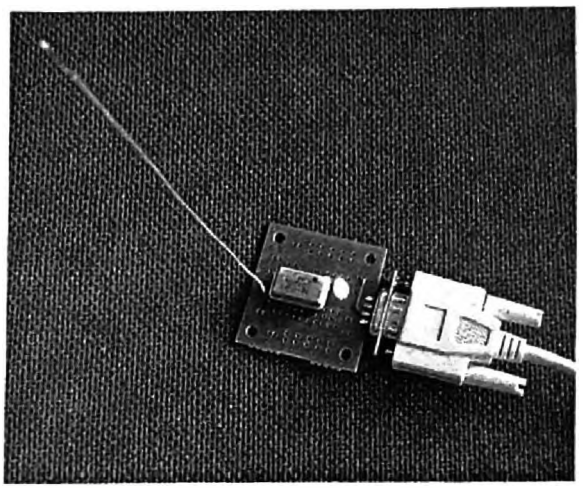
Alıcı Kontrolü

Bu aşamaya geçmeden önce, Mors seslerini radyodan kulağınızla dinleyebiliyor olmalısınız. Bu alfabeyi duyarak çözmek biraz zor olacağından, bunu bir yazılıma yaptıracağız. Internet üzerinde bu işi yapan birçok ücretsiz program var. Bunlardan birini `http://www.scitoys.com/scitoys/scitoys/radio/computer/cwget.zip` adresinden indirebilirsiniz. Bu programı kullanabilmek için, AM çekebilen radyoyu, ses kartınızın girişine takmalısınız. Daha başka bir şey yapmanıza gerek yok. Bu programın ekran görüntüsü ise Şekil 15.5'teki gibidir.

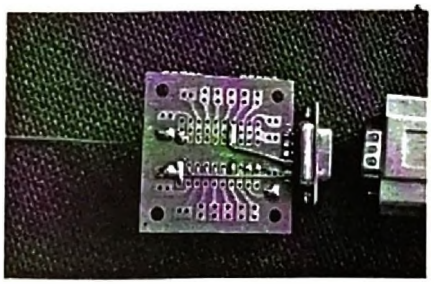


ŞEKİL 15.5: Alıcı kontrol yazılımı.

Şimdi devrenin nasıl çalıştığından bahsedelim. Sakın biz bu devreyi seri porta taktık diye, seri iletişim kullandığımızı zannetmeyin. Seri iletişim daha ciddi protokoller çerçevesinde gerçekleşir. Bizim burada yaptığımız, osilatörü seri portun DTR pininden beslemek. Kullandığımız programın yaptığı temel iş ise, portun DTR sinyalini, Mors alfabesine göre, uzun veya kısa aralıklarla açıp kapayarak osilatörün açılıp kapanmasını sağlamak. Verici program, sinyalleri gönderirken DTR pini ile birlikte RTS pinini de aktive eder. İsterseniz, bu pine bir led bağlayarak sinyalleri gözle izleyebilirsiniz. Devrenin bitmiş halini Şekil 15.6 ve Şekil 15.7'de görebilirsiniz.

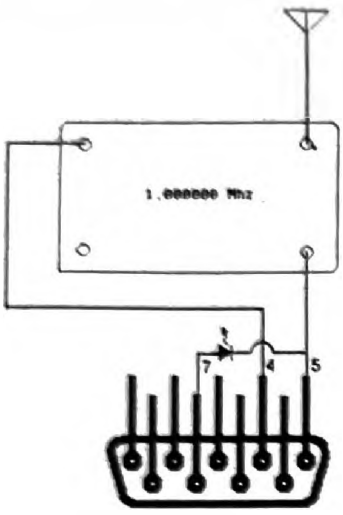


ŞEKİL 15.6: Devrenin tamamlanmış hali.



ŞEKİL 15.7: Devreyi bilgisayara seri kablo ile bağlayabilirsiniz.

Unutmayın ki, bu uygulama amatör amaçlı mesajlaşmalar için kullanılabilir. Profesyonel uygulamalar için çok sağlıklı olmayacaktır. Yani mesajlarınız kusursuz gidip gelecek diye bir durum söz konusu değildir. Radyo sinyalini ve iletişimi etkileyen çeşitli faktörler bazı aksaklıklara yol açabilir. Kullanılan yazılım ve donanım geliştirilerek daha kontrollü veri akışı sağlamak da mümkün olabilir.



ŞEKİL 15.8: Kristal ile seri portun bağlantı şeması.

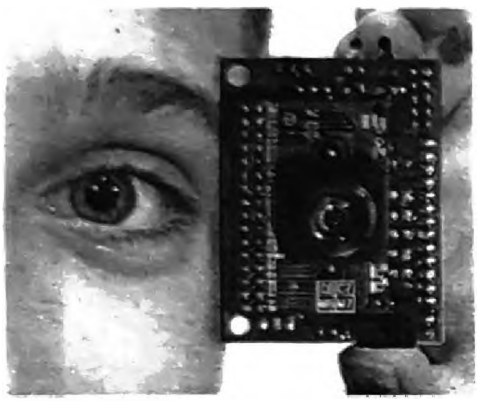
ONALTINCI BÖLÜM

16

CMUCam ELEKTRONİK GÖZ

Görüntü Algılama

İnsanların objeleri algılamasında ilk görevi şüphesiz insan gözü yapar. Bir insan gördüğü herhangi bir cismin ne olduğunu hemen tespit edebilir. Bunun nedeni, gözden alınan sinyallerin beyinde çok hızlı bir şekilde test edilmesidir. İnsan anatomisi bunu mükemmel bir şekilde yapabiliyor. Basitçe "görmek" dediğimiz bu işlev gerçekte taklit edilmesi çok zor ve karmaşık bir sistemdir. Elektronik cihazlarda, halk arasında göz diye tabir edilen şeyler ise genellikle foto transistörlerdir. Mesela, "Televizyon kumandasının gözü bozuldu" deriz. Foto transistörlerin gerçekte gözle pek alakaları yoktur. Yaptıkları temel şey, ışığın değişimine göre tepki vermektir. Sadece ışığı algılarlar. Elektronik olarak bu tepkiden yararlanmak çok kolaydır.



ŞEKİL 16.1: CMUCam.

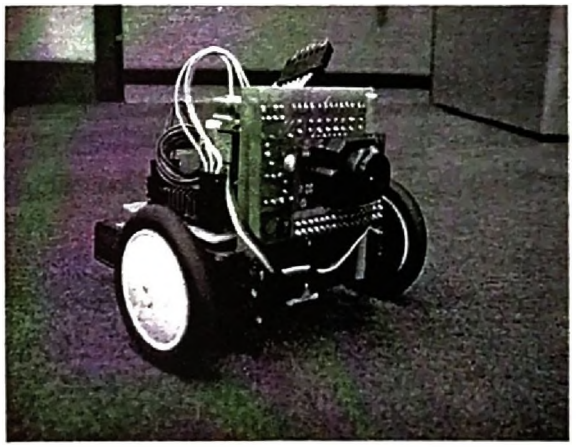
Karşılaşılacak cisimleri algılayabilmek, takip edebilmek için ilk olarak cismin görüntüsüne dijital olarak sahip olmak gerekir. Piyasadaki dijital kameralar, görüntüyü CCD'lere, oradan da dijital bir ortama aktarabilmektedirler. Burada inceleyeceğimiz ürün de gerçekte bir dijital kameradır. Fakat bu ürünün dijital kameradan birçok farkı var. Bir robot için dijital kamera ile görüntü elde etmek, o cismi tanımak anlamına gelmeyecektir. Karşılaşılacak bir cismin ne olduğunu anlamak çok karmaşık ve zor bir iştir. Bunun en büyük nedenlerinden bir tanesi, elde edilen görüntünün iki boyutlu olmasıdır. Gerçek hayattaki cisimler ise üç boyutludur. Buna rağmen, yüzünü gördüğü bir kişiyi tanıyabilen yazılımlar yapılmıştır. Basit bir robot için insan yüzü çok kompleks bir yapıdır. Bu nedenle, spesifik renkler kullanılarak robot gözlerine faydalı işler yaptırmak daha kolaydır.

CMUCam

CMUCam, renklere duyarlılık gösterebilen dijital bir kameradır. CMU harfleri, kamerayı geliştiren Carnegie Mellon Üniversitesi'nin baş harflerinden gelmektedir. Bu minik kamera ile saniyede 17 görüntü elde edebilirsiniz. Aynı zamanda renkli bir cismi takip edebilirsiniz. Düşük voltaj ile çalışabilen devre ile, elde ettiğiniz verileri seri port üzerinden bilgisayara ya da PIC, Basic-Stamp, BASIC-Tiger ve benzeri ürünlere aktarabilirsiniz. CMUCam'i kullanmak oldukça kolay bir iştir. Tek yapmanız gereken, seri port ile bağlantısını kurduktan sonra, komutları göndermek. Bunu herhangi bir terminal programı ile yapabilirsiniz. Örneğin bir fotoğraf çekmesini istiyorsanız, terminalden "DF" komutunu yazmalı ve ENTER'a basmalısınız. CMUCam görüntü

değerlerini aynı seri port aracılığı ile geri gönderecektir. Buradaki uygulamamızda, DF komutu ile kameradan elde ettiğimiz değerler ile, bilgisayar ekranında çekilen görüntüyü oluşturacağız. Ayrıca kamera içerisinden, örnek Java kaynakları çıkıyor. Devre şemalarını ve kullanımını anlatan dosyalar da mevcut.

Dijital bir resim, piksellerden oluşur. Her pikselin 0-255 arasında üç adet renk değeri vardır. Bunlar, R (Kırmızı), G(Yeşil) ve B(Mavi)'dir. RGB renk kodlamasına göre, tüm renkler bu üç rengin karışımı olarak elde edilir. Örneğin, RGB olarak 255,0,0 üçlüsü bize tam kırmızıyı verecektir. Bu rakamlar Hex değeri olarak ifade edildiğinde, HTML dilinde de kullanılan FF0000 şeklinde bir değer elde edilir. Ekrandaki görüntü, bu renklerden oluşan piksellerin birleşmesi ile meydana gelir. Monitörün çözünürlüğü olarak bilinen değer ise, bu piksellerden ekrana kaç tane sığdığı ile ilgilidir. Eğer 1024x768 çözünürlüğe sahipseniz, ekranınızda soldan sağa doğru 1024 sütun ve aşağıya doğru 768 tane satır piksel var demektir. Bu durumda, böyle bir çözünürlükte ekranda tam 786,432 adet piksel var demektir. Çözünürlük standardına göre, ekran genişliği ile uzunluğu arasında yaklaşık 1.3'lük bir katsayı olmalıdır. 1024:768=1.333... gibi. Örneğin 2048x1536 çözünürlükte fotoğraf çekebilen bir kamera 3,145,728 piksel saklayabilir demektir. Bu sayıya sahip bir makine de 3,1 Milyon megapixel şeklinde pazarlanmaktadır. Görüldüğü gibi zamanla milyon cinsinden verilen piksel sayısı bir çözünürlük standardı olmuştur.



ŞEKİL 16.2: CMUCam hareketli robotlar için ideal bir devre.

Dikkat ederseniz, dönüp dolaşıp kullandığımız değerler yine 0-255 arasındadır. Bu da 8 bit olarak ifade edilebileceğimiz bir değer anlamına gelmektedir. Hatırlarsanız 255 değerini "11111111" şeklinde ifade edebiliyorduk. Sonuçta herhangi bir dijital fotoğraf makinesi bu renk değerlerini, peşi sıra gönderdiği 1 ve 0 değerleri ile bilgisayara aktarmaktadır. Bu da seri iletişim protokolleri çerçevesinde gerçekleşir.

CMUCam'in Terminal Üzerinde Kullanımı

CMUCam üzerinde de bir adet seri port bulunmakta. Bu port ile kamerayı üzerinde seri port bulunan başka bir cihaza bağlayabilirsiniz. Biz doğrudan kamerayı bilgisayarın seri portuna bağlıyoruz. CMUCam 9,600-115,200 BPS hızlarında bağlantıyı desteklemektedir. Terminal programından kameraya bağlandığınız zaman, kamera bağlantıyı tespit ederek terminal

penceresine işleme hazır olduğuna dair bir mesaj gönderiyor. Aynı şekilde, herhangi bir komut sonrasında, kamera işlemin sonucunu aynı port üzerinden geri yolluyor.

DF Komutu ile Görüntü Yakalamak

Örneğin, "RS" komutu, kamerayı resetlemeye yarıyor. Terminalden komut gönderebilmek için komutun arkasından ENTER göndermeniz gerekiyor. "DF" komutu ise fotoğraf çekmeye yarıyor. Eğer terminal penceresinden "DF" komutunu gönderirseniz, CMUCam terminal penceresine yüzlerce acayip karakter gönderecektir. Hatta terminal programı belki kilitlenebilir. Bu karakterler, kameranın gönderdiği piksellerin renk değerleridir. Terminal programı ise bu değerleri karakter kodu olarak değerlendirdiği için saçma karakterler belirecektir. CMUCam 80x143 çözünürlüğe sahip ve verileri şu şekilde gönderir: Renk değerleri 16-240 arasındadır. "1" değeri, yeni bir fotoğraf, "2" değeri yeni kolon, "3" değeri de fotoğraf sonunu gösteriyor. Her piksel için 3 tane renk değeri gönderilir Her bir kolonda 143 piksel bulunmaktadır. Normal görüntü elde etmek için okunan her bir sütun 2 kez kullanılarak, çözünürlüğü 160x143'e çıkarmak gerekir. Aksi durumda, ince uzun bir görüntü elde edilecektir. Her bir sütun için peşi sıra, 143x3=429 renk değeri gönderilir. Her bir sütun bittiğinde ve yeni bir sütun başladığında ise "2" değeri gönderilir. Biz de programda, eğer "2" değeri gelirse, oluşturduğumuz resimde bulunduğumuz sütunu bir arttırarak devam edeceğiz.

CMUCam Kontrol Yazılımı

Programı Visual Basic ile yazacağız ve resmi oluşturmak için bir Picturebox kontrolünü kullanacağız. Kullanacağımız `Picture1.Pset` komutu ile, resim kutusunun içinde istediğimiz koordinata ait piksele renk verebiliriz. `Pset` komutu, `Picture1.Pset (x,y),renk` şeklinde çalışmaktadır. `renk` değişkeni RGB değerinin `long` tipi olarak karşılığıdır. Bu değeri Visual Basic altında `RGB(R,G,B)` fonksiyonu ile elde edebilirsiniz.

Her sütun için 429 değer olduğuna göre, resim için toplam 11440 tane renk kodu gönderilmesi gerekmektedir. Her piksel 1 Byte olsa, resmin tamamı 11 kilobyte'ı bulacaktır. Fakat seri iletişimde bu veriler sıkıştırılarak da gönderilebilmektedir. Bu nedenle Visual Basic içerisinde `Buffer` adında bir `Variant` değişkeni yaratıyoruz ve gelen pikselleri bu değişkene yüklüyoruz. `Variant` değişkenler bir indise sahiptir. `MSComm1.Input` komutu ile seri porttan gelen veri bu değişkene atılır. Port üzerinden ne kadar veri geldiğini, `UBound()` fonksiyonu ile algılayabiliriz. Bu fonksiyon bize bir indisin kaç tane veri içerdiğini döndürür. Daha sonra bu değer kadar çalışan bir döngü yazarız.

Sütunlar için `X`, Satırlar için `Y` değişkenini yaratıyoruz. `renk()` değişkenini ise gelen renk kodlarını ayırt etmek için kullanıyoruz. `renk(1)` kırmızı, `renk(2)` yeşil, `renk(3)` ise mavi için kullanılıyor. Bu döngü içerisinde, eğer gelen veri 1, 2 ya da 3 değilse, gelen veri bir piksele ait renk kodu olmalıdır. Bu nedenle, `Buffer` değişkeninden okunan değerleri, sıra ile `renk()` değişkenlerine aktarıyoruz. Eğer "2" gelirse sütun için yarattığımız `X` değişkenini 2 artırıyoruz. 1 artırmıyoruz, çünkü her sütunu iki kere basıyoruz.

Programı Visual Basic içerisinde örnek olarak gelen `VBTerm` programı üzerine yazdım. Buradaki uygulamayı gerçekleştirebilmek için CMUCam'e sahip olmalısınız. Böyle bir kamerayı durduk yerde gidip satın almanın bir anlamı yok elbette. Bu uygulamayı seçmemin nedenlerinden

biri de, görüntü oluşturmanın mantığını ortaya koyabilmek. Bu nedenle kod açısından daha fazla detaya inmiyorum. Kodu VBTerm'in `MSComm1.CommEvent` olayına eklemeniz yeterlidir.

CMUCam için Visual Basic kod örneği

```
Dim R as Byte,G as Byte,B as Byte
Dim i as Integer,s as Integer,
Dim X as Integer, Y as Integer,renk(3) as Integer,renk_long as Long
Case comEvReceive
      Dim Buffer as Variant
      Buffer=MSComm1.Input
         For i=0 to UBound(Buffer)
            If Buffer(i)="1" then
               Picture1.Cls
               Picture1.Refresh
            End If
            If Buffer="2" then
               X=X+2
               Y=0
            End If
            If Buffer="3" Then
               X=0:Y=0:S=0
               Renk(1)=0: Renk(2)=0: Renk(3)=0
               Msgbox("Resim bitti")
            End If
            If Buffer(i) <>"2" And Buffer(i) <>"1" And Buffer(i) <>"3" then
               s=s+1
               renk(s)=Int(Buffer(i))
               If s=3 then
                  S=0
                  Renk_long=RGB(renk(1),renk(2),renk(3))
                  Picture1.Pset (X,Y),renk_long
                  Picture1.Pset (X+1,Y),renk_long
                  Y=Y+1
               End If
            End If
         Next
   Case ...
   .
   .
   .
```



ŞEKİL 16.3: CMUCam gördüğü cisimlerin rengini algılayarak takip yapabiliyor.

CMUCam'in bir diğer özelliği de renk takibi yapabilmesidir. "TW" komutu ile kamerayı renk takibi moduna alabiliyorsunuz. Kamera bu moda geçtiğinde, pencerenin tam ortasındaki rengi alıyor ve seri porttan bu rengin koordinatlarını gönderiyor. Bu da yetmediği gibi, üzerinde bir tane de servo kontrol portu var. Buraya bağladığınız bir servo motora, rengin koordinatına göre hareket verebiliyor.

Bu kamerayı aynı zamanda Bölüm 28'de anlattığım MAR projesinde de kullandım. Kamera, Türkiye'de bildiğim kadarı ile satılmıyor. Ben yurt dışından `www.robotstore.com` adresinden sipariş ettim. Kamerayı pazarlayan başka siteler de mevcut. Kameranın fiyatı ise, $109. CMUCam robotik uygulamalar için çok kullanışlı. Daha detaylı bilgiyi `http://www-2.cs.cmu.edu/~cmucam/` adresinden elde edebilirsiniz.

İKİNCİ KISIM

II

PIC PROGRAMLAMA

ONYEDİNCİ BÖLÜM

17

PIC PROGRAMLAMAYA GİRİŞ

PIC Programlama

PIC programlama, kısaca anlatılabilecek bir konu değildir. Biz bu kısımda, popüler PIC uygulamalarını inceleyeceğiz ve her uygulamada yeni komutlar öğreneceğiz. Bu kısma geçmeden önce kitabın birinci kısmındaki tüm uygulamaların iyi bir şekilde kavranmış olması gerekmektedir.

PIC Nedir?

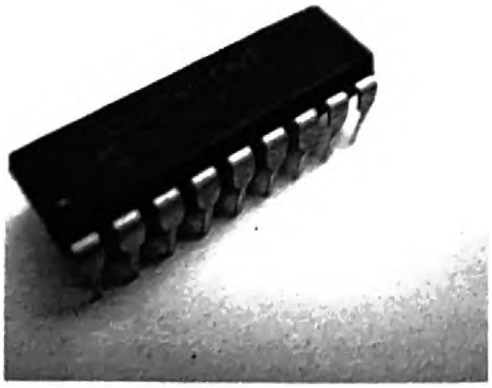
PIC, giriş çıkış elemanlarının denetimini çok hızlı olarak yapabilecek bir şekilde dizayn edilmiş bir çiptir. Örneğin, lamba, röle, motor, ısı ve benzeri sensörlerin denetimi bu çipler ile yapılabilir. Kısım 1'de geliştirdiğimiz uygulamaların çalışabilmesi için bir bilgisayara ve üzerindeki paralel ya da seri porta ihtiyaç duyuyorduk. PIC programlamayı öğrendikten sonra, bu uygulamaları bilgisayardan bağımsız bir hale getirebileceğiz. Bu çipler, RISC (Reduced Instruction Set Computer) mimarisi adı verilen bir yöntem kullanılarak üretildiklerinden, bu PIC'leri programlamak için gereken komutlar oldukça basit ve sayı olarak azdır.

Uygulamalarımızı, PIC 16F84 serisi ile gerçekleştireceğiz. Bu çip PIC16C84 olarak da bilinir. İkisinin de birbirinden hiçbir farkı yoktur. Bu çipi tercih etmemizin en büyük nedeni, Flash teknolojisi ile üretilmiş olmasıdır. Flash memory'ye yüklenen bir program, çipe uygulanan enerji kesilse bile silinmeyecektir. Flash bellekler, bu özellikleri ile EEPROM bellekler olarak da bilinirler. PIC'lerde bellek çeşitleri üç tiptir:

EPROM (Erasable PROgrammable Memory): Bu bellek tipine, elektrik sinyali ile kayıt yapılır. Silip yeniden başka bir program yazabilmek için mor ötesi ışık altında bir süre tutmak gerekir. Bu işlem EPROM silici denen cihazlarla yapılır.

EEPROG (Electrically Erasable PROgrammable Memory): Microchip firması bu bellek tipini FLASH bellek olarak da adlandırır. EPROM dan farkı, silme işleminin hızlı ve elektrik sinyali ile yapılmasıdır. Bizim kullanacağımız PIC 16F84 bu kategoriye girmektedir.

ROM (Read Only Memory): Maliyeti düşüktür, fakat sadece bir kez, fabrikasyon sırasında yazılabilirler. Kötü yanı, üretimden sonra chip programı içinde bir hata tespit edilirse, üretilen bütün çiplerin atılmak zorunda olmasıdır.

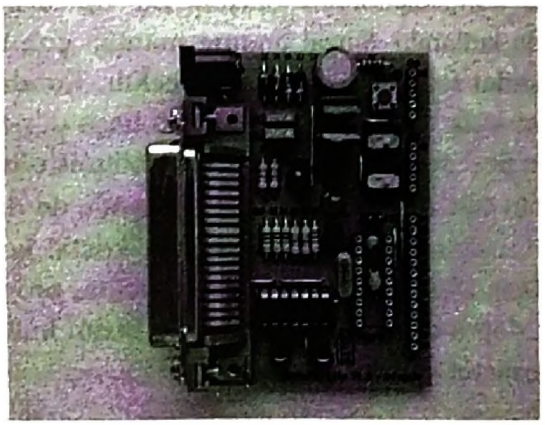


ŞEKİL 17.1: PIC16F84.

PIC 16F84 üzerindeki flash memory sayesinde, çip üzerine bir program yazdıktan sonra, herhangi bir silici cihaza gereksinim duymadan ve beklemeden, kolayca programı silmek ya da yeniden programlamak mümkündür. PIC'ler, PIC Assembly ile programlanır. Toplam 35 komuttan oluşur. Herhangi bir metin editörü ile kodları yazmak mümkündür. Bu kodları derlemek için Microchip firmasının ürettiği, `http://www.microchip.com` adresinden de ücretsiz olarak indirilebilen MPASM derleyicisi kullanılır. Bunun dışında, MPASM ile makine diline çevrilmiş olan bu program kodlarını, PIC'e yazmak için, bir PIC programlayıcı devresine ihtiyacımız vardır. Bu kartı, kendimiz yapabileceğimiz gibi, hazır olarak almak da mümkündür.

PIC Programlama ve Deneme Kartı

PIC'i çalıştırabilmek için, minimum gerekli bazı malzemeler vardır. En azından "OSC1" girişine bir saat sinyali uygulamak ve gerekli voltajı PIC'e sağlamak gerekir. Normalde PIC'i programladıktan sonra, programlayıcıdan söküp deneme devresine takmak gerekir. Bizim kullandığımız programlayıcı, bizi bu zahmetten kurtaracak düzeydedir.



ŞEKİL 17.2: Infogate firmasının ProtoPIC Lite PIC programlama ve deneme kartı.

Bu kitaptaki uygulamanın tümü ProtoPIC Lite Kartı ile gerçekleştirilmiştir. Bu kart aynı zamanda bir deneme kartıdır. PIC'i programladıktan sonra kendi üzerinde çalıştırabilir. Böylece her defasında PIC çalıştırmak için bir devre kurmamıza gerek kalmaz. Bu programlayıcı hakkında detaylı bilgiyi kitabın Kaynaklar bölümünden elde edebilirsiniz.

PIC16F84'ün Uçları

PIC16F84, çok az enerji harcar. Flash belleğe sahip olması nedeni ile, clock girişine uygulanan sinyal kesildiğinde register'ları içerisindeki veri bozulmadan durur. Sinyal tekrar verildiğinde, program kaldığı yerden devam eder. RA0-RA3 pinleri ile RB0-RB7 pinleri giriş çıkış portlarıdır. Aynı daha önceki paralel port uygulamalarımız gibi çalışır. Bu portlar ile dış ortama gönderilecek sinyaller, düşük olacaktır. Bu nedenle, sinyalleri güçlendirmek için daha önce kullandığımız, röle ve transistör gibi ek elemanlara ihtiyacımız olacaktır. Yine de bu enerji, aynı paralel porttaki gibi led yakmak için yeterlidir. Portların maksimum sink akımı 25mA, source akımı ise 20mA'dir.

Vss (5) ve Vdd (14) Uçları (Besleme)

Sink akımı gerilim kaynağından çıkış portuna doğru akan akıma, Source akımı ise GND (toprak) ucuna doğru akan akıma denir. PIC16F84'ün çektiği akım, besleme gerilimine, portlardaki yüke ve clock girişine uygulanan saat sinyalinin frekansına göre değişir. Besleme gerilimi +5 Volt olarak, 5 numaralı Vdd ucundan, toprak bağlantısı ise 14 numaralı Vss ucundan girilir. PIC'e ilk kez enerji verildiğinde, meydana gelebilecek gerilim dalgalanmaları nedeniyle oluşabilecek arızaları önlemek amacı ile, Vdd ile Vss arasına 0.1μf'lık bir dekuplaj kondansatörü bağlamakta fayda vardır.

OSC1/CLK IN (16) ve OSC2/CLK OUT (15) (Clock Giriş ve Çıkışları)

Bölüm 8'de saat (clock) sinyallerinden bahsetmiştik. Saat sinyali, peşi sıra gönderilen lojik "1" ve "0" darbelerinden oluşur. Bu darbelerin aralığı, saat sinyalinin frekansını belirler. Bu sinyali üretmek için osilatör devreleri kullanılır. Bu devrelerin en klasiği RC devreleridir. Bir RC devresi adından da anlaşılacağı gibi, bir direnç ve kondansatörden oluşur. Bu tip osilatörler, zamanlamanın çok önemli olmadığı durumlarda kullanılır. Sinyal üretme mantığı çok basittir. +5 Volt değeri, kondansatör doluma ulaşana kadar toprağa akmak isteyecektir. Kondansatör dolduğu anda gerilim OSC1 girişine akar. Kondansatör tekrar boşalır ve işlem başa döner. Böylece kondansatörün dolup boşalmasına dayanan peşi peşe bir darbe sinyali üretilmiş olur.

OSC1 ucuna uygulanan bu frekansın dörtte biri OSC2 ucundan çıkar. Bu sinyalle de diğer bir eleman sürülebilir. Saat sinyali üretmek için RC osilatör devrelerinin dışında kristal kontrollü osilatörler ya da seramik resonatörler kullanılabilir. Dikkat edilmesi gereken nokta, programlayıcı yazılımdan hangi tip osilatör kullanıldığının doğru olarak belirlenmesidir. Bizim kullandığımız programlayıcı kart üzerinde XT yani kristal osilatör vardır.

Giriş/Çıkış Portları (RA0-RA3 ve RB0-RB7)

PIC16F84'te 13 adet giriş/çıkış pini vardır. Bunlardan 8 tanesi B portu (RB0-RB7) diğer 5 tanesi de A portudur (RA0-RA4). Bu portlar giriş veya çıkış için programlanabilir. Bu portlardan geçebilecek sink ya da source akımları bir ledi sürmek için yeterli enerjiyi verecektir. Çıkış akımının az geldiği durumlarda yükseltici devreler kullanılabilir. B portunun 8 ucu sanki 50Kohm'luk dirençlerle pull-up yapılmış gibi etki gösterir. Bu port uçlarından bir tanesi çıkış olarak ayarlandığında, o uçtaki pull-up direnci iptal edilir. Bu kısımları uygulama sırasında daha rahat kavramak mümkün olacaktır.

Reset Devresi ve Ucu (4)

PIC16F84'e gerilim uygulandığında, programın başlangıç adresinden itibaren çalışmasını sağlayan bir reset devresi vardır. Terslenmiş olan MCLR (4) ucu ile, programın kullanıcı tarafından kesilip baştan başlatılması sağlanabilir. Bunun için MCLR ucuna +5V'luk gerilim uygulamak gerekir. Bu işlem bir buton ile yapılabilir.

ONSEKİZİNCİ BÖLÜM 18

LED KONTROLÜ

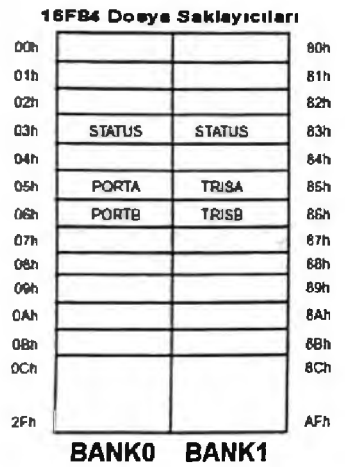
PIC Assembly

Micro işlemci programlamayı kavrayabilmek için, ilk olarak Assembly dilini anlayabilmek gerekir. PIC Assembly komutları, Tablo 18.1'de gözüktüğü gibidir. Bunların hepsini bir seferde kavramak mümkün olmayacaktır. Zamanla geliştireceğimiz uygulamalar ile, komutlara daha hakim olunabilecektir.

Yapacağımız uygulamalarda, istediğimiz pinleri giriş veya çıkış olarak ayarlayıp, girilen değerlere göre, çıkışlar elde edeceğiz. Örneğin, Kısım 1'de gerçekleştirdiğimiz paralel port uygulamalarımızı bilgisayardan bağımsız bir hale getirebileceğiz. Bu bölümde gerçekleştireceğimiz uygulamada PIC ile 3 tane ledi kontrol edeceğiz.

PIC ile Led Kontrol Uygulaması

Programımızı Assembly ile geliştireceğiz. Assembly, Visual Basic gibi hemen girip kodları yazıp, sonrada RUN'a basarak sonucu görebileceğiniz bir dil değildir. Sabırlı olmanızı tavsiye ediyorum. Assembly kodunu herhangi bir metin editöründe yazmanız mümkündür. Notepad, bu iş için yeterlidir. Tercihen PFE (Programmer's File Editor) bu iş için oldukça yaygın bir şekilde kullanılmaktadır. Yazdığımız kodu, `.asm` uzantısı ile kaydetmemiz gerekir. Ancak mikroişlemci, yazdığımız bu `asm` dosyasından pek bir şey anlamaz. PIC, bu kodu ancak Hex sayıları haline çevirebilirsek anlayacaktır. Bunun için bir Assembler derleyicisine ihtiyaç duyulur. Microchip firması, gereken bütün programları, metin editörü de dahil olmak üzere, Internet sitesinden ücretsiz olarak dağıtmaktadır. Kodumuzu derlemek için MPASM programını kullanacağız. Derledikten sonraki işlem ise, Hex formatına dönen bu dosyayı, PIC'e göndermek. Bölüm 17'de bahsettiğimiz bir PIC programlayıcısına ve yazılımına ihtiyacımız olacaktır.



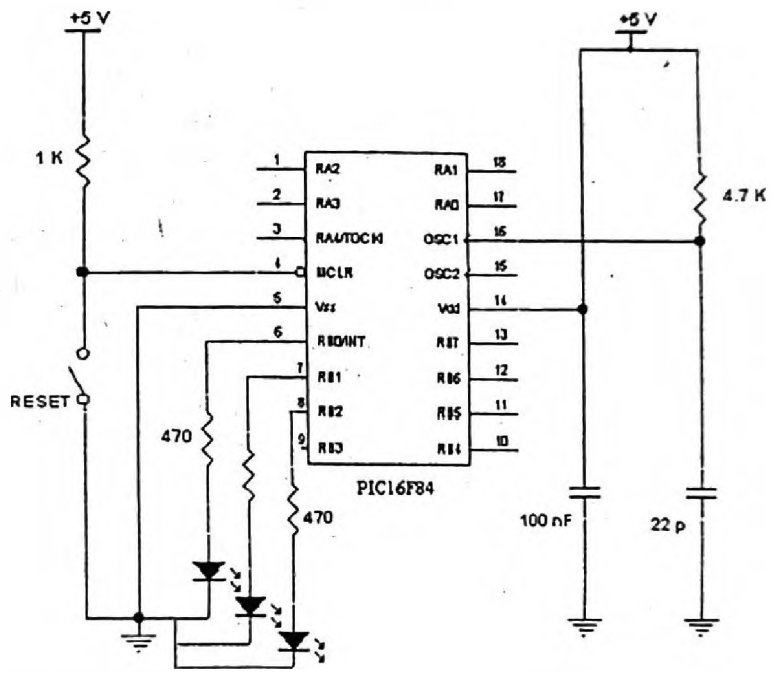
ŞEKİL 18.1: PIC16F84'ün dosya saklayıcı haritası.

Şekil 18.1'de, PIC16F84'ün dosya saklayıcı (file register) haritasını görüyorsunuz. Görüldüğü gibi Oku Yaz Bellek (RAM), iki Bank'tan meydana geliyor. Sol tarafta, kayıt adreslerini görüyorsunuz. Bu adresleri değişik komutlar ile kullanarak istediğimizi yaptırabileceğiz. Ko-

nuyu daha iyi irdeleyebilmek için örnek bir program yazarak ilk uygulamamıza başlayacağız ve kullandığımız komutları açıklayacağız.

Amacımız, PIC'in 3 tane pinini çıkış olarak ayarlayıp, bu 3 pine bağlı olan ledleri yakmak olsun. Pin olarak PORTB'nin 0, 1 ve 2 numaralı olanlarını kullanalım. Yapmamız gereken ilk şey, board üzerinde devremizi kurmak.

Normalde uygulamalarımızı, ProtoPIC programlayıcı kartının, deneme kartı özelliğini kullanarak gerçekleştireceğiz. Fakat yine de bu uygulama için öğrenmek amacıyla bir PIC'i çalıştırabilmek için gereken en basit devreyi board üzerinde kuracağız. Bu devrenin şemasını Şekil 18.2'de görebilirsiniz.



ŞEKİL 18.2: RC osilatör ile, 3 tane led bağlı PIC deneme devresi.

Daha önceki bölümde, PIC üzerindeki 6, 7, 8, 9, 10 numaralı pinlerin PORTB'ye 7, 18, 1, 2, 3 numaralı pinlerin de PORTA'ya ait olduğunu söylemiştik. PIC'i programlarken, ilk olarak hangi portların çıkış, hangilerinin giriş olduğunu tanımlamamız gerekecektir. PORTB'nin ilk üç pinini (6, 7, 8) çıkış olarak ayarlayalım. Bunun için, aynı paralel port uygulamalarımızda olduğu gibi 00000111 değerini bu porta gönderirsek, lojik "1"e denk gelen pinler +5 Volt olacaktır. Dolayısı ile bu pinlere bağlı olan ledler yanacaktır. Assembly programında verileri 16'lık (hexadecimal) olarak ya da binary olarak göndermek mümkündür.

KOD: 18.1: Assembly Kodu

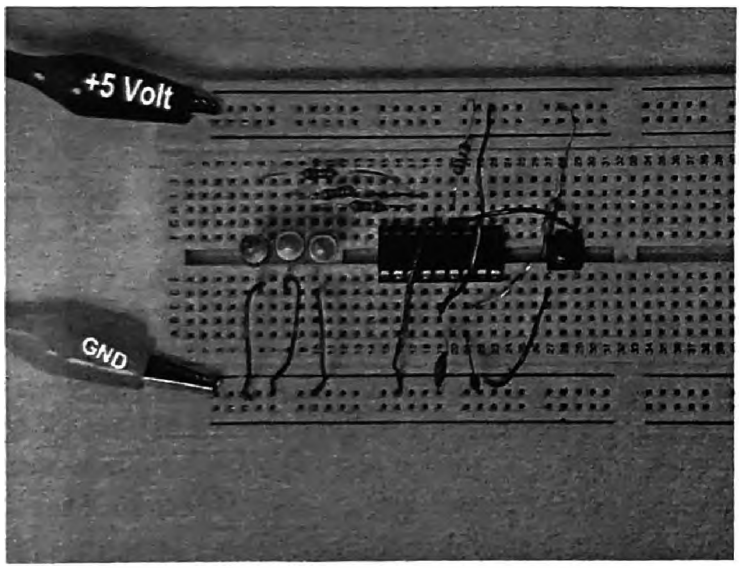
```
LIST  P=16F84  ;PIC16F84'ü assembler'a tanıt
PORTB EQU   0x06
START MOVLW 0x00
      TRIS  PORTB
```

```
        MOVLW 0x07
        MOVWF PORTB
END
```

Yukarıdaki kodu metin editörümüzde yazıp, **.asm** uzantısı ile kaydediyoruz. Şimdi programı satır satır inceleyerek olayı kavramaya çalışalım. İlk olarak dikkat etmeniz gereken, Assembly kodları 4 sütun şeklinde oluşur. TAB tuşu ile sütunlar arası atlamayı yapmalısınız. Birinci sütun etiketler için, İkinci sütun komutlar için, üçüncü sütun saklayıcılar ve adresleri için, en sonuncusu ise açıklama içindir. Dördüncü sütuna açıklama yazabilmek için başına ";" işareti eklenmelidir.

LIST P=16F84 : Bu satırda, PIC modelimizi Assembler'a tanıtıyoruz. **LIST** komutundan önce bir TAB boşluğu bıraktığımıza dikkat edin. Bu durumda assembler **LIST**'i bir komut olarak algılayacaktır. Daha sonra bir TAB tuşuna daha basıp, **P=16F84** yazıyoruz. Böylece bu veri 3. sütunda olduğundan, assembler bunu veri olarak algılayacaktır. Daha sonra bir TAB aralığı daha bırakarak, bir noktalı virgül koyup, istediğimiz açıklamaları ekleyebiliriz. Artık diğer satırları açıklarken, TAB boşluklarından bahsetmeyeceğiz. Sonuçta, **LIST** komutu, kullanılan mikroişlemcinin assembler'a tanıtılmasını sağlayan komuttur.

PORTB EQU 0x06: Bu satırda, değişken tanımlıyoruz. **EQU**, değişken tanımlama komutudur (tıpkı BASIC'te **a=6** der gibi). 0x06 değerini **PORTB** değişkenine atıyoruz. Bundan sonra, **0x06** değerini kullanmak istediğimiz yerlerde, **PORTB** yazarak programı daha kolay bir hale getirebiliyoruz.



ŞEKİL 18.3: RC osilatörlü PIC deneme devresinin board üzerinde kurulmuş hali.

RAM belleğin içerisinde, dosya saklayıcıları bulunur. Bu saklayıcılara yerleştirilen veriler, PIC işlemcisinin çalışmasını kontrol eder. PIC16F84'ün belleği iki banktan oluşur. Bir banktaki saklayıcıyı kullanmak için o banka geçmek gerekir. Şekilde gördüğünüz gibi, Bank 0'da 0x06 adresi PORTB'yi, 0x05 adresi ise PORTA'yı temsil eder. Bu nedenle 0x06 adresini **PORTB**

değişkenine eşitliyoruz. Bu adresi kullanabilmek için önce Bank0'a geçmemiz gerekiyor. İki satır sonra bu işlemi yapacağız zaten.

START MOVLW 0x00: Bu satırda **START** bir etiket adı. Programın buradan başladığını kendimize hatırlatmak için kullanıyoruz. Program içerisinde **GOTO** komutu ile tanıttığımız etiketlere atlama yaptırmamız mümkün. Biz burada herhangi bir atlama yapmayacağız. En önemli komutlardan biri olan **MOVLW** komutu, burada karşımıza çıkıyor. Burada anlatmak istediğimiz, **W** saklayıcısına **0x00** değerini göndermek. Burada **W** saklayıcısını biraz açıklamak gerekebilir.

İşlemci içerisindeki saklayıcıların hepsi, kullanıcıya açık değildir. Genelde bellekten veri okuma için akümülatör kullanılır. Bellekten işlemciye veri almak istersek bu durumda ilgili veriyi önce akümülatör içerisine alırız. Aynı şekilde, belleğe veri yazmak için de yazılacak veri önce akümülatöre, oradan da belleğe aktarılır. Ayrıca herhangi bir matematiksel ve lojik işlemin sonucunda ortaya çıkan matematiksel değer akümülatör içerisine aktarılacağından, bu tip işlemler sonucunda da akümülatörü kontrol etmeliyiz. Görüldüğü gibi aslında akümülatör en çok kullanılan saklayıcıdır. **W** saklayıcısı 16F84 işlemcisi üzerinde bulunan bir saklayıcıdır ve akümülatör gibi çalışır. Sonuçta akümülatör de, **W** saklayıcısı da, 8 bitlik veri depolayabilen saklayıcılardır. Burada biz "00000000" değerine karşılık gelen **0x00** değerini **W** saklayıcısına atıyoruz. Özet olarak, **MOVLW** komutu, **W** saklayıcısına doğrudan atama yapmak için kullanılır.

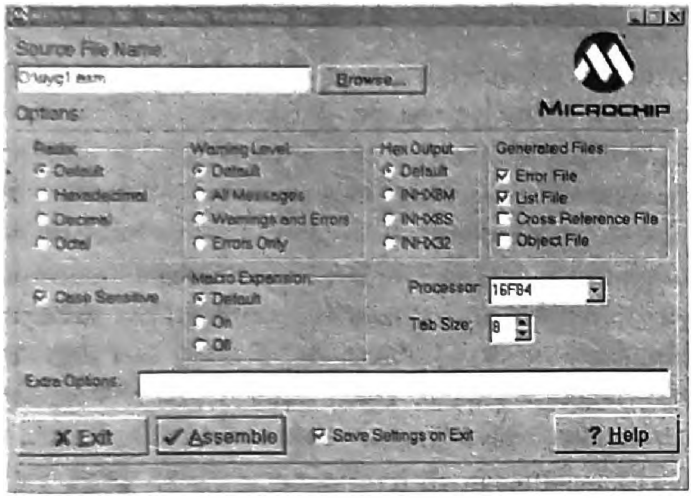
TRIS PORTB: Bir banktaki saklayıcıyı kullanabilmek için, önce o banka geçiş yapmamız gerektiğinden bahsetmiştik. Bunu yapmanın esas yolu, STATUS saklayıcısındaki, bank seçme bitlerini kullanmaktır. Biz şimdilik bunu yapmayacağız. Çünkü **TRIS** komutu bu işi bizim için otomatik olarak yapacaktır. **TRIS** komutu aslında, kullanılması pek tavsiye edilmeyen ve komut setlerinde geçmeyen bir komuttur. Buna rağmen, kullanımı kolaylaştırdığı için, birçok kaynakta bahsi geçer. Biz yine de **TRIS** komutu olmadan, nasıl programlama yapabileceğimizden de bahsedeceğiz. Sonuçta bu komut, **W** saklayıcısının içeriğini, PORTB yönlendiricisinin içeriğine aktarmaya yarar. Yönlendiricinin "0" bitlerine karşılık gelen pinler çıkış için, "1" bitlerine karşılık gelen pinler de giriş için kullanılır. Bu durumda, **W** saklayıcısına **0x00** değerini önceden yüklemiş olduğumuz için, yönlendirici de tüm pinleri çıkış olarak set edecektir. Dosya saklayıcısının şeklinde, yönlendiricileri de görebilmeniz mümkün (**TRISA**, **TRISB**). **TRIS** komutunu kullanmasaydık, önce Bank 1'e geçmemiz, daha sonra da 0x86'da bulunan **TRISB** yönlendiricisinin içeriğini sıfırlamamız, yani "00000000" değerini göndermemiz gerekiyordu. Bu durumda PortB'nin tüm uçları yine çıkış olacaktı. Bu işlem biraz daha uzun olduğundan, şimdilik **TRIS** komutunu kullanıyoruz.

MOVLW 0x07: Daha önce aynı komutu kullanmıştık. Bu sefer **W** saklayıcısına 0x07 değerini yüklüyoruz. Bu değer, "00000111" değerine karşılık geldiğinden, PORTB'deki lojik "1"e karşılık gelen ledleri yakacaktır. Bu da PIC üzerindeki 6, 7 ve 8'nci pinlere denk gelmektedir. Biz de ledlerimizi bu pinlere bağladığımızdan, ilgili ledler yanacaktır.

MOVWF PORTB: MOVWF komutu, **W** saklayıcısı içerisindeki veriyi, verilen **F** saklayıcısına gönderir. Biz burada **F** yerine PORTB değerini giriyoruz. Doğrudan PORTB üzerine bu veriyi gönderemeyiz. Önce **W** saklayıcısına göndermek zorundayız. Zaten az önce **W** saklayıcısına atamış olduğumuz veriyi PORTB'ye göndermiş oluyoruz.

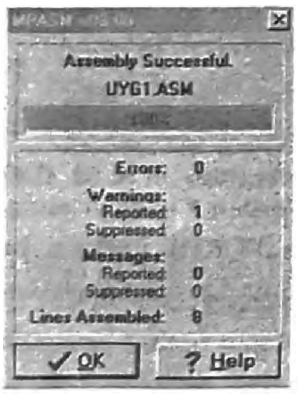
END: END komutu programın sonlandığını assemler'a bildiriyor.

Şimdi, sıra geldi `.asm` uzantısı ile kaydettiğimiz bu dosyayı derlemeye. Ben dosyayı D sürücümün altında, `uyg1.asm` olarak kaydettim. Bir derleyiciye ihtiyacımız olduğu açık. PIC16F84'ün üreticisi olan Microchip firmasının MPLAB yazılımına ihtiyacımız var. Bu programın son versiyonunu http://www.microchip.com adresinden, ücretsiz olarak adresinden elde etmeniz mümkün.



ŞEKİL 18.4: Microchip firmasının MPASM programı.

MPASM programını yükleyip açtığınızda, daha önce yarattığımız `.asm` dosyasını (`uyg1.asm`) bu programı kullanarak açıyoruz. Fakat dikkat etmeniz gereken bir husus, dosyanız 8 karakteri geçen klasörlerin altında olmasın. Doğrudan ana dizin üzerine kaydedebilirsiniz. Bu nedenle ben D sürücümün ana dizinine kaydettim. Dosyayı seçtikten sonra, işlemci listesinden 16F84'ü seçmeyi unutmayın. Daha sonra yapmanız gereken, Assemble düğmesini tıklamak. Daha sonra aşağıdaki gibi bir ekran ile karşılaşacaksınız. Bu ekran, derlemenin başarılı olduğunu gösteriyor. Fakat dikkat ederseniz 1 adet Warning mesajı aldık. Bunun sebebi, kullandığımız `TRIS` komutudur. Error ve Warning mesajlarını, `.asm` dosyanızın bulunduğu klasörde bulabilirsiniz. Program, `uyg1.err` şeklinde bir dosya yaratacak ve bu dosyanın içerisinde, "`Warning[224] D:\UYG1.ASM 4 : Use of this instruction is not recommended.`" mesajı yazacaktır. Bu mesajı şimdilik dikkate almayın. Zira bu bir hata mesajı olmayıp bir uyarı dosyasıdır.

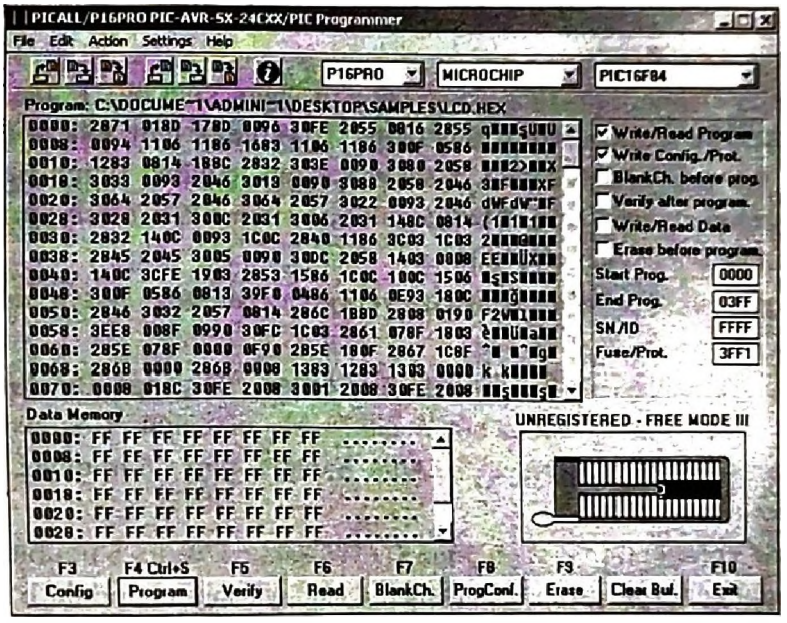


ŞEKİL 18.5: MPASM ile ASM dosyasının derlenmesi.

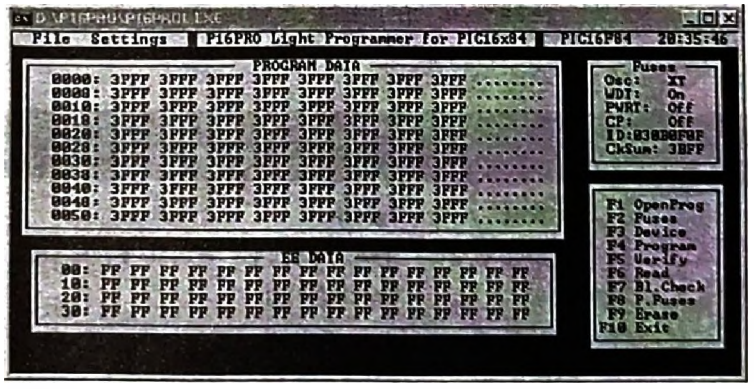
Derleme işlemi, PIC'in anlayacağı bir format olan **.hex** dosyamızı aynı klasörde yaratmış olmalı. Bu dosyanın içeriği aşağıdaki gibidir;

```
:020000040000FA
:0800000000306600073086 00A5
:00000001FF
```

Sıradaki işlem, bu **.hex** dosyasını, bir PIC programlayıcısı yardımı ile, PIC üzerine yüklemek. Bunun için programlayıcı kartınızı destekleyen bir yazılıma ihtiyacınız var. Programlayıcımız, DOS tabanlı P16PRO yazılımını ya da **http://www.picallw.com** sitesindeki Windows tabanlı PicAll yazılımını desteklemektedir. Bu yazılımları sitemizden indirebilirsiniz. Bu konu ile ilgili detaylı bilgi Kaynaklar bölümünde belirtilmiştir.



ŞEKİL 18.6: PICALLW Windows tabanlı PIC programlayıcı yazılımı.



ŞEKİL 18.7: P16PRO DOS tabanlı PIC programlayıcı yazılımı.

Şimdi, eğer **TRIS** komutunu kullanmasaydık nasıl yazmamız gerektiğine bakalım.

KOD: 18.2: Assembly Kodu

```
LIST    P=16F84
PORTB   EQU   0x06
STATUS  EQU   0x03
START   MOVLW 0x00  ;Kullanılmıyor*
        BSF   STATUS,5
        CLRF  0x86
        BCF   STATUS,5
        MOVLW 0x07
        MOVWF PORTB
END
```

İlk kodumuzdan farklı olarak, **BSF**, **CLRF** ve **BCF** komutlarını kullandık. Biz bu üç komut yerine **TRIS** komutunu kullanmıştık. Şimdi bu üç komutu açıklayıp, **TRIS** komutunun ne iş yaptığını daha iyi anlayalım.

BSF STATUS,5: **STATUS** saklayıcısının Bank değiştirme biti 5'tir. Eğer 5'nci bit "1" ise, Bank1 seçilir, "0" ise Bank0 seçilir. **STATUS** saklayıcısının 5'nci bitini "1" yapmak için, **BSF** komutunu kullanıyoruz. Bu komut **BSF f,b** şeklinde kullanılır. **f** saklayıcısının içindeki b'inci bitini "1" yapmaya yarar. Bizim kullandığımız şekle göre, **STATUS** saklayıcısının 5'nci bitini yani Bank seçme bitini "1" yapar. Dolayısı ile Bank1 seçilmiş olur.

Artık Bank 1'i seçtiğimize göre 0x86 adresli saklayıcıya veri gönderebiliriz. Dosya saklayıcıları tablosuna bir göz atın.

CLRF 0x86: **CLRF f** komutu, **f** saklayıcısı içerisindeki tüm bitleri "0" yapar. Bu durumda, 0x86 adresine karşılık gelen **TRISB** saklayıcısının bütün bitleri "0" olacaktır. Dolayısı ile, B portunun tüm uçları çıkış olarak set edilmiş olacaktır. Artık Bank0'a geri dönmeliyiz, çünkü PortB (0x06) o bankta yer alıyor.

BCF STATUS,5: Yukarıda **BSF** komutu ile "1" yaptığımız **STATUS** saklayıcısının 5'nci bitini tekrar "0" yapıyoruz. Yani bu komut **BCF f,b** şeklinde kullanılır. **f** saklayıcısının içindeki b'inci bitini "0" yapmaya yarar diyebiliriz. Artık yeniden Bank 0'dayız.

Buradan sonrasındaki akış, yukarıdaki ile aynıdır. Bir de **START MOVLW 0x00** satırı kullanılmıyor. Bu satırı silseniz de olabilir. Daha önceki **TRIS**'li kodumuzda, **W** saklayıcısı içerisindeki veriyi kullanıyorduk çünkü.

TABLO 18.1: PIC16F84 Komut Seti

Komut	İşlem	Komut	İşlem
ADDWF f,d	d:=W+f	SWAPF f,d	d:= f[4..7] <--> f[0..3] (Yerlerini değiştirir)
ANDWF f,d	d:=W AND f	XORWF f,d	d:= W XOR f
CLRF f	f:=0	BCF f,b	f[b]:=0 (f = Register b = bit)
CLRW	W:=0	BSF f,b	f[b]:=1 (f = Register b = bit)
COMF f,d	d:=NOT(f) (Tersini Alır)	BTFSC f,b	test bit f = register b = bit (Bit sıfır mı?)

TABLO 18.1: PIC16F84 Komut Seti

Komut	İşlem	Komut	İşlem
DECF f,d	d:=f-1	BTFSS f,b	test bit f = register b = bit (Bit bir mi?)
DECFSZ f,d	d:=f-1 (Sıfır mı? diye sorar.)	ADDLW k	W:=W+k
INCF f,d	d:=f+1	ANDLW k	W:=W AND k
INCFSZ f,d	d:=f+1 (Sıfır mı? diye sorar.)	CALL k	k alt programını çağırır.
IORWF f,d	d:=W OR f	CLRWDT	WDT'i siler.
MOVF f,d	d:=f (d = 0 ise sonuç W register'ına, d = 1 ise sonuç tekrar f'e yüklenir.)	GOTO k	k etiketine dallan.
MOVWF f	W:=f	IORLW k	W:=W OR k
NOP	Bir cycle bekletme komutu.	MOVLW k	W:=k
RLF f,d	d=f SHR 1	RETFIE	Interrupt alt programından geri dön.
RRF f,d	d=f SHR 1	RETLW k	w:=k, ve alt programdan geri dön
SLEEP	Uyku moduna geç.	RETURN	Alt programdan geri dön.
SUBLW k	W:=W-k	XORLW k	W:=W XOR k

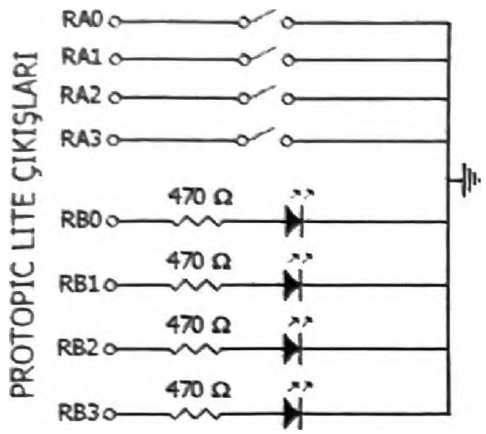
ONDOKUZUNCU BÖLÜM

19

BUTONA BASINCA SÖNEN LED

Butona Basınca Sönen Led Uygulaması

Bu bölümdeki uygulamamız biraz daha gelişmiş sayılır. Bir butona basınca 4 ledden bir tanesini söndüren bir uygulamamız olacak. Önce programımızın adım adım ne yapacağını gösteren bir akış şeması çizeceğiz. Daha sonra, bu şemayı assembly koduna dönüştüreceğiz ve PIC üzerine yükleyeceğiz. Devremiz, PORTA'ya bağlı 4 tane buton ve PORTB'ye bağlı 4 tane ledden oluşuyor. Dirençler, ledleri yüksek akımdan koruyor. Butonlar ise doğrudan toprağa bağlı.



ŞEKİL 19.1: Deneme kartı ile led ve butonların bağlantısı.

Amacımız, PORTA'ya bağlı dört tane butondan bir tanesine bastığımızda, PORTB'de ilgili butona karşılık gelen çıkışta ki ledi söndürmek olsun. Böylece basit bir giriş-çıkış uygulaması gerçekleştirmiş olacağız. İlk olarak PORTA'yı giriş, PORTB'yi de çıkış olarak tanımlamamız gerekiyor. Bunun nasıl yapıldığını Bölüm 18'de anlatmıştık. Yine de kısaca değineceğiz.

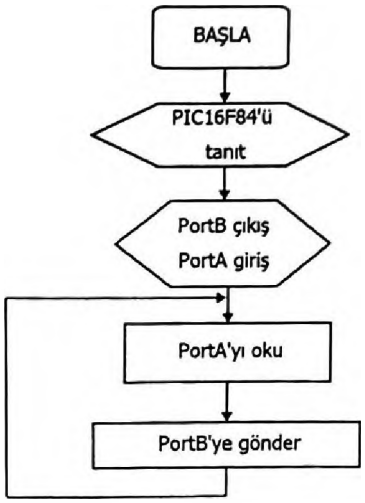
Yapacağımız işlem, PORTA'daki değerlerin aynısını sürekli PORTB'ye göndermek olacak. PORTA'yı giriş olarak tanımladığımız için, bu porta ait pinler Pull-up durumundadır. Eğer bu durumdaki pinlerden herhangi bir tanesini toprağa gönderirsek, ilgili pine ait lojik değer "0" olacaktır. Dikkat ederseniz, girişi "1" verisi ile değil "0" verisi ile yapmış oluyoruz. Aynı daha önce incelediğimiz paralel portun STATUS pinleri gibi. Dolayısı ile, butona bağlı olan pinin lojik değeri "0" oluyor. A portunun değerlerini sürekli B portuna yazdığımız için, B portunda ilgili bite karşılık gelen pinin lojik değeri de "0" olacaktır. Bu durumda, ilgili pine bağlı olan led sönecektir. Programın akışı, Şekil 19.2'deki gibidir.

Aşağıdaki kodların ne anlama geldiğini, önceki bölümde açıklamıştık. Bu sebeple burada daha kısa değineceğiz.

KOD: 19.1: Assembly Kodu

```
LIST    P=16F84
PORTA   EQU   0x05
PORTB   EQU   0x06
STATUS  EQU   0x03
        BSF   STATUS,5
        CLRF  0x86
```

```
            MOVLW 0xFF
            MOVWF 0x85
            BCF   STATUS,5
DONGU
            MOVF  PORTA,W
            MOVWF PORTB
SON
            GOTO  DONGU
END
```



ŞEKİL 19.2: Programın akışı.

LIST P=16F84: Bu satırda, PIC modelimizi assembler'a tanıtıyoruz.

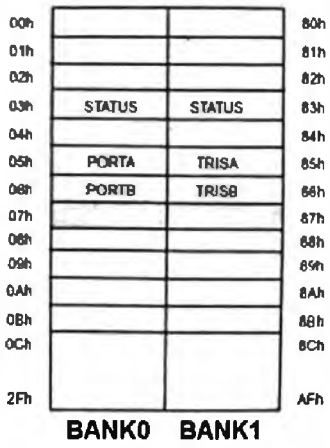
PORTA EQU 0x05: Bu satırda, değişken tanımlıyoruz. **EQU**, değişken tanımlama komutudur. 0x05 değerini **PORTB** değişkenine atıyoruz. Bundan sonra, 0x05 değerini kullanmak istediğimiz yerlerde, **PORTB** yazarak programı daha kolay anlaşılabilir bir hale getirebiliyoruz. Şekil 19.3'te gördüğünüz gibi, Bank 0'da 0x06 adresi PORTB'yi, 0x05 adresi ise PORTA'yı temsil eder. Bu nedenle 0x06 adresini **PORTB** değişkenine eşitliyoruz. Bu adresi kullanabilmek için önce Bank 0'a geçmemiz gerekiyor.Bundan sonraki iki satır ise diğer değişkenleri tanımlamaya yarıyor.

BSF STATUS,5: **STATUS** saklayıcısının Bank değiştirme biti 5'tir. Eğer 5'nci bit "1" ise, Bank1 seçilir, "0" ise Bank0 seçilir. **STATUS** saklayıcısının 5'nci bitini "1" yapmak için, **BSF** komutunu kullanıyoruz. Bu komut **BSF f,b** şeklinde kullanılır. **f** saklayıcısının içindeki **b**'inci bitini "1" yapmaya yarar. Bizim kullandığımız şekle göre, **STATUS** saklayıcısının 5'nci bitini yani Bank seçme bitini "1" yapar. Dolayısı ile Bank1 seçilmiş olur. Artık Bank1'i seçtiğimize göre 0x86 adresli saklayıcıya veri gönderebiliriz.

CLRF 0x86: **CLRF f** komutu, **f** saklayıcısı içerisindeki tüm bitleri "0" yapar. Bu durumda, 0x86 adresine karşılık gelen **TRISB** saklayıcısının bütün bitleri "0" olacaktır. Dolayısı ile, B portunun tüm uçları çıkış olarak set edilmiş olacaktır.

MOVLW 0xFF: Daha önce aynı komutu kullanmıştık. Bu sefer **W** saklayıcısına 0xFF değerini yüklüyoruz. Bu değer, "11111111" değerine karşılık geliyor. Biz bu değeri bir sonraki komut ile 0x85 adresine gönderiyoruz. Bu da Şekil 19.3'te gözüktüğü gibi **TRISA** yönlendiricisine karşılık

geliyor. Yönlendiricinin "0" bitlerine karşılık gelen pinleri çıkış için, "1" bitlerine karşılık gelen pinleri de giriş için kullanılır. W saklayıcısı, işlemci içerisinde, kullanıcıya açık bir saklayıcıdır. Bellekten işlemciye veri almak için önce, veri W saklayıcısına, oradan da belleğe alınır. W saklayıcısından geçen bölümde bahsetmiştik.



ŞEKİL 19.3: PIC16F84 dosya saklayıcıları.

MOVWF 0x85: MOVWF f komutu, W saklayıcısı içerisindeki veriyi, verilen f saklayıcısına gönderir. Biz burada f yerine 0x85 değerini giriyoruz. Doğrudan 0x85 üzerine bu veriyi gönderemeyiz. Önce W saklayıcısına göndermek zorundayız. Az önce W saklayıcısına atamış olduğumuz veriyi 0x85'e göndermiş oluyoruz. Bu durumda PORTA'nın tüm uçları giriş olarak ayarlanmış oluyor.

BCF STATUS,5: Yukarıda **BSF** komutu ile "1" yaptığımız **STATUS** saklayıcısının 5'nci bitini, tekrar "0" yapıyoruz. Yani bu komut **BCF** f,b şeklinde kullanılır. f saklayıcısının içindeki b'inci bitini "0" yapmaya yarar diyebiliriz. Artık yeniden Bank0'dayız.

DONGU: Bu satırı bir etiket olarak kullanıyoruz. İleride **GOTO** komutu ile bu satıra atlama yaptıracağız.

MOVF PORTA,W: MOVF f,d komutu, d yerine W yazarsak, sonucu W saklayıcısına, F yazarsak kendi üzerine yani F saklayıcısına yükler. Burada PORTA'nın içeriğini W saklayıcısına yüklemiş oluyoruz.

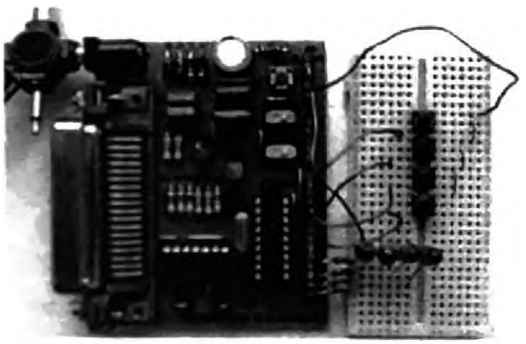
MOVWF PORTB: MOVWF komutu, W saklayıcısı içerisindeki veriyi, verilen adrese gönderir. Biz burada adres yerine **PORTB** değerini giriyoruz. Bu durumda daha önceki satırda PORTA'nın içerisindeki veriyi W saklayıcısına yazmış olduğumuzdan, burada W'nun içindeki değeri PORTB'ye yazmış oluyoruz. Dolayısı ile PORTA'nın içeriğini aynen PORTB'ye aktarıyoruz.

GOTO DONGU: Daha önce tanıttığımız **DONGU** etiketine atlama yapıyoruz. Böylece program sürekli bir döngüye giriyor ve A portundaki değerleri sürekli B portuna yazıyor.

Örneğin butonlardan 1'ncisi ve 3'süne aynı anda basalım ve programın ne yaptığını inceleyelim. Hiçbir butona basılmadığı zaman PORTA'dan okunan değer "11111111"dir. 1'nci ve 3'ncü butonlara aynı anda basınca, PORTA üzerindeki RA0 ve RA2 çıkışları topraklanmış olacak. PIC üzerinde PORTA'ya ait 5 adet pin olduğundan son 3 bit bizi ilgilendirmeyecek. Bizi ilgilendirmeyen pinleri "x" ile gösterelim. Bu durumda, PORTA giriş için ayarlanmış olduğundan,

hiçbir butona basılmadığı zamanki durumunu "xxx11111" olarak gösterelim. Biz 1'nci ve 3'ncü butonlara bastığımız zaman, o pinleri toprakladığımız için PORTA'daki 1'nci ve 3'ncü bitin değerleri "0" olacaktır. Bu durumda PORTA'nın son değeri "xxx11010" olarak gösterilebilir. Bu veri, olduğu gibi PORTB'ye aktarıldığından, PORTB'de, lojik "0"a denk gelen pinlere bağlı ledler sönecektir.

Sıra geldi programımızı derlemeye. Geçen bölümde MPASM'nin kullanımından kısaca bahsetmiştik. Ben yukarıdaki programı Notepad ile hazırlayıp, uzantısını `.asm` olarak kaydettim. Sizde aynı şekilde dosyayı `.asm` uzantısı ile kaydedip MPASM programında derleyebilirsiniz. Bu aşama hakkında geçen bölüm daha detaylı bilgi vermiştik. MPASM programı, `.asm` dosyasını `.hex` uzantılı bir dosya olarak derleyecek. Bu dosyayı P16PRO Light programı ile PIC üzerine göndereceğiz.



ŞEKİL 19.4: Devrenin board üzerinde kurulmuş hali.

YİRMİNCİ BÖLÜM

20

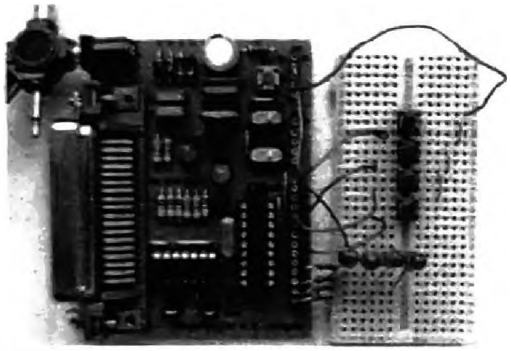
SIRALI YANAN LED'LER

Sıralı Yanan Led Uygulaması

Bu bölümde temel olarak öğreneceğimiz şey, gecikme alt programı. Hatırlarsanız Kısım I'de, bilgisayarımızın paralel portunu kullanarak 8 tane ledi sıra ile yakıp söndürmüştük. Burada yapacağımız, aynı uygulamanın 4 ledli olanı. Programı anladıktan sonra led sayısını arttırmanız çok kolay gelecektir.

Paralel port aracılığı ile bu uygulamayı gerçekleştirirken, porta sıra ile "00000001", "00000010"... şeklinde, lojik "1" değerlerini sola kaydırarak gönderiyorduk. Sonuçta, portun lojik "1" değerine karşılık gelen pinlere bağlı olan ledler yanıyordu. Hatırlarsanız, bu sinyalleri sıra ile uygularken, araya bir gecikme fonksiyonu yazmıştık. Basic ile, içi boş bir **FOR-NEXT** döngüsü yaratıp, bir sonraki ledi yakana kadar birkaç saniyelik bir bekletme sağlamıştık. Bunun nedeni, portun çok hızlı olması ve ledler sıra ile yanarken, işlem çok hızlı olacağından, sanki hepsi birlikte yanıyormuş izlenimi vermesi idi.

Aynı durum, PIC için de geçerli. Zaten bugüne kadar paralel port ile uyguladığımız her şeyi PIC ile bilgisayardan bağımsız olarak uygulayabiliriz. Sonuçta paralel portun üzerindeki 8 bitlik DATA portunu kullanıyorduk. PIC üzerinde de 8 bitlik B portu, aynı işi görecektir. Üstelik, bu portu giriş ya da çıkış olarak programlayabiliriz.



ŞEKİL 20.1: Bölüm 19'daki deneme devremizi kullanıyoruz.

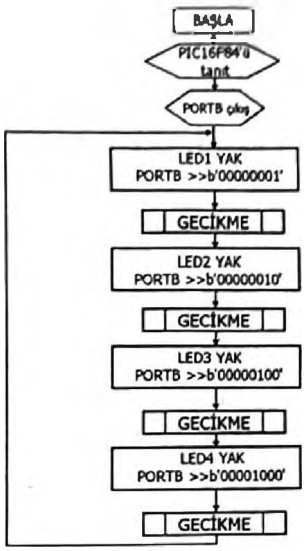
Bu uygulamamızda amacımız 4 tane ledi sıra yakıp söndürmek. Bu uygulamada, alt programları nasıl çağırabileceğimizi ve geciktirme işleminin nasıl yapılabileceğini öğreneceğiz. Kullandığımız komutların çoğu geçen bölümlerdeki ile aynı. Bu sebeple, geçen iki bölümde bahsettiğimiz PIC komutlarını sadece satırın yanına açıklamasını yazarak geçiyorum.

Gecikme Nedir? Nasıl Yapılır?

Programlarımız içerisinde geciktirme alt programlarına ihtiyaç duyabiliriz. Örneğin şimdi gerçekleştireceğimiz led uygulamasında, PORTB'ye sıra ile lojik sinyaller göndereceğiz. Eğer bu sinyaller arasında bir gecikme alt programı kullanmazsak, ledler sıra ile o kadar hızla yanıp söner ki, hepsi aynı anda sürekli yanıyormuş gibi gözükür. Bunun sebebi, PIC içerisinde işlemlerin milisaniyeler mertebesinde yapılmasıdır.

En klasik bekletme komutu **NOP**'tur. Fakat bu komut sadece bir döngü bekletme sağlar. Bu komuttan yüzlerce alt alta yazmak bize ancak saniye mertebesinde bir bekletme sağlar. Aynı zamanda yüzlerce **NOP** komutu belleği gereksiz dolduracaktır. Bu sebeple, kullanılan

yöntemlerden biri, bir döngü içerisinde **NOP** komutunu kullanmaktır. Bir diğer yöntem de sayaç olarak kullanacağımız bir saklayıcı belirlemek ve bu sayacın değerini bir döngü içerisinde sıfır olana kadar sürekli azaltmaktır. Biz programımızda bu çeşit bir döngü kullanıyoruz.



ŞEKİL 20.2: Programın akış diyagramı.

KOD: 20.1: Assembly Kodu

```
LIST     P=16F84         ; PIC modelimizi assembler'a tanıtıyoruz.
PORTB    EQU   0x06      ; PORTB isimli değişkene, 0x06 değerini yüklüyoruz.
STATUS   EQU   0x03      ; STATUS isimli değişkene, 0x03 değerini yüklüyoruz.
         BSF   STATUS,5 ; Bank1 e geçiyoruz. (5=Bank değiştirme biti)
         CLRF  0x86      ; PortB'nin tüm uçlarını çıkış yapıyoruz.
         BCF   STATUS,5 ; Bank0 a geri dönüyoruz.
         CLRF  PORTB     ; PortB'nin tüm uçlarını "0" yapıyoruz.
BASLA
         MOVLW b'00000001' ; W saklayıcısına '00000001' değerini yüklüyoruz.
         MOVWF PORTB       ; W içindeki değeri PortB'ye gönderiyoruz.
         CALL  BEKLE       ; BEKLE alt programını çağırıyoruz.
         MOVLW b'00000010' ; Aynı şekilde, sıra ile diğer değerleri gönderip
bekletiyoruz.
         MOVWF PORTB
         CALL  BEKLE
         MOVLW b'00000100'
         MOVWF PORTB
         CALL  BEKLE
         MOVLW b'00001000'
         MOVWF PORTB
         CALL  BEKLE
         MOVLW b'00000000'
         MOVWF PORTB
         CALL  BEKLE
         GOTO  BASLA
```

```
BEKLE    MOVLW 0xFF          ; W saklayıcısına FF değerini yüklüyoruz.
(255='11111111'=0xFF)
         MOVWF 0x0C          ; W saklayıcısındaki değeri 0x0C adresli bellek gözüne
yazıyoruz.
DONGU1   MOVLW 0xFF          ; Yine W saklayıcısına FF değerini yüklüyoruz.
         MOVWF 0x0D          ; Bu değeri bu sefer 0x0D bellek gözüne yazıyoruz.
DONGU2
         DECFSZ  0x0D,1      ; 0x0D içerisindeki veriyi 1 azalt. Eğer sonuç 0 ise
bir alt satırı geç.
         GOTO  DONGU2        ; DONGU2 satırına git.
         DECFSZ  0x0C,1      ; 0x0C içerisindeki veriyi 1 azalt. Eğer sonuç 0 ise
bir alt satırı geç.
         GOTO  DONGU1        ; DONGU1 satırına git.
         RETURN              ; BEKLE fonksiyonunun çağırıldığı satıra geri
dönüyoruz.
         END                 ; Program sonu
```

Dikkat ederseniz, bazı **MOVLW** komutları içerisinde gönderdiğimiz verileri **b'00000001'** şeklinde gönderdik. Bunun yerine **0x01** de yazabilirdik. Kısacası assembly'e verileri binary (ikilik) sistemde gönderebilirsiniz. ledleri sıra ile yakacağımız için, lojik "1" değerlerini daha iyi izleyebilmek açısından, bu şekilde kullandım. Şimdi, bu bölümde ilk kez kullandığımız komut satırlarını daha detaylı inceleyelim.

CALL BEKLE: CALL komutu alt program çağırmaya yarar. Çağırılan alt program, etiket adı ile çağırılır. Alt program çalışmaya başladıktan sonra, **RETURN** komutu ile program çağırılan satırın bir alt satırına geri döner.

BEKLE Alt Programı: İlk olarak **W** saklayıcısına 0xFF değeri yükleniyor. Sonra bu değer, 0x0C bellek gözüne yazılıyor. Aynı şekilde 0x0D bellek gözüne de aynı değer yazılıyor. Daha sonra **DECFSZ** komutu ile 0x0D adresindeki veri, bir (1) azaltılıyor. **DECFSZ** komutu, belirtilen adresin içeriği SIFIR olana kadar, bir alt satıra geçer. Sonuç SIFIR'a ulaştığında ise bir altındaki satırı atlar. Bir alt satırdaki **GOTO** komutu sebebiyle, 0x0D'nin içeriği SIFIR olana kadar döngü devam edecektir. Daha sonra 0x0C için aynı işlemi yapıyoruz. Ardından 0x0C bir azaltılıyor ve sonuç SIFIR değil ise tekrar **DONGU1**'e geri gidiyor. Dolayısı ile, her azalan 0x0C verisi için, 0x0D her seferinde FF (255) değerinden SIFIR'a gidiyor. Yani **DONGU2** içerisinden **DONGU1** çağırılıyor. Burada dikkat ederseniz, iç içe iki döngü kullandık.

Kullandığımız 0x0C ve 0x0D bellek gözlerini rasgele seçtik. Dikkat edilmesi gereken nokta, kullanmak istediğimiz bellek gözüne ait bank'a geçmemiz gerektiğidir. Bizim kullandığımız bellek gözleri, Şekil 19.3'te görüldüğü gibi Bank0 üzerindedir. Biz de programın en başında, en son Bank0'a geçmiştik. Zaten Bank0'da olduğumuz için bank değiştirme işlemi yapmamıza gerek kalmadı.

PIC'e programı yükledikten sonra, test devresini çalıştırdığınızda, ledler sıra ile yanıp sönecektir. Yanıp sönme hızını, 0x0D ve 0x0C adreslerine yüklediğimiz, 0xFF değerini küçülterek, döngünün daha çabuk bitmesini ve böylece gecikme süresinin azalmasını sağlayabilirsiniz.

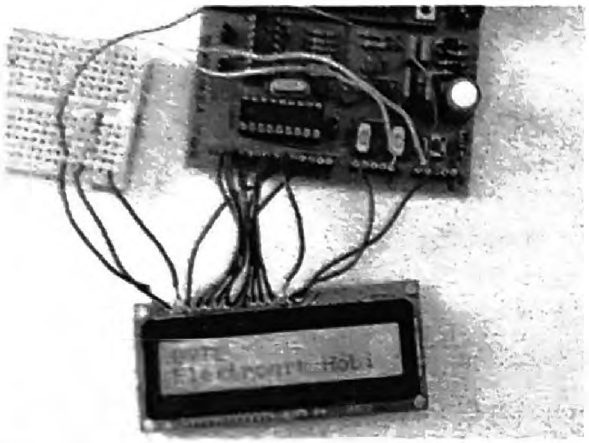
YİRMİBİRİNCİ BÖLÜM

21

LCD PANEL KONTROLÜ

LCD Paneller

Farkına varmasak bile LCD paneller günlük hayatımızın bir parçası gibidir. En basit olarak bazı cep telefonlarında, hesap makinelerinde ve birçok elektronik eşyada bu teknoloji kullanılmaktadır. LCD paneller hafif ve az yer kaplarlar. Yaptığınız devrelere küçük bir LCD panel bağlayarak devreden, kullanıcıya göndermek istediğiniz mesajları bu ekrana yazdırabilirsiniz.



ŞEKİL 21.1: PIC deneme kartı ile LCD panelin bağlantısı.

Standart LCD paneller, +5 Volt ile çalışır. Bu voltajı PIC deneme kartımızın üzerinden alabiliyoruz. Benim kullandığım LCD panelin modeli (2 X 16) SAMSUNG-1602D idi. Kitaplarda sık rastlanan HITACHI HD44780 model LCD panel de işinizi görebilir. Genel olarak 2X16'lık ve 14 konnektörü bulunan LCD panellerin bacak bağlantıları aynı sayılır. Önemli olan, LCD panelin hangi bacaklarının hangi işlev için kullanıldığını bilmektir. Standart bir LCD panelin üzerinde 14 tane konnektör bulunur. Bu bacakların görevleri aşağıdaki gibidir.

TABLO 21.1: LCD Pinlerinin İşlevleri

LCD Pinleri	Pin İşlevi
1	Toprak (Ground)
2	VCC +5 Volt
3	VO (Kontrast)
4	RS (Register Select)
5	R/W (Okuma ya da Yazma Modu)
6	E (Enable)
7-14	D0-D7 (DATA girişleri)
15*	LCD Panel ışığı (+5 Volt)
16*	Toprak (Ground)

* 15 ve 16 numaralı pinler bazı LCD panellerde bulunmayabilir. Eğer LCD panel ışıklı ise iç ampulü bu bacakları kullanarak yakmak mümkündür.

RS (register select): RS girişi "0" (düşük) olduğu zaman LCD panelin komut saklayıcısı, yüksek olduğu zaman ise veri saklayıcısı seçilmiş olur. Biz LCD panelin RS girişini PIC'in RA0 pinine bağlıyoruz.

R/W (read/write): Bu giriş düşük, yani "0" olduğunda, LCD panel yazma modundadır. Karakterleri yazdırırken, yazma modunu kullanacağız. Hatırlarsanız paralel porttan programlarken bu girişi doğrudan topraklamıştık. Bu durumda LCD hep yazma modunda kalmıştı. Dolayısı ile LCD'ye yazma işleminin bitip bitmediğine bakmadan, **FOR-NEXT**'ler ile gecikme döngüleri yazmıştık. Fakat PIC içerisinde bu kadar bekletme sağlamak kolay olmadığından, yazma işleminin bitip bitmediğini belirten meşgul bayrağını (busy flag) okuyacağız. Bunun için, bu girişe "1" uygulamamız gerekecek. Program içerisindeki **LCDBUSY** fonksiyonunda bu durumu daha iyi anlayacaksınız.

E (enable): PORTB üzerine 8 bitlik veriyi yerleştirdikten sonra LCD panele, panelin E ucu ile buradaki veriyi okumasını sağlayabiliriz. Bunun için E ucuna bağlı olan RA2 pinini "1" (yüksek) yapmalıyız. Daha sonra biraz beklemeli ve bir sonraki byte için tekrar "0" (düşük) durumuna getirmeliyiz.

VO (kontrast): 3 numaralı VO kontrast girişini doğrudan 2K5'lik (2.5K) bir direnç üzerinden toprağa gönderiyoruz. Buradaki direncin değerini arttırarak kontrastı düşürebilir, azaltarak ise kontrastı yükseltebilirsiniz. Eğer bu kontrast ile sürekli oynamak istiyorsanız buraya 2K5'lik bir ayarlı direnç bağlayabilirsiniz.

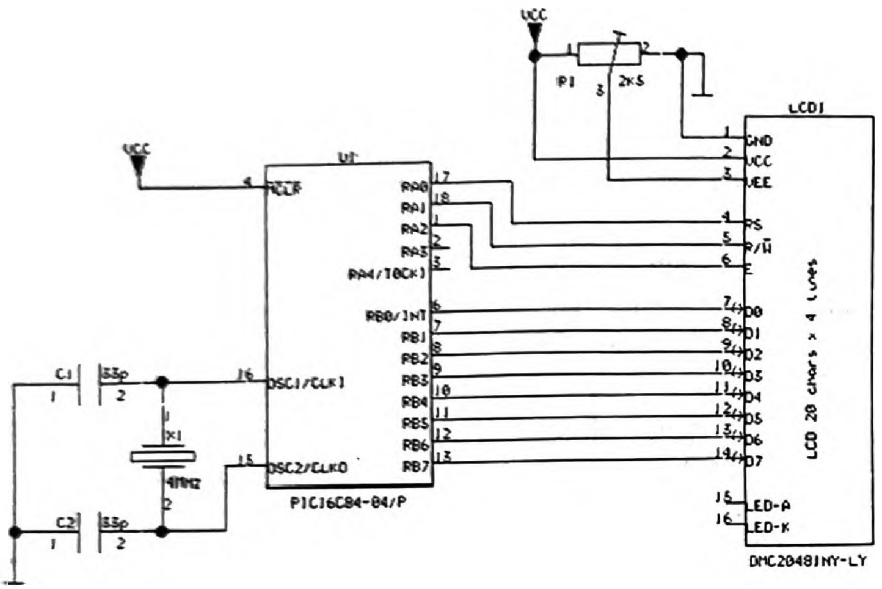
DATA: 7-14 numaralı pinleri doğrudan PORTB'nin bacaklarına bağlıyoruz. Bu sayede istediğimiz komut ya da veriyi LCD panele göndermemiz mümkün olacaktır. Aşağıda Tablo 21.2'de standart LCD panelin komut saklayıcısının fonksiyonlarını yazıyorum. Yukarıda anlattığım şekilde RS girişi ile veri saklayıcısı ya da komut saklayıcısını seçmemiz mümkün olacaktır. Aşağıdaki komutları kullanabilmek için önce komut saklayıcısını seçmeliyiz.

TABLO 21.2: Standart LCD Panel Komut Saklayıcısı Fonksiyonları

İşlem	D7	D6	D5	D4	D3	D2	D1	D0	Açıklama
Ekranı Sil	0	0	0	0	0	0	0	1	Ekranı siler
Ekran seçenekleri	0	0	0	0	1	P	K	Y	(P) Panel AÇ (1) / KAPA (0) (K) Kürsör AÇ (1) / KAPA (0) (Y) Yanıp Sönme AÇ (1) / KAPA (0)
Fonksiyon ayarları	0	0	1	D	N	0	X	X	(D) Veri girişi 8-bit (1) / 4-bit (0) (N) Çift satır (1) / Tek satır (0)
CGRAM Adres ayarı (ilk satırın adresleri)	0	1	C5	C4	C3	C2	C1	C0	(C5-C0) CGRAM'e gidecek veriler.
DDRAM Adres ayarı (İkinci satırın adresleri)	1	D6	D5	D4	D3	D2	D1	D0	(D6-D0) DDRAM'e gidecek veriler.
Busy Flag / Adres sayaç okuma	BF	AC 6	AC 5	AC 4	AC 3	AC 2	AC 1	AC 0	Meşgul bayrağı (BF) RW=1, RS="0" iken D7 pininden okunur. D6-D0'dan address counter verisi okunur.

TABLO 21.2: Standart LCD Panel Komut Saklayıcısı Fonksiyonları

İşlem	D7	D6	D5	D4	D3	D2	D1	D0	Açıklama
Cursor Shift	0	0	0	1	SC	RL	X	X	Shift (1) / Cursor (0) Right(1) / Left(0)
Entry Mode Set	0	0	0	0	0	1	ID	S	Increment (1), Decrement(0) Shift (1) on (0) off.



ŞEKİL 21.2: PIC ile LCD panel kontrol devresi.

PIC ile LCD Panel Kontrol Uygulaması

Bu uygulamayı gerçekleştirmeden önce, daha önceki bölümlerde bulunan bütün led uygulamalarımızı öğrenmiş ve komutların kullanımını anlamış olmanız gerekmektedir. Çünkü anlatımlarda, komutların kullanımını bildiğiniz varsayılmaktadır.

İlk olarak değişkenleri tanımlıyoruz. Dikkat ederseniz, bu uygulamamızda ilk kez `include` komutunu kullandık. Bu komut ile `p16f84.inc` dosyasının içeriğini programımıza eklemiş olduk. Bu dosya içinde PIC içerisindeki tüm saklayıcıların adresleri birer değişkene atanmaktadır. Bu dosya `.asm` dosyanızla aynı klasörde olmalı. MPASM programı ile bu include dosyaları bilgisayarınıza yükleniyor.

Hatırlatmak istediğim bir nokta daha var. Programda kullandığımız alt fonksiyonlara veriler `W` saklayıcısı ile gönderilmektedir. Dikkat ederseniz bir alt fonksiyon çağırmadan önce, eğer bir parametre göndermek gerekiyorsa, `W` saklayıcısına mutlaka bir değer atıyoruz. Zaten kodu inceledikçe fark edeceksiniz.

```
                LIST P=16F84        ;PIC modelini tanıtıyoruz
                include <p16f84.inc>    ;16F84 için INC dosyasını ekliyoruz.

LCD_DATA        EQU   PORTB
```

```
LCD_DATA_TRIS  EQU   TRISB
LCD_CTRL EQU   PORTA
LCD_LINE0      EQU   0x000
LCD_LINE1      EQU   0x014
LCD_LINE3      EQU   0x054

; PORTA bitleri
LCD_E          EQU   2     ; LCD Enable kontrol hattı
LCD_RW         EQU   1     ; LCD Okuma/Yazma kontrol hattı
LCD_RS         EQU   0     ; LCD Saklayıcı seçim kontrol hattı

LCD_TEMP       EQU   0x020 ; LCD alt programlarında kullanıyoruz
TABLE_INDEX    EQU   0x021 ; Tablo satırlarını tutmak için
COUNT          EQU   0x022 ; Sayaç için
DELAY          EQU   0x023 ; DELAYxxx gecikme fonksiyonu için
X_DELAY        EQU   0x024 ; X_DELAYxxx gecikme fonksiyonu için
```

Program bu noktada başlıyor. **W** saklayıcısına 0x0F8 değeri yükleniyor; bu da ikilik sistemde '00011111' değerine karşılık geliyor. Bu durumda "0"a denk gelen uçlar çıkış, "1"e denk gelen uçlar ise giriş oluyor. Aynı şekilde daha önceki uygulamalarımızda olduğu gibi PORTB'yi de çıkışa ayarlıyoruz. Farklı olarak PORTB'deki pull-up'ları kaldırmak için kullanılan değişkenler (**OPTION_REG**, **NOTRBPU**) **.inc** dosyası ile gelmektedir. LCD paneli açmak için **LCDINIT** fonksiyonunu buradan çağırıyoruz.

```
START
CLRF   STATUS              ; Bank 0 seç
CLRF   PORTA               ; Tüm portların çıkışları düşük yap
CLRF   PORTB
BSF    STATUS, RP0         ; Bank 1 seç
MOVLW  0x0F8               ; RA2-0 çıkış, RA4-3 giriş
MOVWF  TRISA
MOVLW  0x000               ; RB7-0 çıkış
MOVWF  TRISB
BSF    OPTION_REG, NOT_RBPU          ; PORTB pull-upları temizle
BCF    STATUS, RP0         ; Bank 0 seç
CALL   LCDINIT             ; LCD panel aç
```

LCD panel açıldıktan sonra, **W** saklayıcısına 1'nci satırın adresini yüklüyoruz ve **LCDSDDA** fonksiyonunu çağırıyoruz. Bu fonksiyon imleci **W** içindeki satırın sol başına alıyor. Yani 1. satırın başına geliyoruz. Daha sonra **TABLE_MSG** fonksiyonunu çağırıyoruz. Bu fonksiyon "BYTE" kelimesi içindeki karakterleri sıra ile 1'nci satıra yazıyor. Aynı şekilde **W** içerisine 2'nci satırın adresini yükleyip ikinci satıra da "Elektronik Hobi" yazdırıyoruz. Elbette esas olan, bu fonksiyonun içinde neler olduğu. Bunu da aşağıda inceleyeceğiz.

```
MOVLW LCD_LINE0   ; 1. Satırın adresini W ye yükle
CALL  LCDSDDA     ; Kürsörü W deki satıra ve sol başa al
CALL  TABLE_MSG   ; Mesajı göster (BYTE)
MOVLW LCD_LINE1   ; 2. Satırın adresini W ye yükle
CALL  LCDSDDA     ; Kürsörü W deki satıra ve sol başa al
CALL  TABLE_MSG2  ; Mesajı göster (Elektronik Hobi)
```

Program burada sona eriyor. Daha doğrusu kısır döngüye giriyor ve bu döngüden hiç çıkmıyor.

```
LOOP
      GOTO  LOOP      ; Kısır döngüye giriliyor
```

Sıra geldi programın en can alıcı fonksiyonlarından bir tanesine. İlk olarak bir indis değişkenimiz (**TABLE_INDEX**) oluyor. Buna ilk başta "0" yüklüyoruz ve **W** saklayıcısına atıyoruz. Daha sonra **MSG1** fonksiyonu çağırılıyor. **MSG1** fonksiyonunda program tablodaki **W** saklayıcısı içindeki değişkene yöneliyor. Bunu **MSG1** fonksiyonunu açıklarken daha net anlayacaksınız. Daha sonra mesajın sonu geldiğinde **MSG1** fonksiyonu "0" döndürüyor. **ANDLW 0xOFF** komutu ile bu içerik **AND**'leniyor. Burada sık kullanılan bir sınama yöntemi öğreniyoruz. Eğer **ANDLW** işlemi sonucunda "0" değeri elde edilirse, **STATUS** yazmacının ZERO (Z) biti 1 olur. Bu durumda eğer **MSG1** fonksiyonundan "0" gelirse ZERO biti "1" olacaktır. Dolayısı ile **BTFSC** komutu bir alt satırdaki komutu işleyecektir. Bu komuttan sonra fonksiyon çağırıldığı yere geri dönüyor. Eğer mesajın sonu gelmemiş ise **W** saklayıcısındaki karakter **LCDPUTCHAR** fonksiyonu ile yazdırılıyor. Daha sonra **TABLE_INDEX** değişkeni **W** içine atılıyor ve **ADDLW** komutu ile bir arttırılıyor. Ardından **MSG1** fonksiyonuna tekrar gidiliyor. W'nun içeriği bir arttığı için bir sonraki karakter çekiliyor. Böylece **MSG1** fonksiyonu "0" döndürene kadar bu işlem sürüyor. **TABLE_MSG2** fonksiyonu da aynı şekilde çalışıyor.

```
TABLE_MSG
            MOVLW 0           ; Mesaj tablosu indexi için W=0 yapılıyor
DISP_MSG
            MOVWF TABLE_INDEX ; TABLE_INDEX değişkenine W değeri atanıyor
            CALL  MSG1
            ANDLW 0x0FF               ; Mesajın sonu geldimi?
            BTFSC STATUS, Z           ; (geldi ise 0 döndür (Z=zero)
            GOTO  TABLE_MSG_END
            CALL  LCDPUTCHAR          ; Karekteri göster
            MOVF  TABLE_INDEX, W      ; Tablodan bir sonraki karekteri al
            ADDLW 1
            GOTO  DISP_MSG
TABLE_MSG_END  RETURN

TABLE_MSG2
            MOVLW 0                   ; Mesaj tablosu indexi için W=0 yapılıyor
DISP_MSG2
            MOVWF TABLE_INDEX         ; TABLE_INDEX değişkenine W değeri atanıyor
            CALL  MSG2
            ANDLW 0x0FF               ; Mesajın sonu geldimi?
            BTFSC STATUS, Z           ; (geldi ise 0 döndür (Z=zero)
            GOTO  TABLE_MSG2_END
            CALL  LCDPUTCHAR          ; Karekteri göster
            MOVF  TABLE_INDEX, W      ; Tablodan bir sonraki karekteri al
            ADDLW 1
            GOTO  DISP_MSG2
TABLE_MSG2_END RETURN
```

LCDINIT fonksiyonunu, LCD paneli açmak için kullanıyoruz. Bu fonksiyon ile Tablo 21.2'den, istediğimiz özellikleri ayarlıyoruz. 0x038 değeri, 56 (00111000) değerine karşılık geliyor. Bu durumda LCD paneli 8 bitlik, çift satır veri girişi için ayarlamış oluyoruz. Aynı şekilde, 0x000 verisini göndererek , önce panel imleci ve yanıp sönmeyi (blink) kapatıyoruz. Daha sonra, sadece paneli açıyoruz. İmleç ise kapalı kalıyor. Bu ayarlar biraz da size kalmış; tüm komutlar Tablo 21.2'de gözüküyor. Daha sonra aynı şekilde 0x002 verisi ile otomatik artışı açıyoruz. Bu artış, bize her karakter basıldıktan sonra bir sonraki karakter boşluğunun adresine geçmemizi sağlar. Böylece her seferine karakter gönderirken adres belirtmemize gerek kalmaz. Hepsi yan yana dizilir. Çağırılan diğer fonksiyonların açıklamalarını da, aşağıda inceleyeceğiz. Temel olarak yapılan işlem LCD paneli yazı yazdırmaya hazır bir hale getirmek.

LCDINIT Komutu

```
LCDINIT
                        ; Meşgul bayrağı henüz geçerli değil
        CLRF  LCD_CTRL  ; PORTA çıkışlarının tümü düşük.
                        ; power-up gecikmesi
        MOVLW 0x01E
        CALL  X_DELAY500 ; 30 * 0.5mS = 15mS
                        ; Meşgul bayrağı artık geçerli
        MOVLW 0x038     ; 8-bit-arayüz, 2-satır
        CALL  LCDPUTCMD
        MOVLW 0x000     ; disp. kapat, kürs. kapat, blink kapat
        CALL  LCDDMODE
        CALL  LCDCLEAR
        MOVLW 0x004     ; disp.aç, kürs.kapat
        CALL  LCDDMODE
        MOVLW 0x002     ; otomatik artış (shift-cursor)
        CALL  LCDEMODE
        RETURN
```

LCDBUSY fonksiyonu, bize meşgul bayrağının pasif olduğunu bildirmeye yarıyor. Bu komut çağırıldığında meşgul bayrağı, LCD'nin sürdürdüğü işlem sona erene kadar sürekli kontrol ediliyor. Sonunda bayrak pasif olunca döngüden çıkılıyor. Önce PORTA'nın 0'ncı biti "0" yapılıyor. İlk bakışta hiç de öyle gözükmüyor olabilir, fakat dikkat ederseniz, **LCD_CTRL** değişkenini en başta PORTA'ya eşitlemiştik. **LCD_RS** değeri de "0"a eşitti. Bu şekilde LCD'nin RS çıkışını "0" yaparak komut saklayıcısı seçiliyor. Sonra RW, "1" yapılıyor ve E çıkışı önce düşük yapılıp, DATA pinlerindeki veri **W**'ye atılıyor. Ardından E tekrar yüksek yapılarak meşgul bayrağı okunmuş oluyor. Aynı yukarıda kullandığımız **ANDLW** komutu ile 0x080 yani "10000000" değeri ile, (Tablo 21.2'den neden bu değeri kullandığımıza bakın) okuduğumuz değeri **AND**'liyoruz. Bu işlem sonucu eğer 0 elde edilirse STATUS'ün ZERO biti "1" oluyor. Hemen ardından bu durumu **BTFSS** komutu ile test ediyoruz. Eğer sonuç "1" ise alt satırı atlıyoruz. Aksi taktirde meşgul bayrağı pasif olana kadar döngü her seferinde yeniden başlıyor. Döngü bitince PORTB tekrar çıkışa ayarlanıyor ve fonksiyon çağırıldığı yere geri dönüyor.

LCDBUSY Komutu

```
LCDBUSY
        BSF   STATUS,RP0          ; Bank 1 seç
        MOVLW 0x0FF               ; PORTB giriş için ayarla
        MOVWF LCD_DATA_TRIS
        BCF   STATUS, RP0         ; Bank 0 seç
        BCF   LCD_CTRL, LCD_RS    ; Komut saklıyıcısı seç
        BSF   LCD_CTRL, LCD_RW    ; Meşgul bayrağını okumaya ayarla
        BSF   LCD_CTRL, LCD_E     ; LCD E hattı yüksek yap
        MOVF  LCD_DATA, W         ; Meşgul bayrağını oku
        BCF   LCD_CTRL, LCD_E     ; LCD E hattı düşük
        ANDLW 0x80                ; Meşgul bayrağını kontrol et, Yüksek = Meşgul
        BTFSS STATUS, Z
        GOTO  LCDBUSY
LCDNOTBUSY  BCF   LCD_CTRL, LCD_RW
        BSF   STATUS, RP0         ; Bank 1 seç
        MOVLW 0x000
        MOVWF LCD_DATA_TRIS       ; PORTB çıkış
        BCF   STATUS, RP0         ; Bank 0 seç
```

LCDCLEAR komutu paneli temizler ve imleci sol başa (sol üst köşe) alır. 0x001 (00000001) değerinin ekran silme komutuna karşılık geldiğini Tablo 21.2'de görebilirsiniz.

LCDCLEAR Komutu

```
LCDCLEAR
        MOVLW 0x001
        CALL  LCDPUTCMD
        RETURN
```

LCDEMODE komutu, Tablo 21.2'deki Entry Mode Set satırı ile ilgili bir durum. Bu komutu **INIT** fonksiyonu sırasında kullanıyoruz. Bu fonksiyona geldiğimizde **W** içindeki değer 0x02 yani 00000010, bu değeri **ANDLW 0x003** (00000011) komutu ile **AND**'liyoruz, dolayısı ile **W**'nun değeri 00000010 oluyor. Bunu da 0x004 (00000100) ile **OR**'luyoruz. Sonuçta 00000110 değeri elde ediliyor. Bu da Tablo 21.2'de Entry Mode Set komutuna karşılık geliyor. Sağdan 2'nci bit otomatik olarak bir sonraki sütunun adresine geçmeyi sağlıyor. En sağdaki 1'nci bit yani "0" değeri ise display shift'i kapatıyor.

LCDEMODE Komutu

```
LCDEMODE
        ANDLW 0x003
        IORLW 0x004
        CALL  LCDPUTCMD
        RETURN
```

LCDDMODE komutu yine **INIT** (açma) rutini içerisinden çağırılıyor. **AND** ve **OR** işlemleri sonucunda 00001100 değeri elde ediliyor. Tablo 21.2'deki Ekran seçenekleri satırından anlaşıldığı gibi bu komut, paneli açıyor, kürsörü ve yanıp sönmeyi kapatıyor.

LCDDMODE Komutu

```
LCDDMODE
        ANDLW 0x007
        IORLW 0x008
        CALL  LCDPUTCMD
        RETURN
```

LCDSDDA fonksiyonunu imlecin yerini ayarlamak için kullanıyoruz. İmleci birinci satırın başına almak için **CGRAM** komutu gönderilir. Tablo 21.2'den de anlaşılacağı gibi **CGRAM** (1'nci satır) için "01000000" yani 64 verisini göndermeliyiz. Her satırın birinci karakterinin adresi 64'tür. Yani birinci satırın birinci sütunu için 64+64=128 verisini komut saklayıcısına göndermeliyiz. İkinci karakter için ise 64+65=129 olmalı. Bu şekilde gidiyor. Fakat her seferinde bu adresleri bir bir girmenize gerek kalmayacaktır. Çünkü siz LCD panele karakter gönderdikçe LCD bu adresi otomatik olarak artıracak ve gönderdiğiniz karakterleri sıra ile yazdıracak. Biz bu komutu sadece hangi satıra yazmak istediğimizi belirtmek için kullanacağız. İkinci satırın ilk karakterine gitmek için **DDRAM** fonksiyonunu "10000000" ile çağıracağız. Bu satırda da ilk karakterin adresi 64 olduğundan 128+64=192 verisini komut saklayıcısına göndermemiz ikinci satırın başına gitmemiz için yeterli olacaktır. Burada yapılan **OR** işlemi sonucunda, **W** içerisinde satırın adresi bulunduğundan bu adrese 64 eklemiş oluyoruz. Buda o satırın ilk karakterine denk gelecektir.

LCDSDDA Komutu

```
LCDSDDA
        IORLW 0x080
        CALL  LCDPUTCMD
        RETURN
```

LCDPUTCHAR komutu **W** içerisindeki karakteri LCD'ye yazdırmaya yarıyor. İlk başta LCD hazır (meşgul bayrağı pasif) olana kadar bekliyoruz. Önce RW düşük yapılıyor. Yani LCD'yi yazma için ayarlıyoruz. Sonra veri saklayıcısı seçiliyor. Sonra E hattı yüksek yapılıyor, veri gönderiliyor ve E hattı tekrar düşük yapılıyor. Bu işlemler LCD'ye karakter göndermek için yapılan standart işlemlerdir. Paralel port ile geliştirdiğimiz uygulamada da aynısını yapıyorduk.

LCDPUTCHAR komutu

```
LCDPUTCHAR
        MOVWF LCD_TEMP          ; W deki karekteri geçici değişkene at
        CALL  LCDBUSY           ; LCD hazır olana kadar bekle
        BCF   LCD_CTRL, LCD_RW  ; LCD yazma moduna al
        BSF   LCD_CTRL, LCD_RS  ; Veri saklayıcısı seç
        BSF   LCD_CTRL, LCD_E   ; LCD E-hattı Yüksek
        MOVF  LCD_TEMP, W       ; Geçici değişkeni tekrar W ye aktar
```

```
        MOVWF LCD_DATA          ; LCD ye veriyi gönder
        BCF   LCD_CTRL, LCD_E   ; LCD E-hattı düşük
        RETURN
```

LCDPUTCMD komutu aynı **LCDPUTCHAR** gibi çalışıyor. Tek farkı, veri saklayıcısı yerine komut saklayıcısı seçilmiş oluyor. Böylece **W** içerisindeki komut, LCD'nin komut saklayıcısına gönderilmiş oluyor.

LCDPUTCMD komutu

```
LCDPUTCMD
        MOVWF LCD_TEMP          ; W deki komutu geçici değişkene at
        CALL  LCDBUSY           ; LCD hazır olana kadar bekle
        BCF   LCD_CTRL, LCD_RW  ; LCD okuma moduna al
        BCF   LCD_CTRL, LCD_RS  ; Komut saklayıcısı seç
        BSF   LCD_CTRL, LCD_E   ; LCD E-hattı yüksek
        MOVF  LCD_TEMP, W       ; Geçici değişkeni tekrar W ye yükle
        MOVWF LCD_DATA          ; LCD ye komutu gönder
        BCF   LCD_CTRL, LCD_E   ; LCD E-hattı düşük
        RETURN
```

Gecikme alt programları, ana programa bekleme sağlamak amacıyla kullanıyor. Daha önceki bölümde gecikme alt programlarını incelediğimiz için, burada tekrar bahsetmiyoruz. Sonuç olarak fonksiyonlar gecikme sağlamak için kullanılıyor.

```
DELAY500          MOVLW D'165'
                  MOVWF DELAY
DELAY500_LOOP     DECFSZ   DELAY, F
                  GOTO  DELAY500_LOOP
DELAY500_END      RETURN
X_DELAY500        MOVWF X_DELAY
X_DELAY500_LOOP   CALL  DELAY500
                  DECFSZ   X_DELAY, F
                  GOTO  X_DELAY500_LOOP
X_DELAY500_END    RETURN
```

Aşağıda sık kullanılan bir tablo okutma yöntemini görüyorsunuz. Burada **PCL**'nin içeriği kaç ise o **RETLW** komutuna sapar ve geri döner. Sonuçta **W** saklayıcısının içeriği, sapılan karakter olur. **PCL** adres yazmacı ise **INCLUDE** ile dahil edilen dosya içerisinde tanımlanmıştır. **return**, **retfie** ve **retlw** komutları yığının en tepesindeki elemanının içeriğini **PCL**'ye aktarır. Sonuçta kullanılan yöntem, **PCL** değeri kaç ise o karakteri **W**'ya yükleyip geri dönmektir. Geri dönüldüğü zaman bu içeriğin "0" olup olmadığı kontrol ediliyor ve sayaç 1 artırılarak, tekrar **MSG1** fonksiyonuna geliniyor. Eğer içeriği sıfır olan **retlw** satırına gelinirse bir daha buraya gelinmiyor.

```
MSG1
        addwf PCL ,F      ;W de belirtilen karektere atla
        retlw 'B'
        retlw 'Y'
        retlw 'T'
```

```
        retlw 'E'
MSG1_END
        retlw 0
MSG2
        addwf PCL ,F        ;W de belirtilen karektere atla
        retlw 'E'
        retlw 'l'
        retlw 'e'
        retlw 'k'
        retlw 't'
        retlw 'r'
        retlw 'o'
        retlw 'n'
        retlw 'i'
        retlw 'k'
        retlw ' '
        retlw 'H'
        retlw 'o'
        retlw 'b'
        retlw 'i'
MSG2_END
        retlw 0
END                         ; Program Sonu
```

YİRMİİKİNCİ BÖLÜM

22

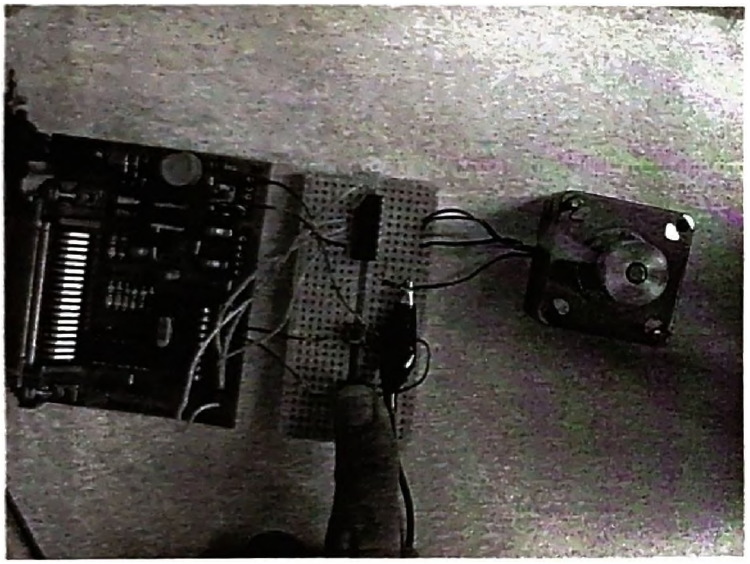
PIC İLE ADIM MOTOR KONTROLÜ

Adım Motorlar

Artık daha eğlenceli PIC uygulamalarına yavaş yavaş geçiyoruz. Bu bölümde adım motor kontrol etmeyi öğreneceğiz. Hatırlarsanız Kısım 1'de adım motor kontrolünü bilgisayarın paralel portu ile gerçekleştirmiştik. Uygulama yine benzer olacak. İki adet buton ile motoru sağa ya da sola çevirebileceğiz. Adım motorun nasıl çalıştığını Kısım 1'den kısa bir alıntı yaparak yeniden gözden geçirmek istiyorum.

Adım motorların, endüstriyel ve elektronik uygulamalarda kullanımı oldukça fazladır. En basitinden, bilgisayarımızdaki floppy disket sürücüsünde ve hard disklerde bu teknolojiye başvurulmuştur. Adım motorlar, girişlerine uygulanan lojik sinyalleri, dönme hareketine çevirirler. İstediğiniz yönde ve derecede döndürebileceğiniz adım motorlar, hassas hareketleri sayesinde, birçok cihazda konum kontrolü amacıyla kullanılmaktadır.

Adım motorun kablolarından bir veya iki tanesi ortaktır (vMotor). Yaptığımız işlem basit olarak bu ortak kabloya sürekli +12 Volt göndermek ve diğer uçları ise belli bir sırada toprağa göndererek bir adım hareketi elde etmek. Board üzerinde, deneme kartı ile kurulmuş devreyi, Şekil 22.1'de görebilirsiniz.



ŞEKİL 22.1: Adım motor devresinin board üzerinde kurulmuş hali.

Adım motor sürücüsü olarak ULN2003 entegresini kullanıyoruz. Sürücü devresi olarak kullanılan ULN2003, içerisinde 7 adet NPN transistör ve dahili diyot barındırmaktadır. Haliyle bizi transistör bacaklarıyla uğraşmaktan kurtarmaktadır. Kullanımı ise oldukça kolaydır. Devre şemasından da anlaşılabileceği gibi, 9 numaralı bacağına +12 Volt ve 8 numaralı bacağına da Toprak (ground) uyguluyoruz. Daha sonra 3 ve 6 numaralı bacaklara da PIC'in PORTB çıkışından gelen lojik +5 Voltluk sinyalleri uygulayacağız. Bu sayede, örneğin 3 numaralı bacağa +5 Volt (lojik voltaj) uyguladığımızda 14 numaralı bacak toprak olacaktır. Aynı şekilde sırayla 4 için 13, 5 için 12, 6 için ise 11 numaralı bacaklar toprak olacaktır.

Her şeyden önce bir adım motora ihtiyacımız var. İşimize en çok yarayacak olan adım motorunu eski 5¼ disket sürücülerinden kolayca sökebilirsiniz. Tabii bundan önce

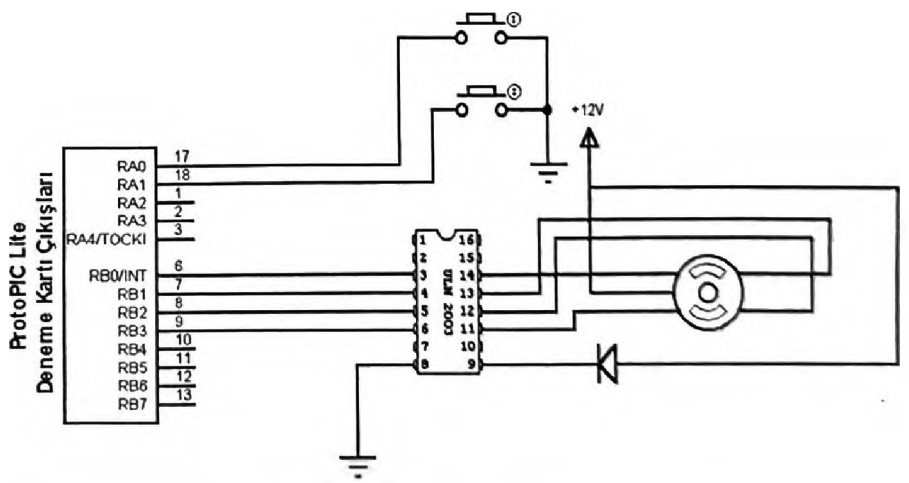
parçalayabilecek bir sürücü bulabilmeniz gerekli. Eğer bulamıyorsanız, adım motor için sanırım biraz elektronikçi dolaşmanız gerekecektir. Bulacağınız adım motoru 4, 5 ya da 6 kablolu olabilir. Ama benim en sık rastladığım 5 kablolu olanı olduğundan, uygulamamızda 5 kablolu adım motoru kullanacağız. Çalışma prensipleri aynı olduğundan kablo sayısının değişmesi yazacağımız programın mantığının değişmesi anlamına gelmeyecektir.

5 kablolu adım motorunun kablolarından bir tanesi vMotor dediğimiz ortak kablodur. Önemli olan bu kablonun hangisi olduğunu bulmaktır. Bunun için avometrenizi OHM ölçere getirin ve kabloların uçlarını ikişer ikişer ölçün. Tüm uçlar ile arasındaki direnç aynı olan kablo ortak vMotor kablosudur. Biraz deneyerek bulabileceğinizden eminim. Diğer dört kablo ise motor kömürlerine (coil) gitmektedir. Bu dört kablonun da bir sırası vardır. Bu sırayı da deneme yanılma yöntemiyle bulmak mümkün olacaktır. Eğer bu kabloları yanlış sırada bağlarsanız, motor dönmek yerine sadece titreme yapacaktır. Yukarıda da bahsettiğim gibi, motora adım attırmak için yapmamız gereken, vMotor kablosuna +12 Volt verirken, diğer kömürlere bağlı kablolara belli bir sıra ile toprak göndermektir.

Yukarıda bahsettiğimiz bu dört kabloya toprak sinyalini göndermek için entegrenin 3, 4, 5 ve 6'ncı bacaklarına sıra ile lojik '1' göndermemiz gerekiyor. Bunun için PORTB çıkışına sıra ile '1' değeri göndermeliyiz. PIC üzerinde bunu gerçekleştirmek için sola ve sağa öteleme yöntemlerini kullanacağız.

PIC ile Adım Motor Kontrol Uygulaması

Bu uygulamayı gerçekleştirmeden önce, PIC ile ilgili önceki bütün LED uygulamalarımızı öğrenmiş ve komutların kullanımını anlamış olmanız gerekmektedir. Çünkü anlatımlarda, komutların kullanımını bildiğiniz varsayılmaktadır. Kullandığımız PIC modeli ise, her zamanki gibi, 16F84. Şekil 22.2'deki devre şemasındaki çıkışlar bu kartın üzerinden alınan PIC çıkışlarıdır. Ayrıca devrede kullandığımız diyotun değerinin herhangi bir önemi yoktur. Motorda adım attıktan sonra oluşacak ters EMK'yı engellemek için kullanıyoruz. Örneğin 1N4001 diyot kullanabilirsiniz.



ŞEKİL 22.2: PIC ile adım motor kontrol devresi.

İlk olarak PIC modelimizi ve değişkenlerimizi tanımlıyoruz. Fakat bu sefer farklı bir yöntem göstermek istiyorum. **CBLOCK – ENDC** komutları içerisinde birden fazla değişken tanımlamanız mümkün. Bu komut içerisinde 0x10 bellek gözünden başlayarak **pos, dc1** ve **dc2** değişkenlerini tanımlıyoruz. Daha önce gördüğümüz **EQU** komutu ile de bunları tek tek yazabilirdik. Sonuçta yapılan işlem aynı.

KOD 22.1: Değişkenlerin Tanımlanması

```
LIST        p=16F84
#include    "P16F84.INC"
CBLOCK   0x10  ; Değişkenleri yarat
 pos
 dc1
 dc2
ENDC
```

Daha sonra, her zamanki gibi, portları ayarlamak için Bank 1'e geçiyoruz. PORTB'nin 0-3 bitlerini çıkış yapıyoruz. Bunun için 0xF0='00001111' değerini **TRISB** yazmacına gönderiyoruz. '1'e denk gelen pinler çıkış olacaktır. Aynı şekilde, PORTA'nın tüm bacakları giriş yapılıyor. Hatırlarsanız, portun giriş için ayarlanan bacakları yüksek duruma gelir. Biz de giriş yapmak istediğimiz pinleri butonlar aracılığı ile toprağa göndererek girişi sağlayacağız.

Şimdi tekrar Bank 0'a geçiyoruz. PORTB'yi temizliyoruz ve PORTA'nın tüm uçlarını giriş için yüksek yapıyoruz. Ardından **pos** değişkenine '3' değerini, yani '00000011' değerini yüklüyoruz. Daha sonra, bu değişkeni sola ya da sağa öteleyerek, PORTB'ye göndereceğiz. Dikkat ederseniz, PORTB'ye sıra ile '1' değeri gönderecektik. Fakat **pos** değişkeni içerisindeki iki tane bite yan yana '1' gönderdik. Bunun sebebi, öteleme işlemini gerçekleştirirken, bir pine uygulanan '1' süresini uzatmak. Eğer sadece bir tane '1' değerini kaydırarak sinyal göndermeye çalışırsak, '1' uygulama süresi çok kısa olacağından, motor hareketi sağlıklı olmayacaktır.

KOD 22.2: Bankların Ayarlanması

```
start
   bsf   STATUS,RP0  ; Bank 1 Seç
   movlw 0xF0        ; PORTB'nin 0-3 bitlerini çıkış yap
   movwf TRISB
   movlw 0x00        ; PORTA'nın tüm uçlarını giriş yap
   movwf TRISA
   bcf   STATUS,RP0  ; Bank 0 Seç
   clrf  PORTB
   movlw 0xFF
   movwf PORTA
   movlw 3           ; W'ye '0011' değerini yükle.
   movwf pos         ; W'deki değeri pos a yükle
```

Sıra geldi programın ana döngüsüne. Burada sürekli PORTA'nın birinci ve sıfırıncı bitlerini test ediyoruz. Eğer butonlardan bir tanesine basılırsa, PORTA'nın, o butona bağlı olan biti '0' olacaktır. Biz de, bu '0' olma durumunu **BTFSS** komutu ile test ediyoruz. Bu komut belirtilen bit

eğer '1' ise bir alt satırı atlayacaktır. PORTA'nın tüm uçları giriş olduğu için bu pinler, butona basılmadığı sürece hep '1'dir. Eğer butona basılırsa, basılan bit '0' olacak ve bir alt satır uygulanacaktır. Bu alt satırlarda ise, döndürme fonksiyonlarını çağırıyoruz. Dolayısı ile hangi butona basılırsa o yöndeki döndürme fonksiyonu çağırılacaktır.

KOD 22.3: Ana Döngü

```
loop  btfss PORTA,0  ; Saat yönü butonunu test et
      call  stepcw     ; Eğer basılı ise STEPCW fonksiyonunu çağır.
      btfss PORTA,1    ; Ters-Saat yönü butonunu test et
      call  stepccw    ; Eğer basılı ise STEPCCW fonksiyonunu çağır.
      Goto  loop
```

Şimdi sırada döndürme fonksiyonları var. Burayı çok dikkatli bir şekilde okumalısınız. İlk olarak, bir adım saat yönünde çevirme fonksiyonunu inceliyoruz. Kaydırma işlemlerini **pos** değişkeni üzerinde gerçekleştireceğiz. Program ilk başladığında, **pos** değeri "0011" idi. Bunu sıra ile sola ötelediğimizi düşünelim. Değerler sıra ile, "0110", "1100" olacaktır. Bir daha sola ötelendiğinde ise, "11000" olacaktır. Biz ilk 4 biti (0-3) kullandığımız için baştan 5'nci bitin '1' olması bizi pek ilgilendirmiyor. Bu durumda sıfırıncı bitin tekrar '1' olması gerekiyor. Yani "1100"dan sonra, "1001" olmalı. Bunu **STATUS** yazmacının CARRY bitini kullanarak halledeceğiz. Sola ötelemelerde, **RLF** komutu en sondaki bit, yani 7'nci bit kaybolmasın diye bu değer otomatik olarak **STATUS** yazmacının CARRY bitine yazılır. Bu değer sonraki ötelemede sağdan tekrar girecektir. Biz ilk 4 (0-3) biti kullandığımız için, hiçbir zaman 7'nci (baştan sekizinci) bit üzerinde bir veri olmayacaktır. Bu nedenle bunu kendimiz CARRY bitine yazıyoruz. Şimdi durumu adım adım inceleyelim.

Kullandığımız **pos** değişkeninin içeriği "1100" olsun. Bu durumda **BTFSC** ile bu durumu test ediyoruz. Eğer 3'ncü bitimiz '1' ise, sonraki satırda bu '1' değerini CARRY bitine yazıyoruz. Sonra öteleme işlemi yapılıyor. CARRY bitinin değeri '1' olduğu için, öteleme yaparken bu değer en sağdaki bite ekleniyor. Yani son durum "1001" oluyor. Adından da anlaşılacağı gibi CARRY biti öteleme sırasında, en sondaki biti, en başa taşıyarak kaybolmasını önlüyor. Dolayısı ile 3'ncü bite gelen '1' değeri, ötelemeden sonra sağdan tekrar giriyor. **RLF** komutu **W pos** içeriğini bir sola öteleyerek son durumu **W** saklayıcısına yazıyor. Daha sonra bu değeri önce **AND**'liyoruz. Bunun sebebi, sola kaymış gereksiz bitlerden kurtulmak. İlk 4 bit bizi ilgilendirdiği için bu işlemden sonra, **W**'nun içeriğindeki ilk 4 bit içinde olan '1' değerleri kalacaktır. Daha sonra bu veriyi PORTB'ye gönderiyoruz. Sonra, bir miktar gecikme sağlıyoruz. PORTB'yi temizleyip tekrar ana döngüye dönüyoruz. Bir sonraki durumda ise sıra '1001' değerine geliyor. Burada da 3'ncü bitin değeri yine '1' olduğu için bu değer önce CARRY bitine aktarılıyor. Dolayısı ile öteleme işleminden sonra, bu bit sağdan tekrar giriyor.

KOD 22.4: Saat yönünde döndürme

```
stepcw   bcf   STATUS,C ;CARRY bitini temizle
   btfsc pos,3          ; Eğer 3. bit 1 ise sonraki satırı uygula ('1100' ve
'1001' durumu)
   bsf   STATUS,C       ; CARRY bitini 1 yap
```

```
    rlf   pos,W          ; pos içeriğini sola ötele
    andlw 0x0F           ; İlk 4 bit bizi ilgilendiriyor.(0-3)
    movwf pos
    movwf PORTB          ; pos un son halini PORTB'ye gönder
    call  delay          ; Gecikme sağla
    clrf  PORTB          ; PORTB'yi temizle
    return
```

Sırada sağa öteleme fonksiyonu var. Burada yaptığımız işlem diğerinden biraz farklı. **pos** değeri sağa öteledikçe, içeriği '1100' ise bu değer, önce '0110', daha sonra da '0011' olacaktır. Eğer sıfırıncı bit '1' ise yani **pos** değeri üzerinde, '0011' ya da '1001' değerleri var ise en sağdaki biti, 4'ncü bite yazacağız. Dolayısı ile bir sonraki ötelemede, 4'ncü bit 3'ncü bite geçeceğinden, en sağdaki bit kaybolmamış olacak. Örneğin, '1001' değerini sağa öteleyeceğimiz zaman, sıfırıncı bit '1' olduğu için bu değer 4'ncü bite yazılıyor. **pos** değişkeninin son durumu '11001' oluyor. **RRF** komutu ile içerik bir sağa ötelendiğinde, '1100' değeri elde ediliyor. En sağdaki bit ise CARRY bitine gidiyor ve bir sonraki ötelemede, en soldaki bitten tekrar giriyor (7'nci bit). Zaten bizi 3'ncü bitten sonrası ilgilendirmiyor. Yani burada CARY bitini kullanmıyoruz.

KOD 22.5: Saat yönü tersine döndürme

```
stepccw  bcf   STATUS,C ; CARRY bitini temizle
         btfsc pos,0          ; pos un 0. biti 1 ise alt satırı uygula ('0011' ve
'1001' durumu)
   bsf   pos,4          ; pos un 4. bitini 1 yap ('10011' ve '11001' durumu)
   rrf   pos,W          ; pos içeriğini sağa ötele
   andlw 0x0F           ; İlk 4 bit bizi ilgilendiriyor.(0-3)
   movwf pos
   movwf PORTB          ; PORTB'yi temizle
   call  delay          ; Gecikme sağla
   clrf  PORTB          ; PORTB'yi temizle
   return
```

Sıra geldi gecikme fonksiyonuna.

KOD 22.6: Gecikme alt programı

```
delay movlw 5  ; Bu değeri arttırdıkça motor yavaşlayacaktır
      movwf dc1
dl1   clrf    dc2
dl2   nop
      nop
      decfsz  dc2,F
      goto  dl2
      decfsz  dc1,F
      goto  dl1
      return
      END
```

YİRMİÜÇÜNCÜ BÖLÜM

23

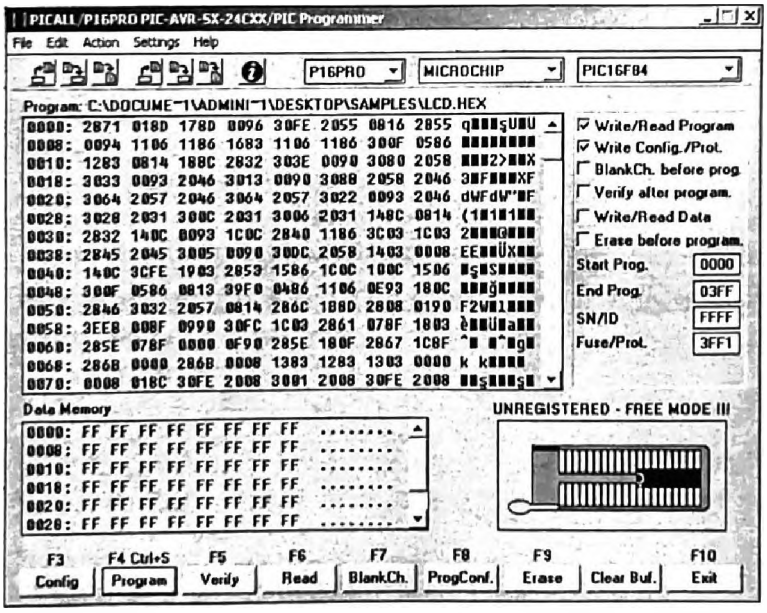
PIC BASIC

PIC Basic'e Giriş

Gerçekleştirdiğimiz birçok PIC uygulamasından bildiğiniz gibi PIC programları PIC Assembly dilinde yazılır ve derlenir. Assembly, doğrudan adresler ile programlandığı için, anlaşılması oldukça zordur ve programlama süresi daha uzundur. Bu nedenle BASIC gibi, daha düşük seviyeli diller ile programlanabilen, Basic Stamp, Basic Tiger gibi mikroişlemciler üretilmiştir. Fakat bu işlemcilerin maliyeti yüksektir. Bu nedenle birçok otomasyon uygulamasında PIC mikroişlemciler tercih sebebidir. PIC'ler için, Basic tabanlı işlemcilere alternatif olarak bazı yazılımlar geliştirilmiştir. Bu yazılımlar sayesinde, Basic ya da C gibi daha kolay diller ile istenilen uygulamayı derlemek mümkündür.

PIC'in içerisine bir program sadece Hex formatında yüklenebilir. Hex kodu ise assembly ile geliştirilen `.asm` uzantılı bir uygulama dosyasının derlenmesi sonucu elde edilir. PIC Basic gibi programlar da, Basic tabanlı dil ile geliştirilmiş bir kodu, `.asm` formatına dönüştürürler. Sonuçta siz programı Basic ile yazarsınız. Yazdığınız Basic programı arka planda önce `.asm` koduna dönüşür. Daha sonra hex koduna dönüşür ve PIC'e yüklenebilecek hale gelir. Bu kodu PIC üzerine yükleyebilmek için önceki uygulamalarımızda da kullandığımız bir PIC programlayıcısına ihtiyacınız vardır.

PIC Basic ile program geliştirmek için programlayıcının dışında, bir PIC Basic derleyicisine ve MPASM programına ihtiyacımız vardır. Alternatif PIC Basic programları Internet üzerinde mevcuttur. Ben PIC Basic derleyicisi olarak Crownhill firmasının, Proton+ Lite programını kullandım. Bu derleyici 50 satıra kadar ücretsiz olarak kullanılabiliyor. Bu programı, `http://www.picbasic.org` adresinden indirebilirsiniz. MPASM programını da, `http://www.microchip.com` adresinden indirmeniz mümkün.



ŞEKİL 23.1: PicAllW, PIC Basic programının MPASM ile yarattığı dosyayı PIC e yüklemeye yarıyor.

PIC Basic Uygulamaları

Uygulama I: PIC Basic ile Led Kontrolü

Birçok uygulamanın başında olduğu gibi, yine led kontrol etmek ile işe başlıyoruz. Daha önce de bahsettiğimiz gibi, eğer port üzerinden bir led yakıp söndürebiliyorsak, aynı port üzerinden birçok otomasyon uygulamasını gerçekleştirebiliriz demektir. PIC16F84 üzerine 2 adet port vardır. Bu uygulamada, PORTB üzerindeki 2 tane ledi yakıp söndüreceğiz. Bunun için port üzerindeki lojik voltajdan yararlanacağız. Herhangi bir pin üzerinden çıkış almak istiyorsak, o pini "yüksek" yani "1" durumuna getirmeliyiz. Yüksek durumda olan bir pin +5V çıkış veriyor demektir. Bu durum, aynı bilgisayardaki paralel port gibidir. Lojik voltaj konusu ile ilgili detaylı bilgiyi sitemizden elde edebilirsiniz.

PIC Basic ile istediğimiz portun istediğimiz pinini yüksek yapmak oldukça kolaydır. İstediğimiz biti yani pini, yüksek yapmak için **HIGH** komutunu, düşük yapmak için ise **LOW** komutunu kullanabiliriz. Komutun kullanımı aşağıdaki gibidir.

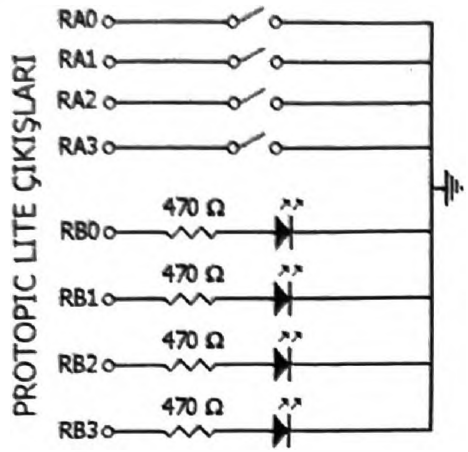
```
HIGH Port ya da Port.Bit
```

Örneğin **HIGH PORTB.3** komutu, PIC üzerindeki PortB'nin üçüncü bitini yüksek yani +5 Volt yapar. Bu durumda bu pine bağlı olan led yanacaktır. Uygulamada da yapılan işlem bunun aynısıdır. Önce, istenilen pin yüksek yapılır, daha sonra **PAUSE** komutu ile milisaniye bazında bekleme yapılır. Örneğin **PAUSE 1000** komutu, tam bir saniyelik bekletme sağlar. Ardından diğer pin yüksek yapılır ve tekrar beklenir. Daha sonra bu işlem kısır döngüye sokulur ve PIC çalıştırıldığında, sonsuza dek 2 tane led sıra ile yanıp sönmeye başlar.

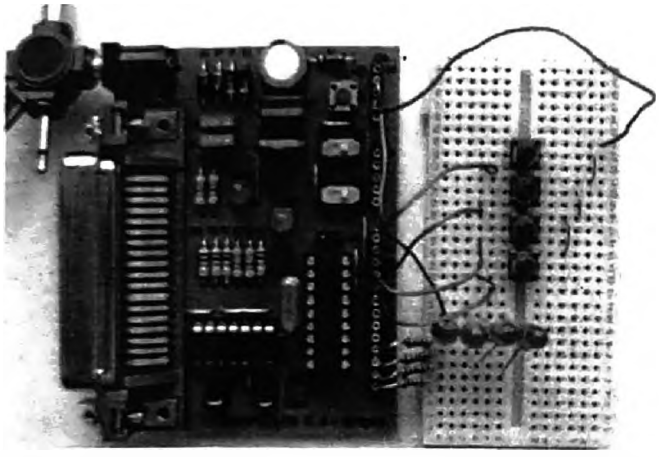
PIC Basic ile led kontrol kodu

```
loop:   High PORTB.3 ' PORTB.3'e bağlı ledi yakar
   Pause 500   ' 0.5 saniye bekletme sağlar
   Low PORTB.3 ' PORTB.3'e bağlı ledi söndürür
   Pause 500   ' 0.5 saniye bekletme sağlar
   High PORTB.2   ' PORTB.0'a bağlı ledi yakar
   Pause 500   ' 0.5 saniye bekletme sağlar
   Low PORTB.2 ' PORTB.0'a bağlı ledi söndürür
   Pause 500   ' 0.5 saniye bekletme sağlar
Goto loop   ' programın başındaki loop satırına döner
End      ' program sonu
```

Ledleri PORTB çıkışlarına, Şekil 23.2'deki gibi bağlayabilirsiniz. Port çıkışlarına ledlerin zarar görmemesi için 330 Ohm'luk dirençler kullanabilirsiniz. Şekil 23.2'de, PORTB'ye bağlı 4 adet led ve PORTA'ya bağlı 4 adet buton görüyorsunuz. Bu butonları şimdilik kullanmıyoruz. Yani bu butonları bağlamasanız da devremiz çalışacaktır.



ŞEKİL 23.2: Led kontrol devresi.



ŞEKİL 23.3: ProtoPIC programlayıcısının çıkışları üzerine kurulmuş deneme devresi.

Uygulama II: PIC Basic ile LCD Panel Kontrolü

PIC Basic ile uygulamalar gördüğünüz gibi oldukça kısa bir programlama ile geliştirilebiliyor. Daha önce Assembly ile gerçekleştirdiğimiz LCD panel kontrol uygulaması yaklaşık 250 satır kod içeriyordu. Bu uygulamamızda, PIC Basic'in işimizi ne kadar kolaylaştırdığını çok daha iyi göreceksiniz.

Programlamaya geçmeden önce, LCD panellerden kısaca bahsetmek istiyorum. LCD paneller günlük hayatımızın bir parçası gibidir. En basitinden bazı cep telefonlarında, hesap makinelerinde ve birçok elektronik eşyada bu teknoloji kullanılmaktadır. LCD paneller hafif ve az yer kaplarlar. Bazen basit bir veri çıkışı almak için koca bir monitöre ihtiyacınız olmayabilir. Yaptığınız devrelere küçük bir LCD panel bağlayarak devreden, kullanıcıya göndermek istediğiniz mesajları bu ekrana yazdırabilirsiniz.

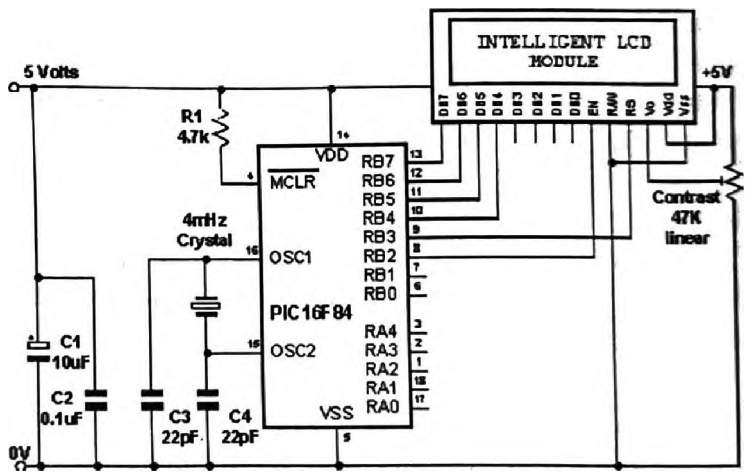
Standart LCD paneller, +5 Volt ile çalışır. Bu voltajı PIC deneme kartımızın üzerinden alabiliyoruz. Benim kullandığım LCD panelin modeli (2 X 16) SAMSUNG 1602D idi. Ya da kitaplarda sık rastlanan HITACHI HD44780 model LCD panel de işinizi görebilir. Genel olarak 2X16'lık ve 14 konnektörü bulunan LCD panellerin bacak bağlantıları aynı sayılır. Önemli olan

LCD panelin hangi pininin hangi işlev için kullanıldığını bilmektir. Standart bir LCD panelin üzerinde 14 tane konnektör bulunur. Bu pinlerin görevleri aşağıdaki gibidir.

TABLO 23.1: LCD Pinlerinin İşlevleri

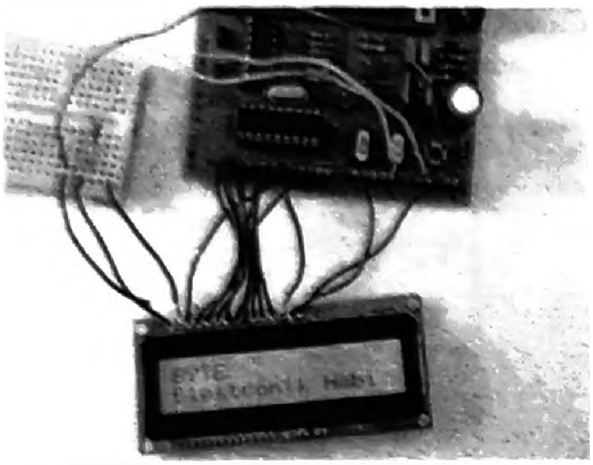
LCD Pinleri	Pin İşlevi
1	Toprak (Ground)
2	VCC +5 Volt
3	VO (Kontrast)
4	RS (Register Select)
5	R/W (Okuma ya da Yazma Modu)
6	E (Enable)
7-14	D0-D7 (DATA girişleri)
15*	LCD Panel ışığı (+5 Volt)
16*	Toprak (Ground)

* 15 ve 16 numaralı pinler bazı LCD panellerde bulunmayabilir. Eğer LCD panel ışıklı ise iç ampulü bu pinlerden yakmak mümkündür.



ŞEKİL 23.4: PIC ile LCD panelin 4-bitlik bağlantı şeması.

PIC Basic ile yazacağımız programda bu pinler ile ilgili hiçbir işlem yapmamız gerekmiyor. Tek önemli olan, PIC ile bağlantılarını doğru yapmak. Bu bağlantıları da Şekil 23.4'te görebilirsiniz. Şekil 23.4'te, PIC'in portları dışına bağlanmış bir sürü devre elemanı görüyorsunuz. Bu elemanlar bir PIC'i çalıştırabilmek için gereken temel devre elemanlarıdır ve bizim kullandığımız programlayıcı kartın üzerinde bu devre elemanları zaten mevcuttur. Bu nedenle bizi ilgilendiren sadece PORT çıkışlarındaki bağlantılardır.



ŞEKİL 23.5: LCD Panel ile PIC deneme kartının bağlantısı.

PIC BASIC ile LCD panel kontrol kodu

```
PRINT $FE , 1           'Ekranı Sil
PRINT $FE , 128         'Kürsörü sol üst köşeye taşı
PRINT  AT 1,1,"BYTE"    '1. Satıra BYTE yaz
PRINT  AT 2,1,"Elektronik Hobi"  '2. Satıra Elektronik Hobi yaz
END                'Programı sonlandır
```

PIC Basic altında LCD ekrana yazı yazdırmak için ihtiyacımız olan temel komut **PRINT**. Bu komutla birlikte **AT** parametresini kullanarak kürsörü istediğimiz konumdan başlatabiliyoruz. Bununla birlikte **$FE** parametresi ile panele bazı komutları gönderebiliyoruz. Bu komutların listesini PIC Basic programının 'HELP' seçeneğinde, indekse 'LCD' yazarak gelen sayfalardan elde edebilirsiniz.

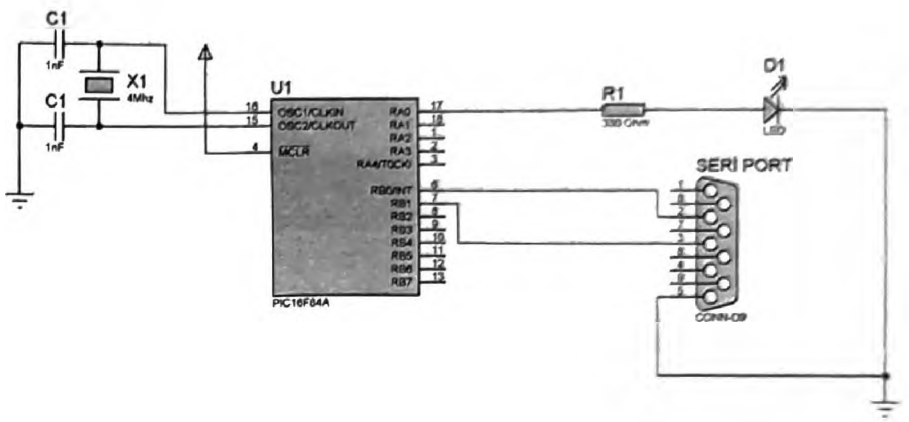
Gördüğünüz gibi Assembly ile 250 satırda yazdığımız uygulamanın aynısını PIC Basic ile 5 satırda gerçekleştirdik. PIC Basic'e geçmeden önce deneme amaçlı olarak herhangi bir Assembly programını MPASM ile derleyip, PicAllw ile ya da programlayıcınızın desteklediği bir yazılım ile PIC'e yükleyebildiğinizden emin olmalısınız.

PIC BASIC ile RS232 Terminali

Bu uygulamada PIC'e bilgisayarın seri portundan bağlanarak işlem yaptıracağız. PIC Basic ile terminal programı oluşturup Windows altındaki Hyperterminal programı ile, PIC'e bağlanacağız ve oluşturduğumuz menüden bir tane led kontrol edeceğiz.

Uygulamalarımızda kullandığımız PIC 16F84'ün üzerinde seri port bulunmamaktadır. Seri port uygulamaları için genelde 16F877 tercih edilir. Fakat PIC Basic programı, PIC 16F84 üzerindeki paralel çıkışlardan istediğimizi seri giriş ya da çıkış olarak ayarlamamıza imkan verir. İstediğimiz pinden seri çıkış alabilmek için, **SEROUT** komutu kullanılır. Bu komut asenkron şekilde seri data göndermeye yarar. Programda PORTB0 seri çıkış için kullanılmaktadır. Bu nedenle PORTB0 çıkışı seri portun girişine takılmalıdır. İki cihazın aynı kablo üzerinden tek yönde haberleşebilmesi için bir tarafın veri gönderirken diğer tarafın alması gereklidir. Bunun için

TxD pinine PIC'in giriş için ayarlanan pini, RxD pinine de PIC'in çıkış için ayarlanan pini bağlanmalıdır. PIC ile seri portun bağlantısını Şekil 23.6'da görebilirsiniz.



ŞEKİL 23.6: PIC ile PC'nin seri port ile bağlanması.

SEROUT Komutu

Bu komut **SEROUT PORT.PIN, BAUDMODE, DATA** şeklinde kullanılır. **PORT.PIN**, hangi portun kaçıncı pininin çıkış olarak kullanılacağını gösterir. **BAUDMODE** ise bağlantı parametrelerini içerir. Buradaki bağlantı parametreleri, Hyperterminal altından PIC'e bağlanırken kullanılan parametreler ile burada kullanılan **BAUDMODE** birbirleri ile aynı olmalıdır.

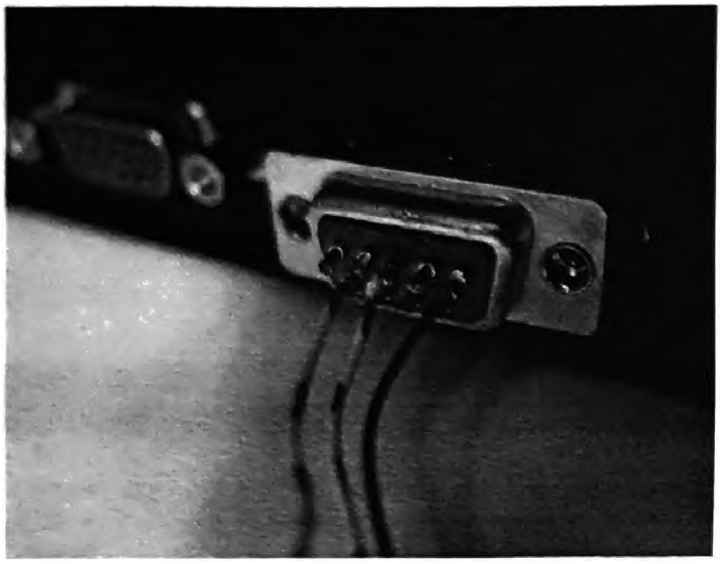
TABLO 23.2: PIC İçin BAUDMODE Parametreleri

BaudRate	8-bit no-parity inverted	8-bit no-parity true	7-bit even-parity inverted	7-bit even-parity true
300	19697	3313	27889	11505
600	18030	1646	26222	9838
1200	17197	813	25389	9005
2400	16780	396	24972	8588
4800	16572	188	24764	8380
9600	16468	84	24660	8276

Bizim programımız içerisinde, 16780 parametresi, 2400 Baudrate, 8 bit, no-parity, inverted şeklinde bir bağlantı kurar. Bu terimler, terminal programında bağlantı yaratırken karşınıza çıkacaktır. Sağlıklı bağlantı için aynı bu şekilde seçmelisiniz.

Bilgisayar seri portlarının kullandığı RS232 sinyallerin elektriksel özelliği, 5 Volt lojik "1" ve 0 Volt lojik "0" değerine karşılık gelen Standart TTL mantığına benzememektedir. RS232'de –12 Volt lojik "1" +12 Volt lojik "0" değerine karşılık gelir. Bu özellik daha uzun mesafeli kablolarda, güçlendirme gerektirmeden iletişim sağlar. RS232 ile birlikte çalışan birçok devrede bu voltaj farklarını dengeleyici ek yongalar kullanılır. Bunlardan en yaygın olarak kullanılan

MAXIM'in ürettiği MAX232 sürücüsüdür. Fakat kullandığımız PIC'in çok iyi I/O (Giriş/Çıkış) yetenekleri dolayısıyla, uzun mesafelere çıkılmadığı sürece böyle çevirici sürücüsüne ihtiyaç yoktur. Yine de akımı sınırlamak amacıyla devreye istenirse direnç eklenebilir.



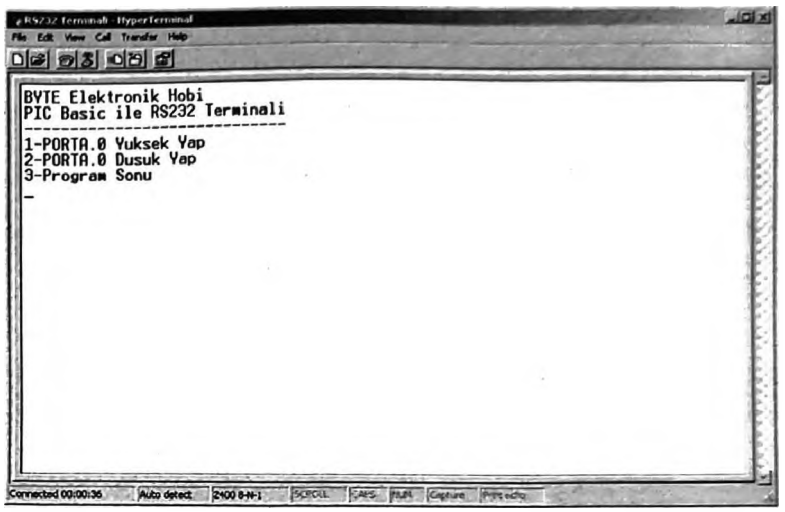
ŞEKİL 23.7: Seri port ile bağlantı için 3 tel kablo yeterlidir.

PIC Basic terminal programı kodu

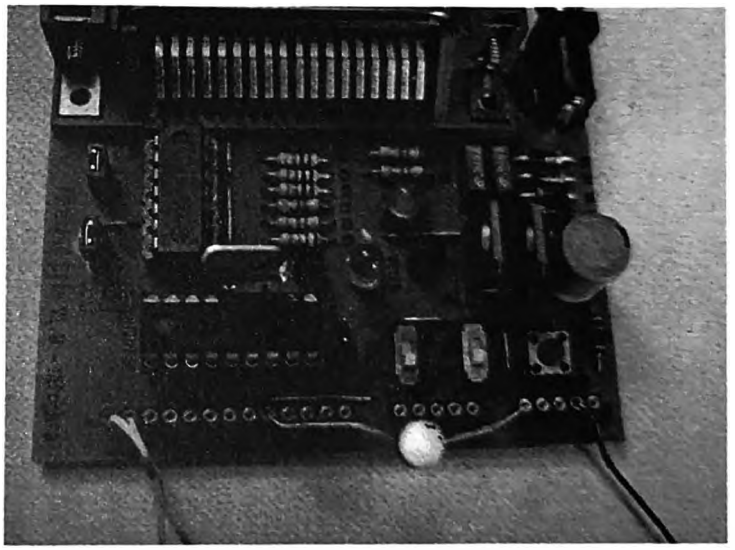
```
dim serindata
SEROUT PORTB.0 , 16780 , ["BYTE Elektronik Hobi", 13, 10]
SEROUT PORTB.0 , 16780 , ["PIC Basic ile RS232 Terminali",13,10]
SEROUT PORTB.0 , 16780 , ["----------------------------",13,10]
SEROUT PORTB.0 , 16780 , ["1-PORTA.0 Yuksek Yap ",13,10]
SEROUT PORTB.0 , 16780 , ["2-PORTA.0 Dusuk Yap",13,10]
SEROUT PORTB.0 , 16780 , ["3-Program Sonu",13,10]
loop:
SERIN PORTB.1 , 16780 , [serindata]
select case serindata
      case "1"
           High PORTA.0     ' PORTA.0'a bağlı ledi yakar
      case "2"
         Low PORTA.0      ' PORTA.0'a bağlı ledi söndürür
      case "3"
       SEROUT PORTB.0 , 16780 , ["Program sonlandi. Terminal baglantisini
kesebilirsiniz.",13,10]
       end
end select
'SEROUT PORTB.0 , 16780 , [serindata] DUMMY TERMINAL İÇİN
        Goto loop
```

SERIN komutu ise, **SEROUT** komutuna çok benzer. Sadece içine verilen değişkene, seri olarak aldığı veriyi aktarır. Program içerisindeki **serindata** değişkeni bu işe yarar. Elde edilen değer bu değişkene yüklendikten sonra **CASE** karşılaştırma yöntemi ile test edilmektedir. Eğer "1"

girilmiş ise PORTA0'a bağlı ledi yakmakta, "2" ise söndürmektedir. "3" ise programı sonlandırmaya yarar. Programın sonunda devre dışı bırakılmış kod satırı, programın Dummy Terminal özelliğini açmaya yarar. Bu satırı aktif hale getirirseniz, klavyeden PIC'e gönderdiğiniz her karakter, PIC'ten bilgisayara geri döner. Dolayısı ile, bastığınız her karakteri terminal ekranından izlemenizi sağlar. Bu şekilde bir porttan aldığı veriyi doğrudan diğer porta aktarmaya yarayan terminal bağlantısı Dummy Terminal olarak adlandırılır.



ŞEKİL 23.8: PIC ile RS232 terminal programı.



ŞEKİL 23.9: Seri porttan gelen komutlar ile bir tane led sürüyoruz.

YİRMİDÖRDÜNCÜ BÖLÜM

24

PIC BASIC İLE SERVO MOTOR KONTROLÜ

Servo Motor Nedir?

Normal hobi servoları, radyo kontrollü araba, uçak, tekne gibi sistemlerde yaygın kullanılır. Servo, konum kontrolü yapmak üzere geliştirilmiş bir çeşit motordur. Hareket aralıkları genelde 90 derece ile 180 derece arasındadır. Hem makul fiyatı hem de yüksek torku, konumlu hareketler için vazgeçilmezdir. Elektriksel olarak kumanda edilirler.



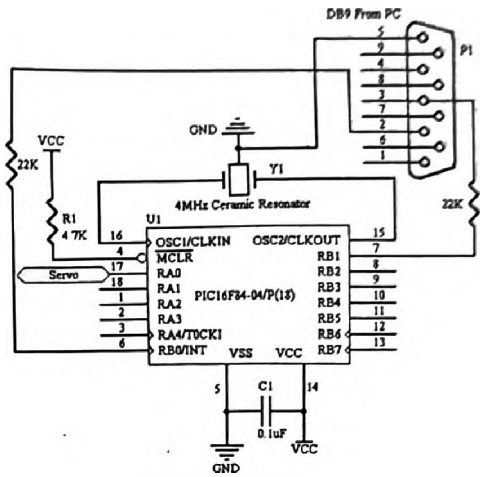
ŞEKİL 24.1: Servo motor.

Bir servo motor yapı olarak dört kısımdan oluşmaktadır. DC elektrik motoru, planetar dişli sistemi, geri besleme potansiyometresi ve DC motor pozisyon kumanda elektroniği. DC motor herhangi bir oyuncak motorundan farklı olmayan çift mıknatıslı bir statora ve fırçalı bobin rotora sahiptir. Motor mili 1:200 ile 1:300 arası dönme oranına sahip bir dişli sistemine bağlanır; bu sayede oldukça yüksek bir tork değerine ulaşılır. Dişli sisteminin çıkışında bir 5k'lık potansiyometre, mil konumunu elektronik kumanda devresine iletir. Elektronik devrenin görevi mil konumunu gelen veri konumuna gelinceye kadar motoru iletimde tutup tam yerinde durdurmaktadır. Servo kumanda motorları model uçak gibi hareketli yüzeyleri bulunan modellerin hareketini kumanda çubuklarının hareketine bire bir uyarlamak için kullanılırlar. Servolar modellerde ağırlık ve hacim sorunu önemli bir problem olduğu için son derece hafif ve küçük olmak zorundadır. Bu nedenle akım çekişi düşük olmalıdır, 4.8V gibi bir voltaj kaynağı ile beslenebilmelidir. Fakat tüm bu kısıtlayıcı kriterlere rağmen 1 metrelik bir moment kolu uzaklığında 100 gram ağırlığı oynatabilmelidir. Bu derece küçük ve sınırlı bir cihazın böylesine güçlü olması insana şaşırtıcı ve imkansız gelmektedir, ama en ucuz servo bile bu kriterleri fazlasıyla sağlar.

Servo motorlar şu anda piyasada lineer veya dairesel bir hareketi kumanda edebileceğiniz en ucuz ekipmandır. Bu kriter de servoyu biz amatörlere ve özellikle robotikle ilgilenenlere çok cazip kılmaktadır.

Elektronik devre servonun konumu algılamak için DC motor kontrolü bölümünden hatırlayacağınız PWM (pulse width modulation) darbe genişlik modülasyonu tekniğinden yararlanmaktadır. Kumanda devresi kumanda çubuğunun konumuyla doğru orantılı olarak 1 ile 2 milisaniye arasında dalga genişliği değişen bir sinyali her 20 milisaniyede bir servoya gönderir. 1 milisaniye tam sol, 2 milisaniyede tam sağ pozisyonu ifade eder. Servo içindeki elektronik devre ilk önce gelen darbelerin darbe genişliğini ölçer, daha sonra potansiyometre konumuna bakar ve kendi darbe osilatörünün darbe genişliği gelen darbelerle eşitlenene kadar motoru

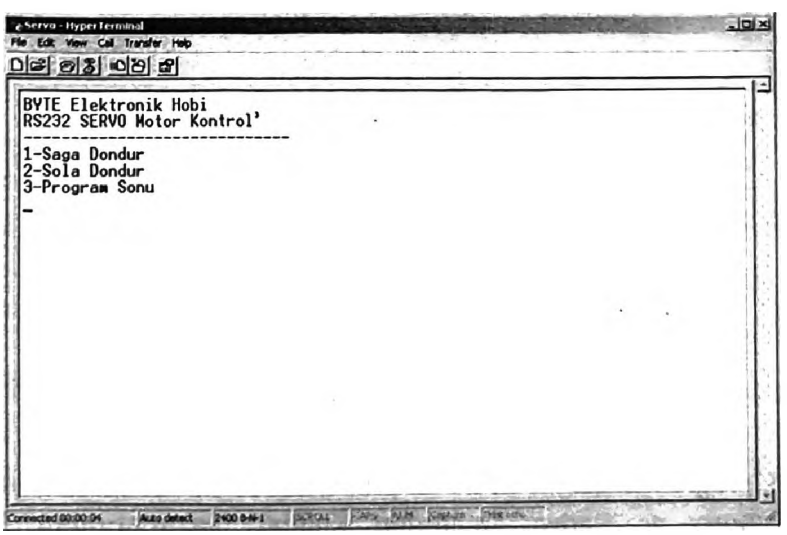
hareket ettirir.Motorun durduğu konum sizin kontrol çubuğunu tuttuğunuz konumla bire bir aynıdır. Servolarda üç adet kablo dışarı çıkar. Bunlardan kırmızı olan +4.8V-5.0 V arası besleme, siyah olan şase yani Ground, diğer kablo ise (turuncu yeşil veya beyaz olabilir) data girişidir.



ŞEKİL 24.2: Seri port, servo ve PIC devresinin bağlantısı.

Seri Port ile Servo Kontrolü

Yazacağımız terminal programı Bölüm 23'teki ile aynı sayılır. Bu nedenle PIC ile nasıl seri iletişim kurulduğundan bu bölümde bahsetmeye gerek görmüyorum. PIC Basic ile servo kontrolü oldukça basittir. PIC Basic sırf bu iş için halihazırda bir komut içermektedir. Komutun kullanım şekli, `SERVO port.pin, donme_degeri` şeklindedir. Devrede servoyu sürmek için PORTA.0 pini kullanılmaktadır. `donme_degeri` ise, uygulanan milisaniyeler ile alakalıdır. 500 değeri motoru bir yöne döndürmeye, 2500 değeri ise tam zıt yöne döndürmeye yarar. Bu değerler ile ilgili detaylı bilgi PIC Basic programının yardım bölümünde bulunmaktadır.



ŞEKİL 24.3: Servo kontrolü terminal yazılımı.

Programda, terminal sürekli bekleme halindedir. Eğer "1" tuşuna basılı tutulacak olursa motor sağa doğru dönecektir. "2" tuşuna basıldığı zaman, tam zıt yöne dönecektir.

PIC Basic ile servo motor kontrol kodu

```
dim serindata
SEROUT PORTB.0 , 16780 , ["BYTE Elektronik Hobi", 13, 10]
SEROUT PORTB.0 , 16780 , ["RS232 SERVO Motor Kontrolü",13,10]
SEROUT PORTB.0 , 16780 , ["-----------------------------",13,10]
SEROUT PORTB.0 , 16780 , ["1-Saga Dondur ",13,10]
SEROUT PORTB.0 , 16780 , ["2-Sola Dondur",13,10]
SEROUT PORTB.0 , 16780 , ["3-Program Sonu",13,10]
loop:
SERIN PORTB.1 , 16780 , [serindata]
select case serindata
      case "1"
            servo PORTA.0,500
         pause 15
      case "2"
         servo PORTA.0,2500
         pause 15
      case "3"
       SEROUT PORTB.0 , 16780 , ["Program sonlandi. Terminal baglantisini
kesebilirsiniz.",13,10]
        end
end select
        Goto loop
```



ŞEKİL 24.4: Servo kontrol devresinin PIC deneme kartı üzerine bağlanmış hali.

Konu ile ilgili farklı örnek programlar için, bu bölüm için kısmen kaynak olarak kullandığım, Antrak Gazetesi (Ankara Telsiz ve Radyo Amatörleri Kulübü) yazarlarından Barbaros Aşuroğlu'nun hazırladığı `http://www.antrak.org.tr/gazete/081999/barbar.htm` adresini ve

`http://www.rentron.com/SerialServo.htm` adreslerini incelemenizi öneriyorum. Bu sitede bulunan ve Şekil 24.5'te gördüğünüz Visual Basic ile servo kontrol programı, bizimkisine çok benzer bir devre kullanılarak yapılmıştır.



ŞEKİL 24.5: Rentron'un sitesindeki servo motor kontrol programı.

YİRMİBEŞİNCİ BÖLÜM

25

BASIC TIGER

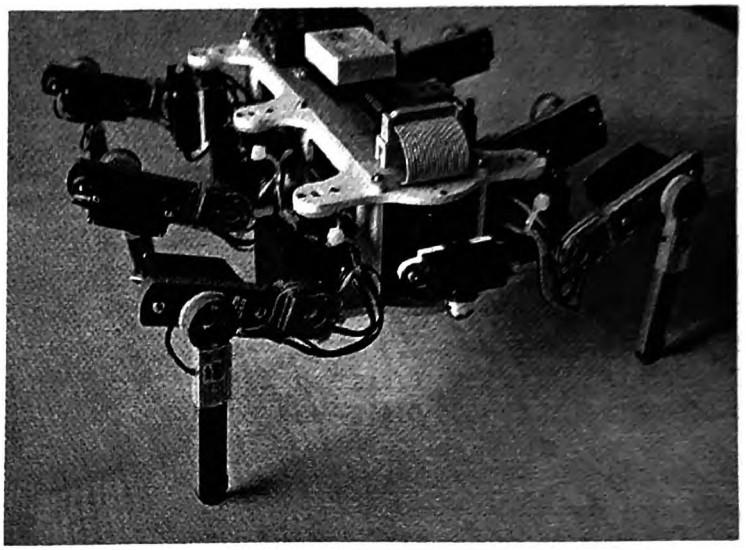
Tek Çipte Bilgisayar

PIC kullanarak, bilgisayarın paralel portu ile geliştirdiğimiz bütün uygulamaları gerçekleştirebiliyorduk. Fakat birçok otomasyon uygulamasında, PIC kullanmak programlama süresi ve programlama dili açısından bazen tercih edilmemektedir. Çünkü PIC programlayabilmek için, Assembly bilmek gerekmektedir. Halbuki, bu işi Basic gibi bir programlama dili ile yapmak da mümkün. Bunun için birçok ürün var. Örneğin Basic-Stamp, Java-Stamp gibi ürünler, robotik uygulamalarda sık tercih edilmektedir. Bu bölümde inceleyeceğimiz ürünün adı Basic Tiger. Bu ürün aynı zamanda Kısım 3'teki Mobil Araştırma Robotu projesinde de kullanılmıştır. Üründen kısaca bahsetmek istiyorum.

Alman kökenli Basic Tiger, tek bir çip içerisinde bütünleştirilmiş bir bilgisayar. Ancak bu bilgisayarın en büyük modeli bile sadece 63x41x12mm boyutlarında, 50 gram ağırlığında. IBM'in "kişisel bilgisayar" olarak 1982'de piyasaya sürdüğü PC'lerin BASIC'le programlama, endüstri standardı klavye, fare, ses, yazıcı ve seri iletişim bağlantıları gibi bazı temel özelliklerini almış durumda. Basic Tiger, bu bağlantılara ek olarak ev, iş ve genel çevremizde karşılaştığımız sıcaklık, akım, darbe ve ağırlık ölçümü, açma-kapama şalterleri, GPS, kart okuyucular, motor kontrolü gibi değişik işlevlere uyarlanabilir bağlantılar içeriyor. Çeşitli otomasyon ve ölçüm amaçları için kullanılabilecek bu bağlantıları, gerçek zaman kontrollü (real time) ve çok işlemli (multitasking) olarak denetleyebiliyor. Örneğin, programı işlemci üzerinde çalıştırmaya başladığınızda aynı Visual Basic'teki gibi yukarıdaki Toolbar'dan, Stop'a basarak durdurup debug edebiliyorsunuz. En iyi özelliklerinden biri, 5V'ta 50mA'lik bir adaptör bağlandığında, 20ms içinde çalışmaya başlayabilmeleri. –20 ile +850C arasında sıcaklıkla, sarsıntıyla ve ısınmayla ilgili bir sorunları yok. Pil destekli bellekleri 2MB'a, flash kapasiteleri ise 4MB'a kadar çıkabiliyor. Standart ünite üzerinde dört analog giriş, seri arabirim, iki adet PWM çıkışı, metin ve grafik LCD çıkışları ve dijital giriş/çıkışlar bulunuyor. Giriş/çıkış sayısı 4096'ya kadar artırılabiliyor. Fiyatları ise 50 Euro'dan başlıyor. Bu çip, birçok robotik uygulamada da tercih ediliyor. Örneğin, Şekil 25.1'de Basic-Tiger ile geliştirilmiş bir robot görüyorsunuz. Dev bir örümceği anımsatan bu makine, küçük bir robot. 18 santimetre uzunluğunda ve 800 gram ağırlığındaki bu robot Viyana Teknik Üniversitesi'nde öğrenci olan Hanns Tappeiner tarafından yapılmış. Arin'de biri kafada diğerleri bacaklarda olmak üzere toplam 19 step motor kullanılması sayesinde her bacağın 3 serbestlik dereceli hareket edebilme becerisi var. Bu ise hemen her türlü zemin ve engel üzerinde hareket edebilmek anlamına geliyor. Ayrıca 14 tane sensörü var. İşlemcisi ve pillerini üzerinde taşıması sayesinde bütünüyle otonom olarak hareket edebilmekte ve bu hareketleri sırasında bilgi toplayabilmekte. Toplanan bilgiler ise daha sonra PC'ye aktarılmakta. Bunca sensör bilgisini işleyip motor denetimini yapan ve verileri toplayan işlemci yani "beyin" ise Tiny Tiger E isimli bir çipten ibaret. 39x28 x10 mm boyutlarındaki bu çip tek başına komple bir bilgisayar görevi görüyor. Ayrıca bu çipin 512 KB'lık belleği ve yine 512 KB'lık program alanı var. Arin'in ek bilgilerini ve hatta hareket halindeyken çekilmiş filmini `www.tappotec.com` sitesinde bulabilirsiniz.

Programın editörü ise oldukça kullanışlı. Komutları ve sabitleri farklı renklere ayırabiliyor. Programlama süresi ise oldukça hızlı. Örneğin, LCD programını, çipe yüklemek 40 saniyenin altında sürüyor. En iyi özelliklerinden bir tanesi ise, real time debug yapabilmeniz. Programı durdurup, kaldığı yeri görebiliyorsunuz. Programlama ünitesi seri kablo ile bilgisayara bağlanı-

yor. Derleyici, Basic Tiger'ı çok rahat bir şekilde görüyor. Ayrıca Basic Tiger'ın durumunu aktif bir status penceresinden izleyebiliyorsunuz.



ŞEKİL 25.:1. Basic Tiger'la geliştirilmiş bir örümcek robot.

BASIC Tiger ile LCD Panel ve LED Kontrolü

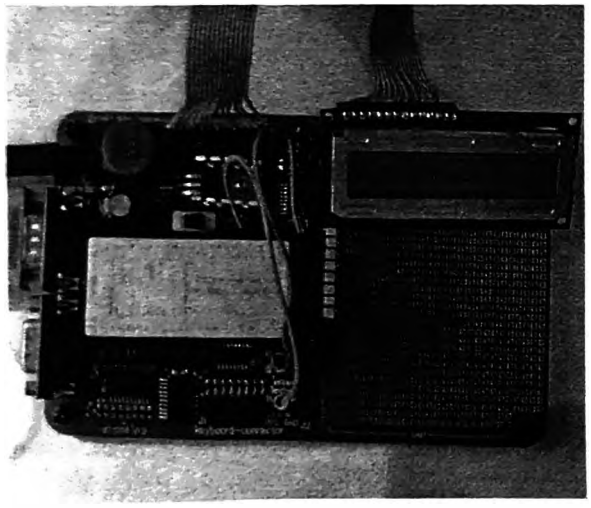
LCD panel programlamak gibi komplike bir uygulamanın ne kadar kolay gerçekleştirilebildiğini görmeniz için bu konuyu seçtim. İleride daha da komplike uygulamalar gerçekleştireceğiz. Ayrıca bu uygulamada, 2 tane de led sürüyoruz. Bu programda, ledlerin bağlı olduğu portun 0'ncı ve 7'nci bit'lerine 500ms aralıklarla lojik "1" gönderiyoruz. Hatırlarsanız, PIC üzerinde gecikme sağlamak için bir alt program yazmak gerekiyordu. Burada ise gereken sadece **WAIT_DURATION 500** yazmak. Programlama kartı aynı zamanda bir deneme kartı. Board üzerinde dahili 8 adet led bulunmakta. Dolayısı ile ledleri buradan izleyebiliyorsunuz. PIC ile LCD panele yazı yazdırmak için yaklaşık 400 satırlık Assembly kodu yazmamız gerekiyordu. Basic Tiger ile bu işlemi sadece 30 satırda bitirebiliyoruz. Üstelik sadece BASIC komutları kullanarak. Aşağıdaki program örneği, LCD ekrana "BYTE EHobi" yazdırmaya yarıyor. Bunların dışında kart üzerinde, röle ve adım motor kontrolü için daha önceden de kullandığımız ULN2003 devresi var. Ayrıca, MAX232 destekli seri port ve 8 adet dahili led içeriyor. Ayrıca, devreye eklemek isteyeceğiniz diğer devre elemanlarını da yerleştirebileceğiniz boş bir board da üzerinde geliyor. Basic Tiger esnek ve hızlı ürün geliştirmek isteyenler için çok uygun bir ürün. Zengin Basic kütüphanesi çevre birimleriyle bağlantı ve iletişim için yazılım geliştirme süresini haftalardan saatlere indiriyor.

Ayrıca birçok geliştirme seti ve geniş ürün yelpazesi var. Ürün hakkında detaylı bilgiyi **www.lima.com.tr** adresinden elde edebilirsini

Basic Tiger ile LCD panel kontrolü

```
#define LCD2_2_8 12 '1 x 16 = 2 x 8 LCD
#define LCD2_2_20 15 '1 x 16 = 2 x 8 LCD
```

```
#define UFCO_LCD_RESET 0B0H 'LCD'yi Resetle
TASK MAIN
'Ekran(LCD) + Klavye + Buzzer Sürücüsünü Yükle (TINY-Tiger)
'Buzzer Kapalı !
INSTALL DEVICE #1, "LCD1.TDD", LCD2_2_8, 0, 0, 0, 0, 0, 0h, 0
'Ekranı Resetle
PUT #1, UFCO_LCD_RESET
'Bekle
WAIT DURATION 2'Kürsor'u Kaldır
PRINT #1,"<1Bh>c"; CHR$(0); "<F0h>"
WAIT DURATION 2
'Ekranda Üst Köşeye Konumlan ve Mesajı (BYTE E-Hobi) Yaz
PRINT #1,"<1Bh>A"; CHR$(0); CHR$(0);"<F0h>BYTE E-Hobi";
'DIR_PORT 8,0 'Port 8'i (Pin 80-87) Çıkış Yap
'OUT 8,255,01010101b 'Çift Bitleri Yukarı Çek
DIR_PIN 8,0,0 'Port 8'in 0 numaralı bitini (80) Çıkış Yap
DIR_PIN 8,6,0 'Port 8'in 6 numaralı bitini (86) Çıkış Yap
LOOP 9999999 'Dön
OUT 8,00000001b, 0 '80'yı Aşağı Çek
OUT 8,01000000b, 255 '86'yı Yukarı Çek
WAIT_DURATION 500 '500 ms Bekle
OUT 8,00000001b, 255 '80'yı Yukarı Çek
OUT 8,01000000b, 0 '86'yi İndir
WAIT_DURATION 500 '500 ms bekle
ENDLOOP
```



ŞEKİL 25.2: BASIC Tiger ile LCD panel kontrolü.

BASIC STAMP

BASIC Stamp'ler

"Beyin", prensip olarak, bildiğimiz bir beyin gibi çalışır. Duyumu veya veriyi, olguya ilişkin ölçümü alır, işler ve karar üretir. Dış dünyanın olgularından alınan duyumlar, işlenmek üzere beyne iletilirler. Tersi yöndeki faaliyet ise, beyinde üretilen kararın dış dünyaya uygulanmak üzere gövdenin ilgili elemanlarına gönderilmesi şeklinde gerçekleşir. Dolayısıyla, dış dünyadan giren veriler yeniden dış dünyaya çıkar. Yani, beyin ile dış dünya arasında giriş ve çıkış diye adlandırdığımız kanallar üzerinden iletişim dediğimiz bir süreç gerçekleşir. Beyin, dış dünyayla arabirimli ve etkileşimli durumdadır. Beynimizin dur durak bilmeksizin yaptığı şey karar üretmektir. Beynimiz, karar üretmek için gördüklerimizi, işittiklerimizi, dokunduklarımızı, kısaca algılarımızı temel alır. Mikrokontrolörler de beynimiz gibi iş görürler. Girişler temelinde dış dünyayı çekip çevirirler.

Mikrokontrolörler programlanabilen cihazlardır. Yani, kod ya da program dediğimiz ve belli durumlar gerçekleştiğinde yapacakları işleri anlatan bir komutlar dizesini içlerinde taşırlar. BASIC Stamp, öğrenilmesi son derece kolay bir dil olan BASIC ile programlanır. Programlar, mikrokontrolörlerin dış dünyada olup bitenlere tepki vermesini sağlayan araçlardır. Basic Stamp ile yapacağınız deneyler için hazır deney setleri mevcuttur. Bu setler üzerinde, kolaylıkla devrenizi oluşturabilir ve Basic Stamp'le birlikte kontrol ya da veri toplamaya yönelik uygulamalarınızı geliştirebilirsiniz.

BASIC Stamp yaygın kullanılan bir mikrokontrolördür. Devre yapısı son derece inceliklidir. Tüm elemanlar oldukça küçük bir kart (PCB, devre kartı) üzerinde yer alırlar. Devre kartının boyutu hakkında bir fikir vermesi açısından, BASIC Stamp'in "entegre devre" denilen cihazların büyük bir çoğunluğuyla hemen hemen aynı büyüklükte olduğunu söylemek yeterlidir.



ŞEKİL 26.1: BASIC Stamp II modülü bir posta pulu boyutlarındadır.

BASIC Stamp için program yazarken BASIC dilinin özel bir biçimi olan PBASIC kullanılır. Birçok mikrokontrolörün gerektirdiğinden farklı olarak, PBASIC ile programlamayı öğrenmek çok kolaydır. Elinizde BASIC Stamp varsa, basit devreleri dakikayla ölçülen sürelerde oluşturmak ve aynı hızla programlamak sıradan bir iştir. Oysa, karmaşık devreleri bu kadar kısa sürelerde gerçekleştirmek mümkün değildir. Böyle durumlarda en çok ihtiyaç duyulan şeylerin başında "beyin" gelir. Eğer bu "beyin" BASIC Stamp ise, en çetrefilli tasarımları, gönül rahatlığıyla ve en çekici şekillerdeki ticari ürünler olarak satışa sunabilirsiniz.

Bu bölümde Basic Stamp ile temel bilgi olması amacı ile iki tane led kontrol eden basit bir program yazacağız. Basic Stamp uygulaması geliştirebilmek ve programlayabilmek için ilk olarak bir programlama kartına ihtiyacımız olacaktır. Ayrıca adı üstünde, bir tane de Basic Stamp'e ihtiyacımız var. Eğer Basic Stamp ile uygulama geliştirmek istiyorsanız, temelde bunlara sahip olmanız gerekir. Benim tavsiyem, tüm bu malzemeleri içeren, Infogate firmasının oluşturduğu "Basic Stamp Temel Eğitim Seti"ni almanız. Bu setin fiyatı 149$. Bu set içerisinde örnek uygulamamız için gereken tüm malzemeler mevcut.

Daha önce bilgisayar programı yazanların bu başlığın ilk bölümüne göz atmalarını, daha önce program yazmamış olanların da korkuya kapılmadan aynı bölümü dikkatle takip etmeleri yeterlidir. Çünkü program yazmak sanıldığı kadar zor bir iş değildir. Bir program, yani mikrokontrolör programı veya yonga kodu, mikrokontrolör tarafından belli bir sıraya göre yürütülen bir komutlar dizisinden ibarettir. Klavye ve monitör yardımıyla PC ortamında oluşturulan bu "kodlar", programlama kablosu üzerinden mikrokontrole yüklenir. Bundan sonrası, programın BASIC Stamp içinde çalışmaya başlamasıdır. Programın yazımı hatasız ise, cihazın çalışmasında sorunla karşılaşılmaz. Yazımda hata varsa, bu hataların giderilmesi gerekir. Başından sonuna programlama sürecinde insana "saç baş yolduracak" bir evre varsa, o da hata giderme evresidir. Hatayı gidermeye çalışmak veya zahmeti en aza indirgemenin en iyi yolu, programı tasarlarken ve yazarken en yüksek düzeyde uyanık olmaktır. En kötü hata, programın öngörülmeyen durumlar nedeniyle umulan sonucu üretmemesine sebep olan ve deneme aşamasında bile fark edilememe ihtimali bulunan hatalardır. Bu tür yazılım hataları İngilizce bug (böcek) ismiyle dile getirilmektedir. Böyle hataları gidermek de "böcek temizliği" (debug) yapmaktan farksızdır.

Program yazarken akılda tutulması gereken ilk şey, komutların sırasıdır. İlk iş olarak Eğitim Kartının 9 volt pille veya şebekeyle bağlantısını kurun. Sonra sert kabloyu PC'ye bağlayın. Daha sonra, büyük yonga kartın altına gelecek şekilde BASIC Stamp II'yi Eğitim Kartına takın. PC'yi açın. BASIC Stamp yazılımı DOS veya Windows 95/98/NT4.0 işletim sistemlerinden herhangi birinde çalışır. Burada kural olarak, Windows 95 ortamında çalıştığımız varsayılmıştır. Diski PC'nizdeki bir klasöre kopyalayın. BASIC Stamp simgesini çift tıkladığınızda zaman çalışmaya başlayan program "Stamp Editörü" olacaktır. Bu editör size program yazmanızı ve bir kablo vasıtasıyla PC'den BASIC Stamp yongasına program yüklemenizi sağlayacaktır.

Görüntüye gelen pencere boştur. Daha önce de belirtildiği üzere, program yazımı, PC'nin monitör, klavye, vb gibi "insan arabirimi" olarak anılan aletler aracılığıyla yapılır. Program editör penceresinde yazılır. Daha mikrokontrolöre gönderilir, yüklenir ve BASIC Stamp tarafından yazıldığı sıraya göre çalıştırılır.

BASIC Stamp ile Led Kontrol Uygulaması

BASIC Stamp ile led kontrolü

```
output 0
reblink :
   out0 = 0
pause 1000
   out0 = 1
```

```
    pause 1000
goto reblink
```

Programda ilk komut **output** komutudur. Sinyaller (yani P0-P15) "giriş" ya da "çıkış" olarak tanımlanabilir. Aslında, mikrokontrolörden ledi "yakıp söndürmesini" istemekle, ondan "dünyevi" bir işi çekip çevirmesini istiyoruz. Dolayısıyla P0'a, "çıkış", yani output olmasını söylemiş oluyoruz. İlk komut, **output 0**, şu anlama gelir: P0'ı çıkış yap. (Demek ki, eğer P1'in çıkış olmasını isteseydik, **output 1** diye yazmamız gerekecekti.) İkinci komut, **reblink :**, özü itibariyle bir komuttan ziyade bir etiket, programın belli bir kısmını gösteren bir işarettir. BASIC Stamp'teki 5 no'lu bacak, ya da bizim P0 dediğimiz sinyal, bir çıkıştır. Bilgisayar düzeneklerimizde bacakların voltajları ya "yüksek" (high) ya da "düşük" (low) olur. Yani bir bacakta belli bir anda ya yüksek voltaj ya da düşük voltaj vardır. Voltaj düzeyi, yüksek için 1 ve düşük için 0 rakamlarıyla da gösterilmektedir.

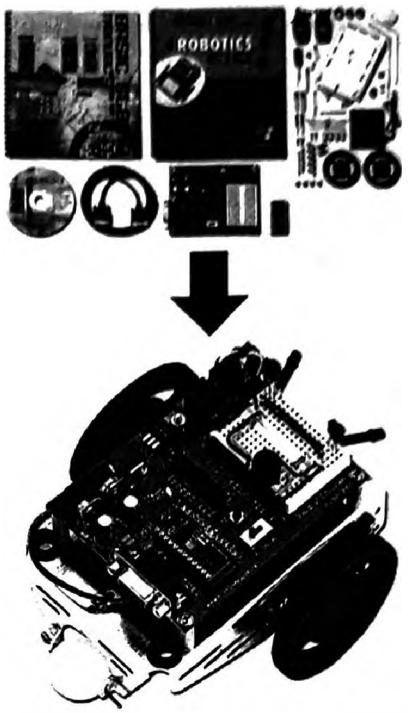
BASIC Stamp'ın 16 bacağı (P0-P15) gerek giriş gerekse çıkış olarak düzenlenebilir. Bir bacağı çıkış yapmak için **output** komutunu kullanırız. Daha sonra, çıkış yapılan bu bacağı, **out0 = 1** ile yüksek veya **out0 = 0** ile düşük düzeye çekebiliriz. Bu durumda iki satır yazmamız gerekir: **output 0** ve **out0 = 0**.Oysa aynı işi PBASIC ile tek satırda halledebiliriz: **high 0** yazarsak P0 bacağını aynı anda hem çıkış hem de yüksek olarak belirlemiş oluruz. Benzer şekilde, **low 0** yazdığımız zaman, P0 yine çıkış olacaktır, fakat düzeyi düşük olarak belirlenecektir. Yukarıda yazdığımız ilk programı bu ifadelere göre yeniden düzenlersek, aşağıdaki satırları görürüz:

BASIC Stamp ile led kontrolü

```
Reblink :
   Low 0        'p0 düşük çıkış
   Pause 1000     'bekleme
   High 0         'p0 yüksek çıkış
   Pause 1000
Goto reblink
```

Bu satırların daha önce yazılanlardan farkı yoktur. Her iki program da aynı sonucu verir. Önemli olan, hangisinin daha işlevsel olduğudur.

Bu bölümdeki uygulama, tamamı Infogate firmasının "Yeni Başlayanlar için Mikrokontrolörler" ve "Mikrokontrolörlerle Endüstriyel Kontrol" kitaplarından alınmıştır. Basic Stamp ile ilgili benim en çok ilgimi çeken kitlerin başında Parallax firmasının Boe-Bot adlı ürünü geliyor. Bu ürünü Türkiye'de yine Infogate **www.infogate.org** firması getirtiyor. Ürünün içerisinden, Basic Stamp eğitim kartı, Basic Stamp Mikrokontrolör modülü, Seri bağlantı kablosu, Boe-Bot robot malzemeleri ve tüm dokümantasyon ve yazılımları içeren bir CD-ROM çıkıyor.



ŞEKİL 26.2: Boe-Bot robot kiti, ihtiyacınız olan her türlü malzemeyi içeriyor.

ÜÇÜNCÜ KISIM

III

ROBOTİK UYGULAMALAR

YİRMİYEDİNCİ BÖLÜM

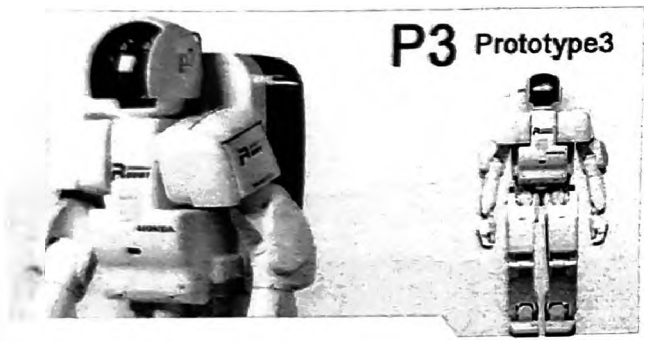
27

ÇİZGİ İZLEYEN ROBOT

Robotlar

Robot yapmak, çok kolay bir iş değildir. Robot yapmaya karar verdiğinizde, önünüze aşmanız gereken bir sürü engel çıkar. Bunlardan bir tanesi, robotun mekanik aksamıdır. Çok üst düzey elektronik bilgisine sahip olsanız bile, ilerleyebilecek bir robotun mekanik aksamını oluşturamadığınız zaman bu bilgi hiçbir işinize yaramaz. Öncelikle malzeme sıkıntısı çekersiniz. Model yaparken bu sıkıntının bir çözümü de Lego kullanmaktır. Bir diğer yöntem ise, oyuncak parçalamaktır. Buna bir oyuncağı akıllı hale getirmek de diyebiliriz.

Hayalimizdeki robotlar her ne kadar korkutucu, yok edici ve çok akıllı makineler olsalar da, günümüz teknolojisiyle geliştirilen robotlar gerçekte gayet kırılgandır ve hareket kabiliyetleri bir insana göre oldukça düşüktür. Bir robotun en önemli problemi hareket etmesidir. Bu işin en kolay yolu tekerlek kullanmaktır. Fakat en basitinden tekerlekli bir robot merdiven çıkamayacaktır. Birçok robot bilimci, yaşantımıza ayak uydurabilecek en kusursuz robot tasarımının insan şeklinde olması gerektiği kanısına varmıştır. Yani kendi dünyamızı insanlar için geliştirdiğimizden, geliştireceğimiz robotların iki ayaklı olması ve insana benzemesi gerekmektedir. İki ayaklı robotların en büyük problemi dengedir. Bir insan yolda yürürken "acaba düşüyor muyum, yoksa düşmüyor muyum" demez. Fakat iki ayaklı bir robotun bunu demesi ve eğer düşüyorsa dengesini ayarlaması gerekmektedir. Bu dengeleme sistemine dinamik balans adı verilir.



ŞEKİL 27.1: Honda'nın P3 robotu.

Örneğin Honda'nın ürettiği HONDA P3 robotu, bu sistemi kullanmakta ve adım atarken bir bacağındaki ağırlığı diğerine geçirerek dengeyi sağlamaktadır. Honda'nın dışında Sony firması da robotik çalışmalar yapmaktadır. Sony kedi ve köpek şeklindeki AIBO robotundan sonra, insan şeklindeki QRIO robotunu 2004 yılında piyasaya sürmüştür. Bu robotun en can alıcı özelliği dengesini kaybedip devrildiği zaman kendi kendine ayağa kalkabilmesidir.

Bir başka yöntem de bacak sayısını artırmaktır. Altı bacaklı bir robotun, üç bacağı havadayken dengesi bozulmayacaktır. Bu dengeleme sistemine ise statik balans denir. Tabii ki altı bacaklı bir robotun herhangi bir A noktasından B noktasına giderken ayaklarını karıştırmaması gerekmektedir. Bu tip robotlardaki diğer bir problem de enerjidir. Çünkü altı bacaklı robot en az altı tane motor demektir. Diğer bir problem de zekadır. Bir robotu her türlü olasılığa karşı programlayamazsınız. Robot çevresini algılamalı ve öğrenebilmelidir. Robot bilimciler öğrenebilen mikroçipler üzerinde hala çalışmaktadırlar. Bu oldukça zor bir iştir. Örneğin MIT

Üniversitesi, bir oda içerisinde hareket edebilen ve dağınık bir odada yön bulabilen bir robot üzerinde dokuz yıl çalıştıktan sonra dört denemeden üçünde başarısız sonuç almıştır. Bir de oda içerisindeki cisimlerin hareket ettiğini düşünürsek iş daha da karmaşık bir hale gelecektir. En büyük problem ise arıza durumudur. Bir insan çok fazla içki içerse en fazla birkaç beyin hücresi ölür ve ertesi sabah baş ağrısı ile ayağa kalkabilir. Fakat bir robotun işlemcisindeki transistörlerden birinin yanması durumunda robotunuzun söyleyeceği tek kelime "Elveda dostum..." olacaktır.



ŞEKİL 27.2: Sony'nin QRIO robotu.

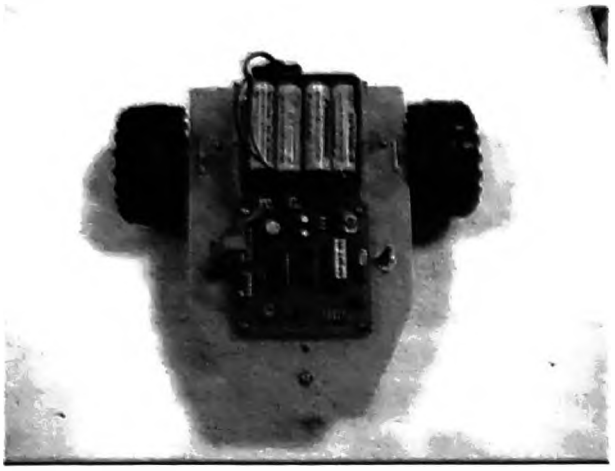
Çizgi İzleyen Robot

UYARI

Çizgi izleyen robot projesine başlamadan önce, Kısım 1 ve Kısım 2'deki tüm uygulamaları anlamış, hatta uygulamış olmanız gerekmektedir. İlk bu uygulama ile işe başlarsanız, bir problem ile karşılaştığınızda, sebebini bulmak ve projeyi sonuçlandırmakta zorluk çekebilirsiniz.

Tekerlekli bir robot yaptığınızda ise karşınıza ilk çıkacak problem, bu robota nasıl manevra yaptıracağınızdır. İki tekerlekli robotlar, manevra kabiliyeti en yüksek robotlardır. Kendi etraflarında 360 derece dönebilirler. Yürüyen ve ayakları olan bir robot yapmak ise daha zor bir iştir. Uzaktan kumandalı oyuncak arabalar, gerçek arabalar gibi manevrayı ön tekerlekleri sayesinde yaparlar. Ön tekerleklerin sağa veya sola dönmesi servo motorlar ile gerçekleştirilir. Servo motorlar robotik uygulamalarda da çok sık tercih edilir. Servo motorlar, içerisinde yine bildiğimiz DC motor barındırırlar. Tek farkı bünyesinde mil konumunu sabitleyebilen bir devre bulunmasıdır. Elektronik devre bu konumu algılamak için PWM (pulse width modulation) darbe genişlik modülasyonu tekniğinden yararlanmaktadır.

Bizim yapacağımız robot, Şekil 27.3'teki gibi 2 tekerlekli olacaktır. Motor olarak DC motor kullanılacaktır. Robotumuz beyaz zemin üzerine çizilmiş siyah bir çizgiyi takip edebilecektir. Bunun için önce bir çizgi algılama sensörü yapmamız gerekecektir.



ŞEKİL 27.3: Hareket için tekerlek kullanan bir robot.

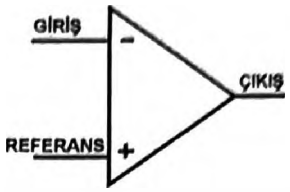
Çizgi Nasıl Algılanır?

Çizgi algılama sensörü, temel olarak şöyle çalışır: Kızıl ötesi bir led, sürekli zemine ışık tutar. Bu ledin hemen arkasında bir foto transistör vardır. Işık yerden foto transistöre yansır. Eğer ışık beyaz zeminden yansımış ise, foto transistör üzerine daha kuvvetli bir ışık düşer. Eğer siyah zeminden yansımış ise, siyah renk ışığı emdiği için daha az bir ışık düşer. Foto transistör üzerine düşen ışık sayesinde transistörün değişen iç direncini kullanarak ışığa bağımlı bir analog sinyal elde edilir. Sonuçta foto transistör üzerindeki ışık kaynağı azaldığında, iç direnç artar. Direnç arttığında ise analog sinyalin değeri düşer. Foto transistörlerin normal transistörlerden tek farkı beyz ucundan yapılan tetiklemenin ışık şiddeti kullanılarak yapılmasıdır. Bu bakımdan genelde foto transistörlerin beyz uçları kullanılmaz. Transistörün üzerinde küçük bir mercek bulunur. Tetikleme işlemi bu merceğe ışık gönderilerek yapılır. Uygulanan ışığın şiddetine göre emitör-kollektör arasındaki iç direnç değişir ve yükün akımı kontrol edilebilir. Işığın şiddeti ile emitör-kollektör arası iç direnç ters orantılıdır. Yani ışık şiddeti arttıkça emitör-kollektör arası iç direnç değeri düşecektir. Foto transistör üzerindeki 0 – 5 Volt arasındaki bu analog değer, bir ADC ile okutulup 0 – 255 arasında bir sayısal değere dönüştürülebilir. ADC ile ilgili detaylı bilgiyi Bölüm 5'te bulabilirsiniz.

ADC ile okutulabilen bu değer 200'ün altına düştüğünde sensör siyah çizgi üzerinde diyebiliriz. Fakat her bir foto transistörden bu veriyi okumak, sonra sayısal sinyale çevirip PIC programı içerisinde karşılaştırma yapmak için çok fazla sayıda ADC ve port gerekir. Bu durumda bize giriş yapılan analog sinyal belli bir voltajın altında ise "0" değerini, belli bir değerin üstünde ise "1" değerini verecek bir devre elemanı gereklidir. PIC ile bu "1" ve "0"ları algılayıp, hangi sensörlerin siyah çizgi üzerine olduğunu anlamak kolay olacaktır. Bu işleri yapacak devre elemanının adı Comparator'dür (Kıyaslayıcı, Karşılaştırıcı).

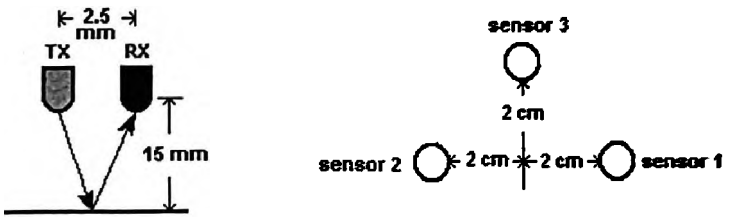
Comparator'ler (Kıyaslayıcı, Karşılaştırıcı)

Esasında kullandığımız devre elemanı temelde bir Opamp. Opamp kelime anlamı işlemsel kuvvetlendiricidir (Operational amplifier).Kısaca transistörlerden oluşan bir yükselteçtir. Opamp'lar yükselteç, karşılaştırıcı, osilatör, filtre vb. işlerlerde çok sık kullanılır.



ŞEKİL 27.4: Opamp'ın comparator olarak kullanımı.

Biz LM serisi bir Opamp olan LM324'ü kullanıyoruz. Eğer çıkışın girişe hiçbir bağlantısı olmazsa, giriş değişince opamp çıkışı besleme voltajına kadar yükseltir. Kazanç sonsuzdur. Bu şekilde de (comparator) karşılaştırıcı olarak çalışır. Giriş gerilimi, yani burada foto transistörden gelen analog voltaj, referans geriliminin üstüne çıkınca, çıkış değişir. Referans voltaj değerini, 20K'lık ayarlı bir direnç üzerinden uygulama sırasında ayarlayacağız. Sensörleri robotun alt ön kısmına, Sensör 3 önde olacak ve aşağıya bakacak şekilde yerleştirmelisiniz. Sensörler arası mesafeler Şekil 27.5'teki gibi olmalıdır.

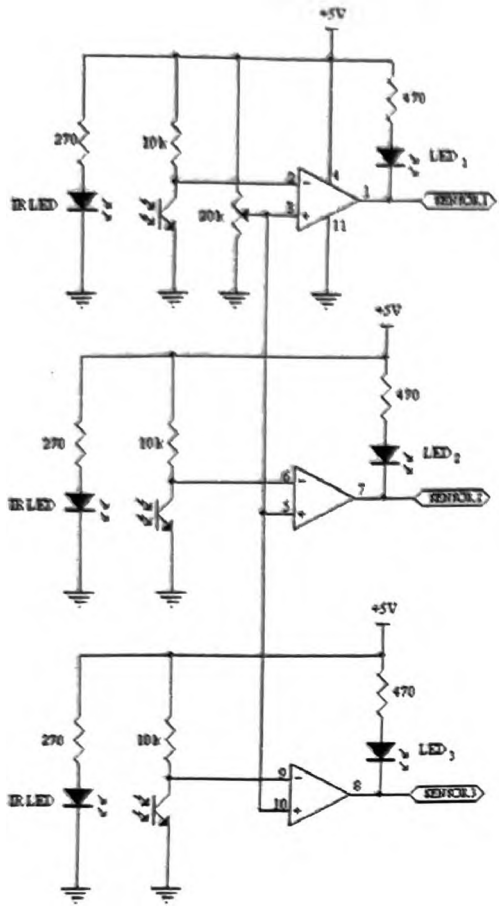


ŞEKİL 27.5: Sensörlerin zemin ve birbirleri ile mesafeleri.

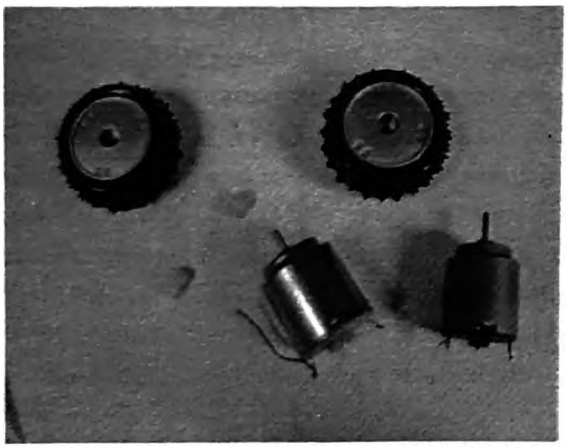
Sensörlerden çizgi olup olmama durumunu nasıl okuyacağımızı yukarıda anlatmıştık. Şimdi sıra, okunan bu verilerin nasıl değerlendirileceğine geldi. Eğer Sensör 3, siyah çizgi üzerinde ve Sensör 1 ve Sensör 2 beyaz alan üzerinde ise, çizgi robotun tam ortasından geçiyor demektir. Bu durumda her iki motor da ileri yönde hareket ederek robotu doğrusal olarak ilerletmelidir. Eğer Sensör 3, siyah çizgiden çıkarsa, bu durumda Sensör 1 veya Sensör 2 siyah çizgiye girmiş demektir. Hangi Sensör siyah çizgiye girdiyse o yöndeki motor durmalıdır ya da yavaşlamalıdır. Böylece robot sensörün bulunduğu yöne doğru dönecektir. Sensör 3 tekrar çizgiye gelene kadar bu dönme devam eder. Sensör 3 ortalandığında tekrar doğrusal harekete geçilir. Bu kontrolü, algoritmayı yazarken daha detaylı inceleyeceğiz. Çizgi sensörünün devre şeması, Şekil 27.6'daki gibidir. Motorları kontrol etmek için ise bir DC motor kontrol elemanına ihtiyacımız var. Paralel port uygulamamızda UDN2993 kullanmıştık. Bu uygulamamızda da, bu entegreyi ya da L293D yongasını kullanabiliriz. Bu H-Bridge sürücülü entegrelerle motorun hız ve yön kontrolünü yapabiliriz. Ya da kendimiz transistörlerle bir H-Bridge sürücüsü yapabiliriz.

Robotun Mekaniği

Robotun mekanik bölümünü çok özenli ve önemseyerek yapmalısınız. "Devreyi oluşturalım da, mekaniğine sonra bakarız" dediğiniz anda, robotu çok zor bitirebilirsiniz. Bir cismi motor gücü ile hareket ettirmek, gözüktüğü kadar kolay değildir. Ben burada basit olsun diye, motorların uçlarına doğrudan tekerlekleri yapıştırdım ve vidaladım. Yani herhangi bir aktarma organı yok, bu da yanında bazı sorunlar getirecek.



ŞEKİL 27.6: Çizgi sensörünün devre şeması.



ŞEKİL 27.7: Hareket mekanizması olarak iki adet oyuncak motor ve tekerlek kullanabilirsiniz.

Size tavsiyem, bu projeyi gerçekleştireceksenız, gidip bir oyuncakçıdan, ufak bir çift motorlu tank satın alın ve devreyi onun üzerinde kurun. Böylece, mekanik problemin büyük bir kısmını atlatmış olacaksınız.



ŞEKİL 27.8: Tekerlekleri motorlara japon yapıştırıcısı veya silikon ile tutturabilirsiniz.



ŞEKİL 27.9: Devreyi ve motorları oturtmak için plastik bir levha kullanabilirsiniz.

Robota hareket verirken karşılaşılacak problemlerden bir tanesi, ilk hareket problemidir. Duran bir cisme ilk hareketi verebilmek, giden bir cismi hızlandırmaktan daha zordur. Bunun için daha çok enerji gerekir. Normalde, motor küçük bir çarkı döndürmeli, aynı mile bağlı büyük bir çark ise tekerleği, ufak bir çark aracılığı ile döndürmelidir. Böylece motora binen yük azalacaktır. Yani hız kaybı olsa bile güç kazancı olacaktır. Ben, oyuncak kullanmadığım için, motoru doğrudan tekerleğe bağladım. Bu durumda motora binen yük arttı. Bu da, robota ilk hareketi vermeyi zorlaştırdı. Bu nedenle, dikkat ederseniz birçok oyuncak, 4 ya da 8 kalem pil ile çalışır. Halbuki, bir tane 9 voltluk pil, hem daha az yer kaplayacak hem daha fazla güç sağlayacaktır diye düşünebilirsiniz. Voltajın artması, motorun daha hızlı dönmesi anlamına gelebilir, ama daha güçlü olması anlamına gelmez. Aynı normal arabalardaki çekiş gücü gibi düşünebilirsiniz. Burada motorun çekişinin daha güçlü olmasını istiyorsanız, yüksek ampere ihtiyacınız var demektir. Bu nedenle bu robotta 8 tane kalem pil kullandım.

Eğer robotun hareketi için gereken çekiş gücünü çarklar vasıtası ile, yani fiziksel olarak azaltabiliyorsanız, o zaman daha düşük amperli piller de kullanabilirsiniz. Eğer buradaki gibi herhangi bir aktarma organı kullanmayacaksanız, mutlaka yüksek amperli kalem pil kullanmalısınız. Benim burada kullandığım kalem piller 1900 mPA yani 1,9 amper. Bu da robota ilk hareketi sağlayacak kadar gücü karşılıyor.

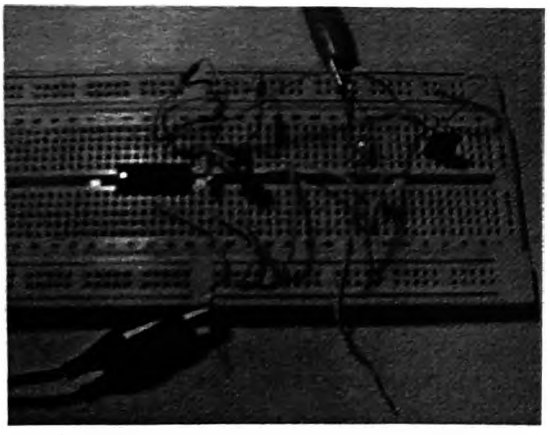


ŞEKİL 27.10: Motorlara gereken güç için yüksek amperli kalem pil kullanabilirsiniz.

Robotumuz, birkaç tane farklı devrenin birleşmesinden oluşuyor. Bunlar; voltaj regülatör devresi, sensörler, motor kontrol devresi, ve PIC çalıştırma devresi. Bu devreleri delikli plaka üzerinde kurmadan önce, mutlaka board üzerinde kurun ve deneyin. Her birinin çalıştığından emin olduktan sonra, devreleri birleştirin. Eğer delikli plaka kullanacaksanız, devrenizde benzer bir kablo karmaşası yaşayacaksınız demektir. Bunun önüne geçmek için, kabloları devrenin altından da lehimleyebilirsiniz. Fakat, bu şekilde lehim atmak, daha zor olacaktır. Daha profesyonel olmasını istiyorsanız, daha önce de anlattığımız baskı devre tekniğini kullanabilirsiniz.

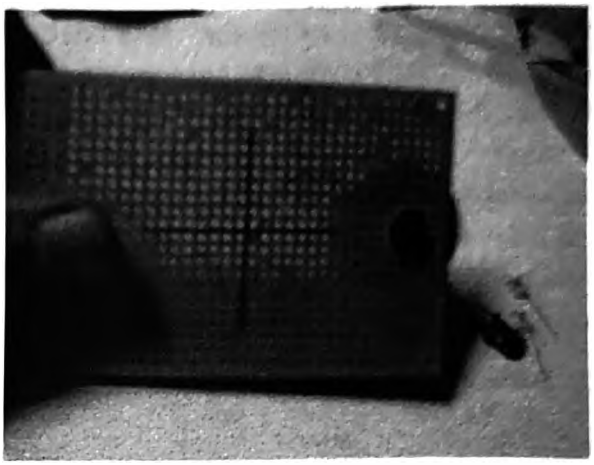
Çizgi Sensörü Devresi

Çizgi sensörünün nasıl çalıştığından bahsetmiştik. Sensörün yaptığı temel işlem, eğer çizgi varsa sensörün ucunu lojik "0" yapmak. Çizgi olma durumunda, sensöre bağlı bir led de, sensördeki sinyali izleyebilmemiz için topraklanıyor. Sensörler aynı zamanda PIC'in A portunun, 0, 1 ve 2'nci uçlarına bağlı olduğundan, hangi sensör çizgide ise, PIC üzerinde portun o pinine giriş yapılmış olacaktır. Biz de, PIC programı içerisinde, port girişlerini test ederek motorlara gerekli hareketi vereceğiz.



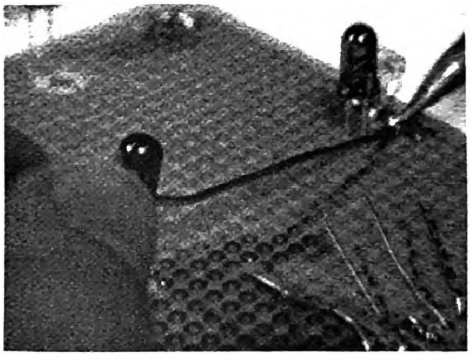
ŞEKİL 27.11: Çizgi sensörünün board üzerinde kurulmuş hali.

Devrenin board üzerinde kurulmuş halini Şekil 27.11'de görüyorsunuz. Sensörlerin yerden yüksekliği 1,5 milimetreyi geçmemeli. Ben sensörleri doğrudan delikli plakanın üzerine lehimledim.

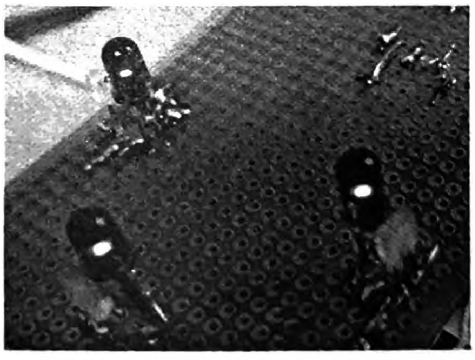


ŞEKİL 27.12: Sensörlerin mesafelerini mutlaka ölçerek ayarlayın.

Ayrıca daha hassas dönüşler için farklı çizgi sensörleri, de tasarlayabilirsiniz. Dikkat etmeniz gereken diğer bir nokta ise, referans voltajını iyi ayarlamanız. Bunu Şekil 27.15'teki gibi devredeki 20K'lık ayarlı direnç ile gerçekleştiriyoruz. Çizgi sensörlerinin iyi çalıştığını, üzerlerine taktığımız normal ledler aracılığı ile de izleyebilirsiniz. Şekil 27.6'da çizgi sensörünün devre şemasını görebilirsiniz. Ayrıca, Şekil 27.14'teki gibi kızıl ötesi ledlerin yansıttığı ışığı gözle göremezsiniz. Ancak dijital bir fotoğraf makinesi ile ledin yanıp yanmadığını izleyebilirsiniz.



ŞEKİL 27.13: Lehim atarken sensörlerin yerden yüksekliğinin aynı olmasına dikkat edin.



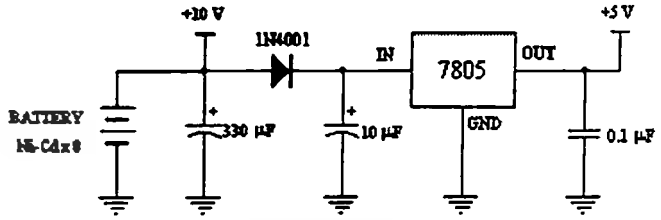
ŞEKİL 27.14: Sensörlerin delikli plakaya monte edilmiş hali.



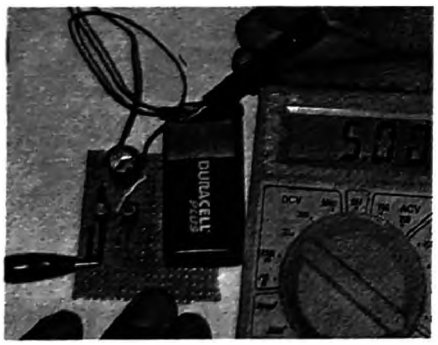
ŞEKİL 27.15: Referans voltajını ayarlayabilmek için, 20k'lık ayarlı direnç kullanabilirisiniz.

Voltaj Regülatörü Devresi

Devrede, birden fazla voltaja ihtiyacımız var. Bunlardan bir tanesi, düşük amperli lojik voltaj, yani +5 Volt. Diğeri ise motorları döndürmek için gereken yaklaşık 9 – 10 volt arası yüksek amperli bir voltaj. Bunun için bir voltaj regülatör devresine ihtiyacımız var. Şekil 27.16'da 7805 voltaj regülatörü ile yapılmış bir güç sağlayıcı devresi görüyorsunuz. Bu devreyi öncelikle, board üzerinde kurmanızı ve gerekli yerlerden gerekli voltajları alıp alamadığınızı bir avometre ile ölçmenizi öneriyorum.



ŞEKİL 27.16: Voltaj regülatörü devre şeması.



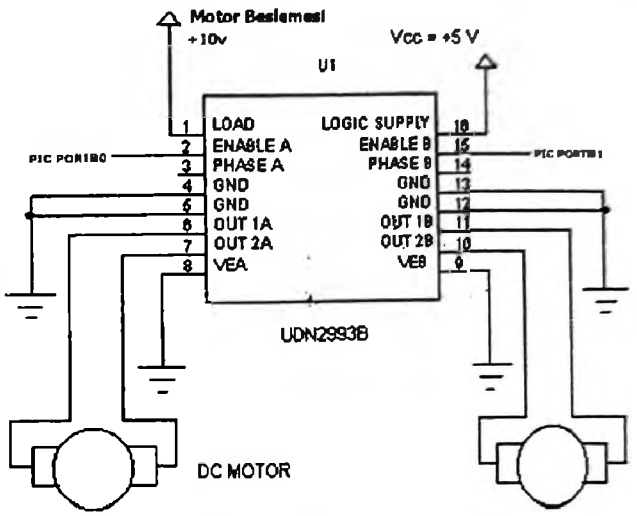
ŞEKİL 27.17: Voltaj regülatörü devresinin board üzerinde kurulmuş hali.

Motor Kontrol Devresi

Motor kontrolü için Bölüm 6'da kullandığımız UDN2993 entegresini kullanıyoruz. Eğer bu entegreyi bulamıyorsanız L293B de kullanabilirsiniz. Fakat bunun için PIC kodunda ve devrede bazı

değişiklikler yapmanız gerekecektir. Bu değişiklikleri Bölüm 6'daki uygulamaya bakıp devreyi ve kodu yeniden tasarlayabilirsiniz.

UDN2993B entegresi ile motorun yönü ve hızı belirlenebilir. Bu entegre ile iki tane motor kontrol etmek mümkündür. 2 - 8 numaralı konnektörler birinci motor için, 9 - 15 arası konnektörler ise ikinci motor için tasarlanmıştır. PHASE girişi motorun yönünü belirlemeye yarar. Bu girişe uygulayacağımız lojik voltaj ile motorun dönme yönünü değiştirebiliriz. Fakat biz motorun yönünü değiştirmeyeceğiz. Burada dikkat etmeniz gereken nokta, motorların birbirlerine göre zıt bağlanmış olmalarıdır. Ancak bu şekilde ikisi de robotu aynı yönde ilerletebilir.

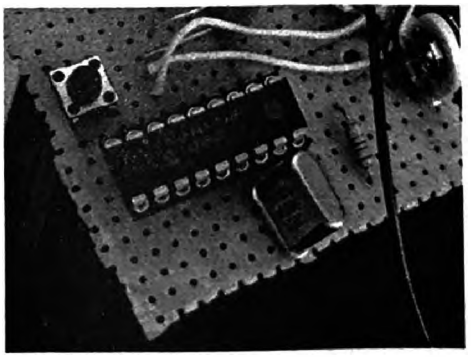


ŞEKİL 27.18: UDN2993 ile motor kontrol devre şeması.

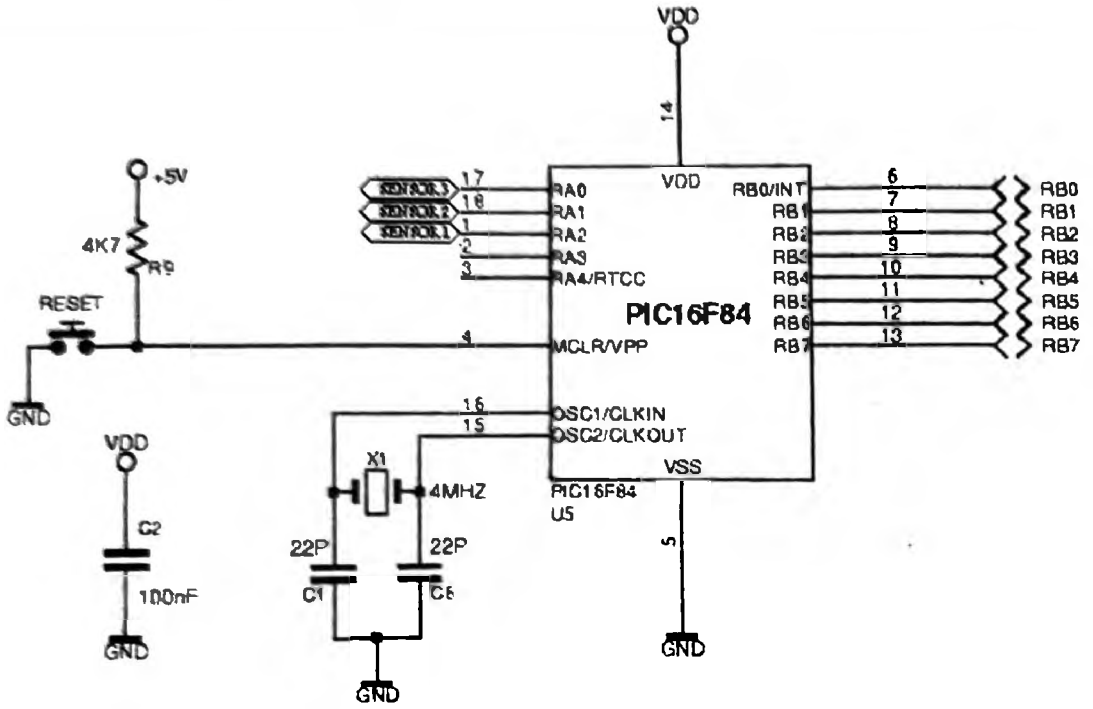
Entegrenin ENABLE girişine, lojik "1" uyguladığımız sürece motor dönecektir. Eğer bu girişe, sinyali darbeler halinde uygulayarak motorun hızını da kontrol edebiliriz. Daha hassas çizgi takibi için, dönmeleri, diğer motorun hızını düşürerek de sağlayabilirsiniz. Devrenin şemasını Şekil 27.18'de görebilirsiniz.

PIC Çalıştırma Devresi

Bu bölüme kadar devrelerimizi hep PIC deneme kartı üzerinde denedik. Artık deneme kartından bağımsız olduğumuza göre, PIC'i robot üzerinde çalıştırabilecek bir devreye ihtiyacımız var. Bu devrenin şemasını Şekil 27.20'de görebilirsiniz.



ŞEKİL 27.19: PIC'i deneme kartından bağımsız hale getiriyoruz.



ŞEKİL 27.20: PIC çalıştırma devresi.

Devre ve PIC Programı Nasıl Çalışıyor

Önce, her zamanki gibi portları giriş ve çıkış için programlıyoruz. PORTA'yı giriş için, PORTB'yi ise çıkış için programlıyoruz. PORTA'ya sensörler ile giriş yapıyoruz. PORTB'nin 2 pinini 2993'ün ENABLE pinlerine bağlıyoruz. Bu iki pinden hangisi toprak ise, o motor dönecektir. Sadece sağdaki motoru döndürmek için PORTB'ye "00000010" yani 0x02 değerini gönderiyoruz. Sadece soldaki motoru döndürmek için ise, "00000001" yani 0x01 değerini gönderiyoruz. Her iki motoru da döndürmek için, "00000000" yani 0x00 değerini gönderiyoruz. Sonuçta, son iki pinden "0"a denk gelen taraftaki motor çalışıyor.

Eğer öndeki sensörde (sensör 3) çizgi var ise, PORTA'nın 0'ncı pini toprak oluyor. Bu durumda, program PORTB'ye 0x00 değerini programdaki **duz** alt programı ile gönderiyor. Bu durumda pinler toprak görevi görüyor ve 2993'ün ENABLE A ve ENABLE B girişleri topraklanıyor. Dolayısı ile her iki motor da çalışıyor.

Eğer sağdaki sensörde (sensör 1) çizgi var ise PORTA'nın 1'nci pini toprak oluyor. Bu durumda, program PORTB'ye 0x02 değerini programdaki **sag** alt programı ile gönderiyor. Böylece PORTB'nin 0'ncı pini toprak görevi görüyor. 1'nci pini ise topraklanmadığı için 2993'ün sadece ENABLE A girişi topraklanıyor. Dolayısı ile sadece ENABLE B girişine bağlı olan motor çalışıyor.

Eğer soldaki sensörde (sensör 2) çizgi var ise PORTA'nın 2'nci pini toprak oluyor. Bu durumda, program PORTB'ye 0x01 değerini programdaki **sol** alt programı ile gönderiyor. Böylece PORTB'nin 1'nci pini toprak görevi görüyor. 0'ncı pini ise topraklanmadığı için 2993'ün sadece ENABLE B girişi topraklanıyor. Dolayısı ile bu sefer de sadece ENABLE A girişine bağlı olan motor çalışıyor. Kısacası, hangi sensörde çizgi var ise onun tersindeki motoru çalıştırıyoruz.

Çizgi izleyen robot PIC Assembly kodu

```
    LIST P=16F84      ;PIC modelini tanıtıyoruz
    include <p16f84.inc> ;16F84 için INC dosyasını ekliyoruz.
basla
    bsf   STATUS,RP0  ; Bank 1 Seç
          movlw   0x00     ; PORTB'nin tüm uçlarını çıkış yap
          movwf TRISB
          movlw 0xFF     ; PORTA'nin tüm uçlarını giriş yap
    movwf TRISA
    bcf   STATUS,RP0  ; Bank 0 Seç
    clrf  PORTB
    movlw 0xFF
    movwf PORTA
;Ana Döngü
loop  btfss PORTA,0     ; Ön sensörü test et (Sensor 3)
    call  duz      ; Eğer çizgide ise duz git. (sağ ve sol motoru çalıştır)
    btfss PORTA,1     ; Sol sensörü test et
    call  sağ      ; Eğer çizgide ise sola don (sağ motoru çalıştır)
    btfss    PORTA,2          ; Sol sensörü test et
    call  sol      ; Eğer çizgide ise sağa dön (sol motoru çalıştır)
    goto  loop
;Sağ Motoru döndürme
sag   movlw 0x02
    movwf PORTB
          return
;Sol Motoru Döndürme
sol   movlw 0x01
    movwf PORTB
    return
;Sag ve Sol Motoru Dondurme
duz   movlw 0x00
    movwf PORTB
    return
       END
```

Bitirmeden önce mekanik için bazı püf noktaları vermek istiyorum. Robotun tekerlekleri yalpalamamalı ve motor tekerleklerin tam merkezine monte edilmeli. Aksi taktirde, motora itme kuvvetinin yanında bir de kaldırma kuvveti yüklemiş olursunuz. Robotun ön tarafında tekerlek yok. Her yönde dönebilen (aynı mouse topu gibi) bir tekerlek ya da bilye bulabiliyorsanız, robotun ön tarafına bunu monte edebilirsiniz. Robot sensörler üzerinde gitmemeli. Bunun için sensörlerin gerisine bir toplu iğneyi ters olarak monte edebilirsiniz.

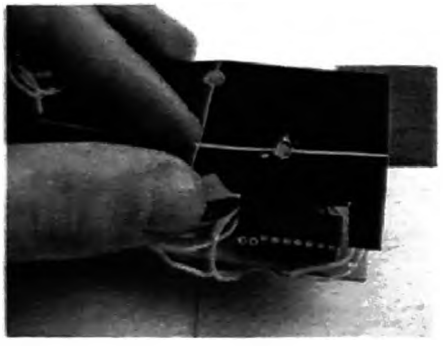
Böylece sürtünmeyi minimum düzeye indirmiş olursunuz. Ayrıca, sensörlerin ortam ışığından etkilenmemesini istiyorsanız, etraflarına siyah bir karton sarabilirsiniz. Ben plaka delikli olduğu için, robotun üstünü siyah bir kağıt ile kaplayarak az da olsa ortam ışığını engelledim. Son olarak, kullandığınız motorlar aynı güçte olmalı. Sadece motorları çalıştırdığınızda robot düz gidebilmeli.

Robotun yapım aşamalarını resimlerden izleyebilirsiniz. Bu iş esasında, iki bölümde anlatılabilecek kadar basit değil. Ben sadece işin mantığını anlatmak için gayet amatör bir

uygulama gerçekleştirdim. Gerçek bir robot için, mekanik hesaplarının, devre tasarımlarının, olasılık hesaplarının ayrı ayrı yapılması gerekmektedir.



ŞEKİL 27.21: Sensörlerin arkasına, yere değmemeleri için ters bir toplu iğne monte edebilirsiniz.



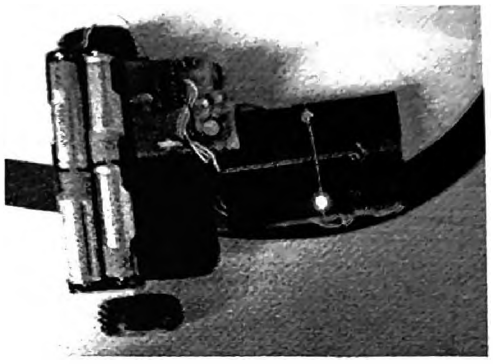
ŞEKİL 27.22: Sensörleri ortam ışığından korumak için siyah bir karton kullanabilirsiniz.



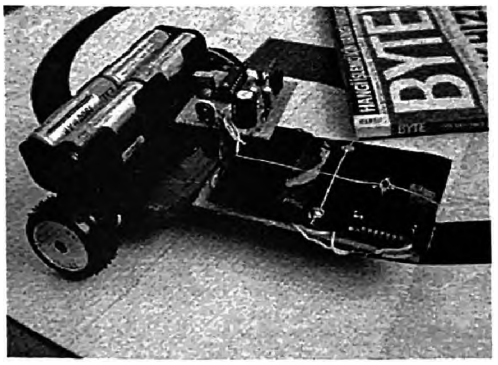
ŞEKİL 27.23: Robotun enerjisini ufak bir anahtar ile kontrol edebilirsiniz.



ŞEKİL 27.24: Robotumuz çizgiyi takip ediyor.



ŞEKİL 27.25: Hangi sensörlerin çizgide olduğunu üzerlerindeki led ile izleyebiliyoruz.



ŞEKİL 27.26: Çizgi izleyen robot.

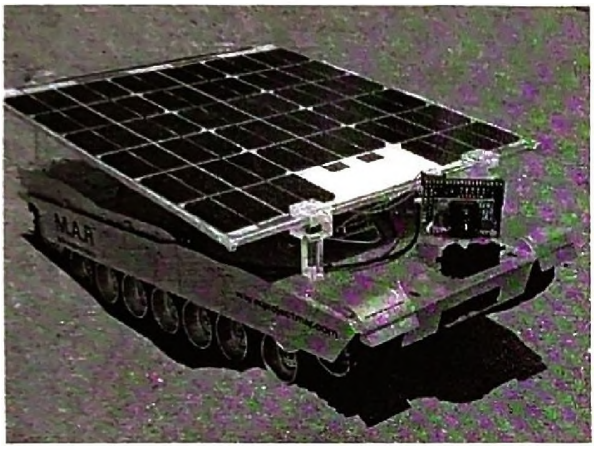
YİRMİSEKİZİNCİ BÖLÜM

28

MOBİL ARAŞTIRMA ROBOTU PROJESİ

Giriş

Bu bölümde, benim İstanbul Üniversitesi, Astronomi ve Uzay Bilimleri bölümünden mezun olurken verdiğim ve hala üzerinde çalıştığım bir uzay araştırma robotu projesinden bahsedeceğiz. Dolayısı ile bu bölüm, bir uygulama bölümü değildir. Robotun nasıl yapıldığından, malzemelerin nasıl bulunduğundan ve özelliklerinden bahsedilecektir. Robotun bütün yazılım kodları ve devre şemaları burada verilmeyecektir. Şundan emin olunuz ki, bu kitaptaki bilgiler ile böyle bir robotu sizin de yapmanız mümkündür.



ŞEKİL 28.1: Mobil Araştırma Robotu.

MAR Projesi

Projenin ana amacı insansız bir araştırma robotu tasarlamak ve uygulamak olarak verilebilir. Güneş sistemi içerisinde bile diğer gezegenlere ulaşmak uzun yıllar alabildiğinden, insan ile yapılan yolculukların oldukça zahmetli, pahalı ve zor olmasının yanı sıra, insan faktöründen kaynaklanabilecek hataların da, bu gibi insansız sistemlere oranla fazla olabileceği göz ardı edilmemelidir. Bunun yerine içerisinde insan olmayan sistemlerin ön araştırmalar için kullanılması söz konusu olabilmektedir. Günümüzde, yapılmış olan projelerden birisi NASA'nın Mars gezegenine gönderdiği Pathfinder robotudur. Bu projede yapılacak olan robot NASA'nın Pathfinder'ının bir modeli olarak düşünülebilir.

Tasarlanacak olan araştırma robotu sistemi üç farklı modül yardımıyla kullanılacaktır. İlk modül örnekleri toplayacak ve araştırmaları bizzat yapacak olan araştırma robotu, ikinci modül ise robotun 1 – 2 km kadar uzaklığında bulunması gereken bir baz istasyon, üçüncü ve son modül ise Dünya'da bulunan ve baz istasyonun gönderdiği resim ve örnekleri alan işlem istasyonudur.

Ana Modüller

Mobil AR

Bu modül, araştırma yapılması istenilen alan içerisinde kendisine girilen koordinatlar üzerinde hareket edebilecek, işlem istasyonuna fotoğraf gönderebilecek paletli bir uzay aracıdır. Bu baz

istasyon elde ettiği fotoğraf ve verileri önce baz istasyona gönderecek ve baz istasyon ise bu verileri kuvvetlendirerek işlem istasyonuna gönderecektir.

Baz İstasyon

Mobil AR'den aldığı verileri direkt olarak işlem istasyonuna gönderecek donanıma sahip olmalıdır. Bu donanım özellikleri aynı zamanda yeryüzünden gelecek olan emir ve işlemlerin de tekrar mobil AR'ye iletilmesini sağlamalıdır. Burada baz istasyon ile mobil AR arasındaki mesafe pek fazla olmamakla beraber, asıl mesafe ve dolayısıyla iletişim problemi baz istasyon ile yeryüzündeki işlem istasyonu arasında kaynaklanmaktadır. Bu durumda Mobil AR üzerinde bulunamayacak kadar büyük olan güç kaynağı baz istasyonda bulunabilecektir. Bu problemde zaten baz istasyonun varolmasının temel nedenlerinin en başında gelmektedir.

Bunlara ek olarak, baz istasyonunda bir kişi tarafından da kullanılabileceği düşünüldüğünden bu istasyon üzerinde de mobil AR ile iletişim arayüzü yüklü olmalıdır. Biz burada her ne kadar insansız bir sistem tasarlamaya çalışsak da yine de AR yakınlarında bulunabilecek ve AR'yi yönlendirmek isteyebilecek bir operatör olabileceği göz önünde bulundurulmalıdır.

İşlem İstasyonu

İşlem istasyonu, baz istasyondan aldığı verileri işleyebilecek ve yeryüzünde bulunan tam donanımlı ve sonsuz ömürlü bir istasyondur. Bu istasyon baz istasyondan gelen resimleri bir görüntü işleme sürecine sokabilecek yazılımlara sahip, bu görüntüler üzerinde istenilen noktaları işaretleyebilecek ve işaretleri, mobil AR'de uygulanması için baz istasyona gönderebilecek donanıma sahip olmalıdır.

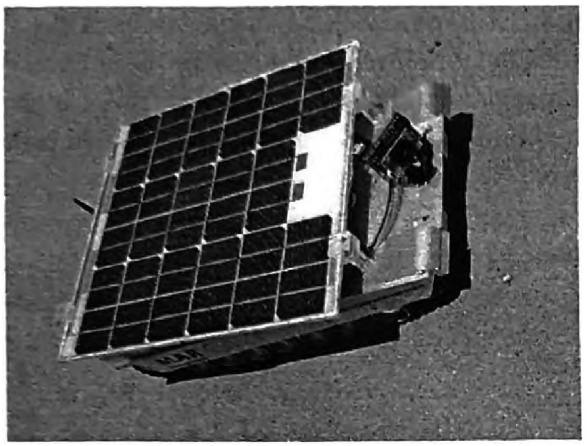
Performans Sorunları

Yapılacak olan mobil AR projesinin en ciddi performans sorunu, aradaki mesafeden ve uzay koşullarından kaynaklanan iletişimdeki yetersiz hız problemidir. Her ne kadar bu problem bir baz istasyon aracılığıyla bir miktar giderilmeye çalışılsa da, hala en kuvvetli performans sorunlarının başında gelmektedir. Ortaya çıkan bu gecikme, aslında bu projenin bütünü değil, fakat bir kısmında yer almaktadır. Bunun nedeni, baz istasyon ile mobil AR arasındaki mesafenin uygun bir değerde olması, fakat baz istasyon ile işlem istasyonu olarak tanımlanan yeryüzünde bulunacak olan birim arasında mesafenin göz ardı edilmemesi gereken bir mesafe olmasıdır. Mesafe problemine ek olarak, bir de uzay koşullarının iletişimde yarattığı zorluk, iletişimdeki yavaşlığı daha da artırmaktadır.

Bir diğer performans sorunu ise, mobil AR'nin karşısına çıkabileceği bazı engelleri atlatamayabileceğidir. Bu problem de direkt olarak performansı etkiler, çünkü böyle bir olay ile karşılaşıldığında iletişim zamanı açısından zaten oldukça yetersiz kalınacak olan projede birde araştırma alanında ilerleme problemi eklenecektir. Burada yapılabilecek en etkin sonuç, mümkün olduğu kadar çok engel problemini mobil AR'ye tanıtmak olacaktır. Böylece tatbikat esnasında ortaya çıkabilecek herhangi bir problemin mobil AR'nin ilerlemesini engellemesi olasılığı minimuma indirilecektir.

İlk akla gelen bir diğer performans sorunu ise, enerji sorunu olduğu açıktır. Daha önce de bahsedildiği üzere, insansız bir ortamda enerji probleminin olacağı görülmektedir. Bu proble-

min aslında projenin ömrünü de belirlediğini belirtmiştik. Bu süreci bir miktar daha geciktirmek için yapılacak ilk şey mobil AR tarafından kullanılan enerjinin en etkin şekilde harcanmasıdır. Burada amaç, minimum enerji kullanımı ile maksimum iş yapmak olarak özetlenebilir. Harcanan enerjiyi yeniden elde etmek amacı ile, MAR üzerine pilleri şarj edecek bir güneş paneli de yerleştirilmiştir.



ŞEKİL 28.2: Pilleri şarj etmek amacı ile güneş panelleri kullanılmıştır.

Performansı etkileyebilecek bir diğer ortam sorunu ise zemin problemidir. Her zaman düz ve sert bir satıh bulunamayabilir. Mobil AR'nin ilerlemesi gereken satıh sıvı, kum ve hatta bataklık benzeri bir zemin olabilir. Bu gibi performans sorunlarını en aza indirmek için mobil AR'nin yürüyen aksamını paletli modellere uygun yapmak gerekecektir. Fakat yine de dikkat edilmesi gereken husus, mobil AR yönlendirilirken gitmesi için emir verilen bölgenin iyi bir analizinin yapılması ve en düşük riske sahip bölgeler üzerinde hareket ettirilmeye çalışılmasıdır. Şu açıkça görülmektedir ki, mobil AR'nin içinden çıkamayacağı bir durum projenin ömrünü belirleyebilir.

Bu projenin en büyük sınırlaması, finansal kısıtlamalardır. Açık olarak belirtilmelidir ki, bu gibi insansız elektronik sistem tasarımları oldukça maliyetli ve uzun çalışma, deneme ve araştırma gerektiren çalışmalardır. Bu zahmetli çalışmanın ilk getirdiği zorluk, zaman ve finansal zorluklardır. Bu projeye başlanırken aslında her iki zorluk da bir sınırlama olarak karşımıza çıkmaktadır.

Mobil AR'nin elektronik, kasa, yürüyen ve hareketli aksam tasarımları da teknik bir kısıtlama teşkil etmektedir. Gerekli dökümleri ve parçaları bulmak, yaptırmak ve oluşturmak için deneyimli elemanlarla çalışmak daha uygun olacaktır ki, bu konu üzerine deneyimi olan, daha önce bu gibi tasarımlar yapmış torna, döküm, metal atölyeleri bulabilmek de bir sınır olarak karşımıza çıkmaktadır.

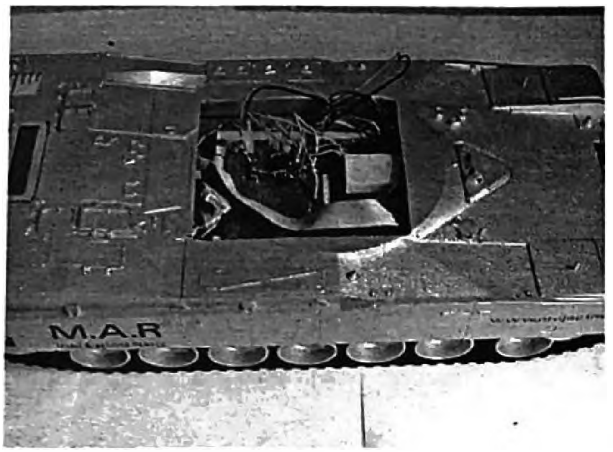
Projenin yapısının getirdiği bir diğer ve önemli olan kısıtlama donanım kısıtlamasıdır. Performans sınırlamalarında da dile getirildiği gibi, oldukça kısıtlı bir şekilde yapılacak olan iletişimi mümkün olabilecek en hızlı konuma getirebilmek için kuvvetli ve hızlı iletişim arabirimleri, pahalı ve etkin donanımlar kullanmak zorunda kalacağımız görülmektedir. Bir

yandan da iletişim esnasında kaybedeceğimiz zamanı donanımsal olarak dengelemek isteyebiliriz. Bu da, aslında finansal sınırlamalar ile paralel bir sınırlamadır.

MAR Mekaniği

Bir uzay araştırma robotunun mekanik tasarımı, görevine göre değişebilir. En basitinden, gideceği gezegenin zemin yapısı bile robotun yürüyen aksamının tasarımını bile değiştirecektir.

Yapılacak robotun mekanik tasarımı ve hesaplamaları, MAR projesinde göz önüne alınmamıştır. Bunun nedeni böyle bir tasarımın yüksek derecede mekanik ve makine mühendisliği bilgisi gerektirmesidir. Mekanik tasarımı minimum düzeye indirgemek için, hazır ve yürüyebilen ve gerekli elektronik donanımı taşıyabilecek bir araca ihtiyaç duyulmaktadır.



ŞEKİL 28.3: Proje için hazır bir oyuncak tank satın alınmış ve boyatılmıştır.

Gereken araç, bir tank gibi paletli tercih edilmelidir. Bunun nedeni, paletli bir aracı istenilen yöne çevirebilmenin daha kolay olmasıdır. Dört tekerlekli bir aracı döndürmek için ise manevra yaptırmak gerekir. Bu da, manevra sırasında koordinat kaybına yol açacaktır.

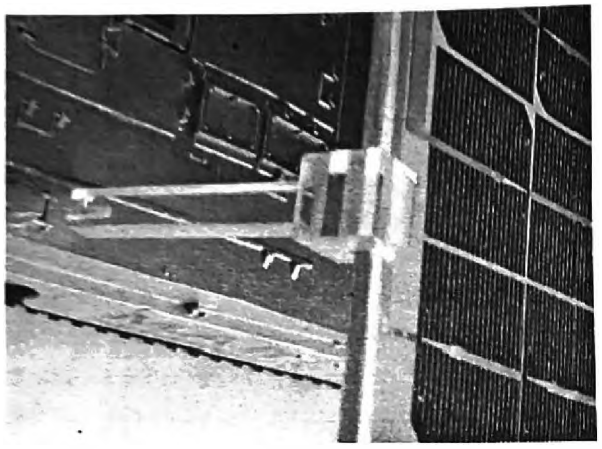
Bu bilgiler ışığında, donanımı üzerine yerleştirmek üzere oyuncak bir tank satın alınmıştır. Paletli ve iki adet DC motora sahip bu tank, üzerine dekupaj cihazı ile bazı bölgelerinde kesim yapılarak içerisine, gerekli donanımın yerleştirilebileceği bir bölüm açılmıştır. Aynı zamanda oyuncak namlu ve tankın üst kısmı kesilerek çıkarılmıştır. Matkap aracılığı ile de donanımların montajını yapabilmek için bazı delikler açılmıştır. Bazı parçalar ise silikon tabancası ile tutturulmuştur.

Güneş panelini tutturmak için Pleksy Glas malzemesinden, 4 adet tutturucu tasarlanmış ve kestirilmiştir. Bu tutturucular sayesine güneş paneli, robota müdahale etmek gerektiğinde kolaylıkla çıkabilmektedir. Tankın dış gövdesi ise tüm delme ve kesme işlemleri bitikten sonra mat metalik griye boyatılmış ve üzerine dijital baskı sistemi ile projenin ve robotun adları yazdırılmıştır.

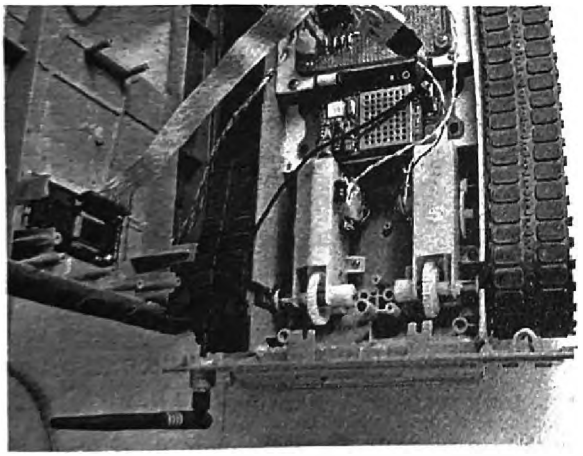
Tüm elektronik donanım ve pillerin yerleştirilmesi ile birlikte robot yaklaşık 3 kg ağırlığa erişmiştir. Motorlar bu ağırlığı kaldırmak için yeterli torku sağlayabilmektedir. Oyuncak tankın motorlar hariç bütün elektronik devreleri sökülmüştür.



ŞEKİL 28.4: Çift paletli bir aracın manevrası daha kolaydır.



ŞEKİL 28.5:Güneş paneli Pleksyglas malzemesi ile robota monte edilmiştir.



ŞEKİL 28.6: Motorlar hariç tüm diğer parçalar sonradan takılmıştır.

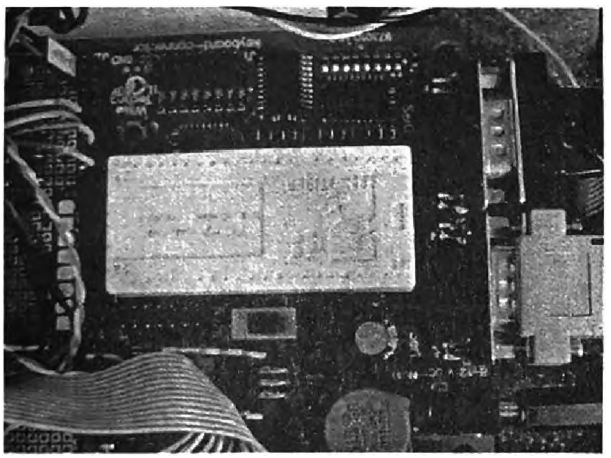
MAR Elektroniği

MAR elektroniği bir kaç ayrı devrenin birleştirilmesinden meydana gelmiştir. Bu devreler şunlardır:

- BASIC-Tiger mikroişlemci devresi
- UKS radyo modem devresi
- L293B motor kontrol devresi
- Cisim algılama devresi
- Pil şarj ve enerji devresi
- Kamera yönü ve servo motor kontrolü
- Kamera devresi

BASIC-Tiger Mikro İşlemci Devresi

İş istasyonundan gönderilen komutları radyo modem aracılığı ile algılayıp yorumlayabilmek için, bir mikroişlemciye ihtiyacımız var. MAR projesi için bu mikroişlemci ilk olarak PIC kullanmak düşünülmüştür. Fakat PIC programlama Assembly aracılığı ile yapıldığından, proje süresini artırmaya sebep olacaktır. Bu nedenle BASIC tabanlı işlemcileri kullanmak tercih edilmiştir. BASIC Tiger bu tip yongalardan biridir. Türkiye'de temsilciliği Lima Endüstriyel Bilgisayar Şirketi tarafından yapılmaktadır. Bu mikroişlemciden Bölüm 25'te bahsetmiştik.



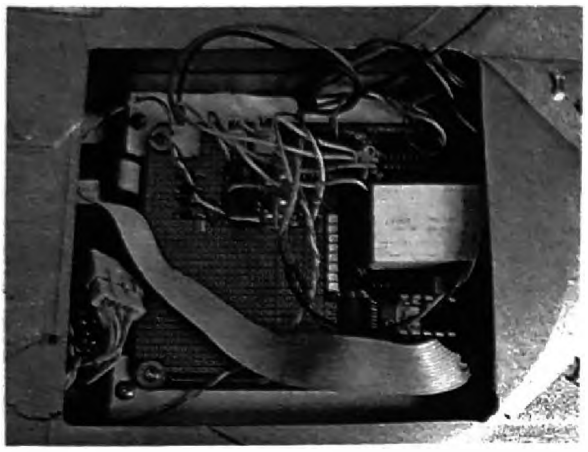
ŞEKİL 28.7: MAR Basic Tiger ile kontrol edilmektedir.

Yukarıda saydığımız bütün bu elektronik devrelerin yerleştirileceği bir ana kart olmalıdır. Anakart olarak, BASIC-Tiger seti ile birlikte gelen board kullanılmaktadır. Bu set mikroişlemcinin çalışabileceği ortamı yaratan bir kart ile birlikte gelmektedir. Bu kart üzerinde, ek devrelerin yerleştirilebileceği boş bir board alanı bulunmaktadır. Boş alan üzerine gerekli devreler lehimlenerek, tüm devrelerin üzerinde çalıştığı bir anakart haline getirilmiştir.

UKS Radyo Modem Devresi

Eğer robot, iş istasyonuna seri bir kablo ile bağlı olabilseydi, iletişim ile ilgili hiç bir problemimiz kalmayacaktı. Bu nedenle uygulama ilk olarak sanki robot kablo ile bağlıymış gibi gerçekleştirilmiştir. Daha sonra, hem MAR'ın hem de iş istasyonunun seri portlarına birer radyo modem takılarak bu iletişim kablosuz hale getirilmiştir. Radyo modem olarak, UKS firmasının

ürettiği WAVELINK-24 serisi radyo modemler kullanılmıştır. Bu modemler ile ilgili detaylı bilgiyi `http://www.kabloyok.com` adresinden elde edebilirsiniz.



ŞEKİL 28.8: Diğer devreler BASIC Tiger ile birlikte gelen board üzerine kurulmuştur.



ŞEKİL 28.9: WAWELINK-24 radyo modem.

WAVELINK-24 Serisi Radyo Modemler

WAVELINK-24 serisi radyo modemler 19,200 bps hızında, 2.4 GHz ISM bandında çalışan kablosuz RS232/485 iletişimi sağlıyorlar. Cihazların entegre antenli veya uzun mesafelerde kullanılmak üzere harici anten bağlantısına izin veren 2 adet modeli bulunmakta. Entegre antenli model ile şehir ortamında 500m, açık alanda 2km (görüş hattı olması şartıyla) mesafeye ulaşabiliyor. Harici anten takılabilen versiyonu ile yüksek kazançlı yönlü antenler kullanarak 20 km ve daha uzun mesafelere ulaşmak mümkün oluyor.

Radyo modemler ağırlıklı olarak; Yaygın SCADA (su dağıtımı, arıtma, doğal gaz hatları, trafik sinyalizasyon otomasyonu) projelerinde, su pompa/depo istasyonları otomasyonunda, uzaktan sayaç okuma sistemlerinde, güvenlik uygulamalarında (uzaktan kamera kontrolü), POS cihazları otomasyonlarında, tartı istasyonu otomasyonunda ve benzeri her türlü RS232/RS485 veya sayısal/analog giriş/çıkış uygulamalarında tercih ediliyor.

Ayrıca bu modemler frekans atlamalı Spread Spectrum sinyallemesi kullandığı için dışarıdan gelebilecek gürültüye karşı oldukça dayanıklı. Şeffaf çalışma prensibi sayesinde de her türlü sisteme kolayca entegre edilebiliyor ve sanal bir kablolu ağ şebekesinden farksız olarak çalışıyor. Benzer diğer radyo modemlere göre çok daha ekonomik oluşu ve yüksek alıcı hassasiyeti ile birim güç ile çok daha fazla mesafeye ulaşılmasını mümkün kılıyor.

Bu cihazlar noktadan noktaya veri iletişiminde kullanılabileceği gibi, ağ kullanımlarına da uygun. Adresleme özellikleri sayesinde pek çok noktadan verinin toplanılıp bir noktada işlenmesi de mümkün kılınıyor. Ayrıca, opsiyonel IO modülleri ile modemlerin sayısal veya analog giriş çıkış taşıması ve hatta işlemesi mümkün olup, pahalı PLC yatırımlarını da gereksiz kılıyor.

Radyo Modem ile İletişim Uygulaması

Modemin nasıl kullanıldığını anlatabilmek amacı ile, BASIC-Tiger seti ile PC arasında karakter göndermeye yarayan basit bir uygulama gerçekleştirmede fayda görmekteyim. Bu nedenle, modemin bir tanesini, BASIC-Tiger test kartının, seri portlarından birine, diğer modemi ise bilgisayarın seri portuna takıyoruz. Her iki programda da dikkat etmemiz gereken noktalardan bir tanesi, portların baud hızını 19,200 olarak ayarlamak. Bunun nedeni, modemin bu hızı desteklemesi. PC tarafındaki program ise, sadece klavyeden aldığı ASCII karakterleri, seri porta göndermeye yarıyor. Bu veriler radyo sinyali ile diğer modeme ulaşıyor. BASIC-Tiger'a ise sürekli sinyal beklemesini ve herhangi bir veri geldiği zaman onu LCD ekrana yazdırmasını sağlayan bir program yüklüyoruz.



ŞEKİL 28.11: BASIC Tiger-PC radyo modem bağlantısı.

Radyo modem ile BASIC Tiger – PC bağlantı uygulaması BASIC Tiger kodu

```
#INCLUDE DEFINE_A.INC      'genel tanımlar
#INCLUDE UFUNC3.INC     'kullanıcı fonksiyonları
TASK MAIN
  BYTE EVER        'sonsuz döngü için
  WORD FREE        'buffer daki boş alan için
  WORD FI
  LONG A
  INSTALL_DEVICE #SER, "SER1B_K1.TDD", & 'seri sürücüyü yükle
```

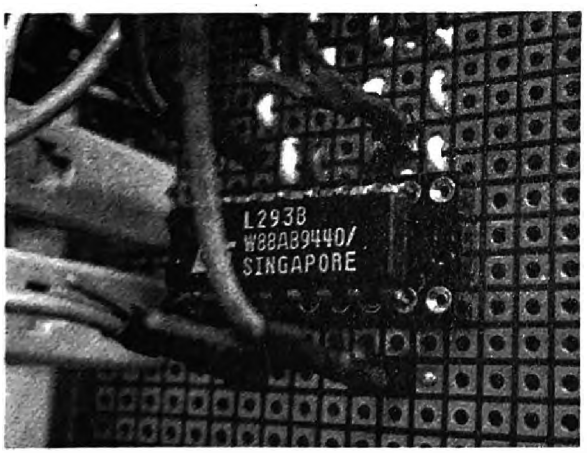
```
    BD_19_200, DP_8N, YES, &     'SER0 portunu ayarla
    BD_19_200, DP_8N, YES     'SER1 portunu ayarla
  INSTALL DEVICE #LCD, "LCD1.TDD"
  FOR EVER = 0 TO 0 STEP 0    'sonsuz döngü
    GET #SER, #0, #UFCI_IBU_FILL, 0, FI    'bufferi oku
    IF FI > 0 THEN         'buffer da birşey var ise
      GET #SER, #0, 1, B                'bir BYTE oku
      PRINT #LCD, CHR$(B);              'okunan BYTE ın karekter karşılığını yaz
    ENDIF
  NEXT
END
```

Radyo modem ile BASIC Tiger – PC bağlantı uygulaması Visual Basic kodu

```
Private Sub text1_KeyPress(KeyAscii As Integer)
   Dim Buffer As Variant
   Buffer = Chr$(KeyAscii)
   MSComm1.Output = Buffer
End Sub
Private Sub Form_Load()
   MSComm1.CommPort = 1
   MSComm1.PortOpen = True
End Sub
```

L293B Motor Kontrol Devresi

Buradaki devre Bölüm 22'de gerçekleştirdiğimizin aynısıdır. L293B entegresi IN1 ve IN2 girişine uygulanan lojik sinyaller ile, DC motora yön kontrolü yapabilmektedir. Gereken bu lojik sinyaller BASIC-Tiger'ın 8'nci portundan alınmaktadır. BASIC-Tiger içerisindeki program, iş istasyonundan gelen komutun parametresini değerlendirerek gerekli lojik sinyali port üzerine yerleştirmektedir.



ŞEKİL 28.12: L293B ile DC motor yön kontrolü sağlanmaktadır.

L293B yongası BASIC-Tiger'ın 8 numaralı portunun 0, 1, 2, 3 numaralı pinlerine bağlıdır. Bağlantı şekli ise yukarıdaki şeklin sol tarafındaki gibidir. Bir L293B ile 2 adet DC motor kontrol

edilebilmektedir. MAR üzerinde 2 adet DC motor bulunduğu için 1 adet L293B devresi uygulama için yeterlidir.

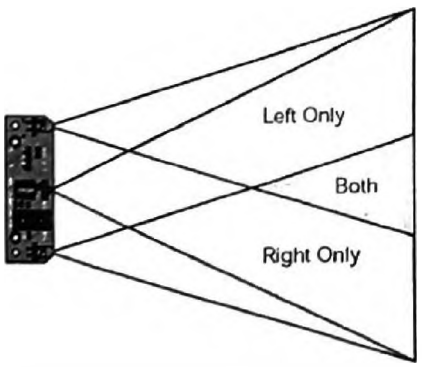
Cisim Algılama Devresi

MAR kendini önüne çıkan engellere karşı korumak zorundadır. En basitinden, önündeki bir kayayı görmemesi, MAR'ın devrilmesine ya da hasar görmesine sebep olabilir. Bu da hiç şüphesiz projenin durması ve her şeyin sona ermesi anlamına gelmektedir.



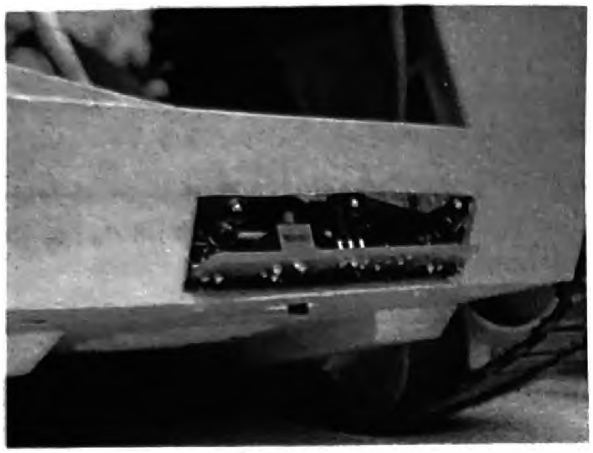
ŞEKİL 28.13: Lynx Motion'un cisim algılama sensörü.

MAR, önündeki cisimleri görmek için kızıl ötesi ışık kullanmaktadır. Lynx Motion tarafından üretilen bu sensör, tam güneşte bile çalışabilmektedir. Sensörün iki tarafındaki mor ve kızıl ötesi birer led vardır. Bu ledden gönderilen ışınlar, eğer bir cisme çarparsa yansıyacak ve aynı devre üzerindeki özel bir foto transistör ile algılanarak cismin yerini belirleyecektir.



ŞEKİL 28.14: Sensör yansıyan ışığa göre karar almaktadır.

Sensör, yansıyan ışığın değerine göre cismin nerede olduğunu yine lojik sinyaller aracılığı ile, mikroişlemciye BASIC-Tiger'ın 6'ncı portu ile bildirmektedir. Bu iletişim için portun sadece 3 pini kullanılmaktadır. Örneğin MAR'ın sağında bir cisim varsa, bu 3 kablodan birincisi yüksek yani lojik "1" olmaktadır. Mikroişlemcinin 6'ncı portu giriş için ayarlı olduğundan bu veriyi algılamakta ve gerekli işlemi yapmaktadır.

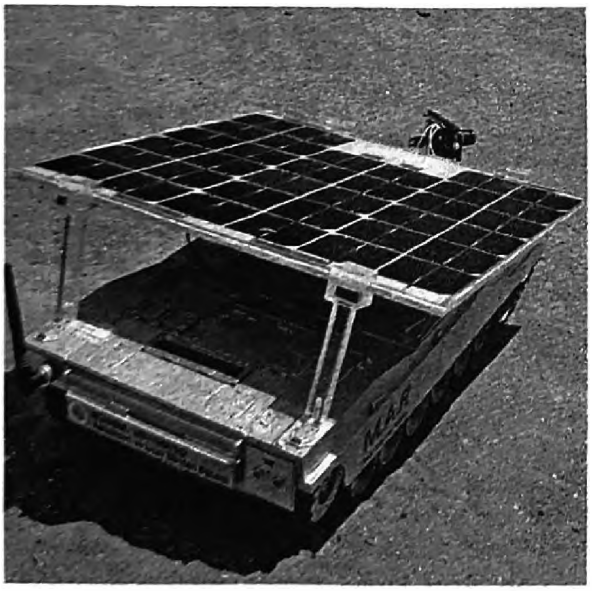


ŞEKİL 28.15: Sensör MAR'ın ön bölümüne yerleştirilmiştir.

Bu sensör `www.robotstore.com` adresinden satın alınmıştır. Aynı site içerisinde robotlar için birçok malzeme bulmak mümkündür.

Pil Şarj ve Enerji Devresi

MAR üzerindeki elektronik devreleri besleyecek iki tane pil bulunmaktadır. Bunlardan bir tanesi 12V 2A enerji sağlamakta ve motorlar hariç tüm elektronik devrelere enerji vermektedir. Diğer pil ise sadece motorlar için 7.2 V 600mA'lik enerji sağlamaktadır. Motorlar çalışırken çok fazla akım çekeceğinden, enerji kaynakları ayrı tutulmuştur. Aksi durumda motorlar çalıştığı sırada diğer elektronik devrelerin çalışmasında problem yaşanabilirdi.



ŞEKİL 28.16: Güneş paneli, MAR durduğu sürece pilleri şarj etmektedir.

Bu piller NiCd olup şarj edilebilme özelliğine sahiptirler. Şarj işlemi dışarıdan takılacak bir adaptör ile olabileceği gibi robotun üzerinde bulunan ve maksimum ışıkta 16V 600mA enerji

sağlayabilen bir güneş paneli ile de olabilmektedir. Güneş paneli robotu sürekli şarj etmekte olsa bile, en sağlıklı şarj, motorlar ve radyo modem çalışmaz iken olacaktır. Bu nedenle MAR'ı iş istasyonundan, şarj moduna alabilecek bir komut bile, komut setine dahil edilmiştir. Ayrıca güneş paneli pile bağlandığı sırada pildeki enerjinin panele doğru akmaması için her iki pilde birer diyot kullanılmıştır. Böylece paneldeki enerji daima tek yönlü olarak pile akacaktır.

Kamera Yönü İçin Servo Motor Kontrolü

Servo motor, MAR'ın görüntü göndermek için kullandığı "CmuCam" kamerasını döndürmek amacıyla kullanılmaktadır. Projedeki servo motor kontrolü Bölüm 24'teki ile aynıdır. Servo motorlar dahili dişli sistemine sahiptirler ve yüksek momentlere ulaşırlar. Çıkış şaftı DC ve step motorlarda olduğu gibi serbestçe dönmez, belli açılarda döner. Servo motorlar bunu elektronik şaft yeri algılayıcısı ve bir kontrol devresiyle yaparlar. Servo motorların 3 çıkışı vardır: Güç, toprak ve kontrol. Çoğunlukla 5 Voltta çalışırlar. Kontrol sinyalleri bir dizi vurgudan oluşur ve şaftın istenilen konumunu belirlerler. Her vurgu bir konum komutunu simgeler. Şaftların çoğu 180 derece dönebilirler.

Servo Motorun İç Yapısı

BASIC Tiger, 1 ile 2 milisaniye arasında dalga genişliği değişen bir sinyali her 20 milisaniyede bir servoya gönderir. 1 milisaniye tam sol, 2 milisaniye de tam sağ pozisyonu ifade eder. Servo içindeki elektronik devre ilk önce gelen darbelerin darbe genişliğini ölçer, daha sonra potansiyometre konumuna bakar ve kendi darbe osilatörünün darbe genişliği gelen darbelerle eşitlenene kadar motoru hareket ettirir.

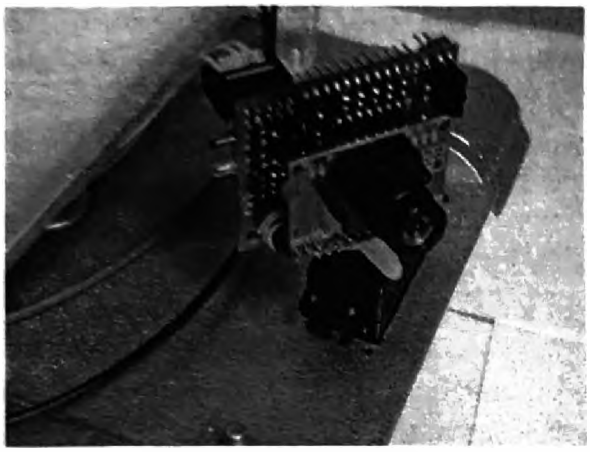
Servo motorun üzerinde CmuCam bulunmaktadır. Kameranın dönmesi için gönderilen komutun parametresine göre, BASIC-Tiger gerekli sinyali 6'ncı port aracılığı ile servo motora göndermekte ve kameranın yönünü belirlemektedir.

Kamera Devresi

CmuCam, ucuz ve fazla voltaj gerektirmeyen bir elektronik gözdür. Hareket halinde olan robotların üzerine takmak için uygun bir devredir. CmuCam verisini seri port üzerinden verdiği için, dış mikroişlemciler ile bağlantısını kurmak çok rahattır. Saniyede 17 resim çıkartabilen CmuCam'in özellikleri şöyle sıralanabilir:

- Renkli bir cismi takip edebilmek
- Objektif önündeki istenen bölgenin RGB (renk) seviyesini ölçebilmek
- Otomatik olarak ilk gördüğü cisme kilitlenip, onu izleyebilmek
- Motor kontrolünü direk olarak, kendi devresi üzerinden yapabilmek
- Görüntüyü 'tamamen' sayısal resim olarak, seri port üzerinden gönderebilmek
- İzlenen cismin görüntüsünü tamamen sayısal resim olarak gönderebilmek.

CmuCam ile bağlantı kurup, komut vermek, veri transferi yapmak, dalma seri port üzerinden olmaktadır. Seri port protokolü, byte'ları sıraya dizdiği için bu sırayı iyi gözetlemek ve RGB değerlerini çözüp resmi PC ortamına aktarabilmek gerekmektedir.



ŞEKİL 28.17:MAR önündeki servonun üzerine CMUCam monte edilmiştir.

Mesela, seri port üzerinden 'DF' harflerini gönderirsek, CmuCam bu komutu 'objektif önünde ne varsa resmini çek, geri yolla' olarak algılayacaktır ve sonuçları gene seri port üzerinden geriye gönderir.CmuCam PC'nin seri portu yerine, MAR üzerinde bulunan seri porta bağlıdır. MAR, kameraya bir komut gönderdiği zaman aynı port üzerinden kameranın cevabını algılamakta ve bunu diğer seri port üzerinden radyo modeme ve oradan da iş istasyonuna aktarmaktadır. CMUCam, Bölüm 16'da detaylı olarak incelenmiştir.

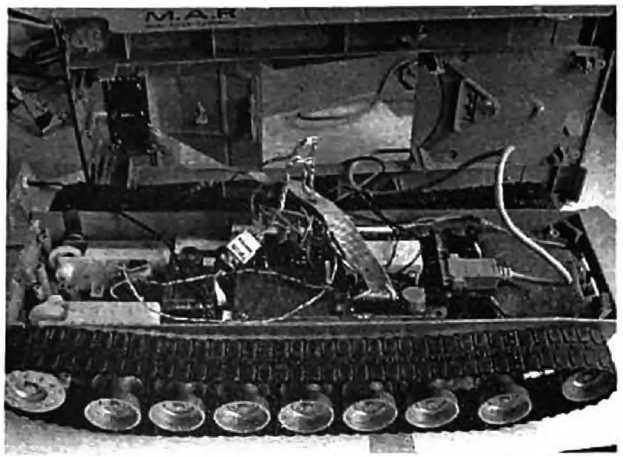
MAR Yazılımı

MAR mikroişlemci yazılımı, BASIC-Tiger yongasının kendi dili ile geliştirilmiştir. Mikroişlemciye yüklenecek olan yazılımın yaptığı temel iş, radyo modem ile iş istasyonundan alınan verilerin bir komut olup olmadığını kontrol etmek, eğer bir komut ise, komutun parametresini okumak, eğer parametre doğru ise komutu gerçekleştirmek. Örneğin robotun 180 derece kendi etrafında dönmesini sağlayabilmek için, `MAR> DON 180` şeklinde bir komut, seri port ve modem aracılığı ile BASIC-Tiger'a gönderilir. Eğer bu bir komut ise program parametreye bakar ve 8'nci portun 1'nci pinini 500ms boyunca yüksek yapar. Bu durumda L293B'nin IN1 girişi yüksek olacak ve soldaki motor IN1 girişi yüksek olduğu sürece çalışacaktır. Sonuçta MAR, kendi etrafında 180 derecelik bir manevra yapmış olacaktır.



ŞEKİL 28.18: MAR.

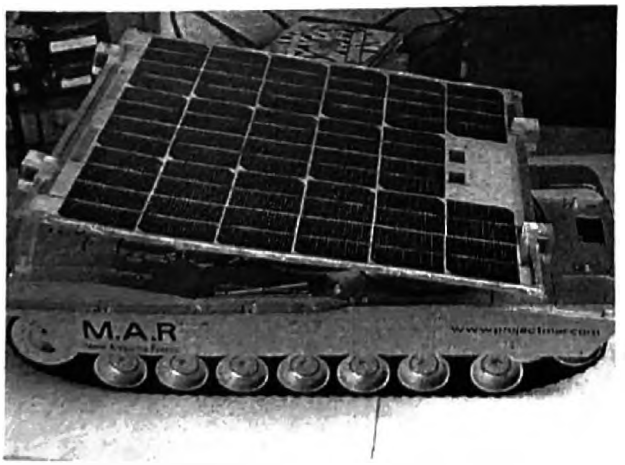
Uygulanan 500ms'lik sürenin MAR'ı gerçekten tam 180 derece döndürüp döndürmediğinden emin olamayız. Bunun birinci nedeni, zemin çok kaygan olabilir ve palet boşa dönmüş olabilir. Diğer bir sebep ise, motor pili yeterli enerjiyi sağlayamamış olabilir ve 500ms'lik bir süre 180 dereceden daha az bir dönme sağlamış olabilir. Ya da MAR yokuşta olabilir. Motorlar, robotu hiç hareket ettirmeyebilir. İşlem tamamlandıktan sonra, MAR tekrar komut bekleme moduna geçecektir. Dönme işleminin ne kadar başarılı olduğu kamera aracılığı ile kontrol edilebilir. Her şekilde, MAR'ın bulunduğu gerçek koordinatlarda sapma söz konusu olabilir. Burada tutulan koordinatlar tahmini olacaktır. İleride bu sapmaları minimize edecek devreler tasarlanarak robota eklenebilir.



ŞEKİL 28.19: MAR.

Multitasking

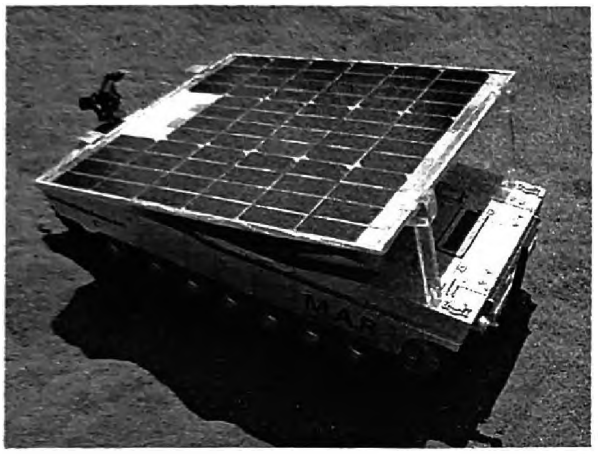
Multitasking, aynı anda bir kaç işlemi yapabilmek anlamına gelmektedir. Mikroişlemci olarak seçilen BASIC-Tiger'ın multitasking özelliği vardır. Örneğin aynı anda port 8'den motorları kontrol ederken, port 6'dan önüne çıkacak engelleri bekleyebilir. Bu özelliğin olmadığı mikroişlemcilerde, kesme (interrupt) alt rutinleri ile alt programlar çağırılabilmektedir. Burada ise, port 6'daki bir sinyali program algılayabilir ve port 8'deki işlemi durdurabilir.



ŞEKİL 28.20: MAR.

MAR Komut Yorumlayıcısı ve Komut Seti

Yukarıda da bahsedilen bu komut yorumlayıcısı, iş istasyonu tarafında çalışacak Visual Basic ile geliştirilmiş basit bir yazılımdır. Amacı, aldığı komutları radyo modem aracılığı ile MAR'a yorumlaması için iletmektir. İş istasyonu ile MAR'ın anlaşabilmesi için bir komut seti oluşturulmuştur. Bu komut seti şimdilik aşağıdaki gibidir. Projenin ilerlemesi durumunda bu komutlar çoğalabilir.



ŞEKİL 28.21: MAR.

MAR'a Gönderilebilen Komutlar

SURK [x,z] x,z:

Metre cinsinden girilen bu koordinatlara gider ve durur.

SURT [yon,sure]:

yon=1: Girilen süre kadar MAR'ı ileriye götürür

yon=0: Girilen süre kadar MAR'ı geriye götürür.

MOT [motor,sure]:

Motor=1 ya da 2. Girilen motoru girilen süre kadar hareket ettirir.

HIZ [kademe]:

kademe=1 ile 5 arasında bir değer. Motorların döneceği hızı belirler.

DURUM:

MAR'ın motor hız durumunu, tahmini koordinatlarını, tahmini pil süresini iş istasyonuna gönderir.

DON [derece]:

MAR'ı istenilen yönde girilen açı kadar döndürür.derece: -360 ile +360 arasında herhangi bir değer olabilir.

KDON [derece]:

MAR'ın önündeki kamerayı istenilen yönde girilen açı kadar döndürür. derece: -180 ile +180 arasında herhangi bir değer olabilir.

FCEK:

MAR'ın fotoğraf çekmesini ve iş istasyonuna iletmesini sağlar.

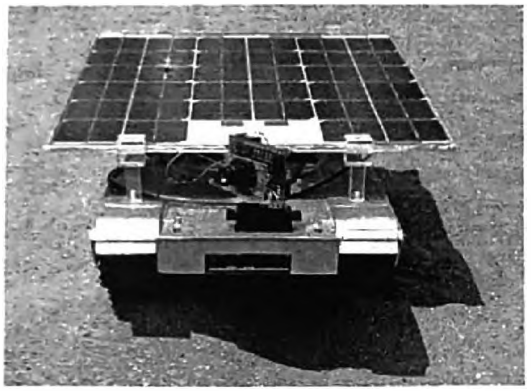
SARJ [mod]:

mod=1: MAR şarj moduna geçer ve moddan çıkarılana kadar modem hariç hiç bir elektronik birime enerji göndermez.

mod=0: MAR'ı şarj modundan çıkarır.

DUR

MAR'ın her ne olursa olsun tüm motorlarını acil durdurur.



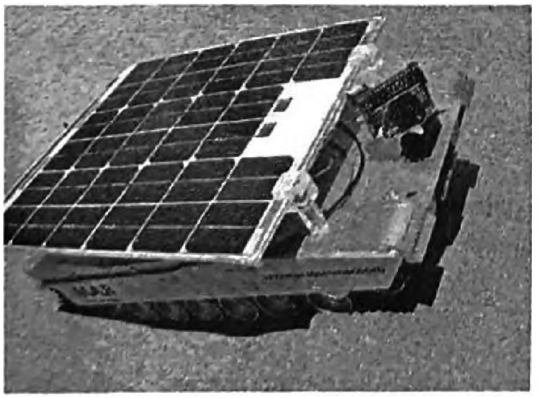
ŞEKİL 28.22: MAR.

MAR Terminal Arayüzü ve Projenin Devamı

MAR projesi teslim edilen bu tez ile sona ermiş değildir. Projenin tez olarak teslim edilen ilk basamağında, temel işlevleri uzaktan yapabilen paletli bir robot tasarımı yapılmıştır. Projenin devamında robotun iş istasyonu üzerinden gerçek anlamda kullanılmasını sağlayacak bir yazılım geliştirilme aşamasındadır.

Görüntü İşleme Yazılımı

MAR'ın çektiği fotoğraflar, ortam ve ışık nedeni ile her zaman net çıkmayabilir. Çekilen fotoğrafları en iyi duruma getirilmesi için bir görüntü işleme sürecine tabi tutulması gereklidir. Bu da bir yazılım ile mümkün olmaktadır. Bu yazılım MAR projesinin paralelinde başka bir tez konusu olarak Astronomi Bölümü mezunlarından olan Ergem PEKER tarafından sürdürülmektedir.



ŞEKİL 28.23: MAR.

Bu yazılımın da tamamlanması sonucunda, MAR'ın iş istasyonu yazılımı ile birleştirilecek ve MAR projesi tamamlanacaktır. Project MAR ile ilgili ilerlemeleri `www.projectmar.com` adresinden takip edebilirsiniz.

YİRMİDOKUZUNCU BÖLÜM

29

FLOPPY ROBOT

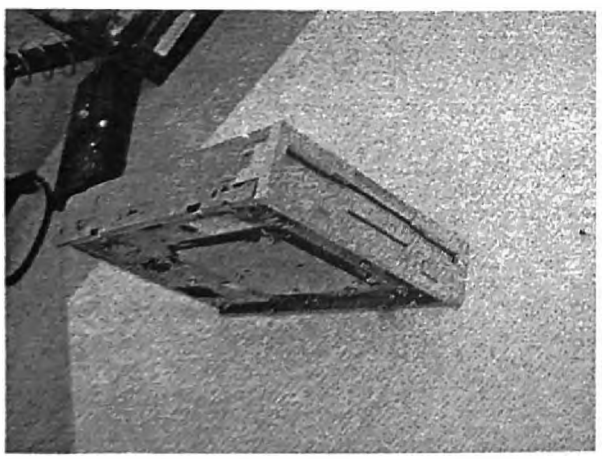
UYARI

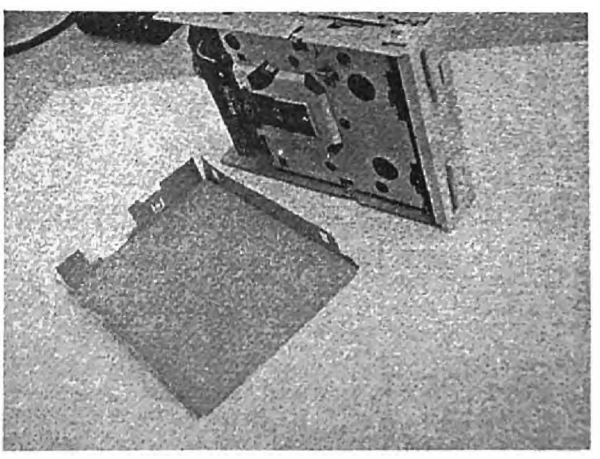
Bu uygulamada kullandığınız disket sürücü, bir daha kullanılamaz hale gelecektir. Ayrıca bu uygulamada bazı floppy sürücülerin markasına göre pin numaralarında farklılık olabilir. Bu nedenle, söktüğünüz sürücünüzün markasını araştırıp, bağlantılarını Internet üzerinden bulmanız ve devreyi tekrar düzenlemeniz gerekebilir. Bu nedenle, uygulamaya başlamadan önce projenin kaynağını incelemenizi öneriyorum.

Bu bölümde 3.5"lik floppy disket sürücüsü ile basit bir robot yapacağız. Internet üzerinde bu proje ile ilgili birkaç site bulmak mümkün. Projenin esas kaynağı olan site `http://ohmslaw.com/robot.htm` Bu sitede floppy robotun farklı versiyonları da mevcut. Araştırdığım diğer siteler ise kaynak olarak genelde bu siteyi kullanmış. Ayrıca `http://www.bilimonline.8k.com` sitesinde de aynı projenin kısmen Türkçeleştirilmiş versiyonu ve oldukça faydalı bilgiler de var. Her iki siteyi de bu yazıya kaynak olarak kullandım.

Proje için ilk ihtiyacımız olan şey, adı üstünde bir floppy disket sürücü. Ayrıca güç devresi için bir adet 7805 Regulatör entegresine, bir adet Floppy data kablosuna, güç kablolarına, bir adet 9 Voltluk pile, 2 adet diskete ve birkaç tane anahtara ihtiyacımız var. Mekanik aksamı oluşturmak için gereken plastik ya da metal parçaları ve tekerlekleri bulmak biraz sizin yaratıcılığınıza kalıyor. Sonuçta robotun mekaniği oldukça basit bir mantığa dayanıyor.

İlk başta floppy sürücünün içini açarak işe başlıyoruz (Şekil 29.1) . Önce üst kapağı çıkarın (Şekil 29.2). Daha sonra ön kapağı da çıkarın (Şekil 29.3). İşinize yaramayacağını düşündüğünüz bütün bölümleri çıkarın. Elinizde sadece motor ve kafa mekanizması kalsın (Şekil 29.4 – Şekil 29.6). İki adet 3.5"lik disket edinin ve kapaklarıyla ortalarındaki yuvarlak kısımları çıkarın (Şekil 29.7). Çıkardığınız yuvarlak kısımları arka arkaya yapıştırın (Şekil 29.8 – Şekil 29.9). Deliklerin üst üste gelmesine dikkat edin (Şekil 29.10). Yapıştırdığınız bu parçayı da sürücünün ortasındaki yuvarlak kısmın tam ortasından yapıştırın (Şekil 29.11).

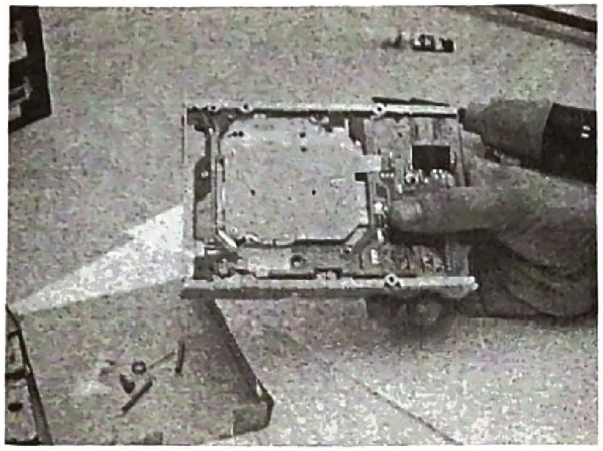




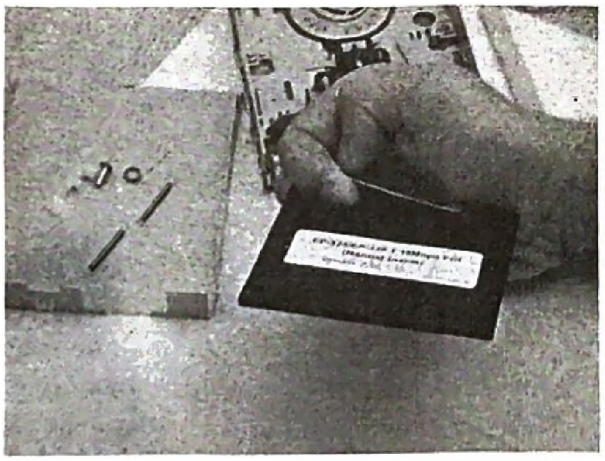
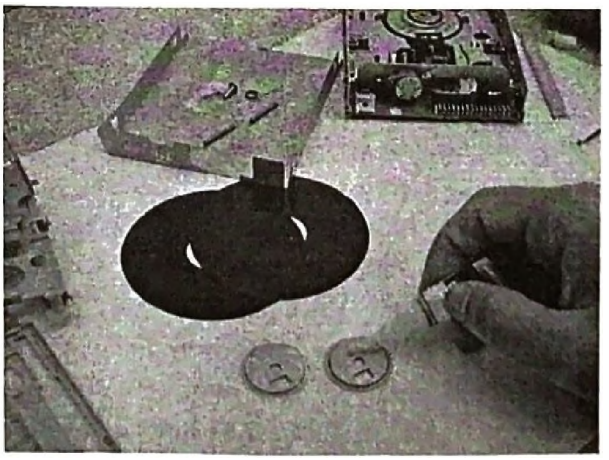
ŞEKİL 29.1: (solda) ve **ŞEKİL 29.2:** (sağda)



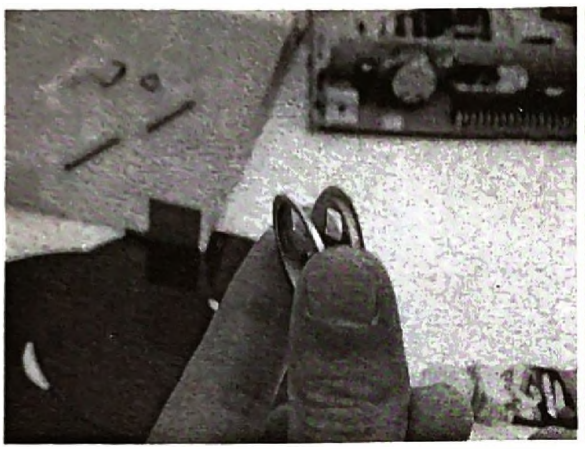

ŞEKİL 29.3: (solda) ve **ŞEKİL 29.4:** (sağda)

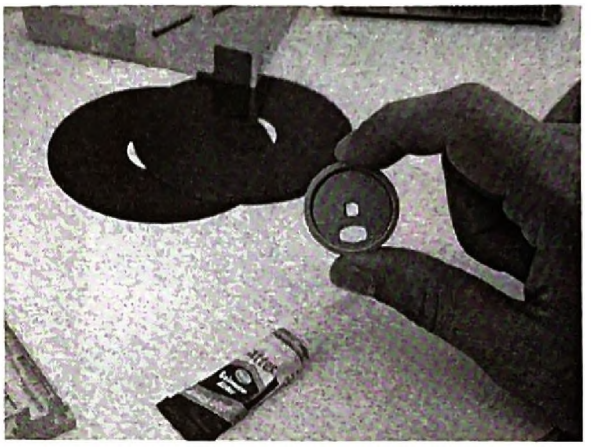



ŞEKİL 29.5: (solda) ve **ŞEKİL 29.6:** (sağda)

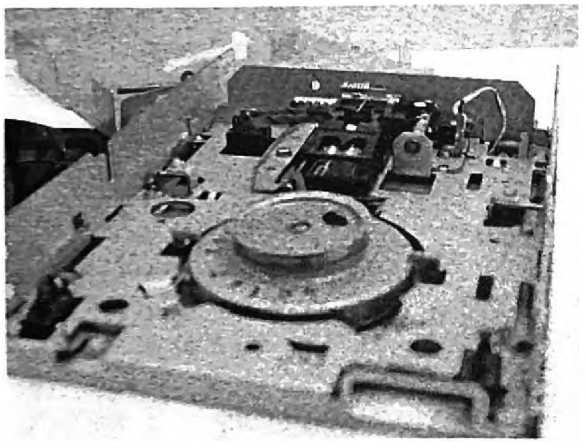



ŞEKİL 29.7: (solda) ve **ŞEKİL 29.8:** (sağda)



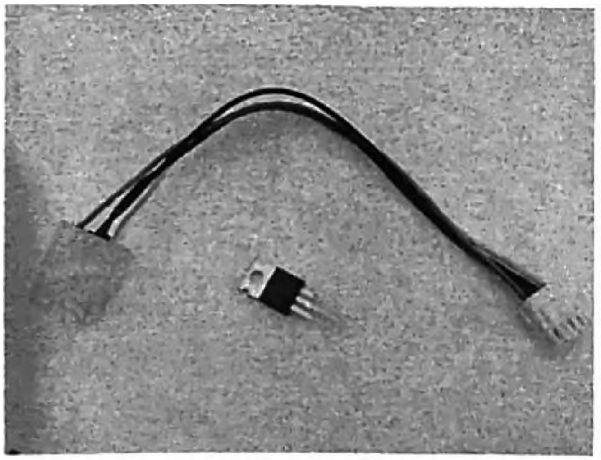
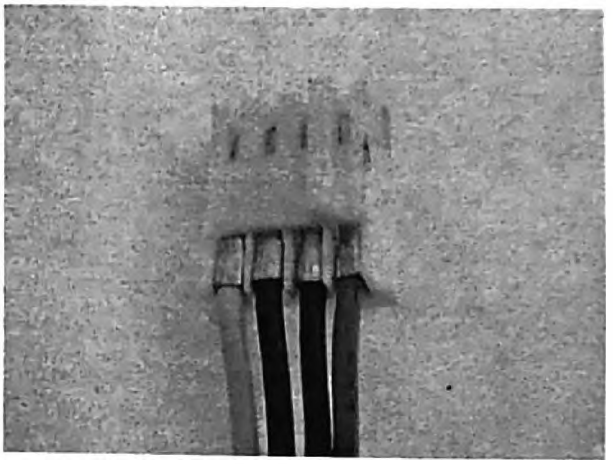


ŞEKİL 29.9: (solda) ve **ŞEKİL 29.10:** (sağda)

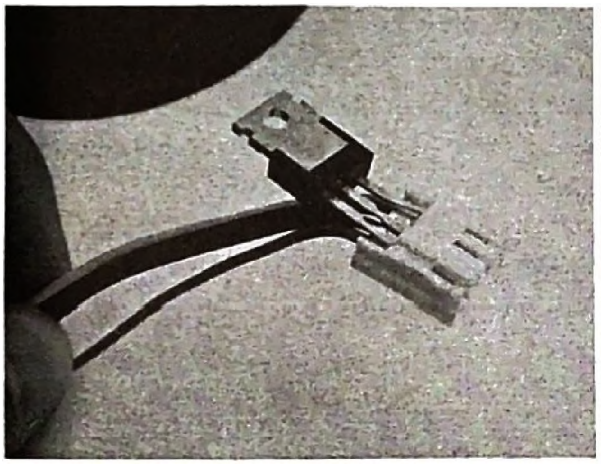


ŞEKİL 29.11:

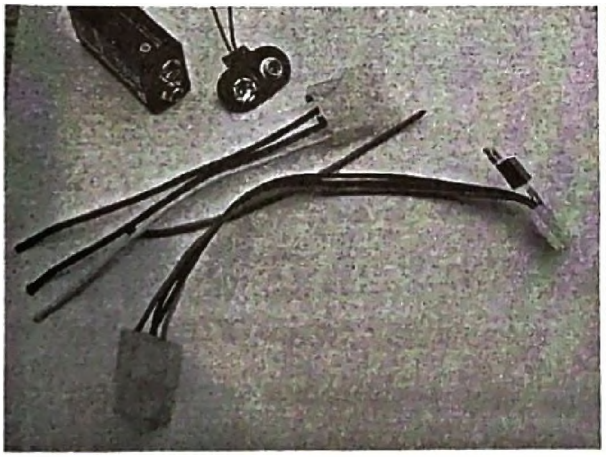
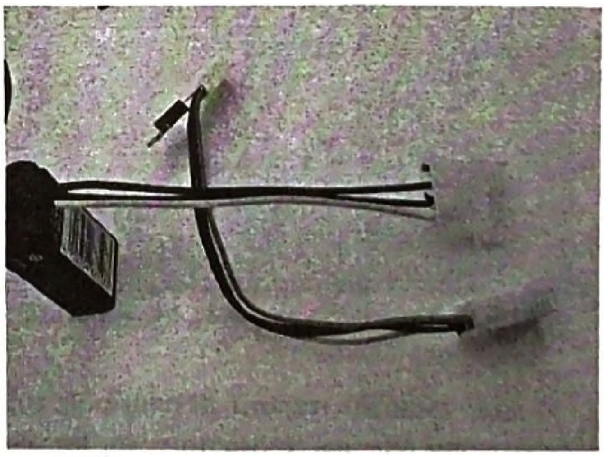
Mekanik olarak daha fazla bir şeyler yapmadan önce, sürücüye enerjiyi sağlayacak güç devresini yapmakta fayda var. Bunun için 7805 voltaj regülatör entegresi kullanıyoruz (Şekil 29.12). Resimlerde gördüğünüz güç kablolarını eski bir bilgisayar kasasının içinden sökebileceğiniz bir güç kaynağından elde edebilirsiniz (Şekil 29.13). 7805'i Şekil 29.14'teki gibi floppy güç kablosuna lehimleyin. Bağlantı şemasını Şekil 29.39'da görebilirsiniz. Güç sağlayıcı olarak, 9 Voltluk pil kullanın (Şekil 29.15). Pilin ucunu, sürücüyü besleyecek bir çıkış haline getirin (Şekil 29.16 – Şekil 29.17). Anahtarlardan bir tanesini Şekil 29.40'daki gibi floppy data kablosu üzerine lehimleyin. Diğerini ise pilden (+) gelen kabloya bağlayın (Şekil 29.18 – Şekil 29.19). Bu devre aynı zamanda disket sürücüyü test etmek için ya da robot bittikten sonra kontrol etmek için de kullanılabilir. Sürücüyü çalıştırmak için data kablosundaki bağlantıları bilmeniz gerekir. Tablo 29.1'deki liste, floppy arabirim konnektöründeki bağlantıları ve ne işe yaradıklarını gösteriyor. Disketi döndüren motoru kontrol etmek için kullanacağınız pinler 12 ve 16 numaralı pinler ve herhangi bir toprak pinidir. Burada kablo üzerinde bir atlama yapmanız gerekiyor. İletken bir kabloyu 12 numaralı pine, diğerini ise herhangi bir toprağa bağlayın (Şekil 29.20).

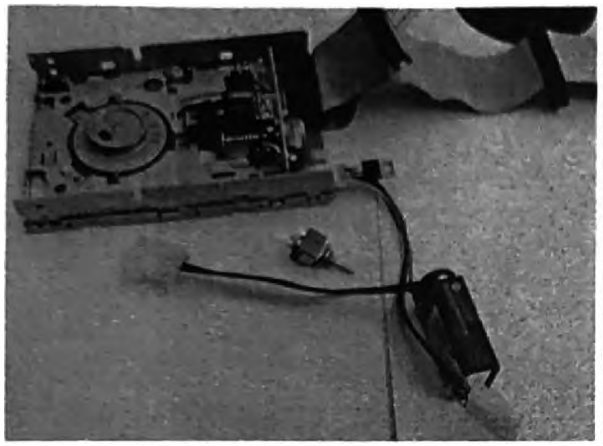
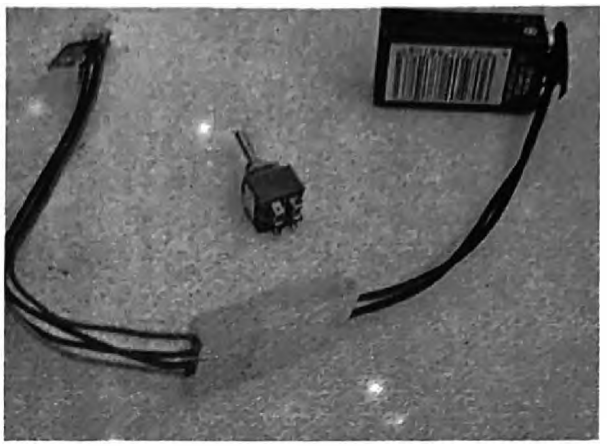
ŞEKİL 29.12: (solda) ve **ŞEKİL 29.13:** (sağda)




ŞEKİL 29.14: (solda) ve **ŞEKİL 29.15:** (sağda)

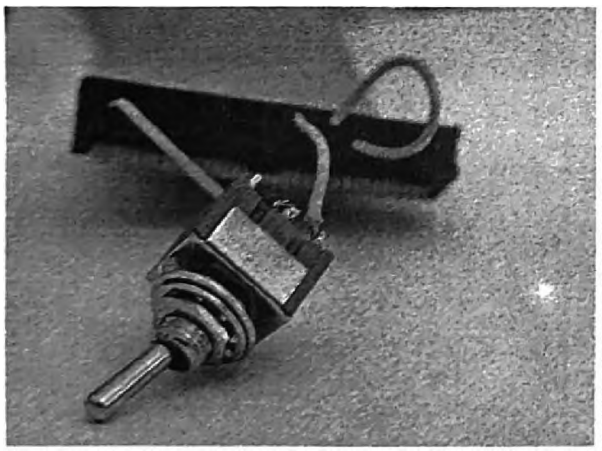
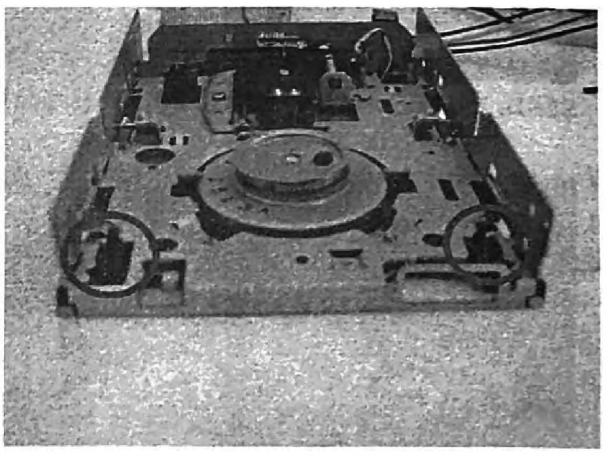



ŞEKİL 29.16: (solda) ve **ŞEKİL 29.17:** (sağda)

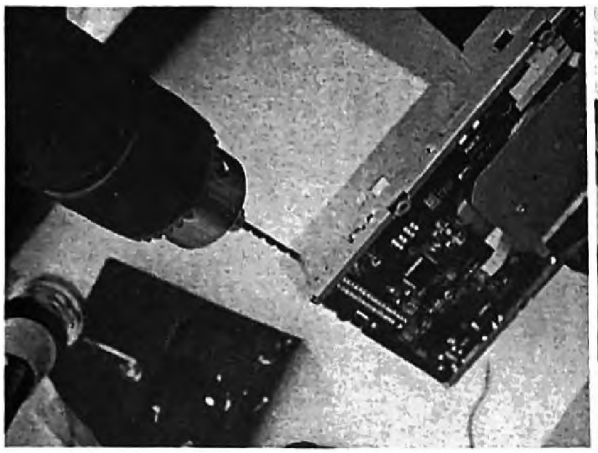
ŞEKİL 29.18: (solda) ve **ŞEKİL 29.19:** (sağda)

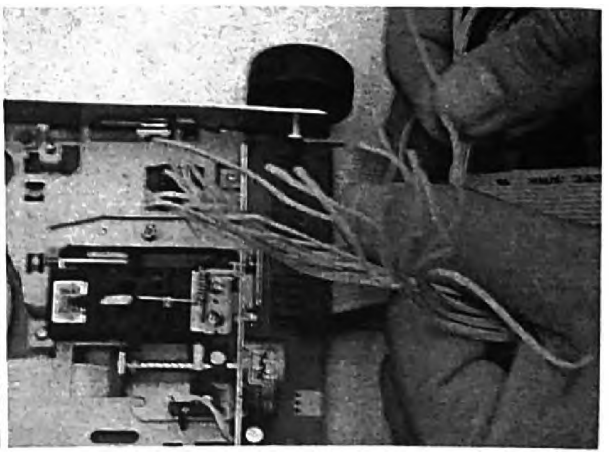
Daha sonra, anahtarın bir ucunu 16 numaralı pine, diğerini de toprağa bağlayın. Güç kablosunu ve pili sürücüye bağlayın. Data kablosunun bir ucunu soketin 1 numaralı pini floppy'nin 1 numaralı pinine gelecek şekilde sürücüye bağlayın. Sürücü üzerinde yazmaya karşı korumayı algılayan bir anahtar bulunur. Yazma koruma switch'lerini bulun ve onlara basın (Şekil 29.21). Güç devresini sürücü üzerine bağlayın Kullanılmayan kabloları ayırın ve kesin (Şekil 29.23). Ayrıca kabloların birbirlerine değmemesine dikkat edin ve gereken yerlerde iletken olmayan bir bant kullanın (Şekil 29.24). Güç devresini sürücüye bağladıktan sonra switch'e bastığınızda ortadaki motorun dönmesi gerekiyor.

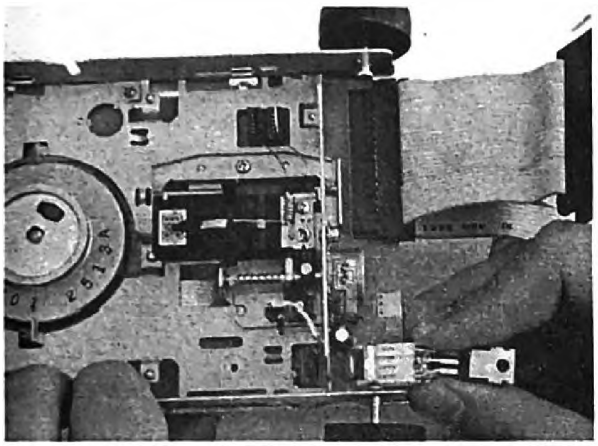
ŞEKİL 29.20: (solda) ve **ŞEKİL 29.21:** (sağda)

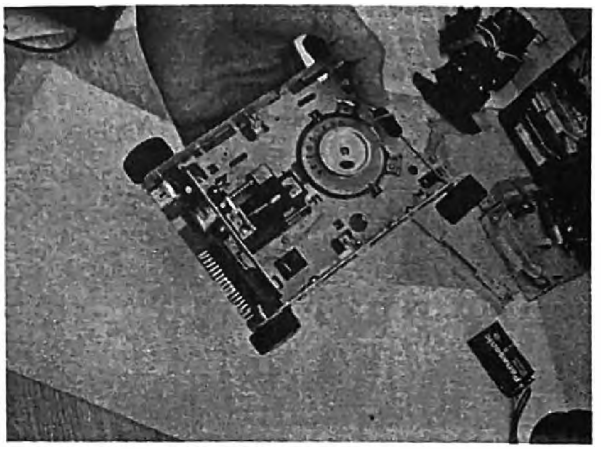
Sıra, ortadaki motoru döndürerek robotun nasıl ilerleyeceğine geldi. Bunun için birkaç bükülebilir parçaya ihtiyacımız var. Bu uygulamada pirinç tabaka kullanılmış. Siz de herhangi bir bükülebilir metal kullanabilirsiniz. İlk olarak 8cm X 1cm'lik iki plaka elde edin (Şekil 29.30). Her ikisini de ortadan ikiye bükün (Şekil 29.31). Üst taraflarında birer delik açın (Şekil 29.32). Sürücünün ön tarafına Şekil 29.33'teki gibi yapıştırın ve ortalarından metal telden birer mil geçirin (Şekil 29.34).





ŞEKİL 29.22: (solda) ve **ŞEKİL 29.23:** (sağda)

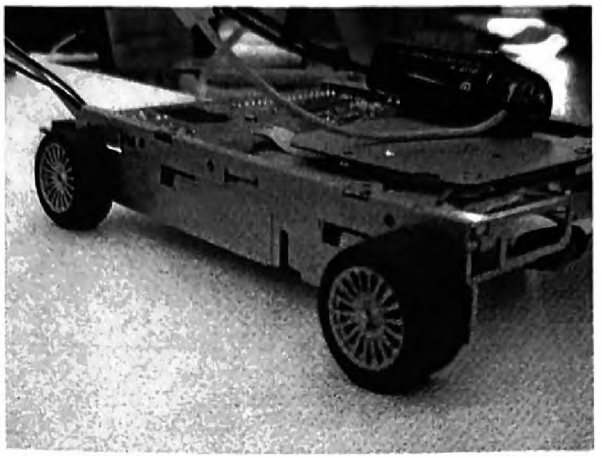




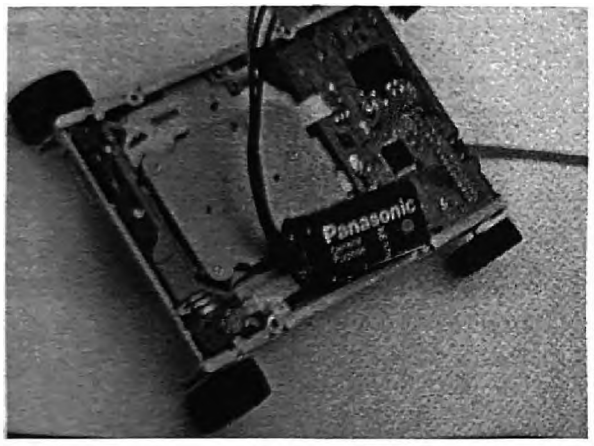
ŞEKİL 29.24: (solda) ve **ŞEKİL 29.25:** (sağda)

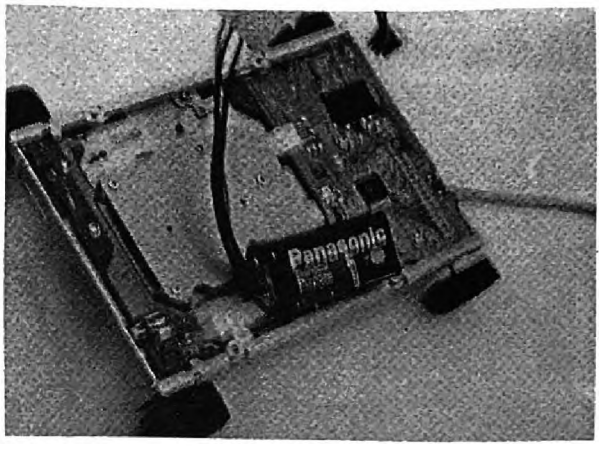
Ön taraftaki milin ortasına Şekil 29.36'daki gibi plastik conta gibi bir şey geçirin. Daha sonra Şekil 29.35'teki gibi daha önceden yapmış ve ana motor üzerine yapıştırmış olduğumuz metal yuvarlak kısım ile, plastik conta arasına bir kayış ya da lastik gibi bir şey yerleştirmelisiniz. Sonuçta Şekil 29.37'deki gibi bir görüntü elde edersiniz. Daha sonra herhangi bir oyuncak arabadan sökebileceğiniz 4 adet tekerleği Şekil 29.38'deki gibi millerin uçlarına takın. Ben kendi yaptığım uygulamada, metal bükmek yerine Şekil 29.22'deki gibi matkap ile milin geçebileceği birer delik açtım. Hangisi daha kolayınıza geliyorsa, o yöntemi uygulayın. Bu noktada robotun mekanik aksamını tamamlamış bulunuyorsunuz. Robotun bitmiş halini Şekil 29.25 – Şekil 29.29'da görebilirsiniz. .





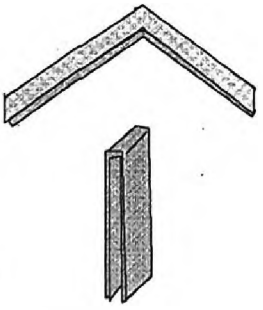
ŞEKİL 29.26: (solda) ve **ŞEKİL 29.27:** (sağda)





ŞEKİL 29.28: (solda) ve **ŞEKİL 29.29:** (sağda)



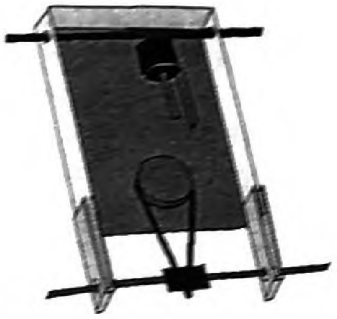


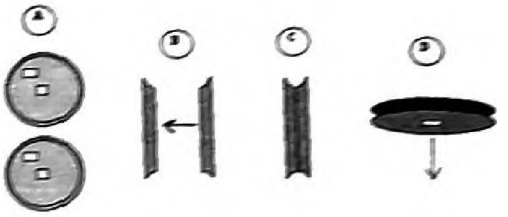
ŞEKİL 29.30: (solda) ve **ŞEKİL 29.31:** (sağda)





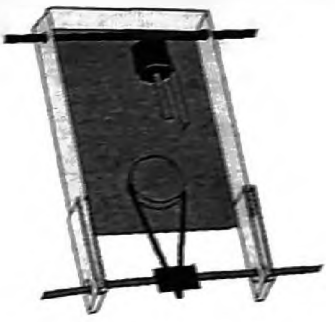
ŞEKİL 29.32: (solda) ve **ŞEKİL 29.33:** (sağda)



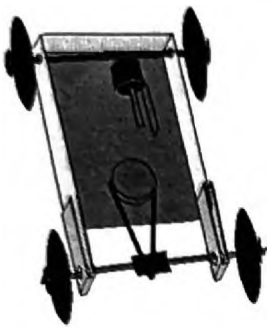


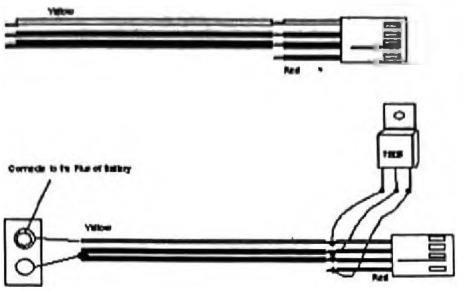
ŞEKİL 29.34: (solda) ve **ŞEKİL 29.35:** (sağda)



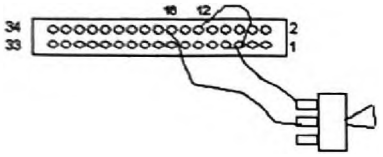


ŞEKİL 29.36: (solda) ve **ŞEKİL 29.37:** (sağda)





ŞEKİL 29.38: (solda) ve **ŞEKİL 29.39:** (sağda)



ŞEKİL 29.40:

TABLO 29.1: Floppy Pinlerinin İşlevleri

Pin Numarası	Açıklama
1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33	Toprak
2	Yüksek yoğunluk seçimi
8	Gösterge Sinyali
Pin 10	A: Motor açık
Pin 12	sürücü seçimi B:
Pin 14	sürücü seçlmi A:
Pin 16	B: Motor açık
Pin 18	Yön
Pin 20	Adım sinyali (Her bir darbe motoru bir adım hareket ettirir.)
Pin 22	Veri yaz
Pin 24	Yazma koruması kapalı
Pin 26	Track 0
Pin 28	Yazma koruması açık

TABLO 29.1: Floppy Pinlerinin İşlevleri

Pin Numarası	Açıklama
Pin 30	Veri oku
Pin 32	Kafa seç
Pin 34	Disk değiştir

Bu robot sadece ilerleyebiliyor. Aynı teknikleri kullanarak dönebilen bir robot da yapmak mümkün. Bunun için 2 adet floppy sürücüye ihtiyacımız var. Bununla ilgili detaylı bilgileri `http://ohmslaw.com/floppy2.htm` adresinden elde edebilirsiniz. Yine aynı site üzerinde robotun bilgisayar kontrolü için derlenmiş bir program da mevcut. Bu basit uygulamayı geliştirmenizi öneriyorum; floppy sürücüler bu aralar oldukça ucuz, fakat bu iş için yeni bir floppy almaya gerek yok. Kullanılmış bir floppy sürücü bulmaya gayret edin. Bağlantıları dikkatli yapın ve her iki siteyi de projeye başlamadan önce mutlaka inceleyin. Bu robot projesi daha önce gerçekleştirdiğimiz PIC ile çalışan çizgi izleyen robot kadar komplike değil. Gerçekte burada yaptığımız, bence tam bir robot da değil, fakat yine de faydalı ve eğlenceli bir uygulama olduğunu düşünüyorum.

Kaynaklar

PIC Programlayıcı

ERA Bilgi Sistemleri (Infogate) firmasının, "Adım Adım PICMicro programlama" kitabının içerisinden çıkan baskılı devre kartı üzerine kurulmuş ProtoPIC Lite Programlayıcısı kullanılmıştır. Bu programlayıcı için www.tekno-market.com adresini ziyaret edebilirsiniz.

Kitaplar

Mikroişlemciler ve Bilgisayarlar, Dr. Haluk Gümüşkaya, Alfa Yayınları

Altaş – Mikrodenetleyiciler ve PIC programlama, Orhan Altınbaşak

ERA Bilgi Sistemleri Yayıncılık ve Bileşim A.Ş (www.infogate.org)

Newses Elektronik Mühendisinin Cep Kitabı

Her Yönüyle Paralel Port, Jan Axelson

Her Yönüyle Seri Port, Jan Axelson

Adım Adım PICMicro Programlama

Yeni Başlayanlar için Mikrokontrolörler

Mikrokontrolörlerle Endüstriyel Kontrol

Mikrokontrolör ve Robotik

Web Siteleri

http://www.elektronikhobi.com

Fikret Sadak, Temel Devre Elemanları makaleleri http://www.elektronikhobi.com

Tom Engdahl, http://www.hut.fi/Misc/Electronics/circuits/irdetector.html

Ahmet Usta, http://www.deliprof.net

Sam J. Engstrom, http://www.pp.clinet.fi/~newmedia/sengelec/index.html

Erkan Birdane, http://picpro.cjb.net

Peter Ouwehand, http://home.iae.nl/users/pouweha/index.shtml

Barbaros Aşuroğlu, http://www.antrak.org.tr/gazete/081999/barbar.htm

http://www.rentron.com/SerialServo.htm

http://www.kmitl.ac.th/~kswichit/LFrobot/LFrobot.htm

http://ohmslaw.com/robot.htm

http://www.bilimonline.8k.com

http://www.belgeler.org/cdersi/cdersi-sayilar-16.html

http://dummy.kreta.dyn.priv.at/temp/howstuffworks/motor.htm

http://www.riccibitti.com

http://www.cs.unibo.it/~lanconel

http://www.scitoys.com/

Dizin

.asm dosyası 130, 168
.hex dosyası 135
.inc dosyası 153
10'luk sayı sistemi 7
2'lik sayı sistemi 7
2'lik tabana dönüşüm 8
4 bantlı direnç 12
5 bantlı direnç 12
555 entegresi 72
8 bit 7
AC gerilimi DC gerilime çevirmek 21
ADC 42, 43, 44, 198
adım motor 28, 162
adım motor entegreli sürücü devresi 28
adım motor kontrolü 28
adım motor kontrolü yazılımı 30
adım motor sürücüsü 162
adres 6
adres aralığı 6
ağ haberleşmesi 69
AH saklayıcısı 71
AIBO robotu 196
akım 5
akım kazancı 21
akış kontrolü 70
akümülatör 133
alt program 148
AM radyo vericisi 112
Amper 25
anahtar 14, 20
anakart 4
Analog - Digital Converter 42
analog sinyal 42
AND kapısı 74
anot 21
araştırma robotu 212
ASC fonksiyonu 38
ASCII kodu 36
asenkron iletim 66
asit 93
Assembler derleyicisi 130
Assembly 130, 168, 184
assembly programlama 71
ayarlı direnç 11
bakır plaka 90
bank 132
Base Adresi 6
BASIC Tiger 217
Basic 168, 184
Basic Stamp 168, 184, 188
Basic Tiger 168, 184
baskı devre 90
baskı devre hazırlamak 90
baskı devre kalemi 90
baud rate 67
baz istasyon 213
BCF komutu 136, 142
bellek gözü 148
beyin 188
beyz 21
BIOS 42
Bi-Directional paralel port 42
bilgisayar 4
bilimsel hesap makinesi 7
binary sayı sistemi 7
Bi-Polar transistör 23
biti terslemek 16
bits per second 67
board 10, 90
bobin 20
Boe-Bot 190
Boole cebiri 76
boş modem bağlantısı 69
bps 67
BSF komutu 136, 141
BTFSS komutu 155
buffer 70
bug 189
busy flag 151
buton 14, 75, 140
butona basınca sönen led 140
C 168
C kodu derlemek 99
C++ ile paralel port kontrolü 99
CALL komutu 148
CARRY biti 165
cc programı 99
CCD 118
CGRAM komutu 38, 157
chat 82, 112
cisim algılama 221
clock sinyali 66
CLRF komutu 136, 141
CMUCam 118, 223
CNC tezgahı 50
COM portu 71
Comparator 198
CONTROL portu 4, 6, 15, 34, 36
CTS 67, 69, 70
çalışma gerilimi 11

çip 126, 184
çizgi algılama sensörü 197
çizgi izleyen robot 197
çizgi sensörü 198, 202
çözünürlük 119
DAC 42, 50
dalga sürümü 31
darbe genişlik modülasyonu 178, 197
data pini 4
DATA portu 4, 6, 15, 20, 42, 81, 146
DB-25 68
DB-9 68
DC elektrik motoru 178
DC motor 50
DC motor kontrolü 50
DC motor pozisyon kumanda elektroniği 178
DDRAM fonksiyonu 38
debug 189
dekuplaj kondansatörü 128
delikli plaka 90
demodülatör 66
derleme 135
derleyici 168
devre kartı 188
Digital-Analog Converter 42, 50
dijital kamera 118
dijital ölçüm 104
dinamik balans 196
dinamik IP adresi 81
direncin hassasiyeti 13
direnç 5, 9, 11, 128
direnç renk kodları 12
dirençlerin paralel bağlanması 13
dirençlerin seri bağlanması 13
diyot 21, 54, 163
donanım kesmesi 67
dosya saklayıcısı 130, 132
döngü 146
dönüşüm 7, 18
DS1621 entegresi 104
DSR 69
DTE 69
DTR 67, 69
Dummy Terminal 175
duty cycle oranı 51
EEPROG 126
EEPROM bellek 126
EIA 68
Electronics Industries Association 68
elektromanyetik anahtar 20
elektronik 4
emiter 21
empedans 5
entegre 28
entegre devre 188
EPROM 126
EPROM silici 126
etiket 148
ev otomasyonu 80, 104
Exclusive OR kapısı 75
Exclusive-Or operatörü 16
F saklayıcısı 133
fan pervanesi 50
fırçalı bobin rotor 178
filtre 198
fiziksel değerleri sayısal değerlere çevirmek 42
Flash bellek 126
Flash memory 126
Flash teknolojisi 126
floppy 230
floppy robot 230
FOR-NEXT döngüsü 31, 146
foto direnç 11
foto transistör 45, 46, 108, 118, 198
FTP 81
full duplex 66
gecikme 31, 146
gecikme alt programı 146
geciktirme alt programı 146
geri besleme bağlantısı 70
geri besleme potansiyometresi 178
germanyum diyot 22
GOTO komutu 142
görev çevrimi oranı 51
görüntü algılama 118
ground 28
güneş paneli 214
half duplex 66
hard disk güç kablosu 24
hata giderme 189
H-Bridge 50
hesap makinesi 7, 18
Hex kodu 168
Hex sayı sistemi 9
hexadecimal sayı sistemi 17
Hexadecimal sayı sistemi 6
HIGH komutu 169
HONDA P3 robotu 196
Hyperterminal 172
I/O port 71
IIC veri yolu 104
Infra Red ışık 46
INP komutu 15
Input komutu 120
INT 14H kesme komutu 71
Internet 80, 112
Internet Servisi Sağlayıcı 81
Internet üzerinden ev otomasyonu 80, 104
IP adresi 80
IPCONFIG 82

IRQ....67
ISR....67
ISS....81
ışığa bağımlı analog sinyal....45
ışık....108
ışık yayan diyot....10
iç direnç....46
include komutu....152
inpout32.dll....8
insert mode....98
inverter....75
istemci....81
işlem istasyonu....213
işlemci....21
işlemsel kuvvetlendiricidir....198
işletim sistemi....98
Java Stamp....184
joystick....58
kamera....118
karbon direnç....11
Karnaugh haritası....76
karşılaştırıcı....198
katot....21
kesme....67, 225
kesme hattı....67
kesme servis rutini....67
KeyDown olayı....62
KeyPreview özelliği....62
KeyUp olayı....62
kıyaslayıcı....198
kızıl ötesi ışık....46
kızıl ötesi iletişim....72
kızıl ötesi led....46, 108
kiloohm....12
kollektör....21
komut saklayıcısı....35, 151
komut seti....226
komut yorumlayıcısı....226
komutGonder() fonksiyonu....38
kondansatör....21, 128
konsol uygulaması....98
kontrast....151
kontrol objesi....71
kontrol saklayıcısı....37
kontrol sinyali....67
konum kontrolü....28, 162
kömür....29
kristal kontrollü osilatör....128
kristal osilatör....112
kumanda....58
kutup....50
kuvvetlendirici....198
kütüphane dosyası....100
L293B....54, 220
L293B entegresi....50
L7805 entegresi....105
lazer....108
lazer ile cisim algılama....108
LCD panel....34, 150, 170, 185
LCD panel kontrol yazılımı....37
LCD panel kontrolü....34
LCDBUSY fonksiyonu....155
LCDCLEAR komutu....156
LCDDMODE komutu....157
LCDEMODE komutu....156
LCDPUTCHAR komutu....157
LCDPUTCMD komutu....158
LCDSDDA fonksiyonu....157
led....4, 9, 10, 130, 146, 169, 190
LED diyot....10
ledin çalışma gerilimi....11
Lego....196
Light Emitting Diode....10
link....66
Linux....98
Linux ile otomasyon....98
Linux'ta C kodu derlemek....99
LM2936 devresi....105
lojik programlama....4
lojik sinyalleri dönme hareketine çevirmek....28
lojik voltaj....4, 20, 42, 81, 169
loop-back bağlantısı....70
LOW komutu....169
makine dili....127
mantık kapısı....74
manyetik kutup....50
MAR....212
MAR projesi....122
maske....16
MAX232 sürücüsü....174
megaohm....12
megapixel....119
meşgul bayrağı....35, 151
metin dosyası....98
metin editörü....98
miknatıs....20, 50
Microsoft Winsock Component....82
mikroişlemci....4, 47, 130, 217
mikrokontrolör....188
Mobil Araştırma Robotu....212
model araba....58
modem....66, 69
modülatör....66
monitörün çözünürlüğü....119
Mors alfabesi....112
motor kontrolü....204
motor kömürü....29
motora normal adım attırmak....31
motora ters adım attırmak....31
motora yarım adım attırmak....31

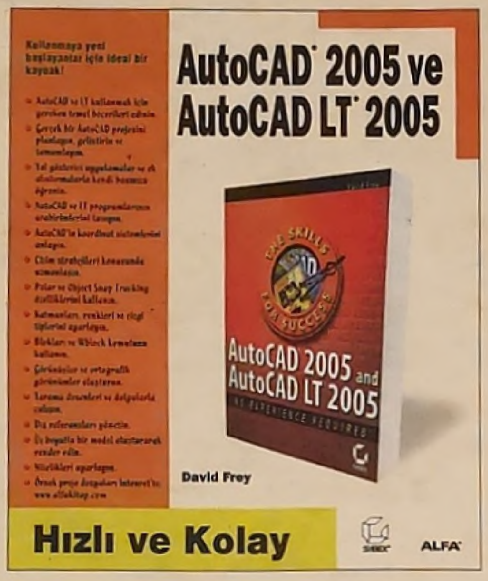
MOVLW 133
MOVLW komutu 133, 141
MOVWF komutu 133, 142
MPASM 130, 143, 168
MPASM derleyicisi 127
MPLAB 134
MSComm 71
multitasking 225
N kutbu 50
NAND kapısı 75
NASA 212
NOP komutu 146
NOR kapısı 75
North kutbu 50
NOT AND kapısı 75
NOT OR kapısı 75
Notepad 98, 130
NPN 21
null modem bağlantısı 69
nüfus evrilmesi 108
off-time 51
ohm 11
oje 90
oksitlenme 20
onaltılık sayı sistemi 6, 17
onaltılık sisteme çevirmek 18
onluk sisteme çevirmek 18
on-time 51
Opamp 198
OR kapısı 74
osilatör 112, 128, 198
osilatör devresi 128
otomasyon 98
OUT komutu 8, 20
output komutu 190
ölçüm 104
öteleme 164
P16PRO yazılımı 135
paralel bağlı dirençler 14
paralel port 4, 20, 80, 98
paralel port ile röle kontrolü 81
paralel port ile veri çıkışı 7
paralel port ile veri girişi 14
paralel port kontrolü 20
paralel port veri çıkışı 9
paralel portun adres aralığı 6
paralel portun taban adresi 15
Pathfinder 212
PAUSE komutu 169
PBASIC 188
PC 69
PC ile adım motor kontrolü 28
PC kontrollü termometre 104
perihedrol 93
PFE 130
PIC 126, 146
PIC 16F84 çipi 126, 127
PIC Assembly 127, 130, 168
PIC Basic 168
PIC Basic derleyicisi 168
PIC Basic ile servo kontrolü 179
PIC deneme kartı 127
PIC ile adım motor kontrolü 162
PIC programlama 47, 126
PIC programlayıcı devresi 127
PIC programlayıcısı 168
PIC16C84 çipi 126
PIC16F84 çipi 127
PicAll yazılımı 135
piksel 119
pin 4
pinin terslenmesi 15
planetar dişli 178
PNP 21
P-n-P baskı devre 90
P-N-P baskı devre 91
polarizasyon 21
PonyProg 104
population inversion 108
port 4, 82
portun Base Adresi 6
portun taban adresi 6
Positive20 90
POST 71
power on self test 71
PRINT komutu 172
Programmer's File Editor 130
protokol 66, 69
Proton+ Lite 168
ProtoPIC 127
Pset komutu 120
pull-up direnci 128
Pulse Width Modulation sinyali 51
PWM 178, 197
PWM sinyali 51
QRIO robotu 196
radyo modem 217, 218
radyo sinyali 112
RC devresi 128
read mode 98
Reduced Instruction Set Computer mimarisi 126
regüle devresi 22
renk kodları 12
renk takibi 122
reset devresi 128
RGB 119
RGB fonksiyonu 120
RISC mimarisi 126
robot 196, 212
robot gözü 45

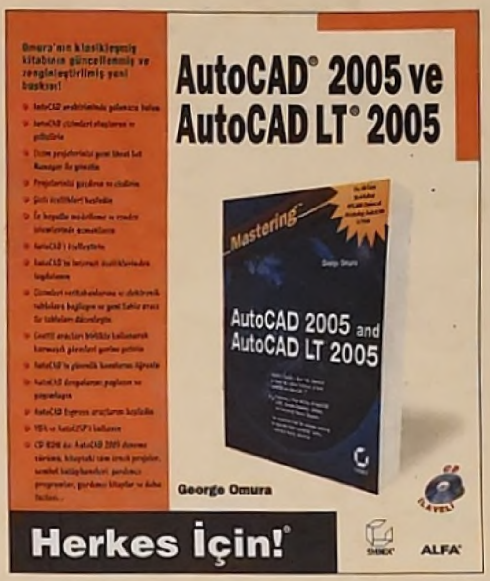
robotik 184
ROM 126
rotor 178
röle 20, 24, 58, 127
röle kontrol devresi 58
röle kontrolü 20, 24, 80
röleler 82
RS232 172
RS-232 el sıkışma sinyali 67
RS232 standardı 68
RTS 67, 69, 70
S kutbu 50
S sinyali 72
saat frekansı 67
saat sinyali 66, 128
sabit direnç 11
saklayıcı 133, 147
saniyedeki bit sayısı 67
sayaç 147
sayı sistemi 7
sayıcı 72
sayısal - analog dönüştürücü 43
sayısal-analog dönüşüm 42
sekizlik sayı sistemi 17
senkron iletim 66
seramik resonatör 128
SERIN komutu 174
seri bağlı dirençler 13
seri I/O 68
seri iletişim 66, 119
seri port 66, 104, 119
seri port ile servo kontrolü 179
serigraf 90
SEROUT komutu 172, 173, 174
servo 178
servo kontrolü 179
servo motor 178, 223
ses kartı 112
ses sinyali 66
sıcaklık algılama 104
sıcaklık sensörü 42, 104
silisyum diyot 22
sink akımı 128
Sony 196
source akımı 128
South kutbu 50
start biti 67
statik balans 196
statik IP adresi 81
stator 178
STATUS portu 4, 6, 15, 42
STATUS Portu ile veri girişi 15
STATUS saklayıcısı 133
step motor 28
stop biti 67
sunucu 81
sürücü devresi 28, 162
şarj 222
taban adresi 6, 15
tam adım 31
TCP/IP ile röle kontrolü 82
TCP/IP protokolü 80
telli direnç 11
terminal 118, 119, 172, 179
termistör 11
termometre 104
termometre devresi 104
ters çevirici 75
ters EMK 30, 54, 163
terslemek 15, 16
tolerans 13
toprak 4, 5, 28
toprak akımı 5
toprak hattı 5
toprak pini 4
tork 178
transistör 21, 45, 127
transistörlü adım motor sürücü devresi 29
transmitter 72
TRIS 133
tristör 220
triyak 220
TTL entegresi 74, 77
tuş butonlar 14
tuz ruhu 93
UART tümdevresi 67
UBound() fonksiyonu 130
UDN2993B 51
UDN2003 entegresi 28, 59, 162
UltiBoard 91
Unix 98
uzaktan kumanda 58
veri 4
veri akış kontrolü 70
veri aktarım hızı 67
veri aktarımı 4
veri çıkışı 4, 7
veri girişi 14
veri iletişimi 6
veri kontrolü 34
veri saklayıcısı 34, 35, 37, 151
veri sıkıştırma 70
verici 72
veriGonder() fonksiyonu 38
vi programı 98
Visual Basic 7, 8
vMotor 28, 29, 162, 163
voltaj 4, 5
voltaj limitleyici 105
voltaj regülatörü 204

W saklayıcısı 133, 142
Web sunucusu 81
WINIPCFG 82
Winsock 82
Xmodem 69
Xoff 70
Xon 70
XOR kapısı 75
XOR operatörü 16
yarı iletken 21
yarım adım 31
yazılım akış kontrolü 70
yönlendirici 133
YÜKSEK 7
yükselteç 198
yükseltici devre 50
zamanlayıcı 72
zener diyot 22
Zmodem 69

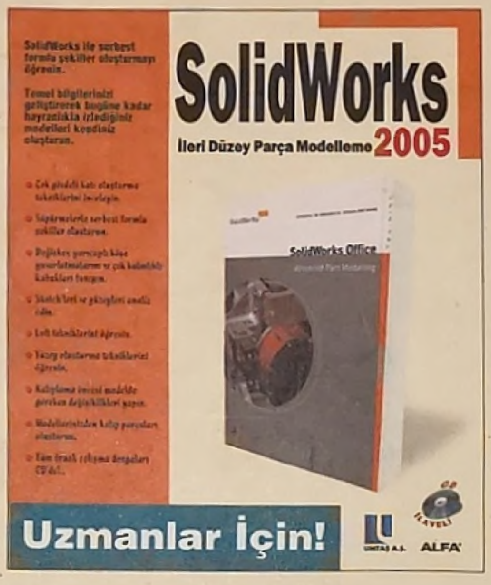
TÜRKİYE BİLGİSAYARI BU KİTAPLARLA ÖĞRENİYOR!

Bilgiye Giden Yol Herkese Açıktır!









Tüm kitaplarımız internette: www.alfakitap.com

ALFA Basım Yayım Dağıtım Ltd.
Ticarethane Sokak No: 53
34410 Cağaloğlu-İstanbul
Tel : +90 (212) 511 53 03
+90 (212) 513 87 51
Fax : +90 (212) 519 33 00
www.alfakitap.com
e-mail: info@alfakitap.com

ISBN 978-975-297-481-4

9 789752 974814

TL 18.00